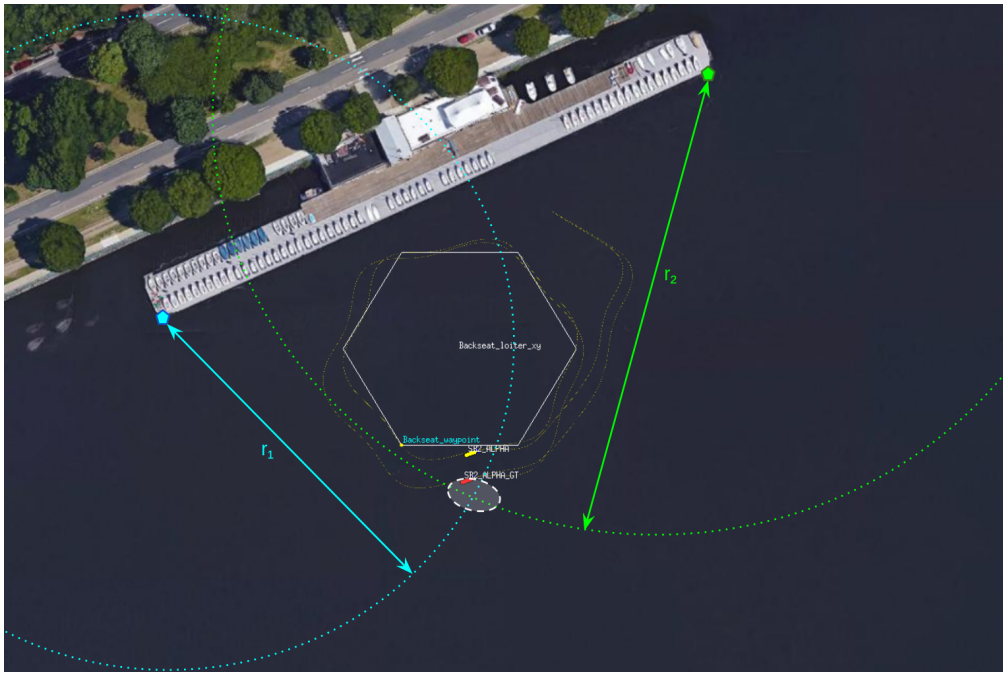
Lab 9b - Underwater Navigation

2.S01 Introduction to Autonomous Underwater Vehicles



Spring 2026

Supun Randeni, supun@mit.edu
Department of Mechanical Engineering
MIT, Cambridge MA 02139

1 Objectives

- Learning how to push and pull code changes into and from github remote server.
- Understanding the importance of virtual ocean for sensor evaluation.
- Understanding the variation of navigation drift with the quality of the AHRS.
- Understanding the concept of navigation-aiding vehicle dynamic models.

2 Updating your git branch with the latest class software features

The instructors frequently update `StackUxV`, `StackUxV-drivers` and `VECTORS` software packages, as well as the `missions-StackUxV` repository. Therefore, you should always begin each lab by updating these software on the topside laptop, Raspberry Pi, and PocketBeagle computers. However, as part of the *Low-level Control Systems of AUVs* lab, you created a personal git branch in `missions-StackUxV`, named after your vehicle (e.g., in Supun's case, a branch named `bee`). We first need to incorporate the latest changes from the `2026_2s01` branch into your own branch, you will need to merge `2026_2s01` into it.

We strongly encourage you to read more about git merging in your own time: https://git-scm.com/book/en/v2/Git-Branching-Basic-Branching-and-Merging

2.1 Ensuring that you have committed all your local changes

Before we merge the `2026_2s01` branch into your own branch, it is good practice to check the status of your branch and make sure there are no uncommitted local changes on your topside laptop:

```
$ git status
On branch bee
Your branch is up to date with 'origin/bee'.

nothing to commit, working tree clean
```

In my case, the `bee` branch does not have any modified or untracked files, so I am ready to merge. However, if your branch contains untracked files or code modifications that have not been committed, you might see something similar to the following:

```
$ git status
On branch bee
Your branch is up to date with 'origin/bee'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        vehicle/bee.def
        vehicle/bee_sensor_config.txt

nothing added to commit but untracked files present (use "git add" to track)
```

In this case you may choose to either destroy these changes using `$ git reset --hard` command or commit these changes (see the *Low-level Control Systems of AUVs* lab for more information).

