Penerbit UTHM



**PAAT**

Journal homepage: http://penerbit.uthm.edu.my/ojs/index.php/paat
e-ISSN : 2821-2924

Progress in Aerospace and Aviation Technology

# Obstacle Detection for Unmanned Aerial Vehicle (UAV)

**Nor Aizatul Nabila Abdul Khaliq[1], Muhammad Faiz Ramli[2*]**

[1]Faculty Mechanical and Manufacturing Engineering,
Universiti Tun Hussein Onn Malaysia (UTHM), Batu Pahat, 86400, MALAYSIA

[2]Aircraft System and Design Research, Faculty Mechanical and Manufacturing Engineering,
Universiti Tun Hussein Onn Malaysia (UTHM), Batu Pahat, 86400, MALAYSIA

*Corresponding Author

DOI: https://doi.org/10.30880/paat.2022.02.01.007
Received 2 July 2022; Accepted 23 August 20220; Available online 28 August 2022

**Abstract:** This study aims to develop an obstacle detection system for unmanned aerial vehicles utilising the ORB feature extraction. In the past, small unmanned aerial vehicles (UAV) were typically equipped with vision-based or range-based sensors. Each sensor in the sensor-based technique possesses different advantages and disadvantages. As a result, the small unmanned aerial vehicle is unable to determine the obstacle's distance or bearing precisely. Due to physical size restrictions and payload capacity, the lightweight Pi Camera and TF Luna LiDAR sensor were selected as the most suitable sensors for integration. In algorithm development and filtration is used to improve the accuracy of the feature matching process, which is required for classifying the obstacle region and free region of any texture obstacle. The experiment was under the environment of OpenCV and Spyder. In real-time experiment, the success rate for good texture (40%), poor texture (55%) and texture-less (45%) The findings indicate that the recommended method works well for detecting textures-less obstacle even though the success rate is only 40% because out of 10 test only one test is fail on detecting free region. The sensor calibration and constructing convex hull for obstacle detection is recommended in future works to improve the efficiency of the obstacle detection system and classified the free region and obstacle region to create safe avoidance path.

**Keywords:** Raspberry Pi, OpenCV, UAV

## 1. Introduction

Advanced processing power and microelectronics have led to development of small unmanned aerial vehicles. All living organisms possess a multisensory fusion, including eyes that detect threats and muscles that move to avoid them. By applying this natural approach, sensor integration for applications involving unmanned aerial vehicles could improve the system's robustness in recognising and avoiding frontal obstacle. The vision-based sensor alone can determine the bearing angle of the obstacle. However, this sensor is incapable of determining the precise distance between the vehicle and the obstacle. As the range-based sensor is inadequate at detecting the bearing angle, the sensor added to the detection system to compensate for the vision-based sensor's inefficiency.[1]

The vision-based sensor can determine the depth of the indoor environment using monocular cue by identifying the perspective lines or edges in the single image. However, this method only limited to only indoor environment when the vanishing point consistently found in the corridor asnd at the staircase. The monocular sensor unable to detect whether the scene is free region or contain obstacle because of the texture-less surface for example the wall, poles etc. [2] The dimensional depth estimation through only one eye is one characteristic of a monocular camera. Relative size is one of the most noticeable monocular cues. It acts as a monocular depth cue by looking at the size of the objects in the sceneries in their various configurations. The depth estimation can be derived by the relation of size and distance if we have two

*Corresponding author: faizr@uthm.edu.my

objects having a similar size. Logically, the smaller the distance of the observer to the object, the bigger the object appears in the observer's eyes.[3]

In agricultural activities, the range-based sensor (LiDAR) has been used on the UAV for collision avoidance. The range-based sensor helps the UAV to avoid texture-less obstacles like poles and wires. The range-based sensor senses the obstacle at a specified distance and then triggers a caution signal. When the UAV is approaching the obstacle at less than the specified distance the UAV will avoid the obstacle.[4] LiDAR sensor is active because it continuously transmits its energy to the surroundings while the vision-based sensor can obtain tremendous information of surroundings or example perspective cues, size relative, motion parallax etc.[5] This relation is the method to integrate both sensors.

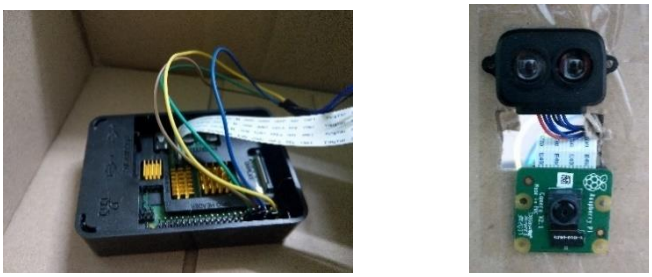
The UAV takes frame-by-frame video from the front camera and transmits it to the computer vision system for depth measurements (depth perception technique). The computer vision system constructs a time series of data on scale-invariant key points from a few frames to estimate depth. The next command for obstacle-free navigation also generated by computer vision. The dependability of feature matching was necessary for accurate image registration when using remote sensing. Outliers reduce the precision and effectiveness of feature matching.[6] This study focuses on matching features with the ORB algorithm [7] in OpenCV. The features collected by the ORB will be analyzed in the Spyder environment by transforming them into a series of NumPy arrays to evaluate the matching result, and then all variables associated with false matching (outliers) will be filtered out.

