Universidade Federal de Campina Grande

Centro de Engenharia Elétrica e Informática

Programa de Pós-Graduação em Engenharia Elétrica

Planejamento de Trajetória Baseado em Visão Computacional para Manipulador Robótico

Fernando Javier Mendiburu

Dissertação de Mestrado

Campina Grande

6 de agosto de 2013

Universidade Federal de Campina Grande

Centro de Engenharia Elétrica e Informática

Programa de Pós-Graduação em Engenharia Elétrica

Fernando Javier Mendiburu

Planejamento de Trajetória Baseado em Visão Computacional para Manipulador Robótico

Dissertação de Mestrado submetida à Coordenação do Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Campina Grande - Campus de Campina Grande como parte dos requisítos necessários para a obtenção do grau de Mestre em ciências no Domínio de Engenharia Elétrica.

Área de concentração: Processamento da Informação

Antonio Marcus Nogueira Lima

Orientador

Marcos Ricardo Alcântara Morais

Orientador

Campina Grande, Paraíba, Brasil

Dedicatória

Dedicação feita especialmente a toda a família Mendiburu-Molina.

Aos meus pais, Elsa Haydée Molina e Luis Alberto Mendiburu, por me darem a possibilidade de estudar, abrindo as portas do conhecimento e dando-me novos horizontes.

Aos meus irmãos, Mariana Alicia Mendiburu e Alejandro Luis Mendiburu por ser meus parceiros e amigos da vida toda.

À minha namorada Angélica Oliveira Velozo que acreditou em mim, me ajudou nos momentos mais complicados e me deu confiança.

Agradecimentos

Agradeço inicialmente a Deus, por todas as oportunidades me dadas durante a vida.

Agradeço aos meus pais, Elsa e Luis, meus irmãos Mariana e Alejandro que torceram por mim a distância e minha namorada Angélica, pelo constante apoio e incondicional crença em meu trabalho, dando-me forças nos momentos difíceis.

Aos meus orientadores, Antonio Marcus Nogueira Lima e Marcos Ricardo Alcântara Morais, por todo conhecimento repassado, pela orientação, amabilidade e conselhos dados ao longo do trabalho. Manifesto gratidão também a Walter Andrés Vermehren Valenzuela por estar sempre disposto a oferecer qualquer ajuda e Simões Soares de Toledo pelo apoio técnico.

Aos colegas e amigos conquistados no Laboratório de Ensaios, Manutenção, Calibração, Aferição e Desenvolvimento - LEMCAD, Laboratório de Instrumentação Eletrônica e Controle - LIEC e na Universidade: Yago Monteiro, Leiva Casemiro Oliveira, Antonio Agripino e Tony Carlos Moura Cavalcanti por estarem sempre presentes e dispostos oferecer qualquer ajuda. Agradecimentos especiais para Ramsés Araújo Gonçalves quem me ajudou na parte técnica, moral e na organização ao longo do meu mestrado estando comigo nos momentos mais difíceis.

Ao CNPq pela concessão da bolsa que possibilitou minha dedicação integral a este projeto.

Resumo

Na presente dissertação aborda-se a integração de um sistema de sensoriamento e um método de planejamento de trajetórias para um robô manipulador de objetos. O sistema tem como objetivo a manipulação inteligente de peças, gerando trajetórias livres de obstáculos sempre que for possível. O espaço de trabalho é determinado por meio de um sistema de visão computacional, que captura a nuvem de pontos da cena realizando uma modelagem dos obstáculos para ser fornecido ao algoritmo de roteamento. A modelagem do manipulador é feita *a priori*, e é determinada se a sua cinemática pode gerar a trajetória livre de colisões. A modelagem e calibração dos dispositivos são feitas, removendo a distorção da lente e determinando os parâmetros intrínsecos das mesmas. A relação das unidades de medida da câmera e as unidades do mundo físico é uma componente crítica em qualquer intento por reconstruir uma cena tridimensional. A calibração entre o sistema de referência do kinect e da base do manipulador é realizada para manter uma coerência nas medidas e para o manipulador comandar os movimentos com boa precisão. A integração das informações das malhas de visão com o planejador de trajetória é importante para definir a tarefa corretamente, para isto deve-se adaptar o algoritmo de roteamento, transformando as configurações de entrada a serem lidas pelo planejador e suas configurações de saída para serem enviadas ao controlador do manipulador. Na solução proposta, foi utilizado o manipulador robótico Pegasus 880-RA2-1-B, um sensor de visão RGB-D instalado externamente ao braço e a câmera monocular CMOS IR-Syntec instalada no efetuador do manipulador. Foram observados resultados positivos na aplicação de processamento e na calibração das câmeras. O transporte da peça ao executar a tarefa definida pelo planejador de trajetória foi realizada com sucesso apresentando bons resultados com diversos obstáculos entre a posição do objeto e o alvo final. Foram obtidos os tempos de execução de cada movimento e de cada bloco do sistema de controle. Os erros no sistema de visão foram calculados e não influem na posição da garra em curtas distâncias. Conseguiu-se boa resposta do sistema de controle que encontra uma trajetória livre de obstáculos, sendo levada a cabo satisfatoriamente. Observa-se maior precisão na detecção do objeto, robô e obstáculos quanto mais perto do dispositivo RGB-D. O sistema apresenta-se robusto a mudanças na iluminação que poderiam degradar a performance do sistema pelo uso de sistemas de visão infravermelhos.

Abstract

In this dissertation the integration of a sensing system and a path planning for object handler robot is addressed. The system objective is the intelligent part handling, generating free trajectories when possible. The workspace is defined by a computational vision system that captures a point cloud for the scene performing modeling of obstacles to be provided to the routing algorithm. The manipulator modelling is performed a priori, and it is determined if the cinematic can generate a collision free trajectory. Device modelling and calibration is realized, removing the lens distortions and determining their intrinsic parameters. The relation between the camera measurement unities and the physical world unities is a critical component in any attempt to reconstruct a tridimensional scene. The calibration between the kinect reference system and the manipulator base is performed to maintain some coherence in the measurements and to obtain good precision in the handler commands. The integration of the vision loops with the trajectory planner is important to define the task correctly, and for it must adapt the routing algorithm, transforming the input settings to be read by the planner and its output settings to be sent to the controller of the manipulator. The robotic manipulator Pegasus 880-RA2-1-B was used in the proposed solution, together with an external RGB-D sensor and a monocular CMOS IR-Syntek camera installed in the manipulator effector. Positive results were observed in the application of the processing and in the cameras calibration. The part transportation by the task execution was defined by the trajectory planner and was successfully performed with good results with several obstacles between the object position and the final target. The execution time for each movement and for each control system block were obtained. The vision system errors were calculated and they don't influence in the grasp position in short distances. A good response for the system was obtained for the control system that finds an obstacle free trajectory, being performed satisfactorily. Greater precision in the object, robot and obstacles detection was observed the closer to RGB-D device they are. The system shows robustness to changes in the illumination that could degrade the system performance by using infrared vision systems.

Lista de Figuras

Lista de Tabelas

Lista de Siglas e Acrônimos

CMOS	Complementary Metal–Oxide–Semiconductor (Semicondutor metal-óxido complementar).
RGB	Red-Green-Blue (Vermelho-verde-azul).
RGB-D	Red-Green-Blue + Depth (Vermelho-verde-azul + profundidade).
IR	Infrared (Infravermelho).
PRM	Probabilistic Roadmap Method (Método de roteamento probabilístico).
SBL	Single-query / Bidirectional / Lazy (Único par de entrada / Bidirecional / Preguiçoso).
SCARA	Selective Compliance Assembly Robot Arm (Braço robótico de montagem seletiva).
PCL	Point Cloud Library (Biblioteca de nuvem de pontos).
CCD	Charge-Coupled Device (Dispositivo de carga acoplada).
CAD	Computer-Aided Detection (Desenho assistido por computador).
SMA	Shape Memory Alloys (Ligas de memória de forma).
PID	Controlador Proporcional, Integral e Derivativo.
PLC	Programmable Logic Controller (Controlador lógico programável).
I/O	Imput / Output (Entrada / Saída).
MCL	Motion Control Language (Linguagem de programação para manipuladores).
USB	Universal Serial Bus (Barramento serial universal).
VRML	Virtual Reality Modeling Language (Linguagem para Modelagem de Realidade Virtual).
API	Application Programming Interface (Interface de programação de aplicativos).
Teach Pendant	Joystick (Controle portátil).

Lista de Símbolos

i, j, k, l	Índices das equações.
Im_D	Imagem de profundidade do sensor RGB-D.
Im_{RGB}	Imagem colorida na paleta RGB.
W_{conf}	Matriz contendo a pose inicial e final do efetuador.
o_i	Matriz contendo os pontos (nuvem de pontos) que representam os obstáculos.
M_{iv}	Malha de triângulos dos obstáculos o_i.
q_{conf}	Vetor de configuração contendo a pose inicial e final do efetuador no espaço de juntas.
q_{task}	Vetor de configuração com a tarefa a ser realizada pelo manipulador.
q_e	Vetor de configuração que representa q_{task} com valores de encoders.
q_{exp}	Vetor de configuração que representa q_{task} com valores de encoders. e adiciona 180 rotaçoes de θ_5.
c, y	Comandos recebidos pelo controlador do manipulador.
e	Comandos enviados do controlador ao manipulador.
Im	Imagem de saída da câmera monocular do braço do manipulador.
f_t	Características da imagem (conjunto de pixels que determinam a região).
z_0, z_1, z_2	Eixo da base, ombro e cotovelo do manipulador.
z_3, z_4, z_6	Eixo de inclinaçao, rolagem e direção do efetuador do manipulador.
θ_1, θ_2, θ_3	Ângulo de rotação da junta base, ombro e cotovelo do manipulador.
θ_4, θ_5, θ_6	Ângulo de inclinaçao, rolagem e direção do efetuador do manipulador.
D_1, D_2, D_3	Deslocamento da junta prismática da base, ombro e cotovelo no eixo z_0, z_1 e z_2, respectivamente.
q	Vetor de configuração do manipulador.
$W_\gamma^{'}$, $W_\delta^{'}$	Poses de carregamento e descarregamento.

W_γ, W_δ	Poses próximas as poses de carregamento e descarregamento.
η, γ	Trajetórias lineares no espaço cartesiano.
$\mathbb{C}$	Espaço total de configuração ou espaço de trabalho do manipulador.
$\mathbb{C}\mathscr{O}$	Conjunto de obstáculos no espaço de configuração.
$\mathbb{F}$	Espaço livre de configuração do manipulador.
$\mathscr{O}_i$	Obstáculo *i* representado no espaço de configuração.
$\theta_{mín}$, $\theta_{máx}$	Limiar mínimo e máximo de uma junta determinada.
n	Quantidade de graus de liberdade do braço.
$d(q,q')$	distância entre a configuração q e q'.
r	Raio.
$B(q,r)$	Vizinhança da configuração q de raio *r*.
$q_{inicial}$	Configuração inicial (ubicação do objeto de interesse).
$q_{objetivo}$	Configuração objetivo (ubicação desejada final do objeto de interesse).
m, $q_{proxima}$	Configurações livres de obstáculos.
q', $q_{aleatoria}$	Configurações aleatórias.
$T_{inicial}$	Árvore gerada a partir de $q_{inicial}$.
$T_{objetivo}$	Árvore gerada a partir de $q_{objetivo}$.
$\pi(T)$	Probabilidade de escolha de $T_{inicial}$.
$\pi(m)$	Probabilidade de escolha de m.
ε	Resolução para avaliar colisões em um segmento u.
ρ	Distância entre configurações.
s	Quantidade máxima de configurações m.
w	Segmento que une $T_{inicial}$ com $T_{objetivo}$.
u	Segmento que une duas configurações.
U	Conjunto de segmentos u não avaliados de colisão.
$k(u)+1$	Quantidade de configurações intermediárias do segmento u.
$\lambda(u)$	Distância normalizada no espaço de configuração do segmento u.
L_∞	Norma infinito da distância linear $d(q,q')$.
d_k	Comprimento da articulação k.
a_k, α_k	Comprimento e ângulo parâmetro do elo, respectivamente.
q_{home}	Configuração inicial do manipulador.
L_k	Trama k de eixos (x_k, y_k, z_k).
R, T	Orientação e posição do efetuador em função do sistema de referencia da base do manipulador.

W, $\tilde{W}$	Pose da ferramenta do manipulador.
q_n	Vetor de configuração com valores de ângulos normalizados.
θ_{ni}, θ_{ei}	Componente i do vetor de configuraçao q_n e q_e, respectivamente.
$Im(x,y)$	Imagem representada no plano xy.
$g(i,l)$	Imagem filtrada.
$h(k,l)$	Máscara de tamanho $K \times L$ do filtro da média.
$h_a(k,l)$, $h_f(k,l)$	Máscara de tamanho $K \times L$ da abertura e fechamento, respectivamente.
τ	Limiar da técnica de limiarização.
$\nabla(x,y)$	Gradiente da imagem $Im(x,y)$.
$\triangle_x$ e $\triangle_y$	Incremento na variável x e y, respectivamente.
$\nabla_M(Im(x,y))$	Gradiente morfológico da imagem $Im(x,y)$.
τ_g	Limiar da técnica de gradiente morfológico.
$m_{k,i}$	Momento de ordem (k,i) de um objeto.
$\mu_{k,i}$	Momento central de ordem (k,i) de um objeto.
$\eta_{k,i}$	Momento normalizado de ordem (k,i) de um objeto.
σ	Centro de massa do objeto $(\bar{x}, \bar{y}, \bar{z})$ $[cm]$.
$o_{im}(u,v)$	Sistema de referência da imagem.
$o_c(\bar{X},\bar{Y},\bar{Z})$	Sistema de referência da câmera.
f	Distância focal $[m]$.
(c_x,c_y)	Translação do sistema de referência da câmera $[pixel]$.
(k_x,k_y)	Fatores de conversão dos pixels $\left[\frac{pixel}{m}\right]$.
k_1, k_2 e k_3	Parâmetros de distorção radial da lente.
p_1 e p_2	Parâmetros de distorção tangencial da lente.
α, β	Ângulo entre o eixo x (y) e a distância focal f, respectivamente.
$p = (x,y,f)$	Ponto em pixels na imagem da câmera.
$P = (X,Y,Z)$	Ponto no espaço tridimensional.
x_{corrig}, y_{corrig}	Coordenadas da imagem sem distorção.
b	Distância entre dispositivos na calibração estéreo.
d_{offset}	Disparidade de deslocamento do sensor Kinect.
d_k	Disparidade do sensor Kinect.
K	Matriz de calibração da lente.
$q_m(t)$, $q_{task}(t)$	Trajetórias continuas definidas pelo controlador do manipulador.
R_{cs}, T_{cs}	Matriz de rotação e translação entre o sensor RGB e CMOS de sensor RGB-D.

q_α, q_β	Configurações aleatórias do manipulador.
${}^{cam}_{bas}H$	Matriz homogênea de transformação entre a câmera e a base do braço.
ϑ	Número de bits da representação de cada pixels na imagem em escala de cinzas.
r_1, r_2, r_3	Vetores coluna da matriz R.
ε_p, ε_o	Erro de posicionamento e de orientação do efetuador.
σ_e, σ_v	Centro de massa estimado e verdadeiro do objeto.

Sumário

Capítulo 1

Introdução

O desejo do homem de criar vida artificial através de máquinas surgiu, inicialmente, da necessidade da simplificação das tarefas cotidianas. Essa idéia foi de fato implementada em maior escala na indústria, onde o trabalho manual foi sendo paulatinamente substituído pela mecanização dos processos produtivos. Assim, as máquinas simplificaram atividades laboriosas que antes eram feitas pelos seres humanos. Tem-se como exemplo as tarefas que exigiam do homem mais trabalho físico diante da dificuldade de transportar peso. Pensando nisso surgiram máquinas de grande porte com o propósito de agilizar os processos.

A realização de tarefas que exigiam do homem precisão, repetitividade e esforço ficaram mais ao cargo dos robôs e das máquinas. Por outro lado, as tarefas de decisão e monitoramento nos processos de automação ficaram sobre a responsabilidade do homem. O que foi mencionado anteriormente marca como a robótica tem ganhado importância nos últimos tempos. A tendência atual é criar máquinas de peso reduzido, com atuadores de elevada relação força-peso, capazes de se adaptar ao espaço de trabalho, ter diferentes habilidades como andar, manipular peças, evitar obstáculos e ter baixo consumo de energia.

A robótica e o uso de manipuladores tem ganhado terreno pelas vantagens que representa o seu uso no ambiente industrial. Um dos principais fatores que determinaram esse avanço foi que a tecnologia robótica pode ser implementada em campos cada vez maiores. A utilização de manipuladores na indústria permite a melhora da qualidade dos produtos, favorece a produção em série, eleva a quantidade de unidades produzidas, reduz o tempo e o custo de produção. A melhoria de qualidade é oferecida pelas características de alta repetitividade e precisão dos robôs.

A implementação de algoritmos de roteamento é importante porque permite gerar trajetórias

livres de obstáculos para a manipulação de diversas peças, além de ter outras características desejáveis como a geração de trajetórias de menor distância que promovam baixo consumo de energia. Os algoritmos são utilizados no manipulador para processos produtivos que requerem movimentação de cargas, para transformação de materiais e para realizar medições diversas no espaço de trabalho [1].

Na indústria, existem espaços de trabalho onde as peças a ser manipuladas, transformadas ou medidas encontram-se de forma desordenada em uma posição desconhecida. Então para atingir sistemas de alto desempenho, sensores exteroceptivos, ou externos, são implementados na plataforma para visualizar, detectar e reconher a peça. A integração desses sensores, como câmeras no manipulador, permite dar uma maior versatilidade, pois podem ser feitas medidas no ambiente onde o robô opera, assim a gama de aplicações aumenta significativamente [2]. Tendo isso em consideração, a informação visual obtida do espaço de trabalho pode ser empregada pelo sistema de controle para gerar um planejamento de trajetória em direção a um objetivo sem colisões no caminho. Então, a implementação de um sistema de visão computacional no manipulador permite obter maior inteligência na execução das tarefas, dando uma maior confiabilidade ao processo realizado e desenvolvendo uma automação mais eficiente.

1.1 Motivação

A necessidade de aplicação de um método de planejamento de trajetória, para o manipulador atingir o alvo sem colidir com obstáculos, é a causa principal que motiva o projeto. Na tentativa de dar soluções a problemas de relevância industrial, decidiu-se de implementar um algoritmo de roteamento livre de obstáculos presentes no espaço de trabalho usando um sensor externo à plataforma. A maneira para obter a informação de interesse no planejamento foi implementar um sistema de visão computacional, capaz de capturar a área de trabalho do manipulador. O sistema é apto a interpretar a imagem e transmitir os comandos necessários para que o robô execute a tarefa.

Apesar do fato de que na indústria geralmente os sistemas robóticos são submetidos ao emprego pré-programado em relacão à peça a ser manipulada sem utilização de sensores externos e com conhecimento *a priori* do espaço de trabalho, optou-se por desenvolver uma solução que englobe situações mais gerais permitindo assim, uma maior flexibilidade dado um caso particular. Vislumbrando-se a possibilidade de aplicação de ferramentas de visão robótica, automação e controle em um manipulador de alta precisão com atuadores lineares tais como servomotores, capaz de adaptar-se inteligentemente às mudanças de situações, ajustável e expansível na

quantidade de sensores, adequado para otimizar os processos de manipulação com uma grande exatidão, apoia a importância de adicionar visão no sistema e justifica o motor da pesquisa neste projeto. Então, optou-se pela abordagem deste tema, visando desenvolver e aplicar um sistema de visão em tarefas de manipulação. Para tanto, busca-se a integração de equipamentos de vídeo encontrados no mercado e integrá-los a um computador para o processamento da imagem. Em termos mais específicos, apesar do emprego de robôs indústrias estarem plenamente estabelecidos em tarefas repetitivas e que dependam apenas de uma trajetória preconcebida, às vezes é necessário executar as tarefas mais inteligentemente. Esse tipo de sistema será o assunto de estudo neste trabalho.

A demanda da indústria e da sociedade de desenvolver tecnologia útil e contribuir com as atuais é o motor que fundamenta o projeto, sendo um caminho para futuras pesquisas e desenvolvimentos em robótica na universidade.

1.2 Objetivos

A meta deste trabalho é implementar um algoritmo probabilístico de roteamento e sistema de visão computacional no manipulador para gerar uma trajetória livre de obstáculos. O projeto é de fácil implementação, em termos de integração e compatibilidade de equipamentos quanto em simplicidade na operação. Os objetivos específicos são:

- Geração do espaço de trabalho: aquisição e conversão da nuvem de pontos dos obstáculos e objeto de interesse provenientes do sensor RGB-D em malhas de triângulos;
- Adaptação do algoritmo de roteamento, transformando as configurações de entrada a serem lidas pelo planejador e suas configurações de saída para serem enviadas ao controlador do manipulador.
- Definição da cinemática e modelagem do manipulador;
- Calibração das câmeras, removendo a distorção da lente e determinação dos parâmetros intrínsecos das mesmas;
- Calibração entre o sistema de referência do sensor RGB-D e da base do manipulador;
- Programação das técnicas e integração das malhas de visão;
- Integração dos dados obtidos das câmeras com o planejador de trajetórias.

1.3 Estrutura do documento

O conteúdo deste trabalho está dividido em capítulos e como introdução é apresentada uma breve descrição da robótica. Também é explicada a importância do uso de manipuladores e a relevância da integração de um sistema de visão computacional com um planejador de trajetórias. Em seguida são apresentadas a motivação e objetivos do presente projeto.

No Capítulo 2 realiza-se uma revisão bibliográfica fazendo-se alusão nos trabalhos de planejamento de trajetória e visão computacional. Após isso, apresenta-se o sistema de controle proposto.

No Capítulo 3 são apresentados conceitos básicos de manipuladores, classificação e aplicações na indústria. É explicada a relevância dos algoritmos e o algoritmo probabilístico implementado no projeto. Também são exibidas as características do manipulador usado no projeto e deduzidas as equações da cinemática direta e inversa da estrutura. Finalmente é feita a modelagem do manipulador e encontradas as equações que relacionam os ângulos das juntas com os valores de *encoders* que devem ser fornecidos aos motores para produzir o movimento.

No Capítulo 4 apresentam-se as técnicas de processamento de imagens, a utilização de filtros para em seguida abordar a segmentação da imagem. É feita uma abordagem do modelo pontual de câmera, a calibração das mesmas com a respectiva obtenção dos parâmetros intrínsecos e de distorção para finalmente apresentar os passos que devem ser realizados para uma calibração estéreo satisfatória.

O sistema de controle definido no projeto é abordado no Capítulo 5, onde é utilizada toda a análise teórica abordada nos anteriores capítulos. São definidas as malhas de controle necessárias para desenvolver a tarefa satisfatoriamente.

Os resultados obtidos na dissertação são exibidos no Capítulo 6. Apresentam-se os resultados da calibração dos sensores de visão, dos sistemas de referência e da trajetória realizada. Finalmente um análise de erros e tempo de execução das tarefas são realizados.

No último capítulo do trabalho são abordadas as sugestões para trabalhos futuros e as conclusões finais do projeto.

Capítulo 2

Realimentação visual e planejamento

Há uma grande variedade de robôs manipuladores nos quais tem se implementado técnicas de planejamento de trajetórias com diversos tipos de sensores para a descrição do espaço de trabalho. Os dois tipos de planejamento mais conhecidos realizam um roteamento probabilístico ou determinístico [2]. Sabendo que o segundo tipo de roteamento possui um custo computacional maior, em geral não são usados no ambiente industrial. A tendência nos manipuladores é usar algoritmos probabilísticos de fácil implementação, de baixo custo computacional, que respondam a requisitos em tempo real, além de outras características desejadas. Por conseguinte, realiza-se uma revisão bibliográfica dando maior relevância aos algoritmos probabilísticos. Também o estudo de sensores de visão foi restrito a dispositivos de captura de imagens, para ser de suporte e ajuda no planejamento da trajetória do manipulador. A seguir apresentam-se os trabalhos relacionados na área de manipulação de objetos que são de importância para definir as características do trabalho proposto nessa dissertação.

Mikawa *et al* [3] apresenta o controle de manipuladores redundantes baseado em visão estéreo, que permite evitar obstáculos além de posicionar o efetuador em uma posição objetivo propondo um método para estimar características da imagem do manipulador que não podem ser obtidas da mesma. Por exemplo, quando tem-se obstrução por obstáculos. As características são estimadas dos parâmetros intrínsecos da câmera e da cinemática do manipulador.

Wong *et al* [4] apresenta um sistema de visão para o planejamento de trajetória de robôs manipuladores. Usa um sistema de visão tridimensional para determinar a posição relativa dos objetos a serem manipulados e dos obstáculos que devem ser evitados. A partir de imagens de intensidade adquiridas por uma câmera CCD montada sobre o braço do robô, as principais características dos objetos são detectadas para posterior agrupamento (classificação). Através das correspondências entre os agrupamentos e as características do modelo, a posição e orien-

tação dos objetos e obstáculos são determinadas e confirmadas pela reprojeção dos pontos na imagem. Uma vez que essas posições são determinadas desenvolve-se uma trajetória que evita obstáculos.

Silva [5] aborda o controle visual de manipuladores em tempo real para a interseção de objetos em movimento na área de trabalho do robô. Utiliza uma câmera instalada no efetuador do braço que permite identificar o objeto e determinar a sua posição espacial utilizando algoritmos de reconstrução tridimensionais e preditores de trajetórias do objeto. As limitações do projeto são que a área de trabalho é reduzida por uma restrição conservadora das juntas do manipulador e que o par estéreo deve ser recalibrado manualmente ante uma variação da posição relativa das câmeras. Um trabalho similar foi feito por Denker *et al* [6] no reconhecimento, rastreamento e interseção de objetos em uma esteira. Também, como Silva, utiliza uma câmera instalada no efetuador do braço. O reconhecimento dos objetos é feito analisando as mudanças no histograma da imagem, logo quando o objeto é detectado usa-se o gradiente para encontrar bordas e extrair parâmetros das características da imagem, os quais são, por exemplo, o número de cantos, o número de ângulos em uma determinada faixa, etc. Finalmente é feita a classificação do objeto encontrado e usando um algoritmo de redes neurais permite realizar a previsão da trajetória.

Leutert *et al* [7] desenvolveu um sistema de planejamento de trajetória que evita obstáculos de forma dinâmica. O propósito do projeto é permitir a colaboração manipulador-humano de uma forma mais segura e eficiente. Inicialmente é determinada a pose das câmeras em relação ao braço. Depois de ser conhecidas com exatidão, as imagens das câmeras são filtradas, para refinamento e redução de ruído. Depois, é possível realizar um modelo do ambiente de trabalho, do objeto e do robô tendo em conta primitivas geométricas, tais como esferas ou paralelepípedos. O modelo do robô é desenvolvido utilizando sua cinemática direta. O planejamento de trajetória utilizado é determinístico para um ambiente não estruturado e dinâmico (com movimento de obstáculos). O ambiente de trabalho é dividido em pequenas células, as quais podem estar livres, ocupadas por objetos ou ser uma combinação de ambas. Com esses dados disponíveis, o planejador de trajetória vai avaliando pontos possíveis ao mesmo tempo que avalia se existem colisões com obstáculos. Esses pontos são fornecidos ao sistema de controle do manipulador que atualiza o modelo do robô e de ambiente de trabalho. Uma das limitações do trabalho é que o conhecimento preciso das regiões livres de obstáculos e as regiões ocupadas por eles é necessário, o que aumenta o custo computacional.

Flandin *et al* [8] desenvolveu um sistema de visão usando uma configuração que aproveita as características de uma câmera montada no efetuador e outra externamente observando o es-

paço de trabalho. A primeira câmera obtêm uma visão local que permite determinar a orientação da garra em diz respeito ao objeto. A câmera exterior, que é fixada, permite obter a posição. Utiliza o jacobiano da imagem (matriz de interação) que relaciona as características da imagem com a lei de controle usada.

Guo *et al* [9] analisa as desvantagens de um controle em tempo real de manipuladores baseado em visão, concluindo que o processamento de imagem não é exato nem rápido e o movimento do braço não é suave. Assim, apresenta soluções melhoradas para obter melhor eficiência na adquisição das imagens e uma rápida resposta no sistema de controle. Faz a fusão um sistema de visão estéreo com uma câmera montada no efetuador do manipulador em uma configuração híbrida.

Rakprayoon *et al* [10] apresenta um método de visão computacional para distinguir manipulador de obstáculos no espaço de trabalho usando um dispositivo RGB-D. Foi feita a calibração do sensor e a modelagem do manipulador da Bosch de seis graus de liberdade. O sensor RGB-D obtém a nuvem de pontos do espaço de trabalho de onde deve-se retirar a nuvem de pontos do manipulador através do seu modelo. O artigo apresenta bons resultados quando não existe oclusão de objetos.

Sabendo da dificuldade de implementação de algoritmos de roteamento por causa da resolução da cinemática inversa de um manipulador redundante, Vande Weghe *et al* [11] implementou um algoritmo probabilístico em um manipulador redundante. Baseado no algoritmo RRT de LaValle *et al* [12], permite a geração de soluções sem necessidade da utilização da cinemática inversa da estrutura. O método resulta ser muito eficiente pois utiliza a transposta do jacobiano em vez de a inversa reduzindo o custo computacional. Também, o crescimento dos galhos do caminho é feito em direção ao objetivo a ser atingido, aumentando a velocidade da convergência do método. O enfoque similar do problema é também abordado por Bertram *et al* [13].

