CAPÍTULO 6

PROGRAMAÇÃO E SIMULAÇÃO DE ROBÔS

Renato Ventura Bayan Henriques

6.1- INTRODUÇÃO

Com o crescente desenvolvimento dos processos de manufatura surgiu a necessidade de linhas de produção que permitissem uma maior flexibilidade na quantidade e na variedade de produtos a serem produzidos. A indústria com a globalização dos processos, sentiu a necessidade de romper com os paradigmas, como por exemplo que toda a empresa deveria obter uma alta taxa de produtividade com uma pequena variedade de produtos, característica da visão Taylorista, e adequar-se as necessidades do mercado mais especificamente do cliente.

Neste processo de mudança surgiu o conceito de *automação flexível* onde basicamente otimizamos o layout existente no chão de fábrica para torná-lo o mais flexível frente a variação de produtos a que a linha de produção esta sujeita. Em outras palavras com um layout mais racional pode-se obter uma alta taxa de produtividade com uma ala variedade de produtos.

Desde o início da automatização dos processos[1] ficou clara a necessidade de uma sistemática que possibilitasse a otimização das linhas de produção. Ao analisarmos a estrutura de uma linha de produção verificamos que algumas tarefas são repetitivas e que perdemos um tempo considerável para programá-las (reprogramá-las), a busca da automatização (ou semi-automatização) de determinadas tarefas passa pela definição de como replicar uma determinada tarefa em outra máquina. Neste livro estamos tratando da robótica e da programação dos robôs de uma célula robotizada, a questão que surge aqui é: Como replicar as tarefas executadas num robô num parque industrial com uma centena deles?

A resposta para esta pergunta nos leva a Sir. Charles Devol[2] que desenvolveu uma forma de registrar uma seqüência de movimentos, iniciando assim a primeira geração de robôs. A partir de Devol começou-se a ter a consciência do conceito de *reprogramabilidade*, que é atualmente a base do desenvolvimento dos robôs comerciais existentes no mercado .

Um dos pontos importantes quando se deseja programar uma determinada tarefa no robô é o fato de ser ou não possível executar uma seqüência específica de comandos. Programar um robô significa descrever os procedimentos a serem tomados pelo manipulador sob a ação do controlador. Devido ao grande número de controladores de robôs existentes e a crescente evolução dos sistemas de programação de alto nível orientados ao usuário faz-se necessário uma interface entre eles. Conforme a norma ISO TR 10562 (*Manipulating industrial robots - Intermediate Code for Robots (ICR)*) um código intermediário deve ser usado como uma interface entre programas de robô orientados ao usuário e controladores de robôs industriais.

Inicialmente passaremos a uma descrição do problema básico a ser enfrentado quando desejamos efetuar a programação de um robô. Em um segundo momento descreveremos as

[1]Protótipo de comando numérico desenvolvido no Instituto de Tecnologia de Massachusets - MIT
[2]Charles Devol, desenvolveu dispositivo controlador para registrar sinais eletricos magneticamente

características básicas de uma linguagem de programação e daremos um breve histórico da sua evolução. Neste capítulo tentar-se-á situar o leitor no contexto de manufatura automatizada, preparando-o para utilizar as principais ferramentas de programação e simulação existentes no mercado.

4.1 - O PROBLEMA BÁSICO DA PROGRAMAÇÃO

Para um dado problema de programação existirão diversas formas de implementarmos um algoritmo para solucioná-lo, destas escolheremos uma em função do índice de desempenho adotado de forma a minimizá-lo ou que seja a mais próxima do ideal. Eis ai a motivação precípoa da programação e consequentemente para a programação de robôs.

Da busca constante da otimização das soluções é que surgiram ao longo dos anos ferramentas computacionais mais avançadas, que propiciam ao projetista uma maior flexibilidade no que tange a tarefa a ser executada.

Inicialmente as linguagens existentes eram baseadas em comandos intuitivos do tipo move to, open tool, etc., que descreviam literalmente o tipo de ação a ser executada. Concomitantemente com estes desenvolvimentos os computadores, unidade básica no processamento destas informações, experimentaram uma constante evolução em termos de quantidade de informação, velocidade de processamento e disponibilidade de periféricos.

A conjunção de fatores tais como: evolução tecnológica dos microcomputadores, desenvolvimento de linguagens e ferramentas de análise e programação é que nos possibilitou atingirmos o estágio atual da automatização robotizada.

Nosso problema no entanto, se reduz a apresentar de forma clara e objetiva o estado atual da programação de robôs propiciando ao leitor uma visão dos procedimentos adotados ao resolver um problema de programação.

4.2 - MÉTODOS DE PROGRAMAÇÃO

A pesquisa na indústria durante os últimos vinte anos tem sido direcionada na criação de técnicas de automação que aplicadas em lotes de produção de pequeno e médio porte produzam resultados adequados do ponto de vista custo-benefício. Isto culminou com o desenvolvimento das máquinas CNC, Sistemas Flexíveis de Manufatura (FMS), Robôs Móveis (AGV's) e braços manipuladores.

O desenvolvimento deste último grupo, tem particular importância, pela complexidade intrínseca envolvida no desenvolvimento da habilidade de emular o comportamento da cadeia de ligamentos do braço manipulador de forma a replicar os movimentos do braço humano.

