TRABALHO DE GRADUAÇÃO

SISTEMA DE CONTROLE E MONITORAMENTO DE UM ROV

Ícaro Alves de Melo

Jéssica Silva Favilla

Brasília, Dezembro de 2015

ENGENHARIA
MECATRÔNICA
UNIVERSIDADE DE BRASÍLIA

UNIVERSIDADE DE BRASILIA
Faculdade de Tecnologia
Curso de Graduação em Engenharia de Controle e Automação

TRABALHO DE GRADUAÇÃO

SISTEMA DE CONTROLE E MONITORAMENTO DE UM ROV

Ícaro Alves de Melo

Jéssica Silva Favilla

Relatório submetido como requisito parcial de obtenção de grau de Engenheiro de Controle e Automação

Banca Examinadora

Prof. Eugênio F. L. Fortaleza, ENM/UnB _____
Orientador

Prof. Eduardo Stockler Tognetti, ENE/UnB _____
Examinador interno

Prof. Carlos Humberto L. Quintero, ENM/UnB _____
Examinador interno

Brasília, Dezembro de 2015

FICHA CATALOGRÁFICA

MELO, ÍCARO ALVES e FAVILLA, JÉSSICA SILVA
Sistema de Controle e Monitoramento de um ROV,

[Distrito Federal] 2015.

x, 74p., 297 mm (FT/UnB, Engenheiro, Controle e Automação, 2015). Trabalho de Graduação – Universidade de Brasília. Faculdade de Tecnologia.

1. Veículo Operado Remotamente
2. Processamento de Imagem
3. Controle Linear
4. Modelagem

I. Mecatrônica/FT/UnB
II. Título (Série)

REFERÊNCIA BIBLIOGRÁFICA

MELO, Í. A., FAVILLA, J. S., (2015).Sistema de Controle e Monitoramento de um ROV, Publicação FT.TG-$n°$017/2015, Faculdade de Tecnologia, Universidade de Brasília, Brasília, DF, 74p.

CESSÃO DE DIREITOS

AUTOR: Ícaro Alves de Melo e Jéssica Silva Favilla

TÍTULO DO TRABALHO DE GRADUAÇÃO: Sistema de Controle e Monitoramento de um ROV

GRAU: Engenheiro ANO: 2015

Ícaro Alves de Melo

Jéssica Silva Favilla

Dedicatórias

À minha família
Júnior, Mônica, Hamilton e Juliana

Jéssica Silva Favilla

À minha família
Alaeste, Arlinda, Luana e Adriene.

Ícaro Alves de Melo

Agradecimentos

Agradeço a minha família, que sempre esteve ao meu lado e me apoiou de todas as maneiras. Aos meus amigos, que sempre me deram força nos momentos difíceis. Ao Lucas Fujita e sua família, a Verinha e a Tia Iara, que nos emprestaram suas casas e piscinas, sem vocês nada desse estudo teria sido realizado. A minha parceira de trabalho e de vida, que sempre me fez ser uma pessoa e um estudante melhor. A Anna Carolina, uma amiga, que sempre nos ajudou em tudo que era necessário e nos ensinou os atalhos da UnB.

Ícaro Alves de Melo

Agradeço primeiramente a Deus, por ter me dado força e conforto nos momentos difíceis e por ter me dado a vida.
Em segundo, agradeço aos meus pais e irmãos que sempre me motivaram a acreditar em mim mesma e nos meus sonhos, mesmo quando nem eu mesma acreditava mais.
Agradeço a minha tia Iara, minha prima Verinha e meu amigo Lucas Fujita e seus familiares, por emprestarem as suas piscinas, pois sem elas não teríamos realizado nenhum dos testes necessários e nos desculpem pelos incômodos.
Aos meus amigos, meu agradecimento por sempre me ajudarem de alguma forma para que esse sonho se realizasse, sem vocês essa longa caminhada não teria terminado. Obrigado pelos risos, abraços, companheirismo e conselhos. Agradeço também, minha Tia Jussara e minha vó Laura, por me emprestarem o conforto do seu lar. Aos meus familiares, agradeço por sempre me apoiarem.
Finalmente, agradeço ao meu parceiro de trabalho e de vida Ícaro Alves de Melo que sempre me ajudou quando precisei. Seu carinho, amor, amizade, companheirismo e inteligência foram essenciais para enfrentar todo os desafios do dia à dia.

Jéssica Silva Favilla

RESUMO

O avanço da tecnologia em campos de petróleo impulsionou o desenvolvimento de sistemas autônomos, devido as condições de exploração serem cada vez mais inóspitas para o ser humano. O ROV, do inglês, *Remotely Operated Vehicle*, é um veículo subaquático controlado remotamente por um operador em uma estação terrestre ou embarcação. A ligação entre o veículo e o operador é assegurada por um cabo umbilical que permite a comunicação bidirecional, assim como o transporte de energia para o veículo. Neste trabalho, descreve-se o projeto da geometria de um veículo operado remotamente e desenvolve-se sua modelagem com o objetivo de se implementar técnicas de controle para que o mesmo se torne autônomo. Por meio de um modelo comprado da *VideoRay*, os controladores foram implementados a nível experimental. O desenvolvimento de um controle PID linear com base em teorias, experimentos e processamento de imagem permitiu o controle da regulação da distância de um objeto até o robô, assim como o deslocamento lateral do objeto frente ao ROV. Para que isso seja possível, é utilizado um programa base em Csharp, onde são feitas implementações e o processamento de imagem.

Palavras-Chaves: veículo operado remotamente, modelagem, controle linear, processamento de imagem.

ABSTRACT

The advance of the technology of the oil field is powering the development of autonomous systems, due to exploration conditions being more and more inhospitable to the human being. The ROV (Remoted Operated Vehicle) is operated by a person on a terrestrial station or a vessel. The linkage between the vehicle and the operator is secured by an umbilical cable which allows a bidirectional comunication, as well as the energy source to the vehicle. On this paper, it is decribed and geometry project of an remoted operated vehicle and is developed its modeling with the goal to implement control techniques so it turns into a autonomous vehicle. With a bought model of VideoRay, the controlers were implemented in a experimental level. The development of a Linear PID Control with embasement on theorys, experiments and image processing allows the control of the regulation of the distance of an object till the robot, as well as the control of the regulation of the lateral displacement of an object in front of the ROV. In order to be possible, it is used one base program in CSharp, where the implementation and the image processing is deed.

Key-Words: remoted operated vehicle, modeling, linear control, image processing.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SÍMBOLOS

Símbolos Latinos

x	Componente do vetor de posição na direção *Surge*
y	Componente do vetor de posição na direção *Sway*
z	Componente do vetor de posição na direção *Heave*
u	Componente do vetor de velocidade linear na direção *Surge*
v	Componente do vetor de velocidade linear na direção *Sway*
w	Componente do vetor de velocidade linear na direção *Heave*
p	Componente do vetor de velocidade angular na direção *Roll*
q	Componente do vetor de velocidade angular na direção *Pitch*
r	Componente do vetor de velocidade angular na direção *Yaw*
X	Força na direção *Surge*
Y	Força na direção *Sway*
Z	Força na direção *Heave*
K	Força na direção *Roll*
M	Força na direção *Pitch*
N	Força na direção *Yaw*
J	Transformação não linear entre os sistemas de coordenadas
m	Massa
F_C	Força de Newton
P_C	Momento Linear
H_C	Momento Angular
I_C	Tensão Inercial sobre o centro de gravidade
M_C	Momento no Centro de gravidade do corpo
$C(V)$	Matriz de Coriólis
$D(V)$	Matriz de Amortecimento
$g(\eta)$	Vetor de forças e momentos gravitacionais
M_{RB}	Matriz adicionada de um corpo rígido
C_{RB}	Matriz de Coriólis de um corpo rígido
M_A	Matriz adicionada devido a inércia em torno no fluido

C_A	Matriz de Corióis devido a inércia em torno do fluido
D_P	Potencial de Amortecimento
$F_d x_m$	Força de arrasto na direção x_m
$F_d y_m$	Força de arrasto na direção y_m
$F_d z_m$	Força de arrasto na direção z_m
A	Área
H	Altura visão frontal
L	Largura visão frontal
R	Raio de uma circunferência
X_A	Massa adicionada na direção X
S	Comprimento do Cabo
S_x	Comprimento na direção x
S_y	Comprimento na direção y
W	Peso por metro
T_0	Esforços axiais do cabo na extremidade A
T_1	Esforços axiais do cabo na extremidade B
T_x	Componente X dos esforços axais
T_y	Componente Y dos esforços axais
G_C	Função de transferência do controlador
K_P	Coeficiente Proporcional
K_I	Coeficiente Integrativo
K_D	Coeficiente Derivativo
d	Distância do robô ate a câmera
d_C	Diâmetro do Cano
$E(s)$	Equação característica desejada
a	Polo real do controlador
$E_P(s)$	Equação característica da planta
F	Comando dado aos propulsores
S_d	Deslocamento do submarino
V	Vetor de velocidade linear e angular

Símbolos Gregos

η_1	Vetor de posição do veiculo submarino
η_2	Vetor de orientação do veículo submarino
ϕ	Componente do vetor de orientação na direção *Roll*
θ	Componente do vetor de orientação na direção *Pitch*
ψ	Componente do vetor de orientação na direção *Yaw*
ν_1	Vetor de velocidade linear

ν_2	Vetor de velocidade angular
τ	Vetor de forças e momentos de entrada
τ_{RB}	Vetor de pertubações externas
τ_h	Vetor de forças e momentos devido aos esforços hidrodinâmicos
τ_e	Vetor de forças e momentos devido aos esforços ambientais
τ_R	Vetor de forças e momentos externos
ρ	Massa específica do fluido
τ_A	Vetor de força cinética
α_0	Ângulo formado entre os esforços axiais e o eixo x da extremidade A
α_1	Ângulo formado entre os esforços axiais e o eixo x da extremidade B

Grupos Adimensionais

C_D	Coeficiente de Arrasto
P_x	Número de *Pixels*
C_f	Coeficiente da Exponencial

Subscritos

c	Referente a Mecânica Newtoniana
RB	Referente ao equacionamento de corpo rígido
A	Massa adicionada
m	Referente ao modelo simplificado do robô
d	Referente ao arrasto

Sobrescritos

T	Matriz Transposta

Siglas

ROV	Remotely Operated Vehicle
AUV	Autonomous Underwater Vehicles
PPRH	Programa de Recursos Humanos da Petrobras
SNAME	Society of Naval Architects and Marine Engineers
RGB	Red, Green, Blue
CMY	Cyan, Magenta, Yellow

CMYK	Cyan, Magenta, Yellow, Black
YCbCr	Família de Cores Especiais
YIQ	Padrão de cores utilizado pelo NTSC
NTSC	Sistema de televisão analógico em uso no Estados Unidos
HSI\HSV	Hue, Saturation, Value
USB	Universal Serial Bus
GNC	Guindance, Navigation and Control
PID	Proporcional, Integrativo e Derivativo
SISO	Single Input, Single Output
JPEG	Joint Photographic Experts Group
PNG	Portable Network Graphics

Capítulo 1

Introdução

1.1 Contextualização

O progresso da tecnologia de exploração de petróleo é um dos fatores a impulsionar as industrias do setor. Busca-se novas formas de se extrair a substância em águas cada vez mais profundas, onde o ser humano não é capaz de alcançar diretamente. Com isso, cresce a importância de sistemas automáticos, que podem suportar grandes pressões, exercer forças maiores, gravar imagens, analisar e processar todos os dados mais rapidamente. Mais especificamente, os veículos operados remotamente, ROVs, do inglês, *Remotely Operated Vehicle*, servem para vários tipos de atividades no meio aquático, sendo utilizados nas áreas militares e industriais.

O uso de ROVs na exploração petrolífera do mundo já é difundido há décadas. O primeiro ROV a ser criado foi desenvolvido por Dimitri Rebikoff, The POODLE, em 1953, com aplicação em arqueologia subaquática. A partir deste, vários outros veículos foram criados e com o avanço tecnológico impulsionado por empresas petrolíferas, os robôs ficaram menores e alcançaram maiores profundidades. No final dos anos 90 (Wernli, 1998), foi estimado que existiam mais de cem fábricas de veículos, e mais de cem operadores utilizando aproximadamente 3 mil veículos de vários tamanhos e capacidades. De acordo com a edição de 2006 do *Remotely Operated Vehicles of the World*, existem mais de 450 construtores e desenvolvedores de ROVs e mais de 175 operadores . O número de veículos em campo, hoje em dia,é maior ainda.[1]

Esses veículos são capazes de realizar inspeções com auxílio de um operador localizado em uma embarcação ou estação terrestre. O robô é controlado por um cordão umbilical, que faz a comunicação com o operador, passando as informações vindas de sensores e sinais de comando e controle. Além de fazer a comunicação entre o operador e o veículo, o cordão umbilical é o responsável pela transmissão de energia para o ROV. Este não funciona sem o auxílio do cordão, devido às elevadas perdas de taxa de transmissão via rádio em água. Ele possui câmeras e lâmpadas resistentes a pressão, pois quanto maior a profundidade, menor é a incidência de luz e maior é a pressão.

Os ROVs podem ser utilizados em diferentes situações e finalidades. Para isso, ele usa ferramentas projetadas exclusivamente para atuarem de acordo com a necessidade. Como, por exemplo, os

braços mecânicos, que servem para pegar objetos, operar uma válvula e quase todas as operações que um mergulhador faria.

Eles são classificados em três básicas categorias [1]:

- De observação (*Observation class*): tem a principal função de filmar grandes volumes de água capturando imagens de vários ângulos. Ele não possui garras mecânicas e costumam ser pequenos.
- De trabalho (*work class*): ao contrário da categoria de observação, o ROV de trabalho possui garras mecânicas de diversas formas para auxiliar no desenvolvimento de tarefas, como por exemplo, no conserto de um ducto com vazamento. Eles costumam ser de grande porte.
- De uso especial (*special use*): são veículos construídos para realizar um tipo específico de tarefa. Como por exemplo, os ROVs que são projetados para arar o fundo do mar para enterrar cabos de telecomunicações.

Normalmente, são necessários pilotos e co-pilotos para as operações. Os pilotos controlam o veículo e os co-pilotos controlam a comunicação, registro de dados, os sensores, entre outros.

Existem também os veículos submarinos autônomos, "AUVs", do inglês *Autonomous Underwater Vehicles* , que não dependem dos comandos de um operador humano. Eles atuam por meio de ações pré-programadas e reações ao ambiente externo. Ou seja, eles têm autonomia para seguir suas trajetórias.

1.1.1 VideoRay Pro 4

A Petrobras, com o Programa Petrobras de Formação de Recursos Humanos - PFRH, adquiriu um veículo operado remotamente para pesquisas. O modelo é o Pro 4, da marca VideoRay. A partir desse submarino, que é operado por controle remoto de fábrica, aspirou-se à realização do trabalho. Para a programação do robô, foi utilizado um programa base, disponibilizado pela própria VideoRay, que opera procedimentos simples do ROV, como mudança de luzes e acionamento dos propulsores. O programa foi disponibilizado em C#, uma linguagem de programação orientada a objetos, e apresenta também uma base para leituras de sensores: eles são de profundidade, de direção (compasso de 3 eixos), de aceleração linear e angular (acelerômetro de 3 eixos), de temperatura, de orientação e de umidade interna [2]. Tendo em mãos as leituras de saída de tais sensores, foi implementado um controle.

O Video Ray Pro 4 possui dois propulsores horizontais e um propulsor vertical, os propulsores são sem escova (*brushless*), que funcionam por meio da aplicação de uma diferença de potencial entre seus polos de ligação [2].

O robô tem as seguintes características [2]:

- Dimensões: 37,5 cm x 28,8 cm x 22,3 cm.

- Massa:

  Submersível: 6,1 kg

  Todo o sistema: 79 kg

- Profundidade máxima: 300 m
- Velocidade máxima: 4,2 nós ≅ 2,16066 m/s
- A garra mecânica consegue transportar 4,5 kgf e se estiver seguro (bem preso) basta puxar o cordão umbilical, podendo assim transportar 20 kg.
- Cordão umbilical de até 600 metros

O ROV também possui ajuste de flutuabilidade, ele se auto ajusta, abrindo um recipiente, adicionando ou removendo os pesos do lastro. A câmera é colorida ou preto e branco, com balanceamento automático do branco, com grande faixa dinâmica, compensação ao escuro, obturador digital lento e zoom com até 2,5 vezes. As luzes possuem dois leds com 3,6 lumens [2].

Figura 1.1: Veículo Operado Remotamente VideoRay Pro 4 [2].

1.2 Objetivos do projeto

Para padronizar e melhorar procedimentos feitos por um ROV, o objetivo foi transformar um ROV em um AUV, ou seja, tornar um ROV autônomo, agindo sozinho em uma trajetória pré-programada e reagindo a qualquer ação externa com o propósito de se manter em tal trajetória, utilizando métodos de controle e uma programação interna. O objetivo principal é identificar as entradas possíveis e as variáveis de saída do submarino, automatizando os processos e melhorando a execução, transformando o veículo em uma plataforma apta a receber controles de movimento.

1.3 Descrição do Trabalho

O trabalho é dividido em 5 capítulos. O primeiro capítulo contém a introdução, a descrição do projeto e a definição do objetivo. No segundo capítulo desenvolve-se a modelagem do sistema onde são feitas todas as análises, mostrando as equações, os sentidos e direções que regem o seu movimento. Para que seja alcançado o objetivo principal do trabalho emprega-se ferramentas computacionais para posterior implementação do sistema. Desta forma, o terceiro capítulo apresenta quais são essas ferramentas computacionais necessárias e o quarto capítulo mostra o que foi modificado nestas ferramentas para atender as especificações de projeto. Além disso, no capítulo de implementação, são feitos testes para a compreensão das ferramentas que foram utilizadas. Já no capítulo 5, é feita a implementação do sistema de controle, demostrando o que foi utilizado, qual a equação do controlador e os resultados encontrados. Por fim, o ultimo capítulo conclui-se e comenta-se o desejo de trabalhos futuros.

Capítulo 2

Modelagem

2.1 Introdução

Como o ROV possui dimensões não desprezíveis, sua massa não está concentrada em um único ponto, ou seja, ela está distribuída sobre a sua estrutura, o que implica em ser um corpo rígido. A modelagem engloba os estudos da estática e da dinâmica de corpos rígidos. A estática consiste no estudo do equilíbrio dos corpos, ou seja, a velocidade constante, enquanto a dinâmica estuda os corpos acelerados. O objetivo principal do trabalho é o controle do robô, com isso, o modelo matemático do sistema apresenta valor fundamental. Além disso, é evidente que uma caracterização do sistema real por meio de expressões matemáticas pode permitir discussões sobre estabilidade e controlabilidade, assim como o seu comportamento no domínio do tempo e da frequência [3]. Este capítulo é dedicado para a modelagem do ROV nos aspectos da cinemática e da dinâmica.

