

UNIVERSIDADE FEDERAL DE PERNAMBUCO
CURSO DE PÓS-GRADUAÇÃO EM ENGENHARIA MECÂNICA

Simulação de Movimento e Planejamento de Trajetória para Robôs Manipuladores

Dissertação submetida à Universidade Federal de Pernambuco para a obtenção
do grau de mestre em Engenharia Mecânica

Erwin Rommel Ferreira Costa

Orientador: Félix Christian Guimarães Santos
Co-orientadora: Noemia Gomes de Mattos de Mesquita

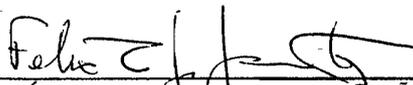
Recife, Dezembro de 2003

“SIMULAÇÃO DO MOVIMENTO E PLANEJAMENTO DA TRAJETÓRIA PARA
ROBÔS MANIPULADORES”.

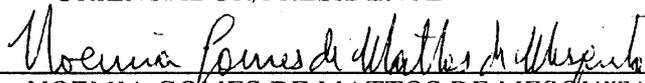
ERWIN ROMMEL FERREIRA COSTA

ESTA DISSERTAÇÃO FOI JULGADA ADEQUADA PARA OBTENÇÃO DO
TÍTULO DE MESTRE EM ENGENHARIA MECÂNICA

ÁREA DE CONCENTRAÇÃO: MECÂNICA COMPUTACIONAL
APROVADA EM SUA FORMA FINAL PELO
CURSO DE PÓS-GRADUAÇÃO EM ENGENHARIA MECÂNICA/CTG/EPP/UFPE



Prof. Dr. FÉLIX CHRISTIAN GUIMARÃES SANTOS
ORIENTADOR/PRESIDENTE

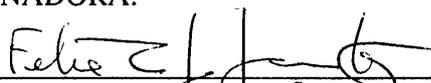


Profa. Dra. NOEMIA GOMES DE MATTOS DE MESQUITA
CO-ORIENTADORA

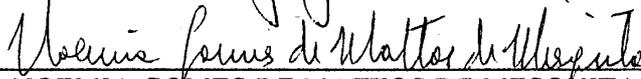


Profa. Dra. RITA DE CÁSSIA FERNANDES DE LIMA
COORDENADORA DO CURSO

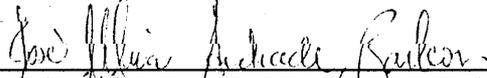
BANCA EXAMINADORA:



Prof. Dr. FÉLIX CHRISTIAN GUIMARÃES SANTOS (UFPE)



Profa. Dra. NOEMIA GOMES DE MATTOS DE MESQUITA (UFPE)



Prof. Dr. JOSÉ MARIA ANDRADE BARBOSA (UFPE)



Prof. Dr. ANNA HELENA REALI COSTA (USP)

Esta dissertação de mestrado é dedicada à minha família. Família esta que eu sei que sempre poderei contar com ela e que sempre estiveram e estarão ao meu lado. Por estas razões não poderia deixar de cita-los: meus tios Rita, Leca, Tonho, Maza e Nana; a prima Flávia; os meus “irmãozinhos” Pammela e Phillip; o sobrinho Thales; a minha irmã Michelli; meu pai João e minha mãe Beta.

AGRADECIMENTOS

Inicialmente agradeço ao Departamento de Engenharia Mecânica da Universidade Federal de Pernambuco por todos esses anos de convivência e ao SENAI – PE por ter aceito financiar uma bolsa de pesquisa par o desenvolvimento deste trabalho, em segundo lugar, mas tão importantes quanto o primeiro, agradeço ao meu paciente orientador prof. Félix Christian e a minha co-orientadora prof^a Noemia Mesquita. Outros que deram também uma colaboração importantíssima neste trabalho foram os alunos Hugo Nunes e Jarbas José, graduandos do curso de Engenharia Mecânica, companheiros neste trabalho por dois anos.

Parte dos resultados finais obtidos deve-se ao fato de termos construído um pequeno robô manipulador chamado *Goblin*, projetado e construído por nós nas dependências do Centro de Formação Profissional Manoel de Brito com a assistência de alguns dos seus instrutores. Faz-se aqui também, o reconhecimento à ajuda recebida dos amigos Paulo Xaxá e Leonardo Resende do Departamento de Eletrônica e Sistemas (DES) da UFPE.

Após terem sido feitos os agradecimentos às pessoas e instituições que tiveram algum envolvimento direto na realização desta dissertação, agradecerei agora aos que indiretamente influenciaram na mesma. Cronologicamente falando, em primeiro lugar agradeço aos professores Ramiro Brito e Paulo Lyra pelo incentivo e apoio dados para que eu tomasse a decisão em fazer o mestrado. Agradeço à prof^a Rita Lima, coordenadora da pós – graduação do DEMEC nos anos de 2002 e 2003, por sua contagiante disposição em solucionar os problemas extra–dissertação; e também à nossa secretária do mestrado, D^a Eliane Alves, que nos proporciona um suporte fantástico.

Finalizando os agradecimentos, não poderia deixar de agradecer aos companheiros e companheiras de mestrado, hoje alguns já mestres e outros futuros mestres, pelos inúmeros momentos bons e inesquecíveis que passamos juntos.

RESUMO

Este trabalho considera robôs manipuladores que empregam motores de passo como força motriz em cada junta. Tendo como dado uma trajetória ideal, o primeiro problema é aproximar esta trajetória por uma outra que seja capaz de ser executada pelo robô. Assim, uma métrica deve ser definida para que a trajetória escolhida seja ótima. O segundo problema é associado à programação do robô. Neste sentido, tendo em vista a trajetória aproximada a ser percorrida, uma estratégia de movimento é escolhida e o programa é gerado. O terceiro problema é associado à simulação do robô de tal forma que todos os componentes eletrônicos ou não, envolvidos no movimento do robô devem ser considerados. Com tal complexidade é possível obter um nível de realismo bastante acentuado, que vai desde a trajetória exata até o movimento do robô. A saída desta simulação pode ser visual, de tal forma que se possa observar o nível de concordância entre a trajetória desejada (aproximada) e a trajetória simulada (considerada como a executada pelo robô). Exemplos teóricos são analisados e um exemplo prático de um braço com três barras e três motores é apresentado.

ABSTRACT

This work considers manipulator robots that use step motor as driving device in each joint. The first problem comprises in approximating a given ideal trajectory by another one, which could be executed by the robot. Therefore a metric has to be defined in order to define an optimality criteria for trajectories. The second problem is associated to the robot programming. In this sense and having in mind the chosen approximated trajectory, a movement strategy is chosen and the program is automatically generated. The third problem is related to the robot simulation in such a way that all its components, either electronic or not, which are involved in the robot movement, should be considered. By considering such a complex system it becomes possible to obtain a high level of realism in the simulation, which goes from the exact trajectory up to the robot movement itself. The simulation output can be presented in a graphic form in such a way that the level of agreement between the ideal and the approximate trajectories could be observed. Theoretical examples are analysed and a practical one consisting of a robot with three bars and three motors is presented.

SUMÁRIO

1. Introdução	01
2. Robótica	03
3. Revisão bibliográfica	17
4. Modelagem matemática	19
5. Programa (1ª parte)	24
6. Componentes mecânicos e eletrônicos	28
7. Programa (2ª parte)	30
8. Planejamento de trajetória e controle de velocidade	33
9. Programa (3ª parte)	41
10. Aplicação	48
11. Trabalhos futuros	51
12. Conclusão	52
Referências bibliográficas	53
Anexos	55

1. INTRODUÇÃO

Há cerca de dois anos e meio atrás o SENAI-PE decidiu adquirir alguns Controladores Lógicos Programáveis (CLP) que são uma espécie de microcomputadores próprios para serem utilizados em linhas de produção industrial. A partir disso firmou-se um convênio com o Departamento de Engenharia Mecânica (DEMEC) da Universidade Federal de Pernambuco (UFPE) para que fosse desenvolvido projeto de pesquisa que envolvesse a área de automação industrial. A tarefa logo de início apresentou-se bastante promissora em termos científicos, pois até então ainda não existiam projetos dessa natureza no DEMEC.

A idéia apresentada ao SENAI e que foi aceita foi construir e controlar via microcomputador um protótipo de manipulador industrial. A tarefa seguinte então foi planejar as etapas necessárias para atingir este objetivo. Primeiro teve-se que aprender como o computador poderia se comunicar com o exterior, ou seja, como se poderia controlar equipamentos através do computador, esta etapa foi vencida graças a colaboração de alguns colegas do Departamento de Eletrônica e Sistemas (DES) da UFPE.

As próximas etapas eram construir o manipulador e adquirir um circuito eletrônico (placa) para fazer a interação entre computador e robô. Estas etapas foram vencidas quase que simultaneamente, após algumas idas e vindas.

Finalmente restava o programa de controle e simulação do movimento, que foi a parte principal do trabalho, e é este programa desenvolvido em linguagem C++ que será apresentado neste trabalho de dissertação.

Na elaboração do programa a nossa primeira preocupação foi com a representação computacional do robô, em outras palavras, como poderíamos fazer para o computador se relacionar com cada componente de um robô. Entre os componentes temos: juntas, eixos, motores, entre outros componentes. Para a representação já existe uma metodologia tradicional que é utilizar transformação homogênea e calcular determinados parâmetros. Aplicando os conceitos citados e criando-se em C++ os objetos adequados, se pode representar quaisquer configurações.

Após conseguir representar de forma adequada a configuração do manipulador tínhamos que incrementar o programa para que se pudesse simular o comportamento de um manipulador real, assim foram criadas outras classes que faziam parte do sistema, tais como placa e porta paralela.

Finalmente vinha a etapa de otimizar a trajetória, pois robôs industriais são utilizados para executar tarefas e na execução destas tarefas ele necessita realizar certas trajetórias, porém existe o problema da resolução, o posicionamento não é contínuo, mas discreto. Além disso, para determinadas trajetórias é requerido que os movimentos das diversas articulações seja sincronizado. Isso requer um planejamento adequado tanto do posicionamento de cada articulação do manipulador para que este possa atingir determinado ponto, como também da trajetória que o manipulador terá de executar para mover-se de sua atual posição até alcançar o referido ponto.

A dissertação contém doze capítulos, sendo o primeiro esta introdução. O capítulo dois trata da robótica como ciência e suas principais aplicações, bem como alguns dos principais conceitos; esse capítulo tem como finalidade familiarizar o leitor com estes conceitos e com os termos utilizados em robótica. O capítulo três é dedicado à revisão bibliográfica de métodos existentes para o planejamento da trajetória e o controle da velocidade de manipuladores. O quarto capítulo apresenta a modelagem matemática utilizada na simulação cinemática. O capítulo cinco mostra e discute as primeiras classes do programa desenvolvido.

O capítulo seis apresenta os principais componentes mecânicos e eletrônicos que compõem um manipulador como o desenvolvido neste trabalho. O capítulo seguinte apresenta as classes criadas para simular o comportamento destes componentes.

Os métodos utilizados para o planejamento da melhor trajetória e o algoritmo para sincronizar o movimento das articulações são descritos no capítulo oito e as implementações dos métodos mostram-se no capítulo nove.

O capítulo dez faz um resumo de como o programa funciona como um todo, um fluxograma de funcionamento é mostrado e discutido.

Os capítulos 11 e 12 são dedicados respectivamente a sugestões para trabalhos futuros que podem vir a serem desenvolvidos tendo este trabalho como fonte inicial de consulta e, as conclusões finais que resumem os resultados obtidos e aqui apresentadas.

Foram acrescentados à dissertação dois anexos: o primeiro apresenta dois elementos eletrônicos utilizados neste trabalho, este anexo, assim como foi o segundo capítulo, é voltado principalmente aqueles que estão dando os primeiros passos em termos de robótica e automação; e o segundo anexo contém um artigo apresentado em congresso e publicado em revista. Este artigo foi escrito um ano antes da submissão desta dissertação à avaliação.

2. ROBÓTICA

2.1 Automação e Robótica

Automação e robótica são duas tecnologias intimamente relacionadas. Num contexto industrial podemos definir a automação como uma tecnologia que se ocupa do uso de sistemas e processos mecânicos, sistemas eletrônicos e computadores na operação e controle de alguma atividade. Exemplos dessa tecnologia incluem máquinas de montagem mecanizadas, sistemas de controle de realimentação (aplicados a processos industriais), máquinas ferramentas dotadas de comandos numéricos e robôs. Conseqüentemente, a robótica é uma forma de automação industrial (Groover, 1988).

A definição oficial de um robô é dada pela Associação das Indústrias de Robótica (RIA):

Um robô industrial é um manipulador reprogramável e multifuncional, projetado para mover materiais, peças, ferramentas ou dispositivos especiais em movimentos variáveis programados para a realização de uma variedade de tarefas.

2.2 Robótica na Ficção Científica

Desconsiderando as limitações das máquinas robóticas atuais, o conceito popular de um robô é o de que ele se parece e age como um ser humano. Esse conceito humanóide foi inspirado e encorajado por inúmeras histórias de ficção científica.

Uma peça tcheca do início dos anos 20 do século passado, de autoria de *Karel Capek*, deu origem ao termo robô. A palavra tcheca *robota* significa servidão ou trabalhador forçado e, quando traduzido para o inglês, transformou-se em *robot*. A história diz respeito a um cientista brilhante que fabrica robôs. O plano dele é que os robôs servirão à humanidade obedientemente e farão todo o trabalho físico. O plano tem um rumo amargo quando os robôs começam a não gostar de seu papel subserviente e rebelam-se contra seus senhores.

Entre os escritores de ficção científica, Isaac Asimov contribuiu com inúmeras histórias sobre robôs, a partir de 1939 (Groover, 1988).

O cinema contribuiu, e ainda contribui, fortemente para a disseminação destas idéias sobre robótica. Alguns filmes descrevem os robôs como servos e companheiros amigáveis, e outros os mostrando de modos diversos.

- O filme *O dia em que a terra parou*, de 1951, mostrava uma missão oriunda de um planeta distante, enviada a Terra num disco voador para estabelecer o alicerce para a paz entre as nações do mundo (figura 2.1).

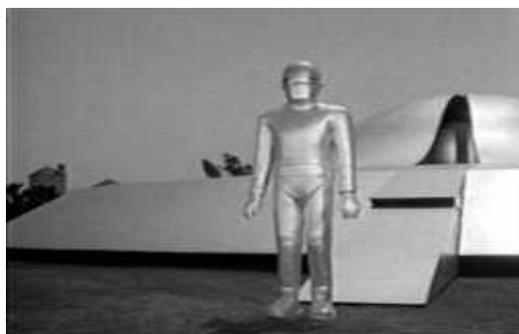


Figura 2.1 - Cena do filme *O dia em que a terra parou* (Classics movies, 2003)

- A série *Guerra nas Estrelas* mostrava robôs amistosos e inofensivos. Os robôs são capazes de andar, são inteligentes e podem comunicar-se com seus senhores humanos. Eles mostram as oportunidades oferecidas pela robótica e outras tecnologias avançadas de serem úteis e não ameaçadoras para o homem (figura 2.2).



Figura 2.2 - Cena do filme *Guerra nas estrelas* (Edson Jedi 2003)

- Recentemente, em 1999, *O Homem Bicentenário* trouxe de volta as histórias de Asimov. O filme apresenta um robô que trabalha para uma família durante gerações, sempre de forma leal e ordeira, mas o maior desejo da “máquina” é um dia se tornar humano (figura 2.3).



Figura 2.3 - Cena do filme *O homem bicentenário* (Adorocinema, 2003)

2.3 Robótica Industrial

A robótica industrial com grandes investimentos e também com equipes de técnicos trabalhando nesta área, teve início em 1956 com *Devol* e *Joseph Engelberger* que se associaram e fundaram a *Unimation Inc.* Os estudos em robótica coordenados pela empresa iniciaram em quinze empresas automotivas e em mais outras vinte indústrias de diversas categorias. A princípio se propôs projetar e construir robôs que executassem tarefas simples, porém pesadas e desagradáveis, realizadas no ambiente das fábricas.

Em 1961 o primeiro protótipo, que havia sido apresentado ao público em 1959, foi instalado em uma fábrica da *General Motors* (figura 2.4).

O desenvolvimento da robótica, iniciado ainda na década de 1950, deve-se em muito ao surgimento de três tecnologias durante a *Segunda Guerra Mundial* e aos seus aprimoramentos posteriores:

- Teoria de Servo Mecanismo
- Computação Digital
- Eletrônica de Estado Sólido

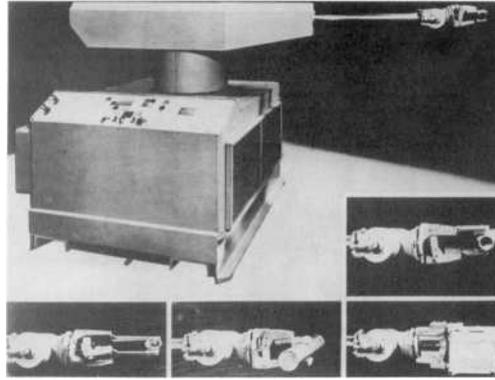


Figura 2.4 – Primeiro robô industrial instalado. (Engelberger, 1999)

O Japão, principal fabricante e consumidor de robôs no mundo, teve sua primeira empresa no setor de robótica criada em 1968. Já em 1971 foi criada no Japão a primeira associação de empresas de robótica do mundo, a *JIRA (Japan Industrial Robot Association)*, que possuía 46 companhias membros.

As grandes empresas de robótica no mundo são (Engelberger, 1999):

Nos Estados Unidos:

- General Motors.
- General Electric
- Westinghouse
- IBM
- United Technologies

Na Europa:

- G.E.C – Inglaterra
- Siemens – Alemanha
- Renault – França
- Fiat – Itália

Além dessas gigantes, existem também as “pequenas” empresas que ficam com certa fatia do mercado, sobretudo na área de robôs especializados, robôs periféricos (auxiliares) e projeto de sistemas robóticos.

Outras grandes empresas de robótica no mundo são (Engelberger, 1999):

- ABB Robotics
- Yashawa Electric Corporation
- Kawasaki Heavy Industries
- Fanuc

2.3.1 Aplicações Industriais

Os robôs possuem inúmeras aplicações na indústria, realizando tarefas que a princípio apresentam grande dificuldade, falta de segurança ou rotineira repetibilidade. Principais aplicações:

- Abastecimento de máquinas
- Solda ponto à ponto
- Pintura
- Montagem
- Acabamento (lixar, polir, retificar, etc.)

2.4 Fundamentos da Tecnologia de Robôs

A robótica é uma ciência da engenharia aplicada que é tida como uma combinação da tecnologia de máquinas ferramentas e ciência da computação, inclui campos aparentemente tão distintos quanto projeto de máquinas, teoria de controle, microeletrônica, programação de computadores, inteligência artificial, fatores humanos e teoria da produção. Pesquisa e desenvolvimento estão prosseguindo em todas essas áreas para aperfeiçoar o modo pelo qual os robôs trabalham e pensam. É provável que os esforços de pesquisa resultem em futuros robôs, que farão com que as máquinas de hoje pareçam bem primitivas. Avanços na tecnologia ampliarão o âmbito das aplicações industriais de robôs.

Para descrever a tecnologia de um robô, devemos definir uma variedade de características técnicas sobre a maneira pela qual o robô é construído e o modo como opera.

2.4.2. Anatomia do Robô

A anatomia do robô refere-se à construção física do corpo, do braço e punho da máquina. A maioria dos robôs usados em fábricas é montada sobre uma base fixada ao piso. O corpo está ligado à base e o braço ao corpo. Na extremidade do braço está o punho, ligado ao punho do robô está uma mão. O termo técnico para a mão é órgão terminal, que não é considerado como parte da anatomia do robô. O conjunto formado pela base, braço e punho é por vezes chamado de manipulador (Groover, 1988).

Movimentos relativos entre os diversos componentes do corpo, braço e punho são proporcionados por uma série de juntas. As juntas do braço e corpo do manipulador são usadas para posicionar o órgão terminal, e as juntas do punho são usadas para orientar o órgão terminal.

2.4.2 Configurações comuns de Robôs

A maioria dos robôs comercialmente disponíveis na atualidade possui uma das seguintes configurações básicas (figura 2.5):

1. Configuração cartesiana
2. Configuração cilíndrica
3. Configuração esférica
4. Configuração tipo SCARA
5. Configuração antropomórfica

Há vantagens e desvantagens relativas às anatomias básicas dos robôs, simplesmente devido a suas geometrias. Em termos de repetibilidade de movimento, que é a capacidade de mover-se para um mesmo ponto no espaço por diversas vezes com erro mínimo, o robô cartesiano tem a vantagem devido à sua estrutura inerentemente rígida. Em termos de alcance (a capacidade do robô de estender seu braço significativamente além de sua base), as configurações esférica e antropomórfica levam vantagem. A capacidade de levantamento de carga do robô é importante em muitas aplicações. O robô de configuração cilíndrica e o robô xyz de pórtico (cartesiano) podem ser projetados para alta rigidez e capacidade de carga. Para aplicações de carregamento de máquina, a capacidade do robô de penetrar numa pequena abertura, sem interferência com os lados da abertura, é importante. A configuração esférica e a configuração cilíndrica possuem uma vantagem geométrica natural em termos dessa capacidade.

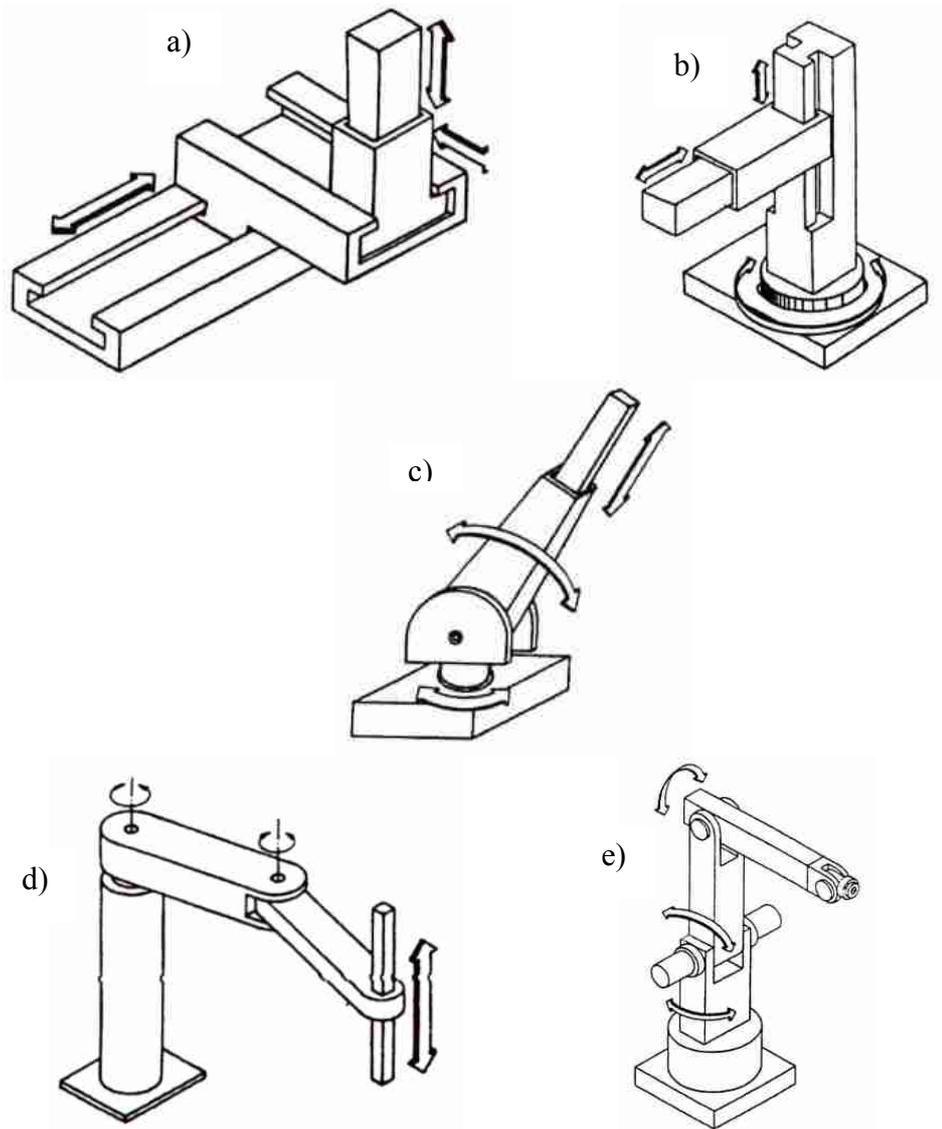


Figura 2.5 – Típicas configurações de manipuladores: a) cartesiana, b) cilíndrica, c) esférica, d) SCARA e e) antropomórfica. (Ferreira, 1998)

Seguem imagens de algumas configurações, figuras 2.6, 2.7 e 2.8.