Os robôs industriais de hoje são mecanismos automatizados projetados para movimentar peças ou ferramentas sobre uma trajetória previamente estabelecida. Como mencionado anteriormente um robô ou um outro do mesmo modelo deverá ser capaz de executar um conjunto de operações ou movimentos diferenciados se a célula de trabalho do mesmo for alterada. O programa de controle do robô deverá ser capaz de adaptar-se as variações de tarefas e ser flexível o suficiente para permitir uma seqüência dinâmica de operações. Pode-se avaliar a flexibilidade de um robô pela extensão do tipo de operações e movimentos que podem ser programados no seu controlador e pela facilidade de entrada ou alteração de um programa.

A programação pode ser feita de duas maneiras: *Programação on-line* e *Programação off-line*. A programação off-line é feita utilizando-se linguagens de programação criadas especialmente para robôs tais como VAL, WAVE, AML, MCL, and SIGLA. Estes programas são

geralmente depurados com uso de simuladores. A *Programação on-line* por outro lado faz uso geralmente de métodos de programação por ensino ou condução.

Os métodos atuais de programação de robôs industriais tem provado serem satisfatórios onde a proporção entre o tempo de programação e de produção é pequeno e também quando a complexidade da aplicação não é tão exigida.

A programação "*off-line*" pode ser definida neste contexto como o processo pelo qual a programação dos robôs é desenvolvida, parcial ou completamente, sem a necessidade do uso do robô.

6.3.1 - Programação On-line

Podemos definir programação *on-line* como a utilização de métodos de programação por ensino para aplicar um programa de controle no controlador do robô. O programador conduz o robô através de uma seqüência de posições desejadas(via teach-pendant ou dispositivo mestre-escravo).

Neste processo de "ensino" envolvemos as tarefas de identificação dos pontos, edição e repetição do trajeto ensinado. Programas de edição são utilizados para adicionar informações relevantes ao programa de controle bem como aos equipamentos de produção associados. O uso de programas de edição nos permite um meio de avaliar e corrigir programas de controle existente ou ajustar pontos quando uma tarefa é modificada.

Durante este processo o programador necessita de funcionalidades que podem incluir repetição do programa para frente e para trás, operações passo a passo, seleção de velocidade de execução entre outras, que facilitam a depuração do programa. Necessariamente esta abordagem utiliza o robô para a programação e de alguma forma é dependente do algoritmo de controle usado para movimentação entre os pontos da trajetória.

Três algoritmos básicos de controle são usualmente utilizados: movimento ponto a ponto, continuo e o controle de trajetoria.

Robôs com controle ponto a ponto movem-se de uma posição a outra sem considerar o trajeto intermediário entre os pontos. Geralmente cada eixo move-se até atingir a posição desejada.

Robôs de movimento contínuo movimentam-se através de pontos com pequenos incrementos entre si, armazenados ao longo de uma trajetória previamente percorrida. As posições de cada eixo são gravadas pela unidade de controle a intervalos de tempo constante recolhendo dados dos sensores de posição durante a movimentação do robô.

O movimento controlado de trajetória envolve o controle coordenado de todas as juntas para percorrer uma trajetória desejada ao longo de dois pontos. Neste método os eixos movem-se suavemente e proporcionalmente para gerar o trajeto de controle especificado.

Em programação on-line duas abordagens básicas são tomadas para ensinar o robô uma trajetória desejada: Métodos de programação por aprendizagem e Linguagens textuais.

6.3.1.1 - Programação por aprendizagem

Os métodos de programação por aprendizagem exigem que o programador conduza o manipulador, movendo-o fisicamente de modo a realizar as manobras que ele deve apreender. Esse método é mais utilizado em tarefas que necessitam de uma trajetória contínua, como pintura por pulverização, corte a jato d´agua, aplicação de selante ou solda a arco.

A programação por aprendizagem envolve o uso de um *joystick*, um teclado comum ou um teclado portátil chamado de *teach box* para guiar o robô ao longo de uma trajetória planejada. Se o programa (ou *teach box)* especificar uma trajetória, continua ou ponto a ponto usando as coordenadas de mundo estes sinais são enviados ao controlador que os transforma em coordenadas de junta e produz os movimentos desejados.

6.3.1.1.1 - Programação por condução

Aprendizado por condução também é utilizado no modo ponto a ponto, onde o robô grava apenas certos pontos da trajetória deslocando-se posteriormente através delas em linha reta(ou por trajetos circulares, se assim for solicitado).

Na programação por condução , o operador move fisicamente o efetuador final pelo trajeto desejado. Para trajetórias contínuas, os sensores do braço enviam continuamente informação sobre a posição de cada junta para o controlador do robô à medida que o braço vai se movimentando. Para trajetórias ponto a ponto, a informação de posição da junta é enviada apenas nos pontos da trajetória onde o operador especificamente a posiciona. Em qualquer caso, uma vez gravados na memória do computador os pontos podem ser chamados a qualquer tempo para reprodução .

Ambos os métodos de programação descritos anteriormente envolve a tarefa de integração entre três fatores básicos:

- As coordenadas dos pontos devem ser identificadas e armazenadas na unidade de controle. Os pontos podem ser armazenados como coordenadas individuais das juntas ou pelas coordenadas geométricas da flange do robô.
- As funções a serem executadas nos pontos específicos devem ser identificadas e gravadas. Por exemplo para cada trajetória podemos identificar a velocidade de avanço, fluxo de selante a ser aplicado, etc.
- Os pontos e dados funcionais são organizados em seqüências lógicas. Isto inclui quando uma dada trajetória deve ser estabelecida ou quando várias condições devem ser checadas.

