UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA MECÂNICA

Desenvolvimento e Implementação de um Programa Computacional para a Supervisão e Controle de Manipuladores Robóticos

Autor: **Cláudio Eduardo Aravéchia de Sá**
Orientador: **João Maurício Rosário**

85/00

UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA MECÂNICA
DEPARTAMENTO DE PROJETO MECÂNICO

Desenvolvimento e Implementação de um Programa Computacional para a Supervisão e Controle de Manipuladores Robóticos

Autor: **Cláudio Eduardo Aravéchia de Sá**
Orientador: **João Maurício Rosário**

Curso: Engenharia Mecânica
Área de Concentração: Mecânica dos Sólidos

Tese de doutorado apresentada à comissão de Pós Graduação da Faculdade de Engenharia Mecânica, como requisito para a obtenção do título de Doutor em Engenharia Mecânica.

Campinas, 2000
S.P. – Brasil

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA - BAE - UNICAMP

Sa11d

Sá, Cláudio Eduardo Aravéchia de
Desenvolvimento e implementação de um programa computacional para a supervisão e controle de manipuladores robóticos / Cláudio Eduardo Aravéchia de Sá.--Campinas, SP: [s.n.], 2000.

Orientador: João Maurício Rosário.
Tese (doutorado) - Universidade Estadual de Campinas, Faculdade de Engenharia Mecânica.

1. Robôs industriais. 2. Manipuladores (Mecanismo). 3. Robôs – Sistemas de controle. 4. Automação. I. Rosário, João Maurício. II.. Universidade Estadual de Campinas. Faculdade de Engenharia Mecânica. III. Título.

UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA MECÂNICA
DEPARTAMENTO DE PROJETO MECÂNICO

TESE DE DOUTORADO

Desenvolvimento e Implementação de um Programa Computacional para Supervisão e Controle de Manipuladores Robóticos

Autor : **Cláudio Eduardo Aravéchia de Sá**
Orientador: **Prof. Dr. João Maurício Rosário**

Prof. Dr. João Maurício Rosário, Presidente
Faculdade de Engenharia Mecânica - UNICAMP

Prof. Dr. José Manoel Balthazar
Faculdade de Engenharia Mecânica - UNICAMP

Prof. Dr. Hans Ingo Weber
Faculdade de Engenharia Mecânica - UNICAMP

Prof. Dr. Luis Gonzaga Trabasso
ITA

Prof. Dr. Marcio Rillo
USP

Campinas, 27 de Outubro de 2000.

Dedicatória

Aos meus pais Eduardo e Ana,
e meus irmãos Rodrigo e Carla.

Agradecimentos

Este trabalho de pesquisa foi elaborado dentro do projeto de cooperação internacioanl Brasil – Alemanha (UWT – 10) na área de Robótica Submarina, envolvendo o GKSS - Instituto Tecnológico de Geestacht (Alemanha), CENPES – PETROBRAS e UNICAMP (Brasil), envolvendo a ajuda de diversas pessoas às quais presto homenagem:

- Aos meus pais e irmãos pelo incentivo em todos os momentos da minha vida.

- Ao Professor João Maurício Rosário pela orientação, incentivo e confiança.

- Ao Professor Hans Ingo Weber, diretor Superintendente de Pesquisa do Centro de Tecnologia da UNICAMP, no período de realização desse projeto de pesquisa envolvendo o Brasil e a Alemanha.

- A todos os pesquisadores da GKSS, CENPES – PETROBRAS e UNICAMP envolvidos no projeto UWT – 10.

- A todos os meus amigos por proporcionarem inúmeros e indispensáveis momentos de descontração e colaboraram no desenvolvimento dessa tese.

- E finalmente a todos que, direta ou indiretamente, contribuíram para a conclusão deste projeto.

Resumo

SÁ, Cláudio Eduardo Aravéchia de, *Desenvolvimento e Implementação de um Programa Computacional para a Supervisão e Controle Manipuladores Robóticos*, Campinas, Faculdade de Engenharia Mecânica, Universidade Estadual de Campinas, 2000. 114 p. Tese Doutorado.

O objetivo deste trabalho é o desenvolvimento de um sistema automático de geração de trajetórias e a implementação de um programa computacional off-line e de um programa de supervisão e controle, com o objetivo de viabilizar a automação de tarefas de robôs. Foi utilizado para o desenvolvimento deste trabalho o robô Manutec fabricado pela Siemens composto de seis juntas rotacionais e o manipulador submarino Kraft mas isto não impede que os resultados obtidos possam ser estendido para robôs com outros tipos de juntas.

Inicialmente é realizada uma discussão sobre a programação "off-line" e sobre a supervisão e controle de dispositivos robóticos. Também é realizada uma breve revisão dos conceitos utilizados na robótica, sobre geração de trajetórias e a formulação matemática para a obtenção do modelo geométrico direto e inverso de um robô e em seguida é apresentado a estrutura do sistema proposto para a geração de trajetórias. E após isso se discorre sobre o tratamento de colisões e criação de obstáculos e a implementação do programa computacional off-line e os resultados obtidos (computacionais e experimentais)

Os programas foram implementados, em linguagem ADA, em um microcomputador compatível com a linha IBM-PC-AT de forma modular, possibilitando alterações.

Palavras Chave

- Modelagem, Geração de Trajetórias, Programação Off-Line, Colisão

Abstract

SÁ, Cláudio Eduardo Aravéchia de, *Development and Implementation of a Program Computacional for the Supervision of Industrial Manipulators*, Campinas, Faculdade de Engenharia Mecânica, Universidade Estadual de Campinas, 2000. 114 p. Tese Doutorado.

The objective of this work is the development of an automatic system of generation of trajectories and the implementation of a program off-line and a program of supervision and control, with the objective of making possible the automation of robots' tasks. The robot used for the development of this work was Manutec manufactured by Siemens composed of six rotational joints and the manipulator submarine Kraft but this doesn't impede that the obtained results can be extended for robots with other types of joints.

Initially a brief revision of the concepts used in the robotics is accomplished, about generation of trajectories and the mathematical formulation for the obtaining of the a robot's direct and inverse geometric model and soon after the structure of the system proposed for the generation of trajectories is presented.

It is then it is made a study on the treatment of collisions and creation of obstacles and the implementation of the program off-line and the obtained results. The programs are implemented, in language ADA, in a compatible microcomputer with the line IBM-PC-ATTN in way to modulate, facilitating alterations.

Key Words

- Modeling, Generation of Trajectories, Off-line Programming, Collision

Índice

Lista de figuras

Lista de tabelas

Capítulo 1

Introdução

O programa de doutoramento descrito nesse plano de pesquisa faz parte das atividades desenvolvidas junto ao CENPES-PETROBRÁS e Instituto Tecnológicas de Geestacht, Alemanha, dentro do programa de pesquisas cujo objetivo principal foi a utilização de robôs industriais marinizados para intervenções submarinas.

Durante a primeira fase desse projeto de pesquisa foram realizados estudos considerando a utilização do robô industrial Manutec R3, tendo sido construídos na UNICAMP e no GKSS um ambiente submarino típico como maquete para testes, uma base móvel sem mecanismo de propulsão e ferramentas específicas para a validação dos resultados obtidos. Para permitir a geração automática de trajetórias (programação "off-line") para o robô Manutec, foi desenvolvido um pacote computacional, e um programa de supervisão viabilizou a integração do robô e base móvel nesse ambiente típico (utilizando sensores e atuadores para pagamento de umbilical).

Em uma segunda fase desse projeto de pesquisa (Niemann, 1995), iniciado em 1994, no âmbito de convênio de cooperação técnico-científica firmado entre a PETROBRAS e a UNICAMP, envolveu a utilização de um manipulador hidráulico submarino Kraft (Anexo I).

Neste projeto foi investigada a utilização dos manipuladores para a execução de tarefas automatizadas (em ambientes submarinos), a partir do desenvolvimento de sistema de supervisão

e controle (CMK) para o manipulador e da extensão da programação "off-line" implementada para o robô industrial.

Dentro deste escopo, este trabalho, dividido em nove capítulos, visa o desenvolvimento de um algoritmo de geração de trajetórias, a implementação do programa computacional "off-line" e a implementação de um programa computacional para a Supervisão e Controle de manipuladores robóticos.

Serão abordados os métodos utilizados durante o projeto de pesquisa em Automação Submarina, para a implementação do programa "off-line" (para a criação de obstáculos, teste de colisão, construção gráfica do robô e do ambiente de trabalho).

Serão apresentados os resultados obtidos pelo programa "off-line", também serão apresentados os resultados experimentais obtidos para a validação de tarefas submarinas por robôs industriais.

Os dispositivos robóticos utilizados para o desenvolvimento deste trabalho foram o robô Manutec fabricado pela Siemens composto de seis juntas rotacionais acionadas eletricamente e o manipulador submarino Kraft (também com seis juntas rotacionais com acionamento hidráulico) para o qual os módulos também serão implementados e apresentados os resultados obtidos mas, podem ser utilizados outros tipos de dispositivos robóticos.

Os módulos, da programação "off-line", foram implementados em linguagem ADA (Watt, 1987) que apresenta alto grau de estruturação, o que permite simplificações na programação de tarefas com alto grau de complexidade.

No capítulo 2 é apresentada uma breve discussão sobre a programação "off-line".

No capítulo 3 é apresentado o pacote de supervisão CMK desenvolvido anteriormente.

No capítulo 4 faz-se uma discussão sobre modelagem, controle e geração de trajetórias de robôs também se discorre sobre o problema geométrico direto e inverso de um robô, as formas de obtê-los e sobre os métodos para a inversão do modelo geométrico.

No capítulo 5 é apresentado o algoritmo de geração de trajetórias utilizando o modelo cinemático inverso, um algoritmo para a interpolação e filtragem de trajetórias e os resultados iniciais obtidos.

No capítulo 6 discorre-se sobre o tratamento de colisões, criação gráfica do robô e de obstáculos.

No capítulo 7 são apresentados os módulos implementados a linguagem utilizada para a implementação dos módulos.

No capítulo 8 são apresentadas as simulações realizadas para mostrar os resultados e a iteratividade entre os módulos implementados (programação "off-line").

No capítulo 9 serão apresentados os resultados experimentais obtidos através do uso do programa de supervisão e do programa "off-line".

Finalmente, no capítulo 10 apresentam-se as conclusões finais e perspectivas futuras.

Capítulo 2

Programação "Off-line" de Robôs

O desenvolvimento de robôs industriais tem grande importância pela complexidade envolvida no desenvolvimento de habilidade de emular o comportamento da cadeia de elos do braço do manipulador de forma a reproduzir os movimentos do braço humano. Entre as operações industriais mais freqüentes podemos citar exemplos: soldagem, pintura, "pick-and-place", montagem, etc.

Mais recentemente, com o avanço do reconhecimento de imagens, sensoriamento (visão, distância, esforços, etc) e inteligência artificial, expandiu-se o número de aplicações nas áreas em que uma realimentação dos sensores se faz necessária

Os robôs vêm sendo amplamente adotados na indústria e na manufatura, onde a maioria das aplicações está orientada a indústria automobilística, que incorporou os benefícios da automação flexível em larga escala nos processos de manufatura automotiva automatizada. Devemos destacar ainda a utilização cada vez mais freqüente de programas de simulação "off-line", como os simuladores WORKSPACE (http://www.rosl.com), AUTOMOD e CATIA (Dassaut) que utilizam técnicas de CAD direcionadas a robótica, buscando soluções que facilitem o trabalho de operadores.

Entretanto estes simuladores são fortemente dependentes dos modelos, tornando-se inviáveis de serem utilizados em aplicações especiais tais como a realização de tarefas em ambiente de trabalho desconhecido.

Apesar das dificuldades, a simulação veio para prover um ambiente de simulação gráfica capaz de gerar uma interação suave com as diversas linguagens comerciais de programação de robôs existentes no mercado. Alguns benefícios, tais como a detecção de colisões "off-line" e capacidade de avaliar e otimizar seqüências de programas sem a necessidade da presença física do manipulador, vêm impulsionando o desenvolvimento das pesquisas direcionadas a programação "off-line" de robôs.

Atualmente existem no mercado diversos programas computacionais de simulação e geração de tarefas "off-line" de robôs industriais. Entretanto a maioria desses pacotes necessita de grande esforço computacional, sendo extremamente dedicados a aplicações industriais direcionadas a área de manufatura, onde o acesso do homem é permitido. Tendo em vista, que normalmente são conhecidos valores aproximados do dispositivo robótico e seu ambiente de atuação (posicionamento, parâmetros geométricos, dimensões de ferramentas), geralmente, esses programas permitem apenas a modelagem do ambiente de atuação do robô, necessitando de um refinamento de trajetórias através da aprendizagem on-line.

A utilização de robôs em ambientes de difícil acesso ao homem tem se tornado cada vez mais objeto de interesse dos pesquisadores, exigindo na maioria das aplicações a utilização de programação "off-line".

Tendo em vista as dificuldades de acesso ao ambiente de atuação do dispositivo robótico, essas aplicações exigem a utilização da programação "off-line", onde a etapa de aprendizagem é realizada através de um console de programação, exigindo o conhecimento preciso do modelo do robô e seu ambiente de atuação, incluindo acessórios e ferramentas dedicadas.

Entre as diferentes aplicações de programação "off-line" podemos citar a NASA que vem desenvolvendo telemanipuladores e robôs, a PETROBRAS que vem realizando estudos na área

de automação com o objetivo de automatizar tarefas através de dispositivos robóticos e HISPANO-SUIZA que utiliza dispositivos robóticos em reatores nucleares. Neste capítulo apresentaremos algumas dessas soluções.

2.1 Programação "Off-line" de Robôs

Na programação "off-line", a definição da tarefa (composta de uma trajetória - conjunto de pontos - ou mais) deve ser pré-planejada a partir do conhecimento das operações a serem realizadas, do modelo geométrico (do robô e do meio ambiente) da instalação ou equipamento que será alvo da intervenção. A utilização de um micro-computador para supervisão e gerenciamento destas operações será uma ferramenta poderosa para visualização destas tarefas.

Facilidades de edição gráfica permitem a construção dos modelos, e recursos adicionais permitem realizar simulação gráfica e detecção de colisões. Ao final, a trajetória pretendida é gerada, podendo então ser enviada através de interface I/O do computador para a interface de acionamento e controle do dispositivo robótico, executando assim uma tarefa pré-planejada.

Alem disso a programação "off-line" permite:

- redução de tempo de programação;
- otimização do processo a ser automatizado: simplificação na programação de tarefas com alto grau de complexidade e segurança de realização;
- capacidade de integração de outros dispositivos automatizados.

Um pacote de programação "off-line" normalmente inclui:

i. Um programa para a geração de arquivos de trajetórias;

ii. Um programa para a criação de ambientes e ferramentas;

iii. Um programa para a simulação e visualização do robô realizando a tarefa em seu ambiente de trabalho, figura 2.1.

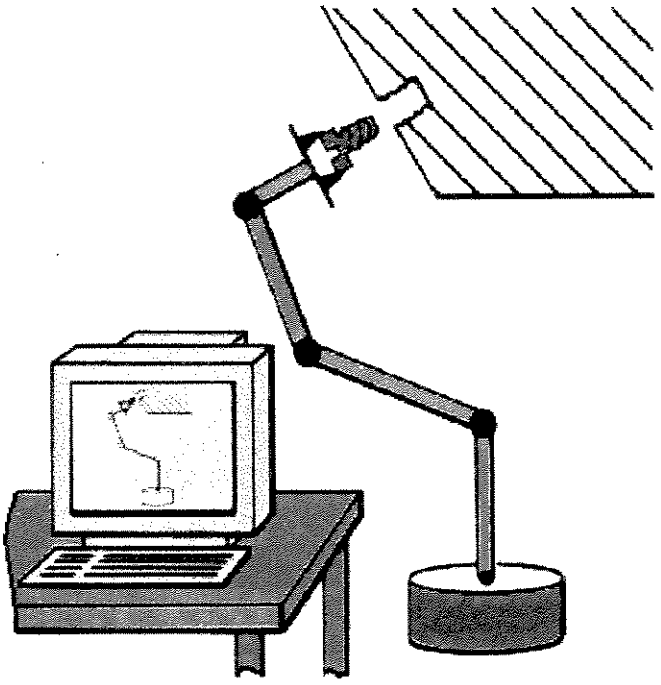
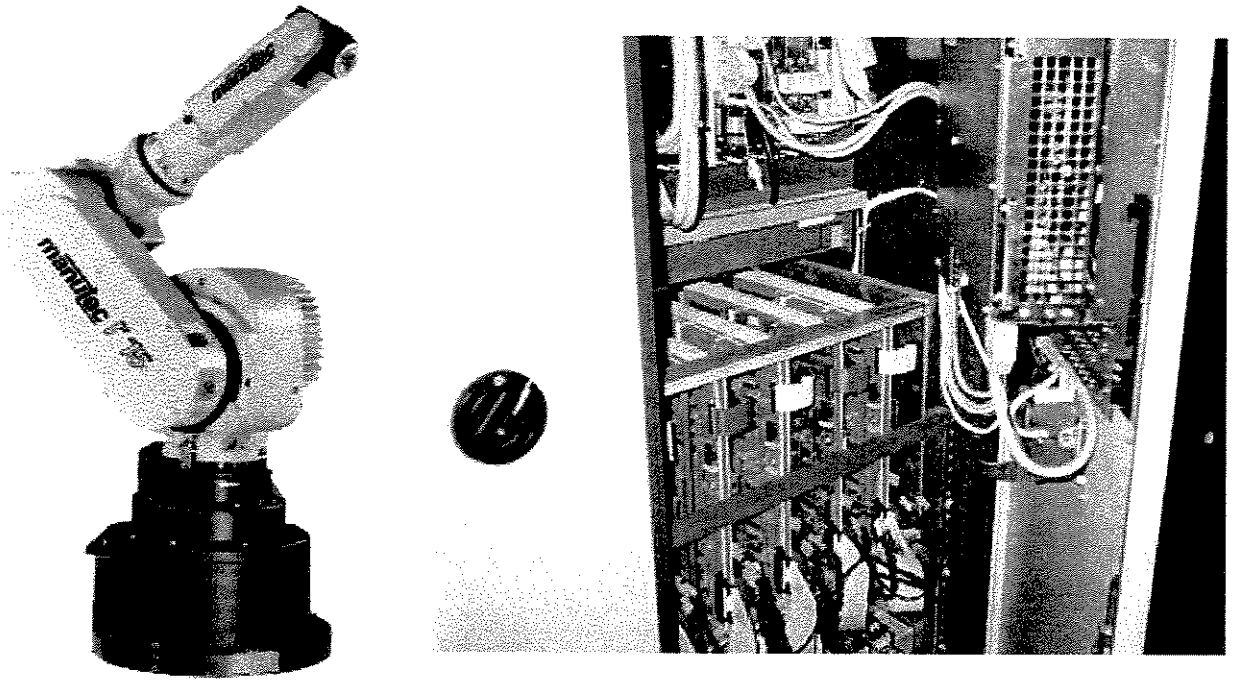


Figura 2.1: Programação "off-line", composto por interface gráfica permite que o robô seja programado sem a necessidade de entrar em contado físico com o mesmo.

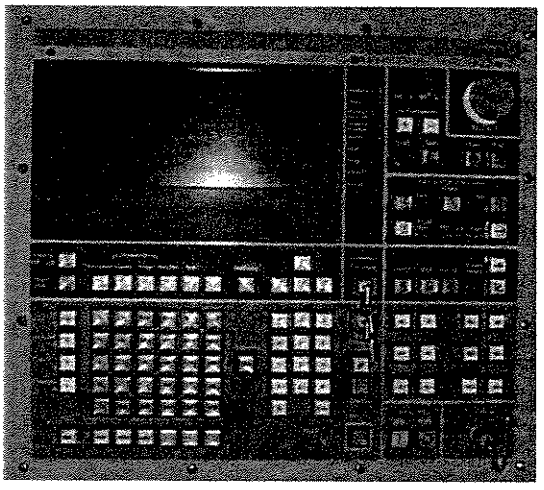
2.2 Teleoperação remota – Sistema Master-Slave – "Mouse" espacial

Num robô industrial o processo de aprendizagem de pontos de uma trajetória é realizado normalmente a partir da movimentação de cada grau de liberdade através de teclas dedicadas do painel de comando, que permite a gravação de pontos de uma determinada trajetória. A interpolação de pontos de passagem (necessários para a realização de uma trajetória) obtidos a partir das juntas.

A figura 2.2 mostra o painel de comando e o robô Manutec.



(a) (b)



(c)

Figura 2.2: (a) robô Manutec, (b) interior do painel de comando e (c) teclas dedicadas.

Uma ferramenta muito utilizada em teleoperação remota, constitui na utilização de um dispositivo "Master-Slave". Este dispositivo é constituído por um simulador físico, que tem a geometria e os sensores idênticos ao robô original ("Master") que permite a aprendizagem de trajetórias a partir do movimento das diferentes juntas do robô original ("slave") permitindo a gravação de pontos intermediários de uma trajetória. A figura 2.3 mostra o sistema "master-slave" (manipulador submarino Kraft).

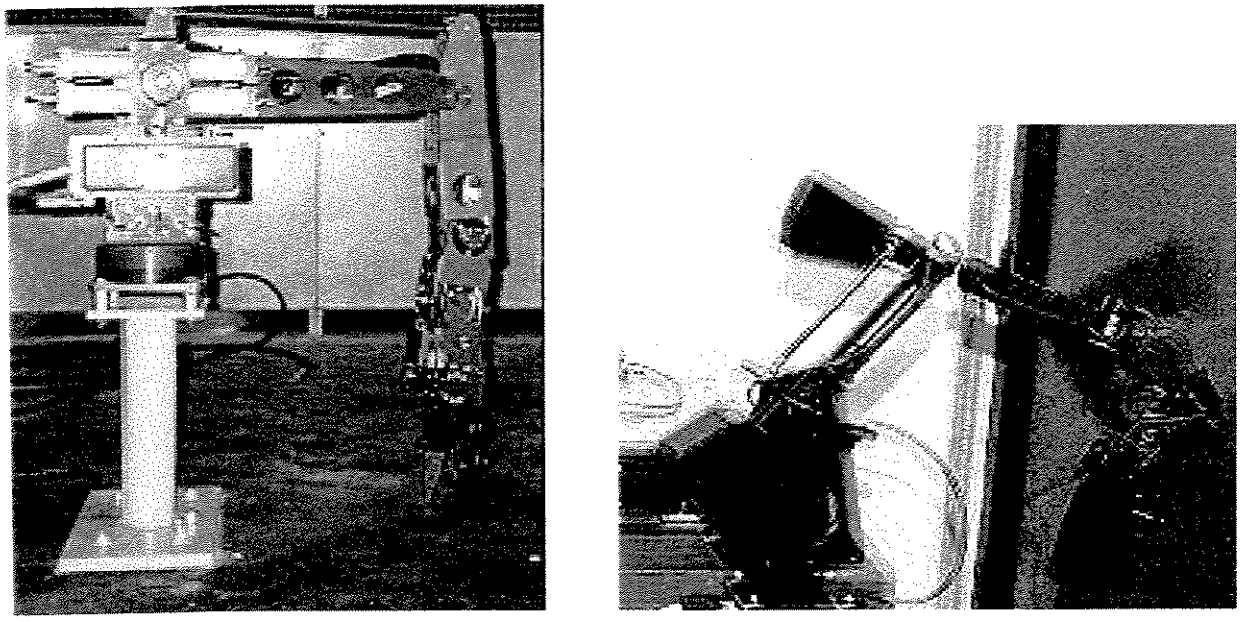


Figura 2.3: Sistema "Maste-Slave", manipulador submarino Kraft.

Um dispositivo, também utilizado em operações remotas, denominado "mouse" espacial figura 2.4, constituí em um sistema que gera sinais de deslocamentos (em coordenadas cartesianas no espaço tridimensional) que são enviadas a um microcomputador a fim de controlar a orientação do manipulador. Este sistema foi construído na Unicamp (Nogueira, 1995).

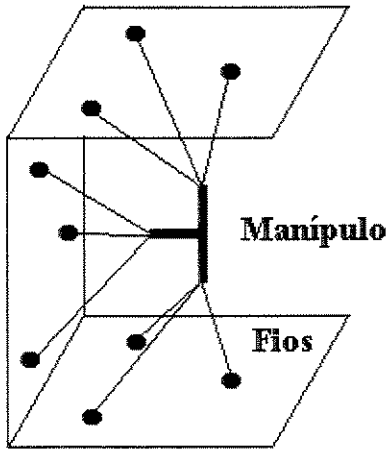


Figura 2.4: Esquema do "mouse" espacial.

Se uma tarefa tiver que ser realizada apenas por software, é necessário que o modelo do dispositivo robótico e do meio ambiente de trabalho seja bem conhecido.

2.3 Exemplos da utilização da programação "off-line"

Como mencionado, a NASA (http://nssdc.gsfc.nasa.gov/planetary/mesur.html) desenvolveu um telemanipulador com a finalidade específica de auxiliar no posicionamento de dispositivos espaciais em órbitas (satélites, antenas, etc) eliminando deste modo a necessidade de que homens arrisquem suas vidas. Também, desenvolveram robôs com a finalidade de explorar solos de outros planetas.

Também a necessidade de prospecção de petróleo em águas profundas, como mencionado anteriormente, também levou a estudos na área de automação com o objetivo de automatizar tarefas (manutenção, inspeção e reparos) através de dispositivos robóticos. Uma outra aplicação freqüente é a utilização destes dispositivos para a inspeção e manutenção do interior de reatores nucleares. A seguir estas aplicações serão descritas de forma mais detalhada.

2.3.1 Área de robótica submarina - Intervenções automatizada em águas profundas

As restrições ao mergulho humano - inviável além de 300 metros de lâmina d'água e, mesmo quando possível, sempre de alto risco - o elevado custo operacional e as dificuldades de acesso inerentes às intervenções submarinas (em pressão equivalente a 1000 metros de lâmina d'água) vêm obrigando o uso crescente de ferramentas dedicadas e dispositivos robóticos em substituição ao homem.

A figura 2.5 mostra um esquema da utilização de um robô (Manutec) e de um dispositivo dedicado (veiculo de operação remota, ROV) em intervenções submarinas.

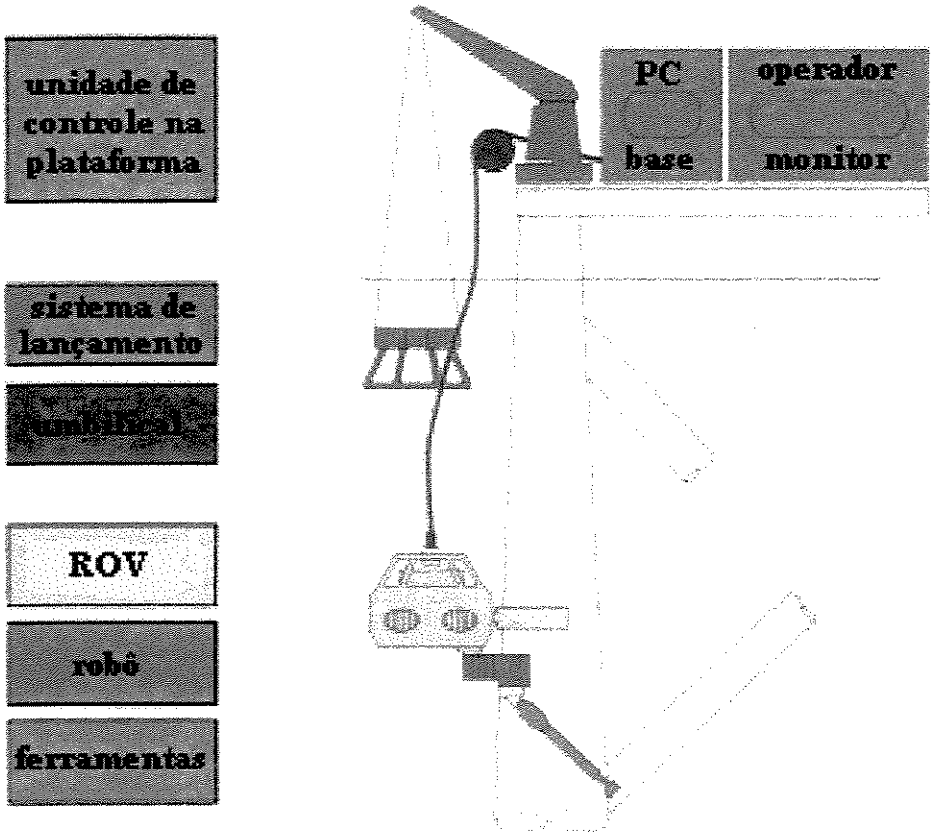


Figura 2.5: Esquema da utilização de um robô (Manutec) e de um dispositivo dedicado (ROV) em intervenções submarinas.

A figura 2.6 mostra o veículo de operação remota (ROV) em uma intervenção na qual é realizada uma solda em uma estrutura.

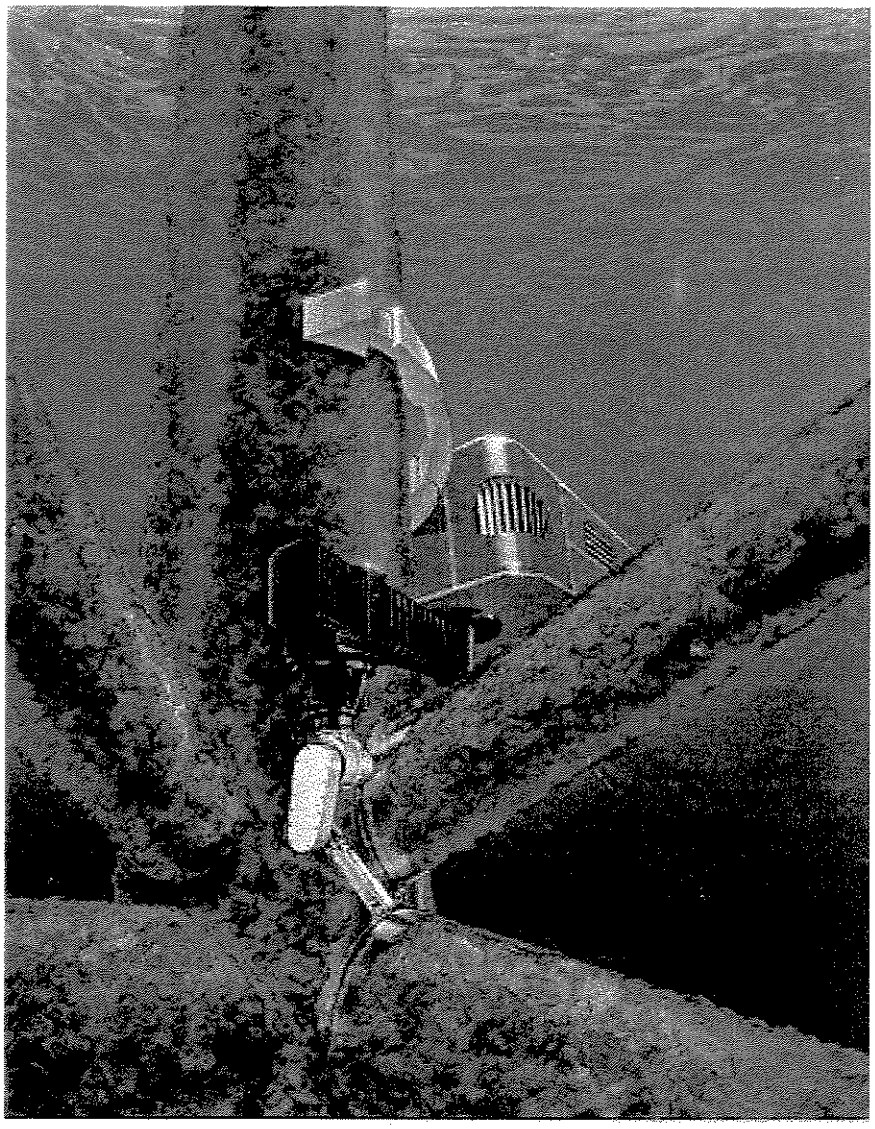


Figura 2.6: Intervenção submarina utilizando robô e ROV.

2.3.2 Área de robótica nuclear -Intervenção automatizada em geradores de vapor de usinas nucleares (manutenção e inspeção automatizada)

Desde a criação das primeiras centrais nucleares de eletricidade na França, houve a necessidade da criação de procedimentos de modo a assegurar operações automatizadas de manutenção e reparo de instalações em meios hostis.

O desenvolvimento do manipulador ISIS, figura 2.7, foi baseado em três modos de operação automatizada:

- modo Master-Slave: Manipulador controlado remotamente utilizando um sistema master-slave (mestre-escravo);

- modo Automático: Manipulador é controlado através de um computador PC a partir de trajetórias geradas de modo "off-line" e/ou master-slave;

- modo Automático com sensoriamento: Manipulador é controlado através de um PC a partir de trajetórias geradas anteriormente, e o seu posicionamento final é realizado utilizando sensores de proximidade e força/torque.

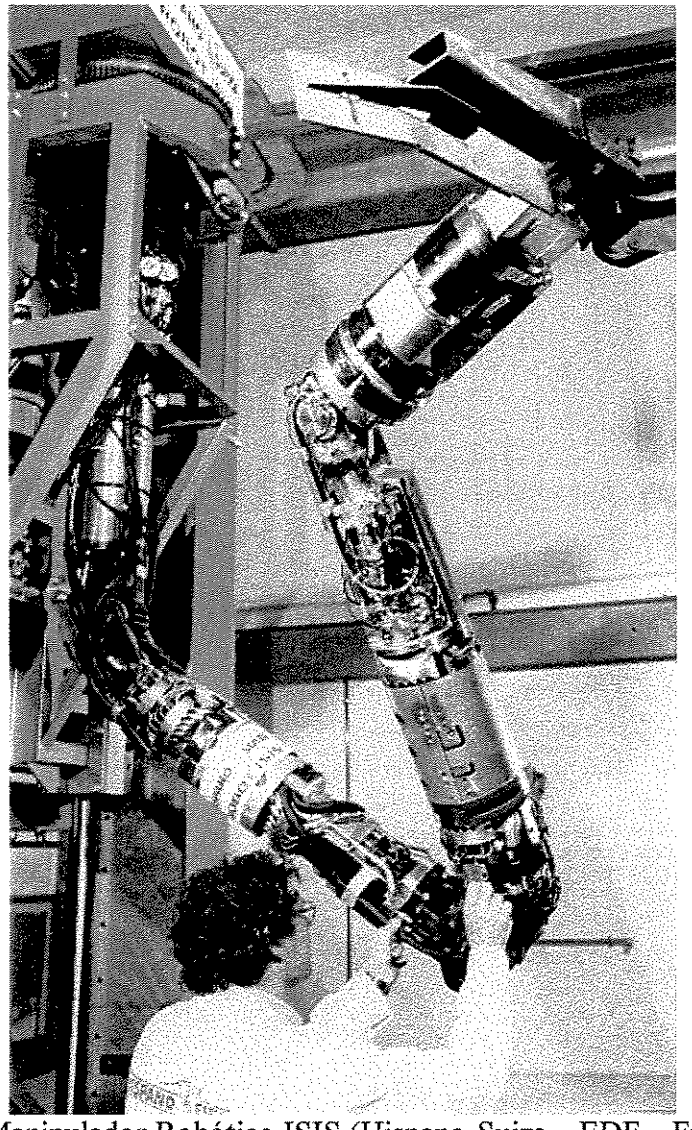


Figura 2.7: Manipulador Robótico ISIS (Hispano-Suiza – EDF – França, 1987)

2.3.3 Área de robótica móvel - Exploração e reconhecimento do planeta Marte (projeto Mars-PathFinder)

O Mars-PathFinder foi o segundo projeto da Nasa (1996) em relação a Marte. A missão (http://nssdc.gsfc.nasa.gov/planetary/mesur.html) é composta por uma estação de lançamento e transmissão de dados que é formado por baterias solares nas suas pétalas que, combinando com baterias recarregáveis, dá a energia necessária (Sagan Memorial Sation) e um robô de exploração de superfícies (Sojourner).

O robô Sojourner, figuras 2.8 e 2.10, é composto de seis rodas, com uma placa de captação de energia solar, com instrumentos de análise química para caracterizar as rochas e solos de Marte. O robô Sojourner era controlado por um operador baseado na Terra que usava imagens obtidas tanto pelo robô quanto pelos sistemas da estação de lançamento e transmissão de dados.

É importante ressaltar que o atraso de tempo é de aproximadamente 10-15 minutos, dependendo da posição relativa de Terra e Marte, necessitando deste modo de alguns controles autônomos da estação.

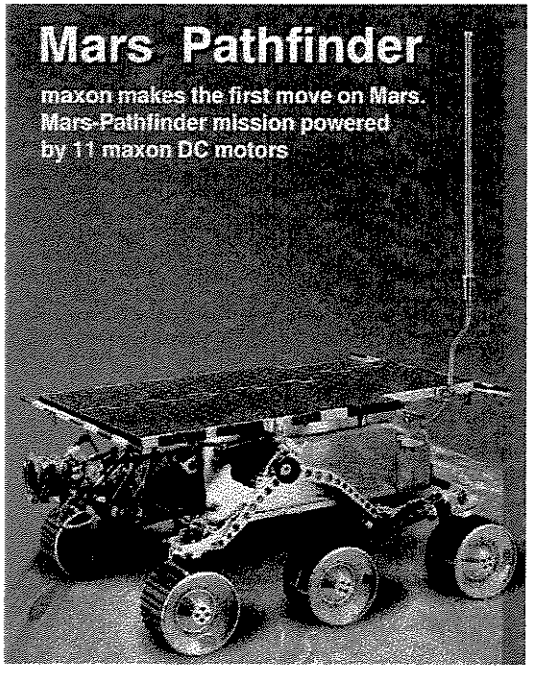


Figura 2.8: Veículo robótico móvel Sojourner.

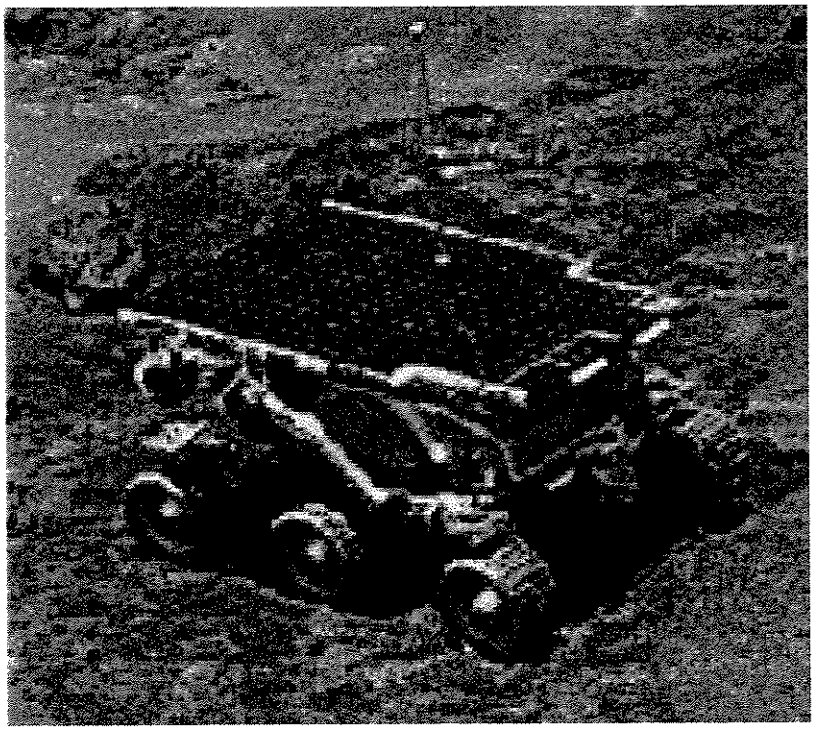


Figura 2.9: Veículo robótico móvel Sojourner no solo de Marte.

2.4 Desenvolvimento do trabalho

Neste trabalho foram realizados estudos para a automatização de tarefas submarinas utilizando manipuladores industriais.

2.4.1 Descrição do Problema

Para a extração de petróleo em alto mar é utilizada uma estrutura denominada Octos™ (projetada pela Petrobrás), figura 2.10. Este sistema possui painéis nas extremidades de seus braços os quais possuem válvulas que necessitam ser abertas ou fechadas. Estes procedimentos só podem ser realizados utilizando robôs devido a grande profundidade em que se encontram.

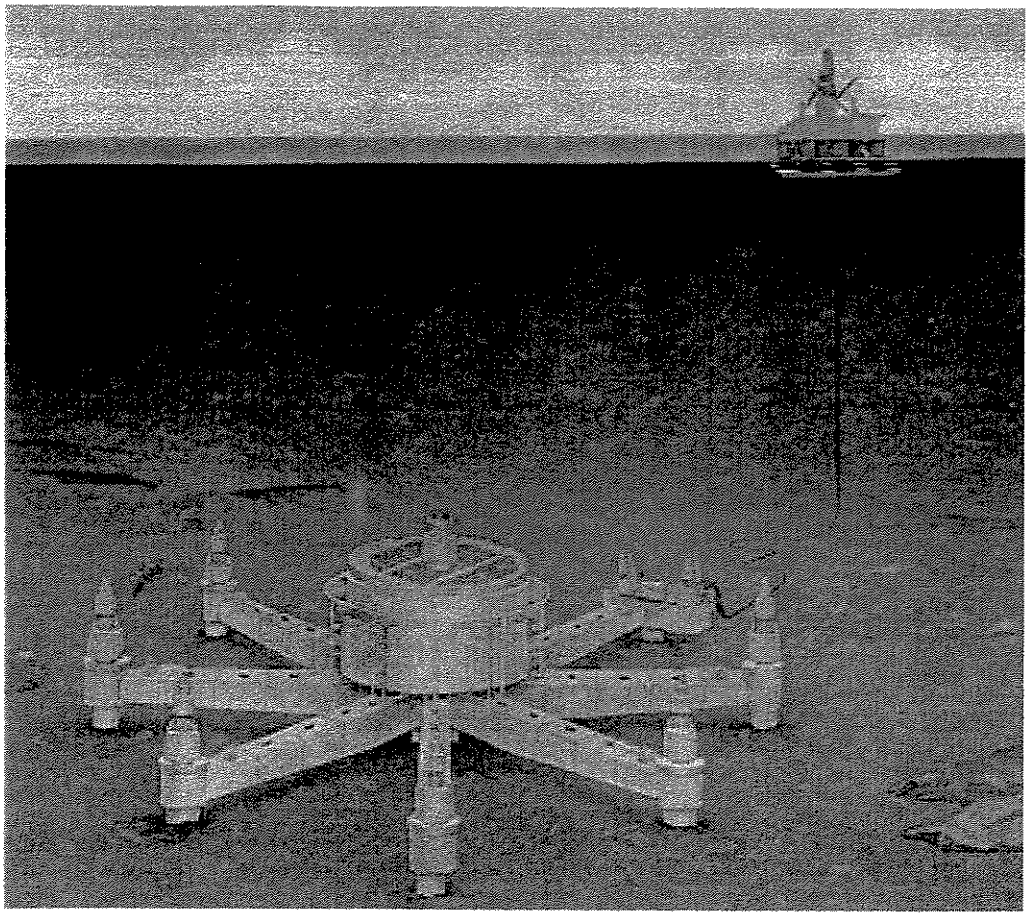


Figura 2.10: Estrutura Octos^{TM}.

Para que o dispositivo robótico atue nesta estrutura foram desenvolvidas uma base móvel sem mecanismo de propulsão e ferramentas dedicadas, figura 2.11 (estas figuras serão detallhadas no decorrer do Capítulo 9). Para a realização dos estudos para automatização de tarefas submarinas, utilizando manipuladores industriais, foram utilizados o robô Manutec e o manipulador submarino Kraft.

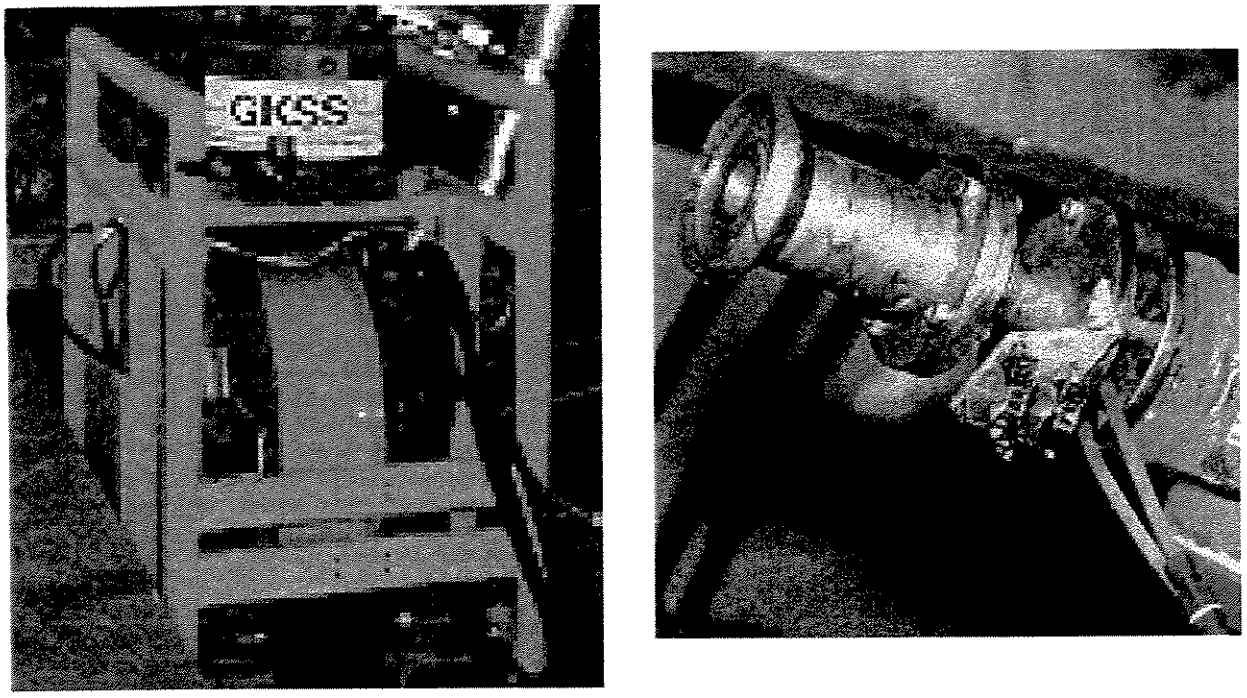


Figura 2.11: Base móvel e ferramenta desenvolvida.

2.4.2 Robô Manutec R3

O robô Manutec é um robô industrial e os seus movimentos são comandados via dispositivo de telecomando, teach-in box. Este dispositivo move cada junta do robô isoladamente para fornecer a posição e orientação da garra.

Visando a integração do robô, com as ferramentas dedicadas e base móvel na estrutura do Octos™ foi desenvolvido um pacote computacional "off-line". A figura 2.12 mostra a estrutura implementada.

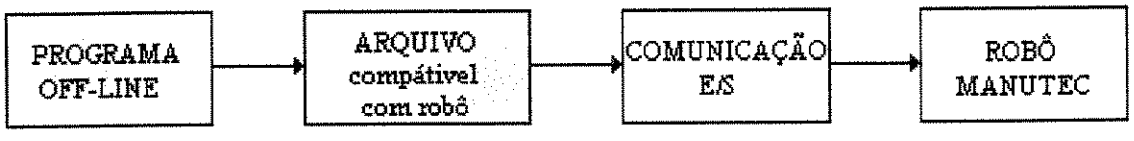


Figura 2.12: Estrutura implementada para o robô Manutec.

No decorrer desta tese de doutorado serão apresentados os métodos que foram utilizados para a implementação do programa computacional "off-line" e os resultados obtidos serão apresentados no Capítulo 8. No capítulo 9 serão apresentados os alguns resultados experimentais obtidos com as ferramentas dedicadas, com a base móvel e com uma maquete do sistema Octos™.

2.4.3 Manipulador Kraft

Como mencionado anteriormente o manipulador Kraft é um sistema robótico do tipo "master-slave". O controlador que existia era primitivo (PD - Proporcional-derivativo), sendo que basicamente, os sinais provenientes dos potenciômetros associados a cada junta do "master" servem como referência ao controlador de posição existente.

Visando a viabilizar o monitoramento e controle do manipulador diretamente via micro-computador, do teclado ou do "mouse" espacial, foi desenvolvido um programa em linguagem C, estruturado em módulos objetivando facilitar modificações e/ou expansões, denominado CMK. Os módulos foram criados a partir da configuração do sistema Kraft, Anexo I tópico I.4.1.

Este programa permite os seguintes modos de operação do sistema, sejam utilizados:

- modo "master-slave": sistema original;

- modo microcomputador: a partir da utilização de teclas dedicadas torna-se possível além do controle direto de cada junta do manipulador (sem a utilização do "master"), o controle direto do elemento terminal ou ferramenta (utilização do modelo cinemático);

- modo "mouse" espacial: a partir de um dispositivo externo, ergonométrico do ponto de vista do operador, possamos controlar diretamente a extremidade da ferramenta.

A partir desses modos de operação será possível, a realização de tarefas automatizadas do manipulador em estudo, flexibilizando a sua utilização. A figura 2.13 mostra a estrutura implementada.

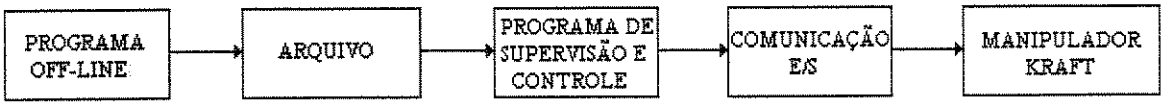


Figura 2.13: Estrutura implementada para o manipulador Kraft.

No Capítulo 3 o pacote computacional de supervisão e controle (CMK) implementado é descrito mais detalhadamente.

Capítulo 3

Descrição do Sistema de Supervisão e Controle Implementado - CMK

O programa CMK (do sistema de supervisão e controle do manipulador) foi codificado na linguagem C, incluindo rotinas para o monitoramento e a visualização dos potenciômetros das juntas, o envio de sinais para os atuadores hidráulicos da juntas e a supervisão de tarefas de comando. Este programa permite ainda a troca de informações através de arquivos de dados com o pacote de programação "off-line" existente.

3.1 Pacote computacional CMK

O pacote computacional CMK viabiliza o monitoramento e controle do manipulador diretamente via micro-computador, através do teclado ou do "mouse" espacial. A tela inicial é mostrada na figura 3.1. Os módulos foram criados a partir da configuração do sistema Kraft, Anexo I tópico I.4.1.

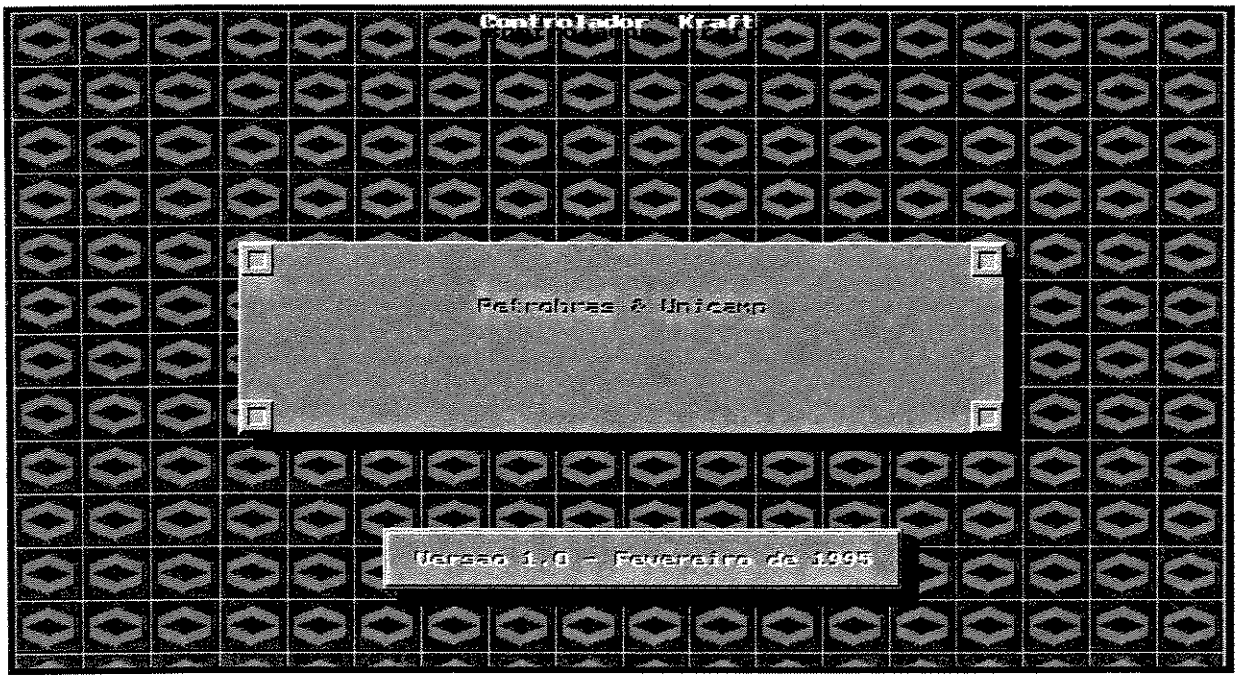


Figura 3.1: Tela inicial do programa CMK.

O sistema é capaz de suportar dois tipos de programação de trajetória: "on-line" e "off-line". Os arquivos com trajetórias geradas na programação "off-line" podem ser enviados ao manipulador num dos modos de operação.

Para permitir o monitoramento e comando do manipulador diretamente pelo microcomputador foram instaladas interfaces A/D, D/A e digital que permitem o acesso a todos os sinais (monitoramento e controle) do manipulador.

3.2 Descrição geral do pacote de supervisão e controle - CMK

O pacote CMK apresenta em seu menu inicial, figura 3.2, as opções "master-slave", microcomputador, teste de "hardware" e Dos.

Figura 3.2: Menu inicial do programa CMK.

As opções disponíveis ao operador a partir do menu inicial são apresentadas no diagrama de blocos, figura 3.3.

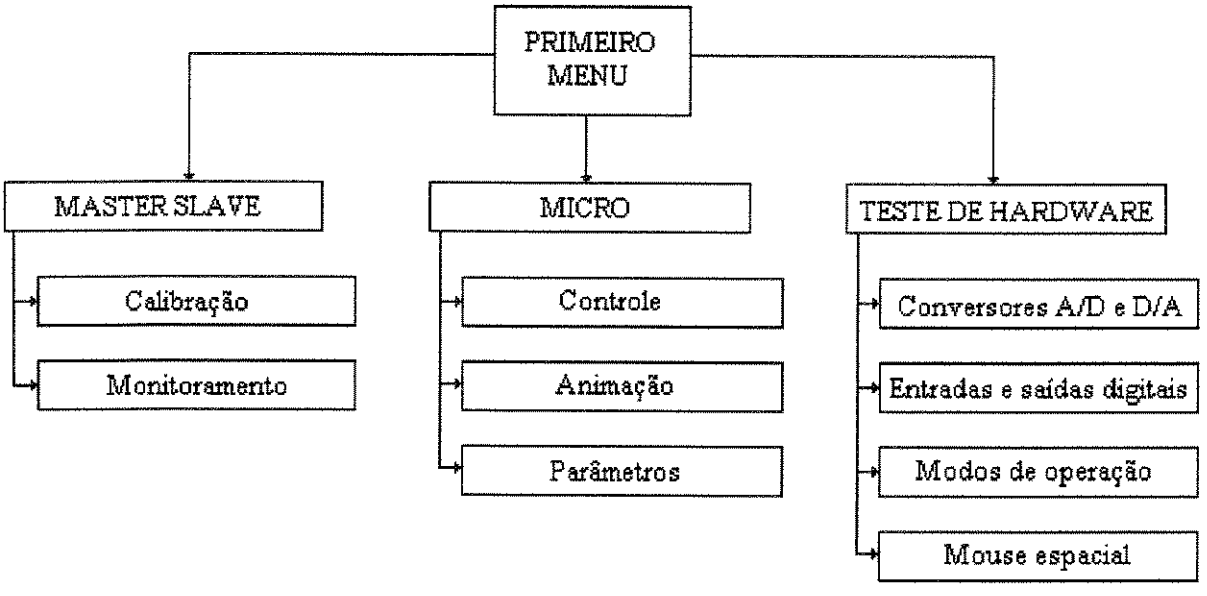


Figura 3.3: Diagrama de blocos do programa.

3.2.1 Descrição dos modos de operação

Três modos básicos de operação do manipulador foram disponibilizados: modo "master-slave", modo microcomputador e modo "mouse" espacial. Estes modos serão descritos a seguir.

3.2.1.1 "Master-slave"

Permite que ocorra a condição original de funcionamento do manipulador, passando o controle do microcomputador para o modo "master". A partir daí, a operação do manipulador é regida pela movimentação no "master".

O "master" é um manipulo antropomórfico, semelhante geometricamente ao "slave" (manipulador), porém em escala reduzida, cuja variação de posição acarreta a correspondente movimentação no manipulador. Possui seis juntas constituídas de potenciômetros de precisão, e mais dois potenciômetros para a medição de abertura/fechamento e rotação da garra. Chaves adicionais controlam funções especiais do manipulador, como rotação da garra (contínuo ou manual) e corte de pressão hidráulica, entre outras.

3.2.1.2 Microcomputador

O controle do manipulador, ao invés de ser exercido pelo "master", passa a ser efetuado pelo microcomputador. A opção Controle, acessível a partir da opção Microcomputador do menu principal, oferece três formas de operação:

Modo 1 - Controle automático

Possibilita ao operador enviar trajetórias previamente tratadas e armazenadas. Podem ser enviados trajetórias únicas (simples) ou um conjunto de trajetórias (compostas) formando uma tarefa específica.

O controle de abertura da garra, fechamento e rotação, são realizados através de uma instrução lógica no final de cada trajetória que compõe a tarefa desejada.

Modo 2 - Controle via teclado

O controle é realizado diretamente através do teclado. As seguintes opções encontram-se disponíveis neste modo, figura 3.4:

- "Mouse": muda o controle do modo teclado para o modo "mouse" espacial;

- Define Reta: define a reta entre dois pontos escolhidos pelo operador. Estes pontos, devidamente tratados, formam uma trajetória, que é armazenada e pode ser enviada ao manipulador;

- Grava Posição: grava em um arquivo a posição atual do manipulador;

- Grava Pontos: ao primeiro toque guarda em arquivo a posição do manipulador. Cada subseqüente chamada da função armazena a posição atual no mesmo arquivo;

- TV: mostra tela para a imagem de uma câmera a ser instalada no manipulador.

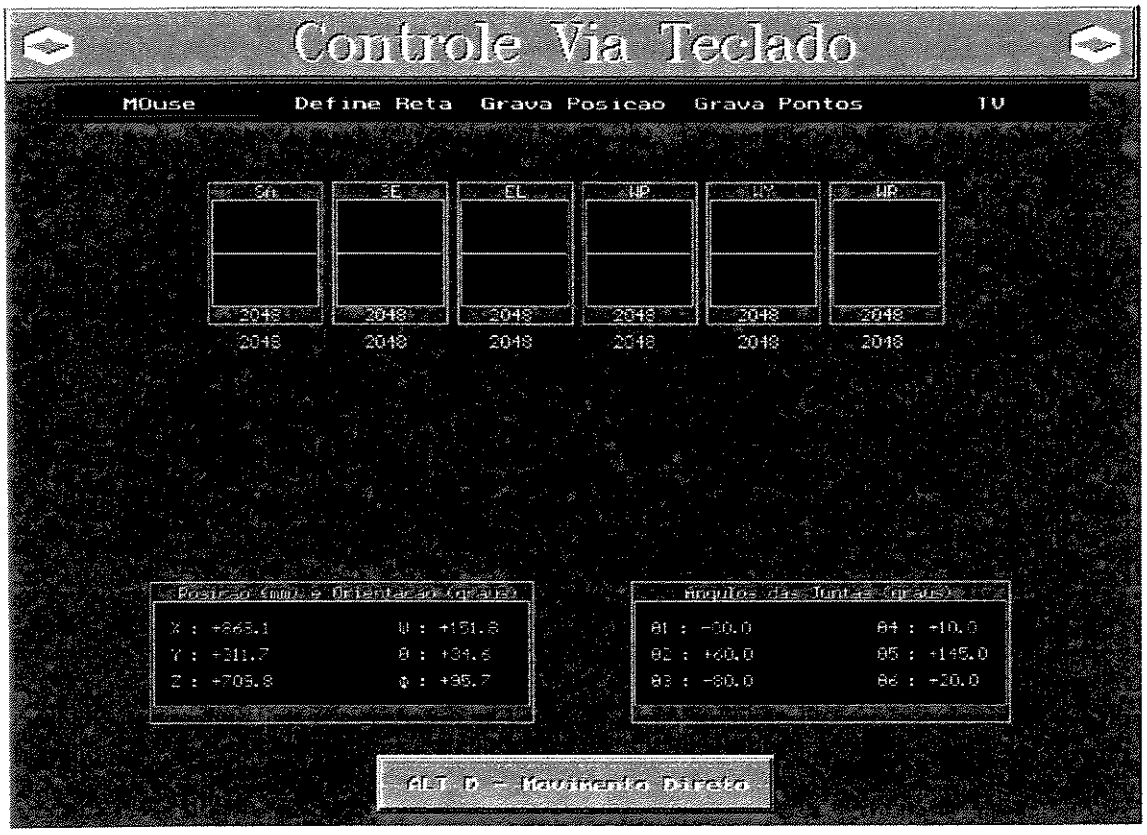


Figura 3.4: Tela do modo Controle via Teclado.

Na tela apresentada na figura 3.4 são mostrados:

- gráficos mostrando os valores convertidos dos potenciômetros do manipulador;
- posição e orientação da garra;
- ângulos das juntas do manipulador.

Este modo apresenta ainda uma função, acionada pelas teclas ALT D, que ativa o movimento direto e desativa o menu acima descrito. Após isso, passa-se a ter a possibilidade de movimentar cada uma das juntas separadamente ou fazer movimentos nas direções X, Y ou Z da orientação da garra.

Modo 3 - Controle via "mouse" espacial

O manipulador é controlado diretamente pelo "mouse" espacial*, figura 3.5. Este sistema, desenvolvido pelo CENPES – Centro de Pesquisas e Desenvolvimento da Petrobrás, gera sinais de deslocamentos em coordenadas cartesianas no espaço tridimensional, que são enviadas ao microcomputador, a fim de controlar a posição e orientação do manipulador. O sistema é composto de um manipulo, fios, molas e potenciômetros, cujos valores de tensão chegam ao microcomputador através de uma porta serial.

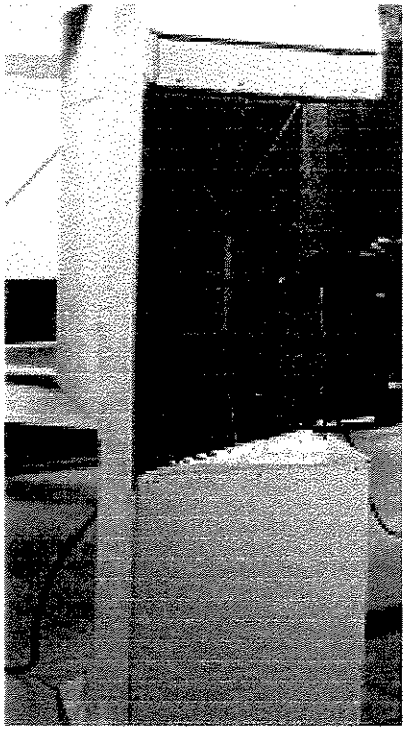


Figura 3.5: "Mouse" espacial.

As seguintes opções, figura 3.6, estão disponíveis neste modo:

- Teclado: muda o controle do modo "mouse" espacial para o teclado;
- Grava Posição: grava em um arquivo a posição atual do manipulador;

* A denominação "mouse" espacial é feita por analogia aos "mouses" bidimensionais tradicionalmente utilizados em microcomputadores.

- Grava Trajetória: grava todas as mudanças de posição do manipulador a partir do momento em que é acionada até que outra opção seja ativada;

- TV: mostra tela para a imagem de uma câmera a ser instalada no manipulador.

Nesta janela são mostrados ainda: gráficos explicitando os valores convertidos dos potenciômetros do manipulador, a posição e a orientação da garra e os ângulos das juntas do manipulador.

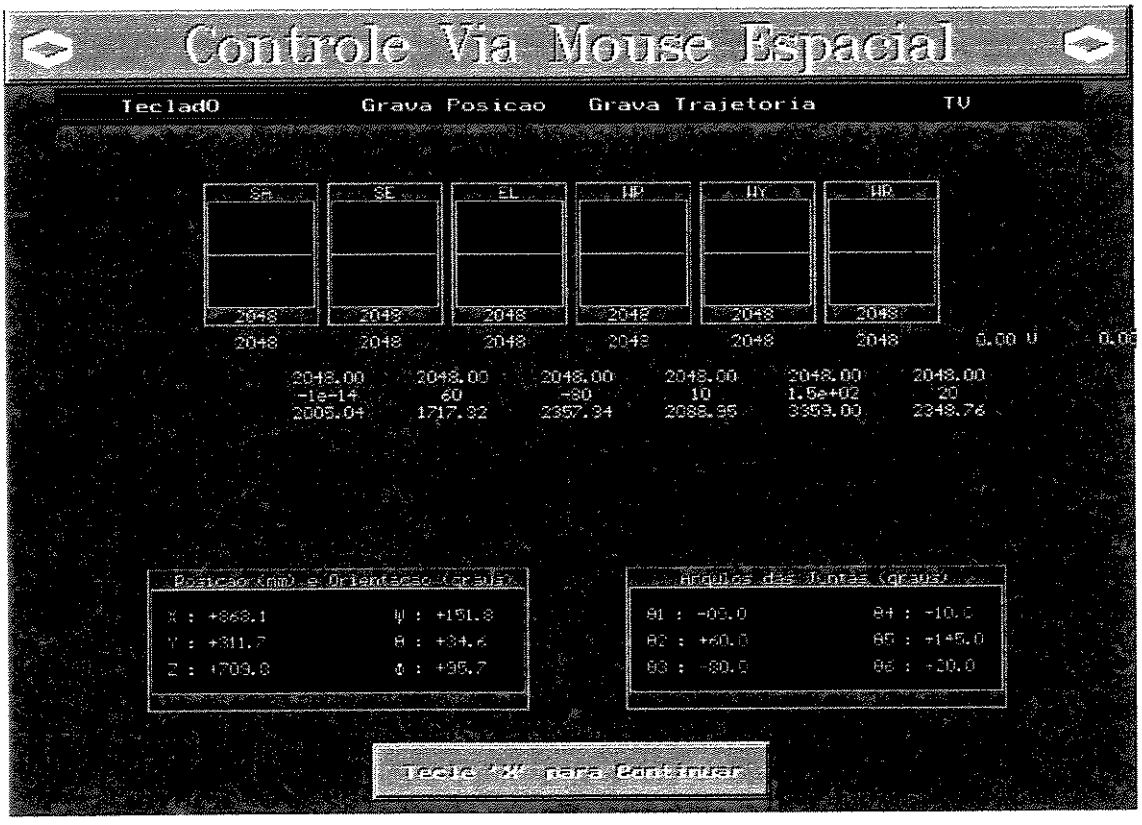


Figura 3.6: Tela do modo Controle via "mouse" espacial.

3.2.2 Chaveamento entre os modos

Em virtude dos manipuladores submarinos serem geralmente utilizados na configuração "master-slave", as funcionalidades disponíveis no controle original do manipulador foram

mantidas no novo sistema. Para tanto, a UNICAMP desenvolveu uma interface permitindo o chaveamento para um microcomputador PC de todos os sinais provenientes do "master" e do "slave", bem como de todos os sinais necessários para o acionamento das juntas (lógicos e analógicos).

A interface de chaveamento de modos KMC* tem a finalidade de colocar o sistema em um dos modos descritos acima, isto é, ele interliga ou o "master" ou o micro ao MSC (modulo de controle do sistema "master").

No Anexo XII são apresentados outros recursos disponíveis no pacote de supervisão e controle CMK.

3.3 Módulo de inicialização automática do sistema

Este módulo tem como objetivo o estabelecimento de um sistema automático para identificação e calibração dos parâmetros do sistema (Campos, 1993; Saramago, 1993), isto é necessário na programação "off-line" para o robô atingir o posicionamento numericamente preciso.

Possui as seguintes características:

- programa de correção dos parâmetros do sistema;
- programa para inicialização automática do sistema no ambiente de trabalho.

A estrutura do sistema de supervisão e controle, para o manipulador Kraft, é apresentada na figura 3.7.

* Esta interface foi projetada e construída na Unicamp com a finalidade de viabilizar a automação do manipulador Kraft

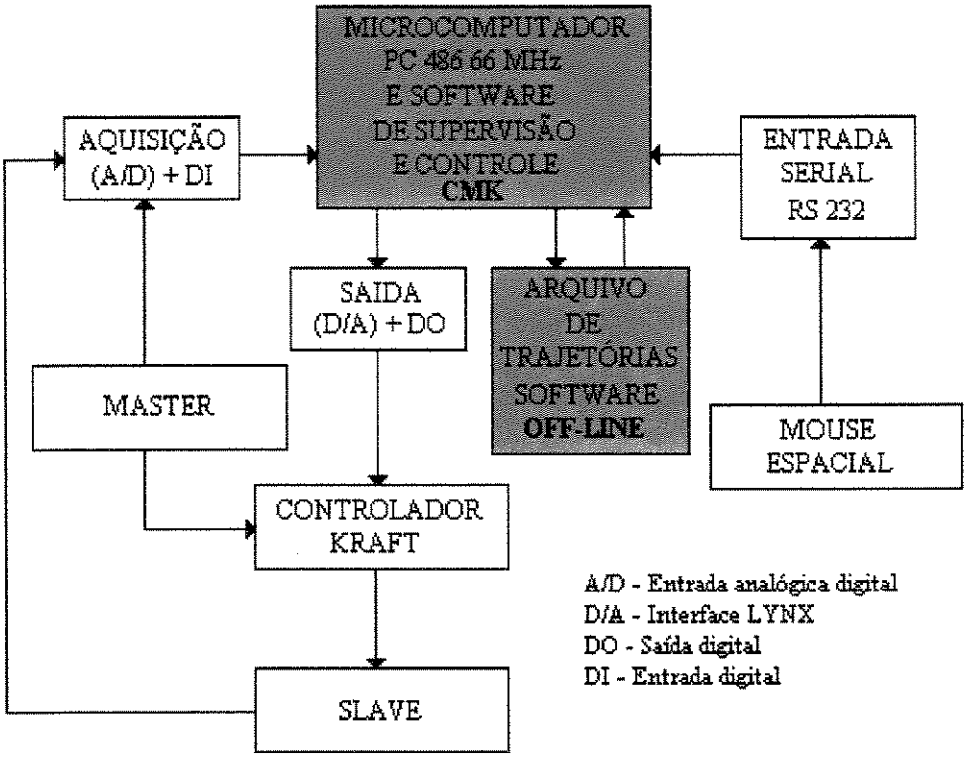


Figura 3.7: Estrutura do sistema de supervisão e controle - completo.

No decorrer dos capitulos, desta tese, serão abordados métodos utilizados para a geração de trajetórias, para testes de colisão, construção gráfica do robô e do ambiente de trabalho que foram implementadas no programa computacional "off-line" (módulos Simula, Trajeto e Obstáculo).

Capítulo 4

Modelagem de Manipuladores

Neste capítulo é feita uma breve discussão sobre a modelagem de robôs, controle e geração de trajetórias de robôs, pois para a obtenção de uma trajetória angular a partir da posição e orientação do elemento terminal, é necessário em primeiro lugar obter o modelo geométrico do robô e, após isso, obter o modelo geométrico inverso.

Será apresentado o modelo geométrico obtido para o robô Manutec utilizando o método de Denavit-Hartenberg e o método de vetores locais, as formas de obter o modelo geométrico inverso (suas vantagens e desvantagens) e uma breve discussão sobre a matriz Jacobiana. No Anexo I será apresentado o modelo geométrico, obtido de ambos os modos, para o manipulador Kraft.

Também será apresentada a obtenção da matriz de orientação do elemento terminal de um robô a partir dos três ângulos: roll, pitch e yaw (que descrevem a orientação do robô em relação às coordenadas da base) e da obtenção destes três ângulos a partir da matriz orientação (problema inverso).

4.1 Revisão geral sobre a modelagem, controle e geração de trajetórias de robôs

4.1.1 Modelagem

Em um robô, os diferentes graus de liberdade podem ser associados a diversos sistemas de coordenadas servindo para descrever o movimento de cada grau de liberdade. Mais especificamente, o modelo geométrico é aquele que expressa a posição e orientação do elemento terminal em relação a um sistema de coordenadas solidário à base do robô, em função de suas coordenadas generalizadas (angulares, no caso de juntas rotacionais), figura 4.1. Esta relação pode ser expressa matematicamente a partir de uma matriz que relaciona o sistema de coordenadas da base com o sistema de coordenadas do último elemento. Esta matriz é chamada de matriz de passagem homogênea do robô.

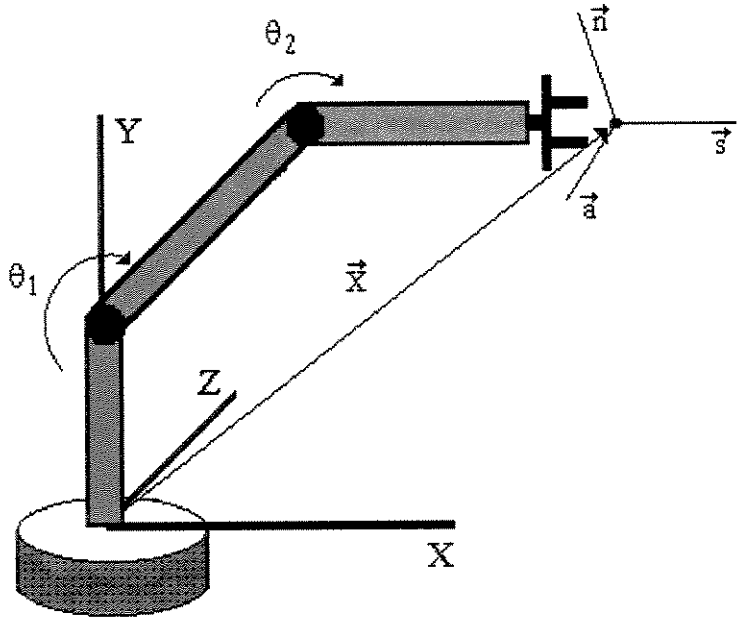


Figura 4.1: O modelo geométrico descreve a posição ($\vec{x}$) e orientação (vetores $\vec{n}$, $\vec{s}$ e $\vec{a}$) do elemento terminal em relação a um sistema de coordenadas solidário à base do robô.

