4. THE PROPOSED FEATURE DETECTORS – *HORB* AND *ACORB*

This chapter, first, introduces the ORB feature detector. It also gives a brief introduction about the basic implementation of the ORB feature detector which will be used as a benchmark for the result analysis and comparison throughout this work. The second and third sections give an insight into the proposed feature detectors, the *HORB* and the *ACORB* respectively with detailed result analysis of both the approaches.

4.1 THE ORB (ORIENTED FAST ROTATED BRIEF) FEATURE DETECTOR

4.1.1 Introduction to the ORB

The digital image is a representation of the real world images into digits, 0s, and 1s. In earlier days, the images were stored in terms of only zeros and ones for monochrome systems as a matrix. Today, for millions of colors available and to represent the pixels in the images with specific color intensities, float-valued matrices are being used for more precision of colors. Following image shows how digital images are represented in the computers.

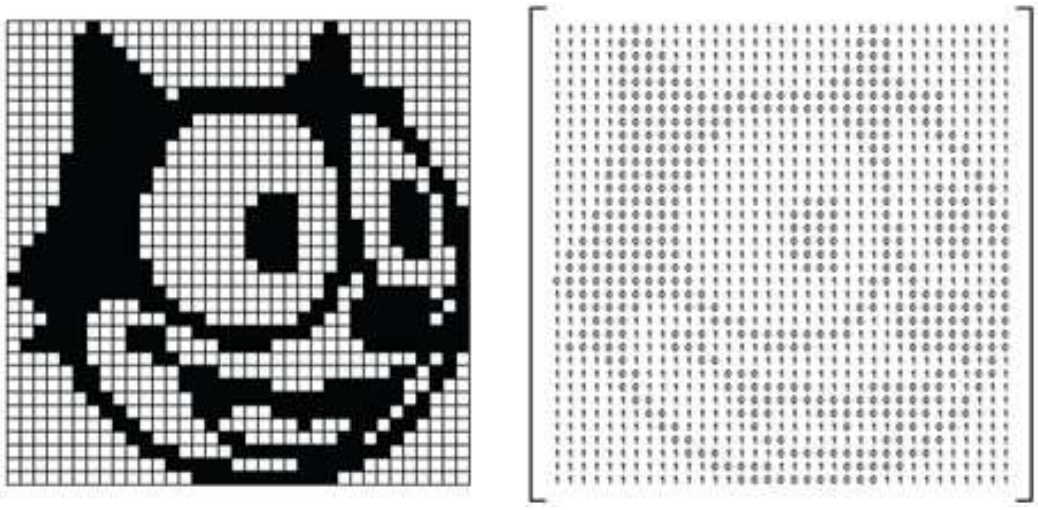

Figure 4.1. Squares showing pixels in image and corresponding matrix of 0s and 1s

Feature detection in the image is the most crucial stage in image identification in computer vision, which helps computer recognizing the image. In the context of images, a feature is an interesting part of the image which may be used as a distinguishing characteristic of the image. Feature detection is the starting point from where image classification starts. A feature detector is an algorithm that chooses points from an image based on some criterion. However, a feature descriptor is a vector of values, which somehow describes the image patch around an interest point. Together an interest point and its descriptor are referred to as a feature. These features are used for many computer vision tasks, like image registration, 3D reconstruction, object detection, and object recognition. As stated in literature study that there are many feature detectors like SIFT, SURF, KAZE, AKAZE, ORB, etc. available. The feature detector that has been used in this work is the ORB [61]. The ORB is a binary descriptor. A binary descriptor is composed out of three parts:

i. A sampling pattern: It decides where to sample the points in the region around the descriptor.

ii. Orientation compensation: It is a mechanism to measure the orientation of the keypoints and rotate it to compensate for rotational changes.

iii. Sampling pairs: The pairs to compare when building the final descriptor.

ORB is a combination of FAST detector and BRIEF descriptor. It is similar to BRIEF except,

i. The ORB uses an orientation compensation mechanism, making it rotation invariant.

ii. The ORB learns the optimal sampling pairs, whereas BRIEF uses randomly chosen sampling pairs.

For the measure of orientation, the ORB uses the intensity centroid. To calculate the centroid, the moment of a patch is calculated. Following formula gives the moment of order $p+q$:

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

Where $x^p y^q$ basic function, $I(x,y)$ is the image intensity and P,q = 1, 2, 3… ∞.

In the two-dimensional case, based on the moment values, the centroid, or 'center of mass' is then given as:

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

The vector from the corner's center O to the centroid gives the orientation of the patch. The orientation of the patch is calculated using the following formula:

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

Where atan2 is a function that converts the moment's coordinates to polar. Once the orientation of the patch is calculated, the patch can be rotated to a canonical rotation and then the descriptor is computed obtaining some rotation invariance. The ORB tries to take sampling pairs which are uncorrelated so that each new pair will bring new information to the descriptor. Thereby, it maximizes the amount of information the descriptor carries. For a feature to be more discriminative, the high variance is required among the pairs. For this, samplings of pairs are obtained over keypoints in the standard datasets. This is done through a greedy evaluation of all the pairs in order of distance from mean until a specific number of desired pairs are obtained. This number of pairs is the size of the descriptor. The descriptor is built using intensity comparisons of the pairs. For each pair, if the first point has greater intensity than the second, then 1 is written else 0 is written to the corresponding bit of the descriptor. The following figure shows how the angle is calculated in the ORB.