2.2 Cinemática

O ROV possui seis graus de liberdade, então são necessários seis componentes com diferentes orientações e posições. Com o intuito de caracterizar o movimento de um corpo no espaço, utilizou-se relações entre diferentes sistemas de coordenadas. São empregados dois sistemas, o sistema de coordenada inercial, que é fixo na Terra, e o sistema de coordenada móvel, fixo ao veículo modelado. Estes são usados para determinar as relações, de acordo com a Figura 2.1 [3].

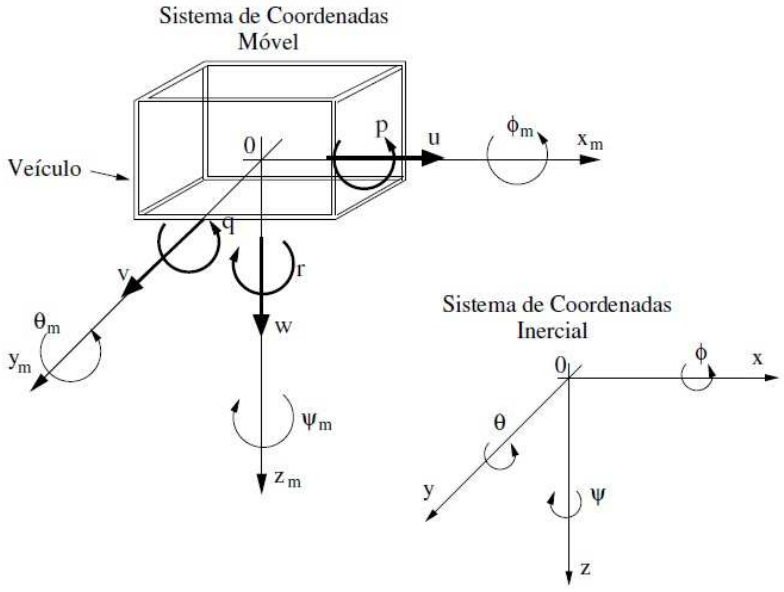


Figura 2.1: Sistema de coordenadas utilizadas.

Tabela 2.1: Variáveis para cada grau de liberdade

Graus de Liberdade		Forças e Momentos	Velocidade Linear e Angular	Posição Inercial
1	Movimento na Direção X (avanço) - *Surge*	X	u	x
2	Movimento na Direção Y (deriva) - *Sway*	Y	v	y
3	Movimento na Direção Z (afundamento) - *Heave*	Z	w	z
4	Rotação sobre o eixo X (jogo) - *Roll*	K	p	ϕ
5	Rotação na Direção Y (arfagem) - *Pitch*	M	q	θ
6	Rotação na Direção Z (guinada) - *Yaw*	N	r	ψ

Agora é possível estabelecer notações para as grandezas associadas à movimentação do veículo. No sistema de coordenadas inercial, a posição, η_1, e a orientação, η_2, podem ser expressas como:

$$\eta_1 = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \qquad \eta_2 = \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} \tag{2.1}$$

É importante notar que, ao calcular as derivadas da matriz, é possível obter as velocidades de translação, $\dot{\eta}_1$, e rotação $\dot{\eta}_2$.

Já para o sistema de coordenadas móveis, pode-se descobrir os valores de uma maneira análoga. Os vetores velocidade de translação v_1, e velocidade de rotação v_2, podem ser escritos como:

$$v_1 = \begin{bmatrix} u \\ v \\ w \end{bmatrix} \qquad v_2 = \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{2.2}$$

Estabelecidas as notações, deve-se restringir o sistema a partir de certas singularidades. Nessa representação, $\theta \pm 90^\circ$ é uma singularidade e pode-se fazer algumas aproximações [3]:

$$-90^\circ \leq \theta \leq +90^\circ \tag{2.3}$$

$$-180^\circ \leq \phi, \psi \leq +180^\circ \tag{2.4}$$

Com as notações estabelecidas e as restrições devido às singularidades consideradas, pode-se começar a representação cinemática do veículo no espaço. As componentes de velocidade translacional inercial e móvel se interagem por meio de um operador.

$$\dot{\eta_1} = J_1(\eta_2)v_1 \tag{2.5}$$

O operador de transformação não linear J_1 é obtido a partir da composição das matrizes de rotação. Uma matriz de rotação pode ser construída a partir da rotação no sentido anti-horário de cada componente de um vetor, em um eixo de rotação por um ângulo qualquer.

Assim, as matrizes de rotação ficam:

$$M_{x,\phi} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\phi & -sen\phi \\ 0 & sen\phi & cos\phi \end{bmatrix} \tag{2.6}$$

$$M_{y,\theta} = \begin{bmatrix} cos\theta & 0 & sen\theta \\ 0 & 1 & 0 \\ sen\theta & 0 & cos\theta \end{bmatrix} \tag{2.7}$$

$$M_{z,\psi} = \begin{bmatrix} cos\psi & sen\psi & 0 \\ -sen\psi & cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.8}$$

Portanto, o operador de transformação será:

$$J_1(\eta_2) = M_{x,\phi}^T M_{y,\theta}^T M_{z,\psi}^T \tag{2.9}$$

$$J_1(\eta_2) = \begin{bmatrix} cos\theta cos\phi & sen\phi sen\theta cos\psi - cos\phi sen\psi & cos\phi sen\theta cos\psi + sen\phi sen\psi \\ cos\theta cos\phi & sen\phi sen\theta sen\psi + cos\phi cos\psi & cos\phi sen\theta sen\psi - sen\phi cos\psi \\ -sen\theta & sen\phi cos\theta & cos\phi cos\theta \end{bmatrix} \tag{2.10}$$

De maneira análoga, pode se definir a relação como:

$$\dot{\eta_2} = J_2(\eta_2)v_2 \tag{2.11}$$

É possível adquirir J_2 usando também as matrizes de rotação. A relação é:

$$v_2 = \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + M_{x,\phi} \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + M_{y,\theta} M_{x,\phi} \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} = J_2^- 1(\eta_2)\dot{\eta_2} \tag{2.12}$$

Resultando em:

$$J_2(\eta_2) = \begin{bmatrix} 1 & sen\phi tan\theta & cos\phi tan\theta \\ 0 & cos\phi & -sen\phi \\ 0 & sen\phi sec\theta & cos\phi sec\theta \end{bmatrix} \tag{2.13}$$

É importante ressaltar que J_2 não é ortogonal:

$$J_2^T(\eta_2) \neq J_2^{-1}(\eta_2) \tag{2.14}$$

Além disso, observa-se que a matriz J_2 não é definida para $\theta = \pm\, 90\,^\circ$ [3]. Apesar dessa restrição, isso não será um problema, já que, como foi dito anteriormente, o veículo estudado possui uma singularidade nesse ponto.

2.3 Dinâmica de Corpo Rígido

2.3.1 Mecânica Newtoniana

A formulação de Newton-Euler tem base na Segunda Lei de Newton:

$$m\dot{V_c} = F_c \tag{2.15}$$

que relaciona a força de um corpo com a massa e a aceleração do corpo. Quando não há força, não há aceleração, portanto conclui-se que o corpo está com velocidade constante ou o corpo está parado. Esse resultado é conhecido como Primeira Lei de Newton. Isaac Newton (1643 - 1727) publicou essas leis em 1687.[4]

Leonhard Euler (1707 - 1783) sugeriu expressar a Segunda Lei de Newton em termos de conservação dos momentos linear(P_c) e angular(H_c):

$$\dot{P_c} \triangleq F_c \tag{2.16}$$

$$P_c \triangleq mV_c \tag{2.17}$$

$$\dot{H_c} \triangleq M_c \tag{2.18}$$

$$H_c \triangleq I_c w \tag{2.19}$$

onde a força F_c e o momento M_c se referem ao centro de gravidade do corpo, w é o vetor de velocidade angular e I_c é a tensão inercial sobre o centro de gravidade do corpo. Como as duas leis de conservação são expressadas em termos de vetor, a aplicação dessas equações é comumente chamada de mecânica vetorial. Esses resultados são conhecidos como Primeiro e Segundo Axiomas de Euler.[4]

De acordo com Fossen (1994), a equação do movimento dinâmico não linear de um robô com seis graus de liberdade é dada por:

$$M\dot{V} + C(V)V + D(V)V + g(\eta) = \tau \tag{2.20}$$

Onde :

M = Matriz inercial (incluindo massa adicionada)
$C(V)$ = Matriz de Coriólis e termos centrípetos (incluindo massa adicionada)
$D(V)$ = Matriz de amortecimento
$g(\eta)$ = Vetor de forças gravitacionais e momentos
τ = Vetor de entrada de controle

Utilizando a formulação de Newton-Euler, é possível simplificar a equação de movimento dinâmico não linear de um robô com seis graus com relação ao eixo de coordenadas fixo no corpo. Chega-se a seguinte representação vetorial, onde RB é a sigla de corpo rígido:

$$M_{RB}\dot{V} + C_{RB}(V)V = \tau_{RB} \tag{2.21}$$

Na expressão, M_{RB} representa a matriz de massa e inércia, C_{RB} representa a matriz de Coriólis e termos centrípetos do corpo rígido, $V = [u, v, w, p, q, r]^t$ representa os vetores de velocidade linear e angular do corpo fixo e $\tau_{RB} = [X, Y, Z, K, M, N]^t$ representa o vetor de todas as forças e momentos externos.[4]

Considerando que o sistema de coordenadas fixo no corpo coincide com os eixos principais de inércia, pode-se chegar nos valores de $[X, Y, Z, K, M, N]$:

$$\begin{aligned}
m[\dot{u} - vr + wq - x_G(q^2 + r^2) + y_G(pq - \dot{r}) + z_G(pr + \dot{q})] &= X \\
m[\dot{v} - wp + ur - y_G(r^2 + p^2) + z_G(qr - \dot{p}) + x_G(qp + \dot{r})] &= Y \\
m[\dot{w} - uq + vp - z_G(p^2 + q^2) + x_G(rp - \dot{q}) + yx_G(rq + \dot{p})] &= Z \\
I_x\dot{p} + (I_z - I_y)qr + m[y_G(\dot{w} - uq + vp) - z_G(\dot{v} - wp + ur)] &= K \\
I_y\dot{q} + (I_x - I_z)rp + m[z_G(\dot{u} - vr + wq) - x_G(\dot{w} - uq + vp)] &= M \\
I_z\dot{r} + (I_y - I_x)pq + m[x_G(\dot{v} - wp + ur) - y_G(\dot{u} - vr + wq)] &= N
\end{aligned} \tag{2.22}$$

As magnitudes do vetor τ_{RB} podem ser atribuídas a três componentes.

$$\tau_{RB} = \tau_h + \tau_e + \tau \tag{2.23}$$

onde τ_h representa as forças e momentos hidrodinâmicos, τ_e representa as forças e momentos do ambiente agindo no veículo e tal as forças e momentos de propulsão[4].

2.3.2 Esforços Hidrodinâmicos

É comum assumir em hidrodinâmica que as forças e os momentos de um corpo rígido podem ser linearmente considerados em dois casos. No primeiro, devido a frequência de excitação das ondas o corpo oscila, não existindo ondas incidentes. No segundo, o corpo está restrito à oscilações e há ondas incidentes regulares. Este segundo caso consiste na força do vento, das ondas e das correntezas e não foi levado em consideração no trabalho [4].

As forças e momentos do primeiro caso, podem ser identificadas pela soma de três componentes, como mostrado na equação (2.24):

$$\tau_R = -M_A\dot{V} - C_A(V)V - D_P(V)V - g(\eta) \tag{2.24}$$

onde $-M_A\dot{v} - C_A(V)v$ representa a massa adicionada devido a inércia em torno do fluido, $D_P(V)v$ representa o potencial de amortecimento devido à energia levada pela geração de ondas superficiais e $g(\eta)$ representa as forças restauradoras devido ao peso e empuxo.

Além do potencial de amortecimento em razão das ondas superficiais, devem ser incluídos outros efeitos de amortecimento como o atrito da superfície, a interação com as ondas e o efeito de vórtice, porém esses efeitos não serão considerados neste trabalho. Com isso, a equação (2.23), fica desta forma:

$$\tau_{RB} = \tau_h + \tau \tag{2.25}$$

A componente hidrodinâmica τ_h, pode ser definida como [4]:

$$\tau_h = -M_A\dot{V} - C_A(V)V - D(V)V - g(\eta) \tag{2.26}$$

2.3.2.1 Força de Arrasto

Segundo ISE(2000), quando um veículo submersível se movimenta em velocidade constante, a propulsão gerada pelos propulsores se iguala à força de arrasto do veículo. Sendo assim, dado as medidas fornecidas pelo manual do robô, é possível determinar o impulso dos propulsores.

A força de arrasto é definida como:

$$\begin{bmatrix} F_d x_m \\ F_d y_m \\ F_d z_m \end{bmatrix} = \begin{bmatrix} -\frac{\rho C_d A v_e |v_e|}{2} & 0 & 0 \\ 0 & -\frac{\rho C_d A v_e |v_e|}{2} & 0 \\ 0 & 0 & -\frac{\rho C_d A v_e |v_e|}{2} \end{bmatrix} * \begin{bmatrix} \hat{x_m} \\ \hat{y_m} \\ \hat{z_m} \end{bmatrix} \tag{2.27}$$

Nesta expressão, ρ é massa específica do fluido, C_d é o coeficiente de arrasto, A é a área molhada e v é a velocidade do veículo. A força de arrasto é negativa, pois ela é contrária ao movimento do corpo.

Para determinar o arrasto do ROV considerado, inicialmente determinou-se a sua direção e sentido de movimento. Como ele possui seis graus de liberdade e três eixos de translação, então foram observadas três forças de arrasto. Considerou-se o modelo simplificado do ROV como pode ser visto na figura 2.2.

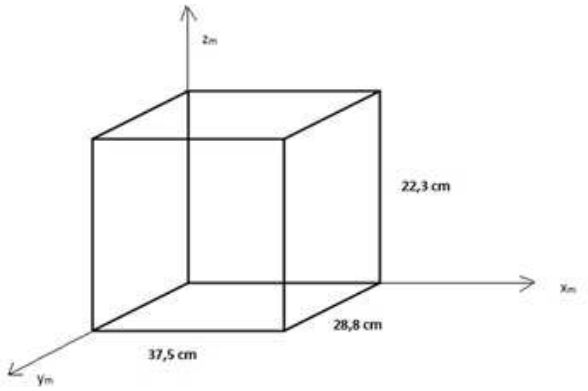


Figura 2.2: Desenho simplificado do veículo estudado mostrando suas dimensões.

Calcula-se dessa forma, utilizando a equação 2.27 e o modelo simplificado, as forças de arrasto em cada eixo.

- Para direção de movimento no eixo x_m:

  A área molhada considerada é 28.8 cm x 22.3 cm que é igual a 642.24 cm^2. O C_d para geometrias retangulares é 0.82. [1]

  Tem-se então F_{dxm}:

$$F_d x_m = -\frac{10^3 \times 0.82 \times 6.4224 \times 10^{-2} \times v|v|}{2}$$

$$F_d x_m = -26.33184 \times v|v|$$

  Considerando a velocidade máxima dada pelo fabricante, obtém-se que:

$$F_d x_m = -122.92892N$$

- Para direção de movimento no eixo y_m:

  A área molhada, novamente, é um retângulo. Como visto na subitem anterior, o C_d para tal geometria é 0.82. conclui-se então que a força de arrasto para o movimento no eixo y_m é:

$$A = 37.5 \times 22.3 = 836.25cm^2 = 8.3625 \times 10^{-2}m^2$$

$$F_d y_m = -\frac{10^3 \times 0.82 \times 6.4224 \times 10^{-2} \times v|v|}{2}$$

$$F_d y_m = -34.28625 \times v|v|$$

  Considerando a velocidade máxima do veículo dada pelo fabricante, tem-se que:

$$F_d y_m = -160.06369N$$

[1] Fonte: https://en.wikipedia.org/wiki/$Drag_coefficient$.

- Para direção de movimento no eixo z_m:

  Assim como nas outras direções, a área estimada será um retângulo, então o C_d é 0.82.

$$A = 37.5 \times 28.8 = 1080cm^2 = 1.08 \times 10^{-1}m^2$$

$$F_d z_m = -\frac{10^3 \times 0.82 \times 1.08 \times 10^{-1} \times v|v|}{2}$$

$$F_d z_m = -44.28 \times v|v|$$

  Considerando a velocidade máxima do veículo dada pelo fabricante, define-se que:

$$F_d z_m = -206.71903N$$

Após o cálculo simplificado, houve a tentativa de especificar melhor o modelo e adquirir valores mais próximos. Foi feita a simplificação de sua estrutura para três cilindros, onde existem dois menores laterais e um maior central. A visão frontal, lateral e superior do ROV simplificadas são representadas na figura 2.3.

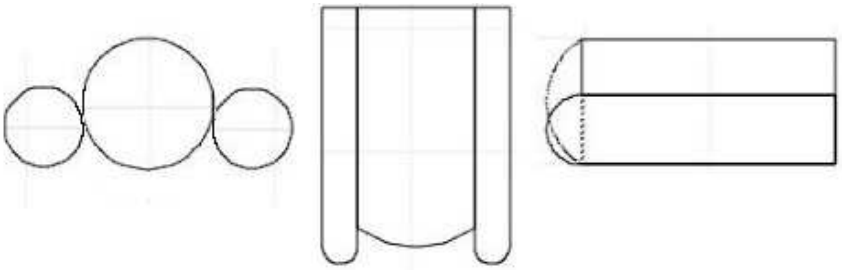


Figura 2.3: Desenhos simplificados da visão frontal, superior e lateral, respectivamente.

Através das dimensões dadas pelo manual e a partir da simplificação da visão frontal dada pela figura 2.3, determinou-se os raios dos cilindros para encontrar a força de arrasto na direção x_m. Na figura 2.4, defini-se quais são os raios e dimensões.