## 2. Methodology

This section provides a full overview involving the equipment implementation and the proposed method to remove the false matching in extracted features by ORB algorithm.

### 2.1 Hardware and software implementation

This experiment's equipment consists of two components: hardware and software. The hardware includes a Raspberry Pi, TF Luna LiDAR and Pi Camera sensors. The software components are Spyder, OpenCV, NumPy, and the ORB algorithm.



(a) (b)

**Fig. 1 - (a) Raspberry Pi; (b) TF-Luna 850nm LiDAR module (upper part) and Pi Camera module (lower part)**

The Raspberry Pi is an inexpensive and credit-card-sized computer that plugs into a computer display or television and uses a standard keyboard and mouse. This single-board computer runs the Raspberry Pi operating system, and VNC Viewer can be used to set up two monitors connected to the same IP address.[8] Raspberry Pi is the core hardware that will act as a small brain of a small UAV.[9]

The mini UART serial port (RXD/TXD) on the Raspberry Pi 4 computer is used to connect TF-Luna, and the 5V pin is used to power it. Python in Spyder IDE will be used to set up the LiDAR module and test it. TF-Luna 850nm LiDAR module uses the time-of-flight (ToF) principle to detect objects in its range of view. Depending on ambient light and surface reflectivity, the TF-Luna can measure objects 20cm to 8m distant. The TF-Luna is a Class 1 laser with a VCSEL, making it safe for practically all applications.[10] Pi Cameras is connected to a Raspberry Pi 4 using the camera module connector, and the Python Pi Camera library will be used to operate the Pi Camera so that it can capture images.

For software components, using Spyder environment the feature points variables in the training image are extracted by ORB algorithm in OpenCV.

### 2.2 Feature Extraction by ORB

The prior step in object detection for computer vision is extracting features from an object in an image. In this research, ORB algorithm which is the fastest algorithm surpassing SIFT and SURF is applied in the feature detection on the obstacle.

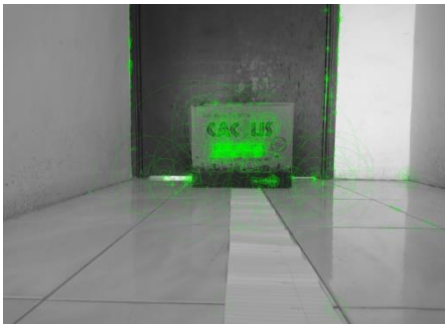


**Fig. 2 - Detected feature by ORB**

In the same scene, two different images of an obstacle are captured at a different distance of the camera to obstacle from approaching to farther from the obstacle to match the detected feature using ORB algorithm. The captured image at approaching the camera, the obstacle is bigger than farther the camera from the obstacle. Feature matching involves train images and query images. The features are firstly detected in the train image and these features become the scheme for the query image. As minimum as possible of false matched features is required in the training process before the feature matching algorithm is applied in the real-time experiment to ensure the efficiency of obstacle detection. Every feature detected by the ORB algorithm comprises key points and descriptors. The variables compiled in the key pints are scale and orientation whilst the descriptors contain the visual description of the patch to compare the similarity between image features. Fig. 3 shows the train image on the right side and the query image on the left side after feature matching.

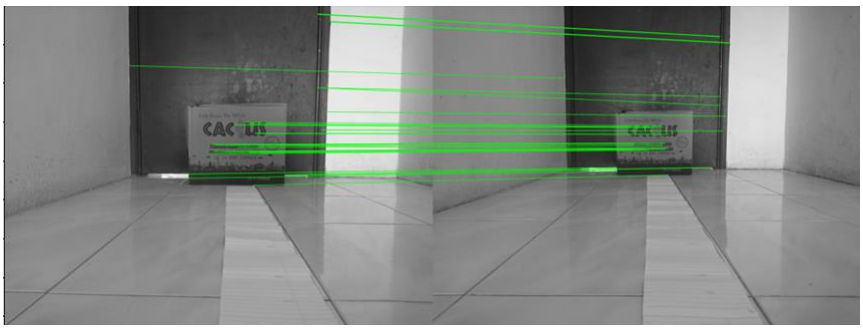


**Fig. 3 - Feature matching by ORB algorithm**

## 2.3 Distance Ratio Cue Technique and Filtration

The depth perception technique used is based on expanding the detected feature points between image frames. With this technique, the proposed obstacle detection and avoidance system can simultaneously detect the presence of dangerous obstacles and clear areas in the environment. The depth cues by expansion are known as distance ratio cues.

