MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA MECÂNICA

SISTEMA DE VISÃO COMPUTACIONAL APLICADO A UM ROBÔ CILÍNDRICO ACIONADO PNEUMATICAMENTE

por

Betânia Vargas Oliveira Medina

Dissertação para obtenção do Título de
Mestre em Engenharia

Porto Alegre, fevereiro de 2015

SISTEMA DE VISÃO COMPUTACIONAL APLICADO A UM ROBÔ CILÍNDRICO ACIONADO PNEUMATICAMENTE

por

Betânia Vargas Oliveira Medina
Tecnóloga em Automação Industrial

Dissertação submetida ao Programa de Pós-Graduação em Engenharia Mecânica, da Escola de Engenharia da Universidade Federal do Rio Grande do Sul, como parte dos requisitos necessários para a obtenção do Título de

Mestre em Engenharia

Área de Concentração: Processos de Fabricação

Orientador: Prof. Dr. Eduardo André Perondi

Aprovada por:

Prof. Dr. Patric Daniel Neis .....DEMEC / UFRGS

Prof. Dr. Rafael Antônio Comparsi Laranja .....PROMEC / UFRGS

Prof. Dr. Soraia Raupp Musse.....Faculdade de Informática / PUCRS

Prof. Dr. Rogério José Marczak
Coordenador do PROMEC

Porto Alegre, 27 de fevereiro de 2015.

AGRADECIMENTOS

À Deus pela oportunidade da vida e por ter me dado saúde e força para superar as dificuldades.

Aos meus pais Janeti Terezinha Cardoso Vargas e Jaques dos Santos Oliveira, a quem devo meu caráter e disciplina ao trabalho, por sempre me incentivarem a buscar o meu desenvolvimento moral e intelectual.

Ao meu marido Robson Rocha Medina pela compreensão nas horas em que me fiz ausente e pelo apoio, carinho e incentivo nos momentos difíceis, de desânimo e cansaço.

Ao meu orientador Prof. Dr. Eduardo André Perondi pela confiança depositada, dedicação e paciência em todas as etapas do desenvolvimento deste trabalho.

À Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) por ter me proporcionado o incentivo financeiro, através da bolsa de estudos de mestrado.

À UFRGS, seu corpo docente, direção e administração que oportunizaram a janela que hoje vislumbro um horizonte superior, eivado pela confiança no mérito e ética nela presentes.

Aos colegas do PROMEC e do LAMECC, em especial ao Leonardo Missiaggia, ao Carlos Arthur Carvalho Sarmanho Junior e ao Rafael Crespo Izquierdo pelo suporte e paciência nas diversas etapas deste curso de mestrado.

Ao Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul - Campus Rio Grande, por ter cedido meu afastamento para estudos, permitindo assim, que eu pudesse me dedicar de forma integral para a realização deste trabalho.

Aos familiares e amigos que, de alguma maneira, me deram apoio com palavras de ânimo e coragem.

RESUMO

O reconhecimento da posição e orientação de objetos em uma imagem é importante para diversos segmentos da engenharia, como robótica, automação industrial e processos de fabricação, permitindo às linhas de produção que utilizam sistemas de visão, melhorias na qualidade e redução do tempo de produção. O presente trabalho consiste na elaboração de um sistema de visão computacional para um robô cilíndrico de cinco graus de liberdade acionado pneumaticamente. Como resultado da aplicação do método desenvolvido, obtêm-se a posição e orientação de peças a fim de que as mesmas possam ser capturadas corretamente pelo robô. Para a obtenção da posição e orientação das peças, utilizou-se o método de cálculo dos momentos para extração de características de uma imagem, além da relação entre suas coordenadas em pixels com o sistema de coordenadas do robô. O desenvolvimento do presente trabalho visou também a integrar a esse sistema de visão computacional, um algoritmo de planejamento de trajetórias do robô, o qual, após receber os valores das coordenadas necessárias, gera a trajetória a ser seguida pelo robô, de forma que este possa pegar a peça em uma determinada posição e deslocá-la até outra posição pré-determinada. Também faz parte do escopo deste trabalho, a integração do sistema de visão, incluindo o planejamento de trajetórias, a um algoritmo de controle dos atuadores com compensação de atrito e a realização de testes experimentais com manipulação de peças. Para a demonstração da aplicação do método através de testes experimentais, foi montada uma estrutura para suportar as câmeras e as peças a serem manipuladas, levando em conta o espaço de trabalho do robô. Para a validação do algoritmo proposto, foi realizado um estudo de caso com sua aplicação no desenvolvimento de um sistema de visualização e geração automática de trajetória para o robô. Os resultados obtidos mostram que o algoritmo proposto de visão computacional determina a posição e orientação das peças permitindo ao robô a captação e manipulação das mesmas.

Palavras-chave: visão computacional; momentos de uma imagem; robô pneumático; planejamento de trajetória, geração de trajetória para robôs manipuladores.

ABSTRACT

The recognition of the position and orientation of objects in an image is important for several technological areas in engineering, such as robotics, industrial automation and manufacturing processes, allowing production lines using vision systems, improvements in quality and reduction in production time. The present work consists of the development of a computer vision system for a pneumatically actuated cylindrical robot with five degrees of freedom. The application of the proposed method furnishes the position and orientation of pieces in a way that the robot could properly capture them. Position and orientation of the pieces are determined by means of a technique based on the method of calculating the moments for an image feature extraction and the relationship between their pixels coordinates with the robot coordinate system. The scope of the present work also comprises the integration of the computer vision system with a (previously developed) robot trajectory planning algorithm that use key-point coordinates (transmitted by the vision system) to generate the trajectory that must be followed by the robot, so that, departing from a given position, it moves suitably to another predetermined position. It is also object of this work, the integration of both vision system and trajectory planning algorithm with a (also previously developed) nonlinear control algorithm with friction compensation. Aiming at to demonstrate experimentally the application of the method, a special apparatus was mounted to support cameras and the pieces to be manipulated, taking into account the robot workspace. To validate the proposed algorithm, a case study was performed, with the results showing that the proposed computer vision algorithm determines the position and orientation of the pieces allowing the robot to capture and manipulation thereof.

Keywords: computer vision; image moments; pneumatic robot; pick-and-place trajectories generation, robot trajectory planning.

ÍNDICE

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS E ABREVIATURAS

CASIA	Academia Chinesa de Ciências
CCD	*Charge-Coupled Device*
CIE	*Commission Internationale de l'Eclairage*
CMOS	*Complementary Metal-Oxide-Semiconductor*
DH	*Denavit-Hartenberg*
DLT	*Direct Linear Transformation*
DOF	*Depth Of Field*
GL	Grau de Liberdade do Robô
ICC	Coeficiente de Correlação Intraclasse
IFR	*International Federation of Robotics*
LAMECC	Laboratório de Mecânica e Controle
P	Junta Prismática
PI	Plano da Imagem
PP	Projeção Perspectiva
R	Junta Rotacional
MMQ	Método dos Mínimos Quadrados
PROMEC	Programa de Pós-Graduação em Engenharia Mecânica
UFRGS	Universidade Federal do Rio Grande do Sul

.

LISTA DE SÍMBOLOS

α	Valor do ângulo de giro de um ponto P em torno no eixo *x* [rad]
β	Valor do ângulo de giro de um ponto P em torno no eixo *y* [rad]
$\boldsymbol{B}$	Matriz de orientação do efetuador
b_u	Valor do comprimento de um pixel [m]
b_v	Valor da altura de um pixel [m]
C_i	Classe *i* do histograma de uma imagem
d	Deslocamento linear de uma junta prismática [m]
d_i	Deslocamento linear da junta prismática *i* [m]
$d_{i_{min}}$	Valor mínimo da junta prismática *i* [m]
$d_{i_{max}}$	Valor máximo da junta prismática *i* [m]
$d_{i_{min\zeta}}$	Valor mínimo da junta prismática *i* considerando-se o amortecimento na junta[m]
$d_{i_{max\zeta}}$	Valor máximo da junta prismática *i* considerando-se o amortecimento na junta[m]
d_ζ	Valor do deslocamento linear gerado pelos amortecedores nas juntas [m]
dif	Valor da distância euclidiana no plano x,y entre o ponto inicial e final do robô [m]
dx	Valor inteiro a ser adicionado à coordenada *x* em uma operação de translação [m]
dy	Valor inteiro a ser adicionado à coordenada *y* em uma operação de translação [m]
dz	Valor inteiro a ser adicionado à coordenada *z* em uma operação de translação [m]
DE	Distância euclidiana entre dois pontos no espaço [m]
Δx	Módulo da diferença entre a coordenada *x* final menos a *x* inicial [m]
Δy	Módulo da diferença entre a coordenada *y* final menos a *y* inicial [m]
Δz	Módulo da diferença entre a coordenada *z* final menos a *z* inicial [m]
$\boldsymbol{e}$	Vetor que apresenta a posição final do efetuador
$\boldsymbol{e_x}$	Valor do eixo coordenado *x* da posição final do efetuador [m]
$\boldsymbol{e_y}$	Valor do eixo coordenado *y* da posição final do efetuador [m]
$\boldsymbol{e_z}$	Valor do eixo coordenado *z da* posição final do efetuador [m]
f	Valor da distância focal da câmera [m]
$f(u, v)$	Valor da intensidade do pixel *(u, v)*

h_i	Histograma da imagem
γ	Valor do ângulo de giro de um ponto P em torno no eixo *z* [rad]
$itm_1(k)$	Intensidade média do valor dos pixels pertencentes à classe C_1
$itm_1(k)$	Intensidade média do valor dos pixels pertencentes à classe C_2
$I_{A_{\overline{m}}}$	Valor da incerteza aleatória estimada da média de *n* medições
I_B	Valor da incerteza sistemática estimada
I_C	Valor das incertezas combinadas do Tipo A e do Tipo B
δI	Valor da incerteza final no resultado (*I*) de medições em um experimento
k_{ij}	Valor do elemento da matriz ***P*** na linha *i* e na coluna *j*
k	Limiar (nível de cinza que reparte o histograma da imagem em duas classes)
k^*	Limiar ótimo segundo Método de Osus
L	Quantidade de tons de cinza em uma imagem
$\boldsymbol{M}$	Matriz de representação de uma imagem
$\boldsymbol{M}(u,v)$	Termos da matriz *M*
$\overline{m}$	Valor da média aritmética das *n* medidas efetuadas
m_n	Valor da *n* medida efetuada
m_{pq}	Momento regular bidimensional de ordem *(p, q)*
μ_{pq}	Momento central bidimensional de ordem (*p, q*)
n_i	Quantidade de pixels da imagem (*Im*) que possuem a intensidade de cinza *i*
n_u	Valor da Altura de uma imagem digital [Pix]
n_v	Valor da Largura de uma imagem digital [Pix]
O	Origem do sistema de coordenadas
θ	Deslocamento angular de uma junta rotativa [rad]
θ_i	Deslocamento angular da junta rotativa *i* [rad]
θ_{4_i}	Deslocamento angular inicial da junta rotativa *4* [rad]
θ_{5_i}	Deslocamento angular inicial da junta rotativa *5* [rad]
θ_{4_f}	Deslocamento angular final da junta rotativa *4* [rad]
θ_{5_f}	Deslocamento angular final da junta rotativa *5* [rad]
$\theta_{i_{min}}$	Deslocamento angular mínimo da junta rotativa *i* [rad]
$\theta_{i_{max}}$	Deslocamento angular máximo da junta rotativa *i* [rad]

$\theta_{i_{min\zeta}}$	Deslocamento angular mínimo da junta rotativa *i* considerando-se o amortecimento na junta [rad]
$\theta_{i_{max\zeta}}$	Deslocamento angular máximo da junta rotativa *i* considerando-se o amortecimento na junta [rad]
θ_{ζ_i}	Deslocamento angular causado pelo amortecimento na junta *i* [rad]
θ	Deslocamento angular de uma junta rotativa [rad]
p	Valor do expoente ao qual a coordenada *u* está elevada no cálculo do momento
$\boldsymbol{p}$	Vetor que representa um ponto no espaço
$\rho(x, y)$	Função de distribuição de densidade do corpo no ponto (x, y)
$\boldsymbol{P}$	Matriz de calibração de câmera
$\boldsymbol{PC}$	Matriz de pontos (n) por onde o manipulador deve passar (*5 x n*)
$\boldsymbol{p_i}$	Iésimo ponto do espaço tridimensional
$\boldsymbol{p'}$	Ponto Transformado (Transladado e/ou Rotacionado)
$\boldsymbol{p_w}$	Ponto no espaço tridimensional ou em coordenadas de mundo (*w*)
$\boldsymbol{p_w^h}$	Ponto no espaço tridimensional em coordenadas homogêneas
$Pb_1(k)$	Probabilidade do nível de cinza *k* ser da classe C_1
$Pb_2(k)$	Probabilidade do nível de cinza *k* ser da classe C_2
q	Valor do expoente ao qual a coordenada *v* está elevada no cálculo do momento
r	Raio de giro do semiarco da trajetória *pick-and-place*
R	Razão entre os momentos centrais μ_{02} e μ_{20}
$\boldsymbol{R}$	Matriz de rotação
res	Resolução [m]
$\boldsymbol{Rx(P)}$	Vetor de rotação do ponto *p* em torno do eixo *x*
$\boldsymbol{Ry(P)}$	Vetor de rotação do ponto *p* em torno do eixo *y*
$\boldsymbol{Rz(P)}$	Vetor de rotação do ponto *p* em torno do eixo *z*
$s_{m_n}^2$	Valor da variância estimada das *n* medidas efetuadas m_n
s_{m_n}	Valor do desvio padrão estimado das *n* medidas efetuadas m_n
$s_{\bar{m}}^2$	Valor da variância experimental da média das *n* medidas efetuadas m_n
$s_{\bar{m}}$	Valor do desvio padrão experimental da média das *n* medidas efetuadas m_n
$\sigma_1^2(k)$	Variância para distribuição de probabilidade do nível de cinza *k* ser da classe C_1
$\sigma_2^2(k)$	Variância para distribuição de probabilidade do nível de cinza *k* ser da classe C_2

$\sigma_B^2(k)$	Variância intraclasses
t_{cf_g}	Valor da distribuição *t* de *student* de Willian Gosset para *g* graus de liberdade e nível de confiança *cf*
t_0	Tempo inicial [s]
t_1	Tempo ao instante 1 [s]
t_2	Tempo ao instante 2 [s]
T(P)	Vetor de translação do ponto P
${}^{i}T_j$	Matriz de Transformação do sistema de coordenadas do elo *j* para o elo *i*
u	Valor do eixo coordenado horizontal *u* na matriz *M(u,v)* de uma imagem [Pix]
$\bar{u}$	Valor da coordenada *u* central de uma imagem [Pix]
u_c	Valor da coordenada *u* do centro ótico mapeado em pixel [Pix]
$\boldsymbol{u^h}$	Vetor que gera a equação de projeção final
u_s	Valor da coordenada *u* de pixel tamanho *s* [Pix]
v	Valor do eixo coordenado vertical *v* na matriz *M(u,v)* de uma imagem [Pix]
$\bar{v}$	Valor da coordenada *v* central de uma imagem [Pix]
v_c	Valor da coordenada *v* do centro ótico mapeado em pixel [Pix]
vp	Valor percentual da distância euclidiana no plano x,y entre o ponto inicial e final
v_s	Valor da coordenada *s* de pixel tamanho *s* [Pix]
x	Valor do eixo coordenado *x* no espaço cartesiano [m]
x_c	Valor do eixo coordenado *x* do ponto onde se encontra a câmera [m]
$\boldsymbol{x_c}$	Vetor de coordenadas do ponto onde se encontra a câmera [m]
$\boldsymbol{x_c^h}$	Vetor de coordenadas do ponto onde se encontra a câmera em coordenadas homogêneas
x_i	Valor do eixo coordenado *x* do ponto inicial do efetuador [m]
x_f	Valor do eixo coordenado *x* do ponto final do efetuador [m]
x_0	Valor do eixo coordenado *x* do vetor de translação $\boldsymbol{x_0}$ [m]
$\boldsymbol{x_0}$	Vetor de translação
$\boldsymbol{x_0^h}$	Vetor de translação em coordenadas homogêneas
x_p	Valor do eixo coordenado *x* do ponto *P* [m]
x_p'	Valor do eixo coordenado *x* do ponto *P* transformado (transladado ou rotacionado) [m]

x_w ou X	Valor do eixo coordenado *x* no espaço cartesiano tridimensional [m]
$\boldsymbol{x_w}$	Vetor de coordenadas mundo
$\boldsymbol{x_w^h}$	Vetor de coordenadas mundo em coordenadas homogêneas
y	Valor do eixo coordenado *y* no espaço cartesiano [m]
y_c	Valor do eixo coordenado *y* do ponto onde se encontra a câmera [m]
y_i	Valor do eixo coordenado *y* do ponto inicial do efetuador [m]
y_f	Valor do eixo coordenado *y* do ponto final do efetuador [m]
y_0	Valor do eixo coordenado *y* do vetor de translação $\boldsymbol{x_0}$ [m]
y_p	Valor de *y* do ponto *P* [m]
$y_p{}'$	Valor de *y* do ponto *P* transformado (transladado ou rotacionado) [m]
y_w ou Y	Valor do eixo coordenado *y* no espaço cartesiano tridimensional [m]
z	Valor do eixo coordenado *z* no espaço cartesiano [m]
z_c	Valor do eixo coordenado *z* do ponto onde se encontra a câmera [m]
z_i	Valor do eixo coordenado *z* do ponto inicial do efetuador [m]
z_f	Valor do eixo coordenado *z* do ponto final do efetuador [m]
z_{min}	Valor mínimo do eixo coordenado *z* atingível pelo efetuador [m]
z_{max}	Valor máximo do eixo coordenado *z* atingível pelo efetuador [m]
z_0	Valor do eixo coordenado *z* do vetor de translação $\boldsymbol{x_0}$ [m]
z_p	Valor de *z* do ponto *P* [m]
$z_p{}'$	Valor de *z* do ponto *P* transformado (transladado ou rotacionado) [m]
z_w ou Z	Valor do eixo coordenado *z* no espaço cartesiano tridimensional [m]
ω	Valor do peso de uma matriz em coordenadas homogêneas

1 INTRODUÇÃO

A robótica é um ramo que cresce incessantemente, nas mais diversas áreas, em aplicações comerciais, residenciais e industriais. Segundo IFR, 2014, um estudo da IFR (*International Federation of Robotics*) mostra que, no ano de 2013, a venda de robôs chegou a um recorde histórico de mais de 178 mil unidades. Esse estudo mostra também que a utilização de robôs manipuladores em sistemas industriais automatizados tornou-se ainda mais frequente, já que melhorias na qualidade e aumento da velocidade de produção requerem sistemas cada vez mais sofisticados e com tecnologia avançada. Para que isto seja possível, é necessário que esses sistemas possam tomar decisões com maior grau de autonomia possível, com pouca ou nenhuma intervenção humana. Para tanto, a visão computacional torna-se uma grande aliada a tais sistemas.

Diante deste quadro, está sendo desenvolvido no Laboratório de Mecatrônica e Controle (LAMECC) da Universidade Federal do Rio Grande do Sul (UFRGS), um robô de configuração cilíndrica com cinco graus de liberdade acionado pneumaticamente (Sarmanho, 2014). Assim, o desenvolvimento do presente trabalho visa a integrar a esse manipulador robótico um sistema de visão computacional que identifique as peças a serem manipuladas (posição e ângulo de giro em relação ao sistema de coordenadas cartesianas do robô) informando a um algoritmo de planejamento de trajetórias do robô os valores de coordenadas necessárias para gerar a trajetória a ser seguida pelo robô, de forma que esse possa pegar a peça em uma determinada posição e deslocá-la até outra posição pré-determinada.

1.1 Descrição do Problema

O robô supracitado apresenta uma configuração única, desenvolvida inteiramente na UFRGS, e é objeto de estudos nas áreas de controle, robótica e automação. Assim, as diversas áreas afeitas ao desenvolvimento de um robô manipulador estão sendo pesquisadas, tais como o planejamento de trajetórias de cada atuador [Missiaggia, 2014], o controle dos atuadores pneumáticos rotacionais [Rijo, 2013] e translacionais utilizando técnicas de controle não linear com compensação de atrito [Sarmanho, 2014] e redes neurais [Gervini, 2014] e o desenvolvimento de uma arquitetura microcontrolada programável aplicada ao controle dos servoposicionadores pneumáticos [Cukla, 2012]. Além disso, outras pesquisas afins são

objeto de estudo no LAMECC/UFRGS, como a proposição de um algoritmo para programação de robôs baseada em linguagem de programação de CLPs por Diagramas de Blocos Funcionais (GRAFCET) e de uma estratégica sistemática para o desenvolvimento de controladores modulares aplicados a robôs manipuladores pneumáticos.

Até a fase que antecede o presente estudo, os desenvolvimentos foram realizados de forma praticamente independente, faltando, ainda, realizar a integração dos diversos sistemas, visando a obter um manipulador robótico que possa ser facilmente integrado a um ambiente produtivo. Assim, é necessário, por exemplo, integrar os sistemas de planejamento de trajetórias (realizado em ambiente de simulação) com o de controle dos atuadores (devidamente testado no ambiente de trabalho do robô). É importante destacar que, nesses trabalhos, a escolha dos pontos que a trajetória do robô deveria percorrer foi realizada utilizando curvas definidas individualmente para cada grau de liberdade, não havendo preocupação imediata com a trajetória resultante do efetuador, verificando somente se os pontos da trajetória desejada estavam dentro do espaço de trabalho do robô.

Visando a contribuir com o estabelecimento de um sistema de geração automática de trajetória, o presente trabalho consiste no desenvolvimento de um sistema de visão computacional que identifique a posição e a orientação das peças a serem manipuladas pelo robô, gerando vetores de coordenadas as quais o programa de planejamento de trajetórias utilizará para realizar as interpolações necessárias para a geração das trajetórias com o *jerk* [1] mínimo, segundo esquema desenvolvido por Missiaggia, 2014. Faz também parte do escopo deste trabalho, integrar estes esquemas com o sistema de controle dos atuadores, avaliando através de experimentos, a eficácia da estratégia proposta aplicada a situações em que o efetuador do robô deve atingir a localização e o ângulo adequado para que possa manipular o objeto identificado pelo sistema de visão.

1.2 Objetivo Geral

O objetivo geral do trabalho é desenvolver um sistema de visão computacional que identifique a posição e orientação de peças, gerando as coordenadas da trajetória que o robô deve percorrer para atingir a localização e o ângulo para manipulá-las. O sistema

[1] *Jerk* é a derivada primeira da aceleração, calculado pela terceira derivada da posição em relação ao tempo.

desenvolvido deve ser integrado ao sistema de controle (proposto por Sarmanho, 2014) e testado através de experimentos de manipulação de peças identificadas pelo sistema de visão.

1.3 Objetivos Específicos

Com a finalidade de atingir o objetivo geral do trabalho, propõe-se os seguintes objetivos específicos:

- Aplicar uma técnica de visão computacional para realizar a determinação da posição e orientação de um objeto em uma cena;
- Propor trajetórias para o robô realizar a movimentação entre dois pontos;
- Testar numericamente as trajetórias propostas integradas ao sistema de planejamento de trajetórias;
- Integrar o algoritmo de visão computacional aos sistemas de planejamento de trajetórias e de controle dos atuadores;
- Definir peças a serem manipuladas nos testes experimentais;
- Construir uma mesa para disposição das peças, assim como um suporte para a câmera, levando em consideração o espaço de trabalho do robô.

1.4 Organização do trabalho

O presente trabalho está organizado da seguinte maneira: no Capítulo 2 é apresentada a revisão bibliográfica, incluindo o estado da arte e a fundamentação teórica, a qual serve de embasamento para compreensão dos capítulos seguintes. No Capítulo 3, é apresentado um método de reconhecimento da orientação e posição de um objeto baseado no conceito de momento geométrico. No Capítulo 4 são apresentados os procedimentos de implantação do sistema no robô pneumático de 5 graus de liberdade e os respectivos resultados de simulações e experimentos. Por fim, no Capítulo 5 são apresentadas as conclusões, considerações finais e sugestão de trabalhos futuros.

2 REVISÃO BIBLIOGRÁFICA

Neste capítulo está apresentado o referencial teórico que serviu como embasamento para realização deste trabalho. Inicialmente, é apresentado o estado da arte abrangendo os trabalhos atuais que versam sobre a visão computacional e o planejamento de trajetória aplicados à robótica. Em seguida, é apresentada a fundamentação teórica considerada necessária para a fácil compreensão do trabalho.

2.1 Estado da arte

Como já mencionado anteriormente, os robôs vêm sendo cada vez mais utilizados nas mais diversas áreas visando a substituir o homem em tarefas repetitivas, fisicamente árduas, de risco e/ou que exigem alta precisão. Segundo Grassi, 2005, a utilização de sistemas periféricos, como, por exemplo, o sistema de visão computacional, associados à atuação do manipulador, melhora o desempenho em determinadas tarefas, dentre as quais pode-se citar:

- Carregamento (manipulação de peças): o sistema de visão pode fornecer coordenadas que variam em relação ao robô, ou seja, fornecer ao robô a localização de uma peça que se aproxima, carregada por uma esteira móvel, por exemplo;
- Seleção e inspeção: o sistema de visão pode auxiliar na verificação se peças manipuladas estão dentro do padrão desejado, o qual pode ser definido por meio de dimensões, formatos, cores, etc.
- Aplicações médicas: o sistema de visão pode proporcionar a realização de cirurgias a distância e diagnósticos de doenças na realização de exames;
- Operações em ambientes perigosos: o sistema de visão aplicado à robótica dispensa cuidados em relação à saúde, à utilização de trajes especiais e sistemas de proteção quando substituem trabalhadores humanos;
- Operações espaciais, submarinas e em ambientes remotos: o sistema de visão aplicado à robótica pode ser melhor adaptado a ambientes espaciais e submarinos do que seres humanos.

Além dessas aplicações bem consolidadas da visão computacional aplicada à robótica, diferentes estudos podem ser encontrados em respeito à utilização de sistemas de visão para o

cálculo de trajetórias de robôs. No caso de robôs autônomos, Rakprayoon et. al., 2011, apresentam algumas técnicas de visão bidimensional e tridimensional utilizadas para a detecção de obstáculos ao longo da linha de movimento. Ao invés de utilizar múltiplas câmeras para a obtenção da profundidade da cena, utilizam um dispositivo chamado *Kinect*[2] que, devidamente calibrado, permite a correlação da profundidade da imagem com as coordenadas tridimensionais, a fim de detectar obstáculos. A calibração é realizada com um padrão (tabuleiro de xadrez) o que dispensa a necessidade de se conhecer os parâmetros intrínsecos da câmera. Rakprayoon et. al., 2011, concluem que apesar do dispositivo *Kinect* ser mais oneroso que a câmera convencional, produz menos ruído e é mais versátil, porém não resolve o problema de oclusões. Para isso, seria necessário a utilização de múltiplas câmeras de profundidade.

Koch et. al., 2013, propõem um algoritmo de processamento de imagem para ser utilizado para auxiliar um robô industrial em tarefas de seguimento de contornos em peças. Enquanto o sensor de visão dirige o robô ao longo da peça, um sistema de controle em malha fechada mantém a força desejada de contato do robô com a peça. A identificação dos contornos combina identificação da trajetória do contorno com medição de força e de aceleração para realização do controle. Para identificação do contorno, um algoritmo de seguimento de contornos é utilizado, constituído de: aquisição de dados, triagem, filtragem, predição e detecção de cantos. O sistema é validado por experimentos que envolvem seguimento de contornos em objetos conformes. Os experimentos mostram que a aplicação de medição da aceleração, direta na ferramenta, melhora a compensação de forças de inércia. O erro ao longo do contorno pode ser reduzido pelo algoritmo que detecta cantos. Como erros maiores ocorrem com maior velocidade na gama de altas curvaturas no caminho, esse problema foi resolvido com a redução automática da velocidade nestas áreas. Além disso, através da aplicação de sensores de aceleração, pode-se melhorar o algoritmo de identificação que emite os parâmetros necessários, a fim de compensar as deformações da peça.

Moreno e Lopez, 2013, descrevem o desenvolvimento de um sistema de planejamento de trajetória que controla o movimento de um robô móvel telecomandado por meio de técnicas de visão de máquina e algoritmos difusos. É utilizada uma câmera digital de telefone celular para captura das imagens e, por meio de técnicas de processamento de imagem, os

[2] *Kinect* é uma câmera com sensor de profundidade acoplado.

obstáculos são segmentados e, utilizando um algoritmo *Fuzzy C-means*[3], os caminhos possíveis são delimitados. A comunicação entre o telefone celular e o sistema de controle central em um computador é feito sem fios via XBee[4], de modo que o robô se move localmente utilizando um microcontrolador. Usando o algoritmo desenvolvido é possível obter um trajeto de deslocamento direto, sem falhas, em que é possível fazer variar o número de obstáculos e onde o tempo de trajeto é minimizado. Embora o tempo de processamento aumenta com um número crescente de obstáculos na cena, o robô móvel ainda é capaz de se mover continuamente, se o número de obstáculos permanece inferior a 6. É possível aplicar este sistema em situações que envolvem objetos em movimento dentro da cena. No entanto, é necessário estimar a distância entre objetos, a fim de determinar se há espaço suficiente para o robô móvel se movimentar através do percurso escolhido.

Carducci et. al., 2004, descrevem um braço robótico pneumático fixo controlado por válvulas liga/desliga para manipulação automática de peças e partes com base na localização de objetos por visão. Nesse caso, a câmera está acoplada na extremidade do manipulador e a imagem é capturada quando o braço manipulador está em uma determinada posição fixa. O reconhecimento do objeto é realizado por técnicas de detecção de bordas e a determinação do centroide pelo cálculo do valor médio dos pixels encontrados. Os resultados experimentais são positivos e o braço robótico acionado pelo sistema de visão é capaz de atingir o objeto alvo, utilizando a imagem e o controle de posição. Novos desenvolvimentos sobre o controle desse braço robótico versam sobre a possibilidade de adquirir sinais a partir da imagem por duas *webcams*[5] convencionadas, de tal forma a localizar o objeto para captá-lo não só em posições pré-fixadas mas também em todas as posições dentro do plano de trabalho.

Keshmiri et. al., 2010, apresentam um método de planejamento de trajetórias para objetos em movimento a serem capturados por um manipulador robótico de 5GL. Um sistema de aquisição de dados baseados em visão é usado para adquirir as posições e velocidades do objeto. Neste caso, a esteira não precisa estar parada para que o objeto seja capturado, já que o algoritmo prevê a posição futura do objeto, baseado nas suas duas últimas posições na imagem e na velocidade da esteira. Com a posição do objeto na primeira imagem em um tempo t_0 juntamente com a posição do objeto em um tempo t_1, é possível calcular a

[3] O *Fuzzy C-means* é um tipo de algoritmo baseado em agrupamento difuso utilizado em segmentação de imagens.
[4] *Xbee* é uma tecnologia utilizada para comunicação sem fios entre dispositivos.
[5] *Webcam* é uma câmera de vídeo que capta imagens e as transfere para um computador.

velocidade do objeto. Desta forma, é possível prever onde este objeto estará em um instante t_2. Porém, os objetos devem ter todos a mesma dimensão, pois não existe uma identificação tridimensional das dimensões dos objetos. Além disso, neste trabalho não é mencionado o reconhecimento da orientação do objeto.

Keshmiri e Wen-Fang, 2014, descrevem um método de planejamento de trajetórias otimizado baseado em servovisão para um robô de 4GL. A câmera é posicionada no efetuador do manipulador e as imagens são capturadas quando este está localizado em uma posição fixa. Neste método, a velocidade de cada junta é parametrizada usando-se perfis de polinômios de segunda ordem, onde um perfil de velocidade geral é concebido para cada campo de profundidade da câmara. Os parâmetros dos perfis de velocidades são determinados para guiar o robô para a posição desejada, minimizando o erro através de um algoritmo de otimização. Esta técnica mantém o robô em seus limites físicos e mantém as características do campo de visão.

Marin et. al., 2011, apresentam um método para segmentação baseado no cálculo de momentos invariantes e em tons de cinza da imagem. Com ele é possível detectar os vasos sanguíneos em imagens digitais da retina do olho humano. A sua eficácia e robustez em diferentes condições de imagem, juntamente com a sua simplicidade e rápida implementação, faz dessa proposta de segmentação de vaso sanguíneo adequada para análises computacionais de imagens da retina, tais como triagem automatizada para a detecção precoce da retinopatia diabética.

Zaeri et. al., 2015, apresentam um método de reconhecimento facial baseado no cálculo de momentos invariantes de imagens captadas por sensores de raios infravermelhos. O uso de imagens de infravermelho termal pode melhorar o desempenho de reconhecimento de face em condições de iluminação não controlados. Neste artigo, Zaeri et. al, 2015, apresentam uma nova técnica de reconhecimento de rosto com base em cálculos estatísticos de imagens térmicas, propondo o uso de momentos invariantes para definir um dos descritores de forma. O vetor de características proposto consiste em 11 momentos diferentes, onde três deles são momentos geométricos e os oito restantes são momentos geométricos centrais que oferecem robustez contra variabilidade (mudanças em regiões localizadas dos rostos). O novo método é testado em um novo banco de dados composto de imagens de diferentes expressões, diferentes iluminações, e captadas em diferentes intervalos de tempo. O trabalho experimental mostra como resultados uma taxa de reconhecimento de 97,5% e propõe como trabalho

futuro, considerar a implementação do método em diferentes poses e outras bases de dados de referência.

Li et. al., 2015, descrevem um método de reconhecimento de aeronaves pousadas com base nos modelos de PCNN[6] e cálculo de momentos invariantes afins. O método utiliza o modelo PCNN para gerar uma sequência binária de imagens em tons de cinza e, em seguida, ele calcula os momentos invariantes afins em sequência para compor o vetor de parâmetros. Os resultados experimentais mostram que o método além de ter a capacidade de resolver problemas de distorções, também pode permite programação simples e redução de etapa.

Hussein e Nordin, 2015, apresentam um sistema de reconhecimento da impressão de palmas[7] utilizando um algoritmo de momentos invariantes baseado em Transformada *Wavelet*[8]. O sistema proposto mostra resultados promissores, invariantes à rotação, translação e escala dos objetos, já que está associado com o uso de uma boa descrição das características da forma. O sistema é testado usando bancos de dados da Academia Chinesa de Ciências (CASIA), em Pequim. O experimento mostra a taxa de identificação de 98% na base de dados CASIA.

Grassi, 2005 descreve um sistema de visão aplicado a um robô industrial para realizar tarefas de manipulação de peças. O sistema identifica a posição e orientação de peças através do cálculo dos momentos da imagem capturada por uma câmera. A altura das peças não é identificada, tendo que ser informada pelo operador ao sistema. A identificação e diferenciação de mais de um objeto na cena captada também não são possíveis. O sistema é aplicado em um robô comercial ABB IRB 1400 de arquitetura fechada e uma rotina pronta de geração de trajetórias que pode receber como entrada a posição *x, y* e a orientação plana do objeto para captura do mesmo. O ponto de descarga desse objeto deve ser definido pelo usuário através do posicionamento manual do manipulador do robô ou pode ser adotado um ponto de descarga padrão.

Missiaggia, 2014, apresenta uma estratégia para a geração de trajetórias otimizadas para um robô cilíndrico de 5GL acionado pneumaticamente, baseada na utilização de um algoritmo de aproximação de pontos através de *splines* compostas por polinômios de sétimo

---

[6] PCNN (*Pulse Coupled Neural Network*) ou Rede Neural de Pulso Acoplado desenvolvida para processamento de imagem biomimética de alto desempenho.

[7] Impressão de palma refere-se a uma imagem adquirida da região da palma da mão. Ela difere de uma impressão digital na medida que contém mais informações, tais como textura, travessões e marcas, o que auxilia na diferenciação entre palmas de pessoas diferentes.

[8] A transformada *wavelet* é uma ferramenta que permite decompor um sinal em diferentes componentes de frequências, permitindo assim, estudar cada componente separadamente em sua escala correspondente.

grau. O sistema deve receber um conjunto de pontos chave em coordenadas *x, y e z* e ângulos dos dois últimos graus de liberdade por onde o efetuador final deve passar. Em seguida é gerada a trajetória através das *splines* que devem percorrer esses pontos chave e são gerados vetores das coordenadas no espaço das juntas incluindo cada ponto a ser percorrido.

Sarmanho, 2014, propõe uma técnica de controle não linear com compensação de atrito aplicada aos cinco graus de liberdade do mesmo robô objeto de estudo de Missiaggia, 2014. Com base em uma trajetória pré-definida, o sistema recebe os vetores com as coordenadas no espaço de juntas de cada ponto a ser percorrido, realiza o controle com compensação de atrito e gera novos vetores a serem executados pelo robô. O hardware utilizado para o controle central é uma placa dSPACE®, modelo DS-1104, que constitui um sistema de prototipagem para algoritmos de controle que se comunica diretamente com o programa Matlab® e a ferramenta de simulação e desenvolvimento de interfaces gráficas Simulink®.