Figura 2.6 Robô com configuração cilíndrica. (Warnecke, 1999)



Figura 2.7 - Robô de configuração antropomórfica (Warnecke, 1999).



Figura 2.8 - Configuração SCARA
(Warnecke, 1999)

2.4.3 Movimentos do Robô

Os robôs industriais são projetados para realizar trabalho produtivo, sendo esse trabalho desenvolvido através de movimentos.

Os movimentos dos robôs podem ser divididos em duas categorias gerais: movimentos do braço e do corpo e movimentos de punho. A descrição dos movimentos das articulações associada a essas duas categorias é denominada pelo termo *graus de liberdade*, e um robô industrial típico está equipado com 4 a 6 graus de liberdade. Porém existem os casos especiais. Observando as figuras 2.9 e 2.10, percebe-se que o primeiro robô é um manipulador antropomórfico com punho de três graus de liberdade colocado sobre trilhos, o que lhe confere ter 7 (sete) graus de liberdade. Na figura 2.10 ver-se um robô usado para lavar aeronaves que possui 11 (onze) graus de liberdade (Sciavicco, 2001).



Figura 2.9 - Robô com 7 graus de liberdade
(Warnecke, 1999)



Figura 2.10 - Robô com 11 graus de liberdade
(Warnecke, 1999)

Três juntas estão normalmente associadas com a ação do braço e corpo, e uma, duas ou até três juntas são normalmente usadas para acionar o punho. Conectando as diversas juntas do manipulador encontram-se membros rígidos, que são chamados eixos. Em qualquer cadeia eixo-junta-eixo, chamaremos o eixo que está mais próximo da base na cadeia de eixo suporte. O eixo movido (ou de saída) é o eixo que se move em relação ao eixo suporte.

As juntas usadas no projeto de robôs industriais envolvem tipicamente um movimento relativo dos eixos adjacentes, que é linear ou rotacional. As juntas lineares envolvem um movimento deslizante ou translacional dos eixos de conexão. Esse movimento pode ser alcançado de diversos modos (por exemplo, por um pistão hidráulico ou um sistema sem-fim-coroa). O termo junta prismática (figura 2.11) é comumente usado na literatura em lugar de

junta linear. As juntas rotativas (figura 2.12) geram, evidentemente, um movimento de rotação entre os eixos do manipulador.

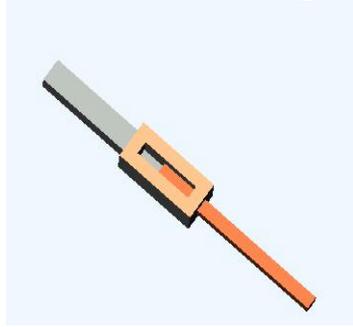


Figura 2.11 - Junta prismática

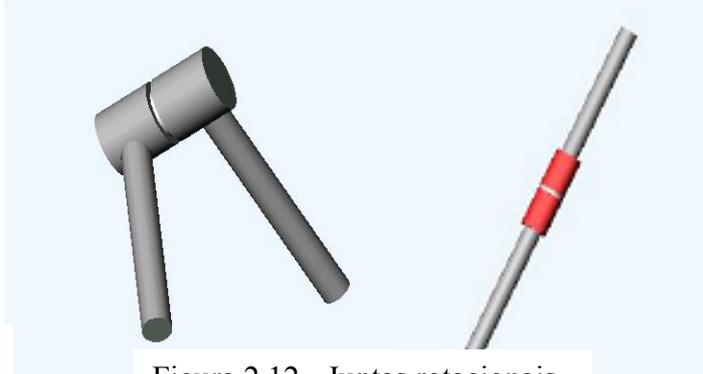


Figura 2.12 - Juntas rotacionais

As juntas de braço e corpo são projetadas para permitir que o robô mova seu órgão terminal para uma posição desejada dentro dos limites do tamanho e movimentos das juntas.

O movimento do punho é projetado para permitir que o robô oriente o órgão terminal adequadamente em relação à tarefa a ser realizada. Por exemplo, a mão tem de estar orientada no ângulo adequado em relação à peça de trabalho, a fim de agarrá-la. Para solucionar esse problema de orientação, o punho é normalmente dotado de até três graus de liberdade. Uma configuração típica é o *punho esférico*, como mostrado na figura 2.13.

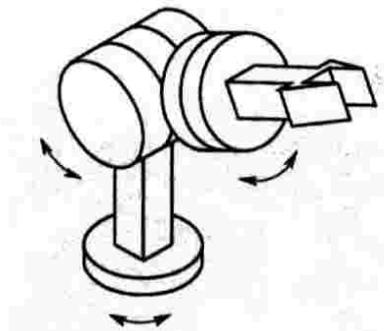


Figura 1.13 – Punho esférico. (Sciavicco; 2001)

2.4.4 Volume de Trabalho

O volume de trabalho é o termo que se refere ao espaço dentro do qual o robô pode manipular a extremidade de seu punho. A convenção de usar a extremidade do punho para definir o volume de trabalho do robô é adotada para evitar a complicação de diferentes tamanhos de órgãos terminais que possam ser fixados ao punho do robô.

O volume de trabalho é determinado pelas seguintes características físicas do robô: a configuração física do robô; o tamanho dos componentes de corpo, braço e punho; os limites dos movimentos das juntas do robô.

A influência da configuração física sobre a forma do volume de trabalho é ilustrada na figura 2.14.

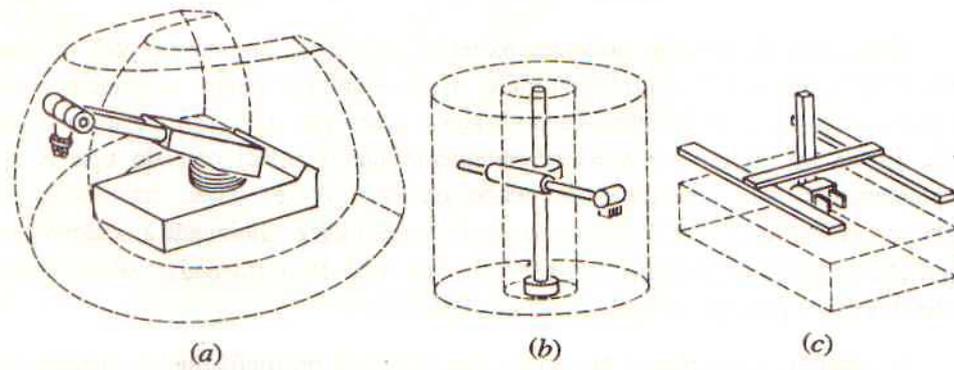


Figura 2.14 - Volume de trabalho das configurações: a) Esférica. b) Cilíndrica. c) Cartesiana. (Groover, 1988)

2.4.5 Sistemas de Acionamento do Robô

A capacidade do robô de mover seu corpo, braço e punho é propiciada pelo sistema usado para acionar o robô. O sistema de acionamento determina a velocidade e precisão dos movimentos do braço, a força do robô e seu desempenho dinâmico.

Robôs industriais são acionados por um dos três tipos de sistemas de acionamento, que são: Acionamento elétrico; Acionamento hidráulico; e Acionamento pneumático.

Os robôs de acionamento elétrico são acionados por motores DC (motores de passo ou servomotores) e motores AC. Esses motores são adequados para o acionamento de juntas rotacionais através de sistemas de eixos e engrenagens, mas podem também ser utilizados para acionar juntas lineares por sistemas de polia ou outros meios translacionais (Groover, 1988).

Os sistemas de acionamento hidráulico podem ser projetados para acionar juntas rotacionais ou lineares. Acionadores de pás podem ser utilizados para suprir movimento rotacional, e os pistões hidráulicos podem ser usados para realizar movimento linear.

O acionamento pneumático é geralmente reservado para robôs de menor porte que possuem um menor número de graus de liberdade. Esses robôs são comumente utilizados em simples operações, do tipo “pega-e-põe” com ciclos rápidos.

Segue quadro com as principais vantagens e desvantagens de cada um dos sistemas de acionamento (quadro 2.1).

Quadro 2.1: Principais vantagens / desvantagens dos sistemas para acionamento de robôs.

SISTEMA DE ACIONAMENTO	VANTAGENS	DESADVANTAGENS
Elétrico	<ul style="list-style-type: none"> › Boa precisão e repetibilidade. › Requer pouco espaço útil no chão › Manutenção simples e fácil. 	<ul style="list-style-type: none"> • Baixa relação potência/peso. • Incapaz de manter um alto torque por muito tempo.
Hidráulico	<ul style="list-style-type: none"> › Altas velocidades. › Grandes forças 	<ul style="list-style-type: none"> • Requer grandes espaços. • Tendência para vazar óleo.
Pneumático	<ul style="list-style-type: none"> › Altas velocidades › Custo relativamente baixo › Manutenção fácil 	<ul style="list-style-type: none"> • Baixa precisão. • Podem surgir vibrações momentâneas quando o sistema for acionado

2.4.6 Especificações de Desempenho

Serão apresentadas agora algumas definições importantes, utilizadas para se mensurar o desempenho de robôs ao realizar movimentos (Groover, 1988; Ferreira, 1998; Gong, 2000).

1. Resolução espacial
2. Precisão
3. Repetibilidade
4. Carga útil

2.4.6.1 Resolução Espacial

A resolução espacial de um robô é o menor incremento de movimento em que o robô pode dividir seu volume de trabalho. A resolução espacial depende de dois fatores: a resolução de controle do sistema e as resoluções mecânicas/eletrônicas do robô.

A resolução do sistema de controle determina o valor mínimo que pode ter o incremento, ou seja, determina o número máximo de pontos alcançáveis (ou pontos endereçáveis). A capacidade de dividir o deslocamento total da junta em incrementos depende da capacidade de armazenamento na memória de controle. Quando se utilizam microcomputadores essa capacidade de armazenamento não é problema, pois as memórias são bem grandes; mas ao se utilizar microcontroladores deve-se levar essa resolução em consideração.

A resolução mecânica/eletrônica refere-se aos elementos que compõem a estrutura física do robô. A resolução mecânica refere-se ao grau de precisão máxima de cada componente mecânico, por exemplo, a resolução dos motores de passo; e a resolução elétrica (eletrônica) refere-se à máxima precisão dos elementos puramente eletrônicos, por exemplo, a frequência de funcionamento dos sensores.

2.4.6.2 Precisão

A precisão refere-se à capacidade de um robô de posicionar a extremidade de seu punho num ponto-meta desejado dentro do volume de trabalho e pode ser definida em termos de resolução espacial, porque a capacidade de atingir um dado ponto-meta depende de quão proximamente o robô pode definir os incrementos de controle para cada um dos movimentos de suas juntas. No pior caso, o ponto desejado se situaria entre dois incrementos de controle adjacentes, então poderíamos inicialmente definir precisão, nessa hipótese do pior caso, como metade da resolução de controle.

Um outro fator que interfere na precisão são as imperfeições mecânicas nos eixos e componentes de junta do robô. Estas imperfeições resultam de flexão elástica nos elementos estruturais, folga nas engrenagens, estiramento de polia, vazamentos de fluidos hidráulicos e outras falhas no sistema mecânico. As imperfeições são também influenciadas por fatores tais como a carga que está sendo manipulada, a velocidade com a qual o braço está se movendo, a condição de manutenção do robô, dentre outros.

2.4.6.3 Repetibilidade

Repetibilidade diz respeito à capacidade do robô posicionar, por diversas vezes, seu punho ou um órgão terminal num ponto no espaço previamente indicado.

A repetibilidade e a precisão referem-se a dois diferentes aspectos de desempenho do robô. Supondo que queremos atingir um determinado ponto, o ponto programado provavelmente será diferente do ponto-meta, devido a limitações da resolução de controle. A repetibilidade refere-se à capacidade do robô de retornar ao ponto programado quando comandado.

A figura 2.15 representa os pontos alcançados por um manipulador que tem como ponto meta o alvo no centro da figura, será utilizada para representar a diferença entre precisão e repetibilidade. Da esquerda para a direita têm-se: baixa precisão e baixa repetibilidade, em seguida baixa precisão e alta repetibilidade e finalmente, alta precisão e alta repetibilidade.



Figura 2.15 - Comparando precisão X repetibilidade (Kao; 1997)

No espaço tridimensional os erros de repetibilidade circundarão o ponto programado, formando uma distribuição cujo limite externo pode ser conceituado como uma esfera. Um fabricante de robô cita tipicamente a repetibilidade de seu manipulador como o raio da esfera idealizada, geralmente exprimindo a especificação como mais ou menos um valor particular (Sciavicco, 2001).

2.4.6.4 Carga Útil

A carga útil é a máxima carga que um robô pode manipular, e deve ser medida em sua posição de máxima sollicitação e com a máxima velocidade, por exemplo, na configuração cilíndrica ou esférica, a medida é feita com o braço completamente alongado.

Essa carga máxima refere-se ao peso total da ferramenta ou garra adicionado à carga propriamente dita.

2.5. Órgãos Terminais e Sensores

2.5.1 Órgãos Terminais

Em robótica o termo órgão terminal é usado para descrever a mão ou ferramenta que está conectada ao punho. O órgão terminal representa o ferramental especial que permite ao robô de finalidades gerais realizar uma aplicação particular (Warnecke, 1999).

Órgãos terminais podem ser divididos em duas categorias: garras e ferramentas. Garras seriam utilizadas para capturar um objeto, usualmente uma peça de trabalho, e retê-la durante o ciclo de trabalho do robô. Existe uma variedade de métodos de retenção que podem ser usados, além dos meios mecânicos óbvios de sujeitar a peça entre dois ou mais dedos, têm-se o uso de ventosas de sucção, ímãs, ganchos e conchas. Ferramentas são usadas como órgão terminal nas aplicações em que seja requerido que o robô realize algum tipo de trabalho na peça em produção. Essas aplicações incluem soldagem a arco, pintura a pistola, furação e outros. Em cada caso, a ferramenta particular é ligada ao punho do robô.

2.5.2 Sensores Robóticos

Os robôs têm a capacidade de localizar-se no espaço de trabalho, identificar sua direção e intensidade dos movimentos, através dos sensores, que funcionam exatamente como os *sentidos* do robô (Warnecke, 1999).

Os sensores usados como dispositivos periféricos na robótica incluem tanto tipos simples, tais como chaves-limite, quanto tipos sofisticados, tais como sistemas de visão de máquina. Evidentemente os sensores são usados como componentes integrais do sistema de controle de realimentação do robô. Sua função como dispositivos periféricos numa célula de trabalho robótica é a de permitir que as atividades do robô sejam coordenadas com outras atividades na célula. Os sensores usados na robótica incluem as seguintes categorias gerais:

1. Sensores de distância. Um sensor de distância é um dispositivo que indica quando um objeto está próximo.
2. Sensores de tato. São sensores que respondem a força de contato com outro objeto. Alguns desses dispositivos são capazes de medir o nível de força envolvido.

3. Visão de máquina. Um sistema de visão de máquina é capaz de visualizar o espaço de trabalho e interpretar o que vê. Esses sistemas são usados na robótica para realizar inspeção, reconhecimento de peças e outras tarefas similares.
4. Tipos diversos. A categoria *diversos* inclui os tipos restantes de sensores usados na robótica. Incluem sensores de temperatura, pressão e outras variáveis.

Os sistemas que possuem sensores são chamados de sistema de malha fechada, pois o sinal que sai do controlador passa para o acionador que repassa à junta, e esta após se mover, ou quando em movimento, interage com o sensor que manda os dados recolhidos com a junta de volta ao controlador (figura 2.16). No sistema de malha aberta, sem sensores, o deslocamento das juntas não é acompanhado ou conferido assim o sistema supõe que o movimento ocorreu sem problemas e parte para o comando seguinte (figura 2.17).

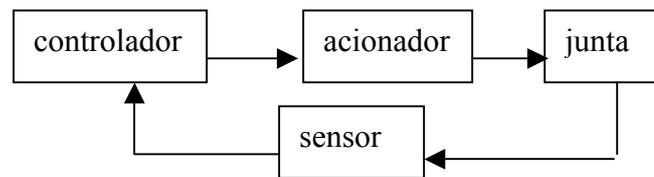


Figura 2.16 - Sistema em malha fechada

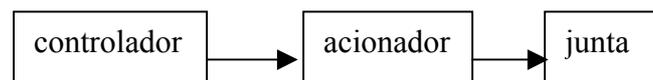


Figura 2.17 - Sistema em malha aberta

2.6 Controle e Programação

2.6.1 Sistemas de Controle e Desempenho Dinâmico

Para operar um robô deve se ter meios de controlar seu sistema de acionamento a fim de regular adequadamente seus movimentos. De acordo com seus sistemas de controle, robôs industriais podem ser classificados em quatro categorias (Groover, 1988):

1. Robôs de seqüência fixa
2. Robôs de repetição com controle ponto a ponto
3. Robôs de repetição com controle de trajetória contínua
4. Robôs inteligentes

Das quatro categorias, os robôs de seqüência fixa representam o mais baixo nível de controle, e os robôs inteligentes são os mais sofisticados.

Robôs de seqüência fixa são controlados mediante instalação de chaves-limite e/ou batentes mecânicos para estabelecer os pontos extremos de deslocamento para cada uma de suas juntas. Com esse método de controle, as juntas somente podem ser movidas para seus limites extremos de deslocamento, o que restringe seriamente o número de pontos distintos que podem ser especificados para esses robôs. Qualquer um dos três sistemas de acionamento pode ser usado com esse tipo de sistema de controle, porém o sistema pneumático é o mais comumente empregado. Aplicações para esse tipo de robô geralmente envolvem movimentos simples, tais como operações “pega-e-põe” (Groover, 1988).

Os robôs de repetição usam uma unidade de controle mais sofisticada, em que uma série de posições ou movimentos é “instruída” ao robô, gravada na memória e, em seguida, repetida

pelo robô sob seu próprio controle. O procedimento de instruir e gravar na memória é denominado programação do robô. Os robôs de repetição geralmente têm alguma forma de servocontrole (isto é, sistema de realimentação em malha fechada) para garantir que as posições que foram alcançadas pelo robô sejam as que foram instruídas.

Os robôs de repetição podem ser classificados em duas categorias: robôs ponto a ponto (PTP) e robôs de trajetória contínua (CP). Os robôs ponto a ponto podem executar ciclos de movimento que consistem em uma série de localizações de ponto desejadas. O robô é instruído sobre cada ponto, e esses pontos são registrados na unidade de controle do robô (figura 2.18). Durante a repetição, o robô é controlado para mover-se de um ponto para outro na seqüência adequada. Robôs ponto a ponto não controlam a trajetória para irem de um ponto até o próximo. Se o programador quiser exercer uma quantidade de controle limitada sobre a trajetória seguida, isto terá de ser feito mediante programação de uma série de pontos ao longo da trajetória desejada. O controle da seqüência das posições é bastante adequado para muitos tipos de aplicações, incluindo máquinas de carregar e descarregar e soldagem a ponto (Pazos, 2002; Freitas, 2002).



Figura 2.18 - Unidade de controle (Freitas, 2002)

Os robôs de trajetória contínua podem realizar ciclos de movimento em que a trajetória seguida pelo robô é controlada. Isto é geralmente realizado fazendo-se com que o robô se desloque através de uma série de pontos intimamente espaçados que descrevem a trajetória desejada. Os pontos individuais são definidos pela unidade de controle e não pelo programador. O movimento linear é uma forma comum de controle por trajetórias contínuas para robôs industriais. O programador especifica o ponto de partida e o ponto final da trajetória, e a unidade de controle calcula a seqüência de pontos individuais que permitem ao robô seguir uma trajetória retilínea. Alguns robôs têm a capacidade de seguir um caminho suave, curvo, que foi definido por um programador que movimentava manualmente o braço através do ciclo de movimento desejado. Atingir um controle por trajetória contínua requer que a unidade controladora seja capaz de armazenar um grande número de localizações de pontos individuais que definem a trajetória curva. O controle por trajetórias contínuas é requerido para certos tipos de aplicações industriais, tais como pintura e soldagem a arco.

Robôs inteligentes constituem uma classe crescente de robôs industriais que possuem capacidade não apenas de repetir um ciclo de movimento programado, mas também interagir com seu ambiente. Invariavelmente a unidade controladora consiste em um computador digital ou dispositivo similar (por exemplo, controlador programável). Os robôs inteligentes podem alterar seu ciclo programado em resposta a condições que ocorrem no local de trabalho. Podem tomar decisões lógicas com base nos dados sensoriais recebidos do local de trabalho. Os tipos de aplicações que são realizadas por robôs inteligentes baseiam-se no uso de uma linguagem de alto nível para realizar as atividades complexas e sofisticadas que podem ser realizadas por esses robôs. Aplicações típicas para robôs inteligentes são operações de montagem e operações de soldagem a arco.

2.7 Programação do Robô

Em sua forma mais básica um programa de robô pode ser definido como uma trajetória no espaço através da qual o manipulador é comandado a mover-se. Essa trajetória também inclui outras ações, tais como controle do órgão terminal e recebimento de sinais dos sensores. A finalidade da programação dos robôs é instruí-los essas ações.

Existem vários métodos usados para programar robôs. As duas categorias básicas de maior importância comercial atualmente são a programação por aprendizagem e a programação por linguagem textual.

A programação por aprendizagem consiste em levar o braço do robô a mover-se na seqüência de movimentos requeridos e registrar os movimentos na memória do controlador. Os métodos por aprendizagem são utilizados para programar robôs de repetição. Utiliza-se uma caixa de controle (chamada “teach-in-box”, figura 2.19) para acionar as juntas dos robôs a cada um dos pontos desejados no espaço de trabalho e registrar os pontos na memória para subsequente repetição.



