UAV Collision Avoidance with Stereo Vision on a Low-Power Embedded System

Technical Report #CSSE17-07

Andrew Freeman†
Samford University
afreema4@samford.edu

Daniel Suen†
University of Florida
dsuen1@ufl.edu

Dr. Richard Chapman
Auburn University
chapmro@auburn.edu

Dr. Saâd Biaz
Auburn University
biaz@auburn.edu

Abstract—This paper presents a computer vision based system to perform real-time collision avoidance for unmanned aerial vehicles (UAVs). The system illustrates that collision avoidance can be achieved by implementing computer vision algorithms on an affordable and lightweight processor such as a Raspberry Pi. Many current techniques either use more powerful processors or offload the data to an external processor for the computation. A robust method using disparity mapping is introduced with real-time processing done onboard the UAV. This work is based partially off of the open-source OpenCV library and Daniel Lee’s StereoVision library [1]. It runs on a Raspberry Pi 2 Model B with two Logitech C615 cameras.

I. INTRODUCTION

The use of unmanned aerial vehicles (UAVs) has become increasingly popular in both civilian and military applications. Autonomous aircraft provide a means of reducing risk for human pilots and can perform a variety of tasks effectively. For instance, UAVs can be used for reconnaissance, collecting data for agriculture, and transporting goods. To accomplish this, UAVs must be able to detect and avoid collisions with high accuracy. This paper proposes an implementation of a real-time collision avoidance system solely using computer vision techniques on a Raspberry Pi system.

While a wide variety of active and passive sensors have been used for collision detection, they all have their disadvantages. For one, the carrying capacity of UAVs is limited, and overloading a UAV with an array of sensors can negatively affect maneuverability and flight time. Active sensors such as RADAR and LIDAR have excellent accuracy and are dependable in a variety of weather conditions [2]. Unfortunately, they are expensive and require existing infrastructure and emit electromagnetic radiation, leaving them prone to detection. On the other hand, passive sensors such as forward looking infrared are smaller, demand less resources, and do not emit radiation. But they require expensive image processing, which is a severe limitation given the computational power of a Raspberry Pi [3].

Lightweight cameras can overcome these problems, as they are relatively inexpensive, easy to implement, and less susceptible to malfunction due to external factors. Additionally, this method allows the UAV to be fully self-reliant with all the processing done onboard and in real time. As such, navigating using computer vision is more advantageous.

Global Positioning System (GPS) implementations of UAV guidance are widespread, but there are situations where obtaining accurate GPS location information may be impossible [4]: if a UAV is flying near enemy territory, for example, the enemy could employ a GPS jamming technology, preventing the UAV from determining its location [5]. If GPS is required for the UAV’s functional operation, jamming the signal would cause the autonomous system to fail, and the UAV could be irrecoverable. In addition, GPS does not work for indoor movement as buildings and man-made infrastructure can attenuate signals.

A vision system can be used as an alternative to or alongside GPS. During regular operations, vision components can augment a GPS signal, offering a more accurate location reading [6]. In this case, vision would also be a failsafe in case the GPS signal is jammed, ensuring that the UAV does not crash. Vision systems

†These authors contributed equally and are listed alphabetically.
can also serve as a UAV’s sole source of navigational information. A vision system void of GPS would allow a UAV to safely fly in indoor and urban environments.

II. PROBLEM STATEMENT

Computer vision is appropriate for an indoor scenario, where obstacles are numerous and airspace is limited. In such scenarios, traditional object detection techniques are unreliable or impractical. While full-sized UAV quadcopters are restricted in their indoor flying ability due to their sheer size, so-called “micro-UAVs” are much smaller and are better suited for indoor flight.

In this paper, a method is proposed for using stereo vision to see and avoid any obstacles within a certain distance of a UAV in an indoor scenario. The system is very robust, and the UAV can avoid any and all objects within its flight path. This research aims to improve upon past machine-learning-focused research in stereo vision by relying instead on disparity mapping to detect any object within a predefined distance. Using disparity mapping creates a generalized algorithm that can work in an environment with many barriers and obstructions.

