Computer Vision "See and Avoid" Simulation using OpenGL and OpenCV

Auburn REU on SMART UAVs 2016

Morgan, Andrew
Youngstown State University
1 University Plaza
Youngstown, OH 44555
asmorgan@student.ysu.edu

Jones, Zach
Marshall University
John Marshall Drive
Huntington, WV 25755
jones867@marshall.edu

Chapman, Richard
Auburn University
Auburn, AL 36849
chapmro@auburn.edu

July 14, 2016

Abstract

In order to maintain safe flying environments and avoid disastrous midair collisions, the U.S. Federal Aviation Administration mandates that pilots must "see and avoid" other aircraft. The purpose of this research is to develop a See-And-Avoid (SAA) system for autonomous Unmanned Aerial Vehicles (UAVs). In this paper we present a novel approach to development of SAA capabilities using simulated cockpit video and computer vision algorithms. The cockpit video is generated using the computer graphics library OpenGL. The computer vision library OpenCV is used to construct algorithms for background filtering, obstacle detection, and collision avoidance. Algorithm design and preliminary test results will be covered in the scope of this paper.

1 Introduction

Accompanied by recent developments in microcontroller and onboard-computer technology, Unmanned Aerial Vehicles (UAVs) are experiencing explosive growth in civilian, commercial, and governmental sectors. To maintain safe flying practices and reduce the likelihood of midair collisions, the Federal Aviation Administration (FAA) implements strict regulations on both manned and unmanned aircraft. A major obstacle to the integration of UAVs into unrestricted airspace

has been the inability of UAVs to maintain safe flying practices and comply with FAA regulations for human pilots.

UAVs are increasingly using autonomous flight guidance systems, and many recent research endeavors have focused on ensuring that autonomous UAVs are maintained and controlled properly. For aircraft that fly autonomously, it is imperative that an aircraft is able to detect and avoid obstacles in its projected path. Much research has been conducted with a variety of technologies and sensor hardware, including cameras, radar, transponders, and data-link information exchange (ADS-B). To date, none of the proposed solutions are effective, reliable, and scalable enough to provide a robust system for real-time obstacle detection and avoidance.

2 Problem Statement and Motivation

Midair collisions between two aircraft or between an aircraft and another object are nearly always disastrous. The consequences of collision often include damage to wildlife and property, loss of the vehicle, damage to infrastructure, and potentially loss of life. In order to be integrated into civil airspace, UAVs must comply with FAA regulations regarding navigation and right-of-way. According too FAA order 7610.4, UAVs will need collision avoidance systems that match or exceed the abilities of a human pilot.

A major component of a robust avoidance solution is the ability to emulate the "See and Avoid" capability of a human pilot. While information exchange systems such as ADS-B have the potential to provide robust navigation and collision avoidance capabilities. However, information exchange technology relies on other aircraft having certain pieces of reliable hardware and requires communication using a known protocol. SAA systems provide not only an excellent auxiliary component to information exchange but also are a completely standalone system capable of detecting obstacles that do not share their location. Furthermore, as noted by Pham et. al. [8], camera-based vision systems are light, scalable, and easy to deploy.

The primary objective of this research is to build a robust "See and Avoid" (SAA) system capable of detecting and avoiding obstacles using visible light computer vision techniques. The ultimate goal is the develop a system capable of being deployed on a UAV. Primary consideration is given to fixed-wing UAVs due to their inability to come to a complete stop or move backwards like multi-rotor platforms. Collision avoidance on a fixed-wing platform requires real-time vision processing and course planning.

A secondary objective of the research is to design a test environment that can simulate cockpit video. The test environment will consist of a 3-dimensional scene rendered by the computer graphics library OpenGL. The scene can simulate the view of a front-facing camera mounted to an aircraft. The camera's viewport and orientation will change as the aircraft moves and pitches in response to navigational commands and avoidance maneuvers. The simulated airspace will be given a realistic background and will contain models of other ob-

stacles and aircraft ranging from commercial airliners to single-propeller planes.

Real-time vision processing will be accomplished through the use of the OpenCV (Open Computer Vison) library. OpenCV will perform many of the filtering and detection algorithms used to identify other aircraft. The proposed algorithm will consist of a series of transformations included in the OpenCV library. By feeding the simulated cockpit video to the algorithm, we hope to simulate real-time flight patterns that the SAA system would enact when integrated with a real aircraft.