Com base nos estudos anteriormente citados, visa-se, no âmbito do presente trabalho, desenvolver para esse robô cilíndrico de 5 graus de liberdade, objeto de estudo de Missiaggia, 2014, e Sarmanho, 2014, um sistema de visão computacional que identifique as peças a serem manipuladas (posição e ângulo de giro em relação ao sistema de coordenadas cartesianas do robô), informando ao algoritmo de planejamento de trajetórias do robô os valores de coordenadas necessários para gerar a trajetória a ser seguida pelo robô, de forma que este possa pegar a peça em uma determinada posição e deslocá-la até outra posição pré-determinada. Nesse contexto, é necessário integrar o sistema de planejamento de trajetórias [Missiaggia, 2014] com o sistema de controle do robô [Sarmanho, 2014] para viabilizar a realização dos testes experimentais. Para a identificação da posição e ângulo de giro da peça, propõe-se o uso do método apresentado por Grassi, 2005, baseado no cálculo dos momentos da imagem e comparação com uma imagem padrão. Além disso, é também necessário o desenvolvimento de um método de calibração das câmeras que possibilite a correlação de coordenadas de pontos no espaço tridimensional com as correspondentes coordenadas desses pontos na imagem, levando em consideração a geometria e o espaço de trabalho do robô. A fim de integrar o sistema de visão com o sistema de planejamento de trajetórias, propõe-se a utilização de um sistema de geração de pontos chave entre o ponto inicial do robô e o ponto final (determinado pelo sistema de visão). Para a realização de testes experimentais da técnica desenvolvida, foi proposta a implantação de uma rotina que corresponde ao movimento de

uma trajetória *pick-and-place* [9] no espaço tridimensional (semelhante à função JUMP descrita em EPSON, 2012), a qual consiste de uma composição de duas rotinas de movimentação vertical com duas de movimentação circular (arcos) e uma de movimentação horizontal. O desenvolvimento dessa função está detalhado na Seção 3.4.

2.2 Fundamentação teórica

Nesta seção é apresentada uma visão geral sobre as técnicas de tratamento de imagens na área da Visão Computacional, sendo destacada a técnica de detecção da posição e orientação de peças através do cálculo dos momentos da imagem. Por fim, são apresentadas técnicas de mapeamento de pontos em *coordenadas mundo*[10] na imagem bidimensional (com coordenadas em pixels).

2.2.1 Visão Computacional

Segundo Shapiro e Stockman, 2001, o principal objetivo da visão computacional é permitir a tomada de decisões úteis sobre cenas e objetos físicos reais com base em imagens de sensoriamento. Para isso, é primeiramente preciso segmentar a imagem, ou seja, identificar os objetos ou as cenas presentes na imagem. De acordo com Gonzalez e Woods, 2007, tal segmentação é realizada com base nos atributos e características da imagem. Tipicamente, faz-se uso das características de descontinuidade e similaridade de regiões, a qual pode ser medida, por exemplo, por meio das cores, formas e tamanhos dos objetos. Portanto, é preciso definir os descritores ou modelos a serem utilizados para compará-los com os objetos ou cenas da imagem, a fim de identificar os mesmos. Existem técnicas de segmentação de imagens que separam o objeto do fundo da imagem. Tais técnicas podem identificar cor, área, posição, ângulo de giro do objeto, entre outros.

A visão computacional, também conhecida como *visão robótica* ou *visão de máquina*, pode ser também definida como o processo de extrair, caracterizar e interpretar informações de imagens de um modo tridimensional. Segundo Fu et al, 1987, tal processo é dividido em seis subprocessos: (1) sensoriamento; (2) pré-processamento; (3) segmentação; (4) descrição;

[9] *Pick-and-place* é um termo utilizado na área da robótica que faz referência à um movimento de pegar um objeto em uma determinada posição, transportá-lo e depositá-lo em outra posição.
[10] coordenadas mundo é um termo utilizado em Visão Computacional, para referir-se às coordenadas cartesianas de pontos no espaço

(5) reconhecimento; e (6) interpretação. O *sensoriamento* é o processo de captar a cena com o uso de um sensor e gerar a imagem em formato digital. O *pré-processamento* é o conjunto de técnicas usadas para preparar a imagem para ser utilizada na aplicação prevista, constituindo-se em tratamentos da imagem como, por exemplo, redução de ruídos e realce de detalhes. A *segmentação* permite dividir a imagem em objetos de interesse. A *descrição* consiste da formação de parâmetros descritores que diferenciam objetos. O *reconhecimento* é a identificação de cada objeto. Por fim, a *interpretação* consiste no conjunto de técnicas que fornece um significado para os objetos, ou seja, os associa a algum dado de interesse no contexto.

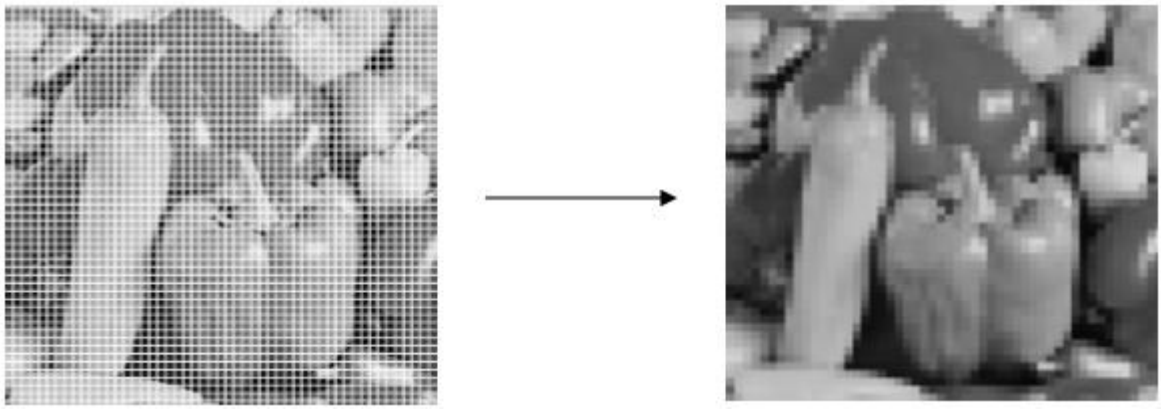
2.2.1.1 Captação da imagem

Uma imagem é a representação de uma forma de modo impresso ou digital, sendo a captação da imagem usualmente feita por uma câmera [Jung, 2013]. Segundo Szeliski, 2010, a imagem digital é formada da seguinte maneira: alguma forma de energia emitida a partir de uma fonte é refletida pelos objetos e é capturada por um sensor. A fonte de energia não precisa ser necessariamente a luz visível, pode ser, por exemplo, raio infravermelho, raio X ou ondas ultravioletas. Conforme apresentam Gonzalez e Woods, 2007, em geral, o sensor tem como entrada a energia provinda da fonte e gera como saída um sinal elétrico proporcional à entrada. Um sensor digital é geralmente composto de vários pequenos sensores relacionados individualmente com um pixel da imagem. A saída de cada um desses sensores é um valor real positivo que é quantificado e armazenado, formando a imagem digital.

Segundo Souza e Cardoza, 2009, existem dois tipos principais de sensores (ou conjunto de sensores): o CCD (*Charge-Coupled Device*) e o CMOS (*Complementary Metal-Oxide-Semiconductor*). O CCD tem a capacidade de transportar a carga elétrica ao longo do circuito com baixa distorção. Isto faz com que tenha elevada qualidade em termos de fidelidade e sensibilidade à luz, ou seja, baixo nível de ruído. São, porém, mais onerosos e consomem cerca de 100 vezes mais energia em comparação aos do tipo CMOS.

Gonzalez e Woods, 2007, descrevem como uma imagem digital é formada a partir dos dados de sensores: é necessário converter o sinal contínuo do sensor para a forma digital, o que envolve os conceitos de amostragem (*sampling*) e quantização (*quantization*). A amostragem é a discretização de um sinal contínuo no domínio espacial, enquanto que a

quantização é a discretização desse sinal em aplitude. A amostragem cria a matriz referente à imagem (define as dimensões da matriz) e, quanto maior o número de amostras, melhor é a resolução da imagem digitalizada. Na Figura 2.1 é apresentado um exemplo de uma imagem digitalizada com um número de amostras de aproximadamente 50 x 50.

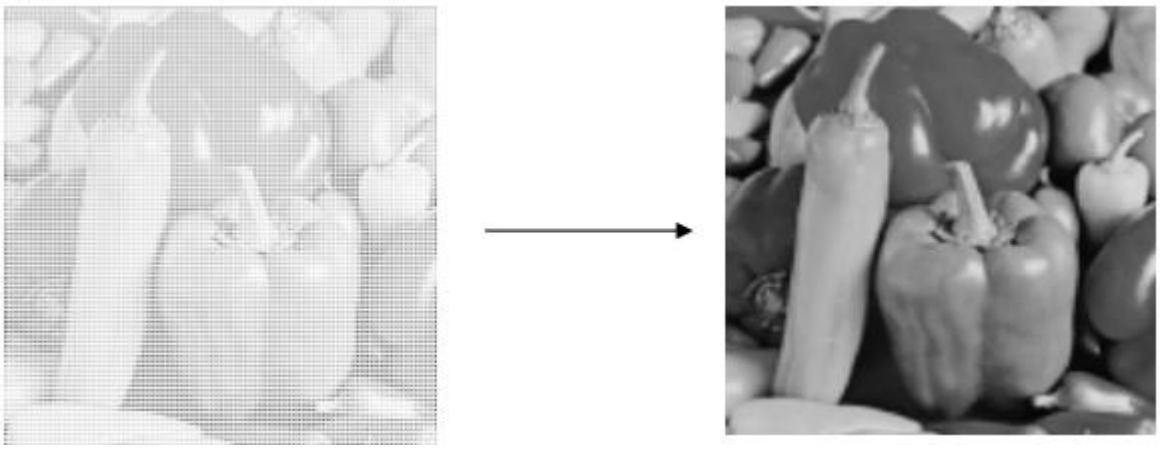


(a) imagem original com a grade de amostragem (b) resultado da imagem digitalizada

Figura 2.1 - Exemplo de imagem digitalizada com ≅ $\boldsymbol{50\ x\ 50}$ amostras

Fonte: JUNG, 2013.

Na Figura 2.2 é apresentada a imagem anterior, porém com número de amostras de aproximadamente 125 x 125. Percebe-se que a imagem da Figura 2.2 (b) tem uma resolução melhor que a imagem da Figura 2.1 (b).



(a) imagem original com a grade de amostragem (b) resultado da imagem digitalizada

Figura 2.2 - Exemplo de imagem digitalizada com ≅ $\boldsymbol{125\ x\ 125}$ amostras

Fonte: JUNG, 2013.

A quantização define a resolução de cor da imagem. Na Figura 2.3 (a) é apresentada uma imagem quantizada em 256 tons de cinza e na Figura 2.3 (b) é apresentada a mesma imagem quantizada em 4 tons de cinza.

(a) 256 tons de cinza (b) 4 tons de cinza

Figura 2.3 - Exemplo de imagens quantizadas

Fonte: JUNG, 2013.

2.2.1.2 Tratamento de imagem

Para se processar numericamente uma imagem é preciso que a mesma esteja no formato digital, dividida em pequenas seções chamadas células ou *pixels*, cuja quantidade depende da resolução da câmera que gerou a imagem. O tamanho de todos os pixels é geralmente igual, de forma que a imagem possa ser representada como uma matriz numérica de dimensões correspondente à quantidade de pixels. Os valores de cada elemento da matriz $\boldsymbol{M}(u, v)$ expressam a intensidade de cor do respectivo pixel na imagem [Shapiro e Stockman, 2001]. Esses valores dependem do espaço de cor em que a imagem está sendo tratada. Segundo Foley et. al, 1990, um espaço de cores é um sistema tridimensional de coordenadas, onde cada eixo refere-se a uma cor primária. A composição de cores primárias necessária para reproduzir uma determinada cor, é realizada através de valores referenciados. Segundo Palmer, 1999, uma cor é uma propriedade psicológica das nossas experiências visuais quando olha-se para objetos e luzes, não sendo uma propriedade física dos objetos ou luzes. Seguindo esse conceito, a cor é o resultado da interação entre a luz física no ambiente e o sistema

visual; assim, tudo o que é visto é o resultado de reflexões e absorções de ondas luminosas de diferentes frequências.

Desta forma, pode-se definir a cor como uma sensação provocada pela ação da luz sobre o observador por comprimentos de onda da luz produzida por uma fonte luminosa e modificada pelo objeto. Seu aparecimento está, portanto, condicionado à presença de três elementos: *fonte de luz*, *objeto* e *observador* [Melchiades; Boschi, 1999].

O sistema visual humano converte um espectro de luz em cor. Quando a luz chega à retina, atinge células fotossensíveis: cones e bastonetes [Gonzalez; Woods, 2007]. Existem três tipos de cones e cada um deles é sensível a uma região diferente do espectro:

- Curto - corresponde à região do azul;
- Médio – corresponde à região do verde;
- Longo – corresponde à região do vermelho.

Juntamente com os bastonetes, esses cones atuam como filtros do espectro, produzindo, cada um, um valor discreto que representa a quantidade absorvida. Assim, pela mistura dessas cores, chamadas de primárias aditivas, em diferentes combinações e níveis variados de intensidade, o total de cores presentes da natureza pode ser simulado de forma muito próxima [Holdschip, 2008]. Dessa forma, cada cor pode ser representada por três valores associados à quantidade de vermelho, verde e azul que foram filtrados e que corresponde a uma dada cor. Essa funcionalidade é chamada tricromaticidade humana [Fraser et. al, 2005].

O sistema de representação de cor mais usado é o RGB (*Red, Green, Blue*), onde cada frequência sensível do espectro é representada por três números que modulam uma combinação de cores, sendo o primeiro valor correspondente à quantidade filtrada de tons em vermelho (*Red*), o segundo em tons de verde (*Green*) e o terceiro em tons de azul (*Blue*). A soma dessas três cores, duas a duas, resulta nas cores ciano (verde + azul), magenta (vermelho + azul) e amarelo (vermelho + verde). Essas são as cores subtrativas, assim denominadas pelo fato de que as cores aditivas são obtidas através destas, pela subtração da cor branca por um conjunto de duas cores subtrativas. Assim, têm-se: vermelho = branco – amarelo – magenta, verde = branco − ciano − amarelo e azul = branco – magenta – ciano [Souto, 2006].

Na Figura 2.4, estão representadas as cores subtrativas, resultantes da soma entre as cores dos círculos correspondentes.

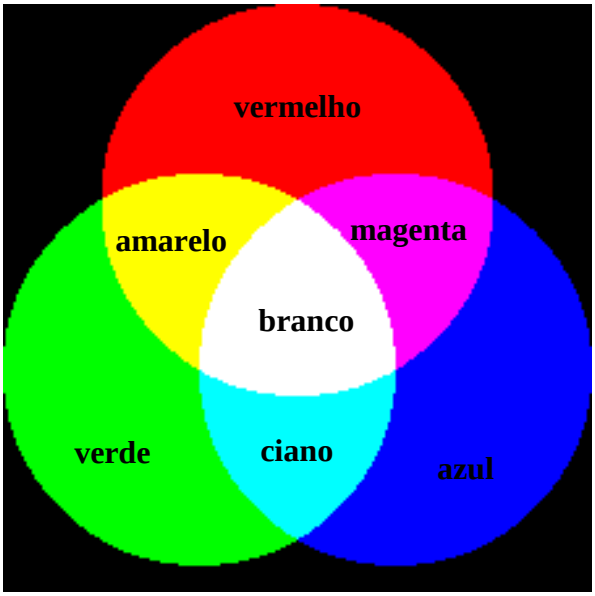


Figura 2.4 - Círculos representando as cores primárias do sistema aditivo.
Fonte: adaptado de CEPRSM, 2014

Pode-se também interpretar que o olho humano percebe cores como componentes de *matiz*, *saturação* e *brilho*. Matiz ou tonalidade é o comprimento de onda dominante de uma cor e representa intuitivamente a cor propriamente dita (verde, amarelo, azul, etc.). A saturação é o grau de pureza de uma cor, ou seja, o quanto ela é misturada com o branco (quanto menos branco ela contiver, mais saturada ela é. O brilho (ou luminância) é a medida de intensidade de luz de uma cor e está ligada ao conceito de percentual de intensidade da iluminação.

Segundo Gonzalez e Woods, 2007, a imagem pode ser representada digitalmente como uma matriz, onde cada posição ***M****(u,v)* corresponde a um pixel diferente, representado por um vetor 3 x 1, que corresponde à intensidade de *Red*, *Green* e *Blue* daquele determinado pixel. Em um sistema RGB com 8 bits, cada cor pode assumir valores de 0 a 255, ou seja, 256 valores. Uma imagem pode ser também representada em tons de cinza, através de uma matriz bidimensional. Desta forma, cada posição da matriz ***M****(u,v)* corresponde a um pixel diferente, representado por um vetor 1 x 1 de 8 bits associado a um tom de cinza correspondente. Este tom de cinza é obtido eliminando-se as informações de saturação e de matiz da cor, mantendo a luminância (densidade da intensidade de uma luz refletida numa dada direção). Desta forma,

como já mencionado, o tom de cinza correspondente a uma determinada cor dependendo apenas do seu valor de luminância.

Outra forma de representar uma imagem é em preto e branco. Segundo Shapiro, 2001, para isso, é preciso determinar um valor de limiar onde todos os valores de tom de cinza que estiverem acima deste serão determinados como branco (ou valor 0 binário) e todos os valores que estiverem abaixo deste limiar serão interpretados como preto (ou valor 1 binário), resultando em uma representação da imagem por uma matriz binária. Esse processo chama-se *limiarização da imagem*.

A limiarização é um processo simples que permite segmentar uma imagem. Ela pode igualmente dividir a imagem em diversos níveis conforme necessário, bastando somente selecionar diversos valores de limiares. A rotina mais comum e mais utilizada é a que separa o objeto do seu fundo, ou seja, objeto claro em fundo escuro ou objeto escuro em fundo claro, conforme pode ser visto na Figura 2.5 [Grassi, 2005].

Figura 2.5 - Processo de limiarização de uma imagem em tons de cinza.
Fonte: Grassi, 2005

Segundo Gonzalez e Woods, 2007, o limiar pode ser estabelecido através da análise visual do histograma da imagem, selecionando-se o valor da amplitude do vale entre as regiões. Geralmente, em sistemas de 8 bits, o valor limiar padrão é de 127,5, já que o tom de cinza pode variar de 0 a 255 (e 127,5 é a metade deste valor). Contudo, este valor pode variar consideravelmente (ora 17, ora 127, por exemplo) se a iluminação ambiente variar, já que uma mudança na iluminação pode fazer aumentar ou diminuir a diferença entre a intensidade média dos pixels do objeto e a intensidade média dos pixels do plano de fundo.

Otsu, 1979, apresenta uma forma de estabelecer automaticamente o valor de limiar, buscando minimizar a variância intraclasse (ou maximizar a variância interclasses) entre os tons de cinza dos dois objetos de interesse (peça e plano de fundo). Segundo Shrout e Fleiss,

1979, o coeficiente de correlação intraclasse (ICC) é uma das ferramentas estatísticas mais utilizadas para a mensuração da confiabilidade de medidas. O ICC é adequado para mensurar a homogeneidade de duas ou mais medidas e é interpretado como a proporção da variabilidade total atribuída ao objeto medido. Segundo Laureano, 2011, o ICC é calculado através de uma razão de variâncias, as quais podem ser obtidas a partir de diferentes métodos de estimação.

A ideia do Método de Otsu, 1979, é aproximar o histograma de uma imagem por duas funções Gaussianas e escolher o limiar de forma a minimizar a variância intraclasses, sendo que cada classe possui suas próprias características, ou seja, sua média e desvio padrão.

Segundo Gonzalez e Woods, 2007, dada uma imagem digital *Im*, de dimensões n_u x n_v e quantizada em *L* níveis de cinza, calcula-se o histograma da imagem através da Equação (2.1):

$$h_i = \frac{n_i}{n_u n_v} \tag{2.1}$$

onde n_i é a quantidade de pixels da imagem (*Im*) que possuem a intensidade de cinza *i*, para *i=0, ..., L-1*. Assim, $n_u n_v = n_0 + n_1 + \cdots + n_{L-1}$ e

$$\sum_{i=0}^{L-1} h_i = 1 \tag{2.2}$$

para todo $h_i \geq 0$.

Conforme mencionado anteriormente, o valor da amplitude do vale entre as regiões de um histograma é o limiar ideal. Desta forma, segundo Gonzalez e Woods, 2007, sendo *k* o nível de cinza que reparte o histograma da imagem em duas classes C_1 e C_2, em que a primeira e a segunda classe compreendem os pixels cujos níveis de cinza pertencem ao intervalo [*0,k*] e [*k+1,L-1*], respectivamente, calcula-se as probabilidades:

$$Pb_1(k) = \sum_{i=0}^{k} h_i \tag{2.3}$$

$$Pb_2(k) = \sum_{i=k+1}^{L-1} h_i = 1 - Pb_1(k) \tag{2.4}$$

sendo, $Pb_1(k)$ a probabilidade do nível de cinza *k* ser da classe C_1 e $Pb_2(k)$ a probabilidade do nível de cinza *k* ser da classe C_2.

Gonzalez e Woods, 2007, apresentam que a intensidade média do valor dos pixels pertencentes à classe C_1 é dada por:

$$itm_1(k) = \sum_{i=0}^{k} iPb(i/C_1) \tag{2.5}$$

e, utilizando a regra de *Bayes*[11], têm-se:

$$itm_1(k) = \sum_{i=0}^{k} i\frac{Pb\left(\frac{C_1}{i}\right)Pb(i)}{Pb(C_1)} \tag{2.6}$$

e, como $Pb(C_1) = Pb_1(k),\ Pb(i)$ é o próprioh_i e $Pb(C_1/i)$ é sempre 1, uma vez que i está no intervalo da própria classe C_1. Desta forma, calcula-se a intensidade média do valor dos pixels pertencentes à classe C_1 por:

$$itm_1(k) = \frac{1}{Pb_1(k)}\sum_{i=0}^{k} ih_i \tag{2.7}$$

e, similarmente, a intensidade média do valor dos pixels pertencentes à classe C_2 é dada por:

$$itm_2(k) = \frac{1}{Pb_2(k)}\sum_{i=k+1}^{L-1} ih_i \tag{2.8}$$

Assim, a variância para cada distribuição de probabilidade por ser determinada por:

$$\sigma_1^2(k) = \frac{1}{Pb_1(k)}\sum_{i=0}^{k}(itm_1(k) - h_i)^2 \tag{2.9}$$

$$\sigma_2^2(k) = \frac{1}{Pb_2(k)}\sum_{i=k+1}^{L-1}(itm_2(k) - h_i)^2\ . \tag{2.10}$$

[11] A regra ou Teorema de Bayes mostra a relação entre uma probabilidade condicional e a sua inversa; por exemplo, a probabilidade de uma hipótese dada a observação de uma evidência e a probabilidade da evidência dada pela hipótese.

Desta forma, a variância intraclasses ($\sigma_B^2(k)$) em relação ao nível de cinza *k* pode ser determinada pela Equação (2.11):

$$\sigma_B^2(k) = \sigma_1^2(k)Pb_1(k) + \sigma_2^2(k)Pb_2(k) \tag{2.11}$$

Então, para todos os valores de *k*, determina-se o limiar ótimo (k^*) a partir da minimização da variância intraclasses, conforme apresentado na Equação (2.12):

$$k^* = \min_{0 \le k \le L-1} \sigma_B^2(k) \tag{2.12}$$

2.2.1.3 Momentos da imagem

Um dos problemas estudados na área da visão computacional é o de reconhecimento de objetos em uma cena. Uma das estratégias é baseada na determinação de padrões com os quais o objeto da imagem deve ser comparado para então reconhecê-lo. O cálculo dos momentos da imagem, método proposto por Hu, 1962, é amplamente utilizado para reconhecimento de padrões. Segundo Zhou e Kornerup, 1995, este método é o mais eficiente para a descrição de imagens. Hu, 1962, divide os cálculos dos momentos em momentos padrões ou regulares, momentos centrais e momentos invariantes. Um momento padrão ou regular é capaz de permitir o cálculo da área e do centroide de um objeto mesmo que este esteja transladado, mas, neste caso, não é capaz de identificar a orientação de um objeto. Enquanto que um momento central é capaz de permitir a identificação da orientação de um objeto mesmo que ele esteja transladado, ou seja, é invariante à translação. No caso dos momentos invariantes, como o próprio nome diz, são invariantes à translação, rotação e escala. São largamente utilizados no reconhecimento de padrões, ou seja, quando se deseja identificar e selecionar objetos em uma cena independente do seu tamanho, orientação e localização.

Segundo Hu, 1962, os momentos bidimensionais de ordem *(p,q)* de uma função de distribuição de densidade $\rho(x, y)$ são definidos em termos de integrais de Riemann como:

$$m_{pq} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x^p y^q \rho(x, y)\, dx\, dy \tag{2.13}$$

onde *p*,*q*=0,1,2.. Se for assumido que $\rho(x,y)$ é uma função contínua por partes, limitada, e que pode ter valores diferentes de zero apenas em uma parte finita no plano *xy*, então existem momentos de todas as ordens e o seguinte teorema de unicidade pode ser provado .

Teorema de unicidade: A sequência de momento duplo $\{m_{pq}\}$ é determinada unicamente por $\rho(x,y)$ e, inversamente, $\rho(x,y)$ é unicamente determinada por $\{m_{pq}\}$.

Existe um grande número de propriedades que representam a forma e a disposição de uma superfície plana em relação a um sistema referencial. Na área da mecânica, o conceito de momentos é largamente utilizado e uma analogia do cálculo de momento de inércia de superfícies pode ser feita com relação a uma imagem [Hu, 1962].

Para compreender os momentos da imagem é interessante representar os pixels pertencentes ao objeto de interesse como um conjunto de pontos em um plano cartesiano. Assim, cada pixel de interesse é representado por duas coordenadas, (u,v), as quais descrevem sua posição na imagem, e uma função $f(u,v)$ que fornece a intensidade (tom de cinza) do pixel em cada coordenada (u,v), conforme pode ser observado na Figura 2.6.

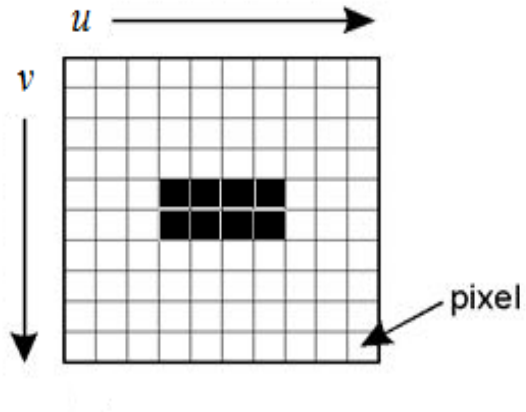


Figura 2.6 - Representação de uma imagem dividida em pixels com seu sistema de coordenadas (*u*,*v*)

Em uma imagem binarizada, muito comumente utilizada na extração de atributos das imagens por momentos, a função $f(u,v)$ possui valor 1 para todos os pixels que pertencem ao objeto de interesse, e valor 0 para os pixels de fundo, ou que não pertencem ao objeto de interesse [Souza e Pistorini, 2005].

Segundo Khotanzad e Hong, 1990, ao tratar uma imagem digital, as integrais podem ser substituídas por somatórios:

$$m_{pq} = \sum_1^{n_u} \sum_1^{n_v} u^p v^q \, f(u,v) \tag{2.14}$$

onde n_u e n_v representam, respectivamente, a largura e a altura da imagem digital. Como $f(u,v)$ é a intensidade do pixel e em uma imagem binarizada só existem intensidades 0 ou 1, quando a intensidade for 0, a multiplicação $u^{\mathrm{p}} v^{\mathrm{q}} f(u,v)$ será igual a zero. Assim, só serão somados os valores 1 da imagem multiplicados por suas coordenadas em u e em v. No caso do momento de ordem *(0,0)*, ou seja, $p = 0$ e $q = 0$, o momento m_{pq} corresponde à área do objeto na imagem, já que consiste no somatório de pixels de valor 1 da imagem. Desta forma, em geral, para uma imagem binária a área do objeto pode ser expressa por:

$$A = m_{00} = \sum_1^{n_u} \sum_1^{n_v} u^0 v^0 \, f(u,v) = \sum_1^{n_u} \sum_1^{n_v} f(u,v) \tag{2.15}$$

onde para calcular os momentos de ordens maiores basta elevar as coordenadas u e v aos índices correspondentes. Os momentos que usam todas as combinações de índices entre 0 e 3 são os mais comuns [Niku, 2001].

Normalmente, as distâncias em u e v são expressas por intermédio da posição do pixel na matriz binária que representa a imagem. A configuração geralmente utilizada em computação gráfica corresponde à situação em que os valores de u aumentam da esquerda para a direita e os valores de v aumentam de cima para baixo, sendo o pixel posicionado na primeira linha e na primeira coluna da matriz, o menor de valor de u e de v. Da mesma forma, o pixel posicionado na última linha e na última coluna da matriz binária é o de maior valor de u e de v [Jain et. al., 1995].

O momento de ordem $p = 1$ e $q = 0$ fornece um somatório de todas as coordenadas u do objeto (formado por valores 1 na imagem). Da mesma forma, o momento de ordem $p = 0$ e $q = 1$ fornece um somatório de todas as coordenadas v do objeto (formado por valores 1 na imagem). Assim, por exemplo, dividindo esse momento pela área da forma em estudo, obtém-se os pontos centrais do objeto $(\bar{u}\ e\ \bar{v})$, que correspondem ao chamado centroide:

$$\bar{u} = \frac{m_{10}}{m_{00}} \tag{2.16}$$

$$\bar{v} = \frac{m_{01}}{m_{00}} \tag{2.17}$$

onde m_{00} pode representar, por exemplo, a massa ou a área do objeto de interesse, já que soma todos os valores 1 da imagem representa o objeto, enquanto m_{10} e m_{01} representam as projeções dos pontos de interesse nos eixos x e y, respectivamente [Chavez e Lhiang, 2003].

Segundo Hu, 1962, os momentos apresentados anteriormente são os chamados momentos padrões ou regulares. Além dos momentos regulares, existem também os momentos centrais (μ_{pq}) os quais são definidos por meio da Equação (2.18):

$$\mu_{pq} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}(u-\bar{u})^p\,(v-\bar{v})^q\rho(u,v)d(u-\bar{u})\,d(v-v) \tag{2.18}$$

onde $\bar{u}$ e $\bar{v}$ são os centroides definidos pelas equações (2.16) e (2.17). Segundo [Hu, 1962], mesmo que haja translações de coordenadas (*u*,*v*) definidas por meio de:

$$u' = u + \alpha \tag{2.19}$$

$$v' = v + \beta \tag{2.20}$$

sendo α e β as constantes de translação. Os momentos centrais não mudam com a translação.

Os momentos μ_{02} (segundo momento de área relativo ao eixo u) e μ_{20} (segundo momento de área relativo ao eixo v) podem variar se o mesmo rotacionar em torno do seu centro de inércia, como no caso do momento de inércia (de massa ou de área) de um objeto assimétrico. Grassi, 2005, apresenta um sistema de reconhecimento de orientação de objetos assimétricos baseado nessa propriedade, por meio do cálculo dos momentos de área em relação ao eixo u (ou v) em diferentes orientações. De acordo com essa estratégia, para identificar a orientação do objeto deve-se montar uma tabela de comparação que contenha esses valores, pois cada orientação corresponde a um valor único para o momento μ_{20} (ou μ_{02}).

Os momentos regulares m_{02} e m_{20} são variantes à translação, ou seja, eles dependem da distância em relação aos eixos da imagem, o que se resolve calculando-os em relação a eixos auxiliares que têm origem no centroide do objeto, ou seja, calculando seus momentos centrais. Dessa forma, pode-se utilizar os momentos centrais μ_{02} e μ_{20} para identificar a orientação de um objeto, independentemente da sua localização. Para simplificar o processo, Grassi, 2005, sugere que ao invés de usar os dois valores de momento para fazer a comparação, se utilize a razão *R*, definida como:

$$R = \frac{\mu_{02}}{\mu_{20}} \tag{2.21}$$

e, como os valores de μ_{02} e μ_{20} repetem, não é possível determinar somente com seu uso o quadrante em que está o eixo principal do objeto, necessitando identificar se determinado valor calculado corresponde a um giro entre 0º e 90º ou entre 90º e 180º, por exemplo. Para realizar esta identificação pode-se utilizar os momentos de terceira ordem. Porém, como são elevados à terceira potência, eles podem amplificar também os ruídos do tipo *salt&pepper*[12]. O ruído deste tipo pode fazer com que alguns pontos localizados fora do objeto sejam identificados como parte dele. No caso do cálculo de momentos de ordem mais baixa, esses ruídos podem não prejudicar o resultado; porém, quando utilizados cálculos de momentos de ordem mais alta, esses ruídos podem gerar grandes diferenças, já que, neste caso, os valores são elevados a uma potência alta.

Uma alternativa proposta por Grassi, 2005, é a de utilizar o momento μ_{11}, o qual fornece o sinal do ângulo do objeto. Na Tabela 2.1 está apresentado um exemplo de tabela de comparação de uma imagem utilizada como padrão.

Observa-se que o sinal do momento μ_{11} varia a cada quadrante, ou seja, quando o sinal de μ_{11} é negativo, o ângulo informado está no 1º quadrante (0º a 90º) ou 3º quadrante (180º a 270º) e, quando o sinal de μ_{11} é positivo, o ângulo informado está no 2º quadrante (90º a 180º) ou 4º quadrante (270º a 360º).

Observa-se, também, que todos os valores do 1º quadrante se repetem no 3º e que todos os valores do 2º quadrante se repetem no 4º. Por isso, não há necessidade de se criar a tabela de comparação de 0º a 360º, fazendo-se necessário criá-la apenas de 0 a 180º.

Após montada a tabela de comparação padrão, basta comparar os resultados dos momentos μ_{02} e μ_{20} da imagem rotacionada com os valores da tabela para avaliar em que faixa de valores, dependendo da resolução (ou do número de valores da tabela), está o ângulo de giro da imagem.

[12] *salt&pepper* é um tipo de ruído que ocorre quando são identificados pontos pretos no fundo branco ou pontos brancos no fundo preto.

Tabela 2.1 - Exemplo de tabela de comparação de uma imagem

Ângulo (graus)	μ_{02} (10^3)	μ_{20} (10^3)	**R** (10^3)	μ_{11} (10^3)
0	1,0200	3,6720	0,0003	0
10	1,1370	3,6768	0,0003	-0,5073
20	1,3659	3,4555	0,0004	-0,8834
30	1,7100	3,0501	0,0006	-1,1439
40	2,1159	2,6779	0,0008	-1,3191
50	2,5168	2,1188	0,0012	-1,2792
60	3,0225	1,6620	0,0018	-1,1410
70	3,3138	1,3398	0,0025	-0,8568
80	3,5657	1,1104	0,0032	-0,4836
90	3,6720	1,0200	0,0036	0
100	3,6768	1,1370	0,0032	0,5073
110	3,4555	1,3659	0,0025	0,8834
120	3,0501	1,7100	0,0018	1,1439
130	2,6779	2,1159	0,0013	1,3191
140	2,1188	2,5168	0,0008	1,2792
150	1,6620	3,0225	0,0005	1,1410
160	1,3398	3,3138	0,0004	0,8568
170	1,1104	3,5657	0,0003	0,4836
180	1,0200	3,6720	0,0003	0
190	1,1370	3,6768	0,0003	-0,5073
200	1,3659	3,4555	0,0004	-0,8834
210	1,7100	3,0501	0,0006	-1,1439
220	2,1159	2,6779	0,0008	-1,3191
230	2,5168	2,1188	0,0012	-1,2792
240	3,0225	1,6620	0,0018	-1,1410
250	3,3138	1,3398	0,0025	-0,8568
260	3,5657	1,1104	0,0032	-0,4836
270	3,6720	1,0200	0,0036	0
280	3,6768	1,1370	0,0032	0,5073
290	3,4555	1,3659	0,0025	0,8834
300	3,0501	1,7100	0,0018	1,1439
310	2,6779	2,1159	0,0013	1,3191
320	2,1188	2,5168	0,0008	1,2792
330	1,6620	3,0225	0,0005	1,1410
340	1,3398	3,3138	0,0004	0,8568
350	1,1104	3,5657	0,0003	0,4836
360	1,0200	3,6720	0,0003	0

2.2.2 Robôs manipuladores industriais

De acordo com Rosário, 2005, os primeiros robôs surgiram com o intuito de reduzir riscos aos operários, substituindo-os em tarefas perigosas ou insalubres que envolviam condições desagradáveis, com altos níveis de calor, ruídos, gases tóxicos, esforço físico extremo ou trabalhos tediosos e monótonos. Posteriormente, os objetivos da utilização de robôs industriais foram ampliados incluindo a melhoria da qualidade e o aumento da velocidade de produção, e a consequente redução de custos. Por isso, percebe-se o forte crescimento da robótica industrial, conforme os dados da IFR apresentados no Capítulo 1.

Como o robô foi desenvolvido para substituir o homem em determinadas tarefas, suas partes podem geralmente ser relacionadas com as do corpo humano. No caso de um robô manipulador industrial, ele pode ser comparado ao braço humano já que possui elos (*links*) e articulações (juntas). Assim, de acordo com Craig, 2005, os manipuladores consistem de conjuntos de elos rígidos que são conectados por juntas que permitem movimentos relativos aos elos vizinhos. Estas juntas são usualmente instrumentadas com sensores de posição que permitem a medição da posição relativa dos elos vizinhos.

Segundo Craig, 2005, a junta pode ser de seis tipos: revolução ou rotacional, prismática, cilíndrica, planar, parafuso ou esférica. Cada articulação fornece um grau de liberdade para o robô, sendo que a maioria deles possui 5 ou 6 graus de liberdade. Contudo, neste trabalho, esta classificação é simplificada para junta prismática (P), a qual permite deslocamento linear (d) e junta rotacional (R), associada ao deslocamento angular (θ). Desta forma os robôs manipuladores podem ser classificados conforme sua configuração, sendo que maioria se enquadra nas categorias: cartesiano (PPP), SCARA (RRP), articulado (RRR), esférico (RRP) ou cilíndrico (RPP).

O primeiro grau de liberdade do robô é o que fornece movimento ao manipulador em relação à sua base. O último grau de liberdade é denominado como "elemento terminal", no qual será acoplado o componente efetuador, o qual pode ser uma garra ou uma ferramenta. A principal função da garra é pegar um objeto para que o robô transporte-o para determinada posição. Já a ferramenta realiza um trabalho sobre uma peça sem necessariamente manipulá-la, como, por exemplo, soldagem, pintura, polimento, entre outros. Segundo Craig, 2005, o número de graus de liberdade (GL) que um manipulador possui é o número de variáveis de

posição independentes que devem ser especificadas para localizar todas as partes do mecanismo.