This research aims to build on existing obstacle avoidance work with disparity mapping by applying the concepts to micro-UAVs. Whereas past research in this area has relied on either offloading computation to a ground computer or else carrying powerful hardware onboard the UAV, this paper explores real-time disparity mapping onboard a micro-UAV with limited computational power. In addition, the method outlined in this paper is cost-effective with little overhead in implementation.

This system includes a Raspberry Pi 2 Model B computer, two Logitech C615 cameras, and a Parrot AR.Drone 2.0. All image processing will be performed in real time on the Pi, using the Open Computer Vision (OpenCV) in Python. Python was selected as it is a high-level language with broad support and functionality. The UAV will be controlled with the PS-Drone API provided by J. Philipp de Graaff [7]. To define the scope of the problem, it is assumed that the UAV can make evasive maneuvers only in the horizontal plane, and that all obstacles encountered will be in this plane. Stereo camera calibration and image rectification will be performed in Python using the StereoVision library written by Daniel Lee [1].

III. LITERATURE REVIEW

Much work towards developing better computer vision techniques and algorithms for working with UAVs has already been completed. A considerable amount of the research is dedicated to the areas of object detection and object avoidance. Many procedures for visual collision avoidance require a way of finding and possibly tracking objects that are on a collision path with the UAV and then, moving the UAV to avoid impacting that object.

A. Visual Systems

Visual approaches for object detection use either monocular or stereo vision, and with the goal of obtaining a passive vision system, both have become well-explored areas in the literature. While monocular systems can provide high-accuracy measurements, depth is more difficult to calculate using only one camera. To determine depth, the camera must be in constant motion, so that it can take pictures from multiple perspectives, which is not always the case with a quadcopter UAV.

Another approach for monocular systems is to use optical flow for object tracking. Optical flow is limited for this project as it cannot easily determine change in depth, only change in horizontal or vertical movement across the camera screen. Also, optical flow can have difficulty separating objects from the background when there are multiple objects to be avoided [8]. Additionally, monocular algorithms such as those that are machine-learning based or are probabilistic in nature are computationally expensive and, therefore, ill-suited for the low-power embedded system this paper requires [9].

Fig. 1. Diagram of stereo vision setup. X denotes the location of the object, O and O’ are the location of the cameras, and f is the focal length [10].

On the contrary, a binocular system emulates the way human eyes work by having two cameras working together to achieve depth perception. The stereo cameras are placed in the same plane, and their images overlap. The distance to any object can be calculated by measuring the difference between the location of an object as it appears in the two images. By knowing the focal length of the cameras and the distance between
the two lenses, this calculation can be accomplished by using similar triangles. The closer an object is to the cameras, the greater the change in its location in the two images. For an embedded system, stereo systems are more advantageous than monocular systems as they require simpler models and algorithms.

B. Stereo Vision Camera Calibration

The cameras must be calibrated in a stereo vision system. Calibration properties are commonly calculated by taking a series of images showing an image of a chessboard pattern. Based on the calibration, each camera’s intrinsic properties and the extrinsic properties between the cameras, determined by how far apart they are and their orientation with respect to each other, are taken into account [11].

A lens may exhibit pincushion or barrel distortions, which make straight objects appear bent or curved in an image. Two cameras of the exact same model may require different calibration parameters due to imperfections in their respective lens elements and differences in their distortion severity [12].

![Fig. 2. Left picture showing barrel distortion of the chessboard. Right picture of the rectified image [11].](image)

When a vision program runs, any image taken by a camera must be rectified using the calibration files before any disparity calculations can be done. The rectification process corrects for any distortion caused by the cameras’ lenses through a series of linear transformations. The stereo matching algorithms need to be able to solve the correspondence problem, in which the pixels in one image are determined to correspond to the appropriate pixels in another. Properly rectifying a pair of images can improve the accuracy of these algorithms. Image rectification processes the images, so that the pair are aligned with one another; in doing so, the algorithm can search in one dimension (along the left-right direction) and determine which pixels correspond in one row at a time.

For this paper, it was determined that proper stereo calibration was required. As opposed to the built-in OpenCV methods for calibration, the StereoVision wrapper library [1] was used due to its simplicity and Python support.