Este três fatores são integrados no processo de aprendizagem e não existem em separado como passos de programação.

6.3.1.1.2 - Programação via teach-pendant

A programação on-line necessariamente faz uso de um teclado portátil chamado teach-pendant, para guiar o robô ao longo de uma trajetória. Isto fica bem claro como tomamos como exemplo (vide figura 6.1). Neste exemplo o robô deverá executar a tarefa de pegar a peça na esteira de entrada colocá-la no centro de usinagem, depois do processamento retirar a peça e colocá-la na esteira de saída. Nesta tarefa faz-se necessária a integração do robô com as esteiras e o operador deverá executar uma seqüência pré-determinada de operações que serão sumarizadas a seguir.

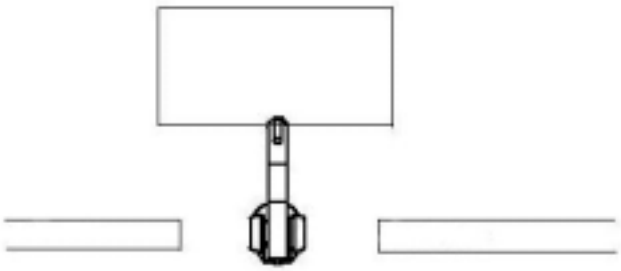


Figura 6.1 *– Célula genérica ser programada*

A seguinte seqüência de pontos deve ser armazenada pelo operador durante a programação on-line do sistema.

1. Mover o braço do robô até que a garra esteja justamente acima da peça na esteira de chegada, então abre-se a garra,
2. Alinha-se o eixo da garra com o da peça a ser transportada,
3. Armazenar o programa apertando a tecla correspondente a *gravar* no teach-pendant,
4. Baixar a garra até que a mesma esteja centrada com o objeto a ser pinçado. Armazenar este ponto,
5. Fechar a garra para que a peça possa ser levantada. Armazenar este ponto.
6. Levantar o braço de forma a liberar o espaço de trabalho e alinhá-lo no nível do centro de usinagem. Armazenar este ponto.
7. Aproximar a garra do centro de usinagem e posicioná-la alinhada com o dispositivo de fixação do centro de usinagem. Armazenar este ponto.
8. Abrir a garra e liberar a peça.
9. Elevar o braço até liberar o espaço de trabalho de forma a retirar o braço do robô do centro de usinagem.
10. Retrair o braço do robô até uma posição intermediária. Armazenar este ponto.
11. Aguarde o sinal do centro de usinagem e retire a peça.
12. Rotacione o braço até a esteira de saída e posicione sobre a superfície da esteira. Armazene este ponto.
13. Abaixe o braço até a superfície da esteira. Armazene este ponto.
14. Abra a garra e libere a peça. Armazene este ponto.
15. Levante o braço ate liberar o espaço de trabalho do robô. Armazene este ponto.
16. Volte para a posição de equilíbrio e reinicie o ciclo.

Os passos listados anteriormente podem variar de acordo com o layout da célula e servem para prover ao controlador as coordenadas dos pontos utilizados no programa. Cabe ao operador entrar via teach-pendant com os parâmetros necessários para a correta sinalização entre o robô e os periféricos (neste caso esteiras e centro de usinagem).

6.3.1.1.2.1 - O controlador do robô e o teach-pendant

O teclado e o display da unidade de controle são freqüentemente usados em conjunto com um teach-pendant padrão. Em geral existem três tipos:

- Teach-pendant genérico (veja figura 6.2)
- Teach-pendant com botões pictográficos (veja figura 6.3)
- Teach-pendant com display (veja figura 6.4)

Figura 6.2 *– Teach-pendant genérico*

Figura 6.3 – *teach-pendant pictográfico*

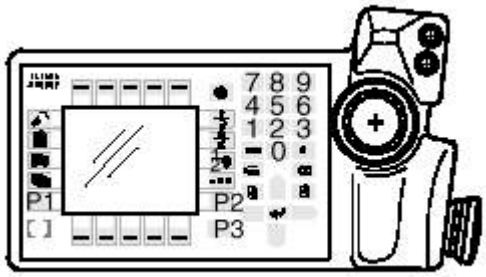


Figura 6.4 *– Teach-pendant com display*

6.3.1.2 - Programação Textual

A primeira linguagem textual de robôs foi WAVE, desenvolvida em 1973 como uma linguagem experimental de pesquisa no Stanford Artificial Intelligence Laboratory. A pesquisa envolvia um robô interfaceado com um sistema de visão. O desenvolvimento de uma linguagem subseqüente começou em 1974, em Stanford. A linguagem foi chamada AL, e podia ser usada para controlar braços múltiplos em tarefas exigindo coordenação dos braços.

Muitos dos conceitos das linguagens WAVE e AL foram aproveitados no desenvolvimento da primeira linguagem textual de programação de robôs comercialmente disponível, a linguagem VAL(Victor Assembly Language, de Victor Scheinman). A linguagem VAL foi introduzida em 1979 pela Unimation Inc. para seus robôs da série PUMA. Essa linguagem foi melhorada para VALII e lançada em 1984.

O trabalho de desenvolvimento de linguagens de programação de robôs prosseguiu também nos laboratórios T.J. Watson Research Labs, da IBM Corporation tendo começado por volta de 1976. Foram desenvolvidas duas linguagens dirigidas para tarefas de montagem e tarefas correlatas, foram elas a AUTOPASS e AML.