Wang *et al* [14] aborda o problema de planejamento de trajetória, propondo um algoritmo de duas etapas que também permite resolver implicitamente a cinemática inversa de manipuladores redundantes, pois emprega uma variante do algoritmo RRT bidirecional. As duas etapas do método são baseadas nas propostas de LaValle *et al* [12] e Kuffner *et al* [15]. O método cria identificações de áreas inativas para as regiões que apresentam problemas de mínimos locais, conseguindo superar esses problemas apresentados nos algoritmos de Bertram *et al* [13] e Vande Weghe *et al* [11].

Liu *et al* [16] apresentam um planejador PRM dinâmico para manipuladores e obstáculos móveis com o propósito de atingir bom desempenho em espaços de trabalhos complexos. O roteador de trajetória deve replanejar frequentemente para encontrar uma nova solução possível

tendo em conta a dinâmica dos obstáculos. Foram feitas simulações com dois manipuladores móveis e observou-se uma redução do tempo consumido no planejamento de trajetória.

Guernane *et al* [17] aplica uma variante do algoritmo PRM, denominado SBL de um único par de entrada, bidirecional e preguiçoso na detecção de colisões reduzindo o tempo de cálculo. O algoritmo foi implementando no manipulador Motoman SV3X de seis graus de liberdade. O CAD do manipulador e a modelagem do espaço de trabalho foram feitas para simulações. São utilizadas técnicas para a otimização e suavização dos caminhos encontrados.

Dos trabalhos apresentados, foi observado que os métodos de planejamento de trajetória probabilísticos são mais eficientes que os determinísticos, não precisam de conhecimento e cálculo exaustivo das células (regiões) livres de obstáculos e a velocidade de convergência é maior. Este trabalho vai se orientar na direção de métodos de roteamento probabilísticos. Dessa forma, é implementado o algoritmo baseado em expansão e conexão de caminhos chamado algoritmo SBL, proposto por Sanchez *et al* [18]. A escolha se fundamenta na sua ampla aplicação em uma grande variedade de problemas. O algoritmo é adaptável a diversas situações, expansível e já foi usado como algoritmo de roteamento para manipuladores [16, 17]. É eficiente também desde o ponto de vista da procura dos pontos livres de colisão, pois tem uma tendência a explorar regiões inexploradas anteriormente. O sistema de visão utilizado será redundante com um sensor RGB-D e uma câmera IR monocular, colocados fora da plataforma robótica e no efetuador do braço, respectivamente. A calibração dos dispositivos será feita para obter melhores resultados na definição dos parâmetros dos objetos e fazer uma estimativa da profundidade com menor erro. Como foi observado nos artigos, uma modelagem dos objetos, obstáculos e manipulador deve ser feita.

Na Figura 2.1 observa-se o sistema de controle proposto. As imagens obtidas pelo sensor RGB-D como da câmera monocular CMOS são de grande importância pois permitem encontrar a pose dos objetos de interesse. A interação das técnicas de visão estéreo, processamento, segmentação, interpretação de imagens e planejamento de movimento definem as malhas de controle. Observa-se que o sistema forma três malhas, uma primeira malha interna de controle de juntas, a segunda de visão monocular e a terceira de visão e planejamento.

A malha de visão da câmera monocular permite determinar a orientação do objeto de interesse. Técnicas de filtragem na imagem para a redução de ruído, segmentação para a detecção de objetos e interpretação da imagem são implementadas nesses blocos para a extração dos parâmetros das características da imagem.

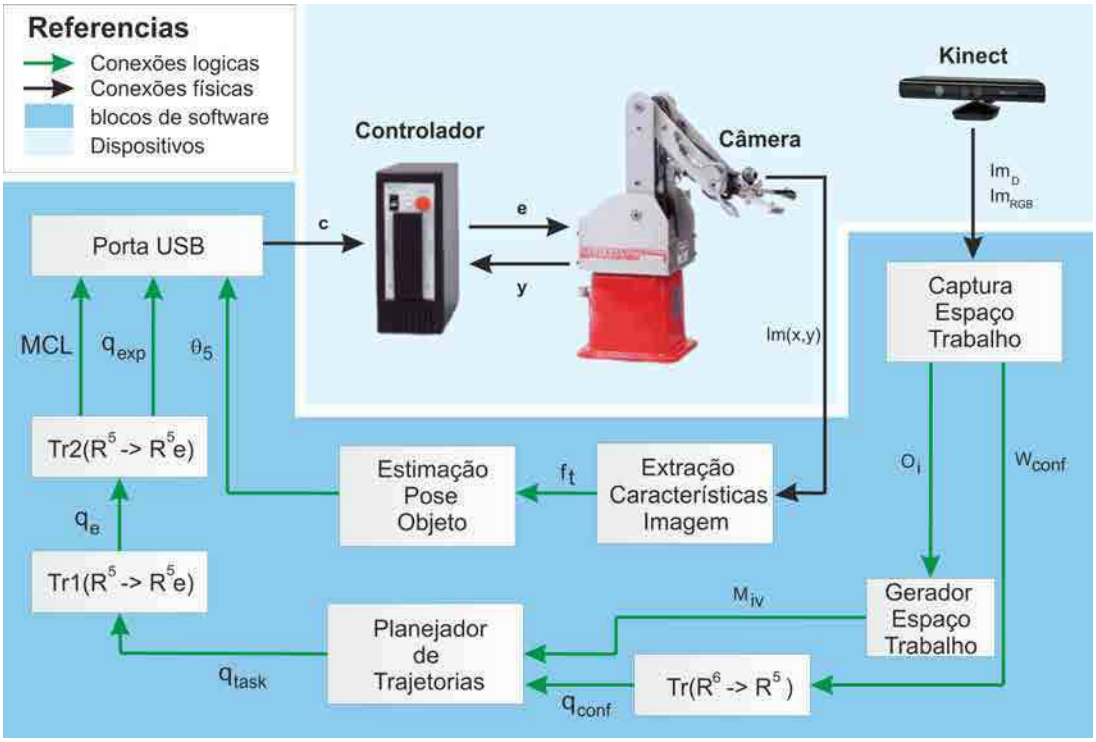


Figura 2.1: Sistema proposto apresentando os módulos e malhas de controle.

A malha de planejamento e visão externa com o sensor RGB-D permite encontrar a posição do objeto de interesse a partir da nuvem de pontos do espaço de trabalho e posteriormente fazer a transformação para malha de triângulos com o algoritmo de Delaunay. Além disto, usa a cinemática do manipulador para transformar a posição cartesiana do objeto de interesse e a posição final do efetuador em valores de juntas no espaço de configuração do manipulador para ser fornecidas para o planejamento de trajetória. O método de roteamento probabilístico usado foi o algoritmo SBL [18] que possui a característica de ser um algoritmo de duas entradas que constrói duas árvores a partir destas posições com um método de detecção de colisões preguiçoso no sentido de que os testes de colisão são adiados até que seu cálculo seja totalmente necessário. O SBL é um algoritmo de roteamento de exploração aleatória com rápida convergência e custo computacional baixo. O sistema tem como objetivo a manipulação inteligente de diversas peças, gerando trajetórias livres de obstáculos sempre que for possível. Na malha realiza-se também a conversão da configuração tarefa (q_{task}) a valores de *encoders* (q_{exp}) para serem executadas pelo controlador do manipulador. Tanto os blocos como as malhas de controle serão explicadas nas próximas seções.

A determinação do espaço de trabalho é problemática devido a complexidade e variação da iluminação na cena. A escolha do sensor RGB-D soluciona a dificuldade por ser mais robusto nesse sentido, uma vez que tem implementado além da câmera RGB um par infravermelho (câmera CMOS/ Projetor IR) que captura a profundidade da cena com maior exatidão. O sistema

apresenta-se robusto à mudanças na iluminação que podem degradar a performance do sistema. Sistemas de visão tem evoluido notoriamente desde a aparição dos dispositivos RGB-D tais como o Kinect [10].

Primeiramente foram analisadas e estudadas as caraterísticas do manipulador, tais como tipo de comunicação com o computador, em número de eixos, repetitividade, velocidade máxima e estudo dos sensores, tipos de atuadores empregados, a área de trabalho da plataforma, etc. Além disso, estudou-se a linguagem de programação utilizada no manipulador e a sua cinemática. Foi feito um estudo de técnicas de visão computacional e processamento de imagem, técnicas de modelagem e calibração de câmeras.

Em outra etapa trabalhou-se na implementação do algoritmo de roteamento no manipulador onde o modelo da cena de trabalho é estimado para a geração da trajetória livre de obstáculos. Paralelamente foi implementada a plataforma de visão computacional onde foram aplicados algoritmos de suavização, segmentação e interpretação da imagem, calibração de câmeras e sistemas de referência, etc. Finalmente foram feitas as simulações do sistema para avaliar, em princípio, o funcionamento do sistema individualmente e logo como um todo. A integração dos algoritmos com os sensores é de vital importância.

Considerando que pesquisas na área são vastas, nesta dissertação é abordado um problema complexo, relevante, atual, e de interesse principalmente industrial e acadêmico.

Capítulo 3

Manipuladores

Um manipulador é um robô multifuncional reprogramável projetado para mover materiais, partes e ferramentas através de movimentos programados variáveis realizando diversas tarefas [19]. Por ser um robô, herda as características dos mesmos: automatizado, reprogramável e multitarefa. Uma máquina automatizada é aquela que pode operar sem a necessidade da intervenção humana. É reprogramável pois pode ser programada várias vezes para executar tarefas de formas diferentes e finalmente é multitarefa pois pode ser programada para realizar tarefas diferentes e simultâneas. A estrutura mecânica de um robô manipulador é constituída por uma sequência de corpos rígidos (elos) interligadas por meio de articulações (juntas). Um manipulador é caracterizado por um braço que garanta a mobilidade, um punho que confere destreza suficiente e um efetuador que executa a tarefa exigida ao robô [2].

O manipulador consiste em atuadores, sensores, elos e juntas que formam uma cadeia cinemática aberta, destinadas a sustentar e posicionar o órgão terminal que fica em contato direto com o processo. De um ponto de vista topológico, a cadeia cinemática é denominada aberta ou serial quando existe apenas uma sequência de elos que ligam as duas extremidades da cadeia. Alternativamente, um manipulador contém uma cadeia cinemática fechada quando uma sequência de ligações forma um laço [1].

3.1 Conceitos gerais

A mobilidade do manipulador é o resultado de uma série de movimentos elementares, independentes entre si, denominados graus de liberdade do robô. O número de juntas determinam o grau de liberdade e a geometria o espaço de trabalho do manipulador. O braço de um robô manipulador é capaz de se mover para várias posições porque possui uniões ou juntas, também

denominadas eixos, que permitem ao manipulador executar tarefas diversas. O movimento da junta de um robô pode ser linear ou rotacional permitindo movimentos lineares ou rotacionais entre dois elos, respectivamente [1, 20, 21, 22].

As juntas do manipulador podem ser divididas em duas classes: juntas do corpo e juntas do pulso. As juntas do corpo são os eixos da base do corpo do robô que permitem posicionar seu órgão terminal (efetuador) no espaço. Essas juntas são denominadas cintura, ombro e cotovelo definindo a classificação dos manipuladores baseada na geometria que será analisada na Seção 3.3.2.

As juntas do pulso são os eixos da cadeia cinemática entre o braço e o efetuador. Permitem orientar seu órgão terminal. As juntas do punho são em geral de revolução fazendo um punho esférico como observa-se na Figura 3.1. Os eixos z_4 , z_5 e z_6 permitem uma rotação das juntas em um ângulo de θ_4, θ_5 e θ_6, respectivamente. Eles são perpendiculares entre si e se interceptam em um ponto. Em geral encontram-se punhos com três, dois e um grau de liberdade, dependendo da aplicação. Observa-se que o punho esférico simplifica a cinemática do manipulador podendo desacoplar a posição da orientação do efetuador.

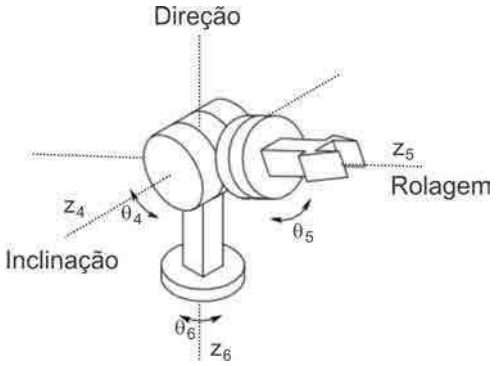


Figura 3.1: Representação simplificada da garra do manipulador, indicando a orientação dos ângulos das três juntas rotativas.

O efetuador é a ferramenta do manipulador que atua diretamente no processo realizando a tarefa. A ferramenta mais simples é a garra, a qual possui a função de abrir e fechar. Enquanto esta é adequada para a transferência de materiais entre diferentes pontos, não é adequada para outras tarefas como soldagem, colagem e montagem. Outros tipos de efetuador são as pistolas pulverizadoras feitas para pintura, furadeiras, polidoras e soldadoras [1].

O espaço de trabalho do manipulador é o volume total varrido pelo seu efetuador quando o manipulador executa todos os seus movimentos possíveis. Essa porção de espaço está limitada pelas restrições geométricas e mecânicas das juntas. Um robô com seis eixos, sendo três para o posicionamento e três para a orientação, é compatível com qualquer tarefa que seja realizada

dentro de seu volume de trabalho; com menos de seis graus de liberdade nem todos os pontos do ambiente de trabalho seriam alcançáveis [21]. Um robô com mais de seis eixos é denominado robô redundante, ou seja, tem mais graus de liberdade do que o mínimo requerido para a execução da tarefa [1, 20, 21].

3.2 Benefícios e aplicações

A robótica e o uso de manipuladores tem ganhado espaço nestes últimos anos pelas vantagens que representa o seu uso no ambiente industrial. Os principais fatores que determinaram esse avanço é que a tecnologia robótica pode ser aplicada em campos cada vez maiores, além disso, deve-se aos benefícios que representam automatizar uma indústria. Por seu significado habitual, o termo designa uma automação destinada a substituir os seres humanos por máquinas no processo de fabricação, no que diz respeito não só as operações físicas, mas também ao processamento inteligente de informação sobre o estado do processo. A automação é então a síntese de tecnologias industriais típicas do processo de fabricação que permite a gestão de informação [1, 20, 21]. Algumas dessas vantagens de usar manipuladores no processo de manufatura são:

- Aumento da produtividade, custo de produção menor, confiabilidade no processo, eficiência e qualidade final do produto;
- Os robôs podem trabalhar em ambientes perigosos, sem conforto ambiental e em tarefas desagradáveis para o ser humano;
- Apresentam uma alta repetitividade e precisão durante todo o ciclo de operação. Além disso, possuem versatilidade (disponibilidade de diferentes efetuadores) e adaptabilidade a diversas situações;
- Trabalham continuamente por longos períodos de tempo sem apresentar fadiga ou cansaço;
- Menor demanda de contratação de mão de obra especializada.

Como todo estrutura sofre desvantagens, podem-se citar as seguintes:

- Medidas de segurança são necessárias para garantir que não sejam causados danos a operadores e máquinas. Não tem a habilidade de responder a emergências, a menos que a situação seja prevista previamente e programada nele;

- São relativamente caros, devido ao custo inicial do equipamento, custo de instalação e treinamento de pessoal para operar eles.

Manipuladores apresentam três capacidades fundamentais que os tornam úteis para o processo de fabricação: movimentação, manufatura e medição [1]. Em um processo de fabricação, pode ser que os objetos tenham que ser transferidos de um local da fábrica para o outro, a fim de ser armazenados, montados ou embalados. Durante a transferência, e em condições normais, as características físicas do objeto não sofrem alteração nenhuma. A capacidade do manipulador para pegar o objeto, movê-lo no espaço em caminhos predefinidos e liberá-lo faz com que o próprio robô seja um candidato ideal para operações de movimentação de materiais. As aplicações típicas incluem:

- Processo de peletização que requer colocar objetos em uma paleta de trabalho de forma ordenada;
- Carga e descarrega de materiais no espaço de trabalho;
- Empacotamento de objetos;
- Classificação de peças.

Obter produtos manufaturados consiste em transformar a matéria prima em produtos acabados. Durante este processo, as suas características físicas são modificados, como resultado de usinagem, ou perdem a sua identidade como um resultado de uma montagem entre várias peças. A capacidade do robô para manipular objetos e peças torná-o adequado para ser empregado nesse tipo de tarefa. Aplicações típicas incluem:

- Soldagem de arco e pontual;
- Pintura e revestimento;
- Colagem e selagem;
- Corte laser, mecânico e hidráulico;
- Fresagem e furação;
- Fundição, pulverização e moagem;
- Parafusamento e fixação;
- Montagem de partes mecânicas e elétricas.

O processo de medição é necessário no processo de manufactura para realizar um controle de qualidade do produto e verificar se adere a especificação. O manipulador pode ser usado para medir essas qualidades. Aplicações típicas incluem:

- Inspeção de objetos;
- Detecção de imperfeições;
- Detecção de características da peça, como bordas, áreas e perímetros.

3.3 Classificação

Manipuladores podem ser classificados tendo em conta vários critérios, como número de eixos, tipo de movimento gerado pelo sistema de controle, tipo de acionamento, geometria e tipo de planejamento implementado [1, 2, 20, 21].

3.3.1 Tipos de acionamento

A classificação é feita em relacão aos atuadores que acionam as juntas do manipulador. Os atuadores são os mecanismos motrizes de uma máquina. Assim tem-se:

- Pneumáticos;
- Elétricos;
- Hidráulicos.

Os atuadores hidráulicos possuem alto torque e velocidade de resposta, sendo adequados para atuar sobre cargas pesadas. Entretanto requerem equipamentos periféricos, como bombas, o que implica na necessidade de manutenção frequênte, além de gerar grande ruído e não ser limpos para algumas aplicações.

Os atuadores pneumáticos são mais baratos e simples, entretanto, não podem ser controlados com precisão e são necessários equipamentos externos, como bombas e válvulas. Eles sofrem dos mesmos problemas de ruído, atrito e sujeira que os hidráulicos.

Atualmente os motores elétricos, como servomotores ou motores de passo, são os mais atrativos para ser empregados em robótica, devido a serem mais baratos e silenciosos. Um problema dos motores é que produzem torques baixos, requerendo adaptações como um sistema

de engrenagens de redução para produzir os torques necessários. Isto melhora a controlabilidade do sistema mas reduz a velocidade máxima aplicada e adiciona atrito.

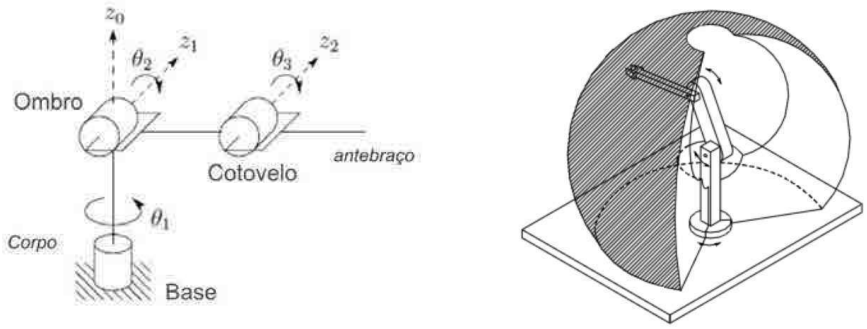
Finalmente, estão sendo aplicados outros métodos de acionamento, tais como piezoelétricos e as SMA. As últimas tem uma alta relação força/peso, operação silenciosa, adaptável na aplicação por ser flexíveis, permitem a redução de peso na plataforma. Sofrem desvantagens como todo atuador: precisam de uma alta energia para gerar o movimento, o ciclo de trabalho é lento e são difíceis de controlar [23, 24].

3.3.2 Geometria do manipulador

A geometria do braço está relacionada com os tipos de juntas que determinam a configuração no espaço de trabalho. A combinação dos dois tipos de juntas, lineares e rotacionais, vão definir os cinco tipos de geometria mais usuais. Dependendo do tipo de aplicação será mais apto um tipo de configuração ou outra. A classificação é feita tendo em conta as três primeiras juntas (juntas do corpo) [1, 20, 21].

3.3.2.1 Configuração articulada (RRR)

O manipulador articulado ou antropomórfico, é um robô com todas as suas juntas da base de revolução (RRR), sendo o mais hábil e flexível para a realização de uma determinada tarefa. O manipulador é chamado também robô de revolução. Essa configuração permite grande liberdade de movimento em um espaço compacto. É denominado de antropomórfico pois claramente se assemelha à anatomia do corpo humano. Como observa-se na Figura 3.2a, o primeiro eixo z_0 é a base ou cintura e representa o corpo da estrutura, a segunda junta é o ombro (eixo z_1) e a terceira junta (eixo z_2) conecta o braço com o antebraço e é chamada de cotovelo.



(a) Manipulador articulado. (b) Espaço de trabalho.

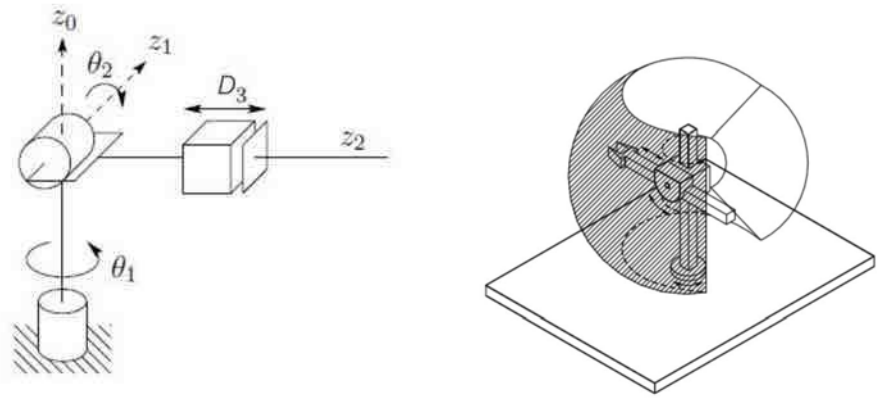
Figura 3.2: Representação da configuração antropomórfica, indicando a orientação das três juntas rotativas e o volume envolvido pelo manipulador.

Os eixos z_1 e z_2 são paralelos entre si e ambos perpendiculares a z_0. A primeira junta de revolução permite rotações de um ângulo θ_1 no eixo vertical z_0, a segunda e a terceira junta de revolução permitem rotações θ_2 e θ_3 nos eixos horizontais z_1 e z_2, respectivamente.

O exemplo mais conhecido é o manipulador PUMA que é um dos projetos mais populares de robôs articulados e foi projetado inicialmente para cumprir com os requisistos de montagem da indústria automobilística. O espaço de trabalho aproxima uma porção de esfera como observa-se na Figura 3.2b. O posicionamento do punho possui uma precisão variável em todo o espaço de trabalho e o acionamento das juntas é feito geralmente com motores elétricos.

3.3.2.2 Configuração esférica (RRP)

Substituindo a terceira junta (cotovelo) de revolução da configuração articulada por uma junta prismática (linear) obtêm-se a configuração esférica ou RRP. Como mostra-se na Figura 3.3a os eixos z_0, z_1 e z_2 são perpendiculares entre si e o ponto de interseção define um sistema de coordenadas esférico que permite determinar mais fácilmente a posição do efetuador. A primeira junta que é de revolução permite rotações de um ângulo θ_1 no eixo vertical z_0, a segunda permite rotações θ_2 no eixo z_1 e a terceira junta é prismática, permitindo movimentos lineares através do eixo z_2 uma distância D_3.



(a) Manipulador esférico. (b) Espaço de trabalho.

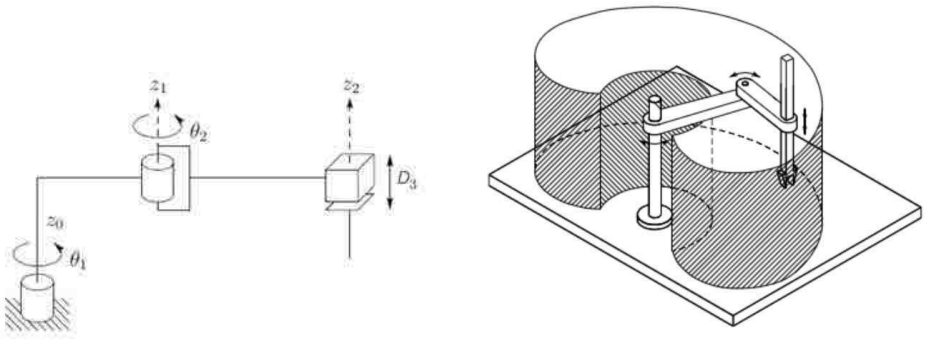
Figura 3.3: Representação da configuração esférica, indicando o volume envolvido pelo manipulador, as rotações das primeiras duas juntas e o deslocamento linear no terceiro eixo.

A rigidez mecânica é baixa comparando-se com a geometria cartesiana e cilíndrica, além de ser mais complexa a construção da plataforma. A precisão da posição do punho decresce radialmente, geralmente usa motores elétricos para acionar as suas juntas, são empregados principalmente para usinagem, tais como serramento, aplanamento, torneamento, fresamento, furação,

brochamento, eletroerosão, entre outros. O espaço de trabalho é uma esfera oca como observa-se na Figura 3.3b.

3.3.2.3 Configuração SCARA (RRP)

O manipulador SCARA é um braço robótico que, como seu nome indica, é adaptado com o propósito de fazer a montagem de peças. Embora tendo a mesma configuração que o robô esférico RRP, é diferente em aparência e nas aplicações onde é usado. Na configuração esférica os três eixos são perpendiculares, diferentemente da configuração SCARA que são paralelos entre si. Da geometria observada na Figura 3.4a, a primeira junta de revolução permite rotações em um ângulo de θ_1 no eixo verticalz_0, a segunda θ_2 no eixo vertical z_1 e a terceira junta prismática deslocamentos D_3 no eixo z_2. Esse tipo de manipulador oferece alta rigidez mecânica nos seus elos e é mais apto para montagens na horizontal. A precisão vertical da posição do pulso é constante e a horizontal decresce quando a distância do punho ao primeiro eixo aumenta (decresce radialmente). O espaço de trabalho é apresentado na Figura 3.4b.



(a) Manipulador SCARA. (b) Espaço de trabalho.

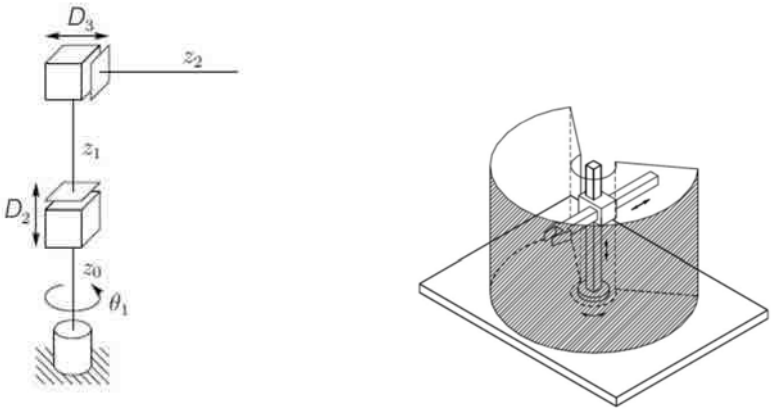
Figura 3.4: Representação da configuração SCARA, indicando o volume envolvido pelo manipulador, as rotações das primeiras duas juntas e o deslocamento linear no terceiro eixo.

3.3.2.4 Configuração cilíndrica (RPP)

Na geometria cilíndrica (Figura 3.5a) a primeira junta é de revolução e produz uma rotação θ_1 no eixo vertical z_0, a segunda e terceira junta são prismáticas provocando deslocamentos lineares D_2 e D_3 nos eixos z_1 e z_2, respectivamente. Os eixos z_0 e z_1 são colineares, sendo ambos perpendiculares com o eixo z_2. Pode-se definir um sistema de coordenadas cilíndrico com origem na sua base para definir a posição do efetuador, sugerindo o nome da configuração.

Possui boa rigidez mecânica e a precisão de posicionamento de pulso diminui com o aumento do curso horizontal com respeito à base. No entanto, a precisão vertical mantem-se

constante em toda a área de trabalho. O espaço de trabalho é um cilindro oco como se mostra na Figura 3.5b.



(a) Manipulador cilíndrico. (b) Espaço de trabalho.

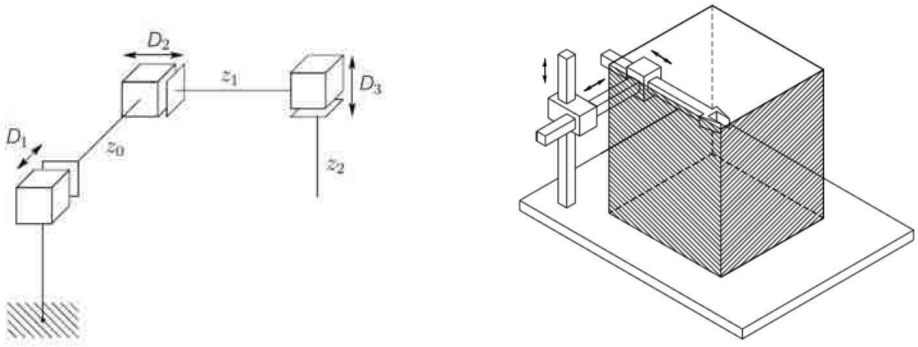
Figura 3.5: Representação da configuração cilíndrica, indicando o volume envolvido pelo manipulador, a rotação da primeira junta e os deslocamentos lineares nos outros dois eixos.

A articulação horizontal prismática faz que o pulso possa acessar as cavidades horizontais. Manipuladores cilíndricos são principalmente utilizados para o transporte de objetos de grandes dimensões, sendo assim o uso de atuadores hidráulicos mais adequados em vez de motores elétricos.