Figura 2.19 - Exemplos de “Teach-in-Box” (Freitas, 2002)

Em caso de robôs com trajetória contínua, ou se registra dois pontos e o controlador calcula os demais para seguir uma linha reta entre os pontos (figura 2.20), ou se a trajetória for mais complexa (pintura, por exemplo), movimenta-se fisicamente o braço do robô através da trajetória que vai sendo memorizada pelo controlador (figura 2.21). Devido à facilidade, conveniência e à ampla gama de aplicações adequadas ao mesmo, o método de aprendizagem é o método de programação mais comum para robôs de repetição.

Os métodos de programação textual empregam uma linguagem de programação para estabelecer a lógica e a seqüência do ciclo de trabalho. Um terminal de computador é usado para dar entrada nas instruções de programa ao controlador e às vezes um “teach-in-box”, devido a dificuldade em definir localizações espaciais das posições a serem usadas no ciclo de trabalho, é necessário para definir os locais dos diversos pontos no espaço de trabalho. As linguagens de robô permitem o uso de cálculos, fluxo lógico mais detalhado, sub-rotinas, e um maior uso de sensores e comunicações. Conseqüentemente o uso das linguagens textuais é freqüente nos robôs inteligentes.



Figura 2.20 - Ensino via Teach-in-box (Freitas, 2002)

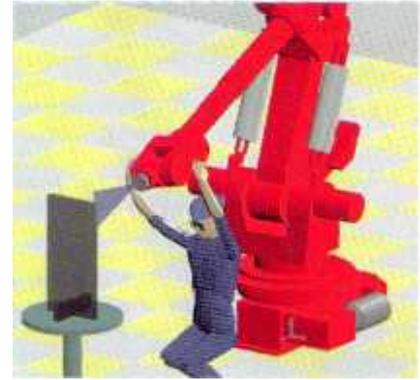


Figura 2.21 - Ensino manual (Freitas, 2002)

3. REVISÃO BIBLIOGRÁFICA

Para a determinação e controle do movimento ao se percorrer uma trajetória existem inúmeras formas, todas elas se baseiam em gerar funções de interpolação que representarão o deslocamento do punho do manipulador ao longo do tempo. A seguir serão resumidamente apresentados algumas destas formas com suas respectivas vantagens e desvantagens, e em seguida serão feitos comentários sobre estas implementações.

a) Deslocamento ponto a ponto

Este tipo de deslocamento se caracteriza pelo fato de que no controle da trajetória apenas se levam em consideração os pontos inicial e final da trajetória. Conhecendo-se as velocidades nestes pontos, a função de interpolação poderá ser um polinômio de terceira ordem (equação 3.1), ou no caso de se fixar também as acelerações inicial e final, utiliza-se um polinômio de ordem cinco (equação 3.2) (Sciavicco, 2001).

$$q(t) = a_3 * t^3 + a_2 * t^2 + a_1 * t + a_0 \quad (3.1)$$

$$q(t) = a_5 * t^5 + a_4 * t^4 + a_3 * t^3 + a_2 * t^2 + a_1 * t + a_0 \quad (3.2)$$

b) Deslocamento com velocidades fixadas

Este item, assim como os seguintes, considera a trajetória como sendo uma curva qualquer representada por mais que dois pontos.

Quando se tem um certo número de pontos N em uma trajetória, estes pontos podem ser interpolados através de um polinômio de ordem $N-1$, porém utilizar este tipo de polinômio acarreta determinadas desvantagens (Sciavicco, 2001):

- Não é possível fixar a velocidade inicial e a velocidade final.
- Quanto maior a ordem do polinômio, maior será a oscilação da trajetória a percorrer.
- A precisão numérica computacional dos coeficientes polinomiais diminui com aumento da ordem.
- O número de equações poderá ser muito grande, conseqüentemente, exigirá maior tempo para a solução.
- Os coeficientes do polinômio dependem de todos os pontos fixados; assim, caso se queira mudar um ponto, todo o sistema terá de ser recalculado.

Para se evitar as desvantagens citadas, uma idéia é aplicar polinômios de terceira ordem entre dois pontos, ou seja, a trajetória completa será formada de $N-1$ polinômios cúbicos.

c) Deslocamento com acelerações fixadas

Metodologia semelhante à anterior, porém aqui são usados polinômios cúbicos especiais, chamados *splines*. Splines são funções suaves que interpolam uma seqüência de pontos assegurando continuidade da função e de suas derivadas (Sciavicco, 2001).

d) Movimento com B-spline

A curva chamada de *B-spline* é um polinômio com as mesmas características da spline já apresentada (garantia da continuidade), porém com algumas características próprias. Quando se têm os pontos da trajetória como dados de entrada, será acrescido a cada ponto um fator numérico chamado *peso*. Este último representa a importância que este ponto terá na construção da trajetória que, ao contrário da spline comum, não contém necessariamente todos os pontos, exceto o primeiro e o último ponto que sempre farão parte da curva (Wagner, 1995; Juttler & Wagner, 1996). A figura 3.1 representa uma B-spline onde p_0, p_1, p_2, p_3 e p_4 são pontos da trajetória e cada um possui seu respectivo peso.

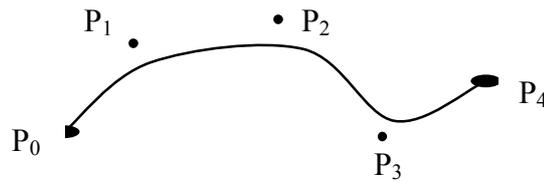


Figura 3.1 – Exemplo de curva B-spline.

Os métodos de geração de trajetória apresentados são largamente utilizados em simulação e controle de robôs manipuladores, porém essas metodologias baseiam-se no fato de que os pontos fixados serão exatamente alcançados, ou seja, o manipulador tem a capacidade de atingi-los, mas isso não está completamente certo. No caso que foi utilizado neste trabalho, por exemplo, as juntas utilizarão motores de passo (Capítulo seis) que possuem uma resolução, assim o ponto que se deseja alcançar poderá estar entre pontos realmente alcançáveis. No capítulo oito uma metodologia por nós desenvolvida que ao mesmo tempo em que seleciona dentre os pontos alcançáveis os que mais se aproxima da trajetória desejada, já cria a trajetória através de retas que ligam os pontos escolhidos.

No tocante à velocidade ela depende da aceleração que pode ser:

- linear
- não – linear

Em termos de minimizar o tempo de realização do percurso o melhor perfil de velocidade é se ter aceleração e desaceleração não – lineares e diferentes (Jeon, 2000).

No presente trabalho opta-se pelo perfil de velocidade trapezoidal, pois se tem maior interesse por precisão de movimento do que por menor tempo de execução de tarefas.

O manipulador desenvolvido que tem como principal aplicação transferir peças de esteiras para máquinas operatrizes e de máquinas operatrizes para estiras ou para bases fixas, ou seja será um manipulador do tipo pega-põe. Desta forma considera-se que a velocidade em cada ponto a ser alcançado será nula.

O programa desenvolvido, considerando-se tudo o que foi explanado neste e no capítulo anterior, pode ser dividido em três módulos:

1) Configuração do manipulador: Módulo responsável por armazenar os dados referentes a construção do manipulador, por exemplo, tipos de juntas e suas posições iniciais, comprimento de eixos, características dos motores de passo utilizados, etc.

2) Seleção de pontos a serem alcançados: Neste módulo, faz-se a seleção, utilizando metodologia própria, dentre os pontos alcançáveis quais devem ser escolhidos como ponto metas na trajetória a ser desenvolvida.

3) Perfil de velocidade: Desenvolvimento do perfil de velocidade trapezoidal entre cada par de pontos meta, onde a velocidade é nula sobre estes pontos, mas existindo aceleração e desaceleração entre eles.

4. FORMULAÇÃO MATEMÁTICA

A simulação das diversas configurações que um manipulador pode ter exige uma adequada representação matemática. O principal objetivo da formulação matemática é conseguir relacionar os movimentos das juntas que compõem o robô, sejam elas rotacionais ou prismáticas, com a posição final do punho do mesmo.

Utilizou-se no trabalho a *representação de Denavit-Hartenberg* para a modelagem. Esta representação consiste em, após ser alocado adequadamente sistemas de coordenadas nas juntas, se fazer, através da Álgebra Linear, algumas transformações homogêneas (rotações e translações), relacionando-se os sistemas de coordenadas contíguos. Segue no tópico 4.1 uma descrição detalhada de todos os passos para a devida utilização da representação de Denavit-Hartenberg e no tópico 4.2 serão vistos alguns exemplos de sua aplicação (Alves, 1985; Sciavicco, 2001).

4.1 Representação de Denavit-Hartenberg

Para a utilização adequada da representação de Denavit-Hartenberg sempre são seguidos três passos: 1) Alocação de sistemas de coordenadas nas juntas. 2) Cálculo dos parâmetros de Denavit-Hartenberg. 3) Substituição dos valores dos parâmetros na matriz homogênea de transformação.

Por uma questão didática comecemos demonstrando a construção da matriz de transformação homogênea. Que relacionará as coordenadas de um ponto dados em função do sistema desta com o sistema de coordenadas da junta anterior.

Suponha dois sistemas de coordenadas coincidentes $X_0Y_0Z_0$ e $X_1Y_1Z_1$. A transformação homogênea que representa esse sistema é (figura 4.1 e equação 4.1):

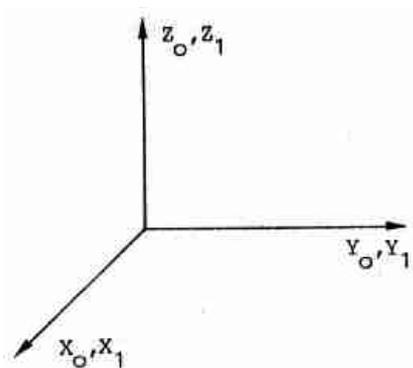


Figura 4.1 – Eixos coincidentes

$$A_0^1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4.1)$$

Admitamos que o sistema $X_0Y_0Z_0$ é a base, isto é, o sistema é fixo. Fazendo agora uma rotação θ em torno de Z_0 (figura 4.2), fica-se com a matriz (equação 4.2):

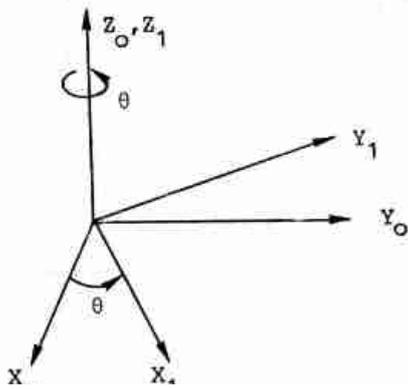


Figura 4.2 – Rotação em torno de Z_0

$$A_0^1 = \begin{bmatrix} \cos \theta & -\text{sen } \theta & 0 & 0 \\ \text{sen } \theta & \cos \theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4.2)$$

O passo seguinte é se fazer um deslocamento ao longo do eixo Z_0 de d unidades (figura 4.3).

A nova matriz (equação 4.3):

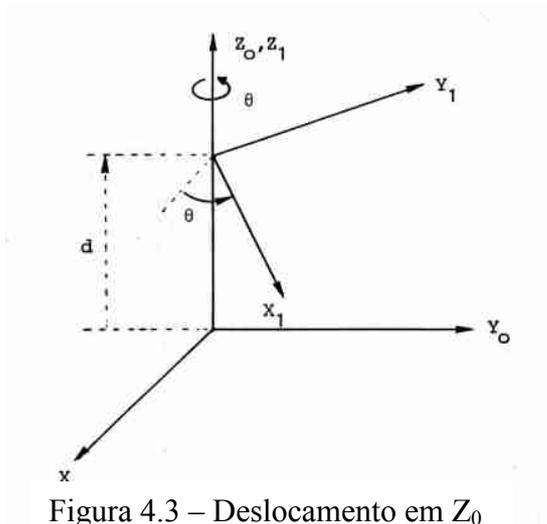


Figura 4.3 – Deslocamento em Z_0

$$A_0^1 = \begin{bmatrix} \cos \theta & -\text{sen } \theta & 0 & 0 \\ \text{sen } \theta & \cos \theta & 0 & 0 \\ 0 & 0 & 1 & d \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4.3)$$

Agora transladando $X_1Y_1Z_1$ ao longo de X_1 de a unidades (figura 4.4 e equação 4.4):

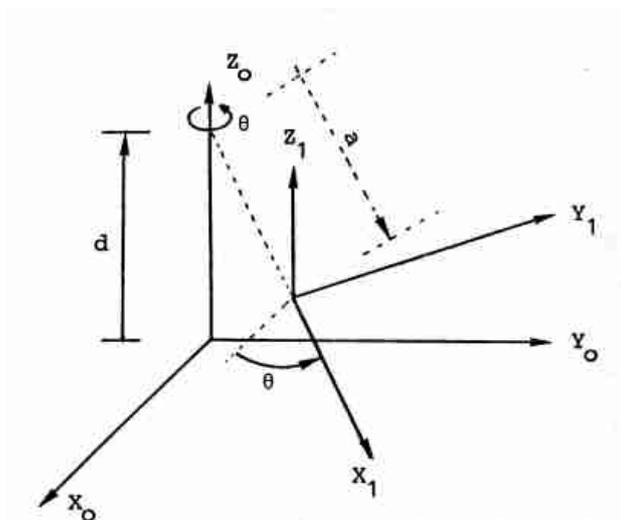


Figura 4.4 – Deslocamento na direção X_1

$$A_0^1 = \begin{bmatrix} \cos \theta & -\text{sen } \theta & 0 & a \\ \text{sen } \theta & \cos \theta & 0 & 0 \\ 0 & 0 & 1 & d \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4.4)$$

Para finalizar, rotacionar $X_1Y_1Z_1$ de α graus ao redor de X_1 (figura 4.5):

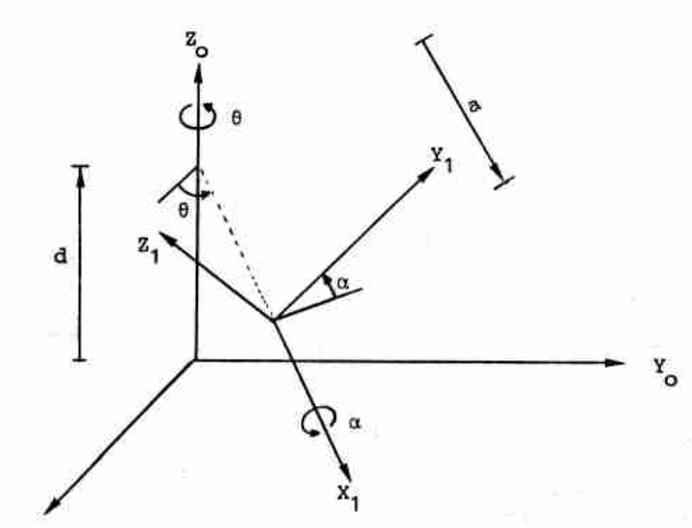


Figura 4.5 – Rotação em torno de X_1

Chega-se assim a matriz de transformação homogênea de Denavit-Hartenberg (equação 4.5):

$$A_0^1 = \begin{bmatrix} \cos \theta & -\cos \alpha * \sin \theta & \sin \alpha * \sin \theta & a * \cos \theta \\ \sin \theta & \cos \alpha * \cos \theta & -\sin \alpha * \cos \theta & a * \sin \theta \\ 0 & \sin \alpha & \cos \alpha & d \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4.5)$$

Para utilizar a transformada apresentada acima precisam-se de quatro parâmetros (θ , d , a e α) onde a e α são constantes e θ e d são as variáveis das juntas rotacional e prismática, respectivamente.

Na obtenção desses parâmetros pressupõe-se que os sistemas de coordenadas estejam apropriadamente alocados nas juntas. A alocação se dá segundo os seguintes passos:

1. Estabelecer sistema de coordenadas da base ($X_0Y_0Z_0$), sendo os três eixos perpendiculares entre si e normalizados, e Z_0 na direção do movimento da junta base.
2. Alinhar Z_i com o eixo do movimento da junta i .
3. Fazer X_i igual ao produto vetorial normalizado entre Z_{i-1} e Z_i . No caso destes eixos serem paralelos fazer X_i dirigir-se de Z_{i-1} para Z_i , ou ainda se forem coincidentes, fazer X_i ser paralelo a X_{i-1} .
4. Fazer Y_i igual ao produto vetorial normalizado entre Z_i e X_i .

Após o devido posicionamento dos diversos sistemas de referências nas juntas, avalia-se dos valores dos parâmetros (Figuras 4.1 a 4.5) (Reis, 1994):

1. d_i é a distância da origem do sistema de referência $i-1$ à intersecção dos eixos Z_{i-1} e X_i ao longo de Z_{i-1} . Variável da junta prismática.
2. θ_i é o ângulo de rotação do eixo X_{i-1} ao eixo X_i em torno de Z_{i-1} . Variável da junta rotacional.
3. a_i é a distância da intersecção dos eixos Z_{i-1} e X_i à origem do sistema de coordenadas i , ao longo de X_i .
4. α_i é o ângulo de rotação do eixo Z_{i-1} ao eixo Z_i , em torno do eixo X_i .

Para utilizar a matriz de *Denavit-Hartenberg* multiplica-se essa matriz pelo vetor posição $[x, y, z, 1]$ que representam as coordenadas x, y e z de um ponto em relação ao sistema de coordenadas X_1, Y_1 e Z_1 . Desta forma encontram-se as coordenadas deste ponto em relação ao sistema de coordenadas fixas X_0, Y_0 e Z_0 .

4.2 Exemplos de aplicação

Nos próximos tópicos serão apresentados dois exemplos de alocação do sistema de coordenadas e da avaliação dos valores dos parâmetros.

Os sistemas de coordenadas são alocados, em ambos os exemplos, segundo as regras descritas no item 4.1; um sistema em cada junta com o eixo Z na direção do movimento e mais um sistema na extremidade do braço. Os eixos X e Y seguem as referidas regras.

4.2.1 Cilíndrico

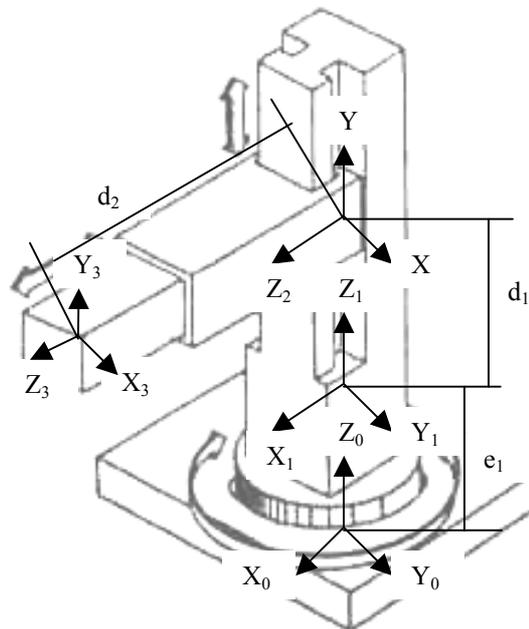
A figura 4.6 mostra um manipulador de configuração cilíndrica com os sistemas de coordenadas devidamente alocados.

Z_0 : alocado na base e aponta na direção de rotação da junta (variável θ_1 em relação a Z_1).

Z_1 : alocado a uma distância e_1 do sistema $X_0Y_0Z_0$ e aponta na direção do deslocamento linear (variável d_1 em relação a Z_2).

Z_2 : situado sobre a segunda junta prismática e apontando na direção da extremidade do braço (variável d_2 em relação a Z_3).

Z_3 : extremidade do braço.



	θ	d	a	α
A_0^1	θ_1	e_1	0	0°
A_1^2	90°	d_1	0	90°
A_2^3	0°	d_2	0	0°

Figura 4.6 – Sistema de coordenadas no robô cilíndrico

4.2.2 Antropomórfico

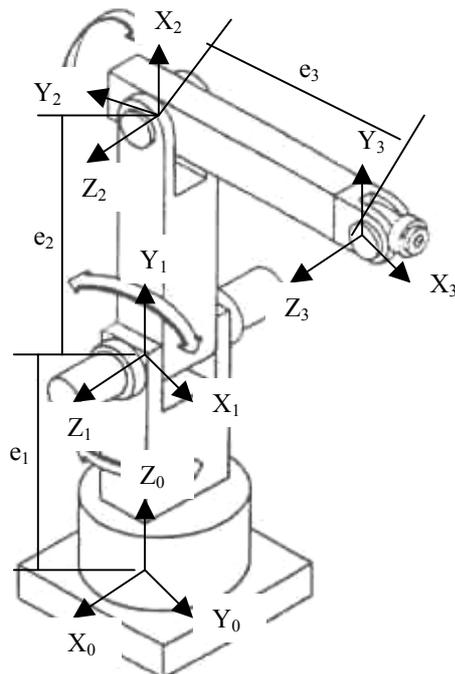
A figura 4.7 mostra um manipulador de configuração antropomórfica com os sistemas de coordenadas devidamente alocados.

Z_0 : alocado na base e aponta na direção de rotação da junta (variável θ_1 em relação a Z_1).

Z_1 : alocado a uma distância e_1 do sistema $X_0Y_0Z_0$ e aponta na direção de rotação da segunda junta (variável θ_2 em relação a Z_2).

Z_2 : situado sobre a terceira junta rotacional, aponta na direção de rotação desta, e a uma distância e_2 (variável θ_3 em relação a Z_3).

Z_3 : extremidade do braço.



	θ	d	a	α
A_0^1	θ_1	e_1	0	90°
A_1^2	θ_2	0	e_2	0°
A_2^3	θ_3	0	e_3	0°

Figura 4.7 – Sistema de coordenadas no robô antropomórfico

5. PROGRAMA (1ª PARTE)

O principal objetivo desse trabalho de dissertação foi desenvolver um programa computacional que pudesse simular os movimentos de diferentes configurações de braços robóticos, ou de forma mais generalizada, simular as mais diferentes configurações de robôs de base fixa. Para chegar a esse objetivo é necessário em primeiro lugar representar computacionalmente a configuração do robô. Em outras palavras, fazer com que o computador possa trabalhar sabendo a localização do robô no espaço, bem como a localização de suas articulações, a quantidade e o tipo das mesmas. Para esta representação aplicou-se a metodologia de Denavit-Hartenberg, e a partir desta representação foi possível computacionalmente (numericamente) atribuir valores e controlarem-se os movimentos da maneira esperada pelo usuário/operador.

Apresenta-se em seguida uma descrição do comportamento do programa desenvolvido em termos da representação geométrica de robôs. As questões de controle de posicionamento e controle de velocidade serão apresentadas em capítulo mais adiante pelo fato de que os algoritmos para estes procedimentos necessitam de informações acerca do comportamento dos elementos de acionamento das juntas robóticas, informações estas que serão explicados no capítulo seguinte.

5.1 Linguagem C++

O programa foi desenvolvido utilizando-se a linguagem de programação C++. Esta linguagem tem vantagens em relação a outras linguagens de programação pelo fato dela ser orientada ao objeto.

A programação orientada ao objeto se baseia em três conceitos: (Curso de Linguagem Java, 2003)

Encapsulamento: Mecanismo que permite agrupar em uma única entidade o código e os dados que esse código manipula. Tais elementos ficam assim protegidos contra interferências externas e utilização inadequada.

Herança: Processo pelo qual um objeto adquire as propriedades de outro objeto.

Polimorfismo: Trata-se de uma característica que permite que uma interface seja usada para uma classe genérica de ações, sendo que a ação específica em cada caso é determinada pela exata natureza da situação.