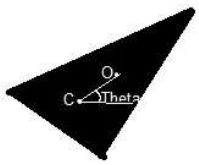

Figure 4.2. The ORB angle calculation

The limitation of the ORB is, it is faster in execution, but SIFT, SURF, KAZE, and AKAZE outperform it in terms of number of feature detection and matching. In [107], Oskar Andersson et al. experimentally proved that the ORB is giving least accuracy of 52.5%. In another attempt, Frazer Noble also showed that the ORB feature detector detects the least number of features [109]. In the similar direction, Ebrahim et al. in [119] also experimentally showed that the average accuracy of the ORB is 51.95%. So, to

improve the performance of the ORB, the following sections discuss the ORB based two hybrid feature detectors with their performance analysis.

4.1.2 The basic implementation of the ORB

To compare the performance of the proposed approaches, the basic ORB has been implemented, and these results will be used as a benchmark to compare the results of all the proposed approaches to check if there is any improvement in the performance of the ORB. Following table shows the performance of the basic ORB implementation.

Denomination	# of Train Images	# of Test Images		# of Correct Classes		Execution Time (Seconds)		Classification Accuracy (%)	
		F	P	F	P	F	P	F	P
5	2	130	80	88	9	220.54	85.99	67.692	11.25
10	4	226	80	145	7	297.41	99.88	64.159	8.75
20	2	220	80	156	11	268.45	94.4	70.909	13.75
50	4	228	80	158	12	279.75	92.78	69.298	15
100	2	268	80	178	6	312.68	88.42	66.418	7.5
200	2	139	52	101	13	154.82	99.76	72.662	25
500	2	160	78	117	4	161.78	72.64	73.125	5.128
500_old	2	165	80	112	2	161.98	92.31	67.879	2.5
1000	2	83	80	62	8	85.57	88.97	74.699	10
2000	2	200	80	152	11	241.82	78.33	76.000	13.75
Average time taken for an image & Accuracy						**1.2011**	**1.1603**	**69.764**	**10.779**
Overall Accuracy = 52.2209%						**Average Time = 1.1889 Seconds**			

Table 4.1 The Basic ORB Performance

The overall performance of the ORB comes out to be an average of 52.22% which is almost similar to the average performance the ORB as stated above. To calculate the average accuracy and time consumption, the following formulas are used throughout this work:

$$\textit{Average Accuracy for a category } A_c = \frac{\sum Correctly\ recognized\ images\ in\ a\ category}{\sum Test\ images\ in\ a\ category}$$

$$\textit{Overall Accuracy of all categories } A_t = \frac{\sum Correctly\ recognized\ images\ in\ all\ category}{\sum Test\ images\ in\ all\ category}$$

$$\textit{Average Time per image in a category } t_c = \frac{\sum Execution\ time\ in\ a\ category}{\sum Test\ images\ in\ a\ category}$$

$$\textit{Average Time per image of all categories } t_t = \frac{\sum Execution\ time\ in\ all\ category}{\sum Test\ images\ in\ all\ category}$$

The next two sections discuss the proposed feature detectors – the *HORB* and the *ACORB*.

4.2 THE *HORB* – A HISTOGRAM BASED ORB FEATURE DETECTOR

4.2.1 Introduction to the *HORB*

An image histogram is a graphical representation of pixel distribution in the image. The pixel distribution is a way to represent the color appearance in the RGB model and the saturation of a color in the HSV model. Histograms are invariant to translation, and they change slowly under different view angles, scales and in the presence of occlusions. Histogram matching is one of the oldest techniques in image matching. It is a transformation of the image into a histogram to match it with another histogram in reference for image identification. This is one of the simplest image classification methods. For histogram comparison, there are various methods like histogram intersection, Pearson correlation, Chi-Square distance, Bhattacharya distance available.

However, the image histogram represents the pixel weights or distribution, but it doesn't tell the pixel positions. Due to this, a histogram of, for example, a banana and a grass-field could have a closer match and may lead to wrong identification. For this reason, the only histogram-based image matching does not give guaranteed and accurate results. In an attempt to improve the accuracy in histogram-based image matching, L Tu et al. in [84] experimentally proved that the histogram based image feature matching could improve the results in image recognition. They showed it using SIFT and ASIFT with Histogram equalization. To address the same issue of the banana-grass field mismatches, a fusion of histogram intersection and the ORB feature detector, *HORB*, is proposed here. The *HORB* has been explained below:

Let us consider a histogram I for the input image and D for the dataset image. The histogram intersection function is defined as follow:

$$\bigcap_{j=1}^{n} \max(I, D_j)$$

Where n is the number of images in the dataset.

This max function returns the image object D_j for which histogram of D_j matches with significantly large numbers of pixels intersecting with a histogram of the input image I.

In the algorithm, first, the captured image is converted into a histogram and matched with the histogram of images in the dataset. In this stage, the histogram intersection technique is being used for finding the closest match. As stated above, the histogram of a banana and a grass-field could have more intersection value and may lead to wrong identification. To filter-out any such Banana-Grass field combination, the images with top k ($k<n$) histogram intersections are used, and corresponding feature subsets are sent for feature matching using the ORB. By getting the top k images with high intersection values, indirectly, the algorithm is reducing the number of feature subsets which will be matched against the input image I. Thus, actually, it is reducing the possibilities of mismatches by eliminating the majority of the images which are going to have fewer matches with the input image I. After getting top k images, the best-matched feature subset passing through a specific threshold value is selected from the top k images' feature set and returned as output label.

