pdfauthor=Vi Kumar, pdftitle=Mathworks Minidrone Competition, pdfkeywords=, pdfsubject=, pdfcreator=Emacs 26.3 (Org mode 9.3.6), pdflang=English

Meeting Minutes

Mathworks Minidrone Competition

keyword(LOCATION)
keyword(TIME)

Address:
www.meetingnotes.no
Enterprise Number:

	PRESENT	ABSENT	FOR YOUR INFORMATION
PARTICIPANTS			
Abdullah Sherif - as394@hw.ac.uk	•		
Vishakh Kumar - vpk2@hw.ac.uk	•		
Vishnu Sarathy - vks2@hw.ac.uk	•		
Dr Mehdi Nazarinia			•
Dr Ityonna Amber			•

PROJECT	DATE	CLASSIFICATION
keyword(PROJECT)	2020-05-13	Unrestricted

1 Agenda

Agenda

- Housingkeeping & Onboarding
- Explain basic model of drone control
- Assign tasks

Task List

#	TASK	RESPONSIBLE	DUE DATE

2 Housekeeping & Onboarding

2.1 Git Repository

Get used to git because that's the easiest way for us to work together. Great documentation at https://git-scm.com/doc

Quick reference: git pull origin master git add . git commit -m "Useful message" git push origin master

Our repo: https://gitlab.com/grokkingStuff/MathWorksMiniDrone

2.2 Onramp & Competition Rules

Would recommend reading the rules pdf Finish off the Onramp and put your certificates to the git repo

2.3 Useful Resources

Look these up and

2.3.1 Youtube

Mostly Steve Brunton

1. Singular Value Decomposition https://www.youtube.com/playlist?list=PLMrJAkhIeNNSVjnsviglFoY2nXildDCcv

   Useful for finding the "most important" eigenvalues. Used to find the most controllable regions of the drone

2. Control Bootcamp https://www.youtube.com/playlist?list=PLMrJAkhIeNNR20Mz-VpzgfQs5zrYi085m

   Useful for picking up basics of Control Theory

2.3.2 Research Papers

So I have zero real knowledge about Control Schemes. And that probably reflects in the literature review I've been doing - it's definitely possible and very likely that I'm missing on a bunch of research because I'm not familiar with field terminology

That said, hope these keywords help you out

- Visual-based
- Line following
- Line Following Robot (LFR)

Some papers I found useful were:

1. Eriksen 2014

   Eriksen, Christopher & Ming, Kristina & Dodds, Zachary. (2014). Accessible Aerial Robotics. Journal of Computing Sciences in Colleges. 29. 218-227.

   This work demonstrates some of the computational capabilities of the inexpensive AR.Drone2 quadcopter. Fundamentally a platform that navigates in 3D, the drone offers a clean interface to 2D sensor data through its video stream. By using image-matching to bridge that "interdimensional" gap, this work demonstrates that the drone offers an accessible, compelling platform for research and independent projects at any institution. In the process, we demonstrate that accessible physical platforms, i.e., the drone, the Kinect, and their scaffolding software, ROS, now offer capabilities with much wider curricular and computational applications.

```
@article{article,
author = {Eriksen, Christopher and Ming, Kristina and Dodds, Zachary},
year = {2014},
month = {04},
pages = {218-227},
```

```
title = {Accessible Aerial Robotics},
volume = {29},
journal = {Journal of Computing Sciences in Colleges}
}
```

2. Mur-Artal 2015

   ORB-SLAM2 is a real-time SLAM library for Monocular, Stereo and RGB-D cameras that computes the camera trajectory and a sparse 3D reconstruction (in the stereo and RGB-D case with true scale). It is able to detect loops and relocalize the camera in real time. We provide examples to run the SLAM system in the KITTI dataset as stereo or monocular, in the TUM dataset as RGB-D or monocular, and in the EuRoC dataset as stereo or monocular. We also provide a ROS node to process live monocular, stereo or RGB-D streams. The library can be compiled without ROS. ORB-SLAM2 provides a GUI to change between a SLAM Mode and Localization Mode, see section 9 of this document.

```
@article{murTRO2015,
  title={{ORB-SLAM}: a Versatile and Accurate Monocular {SLAM} System},
  author={Mur-Artal, Ra\'ul, Montiel, J. M. M. and Tard\'os, Juan D.},
  journal={IEEE Transactions on Robotics},
  volume={31},
  number={5},
  pages={1147--1163},
  doi = {10.1109/TRO.2015.2463671},
  year={2015}
 }
```

3 Action items

3.1 Need to get real-life data for stuff

Need to get Dr Mehdi to send over a recording of sensor data from the drone. While we can't actually work with said drone thanks to the whole lockdown, having some raw data should allow us to make (somewhat rational) decisions about which filters and what parameters to use to analyze said data.