A movimentação das partes do robô ocorre devido à transmissão de potência mecânica originada em um atuador. Os atuadores convertem energia elétrica, hidráulica ou pneumática em energia mecânica e é por meio dos sistemas de transmissão que a potência mecânica gerada pelos atuadores é transmitida aos elos para que se movimentem [Romano e Dutra, 2002]. Dependendo do tipo de energia utilizada, pode-se classificar os atuadores como hidráulicos, elétricos e pneumáticos. A escolha vai depender do tipo de aplicação e do que se deseja em relação a parâmetros a serem levados em consideração, tais como precisão, força, rapidez, ruído, custo ou segurança [Rijo, 2013].

2.2.3 Geometria da imagem

A técnica de reconhecimento da orientação de objetos baseado em seus momentos, apresentada na Seção 2.2.1.3, fornece as coordenadas do objeto em coordenadas de pixel da imagem. Contudo, é preciso relacionar este sistema de coordenadas com o sistema de coordenadas cartesianas do robô, ou seja, relacionar as coordenadas bidimensionais da imagem com o domínio tridimensional definido pelo volume de trabalho do robô. Para tanto, um algoritmo envolvendo as transformações necessárias deve ser disponibilizado. A seguir, na Seção 2.2.3.1, é apresentada uma proposta de estratégia para solução desse problema que será, posteriormente, aplicada ao robô pneumático no estudo de caso.

2.2.3.1 Transformações Geométricas

Para se compreender a projeção de uma cena tridimensional em uma imagem bidimensional é preciso analisar como se dão as transformações geométricas entre as coordenadas de duas figuras, de forma que, a partir de uma figura geométrica original, se forme outra geometricamente igual ou semelhante no mesmo plano ou em planos diferentes [Rodrigues e Rego, 2010].

As transformações geométricas mais aplicadas são *escala*, *rotação*, *translação*, *espelhamento* e *cisalhamento*. As transformações podem ser de um objeto em duas dimensões para uma imagem bidimensional ou de um objeto tridimensional para uma imagem projetada

no plano (bidimensional) [Gonzalez e Woods, 2007]. Nas seções 2.2.3.1.1 e 2.2.3.1.2 são descritas com detalhes as principais transformações de interesse do presente trabalho (translação e rotação, respectivamente).

Enquanto a translação é tratada como uma soma de vetores, a rotação é tratada como a multiplicação de um vetor por uma matriz. As transformações são geralmente não lineares. Assim, o processo de combinar transformações não é trivial. Uma alternativa para tratar as transformações de forma consistente é através da representação de cada ponto ($\boldsymbol{p_i}$) do espaço através de coordenadas homogêneas [Ballard e Brown, 1982].

Segundo esta formalização, um ponto no espaço tridimensional $\boldsymbol{p_w} = [x_w \quad y_w \quad z_w]^{\mathrm{T}}$ pode ser representado por meio de coordenadas homogêneas com um determinado peso (ω). Dessa forma, cada ponto é representado como:

$$\boldsymbol{p_w^h} = \begin{bmatrix} x_w\omega \\ y_w\omega \\ z_w\omega \\ \omega \end{bmatrix} \tag{2.22}$$

e, assim, dois ou mais pontos diferentes em coordenadas homogêneas podem representar o mesmo ponto no espaço tridimensional (mundo) quando um dos pontos é múltiplo do outro [Ballard e Brown, 1982]. Com isso, é possível usar a mesma expressão para determinar uma reta gerada por dois pontos ou por um ponto e um vetor, ou seja, a reta tem a mesma representação. De acordo com Craig, 2005, a representação em coordenadas homogêneas pode ser considerada simplesmente como uma construção usada para apresentar a rotação e translação sob a forma de uma matriz única. Em outros campos de estudo, ela pode ser usada, por exemplo, para calcular perspectiva e operações de dimensionamento.

2.2.3.1.1 Translação

Para realizar a translação de objetos, basta adicionar quantidades inteiras (dx, dy, dz) às suas coordenadas (x, y, z respectivamente). Assim, uma operação ($\boldsymbol{p}'$) de translação sobre um ponto $\boldsymbol{p}(x, y, z)$ é dada pela Equação (2.23) ou, na forma matricial, pela Equação (2.24):

$$T(\boldsymbol{p}) = T(x_p, y_p, z_p) = (x_p + dx, y_p + dy, z_p + dz) \tag{2.23}$$

$$\boldsymbol{p}' = \boldsymbol{p} + T(\boldsymbol{p}) = \begin{bmatrix} x_p \\ y_p \\ z_p \end{bmatrix} + \begin{bmatrix} dx \\ dy \\ dz \end{bmatrix} \tag{2.24}$$

cuja representação na forma homogênea com peso $\omega = 1$ é dada por:

$$\begin{bmatrix} x_p{}' \\ y_p{}' \\ z_p{}' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & dx \\ 0 & 1 & 0 & dy \\ 0 & 0 & 1 & dz \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{bmatrix} \tag{2.25}$$

2.2.3.1.2 Rotação

A rotação é uma transformação que faz a representação numérica de um objeto girar em torno de um de seus eixos $(x, y\ ou\ z)$. Por exemplo, um ponto $\boldsymbol{p}(x, y, z)$ rotacionado um determinado ângulo γ em torno do eixo z resulta no ponto $\boldsymbol{p}'(x', y', z')$, conforme apresentado na Figura 2.7.

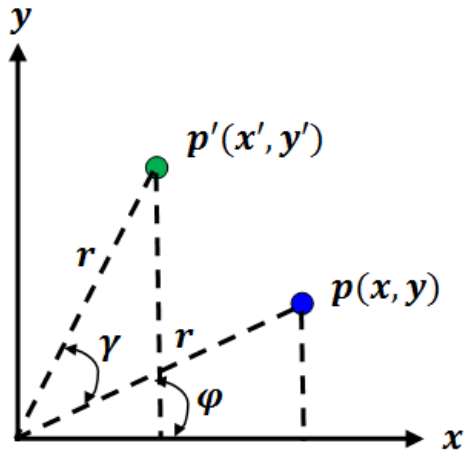


Figura 2.7 - Representação do ponto $\boldsymbol{p}$ rotacionado em relação ao eixo *z*

Pode-se, assim, relacionar os valores de x_p, y_p, z_p com os de x'_p, y'_p e z'_p através das seguintes expressões:

$$x_p = r\cos\varphi \tag{2.26}$$

$$y_p = r\,sen\varphi \tag{2.27}$$

$$x'_p = rcos(\varphi + \gamma) = rcos\varphi cos\gamma - rsen\varphi sen\gamma \tag{2.28}$$

$$y'_p = rs(\varphi + \gamma) = rsen\varphi cos\gamma + rsen\gamma cos\varphi \tag{2.29}$$

$$z'_p = z_p \tag{2.30}$$

substituindo (2.26) em (2.27) e (2.28) em (2.29) tem-se:

$$x'_p = x_p cos\gamma - y_p sen\gamma \tag{2.31}$$

$$y'_p = x_p sen\gamma + y_p cos\gamma \tag{2.32}$$

logo, a rotação de um ponto em relação ao eixo z, $\boldsymbol{Rz}(\boldsymbol{p}) = Rz(x_p, y_p, z_p)$, é dada por:

$$\boldsymbol{Rz}(\boldsymbol{p}) = (x_p cos\gamma - y_p sen\gamma, x_p sen\gamma + y_p cos\gamma, z_p) \tag{2.33}$$

cuja representação na forma homogênea é a seguinte:

$$\begin{bmatrix} x_p{}' \\ y_p{}' \\ z_p{}' \\ 1 \end{bmatrix} = \begin{bmatrix} cos\gamma & -sen\gamma & 0 & 0 \\ sen\gamma & cos\gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{bmatrix} \tag{2.34}$$

Da mesma maneira, pode-se deduzir a rotação em torno do eixo *x*. Desta forma, tendo um ponto no espaço $\boldsymbol{p}(x, y, z)$, ao ser efetuada uma operação de rotação deste ponto em relação ao eixo x, $\boldsymbol{Rx}(\boldsymbol{p}) = \boldsymbol{Rx}(x_p, y_p, z_p)$, obtém-se:

$$\boldsymbol{Rx}(\boldsymbol{p}) = (x_p, y_p cos\alpha - z_p sen\alpha, y_p sen\alpha + z_p cos\alpha) \tag{2.35}$$

cuja representação na forma homogênea é a seguinte:

$$\begin{bmatrix} x_p{}' \\ y_p{}' \\ z_p{}' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos\alpha & -sen\alpha & 0 \\ 0 & sen\alpha & cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{bmatrix} \tag{2.36}$$

Da mesma maneira deduz-se a rotação em torno do eixo *y* $\boldsymbol{Ry}(\boldsymbol{p})$. Resultando:

$$\boldsymbol{Ry}(\boldsymbol{p}) = \left(z_p sen\beta + x_p cos\beta, y_p, z_p cos\beta - x_p sen\beta\right) \tag{2.37}$$

cuja representação na forma homogênea expressa é como:

$$\begin{bmatrix} x_p{}' \\ y_p{}' \\ z_p{}' \\ 1 \end{bmatrix} = \begin{bmatrix} cos\beta & 0 & sen\beta & 0 \\ 0 & 1 & 0 & 0 \\ -sen\beta & 0 & cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{bmatrix} \tag{2.38}$$

2.2.3.2 Projeção Perspectiva

Para obter-se a equação de projeção que relaciona as coordenadas de uma imagem obtida com uma dada câmera com um sistema global externo de coordenadas, é preciso levar em conta parâmetros intrínsecos e extrínsecos da mesma. Os parâmetros intrínsecos são aqueles referentes à geometria e à configuração da câmera, como a distância focal (distância da lente até o plano da imagem (PI)) e o tamanho e forma dos pixels. Já, os parâmetros extrínsecos são relativos à localização da câmera no espaço (rotação e translação) [Gonzalez e Woods, 2007].

O modelo de câmera mais comum é o modelo *Pinhole*. Um exemplo simplificado de formação de imagem deste modelo pode ser visto na Figura 2.8. Neste modelo, o domínio geométrico da lente é assumido como um único ponto. De acordo com Shah, 1997, para este modelo de câmera, um ponto no espaço com coordenadas (X, Y, Z) é representado em coordenadas (x, y) sob projeção perspectiva.

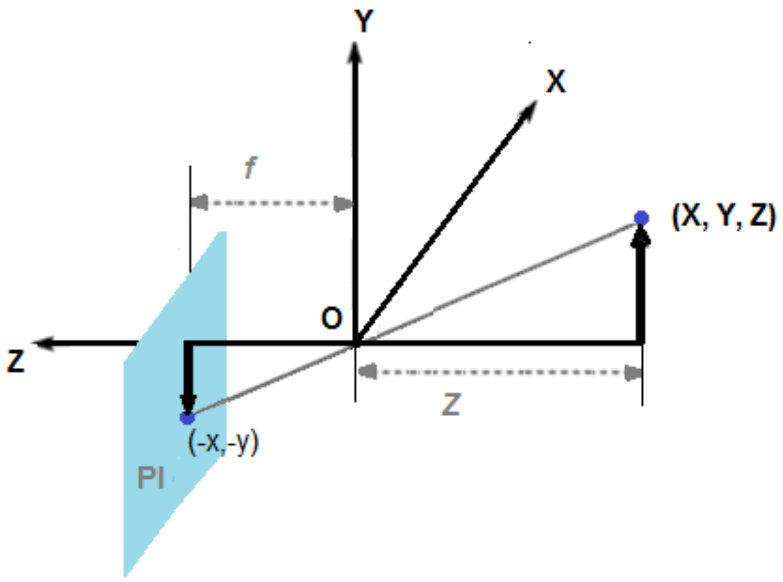


Figura 2.8 - Modelo *Pinhole* de câmera com origem na lente.
Fonte: adaptado de Shah, 1997.

Em operações robóticas de manipulação de peças baseada em visão computacional, é preciso que seja realizada a correlação das coordenadas do robô com as coordenadas da imagem. Porém, conforme já mencionado, existem parâmetros intrínsecos e extrínsecos à câmera que influenciam neste processo. Para se determinar alguns dos parâmetros intrínsecos, deve-se levar em consideração a distância focal além da existência de uma relação entre os pixels da imagem com a configuração da câmera.

2.2.3.3 Parâmetros intrínsecos de uma câmera

Assumindo que a origem do sistema de coordenadas (O) está no centro da lente (Figura 2.8) e que f é a distância focal da câmera. Usando os dois triângulos equivalentes, pode-se escrever:

$$\frac{-y}{Y} = \frac{f}{Z} \tag{2.39}$$

e que permite descrever as coordenadas (x,y) em função da distância focal e da distância do objeto até a lente:

$$y = -\frac{fY}{Z} \tag{2.40}$$

$$x = -\frac{fX}{Z} \tag{2.41}$$

assim, representando estas coordenadas em coordenadas homogêneas com peso 1, têm-se:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{-f}{Z} & 0 & 0 & 0 \\ 0 & \frac{-f}{Z} & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.42}$$

ou, adotando-se peso $\frac{-Z}{f}$ obtém-se:

$$\begin{bmatrix} x \\ y \\ \frac{-Z}{f} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{-1}{f} & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.43}$$

A Equação (2.43) é conhecida como Projeção Perspectiva (PP), a qual relaciona as coordenadas no espaço com as coordenadas da imagem quando a câmera está localizada na origem do sistema de coordenadas. Todavia, na prática, é necessário transladar e rotacionar a câmera para que o objeto de interesse fique dentro do campo de visão da mesma [Shah, 1997].

Conforme já comentado, uma imagem em formato digital é discretizada em pixels, os quais são representados em uma matriz em coordenadas de pixel (u, v) [Weng et al., 1992]. O valor de cada coordenada depende do centro ótico mapeado em pixel (u_c, v_c) e do tamanho de cada pixel $b_u \; x \; b_v$, os quais são definidos de acordo com a Figura 2.9.

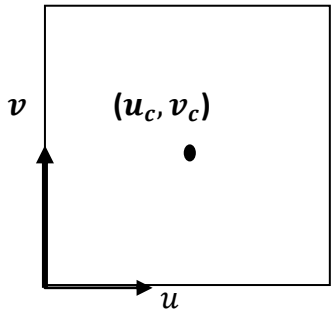


Figura 2.9 - Imagem com seu sistema de coordenadas (u, v) e seu centro ótico (u_c, v_c)
Fonte: JUNG, 2013.

Desta forma, as coordenadas em pixel da imagem podem ser expressas por:

$$u = u_c + b_u(x) \tag{2.44}$$

$$v = v_c + b_v(y) \tag{2.45}$$

então, substituindo a Equação (2.41) na Equação (2.44) e substituindo a Equação (2.40) na Equação (2.45), têm-se:

$$u = u_c + b_u\left(-\frac{fX}{Z}\right) \tag{2.46}$$

$$v = v_c + b_v\left(-\frac{fY}{Z}\right) \tag{2.47}$$

e, representando essas equações na forma matricial em coordenadas homogêneas de peso ω, têm-se:

$$\begin{bmatrix} u_\omega \\ v_\omega \\ \omega \end{bmatrix} = \begin{bmatrix} b_u & 0 & u_c \\ 0 & b_v & v_c \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ -Z/f \end{bmatrix} \tag{2.48}$$

$$(u, v) = (u_\omega/\omega, v_\omega/\omega) \tag{2.49}$$

A Equação (2.48) correlaciona coordenadas da imagem com as do espaço levando em consideração apenas os parâmetros intrínsecos à câmera. Isso só poderia ser utilizado no caso da lente estar localizada sobre o centro de coordenadas adotado no espaço, o que não é usual. Na prática, a câmera costuma estar transladada e rotacionada em relação ao centro de coordenadas adotado no espaço cartesiano tridimensional [Shah, 1997]. Por isso, faz-se necessário levar em consideração estas transformações que geram parâmetros extrínsecos à câmera, conforme apresentado na Seção 2.2.3.4.

2.2.3.4 Parâmetros extrínsecos à câmera: rotação + translação

Como visto anteriormente, para transformar as coordenadas do ambiente tridimensional para coordenadas da câmera, deve-se levar em conta a rotação e a translação da câmera em relação ao sistema de coordenadas de referência.

Considerando a câmera esquematizada na Figura 2.6, a qual apresenta uma rotação e uma translação em relação ao sistema de coordenadas do espaço cartesiano ($\boldsymbol{x_w}$). Neste caso,

o sistema de coordenadas da câmera ($\boldsymbol{x_c}$) sofrerá alterações conforme a rotação ($\boldsymbol{R}$) e a translação ($\boldsymbol{x_0}$) aplicadas.

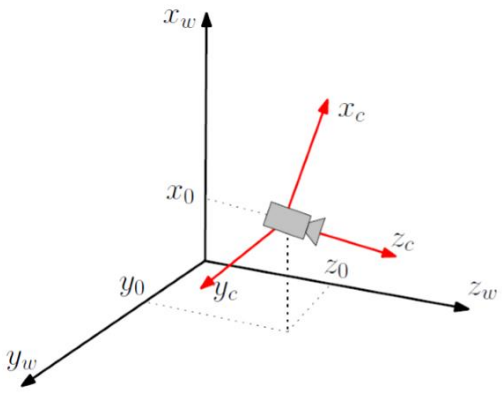


Figura 2.10 - Representação dos sistemas de coordenadas da câmera ($\boldsymbol{x_c}$) em relação ao sistema de coordenadas cartesianas ($\boldsymbol{x_w}$)
Fonte: Jung, 2013.

Desta forma:

$$\boldsymbol{x_c} = \boldsymbol{R}(\boldsymbol{x_w} - \boldsymbol{x_0}) \tag{2.50}$$

sendo,

$$\boldsymbol{x_w^h} = \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}, \boldsymbol{x_c^h} = \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} \; e \; \boldsymbol{x_0^h} = \begin{bmatrix} x_0 \\ y_0 \\ z_0 \\ 1 \end{bmatrix} \tag{2.51}$$

$$\boldsymbol{x_c^h} = \begin{bmatrix} \boldsymbol{R} & -\boldsymbol{x_0} \\ \boldsymbol{0}_{1x3} & 1 \end{bmatrix} \boldsymbol{x_w^h} \tag{2.52}$$

obtém-se, assim, a equação de projeção final, a qual leva em conta a translação do centro ótico em relação ao eixo de coordenadas tridimensional, a rotação do plano da imagem, a distância focal f, o centro ótico da câmera (u_c, v_c) e o tamanho dos pixels (b_u, b_v). Com isso, obtém-se a matriz calibração de câmera ($\boldsymbol{P}$) que é uma matriz que considera todos estes parâmetros intrínsecos e extrínsecos acumulados e que pode ser utilizada para transformar coordenadas de pixel em coordenadas mundo e vice-versa [Shah, 1997].

$$\boldsymbol{u^h} = \begin{bmatrix} u_\omega \\ v_\omega \\ \omega \end{bmatrix} \boldsymbol{P x_w^h} \tag{2.53}$$

ou,

$$\boldsymbol{u^h} = \begin{pmatrix} u_\omega \\ v_\omega \\ \omega \end{pmatrix} \begin{bmatrix} -fb_u & 0 & u_c \\ 0 & -fb_v & v_c \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{R} & -\boldsymbol{R x_0} \\ \boldsymbol{0}_{1x3} & 1 \end{bmatrix} \boldsymbol{x_w^h} \tag{2.54}$$

Isolando a matriz de calibração de câmera ***P*** na Equação (2.54), obtém-se:

$$\boldsymbol{P} = \begin{bmatrix} -fb_u & 0 & u_c \\ 0 & -fb_v & v_c \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{R} & -\boldsymbol{R x_0} \\ \boldsymbol{0}_{1x3} & 1 \end{bmatrix} \boldsymbol{x_w^h} \tag{2.55}$$

Segundo Shah, 1997, como é difícil obter esses parâmetros, já que dependem da medição de ângulos de rotação e translações da câmera em relação ao sistema de coordenadas mundo, além do conhecimento de parâmetros intrínsecos da câmeras, os quais muitas vezes são desconhecidos, pode-se determinar a matriz de calibração de câmera (***P***) sem a necessidade de se conhecer os parâmetros. Esse procedimento é o procedimento de calibração de câmera apresentado a seguir na Seção 2.2.4.

2.2.4 Calibração de câmera

Conforme visto acima, para relacionar as coordenadas no espaço com as coordenadas da imagem é preciso obter a matriz de calibração de câmera ***P***. Contudo, conforme comentado, muitas vezes é difícil obter com precisão todos os parâmetros necessários [Shah, 1997]. Assim, de acordo com Shah, 1997 e também com Ballard e Brown, 1982, uma alternativa é a de realizar a calibração de câmera, a qual permite obter-se uma relação adequada entre as coordenadas da imagem digitalizada e o espaço de trabalho do robô a partir da relação de alguns pontos com coordenadas conhecidas com as coordenadas correspondentes a esses pontos na imagem.

Um método largamente utilizado é o DLT[13] proposto por Abdel-Aziz e Karara, 1971, no qual o objetivo é determinar a relação entre as coordenadas no ambiente tridimensional e

[13] DLT (*Direct Linear Transformation),* Transformação Linear Direta, é um método elaborado por Abdel-Aziz e Karara, 1971, onde coordenadas espaciais de pontos são obtidas através de uma transformação direta de coordenadas da imagem de um analisador para as coordenadas no espaço-objeto, utilizando princípios da estereofotogrametria.

da imagem em pixels a partir da utilização de um determinado conjunto de pontos de controle, dos quais devem ser conhecidas duas coordenadas em ambos os sistemas, de forma que os termos da matriz ***P*** possam ser obtidos sem a necessidade de uso da Equação (2.55).

Assim, como a matriz (***P***) é de dimensão 3 x 4, pode-se representá-la como um arranjo de 12 incógnitas:

$$\boldsymbol{P} = \begin{bmatrix} k_{11} & k_{12} & k_{13} & k_{14} \\ k_{21} & k_{22} & k_{23} & k_{24} \\ k_{31} & k_{32} & k_{33} & k_{34} \end{bmatrix} \tag{2.56}$$

onde k_{ij} é cada incógnita a ser descoberta para montagem da matriz de calibração P.

Substituindo-se a Equação (2.56) na Equação (2.53), tem-se:

$$\begin{pmatrix} u_\omega \\ v_\omega \\ \omega \end{pmatrix} = \begin{bmatrix} k_{11} & k_{12} & k_{13} & k_{14} \\ k_{21} & k_{22} & k_{23} & k_{24} \\ k_{31} & k_{32} & k_{33} & k_{34} \end{bmatrix} \begin{pmatrix} x_w \\ y_w \\ z_w \\ 1 \end{pmatrix} \tag{2.57}$$

Desconsiderando a coordenada *z*, ou seja, considerando o espaço bidimensional, a terceira coluna da matriz (***P***) apresentada na Equação (2.57) deve ser removida já que a mesma seria multiplicada pela coordenada z_w, assim resulta:

$$\begin{pmatrix} u_\omega \\ v_\omega \\ \omega \end{pmatrix} = \begin{bmatrix} k_{11} & k_{12} & k_{14} \\ k_{21} & k_{22} & k_{24} \\ k_{31} & k_{32} & k_{34} \end{bmatrix} \begin{pmatrix} x_w \\ y_w \\ 1 \end{pmatrix} \tag{2.58}$$

onde k_{ij} é cada incógnita a ser descoberta para montagem da matriz de calibração P com coordenadas mundo no plano *x,y*.

Conforme já comentado, a matriz de câmera ***P*** agrega o efeito acumulativo de todos os parâmetros. Assim, para cada ponto escolhido como ponto de controle, definem-se duas equações seguintes:

$$u_\omega = \frac{k_{11}x_w + k_{12}y_w + k_{13}z_w + k_{14}}{k_{31}x_w + k_{32}y_w + k_{33}z_w + k_{34}} \tag{2.59}$$

$$v_\omega = \frac{k_{21}x_w + k_{22}y_w + k_{23}z_w + k_{24}}{k_{31}x_w + k_{32}y_w + k_{33}z_w + k_{34}} \tag{2.60}$$

Desta forma, como há 12 incógnitas e, como para cada ponto escolhido obtém-se 2 equações, é necessário 6 pontos para resolver o sistema de equações. No caso de

desconsiderar a coordenada *z*, são 9 incógnitas (Equação 2.58) e, então, é necessário escolher 5 pontos para resolver o sistema de equações.

Colocando estas equações em forma matricial, verifica-se que se trata de um sistema linear do tipo $\boldsymbol{Ax} = \boldsymbol{b}$, conforme pode ser observado na Equação (2.61), onde n é número de pontos de controle. Trata-se de um sistema homogêneo com múltiplas soluções. Por isso, Shah, 1997, sugere escolher arbitrariamente o valor de uma das incógnitas e determinar os demais a partir deste. Um valor, geralmente arbitrado é $k_{34} = 1$.

$$\begin{bmatrix} x_1 & y_1 & z_1 & 1 & 0 & 0 & 0 & 0 & -u_1x_1 & -u_1y_1 & -u_1z_1 \\ 0 & 0 & 0 & 0 & x_1 & y_1 & z_1 & 1 & -v_1x_1 & -v_1y_1 & -v_1z_1 \\ x_2 & y_2 & z_2 & 1 & 0 & 0 & 0 & 0 & -u_2x_2 & -u_2y_2 & -u_2z_2 \\ 0 & 0 & 0 & 0 & x_2 & y_2 & z_2 & 1 & -v_2x_2 & -v_2y_2 & -v_2z_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_n & y_n & z_n & 1 & 0 & 0 & 0 & 0 & -u_nx_n & -u_ny_n & -u_nz_n \\ 0 & 0 & 0 & 0 & xn & yn & zn & 1 & -v_nx_n & -v_ny_n & -v_nz_n \end{bmatrix} \begin{bmatrix} k_{11} \\ k_{12} \\ k_{13} \\ k_{14} \\ k_{21} \\ k_{22} \\ k_{23} \\ k_{24} \\ k_{31} \\ k_{32} \\ k_{33} \end{bmatrix} = \begin{bmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \\ \vdots \\ u_n \\ v_n \end{bmatrix} \tag{2.61}$$

ou,

$$\boldsymbol{Ax} = \boldsymbol{b} \tag{2.62}$$

Como no sistema apresentado na Equação (2.61) a matriz $\boldsymbol{A}$ e o vetor $\boldsymbol{b}$ são conhecidos, para resolver o sistema pode-se aplicar o MMQ (Método dos Mínimos Quadrados) cujo critério adotado neste método é o da pseudo inversa conforme apresentado nas equações (2.63) e (2.64).

$$\boldsymbol{A}^T\boldsymbol{Ax} = \boldsymbol{A}^T\boldsymbol{b} \tag{2.63}$$

$$\boldsymbol{x} = (\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{b} \tag{2.64}$$

obtendo-se assim, os elementos do vetor *x* que, posteriormente, deve ser rearranjado como a matriz (***P)*** da Equação (2.56). Esta matriz serve para transformar coordenadas mundo em coordenadas da imagem de acordo com a Equação (2.57) e, também, para realizar a Perspectiva Inversa [Ballard e Brown, 1982], ou seja, coordenadas da imagem para coordenadas do espaço, usando a Equação (2.65):

$$\begin{pmatrix} x_w \\ y_w \\ z_w \\ 1 \end{pmatrix} = \begin{bmatrix} k_{11} & k_{12} & k_{13} & k_{14} \\ k_{21} & k_{22} & k_{23} & k_{24} \\ k_{31} & k_{32} & k_{33} & k_{34} \end{bmatrix}^{-1} \begin{pmatrix} u_\omega \\ v_\omega \\ \omega \end{pmatrix} \tag{2.65}$$

A Equação (2.65) permite então, correlacionar coordenadas no espaço com coordenadas em pixels de uma imagem. As coordenadas no espaço são relacionadas a um ponto definido como o centro do sistema de coordenadas. Como o sistema de visão deve ser aplicado à um robô pneumático de 5 GL, escolheu-se como centro do sistema de coordenadas do sistema de visão, o mesmo centro do sistema de coordenadas utilizado pelo sistema de controle do robô.

Como este ponto fica localizado na parte interna do robô, não pode ser visualizado na imagem capturada pela câmera. Por isso, no processo de calibração, escolheu-se o primeiro ponto do padrão de calibração (tabuleiro quadriculado) como centro do sistema de coordenadas e, posteriormente, relacionou-se este ponto com o sistema de coordenadas do robô.

Para isso, fez-se necessário conhecer o funcionamento, os parâmetros, geometria, espaço de trabalho do robô, bem como os algoritmos nele implementados, os quais são apresentados no Capítulo 3.

3 APLICAÇÃO DO MÉTODO EM UM ROBÔ CILÍNDRICO ACIONADO PENUMATICAMENTE

O método apresentado no Capítulo 2 para a identificação da posição e orientação de objetos através do cálculo de seus momentos geralmente é aplicável, independentemente da configuração do robô. Conforme já comentado, no presente trabalho, esse método será aplicado em um estudo de caso baseado em um robô em desenvolvimento no LAMECC/UFRGS, cujas características serão detalhadas a seguir.

3.1 Geometria e espaço de trabalho do robô

Com relação à configuração geométrica, o robô pode ser definido como um manipulador com acionamento pneumático de 5 GL do tipo RPP:RR, ou seja, o braço do manipulador é composto por uma junta rotacional (R) em sua base, enquanto que a segunda e a terceira juntas são prismáticas ortogonais (P). O punho, por sua vez, possui duas juntas rotacionais que conferem os movimentos de arfagem (inclinação) e rolagem ao efetuador. As variáveis de juntas são representadas neste trabalho por θ_i, para juntas rotacionais; e d_i para juntas prismáticas. Na Figura 3.1 é apresentado um esquema geral do sistema, onde pode-se verificar a disposição de suas juntas.

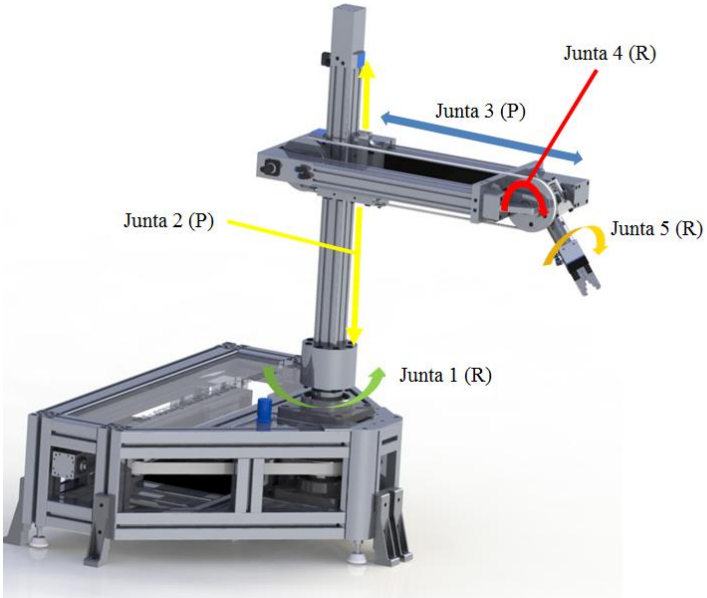


Figura 3.1 – Configuração das juntas do robô

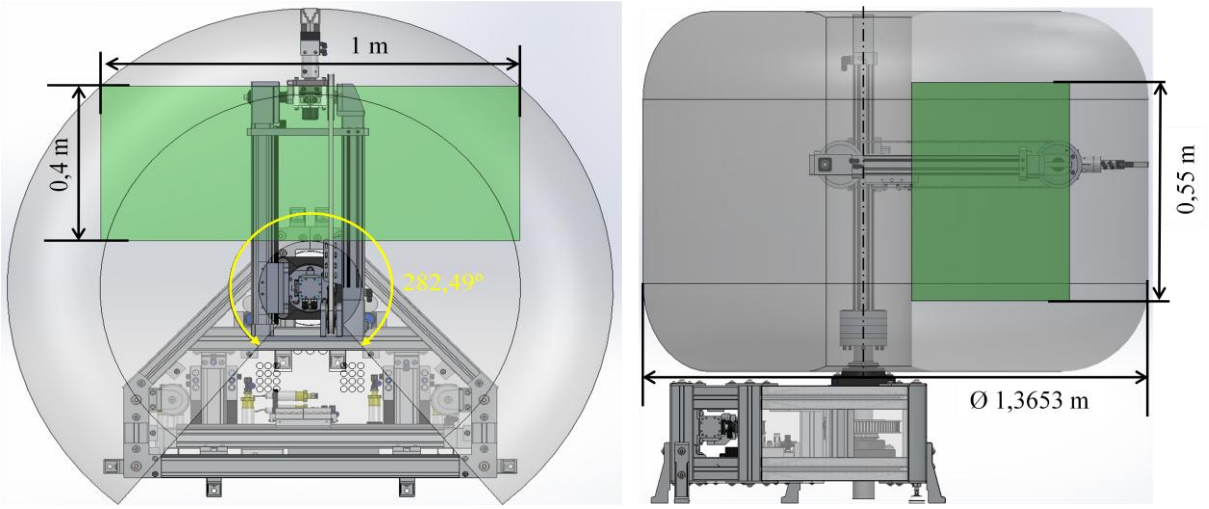
Na Tabela 3.1 são apresentados os valores limites de deslocamentos das juntas do robô pneumático calculados com base em um modelo geométrico do manipulador [Sarmanho, 2014].

Tabela 3.1- Valores limites das juntas do manipulador pneumático.

Fonte: Sarmanho, 2014

Junta	Valores Limites	
	Mínimo	Máximo
1	-2,4652 rad (-141,24°)	2,4652 rad (141,24°)
2	0	0,450 m
3	0	0,300 m
4	-1,9809 rad (-113,50°)	1,9809 rad (113,50°)
5	-2,3562 rad (-135,00°)	2,3562 rad (135,00°)

As dimensões básicas do espaço de trabalho do robô são apresentadas na Figura 3.2.



a) vista no plano xy b) vista no plano xz

Figura 3.2 - Espaço de trabalho do robô

Fonte: Sarmanho, 2014.

3.2 Cinemática do robô

A análise cinemática permite obter as equações que descrevem tanto a cinemática inversa quanto a cinemática direta de um robô. Missiaggia, 2014, realizou a análise cinemática direta do robô com base nos procedimentos clássicos do método Denavit-Hartenberg, enquanto que as equações da cinemática inversa foram obtidas a partir dos resultados da cinemática direta utilizando uma solução algébrica baseada em relações trigonométricas. De acordo com Fu et al, 1987, o método de Denavit-Hatenberg é um método sistemático de representação das relações cinemáticas entre um elo e seus elos adjacentes, onde são necessários quatro parâmetros por elo para a definição completa do mesmo. Na Figura 3.3 apresenta-se uma representação simplificada da estrutura da cadeia cinemática do manipulador pneumático definida a partir do algoritmo de Denavit-Hartenberg.

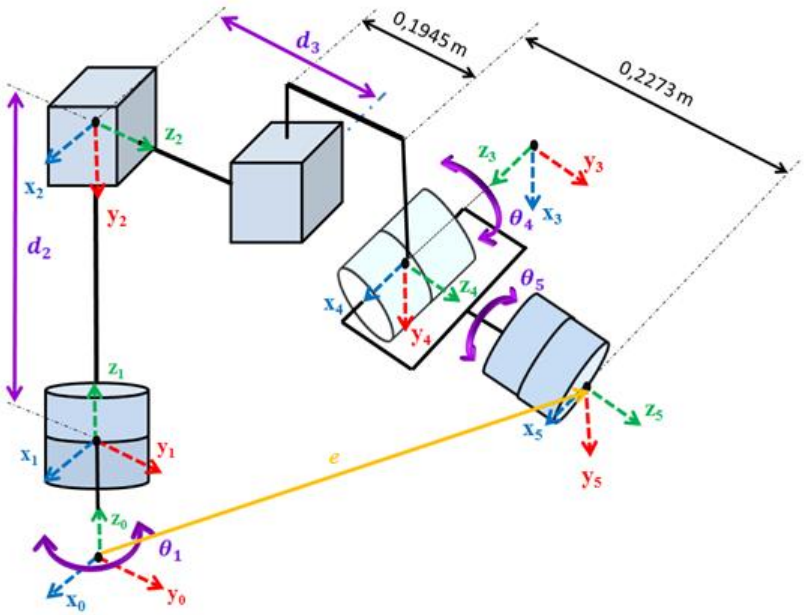


Figura 3.3 - Sistemas de coordenadas na representação simplificada dos elos e juntas do robô
Fonte: adaptado de Sarmanho, 2014.

Conforme já comentado, o trabalho de Missiaggia, 2014, focou o desenvolvimento de uma metodologia de planejamento de trajetória, a qual será descrita em detalhes na Seção 3.3. Na Tabela 3.2 estão apresentados os parâmetros de Denavit-Hartenberg e os parâmetros utilizados nas matrizes homogêneas de transformação. Conforme afirmado anteriormente, um

dos objetivos específicos do presente trabalho é integrar o programa de planejamento de trajetórias de cada atuador desenvolvido por Missiaggia, 2014, com o algoritmo de controle dos atuadores pneumáticos desenvolvido por Sarmanho, 2014.

Os parâmetros de Denavit-Hartenberg foram recalculados por Sarmanho, 2014, e sofreram alterações em relação aos anteriormente calculados por Missiaggia, 2014. Contudo, o comprimento da garra calculado por Sarmanho, 2014, de *0,1883 m* é substituído pelo valor calculado por Missiaggia, 2014, de *0,2273 m*, já que este último contemplava o comprimento até a ponta da garra enquanto que no primeiro era até sua base. Assim, na Tabela 3.2 são apresentados os valores dos parâmetros calculados por Sarmanho, 2014, incluindo a modificação citada, os quais são utilizados como base para a integração dos dois algoritmos.