Fez-se proveito, na elaboração do programa, de toda a versatilidade do C++. Características tais como: a) Classes, cria elementos compostos por atributos e métodos; b) Ponteiros, referência dos objetos através do endereço que eles ocupam na memória do computador, torna o programa mais ágil (Laszlo, 1996; Stevens, 1991; Jamsa, 1999).

5.2 Classe Junta

A classe *Junta* é o principal elemento do programa aqui apresentado. A partir de *Junta* é que são desenvolvidos todos os procedimentos e decisões são tomadas baseando-se nela. Levando-se em consideração a importância desta, abaixo se apresenta a *Classe Junta* com seus atributos e métodos, sobre os quais serão tecidos comentários.

```
class Junta
{ private:
    Ponto *InitialPos;
    Ponto *Initial_e1;
    Ponto *Initial_e2;
    Ponto *Initial_e3;
    Ponto *position;
    Ponto *e1;
```

```

    Ponto *e2;
    Ponto *e3;
    Barra *suporte;
    Barra *movida;
    double d_min;
    double d_max;
public:
    Junta( );
    virtual ~Junta();
    virtual void setInitialPos(Ponto *);
    virtual void setInitial_e1(Ponto *);
    virtual void setInitial_e2(Ponto *);
    virtual void setInitial_e3(Ponto *);
    virtual void setPosition(Ponto *);
    virtual void set_e1(Ponto *);
    virtual void set_e2(Ponto *);
    virtual void set_e3(Ponto *);
    virtual void setLimites(double,double);
    virtual void setSuporte(Barra *);
    virtual void setMovida(Barra *);
    virtual Barra* getMovida();
    virtual Ponto *g_Position();
    virtual Ponto *g_e1();
    virtual Ponto *g_e2();
    virtual Ponto *g_e3();
    virtual Ponto *g_InitPos();
    virtual Ponto *g_Inite1();
    virtual Ponto *g_Inite2();
    virtual Ponto *g_Inite3();
    virtual void GerarSistemLocal(Junta*);
    virtual Barra* getSuporte();
    virtual void move(double);
};

```

Inicialmente vale ressaltar que na classe acima apresentada os atributos são todos privados (private) e os métodos públicos (public), isso quer dizer que os atributos não podem ser acessados diretamente por outras classes, mas somente por meio dos métodos.

Os primeiros atributos de *Junta* são ponteiros para o tipo *Point*, que é a classe mais básica do programa, ela representa uma coordenada no espaço tridimensional. Os primeiros quatro atributos representam a posição inicial da junta, *InitialPos*, e as direções iniciais do sistema de coordenadas alocado nesta, direções X, Y e Z, respectivamente *Initial_e1*, *Initial_e2* e *Initial_e3*. Os quatro atributos seguintes são similares aos atributos que os antecedem, a diferença entre os dois grupos é que o primeiro refere-se à posição inicial e o segundo refere-se à posição atual da junta.

Os demais atributos são: *suporte* e *movida* que são ponteiros que indicam quais as barras (eixos) estão conectadas à junta; e *dmin* e *dmax* são os limites de deslocamento das juntas, limites estes definidos pela construção mecânica do robô e de suas juntas.

Nos métodos têm-se inicialmente o *Construtor* e o *Destrutor* da classe, em seguida os métodos *set* são usados para atribuir valores aos atributos da classe e os métodos *get* e *g_* tem como valor de saída valores dos atributos.

O procedimento *move* faz com que a junta desloque a barra apontada como *Movida* de um ângulo passado como parâmetro, assim toda a estrutura conectada em sua saída se moverá. *GerarSistemLocal* gera os eixos de coordenada X e Y do sistema local.

Existem no programa duas classes que herdam *Junta*, ou seja, ficam com as mesmas características, atributos e métodos da classe base, estas classes são: *J_Revolução* e

J_Prismática. Estas representam as juntas de revolução que têm θ como variável e as juntas prismáticas que têm d como variável.

5.3 Classes Barra e Dena-Hart

Como descrito no primeiro capítulo, um robô é formado em sua forma mais básica por juntas e eixos. Na representação computacional aqui apresentada os eixos estão representados pela classe barra.

```
class Barra
{
private:
    Junta *orig;
    Junta *dest;
    Dena_Hart *DH;
public:
    Barra(Junta *, Junta *);
    virtual ~Barra();
    virtual Junta *getOrig();
    virtual Junta *getDest();
    virtual Dena_Hart *getDH();
};
```

Lembrando a definição de eixo: “elemento de ligação entre duas juntas”. A classe criada para representar este elemento de forma mais realista possível tem como atributos dois ponteiros para *Junta* (*orig* e *dest*), assim a *barra* conhece a junta anterior e a junta posterior a si. Existe ainda um ponteiro (*DH*) para a classe *Dena-Hart* que é a classe responsável pela criação e manutenção dos parâmetros e matrizes da representação de Denavit-Hartenberg.

Entre os métodos o único diferencial entre a classe *barra* e a classe *junta* é o Construtor de *barra* que, ainda segundo a definição, para ser criada precisa que duas juntas sejam passadas como parâmetro.

A classe *Dena-Hart* é utilizada sempre para relacionar dois sistemas de coordenadas diferentes, já que estes sistemas estão acoplados às juntas e as barras conectam as juntas, assim fica óbvio que cada barra tem uma matriz DH (Denavit-Hartenberg) relacionada à ela. Vamos à classe *Dena-Hart*:

```
class Dena_Hart
{
private:
    double d;
    double teta;
    double a;
    double alfa;
public:
    Dena_Hart(Junta*,Junta*);
    ~Dena_Hart();
    void setteta(double);
    void setd (double);
    Matriz MatrizDH();
    void print();
    double getd();
    double getteta();
    double geta();
    double getalfa();
};
```

Os atributos são os parâmetros da representação DH: d , $teta$, a e $alfa$. Os métodos são similares aos métodos de outras classes, exceto *MatrizDH* que retorna um elemento do tipo *Matriz* (outra classe básica do programa, além de *Point*) que vem a ser utilizada pelo programa, e *print* reproduz a matriz DH na tela do computador.

O Construtor de *Dena-Hart* recebe como parâmetro dois ponteiros para *Junta*, onde cada uma das juntas tem o seu sistema de coordenadas adequadamente posicionado, e calcula os parâmetros de DH:

d : 1- Encontra o ponto de intersecção entre o plano perpendicular ao eixo Z_i que contem a origem do sistema de coordenadas da junta $i+1$ e o próprio eixo Z_i .

2- Levando-se em consideração o sentido de Z_i , atribui-se a d a distância da origem do sistema i ao ponto encontrado no passo um, levando-se em consideração o sentido, assim este valor poderá ser positivo ou negativo.

$teta$: 1- Valor do ângulo entre os eixos X_i e X_{i+1} é atribuído à $teta$, levando-se em consideração o sentido de rotação.

a : 1- Encontra o ponto de intersecção entre o plano perpendicular ao eixo X_{i+1} que contém a origem do sistema de coordenadas da junta i e o próprio eixo X_{i+1} .

2- Levando-se em consideração o sentido de X_{i+1} , atribui-se a a a distância da origem do sistema $i+1$ ao ponto encontrado no passo um, levando-se em consideração o sentido.

$alfa$: 1- Valor do ângulo entre os eixos Z_i e Z_{i+1} é atribuído à $alfa$, levando-se em consideração o sentido de rotação.

A representação dos eixos se dá pelas coordenadas do vetor unitário paralelo à eles e a coordenada da origem do sistema local. É requerido sempre que a representação seja unitária (normalizada) para fins de comparações necessárias ao funcionamento do programa.

6. COMPONENTES MECÂNICOS E ELETRÔNICOS

A finalidade do programa desenvolvido e aqui apresentado é representar de forma tão realística quanto possível os movimentos de um manipulador industrial, sendo assim, têm de ser inseridos no contexto da simulação, os elementos mecânicos e eletrônicos que constituem a estrutura de um robô.

6.1 Mecânica

Um robô é composto de inúmeros componentes mecânicos, parafusos de aperto, molas, rolamentos e a própria estrutura do robô são alguns dos componentes, mas na simulação computacional os componentes mecânicos que são imprescindíveis de estarem presentes, ou ao menos algumas de suas características, são os elementos de transmissão, tais como engrenagens, correias, sem-fins-coroa, entre outros.

Os elementos de transmissão interferem no desempenho cinemático (deslocamento e velocidade) e no desempenho dinâmico (esforços) da estrutura, desta forma para uma boa representação estes elementos devem ser levados em consideração. (Hall, 1990)

6.2 Eletrônica

Para movimentar um robô, utilizando o computador como central para processamento de dados, são empregados diversos componentes ou equipamentos eletro-eletrônicos. Como motores de passo, placas de interfaceamento (drivers) e o próprio microcomputador. Através destes se consegue um controle preciso com relativa facilidade.

Os principais equipamentos/componentes eletro-eletrônicos que compõem diferentes mecanismos.

- Motor de Passo
- Porta Paralela
- Circuito Eletrônico

Os motores de passo foram componentes de muita importância neste trabalho, sendo assim o seu funcionamento será descrito abaixo. Os demais componentes, por serem usuais em eletrônica são apresentados à parte nos anexos.

6.2.1 Motor de Passo

Tipo particular de motor elétrico, comumente encontrado em periféricos de computador, a principal característica do motor de passo é sempre girar em incrementos angulares discretos.

Observando o esquema na figura 6.1, tem-se um rotor central e um estator composto por quatro pólos. Energizando-se o pólo 1, o rotor fica na posição mostrada, em seguida energiza-se o pólo 2, o rotor girará no sentido horário para ficar alinhado com o pólo ativado, e assim sucessivamente de forma a fazer com que o rotor gire. Conseguindo realizar uma seqüência de energização em um tempo relativamente curto, o movimento do rotor parecerá contínuo. (Ramos Filho, 2001)

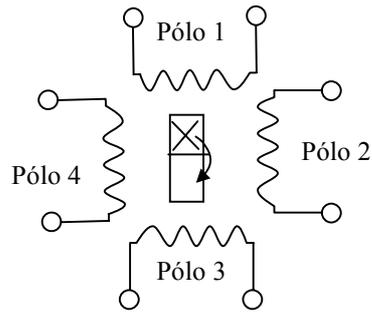


Figura 6.1 - Esquema de bobinas em um motor de passo.

A resolução, número de passos por volta, de um motor de passo é determinada pela quantidade de pólos no estator e rotor. O ângulo de passo de um motor normalmente varia entre $1,8^\circ$ e 12° , sempre um valor inteiro para $360^\circ \div \text{ângulo de passo}$.

Os motores de passo são utilizados em funções que não requerem grandes velocidades nem tampouco grandes esforços, mas em compensação uma precisão de posicionamento muito boa, mesmo sendo um equipamento que normalmente trabalha em malha aberta.

7. PROGRAMA (PARTE 2)

Neste capítulo volta-se a descrição do programa computacional desenvolvido. A apresentação do referido foi dividido em partes com vista à facilitar a compreensão de sua estrutura e passar a idéia ao leitor de como foram sendo desenvolvidos os procedimentos do programa.

7.1 Classe Motor e Motor_Passo

A classe *Motor* representa o elemento que movimenta a junta, como a junta pode ser movida de diferentes formas a classe *Motor* foi criada de forma tal que sirva como classe base para outras classes.

A simulação utiliza *Motor_Passo* como elementos que movimentam as articulações:

```
class Motor_Passo: public Motor
{
public:
    double resoluc;
    int bobina;
public:
    Motor_Passo (double);
    ~Motor_Passo();
    void EnergeMotor(int*);
    void MoveMotor(int);
    double getResoluc();
};
```

A primeira linha de comandos mostra que *Motor_Passo* herda a classe *Motor*. *Motor_Passo* possui apenas dois atributos *resoluc* e *bobina*, o primeiro refere-se a quantidade em graus que o eixo do motor se desloca a cada pulso e *bobina* é um número inteiro que representa qual a bobina que está energizada.

O método *EnergeMotor* recebe um vetor de inteiros como parâmetros e os utiliza para saber qual a próxima bobina que deve ser energizada, em seguida envia uma informação para *MoveMotor* que efetiva a energização e ativa o deslocamento da junta.

7.2 Classe Acionador

Os elementos mecânicos de transmissão (engrenagens, polias, etc) são representados na simulação pela classe *Acionador*:

```
class Acionador
{
public:
    Junta *joint;
    Motor_Passo *mot;
    double reducao;
public:
    Acionador(double, Junta*, Motor_Passo*);
    ~Acionador();
    void setReducao(double);
    Motor_Passo* getMot();
    void MoveJunta (double);
    double getReducao();
    Junta* getJunta();
};
```

Entre os atributos da classe incluem-se *joint* e *mot* que são apontadores, respectivamente, para a junta e para o motor que se conectam ao acionador. O atributo *reducao* armazena o valor da redução na transmissão entre os elementos mecânicos que o constituem.

O método *MoveJunta* recebe como parâmetro o valor em graus do deslocamento do motor e o repassa em radianos para a classe *Junta*, levando em consideração *reducao*, e esta última efetivamente movimenta a junta.

7.3 Classe Porta_Paralela

```
class Porta_Paralela
{
private:
    int *pinos;
public:
    Porta_Paralela ( );
    ~Porta_Paralela ( );
    void Setnum (int,int);
    void Sinal_1 (int*);
    void Sinal_2 (int*);
    int Oposto (int);
    int* GetPinos();
};
```

O Construtor de *Porta_Paralela* inicializa os valores do vetor *pinos*. Vetor este que representa o estado lógico dos pinos (capítulo 6.2.2) da porta paralela do computador. *Setnum* transforma um inteiro em número binário.

Sinal_1 atribui valores aos 8 (oito) primeiros *pinos* de *Porta_Paralela* e *Sinal_2* atribui valores aos 4 (quatro) *pinos* seguintes. A existência de duas funções é justificada pelo fato de que quatro dos doze pinos controlados são tratados de maneira diferenciada dos demais.

O método *GetPinos* retorna o valor *pinos* com seus atuais valores.

7.4 Classe Placa_IORobot

Como ocorre em um controle real, no mundo virtual criado existe um circuito eletrônico montado sobre uma placa que faz a conexão entre computador e os motores de passo.

A classe *Placa_IORobot* tem um comportamento semelhante à placa física real. Ela recebe sinais em sua entrada e os distribui entre suas saídas (três saídas ao todo), saídas às quais são conectados os motores.

```
class Placa_IORobot
{
private:
    int *entrada[12];
    int *saida_1[4],*saida_2[4],*saida_3[4];
public:
    Placa_IORobot();
    ~Placa_IORobot();
    void SetEntrada(int*);
    int* GetSaida(int);
};
```

Placa_IORobot possui os atributos *entrada*, *saida_1*, *saida_2* e *saida_3* que, inicialmente através do Construtor, recebem o valor NULL, pois inicialmente a placa está desativada e não possui sinais em sua entrada e em nenhuma de suas saídas.

O método *SetEntrada* é o procedimento que “lê” os pinos de *Porta_Paralela* e os retransmite aos motores através de *GetSaida*.

7.6 Resumo

Após ter-se apresentado as principais classes referentes à configuração, exceto a classe *Computador* que será apresentada mais adiante, pode ser observado o quanto um problema complexo, referindo-se à simulação computacional de um sistema real composto por diversos componentes pode tornar-se relativamente simples a partir do momento em que se utiliza a filosofia da programação orientada ao objeto.

O sistema que no mundo real é composto por um microcomputador, porta paralela, drive (placa) de comunicação, motores de passo, redutores, juntas rotacionais e/ou prismáticas e eixos, no programa computacional desenvolvido também existem.

A figura 7.1 mostra como os componentes descritos nas seções anteriores estão ligados uns aos outros. O programa desenvolvido, em seu simulador, segue exatamente a mesma seqüência; o computador se comunica com a porta paralela enviando pulsos que são repassados a um circuito eletrônico, este está ligado a um motor de passo acoplado em um acionador que movimentam a junta e conseqüentemente os eixos que compõem o manipulador.

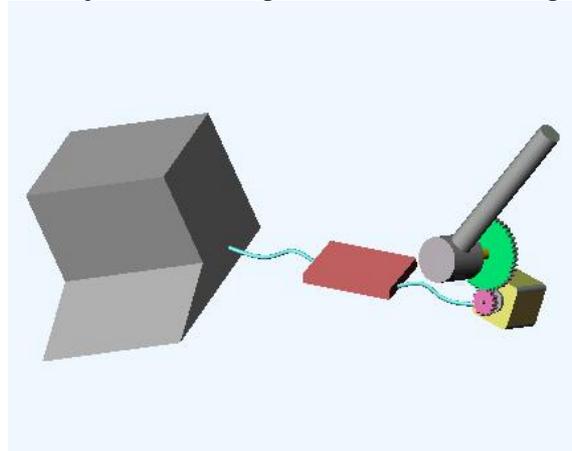


Figura 7.1 - Sistema simulado: computador, placa eletrônica, motor de passo, acionador (reductor), junta e barra.

8. PLANEJAMENTO DE TRAJETÓRIA E CONTROLE DE VELOCIDADE

Será descrito um algoritmo generalizado para o planejamento de trajetórias, mas sua implementação computacional foi feita exclusivamente para a configuração de GOBLIN (antropomórfica) pelo fato de que as variáveis envolvidas no equacionamento da posição do punho do robô dependem da configuração deste, então foi implementado para a configuração do manipulador que se tinha em mãos.

O controle de velocidade utilizado é generalizado, assim o procedimento programado é independente da configuração do robô.

8.1 Trajetória

Para a execução de um ciclo de trabalho é requerido do manipulador que ele descreva uma determinada trajetória para se deslocar de um ponto à outro dentro do volume de trabalho. Esta trajetória pode ser um percurso qualquer que computacionalmente é representado por uma série de pontos. Porém devido à questão da resolução nos movimentos das juntas, o punho do manipulador não possui a capacidade de atingir qualquer ponto no espaço, mas sim pontos discretos.

Surge o primeiro obstáculo na questão de otimizar a realização do movimento. Se for fornecido ao manipulador, como dado de entrada, alguns pontos a serem atingidos e caso alguns deles, ou todos, não puderem ser alcançados com exatidão, alguma decisão deve ser tomada, o que deve ser feito? No presente trabalho aplicam-se duas metodologias, uma delas bem simples e rápida e a outra mais elaborada e com maior tempo de processamento, mas que para determinadas aplicações apresenta um resultado mais próximo do movimento real.

Como estamos trabalhando com movimentos discretos, utilizamos ao invés do espaço real um espaço parametrizado cujas escalas nos eixos dos sistemas de coordenadas variam de zero a 2π e onde todos os pontos alcançáveis são vértices de paralelepípedos, desta forma fica mais fácil visualizar o movimento. No espaço real devido à resolução dos motores, existe uma grande concentração de pontos alcançáveis em determinadas partes do volume de trabalho e poucos pontos em outras partes, enquanto no espaço parametrizado os pontos se distanciam de forma menos diferenciada.

A figura 8.1 apresenta dois exemplos que deixam claro como o espaço parametrizado facilita o entendimento e a visualização: a) pontos alcançáveis por um robô antropomórfico, onde cada um dos motores que o movimenta possui resolução de 30° ; b) o mesmo robô no espaço parametrizado; c) pontos alcançáveis por um robô antropomórfico com resolução de 15° no espaço real; e d) o mesmo robô no espaço parametrizado.

Os eixos de coordenadas no espaço parametrizado, ao invés de XYZ são as posições alcançadas por cada um dos motores entre 0 e 2π . Vale destacar que motores de passo podem ter resoluções bem menores ($3,8^\circ$, $1,6^\circ$, etc).

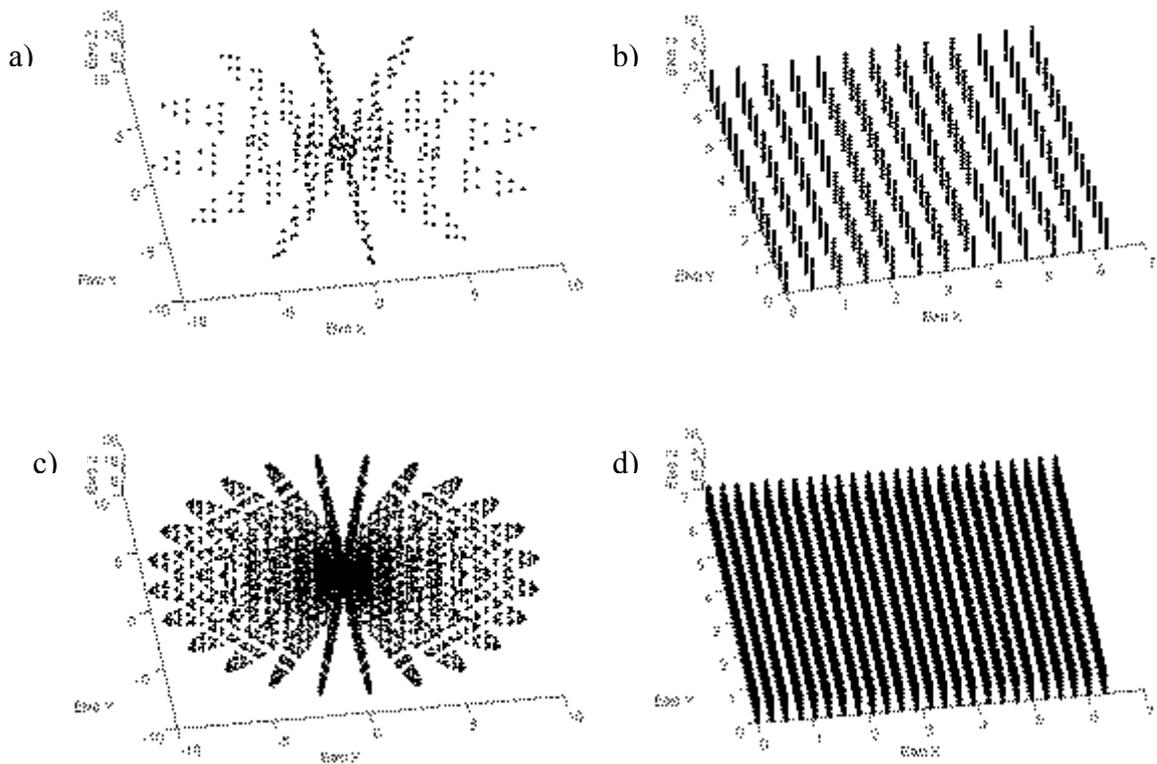


Figura 8.1: Pontos alcançáveis por um robô antropomórfico: a) resolução de 30° no espaço real; b) resolução de 30° no espaço parametrizado; c) resolução de 15° no espaço real; e d) resolução de 15° no espaço parametrizado.

8.1.1 Mínima Distância entre Pontos

O número de pontos alcançáveis pelo manipulador que rodeiam um ponto qualquer que se deseja atingir pode ser até de 2^N , onde N é o número total de articulações do manipulador. No caso específico que mais foi abordado neste trabalho, o manipulador com três graus de liberdade, o ponto que se deseja atingir pode estar contido em um hexaedro (figura 8.2).

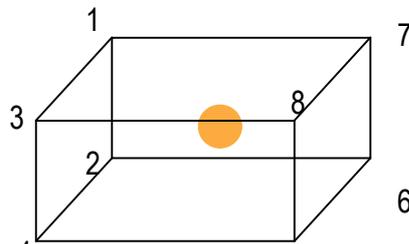


Figura 8.2: Hexaedro contendo o ponto que se deseja alcançar, no caso de manipulador com três graus de liberdade.

