Flight Test Results of the Collision Avoidance System for Unmanned Aerial Systems using Stereoscopic Vision

Anthony Tan[1], Jake Banuelos[2], Kishan Patel[3], Minwoong Chae[4], Jonathan Cratsley[5], Muhammad Moiz Khanqadri[6], Daniel Boebinger[7], Andrew Nipp[8], Jeff Wang[9], Subodh Bhandari[10], and Daisy Tang[11]
Cal Poly Pomona, Pomona, CA 91768

This paper discusses using stereoscopic vision as a means to sensing and detecting obstacles and other aircraft for collision avoidance system for a small UAS. Using two cameras, it is possible to use depth perception to create a depth map that helps with the sensing of the obstacles and their distances. Two Point Grey Chameleon3 cameras are mounted on a SigKadet UAV as well as a multicopter while using an Intel NUC for onboard processing. The Intel NUC communicates with the PixHawk 3DR, which transmits data to the ground control station via Xbee radios. The Intel NUC generates a disparity map using an algorithm that uses the OpenCV library to process the images into the map. The algorithm generates the disparity map that will be provided to the collision avoidance algorithm, which will guide the UAS to the location within the map with the least dense area. The image processing algorithm is designed to remove noise in the image data. Some flight test results are presented.

Nomenclature

O_L	=	left optical center
O_R	=	right optical center
X	=	point captured by both cameras
X_L	=	left image plane point
X_R	=	right image plane point
P	=	point in physical plane
e_L	=	left epipole
e_R	=	right epipole
p_l	=	left projection point
p_r	=	right projection point
x^l	=	*left project point*
x^r	=	*right project point*

[1] Undergraduate Student, Aerospace Engineering Department, and AIAA Student Member.
[2] Undergraduate Student, Aerospace Engineering Department.
[3] Undergraduate Student, Aerospace Engineering Department.
[4] Undergraduate Student, Aerospace Engineering Department, and AIAA Student Member.
[5] Undergraduate Student, Aerospace Engineering Department, and AIAA Student Member.
[6] Undergraduate Student, Aerospace Engineering Department, and AIAA Student Member.
[7] Undergraduate Student, Aerospace Engineering Department, and AIAA Student Member.
[8] Undergraduate Student, Computer Science Department.
[9] Graduate Student, Computer Science Department.
[10] Professor, Aerospace Engineering Department, 17-2116, and AIAA Associate Fellow.
[11] Associate Professor, Computer Science Department, 8-11.

I. Introduction

UNMANNED aerial systems (UAS) have become very popular over the past decade. They can be used for applications such as precision agriculture, aerial photography, surveillance of disaster hit areas, etc. With this popularity, there has been safety concerns about integrating the UASs into the National Airspace System (NAS). The rise in UAS use has caused a call for many safety protocols. One of these is that the UAS must show the capability to sense and avoid static and dynamic obstacles including other aircraft. Methods for obstacle detection include LIDAR, ADS-B sensors, and laser range finders.[1] However, LIDAR and ADS-B can be very expensive and not available to small UASs because of size and weight limitation.[2] This paper proposes the use of two cameras for distance measurement using stereoscopic vision. Stereoscopic vision based obstacle avoidance can be advantageous because of low cost and light weight of cameras. However, one of the main problems is to be able to process the images sufficiently faster in real-time to detect the objects.

Some of the exiting works use one camera with optical flow algorithms.[3] However, optical flow has several disadvantages. Similar to how human perceive depth using two eyes, stereoscopic vision technique uses two images together to be able to find the depth of the correlating images and generate a depth map.[4] The distance to the object is found using an algorithm utilizing the baseline distance between the cameras and the focal lengths. The depth map is then fed to an autopilot to correct the course of the UAS to avoid an impending collision. OpenCV is used for image processing.

The proposed method is tested on a Cal Poly Pomona's Sig Kadet Sport UAS as well as a DJI's S900 multicopter. The project is a continuation of the project on collision avoidance using stereoscopit vision.[5] The goal of this paper was to 1) Develop methods to eliminate noise from the sensing part of the algorithm so that it does not affect the avoidance algorithm; 2) Decrease processing power needed for the vision code to decrease the time to process each disparity map; 3) Increase the range so that the objects at longer distances can be detected; and 4) Test the algorithms in flight tests, and analyze the benefits and drawbacks of stereoscopic vision for a single UAS moving at moderate speeds.

The paper is organized as follows. Section II introduces the concept of stereovision. Hardware used for the project is presented in Sections III. Algorithm development is discussed in the fourth section. Section V discusses the results followed by conclusion and future work in the last section.

II. Stereoscopic Vision

A. Image Rectification

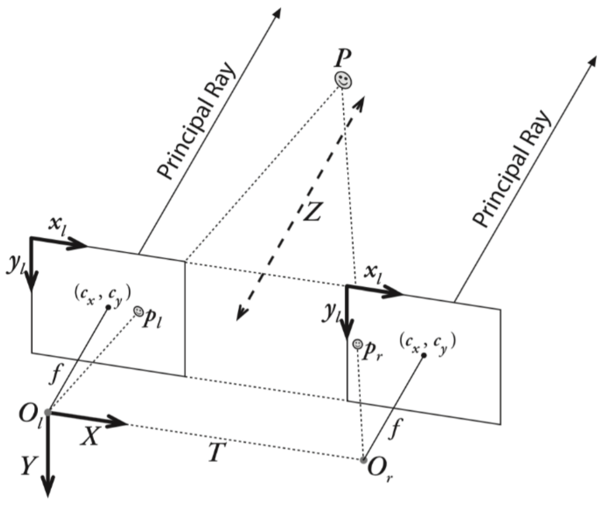


Fig. 1. Stereovision system.

The stereovision system is shown in Fig. 1. The two image planes are coplanar to each other, and the projection point, p_l and p_r, are on the same row. However, the real stereovision system rig is barely reaching the requirement. The corresponding projection points on the two image planes may not locate on the same row. Therefore, the purpose of image rectification is to make them row-aligned, which means making the corresponding projection points to be on the same pixel level. To do the image rectification, epipolar geometry is introduced.

Epipolar geometry is shown in Fig. 2. The two rectangles represent the image plane of left camera and right camera, respectively, and X represents the points captured by both cameras. $\mathrm{O_L}$ and $\mathrm{O_R}$ are optical centers. X projects on the left image plane on point $\mathrm{X_L}$ and projects on the right image plane on point $\mathrm{X_R}$. The line from $\mathrm{O_L}$ to $\mathrm{O_R}$ is called baseline. The intersection points of baseline and two image planes are called epipole, $\mathrm{e_L}$ and $\mathrm{e_R}$. The lines formed by $\mathrm{X_R}$ and $\mathrm{e_R}$ is called epipolar line as the line formed by $\mathrm{X_L}$ and $\mathrm{e_L}$. The goal of the rectification is making the two epipolar lines parallel to make $\mathrm{X_R}$ and $\mathrm{X_L}$ on the same pixel row. The next step would be camera calibration.