C. Vision-Based Object Detection

Machine learning algorithms that rely on specific classifier-based methods are not optimal for this scenario. For one, they require a great amount of training data (consisting of thousands positive and negative examples) to determine what the target is. This is not ideal as it requires prior knowledge of the environment and time to collect the image data and train the program [13].

Even so, this method is not responsive to small changes in objects. This approach is designed to be more general and avoids this problem by focusing on any object that is close by and avoiding it. If a UAV is trained only to avoid pre-specified objects, there is a high chance of collision with unfamiliar objects, especially when flying in scenarios where foreign objects are varied and numerous. For these reasons, a machine learning approach was not chosen for this paper.

D. Disparity Mapping

By comparing two images taken simultaneously via a stereo camera system, one can reconstruct a representation of the three-dimensional structure of the environment.

OpenCV provides two different disparity mapping algorithms: Stereo Block Matching (StereoBM) and Stereo Semi-Global Block Matching (StereoSGBM) [14]. StereoSGBM creates more accurate disparity maps, but is much slower. The StereoSGBM algorithm adds on to the StereoBM algorithm by incorporating a global energy function, which it wants to minimize. Therefore, StereoSGBM performs both locally and globally, which slows its runtime speed [15].

![Fig. 3. Example disparity map generated using StereoBM in OpenCV [10].](image)

On the low-power embedded system used in this paper, StereoBM is a better solution since the computational overhead of StereoSGBM makes real-time image processing difficult. Both matching algorithms use a
sum of absolute differences (SAD) of pixel intensity comparison. It works by comparing a block of pixels in one image to a series of blocks of pixels in another image. If a match is found, the difference in location between those blocks in the two images is calculated. The window with lowest SAD value possible is selected [16].

A larger SAD window size creates a smoother disparity map but increases the computational time. A second parameter is the number of disparities, which is inversely proportional to distance to the object. Therefore, to locate objects that are closer to the camera, a larger number of disparities is needed and vice versa; it is necessary to strike a balance, so that one can locate the objects within a reasonable range of distances [16]. For this paper, a low SAD window size was chosen, to increase speed.

Other work with stereo vision often involves image processing with stationary cameras. By having the cameras fixed, it is possible to subtract the background from the image, isolating only objects in the foreground [17]. Unfortunately, this is not useful for UAV travel since the cameras are always moving, so the background is always changing. The computer has difficulty picking out specific features from a variable background. Thus, for this paper, background subtraction was not performed.

Byrne et al. use image segmentation techniques to reduce the correspondence errors between images in a disparity map. Image segmentation using the minimum cuts algorithm from graph theory can be used to determine which regions in the image are not smoothly varying. Each region is treated as an object and, in coordination with depth information, the system can ignore areas that span a wide range of distances, such as sky and ground areas [3]. While a segmentation approach can greatly help with disparity map accuracy, segmentation was not used in this paper due to its computational overhead and the complexity of indoor scenes.

E. Thresholding

![Example of a binary threshold](image)

Fig. 4. Example of a binary threshold, where all non-black values are made white [18].

A raw disparity map can be processed more easily by turning it into a binary image. A number of thresholding functions are supported by OpenCV, including one that can create binary images or those that can simply restrict the data to a specific range. In particular, the threshold can be specified such that anything below a number $N$ is assigned a 0 and anything above $N$ is assigned a 1 [19]. For this project, it was determined that such a binary threshold is optimal.

Convolutions, which consist of performing local window summations, are time-consuming and can end up performing redundant calculations. Thresholding removes the need to perform any convolutions or Laplacian transformations to detect high frequency components [20]. Binary images have more explicit and well-defined edges, removing the need to run any highpass filters across the image for edge detection. Thus, algorithms suited for contour detection are much more successful when the image is given in binary.

As such, thresholding was found to be useful for this project since it improved the object detection algorithm and created more precise contour detection.

F. Occlusion

Occlusion occurs when no correspondence can be found between a block in stereo images. On a disparity map image generated in OpenCV, occluded areas appear as black. Occlusion can be reduced by using StereoSGBM over StereoBM by adjusting the block matching algorithm parameters, or by using more than two cameras [21]. For this paper’s application, a small amount of occlusion is acceptable, as long as the disparity mapping does not generate “false negative” results, failing to sense a nearby obstacle that the UAV should avoid.

