ROBUST INTELLIGENT SENSING AND CONTROL MULTI AGENT ANALYSIS PLATFORM FOR RESEARCH AND EDUCATION

by

Douglas Spencer Maughan

A thesis submitted in partial fulfillment
of the requirements for the degree

of

MASTER OF SCIENCE

in

Electrical Engineering

Approved:

UTAH STATE UNIVERSITY
Logan, Utah

2016

Abstract

Robust Intelligent Sensing and Control Multi Agent Analysis Platform for Research and Education

by

Douglas Spencer Maughan, Master of Science
Utah State University, 2016

Major Professor: Dr. Rajnikant Sharma
Department: Electrical and Computer Engineering

The aim of this thesis is the development and implementation of a controlled testing platform for the Robust Intelligent Sensing and Controls (RISC) Lab at Utah State University (USU). This will be an open source adaptable expandable robotics platform usable for both education and research. This differs from the many other platforms developed in that the entire platform software will be made open source. This open source software will encourage collaboration among other universities and enable researchers to essentially pick up where others have left off without the necessity of replicating months or even years of work. The expected results of this research will create a foundation for diverse robotics investigation at USU as well as enable attempts at novel methods of control, estimation and optimization. This will also contribute a complete software testbed setup to the already vibrant robotics open source research community.

This thesis first outlines the platform setup and novel developments therein. The second stage provides an example of how this has been used in education, providing an example curriculum implementing modern control techniques. The third section provides some exploratory research in trajectory control and state estimation of the tip of an inverted pendulum atop a small unmanned aerial vehicle as well as bearing-only cooperative localization experimentation. Finally, a conclusion and future work is discussed.

(128 pages)

Public Abstract

Robust Intelligent Sensing and Control Multi Agent Analysis Platform for Research and Education

by

Douglas Spencer Maughan, Master of Science

Utah State University, 2016

Major Professor: Dr. Rajnikant Sharma
Department: Electrical and Computer Engineering

The aim of this thesis is the development and implementation of a controlled testing platform for the Robust Intelligent Sensing and Controls (RISC) Lab at Utah State University (USU). This will be made available to other universities and researchers to use for both education and research. This differs from the many other platforms developed that keep their software for their own use. Our intent is this sharing of software will encourage collaboration among other universities and enable researchers to essentially pick up where others have left off without the necessity of replicating months or even years of work. The expected results of this research will create a foundation for diverse robotics investigation at Utah State University as well as enable attempts at novel methods of control, estimation and optimization.

This thesis first outlines the platform setup and novel developments therein. The second stage provides an example of how this has been used in education, providing an example curriculum implementing modern control techniques. Finally, research regarding control and estimation of a flying inverted pendulum and cooperative localization experimentation is presented demonstrating the unique capabilities of the platform, after which a conclusion is drawn.

Acknowledgments

Foremost, I would like to express my sincere gratitude to both my major professor, Dr. Rajnikant Sharma, and department head, Dr. Todd K. Moon, for making it possible for me to enter this program. Given my non-traditional engineering background I was initially unsure if what I wanted to accomplish was even possible within this relatively short time. I have been blessed by their knowledge and guidance. I cannot imagine myself succeeding without their assistance and willingness to give of their time.

My sincere thanks also goes to Dr. Xiaojun Qi for her assistance in areas outside the common curriculum available at Utah State University. Her encouragement and kind words have fueled my interest and confidence in exploring beyond that which is familiar.

I thank my fellow labmates and friends in the RISC lab: Ishmaal Erekson, Ashish Derhgawen, Imran Sajjad, Soudeh Dadras, Anusna Chakraborty, Parwinder Mehrok, and Abhishek Manjunath, for their friendship and support in my research. Also, I thank Thomas Amely for his assistance in the lab as well as his example of efficient and intelligent use of time.

Last but not the least, I would like to thank my beautiful wife Tiffany, my parents Douglas and Jennifer Maughan, as well as my son Rowan for providing the support to make this possible.

D. Spencer Maughan

Contents

List of Figures

Acronyms

3D	3-Dimensional
ANSI	American National Standards Institute
API	Application Program Interface
BSD	Berkeley Software Distribution
CAD	Computer Aided Drafting
DHCP	Dynamic Host Configuration Protocol
EKF	Extended Kalman Filter
ETH	Eidgenossische Technische Hochschule
GPS	Global Positioning System
GUI	Graphical User Interaface
HD	High Definition
IDE	Integrated Development Environment
IDL	Interface Definition Language
IP	Internet Protocol
LQR	Linear Quadratic Regulator
MAAP	Robust Intelligent Sensing and Control Multi Agent Analysis Platform
MCU	Microcontroller
MIT	Massachusetts Institute of Technology
OpenCV	Open Source Computer Vision Library
PID	Proportional Integral Derivative
PPM	Pulse Position Modulation
RC	Radio Controlled
RISC	Robust Intelligent Sensing and Control
ROS	Robotics Operating System
SCM	source code management
S.D.	Standard Deviation

STAIR	Stanford Artificial Intelligence Robot
SUAS	Small Unmanned Aerial Systems
TF	Tully Foote
UART	Universal Asynchronous Receiver/Transmitter
UAV	Unmanned Aerial Systems
USB	Universal Serial Bus
USU	Utah State University
VTOL	Vertical Take-off and Landing
XML	Extensible Markup Language

Chapter 1

Introduction

Many aspects of robotics are influenced by unpredictable dynamic environments in ways that make reliable estimation and control challenging. Dealing with these challenges in research can lead to confusing or even conflicting results [1]. Consequently, robotics researchers have turned towards controlled testing platforms to validate theory [2] [3] [4]. These platforms generally use high cost cameras that provide high-frequency, low-latency feedback. This feedback can be interpreted to provide accurate position and attitude measurements. These systems allow in-depth analysis and achieve performance that would otherwise be unattainable today in natural environments [5] [6].

Aside from providing superior accuracy and performance, these systems allow researchers to decouple estimation and control. These testbeds aim to provide "ground-truth" estimates that can be used in controls research for direct state feedback or in localization studies for verification of state estimation. Such "ground-truth" estimates are so significant to the world of robotics that the Journal of Autonomous Robots has published datasets for collaborative analysis [7]. However, the systems developed and referenced thus far are unique in their hardware and software. This limits collaboration among researchers requiring significant work to reproduce hardware results that may be impossible given the current differences between test-beds [8] [9].

The aim of this research is the development and implementation of a controlled testing platform for the Robust Intelligent Sensing and Controls (RISC) Lab at Utah State University (USU). This platform differs from the many other platforms in that the entire platform software and hardware will build upon open source robotics tools as much as possible. This is intended to encourage collaboration among other universities and enable researchers to essentially pick up where others have left off without the necessity of replicating months or even years of work.

This research will encompass three specific areas:

- Develop an open source adaptable expandable test bed platform for use in education and research.
  - Chapter 2 provides details related to the motion capture camera system. Specifications and costs are outlined while pictures are included.
  - Chapter 3 discusses the benefits and limitations of using the Robotics Operating System (ROS). The main architecture is outlined. Some common tools are highlighted along with the supported languages. Finally efficiency and availability of ROS are addressed.
  - Chapter 4 offers the motivations for development of the Robust Intelligent Sensing and Control Multi Agent Analysis Platform. Goals of the overall system and how these are achieved is addressed followed by an outline of basic tools developed for research.
  - Chapter 5 outlines hardware systems currently used in the RISC MAAP including the Parrot AR.Drone and 3DR Robotic's IRIS+. Interface and control are discussed along with relevant specifications.
- Demonstrate a complex control problem for use in education using the developed platform.
  - Chapter 6 provides sample curriculum is provided reproducing a recent relevant research topic in controls [10]. Differential flatness is outlined along with a demonstrated application. Results of labs are presented showing the student's demonstration in simulation as well as hardware.
- Extend modern controls research using the developed platform (Chapter 7).

- Flying Inverted Pendulum Trajectory Control on Robust Intelligent Sensing and Control Multi-Agent Analysis Platform

- Using Extended Kalman Filter for Robust Control of a Flying Inverted Pendulum

- Cooperative Bearing-only Localization Experimentation

The expected results of this research will create a foundation for diverse robotics investigation at Utah State University as well as enable attempts at novel methods of control and optimization. This will also contribute a complete software testbed setup to the already vibrant robotics open source research community.

Chapter 2

Motion Capture System

This chapter provides details related to the motion capture camera system. Specifications and costs our outlined while pictures are included.

The MotionAnalysis Motion Capture System relies on infrared cameras (figure 2.1) and reflective markers (figure 2.2) to provide marker position estimates (2 mm S.D.[1]) through imaging processing software named "Cortex"". These markers vary in size from radii of 2 to 14 mm.

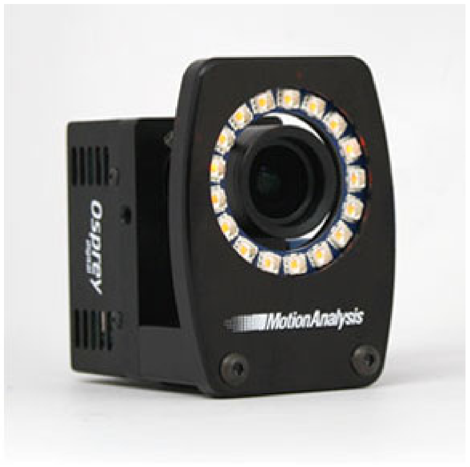


Fig. 2.1: Infrared Camera

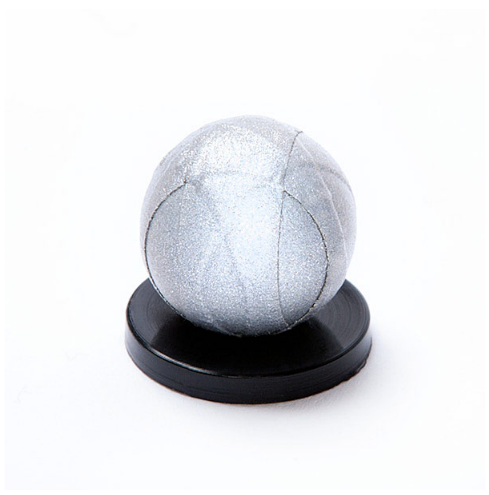


Fig. 2.2: Reflective Marker

This system offers an important feature that allows tracking of specific rigid geometrical configurations of markers known as an "identifying template". This template uses data from all markers within the template to improve overall estimation of marker positions and allows different objects to be distinguishable from one another as shown in figure 2.3. It is important when choosing the marker geometry to ensure a certain asymmetry

[1]Standard Deviation

in configuration to improve unique marker recognition. While velocity and orientation estimation is made available by MotionAnalysis at additional cost (approximately \$4000), such estimation was done in the lab and will be discussed in Chapter 4.

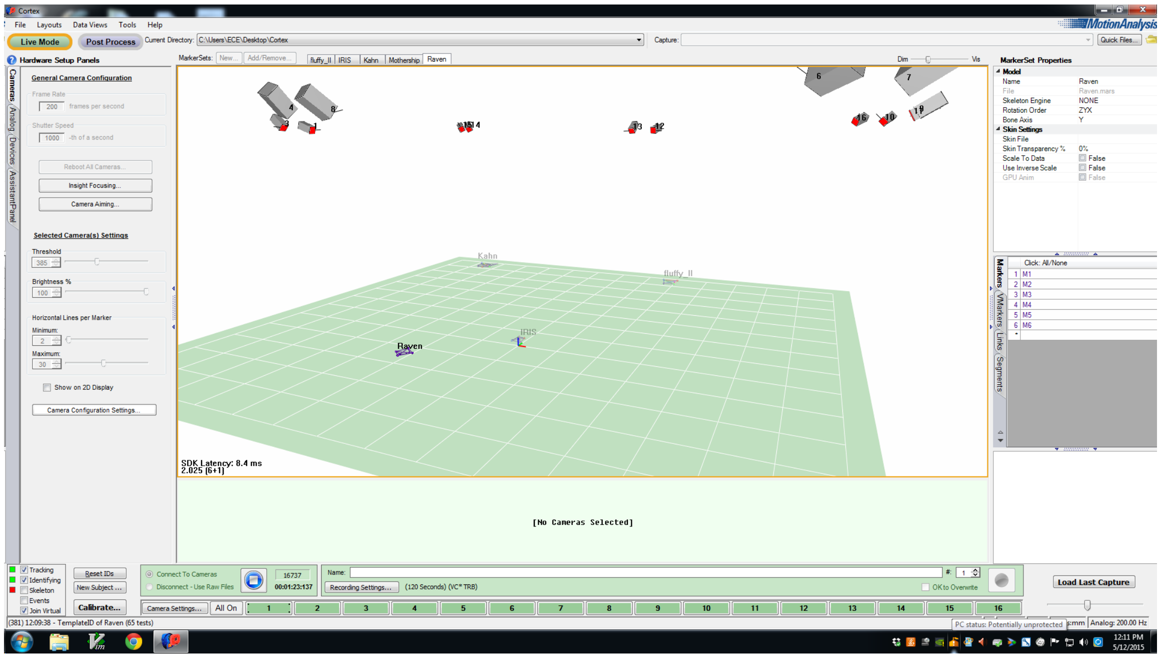


Fig. 2.3: Identifying Templates

The Robust Intelligent Sensing and Control Multi Agent Analysis Platform (RISC MAAP) camera setup is currently using 16 infrared cameras to capture a total volume of 44 cubic meters ($2.4 \times 4 \times 4.6\ m^3$). Data is sent over ethernet connection at 200 Hz. Communication and marker data interpretation will be discussed in Chapter 4.

Chapter 3
Robotics Operating System

This chapter discusses the benefits and limitations of using the Robotics Operating System (ROS). The main architecture is outlined. Some common tools are highlighted along with the supported languages. Finally efficiency and availability of ROS are addressed.

Robotics Operating System (ROS) is not an operating system in the traditional sense of process management and scheduling. ROS rather establishes a structured communications layer above the host operating systems of a heterogeneous computational cluster. It provides a flexible framework for writing robot software offering a collection of tools, libraries, and conventions that aim to simplify the task of creating complex and robust robot behavior across a wide variety of robotic platforms. Due to the many features and integration of powerful libraries, it is not surprising that many of the world's top engineers and researchers in the field of robotics make use of the tools available through ROS [11].

ROS was initially designed to meet a specific set of challenges encountered when developing large-scale service robots as part of the STAIR[1] project [12] at Stanford University and the Personal Robots Program [13] at Willow Garage. However, the resulting architecture is far more general than these and is gaining high popularity in robotics and other fields [14]. The philosophical goals and basic framework of ROS has been summarized in [15]. These central goals are as follows:

- Peer-to-peer: Each computer can act as a server for the others.
- Tools-based: Provide a modular framework of tools easily incorporated into projects.
- Multi-lingual: Allow use of a variety of programming languages and libraries.
- Thin: Lightweight communications

[1]Stanford Artificial Intelligence Robot

- Free and Open-Source

ROS has many aspects, this paper will focus on detailing only the central features which are currently used in the RISC MAAP. These are:

- General Communication
- Navigation
- Message Passing
- Data Logging
- Data Visualization
- Serial Interfacing
- Transforms
- Roslaunch

The following descriptions are summaries of Willow Garage, Stanford University, and MIT[2] publications outlining ROS [15, 16, 17].

3.1 General Communication

ROS consists of a number of processes, potentially on a number of different hosts, connected at runtime in a peer-to-peer topology.

[2]Massachusetts Institute of Technology

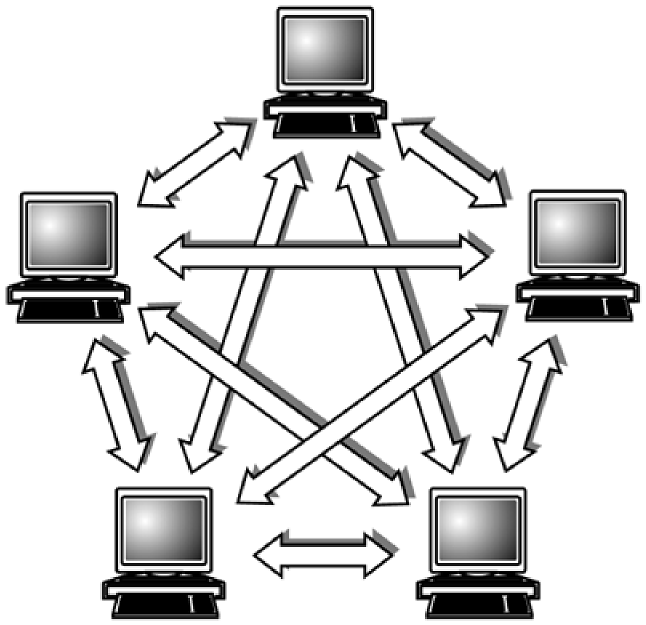


Fig. 3.1: Peer to Peer

This configuration avoids unnecessary traffic flow in contrast to the central-server-based alternative. This allows networking to high-power off-board machines that are running computation-intensive tasks such as computer vision or speech recognition to be much more efficient.

This peer-to-peer topology requires some sort of look up mechanism that allow processes to find each other at runtime. This process is termed "master" and will be described shortly. In an effort to manage the complexity of ROS, developers have opted for a microkernel design, where a large number of small tools are used to build and run the various ROS components.

These tools perform assorted tasks: e.g., navigate the source code tree, get and set configuration parameters, visualize the peer-to-peer connection topology, measure bandwidth utilization, graphically plot message data, auto-generate documentation, and so on. This modularity loses some efficiency compared to implementing core services such as a global clock and a logger inside. However, this is more than offset by the gains in stability and complexity management.

The central components of general communication in ROS are as follows:

- Packages
- Nodes
- Messages
- Topics
- Parameters Server
- Master

3.1.1 Packages

Software in ROS is organized in packages. A package might contain anything that logically constitutes a useful module. The goal of these packages it to provide functionality in an easy-to-consume manner so that software can be easily reused. Packages are often lightweight and contribute to the "Thin" aspect of the ROS philosophy.

ROS packages are merely folders that are linked to the ROS file system and can be used to build code into usable executable files under the package umbrella. Many packages have been developed for use in the RISC lab and will be discussed in Chapter 4.

3.1.2 Nodes

Nodes are processes that perform computation. ROS is designed to be modular at a fine-grained scale. Therefore, a system is typically comprised of many nodes. In this context, the term node is interchangeable with software module. Use of the term node arises from visualizations of ROS-based systems at runtime: when many nodes are running, it is convenient to render the peer-to-peer communications as a graph, with processes as graph nodes and the peer-to-peer links as arcs. Nodes communicate with each other by passing messages.

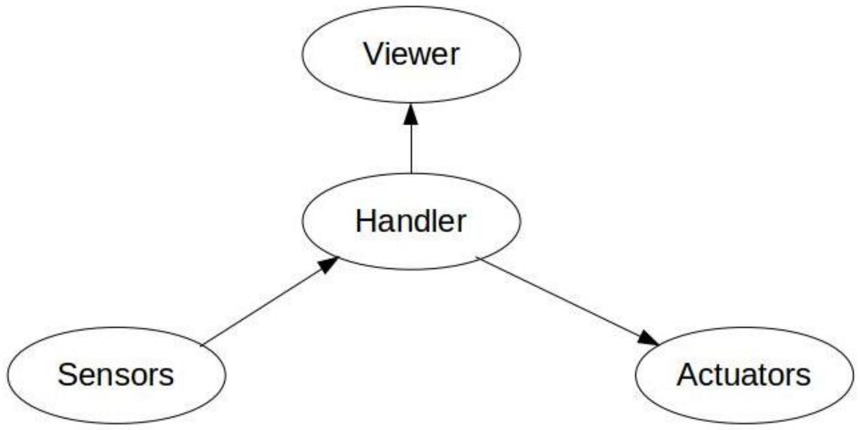


Fig. 3.2: Typical Setup

3.1.3 Messages

A message is a strictly typed data structure. Standard primitive types (integer, floating point, boolean, etc.) are supported, as are arrays of primitive types and constants. Messages can be composed of other messages, and arrays of other messages, nested arbitrarily deep. This is discussed further in Section 3.3.

3.1.4 Topics

A node sends a message by publishing it to a given topic, which is simply a string such as trajectory or controls. A node that is interested in a certain kind of data will subscribe to the appropriate topic. There may be multiple concurrent publishers and subscribers for a single topic, and a single node may publish and/or subscribe to multiple topics. In general, publishers and subscribers are not aware of each other's existence. This allows a flexible node graph that can contain cycles, one-to-many or many-to-many connections.

3.1.5 Parameters Server

A Parameter Server is a shared, multi-variate dictionary that is accessible via network APIs[3]. Nodes use this server to store and retrieve parameters at runtime. As it is not

[3]Application Program Interfaces

designed for high-performance, it is best used for static, non-binary data such as configuration parameters. It is meant to be globally viewable so that tools can easily inspect the configuration state of the system and modify if necessary. This is especially useful for setting simulation or hardware parameters and is widely used in the RISC MAAP.

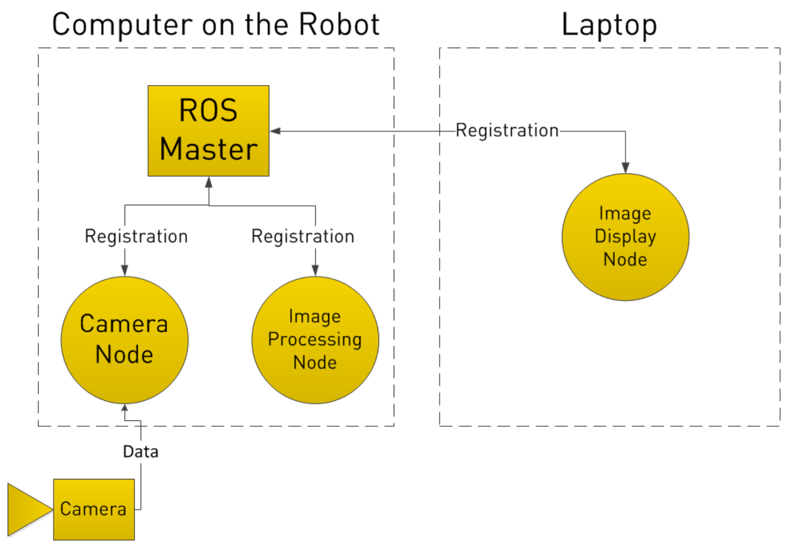


Fig. 3.3: Node Graph

3.1.6 Master

The ROS Master provides naming and registration services to the nodes in the ROS system. It tracks publishers and subscribers to topics. The role of the Master is to enable individual ROS nodes to locate one another. Once these nodes have located each other they communicate with each other peer-to-peer. The Master also provides the Parameter Server. An example image processing node graph is shown in figure 3.3.

3.2 Navigation

The ROS file system is composed of packages and manifests. Packages are the software organization unit of ROS code. Each package can contain libraries, executables, scripts, or other artifacts. Each package also contains a manifest. The manifest is a description of the package and serves to define dependencies between packages and to capture metainformation about the package like version, maintainer, and license. Given the highly modular framework of the ROS file system, navigation via command-line tools such as `ls` and `cd` can be very tedious. Fortunately, ROS developers have provided navigation tools to avoid this. These consist of `rosbash` and `rospack` commands.

3.2.1 rosbash

The following command line utilities are included in `rosbash`:

- `roscd` - change directory starting with package, stack, or location name
- `rosd` - lists directories in the directory-stack
- `rosls` - list files of a ROS package
- `rosed` - edit a file in a package (the default editor is vim)
- `roscp` - copy a file from a package
- `rosrun` - run executables of a ROS package

3.2.2 rospack

`rospack` is a command-line tool for retrieving information about ROS packages available on the file system. It implements a wide variety of commands ranging from locating ROS packages in the file system, to listing available stacks, to calculating the dependency tree of stacks. It is also used in the ROS build system for calculating build information for packages. Some useful commands are:

- `rospack find` - return the absolute path to a package
- `rospack depends` - return a list of all of a package's dependencies
- `rospack depends-on` - return a list of packages that depend on the given package
- `rospack export` - return flags necessary for building and linking against a package

3.3 Message Passing

Nodes communicate with each other by publishing and subscribing to messages via topics. These are generally asynchronous. In ROS, synchronous message passing is called a service.

3.3.1 Subscribers and Publishers

ROS currently supports publishing and subscribing using four very different languages: C++, Python, Octave, and LISP, with other language ports in various states of completion (JAVA, Ruby). The RISC MAAP software is written using C++ and Python. This language neutrality in ROS is possible due to the IDL[4] to describe the messages sent between modules. The IDL uses short text files to describe fields of each message, and allows composition of messages as illustrated by the complete IDL file for a point cloud message:

```
Header header
Point32[] pts
ChannelFloat32[] chan
```

Code generators for each supported language then generate native implementations and are automatically serialized and deserialized by ROS as messages are sent and received. This saves considerable programmer time and reduces errors. The previous 3-line IDL file automatically expands to 137 lines of C++, 96 lines of Python, 81 lines of Lisp, and 99 lines of Octave. Because the messages are generated automatically from such simple text files, it becomes easy to rapidly create custom messages to meet diverse needs. Custom messages used in the RISC MAAP are outlined in Chapter 4. This allows for mixing and matching of subscriber/publisher programming languages.

3.3.2 Services

Synchronous transactions can significantly simplify the design of some nodes. In ROS, this synchronous message passing is called a service, defined by a string name and a pair of

[4]interface definition language

strictly typed messages: one for the request and one for the response. Unlike topics, only one node can advertise a service of any particular name.

3.4 Data Logging

3.4.1 rosbag

`rosbag` is a set of tools for recording from and playing back to ROS topics. It is intended to be high performance and avoids deserialization and reserialization of the messages. It can be used in command-line as well as within C++ or Python code to store data. Current list of supported commands:

- `record` - Record a bag file with the contents of specified topics
- `info` - Summarize the contents of a bag file
- `play` - Play back the contents of one or more bag files
- `check` - Determine whether a bag is playable in the current system, or if it can be migrated
- `fix` - Repair the messages in a bag file so that it can be played in the current system
- `filter` - Convert a bag file using Python expressions
- `compress` - Compress one or more bag files.
- `decompress` - Decompress one or more bag files
- `reindex` - Reindex one or more broken bag files

`rosbag` recording is very lightweight and can handle large data storage at high rates if the bag file is local. For example, storing to the main hard drive would be noticeably easier than if you are attempting to write a bag file through a USB[5] port. In the latter case you

[5] Universal Serial Bus

will likely notice lag for large data sets at high frequency (ex: 2+ HD[6] video streams, sensor and motion capture data).

3.5 Data Visualization

Along with the multilingual features of ROS, there are many methods of visualizing data.

3.5.1 rostopic

`rostopic` is a command-line tool that prints relevant information about a specific topic in the command window. The available commands are:

- `rostopic bw` - bandwidth used by topic
- `rostopic echo` - print messages to screen
- `rostopic find` - find topics by type
- `rostopic hz` - publishing rate of topic
- `rostopic info` - info about active topic
- `rostopic list` - list active topics
- `rostopic pub` - publish data to topic
- `rostopic type` - print topic type

Rostopic can be very useful for debugging or briefly inspecting performance.

