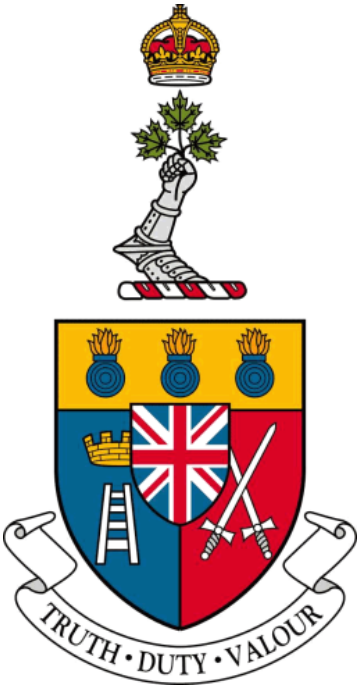
Royal Military College of Canada

Department of Electrical and Computer Engineering



Project Coatimunde Requirements

Computer Optics Analyzing Trajectories In Mostly Unknown, Navigation Denied, Environments

DID-03 - Statement of Requirements

Presented by:
Amos Navarre Hebb & Kara Stephan

Presented to:
Dr. Sidney Givigi & Rachid Beguenane

October 13, 2018

Contents

1 Introduction

1.1 Document Purpose

Using Computer Optics for Analyzing Trajectories in Mostly Unknown, Navigation Denied, Environments (COATIMUNDE) is the goal of this project. The purpose of this document is to outline the requirements for COATIMUNDE. That is, what the project is to accomplish once all of the requirements outlined in this document have been completed and to what standard they shall be considered done. The benefits of meeting these requirements and solving this problem will be outlined and some possible use cases stated. This document will then identify the constraints that these requirements impose on this project.

1.2 Background

Both in the consumer and professional sectors the use of autonomous aerial vehicles is growing quickly. Currently these vehicles rely on skilled pilots to accomplish a very limited set of tasks. Adding obstacle avoidance capabilities to these vehicles and simplifying the task of following targets could allow for these systems to be used in many more situations. This section will give a quick background on obstacle avoidance, unmanned aircraft systems, computer vision, and the platforms we intend to use in this project.

1.2.1 Obstacle Avoidance

Obstacle avoidance is the task of satisfying a control objective, in this case moving toward a visual target, while subject to non-intersection or non-collision position constraints. The latter constraints are, in this case, to be dynamically created while moving in a reactive manner, instead of being pre-computed.

1.2.2 Unmanned Aircraft Systems

Very generally any powered vehicle that uses aerodynamic forces to provide lift without a human operator being carried can be considered an unmanned aerial vehicle. Currently most of these vehicles make up a single component of a larger unmanned aircraft system.

An Unmanned aircraft system (UAS), or remotely piloted aircraft system (RPAS), is an aircraft without a human pilot on-board, instead controlled from an operator on the ground. Such a system can have varying levels of autonomy, something as simple as a model aircraft could be considered a UAS without any automation capabilities. Detecting, recognizing, identifying, and tracking targets of interest in complex environments and integrate with the systems required to process and fuse the collected information into actionable intelligence while operating in a low-to-medium threat environment is the current goal of the RPAS project by the Royal Canadian Air Force (RCAF) [1].

Flying a UAS requires a secure link to the operator off-board. Maintaining this link, particularly while flying close to the ground where more opportunities for interference are introduced is difficult. This difficulty is compounded in environments where potentially hostile actors may be attempting to jam communications. This necessitates a level of automation on-board capable of maintaining flight while denied navigation information.

There are many different types of approaches for this problem, but most involve some form of identifying targets in real time and reacting as they become visible to the aircraft. This has proven successful on a flying robot traveling at high speeds [2]. This system successfully combined trajectory libraries and a state machine to avoid obstacles using very little computational power even at very high speeds [3]. Another solution to obstacle avoidance on flying robots was the creation of NanoMap [4]. This allows for 3D data to be processed at a much faster rate allowing for higher speeds of the robot [4].

1.2.3 Computer Vision

Currently there are many different ways that computers can make high-level decisions based on digital image information. There are many methods to acquire, process, and analyze data from the real world using a camera. While this is a very broad field, we intend to focus especially on the aspects around motion estimation and object recognition. Both will be working with a video stream taken from a camera.

Motion estimation can be accomplished using direct methods which compare whole fields of pixels to each other over successive frames, compared to indirect methods which look for specific features. The information resulting from motion estimation streams can be used to both compensate for motion while analyzing other aspects of an image, and update a state machine.

Object recognition in our project will be accomplishing two tasks. Identifying a marker or target which will require more involved object recognition calculations, and very simple techniques, such as edge detection, to identify obstacles that exist in the path of the robot.