3.3.2.5 Configuração Cartesiana (PPP)

O manipulador que possui as três primeiras juntas prismáticas é conhecido como um braço cartesiano, mostrado na Figura 3.6a. Os três eixos são perpendiculares e o movimento linear é feito sobre esses eixos. A configuração cartesiana define um sistema coordenado cartesiano com origem na base para definir mais facilmente a posição do efetuador. O robô cartesiano tem aplicações como montagem em superfícies planas, trasporte de carga, etc.

A estrutura cartesiana oferece boa rigidez mecânica e a precisão de posicionamento do punho é constante em todo o espaço de trabalho. Possui baixa destreza porque todas as juntas são prismáticas. Isto proporciona um espaço de trabalho em forma de paralelepípedo (Figura 3.6b). Outra estrutura cartesiana pode ser do tipo pórtico, disponibilizando um espaço de trabalho amplo que permite o manuseio de objetos de grandes dimensões e peso. Os atuadores usados para este tipo de configuração são principalmente motores elétricos e atuadores pneumáticos.



(a) Manipulador Cartesiano. (b) Espaço de trabalho.

Figura 3.6: Representação da configuração cartesiana, indicando os deslocamentos sobre os tres eixos e o volume envolvido pelo manipulador.

3.3.3 Tipos de movimento

Outra classificação fundamental é baseada no tipo de movimento gerado pelo sistema de controle do manipulador. Dependendo da tarefa atribuída ao efetuador do braço, pode ser mais útil especificar a trajetória nas articulações ou diretamente no espaço cartesiano. Os dois tipos básicos de movimentos são ponto a ponto e trajetória contínua [20].

No movimento ponto a ponto é necessário atribuir apenas as configurações do efetuador e o manipulador para que este movimente-se nessa sequência discreta de pontos no espaço de trabalho. A trajetória entre os pontos não é definida, em geral, pelo usuário e sim pelo sistema de controle do manipulador. O movimento ponto a ponto é usado em tarefas discretas, por exemplo, para soldagem pontual e aplicações de carregamento e descarregamento de objetos. Para superficies de trabalhos planas, a sequência de movimentos típica para executar tarefas ponto a ponto é apresentada na Figura 3.7.

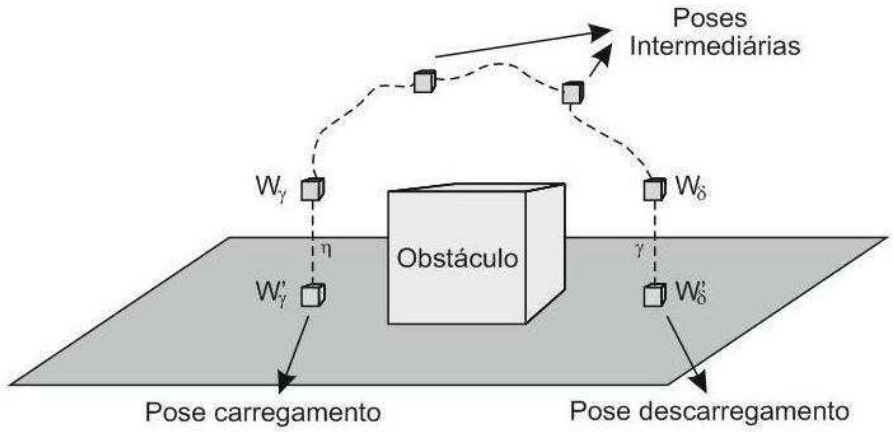


Figura 3.7: Trajetória ponto a ponto, indicando as poses de manipulador.

A pose W_γ (W_δ) é uma pose próxima à pose W'_γ (W'_δ) do objeto. Essas poses posuem a mesma orientação separadas pelas trajetórias η (γ) feitas a baixa velocidade e perpendiculares ao plano de trabalho, para obter maior precisão e são lineares no espaço cartesiano. A escolha de W_γ é feita tal que o efetuator fique posicionado acima do centro de massa σ do objeto. As poses intermediárias permitem evitar obstáculos onde as suas trajetórias podem ser feitas a maiores velocidades e são especificadas pelo controlador do manipulador.

No movimento de trajetória contínua é permitido um controle mais fino: o efetuador e as juntas devem seguir uma trajetória contínua e desejada, onde a velocidade ou a aceleração podem mudar em toda a trajetória de trabalho. O objetivo do planejamento de trajetória contínua é gerar as leis de tempo para as variáveis relevantes (juntas) a partir de uma descrição concisa do movimento desejado. Aplicações deste tipo de movimento podem ser para pintura usando pulverizadores, soldagem, colagem, etc.

3.3.4 Tipos de planejamento

Os dois tipos de planejamento mais conhecidos são probabilísticos e determinísticos. Inicialmente nesta seção, é definida a nomenclatura básica para entender ambos tipos de algoritmos. Em seguida são explicados os algoritmos probabilísticos PRM e SBL, pois tem muitas caracteristicas desejadas sendo mais usados nas industrias [2].

3.3.4.1 Definições e notações

Para caracterizar caminhos livres de obstáculos é necessário definir o conjunto de configurações livres de obstáculos, na seguinte definição formal:

Dada uma configuração do manipulador $q \in \mathbb{R}^n$, onde n é o número de graus de liberdade do robô, define-se $\mathbb{C}$ como o espaço de configurações do manipulador:

$$\mathbb{C} = \{[\theta_{mín}\,,\,\theta_{máx}]^n\}\,. \tag{3.1}$$

O vetor q contém as variaveis admissíveis das juntas do braço: Os ângulos entre os elos no caso de juntas rotacionais ou deslocamentos no caso de juntas prismáticas, sendo $\theta_{mín}$ e $\theta_{máx}$ os limiares mínimos e máximos permitidos de cada junta do manipulador, respectivamente.

Dados os obstáculos $\mathcal{O}_i$ ($i = 1, 2, ..., p$) representados no espaço de configuração, o conjunto

de obstáculos $\mathbb{C}\mathscr{O}$ é definido como:

$$\mathbb{C}\mathscr{O} = \left\{ q \in \mathbb{C} \,/\, q \cap \left[\bigcup_{i=1}^{p} \mathscr{O}_i \right] \neq \phi \right\}. \tag{3.2}$$

O complemento é o espaço livre de obstáculos $\mathbb{F} \subseteq \mathbb{C}$:

$$\mathbb{F} = \mathbb{C} - \mathbb{C}\mathscr{O} = \left\{ q \in \mathbb{C} \,/\, q \cap \left[\bigcup_{i=1}^{p} \mathscr{O}_i \right] = \phi \right\}. \tag{3.3}$$

Este subconjunto $\mathbb{F}$ das configurações q possíveis do robô não causam colisão com obstáculos. $\mathbb{C}$ é um espaço conectado desde que dando duas configurações arbitrárias existirá um caminho que as una. No entanto, pode ser que $\mathbb{F}$ não seja um espaço conectado devido aos obstáculos presentes na cena. Definidos esses conjuntos e um par de entrada $q_{inicial}$ e $q_{objetivo}$, planejar uma trajetória livre de obstáculos é gerar um caminho seguro entre $q_{inicial}$ e $q_{objetivo}$ se ambas pertencem a $\mathbb{F}$, caso contrário reportar uma falha.

Um subespaço usado nos planejadores de trajetória para definir uma nova configuração $q' \in \mathbb{C}$ na vizinhança de q de raio r é $B(q,r)$:

$$B(q,r) = \left\{ q' \in \mathbb{C} \,/\, d\left(q,q'\right) < r \right\}. \tag{3.4}$$

A distância métrica do algoritmo $d(q,q')$ é definida sobre $\mathbb{C}$.

A quantidade de entradas de um algoritmo define o tipo de planejamento de trajetórias, ele pode ser de um par de entrada ($q_{inicial}$ e $q_{objetivo}$) ou múltiplos pares. Um planejador de entradas múltiplas pré-computa um caminho em todo o espaço $\mathbb{F} \subseteq \mathbb{C}$ que depois pode ser utilizado por outros pares de entradas. Os planejadores de um único par devem fazer um novo cômputo para cada novo par de entrada do algoritmo. No entanto, o custo computacional é menor no caso de existir muita dinâmica ou mudanças na cena de trabalho.

3.3.4.2 Planejamento probabilístico

O planejamento probabilístico é um tipo de roteamento baseado em determinar conjuntos de configurações $q \in R^n$ livres de obstáculos em $\mathbb{F}$ e usar essas configurações para fazer o planejamento do movimento do braço. Isto é realizado, escolhendo em cada iteração uma configuração aleatória $q_{aleatoria}$ e verificar se essa colide com os obstáculos do espaço de trabalho. Em caso de existir a colisão, essa configuração é descartada. Caso contrário, é adicionada e conectada, se for possível, a outras configurações previamente armazenadas [2].

Algoritmo PRM O algoritmo PRM inicia gerando $q_{aleatoria}$ usando uma distribuição de probabilidade uniforme em $\mathbb{C}$. Logo se observa se $q_{aleatoria}$ colide com obstáculos usando a cinemática do manipulador para calcular a sua posição e orientação e chamando um algoritmo para detecção de colisões. Se $q_{aleatoria}$ não colide, é adicionada no caminho livre mais próximo e conectada, se for possível, com a configuração $q_{proxima}$ previamente calculada. Esse caminho livre próximo contém $q_{proxima}$ no extremo que é o ponto mais próximo a $q_{aleatoria}$. A geração de um caminho livre local entre $q_{aleatoria}$ e $q_{proxima}$ é delegada a um procedimento conhecido como planejador local. Esse caminho livre pode ser definido como uma reta em $\mathbb{C}$. Logo é observado se essa caminho sofre de colisões com obstáculos, discretizando o caminho com muita resolução. Se o caminho sofre de colisões, é descartado e nenhuma conexão direta é feita entre $q_{aleatoria}$ e $q_{proxima}$. A condição de parada do algoritmo é quando o número máximo de iterações é alcançado ou quando o número de caminhos conectados é menor que um valor predefinido. Nesse ponto, é verificado se é possível resolver o problema de movimento conectando $q_{inicial}$ com $q_{objetivo}$ ao mesmo caminho livre de colisões. Se a solução não é encontrada, o método pode ser executado com maior número de iterações ou reduzir o número de caminhos unindo eles com outro tipo de trajetórias (não lineares). A solução do algoritmo para o caso bidimensional observa-se na Figura 3.8 e o seu uso para soluções particulares na Figura 3.9.

A principais vantagens do algoritmo são a facilidade na implementação, eliminação do cálculo de $\mathbb{C}\mathscr{O}$ e o tempo para calcular a trajetória pode ser de várias ordens de magnitude menor que com outras técnicas não probabilísticas. A desvantagem é que a probabilidade de encontrar a trajetória é unitária quando o tempo de execução tende a ser infinito, significando que se não existe uma solução possível o algoritmo tem um tempo de execução infinito, devendo ser detido por uma condição de parada (em número máximo de iterações).

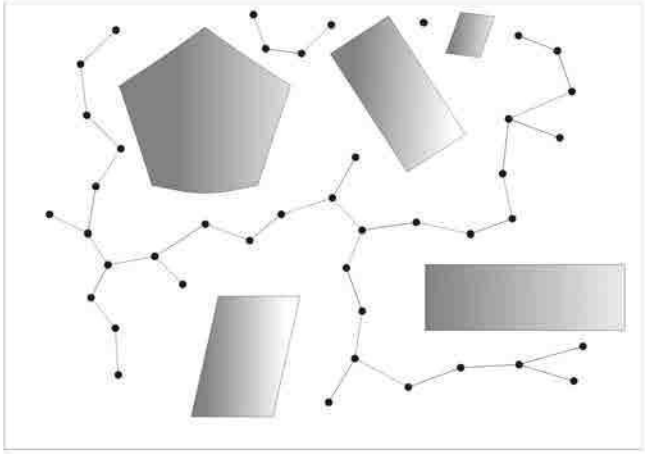


Figura 3.8: Roteamento no espaço de configuração para o caso dimensional usando o algoritmo PRM.

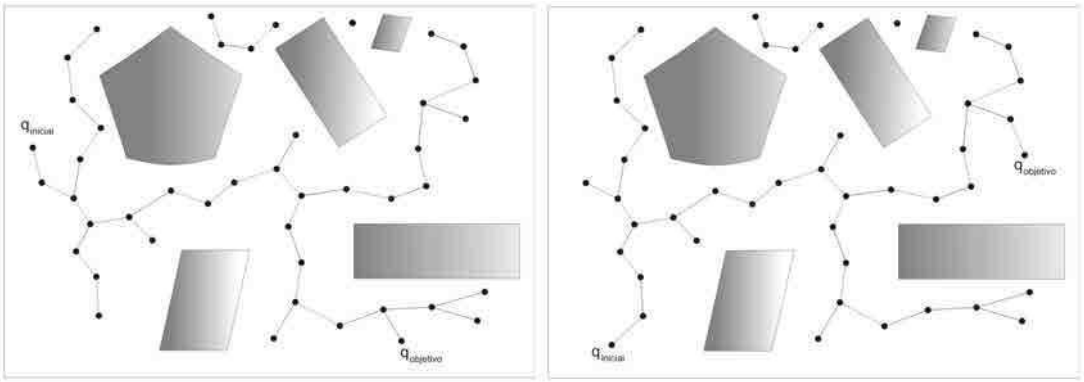


Figura 3.9: Soluções particulares para o roteamento PRM uma vez definido o espaço de configuração livre de obstáculos.

Algoritmo SBL O método SBL é um algoritmo PRM desenvolvido pela equipe de robótica da Universidade de Stanford [18]. O planejador de trajetórias tem a particularidade de usar duas entradas no algoritmo para explorar o menor espaço possível da configuração livre de colisões, diferenciando-lhe de um algoritmo PRM comum pois não realiza um pre-cômputo de uma rota na totalidade do espaço livre. É um algoritmo bidirecional pois faz uma exploração do espaço livre construindo duas árvores em cada entrada: tem-se uma árvore para $q_{inicial}$ e outro para $q_{objetivo}$ do manipulador, a idéia é unir as árvores e gerar o caminho no espaço de configuração.

Tem a particularidade de ser um algoritmo adaptativo pois modifica o passo de discretização ρ, aumentando em áreas abertas ou livres e reduzindo-se em áreas com muito obstáculos.

É um algoritmo "preguiçoso" na detecção das colisões, no sentido que os testes de colisão são pospostos até que seu cálculo seja totalmente necessário.

O algoritmo SBL é uma ferramenta importante e efetiva para a resolução de problemas de planejamento de trajetórias com muitos graus de liberdade. O planejador SBL toma amostras aleatórias do espaço de configuração e mantem aqueles que são livres de colisões. A conexão desses pontos são chamados de caminhos locais quando eles são livres de colisões (esses caminhos são segmentos lineares no espaço de configuração). A junção desses pontos e os caminhos livres de colisões formam o roteamento probabilístico desejado para o braço.

A motivação da utilização do algoritmo é que às vezes é impraticável computar uma representação explicita de um subespaço livre de colisões do espaço de configuração, então é mais eficiente testar se uma dada configuração é livre de colisões.

SBL incrementalmente constrói uma rede de configurações livres de colisões a partir de duas árvores, $T_{inicial}$ e $T_{objetivo}$, roteadas de $q_{inicial}$ e $q_{objetivo}$, respectivamente. Modifica dina-

micamente a toma de amostras dependendo da densidade de obstáculos. A idéia é expandir as árvores adicionando uma nova configuração livre de colisões q' (equivalente a $q_{aleatoria}$ no algoritmo PRM). Não é feito um teste de colisões de forma imediata, sendo feito quando são encontradas sequências de configurações livres de obstáculos entre $q_{inicial}$ e $q_{objetivo}$. O algoritmo possui três parâmetros de ajuste: o número máximo s de configurações livres de obstáculos permitidas no algoritmo, a distância ρ entre duas configurações próximas e a discretização ε do segmento para verificar colisões. O fluxograma apresenta-se na Figura 3.10.

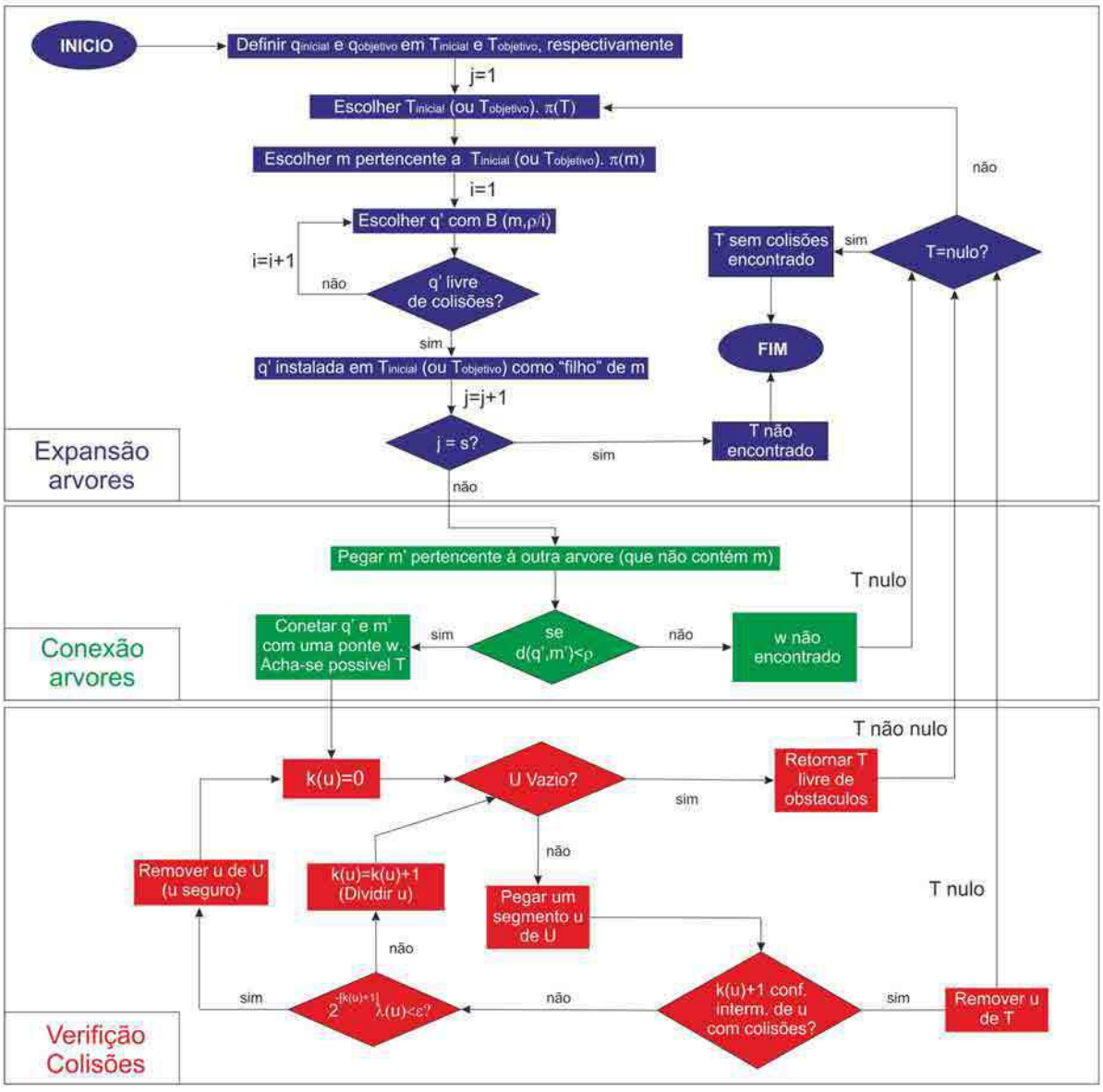


Figura 3.10: Fluxograma apresentando os blocos de expansão, conexão e verificação de colisões do algoritmo SBL.

O algoritmo SBL está definido por três blocos principais: Expansão, conexão das árvores e verificação das colisões.

O bloco expansão das árvores é encarregado de adicionar a cada árvore uma configuração do braço livre de obstáculos. Inicialmente são definidas as configurações $q_{inicial}$ e $q_{objetivo}$ livre de obstáculos como raízes das árvores $T_{inicial}$ e $T_{objetivo}$, respectivamente. Logo é escolhida uma delas para expandir com uma probabilidade $\pi(T) = 0,5$ e se procurará uma configuração livre de obstáculos m contida nessa árvore (equivalente a $q_{proxima}$ no algoritmo PRM). A seleção de m é feita com uma probabilidade $\pi(m)$ inversamente proporcional à quantidade de configurações livres na árvore escolhida ao redor de m. Finalmente, uma configuração livre q' é selecionada a uma distância menor ou igual que ρ de m. A seleção equiprovável entre as duas árvores permite que o crescimento das árvores seja similar.

Como observa-se, uma escolha adaptativa de q' é feita, começando em uma vizinhança $B(m,\rho) = \left\{q' \in \mathbb{C} \,/\, d(m,q') < \frac{\rho}{i}\right\}$, ao redor de m e logo sendo cada vez menor se existirem colisões de q' com obstáculos. A métrica definida sobre $\mathbb{C}$ é a distância linear $d = L_\infty$ onde q é normalizado entre $[0,1]$ representando $\mathbb{C}$ como $[0,1]^n$. Quando o candidato q' verifica que é livre de obstáculos, é mantido na árvore como uma nova configuração m. Os saltos de m a q' são maiores em áreas de $\mathbb{F}$ com menor densidade de obstáculos. Observa-se que a verificação de colisão do segmento entre m a q' é feita no final, quando é encontrado um caminho candidato entre $q_{inicial}$ e $q_{objetivo}$. O planejador retorna uma falha se não é encontrada uma solução depois de ser encontradas s configurações livres de obstáculos e não ter encontrado um caminho livre de colisão entre $q_{inicial}$ e $q_{objetivo}$.

Quando q' é adicionada na árvore, atua o bloco que tentará fazer a conexão das árvores. Denominando m' à configuração livre de obstáculos na outra árvore mais próxima a q', as árvores serão conectadas pelo segmento w, chamado de ponte, unindo q' e m' se a distância dessas duas configurações é menor que ρ. Se essa distância no espaço de configuração é maior então não é feita essa junção das árvores e novas configurações q' serão inseridas se a quantidade de configurações livres de obstáculos nas árvores é menor que o número máximo de iterações s. Quando w é achada, todos os segmentos (incluindo w) que formam o caminho entre $q_{inicial}$ e $q_{objetivo}$ serão avaliados para determinar se são livres de colisão.

O módulo de verificação das colisões determina se os segmentos que formam o caminho são livres de colisão. A cada segmento u da árvore é associado um índice $k(u)$ que indica a resolução que é testado u. Se $k(u) = 0$ somente os dois extremos do segmento foram verificados livres de colisão. Se $k(u) = 1$ os dois extremos e o ponto médio do extremo foram verificados. Em forma geral, para cada $k(u)$, $2^{k(u)} + 1$ pontos equidistantes de u foram verificados como livre de colisão. A distância de u no espaço de configuração é denominada como $\lambda(u)$.

O conjunto U denota todos os segmentos u na árvore que não foram marcados como segu-

ros. Quando um segmento é marcado seguro, ele é livre de colisões e descartado do conjunto U. Se U é vazio quer dizer que todos os segmentos u sao livres de colisão para essa resolução ε e o algoritmo achou uma árvore entre $q_{inicial}$ e $q_{objetivo}$ livre de obstáculos dando finalização ao algoritmo. Se U não está vazio escolhe-se um segmento u e continua dividindo-se em partes iguais analisando se as $k(u)+1$ configurações intermediárias não sofrem colisão.

Quando o algoritmo de verificação detecta uma colisão nessas subconfigurações $k(u)+1$ intermediárias do segmento u, remove esse segmento do caminho. Isto gera que o caminho fique desconectado novamente em duas árvores. Assim, o módulo de expansão das árvores tentará procurar outras configurações livres de obstáculos na tentativa de junção das árvores.

Quando o algoritmo de verificação satisfaz $2^{-(k(u)+1)}\lambda(u) < \varepsilon$ o segmento é marcado como seguro e é declarado livre de obstáculos e removido do conjunto U. O índice de cada novo segmento é inicializado em zero. Se o segmento não satisfaz a resolução ($2^{-(k(u)+1)}\lambda(u) > \varepsilon$) deverá ser dividido em partes menores (incrementando $k(u)$) até a resolução ser maior que cada divisão do segmento.

3.4 Manipulador Pegasus

No projeto decidiu-se utilizar o manipulador disponível no laboratorio adquirido com o propósito de desenvolver trabalhos de controle, automação e visão computacional. O braço é o Pegasus 880-RA2-1-B fornecido pelo fábricante Amatrol [25]. A sua aplicabilidade inclui montagem, manipulação de materiais, colagem e inspeção devido às seguintes caraterísticas:

Possui um punho esférico de dois graus de liberdade, carecendo da rotação θ_6 no eixo direção apresentada na Figura 3.1, e a sua geometria é articulada totalizando cinco graus de liberdade como é observado na Figura 3.11.

Cada eixo é controlado por um PID com um tempo de ciclo máximo de 100 *us* por eixo. Possui motores para acionar as juntas e o controlador do manipulador é multitarefa permitindo controlar até oito eixos[1]. Cada eixo permite obter a faixa de movimento no espaço das juntas apresentada na Tabela 3.1. Contrariamente o que foi informado pelo fabricante, obteve-se um ângulo de 660º para a junta rolagem, obtendo rotações redundantes.

A quantidade de motores implementados pelo fabricante são seis. A junta da base, ombro e cotovelo usam um motor cada. O movimento do eixo de inclinação usa dois motores, os mesmos dois que são usados para mover o eixo de rolagem. A diferença entre os movimentos

[1]Dois eixos para realizar o controle de dispositivos externos ao manipulador, por exemplo, esteiras.

é que no primeiro caso o movimento dos motores é conjunto e no outro funcionam em sentido inverso.

Figura 3.11: Manipulador Pegasus 880-RA2-1-B da Amatrol.

Junta	Faixa de Movimento
Cintura (θ_1)	345°
Ombro (θ_2)	220°
Cotovelo (θ_3)	270°
Inclinação (θ_4)	270°
Rolagem (θ_5)	360°

Tabela 3.1: Faixa de movimento de cada junta do manipulador.

O controlador realiza trajetórias ponto a ponto interpolando linearmente no espaço de juntas da estrutura. A ferramenta usada como efetuador é do tipo garra, que pode suspender cargas de até $1\,kg$ aplicando toda a sua força disponível (não tem sensor de força na garra). A estrutura mecânica possui várias engrenagens e correias necessárias para a transferência do movimento e aplicação de torque em cada eixo, pois alguns motores não atuam diretamente no eixo. A Figura 3.12 mostra os motores e engrenagens da estrutura.

Possui também sensores infravermelhos que permitem levar o manipulador na posição de referência, q_{home}, necessária em cada inicio, pois o manipulador perde a referência de ubicação quando é desligado. A velocidade máxima é de $599,4\,mm/s$ [25].

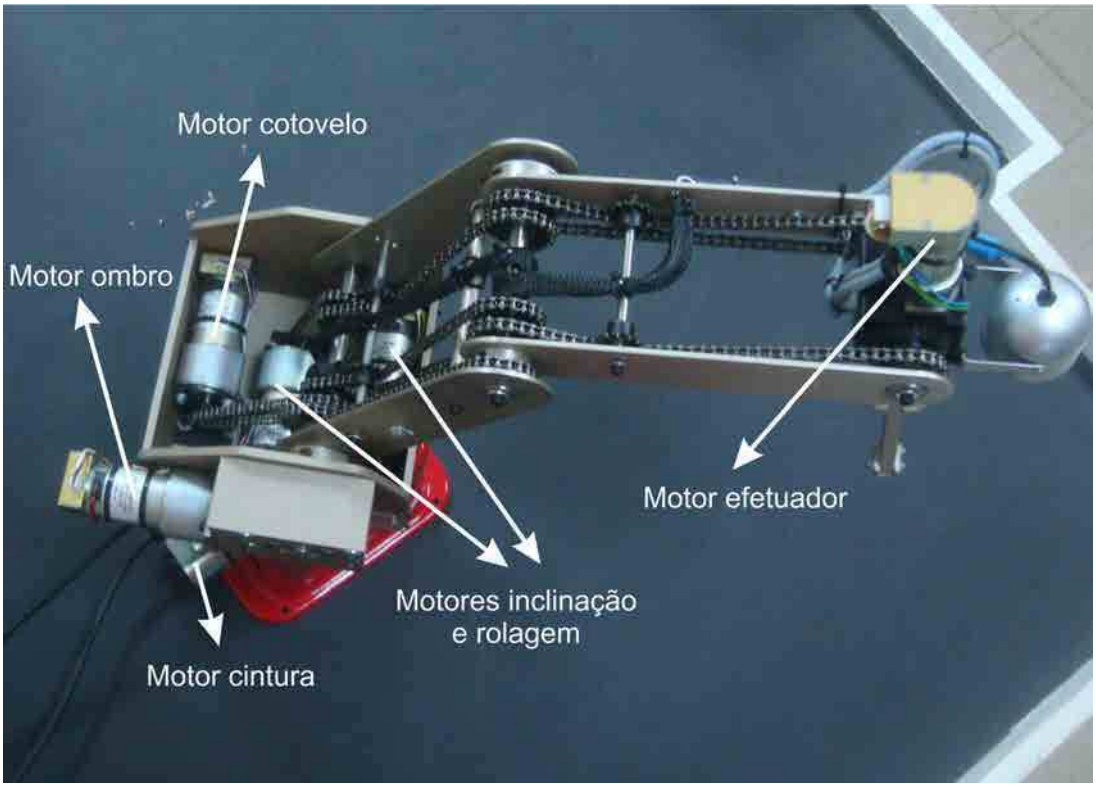


Figura 3.12: Motores e engrenagens do manipulador .

Possui uma interface I/O controlada por um PLC de 16 entradas e 16 saídas digitais permitindo a conexão de vários sensores e atuadores com alimentação de 24VDC facilitando assim uma maior automatização dos processos.