3 Literature Review

A significant amount of research related to SAA systems has been conducted. As noted by Dayton [2], some past research focuses on quadrotor UAVs. Recently the focus has shifted towards fixed-wings in order to take advantage of the increased flight times, payload capacity, range, and speed provided by a fixed-wing platform.

Most research under the domain of SAA systems involves addressing one or more of the following problems: detection of static or moving objects in the UAV's field of vision, determination of collision threats, and procedures for avoiding collision. A complete solution to the SAA problem would require working subsystems that address each problem. However, most research focuses on a particular sub-problem.

Perhaps the most straightforward approach to the SAA problem is a ground-based collision avoidance system similar to that used by air traffic controllers. Herwitz [10] has patented a ground-based "sense-and-avoid display system" (SAVDS) that tracks airborne targets similarly to an air traffic controller. The ground control station has a radar which displays the position of all aircraft. Air traffic controllers can issue commands for UAVs to change course or, as usual, issue commands to human pilots to evade collision. However, complete coverage of the airspace is impossible with such a ground-based system, so many research endeavors have focused on integrating an airborne SAA system on the UAV itself.

Dayton et. al. [2] developed a collision detection system for fixed-wing UAVs capable of handling both moving and non-moving objects. The OpenCV library was used to process a video stream from a front-facing, externally-mounted camera. Shi-Tomasi corner detection was used to identify interesting features to track, and the Lucas-Kanade optical flow method was used to track moving objects and predict collisions. Ortiz and Neogi [7] proposed a similar method for object detection in UAV SAA systems. They noted that many other object tracking algorithms are too computationally intensive to meet the real-time requirements of UAV navigation. They found that although optic flow is a good alternative to more expensive tracking algorithms, it still may not meet real-time speed. However, advancements in micro-processor technology have since made optic flow a viable option for object tracking.

Another common technique for target detection in see-and-avoid systems is

morphological filtering. Morphological operations are a class of algorithms that are able to extract information about shapes and point-like objects in an image. Carnie et. al. [1] use a morphological operation known as a "Close-Minus-Open" (CMO) in order to extract point-like masses from large-scale clutter. Using this technique, they were able to perform long-range detection of collision targets even against heavy background clutter such as storm clouds. Initial tests indicate that their algorithm may be capable of outperforming the human eye at long ranges. However, their system is not yet capable of extracting information when there is both sky and terrain clutter. Lyu et. al. [6] addressed this problem by first performing a sky-ground segmentation, then following up with a morphological CMO operation to detect points of interest.

Rather than motion analysis, some researchers employ stereo vision solutions which use multiple cameras. The most common configuration has two cameras mounted parallel and facing the same direction. Stereo vision systems can provide information about depth by solving a correspondence problems between the images from the two cameras. As noted by Ortiz [7], stereo vision systems are limited by the computational intensity required to process dual sets of images. Some researchers have attempted to address the high computational cost of stereo vision systems using techniques such as dynamic programming in [4]. Another approach is to transform the image matching problem into an optimization problem and solve it using a genetic algorithm as explored by Woo and Dipanda in [11].

Research combining color optic flow and stereo vision techniques as described by Hravar et. al. [5] have potential to be more effective than using only a single technique. However, computational considerations are important due to the real-time nature of vision processing and navigation. Many other techniques have been published on the problem of automatic target detection. Deshpande et. al. [3] use max-mean and max-median filters on infrared sensor images to detect small targets and distinguish them from clutter. Reed et. al. [9] use a technique called "3-D matched filtering" to extract target points from background noise.

The doctoral dissertation of Yu [12] explores the implementation of vision based planning, avoidance, and target tracking on UAVs. Two techniques are used to build maps and plan paths in the local-level frame. The first technique uses a depth map of an environment obtained by computer vision methods, using an Extended Kalman Filter (EKF) to estimate range and sizes of obstacles. The second technique constructs local multi-resolution maps using an occupancy grid, giving higher resolution to the areas that are close to the UAV. Though without triangular vision provided by a stereo vision setup, depth perception was fairly inaccurate.

In summary, a wide variety of computer vision algorithms have been developed and employed in hopes of developing SAA capabilities for an autonomous UAV. To date and to the best of our knowledge, none of the approaches are sufficiently advanced to acceptably replace a human pilot.