Este tipo de programação é realizada de forma semelhante a programação de computadores. O programador digita o programa usando um terminal de computador na linguagem fornecida pelo fabricante do robô.

6.3.2 - Programação Off-line

Os desenvolvimentos na tecnologia de robôs, tanto em software quanto em hardware estão tornando possível que a programação *off-line* se torne cada vez mais viável. Estes desenvolvimentos incluem o uso de controladores mais sofisticados, o aumento da precisão no posicionamento e a adoção de sensores mais avançados.

A programação *"on-line"* de um robô, de um modo geral, pode consumir muito tempo evoluindo desproporcionalmente com o aumento da complexidade das tarefas; consequentemente quando o robô fica fora da linha de produção o tempo gasto na programação pode prejudicar substancialmente a sua utilidade.

Em muitas aplicações envolvendo processos de produção em massa, tais como soldagem a ponto em linhas de produção automobilísticas, os requisitos temporais de reprogramação devem ser minimizados ao máximo.

Logo podemos concluir que para a aplicação de robôs ser factível, em pequenos e médios lotes de produção, onde os tempos envolvidos podem ser substanciais, a utilização de ***Programação Off-line*** é altamente recomendada.

O incremento na complexidade das aplicações em robótica torna as vantagens associadas a programação off-line mais atrativos, essas vantagens podem ser classificadas como segue:

1. **Redução do tempo ocioso:** O robô pode manter-se na linha de produção enquanto a próxima tarefa estiver sendo programada. Isto acrescenta maior flexibilidade aos robôs.
2. **Ambientes potencialmente perigosos:** Redução do tempo de permanência do operador próximo ao robô ,reduzindo assim o risco de acidentes por comportamento anormal do equipamento.
3. **Sistema Simplificado de Programação:** Pode-se usar a programação off-line para programar uma grande variedade de robôs sem a necessidade de conhecer as peculiaridades de cada controlador. Reduz-se assim o índice de reciclagem dos programadores.
4. **Integração com sistemas CAD/CAM:** Habilita a interface com banco de dados de peças, centralizando a programação de robôs com estes sistemas possibilitando o acesso a outras funcionalidades, como por exemplo planejamento e controle.
5. **Depuração de Programas:** Sistemas de programação off-line com CAD/CAM integrados podem produzir um modelo da planta (robô + célula de trabalho)que podem ser usados para detecção de colisões dentro do espaço de trabalho e se será possível executar determinados movimentos evitando assim danos ao equipamento.

6.3.2.1 - Limitações da Programação Off-line

A programação off-line como já foi dito anteriormente necessita obrigatoriamente da existência de um modelo teórico do robô e do ambiente; o objetivo é usar este modelo para simular o comportamento real do robô. A implementação da programação off-line encontra basicamente três problemas principais:

a) Dificuldade em desenvolver um sistema de programação generalizado que seja independente do robô e de suas aplicações;

b) Para reduzir a incompatibilidade entre robôs e sistemas de programação faz-se necessário a definição de padrões para as interfaces;

c) Programas gerados em off-line devem levar em conta os erros e imprecisões que existem entre o modelo idealizado e o mundo real.

Devido a estas imprecisões entre o modelo teórico idealizado e a variáveis inerentes ao processo no mundo real, seqüências simuladas geralmente não atingem o objetivo de controlar o robô sem erros. Na prática o robô não atinge o local calculado pelo modelo ou a ferramenta não é precisamente localizada como definido no modelo. Estas discrepância podem ser atribuídas aos seguintes fatores:

1. **Robô:**
   - Falta de precisão na tolerância da montagem dos ligamentos provocando o aumento na variação do off-set das juntas. Pequenos erros na estrutura são amplificados e produzem grandes erros de posicionamento no efetuador;
   - Falta de rigidez na estrutura do robô. Pode causar grandes erros quando este está sujeito a condições severas de carregamento;
   - Incompatibilidade entre robôs do mesmo modelo. Devido a diferenças no setup do sistema de controle de cada robô a mesma programação off-line pode apresentar pequenos erros.
2. **Controlador:**
   - Resolução insuficiente do controlador. A resolução específica o menor incremento de movimento atingível pelo controlador.
   - Precisão numérica do controlador. É afetado pelo comprimento da palavra do controlador e a eficiência do algoritmo usado para os propósitos de controle.
3. **Ambiente:**
   - Dificuldade na determinação precisa dos objetos(robôs,máquinas, pecas) com relação ao sistema de coordenadas generalizadas.
   - Efeitos do ambiente tais como temperatura, podem causar efeitos adversos ao desempenho do robô.
4. **Modelo e o Sistema de Programação:**
   - A precisão numérica do processador do computador.
   - A qualidade dos dados do modelo real. Isto determina a precisão final do programa gerado em off-line.

A composição destes erros através de todo o sistema de programação off-line pode levar a discrepâncias de magnitude significante. Para que a programação off-line se torne uma ferramenta prática, esta magnitude deve ser reduzida a níveis onde os ajustes do posicionamento final possam ser executados automaticamente.

Na programação off-line, um programa pode estar contido em um disquete ou ser transmitido eletronicamente (via rede); este pode ser parte de uma biblioteca completa de programas desenvolvidos para aquele modelo de robô, embora este não tenha sido desenvolvido especificamente naquele robô.

