TRABALHO DE GRADUAÇÃO

CONSTRUÇÃO DE UM SISTEMA DE MEDIÇÃO DA POSIÇÃO E ORIENTAÇÃO DE MANIPULADORES ROBÓTICOS BASEADO EM VISÃO ATIVA COM PROJEÇÃO DE LUZ LASER

Sara Gomes Cardoso

Brasília, Dezembro de 2019

ENGENHARIA
MECATRÔNICA
UNIVERSIDADE DE BRASÍLIA

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

TRABALHO DE GRADUAÇÃO

CONSTRUÇÃO DE UM SISTEMA DE MEDIÇÃO DA POSIÇÃO E ORIENTAÇÃO DE MANIPULADORES ROBÓTICOS BASEADO EM VISÃO ATIVA COM PROJEÇÃO DE LUZ LASER

Sara Gomes Cardoso

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

Banca Examinadora

Prof. José Maurício S. T. da Motta, ENM/UnB _____
Orientador

Prof. Jones Yudi Mori Alves da Silva, ENM/UnB _____
Examinador interno

Prof. Gerardo Antonio Idobro Pizo, FGA/UnB _____
Examinador interno

Brasília, Dezembro de 2019

FICHA CATALOGRÁFICA

SARA, GOMES CARDOSO
Construção de um Sistema de Medição da Posição e Orientação de Manipuladores Robóticos Baseado em Visão Ativa com Projeção de Luz Laser,

[Distrito Federal] 2019.

viii, 73p., 297 mm (FT/UnB, Engenheiro, Controle e Automação, 2019). Trabalho de Graduação – Universidade de Brasília. Faculdade de Tecnologia.

1. Posição 2.Manipuladores
3. Visão 4.Laser

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

REFERÊNCIA BIBLIOGRÁFICA

CARDOSO, S. G., (2019). Construção de um Sistema de Medição da Posição e Orientação de Manipuladores Robóticos Baseado em Visão Ativa com Projeção de Luz Laser. Trabalho de Graduação em Engenharia de Controle e Automação, Publicação FT.TG-$n°$09, Faculdade de Tecnologia, Universidade de Brasília, Brasília, DF, 73p.

CESSÃO DE DIREITOS

AUTOR: Sara Gomes Cardoso

TÍTULO DO TRABALHO DE GRADUAÇÃO: Construção de um Sistema de Medição da Posição e Orientação de Manipuladores Robóticos Baseado em Visão Ativa com Projeção de Luz Laser.

GRAU: Engenheiro ANO: 2019

Sara Gomes Cardoso

CLN 411, Bloco A, Sala 217.

70866510 Brasília – DF – Brasil.

Agradecimentos

Sou grata a Deus, meu Senhor e salvador, por sempre me proporcionar experiências incríveis em todas as jornadas que escolhi me aventurar. Deus é bom e me mostra isso diariamente, sou grata a Ele por nunca me abandonar, por estar sempre estar ao meu lado, e por me amar como nenhum outro poderia.

Agradeço também a minha família. Eu não mereço tamanho amor e carinho, mas mesmo assim o recebo diariamente, ainda que à distância. Meus pais e meu irmão sempre me apoiam e durante o trabalho de conclusão de curso isso só ficou mais evidente. Obrigada por todas as palavras de consolo e incentivo, eu não teria conseguido sem vocês. Igor, obrigada por ter se mostrado o melhor irmão que eu poderia ter, nunca esquecerei seus esforços para me ajudar em tudo o que eu precisava.

Não poderia deixar de mencionar também todo o apoio e oração que recebi dos meus amigos nesse período. Agradeço aos amigos que ingressaram no curso de engenharia mecatrônica comigo e permaneceram comigo até o final, agradeço aos amigos da DROID que fizeram dessa equipe uma família para mim. Aos amigos que passaram noites em claro ao meu lado escrevendo o relatório final, meu muito obrigada pela companhia. Anderson, Rebeca e Thiago, obrigada por dividirem esse momento comigo. Abdullah, obrigada por todos os momentos que você disponibilizou para me ajudar a tirar dúvidas de visão computacional. Crepe, obrigada por todo o apoio e ajuda que você me propiciou, principalmente nos últimos dias.

Meus sinceros agradecimentos a três pessoas que eu não poderia de mencionar: Thiago, obrigada por ser meu melhor amigo da DROID e por sempre me encorajar, acreditando e me fazendo acreditar que tudo daria certo e que eu era capaz. Obrigada por todas as caronas, por todos os lanches e por todos os desabafos que você aturou. Obrigada por sua amizade. André, em momentos de estresse você me proporcionou risadas com assuntos engraçados e aleatórios; em momentos de desespero, me mostrou o quanto se importava comigo; e em momentos de desabafo, se mostrou um ótimo confidente. Obrigada por todo o seu apoio nessa etapa da minha vida e por ter se voluntariado para corrigir a minha escrita nesse relatório, eu não sei o que eu faria sem você. Jessyca, obrigada por ser minha melhor amiga da vida e por sempre me apoiar. Mesmo à distância sua amizade é um alento para minha alma.

Por fim, agradeço ao meu orientador, responsável pela idealização desse projeto e por me convencer a fazer parte dele.

Sara Gomes Cardoso

RESUMO

O projeto consiste na construção de um sistema de medição da posição e orientação de um manipulador, baseado em visão ativa, em relação a um alvo fixo especialmente construído. O sistema de medição é composto de uma câmera CMOS de pequenas dimensões, de alta resolução, e de um projetor de plano de luz *laser*, cuja imagem da sua interseção com o alvo deverá ser processada e utilizada no controle da posição desejada para o robô. A primeira etapa do projeto consistiu na montagem física do sensor, exigindo calibração óptica e dimensional. Em seguida foram implementados códigos no *software* MATLAB® para processamento das imagens e, por fim, elaborou-se um modelo do controle da posição do robô baseado em coordenadas da imagem, para movimentos diferenciais. O sistema de medição construído terá aplicação em um sistema de calibração de robôs, mas fora do âmbito deste projeto. Os resultados computacionais puderam validar a ideia proposta para alinhamento do sensor, porém não foi possível obter todos os resultados de validação no manipulador robótico. Além disso, houve dificuldade na obtenção de uma conversão ótima do deslocamento em coordenadas da imagem para coordenadas de junta, sendo necessário uma reavaliação dos parâmetros utilizados no algoritmo implementado. Conclui-se que o projeto alcançou em grande parte os resultados pretendidos, porém há a necessidade de ser finalizado.

Palavras Chave: Medição; Posição; Orientação; Visão; CMOS; Laser; Robô; Calibração; Óptica; Processamento; Controle.

ABSTRACT

The project consists in the construction of a position and orientation system for a robot manipulator, using active vision and a specially constructed fixed target. The system consists in a small, high resolution CMOS camera and a laser light plane projector whose image of its intersection with the target will be processed and used to control the selected position for the robot. The project involves a physical assembly of components, requiring optical and calibration. In the second stage, the image processing software will be developed and, in the third step, an image-based robot control model using differential position control. The built measurement system will have application in a robot calibration system, but out of the scope of this project.

Keywords: Position; Orientation; Measurement; Vision; CMOS; Laser; Robot; Calibration; Optical; Processing; Control.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE SÍMBOLOS

Símbolos Gregos

β	Angulação do feixe de luz	[^{o}]
λ	Escalar arbitrário	[ ]
θ	Coordenada polar; Ângulo de junta do manipulador robótico	[^{o}]
ρ	Distância radial	[ ]
Ω	Vetor velocidade angular	[rad/s]
ω	velocidade angular	[rad/s]
δ	Derivada parcial	[ ]

Grupos Adimensionais

i, j, n, m	Contador
A	Objeto
F	Ponto focal
A'	Imagem
C	Centro de projeção da câmera
X	Matriz ou vetor
X^T	Matriz ou vetor transposto
X^{-1}	Matriz inversa
X^*	Vetor de valores esperados de X
$\tilde{X}$	Matriz ou vetor em representação homogênea
$\hat{X}$	Matriz ou vetor aproximado
P	Matriz de projeção da câmera
(u, v)	Coordenadas de um ponto na imagem
(u_0, v_0)	Coordenadas do ponto principal da imagem
K	Matriz de parâmetros intrínsecos
I	Matriz identidade
R	Matriz de rotação
H	Matriz de homografia
(X_f, Y_f)	Coordenadas da imagem em uma distância focal
(s_x, s_y)	Fator de escala
$({X_f}', {Y_f}')$	Coordenadas da imagem após mudança da distância focal
(C_x, C_y)	Coordenadas do centro óptico
(X_c, Y_c, Z_c)	Coordenadas do ponto com respeito ao frame de referência a câmera
J_s	Jacobiano da Imagem
v_c	Velocidade da câmera
T	Vetor velocidade translacional
$\dot{X}$	Derivada de um vetor

Siglas

CMOS	*Complementary Metal-Oxide Semiconductor*
3D	*Three Dimensional*
MATLAB	*Matrix Laboratory*
LASER	*Light Amplification by Stimulated Emission of Radiation*
BFS	*Breadth First Search*
DFS	*Depth First Search*
IBVS	*Image-Based Visual Servoing*
USB	*Universal Serial Bus*
GRACO	*Grupo de Automação e Controle*
TCP	*Transmission Control Protocol*
IP	*Internet Protocol*

Capítulo 1

Introdução

1.1 Contextualização

Técnicas de medição de formas tridimensionais, visão baseada em 3D, tem se destacado bastante como uma importante ferramenta para diversas aplicações, como detecção de objetos, geração de modelos digitais, replicação de objetos, engenharia reversa, prototipagem rápida, inspeção de produtos e controle de qualidade. As suas vantagens são a ausência de contato físico durante a operação de medição, agilidade na aquisição de dados, baixo custo e boa estabilidade [10]. No ambiente industrial, tem-se utilizado essa técnica de inspeção a fim de melhorar o controle de qualidade dos produtos, garantindo assim não apenas sua qualidade, mas também a redução de taxas de rejeição e economia de custos. Essa tecnologia dispõe de um sensor visual e um dispositivo de orientação, com o objetivo de posicionar o sensor para o objeto alvo. Para a indústria, tem-se a necessidade do dispositivo de orientação ser flexível e controlável para aumentar a eficiência e automação da tecnologia de inspeção. Isso tem impulsionado a adoção de robôs industriais como dispositivos de orientação, permitindo a combinação de um robô industrial de alta flexibilidade com um sensor visual de alta exatidão [11].

Este trabalho implementará um sistema de medição da posição e orientação de um manipulador robótico, baseado em visão ativa, com relação a um alvo fixo, e terá aplicação em um sistema de calibração de robôs em projetos futuros.

1.2 Definição do problema

Para realizar uma tarefa, o manipulador deve executar uma sequências de ações definidas pela sua programação. Esta por sua vez, pode classificada em dois tipos, programação *online* e programação *offline*. No primeiro método, o movimento do robô é controlado por um operador por meio de um dispositivo denominado *teach pendant*. As articulações do robô são ajustadas uma por uma até o efetuador terminal atingir a posição de interesse. A configuração obtida é armazenada na memória do controlador para que possa ser alcançada novamente no futuro. O segundo método consiste na implementação de um código, contendo uma sequência de linhas de

comando, que serão transferidas para o controlador do robô executar algum movimento e atingir a posição final desejada para o efetuador terminal. Esse método pode ser executado à distância, não necessitando assim estar no mesmo ambiente que o robô. Para sua validação, pode-se realizar testes tanto em robôs no ambiente de desenvolvimento quanto em softwares que simulem o robô e o ambiente de desenvolvimento em questão [12].

Embora a programação *offline* pareça mais vantajosa por não necessitar que o operador esteja em contato direto com o robô, evitando interrupções na linha de produção, ainda há desafios críticos a serem resolvidos nesse método, impedindo-o de ser amplamente difundido nas indústrias. Os maiores problemas estão associados a repetibilidade e precisão, que podem ser associadas às imprecisões do modelo geométrico utilizado pelo controlador e dificuldades de obter precisamente a posição dos objetos com relação a um determinado sistema de coordenadas [13].

Para resolver os problemas da programação *offline*, opta-se por realizar a calibração de robôs, uma forma economicamente viável de melhorar a exatidão da posição do robô, devido ao fato de identificar uma relação funcional mais exata entre a leitura do transdutor da junta e a posição do efetuador terminal [12, 11].

Yin et al. (2013) propôs uma abordagem de autocalibração de robôs para um sistema robótico de inspeção visual, em que o sensor visual é acoplado ao efetuador terminal, servindo como uma ferramenta, como pode ser observado na Figura 1.1. O ponto central da ferramenta é definido e calibrado utilizando o modelo do sensor. Foi verificado que essa abordagem melhorou a exatidão do robô de forma significativa, além de reduzir a propagação do erro encadeado gerado por outros métodos e tornar os procedimentos de calibração mais convenientes para implementação [11].

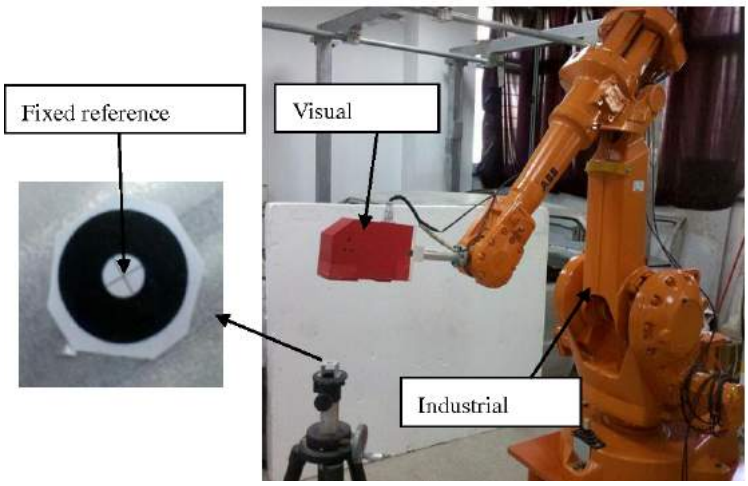


Figura 1.1: Sistema para autocalibração de um robô, apresentado por Yin et al.(2013).

Decidiu-se, portanto, construir um sistema de medição de posição e orientação baseado na abordagem descrita no parágrafo acima, para no futuro o sistema ser aplicado em sistemas de calibração.

1.3 Objetivos do projeto

O principal objetivo desse projeto é a construção de um sistema de medição de posição e orientação de um manipulador robótico baseado em visão ativa, com relação a um alvo que também será construído, utilizando um sensor composto por uma câmera monocular e um projetor de plano de luz *laser*.

A fim de alcançar tais objetivos, serão seguidos os seguintes passos:

- Avaliar e comparar ferramentas de visão computacional e métodos de calibração de câmera aplicáveis ao projeto, utilizando o software MATLAB®.
- Implementar um programa para calibração de câmera monocular no MATLAB®, utilizando um alvo plano de pontos.
- Projetar e construir o sensor para projeção de um plano laser que passe pelo eixo óptico da câmera-lente e o alvo físico de referência.
- Implementar um programa para calibração do centro óptico da imagem no MATLAB®, utilizando um alvo plano de pontos em diferentes posições focais.
- Projetar e construir um alvo físico de referência.
- Desenvolver um programa para cálculo da posição do sensor por meio da análise da imagem, relacionando as variações das coordenadas de imagem obtidas pelo sensor com a posição da câmera.

  Nessa etapa deve-se implementar algoritmos para identificação das linhas nas imagens, para o controle servo visual baseado em imagem e para para o sistema de medição baseado em triangulação construído.
- Realizar simulações no software RobotStudio® com as imagens obtidas pelo robô e a correção da posição e orientação do robô para posicionar o eixo óptico da câmera sobre o alvo. A partir das simulações, comparar e avaliar o sistema de posicionamento do robô.

Capítulo 2

Revisão Bibliográfica

2.1 Introdução

Este capítulo apresentará os conceitos físicos e matemáticos utilizados no desenvolvimento do projeto, nas áreas de visão computacional, óptica e robótica. Inicialmente serão introduzidos conceitos de formação de imagem e triangulação, calibração de câmeras e calibração do centro óptico. Em seguida, o foco será na descrição das técnicas de detecção de bordas e formas utilizadas no projeto. Para finalizar, serão apresentados alguns conceitos de controle servo visual.

2.2 Visão Computacional

Imagens são representações em superfícies bidimensionais de uma cena tridimensional e o campo que estuda a formação, análise e interpretação de imagens, é chamado de visão computacional. Essa área é reservada para os campos que interpretam as informações visuais em termo das propriedades tridimensionais presentes nas imagens [3].

2.2.1 Modelo *Pinhole*

O modelo mais simples utilizado para formar uma imagem é conhecido como *pinhole* e consiste em uma caixa com um pequeno orifício, localizado no centro da projeção, por onde passam os raios de luz, de uma fonte externa, que serão projetados do lado oposto da caixa, em uma camada fotossensível, formando uma imagem invertida do objeto detectado. Esse modelo baseia-se no fato dos raios de luz viajarem em linhas retas, o que acontece quase sempre, e é suficiente para o propósito desse modelo [14].

Esse modelo, entretanto, encontra dificuldades justamente no tamanho ideal que o orifício por onde passam os raios de luz deve ter. Deseja-se, idealmente, uma imagem nítida e focada, porém são duas coisas inversamente proporcionais nesse modelo. Quando o orifício é largo, mais raios passam por ele, deixando a imagem mais iluminada, entretanto, esses raios se espalham pela imagem, deixando-a desfocada. Ao se diminuir o tamanho do orifício, é possível obter um foco

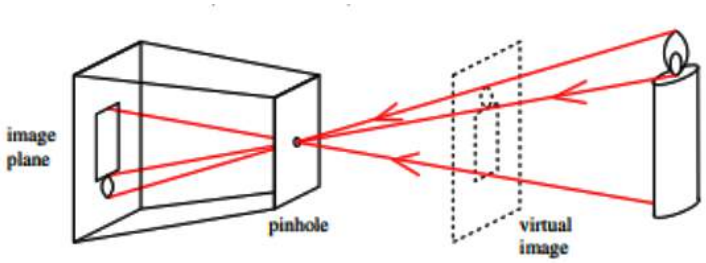


Figura 2.1: Modelo de câmera *pinhole* [1].

melhor, porém uma imagem menos nítida e existe um limite, que quando ultrapassado, acaba resultando na difração da luz, deixando a imagem desfocada e pouco nítida [1].

A solução encontrada para esse problema é o uso de lentes ou espelhos curvos para coletar a luz em uma área maior. Lentes convexas formam imagens da mesma forma que o modelo *pinhole*, mas devido ao seu largo diâmetro, permite que mais luz passe, e assim, imagens mais nítidas são geradas [15].

2.2.2 Construção Geométrica de Imagens com Lentes Finas

Para analisar a formação de imagens a partir de lentes, será necessário estabelecer alguns critérios. A lente deverá ter duas superfícies esféricas de raio R e índice de refração n, além de ser fina, isto é, após sofrer a primeira refração pela borda direita da lente, o raio deve ser imediatamente refratado pela borda esquerda. O meio a ser considerado deve ser o vácuo (o ar é uma boa aproximação), com índice de refração igual a 1 [1].

Na figura 2.2, o ponto A está localizado em z_0, e por ele passam 3 raios. O raio r_0 inicia-se paralelo ao eixo óptico e é refratado após atingir a borda esquerda da lente, imediatamente ele atinge a borda direita e é refratado novamente, produzindo o raio r_0', que intercepta o eixo óptico no ponto focal F. O raio r_1 passa pelo centro da lente convergente e não sofre nenhum desvio, tal qual ocorre no modelo *pinhole*. Por fim, o raio r_2 passa pelo ponto focal F antes de atingir a borda esquerda da lente e após sofrer as duas refrações, se transforma no raio r_3', paralelo ao eixo óptico.

Os três raios partem de um mesmo ponto A, a uma distância h_0 do eixo óptico, e interceptam o mesmo ponto A', a uma distância h_i do eixo óptico, no lado oposto da lente, onde a imagem do objeto é formada. As distâncias z_0 e z_i e o ponto F se relacionam pela equação fundamental das lentes finas:

$$\frac{1}{z_0} + \frac{1}{z_i} = \frac{1}{f} \tag{2.1}$$

Para $z_0 > f$, uma imagem invertida é formada no plano da imagem em $z < -f$. A imagem é considerada real quando $z_i > 0$ e virtual quando $z_i < 0$.

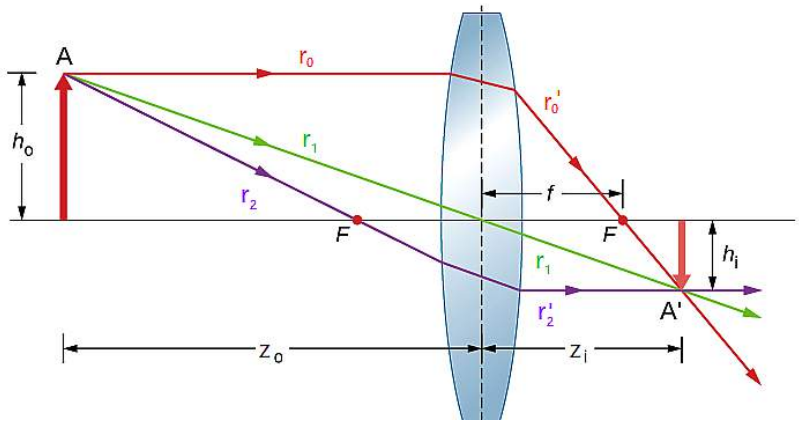


Figura 2.2: Formação de imagem a partir de uma lente fina, adaptado de [2].

2.2.3 Matriz Interna da Câmera

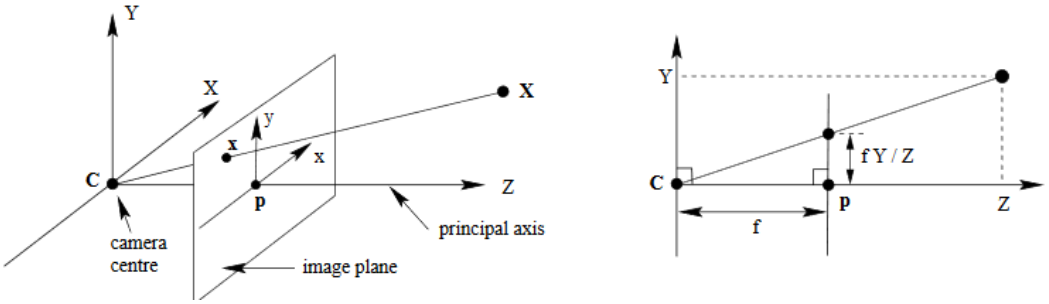


Figura 2.3: Modelo de Projeção Central [3].

Para aplicações de visão computacional, é comum utilizar o modelo de projeção central da imagem, no qual o centro de projeção C é a origem de um referencial euclidiano e o plano em $Z = f$, denominado plano da imagem ou plano focal, é onde se forma a imagem. Nesse modelo, os feixes de luz provenientes do espaço convergem no centro de projeção da câmera, formando uma imagem não invertida projetada no plano da imagem [3, 15].