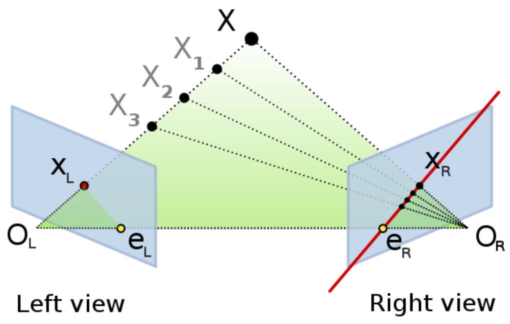


Fig. 2. Epipolar geometry.

B. Camera Calibration

Camera calibration produces parameters that will be used for the image rectification. These parameters consist of intrinsic and extrinsic matrices. Intrinsic matrix contains the information of the property of the camera and the relationship between the camera coordinate and image coordinate. Extrinsic matrix describes the relationship between the world coordinate and camera coordinate. Generating the distortion parameters and doing stereo calibration are also required in this step. Camera lens generates two types of distortion, which are radial distortion and tangential distortion. The distortion parameters would minimize the distortion. Stereo calibration is the process of calculating the geometrical relationship between the two cameras in physical world.

C. Disparity Map Generation

Given a point P in the physical world, two projection points appear on both the left image plane and the right image plane. These two projection points are x^l and x^r, and the disparity between these two projection points is called disparity, as shown in Fig. 3. If the observed point is closer to the image planes, the disparity will be smaller. The larger disparity represent that the disparity between the observed point and image planes is longer. Hence, disparity is utilized for determining the relative distance between object and cameras.

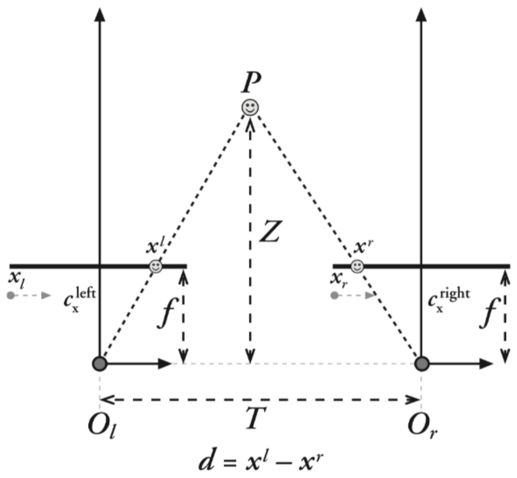


Fig. 3. Disparity map generation.

III. Hardware

A. Airframe

The aircraft used for this project is the Sig Kadet Senior Sport and DJI S900 multicopter, as shown in Fig. 4 and Fig. 5, respectively.

The Sig Kadet Senior was chosen due to its light weight and great stability and maneuverability. The wings are connected by a spar and then hooked into the fuselage. Nylon screws are then placed in the back of the wing to ensure the wing is fully attached and will break off to separate the wings in case of a crash. The tail surfaces and small dihedral angles give the aircraft great maneuverability, so it is easy to fly. The servos control the ailerons, elevators, and rudders. The propeller is attached in the front of the aircraft as seen in Fig. 4 to generate thrust. The Sig Kadet Senior Sport was the ideal option for this project due to its great characteristics.

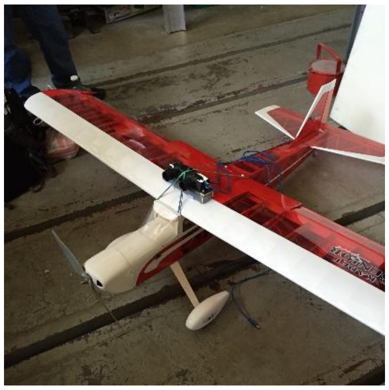


Fig. 4. Sig Kadet Senior Sport UAV.

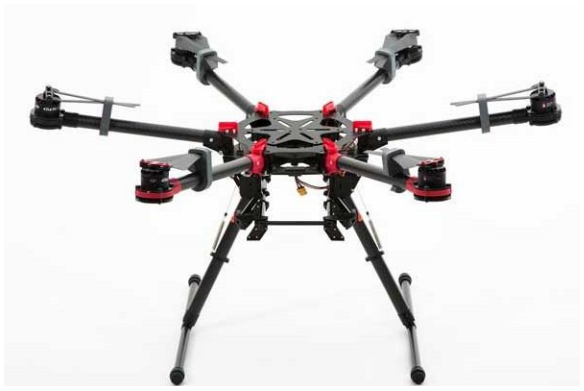


Fig. 5. DJI S900 multicopter.

The DJI 900 multicopter was used as a secondary platform. It has a rotor diameter of 35 inches, and has a maximum take off weight of 18 pounds. This drone was used as backup in case anything went wrong with the Sig Kadet, and also to compare which vehicle would give better disparity maps. The eight propellers attached on the multicopter are capable to carry 10 pounds of payload. By having long arms on the propeller, it increases manueaverability, so the vehicle can move easily in any direction. The long legs give the vehicel great stability and having a T-shaped landing configuration ensures a safe landing so the vehicle and sensors do not get damaged. The Hexacopter was a late addition to the project, but it also displayed great characteristics which was why it was also used for the project. The multicopter seemed to allow the cameras to take better pictures. This was highly due to the placement of the cameras. The cameras on the multicopter were on a flat surface compared to the Sig Kadet where they were placed on top of the wings, where there was some curvature. The cameras on the multicopter seemed more stable compared to the Sig Kadet, which helped reduce movement of the cameras during flight.

B. Flight Computer

The flight computer used is the Intel NUC Skull Canyon NUC6i7KYK kit, which is a powerful mini-computer designed by Intel. This particular NUC has the 6th generation Intel Core i7-6770HU processor. It comes with a 32 GB of DDR4 RAM with two M.2 slot with flexible support for a 42 or 80mm SATA or PCle SSD, four USB 3.0 ports and an Ethernet port. For external peripherals, a simple wireless mouse and keyboard are used. It can be accessed in two ways. The first way is to connect it to an external monitor that supports either Mini-HDMI or HDMI; the second way is to use an external ^{10}/100 MBPS desktop switch to connect to the NUC and a personal computer or laptop using Ethernet cords and then open a Remote Desktop Connection. The cameras are connected into the two USB 3.0 slots in the back.