6.4 - LINGUAGENS DE PROGRAMAÇÃO DE ROBÔS INDUSTRIAIS

Atualmente, após a padronização das linguages de programação de robôs pela ISO, *"International Organization for Standardization"*, a linguagem de programação para comunicação homem-máquina já é bem conhecida, além disso temos a padronização da interface máquina-máquina.

Uma condição importante para as linguagens de programação da inteface homem-máquina é que estas devem necessariamente ser interativas. Atualmente existem duas correntes de pensamento na estruturação da interface homem-máquina. Uma defende que a linguagem deve ser simples e capaz de ser usada pelos próprios operadores sem um treinamento computacional específico. A outra defende que a linguagem deverá prover requisitos computacionais poderosos e que somente técnicos especialmente treinados devem desenvolver a programação. Um exemplo clássico da primeira filosofia é a linguagem ARLA da ABB, como segundo exemplo podemos citar a linguagem KAREL da FANUC.

Atualmente, existem centenas de linguagens de robôs disponíveis comercialmente. Muitas delas baseadas em linguagens clássicas tais como Pascal, C, Modula-2, BASIC, e Assembler. As linguagens de programacão podem ser classificadas de acordo com o sistema de referência do modelo, o tipo de estrutura de controle utilizada, o tipo de especificação de movimento, a interface com o dispositivos externos e os periféricos a serem utilizados. Segundo GONG (1998) os seguintes tipos de linguagens de programação podem ser classificados em:

- Linguagens de movimento ponto a ponto
- Linguagens de movimentacao básica, linguagem de baixo nivel (Assembly)
- Linguagem de programação não estruturada de alto-nivel
- Linguagem de programação estruturada de alto-nivel
- Linguagens do tipo NC (Numeric Command)

Conforme norma ISO TR 10562 a linguagem ICR é um pseudocódigo de baixo nível que possui os elementos básicos para permitir que qualquer linguagem de alto nível seja para ela traduzida. Para que isso seja possível devem ser desenvolvidos compiladores adequados. Cada tradutor irá fazer uso das capacidades da ICR do modo que lhe for mais conveniente a fim de satisfazer o usuário final. Apenas a título de exemplo são citadas algumas técnicas para implementações de linguagens comuns.

6.4.1 - Linguagens tipo BASIC

Este tipo de linguagem se caracteriza por um algoritmo linear e simples sem compilação em módulos separados, sem abstração de dados ou algoritmos, existem apenas tipos de dados predefinidos e as chamadas às subrotinas não utilizam passagem de argumentos. Este tipo de linguagem geralmente é interpretada, traduzida para ICR e somente depois executada, linha por linha. Como as linhas de código são sintaticamente independentes, este método simples de interpretação não causa problemas.

6.4.2 - Linguagens tipo PASCAL

Este tipo de linguagem é caracterizada pelo encorajamento à programação estruturada. As funções e procedimentos podem ter argumentos e as variáveis podem ser globais ou locais (a

uma função ou procedimento). Recursão é permitida e facilmente implementada e em alguns casos (MODULA 2 e ADA) a programação pode ser modular. Neste último caso um ligador deve providenciar a resolução das referências externas e converter os módulos em um único executável ICR. A linguagem ICR não oferece suporte nativo para ligação dinâmica. Se for necessário utilizar esta técnica o ligador dinâmico deve ser implementado de forma independente.

A passagem de parâmetros deve ser feita por meio da pilha da esquerda para a direita, ou seja, o primeiro elemento a ser empurrado para a pilha será aquele mais à esquerda (na linguagem original) e no topo da pilha estará aquele mais à direita. Deve ser prestada especial atenção á estruturação dos blocos da linguagem uma vez que o espaço de pilha reservado para as variáveis depende disso. Como a ICR não faz distinção entre funções e procedimentos, a pilha deve ser esvaziada ao final das chamadas de procedimentos.

6.4.3 - Linguagens tipo C

Estas linguagens se caracterizam principalmente pela possibilidade de declaração de variáveis na abertura de qualquer bloco (delimitado por { } em C) e pela capacidade de usar matrizes ou ponteiros de forma indistinta. A primeira característica pode ser implementada diretamente com os comandos BLKBEG e DECLVAR mas a segunda apresenta problemas pois ICR prefere que os dados tenham tipos explícitos. É recomendado que sempre que não seja realmente impossível, se faça a atribuição de tipo.

6.4.4 - Linguagens do tipo LISP

Este tipo de linguagem tem como característica básica a habilidade de trabalhar com listas encadeadas e a intercambiabilidade de dados e programas. Este tipo de linguagem requer um acurado gerenciamento de memória. Apesar de ser possível escrever compiladores para estas linguagens recomenda se a consulta a literatura específica.

6.4.5 - Linguagens tipo FORTH

Pelo fato de utilizar basicamente operações baseadas na pilha e pela semelhança estrutural, este tipo de linguagem não apresenta maiores problemas de implementação. A única ressalva fica por conta da implementação de múltiplas pilhas.

6.4.6 - Linguagens orientadas a objeto

Linguagens deste tipo estão fora do escopo deste texto e não tem grande utilidade na área aplicada de robótica. Como a ICR possui suporte a todos os tipos de dados comuns em robótica e suporte às operações a eles associadas pode-se geralmente cair em alguns dos casos acima.