Not a priority (yet) and I think Dr Mehdi is busy with exam stuff. Would recommend bugging him about it before our meeting on May 17.

3.2 Need to copy equations of motion from hastily written notes to an actual file

For the UKF because you need a proper statespace model

3.2.1 Need to also add a real simple Simulink implementation of the statespace model you're using. Need some controlability analysis

Mostly so that we know we're not missing out on some really useful info

4 Current Status of Project

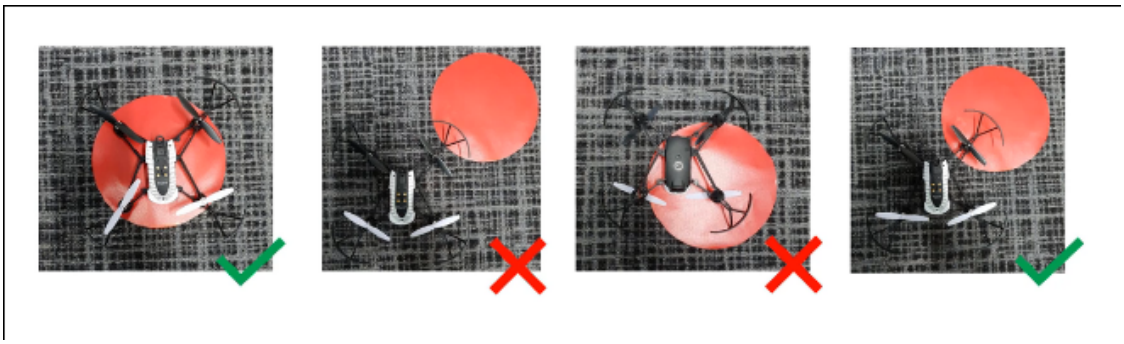
4.1 Mission Timeline

- Take off from a circular pad

- Follow a track laid on floor
  - Track sections have straight lines with no curves
  - Downward facing camera to track lines
- Land on circular end marker

GPS-denied environment and all calculations must be done on-board
Time considered only when the minidrone lands on the end marker. In order for the minidrone to be considered as having landed:

- Minidrone must be upright.
- Minidrone's bottom surface has to touch the floor.



4.2 General Software Stack

Work off ROS inside Simulink.

- Why?

Plenty of existing models in ROS that are industry-tested. Also, it's usually been optimized for the purpose and the ability to just subscribe to a node that gives you information is crazy useful

- So why Simulink?

The control logic is best expressed in Simulink because writing C code that has a decent state machine AND is able to interface with ROS nodes AND is able to help us out with a GUI/debug tools IS HARD! Simulink acts as a glue between our hardware, voodoo blackbox, and our ideas.

4.2.1 Matlab/Simulink

Not gonna write much here because I think everyone is familiar with it

4.2.2 ROS

Robot Operating System (ROS) is a framework of tools, libraries, and software to aid in robot software development. It is a flexible system for programming robots and controlling robotic platforms. ROS was developed by an open-source collaborative community to help grow the world of robotics. Applications for working with hardware, robotic simulation models, path planning, localization and mapping, and many other algorithms are available. For an introduction to ROS, see the ROS Introduction on their website.