As deduções matemáticas a seguir foram retiradas do livro [3]. Pelo modelo de câmera *pinhole*, sabe-se que o ponto no espaço com coordenadas $X = (X, Y, Z)^T$ é mapeado em um ponto no plano da imagem no local em que a linha que passa pelos pontos X e C intercepta o plano da imagem. Usando semelhança de triângulos, pode-se determinar as coordenadas da imagem em função das coordenadas do ponto.

$$x = f\frac{X}{Z}, y = f\frac{Y}{Z} \tag{2.2}$$

Dessa forma, pode-se determinar o mapeamento de um ponto $X = (X, Y, Z)^T$ no plano da imagem como:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \rightarrow \begin{bmatrix} f\frac{X}{Z} \\ f\frac{Y}{Z} \end{bmatrix} \tag{2.3}$$

Em coordenadas homogêneas, pode-se modelar matricialmente uma câmera *pinhole* ideal conforme a equação a seguir:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.4}$$

, que pode ser representada de forma compacta como:

$$\tilde{x} = P\tilde{X} \tag{2.5}$$

, em que as coordenadas no plano da imagem são representadas por $\tilde{x}$ e as coordenadas espaciais por $\tilde{X}$. A matriz homogênea de projeção da câmera é represenada por P.

As coordenadas *(X,Y,Z e x,y)* e a distância focal são medidas com base no sistema métrico decimal, mas na prática, as coordenadas da câmera (x, y) são medidas em pixels. Para resolver essa inconsistência na unidade de medida, realiza-se a conversão para pixel com a introdução de um fator de escala s_x e s_y:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} s_x f & 0 & 0 & 0 \\ 0 & s_y f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.6}$$

As coordenadas do ponto expressas em coordenadas de pixel (u, v) são obtidas pelo mapeamento:

$$\begin{aligned} u &= s_x x + u_0 \\ v &= s_y y + v_0 \end{aligned} \tag{2.7}$$

, em que (u_0, v_0) são as coordenadas em pixel do ponto principal.

A matriz de projeção de uma câmera, pode, portanto, ser representada em coordenadas homogêneas da seguinte forma:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} s_x f & 0 & u_0 & 0 \\ 0 & s_y f & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.8}$$

, onde a matriz de calibração da câmera, ou matriz de parâmetros instrínsecos, é representada por:

$$K = \begin{bmatrix} f s_x & 0 & u_0 \\ 0 & f s_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.9}$$

Cuja forma concisa é dada por:

$$x = K \begin{bmatrix} I & | & 0 \end{bmatrix} X_{cam} \tag{2.10}$$

, em que I é a matriz identidade de dimensão $3x3$.

X_{cam} substituiu $(X, Y, Z, 1)^T$ para enfatizar que a câmera deve estar localizada na origem do sistema de coordenadas Euclidiano com o eixo principal da câmera apontando diretamente para o eixo X. O ponto X_{cam} é expresso nesse sistema de coordenadas, conhecido também como sistema de coordenadas da câmera.

Para uma situação em que os píxeis não sejam quadrados, um novo fator é acrescentado na matriz. Ele representa a relação angular entre as direções do píxel e transforma a matriz 2.9 em:

$$K = \begin{bmatrix} fs_x & s_\theta & u_0 \\ 0 & fs_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.11}$$

2.2.4 Matriz Externa da Câmera

As próximas deduções matemáticas serão baseadas no livro [3]. Pontos no espaço podem ser expressos em termo de um sistema de coordenadas Euclidiano diferente, conhecido como sistema de coordenadas global. Esse sistema se relaciona com o sistema de coordenadas da câmera por meio de uma rotação e translação. As coordenadas de um ponto no sistema de coordenadas global pode ser representada por um vetor $\tilde{X}$ não homogêneo de tamanho 3, dessa forma, podemos escrever $\tilde{X}_{cam} = R(\tilde{X} - \tilde{C})$, onde $\tilde{C}$ representa as coordenadas do centro da câmera no sistema de coordenadas global, e R é uma matriz de rotação 3x3 que representa a orientação do sistema de coordenadas da câmera. Essa equação pode ser escrita em coordenadas homogêneas como:

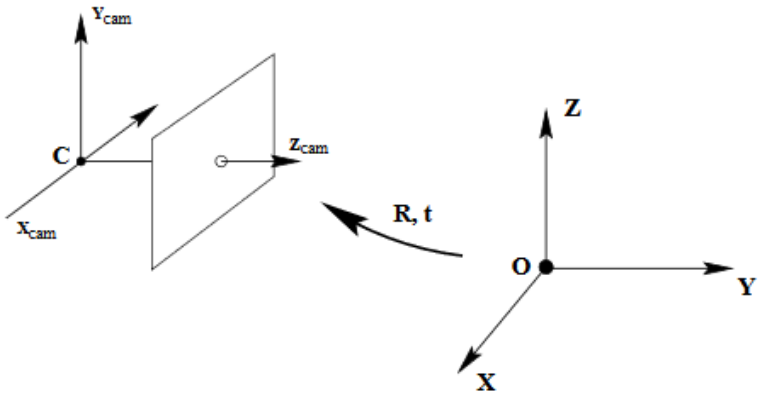


Figura 2.4: Transformação Euclidiana entre os sistemas de coordenadas global e da câmera [3].

$$X_{cam} = \begin{bmatrix} R & -R\tilde{C} \\ 0 & 1 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \begin{bmatrix} R & -R\tilde{C} \\ 0 & 1 \end{bmatrix} X \tag{2.12}$$

Substituindo a equação 2.12 na equação 2.10:

$$x = KR\left[I \quad | \quad -\tilde{C}\right]X \tag{2.13}$$

É comum ao invés de explicitar o centro da câmera, representar a transformação do espaço 3D para imagem como $\tilde{X}_{cam} = R\tilde{X} + t$. Nesse caso, a matriz da câmera é simplificada para:

$$P = K\left[R \quad | \quad t\right] \tag{2.14}$$

Onde $t = -R\tilde{C}$.

2.3 Calibração de Câmera

Calibração de câmera é o processo de determinação das características internas ópticas e geométricas da câmera (parâmetros intrínsecos) e sua posição e orientação tridimensional relativa ao sistema de coordenadas externo (parâmetros extrínsecos). As técnicas utilizadas dependem do conhecimento das coordenadas relativas do conjunto de pontos no espaço 3D e das coordenadas de pixel no plano da imagem. [16].

A literatura apresenta diversos métodos para calibração geométrica de câmeras. A abordagem clássica originada do campo da fotogrametria, resolve o problema minimizando o erro de uma função não linear. Devido a lentidão e demanda computacional, soluções fechadas têm sido sugeridas como alternativas, mas por necessitarem de certas simplificações no modelo da câmera, não conseguem oferecer resultados tão bons quanto os obtidos por minimização não linear. Existem também procedimentos de calibração em que os dois métodos são combinados, dessa forma a calibração é composta de dois passos, no primeiro os valores dos parâmetros iniciais são calculados linearmente e no segundo, os valores finais são obtidos com minimização não linear [16],

2.3.1 Método de Dois Passos

O método a ser descrito, desenvolvido por [17], requer apenas que a câmera observe um padrão planar mostrado em algumas (pelo menos duas) diferentes orientações. Tanto a câmera quanto o padrão planar podem ser movimentados livremente. O procedimento recomendado para calibração consiste em:

- Definir um padrão plano e fotografá-lo em diferentes posições, movendo-o pelo plano ou movimentando a câmera;
- Detectar o sistema de pontos da imagem;
- Estimar os parâmetros intrínsecos f, s_x, s_y, C_x e C_y e os parâmetros extrínsecos R e t, utilizando o primeiro passo do método;

Os parâmetros intrínsecos consistem na distância focal f, nos fatores de escala s_x e s_y (responsáveis pela conversão de milímetros para pixels), e nas coordenadas do centro da imagem Cx e Cy.

Os parâmetros extrínsecos consistem na matriz de rotação R e no vetor de translação t.

- Estimar os coeficientes de distorção radial resolvendo o método linear dos mínimos quadrados;
- Refinar os parâmetros por minimização.

A matriz K obtida a partir dos parâmetros calculados consiste na matriz de parâmetros intrínsecos ou matriz de projeção da câmera.

2.3.1.1 Homografia do Plano Modelo e sua Imagem

A partir da equação 2.5 e assumindo, sem perda de generalidade, que o plano do modelo encontra-se em $Z = 0$ do sistema global de coordenadas, pode-se estabelecer a seguinte relação:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = P \begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix} \tag{2.15}$$

Nesse caso, a matriz de projeção P deve levar em conta o fator de escala s_0. Em função da matriz de rotação R, composta por r_i colunas, a equação 2.15 pode ser reescrita como:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K \begin{bmatrix} r_1 & r_2 & r_3 & t \end{bmatrix} \begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix} = K \begin{bmatrix} r_1 & r_2 & t \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \tag{2.16}$$

Embora [16] utilize M e m para designar, respectivamente, os pontos localizados no plano modelo e na imagem, nesse documento será adotado X e x para preservar a notação utilizada nas seções 2.2.3 e 2.2.4. Como já mostrado na equação 2.15, quando $Z = 0$, o ponto em um plano modelo é representado por $X = \begin{bmatrix} X, Y \end{bmatrix}^T$ e por consequência, $\tilde{X} = \begin{bmatrix} X, Y, 1 \end{bmatrix}^T$. Portanto, o ponto X e sua imagem x são relacionados por uma matriz de homografia H:

$$s\tilde{x} = H\tilde{X}, \quad H = K \begin{bmatrix} r_1 & r_2 & t \end{bmatrix} \tag{2.17}$$

2.3.1.2 Restrições Sobre os Parâmetros Intrínsecos

Definindo a matriz de homografia como $H = \begin{bmatrix} h_1 & h_2 & h_3 \end{bmatrix}$, a equação 2.17 pode ser reescrita como $\begin{bmatrix} h_1 & h_2 & h_3 \end{bmatrix} = \lambda K \begin{bmatrix} r_1 & r_2 & t \end{bmatrix}$, em que λ é um escalar arbitrário.

A matriz de homografia possui 8 graus de liberdade e existem 6 parâmetros extrínsecos (3 correspondendo à rotação e os outros 3 à translação). Por isso, só é possível obter 2 restrições sobre os parâmetros intrínsecos, levando-se em conta o conhecimento de que r_1 e r_2 são ortonormais entre si,

$$h_1{}^T K^{-T} K^{-1} h_2 = 0 \tag{2.18}$$

$$h_1{}^T K^{-T} K^{-1} h_1 = h_2{}^T K^{-T} K^{-1} h_2 \tag{2.19}$$

Considerando $B = K^{-T}K^{-1}$ e utilizando a matriz K definida na equação 2.11,

$$B = \begin{bmatrix} \frac{1}{(fs_x)^2} & \frac{-s_\theta}{(fs_x)^2(fs_y)} & \frac{v_0 s_\theta - u_0(fs_y)}{(fs_x)^2(fs_y)} \\ \frac{-s_\theta}{(fs_x)^2(fs_y)} & \frac{s_\theta{}^2}{(fs_x)^2(fs_y)^2} + \frac{1}{(fs_y)^2} & \frac{-s_\theta(v_0 s_\theta - u_0(fs_y))}{(fs_x)^2(fs_y)^2} - \frac{v_0}{(fs_y)^2} \\ \frac{v_0 s_\theta - u_0(fs_y)}{(fs_x)^2(fs_y)} & \frac{-s_\theta(v_0 s_\theta - u_0(fs_y))}{(fs_x)^2(fs_y)^2} - \frac{v_0}{(fs_y)^2} & \frac{(v_0 s_\theta - u_0(fs_y))^2}{(fs_x)^2(fs_y)^2} + \frac{v_0{}^2}{(fs_y)^2} + 1 \end{bmatrix} \tag{2.20}$$

Seja $v_{ij} = \Big[h_{i1}h_{j1}, h_{i1}h_{j2} + h_{i2}h_{j1}, h_{i2}h_{j2}, h_{i3}h_{j1} + h_{i1}h_{j3}, h_{i3}h_{j2} + h_{i2}h_{j3}, h_{i3}h_{j3}\Big]^T$ um vetor de combinações lineares de h_{ij}, as duas equações de restrição 2.18 e 2.19 podem ser reescritas como 2 equações homogêneas em b:

$$\begin{bmatrix} v_{12}{}^T \\ (v_{11} - v_{22})^T \end{bmatrix} b = 0, \qquad b = \Big[B_{11}, B_{12}, B_{22}, B_{13}, B_{23}, B_{33}\Big]^T \tag{2.21}$$

Para n imagens do modelo plano observado, tem-se:

$$V_b = 0 \tag{2.22}$$

Em que V é uma matriz de dimensão $2n$ x 6. Para $n \geq 3$, tem-se geralmente uma única solução b definida pelo fator de escala.

2.3.1.3 Estimação Máxima de Verossimilhança

A solução acima pode ser obtida pela minimização algébrica de uma distância que não é fisicamente significativa e pode ser refinada pela inferência de máxima verossimilhança. Minimizando a função 2.23, obtém-se a estimação da máxima verossimilhança:

$$\sum_{i=1}^{n} \sum_{j=1}^{x} ||x_{ij} - \hat{x}(K, R_i, t_i, X_j)||^2 \tag{2.23}$$

Em que o termo $\hat{x}(K, R_i, t_i, X_j)$ é a projeção do ponto X_j na imagem i. A matriz de rotação R é parametrizada por um vetor de 3 parâmetros, denotado por r, paralelo ao eixo de rotação e cuja magnitude é igual ao ângulo de rotação.

A minimização da função 2.23 é um problema de minimização não linear e deve ser resolvida com o algoritmo de Levenberg-Marquardt, além de requerir um palpite inicial de $K, R_i, t_i | i = 1..n$ que pode ser obtido usando a técnica descrita na seção 2.3.1.2.

2.4 Calibração do Centro Óptico

As deduções a seguir foram retiradas de [18]. Considerando o modelo de câmera *pinhole*, em que a interseção do eixo óptico com o plano da imagem corresponde ao centro da imagem e assumindo que a mudança da configuração da distância da lente, proporciona uma alteração de distância focal de f para f', tem-se que:

$$\frac{X_f{}' - C_x}{X_f - C_x} = \frac{f_x{}'}{f_x} = \frac{Y_f{}' - C_y}{Y_f - C_y} = \frac{f_y{}'}{f_y} \tag{2.24}$$

, em que (X_f, Y_f) é a coordenada da imagem de uma característica específica antes da mudança da distância focal e $(X_f{}', Y_f{}')$, a coordenada da imagem da mesma característica após a mudança da distância focal.

Reorganizando os termos da equação 2.24 e assumindo que o *pinhole* viaja ao longo do eixo óptico quando a distância focal é variada, tem-se:

$$C_x(Y_f - Y_f{}') + C_y(Xf' - X_f) = X_f{}'Y_f - X_fY_f{}' \tag{2.25}$$

É possível encontrar o centro óptico (C_x, C_y) da equação 2.25, a partir de um sistema de equações com pelo menos duas características de imagens. Para reduzir o ruído, pode-se aumentar o número de características e resolver o sistema com o método dos mínimos quadrados, que consiste em uma técnica de otimização para minimizar a soma dos quadrados dos resíduos obtidos pela diferença entre os valores observados e os valores esperados, resultando em um melhor ajuste para um determinado conjunto de dados.

2.4.1 Sistema de Medição Baseado em Triangulação

Sistemas de medição baseados em triangulação consistem em configurações de dois receptores passivos ou de um transmissor e um receptor que, juntamente com o local da medição, formam um triângulo.

Na Figura 2.5, f é a distância focal da câmera; x_p, a posição do ponto de luz no plano de imagem da câmera; b, a distância da fonte de luz com relação a câmera e β, o ângulo do feixe de luz com respeito ao eixo x do $frame$ da câmera. As coordenadas do ponto medido, com respeito ao $frame$ de referência da câmera, são dadas por X_c e Z_c (assume-se a coordenada Y_c como zero) [4]. A partir do triângulo formado, tem-se:

$$tan\beta = \frac{Z_c}{b - X_c} \tag{2.26}$$

, isolando Z_c e substituindo X_c por seu correspondente apresentado na equação 2.2, tem-se:

$$Z_c = \frac{b}{\frac{1}{tan\beta} + \frac{x_p}{f}} \tag{2.27}$$

Sabendo que a distância b se mantém constante, nota-se que para o β suposto, o aumento de Z_c resulta em uma diminuição de X_c, isto é, o deslocamento do conjunto câmera/$laser$ no sentido

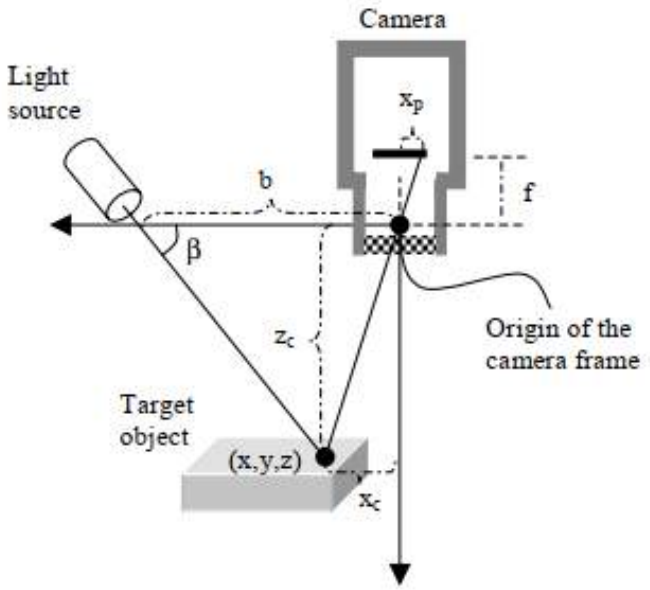


Figura 2.5: Exemplo de um sistema de medição baseado em triangulação [4].

contrário ao indicado pelo eixo da câmera, corresponde a uma aproximação entre a projeção do *laser* na imagem e o centro óptico da câmera. Em contrapartida, a diminuição de Z_c resulta em um aumento de X_c, isto é, o deslocamento do conjunto no sentido do eixo da câmera, resulta em um afastamento entre a projeção do *laser* na imagem e o centro óptico da câmera.

2.5 Detecção de Bordas

O objetivo da detecção de bordas é simplificar a análise de imagens reduzindo drasticamente a quantidade de dados processados, mas preservando a estrutura útil da informação sobre as bordas do objeto [19].

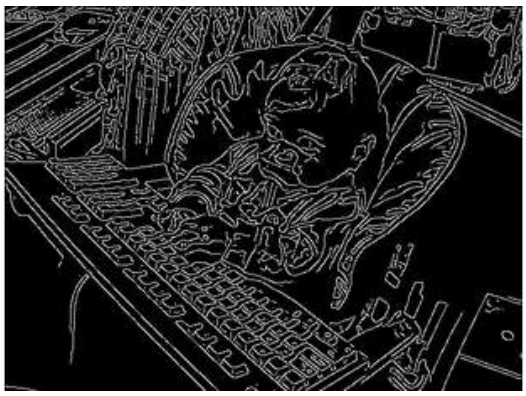
2.5.1 Detector de Bordas Canny

A literatura apresenta diversas formas para detectar bordas em imagens, dentre elas, o detector de bordas Canny, que é amplamente utilizado na visão computacional para localizar mudanças bruscas de intensidade e para encontrar bordas e limiares de objetos em uma imagem [20].

Esse algoritmo busca alcançar um método ótimo para detecção de bordas com base em três critérios:

1. Taxa de Erro: Deve existir uma baixa probabilidade de falha ao marcar os pontos reais de uma borda e uma baixa probabilidade de marcar falsamente pontos não pertencentes à borda. Esse critério corresponde à maximização da relação sinal/ruído, que por sua vez implica na minimização do ruído.

(a) Imagem original sem processamento



(b) Imagem após detecção de borda

Figura 2.6: Detector de borda Canny [5].

2. Localização: Deve-se minimizar a distância entre os pontos marcados como bordas pelo detector e os pontos que determinam bordas reais.

3. Resposta: O detector deve obter uma única resposta para uma única borda. Esse critério está implícito no primeiro, já que quando existem 2 respostas para a mesma borda, uma delas deve ser considerada falsa. No entanto, a formulação matemática do primeiro critério não considera o requisito de múltiplas respostas, sendo necessário explicitá-lo.

Segundo [21], a implementação desse detector pode ser descrita em 5 etapas:

1. Suavizar o ruído na imagem a partir de um filtro gaussiano apropriado, o que resulta em uma redução dos detalhes da imagem, deixando-a mais desfocada.

2. Determinar o gradiente de magnitude e direção de cada pixel.

3. Se o gradiente de magnitude no pixel em análise for maior que nos seus vizinhos na direção do gradiente, deve-se marcar o pixel como borda. Caso contrário, o pixel é setado em 0.

4. Submeter os pixels restantes ao limiar de histerese, que utiliza dois limiares, um alto e um baixo. Todo pixel com um valor acima do limiar alto é marcado como borda forte e todo pixel abaixo do limiar baixo é setado em 0. Todos os pixels entre os dois limiares são marcados como bordas fracas.

5. O passo final é conectar as bordas. Todos pixels bordas fortes são bordas e os pixels bordas fracos são considerados bordas apenas se estiverem ligados a pixels bordas fortes. Para encontrar todas as bordas, pode-se utilizar algoritmos de Busca em Largura (BFS) ou Busca em Profundidade (DFS).

2.5.2 Transformada de Hough

Como apresentado em [21], transformada de Hough é uma forma de detectar linhas em imagens, mas pode ser usada também para detectar qualquer estrutura cuja equação paramétrica seja

conhecida. É considerada um detector robusto sob ruído e oclusão parcial.

Para detectar linhas em imagens, deve-se localizar os conjuntos de pixels que formam linhas retas na imagem. Isso pode ser feito após a aplicação de um detector de bordas, trabalhando assim apenas com os pixels das bordas e obtendo os pixels que formam linhas dentro desse conjunto de pixels.

Existem infinitas retas que passam por um ponto (x_i, y_i), sendo possível definir uma reta que passa por esse ponto como:

$$y_i = ax_i + b \tag{2.28}$$

Usando isso, podemos transformar cada pixel no espaço parametrizado $a - b$, reescrevendo a equação como:

$$b = -ax_i + y_i \tag{2.29}$$

Essa equação representa uma reta no espaço $a-b$ e cada ponto (a, b) presente na reta representa uma possível reta passando pelo ponto (x_i, y_i). Então, cada pixel (x_i, y_i) no conjunto de pixels bordas deve ser transformado no espaço $a - b$ para se obter uma reta.

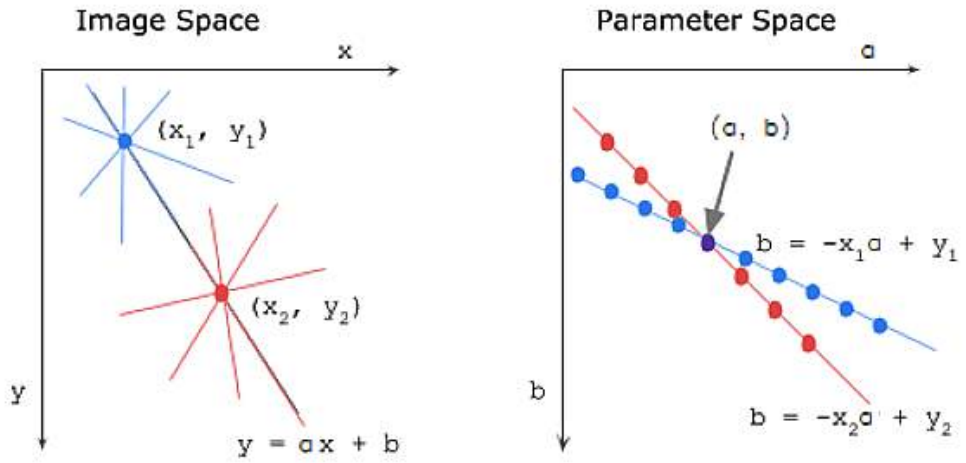


Figura 2.7: Transformação do espaço $x - y$ original para o espaço parametrizado $a - b$, adaptado de [6].