Tabela 3.2- Parâmetros de Denavit-Hartenberg do robô pneumático
Fonte: adaptado de Sarmanho, 2014

Elos	α *[rad]*	*a [m]*	θ *[rad]*	*d [m]*
1	0	0	θ_1	0,263
2	$-\pi/2$	0	0	d_2
3	$\pi/2$	0	$\pi/2$	$0{,}1945 + d_3$
4	$-\pi/2$	0	θ_4	0
5	0	0	θ_5	0,2273

Com relação ao Elo 1, o valor *0,263 m*, apresentado na Tabela 3.2, faz referência à distância vertical que a origem do sistema de coordenadas adotado por Sarmanho, 2014, está da região de contato com o solo. Já, a origem do sistema de coordenadas adotado por Missiaggia, 2014, se encontra sobre a base, ou seja, próximo à região de contato com o solo. A alteração deste parâmetro com relação ao utilizado por Sarmanho, 2014, faz-se necessária no algoritmo de Missiaggia, 2014, a fim de unificar as referências.

Como resultado da aplicação do método de Denavit-Hartenberg, cada elo possui um sistema de coordenadas, desde a base inercial (índice 0) até o efetuador (índice 5 neste caso de 5GL). Na Equação (3.3) é apresentada a matriz de transformação de coordenadas recalculada para a integração dos algoritmos propostos por Missiaggia, 2014 e Sarmanho, 2014. Essa matriz contempla todas as rotações e translações do efetuador em relação à origem do sistema de coordenadas. A partir dessa matriz, é possível determinar as equações de posição e orientação do efetuador. Para se obter essa matriz, realiza-se a multiplicação sucessiva das

matrizes de transformação individuais de cada elo, partindo da base até o efetuador, conforme a Equação (3.1).

$$\mathbf{{}^{0}T_{5}} = \mathbf{{}^{0}T_{1}\ {}^{1}T_{2}\ {}^{2}T_{3}\ {}^{3}T_{4}\ {}^{4}T_{5}} \tag{3.1}$$

Onde ${}^{i}T_{j}$ é a matriz de transformação do elo j para o elo i. Segundo, Missiaggia, 2014, a matriz ${}^{0}T_{5}$ pode ser entendida como uma composição de uma matriz de orientação (***B***) com o vetor (***e***) que apresenta a posição final do efetuador, como pode ser identificado graficamente na Figura 3.3 e conforme é apresentada na Equação (3.2) que segue:

$$\mathbf{{}^{0}T_{5}} = \begin{bmatrix} & \boldsymbol{B} & & \boldsymbol{e} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.2}$$

O vetor (***e***) pode ser decomposto em e_x, e_y e e_z conforme apresentada na última coluna da matriz ${}^{0}T_{5}$ na Equação (3.3).

$${}^{0}T_{5} = \begin{bmatrix} -\cos\theta_1 \operatorname{sen}\theta_5 - \cos\theta_5 \operatorname{sen}\theta_1 \operatorname{sen}\theta_4 & \operatorname{sen}\theta_1 \operatorname{sen}\theta_4 \operatorname{sen}\theta_5 - \cos\theta_1 \cos\theta_5 & -\cos\theta_4 \operatorname{sen}\theta_1 & e_x \\ \cos\theta_1 \cos\theta_5 \operatorname{sen}\theta_4 - \operatorname{sen}\theta_1 \operatorname{sen}\theta_5 & -\cos\theta_5 \operatorname{sen}\theta_1 - \cos\theta_1 \operatorname{sen}\theta_4 \operatorname{sen}\theta_5 & \cos\theta_1 \cos\theta_4 & e_y \\ -\cos\theta_4 \cos\theta_5 & \cos\theta_4 \operatorname{sen}\theta_5 & \operatorname{sen}\theta_4 & e_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.3}$$

sendo,

$$e_x = -0{,}2273\, cos\,\theta_4\, sen\,\theta_1 - (d_3 + 0{,}1945)\, sen\,\theta_1 \tag{3.4}$$

$$e_y = 0{,}2273\, cos\,\theta_1\, cos\,\theta_4 + (d_3 + 0{,}1945)\, cos\,\theta_1 \tag{3.5}$$

$$e_z = 0{,}2273\, sen\,\theta_4 + d_2 + 0{,}263 \tag{3.6}$$

Para casos, como no presente estudo, em que $e_y \geq 0$, Missiaggia, 2014, calcula θ_1 usando $cos\,\theta_1 = \frac{e_y}{\sqrt{{e_x}^2 + {e_y}^2}}$. Assim, de posse dos valores das coordenadas de e_x, e_y e e_z (em relação ao sistema de coordenadas do robô) que definem a posição na qual se deseja que o efetuador se posicione, obtém-se os valores de θ_1, d_2 e d_3, respectivamente, por meio das equações (3.7), (3.8) e (3.9):

$$\theta_1 = sen^{-1}\left(\frac{e_x}{-d_3-0{,}1945-0{,}2273\,cos\,\theta_4}\right) = -sen^{-1}\left(\frac{e_x}{\sqrt{{e_x}^2+{e_y}^2}}\right) \quad (3.7)$$

$$d_2 = e_z - 0{,}263 - 0{,}2273\,sen\,\theta_4 \quad (3.8)$$

$$d_3 = \frac{e_y}{cos\,\theta_1} - 0{,}1943 - 0{,}2273\,cos\,\theta_4 = \sqrt{{e_x}^2 + {e_y}^2} - 0{,}1943 - 0{,}2273\,cos\,\theta_4 \quad (3.9)$$

Desta maneira, dada uma matriz de coordenadas (e_x, e_y, e_z, θ_4 e θ_5), a qual representa um conjunto genérico de pontos no espaço tridimensional, é possível determinar o valor da coordenada das juntas ($\theta_1, d_2\ e\ d_3$).

3.3 Planejamento de trajetórias

A principal aplicação do robô em desenvolvimento é a manipulação de peças, ou seja, sua aplicação usual é em tarefas do tipo *pick-and-place*, que consiste na ação de capturar um objeto em uma determinada posição e movimentá-lo para colocá-lo em uma dada posição desejada. Esse tipo de tarefa usualmente necessita da continuidade e suavidade do movimento [Müller-Karger et. al., 2000].

Para criar e realizar tarefas desse tipo existem duas formas distintas de se programar o robô: ensinando-o ativa ou passivamente (programação *online*), ou através da utilização de software de simulação do robô e do ambiente de trabalho (programação *off-line*). Na programação *off-line*, as tarefas necessárias são realizadas sem a necessidade do uso do robô, geralmente em um computador, para serem enviadas posteriormente ao controlador do manipulador que irá executá-las. Na programação *on-line*, as tarefas são feitas diretamente no robô, seja através de *joysticks* ou através da movimentação direta do braço robótico, sendo necessário gravar manualmente as trajetórias durante a ação do robô para posteriormente repeti-las [Lewis et. al., 2003]. Pacheco, 2010, defende que a programação *off-line* oferece vantagens, como maior precisão no seguimento da trajetória do manipulador, possibilidade de simulações de trajetória, geração de trajetórias suaves, validações de movimentos e possibilidade de realização de simulações físicas, com verificação de colisões do manipulador com objetos no espaço de trabalho ou, até mesmo, a detecção de inválidas. Por outro lado, a programação *off-line* é mais complexa, exige um maior planejamento da atividade, um estudo

aprofundado sobre o funcionamento e limitações do robô, é preciso prever os possíveis imprevistos podendo existir incompatibilidade entre a simulação e a realidade.

Um fator importante consiste na definição da trajetória no modo *off-line* de programação. Uma maneira bastante comum é através de um ponto inicial e final, aplicando um algoritmo que definirá a trajetória baseando-se em critérios como o mínimo gasto de energia em rotas livres de colisões, mínima aceleração, mínimo tempo, mínimo *jerk* (derivada primeira da aceleração), entre outros. Esta tarefa caracteriza um problema de controladores robóticos chamado de "planejamento de trajetória", geralmente envolvendo a modelagem dinâmica do robô, já que a energia utilizada pelo manipulador para realizar a tarefa é levada em consideração na escolha da melhor trajetória. Outra maneira é através da definição de vários pontos que representarão o trajeto completo (coordenadas pelas quais o efetuador deverá passar durante o trajeto), geralmente utilizando-se modelagem cinemática do robô [Shen e Huper, 2005]. Existem, basicamente, duas formas de executar a movimentação de um manipulador sobre uma trajetória: a movimentação ponto a ponto por segmentos de retas; e a movimentação baseada em *splines* ponto a ponto [Pacheco, 2010].

No trabalho realizado por Missiaggia, 2014, são utilizadas funções *splines*[14] obtidas por meio da metodologia proposta por Simon, 2004, para a geração de trajetórias formadas por um determinado número de pontos a serem percorridos pelo efetuador buscando a minimização do *jerk*. Esses pontos definem regiões no espaço onde o efetuador deverá passar, sendo denominados como "pontos chave da trajetória". As trajetórias são geradas através de um algoritmo de aproximação de pontos através de *splines* compostas por polinômios de sétimo grau. Essa escolha garante a continuidade da função de posição, bem como de suas três primeiras derivadas, sendo essa uma condição necessária para a implantação de importantes leis e estratégias de controle (como, por exemplo, a estratégia em cascata, utilizada com sucesso no controle de sistemas servopneumáticos). O método proposto por Simon, 2004, possibilita, através do ajuste de parâmetros em função da exigência de cada aplicação, a obtenção de curvas no espaço das juntas com valores otimizados de *jerk*, aceleração ou velocidade [Missiaggia, 2014].

[14] *Spline* é uma curva definida por dois ou mais pontos de controle ou pontos chave. Elas podem ser obtidas por meio de interpolação que passam exatamente por todos os pontos de controle ou por aproximação que passam próximo a todos os pontos de controle

Desta forma, é necessário informar ao programa quais os pontos o efetuador deve percorrer, incluindo seus ângulos de rotação em cada ponto chave. Esses valores são informados em forma de uma matriz e o programa executa, primeiramente, uma etapa de validação para verificar se os pontos se encontram dentro do espaço de trabalho do robô. Também deve ser definido o tempo máximo de um ponto até o outro.

Com a aplicação das equações da cinemática inversa obtidas através dos parâmetros de Denavit-Hatenberg, apresentadas na Seção 3.2, são obtidos os vetores de posição no espaço das juntas onde é realizada a geração das trajetórias através de *splines* de sétimo grau. Esses vetores são avaliados através de simulações por meio do modelo cinemático direto e são posteriormente enviados ao programa de controle dos atuadores, o qual comanda a execução do movimento do robô.

O programa de reconhecimento da orientação de objetos baseado em seus momentos apresentado no Capítulo 2 fornece a posição em x e em y do centro de massa do objeto e o ângulo de giro da garra do manipulador necessário para capturar corretamente o objeto.

Na Figura 3.3, pode-se observar o ângulo correspondente à junta rotacional 5 (chamado de θ_5) no programa de planejamento de trajetórias. Com isso, a entrada necessária deste programa é uma matriz ($\boldsymbol{PC}$) com os valores de x, y, z, θ_4 e θ_5 dos pontos (n) por onde o manipulador deve passar, sendo θ_4 o ângulo da junta rotacional 4 (Figura 3.3). A Equação (3.10) corresponde à matriz de entrada, onde cada coluna representa as coordenadas cartesianas e os ângulos de cada ponto chave:

$$\boldsymbol{PC} = \begin{bmatrix} x_1 & x_2 & x_3 & \dots & x_n \\ y_1 & y_2 & y_3 & \dots & y_n \\ z_1 & z_2 & z_3 & \dots & z_n \\ \theta_{4_1} & \theta_{4_2} & \theta_{4_3} & \dots & \theta_{4_n} \\ \theta_{5_1} & \theta_{5_2} & \theta_{5_3} & \dots & \theta_{5_n} \end{bmatrix} \tag{3.10}$$

sendo que o valor mínimo de pontos chaves a serem especificados é igual a 4, porém, quanto maior for este número a partir da curva para a obtenção da trajetória, melhor será a capacidade da *spline* em reproduzi-la de maneira satisfatória (Missiaggia, 2014).

O programa de reconhecimento da orientação de objetos baseado em seus momentos apresentado no Capítulo 2 analisa o objeto a partir da sua vista superior, ou seja, ele não leva em consideração a sua altura (coordenada z do robô). No caso de haver somente objetos de mesma altura, isso não se caracteriza um problema, pois basta medir a altura do objeto e, ao

criar a matriz, adotar a mesma coordenada z para todos os pontos. Porém, ao trabalhar com objetos de alturas diversas, é necessário reconhecer a altura dos mesmos, já que a trajetória do manipulador deve evitar possíveis colisões. Para isso, outra câmera, poderia ser usada, caracterizando um sistema estéreo. Trabalhos futuros deverão englobar esta técnica.

3.4 Determinação dos pontos chaves

Conforme já mencionado, o programa de geração de trajetórias proposto por Missiaggia, 2014, gera *splines* entre um ponto inicial e um final, passando por regiões definidas no entorno espacial por pontos intermediários previamente estabelecidos. Diversos parâmetros podem ser alterados conforme a aplicação, permitindo definir se a trajetória terá o mínimo *jerk*, mínima aceleração, mínimo desvio de posição entre o ponto programado e o gerado, entre outras características.

Segundo Pacheco, 2010, para a definição dos pontos chaves que definem uma trajetória entre a posição atual (genérica) do efetuador e a posição identificada por meio da câmera, é necessário estabelecer uma trajetória padrão associada ao tipo de ação desejada. Por exemplo, em uma ação de movimentação de uma peça (*pick-and-place*), entre um ponto inicial (x_i, y_i, z_i, θ_{4_i} *e* θ_{5_i}) e um ponto final (x_f, y_f, z_f, θ_{4_f} e θ_{5_f}), nem sempre o ideal é que o robô faça o menor caminho possível entre esses pontos, já que é preciso levar em consideração fatores como o espaço de trabalho, os obstáculos existentes e a suavidade do movimento. Assim, uma trajetória padrão que supere essas dificuldades deve ser adotada e, a partir dos pontos chaves dessa trajetória, um algoritmo de interpolação pode gerar os pontos intermediários que definem a trajetória em cada passo de processamento de uma forma suave.

O processo de suavização de trajetória permite que uma trajetória suave seja gerada a partir dos pontos chaves que definem a forma geral da curva. O processo de interpolação é geralmente realizado através da utilização de curvas interpoladoras ou de polinômios, podendo-se avaliar a suavidade do trajeto gerado por meio da análise da variação da tangente da curva ao longo de todos os pontos [Pacheco, 2010].

O programa de planejamento de trajetórias proposto por Missiaggia, 2014, leva em consideração uma estratégia de suavização do movimento, a qual depende de uma definição adequada dos pontos chaves. Por exemplo, na Figura 3.4 está apresentado um movimento para uma aplicação *pick-and-place* constituída por uma trajetória em que a garra captura a

peça pelo seu topo, leva-a até uma determinada altura (que deve ser maior que os possíveis obstáculos para evitar colisões), realiza um movimento para cima em forma de um semiarco e movimenta-se horizontalmente até um ponto pré-estabelecido, para então realizar um semiarco para baixo, até um ponto a partir do qual movimenta-se verticalmente até o ponto desejado. Um dos parâmetros que pode interferir na suavidade deste tipo de trajetória é o tamanho do raio do semiarco.

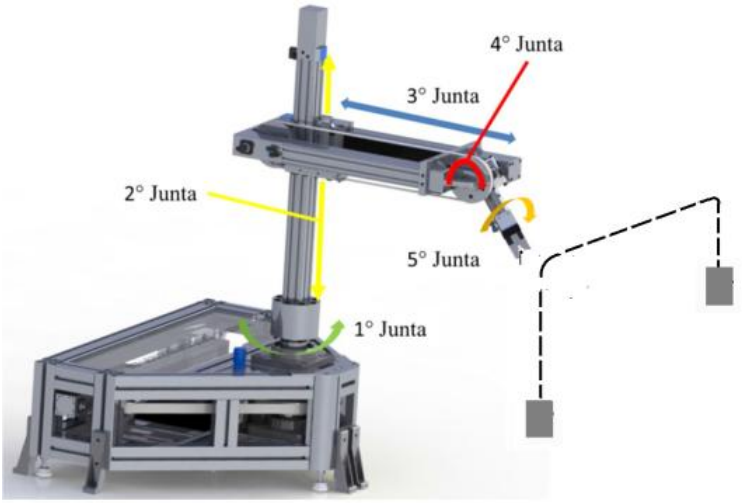


Figura 3.4 - Representação de uma trajetória *pick-and-place*

Conforme pode ser observado na Figura 3.5, a trajetória *pick-and-place* é composta por uma composição de duas trajetórias verticais, uma trajetória horizontal e duas trajetórias circulares.

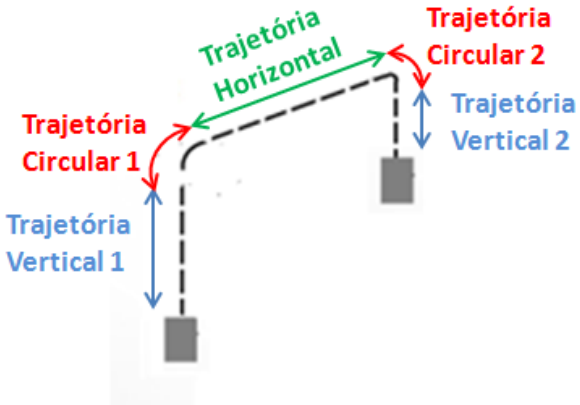


Figura 3.5 - Representação da composição de uma trajetória *pick-and-place*

Para isso, são elaboradas três rotinas para geração dos pontos da trajetória *pick-and-place*, a "*Go Horizontal to Position"*, a *"Go Vertical to Position"* e a *"Go Circular to Position"*, que definem, respectivamente, uma trajetória horizontal, uma trajetória vertical e uma circular. Estas rotinas são descritas a seguir.

a) Go Horizontal To Position:

A rotina trajetória horizontal, gera automaticamente uma sequência de pontos que, ao serem percorridos, formam um trajeto na direção horizontal (perpendicular ao eixo z) entre o ponto inicial e o ponto final. As ações associadas a esta rotina são as seguintes:

- No programa de planejamento de trajetórias, deve ser chamada a rotina "Go Horizontal To Position", a qual solicita as informações que descrevem o ponto incial e o final de deslocamento. Os dados necessários são: x_i, y_i, $z_i, \theta_{4_i}, \theta_{5_i}, x_f, y_f, \theta_{4_f}$ e θ_{5_f}, sendo que o subíndice *i* faz referência às coordenadas do ponto inicial e o subíndice *f* às coordenadas do ponto final.
- O usuário deve definir o número de pontos (n) a serem criados entre o ponto inicial e final;
- Calcula-se a diferença entre os valores x_i e x_f, entre y_i e y_f e entre θ_{5_i}e θ_{5_f}. Atribui-se o valor de -90º a θ_4, correspondendo ao efetuador virado para baixo, visando a pegar a peça na direção do seu topo, conforme pode ser observado na Figura 3.4;
- Divide-se cada uma destas diferenças pelo número de pontos, obtendo-se, assim, a distância entre cada ponto a ser criado;
- Soma-se esse valor ao ponto inicial em x, em y e em θ_5, obtendo-se o próximo ponto. Esta ação é repetida para cada novo ponto até alcançar o valor do ponto final;
- Monta-se a matriz de pontos chaves (***PC***) utilizando a Equação (3.10).

b) Go Vertical To Position

A rotina vertical gera automaticamente uma sequência de pontos que, ao serem percorridos, formam uma linha vertical (paralela ao eixo z) entre o ponto inicial e o ponto final. As ações associadas são as seguintes:

- O programa de planejamento de trajetórias chama a rotina “Go Vertical To Position” que se solicita os seguintes dados: x_i,y_i,z_i, θ_{4_i}, θ_{5_i}, θ_{4_f}e θ_{5_f};
- Define-se $x_i = x_f$ e $y_i = y_f$.
- Atribui-se o valor de -90º a θ_4;
- O usuário define o número de pontos chave a serem criados entre o ponto inicial e final;
- Calcula-se as diferenças entre os valores z_i e z_f e entre θ_{5_i}e θ_{5_f}.
- Divide-se cada uma destas diferenças pelo número de pontos, obtendo-se assim a distância entre cada ponto a ser criado;
- Soma-se este valor ao ponto inicial em z e em θ_5, obtendo-se o próximo ponto, repetindo essa ação até chegar ao valor do ponto final;
- Monta-se a matriz de pontos chaves (***PC***) utilizando a Equação (3.10).

c) Go Circular To Position

A rotina circular gera automaticamente uma sequência de pontos que, ao serem percorridos, formam um semiarco de 0º a 90º entre o ponto inicial e o ponto final. As ações são as seguintes:

- O programa de planejamento de trajetórias solicita os valores de x_i, y_i, $z_i, \theta_{4_i}, \theta_{5_i}, x_f, y_f, \theta_{4_f}$ e θ_{5_f};
- Atribui-se o valor de -90º a θ_4;
- É então chamada a rotina “Go Circular To Position”, na qual são executadas as tarefas que seguem;
- O usuário define o número de pontos a serem criados entre o ponto inicial e final;
- Calcula-se a diferença entre os valores x_i e x_f, entre y_i e y_f, entre z_i e z_f e entre θ_{5_i}e θ_{5_f}, obtendo-se $\Delta x, \Delta y$ *e* $\Delta\theta_5$, respectivamente, conforme ilustrado na Figura 3.6;
- Calcula-se a distância euclidiana (DE) entre os dois pontos, conforme apresentado na Figura 3.6;
- Calcula-se o ângulo (α) formado entre DE e Δx;

- Divide-se o arco de 90º pelo número de pontos menos 1, obtendo-se o ângulo de giro (γ) entre os pontos (Figura 3.7);
- Sendo $\Delta z = |z_f - z_i|$, para que a coordenada z_f seja corretamente atingida com este método, faz-se $\Delta z = DE$, e como $(DE)^2 = \Delta x^2 + \Delta y^2$, então $\Delta z^2 = \Delta x^2 + \Delta y^2$. Por meio do comando "Go Circular To Position", pode-se escolher $\Delta x = \Delta y$, o que permite obter Δz por meio da Equação (3.11):

$$\Delta z = \sqrt{2(\Delta x)^2} \tag{3.11}$$

- A partir de DE e γ, calcula-se a distância de cada ponto da trajetória circular (Figura 3.7) até a origem no plano formado por DE e z;
- Para transladar estes valores para o plano x,y,z, basta multiplicar este valor por cosseno de α para obter o valor em x e por seno de α para obter o valor em y;
- Gera-se os pontos, obtendo-se a matriz da trajetória usando a Equação (3.10).

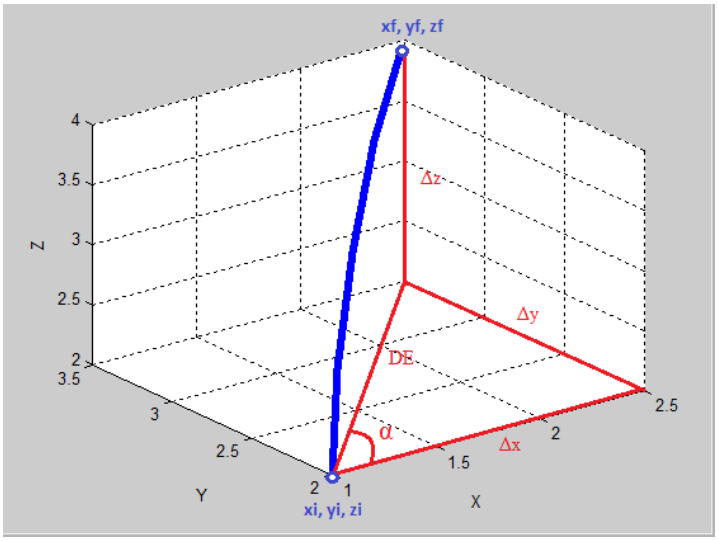


Figura 3.6 - Representação da trajetória de um semiarco

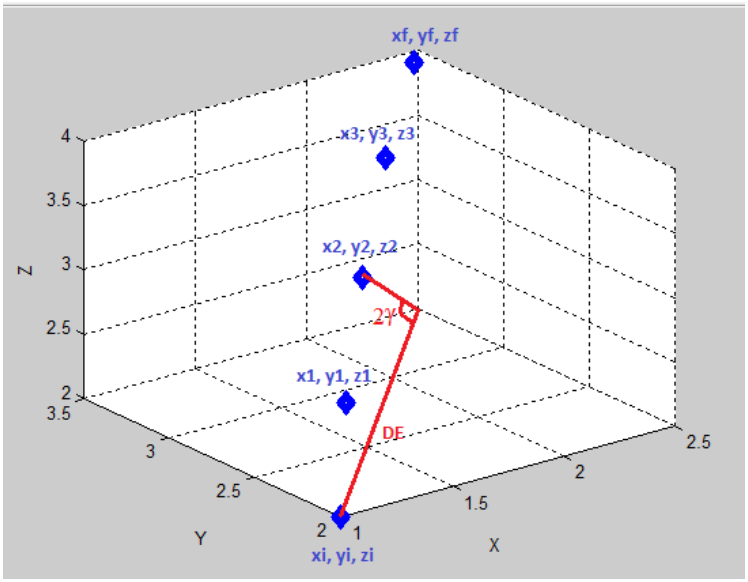


Figura 3.7 - Representação da trajetória de um semiarco com *n*=5

Com as três sub-rotinas apresentadas, é possível montar a rotina que realiza a trajetória apresentada na Figura 3.5, já que a mesma é uma composição das trajetórias geradas por meio das três sub-rotinas (duas trajetórias verticais, uma trajetória horizontal e duas trajetórias circulares), definindo a rotina referente ao comando *pick-and-place*. Para isso, é preciso informar os pontos iniciais e finais de cada trajetória de cada sub-rotina. Na Figura 3.8, pode-se observar os pontos intermediários a serem determinados conforme os procedimentos que seguem.

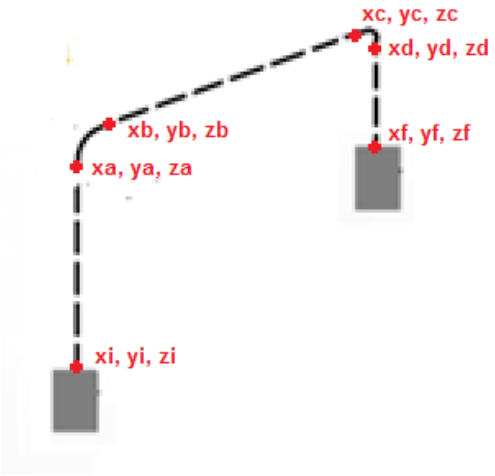


Figura 3.8 - Representação dos pontos da trajetória "*pick-and-place*"

As trajetórias geradas devem estar confinadas no espaço de trabalho do robô. Assim, os valores destes limites são utilizados para estabelecer as restrições impostas às trajetórias geradas. Para defini-los, faz-se necessário adotar o mesmo sistema de coordenadas nos algoritmos de planejamento de trajetórias e de controle dos atuadores. Na Figura 3.9, pode ser obervada a representação do sistema de coordenadas (em vermelho) adotado no algoritmo de controle dos atuadores [Sarmanho, 2014] e do sistema de coordenadas adotado no algoritmo de planejamento de trajetórias (em azul) [Missiaggia, 2014]. Visando à integração dos dois algoritmos, considerou-se o deslocamento de *0,263 m* na coordenada *z* de todos os pontos inseridos no programa de planejamento de trajetórias.

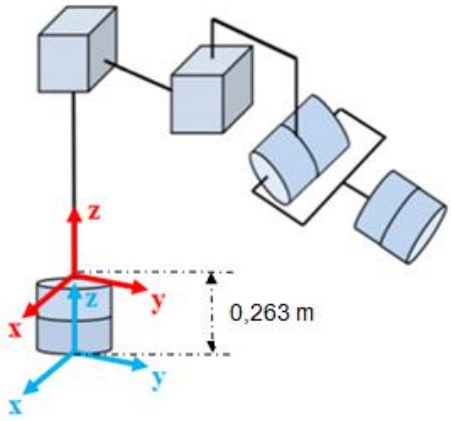


Figura 3.9 - Representação dos sistema de coordenadas adotado por Sarmanho, 2014 (em vermelho) e adotado por Missiaggia, 2014 (em azul)

No caso do movimento vertical, a altura máxima de toda a trajetória, ou seja, o valor das coordenadas z_b e z_c devem ser menores do que o valor máximo de z admitido pelo robô. Pela Tabela 3.1, percebe-se que o valor mínimo da junta 2 ($d_{2_{min}}$) é igual a *0 m* e o valor máximo ($d_{2_{max}}$) é de *0,450 m*, desconsiderando os elementos amortecedores de fim de curso. Segundo Sarmanho, 2014, o valor dos comprimentos amortecedores (d_ζ) nas juntas é de *0,020 m*. Com isso, calcula-se o valor mínimo possível para a junta 2 ($d_{2_{min\zeta}}$) considerando-se os amortecedores:

$$d_{2_{min\zeta}} = \left(d_{2_{min}} + d_\zeta\right) = (0 + 0{,}020)m = 0{,}020\ m \tag{3.12}$$

da mesma forma, calcula-se o valor máximo possível para a junta 2 ($d_{2_{max\zeta}}$) considerando-se os amortecedores:

$$d_{2_{max\zeta}} = \left(d_{2_{max}} - d_{\zeta}\right) = (0{,}450 - 0{,}020)m = 0{,}430\ m \tag{3.13}$$

Como no sistema de coordenadas adotado para o robô por Sarmanho, 2014, o início da junta ($d_{2_{min\zeta}}$) fica localizado a uma distância de *0,263 m* do centro do sistema de coordenadas adotado por Missiaggia, 2014, para obter o valor máximo da coordenada *z* (z_{max}), deve-se somar este valor ao valor máximo calculado para a junta 2 com amortecimento ($d_{2_{max\zeta}}$), como pode ser observado na Equação (3.14):

$$z_{max} = \left(d_{2_{max\zeta}} + 0{,}263\right) m = (0{,}430 + 0{,}263)\ m = 0{,}693\ m \tag{3.14}$$

Da mesma forma, calcula-se o valor mínimo da coordenada *z* (z_{min}):

$$z_{min} = \left(d_{2_{min\zeta}} + 0{,}263\right) m = (0{,}020 + 0{,}263)\ m = 0{,}283\ m \tag{3.15}$$

Conforme já mencionado, no caso da aplicação *pick-and-place* proposta neste trabalho, o efetuador encontra-se posicionado na direção vertical com a garra para baixo para pegar a peça pelo seu topo (Figura 3.4). Desta forma, ao determinar o valor máximo da coordenada *z* (z_{max}), deve-se levar em consideração o comprimento da garra de *0,227 m* (Tabela 3.2). Recalcula-se, assim, o valor mínimo da coordenada *z* (z_{min}) e seu valor máximo (z_{max}) por meio das equações (3.16) e (3.17), respectivamente:

$$z_{max} = \left(d_{2_{max\zeta}} + 0{,}263 - 0{,}227\right) m = (0{,}430 + 0{,}263 - 0{,}227)\ m = 0{,}466\ m \tag{3.16}$$

$$z_{min} = \left(d_{2_{min\zeta}} + 0{,}263 - 0{,}227\right) m = (0{,}02 + 0{,}263 - 0{,}227)\ m = 0{,}056\ m \tag{3.17}$$

O cálculo dos valores máximos de *x* e *y* dependem da composição do posicionamento das juntas 1, 3 e 4. A verificação do atendimento das posições das juntas às restrições do volume de trabalho são realizadas por meio do cálculo das posições de cada junta obtidas por

meio da cinemática inversa a partir dos pontos gerados para uma determinada trajetória. Para isso, é preciso determinar os limites de cada uma das juntas, como realizado, por exemplo, para a junta 2 por meio das equações (3.12) e (3.13).

Devido à construção física do robô, o valor mínimo que a junta 3 consegue atingir ($d_{3_{min}}$) é de *0 m* (Tabela 3.1). Considerando-se os amortecedores nas juntas, o valor mínimo da junta 3 é pode ser calculado por:

$$d_{3_{min\zeta}} = \left(d_{3_{min}} + d_{\zeta}\right) = (0 + 0{,}02)m = 0{,}02\ m \tag{3.18}$$

Da mesma forma, calcula-se o valor máximo possível para a junta 3 ($d_{3_{max\zeta}}$) considerando-se os amortecedores:

$$d_{3_{max\zeta}} = \left(d_{3_{max}} - d_{\zeta}\right) = (0{,}300 - 0{,}020)m = 0{,}280\ m \tag{3.19}$$

Quanto à junta 4, o valor mínimo ($\theta_{4_{min}}$) é de *-1,963 rad* (Tabela 3.2). Considerando-se o deslocamento causado pelos amortecedores na junta de $\theta_{\zeta_4} = 0{,}341\ rad$ [Sarmanho, 2014], o valor mínimo da junta 4 é dado por:

$$\theta_{4_{min\zeta}} = \left(\theta_{4_{min}} + \theta_{\zeta_4}\right) = (-1{,}963 + 0{,}341)\ rad = -1{,}622\ rad \tag{3.20}$$

De forma semelhante, calcula-se o valor máximo possível para a junta 4 ($\theta_{4_{max\zeta}}$) considerando-se os amortecedores:

$$\theta_{4_{max\zeta}} = \left(\theta_{4_{max}} - \theta_{\zeta_4}\right) = (1{,}963 - 0{,}341)\ rad = 1{,}622\ rad \tag{3.21}$$

Quanto à junta 5, o valor mínimo ($\theta_{5_{min}}$) é de *-2,35619 rad* (Tabela 3.2). Considerando-se o deslocamento causado pelos amortecedores na junta de $\theta_{\zeta_5} = 0{,}096\ rad$ [Sarmanho, 2014], o valor mínimo da junta 5 é dado por:

$$\theta_{5_{min\zeta}} = \left(\theta_{5_{min}} + \theta_{\zeta_5}\right) = (-2{,}356 + 0{,}096)\ rad = -2{,}260\ rad \tag{3.22}$$

da mesma forma, calcula-se o valor máximo possível para a junta 5 ($\theta_{5_{max\zeta}}$) considerando os amortecedores pode, por sua vez, ser calculado por:

$$\theta_{5_{max\zeta}} = \left(\theta_{5_{max}} - \theta_{\zeta_5}\right) = (2{,}356 - 0{,}096)rad = 2{,}260\ rad \tag{3.23}$$

Quanto à junta 1, o valor mínimo ($\theta_{1_{min}}$) é de *-2,454281 rad* (Tabela 3.2). Considerando-se o deslocamento causado pelo amortecedor na junta de $\theta_{\zeta_1} = 0{,}218\ rad$ [Sarmanho, 2014], o valor mínimo da junta 1 é dado por:

$$\theta_{1_{min\zeta}} = \left(\theta_{1_{min}} + \theta_{\zeta_1}\right) = (-2{,}454 + 0{,}218)rad = -2{,}236\ rad \tag{3.24}$$

da mesma forma, calcula-se o valor máximo possível para a junta 1 ($\theta_{1_{max\zeta}}$) considerando os amortecedores é calculado por:

$$\theta_{1_{max\zeta}} = \left(\theta_{1_{max}} - \theta_{\zeta_1}\right) = (2{,}454 - 0{,}218)rad = 2{,}236\ rad \tag{3.25}$$

Como já mencionado, os valores mínimos e máximos calculados são utilizados para verificar se os pontos gerados estão dentro do espaço de trabalho do robô, a partir dos valores das coordenadas de junta correspondentes aos pontos gerados. Na Tabela 3.3 são apresentados os limites das juntas considerando os amortecedores.

Tabela 3.3- Valores limites das juntas do manipulador considerando o amortecimento

Junta	Valores Limites	
	Mínimo	Máximo
1	-2,236 *rad (-128°)*	2,236 *rad (128°)*
2	*0,020*	*0,430 m*
3	*0,020*	*0,280 m*
4	*-1,622 rad (-93°)*	*1,622 rad (93°)*
5	*-2,260 rad (-129,5°)*	*2,260 rad (129,5°)*

O raio de giro do semiarco (r) é calculado em função de um valor percentual (vp) da distância euclidiana (dif) no plano x, y entre o ponto inicial e final, de acordo com as equações (3.26) e (3.27).

$$r = \frac{vp\ dif}{100} \tag{3.26}$$

sendo,

$$dif = \sqrt{\left(x_f - x_i\right)^2 + \left(y_f - y_i\right)^2} \tag{3.27}$$

O programador pode alterar o valor do raio de giro do semiarco, alterando a porcentagem desejada, por exemplo, 10%, 20%, 30%, etc. Para obter o valor de z_a (que é o mesmo de z_d), subtrai-se do valor máximo de z (z_{max}) o valor do raio (r). Logo, têm-se:

$$z_a = z_d = z_{max} - r \tag{3.28}$$

$$z_b = z_c = z_{max} \tag{3.29}$$

O valor de p deve estar entre 5% e 50%, e, caso o valor do raio seja maior do que o valor da diferença entre z_a e z_i, uma mensagem de erro é exibida, solicitando ao usuário a escolha de uma porcentagem menor, já que a coordenada z do ponto inicial da trajetória circular deve ficar acima da coordenada z do ponto inicial.