3.5.2 rviz

`rviz` is a powerful tool that allows for real time 3D[7] data visualization in consumable form. It integrates robot states, maps, point clouds, trajectories, laser scanners, force plates, etc. This is particularly useful for debugging and getting a big picture view of data quality.

[6]High Definition
[7]3-Dimensional

It incorporates the robot simulation tool Gazebo. This allows use and sharing robot 3D CAD[8] models.

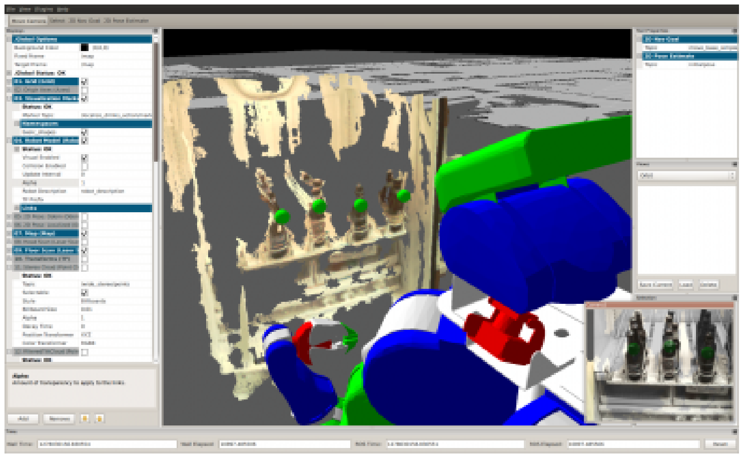
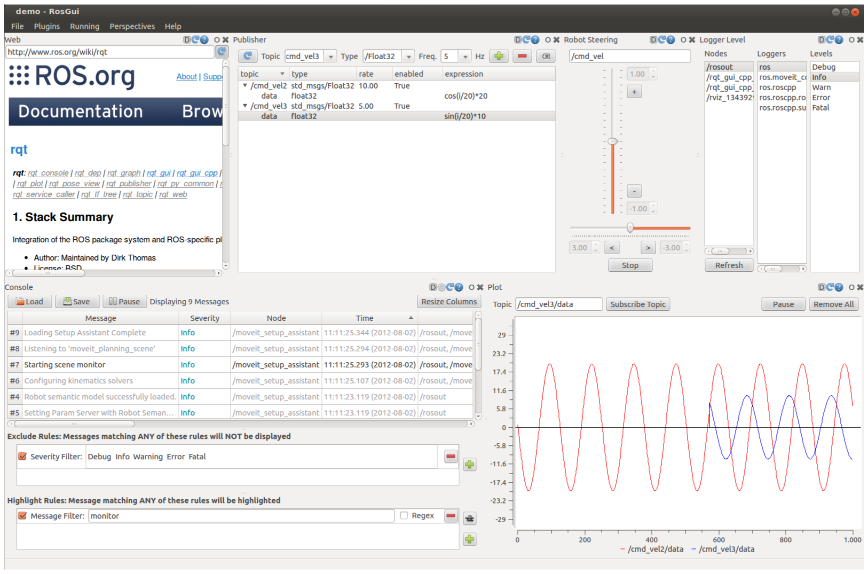


Fig. 3.4: rviz Example

3.5.3 Rqt

`rqt` is a flexible GUI[9] that integrates various plugins for data visualization. This simplifies management of distributed tasks in one modular GUI. The features available using `rqt` are extensive and will not be covered here.

[8]Computer Aided Drafting
[9]Graphical User Interface



http://www.ros.org/wiki/rqt

ROS.org About | Supp

Documentation Brow

rqt

rqt: rqt_console | rqt_dep | rqt_graph | rqt_gui | rqt_gui_cpp | | rqt_plot | rqt_pose_view | rqt_publisher | rqt_py_common | rqt_service_caller | rqt_tf_tree | rqt_topic | rqt_web

1. Stack Summary

Integration of the ROS package system and ROS-specific pl

- Author: Maintained by Dirk Thomas
- License: BSD

Topic cmd_vel3 Type /Float32 Freq. 5 Hz

topic	type	rate	enabled	expression
/cmd_vel2	std_msgs/Float32	10.00	True	
data	float32			cos(i/20)*20
/cmd_vel3	std_msgs/Float32	5.00	True	
data	float32			sin(i/20)*10

/cmd_vel

1.00 + - -1.00 3.00 < > -3.00 Stop

Nodes	Loggers	Levels
/rosout	ros	Debug
/rqt_gui_cpp_	ros.moveit_c	Info
/rqt_gui_cpp_	ros.roscpp	Warn
/rviz_134392	ros.roscpp.ro	Error
	ros.roscpp.su	Fatal

Refresh

Load Save Pause Displaying 9 Messages Resize Columns

	Message	Severity	Node	Time	
#9	Loading Setup Assistant Complete	Info	/moveit_setup_assistant	11:11:25.344 (2012-08-02)	/rosout, /move
#8	Listening to 'moveit_planning_scene'	Info	/moveit_setup_assistant	11:11:25.294 (2012-08-02)	/rosout, /move
#7	Starting scene monitor	Info	/moveit_setup_assistant	11:11:25.293 (2012-08-02)	/rosout, /move
#6	Configuring kinematics solvers	Info	/moveit_setup_assistant	11:11:25.107 (2012-08-02)	/rosout, /move
#4	Robot semantic model successfully loaded.	Info	/moveit_setup_assistant	11:11:23.119 (2012-08-02)	/rosout
#5	Setting Param Server with Robot Seman…	Info	/moveit_setup_assistant	11:11:23.119 (2012-08-02)	/rosout

Exclude Rules: Messages matching ANY of these rules will NOT be displayed

Severity Filter: Debug Info Warning Error Fatal

Highlight Rules: Message matching ANY of these rules will be highlighted

Message Filter: monitor ☐ Regex

Topic /cmd_vel3/data Subscribe Topic Pause Remove All

Fig. 3.5: rqt Example

3.5.4 Python

Python enables the generation of plots using matplotlib, plotly, and other similar libraries [18]. Many plugins provided for `rqt` are generated using these libraries. However, these can be used directly in code to generate custom plots on demand.

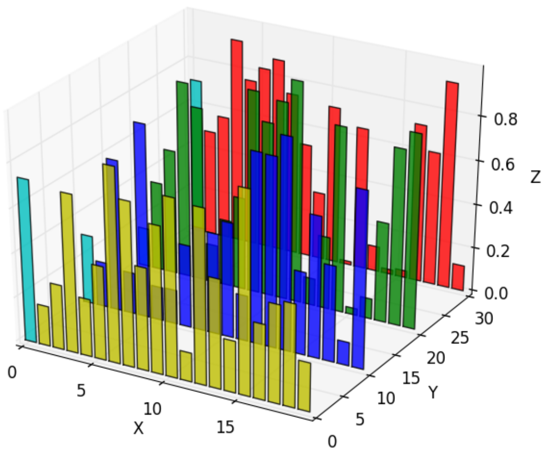


Fig. 3.6: Python Plotting Example

3.6 Serial Interfacing

`rosserial` is a general protocol for wrapping standard ROS serialized messages and sending them over serial links such as a UART[10] or network socket.

3.6.1 protocol

The `rosserial` protocol is aimed at point-to-point ROS communications over a serial transmission line. It uses the same serialization/de-serialization as standard ROS messages, simply adding a packet header and tail which allows multiple topics to share a common serial link.

3.6.2 rosserial_client

`rosserial_client` contains the generic client-side `rosserial` implementation. It is designed for microcontrollers and it can run on any processor for which an ANSI[11] C++ compiler and a serial port connection to a computer running ROS is used.

3.6.3 rosserial_arduino

`rosserial_arduino` contains Arduino-specific extensions required to run `rosserial_client` on an Arduino.

3.6.4 Arduino IDE Setup

The open-source Arduino software (IDE)[12] can be modified to allow ROS communication. This can be done as follows.

- Install recent Arduino IDE
- Install `rosserial` and `rosserial_arduino` packages

[10]universal asynchronous receiver/transmitter
[11]American National Standards Institute
[12] Integrated Development Environment

- Add ros_lib to the Arduino Environment using `rosrun`

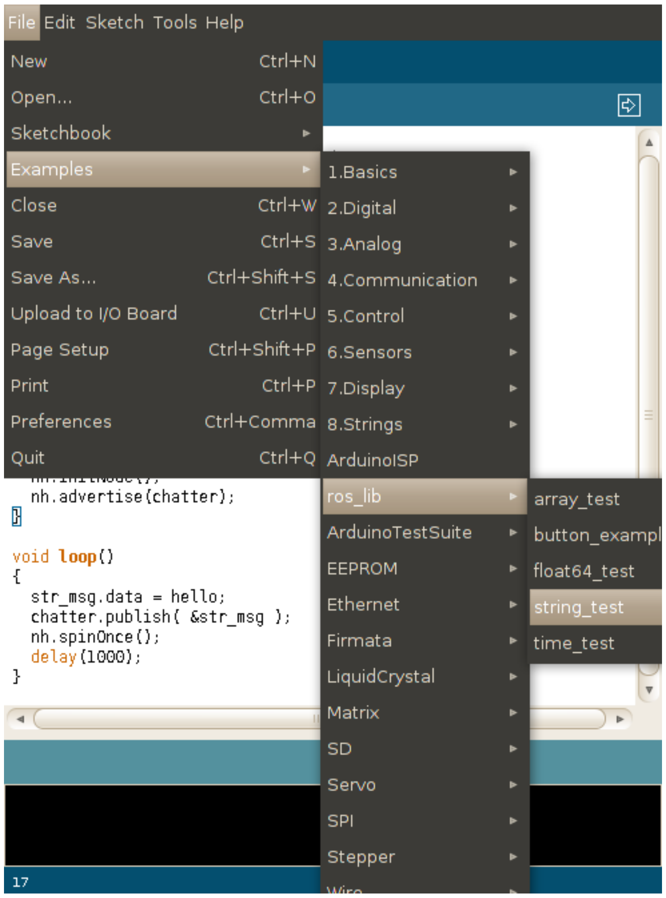
```
rosrun rosserial_arduino make_libraries.py .
```

- Add custom messages from `risc_msgs` package to the Arduino Environment using `rosrun`

```
rosrun rosserial_client make_library.py <>/libraries risc_msgs
```

  <> represents the file path to saved sketches used in the Arduino IDE.

- Restart Arduino IDE and check and see if desired libraries are present



```
  nh.advertise(chatter);
}

void loop()
{
  str_msg.data = hello;
  chatter.publish( &str_msg );
  nh.spinOnce();
  delay(1000);
}
```

Fig. 3.7: ROS Libraries

More information on how this is used for hardware interfacing in the RISC MAAP will be provided in Chapter 5.

3.7 Transforms

`tf`[13] is a package that lets the user keep track of multiple coordinate frames over time. `tf` maintains the relationship between coordinate frames in a tree structure buffered in time, and lets the user transform vector data or points between any two coordinate frames at any desired point in time. A robotic system, such as a quadrotor, typically has many 3D coordinate frames that change over time, such as a world frame, vehicle frame, body frame, etc. These frames of reference specific to the quadrotor are covered in detail in Chapter

[13] Tully Foote

6. `tf` keeps track of all these frames over time. `tf` can operate in a distributed system. This means all the information about the coordinate frames of a robot are available to all ROS components on any computer in the system. There is no central server of transform information.

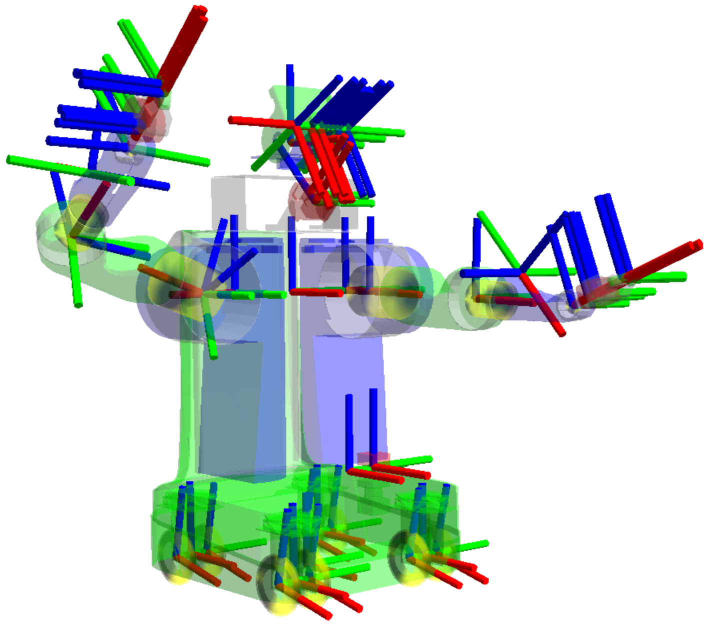


Fig. 3.8: Transforms Visualization

3.8 Roslaunch

ROS provides a tool called `roslaunch`, which reads an XML[14] description of a graph and instantiates the graph on the cluster, optionally on specific hosts. The end-user experience of launching the navigation system then boils down to a single string of script such as:

```
roslaunch package_name launch_file.xml
```

This tool allows for a single Ctrl-C key stroke to close all associated processes. This functionality significantly aids sharing and reuse of large demonstrations of integrative robotics research.

[14]Extensible Markup Language

Chapter 4

RISC MAAP Framework

In this chapter the motivations for development of the Robust Intelligent Sensing and Control Multi Agent Analysis Platform are provided. Goals of the overall system and how these are achieved is addressed followed by an outline of basic vision tools developed for research.

Many aspects of robotics are influenced by unpredictable dynamic environments in ways that make reliable estimation and control challenging. Consequently, robotics researchers have turned towards controlled testing platforms to validate theory [2, 3, 5]. While many testbeds have used supplementary open source software, to our knowledge, no testbed itself has been made open source. Therefore, those who wish to accelerate research using these controlled environments often require months or even years of development and setup before a realizable platform is available.

The central goal of the RISC MAAP is to provide an open-source-adaptable-expandable testbed framework to allow researchers and students to spend their time on the specific research/academic problems rather than on implementing the many necessary, but unrelated parts of the system. To realize this goal, similar to ROS, these four aspects must be addressed [17]:

- Complexity Management
- Information Access
- Advanced Tools
- Open Source

Before describing how the RISC MAAP deals with these specific elements an outline of the RISC MAAP structure will be provided.

4.1 Structure

The overall structure of the RISC MAAP is displayed in figure 4.1. The cameras provide data to the Cortex software which in turn provides marker position data. This data is gathered along with wireless sensor data and processed in ROS using RISC MAAP software. The RISC MAAP software then generates control inputs and sends corresponding wireless commands to the agent. In this example, the AR.Drone is used as the agent. The AR.Drone will be described in detail in Chapter 5.

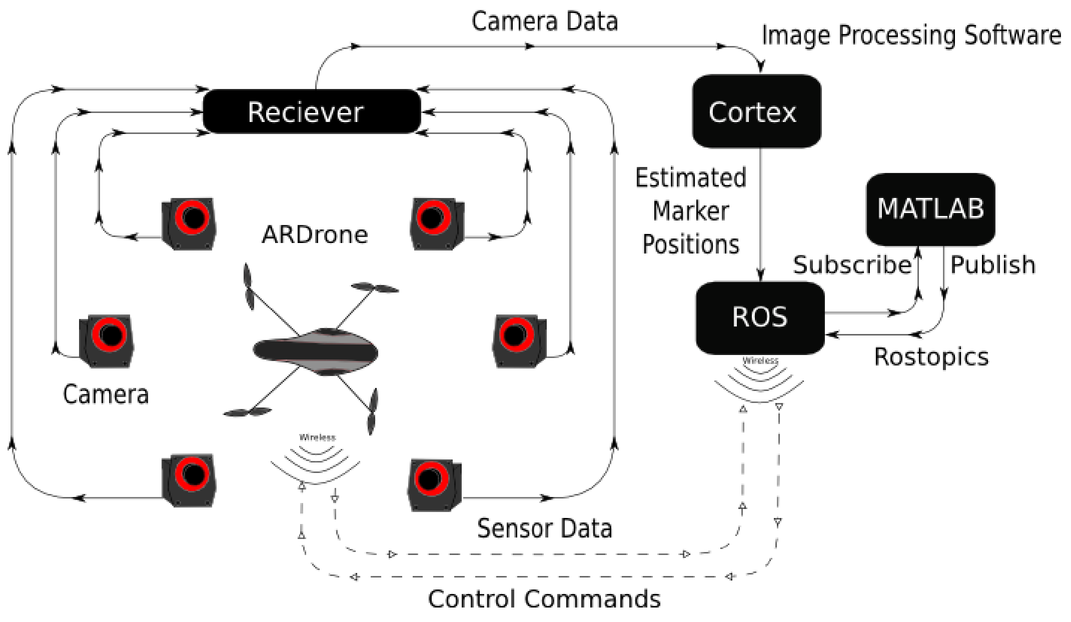


Fig. 4.1: Camera Feedback System

In figure 4.1 ROS is treated as a “black box” for the RISC MAAP software. This software can be described by the combined RISC MAAP packages. These packages that have been developed to meet the needs of the RISC lab and encompass key areas of research:

- Control
- Estimation
- Simulation
- Visualization

- Data Management

4.1.1 RISC MAAP Packages

Control

- `risc_control`: This package contains the software for control algorithms, trajectory generation, and frame of reference transformations. Some control strategies and trajectories used will be discussed in Chapters 6 and 7.

Estimation

- `risc_estimation`: This package contains the software for the state estimation/filtering algorithms for providing feedback necessary for proper control such as kalman filtering, least squares, low pass filters, etc. This will be discussed in greater detail in Section 4.4.

Simulation

- `risc_simulator`: This package currently contains the python based simulations for systems of interest. Quadrotor and 3D Inverted Pendulum are the simulations provided thus far and mimic the actual hardware interfacing using the ROS communication setup.

Visualization

- `risc_visual`: This package provides access to various image processing and visualization tools. Among these are cortex visualization, edge detection, optical flow, color thresholding, and landmark detection. This will be discussed in greater detail in Section 4.4.

Data Management

- `cortex_ros`: This package automatically establishes communication with the Motion Analysis Software (Cortex) sharing the same network as the computer running ROS. It provides marker data and names of templates set up within that system.

- `risc_msgs`: This package is used to define all custom messages that aid in complexity management.

- `data_extraction`: This package was originally authored by Shane Lynn and extracts data from a `rosbag` file and converts it to a csv[1] file for use in Matlab or elsewhere. This has been significantly modified to include many of the RISC MAAP custom messages. This is useful for providing data to those unfamiliar with ROS or for use in Matlab.

4.1.2 Other Packages:

These other packages are also used in the RISC MAAP software. However, they did not originate here and are all well documented on ROS or github websites. Therefore, no extensive description will be provided herein.

- `ardrone_autonomy`: Provides ardrone driver and sensor data.

- `roscopter`: Incorporates the Mavlink library for interfacing with the PixHawk using the USB radio provided by 3DR Robotics. The PixHawk is a central component to the IRIS+ quadrotor used in the RISC lab.

- `uvc_cam`: Enables publishing web cam video feed in ROS using the libuvc_camera driver. This is very useful for testing image processing code using built-in or USB web cams.

[1]Comma Separated Values

4.2 Complexity Management

Complexity management is addressed by using carefully constructed data structures that are intuitive and expandable. These can be easily cycled through using for loops. This permits a solution for a single agent to be quickly integrated to a multi agent system. The structures are multi-lingual due to the IDL message definitions. All of the files referenced below can be found in the `risc_msgs` package.

4.2.1 Custom Message Structures:

- Motion Capture Data: Defined by these three files:
  - Mocap_marker.msg
  - Mocap_markers.msg
  - Mocap_data.msg

  Mocap_data Structure:

```
[risc_msgs/Mocap_data]:
std_msgs/Header header
  uint32 seq
  time stamp
  string frame_id
risc_msgs/Mocap_markers[] Obj
  string name
  float64 residual
  risc_msgs/Mocap_marker[] marker
    float64 x
    float64 y
    float64 z
```

- Quadrotor States: Defined by these two files:
  - States.msg
  - Cortex.msg

  Cortex Structure:

```
 [risc_msgs/Cortex]:
std_msgs/Header header
  uint32 seq
  time stamp
  string frame_id
risc_msgs/States[] Obj
  string name
  bool visible
  float64 x
  float64 y
  float64 z
  float64 u
  float64 v
  float64 w
  float64 phi
  float64 theta
  float64 psi
  float64 p
  float64 q
  float64 r
```

- Quadrotor Control: Defined by these two files:
  - Control.msg

  - Controls.msg

  Controls Structure:

```
[risc_msgs/Controls]:
std_msgs/Header header
  uint32 seq
  time stamp
  string frame_id
risc_msgs/Control[] Obj
  string name
  float64 phi
  float64 theta
  float64 psi
  float64 T
```

- Bearing and Visual Data: Defined by these six files:
  - Angle.msg
  - Angles.msg
  - Observed_angles.msg
  - Risc_roi.msg
  - Risc_rois.msg
  - Observed_rois.msg

  Observed_angles Structure:

```
[risc_msgs/Observed_angles]:
std_msgs/Header header
  uint32 seq
  time stamp
  string frame_id
risc_msgs/Angles[] Obj
  string name
  risc_msgs/Angle[] landmarks
    string name
    bool visible
    float64 azim
    float64 elev
  risc_msgs/Angle[] quads
    string name
    bool visible
    float64 azim
    float64 elev
```

Observed_rois Structure:

```
[risc_msgs/Observed_rois]:
std_msgs/Header header
  uint32 seq
  time stamp
  string frame_id
risc_msgs/Risc_rois[] Obj
  string name
  risc_msgs/Risc_roi[] landmarks
    string name
    bool visible
    int32 x
    int32 y
    float32 width
    float32 height
    float64 angle
  risc_msgs/Risc_roi[] quads
    string name
    bool visible
    int32 x
    int32 y
    float32 width
    float32 height
    float64 angle
```

- Trajectories: Defined by these two files:
  - Trajectory.msg
  - Trajectories.msg

  Trajectories Structure:

```
[risc_msgs/Trajectories]:
std_msgs/Header header
  uint32 seq
  time stamp
  string frame_id
risc_msgs/Trajectory[] Obj
  string name
  float64 x
  float64 y
  float64 z
  float64 psi
  float64 xdot
  float64 ydot
  float64 zdot
  float64 psidot
  float64 xddot
  float64 yddot
  float64 zddot
  float64 psiddot
```

- Landmarks: Defined by these two files:
  - Landmark.msg
  - Landmarks.msg

  Landmarks Structure:

```
[risc_msgs/Landmarks]:
std_msgs/Header header
  uint32 seq
  time stamp
  string frame_id
risc_msgs/Landmark[] Obj
  string name
  float64 x
  float64 y
  float64 z
```

- Pendulum Estimated States: Defined by this file:
  - Pen_with_cov.msg

Pen_with_cov Structure:

```
[risc_msgs/Pen_with_cov]:
std_msgs/Header header
  uint32 seq
  time stamp
  string frame_id
bool visible
float64 x
float64 y
float64 xdot
float64 ydot
float64[16] cov
```

The general convention among these structures permit two means of object identification via index and string name. This redundancy allows for increased robustness and ensures that sent/retrieved data is to/from the intended object. This consistency greatly reduces complexity in code length as well as comprehensibility of multi agent system data.

4.3 Information Access

Access to all relevant information is made readily accessible using the tools already provided by ROS described in Chapter 3. These tools include:

- `rostopic`
- `rviz`
- `rqt`
- `rosbag`

4.4 Advanced Tools

While many advanced robotics tools are available through ROS, the following are being used on the RISC MAAP:

- General Estimation
- OpenCV
- Camera Calibration
- Camera to Body Frame Estimation
- Landmark Recognition

4.4.1 General Estimation

In the package `risc_estimation` two general methods of estimation are widely used: Least Squares and Extended Kalman Filter (EKF).

Least Squares Estimation

The method of least squares is a standard approach to gain the best estimate of an overdetermined system such as: $y = Ax$, where y and A are known and A is a tall non-invertible matrix. The method of least squares provides an overall solution the minimizes the sum of the squared error:

$$\min_{x}(y - Ax)^T(y - Ax) \tag{4.1}$$

$$\hat{x} = (A^T A)^{-1} A^T y \tag{4.2}$$

Where $\hat{x}$ is the solution that minimizes the sum of the squared error. This method can be used to estimate the coefficients for any polynomial given an overdetermined data set. Using a vector of stored position data method of least squares can estimate the slope or velocity of the data. This is implemented in states_estimation.py in the `risc_estimation` package using the Numpy library.

Extended Kalman Filter

The EKF is an extension of the well-known Kalman Filter as described in [19]. An EKF provides an approximate linearized estimate of the states, despite the nonlinear characteristics of the system [20]. A generic EKF can be summarized as follows:

Propagate:

- $\hat{x}_{i+1} = f(\hat{x}_i, u_i)$

  $\hat{x}_i =$ state estimate at iteration i

  $f(\hat{x}_i, u_i) =$ Nonlinear model

- $\hat{P}_{i+1} = \Phi_i P_i \Phi_i^T + Q_{d,i}$

  $\Phi_i = I + \frac{\partial f(\hat{x}_i, u_i)}{\partial x}|_{\hat{x}_i, u_i} \Delta t$ (Approximate Jacobian)

  $Q_{d,i} =$ Process noise covariance

  $P_i =$ Covariance of estimate

Update:

- Measurement: z_i
- Residual: $y_i = z_i - \hat{x}_i$
- Residual Covariance: $\mathfrak{R}_i = H_i P_i H_i^T + R$

  Where R is the measurement noise covariance and H_i is the observation model matrix.
- Kalman Gain: $K_i = P_i^- H_i^T \mathfrak{R}_i^{-1}$
- Update State: $\hat{x}_i^+ = \hat{x}_i^- + K_i y_i$

- Update Covariance: $P_i^+ = (I - K_i H_i)P_i^-$

Thus, $\hat{x}_i^+$ is the new estimated mean and P_i^+ is the corresponding covariance.

Implementation of this filter on the RISC MAAP generally followed this form. One of the many benefits of using this method is the possibility of data loss detection using a chi-square test given a state mean x and covariance P with measurement noise covariance R [21]. The description of the EKF design for our specific research using the RISC MAAP is provided in Chapter 7.

4.4.2 OpenCV

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in commercial products. OpenCV is maintained by WillowGarage and is fully integrated into ROS. Some of the algorithm templates available in the `/risc_visual` package are:

- Edge Detection
- Drawing
- Optical Flow
- Face Recognition
- Thresholding and Image Segmentation
- Camshift Tracking
- Shape Detection

4.4.3 Camera Calibration

The majority of cameras on the market today are built at relatively low cost. This results in imperfections that are manifest by distortion to the generated images. Two major distortions are radial distortion and tangential distortion.