1.2.4 OpenCV

The Open Source Computer Vision Library (OpenCV) of programming functions is a cross-platform and free for use collection of functions primarily aimed at real-time computer vision[5]. Most well documented techniques to accomplish all of the computer vision goals of our project have already been created and refined in OpenCV. For this reason we will be leaning heavily on OpenCV functions.

1.2.5 Gazebo

Gazebo is a robot simulator that allows for creation of a realistic environment which includes both obstacles and markers similar to those being used in the lab. It will then be used to rapidly test algorithms.

1.2.6 Robot Operating System

The Robot Operating System (ROS) is a distributed message system that allows for various sensors and processors to work together to control a robot. It is open source and has been integrated already with OpenCV and Gazebo. There are many additional tools for detecting obstacles, mapping the environment, planning paths, and much more. It is also a robust messaging system that has been proven to be capable of real-time processes.

1.2.7 TurtleBot

The TurtleBot is a robot kit with open-source design and software. A TurtleBot is a robot built to the specification for TurtleBot Compatible Platforms[6]. In our case this is a Kobuki Base, an Acer Netbook running Ubuntu with ROS packages added, an X-Box Kinect, and some mounting plates.

The resulting robot looks like a disk supporting some circular shelves with a small laptop and a camera on the shelves. The base disk is 35.4cm in diameter, the topmost shelf is 42cm from the ground. The robot has a maximum speed of 0.65m/s.

1.2.8 AscTec Pelican

The Ascending Technologies Pelican is a 65.1cm by 65.1cm quad-rotor designed for research purposes[7]. It includes a camera, a high level and low level processor set up for computer vision and SLAM research. It is also capable of interfacing easily with other controllers and can carry up to a kilogram of additional gear.

1.3 Definitions

Many terms used during this project have synonyms, and similarly there are many different ways to word many of the things in this project and often no conventions. For this purpose we will define terms which we intend to have more specific meanings than in normal English, or will clarify what we mean by the often vague descriptions used for many of the components of this project.

1.3.1 Flying Robot

A Flying Robot refers to any vehicle that is able to execute instructions while in the air under its own power.

1.3.2 Marker

A marker will be specifically a high contrast shape, such as one output by ArUco or a QR Code generator, that is optimized for identification by a camera.

1.4 Aim

The aim of this project is to build an air robot that is able to identify a target and move toward it, avoiding any obstacles that are in the way. Tracking targets of interest in complex environments with a flying robot is the ultimate goal of this project.

1.4.1 Targets of Interest

This target will be an object in the environment that has already been identified visually by an operator before navigation information is denied, or pre-programmed into the flying robot before operation.

1.4.2 Complex Environment

The proposed use case of this project would be an environment with obstacles that the flying robot could potentially collide with. Flying at sufficiently low altitude that trees or buildings could come between the flying robot and the target of interest is the core of the project.

1.4.3 Benefits

Having a flying robot capable of accomplishing these tasks while totally autonomous will allow for the use of flying robots in environments closer to the ground, and will assist pilots in complex environments. These general requirements can be used in many situations.

- **Surveillance:** The robot could follow an interesting object, especially in an urban environment, without colliding with obstacles.
- **Search and Rescue:** The robot could move toward visual way-points set by pilots while avoiding obstacles in complex environments assisting in search efforts.
- **Inspections:** The robot could inspect objects in hard to reach environments and complex environments like rooftops or bridges.
- **Disaster Relief:** The robot could check inside buildings that may have compromised structural integrity, rubble on ground, *&c.*
- **Agriculture:** The robot could inspect tree-fruit or crops that can not be observed from overhead or check assets in remote locations such as irrigation equipment.

1.5 Scope

The project shall start by using a ground robot then progressing to a flying robot. The project will use computer vision to find targets and for avoiding multiple obstacles. Both robots shall be highly autonomous, requiring only user input to commence.

The scope of the project is limited due to only being able to test indoors. Ideally the UAV would be operating at a high speed and identifying an arbitrary target in an unpredictable environment. In our case though the testing must be completed within the confines of a relatively small robotics laboratory. Smaller obstacles will be used indoors simply due to space constraints. Lower speeds will be used as well, again due to the lack of space.

2 Requirement Definition Activities

2.1 Information

To better define the requirements for this project the RCAF RPAS project requirements were researched, to attempt to familiarize ourselves with the type of system outlined in the project [1]. Previously created obstacle avoidance systems implemented on similar UAVs were also researched to gain more background information for our knowledge. An article that was identified to be beneficial is about both target tracking and obstacle avoidance on a drone [8]. This outlined some specific problems that this project will likely face as well, and a possible solution to these problems. The discussions with our project supervisor Dr. Givigi revealed that it was possible to build a monocular detection system that can identify targets and multiple obstacles at once. This then can be implemented on an air robot which can take proper action to avoid these obstacles and move towards the identified target. Those discussions inspired the project requirements detailed in the following section.