Por meio da Figura 3.7, pode-se verificar as seguintes correspondências:

$$x_a = x_i \tag{3.30}$$

$$x_d = x_f \tag{3.31}$$

$$y_a = y_i \tag{3.32}$$

$$y_d = y_f \tag{3.33}$$

Assim, resta ainda calcular os valores de x_b, y_b, x_c e y_c. Para tanto, uma vez calculada a distância euclidiana entre os pontos inicial e final, obtém-se o valor do ângulo δ formado entre a reta que passa por ambos os pontos e o eixo x:

$$\delta = cos^{-1}\left(\frac{\sqrt{(x_f - x_i)^2}}{dif}\right) \tag{3.34}$$

A projeção do arco no eixo x pode ser calculada usando as equações (3.35) e (3.36) ou por meio das equações (3.37) e (3.38), levando-se em conta os casos definidos pela direção do vetor entre o valor final (x_f) e o inicial (x_i). Assim, se $x_f > x_i$, então:

$$x_b = x_a + r\,cos\,\delta \tag{3.35}$$

$$x_c = x_d - r\,cos\,\delta \tag{3.36}$$

Por outro lado, quando $x_f \leq x_i$:

$$x_b = x_a - r\,cos\,\delta \tag{3.37}$$

$$x_c = x_d + r\,cos\,\delta \tag{3.38}$$

Da mesma forma, obtém-se y_f e y_d através das equações (3.39) e (3.40) (ou, alternativamente, das equações (3.41) e (3.42)):

Se $y_f > y_i$, então:

$$y_b = y_a + r\,sen\,\delta \tag{3.39}$$

$$y_c = y_d - r\,sen\,\delta \tag{3.40}$$

Senão:

$$y_b = y_a - r\,sen\,\delta \tag{3.41}$$

$$y_c = y_d + r\,sen\,\delta \tag{3.42}$$

Com todas as incógnitas necessárias calculadas, para obter a trajetória completa, chama-se as rotinas que definem cada parte da trajetória:

1. *Go Vertical To Position* entre os pontos x_i, y_i, z_i e x_a, y_a, z_a;
2. *Go Circular To Position* entre os pontos x_a, y_a, z_a e x_b, y_b, z_b;
3. *Go Horizontal To Position* entre os pontos x_b, y_b, z_b e x_c, y_c, z_c;
4. *Go Circular To Position* entre os pontos x_c, y_c, z_c e x_d, y_d, z_d;
5. *Go Vertical To Position* entre os pontos x_d, y_d, z_d e x_f, y_f, z_f.

Finalmente, como o comando gera uma matriz de pontos, basta concatenar as matrizes para gerar a matriz completa da trajetória. Conforme já mencionado, o ângulo θ_4 não varia, assim:

$$\theta_{4_i} = \theta_{4_f} = -1{,}571\, rad \tag{3.43}$$

Para determinar os pontos intermediários entre o ângulo θ_{5_i} e θ_{5_f} adotou-se o seguinte procedimento:

Se $\theta_{5_\mathrm{i}} = \theta_{5_f}$, todos os valores dos ângulos θ_{5_n} serão iguais nos *n* trechos da trajetória, ou seja,

$$\theta_{5_1} = \theta_{5_2} = \theta_{5_3} = \theta_{5_4} = \theta_{5_5} = \theta_{5_6} = \theta_{5_\mathrm{i}} = \theta_{5_f} \tag{3.44}$$

Senão,

$$\theta_{5_1} = \theta_{5_\mathrm{i}} \tag{3.45}$$

$$\theta_{5_2} = \theta_{5_\mathrm{i}} + \frac{\theta_{5_f} - \theta_{5_\mathrm{i}}}{5} \tag{3.46}$$

$$\theta_{5_3} = \theta_{5_\mathrm{i}} + 2\frac{\theta_{5_f} - \theta_{5_\mathrm{i}}}{5} \tag{3.47}$$

$$\theta_{5_4} = \theta_{5_\mathrm{i}} + 3\frac{\theta_{5_f} - \theta_{5_\mathrm{i}}}{5} \tag{3.48}$$

$$\theta_{5_5} = \theta_{5_\mathrm{i}} + 4\frac{\theta_{5_f} - \theta_{5_\mathrm{i}}}{5} \tag{3.49}$$

$$\theta_{5_6} = \theta_{5_\mathrm{i}} + 5\frac{\theta_{5_f} - \theta_{5_\mathrm{i}}}{5} = \theta_{5_\mathrm{f}} \tag{3.50}$$

Por meio da Figura 3.10, pode-se observar um exemplo no qual uma trajetória é gerada pela rotina "*pick-and-place*" com um raio de 10% da distância euclidiana entre os dois pontos projetados no plano x, y.

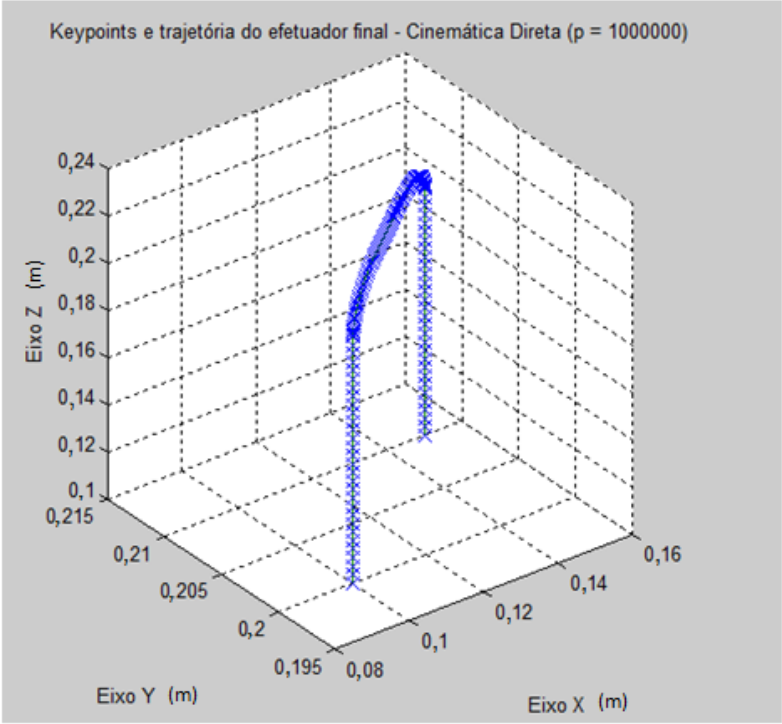


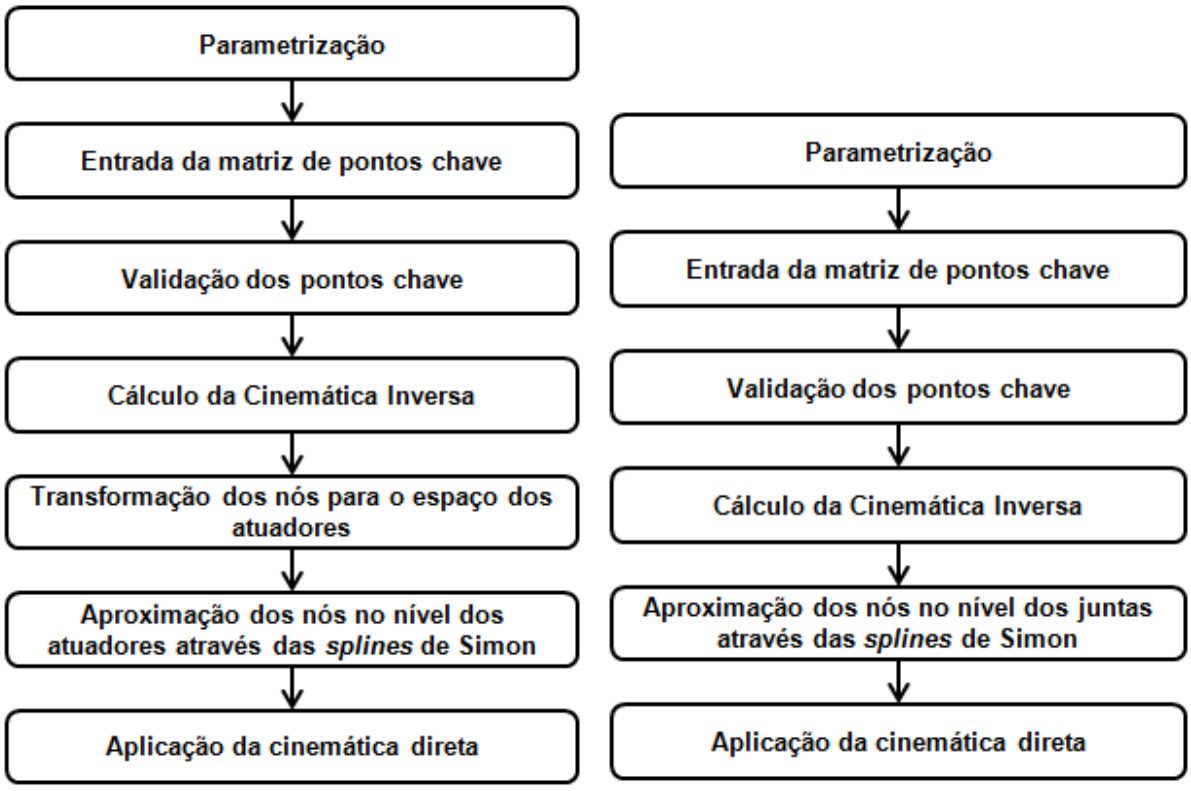
Figura 3.10 – Resultado de uma trajetória tridimensional gerada pela rotina "*pick-and-place*"

3.5 Integração dos programas

Conforme mencionado, no processo de integração do algoritmo de planejamento de trajetórias proposto por Missiaggia, 2014 com o sistema de controle dos atuadores proposto por Sarmanho, 2014, verificou-se a existência de diferenças nos valores dos parâmetros da tabela de Denavit-Hartenberg do robô pneumático utilizados pelos autores. Assim, as matrizes de transformação homogênea geradas para o robô pneumático nos dois sistemas resultaram diferentes. Estudando ambos os sistemas, foi possível identificar as causas das diferenças do valores do parâmetro d para os elos 1 e 5. No caso do elo 1, a diferença é devida aos pontos de referência escolhidos como origem do sistema de coordenadas do robô nos dois algoritmos (Figura 3.8). Conforme já apresentado, adotou-se o parâmetro d apresentado na Tabela 3.2 para o elo 1, alterando-se, assim, o programa de planejamento de trajetórias para este valor de parâmetro. No caso do elo 5, a diferença deu-se em relação à mudança da posição de referência na garra do efetuador final do robô. No trabalho de Missiaggia, 2014, foi considerada a referência final na ponta da garra utilizando o valor do seu comprimento total igual a *0,2273 m*, já que seu programa simulava todo espaço de trabalho do robô. No trabalho

de Sarmanho, 2014, a referência foi considerada na parte interna da garra, resultando em um comprimento igual a *0,1883 m*. Dessa forma, alterou-se também o programa de planejamento de trajetórias para compensar esse desvio.

Outras alterações foram necessárias. No código de Missiaggia, 2014, os cálculos são realizados com a entrada das coordenadas em espaço cartesiano do robô, havendo transformação para o espaço de juntas e em seguida para o espaço de atuadores. Isso foi feito dessa maneira porque os controladores utilizados trabalhavam diretamente com as coordenadas dos pontos no espaço dos atuadores. Contudo, Sarmanho, 2014, utilizou coordenadas no espaço das juntas para implementar os algoritmos ocasionando a necessidade de alterar o programa para que a interpolação fosse realizada no espaço das juntas. Na Figura 3.11 estão apresentados dois fluxogramas que permitem visualizar as alterações introduzidas na estratégia de geração das trajetórias.



(a) Missiaggia (b) Sarmanho

Figura 3.11 – Processos de geração de trajetórias de acordo com o controlador utilizado

Observa-se que a etapa final (cinemática direta) do processo descrito na Figura 3.11 (a) não é necessária para a realização do controle do manipulador, tendo sido introduzida por constituir uma ferramenta para avaliar as trajetórias obtidas antes de submetê-las à execução pelo robô, pois permite, uma vez de posse dos vetores que determinam a trajetória de cada atuador, visualizar a trajetória tridimensional que será realizada posteriormente pelo efetuador.

3.6 Sistema de visão computacional

O sistema de visão computacional é aquele que gera a entrada de dados para o algoritmo de planejamento de trajetórias. Optou-se, propositalmente, por apresentá-lo por último, já que para uma melhor compreensão, faz-se necessário compreender o funcionamento do robô no qual o método será aplicado.

Todos os algoritmos desenvolvidos neste trabalho foram desenvolvidos aplicando o software Matlab®.

3.6.1 Algoritmo de identificação da posição, dimensão e orientação de peças

Geralmente, em um sistema de visão, é necessário realizar inicialmente uma calibração, ou seja, correlacionar as coordenadas de pixel da câmera com as coordenadas cartesianas do robô.

Essa calibração deve ser realizada nos seguintes casos:

- no início do processo;
- toda vez que for trocado o tipo de peça a ser manipulada pelo robô, uma vez que, cada peça com formato diferente da peça anterior, gera uma tabela de comparação diferente;
- em casos em que se mova a mesa ou a câmera.

As etapas seguintes do sistema de visão, dependem de cada tipo de sistema, já que outros métodos de visão computacional podem ser aplicados, dependendo da necessidade.

No presente estudo, as etapas podem ser observadas na Figura 3.12.

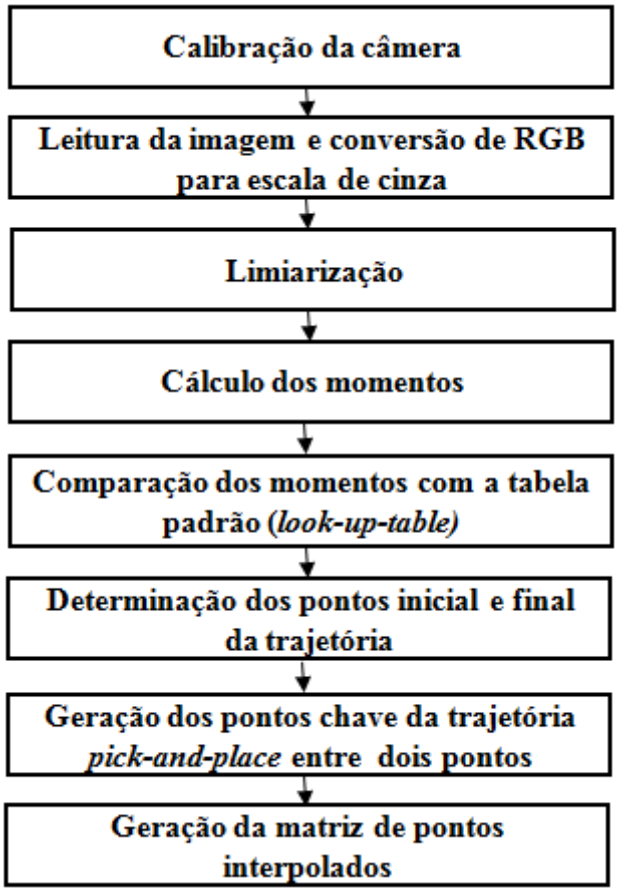


Figura 3.12 – Metodologia do algoritmo do sistema de visão

3.6.1.1 Calibração da câmera

A etapa de calibração da câmera consiste basicamente na obtenção da matriz de calibração ***P***, conforme a Equação (2.44), que correlaciona pixels da câmera com coordenadas do espaço tridimensional (também definida neste trabalho como "coordenadas mundo"). A maneira mais comum de se calibrar uma câmera é utilizar uma imagem impressa padrão. Normalmente, este padrão é composto por um ou dois planos ortogonais com padrões contrastantes nas suas faces, como, por exemplo, um tabuleiro de quadriculado. Então, pode-se considerar a origem do sistema de coordenadas do ambiente como sendo um dos cantos do padrão. Assim, as coordenadas mundo do padrão e suas respectivas projeções no plano da imagem podem ser conhecidos com grande exatidão [Tsai, 1986]. Desta forma, aplica-se a imagem padrão em uma superfície plana, assume-se que a origem do sistema de coordenadas mundo seja um ponto extremo da imagem padrão, com os eixos x_w *e* y_w associados a dois

lados ortogonais do padrão (conforme a Figura 3.13), e o eixo z_w tomado ortogonalmente ao plano x_w, y_w.

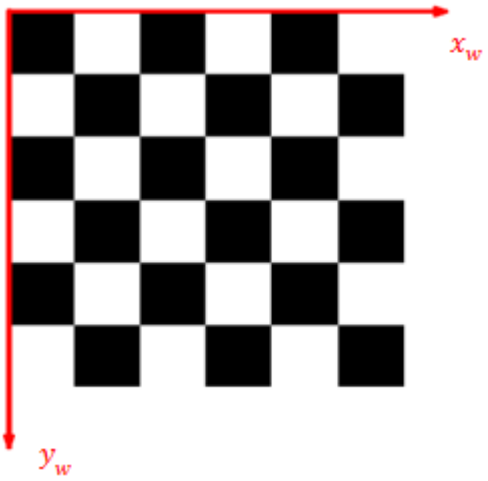


Figura 3.13 – Exemplo de definição do sistemas de coordenadas de um padrão de calibração

Em seguida, seleciona-se um conjunto de pontos nas interseções (quinas) dos quadrados internos, medindo suas coordenadas (x_w, y_w). Conforme visto na Seção 2.2.4, o número de pontos escolhidos deve ser maior do que 6, e, quanto mais pontos escolhidos, menor a possibilidade de ocorrência de erros. Porém, deve-se observar que como a calibração é realizada usando-se o Método dos Mínimos Quadrados (MMQ), se houver uma única medição inadequada todo o resultado será comprometido.

Após a medição, deve-se realizar a leitura da imagem a partir de uma câmera e obter as coordenadas em pixel dos pontos escolhidos. Com isto, monta-se um sistema linear do tipo $\boldsymbol{Ax} = \boldsymbol{b}$ (Equação (2.61)) e obtém-se, com sua resolução, a matriz de calibração de câmera $\boldsymbol{P}$. Assim, é possível obter qualquer coordenadas (x_w, y_w) do padrão, através de suas coordenadas em pixel (u_s, v_s) e da matriz $\boldsymbol{P}$.

Como o objetivo é obter pontos dentro do campo de visão da câmera e determinar suas coordenadas em relação ao robô, o centro do sistema de coordenadas dos pontos medidos deve coincidir com o centro do sistema de coordenadas do robô. Para isso, aplicou-se um adesivo padrão na forma de tabuleiro de xadrez no próprio robô, conforme está apresentado na Figura 3.14.

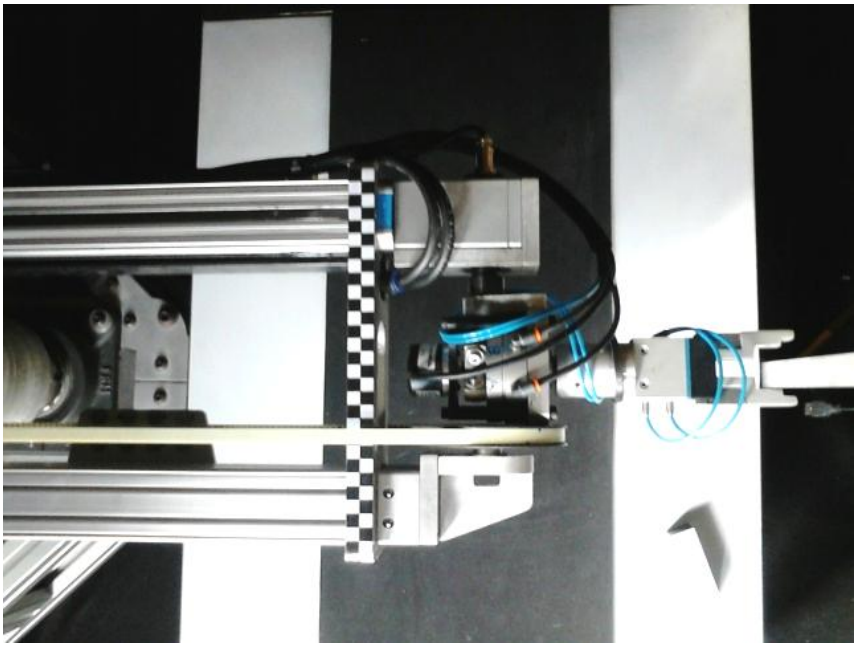


Figura 3.14 – Padrão de calibração aplicado no robô

Assim, colocando-se manualmente o robô em uma posição fixa ($d_1 = 0,\ d_2 = 0,\ d_3 = 0,\ \theta_4 = -90^{\circ},\ \theta_5 = 0^{\circ}$) e obtendo-se suas coordenadas nesta posição, é possível conhecer onde se encontra cada ponto do padrão em relação ao sistema de coordenadas do robô. Com isto, é possível calibrar-se a câmera em relação ao centro do sistema de coordenadas do robô, pois isso é suficiente para correlacionar as coordenadas da peça (em pixels) com as coordenadas do robô.

Para obter o ângulo em que a garra deve estar (θ_5) para captar a peça corretamente, optou-se por utilizar o método proposto por Grassi, 2005, onde é gerada uma tabela de comparação com os valores dos momentos para uma série de ângulos de giro da peça. Para isso, é preciso fornecer ao sistema um ângulo inicial de referência, ou seja, a situação em que a peça deve estar para ser capturada com um ângulo de 0°, ou seja, com a peça posicionada na direção 0° do robô. A seguir, coloca-se o robô na posição desejada ($d_1 = 0,\ d_2 = 0,\ d_3 = 0,\ \theta_4 = -90^{\circ},\ \theta_5 = 0^{\circ}$), encaixa-se a peça na garra (como pode ser observado na Figura 3.15), certificando-se que em qualquer distância de d_3 a peça seria capturada se $\theta_5 = 0^{\circ}$.

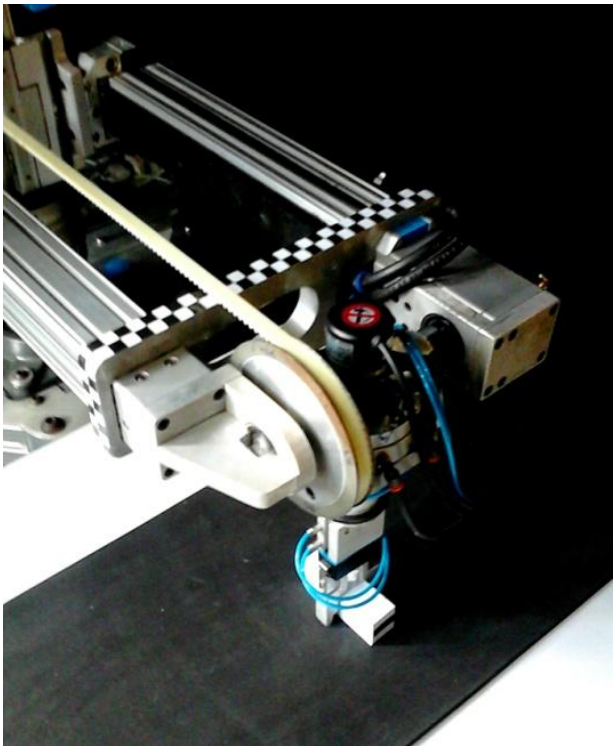


Figura 3.15 – Robô na posição de calibração com a garra a $\theta_5 = 0^{\circ}$.

Em seguida retira-se manualmente o robô cuidando para que ele não toque na peça a fim de que ela permaneça imóvel. Desta forma, pode-se obter a imagem da peça a 0° (conforme pode ser visto na Figura 3.16), a qual é utilizada como referência para a geração da tabela de comparação conforme descrito a seguir.

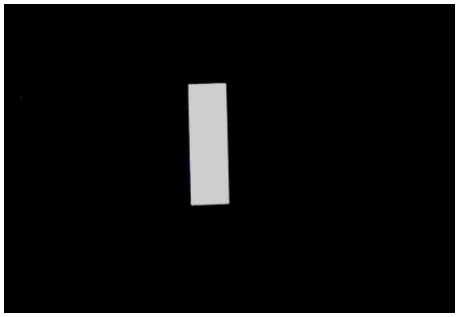


Figura 3.16 – Imagem obtida com a peça disposta a 0°

Desenvolve-se então um algoritmo que, a partir da imagem da peça a 0° (Figura 3.14 (b)), gera imagens rotacionadas em relação a esta posição. Deve-se informar o ângulo inicial, o ângulo final e a resolução (de quantos em quantos graus a imagem será rotacionada). Para cada imagem rotacionada, calcula-se os valores dos momentos que são armazenados em uma

matriz (tabela de comparação), a qual possui na primeira coluna os valores dos giros em graus; na segunda coluna, o valor dos momentos μ_{02}; na terceira coluna, o valor dos momentos μ_{20}, na quarta coluna, a relação μ_{02}/μ_{20}; e na última coluna, o valor dos momentos μ_{11} para cada grau.

Estas duas tarefas: geração da matriz de calibração ***P*** e geração da matriz ***LUT*** (*Look up Table**)*** é chamada etapa de calibração da câmera e é feita uma única vez no início do processo, devendo ser refeita apenas se a peça ou a localização da câmera forem alteradas.

3.6.1.2 Leitura da imagem e conversão de RGB para tons de cinza

Após esta etapa, realiza-se a leitura da imagem no formato RGB da peça localizada na posição em que se deseja identificar. O cálculo dos momentos é geralmente mais preciso quando realizado com uma imagem monocromática. Para isso, faz-se necessário transformar a imagem em tons de cinza, para depois limiarizá-la. Conforme apresentado na Seção 2.2.1.2, para a conversão da imagem RGB para tons de cinza, elimina-se a matiz e informação de saturação da cor e mantendo a luminância.

Segundo Turner, 1976, a luminância (ou brilho), representa o nível de energia pelo qual o olho é estimulado e é controlado pela soma das primárias. As cores de diferentes matizes e saturações podem ter os mesmos níveis de brilho ou níveis diferentes. Na ausência de matiz, as cores de brilho diferente aparecem como diferentes tons de cinza, como, por exemplo, na televisão em preto e branco. Estes aspectos do controle do brilho, e de casamento de matiz e saturação, indicam os meios pelos quais os sinais que representam as três características da imagem a cores podem ser codificados num sistema de transmissão.

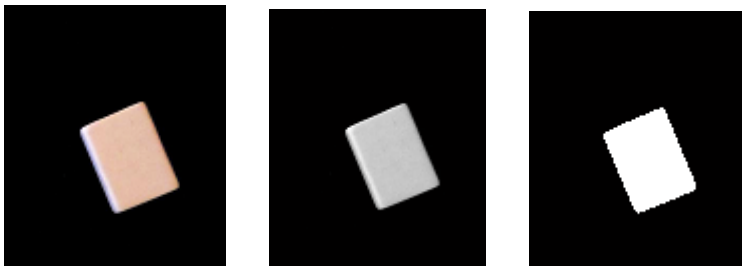
Para realizar a conversão de RGB para tons de cinza, varre-se a matriz tridimensional que representa a imagem em RGB e, transforma-se o vetor 1x3 (RGB) de cada posição da matriz (pixel) em um único valor entre 0 e 255 que irá representar aquela cor em escala de cinza. Segundo Gulati, 2006, para transformar esses três valores em um único valor que represente a sua luminância (*lum*), aplica-se a seguinte fórmula:

$$lum = 0{,}299R + 0{,}587G + 0{,}112B \tag{3.50}$$

sendo que R, G e B são os valores da quantidade de vermelho, verde e azul, respectivamente, de cada vetor da matriz que representa um pixel. Se os valores da resposta espectral dos canais R, B e B, forem somados, na razão de 0,299R, 0,587G e 0,112B, a característica resultante tem aproximadamente a mesma forma que a resposta de um canal em tons de cinza [Turner, 1976].

3.6.1.3 Limiarização da imagem

Com a imagem representada em tons de cinza, é possível realizar a limiarização da mesma, transformando-a em uma matriz binária, resultando em uma representação da imagem em branco e preto, conforme mostrado na Figura 3.17.



(a) RGB (b) Tons de cinza (c) Preto e branco

Figura 3.17 – Diferentes representações da mesma imagem

Para tal, realiza-se a varredura da matriz que representa a imagem em tons de cinza a qual possui valores de 0 a 255. Estipula-se então, o valor de um limiar, como, por exemplo, 127,5 e, substitui-se, por 0 (zero), todos os valores da matriz que forem inferior ao valor do limiar e, por 1 (um), todos os valores da matriz que forem superior ao valor do limiar.

A fim de limiarizar uma imagem em tons de cinza utilizando o limiar obtido pelo método de Otsu, primeiramente, deve-se calcular o histograma da imagem de entrada, conforme a Equação (2.1). Na Figura 3.18 é apresentada uma imagem em RGB de uma das peças utilizadas nos testes experimentais e, na Figura 3.19 é apresentado o histograma calculado para essa imagem.

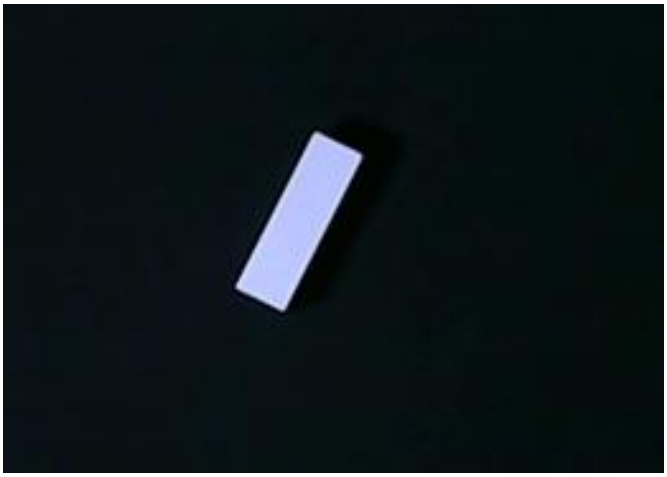


Figura 3.18 - Representação de uma imagem em RGB de uma das peças utilizadas nos testes

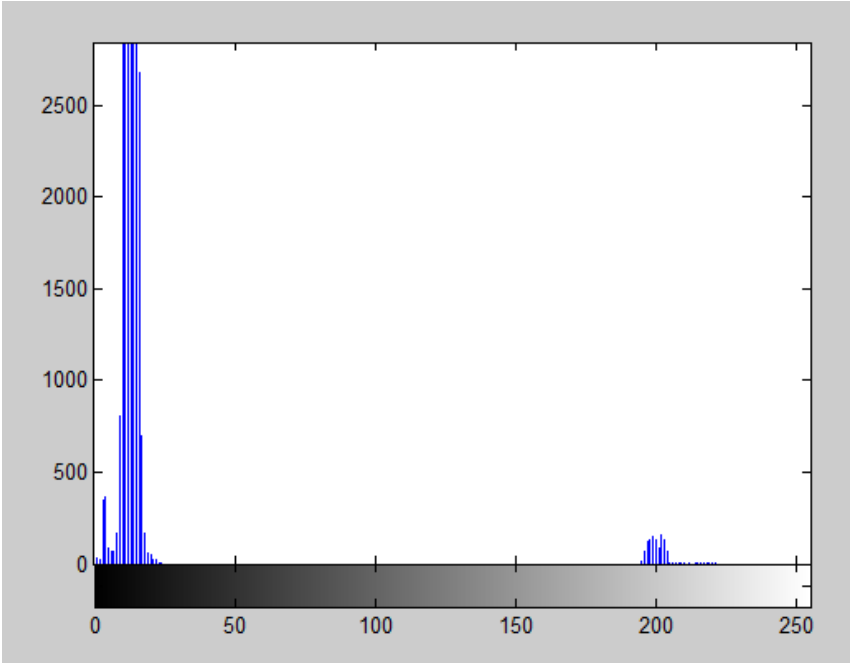


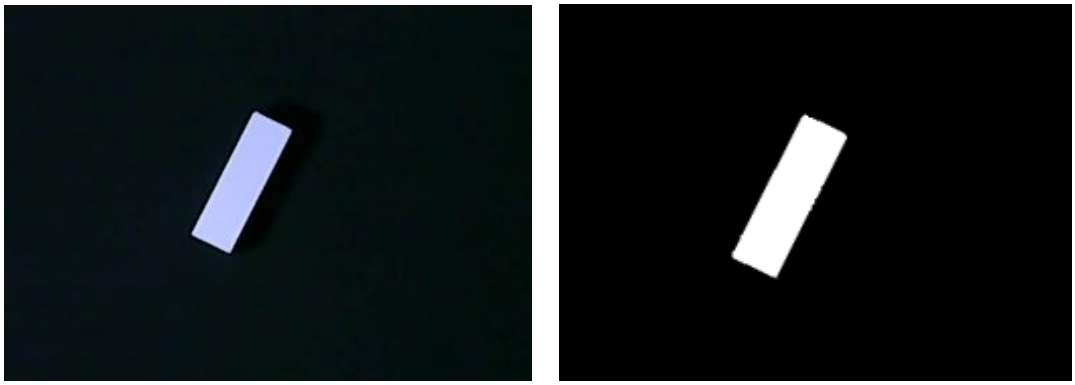
Figura 3.19 - Representação do histograma da imagem da Figura 3.18

O primeiro pico do histograma apresentado na Figura 3.19, representa a quantidade de pixels do fundo (cujas cores são próximas da cor preta), enquanto que o segundo pico do histograma representa a quantidade de pixels do objeto (cujas cores são próximas da cor branca). Percebe-se que, visualmente, o limiar que divide as duas regiões encontra-se próximo à 100, mas não é possível saber o valor exato.

Assim, aplica-se o método proposto por Otsu, 1979, apresentado na Seção 2.2.1.2, para obter o limiar automaticamente da seguinte forma:

- A partir dos valores calculados do histograma pela Equação (2.1), obtém-se h_i, para *i=0,1,2,...,L-1*;
- Calcula-se o somatório da Equação (2.2), obtendo-se $Pb_1(k)$, para *k=0,1,2,...,k*;
- Calcula-se o somatório da Equação (2.3), obtendo-se $Pb_2(k)$, para *k=k+1,k+2,...,L-1*;
- Calcula-se o somatório da Equações (2.7), obtendo-se $itm_1(k)$, para *k=0,1,2,...,k*;
- Calcula-se o somatório da Equação (2.8), obtendo-se $itm_2(k)$, para *k=k+1,k+2,...,L-1*;
- Calcula-se a variância intraclasses a partir da Equação (2.11), obtendo-se $\sigma_B^2(k)$, para *k=0,1,2,...,L-1*;
- Obtém-se o limiar (k^*) a partir da Equação (2.12).

Na Figura 3.20 (a), é apresentado um exemplo de imagem de uma das peças no ambiente de trabalho em RGB e, na Figura 3.20 (b), é apresentada a imagem limiarizada, obtida através da aplicação do método Otsu.



(a) RGB (b) Limiarizada

Figura 3.20 – Imagem obtida pela webcam antes e depois da limiarização

3.6.1.4 Cálculo dos momentos e obtenção da posição e orientação da peça

Com a imagem representada como uma matriz binária, é possível realizar o cálculo dos seus momentos. O primeiro objetivo é encontrar o centroide do objeto para determinar a

posição em que o manipulador deve chegar para pegar a peça. Para isso, foram calculados os momentos m_{00}, m_{10} e m_{01}, de acordo com as Equações (2.15) a (2.17). Desta forma é encontrado o centroide do objeto, conforme pode ser visto na Figura 3.21. Nessa figura, está destacado o ponto calculado para o centroide (em vermelho).

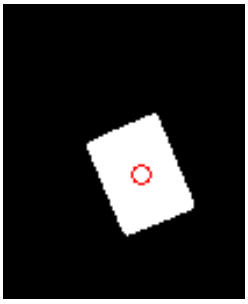


Figura 3.21 – Imagem em preto e branco com centroide encontrado destacado (em vermelho)

Calcula-se então os momentos centrais da imagem de μ_{02}, μ_{20} e μ_{11} por meio da Equação (2.18), comparando-se estes valores com os da tabela de comparação para determinar o ângulo de giro do objeto em relação à imagem padrão a 0°. Evidentemente, a precisão do resultado obtido e o tempo de duração da análise irão depender do número de imagens armazenadas na tabela de comparação.

Após obtidas a posição e o ângulo de giro da peça é gerada a trajetória através dos procedimentos anteriormente descritos na Seção 3.3.

3.6.2 Estrutura física de instalação do sistema de visão

Para operar o sistema de visão computacional juntamente com o robô pneumático, foi necessário projetar e executar um sistema composto por uma mesa para colocação das peças, e de um suporte para a câmera. A escolha da câmera se deu baseada em resultados de testes experimentais realizados com câmeras disponíveis no LAMECC/UFRGS. Foi escolhida a Webcam da marca Clone de 5 Megapixels (resolução interpolada) modelo 10028, apresentada na Figura 3.22.

Figura 3.22 – Webcam Clone 10028

Fonte: Atera, 2014.

Conforme já comentado, a resolução máxima informada pelo fabricante é de 2560 x 1920 pixels, 5 Megapixels. Esta resolução é interpolada, ou seja, com pixels extras gerados entre cada 2 pixels adjacentes. A resolução máxima real dessa câmera é de 640 x 480 pixels. Conforme também já comentado, a escolha de Webcam deu-se pelo fato de que a imagem pode ser manipulada online pelo Matlab®, o que não é diretamente realizável quando se utiliza câmeras manuais. Na Tabela 3.4 são apresentadas as características técnicas da câmera.

Tabela 3.4- Características técnicas da Webcam Clone 10028

Sistema Operacional	Windows XP / Vista
Resolução interpolada	5.0 Mega Pixels
Resolução de hardware	1.3 Mega Pixels
Cód. Produto	10028
Sensor de imagem	CMOS colorido
Formatos da imagem	RGB 24 / YUY2 2560 x 1920 pixels máx.
Velocidade de captura:	30fps (em 640 x 480)
Brilho	Automático
Alimentação	5 Vdc (via porta USB)
Foco	30mm ~ infinito
Ângulo de inclinação	90°
Ângulo de rotação	360°
Comprimento do cabo	1,40m

Após um estudo de identificação *in loco* do espaço de trabalho do robô, com o objetivo de evitar colisões, esboçou-se um suporte para a câmera posicionado acima do robô e que avançasse até o centro da mesa para que a peça ficasse dentro do campo de visão da

câmera. Este suporte também deve estar posicionado fora do espaço de trabalho do robô, devendo ficar à uma altura mínima aproximada de 1,3 m em relação ao chão. Visando a aplicações tridimensionais em trabalhos futuros, foi também projetado um suporte para uma segunda câmera. O esboço dos suportes das câmeras e da mesa está apresentado na Figura 3.23.