Muitos robôs industriais são amplamente utilizados em processos de manufatura tais como tarefas de montagem, manipulação de materiais, soldagem a arco/ponto, pintura, carga e descarga de centros de usinagem e em algumas aplicações especiais tais como exploração submarina e pesquisa de próteses para deficientes. Estimasse que haja no mundo cerca de 100 fabricantes de robôs. A tabela 6.1 lista alguns destes fabricantes e e suas respectivas linguagens de programação.

TABELA 6.1 – Lista de fabricantes de robôs e suas linguagens de programação

Fabricante	Linguagens de Programação
ABB	ARLA, RAPID
Fanuc	KAREL
Reiss	IRL
Staubli	V+
Adept	V+
Comau	PDL2
Eshed	ACL
IBM	AML/2
Kawasaki	AS
Motoman	Inform 1, Inform 2
Nachi	SLIM
Panasonic	Parl-1, Parl-2
PSI	PSI
Samsung	FARL-II
Seiko	DARL 4
Toyota	TL-1
TQ	TQ

6.5 - SIMULAÇÃO DE ROBÔS

Hoje em dia os robôs tem uma larga aplicação na indústria e na Manufatura. Exemplo disto são os vários programas de desenvolvimento lançados a nível Nacional e Internacional como por exemplo: RECOPE[3],ESPRIT,BRITE e EUREKA, que são grandes programas de pesquisa que dentre suas linhas de pesquisa focam a robótica como um de seus temas.

Os principais esforços no desenvolvimento e nas aplicações de robótica sem dúvida recaem na indústria automobilística. Os benefícios da automação flexível foram incorporados pesadamente nos processos de manufatura automatizada automotiva. As indústrias automotivas tem encorajado a utilização de robôs em outras áreas encorajando seus fornecedores a utilizarem a mesma tecnologia.

Muitos robôs comerciais tem largo uso em tarefas de montagem e manufatura tais como manipulação de material, soldagem a arco/ponto, montagem de partes, pintura, carga e descarga, etc.

As técnicas de CAD encontraram extensivo uso nos projeto de engenharia auxiliando o projetista em processos que envolviam desenvolvimento de diagramas e desenhos de alta

[3]Grupo Automação da Manufatura

complexidade. Entretanto no que diz respeito a processos que envolvem a movimentação de peças há necessidade de um estudo mais aprofundado através de simulação.

A extensão das técnicas de CAD/CAM também é desejável no domínio da robótica. Entretanto a cinemática envolvida nos movimentos do robô é consideravelmente mais complexa do que as relacionadas com os movimentos xyz dos centros de usinagem e da relação entre os movimentos do efetuador e das juntas do robô. Somado a isto faz-se necessário a utilização de um ambiente capaz de simular uma grande variedade de robôs e tipos de configurações disponíveis no mercado.

Apesar destas dificuldades, a simulação veio para prover um ambiente gráfico capaz de gerar uma interação suave com as diversas linguagens de programação de robôs existentes no mercado comercial. Benefícios tais como: detecção de colisões em off-line, avaliar e otimizar seqüências de programas sem a necessidade da presença física do manipulador tem impulsionado o desenvolvimento das pesquisa em simulação de robôs.

Em resumo, programação off-line prove uma interligação essencial para sistemas CAD/CAM. O sucesso desta integração pode ser evidenciado pela ampla difusão de simuladores comerciais e pela diminuição do tempo de implementação de Sistemas Flexíveis de Manufatura na indústria.

6.5.1 - Simuladores Comerciais

O primeiro simulador comercial que possibilitou a simulação de robôs foi o GRASP, desenvolvido pela Nottingham University durante um período de sete anos. Utilizado por diversas universidades e instituições de ensino tinha a inconveniência de não poder modelar estruturas cinemáticas não seriadas.

Robographics foi desenvolvido pela Computervision uma companhia bem conhecida pelos desenvolvimentos em pacotes comerciais de CAD. Utilizado pela Austin-rover e Unimation. A McAuto CAD divisão da McDonell-Douglas comercializa uma série de pacotes para simulação de robôs, usados pela Cincinatti-Millacron. O programa PLACE usado para avaliação do layout da célula robotizada. O BUILD usado para modelagem e estudos dinâmicos em 3D. O COMMAND usado para programação off-line e o ADJUST para a calibração do robô.

A Technomatics comercializa o ROBCAD. Este pacote é utilizado por muitas montadoras na europa (Ford, BMW, Volkswagen e OPEL). O IGRIP da Deneb, grupo Dassault Systems também é largamente utilizado em muitas empresas. Semelhante ao ROBCAD o DENEB é baseado em uma estação gráfica. A Deneb crescentemente foi comprada pelo grupo Dassault e está incluindo no pacote o software CATIA.

O software WORKSPACE da Flow Technologies, é um ambiente desenvolvido para plataforma PC que possui interface gráfica de simulação capaz de gerar um modelo do ambiente propiciando a geração e interpretação de programas em off-line.

Um grande número de outros simuladores estão disponíveis comercialmente mas tem um pequeno impacto no mercado.

6.5.2 - Softwares Educacionais

Para a modelagem, simulação, programação off-line, detecção de colisões e análise do layout as indústrias montadoras de robôs utilizam softwares que operam em estações de trabalho que demandam alto investimento em equipamentos.

No entanto existem algumas opções comerciais [4] com diferentes capacidades e disponíveis para pesquisa.