4 See and Avoid System

Our research approach to the "See and Avoid" problem involves two main components. The first is a test environment for SAA algorithms that will be able to simulate cockpit video as would be collected using a front-facing camera mounted to a fixed-wing UAV. The ultimate goal of this component is to provide an excellent test bed for future research of computer-vision based collision avoidance. The second component involves development of a suite of algorithms that can perform the operations required of a SAA system - namely, detection of obstacles, prediction of obstacle paths, and issuing of avoidance commands. The ultimate goal of this research track is to create a SAA system that can emulate the "see and avoid" capabilities of a human pilot.

The first component was developed using the Open Graphics Library (OpenGL). OpenGL is a cross-platform API for rendering 2D and 3D graphics. OpenGL was used to build a cockpit video simulator that can provide a stream of images similar to that seen by a front-facing camera. The simulation includes realistic controls for the speed, direction, and orientation of a fixed-wing UAV. It also includes the capability to add various backgrounds that can be used to simulate large-scale background clutter like that present in real cockpit video. Finally, the simulation is also capable of loading additional aircraft and other obstacles into the same airspace as the UAV. It will be the work of the SAA algorithm to detect these obstacles and avoid them.

The second component was developed using the computer vision library OpenCV. OpenCV is an open-source library aimed at providing efficient implementation of a huge variety of computer vision and image processing algorithms. Using OpenCV, a series of transformations are performed on the current frame of the simulated cockpit video. The transformations are designed to filter out background noise, enhance target features, and track target features from frame to frame. A full description of the vision-processing algorithm is included in the next section.

An interface was also created to pass images from the OpenGL simulation to the SAA system for processing. The SAA system analyzes the current image and stores it in a short history. The SAA system is responsible for recognizing any potential obstacles and, in response, issuing avoidance commands to the UAV. The avoidance commands will be simulated in the cockpit video to provide a realistic feedback loop. The SAA system can be further broken down into two subcomponents - vision processing and collision avoidance.

4.1 Vision Processing

The vision processing algorithm is responsible for detecting objects of interest and providing information about the objects to the collision avoidance system. Care must be taken to distinguish objects that could potentially be obstacles from background noise such as cloud and terrain clutter. The vision processing algorithm must also be efficient enough to process video from the camera in near-real-time. We developed an algorithm based on the edge detection and

contour finding procedures in OpenCV.

The following set of steps are used to process the cockpit video. First, read an image from the camera and convert the image to grayscale. Converting to grayscale helps create more defined edges and removes distracting color information for the next steps. Next, we use OpenCV's *Canny* function to detect edges in the grayscale image. We then dilate the detected edges by 500%. This step removes gaps in the outlines of objects. After dilation, the image is blurred slightly (using a Gaussian blur) to help remove texture information and reduce cloud clutter. Finally, we use OpenCV's *findContours* function to detect closed shapes and draw bounding circles around the shapes. The output at the end of these steps is an array of detected objects with the width, height, and location (in on-screen coordinates) of each detected object. This information is then passed to the collision avoidance system.

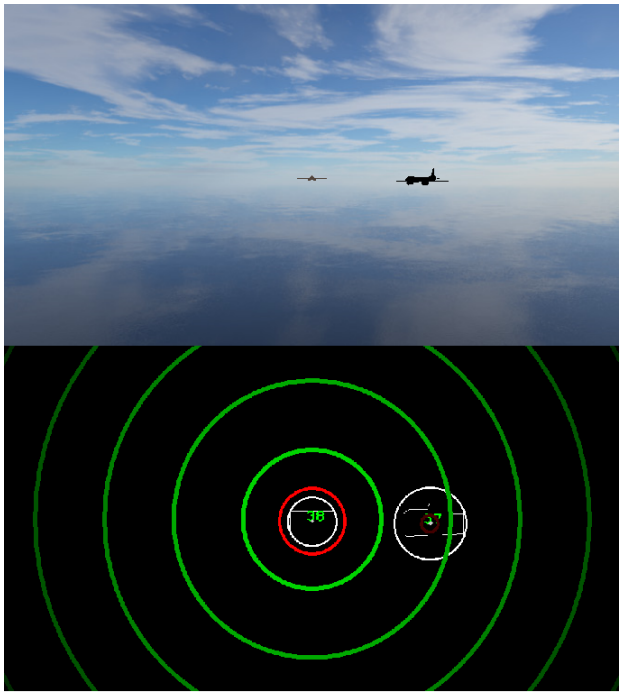

Figure 1: OpenGL Rendering (Top) processed by OpenCV (bottom)

