1 Previous project work

This report is in a way a continuation of a specialization project[?] done previously. The specialization project was focused on setting up a framework to work with. Part of hte specialization project has been starting work on a simulator that can be used to work on a control system. The control system is currently rudimentary implemented but there is room for improvement. The goal of the development for the conttrol sytemis to have it be platform agnostic. That is, the control system does not care whether tis' connected to a simulator or a vessel in hte real world. TThe way this is ideally implemented is by using ROS2. ROS2 is a framework to control robotics, built around publishers and subscribers that all act around topics.

2 State of the Art

2.1 Physical issues

2.2 Simulation issues

3 Mathematical basis

4 Modelling and Control Design

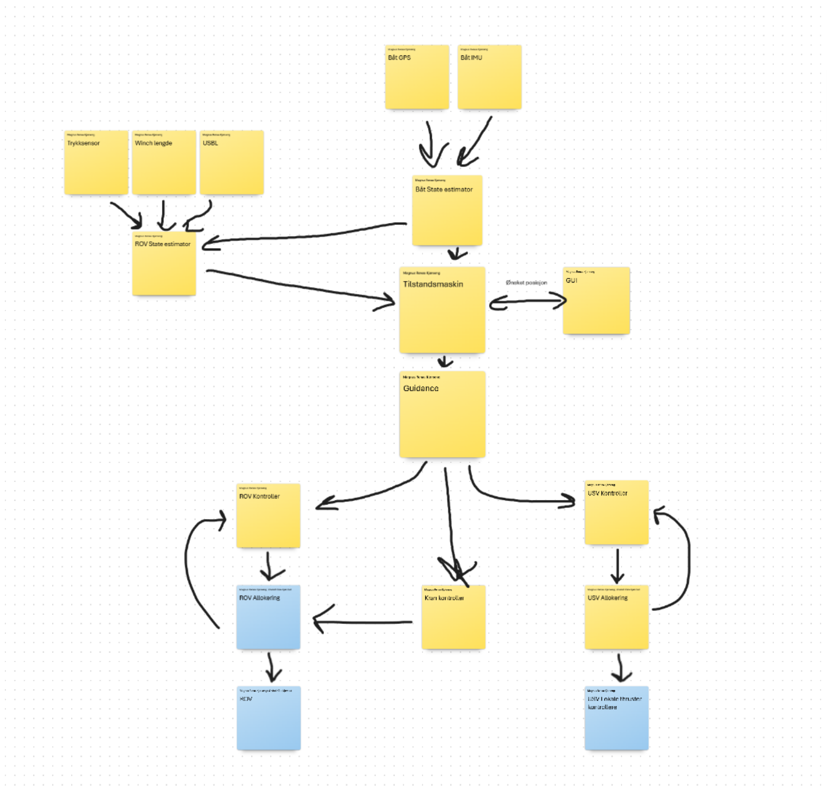

Figure 1: A sketch of the control system design. TODO: Fiks bedre illustrasjon

5 Sensorics

The first step to knowing where to go is knowing where you are.

Alternatives for underwater navigation

5.1 GNSS

Only used above water

5.2 Taut-wire positioning

Mention taut wire, how it is usable somewhat btu not really because the ROV is not heavy enough to have the wire taut

5.3 USBL

Available and accurate

Need to consider limitations of accuracy of USBL re: salinity, water temperature/density, etc.

6 State estimation

The state estimator takes in the various sensor data and builds a single model of position. This is done because different sensors might have different accuracies or update times. The state estimator handles these discrepancies and builds a cohesive model. The state estimator feeds this more accurate position into the guidance system (TODO: Spørre øivind om det er guidance eller state machine som får posijonen. State machine skal jo bare være et informasjonslager?)

7 State machine

The state machine keeps track of variables for the total control system. These include current position, but also things like desired position or operating mode, gathered from the GUI.

8 GUI

The GUI is where a human operator interfaces with the control system. The GUI is supposed to include a position input for the surface vessel, mode switching between USV-Master, ROV-Master and idle/standby modes, along with other functions.

9 Guidance

The guidance system provides finer control of the vessel than what would be achieved by a state machine and controller alone. For instance it smooths acceleration/deceleration of the vessels by providing imaginary set points between the current position and the actual set point.

10 Controller

The controller finds a desired force input based on the difference (error) between the current position and the desired position. The current implementation uses a simple PID for this. This force input is fed forward to the allocator.