- Xanimate: Software educacional para simulação gráfica de robôs, desenvolvido pela University Ohio.
- ROBÔ_SIM: Ambiente de simulação para PC desenvolvido para MATLAB. O pacote consiste de rotinas capazes de desempenhar cálculos específicos (funções para cinemática direta e inversa, dinâmica do manipulador, planejamento de trajetória, controle e simulação).
- Simderela: Software de simulação para ambiente UNIX.
- RobLib : Software de simulação para windows desenvolvido na Faculdade de Engenharia da Universidade do Porto. Capacidade de modelagem de robôs de dois graus de liberdade.

6.6 - EXEMPLOS

Nesta seção será apresentado um exemplo de geração de código para um robô industrial sobre uma aplicação de soldagem (vide figura 6.5), a partir deste serão gerados os códigos apresentados a seguir.

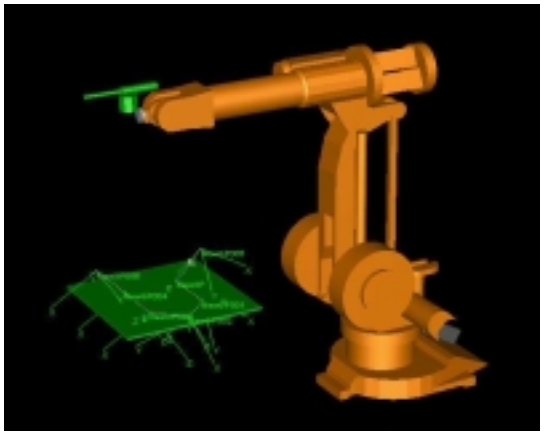


Figura 6.5 – Exemplo didático.

Na seção de apêndices no CD que acompanha o livro serão encontrados exemplos com casos reais de geração de programas em off-line em algumas empresas brasileiras.

[4] Nenhum dos softwares citados tem capacidade para programação off-line

LINGUAGEM RAPID

```
%%%
  VERSION:1
  LANGUAGE:ENGLISH
%%%
MODULE LIVRO_RAPID
VAR robtarget TP1:=[[1144.53,-
0.00,1287.05],[0.70701,0.00000,0.70719,0.00000],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
VAR robtarget WELD_BAS0GP2:=[[686.71,-100.00,630.08],[0.24991,0.06698,0.93303,-0.25000],[0,-
1,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
VAR robtarget WELD_POLGP1:=[[786.71,-100.00,530.08],[0.24993,0.06698,0.93303,-0.24999],[0,-
1,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
VAR robtarget TP2:=[[786.71,-100.00,530.08],[0.24994,0.06698,0.93302,-0.24999],[0,-
1,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
VAR robtarget WELD_POLGP2:=[[786.71,100.00,530.08],[0.24995,0.06698,0.93302,-0.24999],[0,-
1,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
VAR robtarget WELD_POLGP4:=[[986.71,100.00,530.08],[0.24995,-0.93302,0.06698,0.25000],[-
1,0,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
VAR robtarget WELD_POLGP3:=[[886.71,200.00,530.08],[0.00000,0.70707,0.61239,-0.35356],[-
1,0,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
VAR robtarget WELD_POLGP5:=[[986.71,-100.00,530.08],[0.24998,-0.93301,0.06698,0.24999],[-
1,0,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
VAR robtarget TP3:=[[986.71,-100.00,530.08],[0.24999,-0.93301,0.06698,0.24999],[-
1,0,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
VAR robtarget WELD_BAS0GP3:=[[1086.71,-100.00,630.08],[0.25000,-0.93301,0.06698,0.24999],[-
1,0,1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
!! LANGUAGE RAPID
!! MEMORY 32768
!! TEACHPOINTFILE LIVRO_RAPID.PR#
!! ROBOT IRB1400
  PERS tooldata TOCHA:=[TRUE,[[-92.05,-
0.00,189.53],[0.99999,0.00000,0.00000,0.00000]],[0,[0,0,0],[1,0,0,0],0,0,0]];
  PROC main()
    MoveJ [[1144.53,-0.00,1287.05],[0.70710,0.00000,0.70710,0.00000],[-1,0,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]],\V:=1000,fine\Z:=100,TOCHA;
    MoveJ WELD_BAS0GP2,vmax\V:=220,fine\Z:=0,TOCHA;
    MoveL WELD_POLGP1,vmax\V:=220,fine\Z:=0,TOCHA;
    !! ARCWELDON 100,10
    MoveL TP2,vmax\V:=220,fine\Z:=0,TOCHA;
    MoveL WELD_POLGP2,vmax\V:=220,fine\Z:=200,TOCHA;
    MoveC WELD_POLGP3,WELD_POLGP4,vmax\V:=220,fine\Z:=200,TOCHA;
    MoveL WELD_POLGP5,vmax\V:=220,fine\Z:=0,TOCHA;
    !! ARCWELDOFF
    MoveL TP3,vmax\V:=220,fine\Z:=0,TOCHA;
    MoveL WELD_BAS0GP3,vmax\V:=220,fine\Z:=0,TOCHA;
    MoveJ [[1144.53,-0.00,1287.05],[0.70710,0.00000,0.70710,0.00000],[-1,0,-
1,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]],\V:=1000,fine\
Z:=100,TOCHA;
  ENDPROC
ENDMODULE
```