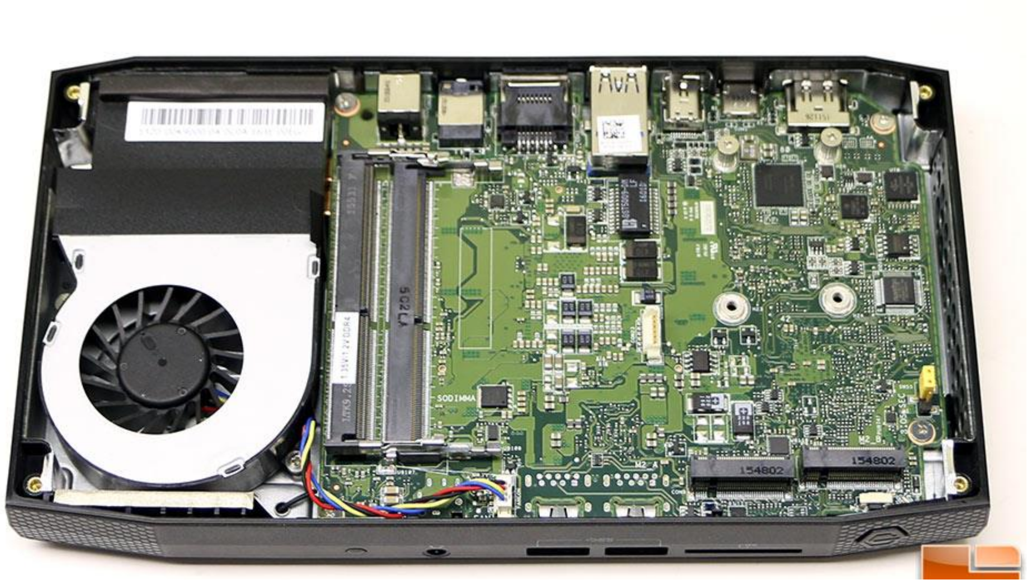


Fig. 6. Intel NUC Skull Canyon NUC6i7KYK kit.

C. Stereo Cameras

For the first half of the project, the cameras of choice were the Logitech HD Webcam C615. Ultimately, these cameras were scrapped because they were of low quality and produced far too much noise. Consumer grade webcams only come with USB 2.0, thus they would all be of too low quality for this project. The project needed light-weight yet powerful USB 3.0 cameras, and after some research the Point Grey Chameleon3 cameras seemed to be a good choice. These cameras can capture color images at 1.3 MP using USB 3.0 with resolutions up to 1280x1024 and up to 149 FPS. The dimensions of these cameras are 44mm x 35mm x 19.5mm with a mass of 54.9 grams. Special software and API are also required for these cameras to function. Fujinon HF25HA-1B lenses were used. These lens, however, must be manually focused for the desired range, and thus are slightly more challenging to properly set up. The USB 3.0 cables used were the ACC-01-2303 USB 3.0 Type-A to Micro-B cable.

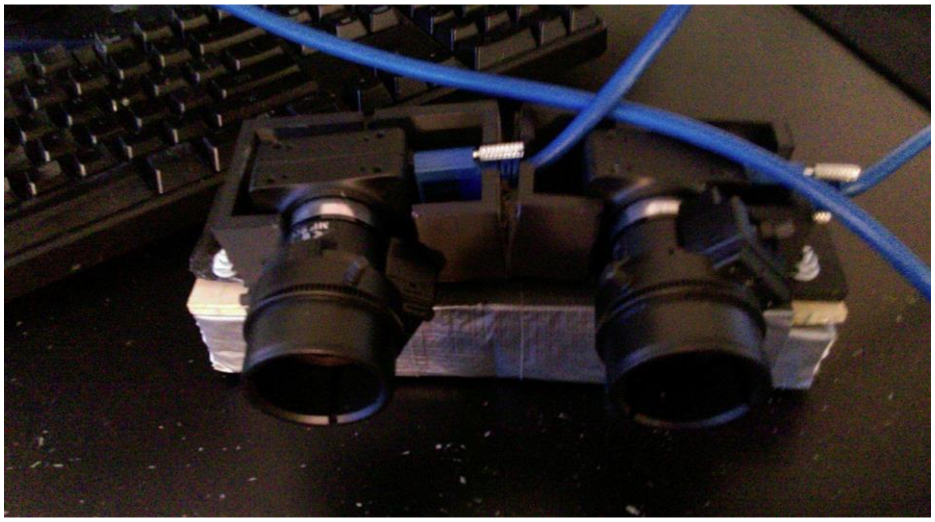


Fig. 7. Point Grey Chameleon cameras for stereo vision.

D. Autopilot

The autopilot that is being utilized for this project is the PixHawk 3DR, which comes with pre-loaded firmware that allows any fixed- or rotary-wing aircraft full autonomous capability. The firmware provides advanced functions such as support for hundreds of three-dimensional waypoints, automatic take-off and landing, as well as sophisticated mission planning and camera controls. It also allows for a compatibility with a variety of Ground Control Station software for programming and mission operations. The PixHawk 3DR can be seen below in Fig. 8.

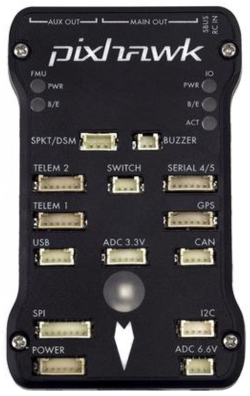


Fig. 8. PixHawk 3DR.

For communication between the autopilot and the ground control station, two XBee radios were used. In addition, a 3DR GPS with magnetometer is connected to the Ardupilot using a 6-pin cable, and the autopilot receives positioning data from the GPS. The autopilot is being powered through the same power source as the NUC onboard, which can communicate with the autopilot. Having the NUC onboard and directly connected to the autopilot enables the aircraft to detect an obstacle and immediately complete the necessary actions to change the current flight path. If the aircraft senses an obstacle on its path, it will use the collision avoidance algorithm to generate a new route to maneuver around the object and continue on its planned path. If no object is sensed in its path, it will continue its route at the same desired elevation and orientation as it was programmed to.

E. Ground Control Station

The APM Mission Planner was chosen as the ground station due to its full featured application already compatible with the Pixhawk.

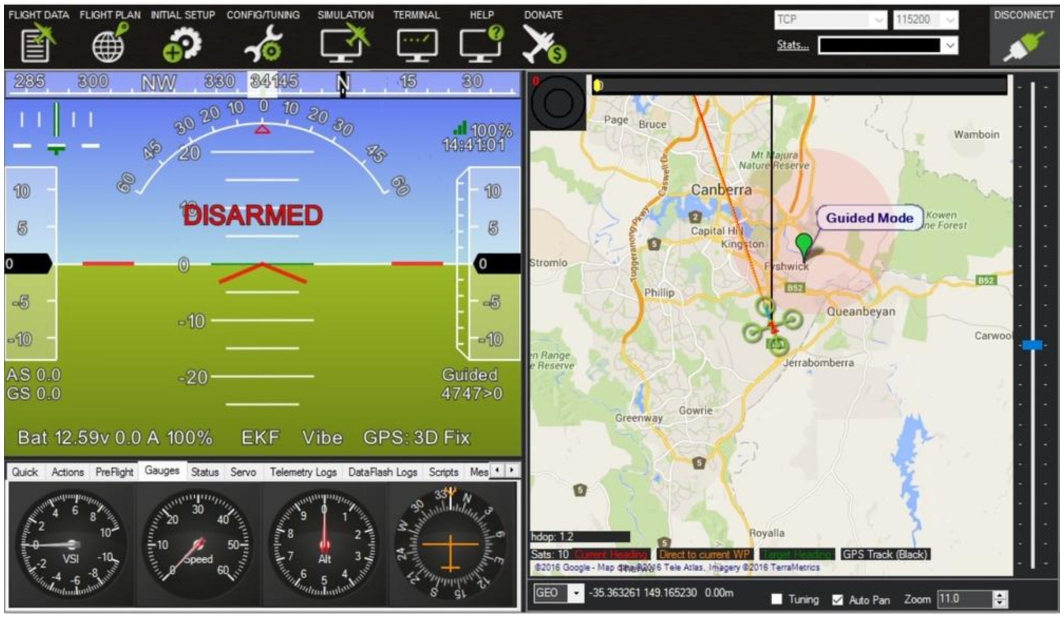


Fig. 9. Mission planner.