The complexity of the algorithm can be given in the form of Big-O notation as $\boldsymbol{O(I_i*D_i*n)+O(k*F_I*F_D)}$. Where, I_i is the i^{th} pixel of the histogram of the input image I and D_i is an i^{th} pixel of image D from a dataset of n images during histogram intersection. After, histogram intersection, for feature matching of remaining k $(k<n)$ feature subsets, F_I is the number of features of the input image I and F_D is a number of features of each of the feature subset. The overall algorithm is given below:

1. Initialize *totalNoOfImages (N)* in the dataset, the *featureSet* with distinguishable features for all images, *imageObject*, in the dataset.
2. Initialize *featureThreshold, similarityDistanceThreshold*
3. Generate histograms of all *N* images, *imageObject*, in dataset and histogram of *inputImage*
4. For each histogram of *imageObject*, perform histogram intersection with a histogram of *inputImage*
5. Find top *K* histogram intersection values and corresponding *imageObject* into *topKIntersects* and *topKImageObjects,* where $K < N$.
6. Create an ORB feature detector for *inputImage*
7. For each *imageObject* in *topKImageObjects* with the corresponding *featureSet*, Perform feature matching:

   If *imageObject* passes *featureThreshold* and *similarityDistanceThreshold* then,

   Add it to the *LabelList*.
8. Use the best-matched *imageObject* from *LabelList* as an output image.

The HORB algorithm

4.2.2 The Testing and Performance Analysis of the *HORB*

Table 4.2 gives an overall view of the performance of the *HORB*.

Denomination	# of Train Images	# of Test Images		# of Correct Classes		Execution Time (Seconds)		Classification Accuracy (%)	
		F	P	F	P	F	P	F	P
5	2	130	80	114	8	351.8	194.3	87.69	10
10	4	226	80	198	6	534.5	204.2	87.61	7.5
20	2	220	80	192	9	514.6	187.5	87.27	11.25
50	4	228	80	197	13	530.6	189.6	86.40	16.25
100	2	268	80	231	6	608.6	183.8	86.19	7.5
200	2	139	52	126	16	322.1	117.9	90.65	30.77
500	2	160	78	143	3	366.9	170.1	89.38	3.846
500_old	2	165	80	148	3	375.5	178.9	89.70	3.75
1000	2	83	80	74	9	187.4	191.0	89.16	11.25
2000	2	200	80	182	7	452.2	183.8	91.00	8.75
Average time taken for an image & Accuracy						**2.3332**	**2.3393**	**88.23**	**10.389**
Overall Accuracy = 65.083%						**Average Time = 2.335 Seconds**			

Table 4.2 *The HORB* Performance

Here, the number of correctly recognized images in case of the ORB is 1352 out of 2589 whereas the same number for the *HORB* is 1685 leading to an overall accuracy of 65.083% as compared to the ORB's 52.220%. Here, the accuracy percentage for fully visible images is 88.235% which is far better than 69.763% of ORB. But, for partially visible images, the accuracy is not improved which is around 10.389% only! The total number of correctly classified images and its accuracy are shown graphically in the following figure 4.3 and figure 4.4 respectively.

