AE483 Lab Manual: Week #1
Sensor Data Collection and Analysis

T. Bretl

August 25, 2019

1 Goal

You will be working with the AscTec Hummingbird Quadrotor this semester, henceforth referred to as the "drone". There are two sources of sensor data that you can use for control design and analysis:

- an inertial measurement unit (IMU) on-board the drone that measures linear acceleration (actually, "specific force" — we will ignore that detail for now) and angular velocity using accelerometers and gyroscopes;
- a motion capture system (mocap) off-board the drone that measures position and orientation using cameras that track spherical markers.

Your goal during this first week is to collect and visualize sensor data from these two sources during a short manual flight. You will do the following things:

- Program the drone to collect on-board sensor data (Section 2).
- Configure the network so that these on-board sensor data are transmitted wirelessly to the ground station (Section 3).
- Run the ground station so both on-board and off-board data are saved to a text file during your flight (Section 4).
- Load, parse, and plot these data using MATLAB (Section 5).

Section 6 has a summary of in-lab deliverables. **Make sure to save your flight data for later analysis.**

2 Program the drone

Follow the instructions in Appendix A of the lab manual to create an Eclipse project (with the Lab 1 starter code), to compile your code, and to flash your executable to the drone. There is no need to modify the code yet. Show your TA that the flash was successful.

3 Configure the network

Follow the instructions in Appendix B of the lab manual to set the transmission rate to 50 Hz and to test data transmission, using the ACI Tool. Show your TA that all six IMU measurements—three components of acceleration and three components of angular velocity—are being received and are being updated as you pick up and move around the drone.

4 Run the ground station and fly your drone

Follow the instructions in Appendix C of the lab manual to create a Visual Studio 2010 project (with the Lab 1 starter code). Before compiling or running your code, you will need to make the following changes to the file `main.c`:

- Specify the IP address of your PC. The ground station needs to know the IP address of your PC in order to communicate with other devices (e.g., the drone and the motion capture system). This IP address is specified in the `main()` function that is defined in `main.c`. You should find the following code starting at around line 739:

```
    //get local server ip
    if (!GetLocalIPAddresses((unsigned long *)&ServerAddress, 1)) { PrintMessage("
    Failed to get local server ip. Exiting..."); PrintExitPrompt(); WSACleanup();
    return 0; }
    ServerAddress.S_un.S_un_b.s_b1 = 192;
    ServerAddress.S_un.S_un_b.s_b2 = 168;
    ServerAddress.S_un.S_un_b.s_b3 = 1;
    ServerAddress.S_un.S_un_b.s_b4 = 97;

    sprintf_s(szServerIPAddress, MAX_ADDLENGTH, "%d.%d.%d.%d", ServerAddress.S_un.
    S_un_b.s_b1, ServerAddress.S_un.S_un_b.s_b2, ServerAddress.S_un.S_un_b.s_b3,
    ServerAddress.S_un.S_un_b.s_b4);

    //get local client ip
    if (!GetLocalIPAddresses((unsigned long *)&MyAddress, 1)) { PrintMessage("Failed
    to get local client ip. Exiting..."); PrintExitPrompt(); WSACleanup(); return 0; }
    MyAddress.S_un.S_un_b.s_b1 = 192;
    MyAddress.S_un.S_un_b.s_b2 = 168;
    MyAddress.S_un.S_un_b.s_b3 = 1;
    MyAddress.S_un.S_un_b.s_b4 = 97;
```

  Change the numbers for both the "local server ip" and the "local client ip" so that they match what is printed on your PC (literally taped to your PC and labeled "PC-IP").

- Specify the directory to which data should be saved. The ground station will save data to a file. This file will be named `YYYYMMDD-hhmmss`, where "Y" is the year, "M" is the month, "D" is the day, "h" is the hour, "m" is the minute, and "s" is the second at which the ground station started running. The directory where this file will be saved is specified as a `#define` in `main.c`. You should find the following code starting at around line 85:

```
//flight data recorder
#define FDR_DIRPATH             "U:\\documents\\AE483-Fall2019-Lab1-Dev\\"
                                // path to directory where flight data will be saved
```

Change the directory to some appropriate place on your own "`U:`" drive.

Now, continue to follow the instructions in Appendix C of the lab manual to compile your code. Make sure you have read the flight guidelines on the lab website:

http://coecsl.ece.illinois.edu/AE483/AE483_flying_guidelines.pdf

Then, run your executable. Show your TA that the ground station is running and that data are being received. Finally, fly your drone! Please don't hit the ceiling!! Be safe!!! **It is highly recommended that you record video of your flight (e.g., with a smartphone) for comparison with your flight data.** When you are done flying, close the ground station (`CTRL-c`) and show your TA the file with extension "`.csv`" that has data from your flight.

5 Load, parse, and plot data

Do the following things:

- Open the file produced by the ground station in a text editor. You will see that the data in this file are comma-delimited. Each line has a row of data. The first row tells you what is contained in each column (`time`, `accel_x`, and so forth).
- Open and parse this same file in MATLAB, for example using

  ```
  data = readmatrix('somefile.csv')
  ```

  as in

  https://www.mathworks.com/help/matlab/ref/readmatrix.html

  You can, of course, organize and rename the result however you like, perhaps saving it all to a `.mat` file for later analysis.
- Plot the position of the drone (x, y, and z) as a function of time.

Show the plot to your TA when you have finished.

6 Summary of in-lab deliverables

You should have done the following things in lab:

1. Show the TA that you flashed the drone with code downloaded from the course website.
2. Show the TA that you received IMU data through the ACI tool.
3. Show the TA that flight data were saved to a text file.
4. Show the TA that you can plot the position of the drone as a function of time.