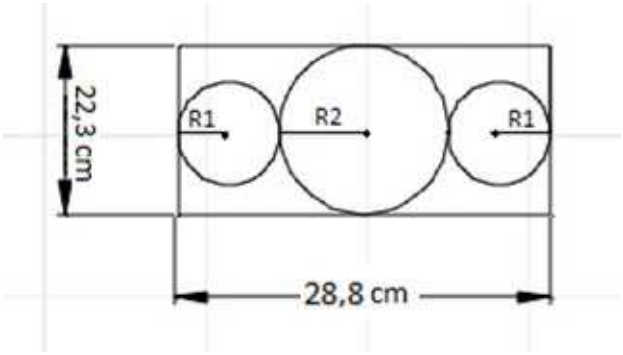


Figura 2.4: Desenho simplificado com as medidas da visão frontal

Os raios encontrados foram:

$$R_1 = 1.625cm \text{ e } R_2 = 11.15cm$$

Com os dados dos raios determinados, utilizou-se dos mesmos para definir as áreas. A área de um círculo é definida como:

$$A = \pi R^2 \tag{2.28}$$

As áreas encontradas foram:

$$A_1 = 8.29577cm^2 \text{ e } A_2 = 390.57032cm^2$$

utilizando a aproximação de π para 5 casas decimais.

O C_d para tal geometria é 0.295[2]. Sendo assim, a partir dos dados encontrados calculou-se as forças de arrasto pra cada cilindro e em seguida foram somadas para encontrar a força resultante final.

$$F_d = 2 \times F_{d1} + F_{d2} \tag{2.29}$$

$$F_{d1} = -\frac{10^3 \times 0.295 \times 8.29577 \times 10^{-4} \times v|v|}{2}$$

$$F_{d1} = -0.61181 \times v|v|$$

$$F_{d2} = -\frac{10^3 \times 0.295 \times 3.90570 \times 10^{-2} \times v|v|}{2}$$

$$F_{d2} = -5.76091 \times v|v|$$

Conclui-se então que a força de arrasto no eixo x_m é $F_d = -6,98454 \times v|v|$. Considerando a velocidade máxima do veículo dada pelo fabricante, admite-se que:

$$F_d x_m = -32,606968N$$

Nota-se, portanto, que os dois valores estão muito diferentes. Para um melhor precisão dos valores de arrasto, é necessário a utilização de um software onde o veículo é desenhado, sendo possível adquirir a força de arrasto com as devidas considerações. Além disso, para trabalhos futuros, também deverão ser feitos novos testes para adquirir a força de arrasto real, para uma comparação com os valores calculados teoricamente.

2.4 Massa Adicionada

A massa adicionada, que pode ser entendida como as forças e momentos de pressão induzidas devido ao movimento forçado do corpo, é proporcional à aceleração. Consequentemente, as forças

[2]Fonte: https://www.grc.nasa.gov/www/k-12/airplane/shaped.html

e acelerações de massa adicionada serão diferentes em 180° de fase do movimento harmônico forçado. Foi assumido que os coeficientes de massa adicionada são constantes e independentes da frequência da onda, para que haja uma completa submersão do veículo [4]. Para se determinar a massa adicionada, foi expressado a força cinética do fluido, de acordo com Lamb (1932):

$$\tau_A = \frac{v^T M_A v}{2} \tag{2.30}$$

Qualquer movimento do veículo induzirá o movimento do fluido estacionário de outra maneira. Para que o fluido passe através do veículo, o mesmo deve mover se ao lado e em seguida atrás do veículo.

Utilizando a expressão (2.31) e a notação de SNAME, a matriz adicionada de inércia pode ser definida como:

$$M_A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \triangleq - \begin{bmatrix} X_{\dot{u}} & X_{\dot{v}} & X_{\dot{w}} & X_{\dot{p}} & X_{\dot{q}} & X_{\dot{r}} \\ Y_{\dot{u}} & Y_{\dot{v}} & Y_{\dot{w}} & Y_{\dot{p}} & Y_{\dot{q}} & Y_{\dot{r}} \\ Z_{\dot{u}} & Z_{\dot{v}} & Z_{\dot{w}} & Z_{\dot{p}} & Z_{\dot{q}} & Z_{\dot{r}} \\ K_{\dot{u}} & K_{\dot{v}} & K_{\dot{w}} & K_{\dot{p}} & K_{\dot{q}} & K_{\dot{r}} \\ M_{\dot{u}} & M_{\dot{v}} & M_{\dot{w}} & M_{\dot{p}} & M_{\dot{q}} & M_{\dot{r}} \\ N_{\dot{u}} & N_{\dot{v}} & N_{\dot{w}} & N_{\dot{p}} & N_{\dot{q}} & N_{\dot{r}} \end{bmatrix} \tag{2.31}$$

A matriz M_A representa a massa adicionada em cada direção e com suas respectivas acelerações. Por exemplo, a massa adicionada na direção X, X_A, com velocidade u, pode ser representada desta forma:

$$X_A \triangleq \frac{\partial X}{\partial \dot{u}}$$

Na maioria das aplicações os termos da diagonal principal da matriz, o que corresponde a $X_{\dot{u}}$,$Y_{\dot{u}}$,$Z_{\dot{w}}$ e assim por diante, são positivos. Porém, para algumas frequências, são negativos, como no caso de catamarãs, seções em forma de bulbo e veículos submersíveis quando próximos à superfície.

Para um corpo rígido em repouso e assumindo que o fluido é ideal, sem a incidência de ondas, sem correnteza e com frequência independente, a massa adicionada, M_A é positiva e igual a sua transposta, M_A^T. Considerando um fluido real, os termos da matrizes serão diferentes, porém $M_A = M_A^T$ ainda será uma boa aproximação [4].

Para veículos, como ROV, operando fora da zona do efeito de ondas, supor simetria e frequência independentes tem se mostrado razoável. Geralmente, os termos da diagonal principal da matriz adicionada nesse veículo são maiores se comparados aos outros elementos [4].

Para um corpo rígido que se move com um fluido ideal, a matriz de Coriolis e os termos centrípetos C_A podem ser sempre parametrizados como o negativo da sua transposta, C_A = - C_A^T.

Desta forma, pode ser definida como [4]:

$$C_A = \begin{bmatrix} 0 & 0 & 0 & 0 & -a_3 & a_2 \\ 0 & 0 & 0 & a_3 & 0 & -a_1 \\ 0 & 0 & 0 & -a_2 & a_1 & 0 \\ 0 & -a_3 & a_2 & 0 & -b_3 & b_2 \\ a_3 & 0 & -a_1 & b_3 & 0 & -b_1 \\ -a_2 & a_1 & 0 & -b_2 & b_1 & 0 \end{bmatrix} \tag{2.32}$$

onde

$$\begin{aligned} a_1 &= X_{\dot{u}}u + X_{\dot{v}}v + X_{\dot{w}}w + X_{\dot{p}}p + X_{\dot{q}}q + X_{\dot{r}}r \\ a_2 &= Y_{\dot{u}}u + Y_{\dot{v}}v + Y_{\dot{w}}w + Y_{\dot{p}}p + Y_{\dot{q}}q + Y_{\dot{r}}r \\ a_3 &= Z_{\dot{u}}u + Z_{\dot{v}}v + Z_{\dot{w}}w + Z_{\dot{p}}p + Z_{\dot{q}}q + Z_{\dot{r}}r \\ b_1 &= K_{\dot{u}}u + K_{\dot{v}}v + K_{\dot{w}}w + K_{\dot{p}}p + K_{\dot{q}}q + K_{\dot{r}}r \\ b_2 &= M_{\dot{u}}u + M_{\dot{v}}v + M_{\dot{w}}w + M_{\dot{p}}p + M_{\dot{q}}q + M_{\dot{r}}r \\ b_3 &= N_{\dot{u}}u + N_{\dot{v}}v + N_{\dot{w}}w + N_{\dot{p}}p + N_{\dot{q}}q + N_{\dot{r}}r \end{aligned}$$

Geralmente, o movimento de veículos submersíveis com 6 graus de liberdade à alta velocidade, será altamente não linear e acoplado. Entretanto, em várias aplicações do ROV, o mesmo opera apenas com velocidades pequenas. Se o veículo também tiver três planos de simetria, isso sugere que pode ser removida a contribuição dos elementos fora da diagonal principal da matriz M_A. Com isso, M_A e C_A serão definidos como [4]:

$$M_A = -diag(X_{\dot{u}}, Y_{\dot{v}}, Z_{\dot{w}}, K_{\dot{p}}, M_{\dot{q}}, N_{\dot{r}}) \tag{2.33}$$

$$C_A = \begin{bmatrix} 0 & 0 & 0 & 0 & -Z_{\dot{w}}w & Y_{\dot{v}}v \\ 0 & 0 & 0 & Z_{\dot{w}}w & 0 & -X_{\dot{u}}u \\ 0 & 0 & 0 & -Y_{\dot{v}}v & X_{\dot{u}}u & 0 \\ 0 & -Z_{\dot{w}}w & Y_{\dot{v}}v & 0 & -N_{\dot{r}}r & M_{\dot{q}}q \\ Z_{\dot{w}}w & 0 & -X_{\dot{u}}u & N_{\dot{r}}r & 0 & -K_{\dot{p}}p \\ -Y_{\dot{v}}v & X_{\dot{u}}u & 0 & -M_{\dot{q}}q & K_{\dot{p}}p & 0 \end{bmatrix} \tag{2.34}$$

Determinar os termos fora da diagonal é muito difícil tanto experimentalmente quanto na teoria. A aproximação para apenas a diagonal é muito boa em várias aplicações. Essa aproximação é possível, pois quando a matriz adicionada é maior que zero, os termos fora da diagonal são muito menores que os da principal [4].

2.5 Cálculo da Tensão vinda do Cordão Umbilical

O cordão umbilical sofre a ação de esforços externos e internos. Os principais esforços externos que agem sobre ele são os de arrasto hidrodinâmico e as forças restaurativas. As forças internas são representas pela tensão e o amortecimento axiais, que independem do ambiente de operação.

Nesta seção foi calculado apenas a tensão axial do cabo, pois não era necessário para o projeto os outros esforços.

Quando os elementos de um cabo são tracionados, eles apresentam apenas esforços axiais. A equação que será formulada abaixo assumiu que o cabo estava sujeito à ação do próprio peso e de esforços de tração que variavam de acordo com o comprimento. A Figura 2.7 representa um cabo preso nas extremidades dos pontos A e B com comprimento S, que foi decomposto nas direções x e y para melhor desenvolvimento e o peso do corpo por metro.

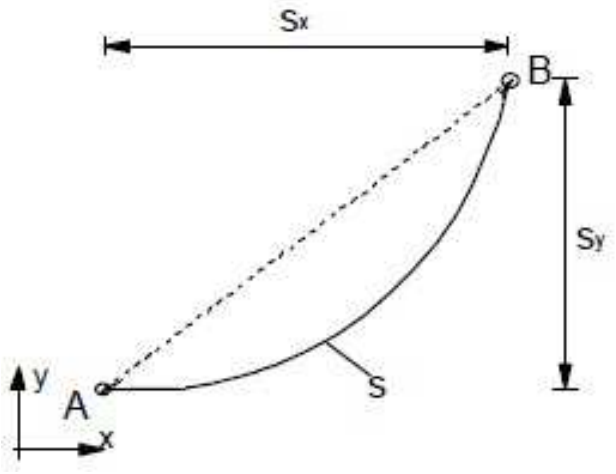


Figura 2.5: Cabo que representa o cabo umbilical, preso nas extremidades.

Os pontos de fixação A e B, criam esforços axiais no cabo representados por T_0 e T_1 que são decompostos segundo as direções x e y. Ao fazer a decomposição das tensões cria-se um ângulo com a direção do cabo, que pode ser observado na Figura 2.8 [5].

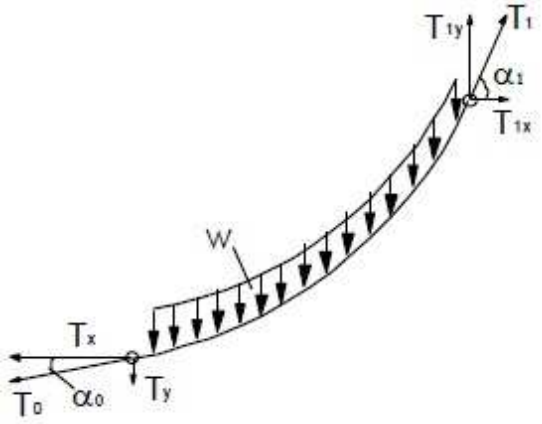


Figura 2.6: Representação das forças que agem no cabo.

O esforço de tração axial é diferente em cada seção transversal ao longo do comprimento S do cabo, porém as reações no eixo x são constantes como pode ser evidenciado ao se utilizar o equilíbrio das forças na direção x, sendo assim tem-se que:

$$T_1 \times cos\alpha_1 - T_0 \times cos\alpha_0 = 0 \tag{2.35}$$

$$T_x = T_0 \times cos\alpha_0 \tag{2.36}$$

Utilizando o equilíbrio das forças para o eixo y:

$$T_1 \times sen\alpha_1 = T_0 \times sen\alpha_0 + W \times S \tag{2.37}$$

onde W é $\frac{peso}{metro}$ e S é dado em metro.

Dividindo todos os termos da equação por T_x, tem-se que:

$$\frac{T_1 \times sen\alpha_1}{T_x} = tan\alpha_0 + \frac{W \times S}{T_x} \tag{2.38}$$

O primeiro termo da equação acima é a derivada de y com relação x, que representa a inclinação da reta, ou seja, a tangente de α. Para melhor entendimento basta observar a figura 2.9, que representa o gráfico das reações:

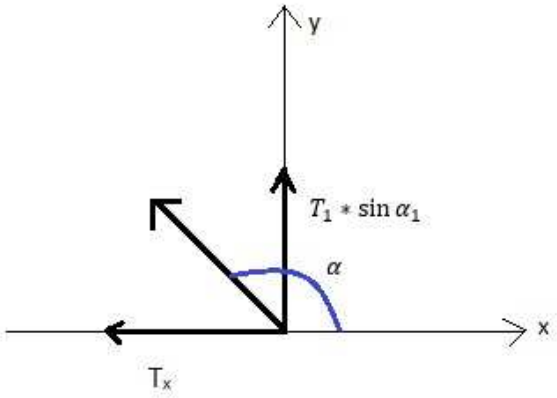


Figura 2.7: Gráfico que representa a derivada das reações.

Sendo assim, a equação fica desta forma [5]:

$$\frac{dy}{dx} = tan\alpha_0 + \frac{W \times S}{T_x} \tag{2.39}$$

Sabendo que o elemento infinitesimal do cabo é dado por:

$$dS = \sqrt{dx^2 + dy^2} \tag{2.40}$$

Com isso, pode se escrever dy em função de dx e dS ou dx em função de dy e dS. Resolvendo as equações define-se que:

$$\frac{dy^2}{dx^2} = \sqrt{tan\alpha_0 + \frac{W \times S}{T_x}} \tag{2.41}$$

$$dx^2 = dS^2 - dy^2 \tag{2.42}$$

Substituindo a equação (2.41) na equação (2.42):

$$dx = \frac{dS}{\sqrt{1 + (tan\alpha_0 + \frac{W \times S}{T_x})}} \tag{2.43}$$

E para dy segue o mesmo principio:

$$dy = \frac{dS}{\sqrt{1 + \frac{1}{(tan\alpha_0 + \frac{W \times S}{T_x})}}} \tag{2.44}$$

Resolvendo as derivadas tem-se que [5]:

$$S_x = \frac{T_x}{W} \times (ln(tan\alpha_0 + \frac{WS}{T_x} + \sqrt{1 + (tan\alpha_0 + \frac{WS}{T_x})^2}) - ln(\frac{sen\alpha_0 + 1}{cos\alpha_0})) \tag{2.45}$$

$$S_y = \frac{\sqrt{T_x^2 + (tan\alpha_0 T_x + WS)^2} - T_x\sqrt{1 + tan{\alpha_0}^2}}{W} \tag{2.46}$$

Reorganizando os termos da fórmula [5]:

$$T_x = \frac{W(S^2 - S_y^2)}{2(S_y\sqrt{1 + tan{\alpha_0}^2} - Stan\alpha_0}) \tag{2.47}$$

Após ser encontrado o valor de T_x, a partir da equação (2.36), acha-se o valor de T_0. Com T_0 definido, determina-se a tensão T_y:

$$T_y = T_0 sen\alpha_0 + WS \tag{2.48}$$

A partir deste cálculo define-se a força resultante do cabo:

$$T_1 = \sqrt{T_x^2 + T_y^2} \tag{2.49}$$

Capítulo 3

Ferramentas Computacionais

3.1 Introdução

Para a funcionalidade do ROV é necessário o uso de ferramentas computacionais para o controle do seu movimento, porém o conjunto de elementos disponibilizados pelo fabricante são limitadas. Tendo em vista o objetivo do projeto, serão utilizados novos recursos juntamente com os esforços e tensões encontrados. Neste capítulo serão apresentadas as ferramentas computacionais utilizadas no robô, assim como as linguagens implementadas.

3.2 Plataforma Convencional

Para a utilização dos ROV's da VideoRay, existe um software chamado *VideoRay Cockpit*. Esta plataforma tem a seguinte forma:

Figura 3.1: VideoRay Cockpit.

Nota-se que todas as informações necessárias para a operação de um piloto estão na plataforma: saídas de sensores, saída da câmera, temperatura da água, bússola, entre outros. Utilizando essa plataforma, é necessário o uso de um controle remoto, que é conectado ao computador, para operar o robô. Apesar de ser uma plataforma amigável e fácil de ser operada, o software *VideoRay Cockpit* não permite incluir lógicas para limitar ou operar o veículo imediatamente, sem a presença de um operador.

3.3 Plataforma Utilizada

Como o objetivo do projeto é transformar o ROV em um AUV, ou seja, utilizar comandos para que o robô seja autônomo, dos quais a plataforma Cockpit não oferece, foi necessário procurar outras plataformas para programação.

Em contato com a *VideoRay*, foi recebido um programa base, na linguagem C#, para o ROV. Utilizando o software *Visual Studio*, da *Microsoft*, foi possível editar e mudar os valores e variáveis da plataforma.

Inicialmente, esse programa base é simples, não contendo todas as saídas de sensores e formas básicas de operar, apenas contêm barras de rolagem para a propulsão e leitura de alguns sensores. Porém, nela existe a possibilidade de edição, onde pode-se programar tudo o que for necessário para o trabalho.