The shape of the force coming out of the controller is as

$$\tau = \begin{bmatrix} X \\ Y \\ N \end{bmatrix}$$

11 Allocation

The allocator works like a translation layer between the controller and the local controllers. The controller provides a force input on the vessel's center of gravity. By inputting forces on the center of gravity, no torques are produced from lateral forces, nor lateral forces from the torque.

In abstract terms, the allocator finds and applies a transformation matrix T such that

$$Tf = \tau$$

where f is a vector of vectors with the lateral forces for each thruster. For this case with two thrusters, it will look something like

$$f = \begin{bmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \end{bmatrix}$$

The transformation matrix can be written explicitly for the USV, since it has a very simple thrust configuration. For larger configurations it might be better to write each thruster's transformation individually and either add or remove them depending on the type of move necessary (larger moves use only larger thrusters etc.).

$$T = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ -l_{y_1} & l_{x_1} & -l_{y_2} & l_{x_2} \end{bmatrix} \tag{1}$$

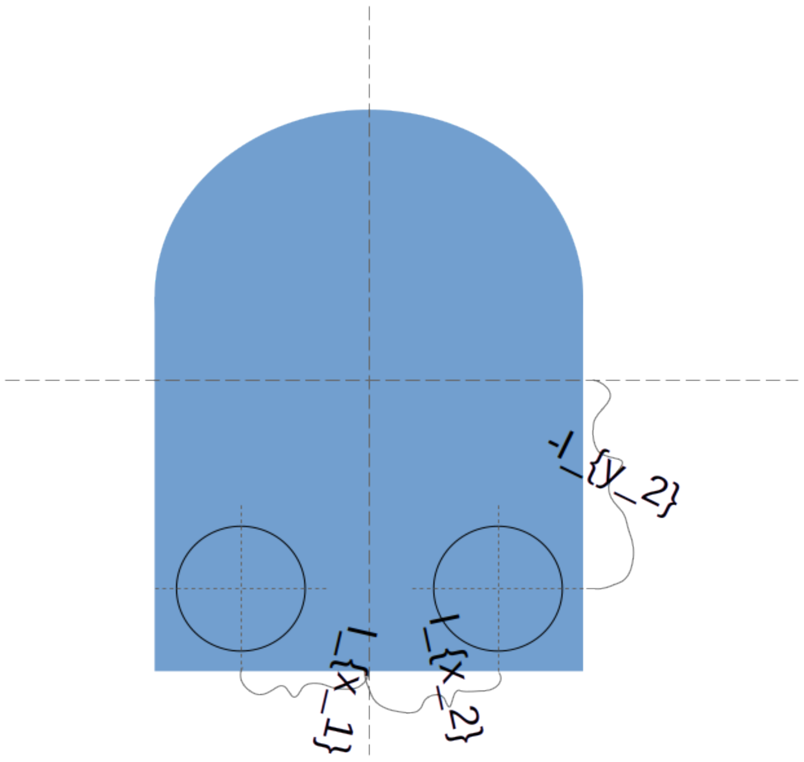

Figure 2: Sketch of how the thruster position values in eq. (1) are found

Since τ and T are known, we can find f by performing a pseudoinverse on T leading us to the equation

$$f = T^{\dagger}\tau \tag{2}$$

where $T^{\dagger}$ is the pseudoinverse of T.

This can also be written longform as

$$\begin{bmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \end{bmatrix} = T^{\dagger} \begin{bmatrix} X \\ Y \\ N \end{bmatrix} \tag{3}$$

The full matrix for $T^{\dagger}$ is omitted because the pseudoinverse of a non-square matrix tends to be large and ugly. It will only be handled by machine hands anyway, and as such doesn't matter right now.

The ROV has a built-in allocator which works well enough. The only issue with the ROV's allocator is that it's configured for a neutrally buoyant vessel. For this system we need to filter the vertical force component so the ROV only handles high-frequent/small-amplitude responses and the crane handles larger amplitudes and lower frequencies. This is also necessary because of elasticity in the lifting cable.

12 Local control and physical response

The vessel in this iteration has two thrusters. The ROV has a closed working solution and will not be considered here except for in the hypothetical. Each of these local controllers receive a two-component vector (or three-component in the case of the ROV) which instructs the controller what the desired thrust is. The azimuth thrusters on the USV are able to apply a force in one direction (parallel to the propeller axis), but they are able to vector this one-dimensional thrust using the azimuth ring.