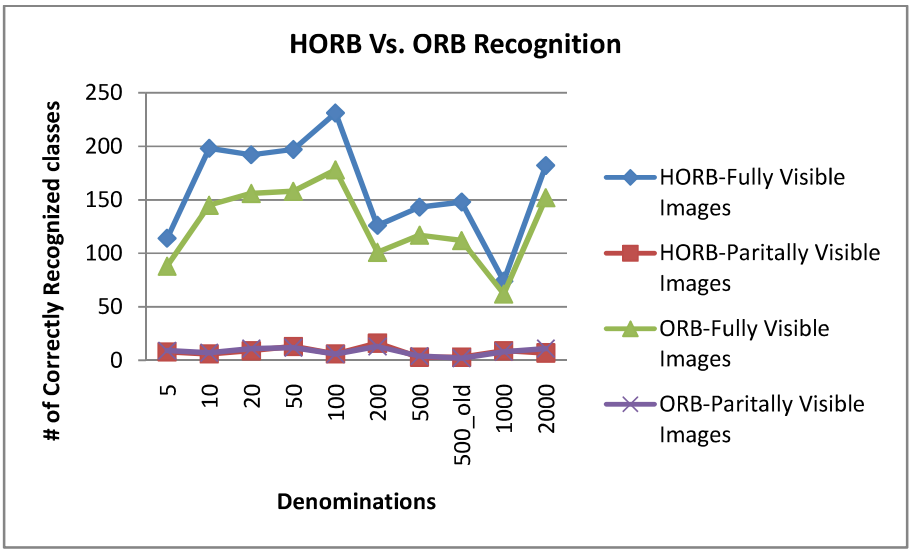

Figure 4.3. Number of correctly identified images using the *HORB* and the ORB

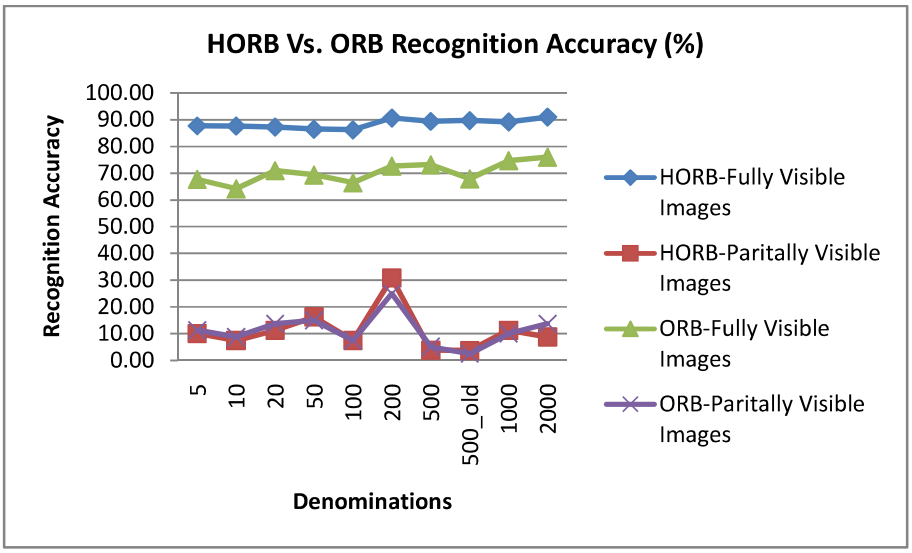

Figure 4.4. Recognition accuracy of the *HORB* and the ORB

In terms of time, the *HORB* takes an average 2.333 and 2.339 seconds per image, for recognition of fully and partially visible images, which are almost, double than the ORB consuming 1.201 and 1.160 seconds per image. But, for improvement in recognition accuracy, this trade-off is necessary. For a set of 2589 images of all denominations, the time consumption for each denomination and average time taken per image are shown in figure 4.5 and 4.6.

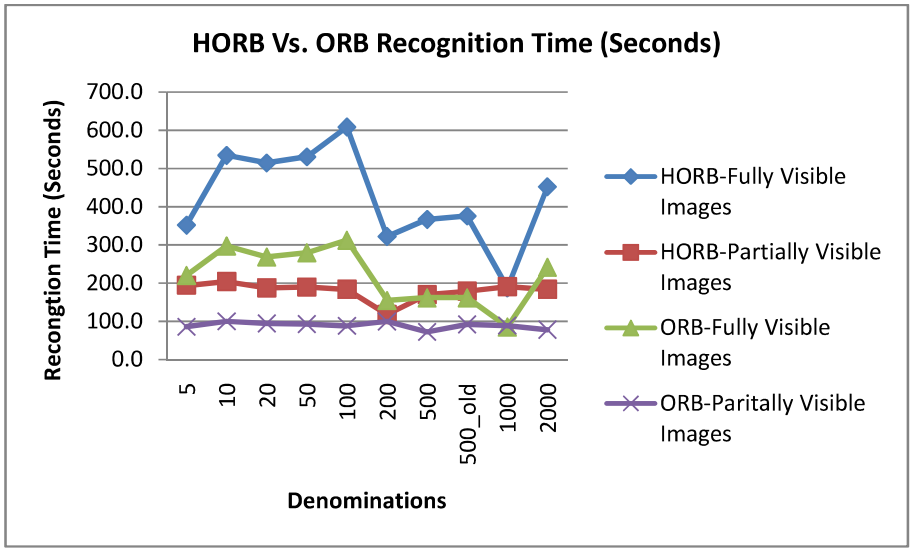

Figure 4.5. Recognition Time taken by the *HORB* and the ORB

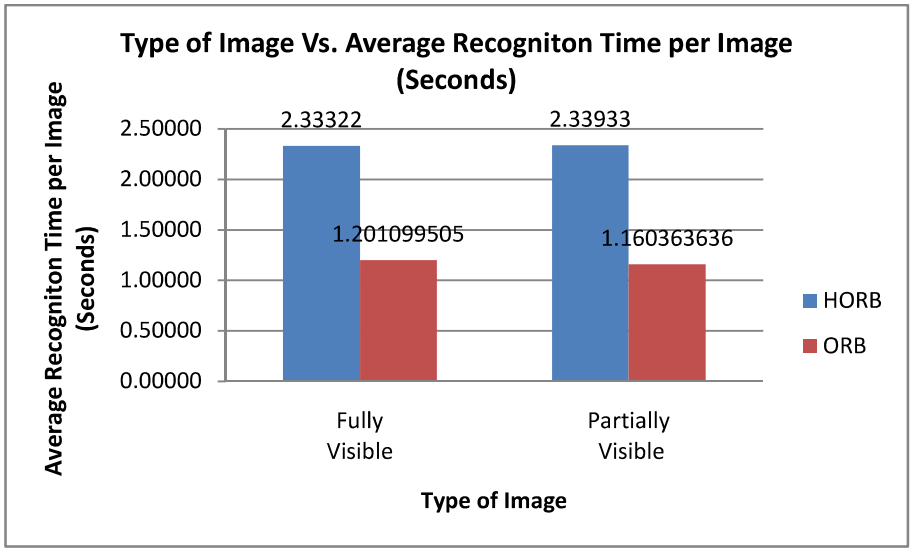

Figure 4.6. Average time taken per image by the *HORB* and the ORB

Figure 4.7 shows an overall improvement of around 12.862% in the ORB using the proposed approach. It proves that the proposed approach, *HORB*, has the edge over the ORB.

