ORB-Based Multiple Fixed Resolution Approach for On-Board Visual Recognition

Daniel Phillips[1(✉)], Akash Pooransingh[1], and Sinem Guven[2]

[1] The University of the West Indies, St. Augustine Campus, St. Augustine, Trinidad and Tobago
{daniel.phillips,apooransingh}@my.uwi.edu
[2] IBM Research, TJ Watson Research Center, Yorktown Heights, New York, USA
sguven@us.ibm.com

Abstract. Maintenance and troubleshooting of hardware on a large scale pose a challenge in deploying expert technicians at multiple sites. Augmented Reality-based technology support equips the technicians with the skills they need to solve hardware problems even without expert level training, thereby reducing training time and cost to the vendor. Enabling Augmented Reality for technology support requires the ability to visually recognize the hardware in real time using mobile devices, and train the underlying algorithms at scale. This paper proposes a novel approach to address these issues. Our ORB-based fixed multi-resolution recognition algorithm achieves over 95% accuracy at a resolution scale of 0.2, and an approximately 60% faster recognition time than the next best comparable method. We also demonstrate the real-world applicability of our algorithm through an implementation of an Augmented Reality application.

Keywords: Oriented FAST rotated BRIEF (ORB) · Scale Invariant Feature Transform (SIFT) · Speeded Up Robust Features (SURF)

1 Introduction

Technical support services for the maintenance of hardware (servers, computers, printers, mobile devices, etc.) is becoming a core service offered by companies responsible for system maintenance of hundreds of components around the globe. They employ tens of thousands of support specialist that maintain equipment made by various vendors. These technicians must be fully trained to troubleshoot various components, which requires substantial training time and incurs cost to the vendor. Conventionally, paper-based manuals or videos have been used to receive technical guidance during repair sessions. As hardware becomes more and more complex, ways of reducing the cost and time of a repair has become a topic of interest. Augmented Reality (AR), which overlays graphics or media

D. Wang and L.-J. Zhang (Eds.): AIMS 2019, LNCS 11516, pp. 54–71, 2019.
https://doi.org/10.1007/978-3-030-23367-9_5

on top of what we see in the real world, is increasingly becoming popular as a new way to visually guide the technicians through the repair process. Object detection and recognition form the fundamental processes toward an AR system, as shown in Fig. 1. Feature point detection based methods are used for both object recognition and object localization. These methods are used for image classification [1], image matching [2], localization [3] and object detection [4] with success. Three common feature point descriptors are the Oriented FAST rotated BRIEF (ORB), Speeded Up Robust Features (SURF) and Scale Invariant Feature Transform (SIFT). These feature point methods have been combined with the methods such as the Bag of Features [5] to yield an accuracy of up to 90% [6]. Feature point methods have been successful, but they are not always the most efficient or the fastest method. Alternative methods, such as Histogram Intersection and Local Binary Patterns, have also yielded success in various applications. Using Local Binary Patters (LBP) combined with Support Vector Machines, accuracy of up to 97% have been achieved [7]. The Histogram Intersection methods have the added advantage of being able to detect objects based on color, which can be useful in various applications. This method can also be combined with a support vector machine to improve its classification of images. However, the accuracy of this method is typically low as reported in [9] of about 58%.

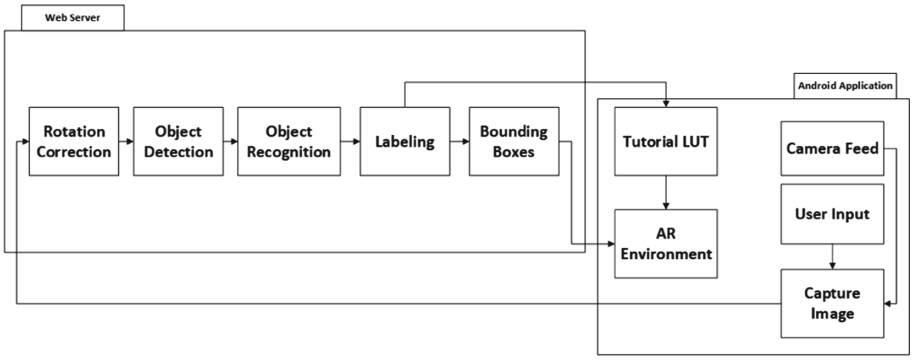

Fig. 1. A typical augmented reality system

2 Feature Point Detection Techniques

Feature Point detection methods contribute significantly toward object recognition applications. In [4], object recognition implementation, which utilized SURF feature points, was able to achieve high recognition accuracy. This method was successful even though background noise was introduced into the images containing the objects. Piccinini [3], also used feature point detection techniques in applications where occlusion occurred. During these applications, the SIFT technique also yielded a high recognition accuracy. Even though these feature point

techniques yield a high accuracy, they are still computationally intensive causing them to perform slowly. The ORB technique is much faster and is able to yield a result in a fraction of the time. It is also able to yield an accurate result especially when combined with other methods as shown in [14]. From these implementations, it can be seen that feature point detection has performed well in various applications despite being exposed to varying environments. This paper proposes the use of the ORB feature point technique as part of our approach and compare it to the traditional feature point methods: SIFT and SURF.