Embora a definição do modelo geométrico seja única, a maneira de obter a matriz de passagem homogênea do robô está associada ao sistema de referência utilizado. Existem diversas maneiras, e cada uma gera expressões diferentes que, embora equivalentes quantitativamente,

podem diferir quanto ao número de operações aritméticas necessárias para fazer o cálculo numérico das mesmas.

Neste trabalho o robô foi modelado através de dois métodos: através da sistemática de Denavit-Hartenberg e por vetores locais. Modelando-se o robô através da sistemática de Denavit-Hartenberg serão obtidas a posição e a orientação finais do elemento terminal. O modelo obtido desta forma é indicado para o uso em programas de geração de trajetórias pois, nestes programas o que interessa é apenas a posição e orientação do elemento terminal.

Modelando-se através de vetores locais serão obtidas as posições e as orientações de diversos pontos de interesse que poderão ser utilizados para a construção gráfica do robô através de elementos primitivos.

Neste trabalho iremos adotar o modelo geométrico para o robô Manutec produzido pela Siemens que está disponível na Unicamp no Laboratório de Automação Integrada e Robótica (LAIR). No capítulo 4 as equações obtidas, através destes dois métodos, serão apresentadas.

4.1.2 Geração de trajetórias e controle

Depois de se obter o modelo geométrico do robô pode-se entrar no problema de geração de trajetórias. A trajetória é a evolução no tempo da posição, da velocidade e aceleração das juntas do robô, isto é, a geração de incrementos angulares de juntas necessárias para que o robô realize uma determinada tarefa, figura 4.2.

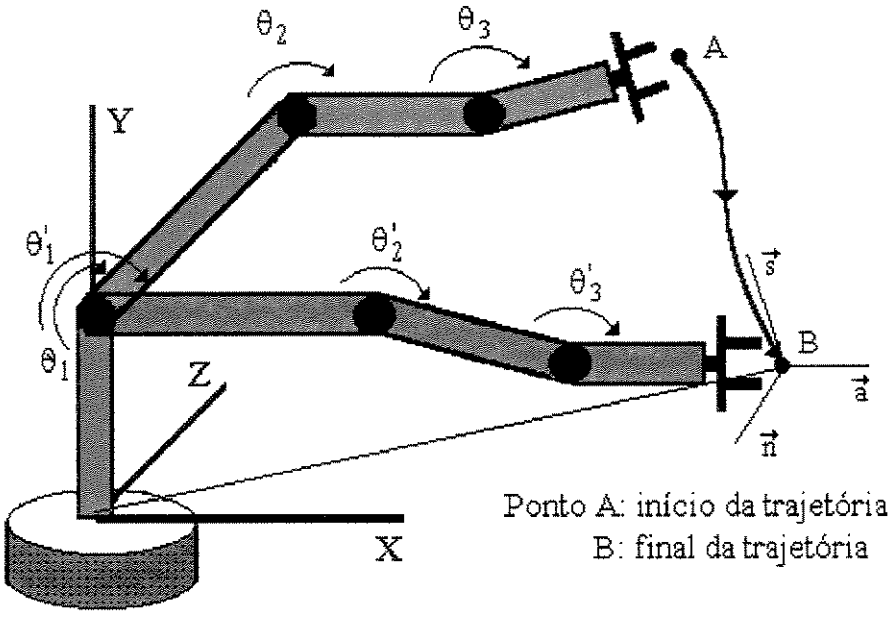


Figura 4.2: Trajetória para o robô ir da posição A até a posição B.

As trajetórias podem ser especificadas em coordenadas de juntas ou cartesianas. Em muitas aplicações a programação de tarefas de robôs é realizada no espaço das juntas, não necessitando de um modelo geométrico, e a trajetória angular de mesma natureza dos sinais provenientes do transdutor de posição servirá como sinal de referência para o controlador de cada junta robótica.

Mas na maioria das aplicações, a realização de tarefas está relacionada com o tipo de ferramenta utilizada, a partir de um sistema de coordenadas cartesianas fixo à base do robô, espaço cartesiano, figura 4.3. Vemos assim que os movimentos desejados e as leis de controle estão em espaços diferentes. A obtenção de referências correspondentes às tarefas definidas no espaço operacional é denominada coordenação de movimentos; a figura 4.4 mostra um esquema simplificado. A coordenação é expressa matematicamente pela inversão do modelo geométrico.

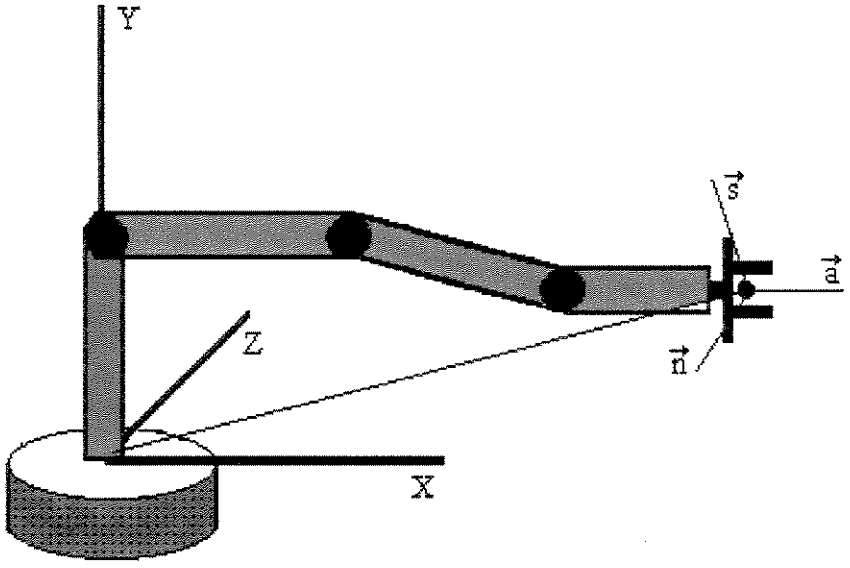


Figura 4.3: Espaço de coordenadas de um robô.

Existem dois métodos para a solução do problema da inversão do modelo geométrico: os métodos analíticos e os métodos numéricos. Estes métodos serão discutidos no decorrer deste capítulo.

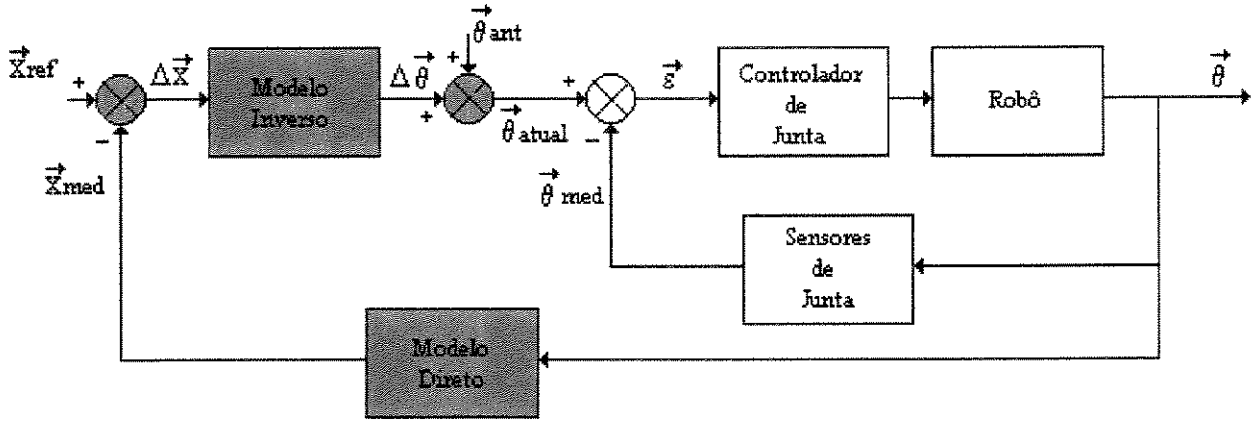


Figura 4.4: Malha simplificada de controle de um robô.

Portanto, para a implementação de um algoritmo de geração de trajetórias no espaço cartesiano, como pode ser observado através da figura 4.4, é necessário o conhecimento do modelo geométrico do robô e também de métodos para a inversão do mesmo, para gerar os ângulos das juntas correspondentes a uma determinada configuração espacial. Além do mais, a

geração de trajetórias no espaço cartesiano possui a vantagem, em relação à programação de tarefas no espaço das juntas, de que a trajetória gerada entre dois pontos é muito bem definida. Um exemplo é quando a ferramenta do robô deve seguir um caminho específico, tal como uma reta ou um arco de círculo. A figura 4.5 mostra um robô executando uma tarefa que necessita de um movimento em linha reta.

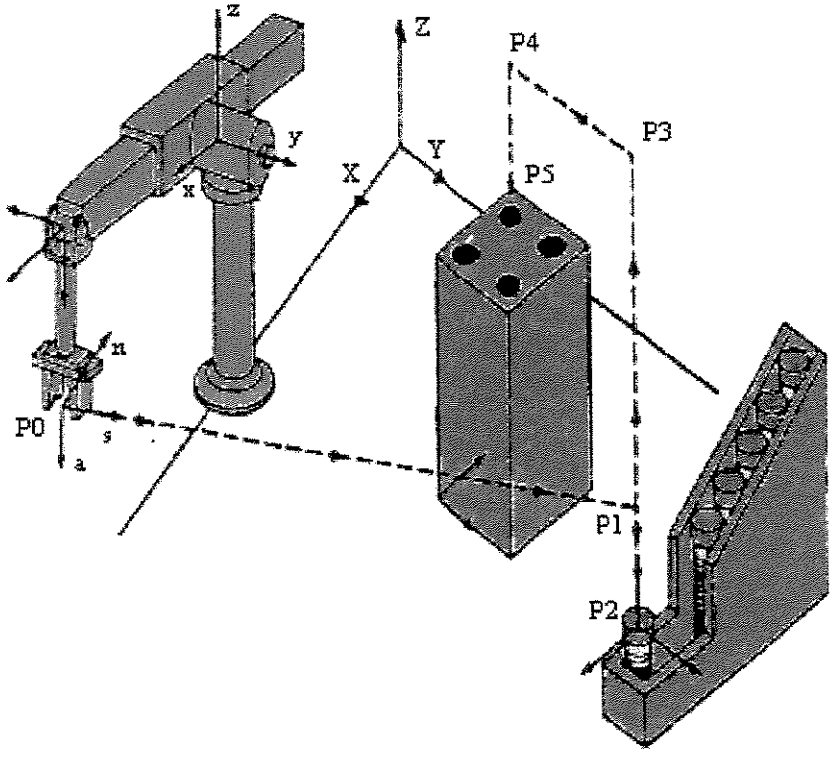


Figura 4.5: Robô executando uma tarefa que necessita de um movimenta em linha reta.

4.2 Modelo geométrico

O modelo geométrico de um robô é definido como a função vetorial $\vec{f}$, que exprime um vetor $\vec{x}$ (espaço cartesiano) em função do vetor $\vec{\theta}$ (coordenadas angulares):

$$\vec{x} = \vec{f}\,(\vec{\theta}) \tag{4.1}$$

onde $\vec{\theta} = (\theta_1, \theta_2, ..., \theta_n)$representa o vetor das posições angulares das juntas;

$\vec{x} = (p_x , p_y , p_z, \psi, \theta, \phi)$ representa vetor posição, onde os três primeiros termos denotam a posição cartesiana e os três últimos a orientação do órgão terminal.

A função $\vec{f}$, que é o modelo geométrico, é então a expressão analítica da composição dos movimentos das juntas para realizar o movimento do elemento terminal do robô.

Embora a definição do modelo geométrico seja única, a maneira de obter a matriz de passagem homogênea do robô está associada ao sistema de referência utilizado. Existem diversas maneiras, e cada uma gera expressões diferentes que, embora equivalentes quantitativamente, podem diferir quanto ao número de operações aritméticas necessárias para fazer o cálculo numérico das mesmas. Pode-se citar duas maneiras de se obter a matriz de passagem homogênea do robô: através da sistemática de Denavit-Hartenberg e através de vetores locais.

A seguir será apresentado o modelo geométrico obtido para o robô Manutec, figura 4.6 (a) através destes dois métodos.

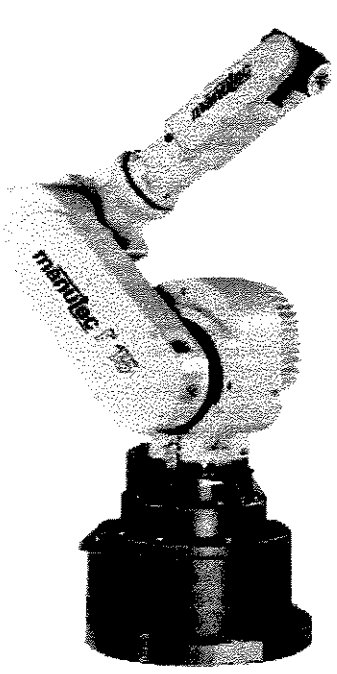
4.2.1 Sistemática de Denavit-Hartenberg

Neste método, Anexo I, são obtidas apenas as coordenadas do elemento terminal do robô. O modelo obtido deste modo pode ser utilizado em programas de geração de trajetórias e de identificação de erros, entre outros, uma vez que são necessárias apenas as coordenadas do elemento terminal.

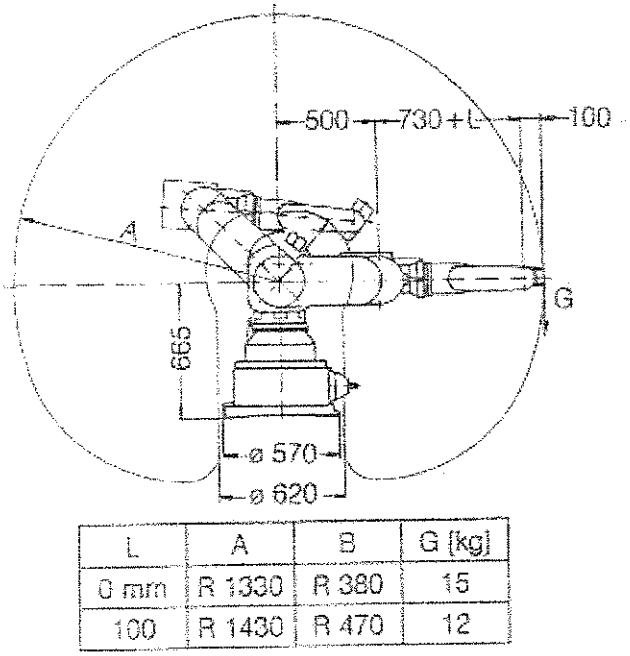
A tabela 4.1 mostra os parâmetros de Denavit-Hartenberg obtidos para o robô Manutec e a tabela 4.2, as matrizes de passagem obtidas a partir destes parâmetros. As dimensões apresentadas na tabela 4.1 foram obtidas a partir da figura 4.6 (b).

JUNTA	θ (graus)	D (mm)	α (graus)	A (mm)	range(graus)
1	θ_1	665.0	-90.0	0.0	-165/+200.5
2	θ_2	0.0	0.0	500.0	-20/+190
3	θ_3	0.0	90.0	0.0	-135/+135
4	θ_4	730.0	-90.0	0.0	± 190
5	θ_5	0.0	90.0	0.0	± 120
6	θ_6	100.0	0.0	0.0	± 265

Tabela 4.1: Parâmetros de Denavit-Hartenberg



(a)



L	A	B	G [kg]
0 mm	R 1330	R 380	15
100	R 1430	R 470	12

(b)

Figura 4.6: Robô Manutec (a) e suas dimensões (b).

$$A_{0,1} = \begin{bmatrix} C1 & 0 & -S1 & 0 \\ S1 & 0 & C1 & 0 \\ 0 & -1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad A_{1,2} = \begin{bmatrix} C2 & -S2 & 0 & a2.C2 \\ S2 & C2 & 0 & a2.S2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_{2,3} = \begin{bmatrix} C3 & 0 & S3 & 0 \\ S3 & 0 & -C3 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad A_{3,4} = \begin{bmatrix} C4 & 0 & -S4 & 0 \\ S4 & 0 & C4 & 0 \\ 0 & -1 & 0 & d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_{4,5} = \begin{bmatrix} C5 & 0 & S5 & 0 \\ S5 & 0 & -C5 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad A_{5,6} = \begin{bmatrix} C6 & -S6 & 0 & 0 \\ S6 & C6 & 0 & 0 \\ 0 & 0 & 1 & d_6 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Onde: $Ci = \cos(\theta_i)$ e $Si = \text{sen}(\theta_i)$ para $i = 1, 2, ..., 6$

Tabela 4.2: Matrizes de passagem.

As equações obtidas para o robô são apresentadas abaixo.

$$T_6(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6) = A_{0,1} * A_{1,2} * A_{2,3} * A_{3,4} * A_{4,5} * A_{5,6} \tag{4.2}$$

$$T_6(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6) = \begin{bmatrix} \vec{n} & \vec{s} & \vec{a} & \vec{p} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.3}$$

onde $\vec{p} = [\, p_x , p_y , p_z \,]$: vetor posição;

$\vec{n} = [\, n_x\ n_y\ n_z \,]$, $\vec{s} = [\, s_x\ s_y\ s_z \,]$ e $\vec{a} = [\, a_x\ a_y\ a_z \,]$ vetores ortonormais que descrevem a orientação.

$$n_x = ((c1*c23*c4 - s1*s4)*c5 - c1*s23*s5)*c6 + (-c1*c23*s4 - s1*c4)*s6$$
$$n_y = ((s1*c23*c4 - c1*s4)*c5 - s1*s23*s5)*c6 + (-s1*c23*s4 - c1*c4)*s6$$
$$n_z = (-s23*c4*c5 - c23*s5)*c6 + s23*s4*c6$$

$$s_x = -((c1*c23*c4 - s1*s4)*s5 - c1*s23*s5)*s6 + (-c1*c23*s4 - s1*c4)*c6$$
$$s_y = -((s1*c23*c4 - c1*s4)*c5 - s1*s23*s5)*s6 + (-s1*c23*s4 - c1*c4)*c6$$
$$s_z = (s23*c4*c5 - c23*s5)*s6 + s23*s4*c6$$

(4.4)

$$a_x = (c1*c23*c4 - s1*s4)*s5 + c1*s23*c5$$
$$a_y = (s1*c23*c4 - c1*s4)*s5 + s1*s23*c5$$
$$a_z = -s23*c4*s5 + c23*c5$$

$$p_x = ((c4*s5*D6)*c23 + (c5*D6 + D4)*s23 + A2*c2)*c1 - s4*s5*D6*s1$$
$$p_y = ((c4*s5*D6)*c23 + (c5*D6 + D4)*s23 + A2*c2)*s1 - s4*s5*D6*c1$$
$$p_z = -(c4*s5*D6)*s23 + (c5*D6 + D4)*c23 - A2*s2 + D1$$

onde $c_i = \cos(\theta_i)$

$s_i = \text{sen}(\theta_i)$

$c_{ij} = \cos(\theta_i + \theta_j)$

$s_{ij} = \text{sen}(\theta_i + \theta_j)$

para i, j = 1, 2, ..., 6.

A figura 4.7 mostra o volume de trabalho do robô Manutec.

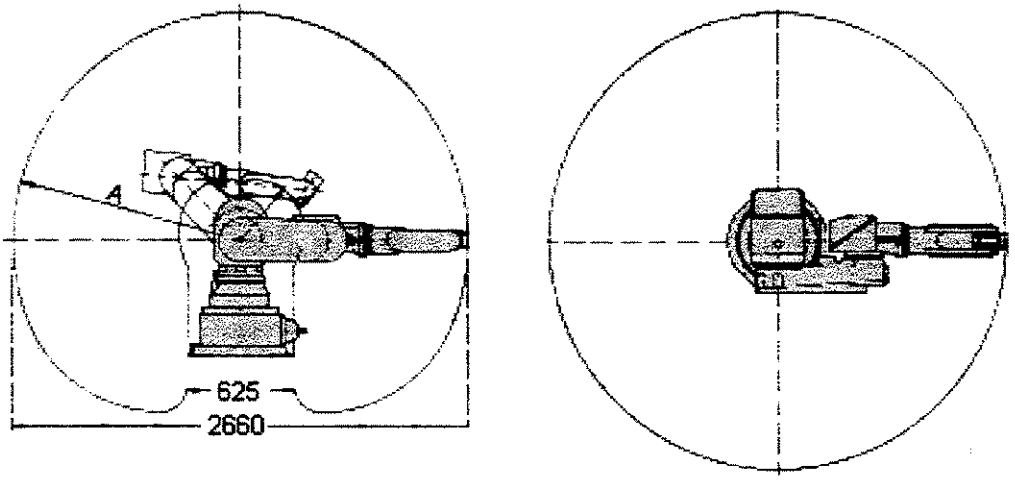


Figura 4.7: Volume de trabalho do robô Manutec.

4.2.2 Vetores Locais

Diferentemente do modelo utilizado no programa de geração de trajetórias onde interessam apenas as coordenadas finais do elemento terminal do robô, no software de programação "off-line" (simulação e visualização) é necessária a determinação das coordenadas de diversos pontos do robô para construir o desenho gráfico através de sólidos primitivos (paralelepípedos, cilindros e outros) que representam o robô e executar testes de colisão com o ambiente.

O método consiste em determinar vetores para cada ponto de interesse de visualização e, a partir da multiplicação destes vetores por matrizes de transformação, obter as coordenadas (posição e orientação) de cada um destes pontos. A figura 4.8 mostra este método.

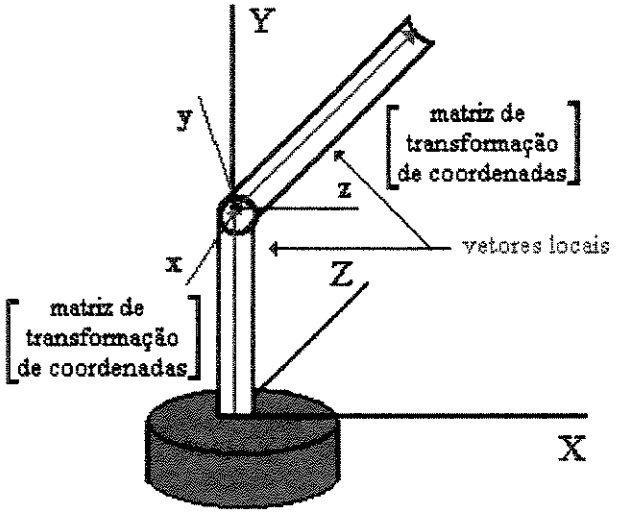


Figura 4.8: Descrição do método de vetores locais.

Portanto, para o software de programação "off-line" é necessário que se modele o robô através de vetores locais de translação e rotação. O modelo obtido desta forma, para o robô Manutec, é apresentado abaixo.

Em primeiro lugar, determinam-se os vetores de rotação e as matrizes de rotação; para isso utiliza-se a figura 4.9.

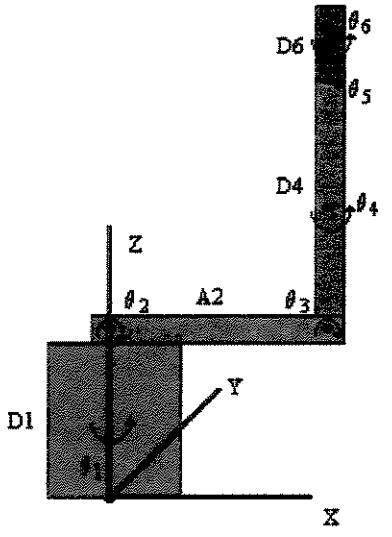


Figura 4.9: Representação do robô.

Os vetores de rotação baseando-se na figura 4.9 são dados por:

$\vec{V}r_1 = [\,0\,,0\,,1\,] - R1$ $\quad \vec{V}r_2 = [\,0\,,1\,,0\,] - R2$ $\quad \vec{V}r_3 = [\,0\,,1\,,0\,] - R3$

$\vec{V}r_4 = [\,0\,,0\,,1\,] - R4$ $\quad \vec{V}r_5 = [\,0\,,1\,,0\,] - R5$ $\quad \vec{V}r_6 = [\,0\,,0\,,1\,] - R6$

As matrizes de rotação são:

$R_1(z, \theta_1)$ $R_2(z, \theta_2)$ $R_3(z, \theta_3)$ $R_4(z, \theta_4)$ $R_5(z, \theta_5)$ $R_6(z, \theta_6)$

onde

$$R_i(z, \theta_i) = \begin{bmatrix} c\theta_i & -s\theta_i & 0 \\ s\theta_i & c\theta_i & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.5}$$

$$R_i(y, \theta_i) = \begin{bmatrix} c\theta_i & 0 & s\theta_i \\ 0 & 1 & 0 \\ -s\theta_i & 0 & c\theta_i \end{bmatrix} \tag{4.6}$$

e $\quad c_i = \cos(\theta_i);$

$\quad s_i = \text{sen}(\theta_i).$

para $i = 1, 2, \ldots, 6.$

A figura 4.10 mostra os pontos de visualização (pontos de interesse). E a tabela 4.3 apresenta as dimensões indicadas nas figuras 4.9 e 4.10.

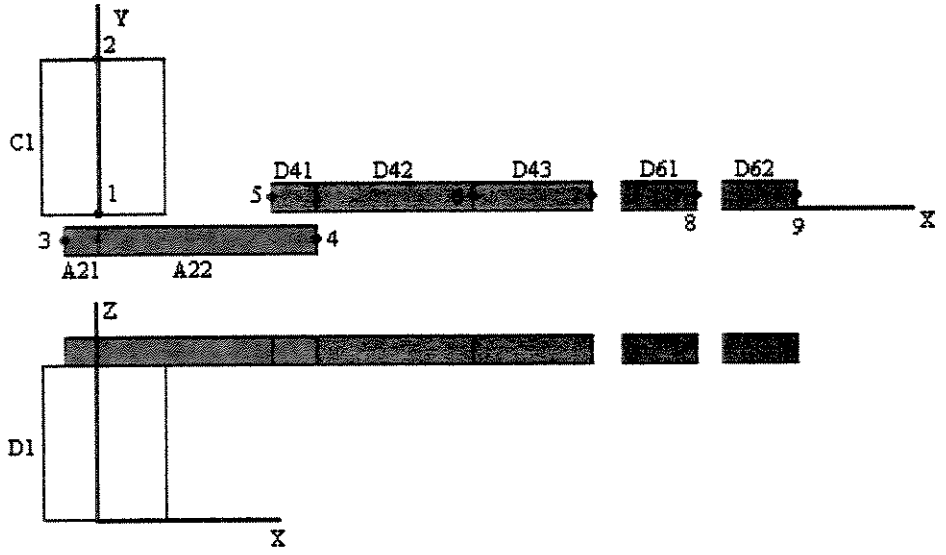


Figura 4.10: Visualização dos pontos de interesse do robô Manutec.

Parâmetro	dimensão (mm)	Parâmetro	dimensão mm	Parâmetro	dimensão mm
C1	370	A2	500	D4	730
A21	120	D41	280	D61	50
A22	500	D42	120	D62	50
D1	665	D43	610	D6	100

Tabela 4.3: Dimensões do robô em milímetros.

A figura 4.11 mostra os vetores de translação ($\vec{V}_i$), especificados em relação ao referencial XYZ.

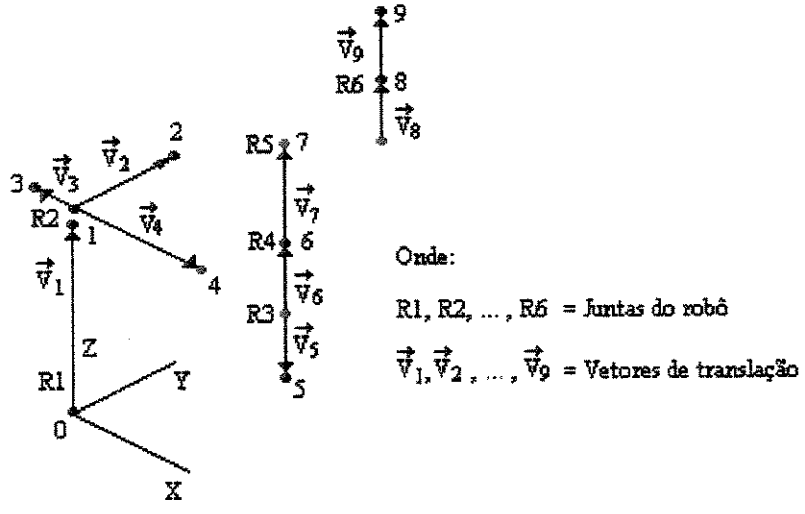


Figura 4.11: Vetores de translação e pontos de interesse.

$\vec{V}_1 = [\ 0\ ,\ 0\ ,\ d1\ ] - R1$ entre os pontos de visualização O0 e O1 inicia-se na junta R1

$\vec{V}_2 = [\ 0\ ,\ c1\ ,\ 0\ ] - R2$ entre os pontos de visualização O1 e O2 inicia-se na junta R2

$\vec{V}_3 = [\ -a21\ ,\ 0\ ,\ 0\ ] - R3$ entre os pontos de visualização O1 e O3 inicia-se na junta R2

$\vec{V}_4 = [\ a22\ ,\ 0\ ,\ 0\ ] - R4$ entre os pontos de visualização O1 e O4 inicia-se na junta R2

$\vec{V}_5 = [\ 0\ ,\ 0\ ,\ -d41\ ] - R5$ entre os pontos de visualização O4 e O5 inicia-se na junta R3

$\vec{V}_6 = [\ 0\ ,\ 0\ ,\ d42\ ] - R6$ entre os pontos de visualização O4 e O6 inicia-se na junta R3

$\vec{V}_7 = [\ 0\ ,\ 0\ ,\ d43\ ] - R7$ entre os pontos de visualização O6 e O7 inicia-se na junta R4

$\vec{V}_8 = [\ 0\ ,\ 0\ ,\ d61\ ] - R8$ entre os pontos de visualização O7 e O8 inicia-se na junta R5

$\vec{V}_9 = [\ 0\ ,\ 0\ ,\ d62\ ] - R9$ entre os pontos de visualização O8 e O9 inicia-se na junta R6

Com o objetivo de obter a visualização do robô em vários ângulos pelo software de programação, define-se:

M_V - matriz de Euler com as variáveis do software que definem o ângulo de visualização gráfica.

Serão inseridos no modelo uma matriz e um vetor de correção de erros identificados representados por:

Rt_a - matriz de correção de erros de orientação identificados,

$\vec{V}$ t_a - vetor de correção de erros de posição identificados.

O objetivo da inserção da matriz de correção de erros de orientação e do vetor de correção de erros de posição, identificados, é de permitir posicionar o meio ambiente gerado no computador (programação "off-line") da mesma forma que ele estará posicionado quando da colocação do robô no meio ambiente. Isto permite um alto grau de precisão de posicionamento e orientação. Em Campos (1993), foi implementado um método para a identificação desses parâmetros.

Neste estudo não preocupou-se em inserir valores medidos para estes parâmetros, pois não está dentro do escopo deste trabalho. Portanto, para não interferir com o modelo, a matriz de correção de erros de orientação e o vetor de correção de erros de posição, identificados, são dados por:

$$Rt\_a = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.7}$$

$$\vec{V}t\_a = [0 \quad 0 \quad 0] \tag{4.8}$$

Definindo-se então as matrizes de transformação M_T(i) associadas a cada elo, tem-se:

$$\begin{aligned} M\_T(1) &= M\_V * Rt\_a \\ M\_T(2) &= M\_T(1) * R_1 \\ M\_T(3) &= M\_T(2) * R_2 = M\_T(1) * R_1 * R_2 \\ M\_T(4) &= M\_T(3) * R_3 = M\_T(1) * R_1 * R_2 * R_3 \\ M\_T(5) &= M\_T(4) * R_4 = M\_T(1) * R_1 * R_2 * R_3 * R_4 \\ M\_T(6) &= M\_T(5) * R_5 = M\_T(1) * R_1 * R_2 * R_3 * R_4 * R_5 \\ M\_T(7) &= M\_T(6) * R_6 = M\_T(1) * R_1 * R_2 * R_3 * R_4 * R_5 * R_6 \end{aligned} \tag{4.9}$$

Por fim, determina-se a posição dos pontos de interesse, em relação ao referencial XYZ, as equações são apresentadas abaixo:

$$
\begin{aligned}
&\vec{O}(0) = M\_T(1) * \vec{V}t\_a \\
&\vec{O}(1) = \vec{O}(0) + M\_T(2) * \vec{V}_1 \\
&\vec{O}(2) = \vec{O}(1) + M\_T(2) * \vec{V}_2 \\
&\vec{O}(3) = \vec{O}(1) + M\_T(3) * \vec{V}_3 \\
&\vec{O}(4) = \vec{O}(1) + M\_T(3) * \vec{V}_4 \\
&\vec{O}(5) = \vec{O}(4) + M\_T(4) * \vec{V}_5 \\
&\vec{O}(6) = \vec{O}(4) + M\_T(4) * \vec{V}_6 \\
&\vec{O}(7) = \vec{O}(6) + M\_T(5) * \vec{V}_7 \\
&\vec{O}(8) = \vec{O}(7) + M\_T(6) * \vec{V}_8 \\
&\vec{O}(9) = \vec{O}(8) + M\_T(7) * \vec{V}_9
\end{aligned}
\tag{4.10}
$$

4.3 Comparação do modelo geométrico obtido através dos dois métodos

Montando-se a matriz:

$$
C_{4x4} = \begin{bmatrix} M\_T(7)_{3x3} & \vec{O}(9)_{3x1} \\ \vec{0}_{1x3} & 1 \end{bmatrix} \tag{4.11}
$$

pode-se confirmar se os valores obtidos para a posição e orientação do elemento terminal do robô, são os mesmos em ambos os métodos. A matriz gerada pela equação 4.11 tem de ser igual a da equação 4.3.

Na tabela 4.4 são apresentados os valores angulares das juntas para os quais foram realizadas as simulações para comparar o modelo geométrico obtido através dos dois métodos.

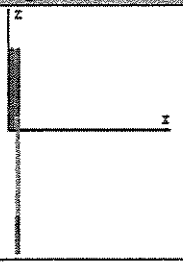
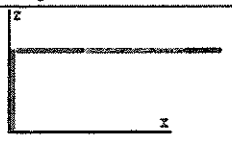
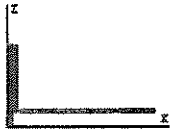
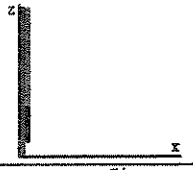
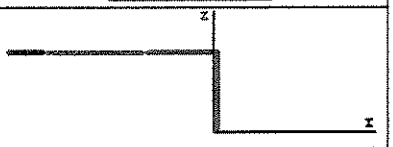
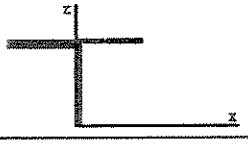
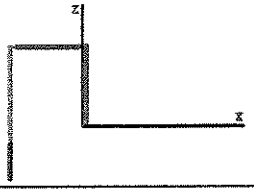
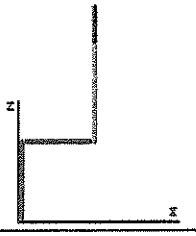
	Posição angular						
Simulação.	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	Configuração do robô
1	0.0	90.0	90.0	0.0	0.0	0.0	
2	0.0	0.0	90	0.0	0.0	0.0	
3	0.0	90.0	0.0	0.0	0.0	0.0	
4	0.0	-90.0	-90.0	0.0	0.0	0.0	
5	0.0	-180.0	90.0	0.0	0.0	0.0	
6	0.0	-180.0	-90.0	0.0	0.0	0.0	
7	0	-180.0	0.0	0.0	0.0	0.0	
8	0.0	0.0	0.0	0.0	0.0	0.0	

Tabela 4.4: Valores angulares das juntas, para cada configuração, para comparar o modelo geométrico obtido através dos dois métodos.

Os valores obtidos, utilizando-se os dois métodos, para a posição e orientação foram os mesmos (tabela 4.5).

Simulação	Matriz obtida	Simulação	Matriz obtida
1	$\begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & -665 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	5	$\begin{bmatrix} 0 & 0 & -1 & -1330 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 665 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
2	$\begin{bmatrix} 0 & 0 & 1 & 1330 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 665 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	6	$\begin{bmatrix} 0 & 0 & 1 & 330 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 665 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
3	$\begin{bmatrix} 0 & 0 & 1 & 830 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 165 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	7	$\begin{bmatrix} -1 & 0 & 0 & -500 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & -165 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
4	$\begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 335 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	8	$\begin{bmatrix} 1 & 0 & 0 & 500 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1495 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Tabela 4.5: Valores obtidos para a posição e orientação a para cada simulação apresentada na tabela 4.4.

4.4 Modelo geométrico inverso

A coordenação de movimentos, que consiste na obtenção de um movimento de referência (angular) para cada junta, para um dado movimento de referência do elemento terminal (cartesiano), é expressa matematicamente pela inversão do modelo geométrico.

A função $\vec{f}$ é não linear e composta de soma de produtos das coordenadas generalizadas das ligações de translação e de somas e produtos de senos e cosenos das coordenadas generalizadas das ligações de rotação. Por isso, a sua inversão é, em geral, não trivial.

A seguir serão apresentados os métodos (Craig, 1986; Paul, 1981) para a inversão do modelo geométrico.

4.4.1 Métodos analíticos

Estes métodos conduzem à obtenção de todas as soluções. Estes métodos não são gerais, isto é, a inversão analítica não é trivial e, além disso, não há garantia de que seja possível fazê-la para um robô qualquer. Os métodos analíticos são adequados para robôs simples, isto é, aqueles que possuem um grande número de parâmetros de Denavit-Hartenberg nulos (Ferreira 1991).

Estes métodos, às vezes, conduzem a soluções múltiplas como pode ser observado na figura 4.12.

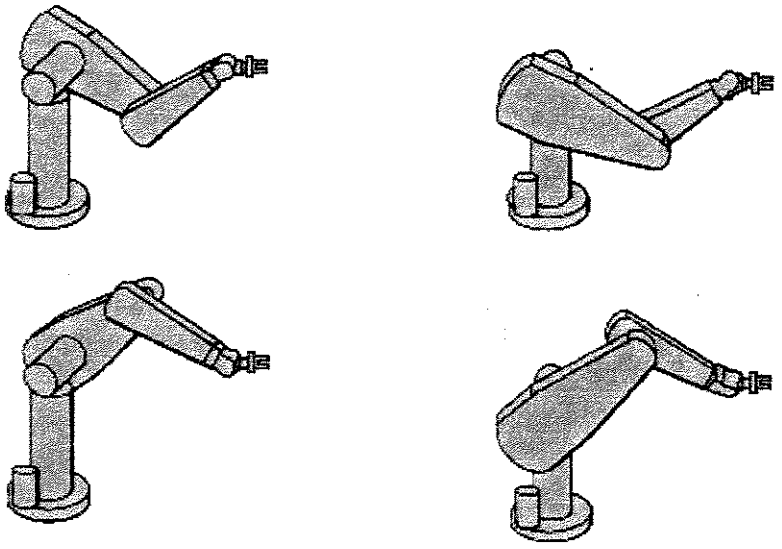


Figura 4.12: Configurações múltiplas para um robô.

4.4.2 Métodos numéricos iterativos

Estes métodos convergem para uma solução possível entre todas as existentes; são de caráter geral e, com o atual desenvolvimento dos microcomputadores, a utilização desses métodos em tempo real é viável. Diferentemente das soluções analíticas, as soluções numéricas

podem ser combinadas com estratégias de anticolisão com objetos, onde as posições e as dimensões dos mesmos são conhecidas.

O interessante nestes métodos é que os pontos gerados a cada iteração podem ser enviados diretamente para o controlador do robô como um ponto de referência.

Existem diversos métodos numéricos iterativos, entre eles o *método recursivo* (Gorla, 1984), figura 4.13, que utiliza ao cálculo do modelo geométrico direto e da matriz Jacobiana inversa.

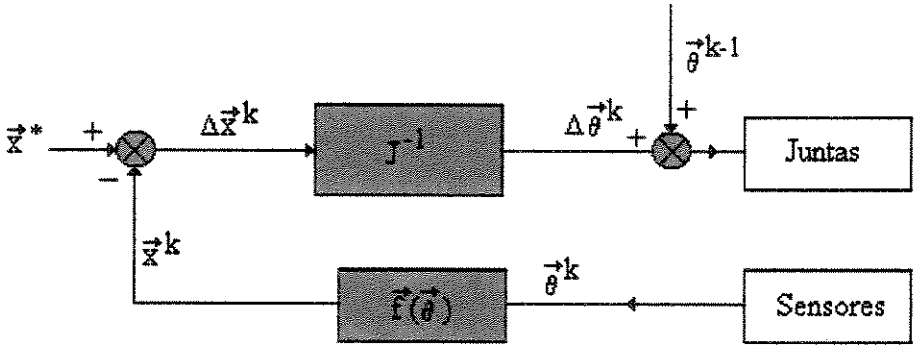


Figura 4.13: Utilização do modelo geométrico direto e da matriz Jacobiana inversa para determinar uma configuração $\vec{\theta}^*$ correspondente a uma situação desejada $\vec{x}^*$.

4.5 Descrição da matriz de orientação através de ângulos

O uso dos vetores $\vec{n}$, $\vec{s}$ e $\vec{a}$ como vetores de direção, para descrever a orientação do elemento terminal do robô, é de difícil compreensão.

Portando torna-se necessário descrever a matriz de orientação, composta pelos componentes dos vetores $\vec{n}$, $\vec{s}$ e $\vec{a}$ em termos de três ângulos: ψ, θ e ϕ. Isto implicará que o posicionamento do elemento terminal do robô será descrito por um vetor de seis dimensões. Esta representação torna mais simples a utilização da posição e orientação do robô para a geração de uma trajetória.

Uma técnica para descrever a orientação de um robô a partir de três ângulos, envolve o uso da representação Roll, Pitch e Yaw do punho. Estes são os três graus de liberdade possíveis associados ao movimento do punho. São respectivamente os ângulos de rotação do punho ao redor dos eixos X, Y e Z, figura 4.14.

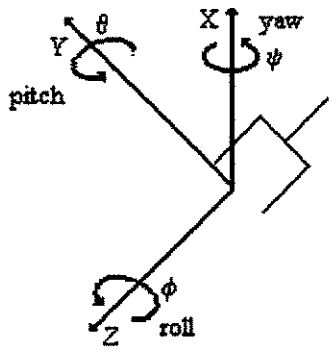


Figura 4.14: Ângulos de rotação Roll, Pithc e Yaw.

4.5.1 Obtenção da matriz RPY a partir da definição de três ângulos

Muitos robôs, entre eles o Manutec, utilizam os ângulos Roll, Pitch e Yaw para expressarem a orientação do elemento terminal em relação às coordenadas da base. Para isto basta efetuar três rotações sucessivas em relação aos eixos X, Y e Z, obtendo-se uma matriz correspondente a essa transformação de coordenadas.

Começando com os dois sistemas de coordenadas A e B coincidentes, inicia-se fazendo-se uma rotação no sistema de coordenadas B sobre $\hat{X}_A$ de um ângulo ψ, depois sobre $\hat{Y}_A$ de um ângulo θ e então sobre $\hat{Z}_A$ de um ângulo ϕ, figura 4.15.

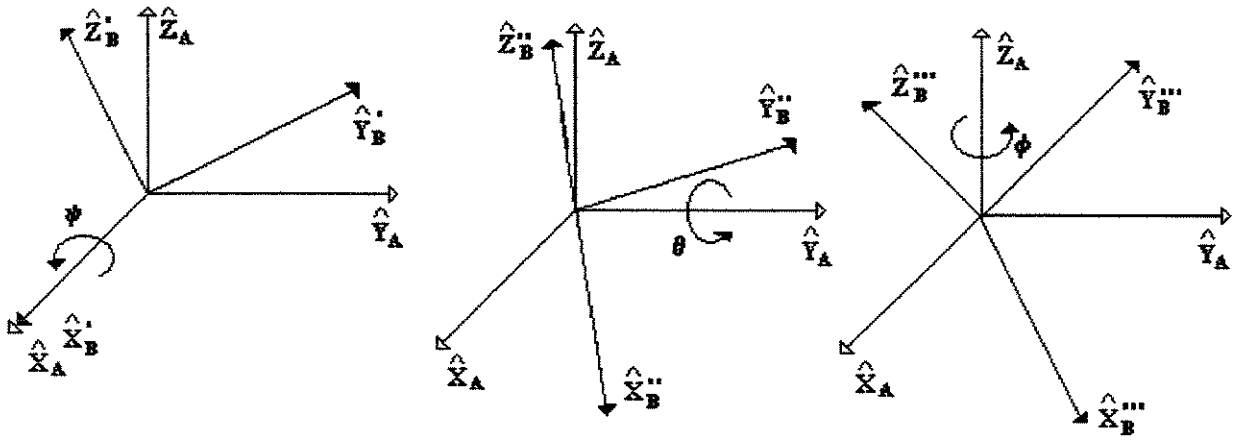


Figura 4.15: Rotações dos eixos do sistema de coordenadas B.

Portanto a matriz de orientação é dada por:

$$\text{RPY}(\psi, \theta, \phi) = \text{rot}(z, \phi) * \text{rot}(y, \theta) * \text{rot}(z, \psi) \tag{4.12}$$

$$\text{RPY}(\psi, \theta, \phi) = \begin{bmatrix} c\phi\, c\theta & c\phi\, s\theta\, s\psi - s\phi\, c\psi & c\phi\, s\theta\, c\psi + s\phi\, s\psi \\ s\phi\, c\theta & s\phi\, s\theta\, s\psi + c\phi\, c\psi & s\phi\, s\theta\, c\psi - c\phi\, s\psi \\ -s\theta & c\theta\, s\psi & c\theta\, c\psi \end{bmatrix} \tag{4.13}$$

4.5.2 Obtenção dos três ângulos a partir da matriz orientação

Para se obter os três ângulos a partir da matriz de orientação, o problema inverso, analisa-se os seus termos. Os casos possíveis, a partir desta observação, serão descritos a seguir.

Escrevendo a matriz de orientação da seguinte forma:

$$\begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{31} & m_{33} \end{bmatrix} \tag{4.14}$$

tem-se:

- caso 1

- se $m_{31} = -1 \quad \Rightarrow \quad \theta = 90°$.

- se $m_{31} = 1 \quad \Rightarrow \quad \theta = -90°$.

- casos 2 a 4

- se $m_{21} < 0$

caso 2 $\Rightarrow$ $\phi = -90°$, $\psi = -90°$ e

$\theta = \text{atan2}(-m_{31}, -m_{21})$; (4.15)

caso 3 $\Rightarrow$ $\phi = -90°$, $\psi = 90°$ e

$\theta = \text{atan2}(-m_{31}, -m_{21})$; (4.16)

caso 4 $\Rightarrow$ $\phi = -90°$,

$\psi = \text{atan2}(-m_{13}, m_{12})$ e (4.17)

$\theta = \text{atan2}(-m_{31}, -m_{21})$. (4.18)

- se $m_{21} > 0$

caso 2 $\Rightarrow$ $\phi = 90°$, $\psi = 90°$ e

$\theta = \text{atan2}(-m_{31}, m_{21})$; (4.19)

caso 3 $\Rightarrow$ $\phi = 90°$, $\psi = -90°$ e

$\theta = \text{atan2}(-m_{31}, m_{21})$; (4.20)

caso 4 $\Rightarrow$ $\phi = 90°$,

$\psi = \text{atan2}(m_{13}, -m_{12})$ e (4.21)

$\theta = \text{atan2}(-m_{31}, m_{21})$. (4.22)

- caso 5

- se $\text{abs}(m_{33}) > 0$ e $\text{abs}(m_{32}) > 0$

$\Rightarrow \quad \phi = \text{atan2}(m_{21}, -m_{11})$, (4.23)

$$\psi = \text{atan2}\,(m_{32}, -m_{33}) \text{ e} \tag{4.24}$$

$$\theta = \text{atan2}\,(-m_{31}, \text{sen}(\psi)). \tag{4.25}$$

- se $m_{33} = 0$ e $m_{32} > 0$

$$\Rightarrow \quad \phi = \text{atan2}\,(m_{13}, -m_{23}), \tag{4.26}$$

$$\psi = 90^{\circ} \text{ e}$$

$$\theta = \text{atan2}\,(-m_{31}, m_{32}). \tag{4.27}$$

- se $m_{33} = 0$ e $m_{32} < 0$

$$\Rightarrow \quad \phi = \text{atan2}\,(-m_{13}, m_{23}), \tag{4.28}$$

$$\psi = -90^{\circ} \text{ e}$$

$$\theta = \text{atan2}\,(-m_{31}, -m_{32}). \tag{4.29}$$

- se $m_{32} = 0$

$$\Rightarrow \quad \phi = \text{atan2}\,(-m_{12}, m_{22}), \tag{4.30}$$

$$\psi = 0^{\circ} \text{ e}$$

$$\theta = \text{atan2}\,(-m_{31}, m_{33}). \tag{4.31}$$

No Anexo A é apresentado o algoritmo para a implementação destes casos. A função Atan2* é utilizada com a finalidade de localizar o quadrante correto do ângulo.

4.6 Matriz Jacobiana

A matriz Jacobiana, como pode ser observado, intervém na solução numérica da inversão do modelo geométrico (método recursivo) e, portanto, nas soluções de controle implementado diretamente no espaço operacional, como pode ser observado na figura 4.13.

---

* A função Atan2 (y , x) calcula $\tan^{-1}$ (y/x) mas usa o sinal de ambos (x e y) para determinar o quadrante correto no qual o ângulo esta. Por exemplo Atan2(-2.0, -2.0) = -135.0° e Atan2 (2.0, 2.0) = 45.0°. Se x e y são positivos implica que $0 \leq \theta \leq 90$, se x é negativo e y positivo então $90 \leq \theta \leq 180$, se x e y são negativos então $-180 \leq \theta \leq -90$ e se x é positivo e y negativo tem-se $-90 \leq \theta \leq 0$.

Ela é uma forma multidimensional da derivada e relaciona a velocidade no espaço de juntas à velocidade no espaço cartesiano, isto é:

$$\Delta \vec{x} = J \, \Delta \vec{\theta} \qquad (4.32)$$

Uma técnica para determinar o Jacobiano de um robô de seis graus de liberdade é fazendo uso das matrizes de transformações que definem a geometria do robô. Esta técnica é apresentada no Anexo II.

4.6.1 Matriz Jacobiana inversa

No método recursivo, figura 4.13, existe a necessidade da inversão da matriz Jacobiana para se poder obter o conjunto de ângulos (referencias angulares) para um determinado trabalho, definido no espaço cartesiano. A matriz Jacobiana inversa pode ser obtida através de dois métodos:

- Inversão simbólica: se deixarmos a matriz Jacobiana em sua forma original, forma simbólica, pode-se encontrar a inversa, usando a álgebra. Mas a complexidade do Jacobiano torna este procedimento muito difícil ou até mesmo impossível.

- Inversão numérica: para cada instante, a configuração do robô define um conjunto de variáveis de juntas, deste modo, então, a matriz Jacobiana, em cada instante, é uma matriz numérica. A literatura em análise numérica apresenta diversas técnicas para a inversão de matrizes. Tais técnicas não devem apenas serem rápidas, mas também, retornarem respostas precisas.

Como está sendo utilizado o método recursivo neste trabalho, para a implementação de um programa de geração de trajetórias, existe a necessidade da utilização de um método numérico para a inversão da matriz Jacobiana

Existem diversos métodos para a inversão da matriz Jacobiana. Entre eles podemos citar os métodos de Gauss (Kreysizig, 1983), utilizado para o cálculo da matriz inversa de J e Greville

(Gorla, 1984), utilizado para o cálculo da matriz pseudoinversa de J. Estes métodos foram estudados e implementados em Sá (1996).

Estes métodos são ditos métodos diretos e, em princípio, produzirão uma solução exata (desprezando os erros de arredondamento), se houver, em um número finito de operações. Entretanto, as soluções podem perder o significado, caso o sistema linear seja mal-condicionado.

O método de Gauss possui o inconveniente de que, em muitos casos a solução de um sistema linear existe, mesmo quando a inversa da matriz não existe e também devido ao fato do alto número de operações aritméticas necessárias. Já o método de Greville, pseudoinversa (Nashed, 1976), não apresenta esses inconvenientes, motivo pelo qual ele será utilizado. Este método é apresentado no Anexo III.

Pode-se notar que o modelo geométrico do robô é de grande importância pois, através dele, pode-se calcular a matriz Jacobiana do robô e, além disso, ele é utilizado diretamente na malha de controle para a geração de uma trajetória ponto a ponto em tempo real.

No capítulo seguinte será apresentado um algoritmo de geração de trajetórias para um robô, utilizando o método recursivo para a inversão do modelo cinemático obtido através de Denavit-Hartenberg. Também será apresentado um algoritmo para a geração de trajetórias através de interpolação e filtragem (de pontos de passagem) uma vez que em muitos casos é necessária a geração de trajetórias a partir de determinados pontos de passagem angulares obtidos diretamente das juntas do robô.

Capítulo 5

Geração de Trajetórias e Controle de Movimentos

Como mencionado anteriormente, em muitas aplicações a programação de tarefas de robôs é realizada no espaço das juntas e a trajetória angular, de mesma natureza dos sinais provenientes do transdutor de posição, servirá como sinal de referência para o controlador de cada junta robótica, após interpolação.

Entretanto, na maioria das aplicações, a realização de tarefas está relacionada com o tipo de ferramenta utilizada, a partir de um sistema de coordenadas cartesianas fixo à base do robô, espaço cartesiano. Portanto, os movimentos desejados e as leis de controle estão em espaços diferentes, isto pode ser observado na figura 5.1.

Nesta figura pode-se observar que a partir da comparação de $\vec{X}_d$, posição e orientação desejadas, e $\vec{X}_{ref}$, referência, um sinal de erro é gerado.e transposto em termos de erros de coordenadas articulares $\Delta\theta_i$ as quais serão ser utilizadas como sinal de referência para o controle das juntas do robô

A parte envolta por linhas pontilhadas representa o controle de trajetórias a nível de juntas, mostrando deste modo que o sistema suporta a concepção de trajetórias também no espaço das juntas

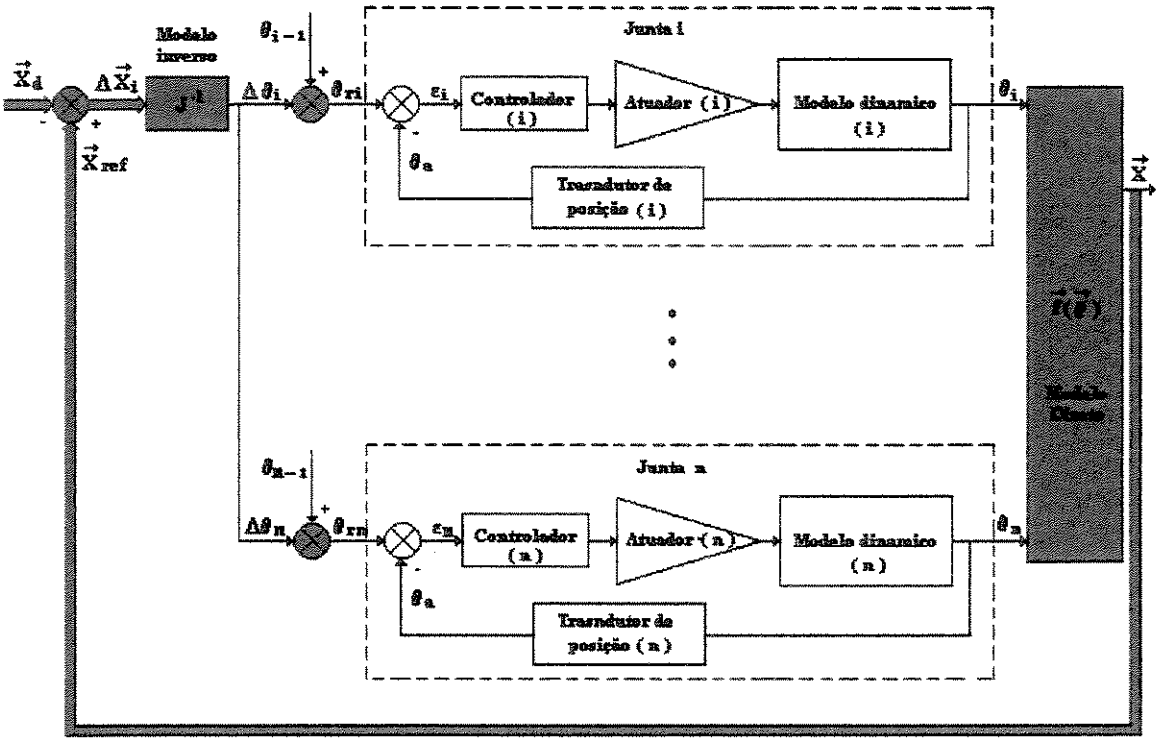


Figura 5.1: Malha de controle de um robô.

Este capítulo apresenta a implementação de um algoritmo de geração de trajetórias, no espaço cartesiano, para um robô e de um algoritmo de interpolação e filtragem. Serão apresentados também os resultados iniciais para validar os algoritmos.

5.1 Algoritmo para a geração de uma trajetória angular, espaço cartesiano

Observando a malha de controle, figura 5.1, para a implementação de um algoritmo de geração de trajetórias, no espaço cartesiano, é necessária a inversão do modelo geométrico, que neste caso, pode ser obtido através do método de Denavit-Hartenberg, pois são necessárias apenas as coordenadas do elemento terminal do robô.

O algoritmo também deverá calcular a matriz Jacobiana do sistema a cada iteração e parar as iterações sempre que o erro máximo permitido para a posição e orientação for alcançado ou quando o número máximo de iterações for alcançado. Este algoritmo permitirá que o elemento terminal do robô siga uma trajetória entre dois pontos bem definida. O algoritmo proposto para a geração de trajetórias é apresentado na figura 5.2.

O algoritmo permite que a trajetória gerada seja composta por diversos segmentos. Isto permite, por exemplo, que em uma parte da trajetória o robô siga uma linha reta e na outra uma curva qualquer.

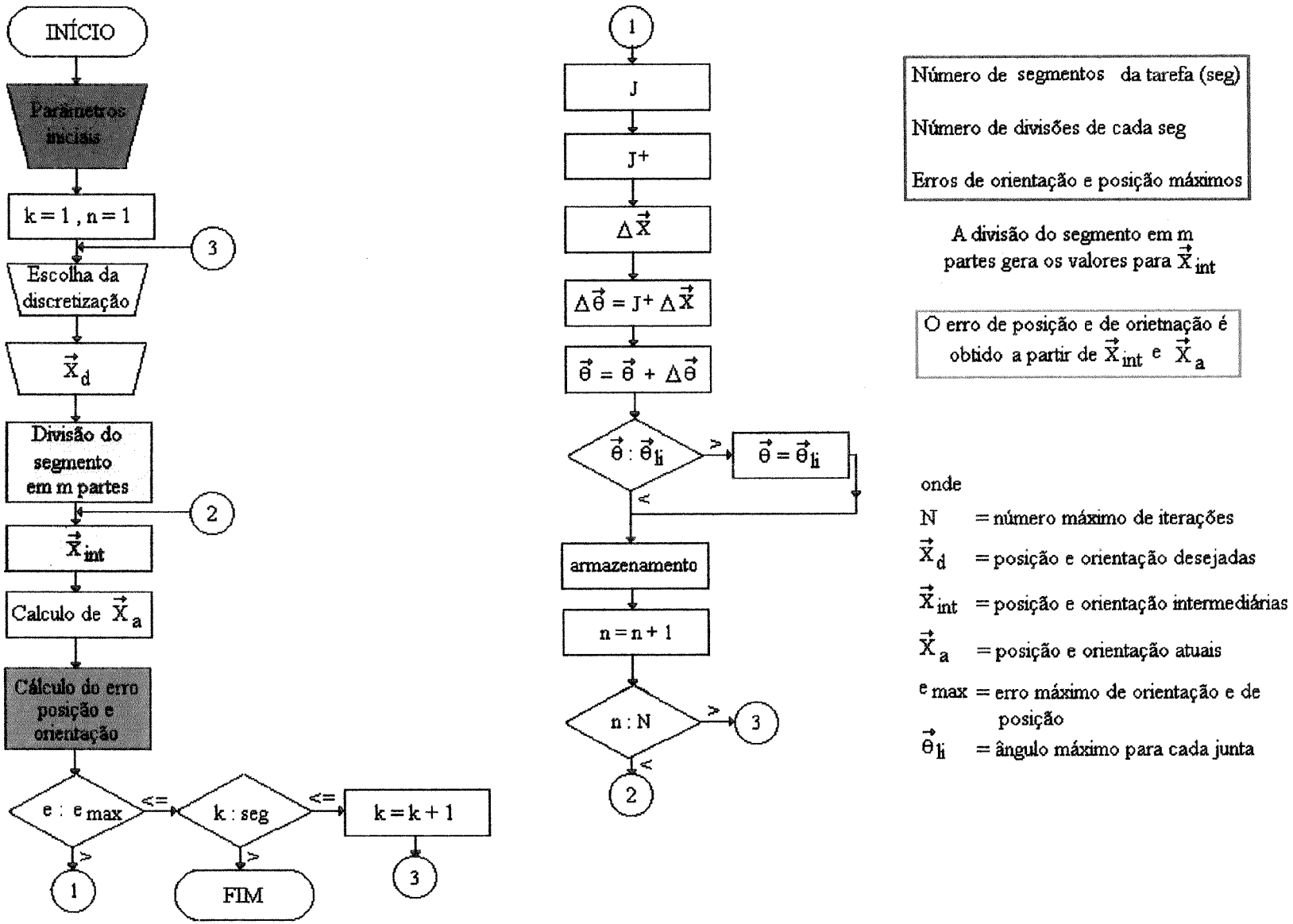


Figura 5.2: Algoritmo para a geração de uma trajetória.

No algoritmo proposto em cada segmento da trajetória existe a necessidade da escolha da discretização, isto permite que o elemento terminal do robô siga um caminho pré-determinado. Os tipos de discretização implementados serão apresentados no próximo tópico.

Um algoritmo simplificado é apresentado na figura 5.3.

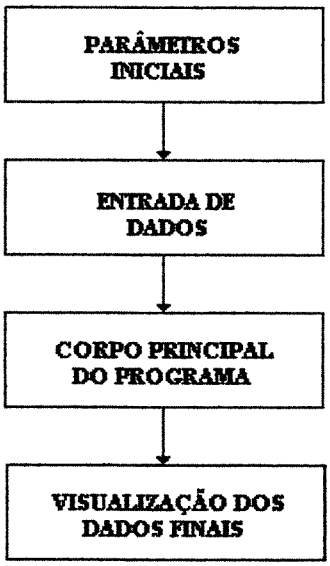


Figura 5.3: Algoritmo simplificado para a geração de trajetórias.

No bloco denominado P*arâmetros iniciais* é feita a escolha da discretização, escolhido o número de segmentos da tarefa a ser realizada (*seg*), o número de divisões de cada segmento (*m*) e determinados os erros máximos de posição e orientação permitidos.

No bloco denominado E*ntrada de dados* é dado o valor da posição **(em mm)** e orientação **(em termos dos ângulos RPY)** finais desejadas para cada segmento da tarefa a ser realizada. A posição e orientação finais de cada segmento tornam-se a inicial para o outro segmento. Nesta fase é permitida a alteração do tipo de discretização a ser aplicada ao segmento posterior.

No bloco denominado *Corpo principal do programa* é realizada a divisão do segmento em *m* pontos utilizando a discretização escolhida. Cada ponto passa a ser uma posição e orientação finais desejadas. É realizado o cálculo do erro de posição e orientação. Neste ponto o algoritmo pode: armazenar o conjunto de ângulos para a posição e orientação atual, caso os erros sejam menores que os desejados, ou inicia-se o cálculo dos incrementos das juntas. Neste ponto, são

calculados J, J^{+}, $\Delta\vec{X}$, $\Delta\vec{\theta}$ e $\vec{\theta}$. Verifica-se se cada ângulo de junta está dentro do limite físico permitido para a mesma. Se não estiver, atribui-se a esta junta o valor máximo permitido e, então, armazena-se o conjunto de ângulos obtidos.

Este procedimento é realizado até que o valor de *m* seja alcançado e após isso, inicia-se o processo para o próximo segmento, até que o valor dado para *seg* seja atingido.

No bloco denominado *Dados finais* é obtido o conjunto de ângulos (trajetória gerada) em arquivo ASCII.

5.1.1 Discretização do caminho

Com o objetivo de fazer com que o elemento terminal siga um caminho pré-determinado e bem definido entre a posição e orientação iniciais e a posição e orientação finais desejadas no espaço (por exemplo um círculo ou outro caminho qualquer), o caminho é discretizado em *m* partes, utilizando equações que gerem o caminho desejado.

A figura 5.4 (a) mostra a discretização entre o ponto inicial da trajetória $\vec{x}$ e o ponto final $\vec{x}_d$ para que o elemento terminal do robô siga uma reta. A figura 5.4 (b) mostra a discretização entre o ponto inicial da trajetória $\vec{x}$ e o ponto final $\vec{x}_d$ para que o elemento terminal do robô siga um semicírculo.

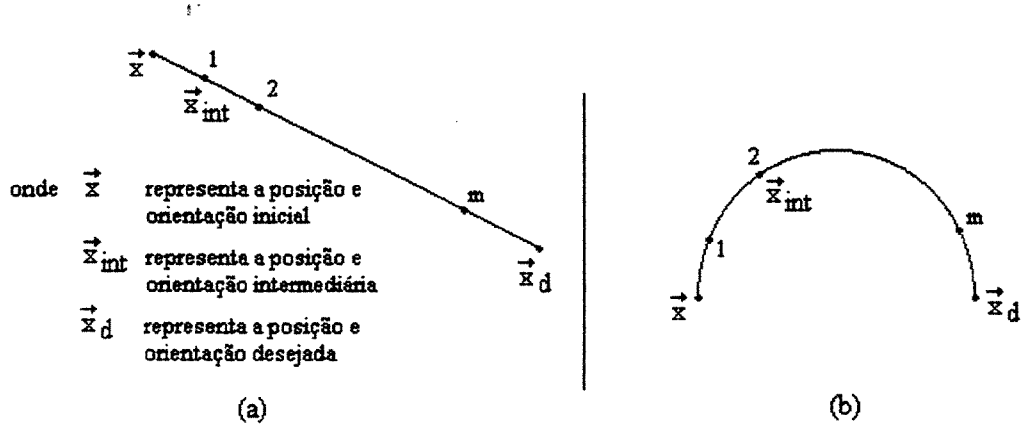


Figura 5.4: Discretização do caminho em *m* partes, para o robô seguir uma reta (a) e para seguir um semicírculo (b).

Deste modo então, o algoritmo de geração de trajetórias calcula a trajetória da posição inicial até a posição intermediária e, após ela ser atingida, inicia-se o cálculo da trajetória desta posição até a posição intermediária posterior. Isto é realizado até que a posição final desejada seja alcançada.

Para este trabalho foram implementados dois tipos de discretização: Linear e em Semicírculo. A princípio foram implementados apenas estes dois, com o objetivo de ver os resultados obtidos, mas não impede que outros tipos possam ser implementados.

A **orientação*** do robô é discretizada de forma linear mesmo quando utiliza-se a discretização em Semicírculo.

Portando, quando se discretiza o caminho em forma de um semicírculo, estamos discretizando apenas entre a posição inicial e final.

5.1.1.1 Discretização linear

Discretiza o caminho desejado, em *m* partes, de forma linear, fazendo desta forma com que o elemento terminal do robô siga uma linha reta.

Seja (X_1, Y_1, Z_1) o vetor posição inicial do elemento terminal do robô e (X_2, Y_2, Z_2) o seu vetor posição final.

Para a discretizar a posição tem-se as seguintes equações:

---

* Para os cálculos internos do programa é utilizada a matriz $[\vec{n}, \vec{s}, \vec{a}]$ uma vez que o modelo geométrico obtido através de Denavit-Hartenberg fornece esta matriz mas, para melhor compreensão dos resultados finais obtidos, apresentados ao usuário, a matriz $[\vec{n}, \vec{s}, \vec{a}]$ será descrita em termos dos ângulos ψ, θ e ϕ (RPY).

$$
\begin{aligned}
x_i &= x_{i-1} + Inc_1 \\
y_i &= y_{i-1} + Inc_2 \\
z_i &= z_{i-1} + Inc_3
\end{aligned}
\qquad (5.1)
$$

onde

$i = 1,2, \ldots\ldots, m;$

x_i, y_i e z_i = pontos intermediários do vetor posição;

$Inc_1 = (X_1 - X_2) / m;$

$Inc_2 = (Y_1 - Y_2) / m;$

$Inc_3 = (Z_1 - Z_2) / m;$

e m é o número de partes em que o caminho será dividido.

O vetor (x_i , y_i , z_i) é denominado vetor posição intermediário.

Seja $[\vec{n}_1 , \vec{s}_1 , \vec{a}_1]$ a matriz orientação inicial do elemento terminal do robô e $[\vec{n}_2 , \vec{s}_2 , \vec{a}_2]$ a sua matriz orientação final. Para discretizar a orientação tem-se:

$$
\begin{aligned}
\vec{n}_i &= \vec{n}_{i-1} + Inc_4 \\
\vec{s}_i &= \vec{s}_{i-1} + Inc_5 \\
\vec{a}_i &= \vec{a}_{i-1} + Inc_6
\end{aligned}
\qquad (5.2)
$$

onde

$i = 1,2, \ldots\ldots, m;$

$\vec{n}_i$, $\vec{s}_i$ e $\vec{a}_i$ = pontos intermediários do vetor orientação;

$Inc_4 = (\vec{n}_1 - \vec{n}_2) / m;$

$Inc_5 = (\vec{s}_1 - \vec{s}_2) / m;$

$Inc_6 = (\vec{a}_1 - \vec{a}_2) / m.$

A matriz $[\vec{n}_1 , \vec{s}_1 , \vec{a}_1]$ é denominada matriz de orientação intermediária.

5.1.1.2 Discretização em semicírculo

Discretiza o caminho desejado em forma de um semicírculo. A equação de um círculo é dado por:

$$(x - h)^2 + (y - k)^2 = r^2 \tag{5.3}$$

Os tipos de discretizações possíveis em semicírculo são:

- no plano X-Y podendo variar o valor da posição Z e com Y = f(X).
- no plano X-Z podendo variar o valor da posição Y e com Z = f(X).
- no plano Y-Z podendo variar o valor da posição X e com Y = f(Z).

Seja (X_1, Y_1, Z_1) o vetor posição inicial do elemento terminal do robô e (X_2, Y_2, Z_2) o seu vetor posição final.

Temos então para a discretização em semicírculo, entre a posição inicial e a posição final, as seguintes equações:

$$\begin{aligned} y_i &= (r_1^2 - (c_1 - h_1)^2)^{1/2} + k_1 && \text{semicírculo no plano x-y} \\ z_i &= (r_2^2 - (c_2 - h_2)^2)^{1/2} + k_2 && \text{semicírculo no plano x-z} \\ z_i &= (r_3^2 - (c_3 - h_3)^2)^{1/2} + k_3 && \text{semicírculo no plano y-z} \end{aligned} \tag{5.4}$$

onde

$i = 1,2, ..., m;$

$r_1 = abs(X_2 - X_1) / 2;$

$r_2 = abs(X_2 - X_1) / 2;$

$r_3 = abs(Y_2 - Y_1) / 2;$

$h_1 = X_1$;

$h_2 = X_1$;

$h_1 = Y_1$;

$k_1 = Y_1$;

$k_2 = Z_1$;

$k_3 = Z_1$;

$c_1 = x_{1-1} + Inc_1$ onde $Inc_1 = (X_1 - X_2) / m$;

$c_2 = y_{1-1} + Inc_2$ onde $Inc_2 = (X_1 - X_2) / m$;

$c_3 = y_{1-1} + Inc_3$ onde $Inc_3 = (Y_1 - Y_2) / m$;

e *m* é o número de partes em que o caminho será dividido.

A figura 5.5 mostra como exemplo o semicírculo no plano x-y.

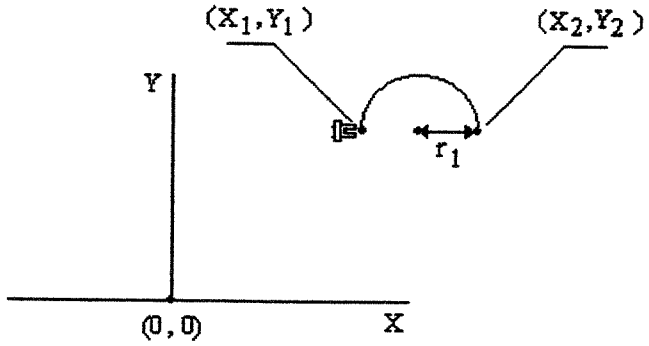


Figura 5.5: Discretização em semicírculo no plano x-y.