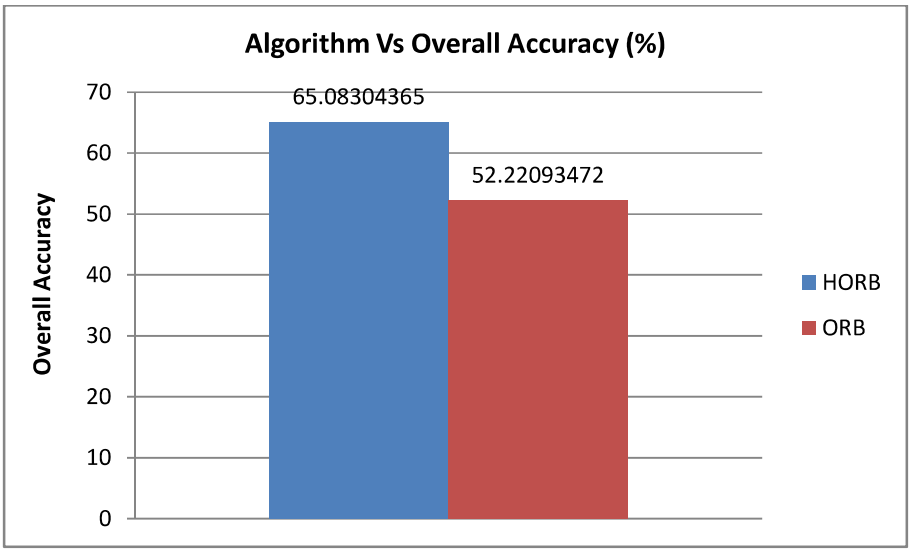

Figure 4.7. The overall performance of the *HORB* and the ORB

4.3 THE *ACORB* – AN ANT COLONY OPTIMIZATION BASED ORB FEATURE DETECTOR

4.3.1 Introduction to the *ACORB*

The ant colony algorithm is a probabilistic algorithm to find the optimal path which depends on the behavior of ants searching for food. Initially, the ants roam around randomly. Here, the ant is an agent that constructs the solution for the given problem. The probability of k^{th} ant, of total m ants, moving from i^{th} place to j^{th} place is given by:

$$p(i,j) = \frac{\tau_{ij} * \eta_{ij}}{\sum \tau_{ij} * \eta_{ij}}$$

Where τ_{ij} is the amount of pheromone on edge *ij*

η_{ij} is the amount of heuristic value associated with edge *ij*

When the ant gets the source of food, it goes back to her colony leaving some identification, called pheromones, to remember the path of the food source. When the other ants come to the point while roaming, they follow that path due to the pheromones.

These pheromones are assigned a specific probability value with some formula and updated by the ants. As more and more ants started following that path, the probability value becomes larger indicating that the path leads to a food source. The following formula used to update pheromone value:

$$\tau_{ij} = (1 - \rho) * \tau_{ij} + \Delta\tau_{ij}$$

Where τ_{ij} is the amount of pheromone on edge *ij*

ρ is the rate of pheromone evaporation

$\Delta\tau_{ij}$ is the amount of pheromone deposited and is given by

$$\Delta\tau_{ij} = \begin{cases} \frac{1}{L_k} \\ 0 \end{cases}$$

L_k is the cost of the k^{th} ant's tour (normally the distance between node *i* and *j*)

But the algorithm does not stop here. Due to the pheromones dropping by the ants whenever they find the food and come back to the colony, the shorter paths are more likely to be stronger as the different ants may travel through different paths. This is how the optimal path is found. In the meantime, some ants would still be roaming in the hunt of nearest food sources. This approach has many applications in finding the optimal solution. Traveling salesman problem is one of the approaches where ACO is being used. The following figure shows how ACO works.

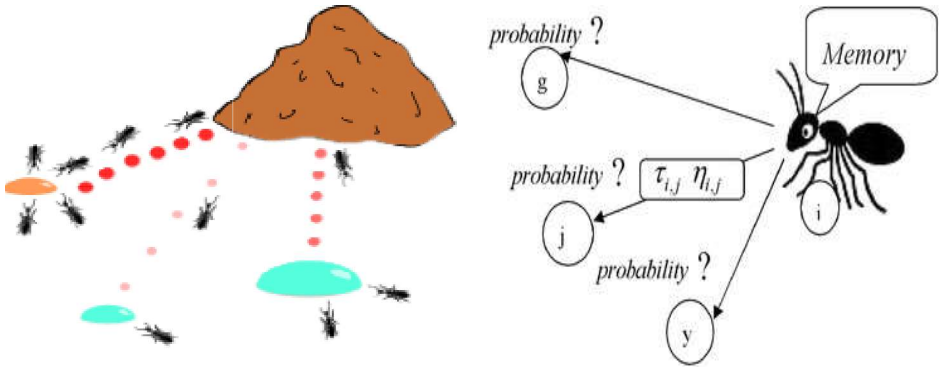

Figure 4.8. Ant Colony, red dots show the pheromones dropped by ants

In his work, Kwang-Kyu Seo [66] used the ACO for content-based image retrieval using HSV and RGB color model with Textures. He showed in his work that ant colony optimization in case of image feature extraction converges quickly with a specific set of features. To take advantage of this, an attempt has been made to use the ACO for feature detection using the ORB, wherein along with colors and textures, specific feature vectors of distinguishable features have also been used. This approach has been named as *ACORB*. The *ACORB* is described below:

1. Initialize the *numberOfAnts*, *pheromone*, *maximumIterations*, *distanceThreshold,* and *foodQty*
2. Assign any random *Ant* to the *foodLocation*. Evaluation criteria are the least distance between the *Ant* and the *foodLocation,* i.e., *distanceThreshold* and the *foodQty*.
3. If the *Ant* does not get food at the *foodLocation* in specific *foodQty* in *maximumIterations* then,

   Select another *Ant* to build the solution.

   else

   Evaluate the *Ant*'s food selection based on *distanceThreshold*.
4. If food selection passes the *distanceThreshold*, add it to *AntList* and update the *pheromone* value
5. Select the best *Ant* from AntList for labeling.