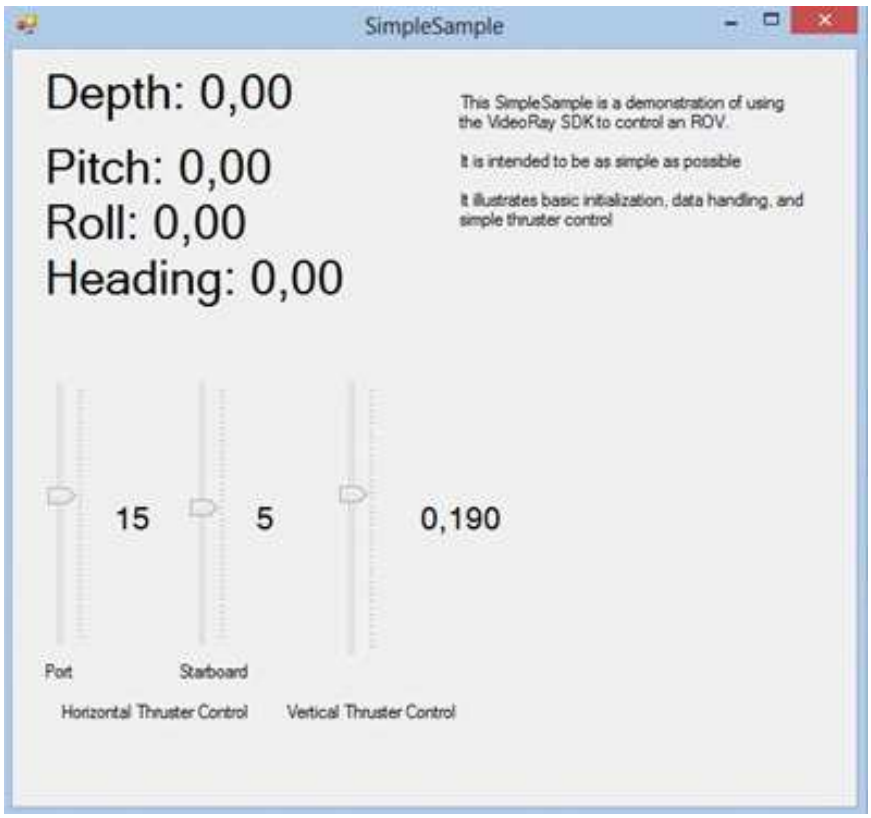


Figura 3.2: Painel Base.

No programa *SimpleSample*, existe um laço, geralmente chamado de *loop*, englobando todas as interações existentes. Esse laço garante uma atualização do programa a cada ciclo, mais especificamente uma taxa de 10 Hertz, o que significa que o programa se atualiza a cada 0,1 segundos. Além disso, pode-se colocar valores de -100 a 100 nos propulsores horizontais e valores de -1 a 1 no propulsor vertical.

O robô, como visto na sessão 2.2 possui um sistema de coordenadas móvel: x, y, z. E os sensores de movimento em relação a esses eixos, respectivamente, são: *Roll* , *Pitch* e *Heading* . Na figura 2.6 é possível visualizar em qual eixo os sensores estão relacionados. Observa-se na figura que *Yaw* é o mesmo que *Heading*.

Com essas variáveis e utilizando o poder computacional de uma linguagem de programação qualquer, é possível utilizar uma lógica para aplicar comandos. Essa lógica se baseia em operadores condicionais, os quais, dependendo de um valor ou limite, levam a respostas de processos diferentes. As combinações dessas lógicas levam à aplicação de um controle, levando o ROV à autonomia.

3.4 Processamento de Imagens

Uma imagem pode ser definida como uma função bidimensional $f(x, y)$, onde x e y são coordenadas espaciais e a amplitude de f, para qualquer pares de coordenadas, é chamado de intensidade ou nível de cinza nesse ponto. Quando x, y e a intensidade forem finitos, quantidades discretas, chama-se de imagem digital. Ela é composta por um número finito de elementos, cada um dos quais tem uma determinada localização e valor. Esses elementos são chamados de *picture elements*, *image elements*, *pels* e *pixels*. Onde *pixels* é o termo mais comumente utilizado para denotar os elementos. O principal objetivo para o aprimoramento da imagem por meio do processamento se deve ao fato do resultado ser melhor que a imagem original para uma específica aplicação [6].

Existem duas divisões de categorias quanto ao modo de melhoria das imagens: métodos no domínio espacial e métodos no domínio da frequência. O método no domínio especial refere-se ao próprio plano da imagem e abordagens nesta categoria são baseadas na manipulação direta de *pixels*. No domínio da frequência, as técnicas são baseadas em modificar a transformada de Fourier da imagem [6].

Para bom entendimento do processamento, é preciso compreender o conceito de histograma. Em uma imagem, o conjunto de números que indicam o percentual de *pixels* que apresentam um determinado nível de cinza é chamado de histograma. Normalmente, esses valores são representados por um gráfico de barras que fornece para nível de cinza o número de *pixels* correspondentes na imagem. O histograma pode indicar a qualidade de uma imagem quanto ao nível de contraste e quanto ao brilho médio, ou seja, se a imagem é predominantemente clara ou escura [7].

No histograma da imagem acima, há grande concentração de *pixels*, nos valores mais baixos de cinza, correspondendo desta forma uma imagem predominantemente escura.

Figura 3.3: Histograma [7].

3.4.1 Limiarização

A limiarização ou *Thresholding* respalda-se em separar as regiões de uma imagem quando esta possui duas classes (o fundo e o objeto). O processo de limiarizar uma imagem é chamado de binarização, pois a saída do procedimento é binária. A forma mais simples de limiarização consiste na bipartição do histograma, ou seja, através de um certo valor de limiar (*threshold*), os *pixels* cujo valor de cinza são iguais ou maiores que este parâmetro são convertidos para preto e os demais para branco. Quando em uma imagem de escala de cinza, existe no histograma apenas dois tons de cinza, portanto a limiarização fica trivial, ou seja, um tom deve ser preto e o outro deve ser branco [7]. A partir do problema a ser resolvido, é escolhido o limiar.

3.4.2 Modelo de Cores

O modelo de cores é uma representação tridimensional na qual cada cor é representada por um ponto no sistema de coordenadas. Existem vários modelos de representação de cores, os mais utilizados são: RGB (*red, green, blue*), CMY (*cyan, magenta, yellow*), CMYK (mesmas cores do CMY, porem adicionou-se a cor preta, representada pela letra K), YCbCr (família de cores especiais e utilizado em várias técnicas de compressão de vídeo), YIQ (padrão NTSC de TV em cores) e HSI (*hue, saturation, intensity*), também denominado de HSV (*hue, saturation, value*).

O modelo utilizado no projeto é o RGB e é o mais empregado em câmeras e monitores de vídeo. Este modelo é baseado em um sistema de coordenadas cartesianas representado por um cubo onde três dos seus vértices são as cores primárias (vermelho, amarelo e azul), os outros três as cores secundárias. O vértice junto à origem é o preto e o mais afastado corresponde à cor branca, conforme ilustrado na figura 3.5. A escala de cinza, nesse modelo, se estende através da diagonal do cubo, entre o vértice que representa o preto e o que representa o branco. Por conveniência, geralmente assume-se que os valores máximos de R, G e B estão normalizados na faixa de 0 a 1 [7].

3.5 Linguagem

A linguagem C# é uma linguagem de programação orientada a objetos, fortemente tipada, criada pela *Microsoft*. A companhia baseou C# na linguagem C++ e Java. Embora desenvolvida

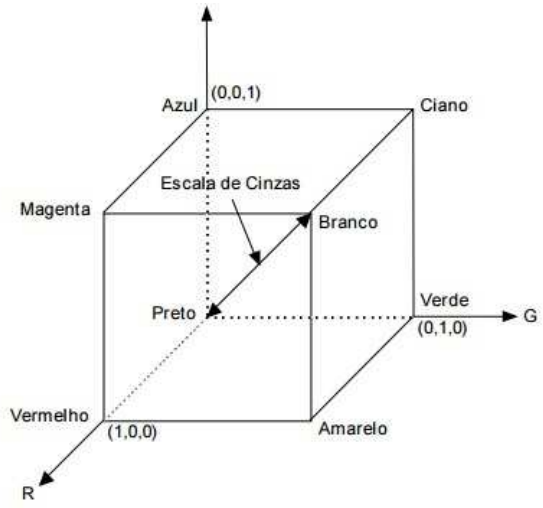


Figura 3.4: Representação do modelo RGB.

por várias pessoas, a criação da linguagem C# é atribuída a Anders Hejlsberg. Este era um desenvolvedor de compiladores na Borland e possuía criações conhecidas, como o Turbo Pascal e o Delphi. Hoje, Anders é um *Distinguished Engineer* na *Microsoft*, um cargo maior que um engenheiro sênior, normalmente concedido para inventores na área de programação computacional. [8]

A linguagem C# é restrita e melhorada quando comparada ao C e ao C++. Por exemplo, ponteiro e aritmética sem checagem só podem ser utilizados no modo inseguro, uma modalidade especial. Ou seja, ao utilizar ponteiros, fica impossível o uso de um endereço inválido que possa danificar o computador. Além disso, em operações aritméticas, há uma checagem de sobrecarga (*overflow*), impedindo uma resposta errada do sistema.

Outro dos objetivos da linguagem C# é simplificar a implementação do ambiente de execução, que vem com a não permissão de herança múltipla. Herança múltipla é a utilização de classes que internamente possuem outras classes. O seguido uso desse tipo de ferramenta acaba tornando o código confuso e de difícil compreensão.

Em comparação com o C++, o C# é mais seguro em relação aos tipos de variável, como inteiro ou ponto flutuante (*float*). Na linguagem C#, é necessário a marcação explícita de qualquer conversão implícita realizada pelo usuário, tentando evitar diretamente algum comportamento não esperado pelo usuário do programa. Por exemplo, o usuário não quer uma conversão, mas acaba fazendo uma operação aritmética de um *float* com um inteiro, resultando em um valor de resposta errado. O C# trava esse tipo de operação.

A lista de características principais são:

- Simplicidade
- Completamente orientada a objetos
- Fortemente tipificada

- Gera código gerenciado
- Tudo é um objeto
- Controle de versões
- Suporte a código legado
- Flexibilidade
- Linguagem gerenciada

Com o programa base aquirido, após aprender e estudar a linguagem C#, partiu-se para a implementação de lógicas.

Capítulo 4

Implementação Computacional

4.1 Introdução

Após a apropriação das ferramentas computacionais, foram aplicados diferentes tipos de lógicas para fins de projeto. O programa cedido pela *Video Ray*, possui uma biblioteca própria (*vrlib*), a qual foi estudada e analisada para se extrair o máximo de informações necessárias. Nessa biblioteca existem todas as funções, variáveis e sensores que podem ser utilizados no robô. Neste capítulo, serão mostradas todas as implementações e melhorias feitas no programa base e também alguns experimentos que foram feitos para identificar o que as saídas do sistema representam para que fosse feito um controle com os parâmetros corretos.

4.2 Lógicas Implementadas

Inicialmente, foi preciso encontrar as variáveis que representavam as saídas dos outros sensores que seriam necessários, como: acelerômetro linear, acelerômetro angular, orientação, pressão e luzes. Ao encontrá-las, foi aplicado um comando para que elas fossem colocadas no painel para visualização das medidas. Isto foi feito a partir do código abaixo:

```
AceleracaoLinearLabel.Text = "Aceleracao Linear: " +
rov.navigation.Accel_linear[0].ToString("F02") + "    " +
rov.navigation.Accel_linear[1].ToString("F02") + "    " +
rov.navigation.Accel_linear[2].ToString("F02");
```

Onde o *AceleracaoLinearLabelText* representa a variável do lugar no painel e os termos após a igualdade é o que aparece na tela. Existe o vetor *rov.navigation.Accell_linear*, com 3 valores, que representam as acelerações lineares no eixo x, y e z. O comando *.ToString("F02")* representa a passagem desses valores para *strings*, que são a representação destes em carácteres de fácil compreensão para o ser humano. " " representa o espaço. Dessa maneira, o exemplo do código mostra no painel os 3 valores da aceleração linear em x, y e z com espaço entre eles.

O sistema de Orientação do ROV é determinado por uma Bússola Giroscópica, também conhecida como Girocompasso, semelhante a um Giroscópio. Este mecanismo consiste de um rotor que gira livremente em um sistema de eixos. Ele indica o Norte Verdadeiro utilizando eletricidade.

Para os sensores comentados foram feitos os mesmos comandos da aceleração, construindo dessa forma o novo painel. Além disso, adicionou-se certos parâmetros para facilitar os experimentos como tempo e um botão de *stop* geral, para casos de emergência.

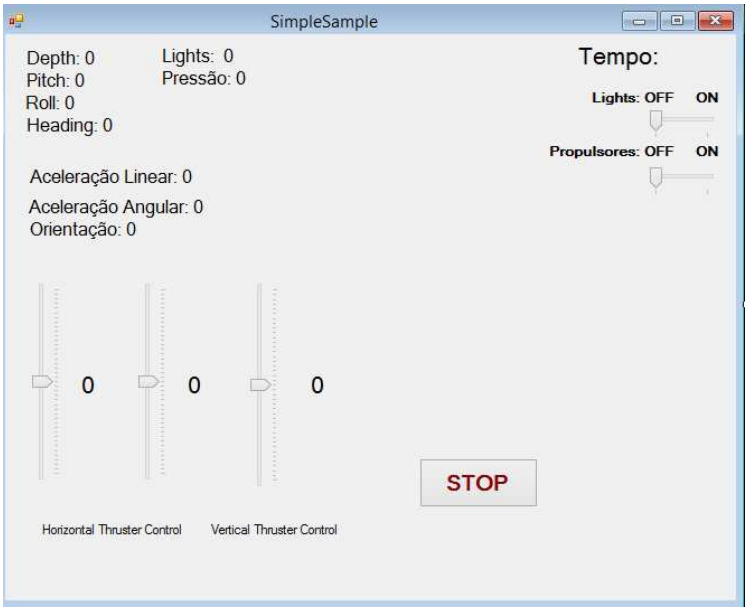


Figura 4.1: Painel Modificado.

Com essas melhorias no painel, pode-se fazer alguns experimentos para comprovar a possibilidade de controle no ROV e acertar alguns parâmetros. Houve a necessidade de se analisar os dados após a realização dos experimentos, para posterior utilização dos mesmos para checar se ocorreu tudo de forma correta e para construção de gráficos. Com isso, foi implementado a função *AnotaDados*, que salva todos os dados em arquivos *.txt*, com todas as respostas em um único arquivo e também em arquivos separados.

```
private void AnotaDados(int PropHori, int Propulsao)
{
        StreamWriter wr = new StreamWriter(@"C:\Users\UNB\SaidaROV.txt"
        , true);
        wr.WriteLine("Acelereação Linear: " + AceleracaoLinear.ToString
        ("F02") + "s");
        StreamWriter wr = new StreamWriter(@"C:\Users\UNB\SaidaROV.txt",
         true);
        wr.WriteLine(Tempo.ToString("F02"));
        wr.Close();
}
```

4.3 Identificação dos dados de leitura do sensores

No primeiro teste observou-se a leitura dos dados quando o robô não está em água e está parado. Na tabela 5.1, têm-se as medidas dos sensores: Tendo em vista que as medidas dos sensores *Depth*

Tabela 4.1: Leitura sensores painel

Sensores	Medidas
Depth	0.00
Pitch	$-1.5\,^{\circ}$
Roll	$3.9\,^{\circ}$
Heading	$89.6\,^{\circ}$
Pressão	0.00
Aceleração Linear	26.00 67.00 985.00 cm/s^2
Aceleração Angular	-16088.00 -23088.00 -17088.00
Orientação	$3.900\,^{\circ}$ $-1.500\,^{\circ}$ $89.600\ ^{\circ}$

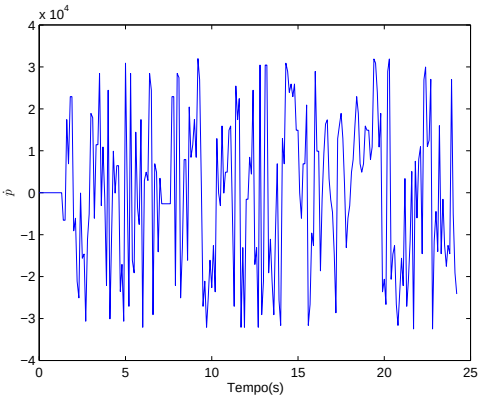
(profundidade) e Pressão são iguais a zero, conclui-se que a pressão medida é a hidrostática, já que a mesma é linearmente dependente da profundidade.

A leitura dos sensores *Pitch* e *Roll* deveria ser zero, pois não há rotação em Y e X. Porém, pode se concluir que ao realizar o experimento, o veículo estava inclinado devido a imperfeição do solo, o que não foi notado pelos aparelhos utilizados para medir a inclinação. Além disso, o dispositivo de medida não era confiável, sendo possível que os valores encontrados estejam com erros associados.

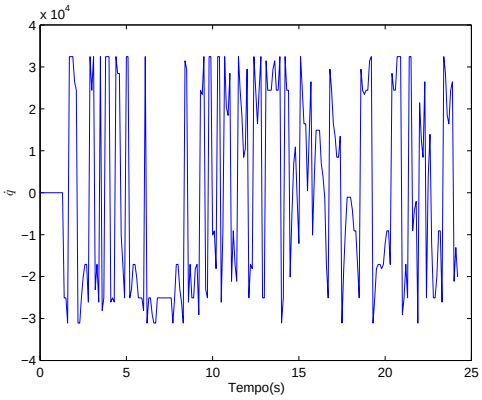
O sensor *Heading* é a rotação no eixo Z, comumente chamado de guinada, e sua medida está relacionada com a bússola, o Norte é referente ao zero. Ao rotacionar o robô, foi percebido que algumas medidas de sensores ao dar um giro de 360 graus, indicavam 3600 de leitura, o que pode se concluir é que os valores dos sensores estão multiplicados por 10.

A aceleração linear no eixo Z é igual a 985, e sabe-se que nesse eixo o robô sofre ação da gravidade, que mede aproximadamente 9.8 m/s^2, inferindo-se então que as medidas de aceleração estão em centímetro por segundo ao quadrado (cm/s^2). As outras medidas de aceleração também dependem da inclinação do robô, com isso, se o robô estiver um pouco inclinado, os outros eixos sofrem também com a ação da gravidade.

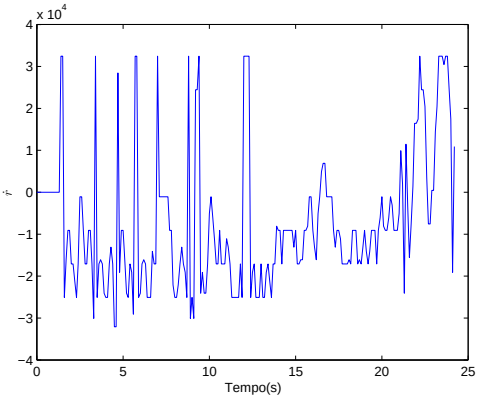
Após a verificação dos dados de aceleração linear, foi começada a análise dos dados de aceleração angular. Foram colhidos dados de leitura do sensor, quando o ROV estava submerso em movimento de avanço, com os dois propulsores horizontais ligados com a mesma intensidade. Seguem os gráficos, na figura 4.2, nos eixos X, Y e Z:



(a) Em X.



