Naiad AUV motion control system

Emil Bergström
ebm09003@student.mdh.se

ABSTRACT

This report will cover the motion control system for the Naiad AUV platform developed during the fall of 2013 at Mälardalens University. The Naiad AUV is an autonomous underwater vehicle developed as a research platform for the RALF III research project at Mälardalens University. The purpose of the Naiad AUV is to be able to locate and salvage toxic waste in the baltic sea. The motion control system for the Naiad AUV has its basis in six PID controllers, one for each component of the crafts current orientation and position. These will produce control values for each of the six thrusters that propels the craft. This system is built as a layered software system where modularity and maintainability is in focus. The result is a fully functional control system for the Naiad AUV that is very reliant on calibration of each individual PID controller.

Keywords

PID control, motion control, autonomous underwater vehicle, Naiad AUV, component based software engineering

1. INTRODUCTION

This report will cover the motion control system for the Naiad AUV research platform. Then Naiad AUV was developed during a robotics project at Mälardalens University autumn of 2013. The goal of the project was to build an autonomous underwater vehicle for entering the SAUC-E and RoboSub competitions during the summer of 2014. The long term goal of the Naiad AUV was to provide the RALF III research project at Mälardalens university with a platform to base its research on and hopefully in the future be able to build a shoal of AUVs to clean up industrial waste in the Baltic sea.

In this report we will go over the reasoning and the implementation of the motion control system for the Naiad AUV.

1.1 The craft

The Naiad AUV is a small lightweight autonomous underwater vehicle where modularity and ease of modification are key features. Both Naiads hardware and software is built as a distributed system with its basis in one single CAN-bus. This gives the platform the ability to add, remove and replace nodes on the bus to change certain aspects of the craft. One of the key features of the platform is to build it small, this will give the users the opportunity to use several crafts in a swarm to cover a larger area in a short time span without a huge investment if needed.

1.2 Motion control not navigation

The motion control system of the Naiad AUV is developed with the purpose of giving the Naiad AUV the ability to move from position and orientation A to position and orientation B in a suitable manner. The motion control system should not be mixed up with navigation and path planning. It provides the mission control system with a movement service.

The motion control system of the Naiad AUV is a layered software system which has its basis in six PID controllers. Each component of the current location (orientation and position) of the craft is controlled by a separate PID controller.

All PID based calculations in the system are based on an optimal thruster configuration where each thruster only affects one component.

A conversion from the optimal thruster configuration to the real system is done through a matrix multiplication with the inverse of a matrix that holds the effects of each thruster on each component, this matrix can be seen in figure 2.

With this approach the flow of data in the system is easy to follow. It provides the Naiad AUV with a reliable motion control system that executes its task flawlessly with proper configuration.

This report is organized as follows. In section 2 the implementation of the system in question is explained. In section 3 the results are shown and in section 4 the authors conclusions are given.

2. IMPLEMENTATION

The motion control system for the naiad AUV is a layered system where the location of the craft is successively broken down into its components. It is borken down from location to one component for each positional and orientation part, x, y, z, rotation around x, rotation around y and rotation around z.

Each of these parts are controlled by its own PID controller.

2.1 Error calculations

PID control bases its control loop on the current error in the system, the difference between the current state of the system and a wanted state of the system. This implementation must be able to calculate both the error in position and orientation.

The positional errors are very straight forward. Since the position of the craft is represented by a vector with three components it is simply the difference between the current value, x, y, z, of each component and the wanted value, x̂, ŷ, ẑ, of each component as seen in (1) (2) (3)

The error in orientation is a bit more complex. To be able to calculate the error in orientation one needs to move the wanted and current orientation to local space. Since the orientation of the craft is represented with an orientation matrix, where each column is one of the axis vectors as seen in figure 4. one can calculate the relative orientation of the craft compared to the wanted orientation by multiplying the inverse of the current orientation and the wanted orientation as seen in figure 5. This puts both orientations in local space, with the current orientation being the identity matrix.

From that, the plane containing both axis vectors that are perpendicular to the axis vector of interest can be utilized, i.e for rotation around z, the plane contains the x and y axis vector. A plane containing the same axes in the wanted orientation is also utilized.

The full orientation of the craft can be represented by a change in the plane of the craft and the direction of the craft in said plane. With these two planes the rotation matrix needed for moving from one plane to the other can be retrieved. The only difference between this new orientation matrix and the original wanted orientation matrix is the angular difference around the axis vector of interest.

With this in mind a vector in the original plane can be rotated with the help of the new rotation matrix and a duplicate vector with the wanted orientation matrix. The angular difference between these two vectors is the angular error for said component.