The ACORB algorithm

The ACO has been used to explore the feature subsets of a given set. If the selected feature subset is suitable or not is decided using the heuristic function. With the collection of feature subsets found, the best is selected as output label. For this algorithm, the distance between the ant and the food location L is taken as a parameter for τ calculation, and food quantity is taken as heuristic η. The probability of ant reaching to a source of food, pheromone updates and the amount of pheromone deposited are calculated using the same formula stated above. According to the Big-O notations, the complexity of this algorithm will be $\boldsymbol{O(n*F_I*F_D*m)}$ where *n* is the maximum number of

iterations, F_I is the number of features of the input image I, F_D is a number of features of each of the feature subset ant and m are the number of ants.

4.3.2 The Testing and Performance of the *ACORB*

Table 4.3 gives an overall performance of the *ACORB*.

Denomination	# of Train Images	# of Test Images		# of Correct Classes		Execution Time (Seconds)		Classification Accuracy (%)	
		F	P	F	P	F	P	F	P
5	2	130	80	115	9	348.6	162.9	88.46	11.25
10	4	226	80	197	6	522.4	198.6	87.17	7.5
20	2	220	80	194	8	509.5	154.3	88.18	10
50	4	228	80	199	12	531.9	155.4	87.28	15
100	2	268	80	234	6	604.8	149.8	87.31	7.5
200	2	139	52	127	15	614.1	87.7	91.37	28.85
500	2	160	78	146	3	346.2	138.6	91.25	3.846
500_old	2	165	80	148	3	349.9	132.8	89.70	3.75
1000	2	83	80	75	9	178.0	148.5	90.36	11.25
2000	2	200	80	182	7	439.4	138.0	91.00	8.75
Average time taken for an image & Accuracy						**2.4435**	**1.9046**	**88.89**	**10.12**
Overall Accuracy = 65.4692%						**Average Time = 2.2832 Seconds**			

Table 4.3 The *ACORB* Performance

Referring to the benchmark table 4.1, the number of correctly recognized images in case of the ORB is 1352 out of 2589, whereas the same number for the *ACORB* is 1695 (an increase of 10 images than the *HORB*'s 1685) leading to an overall accuracy of 65.469% as compared to the ORB's 52.220%. Here, the accuracy percentage for fully visible images is 88.895% which is also far better than 69.763% of the ORB and 0.659% more than the *HORB* for fully visible images. But here also, for partially visible images, the accuracy does not seem to be improved. The total number of correctly classified images and its accuracy are shown in figure 4.9 and 4.10 respectively.