The function in Fig. 4 indicated that feature points distance is inversely proportional to the distance from the camera sensor. This concluded that as the camera sensor moves farther from the object, the feature points of an object is undistinguishable compared to an object approaching the camera sensor. This information will help detect the occurrence of obstacles in the drone's operating environment and ultimately identify the appropriate path for the drone to safely take evasive action(maneuver).

To construct the depth cue graph, raw data is collected from three selected features on the obstacle. Fig. 4 shows the three features Line 1(SOP1), Line 2(SOP2) and Line 3(SOP3). Those marked features are the same for all frames captured at 10 cm to 200 cm at 5 cm intervals.

Then, line distance for SOP1, SOP2 and SOP3 is calculated. This process repeated for every captured image from 10 cm to 200cm. In the Microsoft excel, the calculated distance is analyzed to find the distance ratio and average distance ratio for every marked line by 30 cm distance separation difference and the distance ratio template.

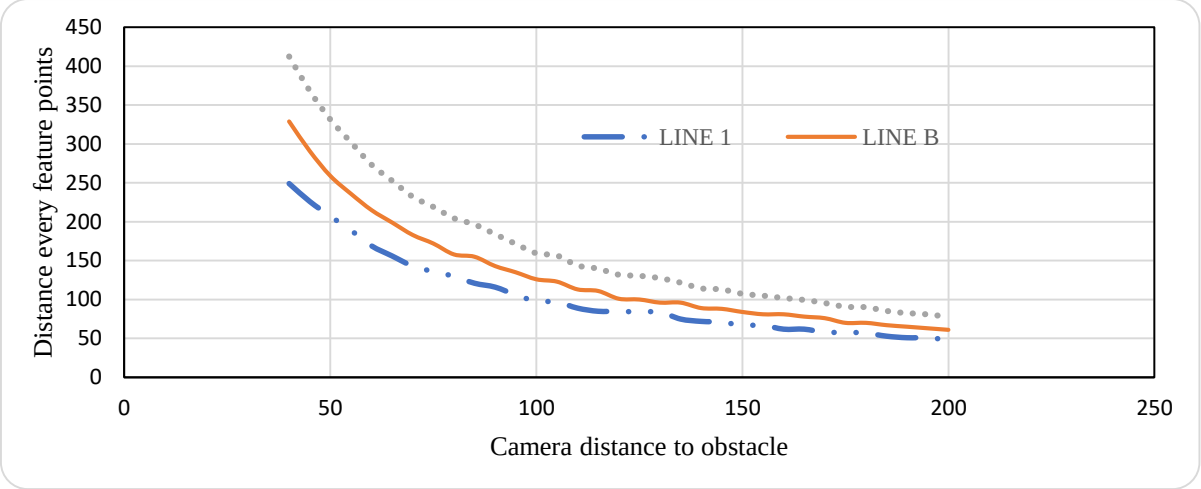



**Fig. 4 – Distance ratio cue graph**

**Fig. 5 - SOP1(yellow), SOP2(pink) and SOP3(green)**

**Table 1 - Distance ratio for 30 cm distance separation**

| Distance | SOP1 | SOP2 | SOP3 | dR1 | dR2 | dR3 | Average dR |
|---|---|---|---|---|---|---|---|
| 70 | 143 | 183 | 232.24 | 1.74 | 1.79 | 1.77 | 1.77 |
| 80 | 130 | 158 | 204.60 | 1.59 | 1.63 | 1.62 | 1.61 |
| 90 | 116 | 143 | 184.13 | 1.45 | 1.5 | 1.48 | 1.48 |
| 100 | 98 | 126 | 159.62 | 1.45 | 1.45 | 1.45 | 1.45 |
| 110 | 89 | 113 | 143.84 | 1.46 | 1.39 | 1.42 | 1.43 |
| 120 | 85 | 101 | 132.00 | 1.36 | 1.41 | 1.40 | 1.39 |

**Table 2 - Distance ratio template**

| Distance | Average dR |
|---|---|
| 70 | 1.76 |
| 80 | 1.61 |
| 90 | 1.50 |
| 100 | 1.43 |
| 110 | 1.38 |
| 120 | 1.34 |
| 130 | 1.30 |
| 140 | 1.28 |
| 150 | 1.26 |

The selection of distance ratio value is by referring to the template in Table. 2 and distance ratio cue graph in Fig. 4. From the ratio cue graph, the permissible distance ratio is from 150 cm to 70 cm. In the thresholding process, 1.26 is selected and applied to the observed reference point. In the detection algorithm, the matching feature points is an obstacle if the distance ratio is greater than 1.26. Table. 3 is an example the determined obstacle and free region feature points which denoted with logic 1 and 0.

**Table 3 - Thresholding process with ORP**

| Distance ratio | Thresholding value |
|---|---|
| 0 | 0 |
| 1.28335 | 1 |
| 1.30418 | 1 |
| 1.24054 | 0 |
| 1.27453 | 1 |

From distance ratio template, the tolerant value also can be set to remove the false matching. In the algorithm development, the permissible tolerant value is between 1.26 to 1.70 as this tolerant is applicable to produce a perfect matching result after applying the scale changes filtering.