Os semicírculos podem apresentar as configurações indicadas na tabela 5.1. Estas configurações são alcançadas adicionando-se nas equações dos semicírculos uma variável que pode ter o valor +1 (direção positiva) ou –1 (direção negativa). As equações tornam-se então

$$\begin{aligned} y_i &= ((r_1^2 - (c_1 - h_1)^2)^{1/2}) * \text{Direção} + k_1 && \text{semicírculo no plano x-y} \\ z_i &= ((r_2^2 - (c_2 - h_2)^2)^{1/2}) * \text{Direção} + k_2 && \text{semicírculo no plano x-z} \\ z_i &= ((r_3^2 - (c_3 - h_3)^2)^{1/2}) * \text{Direção} + k_3 && \text{semicírculo no plano y-z} \end{aligned} \qquad (5.5)$$

onde Direção pode ter o valor de +1 ou –1.

O vetor (x_i , y_i , z_i) é denominado vetor posição intermediário.

A orientação, como anteriormente mencionada, é discretizada de forma linear utilizando, deste modo, as equações 5.2.

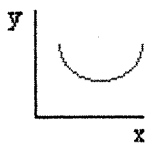
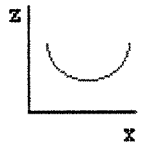
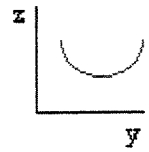
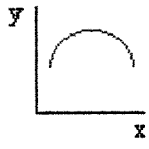
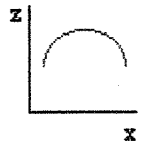
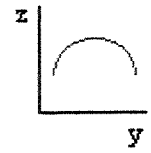
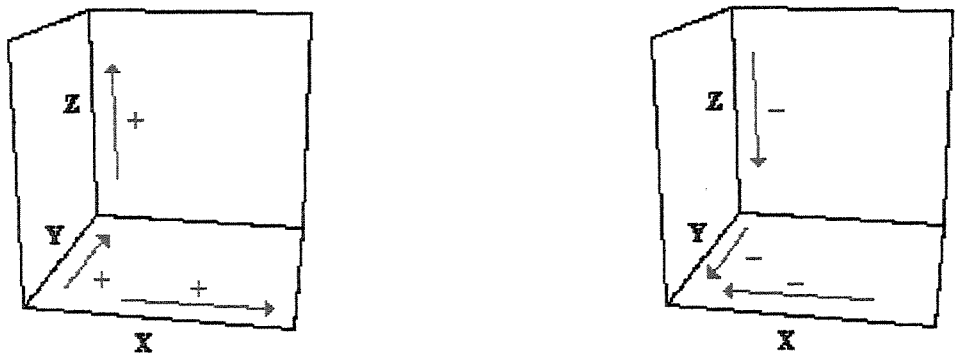
Direção	Plano x-y	Plano x-z	Plano y-z
positiva (semicírculo abre-se na direção crescente do eixo)			
negativa (semicírculo abre-se na direção decrescente do eixo)			

Tabela 5.1: Configurações possíveis dos semicírculos.

A figura 5.4 (a) mostra o sentido crescente de cada eixo e a figura 5.5 (b) mostra o sentido decrescente de cada eixo.



(a) (b)

Figura 5.6: Sentido crescente (a) e decrescente (b).

5.1.2 Resultados iniciais obtidos

Com o objetivo de testar o algoritmo de geração de trajetórias, foram realizadas as simulações apresentadas na tabela 5.2. Esta tabela contém a posição e orientação finais desejadas para o elemento terminal. A posição inicial do robô é dada por (500.0, 0.0, 1495.0) e a orientação inicial é dada por (0.0, 0.0, 0.0), para todas as simulações, que corresponde a uma posição angular das juntas, em graus, dada por (0.0, 0.0, 0.0, 0.0, 0.0, 0.0). As trajetórias angulares foram obtidas utilizando-se o algoritmo descrito acima.

A seguir são apresentados os gráficos da evolução angular das juntas e da trajetória espacial seguida pelo elemento terminal para cada simulação.

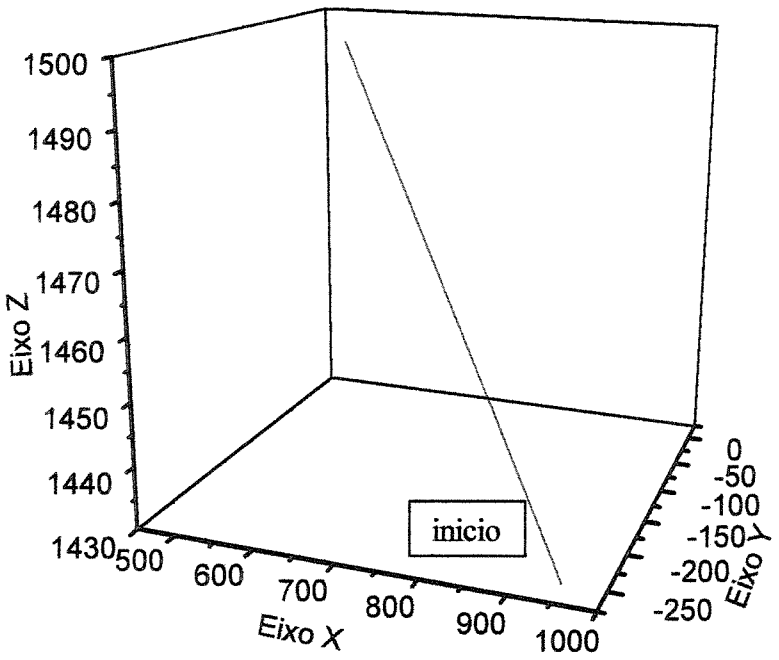
	Posição (em mm) e orientação (em graus) finais						Trajetória seguida pelo e. terminal.	Direção
Simulação	X	Y	Z	Psi	Teta	Phi		
1	950.0	-250.0	1430.0	0.0	25.0	30.0	linear	
2	850.0	y = f(x)	fixo	10.0	0.0	30.0	semicírculo (x-y)	+
3	850.0	fixo	z = f(x)	10.0	0.0	30.0	semicírculo (x-z)	+
4	fixo	-400.0	z = f(y)	0.0	25.0	35.0	semicírculo (y-z)	+
5	850.0	y = f(x)	1200.0	10.0	0.0	30.0	semicírculo (x-y)	+
6	850.0	300.0	z = f(x)	10.0	0.0	30.0	semicírculo (x-z)	+
7	50.0	-400.0	z = f(y)	10.0	0.0	30.0	semicírculo (y-z)	+
8	700.0	y = f(x)	1600.0	0.0	0.0	0.0	círculo (x-y)	+
	500.0	y = f(x)	1495.0	0.0	0.0	0.0		-
9	600.0	100.0	1495.0	15.0	12.0	20.0	linear	
	600.0	100.0	1700.0	15.0	12.0	20.0		
10	550.0	0.0	1500.0	9.0	0.0	3.0	linear	
	750.0	y=f(x)	1500.0	9.0	0.0	3.0	semicírculo (x-y)	+

Tabela 5.2: Posição e orientação finais desejadas para o elemento terminal para cada simulação.

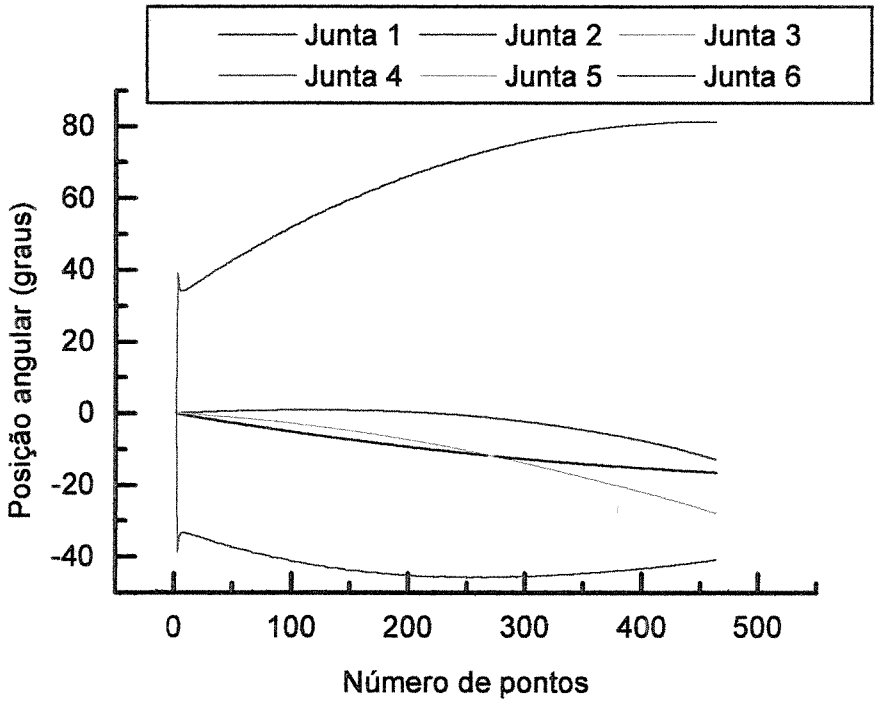
A tabela 5.3 mostra a posição angular final das juntas do robô para cada simulação realizada e o número total de pontos para cada trajetória gerada.

Simulação	Posição angular final das juntas (em graus)						Num. pontos
	1	2	3	4	5	6	
1	-16.5	-12.95	52.56	-40.9	-27.92	81.15	464
2	1.03	-10.46	39.08	20.89	-25.21	10.29	559
3	1.02	-10.45	39.08	20.89	-25.21	10.29	540
4	-42.35	-2.41	12.68	99.71	24.73	-24.57	621
5	1.02	22.07	9.11	19.12	-27.62	12.27	577
6	20.52	-15.29	50.04	17.66	-34.35	-5.11	612
7	-83.87	-1.18	-7.71	-12.82	18.47	101.38	747
8	0.0	-16.12	33.63	0.0	-17.51	0.0	329
	0. 0	0.0	0.0	0. 0	0. 0	0.0	328
9	11.56	-1.18	7.78	-60.97	14.96	68.5	130
	11.6	-26.55	37.36	76.34	13.49	84.41	227
10	1.16	-0.89	4.71	-112.01	9.56	113.65	61
	1.19	-5.87	26.06	-155.06	21.77	158.47	320

Tabela 5.3: Posição angular final das juntas do robô e número total de pontos para cada trajetória.

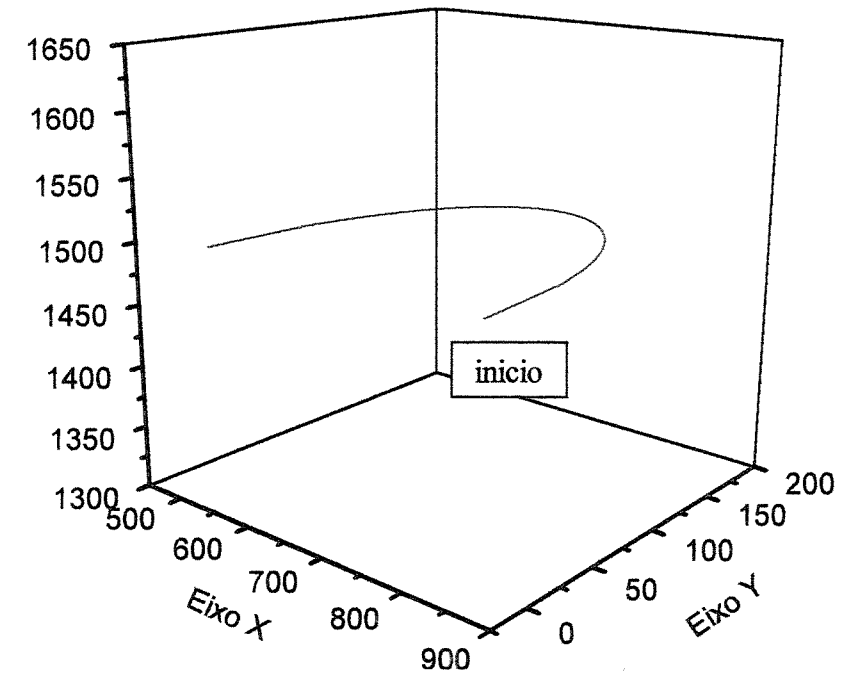


(a) Trajetória espacial seguida pelo elemento terminal.

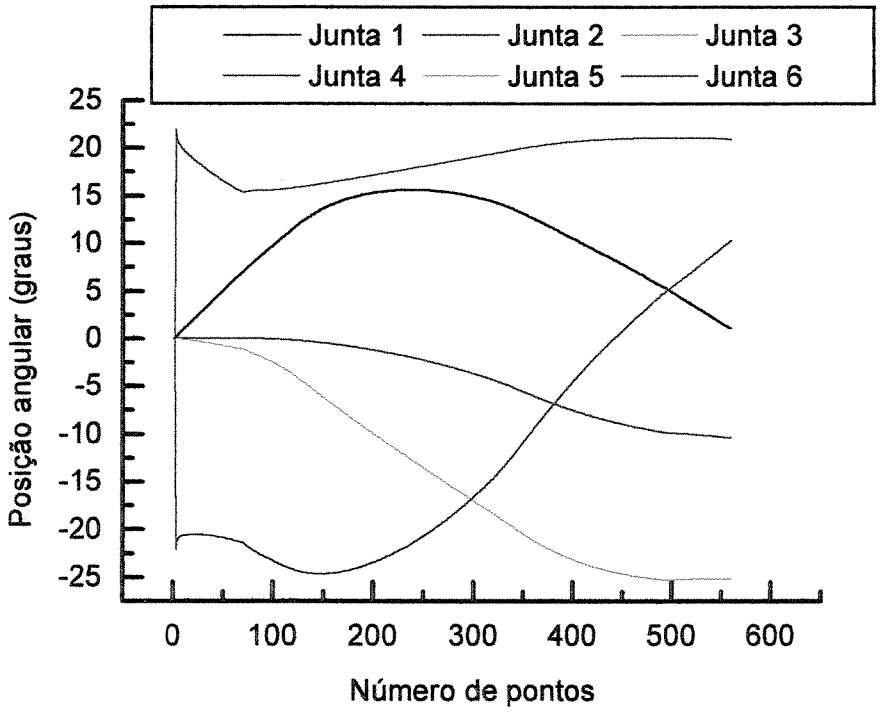


(b) Evolução angular das juntas.

Figura 5.7: Simulação 1, trajetória linear.

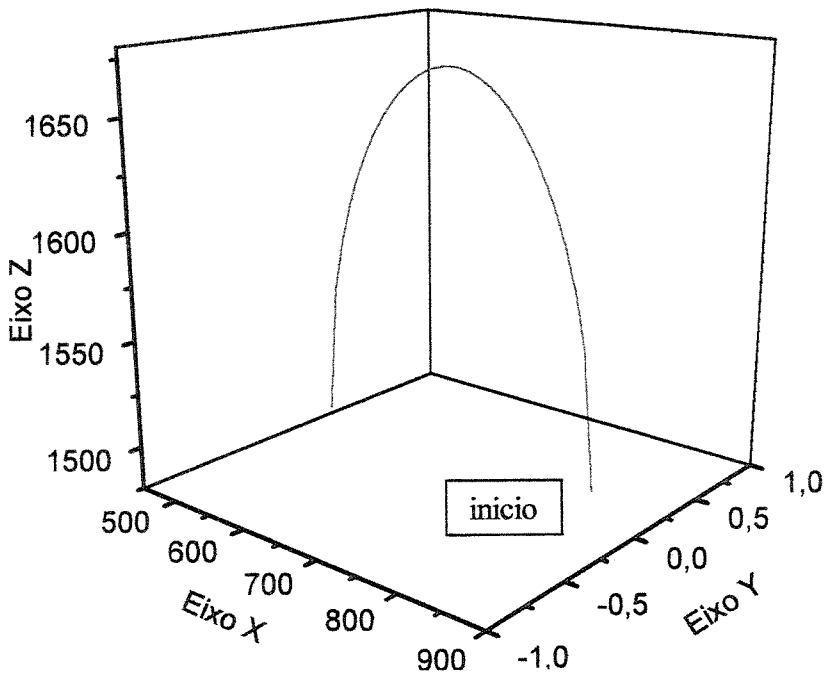


(a) Trajetória espacial seguida pelo elemento terminal.

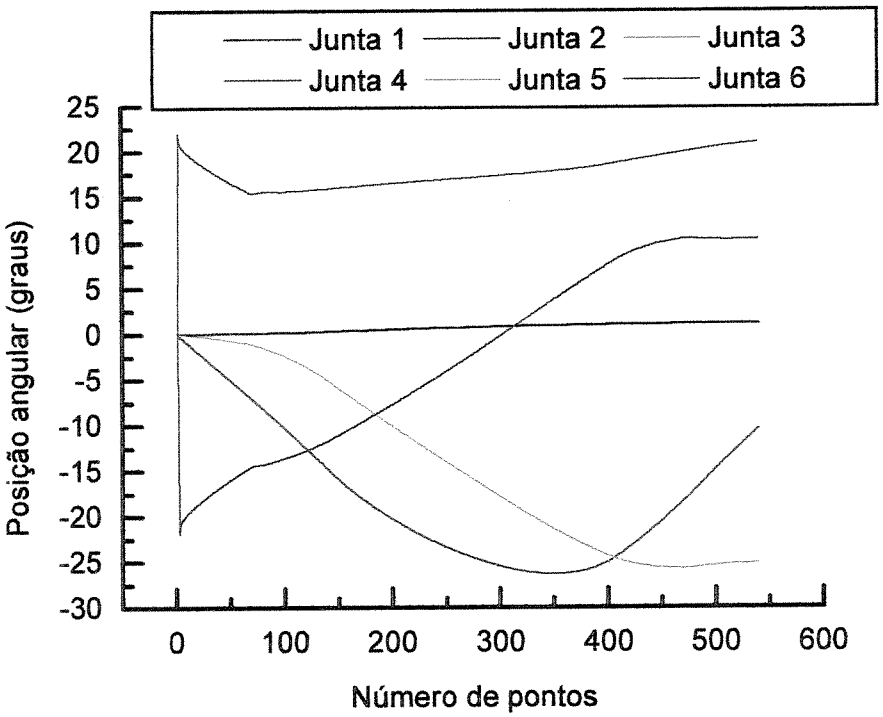


(b) Evolução angular das juntas.

Figura 5.8 : Simulação 2, trajetória semicírculo plano x-y sem variar z, direção positiva.

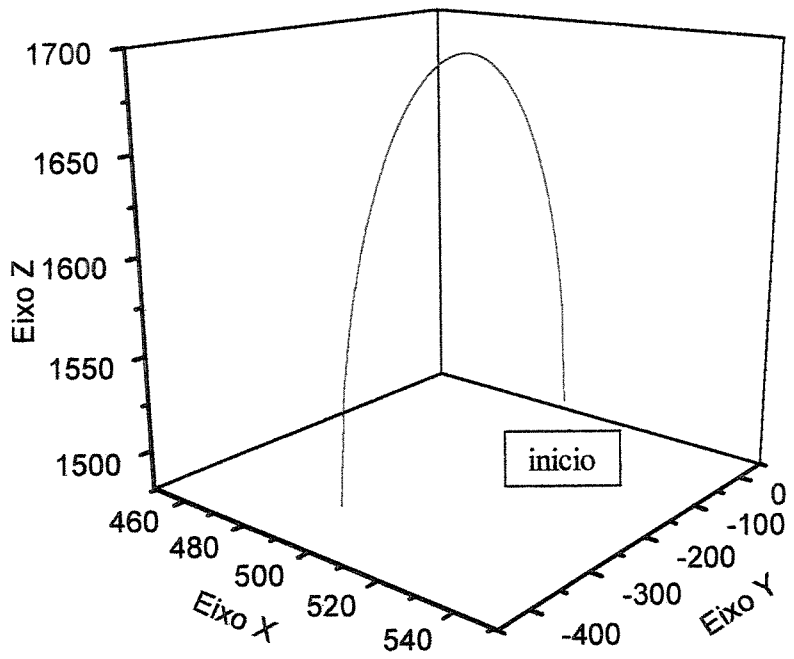


(a) Trajetória espacial seguida pelo elemento terminal.

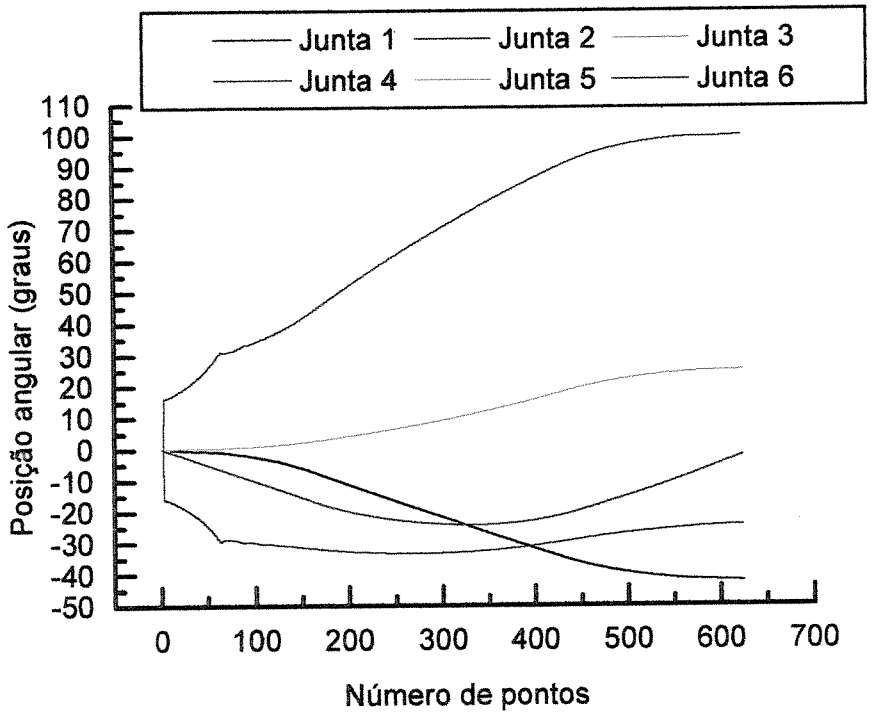


(b) Evolução angular das juntas.

Figura 5.9: Simulação 3,trajetória semicírculo plano x-z sem variar y, direção positiva.

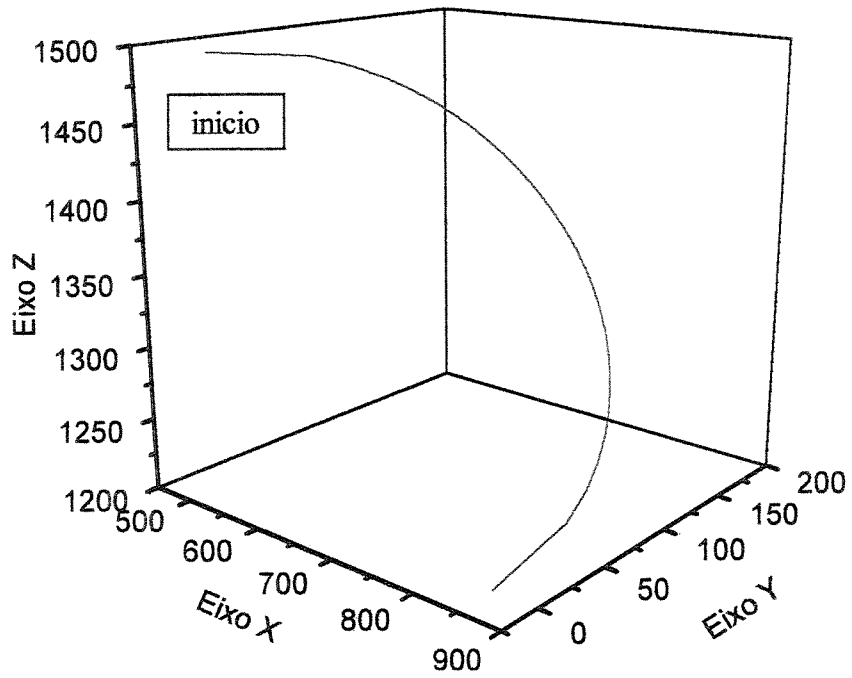


(a) Trajetória espacial seguida pelo elemento terminal.

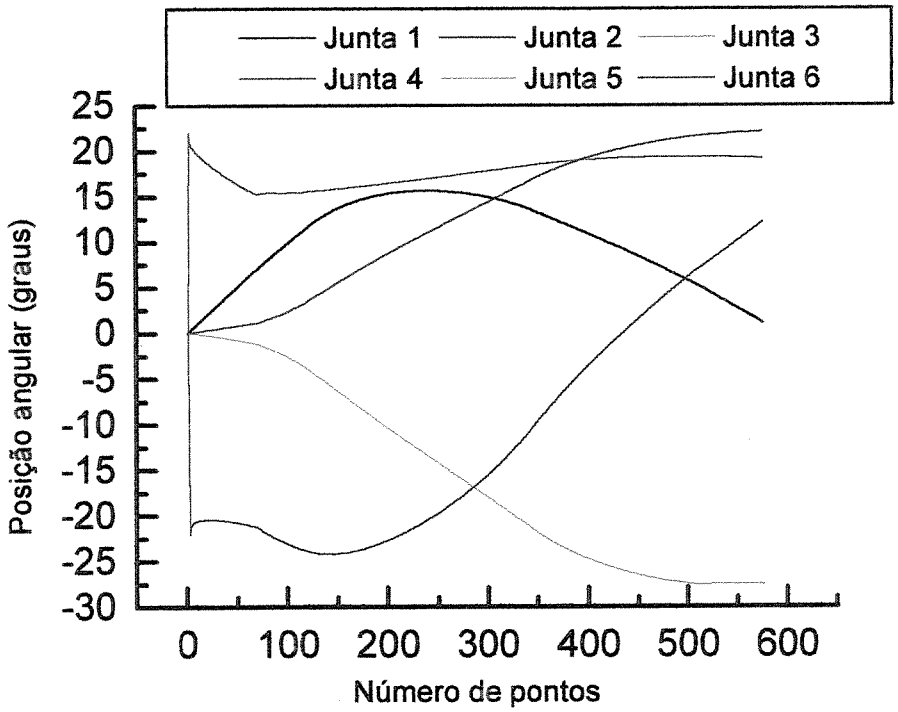


(b) Evolução angular das juntas.

Figura 5.10: Simulação 4, trajetória semicírculo plano y-z sem variar x, direção positiva.

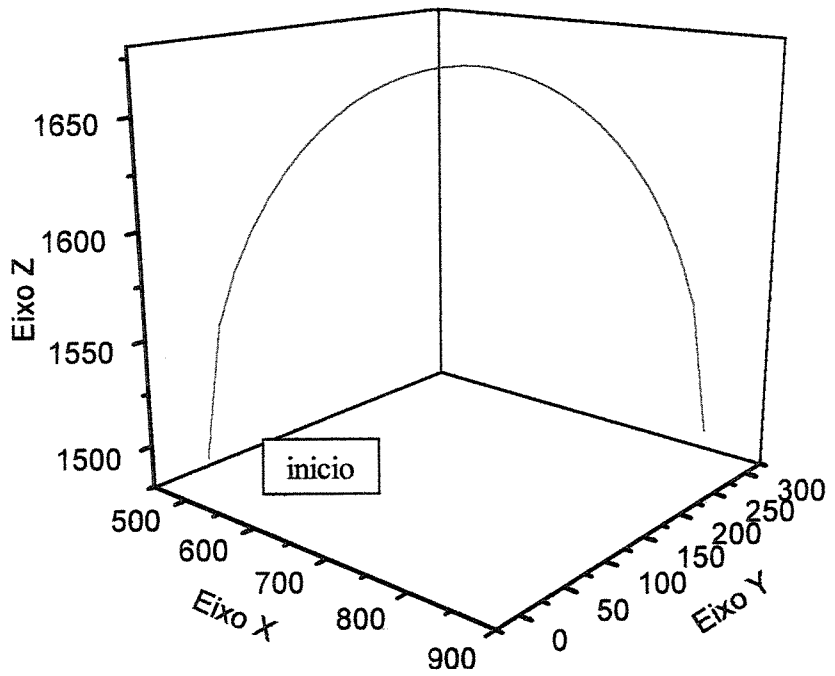


(a) Trajetória espacial seguida pelo elemento terminal.

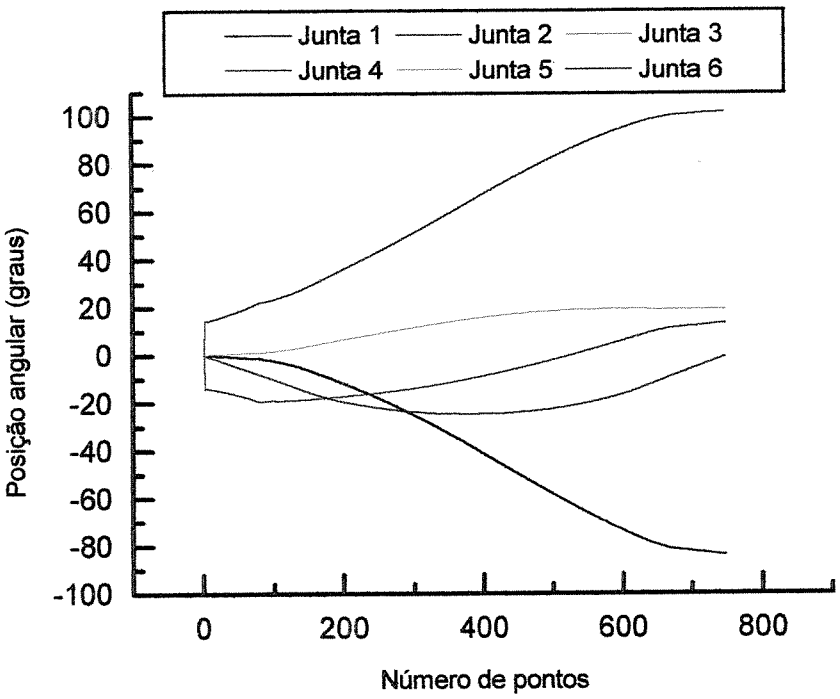


(b) Evolução angular das juntas.

Figura 5.11: Simulação 5, trajetória semicírculo plano x-y variando z, direção positiva.

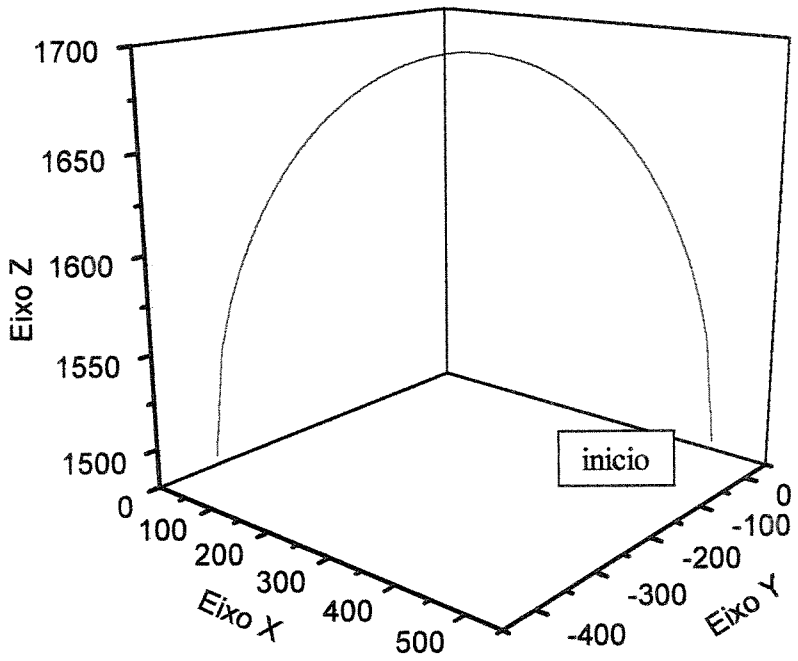


(a) Trajetória espacial seguida pelo elemento terminal.

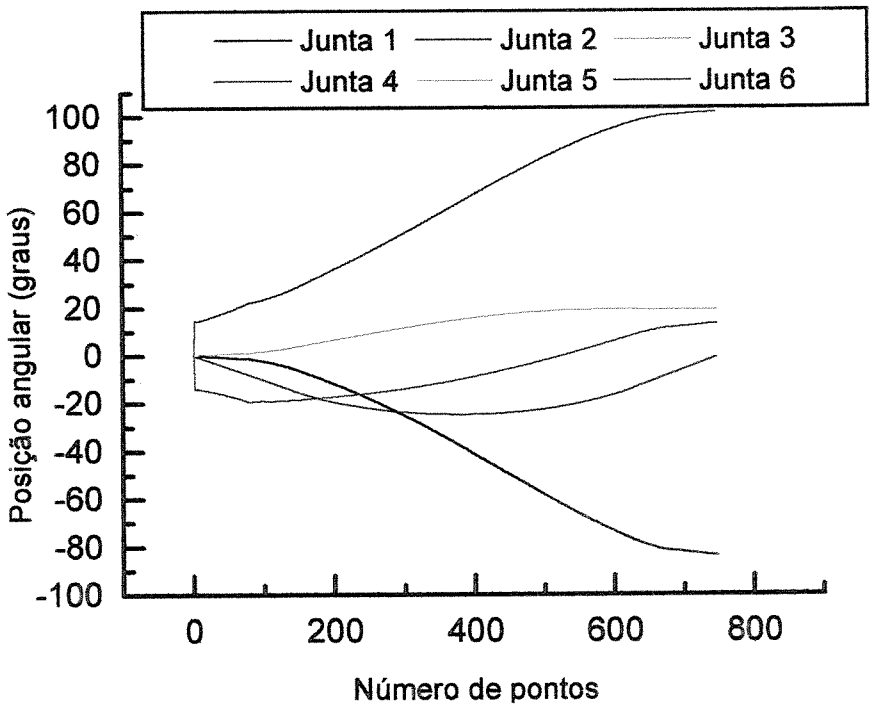


(b) Evolução angular das juntas.

Figura 5.12: Simulação 6, trajetória semicírculo plano x-z variando y, direção positiva.

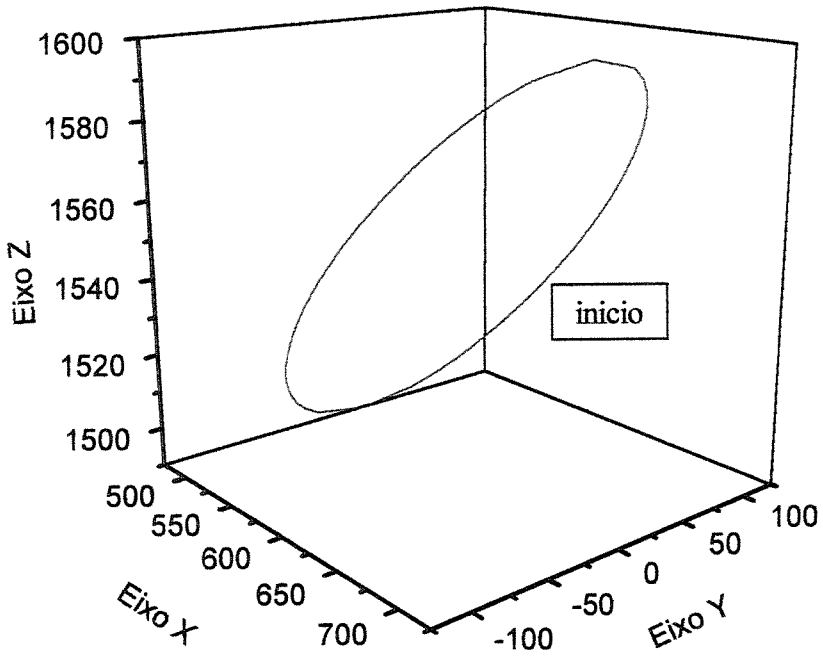


(a) Trajetória espacial seguida pelo elemento terminal.

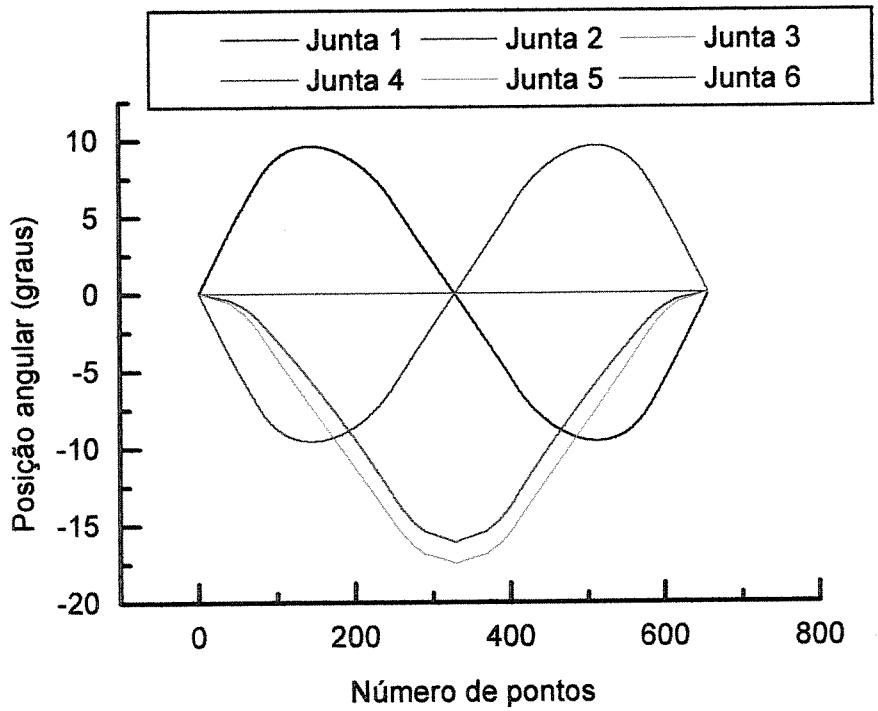


(b) Evolução angular das juntas.

Figura 5.13: Simulação 7, trajetória semicírculo plano y-z variando x, direção positiva.

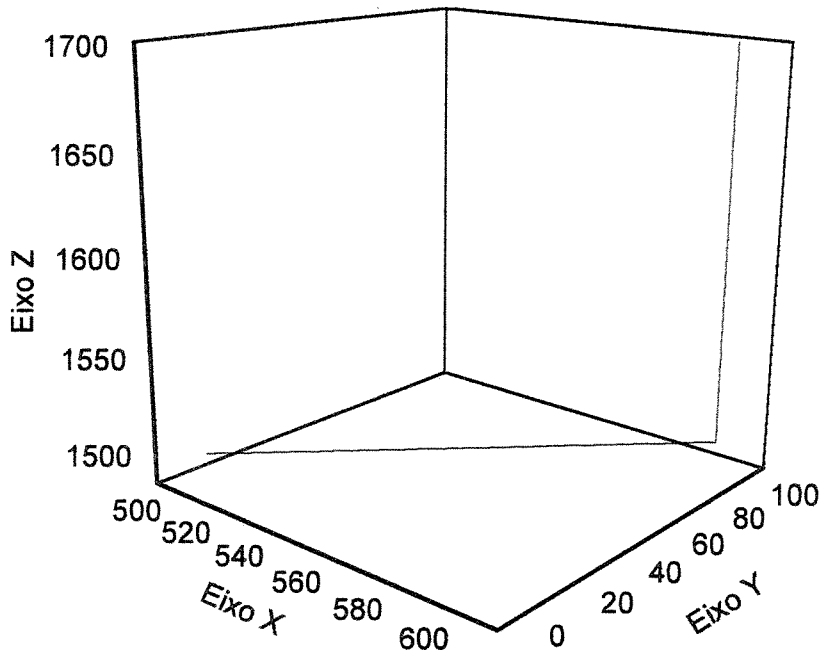


(a) Trajetória espacial seguida pelo elemento terminal.

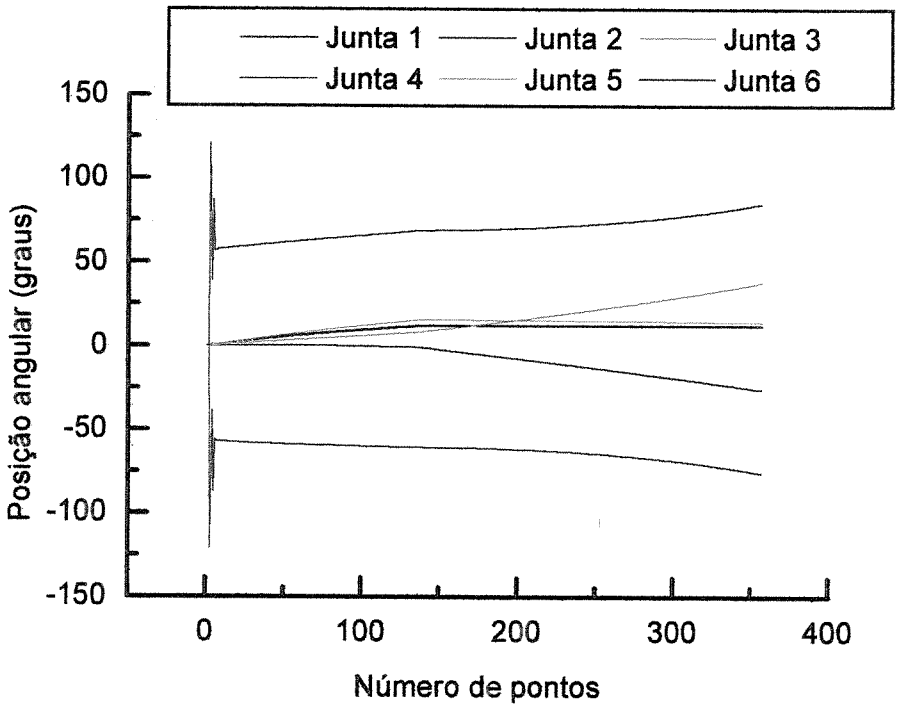


(b) Evolução angular das juntas.

Figura 5.14: Simulação 8, trajetória círculo plano x-y variando z (composta de duas partes).

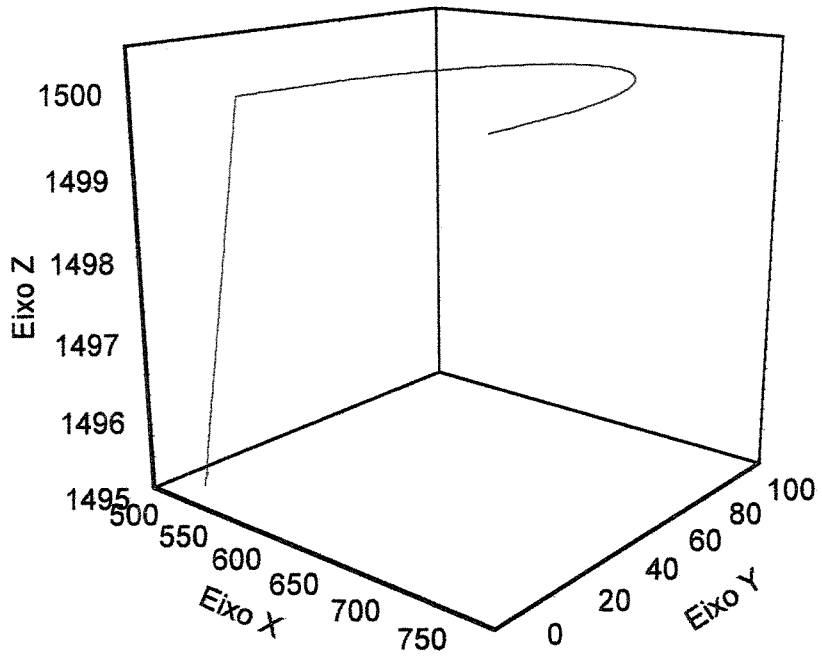


(a) Trajetória espacial seguida pelo elemento terminal.

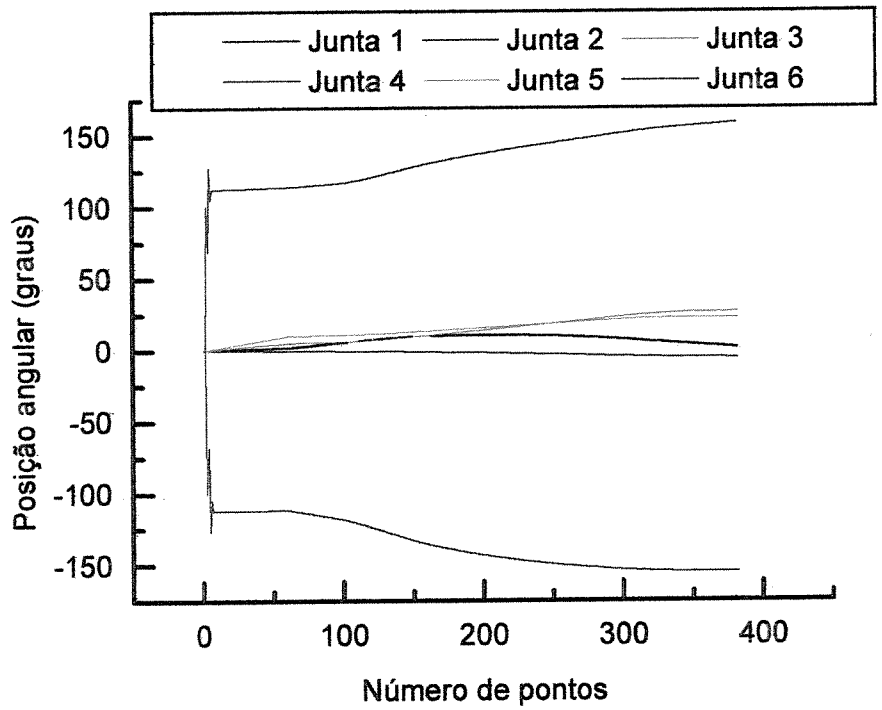


(b) Evolução angular das juntas.

Figura 5.15: Simulação 9, trajetória linear (composta de duas partes).



(a) Trajetória espacial seguida pelo elemento terminal.



(b) Evolução angular das juntas.

Figura 5.16: Simulação 10, trajetória composta de duas partes uma linear e a outra um semicírculo planox-y sem variar z.

Não foram apresentadas todas as simulações realizadas uma vez que, de uma maneira geral, os resultados obtidos foram semelhantes.

5.1.3 Parâmetros

Para as simulações apresentadas anteriormente foram utilizados os seguintes parâmetros:

- erro máximo de posição permitido: 0.5 mm;
- erro máximo de orientação permitido: 0.1^{o};
- número de divisões de cada segmento: 30 divisões.

5.2 Algoritmos para a interpolação e filtragem de pontos de passagem no espaço da juntas

Com o objetivo de gerar uma trajetória a partir de determinados pontos de passagem obtidos pelo operador, no espaço das juntas, foram implementados algoritmos de interpolação e filtragem.

Realizando apenas a interpolação, iria ocorrer no inicio da trajetória, uma aceleração alta (Snyder, 1985), o que não é desejado para qualquer dispositivo robótico.

Para que esta aceleração seja baixa, que resulta em uma velocidade alta da junta, deve-se colocar mais pontos na parte inicial e final da trajetória. Isto é feito através da filtragem da trajetória interpolada, figura 5.17.

Para a realização de uma tarefa, de um modo em geral, o robô executa diversas trajetórias (intervalos) e deste modo a técnica implementada tem de suavizar as extremidades de cada um destes intervalos.

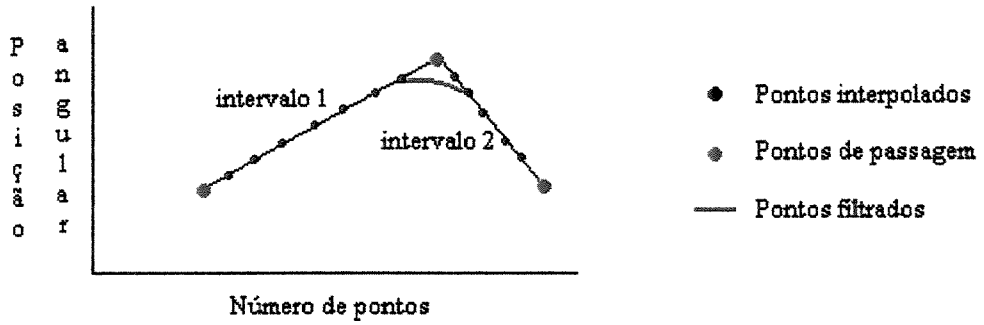


Figura 5.17: Interpolação e filtragem de pontos de passagem.

5.2.1 Interpolação Linear da Trajetória

O objetivo de se fazer uma interpolação linear é o de criar uma seqüência de pontos de passagem que interligam os pontos da trajetória inicial dada.

5.2.1.1 Algoritmo

O primeiro passo é conhecer os valores (ângulos das juntas) dos pontos de passagem a serem interpolados. Uma vez conhecidas as configurações, pode-se calcular o deslocamento de cada junta para cada intervalo, figura 5.17.

$$\Delta_{ij} = \theta_{ij\ FINAL} - \theta_{ij\ INICIAL} \qquad (5.6)$$

para i : 1 ... número de graus de liberdade do robô e j : 1 ... número total de intervalos.

onde

Δ_{ij} = deslocamento que cada junta terá de realizar;

$\theta_{ij\ FINAL}$ = posição angular final da junta i no intervalo j;

$\theta_{ij\ INICIAL}$ = posição angular inicial da junta i no intervalo j;

Após isso se determina qual é o valor do maior deslocamento.

$$\Delta max_j = \max (\Delta_i) \quad (5.7)$$

O próximo passo é calcular qual será o número total de pontos a serem inseridos em cada intervalo, isto é:

$$n_j = \Delta max_j / Vit \quad (5.8)$$

onde

Vit = velocidade máxima utilizada para cada junta varia de 1 a 10 onde 1 representa 10% da velocidade máxima de cada junta, 2 representa 20% e assim por diante.

Os incrementos a serem enviados para cada junta são dados por:

$$\Delta\theta_{ij} = (\theta_{i\,FINAL} - \theta_{i\,INICIAL}) / n_j = \Delta_i / n_j \quad (5.9)$$

onde

$\Delta\theta_{ij}$: incremento angular a ser enviado para a junta i no intervalo j.

O tempo Δt entre cada ponto da trajetória é dado por:

$$\Delta t = \Delta\theta_{ij} / Vit = (\Delta_{ij} / n_j)/(\Delta max_j / n_j) = \Delta_{ij} / \Delta max_j \quad (5.10)$$

5.2.1.2 Exemplo

Com o objetivo de mostrar a simplicidade deste método será apresentado a seguir um exemplo, passo a passo, no qual serão obtidos os incrementos necessários para a interpolação. A tabela 5.4 mostra os pontos de passagem que sofrerão interpolação.

	Articulação	
Ponto	1 (em graus)	2 (em graus)
1	10.0	10.0
2	20.0	100.0
3	400.0	5.0
4	200.0	500.0

Tabela 5.4: Pontos de passagem a serem interpolados.

- Cálculo dos deslocamentos que cada junta terá de realizar, equação 5.6 e dos deslocamentos angulares máximos, equação 5.7, para cada intervalo.

	Δ (deslocamento de cada junta)		
Intervalo	1 (em graus)	2 (em graus)	Δmax_i
1-2	10.0	90.0	90
2-3	380.0	-95.0	380
3-4	-200.0	495.0	495

Tabela 5.5: Cálculos dos deslocamentos angulares máximos.

- Cálculo do número de pontos a serem inseridos em cada intervalo, equação 5.8, e dos incrementos, equação 5.9.

O valor de Vit utilizado em ambos os casos foi de 10.0 (graus).

		Δθ	
Intervalo	n_i	1 (em graus)	2 (em graus)
1-2	9	1.11	10.0
2-3	38	10.0	-2.5
3-4	49	-4.04	10.0

Tabela 5.6: Incrementos das juntas para cada intervalo.

5.2.2 Filtragem da trajetória interpolada

Como foi mencionado anteriormente, para que não haja grande variação da velocidade, devido a problemas de dinâmica (inércias) e mudanças bruscas de direção, deve-se colocar mais pontos nas extremidades de cada intervalo que forma o caminho (angular) a ser seguido pelas juntas.

A seguir serão descritos dois tipos de filtros que podem ser utilizados: triangular e retangular.

5.2.2.1 Filtro triangular

A figura 5.18 mostra um filtro do tipo janela triangular utilizado para alisar o sinal interpolado. A área da janela triangular é unitária para que a energia seja constante (Lau, 1993).

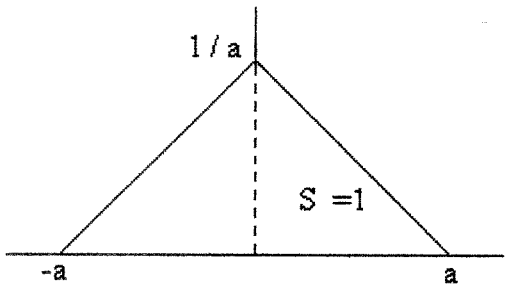


Figura 5.18: Filtro tipo janela triangular.

Utilizando a equação:

$$y = Ax + B \tag{5.11}$$

tem-se:

- para o intervalo entre –a e 0

para $y = 1 / a$ tem-se $x = 0 \Rightarrow B = 1 / a$

$y = 0$ tem-se $x = -a \Rightarrow A = B / a \therefore A = 1 / a^2$

deste modo os valores da janela triangular (w), para este intervalo, serão dados por:

$$w(i) = (1 / a^2) * (i + a) \tag{5.12}$$

para i : -a, 1-a, ...,0.

- para o intervalo entre 0 e a

para $y = 1 / a$ tem-se $x = 0 \Rightarrow B = 1 / a$

$y = 0$ tem-se $x = a \Rightarrow A = B / a \therefore A = - 1 / a^2$

deste modo os valores da janela triangular, para este intervalo, serão dados por:

$$w(i) = - (1 / a^2) * (i - a) \tag{5.13}$$

para i : 0, 1, ... , a.

5.2.2.2 Filtro retangular

A figura 5.19 mostra um filtro do tipo janela retangular.

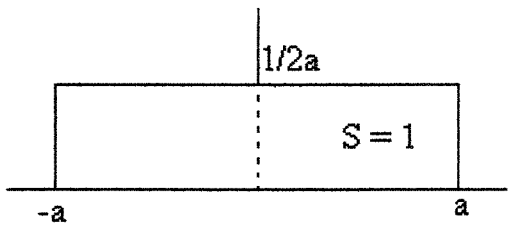


Figura 5.19: Filtro tipo janela retangular.

Os valores da janela retangular (w) para o intervalo de [-a, a] são dados por:

$$w(i) = 1 / 2*a \tag{5.14}$$

5.2.2.3 Exemplo

Na tabela a seguir são apresentados os valores obtidos para w, para uma janela triangular, para *a* igual a 5.

	Intervalo			
para o intervalo [-5 , 0]	i	[-5 , 0]	i	[0 , 5]
		w		w
w(i) = (1 / 25) * (i + 5)	-5	0.0	0	0.2
	-4	0.04	1	0.16
para o intervalo [0 , 5]	-3	0.08	2	0.12
	-2	0.12	3	0.08
w(i) = - (1 / 25) * (i - 5)	-1	0.16	4	0.04
	0	0.2	5	0.0

Tabela 5.7: Valores do filtro triangular para o intervalo [-5 , 5].

5.2.3 Equação para a obtenção da trajetória filtrada e interpolada

Após a obtenção do vetor que contém o filtro (triangular ou retangular) e do vetor que contém a trajetória interpolada, é realizada uma convolução entre os dois vetores. Tal convolução para sinais discretos no tempo pode ser definida pela seguinte expressão:

$$Y(n) = \Sigma\, x(k) * w(n-k) \qquad (5.15)$$

onde

Y = vetor que contém os pontos interpolados e filtrados;
x = vetor que contém os pontos interpolados;
k = número de pontos da trajetória;
w = vetor que contém a janela triangular ou retangular;
n = número de pontos do filtro.

A figura 5.20 mostra o fluxograma implementado para a interpolação e filtragem (a parte em preto fluxograma de interpolação a parte em azul de filtragem).

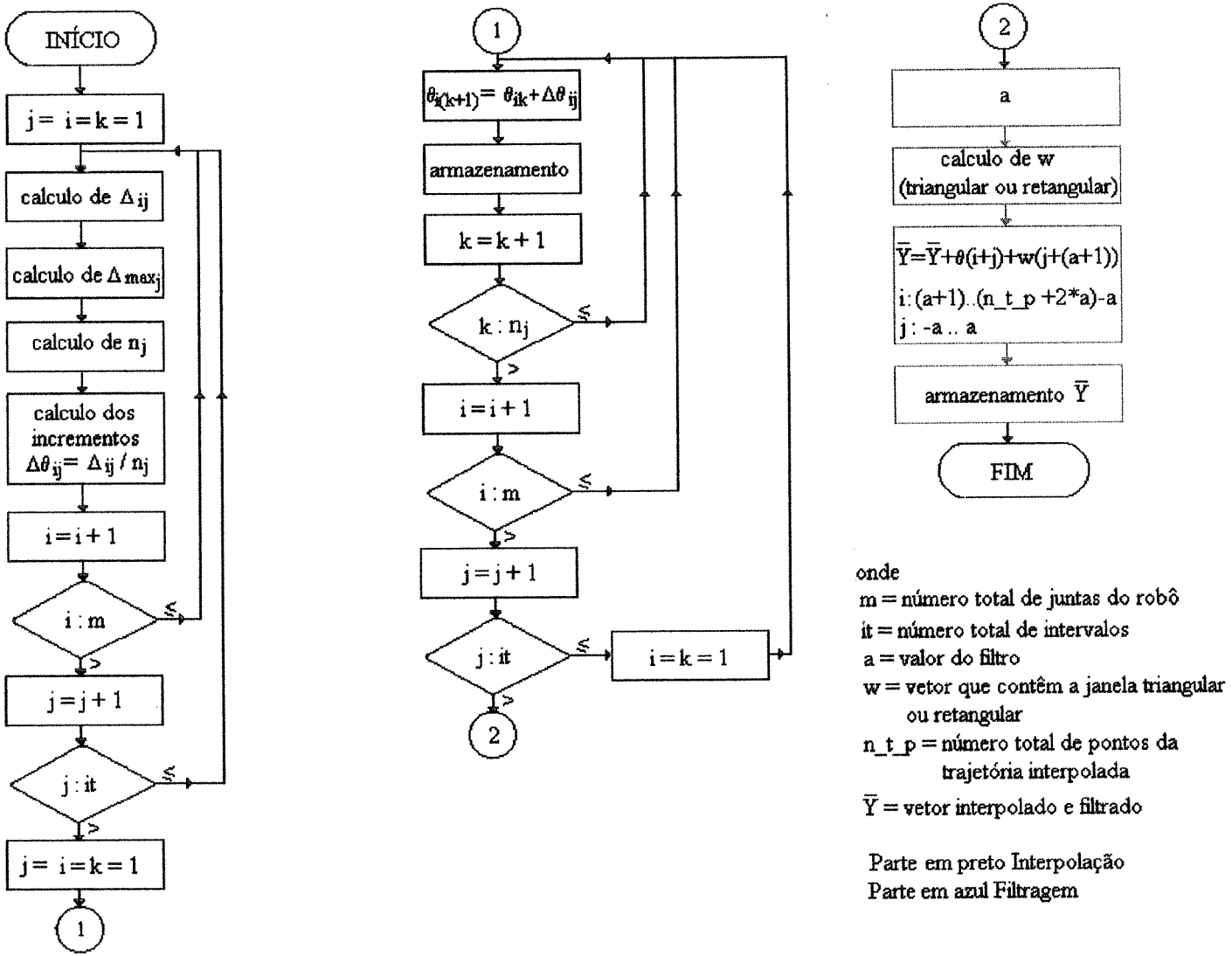


Figura 5.20: Fluxogram para a interpolação e filtragem.

5.2.4 Resultados iniciais obtidos

Será apresentada apenas uma simulação uma vez que os resultados obtidos foram semelhantes nas demais simulações. Na tabela 5.8 observamos os pontos de passagem para um robô com seis graus de liberdade que sofrerão a interpolação e filtragem. Estes pontos podem ser observados na figura 5.21.

	Junta					
Pontos de passagem	1	2	3	4	5	6
1	0.0	0.0	0.0	0.0	0.0	0.0
2	0.10	0.04	0.02	-0.05	-0.02	-0.05
3	0.18	0.08	-0.04	65.28	-0.04	-65.5
4	0.34	0.16	-0.07	73.86	-0.02	-74.21

Tabela 5.8: Pontos de passagem.

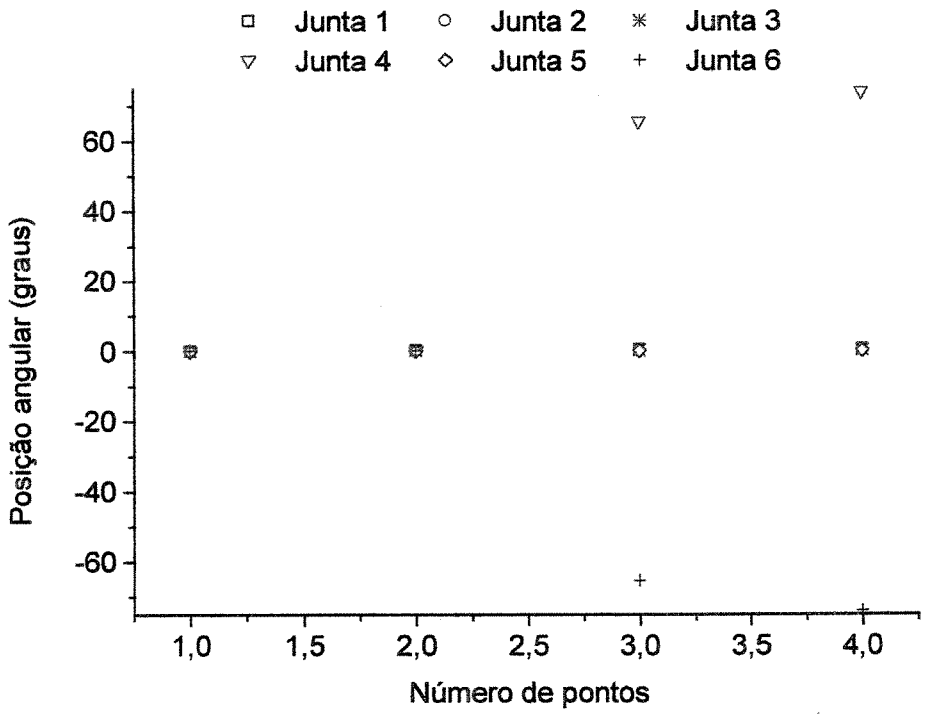


Figura 5.21: Pontos da trajetória a serem interpolados.

Com o objetivo de se obter uma visualização melhor dos pontos de passagem e do efeito da filtragem serão apresentados gráficos para as juntas 1 e 5 separadamente.

As figuras 5.22 e 5.24 (juntas 1 e 5 respectivamente) mostram os pontos de passagem que sofrerão a interpolação e filtragem. As figuras 5.23 e 5.25 (juntas 1 e 5 respectivamente) mostram as trajetórias interpoladas e filtradas utilizando-se os filtros triangular e retangular.

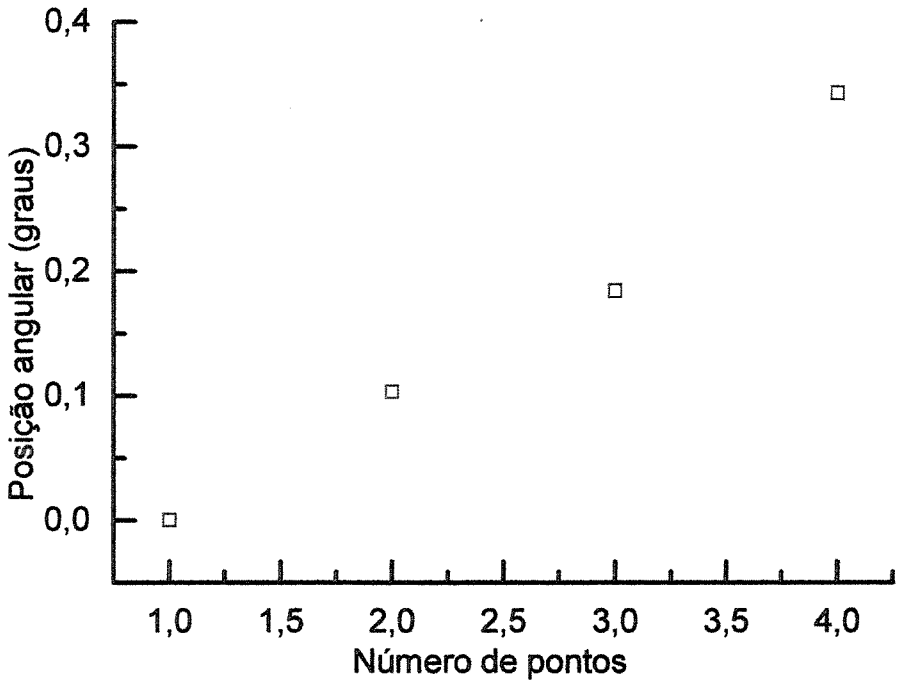


Figura 5.22: Pontos de passagem da junta 1.

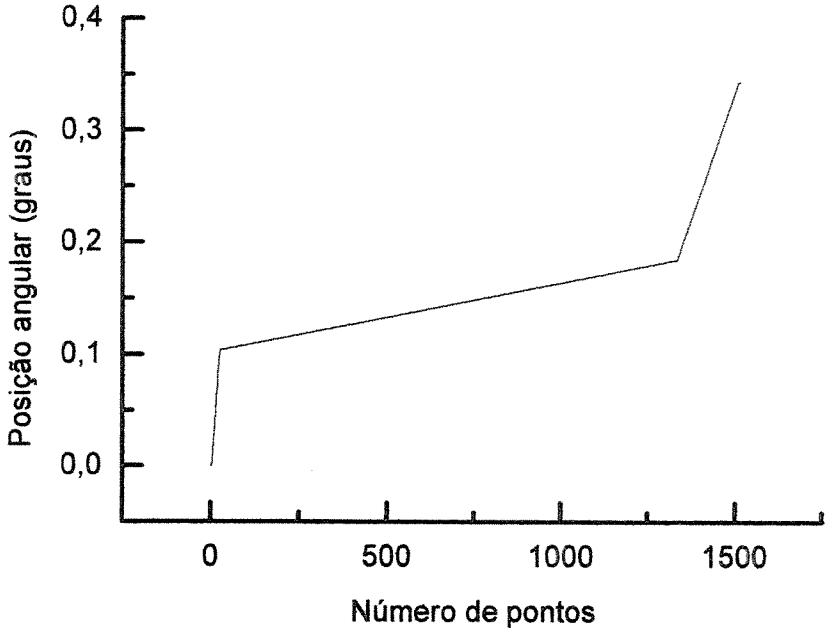


Figura 5.23: Trajetória interpolada e filtrada (utilizando o filtro triangular e retangular) para a junta 1.

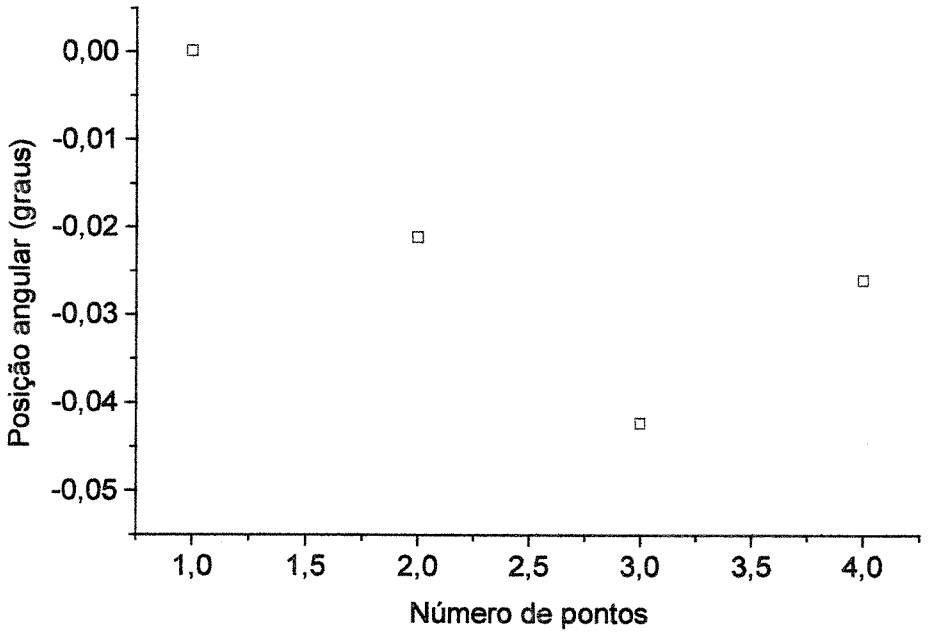


Figura 5.24: Pontos de passagem da junta 5.

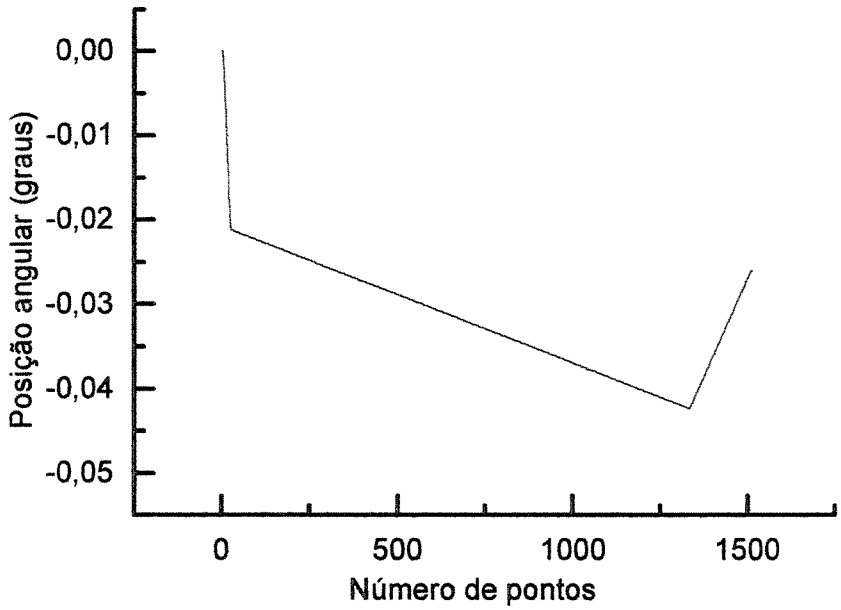


Figura 5.25: Trajetória interpolada e filtrada (utilizando o filtro triangular e retangular) para a junta 5.

Para se observar o efeito da filtragem sobre a trajetória interpolada são apresentadas na tabela 5.9 as posições angulares iniciais para o primeiro intervalo (entre os pontos de passagem 1 e 2 tabela 5.8) da junta 1. Nesta tabela pode-se observar os pontos iniciais da trajetória após a interpolação e após a filtragem, utilizando os filtros triangular e retangular. Observa-se que houve um aumento do número de pontos o que faz com que a velocidade do robô seja menor, o que é desejado. A figura 5.26 mostra o gráfico dos pontos desta tabela.

	Trajetória interpolada		
	sem filtragem	filtro triangular	filtro retangular
Ponto	Posição angular para a junta 1 (graus)		
1	0.0000	0.0000	0.0000
2	0.0052	0.0002	0.0006
3	0.0103	0.0008	0.0017
4	0.0155	0.0021	0.0034
5	0.0206	0.0041	0.0057
6	0.0258	0.0072	0.0086
7		0.0111	0.0120
8		0.0157	0.0160
9		0.0206	0.0206
10		0.0258	0.0258

Tabela 5.9: Posições angulares iniciais para o primeiro intervalo (junta 1) após a interpolação e após a filtragem (filtros triangular e retangular)

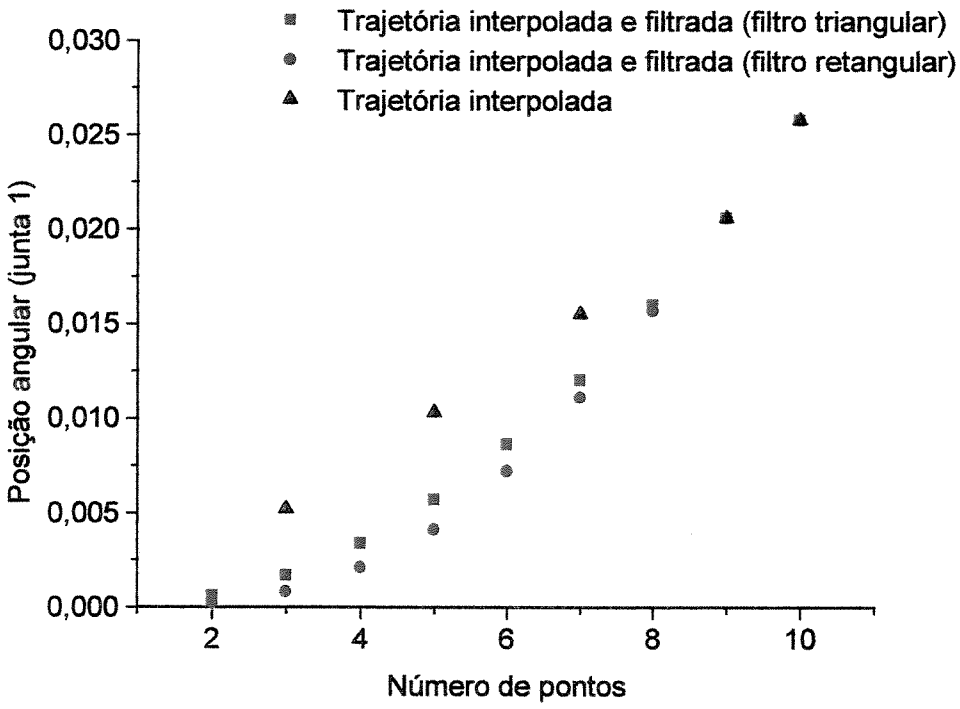


Figura 5.26: Gráfico dos pontos apresentados na tabela 5.9.

5.3 Conclusões iniciais sobre os algoritmos de geração de trajetórias

O algoritmo de geração de trajetórias apresentou excelentes resultados tanto para a discretização linear como em semicírculo. A partir disto podem-se implementar novos tipos de equações para a discretização, dependendo do caminho espacial que o elemento terminal necessita de seguir.