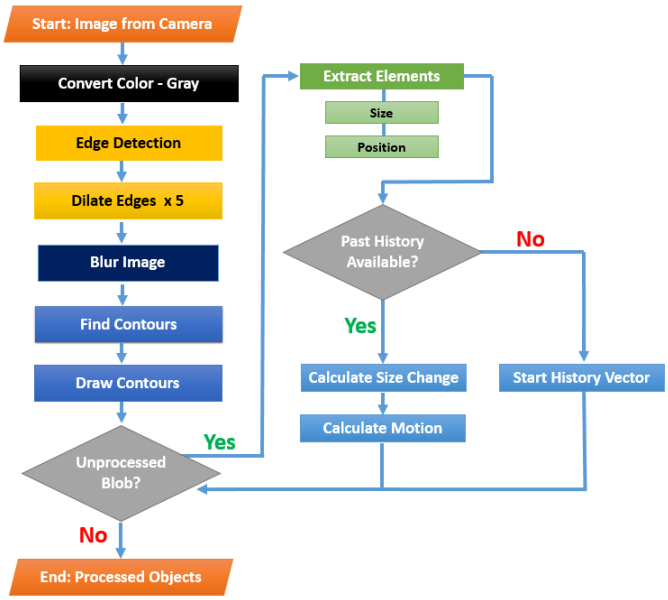

Figure 2: Vision Processing Flow Chart

4.2 Collision Avoidance

For a vision-based avoidance system, there are two broad categories of avoidance techniques. The first involves using visual stimuli to estimate the size of an identified object. If the size is known, then distance and velocity can be estimated. The second technique is distance-agnostic. That is, no assumptions are made about distance, and an avoidance maneuver is chosen that considers only information available from the 2D image captured by the front-facing camera. For our SAA system, we implemented a version of both techniques. The two algorithms are described below.

Distance-Based Avoidance

The distance-estimation technique used in our research involves the use of a reference frame and triangle similarity. This technique requires that the tracked object's actual size be known and requires an image of the object to be captured at a known distance. This image is known as the reference frame. The object's size (in pixels) is measured in the reference frame, and the camera's focal length F is computed as $F = \frac{p \cdot D}{S}$, where p is the measured size in pixels, D is the known distance to the object in the reference frame, and S is the actual size of the object. Then for any future image containing the object, the distance d to the object can be computed as $d = \frac{S \cdot F}{\hat{p}}$, where $\hat{p}$ is the new measured size in pixels. This technique suffers from the obvious drawback that it must be calibrated for every object one wishes to recognize. For our system, we provided calibration data for a large, commercial airliner, a mid-size fighter,

and a small personal plane similar to a Cessna 172. The problem of identifying the object detected by the vision processing algorithm as a large, medium, or small-class aircraft is a separate research problem. For our SAA system, we provided calibration information about the type of aircraft detected to the collision avoidance system.

Using the triangle similarity technique described above, the distance to the objects detected by the vision processing algorithm can be estimated every frame. Since the camera's FOV is known, the on-screen position of the object and the distance to the object is enough to estimate the object's position. By tracking the objects position over several frames, we can estimate the object's velocity and projected flight path. To determine if a collision between our aircraft and the object is likely, we express the distance between our projected path and the object's path as a function of time, and minimize the function using elementary techniques from calculus. If the minimum distance is below some threshold, then the object is identified as a collision threat, and an avoidance maneuver is required.

Our algorithm attempts to choose an avoidance maneuver that safely avoids the object with minimal deviation from the intended course. To avoid the object, we seek to increase the deflection between our current flight path, and the projected path of the object. The algorithm responds to five different cases:

1. If the object is to our left and moving to our right, then turn sharply left.
2. If the object is to our left and moving to our left, then turn slightly right.
3. If the object is to our right and moving right, then turn slightly left.
4. If the object is to our right and moving left, then turn hard right.
5. If the object is approaching head-on, then turn slightly left or slightly right depending on the location of other objects in the sky.

This algorithm was testing in a variety of scenarios. The test cases and results are summarized in the "Testing and Results" section.

Distance Agnostic Avoidance

Although distance estimation allows for more sophisticated decisions within an avoidance algorithm, distance estimation techniques have potential drawbacks. Distance estimation is not always possible and, even at its best, is often inaccurate. Furthermore, it requires the SAA system to make assumptions about the size and type of aircraft in the field of view. It is not particularly practical to calibrate a distance-estimating system for every possible type of aircraft or airborne obstacle a UAV may encounter. Thus, it is desirable to develop a distance-agnostic avoidance algorithm that deflects the problems inherent to distance estimation.