A aplicação da técnica da Mínima Distância consiste em se mensurar a distância do ponto que se quer alcançar aos pontos que compõem o elemento que o contém e seleciona-se de imediato o que estiver mais próximo. Chama-se de elemento à figura geométrica que tem por vértices os pontos da vizinhança do ponto desejado.

Esta técnica é bem simples de ser compreendida, implementada e de rápida resolução computacional. Mais adiante será feita uma comparação com outra metodologia.

8.1.2 Mínima Área entre Curvas

Quando se deseja ir de um ponto à outro (A-B) e não é possível se fazer este percurso, mas somente um aproximado (C-D) comete-se um erro (figura 8.3), e decidiu-se mensurar esse erro, como sendo a área entre os segmentos de reta, ou seja, quanto menor for a área, menor será o erro, e mais próxima a trajetória realizada estará da trajetória requerida.

A área calculada será a área do quadrilátero formado pelos segmentos AB, BD, DC e CA.

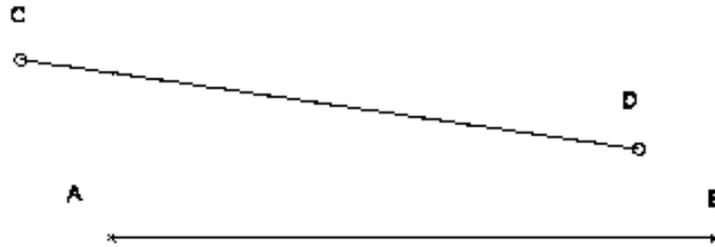


Figura 8.3 – Trajetória desejada AB e trajetória possível CD.

Para o cálculo da área divide-se o quadrilátero em dois triângulos (ABC e BCD), calcula-se a área de cada um deles e soma-se para ser obtida a área do quadrilátero (figura 8.4).

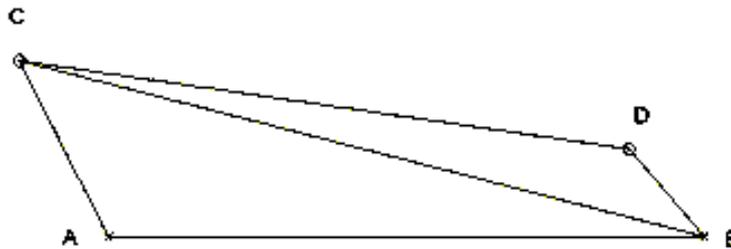


Figura 8.4 – Divisão do quadrilátero em dois triângulos.

Em várias situações as retas se cruzam. Nestes casos o procedimento é o mesmo, pois se cálculo da área do triângulo ABC e do triângulo BCD, que é a mesma área do quadrilátero ABCD que se obtém ao rebater a linha CD ao redor de CB (figura 8.5).

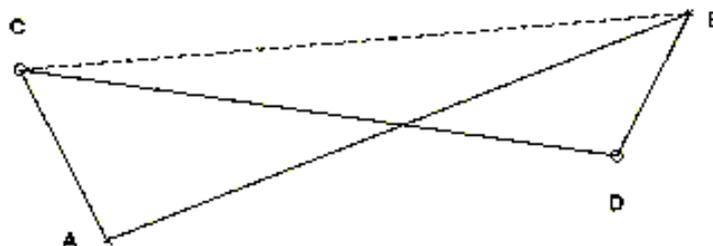


Figura 8.5 – Trajetórias cruzadas.

Para a aplicação desta técnica monta-se uma estrutura de dados da seguinte forma:

- 1- Cria-se uma matriz de ponteiros do tipo *Ponto* onde cada linha desta matriz é formada pelos pontos vizinhos ao ponto que se deseja alcançar.
- 2- Percorre-se a matriz selecionando-se um elemento por linha, criando um “caminho”.
- 3- Compara-se este caminho com a trajetória requerida e calcula-se o erro entre as duas curvas.

- 4- O passo 3 é repetido para todos os possíveis caminhos e aquele que apresentar o menor erro será utilizado como trajetória ótima.

Vale a ressalva de uma aparente “falha” neste método, isto ocorre quando os quatro pontos, dois da trajetória ideal e dois da trajetória possível forem colineares (figura 8.6), neste caso o erro encontrado será zero, pois não haverá área entre os dois segmentos de reta. O erro zero pode dar a impressão de que as trajetórias são idênticas, ou seja, que o robô realizará efetivamente a trajetória requerida, e isto pode não ser verdadeiro.



Figura 8.6 – Pontos coincidentes.

Na figura 8.7 apresenta-se uma série de figuras que mostram a minimização da área entre as curvas, e conseqüentemente uma melhor forma de aproximar as curvas.

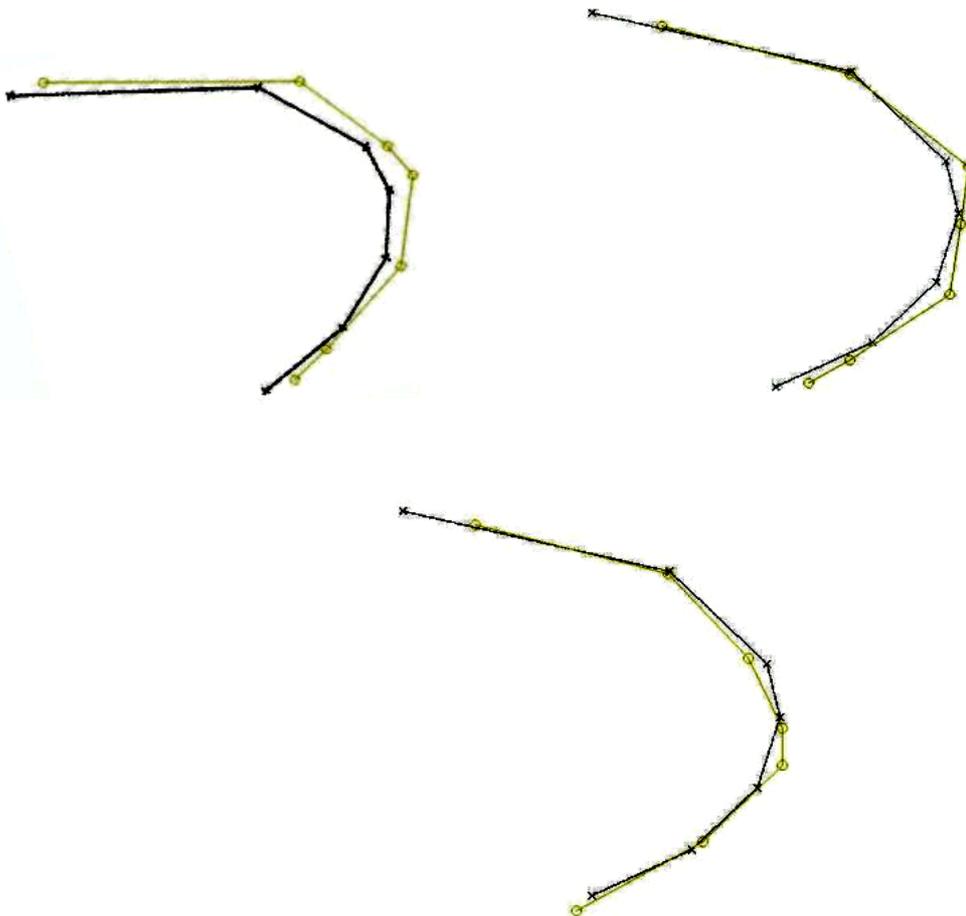


Figura 8.7 – Aplicação do método das mínimas áreas.

8.1.3 Mínima Distância X Mínima Área

Fazendo-se uma comparação entre os dois métodos chegam-se a algumas conclusões.

- 1) O método da Mínima Distância é muito mais rápido do que o Método da Mínima Área, sobretudo quando se considera utilizado na implementação do método (este sistema será detalhado mais adiante).
- 2) A Mínima Área tende a escolher caminhos que sejam os mais “paralelos” possíveis com a trajetória desejada, enquanto no método das Mínimas Distâncias podem haver vários cruzamentos de linhas.

A princípio pode-se imaginar que os resultados em ambos os métodos são os mesmos, porém essa idéia não é verdadeira, para demonstrar isso utilizaremos a figura 8.8 que destaca um trecho de uma dada trajetória.

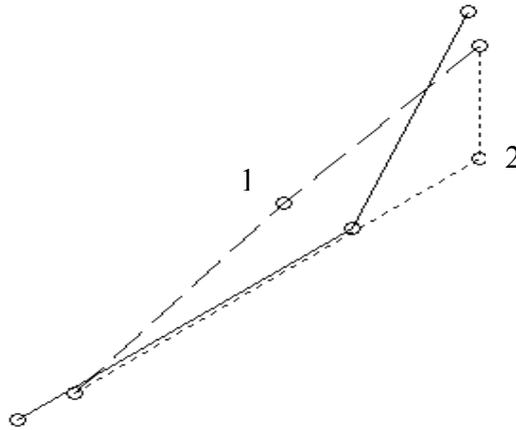


Figura 8.8 – Comparação dos métodos aplicados. Os círculos ligados pela linha tracejada são encontrados pelo método das mínimas distâncias e os círculos ligados pela linha pontilhada pelo método das mínimas áreas.

A linha “cheia” representa a trajetória desejada, a linha tracejada representa a trajetória pela Mínima Distância e a trajetória pontilhada representa a trajetória selecionada pelas Mínimas Áreas. Percebe-se que os primeiros e últimos pontos em ambos os métodos são os mesmos, mas os pontos intermediários são diferentes, isto acontece porque as áreas formadas quando se escolhe 2 são menores que as áreas formadas pelo ponto 1, por esta razão existe a diferença.

8.2 Velocidade

As articulações de um manipulador para realizar seus movimentos com precisão e boa flexibilidade em ser reprogramado para novas tarefas, necessitam de um controle de velocidade. Uma questão que não pode ser desconsiderada neste controle é o sincronismo entre as juntas de forma tal que se possa realizar o movimento como planejado.

Estamos trabalhando com motores de passo, e estes funcionam via pulsos elétricos. Assim, por exemplo, se em um determinado robô contendo dois graus de liberdade lineares (X e Y) para alcançar determinado ponto são necessários 10 (dez) pulsos em X e 8 (oito) pulsos em Y, estes pulsos podem ser dados em qualquer combinação (figura 8.9).

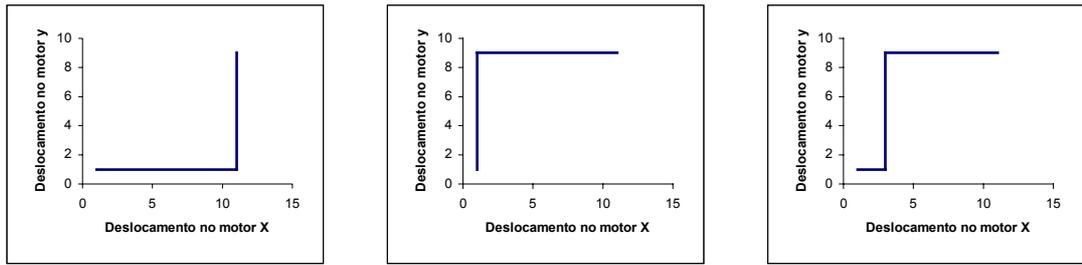


Figura 8.9 – Aplicação de pulsos aos motores. Da esquerda para a direita: 10 pulsos em X e 8 em Y; 8 pulsos em Y e 10 em X; 2 pulsos em X, 8 pulsos em Y e 8 pulsos em X.

Estes pulsos podem também ser simultâneos como mostra a figura 8.10:

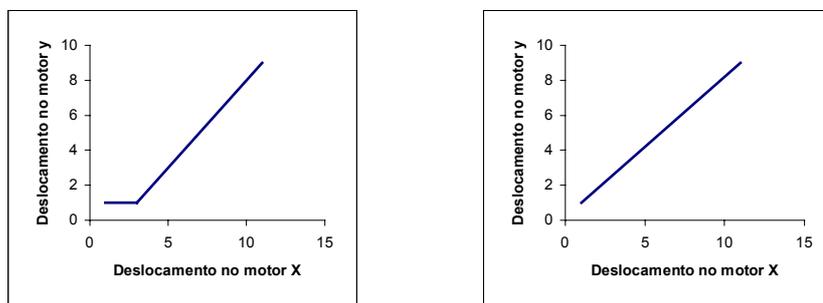


Figura 8.10 – Sincronismo de movimentos. A figura da esquerda mostra 2 pulsos em X e em seguida 8 pulsos em X e Y simultâneos. Na figura da direita s tem os 10 pulsos em X e os 8 pulsos em Y realizados ao mesmo tempo.

A última trajetória da figura 8.10 é que verdadeiramente representa a trajetória planejada, desta forma chega-se a conclusão de que além da seleção dos pontos que melhor representam o percurso requerido, deve-se também ter pleno controle da velocidade do manipulador durante o movimento para que possa haver um sincronismo de movimentos.

8.2.1 Perfil de Velocidade

Baseando-se em livros, artigos e em experiência própria, concluiu-se que os motores para serem utilizados com uma boa confiabilidade, teriam de ter um perfil de velocidade com aceleração e desaceleração definidas (Jeon, 2000). Para acelerar um motor de passo vai se aumentando a frequência com que os pulsos são enviados ao motor, para desacelerar basta diminuir a frequência dos pulsos. No presente caso o que o programa faz é enviar à *Porta Paralela* um conjunto de pulsos com tamanho pré-definido a uma determinada frequência. Em seguida envia-se outro conjunto a uma frequência maior e segue desta forma até atingir a velocidade de cruzeiro, e depois faz-se o mesmo processo em ordem inversa. Para desacelerar, envia-se um conjunto de pulsos à uma determinada frequência, em seguida outro conjunto em uma frequência mais baixa e segue-se desta maneira até cessar o movimento.

Obtem-se para os motores de passo que sigam essa metodologia de variação de velocidade um perfil *tempo X velocidade*, como apresentado:

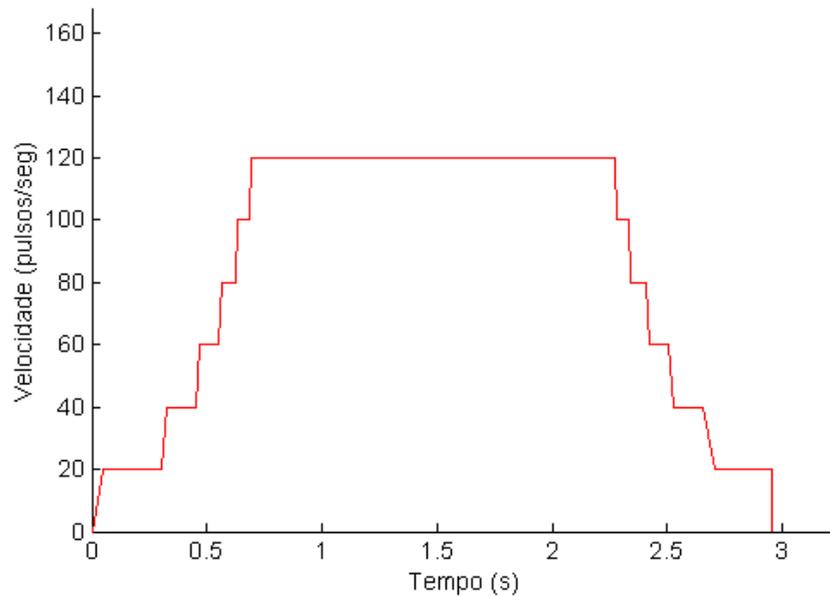


Figura 8.11 – Perfil de velocidade.

Nas aplicações práticas se tem mais de um motor, considerando que os valores de aceleração e desaceleração são os mesmos obtém-se, por exemplo, para três motores, o gráfico da figura 8.12.

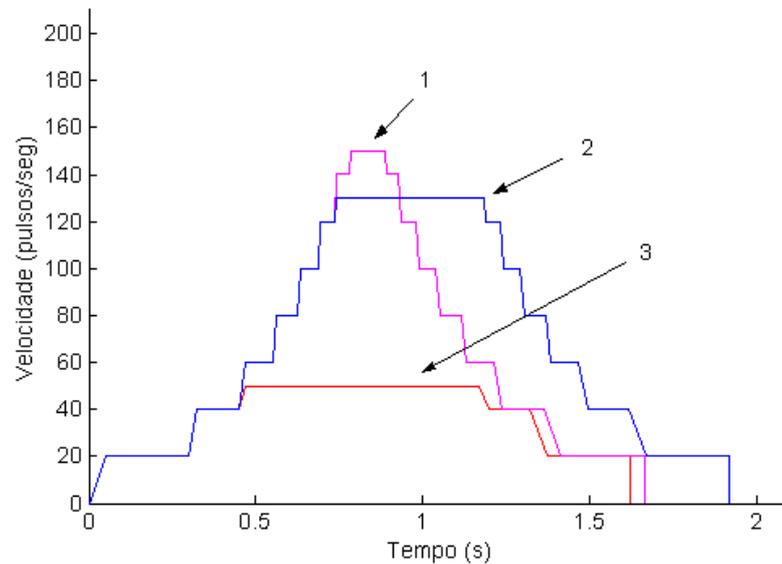


Figura 8.12 – Perfis de velocidade dos motores envolvidos no movimento.

Observando os gráficos que representam a variação da velocidade em cada um dos três motores que se movimentam conjuntamente para atingir um determinado ponto, percebe-se que cada um dos motores tem um tempo de duração do movimento diferente dos demais, mas para termos a trajetória planejada percorrida de forma adequada é necessário um sincronismo de movimentos, ou seja, que ao menos os motores realizem seus deslocamentos em um mesmo período de tempo.

Para conseguir fazer com que os motores se desloquem de forma contínua e em um mesmo intervalo de tempo decidiu-se por escolher como tempo padrão o tempo de deslocamento do motor que necessitar de um maior tempo para mover-se, e diminuir a aceleração e

desaceleração dos demais motores, de forma a retardar seus movimentos, conseqüentemente aumentando o tempo para a realização da trajetória do motor.

A equação 8.1 é utilizada no cálculo do tempo de duração do movimento de cada motor.

$$t = \sum_{i=1}^{N-1} [(\text{int} * (N + 1 - i)) * L * 2] + \text{int} * N_F \quad (8.1)$$

Onde: t = tempo de duração do movimento.

N = número total de degraus no perfil de velocidade.

int = tempo do menor intervalo entre pulsos.

L = largura do degrau.

N_F = quantidade de pulsos na velocidade de cruzeiro.

A mudança na aceleração/desaceleração é feita aumentando-se a largura do degrau do perfil de velocidade, especificamente falando, o que se faz é aumentar o tamanho do conjunto de pulsos que se envia ao motor em cada faixa de frequência, assim se diminui a velocidade máxima alcançada e conseqüentemente retarda-se o movimento.

9. PROGRAMA (PARTE 3)

9.1 Classe *Computador*

A classe *Computador* é responsável pela movimentação do robô. Ela contém os métodos que manipulam os dados de movimento, como também ponteiros para outras classes que servem como apoio para a minimização de ocorrência de falhas.

```
class Computador
{
private:
    Matriz *P_V;
    Porta_Paralela **porta;
public:
    Computador(Robo*,Placa_IOrobot**,char*);
    ~Computador();
    Ponto** ArquivoMove(Robo*,char*,int);
    void Planeja_Trajec(Robo*,Ponto**,int);
    void Planeja_Trajec_Dist(Robo*,Ponto**,int);
    void EnviaSinalPorta_IOrobot(Robo*,Placa_IOrobot**,int*,double*,char*,int);
    void EnviaSinalPorta_IOrobot_2(Robo*,Placa_IOrobot**,char*,int,Matriz*,double,
        double);
    void plotar(Robo*,char*);
};
```

O atributo *porta* é um ponteiro para diversos elementos do tipo *Porta_Paralela*, isso é necessário quando tivermos mais que três articulações, porque considerando o tipo de placa que possuímos, cada placa, conseqüentemente cada porta paralela, controla até três motores, então quando tivermos, por exemplo, sete articulações, serão necessárias três portas, duas portas para que cada uma controle três motores de passo e uma terceira porta paralela conecte-se ao sétimo motor.

O atributo *P_V* que é um ponteiro para a classe *Matriz*, armazena o número de pulsos e a velocidade máxima em cada motor para realizar a trajetória planejada. Em (9.1) as colunas ímpares representam o número de pulsos e as colunas pares a frequência máxima, e cada linha da matriz é um ponto a ser atingido.

$$\begin{bmatrix} 10 & 50 & 30 & 80 & 27 & 90 \\ -12 & 40 & 30 & 80 & 70 & 120 \end{bmatrix} \quad (9.1)$$

A primeira linha em (9.1) tem as ordens necessárias para se atingir o primeiro ponto. O motor 1 deve girar de 10 pulsos ou 10 passos a uma velocidade máxima de 50 pulsos/seg, o motor 2 gira 30 pulsos à uma frequência máxima de 80 pulsos/seg e o motor 3 gira 27 pulsos à 90 pulsos/seg. A linha seguinte possui as ordens para se atingir o segundo ponto da trajetória, e assim sucessivamente. O sinal negativo no motor 1 significa que a rotação ocorrerá no sentido oposto.

O método *ArquivoMove* lê um arquivo de entrada de dados com extensão “*txt*”, este arquivo possui dois comandos:

desloca: a) O valor angular, em graus, do deslocamento da junta para uma configuração qualquer de manipulador; ou b) As coordenadas X, Y e Z do ponto que se quer alcançar.

velocidade: Frequência máxima a ser atingida pelo motor de passo. Essa frequência tem como unidade dimensional pulsos/segundo.

Planeja_Trajec faz o planejamento da trajetória a ser percorrida pelo manipulador utilizando o método da mínima área (método apresentado anteriormente) utilizando como trajetória ideal os pontos lidos em *ArquivoMove*. *Planeja_Trajec_Dist* aplica o método da mínima distância para planejar a trajetória. Para não ficarem dúvidas, ambos os métodos descritos fazem “apenas” o planejamento da trajetória, encontram os deslocamentos ótimos para as juntas atingirem determinadas posições, mas não têm participação na questão de sincronismo entre os motores e o controle das frequências.

Os métodos *EnviaSinalPorta_Iorobot* e *EnviaSinalPorta_Iorobot_2* são responsáveis por enviar os pulsos necessários à classe *Porta_Paralela*. O primeiro deles envia os pulsos aos motores de forma seqüencial, o motor 1 recebe seus pulsos, em seguida o motor 2 e assim sucessivamente sem sincronismo entre os motores. Enquanto *EnviaSinalPorta_Iorobot_2* envia à *Porta_Paralela* e aos motores o sinal de forma sincronizada, baseando-se para isso na matriz montada pela classe *Velocidade* que será discutida mais adiante.

plotar é um método responsável por gerar um arquivo de saída de extensão “.txt” com as posições atingidas pelas juntas ao longo do percurso, e este arquivo é lido e apresentado em movimentos no software *Matlab* (Hanselman, 1999). Esta visualização é necessária para poder-se conferir como o manipulador se desloca.