(b) Em Y.

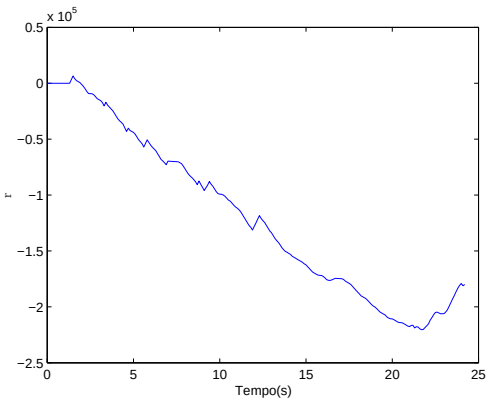


(c) Em Z.

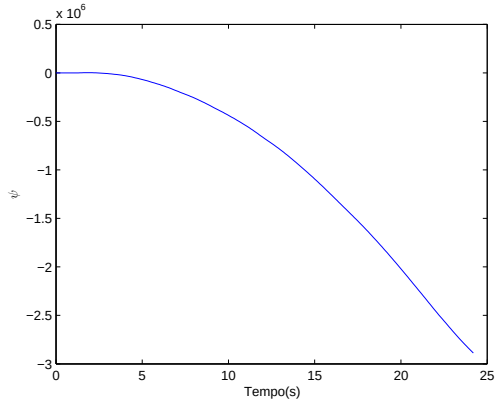
Figura 4.2: Leitura do sensor de Aceleração Angular

Constatou-se que os dados foram muito ruidosos, por isso ficou impossível qualquer análise no gráfico apresentado. Sendo assim, foi iniciado uma série de testes para identificar a adversidade.

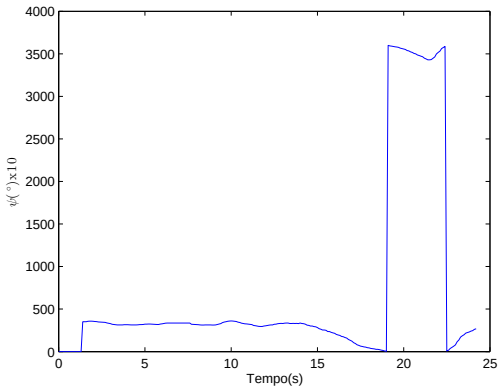
Foi pertinente analisar e identificar a unidade de medida dessa variável. Um sensor que pode ser relacionado com a aceleração angular é a orientação, que é uma representação do deslocamento angular. A integral da aceleração é a velocidade, e a integral da velocidade é o deslocamento, isto é, por meio de operações matemáticas pode-se chegar a dois gráficos que representam o mesmo valor. Como os dados de Orientação em Z já foram testados e justificados, é viável identificar a integridade dos valores gerados pelos sensores de aceleração angular. Sendo assim, integrou-se duas vezes os gráficos de aceleração angular, figura 4.3. Como os dados de orientação em Z são os mais compreensíveis, analisou-se os valores de aceleração angular apenas nesse eixo.



(a) Primeira Integral da Aceleração Angular.



(b) Segunda Integral da Aceleração Angular.

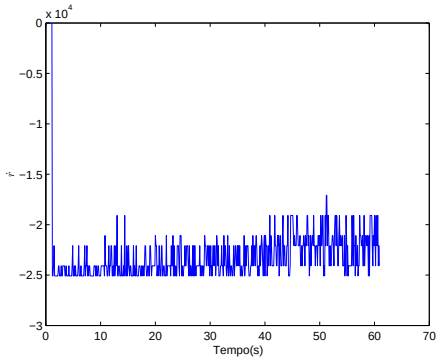


(c) Orientação.

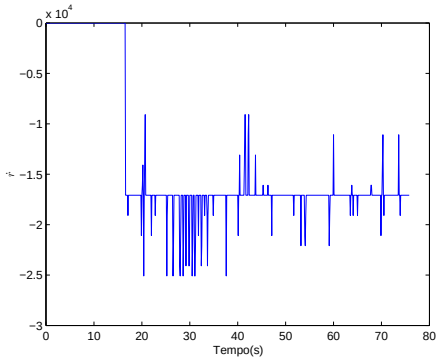
Figura 4.3: Comparação entre Aceleração Angular e Orientação em Z

Nota-se que os valores de orientação e aceleração angular em Z integrada duas vezes não possuem relação. Desta forma, é necessária uma análise mais profunda dos ruídos que estão nos sensores dos robôs, por isso, são feitos novos testes, mais primários, para distingui-los.

A princípio, foram analisados dois dados simplificados. O primeiro, a partir de um teste já feito anteriormente, do veículo ligado em local seco, sem movimentação. No segundo, a partir de um novo teste, onde o robô foi colocado na água sem nenhum valor de propulsão. Além de tentar identificar algum padrão na aceleração angular, também foi possível utilizar esse teste para identificar ruídos não atrelados às perturbações da piscina nos outros valores aferidos. Assim, foi viável a identificação dos ruídos causados pela perturbação da água. Os dados encontrados estão na figura 4.4:



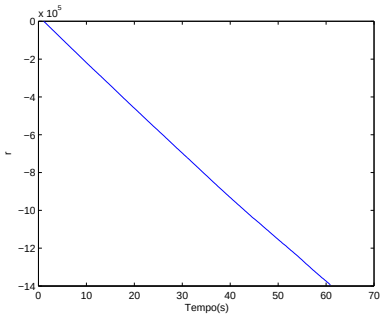
(a) Experimento a seco.



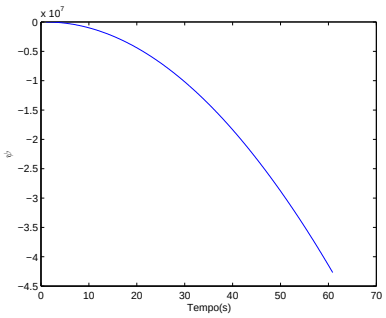
(b) Experimento na água.

Figura 4.4: Aceleração Angular em Z

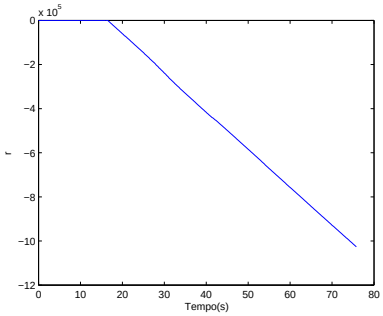
Mesmo com o veículo parado, os dados de aceleração angular não melhoraram e continuaram impossíveis de serem entendidos. Com a esperança de que a simplificação no experimento pudesse esclarecer alguns questionamentos, a partir do mesmo método utilizado anteriormente, integra-se duas vezes o valor de aceleração angular em Z, figura 4.5:



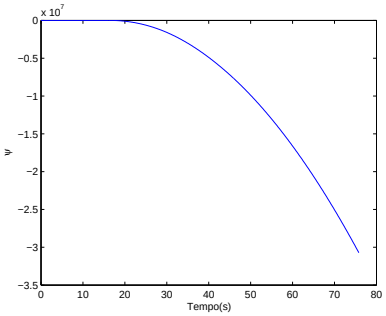
(a) Primeira Integral da Aceleração Angular a Seco.



(b) Segunda Integral da Aceleração Angular a Seco.



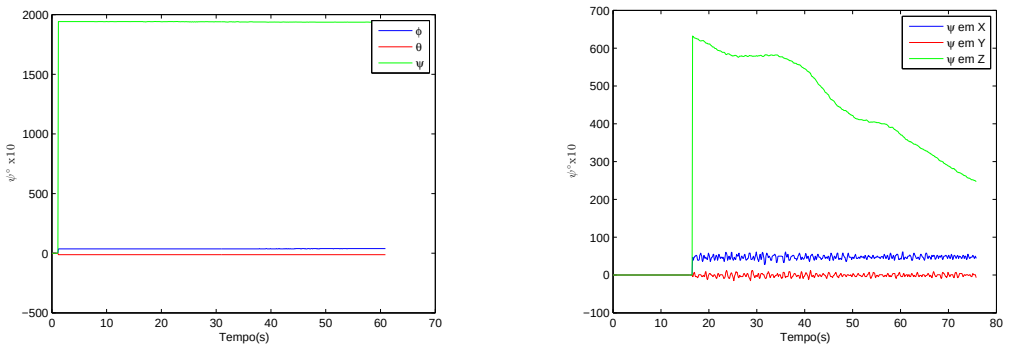
(c) Primeira Integral da Aceleração Angular na Agua.



(d) Segunda Integral da Aceleração Angular na Agua.

Figura 4.5: Integrações da Aceleração Angular

Os dois gráficos, figura 4.5, estão muito parecidos, o que representaria um movimento igual em relação ao eixo Z nas duas situações. Evidentemente, na água o movimento do robô não é mesmo no experimento a seco, o que pode implicar em uma danificação dos sensores de aceleração angular. Para comprovar essa suspeita, verificou-se os movimentos de orientação em Z nas duas situações, observado na figura 4.6:



(a) Experimento a seco.

(b) Experimento na água.

Figura 4.6: Orientação Angular em Z

Claramente o ROV apresenta um movimento giratório por causa das perturbações da água, havendo uma modificação na orientação em Z, o que está sendo apresentado de forma totalmente oposta nos gráficos integrados da aceleração angular. Isso atesta que ou há defeito nos valores aferidos do sensor de aceleração angular ou é necessário um tratamento mais sofisticado para interpretação desses dados. Uma tentativa possível de distinguir esses dados é a passagem de filtro pelo sinal, devido à suspeita de ruídos. Para isso, fez-se necessário o uso de um processamento digital de sinais.

Após uma análise e pesquisa sobre o assunto, foi considerado o filtro de média móvel. Ele é o mais comum em processamento digital de sinais, principalmente por ser entre esses tipos de filtro o mais fácil de se entender e de usar. Apesar da sua simplicidade, o média móvel é ótimo para uma tarefa comum, a de reduzir ruídos aleatórios enquanto retém uma resposta ao degrau nítida. Isso faz dele a principal escolha para sinais codificados no domínio do tempo [9]. Como no momento o objetivo é filtrar uma resposta ao degrau no domínio do tempo, nota-se que pela simplicidade esse filtro é o mais recomendado. Se fosse um sinal codificado no domínio da frequência, o média móvel não seria recomendado, pois ele não consegue separar uma banda passante de outra.

O filtro escolhido é obtido calculando-se a média de um conjunto de valores, onde esse conjunto tem o tamanho de uma janela de amostras escolhida. Ocorrem várias interações para se filtrar o sinal, sempre se adicionando o próximo valor do sinal ao conjunto e se descartando o mais velho da média.

$$y[n] = \frac{1}{N+1} \sum_{k=0}^{N} x[n-k] \tag{4.1}$$

onde n é o índice dos vetores utilizados (tempo atual), $N+1$ é o tamanho do conjunto dos valores, $y[n]$ é o sinal filtrado e $x[n-k]$ representa o conjunto dos valores a serem somados. Existe também

a possibilidade de implementar o filtro de média móvel por uma função recursiva. Porém, neste trabalho, não é vital a filtragem em tempo real. Os dados são tratados no final. Portanto, a fórmula sem recursividade é suficiente.

As saídas de Aceleração Angular filtradas não mudaram em nada. A não mudança de nenhum valor do sinal, mesmo com diferentes tamanhos de números de amostras, prova finalmente a não funcionalidade dos sensores. Por isso, para fins de projeto, essas acelerações vão ser desconsideradas.

Como foi comentado anteriormente, os dados simplificados a seco e na água também podem trazer benefícios para as outras medidas dos sensores. Com base nos ruídos reconhecidos, é possível subtrair esses valores nos dados adquiridos em outros procedimentos e refinar as respostas. Outra consequência dos testes simplificados foi a evidência de que nos experimentos de Orientação a seco, na figura 4.6(a), não houve um ruído considerável. Para confirmar essa indicação, se faz necessário a análise do último valor aferido no experimento, a Aceleração Linear, figura 4.7.

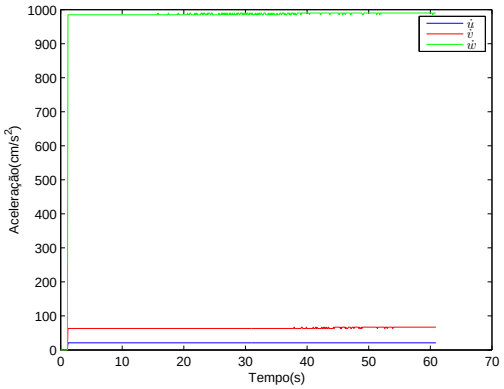
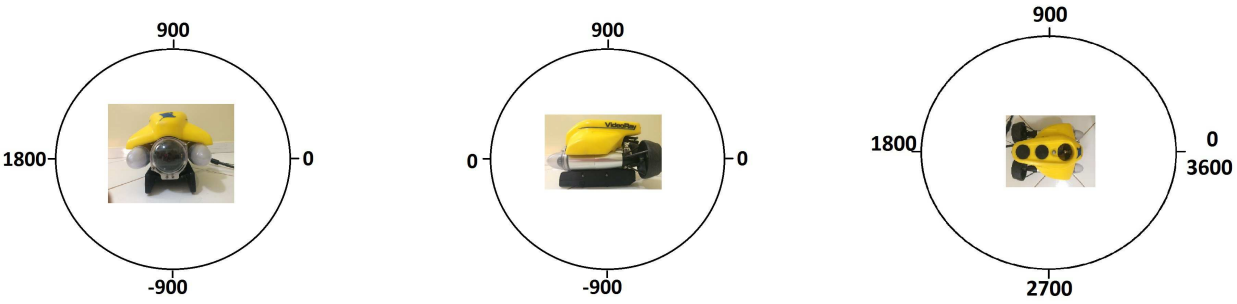


Figura 4.7: Aceleração a Seco.

Esse gráfico, figura 4.7, representa a comprovação de que os dados aferidos não sofrem ruídos de medição consideráveis. As respostas da aceleração sofrem um degrau nos segundos iniciais, devido ao atraso do programa em fazer a leitura dos dados dos sensores. Assim, em outros experimentos, para aprimorar a resposta, apenas o erro acarretado pelas perturbações da água será tratado.

Do mesmo modo que foi feita a tentativa com a aceleração angular, foi relevante determinar a unidade de medida de todos os valores de orientação e aceleração linear. Com essa unidade definida, pode-se analisar melhor os resultados e encontrar um sentido nos testes. É interessante distinguir alguns vínculos para checar alguma falha dos dispositivos. Por exemplo, as medidas de orientação seguem os valores de *Roll*, *Pitch* e *Heading*, indicando a direção e o sentido que o ROV está voltado.

Para identificar as medidas de orientação do veículo, gerou-se um teste no qual eram feitas rotações no robô, com a meta de determinar o que cada posição indica e qual valor está associado a este lugar nos sensores. Uma representação das posições e valores correspondentes é exibido a seguir na figura 4.8:



(a) *Roll*. (b) *Pitch*. (c) *Heading*.

Figura 4.8: Representação das medidas do sensor

É possível notar que, na figura 4.8 (a), a posição que o robô está representa a posição zero. Se rotacioná-lo 90 graus, a variável *Roll* teria o valor de 900. Se rotacioná-lo 180 graus teria 1800. As figuras 4.8(b) e 4.8(c) seguem a mesma lógica.

Após conhecer os valores das variáveis que o ROV pode apresentar, houve a necessidade de afirmar, a partir de um experimento, se era possível setar comando aos propulsores e luzes com a mudanças dos valores dos sensores, como proposto no objetivo do trabalho. Este procedimento determinava uma mudança da intensidade das luzes de acordo com a mudança da leitura do sensor de arfagem do submarino (*Pitch*), através de uma equação linear. Como foi visto anteriormente, os valores de *Pitch* variam de 900 a -900. A intensidade da luz varia de 0 a 1, porém no experimento, para não correr o risco de danificar as luzes, não foi usado o valor máximo de intensidade, apenas a metade. O objetivo era encontrar uma equação que representasse essa dependência de forma simples. A equação escolhida foi a linear, portanto: $y = A * x + B$. Os pontos da reta desejada são: (0.5, -900), (0.25, 0) e (0, 900). Calculando, têm-se a equação aproximada:

$$\text{Intensidade da luz} = -0.0003 * Pitch + 0.25$$

(a) 900 de *Pitch* e 0 de intensidade de luz. (b) 450 de *Pitch* e 0.115 de intensidade de luz.

Figura 4.9: Intensidade luz X *Pitch*

O experimento como visto nas figura 4.9 deu certo, então foi possível começar a implementação

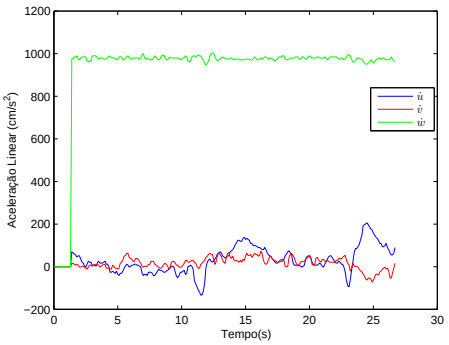
do sistema de controle.

4.4 Aplicação de Degraus nos Propulsores

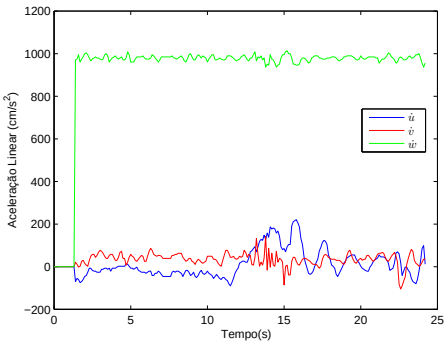
Foram realizados testes aplicando uma entrada Degrau nos propulsores para analisar a resposta dos sensores quanto às perturbações do meio. Para ter uma melhor especificação da planta, implementou-se vários tipos de degraus, em tempos diferentes, para se obter corretamente os parâmetros.