2.1 ORB

ORB has been recognized for its speed as it is easily one of the fastest and light weight feature descriptors currently in existence [7]. This detector is both scale invariant and rotation invariant making it very useful for a wide array of applications. ORB was built by merging in the Features from Accelerated Segment Test (FAST) key point detector with the Binary Robust Independent Elementary Features (BREIF) descriptor. Since neither FAST nor BRIEF are inherently rotation invariant, an intensity centroid was assessed to measure the corner orientation [8].

2.2 SIFT

The Scale Invariant Feature Transform (SIFT) was first presented in [15]. This has been recognized to be one of the more robust feature point detector and descriptor. This has been proven to be robust against the affine transform, intensity changes and even certain view point changes. Just like ORB, this detector is both rotation invariant and scale invariant. SIFT works by first computing a Difference of Gaussians to get a similar effect to the Laplacian of Gaussians in a less computational intensive way [16].

2.3 SURF

The SIFT descriptor can be considered a successful descriptor as it has been used in a wide array of applications. One major problem that arises during its implementation is that, it is computationally intensive and not suitable for low powered devices. Therefore, more development was done to find a less computationally intensive, but still robust detector and descriptor [17]. In [2], produced the Speeded Up Robust Features (SURF) descriptor and detector. This was developed to be both rotation and scale invariant, just as SIFT was. Instead of approximating the Laplacian of Gaussians (LoG) using a Difference of Gaussians (DoG) as was done with the SIFT detector, SURF uses a box filter. The box filter was chosen since the integral of the image can be used, which makes the algorithm much more efficient. Also, the approximation can be more easily calculated and done in parallel, reducing the computational time requirements.

3 Non-Feature Point Techniques

Non-feature point methods have also been very successful in the areas of object recognition. Unlike feature point methods, they seek to recognize objects based on physical features, such as color and texture. These methods in some cases are less affected by rotation, scaling, background noise and even occlusion [10]. These methods have been successfully tested in areas, such as facial recognition, yielding good results [11].

3.1 Local Binary Patterns

Introduced in 1996, [12], Local Binary Patterns (LBP) have been used to extract features from images using thresholding. This method has been used to tackle difficult recognition problems, such as facial recognition with success [13]. This is accomplished by using a 3×3 pixel block of an image and thresholding the outer pixels of an image block using the blocks center pixel value multiplied by a power of 2. The resulting values are then summed together to produce the value or label for that center pixel [18]. This can be defined as follows:

$$LBP(x_c, y_c) = \sum_{n=0}^{7} 2^n g(I_n - I(x_c, y_c))$$

where (x_c, y_c) is the LBP centre pixel value, I_n represents the neighbor pixel value. $I(x_c, y_c)$ represents the centre pixel value and n represents the index of the neighbor pixels. The function is designed to function as follows:

$$g(x) = \begin{cases} 0, x < 0 \\ 1, x \geq 0 \end{cases}$$

Therefore, during the thresholding, if the current block pixel value is larger than the center pixel value, it is assigned a value of 1, if not, it is assigned a value of 0. An 8-bit number is used giving the center pixels 256 possible values [12]. This method was improved in [3] to make the method more customizable improving its performance in a wider array of applications. These improvements removed the restrictions on size of the sampling operator and number of sampling points used. Also, it allowed for variable block size [18]. Linear Binary patterns have also been paired with Support Vector Machines for improved image classification. The LBP-SVM was tested in [7], with the Indian pines data set, Pavia Data set and the Salinas Data Set, which contained hyperspectral imagery taken by satellites and airborne vehicles. The percentage accuracy achieved from these datasets were between 89.47% to 97.53% for this method.

3.2 Histogram Intersection

The Histogram Intersection method was introduced by Swain and Ballard [10]. This method was developed as a technique for comparing the similarity between

two histograms. This method has been useful in many recognition tasks such as identifying species of trees in Malaysia [19]. The match between the histograms are computed and the result yields a number. For this method, the greater the number the better the match. If the histograms are normalized, the result provided would be a number between 0 and 1. This method is robust against many changes such as resolution, occlusion, view point changes, background changes and even rotational changes. This makes this method useful for the recognition of the same objects in different environments. If the histograms are normalized, this improves the robustness of this method against the distraction of background pixels in the image. The intersection between the two histograms is described as follows:

$$\cap(I, M) = \sum^{n} min(I, M)$$

The histograms are normalized by using the number of pixels in the model histogram. The normalized histogram can be defined as follows:

$$H(I, M) = \frac{\sum_{j=1}^{n} min(I_i, M_j)}{\sum_{j=1}^{n} M_j}$$

Histogram intersection is not the only method of histogram comparison, but was found to be one of the most robust methods [19]. Histogram intersection can be used with many different color schemes. The most common schemes are the HSV (Hue, Saturation, Value) scheme and the RGB (Red, Green, Blue) color scheme. RGB has been used in many applications with a good result despite being very sensitive to light changes and noise. It is also composed of components that are highly correlated with each other and not intuitive to human perception. Also the chrominance and luminance values cannot be separated easily making images difficult to process in this scheme [20]. Using the HSV scheme can, however, be used to remove the luminance information from the image so that this method would be invariant to light changes improving the accuracy of the method.

4 Classification Techniques

4.1 Bag of Features

The bag of Features model was introduced in 2004 as a way of classifying images [5]. This method can be broken down into four main parts. This includes detection and description, the assignment of descriptors to predetermined clusters, constructing a bag of key points and the application of a multi-class classifier. The first step, detection and description, can be accomplished with a feature point detector and descriptor. The more invariant and robust the descriptor, the more suited it would be for this process. Since it would be used to train an image classifier, the descriptor should be also invariant to illumination and the affine transform. For the next stage, the construction of the visual vocabulary

is accomplished. The visual vocabulary is used to relate descriptors from a test image back to previously collected descriptors in the training images. To improve the efficiency of the comparison of the training descriptors to the test descriptors, clustering is used. K means clustering is used to accomplish this in an efficient manner. The categorization is then completed on the clusters produced. During the supervised training of the model, the labels of the images are sent to the categorizer to develop a statistical method for distinguishing categories. This categorization step can be completed using a support vector machine. Since this is a multiclass classification problem, the support vector machine is trained using a one-against-all approach. For this stage, a Naive Bayes method can also be used. During testing however, the Support Vector Machine method outperformed the Naive Bayes method [21].

5 Multiple Fixed Resolution Based On-Board Component Recognition Algorithm

This paper proposes a simple multiple fixed resolution method for recognition. This approach is a two staged process as shown in Fig. 2.

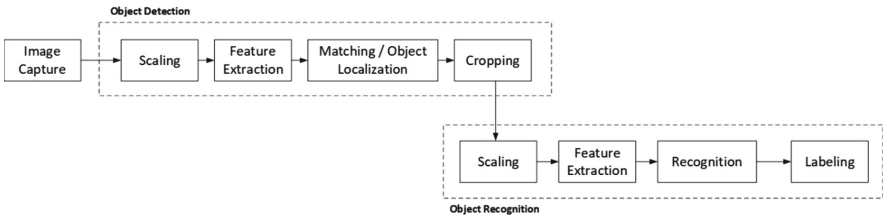

Fig. 2. Recognition algorithm steps

5.1 Algorithm

Detection

1. Convert image to greyscale
2. Scale image
3. Extract SIFT feature points
4. Complete brute force matching

Recognition

1. Convert to grayscale
2. Scale image
3. Compute local descriptors using ORB features
4. Load bag of features model
5. Image prediction using bag of features model
6. Label component

5.2 Image Capture

The images were captured from the mobile device using an appropriate resolution. During the implementation of this algorithm, we used the Wikitude Camera API extension along with a camera plug in to capture images. This allowed for direct access to the camera feed at the resolution specified. The captured Image was then stored to internal storage of the device in the PNG format since it was lossless compression scheme. Next, this image was sent to the web server to process the image. During this stage, the image was also converted to gray-scale using the following equation:

$$Y \leftarrow 0.299R + 0.587G + 0.114B$$

5.3 Object Detection

Scaling. The images are scaled on two different occasions. The first scaling occurs after image capture to allow the image to be at the optimal resolution, where the object detection would operate at its peak accuracy. Each image at this stage was scaled equally in the x and y plane at a fixed scale. (The second scaling occurs after cropping of the localized components in the image, as later described in the Recognition section.)

Feature Extraction. The feature extraction during the object detection stage was accomplished using SIFT descriptors. These were extracted from the training images of the various components. These training images included crops of each component of interest that were present and absent on the board. These feature points were extracted and stored. At this stage, feature points were also extracted from the full motherboard image, which was the testing image and these feature points were also stored for use in the localization step.

Matching - Object Localization. At this stage, the previously extracted feature points of the components were matched with the feature points in the testing image. For this stage, a knn Brute-Force Matcher was used. The matches were calculated using the L^2 distance. This distance can be expressed using the following equation.

$$d(f_a, f_b) \sum (f_a - f_b)^2$$

where f_a and f_b represent the feature points descriptors from the query and test image respectively. Once the feature matching was completed, outlying points were detected and removed using the RANSAC algorithm. This step was done to remove the false matches allowing for a more accurate localization of the components. Next, the in-lying points were used to calculate a homography matrix so that a bounding box can be calculated and placed over the components found.