2.2 Merging the 2026_2s01 branch into your own branch

Before we merge, let's make sure the latest updates to 2026_2s01 have been pulled onto your local topside computer. To do this, switch to that branch using the `$ git checkout 2026_2s01` command:

```
$ git checkout 2026_2s01
Switched to branch ‘2026_2s01’
Your branch is up to date with ‘origin/2026_2s01’.
```

Pull the latest changes to 2026_2s01 from the remote Git server onto your topside laptop using the `$ git pull origin 2026_2s01` command:

```
$ git pull origin 2026_2s01
Warning: the ECDSA host key for ‘github.com’ differs from the key for the IP address ‘140.82.112.4’
Offending key for IP in /home/sb-topside-15/.ssh/known_hosts:4
Matching host key in /home/sb-topside-15/.ssh/known_hosts:60
Are you sure you want to continue connecting (yes/no)? yes
From github.com:supun-randeni/missions-StackUxV
 * branch            2026_2s01      -> FETCH_HEAD
Already up to date.
```

Let's switch back to your own branch (in my case, the branch named bee, using `$ git checkout bee` command). **It is very important that you do this properly. Otherwise, you might merge your own branch into the 2026_2s01 branch.**

```
$ git checkout bee
Switched to branch ‘bee’
Your branch is up to date with ‘origin/bee’.
```

Finally we can merge the changes to 2026_2s01 branch into your own branch using `$ git merge 2026_2s01` command:

```
$ git merge 2026_2s01
```

A terminal window might pop up with nano text editor, asking to the enter a comment about this merge. You may enter a comment and exit by pressing ctrl+x key. Then you should see an output similar to the follows:

```
$ git merge 2026_2s01
Merge made by the ‘recursive’ strategy.
 build_scripts/build_mitfs.sh                     | 145 ++++++++++++++++++++++++++++++++++++++++++++++++++++
 cruise/lab_7_control.def                         |  22 ++++++++
 cruise/lab_7_control_pMITFS_MissionScript.plug |  44 ++++++++++++++++
 3 files changed, 211 insertions(+)
 create mode 100755 build_scripts/build_mitfs.sh
 create mode 100644 cruise/lab_7_control.def
 create mode 100644 cruise/lab_7_control_pMITFS_MissionScript.plug
```

If you and the instructors both have edited the same file (which should not be the case here, but it will likely happen in the future), you will get a warning about a merge conflict. In such situations both parties need to discuss and resolve the merge conflict. In case if you see a merge conflict, please contact the instructors.

2.3 Pushing your updated branch to the git remote server

By merging `2026_2s01` branch into your own branch, you have made some changes to your own branch in the topside computer. We call them local changes. Now you need to push them to the git remote server, so the latest version of your branch (e.g., `bee` branch in my case) can also be pulled on to your Raspberry Pi and PocketBeagle later:

```
$ git push origin bee
Warning: the ECDSA host key for ‘github.com’ differs from the key for the IP address ‘140.82.113.4’
Offending key for IP in /home/sb-topside-15/.ssh/known_hosts:9
Matching host key in /home/sb-topside-15/.ssh/known_hosts:60
Are you sure you want to continue connecting (yes/no)? yes
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 16 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 282 bytes | 282.00 KiB/s, done.
Total 2 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:supun-randeni/missions-StackUxV.git
   15e9fd6..fed544d  bee -> bee
```

3 Updating your computers to the latest software

3.1 Updating `missions-StackUxV`

Now that you have updated your own branch of the `missions-StackUxV` repository and pushed the latest version to the Git remote server, you can pull these changes onto your embedded computers.