3 Product Requirements

3.1 Functional Requirements (FR)

3.1.1 FR-01: Movement Toward Target

The TurtleBot and Flying Robot shall move toward a target under control.

3.1.2 FR-02: Identification of Target

The robots shall recognize optically an OpenCV's ArUco shape and be able to give information on the location of the target relative to the robot.

3.1.3 FR-03: Identification of Obstacle

The robots shall optically recognize obstacles in its environment and identify where they are relative to itself. The obstacle will be defined as any solid opaque object that is at least the height of the camera on the robot at the time of detection.

3.1.4 FR-04: Avoiding Obstacle

The robots shall be able to make a deviation from its current movement pattern to avoid an obstacle in its path and then return to this pattern. The obstacle will be defined as any solid opaque object that is at least the height of the camera on the robot at the time of detection.

3.1.5 FR-05: Multiple Obstacles

The robots should be able identify multiple obstacles and avoid them accordingly. See FR-04.

3.2 Performance Requirements (PR)

3.2.1 PR-01: Target Identification

The robots shall be able to identify and locate OpenCV's ArUco shapes within a 15m radius. See FR-02.

3.2.2 PR-02: Interpretation of Visual Data

The Flying Robot should be able to process an image and plot obstacles at a rate of 10 Hz. See FR-03.

3.2.3 PR-03: Multiple Obstacles

The Flying Robot shall be able to navigate an environment and detect up to two obstacles simultaneously. See FR-04.

3.3 Interface Requirements (IR)

3.3.1 IT-01: Gazebo Simulator

The Gazebo simulator shall be run on a laptop. Robots within the simulation will be interfaced through Robot Operating System.

3.3.2 IR-02: Turtlebot Communication through ROS

Communication to the Turtlebot will be done through Robot Operating System over USB.

3.3.3 IR-03: Air Robot Communication

The Flying Robot will be communicating through Robot Operating System over WiFi wireless network.

3.3.4 IR-04: Off-board User Interface

A simple interface, perhaps command line, will allow a user to select a marker as a target to start the robot toward this target. The robot should receive this signal over a wireless transmission.

3.4 Simulation Requirements (SimR)

3.4.1 SimR-01: Empty Lab TurtleBot in Gazebo

A Gazebo simulation environment which roughly approximates a lab environment with markers placed around will be created. A model of the TurtleBot will be able to navigate toward markers in this environment.

3.4.2 SimR-02: Lab with Obstacles TurtleBot in Gazebo

Stationary obstacles will be added to the Lab simulation environment and a TurtleBot shall navigate toward markers in the lab while avoiding these obstacles.

3.4.3 SimR-03: Lab with Obstacles Flying Robot

An environment with obstacles will be created and a Flying Robot shall navigate toward markers in the simulation while avoiding obstacles.

3.5 Implementation Requirements (ImpR)

3.5.1 ImpR-01: Computer Vision

OpenCV programs shall be created that can use a single video stream and identify both markers and obstacles.

3.5.2 ImpR-02: Gazebo Simulation

Prior to testing on the Turtlebot, the program shall be implemented on the gazebo simulation. This simulation environment will be able to create a realistic video stream that can be used for testing.

3.5.3 ImpR-03: Robot Operating System

Using the robot in the simulation environment the appropriate components, tools, and libraries to interpret an OpenCV stream, make decisions based on the environment, and execute instructions will be developed.

3.5.4 ImpR-04: Turtlebot in Simulation

The simplest obstacle avoidance algorithm must be implemented on a Turtlebot using the Robot Operating System.

3.5.5 ImpR-05: Turtlebot in Lab

The Turtlebot will operate in the real world, identifying targets and avoiding obstacles.

3.5.6 ImpR-06: Flying Robot in Simulation

The obstacle avoidance algorithm used for a Flying Robot will be implemented in a simulation avoiding obstacles and identifying targets.

3.5.7 ImpR-07: Flying Robot in Lab

The Flying Robot will operate in the lab, identifying targets and avoiding obstacles.

3.6 Schedule Restrictions (SchR)

3.6.1 SchR-01: Simulation

The first TurtleBot simulation shall be able to operate in a Gazebo simulation environment no later than November 5^{th}. It shall be able to identify an ArUco shape as a marker, move toward the marker, and avoid a stationary obstacle placed into the simulation environment.

3.6.2 SchR-02: TurtleBot Prototype

The first functional prototype shall be a TurtleBot robot capable of positively identifying a marker, moving toward the marker, and avoiding an obstacle placed into its environment no later than December 18^{th}.

3.6.3 SchR-03: Flying Prototype

The first functional flying prototype shall be capable of identifying a marker, moving toward the marker, and avoiding an obstacle placed into its environment no later than February 18^{th}.