A interseção entre as linhas no espaço $a - b$ representa os valores a, b da linha $y_i = ax_i + b$ que passa por esses pontos, como observado na Figura 2.7.

O espaço $a-b$, entretanto, é limitado e não consegue representar linhas verticais. Esse problema pode ser resolvido utilizando coordenadas polares para representar as linhas. Supondo um conjunto $(x_1, y_1), .., (x_n, y_n)$ de n pontos de uma imagem, os quais se deseja saber o conjunto de linhas que se ajustam a eles, deve-se transformar os pontos (x_i, y_i) em curvas senoidais no plano $\theta - \rho$ definido por:

$$x_i cos\theta + y_i sin\theta = \rho \tag{2.30}$$

Restringindo o intervalo de θ para $[0, \pi]$, os parâmetros para uma linha tornam-se únicos. Com essa restrição, toda linha no plano $x - y$ corresponde a um único ponto no plano $\theta - \rho$.

Nota-se pela Figura 2.8 que as curvas correspondentes aos pontos colineares da imagem têm um ponto de interseção comum. Esse ponto, no plano $\theta - \rho$, dado por (θ_0, ρ_0), define a linha que passa por esses pontos colineares. Então o problema de detectar pontos colineares pode ser convertido no problema de achar curvas concorrentes.

Suponha um conjunto de pontos no plano $\theta - \rho$,todos situados na curva $\rho = x_i cos\theta + y_i sin\theta$. Então é fácil mostrar que esses pontos correspondem a linhas no plano $x - y$ passando pelo ponto (x_i, y_i) [22].

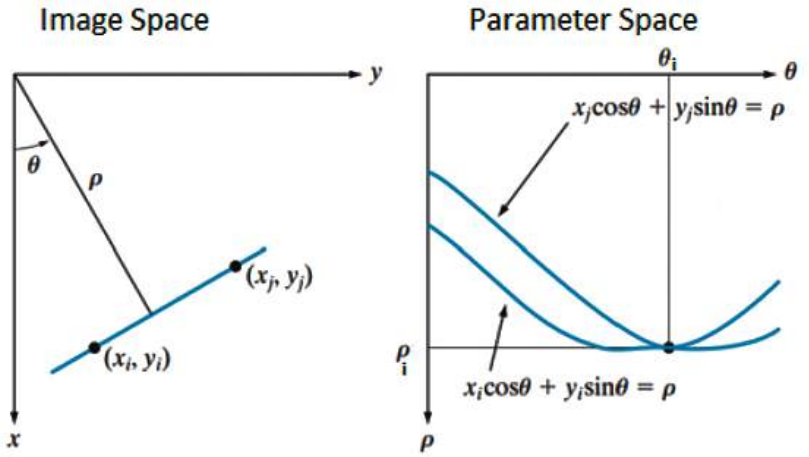


Figura 2.8: Transformação do espaço $x - y$ original para o espaço parametrizado $\theta - \rho$, adaptado de [7].

2.6 Controle Servo Visual

Controle servo visual é uma das técnicas adotada atualmente para o controle de manipuladores robóticos. Ele se baseia em dados de visão computacional coletados a cada ciclo do controle a fim de corrigir a posição do manipulador até que a posição desejada seja atingida, configurando um sistema de malha fechada.

Os aspectos que devem ser levados em consideração nesse tipo de controle são a posição da câmera com relação ao robô, isto é, se ela se movimenta junto com o robô ou se está fixada em um ponto externo (Eye-in-Hand ou Eye to Hand); e a quantidade de câmeras no sistema, pois este pode ser monocular ou binocular "estéreo" [23].

2.6.1 Controle Servo Visual Baseado em Imagem

No controle servo visual baseado em imagem (IBVS, do inglês image-based visual servoing) o usuário deve selecionar conjuntos de características s e s^*, que correspondem, respectivamente, a coordenadas específicas da imagem real e da imagem esperada após o deslocamento, e a partir da comparação entre elas, deve-se gerar o sinal de erro. Entretanto, esse sinal é gerado no espaço da

imagem e precisa ser transformado pro espaço da tarefa através de uma matriz de interação de pontos ou jacobiano da imagem [24].

O jacobiano da imagem J_s se relaciona com s e desempenha um papel crucial na lei de controle, sendo descrito por:

$$\dot{s} = \left(\frac{\delta s}{\delta r}\right)\frac{dr}{dt} = J_s v_c \tag{2.31}$$

, onde $v_c = \frac{dr}{dt} = [T^T, \Omega^T]^T$ é a velocidade da câmera.

2.6.1.1 Matriz Jacobiano da Imagem

As deduções a seguir foram retiradas de [25, 24], porém modificadas para coordenadas em pixels ao invés de permanecer nas unidades métricas. O jacobiano da imagem J_s relaciona o movimento da câmera em relação a um referencial global com as variações das características no referencial da imagem. O sistema de coordenadas da imagem (u, v) se relaciona com o sistema de coordenadas espaciais (X, Y) conforme as equações 2.7. Combinando com as equações 2.2, obtem-se:

$$\begin{aligned} u &= s_x f\frac{X}{Z} + u_0 \\ v &= s_y f\frac{Y}{Z} + v_0 \end{aligned} \tag{2.32}$$

A velocidade da câmera v_c em função de um referencial externo é expressa por:

$$\begin{aligned} \dot{X} &= -T_x - \omega_y Z + \omega_z Y \\ \dot{Y} &= -T_y - \omega_z X + \omega_x Z \\ \dot{Z} &= -T_z - \omega_x Y + \omega_y X \end{aligned} \tag{2.33}$$

, em que $T(t) = [T_x(t), T_y(t), T_z(t)]^T$ e $\Omega = [\omega_x(t), \omega_y(t), \omega_z(t)]^T$ compõem o vetor velocidade de forma translacional e angular, respectivamente.

A variação das coordendas de imagem, representadas na equação 2.32, em função do tempo é dada por:

$$\begin{aligned} \dot{u} &= \frac{s_x f\dot{X} - (u - u_0)\dot{Z}}{Z} \\ \dot{v} &= \frac{s_y f\dot{Y} - (v - v_0)\dot{Z}}{Z} \end{aligned} \tag{2.34}$$

Substituindo 2.33 em 2.34:

$$\begin{aligned} \dot{u} &= -s_x f\frac{T_x}{Z} + (u - u_0)\frac{T_z}{Z} + (v - v_0)(u - u_0)\frac{\omega_x}{s_y f} - \left(s_x f + \frac{(u - u_0)^2}{s_x f}\right) w_y + (v - v_0)\frac{s_x \omega_z}{s_y} \\ \dot{v} &= -s_y f\frac{T_y}{Z} + (v - v_0)\frac{T_z}{z} + \left(s_y f + \frac{(v - v_0)^2}{s_y f}\right)\omega_x - (u - u_0)(v - v_0)\frac{\omega_y}{s_x f} - (u - u_0)\frac{s_y \omega_z}{s_x} \end{aligned} \tag{2.35}$$

, encontra-se a velocidade dos pontos da imagem em função da velocidade da câmera.

A partir das equações 2.35 e 2.31, deduz-se a matriz L_s como:

$$J_s(u,v,Z) = \begin{bmatrix} \frac{-s_x f}{Z} & 0 & \frac{\overline{u}}{Z} & \frac{\overline{vu}}{s_y f} & -\left(s_x f + \frac{\overline{u}^2}{s_x f}\right) & \frac{\overline{v} s_x}{s_y} \\ 0 & -\frac{s_y f}{Z} & \frac{\overline{v}}{Z} & s_y f + \frac{\overline{v}^2}{s_y f} & -\frac{\overline{uv}}{s_x f} & -\frac{\overline{u} s_y}{s_x} \end{bmatrix} \tag{2.36}$$

Em que $\overline{u} = u - u_0$ e $\overline{v} = v - v_0$. Deve-se lembrar que para pixels quadrados, $s_x = s_y$, resultando em alguns cancelamentos na matriz acima.

J_s é considerada uma estimativa $\hat{J}_s$ do valor real. Considerando $\hat{J}_s = J_s^*$, $\hat{J}_s$ se mantém constante durante a tarefa de controle e o seu valor é calculado pela matriz de interação J_s na configuração desejada. A garantia de estabilidade nesse caso só pode ser garantida na vizinhança da posição desejada e a trajetória é pouco restrita, podendo ocorrer a perda de pontos no campo visual.

2.6.1.2 Movimentação de Três pontos

O jacobiano da imagem precisa ter dimensão 6x6 para satisfazer a equação 2.31, dessa forma:

$$\begin{bmatrix} \dot{u_1} \\ \dot{v_1} \\ \dot{u_2} \\ \dot{v_2} \\ \dot{u_3} \\ \dot{v_3} \end{bmatrix} = \begin{bmatrix} J_s(u_1, v_1, Z_1) \\ J_s(u_2, v_2, Z_2) \\ J_s(u_3, v_3, Z_3) \end{bmatrix} \begin{bmatrix} T_x \\ T_y \\ T_z \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \tag{2.37}$$

Isolando o vetor de velocidade da câmera em função do vetor de velocidade dos pixels, tem-se:

$$\begin{bmatrix} T_x \\ T_y \\ T_z \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} J_s(u_1, v_1, Z_1) \\ J_s(u_2, v_2, Z_2) \\ J_s(u_3, v_3, Z_3) \end{bmatrix}^{-1} \begin{bmatrix} \dot{u_1} \\ \dot{v_1} \\ \dot{u_2} \\ \dot{v_2} \\ \dot{u_3} \\ \dot{v_3} \end{bmatrix} \tag{2.38}$$

A partir da lei de controle servo visual [26], a velocidade desejada para as coordenadas em pixel, em função das coordenadas de pixel atuais (u_i, v_i) e das coordenadas que se desejam ser alcançadas $({u_i}^*, {v_i}^*)$ pode ser escrita como:

$$\begin{bmatrix} \dot{u}_i^* \\ \dot{v}_i^* \end{bmatrix} = \lambda \left\{ \begin{bmatrix} {u_i}^* \\ {v_i}^* \end{bmatrix} - \begin{bmatrix} u_i \\ v_i \end{bmatrix} \right\} \tag{2.39}$$

,em que λ é um escalar positivo.

Para simplificar, pode-se escrever $s_i = [u_i, v_i]^T$ e a equação 2.38 passa a ser escrita da seguinte forma:

$$\begin{bmatrix} {T_x}^* \\ {T_y}^* \\ {T_z}^* \\ {\omega_x}^* \\ {\omega_y}^* \\ {\omega_z}^* \end{bmatrix} = \lambda \begin{bmatrix} J_s(u_1, v_1, Z_1) \\ J_s(u_2, v_2, Z_2) \\ J_s(u_3, v_3, Z_3) \end{bmatrix}^{-1} \begin{bmatrix} {s_1}^* - s_1 \\ {s_2}^* - s_2 \\ {s_3}^* - s_3 \end{bmatrix} \tag{2.40}$$

Obtendo-se, assim, a velocidade da câmera necessária para que haja o deslocamento entre os pontos s_i e ${s_i}^*$.

2.7 Visão Geral do Projeto

A partir da revisão bibliográfica apresentada nesse capítulo, determinou-se a fundamentação teórica necessária para implementação dos códigos referentes ao projeto. A câmera será calibrada pelo Método de Dois Passos, devido a sua proposta de simplificação e agilidade na obtenção dos parâmetros da câmera. O centro da imagem deverá ser calculado por meio do procedimento apresentado na Seção 2.4, devido a sua simplicidade e concordância com o modelo de câmera utilizado. A detecção das linhas nas imagens ocorrerá com a aplicação da Transformada de Hough e o deslocamento do manipulador robótico será calculado com o algoritmo IBVS. Por fim, a projeção do *laser* no centro da imagem será obtida com base na triangulação do sistema de medição utilizado.

Capítulo 3

Desenvolvimento

3.1 Introdução

Serão apresentados nesse capútlo os equipamentos utilizados para a construção do sistema de medição baseado em triangulação, os códigos implementados para atingir o alinhamento do *laser* com a câmera e a plataforma utilizada para obter os ângulos de junta do manipulador robótico.

3.2 Equipamentos

A câmera utilizada no projeto é do modelo *acA2500-14µm*, fabricado pela empresa *Basler*. Ela dispõe de uma interface *USB 3.0*, um sensor *ON Semiconductor MT9P031 mono CMOS* e consegue entregar 14 frames por segundo com uma resolução de 5MP [27]. Acoplada na câmera, utilizou-se uma lente do modelo *CFFL F1.4 f16mm 2/3"*, que possui uma distância focal fixa de 16mm e um intervalo de abertura de $F1.4 - F16$ [28].

(a) Câmera $acA2500 - 14\mu m$ [27].

(b) Lente *CFFL F1.4 f16mm 2/3"* [28].

(c) *Laser* $LDM115$ e gerador de linha óptico $LGO115$ [29].

Figura 3.1: Equipamentos utilizados para construção do sensor.

Utilizou-se também o módulo *Laser LDM115*, de 11mm de diâmetro e comprimento de onda 633nm com uma alimentação de 5mW como padrão [30]. Junto ao laser, utilizou-se o gerador de linha óptico *LGO115*, a fim de converter a saída do laser em um feixe de 28^o, cuja projeção no plano equivale a uma linha reta [31].

Tabela 3.1: Especificações da Câmera

Especificações da Câmera	
Resolução (HxV pixels)	2592 x 1944
Tipo do Sensor	ON Semiconductor MT9P031 Progressive scan CMOS Rolling shutter
Tamanho do Sensor óptico	$1/25''$
Tamanho Efetivo na Diagonal	7.2mm
Tamanho do Pixel (HxV)	2.2μm x 2.2μm
Taxa de Quadros Padrão	14fps
Mono/Color	Mono
Interface	USB 3.0, Taxa máx. 5Gbit/s
Alimentação	5VDC, via USB
Dimensões	29.3mm x 29mm x 29mm (sem lentes ou conectores)
Peso	$<$ 80g
Software	Basler pylon Camera Software Suite (versão 4.0 ou maior)

Tabela 3.2: Especificações da Lente

Especificações da Lente	
Distância Focal	16.0mm
Abertura	$F1.4 - F16.0$
Tipo de Abertura	Manual
Círculo de Imagem Máximo	2/3”
Distância de Trabalho	100mm
Peso	74g

Tabela 3.3: Especificações do Laser

Especificações d *Laser*	
Comprimento de Onda	633nm
Diâmetro do Feixe	4x2mm
Intervalo do foco	35mm$-\infty$
Dimensões	37mm de comprimento e 11mm de diâmetro
Classe do *Laser* - CDRH	IIIa
Cor	Vermelho
Precisão de Apontamento	$<$ 25mrad
Corrente de Operação	$<$ 70mA
Alimentação	$3.5 - 5$VDC
Peso	9.5g

O manipulador robótico *ABB IRB 140* escolhido para o projeto é um robô industrial multi-funcional que possui 6 eixos e um alcance de 810mm até o eixo 5 [8].

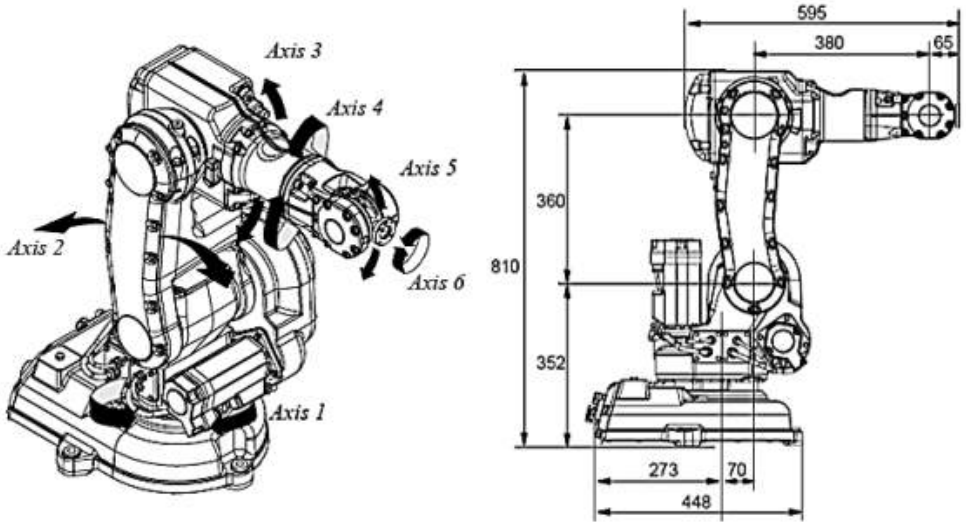
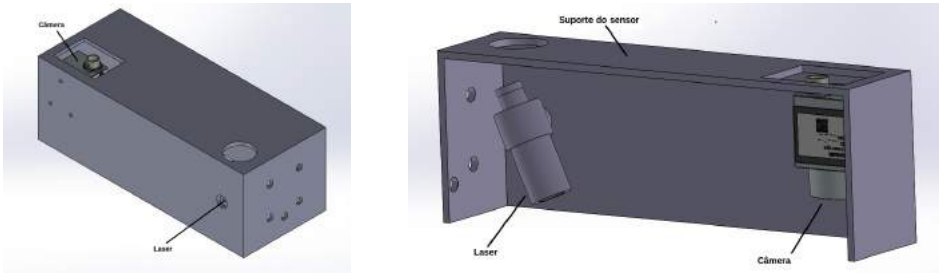


Figura 3.2: Desenho técnico do robô industrial *ABB IRB 140* em milímetros [8].

3.3 Modelagem do Sensor

O sensor consiste em um sistema de medição baseado em triangulação, composto por uma câmera e um *laser*, acoplado na flange do 6^o eixo do manipulador robótico. Ele foi projetado como uma caixa de volume 200x80x70mm^3, no $software$ de modelagem 3D SolidWorks®.

Para o modelo de triangulação escolhido, o *laser* precisa estar inclinado em relação a câmera, por essa razão, desenvolveu-se um mecanismo de ajuste manual para garantir a movimentação angular do dispositivo. Esse mecanismo consiste em um fuso, que ao ser apertado, permite que o laser fique fixo na posição em que se encontra e ao ser afrouxado, permite a alteração da sua inclinação. Nesse modelo, a câmera não deve se movimentar, por isso ela foi fixada diretamente na caixa.



(a) Vista isométrica da caixa com a câmera e o *laser*.

(b) Vista cortada por um plano para facilitar visualização.

Figura 3.3: Modelo 3D do sensor

3.4 Funções Utilizadas

As funções listadas a seguir podem ser melhor detalhadas consultando a ferramenta *help* do MATLAB®.

1. *imageLocation = fullfile(filepart1,...,filepartN)*

   Cria uma especificação completa do arquivo a partir da pasta e dos nomes dos arquivos disponibilizados.

2. *imgSet = imageSet(imageLocation)*

   Retorna o objeto *imgSet* para armazenamento de um conjunto de dados de uma imagem ou de uma coleção de imagens. O objeto contém descrição das imagens, localização das imagens e o número de imagens do conjunto.

3. *sz = size(A)*

   Retorna um vetor linha cujos elementos são as dimensões de A.

4. *[worldPoints] = generateCheckerboardPoints(boardSize,squareSize)*

   Retorna a matriz $M[x, y]$ contendo as coordenadas dos cantos dos quadrados de um padrão xadrez. A coordenada $[0, 0]$ corresponde ao canto direito inferior do quadrado superior esquerdo do padrão. Para o caso de um padrão de pontos circulares, as coordenadas retornadas são referentes aos centroides dos pontos e $[0, 0]$ corresponde ao centroide do ponto superior esquerdo do padrão.

5. *A = imread(filename)*

   Lê a imagem especificada pelo $filename$ e retorna a matriz A. O formato do arquivo é inferido pelo seu conteúdo.

6. *I = rgb2gray(RGB)*

   Converte uma imagem RGB para uma imagem em escala de cinza, eliminando as informações de matiz e saturação enquanto se mantém a luminância.

7. *BW = im2bw(A,level)*

   Converte a imagem A, em escala de cinza, para uma imagem binária. Todos os pixels com luminância maior que o nível indicado pelo parâmetro *level* tornam-se 1 (branco), o restante torna-se 0 (preto). O parâmetro *level* varia entre 0 e 1, conforme o sinal possível de cada classe de imagem.

8. *[centers,radii] = imfindcircles (A,radiusRange,Name,Value,...)*

   Encontra círculos com raio dentro do intervalo especificado por *radiusRange*. O argumento de saída *radii* contém uma estimação do raio correspondente para cada centroide encontrado e armazenado em *centers*. Pode-se especificar a característica de luminosidade do círculo a ser detectado com *dark* ou *bright* e aumentar a sensibilidade da detecção com *Sensitivity*, que varia entre 0 e 1.

9. *viscircles(centers,radii)*

   Desenha círculos com os centros e raios especificados.

10. B = *sortrows(A)*

    Ordena de forma crescente as linhas da matriz, com base nos elementos da primeira coluna. Para valores repetidos nas colunas analisadas, a função ordena com base nos valores da próxima coluna. Para especificar a coluna base para ordenação, basta acrescentar sua posição como argumento de entrada.

11. *[cameraParams,imagesUsed,estimationErrors] = estimateCameraParameters(imagePoints,worldPoints)*

    Retorna o objeto $cameraParameters$, que contém as estimações dos parâmetros intrínsecos e extrínsecos e a distorção dos coeficientes da câmera. A função retorna também as imagens usadas para estimar os parâmetros da câmera e os erros de estimativa padrão da câmera. O método de calibração utilizado por essa função foi explicado na Seção 2.3, porém deve-se levar em conta a transposição das matrizes e alteração da ordem dos fatores ao interpretar os resultados dados pela variável de saída $cameraParams$.

    A matriz de câmera *pinhole* ideal apresentada na equação 2.5, após as transposições, passa a ser descrita como:

$$\tilde{x}^T = \tilde{X}^T P_1 \tag{3.1}$$

    Em que P_1 é a matriz da câmera, representada por:

$$P_1 = \begin{bmatrix} R^T & t^T \end{bmatrix} K^T \tag{3.2}$$

12. $x = A \backslash B$

    Resolve o sistema linear de equações $Ax = B$, em que as matrizes A e B devem ter o mesmo número de linhas, com base no método dos mínimos quadrados.

13. M = *mean(A)*

    Retorna a média dos elementos do vetor A.

14. B = *imtranslate(A,translation)*

    Translada a imagem A a partir das dimensões especificadas no vetor *translation*.

15. J = *imcrop(I,rect)*

    Recorta a imagem I de acordo com a posição e dimensão do retângulo, especificadas em *rect*. Esse vetor é da forma $[x, y, width, height]$ e especifica o tamanho e posição do recorte da imagem. A imagem recortada contém todos os pixels da imagem de entrada que estão parcialmente ou completamente englobados pelo retângulo de corte.

16. J = *imcomplement(I)*

    Calcula o complemento da imagem I e retorna o resultado em J. No complemento da imagem binária, os pixels que estão em 0 (preto) tornam-se 1 (branco) e vice-versa.

17. *[H,theta,rho] = hough(BW,Name,Value,...)*

    Calcula a Transformada de Hough da imagem binarizada BW. A função Hough é designada para detectar linhas a partir da equação de parametrização 2.30. A função retorna os parâmetros *rho* e *theta*, que correspondem, respectivamente, à distância da origem até a linha ao longo do vetor perpendicular à linha e *theta*, ao ângulo em graus entre o eixo x e esse vetor, conforme pode-se observar na Figura 2.8. A função também retorna a matriz espacial H, que é uma matriz espacial cujas linhas e colunas correspondem aos valores de *rho* e *theta*, respectivamente.