Mission Planner can be used as a configuration utility or as a dynamic control supplement for the autonomous aircraft. Once the firmware is configured on the Ardupilot it is possible to setup, configure, and tune the aircraft for optimum and desired performance. It is possible to plan, save, and load autonomous missions into the autopilot straight from the Mission Planner. Figure 9 shows the mission planner.

F. Xbee Communication

To get the XBee radios to work with the Mission Planner, multiple parameters have to be configured on to the two XBees. The configuration was done by using XCTU, a software application designed by Digi International that is used to write parameters onto the XBee. The XBees need to be connected to the computer to be able to implement the parameters on XCTU. The parameters that were changed from the default settings are:

- Baud Rate: 57600
- Pan ID: 155
- Node Identifier: “Ground Station” & “OnBoard”
- Destination Address High:

The XBee with the Node Identifier “OnBoard” will be with the aircraft while the XBee with the Node Identifier “Ground Station” will be connected to the computer that is running the Mission Planner. To be able to get the connection, an XBee explorer and a FMU 4-pin UART port are needed as shown Fig. 10. The “Ground Station” XBee will be connected to the XBee explorer, which is connected with a mini USB cable to the ground station computer. The 4-pin is connected to the “OnBoard” XBee. The 4-pin is connected directly to the Pixhawk to provide the flight data to the “Ground Station” XBee.

When connecting everything on the onboard module, the Pixhawk pin out is connected to the 4-pin with the pin layout for the Pixhawk. This will allow the Pixhawk to power the "OnBoard" XBee for the telemetry to be viable. With the setup complete, the 4-pin will be able to transmit data through the telemetry link.

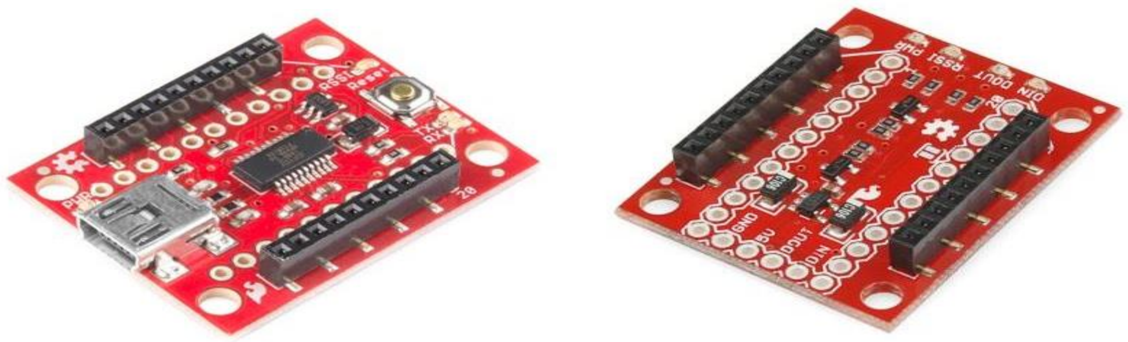


Fig. 10. XBee Explorer (left) & PX4 FMU 4-pin UART port (right).

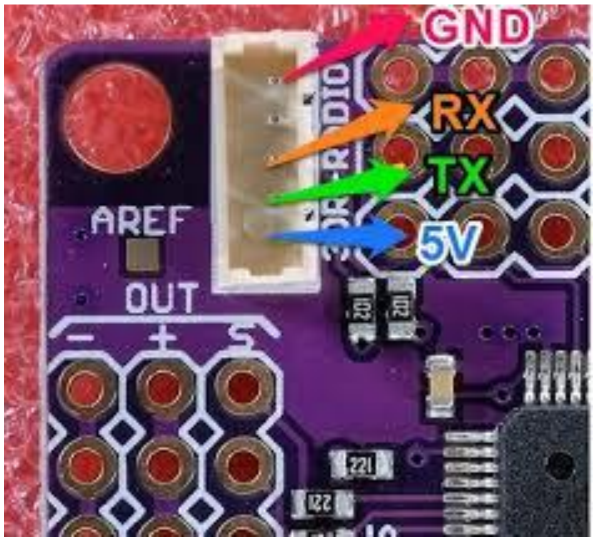


Fig. 11. Pixhawk 2.6 telemtry pin out.

The way the 4-pin is able to communicate with the Pixhawek flight computer is through MAVLink. MAVLink is able to send data packets through the serial channels between the XBees. MAVLink is the main tool used by Mission Planner that allows for the communication between the ground station and the aircraft.

Figure 12 shows the shows the overall hardware architecture. Each component is connected to another using UART or USB connection.