Pode-se observar através da tabela 5.3 que a posição angular final das juntas para as simulações 2 e 3 são as mesmas, portanto, a posição espacial atingida pelo elemento terminal do robô, em ambas as simulações, são as mesmas (isto não é considerada uma configuração múltipla).

A posição espacial final alcançada em ambos os casos é 850, 0 e 1495 (em mm). O que difere é o caminho espacial seguido pelo elemento terminal em cada simulação. Os caminhos podem ser observados na figura 5.27 ou nas figuras 5.8(a) e 5.9(a).

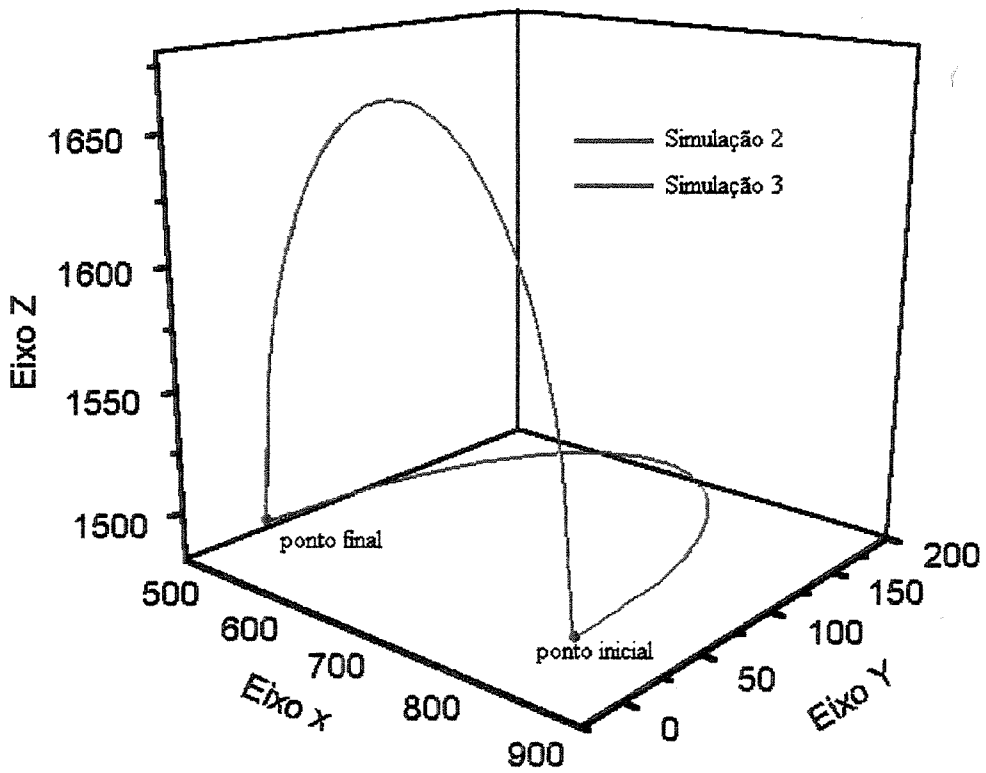


Figura 5.27: Posição espacial final igual obtida através de dois caminhos diferentes.

O algoritmo de interpolação e filtragem também mostrou excelentes resultados como pode ser observado na simulação realizada. No algoritmo implementado é realizada automaticamente a filtragem após a interpolação.

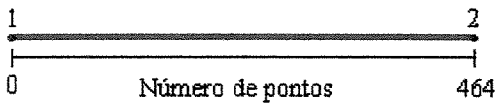
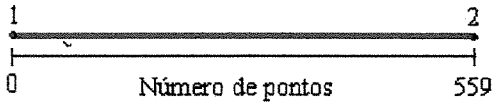
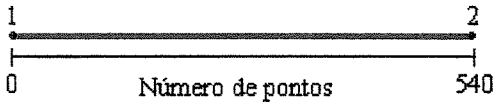
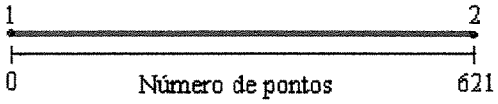
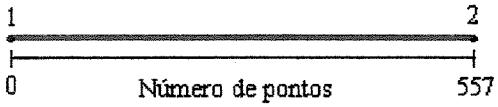
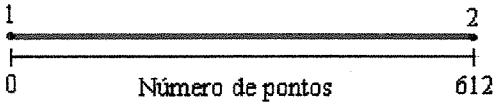
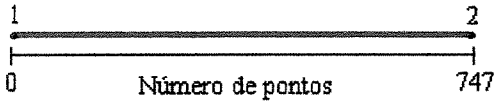
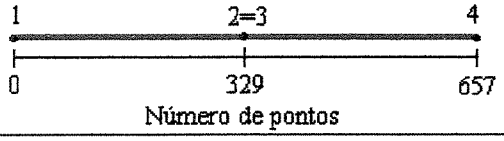
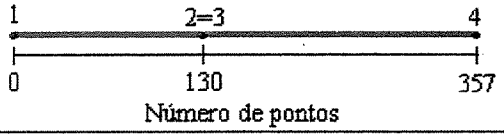
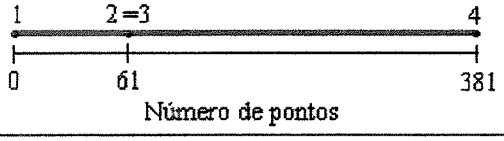
5.3.1 Estudo do comportamento da orientação do elemento terminal

Em muitas aplicações existe a necessidade de que a orientação do elemento terminal permaneça constante, enquanto a sua posição varia. Isto pode ser útil para o robô, por exemplo, para realizar uma solda ou um corte em uma chapa. Na tabela 5.10 pode-se observar em quais simulações este comportamento está ocorrendo.

Pela tabela, pode-se observar que nas simulações 8, 9 e 10, em determinados segmentos da trajetória, este comportamento ocorre. Na simulação 8 o elemento terminal do robô descreve um círculo no espaço, figura 5.14, mantendo a orientação (0.0, 0.0,0.0) em todo o percurso.

Na simulação 9 o elemento terminal percorre dois caminhos lineares, figura 5.15, no primeiro a orientação parte de (0.0, 0.0, 0.0) e vai para (15.0, 12.0, 20.0) e permanece nesta orientação até o final do segundo caminho.

Na simulação 10 o elemento terminal percorre dois caminhos, um linear e um semicírculo, figura 5.15. No primeiro (linear), a orientação parte de (0.0, 0.0, 0.0) e vai para (9.0, 0.0, 3.0) e permanece nesta orientação até o final do segundo caminho (semicírculo). Nas demais simulações, a orientação varia no decorrer da trajetória.

Simulação	Comportamento do elemento terminal durante a trajetória	ponto	Orientação		
			psi	teta	phi
1		1	0.0	0.0	0.0
		2	0.0	25.0	30.0
2		1	0.0	0.0	0.0
		2	10.0	0.0	30.0
3		1	0.0	0.0	0.0
		2	10.0	0.0	30.0
4		1	0.0	0.0	0.0
		2	0.0	25.0	35.0
5		1	0.0	0.0	0.0
		2	10.0	0.0	30.0
6		1	0.0	0.0	0.0
		2	10.0	0.0	30.0
7		1	0.0	0.0	0.0
		2	10.0	0.0	30.0
8		1	0.0	0.0	0.0
		2 = 3	0.0	0.0	0.0
		4	0.0	0.0	0.0
9		1	0.0	0.0	0.0
		2 =3	15.0	12.0	20.0
		4	15.0	12.0	20.0
10		1	0.0	0.0	0.0
		2 = 3	9.0	0.0	3.0
		4	9.0	0.0	3.0

onde

a orientação do elemento terminal varia

a orientação do elemento terminal não varia

Tabela 5.10: Comportamento do elemento terminal durante a trajetória.

Portanto, pode-se notar que o algoritmo de geração de trajetórias permite que a orientação do elemento terminal permaneça constante enquanto se varia a sua posição.

Neste capítulo foram apresentados os algoritmos de geração de trajetórias, interpolação e filtragem e os resultados iniciais obtidos para o algoritmo de geração de trajetórias.

Após a determinação das trajetórias, para que o robô realize determinada tarefa, são possíveis a realização de testes de colisão do robô com ele mesmo e com o ambiente de trabalho (obstáculos e ferramentas). O robô e o ambiente de trabalho podem ser construídos a partir de elementos primitivos (esferas, paralelepípedos e cilindros e outros).

No próximo capítulo será feita uma breve discussão sobre a construção de ambientes e sobre testes de colisão.

Capítulo 6

Visualização Gráfica do Robô e do Ambiente de Trabalho e Tratamento de Colisões

A construção gráfica do robô e do ambiente de trabalho (obstáculos e ferramentas) é de grande importância porque, deste modo, é possível realizar testes de colisão para a realização de um determinado trabalho, antes que o robô propriamente dito o realize, evitando, assim, algum dano ao equipamento; além disso permite que o programador não entre em contato com o ambiente de trabalho real.

Para a construção gráfica do robô, a partir de elementos primitivos (esfera, paralelepípedo, cilindro e outros), foi utilizado o modelo geométrico obtido através de vetores locais, apresentado no capítulo 4, uma vez que se necessita da posição e da orientação de diversos pontos de visualização (pontos de interesse).

Neste capítulo discorre-se sobre a criação do meio ambiente e sobre a construção do robô. Também será apresentado um método para o tratamento de colisões que pode ser utilizado para qualquer robô.

6.1 Visualização gráfica

Com o objetivo de obter a visualização gráfica do robô e do ambiente de trabalho foram implementados um conjunto de elementos primitivos básicos (cilindro, paralelepípedo e esfera) e variações dos mesmos (círculo, semicírculo, semicilindro e cone).

6.1.1 Elementos primitivos básicos implementados

Os elementos primitivos básicos implementados serão descritos a seguir. Os elementos primitivos: círculo, semicírculo, semicilindro e cone não serão descritos, uma vez que são constituídos a partir dos elementos primitivos básicos.

- Elemento primitivo esfera

Um modelo esférico é definido por uma origem, uma orientação e por um raio.

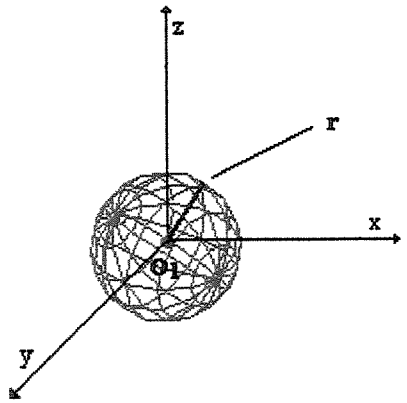


Figura 6.1: Modelagem de uma esfera.

- Elemento primitivo cilindro

Um cilindro é definido por uma origem, uma orientação, por um raio e pelo comprimento do eixo. O eixo do cilindro será definido ao longo do eixo z e a sua origem no centro do círculo da base.

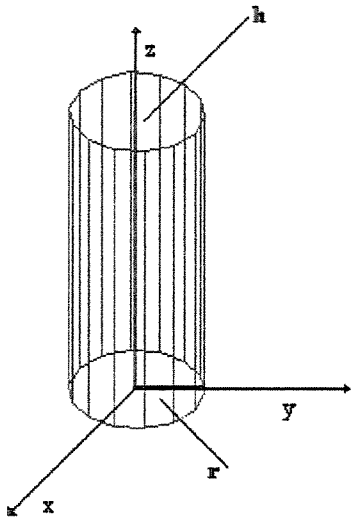


Figura 6.2: Modelagem de um cilindro.

- Elemento primitivo paralelepípedo

Um paralelepípedo é definido por uma origem, uma orientação, por um comprimento ao longo do eixo x, por um comprimento ao longo do eixo y e por um comprimento ao longo do eixo z.

A origem do paralelepípedo será definida de tal forma que o conjunto das medidas do mesmo sejam definidos na direção positiva dos eixos.

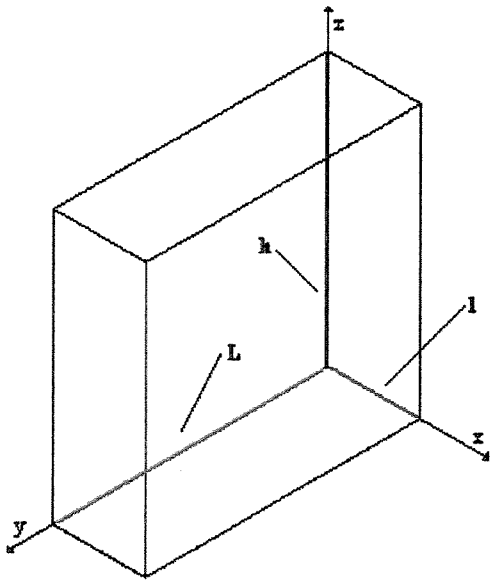


Figura 6.3: Modelagem de um paralelepípedo.

A seguir discorre-se sobre a criação do ambiente de trabalho e sobre a construção do robô.

6.1.2 Criação do ambiente de trabalho - Obstáculos e Ferramentas

O ambiente de trabalho é criado a partir da definição do modelo geométrico do mesmo. Não existe nenhuma diferença entre o modelo geométrico de ferramentas, painéis, módulos ou obstáculos no momento de criação. Qualquer uma destas opções é composta por um ou um conjunto (elemento composto) de elementos primitivos básicos.

Para a edição arquivo do modelo geométrico de um obstáculo ou ferramenta utiliza-se um editor de texto comum, criando-se um arquivo com extensão .def (arquivo de **definição**).

6.1.2.1 Arquivo do modelo geométrico do ambiente de trabalho

A estrutura deste arquivo é idêntica em todos os modelos e cada linha possui uma única informação, dependendo apenas do elemento primitivo básico que esta sendo utilizado. Os ângulos são expressos em graus e os comprimentos são expressos em mm. A tabela 6.1 apresenta a estrutura genérica deste tipo de arquivo.

COMENTÁRIO	
TIPO DE ELEMENTO PRIMITIVO	esfera , cilindro, paralelepípedo e composto
ORIGEM	
posição X	em mm
posição Y	em mm
posição Z	em mm
ORIENTAÇÃO	
Ângulo ϕ	em graus
Ângulo θ	em graus
Ângulo ψ	em graus

Tabela 6.1: Estrutura dos arquivos de definição

No Anexo IV é descrita mais detalhadamente a estrutura dos arquivos (em ASCII) para cada elemento primitivo básico. Também será apresentada a estrutura para a criação dos arquivos compostos (elementos compostos).

6.1.2.2 Exemplo de criação de um obstáculo

As figuras abaixo mostram um exemplo da criação de um obstáculo, no caso um painel e um cilindro, mudando-se a orientação. A posição espacial a partir da qual o paralelepípedo é desenhado é dada por (900.0, 0.0, 400.0). As dimensões são: espessura de 50, largura.de 500 e altura de 1200 todos os valores são dados em mm. A tabela 6.2 mostra, para cada figura, a orientação utilizada para a construção do obstáculo.

Figura	Orientação(graus)
5.4	0.0 , 0.0 , 0.0
5.5	90.0 , 0.0 , 0.0
5.6	0.0, 90.0, 0.0
5.7	0.0 , 0.0 , 90.0

Tabela 6.2: Orientação utilizada para cada figura.

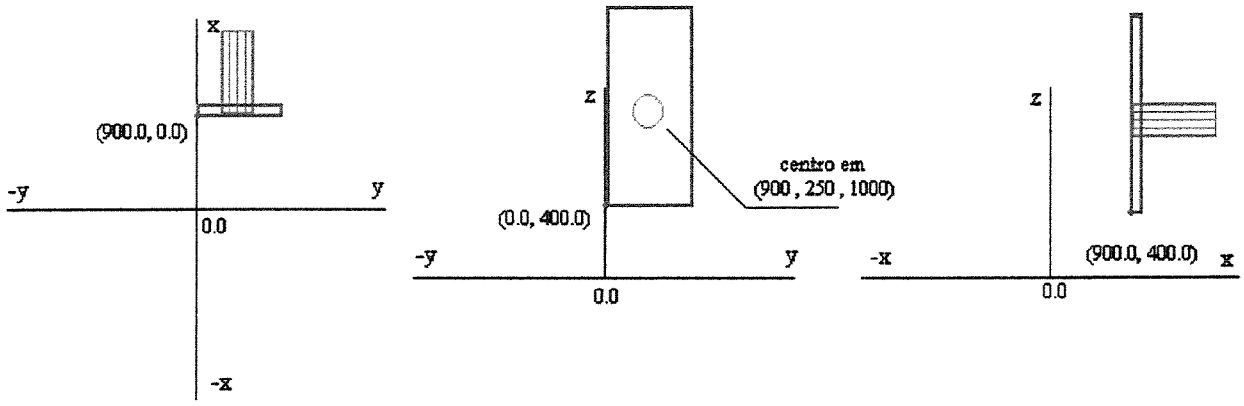


Figura 6.4: Obstáculo com a orientação dada por (0.0, 0.0, 0.0).

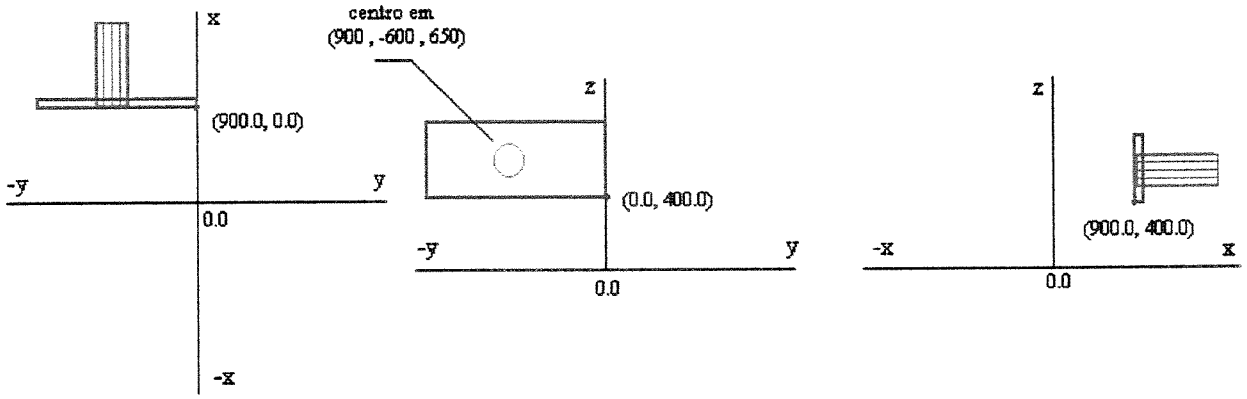


Figura 6.5: Obstáculo com a orientação dada por (90.0, 0.0, 0.0).

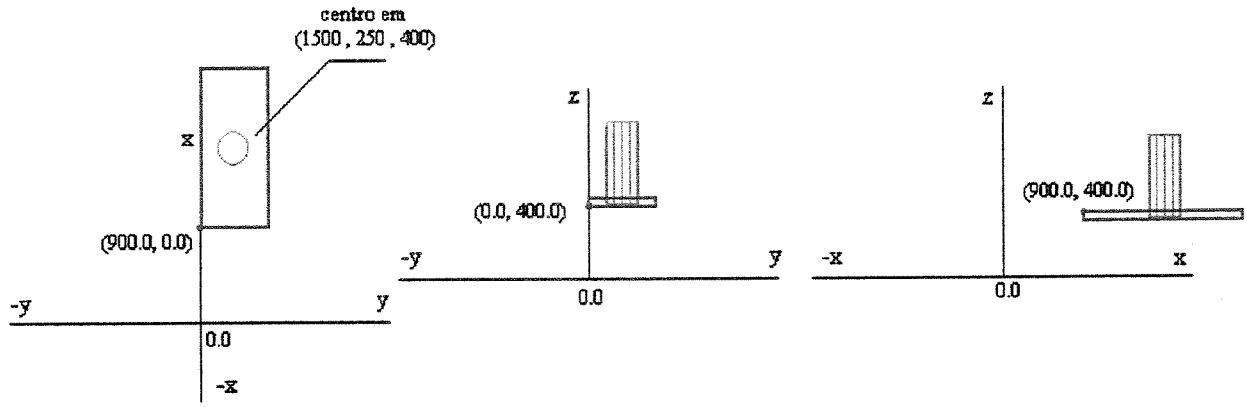


Figura 6.6: Obstáculo com a orientação dada por (0.0, 90.0, 0.0).

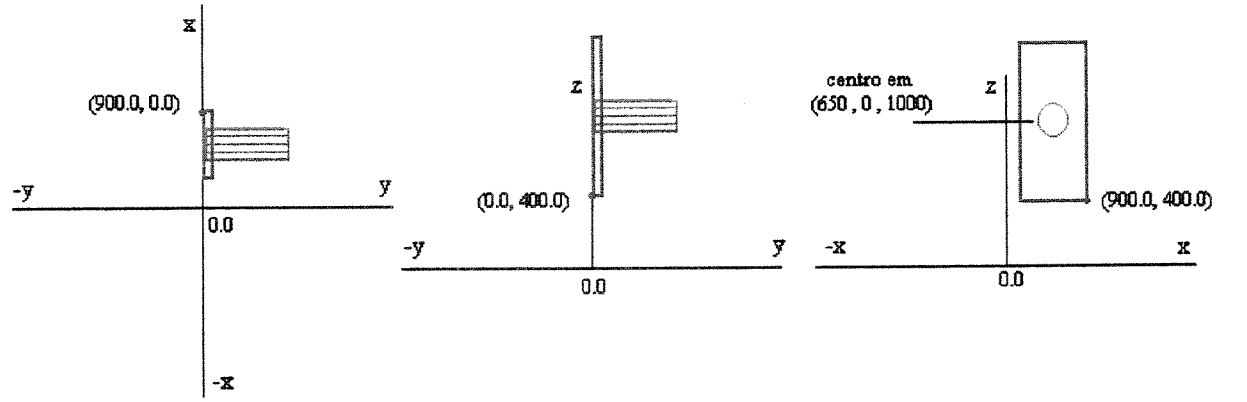


Figura 6.7: Obstáculo com a orientação dada por (0.0, 0.0, 90.0).

A partir da criação do meio ambiente é possível realizar testes de colisão desse meio com o robô.

6.1.3 Construção gráfica do robô

Para a construção gráfica do robô foi utilizado o modelo geométrico obtido através de vetores locais pois necessita-se da posição e orientação de diversos pontos (pontos de interesse).

Além dos elementos primitivos básicos (utilizados para a construção do meio ambiente) foram utilizadas as variações (círculo, semicírculo, semicilindro e cone) com o objetivo de melhorar a visualização.

Estas variações não foram implementadas para a construção do ambiente de trabalho uma vez que um obstáculo (ou ferramenta) é estático, podendo o desenho ser refinado através de elementos compostos.

Nas figuras a seguir serão indicadas, no robô, as posições das juntas e os pontos de interesses utilizados para a construção gráfica do mesmo. Na figura 6.8 pode-se observar as posições das juntas e nas figuras de 6.9 a 6.11 são apresentados os pontos de interesse. Estes pontos de interesse correspondem aos pontos indicados na figura 4.11 página 43.

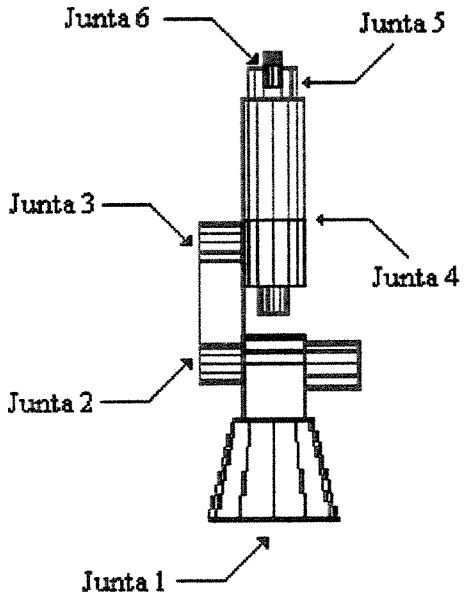


Figura 6.8: Posição das juntas do robô.

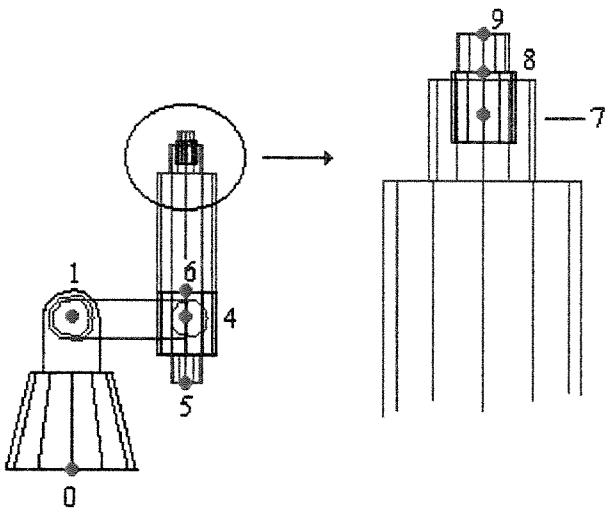


Figura 6.9: Pontos de interesse no plano z-x.

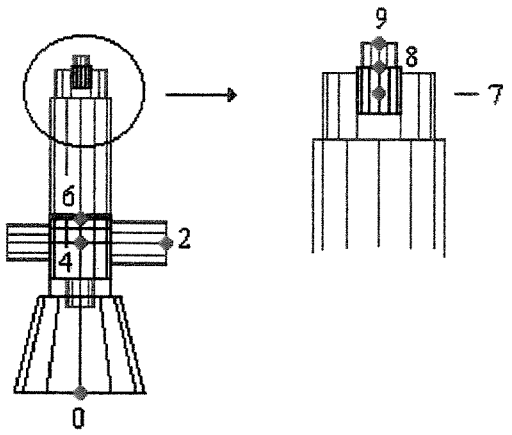


Figura 6.10: Pontos de interesse no plano z-y.

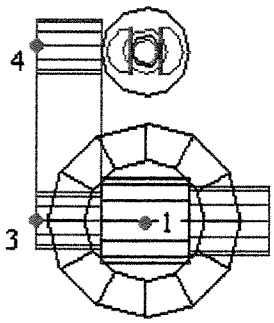


Figura 6.11: Pontos de interesse no plano x-y.

6.2 Tratamento de Colisões

Com a obtenção da trajetória angular para determinada tarefa e a construção gráfica do robô através dos sólidos geométricos podem-se realizar testes de colisão para cada conjunto de ângulos da mesma.

O objetivo principal do tratamento de colisões é o de detectar as colisões que possam existir entre elementos pré-definidos que compõem o robô e o meio ambiente. Partindo, então, para o problema global, ele deve detectar a possibilidade de colisões entre:

- o robô com ele mesmo;
- o robô com os painéis de atuação e base móvel;
- o robô com o ambiente estático;
- a ferramenta com os painéis de atuação e base móvel;
- a ferramenta com o ambiente estático.

6.2.1 Método utilizado para o tratamento de colisões

As idéias básicas do método são apresentadas nas figuras 6.12 e 6.13. Basicamente, se o obstáculo é considerado um plano (medida da altura e a largura é muito maior que a sua espessura, por exemplo, um painel) é calculada a distância entre um ponto, pertencente ao braço do robô, e o plano e, se o obstáculo não é considerado um plano, então, o método utilizado é o de discretizar todos os elementos (esferas, paralelepípedos e cilindros) componentes, tanto do robô, quanto do meio ambiente (obstáculos) em esferas (envelope esférico) e testar a possibilidade de interação entre estas para cada conjunto de ângulos da trajetória a ser realizada.

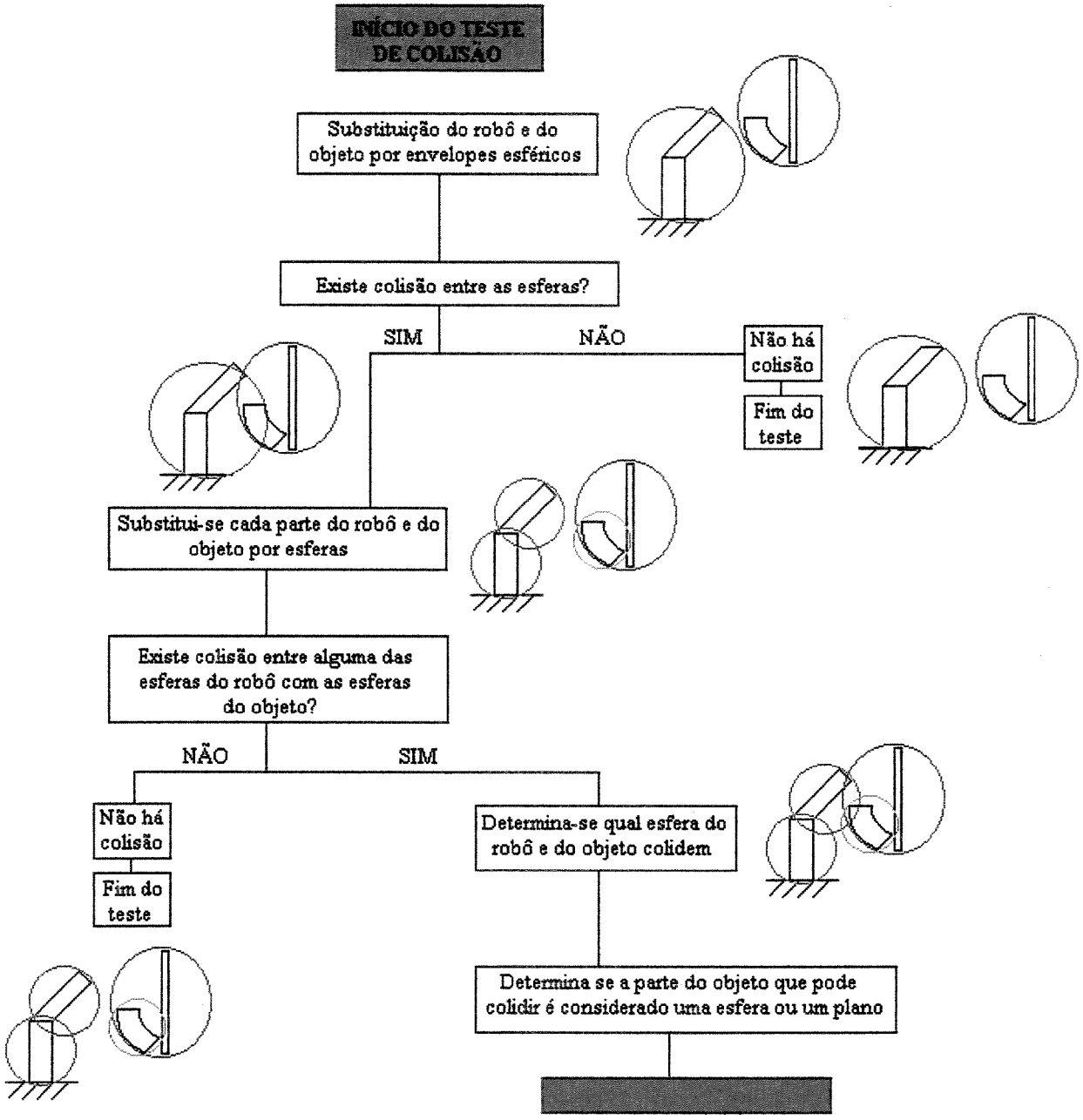


Figura 6.12: Método para o tratamento de colisões PARTE 1.

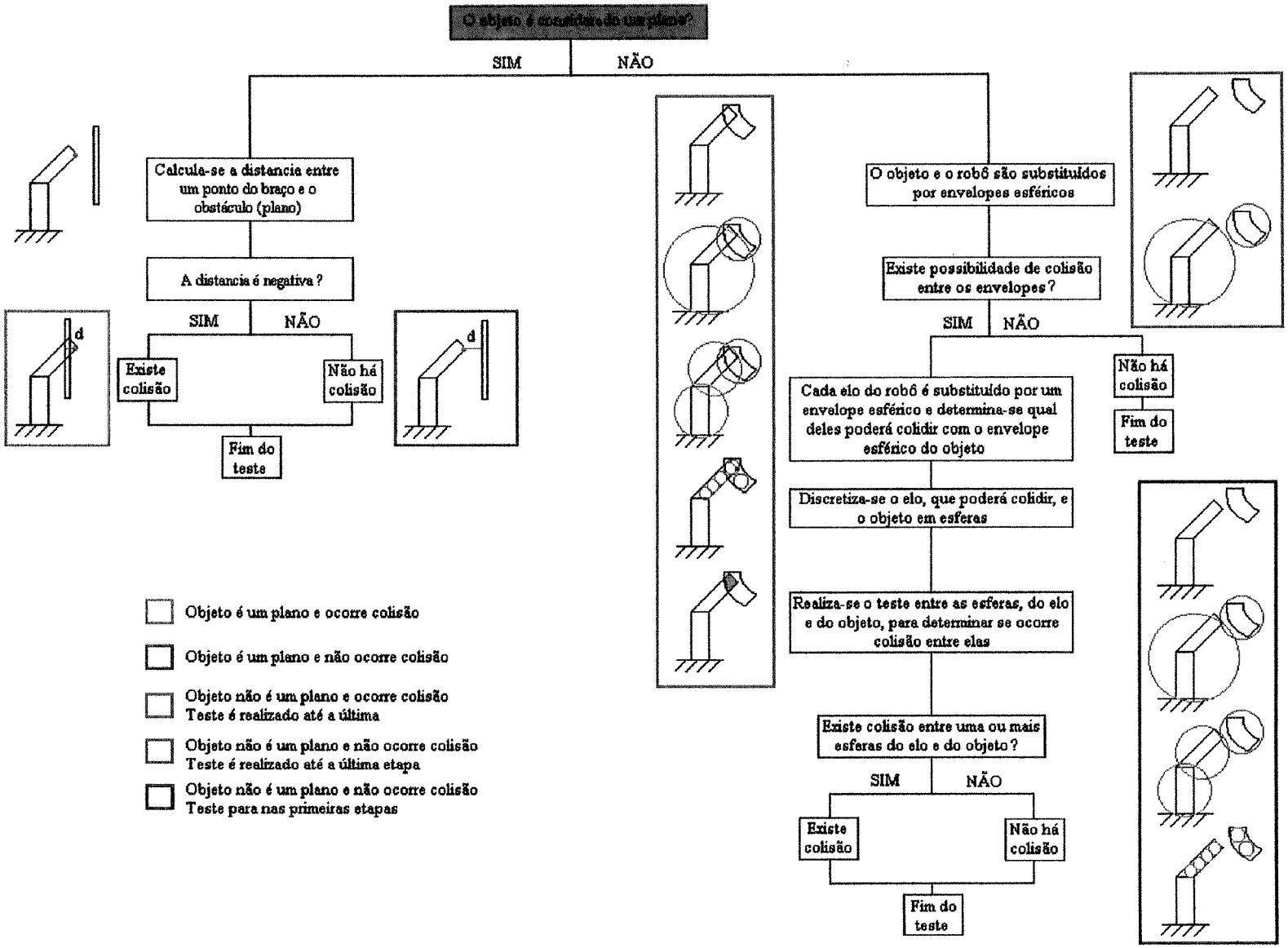


Figura 6.13: Método para o tratamento de colisões PARTE 2.

Portanto se o elemento não é considerado um plano, no ponto de vista do programa, para o teste de colisão, o robô e o obstáculo são considerados um conjunto de envelopes esféricos. A estrutura de árvore apresentada na figura abaixo mostra essa situação.

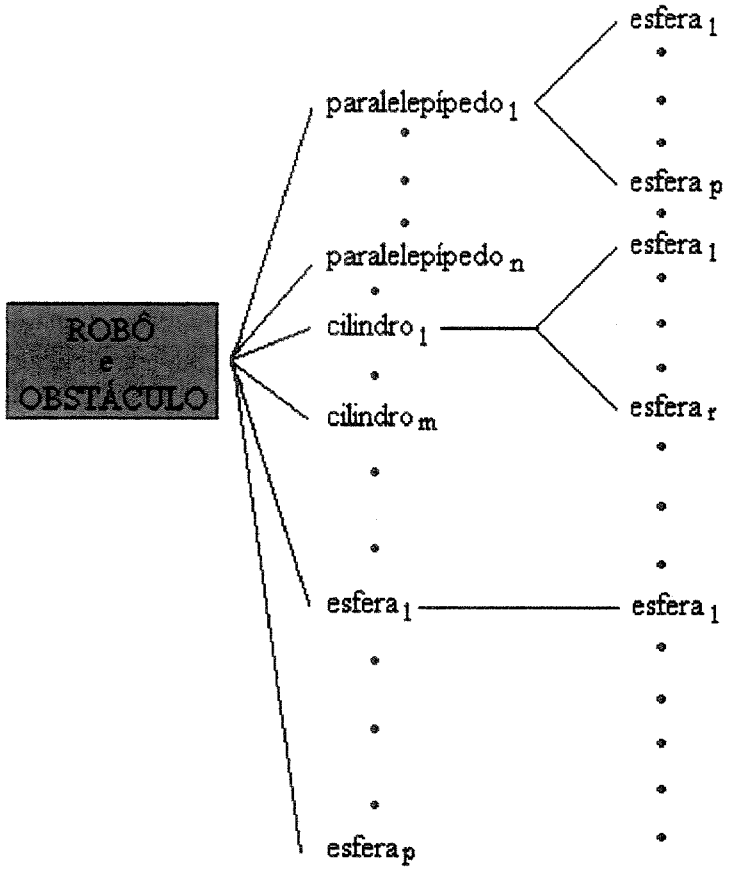


Figura 6.14: Ponto de vista do programa para o teste de colisão: robô e obstáculo são considerados um conjunto de envelopes esféricos.

Nos tópicos a seguir serão apresentados diversos conceitos utilizados no algoritmo de detecção de colisões. No tópico 6.3 serão apresentados o conceito de envelope esférico e as formas para se calcular os envelopes dos elementos primitivos. No tópico 6.4 serão apresentados os testes para a detecção de colisões para obstáculos não planos e no tópico 6.5 para obstáculos planos

6.3 Cálculo do envelope esférico para um obstáculo não plano

A idéia do envelope esférico é de substituir o elemento em questão por uma esfera e realizar o teste entre esferas para uma primeira aproximação. O teste é feito calculando-se a distância entre as duas esferas. Isto acelera o método, pois este teste trata-se do mais simples, como poderá ser visto mais adiante, também fazendo desnecessária a utilização dos refinamentos, caso já não haja uma iteração. Para cada elemento existe uma forma de calcular o envelope esférico.

6.3.1 Cálculo do envelope esférico para uma esfera

O envelope esférico de uma esfera é a própria esfera. Portanto é criado um envelope com o mesmo centro e mesmo raio da esfera em estudo.

6.3.2 Cálculo do envelope esférico para um cilindro

Considere um cilindro de altura h e raio R, o envelope esférico do mesmo será descrito por uma esfera de centro, c, e raio, r, dados por:

$$c = (0 , 0 , h/2) \tag{6.1}$$

$$r = 1/2 * (h^2 + 4*r^2)^{½} \tag{6.2}$$

6.3.3 Cálculo do envelope esférico para um paralelepípedo

Considere um paralelepípedo de altura h, comprimento 1, e profundidade p. O envelope esférico do mesmo será descrito por uma esfera de centro, c, e raio, r, dados por:

$$c = (p/2 , l/2 , h/2) \tag{6.3}$$

$$r = 1/2 * (h^2 + l^2 + t^2)^{½} \tag{6.4}$$

6.3.4 Cálculo do envelope esférico para um composto

O elemento composto é formado por um conjunto de elementos primitivos. Desta forma, o cálculo do envelope esférico total deve então combinar os envelopes esféricos dos seus elementos integrantes.

O procedimento consiste em calcular o envelope do primeiro elemento e do segundo e então fazer uma combinação dos dois envelopes, gerando um novo que engloba os dois. Na próxima passagem combina-se este novo envelope com o envelope do terceiro elemento e assim sucessivamente.

O procedimento para a combinação de duas esferas é apresentado a seguir.

Considerando que os centros e os raios das esferas em estudo sejam dados respectivamente por O_1 e r_1, O_2 e r_2, o raio do novo envelope esférico será dado por:

$$R = (\| O_1O_2 \| + r_1 + r_2) / 2 \qquad (6.5)$$

onde:

$\| O_1O_2 \|$ = módulo do vetor que une os dois centros;

O novo envelope esférico tem seu centro O definido pela equação:

$$O = O_1 + (R - r_1) * (O_1O_2 / \| O_1O_2 \|) \qquad (6.6)$$

6.4 Testes de detecção de colisões entre elementos primitivos (simples e compostos) para obstáculos não planos

A seguir serão descritos os modos como são tratadas as possíveis colisões entre os elementos primitivos (simples e compostos) para o teste de colisão de um obstáculo não plano.

6.4.1. Teste entre esfera e esfera

Para determinar se existe ou não colisão entre duas esferas um primeiro passo é determinar a distância entre os centros das duas esferas. A distância entre os centros de duas esferas, com centros em O1 e O2 respectivamente, é dada por:

$$d(O_1, O_2) = ((x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2)^{1/2} \quad (6.7)$$

onde

$(x_1, y_1 \text{ e } z_1)$ é a posição do centro da primeira esfera;
$(x_2, y_2 \text{ e } z_2)$ é a posição do centro da segunda esfera.

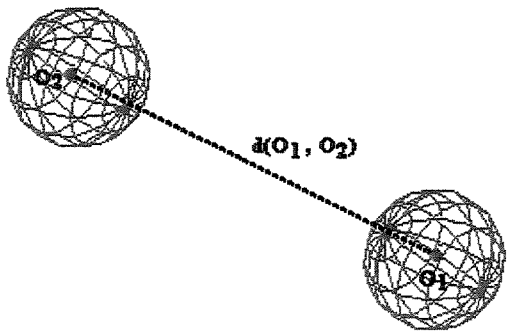


Figura 6.15: Distância entre os centros de duas esferas.

Para determinar se ocorre colisão entre duas esferas utiliza-se a equação:

$$T = r_1 + r_2 + \text{segurança} \quad (6.8)$$

se $T \geq d(O1, O2)$ não existe colisão entre as esferas

$T \leq d(O1, O2)$ existe colisão entre as esferas

onde

r_1 = raio da primeira esfera;

r_2 = raio da segunda esfera;

T = valor mínimo da distância entre os centros das esferas para o qual elas não se tocam.

(a) (b)

Figura 6.16: Teste entre duas esferas: (a) $T \geq d(O1, O2)$ e (b) $T \leq d(O1, O2)$.

O fator de segurança, incluído na equação 6.8, exprimi o grau de confiança que se tem no equipamento e na descrição do meio ambiente.

No caso do robô, em termos de equipamento, pode-se considerar um baixo fator devido à alta precisão e exatidão do robô. No caso do manipulador já devemos utilizar um fator maior, já que seu sistema de controle não é tão preciso, gerando um erro de posição maior.

A descrição do meio ambiente é o parâmetro de maior importância, já que normalmente são feitas simplificações na representação, para que o meio possa ser representado com esferas, cilindros e paralelepípedos. Devemos poder considerar, então, um fator que leve em conta os erros de representação.

São baseados neste teste todos os demais testes.

6.4.2. Teste entre esfera e cilindro

É considerado que sejam conhecidos a posição e tamanho da esfera, assim como a posição, dimensões e orientação do cilindro. A idéia é projetar a esfera na direção perpendicular ao eixo do cilindro e achar o ponto onde seria o novo centro. A partir daí deve-se substituir o cilindro por una esfera centrada naquele ponto e de raio igual ao raio do cilindro e realizar o teste de colisão entre duas esferas.

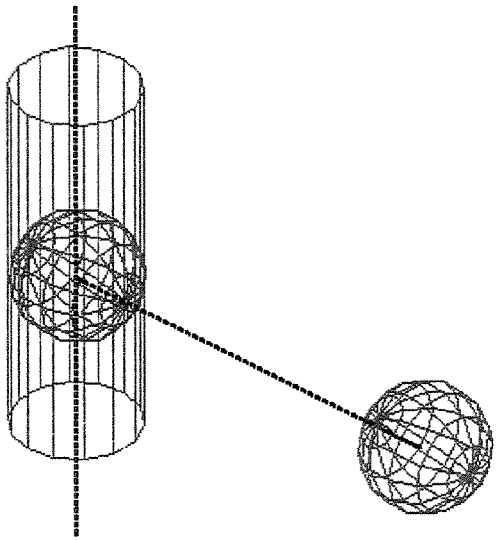


Figura 6.17: Teste de colisão entre esfera-cilindro.

6.4.3 Teste entre esfera e paralelepípedo

0 paralelepípedo é discretizado em cilindros, onde todos os cilindros possuem o comprimento igual à maior dimensão e diâmetros iguais à menor dimensão. Ao longo da dimensão intermediária serão inseridos os vários cilindros.

Uma vez feita a discretização, realiza-se o teste de colisão entre a esfera e cada um dos cilindros componentes do paralelepípedo, caindo no caso anterior de teste (entre esfera e cilindro).

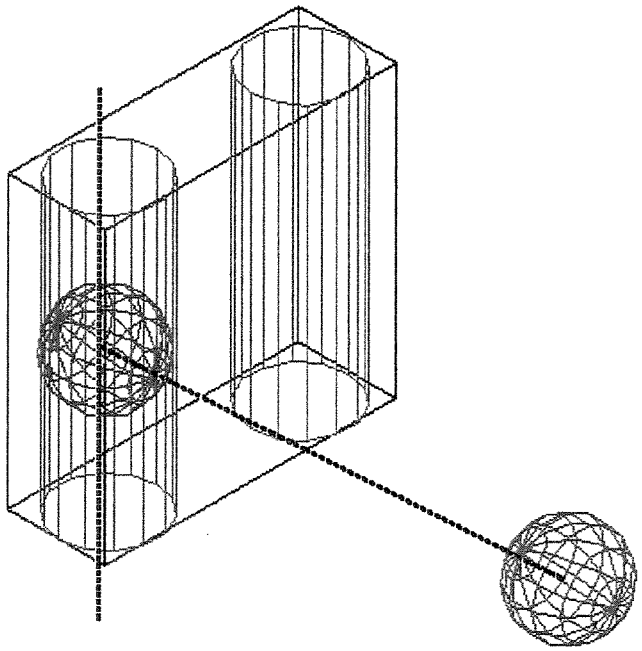


Figura 6.18: Teste de colisão entre esfera-paralelepípedo.

6.4.4 Teste entre cilindro e cilindro

No caso da iteração entre dois cilindros discretiza-se os cilindros em esferas e então realiza-se o teste entre as esferas.

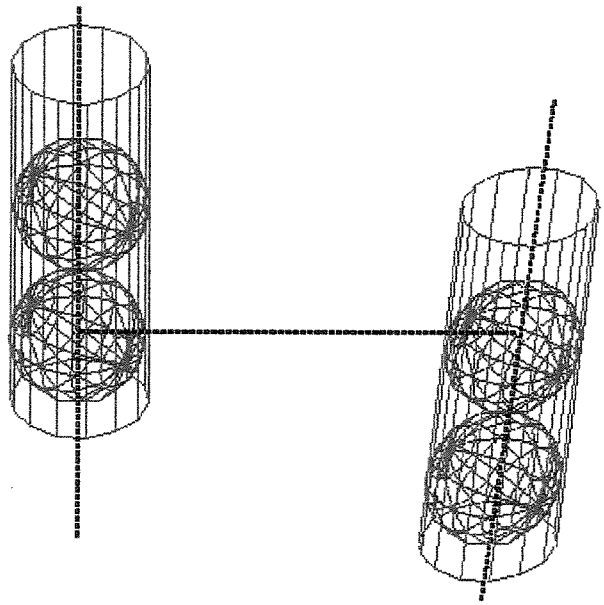


Figura 6.19: Teste de colisão entre cilindro-cilindro.

6.4.5 Teste entre cilindro e paralelepípedo

Discretiza-se o paralelepípedo em cilindros e, após isso, discretizam-se os cilindros em esferas e, então, realiza-se o teste entre as esferas.

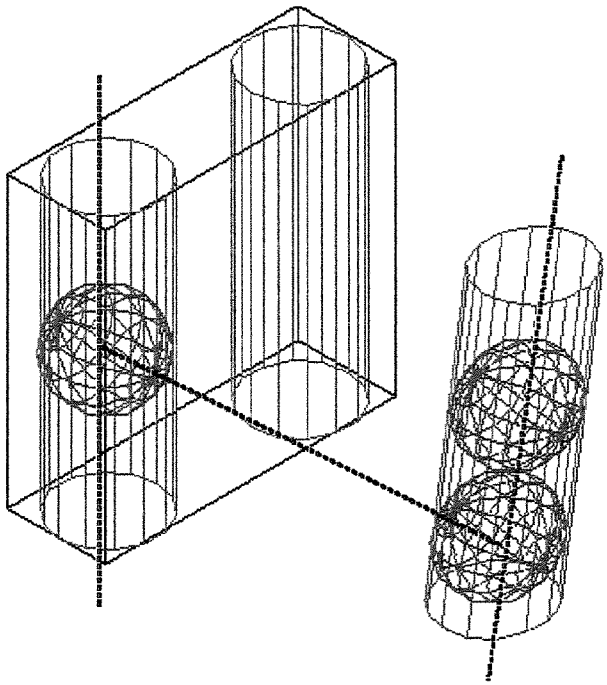


Figura 6.20: Teste de colisão entre cilindro-paralelepípedo.

6.4.6 Teste entre paralelepípedo e paralelepípedo

Discretizam-se os paralelepípedos em cilindros e após isso discretizam-se os cilindros em esferas e, então, realiza-se o teste de colisão entre as esferas.

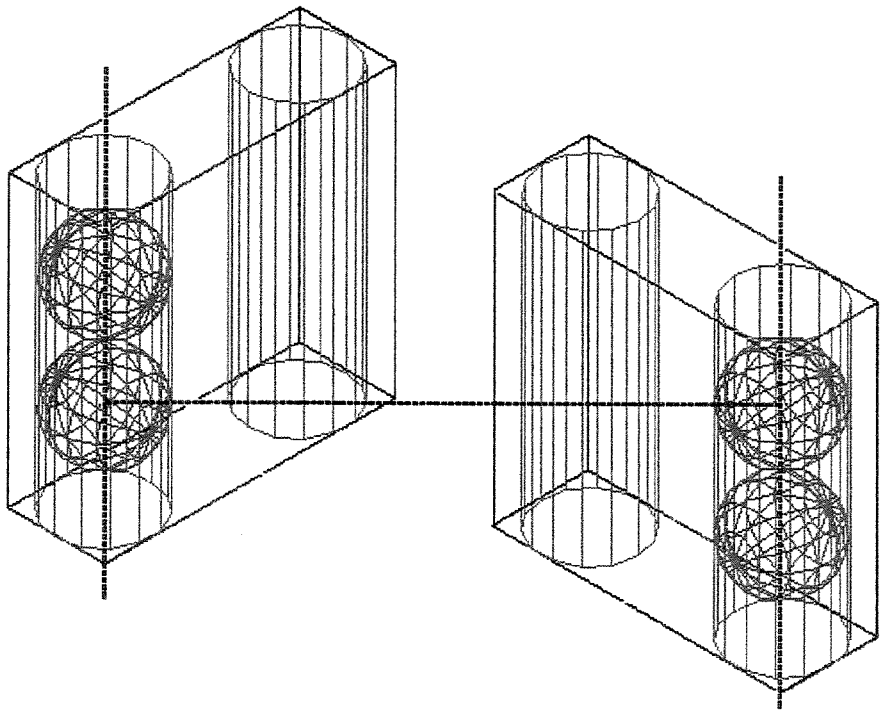


Figura 6.21: Teste de colisão entre paralelepípedo-paralelepípedo.

6.4.7 Teste entre elementos compostos e primitivos

Como o elemento composto é formado por um conjunto de elementos primitivos, basta realizar o teste entre o elemento primitivo em questão (esfera, cilindro e paralelepípedo) e cada um dos elementos componentes do elemento composto. Isto retorna aos testes já descritos.

6.4.8 Teste entre dois elementos compostos

Como os elementos são compostos e, portanto, formados por um conjunto de elementos primitivos, basta realizar o teste de todos os elementos primitivos (um de cada vez) de um dos elementos compostos com todos os elementos primitivos do outro, isto reduz aos testes anteriores.

6.5 Teste de detecção de colisões para obstáculos planos

Quando o obstáculo é considerado um plano (medida da altura e a largura é muito maior que a sua espessura), é calculada a distância entre um ponto, pertencente ao braço do robô, e o plano. A seguir será apresentado um método para a determinação da distância entre um ponto e um plano

6.5.1 Distância entre um ponto e um plano

Considere um plano definido pelos pontos P_1, P_2, P_3 e P_4 mostrado na figura abaixo:

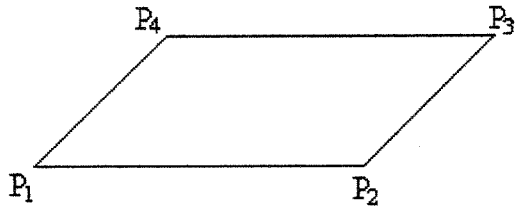


Figura 6.22:Plano definido pelos pontos P_1, P_2, P_3 e P_4.

onde

$P_1 = (x_{p1}, y_{p1}, z_{p1})$

$P_2 = (x_{p2}, y_{p2}, z_{p2})$

$P_3 = (x_{p3}, y_{p3}, z_{p3})$

$P_4 = (x_{p4}, y_{p4}, z_{p4})$

Seja:

Q um ponto no espaço dado por (x_q, y_q, z_q);

$\vec{A}$ um vetor contido no plano representado a reta P_2P_1;

$\vec{B}$ um vetor contido no plano representado a reta P_4P_1.

onde

$$\vec{A} = (x_{p2} - x_{p1})\ \hat{i} + (y_{p2} - y_{p1})\ \hat{j} + (z_{p2} - z_{p1})\ \hat{k} \tag{6.8}$$

$$\vec{B} = (x_{p4} - x_{p1})\ \hat{i} + (y_{p4} - y_{p1})\ \hat{j} + (z_{p4} - z_{p1})\ \hat{k} \tag{6.9}$$

Seja $\vec{V}$ o vetor que representa a reta QP_1 isto é:

$$\vec{V}(QP_1) = (x_{p1} - x_q)\ \hat{i} + (y_{p1} - y_q)\ \hat{j} + (z_{p1} - z_q)\ \hat{k} \tag{6.10}$$

e $\vec{N}$ um vetor normal ao plano dado por:

$$\vec{N} = \vec{A} \times \vec{B} \tag{6.11}$$

A distância, d, entre o ponto P e o plano é dado por:

$$d = (\vec{N} \cdot \vec{V}(PQ))/ \|\vec{N}\| \tag{6.12}$$

Na figura 6.23 pode-se observar os vetores definidos anteriormente.

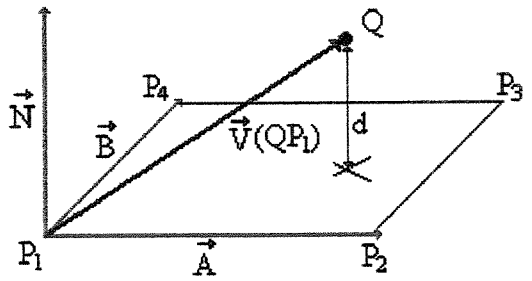


Figura 6.23: Vetores utilizados para o cálculo de d.

6.5.2 Teste de colisão

Para a construção gráfica do robô está sendo utilizado o modelo geométrico do robô, obtido através do método de vetores locais, deste modo é conhecida a posição espacial de diversos pontos (pontos de interesse) do braço do robô.

Para determinar se existe ou não colisão entre um elo do robô e o obstáculo (considerado plano) determina-se a distância entre estes pontos do braço, no mínimo dois, e o obstáculo, utilizando-se a equação abaixo:

$$T = d - L - \text{segurança} \qquad (6.13)$$

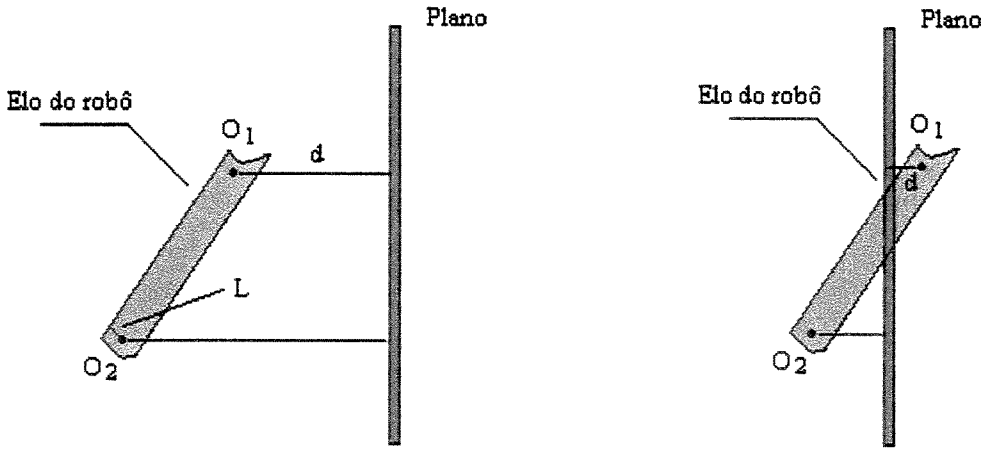
se $T > 0$ não ocorre colisão;

$T \leq 0$ ocorre colisão.

onde d = distância do ponto ao plano, equação 6.12;

L = distância do ponto a borda do elo, indicado na figura 6.24 (a).

Se a distância de pelo menos um dos pontos for menor que zero, existe a colisão entre o braço do robô e o obstáculo. O fator de segurança, incluído na equação 6.13, exprime o grau de confiança que se tem no equipamento e na descrição do meio ambiente. A figura 6.24 exemplifica este método.



(a) (b)

Figura 6.24: Determinação da existência ou não de colisão: (a) não existe colisão, $T > 0$ (b) existe colisão $T \leq 0$.

No capítulo seguinte serão apresentados os módulos desenvolvidos de geração de trajetórias, a partir dos algoritmos apresentados neste capítulo, e o de simulação gráfica

Capítulo 7

Implementação dos Módulos e Integração Computacional

A partir dos algoritmos e métodos apresentados no decorrer dos capítulos anteriores foi implementado um programa computacional "off-line" composto por três módulos:

- módulo Trajeto: para a geração de trajetórias;
- módulo Obstáculo: para a geração do ambiente de trabalho (obstáculos e ferramentas nos quais o robô atua).
- módulo Simula: para a visualização do robô realizando sua tarefa em seu ambiente de trabalho ("off-line") e de envio de sinais ao robô para a realização das tarefas;

Neste capítulo serão apresentados os módulos e uma breve descrição da linguagem utilizada para a implementação dos mesmos.

7.1 Linguagem ADA

Os módulos foram gerados em linguagem ADA (Watt, 1987). Esta linguagem apresenta alto grau de estruturação, o que permite simplificações na programação de tarefas com alto grau de complexidade.

Esta linguagem permite que o programa seja dividido em pacotes, formando uma biblioteca. Desta forma, os pacotes podem ser testados separadamente e incorporados em diferentes módulos, permitindo uma integração entre os mesmos.

Para este trabalho foram implementados dois módulos para testes dos pacotes. Estes módulos são apresentados no Anexo V.

A hierarquia proposta para a implementação dos pacotes foi:

- foi definido um primeiro nível de funções elementares para a aplicação;

- a partir disso foram implementados outros pacotes representando uma camada superior das funcionalidades, que serão especificadas previamente.

Na linguagem ADA, é necessário termos para cada pacote um programa que defina as especificações (citação de 'procedures' e variáveis importantes do pacote) e um outro que seja o corpo do pacote (definição e descrição das 'procedures' e variáveis).

Os módulos podem ser utilizados para outros robôs alterando-se apenas o os pacotes *Parameters* que contêm informações tais como limites físicos das juntas, dimensões, entre outras, *Robô* que contém as informações referentes à construção gráfica do robô e *Model* que contém informações relativas ao modelo do robô.

7.1.1 Pacotes implementados

Cada pacote implementado possui uma função específica como pode ser notado acima. No Anexo VI são descritas as funções de cada pacote.

A figura 7.1 mostra a organização da biblioteca gerada a partir dos pacotes implementados.

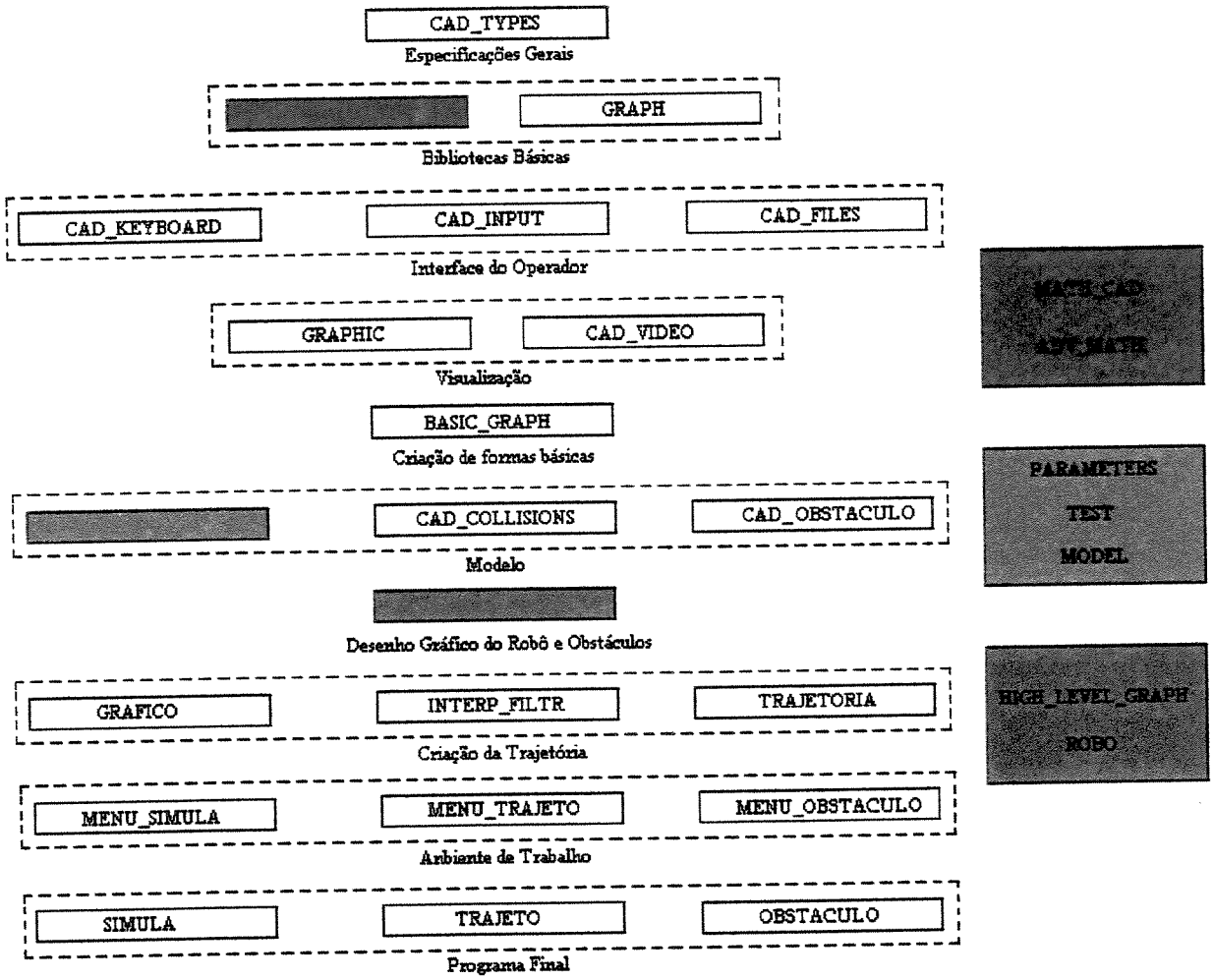


Figura 7.1: Organização da biblioteca.

7.2 Módulos implementados

Os módulos implementados permitirão:

- redução do tempo em que o robô está fora da linha de produção;
- o programador não precisará entrar em contato com o ambiente de trabalho;
- integração com sistemas CAD-CAM permitindo uma maior integração entre as fases de projeto e de produção e, conseqüentemente, reduzindo o tempo do processo de produção;

- visualização do robô e do ambiente de trabalho;

- segurança na geração de trajetórias através da detecção de possíveis colisões com o meio ambiente.

A seguir serão descritos os módulos implementados, suas funções e características.

7.2.1 Módulo de geração de trajetórias - MÓDULO TRAJETO

Este módulo é responsável pela geração de trajetórias angulares utilizando dois modos:

- utilizando o modelo cinemático inverso no qual são dadas apenas a posição e orientação finais do elemento terminal do robô;

- através de interpolação e filtragem de pontos de passagem angulares (estes pontos de passagem podem ser obtidos pelo módulo Simula).

Este módulo permite que uma trajetória seja composta de um ou de vários segmentos; isto torna possível a obtenção de trajetórias com um alto grau de complexidade (cada segmento pode ser discretizado de uma forma diferente obtendo, desta forma, segmentos com perfis diferentes). Este módulo gera trajetórias lineares e em forma de semicírculo mas outras formas podem ser obtidas definindo novas funções matemáticas para as trajetórias.

Neste módulo um primeiro teste de colisão realizado é o do robô/robô. A tela principal deste programa é mostrada na figura 7.5.

A figura 7.2 mostra o fluxograma dos pacotes envolvidos na geração deste módulo. Os pacotes em cor azul são os pacotes que são comuns a todos os módulos implementados.

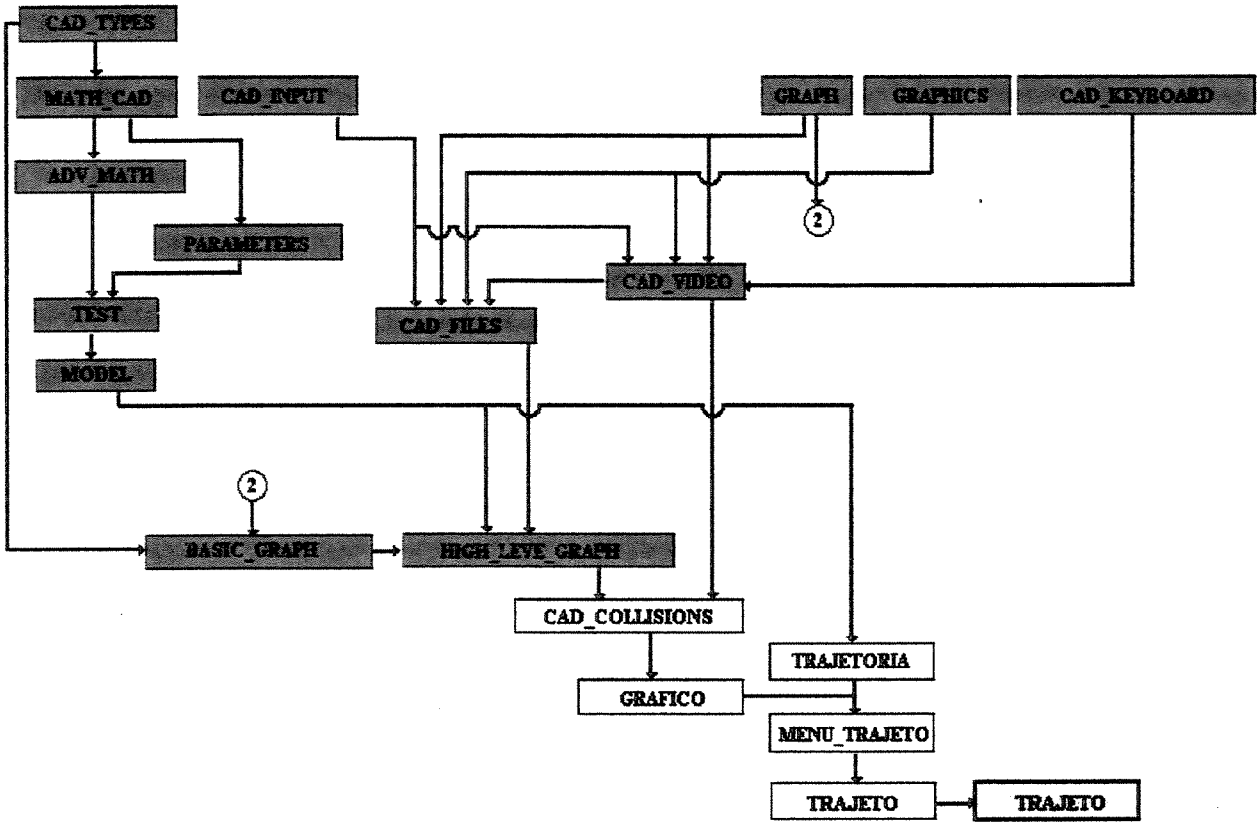


Figura 7.2: Fluxograma dos pacotes para a geração do módulo Trajeto.

7.2.2 Módulo de criação do ambiente de trabalho – MÓDULO OBSTACULO

Este módulo tem como objetivo a compilação dos modelos geométricos das ferramentas e do ambiente de trabalho. Ele gera arquivos contendo diferentes tipos de meio ambiente e ferramentas utilizadas durante uma operação robotizada. Apresenta as seguintes características:

- utilização dos elementos primitivos básicos (esfera, cilindro e paralelepípedo) para a representação de ambientes e ferramentas complexas;

- imagem gráfica do meio ambiente e das ferramentas em três posições possíveis planta, face e esquerda (ao mesmo tempo ou combinações entre elas a figura 7.8 mostra três possibilidades de combinação);

A partir da criação, em editor ASCII, de um arquivo de definição do obstáculo/ferramenta, este será lido pelo módulo Obstáculo, responsável pela criação de um arquivo binário obstáculo/ferramenta. Um fluxograma para a geração deste módulo é apresentado no Anexo 1, figura I.4. A tela principal deste programa é mostrada na figura 7.6.

A figura 7.3 mostra o fluxograma dos pacotes envolvidos na geração deste módulo.

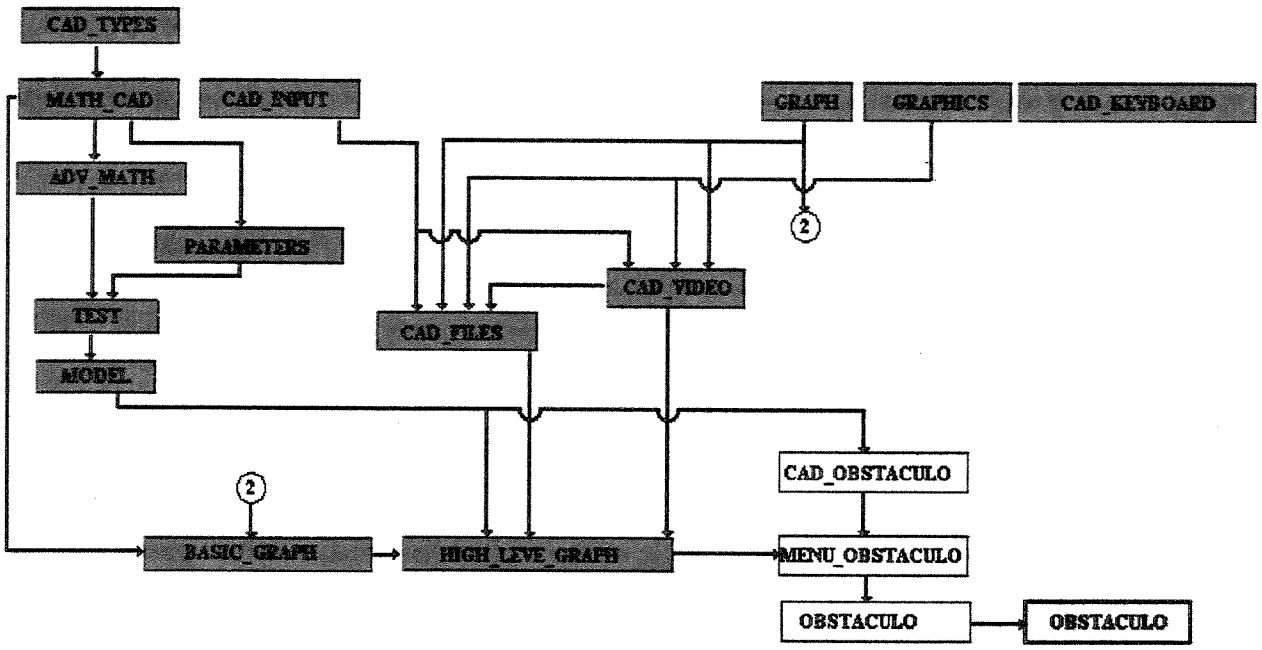


Figura 7.3: Fluxograma dos pacotes para a geração do módulo Obstáculo.

7.2.3 Módulo de simulação – MÓDULO SIMULA

Este módulo possibilita a edição gráfica e interativa da trajetória, realizando a simulação e visualização do cenário completo de atuação, o qual contém o dispositivo robótico, ferramentas dedicadas, acessórios, etc. A tela principal deste programa é mostrada na figura 7.7 e a figura 7.4 mostra o fluxograma dos pacotes envolvidos na geração deste módulo.

Apresenta, ainda, as seguintes características:

- simplicidade na utilização isto é, possui uma interface amigável com o operador;

- possibilidades de testes de colisão, são permitidos os seguintes testes: fim de curso, robô/robô, robô/ferramenta, robô/painel, robô/módulo, robô/meio ambiente, ferramenta/painel, ferramenta/módulo, ferramenta/meio ambiente, todos os testes (a finalidade de cada um destes testes são descritos no Anexo XIII). É permitida a alteração da variável de segurança prevista nas equações, 6.8 e 6.13.;

- imagem gráfica em três posições possíveis planta, face e esquerda (ao mesmo tempo ou combinações entre elas a figura 7.9 mostra três possibilidades de combinação) e simulação da tarefa no cenário;

- possibilidade de movimentação cartesiana a partir de um toque no teclado;

- possibilidade de movimentação das juntas (de incrementos pré definidos) a partir de um toque no teclado;

- permite que as ferramentas ou os obstáculos gerados pelo módulo Obstáculo sejam utilizados nas simulações.