Due to radial distortion, straight lines will appear curved. Its effect increases away from the center of image. An image of a doorway taken from the AR.Drone front camera is shown below. Lines expected to be straight near the perimeter are bulging out uncharacteristic of reality.

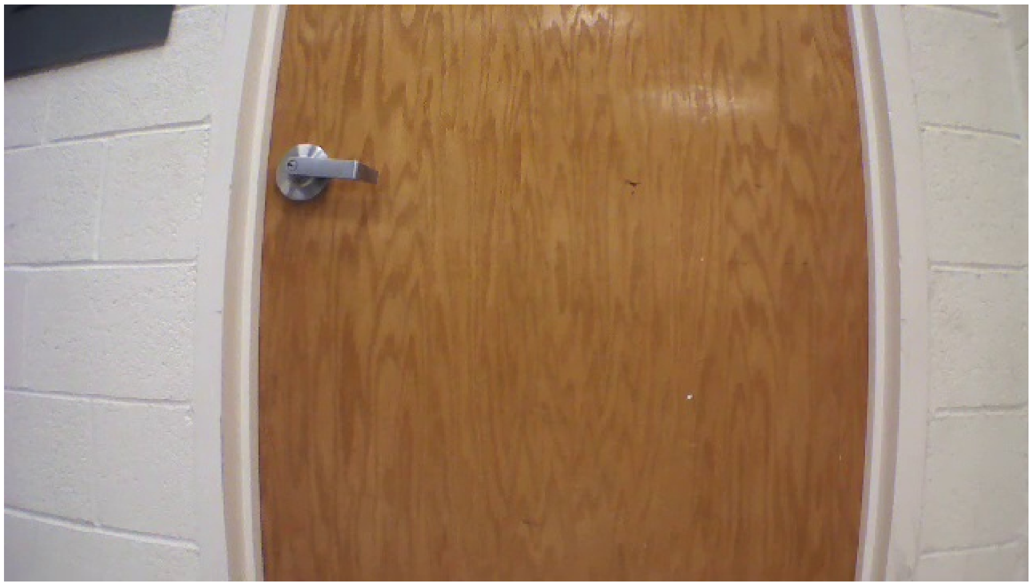


Fig. 4.2: AR.Drone Front Camera View

This distortion can be described as follows:

$$x_{corrected} = x(1 + k_1r^2 + k_2r^4 + k_3r^6)$$

$$y_{corrected} = y(1 + k_1r^2 + k_2r^4 + k_3r^6)$$

Similarly, another distortion is the tangential distortion which occurs because image taking lense is not aligned perfectly parallel to the imaging plane. Therefore, some areas in image may appear closer than expected.

This distortion can be described as follows:

$$x_{corrected} = x + [2p_1xy + p_2(r^2 + 2x^2)]$$

$$y_{corrected} = y + [p_1(r^2 + 2y^2) + 2p_2xy]$$

Five parameters known as distortion coefficients are given by:

$$\text{Distortion coefficients} \; = \begin{bmatrix} k_1 & k_2 & p_1 & p_2 & k_3 \end{bmatrix}$$

In addition to this, intrinsic parameters of a camera must be known to fully describe the camera. Intrinsic parameters are specific to a camera. It includes information like focal length (f_x, f_y), optical centers (c_x, c_y) etc. It is also called the camera matrix. It depends on the camera only. Once determined, it can be stored for future purposes. It is expressed as a 3x3 matrix:

$$\text{camera matrix} \; = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

In order to find all these parameters, sample images of a well-defined pattern (eg, chess board) must be provided. Then a solution can be found given enough data. Fortunately, ROS has integrated an OpenCV-based package called camera_calibration that provides a convenient gui interface to accomplish this. Using a predefined checkerboard size this package can provide a matrix description of a camera that will take an image like that shown in figure 4.3 and correct it to produce images like that shown in figure 4.4.

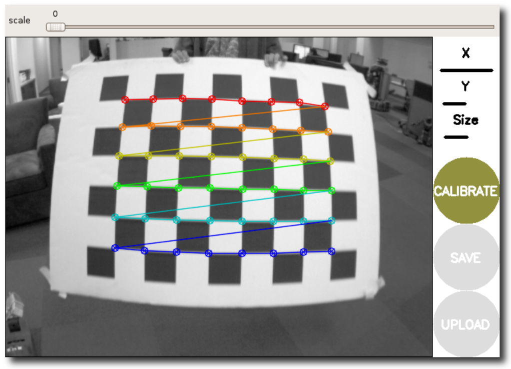


Fig. 4.3: Distorted Image

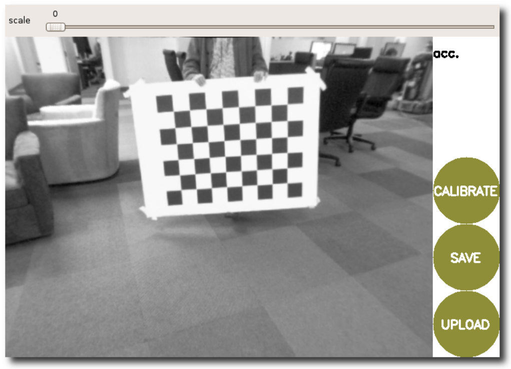


Fig. 4.4: Corrected Image

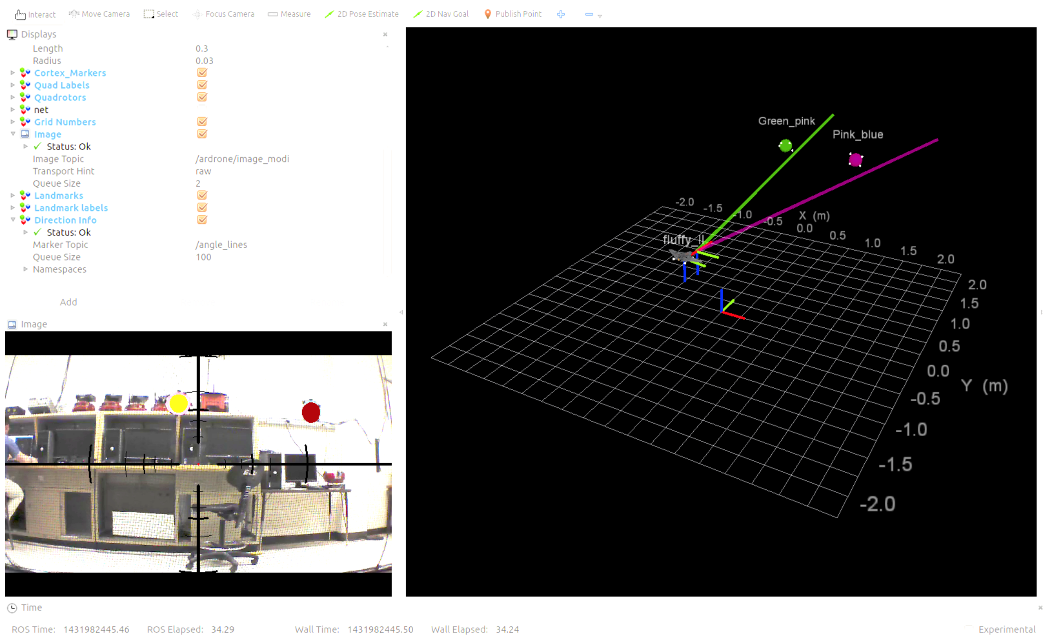
4.4.4 Camera to Body Frame Estimation

Camera calibration takes care of intrinsic characteristics. However, in order to interpret camera data with respect to an agent, a mapping from camera frame to body frame is required. For example, the body frame of a quadrotor is defined as $\mathcal{F}^b$. The origin is the center-of-gravity, $\hat{i}^b$ points out the nose of the airframe, $\hat{j}^b$ points out the right wing, and $\hat{k}^b$ points out the belly. In contrast, the camera frame of the AR.Drone $\mathcal{F}^c$ is centered at the focal point of the camera with an associated rotation relative to the body frame.

A correct mapping of the translation and rotation from body frame to camera frame allows for proper interpretation of camera data. This can be a challenging process and is a vast area of current research [22, 23, 24]. Using the RISC MAAP, a novel gui-based tuning method has been developed to achieve this relative pose calibration. Using landmarks tracked using the camera sensor and motion capture system, a 3D representation of the expected position of the landmarks can be compared to the actual using rviz. The expected position of the landmarks relative the camera frame is shown by rays and the actually landmarks are represented using spheres as shown in figure 4.5. The user then tunes the three axes rotation translation values to visually match the expected value to the true data streaming from motion capture data. The result is a highly accurate camera pose estimate.

Given proper use of this relative pose calibration the camera data can be used to gain azimuth and elevation angles of targets reliably. In summary, pixel to angle mapping can be done as follows:

- Calibrate the camera, obtaining the camera matrix and distortion parameters.
- Remove the nonlinear distortion from the pixel positions of interest.
- Back-project the pixels of interest into rays (unit vectors with the tail at the camera center) in camera 3D coordinates, by multiplying their pixel positions in homogeneous coordinates times the inverse of the camera matrix.
- The angle is between the above vectors and $\begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$, the vector associated to the camera's focal axis.



Length 0.3
Radius 0.03
Cortex_Markers ✓
Quad Labels ✓
Quadrotors ✓
net
Grid Numbers ✓
Image ✓
Status: Ok
Image Topic /ardrone/image_modi
Transport Hint raw
Queue Size 2
Landmarks ✓
Landmark labels ✓
Direction Info ✓
Status: Ok
Marker Topic /angle_lines
Queue Size 100
Namespaces

ROS Time: 1431982445.46 ROS Elapsed: 34.29 Wall Time: 1431982445.50 Wall Elapsed: 34.24

Fig. 4.5: Camera Frame to Body Frame Calibration

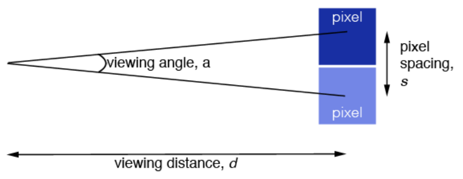


Fig. 4.6: Angle to Pixel Mapping

This can also be done in the reverse direction when adjusted for data transport delay. This could possibly be used for automatic image labeling or for providing highly accurate synthetic camera measurements for use in research without the need of performing real-time image processing.

4.4.5 Landmark Recognition

In order to perform autonomous visual localization research reliable object detection and tracking is necessary. Real-time robust object tracking is an important and active research area in robotics. This is a complex problem due to:

- loss of information caused by projection of the 3D world on a 2D image
- noise in image
- complex object motion

- nonrigid or articulated nature of objects
- partial and full object occlusions
- complex object shapes
- scene illumination changes
- real-time processing requirements

There are many approaches to object tracking and detection [25]. In the RISC Lab reliable "Landmark" object tracking is required to be robust to invariant lighting, backgrounds, blur, loss of field of view, and also remains computationally efficient enough to run in real-time. A solution has been devised using the following tools:

- Relatively Invariant Shape and Coloring
- Broad Hue Saturation Value Thresholding
- Contour Detection and Sorting
- Perimeter Histogram Comparison

Relatively Invariant Shape and Coloring

The ideal invariant shape would be a sphere. However, a sphere can be difficult to work with due to mounting difficulties and multicolor configurations become variant to perspective. Multicolored discs were chosen as they can always be described as an ellipse when projected from 3D to a 2D image plane. Coloring was chosen to use only two color rings significantly separated with respect to the visual spectrum. Example landmarks are shown in figures 6.2, 6.3, and 6.4.

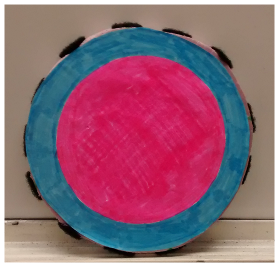


Fig. 4.7: Landmark 1

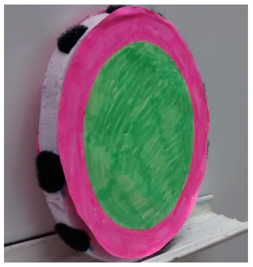


Fig. 4.8: Landmark 2

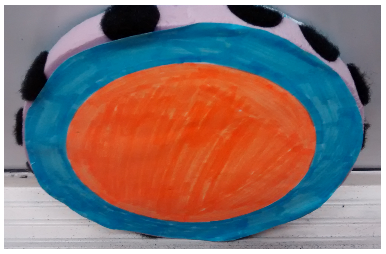


Fig. 4.9: Landmark 3

Broad Hue Saturation Value Thresholding

Image data is generally provided using the Red Green Blue (RGB) color model. This significantly simplifies display as it mimics how are eyes perceive images. However, it does not inherently separate luma, or the image intensity, from chroma similar to how our brains interpret what is seen. This is remedied by using the Hue Saturation Value (HSV) color model. Obtaining this model from RGB values can be done by the following transformation:

$$V_{max} \leftarrow \max(R, G, B)$$

$$V_{min} \leftarrow \min(R, G, B)$$

$$L \leftarrow \frac{V_{max} + V_{min}}{2}$$

$$S \leftarrow \begin{cases} \frac{V_{max}-V_{min}}{V_{max}+V_{min}} & ifL < 0.5 \\ \frac{V_{max}-V_{min}}{2-(V_{max}+V_{min})} & ifL \geq 0.5 \end{cases}$$

$$H \leftarrow \begin{cases} \frac{60(G-B)}{S} & ifV_{max} = R \\ \frac{120+60(B-R)}{S} & ifV_{max} = G \\ \frac{240+60(R-G)}{S} & ifV_{max} = B \end{cases}$$

$$V \leftarrow V_{max}$$

Using an 8-bit image the resulting values are:

$$\begin{cases} H = & H/2 \\ S = & S * 255 \\ V = & V * 255 \end{cases}$$

Using this convention a broad image mask can be created as shown in figure 4.11 from the original image shown in figure 4.10.

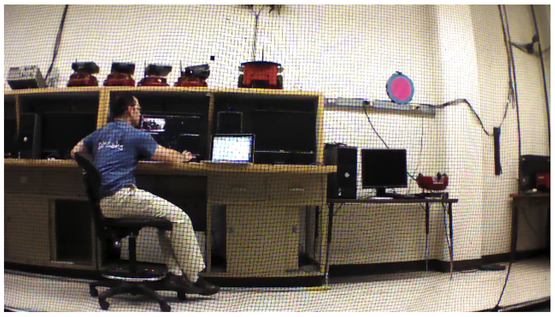


Fig. 4.10: Original Image

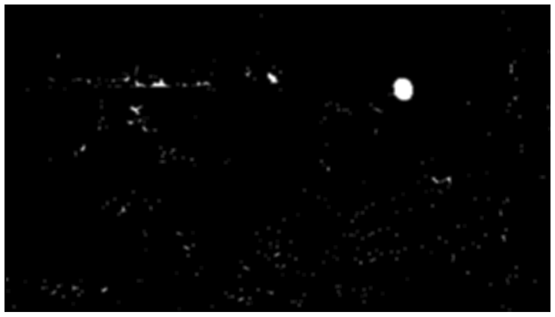


Fig. 4.11: Hue Threshold

Contour Detection and Sorting

Contour detection is supported by the OpenCV library and uses the algorithm described in [26]. The method used is the second algorithm described by Suzuki. These contours are found using border-following algorithms by determining the surroundness relations among the borders of a binary image. Suzuki's second algorithm follows only the outermost borders. As suggested by Suzuki, this can be effectively used in component counting, shrinking, and topological structural analysis of binary images. This algorithm provides contour or blob descriptions that can easily be sorted by size and shape.

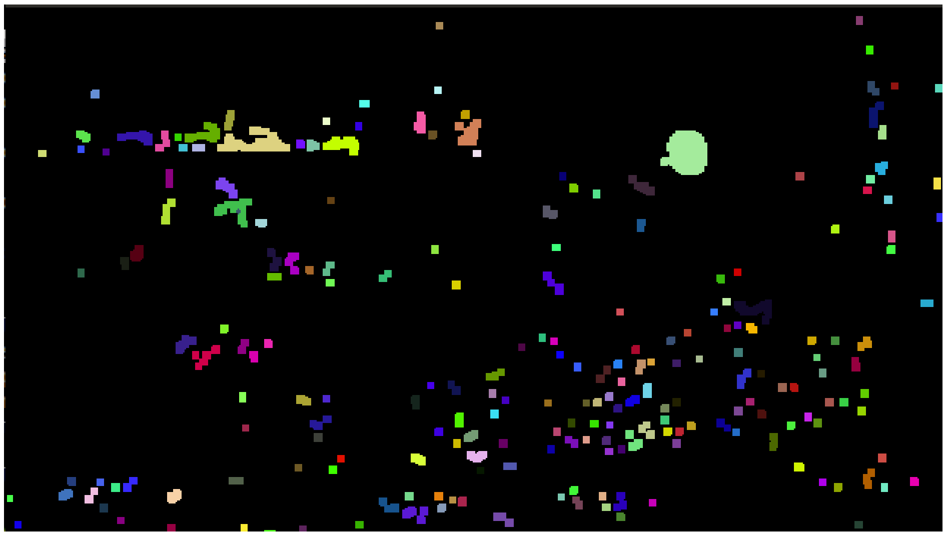


Fig. 4.12: Contours Blobs

These blobs, shown in figure 4.12, can be sorted by size threshold and similarity to an ellipse using the moment of the shape. Given a continuous function f(x,y) defining a shape the moment of order $(p+q)$ is defined as:

$$M_{pq} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x^p y^q f(x,y)\, dx\, dy \tag{4.3}$$

This is implementation is described in [27]. The result is an assortment of candidate blobs organized such that a probability of landmark existence can be determined.

Perimeter Histogram Comparison

To further determine if a landmark truly exists, these candidate blobs can be used to build a mask about the perimeter as shown in figure 4.13. This mask can then be used to gain a histogram of bordering pixels to the blob as shown in figure 4.14. Since the landmark color configurations are already known this results in a positive or negative detection based on a threshold determined by the user.

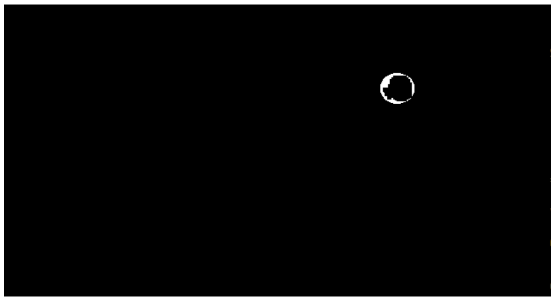


Fig. 4.13: Outer Ring

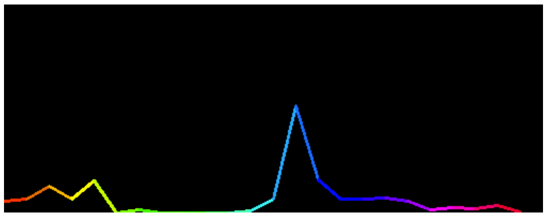


Fig. 4.14: Color Histogram

While this process may seem involved this has shown to be relatively robust to invariant lighting, backgrounds, blur, loss of field of view, and also remains computationally efficient enough to run in real-time.

4.5 Open Source

Nothing is really open source unless it is accessible and can be easily studied. The entire RISC MAAP software will be made available on GitHub noted as the world's largest code hoster [28] [29]. GitHub is a web-based Git repository hosting service, which offers all of the distributed revision control and SCM[2] functionality of Git as well as added features. Using GitHub, all of the code is easily inspected, as is its entire history.

[2]source code management

Chapter 5
Current Hardware Systems

In this chapter hardware systems currently used in the RISC MAAP are outlined including the Parrot AR.Drone and 3DR Robotic's IRIS+. Interface and control are discussed along with relevant specifications.

The RISC lab's systems of interest have recently centered around Vertical Take-Off and Landing (VTOL) Unmanned Aerial Vehicles (UAVs). Two quadrotor platforms are currently set up and in use: Parrot AR.Drone and 3DR Robotic's IRIS+.

5.1 AR.Drone

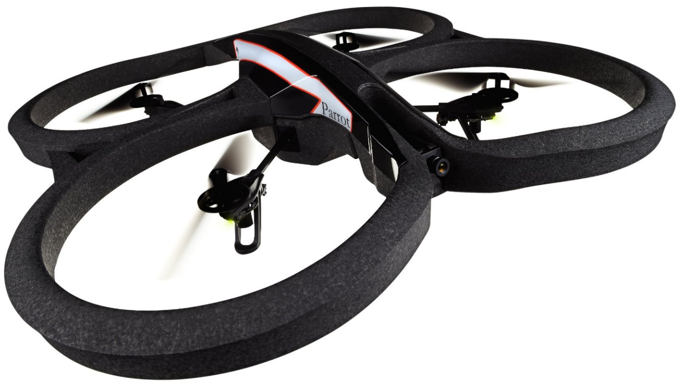


Fig. 5.1: Parrot AR.Drone

The AR.Drone shown in figure 5.1 is an electrically powered quadcopter initially intended for augmented reality games. As shown in figure 5.2 it consists of a carbon-fiber support structure, plastic body, four high-efficiency brushless motors, sensor and control board, two cameras and various removable hulls. This model is designed to be lightweight, user-friendly and safe. It includes the safety feature of instantly locking the propellers in case of a foreign body contact, while also assisting the user with difficult maneuvers such as takeoff and landing. The user can set directly its roll, pitch, yaw, and vertical speed while the on-board controller adjusts the motor speeds to stabilize the drone at the required pose. The drone can achieve speeds over 5 m/s and its battery provides enough energy up to 13 minutes of continuous flight [30].

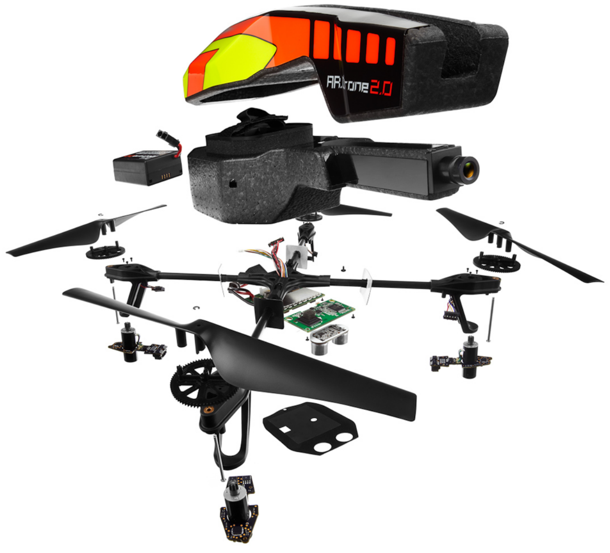


Fig. 5.2: AR.Drone Hardware

The main on-board computer is based on the ARM9 processor running at 468MHz with 128 MB of DDR[1] RAM[2] running at 200MHz. The sensory equipment consists of a 6-degree-of-freedom inertial measurement unit, sonar-based altimeter, and two cameras. The first camera is aimed forward and provides 640 x 480 pixel color image. The second one, mounted on the bottom, provides color image with 176 x 144 pixel resolution [30].

Once the AR.Drone is initialized, an ad-hoc WiFi appears and an external computer might connect to it using a fetched IP[3] address from the drone DHCP[4] server. Using the ROS package ardrone_autonomy the ardrone_driver may be initialized to communicate with the vehicle to retrieve sensor data as well as send desired roll, pitch, yaw rate, and vertical

[1]Double Data Rate
[2]Random Access Memory
[3]Internet Protocol
[4]Dynamic Host Configuration Protocol

speed. The channel receives commands at 30 Hz. Sensor data (including cameras) can be received at rates up to 35 Hz.

Getting Data from AR.Drone

As soon as the ardrone_autonomy driver is running, the corresponding Navdata message will be published as a rostopic. This data includes:

- batteryPercent: The remaining charge of the drone's battery (%)
- state: The Drone's current state: 0: Unknown, 1: Inited, 2: Landed, 3,7: Flying, 4: Hovering, 5: Test, 6: Taking off, 8: Landing, 9: Looping
- rotX: Roll Position (In Degrees)
- rotY: Pitch Position (In Degrees)
- rotZ: Yaw Position (In Degrees)
- magX, magY, magZ: Magnetometer readings
- pressure: Pressure sensed by Drone's barometer (Pa)
- temp : Temperature sensed by Drone's sensor
- wind_speed: Estimated wind speed
- wind_angle: Estimated wind angle
- wind_comp_angle: Estimated wind angle compensation
- altd: Estimated altitude (In Millimeters)
- motor1..4: Motor PWM values
- vx, vy, vz: Linear velocity (mm/s) (Not published)
- ax, ay, az: Linear acceleration (g)
- tm: Number of micro-seconds passed since Drone's boot-up

Sending Data to the AR.Drone

While the ardrone_driver node is running, the AR.Drone is subscribing to the following topics:

- ardrone/takeoff
- ardrone/land
- ardrone/reset (Publishing an empty message to any of these three topics will result in the corresponding action.)
- cmd_vel This message type is geometry_twist. The corresponding actions from inputs are as follows:
  - -linear.x: pitch backward
  - +linear.x: pitch forward
  - -linear.y: roll right
  - +linear.y: roll left
  - -linear.z: down
  - +linear.z: up
  - -angular.z: yaw left
  - +angular.z: yaw right

AR.Drone Mapping to Generalized Inputs

The generalized quadrotor model commonly used in research assumes the inputs to the system are roll, pitch, yaw, and thrust. However, the AR.Drone takes vertical speed in lieu of thrust. Therefore, an input mapping must take place in order to apply the generalized model. The generalized quadrotor model, state definitions and corresponding frames of reference are described in Chapter 6. The following is the mapping of thrust command to vertical velocity command:

$$R_b^{v2} = \begin{bmatrix} \cos(\phi) & 0 & \sin(\phi) \\ 0 & 1 & 0 \\ -\sin(\phi) & 0 & \cos(\phi) \end{bmatrix} \tag{5.1}$$

$$R_{v2}^{v1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{bmatrix} \tag{5.2}$$

Therefore, the acceleration in the vehicle 1 frame of the quad, a_v, can be determined by:

$$a_v = R_b^{v2} R_{v2}^{v1} \begin{bmatrix} 0 & 0 & T \end{bmatrix}^T \tag{5.3}$$

Where T is the commanded thrust. The desired vertical velocity, $\dot{z}_c$, can then be found given: the current vertical velocity $\dot{z}_{act}$, an estimate of the transport delay (δ), the z component of acceleration $a_v(3)$, the mass of the quad (m), and gravity (g).

$$\dot{z}_c = \dot{z}_{act} + \delta(\frac{a_v(3)}{m} - g) \tag{5.4}$$

Performance Among Competing Networks

Competing WiFi networks around the 2.4 GHz frequency are common in developed locations such as a university campus. The AR.Drone relies on this frequency for communication. Though the communication is rarely compromised, delays upwards of 300 milliseconds have been documented when more than ten networks are competing on the same channel as shown in figure 5.3. This issue only becomes problematic when low latency is in high demand. For example, attempting aggressive maneuvers using off-board controls. This network interference is likely out of the control of the user. Instead of modifying the

AR.Drone in attempts to decrease these latencies, the IRIS+ has been integrated into the RISC MAAP.