Given a set of feature point matches that contains both true positive matches and false positive matches, it would be difficult to determine which matches are the true positives. Using the distance ratio proposed by Lowe [21] alone may cause many points that are true positives to be also removed. To solve this problem the Random Sample Consensus (RANSAC) method can be used. RANSAC is used to first estimate a global relation that fits the dataset using the hypothesize and verify framework. This is achieved by first sampling the subsets of data at random enough intervals so that a solution can be computed. This allows for the global relation to be found while simultaneously dividing the data into outlying and inlying points. Using the data that was sampled, a hypothesis for the model is determined and then all the points are compared against the hypothesis to verify it. To make this method more robust, this step is repeated until some pre-defined termination criteria is met based on a confidence criterion. This confidence criteria is usually satisfied when one of the subsets are outlier free [22]. This termination criteria can be calculated using the following equation:

$$N = \frac{log(1-p)}{log(1-(1-v)^m)}$$

where N is the number of iterations, p is the probability that at least of the sets have no outliers and m is the minimum number of points required.

Cropping. Cropping was accomplished using the bounding boxes produced in in the matching and localization stage. The locations, where the bounding boxes were placed, were cropped by extracting the pixels within the range of the bounding boxes.

5.4 Object Recognition

Scaling. The second scaling occurs after cropping of the localized components in the image, as per Fig. 2. This scaling was done so that the object recognition algorithm can operate at its maximum recognition accuracy. Each image at this stage was scaled equally in the x and y plane at a fixed scale.

Feature Extraction. The feature extraction at this stage was accomplished for the methods that required features to be extracted from the images. These included ORB, SIFT, SURF and the Local Binary Patterns (LBP) method. For the Histogram Intersection method, this step was ignored. During this stage, the images were converted to grey-scale before the features were extracted. These feature points once extracted were stored for use in the subsequent step.

Recognition. The recognition module was done with the Bag of Features Model. This model was trained using the orb features extracted from the training images. A support vector machine was also used to assist with making predictions

from the collection of feature points in the model. The support vector machine was used to find the best hyperplane to separate data usually of two classes. The best hyperplane is defined as the hyperplane with the largest margin between the two classes of data. The support vectors are the data points closest to the hyperplane. The general equation for a hyperplane can be written as follows:

$$f(x) = x^{'}\beta + b = 0$$

where $\beta \epsilon R^d$ and b is a real number. Support vector machine seek to find a solution for β and b that minimize $\| \beta \|$ such that for all the data point (x_j, y_j).

Labeling. Once the prediction was completed, the components were labeled using the output of the recognition step. These labels were used to determine which components were present so that the user can be prompted in the Augmented Reality application.

6 Results

The image processing algorithms and code were executed on a computer which ran Ubuntu 16.04 using Intel Core i7-2620M operating at 2.70 GHz with 16 GB of RAM operating at 1333 MHz. During testing, Open-CV 3.1.0 was used along with python 2.7.

6.1 Web Server Processing

Since the feature point matching and extraction steps were heavily CPU and memory intensive processes, they could not be accomplished on a mobile device. Therefore they were done on a remote server and results were sent back to the mobile device. The camera feed on the mobile Android device was used to acquire the images of the scene and then the images were scaled and sent to the sever for processing. Upon the completion of processing, the co-ordinates and labels were sent back to the mobile device for plotting and displaying in the AR application. This allowed the object recognition to also run at faster speeds as it did not have to share resources with the AR application.

6.2 Augmented Reality Application

The AR application was designed to provide visual guidance to technicians during repair sessions. It is implemented using the Wikitude Android Native API. After the bounding box location was calculated, the relevant bounding boxes were placed in the AR scene to be viewed by the user. The boxes were tracked with 6DOF and anchored to the recognized components in the camera feed so that even if the mobile device was moved, the bounding boxes would remain in same position over the components. The current component relevant to the tutorial instruction was also highlighted in this environment. The bounding box visualization was implemented using the instant tracking module, as shown in Fig. 3.