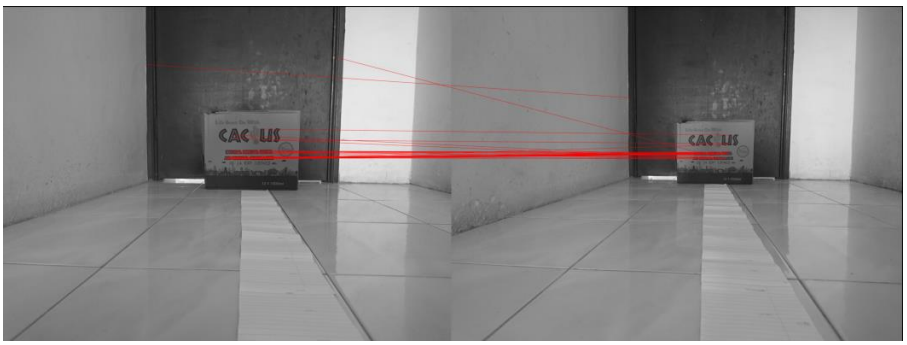


**Fig. 6 - Feature matching with dr at 1.26**

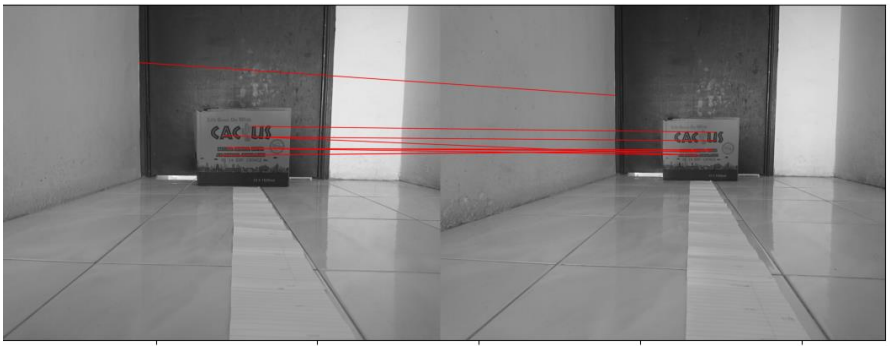


**Fig. 7 - Feature matching with dr between 1.26 – 1.7**

## 2.4 Scale Changes Filtering

The basic ORB feature matching is in the following step: extraction, description and match the features. This feature detector algorithm is greedily detecting all the features and match them according to the same description in the train and query image. It is found that some matched features that hold negative scale difference values in the key point. From Fig. 8, there are two false matched features, the false matched are with negative octave as in Table. 4.

**Fig. 8 - Matched features at distance 170 cm as query image and 200 cm as train image.**

**Table 4 - Exact location and octave scales for matched features**

| Image at 170 cm from obstacle | | Image at 200 cm from obstacle | | Octave scale difference |
|---|---|---|---|---|
| **Location** | **Octave 1** | **Location** | **Octave 2** | |
| (549.0,786.0) | 0 | (293.76,551.23) | 3 | -3 |
| (2133.73,582.68) | 4 | (2106.43,566.79) | 3 | 1 |
| (606.0,812.0) | 0 | (295.20,550.08) | 3 | -3 |
| (2252.0,739.0) | 0 | (2228.0,686.0) | 0 | 0 |
| (1278.0,1415.36) | 6 | (1280.99,1355.64) | 6 | 0 |

## 2.5 Angle Filtering

In improving a reliable feature matching from two corresponding points between the train and query image, the angle filtering last method was proposed to improve the false matching from a simple ORB algorithm. From feature matching in Fig. 9, there are one obvious false matching. Logically, the polar coordinate in the train image is greater than in the query image. The difference of polar coordinate between both images is relatively small and some values is approaching 0 and this characteristic is set in the angle filtration method for good feature matching. $15^0$ of polar coordinate difference is chosen because this is the most suitable value to give small convergence to remove the false matching. Fig. 10 and Table. 5 show more detail the pattern of false matching and their polar coordinate.