Occlusion also occurs when another object obstructs a cameras view of an object. Predictive algorithms have been used to help alleviate this effect [22], but occlusion avoidance is outside the scope of this paper.

G. Morphology

Morphology is the practice of morphing an image to remove unwanted noise and make the image easier to process. OpenCV has several built-in morphological functions that can be performed on binary images. A combination of erosion and dilation is useful for eliminating speckle noise that can be found in the binary image [24].

Morphology was tested in this project, but determined to greatly reduce runtime speed. To increase disparity mapping accuracy without hurting performance, bounding rectangles were used instead.
H. Bounding Rectangles

Contours are a way of encapsulating a desired object with a specific shape. OpenCV has several functions dedicated to fitting rectangles, circles, ellipses, and generalized convex hulls [25].

In addition, the shoelace formula can compute the area of any simple polygon given the coordinates of its vertices. Suppose that a detected object is surrounded by a bounding rectangle. It follows that the size of the object in question can be reasonably approximated by the size of the rectangle, which is calculated in $O(1)$ time. By surrounding a detected object with a bounding box, one can have a better understanding of the object, estimating its position and size by using the position and size of the bounding box. Hence, bounding rectangles were selected for the method in this paper due to their simple approach and efficient implementation.

I. Obstacle Avoidance

Once a segment of the image frame has been determined to be an object, the system must decide if that object poses a collision threat to the vehicle. If a collision is foreseen, the vehicle should maneuver to avoid the collision.

CAMshift is a generalization of the meanshift algorithm, allowing for changes in object size and orientation. Unfortunately, this method is limited to tracking a specific color histogram, so it cannot be used to follow objects that are of different colors. This was determined to be unnecessary for purposes of this project, as there is no need to monitor object movement across the screen [26].

Matthies et al. demonstrate the use of the CL-RRT path-planning algorithm to guide a stereo-vision UAV in a highly-textured outdoor environment [27]. Such path planning was outside the scope of this paper, as the UAV was not given a specific destination.

Hrabar shows the use of stereo cameras to build a digital roadmap of the area and plan a safe path for the UAV. This method of obstacle avoidance is very slow, however, as the average time to generate the roadmap and plan the first safe path is 22.4s [28]. Since this path planning is not performed in real time, such an implementation would not work for this work.

“See and avoid” systems function by simply moving the vehicle until the collision threat is no longer in sight of the vehicles sensors [29], but they are the least robust. A “see and avoid” maneuver can be evasive, where the UAV will only have limited time and space to move to avoid a collision [30]. Since these algorithms are extremely fast compared to predictive path-planning algorithms, and can be performed in real time, a “see and avoid” approach was used for this paper.

Existing Python modules can be used to interact with and control a UAV. For this project, the PS-Drone library will be used on the Raspberry Pi to interface with the Parrot AR.Drone 2.0 UAV.

IV. System Description

The system implemented on the Raspberry Pi 2 consists of real-time disparity maps, obstacle detection using contours, and a “see and avoid” algorithm. A pair of identical Logitech C615 Webcams is mounted on a Parrot AR.Drone 2.0 approximately three inches apart. OpenCV is compiled with OpenMP and TBB enabled to utilize multiple processor cores during runtime. The Raspberry Pi 2 is attached onboard and held securely within the plastic hull; a picture of the final setup is provided in Figure 6. A block diagram outlining the main processing steps of the system is provided in Figure 7.

A. Stereo Camera Calibration

The stereo camera system is calibrated with Daniel Lee’s StereoVision utility [1]. A pair of images from
the left and right cameras is captured when the program detects a checkerboard in the frame. A 9x6 square checkerboard pattern was used, and 50 pairs of calibration images were collected at the resolution 640x480.

In the obstacle avoidance system, pairs of images are again captured simultaneously from the two cameras at the same resolution of 640x480. The images are then rectified based on the calibration parameters generated by the StereoVision tool.

B. Disparity Mapping Using Stereo Vision

A disparity map is generated from the rectified images, using Daniel Lee’s StereoVision library [1], which is based on OpenCV’s StereoBM. The disparity map is a normalized grayscale image, taking continuous values between 0 and 1. Closer detected objects in the image have values closer to 1, and further detected objects have values closer to 0. The former are shown as nearly white while the latter are shown as dark gray. Regions where no correspondence is found between the stereo images are shown to be black.