1. ROS Toolbox in Simulink

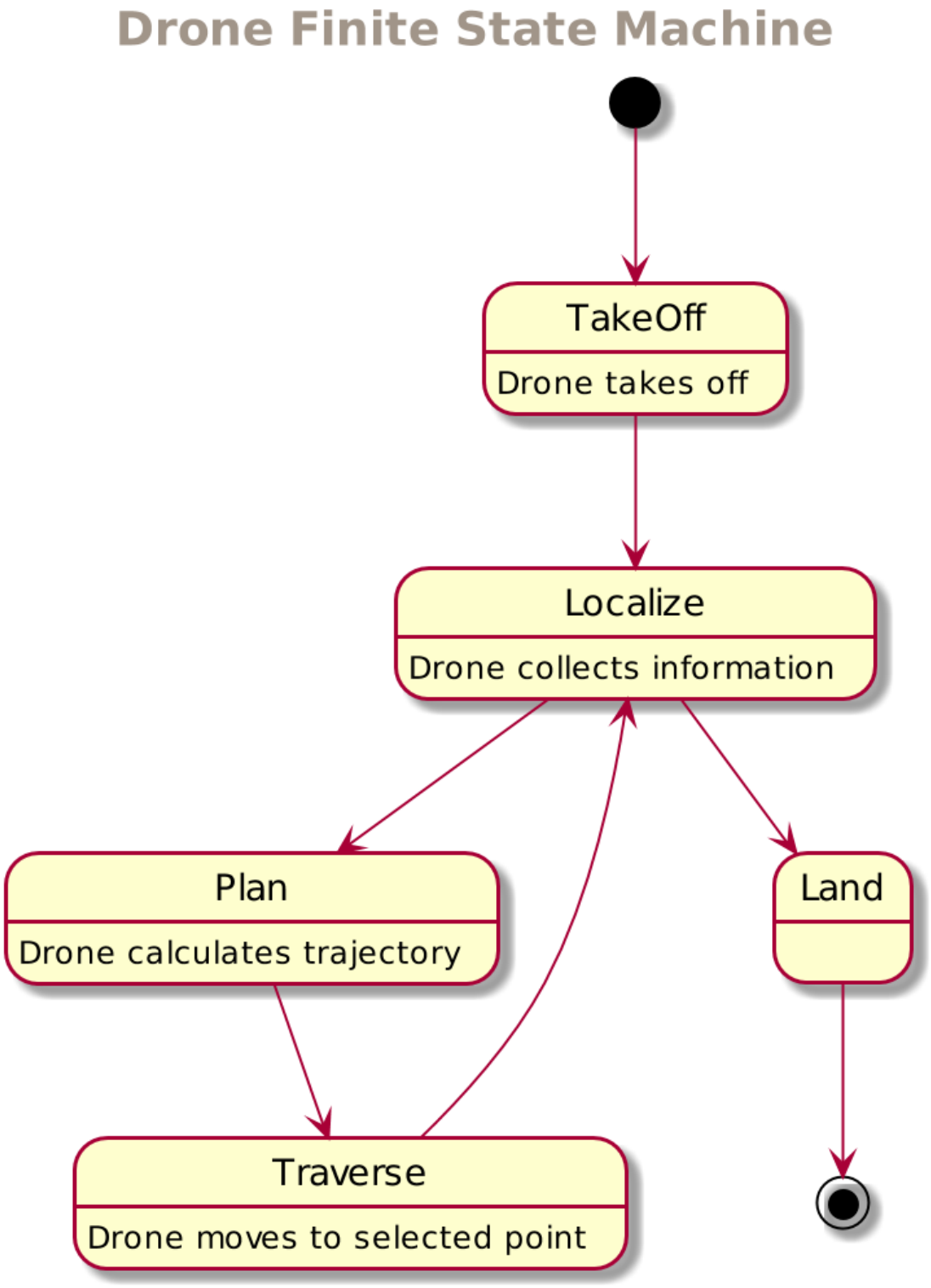
   https://www.mathworks.com/help/ros/gs/robot-operating-system-ros.html

   ROS Toolbox allows you to access ROS functionality in MATLAB® . Use MATLAB to communicate with a ROS network, interactively explore robot capabilities, and visualize sensor data. You can develop robotics applications by exchanging data with ROS-enabled robots and robot simulators such as Gazebo. You can also create Simulink® models that exchange messages with a ROS network. Verify your model within the Simulink environment by receiving messages from, and sending messages to, ROS-enabled robots and robot simulators. From your model, you can also generate C++ code for a standalone ROS application.

   Both MATLAB and Simulink support the TCPROS transport layer (see TCPROS). The UDPROS transport is not supported.

   ROS Toolbox supports ROS Indigo and Hydro platforms, but your own ROS installation may have different message versions. If you would like to overwrite our current message catalog, you can utilize ROS Custom Message Support to generate new message definitions. For ROS 2, ROS Toolbox supports the Dashing Diademata platform.