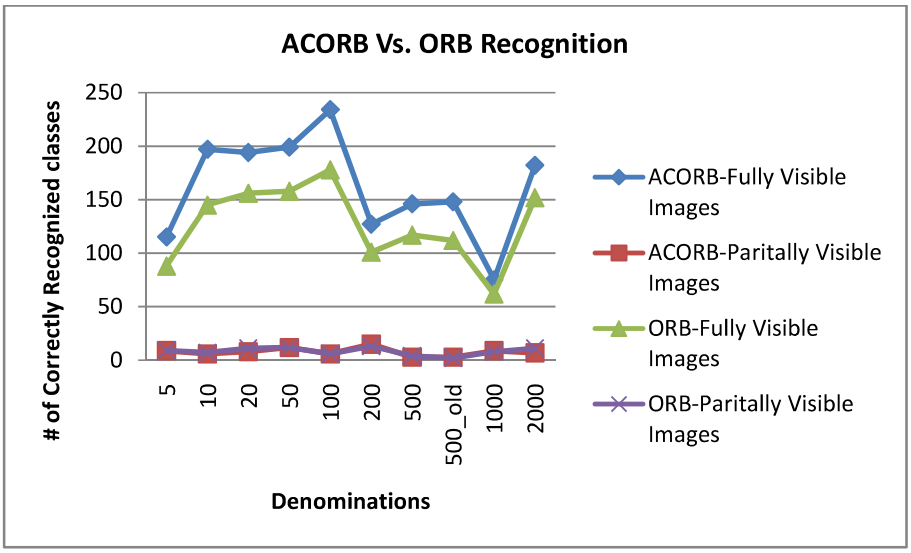

Figure 4.9. Number of correctly identified images using the *ACORB* and the ORB

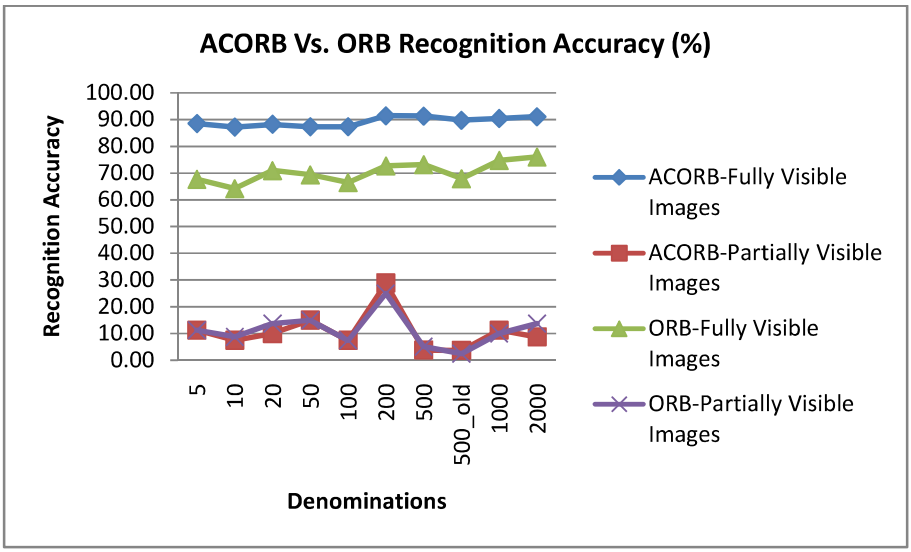

Figure 4.10. Recognition accuracy of the *ACORB* and the ORB

In terms of time, the *ACORB* takes an average 2.443 and 1.904 seconds per image, for recognition of fully and partially visible images, higher than the ORB, consuming 1.201 and 1.160 seconds per image. The time consumption is almost as nearby as the *HORB*. The average time taken by the *ACORB* is 2.283 seconds, which is a bit less than *HORB*'s 2.335 seconds. These results prove that the heuristic-based approaches like ACO can be applied and works fine for image feature detection. The total time consumption for each denomination and average time taken per image are shown in figure 4.11 and 4.12.

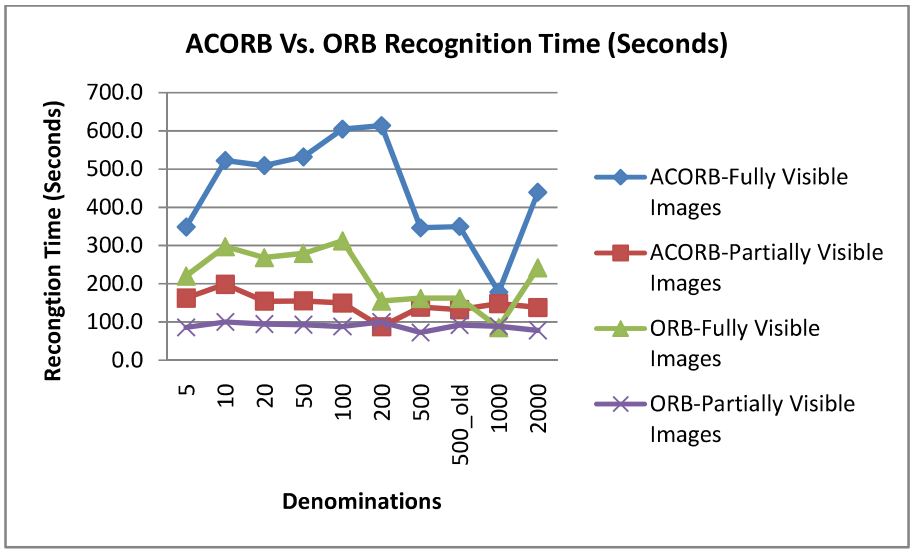

Figure 4.11. Recognition Time taken by the *ACORB* and the ORB

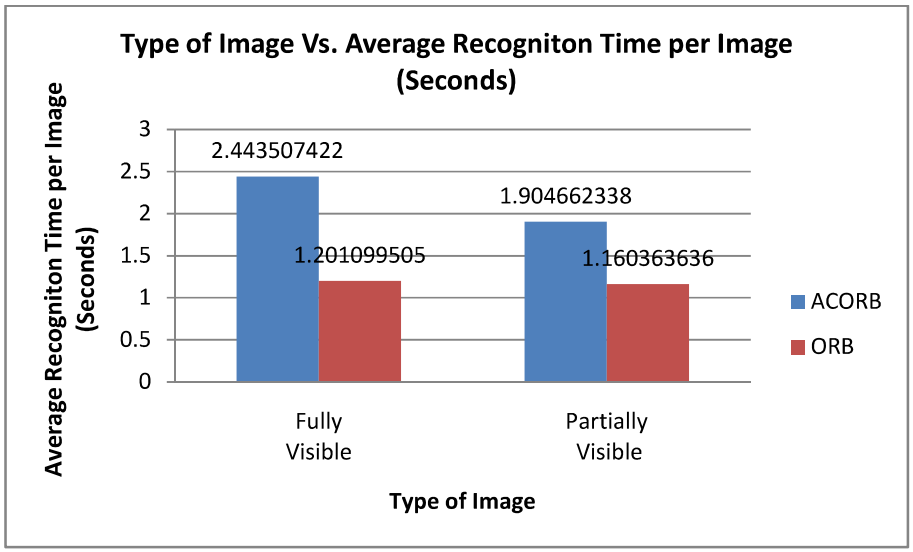

Figure 4.12. Average time taken per image by the *ACORB* and the ORB

The figure 4.13 shows an overall improvement of around 13.248% in the ORB through the heuristic based approach *ACORB*. It is also 0.386% higher than the *HORB*.