A programação do braço usa a linguagem MCL, padrão nos manipuladores, com algumas modificações feitas pela Amatrol para o Pegasus. Possui mais de 140 comandos para o movimento, comunicação com o PLC e a porta serial, etc.

Para facilitar o controle remoto da plataforma, assim como também a sua programação em linha, o Pegasus tem um controlador portátil denominado de "Teach Pendant". Esse terminal possui as mesmas funções que o software necessário para que o braço possa funcionar.

A comunicação do manipulador com o computador é USB emulando também uma porta serial.

3.4.1 Cinemática

A Cinemática é a ciência do movimento. Dentro desta ciência são estudados a posição, velocidade e aceleração de uma estrutura. A cinemática trata o movimento sem considerar as forças que o causam. Refere-se a todas as propriedades geométricas baseadas no tempo

considerando a posição e orientação das ligações dos manipuladores em situações estáticas, não tendo em conta a dinâmica da estrutura. Essas relações entre os movimentos, forças e torques são consideradas na dinâmica da estrutura.

Para compreender a geometria do manipulador, serão adicionados sistemas de coordenadas para as várias partes do mecanismo e, em seguida, são descritas as relações entre eles. O estudo da cinemática de manipuladores relaciona-se, entre outras coisas, com a mudança de esses sistemas na medida em que o mecanismo se movimenta. O foco desta seção é usar um método para calcular a posição e orientação do efetuador relativo à base do manipulador, como uma função das variáveis das articulações.

Conforme foi mostrado em seções anteriores, o robô manipulador pode ser modelado como uma cadeia de corpos rígidos denominada elos. Os elos são interconectados por articulações. Os extremos finais dessa cadeia são fixos na base e móveis no efetuador. O objetivo é controlar tanto a posição como a orientação do efetuador no espaço. A ferramenta, pode ser programada para seguir uma trajetória planejada para manipular objetos no espaço de trabalho. Para programar o movimento do efetuador, primeiramente deve-se formular a relação entre as variáveis de junta e a posição e orientação da ferramenta. Assim deve-se definir a cinemática direta e inversa da estrutura [1, 2, 20, 21, 22].

3.4.1.1 Cinemática direta

A cinemática direta permite determinar a posição e a orientação do efetuador conhecendo as variáveis de junta da estrutura. Essa pose é obtida em relação a algum sistema de coordenadas de interesse, como, por exemplo, a base da estrutura. As variáveis de juntas são os ângulos entre os elos no caso de juntas rotacionais ou deslocamentos no caso de juntas prismáticas. A cinemática descreve o movimento do manipulador sem considerar as forças ou torques que atuam na estrutura.

O objetivo é atribuir sistematicamente sistemas de coordenadas para cada elo. Uma vez feito isso, a equação geral do braço que representa a cinemática dos elos do manipulador é obtida. Na Figura 3.13 foram definidas as tramas em cada elo do manipulador utilizando a notação Denavit - Hartenberg (D-H). Os ângulos θ_k são ângulos de rotação entre o eixo x_k e o eixo x_{k-1} medidos sobre o eixo z_{k-1}. A distância d_k é o comprimento da articulação k, coincidente com o eixo z_{k-1} e medida desde a trama L_{k-1} à trama L_k. O comprimento a_k é a distância do elo k desde a trama L_{k-1} à trama L_k coincidente com o eixo x_k. Finalmente tem-se o ângulo α_k medido desde o eixo z_{k-1} até o eixo z_k sobre o eixo x_k. Os parâmetros dos elos (a_k e α_k) são sempre constante e específicados como parte do desenho mecânico do manipulador.

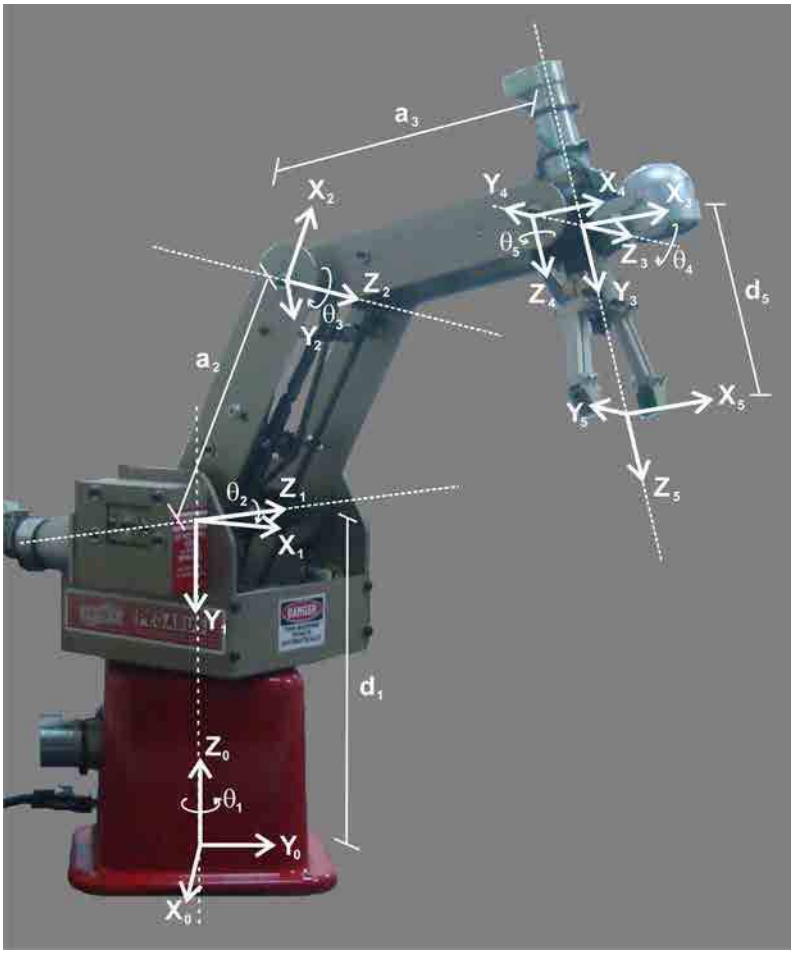


Figura 3.13: Definição da pose de cada trama do manipulador, indicando a rotação de cada junta e as dimensões de cada elo.

Observa-se que no punho coincidem as tramas L_3 é L_4 por ser ele um pulso de dois graus de liberdade. Na Tabela 3.2 tem-se os parâmetros da cinemática para o Pegasus.

Eixo	k	θ_k	$d_k(cm)$	$a_k(cm)$	α_k	q_{home}
Base	1	θ_1	31,25	0	$-\frac{\pi}{2}$	$\frac{\pi}{2}$
Ombro	2	θ_2	0	22.5	0	$-\frac{\pi}{2}$
Cotovelo	3	θ_3	0	22.5	0	$\frac{\pi}{2}$
Inclinação	4	θ_4	0	0	$-\frac{\pi}{2}$	$-\frac{\pi}{2}$
Rolagem	5	θ_5	15	0	0	0

Tabela 3.2: Parâmetros da cinemática do braço Pegasus.

Foi observado que a rotação dos ângulos $\theta_1, \theta_2, \theta_3$ e θ_4 no controlador do manipulador é feita em sentido contrário que a mostrada na Figura 3.13. Além disso, as referências estão rotacionadas modificando tanto a posição q_{home} como outras medidas. Por isto, são feitas mo-

dificações da referência dos ângulos: translações além de modificar o sentido de rotação, assim tem-se que para cada configuração i:

$$q(i,:) = \begin{bmatrix} -\theta_1 + 90 & -\theta_2 - 90 & -\theta_3 + 90 & -\theta_4 - 90 & \theta_5 \end{bmatrix}. \tag{3.5}$$

Assim obtém-se uma nova posição $q_{home} = \begin{bmatrix} 0 & 0 & 0 & -\frac{\pi}{2} & 0 \end{bmatrix}$.

A posição q_{home} é a posição inicial do manipulador, onde todos os sensores são inicializados. Definidas as tramas nos elos do manipulador, pode-se transformar os pontos de um sistema de coordenadas L_k a outro L_{k-1} usando a matriz de transformação de coordenadas homogêneas. Essa matriz estabelece as relações de coordenadas entre um sistema e o outro. Multiplicando as transformações entre as diferentes tramas obtêm-se uma matriz de transformação composta na qual são mapeadas as coordenadas do efetuador diz respeito a base da articulação [20]. Como as tramas foram definidas de acordo com o algoritmo D-H, a transformação do sistema de coordenadas k para o $k-1$ é definida através da Equação 3.6 usando os parâmetros da Tabela 3.2.

$$T_{k-1}^{k} = \begin{bmatrix} \cos(\theta_k) & -\cos(\alpha_k)\sin(\theta_k) & \sin(\alpha_k)\cos(\theta_k) & a_k\cos(\theta_k) \\ \sin(\theta_k) & \cos(\alpha_k)\cos(\theta_k) & -\sin(\alpha_k)\cos(\theta_k) & a_k\sin(\theta_k) \\ 0 & \sin(\alpha_k) & \cos(\alpha_k) & d_k \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{3.6}$$

onde $1 \leq k \leq n$ com $n = 5$. Assim, dado esse conjunto de tramas $\{L_0, L_1, ..., L_n\}$, $[q]^k$ e $[q]^{k-1}$ sendo as coordenadas homogêneas de um ponto respeito ao sistema k e $k-1$, tem-se que $[q]^{k-1} = T_{k-1}^{k}[q]^k$. Assim no Pegasus T_0^1 mapeia as coordenadas do ombro na base, T_1^2 mapeia as coordenadas do cotovelo no ombro, etc.

Usando a Equação 3.6 e algumas identidades trigonométricas obtêm-se que a transformação de coordenadas do efetuador à base $T_{base}^{efetuador}$ da Equação 3.8 conhecida também como a pose da ferramenta W. Isto é feito multiplicando as transformações entre as diferentes tramas obtendo uma matriz de transformação composta na qual são mapeadas as coordenadas do efetuador relativo à base da articulação (Equação 3.7):

$$T_{base}^{efetuador} = T_{base}^{ombro} T_{ombro}^{cotovelo} T_{cotovelo}^{inclinação} T_{inclinação}^{efetuador} = \left[\begin{array}{c|c} R_{3\times3} & P_{3\times1} \\ \hline 0 & 1 \end{array}\right], \tag{3.7}$$

$$T_{base}^{efetuador} = \begin{bmatrix} c_1c_{234}c_5 + s_1s_5 & -c_1c_{234}s_5 + s_1c_5 & -c_1s_{234} & c_1(a_2c_2 + a_3c_{23} - d_5s_{234}) \\ s_1c_{234}c_5 - c_1s_5 & -s_1c_{234}s_5 - c_1c_5 & -s_1s_{234} & s_1(a_2c_2 + a_3c_{23} - d_5s_{234}) \\ -s_{234}c_5 & s_{234}s_5 & -c_{234} & d_1 - a_2s_2 - a_3s_{23} - d_5c_{234} \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (3.8)$$

com $c_i = \cos(\theta_i)$, $s_i = \sin(\theta_i)$, $c_{ij} = \cos(\theta_i + \theta_j)$, $s_{ij} = \sin(\theta_i + \theta_j)$, $c_{ijk} = \cos(\theta_i + \theta_j + \theta_k)$ e $s_{ijk} = \sin(\theta_i + \theta_j + \theta_k)$. Da Equação 3.7 pode-se observar que a transformação permite obter a posição P e a orientação R relativa à base do manipulador. A cinemática direta então permite conhecer a configuração da ferramenta $T_{base}^{efetuador}$ dadas as variáveis das juntas q. A primeira é definida em um espaço $\mathbb{R}^6$ (a pose) e a segunda em um espaço $\mathbb{R}^n$, com n o número de juntas sendo $n = 5$ para o Pegasus.

Outra forma de representação da pose da ferramenta W é usar uma forma não redundante. Na Equação 3.8, a matriz R está escrita de uma forma redundante e pode ser escrita de uma forma mais sintética com a mesma informação. Observando a Figura 3.1, a orientação da ferramenta R pode ser especificada como um vetor unitário na direção do eixo z_5 e o ângulo θ_5 de rotação sobre esse eixo como observa-se na Equação 3.9. Esse vetor unitário é a terceira coluna da matriz R: r_3.

$$R_{3x1} = e^{(\theta_5/180)} r_3. \quad (3.9)$$

Usou-se a função exponencial que escala o vetor r_3 por uma quantidade positiva não mudando assim a direção do vetor [20]. Dessa maneira, codifica-se o ângulo θ_5 no vetor r_3 para produzir a mínima representação da orientação da ferramenta W. Então W pode-se escrever como apresentado na Equação 3.10 como um vetor de 6×1.

$$W(q) = \begin{bmatrix} \frac{P_{3\times1}}{R_{3\times1}} \end{bmatrix} = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \\ w_6 \end{bmatrix} = \begin{bmatrix} c_1(a_2c_2 + a_3c_{23} - d_5s_{234}) \\ s_1(a_2c_2 + a_3c_{23} - d_5s_{234}) \\ d_1 - a_2s_2 - a_3s_{23} - d_5c_{234} \\ -e^{(\theta_5/180)}c_1s_{234} \\ -e^{(\theta_5/180)}s_1s_{234} \\ -e^{(\theta_5/180)}c_{234} \end{bmatrix}. \quad (3.10)$$

Antes de alimentar a Equação 3.8 ou Equação 3.10 com os ângulos q das juntas, é necessário saber se os ângulos estão expressos na forma relativa ou absoluta. O controlador do manipulador trabalha com ângulos absolutos medidos em função do sistema de referência da junta. Se esse

for o caso, deve-se levar na forma relativa (ângulos medidos em função dos ângulos das outras juntas) para serem substituídos nas equações da cinemática. Levar os ângulos do controlador na forma relativa é fazer a seguinte atualização:

$$\theta_3 \leftarrow \theta_3 - \theta_2, \tag{3.11}$$

$$\theta_4 \leftarrow \theta_4 - \theta_3 - \theta_2. \tag{3.12}$$

3.4.1.2 Cinemática inversa

A cinemática direta permite determinar a posição e a orientação da ferramenta do manipulador respeito à base dados os ângulos de junta. A cinemática inversa faz o oposto determinando os ângulos das juntas conhecidas a posição e orientação do efetuador. Em geral as tarefas dos manipuladores estão definidas em termos de poses desejadas do efetuador, como neste projeto. Quando sensores externos ao manipulador, como câmeras, são usadas para planejar o movimento, a informação fornecida é a pose do objeto que, mediante a cinemática inversa, é transformada em ângulo das juntas [20].

A solução da cinemática inversa em geral é mais complexa que a direta, pois não existe um procedimento análogo ao algoritmo D-H. Como resultado, cada robô deve ser tratado de forma separada. No entanto, é mais usada pois permite controlar o movimento das juntas com a informação da pose obtida de sensores externos [2]. A solução da cinemática inversa é útil incluindo-se quando não se dispõe de sensores externos, pois é possível especificar a tarefa descrevendo a pose desejada do efetuador, no espaço cartesiano, que é mais intuitivo do que fornecer os valores dos ângulos das juntas.

Propriedades gerais da solução As soluções para o problema de cinemática inversa variam de acordo com os graus de liberdade do manipulador. Examinando as condições sobre as quais existe uma solução, pode-se observar, primeiramente, que se a posição desejada da ferramenta P está fora do espaço de trabalho, não pode existir solução do problema. Quando P está dentro do espaço de trabalho, podem existir certas orientações R onde são violados os limites das juntas. Isso acontece quando o punho possui menos de três graus de liberdade para a orientação do efetuador, então existem orientações não realizáveis.

Da Equação 3.7 podem-se obter 12 equações algébricas não lineares de R e P com n incógnitas (no caso do Pegasus $n = 5$). No entanto, essas 12 equações não são independentes uma das

outras, observa-se que R é a matriz de transformação que representa uma rotação pura de um sistema ortonormal para o outro formando também um espaço ortonormal. Essa ortogonalidade gera 3 equações linearmente independentes entre as colunas r_1, r_2 e r_3 da matriz R. As três equações são $r_1.r_2 = 0$, $r_1.r_3 = 0$ e $r_2.r_3 = 0$. Também são criadas mais três equações, pois a norma dos vetores coluna de R é unitária ($\|r_1 = 1\|$, $\|r_2 = 1\|$ e $\|r_3 = 1\|$). Então, na realidade, dessas 12 equações dependentes apenas 6 são linearmente independentes para as n incógnitas das variáveis de junta q. Portanto, uma condição necessária mas não suficiente para a solução do problema da cinemática inversa é que o manipulador tenha $n \geqslant 6$ juntas quando configurações arbitrárias são especificadas. Assim, a posição e orientação do efetuador deve estar no espaço de trabalho e os limites das variáveis de junta não serem violadas, respectivamente.

Quando no problema existem soluções, podem ser única ou múltiplas. No caso do robôs com $n > 6$ tipicamente existem infinitas soluções no problema de cinemática inversa. Nesse caso os manipuladores são ditos redundantes, pois tem mais graus de liberdade que os necessários para estabelecer poses arbitrárias do efetuador. Esses graus de liberdade adicionam flexibilidade à estrutura. Nos manipuladores não redundantes essa flexibilidade encontra-se restrita. Um manipulador redundante poderia ser comandado para atingir um objeto inacessível por ser encontrados obstáculos na trajetória. Aqui os graus de liberdade redundantes ajudam a evitar obstáculos, enquanto os outros permitem configurar a pose da ferramenta. Incluindo-se quando os manipuladores não são redundantes, existem casos nos quais obtêm-se varias soluções.

No exemplo do Pegasus encontram-se duas soluções para a mesma pose do efetuador, denominadas cotovelo-acima e cotovelo-abaixo (Equação 3.14). No espaço da configuração da ferramenta $W \in \mathbb{R}^6$ ambas soluções são idênticas, pois produzem a mesma orientação R e translação P. No entanto, no espaço das variáveis de junta $q \in \mathbb{R}^5$ as soluções são diferentes. Em geral é preferível a solução cotovelo-acima pois reduz a chance de colisão dos elos com obstáculos e objetos na superfície de trabalho.

Para desenvolver a solução da cinemática inversa, a configuração da ferramenta W da Equação 3.10 deve ser especificada como a informação de entrada. A solução no Pegasus pode ser encontrada de forma fechada, aproveitando a cinemática direta e que o seu punho é quase esférico, simplificando os cálculos [20]. Para extrair as variáveis de junta q da configuração $W(q)$ aplicam-se operações nas filas e usam-se identidades trigonométricas para isolar-lhes. Assim obtêm-se os resultados das Equações 3.13 a 3.17 sabendo que $\theta_{234} = \theta_2 + \theta_3 + \theta_4$.

$$\theta_1 = \tan 2 \left(\frac{w_2}{w_1} \right), \tag{3.13}$$

$\theta_{234} = \tan 2\left(\frac{-c_1 w_4 + s_1 w_5}{-w_6}\right)$,

$b_1 = c_1 w_1 + s_1 w_2 + d_5 s_{234}$,

$b_2 = d_1 - d_5 c_{234} - w_3$,

$$\theta_3 = \pm \arccos\left(\frac{\|b\|^2 - a_2^2 - a_3^2}{2a_2 a_3}\right), \tag{3.14}$$

$$\theta_2 = \tan 2\left(\frac{(a_2 + a_3 c_3)\, b_2 - a_3 s_3 b_1}{(a_2 + a_3 c_3)\, b_1 + a_3 s_3 b_2}\right), \tag{3.15}$$

$$\theta_4 = \theta_{234} - \theta_2 - \theta_3, \tag{3.16}$$

$$\theta_5 = \pi \ln\left(\sqrt{w_4^2 + w_5^2 + w_6^2}\right). \tag{3.17}$$

O controlador do manipulador Pegasus usa uma forma alternativa à Equação 3.8 e Equação 3.10 para representar a configuração da ferramenta W. A forma é denominada vetor de configuração da ferramenta reduzida $\widetilde{W} \in \mathbb{R}^5$, similar à Equação 3.10 mais as últimas duas componentes estão expressas em função dos ângulos de rolagem e inclinação das juntas. $\widetilde{W}$ é definido na Equação 3.18:

$$\widetilde{W} = \begin{bmatrix} \widetilde{w_1} \\ \widetilde{w_2} \\ \widetilde{w_3} \\ \widetilde{w_4} \\ \widetilde{w_5} \end{bmatrix} = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ q_p \\ q_r \end{bmatrix} = \begin{bmatrix} c_1(a_2 c_2 + a_3 c_{23} - d_5 s_{234}) \\ s_1(a_2 c_2 + a_3 c_{23} - d_5 s_{234}) \\ d_1 - a_2 s_2 - a_3 s_{23} - d_5 c_{234} \\ \theta_4 + \theta_3 + \theta_2 \\ \theta_5 \end{bmatrix}, \tag{3.18}$$

onde $\widetilde{w_4}$ é definido o ângulo inclinação do manipulador e o termo $\widetilde{w_5}$ é definido como o ângulo de rolagem do punho.

A saída das equações da cinemática inversa são ângulos relativos em função das juntas anteriores. Para obter os ângulos absolutos do manipulador é necessário realizar o procedimento inverso das Equações 3.21 e 3.12. Ou seja:

$$\theta_4 \leftarrow \theta_4 + \theta_3 + \theta_2, \tag{3.19}$$

$$\theta_3 \leftarrow \theta_3 + \theta_2. \tag{3.20}$$

3.5 Modelagem do manipulador

Para gerar o planejamento do manipulador Pegasus deve-se criar o modelo do manipulador e especificar a sua cinemática, definir os limiares das juntas e especificar as dimensões dos elos. A cinemática da estrutura foi definida na Seção 3.4.1. Os comprimentos dos elos e eixos foram especificados na Tabela 3.2 e a faixa de movimento da estrutura definida na Tabela 3.1. Diversos testes mostram que os limiares das juntas tem uma variação de teste em teste e o fabricante da plataforma não especifica claramente o final da trajetória de cada junta. Por isso, por questões de segurança a rotação dos cinco eixos do manipulador foram limitados aos seguintes valores como apresenta-se na Tabela 3.3.

Junta	Faixa de movimento ($\theta_{mín}$ a $\theta_{máx}$)
Cintura (θ_1)	-160° a 160°
Ombro (θ_2)	-96° a 71°
Cotovelo (θ_3)	-35° a 200°
Inclinação (θ_4)	-220° a 40°
Rolagem (θ_5)	-180° a 180°

Tabela 3.3: Limiares seguros dos cinco eixos do manipulador.

A estratégia é baseada na idéia de limitar as juntas para não perder muita faixa de atuação, o suficente para o sistema ser seguro e o efetuador nao sair do espaço de trabalho. A junta de rolagem foi limitada a uma rotação completa, eliminando redundância de movimentos.

Na Figura 3.14 representou-se cada peça do manipulador no aplicativo AC3D [26] de desenho CAD para desenvolvimento de modelos.

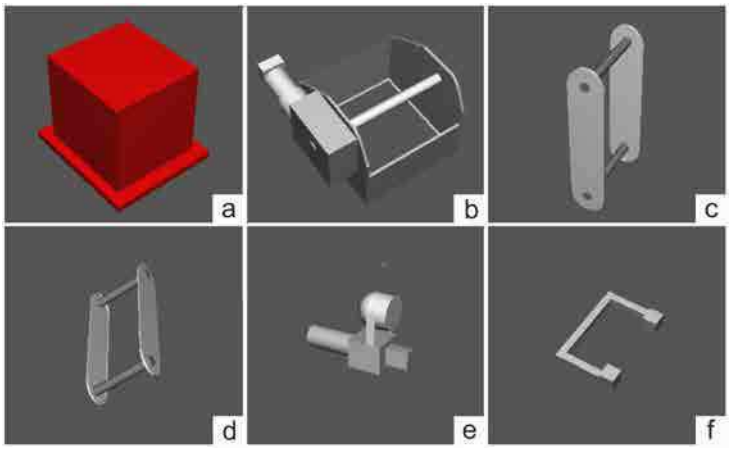


Figura 3.14: Representação tridimensional em modelos CAD de cada peça do manipulador.

As peças do manipulador foram definidas da seguinte forma: Base (Figura 3.14.a), ombro (Figura 3.14.b), úmero (Figura 3.14.c), antebraço (Figura 3.14.d), inclinação (Figura 3.14.e) e rolagem (Figura 3.14.f).

Como os desenhos das peças são aproximados, realizou-se um sobredimensionamento delas para evitar supostas colisões que aconteceriam se modelos subdimensionados fossem feitos. Os modelos possuem sobredimensionamento não maiores a $1\,cm$ de comprimento da cara se as peças são envolvidas em sua totalidade em paralelepípedos. O modelo final simplificado do manipulador observa-se na Figura 3.15.

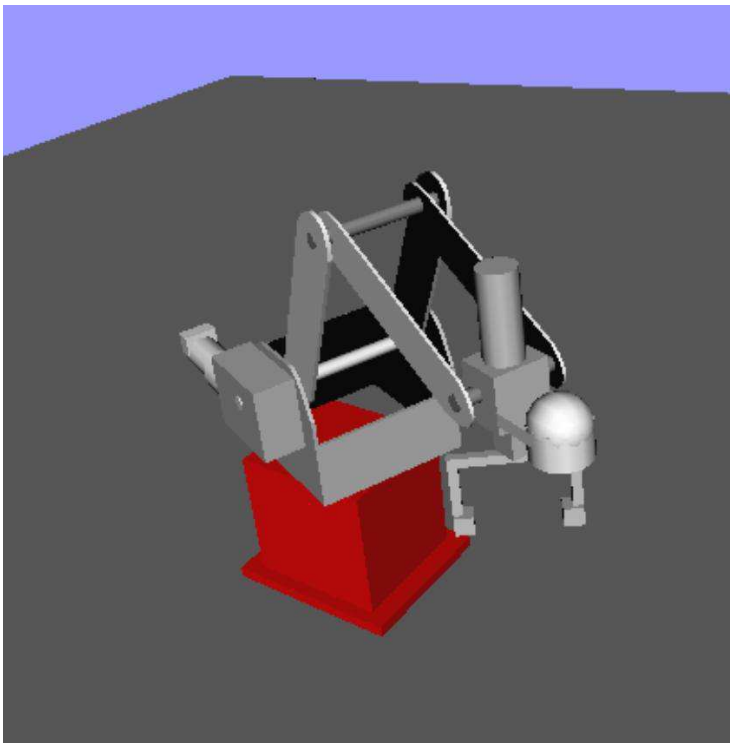


Figura 3.15: Representação tridimensional do manipulador através da junção de cada peça usando a cinemática da estrutura.

3.6 Normalização e valores de encoders

É importante definir uma nova forma de expressar os ângulos das juntas do manipulador. A forma normalizada permite expressar os angulos de cada junta no intervalo $[0,1]$. Assim, define-se o mapeamento linear $[\theta_{mín}, \theta_{máx}] \leftrightarrow [0,1]$. Ângulos relativos na forma normalizada são inseridos nos planejadores de trajetória para o cálculo do caminho livre de obstáculos. Tendo $q \in \mathbb{R}^5$ com $q = \begin{bmatrix} \theta_1 & \theta_2 & \theta_3 & \theta_4 & \theta_5 \end{bmatrix}$ na forma absoluta (veja-se Equação 3.19 e Equação 3.20) então os ângulos normalizados $q_n = \begin{bmatrix} \theta_{n1} & \theta_{n2} & \theta_{n3} & \theta_{n4} & \theta_{n5} \end{bmatrix}$ serão:

$$\theta_{n1} = \frac{1}{2} + \frac{1}{320}\theta_1, \quad -160^o < \theta_1 < 160^o. \tag{3.21}$$

$$\theta_{n2} = \frac{96}{167} + \frac{1}{167}\theta_2, \quad -96^o < \theta_2 < 71^o. \tag{3.22}$$

$$\theta_{n3} = \frac{7}{47} + \frac{1}{235}\theta_3, \quad -35^o < \theta_3 < 200^o. \tag{3.23}$$

$$\theta_{n4} = \frac{11}{13} + \frac{1}{260}\theta_4, \quad -220^o < \theta_4 < 40^o. \tag{3.24}$$

$$\theta_{n5} = \frac{1}{2} + \frac{1}{360}\theta_5, \quad -180^o < \theta_5 < 180^o. \tag{3.25}$$

O manipulador Pegasus possui a opção de enviar diversos comandos e parâmetros via porta serial. Não foi encontrada uma opção para enviar poses de efetuador por esse meio em tempo real. Uma forma alternativa de enviar poses do manipulador, é escrever na etapa prévia à execução dos movimentos um arquivo o qual é lido pelo controlador e executado. Esse arquivo possui os valores q_e dos encoders das cinco juntas para cada posição. Como o controlador do manipulador recebe únicamente valores de encoders q_e para realizar seus movimentos, realizou-se um mapeamento de ângulos para valores de encoders em cada junta. Os ângulos q foram determinados usando a cinemática inversa do manipulador, pois o controlador fornece únicamente a pose do efetuador na forma $\tilde{W}$.