5.2 IRIS+

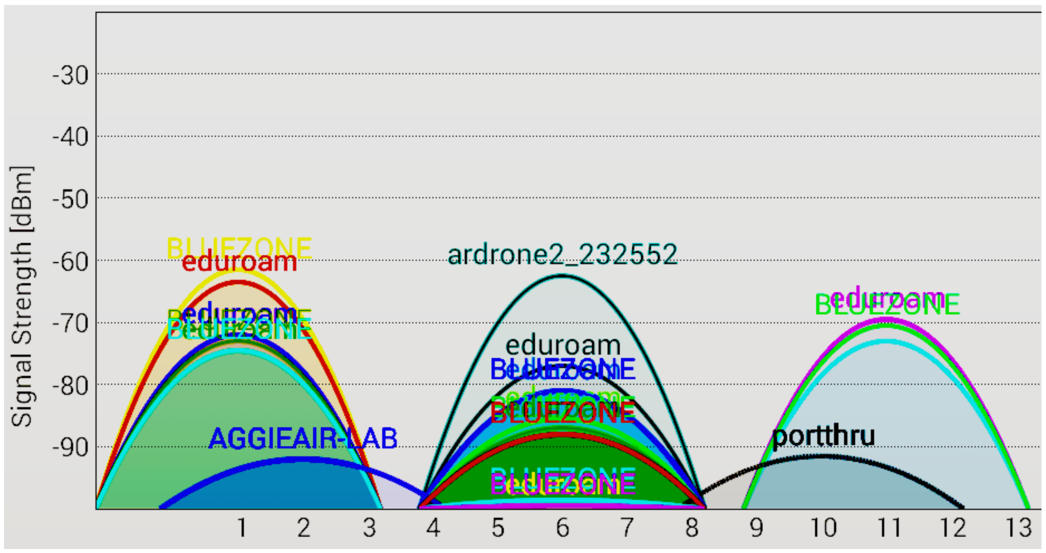


Fig. 5.3: Competing Networks

The IRIS+ shown in figure 5.4 is an electrically powered quadcopter intended for hobbyists and video recording. It can be described by the following specifications:

- Motor to motor dimension: 550 mm
- Height: 100 mm
- Weight (with battery): 1282 g
- Average flight time: 9 - 14 minutes
- 400 g payload capacity
- Battery: 3-cell 11.1 V 3.5 Ah lithium polymer with XT-60 type connector. Weight: 262 g
- Propellers: (2) 10 x 4.7 normal-rotation, (2) 10 x 4.7 reverse-rotation

- Motors: AC 2830, 850 kV
- Telemetry/Control radios available in 915 MHz or 433 MHz
- 32-bit Pixhawk autopilot system with Cortex M4 processor (flight control unit)
- GPS[5] receiver with integrated magnetometer

This model is designed to be very stable and easy to use. It is significantly more powerful when compared to the AR.Drone. It maintains an average flight time of 15 minutes continual flight. The IRIS+ has the advantage of following the conventions of RC[6] flight and uses the general inputs of roll, pitch, yaw rate and thrust. This can be flown using the DX7s 7-Ch DSMX Radio System shown in figure 5.5. The on-board reciever and data transmitter are decoupled. This contributes significantly to decreased latency such that delay is physically imperceptible during manual flight as opposed to the AR.Drone.

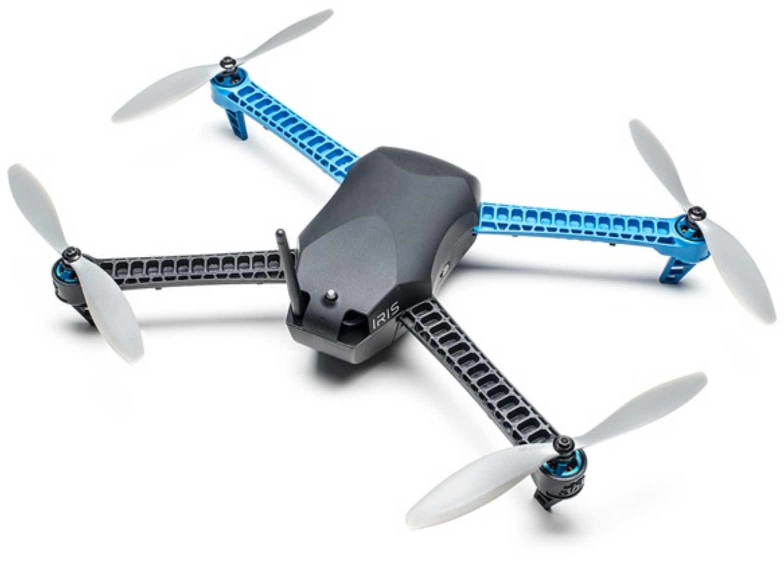


Fig. 5.4: 3DR IRIS+ Quadrotor

5.2.1 ARM Based ROS Integration

[5]Global Positioning System
[6]Radio Controlled

In order to take advantage of the low latency connection using the DX7s a Pulse Position Modulation (PPM) signal is generated and sent to the trainer port of the radio. This was done using the AT91SAM3X8E ARM-Based MCU[7] on the Arduino Due board shown in figure 5.6. ARM-based MCU PPM Signal Generation code is included in Appendix A. A PPM signal modulation is a common method of communication among RC units. This signal encodes values by the amount of time measured between pulses as shown in figure 5.7.

SPEKTRUM
DX7s
SPEKTRUM
01:Acro
4.9V
5:00
7 CHANNELS 20 MODELS 2 TYPES
2.4GHz DSM® SPREAD SPECTRUM TELEMETRY SYSTEM

Fig. 5.5: DX7s 7-Ch DSMX Radio System

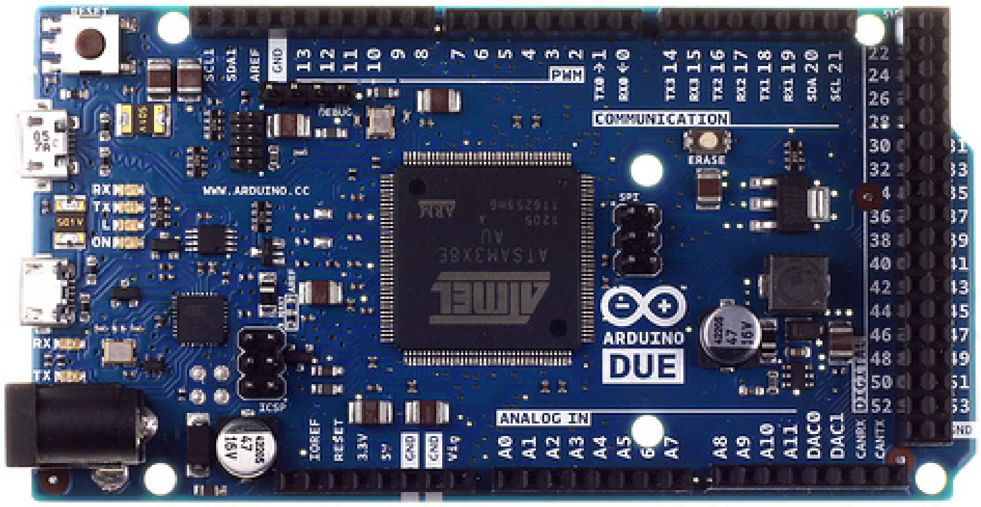


Fig. 5.6: Arduino DUE

Once the signal is generated using the setup described in Appendix A, the signal can be fed to the trainer port as shown in figure 5.8. The simple wiring is shown in figure 5.9.

[7]Microcontroller

This is a desirable solution as it can be expanded to many RC-based communications.

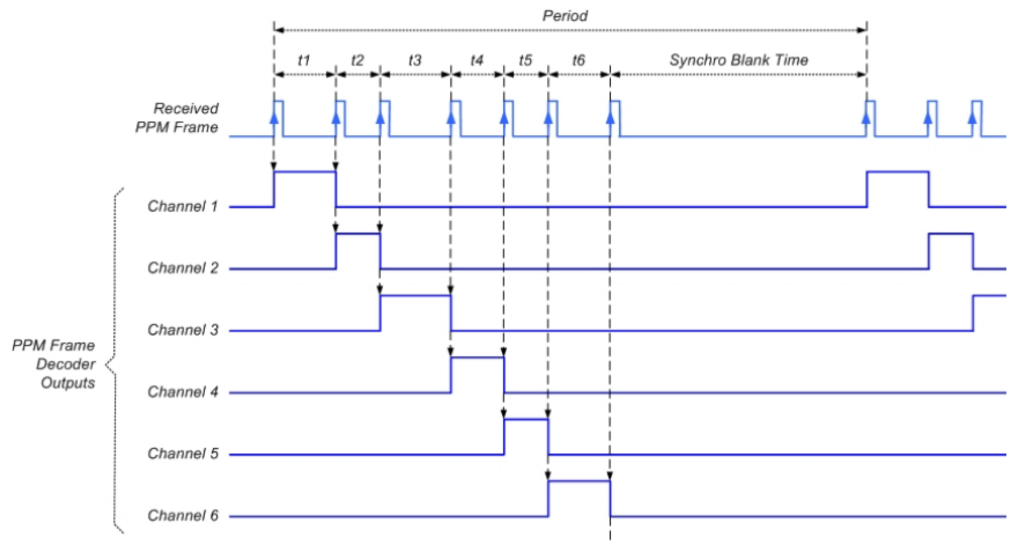


Fig. 5.7: Example PPM Stream Quadrotor

5.2.2 State/Input Mapping

Roll, pitch and yaw rate input to state mappings are configured using PixHawk conventions and can be understood as exact angle and angular rate units. However, the thrust commanded varies with battery level. Deriving a simple PID[8] controller for position hold we were able to gather stable hovering flight data from the IRIS. This data was interpreted using method of least squares to map from battery level to the throttle midpoint as shown in figure 5.10.

The battery percentage B is an integer value 0 to 100. Throttle output is from -1

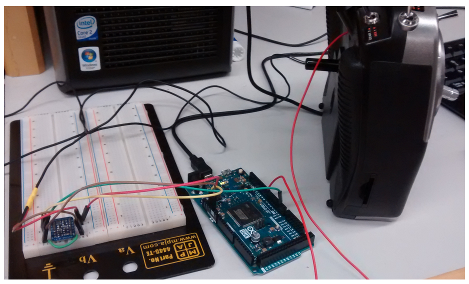


Fig. 5.8: Hardware Setup

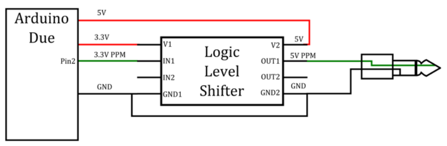


Fig. 5.9: Simplified Diagram

[8]Proportional Integral Derivative

to 1. A fourth order fit was used to match the sigmoidal behavior of the desired throttle. Method of least squares results in the following mapping to desired hover throttle T_{des}:

$$c_0 = 0.491674747062374$$

$$c_1 = -0.024809293286468$$

$$c_2 = 0.000662710609466$$

$$c_3 = -0.000008160593348$$

$$c_4 = 0.000000033699651$$

$$T_{des} = \sum_{i=0}^{4} c_i B^i \tag{5.5}$$

This mapping requires battery feedback and allows more exact mapping of desired to actual thrust expected from the quadrotor.

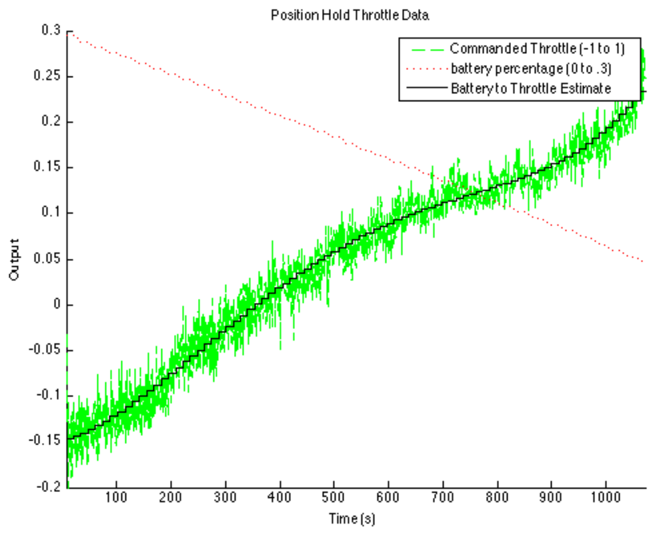


Fig. 5.10: Throttle Mapping

Chapter 6

Educational Use

In this chapter a sample curriculum is provided reproducing a relevant research topic in controls [10]. Differential flatness is outlined along with a demonstrated application. Results of student labs are presented showing demonstration in simulation and hardware.

Given the rapid development and success of the RISC MAAP integration into educational use has already been made possible. The RISC MAAP was used Spring Semester 2015 as a lab portion of the Small Unmanned Aerial Systems course taught at Utah State University. Students gained understanding of sensor limitations as well as the similarities between simulation and hardware. They developed control methods in simulation and demonstrated them on hardware. As solutions and software will likely be used for further courses they will be provided only upon request. An outline of the curriculum follows:

6.1 Curriculum

Students were able to demonstrate a recent paper outlining a novel use of differential flatness control for aggressive trajectory tracking on a quadrotor [10] as well as familiarize themselves with the RISC MAAP.

6.1.1 Differential Flatness Control

Flatness in terms of systems theory can be thought of as an extension of the notion of controllability from linear systems to nonlinear dynamical systems. This general idea can be traced back to works by D. Hilbert and E. Cartan in the early 1900s on underdetermined systems of differential equations, where the number of equations is strictly less than the number of unknowns [31] . A dynamic system which is linearizable via endogenous feedback is said to be (differentially) flat. The terminology flat is due to the fact that the output plays an analogous role to the flat coordinates in the differential geometric approach to the Frobenius theorem [32, 33]. The flatness of a nonlinear system can be characterized by its tangent approximation [34]. If a system is differentially flat the state and input variables can be directly expressed in terms of the (fictitious) flat output and a finite number of its derivatives without integrating any differential equations. This property has been found to be useful in trajectory tracking [10, 31]. This method of control has received increasing attention and been developed as a general solution for various applications [35].

6.1.2 Frames of Reference

Frames of reference for this course have been adopted from [36]. The coordinate frames of interest are defined as follows:

- $\mathcal{F}^i$ = the inertial coordinate frame unit vector $\hat{i^i}$ is directed North, $\hat{j^i}$ is directed East, and $\hat{k^i}$ is toward the center of the earth. Due to the small workspace required for this project, we shifted the origin of this frame from earth centered to a locally determined point for convenience.

- $\mathcal{F}^v$ = the vehicle frame. These axes are aligned with those of $\mathcal{F}^i$. However, the origin is at the center of mass of the quadrotor.

- $\mathcal{F}^{v1}$ = the vehicle 1 frame. $\mathcal{F}^{v1}$ shares the same origin as $\mathcal{F}^v$. However, $\mathcal{F}^{v1}$ is positively rotated about $\hat{k^v}$ by the yaw angle ψ so that if the airframe is not rolling or pitching, then $\hat{i^v}$ would point out the nose.

- $\mathcal{F}^{v2}$ = the vehicle 2 frame. $\mathcal{F}^{v2}$ shares the same origin as both $\mathcal{F}^v$ and $\mathcal{F}^{v1}$. This frame is obtained by rotating $\mathcal{F}^{v1}$ in a right-handed rotation about the $\hat{j^{v1}}$ axis by the pitch angle θ. If the roll angle is zero, then $\hat{i^{v2}}$ points out the nose of the airframe.

- $\mathcal{F}^b$ = the body frame. This is obtained by rotating $\mathcal{F}^{v2}$ in a right handed rotation about $\hat{i^{v2}}$ by the roll angle ϕ. Therefore, the origin is the center-of-gravity, $\hat{i^b}$ points out the nose of the airframe, $\hat{j^b}$ points out the right wing, and $\hat{k^b}$ points out the belly.

6.1.3 Quadrotor General Dynamic Model

All equations of motion for the quadrotor are independent of any trajectory and can therefore be generalized to a variety of cases. The state variables of the quadrotor are the following twelve quantities:

- p_n = displacement along $\hat{i}^i$ in $\mathcal{F}^i$
- p_e = displacement along $\hat{j}^i$ in $\mathcal{F}^i$
- h = displacement along $-\hat{k}^i$ in $\mathcal{F}^i$
- u = the body frame velocity along $\hat{i}^b$ in $\mathcal{F}^b$
- v = the body frame velocity along $\hat{j}^b$ in $\mathcal{F}^b$
- w = the body frame velocity along $\hat{k}^b$ in $\mathcal{F}^b$
- ϕ = the roll angle defined with respect to $\mathcal{F}^{v2}$
- θ = the pitch angle defined with respect to $\mathcal{F}^{v1}$
- ψ = the yaw angle defined with respect to $\mathcal{F}^v$
- p = the roll rate measured along $\hat{i}^b$ in $\mathcal{F}^b$
- q = the pitch rate measured along $\hat{j}^b$ in $\mathcal{F}^b$
- r = the yaw rate measured along $\hat{k}^b$ in $\mathcal{F}^b$

Derivations of the following model for the quadrotor dynamics given these states can be found in [36].

$$\begin{bmatrix} \dot{p}_n \\ \dot{p}_e \\ \dot{h} \end{bmatrix} = \begin{bmatrix} c\theta c\psi & s\phi s\theta c\psi - c\phi s\psi & c\phi s\theta c\psi + s\phi s\psi \\ c\theta s\psi & s\phi s\theta s\psi + c\phi c\psi & c\phi s\theta s\psi - s\phi c\psi \\ s\theta & -s\phi c\theta & -c\phi c\theta \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{6.1}$$

$$\begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} = \begin{bmatrix} rv - qw \\ pw - ru \\ qu - pv \end{bmatrix} + \begin{bmatrix} -gs\theta \\ gc\theta s\phi \\ gc\theta c\phi \end{bmatrix} + \frac{1}{m} \begin{bmatrix} 0 \\ 0 \\ -F \end{bmatrix} \tag{6.2}$$

$$\begin{bmatrix}\dot{\phi}\\ \dot{\theta}\\ \dot{\psi}\end{bmatrix} = \begin{bmatrix}1 & s\phi t\theta & c\phi t\theta\\ 0 & c\phi & -s\phi\\ 0 & \frac{s\phi}{c\theta} & \frac{c\phi}{c\theta}\end{bmatrix}\begin{bmatrix}p\\ q\\ r\end{bmatrix} \tag{6.3}$$

$$\begin{bmatrix}\dot{p}\\ \dot{q}\\ \dot{r}\end{bmatrix} = \begin{bmatrix}\frac{J_y-J_z}{J_x}qr\\ \frac{J_z-J_x}{J_y}qr\\ \frac{J_x-J_y}{J_z}qr\end{bmatrix} + \begin{bmatrix}\frac{1}{J_x}\tau_\phi\\ \frac{1}{J_y}\tau_\theta\\ \frac{1}{J_z}\tau_\psi\end{bmatrix} \tag{6.4}$$

J_x, J_y, and J_z represent the quadrotor's inertial constants, g is the acceleration of gravity (9.80665 $^{m}/_{s^2}$), c:=cos, s:=sin, t:=tan and F:=thrust due to rotors.

6.1.4 ROS

Students familiarize themselves with ROS walking through key concepts:

- Packages
- Node Communication
- Writing Subscribers/Publishers
- Plotting and Data Visualization

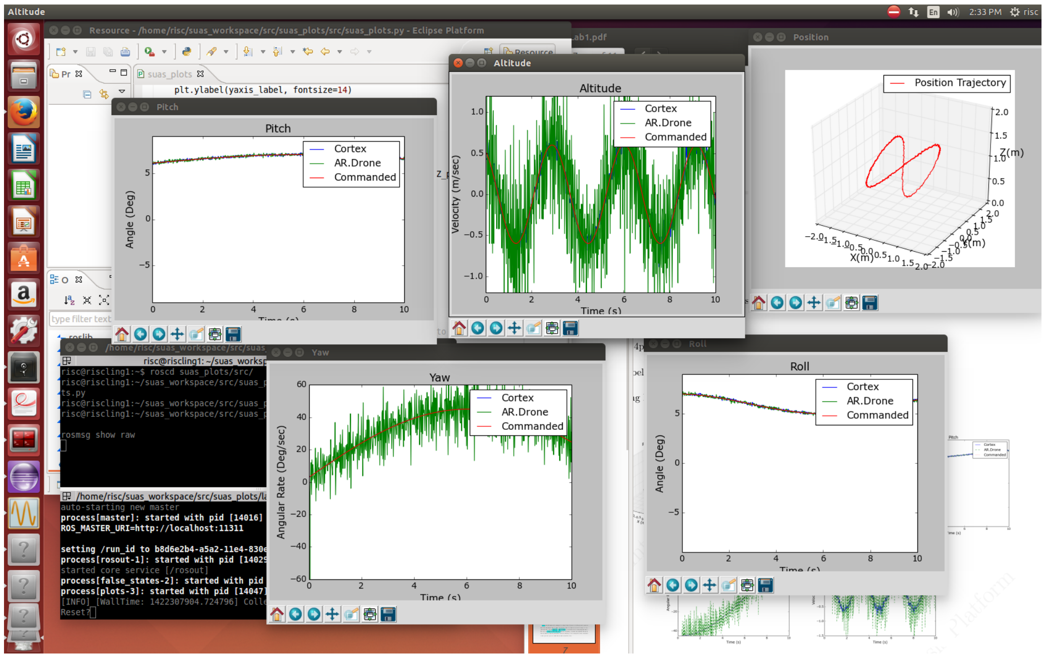
6.1.5 RISC MAAP

Students manually flew the AR.Drone using RISC MAAP conventions via joystick and compared Motion Capture Data to the on board sensor data and commanded versus actual/interpreted output generating plots using python. A screen shot of a students labwork is seen in figure 6.1

6.1.6 Simulation to Hardware

Students used the generic quad model to derive a differential flatness-based controller using the AR.Drone simulation included in the RISC MAAP software for verification. They flew 3 Trajectories shown below using `rviz`.

This was then seamlessly demonstrated on hardware using the same configurations used in simulation. The only modification was using a new launch file to initialize the camera system and the communication with the AR.Drone.



```
plt.ylabel(yaxis_label, fontsize=14)
```

```
risc@riscling1:~$ roscd suas_plots/src/
risc@riscling1:~/suas_workspace/src/suas_p
ts.py
risc@riscling1:~/suas_workspace/src/suas_p
risc@riscling1:~/suas_workspace/src/suas_p

rosmsg show raw
```

```
auto-starting new master
process[master]: started with pid [14016]
ROS_MASTER_URI=http://localhost:11311

setting /run_id to b8d6e2b4-a5a2-11e4-830e
process[rosout-1]: started with pid [14029
started core service [/rosout]
process[false_states-2]: started with pid
process[plots-3]: started with pid [14047]
[INFO] [WallTime: 1422307904.724796] Colle
Reset?
```

Fig. 6.1: Lab Work Screenshot

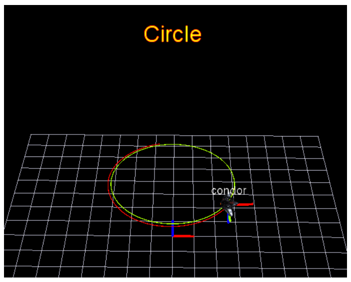


Fig. 6.2: Circle

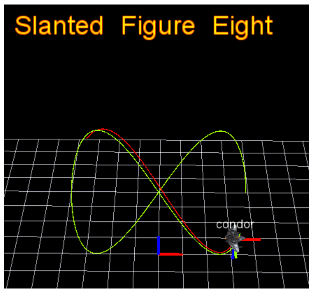


Fig. 6.3: Slanted Figure 8

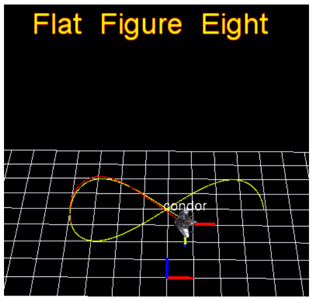


Fig. 6.4: Flat Figure 8

Chapter 7

Research Use

Following the conventions of the RISC MAAP and using the software/hardware supported therein, two novel research papers have been submitted for review [37, 38]. Both have been accepted to the IEEE International Conference on Unmanned Aircraft Systems and the IEEE Signal Processing & SP Education Workshop 2015 respectively. These papers are provided herein. Two more novel research papers will be submitted within the year. One demonstrates bearing-only cooperative localization. The other will generally showcase the RISC MAAP and be submitted to the Journal of Intelligent and Robotic Systems.

Flying Inverted Pendulum Trajectory Control on Robust Intelligent Sensing and Control Multi-Agent Analysis Platform

We demonstrate trajectory control of the tip of an inverted pendulum atop a small unmanned aerial vehicle. In this paper we discuss how exploiting the differential flatness of combined systems provides a realization of adequate control. A differential flatness controller is derived for trajectory control of a pendulum tip. Simulation results are presented for tip trajectory tracking with and without added noise. A framework for hardware demonstration is established. Vertical take-off and landing (VTOL) system capabilities are further explored as well as active/passive manipulation using quadrotors.

7.1 Introduction

The inverted pendulum is well known as one of the classic control problems. It offers a well described and realizable nonlinear unstable system. As such, this problem has been investigated for several decades [39, 40]. Vertical takeoff and landing (VTOL) systems are emerging as a popular platform on which to demonstrate various control techniques such as reinforcement learning [41], neural networks [42], and fuzzy control [43] to name a few. The complex control problem of an inverted pendulum on top of a small unmanned aerial system (SUAS) has been achieved only recently [44]. This was done using an infinite-horizon linear-quadratic regulator (LQR) design [45]. The combination of these systems is relatively new and currently limited by the availability of a visual feedback system providing high accuracy indoor control [46]. Differential flatness based control has been shown to be simple and robust [47, 48, 49]. In this paper we propose and demonstrate a control strategy for this combined system exploiting the property of differential flatness.

Maintaining robust control of an unmanned VTOL system is a vast area of current research. However, extending controllability of a quadrotor to cooperate in tandem with an inherently unstable passive system such as an inverted pendulum has received little attention [44, 50, 51]. In order for VTOL systems to become viable options for various applications such as bridge/industrial inspection, service industry or indoor automation further exploration must occur. This research contributes to that exploration. This high level extended trajectory control given the under powered and actuated system has been, as yet, unrealized [52]. While some comparable work has been done on flying inverted pendulums [44], trajectory control extended to the passive system has not been attempted until now. The main contributions of this paper are as follows:

- A differential flatness controller is derived for trajectory control of a pendulum tip.
- Simulation results are presented for tip trajectory tracking with and without added noise.
- A framework for hardware demonstration is established.

Section 2 will introduce the dynamic models used in the controller design. Section 3 presents the benefits of differential flatness based controller design and the corresponding state mappings. The simulation model and results are shown in Section 4 and conclusions are drawn in Section 5, where an outlook is also presented.