18. *peaks = houghpeaks(H,numpeaks, Name,Value,...)*

    Localiza os picos da matriz H, gerada pela função *hough*. *numpeaks* especifica o número máximo de picos que serão identificados. A função retorna uma matriz contendo as coordenadas de linha e coluna dos picos.

19. *lines = houghlines(BW,theta,rho,peaks, Name,Value,...)*

    Extrai segmentos de linha, na imagem BW, associados a posições específicas na Transformada de Hough. *theta* e *rho* são vetores retornados pela função *hough*. *peaks* é a matriz que a função *houghpeaks* retorna, contendo as coordenadas de linha e coluna das posições específicas na Transformada de Hough usadas para procurar os segmentos de linha. A variável de retorno *lines* é um vetor de tamanho igual ao número de segmentos de linha encontrados.

20. *p = polyfit(x,y,n)*

    Retorna os coeficientes de um polinômio $p(x)$ de grau n da melhor curva para os dados especificados em y. Os coeficientes do polinômio p são dispostos da seguinte forma:

$$p(x) = p_1 x^n + p_2 x^{n-1} + ... + p_n x + p_{n+1} \tag{3.3}$$

21. *D = pdist(X)*

    Retorna a distância euclidiana entre os pares observados no vetor X.

22. *J = imresize(I,scale)*

    Retorna a imagem J com a escala alterada. Para $scale = 1.1$, por exemplo, a imagem J resultante é 10% maior que a imagem I.

3.5 Calibração da Câmera

A calibração da câmera consistiu em uma análise das várias imagens capturadas de um alvo de pontos, a fim de se obter os parâmetros intrínsecos e extrínsecos.

O padrão escolhido para o projeto, mostrado na Figura 3.4, é composto por 49 pontos dispostos em 7 linhas e 7 colunas, espaçados entre si em 26.5mm. A circunferência maior possui diâmetro de 20mm e as menores, 12mm.

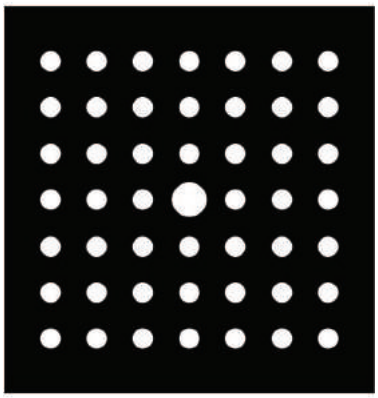


Figura 3.4: Padrão de pontos 7x7.

O código implementado para essa etapa, apresentado na Seção I.1 do Apêndice, pode ser resumido nos seguintes passos:

- Especificação do diretório no qual estão localizados as imagens e os códigos desenvolvidos e determinação do objeto contendo os dados das imagens lidas;

  Para especificação do diretório, utiliza-se a função 1 e para a determinação do objeto, a função 2.

- Detecção dos pontos em coordenadas globais a partir das características do padrão escolhido;

  A definição do parâmetro *boardSize* é feita com base na quantidade de linhas e colunas do padrão escolhido. Já o parâmetro *DistanceInMM*, com base na distância em milímetros entre os centros dos pontos. Para obter os pontos em coordenadas globais, utiliza-se esses parâmetros de entrada na função 4.

- Detecção dos pontos em coordenadas de pixel

  O primeiro passo para essa detecção é a leitura e binarização das imagens, realizadas a partir das funções 5 e 6;

  Em seguida, utiliza-se um filtro morfológico para identificar os centroides e raios dos pontos com a função 8;

  Por fim, ordena-se o vetor de pontos com a função 10 para poder usá-lo como parâmetro de entrada da função 11;

- Estimação dos parâmetros extrínsecos e intrínsecos da câmera;

  Utiliza-se os pontos em coordenadas de pixel e em coordenadas globais para estimar os parâmetros da câmera com a função 11.

- Apresentação visual dos resultados

As coordenadas detectadas e as coordenadas obtidas após a estimação dos parâmetros da câmera são plotadas nas imagens em escala de cinza.

Plota-se um gráfico com os erros de estimação obtidos para cada imagem.

A sequência de passos descritos também pode ser verificada no fluxograma da Figura 3.5:

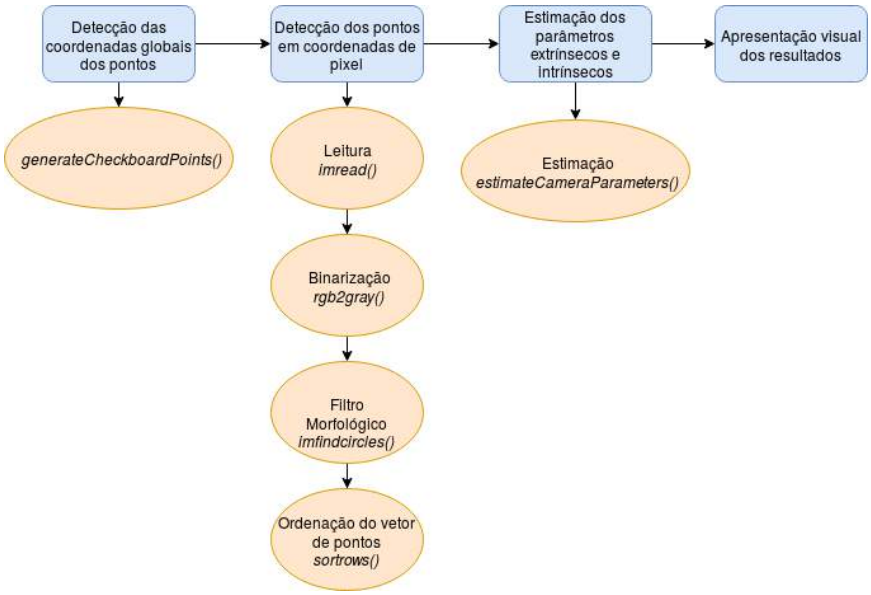


Figura 3.5: Etapas para a calibração da câmera.

3.6 Sistema de Medição Baseado em Triangulação

O alinhamento entre o *laser* e o centro do alvo pode ser obtido a partir do modelo de triangulação apresentado na seção 2.4.1, contanto que o centro do alvo coincida com o centro óptico da câmera e Z_c esteja paralelo ao eixo Z da câmera.

Para habilitar o uso desse modelo, implementou-se um código para obtenção do centro óptico da câmera sem a necessidade de calibrar a câmera propriamente dita. Em seguida, desenvolveu-se um código para alinhamento do centro óptico da câmera com o centro do alvo e um código para o cálculo da distância Z_c em que o sistema deve se mover para que o *laser* se aproxime do alvo. Implementou-se também um código para que a projeção do laser se alinhe com alguma das linhas do centro do alvo. Todos os códigos foram reunidos em um código principal.

3.6.1 Rotinas de Funcionamento

Elaborou-se um código principal com as rotinas de funcionamento do procedimento de alinhamento do sensor visual com o alvo de referência. Nele foram agrupados os códigos implementados e pode ser resumido conforme a Figura 3.6 e os passos à seguir:

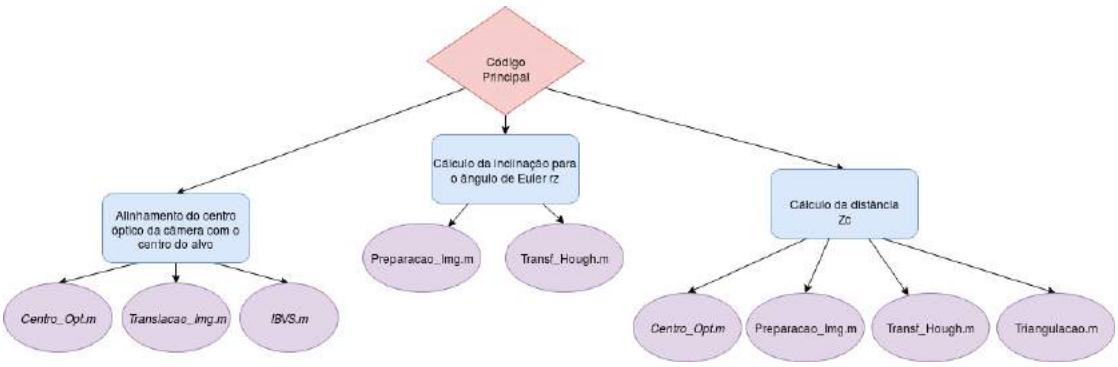


Figura 3.6: Rotina de funcionamento para os códigos implementados.

- Especificação do diretório no qual estão localizados as imagens e os códigos desenvolvidos e determinação do objeto contendo os dados das imagens lidas;

  Para especificação do diretório, utiliza-se a função 1 e para a determinação do objeto, a função 2.

- Códigos chamados pelo código principal e seus objetivos:

  Centro_Opt.m calcula o centro óptico da câmera;

  Translacao_Img.m calcula o deslocamento a ser realizado para alinhar o centro do alvo com o centro óptico (centro da imagem);

  IBVS.m calcula as posições de junta do efetuador terminal para que ocorra o deslocamento;

  Preparacao_Imgṁ binariza as imagens com e sem o laser, além de realizar um recorte nessa última para posterior análise;

  Transf_Hough.m identifica as retas que definem o laser e as linhas do alvo;

  Triangulacao.m calcula a distância Z_c.

- Apresentação visual dos resultados

  Após o cálculo da posição do efetuador terminal, projeta-se na imagem original transladada os pontos antes e depois do deslocamento;

  Após a determinação das linhas utilizando a Transformada de Hough, plota-se as linhas do *laser* e do alvo na imagem original, bem como a menor reta que liga o laser ao centro do alvo.

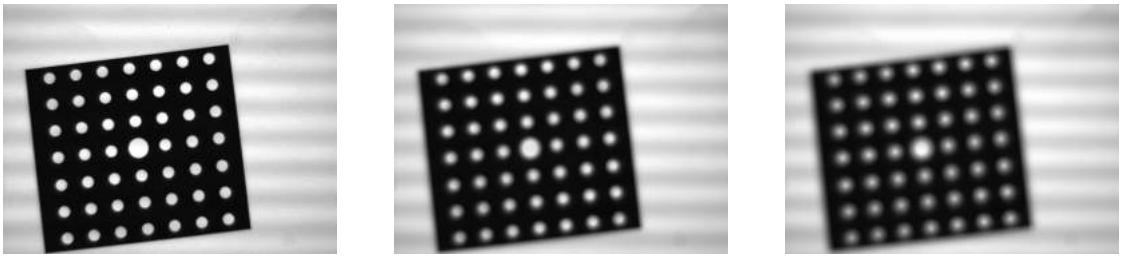
  Após identificação da linha do *laser* na imagem com escala alterada, plota-se essa linha na imagem original para análise do deslocamento.

O processo é feito de forma iterativa, então a cada movimentação do robô, novas fotos devem ser capturadas e uma nova análise deve ser feita. Primeiro deve-se deslocar o eixo da câmera para que o centro óptico projetado na imagem coincida com centro do alvo, habilitando os códigos $Centro\_Opt.m$, $Translacao\_Img.m$ e $IBVS.m$. Em seguida, para a nova posição estabelecida e

novas imagens capturadas, pode-se calcular a inclinação da reta com relação ao alvo, habilitando os códigos $Prepara\_Img.m$ e $Transf\_Hough$ ou calcular a distância Z_c, habilitando os códigos $Centro\_Opt.mO$, $Preparacao\_Img.m$, $Transf\_Hough.m$ e $Triangulacao.m$.

3.6.2 Calibração do Centro Óptico

A calibração da câmera foi implementada no Trabalho de Graduação 1, porém durante a implementação do Trabalho de Graduação 2, notou-se que não era necessário o conhecimento de todos os parâmetros intrínsecos e extrínsecos da câmera para realização do projeto, apenas o conhecimento do centro óptico da câmera em coordenadas da imagem. A fim de facilitar o procedimento, implementou-se um código com base nas deduções matemáticas apresentadas na Seção 2.4.



(a) (b) (c)

Figura 3.7: Alvos em diferentes distâncias focais.

O procedimento consiste em capturar algumas imagens de um mesmo padrão de pontos, teoricamente posicionados no mesmo local, alterando-se apenas o foco da lente da câmera. Em seguida deve-se detectar os pontos e como se relacionam após as mudanças de foco.

O código apresentado na Seção I.2.1 do Apêndice pode ser resumido conforme a Figura 3.8 e os passos à seguir:

- Detecção dos centroides dos pontos circulares em coordenadas de pixel;

  O primeiro passo para essa detecção consiste na leitura e binarização das imagens, realizadas a partir das funções 5 e 7;

  Em seguida, utiliza-se um filtro morfológico para identificar os centroides e raios dos pontos com a função 8. As coordenadas e raios encontrados são plotados na imagem binarizada utilizando a função 9;

  Por fim, ordena-se o vetor de pontos com a função 10 para facilitar a utilização dos dados.

- Determinação dos sistemas de equações, de forma matricial, relacionando as coordenadas em pixels dos pontos obtidos em diferentes distâncias focais, conforme apresentado na equação 2.25;

  Sistema 1: Equações para análise dos 49 pontos das Figuras 3.7(a) e 3.7(b);

Sistema 2: Equações para análise dos 49 pontos das Figuras 3.7(b) e 3.7(c);

Sistema 3: Equações para análise dos 49 pontos das Figuras 3.7(a) e 3.7(c);

Sistema 4: Sistemas 1 e 2;

Sistema 5: Sistemas 1 e 3;

Sistema 6: Sistemas 2 e 3;

Sistema 7: Sistemas 1 e 2 e 3;

- Solução dos sistemas de equações aplicando o método dos mínimos quadrados oferecido pela função 12;
- Cálculo da média aritmética dos valores encontrados para o centro óptico a partir da função 13.

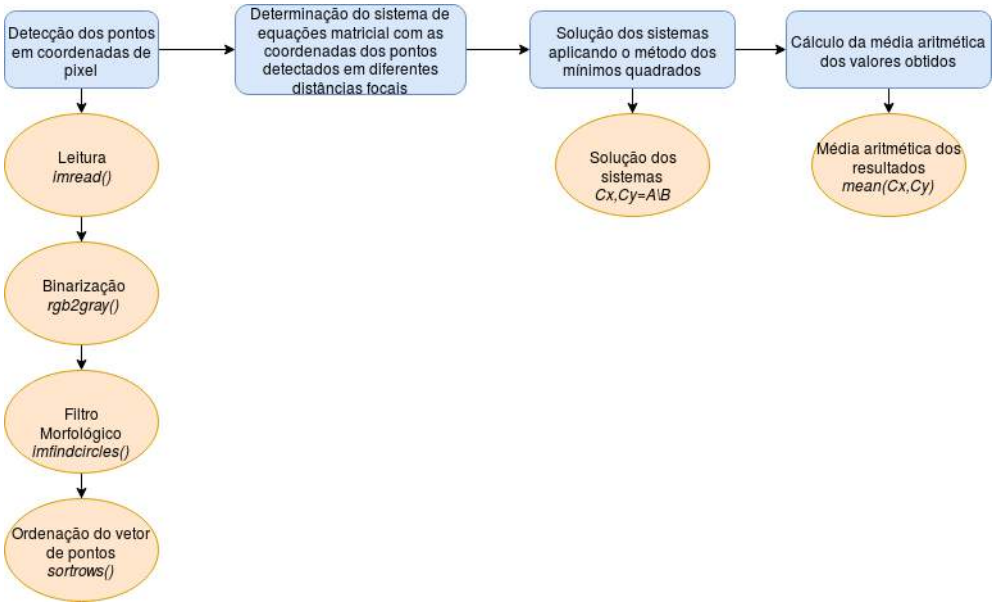


Figura 3.8: Etapas para calibração do centro óptico.

3.7 Alinhamento do Centro do Alvo com o Centro da Imagem

O procedimento consistiu em capturar uma imagem do alvo e em seguida, a partir das coordenadas em pixel do centro óptico calculado na Seção 3.6.2 com um sistema de equações utilizando o que foi definido em 2.25, aplicar o código desenvolvido para realizar o alinhamento entre o centro do alvo e o centro óptico da câmera na imagem.

O código apresentado na Seção I.2.2 pode ser resumido conforme a Figura 3.10 e os passos à seguir:

- Identificação do centroide do círculo mais claro;

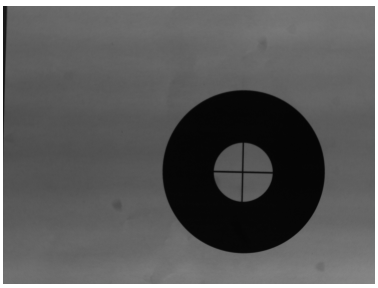


Figura 3.9: Exemplo de imagem a ser capturada.

Leitura e binarização da imagem, realizadas a partir das funções 5 e 7 da Seção 3.4;

Em seguida, utiliza-se um filtro morfológico para identificar o centroide e o raio do círculo mais claro na imagem, com a função 8. Para melhorar os resultados dessa função, adiciona-se os parâmetros $ObjetctPolarity$, definido nesse caso como $bright$ e $Sensitivity$, definida como 0.92.

- Determinação da distância para deslocamento;

  Realiza-se uma subtração entre as coordenadas do centro óptico Cx, Cy e as coordenadas do centroide encontrado $centerCircle(1,1), centerCircle(1,2)$.

- Determinação de 3 coordenadas para aplicação do IBVS;

  São escolhidos 3 pontos na imagem binarizada: o centroide do alvo e outros dois pontos localizados no contorno da circunferência (a partir das coordenadas do centroide, soma-se o valor do raio na coordenada x para obtenção de um dos pontos e na coordenada y para obtenção do outro ponto). Em seguida, calcula-se as novas coordenadas desses pontos após o deslocamento.

- Translação da imagem para alinhamento do centroide com o centro óptico da câmera por meio da função 14.

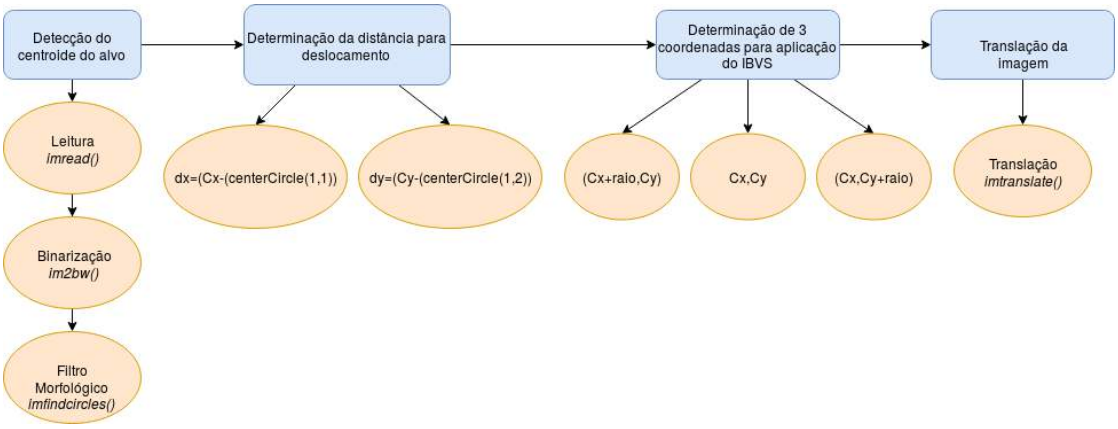


Figura 3.10: Etapas para alinhamento entre o centro do alvo e o centro da imagem.

3.8 Controle Servo Visual Baseado em Imagem

Utilizando o conjunto de coordenadas estabelecido na Seção 3.7 e aplicando as equações deduzidas na Seção 2.6, implementou-se o algoritmo IBVS a fim de encontrar as velocidades da câmera desejadas e consequentemente, o deslocamento desejado para que o efetuador terminal alinhe o centro óptico da câmera com o centro do alvo.

(a) Alvo sem o *laser*. (b) Alvo com o *laser*.

Figura 3.11: Exemplo das imagens a serem capturadas.

O código implementado na Seção I.2.3 do Apêndice pode ser resumido conforme a Figura 3.12 e os passos à seguir:

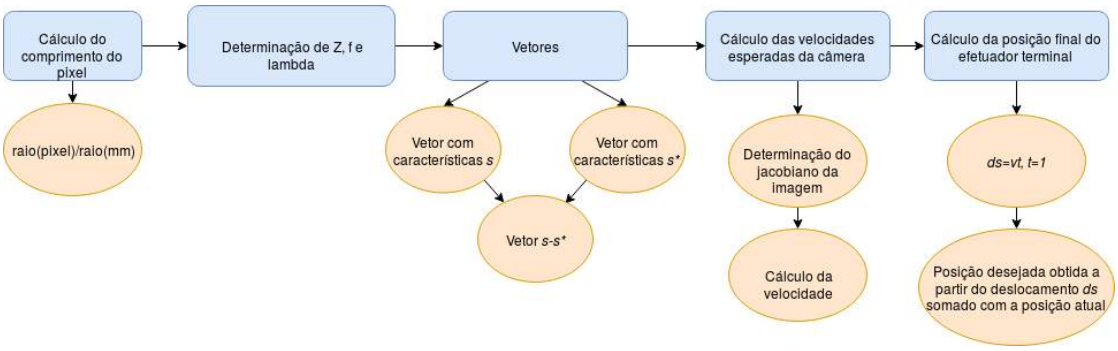


Figura 3.12: Etapas para o algoritmo IBVS.

- Determinação dos parâmetros

  Cálculo do comprimento do pixel, realizado a partir da divisão do raio do círculo mais claro do alvo em coordenadas de pixel pelo mesmo raio em unidades métricas;

  Definição dos parâmetros Z e f. Z foi estabelecido arbitrariamente e f é a distância focal da lente utilizada.

- Definição das coordenadas desejadas e das coordenadas atuais em vetores

  Criação de dois vetores, um contendo as coordenadas da imagem antes do deslocamento e o outro contendo as coordenadas da imagem após o deslocamento, ambas definidas na seção 3.7;

  Criação de um terceiro vetor contendo a diferença entre o vetor das coordenadas após o deslocamento e o vetor das coordenadas antes do deslocamento.

- Cálculo da velocidade da câmera a partir da equação 2.40.

  Determinação do jacobiano da imagem, conforme apresentado na equação 2.36;

  Utiliza-se o jacobiano da imagem e o vetor contendo a diferença entre as coordenadas antes e após o deslocamento.

- Cálculo da posição final do efetuador terminal

  Definição o deslocamento a partir da velocidade calculada (considera-se o tempo igual a 1s);

  A posição é definida pela soma do deslocamento com a posição atual do efetuador terminal do robô.

3.9 Determinação das Retas que Definem o *Laser* e as Linhas do Alvo

O procedimento consistiu em capturar duas imagens do alvo, uma com a projeção do *laser* e a outra sem. Primeiro aplicou-se um código para leitura e binarização das imagens (*Prepara.m*); Em seguida, um código para obter-se todas as linhas que podem definir o laser e o alvo (*Transf_Hough.m*); E por fim, dois códigos que calculam as linhas que melhor se ajustam ao *laser* e ao alvo (*CalcNormal.m* e *CalcIncl.m*). Ao final, obtém-se a inclinação entre as duas retas e a menor distância entre a linha do *laser* e o centro do alvo.