3.7 SchR-04: Data Item Descriptions

Data Item Descriptions (DID) to be presented are DID-04: Preliminary Design Specification due November 22nd, DID-05: Preliminary Design Review Presentations due November 29th, and the DID-06: Schedule Update is due January 17th. The DID-07: Final Detailed Design Document is due March 21st, the Final Project Presentation, DID-08, is March 28th and the Final Project Demonstration, DID-09, is on April 9th.

4 Risk Assessment

4.1 Identifying Markers

The markers we intend to use, at least initially, are intended for use in Augmented Reality purposes. Tests of these shapes show that they are capable of being identified at many angles, but the reliability with which they can be identified decreases quickly as the viewing angle changes.

There is potential that markers which are reliably identified at low speeds and in simulation may not be detectable on a flying vehicle moving faster.

4.1.1 Likelihood

Both the TurtleBot and Flying Robot shall be traveling at lower speeds, therefore the likelihood of this risk is low.

4.1.2 Impact

High impact, the primary task that our robot must accomplish is moving toward a visual target. If the robot is not able to reliably identify a marker then there will be no way to verify its capability.

4.1.3 Process Solution

Alternatives to the ArUco markers can be tested, or if these prove truly unreliable then coloured areas or illuminated targets could be incorporated. These are less desirable as ideally the robot would be able to fly toward an arbitrary target.

4.2 Computer Hardware Limitations

Image processing, especially quickly and with multiple goals, is computationally expensive. While this should not be an issue on a ground vehicle moving at slower speeds, it may become more of an issue on a Flying Robot if it operates at higher speeds and is incapable of carrying as much on-board computational capability.

4.2.1 Likelihood

High likelihood due to the amount of computational power required to do image processing.

4.2.2 Impact

Medium impact, presumably we would still be able to prove our systems on the ground robot which is capable of carrying as much processing power as needed. It may also only be a limitation at higher speeds or may impose limits on what the Flying Robot is capable of identifying and tracking.

4.2.3 Process Solution

Testing many algorithms and working to streamline the algorithm used, especially for the Flying Robot, as well as selecting hardware that is complimentary to the kind of loads created by running such an algorithm can mitigate these risks. Using FPGA can help with the solution!!!

4.3 Flight Control System Inaccuracy

Many UAV's are not able to execute arbitrary movements with high accuracy. Verifying that the drone is actually executing instructions may be difficult.

4.3.1 Likelihood

Medium likelihood, depending on how well developed the libraries for our particular drone these issues may have all been sufficiently worked out for the purposes of this project.

4.3.2 Impact

Low impact, we could still observe the Flying Robot making decisions even if it is unable to execute the instructions given properly.

4.3.3 Process Solution

Using smaller number of obstacles and moving at a slow enough speed should ensure that the robot never needs to execute extreme moves. These more extreme moves are where inaccuracies in control models will exposes themselves the most.

5 Conclusion

The project requirement have been listed for the Flying Robot obstacle avoidance system. Background and requirement defining activities have been given to lay the groundwork and a basic understand of the current research in this domain. This document will be used for guiding the project and aiding in the creation of the preliminary design.

References

[1] *Royal canadian air force — news article — update and new name for the joint unmanned surveillance target acquisition system (justas) project*, Jul. 2018. [Online]. Available: `http://www.rcaf-arc.forces.gc.ca/en/article-template-standard.page?doc=update-and-new-name-for-the-joint-unmanned-surveillance-target-acquisition-system-justas-project/j9u7rzyf`.

[2] A. J. Barry and R. Tedrake, "Pushbroom stereo for high-speed navigation in cluttered environments", in *Robotics and automation (icra), 2015 ieee international conference on*, IEEE, 2015, pp. 3046–3052.

[3] A. J. Barry, P. R. Florence, and R. Tedrake, "High-speed autonomous obstacle avoidance with pushbroom stereo", *Journal of Field Robotics*, vol. 35, no. 1, pp. 52–68, 2018.

[4] P. R. Florence, J. Carter, J. Ware, and R. Tedrake, "Nanomap: Fast, uncertainty-aware proximity queries with lazy search over local 3d data", *arXiv preprint arXiv:1802.09076*, 2018.

[5] *About - opencv library*. [Online]. Available: `https://opencv.org/about.html`.

[6] M. Wise and T. Foote, *Rep: 119 - specification for turtlebot compatible platforms*, Dec. 2011. [Online]. Available: `http://www.ros.org/reps/rep-0119.html`.

[7] *Asctec pelican - uas for computer vision and slam.* [Online]. Available: `http://www.asctec.de/en/uav-uas-drones-rpas-roav/asctec-pelican/`.

[8] A. C. Woods and H. M. La, "Dynamic target tracking and obstacle avoidance using a drone", in *International Symposium on Visual Computing*, Springer, 2015, pp. 857–866.