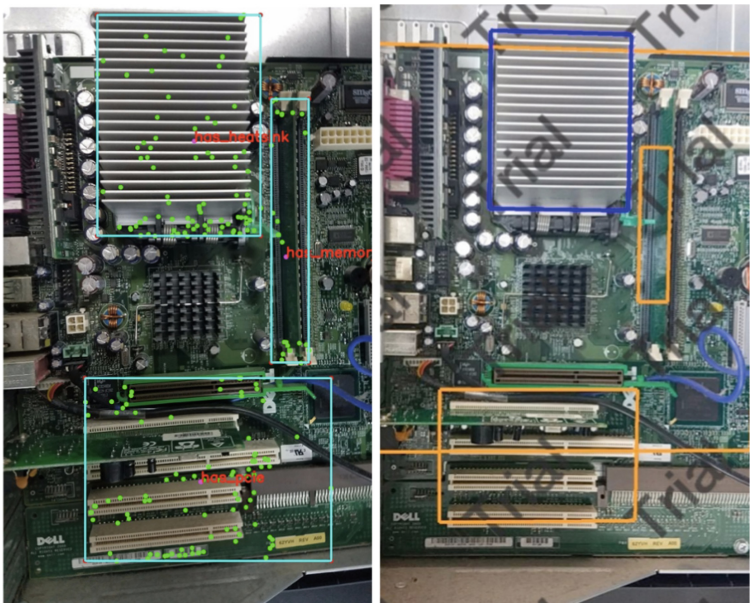

Fig. 3. (a) Showing the feature points, bounding boxes and labels placed during the recognition stage with (b) Showing the bounding boxes placed in the augmented reality application.

7 Training

Some methods, such as the Bag of Words Method and Local Binary Patterns Method, required a much larger training set than the Histogram Intersection method. Therefore a different training dataset was used for this implementation. The breakdown of this dataset is described in Table 1.

Table 1. Showing the image breakdown for the training dataset

	Heatsink	Memory	PCIE	Heatsink	Memory	PCIE
Component present	Yes	Yes	Yes	No	No	No
Image count	56	56	56	56	56	56

8 Testing

8.1 Motherboard Image Dataset

A dataset was compiled for testing the performance of various algorithms of interest. This dataset comprised of images of a Dell Optiplex g × 260 motherboard still in the case. The images were taken with a Samsung S7 (SM-G930A)

camera, had a height of 3024 pixels and a width of 4032 pixels, and were saved to a JPEG format. Different images were taken with the motherboard containing different components while varying the proximity of the camera to the motherboard. Also, the dataset contained slight changes in lighting conditions, but all of the photos were taken indoors. All images were taken parallel to the motherboard to reduce errors caused by changing the pitch of the camera. The breakdown of the dataset can be seen in Table 2.

Table 2. Showing the image breakdown for the training dataset

Heatsink	Memory	PCIE	Image count
Yes	Yes	Yes	14
Yes	Yes	No	14
Yes	No	Yes	14
Yes	No	No	14
No	Yes	Yes	14
No	Yes	No	14
No	No	Yes	14
No	No	No	14
Total			113

8.2 Motherboard Component Dataset

Another dataset was created to test the recognition of the mother board components. To create this dataset, the images from the Motherboard image dataset were used and the individual components of interest were cropped out and saved. These images were stored in the same JPEG format as the original images. The dimensions of the images varied depending on the component. The breakdown of this dataset can be seen in Table 3 below.

Table 3. Showing the image breakdown for the component dataset

	Heatsink	Memory	PCIE	Heatsink	Memory	PCIE
Component present	Yes	Yes	Yes	No	No	No
Image count	56	56	56	56	56	56

8.3 Object Detection Testing

Feature Point Method Testing. The feature point object detection algorithm was also tested against the Motherboard Image Dataset. To test the algorithm, three different feature point detectors were tested against the dataset. These

detectors included ORB, SURF and SIFT detectors and descriptors. For each descriptor tested, the feature point matches detected were drawn on the images so that false matches can be easily detected. Each feature point method was assigned a different color so that the results can be easily detected. For binary string descriptors, such as ORB, the hamming distance was used as the measurement, but for descriptors, such as SIFT and SURF, euclidean distance was used. This distance was used as a threshold for the point matches so that matches that were far off were not plotted. To assist with evaluating the performance of the algorithm, many parameters were recorded. The algorithms were tested at different image scaling to see how re-sizing the images impacted the performance of the algorithm. The size of the image was varied by increments of 10%. The number of feature points in the training image and the number of feature points matched in the test image was recorded. The percentage of matches was calculated using this value. The processing time taken for each iteration was also recorded. The results from this testing are seen in Table 4.