The design of the distance agnostic algorithm is dependent on the hardware used. The first hardware variable is the camera's field of view. Typical

webcam-like cameras applicable for this use have a field of view (FOV) of 40-45 degrees. Wide-angle lenses typically provide a FOV of 60 degrees or more, but also have significant focal distortion as the distance from the center of the FOV increases. This algorithm was built assuming a 42-degree FOV. A second variable is the resolution of the camera. The algorithm presented below relies on several threshold values which will change with the resolution of the camera. During development, a resolution of 960x540 was assumed. For other resolutions, calibration and adjustment of the parameters would be necessary.

The algorithm receives as input the "blob" information from the vision processor. Each blob represents a potential threat and contains information about its current size, current position, and its change in size/position over the past 30 frames. For each blob, we must decide if the blob is a collision threat. A blob must meet two criteria to be considered a threat.

First, it must have obtained a sufficiently high "danger value" by remaining near the center or moving closer to the center of the camera's FOV during the past few frames. To compute the danger value, we define a weight function $f(D) = \frac{10}{1+e^{-0.0001(25000-D}}$, where D is the distance from the blob's center to the center of the FOV. Each frame, we compute the value of the weight function and add it to a running total of values from the past 30 frames. The weight function is bounded above by 300. Values near 300 indicate that the blob stayed near the center of the viewport for the past 30 frames. Values near 0 indicate that the blob is near the edges of the viewport or moved away from the center during the past few frames. If the danger value is above some threshold, then the blob has passed the first criteria. Second, the blob must be sufficiently large before an avoidance maneuver is attempted. This prevents the algorithm from being too aggressive and ensures an avoidance maneuver is made only when necessary. Also, if a blob is sufficiently large, the first criterion is dropped, an avoidance maneuver is immediately attempted.

After the threat criteria are met, the algorithm must determine an avoidance maneuver. This algorithm was designed to prefer avoidance maneuvers that involves changes in elevation. If a blob is located in the top half of the viewport, then the UAV will be instructed to avoid by descending to a lower altitude. If the blob is located in the bottom half of the viewport, then the UAV will instead be instructed to ascend.

To achieve even more robust avoidance, the algorithm will also add a horizontal component to the avoidance maneuver when necessary. If the blob is moving to the right relative to the camera perspective, then the UAV will avoid to the left. If the blob is moving the left, then the UAV will avoid to the right. The combination of elevation change and horizontal path modification is nearly always enough to avoid a collision, provided that the maneuver is completely in a timely fashion. However, a horizontal translation nearly always results in a larger deviation from the intended path than an elevation change, so when the horizontal velocity of a blob is not significant, no horizontal component will be present in the avoidance maneuver. Additionally, if further "dangerous" blobs are detected in the middle of an avoidance maneuver, the current maneuver will be abandoned and a new one will be planned based on the new threats.

4.3 Testing and Results

The OpenGL test bed previously described allows for quick construction and simulation of various scenarios. We were able to use the OpenGL simulation to test the vision processing and avoidance algorithms in collision scenarios that mimic what an autonomous UAV might experience.

The simplest possible collision scenario involves only two planes - the plane being autonomously controlled and some other "obstacle" plane. We tested two-plane collision scenarios where the obstacle plane was approaching the autonomous plane at an angle of 0 degrees (head-on), 45 degrees, 90 degrees, and 160 degrees (overtaking from the rear).

More complex collision scenarios can be devised by adding more obstacle planes. In addition to the two-plane scenarios described above, we also tested three- and four-plane scenarios. We included all possible combinations planes approaching at either 0, 45, 90, or 160 degrees, neglecting the cases where two planes would approach from the same direction.

The results for two-, three-, and four-plane collision scenarios are summarized in the tables below. The left column describes the orientation of obstacle planes as an ordered pair. For example, (0, 90, 160) indicates that there were three obstacle planes, one approaching at 0 degrees, one approaching from 90 degrees, and one overtaking from the rear.

