Royal Military College of Canada
Department of Electrical and Computer Engineering

Designing Coatimunde

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

DID-04 - Preliminary Design

Presented by:
Amos Navarre Hebb & Kara Stephan

Presented to:
Dr. Sidney Givigi & Dr. Rachid Beguenane

November 14, 2018

Contents

1 Introduction

1.1 Document Purpose READ OVER THIS

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 preliminary design for COATIMUNDE. That is, what the design of the project is, how it will be build, and how this meets the requirements for the project. These requirements have been outlined in the Statement of Requirements. The design of the project thus far will be shown and discussed within the Design Section. This document will then identify the constraints, risks, difficulties that have been faced so far in the project.

1.2 Background ADD MORE FLUFF

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) **RPAS**.

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 **barry2015pushbroom**. This system successfully combined trajectory libraries and a state machine to avoid obstacles using very little computational power even at very high speeds **barry2018high**. Another solution to obstacle avoidance on flying robots was the creation of NanoMap **2018nanomap**. This allows for 3D data to be processed at a much faster rate allowing for higher speeds of the robot **2018nanomap**.

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**opencv**. 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**wise_foote_2011**. 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**asctec**. 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 STILL WANT THIS?

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 DO WE NEED THIS?

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.

2 Design Section

2.1 Block Diagram

The block diagram that is shown in Figure ?? contains the outline of the system for the COATIMUNDE project.

2.2 Node Diagram

2.3 Interfacing

2.4 Verification and Validation

3 Equipment

3.1 Equipment Table

4 Schedule

5 Unresolved Issues and Risks ADD MORE RISKS (we don't seem dangerous enough, lame)

5.1 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.

5.1.1 Likelihood

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

5.1.2 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.

5.1.3 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.

5.2 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.

5.2.1 Likelihood

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

5.2.2 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.

5.2.3 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 an FPGA could aid with this problem.

5.3 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.

5.3.1 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.

5.3.2 Impact

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

5.3.3 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.

6 Conclusion

The preliminary design has been shown 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 and referenced for the rest of the design and building processes for the project. The design laid out in this document will also be used in the the Preliminary Design Review and Detailed Design Document.