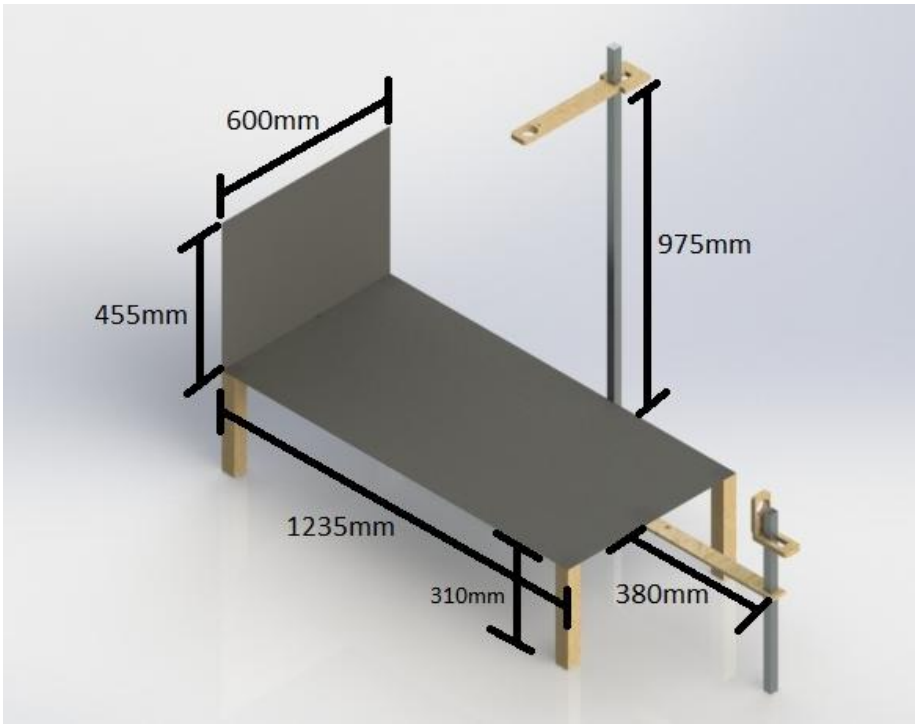


Figura 3.23 – Esboço dos suportes das câmeras e da mesa

A mesa construída para disposição das peças possui uma lâmina de borracha na cor preta que, deverá futuramente, dar lugar a uma esteira. Do lado oposto ao suporte da câmera lateral, foi disposto um fundo preto. Esta disposição facilita a diferenciação da peça do seu fundo quando executada a determinação da altura da peça. A disposição do sistema é tal que o campo de visão da câmera é maior que do que o da imagem necessária, ou seja, ao ser captada uma imagem, ela não abrange apenas a peça e seu fundo preto, porém, isso é facilmente resolvido por meio de um algoritmo que corta automaticamente as regiões que não são de interesse.

O sistema físico construído está apresentado na Figura 3.24, onde é possível visualizar o fundo de cor preta, o robô manipulador na sua versão atual, a mesa (com a lâmina de

borracha de cor preta), o suporte da câmera vertical, o suporte da câmera lateral e a peça a ser manipulada.

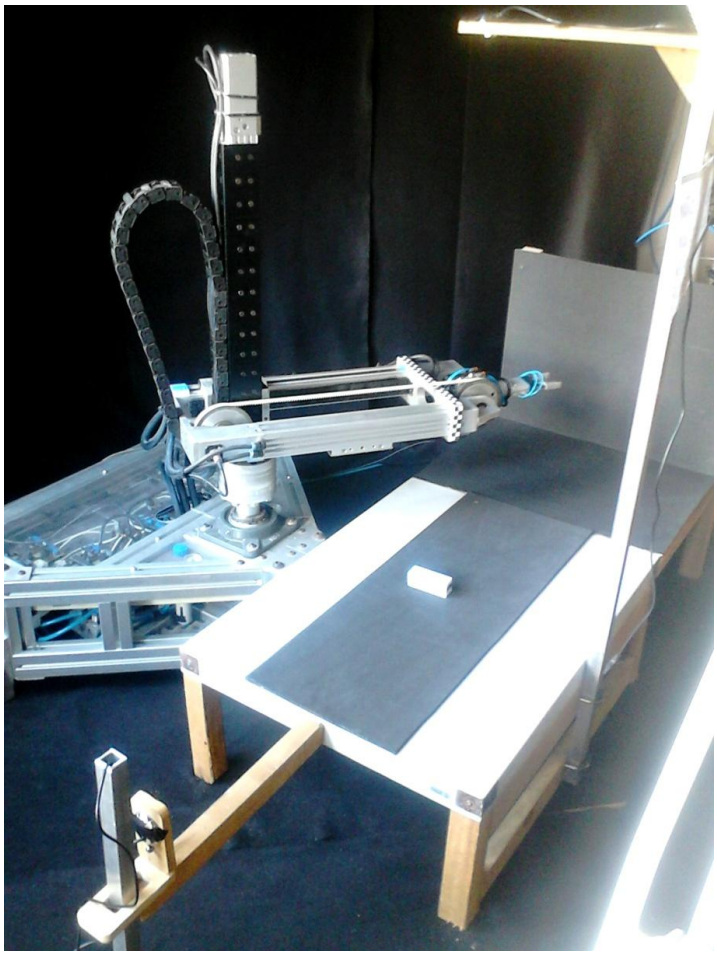


Figura 3.24 – Mesa e suporte construídos

Para manipulação, escolheu-se 3 peças (PA, PB e PC) em formato de paralelepípedo com dimensões aproximadas de:

- PA: 80,25 mm x 25,30 mm x 30,00 mm (Figura 3.25);
- PB: 60 mm x 25,30 mm x 30,00 mm;
- PC: 50 mm x 25,30 mm x 30,00 mm.

As medidas das peças foram obtidas por meio de um paquímetro da marca Mitutoyo, BH006190, com resolução de 0,05 mm. Na Figura 3.25, pode-se verificar um exemplo com as relações das medidas de uma das peças utilizadas para os testes.

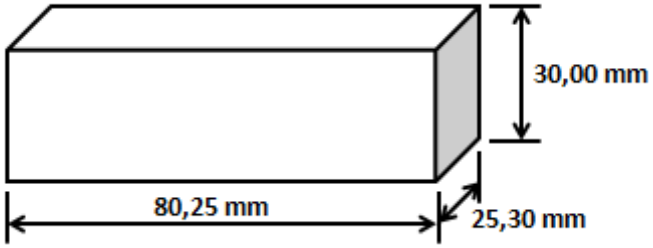


Figura 3.25 – Exemplo com as relações de medida de peças PA testadas

As peças são de acrílico e para adequar o contraste com o fundo, foram revestidas com papel adesivo branco opaco, conforme o exemplo apresentado na Figura 3.26.

Figura 3.26 – Exemplo de uma peça manipulada pelo robô

A imagem do padrão de calibração fixada no robô (Figura 3.27) foi confeccionada em uma impressora multifuncional de resolução de 5760x1440 DPI. Esta especificação significa que ela consegue pintar 5760 pontos na direção horizontal e 1440 na direção vertical em uma peça *25,40 mm* x *25,40 mm*. Assim, a cada 1 mm, o padrão pinta aproximadamente 226 pontos, ou seja, cada ponto representa aproximadamente *0,0044 mm* na direção horizontal. Da mesma forma, a cada *1 mm*, o padrão pinta aproximadamente 56 pontos, ou seja, cada ponto representa aproximadamente *0,017 mm* na direção vertical.

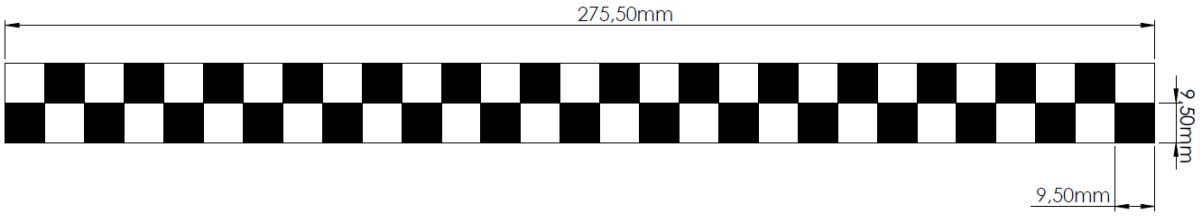


Figura 3.27 – Padrão impresso usado como referência

Através das medidas do padrão impresso, é possível determinar a relação pixel/milímetros. Sabe-se que cada quadrado do padrão mede *9,50* x *9,50 mm* e que, cada quadrado é representado na imagem com dimensões 16 x 16 pixels, conforme pode ser observado na Figura 3.28. A resolução do monitor onde a imagem é gerada é de 1366 x 768 pixels.

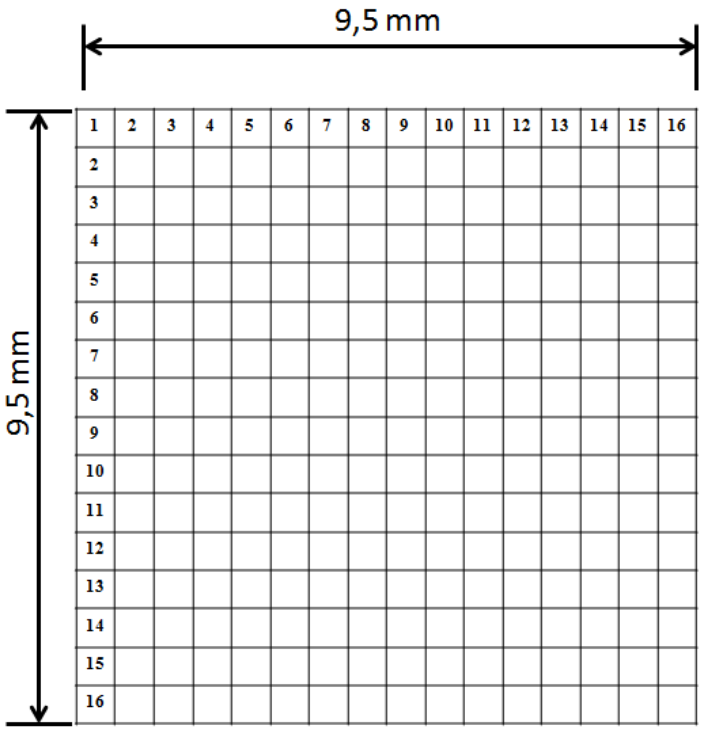


Figura 3.28 – Representação de um quadrado do padrão com dimensões em pixels e em milímetros

Logo, calcula-se que se *9,50 mm* correspondem 16 pixels, 1 pixel corresponde à aproximadamente *0,59 mm* e, da mesma maneira, 1 mm corresponde à aproximadamente 1,68 pixel. Porém, como na prática estas medidas são verificadas com o paquímetro anteriormente citado e, a resolução do paquímetro de *0,05 mm*, pode-se afirmar que 1 pixel corresponde à aproximadamente *0,60 mm*.

4 IMPLANTAÇÃO EXPERIMENTAL

Neste capítulo são abordados os principais aspectos da implantação experimental do sistema de visão computacional para o robô pneumático. Os equipamentos apresentados a seguir foram implementados no Laboratório de Mecânica e Controle da Universidade Federal do Rio Grande do Sul (LAMECC/UFRGS). Na Figura 4.1 está apresentado o sistema completo.

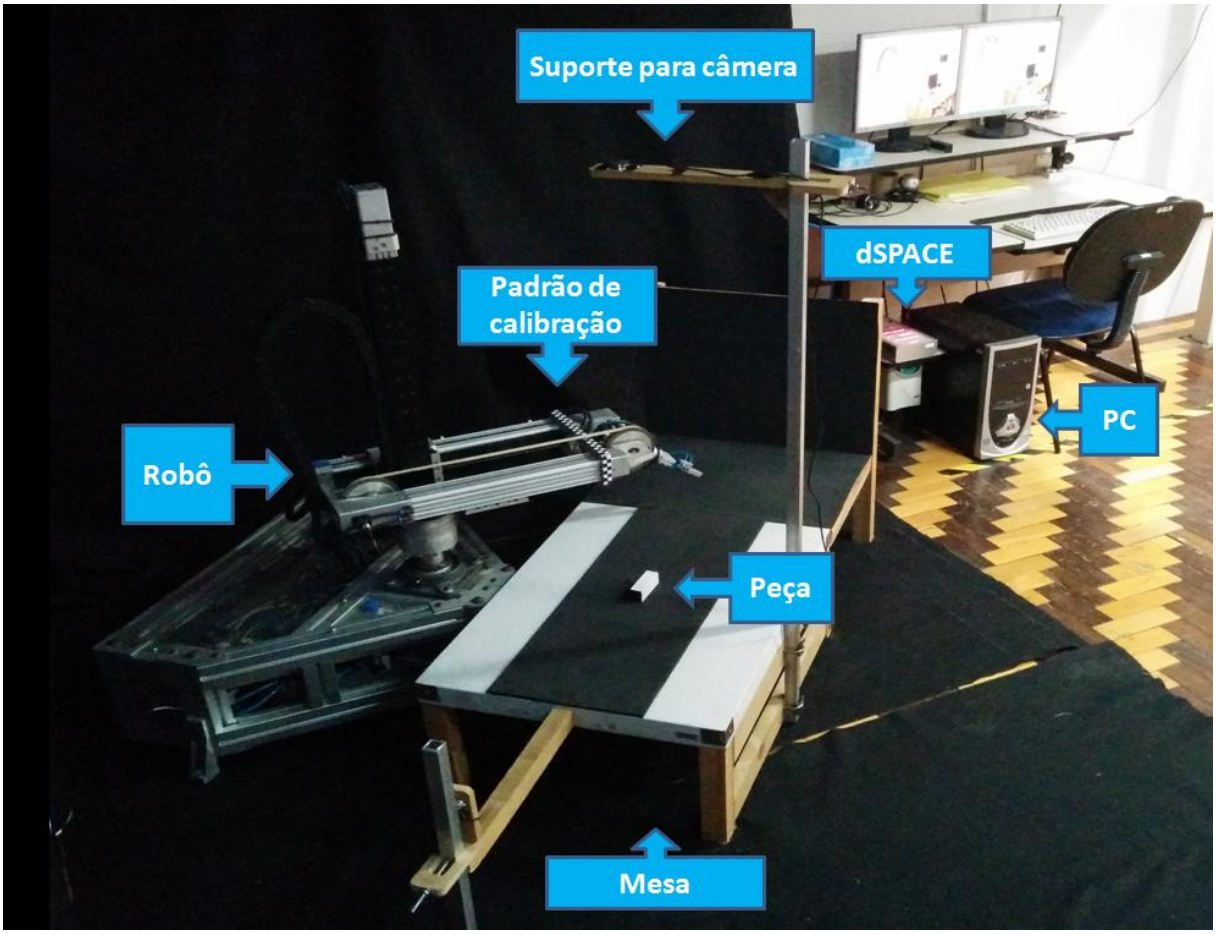


Figura 4.1 – Bancada de testes montada no LAMECC/UFRGS

O sistema experimental é composto: pelo robô pneumático, cujas informações detalhadas podem ser encontradas em Sarmanho, 2014; pela mesa e suporte das câmeras (Seção 3.6.2); pela plataforma de controle dSPACE® modelo DS-1104 (dSPACE, 2015); por um computador que hospeda a dSPACE®; pelos sistemas de sensoriamento e de alimentação

de energia elétrica e ar comprimido. Maiores detalhes sobre esses dispositivos podem ser encontrados em Cukla, 2012.

As placas dSPACE® permitem o desenvolvimento de algoritmos de controle comunicando-se diretamente com o pacote computacional Matlab-Simulink®. A placa DS-1104 é composta por uma interface de aquisição de dados e controle, associada a um pacote de software disponibilizado pelo proprietário. Esta placa apresenta um número de entradas e saídas, tanto digitais como analógicas, além de entradas para encoders (absolutos ou incrementais), saídas de controle PWM e portas dedicadas à comunicação. Além disso, o hardware da dSPACE® é acompanhado pelo software ControlDesktop®, através do qual é possível monitorar e modificar variáveis do sistema controlado, bem como gravar e exportar dados adquiridos através da dSPACE® (Rijo, 2013).

O microcomputador usado para hospedar a placa é do tipo Intel Pentium Core 2 Duo® com 2 GHz de processamento e 2 GB de memória RAM.

A seguir são apresentados os testes para verificação do funcionamento do sistema de visão computacional de forma independente, ou seja, a identificação da posição e da orientação de uma peça em na imagem. Em seguida, são apresentados os testes de integração com o código de geração de trajetórias e, por fim, os testes da integração com o sistema de controle do robô.

4.1 Sistema de visão computacional

Avaliou-se primeiramente o funcionamento do algoritmo de calibração. Para tanto, escolheu-se pontos arbitrários no padrão tipo tabuleiro de xadrez afixado ao robô e calculou-se suas coordenadas mundo (x_w, y_w). Posteriormente, selecionando os mesmos pontos na imagem, obteve-se suas coordenadas em pixel (u_ω, v_ω).

É necessário destacar que para obter-se as coordenadas em pixels (u_ω, v_ω) de cada ponto, deve-se primeiramente selecionar o pixel da imagem correspondente ao ponto. Ao executar o algoritmo de calibração da câmera, é exibida uma tela (Figura 4.2) com a imagem do robô com uma mensagem solicitando ao usuário que amplie a imagem no ponto desejado até que esta exiba discriminadamente seus pixels.

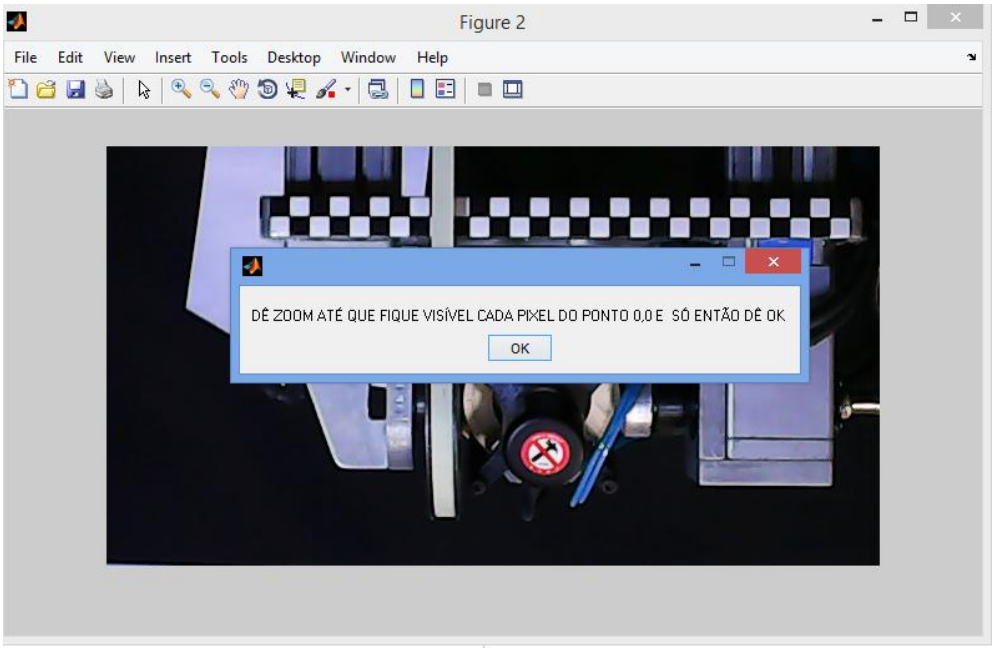


Figura 4.2 – Tela inicial da calibração

Após ampliar a imagem até a escala adequada, o usuário pode selecionar o pixel que melhor representa o ponto, conforme pode ser observado na Figura 4.3.

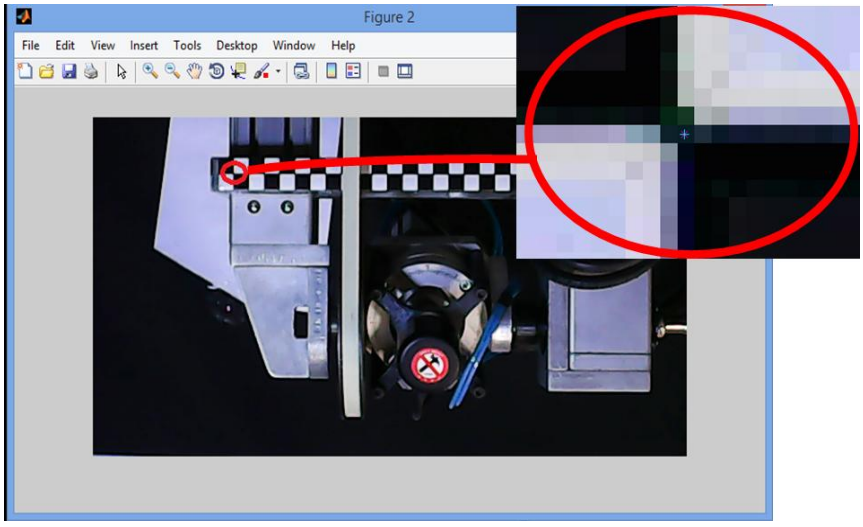


Figura 4.3 – Imagem do ponto selecionado a nível de pixel

Os pontos que devem ser selecionados são aqueles que correspondem ao encontro dos quadrados pretos com os brancos do padrão, pois é exatamente nestes pontos que obteve-se as medidas do padrão impresso. Na Figura 4.4, é possível observar, em vermelho, o ponto de encontro mencionado.

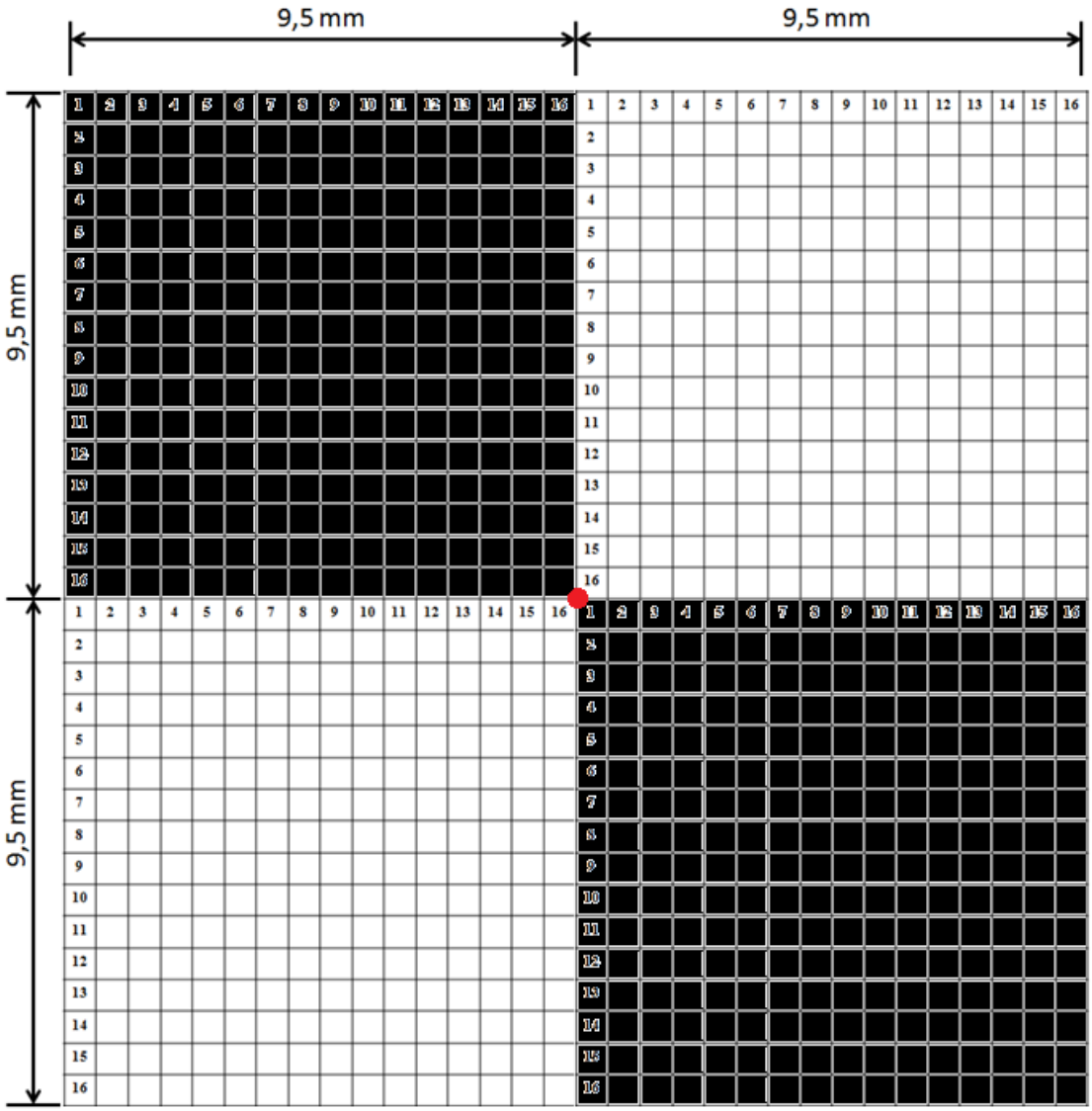


Figura 4.4 – Exemplo de uma parte do padrão à nível de pixel onde é mostrado, em vermelho, o ponto a ser selecionado

Nota-se que o ponto selecionado não se encontra inserido em nenhum pixel específico, mas sim, entre 4 pixels adjacentes. Porém, com o Matlab®, não é possível selecionar um ponto entre pixels. Ao clicar em tal ponto, é adotado qualquer um dos 4 pixels adjacentes ao invés da intersecção. Com isso, obtém-se um erro de posicionamento. Na Figura 4.5, pode ser observado um exemplo do ponto adotado pelo Matlab® (em vermelho) quando o usuário tenta selecionar o ponto de intersecção de quatro pixels adjacentes (destacados com o quadrado em azul).

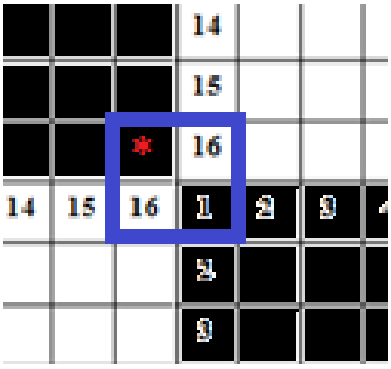


Figura 4.5 – Exemplo de pixel adotado pelo Matlab®

Uma forma de solucionar este problema é propositalmente selecionar o primeiro pixel dos 4 pixels adjacentes, conforme pode ser observado na Figura 4.6, e somar o valor de 1/2 pixel tanto na direção *x*, quanto na *y*. Como ao selecionar um pixel, o Matlab adota a posição central do pixel, ao somar 1/2 pixel na direção vertical e 1/2 pixel na direção horizontal, obtém-se o ponto da intersecção desejado.

Assim, o procedimento definido pelo algoritmo é o de o usuário selecionar como ponto de referência do sistema local de coordenadas (o pixel que corresponde à coordenada $(x_w = 0, y_w = 0)$ do padrão) não o ponto de intersecção desejado, mas sim o pixel acima e à esquerda do ponto desejado definido pelo ponto em vermelho na Figura 4.6.

Figura 4.6 – Exemplo de pixel a ser selecionado entre 4 pixels adjacentes

Ao adotar este pixel como referência, basta adicionar às suas coordenadas em pixel (u_s, v_s) o valor de 1/2 pixel, para que todos os outros pontos sejam posicionados de maneira correta, ou seja, conforme apresentado na Figura 4.4.

Assim, as coordenadas do ponto de referência (u_0, v_0) podem ser calculadas conforme segue:

$$u_0 = u_0 + 1/2 \tag{4.1}$$

$$v_0 = v_0 + 1/2 \tag{4.2}$$

Cada um dos pontos pode ser selecionado manualmente, como realizado com o ponto de referência, porém, além de trabalhoso, esta técnica estaria sujeita a erros de seleção dos pontos pelo usuário. Como, conforme visto na Seção 3.6.2, cada quadrado interno do padrão, possui dimensão de 16 x 16 pixels, foi desenvolvida uma estratégia para a seleção automática dos pontos adicionando (ou subtraindo) 16 pixels da coordenada do ponto anteriormente definido. Assim, o usuário necessita definir apenas o primeiro (o de referência) e o algoritmo fornece automaticamente os demais, como mostrado na Figura 4.7.

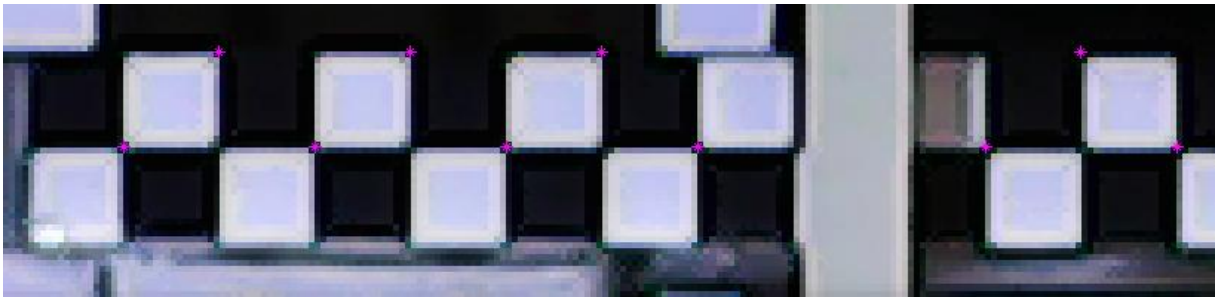


Figura 4.7 – Imagem com todos os pontos selecionados

Usando as coordenadas definidas conforme a estratégia proposta, é possível obter a matriz de calibração de câmera (***P***) (Equação 2.44). Para avaliar o funcionamento dessa estratégia de calibração, realizou-se um esse com outros pontos escolhidos aleatoriamente. Para tanto, uma vez calculada a matriz de calibração, escolheu-se pontos aleatórios na imagem e obteve-se as suas coordenadas em pixel, as quais, multiplicadas pela inversa da matriz ***P,*** fornecem as suas respectivas as coordenadas de mundo (x_w, y_w) (Equação 2.52). Essas coordenadas mundo podem, finalmente, ser comparadas com valores medidos na bancada. É necessário destacar que as coordenadas mundo (x_w, y_w) de cada ponto utilizadas nos equacionamentos foram definidas em relação ao ponto de referência (x_w =0, y_w =0) do próprio padrão impresso. Para correlacionar esses pontos com o sistema de coordenadas do robô, adotou-se uma referência associada a uma posição fixa do mesmo ($\theta_1 = 0\ rad,\ d_2 = 0$

m, $d_3 = 0{,}300\ m, \theta_4 = 0\ rad\ e\ \theta_5 = 0\ m$) e mediu-se com paquímetro e trena a distância do ponto de referência do padrão em relação ao centro de coordenadas do robô ($x_w = 0{,}12333\ m, y_w = 0{,}4045\ m$). Desta forma, de posse das coordenadas de um ponto qualquer em relação ao ponto de referência do padrão impresso, é possível determinar as coordenadas deste ponto em relação ao centro de coordenadas do robô.

Realizou-se 20 (vinte) testes onde escolheu-se aleatoriamente pontos do padrão de calibração. Comparou-se os valores das coordenadas mundo medidas com o paquímetro anteriormente mencionado (de resolução de 0,05 mm), com os valores calculados pelo algoritmo. Verificou-se que o algoritmo é eficaz na identificação das coordenadas mundo dos pontos de uma imagem em relação ao centro de coordenadas do robô, já que, em 100% dos testes realizados, fora capaz de fornecer os valores inteiros em *mm* iguais aos valores medidos pelo paquímetro, ou seja, com desvio igual a 0 mm.

A segunda etapa dos testes é constituída pela análise dos resultados do algoritmo de reconhecimento da posição e da orientação das peças. Para isso, seguiu-se o passo a passo do fluxograma apresentado na Figura 3.12 da Seção 3.6.1, com a leitura da imagem, transformação de RGB para tons de cinza e limiarização. Com a imagem limiarizada, realizou-se o método de cálculo dos momentos para obtenção do centroide e da orientação de objetos. Primeiramente, foi testado com imagens criadas via software (não se trata de objetos reais), conforme pode ser observado por meio da Figura 4.8.

Para cada imagem, gerou-se uma tabela de comparação de 0º a 360º com resolução de 1º considerando uma posição inicial como referência de 0º. Na Figura 4.8, estão apresentadas algumas imagens utilizadas para os testes em suas posições definidas como de referência (0º).

(a) retângulo (b) círculo (c) elipse

Figura 4.8 – Imagens criadas via software

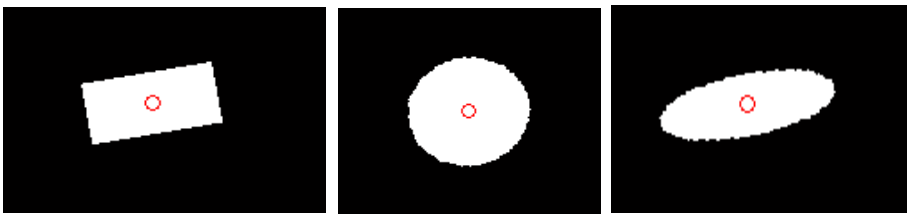
Em seguida, criou-se imagens rotacionadas utilizando um algoritmo desenvolvido especificamente para este fim. O programa permite que o usuário selecione a imagem padrão

e o ângulo de giro desejado, obtendo como saída a imagem rotacionada. Na Figura 4.9, podem ser observadas as imagens da Figura 4.8 rotacionadas em 10º por meio do algoritmo utilizado.

(a) retângulo (b) círculo (c) elipse

Figura 4.9 – Imagens rotacionadas em 10º

O uso desse algoritmo permite obter imagens rotacionadas que podem ser usadas para a montagem de um banco de dados (tabela de comparação) com as informações que permitem identificar o giro de uma peça a ser manipulada. O método de cálculo dos momentos descrito na Seção 2.2.1.3 foi aplicado com o objetivo de encontrar o centroide dos objetos, os quais podem ser vistos na Figura 4.10.



(a) retângulo (b) círculo (c) elipse

Figura 4.10 – Imagens com seus respectivos centroides

Uma vez conhecidos os centroides, é possível calcular os momentos centrais de ordem (p,q) e gerar a tabela de comparação para as figuras rotacionadas em 1º (de 0º a 360º).

Realizou-se 20 (vinte) testes com cada uma das três formas apresentadas na figura 4.8. O teste deu-se da seguinte forma: a partir da imagem a 0º criada via software, aplicou-se o algoritmo desenvolvido para gerar imagens rotacionadas com valores arbitrários de ângulos e comparou-se com o ângulo calculado pelo algoritmo baseado nos momentos da imagem. Verificou-se que o algoritmo é eficaz na identificação do giro das mesmas, tendo sido capaz, em 100% dos testes feitos, de fornecer o valor inteiro (em graus) mais próximo do valor testado.

O algoritmo desenvolvido para gerar imagens rotacionadas é baseado na operação de rotação nas transformações de matrizes apresentada na Seção 2.2.3.1.2. O objeto deve girar em torno do seu eixo central (centroide). Para isso, encontra-se seu centroide, realiza-se a translação do objeto levando seu centroide até a origem do sistema de coordenadas, efetua-se a rotação e, por último, efetua-se a translação oposta do objeto levando seu centroide até o ponto centroide anterior, conforme apresentada a sequencia de ilustrações da Figura 4.11.

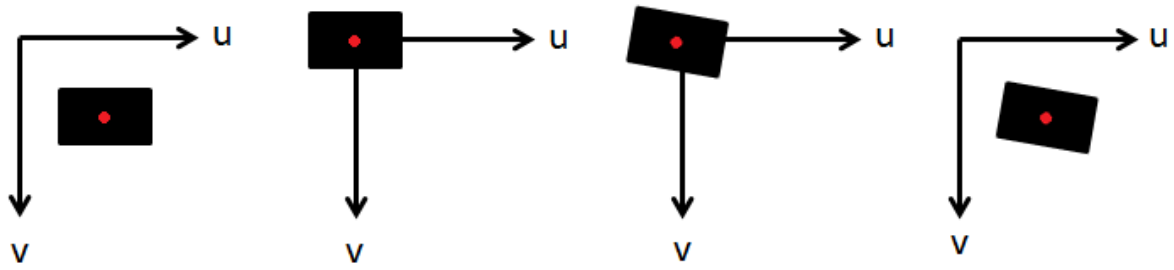


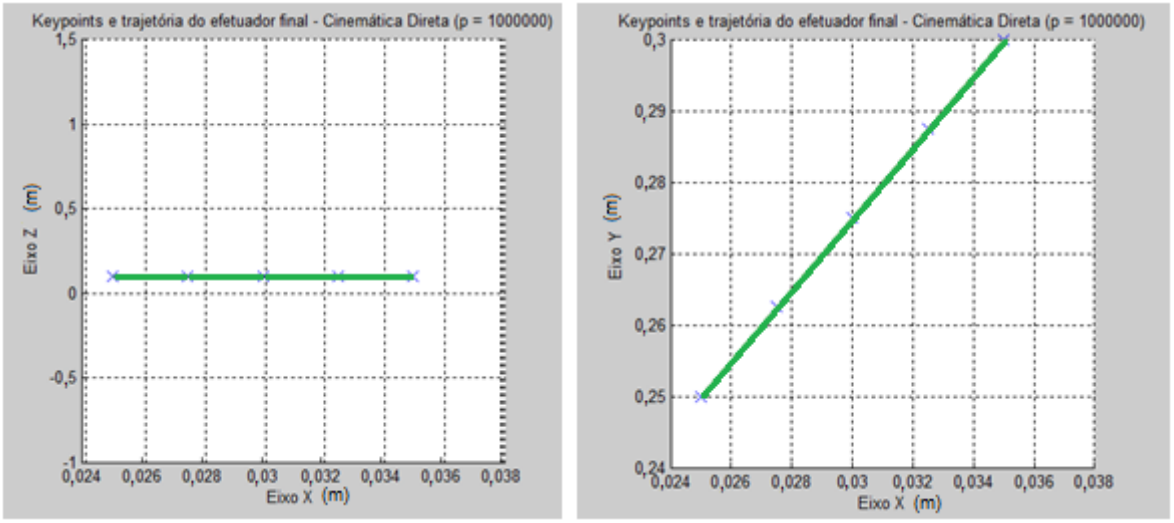
Figura 4.11 – Sequencia de operações de rotação de um objeto em torno do seu centroide

Foram, também, realizados 20 testes com imagens de fotografias de cada uma das três peças a serem manipuladas no ambiente de trabalho do robô, conforme apresentado na Seção 3.6.2. O teste deu-se da seguinte forma: a partir da imagem a 0º das peças no ambiente de trabalho do robô, aplicou-se o algoritmo desenvolvido para gerar imagens rotacionadas com valores arbitrários de ângulos e comparou-se com o ângulo calculado pelo algoritmo baseado nos momentos da imagem. Verificou-se que o algoritmo é eficaz na identificação do giro das três peças testadas, tendo sido capaz, em 100% dos testes feitos, de fornecer o valor inteiro (em graus) mais próximo do valor testado.