Table 4. Showing the object detection accuracy

	SIFT	SURF	ORB	SIFT	SURF	ORB	SIFT	SURF	ORB
	True positives			False positives			Accuracy		
0.1	122	13	0	217	326	339	35.99	3.83	0.00
0.2	209	150	0	130	189	339	61.65	44.25	0.00
0.3	273	201	0	66	138	339	80.53	59.29	0.00
0.4	285	241	0	54	98	339	84.07	71.09	0.00
0.5	285	221	0	54	118	339	84.07	65.19	0.00
0.6	279	215	0	60	124	339	82.03	63.42	0.00
0.7	258	217	0	81	122	339	76.11	64.01	0.00
0.8	241	224	0	98	115	339	71.09	66.08	0.00
0.9	237	222	1	102	117	338	69.91	65.49	0.29
1.0	240	217	6	99	122	333	70.80	64.01	1.77

8.4 Object Recognition Testing

Histogram Intersection Testing. The Histogram Intersection Algorithm was tested against the Motherboard Component Dataset discussed previously. For this testing, the dataset was trained with sample images and then the algorithm was used to attempt to recognize the images. The size of the images varied by 10% and the time taken for the recognition was recorded. This would be used to assess how scaling of the image affects the accuracy of the algorithm. Also the luminance component was removed from the image in an attempt to remove the effect of light intensity changes on the image. The test was then redone with these changes to determine the impact of these changes on the performance

of the module. The testing images were labelled before testing, and the accuracy was calculated based on how many of the predicted labels matched the pre-labels of the images. Based on the results, best average accuracy achieved for the Histogram Intersection without the luminance removal was 51% and it occurred at a scale of 0.8 as seen in Fig. 4. The accuracy of this method increased when the luminance component of the images was removed to about 74% and the recognition accuracy of each component was also increased as seen in Fig. 5. The comparative accuracy of both methods can be seen in Fig. 6.

Bag of Features Testing. The Bag of Features method was tested using the Motherboard Component Dataset discussed previously. To test this algorithm, the Bag of Features model was trained with discussed feature point detectors. First, it was trained with SIFT features, then with SURF features and last ORB features. The training was accomplished with the training component dataset discussed in the previous section. During testing, images were also scaled down by increments of 10% to assess the impact of scaling on this method of object recognition. Accuracy of each method of feature point detection was also determined. This was accomplished by manually labelling the test images and comparing these output of the Bag of Features model predictions. The SIFT performed the best at a scale of 0.2. The results obtained for the accuracy for each component can be seen in Fig. 9. At this scale, the average accuracy for all components was 95%. The SURF features, however, performed the best with this model at a scale of 0.8. At this scale, it achieved an average accuracy of 97%. The results for this is presented in Fig. 11. Finally, the orb features was able to achieve a the max accuracy of 96% at a scale of 0.2. The accuracy for each individual component can be seen in Fig. 10. A comparison of the accuracy of all the methods and over changes in scale is presented in Fig. 12. From this it can be seen that, at a 0.2 scale factor, the ORB Bag of Features achieved its peak accuracy. It can also be seen that the SURF and SIFT bag of features achieved an even higher accuracy than the ORB at certain scale factors. This was, however, a marginally better performance. A comparison of the average recognition time for each feature point technique with the Bag of Features Model is presented in Fig. 13. From this, it can be deduced that for a comparable accuracy the recognition time is shorter for the ORB bag of features method, making it the best method for this solution.

Local Binary Patterns - Support Vector Machine. The Local Binary Patterns algorithm was also tested using the Motherboard Component Dataset. The algorithm was implemented first using the training dataset discussed above, which contained images of the different motherboard components. To test the performance of this method, the radius of the Local Binary Pattern operator was varied by increments of 10 pixels. Also the image was scaled at increments of 10%. Just as before, the testing images were pre-labelled and the Support Vector Machine Training was used to label the images. The accuracy computed just as before using the number of correct labels divided by the total number

of images. The LBP SVM performed achieved a peak accuracy of 78% when it had a scale of 0.4 and radius of 10. The recognition accuracy for each component at the peak accuracy is presented in Fig. 7. The average recognition time results for the support vector implementation of the LBP is shown in Fig. 8.

Overall System Testing. The overall recognition system was tested using the Motherboard Image dataset that was discussed. This testing was done to assess the accuracy of the final recognition system and also the speed of operation. Using the best preforming algorithms at their maximum performance, configurations were determined by testing and reviewing the results. During this testing, the number of correct recognition cases over the total number of images was assessed. Also the time taken to process each image on average was recorded. Lastly a confusion matrix was developed for the system so that the performance can be more deeply analyzed.

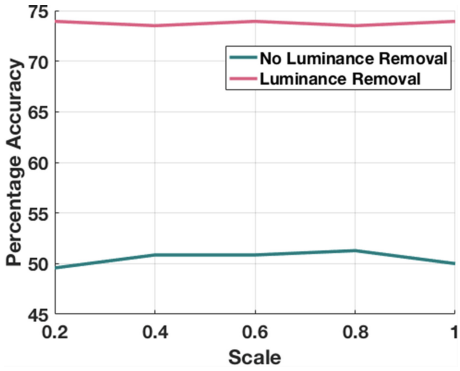

Fig. 4. Showing the recognition accuracy of both histogram intersection methods

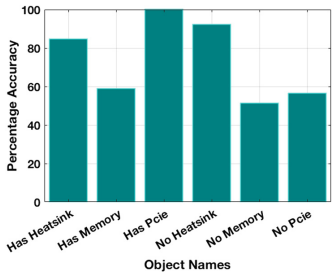

Fig. 5. Showing the recognition accuracy of the histogram intersection method for each component with luminance removal