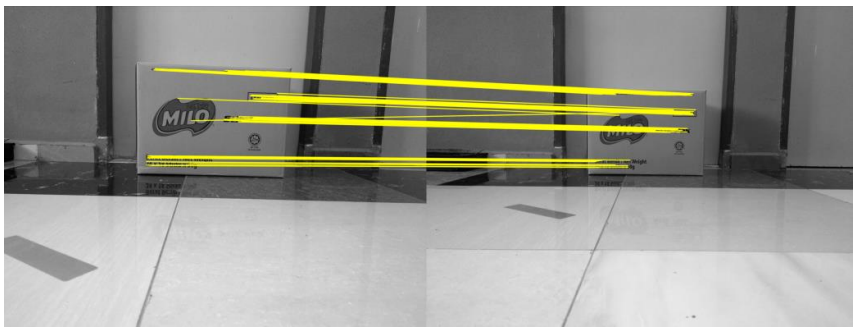



**Fig. 9 - Feature matching before angle filtration applied**

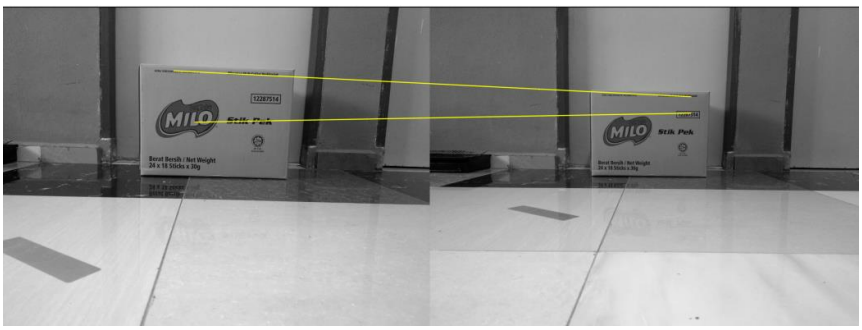



**Fig. 10 - False feature matching**

**Table. 5 - Location and polar coordinate of false matching**

| Index | X1 | Y1 | X2 | Y2 | $\alpha_1$ | $\alpha_2$ | $\alpha_1 - \alpha_2$ |
|---|---|---|---|---|---|---|---|
| 106 | 1654.56 | 594.72 | 2425.2 | 835.2 | 1.12716 | 176.549 | $175.422^{O}$ |
| 129 | 1848.92 | 1071.37 | 2525.2 | 987.6 | 56.3269 | 150.524 | $94.1973^{O}$ |

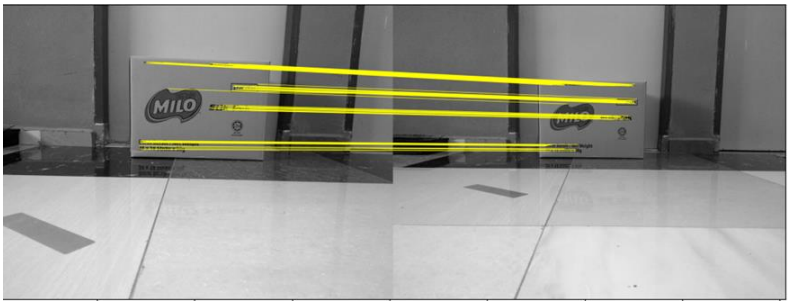



**Fig. 11 - Good feature matching result**

## 3. Result and Discussion

In real time experiment, the Raspberry Pi equipped with TF Luna LiDAR and Pi Camera is carried approaching the obstacle. when the range-based sensor detects the distance at 150 cm and 120 cm the query and train image is captured. This experiment purpose is to observe the performance of obstacle detection on different obstacle texture especially on the texture-less obstacle. The three obstacles are as follows:

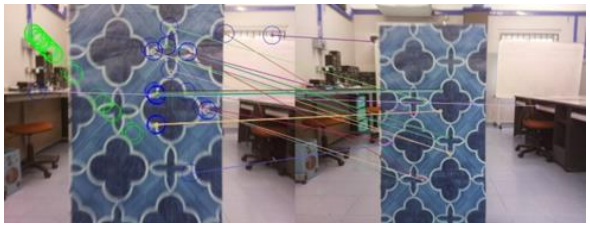


**Fig. 12 - Good texture**

**Fig. 13 - Poor texture**

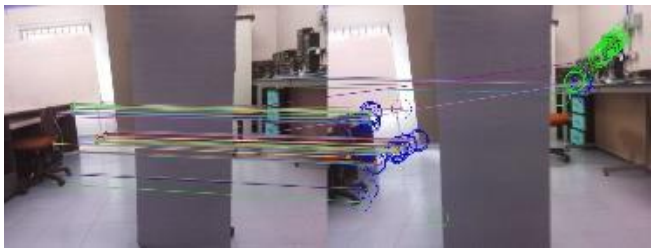


**Fig. 14 - Texture-less**

The proposed obstacle detection algorithm is validated by 10 series of validation test. Using OpenCV circle, the obstacle and free region is represented by the blue and green circle

According to Table. 7, the blacked items represent values larger than 0.5. The result indicates that there is no value to represent the detected obstacle region for textureless obstacles. In Fig. 15, texture-less obstacle in test 6, the feature points for free and the obstacle region are detected on the left side of the image, but the obstacle region is not detected on the texture-less obstacle. The explanation is that the detection system "seen" a texture-rich background on the left side. There is a significant gap between the green and blue circles. The algorithm is able to classify the obstacle detection region on the side of a texture-less obstacle despite the absence of an obstacle region detection. When comparing Fig. 15 and Fig.16, it can be seen that the obstacle size detected in the train frame and the query frame differs. Logically, the distance between each collected image frame that is maintained at 30 centimeters, the size of the obstacle in each test should remain unchanged (150 cm to 120cm).