Table 1: Two-plane test results

Scenario	**Distance-Agnostic**	**Distance-Based**
0 degrees	Avoided	Avoided
45 degrees	Avoided	Avoided
90 degrees	Avoided	Avoided
160 degrees	Collided	Collided

Table 2: Three-plane test results

Scenario	**Distance-Agnostic**	**Distance-Based**
(0, 45)	Avoided	Avoided
(0, 90)	Avoided	Avoided
(0, 160)	Avoided	Avoided
(45, 90)	Avoided	Avoided
(45, 160)	Avoided	Collided
(90, 160)	Avoided	Collided

Table 3: Four-plane test results

Scenario	Distance-Agnostic	Distance-Based
(0, 45, 90)	Avoided	Avoided
(0, 45, 160)	Avoided	Avoided
(0, 90, 160)	Avoided	Avoided
(45, 90, 160)	Avoided	Collided

In addition to the controlled, multi-plane scenarios described above, we also ran tests in which the autonomously controlled plane would fly through a series of checkpoints and would be required to avoid any other planes in the airspace. In reality, most aircraft will never encounter more than one "obstacle plane" at a time unless the aircraft was near a crowded airspace such as an airport.

With this consideration in mind, we devised a scenario in which many planes were maneuvering in an airspace similar to that of an airport. We then instruct the autonomous plane to fly a course that crosses through the airport space several times at a variety of angles. While unrealistic for legal reasons, this scenario provides a robust test for an autonomously guided UAV. As a control, we ran the simulation with no collision avoidance. We then ran the simulation with the distance-based and distance-agnostic algorithms and recorded the results.

The most important property of a collision avoidance system is its ability to avoid collisions. Success of the avoidance algorithm can be measured by the number of collisions per completed waypoint. Naturally, lower numbers indicate more successful collision avoidance. The following line chart shows collisions detected versus waypoints completed. The bar chart shows the average number of collisions per waypoint.

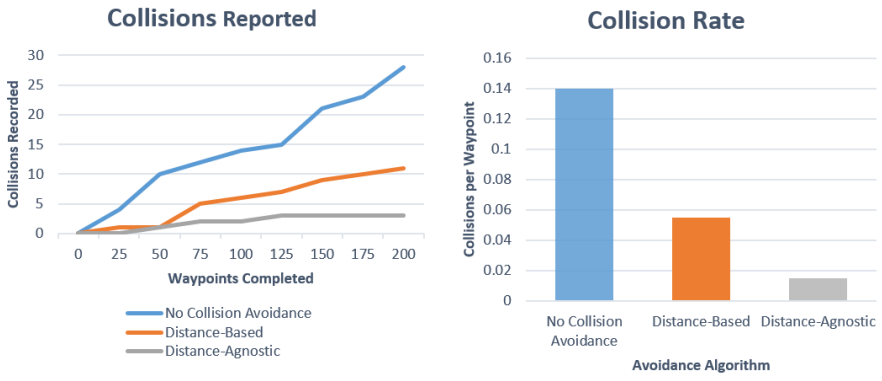

Figure 3: Collision avoidance comparison)

A desirable secondary property of a collision avoidance system is to minimize

deviation from the intended course. For our simulation, deviation from the intended course can be measured by the average time to complete a waypoint. Lower times indicate more efficient avoidance maneuvers. The following line chart shows flight time versus waypoints completed. The bar chart shows the average time (in minutes) required to complete one waypoint.

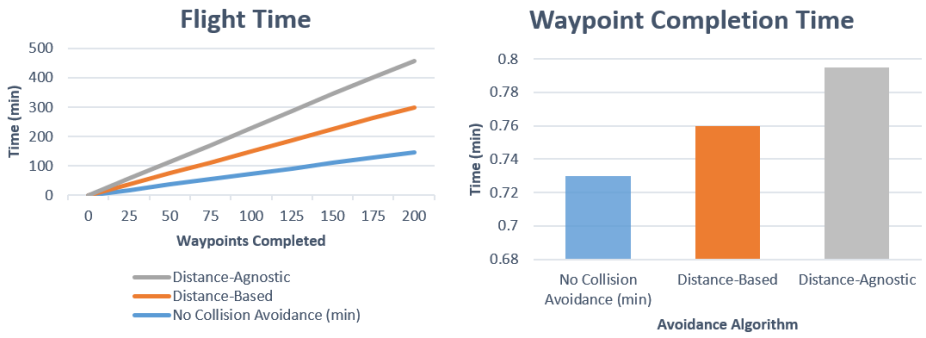

Figure 4: Flight efficiency comparison

The data show that both the distance-based and distance-agnostic algorithms significantly reduce the number of collisions. Our version of a distance-agnostic avoidance outperforms the distance-based avoidance algorithm in terms of collision avoidance in this simulation. However, we do not believe this result holds in the general case.