7.2 DYNAMIC MODELS

All equations of motion for the quadrotor and the inverted pendulum are independent of any trajectory and can therefore be generalized to a variety of cases. It is also important to note that in the derivations of these models the effect of the pendulum mass on the quadrotor is neglected. This assumption and relating justifications have been discussed by

Hehn [44]. Frames of reference and the general model of the quadrotor were provided in Chapter 6.

7.2.1 Inverted Pendulum

Definition of variables:

- x_v = displacement along $\hat{i^v}$ in $\mathcal{F}^v$
- y_v = displacement along $\hat{j^v}$ in $\mathcal{F}^v$
- $\dot{x}_v$ = velocity along $\hat{i^v}$ in $\mathcal{F}^v$
- $\dot{y_v}$ = velocity along $\hat{j^v}$ in $\mathcal{F}^v$
- L = length of the pendulum.
- $\zeta = \sqrt{L^2 - x_v^2 - y_v^2}$, displacement along $-\hat{k^v}$ in $\mathcal{F}^v$
- $\ddot{x}$ = quadrotor acceleration along $\hat{i^v}$ in $\mathcal{F}^v$
- $\ddot{y}$ = quadrotor acceleration along $\hat{j^v}$ in $\mathcal{F}^v$
- $\ddot{z}$ = quadrotor acceleration along $-\hat{k^v}$ in $\mathcal{F}^v$

The pendulum's equations of motion were derived using Lagrange's method and are given by:

$$\begin{bmatrix} \ddot{x}_v \\ \\ \ddot{y}_v \end{bmatrix} = \begin{bmatrix} \frac{\alpha(\zeta^2+x_v^2)-\beta x_v y_v}{(\zeta^2+x_v^2)-\frac{x_v^2 y_v^2}{\zeta^2+y_v^2}} \\ \\ \beta - \frac{\ddot{x_v} x_v y_v}{(\zeta^2+y_v^2)} \end{bmatrix} \tag{7.1}$$

$$\alpha = \frac{-\zeta^2}{(\zeta^2+x_v^2)}\left(\ddot{x} + \frac{\dot{x}_v^2 x_v + \dot{y}_v^2 x_v}{\zeta^2} + \frac{\dot{x}_v^2 x_v^3 + 2\dot{x}_v \dot{y}_v x_v^2 y_v + \dot{y}_v^2 y_v^2 x_v}{\zeta^4} - \frac{x_v(\ddot{z}+g)}{\zeta}\right) \tag{7.2}$$

$$\beta = \frac{-\zeta^2}{(\zeta^2+y_v^2)}\left(\ddot{y} + \frac{\dot{y}_v^2 y_v + \dot{x}_v^2 y_v}{\zeta^2} + \frac{\dot{y}_v^2 y_v^3 + 2\dot{y}_v \dot{x}_v y_v^2 x_v + \dot{x}_v^2 x_v^2 y_v}{\zeta^4} - \frac{y_v(\ddot{z}+g)}{\zeta}\right) \tag{7.3}$$

*For a more detailed derivation of the above equations refer to Appendix D.

The control design will require measurement of the states of the pendulum tip for trajectory control. These are functions of previously defined states.

- $x_i = p_n + x_v$, displacement along $\hat{i^i}$ in $\mathcal{F}^i$
- $y_i = p_e + y_v$, displacement along $\hat{j^i}$ in $\mathcal{F}^i$
- $z_i = h - \zeta$ displacement along $\hat{k^i}$ in $\mathcal{F}^i$
- $\dot{x_i} = \dot{p_n} + \dot{x}_v$ velocity along $\hat{i^i}$ in $\mathcal{F}^i$
- $\dot{y_i} = \dot{p_e} + \dot{y}_v$ velocity along $\hat{j^i}$ in $\mathcal{F}^i$
- $\dot{z_i} = \dot{h} - \dot{\zeta}$ velocity along $\hat{k^v}$ in $\mathcal{F}^i$

7.3 CONTROLLER DESIGN

A two-tier hierarchal controller was designed for any arbitrary trajectory using full state feedback and feedforward methodology by exploiting the differential flatness of the system.

7.3.1 Pendulum Tip Trajectory Controller

The first tier in the hierarchal controller seeks to control the trajectory of a simple point mass (our pendulum tip). We use an infinite-horizon linear-quadratic regulator (LQR) design for feedback [45] and provide feedforward from the given trajectory.

Appropriate states for a given trajectory:

$$\bar{x_r} = \begin{bmatrix} x_{ir} & y_{ir} & z_{ir} & \dot{x_{ir}} & \dot{y_{ir}} & \dot{z_{ir}} \end{bmatrix}^T \tag{7.4}$$

Actual states of the pendulum tip:

$$\bar{x} = \begin{bmatrix} x_i & y_i & z_i & \dot{x_i} & \dot{y_i} & \dot{z_i} \end{bmatrix}^T \tag{7.5}$$

State space for computing the LQR gain matrix:

$$\dot{\bar{x}} = A\bar{x} + B\bar{u} \tag{7.6}$$

$$A = \begin{bmatrix} 0_{3\times3} & I_{3\times3} \\ \cdots & 0_{3\times6} \end{bmatrix} \text{ and } B = \begin{bmatrix} 0_{3\times3} \\ I_{3\times3} \end{bmatrix}$$

Desired acceleration given the LQR gain matrix K:

$$\bar{u} = -K(\bar{x_r} - \bar{x}) \tag{7.7}$$

Input for a given trajectory:

$$\bar{u_r} = \begin{bmatrix} \ddot{x_{ir}} & \ddot{y_{ir}} & \ddot{z_{ir}} \end{bmatrix}^T \tag{7.8}$$

$$u_{pm} = \bar{u} + \bar{u_r} \tag{7.9}$$

Total input to the point mass system:

$$u_d = u_{pm} + \begin{bmatrix} 0 & 0 & g \end{bmatrix}^T \tag{7.10}$$

7.3.2 Mapping to Quadrotor/Pendulum System

To realize the desired acceleration, u_{pm}, on the pendulum tip we require a mapping from input variables to system state variables. This is provided given the assumption that the quadrotor can only act on the pendulum tip in the direction along the pendulum shaft as seen in figure 7.1.

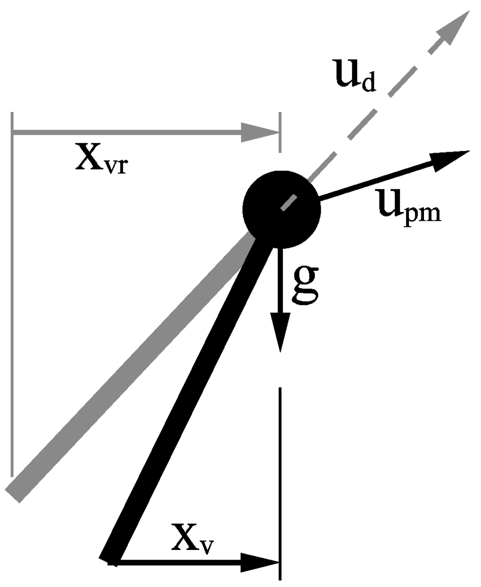


Fig. 7.1: Acceleration Mapping

Thus, the required system states can be found as follows:

$$x_{vr} = L\frac{u_d(1)}{\|u_d\|} \tag{7.11}$$

$$y_{vr} = L\frac{u_d(2)}{\|u_d\|} \tag{7.12}$$

$$p_{nr} = x_i - x_{vr} \tag{7.13}$$

$$p_{er} = y_i - y_{vr} \tag{7.14}$$

$$h_r = z_i - \sqrt{L^2 - x_{vr}^2 - y_{vr}^2} \tag{7.15}$$

$$\psi_r = 0 \tag{7.16}$$

The following have been set to zero since an appropriate mapping would require estimating a desired jerk applied to the pendulum tip.

$$\dot{x_{vr}} = \dot{y_{vr}} = u_r = v_r = w_r = 0 \tag{7.17}$$

7.3.3 Quadrotor/Pendulum System Controller

The second tier in our hierarchal controller seeks to control the state of combined system of the inverted pendulum and quadrotor. We use the same methods as those described in the first tier. However, there is no reference acceleration.

Appropriate states given the previous mapping:

$$x_r = \begin{bmatrix} x_{vr} & y_{vr} & \dot{x_{vr}} & \dot{y_{vr}} & p_{nr} & p_{er} & h_r & u_r & v_r & w_r & \psi_r \end{bmatrix}^T$$

Actual states of the system:

$$x = \begin{bmatrix} x_v & y_v & \dot{x_v} & \dot{y_v} & p_n & p_e & h & u & v & w & \psi \end{bmatrix}^T$$

State space for computing the LQR gain matrix:

$$\dot{x} = Ax + Bu \tag{7.18}$$

$$A = \begin{bmatrix} 0_{2\times2} & I_{2\times2} & 0_{2\times7} \\ \cdots & 0_{2\times11} & \cdots \\ 0_{3\times7} & I_{3\times3} & 0_{3\times1} \\ \cdots & 0_{4\times11} & \cdots \end{bmatrix} , \quad B = \begin{bmatrix} \cdots & 0_{2\times6} \\ I_{2\times2} & 0_{2\times4} \\ \cdots & 0_{3\times6} \\ 0_{4\times2} & I_{4\times4} \end{bmatrix}$$

Desired acceleration given the LQR gain matrix K:

$$u = -K(x_r - x) \tag{7.19}$$

7.3.4 Mapping to Quadrotor Inputs

Assuming the pendulum states to be small values, the system is linearized about a balanced pendulum. This results in:

$$\begin{bmatrix} \ddot{x}_v \\ \ddot{y}_v \end{bmatrix} = \left. \begin{bmatrix} \frac{\alpha(\zeta^2+x_v^2)-\beta x_v y_v}{(\zeta^2+x_v^2)-\frac{x_v^2 y_v^2}{\zeta^2+y_v^2}} \\ \beta - \frac{\ddot{x_v} x_v y_v}{(\zeta^2+y_v^2)} \end{bmatrix} \right|_{x_{eq}} \tag{7.20}$$

Thus,

$$\begin{bmatrix} \ddot{x}_v \\ \ddot{y}_v \end{bmatrix} \approx \begin{bmatrix} -\ddot{x} \\ -\ddot{y} \end{bmatrix} \tag{7.21}$$

Therefore u can be expressed as:

$$\begin{bmatrix} u_{p1} \\ u_{p2} \\ u_{p3} \\ u_{p4} \end{bmatrix} = \begin{bmatrix} u_3 - u_1 \\ u_4 - u_2 \\ u_5 \\ u_6 \end{bmatrix} \tag{7.22}$$

For the general model of the quad we define the vector of inputs to be:

$$\nu = \begin{bmatrix} T_d & \Phi_d & \theta_d & r_d \end{bmatrix}^T \tag{7.23}$$

The on-board attitude controller closes the loop on the desired input commands which eliminates the need to provide any further control on the system. The mapping from u to ν has been adopted from Ferrin's text [10]. Let m_q be defined as the mass of the quad.

$$T_d = m_q \left\| \begin{bmatrix} u_{p1} & u_{p2} & u_{p3} \end{bmatrix}^T \right\| \tag{7.24}$$

$$z = -R(\psi) \begin{bmatrix} u_{p1} & u_{p2} & u_{p3} \end{bmatrix}^T \frac{m_q}{T_d} \tag{7.25}$$

$$\Phi_d = sin^{-1}(-z_2) \tag{7.26}$$

$$\theta_d = tan^{-1}(\frac{z_1}{z_3}) \tag{7.27}$$

$$r_d = u_{p4} \tag{7.28}$$

The limits and benefits of this approach are described in [10].

7.4 Results

The algorithms presented were implemented initially using Simulink and then transferred to the Robust Intelligent Sensing and Control Multi-Agent Analysis Platform (RISC MAAP) using python as described in [53]. We present results demonstrating the performance of the controller designed in the previous section. Two vehicles were evaluated for demonstration. The Parrot AR.Drone (suggested by Krajnik for use in education [30]) and the 3DR IRIS (a Pixhawk based system from 3D Robotics). These unmanned aerial systems are significantly different. Thus, our simulation was designed for a generic case using the RISC MAAP as shown in figure 4.1. The RISC MAAP is equipped with an infrared motion tracking system that provides accurate vehicle position and attitude measurements at 200 Hz. The communication systems for both vehicles are wireless and can be assumed to incorporate some delay. The inverted pendulum is composed of a carbon fiber tube approximately 1 meter in length. The top end of the pendulum is mounted with similar markers as the quadrotor, allowing the position of this point to be determined through the motion tracking system.

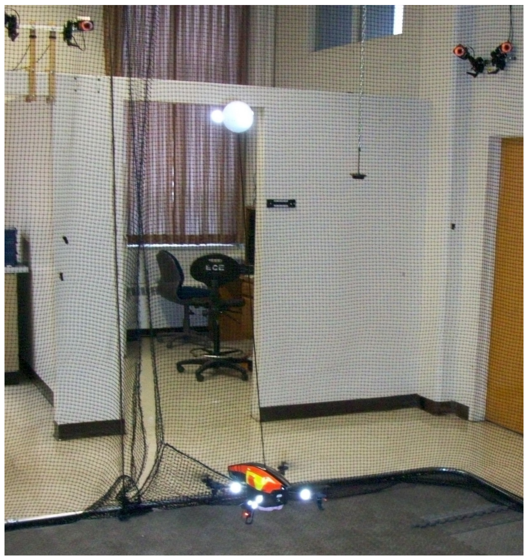


Fig. 7.2: AR.Drone with Pendulum

7.4.1 Simulation

A flat figure eight trajectory was chosen since it is a simple path that is mathematically realizable. The system's initial conditions were set outside the given trajectory to demonstrate stability. Keeping in mind our assumed data rates and delays, we see a tightly held trajectory in figure 7.3. Desired and actual trajectories for each axis can be seen in figure 7.4, while overall positional errors are displayed in figure 7.5. These results include a 2mm S.D. white noise on the position of the pendulum and the quadrotor. This is to account for the noise in the RISC MAAP system.

As can be seen in the above figures, the actual states converge to the desired trajectory within a short time frame (∼10 seconds). While there is some oscillation about the trajectory, the pendulum tip remains stabilized within acceptable bounds. This was achieved with minimal tuning. However, for demonstration on hardware more aggressive tuning will likely be necessary.

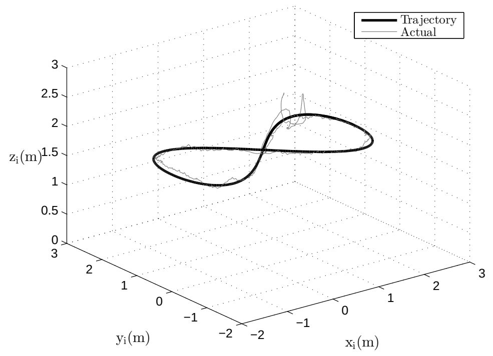


Fig. 7.3: Figure Eight Trajectory Pendulum Tip Position

7.5 CONCLUSION

We have demonstrated trajectory control in simulation of the tip of an inverted pendulum atop a small unmanned aerial vehicle. State mapping permitted through exploitation of the differential flatness of the combined systems has proven a useful tool for extending control to a passive system. The simulation displayed significant robustness in the presence of noise well within the bounds expected using the RISC MAAP (2 mm S.D.). Given the data provided in this paper, this experiment is expected to be realizable on the current RISC MAAP system.

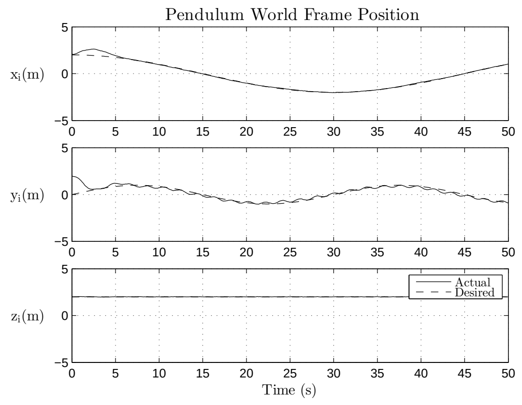


Fig. 7.4: Figure Eight Trajectory and Position

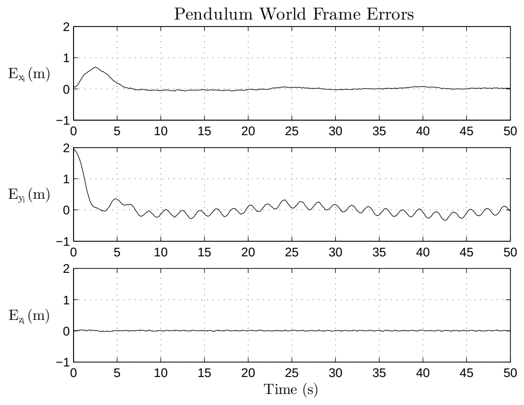


Fig. 7.5: Figure Eight Trajectory Errors

This research explores and exposes as yet unrealized capabilities and possibilities of VTOL platforms. We suggest there is still much research to be done in order to understand the possible extensions to the sensing/acting capacities of these unmanned aerial systems.

Using Extended Kalman Filter for Robust Control of a Flying Inverted Pendulum

We propose the use of an Extended Kalman Filter (EKF) for reliable state estimation in order to permit advanced control of the tip of a flying inverted pendulum while maintaining safety. We demonstrate the capabilities of an EKF in tandem with an accurate model to overcome bad or false data from a multiple camera motion capture system used for positioning.

7.6 Introduction

Many aspects of robotics are influenced by unpredictable dynamic environments in ways that make reliable estimation challenging. Consequently, robotics researchers have turned towards controlled testing platforms [2] [3] to validate theory. Perhaps among the most famous of these is The Flying Machine Arena at ETH Zurich, which contains a net-enclosed workspace of one thousand cubic meters [5]. These platforms aim for a controlled environment that provides reliable ground truth estimates of robotic systems. This is usually achieved using high frequency, low latency feedback from cameras interpreted to provide accurate position measurements. These systems allow in-depth analysis and performance that would otherwise be unattainable today in natural environments.

While these camera systems can provide useful data, under suboptimal conditions they may also supply false data. These undesirable conditions result in two significant difficulties. First, false data is indistinguishable from desired data due to lack of a model-driven estimator. Second, overall quality of state estimation degrades and system observability over time is lacking. For aggressive applications on powerful highly mobile robotics this situation can be dangerous or even life-threatening.

We propose implementation of an Extended Kalman Filter (EKF) for difficult aggressive tasks of a well modeled system to overcome these pitfalls. Specifically, we demonstrate the capabilities of advanced filtering to solve the problem of controlling a flying inverted pendulum tip to follow a desired trajectory using a quadrotor. This control strategy was previously developed in [37] and exploits differential flatness using a hierarchal infinite horizon linear quadratic regulator.

7.7 SENSOR CHALLENGES

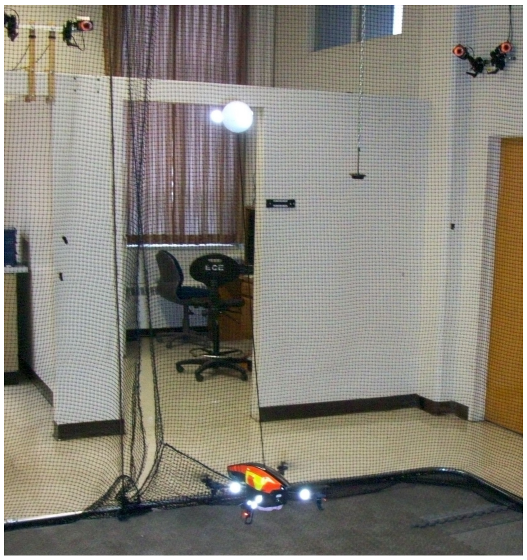


Fig. 7.6: Flying Inverted Pendulum

The Robust Intelligent Sensing and Control (RISC) Lab uses the Multi-Agent Analysis Platform (MAAP) [37] camera setup illustrated in figure 4.1. This system uses MotionAnalysis infrared cameras (shown in figure 2.1) and reflective markers to provide position estimates of objects. An array of markers in a fixed geometry provides the object template. This template can be used to provide pose estimation. From experience using this system we have developed the following list of potential drawbacks.

- Poor marker template geometry can decrease accuracy
- Infrared spectrum interference due to natural lighting or unwanted reflective materials will result in false data

- Vibrations affecting the camera pose require constant recalibration requirements
- Unfocused or suboptimal camera setup creates uneven volume capture
- Camera view obstruction results in data loss

This list is not exhaustive. However, it is intended to illustrate the limitations of this sensing environment. Even the high quality systems require constant calibration and maintenance to compensate for these issues. False information becomes unavoidable in lower quality or suboptimal operating locations.

Data similar to that depicted in figure 7.7 is not improbable. The dark bars on the plot signify detectable data loss. Data loss detection without an EKF is currently done by eliminating data points that lie outside the physically realizable space. In the absence of data the controller drives the quadrotor to produce zero acceleration since an alternative estimate does not exists. However, this has been insufficient to deal with sensing deficiencies and will be discussed further in Section 5.

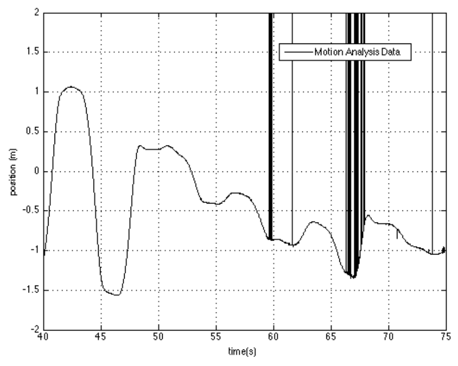


Fig. 7.7: Actual Data From RISC MAAP

7.8 EXTENDED KALMAN FILTER

The EKF is an extension of the well-known Kalman Filter as described in [19]. An EKF provides an approximate linearized estimate of the states despite the nonlinear characteristics of the system [20]. A generic EKF can be summarized as follows:

7.8.1 Propagate:

- $\hat{x}_{i+1} = f(\hat{x}_i, u_i)$

  $\hat{x}_i$ = state estimate at iteration i

$f(\hat{x}_i, u_i) =$ Nonlinear model

- $\hat{P}_{i+1} = \Phi_i P_i \Phi_i^T + Q_{d,i}$

  $\Phi_i = I + \frac{\partial f(\hat{x}_i, u_i)}{\partial x}|_{\hat{x}_i, u_i} \Delta t$ (Approximate Jacobian)

  $Q_{d,i} =$ Process noise covariance

  $P_i =$ Covariance of estimate

7.8.2 Update:

- Measurement: z_i
- Residual: $y_i = z_i - \hat{x}_i$
- Residual Covariance: $\mathfrak{R}_i = H_i P_i H_i^T + R$ where R is the measurement noise covariance and H_i is the observation model matrix.

- Kalman Gain: $K_i = P_i^- H_i^T \mathfrak{R}_i^{-1}$
- Update State: $\hat{x}_i^+ = \hat{x}_i^- + K_i y_i$
- Update Covariance: $P_i^+ = (I - K_i H_i) P_i^-$

Thus, $\hat{x}_i^+$ is our new estimated mean and P_i^+ is the corresponding covariance.

Our implementation generally followed this form. One exception was the necessity of using the estimated acceleration of the quad as an input to the pendulum base model. Data loss detection was performed using a chi-square test given a state mean x and covariance$P$ with measurement noise covariance R [21]. Overall, the flying inverted pendulum relies on valid estimates of the 16 states as defined in Section 2. Due to limited space in this paper, Jacobians and derivations have been added in Appendix E and F while simulation videos are available on our website [53].

The resulting EKF performance is depicted in figure 7.8 and 7.9. The estimate closely follows the true data in the presence of noise and accurately displays the degradation of information while data is lacking.

7.9 RESULTS

The EKF developed in this paper is intended to enable the advanced control task of flying inverted pendulum tip to follow a desired trajectory as described in [37] in a suboptimal motion capture environment. These simulations focus on a following a figure eight trajectory as depicted in figure 7.10.

.

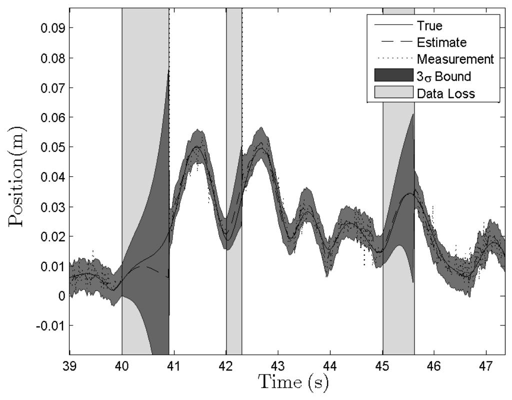


Fig. 7.8: Pendulum Position Estimate

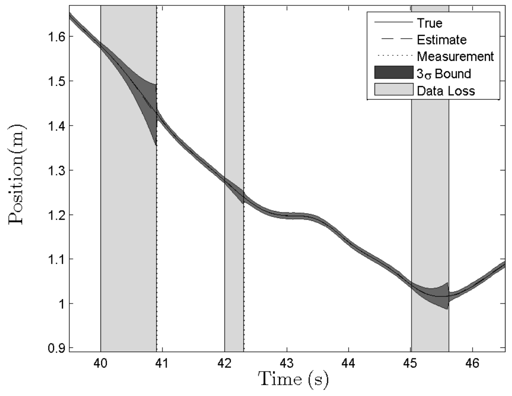


Fig. 7.9: Quadrotor Position Estimate

Results are evaluated using the ideal controller with complete and accurate data defining the baseline. Therefore, effectiveness of the estimate is evaluated with respect to the root mean squared error of the pendulum tip and the desired position in figure 7.11 and velocity in figure 7.12. The ideal controller performance is shown as complete data. The noisy data relies on a simple law setting the desired quad acceleration to zero if data is recognized as false or absent.

As can be seen below, without the EKF the pendulum is unable to maintain the trajectory. While, the controller using the EKF estimates better tracks the ideal.

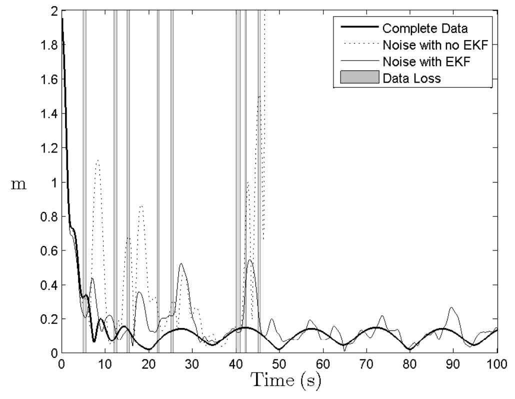


Fig. 7.11: Pendulum Position Errors

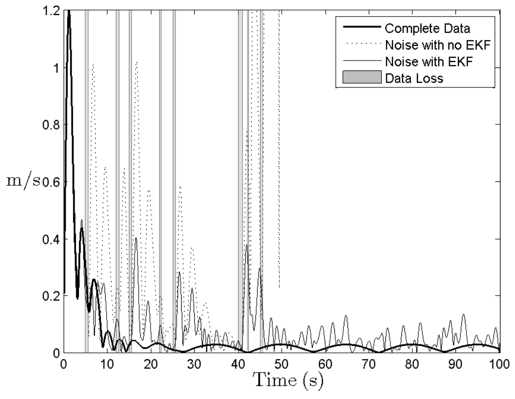


Fig. 7.12: Pendulum Velocity Errors

7.10 CONCLUSION

In this paper we have argued that motion capture camera systems can provide useful data. However, these systems may require significant filtering such as an EKF for reliable state estimation in advanced applications such as flying inverted pendulum control. By employing this model driven filter we have overcome two significant difficulties. First, the unreliable or bad-data can be distinguished from desired data due to a model-driven estimator.

