DEPARTAMENTO DE

ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES

Simulação e Programação *Off-line* de Robôs de Montagem

Manuel Fernando dos Santos Silva

FACULDADE DE ENGENHARIA

UNIVERSIDADE DO PORTO

Rua dos Bragas, 4099 Porto Codex – PORTUGAL

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Simulação e Programação *Off-line* de Robôs de Montagem

Manuel Fernando dos Santos Silva

Licenciado em Engenharia Electrotécnica e de Computadores
pela Faculdade de Engenharia da Universidade do Porto

Dissertação submetida para satisfação parcial dos
requisitos do grau de mestre
em
Engenharia Electrotécnica e de Computadores
(Área de especialização de Informática Industrial)

Dissertação realizada sob a supervisão de
Professor Doutor Carlos Manuel de Araújo Sá,
do Departamento de Engenharia Electrotécnica e de Computadores
da Faculdade de Engenharia da universidade do porto

Porto, Setembro de 1996

Manuel Fernando dos Santos Silva

mfs@ccp.up.pt

Aos meus Pais

que sempre defenderam que

"o saber não ocupa lugar"

Resumo

Na indústria existe uma apetência crescente por sistemas flexíveis de montagem automática, como resposta a desafios de ciclos de vida curtos, a *product mixes* cada vez mais variados, a mudanças mais rápidas e a lotes de tamanho cada vez menores. A utilização de robôs, aproveitando toda a flexibilidade inerente a estes equipamentos, pode ser uma grande vantagem especialmente para indústrias de pequena e média dimensão, tendo uma produção caracterizada por pequenos volumes de itens, muito variados.

Pode ser alcançada uma elevada produtividade se o robô não for utilizado na fase de programação, que pode ser múltipla e necessária quando os produtos são modificados de forma a satisfazer os requisitos do mercado. Pelos motivos apresentados, a simulação de robôs e de células robotizadas é um aspecto que tem recebido uma grande importância na área de investigação em robótica.

Particularizando ainda mais, na área da simulação e programação *off-line* de robôs alguns dos principais progressos têm-se centrado ao nível do desenvolvimento de planeadores de montagem que necessitem do mínimo de informação possível para executarem as respectivas tarefas. O planeamento semi-automático da montagem, usando um modelo do produto (modelo geométrico) criado por um sistema de CAD, é uma forma de melhorar a flexibilidade do planeamento. Este objectivo só pode ser alcançado quando o plano de montagem é projectado em conjunto com o próprio produto e actualizado quando o produto é modificado.

Poderemos concluir que, nesta área o paradigma é o desenvolvimento de um sistema integrado, em que seja completamente transparente o fluxo de informação entre sistemas de CAD (onde se projectam os componentes da montagem), o planeador (que desenvolve o plano para a montagem dos componentes modelizados no CAD, transformando esse plano em instruções do equipamento que o deverá executar), e o *software* de simulação (que permite testar o programa desenvolvido).

Esta Tese de Dissertação aborda estes problemas, apresentando a arquitectura genérica dos produtos de simulação dedicados à programação *off-line* de robôs e referindo as suas vantagens sobre os meios convencionais de projecto de células robotizadas e de programação dos seus equipamentos. Após esta primeira abordagem ao problema, são referidos os passos geralmente seguidos para o desenvolvimento de uma aplicação robotizada, desde o projecto da célula (usando *software* de simulação de robôs), até à execução do programa (gerado *off-line*) no robô real que se encontra na planta fabril e monitorização do seu funcionamento. Seguidamente refere-se a forma como pode ser realizada a integração do planeamento de tarefas de montagem com a geração das instruções para os equipamentos que as devem realizar (concretamente, robôs afectos a tarefas de montagem).

Por último, é abordada a implementação concreta destes aspectos ao caso de um robô SCARA, existente na Célula de Montagem do CCP - Centro de CIM do Porto

Abstract

There is a growing need in industry for automated and flexible assembly systems as an answer to smaller life cycles, varied product mixes, fastest changes and smaller sized batches. The use of robots, by taking advantage of all the flexibility that is inherent to these equipments, may be an enormous advantage specially for industries with small and medium dimension, having a production characterised by small volumes of items, with great differences between them.

Increased productivity can be achieved if the robot isn't used during its programming, which can be multiple and needed when the products are modified in order to satisfy the market requests. For the reasons above stated, robot and workcell simulation is an aspect which has received a great importance in robotics R&D

Particularly in robot simulation and off-line programming some of the main progress have been focused in the development of assembly planners that need minimum information in order to execute their respective tasks. The semi-automatic assembly planning, using a product model (geometric model) created by a CAD system, is one way of improving the planning flexibility. This goal can only be achieved when the assembly plan is projected with the product itself and updated when the product is modified.