To reduce the amount of computation, a disparity map is calculated once every five frames. A search range of 48 and a SAD window size of 31 were used. For display purposes, the values in the disparity map are multiplied by 255 to force the pixel values into the range of 0 to 255.

C. Object Detection

The disparity map is thresholded as a binary image, where every pixel color value above 0.5 becomes 1, and every pixel below that value becomes 0. Because the areas closest to the cameras are white or nearly white in the disparity map, the threshold value is just below the pixel value for this color.

After the disparity map is thresholded, contour detection is performed on the binary image to find edges. This is necessary to determine the boundaries between white and black regions. Then, minimum area bounding rectangles are calculated; they are drawn on the image if they meet a length-width ratio of at most 3, have at least an area of 2,000 pixels, and are at least partly within the central third of the image. By enforcing these conditions, the algorithm is less likely to detect noise as objects. In particular, a bounding rectangle is drawn if and only if the algorithm determines that there is an object to be avoided.

D. Obstacle Avoidance Algorithm

The “see and avoid” algorithm is fairly simple for testing purposes, consisting of only forward and backward commands. It operates directly after the disparity map has been calculated, so that it is called every five frames. If an object has been detected in the disparity map, then the UAV moves back away from the object. Otherwise, every tenth frame, if the disparity map shows no object, the UAV proceeds forward. This method gives the UAV two opportunities to detect a possible collision before moving forward. The movement commands are issued with the PS-Drone library [7].
V. EXPERIMENTAL RESULTS

The cameras were calibrated using the above parameters, and then forty-five consecutive tests were performed. The test environment involved a line of several boxes and chairs, with the UAV launching from a distance of 6ft away, facing directly towards the obstacles. The program would launch the UAV, then wait for a remote keyboard input to put the UAV in autonomous flying mode. The vehicle would then incrementally move forward towards the obstacles. A successful test is one where the UAV detects an obstacle and then flies back autonomously to avoid it. A failed test is one where the UAV fails to detect an obstacle and continues moving forward to collide with it.

Fig. 8. Image of the test environment. Various cardboard boxes and chairs were used, with the UAV moving towards them from a starting distance of 6ft at a low hovering altitude.

All forty-five tests were successful, with the UAV detecting and avoiding the line of obstacles. Most objects were detected at a distance of about 2ft; however, it was found that this distance was somewhat inconsistent. During three tests, an obstacle was detected at a distance of 5ft. Furthermore, the movements of the UAV were not consistently straight, but this is due to the imprecision of the PS-Drone library used to control it programmatically [7]. There were no “false positive” results, where the system would detect an obstacle that was not actually there.

It was found that the system processed images with an average performance of 1Hz. Therefore, the UAV had to move towards the obstacles at a very slow rate. This prevented the UAV from being able to detect very fast moving obstacles, which is why only static obstacles were used in the experimental tests. If non-USB cameras such as the Raspberry Pi Camera Modules were used, the performance could be improved dramatically; this would allow the UAV to move faster and respond to collision threats more quickly. These cameras modules interface through a specialized Camera Serial Interface port to the system’s GPU, whereas the USB webcams used in this work are very CPU-intensive [31].

VI. CONCLUSIONS

Based on the experimental results, it can be concluded that an inexpensive stereo vision system can be reliable for autonomous UAV guidance and obstacle avoidance in GPS-denied environments, environments congested with many obstacles, or covert operations. The system described uses disparity mapping on rectified 640x480 images with 1Hz performance. This system was demonstrated successfully in forty-five experimental indoor flights with a Raspberry Pi 2 Model B aboard an AR.Drone 2.0 micro-UAV. Future work includes using a more powerful embedded system for better performance, using stereo cameras on each side of the UAV for omni-directional collision avoidance, and using a more accurate control library for more complex avoidance maneuvers.

VII. ACKNOWLEDGEMENTS

The authors would like to thank Ms. Neda Topuz and the Department of Computer Science and Software Engineering at Auburn University for their support in hosting the team. The work was funded by the National Science Foundation and the Department of Defense under grant #CNS-1540326.

REFERENCES