As we discussed in previous labs, we created the `update_missions_stackuxv` program to quickly update the current branch of `missions-StackUxV` on both the Raspberry Pi and the PocketBeagle at the same time. When you run this program on the Raspberry Pi, it first updates the Raspberry Pi and then remotely executes the update commands on the PocketBeagle by sending commands over SSH. Please make sure to run this program on the Raspberry Pi, not on the PocketBeagle.

```
$ update_missions_stackuxv
```

3.2 Updating StackUxV, StackUxV-drivers and VECTORS

As we discussed in previous labs, we can update StackUxV, StackUxV-drivers, and VECTORS by updating the StackUxV-DEBIAN-PKG-ARCHIVE repository and re-running the install_packages.sh script. Let's start with the topside computer; cd into the StackUxV-DEBIAN-PKG-ARCHIVE directory on the topside computer:

```
$ cd ~/StackUxV-DEBIAN-PKG-ARCHIVE/
```

First, update the StackUxV-DEBIAN-PKG-ARCHIVE Git repository to obtain the latest packages by using the git pull origin main command. This will pull the latest version of the repository from the Git server (that is, origin) on the main branch:

```
$ git pull origin main
From github.com:supun-randeni/StackUxV-DEBIAN-PKG-ARCHIVE
 * branch            main       -> FETCH_HEAD
Already up to date.
```

Now you can use the install_packages.sh script to re-install the latest versions of StackUxV, StackUxV-drivers and VECTORS.

Now you can repeat the same steps to update StackUxV, StackUxV-drivers and VECTORS on the Raspberry Pi and PocketBeagle in your training-kit.

4 Exercise: IMU quality vs. navigation drift

In the *Underwater Navigation* lecture, we discussed how the quality of navigation sensors can impact navigation accuracy. Our virtual ocean environment, VECTORS, allows us to simulate various sensor dynamics and observe their effects on navigation accuracy and overall autonomy. In this exercise, we will modify the dynamics of the AHRS/IMU and observe their effects in software-in-the-loop simulations.

4.1 Test the lab_9_navigation cruise

Before we make changes to the IMU sensor dynamics, let us first verify that you can run the lab_9_navigation cruise designed for this lab. On the topside computer:

1. Configure the architecture to seabeaver_iii.
2. Configure the vehicle to your AUV (i.e. the definition that you just created).
3. Configure the cruise to lab_9_navigation.
4. Launch the mission in simulation. You may choose to run the simulation at higher time warps using $ ./launch_simulation.sh 5

You will see a mission similar to the one shown in Figure 1. The vehicle with the name suffix _GT represents the ground-truth position of the simulated vehicle. The other vehicle shows the computed navigation solution; i.e., where the AUV believes it is. Note how the autonomy helm makes decisions based on this estimated position.

Figure 1: The lab_6_navigation mission

4.2 Change the quality of the IMU and observe the rate of navigation drift

The IMU/AHRS simulation MOOS application in the VECTORS virtual environment is called uSimVECTORS_IMU. The configuration plug for this application is called uSimVECTORS_IMU.plug, and is located in the directory:

```
missions-StackUxV/architecture/seabeaver_iii/architecture_plugs/virtual_ocean_plugs/uSimVECTORS_IMU.plug
```