We may conclude that the paradigm in this area is the development of an integrated system, in which the flow of information, between CAD systems (where the assembly parts are projected), the planner (which develops the assembly plan for the parts that were modelled in the CAD, transforming that plan in specific instructions for the equipment that must execute it), and the simulation software (which allows to test the developed program), is completely transparent to the user.

This Thesis deals with these problems, presenting the generic arquitecture of simulation products devoted to off-line programming of robots and stating its advantages over conventional means of workcell project and programming of their equipments. After this first approach to the problem, the steps generally followed to the development of a robotic application, from the cell project (using robot simulation software) to the program execution (off-line generated) in the real robot which is in the shop floor and the monitoring of its state, are presented. After this it is refered the way how the integration between assembly planning and the generation of the equipment programs that must implement those plans (particularly, robots working on assembly tasks) is accomplished.

Finally, the specific implementation of these aspects to the case of a SCARA robot, which exists in the CCP - Centro de CIM do Porto Assembly Cell, is treated.

Agradecimentos

Em primeiro lugar gostaria de agradecer aos meus pais o facto de me terem possibilitado a realização da Licenciatura e todo o incentivo e apoio que sempre me deram na prossecução dos meus estudos.

Agradeço ao meu orientador, o Prof. Dr. Carlos Manuel de Araújo Sá a sua orientação e coordenação durante a realização dos trabalhos conducentes à elaboração desta Tese de Dissertação, assim como todo o trabalho que teve durante a sua orientação. Agradeço ainda o esforço desenvolvido na leitura e as sugestões de revisão que permitiram o enriquecimento do texto desta Tese.

Um agradecimento muito especial à Paula pela sua companhia durante todo o tempo e também pela sua preciosa colaboração e ajuda na edição e revisão da Tese.

Desejo também agradecer ao CCP - Centro de CIM do Porto, na pessoa do seu Director, Prof. Dr. Francisco Restivo, as facilidades que me concedeu e que permitiram a realização do trabalho conducente à elaboração desta Tese de Dissertação, assim como o facto de ter disponibilizado todos os meio materiais ao seu alcance que permitiram a evolução deste trabalho. Espero que continuem a defender, tal como Joe Carter da Andersen Consulting que "A base da vantagem competitiva, na década de 1990, é quão bem uma empresa adquire e utiliza os seus activos de conhecimento".

Por último, desejo agradecer a todos os meus colegas do CCP, todas as trocas de ideias que tivemos e o óptimo ambiente de trabalho que têm criado. Um agradecimento muito especial ao Sr. Victor Monteiro, por todo o apoio dispensado durante o estágio e por estar sempre disponível quando o equipamento resolvia deixar de funcionar nas alturas menos convenientes.

Índice

Índice de Figuras

Índice de Tabelas

Tabela de Acrónimos

AGV	Automatic Guided Vehicle
AS/RS	Automatic Storage/Retrieval System
BOM	Bill Of Materials
CAD	Computer Aided Design
CAD*I	CAD Interfaces
CAE	Computer Aided Engineering
CAM	Computer Aided Manufacturing
CATM	Célula de Armazenamento e de Transporte de Materiais
CCD	Charge Coupled Device
CCF	Controlador da Célula de Fabrico
CCM	Controlador da Célula de Montagem
CFF	Célula de Fabrico Flexível
CIM	Computer Integrated Manufacturing
CLDATA	Cutter Location DATA
CM	Célula de Montagem
CNC	Computer Numerical Control
CNR	Consiglio Nazionale delle Ricerche
CPAF	Célula de Paletização e de Aferição de Ferramentas
CSG	Constructive Solid Geometry
DFA	Design For Assembly
DFM	Design For Manufacturing
ESPRIT	European Strategic Program for the Research in Information Technologies
GSL	Graphic Simulation Language
ICR	Intermediate Code for Robots
IGES	Initial Graphics Exchange Specification
IGRIP	Interactive Graphics Robot Instruction Program
IMPRES	IMplicit PRogramming Expert System

IRDATA	Industrial Robot DATA
IRL	Industrial Robot Language
ISO	International Standards Organization
I&D	Investigação e Desenvolvimento
JIS	Japan Industrial Standards
KISMET	KInematic Simulation, Monitoring and off-line programming Environment for Telerobotics
LASER	Light Amplification by Stimulated Emission of Radiation
LED	Light Emitting Diode
NAM	Standards for Machine Tools
NFS	Network File System
NIRO	Neutral Interfaces for RObotics
OPS	Overload Protection System
OSI	Open System Intercommunications
PGP	Planeamento e Gestão da Produção
PLR	Programming Language for Robots
PMEs	Pequenas e Médias Empresas
PSI	Gesellschaft für Prozeβsteuerungsund Informationssysteme mbH
QUEST	QUeuing Event Simulation Tool
ROPSIM	Robot Off-line Programming and real-time SIMulation system
SCARA	Selective Compliance Assembly Robot Arm
SFC	Shop Floor Controller (Controlador de Oficina)
STD	State Transition Diagrams
STEP	Standard for the Exchange of Product Data
STROLIC	STandard RObot Language in Intermediate Code
SVA	Sistema de Visão Artificial
TCP	Tool Center Point
VME	Virtual Machine Extension BUS bar