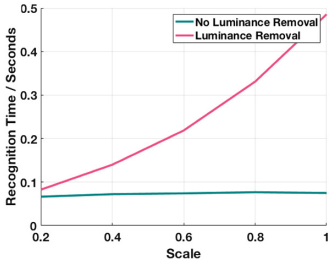

Fig. 6. Showing the comparison of average recognition time for the histogram methods

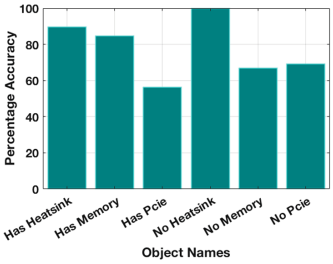

Fig. 7. Showing the recognition accuracy of the LBP at 0.4 scale and radius 10 – support vector machine.

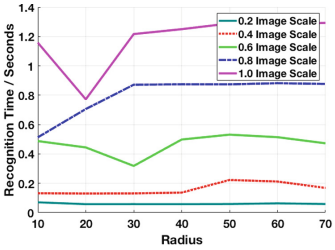

Fig. 8. Showing the average recognition time for the support vector machine implementation of local binary patterns

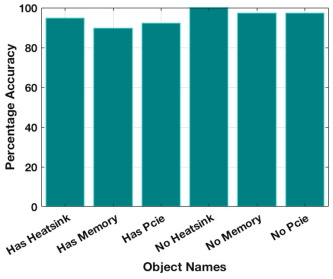

Fig. 9. Showing the recognition accuracy of the SIFT at 0.2 scale – bag of words method

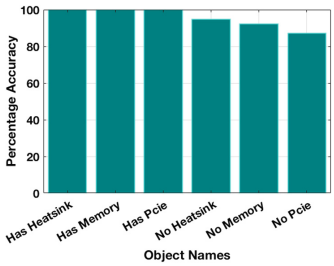

Fig. 10. Showing the recognition accuracy of the ORB at 0.2 scale – bag of words method

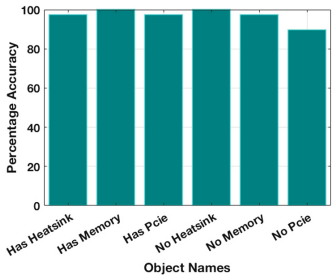

Fig. 11. Showing the recognition accuracy of the SURF at 0.8 scale – bag of words method

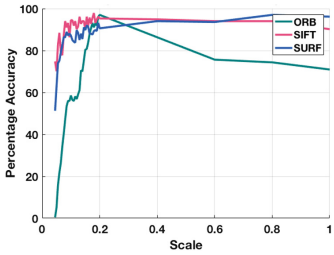

Fig. 12. Showing the average accuracy of the bag of features implementation of ORB, SIFT and SURF.

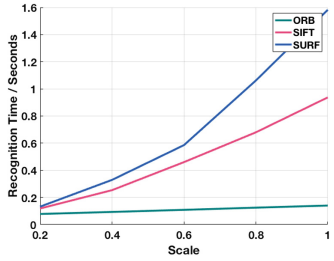

Fig. 13. Showing the average recognition time of the bag of features implementation of ORB, SIFT and SURF.

9 Conclusion

We presented a novel approach to object detection and recognition for real-time Augmented Reality applications. We implemented this algorithm in the context of an AR application for technology support to visually guide the field technicians through repair actions. We concluded that the ORB - Bag of Features method was the best suited method for object recognition, while the SIFT feature point method was the best suited for object detection. The ORB - Bag of Features method had a peak accuracy of 95% at a resolution scale of 0.2, while SIFT feature point method peaked at a resolution scale of 0.4. When compared the SIFT and SURF, Bag of Features method had a marginally better performance than the ORB method, but the ORB had a significantly lower recognition time making it the faster and better recognition method.

One significant limitation was that bounding boxes sometimes cannot be place at the desired location in the Augmented Reality application. This resulted in failed bounding box placements, which meant the boxes were sometimes not visible to the user. This was due to the sparse point cloud created by the Wikitude API, which was used for this implementation.

In order to increase the reliability of our algorithm, further improvements can be made to transfer the rotation of the bounding boxes to the Android side of the application. This would allow for correct placement of the Augmented Reality bounding boxes even under cases of extreme rotation. There is scope for exploration in the area of Augmented Reality implementations allowing for a more dense point cloud to be used so that the boxes can be more accurately drawn. The system can also be expanded to support more types of hardware systems and possibly using a database of images so that it can detect for an array of systems. One method of improving the results of the recognition and detection components may be with the use of a deep learning approach. This approach however would also require the collection of more training data, and may not scale well for unique hardware devices.