Similarly, the distance-based algorithm outperforms the distance-agnostic algorithm in terms of flight efficiency, but this result is rather spurious as the distance-based algorithm made fewer avoidance maneuvers. In both cases, it should be noted that the average waypoint completion time is close to that of no avoidance, indicating that the algorithms have good efficiency. Avoidance maneuvers appear to take near-minimal deviations from the true course.

5 Conclusion

One of the main obstacles to integration of autonomous UAVs in civil airspace is the lack of robust collision avoidance systems. In this paper we explored a collision avoidance system based on a front-facing visible-light camera. The objective was to emulate the "see and avoid" mechanism of a human pilot.

We developed a test environment of practical airspaces using OpenGL. Furthermore, we constructed a vision processing algorithm and avoidance algorithms to identify threats in the sky and to avoid them efficiently. Using the test environment, we found that both algorithms were able to avoid many otherwise disastrous collision events while maintaining efficient flight patterns.

We found that the vision processing algorithm is quite robust. Nearly all background noise is filtered, and collision threats are nearly always identified in plenty of time to execute an avoidance maneuver. However, the avoidance algorithms we developed are still susceptible to poor analysis of obstacles and poor choices for avoidance maneuvers. Single camera vision cannot be implemented as the sole avoidance system due to its lacking FOV. Future work will focus on improving the ability of the SAA system to identify obstacle aircraft type, direction, and speed. Using this information, sophisticated avoidance algorithms could be used to minimize collision rates and maximize efficiency.

References

[1] Carnie, Walker, and Corke. Computer-vision based collision avoidance for uavs. In Proceedings 11th Australian International Aerospace Congress, Melbourne, Australia. Accessed 5/25/2016., 2005. http://eprints.qut.edu.au/4627/1/4627.pdf.

[2] Dayton, Enriquez, Gan, Liu, Quintana, and Richards. Obstacle avoidance system for uavs using computer vision. Accessed 5/25/16. https://www.cpp.edu/~cppsrc/documents/Richards.pdf.

[3] Deshpand, Venkateswarlu, and Chan. Max-mean and max-median filters for detection of small targets. *Conference on Signal and Data Processing of Small Targets*, 1, 1999.

[4] Gonzalez, Cancelas, Alvarez, Fernhndez, and Enguita. Fast stereo vision algorithm for robotic applications. *7th IEEE International Conference on Emerging Technologies and Factory Automation, 1999. Proceedings.*, 1, 1999.

[5] Hrabar, Sukhatme, Corke, Usher, and Roberts. Combined optic-flow and stereo-based navigation of urban canyons for a uav. 2005 IEEE International Conference on Intelligent Robots and Systems. Accessed 7/12/2016.

[6] Lyu, Pan, Zhao, Zhu, Tang, and Zhang. A vision based sense and avoid system for small unmanned helicopter. 2015 International Conference on Unmanned Aircraft Systems (ICUAS). Accessed 5/25/2016. http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5698303, June 2015.

[7] Ortiz and Neogi. Color optic flow: A computer vision approach for object detection on uavs. 25th Digital Avionics Systems Conference October 15, 2006. Accessed 5/25/2016. http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4106304.

[8] Pham, Smolka, Stoller, Phan, and Yang. A survey on unmanned aerial vehicle collision avoidance systems. Department of Computer Science, Stony Brook University, Stony Brook, NY, USA. Accessed 5/25/2016. https://arxiv.org/ftp/arxiv/papers/1508/1508.07723.pdf.

[9] Reed, Gagliardi, and Stotts. Optical moving target detection with 3-d matched filtering. *IEEE Transactions on Aerospace and Electronic Systems*, 24(4):327 – 336, July 1988.

[10] Herwitz S.R. Ground-based sense-and-avoid display system (savds) for unmanned aerial vehicles, 09 2007.

[11] Woo and Dipanda. Matching lines and points in an active stereo vision system using genetic algorithms. *Proceeding of the International Conference on Image Processing*, 3:332 – 335, 2000.

[12] Yu. *Vision-based Path Planning, Collision Avoidance, and Target Tracking for Unmanned Air and Ground Vehicles in Urban Environments*. PhD thesis, Provo, UT, USA, 2011. AAI3502484 http://search.proquest.com/docview/963753397.