Dois testes foram feitos para adquirir a leitura dos sensores. Nos dois, utilizou-se a rotina no código que guardava os dados dos sensores em arquivos de texto. Após os experimentos, notou-se que algumas realizações não tinham sido validadas, tanto por conta das perturbações nas piscinas onde foram realizados os experimentos, tanto por não serem perceptíveis as modificações das respostas quando testado com diferentes propulsões. Por exemplo, colocando um valor de 25 ou 35 na propulsão horizontal não houve alteração visível do comportamento do ROV.

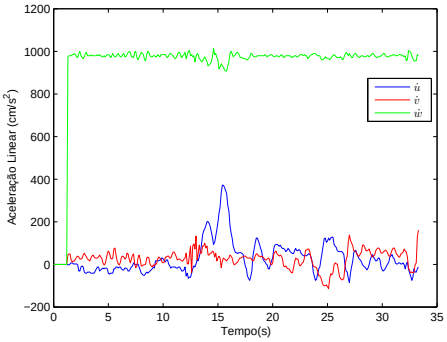
Dessa maneira, apenas em alguns testes obteve-se um resultado positivo. Além disso, como já atestado anteriormente, nenhum dos valores obtidos de aceleração angular é plausível. Seguem alguns gráficos encontrados, figura 4.10:



(a) Com 25.



(b) Com 35.



(c) Com 50.

Figura 4.10: Aceleração linear para diferentes propulsões.

Analisando os gráficos da figura 4.10 conclui-se que: a saída é bem ruidosa, devido ao fato de ter perturbações na água e que a aceleração em Z mantém-se na faixa de 980. Isso prova a integridade dos dados, tendo em vista que a gravidade é de aproximadamente 9.8 m/s^2 e em direção do eixo Z. A propulsão comandada nos dois propulsores horizontais foi para o movimento de avanço submerso, então a mudança de resposta que deve ser analisada é a do eixo X, sinalizada por vermelho nos gráficos. A ligação da propulsão dos motores só começava a partir de 10 segundos após a inicialização do programa e mantinha o valor do degrau durante 5 segundos.

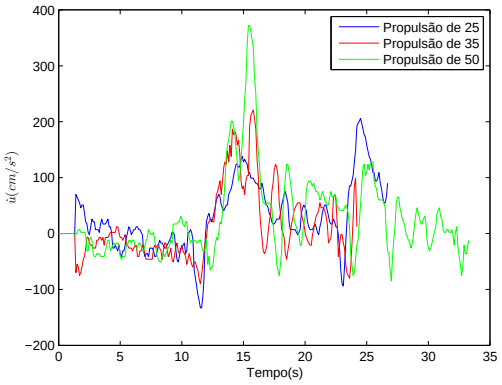


Figura 4.11: Acelerações Lineares em X com 25, 35 e 50.

Como já visto previamente, as respostas do sistema às entradas de 25 e 35 não possuem muita diferença. Isso é comprovado graficamente, figura 4.11, onde a resposta acaba se confundindo com os ruídos, sendo impossível notar a diferença. Porém, a diferença entre valores de propulsão com maior intensidade e os dois de menor é bem perceptível. O movimento deveria começar nos 10 segundos, entretanto a primeira reação vista nos gráficos é uma aceleração negativa. Isso acontece pois o propulsor estava parado e, ao ser ligado em uma potência alta, acarreta uma reação no sentido inverso ao movimento do propulsor, devido a inércia. Após o fim da inércia a aceleração cresce normalmente.

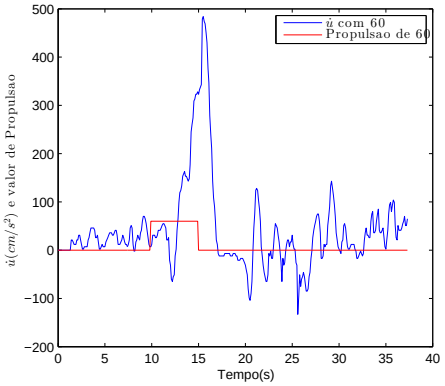


Figura 4.12: Ensaio de propulsão horizontal com 60.

Em uma propulsão ainda maior, de 60, observa-se que a aceleração linear chegou a mais de $4m/s^2$, figura 4.12.

Com esses valores tentou-se encontrar uma equação que poderia reger esses parâmetros. Apesar de tentar uma aproximação, não foi possível achar tal equação, visto a complexidade do sistema e os ruídos associados, figura 4.13.

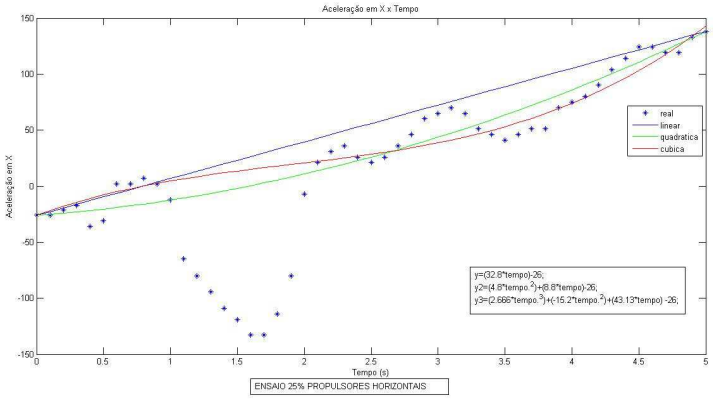
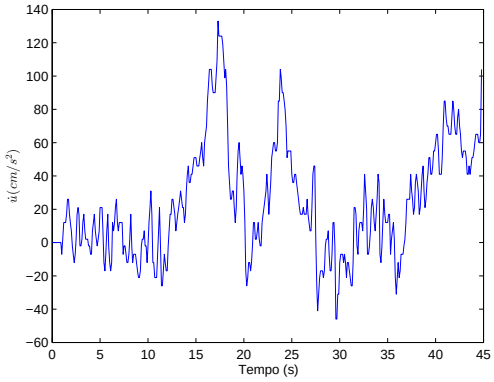


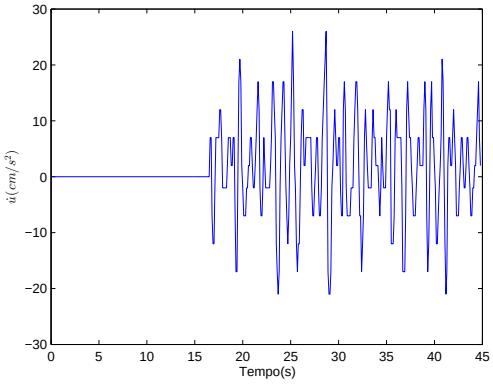
Figura 4.13: Tentativas de aproximação da curva de aceleração.

Fundamentado nessas curvas, figura 4.13, comprova-se a não acurácia desse método de aproximação de curva. Assim, é necessário fazer um estudo aprofundado sobre a identificação dos parâmetros da planta. Uma das opções seria gerar novos experimentos e, utilizando uma ferramenta do Matlab, o *System Identification Tool*, adquirir a função de transferência da planta. A partir dela pode ser possível prever todo o comportamento do sistema.

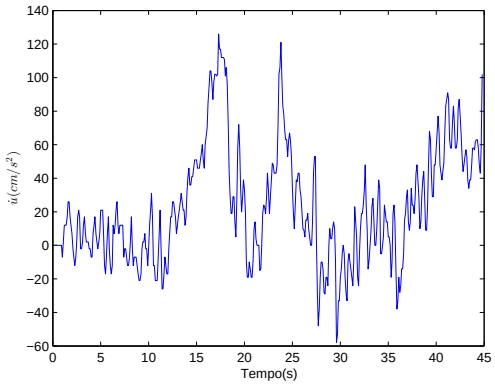
Com esses resultados vagos e sabendo que a saída é muito ruidosa, se encontra a necessidade de refiná-la para obter respostas contundentes de aceleração e orientação. A perturbação devido à água já foi identificada utilizando o gráfico aferido apenas com essa perturbação, foram feitas diferentes operações matemáticas na tentativa de filtrar a resposta. Uma das manobras possíveis foi subtrair o valor do ruído da água do sinal sendo analisado, porém não se pode afirmar que as mudanças das ondas da piscina aconteceram no mesmo instante nos dois experimentos. Outra operação possível é a média dos valores de ruído e a subtração do sinal investigado. Novamente, é essencial uma representação em gráficos para aprovar tais métodos de refinamento mostrado na figura 4.14.



(a) Aceleração Linear em X.



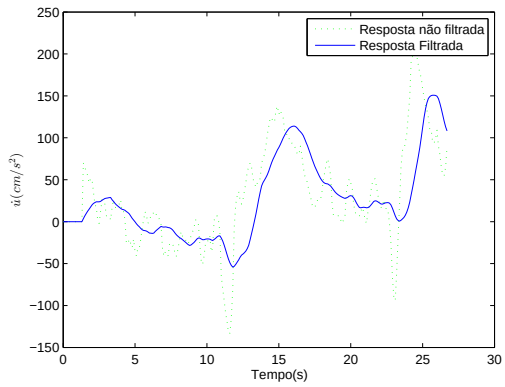
(b) O ruído da Aceleração em X em repouso na água.



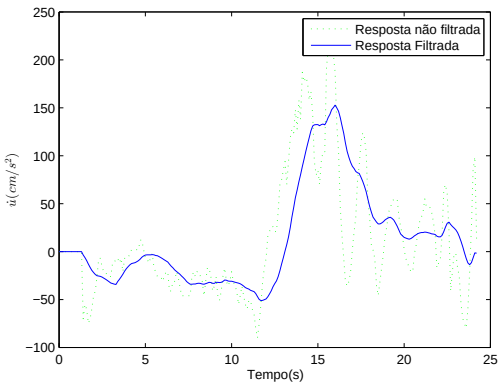
(c) Subtração dos Sinais

Figura 4.14: Tentativa de refinamento de sinal

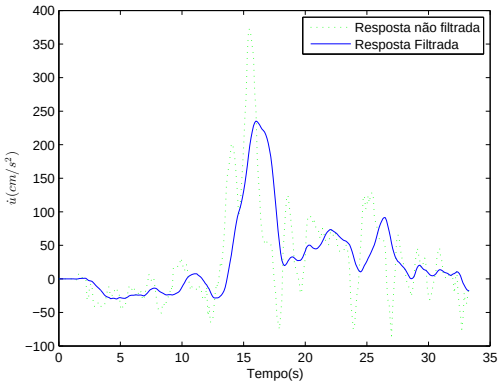
Considerando os resultados da figura 4.14, a subtração do sinal analisado com o ruído da água não alterou em nada a saída. Conclui-se que não é possível refinar a resposta dessa maneira, pois no experimento parado na água, o submarino não realizou nenhuma força ou impulso, o que não acarretou em forças de reação maiores. Já em um experimento onde ocorre propulsão, a força de reação gerada aliada a colisões da água na parede da piscina aumentam a perturbação, deixando os dados de saída ruins para inspeção. Outro modo já comentado seria a média do erro, mas esse método apenas deslocaria o sinal analisado no eixo Y. Sendo assim, a única configuração possível é a construção de um filtro para a retirada desse ruído. Como visto na seção 4.3, o filtro de média móvel é uma boa opção para a situação. Seguem os resultados de filtragem de média móvel com número de amostras de 20 para a aceleração linear em X, na figura 4.15.



(a) Com 25.



(b) Com 35.



(c) Com 50.

Figura 4.15: Aceleração linear para diferentes propulsões.

Constatou-se uma melhora evidente nas saídas, sendo possível agora identificar os máximos e mínimos do sinal. Nota-se a diferença nas acelerações e o dimensionamento correto para as seguintes propulsões, o que não era possível nos sinais sem filtro. Admite-se que o filtro de médias móveis foi um sucesso para gráfico de aceleração linear.

4.5 Rotina da Câmera

4.5.1 Identificação da câmera, visualização no painel e gravação na memória

Um dos parâmetros mais importantes para operação autônoma do robô é a sua posição e orientação. Com ela, o robô tem a direção, o sentido e a velocidade necessária para se mover para qualquer outra posição e realizar suas operações. Um dos modos de saber a posição é a utilização de um giroscópio. Tal ferramenta, aliada a alguns valores de referência, consegue detalhar ao robô a sua posição, direção e sentido. O VideoRay Pro 4 possui um giroscópio de 3 eixos [Video Ray Pro 4 Manual, 2010] o qual consegue detalhar tudo ao robô. Porém, como são sensores, existem erros de medida associados, que em um regime estacionário, acabam associando valores totalmente errôneos de posição. Para tentar evitar isso, é necessária uma outra ferramenta na qual pode-se

checar a posição e compensar se houve algum erro associado no giroscópio.

Uma ferramenta disponível que pode ser utilizada é a câmera do robô. Com isso, foi inserido, no programa modificável, a câmera e sua visualização no painel. O dispositivo da câmera do robô é reconhecido pelo computador através de um cabo USB, o que facilita a programação. Dessa maneira, utilizou-se uma biblioteca já configurada para recuperar imagens de câmeras conectadas por um cabo USB. Após pesquisas, foi notada uma biblioteca chamada *AForge.NET*. É uma biblioteca atual, lançada em julho de 2013 e que possui inúmeras funcionalidades para a câmera.[1]. Porém, dessa biblioteca utilizou-se apenas o reconhecimento da câmera, a visualização no painel e a gravação das imagens na memória do computador, para uma análise posterior.

Para visualizar a imagem da câmera no painel, foi adicionada uma nova forma ao painel. Essa forma tem o nome de *picturebox*, que é um componente que mostra imagens e vídeos no painel. Após isso, foi referenciado a biblioteca *AForge.NET* ao programa. Para isso, utilizou-se as bibliotecas de vínculo dinâmico, chamadas *DLLs*.[2]. As bibliotecas utilizadas para essa parte da programação são *AForge.Video.dll* e *AForge.Video.DirectShow.dll*. Estas duas *DLLs* são ligadas a identificação da câmera, a visualização no painel e a gravação das imagens na memória. Após isso, deve-se direcionar cada classe para o seu parâmetro. A câmera USB é automaticamente identificada e a forma *picturebox* tem que ser atualizada a cada nova interação da câmera do robô. É necessário a descrição do caminho onde a imagem vai ser salva no computador e o formato de codificação de imagem desejado, como por exemplo JPEG ou PNG.

(a) Em água. (b) A seco.

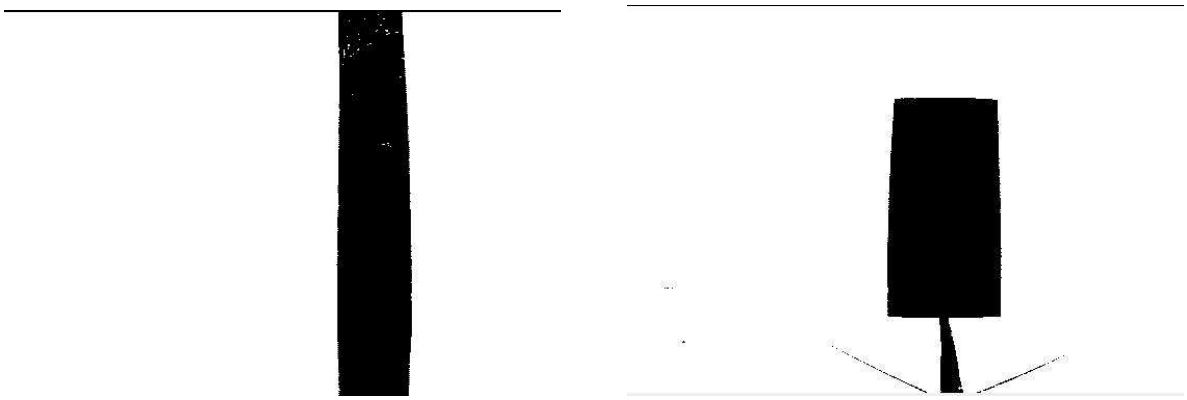
Figura 4.16: Imagens salvas pelas camêra

4.5.2 Tratamento da imagem

Com a posse das imagens, agora deve-se adquirir as informações destas, ou seja, transformá-las em valores, para serem utilizadas em lógicas matemáticas. Neste trabalho, foi importante conhecer, por exemplo, o tamanho de algum objeto. Analisando em prospecção de petróleo, ter o tamanho de um ducto. Para isso, foi preciso tratar uma imagem.

[1]Fonte: aforgenet.com/framework/downloads.html
[2]Fonte: https://support.microsoft.com/pt-br/kb/815065

Uma imagem, em programação, é considerada uma matriz. Cada elemento da matriz possui um valor que representa uma cor. Dessa forma, para caracterizar um objeto, é necessário identificar uma parte de uma matriz uniforme e de mesma cor. Ao considerar que existem muitos tons diferentes de uma cor em um objeto, concluiu-se que, para caracterizar os objetos de forma mais simples, deve-se deixar a imagem em preto e branco (binarizar). Para as cores serem passadas para preto ou branco depende de um limiar *threshold* [Gonzalez, 2007]. Esse limiar tem que ser escolhido de forma que, o fundo seja branco e o objeto a ser analisado seja preto. A escolha desse limiar é determinante no sucesso do tratamento da imagem.



(a) Em água. (b) A seco.

Figura 4.17: Imagens em preto e branco.

Para o primeiro passo, deve-se binarizar a imagem da câmera. A cada interação de um laço, analisa-se o valor de cada elemento da matriz e através do limiar determinado, esse valor se transforma em um elemento que representa preto ou branco. Assim obtém-se a imagem binarizada. Após isso, é preciso adquirir as informações da imagem. Sabendo que cada elemento da matriz representa um *pixel* da imagem, pode-se contar os elementos da matriz, para adquirir o tamanho do objeto na imagem em *pixels*. Como a imagem tratada possui os *pixels* que foram analisados na cor preta, então foi apenas foi feita a contagem destes. A partir do valor de *pixels* que um objeto possui, é descoberta algumas informações. Se é conhecida a distância até o objeto, é possível calcular o tamanho deste no Sistema Internacional de Medidas. Já se o tamanho do objeto é conhecido, é possível calcular a distância até ele no mesmo sistema. Adicionalmente, em uma linha da matriz, se forem contados os *pixels* em sequência de cor preta, adquire-se o comprimento do objeto em relação a tal linha.

Apesar de ser facilmente explicável, existe um grande problema de fazer isso em lógica de programação. Ao analisar cada elemento de uma matriz, levando em conta que a imagem adquirida pelo ROV é de 640 por 480 *pixels*, são necessárias 307200 interações para apenas binarizar a imagem, e para adquirir o tamanho do objeto em *pixels* seriam mais outras 307200 interações. Para obter o comprimento de um objeto em apenas uma linha seriam 640 interações. O número elevado de interações torna o programa lento. Como o objetivo inicial seria a operação do ROV em tempo real, foram estudadas maneiras de diminuir o tempo de processamento.