4.2 Integração com o algoritmo de planejamento de trajetórias

Esta etapa dos testes experimentais visou a integrar o sistema de visão computacional com o algoritmo de planejamento de trajetórias implementado por Missiaggia, 2014. Para que isso fosse possível, foi inicialmente necessário gerar os pontos chave da trajetória, de acordo com os procedimentos descritos na Seção 3.4. Conforme já mencionado na Seção 3.3, esses valores são armazenados em uma matriz, contendo informações das trajetórias de cada junta (posição, velocidade, aceleração e *jerk*).

A fim de validar método de geração de pontos, testou-se inicialmente cada trecho da trajetória *pick-and-place* de forma separada (Seção 3.4). Para tanto, foram escolhidos pontos iniciais e finais aleatórios dentro do espaço de trabalho do robô. A primeira rotina testada foi a "*GoHorizontalToPosition*" (Seção 3.4), a qual gera pontos que formam uma linha horizontal (paralela ao plano *xy*). Na Figura 4.12 está representada uma trajetória horizontal gerada com os seguintes parâmetros: $x_i = 0{,}025$ m, $y_i = 0{,}250\ m$, $z_i = 0{,}100\ m$, $x_f = 0{,}035\ m$, $y_f = 0{,}300\ m$, $\theta_{5_i} = 1{,}900\ rad$ e $\theta_{5_f} = 1{,}950\ rad$.



(a) plano *xz* (b) plano *xy*

Figura 4.12 – Trajetória horizontal (a) no plano *xz* e (b) no plano *xy*

Na Figura 4.13 são apresentados os gráficos da posição em relação ao tempo de cada junta para a trajetória horizontal.

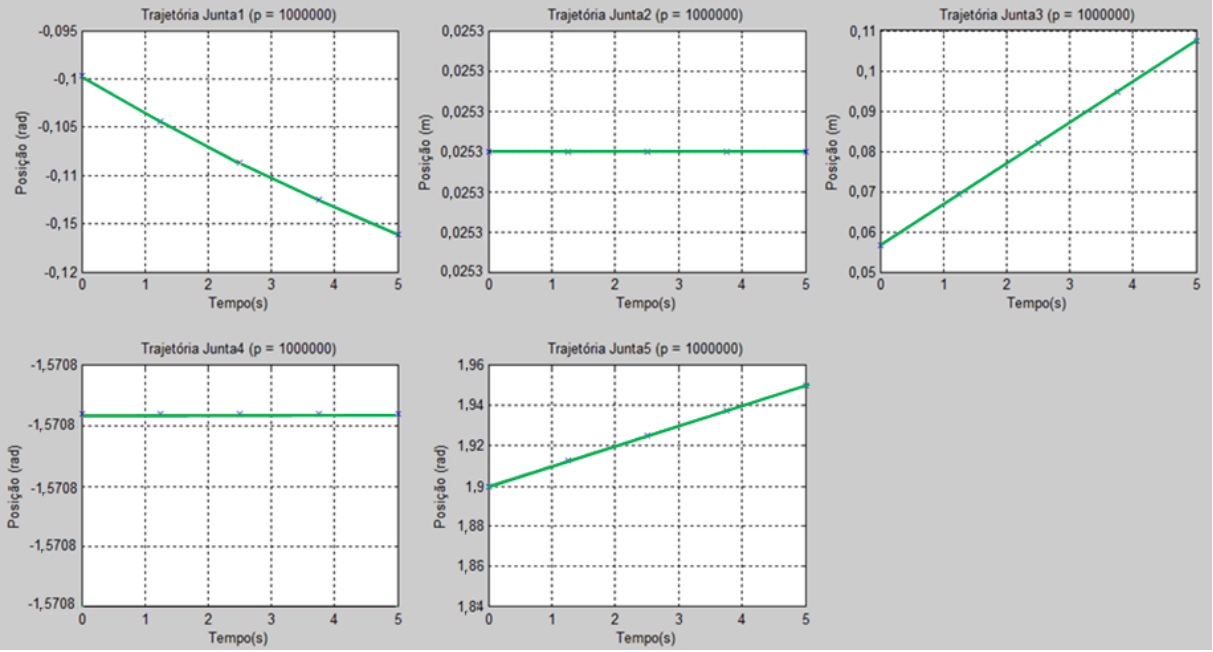


Figura 4.13 – Posição em relação ao tempo de cada junta em uma trajetória horizontal

Em seguida, testou-se a rotina "*GoVerticalToPosition*" (Seção 3.4) que gera pontos que, ao serem ligados, formam uma linha vertical (perpendicular ao plano *xy*). Na Figura 4.16 está apresentada uma trajetória vertical gerada com os seguintes parâmetros: $x_i = 0{,}025\ m$, $y_i = 0{,}250\ m$, $z_i = 0{,}100\ m$, $z_f = 0{,}150\ m, \theta_{5_i} = 1{,}900\ rad, \theta_{5_f} = 1{,}950\ rad$.

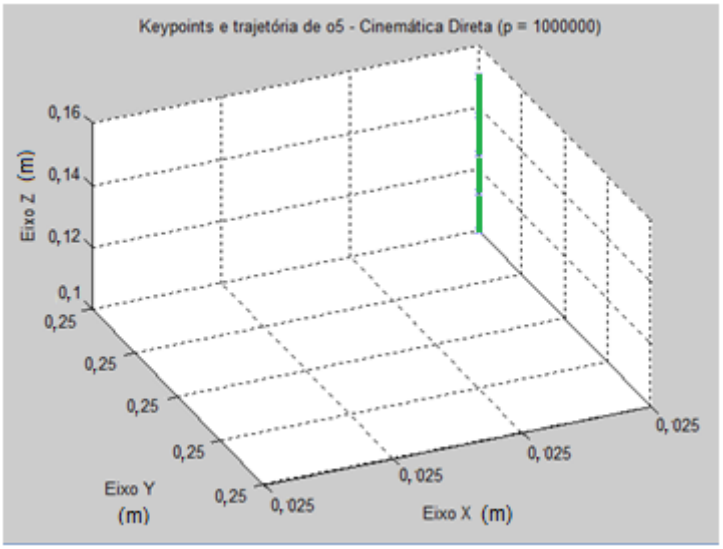


Figura 4.14 – Trajetória vertical no plano tridimensional

Na Figura 4.15 são apresentados os gráficos da posição em relação ao tempo de cada junta para a trajetória vertical.

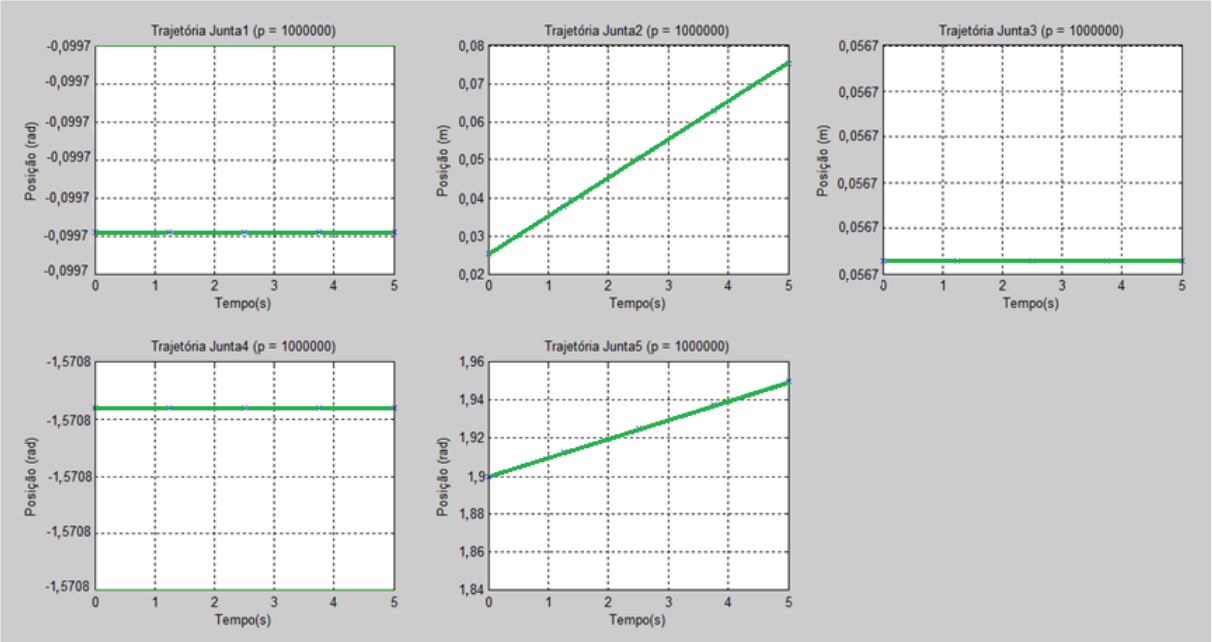


Figura 4.15 – Posição em relação ao tempo de cada junta em uma trajetória vertical

Em seguida, testou-se a rotina "*Go Circular To Position*" (Seção 3.4), a qual gera pontos que, ao serem ligados, formam semi arcos entre si. Na Figura 4.18 está apresentada uma trajetória circular gerada com os seguintes parâmetros: x_i=*0,1 m*, $y_i = 0{,}2\ m$, $z_i = 0{,}15\ m$, $z_f = 0{,}200\ m, x_f = 0{,}135\ m, y_f = 0{,}235\ m, \theta_{5_i} = 0\ rad, \theta_{5_f} = 0{,}100\ rad$.

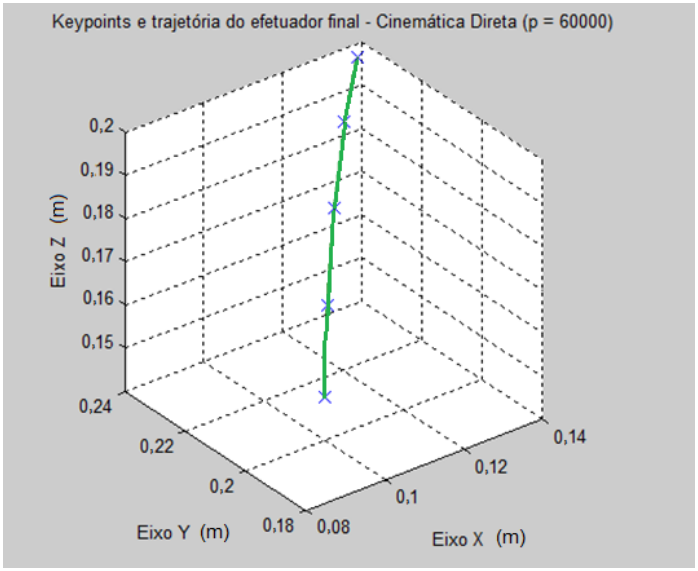


Figura 4.16 – Trajetória circular no plano tridimensional

Na Figura 4.17 são apresentados os gráficos da posição em relação ao tempo de cada junta para a trajetória circular.

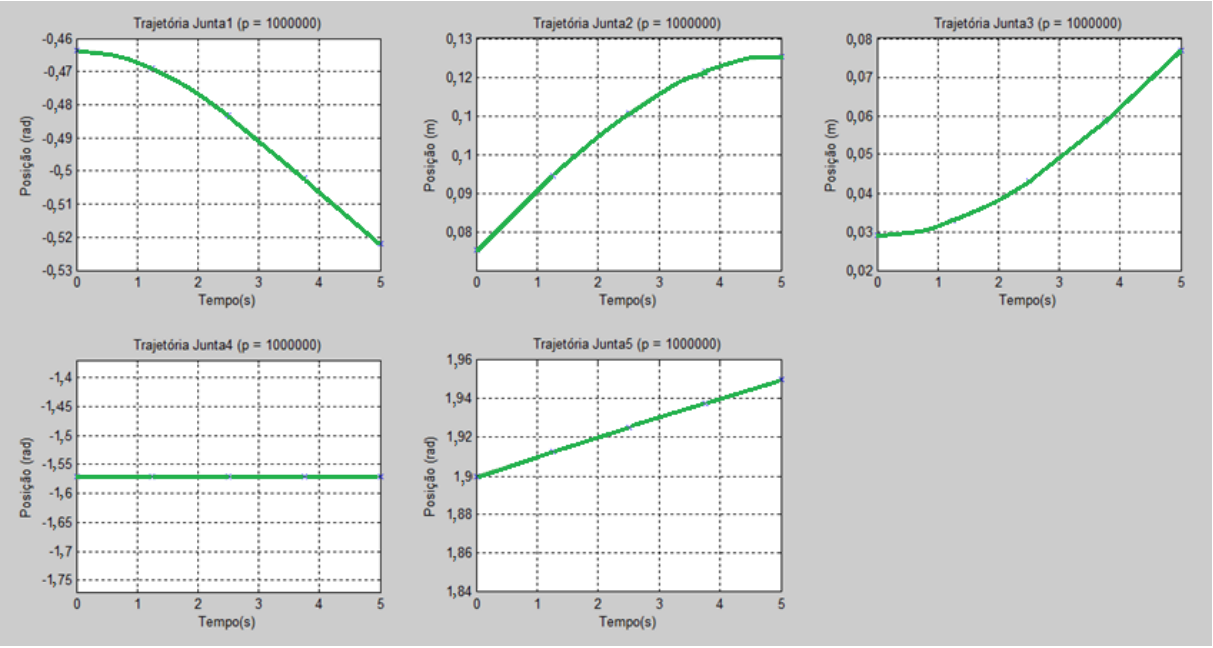


Figura 4.17 – Posição em relação ao tempo de cada junta em uma trajetória vertical

Em seguida, testou-se a rotina "*Place To Position*" (Seção 3.4), a qual gera pontos que, ao serem ligados, formam a trajetória *pick-and-place*. Na Figura 4.18 está apresentada uma trajetória gerada com os seguintes parâmetros: x_i=0,100 m, $y_i = 0{,}200\ m$, $z_i = 0{,}100\ m, x_f = 0{,}150\ m, y_f = 0{,}210\ m, z_f = 0{,}100\ m, \theta_{5_i} = 1{,}900\ rad$ e $\theta_{5_f} = 1{,}1950\ rad$.

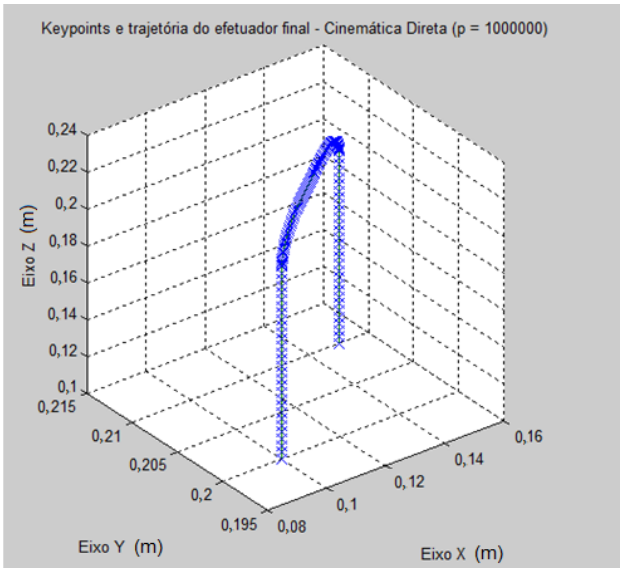


Figura 4.18 – Trajetória *pick-and-place* no plano tridimensional

Na Figura 4.19 são apresentados os gráficos da posição em relação ao tempo de cada junta para a trajetória *pick-and-place*.

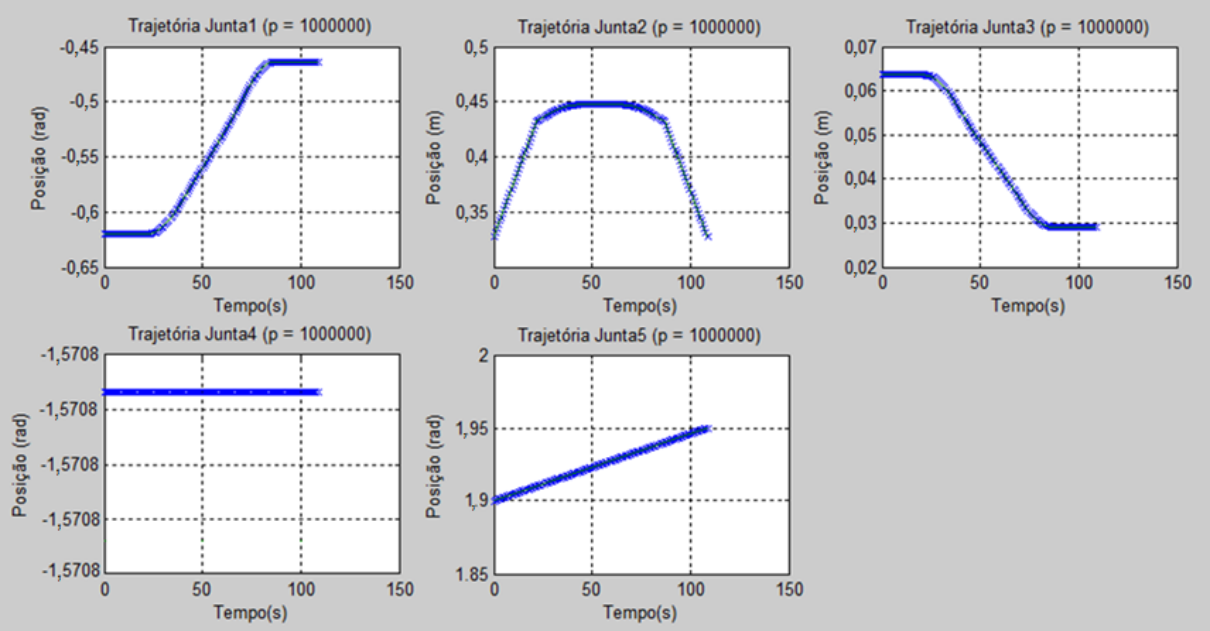


Figura 4.19 – Posição em relação ao tempo de cada junta em uma trajetória vertical

4.3 Integração com o algoritmo de controle do robô

Para realização dos movimentos desejados, os vetores das posições das juntas gerado pelo algoritmo de planejamento de trajetórias devem ser fornecido ao algoritmo de controle dos atuadores do robô. Para tal, a taxa de amostragem do controlador (tx_a) já fixada em *0,002 s* por Sarmanho, 2014, deve ser utilizada para a geração dos vetores no espaço das juntas.

O usuário deve fornecer o tempo total da trajetória, o número de pontos entre cada trecho da trajetória *pick-and-place* e a porcentagem do valor do raio de giro do semiarco (por exemplo, 10%, 20%, 30%, etc., conforme descrito anteriormente na Seção 3.4). Missiaggia, 2014, define que 4 (quatro) é o número mínimo de pontos que uma trajetória precisa possuir. Porém, ao realizar os testes, é preferível que um maior número de pontos entre cada trecho seja adotado, já que, quanto mais pontos são definidos, mais próxima da trajetória desejada é a trajetória realizada. Quanto ao tempo total da trajetória, não pode ser nem muito curto (menor que 4 segundos) nem muito longo (maior que 10 segundos). O tempo muito longo é influenciado excessivamente pelo atrito, dificultando o controle. O tempo muito curto está relacionado com a compressibilidade do ar, o que também dificulta o controle de posição.

Ao escolher o tempo total (tt_T) e o número de pontos da trajetória ($Nump$), deve-se escolher valores que façam com que tanto a relação $tx_a/tt_T/(Nump-1)$ quanto a relação $tt_T/(Nump-1)/tx_a$ resulte em um número inteiro, já que, estas relações serão utilizadas para a geração das *splines* de Simon, 2004 e, estas, são geradas através de matrizes, as quais não podem ser geradas com números que não sejam inteiros.

Primeiramente, testou-se a integração do algoritmo de planejamento de trajetórias com o de controle dos atuadores, sem levar em conta o sistema de visão, ou seja, adotou-se pontos inicial e final arbitrários para a geração das trajetórias. Na Figura 4.20 é apresentado um exemplo de trajetória *pick-and-place* utilizada para teste da integração dos dois algoritmos, com percurso completo, ou seja, o robô se desloca até a peça para capturá-la e volta para sua posição inicial.

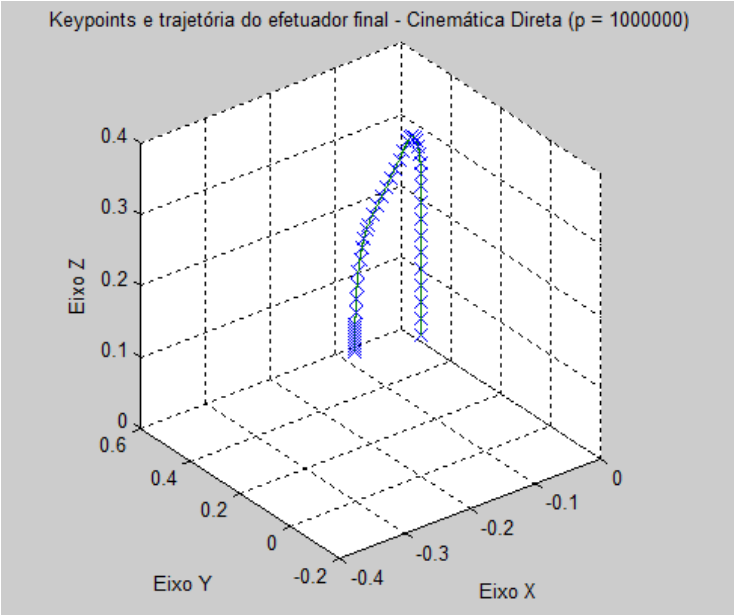


Figura 4.20 – Trajetória *pick-and-place* utilizada como teste para integração dos dois algoritmos

Na Figura 4.21 são apresentados os gráficos da posição em relação ao tempo de cada junta no percurso da trajetória *pick-and-place* testada gerados pelo algoritmo de planejamento de trajetórias (Missiaggia, 2014) a partir dos seus pontos chaves.

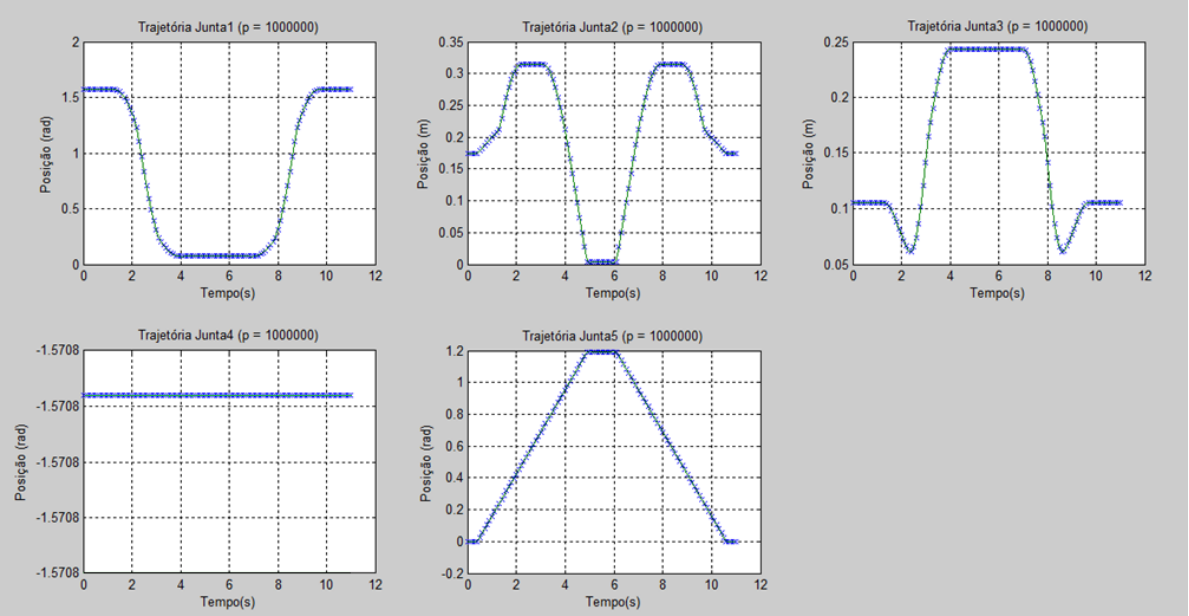


Figura 4.21 – Posição em relação ao tempo de cada junta no percurso da trajetória *pick-and-place* testada

Após validadas as trajetórias simuladas no programa de planejamento de trajetórias, levando-se em consideração o espaço de trabalho do robô, realizou-se testes experimentais para trajetórias *pick-and-place* baseadas na orientação e posição das peças obtidas pelo algoritmo de visão computacional. No Capítulo 5, são apresentados esses resultados experimentais e discussões.

4.4 Análise de performance dos algoritmos

Para realização da análise de performance de um algoritmo, deve-se levar em consideração a complexidade de tempo e de espaço. Segundo Cook, 1983, a medida de complexidade de um algoritmo mais importante é a medida de tempo, devido às pesquisas serem direcionadas para projetar e analisar algoritmos quanto à eficiência, fornecendo a solução de um problema com a rapidez desejada.

Com a ferramenta Matlab® é possível, ao executar o algoritmo, obter automaticamente o tempo de cada função criada dentro algoritmo. Desta forma, é possível, além de analisar a performance do algoritmo como um todo, identificar aquelas funções que o estão deixando mais lento. Assim, se possível, pode-se melhorar o desempenho dessa função.

Primeiramente, analisou-se a performance do sistema de visão computacional, sendo que a imagem utilizada tinha dimensões (185 x 261 pixels). Quanto maior a imagem, maior o tempo de execução do algoritmo. Quanto à calibração da câmera, o algoritmo levou um tempo de *24,028 s* para computar a geração da matriz de calibração ***P*** e, um tempo de *204,389 s* para computar a geração da tabela de comparação **LUT**. É importante destacar que a tarefa de calibração de câmera só é realizada no início do processo, só sendo necessário repeti-la, no caso de troca do tipo de peça. Em relação à obtenção da identificação e posição da peça, o algoritmo levou um tempo de *2,470 s*.

Em seguida analisou-se a integração com o algoritmo de planejamento de trajetórias. Quanto à geração da trajetória *pick-and-place* e posterior criação da matriz de pontos PC, o tempo estimado pelo Matlab® é de *4,019 s*. Quanto à geração das *splines* e posterior transformação para o espaço das juntas, o tempo estimado é de *6,914 s*.

Desta forma, estima-se que o tempo total que o algoritmo leva para identificar uma peça e gerar os vetores das juntas é de aproximadamente *13,4 s*.

Quanto à performance do algoritmo de controle do robô, não foi possível estimar já que sua parametrização não é automática pois depende de diversos fatores ainda não controlados.

5 RESULTADOS EXPERIMENTAIS E DISCUSSÕES

O processo de geração de trajetória proposto no presente trabalho é constituído das seguintes etapas: identificação da posição e orientação de peças a serem manipuladas pelo robô; criação de matrizes de valores de coordenadas que o robô deve percorrer e interpretação das informações pelo sistema de controle dos atuadores. Para exemplificar o funcionamento do sistema, neste capítulo será apresentado um estudo de caso em que o efetuador do robô deve percorrer um caminho formado a partir de um determinado número de pontos localizados em seu volume de trabalho, sendo as coordenadas do ponto final (onde se localiza a peça) obtidas através do sistema de visão apresentado no Capítulo 3.

Primeiramente, são apresentados os resultados do método do sistema de visão computacional, ou seja, os da identificação da posição e da orientação de uma peça a ser manipulada para poder-se avaliar a eficácia do método proposto para identificação da posição e giro da peça com relação ao sistema de referência. Em seguida, são apresentados os resultados de simulações realizadas com o código de geração de trajetórias, assim como resultados de ensaios experimentais.

Para validação do método foram realizados 10 (dez) testes experimentais com cada uma das três peças a serem manipuladas no ambiente de trabalho do robô, conforme apresentado na Seção 3.6.2, sendo as peças de cor branca de acrílico e a mesa de cor preta. Conforme já mencionado, esta escolha (peça branca em fundo preto) facilita a limiarização e segmentação da imagem. A primeira etapa do procedimento consistiu na realização da calibração da câmera, relacionando coordenadas mundo com as coordenadas em pixel da imagem. Para isso, utilizou-se o padrão gráfico (tabuleiro quadriculado) fixado em uma superfície plana do próprio robô, obtendo-se, assim, as coordenadas mundo referentes a cada ponto do padrão e suas coordenadas correspondentes em pixel e, assim, possibilitando a construção da matriz de calibração da câmera.

O segundo passo consistiu na aplicação do método de reconhecimento da posição e orientação da peça, apresentado no Capítulo 3, obtendo-se as coordenadas em pixel do seu centroide assim como sua orientação. Após essa etapa, se transformou os valores do centroide da peça para coordenadas mundo em relação ao robô e o valor do ângulo de graus para radianos para que esses dados pudessem ser utilizados pelo programa de geração de

trajetórias. A Figura 5.1 ilustra através de um ponto azul, o centroide encontrado para a peça em estudo.

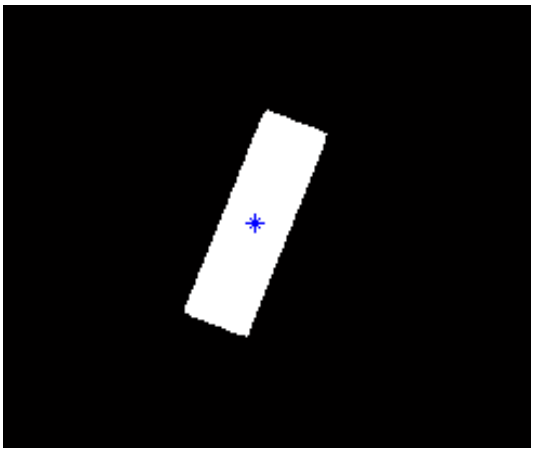


Figura 5.1 – Identificação do centroide obtido pelo algoritmo

A tabela de comparação utilizada neste exemplo foi gerada com resolução de *1º*, assim, o algoritmo identificou o ângulo de giro da peça como *62º* ou *1,0821 rad*.

Uma vez obtidos os valores numéricos em *x* e *y*, e o ângulo de giro da peça em relação ao eixo central no sistema de coordenadas do robô, gerou-se a matriz de pontos (posições das juntas) para a realização de uma trajetória na qual o manipulador robótico, partindo de um determinado ponto no espaço, seja capaz de efetuar uma tarefa de *pick-and-place* entre a posição atual do manipulador e a posição da peça.

Após gerada a matriz de pontos chaves, transferiu-se os valores para o programa de geração de trajetórias desenvolvido por Missiaggia (2014) que, por sua vez, forneceu como resultado os vetores em coordenadas de juntas que devem ser enviados ao algoritmo de controle do robô.

A sequência de operações descrita anteriormente permite que o algoritmo de identificação da posição e orientação de peças seja avaliado experimentalmente, já que se o mesmo é efetivo no reconhecimento da posição e orientação da peça, o robô será capaz de capturar através de seu efetuador o objeto de estudo.

Nos primeiros testes realizados, verificou-se que a orientação do 5º GL do manipulador esteve sempre de maneira correta, permitindo a captação da peça. Porém, verificou-se um desvio nas coordenadas *x* e *y* do centroide, os quais se repetiram de maneira igual em todos os 4 primeiros testes, caracterizando um desvio de fácil compensação.

Segundo Sarmanho, 2014, não existem garantias de que o comportamento do manipulador projetado irá atingir a precisão do posicionamento do efetuador final ou mesmo a aplicação de forças pelo manipulador exigidas em determinadas aplicações. Desta forma, esse desvio identificado pode ter sua causa tanto na construção (execução e montagem) quanto no controle do robô. Outra causa possível, é um erro na obtenção da origem do sistema de coordenadas do robô, visto que, como é localizada na parte interna do robô, é de difícil acesso. Tal desvio foi, então, medido com o paquímetro, sendo, em coordenadas *x*, de + 6 mm e, em coordenadas *y* de +28 mm. Foi corrigido subtraindo-se os valores de desvio, relativos à cada coordenada, das coordenadas do centroide obtido.

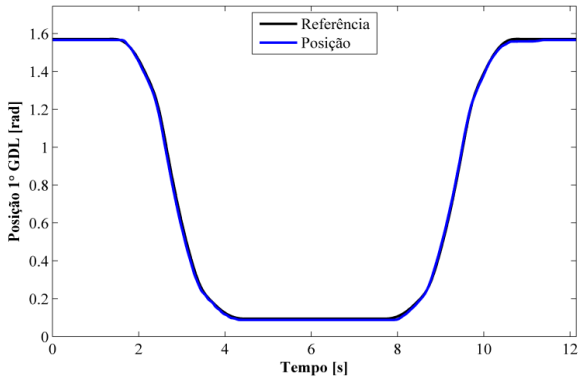
Após corrigido este desvio, realizou-se o restante dos testes, conforme apresentado o resumo de testes na Tabela 5.1, onde S representa que o manipulador conseguiu capturar a peça, e N representa que ele não conseguiu.

Tabela 5-1- Resumo de testes realizados

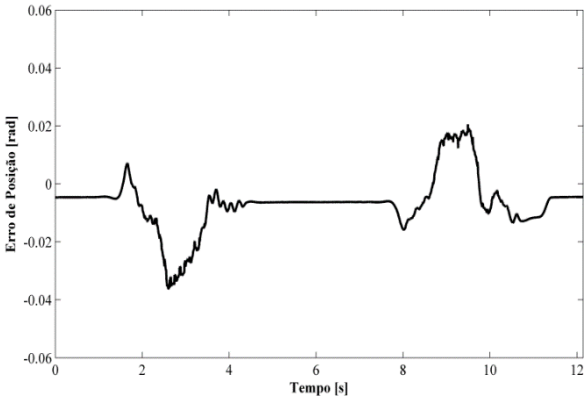
	Teste									
Peça	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
PA	N	N	N	N	S	S	S	S	S	S
PB	S	S	S	S	S	S	S	S	S	S
PC	S	S	S	S	S	S	S	S	S	S

Como, após compensado o desvio percebido nos 4 (quatro) primeiros testes, verificou-se que o algoritmo é eficaz na identificação do giro das três peças testadas, tendo sido capaz, em 100% dos 26 testes restantes, de capturar e manipular a peça.

Uma análise dos desvios das trajetórias prevista e realizada é possível através das figuras 5.2.a, 5.3.a, 5.4.a e 5.5.a, onde as trajetórias de referência (na cor preta), obtidas no programa de planejamento de trajetórias para cada junta e os resultados experimentais do seguimento das respectivas trajetórias pelas juntas do robô pneumático (na cor azul), do exemplo da Figura 5.1 são apresentadas. Nas figuras 5.2.b, 5.3.b, 5.4.b e 5.5.b, estão apresentados os desvios de seguimento das trajetórias.

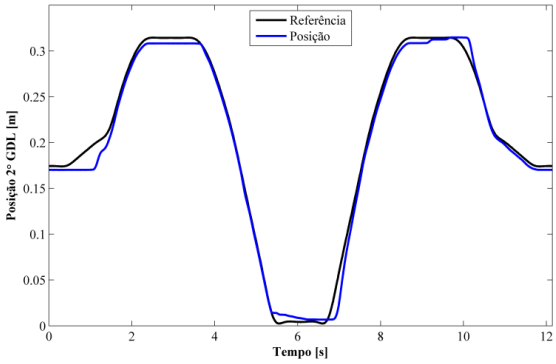


(a) Trajetória de Referência e Realizada

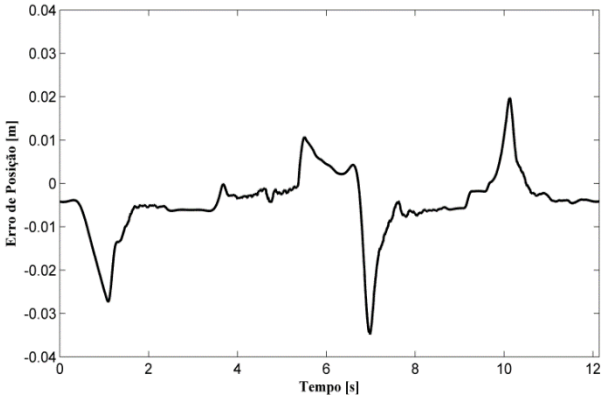


(b) Desvios no Seguimento de trajetória

Figura 5.2 - Trajetória de Referência e Realizada no 1° GL e seus respectivos desvios.