Utilizou-se o método de mínimos quadrados para encontrar q_e. Os dados foram ajustados com um polinômio de primeira ordem (uma reta) pois os motores mostraram características lineares. A forma de obtenção dos dados foi realizando rotações de aproximadamente 10º em cada junta e cobrindo toda sua faixa de valores permitidos, anotando para cada ângulo seu valor de encoder. Observou-se que as retas são um bom ajuste dos dados para as cinco juntas do manipulador. As retas de cada junta que determinam $q_e = \begin{bmatrix} \theta_{e1} & \theta_{e2} & \theta_{e3} & \theta_{e4} & \theta_{e5} \end{bmatrix}$ são as seguintes:

$$\theta_{e1} = \frac{8051200}{9}\theta_{n1} - \frac{8051200}{18}, \tag{3.26}$$

$$\theta_{e2} = \frac{4201720}{9}\theta_{n1} - \frac{805120}{3}, \tag{3.27}$$

$$\theta_{e3} = \frac{1809500}{3}\theta_{n1} - \frac{269500}{3}. \tag{3.28}$$

Observou-se que os motores do eixo rolagem e inclinação estão acoplados, isto acontece pois o torque das últimas juntas não está aplicado diretamente nos eixos, e sim através de cadeias e engrenagens. Para encontrar o acoplamnento, foi observado o funcionamento do controlador do manipulador, determinando-se que rotações positivas ou negativas no ângulo θ_5 (θ_{n5}) acontecem quando ambos θ_{e4} e θ_{e5} aumentam ou diminuem, respectivamente. Aumentos no ângulo da junta θ_4 (θ_{n4}) são gerados por aumentos no valor de encoder θ_{e4} e redução de θ_{e5} em igual proporção. Redução de θ_4 (θ_{n4}) são gerados por redução no encoder θ_{e4} e aumento de θ_{e5} também em igual proporção. Então o acoplamento encontrado é observado nas Equações 3.29 e 3.30:

$$\theta_{e4} = (295760\,\theta_{n5} - 147880) + \left(\frac{1922440}{9}\theta_{n4} - \frac{961220}{9}\right), \tag{3.29}$$

$$\theta_{e5} = (295760\,\theta_{n5} - 147880) - \left(\frac{1922440}{9}\theta_{n4} - \frac{961220}{9}\right). \tag{3.30}$$

Quando os motores desses últimos dois eixos rotacionam na mesma direção estão afetando somente o eixo de inclinação e quando rotam em sentidos opostos afetam somente o eixo de rolagem.

3.7 Considerações finais

Neste capítulo apresentou-se uma breve explicação dos conceitos gerais de manipuladores. No embasamento teórico foi feita uma classificação tendo em conta os tipos de acionamento, geometria, tipo de movimento, tipo de planejamento utilizado e se mostraram os benefícios e aplicações de seu uso no ambiente industrial. No capítulo apresentou-se o planejador de trajetória SBL implementado no projeto, ele apresenta rápida convergência pois os testes de colisão dos segmentos são adiados até que seu cálculo seja totalmente necessário. Também é eficiente pois é um algoritmo adaptativo. Foi apresentado o manipulador Pegasus e desenvolvida as equações da cinemática que permitem relacionar os ângulos das juntas com a pose do efetuador no espaço cartesiano. Logo foi feita a transformação dos ângulos em valores normalizados e a modelagem do manipulador para ser representado no algoritmo de roteamento. Finalmente, são apresentadas as equações de transformação dos ângulos a valores de encoders necessários para o controlador realizar os movimentos.

Capítulo 4

Processamento de Imagens

A informação visual, ao contrário das informações fornecidas por outros tipos de sensores, é muito rica e variada e, portanto, requer de uma serie de complexas transformações antes que ele possa ser utilizada para controlar um sistema robótico [2]. O objetivo dessas transformações é a extração de informação numérica a partir da imagem $Im(x,y)$, o que fornece uma descrição sintética e robusta dos objetos de interesse na cena, através dos chamados parâmetros das características da imagem, como são valores de áreas, bordas e perímetros.

Para a obtenção desses parâmetros, primeiramente deve-se fazer uma segmentação da imagem, que permite obter uma representação adequada da imagem para uma identificação mais simples de características mensuráveis. A operação subsequente, denominada interpretação preocupa-se com a medição dos parâmetros de recursos da imagem [2, 27].

Na presente dissertação foram usadas técnicas de suavização das imagens através de filtros, os quais permitem a homogeneização da luminosidade e saturação da imagem obtendo-se uma extração mais eficiente das características desejadas, como áreas e bordas. além disso, na extração anteriormente mencionada, utilizou-se segmentação baseada em regiões, usando um limiar adequado para transformar a imagem em escala de cinzas e assim extrair os objetos desejados através das áreas ou bordas encontrados. Essas técnicas são implementadas na câmera instalada no efetuador do manipulador e serão explicadas brevemente nas próximas seções.

4.1 Filtros de suavização

A suavização é uma operação de processamento simples e frequentemente usada em processamento de imagens. Há muitas razões para suavizar, mas geralmente é feito para reduzir

o ruído ou a resolução de uma imagem. Existem diferentes filtros para suavizar uma imagem, como por exemplo, o filtro de média e mediana [28, 27]. Uma breve explicação dos filtros é apresentada nas próximas seções.

4.1.1 Filtro de média

A operação mais simples de suavização é quando cada pixel da imagem é substituído pelo valor meio de todos os pixels vizinhos contidos em uma janela determinada. A janela é uma máscara, geralmente quadrada, que permite definir quais pixels são afetados pelo operador. É um filtro computacionalmente rápido. O pixel de saída tem a seguinte forma:

$$g(i,l) = \sum_{k,l} f(i+k, j+l)h(k,l), \tag{4.1}$$

onde $k = 1, 2, .., K$ e $l = 1, 2, .., L$ sendo K e L o tamanho da janela $h(k,l)$. A saída do pixel é a soma ponderada dos pixels da máscara que possui a forma geral:

$$h(k,l) = \frac{1}{KL}\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1L} \\ a_{21} & a_{22} & \cdots & a_{2L} \\ \vdots & \vdots & \ddots & \vdots \\ a_{K1} & a_{K2} & \cdots & a_{KL} \end{bmatrix}, \tag{4.2}$$

Para um filtro de média com ponderação igualitária todos os coeficientes da máscara são iguais a 1. Observa-se que a aplicação de suavização da Equação 4.1 permite homogeneizar superfícies com a respectiva redução de qualidade, quanto maior é a máscara aplicada da Equação 4.2 maior será a suavização [2].

4.1.2 Filtro de mediana

O filtro de mediana substitui o pixel central da janela pelo valor da mediana. Assim, escolhendo uma janela de $K \times L$ e ordenando os pixels de menor a maior na janela o pixel central dela será substituído pelo valor do pixel de posição $(N+1)/2$ com $N = K \times L$. O filtro de média pode ser sensível ao ruído na imagem, especialmente ao ruído conhecido como "sal e pimenta" que é uma espécie de ruído onde pixels isolados têm valores atípicos em relacão aos vizinhos, com muita luminosidade (sal) ou pouca (pimenta). Esses pixels ruidosos podem gerar um deslocamento grande no valor médio. No entanto, o filtro de mediana pode ignorar esses valores discrepantes quando é feita a escolha da mediana na janela pré-determinada. Da mesma forma

no caso do filtro de média, quanto maior é a janela maior suavização será obtida [2].

4.2 Segmentação da imagem

Segmentar uma imagem é um processo de agrupamento, através do qual a imagem é dividida em um certo número de grupos, referidos como os segmentos, de modo que os componentes de cada grupo são semelhantes no que diz respeito a uma ou mais características [2, 27].

Tem-se duas abordagens para essa classificação, uma é feita baseada na pesquisa por regiões conectadas na imagem e a outra procurando as fronteiras dos objetos na imagem [2, 27]. O objetivo da segmentação baseada em regiões é de agrupar conjuntos de pixels que compartilham características comuns, com a suposição que as regiões resultantes correspondem a superfícies reais de objetos na cena. Por outro lado, a segmentação baseada em fronteiras identifica os pixels correspondentes aos contornos dos objetos isolando-os do resto da imagem. A fronteira ou a área de um objeto, uma vez extraída, pode ser usada para definir a posição, orientação e forma do objeto. Observa-se que esses dois enfoques são complementares, desde que as fronteiras de um determinado objeto podem ser obtidas isolando os contornos da região e, inversamente, uma região pode ser determinada considerando o conjunto de pixels contidos dentro da fronteira.

Tem-se várias formas de realizar segmentação. Por exemplo, o enfoque de limiarização é o mais usado para a segmentação baseada em regiões pela simplicidade na detecção de áreas. No entanto, usa-se o cálculo do gradiente para definir as fronteiras de objetos. Geralmente antes de aplicar segmentação, a imagem é convertida a escalas de cinza. Na sua representação a ϑ bits, permite representar cada pixel com valores de luminosidade no intervalo $[0, 2^{\vartheta}]$.

4.2.1 Segmentação baseada em regiões: Limiarização

A limiarização permite transformar uma imagem a sua forma binária. É usada, em geral, em imagens em escala de cinzas ou em imagens multibanda aplicando limiarização em cada canal. Na limiarização escolhe-se um limiar de disparo τ apropriado no intervalo $[0, 2^{\vartheta}]$ e os pixels da imagem são comparados com esse limiar. Para objetos claros com fundo obscuro, aqueles pixels que estejam acima daquele limiar serão considerados objeto, sendo os outros fundo da cena. Inversamente pode ser feita uma comparação com objetos obscuros e fundo claro.

Um fator importante é a escolha desse limiar, feito, a maioria das vezes, em base ao histograma de escala de cinzas da imagem. Esse histograma é uma representação gráfica dos pixels agrupados em função da sua luminosidade, e em cada conjunto é atribuída a frequência com

que os os valores de luminosidade estão presentes. Na prática, o histograma na escala de cinzas é ruidoso e não há uma separação clara entre os níveis não permitindo determinar claramente os objetos do fundo. Para este fim, várias técnicas têm sido desenvolvidas para aumentar a robustez da segmentação, usando uma filtragem adequada da imagem antes de binarização [2, 27], como apresentado na Seção 4.1.

4.2.2 Segmentação baseada em fronteiras: Gradiente

A técnica de segmentação baseada em fronteiras agrupa as bordas dos objetos que correspondem a descontinuidades do nível de cinza entre o fundo da cena e os objetos. Uma fronteira representa uma mudança abrupta no nível de intensidade entre pixels vizinhos. Muitas técnicas de detecção das bordas existem baseadas em cálculo de gradientes[1]. Uma vez que uma fronteira é definida como uma transição entre duas regiões com níveis de cinzas significativamente diferentes, pode se observar nela uma grande magnitude do valor do gradiente. O gradiente $\nabla(x,y)$ na forma mais simples pode ser calculado como segue [2]:

$$\nabla(x,y) = \sqrt{\triangle_x^2 + \triangle_y^2} \measuredangle \arctan\left(\frac{\triangle_y}{\triangle_x}\right), \tag{4.3}$$

onde $\triangle_x$ e $\triangle_y$ são os incrementos na imagem $Im(x,y)$ em x e y, respectivamente:

$$\triangle_x = Im(x+1,y) - Im(x,y), \tag{4.4}$$

$$\triangle_y = Im(x,y+1) - Im(x,y). \tag{4.5}$$

O gradiente espacial, então, mede a taxa de alteração do nível de cinza nos pixels da imagem, a direção do vetor de gradiente será a direção da variação máxima. A detecção das bordas pode ser realizada agrupando os pixels em que a magnitude do gradiente é superior a um limiar τ_g.

4.2.3 Morfologia

A morfologia é a teoria e técnica para análise e processamento de estruturas geométricas. A morfologia é comumente aplicada no processamento de imagem. Conceitos topológicos e geométricos do espaço contínuo, tais como o tamanho, forma e convexidade, são estudados

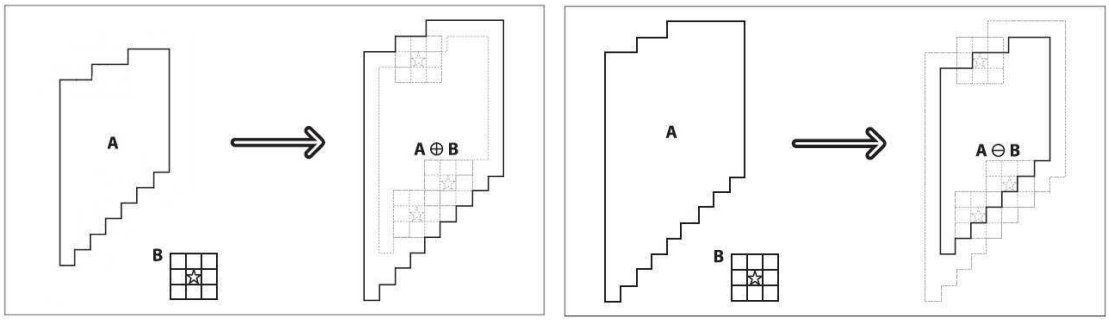
[1] Na subseção 4.2.3.3 será feita outra abordagem usando morfologia.

pela morfologia. As operações básicas morfológicas são dilatação e erosão, que permitem isolar elementos na cena, eliminar ruído e juntar elementos em uma imagem. Também pode ser usada para encontrar gradientes (por exemplo, bordas) e encontrar saliências ou furos em um elemento da imagem.

Operações morfológicas serão usadas no trabalho principalmente para concatenar pixels de uma mesma área, pois o ruído pode gerar degradações que façam que uma área definida seja interpretada como um conjunto de áreas quando na realidade é uma única região. Assim, como uma área de interesse pode ser definida como um objeto particular, o ruído pode gerar uma degradação severa da imagem que não permita a detecção do objeto.

4.2.3.1 Dilatação e Erosão

A dilatação é a convolução de uma imagem, ou uma região dela, com uma máscara ou modelo [28]. Quando a máscara é escaneada sobre a imagem, é calculado o máximo local na janela que é o maior valor dos pixels sobreposto com a máscara e logo se faz a substituição dos pixels na janela por esse valor máximo. Isto faz que as regiões de maior luminosidade em uma imagem cresçam como é esquematizado na Figura 4.1a. Observa-se que o objeto foi denominado com a letra A e a janela com a letra B.



(a) Operador Dilatação. (b) Operador Erosão.

Figura 4.1: Representação da convolução A⊕B entre o modelo A e a mascara B para os operadores dilatação e erosão.

A erosão é a operação inversa e está esquematizada na Figura 4.1b. É equivalente a calcular o mínimo local sobre a área da máscara. Erosão gera uma nova imagem, utilizando o seguinte algoritmo: a máscara é varrida sobre a imagem, calcula-se o valor mínimo do pixel sobreposto com a janela e se substitui todos os valores dos pixels dentro dessa máscara por esse valor mínimo [28, 27].

Em termos gerais, a dilatação e a erosão expandem e contraem a região A, respectivamente.

A dilatação tende a suavizar concavidades e a erosão tende a suavizar protuberâncias. Os resultados obtidos dependem do tipo de máscara que está sendo usada. A erosão permite reduzir o ruído desde que erode essas pequenas regiões, para que as grandes não sejam afetadas. A operação dilatação é usada para conectar componentes, desde que às vezes grandes regiões da imagem poderiam ser separadas em pequenos componentes como resultado do ruído, sombras ou algum elemento similar. Grandes dilatações geram a junção de componentes.

4.2.3.2 Abertura e fechamento

Os operadores de abertura e fechamento são combinações dos operadores de erosão e dilatação. No caso da abertura, primeiro e feita uma erosão e depois é feita uma dilatação da imagem com uma única mascara $h_a(k,l)$. É usado geralmente para realizar a contagem de regiões em uma imagem binária. No entanto, no fechamento é feito primeiro uma dilatação e logo uma erosão com a máscara $h_f(k,l)$. É usado na maioria dos casos para reduzir ruído e agrupar componentes [28].

Os resultados de abertura e fechamento são similares à erosão e dilatação, respectivamente. A diferença é que as áreas e formas dos componentes são aproximadamente conservadas. O efeito mais proeminente do fechamento é eliminar valores isolados que são de menor valor em intensidade do que os seus vizinhos enquanto que o efeito da abertura consiste em eliminar valores isolados que são mais elevados do que os seus vizinhos.

4.2.3.3 Gradiente Morfológico

Outro operador morfológico é o gradiente, que permite isolar perímetros de regiões. O algoritmo faz a subtração de uma dilatação da imagem com uma erosão da mesma imagem.

Aplicam-se as operações da Seção 4.2.3.1 na Figura original: operador dilatação, $dilat(Im)$, e o operador erosão, $eros(Im)$ como apresenta a Equação 4.6, chamando $Im(x,y)$ e ∇_M à imagem e ao operador gradiente, respectivamente. Finalmente é feita a subtração pixel a pixel entre o resultado da dilatação e o da erosão.

$$\nabla_M(Im(x,y)) = dilat(Im(x,y)) - eros(Im(x,y)). \tag{4.6}$$

O resultado da Equação 4.6 é o perímetro completo da região encontrado através de uma versão da imagem expandida e subtraída com uma versão contraída da região, deixando somente as bordas do perímetro.

4.3 Interpretação da imagem

Interpretação da imagem é o processo de calcular os parâmetros das características da imagem a partir das regiões ou fronteiras obtidas dos objetos no processamento de imagem. Esses parâmetros usualmente requerem o cálculo dos denominados momentos. Eles servem para ter informação da forma, posição e orientação do objeto em relação ao referêncial da câmera [2].

No trabalho de dissertação de mestrado implementou-se a técnica dos Momentos para determinar a posição e orientação do objeto em relação ao sistema de coordenadas da câmera. Usando a cinemática do manipulador, permite-se obter a pose do objeto relativa à base do braço.

4.3.1 Momentos

Os momentos de um objeto definem de forma geral as características dos contornos e regiões pela soma de todos os pixels contidos nessa fronteira [20]. A definição do momento $m_{k,i}$ de um objeto é o seguinte [28, 27]:

$$m_{k,i} = \sum_{j=1}^{l} Im(x,y)x^k y^i, \tag{4.7}$$

onde k e i são os ordens de x e y, respectivamente. A quantidade de pixels contidos nessa fronteira é l. Tem-se momentos que são de interesse para determinar alguns parâmetros das características da imagem: o momento $m_{0,0}$ é definido como a área medida em pixels desse objeto. Os momentos permitem também comparar objetos para determinar se eles pertencem ao mesmo grupo, se são similares, etc.

No entanto, os momentos calculados da Equação 4.7 não são os melhores parâmetros para realizar essas comparações pois eles dependem do sistema de coordenadas escolhido, não obtendo bons resultados na comparação se eles estão rotacionados ou transladados [20]. Por isso, são calculados os momentos centrais e normalizados.

Os momentos centrais $\mu_{k,i}$ são invariantes respeito à translação, o cálculo é dado pela Equação 4.8:

$$\mu_{k,i} = \sum_{j=1}^{l} Im(x,y)(x-\bar{x})^k (y-\bar{y})^i. \tag{4.8}$$

Observa-se que os valores de x e y estão deslocados uma quantidade $\bar{x} = \frac{m_{1,0}}{m_{0,0}}$ e $\bar{y} = \frac{m_{0,1}}{m_{0,0}}$

chamado de centro de massa σ do objeto. Essas coordenadas permitem detetar unicamente a posição do objeto ou região na imagem.

Os momentos normalizados $\eta_{k,i}$ são independente do tamanho do objeto, permitindo fazer comparações entre objetos da mesma forma mas com diferente escala [20]. Esses momentos são escalados por uma potência apropriada da área, como é observado na Equação 4.9:

$$\eta_{k,i} = \frac{\mu_{k,i}}{m_{0,0}^{(k+i+2)/2}}, \tag{4.9}$$

Se a região é assimétrica, os momentos podem se combinar como na Equação 4.10 e assim é possível caracterizar a orientação do objeto em termos do ângulo θ_5:

$$\theta_5 = \frac{1}{2}\arctan 2\left(\frac{2\mu_{1,1}}{\mu_{2,0}-\mu_{0,2}}\right) + 90°. \tag{4.10}$$

4.4 Modelagem, calibração de câmera e visão estéreo

A visão começa com a detecção da luz no espaço. Essa luz é proveniente de uma fonte de luz (pode ser o sol ou qualquer fonte de iluminação criada pelo ser humano) e quando incide nos objetos parte é absorvida. A parte não absorvida é percebida como a cor da luz. A luz refletida é capturada seja pela retina dos olhos ou por um dispositivo de captura de imagem. A geometria do raio de luz viajando do objeto para o dispositivo de captura é de muita importância na visão computacional [28]. Tanto a modelagem como a calibração de câmera são importantes para relacionar as informações obtidas pela câmera com o mundo físico tridimensional. Por isso, obter os parâmetros intrínsecos da câmera e os coeficientes de distorção necessários para fazer a correção da imagem são fundamentais. Essas correções são importantes no presente trabalho uma vez que da imagem será estimada a pose dos objetos e obstáculos, sendo mais preciso o planejamento de trajetória no intuito de evitar obstáculos.

4.4.1 Modelo de Câmera pontual

O modelo mais simples é denominado modelo de câmera pontual. Neste modelo considera-se que os raios de luz interceptados pela câmera são provenientes de fontes emissoras ou refletoras (objetos do ambiente) e viajam em linha reta projetando-se plano da imagem, com sistema de coordenadas $o_{im}(u,v)$, convergindo em um ponto denominado de centro óptico ou centro focal. Observa-se o esquema na Figura 4.2:

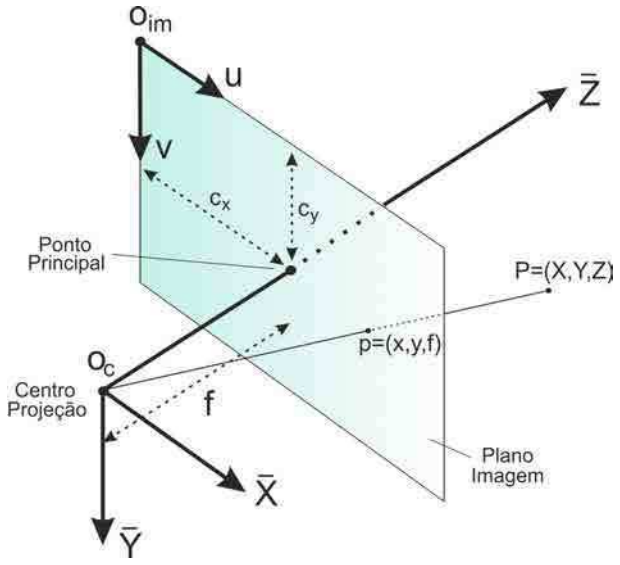


Figura 4.2: Representação do modelo de câmera pontual, indicando os parâmetros intrínsecos da câmera e seus sistemas de referencia associados.

Esse modelo leva em consideração que os raios de luz quando passam pela lente da câmera, sofrem uma inversão de posição. A suposição feita no modelo é que o objeto de coordenadas espaciais (X,Y,Z) no referêncial da câmera $o_c(\bar{X},\bar{Y},\bar{Z})$ está em uma distância grande da câmera. A orientação do sistemas de coordenadas é a mesma no modelo por simplicidade. O plano da imagem contém o ponto principal da imagem que é o ponto onde o eixo óptico Z da câmera intercepta o plano da imagem. A distância entre esse ponto e o centro focal da câmera é a distância focal f da câmera. Observando a Figura 4.2 pode-se relacionar um ponto no espaço com o correspondente no plano de imagem obtendo as seguintes relações:

$$\tan(\alpha) = \frac{x}{f} = \frac{X}{Z} \;\Rightarrow\; \frac{f}{Z} = \frac{x}{X}, \tag{4.11}$$

$$\tan(\beta) = \frac{y}{f} = \frac{Y}{Z} \;\Rightarrow\; \frac{f}{Z} = \frac{y}{Y}. \tag{4.12}$$

Escrevendo a Equação 4.11 e Equação 4.12 na forma matricial obtêm-se:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \frac{1}{Z} \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}. \tag{4.13}$$

As unidades das coordenadas (X,Y,Z) e de f estão expressas em distância (m). Como a

unidade fundamental da imagem é o pixel deve-se fazer uma transformação de unidades. Considerando que em nenhuma câmera convencional os pixels não são quadrados tem-se dois fatores k_x e k_y (com unidades de $\frac{pixel}{m}$) que representam os fatores de conversão na direção horizontal e vertical, respectivamente. Também, uma translação (c_x, c_y) $[pixel]$ do sistema de referência da imagem é necessário para relacionar-lhe com o sistema da câmera. O resultado dessas operações é mostrado na matriz da Equação 4.14 denominada equação de projeção perspectiva:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \frac{1}{Z} \underbrace{\begin{bmatrix} f.k_x & 0 & c_x & 0 \\ 0 & f.k_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{K} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}. \tag{4.14}$$

A matriz de parâmetros K apresentada na Equação 4.14 é denominada Matriz de calibração onde os coeficientes dessa matriz são conhecidos como os parâmetros intrínsecos da câmera, calculados calibrando a câmera manualmente ou procurando a sua especificação técnica [28].

Observa-se que no modelo de câmera pontual, pouca luz incide pela abertura. Na prática as imagens seriam tomadas muito lentamente pois a câmera deverá esperar por uma certa quantidade de luz para a imagem ser nítida. Para formar imagens em um ritmo mais rápido, tem-se que reunir uma grande quantidade de luz sobre uma área mais ampla e dobrar (enfocar) a luz a convergir no ponto de projeção. Para conseguir isso, usam-se lentes. Uma lente pode concentrar uma grande quantidade de luz em um ponto e fornece imagens de forma mais rápida, mas com o custo de introduzir distorções. Para sua eliminação usa-se a calibraçao da lente.

4.4.2 Calibração

Calibração de câmera é importante para relacionar as informações obtidas pela câmera com o espaço de trabalho do manipulador. A relação das unidades de medida de uma câmera (pixels) e as unidades do mundo físico (metro) é uma componente crítica em qualquer intento por reconstruir uma cena tridimensional. O processo de calibração de câmera permite obter o modelo da geometria da câmera e de distorção da lente. Essas duas informações definem os parâmetros intrínsecos da câmera. Com esses parâmetros pode-se corrigir a distorção da lente e assim interpretar o mundo físico com maior precisão [2, 28].

Na rotina proposta, o método de calibração é apontar a câmera em uma estrutura conhecida que tem muitos pontos identificáveis. Em princípio, qualquer objeto caracterizado apropriadamente poderia ser usado como um objeto de calibração da câmera. Ao ver esta estrutura a partir

de uma variedade de ângulos, é possível em seguida, calcular a posição e a orientação relativa da câmera no momento de cada imagem, bem como os parâmetros intrínsecos da câmera. A fim de fornecer várias exibições, rotaciona-se e translada-se o objeto.

4.4.2.1 Distorção da Lente

Em teoria, é possível definir uma lente que não introduz distorções. Na prática, no entanto, nenhuma lente é perfeita. Isto é principalmente por razões de fabricação, é muito mais fácil fazer uma lente "esférica" do que para fazer uma lente "parabólica" ideal. Aqui se descrevem duas distorções principais da lente: Distorções radiais surgem como resultado da forma da lente, enquanto que as distorções tangenciais surgem do processo de montagem da câmera [28].

A distorção radial é uma deformação da imagem que acontece por causa das lentes não ideais, ela é observada e acentuada nos cantos da imagem. O efeito é conhecido como barril ou efeito olho de peixe. Com algumas lentes de baixo custo, os raios mais distantes do centro da lente são dobrados mais do que às próximas. Distorção é particularmente visível em webcams de baixo custo. Para distorções radiais, a distorção é nula no centro óptico da imagem e aumenta em direção à periferia. Na prática, essa distorção é pequena e pode ser caracterizada pelos primeiros termos da série de Taylor em torno do centro de $r = x^2 + y^2$. Nas Equações 4.15 a 4.18 é representada a posição original (x, y) do pixel com distorção radial e tangencial [28].

$$x_{corrig} = x\left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right), \tag{4.15}$$

$$y_{corrig} = y\left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right), \tag{4.16}$$

onde k_1, k_2 e k_3 são os coeficientes da serie de Taylor para ser determinados em função da câmera utilizada.

O segundo tipo de distorção que acontece nas câmeras é a tangencial. Esse tipo de deformação acontece pelos defeitos na montagem onde a lente não está exatamente paralela ao plano de imagem. Essa distorção é caracterizada por dois termos (p_1 e p_2) tendo as seguintes equações [28]:

$$x_{corrig} = x + \left[2p_1 y + p_2(r^2 + 2x^2)\right], \tag{4.17}$$

$$y_{corrig} = y + \left[2p_2 x + p_1(r^2 + 2y^2)\right]. \tag{4.18}$$

Em geral o efeito de distorção mais acentuado é o radial, em câmeras de maior custo o efeito tangencial pode ser desprezível.

4.4.3 Visão Estéreo

A extração de informação de profundidade é feita através do sensor RGB-D utilizando o par projetor IR/câmera CMOS junto com a câmera RGB instaladas nele. Essa informação permite reconstruir a cena tridimensional [2].

4.4.3.1 Sensor RGB-D Kinect

O sensor Kinect é um dispositivo periférico da consola Xbox 360 da Microsoft©. Consiste em um conjunto de microfones nas laterais, um motor DC para controle de inclinação, uma câmera RGB, uma câmera monocromática CMOS e um projetor infravermelho, como observa-se na Figura 4.3.

Figura 4.3: Sensor Kinect, indicando o atuador e sensores presentes.

O projetor produz um padrão de luz estruturado na cena de trabalho, o qual é capturado pela câmera CMOS. O deslocamento entre a câmera monocromática e o projetor permite o cálculo da distância de objetos usando retificação, correspondência e triangulação. O sensor permite capturar imagens RGB com uma resolução de até $640x480$ pixels a 30 quadros por segundo. A faixa de profundidade capturada é de $0,6m$ a $3,5m$. Possui uma faixa de visão de 57° e 43° na horizontal e na vertical, respectivamente. O dispositivo é robusto com a variação de luminosidade da cena.

A visão estéreo inclui quatro passos fundamentais os quais são [28]:

1. Remover a distorção radial e tangencial da lente como foi realizado na seção anterior. Assim obtêm-se imagens não distorcidas através do conhecimento do mapa de distorção.