Ainda referindo-se a trajetória, já foi dito que para o caso do manipulador com configuração antropomórfica o arquivo de entrada pode conter as coordenadas dos pontos que idealmente devem ser alcançados, mas por quê este tipo de dado só é aceito para esta configuração? O fato é que os motores trabalham exclusivamente com ângulos, então se conhecêssemos os valores destes ângulos, facilmente encontram-se os valores das coordenadas X, Y e Z. A este processo é dado o nome de *problema cinemático direto*, que pela modelagem matemática aplicada, Denavit-Hartenberg, o procedimento e as matrizes serão as mesmas para qualquer configuração. Porém para os casos mais gerais quando o dado de entrada é a coordenada do ponto a ser alcançado e tem-se que descobrir qual a combinação de ângulos que posiciona a extremidade do manipulador no ponto desejado, temos o *problema cinemático inverso*. (Sciavicco, 2001) (Lucas, 2000)

O problema cinemático inverso se caracteriza pelo fato de que conhecidos os valores para X, Y e Z, se quer descobrir os valores das variáveis das juntas, θ_i e/ou d_i . Normalmente este tipo de problema recai em se resolver um sistema de equações não-lineares, para isso utilizam-se métodos numéricos que podem trazer problemas, como também devido à redundância (mais de uma combinação de ângulos de junta resolvem o problema) pode apresentar resultados incoerentes. Por estes problemas a forma mais tradicional de se resolver problemas cinemáticos inversos é, caso seja possível, utilizando a solução analítica, isto significa encontrar equações do tipo $\theta=f(X,Y,Z)$, onde de forma direta conhecendo os valores de X, Y e Z encontram-se os valores dos vários θ .

Voltando ao programa tema deste trabalho, para podermos ter coordenadas de pontos utilizadas como dados de entrada para todos os tipos de configuração necessitaria-se ter as equações inversas de todas as configurações. Como isto é praticamente impossível, pois não é possível se imaginar todas as configurações possíveis adotadas na construção de um manipulador, embora muitas das equações sejam facilmente encontradas na literatura, no programa desenvolvido foram introduzidas apenas as equações para o problema inverso da configuração antropomórfica (equações 9.2). Porém caso seja de interesse do usuário, existe a liberdade de serem substituídas estas equações referentes à configuração antropomórfica por outras equações cinemáticas inversas de diferentes configurações.

$$\begin{aligned}
\theta_1 &= a \tan 2(p_y, p_x) \\
\theta_3 &= a \tan 2(s_3, c_3) \\
c_3 &= \frac{p_x^2 + p_y^2 + p_z^2 - a_2^2 - a_3^2}{2 * a_2 * a_3} \\
s_3 &= \pm \sqrt{1 - c_3^2} \\
\theta_2 &= a \tan 2(s_2, c_2) \\
s_2 &= \frac{(a_2 + a_3 * c_3) * p_z - a_3 * s_3 * \sqrt{p_x^2 + p_y^2}}{p_x^2 + p_y^2 + p_z^2} \\
c_2 &= \frac{(a_2 + a_3 * c_3) * \sqrt{p_x^2 + p_y^2} - a_3 * s_3 * p_z}{p_x^2 + p_y^2 + p_z^2}
\end{aligned} \tag{9.2}$$

$\theta_1, \theta_2, \theta_3$ – ângulos a serem calculados.

p_x, p_y, p_z – coordenadas do ponto que se quer alcançar.

$atan2$ – função (Matlab ou C++) que retorna o valor do arco cuja tangente seja igual ao valor do parâmetro, levando em consideração o quadrante do ângulo no círculo trigonométrico.

s_i, c_i – valores do seno e cosseno do ângulo θ_i .

a_1, a_2, a_3 – comprimento dos eixos.

9.2 Classe *Estrutura_Dados*

A *Estrutura_Dados* foi criada para ser usada no momento em que se aplicar o método das mínimas áreas, porque neste método se precisa de alguma forma organizar pontos em uma determinada ordem e que estes mesmos pontos sejam acessados segundo algum tipo de critério. Utiliza-se uma estrutura de grafo com pesos. Desta forma a classe aqui explanada foi adotada como uma primeira alternativa.

```

class Estrutura_Dados
{
private:
    Ponto ***pt_possiv;
    int *caminho;
    int *better;
    int nelem;
public:
    Estrutura_Dados(int);
    ~Estrutura_Dados();
    void SetNivel(Ponto**);
    Ponto** PercorreRepete(Ponto**);
    double Erro_TriangEspaço(Ponto**);
};

```

pt_possiv é um ponteiro para uma estrutura semelhante a uma matriz, formada por apontadores para elementos do tipo *Ponto*, $caminho$ contém o endereço dos pontos que estão em análise naquele momento e $better$ guarda o melhor caminho entre todos analisados até aquele momento.

O construtor de *Estrutura_Dados* recebe um número inteiro como parâmetro, este número indica o número máximo de pontos que rodeiam o ponto que se deseja alcançar. No caso da configuração antropomórfica este número será oito.

SetNivel acrescenta uma linha a estrutura de dados criada. Cada uma destas linhas contém ponteiros para os pontos vizinhos ao ponto que se quer alcançar.

O método *PercorreRepete* é responsável por percorrer todos os caminhos possíveis (figura 9.1), e ao final apresentar qual teve a melhor performance, em nosso caso o menor erro. Já *Erro_TriangEspaço* é o método que calcula o erro, ele divide o quadrilátero passado como parâmetro em dois triângulos e calcula a área de cada um destes triângulos utilizando produto vetorial e em seguida os soma para obter a área total do quadrilátero.

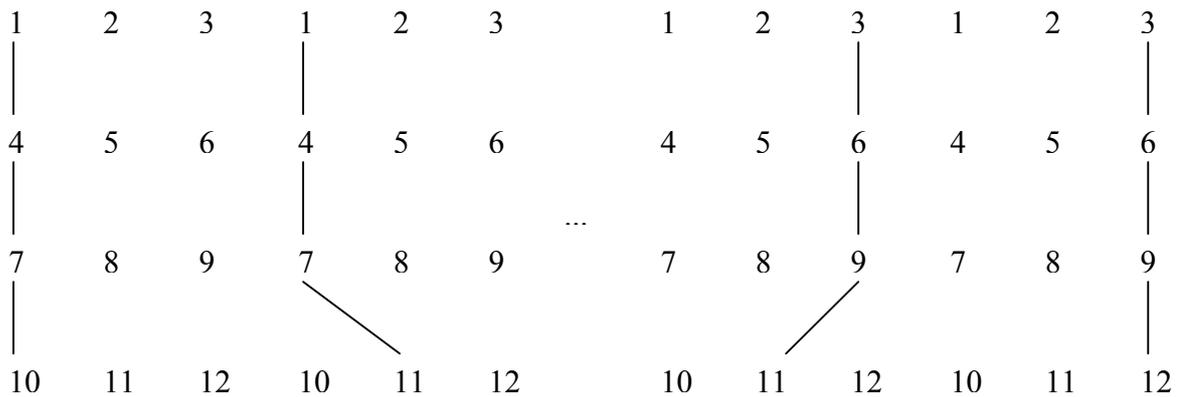


Figura 9.1 – Representação do funcionamento da estrutura de dados montada para ser utilizada com o método das mínimas áreas. Os caminhos possíveis vão sendo percorridos e ao final se classifica o melhor.

9.3 Classe *Velocidade*

Quando se trabalha com frequência de envio de uma série de pulsos, se utiliza um laço de repetição para fazê-lo. As linhas da programação apresentadas a seguir são uma demonstração dessa idéia.

```
for(int i=1;i<=20;i++)
    _outp(0x378,2);
```

onde: *_outp* – comando da biblioteca *DOS.h* que serve para enviar sinais à porta paralela.
0x378 – endereço da porta paralela.

2 - número decimal que representa, em binário, qual pino da porta paralela endereçada deve ser acionado. Maiores detalhes sobre endereçamento e acionamento de pinos foram tratados no tópico relativo à porta paralela.

O comando acima acionará o pino *DI* vinte vezes, todavia não terá efeito algum, pois não adianta tentar ativar um pino que já está ativo, é o mesmo que tentar acender uma lâmpada por vinte vezes sem apagá-la uma única vez. Desta maneira para funcionar de forma correta o comando deve ser:

```
for(int i=1;i<=20;i++)
{
    _outp(0x378,2);
    _outp(0x378,0);
}
```

Agora ao executar os comandos o pino passará a ser ativado e desativado por vinte vezes, assim espera-se que ele funcione, porém existe ainda um empecilho. A frequência do computador é altíssima, algo em torno de algumas centenas de megahertz. Assim suponha que houvesse um led conectado ao pino com o qual se está trabalhando. Quando se ativa o pino o led acende e ao se desativar o pino o led apaga. Com a frequência agindo livremente, o led piscará tão rápido que nem seremos capazes de observar o piscar da luz. Então, para completar:

```
for(int i=1;i<=20;i++)
{ _outp(0x378,2);
  _sleep(200);
  _outp(0x378,0);
  _sleep(200); }
```

O comando `_sleep` adicionado ao laço tem por fim parar o processador, e conseqüentemente o andamento do programa por um determinado tempo. Este comando recebe como parâmetro um valor numérico que acarreta uma pausa no trabalho do processador por uma quantidade de milésimos de segundo ou milissegundos. Com essa explicação fica bem inteligível o que esse trecho de programa faz:

- 1 – acende o led.
- 2 – pausa o programa por 200 milissegundos, enquanto isso o led continua aceso.
- 3 – apaga o led.
- 4 – espera 200 milissegundos e recomeça o ciclo.

Os passos descritos serão repetidos 20 (vinte) vezes. Então, agora pode-se observar claramente o led piscando. Mudando-se o valor passado como parâmetro em `_sleep` muda-se a frequência com que o led pisca. No caso de ser aumentado o tempo de espera, o led pisca mais lentamente e caso se diminua o tempo o pisca-pisca aumenta de frequência.

Toda esta explanação tem como principal finalidade fazer com que o leitor entenda como se faz a mudança da velocidade nos motores de passo. A mudança de velocidade funciona para os motores assim como o piscar funciona para o led, assim quando se quer incrementar a velocidade de rotação do motor basta aumentar a frequência com a qual os pulsos são enviados à porta paralela e para diminuir a velocidade diminui-se a frequência, em outras palavras, aumenta-se o intervalo entre os pulsos.

Para deixar claro, da mesma maneira como o pisca-pisca do led falha quando temos altas frequências no envio dos pulsos, os motores de passo também têm problemas quando o intervalo entre pulsos é muito pequeno, o motor acaba “perdendo passo”.

Apresenta-se agora a classe *Velocidade*:

```
class Velocidade
{
private:
  double intmin;
public:
  Matriz *Planeja_Veloc(double*,int);
  double tempo(int,int,int,double,int);
  Matriz *Matriz_Pulsos(int[],int[],int[],int,int,int[]);
};
```

O atributo *intmin* representa o tempo mínimo, em milissegundos, que durará o intervalo entre pulsos, este é o menor intervalo de tempo em que não existe perda de pulso, conseqüentemente o inverso deste valor é a frequência mínima com a qual o motor trabalha de maneira confiável.

O método *Planeja_Veloc* se divide em duas partes,. Na primeira parte ele calcula os principais parâmetros do perfil de velocidade (tópico 8.2.1), tais como número de degraus do

perfil e tempo de duração do movimento. Isto é feito para cada articulação do manipulador e para o percurso de um ponto ao ponto seguinte. Com os dados calculados vê-se qual movimento tem maior tempo de duração. Em seguida faz-se uma comparação entre os tempos de duração de cada movimento e, no caso de algum deles estar acima de um percentual considerado razoável, o método parte para a segunda parte que é a tarefa de retardar um pouco o movimento dos demais motores, de forma a tentar igualarem-se os tempos de duração do movimento em todas as articulações, assim se terá um sincronismo de movimentos. O retardamento dos movimentos se faz aumentando a largura dos degraus no perfil de velocidade.

Nem sempre é possível sincronizar o movimento de todos os motores. Isto acontece porque as vezes o movimento à mais baixa velocidade ocorre em um tempo inferior ao deslocamento de um outro motor em uma velocidade mais alta. Estes casos surgem quando as quantidades de pulsos para o deslocamento dos motores são muito diferentes.

tempo é responsável pelo cálculo do tempo de duração do movimento (equação 8.1).

O método *Matriz_Pulsos* serve para montar a matriz dos pulsos que vão para os motores. Para a compreensão da maneira como este método funciona dois pontos devem ser levados em consideração.

1 – A existência de um tempo de intervalo mínimo entre os pulsos.

2 – Os demais intervalos, que referem-se a outras frequências menores que a anterior, devem ser múltiplos do intervalo mínimo.

Começando com um exemplo de um motor de passo que possui 3 (três) valores de velocidade e vai receber cinco pulsos. Supondo que os pulsos serão dados na frequência mínima, que é o máximo intervalo de tempo entre as três velocidades, a matriz montada representa onde aparece “0” não tem pulsos e onde aparece “1” enviar pulso. Cada coluna desta matriz é lida e em seguida o programa fica em pausa por um tempo igual ao tempo de intervalo mínimo, sendo assim cada pulso será enviado a um intervalo igual a três vezes o intervalo mínimo, neste caso cinco pulsos.

```
0 0 1 0 0 1 0 0 1 0 0 1 0 0 1
```

Vamos supor agora enviar 5 (cinco) passos, sendo um destes passos em um intervalo menor. A matriz para este motor fica:

```
0 0 1 0 0 1 0 1 0 0 1 0 0 1
```

A diferença de tempo entre o segundo e o terceiro passo é de duas vezes o intervalo mínimo entre pulsos. Sabendo que cada coluna é lida em um certo tempo o *movimento 1* dura quinze vezes esse intervalo, enquanto o *movimento 2* tem duração de quatorze vezes o intervalo mínimo.

Agora vamos ao caso de termos dois motores de passo tendo de realizar cinco passos cada e ambos na velocidade mínima. A diferença deste para os exemplos anteriores é que agora a matriz passa a ter duas linhas.

```
0 0 1 0 0 1 0 0 1 0 0 1 0 0 1
0 0 1 0 0 1 0 0 1 0 0 1 0 0 1
```

Cada uma das colunas continuará sendo lida em um determinado período de tempo, a diferença é que agora quando existe pulso na linha 1 será o motor 1 quem se moverá, e o motor 2 se moverá quando houver pulso na segunda linha. No caso acima os motores sempre se moverão ao mesmo tempo.

O último caso a ser mostrado é quando os motores têm intervalos diferentes, caso mais geral.

```
0 0 1 0 0 1 0 0 1 0 0 1 0 0 1
0 0 1 0 0 1 0 1 0 0 1 0 0 1 0
```

Para este tipo de matriz o procedimento é o mesmo do caso anterior, alguns ocorrerão em intervalos diferentes, porém o movimento parecerá contínuo, pois a diferença de tempo entre pulsos é da ordem do intervalo mínimo que já é um valor bem baixo.

A matriz de pulsos mais geral, quando se tem um maior número de frequências possíveis de se trabalhar ou quando o número de pulsos for maior, terá um tamanho bem maior do que as apresentadas nestes exemplos. Devido a aceleração que acontece no motor, o espaçamento entre os pulsos vai diminuindo, até se alcançar a velocidade máxima de movimento ou velocidade de cruzeiro. Em seguida, devido à desaceleração, os pulsos voltam a ficar cada vez mais espaçados.

10. APLICAÇÃO

Neste capítulo serão apresentados dois exemplos do funcionamento do programa passo a passo. No primeiro exemplo os dados de entrada são de um robô de configuração esférica, o qual não se faz o planejamento da trajetória e o segundo exemplo, um robô de configuração antropomórfica com o planejamento da trajetória.

10.1 Manipulador de Configuração Esférica

Arquivo de entrada de dados:

junta	1	0	0	0
z		0	1	0
motor	3.6			
acionador	4			
junta	1	0	10	0
z		0	0	1
motor	7.5			
acionador	6			
junta	2	0	10	0
z		1	0	0
motor	1.8			
acionador	4			
junta	1	10	10	0

Onde:

junta: Os dados que seguem referem-se a *Junta*. O primeiro número indica o tipo de junta: 1 – junta rotacional e 2 – junta prismática. Os últimos três números são as coordenadas X,Y e Z da junta antes de iniciar o movimento.

z: Direção do eixo “z” da junta.

acionador: O número representa o valor da redução que o acionador aplica ao movimento do motor.

motor: Número representa a resolução em graus do motor de passo.

Este arquivo é passado como parâmetro para a classe *Robot* que cria o robô computacionalmente, fazendo todas as conexões entre junta – motor – acionador – barra necessárias para construí-lo. Também são calculados todos os parâmetros (*Denavit-Hartenberg*) para a montagem das matrizes de transformação homogênea.

Em seguida dependendo do número de articulações são criadas *Placa_Iorobot* suficiente para ligar-se à todas. Neste exemplo o robô possui três articulações, então, só é necessária uma placa.

Depois a classe *Computador* lê um arquivo (10.1) que contém a angulação que cada junta deve deslocar-se para realizar o movimento.

desloca	45.9	0	71.4	
desloca	5.2	30	-14	(10.1)
desloca	24	-30.2	52.4	

Em *Computador* o valor do ângulo de deslocamento é enviado direto para a *Porta_Paralela* um motor a cada vez e se faz um arredondamento para o movimento ser exato, por exemplo, o usuário pode solicitar que determinada junta rotacione 42° e esta junta, já considerando a

resolução do motor e a redução aplicada pelo acionador, só se desloque de 5° , assim a junta ou vai para 40° ou vai para 45° , como é feito apenas um arredondamento direto, neste caso, a junta se desloca para 40° que está mais próximo de 42° . Este procedimento é repetido para todos os motores e para todas as linhas de comando até o fim do arquivo.

Fluxograma do programa na figura 10.1.

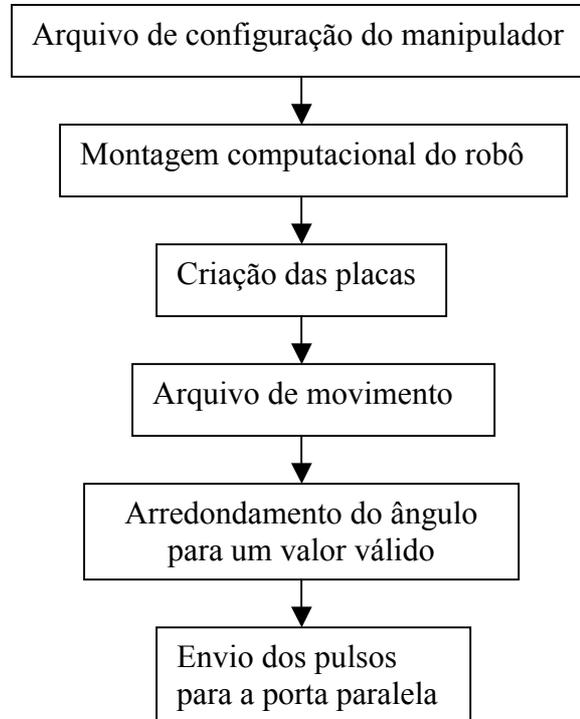


Figura 10.1 – Fluxograma de funcionamento do programa para os casos no qual não se precisa, ou não se quer, o planejamento da trajetória.

10.2 Manipulador de Configuração Antropomórfica

Os primeiros procedimentos são os mesmos do exemplo anterior. Leitura de um arquivo de configuração (10.2), montagem do robô, criação das placas, que neste caso será uma única também, pois temos apenas três articulações.

junta	1	0	0	0
z		0	1	0
motor	10			
acionador	1			
junta	1	0	0	10
z		0	0	1
motor	10			
acionador	1			
junta	1	5	0	10
z		0	0	1
motor	10			
acionador	1			
junta	1	10	0	10

(10.2)

O arquivo de movimento é diferenciado, pois aqui os dados de entrada no comando *desloca* são as coordenadas do ponto que se quer alcançar, além dos valores das velocidades máximas para cada motor com as quais se deseja que ele se movimente.

desloca	4.59	0	7.14
desloca	5.64	0	8.95
velocidade	50	60	50
desloca	5.8	0	12.55
desloca	4.24	0	15.24

Computador tendo as coordenadas dos pontos utiliza-se das equações inversas para chegar aos valores ideais das variáveis das juntas. Em seguida utiliza-se um método para planejar a trajetória, tem-se dois: mínima distância ou mínima área. Para finalizar depois de ter chegado à trajetória ótima se faz o planejamento da velocidade e se monta a matriz de pulsos.

Tendo a matriz de pulsos montada se passa a percorrê-la coluna à coluna e mandar os respectivos pulsos à *Porta Paralela*.

A figura 10.2 apresenta o fluxograma do programa:

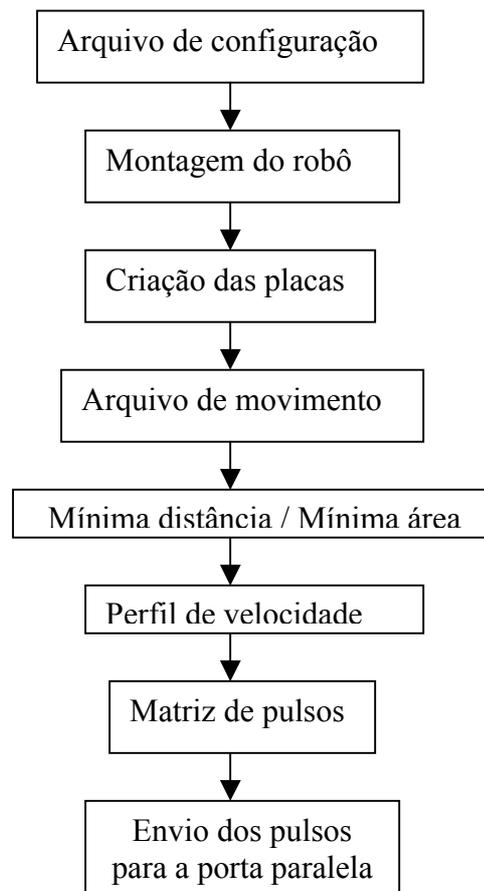


Figura 10.2 – Fluxograma do movimento com planejamento de trajetória.

11. TRABALHOS FUTUROS

Qualquer bom trabalho de pesquisa não se completa nele mesmo. Na verdade acredita-se que nunca esteja completo, pois pode sempre ser aperfeiçoado ou adaptado a novas situações. Crê-se que o trabalho apresentado se enquadra nesta modalidade, sobretudo lembrando que este foi um primeiro trabalho entre muitos que virão pela frente, e também porque, desde o início, o trabalho foi visto como um trabalho de base.

Entre diversas idéias acredita-se que as mais promissoras sejam desenvolver a simulação dinâmica, onde seriam estudados os esforços envolvidos nos movimentos de diferentes manipuladores. A outra boa idéia é acoplar sensores no sistema de forma que se conseguisse interagir com o ambiente em tempo real.

Estas são as idéias de expansão do programa ou da tecnologia que o envolve, mas existem outras sugestões que visam melhorar o programa sem alterá-lo substancialmente. Adaptar programa para trabalhar em cadeia fechada, do tipo mecanismos de barras. Construir uma interface gráfica para o programa daria a ele um novo charme, ou melhor, uma forma de atrair a atenção das pessoas mais facilmente, inclusive facilitando também a obtenção de resultados para verificação do desempenho do mesmo.

Para corrigir determinadas “falhas” do programa faz-se duas recomendações: 1) Utilizar um sistema de busca no método das mínimas áreas mais eficiente que o aplicado. 2) Otimizar algoritmo de sincronização de movimentos, para melhorar os resultados obtidos com a idéia aqui adotada de variar a aceleração/desaceleração.