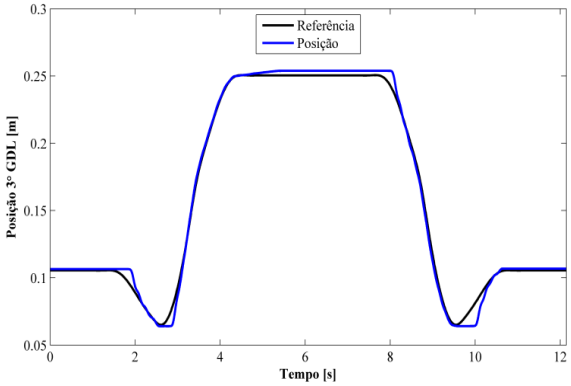


(a) Trajetória de Referência e Realizada

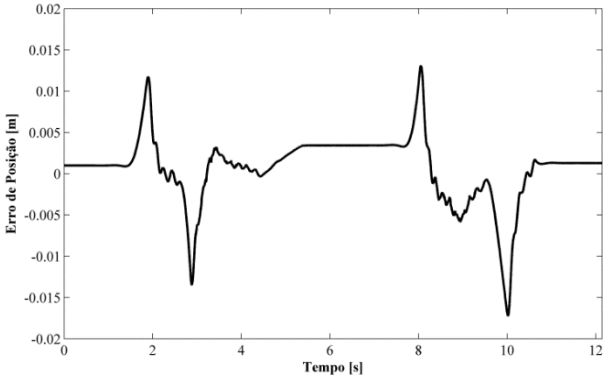


(b) Desvios no Seguimento de trajetória

Figura 5.3 - Trajetória de Referência e Realizada no 2° GL e seus respectivos desvios.

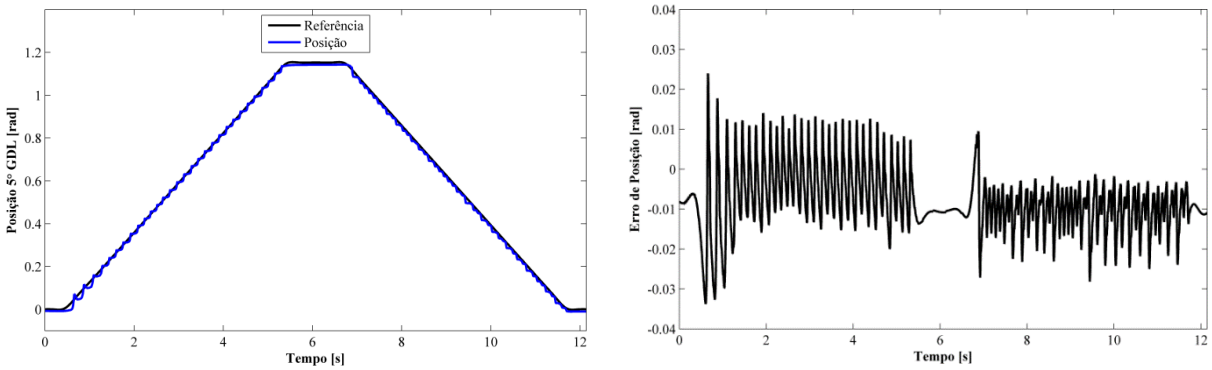


(a) Trajetória de Referência e Realizada



(b) Desvios no Seguimento de trajetória

Figura 5.4 - Trajetória de Referência e Realizada no 3° GL e seus respectivos desvios.



(a) Trajetória de Referência e Realizada (b) Desvios no Seguimento de trajetória

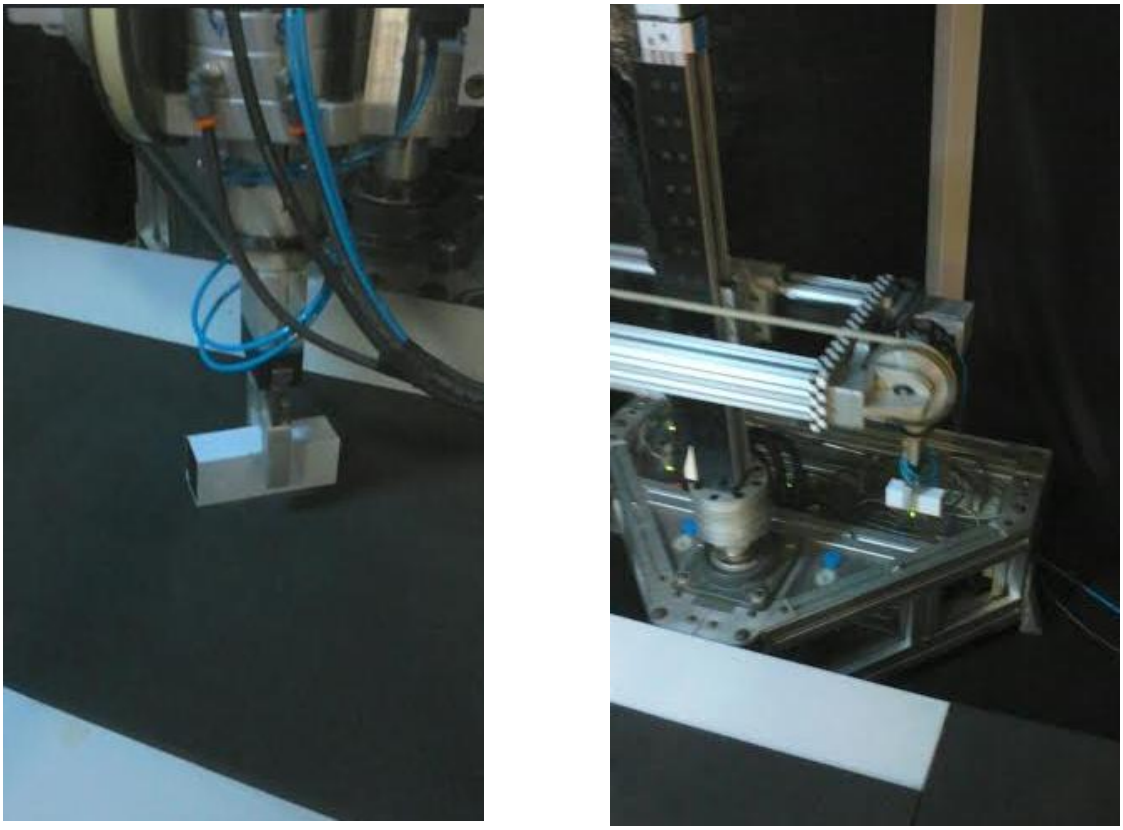
Figura 5.5 - Trajetória de Referência e Realizada no 5° GL e seus respectivos desvios.

Destaca-se que, nos resultados apresentados nas figuras 5.2 a 5.5, não são expressos os gráficos referentes à situação de seguimento de trajetória para o 4° grau de liberdade do manipulador. Isto se deve ao fato de que para este GL, a junta foi mantida durante toda a trajetória na posição de *-1,5708 rad (-90°)*, apresentando um erro de regime de aproximadamente *0,024 rad*, considerado aceitável para aplicações de movimentação estudadas.

O fato do 4° GL não ser movimentada nos testes apresentados é justificada, através da análise de uma situação de ambiente fabril automatizado, onde a peça é movimentada através das estações utilizando-se esteiras. Dessa forma, os movimentos de *pick-and-place*, no ponto de vista de um robô, normalmente ocorrem manipulando-se as peças por suas laterais, permitindo assim que as mesmas sejam inseridas em máquinas ferramenta, por exemplo, ou, no caso em estudo, que a peça seja movimentada para outra esteira (situação de descarte de peças defeituosas).

Verifica-se que os testes experimentais apresentaram resultados satisfatórios, com desvio máximo de *+/- 0.04 m* no 2° GL e *+/- 0,024 rad* no 5° GL, fato corroborado através dos resultados apresentados nas figuras 5.2 a 5.5, observando-se ainda que, o robô manipulador foi capaz de capturar as peças em estudo, conforme pode ser observado através na Figura 5.6.

Os desvios apresentados nas figuras 5.2 a 5.5 estão associados, principalmente, às características de controlador utilizado no robô para o seguimento da trajetória proposta, a qual está dentro dos requisitos propostas por Sarmanho, 2014.



(a) Manipulador capturando a peça (b) Manipulador transportando a peça

Figura 5.6 - Resultados experimentais da remoção de uma peça da mesa

Os testes experimentais abrangendo o sistema de visão integrado ao planejamento da trajetória com mínimo *jerk* e ao sistema de controle, abrangeram as 3 peças em formato de paralelepípedo conforme descritas na Seção 3.6.2, na cor branca com o fundo preto, delimitando assim este trabalho a essas condições. Contudo, o sistema de visão foi capaz de identificar a posição e a orientação de peças também em formato elíptico e circular, conforme apresentado na Seção 4.1, podendo assim ser aplicado a um manipulador com outros tipos de efetuador final que possam manipular peças desse tipo, como por exemplo, um manipulador com uma ventosa ao invés de garra.

6 CONCLUSÕES E SUGESTÕES DE TRABALHOS FUTUROS

Com base nos resultados obtidos através das simulações e testes experimentais, apresentados no Capítulo 5, pode-se afirmar que os objetivos delimitados ao princípio deste trabalho foram alcançados, visto que o algoritmo proposto realizou a determinação da posição e orientação de objetos, podendo, assim, fornecer as trajetórias necessárias para o controle de posição do robô utilizado, permitindo a manipulação de peças.

A metodologia proposta no Capítulo 3 para identificação da posição e orientação de objetos depende da resolução da impressora utilizada para geração do padrão (tabuleiro de xadrez) e da resolução definida para a tabela de comparação. Esse método pode ser aplicado em outros robôs e com outras câmeras, sendo que essas podem ter seus lugares modificados, já que o sistema é calibrado automaticamente.

Ao longo do desenvolvimento deste estudo verificou-se também que os desvios dos resultados dependem de condições de operação, como, por exemplo, iluminação do local, cor da peça e da superfície de trabalho (fundo da imagem), além da textura da superfície da peça e da sua geometria. No caso da geometria da peça, notou-se que, quanto menos suavizados são os cantos, melhores são os resultados obtidos na medição da orientação. Percebeu-se que um fator que influencia nos resultados experimentais é a iluminação, já que qualquer alteração nela pode gerar sensíveis variações nos resultados de posição e orientação obtidos. Isto se deve ao fato de que a limiarização é realizada baseada nas cores da imagem, as quais, por sua vez, são dependentes da iluminação. Uma alternativa como a utilizada com sucesso no presente trabalho, é a utilização do método de Otsu para seleção automática do limiar, adaptando-se assim às mudanças nas cores da imagem influenciadas pela iluminação.

A proposta desse trabalho limita-se a identificação da posição e orientação de peças dentro de um espaço de trabalho bem definido e calibrado para operações específicas de manipulação. Nessas condições é suficiente, para a extração das informações necessárias, a análise de imagens bidimensionais com o emprego de apenas uma câmera. Limita-se, portanto, à verificação da posição e orientação de objetos no plano *x,y*, não extraindo informação de altura (coordenada *z* do espaço cartesiano do robô), peculiar de, por exemplo, sistemas de visão *stereo*, que emprega duas ou mais câmeras ou de sistemas de visão com utilização de *kinect*. O trabalho se limita também a identificação e análise de imagens com apenas um objeto.

Como sugestão de trabalhos futuros, propõe-se:

- Implantar um sistema de visão *stereo* com a utilização de mais uma câmera para identificação da altura da peça;
- Realizar uma análise mais aprofundada sobre as incertezas, considerando os possíveis erros ocasionados pela aplicação do método desenvolvido no robô pneumático;
- Ampliar o método, introduzindo a capacidade de detecção de possíveis obstáculos no espaço de trabalho do manipulador a fim de gerar trajetórias seguras perante a possibilidade de impacto com obstáculos;
- Elaborar ferramentas computacionais para facilitar a interface homem-máquina, permitindo uma troca de informação mais amigável entre o usuário e o algoritmo desenvolvido;
- Estudar os efeitos das características das trajetórias geradas sobre o desempenho do algoritmo de controle.

REFERÊNCIAS BIBLIOGRÁFICAS

Abdel-Aziz, Y.I.; Karara, H.M. **Direct linear transformation from comparator coordinates into object space coordinates in close-range photogrammetry.** Proceedings of the Symposium on Close-Range Photogrammetry *(pp. 1-18)*. Falls Church, VA: American Society of Photogrammetry, 1971.

Atera. **Atera Informática.** WebCam Clone 10028. Disponível em: < http://www.atera.com.br/dispprod.asp?COD=WVCONF10028>. Acesso em 15 jul. 2014.

Ballard, D. H.; Brown, C. M. **Computer Vision.** Prentice-Hall, Englewood Cliffs, NJ, 1982.

Berendsen, H.J.C. **A Student's Guide to Data and Error Analysis.** Cambridge University Press, NY, 2011.

BIPM. Joint Commitee for Guides in Metrology (JCGM). **Evaluation of Measurement Data: Guide to the Expression of Uncertainty in Measurement** (GUM). Sèvres, 2008, 1ª ed, 120 p. Disponível em <http://www.bipm.org/en/publications/guides>. Acesso em 17 Set. 2014.

Cantor, G. **Optics after Newton: theories of light in Britain and Ireland**, 1704 - 1840, Manchester University Press, Manchester, 1983.

Carducci, G.; Foglia, M.; Gentile, A.; Giannoccaro, N.I.; Messina, A. **Pneumatic robotic arm controlled by on-off valves for automatic harvesting based on vision localisation.** IEEE ICIT '04. 2004 IEEE International Conference on Industrial Technology, 2004.

CEPRSM. **Página Dinâmica para Aprendizado do Sensoriamento Remoto.** Centro Estadual de Pesquisas em Sensoriamento Remoto e Meteorologia. Universidade Federal do

Rio Grande do Sul. Disponível em: <http://www.ufrgs.br/engcart/PDASR/formcor.html>. Acesso em 09 jul. 2014.

Chavez, G. C.; Lhiang, Z. **Sistema celular para reconhecimento de padrão invariante.** In IV Workshop em Tratamento de Imagens, pages 2–3, 2003.

Cook, S. A. **An overview of computational complexity.** Communications of the ACM, v.16, n. 6, p. 401-407. New York, 1983.

Craig. J. J. **Introduction to Robotics: Mechanics and Control.** 3. ed. Pearson Prentice Hall. Upper Saddle River: Prentice Hall, 2005.

Cukla, A. R., **Arquitetura Microcontrolada Programável Aplicada ao Controle de um Servoposicionador Pneumático**. Dissertação de Mestrado, Programa de Pós-graduação em Engenharia Mecânica, Universidade Federal do Rio Grande do Sul, 2012.

dSPACE. **DS1104 R&D Controller Board.** Disponível em: <www.dspace.com/en/inc/home.cfm>. Acesso em 08 jan. 2015.

EPSON. **EPSON RC+ 7.0 (Ver.7.0) SPEL+ Language Reference Rev.3.** 2012.

Farina, M. **Psicodinâmica das cores em comunicação.** 4. ed. Edgard Blucher, São Paulo, 1990. 242p.

Feliciano, F.F.; Souza, I. L.; Leta, F. R. **Visão Computacional Aplicada à Metrologia Dimensional Automatizada: Considerações sobre sua Exatidão.** ENGEVISTA, v.7, n.2, p.38-50, Dezembro 2005.

Figueiredo, **A. Fundamentos de produção gráfica para quem não é produtor gráfico.** Rubio, Rio de Janeiro, 2003.

Figueiredo, D. **Conceitos Básicos de Sensoriamento Remoto.** Companhia Nacional de Abastecimento - CONAB. Brasília, DF, 2005. Disponível em: <http://72.14.205.104/search?q=cache:r9r3jyI5bKsJ:www.conab.gov.br/conabweb/download/SIGABRASIL/manuais/conceitos_sm.pdf+divino+figueiredo,+conceitod+basicos+sensoriamento+remoto&hl=pt-BR&ct=clnk&cd=1&gl=br>. Acesso em 10 fev. 2008.

Foley, J. D.; Van Dam, A.; Feiner, S.K.; Hughes, J. F. **Computer graphics: principles and practice.** Reading, MA: Addison-Wesley, 1990. 1176p.

Fraser, B.; Murphy, C.; Bunting, F. **Real world color management**. Peachpit, Berkeley, EUA, 2005.

Fu, K. S.; Gonzalez, R. C.; Lee, C. S. G. **"Robotics: Control, Sensing, Vision and Intelligence"**. McGraw-Hill, New York, 1987.

Gervini, V. I. **Modelagem e Controle de um Servopocisionador Pneumático Utilizando Redes Neurais.** Tese de Doutorado, Universidade Federal do Rio Grande do Sul, Programa de Pós-Graduação em Engenharia Mecânica, 2014.

Gonzalez, R. C.; Woods, R. E. **Digital Image Processing**. Upper Saddle River: Prentice Hall, 2007.

Grassi, M. V. **Desenvolvimento e aplicação de um sistema de visão para robô industrial de manipulação.** Dissertação de Mestrado, Universidade Federal do Rio Grande do Sul, Programa de Pós-Graduação em Engenharia Mecânica, 2005.

Gulati, R. R. **Monochrome and colour television.** New Age International Publishers. New Delhi, 2006.

Holdschip, R. **A Influência dos Sistemas de Gerenciamento de Cores em Provas Digitais.** Universidade Estadual Paulista, Bauru, 2008.

Hu, M. K. **Visual Pattern Recognition by Moment Invariants.** *IRE Transactions on Information Theory*, p. 179-187, 1962.

Hussein, I. S.; Nordin, M. J. **Palmprint Identification Using Invariant Moments Algorithm Based on Wavelet Transform.** Advanced Computer and Communication Engineering Technology. Lecture Notes in Electrical Engineering, v. 315, p. 905-914, 2015.

IFR. **World Robotics - Industrial Robots 2014.** International Federation of Robotics. Disponível em: < http://www.ifr.org/industrial-robots/statistics>. Acesso em 20 out. 2014.

Jain, R.; Kasturi, R.; Schunck, B. G. **Machine Vision.** MIT Press and McGraw-Hill, Inc.USA, 1995.

Jung, C.R. **Introdução à Visão Computacional: Notas de aula**, Universidade Federal do Rio Grande do Sul, Instituto de Informática, INF, Porto Alegre, 2013.

Keshmiri, M.; Keshmiri, M.; Mohebbi, A. **Augmented online point to point trajectory planning, a new approach in catching a moving object by a manipulator.** IEEE 8th International Conference on Control and Automation (ICCA), 2010.

Keshmiri, M.; Wen-Fang, Xie. **Visual servoing of a robotic manipulator using an optimized trajectory planning technique.** IEEE 27th Canadian Conference on Electrical and Computer Engineering (CCECE), 2014.

Khotanzad, A.; Hong, Y. H. **Invariant image recongnition by zernike moments.** IEEE Transactions on Pattern analysisand machine intelligence, 12:1, 1990.

Kline, S. J.; F. A. McClintock. **Describing Uncertainties in Single-Sample Experiments. Mechanical Engineering.** Vol. 75, No. 1, January 1953: 3-8.

Koch, H.; Konig, A.; Weigl-Seitz, A.; Kleinmann, K.; Suchy, J. **Multisensor Contour Following With Vision, Force, and Acceleration Sensors for an Industrial Robot.** IEEE Transactions on Instrumentation and Measurement, , vol.62, no.2, pp.268,280, Feb. 2013

Laureano, G.H.C. **Coeficiente de Correlação Intraclasse: Comparação entre métodos de estimação clássico e bayesianos.** Trabalho de Conclusão de Curso, Universidade Federal do Rio Grande do Sul, Instituto de Matemática, 2011.

Lewis, F. L.; Abdallah, C. T.; Dawson, D. M. **Robot Manipulator Control: Theory and Practice.** Second Edition, 2003.

Li, H.; Jin, X.; Yang, N.; Yang, Zhe. **The recognition of landed aircrafts based on PCNN model and affine moment invariants.** International Association for Pattern Recognition - Pattern Recognition Letters, v. 51, p. 23–29, 1 Jan 2015.

Marin, D.; Aquino, A.; Gegundez-Arias, M. E.; Bravo, J.M. **A New Supervised Method for Blood Vessel Segmentation in Retinal Images by Using Gray-Level and Moment Invariants-Based Features.** Departement of Electronics, Computer Science. & Automation Engeneering. University of Huelva, Palos de la Frontera, Spain. IEEE Transactions on Medical Imaging, v. 30, p. 146-158, jan, 2011.

MathWorks. **Matlab**. Disponível em: < http://www.mathworks.com>. Acesso em 08 jan. 2015.

Melchiades, F. G.; Boschi, A. O. Cores e revestimentos cerâmicos. **Cerâmica Industrial**. São Paulo, v. 4, n. 1-6, p. 11-8, jan/dez, 1999.

Missiaggia, L. **Planejamento otimizado de trajetória para um robô cilíndrico acionado pneumaticamente.** Dissertação de Mestrado, Universidade Federal do Rio Grande do Sul, Programa de Pós-Graduação em Engenharia Mecânica, 2014.

Moreno, R. J.; Lopez, J. D. **Trajectory Planning for a Robotic Mobile Using Fuzzy C-Means and Machine Vision.** IEEE/STSIVA XVIII Symposium of Image, Signal Processing, and Artificial Vision, 2013.

Müller-Karger, C. M.; Mirena, L. G.; López, T. S. **Hyperbolic Trajectories for Pick-and-Place Operations to Elude Obstacles.** IEEE Transactions on Robotics and Automation, v. 16, n. 3, p. 294-300, 2000.

Niku, S. B. **Introduction to Robotics: Analysis, Systems, Applications.** Prentice Hall, Upper Saddle River, 2001.

Otsu, N. **A threshold selection method from gray-level histograms.** IEEE Transactions on Systems, Man and Cybernetics, v. 9, n. 1, p. 62-66, 1979.

Pacheco, R.R. **Trajetórias Suaves em Ambiente Robótico Simulado.** Universidade do Estado de Santa Catarina – Centro de Ciências Tecnológicas, 2010.

Palmer, S. E. **Vision Sciense: Photons to phenomenology.** MIT Press, Cambridge, MA, 1999.

Pearson, E. S.; Gosset, W. S.; Plackett, R. L.; Barnard, G. A. (1990). **Student: a statistical biography of William Sealy Gosset.** Oxford University Press, USA.. Student: a statistical biography of William Sealy Gosset. Oxford University Press, USA, 1990.

Rakprayoon, P.; Ruchanurucks, M.; Coundoul, A. **Kinect-based obstacle detection for manipulator**. IEEE/SICE International Symposium on System Integration (SII), 2011.

Rijo, M.G.Q. **Desenvolvimento da Base e Controle do Grau de Liberdade Rotacional de um Robô Cilíndrico com Acionamento Pneumático**. Dissertação de Mestrado, Universidade Federal do Rio Grande do Sul, Programa de Pós-Graduação em Engenharia Mecânica, 2013.

Rodrigues, E. T.; Rêgo, R. G. **A dinâmica das transformações geométricas e simetria utilizando o livro de espelhos.** X Encontro Nacional de Educação Matemática, 2010.

Romano, V. F.; Dutra, M. S., **Introdução à Robótica Industrial.** In: Vitor Romano. (Org.). Robótica Industrial: Aplicação na Indústria de Manufatura e de Processos, 1 ed. São Paulo: Edgard Blücher, pp. 1-19, 2002.

Rosário, J. M. **Princípio de Mecatrônica**. Prentice Hall, 2005.

Sarmanho Jr., C. A. C. **Desenvolvimento de um Robô Pneumático de 5 Graus de Liberdade com Controlador Não Linear com Compensação de Atrito.** Tese de Doutorado, Universidade Federal do Rio Grande do Sul, Programa de Pós-Graduação em Engenharia Mecânica, 2014.

Shah, M. **Fundamentals of Computer Vision.** Computer Science Department. University of Central Florida. Orlando, 1997.

Shapiro, L.G.; Stockman, G.C. **Computer Vision**. Prentice-Hall, New Jersey, 2001.

Shen, Y.; Huper, K. **Optimal Trajectory Planning of Manipulators Subject to Motion Constraints**. IEEE Transactions on Robotics and Automation. Julho, p. 9-16, 2005.

Shrout, P. E.; Fleiss, J.L. **Intraclass correlations: Uses in assessing reliability.** Psychological Bulletin, v. 86, n°. 2, p. 420-428, 1979.

Silva, C. C.; Martins, R. A. **A teoria das cores de Newton: um exemplo do uso da história da ciência em sala de aula.** Ciência e Educação, v. 9, n°. 1, p. 53-65, 2003.

Simon, D. **Data Smoothing and Interpolation Using Eighth-order Algebraic Splines**. IEEE Transactions on Signal Processing, vol. 52, n°. 4, 2004.

Souza, J.S.; Cardoza, J. A. S. **Sensores de Imagens Digitais CCD e CMOS.** Congresso Norte e Nordeste de Pesquisa e Inovação - VII CONNEPI, 2009.

Souza, K.P.; Pistori, H. **Implementação de um Extrator de Características baseado em Momentos da Imagem.** Universidade Católica Dom Bosco, Campo Grande/MS, Grupo de Pesquisa em Engenharia e Computação, 2005.

Spong, M. W.; Vidyasagar, M. **Robot Dynamics and Control**. John Wiley & Sons, Inc., 1989.

Szeliski, R. **Computer Vision: Algorithms and Applications.** Springer Science & Busniness Media, 2010.

Tsai, R.Y. **An efficient and accurate camera calibration technique for 3d machine vision.** IEEE Computer vision and pattern recognition, pp. 364-374, 1986.

Turner, L. A. W. **Eletronics Engineers's Reference Books.** Butterworth & Co. Publishers, 4th Edition. London, 1976.

Weng, J., Cohen, P., Hermiou, M. **Camera calibration with distortion models and accuracy evaluation.** IEEE Trans. On Pattern Analysis and Machine Intelligence, Vol 14, no 10, pp 965-980, Oct., 1992.

Zaeri, N.; Baker, F. Dib, R. **Thermal Face Recognition Using Moments Invariants.** International Journal of Signal Processing Systems, v. 3, n. 2, Dec., 2015

Zhou, F.; Kornerup, P. **Computing moments by prefix sums.** Relatório Técnico PP-Publisher University of Southern Denmark, v. 1, 1995.

APÊNDICE A - Alguns conceitos de ondas eletromagnéticas

A luz é uma onda eletromagnética, cujo comprimento de onda se inclui num determinado intervalo dentro do qual o olho humano é a ela sensível [Cantor, 1983]. Ou seja, o que chamamos de luz, é uma radiação eletromagnética que podemos ver e que se situa entre a radiação infravermelha e a radiação eletromagnética do espectro como pode ser visto na Figura A.1. Cujos valores admitidos internacionalmente desde 1931 pela CIE (*Commission Internationale de l'Eclairage*), são 435,8 nm, 546,1 nm e 700 nm como os que representam espectralmente as três cores primárias aditivas, azul, verde e vermelho respectivamente.

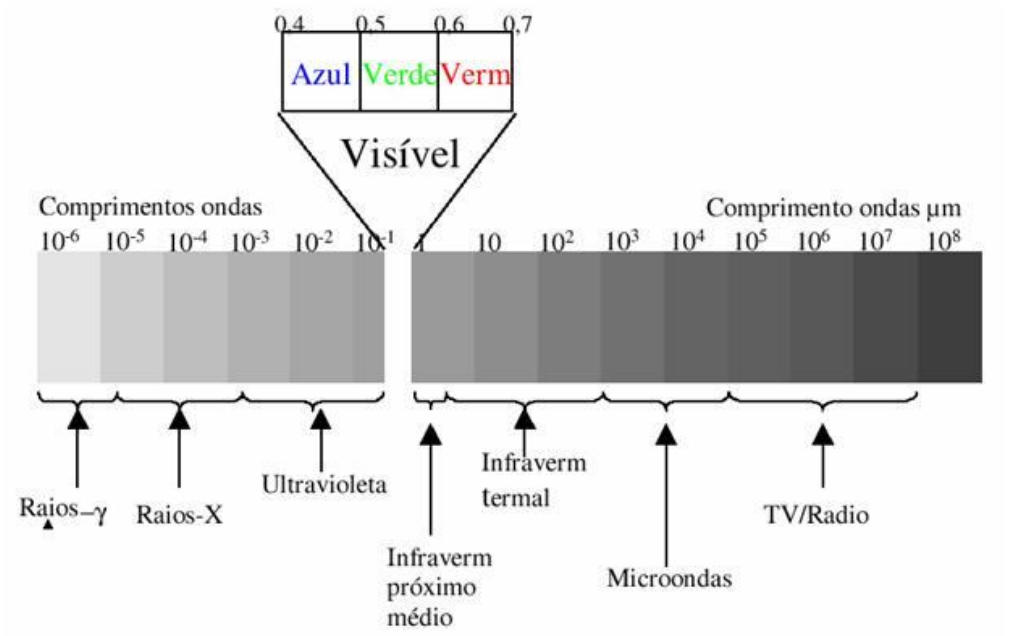


Figura A.1 - Espectro Eletromagnético.
Fonte: Figueiredo, 2005.

O espectro de cores pode ser dividido em seis grandes regiões: violeta, laranja azul, verde, amarelo e vermelho e, entre estas cores, existem todas as outras. O resultado da combinação de todas estas cores é a luz branca. Isaac Newton em 1672 [Figueiredo, 1997; Silva; Martins, 2003] provou que a luz branca não é homogênea, mas sim uma composição de comprimentos de onda coloridas que, ao atravessar um prisma, se decompõe e se refrata em raios de cores que compõem o espectro [Farina, 1990].

Quando um raio de luz atinge um ponto em um objeto: parte da luz é absorvida (convertida em outras formas de energia, por exemplo, calor), parte da luz é transmitida através do objeto (por refração) e parte da luz é refletida (o que pode ser em várias direções ao mesmo tempo). A quantidade luz que é refletida depende do objeto, ou seja, depende de sua cor e de sua superfície que pode ser mais transparente, mais brilhante ou mais opaca. O grau com que os objetos refletem alguns comprimentos de onda e absorvem outros é chamado de refletância espectral que é uma propriedade constante do objeto e independe da fonte de luz [Holdschip, 2008].

APÊNDICE B - Análise de incertezas

Segundo Feliciano et. al., 2005, em aplicações de medição, sobretudo na realização de calibrações, é necessário um estudo para o cálculo da incerteza de medição, a qual expressa, em suma, o valor provável do erro em uma medição. Nessa análise são tratadas informações e critérios para avaliação dos componentes que influenciam o resultado de uma medição. Segundo BIPM, 2014, o cálculo da incerteza depende do conhecimento detalhado da natureza do mensurando, do padrão, das influências externas e do processo de medição.

Classifica-se as incertezas experimentais em aleatórias e sistemáticas. As incertezas aleatórias dependem de uma série de observações da mesma grandeza física. Nesses casos, o resultado da medição pode ser considerado como a média aritmética ($\overline{m}$) das *n* medidas efetuadas (m_n), calculada por:

$$\overline{m} = \frac{\sum m_n}{n} \tag{B.1}$$

Pode-se, então, calcular a variabilidade dos valores medidos m_n através do cálculo da variância estimada ($s^2_{m_n}$) ou do desvio padrão estimado (s_{m_n}), conforme as equações (B.2) e (B.3):

$$s^2_{m_n} = \frac{\sum(m_n - \overline{m})^2}{n-1} \tag{B.2}$$

$$s_{m_n} = \sqrt{\frac{\sum(m_n - \overline{m})^2}{n-1}} \tag{B.3}$$

Ainda, segundo Feliciano et. al., 2005, a melhor estimativa da variância experimental da grandeza a medir é a variância experimental da média ($s^2_{\overline{m}}$), cuja expressão é dada por:

$$s^2_{\overline{m}} = \frac{s^2(m_n)}{n} \tag{B.4}$$

Já, o desvio padrão experimental da média ($s_{\overline{m}}$) qualifica o quanto o valor médio ($\overline{m}$) representa a grandeza a ser medida. Quanto maior o número de medições efetuadas, maior será esta estimativa. O desvio padrão experimental da média pode ser calculado por meio de:

$$s_{\bar{m}} = \sqrt{\frac{s^2(m_n)}{n}} \tag{B.5}$$

A expressão da incerteza aleatória, determinada a partir de *n* medições de uma grandeza *X*, para *g* graus de liberdade e nível de confiança *cf*=95%, é dada por:

$$I_{A_{\bar{m}}} = \pm t_{cf_g} s_{\bar{m}} \tag{B.6}$$

onde *g=n-1* e t_{cf_g} é a distribuição *t* de *student* de Willian Gosset [Pearson et. al. 1990].

Uma alternativa para determinar a Incerteza tipo A é fotografar o padrão diversas vezes, mantendo fixas a altura da câmera em relação ao objeto e a resolução da mesma. Assim, escolhendo-se pontos diferentes do padrão pode-se calcular o desvio em relação à medição dos mesmos. Da mesma forma que em um processo de medição com um micrômetro, por exemplo, há possibilidade de se obter leituras diferentes de uma medida ao se repetir o processo algumas vezes. Contudo, no presente caso, os valores de desvio forneceram valores nulos, permitindo desconsiderar a incerteza do Tipo A. Isto deve-se ao caráter automático da seleção dos pontos, realizado pelo algoritmo de visão.

Segundo Feliciano et. al., 2005, incertezas do tipo B são determinadas a partir de informações acessórias e externas ao processo de medição. Estas informações podem ser obtidas de resultados de medições similares anteriores, experiência ou conhecimento do comportamento do instrumento, dados do fabricante, dados fornecidos por certificados de calibração, referências de manuais de instrução, etc. Como exemplos deste tipo de incerteza, têm-se: gradiente de temperatura durante a medição; afastamento da temperatura ambiente em relação à temperatura de referência, tipo do indicador: analógico ou digital, instabilidade na rede elétrica, paralaxe, incerteza do padrão, instabilidade do padrão, erros geométricos, deformações mecânicas, histerese, estabilidade temporal, etc.

No caso do sistema de visão computacional apresentado no Capítulo 3, deve-se levar em conta que as medições das dimensões do padrão de calibração consiste em um fator externo relevante que pode contribuir significativamente para o erro. Assim, ao calibrar a câmera, há uma incerteza associada ao fato de as medidas conhecidas da peça e do padrão serem obtidas por um processo de medição cuja incerteza associada irá contribuir para a incerteza da medição realizada pelo sistema de visão. Uma possibilidade para diminuir esse

efeito seria calibrar a câmera com uma foto de um bloco padrão. Neste caso, o valor da incerteza associada poderia ser obtido por meio do certificado de calibração, o que geralmente acompanha o conjunto de blocos padrão quando o mesmo é adquirido. Porém, como no método proposto a referência deveria estar afixada no robô, não foi possível utilizar blocos padrão. Desta forma, utilizou-se na análise, a resolução da impressora que gerou o padrão.

No caso da medição do padrão e das peças, utilizou-se a resolução (*res*) do paquímetro que é, segundo seu *datasheet*, de 0,05 mm (Seção 3.6.2). Como não se trata de um paquímetro digital, deve-se levar em conta possíveis erros de paralaxe e pressão no instrumento. Assim, optou-se pelo uso da metade da menor divisão da escala (*res*) desse instrumento conforme recomendam Feliciano et. al., 2005. Desta forma, utilizou-se a Equação (B.7) para o cálculo da incerteza das medidas do paquímetro em função da sua resolução.

$$I_B = \frac{res}{2} \tag{B.7}$$

Assim, a incerteza do tipo B para o sistema de visão do presente estudo é dada por:

$$I_B = \frac{0{,}05}{2} = 0{,}025\ mm \tag{B.8}$$

Para que a análise de incertezas abranja o sistema de visão integrado ao robô do presente estudo, faz-se necessário levar em conta as medições do robô. Os instrumentos utilizados para isso foram: 1 paquímetro (*res=0,05 mm*), 1 trena (*res=1 mm/m*) e os 4 sensores de posição dos atuadores possuem ($res = 5\mu m$) referentes ao 4 primeiros GL do robô e 1 encoder ($res = 20000 pulsos/rotação$). Porém, outros fatores influenciam na incerteza das medições do robô, como por exemplo, afastamento da temperatura ambiente em relação à temperatura de referência, gradiente de temperatura durante a medição, deformações mecânicas, dentre outras. Um estudo aprofundado deverá ser realizado em trabalhos futuros abordando as incertezas de medição do robô. No presente estudo, são analisadas somente as incertezas quanto ao método de visão computacional.

Segundo Berendsen, 2001, Uma possível maneira de se estimar a incerteza final (δI), pode ser obtida através da chamada combinação da pior situação. Assim,

$$\delta I = \left|\frac{\delta I}{\delta x_1}\delta x_1\right| + \left|\frac{\delta I}{\delta x_2}\delta x_2\right| + \cdots + \left|\frac{\delta I}{\delta x_i}\delta x_i\right| \tag{B.8}$$

onde, δI é a incerteza no resultado de medição e δx_i é a incerteza em cada variável x_i.

As derivadas parciais são chamadas de coeficientes de sensibilidade e medem o quão sensível o resultado (I) é a cada variável x_i. Porém, é geralmente pouco provável que se combinem da pior maneira possível como propõe a expressão acima. As previsões obtidas com esta expressão são normalmente exageradas quando comparadas com experimentos reais. Kline e McClintock, 1953, propuseram uma outra forma de calcular a propagação de incertezas experimentais que fornece boa previsão das incertezas:

$$\delta I = \sqrt{\left(\frac{\delta I}{\delta x_1}\delta x_1\right)^2 + \left(\frac{\delta I}{\delta x_2}\delta x_2\right)^2 + \cdots + \left(\frac{\delta I}{\delta x_i}\delta x_i\right)^2} \tag{B.9}$$

Conforme visto, têm-se duas incertezas no sistema de visão computacional proposto, uma do tipo A (incerteza no procedimento de medição computacional) e uma do tipo B (medida do padrão de referência). No presente estudo, para o sistema de visão proposto, pode-se simplificar o cálculo das incertezas combinadas da Equação (B.9) por:

$$I_C = \sqrt{(I_A)^2 + (I_B)^2} \tag{B.10}$$

Desta forma, pode-se determinar as incertezas combinadas por:

$$I_C = \sqrt{(0)^2 + (0{,}025)^2} = 0{,}025\ mm \tag{B.11}$$