2.2 Transformation to a usable value

The control value calculated for each component does not take into account each thrusters effect in the current orientation of the craft since its a single value.

After the control values of each component has been calculated, a control configuration is obtained that corresponds to a craft that has one thruster that only creates force along its component. i.e one thruster that only gives rotation around x and one thruster that only gives translation along x, lets call this configuration the optimal thruster configuration.

$$error(x) = \hat{x} - x \tag{1}$$

$$error(y) = \hat{y} - y \tag{2}$$

$$error(z) = \hat{z} - z \tag{3}$$

Figure 1: Positional error calculations

This configuration needs to be converted into a configuration that takes the current orientation of the craft into account, since the effect of a thruster and which axes it affects is dependent on the current orientation of the craft in question.

A matrix of the relative effects of each thruster on all the different components is created. This matrix can be seen in Figure 2. With this matrix the thruster values from the optimal thruster configuration can be converted to a real thruster configuration. The inverse of this thruster configuration matrix is calculated and the control value configuration created for the optimal thruster configuration is multiplied with this matrix. This gives the real control value configuration needed for the craft to move closer to the wanted position and orientation of the craft.

Since the position and orientation of the craft is calculated separately from each other the full range from -100% to 100% of each thruster is available to both the calculations for orientation and the positional calculations. After each part has calculated its control configuration these sets are just added to each other before they are transferred from the optimal configuration to the real configuration. Since the values that are supplied to the thruster controllers one can not have values outside the -100% to 100% these values need to be scaled within the range.

This is done in two steps. The first is to remove the control values for orientation and only try to preform the transition from optimal to real configuration only on the positional control configuration. If this configuration still yields a configuration that has values outside of the range of the thruster controllers then one scales this configuration based on the component that has moved furthest outside of its range. A ratio between that components control value and the range of thruster controllers is calculated and that ratio is multiplied with each component to keep the ratio between each thruster but move each component within the range.

3. RESULT

The current implementation yielded a system that is very reliant on calibration as with all PID based control systems. Without proper calibration and configuration of the different PID controllers the system will not behave in a desired way.

The Zeiger-Nichols method [1] did not prove to be a reliable tuning method for this system, manual tuning with trial and error as a strategy was used to find a usable configuration for each component in the system.

Thr	x	y	z	x^o	y^o	z^o
1	-0.866025	0.5	0	-0.0205	-0.35507	0.248963
2	0	-1	0	0.035	0	0.338
3	0.866025	0.5	0	-0.0205	0.035507	0.292265
4	0	0	1	0.111	-0.095	0
5	0	0	1	-0.025	0.476	0
6	0	0	1	-0.161	-0.095	0

Figure 2: Thruster configuration matrix. These values are measured from the solid works model. Index explanation can be seen in figure 3.

Thruster	Position
1	Horizontal front right
2	Horizontal back
3	Horizontal front left
4	Vertical front left
5	Vertical front right
6	Vertical back

Figure 3: Thruster indices

$$O = \begin{bmatrix} Ux & Vx & Wx \\ Uy & Vy & Wy \\ Uz & Vz & Wz \end{bmatrix}$$

Figure 4: Representation of an orientation matrix. Where each column is a axis vector.

$$c = \text{relative current orientation matrix}$$
$$w = \text{relative wanted orientation matrix}$$
$$C = \text{absolute current orientation matrix}$$
$$W = \text{absolute wanted orientation matrix}$$
$$I = \text{identity matrix}$$

$$c = M^[- 1] * M = I \tag{4}$$

$$w = M^[- 1] * W \tag{5}$$

Figure 5: Calculation of relative current and wanted orientation

Since all the tests for this system was done in a simulated enviroment it is not certain if the current configuration will suffice in a real enviroment. However with the tests done in the simulated enviroment a proof of concept for the structure and implementation could be presented.

4. CONCLUSION

With this implementation a stable and reliable motion control system was creaded.

With the current structure it is according to the authors, easy to understand the codebase and to follow the flow of data through the system. With the successive breakdown of the location of the craft into its components it is easy to understand how the system performs its calculations and retreives the next sequence of control values to the thruster controllers.

With the component based class structure a maintainable system with extensive unit testing has been written. And with this in mind the authors can confidently say that they have created a stable and reliable system that fulfills its purpose.

With correct calibration one can place the craft in any orientation and move towards the current setpoint. This is according to the authors very good since all six degrees of freedom can be used. If the craft got weighted down in some way it would compensate for that so transporting objects without loosing control of the craft is possible.

5. REFERENCES

[1] Optimum settings for automatic controllers. `http://chem.engr.utc.edu/Student-files/x2008-Fa/435-Blue/1942-paper.pdf`. Accessed 2014-01-15.