Uma última modificação que trará grande versatilidade ao programa está na mudança da classe *Junta*, fazendo-se com que um elemento deste tipo possa apontar para mais de uma *Junta* para formar a cadeia (figura 11.1). Poderia-se simular o controle, por exemplo, de uma mão, onde uma junta está conectada em uma de suas extremidades a várias outras. Uma estrutura que poderia ser usada para este fim é a estrutura de grafos orientados, onde cada articulação representaria um nó no grafo (figura 11.2).

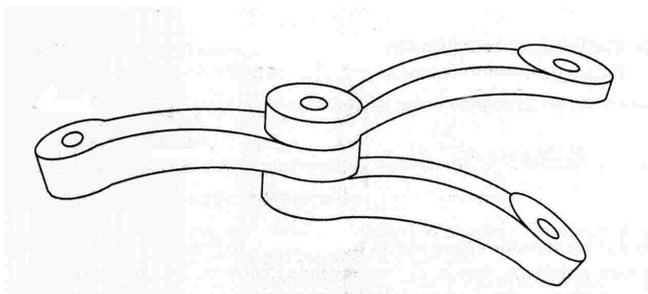


Figura 11.1 – Junta que possui uma barra de entrada e duas de saída (Sciavico; 2001)

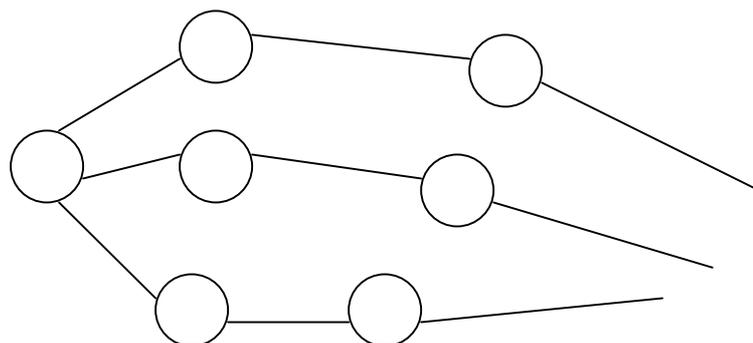


Figura 11.2 – Grafo utilizado para representar uma mão.

12. CONCLUSÃO

O trabalho desenvolvido alcançou muitas das metas idealizadas no início e certamente servirá como uma boa base para futuros trabalhos.

O programa desenvolvido consegue, hoje, simular o movimento de qualquer configuração de manipulador de cadeia aberta usando a cinemática direta ou a cinemática inversa, além de terem sido desenvolvidas idéias de otimização de trajetória, que mostraram-se válidas e bem úteis.

Além do mais a robótica e a automação são áreas que cresceram bastante nas últimas décadas e sempre é válido expandir conhecimentos e transpor obstáculos rumo ao desenvolvimento de novas tecnologias, desde que estas tecnologias tragam algo de positivo para o mundo. Baseado nesta idéia de desenvolvimento tecnológico e olhando para o nosso país e, sobretudo para a nossa região Nordeste, percebe-se o quanto podemos ganhar desenvolvendo pesquisa nestas áreas.

Evidentemente que robótica e automação como qualquer campo de pesquisa tem suas divisões que se desenvolvem em paralelo. Neste trabalho falamos essencialmente de modelagem matemática de manipuladores, mas existem muitas outras áreas tão interessantes quanto, e conseqüentemente promissoras. Entre elas tem-se inteligência artificial, robôs móveis, robôs especiais, e em automação as células automatizadas de fabricação.

REFERÊNCIAS BIBLIOGRÁFICAS

- Adorocinema, <http://adorocinema.cidadeinternet.com.br/filmes/homem-bicentenario/homem-bicentenario.htm>, nov/2003.
- Alves, João Bosco M., *Controle de Robô*, Cartografia TDA, 1985.
- Classics Movies, <http://classics.www5.50megs.com>, nov/2003.
- Curso de Linguagem Java “Versão Gratuita”. www.tarcisiolopes.com, dez/2003.
- Edson Jedi, <http://www.geocities.com/edsonstarwars>, nov/2003.
- Engelberger, Joseph F. *Historical Perspective and role in automation*. In: Handbook of Industrial Robotics. Second Edition, United State of América, John Wiley & Sons, Inc., 1999.
- Ferreira, Marco T. *Robótica*. Florianópolis, SENAI / CTAI , 1998.
- Freitas, Júlio César de Almeida. *Ensinando o Robô*. Em: Mecatrônica Atual, Editora Saber Ltda, Setembro/2002.
- Gong, Chunhe. *Nongeometric Error Identification and Compensantion for Robotic System by Inverse Calibration*. In: International Journal of Machine Tools & Manufacture. 2000, pp 2119-2137.
- Groover, Mikell P. *Robótica – Tecnologia e Programação*. São Paulo, McGraw – Hill, 1988.
- Hall, Allen S. *Elementos Orgânicos de Máquinas*. Mc Graw – Hill do Brasil, 1990.
- Hanselman, Duane. *MATLAB 5 – Versão do usuário*. São Paulo, MAKRON Books, 1999.
- Idoeta, Ivan V. *Elementos de Eletrônica Digital*. Érica Editora, 1988.
- IORobotics, <http://www.iorobotics.com>, jul/2002.
- Jamsa, Kris. *Aprendendo C++*. São Paulo, MAKRON Books, 1999.
- Jeon, J. W. *A Generalized Approach for the Accceleration and Deceleration of Industrial Robots and CNC Machine Tools*. In: IEEE Transactions on Industrial Electronics. Vol 47, 2000.
- Juttler, B. and Wagner, M. G. *Computer - Aided Design with Spatial Rational B – spline Motions*. In: Journal of Mechanical Design, June/1996, pp 193-201.
- Laszlo, Michael J. *Computational Geometry and Computer graphics in C++*. Prentice – Hall. Inc, 1996.
- Kao, Imin & Gong, Chunhe; *Robot – Based Computer – Integrated Manufacturing as Applied in Manufacturing Automation*; Robotic & Computer – Integrated Manufacturing; 1997; p. 159.

- Lucas, Stuart R. *Real-Time Solution of the Inverse Kinematic-Rate Problem*. In: The International Journal of Robotics Research. Vol 19, Sage Publications, 2000, pp 1236-1244.
- Mecatrônica Fácil*; São Paulo, Editora Saber, nov/2001, p 13.
- Pazos, Fernando A. *Robôs Manipuladores – 2ª Parte*. Em: Mecatrônica Atual, Editora Saber Ltda, Abril/2002.
- Porta Paralela*. www.rogercom.com.br, 2002.
- Ramalho Júnior, Francisco. *Os fundamentos da Física 3 – Eletricidade*. Editora Moderna, 1983.
- Ramos Filho, F.B. *Motor de Passo, Características, Aplicações e Controle para Motor de Passo*. Recife, 2001. (trabalho avulso).
- Reis, Genésio Lima. *Geometria Analítica*. Editora LTC, 1994.
- Sciavicco, L. *Modeling and Control of Robot Manipulators*. 2nd printing, Great Britain, Springer Limited, 2001, chapter 1.
- Stevens, Al. *Aprenda Você Mesmo... C++*. Rio de Janeiro, LTC – Livros Técnicos e Científicos, 1991.
- Wagner, Michael G. *Planar Rational B-spline Motion*. In: Computer – Aided Desing, Vol 27, February/1995, pp 129-137.
- Warnecke, H. J. *Manipulator Design*. In: Handbook of Industrial Robotics. Second Edition, United State of América, John Wiley & Sons, Inc., 1999.

ANEXO A

1. Porta Paralela

O microcomputador pode se comunicar ou interagir com o exterior, de diversas formas, uma delas é através da porta paralela, conhecida também como saída da impressora por servir para conectar este tipo de periférico.

A porta paralela tem esse nome porque pode enviar e receber diversos dados simultaneamente (de forma paralela). Ela é composta basicamente por 25 pinos (figura A.1), sendo alguns destes de entrada de sinal, alguns de saída de sinal, alguns são “terra” e outros programáveis (entrada ou saída), e todos eles funcionam de forma binária, ou seja, ou estão ligados ou estão desligados (1 ou 0).

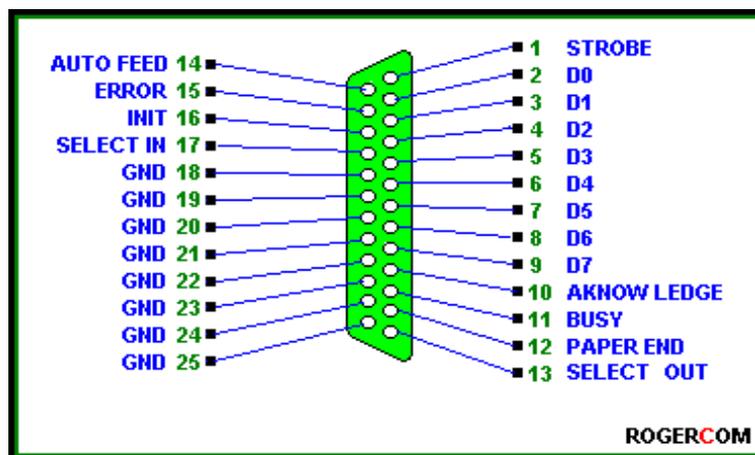


Figura A.1 – Pinagem da porta paralela. (Porta Paralela, jan/2002)

Toda linguagem de programação possui funções pré-definidas ou bibliotecas específicas para se trabalhar com a porta paralela, e através dessas funções consegue-se comunicação com equipamentos externos.

Neste trabalho se fez uso dos pinos *D0* à *D7*, que se faz referência através do endereço *378*, conectando quatro desses pinos ao primeiro motor de passo e os outros quatro pinos a um segundo motor de passo. Os pinos *Strobe*, *Auto Feed*, *Init* e *Select In* (o primeiro e os dois últimos trabalham de maneira inversa, quando se envia sinal a saída é desligada e quando se corta o sinal a saída é ativada), cujo endereço de referência é *37A*, são ligados ao terceiro motor. (Porta Paralela, 2002)

2. Circuito Eletrônico

Para se fazer a conexão entre a porta paralela de um microcomputador e um equipamento externo (impressora, monitor, led, etc) é necessário um circuito eletrônico para controlar/codificar o sinal vindo do computador. Por questões de segurança e até de melhora no funcionamento um componente é essencial, o transistor. (Ramalho Junior, 1983; Idoeta, 1988)

Transistor é um componente eletrônico que funciona da seguinte forma (figura A.2): Ao receber uma determinada intensidade de corrente em sua base, ele “fecha” e permite a circulação de corrente entre o coletor e o emissor, assim o transistor funciona como uma chave a qual deixa passar corrente se for acionada e também pode fazer a ampliação do sinal, pois a corrente na saída (emissor ou coletor) pode ser bem maior que a corrente na entrada.

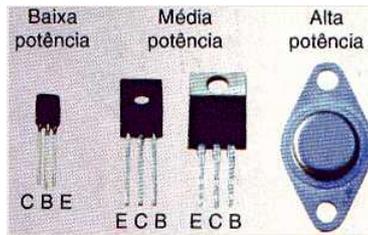


Figura A.2 – Transistores (Mecatrônica Fácil, 2001)

Fazendo uso de transistores pode-se controlar vários elementos que funcionam através de correntes elétricas, por exemplo, motores de passo. Ligando-se um transistor em cada uma das saídas do motor e enviando sinais alternadamente para os quatro transistores, faz-se o movimento do motor, mesmo daqueles que necessitam de altas correntes.

Na figura A.3 é mostrado um exemplo de circuito eletrônico para movimentar motores de passo.

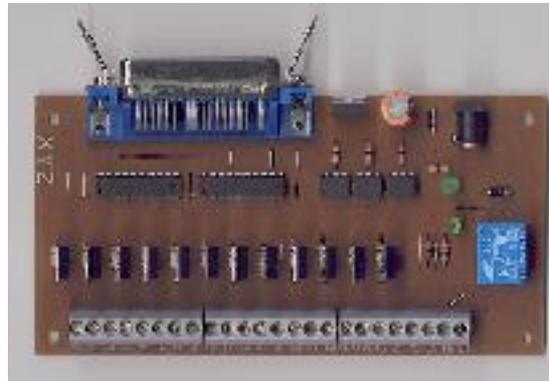


Figura A.3 – Placa para controle de motores de passo. (IORobotics, 2002)

ANEXO B

Segue o artigo apresentado no 2º Congresso Brasileiro de Engenharia de Fabricação (COBEF) realizado em maio/2003 na cidade de Uberlândia – MG, promovido pela Associação Brasileira de Ciências Mecânicas (ABCM); e publicado na revista *Máquinas e Metais*, Aranda Editora, nº 452 em setembro/2003.

PROJETO DE UMA GARRA MECÂNICA PARA UM MANIPULADOR INDUSTRIAL

Erwin Rommel

Universidade Federal de Pernambuco (UFPE), Centro de Tecnologia e Geociências,
Departamento de Engenharia Mecânica, Rua Acad. Hélio Ramos s/n, CEP: 50740-530

Hugo Leonardo Nunes dos Santos

Universidade Federal de Pernambuco (UFPE), Centro de Tecnologia e Geociências,
Departamento de Engenharia Mecânica, Rua Acad. Hélio Ramos s/n, CEP: 50740-530,
hugonunes@bol.com.br

Jarbas José Carneiro Bezerra de Jesus

Universidade Federal de Pernambuco (UFPE), Centro de Tecnologia e Geociências,
Departamento de Engenharia Mecânica, Rua Acad. Hélio Ramos s/n, CEP: 50740-530,
jarbas.jose@bol.com.br

Profª Drª Noemia Gomes de Mattos de Mesquita

Universidade Federal de Pernambuco (UFPE), Centro de Tecnologia e Geociências,
Departamento de Engenharia Mecânica, Rua Acad. Hélio Ramos s/n, CEP: 50740-530,
ngmm@npd.ufpe.br

Prof. PhD. Félix Christian Guimarães Santos

Universidade Federal de Pernambuco (UFPE), Centro de Tecnologia e Geociências,
Departamento de Engenharia Mecânica, Rua Acad. Hélio Ramos s/n, CEP: 50740-530

Resumo.

Este projeto visa o desenvolvimento de uma garra mecânica para um manipulador industrial que tem a finalidade de posicionar peças brutas a serem usinadas em um torno CNC. Este trabalho repetitivo, em geral, ainda é executado manualmente, exigindo do operador esforço físico, expondo-o a riscos de acidentes, além de ser executado num tempo relativamente alto. A execução do projeto passa pela construção de um protótipo de um manipulador industrial para melhor estudo dos movimentos a serem executados e para posterior construção do manipulador de maior porte com uma garra mecânica para posicionar peças em um torno CNC. As garras mecânicas têm várias configurações, que estão diretamente relacionadas com o grau de complexidade da tarefa a ser executada. A desenvolvida no trabalho pode manipular peças não só cilíndricas, mas também de formas variadas (cônicas, cilíndricas de diâmetros diferentes). Para a operação “de pega” de peças de formatos variados, a pesquisa realizada mostrou que as garras de três dedos são mais adequadas, pois elas podem apresentar independência entre os movimentos dos dedos. Utilizando um deles como apoio inferior (polegar), este dedo passa pela linha de ação do baricentro da peça, então os outros dois dedos superiores, equilibram a peça impedindo sua rotação. As seguintes características também foram analisadas nesse trabalho: precisão de posicionamento; tipo de acionamento; e tipos de mecanismo. O protótipo construído está sendo utilizado no estudo do controle e da precisão dos movimentos.

Palavras-chave: Célula de Manufatura, Usinagem, Automação, Robô Industrial, Garra Mecânica

1. INTRODUÇÃO

Robôs industriais e máquinas CNC têm características semelhantes, pois ambos operam através da tecnologia de comando numérico. Suas aplicações e modo de operações são freqüentemente planejados pelas mesmas pessoas. Um robô, no entanto, jamais substituirá a máquina ferramenta CNC. Ele atua, na verdade, como um suporte para aumentar o grau de automação.

Quando se fala em robôs, a primeira imagem que se tem em mente é geralmente a de um “humanóide” com duas pernas, dois braços, mãos, um corpo e, mais importante de tudo, com uma cabeça que possui olhos e ouvidos reproduzidos por fotocélulas e antenas. Esta imagem foi formada muito antes da aplicação destes “homens máquinas” e tem sido muito utilizada em ficção científica. As impressões iniciais de uma criatura superior aos seus rivais e subserviente apenas ao seu inventor persistem até hoje e esta visão dos robôs é em parte responsável pelas reações defensivas mostradas por muitos trabalhadores e sindicatos (Kief, H.B., 1992).

O fato é que, os setores industriais que não se automatizarem não vão conseguir acompanhar o desenvolvimento industrial. No nosso estado a necessidade desta automação foi mostrada em pesquisa conduzida pelo SENAI/PE alguns anos atrás.

Em contraste com essas impressões iniciais, o robô industrial de hoje tem um novo papel interativo apresentando as seguintes características: ajudante (não rival); máquina (não pessoa); programável (não inteligente); flexível (não inovador).

A noção de que algum dia todas as operações personalizadas de uma produção industrial será substituída por robôs ainda não é realidade. O ser humano faz uso de todas as suas funções sensoriais e ainda não é totalmente copiada pelos robôs que não têm nem consciência, nem criatividade, o que não os habilita a lidar com problemas não estruturados.

Um robô industrial pode ser definido como um manipulador multifuncional programável, designado a mover material, peça, ferramentas ou dispositivos especiais através de movimentos programáveis variados para realizar uma variedade de tarefas.

Robôs são cada vez mais utilizados na produção onde é exigido repetibilidade e precisão. A maioria deles, são simplesmente dispositivos de operação, chamados de unidades de “pegar” e “colocar” em outro lugar, para, por exemplo, carregar e descarregar máquinas.

Dispositivos de aplicação universal totalmente programáveis representam apenas 20% da população dos robôs. Pesquisas continuam sendo desenvolvidas para aperfeiçoar as funções de ver, sentir, falar e ouvir dos robôs.

Este trabalho tem como objetivo dar uma contribuição para um maior nível de automação do setor industrial nordestino, através da aplicação de um robô industrial na usinagem. Este trabalho visa também diminuir a interferência do operador com o torno CNC na usinagem de peças. Desta forma é possível diminuir os tempos mortos de usinagem e o esforço empregado pelo operador, aumentando assim a produção e, conseqüentemente, obtendo mais lucro.

O robô projetado, controlado por equipamentos programáveis, deve trabalhar junto a uma máquina CNC alimentando-a com a peça bruta a ser usinada, virando a peça, quando for necessário usina-la em diferentes posições, e retirando a peça usinada da máquina.

Para um bom rendimento é necessário analisar o movimento, incluindo suas acelerações e desacelerações, de modo a se ter um movimento contínuo. As características da aceleração são independentes da desaceleração, fazendo com que aí se tenha estudos diferentes, ao contrário do que se imagina (Jeon, J.W., Ha, Y.Y., 2000).

2. ESTADO DA ARTE

O estudo da robótica foi impulsionado pela ficção científica e pelas necessidades de conforto e lazer que o homem busca. Alguns dos desenvolvimentos iniciais de dispositivos automáticos auxiliaram esse estudo, embora, nem todos tratem diretamente da robótica.

Em épocas mais recentes, o comando numérico e o telecomando são duas tecnologias no desenvolvimento da robótica. O comando numérico apareceu em meados do século XX, mais especificamente nas décadas de 40 e 50. Os trabalhos iniciais de comando numérico baseiam-se nos estudos de John Parsons, através de cartões perfurados. Já no ano de 1946 o inventor americano G.C. Devol desenvolveu um dispositivo controlador que podia registrar sinais elétricos magneticamente e reproduzi-los para operar uma máquina mecânica. Em 1951, foi desenvolvido por Goertz e Bergsland um trabalho em teleoperadores (manipuladores por controle remoto) para manusear materiais radioativos. O inventor britânico C.W.Kenward desenvolveu em 1954 a patente para o projeto de robô. No ano de 1959 foi desenvolvido o primeiro robô comercial produzido pela Planet Corporation. A Stanford University em 1971 desenvolveu um pequeno braço de robô acionado eletricamente. Em 1975, o robô Sigma, da Olivetti, usado na operação de montagem, uma das primeiras aplicações de montagem efetivas da robótica. Em 1979 ocorreu o desenvolvimento do robô SCARA (Selective Compliance Arm for Robotic Assembly – Braço de Deslocamento Seletivo para Montagem Robotizada) na Universidade de Yamanashi, no Japão. No ano de 1981, foi criado o Robô de Acionamento Direto pela Universidade Carnegie-Melon. Este usava motores elétricos localizados nas juntas manipuladores, sem a transmissão mecânica usual (Groover, M.P. e Weiss, M., 1988).

Com relação aos estudos de garras mecânicas e manipuladores industriais, vale destacar os estudos dos aspectos da cinemática de garras mecânicas de múltiplos dedos feitos por Mason e Salisbury em 1985, (DasGupta, A., Hatwal, H., 1998).

O MIT (Massachusetts Institute of Technology) desenvolveu uma garra mecânica (Utah/MIT) que foi construída com dedos acionados por tendões e com sensores de toque. Este estudo foi iniciado por Jacobson em 1984. Em 1990, foi desenvolvida uma garra antropomórfica de cinco dedos, a Belgrade/USC hand, num estudo desenvolvido por Bekey, (Matsuoka, Y., 1997).

Em 1992 destaca-se o estudo de Young Park e Starr na Síntese de pega de objetos poligonais usando uma garra de três dedos, pela Universidade do Novo México, (Park, Y; Starr, G., 1992)

Em 1997, Yoky Matsuoka apresenta um estudo de um mecanismo para uma garra mecânica antropomórfica, pelo MIT, (Matsuoka, Y., 1997).

Os robôs industriais, geralmente, apresentam um custo elevado para a implementação numa linha de fabricação. Isso faz com que empresas de menor porte não tenham a possibilidade de fazer esta melhoria em suas linhas de produção. Este trabalho tem como objetivo de projetar e construir um robô que tenha um custo de produção menor que os robôs industriais existentes no mercado, assim dando uma contribuição ao ramo da automação industrial na região.

3. PROJETO DE UMA CÉLULA DE MANUFATURA

Na figura 1 é mostrada a seqüência do funcionamento do robô de médio porte, numa célula de manufatura. O processo começa a partir do estoque, que fica em uma parte mais elevada, contendo várias peças brutas a serem usinadas. Quando a tampa do estoque é aberta apenas uma peça desce a rampa, parando na posição 1. A partir daí o robô sai da posição neutra e vai pegar a peça nesta posição. Depois de fechada a garra, a peça é levada para a posição 2, no torno. Quando a peça está fixa nas castanhas, o robô volta para a posição neutra, a porta do torno é fechada e se inicia a usinagem. Ao término da usinagem a porta do torno se abre, o robô vai pegar a peça na posição 2 e a leva até a posição 3, onde são armazenadas as peças usinadas. Para que a tampa e a porta possam ser abertas e fechados são necessários atuadores, que são ativados e desativados pelo CLP. O computador movimentará o robô de acordo com os sinais recebidos do CLP, e para o CLP enviar estes sinais ele necessita de informações que são recebidas através dos sensores e chaves. Os sensores, chaves e atuadores

são (ver figura 1): um sensor fotoelétrico de interrupção de feixe (sensor1); um sensor de sobrecarga (sensor 2) na garra; uma chave limite na porta do torno; uma chave seletora (peça fixa/solta); um botão de início de usinagem (start); um motor de corrente contínua para a tampa do estoque e um para a porta do torno.