**Table 6 - Result from 10 Tests**

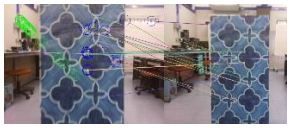
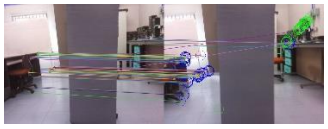
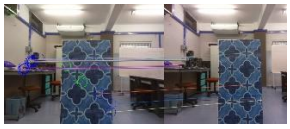
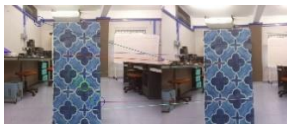
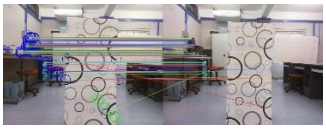
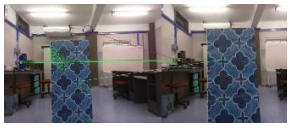
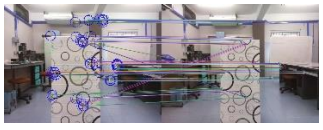
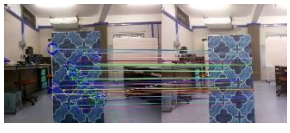
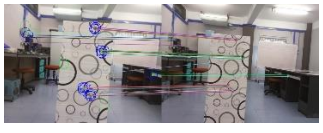
| Test | Good Texture | Poor texture | Texture-less |
|---|---|---|---|
| 1 |  |  |  |
| 2 |  |  |  |
| 3 |  |  |  |
| 4 |  |  |  |
| 5 |  |  |  |
| 6 |  |  |  |
| 7 |  |  |  |
| 8 |  |  |  |
| 9 |  |  |  |
| 10 |  |  |  |

**Table. 7 - Success rate for every texture obstacle**

| Test | Good texture | | Poor texture | | Texture-less | |
|---|---|---|---|---|---|---|
| | Free region | Obstacle Region | Free region | Obstacle region | Free region | Obstacle region |
| 1 | 1/4 | 3/7 | 3/3 | 3/8 | 2/2 | 0/16 |
| 2 | 15/26 | 24/26 | 7/15 | 3/12 | 7/7 | 0/18 |
| 3 | 9/13 | 3/5 | 13/30 | 5/9 | 20/20 | 0/30 |
| 4 | 1/5 | 5/15 | 1/3 | 0/1 | 4/4 | 0/2 |
| 5 | 0/5 | 5/6 | 1/2 | 9/11 | 19/19 | 0/8 |
| 6 | 1/8 | 1/5 | 25/39 | 18/20 | 19/19 | 0/13 |
| 7 | 0/4 | 8/11 | 0/8 | 20/30 | 3/4 | 0/8 |
| 8 | 4/15 | 8/12 | 2/4 | 35/45 | 2/6 | 0/14 |
| 9 | 0/3 | 7/15 | 0/1 | 36/57 | 13/13 | 0/4 |
| 10 | 0/0 | 25/30 | 2/11 | 19/26 | 4/7 | 0/5 |
| Success rate | 40% | | 55% | | 45% | |

Fig. 16 depicts an issue occur in texture-less obstacle detection also happen to poor texture obstacle. Because background is more texture rich and has unique features, the system discovered more obstacle regions in the background. At some point, the obstacle detection yields a satisfactory result. This occurs when we divide the frame into two sections, as shown in Fig. 18, with the free region grouped on the left side and the obstacle region perfectly detected on the poor texture obstacle. If we combine this detection method with a safe avoidance path algorithm, the UAV will do a manoeuvre to the left of the obstacle.