References

1. Garcia, M.L.D., Soto, D.A.P., Mihaylova, L.S.: A bag of features based approach for classification of motile sperm cells. In: 2017 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), pp. 104–109. IEEE (2017)
2. Karami, E., Prasad, S., Shehata, M.: Image matching using SIFT, SURF, BRIEF and ORB: performance comparison for distorted images (2015)
3. Piccinini, P., Prati, A., Cucchiara, R.: Real-time object detection and localization with SIFT-based clustering. Image Vis. Comput. **30**(8), 573–587 (2012)
4. Seib, V., Kusenbach, M., Thierfelder, S., Paulus, D.: Object recognition using Hough-transform clustering of SURF features. In: Workshops on Electronical and Computer Engineering Subfields, pp. 169–176 (2012)

5. Csurka, G., Dance, C., Fan, L., Willamowski, J., Bray, C.: Visual categorization with bags of keypoints. In: Workshop on Statistical Learning in Computer Vision, ECCV, vol. 1, no. 1–22, pp. 1–2, Prague (2004)
6. O'Hara, S., Draper, B. A.: Introduction to the bag of features paradigm for image classification and retrieval. arXiv preprint arXiv:1101.3354 (2011)
7. Li, W., Chen, C., Su, H., Du, Q.: Local binary patterns and extreme learning machine for hyperspectral imagery classification. IEEE Trans. Geosci. Remote Sens. **53**(7), 3681–3693 (2015)
8. Rublee, E., Rabaud, V., Konolige, K., Bradski, G.: ORB: an efficient alternative to SIFT or SURF. In: ICCV, pp. 2564–2571 (2011)
9. Asmara, R.A., Rahutomo, F., Hasanah, Q., Rahmad, C.: Chicken meat freshness identification using the histogram color feature. In: 2017 International Conference on Sustainable Information Engineering and Technology (SIET), pp. 57–61 (2017)
10. Swain, M.J., Ballard, D.H.: Color indexing. Int. J. Comput. Vis. **7**(1), 11–32 (1991)
11. Nguyen, D.T., Zong, Z., Ogunbona, P., Li, W.: Object detection using non-redundant local binary patterns. In: 2010 IEEE International Conference on Image Processing, pp. 4609–4612 (2010)
12. Ojala, T., Pietikäinen, M., Harwood, D.: A comparative study of texture measures with classification based on featured distributions. Pattern Recogn. **29**(1), 51–59 (1996)
13. Meena, K., Suruliandi, A.: Performance evaluation of local binary patterns and it's derivatives for face recognition. In: 2011 International Conference on Emerging Trends in Electrical and Computer Technology, pp. 742–746 (2011)
14. Qin, Y., Xu, H., Chen, H.: Image feature points matching via improved ORB. In: 2014 IEEE International Conference on Progress in Informatics and Computing, pp. 204–208 (2014)
15. Lowe, D. G.: Object recognition from local scale-invariant features. In: The Proceedings of the Seventh IEEE International Conference on Computer Vision, 1999, vol. 2, pp. 1150–1157. IEEE (1999)
16. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vis. **60**(2), 91–110 (2004)
17. Bay, H., Tuytelaars, T., Van Gool, L.: SURF: speeded up robust features. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006. LNCS, vol. 3951, pp. 404–417. Springer, Heidelberg (2006). https://doi.org/10.1007/11744023_32
18. Pietikinen, M., Hadid, A., Zhao, G., Ahonen, T.: Computer Vision Using Local Binary Patterns. Computational Imaging and Vision, vol. 40, p. 222. Springer, London (2011). https://doi.org/10.1007/978-0-85729-748-8
19. Ahmad, A., Yusof, R., Mitsukura, Y.: Identifying the dominant species of tropical wood species using histogram intersection method. In: IECON 2015–41st Annual Conference of the IEEE Industrial Electronics Society, pp. 003075–003080 (2015)
20. Chernov, V., Alander, J., Bochko, V.: Integer-based accurate conversion between RGB and HSV color spaces. Comput. Electr. Eng. **46**, 328–337 (2015)
21. Chien, H.J., Chuang, C.C., Chen, C.Y., Klette, R.: When to use what feature? SIFT, SURF, ORB, or A-KAZE features for monocular visual odometry. In: 2016 International Conference on Image and Vision Computing New Zealand (IVCNZ), pp. 1–6 (2016)
22. Raguram, R., Frahm, J.-M., Pollefeys, M.: A comparative analysis of RANSAC techniques leading to adaptive real-time random sample consensus. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) ECCV 2008. LNCS, vol. 5303, pp. 500–513. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-88688-4_37