4.3 Drone Finite State Machine



So our drone needs a way to figure out what to do and how to do it. A really simple Finite State Machine is below. Should probably ask someone who knows what they're doing.

The SLAM algorithm is commputer intensive BUT once localization is done, it's pretty fast.

So our first pass will be super slow to collect info. Then, we can use the state machine to switch between the Localize & Traverse states and make optimal use of information.

4.4 ORB-SLAM2

Github-Repo: https://github.com/raulmur/ORB_SLAM2 Youtube-Example: https://www.youtube.com/watch?v=IuBGKxg

> ORB-SLAM2 is a real-time SLAM library for Monocular, Stereo and RGB-D cameras that computes the camera trajectory and a sparse 3D reconstruction (in the stereo and RGB-D case with true scale). It is able to detect loops and relocalize the camera in real time. We provide examples to run the SLAM system in the KITTI dataset as stereo or monocular, in the TUM dataset as RGB-D or monocular, and in the EuRoC dataset as stereo or monocular. We also provide a ROS node to process live monocular, stereo or RGB-D streams. The library can be compiled without ROS. ORB-SLAM2 provides a GUI to change between a SLAM Mode and Localization Mode, see section 9 of this document.

So why should we use this? We have a static environment that we have multiple passes over. Being able to utilize previous information from the environment (especially if there's some change in lighting) will be rather useful.

Also, it's a fairly plug and play model that takes care of the visual odometry part.

ROS-Wiki Link: http://wiki.ros.org/orb_slam2_ros License: GPLv3

4.4.1 ROS Parameters

There are three types of parameters right now: static- and dynamic ros parameters and camera settings from the config file. The static parameters are send to the ROS parameter server at startup and are not supposed to change. They are set in the launch files which are located at ros/launch. The parameters are:

- $load_{map}$: Bool. If set to true, the node will try to load the map provided with map_{file} at startup.
- map_{file}: String. The name of the file the map is saved at.
- $settings_{file}$: String. The location of config file mentioned above.
- voc_{file}:String. The location of config vocanulary file mentioned above.
- $publish_{pose}$: Bool. If a PoseStamped message should be published. Even if this is false the tf will still be published.
- $publish_{pointcloud}$: Bool. If the pointcloud containing all key points (the map) should be published.
- $pointcloud_{frameid}$: String. The Frame id of the Pointcloud/map.
- $camera_{frameid}$: String. The Frame id of the camera position.

Dynamic parameters can be changed at runtime. Either by updating them directly via the command line or by using $rqt_{reconfigure}$ which is the recommended way. The parameters are:

- $localize_{only}$: Bool. Toggle from/to only localization. The SLAM will then no longer add no new points to the map.
- $reset_{map}$: Bool. Set to true to erase the map and start new. After reset the parameter will automatically update back to false.
- $min_{numkfinmap}$: Int. Number of key frames a map has to have to not get reset after tracking is lost.

Finally, the intrinsic camera calibration parameters along with some hyperparameters can be found in the specific yaml files in orb_{slam2}/config.

4.4.2 ROS Published topics

The following topics are being published and subscribed to by the nodes:

- All nodes publish (given the settings) a PointCloud2 containing all key points of the map.
- Live image from the camera containing the currently found key points and a status text.
- A tf from the pointcloud frame id to the camera frame id (the position).

4.4.3 ROS Subscribed topics

- The mono node subscribes to /camera/$image_{raw}$ for the input image.
- The RGBD node subscribes to /camera/rgb/$image_{raw}$ for the RGB image and
- /camera/$depth_{registered}$/$image_{raw}$ for the depth information.
- The stereo node subscribes to $image_{left}$/$image_{colorrect}$ and
- $image_{right}$/$image_{colorrect}$ for corresponding images.