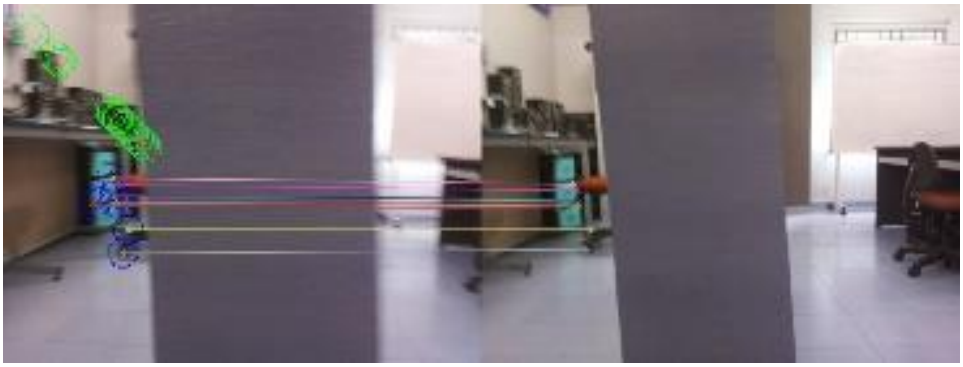


**Fig. 15 - Result from test 6 for texture-less obstacle**

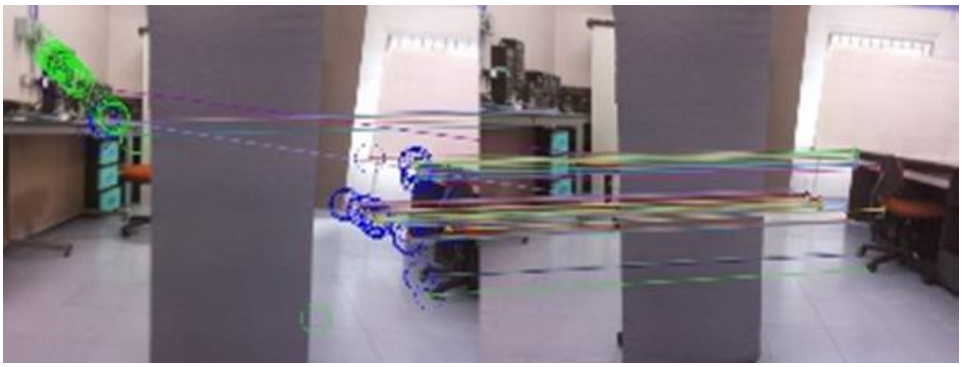


**Fig. 16 - Result from test 3 for texture-less obstacle**

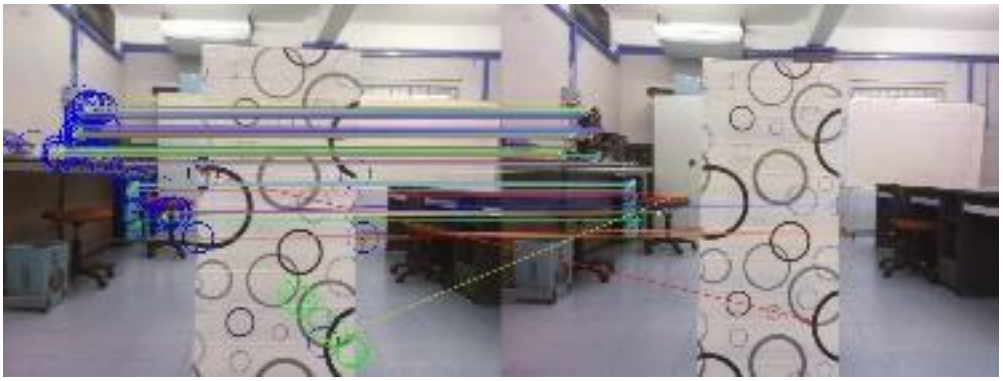


**Fig. 17 - Result from test 5 of poor texture obstacle**

**Fig. 18 - Result from test 10 of poor texture obstacle**

Because the success rate on good texture obstacle is the lowest in the success rate analysis in Table. 7, one question has arisen regarding this obstacle detection. Why are the free region and obstacle region identified on the obstacle? As we can see, the good texture characteristic introduced in this experiment has more shapes and unique features, resulting in more features spotted on the obstacle. The obstacle region is perfectly detected in Fig. 19, but the free region is also detected on it. By dividing the train image frame into two sections, it is possible to see that the obstacle region is prominent on the left side. Even if the free region is detected on the obstacle, the detected free region is prominent on the right side. Fig. 20 shows that the freer region detected outside the obstacle while they are more obstacle region detected on the obstacle. When comparing the size of the images obtained in Fig. 19 and Fig. 20, the size of the obstacles differs, and the image captured in Fig. 20 is blurry. This issue also occurred in another test, and the cause of this circumstance is the TF Luna LiDAR delay.