3.9.1 Preparação das Imagens

O código *Prepara.m*, apresentado na Seção I.2.4, pode ser resumido conforme a Figura 3.13 e os passos à seguir:

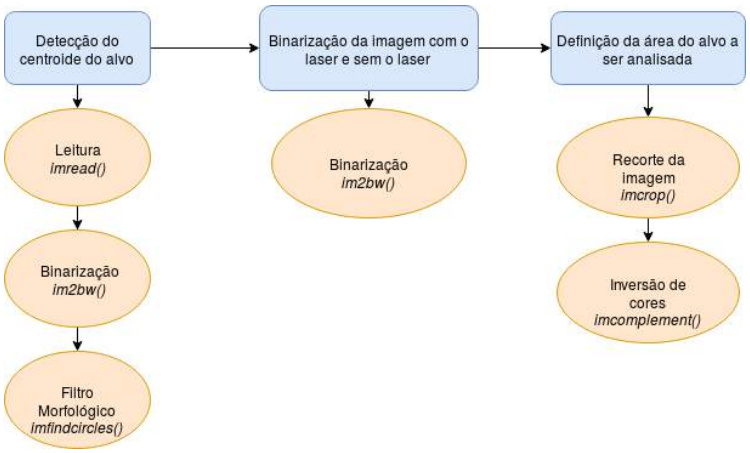


Figura 3.13: Etapas para o algoritmo de preparação da imagem.

- Identificação do centro do alvo;

Leitura e binarização das imagens, com e sem o laser, realizadas a partir das funções 5 e 7;

Em seguida, utiliza-se um filtro morfológico para identificar o centroide e o raio do círculo mais claro na imagem sem o *laser*, com a função 8. Para melhorar os resultados dessa função, adiciona-se os parâmetros $ObjetctPolarity$, definido nesse caso como *bright* e a $Sensitivity$, definida como 0.92.

- Definição da área a ser analisada na imagem sem o *laser*

  Define-se um intervalo dentro do círculo do alvo para que o recorte seja feito utilizando a função 15;

  Inverte-se as cores da imagem recortada utilizando a função 16.

3.9.2 Transformada de Hough

O código $Transf\_Hough.m$ pode ser explicado com a Figura 3.14 e as etapas à seguir:

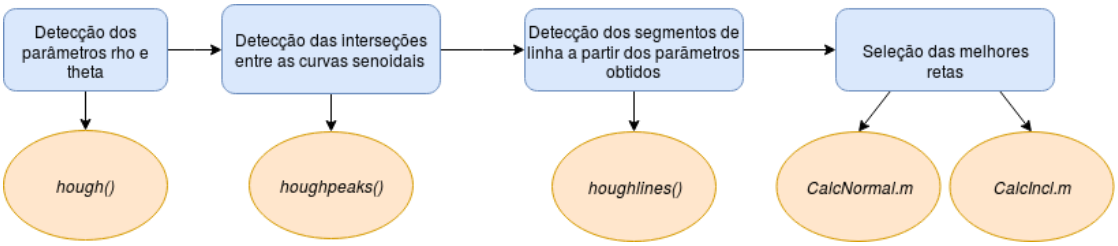


Figura 3.14: Etapas para a Transformada de Hough nas imagens capturadas.

- Aplicação da Transformada de Hough

  Utilizando as funções 17, 18 e 19, obtém-se o conjunto de linhas que passam pelas bordas existentes nos alvos com e sem o *laser*.

- Seleção das melhores retas e da inclinação entre elas.

  O código *CalcNormal.m* seleciona a menor normal existente entre as linhas formadas no laser e o centro do alvo;

  O Código *CalcIncl.m* seleciona a reta com inclinação mais próxima a do *laser*, dentre as formadas no alvo.

3.9.3 Seleção das linhas obtidas pela Transformada de Hough

O código $CalcNormal.m$ pode ser explicado com a Figura 3.15(a) e as etapas à seguir:

- Cálculo das retas normais entre o centro do alvo e as retas do laser;

  Utiliza-se a função 20 para calcular os coeficientes linear e angular das retas obtidas pela função 19.

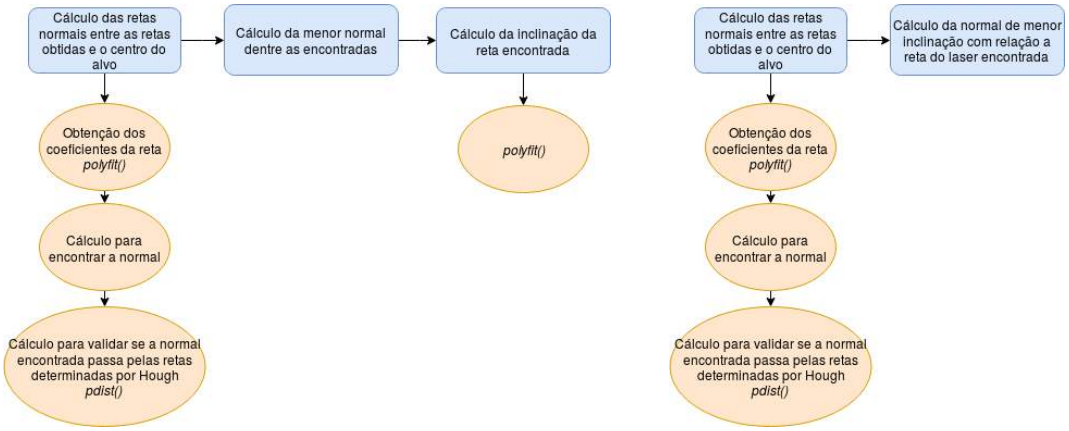
A partir da trigonometria que define a distância entre um ponto e uma reta, encontra-se os pontos de intersecção entre as retas do *laser* e as retas mais próximas ao centro do alvo.

Valida-se os resultados obtidos confirmando se o ponto de intersecção está contido entre os dois pontos que definem as retas do *laser*. Para o cálculo de distância euclidiana entre os pontos, utiliza-se a função 21;

- Cálculo da menor normal dentre as encontradas

  Compara-se as distâncias das retas normais obtidas e seleciona-se a menor, juntamente com sua reta correspondente;

  Utilizando novamente a função 20, calcula-se a inclinação da reta em radianos e em graus.



(a) (b)

Figura 3.15: Etapas para seleção das melhores retas que correspondam ao *laser* e ao alvo.

O código $CalcIncl.m$ pode ser descrito na Figura 3.15(b) nos seguintes passos:

- Cálculo das retas normais entre o centro do alvo e as retas do alvo;

  Utiliza-se a função 20 para calcular os coeficientes linear e angular das retas obtidas pela função 19;

  A partir da trigonometria que define a distância entre um ponto e uma reta, encontra-se os pontos de intersecção entre as retas do alvo e as retas perpendiculares a elas que interceptam o centro do alvo;

  Valida-se os resultados obtidos confirmando se o ponto de intersecção está contido entre os dois pontos que definem as retas do alvo. Para o cálculo de distância euclidiana entre os pontos, utiliza-se a função 21;

  Calcula-se também a inclinação das novas retas obtidas e a diferença de inclinação entre elas e a linha do *laser*.

- Cálculo da menor inclinação dentre as encontradas;

Compara-se os as inclinações das retas obtidas e seleciona-se a menor, juntametne com sua reta correspondente.

- Conversão dos resultados obtidos para a imagem original.

  Realiza-se um cálculo para que as coordenadas das retas encontradas possam ser projetadas adequadamente na imagem original.

3.9.4 Triangulação

O código apresentado na Seção I.2.5 representa uma simulação da movimentação do sistema de medição no eixo Z_c do robô quando paralelo ao eixo Z da câmera.

O procedimento para essa simulação consistiu na obtenção de imagens do alvo com e sem o *laser*, em que centro óptico estivesse alinhado com o centro do alvo. Em seguida, rodou-se o código para calcular o deslocamento Z_c em valor absoluto e percentual. A partir desse valor, alterou-se a escala da imagem para observar se o *laser* estava alinhado ou não com o centro do alvo da imagem original.

O código pode ser descrito conforme a Figura 3.16 e os passos à seguir:

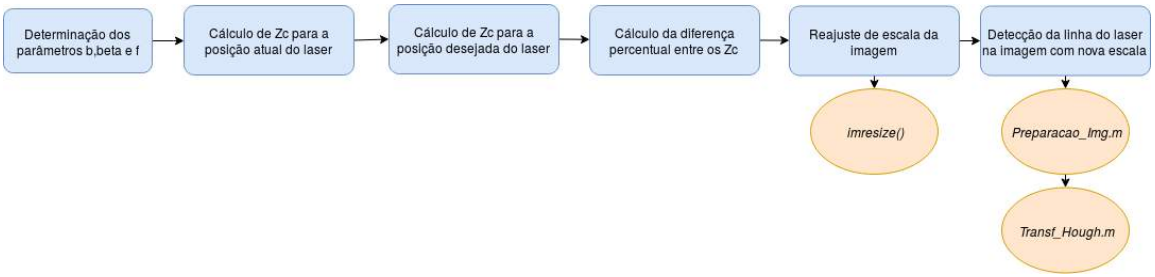


Figura 3.16: Etapas para o algoritmo de triangulação.

- Determinação dos parâmetros;

  $dist\_b$ corresponde a distância em milímeros entre a câmera e o *laser*;

  ang_beta é o ângulo de inclinação do *laser*;

  $dist\_f$ é a distância focal da lente da câmera.

- Cálculo da distância Z_c no ponto atual e desejado, a partir da equação 2.27.

  Determinação da diferença percentual entre os valores. Para situações em que o sistema de medição deve se deslocar para cima ($Zc\_ > Zc$), deve-se diminuir a escala da imagem, pois a câmera vai se afastar dela. Para situações em que o sistema de medição deve se deslocar para baixo, ($Zc\_ < Zc$), deve-se aumentar a escala da imagem, pois a câmera estará se aproximando.

- Ajuste da escala da imagem

  Ajustou-se a escala das imagens com e sem o *laser* a partir da função 22.

- Para detectar as novas coordenadas do *laser* na imagem e provar que houve um deslocamento para o centro do alvo, roda-se novamente os códigos *Preparacao_Img.m* e *Transf_Hough.m*.

3.10 Software para Simulação do Robô

Para simular as movimentações no manipulador robótico e calcular os ângulos de junta a partir das coordendas da posição calculadas para o efetuador terminal, utilizou-se o *software* RobotStudio®. Ele consiste em uma plataforam para programação *offline* de manipulares robóticos da empresa ABB e por ser construído no *VirtualController* da própria empresa, representa uma cópia exata do *software* real que roda nos manipuladores, oferecendo simulações extremamente realistas [9].

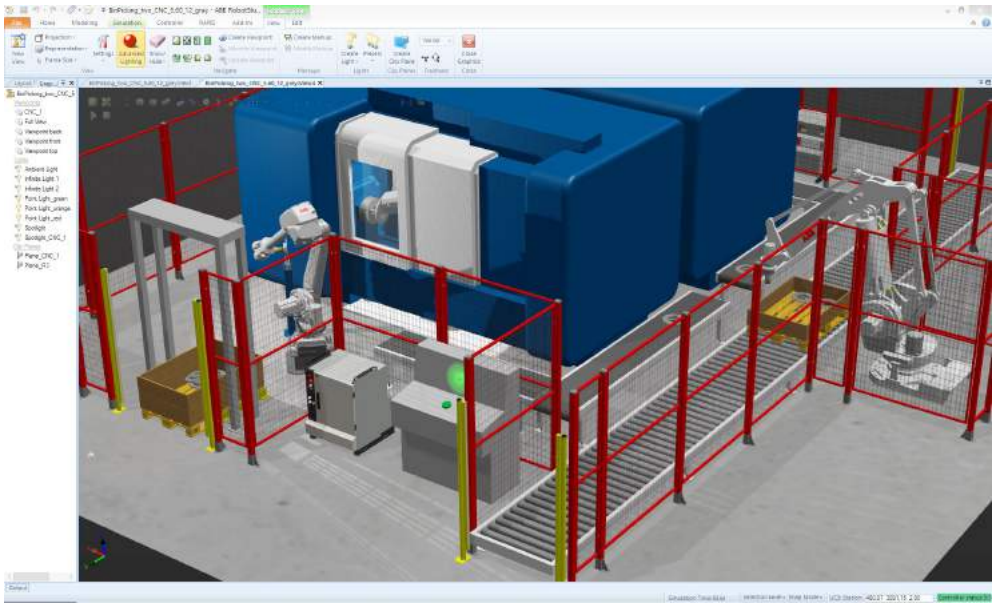


Figura 3.17: Ambiente de trabalho oferecido pela plataforma RobotStudio® [9].

A comunicação TCP/IP com o manipulador robótico foi realizada a partir de um código no MATLAB® disponível no computador do laboratório. Os ângulos de junta obtidos na plataforma RobotStudio® foram enviados para o manipulador por meio desse código a fim de obter a posição desejada.

Capítulo 4

Resultados

4.1 Introdução

Este capítulo tem o objetivo de apresentar os resultados obtidos durante todo o projeto. Serão aprsentados os dados encontrados após a calibração da câmera e do centro óptico da câmera, a partir do padrão de pontos apresentado na Figura 3.4; os resultados referentes a detecção do alvo na imagem e ao deslocamento estimado para alinhamento entre o seu centro e o centro óptico da câmera; a detecção das linhas correspondentes ao *laser* e a cruz central do alvo na imagem; o alinhamento entre as linhas, resultante do ângulo de inclinação calculado entre elas e por fim, a simulação referente ao modelo de triangulação escolhido para o projeto.

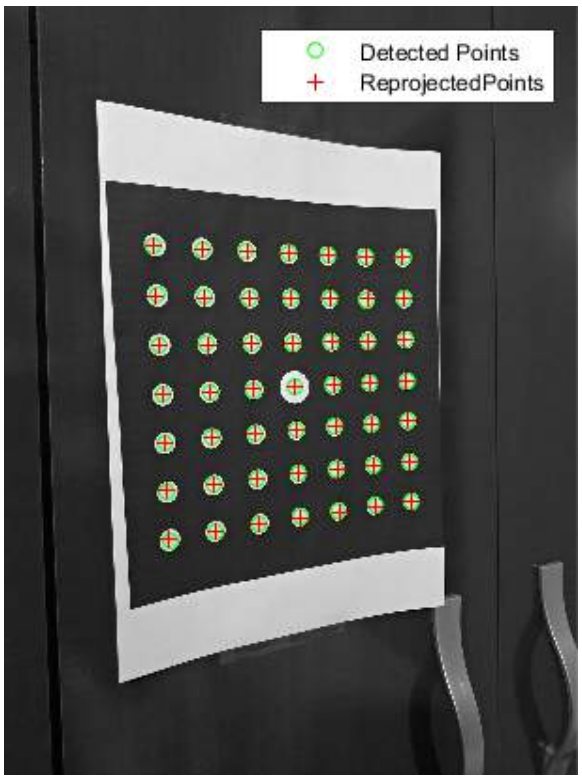
4.2 Reprojeção dos pontos

Inicialmente pensou-se em separar a calibração da câmera em duas etapas. A primeira consistiria na captura das imagens a partir de uma câmera convencional de celular, para validar o código implementado a partir dos resultados obtidos para os parâmetros intrínsecos e extrínsecos e a segunda etapa consistiria em capturar novas imagens, porém com a câmera do projeto. Entretanto, como explicado na Seção 3.6.2, seguiu-se com outra metodologia para obtenção do centro óptico da câmera e os resultados a seguir consistem na calibração de uma câmera, mas não a utilizada no projeto.

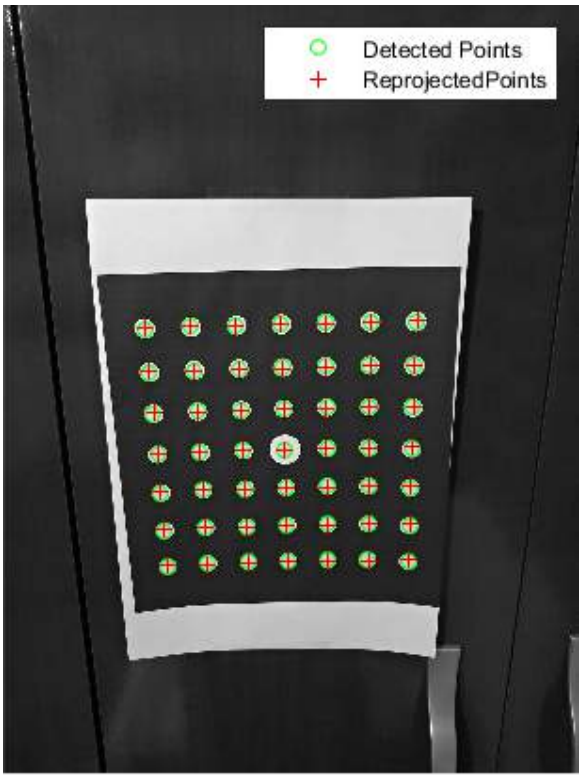
Utilizou-se 5 imagens para realização da calibração, dispondo o padrão de pontos em diferentes orientações. A reprojeção dos pontos após o processo de calibração pode ser observado no conjunto de Figuras 4.0.


Detected Points
ReprojectedPoints

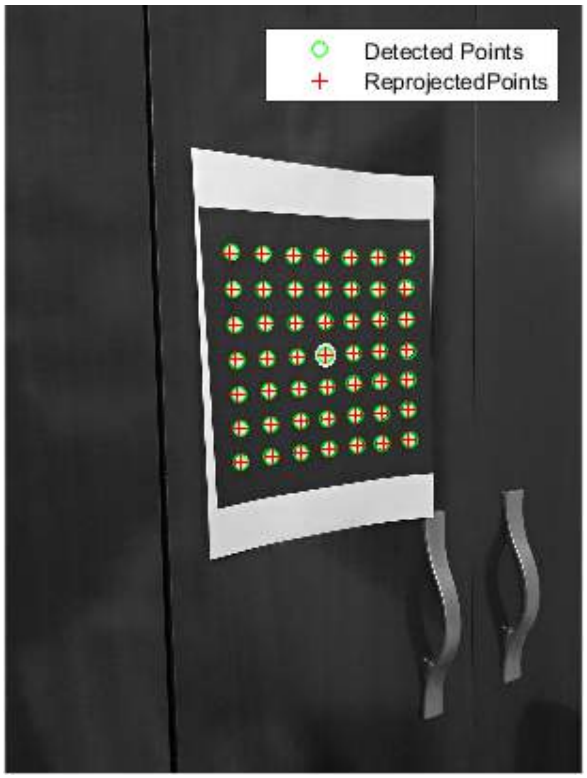
(a)


Detected Points
ReprojectedPoints

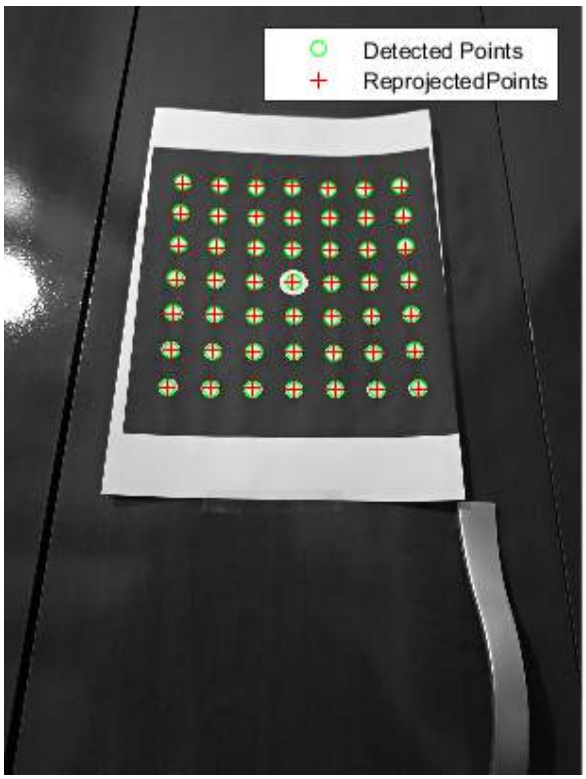
(b)


Detected Points
ReprojectedPoints

(c)


Detected Points
ReprojectedPoints

(d)



(e)

Figura 4.0: Pontos reprojetados após a calibração.

A Figura 4.1 apresenta os erros de reprojeção dos pontos nas imagens utilizadas para calibração, o erro médio correspondeu a 1.26 pixels, enquanto o erro máximo, a 1.56 pixels e o erro mínimo a 1.03 pixels. Nota-se que para imagens mais inclinadas com relação a câmera, obteve-se erros maiores de reprojeção.

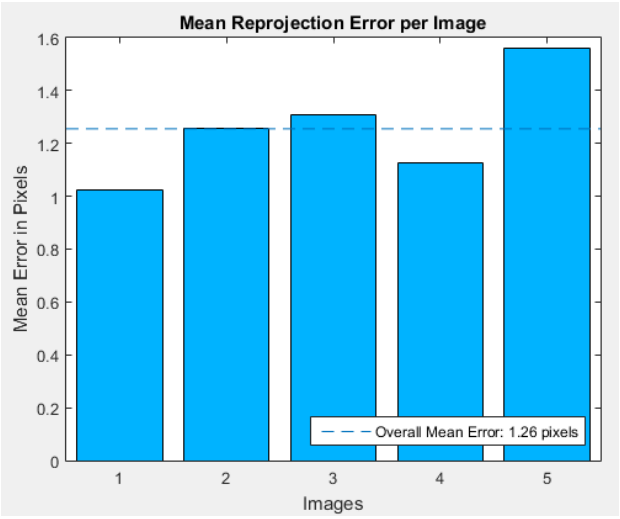


Figura 4.1: Erro médio de reprojeção por imagem.

4.3 Parâmetros intrínsecos e extrínsecos obtidos

Os parâmetros intrínsecos obtidos, conforme a notação apresentada nas equações 3.1 e 3.2 foram:

$$K^T = \begin{bmatrix} fs_x & 0 & 0 \\ s_\theta & fs_y & 0 \\ u_0 & v_0 & 1 \end{bmatrix} = \begin{bmatrix} 904.5464 & 0 & 0 \\ 0 & 904.5398 & 0 \\ 453.6982 & 614.1639 & 1 \end{bmatrix} \tag{4.1}$$

Nota-se que a distância focal (fs_x, fs_y), em pixels, é $(904.5464, 904.5398)$ e o centro óptico (u_0, v_0), em pixels, $(453.6982, 614.1639)$. s_θ é o coeficiente de inclinção, que nesse caso vale 0, porque os eixos da imagem são perpendiculares entre si.

Os parâmetros extrínsecos, isto é, as matrizes de rotação e translação transpostas, obtidos para cada imagem foram:

Figura	R^T	t^T
4.0 (a)	$\begin{bmatrix} 0.9994 & -0.0342 & -0.0043 \\ 0.0343 & 0.9984 & 0.0441 \\ 0.0028 & -0.0442 & 0.9990 \end{bmatrix}$	$\begin{bmatrix} -82.1180 & -86.1606 & 322.7253 \end{bmatrix}$
4.0 (b)	$\begin{bmatrix} 0.9195 & -0.0535 & 0.3895 \\ 0.0434 & 0.9985 & 0.0347 \\ -0.3908 & -0.0150 & 0.9204 \end{bmatrix}$	$\begin{bmatrix} -70.5505 & -77.3703 & 310.9691 \end{bmatrix}$
4.0 (c)	$\begin{bmatrix} 0.9996 & -0.0250 & -0.00094 \\ 0.0266 & 0.9570 & 0.2890 \\ 0.0018 & -0.2891 & 0.9573 \end{bmatrix}$	$\begin{bmatrix} -83.4749 & -35.3856 & 337.2996 \end{bmatrix}$
4.0 (d)	$\begin{bmatrix} 0.9051 & -0.0727 & 0.4190 \\ 0.0426 & 0.9958 & 0.0808 \\ 0.4231 & -0.0553 & 0.9044 \end{bmatrix}$	$\begin{bmatrix} -44.5659 & -101.7815 & 430.6230 \end{bmatrix}$
4.0 (e)	$\begin{bmatrix} 0.9995 & 0.0072 & 0.0300 \\ 0.0046 & 0.9272 & -0.3746 \\ -0.0305 & 0.3746 & 0.9267 \end{bmatrix}$	$\begin{bmatrix} -76.2366 & -148.5227 & 422.4864 \end{bmatrix}$

Cada matriz R de dimensão 3x3 representa a rotação 3D do plano de imagem da câmera relativo ao padrão de calibração utilizado. Os vetores de translação t contêm o padrão de calibração

que estima os parâmetros de calibração, descrevendo a translação da câmera relativa ao padrão correspondente, expresso em coordenadas globais.