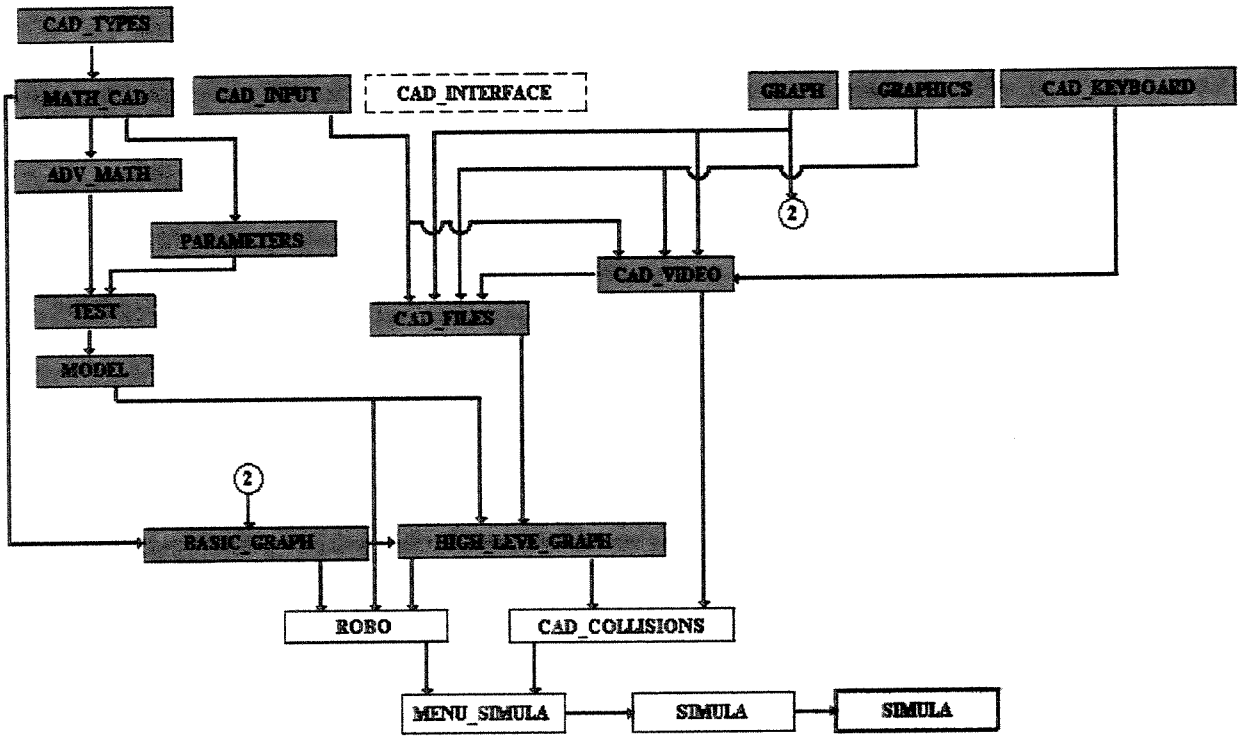


Figura 7.4: Fluxograma dos pacotes para a geração do módulo Simula.

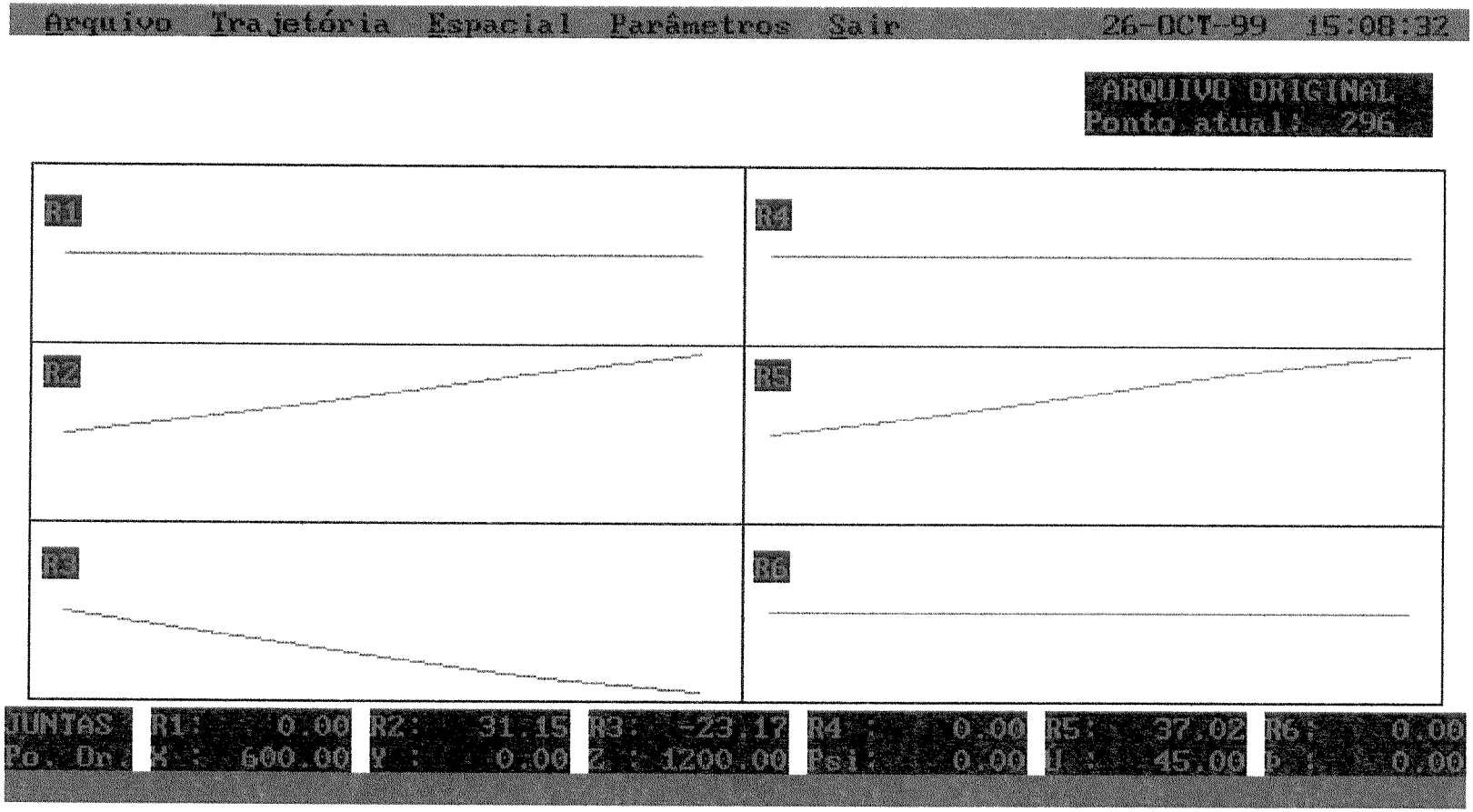


Figura 7.5: Tela principal do módulo Trajeto.

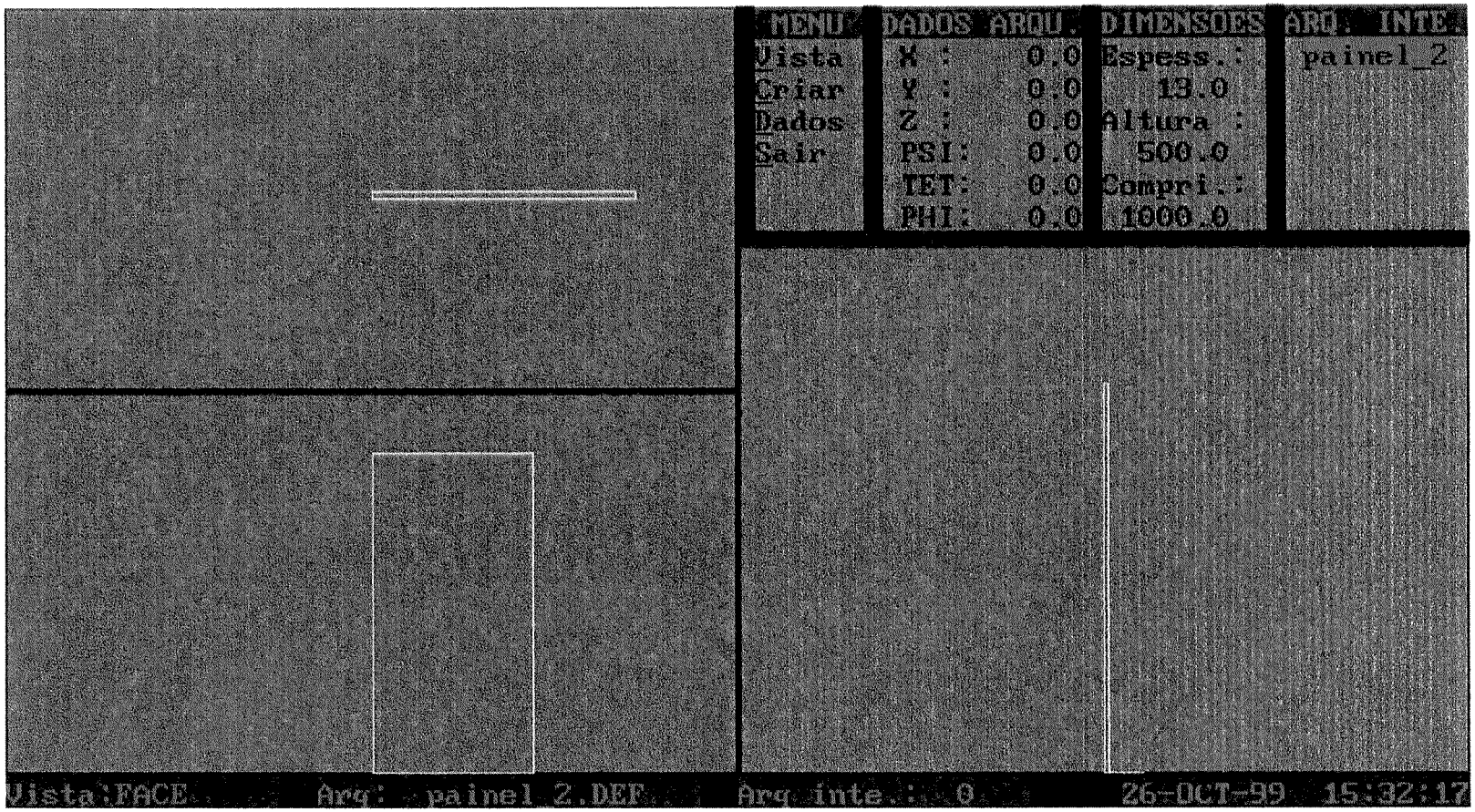


Figura 7.6: Tela principal do módulo Obstáculo.

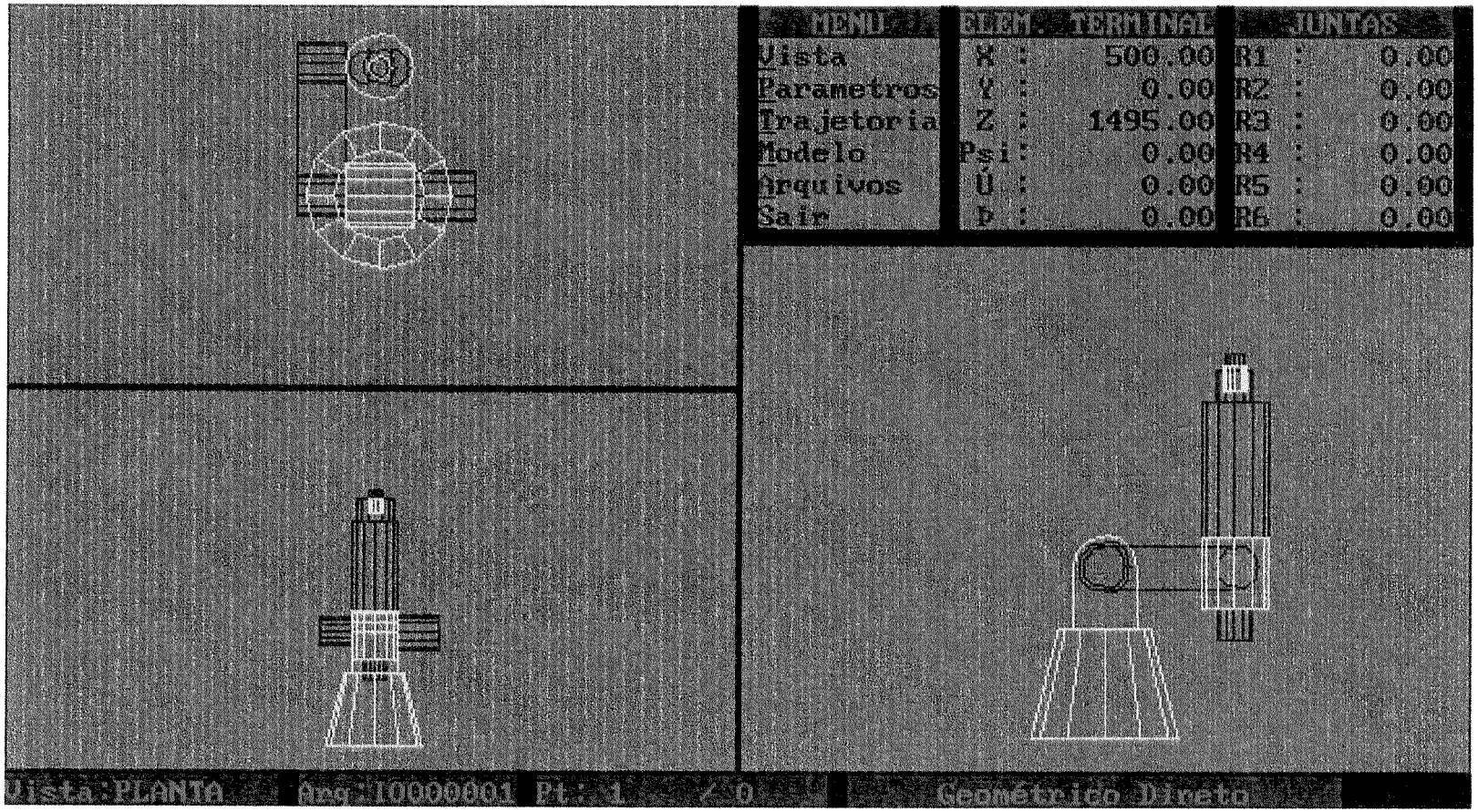


Figura 7.7: Tela principal do módulo Simula para o robô Manutec.

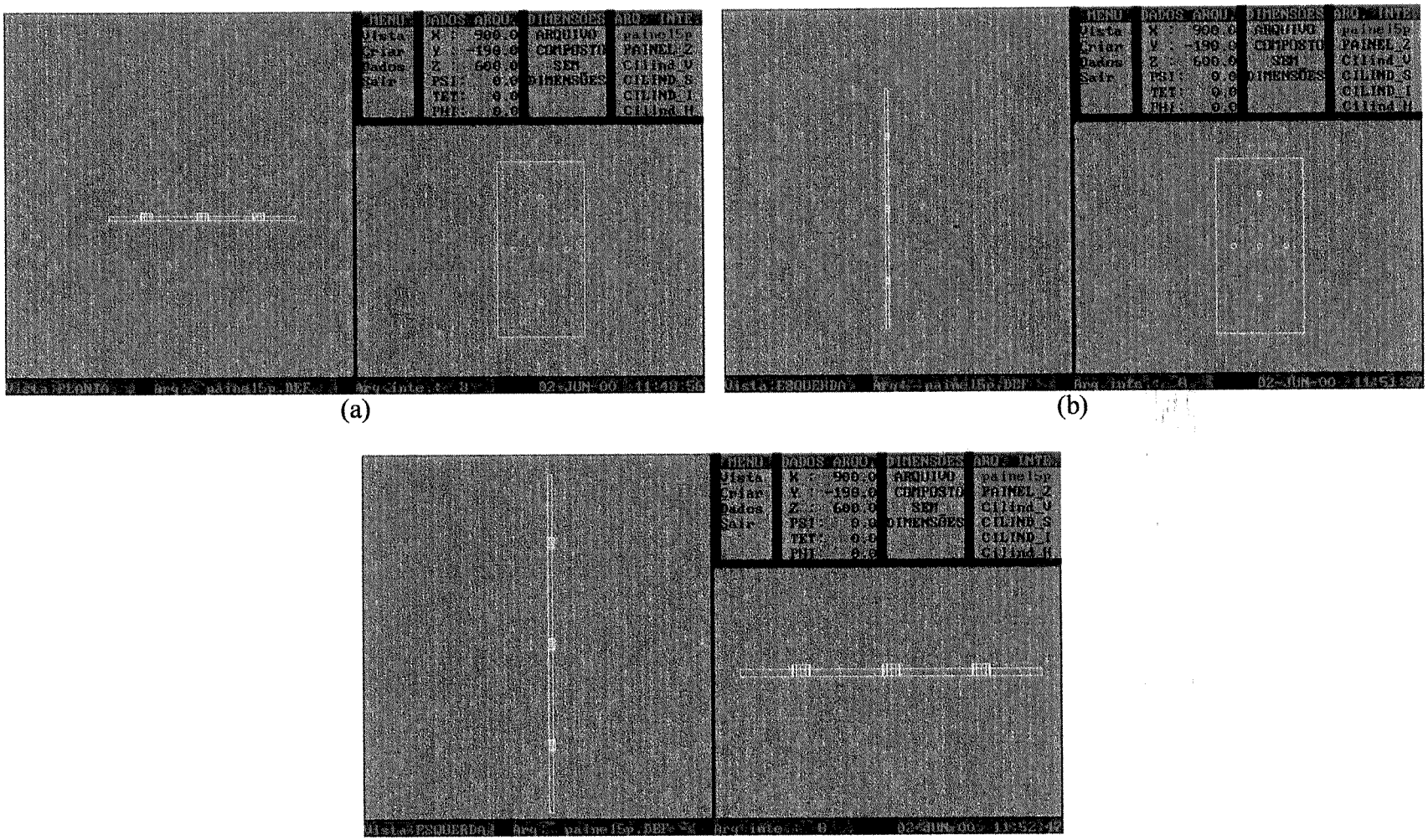


Figura 7.8: Três possíveis combinações de Vistas no módulo Obstáculo: (a) planta-face, (b) esquerda-face e (c) esquerda-planta.

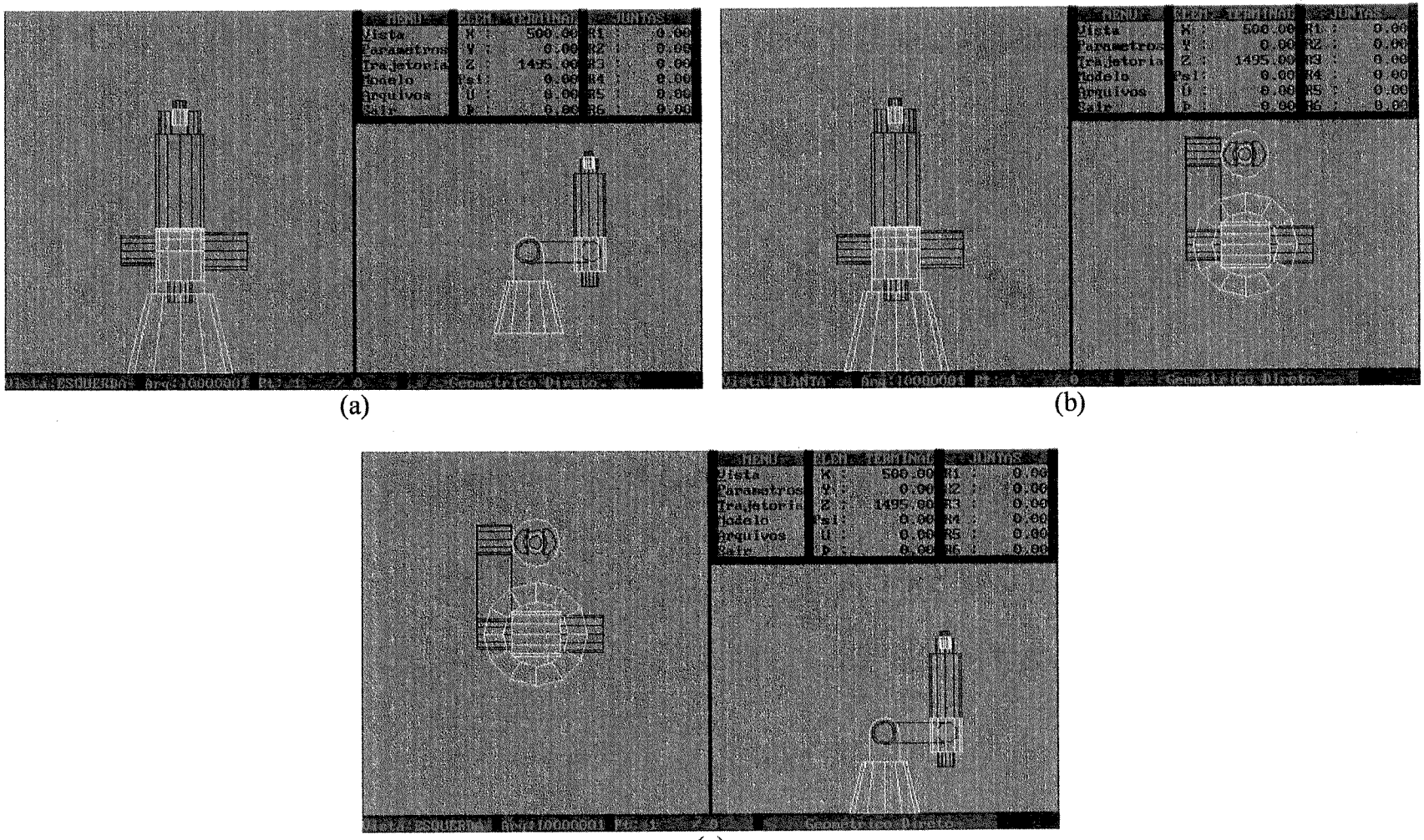


Figura 7.9: Três possíveis combinações de Vistas no módulo Simula: (a) face-esquerda, (b) face-planta e (c) planta-esquerda.

7.3 Menus dos módulos

As funções dos menus permitem, entre outras, alterar as vistas dos objetos, no caso dos módulos Simula e Obstáculo, inserir e deletar pontos nos arquivos de trajetórias e também mover-se entre os pontos da trajetória, no caso dos módulos Trajeto e Simula, e muitas outras.

No Anexo VIII são apresentadas mais detalhadamente todas as funções dos menus de cada um dos módulos implementados.

7.4 Fluxo de dados entre os módulos - iteratividade

A figura 7.10 mostra o fluxo de dados entre os módulos. Os módulos Simula e Trajeto lêem dados de arquivos que contêm as trajetórias angulares, que podem ser geradas em ambos os módulos. Estes dados são escritos em arquivos no formato ASCII.

Os dados gerados pelo módulo Obstáculo, referentes a obstáculos criados, podem ser lidos apenas pelo módulo Simula. Estes dados são escritos em arquivos no formato binário.

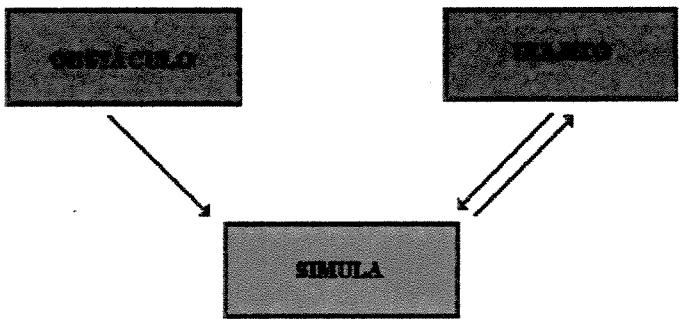


Figura 7.10: Fluxo de dados entre os módulos.

A tabela 7.1 mostra os tipos de arquivos que podem ser lidos e gerados por cada módulo.

	Leitura de arquivos de			Geração de arquivos de	
MÓDULO	trajetória angular (ascii)	obstáculos (ascii)	obstáculos (binários)	trajetória angular (ascii)	obstáculos (binários)
Trajeto	sim	não	não	sim	não
Obstáculo	não	sim	não	não	sim
Simula	sim	não	sim	sim	não

Tabela 7.1: Tipos de arquivos que podem ser lidos e gerados por cada módulo.

Os arquivos citados acima são descritos detalhadamente no Anexo XI.

7.5 Teclas rápidas utilizadas pelos módulos

Estas teclas têm como objetivo realizar um acesso rápido a determinadas funções que podem ser encontradas nos menus dos módulos Trajeto, Simula e Obstáculo. Estas teclas são apresentadas no Anexo VII.

Estas teclas são identificadas em um teclado, através de figuras, facilitando em muito a sua utilização. Este teclado é apresentado na figura VII.1 (Anexo VII).

7.6 Tela do módulo Simula para o manipulador Kraft

A figura 7.11 mostra a tela principal do módulo Simula para o manipulador Kraft. Este módulo foi gerado alterando-se apenas, como mencionado anteriormente, os pacotes *Parameters* que contêm informações tais como limites físicos das juntas, dimensões, entre outras, *Robô* que contém as informações referentes à construção gráfica do robô e *Model* que contém informações relativas ao modelo do robô.

As funções dos menus dos módulos são idênticos para ambos robôs (Manutec e Kraft). As figuras dos demais módulos Trajeto e Obstáculo são iguais aos das figuras 7.5 e 7.5 respectivamente.

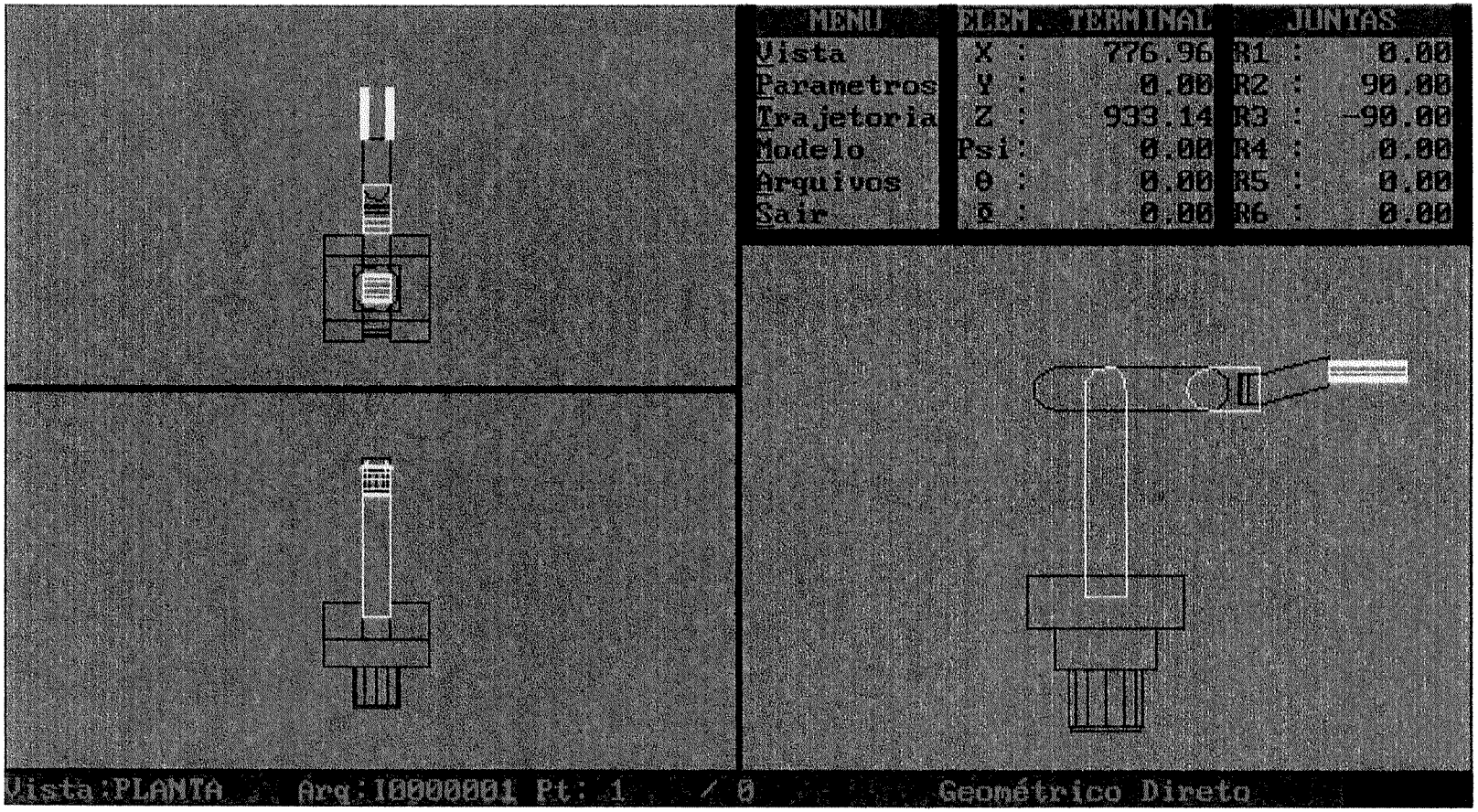


Figura 7.11: Tela principal do modulo Simula para o manipulador Kraft.

7.7 Programa para o gerenciamento dos programas "off-line" e "on-line"

Com a finalidade de facilitar o uso do pacote computacional "off-line" (Simula, Trajeto e Obstáculo) e do programa de supervisão e controle (CMK - "on-line") foi implementado um programa que possui a função de executar estes programas a figura 7.12 mostra a tela deste programa.

Neste capítulo foram apresentados os módulos Trajeto, Simula e Obstáculo que compõem o pacote computacional "off-line" de programação de robôs.

No capítulo seguinte serão apresentadas as simulações realizadas para testar os módulos Trajeto, Simula e Obstáculo e mostrar a iteratividade entre eles.

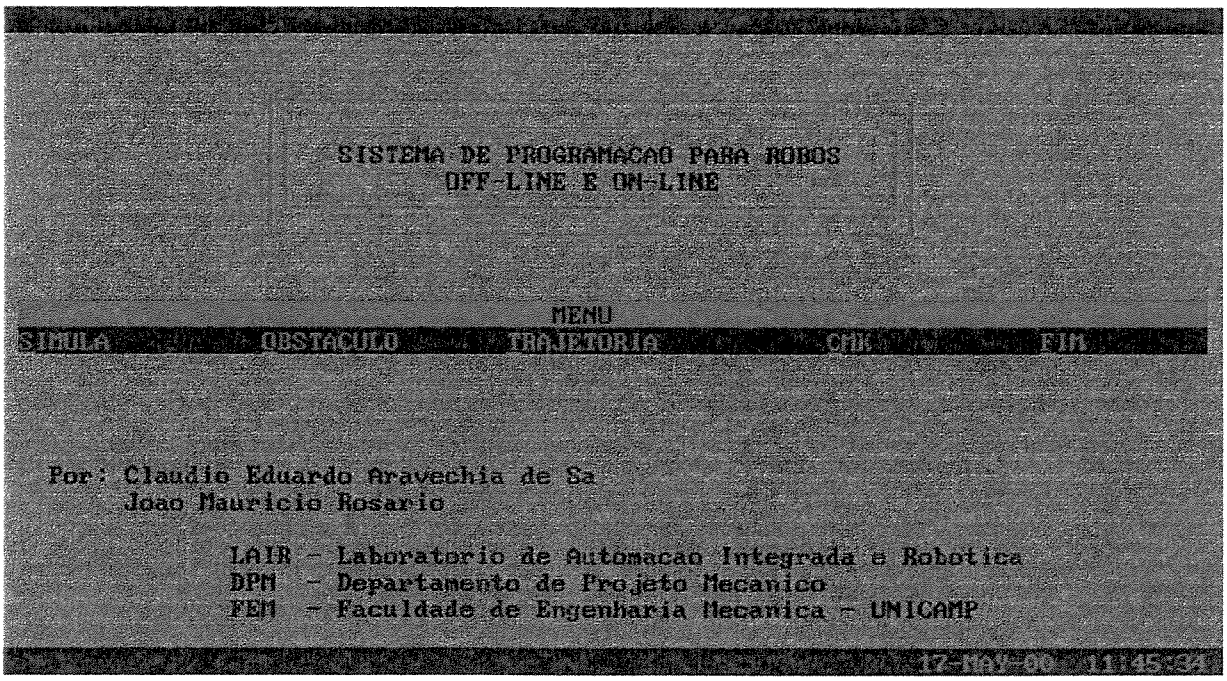


Figura 7.12: Tela do modulo Sistema.

Capítulo 8

Simulações e Resultados Obtidos

Neste capítulo serão apresentadas as simulações realizadas para mostrar a iteratividade entre os módulos Trajeto, Simula e Obstáculo e os resultados obtidos.

Serão apresentadas simulações em que o robô segue as extremidades de um círculo e de um retângulo, de inserir pinos em painéis e a simulação de giro do punho do robô, simulando o fechamento ou abertura de uma válvula.

8.1 Simulações

Com o objetivo de testar os módulos desenvolvidos, foram realizadas simulações de criação de trajetórias, painéis e visualizações gráficas que serão apresentadas a seguir.

As simulações realizadas pelo módulo Simula utilizam os arquivos gerados pelos módulos Obstáculo e Trajeto. As trajetórias foram obtidas no módulo Trajeto (Simulações de 1 a 4) e por interpolação e filtragem (Simulação 5).

No Anexo IX são apresentados os arquivos de definição, utilizados pelo módulo Obstáculo, para a construção de dois painéis, e o desenho dos mesmos, que serão utilizados nas simulações a seguir.

8.1.1 Seguir as extremidades de um painel retangular – Simulação 1

A trajetória de referência é constituída de cinco segmentos a figura 8.1 mostra a trajetória angular obtida para esta operação. A tabela 8.1 apresenta as posições e orientações iniciais e finais de cada segmento desta trajetória (posição em mm e orientação em graus) no formato px, py, pz, psi, teta e phi. Também apresenta o número de pontos (conjunto de ângulos) gerados para cada segmento. O número total de pontos gerados é de 1907. A discretização utilizada em todos os segmentos desta trajetória é linear.

Segmento	Posição (mm) e orientação (graus)		N Pto
	sai de	vai para	
1	500, 0, 1495, 0, 0, 0	900, 0, 1100, 0, 89, 0	557
2	900, 0, 1100, 0, 89, 0	900, 0, 1500, 0, 89, 0	450
3	900, 0, 1500, 0, 89, 0, 0	900, 200, 1500, 0, 89, 0	225
4	900, 200, 1500, 0, 89, 0	900, 200, 1100, 0, 89, 0	450
5	900, 200, 1100, 0, 89, 0	900, 0, 1100, 0, 89, 0	225

Tabela 8.1: Posições iniciais e finais da trajetória (posição e orientação)e o número de pontos para cada segmento (n pto).

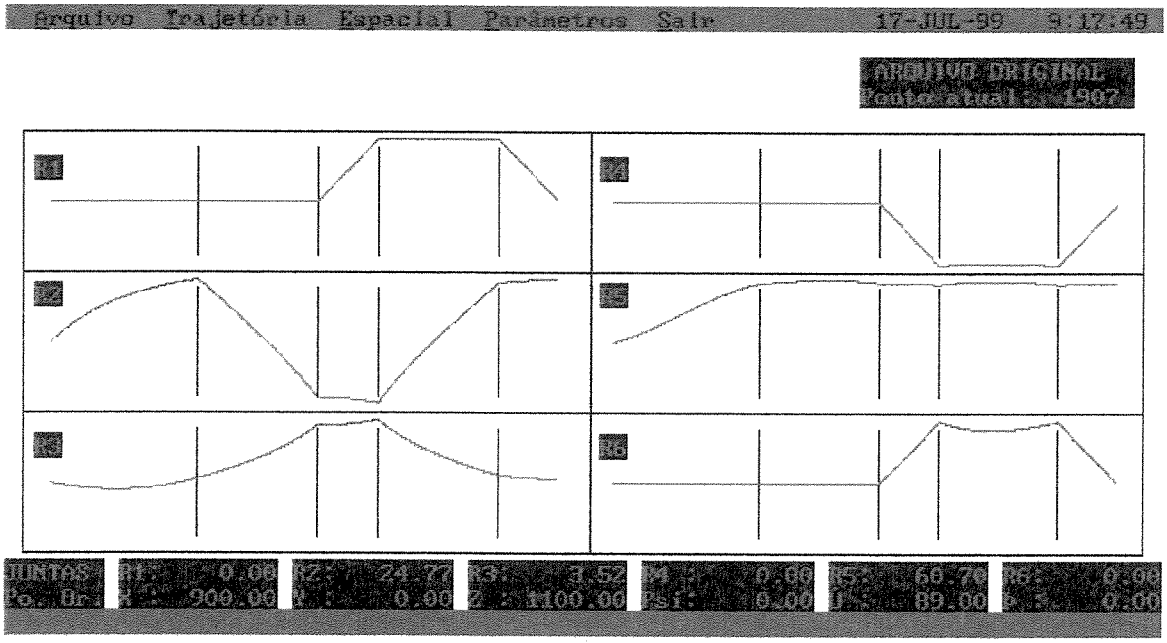


Figura 8.1: Trajetória angular obtida.

As figuras de 8.2 a 8.7 mostram respectivamente a configuração espacial do robô, no módulo Simula, no final de cada segmento.

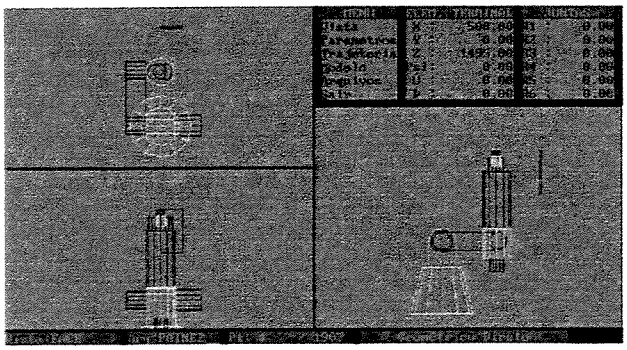


Figura 8.2: Ponto inicial de saída do robô.

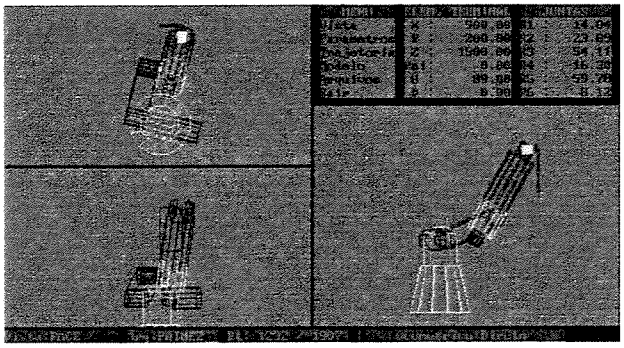


Figura 8.5: Configuração espacial do robô no final do segmento 3.

Figura 8.3: Configuração espacial do robô no final do segmento 1.

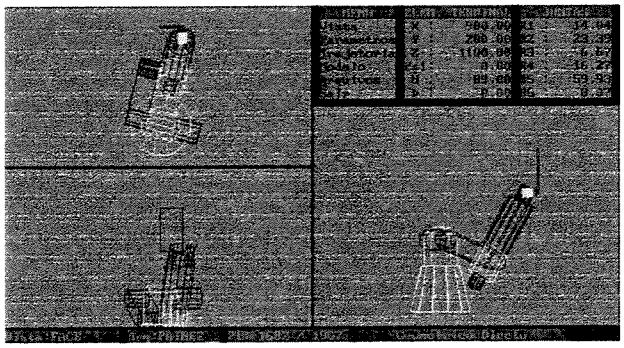


Figura 8.6: Configuração espacial do robô no final do segmento 4.

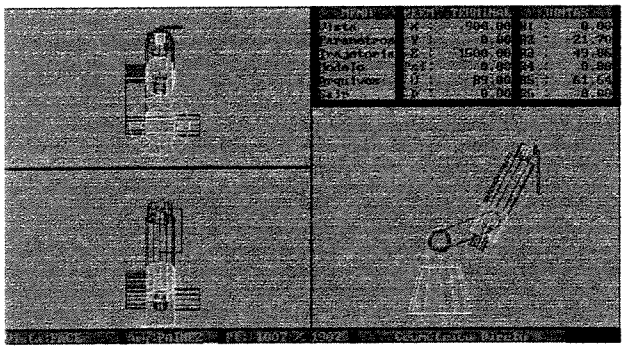


Figura 8.4: Configuração espacial do robô no final do segmento 2.

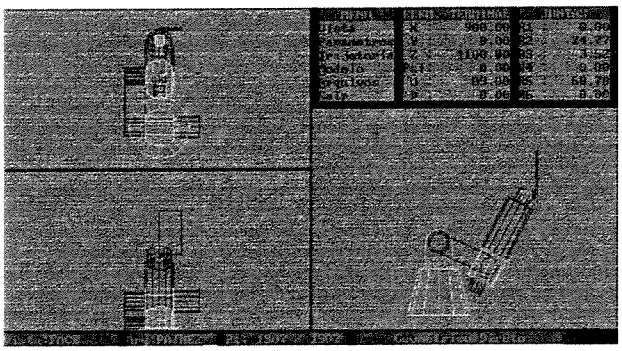


Figura 8.7: Configuração espacial do robô no final do segmento 5.

8.1.2 Seguir as extremidades de um círculo – Simulação 2

A trajetória de referência é constituída de três segmentos. A figura 8.8 mostra a trajetória angular obtida para esta operação. A tabela 8.2 apresenta as posições iniciais e finais de cada segmento desta trajetória (posição em mm e orientação em graus) no formato px, py, pz, psi, teta e phi. Também apresenta o número de pontos (conjunto de ângulos) gerados para cada segmento. O número total de pontos gerados é de 1644.

Segmento	Posição (mm) e orientação (graus)		N Pto
	sai de	vai para	
1	500, 0, 1495, 0, 0, 0	850, -200, 1400, 0, 89, 0	376
2	850, -200, 1400, 0, 89, 0	850, 200, (1400), 0, 89, 0	634
3	850, 200, (1400), 0, 89, 0	850, -200, (1400), 0, 89, 0	634

Tabela 8.2: Posições iniciais e finais da trajetória (posição e orientação) e o número de pontos para cada segmento (n pto).

No segmento 2 a discretização da trajetória utilizada é a de um semicírculo no plano Y-Z, raio positivo. No segmento 3 a discretização da trajetória utilizada é a de um semicírculo no plano Y-Z, raio negativo. O círculo possui centro em: 850, 0, 1400.

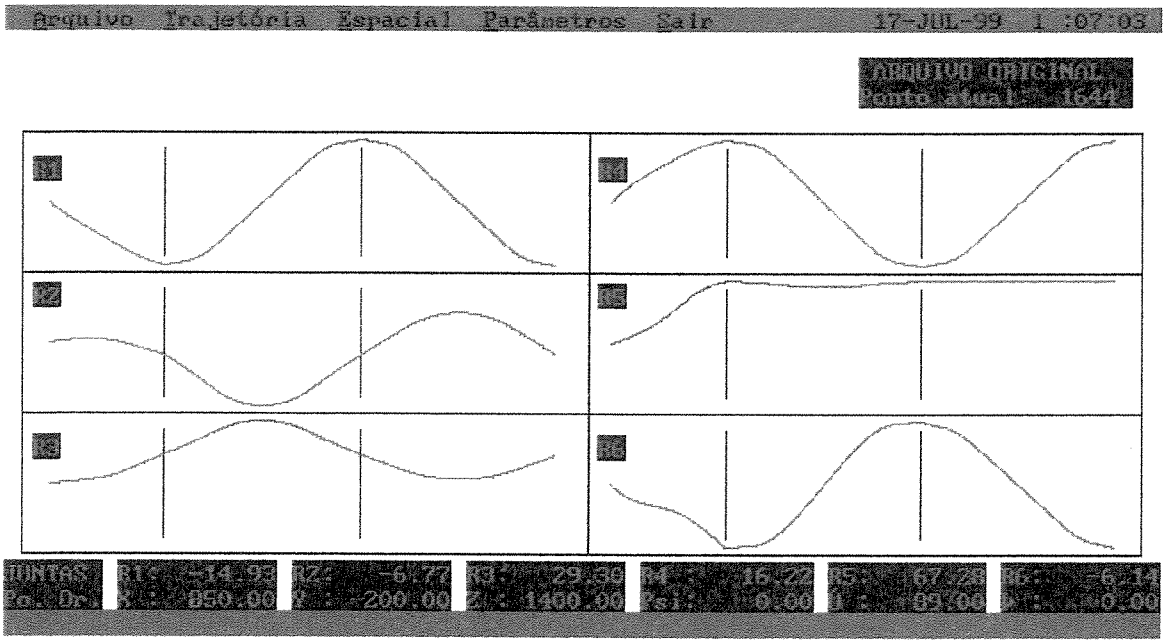


Figura 8.8: Trajetória angular obtida.

As figuras de 8.9 a 8.12 mostram, respectivamente, a configuração espacial do robô, no módulo Simula, no final de cada segmento.

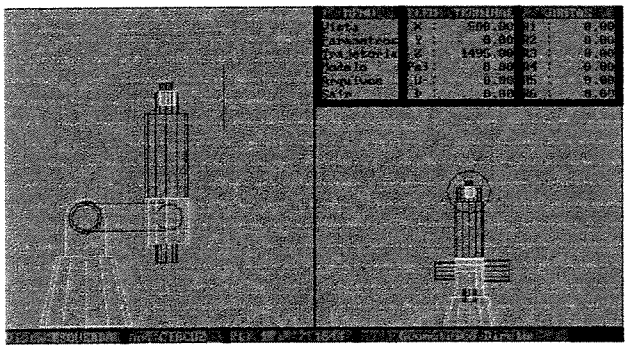


Figura 8.9: Ponto inicial de saída do robô.

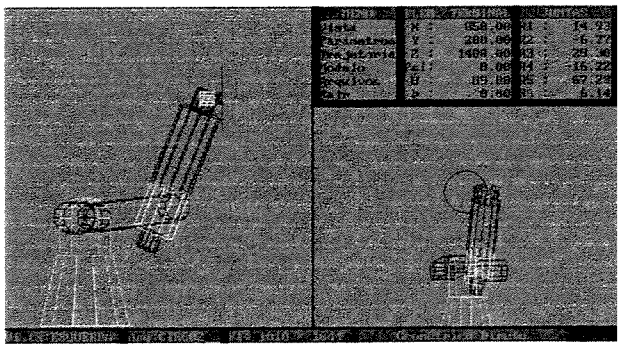


Figura 8.11: Configuração espacial do robô no final do segmento 2.

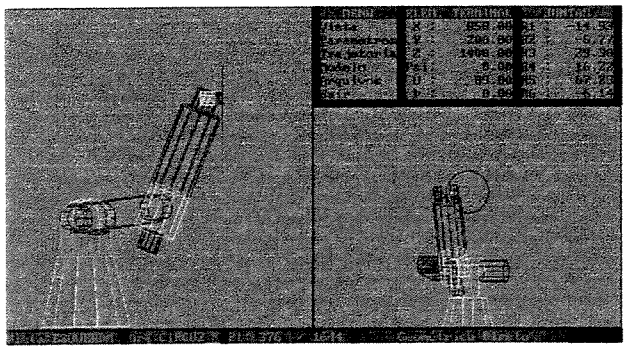


Figura 8.10: Configuração espacial do robô no final do segmento 1.

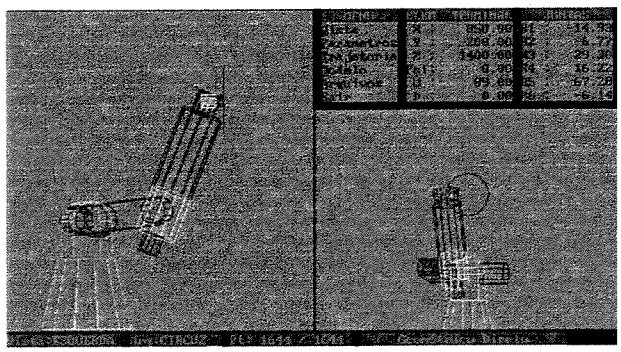


Figura 8.12: Configuração espacial do robô no final do segmento 3.

8.1.3 Insere um pino em dois painéis distintos – Simulação 3

A trajetória de referência é constituída de vinte e cinco segmentos. A figura 8.13 mostra a trajetória angular obtida para esta operação. A tabela 8.3 apresenta as posições iniciais e finais de cada segmento desta trajetória (posição em mm e orientação em graus) no formato px, py, pz, psi, teta e phi. Também apresenta o número de pontos (conjunto de ângulos) gerados para cada segmento. O número total de pontos gerados é de 8558.

A discretização utilizada em todos os segmentos desta trajetória é linear.

Segmento	Posição (mm) e orientação (graus)		N Pto
	sai de	vai para	
1	500, 0, 1495, 0, 0, 0	650, 60, 1400, 0 89, 0	186
2	650, 60, 1400, 0 89, 0	900, 60, 1400, 0 89, 0	280
3	900, 60, 1400, 0 89, 0	650, 60,1400, 0, 89, 0	280
4	650, 60, 1400, 0 89, 0, 0	650, 60, 1100, 0, 89, 0	335
5	650, 60, 1100, 0, 89, 0	900, 60, 1100, 0, 89, 0	280
6	900, 60, 1100, 0 89, 0, 0	650, 60, 1100, 0, 89, 0	280
7	650, 60, 1100, 0, 89, 0	650, 60, 800, 0, 89, 0	336
8	650, 60, 800, 0 89, 0, 0	900, 60, 800, 0, 89, 0	280
9	900, 60, 800, 0, 89, 0	650, 60, 800, 0, 89, 0	280
10	650, 60, 800, 0 89, 0, 0	500, 0, 1495, 0, 10, 0	716
11	500, 0, 1495, 0, 0, 0	100, 550, 1100, -89, 0, 0	684
12	100, 550, 1100, -89, 0, 0	100, 800, 1100, -89, 0, 0	280
13	100, 800, 1100, -89, 0, 0	100, 550, 1100, -89, 0, 0	280
14	100, 550, 1100, -89, 0, 0	400, 550, 1100, -89, 0, 0	335
15	400, 550, 1100, -89, 0, 0	400, 800, 1100, -89, 0, 0	280
16	400, 800, 1100, -89, 0, 0	400, 550, 1100, -89, 0, 0	280
17	400, 550, 1100, -89, 0, 0	700, 550, 1100, -89, 0, 0	335
18	700, 550, 1100, -89, 0, 0	700, 800, 1100, -89, 0, 0	280
19	700, 800, 1100, -89, 0, 0	700, 550, 1100, -89, 0, 0	280
20	700, 550, 1100, -89, 0, 0	500, 0, 1495, -10, 0, 0	681
21	500, 0, 1495, -0, 0, 0	650, 210, 1100, 0, 89, 0	415
22	650, 210, 1100, 0, 89, 0	900, 210, 1100, 0, 89, 0	280
23	900, 210, 1100, 0, 89, 0	650, 210, 1100, 0, 89, 0	280
24	650, 210, 1100, 0, 89, 0	650, -90, 1100, 0, 89, 0	335
25	650, -90, 1100, 0, 89, 0	900, -90, 1100, 0, 89, 0	280

Tabela 8.3: Posições iniciais e finais da trajetória (posição e orientação) e o número de pontos para cada segmento (n pto).

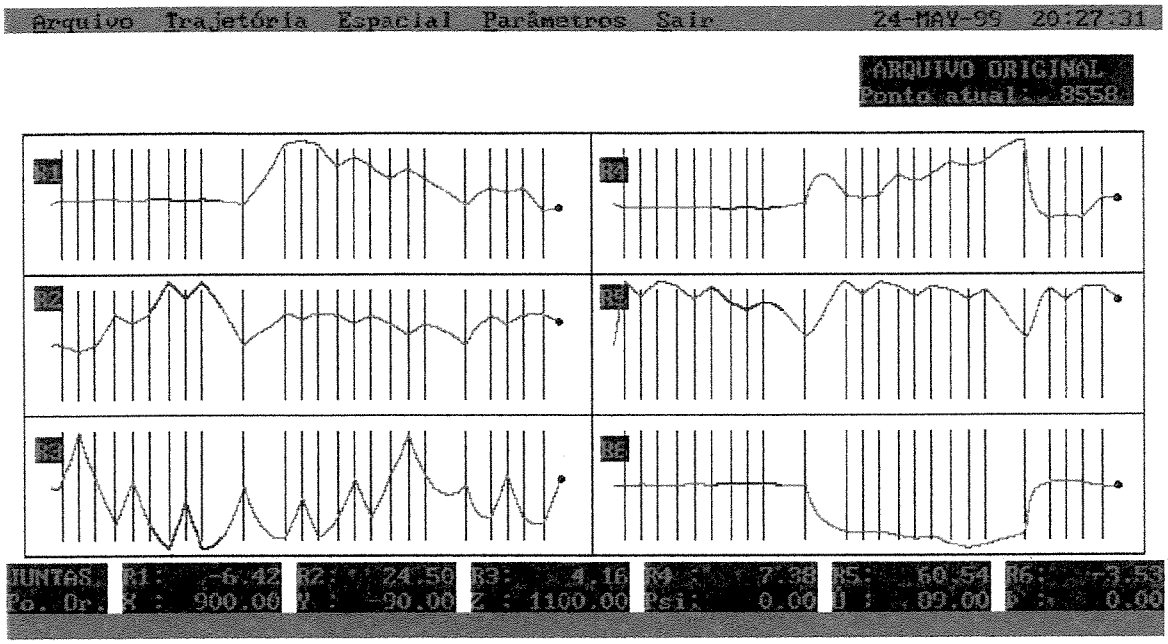


Figura 8.13: Trajetória angular obtida.

A figura 8.14 mostra a tela do módulo Simula com os painéis de atuação e a ferramenta (pino) do robô.

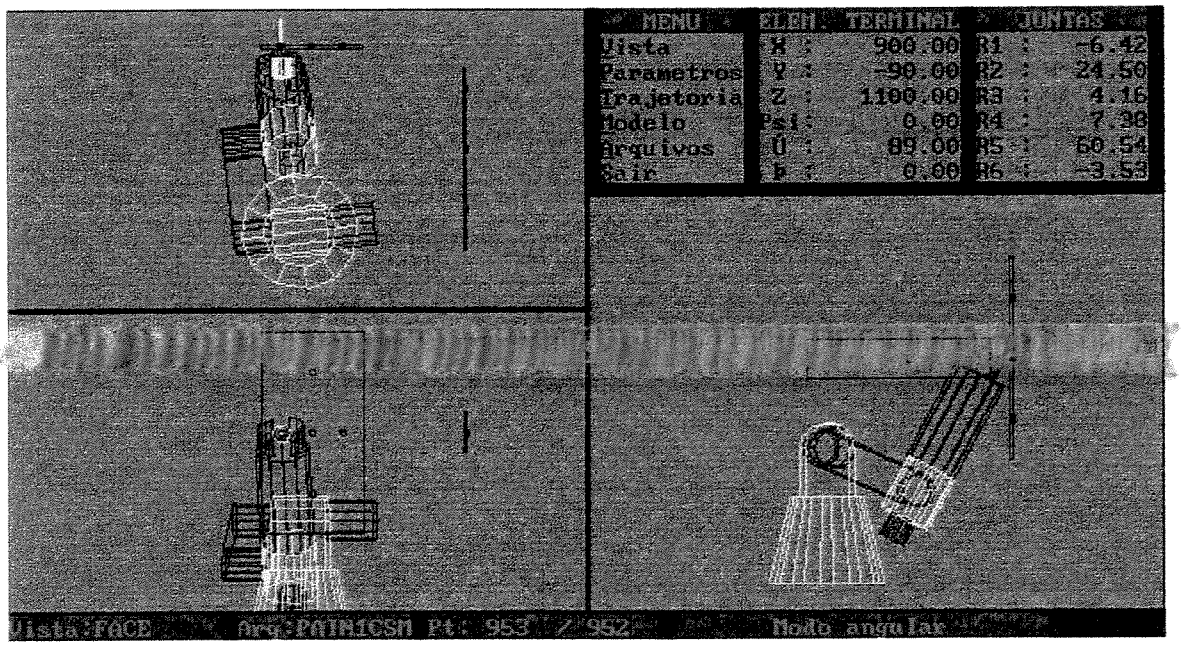


Figura 8.14: Tela do módulo Simula com os painéis e ferramenta de atuação do robô.

8.1.4 Ferramenta com quatro pontas aproxima-se de um painel e realiza um giro simulando o fechamento ou abertura de uma válvula – Simulação 4

A trajetória de referência é constituída de três segmentos. A figura 8.15 mostra a trajetória angular obtida para esta operação. A tabela 8.4 apresenta as posições iniciais e finais de cada segmento desta trajetória (posição em mm e orientação em graus) no formato px, py, pz, psi, teta e phi. Também apresenta o número de pontos (conjunto de ângulos) gerados para cada segmento. O número total de pontos gerados é de 472.

A discretização utilizada em todos os segmentos desta trajetória é linear.

Segmento	Posição (mm) e orientação (graus)		N Pto
	sai de	vai para	
1	500, 0, 1495, 0, 0, 0	800, 100, 1360, 0, 89, 0	335
2	800, 100, 1360, 0, 89, 0	900, 100, 1360, 0, 89, 0	120
3	900, 100, 1360, 0, 89, 0	900, 100, 1360, 0, 89, -45	17

Tabela 8.4: Posições iniciais e finais da trajetória (posição e orientação) e o número de pontos para cada segmento (n pto).

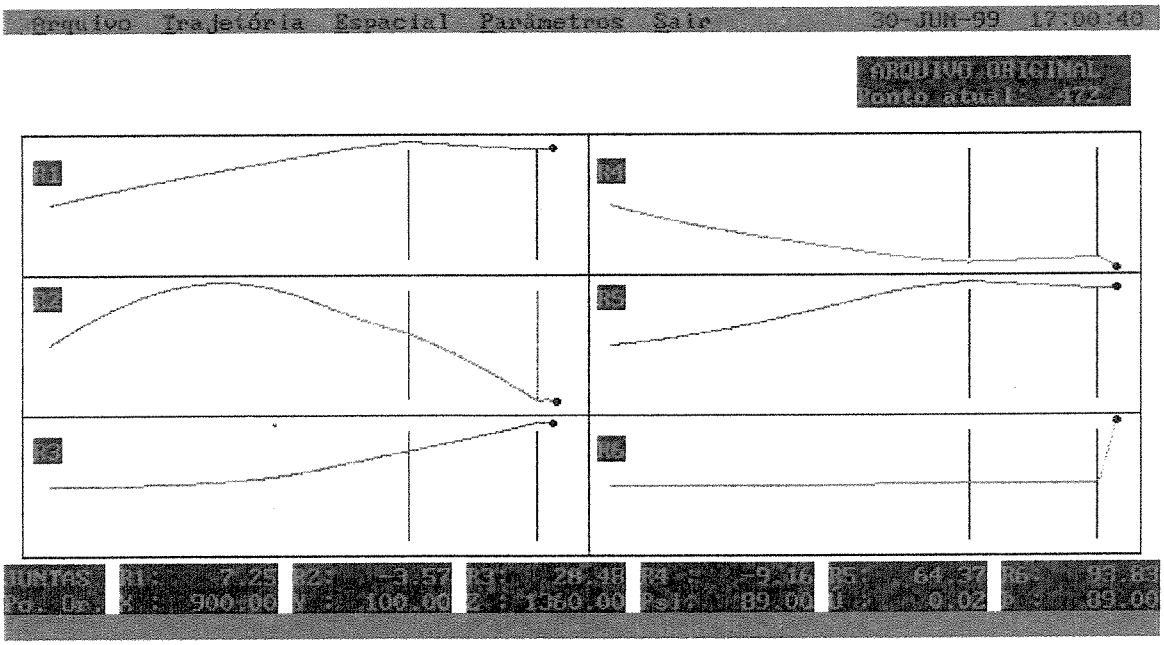


Figura 8.15: Trajetória angular obtida.

A figura 8.16 mostra a tela do módulo Simula com o painel de atuação e a ferramenta com quatro pontas utilizadas pelo do robô.

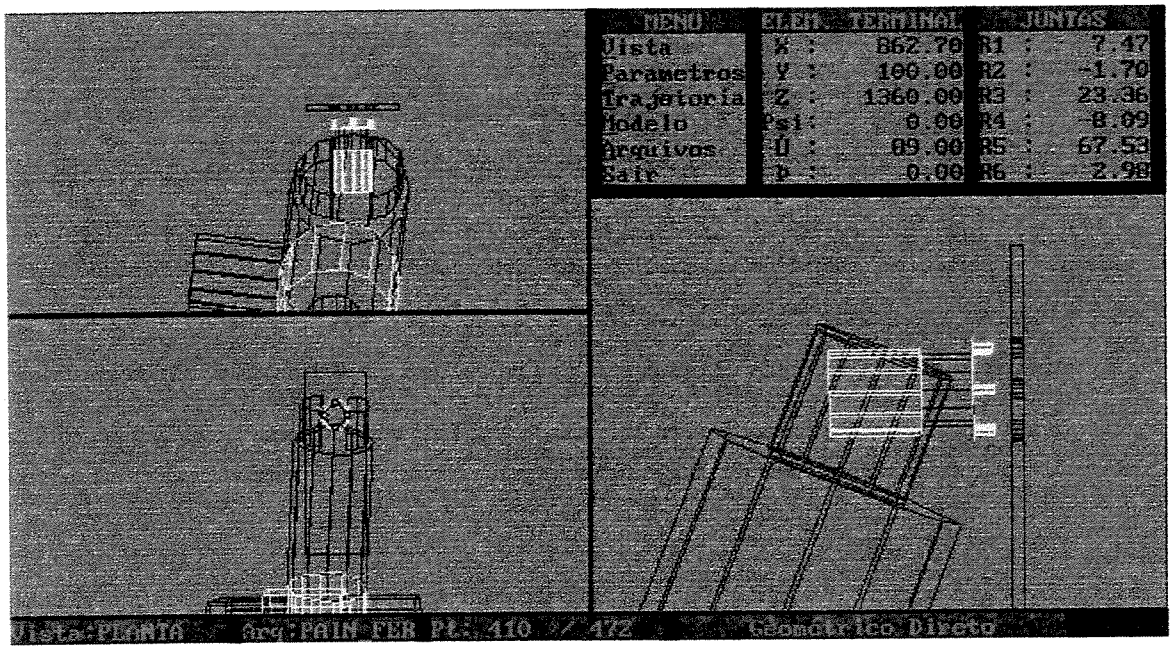


Figura 8.16: Tela do módulo Simula com o painel e ferramenta de quatro pontas.

8.1.5 Trajetória obtida a partir da interpolação e filtragem utilizando os pontos relativos ao giro do elemento terminal da Simulação 4 – Simulação 5

Os pontos de passagem (dezoito) foram obtidos a partir da trajetória gerada pelo modelo cinemático inverso para a tarefa da ferramenta com quatro pontas; estes pontos correspondem ao segmento 3 da figura 8.16 (acima). Foram utilizados os pontos relativos ao giro do elemento terminal; estes pontos são apresentados na figura 8.17. O número total de pontos obtidos após a interpolação e filtragem foi de 1827.

Na figura 8.18 observamos a trajetória obtida a partir da interpolação e filtragem dos pontos de passagem.

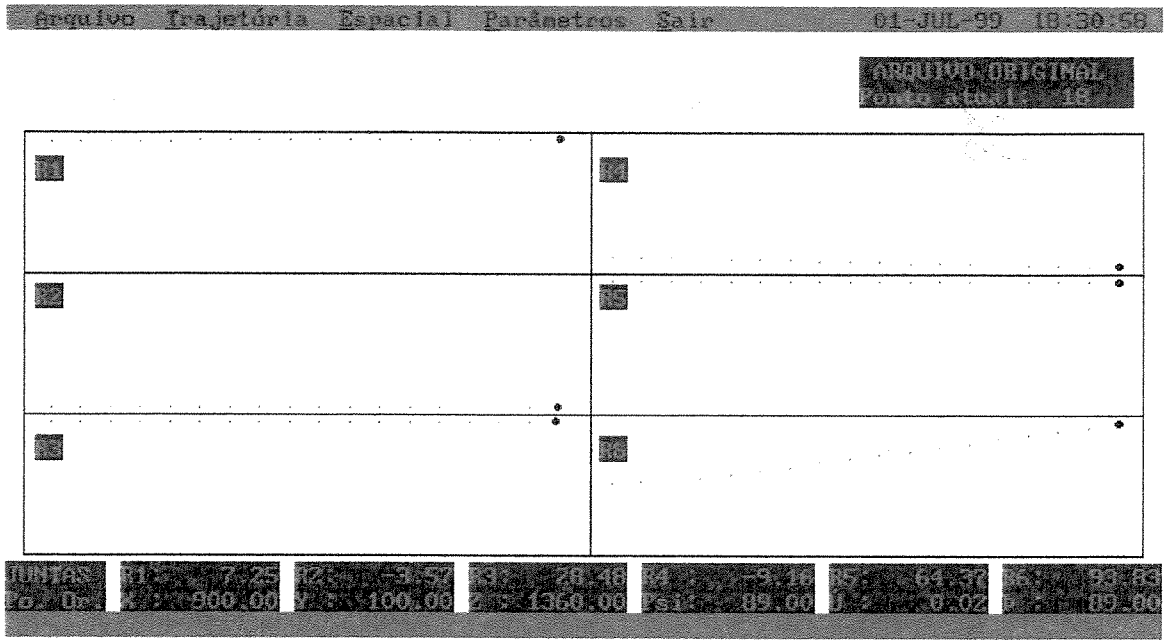


Figura 8.17: Pontos relativos ao giro do elemento terminal.

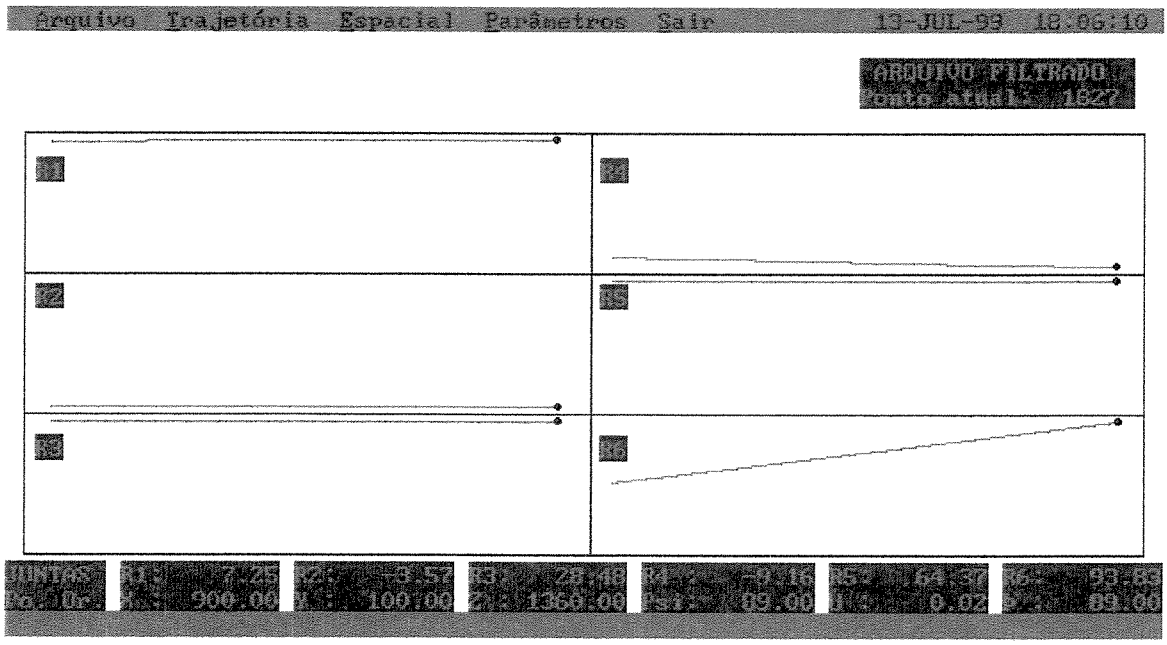


Figura 8.18: Trajetória obtida a partir da interpolação e filtragem.

8.2 Concatenação de arquivos

O módulo de geração de trajetórias permite que seja gerado um arquivo (de trajetórias angulares) a partir de dois ou mais arquivos. Isto é necessário para que arquivos gerados de maneiras diferentes (por interpolação e filtragem, por aprendizagem ou utilizando a cinemática inversa) e que fazem parte de uma mesma tarefa possam ser colocados em um único arquivo fazendo com que a tarefa seja realizada de forma correta.

Esta situação pode ser observada nas simulações 4 e 5 apresentadas anteriormente. Como existe a necessidade da inserção de mais pontos no segmento 3 (tabela 7.4, simulação 4) foi gerado um arquivo (simulação 5) a partir da interpolação e filtragem dos pontos contidos neste segmento gerando deste modo, um novo arquivo com estes pontos. Como este arquivo faz parte da tarefa realizada na simulação 4 os dois arquivos foram concatenados. O arquivo gerado possui um total de 2282 pontos.

8.3 Parâmetros utilizados pelo módulo trajeto

Foram utilizados pelo módulo Trajeto, em todas as simulações, os seguintes parâmetros:

- Erro de posição máximo permitido: 0.5 mm
- Erro de orientação máximo permitido: 0.1°
- Número de divisões de cada segmento: 30 divisões

8.4 Simulação 3 em disquete que segue em anexo

Em anexo a esta tese segue um disquete que contém os módulos Trajeto, Simula, Obstáculo e Sistema. Também contém os arquivos necessários para realizar a Simulação 3 apresentada anteriormente (arquivos de definição e de referência). Estes arquivos se encontram no diretório denominado simulação.

Inicialmente execute o módulo Sistema, para observar as trajetórias angulares apresentadas na figura 8.13, tecla-se Alt+T.(que executara o módulo Trajeto) após isso entra-se no menu *Arquivo* (Alt+A) na opção *Ler Arq.* (L) e na opção *Angular* (A) e digita-se o nome do arquivo e tecla-se Enter, Pain2com (arquivo de referência) que é o arquivo que contém os ângulos das juntas (trajetória) em sua totalidade ou Pain2csm que contém um número menor de pontos. Ao final sai-se do módulo Trajeto (Alt+S) que automaticamente retorna-se para o módulo Sistema.

Para visualizar os painéis de atuação no módulo Obstáculo, tecla-se Alt+O no módulo Sistema, entra-se no menu *Criar* (Alt+C), digita-se o nome do arquivo de definição (Painel5p) e tecla-se Enter.

Para realizar a simulação, tecla-se Alt+S no módulo Sistema, entra-se no menu *Arquivo* (Alt+A) e na opção *Trajetória Atual* (T), digita-se o nome do arquivo e tecla-se Enter, Pain2com que é o arquivo que contém os ângulos das juntas em sua totalidade ou Pain2csm que contém um número menor de pontos, para o robô realizar a tarefa especificada na Simulação 3. Para visualizar o robô realizando a tarefa, entra-se no menu *Trajetória* (Alt+T) na opção *Simulação* (S) e na opção *Automática* (A).

8.5 Interpretação dos resultados obtidos

8.5.1 Estudo de colisão

Como mencionado anteriormente, o módulo Trajeto realiza o teste de colisão robô/robô deste modo, como pôde ser observado através da figura 8.13, simulação 3, existe a colisão robô/robô e que não existe colisão robô/robô nas tarefas mostradas nas figuras 8.1, 8.8, 8.15, 8.17 e 8.18 simulações 1, 2, 4 e 5, respectivamente. O trecho da trajetória angular em que existe colisão é indicada em cor vermelha.

A seguir será discutida a colisão detectada na simulação 3 pelo módulo Trajeto. Esta colisão será estudada mais detalhadamente no módulo Simula será apresentado também um

estudo sobre a alteração da variável de segurança (esta variável está prevista nas equações, 6.8 e 6.13, para a determinação da existência ou não de colisão - Capítulo 6).

8.5.1.1 Colisão prevista durante a geração da trajetória pelo módulo Trajeto

Durante a geração da trajetória, pelo módulo Trajeto, foi indicada a existência de colisão entre os segmentos 7 e 10, figura 8.13. Neste módulo o teste de colisão (robô/robô) é realizado automaticamente e o valor da variável de segurança utilizado foi de 0.05.

A figura 8.19 mostra a trajetória do elemento terminal do robô no plano X-Z nos segmentos onde ocorre a colisão.

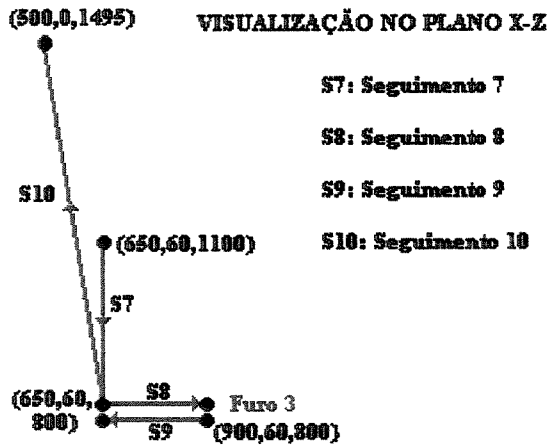


Figura 8.19: Trajetória do elemento terminal do robô no plano X-Z nos segmentos onde ocorre a colisão.

8.5.1.2 Estudo da colisão prevista pelo módulo Trajeto utilizando o módulo Simula

O teste de colisão não é realizado automaticamente pelo módulo Simula deste modo, o teste tem de ser escolhido entre os testes possíveis mencionados anteriormente (nas figuras que serão apresentadas a seguir o teste de colisão realizado foi o de Robô/Robô). A colisão ocorre entre a base e a extremidade inferior de um dos elos do manipulador.

Na figuras 8.20 e 8.21 o valor da variável de segurança utilizado foi de 0.05 e nas figuras 8.22 e 8.23 o valor da variável de segurança foi de 10.0.