Open this file with your preferred text editor and examine the various configuration parameters:

```
//--------------------------------------------
// uSimVECTORS_IMU config block

ProcessConfig = uSimVECTORS_IMU_$(VEHICLE)
{
   AppTick   = 4
   CommsTick = 4

   platform_id      = $(VEHICLE_ID)
   sensor_id        = $(SENSOR_ID_XSENS_AHRS)

   #ifdef IMU_ERROR_MODEL_SCALE
     imu_scale_error    = $(IMU_ERROR_MODEL_SCALE)
   #else
     imu_scale_error    = 0.0
   #endif

   #ifdef IMU_ERROR_MODEL_RANDOM
     imu_random_error   = $(IMU_ERROR_MODEL_RANDOM)
   #else
     imu_random_error   = 0.5
   #endif

   #ifdef IMU_ERROR_MODEL_PITCH_BIAS
     pitch_bias_error   = $(IMU_ERROR_MODEL_PITCH_BIAS)
   #else
     pitch_bias_error   = 0.005
   #endif

   #ifdef IMU_ERROR_MODEL_ROLL_BIAS
     roll_bias_error    = $(IMU_ERROR_MODEL_ROLL_BIAS)
   #else
     roll_bias_error    = 0.005
   #endif


   #ifdef IMU_ERROR_MODEL_HEADING_SCALE
     heading_scale_error = $(IMU_ERROR_MODEL_HEADING_SCALE)
   #else
     heading_scale_error = 0.0
   #endif

   #ifdef IMU_ERROR_MODEL_HEADING_RANDOM
     heading_random_error = $(IMU_ERROR_MODEL_HEADING_RANDOM)
   #else
     heading_random_error = 2.0
   #endif

   #ifdef IMU_ERROR_MODEL_HEADING_BIAS
     heading_bias_error   = $(IMU_ERROR_MODEL_HEADING_BIAS)
   #else
     heading_bias_error   = 0.01
   #endif

   // imu_scale_error      -> Roll & pitch scale error
   // imu_random_error     -> Roll & pitch random error
   // pitch_bias_error     -> Pitch bias error
   // roll_bias_error      -> Roll bias error
   // heading_scale_error  -> Heading scale error
   // heading_bias_error   -> Heading bias error
   // heading_random_error -> Heading random error
}
```

The sensor dynamics for heading measurements are modeled using the parameters:

- heading_scale_error
- heading_bias_error
- heading_random_error

The sensor dynamics for roll and pitch measurements are modeled using the parameters:

- imu_scale_error
- imu_random_error
- pitch_bias_error
- roll_bias_error

Let's play with the sensor dynamics for heading measurements first since it is more visible in pMarineViewer.

4.2.1 Heading bias

Try the following heading bias values, one at a time, and observe the navigation drift:

1. heading_bias_error = 20
2. heading_bias_error = 10
3. heading_bias_error = 5

When the high heading bias value is large, you will see a large navigation drift as shown in Figure 2.