Analisa-se uma situação especifica de manutenção preventiva de um cano em posição vertical, como na figura 4.18:

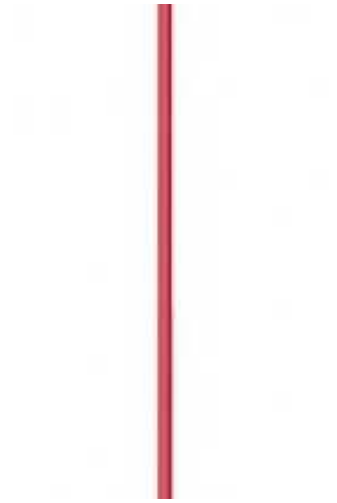


Figura 4.18: Representação de um cano na vertical.

Para a manutenção preventiva citada na figura 4.18, o robô teria que inspecionar todo o cano de uma mesma posição. Para isso, seria necessário que o ROV continuasse a uma mesma distância do cano enquanto faz a sua verificação. Por isso, é importante conhecer a distância do cano até a câmera. Tendo em mãos uma tabela que contém o número de *pixels* e a respectiva distância para um tamanho de cano especifico, identifica-se a distância do objeto até o ROV. Além do giroscópio mostrando a posição, direção e sentido do veículo, por essa contagem de *pixels*, é possível conhecer de outra forma a posição do veículo. Utilizando essas duas informações juntas, diminui-se os erros estacionários. Além disso, nesse exemplo específico, existe uma maneira de diminuir bastante o processamento e a lentidão da parte computacional. Como o cano estará na posição vertical, admitindo que o *pitch* do ROV tem valor 0, todas as linhas da imagem vão ter o mesmo número de *pixels* do cano, ou seja, em toda a imagem, o comprimento em *pixels* do cano vai ser o mesmo em todas linha. Sendo assim, ao invés de realizar 307200 interações para obter o comprimento do cano, realiza-se apenas 640 interações. Desta forma, binarizando toda a imagem e contando o comprimento do objeto em apenas uma linha estão acontecendo 307840 interações.

Para melhor análise, foram feitos testes para determinar os atrasos ocorridos ao processar a imagem.

Tabela 4.2: Atraso devido ao Processamento de Imagem do Programa

Frequência	Atraso por Minuto
1 em 1 segundo	32 segundos
2 em 2 segundos	24 segundos
3 em 3 segundos	20 segundos
5 em 5 segundos	14 segundos

Como pode ser visto na tabela 4.2, se o tratamento de imagem acontecer de 1 em 1 segundo

resulta em um atraso de 32 segundos a cada minuto, o que em uma hora acarretaria em um atraso de 1920 segundos. Analisando o caso de 5 em 5 segundos, o atraso em uma hora seria de 840, ou seja, menor, porém ainda muito grande.

Em um trabalho no qual o objetivo é um processamento em tempo real, não se pode haver atrasos tão grandes. Enquanto há esse tipo de erro, a programação do robô não consegue compensar os distúrbios através do controle corretamente. Se mostra indispensável uma nova otimização.

O novo aprimoramento consiste em apenas binarizar a linha que será contada. Isso representa 99,79% menos interações, o que demonstra uma excelente otimização.

(a) Em água.

(b) A seco.

Figura 4.19: Imagem tratada apenas na linha do meio.

Realizou-se um teste para confirmar se o atraso realmente diminuiu ao binarizar uma única linha da imagem. O resultado encontrado foi muito satisfatório, os atrasos encontrados ao tratar a imagem foram de 4 segundos por minuto. Lembrando que esse atraso pode ser menor que quatro ou próximo de zero, levando em conta os erros experimentais de medição.

Tabela 4.3: Atraso devido ao Processamento de Imagem do Programa

Frequência	Atraso por Minuto
1 em 1 segundo	4 segundos
2 em 2 segundos	4 segundos
3 em 3 segundos	4 segundos
5 em 5 segundos	4 segundos

Com esses resultados, devido a necessidade de processamento em tempo real, decidiu-se que o controle e aquisição de valores pela câmera seriam feitos apenas com objetos na forma de cano. Aferir as medidas ou até apenas detectar outros objetos se mostra inviável com o atraso acarretado pelo processamento da imagem.

Capítulo 5

Implementação do Sistema de Controle

5.1 Introdução

Como o objetivo do projeto é que o ROV se torne autônomo em certas atividades, então é preciso implementar uma teoria de controle que se aplique aos movimentos do mesmo. O funcionamento do controle é baseado na compensação dos distúrbios causados pelo ambiente por meio dso propulsores. O sistema de controle de movimento é usualmente constituído por três blocos independentes: Orientação, Navegação e Controle (GNC - *Guidance, Navigation and Control*). Esses três sistemas interagem entre si através de dados e sinais de transmissão [10].

Por exemplo, o ROV está localizado a uma certa orientação e distância de um cano e precisa seguir em linha reta até ele, através de uma potência setada nos propulsores. Porém se uma correnteza tirá-lo de sua trajetória, os próprios propulsores mudarão suas potências para que o robô retorne a sua posição. Será implementado um controle utilizando a imagem da câmera como entrada para o sistema. Neste capítulo descreve-se o controle utilizado e também como o controle foi implementado.

5.2 Sistema de Controle

Para um sistema de seis graus de liberdade e com combinações de movimentos e magnitudes de velocidade, fica nítido a complexidade do projeto. Uma estratégia muito utilizado nesses tipos de sistemas é considerar que o sistema é desacoplado, ou seja SISO (*Single Input Single Output*), com isso para um sistema com seis de graus de liberdade, existem seis equações diferenciais desacopladas e a dinâmica de uma equação não interfere na outra [3].

Apesar do comportamento não-linear na dinâmica de veículos submarinos, é justificável o uso de controladores descentralizados quando o acoplamento entre os graus de liberdade pouco influencia em sua dinâmica, podendo ser caracterizado por um modelo linear. Para baixas velocidades, situação usualmente presente quando existe a análise do posicionamento dinâmico, este comportamento linear é encontrado. Com isso, é possível reformular um sistema de controle com uma faixa

de operação desconsiderando as não-linearidades [3].

Para representar o modelo linear da planta manipula-se a equação (2.21). Os termos não-lineares dessa equação serão desconsiderados, representados pela matriz de Coriólis, C_{RB}. O termo τ_{RB} como visto na equação (2.23), é constituído de forças e momentos hidrodinâmicos, do ambiente e da propulsão. Os esforços do ambiente também serão desconsiderados, já que são não-lineares. Desta forma, τ_{RB} é definido como:

$$\tau_{RB} = \tau_h + \tau \tag{5.1}$$

A nova equação que rege o movimento da planta é definida como:

$$M_{RB}\dot{V} = \tau_h + \tau \tag{5.2}$$

Os esforços hidrodinâmicos, como visto na equação (2.27), dependem linearmente da velocidade. Sendo assim, podem ser representados por uma constante (k') multiplicada pela velocidade. Como o arrasto remove energia do sistema, então o seu sinal é negativo. Para fins de controle, a equação (5.2) será em função do deslocamento, nomeado de S_d:

$$\ddot{S_d} = -\alpha\dot{S_d} + \beta\tau \tag{5.3}$$

onde $\alpha = \frac{k'}{M_{RB}}$ e $\beta = \frac{1}{M_{RB}}$.

Paralelamente com esse projeto, existe um estudo no mesmo submarino da *Video Ray*, *Controle por planejamento e acompanhamento de trajeória para veículo operado remotamente*, desenvolvido pela graduanda Ana Paulino. Nessa pesquisa, houve uma identificação dos parâmetros do modelo através do método dos mínimos quadrados onde determinou-se o coeficiente β igual a 0.4 [12]. Foi utilizado este valor para o modelo.

Considerou-se a aproximação do sistema para primeira ordem. Sabe-se que para equações de primeira ordem, o coeficiente que acompanha a variável de saída, é o inverso do atraso da resposta. É viável encontrar essa constante graficamente, a partir de uma resposta ao degrau, figura 5.1.

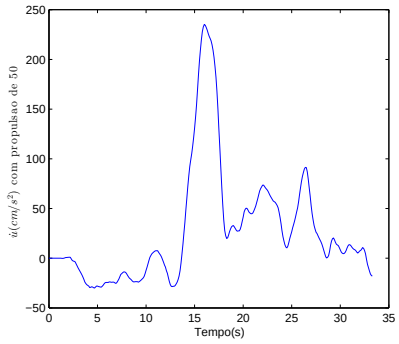


Figura 5.1: Análise do atraso de resposta do Sistema.

Analisando o gráfico, atestou-se que o atraso de resposta é aproximadamente 3 segundos, ou seja, o espaço de tempo entre um envio de sinais de comando aos propulsores e uma aceleração real acontecer, resultando em um $\alpha \cong 0.3$.

Desta forma, é definida a equação da planta:

$$\ddot{S_d} = -0.3\dot{S_d} + 0.4\tau \tag{5.4}$$

onde S_d é medido em metros e τ é o comando dado aos propulsores e não possui unidade de medida.

5.2.1 Controle PID

O PID (Proporcional-Integral-Derivativo) é muito utilizado como estratégia de controle da maioria dos veículos submarinos. Isso se deve ao fato do controle linear PID ser de fácil implementação e de sintonia dos parâmetros do controlador. Em algumas situações, o controle PID não é adequado para veículos submarinos, devido a não linearidade do movimento dos mesmos, com isso um controlador linear não permite um desempenho satisfatório para condições muito diferentes do qual o controlador não foi projetado. Desta forma, são necessários vários controladores, para cada região de operação [3].

Para um sistema de malha fechada, a localização dos polos determina a equação característica da resposta transitória. Os polos de malha fechada são as raízes do equação característica do sistema. Um método para a determinação dessas raízes foi desenvolvido por W.R. Evans e é muito utilizado na engenharia de controle. Este método é o "Lugar das Raízes"que consiste em representar graficamente todos os valores do sistema. Por este método, o projetista pode prever quais os efeitos da variação do valor do ganho ou a da adição de polos e zeros no sistema. Outro método utilizado é a sintonização dos ganhos do controlador para atender às especificações no domínio da frequência [11].

O controle PID é representado desta forma:

$$G_C(s) = K_P + \frac{K_I}{s} + K_D s \tag{5.5}$$

O K_P quando elevado diminui o tempo de subida e o erro estacionário da resposta (mas nunca o elimina). O K_I elimina o erro estacionário e torna a resposta mais oscilatória. O K_D melhora a estabilidade do sistema, reduzindo a sobreelevação e melhorando a resposta transitória.

Para encontrar os coeficientes do controlador, foi utilizado o método de sintonia através da alocação de polos da função de transferência de malha fechada. Para isto, basta igualar os coeficientes do denominador da função de malha fechada com o polinômio característico desejado.

Ele foi escolhido para se ter um coeficiente de amortecimento pequeno. Com isso, determinou-se polos reais e idênticos, pois se não forem reais o sistema será oscilante. O polinômio característico desejado é:

$$E(s) = (s+a)^3 = s^3 + 3a^2 s + 3as^2 + a^3 \tag{5.6}$$

O sistema desejado, então, é:

$$\frac{b}{U_{ref}} = \frac{K(s)}{s^3+3a^2 s+3as^2+a^3}$$

$$bs^3 + 3abs^2 + 3a^2bs + a^3b = U_{ref}K(s) \tag{5.7}$$

A equação da planta foi definida anteriormente. Desta forma, fazendo a transformada de Laplace da equação 5.3:

$$S_d s^2 = -0.3 S_d s + 0.4\tau$$

Com o intuito de relacionar as duas equações, $bs = S_d$:

$$bs^3 = -0.3bs^2 + 0.4\tau$$

Para fins de controle, o termo que acompanha o bs^2 na equação desejada deve ter a mesma proporção do valor de α. O valor escolhido foi 0.6, portanto $a = 0.2$. Igualando as equações:

$$bs^3 = -0.6bs^2 - 0.12bs - 0.008b + U_{ref}K(s)$$

$$bs^3 = -0.3bs^2 + 0.4\tau$$

$$\tau = -0.75bs^2 - 0.3bs - 0.02b + \frac{U_{ref}K(s)}{0.4} \tag{5.8}$$

Nos equacionamentos do controle, o cálculo do proporcional e do integrativo precisa do valor referencial. Por isso, é inferido que o K(s) tem a forma:

$$K(s) = 0.12 + \frac{0.008}{s} \tag{5.9}$$

Substituindo o valor de K(s), substituindo o deslocamento S_d e fazendo a transformada inversa de Laplace, tem-se que:

$$\tau = -0.75\dot{S}_d - 0.3(S_d - U_{ref}) - 0.02(\int S_d - \int U_{ref}) \tag{5.10}$$

Para facilitar a notação na programação, o τ é nomeado como F,que é o comando dado aos propulsores. A equação do final do controlador é:

$$F = -0.75\dot{S}_d - 0.3(S_d - U_{ref}) - 0.02(\int S + d - \int U_{ref}) \tag{5.11}$$

onde

K_P = - 0.3
K_D = - 0.75
K_I = - 0.02

Essa equação está dimensionada para propulsão de -1 a 1, porém no programa a propulsão horizontal é de -100 a 100, com isso deve-se dividir os coeficientes da equação por 100. Para facilitar, os dados de distância foram dimensionados em centímetros, ou seja, é o mesmo que dividir por 100 todos os coeficientes.

Por medida de segurança, utilizou-se um limitador de comandos, pois algumas vezes os valores encontrados no cálculo são abruptos para os motores ou mudam rapidamente sem que a compensação seja realizada. Para isso, é realizado uma espécie de filtro, onde o valor final de compensação (F4) atual é a soma de 80% do valor de compensação final do instante anterior com 20% do valor calculado de F.

5.3 Sistema de Controle através do Processamento da Imagem

A partir do processamento de imagem é possível realizar sistemas de controle que não dependem dos sensores do robô, apenas da câmera. Para isso, é necessário projetar situações as quais a imagem poderá realimentar o sistema e dizer se há algum erro no comportamento atual, ou seja, se houver algo diferente da referência, isso vai ser percebido pelo processamento de imagem e vão ser realizados comandos para compensar essa diferença. Na conjuntura atual, foram projetados dois tipos de sistema. Em um, utilizando uma equação aproximada e com o tamanho de um cano sendo analisado, é possível aferir a distância do ROV do objeto. Com essa medida, pode-se tentar manter um valor de distância compensando pertubações externas, provendo comandos para os propulsores com o intuito de realizar avanços. Outro sistema se atém na centralização do objeto pela câmera do robô. Com uma simples contagem de *pixels* na imagem da câmera, é possível identificar se o objeto observado está e quanto está deslocado do centro da imagem. Com esse valor, da mesma maneira feita com a distância, existe a possibilidade de compensar as pertubações externas e manter o objeto centralizado por meio de comandos para o propulsor, com a intenção de realizar guinadas no ROV.

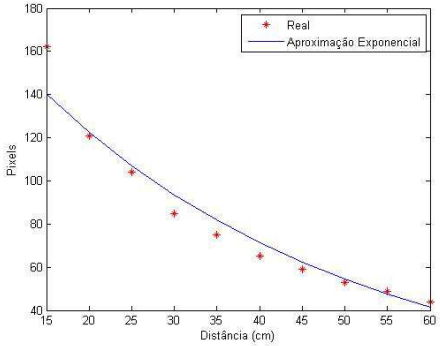
5.3.1 Controle de Regulação da Distância

Para um controle de um robô onde todos os valores de sensores aferidos estão no Sistema Internacional de Medidas, só é possível realizar os cálculos dos comandos dos propulsores com a distância sendo medida em metros. Porém, em uma imagem lida pela câmera, só está disponível o número de *pixels*. Dessa maneira, com o propósito de adquirir a relação de número de *pixels* com a distância, foram feitos experimentos com diferentes tamanhos de cano.

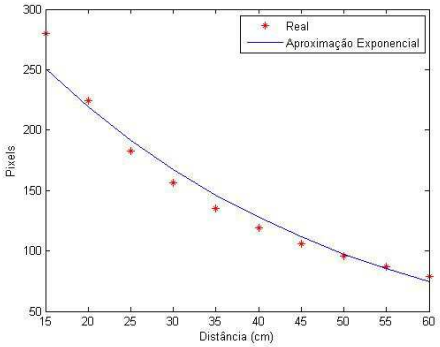
Tabela 5.1: Relação entre o número de *pixels* e a distância, para os diferentes diâmetros de cano

Distância	Número de *Pixels* Cano 20.5 cm	Número de *Pixels* Cano 15 cm	Número de *Pixels* Cano 8 cm
15 cm	370	280	158
20 cm	308	224	126
25 cm	250	183	101
30 cm	219	156	86
35 cm	189	135	74
40 cm	167	119	64
45 cm	146	106	58
50 cm	133	96	51
55 cm	121	87	46
60 cm	110	79	42

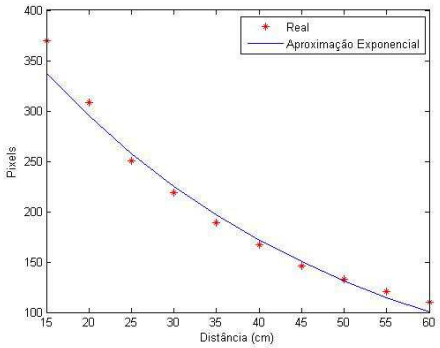
A quantidade de *pixels* varia com a distância de uma forma proporcional e de forma similar para os diferentes diâmetros de canos testados. Para melhor análise, foram projetados os pontos nos gráficos e aproximados para uma função exponencial.



(a) Cano 8 cm.



(b) Cano 15 cm.



(c) Cano 20.5 cm.

Figura 5.2: Gráfico das Aproximações.

Os gráficos observados são parecidos e podem ser dimensionados para uma única função. A escolha da aproximação pela função exponencial acontece pelo fato da simplicidade inerente a ela. Além disso, não foi encontrada nenhuma relação entre os outros tipos de funções. As exponenciais de cada gráfico são definidas:

$$P_x = 210.39e^{-0.027d} \text{ , para cano 8 cm}$$
$$P_x = 375.76e^{-0.027d} \text{ , para cano 15 cm}$$
$$P_x = 505.87e^{-0.027d} \text{ , para cano 20.5 cm}$$

onde P_x é o número de *pixels* e d é a distância.

Como é possível notar, nas aproximações exponenciais todos os expoentes são os mesmos. Isso possibilita considerar que os coeficientes que acompanham a exponencial são lineares, ou seja, é possível adquirir uma função para conseguir qualquer coeficiente a partir do tamanho do cano que está sendo analisado. Com o coeficiente que acompanha a exponencial e a quantidade de *pixels* que representam o cano na imagem da câmera do robô, adquiriu-se a distância do objeto ao ROV.