4.4.4 ROS Services

All nodes offer the possibility to save the map via the service $node_{type}$/$save_{map}$. So the $save_{map}$ services are:

- /$orb_{slam2rgbd}$/$save_{map}$
- /$orb_{slam2mono}$/$save_{map}$
- /$orb_{slam2stereo}$/$save_{map}$

4.5 Unscented Kalman Filter

Like an Extended Kalman Filter but more performant. Able to deal with the drone's non-linearities and should give us a decent idea of where and how fast our drone is moving.

for the accelerometer, gyroscope and stuff This is what keeps the drone actually flying in the air. The ORB-SLAMv2 is really just a way to identify points

5 Background Information

Email: roboticsarena@mathworks.com

Round 1 - Simulation in Simulink Round 2 - Deployment Round on Parrot Mambo Minidrone

5.1 Competition Timeline

Competition launch	03 Feb 2020
Round 1 application closure	06 July 2020, 6 PM BST
Round 1 submission	27 July 2020
Round 1 result declaration	01 Sept 2020
Round 2 live event and winners	01 Oct 2020

5.1.1 Submit initial application

Should be as simple as sending in a list of names.

5.1.2 Round 1 - Simulation in Simulink

Simulink model is submitted.
Submitted models are graded on:

- Accuracy of the traced path
- Time taken to complete track
- Successful landing on end marker
- NOTE Code generation is required.
- NOTE Multiple evaluation passes are done by Mathworks Engineers

5.1.3 Round 2 - Deployment Round

<2020-04-26 Sun> Somewhat uncertain due to the whole coronavirus lockdown. Probably going to be a virtual thing. ~Dr Mehdi
Practice Round

- not evaluated
- two slots of 15 minutes
- Calibrate and test algorithms

Live Round

- one 15-minute slot
- Seven chances to run the hardware
- Nomination of one chance.
- Graded based on number of track sections completed.
- Judges decide which stages are considered complete.

5.2 Software Requirements slide

- Latest version of Matlab
- Optional Simulink OnRamp
- Simulink Package for ParrotDrone
  - Allows you to deploy simulink models to Parrot Drone

6 Parrot Mambo Drone Info slide

6.1 Miscellaneous

6.1.1 Energy

660mAh LiPo Battery 8 min autonomy with accessory connected or bumpers 10 min autonomy with neither accessory nor bumpers 30 min charging time with a 2,1A charger

6.1.2 SDK

SDK: OS Linux. SDK available on Parrot.com We might find documentation useful, especially if the Simulink model neglects to mention something.

6.2 Sensors

6.2.1 Inertial Measurement Unit

Inertial Measurement Unit to evaluate speed, tilt and obstacle contact

- 3-axis accelerometer
- 3-axis gyroscope

Definitely need to get accurate specs for this. The Parrot AR Drone had pretty good IMU chips so pretty sure that even a "low-cost" model should have something with:

- accelerometer +-2g
- Bandwidth ~1000Hz
- Low cross-axis misalignment

Not sure how Mathworks's Simulink package deals with the hardware flags but should be interesting to see.

1. Accelerometer Characterization
    - Bias factor
    - Scale factor
    - Thermal drift (check if it's relevant? Should be a simple correction)

6.2.2 Downward facing camera

60 FPS vertical camera 120x160 pixel resolution Ultrasound sensor

- Do we need to worry about the actual picture being distorted? OpenCV has a little camera calibration thingy that takes care of camera distortion. A chessboard pattern? Something similar here would be sweet.

6.2.3 Ultrasound sensor

useless? Maybe useful in determing distance off the ground and as a way to not have to rely on the downward facing camera

6.2.4 Pressure sensor

Useless!

6.2.5 Streaming Camera

Streaming and Recording HD 720p 30 FPS FOV 120°

6.3 Physical Characteristics

Need to get a MoI matrix from this

6.3.1 Weight

Weight: 2.22 oz / 63g (without bumpers or accessories) Weight with Camera: 73g

6.3.2 Dimensions

7.1 x 7.1 in. / 18 x 18 cm with Bumpers

6.3.3 Rotor Characteristics

Right-hand Coordinate Frame centered at Center of gravity.

Rotor #1 rotates positively with respect to the z-axis. It is located parallel to the xy-plane, -45 degrees from the x-axis.

Rotor #2 rotates negatively with respect to the body's z-axis. It is located parallel to the xy-plane, -135 degrees from the x-axis.

Rotor #3 has the same rotation direction as rotor #1. It is located parallel to the xy-plane, 135 degrees from the x-axis.

Rotor #4 has the same rotation direction as rotor #2. It is located parallel to the xy-plane, 45 degrees from the x-axis.