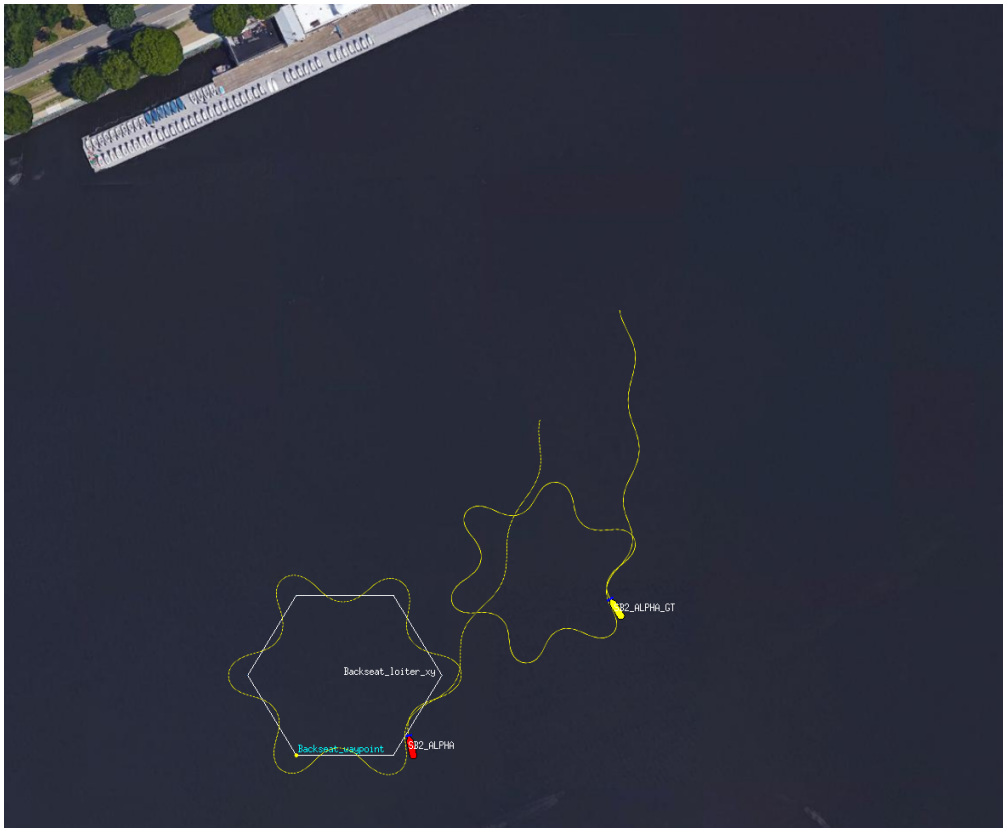


Figure 2: The `lab_6_navigation` mission with a high heading sensor bias

4.2.2 Heading random error

Try the following heading random error values, one at a time, and observe the response:

1. `heading_random_error = 10`
2. `heading_bias_error = 5`
3. `heading_bias_error = 2`

4.2.3 Sensor dynamics for roll and pitch measurements

Similarly, you can also change the `pitch_bias_error` and `imu_random_error` values and observe the response.

4.3 Destroy local code changes

At this point, we have made some temporary changes to the `missions-StackUxV` sensor model configurations, that we do not wish to commit. We can see those changes using `$git status` command:

```
$ git status
On branch bee
Your branch is up to date with 'origin/bee'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   ../plugs/vectors_plugs/uSimVECTORS_IMU.plug

no changes added to commit (use "git add" and/or "git commit -a")
```

We can destroy those changes by using the command `$ git reset --hard`:

```
$ git reset --hard
HEAD is now at 401de4b Changing the name of the navigation lab
```

5 Instructor Check-off Assessment

In the *Underwater Navigation* lecture, we discussed various velocity-aiding navigation sensors. In this class, our SeaBeaver AUVs use a simple vehicle navigation model to predict surge velocity based on the propeller speed as given in the equation below:

$$\textit{Surge speed} = \texttt{prop\_percent\_sq}\ x\ \textit{Propeller Speed}^2 + \texttt{prop\_percent}\ x\ \textit{Propeller Speed}$$

where, `prop_percent_sq` and `prop_percent` are vehicle dependant parameters configured in the vehicle definition file; e.g. in `missions-StackUxV/vehicle/bee.def`:

```
// ********************************************************************************
// HydroMAN navigation related
// ********************************************************************************
#define HYDROMAN_MODEL_SURGE    hydroman_surge_model: prop_percent_sq=0, prop_percent=0.045
```

5.1 Optimizing the vehicle navigation model in software-in-the-loop simulations

In this exercise, we will try to optimize these parameters (i.e. `prop_percent_sq` and `prop_percent`) within the virtual ocean by conducting software-in-the-loop simulations using a trial-and-error approach.

Edit your vehicle definition file (e.g. `missions-StackUxV/vehicle/bee.def`, in my case) and try different values for `prop_percent_sq` and `prop_percent` and re-run simulations to see if your improved model reduces the navigation drift. For example, you can try values such as:

```
// ********************************************************************************
// HydroMAN navigation related
// ********************************************************************************
#define HYDROMAN_MODEL_SURGE    hydroman_surge_model: prop_percent_sq=0, prop_percent=0.02
```

By doing this, you are trying to optimize the vehicle navigation model for the simulated vehicle's

dynamics.

5.2 Pushing your optimized vehicle navigation model to the git remote server

Once you are satisfied with the performance of your improved vehicle navigation model, we can conduct a hardware-in-the-loop simulation using this updated model. So let's commit these changes and push them to your branch in the git remote server. If you do not remember how to do this, please follow the instruction given in the previous labs.

5.3 Testing your optimized vehicle navigation model in hardware-in-the-loop simulations

Once you pushed the changes to your branch in the git remote server, pull these updates to the RaspBerry Pi and PocketBeagle. Now you can run the `lab_9_navigation` mission with your improved navigation model in hardware-in-the-loop simulation style.