2. Ajustar o ângulo e distância entre as câmeras. Esse processo chama-se retificação. Da retificação obtêm-se imagens que são alinhadas nas filas, isto quer dizer que os dois planos das imagens são coplanares e as suas filas estão alinhadas, ou seja, com a mesma coordenada y. A retificação realiza-se com a matrizes de rotação R_{cs} e translação T_{cs} entre a câmera CMOS e a RGB do Kinect obtidas no processo de calibração do sensor. Essas matrizes relacionam pixels de uma câmera a outra da seguinte forma:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix}_{rgb} = R_{cs} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + T_{cs}\,. \tag{4.19}$$

3. Encontrar as mesmas características na imagem direita e esquerda, o processo é denominado como correspondência. Obtendo a correspondência entre os pontos pode se encontrar o mapa de disparidade, onde as disparidades são a diferença entre as coordenadas x de uma mesma característica observada nas duas imagens. Uma vez feito o processo de calibração e retificação, obtém-se a disparidade do Kinect d_k que permite obter a profundidade Z da cena. A disparidade do Kinect está representada a 11 bits com valores de 0 a 2048. O Kinect possui uma disparidade offset de $d_{offset} = 1090$.

4. Conhecendo a configuração geométrica das câmeras e o mapa de disparidade, pode-se obter as distâncias espaciais através do processo chamado triangulação reprojetando os pontos no espaço tridimensional e obtendo o mapa de profundidade. A profundidade da cena é obtida através da Equação 4.20:

$$Z = \frac{bf}{\frac{1}{8}(d_{offset} - d_k)}\,, \tag{4.20}$$

   com $b = 7{,}5\,cm$ a distância entre o projetor e a câmera CMOS infravermelha do Kinect e f a distância focal do sensor em pixels.

Finalmente, utilizando a inversa da Equação 4.14 transforma-se a valores mensuráveis no espaço cartesiano (X,Y,Z).

4.4.3.2 Nuvem de pontos

A nuvem de pontos é um conjunto de vértices em um sistema de coordenadas tridimensional. Tipicamente, representa uma superfície externa de um objeto.

A saída do sensor RGB-D é representada como a nuvem de pontos medida pelo dispositivo através da biblioteca PCL [29]. O conjunto de pontos obtido serve para vários propósitos, in-

cluindo modelagens tridimensionais CAD de diversos objetos, medidas, qualidade de inspeção de produtos, etc.

Enquanto as nuvens de pontos podem ser representadas e inspecionados, geralmente esses vértices não são diretamente utilizáveis na maioria das aplicações tridimensionais e, portanto, geralmente são convertidos em malhas de polígonos ou modelos de malhas de triângulo para a reconstrução da superfície. Existem muitas técnicas para a conversão de uma nuvem de pontos a uma superfície tridimensional. Algumas abordagens, como a triangulação de Delaunay [29], permitem a construção de redes de triângulos ao longo dos vértices existentes da nuvem de pontos.

4.5 Considerações finais

Neste capítulo foi apresentado o embasamento teórico das técnicas de processamento de imagens que são implementadas no projeto. Foram brevemente explicadas técnicas de pré-processamento de imagem como filtragem que permite a homogeneização de regiões. Além disso, descreveram-se técnicas de extração das bordas e áreas através da segmentação observando as vantagens de fazer um pré-processamento da imagem. Depois foi analisada a importância de realizar uma calibração das câmeras e um modelo das mesmas, isto permite uma extração mais precisa dos parâmetros das características da imagem. Finalmente é apresentado o sensor RGB-D e as técnicas de visão estéreo que permitem a captura da profundidade da cena de trabalho para posterior reconstrução da cena tridimensional.

Capítulo 5

Integração do sistema

A integração das técnicas de visão e o algoritmo de planejamento de trajetória no manipulador serão abordadas nesta seção. Para isso, primeiramente, deve-se definir os obstáculos e objetos de interesse da cena de trabalho. A restrição é que eles devem encontrar-se na área de trabalho. Para a realização de uma dada tarefa de visão computacional é requerida apenas uma pequena parte da toda a informação que compõe a imagem, como por exemplo certos objetos de interesse. Assim surge a necessidade de reconhecer apenas os elementos importantes presentes no cenário.

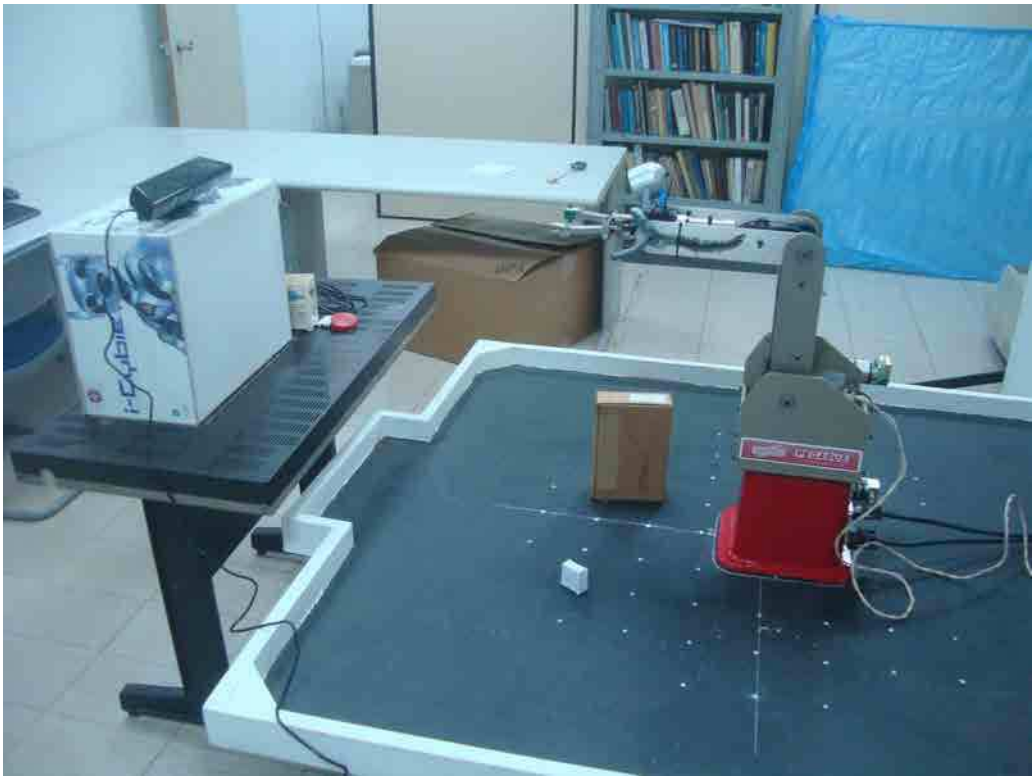


Figura 5.1: Caso particular de espaço de trabalho com um único obstáculo e objeto de interesse.

O manipulador, os sensores de visão, objeto de interesse e obstáculos apresentam-se na Figura 5.1 para um caso particular de espaço de trabalho, observa-se somente um único objeto de interesse (a caixa branca) e o obstáculo. Podem ser definidos diversas cenas de trabalho e a quantidade de obstáculos pode ser variavél.

Na cena de trabalho os objetos podem ser considerados como objetos de interesse ou obstáculos. Os objetos que dificultam ou impedem a realização de uma tarefa específica são considerados obstáculos. Por exemplo, máquinas que se encontram na trajetória de manipulação do braço ou mesmo as peças a ser manipuladas quando não são de interesse em uma tarefa pré determinada. Independentemente da natureza do manipulador, um modelo dos obstáculos é necessário. No projeto, esses obstáculos são estáticos na cena de manipulação e simulam obstáculos em situações reais encontradas diariamente na indústria.

Os objetos de interesse na cena são as peças que devem ser manipuladas pelo braço e o operário da planta deve especificar-lhes *a priori* antes da manipulação. Estes objetos estarão colocados em uma superfície plana simulando um produto de manufatura já pronto para ser enviado a outro processo, por exemplo, de empacotamento ou controle de qualidade. O tamanho desses objetos não deve ser superior a $8,5\,cm$ de diâmetro ou comprimento da face menor do paralelepípedo, respectivamente, tendo em conta que essa é a máxima abertura da garra do manipulador e também devem ter um peso menor a $1\,kg$ que é a máximo que o efetuador pode carregar e manipular. Esses objetos estarão colocados de forma aleatória, no chão da cena e na área de visão e alcance do manipulador.

O reconhecimento de objetos é uma área vasta da visão computacional, que abarca uma grande variedade de metodologias e que tiram partido das distintas características que os objetos a manipular apresentam no meio onde se encontram. Na seguinte seção, apresenta-se o sistema de controle que implementa o roteamento probabilístico e o sistema de visao computacional que define esses objetos.

5.1 Sistema de controle

A interação das técnicas de visão estéreo, processamento, segmentação, interpretação de imagens e planejamento de movimento definem as malhas de controle. O esquema de controle que é proposto está esquematizado na Figura 2.1 e repetido na Figura 5.2 por conveniência:

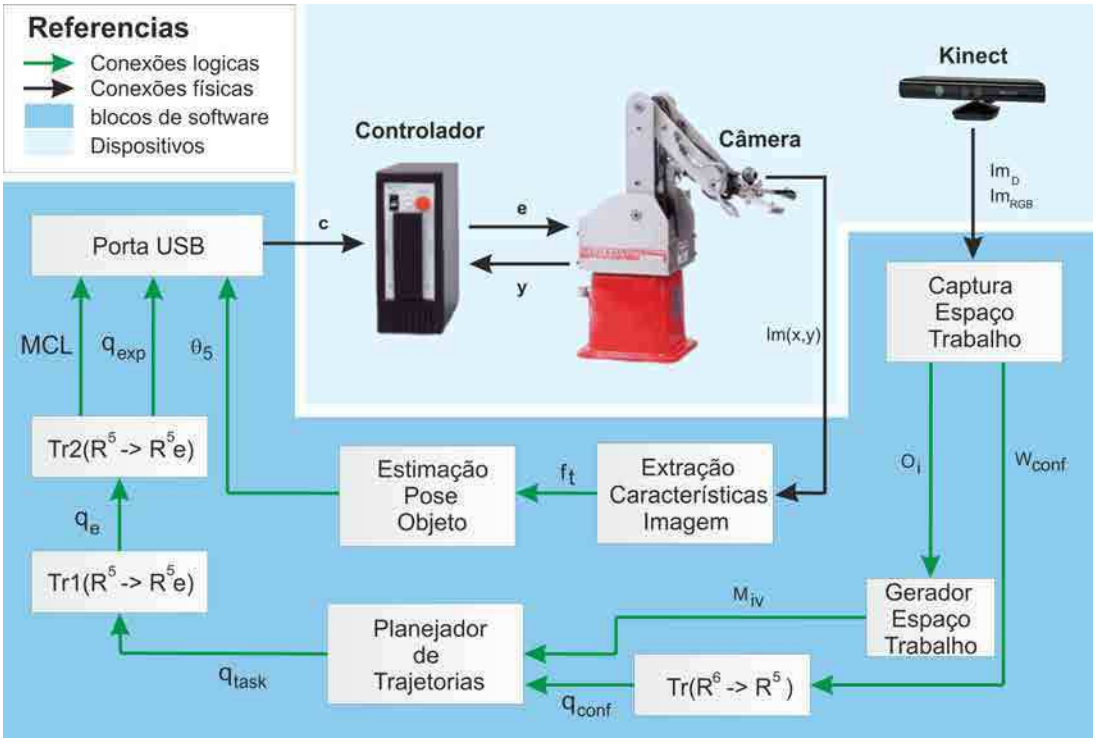


Figura 5.2: Sistema proposto apresentando os módulos e malhas de controle.

Observa-se na Figura 5.2 que o sistema de controle forma três malhas, sendo uma rápida interna de controle de juntas (comandos *y*, *e*) e duas mais lentas e externas de visão e planejamento com origem nos dispositivos de captura de imagem. Não são consideradas restrições cinemáticas nem as singularidades do manipulador. Na realidade, as singularidades já são consideradas no controlador do manipulador da Amatrol, permitindo separar as singularidades do mecanismo da malha externa de controle. Assim, desde o ponto de vista dos algoritmos de planejamento de trajetória e visão, o braço pode ser considerado como um dispositivo ideal de movimento cartesiano [30].

A descrição e função de cada bloco é explicada como segue:

5.1.1 Captura do espaço de trabalho

A modelagem da cena de trabalho é o primeiro passo na determinação espacial do robô, objetos e obstáculos. O dispositivo de captura usado para esse propósito é o Kinect.

Isto é possível usando a idéia básica de segmentação baseada em regiões da Seção 4.2 aplicada para nuvem de pontos.

O propósito do algoritmo é agrupar pontos da nuvem de pontos que estejam o suficientemente próximos em termos de restrições de suavidade. A saída do algoritmo é o conjunto

de agrupamentos de pontos, onde cada agrupamento é considerado parte do mesmo objeto. O algoritmo está baseado na comparação de ângulos entre as normais dos pontos. Primeiro o algoritmo ordena os pontos por sua curvatura. As regiões começam crescer por pontos que tem menores valores de curvatura, chamados de "sementes", colocadas nas superfícies planas. O crescimento desde as áreas mais planas é escolhido porque permite a redução do número de segmentos. É comparado o ângulo entre as normais de cada ponto vizinho a semente com o próprio ângulo da semente. Se o ângulo é menor que um limiar então esse vizinho é adicionado na região.

O bloco foi implementado em Matlab onde é chamada a função programada em c++ baseada na biblioteca de Burrus [31]. A adquisição dos modelos são obtidas com uma única imagem de profundidade da cena (Im_D e Im_{RGB}). O sensor RGB-D captura a cena de trabalho onde os objetos estão colocados em uma superfície plana. O algoritmo estima a equação do plano da mesa e depois extrai os agrupamentos (objetos) usando a biblioteca PCL [29]. A estimação dos modelos é feita por extrusão dos pontos superiores dos agrupamentos até a mesa, assumindo que os objetos são planos. A reprojeção dos pontos na imagem a cor é feita como explicado em Seção 4.4.3 e é estimado o volume, cor e centro de massa. As características extraídas de cada objeto permitem definir a posição do objeto de interesse (usando o centro de massa σ) e pose dos obstáculos. Tanto a pose do objeto de interesse como a final do efetuador (a pose desejada do objeto) são definidas em W_{conf} que é uma matriz 2×6. A pose do objeto é definida pela posição do centro de massa e a orientação. Como a orientação do objeto é encontrada na malha da câmera monocular (seções 5.1.9 e 5.1.10), então a orientação é definida *a priori* em $\theta_5 = 0º$.

Usando o volume e cor calculado é determinado o objeto de interesse, desta forma é eliminada a sua nuvem de pontos da modelagem da cena. Também é calculado para cada objeto o ponto (X,Y,Z) com menor distância euclidiana d ao sistema de coordenadas da base do manipulador de cada objeto. Tendo o valor d de cada objeto pode-se eliminar a nuvem de pontos do manipulador da cena de trabalho pois o valor d do manipulador será o mínimo de entre todos os objetos. A nuvem de pontos de cada obstáculo i (o_i), armazenada em arquivos VRML 1.0, é a saída do bloco.

5.1.2 Gerador do espaço de trabalho

O bloco foi implementado em Matlab. Recebe a nuvem de pontos dos obstáculos o_i e cria um arquivo M_{iv} do espaço de trabalho, em formato VRML, contendo esses obstáculos, o modelo do robô apresentado na Seção 3.5 e o modelo de chão considerado plano. O modelo M_{iv} será em seguida enviado para o planejador de trajetória.

As nuvens de pontos dos obstáculos o_i são transformadas em malhas mediante a Triangulação de Delaunay. A triangulação de Delaunay é uma rede de triângulos que satisfaz a condição que a esfera que contém cada triangulo não deve conter nenhum vértice de outro triangulo. Em modelos tridimensionais de objetos usam-se redes de polígonos para modelar a superfície do objeto. Usou-se essa modelagem pois os triângulos são os polígonos mais simples de representar e tem muitas propriedades favoráveis. Segundo a definição de Delaunay, a esfera é vazia, se não contém outros vértices além dos três do triangulo contido nela. A condição, chamada de Delaunay, assegura que os ângulos interiores dos triângulos são os maiores possíveis maximizando os ângulos dos triângulos e a longitude dos lados dos triângulos é mínima. A triangulação é única se em nenhuma borda da esfera tem mais de três vértices. A união dos vértices em triângulos usando esse método foi baseado na programação feita em [32]. Essas malhas o_i são armazenadas como triângulos em arquivos VRML 1.0.

5.1.3 $Tr\left(\mathbb{R}^6 \rightarrow \mathbb{R}^5\right)$

O bloco permite transformar as configurações de $\mathbb{R}^6$ a $\mathbb{R}^5$ através do operador Tr que é uma transformação usando a cinemática inversa da estrutura. O modulo foi programado em Matlab e calcula a posição do alvo e a pose final do efetuador W_{conf} (matriz de $2x6$) em função do espaço de configuração do manipulador q_{conf} (matriz de $2x5$), dando os ângulos das juntas equivalentes à posição espacial do objeto. Para a obtenção dessas configurações usa-se a cinemática inversa do manipulador da Seção 3.4.1.2. Os ângulos são em seguida transformados na forma normalizada para serem fornecidos no algoritmo de roteamento, como explicado na Seção 3.6 através das Equações 3.21 a 3.25. As duas configurações no espaço de configuração q_{conf} são salvas em um arquivo de texto e são usadas pelo planejador de trajetórias para gerar a rota livre de obstáculos.

5.1.4 Planejador de trajetória

Na presença de obstáculos, é necessário planejar movimentos que permitam executar tarefas evitando colisões. Existem diferentes algoritmos que lidam com esse cálculo. Um planejamento de movimento exitoso permite realizar a trajetória de uma posição inicial até uma posição final evitando obstáculos.

O desenvolvimento de métodos para o planejamento de movimento é complexo: O pensamento intuitivo usado pelos seres humanos para se movimentar entre obstáculos com segurança ainda contínua sendo uma tarefa difícil, ainda impossível, de ser replicada em um algoritmo para

ser executado por algum robô [2]. Atualmente o planejamento de movimento é um tópico de pesquisa, com contribuições provenientes de diferentes áreas, tais como a teoria de algoritmos, geometria computacional e controle automático [2].

O problema de planejamento pode ser especificado da seguinte forma: Dada uma posição inicial e final do manipulador no espaço de trabalho, encontrar um caminho existente (sequência contínua de poses do braço) que leve o robô entre essas duas posições sem colidir nem entrar em contato com obstáculos, dando uma falha ou erro se tal trajetória não existe ou não é encontrada. A suposição tomada é que os obstáculos serão fixos no espaço de trabalho. No entanto, os métodos que resolvem com sucesso esta versão simplificada do problema levam a uma extensão do algoritmo de um modo mais geral que pode abranger situações mais complexas [2].

O bloco de planejamento é baseado na biblioteca de Schwarzer *et al* [33] que recebe como entrada o arquivo de espaço de trabalho M_{iv} e as configurações q_{conf} criando a trajetória livre de obstáculos. O módulo é executado três vezes e é escolhida a trajetória com menos configurações. A saída q_{task} é uma matriz de $m \times 5$ onde m é o número mínimo de configurações normalizadas que realizam a tarefa. O módulo também implementa um suavizador de trajetórias que permite criar atalhos entre as configurações obtendo a trajetória de menor percurso (pode não ser a de menor quantidade de configurações). A saída q_{task} do bloco é um arquivo onde é especificada a tarefa a realizar pelo controlador do manipulador. A biblioteca implementa o algoritmo de roteamento SBL explicada na Seção 3.3.4.2.

5.1.5 $Tr1\,(\mathbb{R}^5 \rightarrow \mathbb{R}^5 e)$

Bloco implementado em Matlab que recebe a tarefa no espaço de configuração normalizada do q_{task} e transforma esses ângulos em valores de encoders q_e (matriz de $m \times 5$) dos motores das juntas do braço como explicado na Seção 3.6 através das Equações 3.26 a 3.30. Esses valores de encoders são os que o controlador pode ler para realizar os movimentos.

5.1.6 $Tr2\,(\mathbb{R}^5 \rightarrow \mathbb{R}^5 e)$

Sabendo que o controlador não permite passagens de configurações da garra em tempo real via porta serial, é necessário realizar isto previamente antes da execução da aplicação do manipulador. Então a função $Tr2\,(\mathbb{R}^5 \rightarrow \mathbb{R}^5 e)$, similar à $Tr1\,(\mathbb{R}^5 \rightarrow \mathbb{R}^5 e)$, implementada em Matlab define 180 orientações da garra para a posição do objeto, definindo assim, 180 poses de garra das quais uma delas vai se aproximar à pose do objeto de interesse.

O bloco encarrega-se de gerar para o controlador dois arquivos: o primeiro possui os valores

de encoders das posições achadas no planejador somadas as 180 posições restantes correspondentes à variação da junta θ_5 gerando q_{exp} que é uma matriz de $(180+m)\times 5$.

O segundo arquivo possui a programação, feita na linguagem MCL, que fará o controlador realizar os movimentos livres de obstáculos, define os retrasos entre os comandos, a comunicação via porta serial com o sistema de visão da câmera montada no efetuador, quando pegar o objeto, liberá-lo, etc. Também permite a interação com as entradas e saídas do PLC do manipulador: definir quando ligar uma esteira, uma bomba ou motor, entre outras aplicações. Finalmente é aberto um aplicativo [34] que permite automatizar a API do controlador. Isto é devido a que essa aplicação não pode ser controlada pela via de comandos do sistema.

5.1.7 Porta USB

O bloco tem a função de realizar a integração lógica e física entre as duas malhas de visão além de comunicar-se constantemente com o controlador do manipulador através da porta USB. Recebe como entrada o ângulo θ_5 que é a orientação obtida na malha de visão mono quando a câmera do manipulador posiciona-se sobre o objeto. Também recebe as saídas do bloco $Tr2\,(\mathbb{R}^5 \rightarrow \mathbb{R}^5 e)$**.** A porta serial é uma porta virtual que permite enviar o ângulo θ_5 para completar a pose do objeto e o manipulador conseguir pegá-lo corretamente. A saída do bloco são os comandos c necessários para o controlador realizar os movimentos desejados.

5.1.8 Controlador do manipulador

O controlador recebe os comandos c e as medidas y feitas pelos encoders do manipulador. Mediante o uso da sua cinemática direta envia os comandos u para os motores serem acionados atuando nas juntas do manipulador para a sequência de poses, com velocidades e acelerações das juntas definidas. As trajetórias $q(t)$ geradas entre as configurações obtidas pelo planejador de trajetórias são lineares:

$$q_m(t) = \left(\frac{q_\beta - q_\alpha}{t_1 - t_0}\right)(t - t_1) + q_\beta, \qquad t_0 \leqslant t \leqslant t_1. \tag{5.1}$$

A trajetória $q(t) \in \mathbb{C}$ é definida entre duas configurações aleatórias $q_\alpha \in \mathbb{R}^5$ e $q_\beta \in \mathbb{R}^5$ e ambas $\in \mathbb{C}$.

5.1.9 Extração das características da imagem

O objetivo do bloco é a extração de informação a partir da imagem Im, o que fornece uma descrição sintética e robusta dos objetos de interesse na cena, através das chamadas características da imagem f_t, como são áreas, bordas, perímetros, etc. O objeto de interesse é capturado tomando uma imagem superior dele com a câmera IR do braço. A posição do objeto é fornecida pelo sensor RGB-D.

Para a obtenção das características do objeto a ser manipulado pela garra, primeiramente usa-se o filtro de média (Seção 4.1.1) que permite a homogeneização da luminosidade e saturação da imagem obtendo-se uma extração mais eficiente das características desejadas, como áreas e bordas. O seu uso permite a redução do ruido com um custo computacional baixo.

Após isso, foi implementada a segmentação baseada em regiões explicada na Seção 4.2.1, que permite obter uma representação adequada da imagem para uma identificação mais simples de características mensuráveis em uma imagem. Usa-se a técnica de limiarização para transformar a imagem de escala de cinzas a binária e assim extrair os objetos desejados através das áreas encontradas. Também usam-se operações morfológicas (Seção 4.2.3) como abertura e fechamento que combinam os operadores de dilatação e erosão conservando a forma dos objetos e eliminando valores isolados em relação com pixels vizinhos. A implementação do bloco foi feita em OpenCV [35].

5.1.10 Estimação da pose do objeto

Bloco implementado em OpenCV que se encarrega da interpretação da imagem da câmera do braço. Preocupa-se com a medição dos parâmetros de recursos da imagem a partir das regiões obtidas. Com as características f_t obtidas do bloco anterior é calculada a orientação do objeto de interesse a ser manipulado com a técnica dos Momentos (Seção 4.3.1). A saída do bloco é o ângulo de orientação θ_5 que completa a pose W_{conf} fornecida pelo Kinect.

5.2 Inicialização da plataforma

A inicialização do sistema engloba todo o processo prévio de configuração para o correto funcionamento da plataforma. A inicialização é de grande importância e inclui:

- Inicializar o manipulador, verificando questões de segurança como o fornecimento de energía, as saídas do PLC e as sinais de habilitação do controlador. Levar o manipulador

a configuração q_{home} que permite a inicialização de todos os encoders;

- Alinhamento entre os sistemas de referências do sensor RGB-D e a base do manipulador. Isto é feito determinando a matriz ${}^{cam}_{bas}H$ que relaciona coordenadas do sistema de referência do sensor RGB-D em função de coordenadas da base do manipulador. Assim, tanto a cena de trabalho como as configurações do efetuador são especificadas com origem no sistemas de coordenadas da base do braço.

- Calibração dos sensores de vídeo que permite medir paramêtros no mundo físico com maior precisão;

- Definir a pose final ou desejada do objeto de interesse;

- Definir os parâmetros do algoritmo de roteamento (ε, ρ, s) e sistema de visão (filtros, máscaras e limiares).

5.3 Considerações finais

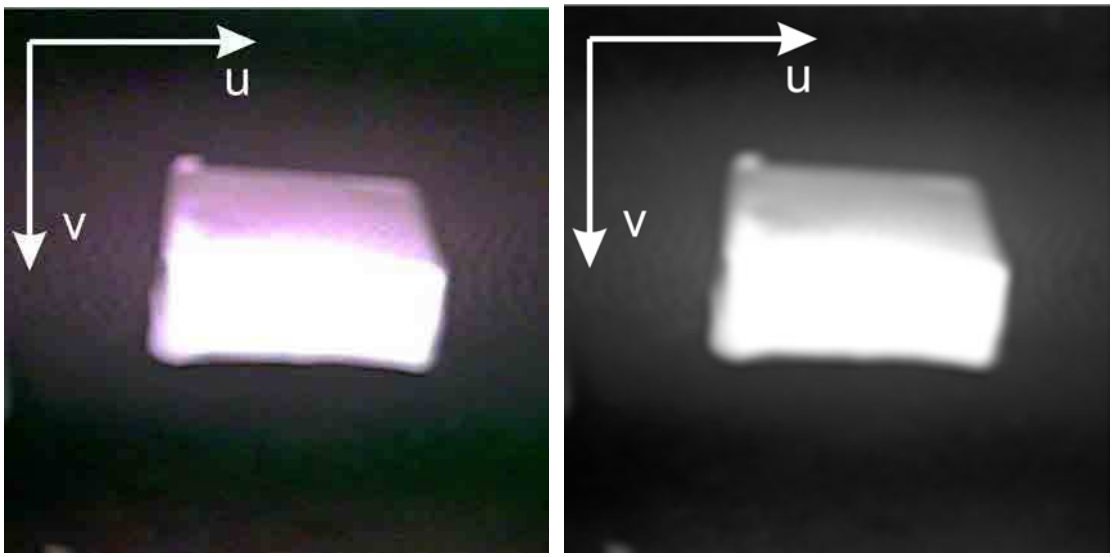
No capítulo foram explicados os blocos das malhas de controle e como estes são integrados para a deteção de objetos e obstáculos e a posterior criação da trajetória livre de colisões. O sistema escolhido consiste de três malhas, com a malha rápida interna que permitem realizar o controle de juntas e duas mais lentas de visão computacional para a detecção do objeto de interesse e obstáculos. Observa-se que a posição do objeto de interesse é definida na malha externa de visão e a orientação é obtida em tempo real pela malha de visão da câmera monocular uma vez que o efetuador posiciona-se sobre o objeto. Finalmente foi abordada a inicialização da plataforma concluindo que é de grande importância para seu correto funcionamento.

Capítulo 6

Resultados experimentais

6.1 Plataforma de visão

No pré-processamento da imagem, técnicas de filtragem de média foram aplicadas. A implementação da suavização na imagem da câmera monocular facilita o posterior reconhecimento de objetos de interesse, além de reduzir o ruído. Vários testes foram feitos com as técnicas explicadas na Seção 4.1. A câmera monocular toma uma imagem *Im* de 640×480 pixels a qual é cortada (a 360×360 pixels) pelo sistema de visão para centrar o objeto de interesse (Figura 6.1a). Observa-se pouca luminosidade da cena e a detecção IR da câmera foi ativada.