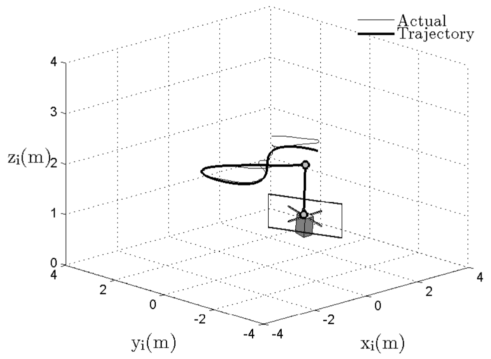


Fig. 7.10: 3D Trajectory

Second, the estimator provides a measurement of data quality allowing the user to gain a better understanding of system observability over time. This enables the user to achieve greater safety and control in potentially harmful and aggressive maneuvers. The benefits of this method were demonstrated in a realistic example.

This paper has considered only specific aspects of misinformation from motion capture systems. Future work will deal with hardware implementation of this specific problem and likely exploration into more advanced applications.

Cooperative Bearing-only Localization Experimentation

Software and hardware setup for multi agent systems are explained. Vision-based landmark detection is discussed.

7.11 EXPERIMENTAL MOTIVATION

Cooperative Localization is a vast area of current research and experimental validation of theoretical results is difficult due to the involvement and coordination of many systems and the necessity of reliable "ground-truth" for validation [54] [55] [56]. The focus of this experimental setup will be to provide validation of bearing-only cooperative localization as an extension of Sharma's work [57] to a 3D environment using quadrotors. The following aspects of this experiment will be discussed: Hardware, Communication, Software and Data Collection.

7.12 DATA COLLECTION

Much of the data analysis was implemented in Matlab. This required a conversion from `rosbag` to csv[1] file. This was done using a significantly modified version of Shane Lynn's data_extraction package. The data gathered in experimentation were the following:

- 3 HD cameras streamed over a wireless network at 30 hz
- Navdata provided by AR.Drone SDK
- Observed landmark bearing angles
- Control inputs to the AR.Drones
- Camera to body frame transforms
- Regions of interest of detected landmarks
- Desired trajectories

[1]Comma Separated Value

- Ground truth estimates of quadrotor states
- Ground truth landmark positions

These data in form of bag files are quite large. 10 minutes of data corresponds to 8 GB of data. The majority of this data was visualized in real-time using `rviz` tools. An experimental setup is shown in figure 7.13 and the corresponding `rviz` representation of the same experiment can be seen in figure 7.14.

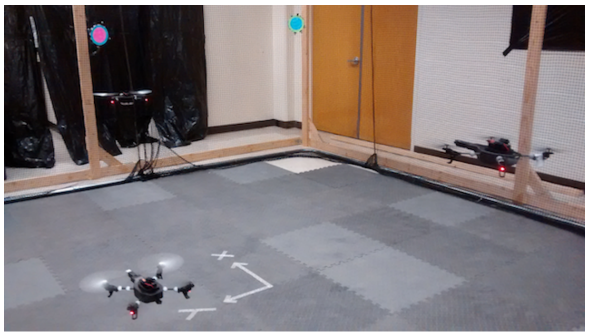


Fig. 7.13: Multi-agent Analysis in RISC Lab

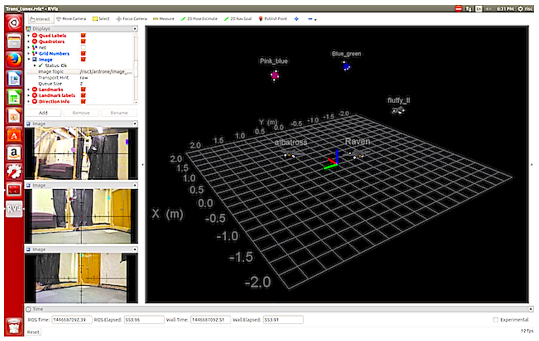


Fig. 7.14: Multi-agent Analysis in `rviz`

There are many benefits of using bagged data. One of these is the ease of playing back any portion of interest in real-time. This enables verification and analysis of research algorithms that may be too computationally intensive to run during a live experiment.

Conclusions regarding these data have not yet been drawn. Further analysis and discussion of the significance of this experimentation data is the topic of current research in the RISC Lab. However, the ability to readily perform such complex experiments repeatedly illustrates the utility of the RISC MAAP.

Chapter 8
Conclusion

As stated in the introduction, three goals of this research were:

- Develop an open source adaptable expandable test bed platform for use in education and research.
- Demonstrate a complex control problem for use in education using the developed platform.
- Extend modern controls research using the developed platform

All of these goals have been realized and will be discussed in turn, followed by an overview of future work.

An open source adaptable, expandable test bed platform has been developed for use in education and research. In Chapter 2 details related to the motion capture camera system were provided while other relevant hardware used within the RISC MAAP was detailed in Chapter 5. Software architecture was outlined in Chapter 3 along with the benefits and limitations of using the Robotics Operating System (ROS) as a foundation. Goals regarding efficiency and usability of the overall system and how these are addressed was provided in Chapter 4.

The Robust Intelligent Sensing and Control Multi Agent Analysis Platform has been shown to be adaptable and expandable. This encourages reuse of code and solutions for future work ensuring that redundant work is avoided. This allows researchers to focus on specific research/academic problems rather than on implementing the many necessary, but unrelated parts of the system.

Given the rapid development of this platform, it has already been used extensively for education. Chapter 6 provided sample curriculum for reproducing a relevant research topic in controls [10]. Results of lab work were presented showing the student's demonstration in simulation as well as hardware. This demonstrated a complex control problem for use in education using the developed platform. Aside from being indispensable to research and education in the RISC Lab, the flexibility of the platform has encouraged use for projects and research by the electrical, civil and mechanical engineering departments.

Chapter 7 provided examples of how modern controls and localization research has been extended using this developed platform. Trajectory control in simulation of the tip of an inverted pendulum atop a small unmanned aerial vehicle was demonstrated. State mapping permitted through exploitation of the differential flatness of the combined systems has proven a useful tool for extending control to a passive system. The simulation displayed significant robustness in the presence of noise well within the bounds expected using the RISC MAAP. Exploration of motion capture camera systems has provided useful insight in the role of model driven filtering while performing aggressive maneuvers. Also, multi-agent experimentation has been shown to be achievable in a manageable repeatable framework.

Future work will likely include the support of ground or even aquatic vehicles. This system is especially equipped to facilitate exploratory research in areas of GPS-denied or indoor navigational systems as well as aggressive controls. Structural analysis, ground effect characterization, and system identification are also current topics of future work discussed. In summary, future work related to the RISC MAAP will likely expand far beyond the topics discussed herein. However, this research explores and exposes the possibilities of sharing work across a unified testbed. The hope is that this work can be used not only at Utah State University but throughout the world to accelerate robotics research and encourage collaboration.

References

[1] H. S. Witsenhausen, “Separation of estimation and control for discrete time systems,” *Proceedings of the IEEE*, vol. 59, pp. 1557– 1566, 1971. [Online]. Available: http://ieeexplore.ieee.org/ielx5/5/31150/01450418.pdf?tp=&arnumber=1450418&isnumber=31150

[2] H. J. Kim and D. H. Shim, “A flight control system for aerial robots: Algorithms and experiments,” *Control Engineering Practice*, vol. 11, pp. 1389–1400, 2003.

[3] M. Valenti, B. Bethke, D. Dale, A. Frank, J. McGrew, S. Ahrens, J. P. How, and J. Vian, “The MIT indoor multi-vehicle flight testbed,” in *Proceedings - IEEE International Conference on Robotics and Automation*, 2007, pp. 2758–2759.

[4] A. Richards and J. Bellingham, “Coordination and control of multiple UAVs,” *. . . , Navigation, and Control . . .* , pp. 1–11, 2002. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.208.1820&rep=rep1&type=pdf

[5] S. Lupashin, A. Schöllig, M. Sherback, and R. D’Andrea, “A simple learning strategy for high-speed quadrocopter multi-flips,” in *Proceedings - IEEE International Conference on Robotics and Automation*, 2010, pp. 1642–1648.

[6] S. Hanks, M. E. Pollack, and P. R. Cohen, “Benchmarks, Test Beds, Controlled Experimentation, and the Design of Agent Architectures,” *AAAI-1993*, vol. 14, pp. 17–42, 1993.

[7] J. L. Blanco, F. A. Moreno, and J. Gonzalez, “A collection of outdoor robotic datasets with centimeter-accuracy ground truth,” *Autonomous Robots*, vol. 27, pp. 327–351, 2009.

[8] M. Amoretti and M. Reggiani, “Architectural paradigms for robotics applications,” *Advanced Engineering Informatics*, vol. 24, pp. 4–13, 2010.

[9] S. Cousins, B. Gerkey, K. Conley, and W. Garage, "Sharing software with ROS," *IEEE Robotics and Automation Magazine*, vol. 17, pp. 12–14, 2010.

[10] J. Ferrin, R. Leishman, R. Beard, and T. McLain, "Differential flatness based control of a rotorcraft for aggressive maneuvers," *IEEE International Conference on Intelligent Robots and Systems*, pp. 2688–2693, 2011.

[11] WillowGarage, "Robot Operating System(ROS)," 2014. [Online]. Available: http://www.ros.org/testimonials/

[12] M. Quigley, E. Berger, A. Y. Ng *et al.*, "Stair: Hardware and software architecture," in *AAAI 2007 Robotics Workshop, Vancouver, BC*, 2007, pp. 31–37.

[13] K. A. Wyrobek, E. H. Berger, H. F. M. Van Der Loos, and J. K. Salisbury, "Towards a personal robotics development platform: Rationale and design of an intrinsically safe personal robot," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2008, pp. 2165–2170.

[14] A. Jiménez-González, J. R. Martínez-de Dios, and A. Ollero, "An integrated testbed for cooperative perception with heterogeneous mobile and static sensors," *Sensors*, vol. 11, pp. 11 516–11 543, 2011.

[15] M. Quigley, K. Conley, B. Gerkey, J. FAust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Mg, "ROS: an open-source Robot Operating System," *ICRA*, vol. 3, p. 5, 2009.

[16] WillowGarage, "Robot Operating System(ROS)," 2012.

[17] M. Fleder, "ROS : Robot Operating System," in *RSS*, 2012, pp. 1–15. [Online]. Available: http://courses.csail.mit.edu/6.141/spring2012/pub/lectures/Lec06-ROS.pdf

[18] P. Green, "Using Python for Interactive Data Analysis," *Reading*, vol. 100, p. 144, 2007. [Online]. Available: http://stsdas.stsci.edu/perry/pydatatut.pdf

[19] R. E. Kalman, “A New Approach to Linear Filtering and Prediction Problems,” *Transactions of the ASME-Journal of Basic Engineering*, vol. 82, pp. 35–45, 1960. [Online]. Available: http://fluidsengineering.asmedigitalcollection.asme.org/article.aspx?articleid=1430402

[20] G. Einicke and L. White, “Robust extended Kalman filtering,” *IEEE Transactions on Signal Processing*, vol. 47, 1999.

[21] B. Brumback and M. Srinath, “A Chi-square test for fault-detection in Kalman filters,” *IEEE Transactions on Automatic Control*, vol. 32, pp. 552–554, 1987.

[22] S. Weiss, M. W. Achtelik, S. Lynen, M. Chli, and R. Siegwart, “Real-time onboard visual-inertial state estimation and self-calibration of MAVs in unknown environments,” in *Proceedings - IEEE International Conference on Robotics and Automation*, 2012, pp. 957–964.

[23] J. Kelly and G. S. Sukhatme, “Fast Relative Pose Calibration for Visual and Inertial Sensors,” in *Springer Tracts in Advanced Robotics*, vol. 54, 2009, pp. 515–524.

[24] J. Lobo and J. Dias, “Relative Pose Calibration Between Visual and Inertial Sensors,” pp. 561–575, 2007.

[25] A. Yilmaz, O. Javed, and M. Shah, “Object tracking: A survey,” *ACM Computing Surveys*, vol. 38, p. 13, 2006. [Online]. Available: http://dx.doi.org/10.1145/1177352.1177355

[26] S. Suzuki and K. Abe, “Topological structural analysis of digitized binary images by border following,” p. 396, 1985.

[27] M. K. Hu, “Visual Pattern Recognition by Moment Invariants,” *IRE Transactions on Information Theory*, vol. 8, pp. 179–187, 1962.

[28] G. Gousios, B. Vasilescu, A. Serebrenik, and A. Zaidman, “Lean GHTorrent: GitHub data on demand,” *Proceedings of the 11th Working Conference on*

Mining Software Repositories - MSR 2014, pp. 384–387, 2014. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2597073.2597126

[29] "Robust intelligent sensing and control multi agent analysis platform for education and research," https://github.com/riscmaster/risc_maap, 2015.

[30] T. Krajník, V. Vonásek, D. Fišer, and J. Faigl, "AR-drone as a platform for robotic research and education," in *Communications in Computer and Information Science*, vol. 161 CCIS, 2011, pp. 172–186.

[31] M. Fliess, J. Lévine, P. Martin, and P. Rouchon, "Flatness and defect of non-linear systems: Introductory theory and examples," pp. 1327–1361, 1995.

[32] S. S. Sastry and A. Isidori, "Adaptive control of linearizable systems," *Automatic Control, IEEE Transactions on*, vol. 34, no. 11, pp. 1123–1131, 1989.

[33] H. Nijmeijer, "van der schaft," *Nonlinear dynamical control systems*, 1990.

[34] M. Franke and K. Röbenack, "Some remarks concerning differential flatness and tangent systems," *PAMM*, vol. 12, no. 1, pp. 729–730, 2012.

[35] W. V. Loock, G. Pipeleers, M. Diehl, J. D. Schutter, and J. Swevers, "Optimal Path Following for Differentially Flat Robotic Systems Through a Geometric Problem Formulation," *IEEE Transactions on Robotics*, vol. 30, pp. 980–985, 2014.

[36] R. W. Beard and T. W. McLain, *Small unmanned aircraft: Theory and practice.* Princeton University Press, 2012.

[37] D. S. Maughan, I. Erekson, and R. Sharma, "Flying Inverted Pendulum Trajectory Control on Robust Intelligent Sensing and Control Multi-Agent Analysis Platform," in *Proceedings - IEEE International Conference on Unmanned Aircraft Systems*, 2015.

[38] ——, "Using Extended Kalman Filter for Robust Control of a Flying Inverted Pendulum," in *Proceedings - IEEE Signal Processing and SP Education Workshop 2015*, 2015.

[39] S. Mori, H. Nishihara, and K. Furuta, "Control of unstable mechanical system Control of pendulum," pp. 673–692, 1976.

[40] J. Shen, A. Sanyal, N. Chaturvedi, D. Bernstein, and H. McClamroch, "Dynamics and control of a 3D pendulum," *Proceedings of the IEEE Conference on Decision and Control*, vol. 1, pp. 323–328, 2004. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1428650

[41] M. Lagoudakis, "Least-squares policy iteration," *The Journal of Machine Learning Research*, vol. 4, pp. 1107–1149, 2003. [Online]. Available: http://dl.acm.org/citation.cfm?id=964290

[42] V. Williams and K. Matsuoka, "Learning to balance the inverted pendulum using neural networks," *[Proceedings] 1991 IEEE International Joint Conference on Neural Networks*, 1991.

[43] H. O. Wang, K. Tanaka, and M. F. Griffin, "An approach to fuzzy control of nonlinear systems: Stability and design issues," *Ieee Transactions on Fuzzy Systems*, vol. 4, pp. 14–23, 1996. [Online]. Available: ⟨GotoISI⟩://WOS:A1996TV80400002

[44] M. Hehn and R. D'Andrea, "A flying inverted pendulum," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2011, pp. 763–770.

[45] D. P. Bertsekas, *Dynamic Programming and Optimal Control, Vol. II*, 2007, vol. 2. [Online]. Available: http://portal.acm.org/citation.cfm?id=1396348

[46] S. Grzonka, G. Grisetti, and W. Burgard, "A fully autonomous indoor quadrotor," *IEEE Transactions on Robotics*, vol. 28, pp. 90–100, 2012.

[47] M. B. Colton, L. Sun, D. C. Carlson, and R. W. Beard, "Multi-vehicle dynamics and control for aerial recovery of micro air vehicles," *International Journal of Vehicle Autonomous Systems*, vol. 9, no. 1/2, p. 78, 2011.

[48] R. M. Murray, M. Rathinam, and W. Sluis, "Differential flatness of mechanical control systems: A catalog of prototype systems," *Proc. of the ASME Mechanical*

Engineering Congress and Expo, pp. 1–9, 1995. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.55.7225&rep=rep1&type=pdf

[49] M. van Nieuwstadt, M. Rathinam, and R. Murray, "Differential flatness and absolute equivalence," *Proceedings of 1994 33rd IEEE Conference on Decision and Control*, vol. 1, 1994.

[50] D. Mellinger, M. Shomin, and V. Kumar, "Control of Quadrotors for Robust Perching and Landing," in *International Powered Lift Conference*, 2010, pp. 119–126.

[51] K. Sreenath, N. Michael, and V. Kumar, "Trajectory generation and control of a quadrotor with a cable-suspended load - A differentially-flat hybrid system," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2013, pp. 4888–4895.

[52] D. Mellinger, N. Michael, M. Shomin, and V. Kumar, "Recent advances in quadrotor capabilities," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2011, pp. 2964–2965.

[53] "Flying Inverted Pendulum Extended Kalman Filter Webpage," 2015. [Online]. Available: https://risclab.wordpress.com/additional-links/flying-inverted-pendulum-ekf/

[54] A. Bahr, M. R. Walter, and J. J. Leonard, "Consistent cooperative localization," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2009, pp. 3415–3422.

[55] C. H. Chang, S. C. Wang, and C. C. Wang, "Vision-based cooperative simultaneous localization and tracking," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 5191–5197, 2011.

[56] K. T. Song, C. Y. Tsai, and C. H. C. Huang, "Multi-robot cooperative sensing and localization," *Proceedings of the IEEE International Conference on Automation and Logistics, ICAL 2008*, pp. 431–436, 2008.

[57] R. Sharma, S. Quebe, R. Beard, and C. Taylor, "Bearing-only Cooperative Localization," *Journal of Intelligent and Robotic Systems*, vol. 72, pp. 429–440, 2013.

Appendices

Appendix A
ARM-based MCU PPM Signal Generation

The following code was developed to setup a standard or inverted PPM signal using interrupts on a Atmel ARM-based MCU. Register definitions can be found using the "SAM3X / SAM3A Series Atmel — SMART ARM-based MCU Datasheet". Page references in code refer to this datasheet.

```
/****************************************
*   Globals, Constants and Definitions
****************************************/
#define PPM_PIN 2                          // PPM output pin, Pin 2, PB25      **Page41**
#define MAX_PPM_CHANNELS 4                 // Number of PPM channels
#define PPM_FRAME_LENGTH 20000             // Frame length usually 20-22.5ms in microseconds... not critical
#define PPM_PULSE_LENGTH 250               // Pulse length, not critical. This is the high time
#define TICKS_PER_us  42                   // Timer is clocked at 42MHz, so 42 ticks per us

volatile int ppm[MAX_PPM_CHANNELS];        // Array of channels, this will store the commanded periods
int PPMmin = 900;                          // Minimum pulse width in useconds, only used in initialization
```

```
byte PPM_cur_ch = 0;                    // Index of current channel in use. DO NOT USE OUTSIDE OF INTERRUPT
unsigned int PPM_sum = 0;               // Running measure of current used frame. DO NOT USE OUTSIDE OF INTERRUPT


/*******************************************
*     Setup Function runs once to configure
*     all that which requires configuring
*******************************************/
void setup() {
  pinMode(PPM_PIN, OUTPUT);                           // Port B pin 25 configure as output              **Page 6,41**
  analogWrite(PPM_PIN, 1);                            // Allows bypassing Arduino configs

  // Timer counter descriptions starts on  **Page 856**
  REG_PIOB_PDR  = 1 << 25;                            // Disable PIO, enable peripheral                 **Page 634**
  REG_PIOB_ABSR = 1 << 25;                            // Select peripheral B (Timer counter 0 output)   **Page 656**
  REG_TC0_WPMR  = 0x54494D00;                         // Enable write to registers "TIM" in ASCII!      **Page 908**
  REG_TC0_CMR0   = 0b00000000000010011100010000000000;          // Set channel mode register (see datasheet)          **Page 883**
// REG_TC0_CMR0= 0b00000000000001101100010000000000;          // Alternative CMR for inverted output              **Page 883**
  REG_TC0_RC0    = 100000000;                                    // Counter period... just any value to init          **Page 891**
  REG_TC0_CCR0   = 0b101;                                        // Enable clock, software trigger(kick start)        **Page 880**
  REG_TC0_IER0   = 0b00010000;                                   // Enable interrupt on counter = rc                  **Page 894**
  REG_TC0_IDR0   = 0b11101111;                                   // Disable other interrupts                          **Page 896**
  REG_TC0_RA0    = PPM_PULSE_LENGTH * TICKS_PER_us;              // Pulse length setting for RA compare

  for (int i = 0; i < MAX_PPM_CHANNELS; i++) {                   // Set all channels to minimum to avoid interrupt weirdness
    ppm[i] = PPMmin;
  }
  NVIC_EnableIRQ(TC0_IRQn);                                      // Enable TC0 interrupts

}
```

```


/*************************************************
*  Loop Function
*     This function runs repeatedly, for EVER!...
*************************************************/
void loop()
{
  // This is where the value for the PPM are set, values between
      ~900 and ~2000
  // Channel order varies from manufacturer to manufacturer...
  ppm[0] = 1000;
  ppm[1] = 1500;
  ppm[2] = 1500;
  ppm[3] = 1000;

  while(1){}
}


/***************************************
*  TC0 Handler
*     This function is the interrupt
*     handler for Timer Counter 0(TC0)
***************************************/
void TC0_Handler()
{
  long dummy = REG_TC0_SR0;
      // Vital - reading this clears flag, MUST BE DONE!
  if (PPM_cur_ch < MAX_PPM_CHANNELS)
      // Do this if we haven't output sent all the channels yet
  {
    REG_TC0_RC0 = ppm[PPM_cur_ch] * TICKS_PER_us;
        // Update RC to next channel in line
    PPM_sum += ppm[PPM_cur_ch];
        // Add channels run time to the running count of time
    PPM_cur_ch++;
        // Increment channel index for the next interrupt
  }
  else
      // Do this if we've run out of channels to update
  {
    REG_TC0_RC0 = (PPM_FRAME_LENGTH - PPM_sum) * TICKS_PER_us;
        // Update RC to the remaining time so we just idle
    PPM_cur_ch = 0;
        // Reset channel index
    PPM_sum = 0;
        // Reset running count of time
  }
}
```

dueppm.ino

Appendix B
Cooperative Environment Setup Code

The following code was developed to setup a standard environment to allow operating a remote machine with similar permissions and access. To function properly this file should be located in the devel folder within the catkin workspace.

```
#!/usr/bin/env sh
# generated from catkin/cmake/template/setup.sh.in

# Sets various environment variables and sources additional
    environment hooks.
# It tries it's best to undo changes from a previously sourced
    setup file before.
# Supported command line options:
# --extend: skips the undoing of changes from a previously
    sourced setup file

# indigo environment setup

# generated from catkin/cmake/templates/env.sh.in

if [ $# -eq 0 ] ; then
  /bin/echo "Usage: env.sh COMMANDS"
  /bin/echo "Calling env.sh without arguments is not supported
      anymore. Instead spawn a subshell and source a setup file
      manually."
  exit 1
fi

# ensure to not use different shell type which was set before
CATKIN_SHELL=sh

# source setup.sh from same directory as this file
_CATKIN_SETUP_DIR=$(cd "`dirname "$0"`" > /dev/null && pwd)
. "$_CATKIN_SETUP_DIR/setup.sh"
exec "$@"


# since this file is sourced either use the provided
    _CATKIN_SETUP_DIR
# or fall back to the destination set at configure time
: ${_CATKIN_SETUP_DIR:=/home/risc/ros_ws/devel}
_SETUP_UTIL="$_CATKIN_SETUP_DIR/_setup_util.py"
unset _CATKIN_SETUP_DIR

if [ ! -f "$_SETUP_UTIL" ]; then
  echo "Missing Python script: $_SETUP_UTIL"
  return 22
fi

# detect if running on Darwin platform
_UNAME=`uname -s`
_IS_DARWIN=0
if [ "$_UNAME" = "Darwin" ]; then
```

```
  _IS_DARWIN=1
fi
unset _UNAME

# make sure to export all environment variables
export CMAKE_PREFIX_PATH
export CPATH
if [ $_IS_DARWIN -eq 0 ]; then
  export LD_LIBRARY_PATH
else
  export DYLD_LIBRARY_PATH
fi
unset _IS_DARWIN
export PATH
export PKG_CONFIG_PATH
export PYTHONPATH

# remember type of shell if not already set
if [ -z "$CATKIN_SHELL" ]; then
  CATKIN_SHELL=sh
fi

# invoke Python script to generate necessary exports of
    environment variables
_SETUP_TMP=`mktemp /tmp/setup.sh.XXXXXXXXXX`
if [ $? -ne 0 -o ! -f "$_SETUP_TMP" ]; then
  echo "Could not create temporary file: $_SETUP_TMP"
  return 1
```

```
fi
CATKIN_SHELL=$CATKIN_SHELL "$_SETUP_UTIL" $@ > $_SETUP_TMP
_RC=$?
if [ $_RC -ne 0 ]; then
  if [ $_RC -eq 2 ]; then
    echo "Could not write the output of '$_SETUP_UTIL' to
        temporary file '$_SETUP_TMP': may be the disk if full?"
  else
    echo "Failed to run '\"$_SETUP_UTIL\" $@': return code $_RC"
  fi
  unset _RC
  unset _SETUP_UTIL
  rm -f $_SETUP_TMP
  unset _SETUP_TMP
  return 1
fi
unset _RC
unset _SETUP_UTIL
. $_SETUP_TMP
rm -f $_SETUP_TMP
unset _SETUP_TMP

# source all environment hooks
_i=0
while [ $_i -lt $_CATKIN_ENVIRONMENT_HOOKS_COUNT ]; do
  eval _envfile=\$_CATKIN_ENVIRONMENT_HOOKS_$_i
  unset _CATKIN_ENVIRONMENT_HOOKS_$_i
  eval
      _envfile_workspace=\$_CATKIN_ENVIRONMENT_HOOKS_${_i}_WORKSPACE
  unset _CATKIN_ENVIRONMENT_HOOKS_${_i}_WORKSPACE
  # set workspace for environment hook
  CATKIN_ENV_HOOK_WORKSPACE=$_envfile_workspace
  . "$_envfile"
  unset CATKIN_ENV_HOOK_WORKSPACE
  _i=$((_i + 1))
done
unset _i

unset _CATKIN_ENVIRONMENT_HOOKS_COUNT
```