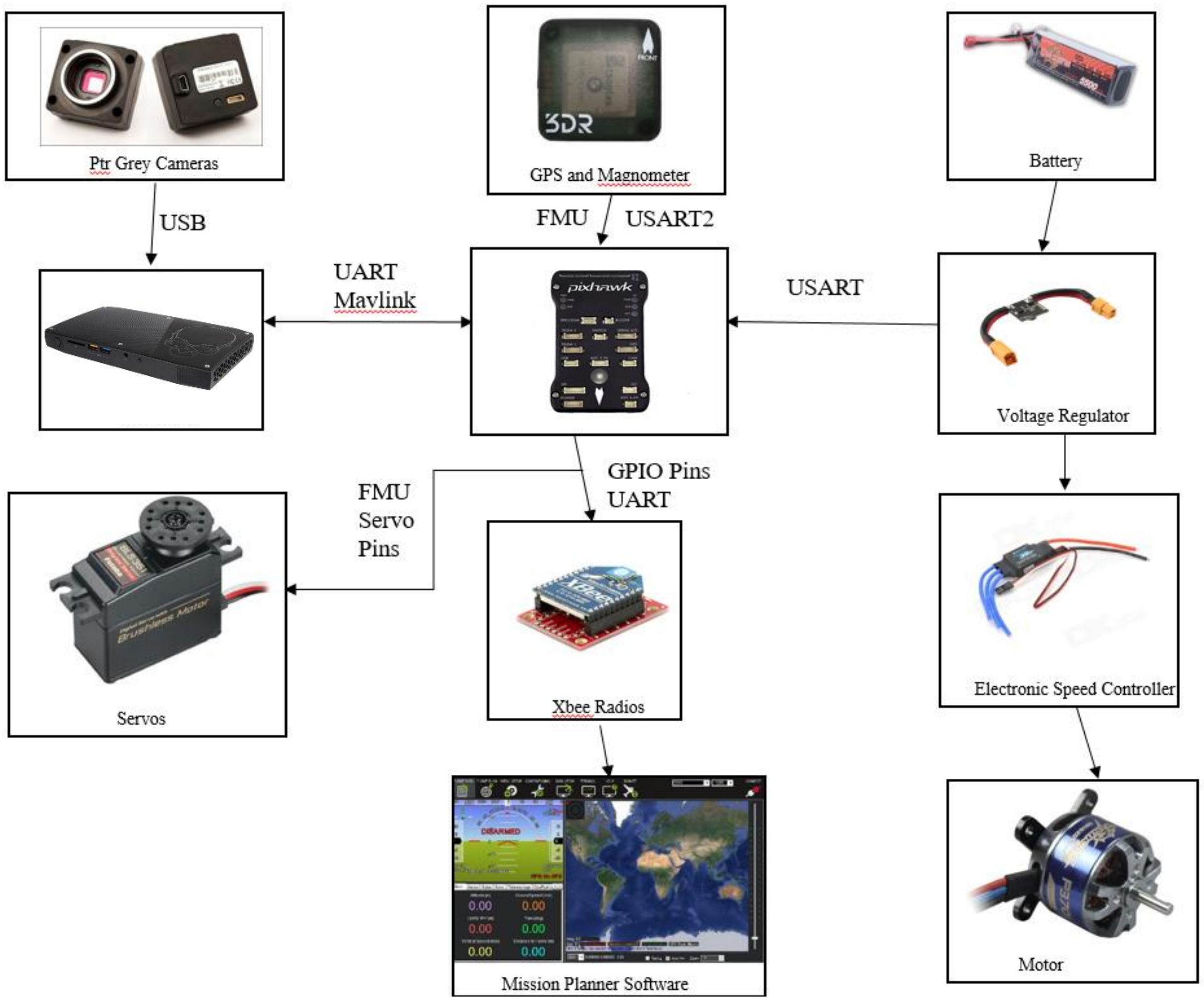


Fig. 12. System Architecture block diagram.

IV. Software

The software include disparity map imaging, collision detection, and collision avoidance. C++ and Open Source Computer Vision (OpenCV) library are used.

A. Camera Calibration and Image Rectification

In order to calibrate each camera with OpenCV, each camera needs to capture a series of chessboard. The series of images include 15 different chessboard directions. The corners of the chessboard are detected and utilized to generate disparity parameters. For stereo calibration, the two cameras need to mount still, and the two cameras capture a series of chessboard images simultaneously.

After OpenCV calculates all the parameters, the prerequisite of image rectification is reached. The set of images are rectified by using the parameters obtained by camera and stereo calibration.

B. Disparity Map Generation

A pixel, from left image as a reference point, a corresponding point, a pixel, on the right image is found, and the distance between the two pixels is calculated. Repeating the process over the entire image produces the disparity map. Figures 13 and 14 are two images captured by left camera and right camera, respectively, for example.

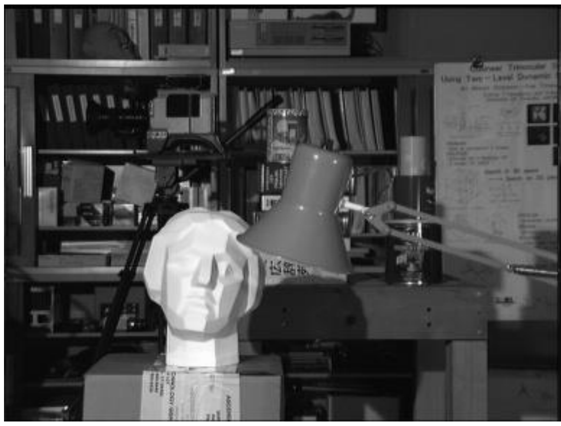


Figure 13. Left image.

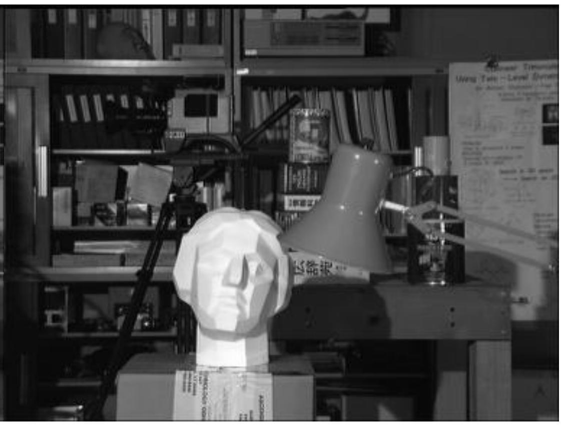


Figure 14. Right image.

Figure 15 shows the blended image of Fig. 13 and Fig. 14. Comparing the box corner, in Fig. 15, shown in the left bottom red circle with the white board angle in the top right red circle, it is seen that the shifted distance of box corner is larger than the shifted distance of white board corner. It represents that the box is closer to the camera than the white board. Using this technique, OpenCV calculates the disparity map.

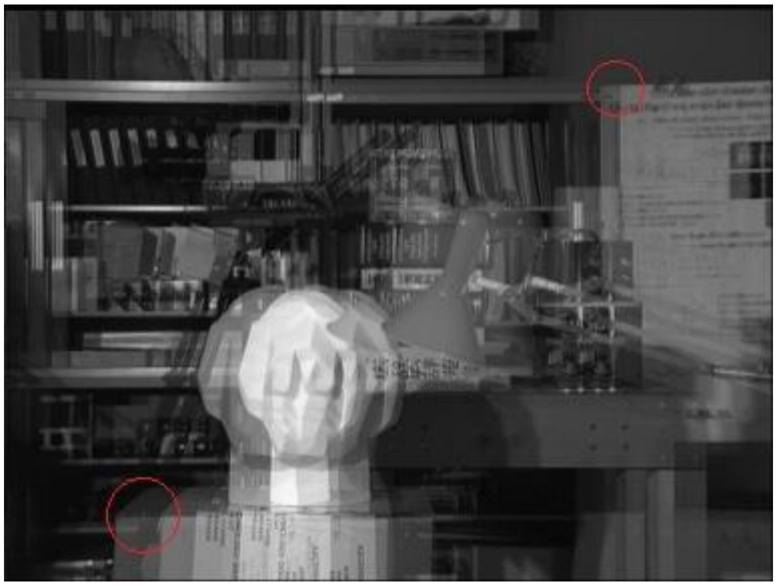


Fig. 15. Blended image.

