MTRN4010. First steps for Project 1

This document is a pre-release of parts of Project 1.
For solving Project 1, we will need to have lectures 1-3 completed.
For solving the items mentioned in this document, lectures 1,2 and 3 and some previous knowledge you have from MATH, are necessary.

In this project we will process sensors' measurement for estimating the 3D attitude of a platform. We will do it in a deterministic fashion. We will need to apply concepts of:

- State space equations (nonlinear cases)
- Discrete time approximation of nonlinear state space equations.
- Linear regression (from Math).
- 3D coordinate transformations.
- 3D attitude.
- Programming (MATLAB plain Programming Language).

The sensors we use are

- 3D gyroscopes (from 3D IMU)
- Depth images from RGB-D camera.

The data we use will be played back, in a way that is equivalent to real-time operation.

We provide an API (Application Program Interface) for controlling the playback sessions, and for certain basic data processing and data visualization functionalities.

In addition, some API functions do implement certain matters that you are required to be implemented by the students (as parts of the project). Those API functions are intended to be used by the students to validate results, comparing theirs with those of those API functions. For that reason, the API is not a clear text M file but a P file (compiled binary version)

The project is composed by nine (9) parts. Parts 0-5 are described in this early release of the document. This document is focussed on describing those parts, which should be solved in weeks 2,3 and early week 4. You should start working on these parts this week (week 2), after the tutorial sessions (in which you will solve the tutorial problems, and will have help for understanding the provided program, "demo01".) You will then inspect "Demo02" in which we use most of those API functions that you may need.

Project 1 has a relevance of 23 marks (out of the 100 of the course)

Partial list of project items (0 to 5) (which should be solved during weeks 2 and 3 and part of week 4.)

0) Play with the example program, to get used to the playback framework, and with the data (3D IMU, RGB camera, Depth camera). The base program already contains parts to visualize data, in "real-time", e.g. via oscilloscopes which do show gyroscopes and accelerometers measurements.
   You are required to add an additional visualization to show the measurements of the 3D magnetometers (0.5 marks).
   Note: You may use the provided API functions, for dynamically plotting those 3 channels, or you may implement your versions, if you prefer. Alternatively, some

toolboxes in MATLAB do offer similar resources (you may search the web to learn about those).

1) Implement the “3D attitude predictor”. For that you will exploit the model by which we integrate the gyroscopes’ measurements (the same equation that is introduced and discussed in Lecture2). The model is a continuous time state equation, so that you will implement its discrete time version, and use it appropriately, for generating estimates of the attitude at the time of each event, in “real-time”, during the playback session.
   You will assume that the initial attitude is Roll=Pitch=Yaw=0 (the platform did have initial roll and pitch well close to those values, since it was resting, static, on the floor).
   In addition to that, you will show the estimated 3D attitude in a separate 3-channel oscilloscope, expressing those attitude components (roll, pitch and yaw) in degrees. (3 marks).

2) Add the capability by which the program does estimate the gyroscopes biases, taking advantage of the fact that the platform is always static during at least the first 5 seconds of its trips (from time=0 to time =5 seconds). During that interval of time (i.e., the “IMU calibration period”) your attitude predictor must not run; it will only run after the calibration process does end. (3.5 marks).
   Note: In the document “RemovingGyrosBiasOFF_LINE_2025.pdf”, you can read about the approach.

3) Modify your implementation of item (1), so that the estimated biases are used for improving the gyroscopes measurements (1 mark).

4) Add some fictitious biases to simulate a lower quality IMU. For example, gyroscope biases of 0.5 degrees/second. Test the performance of (1) and (3) in those cases. This trick will test your implementation, in a difficult case. Use API function ***API.d.PretendIMUBiases(ExtraBias)***.