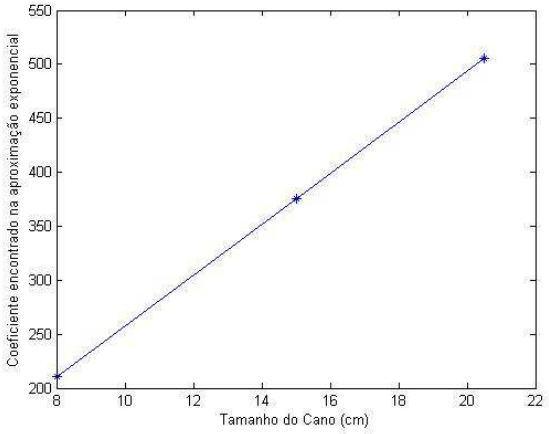


Figura 5.3: Gráfico diâmetro cano por coeficiente.

A função linear do gráfico é definida como:

$$C_f = 0,0423d_c - 0,8994$$

onde C_f é o coeficiente da exponencial e d_c é o diâmetro do cano.

Apesar da simplicidade, existe um problema com a aproximação exponencial. Ela é limitada e em pontos extremos carrega um erro grande. Nos gráficos da figura 5.2, percebe-se a disparidade no valor de 15 centímetros. Por isso, no programa, após implementar a função encontrada, projetou-se um limitador, que evita o programa de colocar no painel valores errados. Uma parte desse limite não é tão prejudicial ao programa, pois a própria câmera possui um limite de tratamento. Por exemplo, se o objeto estiver muito longe, ela não o detecta corretamente ou considera os cantos da imagem como pertencentes ao objeto, pois acabam escurecendo com o aumento da distância. Já o valores abaixo de 20 centímetros que são invalidados trazem um limite no processamento da

câmera. Uma inspeção que precisaria estar muito perto do objeto se torna impossível. A saída é colocar os valores abaixo desse limite diretamente no programa, ou seja, colocar uma tabela que, ao entrar nessa faixa de limite, representaria as medidas exatas de distância com a contagem de *pixels*.

Após o projeto de controle e a referência calculada é pertinente o seu teste para atestar sua funcionalidade. Para a implementação feita no programa criou-se a função *ControleDista*:

```
private void ControleDista(object sender, EventArgs e)
{
        DistaCm = Cm - 30;
        Proporcional2 = -0.3 * DistaCm;
        Integ2 = -0.02 * (Integ2Ant + 0.1 * DistaCm);
        Deriv2 = -0.75 * (DistaCm - DistaCmAnt) / 0.1;
        F2 = (Proporcional2 + Deriv2 + Integ2) / 1;
        F4 = 0.8 * F4Ant + 0.2 * F2;
        Integ2Ant = Integ2;
        DistaCmAnt = DistaCm;
        F4Ant = F4;
}
```

onde os valores que representam os controle proporcional, integrativo e derivativo são multiplicados pelos seus respectivos coeficientes. Para o controle proporcional basta calcular o erro entre a distância real e a referência, para o integrativo basta calcular a soma das áreas a cada intervalo de tempo e o derivativo basta calcular a variação das saídas a cada intervalo de tempo. Após isso, deve se atualizar as variáveis antigas para os valores atuais.

Utilizando um cano de 4 cm de diâmetro e aspirando que o ROV mantivesse uma distância de 30 centímetros para o cano, ou seja, a referência para os cálculos do controle, foram feitos testes para checar se o propulsor compensava as mudanças de distância do submarino em relação ao cano. Seguem os resultados:

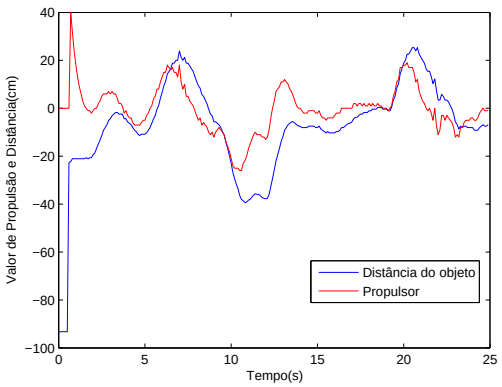


Figura 5.4: Resultados do experimento de regulação da distância.

Mesmo com o problema advindo do tempo de processamento para um sistema de controle, o controlador se mostrou estável e consolidado, carregando resultados interessantes de compensação. Com efeito, foram testados outros tipos de controle, utilizando diferentes mecanismos da câmera.

5.3.2 Controle de Regulação do Deslocamento

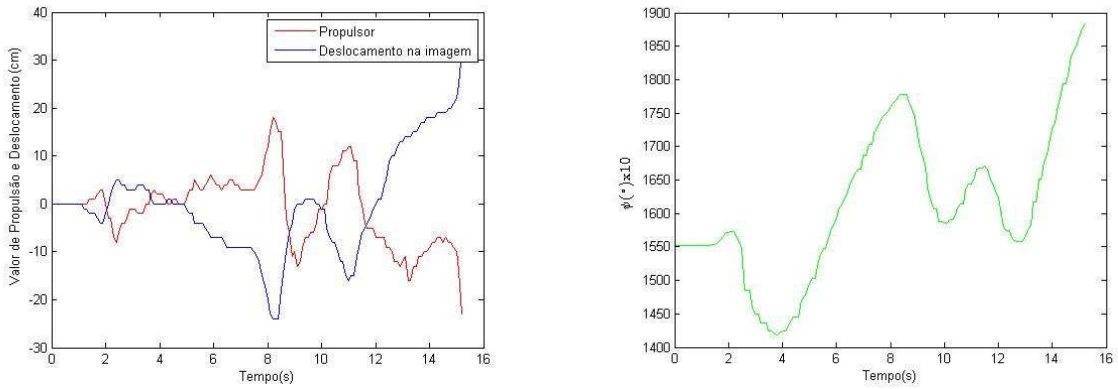
O deslocamento relaciona a centralização do objeto em relação a câmera do ROV. Portanto, um objeto estaria "deslocado"se ele estivesse um pouco a esquerda ou a direita do centro da imagem. Essa centralização é importante para o inspecionamento correto. O controle pelo deslocamento necessitou também dos valores no Sistema Internacional de Medidas. Por isso é preciso uma relação entre metros e *pixels*. Apesar do problema ser semelhante ao controle anterior, a sua solução é mais simples. A medida de deslocamento foi feita no mesmo plano que o objeto se encontra na imagem. Com isso, se houver alguma relação que diga quantos centímetros representam um certo valor de *pixels* nesse plano, é possível utilizar uma relação linear e adquirir o valor correto do deslocamento em metros. O valor do cano é cedido pelo programador e o algoritmo de controle pela distância conta o número de *pixels* do objeto. Só é necessário contar o deslocamento em *pixels* para encontrar o valor em metros. Os coeficientes e as considerações de simplificação desse projeto foram as mesmas do controle de distância e o comando é chamado de F3. Por isso, os valores manuseados e os coeficientes da equação de controle foram os mesmos.

Com esse controle estabelecido, é pertinente rodar testes para atestar os cálculos. Para a implementação feita no programa criou-se a função *ControleDesloc*:

```
 private void ControleDesloc(object sender, EventArgs e)
{
        Proporcional = -0.3 * DeslocCm;
        Integ = -0.02*(IntegAnt + 0.1 * DeslocCm);
        Deriv = -0.75*(DeslocCm - DeslocCmAnt) / 0.1;
        F1 = (Proporcional + Deriv + Integ)/1;
        F3 = 0.8*F3Ant + 0.2*F1;
        IntegAnt = Integ;
        DeslocCmAnt = DeslocCm;
        F3Ant = F3;
}
```

onde os termos seguem a mesma lógica da função *ControleDista*.

Com um cano, foram feitos deslocamentos do centro da imagem e verificou-se se a propulsão age de forma a compensar os erros calculados. Seguem os resultados:



(a) Propulsão e deslocamento da imagem.

(b) Orientação em Z.

Figura 5.5: Resultados do experimento de regulação do deslocamento

Nos valores de deslocamento, o negativo representa o objeto se movendo para a esquerda e o positivo para a direita. Já nos valores de propulsores, o positivo representa um movimento de giro anti-horário, para a esquerda, e o negativo um movimento horário, para a direita.

No primeiro gráfico, figura 5.5(a), é possível observar uma resposta de propulsão muito rápida em todos os deslocamentos. Isso atesta a qualidade do controle. No segundo gráfico, figura 5.5(b), está sendo representado o valor do sensor de *Heading* do ROV. Analisando a curva do gráfico, o movimento do robô acompanha o propulsor. Utilizou-se do mesmo filtro do controle de distância.

(a) Instante 8.40 segundos.

(b) Instante 9.20 segundos.

Figura 5.6: Imagens do painel.

As imagens da figura 5.6 exibem o painel com os dados lidos. O objeto está deslocado na figura 5.6(a), no instante de tempo 8 minutos e 40 segundos, e seu valor de deslocamento é inclusive mostrado como -24. No gráfico de deslocamento, no mesmo instante, o valor correspondente é o mesmo, comprovando que de fato o cano estava deslocado na imagem. Já na imagem 5.6(b), o instante analisado é 9 minutos e 20 segundos, percebe-se que o deslocamento é igual a zero, ou seja, ele foi compensado e o objeto está centralizado. Ao analisar o mesmo instante de tempo no gráfico, comprova-se novamente o sucesso do sistema de controle implementado.

5.3.3 Controle de Regulação do Deslocamento e da Distância

Com o sucesso dos dois controladores, criou-se a meta de juntar os dois, o que tornaria possível a compensação de pertubações externas como um todo. Sendo assim, na inspeção de um cano, o principal exemplo do trabalho, os comandos calculados manteriam o robô numa mesma distância e numa mesma direção para o objeto.

Os comandos estabelecidos para o ROV foram a soma de F3 e F4, o que resulta em uma compensação de guinada e de avanço ao mesmo tempo. O que se viu em testes foi uma compensação mais rápida e mais ajustada de avanço, enquanto a guinada tinha uma resposta lenta e ruim. Isso ocorre, pois a potência em avanço é maior que a potência em guinada com o mesmo valor de propulsão. Para melhorar e consertar a compensação da guinada, multiplicou-se o valor de F3 por 1.5, aumentando o valor dado pelo projeto em 50%. Após essa mudança, as respostas dos dois controles ficaram satisfatórias.

Nos testes, realizou-se diferentes movimentos com um cano, com o objetivo de simular os diferentes erros que o projeto de controle pode compensar. Seguem os resultados na figura 5.7:

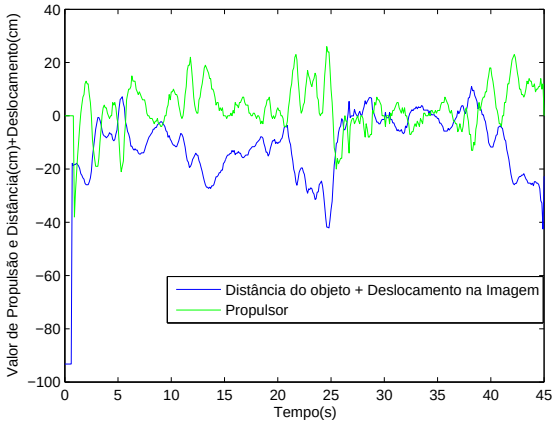


Figura 5.7: Resultados do experimento de regulação da distância e do deslocamento.

Os gráfico de distância e deslocamento foram somados, pois não há com separá-los já que a compensação ocorre como um todo. Os controles de deslocamento e distância se mostraram ótimos. Mesmo com o cálculo simplificado e várias simplificações, o ROV se comportou bem em situações adversas.

Capítulo 6

Conclusão e Trabalhos Futuros

6.1 Conclusão

O objetivo do trabalho foi modelar e projetar um controle para um ROV de forma experimental. Na universidade, o ensino é muito teórico, por isso foi escolhido um trabalho de graduação mais prático. O projeto foi realizado em etapas que fundamentaram o controle. Feita a revisão bibliográfica determinou-se o equacionamento de todo o sistema. A modelagem é complexa, devido as não-linearidades, e nem sempre cálculo teóricos são aplicáveis experimentalmente. Desta forma, como havia um modelo para ser testado, os embasamentos teóricos foram simplificados para que a prática pudesse ser mais focada e funcionasse. Sendo assim, considerou-se que os sistemas eram desacoplados e apenas as linearidades foram utilizadas.

O foco principal do trabalho foi o entendimento do programa e a criação de ferramentas computacionais, de forma a transformar o submarino em um robô totalmente operacional. Com esse foco na prática, vários testes foram realizados, sempre com o intuito de compreendê-lo. Procedimentos foram repetidos de diferentes formas para validar os experimentos. Simplificar a teoria e insistir na prática foi o motivo do sucesso, de maneira que muitos coeficientes calculados foram modificados ou invalidados.

O programa base não possuía a leituras de sensores que eram importantes para o controle, como a aceleração linear, angular e a orientação. Com isso, foi preciso entender o funcionamento da biblioteca da *VideoRay*. Após isso, foi necessário entender os valores do sensores e suas unidades de medida. As saídas encontradas foram ruidosas e para isso utilizou-se de técnicas de processamento digital de sinal.

Como o robô possui uma câmera, empregou-se o processamento de imagens para adquirir informações de visão computacional para projetar o controlador. A grande barreira encontrada foi a de velocidade de processamento. Apenas uma imagem já acarreta um atraso gigantesco. Por isso foram idealizados mecanismos de otimização para manter o controle do veículo em tempo real. Com o auxílio de operações matemáticas, foi encontrada uma relação entre distância e o número de *pixels* de um cano localizado a sua frente. Adicionalmente, também foi encontrada uma relação entre o deslocamento de um objeto em relação ao centro do ROV e o número de *pixels*.

Com os dados estabelecidos e após definir a equação da planta linearizada, projetou-se um controlador a partir da equação característica desejada. Os resultados encontrados foram satisfatórios, o controlador consegue compensar a distância e o deslocamento. Além disso, o tempo de resposta, como observado nos gráficos, foi bom.

6.2 Sugestão de Trabalhos Futuros

O controle de veículos como o ROV estudado, possui ilimitados estudos possíveis. Por isso, existe ainda uma vasta janela de possibilidades de controladores, utilizando por exemplo as próprias referências dos sensores do robô como entrada. O controle ideal possuiria tanto as referências da câmera como a dos sensores, acarretando em uma resposta mais aprimorada. Sugestões de trabalhos futuros:

- Desenhar o submarino em uma plataforma CAD, onde é possível o estudo automatizado de toda a planta, obtendo corretamente os coeficiente e valores considerados no trabalho
- Relatar para a VideoRay o defeito encontrado no acelerômetro angular e adquirir informações de conserto ou de certeza do defeito
- Manter a referência de *Heading* enquanto realiza avanço.
- Manter a referência de *Heading* e o objeto centralizado na imagem enquanto realiza afundamento.
- Modificar os valores de *Heading* no meio da execução do programa.
- Otimização do programa.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] CHRIST, R.D., WERNLI SR, R.L. 2007. *The ROV Manual: A User Guide for Observation Class Remotely Operated Vehicles.* Oxford: Elsevier Ltd. 1 ed.

[2] *Video Ray Pro 4: operator manual.* Estados Unidos: 2010, 201 p.

[3] SOUZA, E. C. *Modelagem e Controle de Veículos Submarinos não tripulados.* 2003. 168 f. Dissertação (Mestrado) - Escola Politécnica, Universidade de São Paulo, São Paulo. 2003.

[4] FOSSEN, T. I. *Guidance and Control of Ocean Vehicles.* John Wiley & Sons, 1994.

[5] PASQUETTI, E. *Estabilidade Estática e Dinâmica de Torres Estaiadas.* 2003. 98 f. Dissertação (Mestrado). Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2003.

[6] GONZALEZ, R.C., WOODS, R.E. *Digital Image Processing.* 2 ed. New Jersey: Prentice Hall, 2002. 779 p.

[7] MARQUES FILHO, O., VIEIRA NETO, H.*Processamento Digital de Imagens.* Rio de Janeiro: Editora Brasport, 1999.

[8] DEITEL, H. M., et al. *C# Como Programar.* 2003. 2 ed. São Paulo: Pearson Education, 2003. 1137 p.

[9] SMITH, S. W. *The Scientist and Engineer's Guide to Digital Signal Processing.* 1 ed. 1997.

[10] FOSSEN, T. I. *Handbook of Marine Craft Hydrodynamics and Motion Control.* John Wiley & Sons, 2011.

[11] OGATA, K. *Engenharia de Controle Moderno.* 4 ed. São Paulo: Pearson Education, 2003. 783 p.

[12] PAULINO, A. *Controle por planejamento e acompanhamento de trajeória para veículo operado remotamente.*

ANEXOS

I. DIAGRAMAS ESQUEMÁTICOS

II. DESCRIÇÃO DO CONTEÚDO DO CD

A implementação computacional utilizou-se do software *Visual Studio* e o arquivo de execução está na pasta 'Implementacao_Computacional'. Nela constam os seguintes arquivos:

- 'vrSimpleSample.sln': este é o arquivo principal, ao iniciá-lo no Visual Studio, basta dar Start para que o programa execute e o painel de controle seja aberto.
- 'Form1.cs': é a rotina principal, nesta classe existem todas as funções implementadas:
  - private void Luzes_Scroll: função que inicializa as luzes.
  - private void button1_Click_1: função do botão STOP.
  - private void button1_Click_2: função que salva as imagens.
  - private void Video_Click: função que tira fotos continuamente em um laço.
  - private void AnotaDados: função que salva os dados no arquivo txt escolhido. O caminho dos arquivos tem que ser colocado no programa.
  - private void GrayScaleFilter: função que binariza a imagem.
  - private void ContagemPixels: função que conta a quantidade de pixels em uma linha.
  - private void ContPix: função que chama as funções GrayScaleFilter e ContagemPixels.
  - private void Pix_To_Cm: função que converte pixels em centímetros.
  - private void ControleDesloc: função de implementação do controle do deslocamento.
  - private void ControleDista: função de implementação do controle da distância.
- 'Form2.cs': classe com as funções de inicialização da câmera.
- No caminho bin\x86\Debug são salvas as imagens.
- 'Image1.bmp': os arquivos de câmera devem ser inicializados, com isso a primeira imagem da câmera na primeira interação é o 'Image1'.

Além dos arquivos computacionais, a pasta raiz do CD contém também o relatório em PDF, principal_relatorio.pdf, com todas as considerações e cálculos realizados no trabalho.