In order to simplify the computational difficulty, the images are converted to gray scale images. The gray image is scaled from 0 to 255. The longer of the pixel shifts, the whiter the object shows on the disparity map. Therefore, if the object is closer to the camera shot, it will look whiter. As shown in Fig. 13 and Fig. 14, because the table light is closer to the camera shot and video recorder is farther, the table light is whiter than the video recorder, as shown in

Fig. 16. The process of finding corresponding point is called matching problem. Because the pair of images is rectified, the searching direction is only on the horizontal pixel row. The approach OpenCV uses is Sum of Absolute Difference (SAD). It compares two image boxes of left and right images by calculating the similarity between them. The center of the image boxes are reference pixel of the left image and target pixel of the right image. Choosing the most similarity boxes, the distance of the centers would be the value of the disparity.

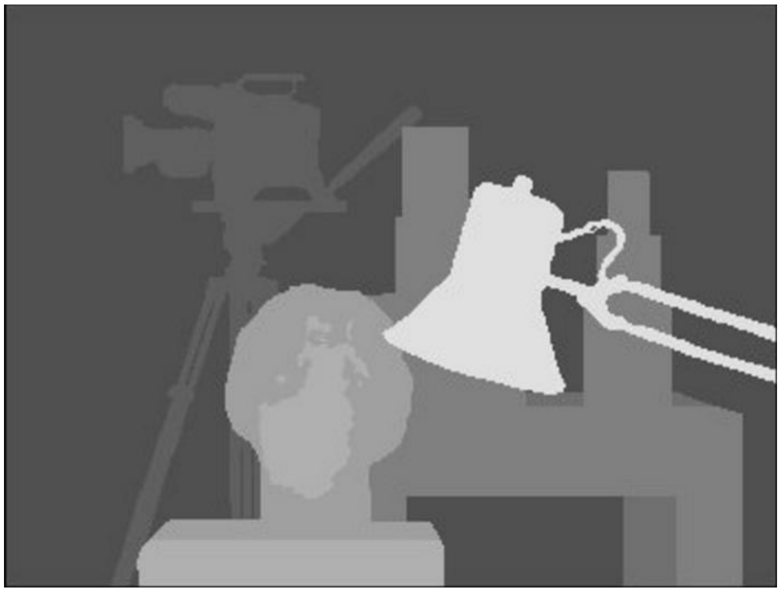


Fig. 16. Ground truth disparity map.

C. Intel NUC and Camera API

The Intel NUC is running Microsoft Windows 7 as an operating system. The IDE, Integrated Development Environment, of choice was Microsoft Visual Studio 2012 Express. The two APIs used are OpenCV and FlyCapture2. The verison of the OpenCV used is Open CV 3.1.0, which has improved algorithms for Stereoscopic Vision and Calibration. The Chameleon3 camera requires the special API called FlyCapture, which was developed by PointGrey. This API is included, for free, within the FlyCapture2 SDK, which includes drivers for the cameras, source code, examples, and the libraries required for the cameras.

D. Collision Avoidance

As discussed above, the algorithm requires the development of a disparity map, which utilizes two cameras to measure field depth and provide the necessary information required, such as distance from the UAS and velocity.

The collision avoidance process that is used in the algorithm and is broken into three sections: sense, detect, and avoid.[6] The sensing phase is the first portion of the collision avoidance program. This involves the development of the disparity map which uses the camera systems to identify objects in front of the UAS and assigns a high pixel density of that object in order to denote the distance from the UAS. After the sensing phase comes the detection phase, where the possible threats to collision are categorized based on the path of the UAS and the speed, direction, and current location of the object. If one or more of the detected object are deemed as a collision threat, the avoidance phase is initiated. This calculates the safest path to travel based on the least pixel dense region of the disparity map. This will be continually updated as the path planning continues with new stereo data to constantly update the new path.[6]

The collision avoidance algorithm works by developing a nine-quadrant layout of the disparity map.[5] Each quadrant is analyzed by recording the pixel intensity and comparing them to one another. When a collision threat is determined, the algorithm will select the quadrant that has the least pixel density, which signifies the least possible chance of object collision.[7] When the quadrant is selected, the UAS will execute a maneuver by flying toward the selected quadrant. The disparity map and quadrant layout can be seen in Fig. 16. The new heading and altitude were then determined using existing flight data and the angle of inclination to the center of the quadrant of the disparity map. Once the new heading was determined, the algorithm sets a GPS waypoint that the UAS will follow for a set distance to clear it of the obstacle.

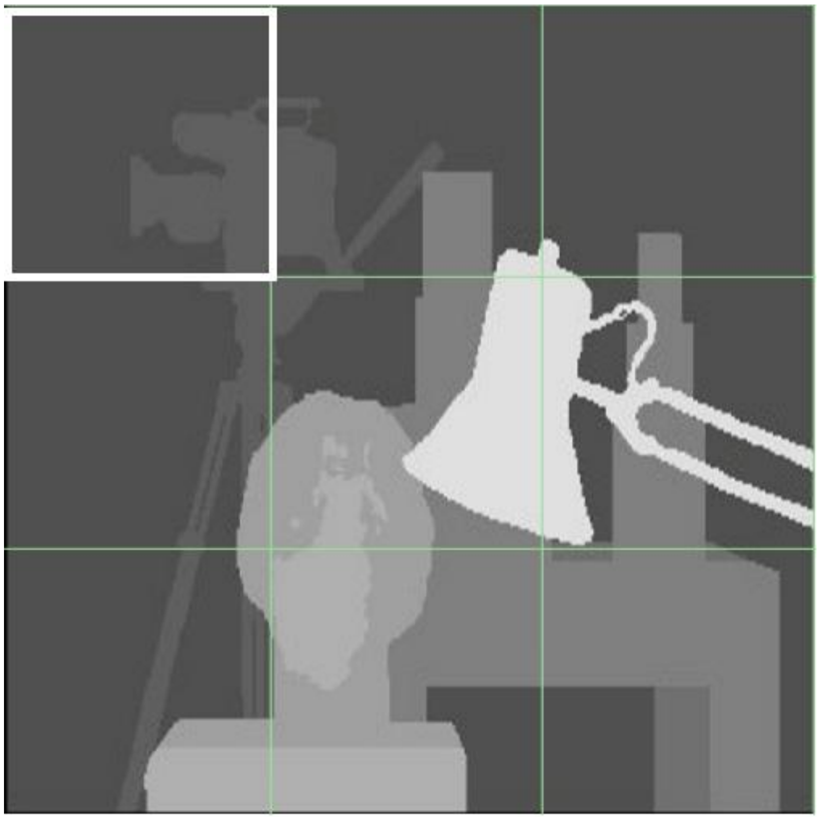
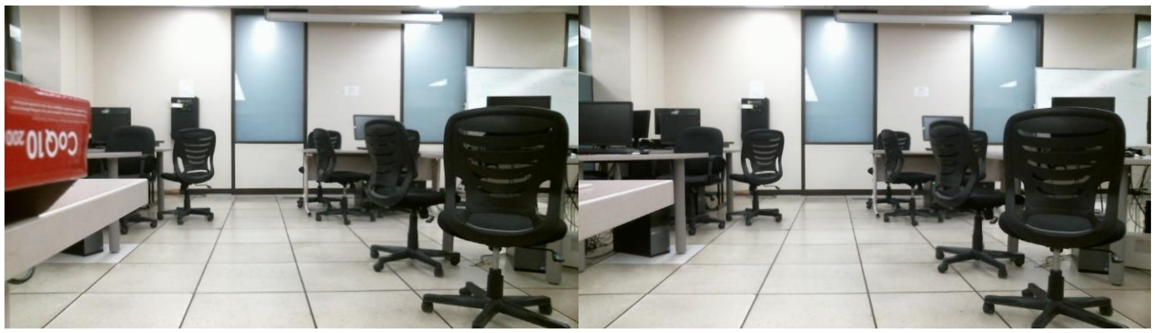


Fig. 17. Grid generation with selected quadrant.