A tabela 8.5 mostra o número do ponto para o qual a colisão se inicia e para o qual a colisão termina e os valores angulares das juntas para esses pontos para os valores do fator de segurança utilizados.

Pode-se observar que com o aumento do valor da variável de segurança, a possibilidade de colisão foi detectada com antecedência (observar o zoom nas figuras de 8.20 a 8.23 e tabela 8.5) permitindo desta forma evitar possíveis danos ao equipamento.

	Variável de segurança	Ponto da trajetória	Posição angular das juntas					
			1	2	3	4	5	6
Ponto inicial de colisão	0.05	1681	6.23	38.91	-25.91	-6.41	76.08	1.44
	10.0	1669	6.23	37.57	-25.15	-6.40	76.66	1.37
Ponto final de colisão	0.05	2886	3.79	41.41	-31.37	-4.4	44.74	0.94
	10.0	2900	3.60	40.04	-30.68	-4.24	43.43	0.86

Tabela 8.5: Posição angular das juntas para os diferentes valores da variável de segurança.

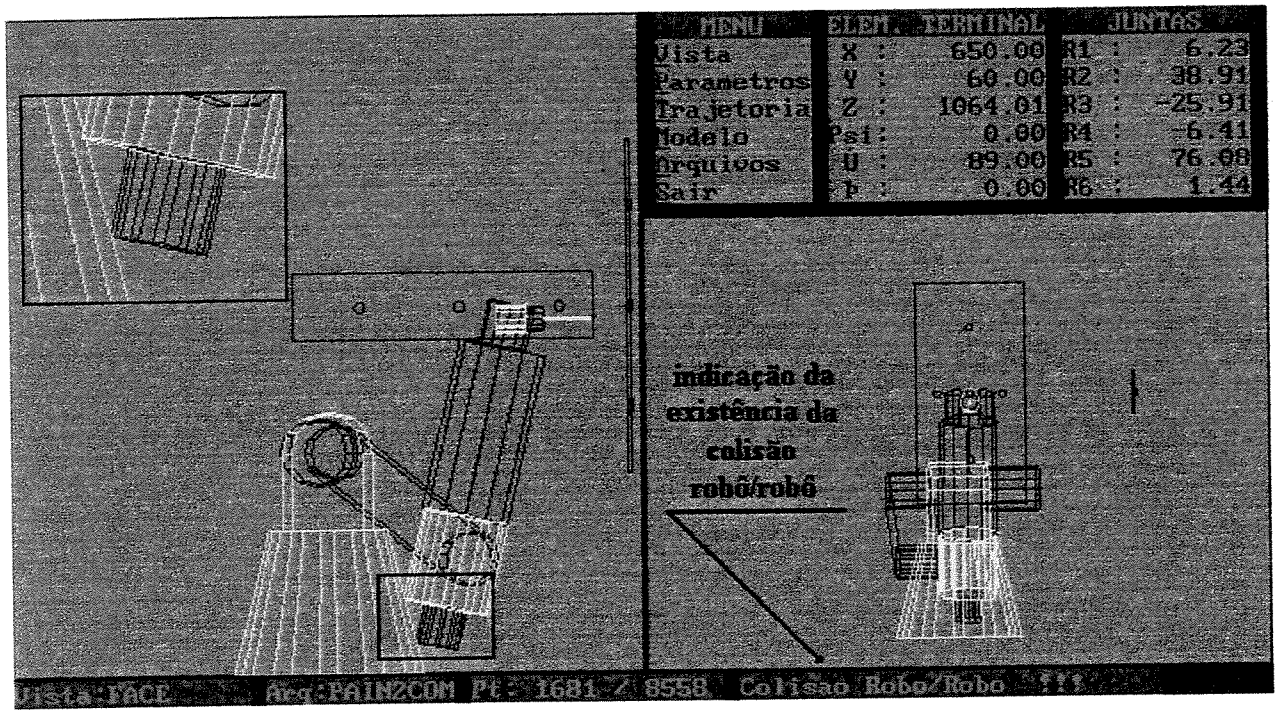


Figura 8.20: Determinação da colisão (ponto inicial) robô/robô, variável de segurança igual a 0.05.

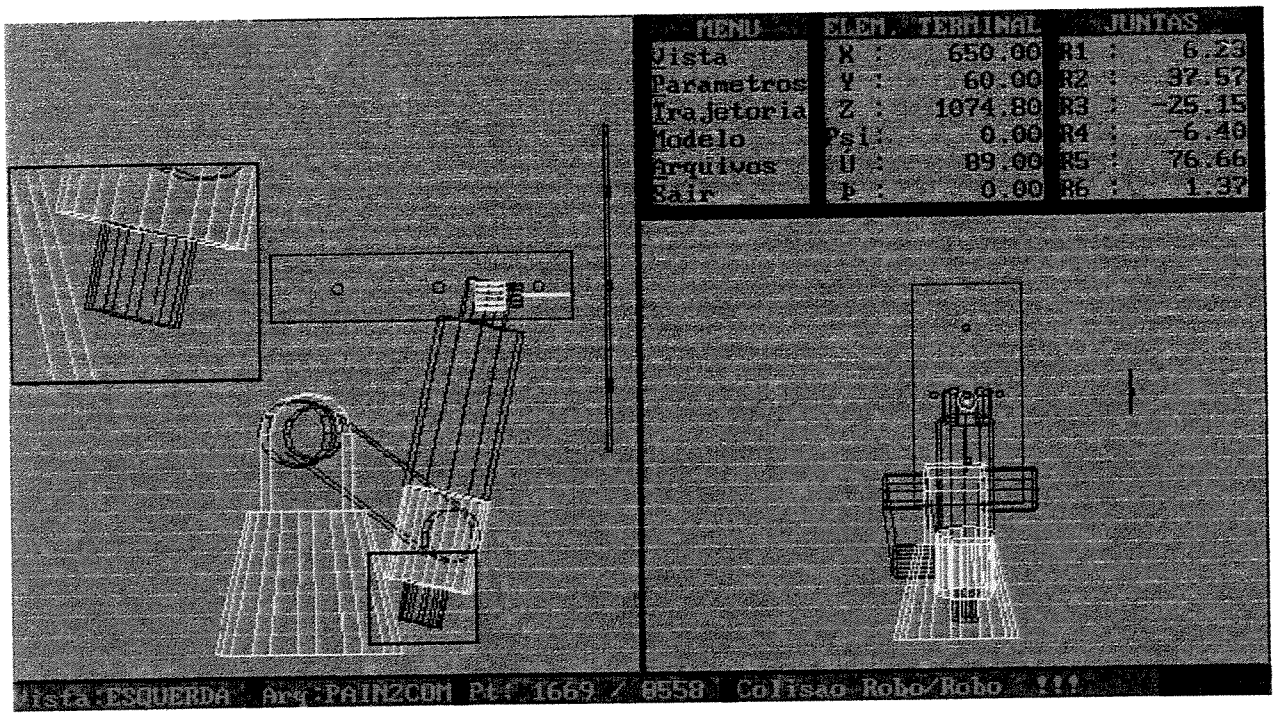


Figura 8.21: Determinação da colisão (ponto inicial) robô/robô, variável de segurança igual a 10.0.

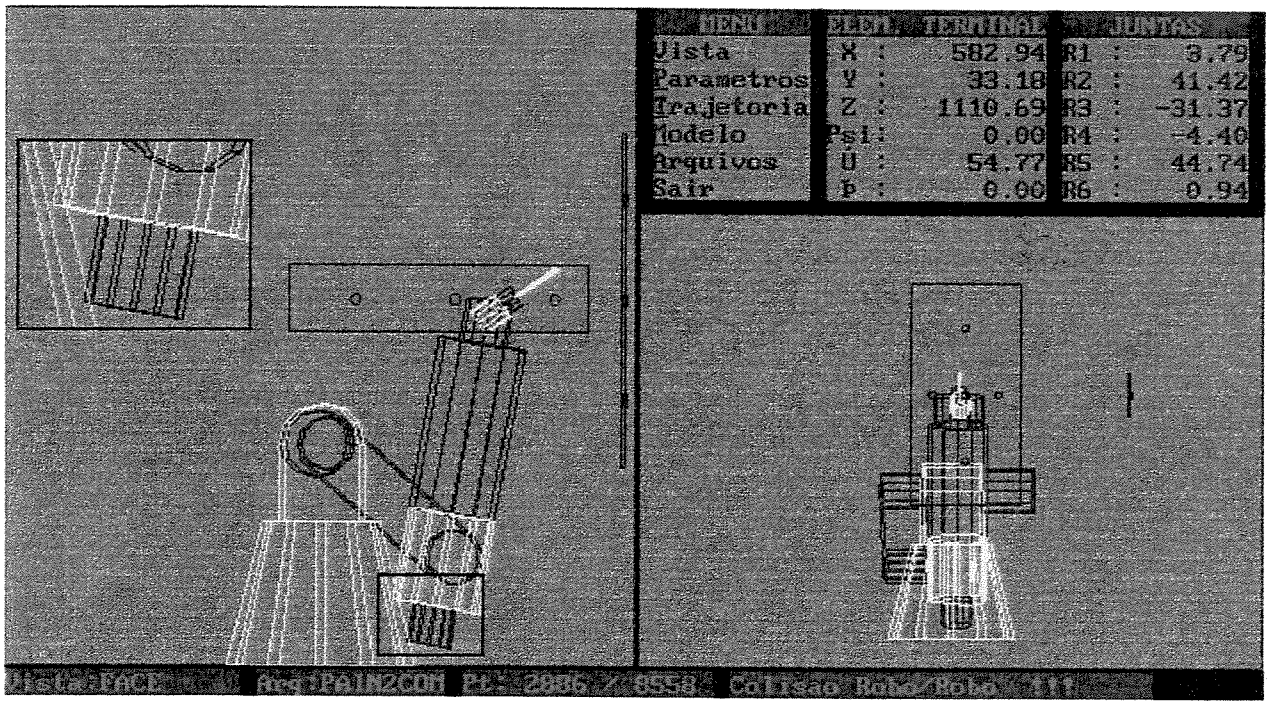


Figura 8.22: Determinação da colisão (ponto final) robô/robô, variável de segurança igual a 0.05.

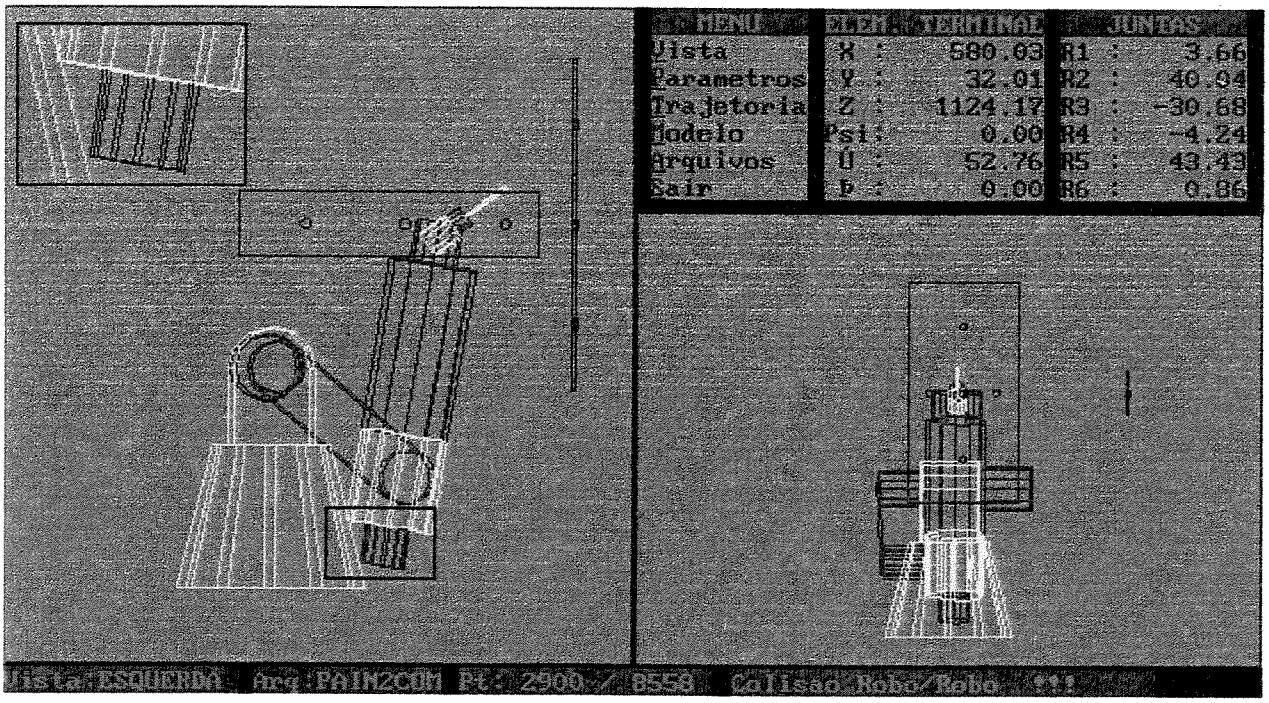


Figura 8.23: Determinação da colisão (ponto final) robô/robô, variável de segurança igual a 10.0.

A posição intermediária entre os pontos iniciais e finais da colisão é mostrada na figura 8.24.

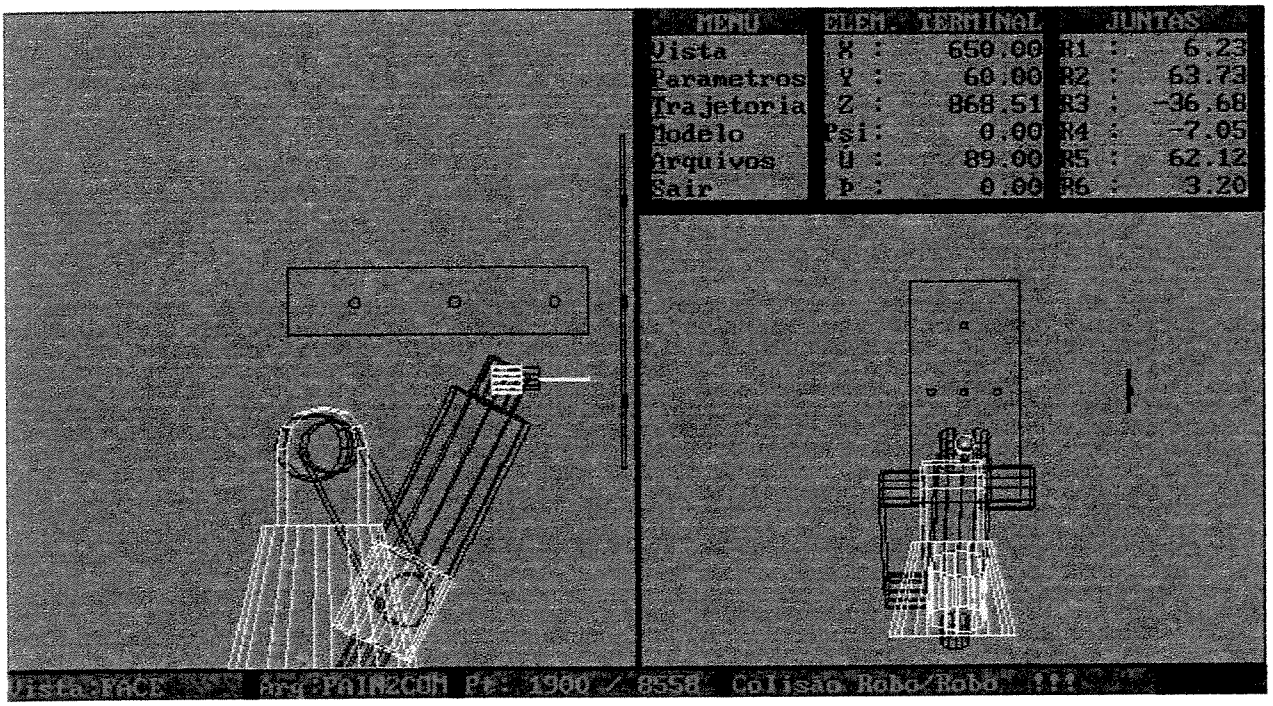


Figura 8.24: Posição intermediária entre o ponto inicial e final da colisão.

8.5.1.3 Conclusão

A colisão observada ocorre porque o painel está muito próximo do robô. Não há outra forma de realizar a trajetória para estes segmentos (sem afastar o painel do robô) a fim de se realizar o trabalho desejado (introduzir o pino no furo 3), figura 8.18.

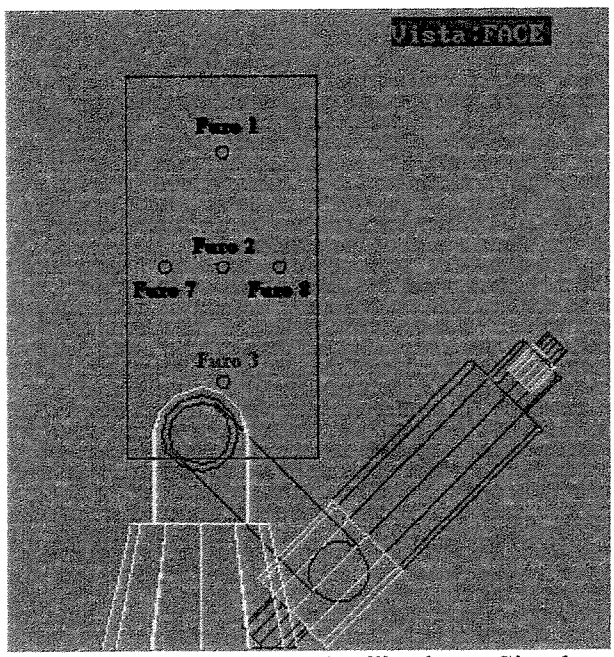


Figura 8.25: Furo 3 do painel utilizado na Simulação 3.

8.5.2 Erro de posição e orientação

Pode ser observado pelas tabelas X.1 a X.4, Anexo X, que a posição e orientação finais, alcançadas apenas em alguns casos, não são iguais as desejada.

Não foram apresentadas todas as simulações e gráficos pois, de uma maneira geral, os resultados obtidos foram semelhantes.

Capítulo 9

Resultados Experimentais

A estrutura final do sistema de supervisão e controle implementado, que contém o software de programação "off-line" e o software de supervisão e controle, é apresentado na figura 9.1.

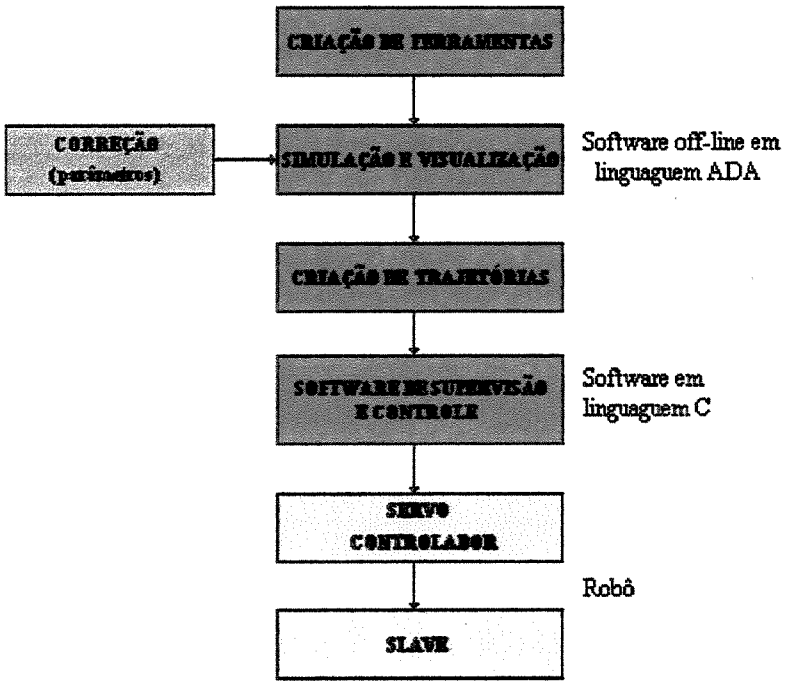


Figura 9.1: Estrutura final do sistema implementado.

Esta estrutura, juntamente com o desenvolvimento de ferramentas dedicadas, permitiu a utilização do robô em ambientes de difícil acesso ao homem.

Neste capítulo serão apresentados a estrutura construída (Octos^{TM}), as ferramentas dedicadas desenvolvidas e os testes em laboratório, visando a validação das tarefas em um típico ambiente submarino a serem realizadas pelo robô Manutec e pelo manipulador Kraft.

9.1 Estrutura mecânica construída

Para a realização dos testes de performance foi construído, na Unicamp, um ambiente submarino típico como maquete em escala real, Mock-up figura 9.2, e uma base móvel sem mecanismo de propulsão isso permitiu a execução de testes a seco na Unicamp e em águas rasas no GKSS (Saramago,93).

Essas maquetes simulam parte de da estrutura projetada pela Petrobrás (OCTOS^{TM}) e foram instaladas no GKSS e na Unicamp para um programa conjunto de testes. A figura 9.3 mostra uma representação do robô Manutec operando sobre a base móvel.

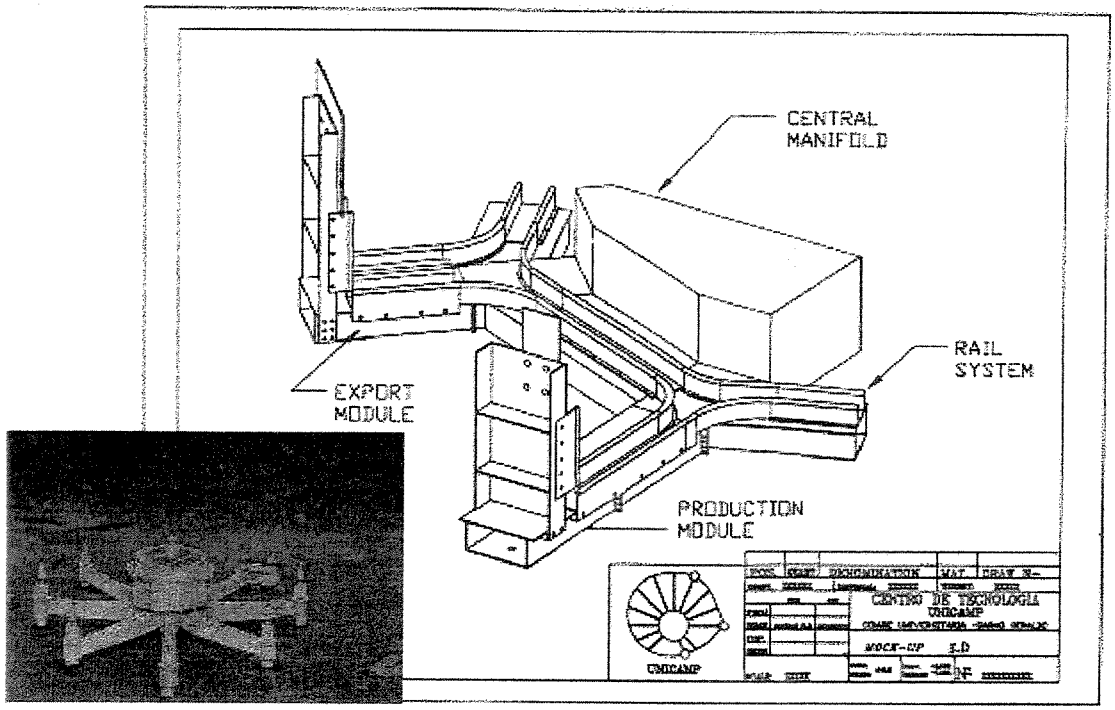
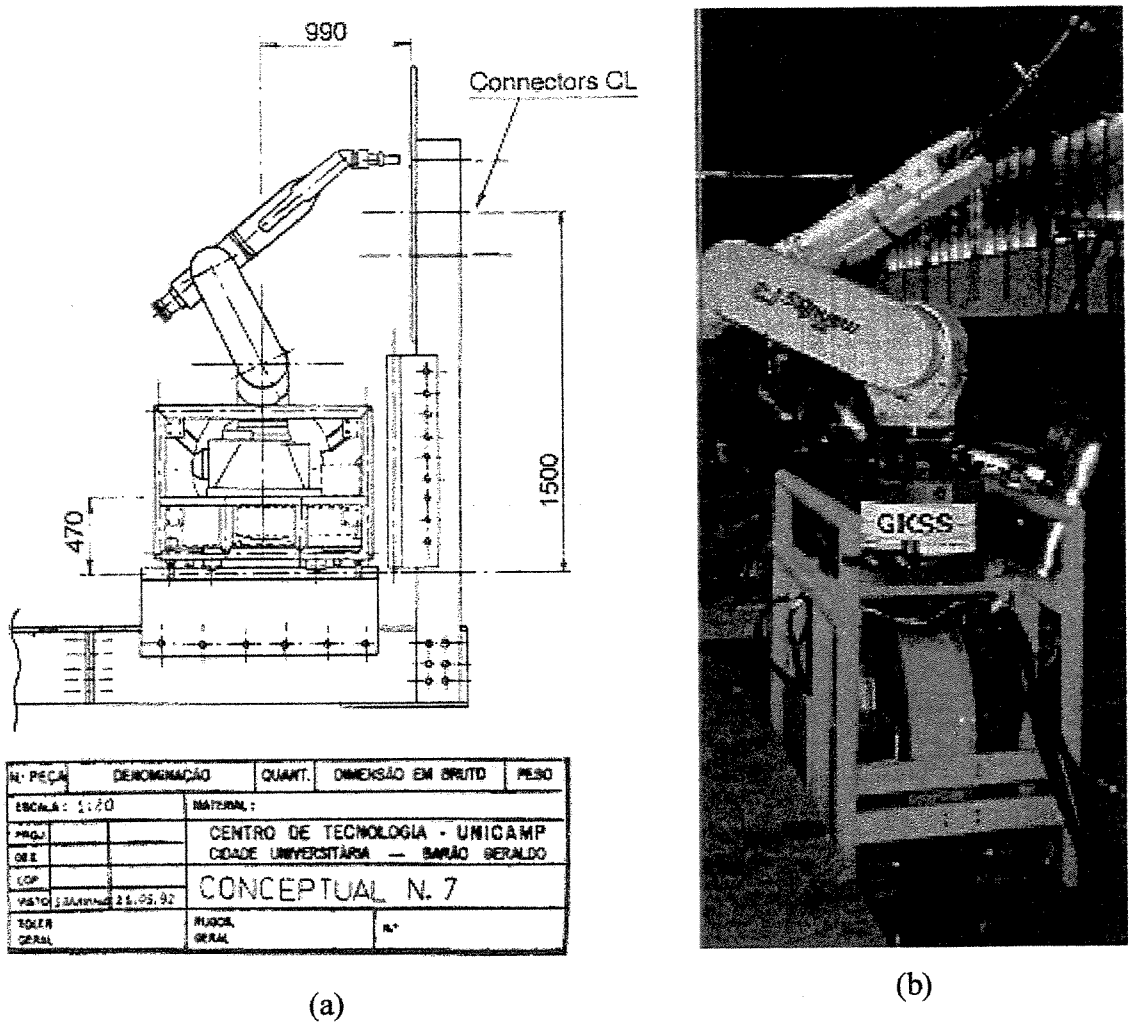


Figura 9.2: Sistema Template Manifold – Mock-up construído na Unicamp. No destaque representação artística do Octos^{TM}.



(a) (b)

Figura 9.3: Base móvel desenvolvida – (a) projeto, (b) implementação - UNICAMP

9.2 Testes realizados com o robô Manutec

A figura 9.4 mostra a estrutura de supervisão e controle final implementada que permitiu a integração do robô, ferramentas dedicadas e base móvel na estrutura do Octos™.

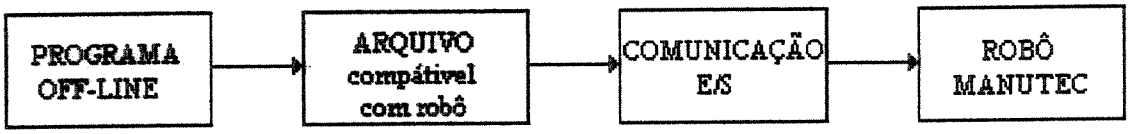
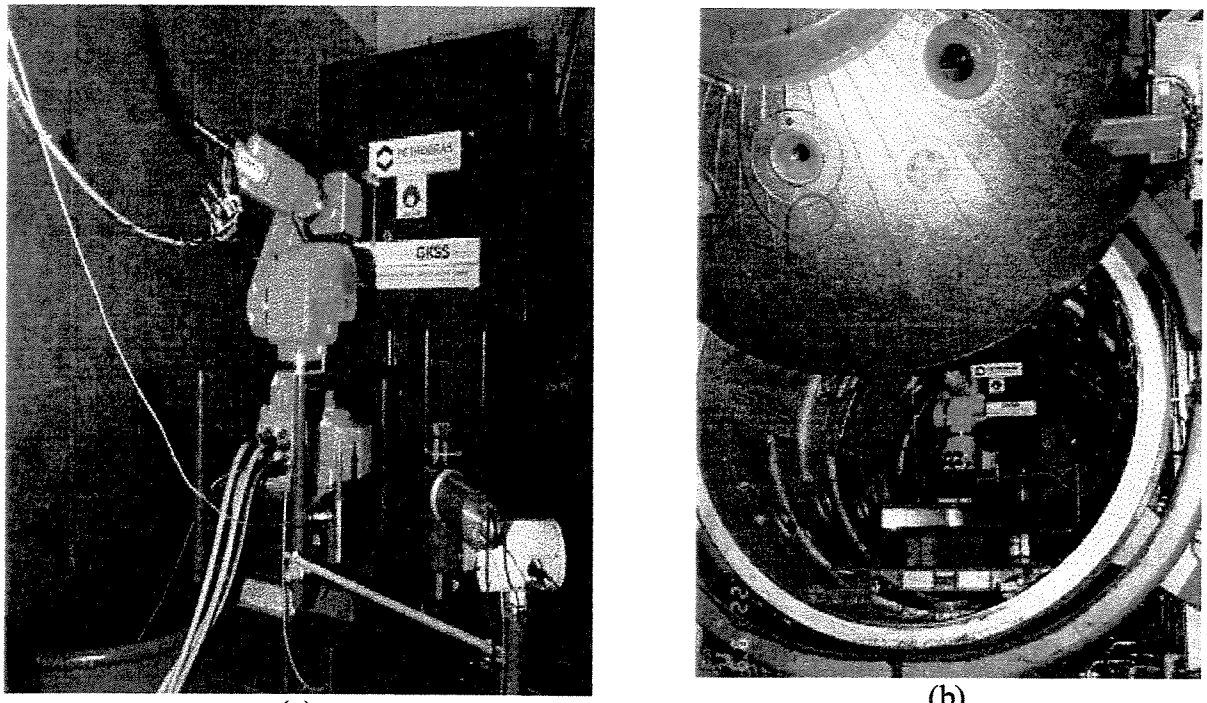


Figura 9.4: Estrutura de controle do robô Manutec.

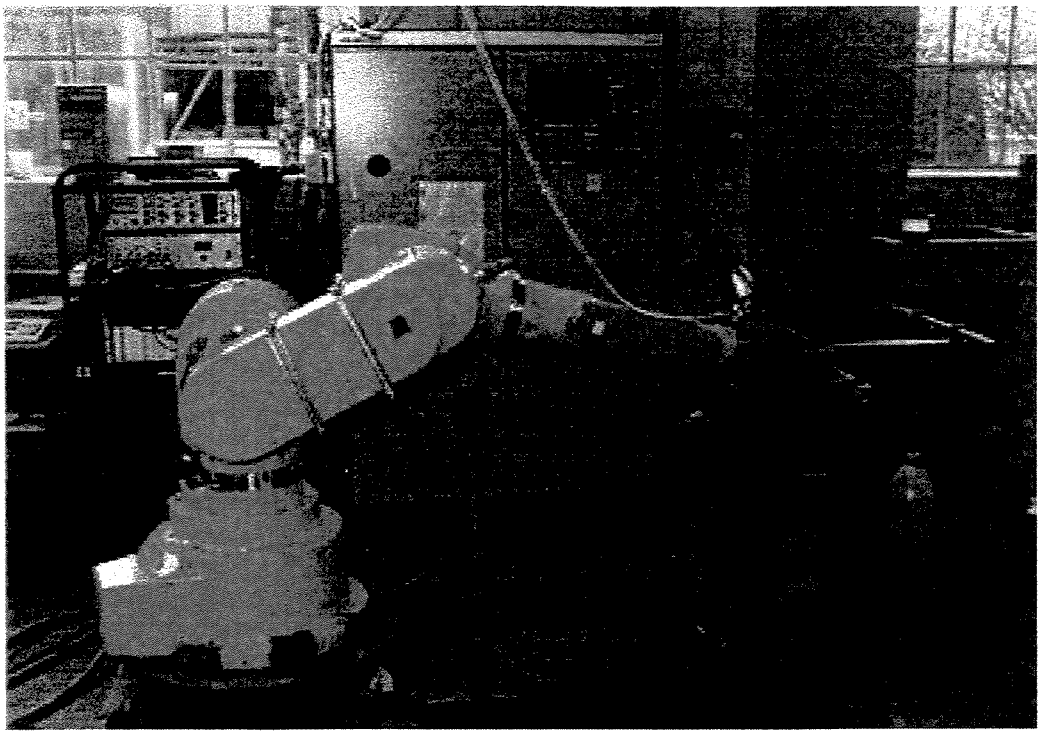
Visando avaliar a execução de tarefas automatizadas pelo robô Manutec, foram realizados testes na qual o robô trabalha sobre a base móvel e desloca-se através de trilhos solidários ao Mock-up até se posicionar em frente ao painel colocado na extremidade, figura 9.5 (a).

Também foram realizados testes em câmaras hiperbáricas, figura 9.5 (b), em ambientes secos, figura 9.6 (a), e em piscinas, figura 9.6 (b).

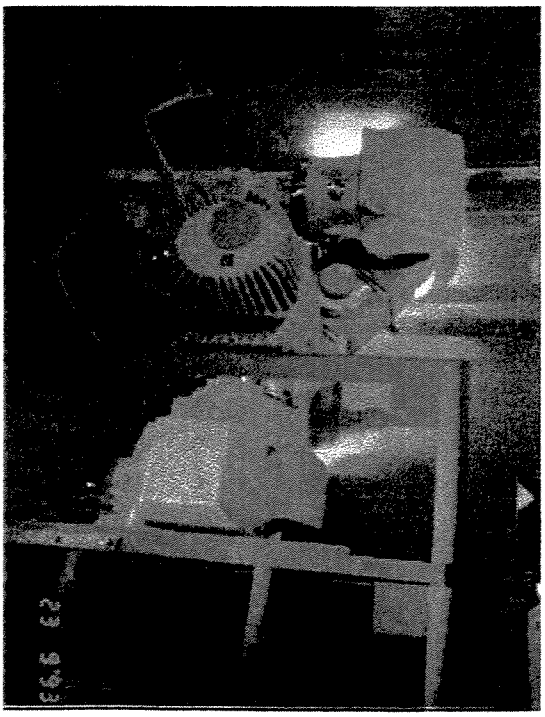


(a) (b)

Figura 9.5: Testes de programação "off-line" – (a) ambiente seco sobre os trilhos, (b) em câmara hiperbárica.



(a)



(b)

Figura 9.6: Testes de programação "off-line" – (a) ambiente seco, (b) em piscina.

9.3 Testes realizados com o manipulador Kraft

Visando avaliar a execução de tarefas automatizadas pelo manipulador Kraft, foram realizados testes utilizando a combinação dos modos de operação existentes ("master-slave", microcomputador e"mouse" espacial) verificando a possibilidade de, a qualquer momento, o operador intervir utilizando um dos modos. A figura 9.7 mostra a estrutura de controle final implementada.

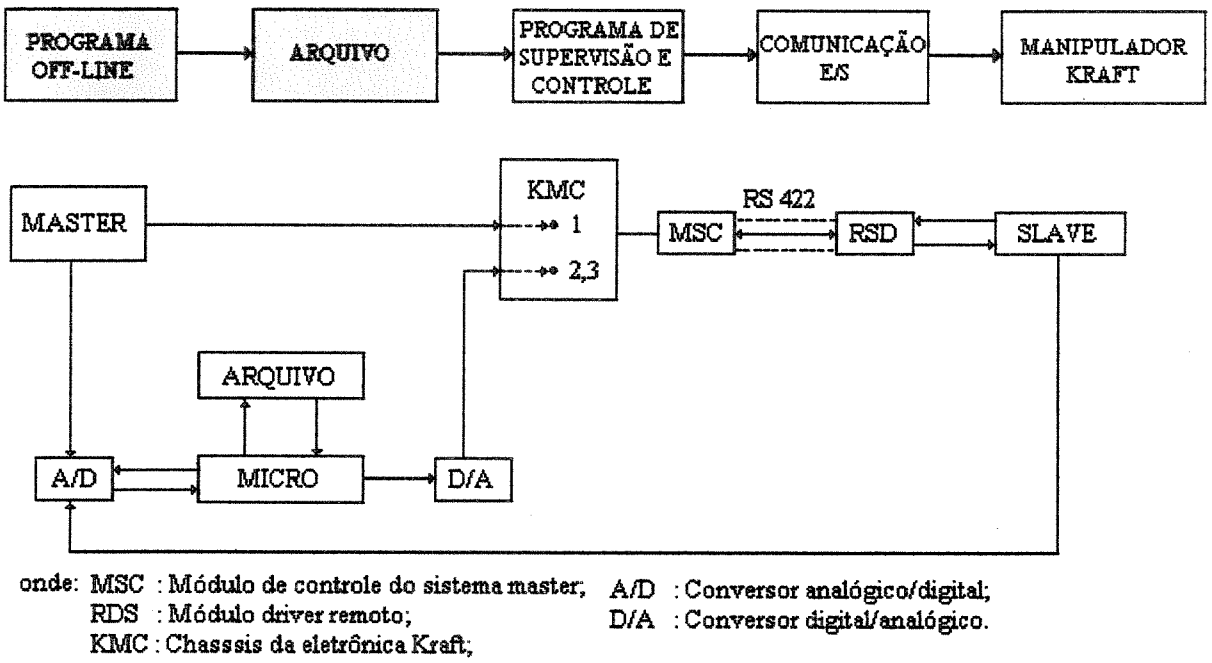


Figura 9.7: Estrutura de controle do manipulador Kraft.

Para validar o sistema desenvolvido de supervisão e controle, foi montada a infra-estrutura de laboratório, na Unicamp, mostrada na figura 9.8, onde aparece o conjunto "master-slave" - controlador Kraft, a interface Kraft desenvolvida, o "mouse" espacial e o painel de testes de conexão de "stab" hidráulico.

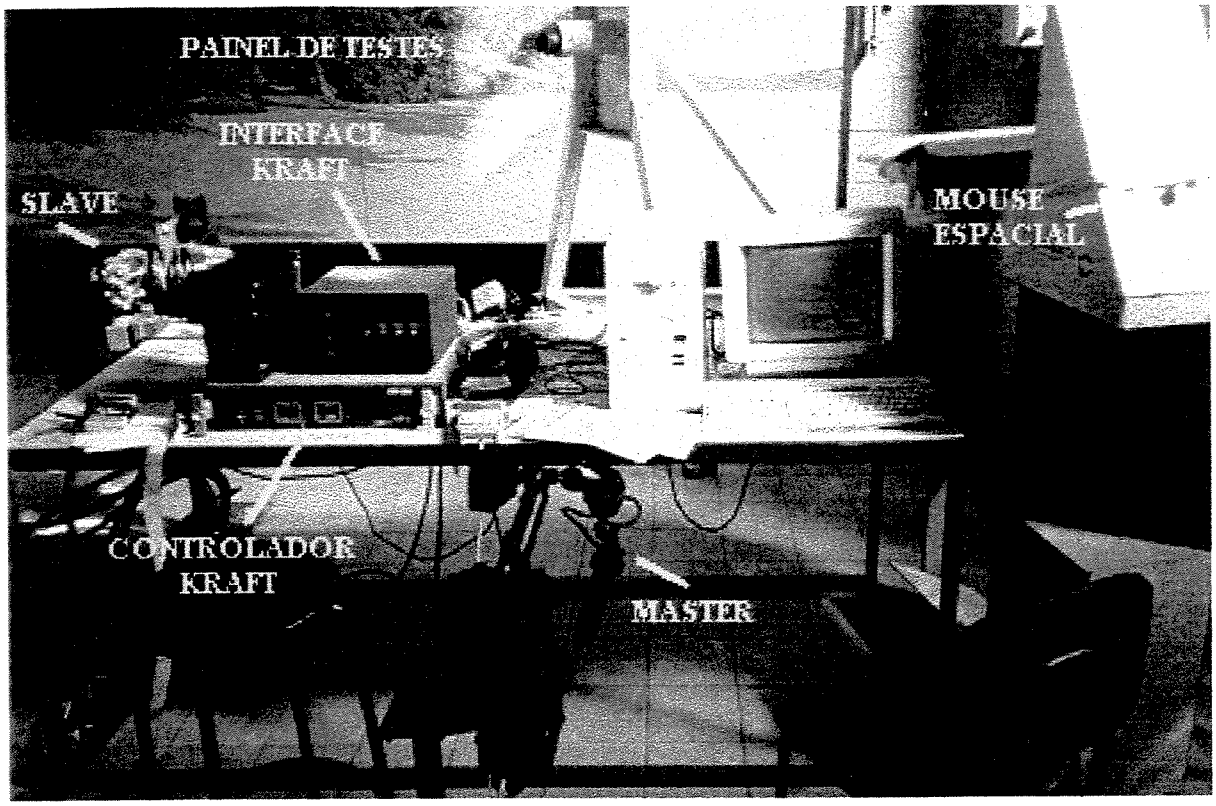


Figura 9.8: Infra-estrutura de laboratório utilizada para a validação do sistema – UNICAMP.

Nos testes realizados em laboratório, verificou-se o funcionamento dos módulos desenvolvidos observando um grau de complexidade crescente das tarefas.

Desde tarefas simples, como pegar objetos de geometria conhecida e em locais de coordenadas definidas, até a simulação de operação de conexão de "stab" hidráulico em painel, figura 9.9, foram testadas considerando programação "on-line" e "off-line".

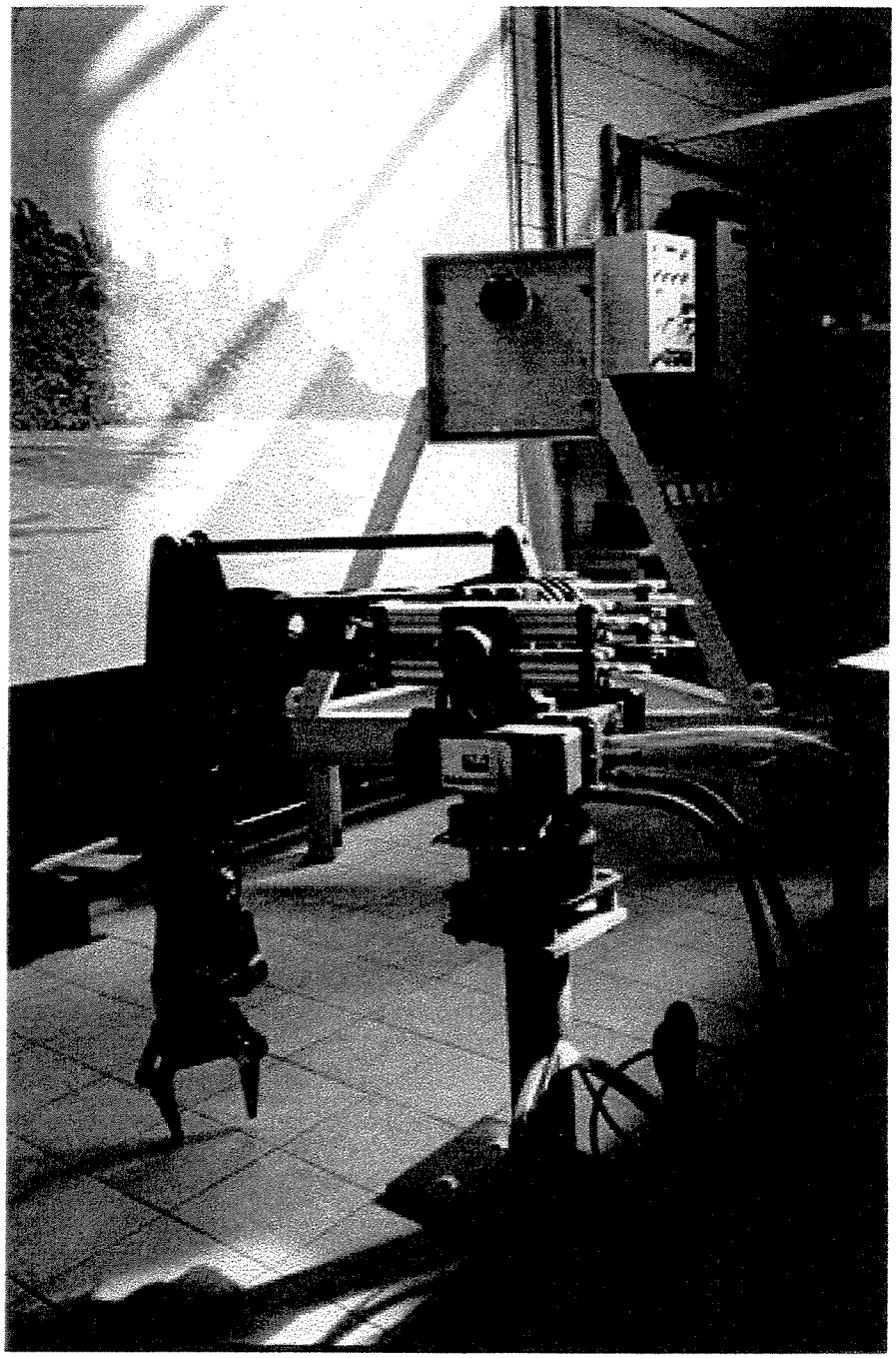


Figura 9.9: Manipulador Kraft com o painel de atuação ao fundo – UNICAMP.

Também foi montada uma infra-estrutura de laboratório semelhante, no CENPES, como pode ser observado na figura 9.10.

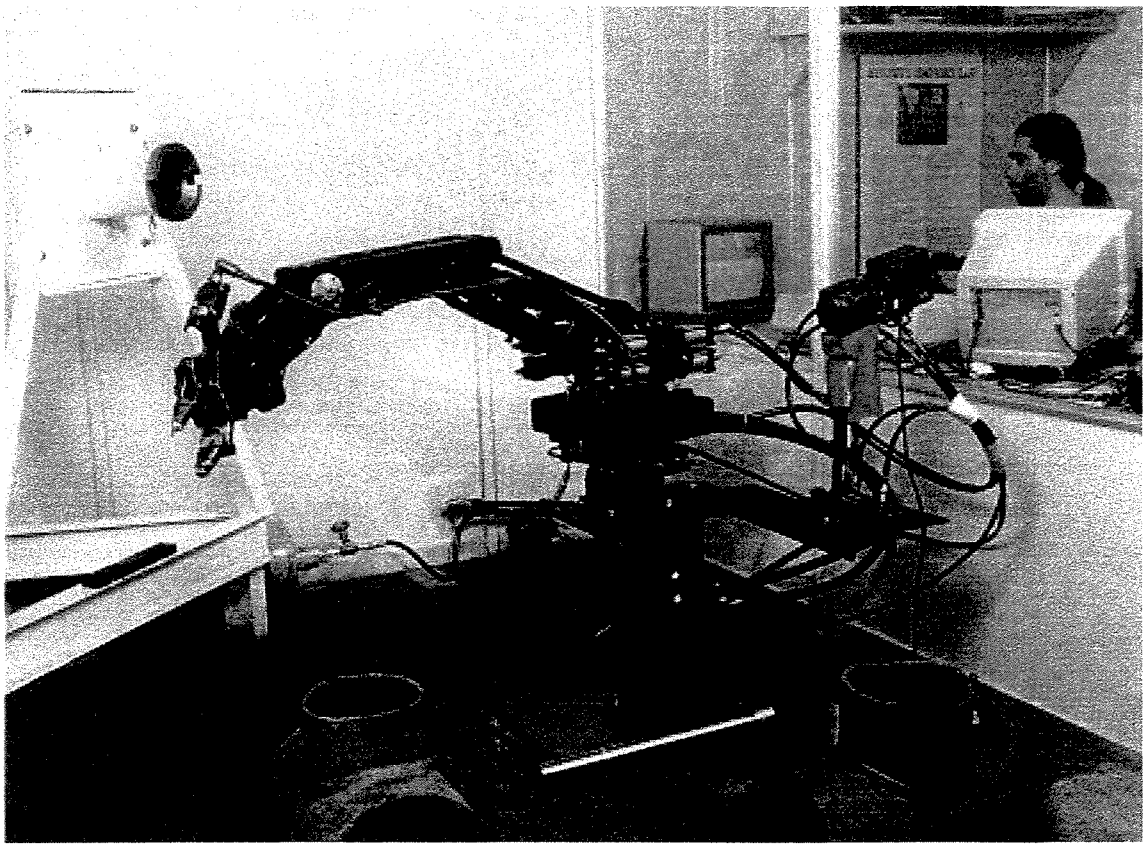
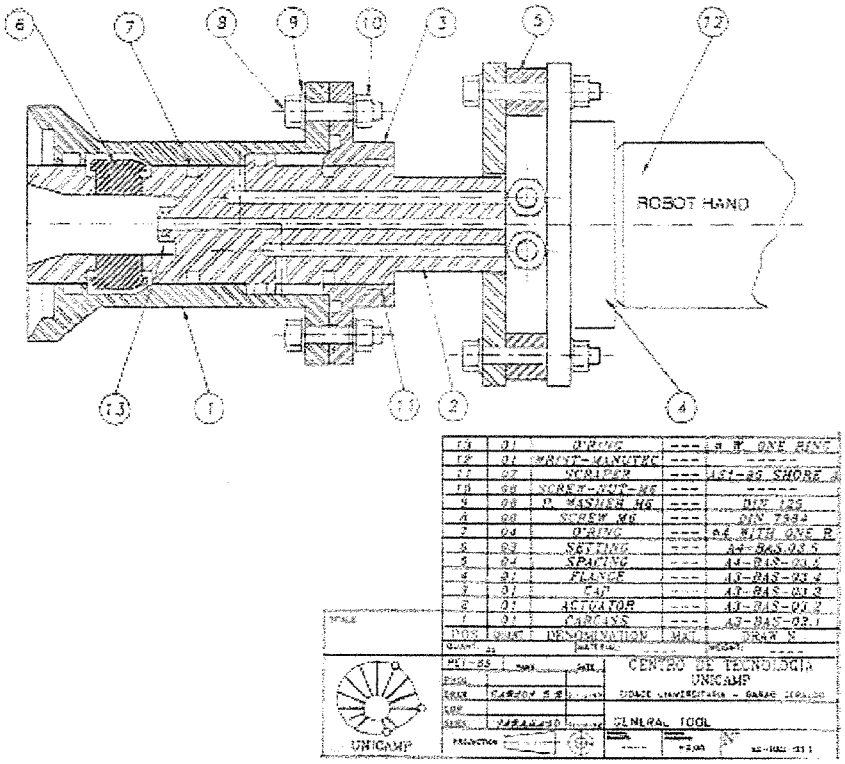


Figura 9.10: Manipulador Submarino Kraft-Grips (6 GL) - CENPES.

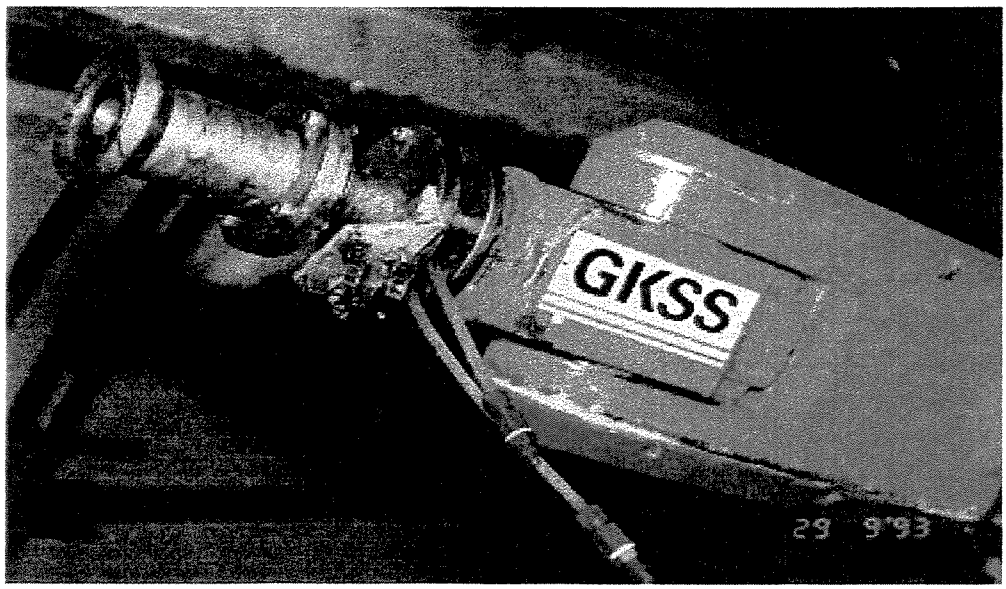
Os resultados obtidos foram positivos, tendo possibilitado alcançar uma melhor compreensão das limitações inerentes à utilização dos manipuladores hidráulicos submarinos, originalmente concebidos para serem operados pelo homem em configuração "master-slave" portanto sem requisitos de acurácia e repetibilidade, para a execução de tarefas automatizadas.

9.4 Ferramentas dedicadas

Alem da construção do ambiente submarino típico apresentado acima, foram projetadas e construídas ferramentas dedicadas que permitiram a validação dos resultados obtidos.



(a)



(b)

Figura 9.11: Ferramenta universal desenvolvida – (a) projeto, (b) implementação de um protótipo de teste (ferramenta pneumática).

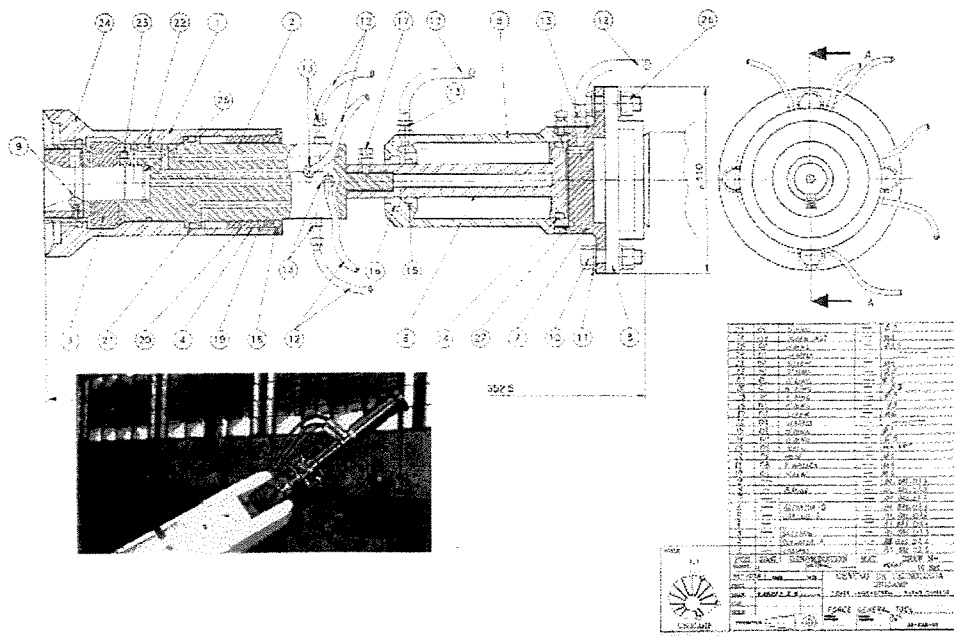
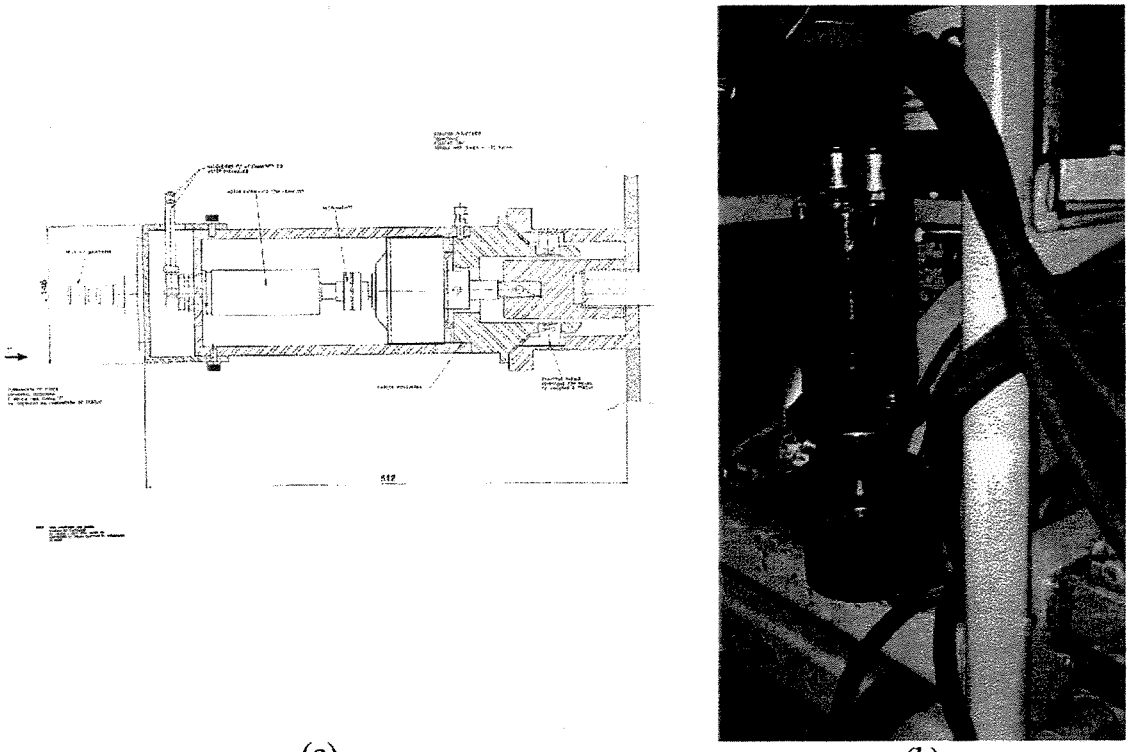


Figura 9.12: Ferramenta hidráulica universal desenvolvida – projeto e implementação final. Os testes foram realizados utilizando componentes reais.



(a) (b)

Figura 9.13: Ferramenta hidráulica de força-torque desenvolvida – (a) projeto, (b) implementação (b) – Unicamp.

Capítulo 10

Conclusões e Perspectivas Futuras

O programa de doutoramento desenvolvido fez parte das atividades desenvolvidas junto ao CENPES-PETROBRÁS, GKSS - Instituto Tecnológicas de Geestacht e UNICAMP, que teve como objetivo o desenvolvimento de um algoritmo de geração de trajetórias, a implementação do programa computacional "off-line" e a implementação de um programa computacional para a Supervisão e Controle de manipuladores robóticos.

Dentro deste contexto foi utilizado o robô industrial Manutec R3 fabricado pela Siemens composto de seis juntas rotacionais acionadas eletricamente e o manipulador submarino Kraft (também com seis juntas rotacionais)

Foi desenvolvido um software de programação "off-line" para o robô Manutec r3 e para o manipulador Kraft. Foi desenvolvido também um pacote de supervisão e controle (pacote computacional CMK) para o manipulador Kraft uma vez que o sistema é aberto e trabalha na configuração "master-slave" sem a necessidade de malha de controle com geração de trajetórias.

10.1 Programa computacional "off-line" para o robô Manutec e manipulador Kraft

A primeira etapa para a implementação do programa "off-line" consistiu na obtenção do modelo geométrico do robô uma vez que ele é necessário tanto para a construção gráfica quanto para a implementação de um algoritmo de geração de trajetórias no espaço cartesiano.

O modelo geométrico foi obtido de dois modos distintos: através da sistemática de Denavit-Hartenberg (utilizado para o algoritmo de geração de trajetórias) e por vetores locais (utilizado para o módulo de visualização).

A seguir foram desenvolvidos os algoritmos para geração de trajetórias no espaço cartesiano e para a interpolação e filtragem de trajetórias e realizadas simulações iniciais. Em todas as simulações utilizando o algoritmo de geração de trajetórias as posições e orientações finais desejadas foram alcançadas e os erros obtidos foram mínimos.

Além disso, o algoritmo para a geração de trajetórias pode ser acrescido de outros tipos de discretização formando, deste modo, uma biblioteca para a geração de trajetórias predefinidas. Neste trabalho foram implementados apenas dois tipos de discretizações (linear e em semicírculo).

As simulações, utilizando o algoritmo de interpolação e filtragem de trajetórias, mostraram que o algoritmo implementado é eficiente.

A partir da obtenção da trajetória, torna-se possível a realização de testes de colisão em geral isto é robô/robô, robô/ambiente de trabalho e robô/ferramenta.

O método proposto para os testes de colisão mostrou-se eficiente e rápida podendo ser utilizada em tempo real. Ela foi utilizada no módulo de geração de trajetórias, Trajeto, para realizar o teste robô/robô e no módulo Simula no qual todos os testes podem ser realizados.

A realização do teste de colisão robô/robô durante a geração de uma trajetória permite que se observe se existira ou não colisão entre os elos do robô e no caso positivo pode-se realizar uma alteração da trajetória inicial para evitar a colisão.

O programa computacional "off-line" que é composto pelos módulos Trajeto (para a geração de trajetórias), Obstáculo (um para a construção de obstáculos e ferramentas) e Simula (para a visualização gráfica do robô, do meio ambiente de trabalho e testes de colisões em geral) cumpre os objetivos inicialmente desejados, isto é:

- programação fora do ambiente industrial;
- redução de tempo de programação;
- otimização do processo a ser automatizado: simplificação na programação de tarefas com alto grau de complexidade e segurança de realização;
- capacidade de integração de outros dispositivos automatizados.

O módulo Trajeto permite que uma trajetória seja composta de um ou de vários segmentos. Isto torna possível a obtenção de trajetórias com um alto grau de complexidade (cada segmento pode ser discretizado de uma forma diferente obtendo, desta forma, segmentos com perfis diferentes) e, como mencionado, permite em um primeiro estágio realizar o teste de colisão robô/robô.

O módulo Obstáculo permite a construção do ambiente de trabalho em geral (obstáculos, ferramentas, painéis) com um alto grau de detalhes utilizando apenas os elementos primitivos básicos. A elaboração dos modelos geométricos para a geração do ambiente não apresenta dificuldades.

O módulo Simula, como pode ser observado, permite que o ambiente de trabalho (gerado no módulo Obstáculo) e a trajetória (módulo Trajeto) sejam simulados em conjunto. Isto permite que a realização de uma tarefa, pelo robô, seja simulada antes o que leva a alcançar os objetivos anteriormente citados.

A utilização das teclas rápidas permite um rápido acesso aos recursos dos módulos, principalmente do módulo Simula.

10.2 Sistema de supervisão e controle CMK para o manipulador Kraft

O pacote computacional CMK viabilizou o monitoramento e controle do manipulador diretamente via micro-computador, através do teclado ou do "mouse" espacial. Este programa permite que os seguintes modos de operação do sistema, sejam utilizados:

- modo "master-slave": sistema original;
- modo microcomputador: a partir da utilização de teclas dedicadas torna-se possível além do controle direto de cada junta do manipulador (sem a utilização do "master"), o controle direto do elemento terminal ou ferramenta (utilização do modelo cinemático);
- modo "mouse" espacial: a partir de um dispositivo externo, ergonométrico do ponto de vista do operador, possamos controlar diretamente a extremidade da ferramenta.

A partir desses modos de operação foi possível, a realização de tarefas automatizadas por dispositivos robóticos, flexibilizando a utilização destes. Este trabalho proporcionou a implementação de métodos para a automação de tarefas submarinas, que poderão ser utilizadas em outras aplicações.

O desenvolvimento de um aplicativo computacional para programação "off-line" de baixo custo computacional (652 KB) e aberto, rodando em um PC, permitiu a utilização de multi-usuários, diferenciado em relação a programas industriais que necessitam de grande esforço computacional e conhecimento especializado para o uso do programa.

Em relação a aplicativos de simulações industriais o programa desenvolvido, nesta tese, possui as seguintes características:

- É um programa aberto o que permite sua alteração e o conhecimento de sua estrutura;

- Permite que métodos específicos para a obtenção do modelo geométrico (existentes ou em desenvolvimento) do dispositivo robótico sejam utilizados;

- Não necessita de grande esforço computacional, pode ser rodado em ambiente DOS;

- Permite que diversas formas de trajetórias sejam geradas tais como, as implementas, semicírculo e linear, e outras que podem ser implementadas a partir da definição de novas funções no programa;

- Permite que outros programas sejam utilizados em conjunto mesmo que estes programas sejam desenvolvidos em outras linguagens;

- Simulação do cenário completo (robô, ferramentas e obstáculos) de diversos pontos de vistas, não apenas de pontos pré-definidos.

Os programas implementados permitem que outros dispositivos robóticos ou biomecânicos sejam utilizados alterando-se apenas alguns pacotes implementados.

Referências Bibliográficas

- **Campos**, R. [1993]. "Implementação de um Mecanismo de Calibração e Identificação de Parâmetros para Robôs". Tese de Mestrado, Faculdade de Engenharia Mecânica, Unicamp.

- **Coutinho**, L. A. F. [1993]. "Um Ambiente Integrado de Desenvolvimento de Software Aplicado a Robótica". Tese de Mestrado, Faculdade de Engenharia Mecânica, Unicamp.

- **Craig**, J. J. [1986]. "Introduction to Robotics Mechanics & Control" . Addison - Wesley Publishing Company, Inc.

- **Cruz**, J. M. da [1993]. "Projeto e Desenvolvimento de um Sistema de Geração Automática de Trajetória para Manipuladores". Tese de Mestrado, Faculdade de Engenharia Mecânica, Unicamp.

- **Dorn**, W. S. , McCracken, D. D. [1972]. "Numerical Methods with Fortran IV Cases Studies". John Wiley & Sons, Inc.

- **Fenton**, R. G., et al [1984]. "Optimal Point to Point Motion Control of Robots With Redundant Degrees of Freedom". Computer – Integrated Manufacturing and Robotics. Ming C. L., Miguel R. M.. The Americam Society of Mechanical Engineers.

- **Ferreira**, E. P. [1991]. "Robótica Básica". Versão Preliminar Publicada para a V Escola Brasileiro - Argentina de Informática, Rio de Janeiro.

- **Fu**, K. S., et al [1987]. "Robotics : Control, Sensing, Vision and Intelligence". McGraw - Hill,Inc.

- **Gerald**, C. F. , Wheatley, P. O. [1992]. "Applied Numerical Analysis" . Addison-Wesley Publishing Company.

- **Gorla**, B. , Renaud, M. [1984]. "Modéles des Robots Manipulateurs Application à Leur Commande" . Cepadues-Éditions.

- **Hayati**, S. A. , Roston, G.P. [1986]. "Inverse Kinematic Solution for Near-Simple Robots and its Application to Robot Calibration". Recente Trends in Robotics Modeling, Control and Education, Elsevier Science Publishing Co., Inc, p 41-50.

- **Huang**, C. H. , Klein, C. A. [1983]. "Review of Pseudoinverse Control for Use With Kinematically Redundant Manipulators" . IEEE Transaction on Systems, Man and Cybernetics, vol. smc-13, no2, March/April,p 245-250.

- **Koivo**, A. J. [1989]. "Fundamentals for Control of Robotic Manipulators" . John Wiley & Sons, Inc.

- **Kraft**, [1985]. "Underwater Manipulator System".

- **Kreysizig**, E. [1983]. "Advanced Engineering Mathematics" . John Wiley & Sons, Inc.

- **Lau**, I. F. [1993]. "Elaboração de um Software para a Visualização e Geração Automática de Trajetórias para Robôs Manipuladores em Intervenções Submarinas em Águas Profundas". Iniciação Cientifica, Faculdade de Engenharia Mecânica, Unicamp.

- **Luh**, J. Y. S., Walker, M. W., Paul, R. P. C. [1980]. "Resolved-Acceleration Control of Mechanical Manipulators". IEEE Transactions on Automatic Control, vol. ac-25, no. 3, June.

- **Meneses**, J. [1993]. "Projeto e Desenvolvimento de um Sistema de Geração Automática de trajetória para Manipuladores". Tese de Mestrado, Faculdade de Engenharia Mecânica, Unicamp.

- **Miss**, R. W. , Schutheis, G. F. [1991]. "The Desing of an Algorithm for the Control of Reduntant Kinematics Chains". 5th ICAR, Pisa, Italy, pp. 1783,1991.

- **Nashed**, M. Z. [1976]"Generalized Inverses and Aplications". Academic Press, Inc.

- **Niemann**, H. R.; Rosário, J. M.; Messina, L. C. P. [1995]. "German/Brazilian Cooperation in Scientifc Reserch and Technological Development *Remote Operation - Project UWT-10"*.

- **Nogueira**, R. G. [1995]. "Controle de Posição e Orientação de um Manipulador Através de um Mouse Espacial". Tese de Mestrado, Faculdade de Engenharia Elétrica, Unicamp.

- **Paul**, R. P. , Shimano, B. , Mayer, G. E. [1981]. "Differential Kinematics Control Equations for Simple Manipulators" . IEEE Transaction on Systems, Man and Cybernetics, vol. smc-11, no 6, June,p 456-460.

- **Paul**, R. P. [1981]. "Robot Manipulators : Mathematics, Programming and Control" . The Mit Press.

- **Paul**. R. [1979]. "Manipulator Cartesian Path Control". IEEE Transactions on Systems, Man and Cybernetics, vol. smc-9, no. 11, November.

- **Sá**, C. E. A. de [1996]. "Implementação de Métodos Numéricos para a Resolução do Problema Cinemático Inverso de Robôs com Ênfase em Controle de Posição". Tese de Mestrado, Faculdade de Engenharia Mecânica, Unicamp.

- **Sá**, C. E. A.; Rosário, J. M. [1996]. "Implementation of Numerical Algorithms for the Resolution of Kinematic Inverse Problem of Robots Manipulators". Icone'96 Second International Conference on Non-Linear Dynamics, Chaos, Control and their Applications in Engineering Sciences, São Pedro, Agosto, ISBN 900351, vol. 1, Chapter 3: Control, Robotics, Neural Networks and Optimization, pp 317-321.

- **Sá**, C. E. A.; Rosário, J. M. [1998]. "Development and Implementation of a Modulate Program in Language Ada for the Creation of Tools, Obstacles, Graphic Simulation and Generation of Trajectories for Robots". III Portuguese Conference on Automatic Control, Coimbra - Portugal, September.

- **Sá**, C. E. A.; Rosário, J. M. [1998]. "Algoritmo Numérico para a Solução do Problema Cinemático Inverso de Manipuladores". V CEM – NNE, Fortaleza, Outubro, pp32-39.

- **Sá**, C. E. A.; Oliveira, C.; Rosário, J. M. [1999]. "Numerical Algorithms for the Solution of Kinematic Problem and Dynamical Control of Manipulator". XV COBEM - Congresso Brasileiro de Engenharia Mecânica, Águas de Lindóia, Novembro.

- **Saramago**, M. A. P. [1993]. "Projeto e Desenvolvimento de um Sistema de Calibração e Medida de Precisão para Robôs Industriais". Tese de Mestrado, Faculdade de Engenharia Mecânica, Unicamp.

- **Spong**, M. W. [1989]. "Robot Dynamics and Control" . John Wiley & Sons, Inc.

- **Vandergraft**, J. S. [1983]. "Introduction to Numerical Computations". Academic Press, Inc..

- **Watt**, D. A. [1987]. "Ada : Laguage and Methodology". Prentice - Hall International (UK) Ltd..

- **Snyder**, W. E. [1985]. "Industrial Robots: Computer, Interfacing and Control". Prentice Hall, Inc..

ANEXO I

Definições Gerais

I.1 Sistemática de Denavit-Hartenberg

Para obter a equação que fornece o posicionamento do elemento terminal de um robô em relação ao sistema de coordenadas da base, Denavit-Hartenberg (Ferreira 1991) propuseram a utilização de quatro parâmetros, θ, α, a e d para descrever uma matriz de transformação homogênea A_i^{i-1} entre dois sistemas de coordenadas vinculados a duas juntas sucessivas. A figura I.1 apresenta os quatros parâmetros de Denavit-Hartenberg. Onde:

a_i é a menor distância entre z_i e z_{i-1}

α_i é o ângulo entre z_i e z_{i-1}

d_i é a menor distância entre x_i e x_{i-1}

θ_i é o ângulo entre x_i e x_{i-1}

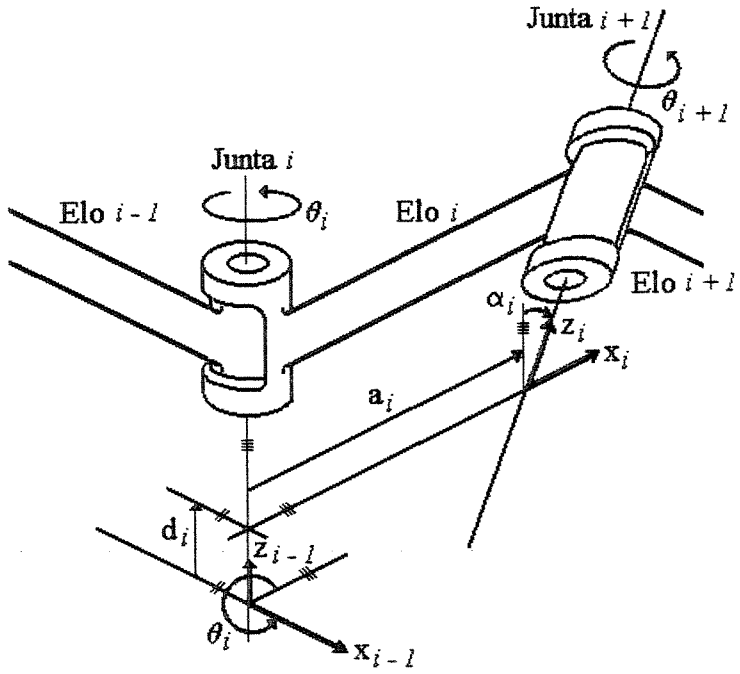


Figura I.1: Parâmetros de Denavit-Hartenberg, θ, α, a e d.