Os movimentos do robô são controlados pelo computador, através de sinais elétricos enviados, em determinados momentos, pelo CLP. Estes sinais estão sendo gerados pelo LADSIM v2.6.2 (simulador de CLP) da Bytronic International Ltda., adquirido no site www.bytronic.co.uk.

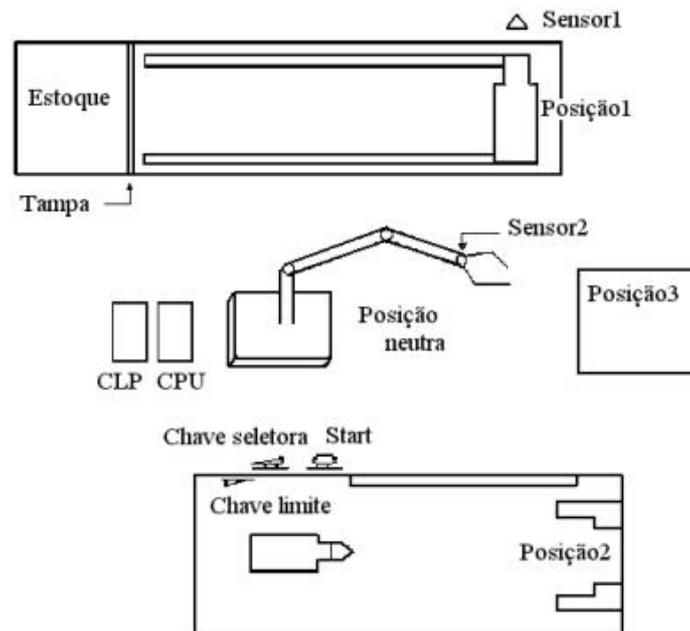


Figura 1: Desenho esquemático da automação da usinagem

4. PROJETO E CONSTRUÇÃO DO PROTÓTIPO

De acordo com a necessidade de testes e estudos, foi elaborado o projeto de um protótipo do robô com três graus de liberdade. Os motores de passo utilizados neste protótipo foram obtidos em sucatas de impressoras e tem pequeno torque. A tabela 1 mostra a relação entre a carga máxima e o tempo em que cada bobina fica energizada do motor de passo que apresenta as seguintes características:

Voltagem = 5 V;

Corrente = 1 ampère / fase;

Resolução = 7,5 graus / passo;

Número de bobinas do motor = 4 bobinas;

Diâmetro do eixo do motor = 10,7 mm;

Para o cálculo da carga máxima que o robô pode suportar, foi feito um teste com os motores que fazem parte do protótipo. O teste foi feito da seguinte maneira: primeiro foi verificado o tempos em que as bobinas permanecem energizadas, dentro do qual o motor responde para funcionar normalmente. Para isto foi desenvolvido um programa de controle na

linguagem Pascal. Em seguida foi fixado ao eixo do motor um suporte onde eram colocados pesos padrões. De acordo com o tempo analisado, foram sendo colocados pesos até que o motor não conseguisse mais girar, ou seja, a carga máxima para o intervalo de tempo analisado, foi determinada como sendo a carga anterior àquela em que o motor não suporta carregar. Nessa análise notou-se que a medida que aumenta-se o tempo em que cada uma das bobinas permanecem energizadas, a carga que o motor suporta também aumenta. Esta primeira análise foi feita levando-se em consideração que apenas uma bobina por vez é acionada em cada energização. Numa segunda análise este teste foi repetido, só que com duas bobinas acionadas simultaneamente em cada energização. Notou-se que o torque do motor aumentou consideravelmente nesta situação. Verificou-se também que a carga máxima com o motor travado foi de 1833,6g. Na tabela são apresentadas as relações entre a carga máxima e o intervalo de tempo entre os pulsos.

Tabela 1: Relação entre carga máxima e o tempo em que as bobinas permanecem energizadas

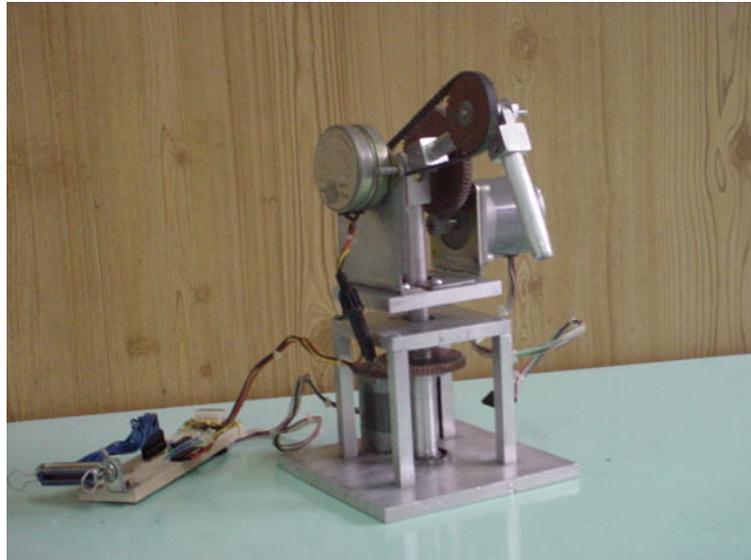
T(ms)	Carga máxima (g) com uma bobina energizada por vez	Carga máxima (g) com duas bobinas energizadas por vez
5	105,5	184,3
7	203,5	
10	313,5	
15	459,5	619,1
22	481,5	
33	492,5	
49	501,5	
73	518,5	738,0
108	457,5	
160	502,5	
237	477,5	738,1

Devido ao fato destes motores de passo terem baixo torque, a estrutura do robô tem de ser leve e com articulações onde não há perdas consideráveis por atrito. De acordo com estas exigências, foi utilizado como matéria prima, em sua maioria, o alumínio, por ser leve e de fácil usinabilidade. O eixo principal do robô, que suporta a maior carga, foi apoiado em rolamentos para diminuir o atrito nos apoios. Para as articulações onde os esforços são menores foram utilizadas buchas de nylon. A figura2 apresenta fotos do protótipo em duas posições.

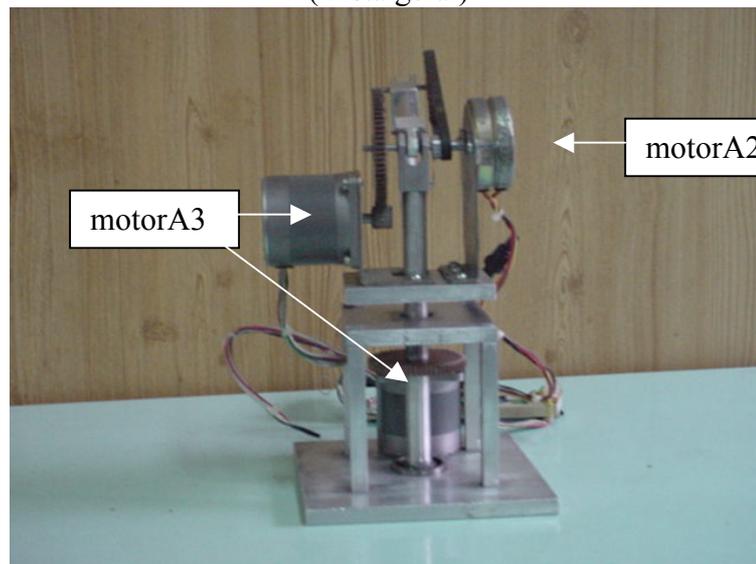
Os componentes do robô foram usinados a partir de uma chapa de 154mm de largura e 9.5mm de espessura; de uma barra quadrada de uma polegada; de um tubo de meia polegada; e de um eixo de 16mm de diâmetro, todos de alumínio, de onde foram fabricados eixos, placas e articulações (junta e garfo). Eixos e pinos foram feitos a partir de outros eixos retirados de impressora. As engrenagens e polias dentadas foram construídas em nylon, celeron ou alumínio. Um dos suportes dos motores foi feito em uma chapa de latão e o outro foi retirado de uma impressora, tendo sofrido pequenas alterações.

O robô é todo apoiado em uma placa maior, a qual foi furada no centro e posteriormente rebaixada a extremidade do furo no torno, para que possa desta forma, acomodar um rolamento (35x15x11mm). Para fixação das hastes foram feitos quatro furos escareados. Na placa maior foram fixadas quatro hastes que foram aplainadas, limadas, furadas e roscadas nas extremidades. Estas hastes suportam uma placa menor, a qual passou pelos mesmos processos de usinagem da placa maior. No centro da placa menor também foi fixado um rolamento. O eixo principal é apoiado pelos rolamentos das placas, nele foi fixada uma engrenagem de celeron usinada na fresadora universal. Uma base para os motores também foi fixada neste

eixo. Esta base foi aplainada, furada no centro e nos quatro cantos. Nela foi fresado um rasgo, o qual suporta um pino de 5mm que passa dentro do eixo principal fazendo a fixação dele com a placa que suporta os motores. Nesta terceira placa são fixados os suportes para os outros dois motores. Um desses motores tem uma polia dentada (menor) presa no seu eixo e o outro motor tem uma engrenagem de alumínio presa em seu eixo.



(Vista geral)



(Vista frontal)

Figura 2: Fotos do protótipo.

Na extremidade do eixo principal tem uma articulação, formada por um garfo e uma junta, os quais foram inicialmente serrados, aplainados, limados e furados. No garfo ainda foram colocadas duas pequenas buchas de nylon, que foram torneadas. Esta articulação é unida por um eixo (retirado de sucata de impressora), no qual ainda se tem uma engrenagem que trabalha com a engrenagem do motor A3. O garfo é preso ao eixo principal, enquanto que na junta é fixado um tubo vazado, que foi serrado, limado e furado. Na outra extremidade deste tubo é fixado outro garfo idêntico ao primeiro. Uma junta é ligada ao garfo por um eixo, no qual ainda é fixada uma polia dentada que é movida por uma correia, ligada a polia dentada do motor A2. Nesta segunda junta é preso um outro tubo, semelhante ao primeiro.

5. CONTROLE DOS MOVIMENTOS

Como já foi dito anteriormente, o protótipo que foi desenvolvido neste trabalho é movido por motores de passo. Estes motores podem ser controlados por CLP (necessitando da utilização de mais entradas e saídas do equipamento), porém, o uso do computador para controlá-lo é mais simples e mais viável financeiramente e com recursos de programação mais flexíveis (Petruzela, F.D., 1996).

O programa para controle dos movimentos pode ser feito em diversas linguagens, no entanto o programa para testes com um motor de passo foi feito em Pascal. O programa completo, para o controle simultâneo de três motores, está sendo desenvolvido em C++ (Jamsa, K, 1999). O sistema de controle do robô industrial utiliza: um computador, que contém o programa de controle (em C++ ou Pascal) e a porta de comunicação paralela; placas de controle dos motores de passo; e fonte de energia.

Quando estes motores são aplicados em uma estrutura, como um braço mecânico, suas articulações podem ser movidas através do programa, porém, o movimento executado pelas articulações pode não ser exatamente o que é desejado no programa, devido a diversos fatores, tais como resolução do motor, perdas por atrito ou defasagem do tempo de energização.

O protótipo é constituído por três motores de passo (três graus de liberdade). O circuito elétrico que controla os três motores é dividido em três partes, uma para cada motor, os quais tem diferentes tensões de alimentação e valores de corrente elétrica, fazendo com que cada parte do circuito tenha pequenas alterações nos seus componentes, porém, a lógica das três partes é a mesma. A lógica de controle de um motor aplica-se para os demais controles.

Cada parte do circuito que controla um motor tem entrada para dois fios (fio A e fio B), onde cada um é ligado a um pino da porta paralela do computador. Esta placa trabalha com código binário.

A lógica do circuito funciona de modo que quando a entrada *A* está desenergizada a leitura da entrada *B* é ignorada. E quando a entrada *A* está energizada, é realizada a leitura da entrada *B*. Esta leitura da entrada *B* indica o sentido de rotação do eixo do motor (bit0 em um sentido, bit1 no sentido oposto), quando a entrada *A* é energizada quer dizer que o motor deve ser girado, e a placa é quem controla a distribuição de pulsos entre as bobinas.

A velocidade de rotação do motor é controlada pelo programa do computador, através do intervalo entre pulsos de energia. A velocidade de rotação do eixo do motor é inversamente proporcional ao intervalo entre pulsos. Através de testes experimentais verificou-se que o torque dos motores é inversamente proporcional a velocidade, porém, existem limites. Se o intervalo entre pulsos for maior que um certo valor, este tempo não causará diferença no torque, pois, funcionará como se o motor estivesse parado. Para intervalos menores que certos valores, o movimento se tornará intermitente, pois o motor não acompanhará a distribuição de pulsos.

Para que o braço seja movimentado a uma velocidade considerável sem que ocorram oscilações no movimento (principalmente na partida e na parada), é necessário que o programa controle uma aceleração e uma desaceleração do movimento. Este controle é feito através da variação do intervalo entre pulsos.

6. PROJETO DE UMA GARRA MECÂNICA

Em paralelo com o projeto do protótipo, desenvolveu-se também um estudo sobre garras mecânicas. Neste estudo foram analisadas variáveis como: número de dedos da garra, tipo de empunhadura, tipo de acionamento, tipo de mecanismo (dispositivo cinemático).

Para operações em que a garra tem como meta pegar objetos que apresentam uma certa simetria, observa-se que a existência de apenas dois dedos é satisfatória para execução da operação de pega. Geralmente em garras que apresentam dois dedos a movimentação destes

ocorre de forma simultânea, mas pode haver também a movimentação de apenas um dos dedos, com o outro permanecendo fixo. Esta opção tira um pouco da versatilidade da garra, mas tem como vantagem a diminuição do peso devido a existência de menos dispositivos mecânicos para a movimentação da garra (engrenagens, parafusos sem-fim, cremalheiras, hastes, cabos de aço) e menos motores. Utilizando o tipo de garra descrito acima, desenvolvemos um projeto de uma garra experimental de dois dedos, em que um deles apresenta-se fixo (dedo esquerdo), e no dedo direito é fixada uma cremalheira que realiza um movimento de abrir e fechar. Esta cremalheira é movida por um pinhão que se encontra num eixo, onde na outra extremidade existe uma polia que transmite o movimento do motor, que se encontra no cotovelo do braço mecânico, através de correia para o eixo.

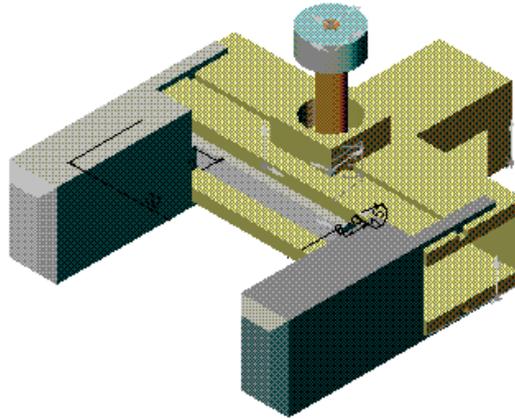


Figura 3: Garra de dois dedos

Para a operação de pega de peças de formatos variados (cônicos, cilindros de diâmetros diferentes), além das variáveis analisadas no projeto levamos em consideração a precisão de posicionamento e a independência entre os dedos. A precisão de posicionamento é necessária para que a colocação da peça na placa do torno CNC ocorra de forma que ela fique centrada, e na posição correta a ser fixada, evitando-se colisões. A interdependência entre os dedos da garra se faz necessária, pois para uma peça de diferentes diâmetros ou de forma cônica, os dedos se adaptam de acordo com o formato da peça.

Com relação ao número de dedos necessários para operação de pega, o uso de três dedos é satisfatório, pois usando um deles como polegar (base), este seria o dedo que passaria pela linha de ação do centro de massa da peça e os outros dois dedos, lado-a-lado equilibrariam a peça impedindo sua rotação em torno do polegar. Estes dedos apresentam duas falanges cada um. Esta subdivisão é necessária para uma melhor pega e uma melhor adequação da garra com o formato da peça.

O tipo de empunhadura é a maneira de segurar a peça, que são duas: por constrição física, onde os dedos envolvem a peça até um certo ponto, impedindo o movimento desta (essa maneira de segurar é conseguida geralmente projetando as superfícies de contato dos dedos numa forma aproximada da geometria da peça); e por atrito, os dedos devem aplicar uma força que é suficiente para reter a peça por atrito contra a gravidade, aceleração e qualquer outra forma de ação de movimento enquanto a peça estiver sendo segura. Entre os dois tipos de empunhadura, notamos que por constrição física seria uma boa opção de empunhadura, utilizando também pequenas almofadas ou material elástico, tipo borracha, nas pontas dos dedos para estes melhor se adequarem a forma da peça.

Para as garras mecânicas temos também três tipos de acionamento (ou fonte de energia para o movimento) analisados: pneumático, hidráulico e elétrico. O acionamento pneumático apresenta uma desvantagem que é a pouca precisão que disponibiliza ao sistema devido a compressibilidade do ar. Já o mesmo não ocorre com o acionamento hidráulico, pois temos o óleo, que é muito menos compressível que o ar. Mas temos o problema de vazamento que não é bom para o sistema, causando assim problemas de sujeira e problemas também de precisão

do sistema. Então optou-se pelo acionamento elétrico devido a melhor precisão e menor custo dos dispositivos necessários.

Em termos do tipo de mecanismo esta variável analisada apresenta as seguintes opções: sistema de braços articulados (hastes), atuação por engrenagem e cremalheira, atuação por cames, atuação por parafuso, atuação por correia e polia, além da atuação por cabos de aço (tendões). De acordo com estudos feitos selecionou-se o sistema de braços articulados, pois este, assegura um movimento das duas falanges que os dedos apresentam, de forma simultânea.

Com relação à precisão necessária para colocação da peça na placa do torno, a garra também terá que contar com motores que farão uma rotação em relação ao sentido longitudinal do braço, e outra no sentido transversal, como num pulso humano. Além da precisão, estes movimentos são necessários para que se a peça precisar ser virada, para continuação da operação de usinagem, esta etapa possa ser executada.

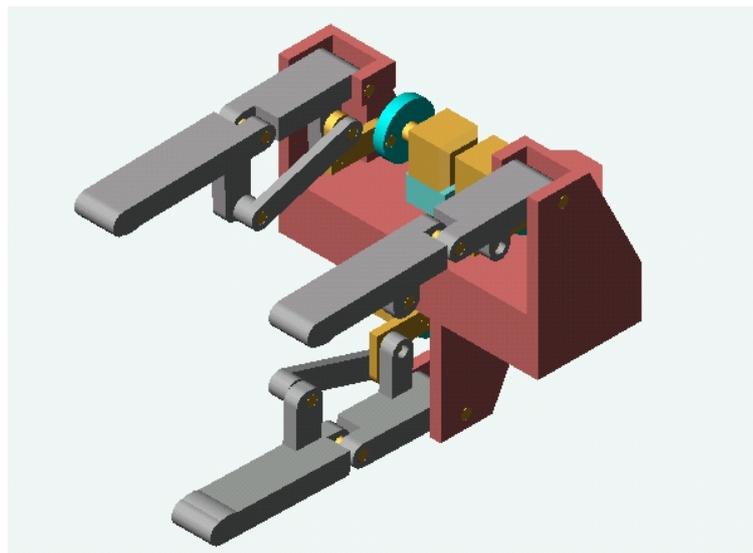


Figura 4: Garra de três dedos

7. CONCLUSÃO

De acordo com análise feita na região sobre o nível, de automação das indústrias, integrado de robôs e máquinas de usinagens, a implantação deste trabalho que foi desenvolvido pode proporcionar um aumento da automação, aumentando assim a produção. A partir deste trabalho é possível desenvolver outras células de manufatura (aplicação do robô em outras máquinas).

Neste trabalho foi projetado e construído um protótipo de um manipulador mecânico para alimentar máquinas CNC. Este protótipo está sendo utilizado para testes de precisão dos movimentos. Os testes preliminares mostram que alguns ajustes devem ser realizados, tanto no mecanismo quanto no programa de controle, para um melhor desempenho do posicionamento e da velocidade dos movimentos desenvolvidos.

8. AGRADECIMENTOS

Ao SENAI, à FACEPE e ao CNPq, pelas bolsas de estudo.

9. REFERÊNCIAS

Cunha, L.S., “Manual prático do mecânico”, ed. Hemus.

- DasGupta, A., Hatwal, H., 1998, "Dynamics and Nonlinear Coordination Control of Multifingered Mechanical Hands", *Journal of Dynamics System, Measurement, and Control*, ASME, Vol. 120, pp 275-281.
- Groover, M.P., Weiss, M., Nagel, R.N., Odrey, N.G., 1988, "Robótica Tecnologia e Programação", McGraw-Hill, São Paulo.
- Jamsa, K., 1999, "Aprendendo C++". MAKRON books
- Jeon, J.W., Ha, Y.Y., 2000, "A generalized Approach for the Acceleration and Deceleration of Industrial Robots and CNC Machine Tools", *IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS*, Vol. 47, Iss 1, pp 133-139.
- Kief, H.B., Waters, T.F., 1992, "Computer Numerical Control" pp 337-338, Glencoe Macmillan, McGraw Hill, New York.
- Matsuoka, Y., 1997, "The Mechanisms in a Humanoid Robot Hand", *KLUWER ACADEMIC PUBLISHERS, Autonomous Robot 4*, pp 199-209.
- Michelman, P., 1998, "Precision Object Manipulation with a Multifingered Robot Hand", *IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS*, Vol. 14, pp 105-113.
- Park, Y.C., Starr, G.P., 1992, "Grasp Synthesis of Polygonal Objects Using a Three-Fingered Robot Hand", *The International Journal of Robotics Research*, Vol. 11, n° 3, pp 163-184.
- Petruzela, F.D., 1996, "Programmable Logic Controllers", Ed.McGraw-Hill, Columbus, cap.6.
- "Manual do programa Solidworks", 2000, Getting start, 272 pp, Massachusetts
- Sun, S. , 2002, "Assessing computer numerical control machines using data envelopment analysis", *INTERNATIONAL JOURNAL OF PRODUCTION RESEARCH*, Vol 40, Iss 9, pp 2011-2039, 4 OXON, ENGLAND.
- Townsend, W.T., "MCB-Industrial Robot Feature Article, The BarrettHand grasper-programmably flexible part handling and assembly", *INDUSTRIAL ROBOT: AN INTERNATIONAL JOURNAL*, Vol. 27, n° 3 , pp 181-188.
- <http://www.irobotics.com>

C898s**Costa, Erwin Rommel Ferreira**

Simulação de movimento e planejamento de trajetória para robôs manipuladores / Erwin Rommel Ferreira Costa . – Recife : O Autor , 2003.

vii, 66 folhas : il. ; fig., tab. e gráficos.

Dissertação (mestrado) – Universidade Federal de Pernambuco . CTG. Engenharia Mecânica, 2003.

Inclui bibliografia e anexos.

1. Robótica . 2. Robôs manipuladores. 3. Simulação computacional - trajetória. I. Título.

621

CDD (21.ed.)

UFPE
BCTG/2004-25