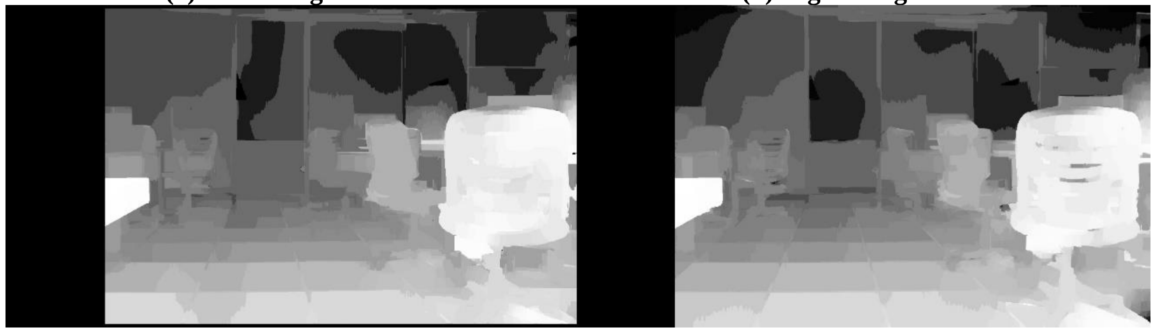
V. Results and Discussion

A. Ground and Static Tests

The project uses two algorithms to generate disparity map, which are block matching (BM) and semi-global block matching (SGBM). The block matching requires less time to produce a disparity map, but semi-global block matching yields more accurate disparity map. For the searching box, block matching uses 3 pixels by 3 pixels, and semi-global block matching uses 7 pixels by 7 pixels. Block matching generates one disparity map in 0.093 second, and semi-global block matching requires 0.18 to produce a disparity map. As shown in Fig. 18 (c) and (d), the disparity map generated by semi-global block matching provides more details. OpenCV provides a function to refine the disparity map. It reduces noise and smoothes the disparity map that yields a more accurate disparity map.[8]

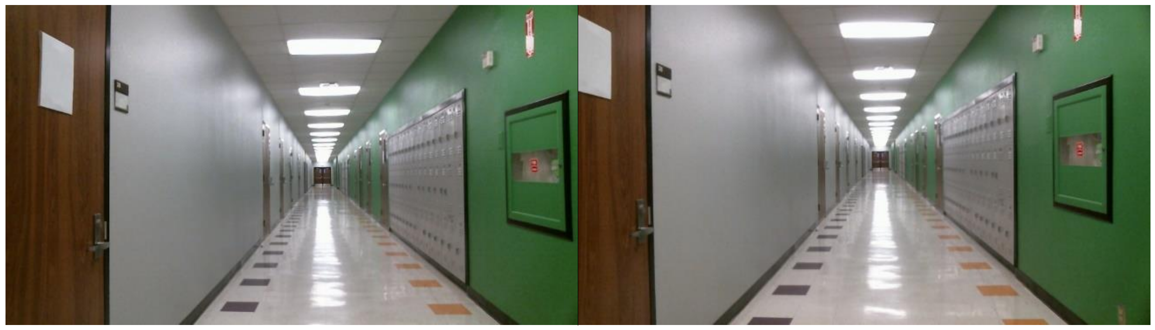


(a) Left image **(b) Right image**

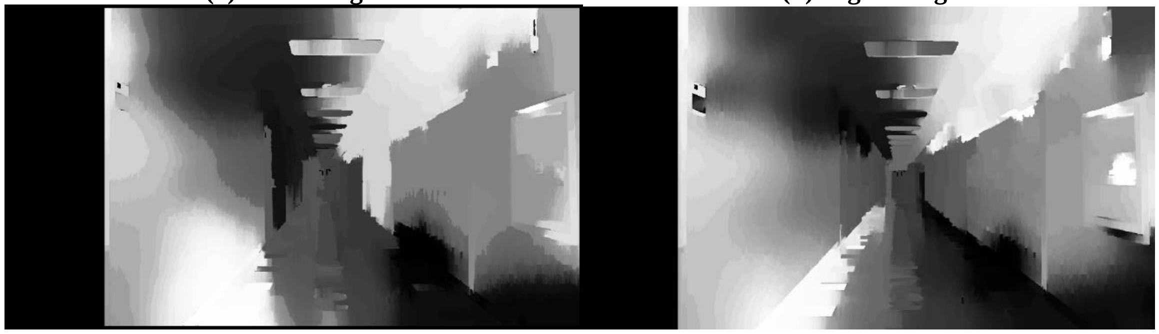


(c) BM disparity map **(d) SGBM disparity map**

Fig. 18. BM and SGBM disparity maps generated of picture 1.



(a) Left image **(b) Right image**



(b) BM disparity map **(d) SGBM disparity map**

Fig. 19. BM and SGBM disparity maps generated of picture 2.

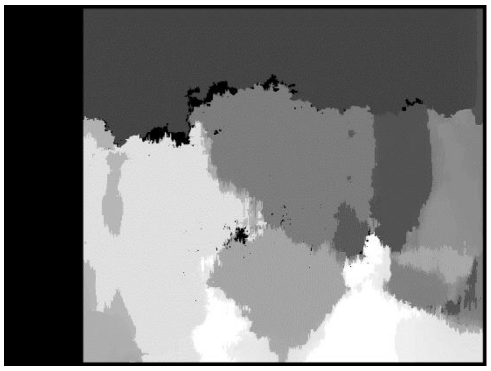
Comparing the set of Fig. 18 (a) and (b) with Fig. 19 (a) and (b), 18 (a) and (b) are captured by an aligned stereo rig, but Fig. 19 (a) and (b) are captured by a unaligned stereo rig. The aligned stereo rig produces better disparity map. It shows more accurate relative depth and without much noise.

However, some conditions cause the error on the disparity map, such as texture-less areas, half-occlusions, and regions near depth discontinuities. OpenCV provides a function, called filter function, to minimize the error, and it requires two disparity maps from a set of left and right images. The first disparity map is same as usual, using left image as reference and searching on the right image. And the second disparity map is using right image as reference image and searching the left image. The filter function compares these two different disparity maps and minimizes the errors. As seen, aligned stereo rig provides a good disparity map, so the future work would be reduce the time of producing disparity map. However, if the unaligned stereo rig is being used, an algorithm that has both features of fast and accuracy need to be developed and applied on the stereo system.