coop.sh

Appendix C
Cooperative Environment Setup Code

The following code was developed to setup a multiprocess experiment across multiple remote machines and agents using a single launch command.

```
<launch>


<!-- Declare Machines -->
<machine name="risc1" address="129.123.5.240"
env-loader="/home/risc/ros_ws/devel/coop.sh"
user="" password="" />
<machine name="risc2" address="129.123.5.200"
env-loader="/home/risc/ros_ws/devel/coop.sh"
user="" password="" />
<machine name="risc3" address="129.123.5.59"
env-loader="/home/risc/ros_ws/devel/coop.sh"
user="" password="" />
<machine name="risc4" address="129.123.85.143"
env-loader="/home/risc/ros_ws/devel/coop.sh"
user="" password="" />


<!-- ========================
     Main Nodes for Operation
     ======================== -->


<!-- Get Cortex marker data Streaming -->
```

```
<node machine="risc1" pkg="cortex_ros"
type="stream_markers" name="stream_markers"
output="screen">
</node>

<!-- Get Cortex State estimation given Templates -->
<node machine="risc1" pkg="risc_estimation"
type="states_estimation.py" name="states_estimation"
output="screen">
</node>

<!-- Get trajectory -->
<node machine="risc1" pkg="risc_control"
type="circles3_traj.py"
name="trajectory"
output="screen">
</node>

<!-- Get Controls -->
<node machine="risc1" pkg="risc_control"
type="generic_DF_controller.py" name="controller"
output="screen">
</node>

<!-- ===============
      Risc 2 Nodes
     =============== -->
```

```
<group ns="risc1">
<!-- Launch the AR.Drone driver -->
<node machine="risc2" name="ardrone_driver"
pkg="ardrone_autonomy"
type="ardrone_driver" output="screen"
clear_params="true">
<!-- Set up rosparams-->
<param name="outdoor" value="0" />
<param name="flight_without_shell" value="0" />

<param name="altitude_max" value="3000" />
<param name="altitude_min" value="50" />
<param name="euler_angle_max" value="0.349066" />
<param name="control_vz_max" value="1000" />
<param name="control_yaw" value=".349066" />
<param name="detect_type" value="10" />
<param name="detections_select_h" value="32" />
<param name="detections_select_v_hsync" value="128" />
<param name="enemy_colors" value="3" />
<param name="enemy_without_shell" value="0" />
</node>

<!-- Launch the joystick publisher -->
<node machine="risc2" name="joy_node" pkg="joy"
type="joy_node"
output="screen" clear_params="true">
<!-- Tell the computer where the joystick is connected.
If you don't know enter in terminal$ ls /dev/input
```

```
and you should see something similar to the value below.-->
    <param name="dev" type="str" value="/dev/input/js0"/>
</node>

<!-- Launch the joystick controller -->
<node machine="risc2" name="joystick_controller"
pkg="risc_control"
type="risc1_controller.py" output="screen"
required="true">

<!-- Configures the joystick button mapping -->
<param name="JoytoWay"                 value="0" />
<param name="ButtonEmergency"         value="2" />
<param name="TakeoffLand"         value="6" />
<param name="Up"           value="9" />
<param name="Down" value="10" />
<param name="ShutDownNode" value="7" />



<!-- Configures the joystick axis mapping -->
<param name="AxisRoll" value="6" />
<param name="AxisPitch" value="7" />
<param name="AxisYaw" value="3" />

<!-- Configures the joystick mapping -->
<param name="ScaleRoll" value="3" />
<param name="ScalePitch" value="3" />
<param name="ScaleYaw" value="1" />
```

```
<param name="ScaleZ" value="1" />

<!-- Configures Transition State Variables -->
<param name="Simulation" value="False" />
<param name="delay"          value="0.3" />
<param name="/controller_status" value="False" />

</node>
</group >

<group ns="/risc1/ardrone/">
<!-- Camera Calibration -->
<node machine="risc2" name="image_proc"
pkg="image_proc"
type="image_proc"/>
<param name="camera_info_url"
value="file://$(find risc_visual)/
camera_calibration_files/risc1.yaml"/>



<!-- ===========================================

NOTE: There must exist the following file path
on the target machine:
  "~/.ros/camera_info/ardrone_front.yaml"
  This is used in initializing the
  camera for calibration.
```

```
============================================ -->


<!-- ===============
      Risc 3 Nodes
     =============== -->

<group ns="risc2">
<!-- Launch the AR.Drone driver -->
<node machine="risc3" name="ardrone_driver"
pkg="ardrone_autonomy"
type="ardrone_driver" output="screen"
clear_params="true">
<!-- Set up rosparams -->
<param name="outdoor" value="0" />
<param name="flight_without_shell" value="0" />

<param name="altitude_max" value="3000" />
<param name="altitude_min" value="50" />
<param name="euler_angle_max" value="0.349066" />
<param name="control_vz_max" value="1000" />
<param name="control_yaw" value=".349066" />

<param name="detect_type" value="10" />
<param name="detections_select_h" value="32" />
<param name="detections_select_v_hsync" value="128" />
<param name="enemy_colors" value="3" />
<param name="enemy_without_shell" value="0" />
```

```
</node>

<!-- Launch the joystick publisher -->
<node machine="risc3" name="joy_node" pkg="joy"
type="joy_node"
output="screen" clear_params="true">
<!-- Tell the computer where the joystick is connected.
If you don’t know enter in terminal$ ls /dev/input
use something similar to the value below.-->
<param name="dev" type="str" value="/dev/input/js0"/>
</node>

<!-- Launch the joystick controller -->
<node machine="risc3" name="joystick_controller"
pkg="risc_control" type="risc2_controller.py"
output="screen" required="true">

<!-- Configures the joystick button mapping -->
<param name="JoytoWay"                 value="0" />
<param name="ButtonEmergency"         value="2" />
<param name="TakeoffLand"         value="6" />
<param name="Up"           value="9" />
<param name="Down" value="10" />
<param name="ShutDownNode" value="7" />


<!-- Configures the joystick axis mapping -->
<param name="AxisRoll" value="6" />
```

```
<param name="AxisPitch" value="7" />

<param name="AxisYaw" value="3" />


<!-- Configures the joystick mapping -->

<param name="ScaleRoll" value="3" />

<param name="ScalePitch" value="3" />

<param name="ScaleYaw" value="1" />

<param name="ScaleZ" value="1" />


<!-- Configures Transition State Variables -->

<param name="Simulation" value="False" />

<param name="delay"          value="0.3" />

<param name="/controller_status" value="False" />


</node>


</group >

<group ns="/risc2/ardrone/">

<!-- Camera Calibration -->

<node machine="risc3" name="image_proc"

pkg="image_proc"

type="image_proc"/>

<param name="camera_info_url"

value="file://$(find risc_visual)/

camera_calibration_files/risc2.yaml"/>

</group >
```

```
<!-- ===============
      Risc 4 Nodes
     =============== -->

<group ns="risc3">
<!-- Launch the AR.Drone driver -->
<node machine="risc4" name="ardrone_driver"
pkg="ardrone_autonomy" type="ardrone_driver"
output="screen" clear_params="true">
<!-- Set up rosparams -->
<param name="outdoor" value="0" />
<param name="flight_without_shell" value="0" />

<param name="altitude_max" value="3000" />
<param name="altitude_min" value="50" />
<param name="euler_angle_max" value="0.349066" />
<param name="control_vz_max" value="1000" />
<param name="control_yaw" value=".349066" />



<param name="detect_type" value="10" />
<param name="detections_select_h" value="32" />
<param name="detections_select_v_hsync" value="128" />
<param name="enemy_colors" value="3" />
<param name="enemy_without_shell" value="0" />
</node>

<!-- Launch the joystick publisher -->
```

```
<node machine="risc4" name="joy_node" pkg="joy"
type="joy_node" output="screen" clear_params="true">
<!-- Tell the computer where the joystick is connected.
If you don’t know enter in terminal$ ls /dev/input
use something similar to the value below.-->
<param name="dev" type="str" value="/dev/input/js0"/>
</node>

<!-- Launch the joystick controller -->
<node machine="risc4" name="joystick_controller"
pkg="risc_control" type="risc3_controller.py"
output="screen" required="true">

<!-- Configures the joystick button mapping -->
<param name="JoytoWay"                 value="0" />
<param name="ButtonEmergency"         value="2" />
<param name="TakeoffLand"         value="6" />
<param name="Up"           value="9" />
<param name="Down" value="10" />
<param name="ShutDownNode" value="7" />



<!-- Configures the joystick axis mapping -->
<param name="AxisRoll" value="6" />
<param name="AxisPitch" value="7" />
<param name="AxisYaw" value="3" />

<!-- Configures the joystick mapping -->
```

```
<param name="ScaleRoll" value="3" />

<param name="ScalePitch" value="3" />

<param name="ScaleYaw" value="1" />

<param name="ScaleZ" value="1" />


<!-- Configures Transition State Variables -->

<param name="Simulation" value="False" />

<param name="delay"          value="0.3" />

<param name="/controller_status" value="False" />


</node>


</group >


<group ns="/risc3/ardrone/">

<!-- Camera Calibration -->

<node machine="risc4" name="image_proc"

pkg="image_proc"

type="image_proc"/>

<param name="camera_info_url"

value="file://$(find risc_visual)/

camera_calibration_files/risc3.yaml"/>

</group >



<!-- =======================================

Nodes for Camera Calibration and Transforms

This Perhaps Should be Run on a Designated
```

```
Image Processing Machine to decrease the
load on other machines.
========================================= -->

<!-- Landmark Recognition -->
<node machine="risc1" name="landmarks"
pkg="risc_visual"
type="risc_landmarks" output="screen"
required="true">
</node >

<!-- Angle Estimation -->
<node machine="risc1" name="angles"
pkg="risc_estimation"
type="angle_estimation.py"
output="screen" required="true">
</node >

<!-- Angle Drawing -->
<node  machine="risc1" name="draw_angles"
pkg="risc_visual"
type="draw_angles" output="screen"
required="true">
</node >

<!-- =========================================
Nodes for Camera Calibration and Transforms
========================================== -->
```

```
<!-- Create Transforms -->
<node machine="risc1" name="transform_tuner"
pkg="risc_control" type="transforms"
output="screen required="true">
</node >

<!-- ================================
Nodes for RVIZ Markers and MarkerArrays
================================== -->

<!-- Draw the RISC MAAP -->
<node machine="risc1" name="draw_risc_maap"
pkg="risc_visual" type="draw_risc_maap"
output="screen" required="true">
</node >

<!-- Draw Cortex Objects -->
<node machine="risc1" name="draw_cortex"
pkg="risc_visual"
type="draw_cortex" output="screen"
required="true">
</node>


<!-- Start RVIZ using Diff_flat config file -->
<node machine="risc1" name="rviz" pkg="rviz"
type="rviz" args="-d
```

```
$(find risc_visual)/config/Trans_tuner.rviz"
output="screen"
required="true">
</node>

<!-- ==========
      Bag Data
     ========== -->

<node machine="risc1" pkg="rosbag" type="record"
name="bagger"
args="record -o /tmp/coop_loc_data
/risc4/ardrone/image_rect_color
/risc2/ardrone/image_rect_color
/risc3/ardrone/image_rect_color
/states3 /cortex_raw /controls
/angles /risc4/ardrone/navdata
/risc2/ardrone/navdata /risc3/ardrone/navdata
/risc2/cmd_vel /risc3/cmd_vel /risc4/cmd_vel
/trajectory /tf /rosout"/>



</launch>
```

Appendix D

Inverted Pendulum Equations of Motion Derivation

D.1 Inverted Pendulum

Let $L =$ length of the pendulum, $m =$ the mass of the pendulum, and x, y, z designate the base of the pendulum while x_v, y_v, ζ designate the center of mass of the pendulum relative to the base in the inertial frame. The center of mass of the pendulum is assumed to be at the tip.

D.2 Description of ζ terms

With the above definition of ζ, the following is known:

$$\zeta = \sqrt{L^2 - x_v^2 - y_v^2} \tag{D.1}$$

Derivatives are taken for future use and defined as follows:

- $\zeta_{x_v} = \frac{\partial \zeta}{\partial x_v} = -\frac{x_v}{\zeta}$
- $\zeta_{y_v} = \frac{\partial \zeta}{\partial y_v} = -\frac{y_v}{\zeta}$
- $\dot{\zeta} = -\frac{x_v \dot{x}_v + y_v \dot{y}_v}{\zeta}$
- $\dot{\zeta}_{\dot{x}_v} = \frac{\partial \dot{\zeta}}{\partial \dot{x}_v} = -\frac{x_v}{\zeta}$
- $\dot{\zeta}_{\dot{y}_v} = \frac{\partial \dot{\zeta}}{\partial \dot{y}_v} = -\frac{y_v}{\zeta}$
- $\dot{\zeta}_{y_v} = \frac{\partial \dot{\zeta}}{\partial y_v} = -(\frac{\dot{y}_v}{\zeta} + \frac{y_v^2 \dot{y}_v}{\zeta^3})$
- $\dot{\zeta}_{x_v} = \frac{\partial \dot{\zeta}}{\partial x_v} = -(\frac{\dot{x}_v}{\zeta} + \frac{x_v^2 \dot{x}_v}{\zeta^3})$

- $\frac{d}{dt}\dot{\zeta}_{\dot{x}_v} = -(\frac{\dot{x}_v}{\zeta} - \frac{x_v\dot{\zeta}}{\zeta^2})$
- $\frac{d}{dt}\dot{\zeta}_{\dot{y}_v} = -(\frac{\dot{y}_v}{\zeta} - \frac{y_v\dot{\zeta}}{\zeta^2})$
- $\ddot{\zeta} = -\frac{x_v\ddot{x_v}+y_v\ddot{y_v}+\dot{x}_v^2+\dot{y}_v^2}{\zeta}$

D.3 Kinetic Energy

$T = \frac{1}{2}m((\dot{x}+\dot{x}_v)^2 + (\dot{y}+\dot{y}_v)^2 + (\dot{z}+\dot{\zeta})^2)$

D.4 Potential Energy

$V = mg(z+\zeta)$

D.5 Legrangian

$\mathcal{L} = T - V = \frac{1}{2}m((\dot{x}+\dot{x}_v)^2 + (\dot{y}+\dot{y}_v)^2 + (\dot{z}+\dot{\zeta})^2) - mg(z-\zeta)$

D.6 Partials and Derivatives

$\frac{d}{dt}\frac{\partial L}{\partial \dot{x}_v} = m\frac{d}{dt}(\dot{x}+\dot{x}_v+\dot{z}\dot{\zeta}_{\dot{x}_v}+\dot{\zeta}\dot{\zeta}_{\dot{x}_v})$

$m(\ddot{x}+\ddot{x_v}+\ddot{z}\dot{\zeta}_{\dot{x}_v}+\dot{z}\frac{d}{dt}\dot{\zeta}_{\dot{x}_v}+\ddot{\zeta}\dot{\zeta}_{\dot{x}_v}+\dot{\zeta}\frac{d}{dt}\dot{\zeta}_{\dot{x}_v})$

$\frac{d}{dt}\frac{\partial L}{\partial \dot{y}_v} = m\frac{d}{dt}(\dot{y}+\dot{y}_v+\dot{z}\dot{\zeta}_{\dot{y}_v}+\dot{\zeta}\dot{\zeta}_{\dot{y}_v})$

$m(\ddot{y}+\ddot{y_v}+\ddot{z}\dot{\zeta}_{\dot{y}_v}+\dot{z}\frac{d}{dt}\dot{\zeta}_{\dot{y}_v}+\ddot{\zeta}\dot{\zeta}_{\dot{y}_v}+\dot{\zeta}\frac{d}{dt}\dot{\zeta}_{\dot{y}_v})$

$\frac{\partial L}{\partial x_v} = m((\dot{z}+\dot{\zeta})\dot{\zeta}_{x_v} - g\zeta_{x_v})$

$\frac{\partial L}{\partial y_v} = m((\dot{z}+\dot{\zeta})\dot{\zeta}_{y_v} - g\zeta_{y_v})$

D.7 Legrangian Mechanics

$\frac{d}{dt}(\frac{\partial L}{\partial \dot{x}_v}) - \frac{\partial L}{\partial x_v} = 0$ and $\frac{d}{dt}(\frac{\partial L}{\partial \dot{y}_v}) - \frac{\partial L}{\partial y_v} = 0$

$$m(\ddot{x} + \ddot{x_v} + \ddot{z}\dot{\zeta}_{\dot{x}_v} + \dot{z}\tfrac{d}{dt}\dot{\zeta}_{\dot{x}_v} + \ddot{\zeta}\dot{\zeta}_{\dot{x}_v} + \dot{\zeta}\tfrac{d}{dt}\dot{\zeta}_{\dot{x}_v}) - (m((\dot{z} + \dot{\zeta})\dot{\zeta}_{x_v} - g\zeta_{x_v})) = 0$$

Expand only the terms containing $\ddot{x_v}$. Therefore, only $\ddot{\zeta}$ term is expanded.

$$m(\ddot{x} + \ddot{x_v} + \ddot{z}\dot{\zeta}_{\dot{x}_v} + \dot{z}\tfrac{d}{dt}\dot{\zeta}_{\dot{x}_v} + (-\tfrac{x_v\ddot{x_v} + y_v\ddot{y_v} + \dot{x}_v^2 + \dot{y}_v^2}{\zeta})\dot{\zeta}_{\dot{x}_v} + \dot{\zeta}\tfrac{d}{dt}\dot{\zeta}_{\dot{x}_v}) - (m((\dot{z} + \dot{\zeta})\dot{\zeta}_{x_v} - g\zeta_{x_v})) = 0$$

$$\ddot{x_v}(1 - \tfrac{x_v\dot{\zeta}_{\dot{x}_v}}{\zeta}) + \ddot{x} + \ddot{z}\dot{\zeta}_{\dot{x}_v} + \dot{z}\tfrac{d}{dt}\dot{\zeta}_{\dot{x}_v} - \tfrac{y_v\ddot{y_v} + \dot{x}_v^2 + \dot{y}_v^2}{\zeta}\dot{\zeta}_{\dot{x}_v} + \dot{\zeta}\tfrac{d}{dt}\dot{\zeta}_{\dot{x}_v} - (\dot{z} + \dot{\zeta})\dot{\zeta}_{x_v} + g\zeta_{x_v}) = 0$$

$$\ddot{x_v} = \tfrac{1}{(1 - \frac{x_v\dot{\zeta}_{\dot{x}_v}}{\zeta})}(-\ddot{x} - \ddot{z}\dot{\zeta}_{\dot{x}_v} - \dot{z}\tfrac{d}{dt}\dot{\zeta}_{\dot{x}_v} + \tfrac{y_v\ddot{y_v} + \dot{x}_v^2 + \dot{y}_v^2}{\zeta}\dot{\zeta}_{\dot{x}_v} - \dot{\zeta}\tfrac{d}{dt}\dot{\zeta}_{\dot{x}_v} + (\dot{z} + \dot{\zeta})\dot{\zeta}_{x_v} - g\zeta_{x_v})$$

Expand and simplify

$$\ddot{x_v} = \tfrac{1}{(1 - \frac{x_v\dot{\zeta}_{\dot{x}_v}}{\zeta})}(-\ddot{x} - \ddot{z}\dot{\zeta}_{\dot{x}_v} - \dot{z}\tfrac{d}{dt}\dot{\zeta}_{\dot{x}_v} + \tfrac{y_v\ddot{y_v} + \dot{x}_v^2 + \dot{y}_v^2}{\zeta}\dot{\zeta}_{\dot{x}_v} - \dot{\zeta}\tfrac{d}{dt}\dot{\zeta}_{\dot{x}_v} + (\dot{z} + \dot{\zeta})\dot{\zeta}_{x_v} - g\zeta_{x_v})$$

Expand terms

$$\ddot{x_v} = \frac{1}{(1+\frac{x_v^2}{\zeta^2})}(-\ddot{x} + \frac{\ddot{z}x_v}{\zeta} + \frac{\dot{z}\dot{x}_v}{\zeta} - \frac{\ddot{y_v}x_vy_v}{\zeta^2} - \frac{2\dot{x}_v^2x_v}{\zeta^2} - \frac{\dot{x}_v\dot{y}_vy_v}{\zeta^2} - \frac{\dot{y}_v^2x_v}{\zeta^2} - \frac{\dot{z}\dot{y}_vx_vy_v}{\zeta^3} + \frac{\dot{z}\dot{x}_vx_v^2}{\zeta^3} - \frac{2\dot{x}_v^2x_v^3}{\zeta^4} - \frac{4\dot{x}_v\dot{y}_vx_v^2y_v}{\zeta^4} - \frac{2\dot{y}_v^2y_v^2x_v}{\zeta^4} - \frac{\dot{z}\dot{x}_v}{\zeta} - \frac{\dot{z}\dot{x}_vx_v^2}{\zeta^3} - \frac{\dot{z}\dot{y}_vx_vy_v}{\zeta^3} - \frac{\dot{\zeta}\dot{x}_v}{\zeta} - \frac{\dot{\zeta}\dot{x}_vx_v^2}{\zeta^3} - \frac{\dot{\zeta}\dot{y}_vx_vy_v}{\zeta^3} + \frac{gx_v}{\zeta})$$

$$\ddot{x_v} = \frac{1}{(1+\frac{x_v^2}{\zeta^2})}(-\ddot{x} + \frac{\ddot{z}x_v}{\zeta} - \frac{\ddot{y_v}x_vy_v}{\zeta^2} - \frac{\dot{x}_v^2x_v}{\zeta^2} - \frac{\dot{y}_v^2x_v}{\zeta^2} - \frac{2\dot{x}_v^2x_v^3}{\zeta^4} - \frac{4\dot{x}_v\dot{y}_vx_v^2y_v}{\zeta^4} - \frac{2\dot{y}_v^2y_v^2x_v}{\zeta^4} + \frac{\dot{x}_v^2x_v^3}{\zeta^4} + \frac{\dot{x}_v\dot{y}_vx_v^2y_v}{\zeta^4} - \frac{\dot{x}_v\dot{y}_vx_v^2y_v}{\zeta^4} + \frac{\dot{y}_v^2y_v^2x_v}{\zeta^4} + \frac{gx_v}{\zeta})$$

Simplified to individual Terms

$$\ddot{x_v} = \tfrac{\zeta^2}{(\zeta^2 + x_v^2)}(-\ddot{x} + \tfrac{\ddot{z}x_v}{\zeta} - \tfrac{\ddot{y_v}x_vy_v}{\zeta^2} - \tfrac{\dot{x}_v^2x_v}{\zeta^2} - \tfrac{\dot{y}_v^2x_v}{\zeta^2} - \tfrac{\dot{x}_v^2x_v^3}{\zeta^4} - \tfrac{2\dot{x}_v\dot{y}_vx_v^2y_v}{\zeta^4} - \tfrac{\dot{y}_v^2y_v^2x_v}{\zeta^4} + \tfrac{gx_v}{\zeta})$$

$$\ddot{y_v} = \tfrac{\zeta^2}{(\zeta^2 + y_v^2)}(-\ddot{y} + \tfrac{\ddot{z}y_v}{\zeta} - \tfrac{\ddot{x_v}x_vy_v}{\zeta^2} - \tfrac{\dot{y}_v^2y_v}{\zeta^2} - \tfrac{\dot{x}_v^2y_v}{\zeta^2} - \tfrac{\dot{y}_v^2y_v^3}{\zeta^4} - \tfrac{2\dot{y}_v\dot{x}_vy_v^2x_v}{\zeta^4} - \tfrac{\dot{x}_v^2x_v^2y_v}{\zeta^4} + \tfrac{gy_v}{\zeta})$$

$$\alpha = \frac{\zeta^2}{(\zeta^2 + x_v^2)}(-\ddot{x} + \frac{\ddot{z}x_v}{\zeta} - \frac{\dot{x}_v^2x_v}{\zeta^2} - \frac{\dot{y}_v^2x_v}{\zeta^2} - \frac{\dot{x}_v^2x_v^3}{\zeta^4} - \frac{2\dot{x}_v\dot{y}_vx_v^2y_v}{\zeta^4} - \frac{\dot{y}_v^2y_v^2x_v}{\zeta^4} + \frac{gx_v}{\zeta})$$

$$\beta = \frac{\zeta^2}{(\zeta^2 + y_v^2)}(-\ddot{y} + \frac{\ddot{z}y_v}{\zeta} - \frac{\dot{y}_v^2y_v}{\zeta^2} - \frac{\dot{x}_v^2y_v}{\zeta^2} - \frac{\dot{y}_v^2y_v^3}{\zeta^4} - \frac{2\dot{y}_v\dot{x}_vy_v^2x_v}{\zeta^4} - \frac{\dot{x}_v^2x_v^2y_v}{\zeta^4} + \frac{gy_v}{\zeta})$$

Thus,

$\ddot{x_v} = \alpha - \frac{\ddot{y_v} x_v y_v}{(\zeta^2 + x_v^2)}$

$\ddot{y_v} = \beta - \frac{\ddot{x_v} x_v y_v}{(\zeta^2 + y_v^2)}$

$\ddot{x_v}$ is still in terms of $\ddot{y_v}$. Solving for $\ddot{x_v}$ we see:

$\ddot{x_v} = (\alpha - \frac{\beta x_v y_v}{(\zeta^2 + x_v^2)\zeta^2})(1 - \frac{x_v^2 y_v^2}{(\zeta^2 + x_v^2)^2 \zeta^4})$

Our Derivation results

$\ddot{x_v} = \frac{\alpha(\zeta^2 + x_v^2) - \beta x_v y_v}{(\zeta^2 + x_v^2) - \frac{x_v^2 y_v^2}{\zeta^2 + y_v^2}}$

$\ddot{y_v} = \beta - \frac{\ddot{x_v} x_v y_v}{(\zeta^2 + y_v^2)}$

Appendix E

Quadrotor Jacobian Derivation

Starting from the Quadrotor Equations (1-3):

$\dot{P}_n = c\theta c\psi u + (s\phi s\theta c\psi - c\phi s\psi)v + (c\phi s\theta c\psi + s\phi s\psi)w$