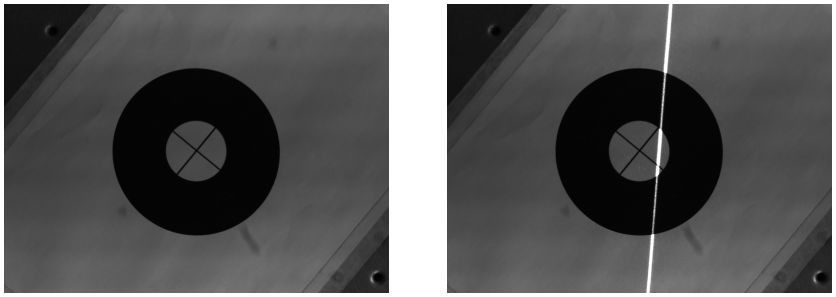
4.4 Sistema de Medição Baseado em Triangulação

A partir da modelagem em 3D realizada para o sensor e os códigos desenvolvidos, serão apresentados os resultados referentes ao sistema de medição idealizado para o projeto.

Para validação, escolheu-se uma posição (X, Y, Z) em coordenadas espaciais para o efetuador terminal e a configuração $EulerXYZ$ para descrever as rotações rx, ry, rz ao redor dos eixos XYZ de referência. A posição determinada foi $(X, Y, Z) = (440.06, 0, 713.22)$, com ângulos de *Euler* iguais a $(rx, ry, rz) = (0, 88.3, 0)$. Os resultados de cada processamento de imagem e movimentação nos ângulos de junta do robô serão apresentados e comentados a seguir.

(a) (b)

Figura 4.2: Posição 1 do alvo na mesa.



(a) (b)

Figura 4.3: Posição 2 do alvo na mesa.

4.4.1 Sensor e Alvo construídos

A caixa modelada na Seção 3.3 foi construída em acrílico e confeccionada pelo técnico do Laborátorio GRACO, sem seguida foi conectada à flange do robô, juntamente com a câmera e o *laser*. O alvo consiste em dois círculos concêntricos, um mais externo de cor preta e o outro mais interno de cor branca. No círculo mais interno passam duas linhas que se interceptam em seu

centroide e servem de orientação para o *laser* durante o processo de alinhamento. A existência de dois círculos de cores diferentes é necessária para facilitar a detecção durante o processamento de imagens. O sensor final e seu alvo de referência podem ser observados na Figura 4.4:

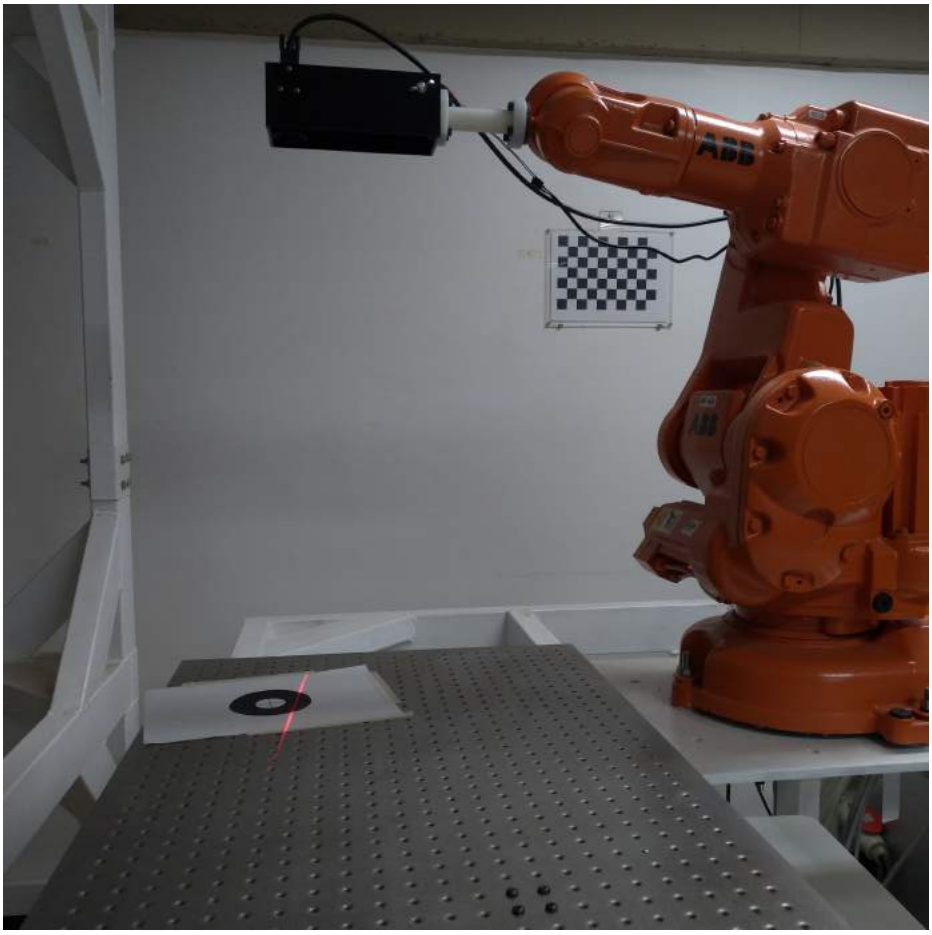


Figura 4.4: Sensor e alvo construídos para o projeto.

4.4.2 Calibração do Centro Óptico

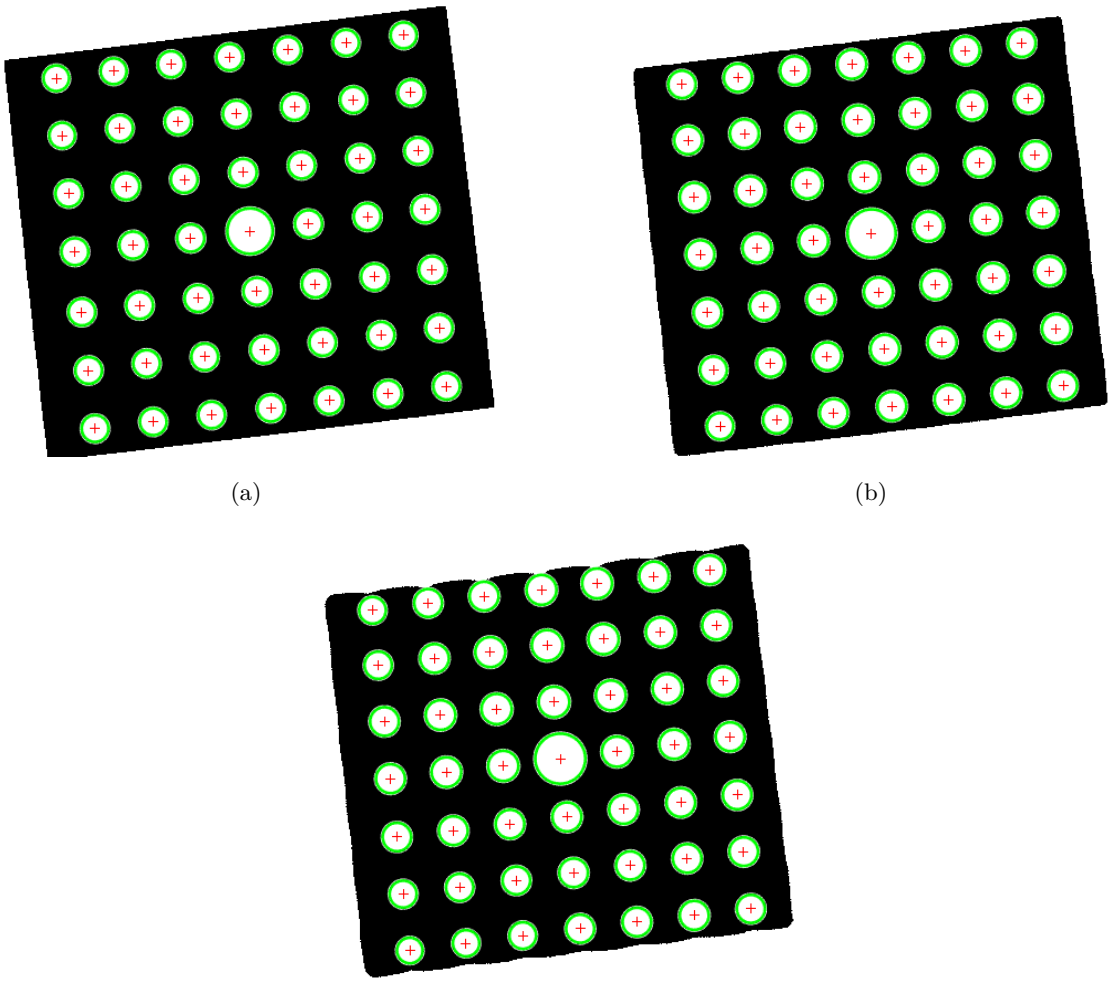
O resultado da detecção dos círculos e seus centroides, realizada pela função 8 no código $Centro\_Opt.m$, é apresentada no conjunto de figuras 4.5 e as soluções encontradas para os sistemas lineares construídos são disponibilizadas na Tabela 4.1.

Nota-se que o resultado obtido entre entre as imagens 3.7(a) e 3.7(b), foi aumentado em x e diminuindo em y, quando comparado com os resultados seguintes entre 3.7(a) e 3.7(c) e entre 3.7(b) e 3.7(c), permitindo concluir que, durante o procedimento de captura das imagens e alteração da distância focal, a câmera se deslocou um pouco para a direita e para cima no sistema de coordenadas da imagem.

A fim de reduzir o erro entre os valores obtidos, calculou-se uma média de todos os valores, obtendo-se, por fim, um centro óptico na imagem com coordenadas: $(1640.4, 1276.8)$.

Tabela 4.1: Coorndenadas encontradas para o centro óptico na imagem

Sistema	Pontos analisados	Coordenada C_x	Coordenada C_y
1	49 pontos das Figuras 3.7(a) e 3.7(b)	1506.9	1326.3
2	49 pontos das Figuras 3.7(a) e 3.7(c)	1696	1185.9
3	49 pontos das Figuras 3.7(b) e 3.7(c)	1715.3	1304.2
4	Sistemas 1 e 2	1554.2	$1253,4$
5	Sistemas 1 e 3	1650.4	1304.1
6	Sistemas 2 e 3	1708.2	1280
7	Sistemas 1, 2 e 3	1615.9	1283.9



(a) (b)

(c)

Figura 4.5: Detecção dos centroides no padrão em diferentes distâncias focais.

4.4.3 Alinhamento do Centro Óptico da Câmera com o Centro do Alvo

A análise nos conjuntos de figura 4.2 e 4.3 permitiu identificar deslocamentos x e y necessários para alinhamento entre o centro do alvo e o centro óptico da câmera na imagem. O resultado após a translação da imagem pode ser observado nas Figuras 4.6 e 4.7. Os pontos na imagem foram escolhidos arbitrariamente, com fins de visualização do processo.

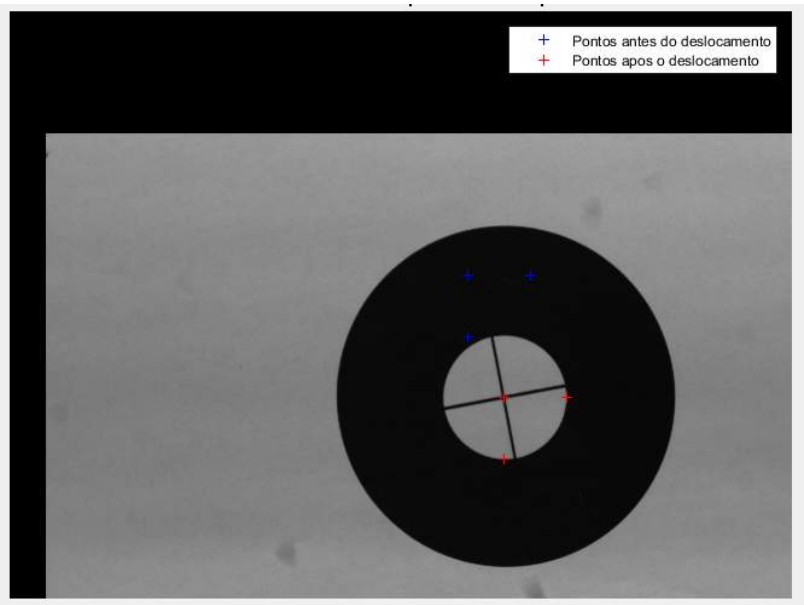


Figura 4.6: Imagem 4.2(a) transladada a fim de garantir o alinhamento entre o centro do alvo e o centro óptico da câmera.

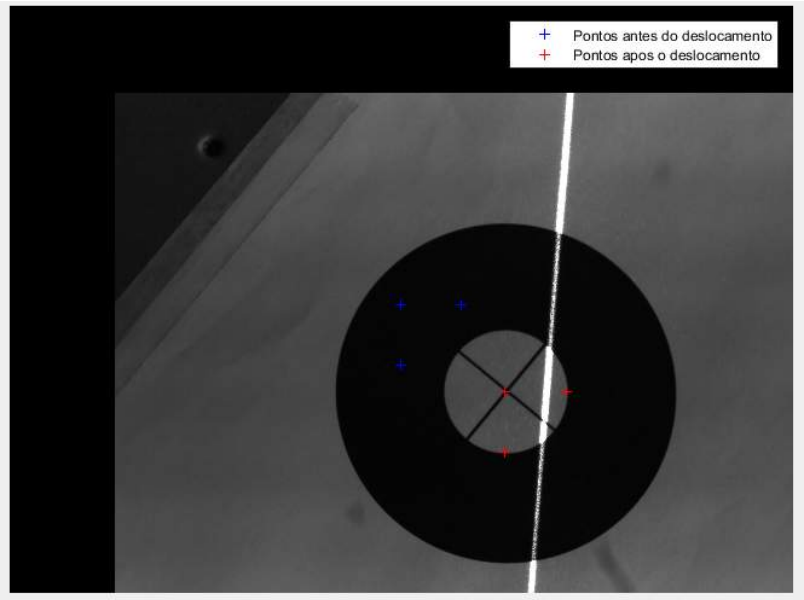


Figura 4.7: Imagem 4.3(b) transladada a fim de garantir o alinhamento entre o centro do alvo e o centro óptico da câmera.

As coordenadas em pixel encontradas para o centro do alvo na Figura 4.2(a) foram $(1520.2, 874.1)$, resultando em um deslocamento de 120.19 pixels em x e 402.69 pixels em y. Na Figura 4.3(a), as coordenadas do centro do alvo foram $(1293.9, 989.4)$, resultando em um deslocamento de $346,5$ pixels em x e 287.4 pixels em y.

4.4.4 Controle Servo Visual Baseado em Imagem

A partir das distâncias de deslocamento calculadas na Seção 4.4.3, calculou-se a nova posição do efetuador terminal para que a posição requerida fosse atingida, utilizando o algoritmo *IBVS*.

Para o primeiro caso, a nova posição estabelecida para o efetuador terminal, em coordenadas espaciais, foi: $(462.5303, -36.0945, 761.2498)$, resultando nos ângulos de junta $\theta_1 = -5.19^o$, $\theta_2 = -7.56^o$, $\theta_3 = -0.16^o$, $\theta_4 = 38.52^o$, $\theta_5 = 8.34^o$ e $\theta_6 = 322^o$. O resultado obtido após o deslocamento foi apresentado na Figura 4.8.

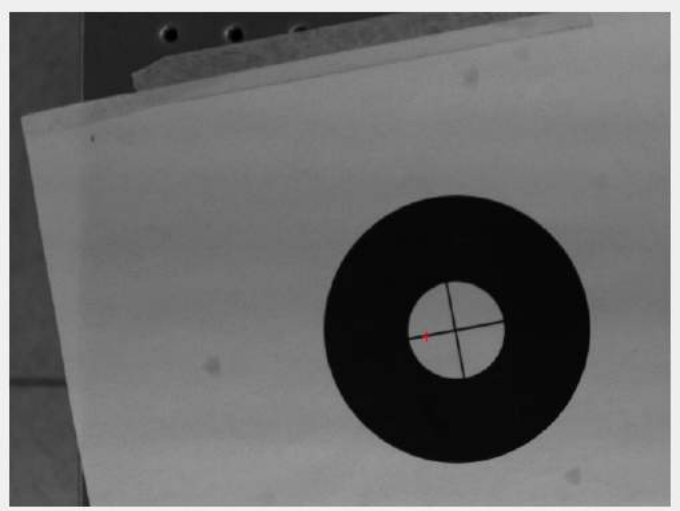


Figura 4.8: Imagem 4.2(b) após o deslocamento do sensor para garantir o alinhamento entre o centro do alvo e o centro óptico da câmera.

Nota-se que o deslocamento não foi perfeito, pois ainda há um deslocamento do centro do alvo com relação ao eixo da câmera. Isso pode ser explicado devido ao fato da câmera não estar completamente alinhada com o efetuador terminal. Para tentar amenizar esse erro de alinhamento, descobriu-se empiricamente o *offset* de x e y nessa configuração (-10 para x e $+2$ para y). O novo resultado obtido encontra-se na Figura 4.9.

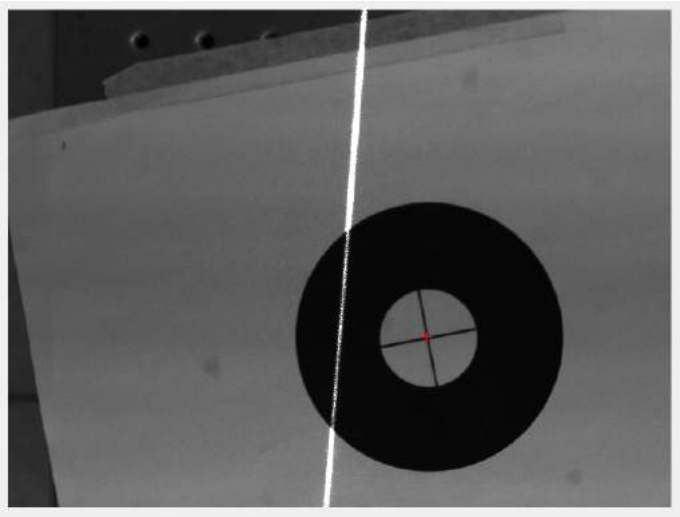


Figura 4.9: Imagem 4.2(b) após ajuste de *offset* para garantir o alinhamento entre o centro do alvo e o centro óptico da câmera.

Para o segundo caso, a nova posição estabelecida para o efetuador terminal, em coordenadas espaciais, foi: $(438.2223, -9.0829, 713.0623)$. Essa variação foi muito inferior ao que se esperava para atingir uma diferença do posicionamento do eixo da câmera no centro do alvo da imagem. Entende-se que o problema ocorreu justamente na implementação do algoritmo IBVS, que depende de alguns parâmetros estabelecidos arbitrariamente, como distância Z e o fator de multiplicação λ, alterando bastante o resultado final quando variados.

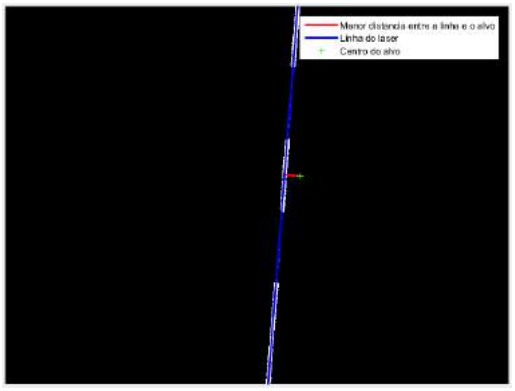
4.4.5 Determinação das retas que Definem o *Laser e as Linhas do Alvo*

Nessa etapa, binarizou-se as imagens referentes ao alvo com e sem *laser* a fim de realizar um procedimento de identificação das linhas que os definiam.

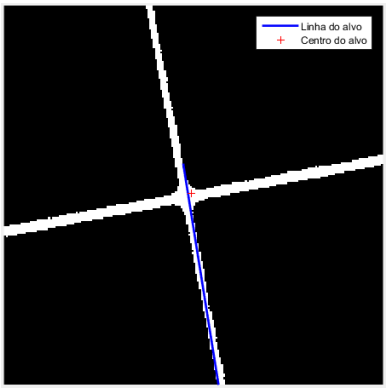
4.4.5.1 Transformada de Hough

Para identificação das linhas, utilizou-se o princípio da Transformada de Hough. Na identificação da linha do *laser*, decidiu-se selecionar sempre a linha que estivesse mais próxima do centro do alvo. Na identificação da linha do alvo, decidiu-se selecionar a linha que tivesse inclinação mais próxima com a do *laser*, a fim de garantir um menor deslocamento do manipulador robótico na realização do procedimento.

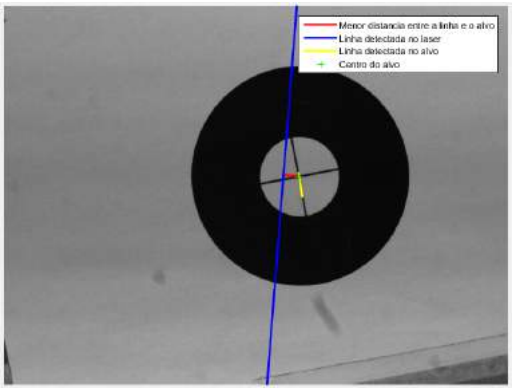
No primeiro caso, representado pela Figura 4.2(b), encontrou-se uma inclinação de -85.52^o para a linha do *laser* e 80.9285^o para a linha do centro do alvo. Dessa forma, a inclinação a ser imposta no ângulo de *Euler* Rz seria 4.59^o. Entretanto, para que se houvesse uma mudança visual na inclinação, seria necessário adicionar um *offset* de aproximadamente 10^o no valor atribuido a Rz.



(a) Detecção da linha do *laser* mais próxima ao centro do alvo.

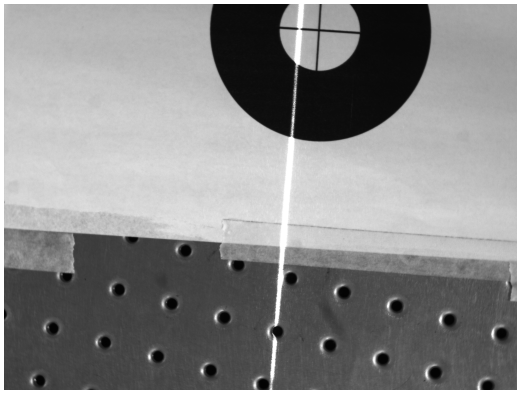


(b) Detecção da linha do alvo com inclinação mais próxima com relação a linha do *laser* encontrada.