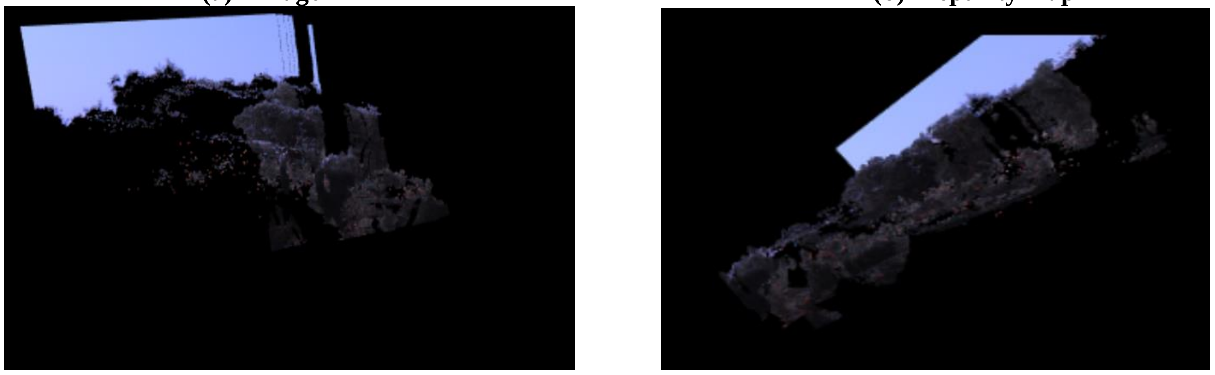
B. Flight Test

Due to API problems from the PointGrey Cameras, the images were recorded for post processing. The cameras were placed on top of the wing about 10 inches away from each other. Figure 20 (a) shows the image that was taken during flight. Figure 20 (b) shows the disparity map and Fig. 20 (c) shows the 3-D view of the image. The image picked was the one with the most depth and it was the most clear. The day was particularly clear so the effectiveness of the algorithm was of high quality. As can be seen in the figure, the disparity map is clear, although the depth of the image is not very far. The images could be processed around 20 disparity maps per second, but the calibration of the cameras are preventing the better quality disparity map generation.

(a) Image



(b) Disparity map



(c) 3-D view of image

Fig. 20. Disparity map and 3-D view of picture.

The image seems to only generate disparity maps about 40 feet in front of the aircraft. The image was the best one that was captured by the cameras. However, it can be interpreted from the pictures in the figure that the disparity map is still slightly incorrect. In Fig. 20 (a), the tree to the bottom of the image is actually the closest to the cameras which can be seen in Fig. 20 (c) by the 3-D view, but in Fig. 20 (b), the disparity map showcases the tree as light grey which means it is not as close compared to the tree parts that are in white. This will create a problem for the avoidance algorithm because of the bad quadrant picking. The code does need to be optimized for speed, or processing, and to be able to read further depth so that collision avoidance maneuver can be executed at least 4 seconds before the collision. The moving obstacles such as other aircraft must be detected faster. However, for slow moving vehicles such as the multicopters, this seems to be sufficient.

VI. Conclusion and Future Work

The quality of disparity map has improved with the use of the new Intel NUC. Flight test results show that the system can detect objects approximately 40 feet in front of the aircraft. A problem that was run into was the stereo calibrations were not properly aligned so the code would give improper readings for the disparity maps. This will need to be improved so that the avoidance code can get correct data from the map on the physical world.

Future work will involve fast image processing techniques so that disparity maps can be generated constantly for accurate readings. The application of the Census transform may be used as well for to be able to speed up the processing of the stereo matching since Census masks are large and sparse and perform with the same processing as small dense masks.[9]

Future work will also involve navigation using blob detection or the morphological filtering approach for obstacle velocity estimation.[10] This method will set the waypoint based on velocity of incoming obstacle (identified using blob detection algorithm and rangefinding method),[11] and selecting optimal path from multiple low-cost regions of the disparity map.

A method proposed is using a third camera to increase the accuracy of the disparity maps while using block-based and wireframe modeling techniques.[12] In turn, using the block-based imaging technique can create a byproduct for compensating 2-D and 3-D motion as well, which will improve the onboard processing to the autopilot.[12]

References

[1]Schaeffer, R., "A Standards-Based Approach to Sense-and-Avoid Technology," *AIAA 3rd Unmanned Unlimited Technical Conference, Workshop and Exhibit*, September, 2004.

[2]Saneyyoshi, K., "Stereo Vision System on Automobile for Collision Avdoiance," *IAPR Conference on Machine Vision Applications*, June, 2011.

[3]Blin, R., Bhandari, S. et al. "Obstacle Avoidance System for UAVs using Computer Vision." *Proceedings of AIAA Invotech@Aerospace Conference*, FL, 5-9 January, 2015.

[4]Park, J., Kim, Y., "Obstacle Detection and Collision Avoidance of Quadrotor UAV Using Depth Map of Stereo Vision," *AIAA Guidance, Navigation, and Control (GNC) Conference*, 2013.

[5]Bhandari, S., Srinivasan, T., Gray, J., Torstenbo, M., Corral, J., Brown, N., Mcdorman, N., Wood, J., and Raheja, A., "Collision Avoidance System using Stereoscopic Vision for Unmanned Aerial Systems," *AIAA Infotech @ Aerospace*, 5-9 January, 2016.

[6]Lacher, A. R., Maroney, D. R., and Zeitlin, A. D., "Unmanned Aircraft Collision Avoidance - Technology Assessment and Evaluation Methods," *The MITRE Corporation*, 2007.

[7]Hrabar, S., "3D Path Planning and Stereo-based Obstacle Avoidance for Rotorcraft UAVs," *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2008.

[8]Solh, M., and Alregib, G., "Hierarchical Hole-Filling for Depth-Based View Synthesis in FTV and 3D Video." *IEEE Journal of Selected Topics in Signal Processing IEEE*, Vol. 6, No. 5, pp. 495-504, June 2012.

[8]Humenberger, M., Zinner, C., Weber, M., Kubinger, W., and Vincze, M., "A Fast Stereo Matching Algorithm Suitable for Embedded Real-time Systems," *Computer Vision and Image Understanding*, Vol. 114, No. 11, pp. 1180-1202, November 2010.

[10]Carnie, R. J.,Walker, R. A., and Corke, P. I., "Computer Vision Based Collision Avoidance for UAVs." *Proceedings of 11th Australian International Aerospace Congress*, Melbourne, Australia, 2005.

[11]Lai, X., Wang, H., and Xu, Y., "A Real-time Range Finding System with Binocular Stereo Vision," *International Journal of Advanced Robotic System*, 2012.

[12]Tzovaras, D., Grammalidis, N., and Strintzis, M. G., "Disparity Field and Depth Map Coding for Multiview 3D Image Generation." *Signal Processing: Image Communication*, Vol. 11, No. 3, pp. 205-30, 1998.