$\frac{\partial \dot{P}_n}{\partial u} = c\theta c\psi$

$\frac{\partial \dot{P}_n}{\partial v} = (s\phi s\theta c\psi - c\phi s\psi)$

$\frac{\partial \dot{P}_n}{\partial w} = (c\phi s\theta c\psi + s\phi s\psi)$

$\frac{\partial \dot{P}_n}{\partial \phi} = (c\phi s\theta c\psi + s\phi s\psi)v + (c\phi s\psi - s\phi s\theta c\psi)w$

$\frac{\partial \dot{P}_n}{\partial \theta} = -s\theta c\psi u + (s\phi c\theta c\psi)v + (c\phi c\theta c\psi)w$

$\frac{\partial \dot{P}_n}{\partial \psi} = -c\theta s\psi u + (-s\phi s\theta s\psi - c\phi c\psi)v + (-c\phi s\theta s\psi + s\phi c\psi)w$

$\dot{P}_e = c\phi s\theta c\psi u + (s\phi s\theta c\psi + c\phi c\psi)v + (c\phi s\theta s\psi - s\phi c\psi)w$

$\frac{\partial \dot{P}_e}{\partial u} = c\phi s\theta c\psi$

$\frac{\partial \dot{P}_e}{\partial v} = (s\phi s\theta c\psi + c\phi c\psi)$

$\frac{\partial \dot{P}_e}{\partial w} = (c\phi s\theta s\psi - s\phi c\psi)$

$\frac{\partial \dot{P}_e}{\partial \phi} = -s\phi s\theta c\psi u + (c\phi s\theta c\psi - s\phi c\psi)v + (-s\phi s\theta s\psi - c\phi c\psi)w$

$\frac{\partial \dot{P}_e}{\partial \theta} = c\phi c\theta c\psi u + (s\phi c\theta c\psi)v + (c\phi c\theta s\psi)w$

$\frac{\partial \dot{P}_e}{\partial \psi} = -c\phi s\theta s\psi u + (-s\phi s\theta s\psi - c\phi s\psi)v + (c\phi s\theta c\psi + s\phi s\psi)w$

$\dot{h} = \sin(\theta)u - \sin(\phi)\cos(\theta)v - \cos(\phi)\cos(\theta)w$

$\frac{\partial \dot{h}}{\partial u} = \sin(\theta)$

$\frac{\partial \dot{h}}{\partial v} = -\sin(\phi)\cos(\theta)$

$\frac{\partial \dot{h}}{\partial w} = -\cos(\phi)\cos(\theta)$

$\frac{\partial \dot{h}}{\partial \phi} = -\cos(\phi)\cos(\theta)v + \sin(\phi)\cos(\theta)w$

$\frac{\partial \dot{h}}{\partial \theta} = \cos(\theta)u + \sin(\phi)\sin(\theta)v + \cos(\phi)\sin(\theta)w$

$\frac{\partial \dot{h}}{\partial \psi} = 0$

$\dot{u} = rv - qw - g\sin(\theta)$

$\frac{\partial \dot{u}}{\partial u} = 0$

$\frac{\partial \dot{u}}{\partial v} = r$

$\frac{\partial \dot{u}}{\partial w} = -q$

$\frac{\partial \dot{u}}{\partial \phi} = 0$

$\frac{\partial \dot{u}}{\partial \theta} = -g\cos(\theta)$

$\frac{\partial \dot{u}}{\partial \psi} = 0$

$\frac{\partial \dot{u}}{\partial p} = 0$

$\frac{\partial \dot{u}}{\partial q} = -w$

$\frac{\partial \dot{u}}{\partial r} = v$

$\frac{\partial \dot{u}}{\partial F} = 0$

$\dot{v} = pw - ru + g\cos(\theta)\sin(\phi)$

$\frac{\partial \dot{v}}{\partial u} = -r$

$\frac{\partial \dot{v}}{\partial v} = 0$

$\frac{\partial \dot{v}}{\partial w} = p$

$\frac{\partial \dot{v}}{\partial \phi} = g\cos(\theta)\cos(\phi)$

$\frac{\partial \dot{v}}{\partial \theta} = -g\sin(\theta)\sin(\phi)$

$\frac{\partial \dot{v}}{\partial \psi} = 0$

$\frac{\partial \dot{v}}{\partial p} = w$

$\frac{\partial \dot{v}}{\partial q} = 0$

$\frac{\partial \dot{v}}{\partial r} = -u$

$\frac{\partial \dot{v}}{\partial F} = 0$

$\dot{w} = qu - pv + g\cos(\theta)\cos(\phi) - \frac{F}{m}$

$\frac{\partial \dot{w}}{\partial u} = q$

$\frac{\partial \dot{w}}{\partial v} = -p$

$\frac{\partial \dot{w}}{\partial w} = 0$

$\frac{\partial \dot{w}}{\partial \phi} = -g\cos(\theta)\sin(\phi)$

$\frac{\partial \dot{w}}{\partial \theta} = -g\sin(\theta)\cos(\phi)$

$\frac{\partial \dot{w}}{\partial \psi} = 0$

$\frac{\partial \dot{w}}{\partial p} = -v$

$\frac{\partial \dot{w}}{\partial q} = u$

$\frac{\partial \dot{w}}{\partial r} = 0$

$\frac{\partial \dot{w}}{\partial F} = -\frac{1}{m}$

$\dot{\phi} = p + \sin(\phi)\tan(\theta)q + \cos(\phi)\tan(\theta)r$

$\frac{\partial \dot{\phi}}{\partial u} = 0$

$\frac{\partial \dot{\phi}}{\partial v} = 0$

$\frac{\partial \dot{\phi}}{\partial w} = 0$

$\frac{\partial \dot{\phi}}{\partial \phi} = \cos(\phi)\tan(\theta)q - \sin(\phi)\tan(\theta)r$

$\frac{\partial \dot{\phi}}{\partial \theta} = \sin(\phi)\sec^2(\theta)q + \cos(\phi)\sec^2(\theta)r$

$\frac{\partial \dot{\phi}}{\partial \psi} = 0$

$\frac{\partial \dot{\phi}}{\partial p} = 1$

$\frac{\partial \dot{\phi}}{\partial q} = \sin(\phi)\tan(\theta)$

$\frac{\partial \dot{\phi}}{\partial r} = \cos(\phi)\tan(\theta)$

$\dot{\theta} = \cos(\phi)q - \sin(\phi)r$

$\frac{\partial \dot{\theta}}{\partial u} = 0$

$\frac{\partial \dot{\theta}}{\partial v} = 0$

$\frac{\partial \dot{\theta}}{\partial w} = 0$

$\frac{\partial \dot{\theta}}{\partial \phi} = -\sin(\phi)q - \cos(\phi)r$

$\frac{\partial \dot{\theta}}{\partial \theta} = 0$

$\frac{\partial \dot{\theta}}{\partial \psi} = 0$

$$\frac{\partial \dot{\theta}}{\partial p} = 0$$
$$\frac{\partial \dot{\theta}}{\partial q} = \cos(\phi)$$
$$\frac{\partial \dot{\theta}}{\partial r} = -\sin(\phi)$$

$$\dot{\psi} = \frac{\sin(\phi)}{\cos(\theta)}q + \frac{\cos(\phi)}{\cos(\theta)}r$$
$$\frac{\partial \dot{\psi}}{\partial u} = 0$$
$$\frac{\partial \dot{\psi}}{\partial v} = 0$$
$$\frac{\partial \dot{\psi}}{\partial w} = 0$$
$$\frac{\partial \dot{\psi}}{\partial \phi} = \frac{\cos(\phi)}{\cos(\theta)}q + \frac{-\sin(\phi)}{\cos(\theta)}r$$
$$\frac{\partial \dot{\psi}}{\partial \theta} = (\sin(\phi)q + \cos(\phi)r)\frac{\sin(\theta)}{\cos^2(\theta)}$$
$$\frac{\partial \dot{\psi}}{\partial \psi} = 0$$
$$\frac{\partial \dot{\psi}}{\partial p} = 0$$
$$\frac{\partial \dot{\psi}}{\partial q} = \frac{\sin(\phi)}{\cos(\theta)}$$
$$\frac{\partial \dot{\psi}}{\partial r} = \frac{\cos(\phi)}{\cos(\theta)}$$

Appendix F

Inverted Pendulum Jacobian Derivation

Equations to Differentiate:

$\ddot{x}_v = \frac{\alpha(\zeta^2+x_v^2)-\beta x_v y_v}{(\zeta^2+x_v^2)-\frac{x_v^2 y_v^2}{\zeta^2+y_v^2}}$ and $\ddot{y}_v = \beta - \frac{\ddot{x_v} x_v y_v}{(\zeta^2+y_v^2)}$

$\underline{\ddot{\mathbf{y_v}} = \beta - \frac{\ddot{\mathbf{x_v}}\mathbf{x_v y_v}}{(\zeta^2+\mathbf{y_v^2})}}$

$$\frac{\partial \ddot{y}_v}{\partial \ddot{x}} = \frac{\partial \ddot{x}_v}{\partial \ddot{x}}(\frac{x_v y_v}{\zeta^2+y_v^2})$$

$$\frac{\partial \ddot{y}_v}{\partial \ddot{y}} = \frac{\partial \beta}{\partial \ddot{y}} - \frac{\partial \ddot{x}_v}{\partial \ddot{y}}(\frac{x_v y_v}{\zeta^2+y_v^2})$$

$$\frac{\partial \ddot{y}_v}{\partial \ddot{z}} = \frac{\partial \beta}{\partial \ddot{z}} - \frac{\partial \ddot{x}_v}{\partial \ddot{z}}(\frac{x_v y_v}{\zeta^2+y_v^2})$$

$$\frac{\partial \ddot{y}_v}{\partial \dot{x}_v} = \frac{\partial \beta}{\partial \dot{x}_v} - \frac{\partial \ddot{x}_v}{\partial \dot{x}_v}(\frac{x_v y_v}{\zeta^2+y_v^2})$$

$$\frac{\partial \ddot{y}_v}{\partial \dot{y}_v} = \frac{\partial \beta}{\partial \dot{y}_v} - \frac{\partial \ddot{x}_v}{\partial \dot{y}_v}(\frac{x_v y_v}{\zeta^2+y_v^2})$$

$$\frac{\partial \ddot{y}_v}{\partial x_v} = \frac{\partial \beta}{\partial x_v} - \frac{\partial \ddot{x}_v}{\partial x_v}(\frac{x_v y_v}{\zeta^2+y_v^2}) - \ddot{x}_v(\frac{\zeta^2 y_v + y_v^3 + 2x_v^2 y_v}{(\zeta^2+y_v^2)^2})$$

$$\frac{\partial \ddot{y}_v}{\partial y_v} = \frac{\partial \beta}{\partial y_v} - \frac{\partial \ddot{x}_v}{\partial y_v}(\frac{x_v y_v}{\zeta^2+y_v^2}) - \ddot{x}_v(\frac{x_v}{\zeta^2+y_v^2})$$

$\underline{\ddot{\mathbf{x}}_\mathbf{v} = \frac{\alpha(\zeta^2+\mathbf{x_v^2})(\zeta^2+\mathbf{y_v^2})-\beta \mathbf{x_v y_v}(\zeta^2+\mathbf{y_v^2})}{\zeta^2(\zeta^2+\mathbf{x_v^2}+\mathbf{y_v^2})}}$

$$\frac{\partial \ddot{x}_v}{\partial \ddot{x}} = \frac{\partial \alpha}{\partial \ddot{x}}(\frac{(\zeta^2+x_v^2)(\zeta^2+y_v^2)}{\zeta^2(\zeta^2+x_v^2+y_v^2)})$$

$$\frac{\partial \ddot{x}_v}{\partial \ddot{y}} = \frac{\partial \alpha}{\partial \ddot{y}}(\frac{(\zeta^2+x_v^2)(\zeta^2+y_v^2)}{\zeta^2(\zeta^2+x_v^2+y_v^2)}) - \frac{\partial \beta}{\partial \ddot{y}}(\frac{x_v y_v(\zeta^2+y_v^2)}{\zeta^2(\zeta^2+x_v^2+y_v^2)})$$

$$\frac{\partial \ddot{x}_v}{\partial \ddot{z}} = \frac{\partial \alpha}{\partial \ddot{z}}(\frac{(\zeta^2+x_v^2)(\zeta^2+y_v^2)}{\zeta^2(\zeta^2+x_v^2+y_v^2)}) - \frac{\partial \beta}{\partial \ddot{z}}(\frac{x_v y_v(\zeta^2+y_v^2)}{\zeta^2(\zeta^2+x_v^2+y_v^2)})$$

$$\frac{\partial \ddot{x}_v}{\partial \dot{x}_v} = \frac{\partial \alpha}{\partial \dot{x}_v}(\frac{(\zeta^2+x_v^2)(\zeta^2+y_v^2)}{\zeta^2(\zeta^2+x_v^2+y_v^2)}) - \frac{\partial \beta}{\partial \dot{x}_v}(\frac{x_v y_v(\zeta^2+y_v^2)}{\zeta^2(\zeta^2+x_v^2+y_v^2)})$$

$$\frac{\partial \ddot{x}_v}{\partial \dot{y}_v} = \frac{\partial \alpha}{\partial \dot{y}_v}(\frac{(\zeta^2+x_v^2)(\zeta^2+y_v^2)}{\zeta^2(\zeta^2+x_v^2+y_v^2)}) - \frac{\partial \beta}{\partial \dot{y}_v}(\frac{x_v y_v(\zeta^2+y_v^2)}{\zeta^2(\zeta^2+x_v^2+y_v^2)})$$

$$\frac{\partial \ddot{x}_v}{\partial x_v} = \frac{\partial \alpha}{\partial x_v}(\frac{(\zeta^2+x_v^2)(\zeta^2+y_v^2)}{\zeta^2(\zeta^2+x_v^2+y_v^2)}) + \alpha(\frac{2x_v y_v^2(\zeta^2+x_v^2)}{\zeta^4(\zeta^2+x_v^2+y_v^2)}) - \cdots$$

$$\cdots \frac{\partial\beta}{\partial x_v}(\frac{x_v y_v(\zeta^2+y_v^2)}{\zeta^2(\zeta^2+x_v^2+y_v^2)}) - \beta(\frac{y_v(\zeta^4+\zeta^2 y_v^2-2\zeta^2 x_v^2+2\zeta^2 x_v y_v+2x_v y_v^3)}{\zeta^4(\zeta^2+x_v^2+y_v^2)})$$

$$\frac{\partial\ddot{x}_v}{\partial y_v} = \frac{\partial\alpha}{\partial y_v}(\frac{(\zeta^2+x_v^2)(\zeta^2+y_v^2)}{\zeta^2(\zeta^2+x_v^2+y_v^2)}) + \alpha(\frac{2(\zeta^2+x_v^2)(x_v^2 y_v-\zeta^2 x_v+\zeta^2 y_v)}{\zeta^4(\zeta^2+x_v^2+y_v^2)}) - \cdots$$
$$\cdots \frac{\partial\beta}{\partial y_v}(\frac{x_v y_v(\zeta^2+y_v^2)}{\zeta^2(\zeta^2+x_v^2+y_v^2)}) - \beta(\frac{(\zeta^2+y_v^2)(\zeta^2 x_v+2x_v y_v^2)}{\zeta^4(\zeta^2+x_v^2+y_v^2)})$$

Plug the $\ddot{x}_v$ derivatives back into the $\ddot{y}_v$ derivatives:

$$\underline{\mathbf{\ddot{y}_v} = \beta - \frac{\mathbf{\ddot{x}_v x_v y_v}}{(\boldsymbol{\zeta}^\mathbf{2}+\mathbf{y_v^2})}}$$

$$\frac{\partial\ddot{y}_v}{\partial\ddot{x}} = \frac{\partial\alpha}{\partial\ddot{x}}(\frac{(\zeta^2+x_v^2)x_v y_v}{\zeta^2(\zeta^2+x_v^2+y_v^2)})$$

$$\frac{\partial\ddot{y}_v}{\partial\ddot{y}} = \frac{\partial\beta}{\partial\ddot{y}} - \frac{\partial\alpha}{\partial\ddot{y}}(\frac{(\zeta^2+x_v^2)x_v y_v}{\zeta^2(\zeta^2+x_v^2+y_v^2)}) + \frac{\partial\beta}{\partial\ddot{y}}(\frac{x_v^2 y_v^2}{\zeta^2(\zeta^2+x_v^2+y_v^2)})$$

$$\frac{\partial\ddot{y}_v}{\partial\ddot{z}} = \frac{\partial\beta}{\partial\ddot{z}} - \frac{\partial\alpha}{\partial\ddot{z}}(\frac{(\zeta^2+x_v^2)x_v y_v}{\zeta^2(\zeta^2+x_v^2+y_v^2)}) + \frac{\partial\beta}{\partial\ddot{z}}(\frac{x_v^2 y_v^2}{\zeta^2(\zeta^2+x_v^2+y_v^2)})$$

$$\frac{\partial\ddot{y}_v}{\partial\dot{x}_v} = \frac{\partial\beta}{\partial\dot{x}_v} - \frac{\partial\alpha}{\partial\dot{x}_v}(\frac{(\zeta^2+x_v^2)x_v y_v}{\zeta^2(\zeta^2+x_v^2+y_v^2)}) + \frac{\partial\beta}{\partial\dot{x}_v}(\frac{x_v^2 y_v^2}{\zeta^2(\zeta^2+x_v^2+y_v^2)})$$

$$\frac{\partial\ddot{y}_v}{\partial\dot{y}_v} = \frac{\partial\beta}{\partial\dot{y}_v} - \frac{\partial\alpha}{\partial\dot{y}_v}(\frac{(\zeta^2+x_v^2)x_v y_v}{\zeta^2(\zeta^2+x_v^2+y_v^2)}) + \frac{\partial\beta}{\partial\dot{y}_v}(\frac{x_v^2 y_v^2}{\zeta^2(\zeta^2+x_v^2+y_v^2)})$$

$$\frac{\partial\ddot{y}_v}{\partial x_v} = \frac{\partial\beta}{\partial x_v}(1 + \frac{x_v^2 y_v^2}{\zeta^2(\zeta^2+x_v^2+y_v^2)}) - \frac{\partial\alpha}{\partial x_v}(\frac{(\zeta^2+x_v^2)x_v y_v}{\zeta^2(\zeta^2+x_v^2+y_v^2)}) - \ \ldots$$
$$\ldots\ \alpha(\frac{y_v(\zeta^2+x_v^2)(\zeta^2+2x_v^2)}{\zeta^4(\zeta^2+x_v^2+y_v^2)}) + \beta(\frac{2x_v y_v^2(x_v^2\zeta^2+x_v y_v^3+x_v y_v\zeta^2+y_v^2\zeta^2+\zeta^4)}{(\zeta^2+y_v^2)\zeta^4(\zeta^2+x_v^2+y_v^2)})$$

$$\frac{\partial\ddot{y}_v}{\partial y_v} = \frac{\partial\beta}{\partial y_v}(1 + \frac{x_v y_v(\zeta^2+y_v^2)}{\zeta^2(\zeta^2+x_v^2+y_v^2)}) - \frac{\partial\alpha}{\partial y_v}(\frac{(\zeta^2+x_v^2)(\zeta^2+y_v^2)}{\zeta^2(\zeta^2+x_v^2+y_v^2)}) - \ \ldots$$
$$\ldots\ \alpha(\frac{(\zeta^2+x_v^2)(2x_v^2 y_v-\zeta^2 x_v+2\zeta^2 y_v)}{\zeta^4(\zeta^2+x_v^2+y_v^2)}) + \beta(\frac{(\zeta^2+y_v^2)(\zeta^2 x_v+2x_v y_v^2)+x_v^2 y_v\zeta^2}{\zeta^4(\zeta^2+x_v^2+y_v^2)})$$

$$\underline{\alpha = \frac{\boldsymbol{\zeta}^\mathbf{2}}{(\boldsymbol{\zeta}^\mathbf{2}+\mathbf{x_v^2})}(-\mathbf{\ddot{X}} + \frac{\mathbf{\ddot{z}x_v}}{\boldsymbol{\zeta}} - \frac{\mathbf{\dot{x}_v^2 x_v}}{\boldsymbol{\zeta}^\mathbf{2}} - \frac{\mathbf{\dot{y}_v^2 x_v}}{\boldsymbol{\zeta}^\mathbf{2}} - \frac{\mathbf{\dot{x}_v^2 x_v^3}}{\boldsymbol{\zeta}^\mathbf{4}} - \frac{\mathbf{2\dot{x}_v\dot{y}_v x_v^2 y_v}}{\boldsymbol{\zeta}^\mathbf{4}} - \frac{\mathbf{\dot{y}_v^2 y_v^2 x_v}}{\boldsymbol{\zeta}^\mathbf{4}} + \frac{\mathbf{g x_v}}{\boldsymbol{\zeta}})}$$

$$\frac{\partial\alpha}{\partial\ddot{x}} = \frac{-\zeta^2}{\zeta^2+x_v^2}$$

$$\frac{\partial\alpha}{\partial\ddot{y}} = 0$$

$$\frac{\partial\alpha}{\partial\ddot{z}} = \frac{\zeta x_v}{\zeta^2+x_v^2}$$

$$\frac{\partial\alpha}{\partial\dot{x}_v} = -\frac{2\dot{x}_v x_v}{\zeta^2+x_v^2} - \frac{2\dot{x}_v x_v^3}{\zeta^4+\zeta^2 x_v^2} - \frac{2\dot{y}_v x_v^2 y_v}{\zeta^4+\zeta^2 x_v^2}$$

$$\frac{\partial\alpha}{\partial\dot{y}_v} = -\frac{2\dot{y}_v x_v}{\zeta^2+x_v^2} - \frac{2\dot{x}_v x_v^2 y_v}{\zeta^4+\zeta^2 x_v^2} - \frac{2\dot{y}_v x_v y_v^2}{\zeta^4+\zeta^2 x_v^2}$$

$$\frac{\partial\alpha}{\partial x_v} = \frac{2x_v\ddot{x}-\dot{x}_v^2-\dot{y}_v^2}{\zeta^2+x_v^2} + \frac{(\ddot{z}+g)(\zeta^2-x_v^2)}{\zeta(\zeta^2+x_v^2)} - \ \ldots$$
$$\ldots\ \frac{(\dot{x}_v x_v+\dot{y}_v y_v)(2\dot{x}_v x_v^3+2\dot{y}_v y_v x_v^2+3\dot{x}_v x_v\zeta^2+\dot{y}_v y_v\zeta^2)}{\zeta^4(\zeta^2+x_v^2)}$$

$$\frac{\partial\alpha}{\partial y_v} = \frac{2\ddot{x}x_v^2y_v}{(\zeta^2+x_v^2)^2} - \frac{2\dot{x}_v^2x_vy_v}{(\zeta^2+x_v^2)^2} - \frac{2\dot{y}_v^2x_vy_v}{(\zeta^2+x_v^2)^2} + \frac{x_vy_v(\ddot{z}+g)(\zeta^2-x_v^2)}{\zeta(\zeta^2+x_v^2)^2} - \;\ldots$$

$$\ldots\; \frac{(4\zeta^2y_v+2y_vx_v^2)(\dot{x}_v^2x_v^3+2\dot{x}_v\dot{y}_vx_v^2y_v+\dot{y}_v^2y_v^2x_v)+\zeta^2(\zeta^2+x_v^2)(2\dot{x}_v\dot{y}_vx_v^2+2\dot{y}_v^2y_vx_v)}{\zeta^4(\zeta^2+x_v^2)^2}$$

$$\underline{\beta = \frac{\boldsymbol{\zeta^2}}{(\boldsymbol{\zeta^2}+\mathbf{y_v^2})}\left(-\ddot{\mathbf{y}} + \frac{\ddot{\mathbf{z}}\mathbf{y_v}}{\zeta} - \frac{\dot{\mathbf{y}}_\mathbf{v}^\mathbf{2}\mathbf{y_v}}{\boldsymbol{\zeta^2}} - \frac{\dot{\mathbf{x}}_\mathbf{v}^\mathbf{2}\mathbf{y_v}}{\boldsymbol{\zeta^2}} - \frac{\dot{\mathbf{y}}_\mathbf{v}^\mathbf{2}\mathbf{y}_\mathbf{v}^\mathbf{3}}{\boldsymbol{\zeta^4}} - \frac{\mathbf{2}\dot{\mathbf{y}}_\mathbf{v}\dot{\mathbf{x}}_\mathbf{v}\mathbf{y}_\mathbf{v}^\mathbf{2}\mathbf{x_v}}{\boldsymbol{\zeta^4}} - \frac{\dot{\mathbf{x}}_\mathbf{v}^\mathbf{2}\mathbf{x}_\mathbf{v}^\mathbf{2}\mathbf{y_v}}{\boldsymbol{\zeta^4}} + \frac{\mathbf{g}\mathbf{y_v}}{\zeta}\right)}$$

$$\frac{\partial\beta}{\partial\ddot{x}} = 0$$

$$\frac{\partial\beta}{\partial\ddot{y}} = -\frac{\zeta^2}{\zeta^2+y_v^2}$$

$$\frac{\partial\beta}{\partial\ddot{z}} = \frac{\zeta y_v}{\zeta^2+y_v^2}$$

$$\frac{\partial\beta}{\partial\dot{x}_v} = -\frac{2y_v(\dot{x}_v(\zeta^2+x_v^2)+\dot{y}_vy_vx_v)}{\zeta^2(\zeta^2+y_v^2)}$$

$$\frac{\partial\beta}{\partial\dot{y}_v} = -\frac{2y_v(\dot{y}_v(\zeta^2+y_v^2)+\dot{x}_vx_vy_v)}{\zeta^2(\zeta^2+y_v^2)}$$

$$\frac{\partial\beta}{\partial x_v} = \frac{2x_vy_v(\ddot{y}y_v-\dot{y}_v^2-\dot{x}_v^2)}{(\zeta^2+y_v^2)^2} + \frac{x_vy_v(\ddot{z}+g)(\zeta^2-y_v^2)}{\zeta(\zeta^2+y_v^2)^2} - \;\ldots$$

$$\ldots\; \frac{2y_v(\dot{x}_vx_v+\dot{y}_vy_v)(\dot{x}_vx_v^2y_v^2+2\dot{x}_vx_v^2\zeta^2+\dot{y}_vx_vy_v^3+2\dot{y}_vx_vy_v\zeta^2+\dot{x}_vy_v^2\zeta^2+\dot{x}_v\zeta^4)}{\zeta^4(\zeta^2+y_v^2)^2}$$

$$\frac{\partial\beta}{\partial y_v} = \frac{2\ddot{y}y_v-\dot{x}_v^2-\dot{y}_v^2}{\zeta^2+y_v^2} + \frac{(\ddot{z}+g)(\zeta^2-y_v^2)}{\zeta(\zeta^2+y_v^2)} - \;\ldots$$

$$\ldots\; \frac{(\dot{x}_vx_v+\dot{y}_vy_v)(2\dot{y}_vy_v^3+2\dot{x}_vx_vy_v^2+3\dot{y}_vy_v\zeta^2+\dot{x}_vx_v\zeta^2)}{\zeta^4(\zeta^2+y_v^2)}$$