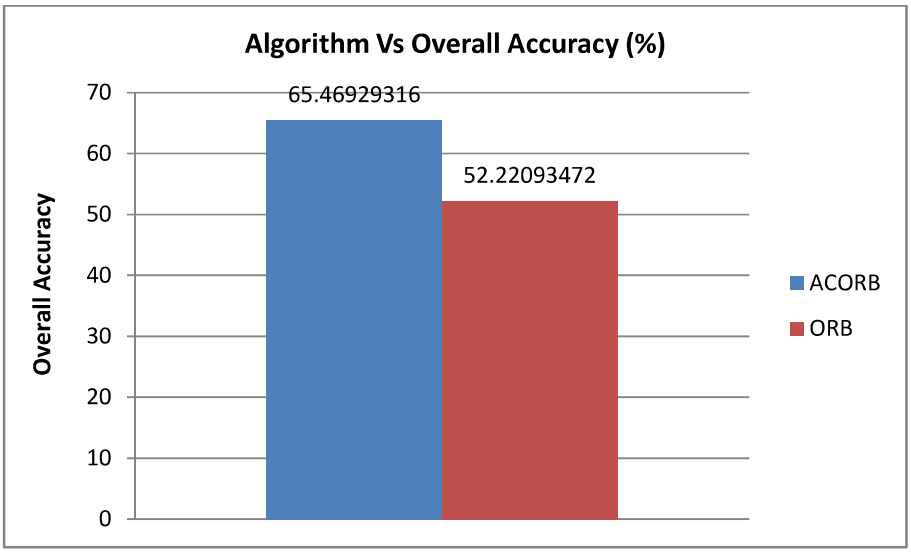

Figure 4.13. The Overall performance of the *ACORB* and the ORB

The following figures summarize the performance comparison of the ORB, the *HORB,* and the *ACORB* in terms of accuracy and time taken for recognition both.

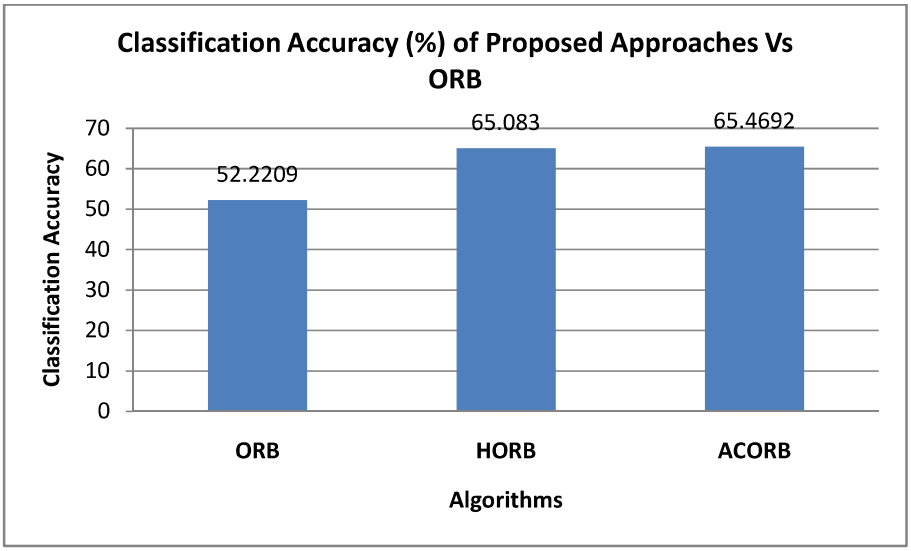

Figure 4.14. The Accuracy performance of the ORB, the *HORB,* and the *ACORB*

Overall, it can be observed that there is no significant increase in the accuracy in, both, *HORB* and *ACORB*. This indicates that only feature based identification using ORB does not improve the accuracy in image identification.

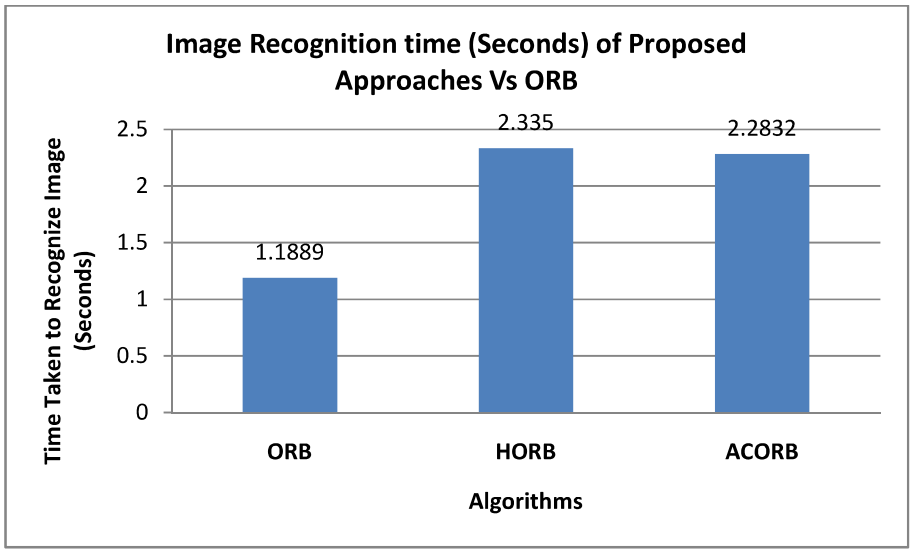

Figure 4.15. The time performance of the ORB, the *HORB* and the *ACORB*

SUMMARY

This chapter introduced the ORB feature detector in the beginning. Then, it discussed the two novel feature detectors based on Histogram and ACO in fusion with ORB, the *HORB,* and the *ACORB* respectively with their performance analysis. Finally, it summarized the performance of the proposed approaches with reference to the ORB in terms of time and accuracy both. The next chapter discusses two proposed classifiers which are based on the *HORB* and the *ACORB* with a bag of visual words.