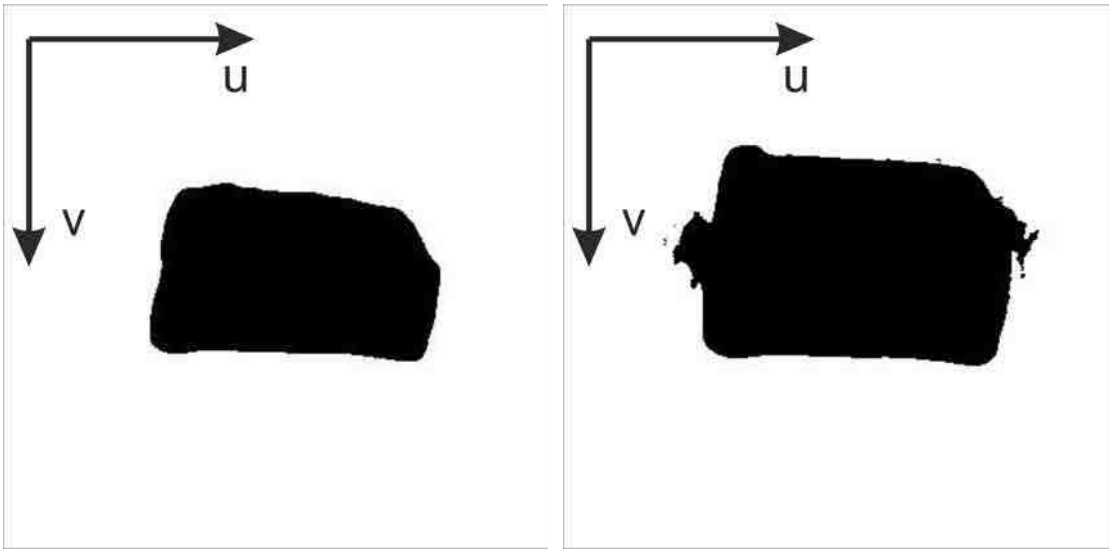
(a) Imagem original. (b) Filtragem por média.

Figura 6.1: Utilização do filtro de média na imagem capturada pela câmera monocular do efetuador.

Transformando a imagem da Figura 6.1a a escala de cinzas (cada pixel foi representado em $\vartheta = 8$ bits) e aplicando a Equação 4.1 do filtro de média na imagem obtêm-se a imagem resultante na Figura 6.1b.

Pode-se observar que na aplicação do filtro obteve-se uma imagem um pouco mais nítida e foi reduzido o ruido. No entanto, a redução de qualidade permitiu homogeneizar as superfícies e isso permite uma detecção de objetos mais simples. A janela aplicada $h(k,l)$ foi quadrada de $K = L = 7$ pixels na imagem de resolução 360×360 pixels. A escolha da janela deve-se a que com janelas menores não se observaram vantagens do seu uso e janelas maiores geram uma suavização muito acentuada onde fica comprometida deteção da região do objeto.

Na seguinte etapa de segmentação, são apresentados os resultados da aplicação da técnica de limiarização. Realizaram-se vários testes incrementando o limiar em 10 unidades no intervalo $[0, 2^8]$ para observar os efeitos no objeto. Resultados da limiarização na imagem da Figura 6.1b apresentam-se na Figura 6.2 para o caso de um limiar $\tau = 170$ (Figura 6.2a) e $\tau = 100$ (Figura 6.2b).



(a) Técnica de limiarização com $\tau = 170$.

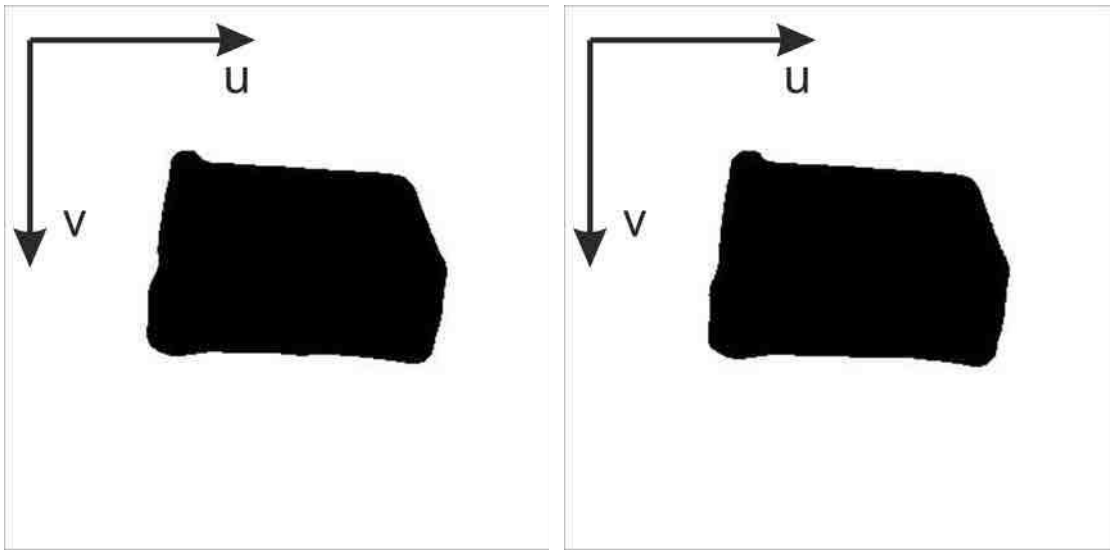
(b) Técnica de limiarização com $\tau = 100$.

Figura 6.2: Utilização do operador limiarização na câmera monocular do efetuador, indicando os limiares usados.

Observa-se que o resultado de aplicar limiarização produz uma imagem binária onde as duas regiões estão definidas. A área de objeto é composta pela região em preto. Essa área varia em função do limiar usado. Pode-se observar que disparos próximos de $\tau = 100$ representam melhor o objeto de interesse. Disparos baixos (menores a $\tau = 70$) distorcem a orientação e não permitem uma boa detecção do contorno do objeto. No entanto, disparos altos como $\tau = 170$

embora não seja a melhor escolha nesta imagem, mantem a orientação do objeto.

As operações básicas de dilatação e erosão podem ser combinadas para obter dois operadores mais interessantes em processamento de imagem: abertura e fechamento. Como estudado no fundamento teórico, na abertura primeiro é realizada uma erosão e depois uma dilatação da imagem. O resultado da aplicação do operador abertura em Figura 6.2a é apresentado na Figura 6.3a, usando uma janela $h_a(k,l)$ quadrada de 7×7 pixels aplicando-se o operador duas vezes para obter melhor conservação da área.



(a) Operador abertura. (b) Operador Fechamento.

Figura 6.3: Utilização dos operadores abertura e fechamento na câmera monocular do efetuador no intuito de conservação de área.

A aplicação do operador fechamento sobre a Figura 6.2b é mostrada na Figura 6.3b. O operador permite concatenar pixels muito próximos conservando as áreas dos mesmos. O operador usa uma janela $h_f(k,l)$ quadrada de 7×7 pixels e foi aplicado uma vez. Pode-se observar a conservação de área em ambas imagens. Dos testes feitos com o objeto colocado em diversas posições da mesa com diferentes luminosidades concluiu-se que para abranger todas as possibilidades tem que se escolher um limiar alto embora não seja conservada a área. Assim na implementação escolheu-se $\tau = 170$ com operador de fechamento aplicado uma vez com máscaras quadradas e de 7×7 pixels para o filtro de média e fechamento. A escolha de $\tau = 170$ não justifica a aplicação de um número maior de vezes do operador morfológico pois a área não se conservaráa pelo alto limiar utilizado. A combinação das técnicas de filtragem, segmentação e identificação permitiram obter um ângulo de $\theta_5 = 94°$.

6.1.1 Calibração das câmeras

Para o cálculo dos parâmetros intrínsecos da câmera do braço, foi utilizada a rotina de calibração manual feita em Matlab por Bouguet [36]. Escolheu-se essa rotina de calibração, pois apresentou melhores resultados que a rotina automática implementada na biblioteca OpenCV [37], devido à grande distorção da lente.

Foram tomadas 30 fotos e, usando inspeção visual, foram escolhidas as 20 imagens mais nítidas (Figura 6.5). Foi escolhido um tabuleiro de xadrez de 8×7 por ser um objeto plano e de padrão regular observando os cantos de cada quadrado do padrão e aplicando um refinamento a nível de subpixel para maior precisão.

Na Figura 6.4 observa-se a posição e orientação para as 20 imagens. As imagens caracterizam-se por terem sido obtidas de vários ângulos e posições para fornecer maior informação no cálculo e reduzir o erro na calibração.

Realizada a calibração foram obtidos os parâmetros intrínsecos e de distorção da lente da câmera do efetuador visualizados na Tabela 6.1.

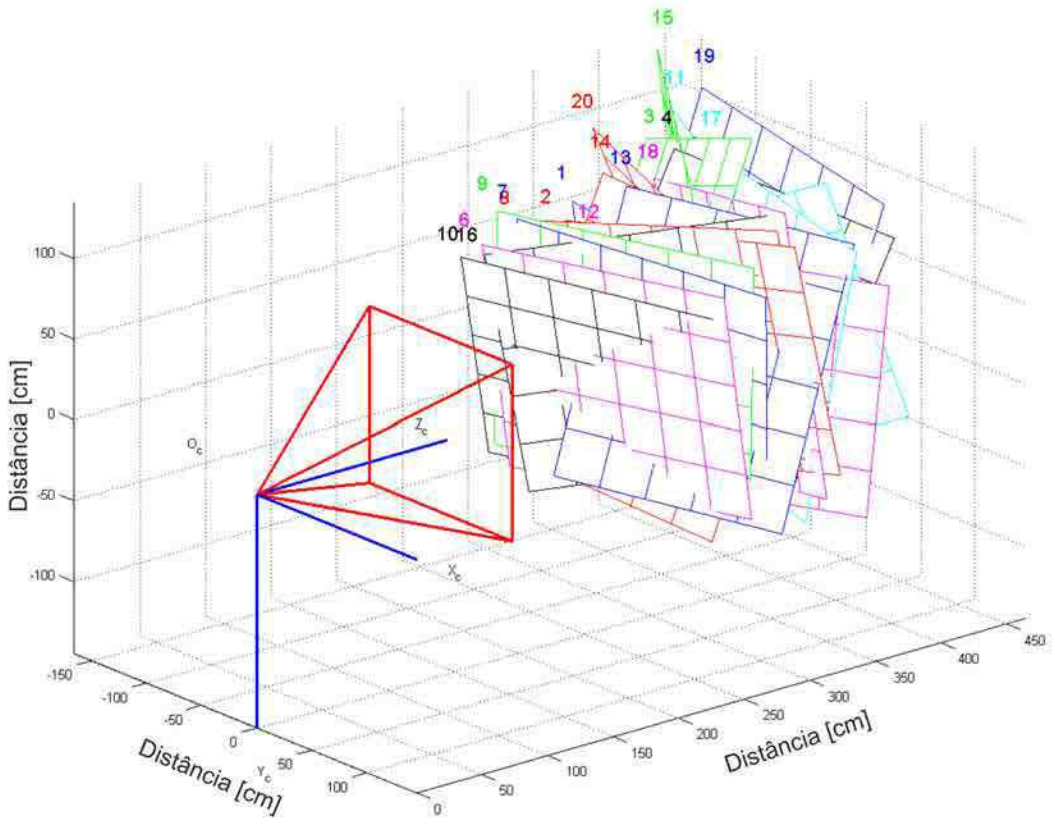


Figura 6.4: Pose do tabuleiro respeito à câmera para as 20 imagens.

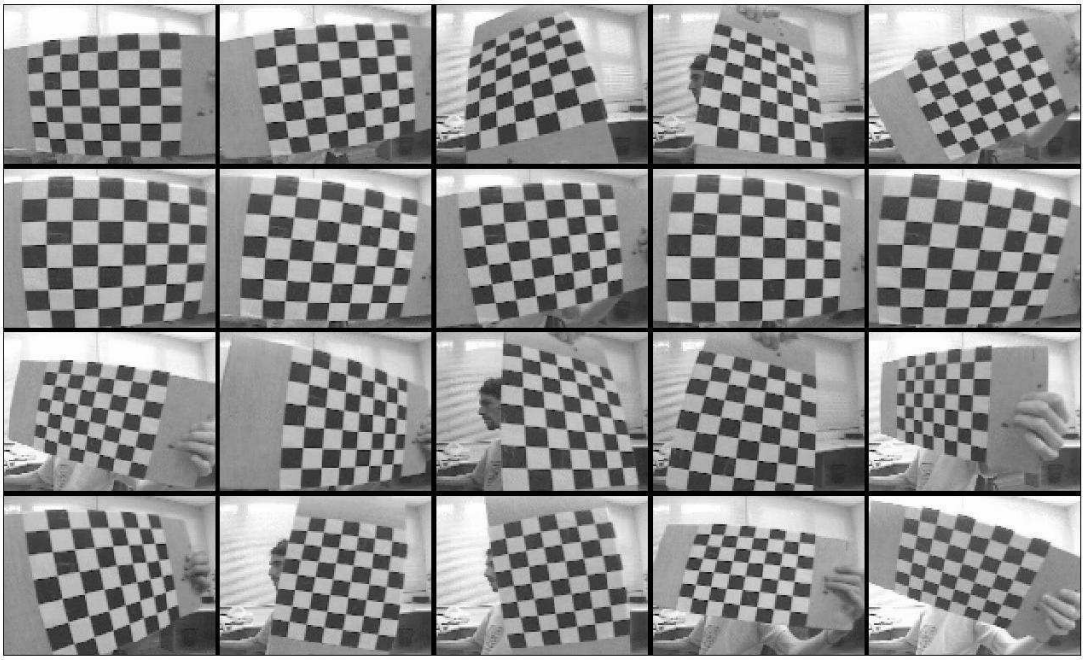


Figura 6.5: Visualização das 20 imagens adquiridas pela câmera monocular do efetuador e a distorção presente em cada.

Parâmetro	Valor (pixels)
$f.k_x$	714,33
$f.k_y$	634,56
c_x	349,85
c_y	275,47

(a) Parâmetros Intrínsecos.

Parâmetro	Valor (pixels)
k_1	-0,44924
k_2	0,28415
P_1	0,00123
P_2	-0,00408
k_3	0,036811

(b) Parâmetros de Distorção

Tabela 6.1: Parâmetros da câmera monocular do braço do manipulador.

Observa-se que a distorção tangencial pode ser desprezada por ser duas vezes menor em ordem de magnitude que a radial [28]. Esses parâmetros obtidos são de importância, pois podem ser usados para a correção da imagem. A Figura 6.6 mostra uma das vinte imagens e a sua correspondente correção, observando a redução da distorção.

Na calibração do Kinect não foi necessária uma calibração manual pois foi observada uma boa detecção das bordas do tabuleiro de xadrez. Utilizou-se a rotina automática de calibração proposta na biblioteca OpenCV [37] e implementada por Burrus [31].

(a) Imagem com distorção. (b) Imagem sem distorção.

Figura 6.6: Correção da distorção na imagem da câmera monocular.

Na câmera RGB do Kinect obtiveram-se os parâmetros intrínsecos e de distorção apresentados na Tabela 6.2. Os parâmetros de distorção e intrínsecos da câmera CMOS do Kinect apresentam-se na Tabela 6.3.

Parâmetro	Valor (pixels)
$f.k_x$	514,10
$f.k_y$	514,37
c_x	325,94
c_y	251,14

(a) Parâmetros Intrínsecos

Parâmetro	Valor (pixels)
k_1	0,241225
k_2	-0,813889
P_1	-0,002733
P_2	-0,000858
k_3	0,861181

(b) Parâmetros de Distorção

Tabela 6.2: Parâmetros intrínsecos e de distorção da câmera RGB do Kinect.

Parâmetro	Valor (pixels)
$f.k_x$	561,68
$f.k_y$	561,02
c_x	317,21
c_y	227,74

(a) Parâmetros Intrínsecos

Parâmetro	Valor (pixels)
k_1	-0,206506
k_2	0,808835
P_1	-0,003617
P_2	-0,000449
k_3	-0,995958

(b) Parâmetros de Distorção

Tabela 6.3: Parâmetros intrínsecos e de distorção da câmera CMOS do Kinect.

A calibração estéreo da a orientação R_{cs} e translação T_{cs} entre as câmeras do Kinect.

$$R_{cs} = \begin{bmatrix} 0,999639 & -0,000077 & 0,026854 \\ 0,001331 & 0,998909 & -0,046689 \\ -0,026821 & 0,046709 & 0,998548 \end{bmatrix}, \tag{6.1}$$

$$T_{cs} = \begin{bmatrix} 0,024658 \\ 0,001728 \\ -0,002218 \end{bmatrix} [m]. \tag{6.2}$$

Observa-se que as câmeras do Kinect estão separadas aproximadamente $2,5\,cm$ e possuem a mesma orientação, pois R_{cs} é praticamente a matriz identidade.

6.1.2 Sistemas de referência

A saída do kinect é representada como uma nuvem de pontos que caracterizam o espaço de trabalho, essa nuvem está especificada em função do sistema de coordenadas do kinect. Na Figura 6.7 observa-se a nuvem de pontos obtida do kinect com sistema de referência o_k. Utilizou-se a biblioteca PCL [29] para visualizar a nuvem de pontos.

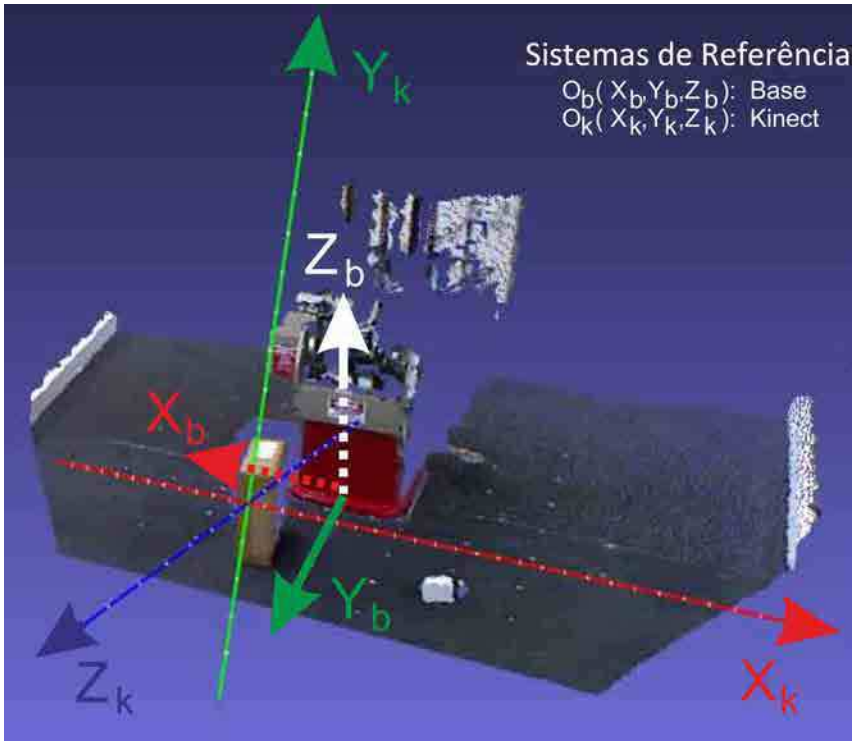


Figura 6.7: Representação do espaço de trabalho através do PCL, indicando os sistemas de referencia da base do manipulador e do sensor RGB-D.

Finalmente, tem-se a necessidade de representar a nuvem de pontos em função do sistema de referência do manipulador o_b colocado na sua base. Ou seja, tem-se que encontrar a matriz homogênea de transformação ${}^{cam}_{bas}H$ que relaciona pontos de um sistema de coordenadas para o outro. Isto é necessário para ser feito posteriormente o roteamento do braço. Feito um alinhamento manual dos sistemas de referência obteve-se a matriz homogênea.

$$ {}^{cam}_{bas}H = \begin{bmatrix} -0,999912 & -0,008565 & -0,010129 & -0,047482 \\ -0,006124 & -0,379225 & 0,925284 & 0,999258 \\ -0,011766 & 0,925265 & 0,379139 & 0,583592 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{6.3} $$

No alinhamento usou-se o aplicativo Meshlab [38] onde é feita uma rotação e translação manual do sistema de referencia o_b até conseguir o alinhamento dos três eixos com o sistema o_k.

6.2 Trajetória

Para avaliar o desempenho do sistema foram feitos dois testes [39]. O primeiro foi realizado em um espaço de trabalho aleatório sem obstáculos e o segundo possuindo um obstáculo. Cada teste foi feito dez vezes para realizar comparações com os dados obtidos. A escolha dos parâmetros foi de $\rho = 0,15$, $\varepsilon = 0,05$ e $s = 50000$ configurações, sabendo que os obstáculos são da ordem de centímetros foi aumentado o valor por falta de ε reduzindo assim o tempo de convergência do algoritmo em caso de existir muitos obstáculos. A escolha do roteamento q_{task} é feita em função da menor quantidade de configurações que são necessários para realizar a tarefa. Não foi aplicada suavização da trajetória em nenhum dos dois testes. Assim executam-se três vezes o algoritmo e a solução final será aquela com menor número de configurações.

6.2.1 Trajetória sem obstáculos

Na Figura 6.8 observa-se a simulação da trajetória no teste que não inclui obstáculos. Esse planejamento final possui três configurações e foi o de menor quantidade de poses encontrado nas três execuções. A cena de trabalho contém o modelo de manipulador feito na Seção 3.5. Foi adicionado a nuvem de pontos do objeto de interesse somente para propósitos de visualização, porém ele não é incluído no modelo da cena de trabalho.

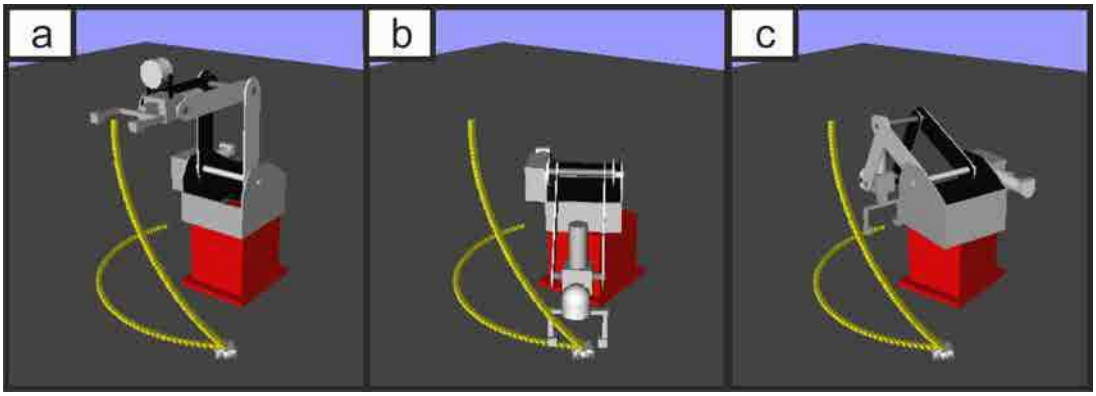


Figura 6.8: Configurações do manipulador para a simulação da trajetória sem obstáculos, indicando a trajetória linear no espaço de juntas.

Na Figura 6.9 apresenta-se a implementação da tarefa no manipulador. Relacionam-se da Figura 6.8.*a*, 6.8.*b* e 6.8.*c* com Figura 6.9.1, 6.9.3 e 6.9.12, respectivamente.

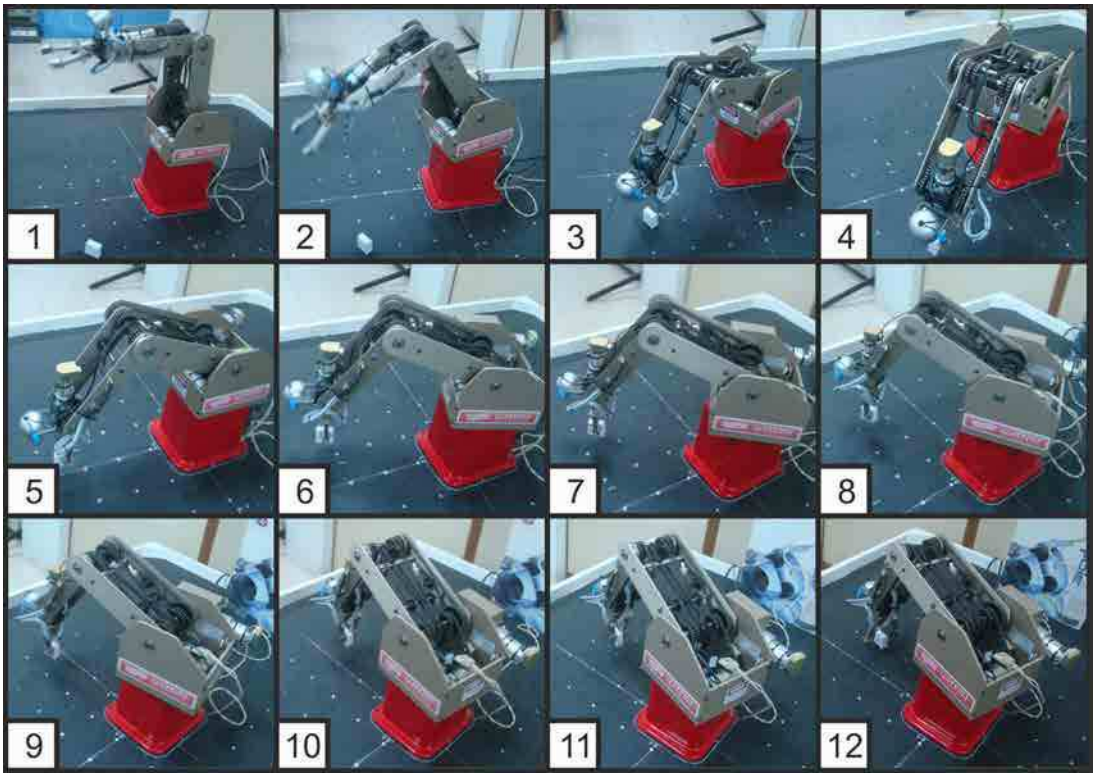


Figura 6.9: Resultados do planejamento de trajetória sem obstáculos, mostrando a translação do objeto até a posição final desejada.

Para o teste sem obstáculos foram registradas W_{task} (matriz de 3×6), q_{task} e q_e (matrizes de 3×5) na Tabela 6.4. A configuração inicial do manipulador q_a definiu-se na configuração q_{home}.

W_a [cm]	$\begin{bmatrix} 0 & 38,1 & 54,61 & 0 & 1 & 0 \end{bmatrix}$
q_a [°]	$\begin{bmatrix} 0,00 & 0,01 & 0,00 & -90,00 & 0,00 \end{bmatrix}$
q_a (norm)	$\begin{bmatrix} 0,5000 & 0,5749 & 0,1489 & 0,5000 & 0,5000 \end{bmatrix}$
q_{ea} (bits)	$\begin{bmatrix} 0 & 23 & -1 & 0 & 0 \end{bmatrix}$
W_b [cm]	$\begin{bmatrix} -20,54 & 29,62 & 2,36 & 0 & 0 & -1 \end{bmatrix}$
q_b [°]	$\begin{bmatrix} -34,77 & -71,49 & -51,00 & -180,00 & 0,00 \end{bmatrix}$
q_b (norm)	$\begin{bmatrix} 0,3913 & 0,1467 & 0,2361 & 0,3500 & 0,5000 \end{bmatrix}$
q_{eb} (bits)	$\begin{bmatrix} -97212 & -199865 & -130904 & -73940 & 73940 \end{bmatrix}$
W_c [cm]	$\begin{bmatrix} 30 & -10 & 2,36 & 0 & 0 & -1 \end{bmatrix}$
q_c [°]	$\begin{bmatrix} 108,43 & -65,18 & -61,59 & -180,00 & 0,00 \end{bmatrix}$
q_c(norm)	$\begin{bmatrix} 0,8389 & 0,1846 & 0,1642 & 0,3907 & 0,5000 \end{bmatrix}$
q_{ec} (bits)	$\begin{bmatrix} 303136 & -182202 & -158090 & -73940 & 73940 \end{bmatrix}$

Tabela 6.4: Configurações do efetuador para a trajetória sem obstáculos.

A configurações do efetuador $q_{task} = \begin{bmatrix} q_a & q_b & q_c \end{bmatrix}^{-1}$ representam os ângulos absolutos de cada junta. $q_e = \begin{bmatrix} q_{ea} & q_{eb} & q_{ec} \end{bmatrix}^{-1}$ são os ângulos em valores de encoders e $W_{task} = \begin{bmatrix} W_a & W_b & W_c \end{bmatrix}^{-1}$ são as poses do efetuador do manipulador no formato apresentado na Equação 3.10[1].

Sabendo que tanto o algoritmo SBL como o roteamento feito no controlador do manipulador são lineares no espaço de configuração, para encontrar a trajetória gerada no espaço tridimensional W_{task} deve-se substituir a trajetória no espaço de juntas da Equação 6.4 na Equação 3.10.

$$q_{task}(t) = \begin{cases} \left(\frac{q_b - q_a}{t_1 - t_0}\right)(t - t_1) + q_b & t_0 \leqslant t \leqslant t_1 \\ \left(\frac{q_c - q_b}{t_2 - t_1}\right)(t - t_2) + q_c & t_1 \leqslant t \leqslant t_2 \end{cases} . \tag{6.4}$$

Como se explicou na Seção 3.4.1 deve-se realizar as modificações das referências (Equação 3.5) e transformar em ângulos relativos (Equação 3.11 e Equação 3.12) antes da substituição.

A malha externa de visão detectou 2 objetos, um deles é o robô e o outro é o objeto de interesse. Ambos foram eliminados do modelo de ambiente como apresentado na bloco "Captura do espaço de trabalho" da Seção 5.1.

[1] Somente a posição está expressada em *cm*, a orientação não possui unidades.

As dimensões do objeto de interesse são de $42 \times 20 \times 40 cm$. O centro de massa detectado foi de$\sigma_e = \begin{pmatrix} -19,98 & 30,13 & 1,85 \end{pmatrix} cm$ em função do sistema de referência da base do manipulador. Observa-se que o efetuador se posiciona sobre o objeto de interesse com um ângulo pré-definido $\theta_5 = 0°$. O sistema de visão da câmera monocular permitiu calcular um ângulo $\theta_5 = 94°$.

6.2.2 Trajetória com obstáculos

Neste segundo teste, definiu-se um obstáculo entre a configuração W_b e W_c que fará o manipulador ter que recalcular uma nova rota livre de colisões. Na Figura 6.7 apresentou-se o espaço de trabalho capturado pelo sensor RGB-D. Observam-se espaços de sombra devido a oclusões dos objetos. Na Figura 6.10 contém a simulação da trajetória onde são observadas a modelagem do obstáculo e do manipulador no espaço de trabalho.