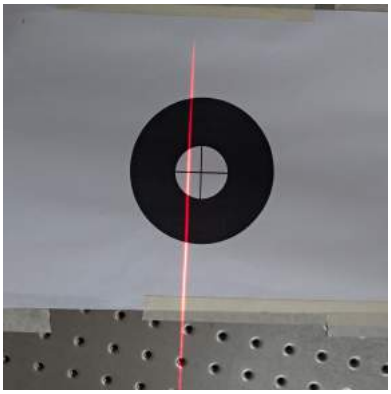


(c) Linhas detectadas plotadas na imagem original.

Figura 4.10: Detecção das linhas da Figura 4.2(b) utilizando Transformada de Hough.



(a) Inclinação capturada pela câmera do sensor.

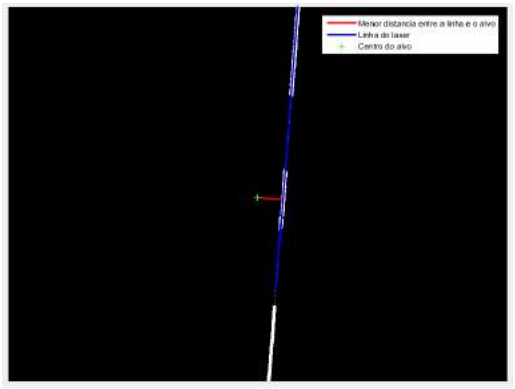


(b) Inclinação capturada pela câmera do celular.

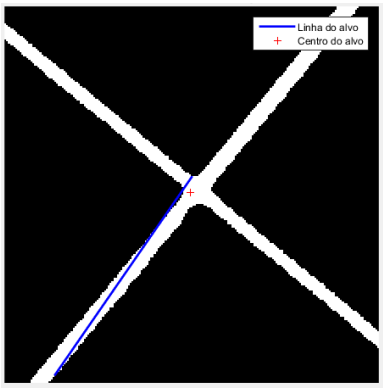
Figura 4.11: Inclinação, referente à Figura 4.2(b), capturada pela câmera do sensor e por uma câmera de celular após o deslocamento do manipulador.

No segundo caso, representado pela Figura 4.3(b), encontrou-se uma inclinação de -85.5^o para

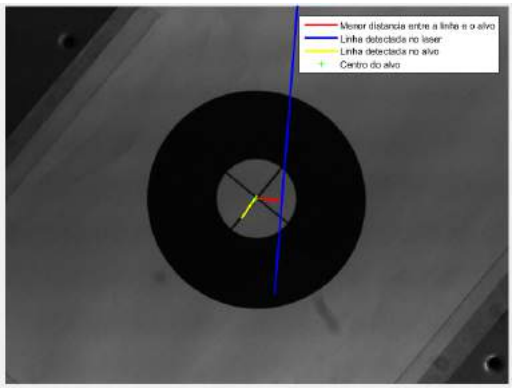
a linha do *laser* e -55.3^o para a linha do centro do alvo. Dessa forma, a inclinação a ser imposta no ângulo de *Euler* Rz seria -30.2^o. Nesse caso adicionou-se um *offset* de aproximadamente -10^o no valor atribuido a Rz.



(a) Detecção da linha do *laser* mais próxima ao centro do alvo.



(b) Detecção da linha do alvo com inclinação mais próxima com relação a linha do *laser* encontrada.



(c) Linhas detectadas plotadas na imagem original.

Figura 4.12: Detecção das linhas da Figura 4.3(b) utilizando Transformada de Hough.

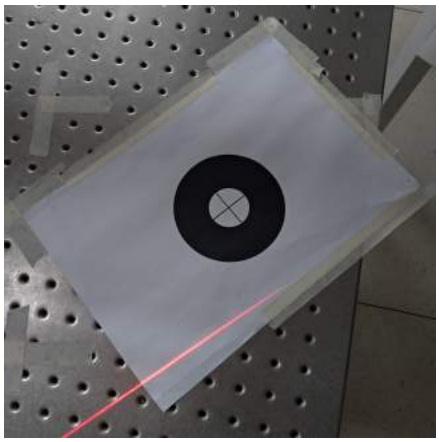


Figura 4.13: Inclinação, referente à Figura 4.3(b), capturada por uma câmera de celular após o deslocamento do manipulador.

Nota-se que em ambos os casos houve o alinhamento referente à linha do alvo utilizada como base. No caso 1 ainda foi possível observar a imagem capturada pela câmera do sensor, mas no caso 2, o alvo ficou muito afastado do eixo da câmera, sendo necessário capturar a imagem a partir de outro dispositivo.

4.4.6 Triangulação

Essa etapa não foi realizada no robô, apenas no MATLAB®, onde simulou-se o deslocamento que deveria ocorrer no eixo Z da câmera, considerando o alinhamento entre o centro óptico da câmera e o centro do alvo. Para essa simulação, calculou-se o Z_c correspondente a coordenada x do ponto de interseção entre a reta do *laser* e a reta normal que passa pelo centro do alvo e o Z_c correspondente a coordenada x do centro do alvo. Os valores de Z_c encontrados foram usados para alterar a escala da imagem original e assim, analisar o deslocamento do *laser* na imagem.

Para o caso 1, utilizou-se a Figura 4.9, encontrando-se o ponto de cruzamento da reta normal ao *laser* e ao centro do alvo da imagem como $(1302.3, 1215.2)$. A partir da coordenada em x desse ponto, obteve-se $Zc = 1.6080$mm. Já para a coordenada em x do centroide do alvo, localizado em $(1640, 1277)$, encontrou-se $Z_c = 1.2939$mm. Nesse caso, nota-se que Z_c diminuiu e que a condição necessária para o alinhamento do *laser* seria a aproximação do sensor visual com o alvo. Para essa simulação, isso representa um aumento da escala da imagem, cujo valor calculado a partir da diferença percentual entre os Z_c encontrados foi de $scale = 1.2427$.

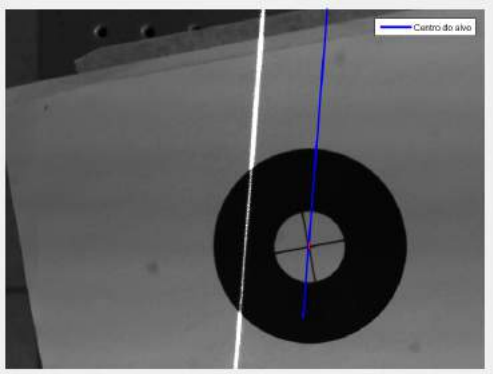


Figura 4.14: As coordenadas da linha do *laser* obtidas após a mudança de escala da imagem foram projetadas na imagem original (simulação feita na Figura 4.9).

Para o caso 2, utilizou-se a Figura 4.7 sem os pontos para realizar a simulação. Tomou-se essa decisão tendo em vista que o alinhamento real não foi possível de ser realizado na prática, mas entende-se que os resultados divergirão um pouco da realidade, já que durante o alinhamento do centro óptico, a distância entre o*laser* e o centro do alvo muda, fato não considerado na translação da imagem apenas no MATLAB®.

O ponto de cruzamento da reta normal ao *laser* e ao centro do alvo da imagem encontrado foi $(1774.5, 1287.2)$. A partir desse valor, calculou-se o valor de Z_c para a coordenada x e o valor de Z_c para a coordenada x do centro do alvo, encontrando-se, respectivamente, $Z_c = 1.2009$mm e

$Z_c = 1.2939$mm. Nessa situação, nota-se que Z_c aumentou que o alinhamento do *laser* com o alvo depende de um afastamento do sensor visual com o alvo. Para essa simulação, isso corresponde a uma redução da escala da imagem, cujo valor calculado a partir da diferença percentual dos Z_c foi $scale = 0.9281$.

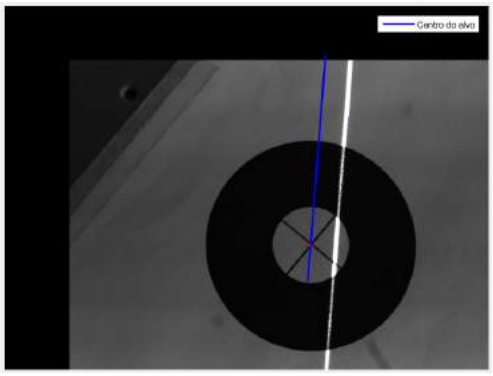


Figura 4.15: As coordenadas da linha do *laser* obtidas após a mudança de escala da imagem foram projetadas na imagem original (simulação feita na Figura 4.7).

Capítulo 5

Conclusões

O projeto consistiu em uma tentativa de replicar um sistema de medição baseado em visão ativa, realizando o processamento de imagens no *software* MATLAB®. Os resultados computacionais obtidos em imagem validaram o que a teoria propunha, porém notou-se alguns empecilhos ao repassar a tarefa para o manipulador robótico.

A calibração da câmera e do centro óptico apresentaram resultados aceitáveis ao projeto, mesmo com o deslocamento do eixo óptico percebido nos resultados. Para melhorar os resultados obtidos, pode-se aumentar a quantidade de imagens e pontos analisados em ambos os processos.

A caixa construída para o sensor e fixada na flange do robô não apresentou vibrações significativas durante a sua movimentação e atendeu as expectativas de fixação da câmera e do *laser*, permitindo sua mudança de angulação sem muita dificuldade. Algo que pode ser revisto é o alinhamento do eixo da câmera com o eixo do punho do manipulador, a fim de certificar sua influência nos resultados.

O alinhamento do centro óptico da câmera com o centro do alvo, embora tenha apresentado resultados coerentes no processamento de imagens, não pode ser validado por completo no manipulador robótico. O algoritmo IBVS implementado deve ser reavaliado, investigando a fundo as influências de algumas variáveis, escolhidas a princípio de forma arbitrária, e a melhor forma de serem definidas.

A identificação das linhas utilizando Transformada de Hough mostrou resultados aceitáveis para o projeto, mas deve-se destacar aqui a relevância de uma boa binarização das imagens e a influência do ambiente com relação a iluminação para obtenção de resultados bons e confiáveis. Pode-se notar, pelos resultados, que a espessura das bordas a serem analisadas interefere bastante nos resultados, em bordas finas, a probabilidade de serem detectadas linhas mais inclinadas com relação ao contorno da borda é bem menor.A movimentação do robô a fim de diminuir a inclinação entre o *laser* e a reta do alvo mostrou resultados positivos após a inserção do *offset* no ângulo Rz, mas deve-se estudar esse procedimento a fim de atingir resultados mais precisos.

A simulação da movimentação em Zc, pelo método da triangulação para aproximação do *laser* ao centro óptico projetado na imagem mostrou-se bastante satisfatória, mas não chegou a ser

implementada no manipulador.

Deve-se levar em conta nesse projeto que cada movimentação proposta para a câmera pode interferir no posicionamento do *laser* na imagem e que os processos devem ser feitos de forma iterativa. Dessa forma, um procedimento pode acabar desvalidando o anterior e tendo que ser refeito.. Outro aspecto a ser considerado é a importância de uma boa cinemática inversa e direta do manipulador a ser utilizado e a constante realização de testes no próprio manipulador a fim de detectar as alterações que devem ser feitas entre os resultados obtidos computacionalmente e os dados a serem inseridos no robô.

5.1 Perspectivas Futuras

O projeto apresentado pode ser considerado o passo inicial para a construção de um sistema de calibração de robôs.

O próximo passo a ser realizado deve ser fechamento do modelo, para que todos os passos necessários para o alinhamento do *laser* com o alvo possam ser realizados pelo manipulador robótico e não apenas computacionalmente. Para melhorar ainda mais o sistema, pode-se desenvolver um algoritmo para comunicar a câmera diretamente com o MATLAB® e implementar algoritmos para cálculo da cinemática inversa e direta do manipulador, a fim de tornar reduzir o tempo gasto entre os testes.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] FORSYTH, D. A.; PONCE, J. *Computer vision: a modern approach*. [S.l.]: Prentice Hall Professional Technical Reference, 2002.

[2] OPENSTAX UNIVERSITY PHYSICS. *Thin Lenses*. Disponível em: <http://bit.do/phys-libretexts-org>. Acesso em: 27 nov. 2019.

[3] HARTLEY, R.; ZISSERMAN, A. *Multiple view geometry in computer vision*. [S.l.]: Cambridge university press, 2003.

[4] FORSMAN, P. et al. *Three-dimensional localization and mapping of static environments by means of mobile perception*. [S.l.]: Helsinki University of Technology, 2001.

[5] JUSTIN LIANG. *Canny Edge Detection*. Disponível em: <http://justin-liang.com/tutorials/canny/>. Acesso em: 27 nov. 2019.

[6] ALYSSA QUEK. *Understanding Hough Transform With Python*. Disponível em: <https://alyssaq.github.io/2014/understanding-hough-transform/>. Acesso em: 27 nov. 2019.

[7] CHUBAK BIDPAA. *Straight Outta Hough: Line Detection with Hough Transform*. Disponível em: <http://partlyshaderly.com/author/chubak-bidpaa/>. Acesso em: 27 nov. 2019.

[8] ROBOTICS, A. Product manual irb 140. *ABB Robotics Products AB publication*, n. M2000, p. 7564–1, 2003.

[9] ABB. *RobotStudio®*. Disponível em: <https://new.abb.com/products/robotics/pt/robotstudio>. Acesso em: 04 dez. 2019.

[10] WANG, Z. et al. Three-dimensional shape measurement with a fast and accurate approach. *Applied optics*, Optical Society of America, v. 48, n. 6, p. 1052–1061, 2009.

[11] YIN, S. et al. A vision-based self-calibration method for robotic visual inspection systems. *Sensors*, Multidisciplinary Digital Publishing Institute, v. 13, n. 12, p. 16565–16582, 2013.

[12] MOORING, B. W.; ROTH, Z. S.; DRIELS, M. R. *Fundamentals of manipulator calibration*. [S.l.]: Wiley New York, 1991.

[13] REIS, L. V. S. V. Sistema para calibração de robô paralelo baseado em compensação estatística de erros com avaliação experimental. 2018.

[14] REIN VAN DEN BOOMGAARD. *Lecture Notes Image Processing and Computer Vision*. Disponível em: <https://staff.fnwi.uva.nl/r.vandenboomgaard/IPCV20162017/LectureNotes/index.html>. Acesso em: 27 nov. 2019.

[15] CORKE, P. *Robotics, vision and control: fundamental algorithms in MATLAB® second, completely revised*. [S.l.]: Springer, 2017.

[16] TSAI, R. A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses. *IEEE Journal on Robotics and Automation*, IEEE, v. 3, n. 4, p. 323–344, 1987.

[17] ZHANG, Z. A flexible new technique for camera calibration. *IEEE Transactions on pattern analysis and machine intelligence*, v. 22, 2000.

[18] ZHUANG, H.; ROTH, Z. S. *Camera-aided robot calibration*. [S.l.]: CRC press, 2018.

[19] CANNY, J. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, Ieee, n. 6, p. 679–698, 1986.

[20] DING, L.; GOSHTASBY, A. On the canny edge detector. *Pattern Recognition*, Elsevier, v. 34, n. 3, p. 721–725, 2001.

[21] KRISHNA, R. *COMPUTER VISION FOUNDATIONS AND APPLICATIONS*. [S.l.]: Stanford: Stanford University, 2017.

[22] DUDA, R. O.; HART, P. E. *Use of the Hough transformation to detect lines and curves in pictures*. [S.l.], 1971.

[23] LIMA[1], D. M. de S.; SILVA[1], T. N. B. da; PINHEIRO, O. R. Controle servo visual aplicado a manipuladores robóticos.

[24] CHAUMETTE, F. Potential problems of stability and convergence in image-based and position-based visual servoing. In: *The confluence of vision and control*. [S.l.]: Springer, 1998. p. 66–78.

[25] MUÑOZ, G. L. L. Análise comparativa das técnicas de controle servo-visual de manipuladores robóticos baseadas em posição e em imagem. 2011.

[26] PETER CORKE. *Image-Based Visual Servoing*. Disponível em: <https://robotacademy.net.au/lesson/image-based-visual-servoing/>. Acesso em: 27 nov. 2019.

[27] BASLER AG. *acA2500-14um*. Disponível em: <https://docs.baslerweb.com/aca2500-14um.html>. Acesso em: 27 nov. 2019.

[28] BASLER AG. *Edmund Optics Lens CFFL F1.4 f16mm 2/3- Lenses*. Disponível em: <https://www.baslerweb.com/en/products/vision-components/lenses/edmund-optics-lens-cffl-f1-4-f16mm-2-3/>. Acesso em: 27 nov. 2019.

[29] GLOBAL LASER. *LDM115 Laser Diode Module*. Disponível em: <http://www.global-lasertech.co.uk/our-products/multipurpose-range/ldm115-laser-diode-module/>. Acesso em: 27 nov. 2019.

[30] EDMUND OPTICS INC. *5.0mW 633nm, Miniature LDM Laser Diode*. Disponível em: <https://www.edmundoptics.com/p/50mw-633nm-miniature-ldm-laser-diode/10814/>. Acesso em: 27 nov. 2019.

[31] GLOBAL LASER. *LGO115 & LGO145Line & Cross Generator Optics*. Disponível em: <http://www.global-lasertech.co.uk/wp-content/uploads/2018/11/LGO_CGO_Userguide_EN.pdf>. Acesso em: 27 nov. 2019.

APÊNDICE

I. CÓDIGOS IMPLEMENTADOS

I.1 Calibração da Câmera

Aqui está contido o código implementado para calibração de câmera, utilizando um alvo de padrão de pontos, no *MATLAB*.

```
1
2  %% Universidade de Brasilia
3      %TG 1
4      %Autor: Sara Gomes Cardoso.
5
6  %% Calibracao de camera - Padrao de pontos
7
8  %% Especificacao do diretorio e do objeto com os dados das imagens
9      imgFolder = fullfile(matlabroot, 'bin', 'TG', 'TG1');
10     imgSet = imageSet(imgFolder);
11     imageFileNames = imgSet.ImageLocation;
12
13 %% Variaveis
14     n = 5; %Numero de imagens utilizadas
15     p_row = 7;
16     p_col = 7;
17     row = p_row + 1;
18     col = p_col + 1;
19     points = p_row * p_col;
20     DistanceInMM = 26.5; %Distancia em mm entre os centros dos pontos
21     boardSize = [row,col];
22     minRadius = 9;
23     maxRadius = 30;
24
25     aux = cell(1,n);
26     img = cell(1,n);
27      for u = 1:n
28          imagePoints = zeros(points, 2, n);
29          aux_imagePoints = zeros(points, 2, n);
30          imageSize = zeros(1, 2, n);
31          radiiStrong = zeros(points, n);
32      end
33
```

```
34 %% Deteccao dos pontos em coordenadas globais
35     worldPoints = generateCheckerboardPoints(boardSize,DistanceInMM);
36
37 %% Deteccao dos pontos em coordenadas da imagem
38     for i = 1:n
39         aux{i} = imread(sprintf('Img_%d.jpg', i));
40         img{i} = rgb2gray(aux{i});
41         [centers, radii, metric] = imfindcircles(img{i} , [minRadius
               maxRadius]);
42         centersStrong = centers(1:points, : );
43         aux_imagePoints(:, :, i) = centersStrong;
44     end
45     %Ordenacao do vetor de pontos obtido
46         for i=1:n
47             aux_imagePoints(1:points,:,i)=sortrows(aux_imagePoints(1:
                   points,:,i));
48             j=p_col;
49             for k=1:p_col:points
50                 imagePoints(k:j,:,i)=sortrows(aux_imagePoints(k:j,:,i)
                       ,2);
51                 j=j+p_col;
52             end
53         end
54
55 %% Estimacao dos parametros da camera obtidos
56     for i = 1:n
57         imageSize = [size(img{i}, 1),size(img{i}, 2)];
58         params = estimateCameraParameters(imagePoints,worldPoints);
59         figure;
60         imshow(img{i});
61         hold on;
62         plot(imagePoints(:, 1, i), imagePoints(:, 2, i), 'go');
63         plot(params.ReprojectedPoints(:, 1, i),params.
               ReprojectedPoints(:, 2 ,i), 'r+');
64         legend('Pontos detectados','Pontos Reprojetados');
65         hold off;
66     end
67      figure;
68      showReprojectionErrors(params);
69      figure;
70      showExtrinsics(params);
```

I.2 Código Principal

Aqui está contido o código principal do sistema de medição baseado em triangulação.

```
%%Universidade de Brasilia
    %TG2
    %Autor: Sara Gomes Cardoso.

%% Instrucoes

    %Para rodar esse codigo, deve-se verificar se a pasta contem as
        imagens dos alvos e dos padroes de calibracao, alem dos codigos
         Centro_Opt.m, Translacao_Img.m, IBVS.m, Preparacao_Img.m,
        Transf_Hough.m, CalcNormal.m, CalcIncl.m e Triangulacao.m

%Acesso a pasta com os arquivos das imagens (Mudar diretorio se
    necessario)
    imgFolder = fullfile(matlabroot, 'bin', 'TG', 'TG2_2');
    imgSet = imageSet(imgFolder);
    imageFileNames = imgSet.ImageLocation;

%% Calculo das coordenadas do centro optico da camera
    run Centro_Opt.m

%% Translacao do centro do alvo para o centro optico
    run Translacao_Img.m

%% Controle servo visual
    run IBVS.m
    figure, imshow(ImgOrg), hold on
    plot ([u1;u2;u3],[v1;v2;v3], 'b+');
    plot ([u1_;u2_;u3_],[v1_;v2_;v3_], 'r+');
    hold off
    title ('Deslocamento do centro do alvo para o centro optico da
        camera')
    legend('Pontos antes do deslocamento','Pontos apos o deslocamento'
        )

%% Preparacao da Imagem
    run Preparacao_Img.m

%% Deteccao de linhas
    run Transf_Hough.m
```

```
34      figure, imshow(ImgVec{1}), hold on
35      plot([centerCircle(1,1) Crossp1(1,1)],[centerCircle(1,2) Crossp1
            (1,2)],'LineWidth',2,'Color', 'red');
36      plot (Edgesp1(:,1),Edgesp1(:,2),'LineWidth',2,'Color','blue');
37      plot (Edgesp2(:,1),Edgesp2(:,2),'LineWidth',2,'Color','yellow');
38      plot(centerCircle(1,1),centerCircle(1,2),'g+');
39      hold off
40      title ('Menor distancia entre a linha do laser e o centro do alvo'
            )
41      legend('Menor distancia entre a linha e o alvo','Linha detectada
            no laser','Linha detectada no alvo','Centro do alvo')
42
43  %% Triangulacao
44      ImgOrg1=imread('Alvo2.png');
45      ImgOrg2=imread('Alvo22.png');
46      run Triangulacao.m
47      figure, imshow(Img_Org), hold on
48      plot (Edgesp1(:,1),Edgesp1(:,2),'LineWidth',2,'Color','blue');
49      plot(Cx,Cy,'r+');
50      hold off
51      title ('Projecao do Laser deslocado na imagem original')
52      legend('Centro do alvo')
```