LINGUAGEM ARLA

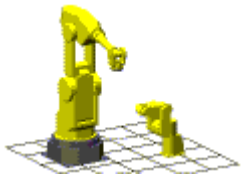
livro_ARLA.irb

```
UNIT=METRIC
INCLUDE LIVRO_ARLA.CRD
PROGRAM 1
COMMENT ! LANGUAGE ARLA
COMMENT ! MEMORY 8192
COMMENT ! ROBOT IRB1400
  V=100.0 MAX=1500.0
  POS V=100.0% FINE X=1144.53 Y=-0.34 Z=1287.05 ->
  Q1= 0.70710 Q2= 0.00000 Q3= 0.70710 Q4= 0.00000
  ROBOT COORD
  POS V=220.0% FINE WELD_BAS0GP2
  RECT COORD
  POS V=220.0% FINE WELD_POLGP1
  ROBOT COORD
  COMMENT ! ARCWELDON 100,10
  RECT COORD
    POS V=220.0% FINE TP1
  POS V=220.0% PATH WELD_POLGP2
  ROBOT COORD
  POS V=100% CIRCLE WELD_POLGP3
  POS V=220.0% PATH WELD_POLGP4
  RECT COORD
  POS V=220.0% FINE WELD_POLGP5
  ROBOT COORD
  RECT COORD
    POS V=220.0% FINE TP2
  COMMENT ! ARCWELDOFF
  POS V=220.0% FINE WELD_BAS0GP3
  ROBOT COORD
  POS V=100.0% FINE X=1144.53 Y=-0.34 Z=1287.05 ->
  Q1= 0.70710 Q2= 0.00000 Q3= 0.70710 Q4= 0.00000
STOP
```

livro_ARLA.cdr

```
WELD_BAS0GP2  686.71 -100.00 630.08 Q   0.24991  0.06698  0.93303 -0.25000 30
WELD_POLGP1  786.71 -100.00 530.08 Q   0.24993  0.06698  0.93303 -0.24999 29
TP1  786.71 -100.00 530.08 Q   0.24994  0.06698  0.93302 -0.24999 29
WELD_POLGP2  786.71 100.00 530.08 Q   0.24995  0.06698  0.93302 -0.24999 29
WELD_POLGP4  986.71 100.00 530.08 Q   0.24995 -0.93302  0.06698  0.25000 2
WELD_POLGP3  886.71 200.00 530.08 Q   0.00000  0.70707  0.61239 -0.35356 2
WELD_POLGP5  986.71 -100.00 530.08 Q   0.24998 -0.93301  0.06698  0.24999 1
TP2  986.71 -100.00 530.08 Q   0.24999 -0.93301  0.06698  0.24999 1
WELD_BAS0GP3  1086.71 -100.00 630.08 Q   0.25000 -0.93301  0.06698  0.24999 1
```

FANUC



LINGUAGEM KAREL

```
PROGRAM LIVRO_KAREL
-- ! LANGUAGE KAREL 2
-- ! MEMORY 8192
-- ! ROBOT IRB1400
-- TEACHPOINT DECLARATIONS
VAR
  WELD_BAS0GP3: POSITION
  WELD_POLGP5: POSITION
  WELD_POLGP4: POSITION
  WELD_POLGP3: POSITION
  WELD_POLGP2: POSITION
  WELD_POLGP1: POSITION
  WELD_BAS0GP2: POSITION
  TP1 : POSITION

BEGIN
  $UTOOL=POS(154.8749,-0,67.6648,90,0,90,'')
  $USEMAXACCEL=TRUE
  %INCLUDE LIVRO#
  WITH $MOTYPE=JOINT
    MOVE TO $HOME:$UTOOL
  WITH $MOTYPE=Joint, $TERMTYPE=FINE, $SPEED=100
    MOVE TO WELD_BAS0GP2
  WITH $MOTYPE=Joint, $TERMTYPE=FINE, $SPEED=100
    MOVE TO WELD_POLGP1
--   ! ARCWELDON 100.0,30
  WITH $MOTYPE=Linear, $TERMTYPE=FINE, $SPEED=100
    MOVE TO WELD_POLGP2
  WITH $MOTYPE=Linear, $TERMTYPE=FINE, $SPEED=100
    MOVE TO WELD_POLGP3
  WITH $MOTYPE=Linear, $TERMTYPE=FINE, $SPEED=100
    MOVE TO WELD_POLGP4
  WITH $MOTYPE=Linear, $TERMTYPE=FINE, $SPEED=100
    MOVE TO WELD_POLGP5
--   ! ARCWELDOFF
  WITH $MOTYPE=Linear, $TERMTYPE=FINE, $SPEED=100
    MOVE TO WELD_BAS0GP3
  WITH $MOTYPE=JOINT
    MOVE TO $HOME:$UTOOL
END LIVRO_KAREL
```

6.7 - REFERÊNCIAS

CRAIG, J. J., *Introduction to Robotics, Mechanics and Control*, Addison-Wesley Publishing Company, 2 ed., Massachusetts, 1986.

GONG, W., *Automatic Robot Path Generation for Manufacturing on Sculptured Surfaces*, Master Thesis, University of Windsor, 1998.

GROOVER, P. M., *Automation, Production Systems, and Computer-Integrated Manufacturing*, Prentice-Hall Inc., New Jersey, 1987.

MARHEFKA, W. D., ORIN, E. D., “Xanimate: AN Educational Tool For Robot Graphical Simulation”, *IEEE Robotics and Automation Magazine*, pp. 6-14, 1996.

NOF, S. Y., Industrial Assembly, Chapman & Hall, 1 ed., London, 1997.