5) Implement the following capability, a button for triggering the following actions: The user is asked to specify a rectangular Region if interest (ROI) in the current RGB image (*), then the 3D points associated to those selected pixels are obtained (*), and then are used to estimate an approximating plane, in the platform’s CF, and thus its normal vector (expressed in the platform’s CF) will be calculated as well. It is assumed that the user does select a ROI that does fully correspond to part of the floor (which is assumed to be flat and perfectly horizonal in the global coordinate frame (GCF), so that ideally, in the GCF, its normal vector is perfectly vertical, i.e. [0;0;1]; however, in the platform’s CF, that normal vector will depend on the platform’s roll and pitch at that time. You will estimate the normal vector and print its value. Based on that normal vector you will evaluate the ROLL and PITCH angles, at that time.
   For validation of you results you will use an API function (the same one is used in example program Demo02_showData.m). That API function will calculate the normal vector and will estimate the roll and pitch of the platform, and it will print those values. Both results (student’s ones and API calculated ones) will need to have a discrepancy lower than 2 degrees. (4 marks)
   (timing: this item should be solved by the end of week 3, or during early week 4)

   (*): there is an API function for that purpose.

   The lecturer will show typical results during weeks 2 (video) and 3 (Lecture time).

API functions: There are several API functions offered to students to simplify implementation matters. Those are usually used in the example programs. You can also have information about them in file "*API_v04__help.pdf*".
In your program you will use "*API4010_v04.p*" or any posterior version, if that were available.

You may modify and use the example program "*Demo02_ forStudents.m*", if you want.

Additional useful information:

The datasets offered for Project1 are of short duration (1 minute in average), which means that once the gyroscopes are initially calibrated, the estimated biases are valid for the full trip of the platform.

We always expect the estimates of the platform initial and final poses to be very similar, because in those trips the platform's initial and final poses were very similar (but not identical, due to inaccuracies of the platform's control).

The RGB-D camera and the IMU sensor are rigidly attached to the platform. The IMU is well aligned to the platform's chassis, in a way that we consider that the IMU and the platform do have the same coordinate frame (CF) for their measurements.

The camera's Y-axis is aligned to that of the platform's CF. However, the camera is rotated in pitch, so that its X-axis and Z-axis are rotated respect to the platform Y-axis. That rotation is a pure rotation in pitch, in the platform's CF. The value of that angular shift in pitch is close to 18.5 degrees. This must be considered for expressing, in the platform's CF, those 3D points clouds generated by the RGB-D camera. Read the example code, in which that compensation is applied. You will apply the same procedure.

Convention used in our coordinate frames, in this project: If I used a CF for my body, my local CF would be:

+X : my "ahead".
+Y : my left.
+Z : my upwards.

Convention for attitude angles:

φ_x (ROLL), is positive if the rotation is equivalent to +Y rotating towards +Z.

φ_y (PITCH), is positive if the rotation is equivalent to +Z rotating towards +X.

φ_z (YAW), is positive if the rotation is equivalent to +X rotating towards +Y.

Engineering units used in the sensors' data.

Accelerometers	% gravities, "G's". (1G = 9.8 m/s².)
Gyros	% radians/second
Magnetometers=	% micro-teslas

(In this project, we only use gyroscopes' measurements.)

Depth images from RGB-D camera, are expressed in millimetres (mm).

After reading those measurements you may scale them to the units you prefer, for your calculations.

For presentation/visualization of results, when we refer to angles, we will usually require those to be in expressed in degrees. You will do the necessary scaling, if needed, in your program.

Inertial Measurement Unit sensor (IMU). 3D IMU, it provides 9 channels, sampled at 200HZ.

- 3D accelerometers.
- 3D gyroscopes
- 3D magnetometers.

IMU is aligned to the platform, as follows:

- X-axis: +X pointing ahead the platform.
- Y-axis: +Y pointing to the left of the platform.
- Z-axis: +Z pointing up.

RGB-D camera.

- RGB images.
- Depth mages.

We use it in low resolution, 320x240 (320 horizontal, 240 vertical lines).
RGB/colour: RGB24 (3 channels, 8 bits each)
Depth: 16 bits, 1mm precision (but accuracy is usually worse).
The original frame rate of the camera is 30HZ. However, it has been subsampled in time in the datasets, to 5HZ.

We will see these sensors, in class (lectures 2 and 3). Also, we will see them working in real time. Those sensors we see in class, are the same models of those used in the project.

The full specifications of Project 1 will be released early week 3. In that document you will have details about all the items in the project, and about the marking criteria, in addition to the way of submitting your work, deadlines, penalties due to late submission, etc.

Questions? Ask the lecturer, j.guivant@unsw.edu.au

(end of document)