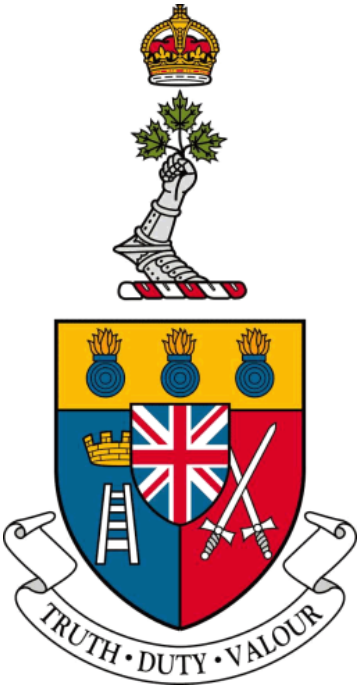
Royal Military College of Canada

Department of Electrical and Computer Engineering



DID-03 - Statement of Requirements

Presented by:
Amos Navarre Hebb & Kara Stephan

Presented to:
Dr. Sidney Givigi

September 29, 2018

Contents

1 Introduction

1.1 Document Purpose

The purpose of this document is to outline the project requirements. 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

1.2.1 Obstacle Avoidance

Both in the consumer and professional sectors the use of autonomous aerial vehicles is growing quickly.

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. Those later constraints are, in this case, to be dynamically created while moving in a reactive manner, instead of being pre-computed.

1.2.2 Uncrewed Aircraft Systems

Very generally any powered vehicle that uses aerodynamic forces to provide lift that does not carry a human operator can be considered an uncrewed aerial vehicle. Currently most of these vehicles make up a single component of a larger uncrewed aircraft system.

An Uncrewed 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 automation, something as simple as a model aircraft could be considered a UAS without any automation capabilities. A UAS capable of 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 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, and in environments where potentially hostile actors may be attempting to jam communications, necessitate 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, or indirect methods which look for specific features. The infor-

mation 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 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.

1.2.6 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 to the target, 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:** Robots could move toward visual way-points set by pilots while avoiding obstacles in complex environments assisting in search efforts.
- **Inspections:** Objects in hard to reach environments and complex environments like rooftops or bridges.
- **Disaster Relief:** Checking inside buildings that may have compromised structural integrity, rubble on ground, *&c.*
- **Agriculture:** Inspection of tree-fruit and assets in remote locations such as irrigation equipment which can not be done from overhead.

1.5 Scope

The scope of the project is limited due to only being able to test indoors.

Ideally the drone 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.

For this reason markers will be used as targets, this will make it easier to ensure the drone is accurately identifying the targets as it will be possible to encode messages into the markers and have the robot provide feedback about exactly what it is able to see and when.

Obstacles such as buildings or trees will be difficult to put inside the lab as well, so initially something simpler like a panel or box will be used. 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. Hopefully combining simulations at higher speeds with proof of concept in the real world at lower speed will be sufficient to show that this robot is able to identify targets and avoid obstacles at useful speeds.

2 Requirement Definition Activities

2.1 Information

To better define the requirements for this project the CAF 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: State Machine

Create a state machine for both turtlebot and air robots that allows for a movement decision to be made based on an input from the robot.

3.1.2 FR-02: Movement Toward Target

The air and land robots will be able to move toward a target under control.

3.1.3 FR-03: Trajectory Library

Create a trajectory library that will contain all possible allowed movements for both the land and air robots.

3.1.4 FR-04: Trajectory Library Updating State Machine

The Trajectory Library will update the state machine when a movement was made. The state machine will then account for this displacement and later correct the movement to keep the robot on the path to the target.

3.1.5 FR-05: Identification of Target

The robot shall recognize optically a goal and be able to give information on the targets location relative to the robot.

3.1.6 FR-06: Avoiding Obstacle

The robot 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.

3.1.7 FR-07: Identification of Obstacle

The robot shall recognize optically obstacles in its environment and identify where they are relative to itself.

3.1.8 FR-08: Multiple Obstacles

The robot should be able identify multiple obstacles and determine which is the most dangerous and avoid it accordingly.

3.2 Performance Requirements (PR)

3.2.1 PR-01: Target Identification

The robot will be able to identify and locate OpenCV's ArCuo shapes within a 15m radius. See FR-05.

3.2.2 PR-02: State Machine Corrections

The state machine will be able to correct for movements made and put the robot back on its original path without deviating more than 30cm. See FR-04. This can be verified using the motion capture equipment installed in the lab to record the path taken by the robot.

3.2.3 PR-03: 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-07.

3.2.4 PR-04: Multiple Obstacles

The flying robot shall be able to navigate an environment with two obstacles. See FR-08.

3.3 Interface Requirements (IR)

3.3.1 IR-01: Turtlebot Communication through ROS

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

3.3.2 IR-02: Air Robot Communication

The air robot will be communicating through Robot Operating System over wireless network.

3.3.3 IR-03: Air Robot Radio Denied

The air robot shall be able to identify a pre-determined marker and fly toward it while avoiding obstacles.

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: Gazebo with Turtlebot

Initially a simulation environment built for the TurtleBot will be created and most logic will be tested in this environment before using the robot in the physical world.

The first simulation should be in an environment which contains both obstacles and markers for the robot to navigate around and toward.

3.4.2 SimR-02: Flying Robot

A mechanism to simulate a flying robot will be created to test any logic added for navigation in 3D space.

3.5 Implementation Requirements (ImpR)

3.5.1 ImpR-01: Gazebo Simulation

Prior to testing on the Turtlebot, the program shall be implemented on the gazebo simulation.

3.5.2 ImpR-02: Turtlebot Robot Operating System

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

3.5.3 ImpR-03: Turtlebot in Lab

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

3.5.4 ImpR-04: 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 simulation shall be able to operate in a Gazebo simulation environment no later than November 5^{th}. It shall be able to identify a marker, move toward the marker, and avoid an 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 Miscellaneous Requirements (MiscR)

3.7.1 MiscR-01: Familiarization with TurtleBot

Before executing code on a physical TurtleBot it will be necessary to learn how this machine starts, is charged, and can have software loaded into it for use.

3.7.2 MiscR-02: Familiarization with Flying Robot

Before executing code on a physical Flying Robot it will be necessary to learn how to start, install batteries, and execute instructions on a flying robot in a lab environment.

3.7.3 MiscR-03: Motion Tracking Equipment Use

To verify that instructions are being executed accurately it may be necessary to use the existing motion tracking suite in the robotics lab. Basic use of this system will be necessary before executing instructions to see if the robot correctly follows routes.

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

Low likelihood due to low speeds that even the flying robot will be operated at.

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, although hardware for flying robot has not yet been chosen there are currently very few hardware devices capable of executing complex computer vision instructions quickly.

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 air 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.

4.3 Flight Control System Inaccuracy

Many drones 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 air 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.