I.2.1 Calibração do Centro Óptico

Aqui está contido o código *Centro_Opt.m* para calibração do centro óptico.

```
1  %% Calculo do Centro Optico da camera
2
3  %% Declaracao de variaveis, celulas e matrizes
4
5       n=3; %Alterar conforme a quantidade de imagens
6       Col = 7; Row = 7; T = Col*Row; %Alterar conforme o padrao de
            calibracao usado
7       k = 98; j = 49;
8       ImgAux = cell(1,n);
9       Img = cell(1,n);
10      points = cell(j,1);
11
12       for u= 1:n
13          ImgPoints = zeros(j, 2, n);
14          ImgPointsAux = zeros(j, 2, n);
```

```
15        end
16
17        for u= 1:k
18          Points= zeros(k,2);
19          Results = zeros(k,1);
20        end
21
22        for u= 1:Col
23          Sol= zeros(u,2);
24        end
25
26 %%Deteccao dos circulos dispostos no padrao
27      for i = 1:n
28          ImgAux{i} = imread(sprintf('Calibracao_%d.png', i));
29          Img{i} = im2bw(ImgAux{i},0.3);
30          figure
31          imshow(Img{i});
32          [centers, radii, metric] = imfindcircles(Img{i} , [35 100]);
33          centersStrong = centers(1:j, : );
34          ImgPointsAux(:, :, i) = centersStrong;
35          viscircles(centersStrong, radii,'EdgeColor','g');
36          hold on
37          plot(centersStrong(:,1),centersStrong(:,2), 'r+');
38      end
39
40      %Ordenacao dos pontos detectados
41          for i=1:n
42              ImgPointsAux(1:j,:,i)=sortrows(ImgPointsAux(1:j,:,i));
43              q=Col;
44              for r=1:Col:49
45                  ImgPoints(r:q,:,i)=sortrows(ImgPointsAux(r:q,:,i),2);
46                  q=q+Col;
47              end
48          end
49 %%Determinacao dos sistemas de equacoes
50      %Obtencao das variaveis da equacao Cx*(Yf - Yf') + Cy*(Xf' - Xf) =
            Xf'*Yf - Xf-Yf'
51           x=1; y=2; Img1=1; Img2=2; Img3=3;
52           for i=1:j
53              points{i}=i;
54              %Yf - Yf'
```

```
55              A1=ImgPoints(points{i},y,Img1) - ImgPoints(points{i},y,
                    Img2);
56              A2=ImgPoints(points{i},y,Img2) - ImgPoints(points{i},y,
                    Img3);
57              A3=ImgPoints(points{i},y,Img1) - ImgPoints(points{i},y,
                    Img3);
58
59              %Xf' - Xf
60              B1=ImgPoints(points{i},x,Img2) - ImgPoints(points{i},x,
                    Img1);
61              B2=ImgPoints(points{i},x,Img3) - ImgPoints(points{i},x,
                    Img2);
62              B3=ImgPoints(points{i},x,Img3) - ImgPoints(points{i},x,
                    Img1);
63
64              %Xf'*Yf - Xf-Yf'
65              C1=(ImgPoints(points{i},x,Img2)*ImgPoints(points{i},y,Img1
                    )) - (ImgPoints(points{i},x,Img1)*ImgPoints(points{i},y
                    ,Img2));
66              C2=(ImgPoints(points{i},x,Img3)*ImgPoints(points{i},y,Img2
                    )) - (ImgPoints(points{i},x,Img2)*ImgPoints(points{i},y
                    ,Img3));
67              C3=(ImgPoints(points{i},x,Img3)*ImgPoints(points{i},y,Img1
                    )) - (ImgPoints(points{i},x,Img1)*ImgPoints(points{i},y
                    ,Img3));
68
69              Points(i,1)= A1; Points(i+j,1)= A2; Points(i+k,1)= A3;
70              Points(i,2)= B1; Points(i+j,2)= B2; Points(i+k,2)= B3;
71              Results(i,1)=C1; Results(i+j,1)=C2; Results(i+k,1)=C3;
72
73          end
74
75  %% Solucoes do sistema de equacoes da forma Points*[x;y] = Results
        usando o metodo dos minimos quadrados
76      Sol(1,:) = Points(1:j,:)\Results(1:j,:); %Imagens 1,2
77      Sol(2,:) = Points(50:k,:)\Results(50:k,:); %Imagens 2,3
78      Sol(3,:) = Points(k:147,:)\Results(k:147,:); %Imagens 1,3
79      Sol(4,:) = Points(1:k,:)\Results(1:k,:); %Imagens 1,2 e 2,3
80      Sol(5,:) = Points([1:j,k:147],:)\Results([1:j,k:147],:) %Imagens
            1,2 e 1,3
81      Sol(6,:) = Points(50:147,:)\Results(50:147,:); %Imagens 2,3 e 1,3
82      Sol(7,:) = Points\Results; %Imagens 1,2 e 2,3 e 1,3
```

```
83
84 %% Media dos valores obtidos
85      OptCenter = mean(Sol);
86      Cx = OptCenter(1,1);
87      Cy = OptCenter(1,2);
```

I.2.2 Alinhamento do Centro do Alvo com o Centro da Imagem

Aqui está contido o código *Translacao_Img.m* para calcular a distância que deve-se transladar a iamgem para que o centro óptico se alinhe com o centro do alvo.

```
1  %% Translacao do centro do alvo para o centro optico
2
3  %% Identificacao dos centroides
4
5      %Leitura das imagens
6          ImgVec = imread('Alvo_1.png'); %Alvo sem laser
7          %figure, imshow(ImgVec)
8
9      %Binarizacao
10         ImgBin = im2bw(ImgVec, 0.2); %Variar entre 0.1 e 0.3 para
               detectar o circulo
11         %figure, imshow(ImgBin)
12
13     %Deteccao do circulo no alvo
14         [centersBright, radiiBright] = imfindcircles(ImgBin,[50
               600],...
15         'ObjectPolarity','bright','Sensitivity',0.92);
16         centerCircle = centersBright(1, : );
17         %figure, imshow(ImgBin), hold on
18         %plot(centerCircle(1,1),centerCircle(1,2),'g+');
19         %hold off
20
21 %% Calculo da distancia para translacao da imagem
22     Dx= Cx - centerCircle(1,1);
23     Dy= Cy - centerCircle(1,2);
24
25 %% Coordenadas antes e apos o deslocamento
26     %Pontos base
27         u1=centerCircle(1,1)+radiiBright; v1=centerCircle(1,2);
28         u2=centerCircle(1,1); v2=centerCircle(1,2);
29         u3=centerCircle(1,1); v3=centerCircle(1,2)+radiiBright;
```

```
30
31      %Pontos desejados
32          u1_=u1+Dx; v1_=v1+Dy;
33          u2_=u2+Dx; v2_=v2+Dy;
34          u3_=u3+Dx; v3_=v3+Dy;
35
36  %% Translacao da imagem
37      ImgT = imtranslate(ImgBin,[Dx, Dy]);
38      %figure, imshow(ImgT)
39      ImgOrg = imtranslate(ImgVec,[Dx, Dy]); %Alvo nao binarizado
```

I.2.3 Controle Servo Visual Baseado em Imagem

Aqui está contido o código *IBVS.m* para determinação da posição a ser atingida pelo efetuador terminal do manipulador robótico.

```
1  %% Implementacao do Algoritmo IBVS
2
3  %% Parametros iniciais
4      %Comprimento do pixel, distancia focal e z
5       S = radiiBright/18;
6       f=16;
7       Z=1;
8       lambda=1;
9
10     %Coordenadas do centro optico
11         u0 = OptCenter(1,1);
12         v0 = OptCenter(1,2);
13
14 %% Coordenadas atuais e coordenadas desejadas
15     %Pontos obtidos em Translacao_Img.m
16         p1=[u1;v1]; p1_=[u1_;v1_];
17         p2=[u2;v2]; p2_=[u2_;v2_];
18         p3=[u3;v3]; p3_=[u3_;v3_];
19
20     %Offset
21         point1=p1_-p1;
22         point2=p2_-p2;
23         point3=p3_-p3;
24         Offset=[point1;point2;point3];
25
26 %% Calculo da velocidade
```

```
27      % Jacobiano da Imagem
28          Ls=[-S*f/Z 0 (u1-u0)/Z (v1-v0)*(u1-u0)/(S*f) -S*f-((u1-u0)^2)
                /(S*f) (v1-v0);
29              0 -S*f/Z (v1-v0)/Z S*f+((v1-v0)^2)/(S*f) -(u1-u0)*(v1-v0)
                    /(S*f) -(u1-u0);
30              -S*f/Z 0 (u2-u0)/Z (v2-v0)*(u2-u0)/(S*f) -S*f-((u2-u0)^2)
                    /(S*f) (v2-v0);
31              0 -S*f/Z (v2-v0)/Z S*f+((v2-v0)^2)/(S*f) -(u2-u0)*(v2-v0)
                    /(S*f) -(u2-u0);
32              -S*f/Z 0 (u3-u0)/Z (v3-v0)*(u3-u0)/(S*f) -S*f-((u3-u0)^2)
                    /(S*f) (v3-v0);
33              0 -S/Z (v3-v0)/Z S*f+((v3-v0)^2)/(S*f) -(u3-u0)*(v3-v0)/(S
                    *f) -(u3-u0)];
34
35      % Velocidade desejada
36          %Velocity=inv(Ls)*Position;
37          Velocity=lambda*Ls\Offset;
38
39  %% Posicao final
40      p1=Velocity(1,1)+440.1;
41      p2=Velocity(2,1)+0;
42      p3=Velocity(3,1)+711.9;
```

I.2.4 Determinação das Retas que Definem o *Laser* e as Linhas do Alvo

Os códigos a seguir apresentam a detecção das linhas nas bordas desejadas das imagens binarizadas a partir da Transformada de Hough. Após essa detecção, calcula-se as retas que mais se adequam ao projeto.

I.2.4.1 Preparação das Imagens

Aqui está contido o código *Preparacao_Img.m*.

```
1  %%Preparacao
2   n=2;
3   ImgVec = cell(1,n);
4   ImgID = 11;
5
6  %%Deteccao do centroide
7      %Leitura das imagens
8          ImgVec{1} = imread('Alvo_1.png'); %Alvo sem laser
9          %figure, imshow(ImgVec{1})
```

```
10          ImgVec{2} = imread(sprintf('Alvo_%d.png', ImgID)); %Alvo com
                laser
11          %figure, imshow(ImgVec{2})
12
13      %Imagens para triangulacao
14          %ImgVec{1} = ImgOrg1;
15          %ImgVec{2} = ImgOrg2;
16      %Binarizacao
17          ImgBin = im2bw(ImgVec{1}, 0.2); %Variar entre 0.1 e 0.3 para
                detectar o circulo
18          %figure, imshow(ImgBin)
19          ImgL = im2bw(ImgVec{2}, 0.9); %Binarizacao para detectar laser
20          %figure, imshow(ImgBin2)
21          ImgR = im2bw(ImgVec{1}, 0.2); %Binarizacao para detectar cruz
22          %figure, imshow(ImgBin1)
23
24      %Deteccao do centroide no alvo sem laser
25          [centersBright, radiiBright] = imfindcircles(ImgBin,[50
                600],...
26          'ObjectPolarity','bright','Sensitivity',0.92);
27          centerCircle = centersBright(1, : );
28          %figure, imshow(ImgBin), hold on
29          %plot(centerCircle(1,1),centerCircle(1,2),'g+');
30          hold off
31
32  % Redimensionamento da imagem sem o laser
33      %Coordenadas do canto superior esquerdo
34          x_r=centerCircle(1,1)-radiiBright/2;
35          y_r=centerCircle(1,2)-radiiBright/2;
36      %Recorte
37          ImgR = imcrop(ImgR,[x_r, y_r, radiiBright, radiiBright]);
38      %Inversao de cores
39          ImgR = imcomplement(ImgR);
40          %figure, imshow(ImgR)
```

I.2.4.2 Transformada de Hough

Aqui está contido o código *Transf_Hough.m*.

```
1  %% Transformada de Hough
2
3  %Transformada de Hough do Alvo com Laser
```

```
        [H,T,R] = hough(ImgL,'RhoResolution',0.7,'Theta',-90:0.5:89);
        P  = houghpeaks(H,50,'threshold',ceil(0.4*max(H(:))));
        lines = houghlines(ImgL,T,R,P,'FillGap',30,'MinLength',30);
        figure, imshow(ImgL), hold on
        run CalcNormal.m

%Transformada de Hough do Alvo sem Laser
        [H,T,R] = hough(ImgR,'RhoResolution',0.6,'Theta',-90:0.5:89);
        P  = houghpeaks(H,50,'threshold', ceil(0.4*max(H(:))));
        lines = houghlines(ImgR,T,R,P,'FillGap',30,'MinLength',50);
        figure, imshow(ImgR), hold on
        run CalcIncl.m
```

I.2.4.3 Seleção das retas que melhor se adequam ao projeto

Aqui está contido o código *CalcNormal.m*, responsável por calcular as retas normais ao centro do alvo e selecionar a menor, tendo como base as retas encontradas pela Transformada de Hough.

```
%% Calculo da menor normal entre as linhas detectadas e o centro do
   alvo

%% Declaracao de vetores e matrizes
    for u= 1:2
            CrossP = zeros(length(lines), u);
            Crossp1 = zeros(u,u);
            Edgesp1 = zeros(u,u);
            Dist = zeros(length(lines),1);
            Perp = zeros(length(lines),u);
            Edges = zeros(u,u,length(lines));
    end
          a=cell(1,2);
          b=cell(1,2);
          f=1;

%% Calculo das retas normais
    for k = 1:length(lines)
        xy = [lines(k).point1; lines(k).point2];
        %plot(xy(:,1),xy(:,2),'LineWidth',2,'Color','green');

        %Coeficientes da reta obtida por Hough
            coefficients = polyfit([xy(1,1);xy(2,1)], [xy(1,2);xy(2,2)
               ], 1);
```

```
23              a{k} = coefficients(1,1);
24              b{k} = coefficients(1,2);
25
26          %Ponto de interseccao entre as retas
27              CrossP(k,1)=(a{k}*b{k} - a{k}*centerCircle(1,2) -
                    centerCircle(1,1))/(-1 -a{k}^2);
28              CrossP(k,2)= ((a{k}*b{k} - a{k}*centerCircle(1,2) -
                    centerCircle(1,1))/(-1 -a{k}^2)*(-1/a{k})) +
                    centerCircle(1,2) + centerCircle(1,1)/a{k};
29              %plot([CrossP(k,1);centerCircle(1,1)],[CrossP(k,2);
                    centerCircle(1,2)],'LineWidth',2,'Color', 'yellow');
30
31          %Distancia entre os pontos
32              vec1 = [xy(1,1), xy(1,2); CrossP(k,1), CrossP(k,2)];
33              vec2 = [CrossP(k,1), CrossP(k,2); xy(2,1), xy(2,2)];
34              vec3 = [xy(1,1), xy(1,2); xy(2,1), xy(2,2)];
35              dist1 = pdist(vec1,'euclidean');
36              dist2 = pdist(vec2,'euclidean');
37              dist3 = pdist(vec3,'euclidean');
38
39          %Selecao dos pontos de interseccao que realmente cortam as
                retas obtidas por Hough
40              if(dist1+dist2==dist3 || (dist1+dist2>dist3 && dist1+dist2
                    <dist3+0.0000001) || (dist3>dist1+dist2 && dist3<dist1+
                    dist2+0.0000001))
41                  vec4 = [CrossP(k,1), CrossP(k,2); centerCircle(1,1),
                        centerCircle(1,2)];
42                  Dist(f,1) = pdist(vec4, 'euclidean');
43                  Edges(:,1,f) = xy(:,1); Edges(:,2,f) = xy(:,2);
44                  Perp(f,1) = CrossP(k,1); Perp(f,2)= CrossP(k,2);
45                  %plot([Perp(f,1);centerCircle(1,1)],[Perp(f,2);
                        centerCircle(1,2)],'LineWidth',2,'Color', 'yellow')
                        ;
46                  f = f+1;
47              end
48      end
49      f = f-1;
50
51  %% Calculo da menor normal
52      D = Dist(1,1);
53      Crossp1(1,1) = Perp(1,1);
54      Crossp1(1,2) = Perp(1,2);
```

```
55      Edgesp1(:,1) = Edges(:,1,1);
56      Edgesp1(:,2) = Edges(:,2,1);
57      for j=1:(f-1)
58          if Dist(j+1,1)<D
59              D=Dist(j+1,1);
60              Crossp1(1,1) = Perp(j+1,1);
61              Crossp1(1,2) = Perp(j+1,2);
62              Edgesp1(:,1) = Edges(:,1,j+1);
63              Edgesp1(:,2) = Edges(:,2,j+1);
64          end
65      end
66      plot([centerCircle(1,1) Crossp1(1,1)],[centerCircle(1,2) Crossp1
            (1,2)],'LineWidth',2,'Color', 'red');
67      plot (Edgesp1(:,1), Edgesp1(:,2), 'LineWidth', 2, 'Color', 'blue')
            ;
68      plot(centerCircle(1,1),centerCircle(1,2),'g+');
69      hold off
70      title ('Menor distancia entre o laser e o centro do alvo')
71      legend('Menor distancia entre a linha e o alvo','Linha do laser',
            'Centro do alvo')
72
73      %Calculo da inclinacao da reta
74          coefficients = polyfit([Edgesp1(1,1);Edgesp1(2,1)], [Edgesp1
                (1,2);Edgesp1(2,2)], 1);
75          incl = coefficients(1,1);
76          incl_deg = atand(incl);
```

Aqui está contido o código *Calc_Incl.m*, responsável por calcular a reta com inclinação mais próxima à reta do *laser*, a fim de garantir que estejam em orientações próximas.

```
1  %% Calculo da reta do alvo com inclinacao parecida a reta do laser
2
3  %% Declaracao de vetores e matrizes
4      for u= 1:2
5              CrossP = zeros(length(lines), u);
6              Crossp2 = zeros(u,u);
7              Edgesp2 = zeros(u,u);
8              Dist = zeros(length(lines),1);
9              a_deg = zeros(length(lines),1);
10             dif = zeros(length(lines),1);
11             Perp = zeros(length(lines),u);
12             Edges = zeros(u,u,length(lines));
13     end
```

```
14            a=cell(1,2);
15            b=cell(1,2);
16            f=1;
17
18 %% Adaptacao para imagem cortada
19    Cx_= radiiBright/2;
20    Cy_ = radiiBright/2;
21
22 %% Calculo das retas normais
23     for k = 1:length(lines)
24         xy = [lines(k).point1; lines(k).point2];
25         %plot(xy(:,1),xy(:,2),'LineWidth',2,'Color','green');
26
27         %Coeficientes da reta obtida por Hough
28              coefficients = polyfit([xy(1,1);xy(2,1)], [xy(1,2);xy(2,2)
                    ], 1);
29              a{k} = coefficients(1,1);
30              b{k} = coefficients(1,2);
31
32         %Ponto de interseccao entre as retas
33              CrossP(k,1)=(a{k}*b{k} - a{k}*Cy_ -Cx_)/(-1 -a{k}^2);
34              CrossP(k,2)= ((a{k}*b{k} - a{k}*Cy_ -Cx_)/(-1 -a{k}^2)
                    *(-1/a{k})) + Cy_ + Cx_/a{k};
35             %plot([CrossP(k,1);Cx],[CrossP(k,2);Cy],'LineWidth',2,'
                    Color', 'yellow');
36
37         %Distancia entre os pontos
38              vec1 = [xy(1,1), xy(1,2); CrossP(k,1), CrossP(k,2)];
39              vec2 = [CrossP(k,1), CrossP(k,2); xy(2,1), xy(2,2)];
40              vec3 = [xy(1,1), xy(1,2); xy(2,1), xy(2,2)];
41              dist1 = pdist(vec1,'euclidean');
42              dist2 = pdist(vec2,'euclidean');
43              dist3 = pdist(vec3,'euclidean');
44
45         %Selecao dos pontos de interseccao que realmente cortam as
               retas obtidas por Hough
46              if(dist1+dist2==dist3 || (dist1+dist2>dist3 && dist1+dist2
                    <dist3+0.0000001) || (dist3>dist1+dist2 && dist3<dist1+
                    dist2+0.0000001))
47                   vec4 = [CrossP(k,1), CrossP(k,2); Cx_, Cy_];
48                   Dist(f,1) = pdist(vec4, 'euclidean');
49                   Edges(:,1,f) = xy(:,1); Edges(:,2,f) = xy(:,2);
```

```
50                  Perp(f,1) = CrossP(k,1); Perp(f,2)= CrossP(k,2);
51                  %plot([Edges(1,1,f);Edges(2,1,f)],[Edges(1,2,f);Edges
                        (2,2,f)],'LineWidth',2,'Color', 'green');
52                  a_deg(f,1) = atand(a{k});
53                  dif(f,1)=abs(abs(incl_deg)-abs(a_deg(f,1)));
54                  f = f+1;
55              end
56      end
57      f = f-1;
58
59  %% Calculo da reta de inclinacao mais proxima ao laser
60      D = Dist(1,1);
61      ang = dif(1,1);
62      incl2_deg=a_deg(1,1);
63      Crossp2(1,1) = Perp(1,1);
64      Crossp2(1,2) = Perp(1,2);
65      Edgesp2(:,1) = Edges(:,1,1);
66      Edgesp2(:,2) = Edges(:,2,1);
67      for j=1:(f-1)
68          if dif(j+1,1)<ang
69              ang=dif(j+1,1);
70              incl2_deg=a_deg(j+1,1);
71              Crossp2(1,1) = Perp(j+1,1);
72              Crossp2(1,2) = Perp(j+1,2);
73              Edgesp2(:,1) = Edges(:,1,j+1);
74              Edgesp2(:,2) = Edges(:,2,j+1);
75          end
76      end
77      %plot([Cx_ Crossp2(1,1)],[Cy_ Crossp2(1,2)],'LineWidth',2,'Color',
            'red');
78      plot (Edgesp2(:,1), Edgesp2(:,2), 'LineWidth', 2, 'Color', 'blue')
          ;
79      plot(Cx_,Cy_,'r+');
80      hold off
81      title ('Deteccao da linha do alvo na imagem recortada')
82      legend('Linha do alvo','Centro do alvo')
83
84
85  %% Adaptacao para imagem original
86      Crossp2(1,1)=Crossp2(1,1)+centerCircle(1,1)-Cx_;
87      Crossp2(1,2)=Crossp2(1,2)+centerCircle(1,2)-Cy_;
88      Edgesp2(:,1)=Edgesp2(:,1)+centerCircle(1,1)-Cx_;
```

```
89      Edgesp2(:,2)=Edgesp2(:,2)+centerCircle(1,2)-Cy_;
90
91
92      figure, imshow(ImgVec{1}), hold on
93      %plot([centerCircle(1,1) Crossp2(1,1)],[centerCircle(1,2) Crossp2
            (1,2)],'LineWidth',2,'Color', 'red');
94      plot (Edgesp2(:,1),Edgesp2(:,2), 'LineWidth', 2, 'Color', 'blue');
95      plot(centerCircle(1,1),centerCircle(1,2),'r+');
96      hold off
97      title ('Deteccao da linha do alvo na imagem original')
98      legend('Linha do alvo','Centro do alvo')
```

I.2.5 Triangulação

Aqui está contido o código *Triangulacao.m*, responsável por calcular a distância Z_c a partir dos pontos da reta normal entre o *laser* e a linha do alvo.

```
1  %%Declaracao de Variaveis
2       dist_b=140;
3       ang_beta=10;
4       dist_f=16;
5
6  %%Calculo Zc
7      xp = Crossp1(1,1); yp = Crossp1(1,2);
8      xp_ = Cx; yp_ = Cy;
9
10     Zc = dist_b/((1/tand(ang_beta)) + xp/dist_f);
11     Zc_ = dist_b/((1/tand(ang_beta)) + xp_/dist_f);
12     %delta_Zc = Zc_ - Zc;
13
14 %%Calculo da escala
15     if Zc>Zc_
16         scale = Zc/Zc_;
17     elseif Zc<Zc_
18         scale = Zc_/Zc;
19     end
20     Im_Org=ImgOrg2;
21     ImgOrg2 = imresize(ImgVec{2},scale);
22     ImgOrg1 = imresize(ImgVec{1},scale);
23     run Preparacao_Img.m
24     run Transf_Hough.m
```