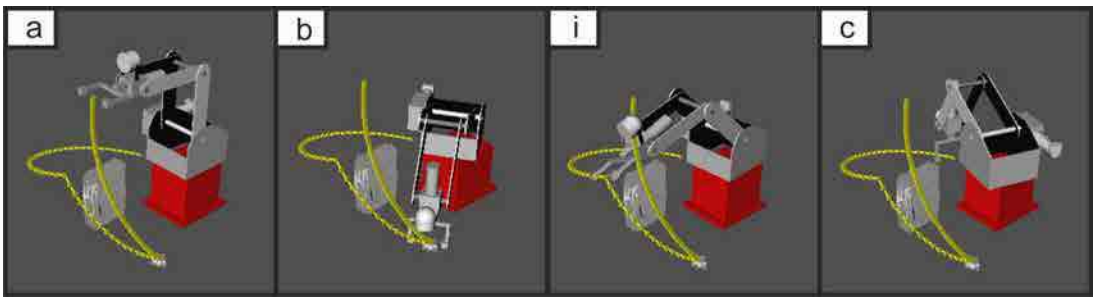


Figura 6.10: Configurações do manipulador para a simulação da trajetória com obstáculos, indicando a trajetória linear no espaço de juntas.

Foi adicionado a nuvem de pontos do objeto de interesse somente para propósitos de visualização, como no teste anterior.

Na Figura 6.11 apresenta-se a implementação no manipulador. Relacionam-se da Figura 6.11.*a*, 6.11.*b*, 6.11.*i* e 6.11.*c* com Figura 6.11.1, 6.11.4, 6.11.6 e 6.11.12, respectivamente.

A poses do efetuador W_{task} foram similares ao teste sem obstáculos com a adição de uma pose intermediária W_i entre W_b e W_c que permite evitar o obstáculo. Na Tabela 6.5 apresenta-se os resultados incluindo a nova configuração W_i.

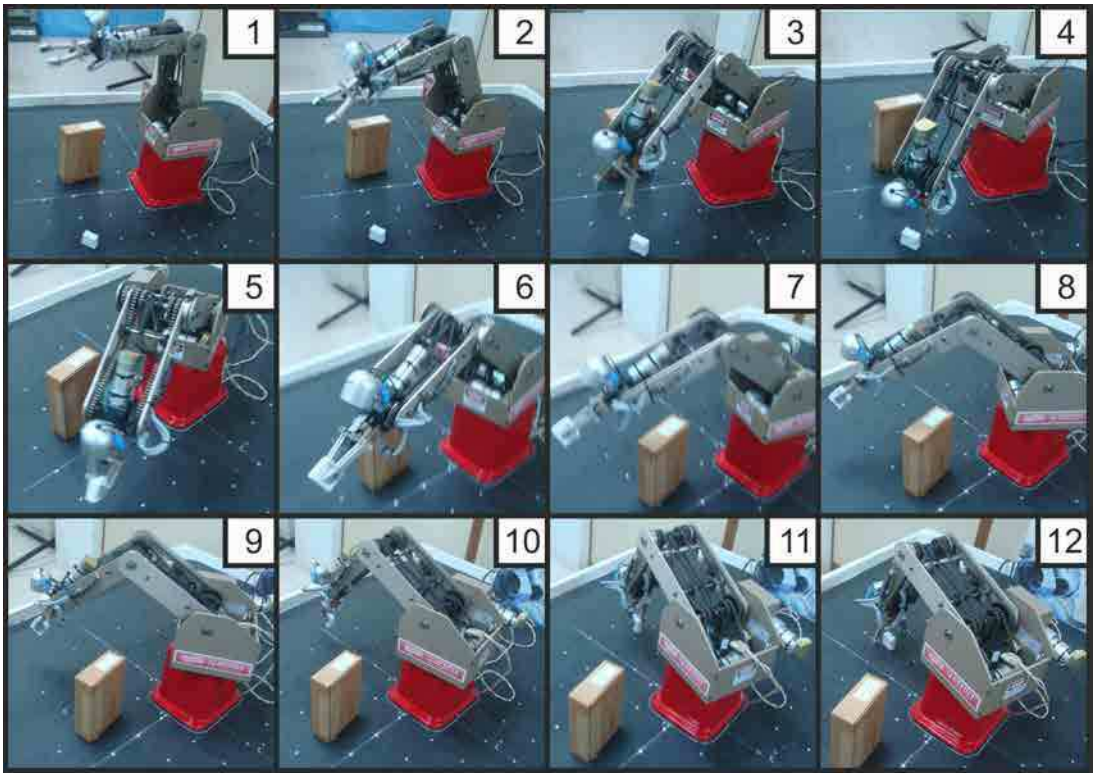


Figura 6.11: Resultados do planejamento de trajetória com obstáculos, mostrando a translação do objeto até a posição final desejada.

No teste $W_{task} = \begin{bmatrix} W_a & W_b & W_i & W_c \end{bmatrix}^{-1}$ é uma matriz de 4×6, $q_{task} = \begin{bmatrix} q_a & q_b & q_i & q_c \end{bmatrix}^{-1}$ e $q_e = \begin{bmatrix} q_{ea} & q_{eb} & q_{ei} & q_{ec} \end{bmatrix}^{-1}$ são matrizes 4×5.

O planejamento da tarefa no espaço de juntas na Equação 6.5:

$$q_{task}(t) = \begin{cases} \left(\frac{q_b - q_a}{t_1 - t_0}\right)(t - t_1) + q_b & t_0 \leqslant t \leqslant t_1 \\ \left(\frac{q_i - q_b}{t_2 - t_1}\right)(t - t_2) + q_i & t_1 \leqslant t \leqslant t_2 \\ \left(\frac{q_c - q_i}{t_3 - t_2}\right)(t - t_3) + q_c & t_2 \leqslant t \leqslant t_3 \end{cases} . \tag{6.5}$$

Tendo em conta as mesmas considerações para transformar $q_{task}(t)$ em $W_{task}(t)$ feitas no teste anterior.

A malha externa de visão detectou 3 objetos, robô, objeto de interesse e obstáculo. A nuvem de pontos do robô e do objeto de interesse foram eliminadas. As dimensões do obstáculo são $53 \times 140 \times 197\,cm$ com centro de massa em $\sigma_e = \begin{pmatrix} -11,10 & 24,32 & 9,83 \end{pmatrix} cm$. O sistema de visão mono permitiu calcular o mesmo ângulo de $\theta_5 = 94°$ do objeto de interesse em função do efetuador.

W_a [cm]	$\begin{bmatrix} 0 & 38.1 & 54.61 & 0 & 1 & 0 \end{bmatrix}$
q_a [°]	$\begin{bmatrix} 0,00 & 0,01 & 0,00 & -90,00 & 0,00 \end{bmatrix}$
q_a (norm)	$\begin{bmatrix} 0,5000 & 0,5749 & 0,1489 & 0,5000 & 0,5000 \end{bmatrix}$
q_{ea} (bits)	$\begin{bmatrix} 0 & 23 & -1 & 0 & 0 \end{bmatrix}$
W_b [cm]	$\begin{bmatrix} -19,98 & 30,13 & 6 & 0 & 0 & -1 \end{bmatrix}$
q_b [°]	$\begin{bmatrix} -33,55 & -71,64 & -50,78 & -180,00 & 0,00 \end{bmatrix}$
q_b (norm)	$\begin{bmatrix} 0,3952 & 0,1458 & 0,2377 & 0,3491 & 0,5000 \end{bmatrix}$
q_{eb} (bits)	$\begin{bmatrix} -93789 & -200285 & -130323 & -73940 & 73940 \end{bmatrix}$
W_i [cm]	$\begin{bmatrix} -6,10 & 53,35 & 37,93 & -0,13 & 1,11 & -0,35 \end{bmatrix}$
q_i [º]	$\begin{bmatrix} -6,52 & -47,27 & -12,15 & -107,57 & 28,96 \end{bmatrix}$
q_i(norm)	$\begin{bmatrix} 0,4796 & 0,2918 & 0,0972 & 0,4324 & 0,5808 \end{bmatrix}$
q_{ei} (bits)	$\begin{bmatrix} -18224 & -132152 & -31192 & 9359 & 38226 \end{bmatrix}$
W_c [cm]	$\begin{bmatrix} 30 & -10 & 6 & 0 & 0 & -1 \end{bmatrix}$
q_c [°]	$\begin{bmatrix} 108,43 & -65,18 & -61,59 & -180,00 & 0,00 \end{bmatrix}$
q_c(norm)	$\begin{bmatrix} 0,8389 & 0,1846 & 0,1642 & 0,3907 & 0,5000 \end{bmatrix}$
q_{ec} (bits)	$\begin{bmatrix} 303136 & -182202 & -158090 & -73940 & 73940 \end{bmatrix}$

Tabela 6.5: Configurações do efetuador para a trajetória com obstáculos.

6.2.3 Tempos de execução das tarefas

Em função de avaliar o tempo de resposta do sistema, na Tabela 6.6 apresenta-se os tempos de execução médios da tarefa para dez execuções da aplicação.

Definindo os seguintes movimentos:

$W_a \rightarrow W_b$: Trajetória entre a pose a e a pose b.

$W_b \rightarrow ROT$: Rotação de θ_5 para que a orientação do objeto e efetuador sejam iguais.

$ROT \rightarrow W_b'$: Trajetória η para descer da posição próxima de carregamento até o centro de massa do objeto de interesse.

$W_b' \rightarrow Carrega$: Movimento de fechamento da garra para captura.

$Carrega \rightarrow W_b'$: Trajetória η para subir novamente na posição próxima de carregamento.

$W_b \rightarrow W_c$: Trajetória entre a pose b e a pose c.

$W_b \rightarrow W_i$: Trajetória entre a pose b e a pose i.

$W_i \rightarrow W_c$: Trajetória entre a pose i e a pose c.

$W_c \rightarrow W_c'$: Trajetória γ da posição próxima de descarregamento até a posição desejada do objeto.

$W_c' \rightarrow Descarrega$: Movimento de abertura da garra.

Trajetória sem obstáculos	Tempo $[s]$	Velocidade	Tempo total $[\%]$
$W_a \rightarrow W_b$	2,11	599,4 mm/s	6,96
$W_b \rightarrow ROT$	14,17	1,02 rad/s	46,75
$ROT \rightarrow W_b'$	1,97	20,3 mm/s	6,50
$W_b' \rightarrow Carrega$	1,94	-	6,40
$Carrega \rightarrow W_b$	2,01	19,9 mm/s	6,63
$W_b \rightarrow W_c$	4,38	599,4 mm/s	14,45
$W_c \rightarrow W_c'$	1,91	20,9 mm/s	6,30
$W_c' \rightarrow Descarrega$	1,82	-	6,01
Total	30,31	-	100

(a) Tempo médio teste sem obstáculos.

Trajetória com obstáculos	Tempo $[s]$	Velocidade	Tempo total $[\%]$
$W_a \rightarrow W_b$	2,20	599,4 mm/s	6,95
$W_b \rightarrow ROT$	15,07	1,04 rad/s	47,63
$ROT \rightarrow W_b'$	1,93	20,7 mm/s	6,10
$W_b' \rightarrow Carrega$	1,83	-	5,78
$Carrega \rightarrow W_b$	2,12	18,9 mm/s	6,70
$W_b \rightarrow W_i$	1,53	599,4 mm/s	4,84
$W_i \rightarrow W_c$	3,22	599,4 mm/s	10,18
$W_c \rightarrow W_c'$	1,97	20,3 mm/s	6,23
$W_c' \rightarrow Descarrega$	1,77	-	5,59
Total	31,64	-	100

(b) Tempo médio teste com obstáculos.

Tabela 6.6: Tempo médio de realização das tarefas para as dez execuções.

Observa-se que a rotação $W_b \rightarrow ROT$ consome quase a metade do tempo de planejamento em ambos planejamentos (46,75% e 47,63%).

Os tempos médios de execução dos blocos estão colocados na Tabela 6.7, onde "Inic_app_ctrl" é o tempo para o controlador começar realizar os movimentos.

Bloco	Sem obstáculos		Com obstáculos	
	tempo [s]	Tempo [%]	tempo [s]	Tempo [%]
Captura do espaço de trabalho	23,38	55,59	25,11	53,66
Gerador do espaço de trabalho	9,46	22,49	11,73	25,07
Tr($\mathbb{R}^6 \rightarrow \mathbb{R}^5$)	0,34	0,81	0,39	0,83
Planejador de trajetória	0,03	0,07	0,36	0,77
Tr1($\mathbb{R}^5 \rightarrow \mathbb{R}^5 e$)	0,61	1,45	0,84	1,80
Tr2($\mathbb{R}^5 \rightarrow \mathbb{R}^5 e$)	1,12	2,66	1,37	2,93
Inic_app_ctrl	6,62	15,74	6,49	13,87
Extração das caract. da imagem	0,13	0,31	0,19	0,41
Estimação da pose do objeto	0,37	0,88	0,31	0,66
Total	42,06	100	46,79	100

Tabela 6.7: Tempo médio de execução do programa para dez execuções.

Na Tabela 6.7 pode ser observado que a captura e a geração do espaço de trabalho da malha de visão externa consomem aproximadamente o 80% do tempo. O planejador de trajetória para o teste consome aproximadamente 1% do tempo, apresentando bom tempo de execução. Finalmente, os blocos de visão da câmera monocular apresentam-se eficientes e no seu conjunto utilizam em média aproximadamente 1% do tempo.

Observando a Tabela 6.6 e 6.7 pode-se concluir que o tempo total de execução para realizar a tarefa então é de $72,37\,s$ para o teste sem obstáculos e de $78,43\,s$ para quando existem obstáculos.

Como a API do controlador é fechada e não permite o envio de configurações pela via de comandos do sistema, as execuções das tarefas demoraram mais tempo do que o necessário. No teste sem obstáculos o movimento $W_b \rightarrow ROT$ poderia ter sido executado em $W_b \rightarrow ROT = 1,61\,s$ e o inicio da aplicação poderia ter demorado Inic_app_ctrl$\sim 1\,s$. Para o caso do teste com obstáculos acontece o mesmo, os tempos poderiam ter sido de $W_b \rightarrow ROT = 1,58\,s$ e Inic_app_ctrl$\sim 1\,s$ reduzindo o tempo total em aproximadamente $18,2\,s$ para o teste sem obstáculos e $19,0\,s$ para o caso com obstáculos (25,15% e 24,23%, respectivamente).

6.2.4 Erros

O erro de posicionamento do efetuador ε depende principalmente do sistema de visão da malha exterior quando realiza o cálculo do centro de massa σ: se a nuvem de pontos não é capturada corretamente pode afetar a performance do sistema.

Também depende da calibração feita dos sistemas de referência e em menor medida (desprezível) dos encoders do manipulador [2]. Calculou-se o erro de posicionamento da garra executando o unicamente o bloco "Captura do espaço de trabalho" que permite achar, entre outras coisas o centro de massa. Sabendo que o objeto de interesse possui o centro de massa em $\sigma_v = \begin{pmatrix} -20 & 30 & 2 \end{pmatrix}$ *cm* obtiveram-se os seguintes resultados executando o bloco dez vezes, onde o error calcula-se como $\varepsilon = \|\sigma_v - \sigma_e\|$, com σ_e a posição estimada do centro de massa.

Centro de massa σ_e $[cm]$	$\varepsilon_p = \|\sigma_v - \sigma_e\|$ $[cm]$
$\begin{pmatrix} -19.97 & 30.14 & 1,83 \end{pmatrix}$	0,2223
$\begin{pmatrix} -19,98 & 30,11 & 1,74 \end{pmatrix}$	0,2830
$\begin{pmatrix} -19,91 & 30,23 & 1,76 \end{pmatrix}$	0,3444
$\begin{pmatrix} -19,94 & 30,24 & 1,86 \end{pmatrix}$	0,2843
$\begin{pmatrix} -19,96 & 30,16 & 1,86 \end{pmatrix}$	0,2163
$\begin{pmatrix} -19,89 & 30,13 & 1,80 \end{pmatrix}$	0,2627
$\begin{pmatrix} -19,93 & 30,18 & 1,84 \end{pmatrix}$	0,2508
$\begin{pmatrix} -19,96 & 30,18 & 1,80 \end{pmatrix}$	0,2720
$\begin{pmatrix} -19,97 & 30,16 & 1,71 \end{pmatrix}$	0,3326
$\begin{pmatrix} -19,91 & 30,20 & 1,72 \end{pmatrix}$	0,3557
Erro médio	0,2823

Tabela 6.8: Distâncias medias ao centro de massa do objeto de interesse e erro médio total para dez execuções.

Ter em conta que o obstáculo está posicionado em uma distância aproximada de $1\,m$ do sensor RGB-D. Erros maiores observam-se quando a distância entre o objeto e o sensor aumentam.

O erro de orientação do efetuador ε_o depende do sistema da camera monocular do manipulador, encontram-se erros de $\varepsilon_o \leq 1^\circ$ que é o tamanho de discretização usado no sistema de visão da malha interna quando são definidas as 180 posições de rotação.

6.3 Considerações finais

Neste capítulo, foram mostrados resultados de testes feitos no sistema de controle aplicando as técnicas descritas em capítulos anteriores. Observam-se os resultados positivos da aplicação de processamento e calibração de câmeras que permite uma redução da distorção facilitando a extração dos parâmetros das características da imagem com menor erro. Observa-se que a calibração dos sistemas de referência é de importância para obter um bom desempenho do sistema como um todo. Foram realizados testes sem obstáculos e com obstáculos apresentando os valores das juntas para realizar a tarefa. Foram definidos os tempos de execução de cada movimento e de cada bloco do sistema de controle concluindo que podem ser reduzidos capturando e gerando o espaço de trabalho de forma mais eficiente. A API fechada do controlador é um fator que limita o desempenho do sistema. Finalmente os erros no sistema de visão foram tratados não influindo na posição da garra em curtas distancias. Portanto, foram apresentadas boas respostas no sistema de controle, onde foi encontada uma trajetória livre de obstáculos sendo levada a cabo satisfatoriamente.

Capítulo 7

Conclusões

Nesta dissertação foram abordadas a integração de um sistema de sensoriamento e a aplicação de um método de planejamento de trajetórias para um robô manipulador de objetos. A integração do sistema para a manipulação inteligente de diversas peças, gerando trajetórias livres de obstáculos foi desenvolvida com sucesso.

O manipulador Pegasus foi apresentado e em seguida desenvolvidas as equações da cinemática da plataforma. Definiu-se a transformação dos ângulos em valores normalizados e valores de encoders que permitem a comunicação com o planejador de trajetorias e controlador do manipulador, respectivamente. Explicou-se o planejador de trajetória SBL definindo os parâmetros mais apropriados para sua execução e discutiu-se a importância de sua implementação em manipuladores robóticos. Desenvolveu-se a modelagem do manipulador para ser representado no algoritmo de roteamento e foi concluído que sobredimensionamento nas peças são necessárias para evitar possíveis colisões.

Foram apresentados o embasamento teórico das técnicas de processamento de imagens analisando a importância de realizar um modelo e calibração das câmeras. Foi apresentado o sensor RGB-D e as técnicas de visão estéreo que permitem a captura da profundidade da cena de trabalho para posterior reconstrução da cena tridimensional. Foram realizadas calibrações dos sensores obtendo melhoras tanto na detecção do objeto e da sua orientação como da reprojeção dos pontos obtidos pelo sensor. Uma redução da distorção das lentes foi observada facilitando a extração dos parâmetros das características da imagem com menor erro. Foi feito a relação de coordenadas do sistema de referência do sensor RGB-D em função de coordenadas da base do manipulador, observou-se que realizar isto de forma errônea acrescenta de sobremaneira o erros de posicionamento da garra. Foi realizada a adquisição da nuvem de pontos do espaço de trabalho, definindo os obstáculos e a conversão deles a malhas de triângulos.

Foram realizados testes sem obstáculos e com obstáculos apresentando os valores das juntas para realizar a tarefa. O transporte de uma peça executando a tarefa definida pelo planejador de trajetória foi realizado com sucesso apresentando bons resultados com diversos obstáculos entre a posição do objeto e o alvo final. Foram definidos os tempos de execução de cada movimento e de cada bloco do sistema de controle, concluindo que podem ser reduzidos capturando e gerando o espaço de trabalho mais eficientemente. Observa-se também que a API fechada do manipulador limita o tempo de resposta da plataforma.

Finalmente, erros no sistema de visão foram tratados não influindo na posição da garra em curtas distancias. Conseguiu-se boa resposta do sistema de controle que encontra uma trajetória livre de obstáculos e ela é levada a cabo satisfatoriamente. A detecção do objeto, robô e obstáculos atingiu-se observando maior precisão quanto mais perto do dispositivo RGB-D. O sistema apresenta-se robusto a mudanças na iluminação que podem degradar a performance do sistema pelo uso de sistemas de visão infravermelhos.

7.1 Sugestões para trabalhos futuros

Na Seção são sugeridas possíveis melhoras tanto no sistema de visão como no planejador de trajetória.

- Automação do alinhamento dos sistemas de referência entre o Kinect e o manipulador como feito por Rakprayoon *et al* [10], isto permite a redução de erros de calibração dos sistemas de referência;
- A colocação de mais dispositivos RGB-D e a substituição da câmera monocular por outro sensor RGB-D são futuras implementações para obter melhores respostas no sistema de visão evitando oclusões de objetos [40];
- Uso da técnica dos momentos aplicada direitamente a nuvem de pontos obtendo melhoras no tempo de execução do roteamento e também pode-se prescindir da câmera monocular do braço;
- Vislumbra-se melhoras no sistema de controle do manipulador, substituindo a sua API e fazendo um controle diretamente nas juntas;
- Desenvolvimento ou implementação de um sistema de planejamento de trajetória que evite obstáculos de forma dinâmica;

- Detecção de objetos de interesse em casos mais gerais além de detectar aqueles que se encontram em uma mesa plana.

Referências Bibliográficas

[1] SPONG, M. W.; HUTCHINSON, S.; VIDYASAGAR, M. *Robot Modeling and Control*. [S.l.]: John Wiley & Sons, Inc., 2005. ISBN 0471649902.

[2] SICILIANO, B. et al. *Robotics: Modelling, Planning and Control*. [S.l.]: Springer, 2009. ISBN 978-1-84628-641-4.

[3] MIKAWA, M. et al. Vision-based redundancy control of robot manipulators for obstacle avoidance. *23rd International Conference on Industrial Electronics, Control and Instrumentation (IECON)*, v. 3, p. 1373 –1378, Novembro 1997.

[4] WONG, A. K. C. et al. A vision based online motion planning of robot manipulators. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, v. 2, p. 940 –948, Novembro 1996.

[5] SILVA, P. M. *Controlo Visual de Robôs Manipuladores Utilizando um Sistema de Visão Estéreo*. Dissertação (Mestrado) — Universidade Técnica de Lisboa, Maio 2008.

[6] DENKER, A.; ADIGüZEL, T. Vision based robotic interception in industrial manipulation tasks. *International Journal of Information and Mathematical Sciences*, p. 296 –302, 2007.

[7] LEUTERT, F.; FREIER, C.; SCHILLING, K. 3d-sensor based dynamic path planning and obstacle avoidance for industrial manipulators. *7th German Conference on Robotics (ROBOTIK)*, p. 1 –6, Maio 2012.

[8] FLANDIN, G.; CHAUMETTE, F.; MARCHAND, E. Eye-in-hand/eye-to-hand cooperation for visual servoing. *IEEE International Conference on Robotics and Automation (ICRA)*, v. 3, p. 2741 –2746, 2000.

[9] GUO, D.; JU, H.; YAO, Y. Research of manipulator motion planning algorithm based on vision. *Sixth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, v. 5, p. 420 –425, Agosto 2009.

[10] RAKPRAYOON, P.; RUCHANURUCKS, M.; COUNDOUL, A. Kinect-based obstacle detection for manipulator. *IEEE/SICE International Symposium on System Integration (SII)*, p. 68 –73, 2011.

[11] WEGHE, M. V.; FERGUSON, D.; SRINIVASA, S. S. Randomized path planning for redundant manipulators without inverse kinematics. *7th IEEE-RAS International Conference on Humanoid Robots*, p. 477 –482, Dezembro 2007.

[12] LAVALLE, S. M.; KUFFNER, J. J. Randomized kinodynamic planning. *IEEE International Conference on Robotics and Automation (ICRA)*, v. 1, p. 473 –479, 1999.

[13] BERTRAM, D. et al. An integrated approach to inverse kinematics and path planning for redundant manipulators. *IEEE International Conference on Robotics and Automation (ICRA)*, p. 1874 –1879, Maio 2006.

[14] WANG, W.; LI, Y. Path planning for redundant manipulator without explicit inverse kinematics solution. *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, p. 1918 –1923, Dezembro 2009.

[15] KUFFNER, J. J.; LAVALLE, S. M. Rrt-connect: An efficient approach to single-query path planning. *IEEE International Conference on Robotics and Automation (ICRA)*, v. 2, p. 995 –1001, 2000.

[16] LIU, H. et al. Hierarchical roadmap based rapid path planning for high-dof mobile manipulators in complex environments. *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, p. 189 –195, 2009.

[17] GUERNANE, R.; BELHOCINE, M. A smoothing strategy for prm paths application to six-axes motoman sv3x manipulator. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, p. 4155 –4160, 2005.

[18] SANCHEZ, G.; LATOMBE, J. C. A single-query bi-directional probabilistic roadmap planner with lazy collision checking. *Int. Symposium on Robotics Research (ISRR)*, Novembro 2001.

[19] ROBOT INSTITUTE OF AMERICA (RIA). Disponível em http://www.robotics.org. Acesso em: 30 de outubro de 2012.

[20] SCHILLING, R. J. *Fundamentals of Robotics: Analysis and Control*. [S.l.]: Prentice - Hall, Inc., 1990. ISBN 0-13-344433-3.

[21] SPONG, M. W.; VIDYASAGAR, M. *Robot Dynamics and Control*. [S.l.]: John Wiley & Sons, Inc., 1989. ISBN 0-471-61243-X.

[22] CRAIG, J. J. *Introduction to Robotics*. [S.l.]: Prentice - Hall, Inc., 2006. ISBN 0201543613.

[23] VALENZUELA, W. A. V. *Sistema de articulação inteligente por meio da liga com memória de forma*. Dissertação (Mestrado) — Universidade Federal de Campina Grande, 2005.

[24] VALENZUELA, W. A. V. *Robô Bípede Acionado Com Ligas de Memória de Forma*. Tese (Doutorado) — Universidade Federal de Campina Grande, 2011.

[25] PEGASUS 880-RA2-1-B. Disponível em: http://www.amatrol.com. Acesso em: 04 de novembro de 2012.

[26] AC3D. Disponível em: http://www.inivis.com/. Acesso em: 23 de novembro de 2013.

[27] GONZALEZ, R. C.; WOODS, R. E. *Digital Image Processing*. 2.. ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2001. ISBN 0201180758.

[28] BRADSKI, G.; KAEHLER, A. *Learning OpenCV: Computer Vision with the OpenCV Library*. 1.. ed. [S.l.]: O'Reilly Media, Inc., 2008. ISBN 978-0-596-51613-0.

[29] RUSU, R. B.; COUSINS, S. 3d is here: Point cloud library (pcl). *IEEE International Conference on Robotics and Automation (ICRA)*, Maio 9-13 2011.

[30] HUTCHINSON, S.; HAGER, G. D.; CORKE, P. I. A tutorial on visual servo control. *IEEE Transactions on Robotics and Automation*, v. 12, n. 5, p. 651 –670, Outubro 1996.

[31] BURRUS, N. *RGBDemo*. Disponível em: http://labs.manctl.com/. Acesso em: 01 de maio de 2013.

[32] UNIVERSITY OF WESTERN AUSTRALIA - COMPUTER SCIENCE. Disponível em: http://www.csse.uwa.edu.au/ ajmal/code.html. Acesso em: 23 de abril de 2013.

[33] SCHWARZER, F.; LATOMBE, J. C.; SAHA, M. *Motion Planning Kit (MPK)*. Disponível em: http://robotics.stanford.edu/ mitul/mpk/. Acesso em: 30 de setembro de 2012.

[34] AUTOHOTKEY: Automation, Hotkeys. Scripting. Disponível em: http://www.autohotkey.com/. Acesso em: 07 de março de 2013.

[35] OPEN SOURCE COMPUTER VISION LIBRARY (OPENCV). Disponível em: http://www.opencv.org/. Acesso em: 10 de junho de 2012.

[36] BOUGUET, J. Y. *Camera Calibration Toolbox for Matlab*. Disponível em: http://www.vision.caltech.edu/bouguetj. Acesso em: 05 de novembro de 2012.

[37] OPEN SOURCE COMPUTER VISION LIBRARY (OPENCV). *Camera Calibration and 3D Reconstruction*. Disponível em: http://opencv.willowgarage.com/documentation/cpp/. Acesso em: 17 de julho de 2012.

[38] VISUAL COMPUTING LAB ISTI - CNR. *MeshLab*. Disponível em: http://meshlab.sourceforge.net/. Acceso em: 30 de abril de 2013.

[39] MENDIBURU, F. J.; MORAIS, M. R. A.; LIMA, A. M. N. Visual feedback trajectory planning for object handling and obstacle avoidance. *39th Annual Conference of the IEEE Industrial Electronics Society (IECON)*, novembro 2013.

[40] MENDIBURU, F. J.; MORAIS, M. R. A.; LIMA, A. M. N. Obstacle avoidance for a robot manipulator based on visual feedback. *Simpósio Brasileiro de Automação Inteligente (SBAI)*, outubro 2013.