**Fig. 19 - Result test 9 on good texture**

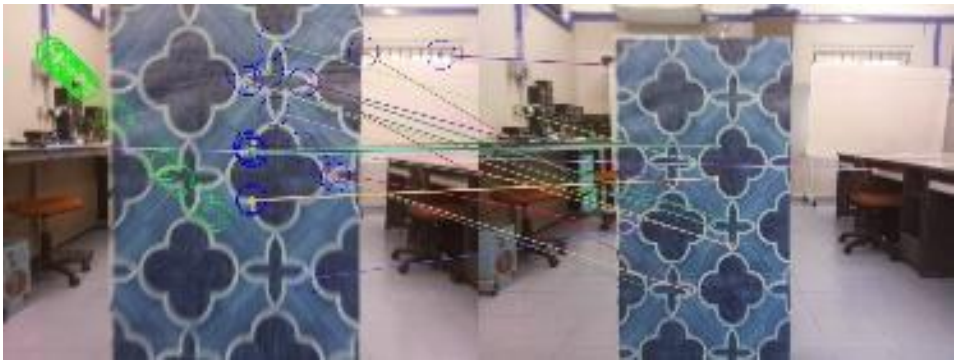


**Fig. 20 - Result test 2 on good texture**

## 4. Conclusion and Recommendation

The purpose of this research was to investigate the proposed method for the detection of free region towards the textureless obstacle. As stated in previous section, most of the previous research focused only on object detection using the vision-based or range-based sensor. It is implemented in this research the integration of Pi Camera with TF Luna LiDAR and then this sensor equipped the Raspberry pi. The input obtained from the sensors is processed by the ORB algorithm and proposed filtration for feature matching. The interaction of Python's module; OpenCV module and NumPy module enable to analyze the variables related to the features and develop the scale filtration and angle filtration. The filtration methods improve the training process of feature matching to reject the false matching or outliers. Lastly, using OpenCV circle the detected feature are classified into the free region and obstacle region. When the percentage of detected free region or obstacle region is greater than 50%, the feature matching is deemed successful. The experiment measured the detection success rate for obstacles with good texture (40%), poor texture (55%), and texture-less (45%). The result of success rate for texture-less obstacle is below the standard success rate that was set for this research. However, through in-depth analysis of 10 series of feature matching the limitations in the detection system is the sensor delay. In future works, solution to avoid sensor delay is by making sure the working temperature of TF Luna LiDAR is from $-10^0$C to $60^0$C only. If the temperature is out of the range, stop for a moment to avoid the risk of damage and bad sensor performance. It is also recommended to do the camera calibration by the calibration algorithm from OpenCV to obtain the extrinsic and intrinsic parameters of the camera and also the distortion coefficients to correct the vision-based sensor distortion. In the obstacle detection algorithm, the creation of a convex hull is suggested as additional effort to handle the problem of classifying the free zone and the obstacle region. The concept of the convex hull method is derived from the quick hull algorithm.[11] The expansion of a triangle produced from scattered point data inspired the development of the fast hull method. Differentiation between free and obstacle regions on detected obstacles, using the assumption that obstacle feature points are the larger feature points from the first acquired frame. The convex hull will be constructed using these feature points.[12]

## Acknowledgement

This research was supported by Universiti Tun Hussein Onn Malaysia (UTHM) through Tier 1 Grant (H920)

## References

[1] M. F. bin Ramli, S. S. Shamsudin, and A. Legowo, "Safe avoidance path detection using multi sensor integration for small unmanned aerial vehicle," in *5th IEEE International Workshop on Metrology for AeroSpace, MetroAeroSpace 2018 - Proceedings*, Aug. 2018, pp. 101–106. doi: 10.1109/MetroAeroSpace.2018.8453521.

[2] A. Saxena, S. H. Chung, and A. Y. Ng, "3-D depth reconstruction from a single still image," *International Journal of Computer Vision*, vol. 76, no. 1, pp. 53–69, Jan. 2008, doi: 10.1007/s11263-007-0071-y.

[3] C. Bills, J. Chen, and A. Saxena, "Autonomous MAV flight in indoor environments using single image perspective cues," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2011, pp. 5776–5783. doi: 10.1109/ICRA.2011.5980136.

[4] S.-A. Song, S. Kim, J. Kim, S. Song, and J. Suk, "APISAT2014," 2014. [Online]. Available: www.sciencedirect.comwww.elsevier.com/locate/procedia1877-7058

[5] F. Kendoul, "Survey of advances in guidance, navigation, and control of unmanned rotorcraft systems," *Journal of Field Robotics*, vol. 29, no. 2, pp. 315–378, Mar. 2012, doi: 10.1002/ROB.20414.

[6] L. Mi, Y. Qiao, J. Yang, and L. Bai, "Robust matching of SIFT keypoints via adaptive distance ratio thresholding," in *Sixth International Conference on Machine Vision (ICMV 2013)*, Dec. 2013, vol. 9067, p. 90670I. doi: 10.1117/12.2049905.

[7] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *Proceedings of the IEEE International Conference on Computer Vision*, 2011, pp. 2564–2571. doi: 10.1109/ICCV.2011.6126544.
[8] "Raspberry Pi." https://www.raspberrypi.com/ (accessed Jun. 23, 2022).
[9] T. Q. Khoi, N. A. Quang, and N. K. Hieu, "Object detection for drones on Raspberry Pi potentials and challenges," *IOP Conference Series: Materials Science and Engineering*, vol. 1109, no. 1, p. 012033, Mar. 2021, doi: 10.1088/1757-899x/1109/1/012033.
[10] "Laser Classification Explanation." https://ehs.lbl.gov/resource/documents/radiation-protection/laser-safety/laser-classification-explanation/ (accessed Jun. 23, 2022).
[11] J. S. Greenfield, J. Greenfield, and J. Greenfield, "A Proof for a QuickHull Algorithm A Proof for a QuickHull Algorithm A Proof for a QuickHull Algorithm A Proof for a QuickHul1 Algorithm," 1990. [Online]. Available: https://surface.syr.edu/eecs_techreports/65
[12] A. Al-Kaff, F. García, D. Martín, A. de la Escalera, and J. M. Armingol, "Obstacle detection and avoidance system based on monocular camera and size expansion algorithm for UAVs," *Sensors (Switzerland)*, vol. 17, no. 5, May 2017, doi: 10.3390/s17051061.