A matriz de passagem A_i^{i-1} é expressa como o produto de quatro transfomações homogêneas, isto é:

$$A_i^{i-1} = \text{rot}(z, \theta)\ \text{trans}(0, 0, d)\ \text{trans}(a, 0, 0)\ \text{rot}(x, \alpha) \tag{I.1}$$

então

$$A_i^{i-1} = \begin{bmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\theta_i\sin\alpha_i & a_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -\cos\theta_i\sin\alpha_i & a_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{I.2}$$

A matriz homogenea T_i^0 que especifica a localização do i-ésimo sistema de coordenadas em relação ao sistema de coordenadas da base é o produto das sucessivas matrizes A_i^{i-1} e é expessa como:

$$T_i^0 = A_1^0\, A_2^1 \ldots A_i^{i-1} \tag{I.3}$$

$$T_i^0 = \begin{bmatrix} \vec{n} & \vec{s} & \vec{a} & \vec{p} \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \text{(I.4)}$$

onde [$\vec{n}$, $\vec{s}$, $\vec{a}$] é a matriz orientação do i-ésimo sistema de coordenadas em relação ao sistema de coordenadas da base e $\underline{p}$ é o vetor posição do i-ésimo sistema de coordenadas em relação ao sistema de coordenadas da base. Os vetores $\vec{n}$, $\vec{s}$ e $\vec{a}$ e $\vec{p}$ são mostrados na figura I.2.

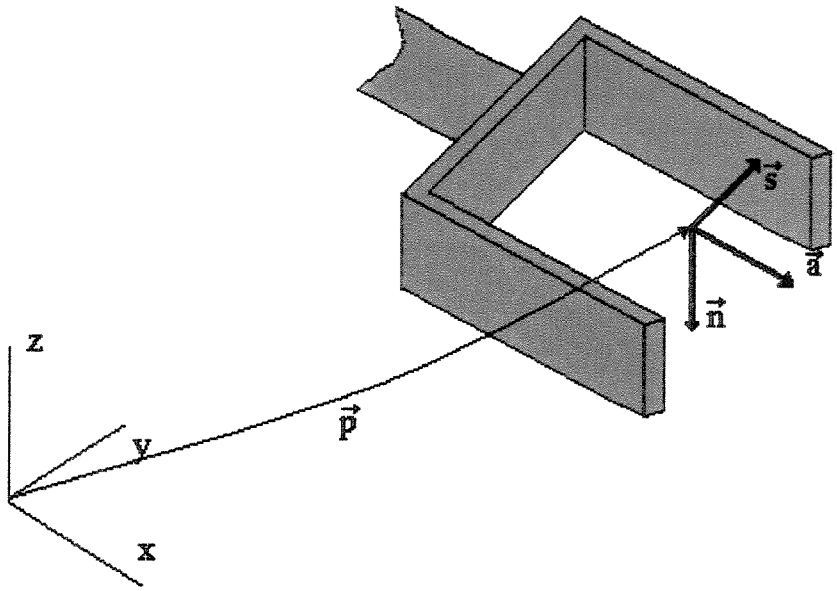


Figura I.2: Vetores $\underline{n}$, $\underline{s}$, $\underline{a}$ e $\underline{p}$.

Neste método são obtidas apenas as coordenadas do elemento terminal do robô. O modelo obtido deste modo pode ser utilizado em programas de geração de trajetórias e de identificação de erros, entre outras, uma vez que são necessárias apenas as coordenadas do elemento terminal.

I.2 Fluxograma para o cálculo dos ângulos RPY a partir da matriz orientação

A figura I.3 apresenta o fluxograma para o cálculo dos ângulos RPY a partir da matriz orientação (cálculo inverso).

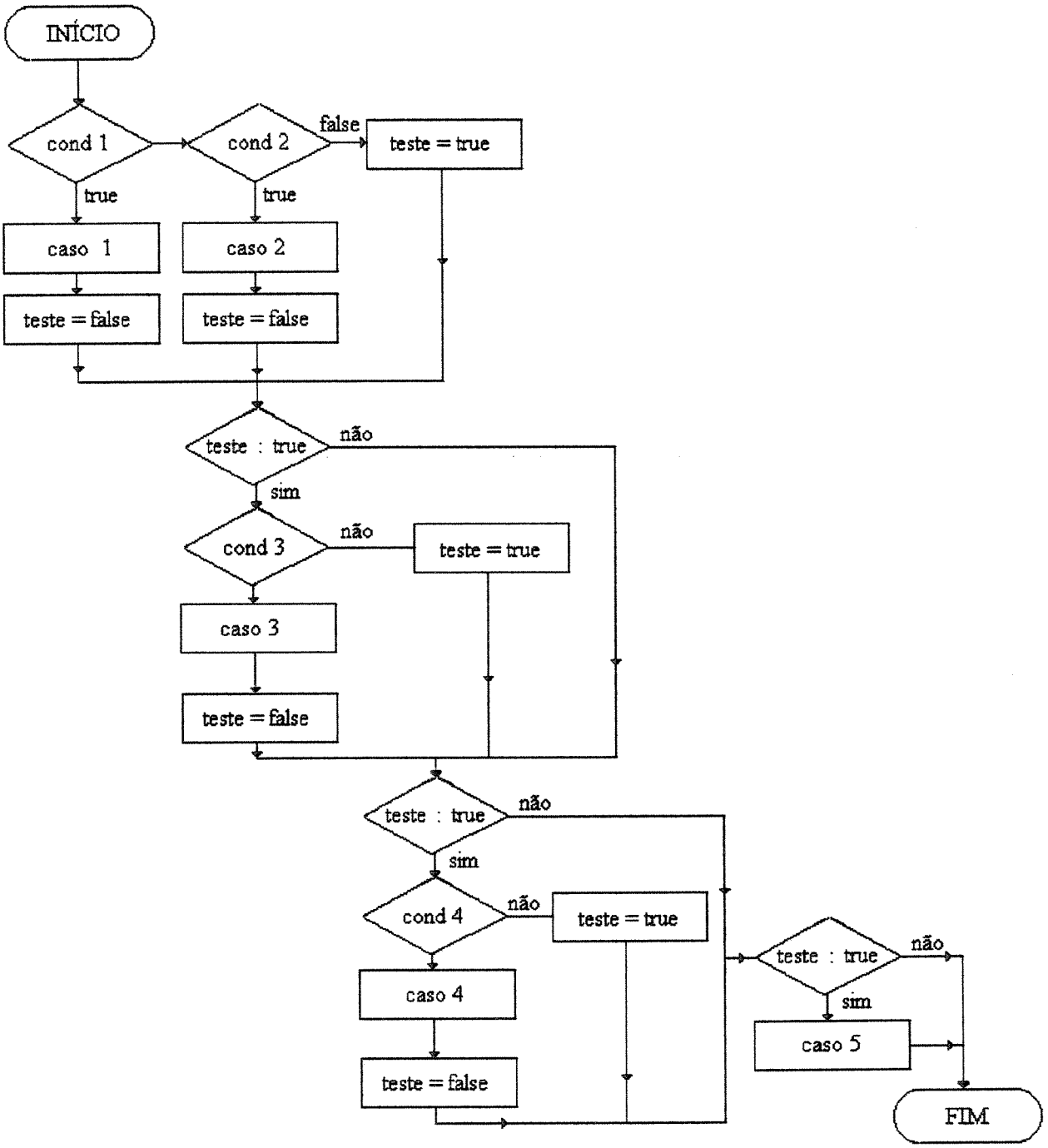


Figura I.3: Fluxograma para o cálculo dos ângulos Rpy a partir da matriz orientação.

Onde os casos referem-se aos citados no Capítulo 3, tópico 3.4.2 e cond 1 (condição 1), cond 2, cond 3 e cond 4 são dados por:

Cond 1: $m_{11} = 0$ e $m_{32} = 0$ e abs(m_{31}) =1;

Cond 2: $m_{11} = 0$ e abs(m_{31}) < 1 $m_{13} = 1$ e $m_{12} = 0$;

Cond 3: $m_{11} = 0$ e abs(m_{31}) < 1 $m_{13} = 1$ e $m_{13} = -1$ e $m_{12} = 0$;

Cond 4: $m_{11} = 0$ e abs(m_{31}) < 1.

I.3 Fluxograma para a criação do ambiente de trabalho – módulo Obstáculo

Na figura a seguir é apresentado o fluxograma para a criação do ambiente de trabalho do robô.

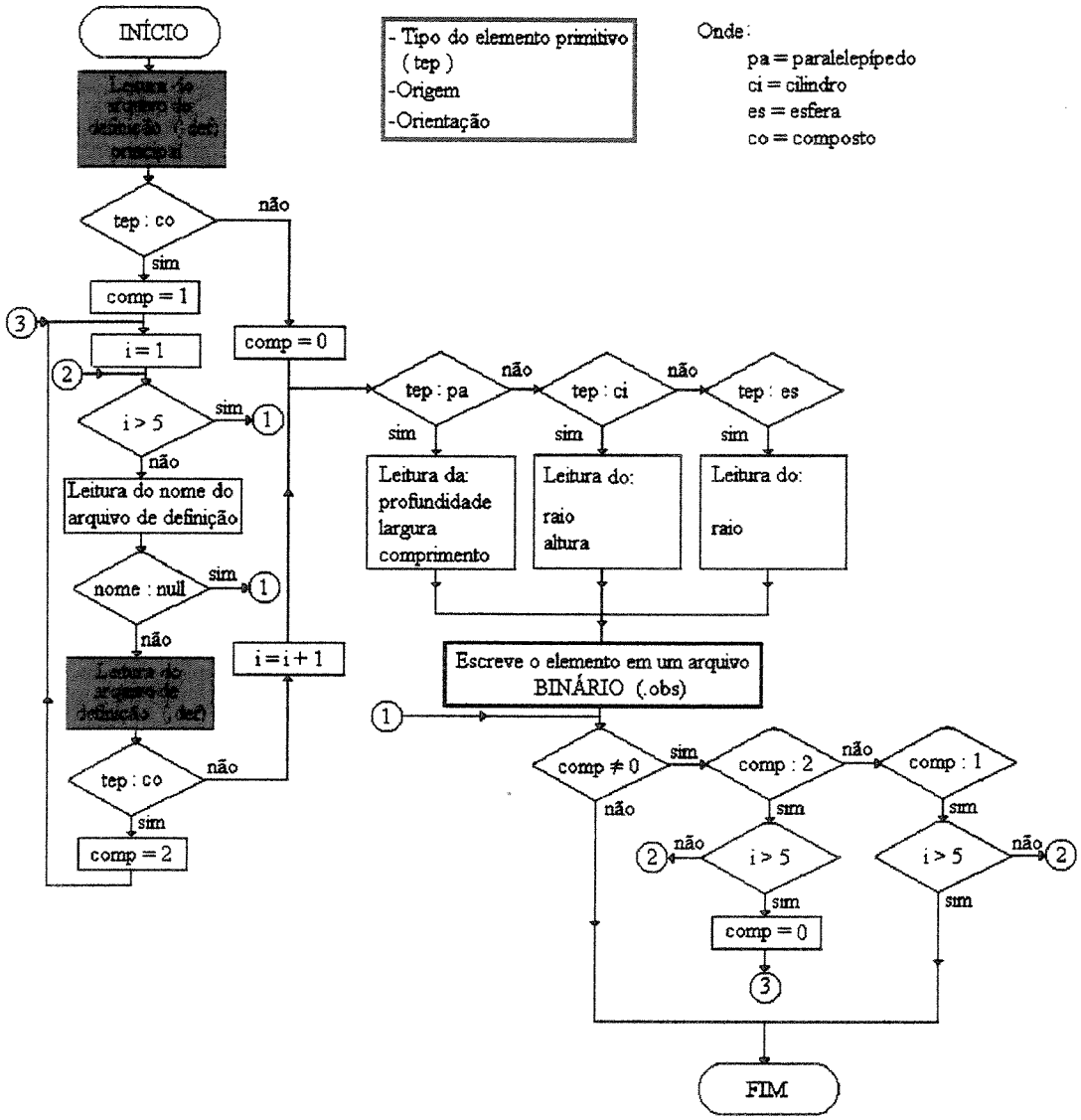


Figura I.4: Algoritmo para a criação do ambiente de trabalho.

I.4 Modelo geométrico do manipulador Kraft

A seguir será apresentado o modelo geométrico obtido para o manipulador Kraft (Kraft, 1985) através dos métodos de Denavit-Hartenberg e por vetores locais, apresentados no Capítulo 3.

I.4.1 O manipulador

O manipulador Kraft, figura I.5, possui seis juntas rotacionais e foi desenvolvido para executar tarefas gerais em ambientes submarinos. Os seus movimentos são comandados à distância através de um controle chamado "master", figura I.6, que é um modelo em escala reduzida do manipulador. Suas trajetórias podem ser comandadas pelo operador ou por programações pré-definidas.

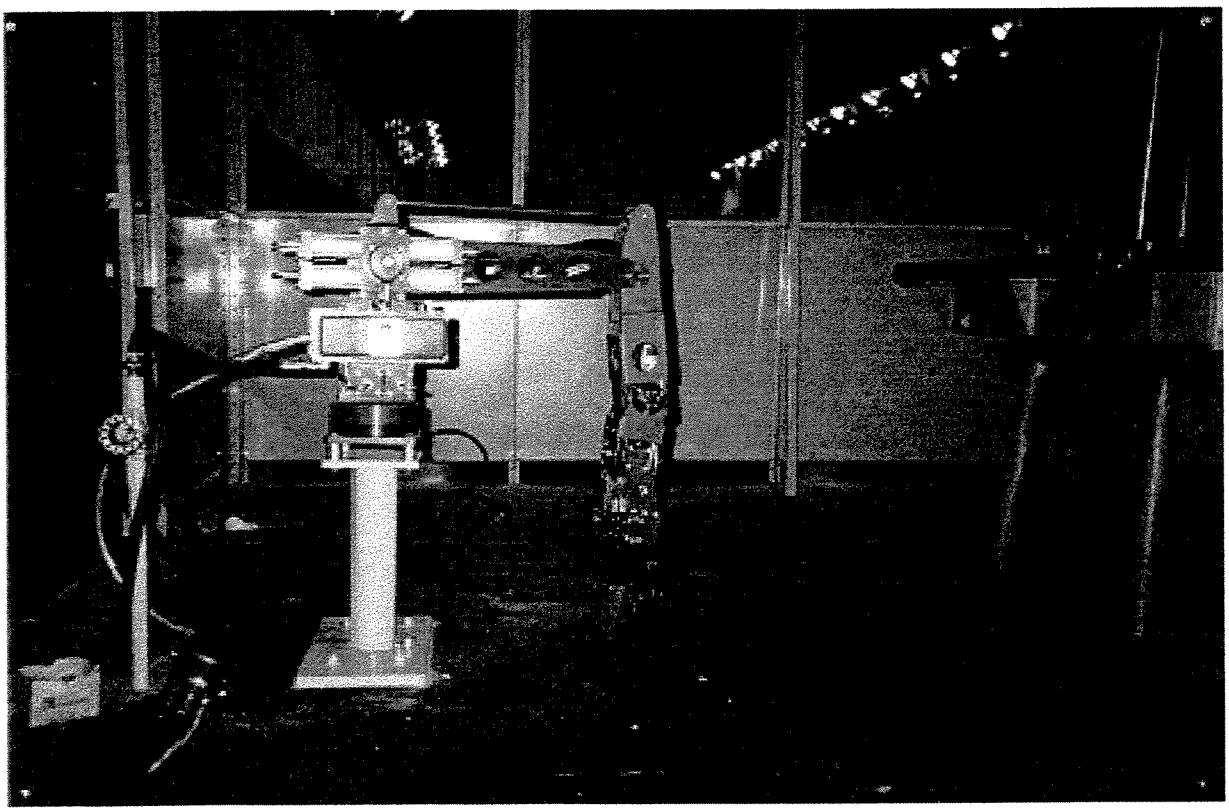


Figura I.5 Manipulador submarino Kraft ("slave").

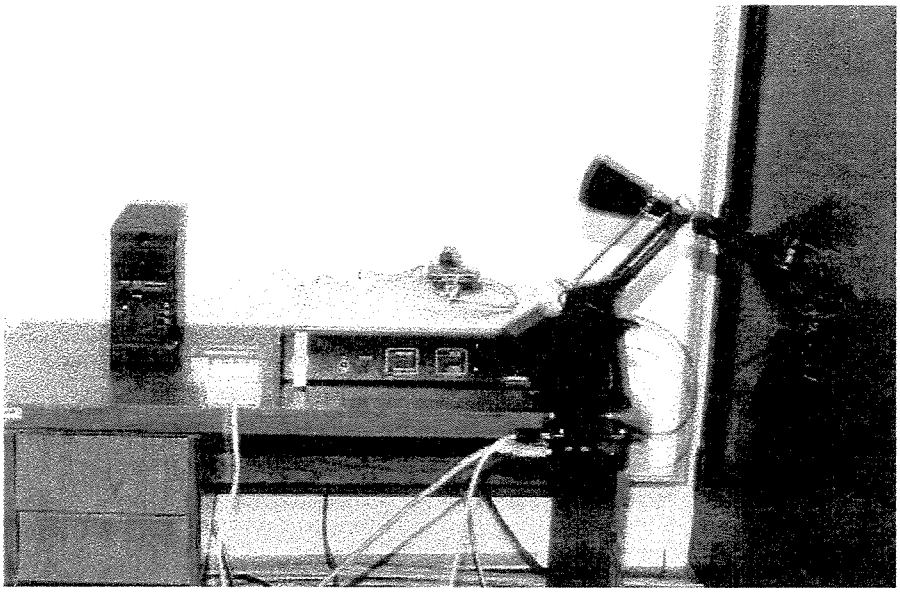


Figura I.6 "Master" do manipulador Kraft (manipulo à direita).

I.4.2 Sistemática de Denavit-Hartenberg

A tabela I.1 mostra os parâmetros de Denavit-Hartenberg obtidos para o manipulador Kraft e a tabela I.2, as matrizes de passagem.

JUNTA	θ (graus)	D (mm)	α (graus)	A (mm)	range(graus)
1	θ_1	352.43	90.0	0.0	- 90.0 /+ 90.
2	θ_2	0.0	0.0	532.65	0 / + 120.0
3	θ_3	0.0	0.0	264.32	-20.0/-130.0
4	θ_4	0.0	-90.0	132.16	-42.0 /+58.0
5	θ_5	48.06	90.0	0.0	+ 52.5 / + 52.5
6	θ_6	48.06	0.0	0.0	- 90.0 /+ 90.0

Tabela I.1: Parâmetros de Denavit-Hartenberg

$$T_{0,1}=\begin{bmatrix} c_1 & 0 & s_1 & 0 \\ s_1 & 0 & -c_1 & 0 \\ 0 & 1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad T_{1,2}=\begin{bmatrix} c_2 & -s_2 & 0 & a_2c_2 \\ s_2 & c_2 & 0 & a_2s_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_{2,3}=\begin{bmatrix} c_3 & -s_3 & 0 & a_3c_3 \\ s_3 & c_3 & 0 & a_3s_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad T_{3,4}=\begin{bmatrix} c_4 & 0 & -s_4 & a_4c_4 \\ s_4 & 0 & c_4 & a_4s_4 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_{4,5}=\begin{bmatrix} c_5 & 0 & s_5 & 0 \\ s_5 & 0 & -c_5 & 0 \\ 0 & 1 & 0 & d_5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad T_{5,6}=\begin{bmatrix} c_6 & -s_6 & 0 & 0 \\ s_6 & c_6 & 0 & 0 \\ 0 & 0 & 1 & d_6 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Onde: $C_i = \cos(\theta_i)$ e $S_i = \text{sen}(\theta_i)$ para i = 1, 2, ..., 6

Tabela I.2: Matrizes de passagem.

As equações obtidas para o manipulador, a partir da equação 3.2 e 3.3, são apresentadas abaixo.

Orientação

$n_x = c_1 (c_5 c_6 c_{234} - s_6 s_{234}) - s_1 s_5 c_6,$

$n_y = s_1 (c_5 c_6 c_{234} - s_6 s_{234}) + c_1 s_5 c_6,$

$n_z = c_5 c_6 s_{234} - s_6 c_{234},$

$s_x = - c_1 (c_5 s_6 c_{234} + c_6 s_{234}) + s_1 s_5 s_6,$

$s_y = - s_1 (c_5 s_6 c_{234} + c_6 s_{234}) - c_1 s_5 s_6,$

$s_z = - c_5 s_6 s_{234} + c_6 c_{234},$

$a_x = c_1 s_5 c_{234} + s_1 c_5,$

$a_y = s_1 s_5 c_{234} - c_1 c_5,$

$a_z = s_5 s_{234}.$

Posição

$$p_x = d_6 (c_1 s_5 c_{234} + s_1 c_5) + c_1 (- d_5 s_{234} + a_4 c_{234} + a_3 c_{23} + a_2 c_2),$$

$$p_y = d_6 (s_1 s_5 c_{234} - c_1 c_5) + s_1 (- d_5 s_{234} + a_4 c_{234} + a_3 c_{23} + a_2 c_2),$$

$$p_z = d_6 s_5 s_{234} + d_5 c_{234} + a_4 s_{234} + a_3 s_{23} + a_2 s_2 + d_1 .$$

Onde

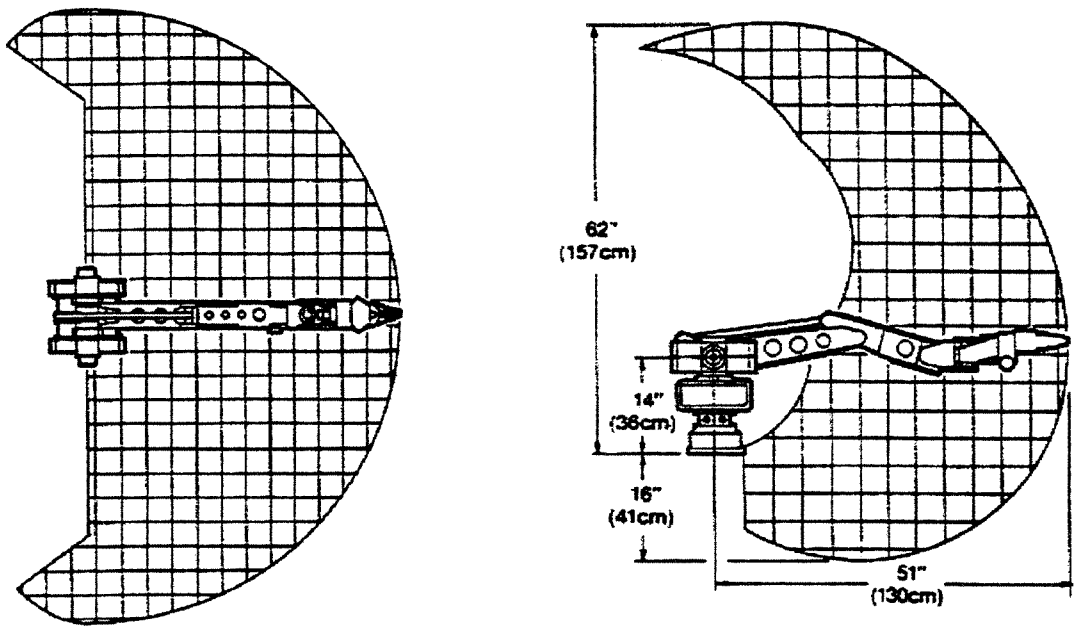
$$c_{23} = c_2 c_3 - s_2 s_3 ,$$

$$s_{23} = s_2 c_3 + s_3 c_2 ,$$

$$c_{234} = c_{23} c_4 - s_{23} s_4 ,$$

$$s_{234} = c_{23} s_4 - s_{23} c_4 .$$

A figura I.7 mostra o volume de trabalho do manipulador Kraft.



(a) (b)

Figura I.7: Volume de trabalho do manipulador Kraft: (a) planta e (b) face.

I.4.3 Através de vetores Locais

Os vetores de rotação são dados por:

$\vec{V}r_1 = [\,0\,,0\,,1\,] - R1$ $\quad$ $\vec{V}r_2 = [\,0\,,1\,,0\,] - R2$ $\quad$ $\vec{V}r_3 = [\,0\,,1\,,0\,] - R3$

$\vec{V}r_4 = [\,0\,,1\,,0\,] - R4$ $\quad$ $\vec{V}r_5 = [\,0\,,0\,,1\,] - R5$ $\quad$ $\vec{V}r_6 = [\,1\,,0\,,0\,] - R6$

As matrizes de rotação são:

$$R_1(z,\theta_1)\quad R_2(z,\theta_2)\quad R_3(z,\theta_3)\quad R_4(z,\theta_4)\quad R_5(z,\theta_5)\quad R_6(z,\theta_6)$$

onde $\quad R_i(z,\theta_i) = \begin{bmatrix} c\theta_i & -s\theta_i & 0 \\ s\theta_i & c\theta_i & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad R_i(y,\theta_i) = \begin{bmatrix} c\theta_i & 0 & s\theta_i \\ 0 & 1 & 0 \\ -s\theta_i & 0 & c\theta_i \end{bmatrix}$

e $\quad c_i = \cos(\theta_i)$;

$s_i = \text{sen}(\theta_i)$.

para $i = 1, 2, \ldots, 6$.

A tabela I.3 apresenta as dimensões para o manipulador Kraft.

Parâmetro	Dimensão (mm)	Parâmetro	dimensão mm	Parâmetro	dimensão mm
c1	200.0	a2	532.65	d63	200.24
a31	128.15	a4	132.16	d7	200.00
a32	264.32	d61	180.22		
d1	352.43	d62	48.06		

Tabela I.3: Dimensões do manipulador em milímetros.

Os vetores de translação ($\vec{V}_i$), especificados em relação ao referencial XYZ são dados por:

$\vec{V}_1 = [\,0\,,\,0\,,\,d1\,] - R1$

$\vec{V}_2 = [\,0\,,\,c1\,,\,0\,] - R2$

$\vec{V}_3 = [\,a2\,,\,0\,,\,0\,] - R3$

$\vec{V}_4 = [\,-a31\,,\,0\,,\,0\,] - R4$

$\vec{V}_5 = [\,a32\,,\,0\,,\,0\,] - R5$

$\vec{V}_6 = [\,a4\,,\,0\,,\,0\,] - R6$

$\vec{V}_7 = [\,d61\,,\,0\,,\,d62\,] - R7$

$\vec{V}_8 = [\,d63\,,\,0\,,\,0\,] - R8$

$\vec{V}_9 = [\,0\,,\,0\,,\,0\,] - R9$

A matriz (Rt_a) e o vetor ($\vec{V}$ t_a) de correção de erros identificados[1] são dados por:

$$RT\_A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\vec{V}\,t\_a = [0 \quad 0 \quad 0]$$

As matrizes de transformação M_T(i) associadas a cada elo são dadas por:

$M\_T(1) = M\_V * Rt\_a$

$M\_T(2) = M\_T(1) * R_1$

$M\_T(3) = M\_T(2) * R_2 = M\_T(1) * R_1 * R_2$

$M\_T(4) = M\_T(3) * R_3 = M\_T(1) * R_1 * R_2 * R_3$

$M\_T(5) = M\_T(4) * R_4 = M\_T(1) * R_1 * R_2 * R_3 * R_4$

$M\_T(6) = M\_T(5) * R_5 = M\_T(1) * R_1 * R_2 * R_3 * R_4 * R_5$

$M\_T(7) = M\_T(6) * R_6 = M\_T(1) * R_1 * R_2 * R_3 * R_4 * R_5 * R_6$

As equações para a determinação da posição dos pontos de interesse são apresentadas abaixo:

$$\vec{O}(0) = M\_T(1) * \vec{V}t\_a$$

$$\vec{O}(1) = \vec{O}(0) + M\_T(2) * \vec{V}_1$$

$$\vec{O}(2) = \vec{O}(0) + M\_T(2) * \vec{V}_2$$

$$\vec{O}(3) = \vec{O}(1) + M\_T(3) * \vec{V}_3$$

$$\vec{O}(4) = \vec{O}(3) + M\_T(4) * \vec{V}_4$$

$$\vec{O}(5) = \vec{O}(3) + M\_T(4) * \vec{V}_5$$

$$\vec{O}(6) = \vec{O}(5) + M\_T(5) * \vec{V}_6$$

$$\vec{O}(7) = \vec{O}(6) + M\_T(6) * \vec{V}_7$$

$$\vec{O}(8) = \vec{O}(7) + M\_T(7) * \vec{V}_8$$

$$\vec{O}(9) = \vec{O}(8) + M\_T(7) * \vec{V}_9$$

[1] Neste estudo não se preocupou em inserir valores medidos para estes parâmetros, pois não está dentro do escopo deste trabalho.

ANEXO II

Determinação da Matriz Jacobiana

A matriz Jacobiana intervém diretamente na solução numérica do modelo cinemático direto e em outras aplicações em robótica sendo, deste modo, de grande importância. Paul [1981] utiliza matrizes de transformação 4 x 4 para obter translações e rotações diferenciais sobre um sistema de coordenadas do qual a matriz Jacobiana pode ser derivada.

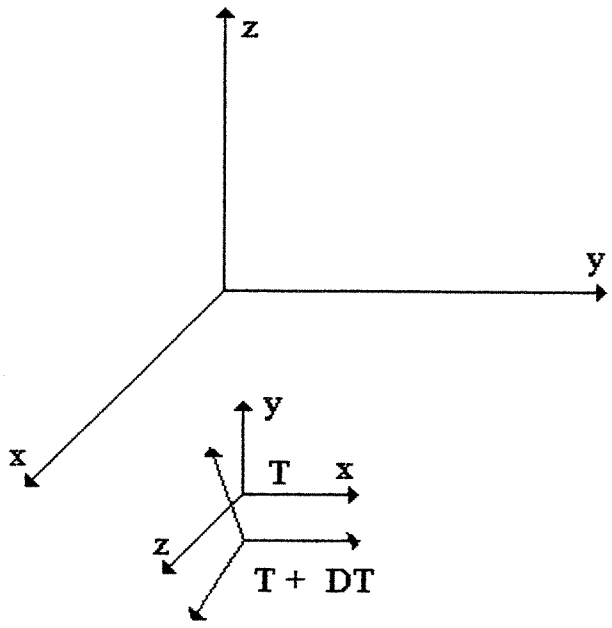


Figura II.1: Deslocamento infinitesimal T.

Dado um sistema de coordenadas T, uma mudança diferencial em T, figura II.1, corresponde a uma translação e uma rotação diferencial ao longo e sobre o sistema de coordenadas da base, isto é:

$$T + dT = \begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & -\delta_z & \delta_y & 0 \\ \delta_z & 1 & -\delta_x & 0 \\ -\delta_y & \delta_x & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} . T \qquad \text{(II.1)}$$

ou

$$dT = \left[\begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & -\delta_z & \delta_y & 0 \\ \delta_z & 1 & -\delta_x & 0 \\ -\delta_y & \delta_x & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} - \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \right] T = \Delta T \qquad \text{(II.2)}$$

onde

$$\Delta \triangleq \begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & -\delta_z & \delta_y & 0 \\ \delta_z & 1 & -\delta_x & 0 \\ -\delta_y & \delta_x & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} - \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \text{(II.3)}$$

$\underline{\delta} = (\delta_x , \delta_y , \delta_z)^t$ é a rotação diferencial sobre os eixos principais do sistema de coordenadas da base e $\underline{d} = (d_x , d_y , d_z)^t$ é a translação ao longo dos eixos principais do sistema de coordenadas da base. Similarmente, a mudança diferencial em T pode ser expressa como uma translação e uma rotação diferenciais em torno do sistema de coordenadas T:

$$T + dT = T \begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & -\delta_z & \delta_y & 0 \\ \delta_z & 1 & -\delta_x & 0 \\ -\delta_y & \delta_x & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{(II.4)}$$

ou

$$dT = T \left[\begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & -\delta_z & \delta_y & 0 \\ \delta_z & 1 & -\delta_x & 0 \\ -\delta_y & \delta_x & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} - \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \right]$$

$$\triangleq (T)({}^{T}\Delta) \quad \text{(II.5)}$$

onde ${}^{T}\Delta$ possui a mesma estrutura da equação (II.3), exceto que as definições de $\underline{\delta}$ e $\underline{d}$ são diferentes. $\underline{\delta} = (\delta_x , \delta_y , \delta_z)^t$ é a rotação diferencial sobre os eixos principais do sistema de coordenadas T e $\underline{d} = (d_x , d_y , d_z)^t$ é a translação ao longo dos eixos principais do sistema de coordenadas T.

A partir das equações (II.2) e (II.5), obtêm-se a relação entre Δ e ${}^{T}\Delta$:

$$\Delta T = (T)({}^{T}\Delta)$$

ou

$${}^{T}\Delta = T^{-1} \Delta T \quad \text{(II.6)}$$

Fu [1987] define a inversa de uma matriz de transformação homogênea como sendo:

$$T^{-1} = \begin{bmatrix} n_x & n_y & n_z & -\underline{n}^T \underline{p} \\ s_x & s_y & s_z & -\underline{s}^T \underline{p} \\ a_x & a_y & a_z & -\underline{a}^T \underline{p} \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \text{(II.7)}$$

Usando-se a equação (II.7), a equação (II.6) torna-se

$$^{T}\Delta = T^{-1}\,\Delta T = \begin{bmatrix} \underline{n}(\underline{\delta} \times \underline{n}) & \underline{n}(\underline{\delta} \times \underline{s}) & \underline{n}(\underline{\delta} \times \underline{a}) & \underline{n}(\underline{\delta} \times \underline{p}) + \underline{d}\,\underline{n} \\ \underline{s}(\underline{\delta} \times \underline{n}) & \underline{s}(\underline{\delta} \times \underline{s}) & \underline{s}(\underline{\delta} \times \underline{a}) & \underline{s}(\underline{\delta} \times \underline{p}) + \underline{d}\,\underline{n} \\ \underline{a}(\underline{\delta} \times \underline{n}) & \underline{a}(\underline{\delta} \times \underline{s}) & \underline{a}(\underline{\delta} \times \underline{a}) & \underline{a}(\underline{\delta} \times \underline{p}) + \underline{d}\,\underline{n} \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad \text{(II.8)}$$

onde $\underline{\delta} = (\delta_x , \delta_y , \delta_z)^t$ é a rotação diferencial sobre os eixos principais do sistema de coordenadas da base e $\underline{d} = (d_x , d_y , d_z)^t$ é a translação ao longo dos eixos principais do sistema de coordenadas da base. Utilizando as identidades vetoriais

$$\underline{x}\,(\underline{y} \times \underline{z}) = -\,\underline{y}\,(\underline{x} \times \underline{z}) = \underline{y}\,(\underline{z} \times \underline{x})$$

$$\underline{x}\,(\underline{x} \times \underline{y}) = 0$$

a equação (II.6) torna-se

$$ {}^{T}\Delta = \begin{bmatrix} 0 & -\underline{\delta}\ (\underline{n}\times\underline{s}) & \underline{\delta}\ (\underline{a}\times\underline{n}) & \underline{\delta}\ (\underline{p}\times\underline{n})+\underline{d}\ \underline{n} \\ \underline{\delta}\ (\underline{n}\times\underline{s}) & 0 & \underline{\delta}\ (\underline{s}\times\underline{a}) & \underline{\delta}\ (\underline{p}\times\underline{s})+\underline{d}\ \underline{s} \\ -\underline{\delta}\ (\underline{a}\times\underline{n}) & \underline{\delta}\ (\underline{s}\times\underline{a}) & 0 & \underline{\delta}\ (\underline{p}\times\underline{a})+\underline{d}\ \underline{a} \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad \text{(II.9)} $$

Desde que os vetores $\underline{n}$, $\underline{s}$ e $\underline{a}$ sejam ortonormais, isto é:

$$ \underline{n}\times\underline{s}=\underline{a} \qquad \underline{s}\times\underline{a}=\underline{n} \qquad \underline{a}\times\underline{n}=\underline{s} $$

então a equação (II.9) torna-se:

$$ {}^{T}\Delta = \begin{bmatrix} 0 & -\underline{\delta}\ \underline{a} & \underline{\delta}\ \underline{s} & \underline{\delta}\ (\underline{p}\times\underline{n})+\underline{d}\ \underline{n} \\ \underline{\delta}\ \underline{a} & 0 & \underline{\delta}\ \underline{n} & \underline{\delta}\ (\underline{p}\times\underline{s})+\underline{d}\ \underline{s} \\ -\underline{\delta}\ \underline{s} & \underline{\delta}\ \underline{n} & 0 & \underline{\delta}\ (\underline{p}\times\underline{a})+\underline{d}\ \underline{a} \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad \text{(II.10)} $$

Se fizermos os elementos da matriz de ${}^{T}\Delta$ serem:

$$ {}^{T}\Delta = \begin{bmatrix} 0 & -{}^{T}\delta_z & {}^{T}\delta_y & {}^{T}d_x \\ {}^{T}\delta_z & 0 & -{}^{T}\delta_x & {}^{T}d_y \\ -{}^{T}\delta_y & {}^{T}\delta_x & 0 & {}^{T}d_z \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad \text{(II.11)} $$

e igualando os elementos das equações (II.10) e (II.11) têm-se:

$$ {}^{T}d_x = \underline{\delta}\,(\underline{p} \times \underline{n}) + \underline{d}\,\underline{n} = \underline{n}\,[\,(\underline{\delta} \times \underline{p}\,) + \underline{d}\,] $$

$$ {}^{T}d_y = \underline{\delta}\,(\underline{p} \times \underline{s}) + \underline{d}\,\underline{s} = \underline{s}\,[\,(\underline{\delta} \times \underline{p}\,) + \underline{d}\,] $$

$$ {}^{T}d_z = \underline{\delta}\,(\underline{p} \times \underline{a}) + \underline{d}\,\underline{a} = \underline{a}\,[\,(\underline{\delta} \times \underline{p}\,) + \underline{d}\,] \qquad \text{(II.12)} $$

$$ {}^{T}\delta_x = \underline{\delta}\,\underline{n} $$

$$ {}^{T}\delta_y = \underline{\delta}\,\underline{s} $$

$$ {}^{T}\delta_z = \underline{\delta}\,\underline{a} $$

Expressando o conjunto de equações II.12 na forma matricial, têm-se:

$$ \begin{bmatrix} {}^{T}d_x \\ {}^{T}d_y \\ {}^{T}d_z \\ {}^{T}\delta_x \\ {}^{T}\delta_y \\ {}^{T}\delta_z \end{bmatrix} = \begin{bmatrix} [\,\underline{n},\,\underline{s},\,\underline{a}\,]^T & \left[(\underline{p}\times\underline{n}),(\underline{p}\times\underline{s}),(\underline{p}\times\underline{a})\right]^T \\ 0 & [\,\underline{n},\,\underline{s},\,\underline{a}\,]^T \end{bmatrix} \begin{bmatrix} d_x \\ d_y \\ d_z \\ \delta_x \\ \delta_y \\ \delta_z \end{bmatrix} \qquad \text{(II.13)} $$

onde 0 é uma submatriz 3×3,com todos os elementos iguais a zero. A equação (II.13) mostra a relação da translação e rotação diferenciais do sistema de coordenadas da base em relação a translação e rotação diferenciais do sistema de coordenadas de T.

Aplicando a equação (II.5) à equação cinemática de um robô de seis graus de liberdade, obtém-se o diferencial de ${}^{0}T_{6}$:

$$d\,{}^{0}T_{6} = {}^{0}T_{6}\,{}^{T_6}\Delta \tag{II.14}$$

No caso de um robô de seis graus de liberdade, um movimento diferencial na junta *i* pode induzir uma mudança equivalente em ${}^{0}T_{6}\,{}^{T_6}\Delta$:

$$d\,{}^{0}T_{6} = {}^{0}T_{6}\,{}^{T_6}\Delta = {}^{0}A_{1}\,{}^{1}A_{2}\ldots{}^{i-2}A_{i-1}\,{}^{i-1}\Delta_{i}\,{}^{i-1}A_{i}\ldots{}^{5}A_{6} \tag{II.15}$$

onde ${}^{i-1}\Delta_{i}$ é definido como a transformação diferencial ao longo e sobre o eixo da junta *i* do movimento e é definido como:

$${}^{i-1}\Delta_{i} = \begin{bmatrix} 0 & -d\theta_{i} & 0 & 0 \\ d\theta_{i} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{II.16}$$

A partir da equação (II.15), obtêm-se ${}^{T_6}\Delta$, devido a mudança diferencial no movimento da junta *i*:

$${}^{T_6}\Delta = ({}^{i-1}A_i\,{}^{i}\Delta_{i+1}\ldots{}^{5}A_6)^{-1}\;{}^{i-1}\Delta_i\;({}^{i-1}A_i\,{}^{i}\Delta_{i+1}\ldots{}^{5}A_6)$$

ou

$${}^{T_6}\Delta = U_i^{-1}\;{}^{i-1}\Delta_i\,U_i \qquad \text{(II.17)}$$

onde

$$U_i = {}^{i-1}A_i\,{}^{i}\Delta_{i+1}\ldots{}^{5}A_6$$

Expressando U_i na forma geral da matriz de transformação homogênea, isto é:

$$U_i = \begin{bmatrix} n_x & s_x & a_x & p_x \\ n_y & s_y & a_y & p_y \\ n_z & s_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \text{(II.18)}$$

e utilizando as equações (II.16) e (II.18) a equação (II.17) torna-se:

$${}^{T_6}\Delta = \begin{bmatrix} 0 & -a_z & s_z & p_x n_y - p_y n_x \\ a_z & 0 & -n_z & p_x s_y - p_y s_x \\ -s_z & n_z & 0 & p_x a_y - p_y a_x \\ 0 & 0 & 0 & 0 \end{bmatrix} d\theta_i \qquad \text{(II.19)}$$

A partir dos elementos de ${}^{T_6}\Delta$ definidos na equação (II.11), e igualando os elementos das matrizes nas equações (II.11) e (II.19), têm-se:

$$\begin{bmatrix} {}^{T_6}d_x \\ {}^{T_6}d_y \\ {}^{T_6}d_z \\ {}^{T_6}\delta_x \\ {}^{T_6}\delta_y \\ {}^{T_6}\delta_z \end{bmatrix} = \begin{bmatrix} p_x n_y - p_y n_x \\ p_x s_y - p_y s_x \\ p_x a_y - p_y a_x \\ n_z \\ s_z \\ a_z \end{bmatrix} d\theta_i \qquad \text{(II.20)}$$

onde i = 1, 2, . . ., 6.

Então, o Jacobiano de um robô pode ser obtido a partir da equação (II.20). Para um robô de seis graus de liberdade com juntas rotacionais:

$$\begin{bmatrix} {}^{T_6}d_x \\ {}^{T_6}d_y \\ {}^{T_6}d_z \\ {}^{T_6}\delta_x \\ {}^{T_6}\delta_y \\ {}^{T_6}\delta_z \end{bmatrix} = J(\underline{q}) \begin{bmatrix} dq_1 \\ dq_2 \\ dq_3 \\ dq_4 \\ dq_5 \\ dq_6 \end{bmatrix} \qquad \text{(II.21)}$$

onde, as colunas da matriz Jacobiana são obtidas a partir da equação (II.20).

A matriz Jacobiana para um robô que possui juntas rotacionais e prismáticas ao mesmo tempo é apresentado em Fenton (1984).

ANEXO III

Pseudoinversa (Inversa Generalizada)

Em muitos casos, a solução de um sistema de equações lineares existe, mesmo quando a inversa da matriz não existe. Este problema e muitos outros podem ser resolvidos através do conceito da pseudoinversa, ou inversa generalizada (Nashed, 1976), da matriz.

A inversa generalizada deve atender a algumas das seguintes propriedades para obter sucesso:

i) deve reduzir-se a A^{-1} se A é não singular;

ii) deve sempre existir;

iii) deve possuir algumas das propriedades da inversa (ou modificações destas)

iv) quando usadas no lugar da inversa, deve ser capaz de proporcionar respostas sensíveis para questões importantes tal como; consistência das equações, ou soluções de mínimos quadrados.

Moore e Penrose definiram a pseudoinversa de uma matriz, A^{+}, como a única solução para:

$$A \times A^{+} \times A = A$$

$$A^{+} \times A \times A^{+} = A^{+}$$

$$(A \times A^{+})^{t} = A \times A^{+}$$

$$(A \times A^{+})^{t} = A^{+} \times A$$

III.1 Determinação da matriz Pseudoinversa pelo método de Greville

Seja a_k a matriz formada pela k-ésima coluna de A e A_k a matriz formada pelas k primeiras colunas de A.

A matriz pseudoinversa de A_k denotada por A_k^{+}, é obtida (Gorla, 1984) pela equação:

$$A_k^{+} = \begin{bmatrix} A_{k-1}^{+} - d_k b_k \\ \hline b_k \end{bmatrix} \qquad \text{(III.1)}$$

onde

$$\underline{d}_k = A_{k-1}^{+} \underline{a}_k \qquad \text{(III.2)}$$

$$\underline{b}_k = \begin{cases} (\underline{c}_k^{t} \underline{c}_k)^{-1} \underline{c}_k^{t} & \text{se } \underline{c}_k = 0 \\ (1 - \underline{d}_k^{t} \underline{d}_k)^{-1} \underline{d}_k^{t} A_{k-1}^{+} & \text{se } \underline{c}_k \neq 0 \end{cases} \qquad \text{(III.3)}$$

onde $\underline{c}_k = \underline{a}_k - A_{k-1} \underline{d}_k$ para $k = 1, 2, 3, \ldots, n$.

A inicialização das iterações depende da primeira coluna de A:

$$A_1^+ = \begin{cases} \vec{0}^t & \text{se } \underline{a}_1 = 0 \\ \underline{a}_1^t / (\underline{a}_1^t \underline{a}_1) & \text{se } \underline{a}_1 \neq 0 \end{cases} \qquad \text{(III.4)}$$

O fluxograma para a implementação deste método é apresentado na figura III.1.

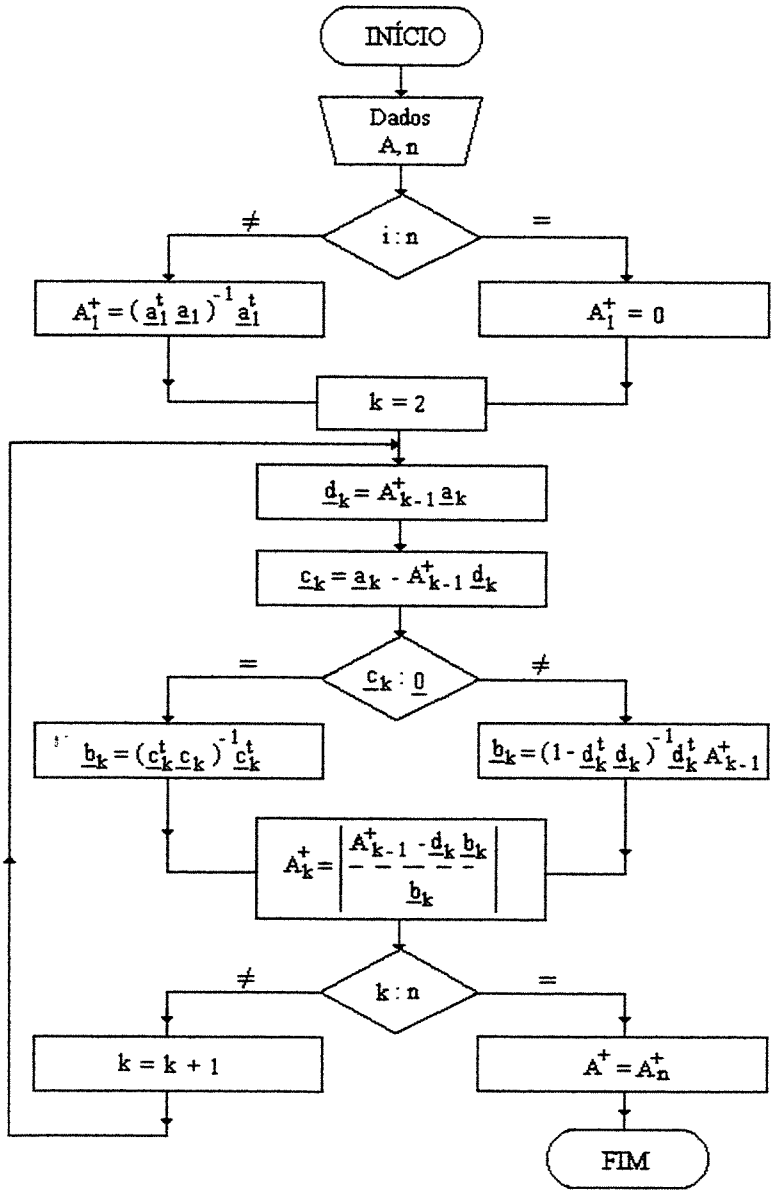


Figura III.1: Fluxograma para a determinação da matriz pseudoinversa pelo método de Greville.

ANEXO IV

Estrutura dos Arquivos Definição - Modelo Geométrico do Ambiente de Trabalho

Como mencionado anteriormente, cada linha deste arquivo possui uma única informação, dependendo apenas do elemento primitivo a que este arquivo se refere.

A seguir será apresentada a estrutura deste tipo de arquivo para cada elemento primitivo implementado.

Os elementos primitivos implementados são: pavê, cylindre e sphere. Com o objetivo de se utilizar todos os elementos primitivos ao mesmo tempo foi implementado um "elemento primitivo" denominado compose.

IV.1 Elemento primitivo Pavê (Paralelepípedo)

O quadro IV.1 mostra a estrutura do arquivo de definição para este elemento primitivo. O objeto gerado a partir deste arquivo terá a forma de um paralelepípedo.

-- PAINEL_VERTICAL	
PAVE	
-- POSICAO	
0.0	
0.0	Posição em **mm.**
0.0	
-- ORIENTACAO	
0.0	
0.0	Orientação em **graus.**
0.0	
-- DIMENSOES	
13.0	Profundidade em **mm.**
500.0	Largura em **mm.**
1000.0	Comprimento em **mm.**

Quadro IV.1: Arquivo de definição com o parâmetro pave.

IV.2 Elemento primitivo Cylindre (Cilindro)

O quadro IV.2 mostra a estrutura do arquivo de definição para este elemento primitivo. O objeto gerado a partir deste arquivo terá a forma de um cilindro.

-- cilindro meio	
CYLINDRE	
-- POSICAO	
0.0	
250.0	Posição em **mm.**
500.0	
-- ORIENTACAO	
0.0	
90.0	Orientação em **graus.**
0.0	
-- DIMENSOES	
-- RAIO	
15.0	Raio do cilindro em **mm.**
-- ALTURA	
20.0	Altura do cilindro em **mm.**

Quadro IV.2: Arquivo de definição com o parâmetro cylindre.

Fornecendo a altura o valor 0 (zero) temos um *círculo* com o raio especificado.

IV.3 Elemento primitivo Sphere (Esfera)

O quadro IV.1 mostra a estrutura do arquivo de definição para este elemento primitivo. O objeto gerado a partir deste arquivo terá a forma de uma esfera.

-- esfera	
SPHERE	
-- POSICAO	
0.0	
0.0	Posição em **mm.**
0.0	
-- ORIENTAÇÃO	
0.0	
45.0	Orientação em **graus.**
30.0	
-- DIMENSOES	
-- RAIO	
150.0	Raio da esfera em **mm.**

Quadro IV.3: Arquivo de definição com o parâmetro sphere.

IV.4 "Elemento primitivo" Compose (Composto)

O objeto gerado a partir deste arquivo será uma composição de diversos arquivos de definição cada um contendo as especificações de um elemento primitivo.

Este arquivo pode referir-se a cinco outros arquivos de definição mas pode-se utilizar um número menor. Apenas é necessário que a palavra **null** (em minúsculo) seja escrita ate completar cinco arquivos.

O quadro IV.1 mostra a estrutura do arquivo de definição para este elemento primitivo.

-- cilindros verticais	
COMPOSE	Indica que o arquivo é composto.
-- POSICAO	
0.0	Posição em **mm.**
0.0	
0.0	
-- ORIENTACAO	
0.0	
0.0	Orientação em **graus.**
0.0	
-- ARQUIVOS INTEGRANTES	
CILIND_S.DEF;	
CILIND_I.DEF;	
null;	Arquivos integrantes.
null;	
null;	

Quadro IV.4: Arquivo de definição com o parâmetro compose.

ANEXO V

Módulos Testes

Os pacotes desenvolvidos, como mencionado anteriormente, podem ser testados separadamente permitindo, desta forma, uma maior confiabilidade dos módulos finais gerados. Para a implementação dos três módulos (Trajeto, Simula e Obstáculo) foram gerados dois módulos testes: um para testar o modelo geométrico direto e inverso e outro para testar procedimento de obtenção dos ângulos Rpy (roll, pitch e yaw).

As seguir estes dois módulos serão descritos.

V.1 Módulo Teste_Rpy

Este módulo permite realizar testes com os ângulos Roll, Pitch e Yaw. Estes ângulos expressam a orientação do elemento terminal em relação às coordenadas da base. A figura V.1 mostra a tela principal deste módulo. Este módulo realiza basicamente dois tipos de testes que serão descritos a seguir.

V.1.1 Obtenção dos ângulos Roll, Pitch e Yaw a partir da matriz de orientação

Neste teste é dada a matriz de orientação e a partir deste são obtidos os ângulos Roll, Pitch e Yaw. Os valores obtidos estão em graus.

V.1.2 Obtenção da matriz de orientação a partir dos ângulos Roll, Pitch e Yaw

Neste teste são dados inicialmente os valores dos ângulos roll, pitch e yaw (em graus). A partir destes ângulos é calculada a matriz orientação e a partir desta matriz são calculados os ângulos roll, pitch e yaw novamente. Isto é feito para se ter certeza de que os valores obtidos para a matriz de orientação correspondem aos ângulos roll, pitch e yaw, inicialmente dados.

V.2 Módulo Teste_Modelo

Este módulo permite realizar o teste do modelo geométrico direto e, inverso obtido para o robô. A figura V.2 mostra a tela principal deste módulo.

Para realizar o teste são dados inicialmente os ângulos das juntas e, a partir do modelo geométrico direto serão calculados a posição e a orientação do robô. Depois, a partir da posição e orientação obtidas são calculados, a partir do modelo geométrico inverso, os ângulos das juntas.

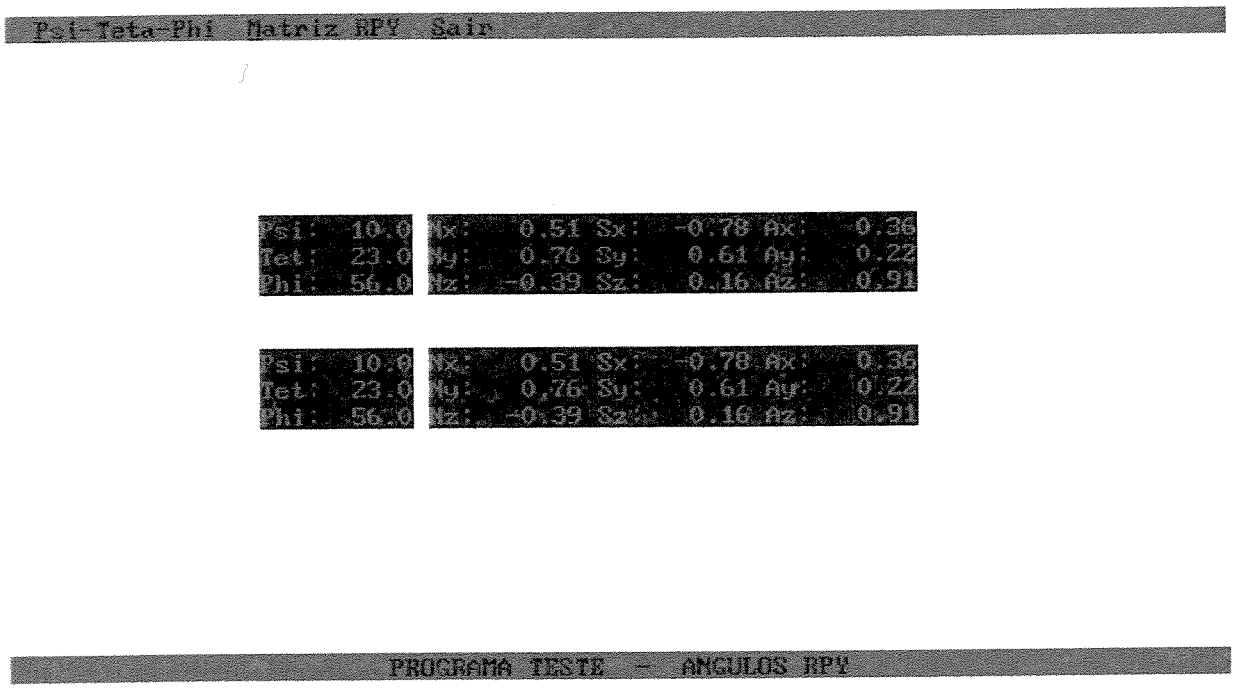


Figura V.1: Tela principal do módulo teste Teste_Rpy.

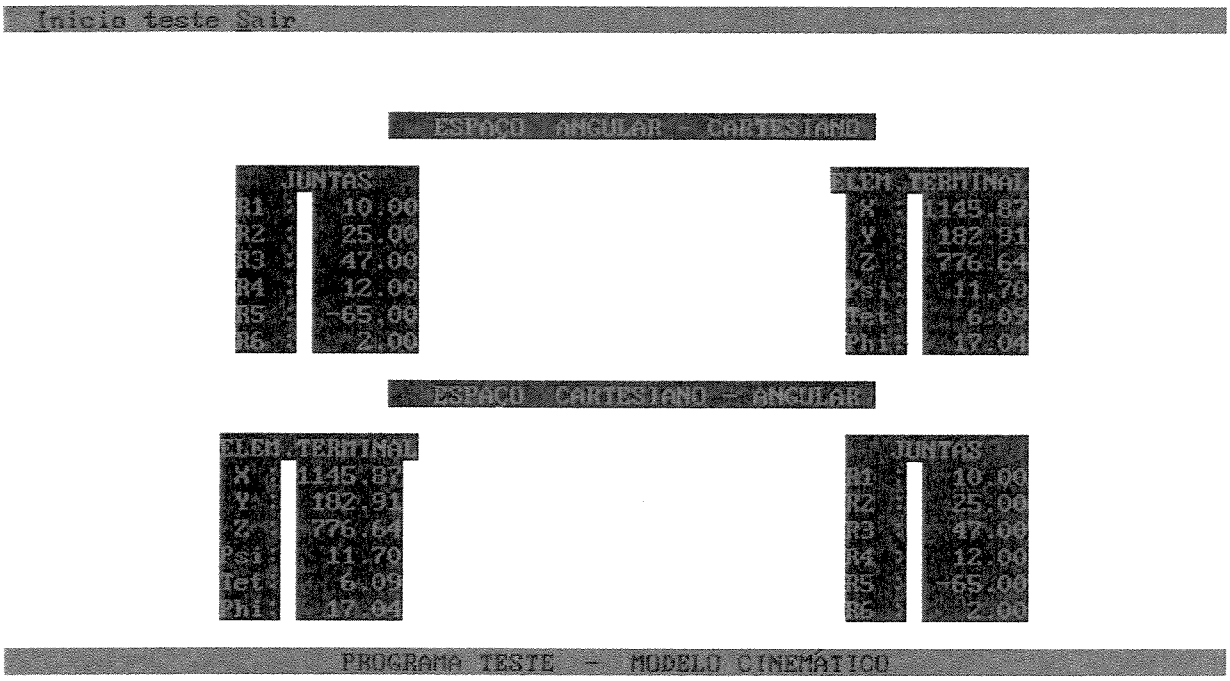


Figura V.2: Tela principal do módulo teste Teste_Modelo.

ANEXO VI

Descrição dos Pacotes que Compõem a Biblioteca em Linguagem ADA

Os módulos são gerados a partir dos pacotes descritos abaixo.

- Cad_Types: definições dos diversos tipos e variáveis utilizadas em outros pacotes tais como definições de matrizes, vetores, entre outros.

- Math_Cad: contém as funções matemáticas básicas utilizadas em outros pacotes, tais como as funções seno, coseno, tangente, multiplicação de matrizes e vetores, soma de variáveis, entre outras.

- Cad_Input: possui as rotinas e procedimentos para a interface do programa com o usuário.

- Graph: Contém definições de primitivas básicas de desenhos (linha, ponto).

- Graphic: É responsável pela manipulação e representação dos elementos desejados em janelas no monitor, controla a inserção de elementos nas janelas, desenha e apaga janelas, define a posição, cor, tamanho, etc.

- Cad_Keyboard: interface entre o programa e o teclado.

- Adv_Math: contém as funções matemáticas avançadas utilizadas em outros pacotes, tais como o cálculo da matriz inversa, métodos matemáticos para a criação de trajetórias, entre outras.

- Cad_Video: biblioteca em que estão os recursos de gestão do monitor, tais como escrever na tela, apagar a tela, entre outras.

- Cad_Files: possui recursos para a abertura, leitura, alterações e fechamento de arquivos de dados.

- Basic_Graph: possui funções básicas para a construção de círculos, semicírculos, cones, cilindros, entre outros, para a construção gráfica do dispositivo robótico em estudo.

- High_Level_Graph: interface gráfica de alto nível.

- Cad_Collisions: contém funções em que se realizam testes de colisões para o dispositivo robótico em estudo.

- Gráfico: contém as funções para a construção e visualização dos gráficos para as trajetórias angulares e espaciais obtidas no programa Trajeto.

- Parameters: contém as definições das dimensões, vetores locais limites das juntas, entre outros, do dispositivo robótico em estudo.

- Test: contém funções de teste dos limites da junta para o dispositivo robótico em estudo.

- Model: contém as funções para o cálculo do modelo direto e inverso e o modelo geométrico para o dispositivo em estudo.

- Robô: contém as funções que desenham o dispositivo em estudo na tela.

- Interp_Filtr: contém as funções responsáveis pela interpolação e filtragem de uma trajetória angular (.ref).

- Trajetória: contém as funções responsáveis pela geração de arquivos angulares para o dispositivo robótico em estudo.

- Cad_Obstáculo: contém as funções responsáveis pela geração de obstáculos, ferramentas e ambientes de trabalho para o robô.

- Menu_Simula: Tem a função de criar o ambiente de trabalho do programa Simula.

- Menu_Trajeto: Tem a função de criar o ambiente de trabalho do programa Trajeto.

- Menu_Obstáculo: Tem a função de criar o ambiente de trabalho do programa Obstáculo.

- Obstáculo: Gera o programa Obstáculo.

- Simula: Gera o programa Simula.

- Trajeto: Gera o programa Trajeto.

ANEXO VII

Teclas Rápidas Utilizadas pelos Módulos

Estas teclas permitem um acesso rápido a determinadas funções que foram implementadas nos módulos Simula, Trajeto e Obstacle.

VII.1 Teclas relativas a opção Vista

Estas teclas e suas funções são apresentadas na tabela VII.1. São utilizadas no módulos Simula e Obstáculo.

VII.2 Teclas que definem as funções que as teclas 1, 2, 3, 4, 5 e 6 realizarão

Estas teclas e suas funções são apresentadas na tabela VII.2. São utilizadas pelo módulo Simula.

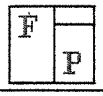
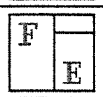
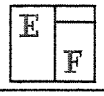
Tecla	Representação	Descrição
F1		Mostra a visualização do robô em três modos de vista possíveis: face, planta e esquerda.
F2		Mostra a visualização do robô em dois modos de vista: face e planta.
F3		Mostra a visualização do robô em dois modos de vista: face e esquerda.
F4		Mostra a visualização do robô em dois modos de vista: planta e esquerda.
F5		Mostra a visualização do robô em dois modos de vista: planta e face.
F6		Mostra a visualização do robô em dois modos de vista: esquerda e face.
F7		Mostra a visualização do robô em dois modos de vista: esquerda e planta.

Tabela VII.1: Teclas relativas a opção Vista.

Tecla	Representação	Descrição
U	JT + θ	Geométrico direto - dá ao ângulo o valor digitado no campo.
I	$\underline{X}+\Delta\underline{X}$	Geométrico direto - incrementa o valor do ângulo desejado de um valor pré definido.
O	$\underline{X}-\Delta\underline{X}$	Geométrico direto - decresce o valor do ângulo desejado de um valor pré definido.
K	JT-Δθ	Cinemático Inverso - determina o valor do ângulo das juntas a partir do decréscimo (de um valor pré definido) em um dos valores da posição (X, Y ou Z) ou de um dos valores da orientação (Psi, Teta ou Phi).
L	JT+Δθ	Cinemático inverso - determina o valor do ângulo das juntas a partir do incremento (de um valor pré definido) em um dos valores da posição (X, Y ou Z) ou de um dos valores da orientação (Psi, Teta ou Phi).

Tabela VII.2: Teclas que definem as funções que as teclas 1, 2, 3, 4, 5 e 6 realizarão.

VII.3 Teclas que definem as funções que as teclas ↑, ↓, ← e → realizarão

Estas teclas e suas funções são apresentadas na tabela VII.3. São utilizadas pelos módulos Simula e Obstáculo.

Tecla	Representação	Descrição
P	0 0 0	Zera a orientação de **visualização** do objeto na tela.
B	← → ↑ ↓	Faz com que as setas tenham a função de posicionar o objeto na tela.
N		Faz com que as setas ↑, ↓, ← e → tenham a função de mudar o ponto de visualização do objeto na tela.
H		Faz com que as setas ← e → girem o objeto em torno do eixo Z (Phi).
J		Faz com que as setas ← e → girem o objeto em torno do eixo Y (Psi).

Tabela VII.3: Teclas que definem as funções que as teclas ↑, ↓, ← e → realizarão.

VII.4 Teclas relativas a movimentação das juntas do robô

Estas teclas e suas funções são apresentadas na tabela VII.4. São utilizadas no módulo Simula.

VII.5 Teclas relativas à opção de zoom, posicionamento e orientação do robô ou obstáculo na tela

Estas teclas e suas funções são apresentadas na tabela VII.5. São utilizadas pelos módulos Simula e Obstáculo.

Tecla	Representação	Descrição
1	JT1 X	- Incrementa ou decresce o valor da junta 1 de um valor pré definido. - Dá ao ângulo da junta 1 um valor final desejado. - Incrementa ou decresce o valor da posição X de um valor pré definido.
2	JT2 Y	- Incrementa ou decresce o valor da junta 2 de um valor pré definido. - Dá ao ângulo da junta 2 um valor final desejado. - Incrementa ou decresce o valor da posição Y de um valor pré definido.
3	JT3 Z	- Incrementa ou decresce o valor da junta 3 de um valor pré definido. - Dá ao ângulo da junta 3 um valor final desejado. - Incrementa ou decresce o valor da posição Z de um valor pré definido.
4	JT4 PSI	- Incrementa ou decresce o valor da junta 4 de um valor pré definido. - Dá ao ângulo da junta 4 um valor final desejado. - Incrementa ou decresce o valor da orientação Psi um valor pré definido.
5	JT5 TETA	- Incrementa ou decresce o valor da junta 5 de um valor pré definido. - Dá ao ângulo da junta 5 um valor final desejado. - Incrementa ou decresce o valor da orientação Teta de um valor pré definido.
6	JT6 PHI	- Incrementa ou decresce o valor da junta 6 de um valor pré definido. - Dá ao ângulo da junta 6 um valor final desejado. - Incrementa ou decresce o valor da orientação Phi de um valor pré definido.
0	JT (1..6) = 0	Zera todas as juntas do robô.

Tabela VII.4: Teclas relativas à movimentação das juntas do robô.

Tecla	Representação	Descrição
-	ZOOM -	Diminui o objeto de um valor pré definido.
+	ZOMM +	Aumenta o objeto de um valor pré definido.
↑	 ↑	- Posiciona o objeto na tela, para cima. - Gira o objeto em torno do eixo X (Teta).
↓	 ↓	- Posiciona o objeto na tela, para baixo. - Gira o objeto em torno do eixo X (Teta).
←	 ←	- Posiciona o objeto para o lado esquerdo da tela - Gira o objeto em torno do eixo Y ou Z dependendo da funçã atribuída.
→	 →	- Posiciona o objeto para o lado direito da tela - Gira o objeto em torno do eixo Y ou Z dependendo da funçã atribuída.

Tabela VII.5: Teclas relativas à opção de zoom e posicionamento e orientação do robô ou obstáculo na tela.

VII.6 Teclas relativas à alteração da vista ativa

Estas teclas e suas funções são apresentadas na tabela VII.6. São utilizadas nos módulos Simula e Obstáculo.

Tecla	Representação	Descrição
Page Up		Altera a vista ativa, seguindo a sequência Face > Planta > Esquerda.
Page Dowm		Altera a vista ativa, seguindo a sequência Face > Esquerda > Planta.

Tabela VII.6: Teclas relativas à alteração da vista ativa.

VII.7 Teclas relativas à navegação pelos nomes dos arquivos que geram o obstáculo

Estas teclas e suas funções são apresentadas na tabela VII.7, no módulo Obstáculo.

Tecla	Representação	Descrição
Home		Navega no sentido do primeiro nome do arquivo integrante de um obstáculo.
End		Navega no sentido do último nome do arquivo integrante de um obstáculo.

Tabela VII.7: Teclas relativas à alteração da vista ativa.

VII.8 Teclas relativas à navegação pelos pontos (conjunto de ângulos) dos arquivos .ref

Estas teclas e suas funções são apresentadas na tabela VII.8. Elas são utilizadas pelos módulos Trajeto e Simula.

Tecla	Representação	Descrição
Z	Ant.	Vai para o ponto (conjunto de ângulos) posterior ao ponto atual.
X	Post.	Vai para o ponto (conjunto de ângulos) anterior ao ponto atual.

Tabela VII.8: Teclas relativas à navegação pelos pontos (conjunto de ângulos) dos arquivos .ref.

VII.9 Barra de espaços e tecla "Esc"

Estas teclas e suas funções são apresentadas na tabela VII.9 são utilizadas pelo módulo Simula.

Tecla	Representação	Descrição
Barra de Espaços		Avança para o próximo conjunto de ângulos da trajetória, carregado durante a simulação no modo manual.
Esc	Esc	Para a simulação da tarefa quando está no modo automático.

Tabela VII.9: Tecla utilizada pelo módulo Simula.

A figura VII.1 mostra o a disposição das teclas rápidas no teclado de um microcomputador.

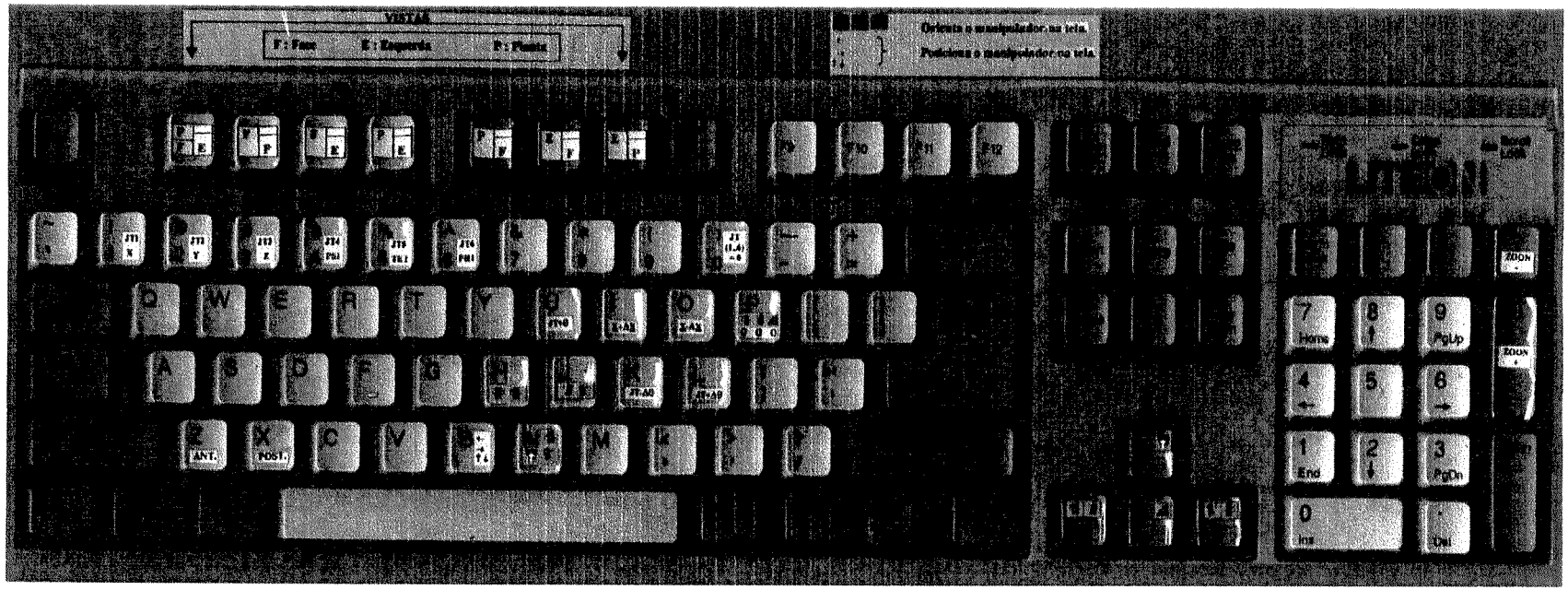


Figura VII.1: Disposição das teclas rápidas no teclado

ANEXO VIII

Descrição dos Menus dos Módulos Implementados

A seguir serão descritas as funções dos menus dos módulos implementados. As extensões dos arquivos que serão citados no decorrer deste anexo são descritas no Anexo XI.

VIII.1 Descrição dos menus do módulo Trajeto

Abaixo serão descritas as funções dos menus do módulo de criação de Trajetórias.

VIII.1.1 Menu Arquivo

Apresenta as opções de leitura de arquivos de trajetórias, salvar e informação de arquivos, entre outros. As opções são:

Ler Arq. (arquivo)

Angular: Lê os arquivos com extensão .ref (********, ******sm, ******fl) e automaticamente plota as curvas.

Espacial: Lê os arquivos com extensão .esp (espacial) e automaticamente plota as curvas.

Salvar (.ref): Salva os arquivos com extensão .ref (********, ******sm, ******fl).

Informação: Mostra informações da trajetória carregada, tais como: nome, número, total de pontos, pontos de passagem no caso de trajetórias interpoladas e outras.

Reduzir Arq. (arquivo): Lê um arquivo de extensão .ref e reduz o número de linhas e grava com o nome ******sm.ref (simula)

VIII.1.2 Menu Trajetória

Apresenta as opções de criação de trajetórias, interpolação e filtragem e alteração de arquivos carregados, entre outros. As opções são:

Criar: Gera os arquivos angulares a partir da entrada da posição e orientação finais desejadas, com o formato ********.ref e também o arquivo ******sm.ref (com um número menor de pontos) para ser utilizado para a simulação no programa Simula.

Tipo

Simples: Indica que a trajetória terá apenas um segmento.

Composta: Permite planejar a trajetória com o número de segmentos desejados (trajetória composta).

Unir traj.: Permite concatenar, em um mesmo arquivo, n arquivos diferentes de trajetórias angulares (extensão .ref).

Int. e Filt.: Interpola e filtra uma trajetória angular com extensão .ref; a trajetória a ser interpolada e filtrada pode ser criada tanto no programa Trajeto quanto no programa Simula (interpola no máximo 1500 pontos)

Destruir ponto: Destrói o ponto (conjunto de ângulos) atual que está sendo observado (apenas na trajetória do tipo ********.ref).

Posterior: Vai para o ponto (conjunto de ângulos) posterior ao ponto atual da trajetória carregada (nas trajetória do tipo .ref).

Anterior: Vai para o ponto (conjunto de ângulos) anterior ao ponto atual da trajetória carregada (nas trajetória do tipo .ref).

Vai para: Vai para o ponto (conjunto de ângulos) desejado de uma determinada trajetória carregada (nas trajetória do tipo .ref).

VIII.1.3 Menu Espacial

Apresenta as opções de criação de um arquivo espacial a partir de um arquivo angular e plotar os gráficos a partir destes dados. As opções são:

Cria: Gera o arquivo espacial ********.esp para a visualização no espaço da trajetória seguida pelo elemento terminal do robô, a partir do arquivo ********.ref.

Plota

X-Y e X-Z: Plota a projeção da trajetória espacial, realizada pelo elemento terminal do robô, nos planos x-y e x-z.

Z-Y: Plota a projeção da trajetória espacial, realizada pelo elemento terminal do robô, no plano y-z.

VIII.1.4 Menu Parâmetros

Apresenta as opções de alteração dos parâmetros para a geração de uma trajetória angular.

As opções são:

1 Erro de posição: Erro máximo permitido de posição em relação à posição final desejada, em mm.

2 Erro de orientação: Erro máximo permitido de orientação em relação à orientação final desejada, em graus.

Caminho (divisão): Indica o número em que os segmentos determinados no menu Trajetória - Tipo (Simples - Composta) serão divididos.

Discretização

Linear: Discretiza os segmentos determinados no menu Trajetória - Tipo(Simples - Composta) de forma Linear, fazendo desta forma com que o elemento terminal do robô siga uma linha reta. O número de pontos em que o segmento é discretizado é indicado no menu Parâmetros – Caminho (Divisão).

Semicírculo: Discretiza os segmentos determinados no menu Trajetória –Tipo (Simples - Composta) em forma de um semicírculo.

1 plano X-Y var. Z: Discretiza os segmentos determinados no menu Trajetória – Tipo (Simples - Composta) em forma de um semicírculo no plano X-Y (onde Y = f(X)), podendo variar o valor da posição Z, fazendo desta forma com que o elemento terminal do robô siga um semicírculo. O número de pontos em que o segmento é discretizado é indicado no menu Parâmetros – Caminho (Divisão).

2 plano X-Z var. Y: Discretiza os segmentos determinados no menu Trajetória - Tipo (Simples - Composta) em forma de um semicírculo no plano X-Z (onde Z = f(X)), podendo variar o valor da posição Y, fazendo desta forma com que o

elemento terminal do robô siga um semicírculo. O número de pontos em que o segmento é discretizado é indicado no menu Parâmetros – Caminho (Divisão).

3 plano Y-Z var. X: Discretiza os segmentos determinados no menu Trajetória - Tipo (Simples - Composta) em forma de um semicírculo no plano Y-Z (onde Y = f(Z)), podendo variar o valor da posição X, fazendo desta forma com que o elemento terminal do robô siga um semicírculo. O número de pontos em que o segmento é discretizado é indicado no menu Parâmetros – Caminho (Divisão).

Direção (+ ou -)

Positiva: Indica que o semicírculo vai na direção positiva do eixo a que se refere.

Negativa: Indica que o semicírculo vai na direção negativa do eixo a que se refere.

VIII.2 Descrição dos menus do módulo Simula

Abaixo serão descritas as funções do módulo de simulação gráfica.

VIII.2.1 Menu Vista

Apresenta as opções de visualização na tela dos modos de vista possíveis para o robô em estudo. As opções são:

1 Face-Planta-Esquerda: Mostra a visualização do robô em três modos de vista possíveis: face, planta e esquerda.

2 Face-Planta: Mostra a visualização do robô em dois modos de vista: face e planta.

*3 **Face-Esquerda***: Mostra a visualização do robô em dois modos de vista: face e esquerda.

*4 **Planta-Esquerda***: Mostra a visualização do robô em dois modos de vista: planta e esquerda.

*5 **Planta-Face***: Mostra a visualização do robô em dois modos de vista: planta e face.

*6 **Esquerda-Face***: Mostra a visualização do robô em dois modos de vista: esquerda e face.

*7 **Esquerda-Planta***: Mostra a visualização do robô em dois modos de vista: esquerda e planta.

VIII.2.2 Menu Parâmetros

Permite a mudança dos parâmetros do teclado, tais como fator de multiplicação para o zoom, deslocamento através das setas para posicionar o robô na tela de visualização, entre outros. As opções são:

Deslocamento

Passo: Altera o valor da variável de posicionamento do robô na tela; esta opção permite uma melhor visualização do mesmo; seu valor é dado em pixels.

Alto: Movimenta o robô (na vista ativa) para cima, com número de pixels estipulado no valor do *Passo*.

Baixo: Movimenta o robô (na vista ativa) para baixo, com número de pixels estipulado no valor do *Passo*.

Direita: Movimenta o robô (na vista ativa) para a direita, com número de pixels estipulado no valor do *Passo*.

Esquerda: Movimenta o robô (na vista ativa) para a esquerda, com número de pixels estipulado no valor do *Passo*.

Escala

Passo: Altera o valor da variável de zoom - esta opção permite uma melhor visualização de detalhes do robô.

Aumentar: Aplica um zoom positivo à imagem do robô (na vista ativa) no valor estipulado acima.

Diminui: Aplica um zoom negativo à imagem do robô (na vista ativa) no valor estipulado acima.

Orientação

1 Direção Psi: Altera a posição do robô no espaço de um ângulo Psi, rotação em torno do eixo X, para uma melhor visualização do mesmo.

2 Direção Teta: Altera a posição do robô no espaço de um ângulo Teta, rotação em torno do eixo Y, para uma melhor visualização do mesmo.

3 Direção Phi: Altera a posição do robô no espaço de um ângulo Phi, rotação em torno do eixo Z, para uma melhor visualização do mesmo.

Padrão (0, 0, 0): Volta a posição do robô no espaço em todas as vistas para a posição original (Psi = 0, Teta = 0 e Phi = 0).

Passo Teclado Angular: Altera o valor da variável que movimenta as juntas do robô no espaço angular.

Passo Teclado Cartesiano: Altera o valor da variável que movimenta as juntas do robô no espaço cartesiano.

VIII.2.3 Menu Trajetória

Apresenta as opções referentes à alteração de um arquivo de trajetória carregado na memória. As opções são:

Posterior: Vai para o ponto (conjunto de ângulos) posterior ao ponto atual da trajetória carregada.

Anterior: Vai para o ponto (conjunto de ângulos) anterior ao ponto atual da trajetória carregada.

Inserir: Insere um ponto (conjunto de ângulos) na trajetória posterior ao ponto atual da trajetória carregada, ou insere um ponto em um arquivo novo.

Destruir: Destrói o ponto (conjunto de ângulos) atual que está sendo indicado.

Guardar: Guarda o conjunto de ângulos referentes à posição e orientação atuais do robô.

Vai para: Vai para o ponto (conjunto de ângulos) desejado de uma determinada trajetória carregada.

Simulação

Manual: Simula o movimento do robô a partir do arquivo que está carregado ponto por ponto com a utilização da tecla de espaço.

Automática: Simula o movimento do robô a partir do arquivo que está carregado ponto por ponto automaticamente

VIII.2.4 Menu Modelo

As opções disponíveis possíveis são:

Geométrico Direto

1 Rotação 1: rotaciona a junta número 1 do robô de uma quantidade desejada em graus.

2 Rotação 2: rotaciona a junta número 2 do robô de uma quantidade desejada em graus.

3 Rotação 3: rotaciona a junta número 2 do robô de uma quantidade desejada em graus.

4 Rotação 4: rotaciona a junta número 2 do robô de uma quantidade desejada em graus.

5 Rotação 5: rotaciona a junta número 2 do robô de uma quantidade desejada em graus.

6 Rotação 6: rotaciona a junta número 2 do robô de uma quantidade desejada em graus.

Cinemático Inverso

X – Eixo X: Faz com que o robô ande na direção X de uma quantidade desejada em mm.

Y – Eixo Y: Faz com que o robô ande na direção Y de uma quantidade desejada em mm.

Z – Eixo Z: Faz com que o robô ande na direção Z de uma quantidade desejada em mm.

1 – Direção Psi: Muda a orientação do elemento terminal do robô de um ângulo determinado em torno de X.

2 – Direção Teta: Muda a orientação do elemento terminal do robô de um ângulo determinado em torno de Y.

3 – Direção Phi: Muda a orientação do elemento terminal do robô de um ângulo determinado em torno de Z.

Cálculo Matricial: Faz os cálculos da posição final das juntas, através do modelo cinemático inverso, do robô com os dados que foram entrados acima.

Modelo Ferramenta

X – Eixo X: Muda a posição da ferramenta do robô, permanentemente, de um valor determinado na direção do eixo de X.

Y – Eixo Y: Muda a posição da ferramenta do robô, permanentemente, de um valor determinado na direção do eixo de Y.

Z – Eixo Z: Muda a posição da ferramenta do robô, permanentemente, de um valor determinado na direção do eixo de Z.

1 – Direção Psi: Muda a orientação da ferramenta do robô, permanentemente, de um ângulo Psi em torno de X.

*2 – **Direção Teta***: Muda a orientação da ferramenta do robô, permanentemente, de um ângulo Teta em torno de Y.

*3 – **Direção Phi***: Muda a orientação da ferramenta do robô, permanentemente, de um ângulo Phi em torno de Z.

Teste de Colisão

*1 **Fim de Curso***: Realiza o teste dos limites físicos de cada junta do robô.

*2 **Robô/Robô***: Realiza o teste de colisão do robô com ele próprio, para determinado conjunto de ângulos das juntas.

*3 **Robô/Ferramenta***: Realiza o teste de colisão do robô com a ferramenta em uso, para determinado conjunto de ângulos das juntas.

*4 **Robô/Painel***: Realiza o teste de colisão do robô com o painel inserido no meio ambiente, para determinado conjunto de ângulos das juntas.

*5 **Robô/Módulo***: Realiza o teste de colisão do robô com o módulo de trabalho, para determinado conjunto de ângulos das juntas.

*6 **Robô/Meio ambiente***:

*7 **Ferramenta/Painel***: Realiza o teste de colisão da ferramenta utilizada pelo robô com o painel inserido no meio, ambiente para determinado conjunto de ângulos das juntas.

*8 **Ferramenta/Módulo***: Realiza o teste de colisão da ferramenta utilizada pelo robô com o módulo de trabalho, para determinado conjunto de ângulos das juntas.

9 Ferramenta/Meio ambiente:

0 Todos os testes: Realiza todos os testes acima de uma só vez.

Segurança (var.): Permite a mudar o valor da variável de segurança (prevista nas equações, 6.8 e 6.13, para a determinação da existência ou não de colisão - Capítulo 6).

VIII.2.5 Menu Arquivo

Apresenta as opções de leitura de arquivos de trajetórias, obstáculos, painéis e salvar, entre outros. As opções são:

Trajetória Atual: Carrega um arquivo .ref para a simulação gráfica.

Anterior: Carrega a trajetória precedente indicada no cabeçalho do arquivo (.ref) que está carregado.

Módulo de Trabalho: Insere um módulo de trabalho no ambiente de trabalho do robô, arquivo com extensão .obs.

Painel de Atuação: Insere um painel de trabalho no ambiente de trabalho do robô, arquivo com extensão .obs.

Base Móvel: Insere uma base móvel no robô, arquivo com extensão .obs.

Obstáculos: Insere um obstáculo no ambiente de trabalho do robô, arquivo com extensão .obs.

Ferramenta: Insere uma ferramenta na ponta do robô, arquivo com extensão .obs.

Correção (arq.):

Salvar : Salva a trajetória (.ref) que está carregada na memória do computador.

VIII.3 Descrição dos menus do módulo Obstáculo

Abaixo serão descritas as funções do módulo de construção de obstáculos.

VIII.3.1 Menu Vista

Apresenta as opções de visualização na tela dos pontos de vista possíveis para o robô em estudo.

1 Face-Planta-Esquerda: Mostra a visualização do robô em três pontos de vista possíveis face, planta e esquerda.

2 Face-Planta: Mostra a visualização do robô em dois pontos de vista: face e planta.

3 Face-Esquerda: Mostra a visualização do robô em dois pontos de vista: face e esquerda.

4 Planta-Esquerda: Mostra a visualização do robô em dois pontos de vista: planta e esquerda.

5 Planta-Face: Mostra a visualização do robô em dois pontos de vista: planta e face.

6 Esquerda-Face: Mostra a visualização do robô em dois pontos de vista: esquerda e face.

7 Esquerda-Planta: Mostra a visualização do robô em dois pontos de vista: esquerda e planta.

VIII.3.2 Menu Criar

Cria um arquivo de obstáculo (********.obs), que será utilizado no módulo Simula, a partir do arquivo de definição (********.def).

VIII.3.3 Menu Dados

Apresenta a posição e orientação das partes que compõem o obstáculo, criado a partir do(s) arquivo(s) de definição(ões). A posição das partes que compõem o obstáculo é indicada por uma seta.

1: ***do arquivo***: Mostra na tela os dados das partes que compõem o obstáculo a partir do arquivo de origem, isto é, os dados dos arquivos de definição.

2: ***do meio amb. (ambiente)***: Mostra na tela os dados das partes que compõem o obstáculo em relação ao meio ambiente.

VIII.4 Descrição das informações que aparecem nas telas dos módulos implementados

A seguir serão apresentadas as informações que aparecem nas telas dos módulos implementados.

As figuras VIII.1, VIII.2 e VIII.3 mostram as informações que aparecem nas telas dos módulos Trajeto, Simula e Obstáculo, respectivamente.

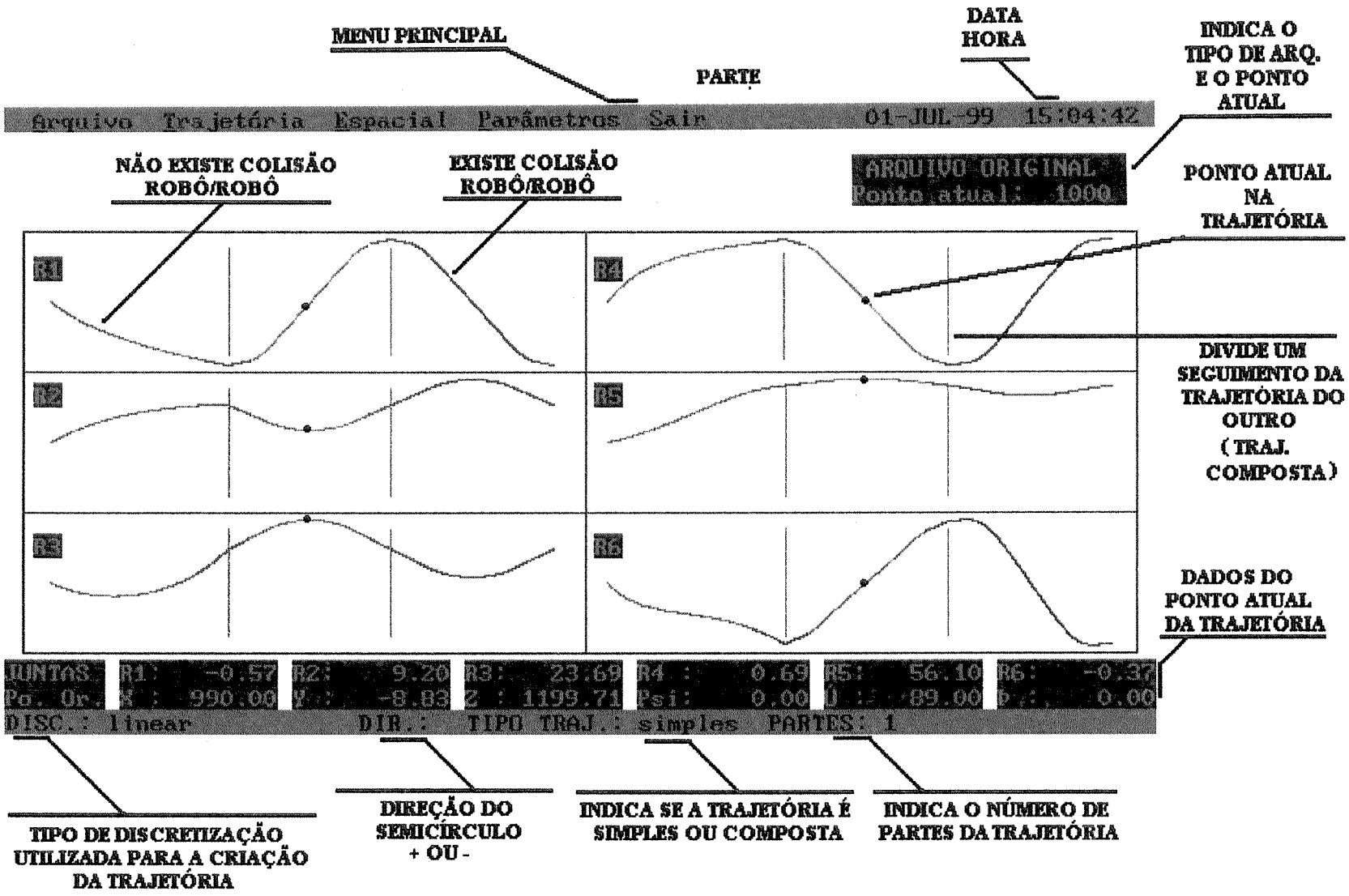


Figura VIII.1: Descrição das informações que aparecem na tela do módulo Trajeto.

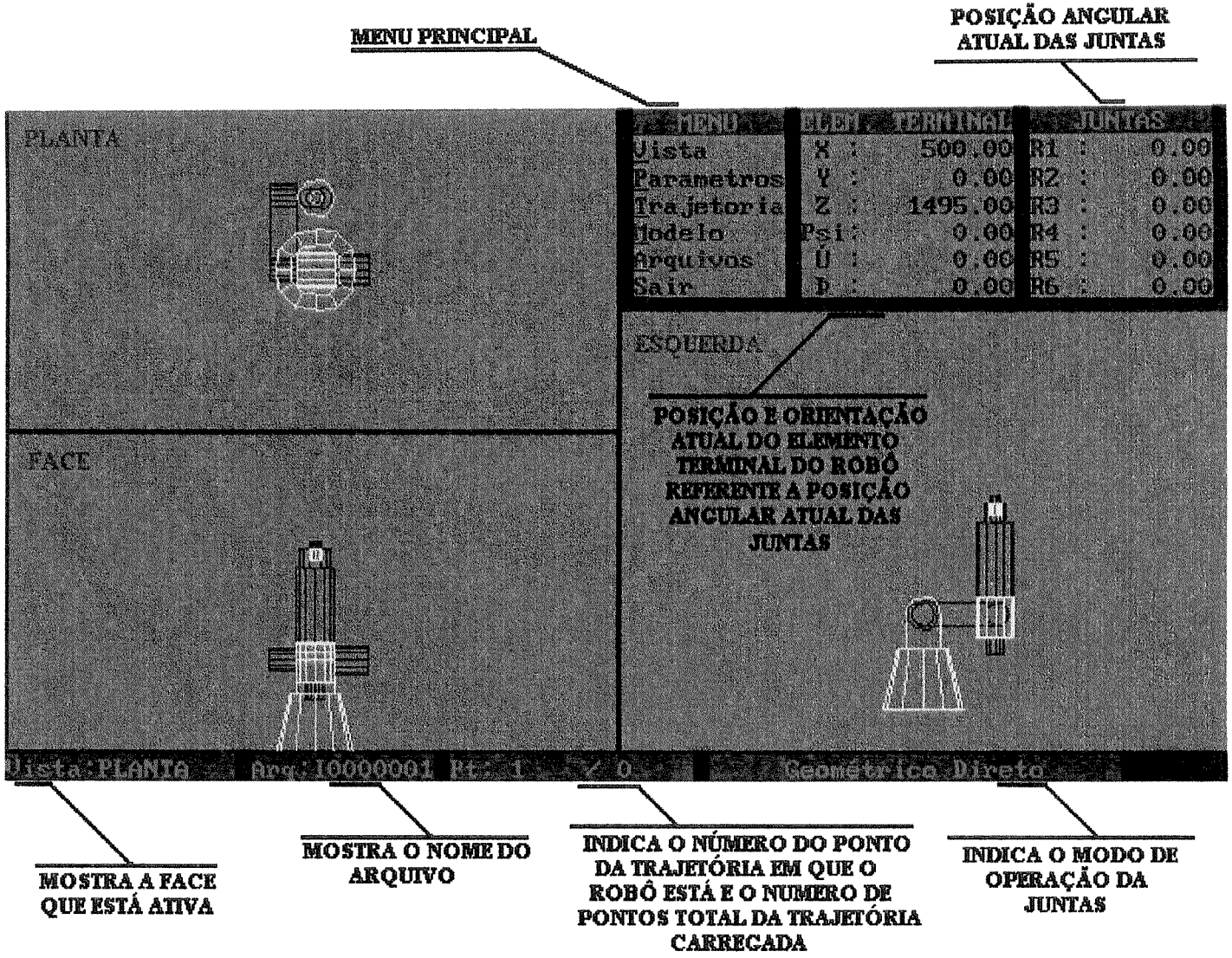


Figura VIII.2: Descrição das informações que aparecem na tela do módulo Simula.

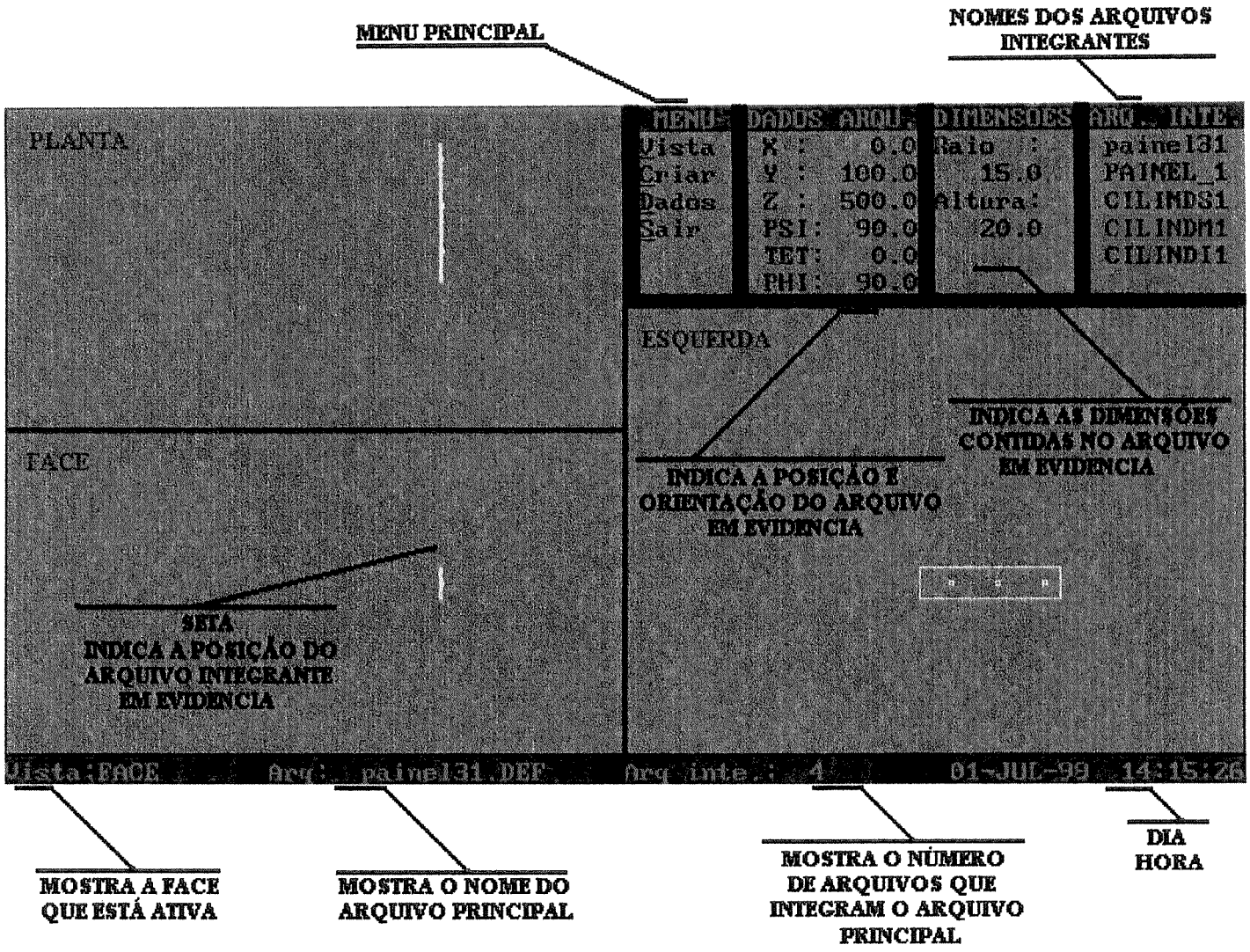


Figura VIII.3: Descrição das informações que aparecem na tela do módulo Obstáculo.

ANEXO IX

Construção de dois Painéis Utilizando o Módulo Obstáculo

São apresentadas nas simulações abaixo, as estrutura dos arquivos do painel31.def e dopainel5p.def que criam os painéis, utilizados nas simulações, para a atuação do robô.

IX.1 Simulação 1: painel com 3 furos

Abaixo são apresentados os arquivos de definições necessários para a criação de um painel com três furos.

```
-- Painel31
Compose
-- Posicao
-100.0
800.0
1000.0
-- Orientacao
90.0
0.0
90.0
-- Arquivos Integrantes
Painel_1.Def;
        - Painel_Vertical
        Pave
        -- Posicao
        0.0
        0.0
        0.0
        -- Orientacao
        0.0
        0.0
        0.0
        -- Dimensoes
        13.0
        200.0
        900.0
Cilinds1.Def;
        -- Cilindro Superior
        Cylindre
        -- Posicao
        0.0
        100.0
        800.0
        -- Orientacao
        90.0
        0.0
        90.0
        -- Dimensoes
        -- Raio
        15.0
        -- Altura
        20.0
Cilindm1.Def;
        -- Cilindro Meio
        Cylindre
        -- Posicao
        0.0
        100.0
        500.0
        -- Orientacao
        90.0
        0.0
        90.0
        -- Dimensoes
        -- Raio
        15.0
        -- Altura
        20.0
Cilindi1.Def;
        -- Cilindro Inferior
        Cylindre
        -- Posicao
        0.0
        100.0
        200.0
        -- Orientacao
        90.0
        0.0
        90.0
        -- Dimensoes
        -- Raio
        15.0
        -- Altura
        20.0
null;
null;
```

As figuras IX.1, IX.2 e IX.3 mostram, respectivamente, o painel, no módulo Obstáculo, com a orientação dada por (0.0, 0.0 e 0.0), (90.0, 0.0 e 0.0) e (90.0, 0.0 e 90.0). Os valores

indicados em cada furo referem-se ã posição espacial indicada nos arquivos referentes a cada um destes. Estas indicações não se referem à posição em relação ao referencial do robô.

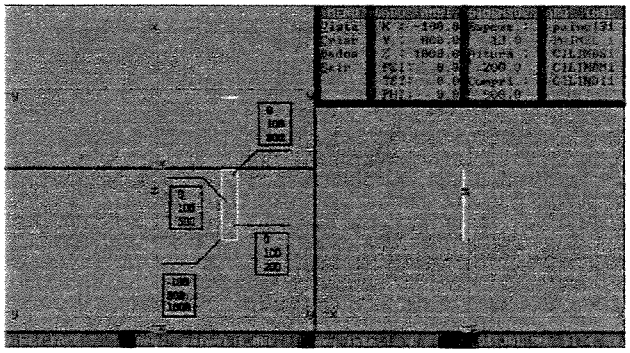


Figura IX.1: Posição do painel com a orientação 0.0, 0.0 e 0.0 no arquivo Painel31.ref.

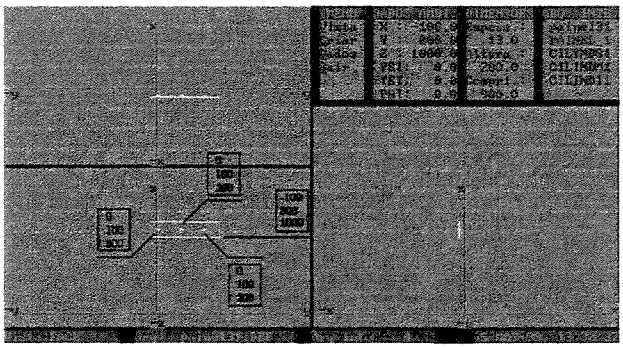


Figura IX.2: Posição do painel com a orientação 90.0, 0.0 e 0.0 no arquivo Painel31.ref.

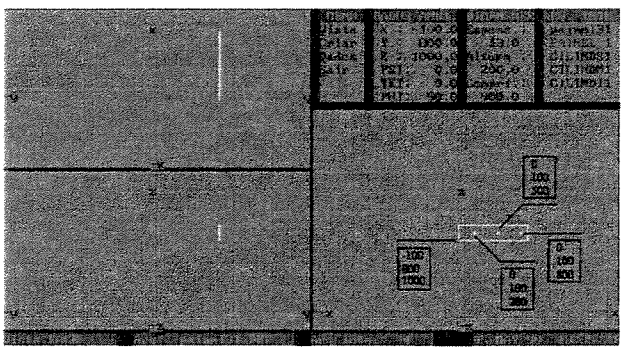


Figura IX.3: Posição do painel com a orientação 90.0, 0.0 e 90.0 no arquivo Painel31.ref.

A figura IX.4 mostra a posição espacial dos furos em relação ao referencial do robô. Estas posições podem ser obtidas a partir do módulo Obstáculo através da opção *Dados do Meio Ambiente*, no menu Dados.

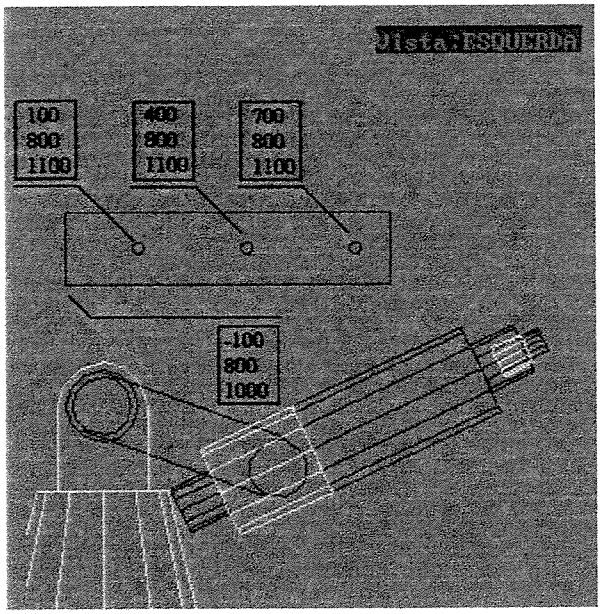


Figura IX.4: Posição espacial dos furos em relação ao referencial do robô.

IX.2 Simulação 2: painel com 5 furos

Abaixo são apresentados os arquivos de definições necessários para a criação de um painel com cinco furos.

```
-- Painel5p
Compose
-- Posicao
900.0
-190.0
600.0
-- Orientacao
0.0
0.0
0.0
-- Arquivos Integrantes
Painel_2.Def;
        -- Painel_Vertical
        Pave
        -- Posicao
        0.0
        0.0
        0.0
        -- Orientacao
        0.0
        0.0
        0.0
        -- Dimensoes
        13.0
        500.0
        1000.0
Cilind_V.Def;
        -- Cilindros Verticais
        Compose
        -- Posicao
        0.0
        0.0
        0.0
        -- Orientacao
        0.0
        0.0
        0.0
        -- Arquivos Integrantes
        Cilind_S.Def;
                -- Cilindro Superior
                Cylindre
                -- Posicao
                0.0
                250.0
                800.0
                -- Orientacao
                0.0
                90.0
                0.0
                -- Dimensoes
                -- Raio
                15.0
                -- Altura
                20.0
        Cilind_I.Def;
                -- Cilindro Inferior
                Cylindre
                -- Posicao
                0.0
                250.0
                200.0
                -- Orientacao
                0.0
                90.0
                0.0
                -- Dimensoes
                -- Raio
                15.0
                -- Altura
                20.0
        null;
        null;
        null;
Cilind_H.Def;
        -- Cilindros Horizontais
        Compose
        -- Posicao
        0.0
        0.0
        0.0
        -- Orientacao
        0.0
        0.0
        0.0
        -- Arquivos Integrantes
        Cilind_E.Def;
                -- Cilindro Esquerdo
                Cylindre
                -- Posicao
                0.0
                100.0
                500.0
                -- Orientacao
                0.0
                90.0
                0.0
                -- Dimensoes
                -- Raio
                15.0
                -- Altura
                20.0
        Cilind_M.Def;
                -- Cilindro Meio
                Cylindre
                -- Posicao
                0.0
                250.0
                500.0
                -- Orientacao
                0.0
                90.0
                0.0
                -- Dimensoes
                -- Raio
                15.0
                -- Altura
                20.0
        Cilind_D.Def;
                -- Cilindro Direito
                Cylindre
                -- Posicao
                0.0
                400.0
                500.0
                -- Orientacao
                0.0
                90.0
                0.0
                -- Dimensoes
                -- Raio
                15.0
                -- Altura
                20.0
        null;
        null;
                null;
                null;
                null;
```

A figura IX.5 mostra o painel, no módulo Obstáculo, com a orientação dada por (0.0, 0.0 e 0.0). Os valores indicados em cada furo referem-se à posição espacial indicada nos arquivos referentes a cada um destes. Estas indicações não se referem à posição em relação ao referencial do robô.

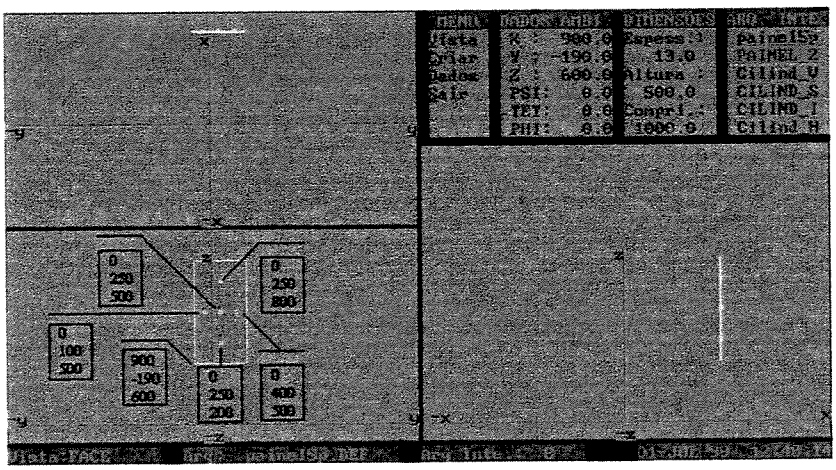


Figura IX.5: Posição do painel com a orientação 0.0, 0.0 e 0.0 no arquivo Painel5p.ref.

A figura IX.6 mostra a posição espacial dos furos em relação ao referencial do robô. Estas posições podem ser obtidas a partir do módulo Obstáculo através da opção *Dados do Meio Ambiente*, no menu Dados.

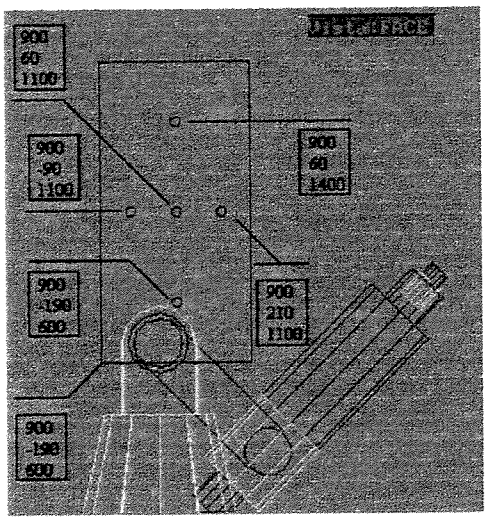


Figura IX.6: Posição espacial dos furos em relação ao referencial do robô.

ANEXO X

Apresentação dos Erros de Posição e Orientação para as Trajetórias Geradas no módulo Trajeto

Nas tabelas abaixo são apresentadas as posições (px, py e pz em mm) e orientações (psi, teta e phi em graus) finais desejadas para cada segmento de trajetória e as posições e orientações finais alcançadas, após a geração dos segmentos, pelo módulo Trajeto. Os valores em vermelho indicam que o valor desejado não foi alcançado e os valores em azul indicam que o valor desejado foi atingido.

Seg.		Final desejada	Final alcançada
1	Px	900	899.874
	Py	0	0
	Pz	1100	1100.123
	Psi	0	0
	Teta	89	88.981
	Phi	0	0
2	Px	900	899.99
	Py	0	0
	Pz	1500	1500
	Psi	0	0
	Teta	89	89
	Phi	0	0
3	Px	900	900
	Py	200	200
	Pz	1500	1500
	Psi	0	0
	Teta	89	89
	Phi	0	0
4	Px	900	900
	Py	200	200
	Pz	1100	1100
	Psi	0	0
	Teta	89	89
	Phi	0	0
5	Px	900	900
	Py	0	0
	Pz	1100	1100
	Psi	0	0
	Teta	89	89
	Phi	0	0

Tabela X.1: Posição e orientação finais desejadas e alcançadas pelos segmentos de trajetória para a Simulação 1.

Seg.		Final desejada	Final alcançada
1	Px	850	849.999
	Py	-200	-200
	Pz	1400	1400
	Psi	0	0
	Teta	89	89
	Phi	0	0
2	Px	850	849.999
	Py	200	200
	Pz	(1400)	(1400)
	Psi	0	0
	Teta	89	89
	Phi	0	0

Seg.		Final desejada	Final alcançada
3	Px	850	850
	Py	-200	-200
	Pz	(1400)	(1400)
	Psi	0	0
	Teta	89	89
	Phi	0	0

Tabela X.2: Posição e orientação finais desejadas e alcançadas pelos segmentos de trajetória para a Simulação 2.

Seg.		Final desejada	Final alcançada
1	Px	650	650
	Py	60	60
	Pz	1400	1400
	Psi	0	0
	Teta	89	89
	Phi	0	0
2	Px	900	900
	Py	60	60
	Pz	1400	1400
	Psi	0	0
	Teta	89	89
	Phi	0	0
3	Px	650	650
	Py	60	60
	Pz	1400	1400
	Psi	0	0
	Teta	89	89
	Phi	0	0
4	Px	650	649.998
	Py	60	60
	Pz	1100	1100.408
	Psi	0	0
	Teta	89	89
	Phi	0	0
5	Px	900	900
	Py	60	60
	Pz	1100	1100
	Psi	0	0
	Teta	89	89
	Phi	0	0
6	Px	650	650
	Py	60	60
	Pz	1100	1100
	Psi	0	0
	Teta	89	89
	Phi	0	0

Seg.		Final desejada	Final alcançada
14	Px	400	400
	Py	550	550
	Pz	1100	1100
	Psi	-89	-89
	Teta	0	0
	Phi	0	0
15	Px	400	400
	Py	800	800
	Pz	1100	1100
	Psi	-89	-89
	Teta	0	0
	Phi	0	0
16	Px	400	400
	Py	550	550
	Pz	1100	1100
	Psi	-89	-89
	Teta	0	0
	Phi	0	0
17	Px	700	700
	Py	550	499.998
	Pz	1100	1100
	Psi	-89	-89
	Teta	0	0
	Phi	0	0
18	Px	700	700
	Py	800	800
	Pz	1100	1100
	Psi	-89	-89
	Teta	0	0
	Phi	0	0
19	Px	700	700
	Py	550	550
	Pz	1100	1100
	Psi	-89	-89
	Teta	0	0
	Phi	0	0

Tabela X.3: Posição e orientação finais desejadas e alcançadas pelos segmentos de trajetória para a Simulação 3 (parte 1).

7	Px	650	650
	Py	60	60
	Pz	800	800
	Psi	0	0
	Teta	89	89
	Phi	0	0
8	Px	900	900
	Py	60	60
	Pz	800	800
	Psi	0	0
	Teta	89	89
	Phi	0	0
9	Px	650	650
	Py	60	60
	Pz	800	800
	Psi	0	0
	Teta	89	89
	Phi	0	0
10	Px	500	500.042
	Py	0	0.017
	Pz	1495	1494.798
	Psi	0	0
	Teta	10	10.017
	Phi	0	0
11	Px	100	99.99
	Py	550	549.999
	Pz	1100	1100
	Psi	-89	-89
	Teta	0	0
	Phi	0	0
12	Px	100	100
	Py	800	800
	Pz	1100	1100
	Psi	-89	-89
	Teta	0	0
	Phi	0	0
13	Px	100	100
	Py	550	550
	Pz	1100	1100
	Psi	-89	-89
	Teta	0	0
	Phi	0	0
20	Px	500	499.999
	Py	0	0
	Pz	1495	1495
	Psi	-10	-10
	Teta	0	0
	Phi	0	0
21	Px	650	649.982
	Py	210	209.976
	Pz	1100	1100.045
	Psi	0	-0.036
	Teta	89	88.993
	Phi	0	-0.036
22	Px	900	900
	Py	210	210
	Pz	1100	1100
	Psi	0	0
	Teta	89	89
	Phi	0	0
23	Px	650	650
	Py	210	210
	Pz	1100	1100
	Psi	0	0
	Teta	89	89
	Phi	0	0
24	Px	650	649.998
	Py	-90	-89.505
	Pz	1100	1100
	Psi	0	0
	Teta	89	89
	Phi	0	0
25	Px	900	900
	Py	-90	-90
	Pz	1100	1100
	Psi	0	0
	Teta	89	89
	Phi	0	0

Tabela X.4: Posição e orientação finais desejadas e alcançadas pelos segmentos de trajetória para a Simulação 3 (parte 2).

Seg.		Final desejada	Final alcançada	Seg.		Final desejada	Final alcançada
1	Px	800	799.999	3	Px	900	900
	Py	100	100		Py	100	100
	Pz	1360	1359.999		Pz	1360	1360
	Psi	0	0		Psi	0	0
	Teta	89	89		Teta	89	89
	Phi	0	0		Phi	-45	-45
2	Px	900	900				
	Py	100	100				
	Pz	1360	1360				
	Psi	0	0				
	Teta	89	89				
	Phi	0	0				

Tabela X.5: Posição e orientação finais desejadas e alcançadas pelos segmentos de trajetória para a Simulação 4.

ANEXO XI

Descrição dos Tipos de Arquivos Existentes

Os nomes dos arquivos possuem o formato ********.*** (oito letras e a extensão). Existem cinco tipos de extensões de arquivos: .REF, .ESP, .DEF, .OBS e .LST.

XI.1 Arquivos com extensão .REF

Estes arquivos contêm as posições angulares das juntas do robô em estudo. Um exemplo, da forma deste arquivo é apresentado abaixo.

R1	R2	R3	R4	R5	R6
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
-0.0573	0.0149	0.0496	0.0560	-0.0193	0.0560
-0.1170	0.0297	0.0993	-38.7493	-0.0386	38.9755
-0.1766	0.0444	0.1493	-34.5783	-0.0620	34.9190

Os arquivos que possuem o formato ********.**ref** possuem todos os pontos gerados no módulo *Trajeto* sendo, deste modo, os arquivos **originais.**

Os arquivos que possuem o nome no formato ******sm.ref** (simula) possuem os pontos gerados a partir do arquivo **original**, diferenciando apenas no número de pontos, que é menor. Estes arquivos são utilizados no módulo *Simula* para a simulação gráfica do movimento do robô em estudo, uma vez que não há necessidade de simular todos os pontos do arquivo **original.**

Os arquivos que possuem o nome no formato ******fl.ref (filtrado) possuem os pontos gerados a partir da interpolação e filtragem de pontos de passagem. Este arquivo é gerado no módulo *Trajeto*.

XI.2 Arquivos com extensão .ESP

Os arquivos que contêm o nome no formato ********.esp possuem as posições espaciais do elemento terminal do robô em estudo para cada conjunto de posição angular das juntas do arquivo ********.ref. Este arquivo é gerado no módulo *Trajeto*. Este arquivo também pode ser criado a partir dos arquivos de ******sm.ref e ******fl.ref. Abaixo é mostrado a maneira como estes arquivos se apresentam.

X	Y	Z	-	-	-
500.0000	0.0000	1495.0000	0.0000	0.0000	0.0000
500.9004	-0.5010	1494.8695	0.0000	0.0000	0.0000
501.8152	-0.9826	1494.7388	0.0000	0.0000	0.0000
502.7145	-1.4881	1494.6080	0.0000	0.0000	0.0000

XI.3 Arquivos com extensão .LST

Os arquivos que contêm o nome no formato ********.lst possuem informações dos arquivos gerados.

XI.4 Arquivos com extensão .DEF

Estes arquivos possuem o formato ********.def e são chamados de arquivos de definição. São utilizados apenas pelo módulo Obstáculo.

Possuem as posições, orientações e medidas dos elementos primitivos (esfera, cilindro e paralelepípedo) que compõem o objeto (obstáculo, painel, ferramenta, etc.) a serem criadas.

XI.5 Arquivos com extensão .OBS

Possuem o formato ********.obs. Estes arquivos, escritos na forma binária, são gerados a partir dos arquivos **.def** pelo módulo Obstáculo; podem ser lidos apenas pelo módulo Simula.

A tabela XI.1 mostra detalhadamente os tipos de arquivos que podem ser lidos (L) e gerados (G) por cada módulo. Apresenta também uma breve descrição e em que formatos se apresentam.

		MÓDULOS				
ARQUIVOS	L/G	Trajeto	Obstáculo	Simula	DESCRIÇÃO	FORMATO
********.ref	L	sim	não	sim	contém as posições angulares das juntas obtidas através do modelo cinemático inverso (número total de pontos)	ascii
	G	sim	não	sim		
******sm.ref	L	sim	não	sim	contém as posições angulares das juntas obtidas através do modelo cinemático inverso (número reduzido de pontos)	ascii
	G	sim	não	não		
*******tl.ref	L	sim	não	sim	contém as posições angulares das juntas obtidas através de interpolação e filtragem	ascii
	G	sim	não	não		
********.esp	L	sim	não	não	contém as posições espaciais do elemento terminal para cada posição angular	ascii
	G	sim	não	não		
********.def	L	não	sim	não	possuem as posições, orientações e medidas dos elementos primitivos que compõem o ambiente a ser criado	ascii
********.obs	L	não	não	sim	possuem as posições, orientações e medidas dos elementos primitivos que compõem o ambiente a ser criado	binário
	G	não	sim	não		

Tabela XI.1: Tipos de arquivos que podem ser lidos (L) e gerados (G) por cada módulo.

XI.6 Cabeçalho

Todos os arquivos com extensão **.ref** e **.esp** possuem o cabeçalho abaixo:

```
---   TRAJETORIA    ---
          ==========
nome da trajetoria       : pppppppp
trajetoria precedente    : PXYZ0000
correcao posicionamento  : NULO____
modulo de atuacao        : NULO____
painel                   : NULO____
arquivo obstaculos       : NULO____
nome da ferramenta       : NULO____
base movel               : NULO____


--  PONTOS  ) --
================================
```

Os nomes dos arquivos contidos neste cabeçalho são carregados apenas no módulo Simula, deste modo, todo ambiente de trabalho do robô (módulos, painéis, ferramentas, obstáculos e base móvel) é carregado, quando o arquivo da trajetória (.ref) é lido.

O nome "NULO____" significa que não há arquivo a ser carregado para o dispositivo em questão.

Os nomes dos arquivos deste cabeçalho podem ser editados em um editor comum ou a partir do módulo Simula.

Anexo XII

Outros recursos disponíveis no pacote de supervisão e controle - CMK

A seguir serão descritos outros recursos disponíveis no pacote de supervisão e controle.

XII.1 Animação

Esta opção, encontrada a partir da opção Microcomputador, apresenta as mesmas opções do controle sem afetar o manipulador, figura XII.1.

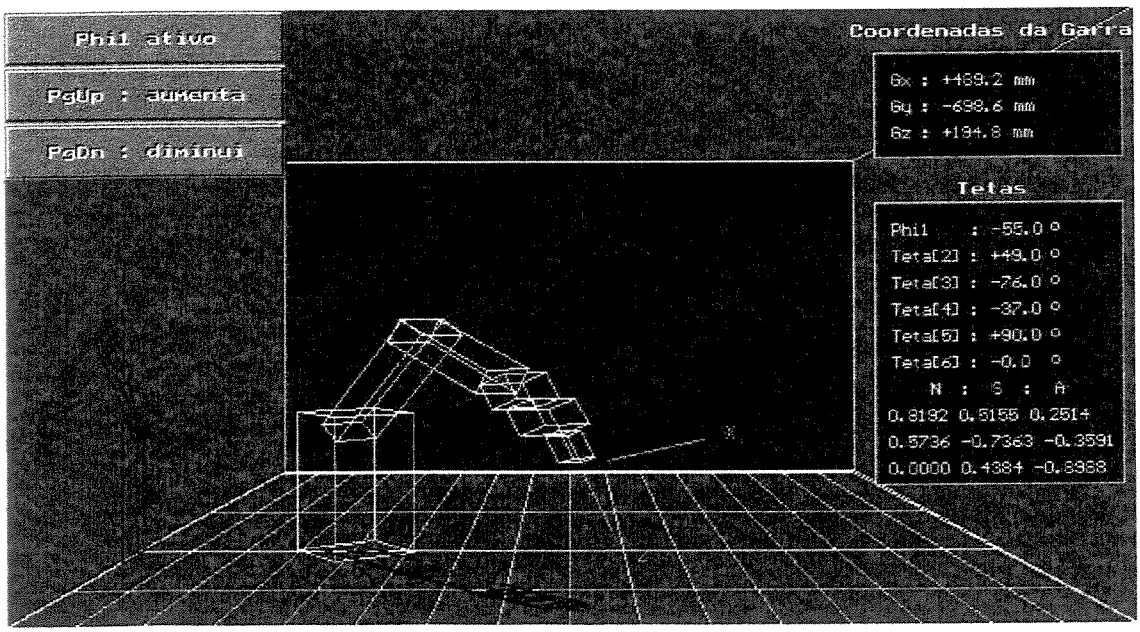


Figura XII.1: Tela do modo Animação.

XII.2 Parâmetros

Esta opção, disponível a partir da opção Microcomputador, permite a mudança nos valores tidos como "default" e de alguns parâmetros do "mouse" espacial, figura XII.2, e do teclado, figura XII.3.

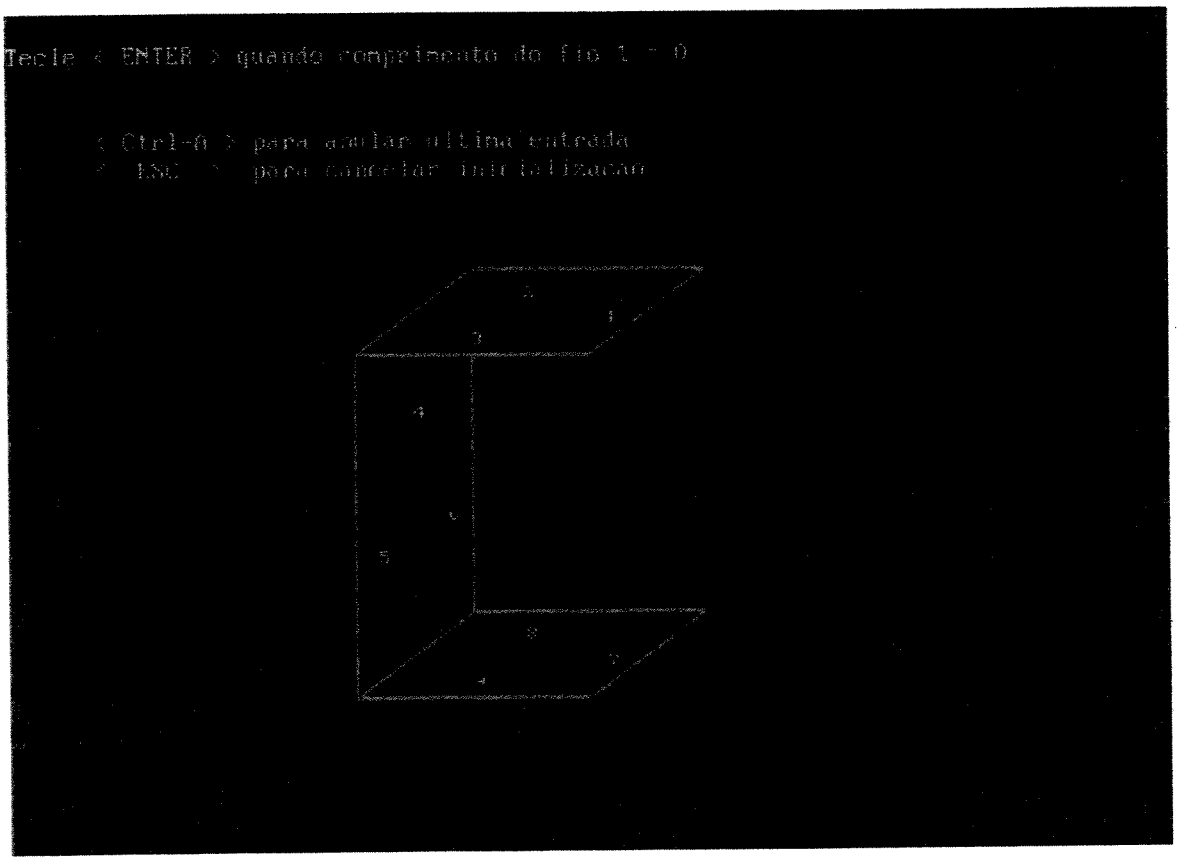


Figura XII.2: Tela de alteração de parâmetros para o "mouse" espacial.

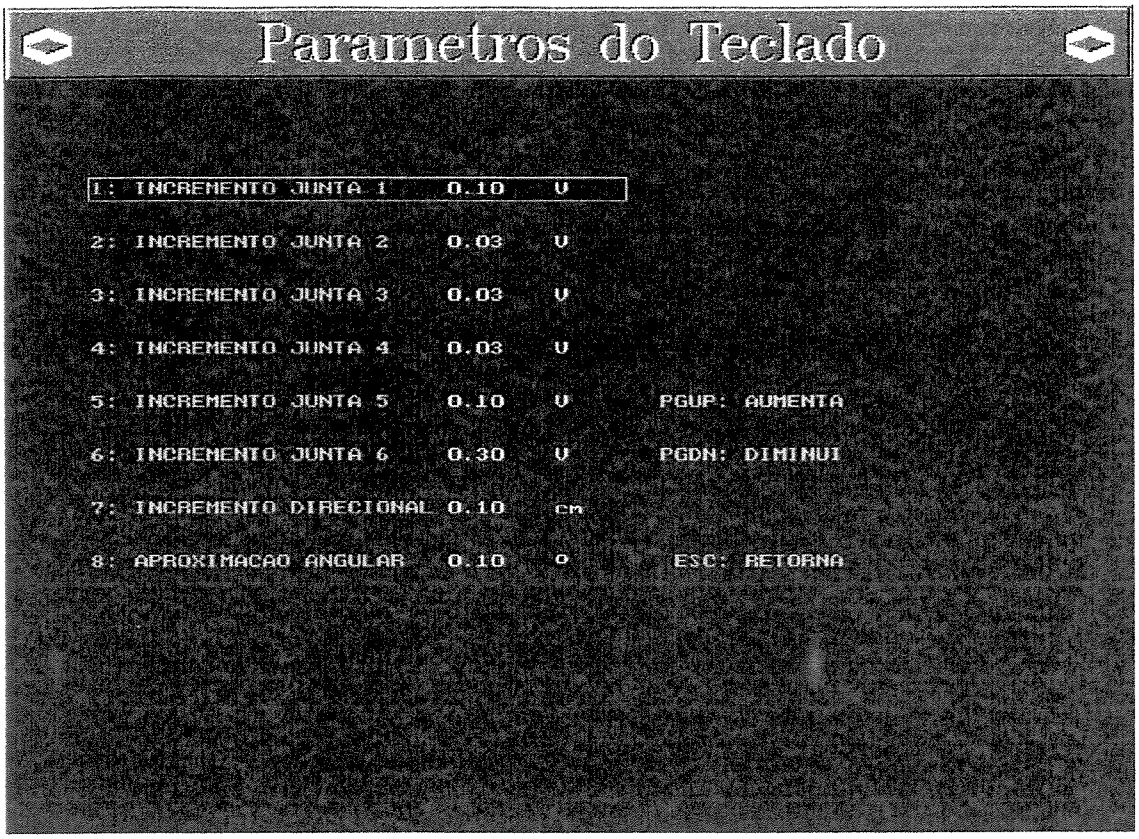


Figura XII.3: Tela de alteração de parâmetros para o teclado.

XII.3 Teste de "hardware"

Esta opção, encontrada a partir da opção Teste de "hardware" do menu principal, permite testar elementos do "hardware" utilizado, tais como conversores A/D e D/A, entradas e saídas digitais, modos de operação e porta serial de conexão do "mouse" espacial. A tela correspondente é apresentada na figura XII.4.

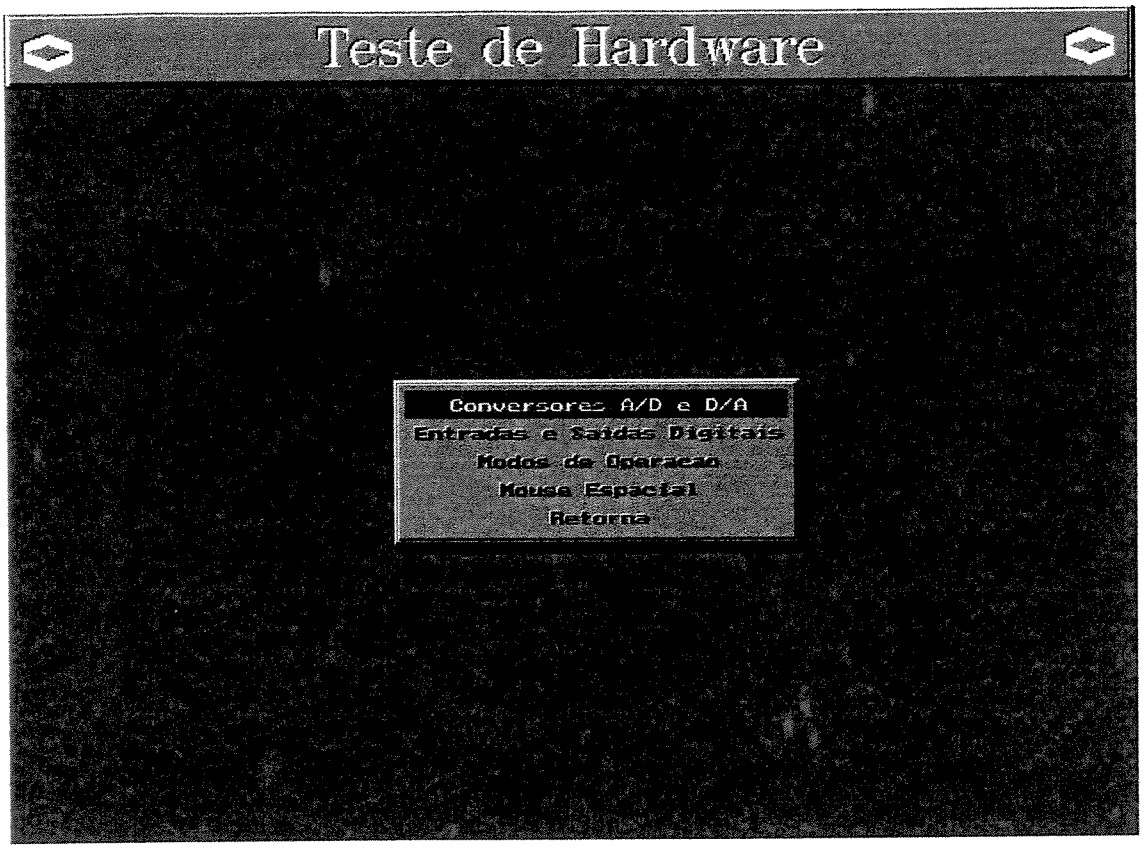


Figura XII: Tela do modo Teste de "hardware".

XII.3.1 Funcionamento

Inicialmente o PC é colocado no modo Teste de Hardware, então aparece uma mensagem para que as placas de conversão A/D e D/A sejam interligadas. A saída da placa D/A, de conector DB-37 é ligada à entrada (também de conector DB-37) da placa A/D, e inicía-se um processo de geração/conversão/comparação, figura XII.5.

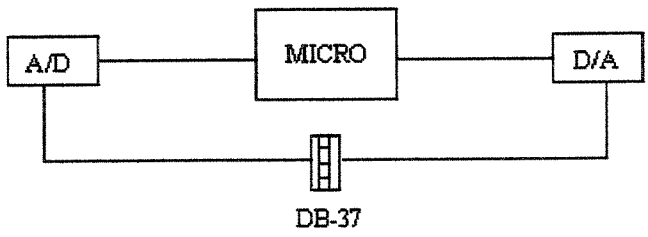


Figura XII.5: Ligação do sistema para o teste das placas A/D e D/ª

O microcomputador envia sinais na forma digital para a placa D/A que os converte para a forma analógica e que por sua vez, através da placa A/D são reconvertidos para digital e enviados de volta ao microcomputador, que faz a comparação entre os valores enviados e recebidos e emite uma tabela de erros para cada canal medido. Se um desses valores de erro ultrapassa um determinado valor de tolerância, a conversão e considerada, incorreta e uma mensagem para que seja feita uma verificação nas placas é enviada. As entradas e'saídas digitais podem ser testadas de forma semelhante.

Os conectores foram montados de forma a permitir a execução desse teste, que por ser um módulo, permanente no sistema, se caracteriza como uma inestimável ajuda ao operador na detecção e eliminação de possíveis problemas de mal-funcionamento.

As características das placas A/D e D/A utilizadas são apresentadas nas tabelas XII.1 e XII.2 respectivamente.

Canais de entrada analógica	Simples	16
	Diferenciais multiplexadas	8
Canais de saída analógica		até 4
Canais de entrada digital		16
Canais de saída digital		16
Resolução do conversor A/D		12 bits (4096 níveis)
Contadores programáveis/Temporizadores		3 (16 bits)
Oscilador a crista		1 2 MIlz
A placa conta ainda com as seguintes características: Sequência de leitura e ganhos programáveis através de memória de cariais; Aquisição em "Burst" propiciada por buffer (FIFO) de 16 posições; Suporte para DMA (Direct Alemory Access: Acesso Direto à Memória), permitindo a velocidade máxima de coleta de sinais independente da UCP do microcomputador; Possibilidade de, com o auxilio de circuitos externos, realizar amostragem simultânea de até 16 canais; Suporte a interrupções.		

Tabela XII.1: Características da placa A/D utilizada.

Canais de saída analógica		8
Faixas de saída configuráveis	Bipolares	± 10,0 v
		+ 550 v
		± 2,5 V
	Unipolares	0 a 10,0 v
		0 a 5,0 v
Mínima carga de saída		10 Kohm
Resolução		12 bits (4096 níveis)
Tipos de representação decimal		complemento de dois
		binário com off-set
Tipos de atualização das saídas		individual por canal
		simultânea comandada pela UCP
		simultânea comandada pelo controlador
Tempo de conversão		4 µs (máximo)
Contadores programáveis		3
Oscilador a cristal		2 MHz
Endereçamento ajustável		de 0200li a 03F8h
Espaço de endereçamento		8 posições
Interrupções selecionáveis		IRQ2, IR.Q3. IRQ4, IRQ5

Tabela XII.2: Características da placa D/A utilizada.

XII.4 Calibração e monitoramento

Estas opções, encontradas a partir da opção "Master-slave" do menu principal, oferecem as seguintes facilidades:

- Calibração: efetua a correspondência entre os valores máximos e mínimos dos ângulos de juntas do "master" e do "slave", figura XII.6.

- Monitoramento: monitora as tensões dos potenciômetros do master-slave. Apresenta também os valores dos ângulos das juntas e a posição e orientação da garra, obtidos via modelo cinemático direto, figura XII.7.

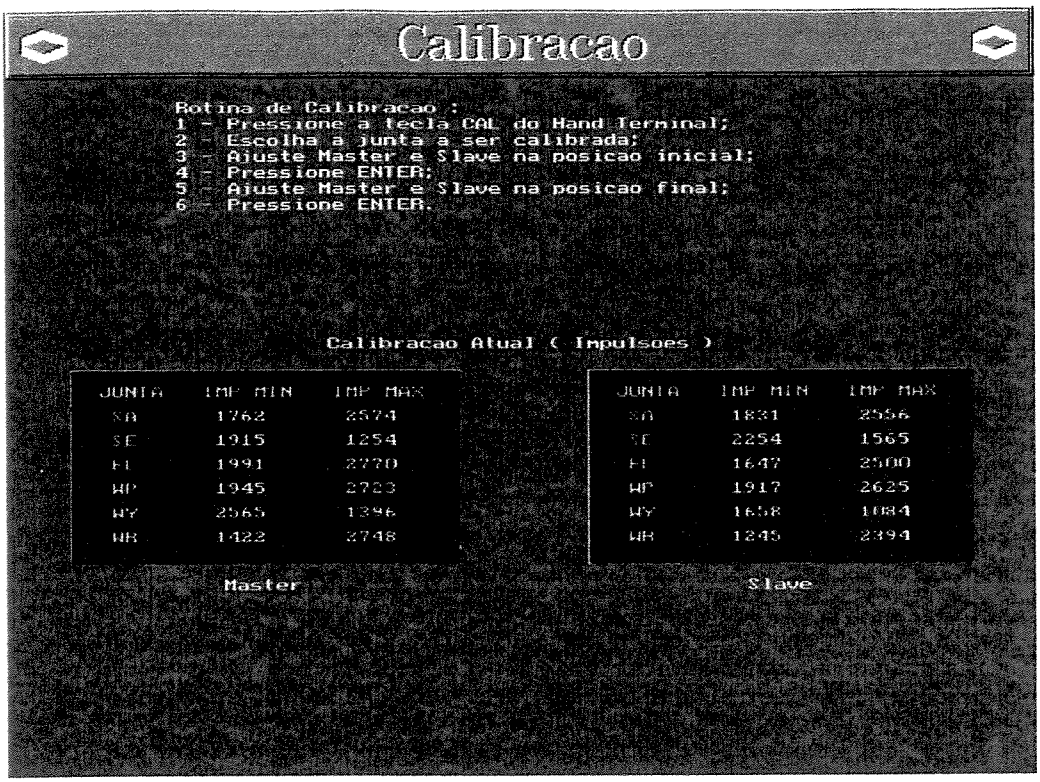


Figura XII.6: Tela do modo Calibração.

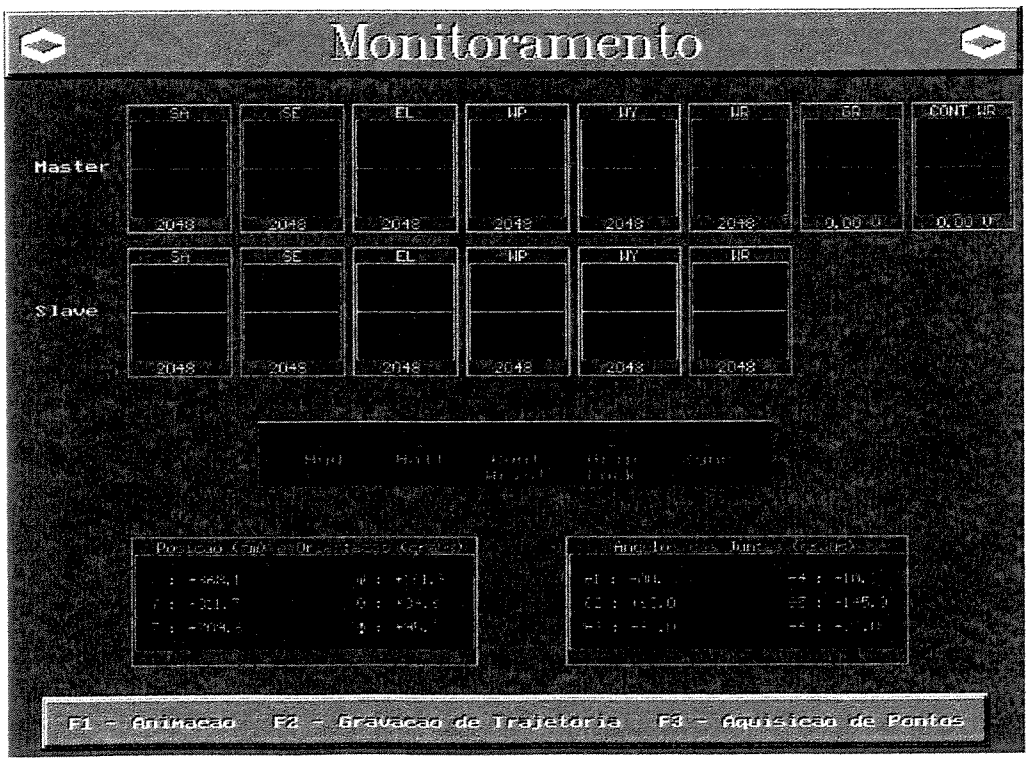


Figura XII.7: Tela do modo Monitoramento.

XII.5 Outras telas disponíveis

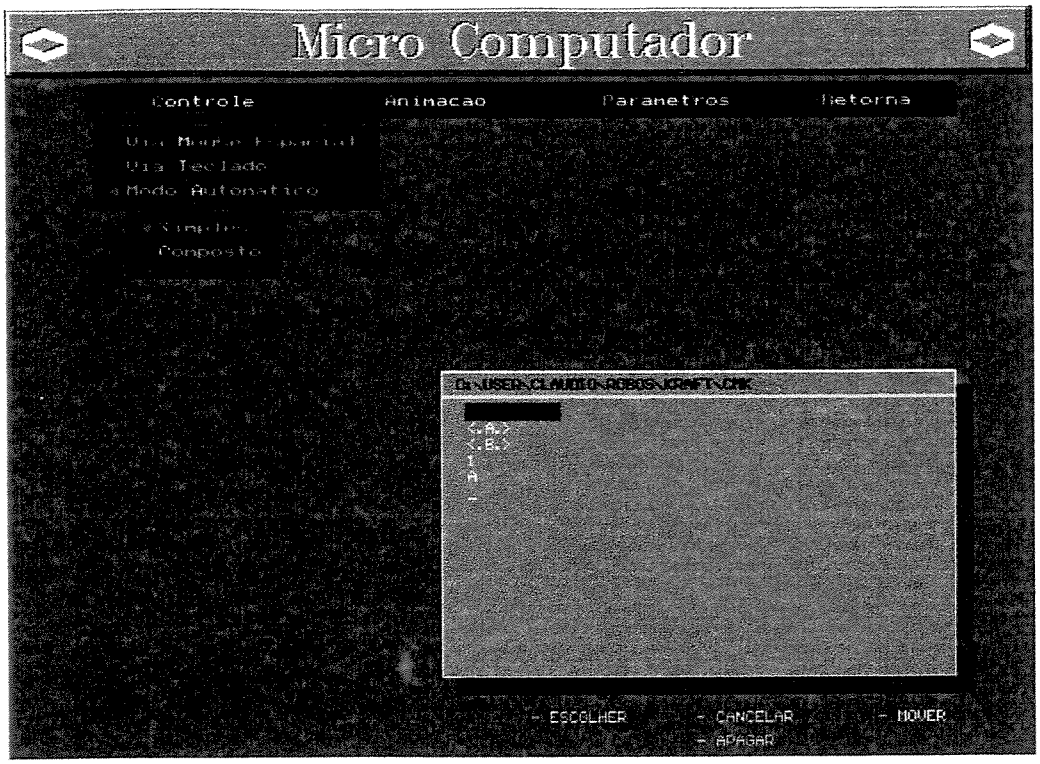


Figura XII.8: Tela de envio de trajetórias simples do modo Automático

Figura XII.9: Tela de envio de trajetórias compostas do modo Automático.