Capítulo 1

Introdução

Neste capítulo procura-se dar resposta às seguintes questões:

- Quais são os problemas com que a indústria se debate;
- Quais são os problemas específicos dos processos de montagem;
- O porquê da simulação e programação *off-line* de robôs e sua aplicação à montagem robotizada;
- Quais os objectivos do trabalho realizado;
- Qual foi a solução adoptada e como foi implementada;
- Qual a estrutura desta Tese de Dissertação.

Todos os dias se ouve falar na televisão, na rádio e nos jornais, no incremento das trocas comerciais entre países, países estes cada vez mais distantes fisicamente entre si, na maioria dos casos situados mesmo em continentes diferentes. Este facto, a livre circulação de mercadorias entre países, assim como a redução das tarifas de transporte a nível mundial, bem como a rapidez com que as mercadorias chegam de um extremo do globo ao outro, muito têm contribuído para que a concorrência entre produtos tenha de ser encarada numa escala global, o que exige das empresas a diferenciação dos seus produtos em relação aos produtos da concorrência, se querem ser bem sucedidas.

Mas não se trata só de diferenças nos produtos. Esta envolvente leva a que os mercados actuais requeiram uma cada vez maior variedade de produtos, com requisitos tecnológicos muito diferentes, com qualidade superior, desenvolvidos em menor tempo e a um custo cada vez mais reduzido, o que leva a que os processos de fabrico se caracterizem por lotes de produtos cada vez menores, a uma diminuição do ciclo de vida dos produtos (Figura 1.1) e ao fabrico dos produtos em sequências opcionais, de forma a que se consiga satisfazer o mercado com qualidade e a tempo e horas. É toda a filosofia produtiva que se encontra em alteração.

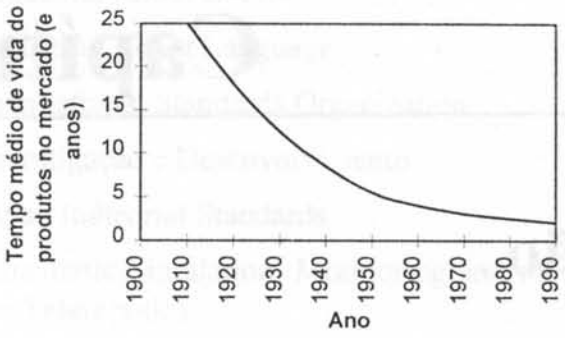


Figura 1.1: Evolução do ciclo de vida dos produtos [Rampersad, 1994]

Esta necessidade de diminuir os custos dos produtos e o tempo de lançamento de novos produtos no mercado colocam problemas ao nível de todo o processo produtivo. Estes, no entanto, fazem-se sentir de uma forma bastante aguda numa parte específica do processo produtivo: a montagem.

Isto mesmo tem sido realçado por vários autores, sendo de destacar um estudo organizado pelo Departamento de Defesa Norte-Americano [Martin-Vega, 1995] em cujas conclusões é afirmado que os processos unitários que dominam os custos da produção de um produto são a montagem (com 20% dos custos do produto) e a inspecção.

A mesma opinião é expressa por Rampersad (1994), que defende que a montagem é geralmente o elo mais fraco de todo o processo produtivo, uma vez que esta actividade consome uma parte substancial dos custos e do tempo total de produção. As razões que apresenta para este facto são o aumento dos custos do trabalho, bem como o aumento da variedade e a diminuição do tamanho das séries dos produtos.

1 A montagem robotizada

Uma forma que Rampersad (1994) encontra para resolver estes problemas passa pela racionalização das actividades de montagem, especialmente através da utilização crescente de células flexíveis de montagem robotizadas.

Na Figura 1.2 representa-se graficamente o que acabou de ser dito. Pode-se ver nesta figura uma indicação de qual o método de montagem mais adequado em função das características do processo produtivo. Observa-se aí que, à medida que o tamanho dos lotes e o volume de produção vão diminuindo, e a variedade dos produtos e a flexibilidade pretendida vão aumentando, é necessário passar das técnicas de montagem mecanizadas para a montagem robotizada.

Também Martin-Vega (1995), no estudo já anteriormente referido, revela que existe uma grande necessidade de Investigação e Desenvolvimento (I&D) nas áreas dos sistemas automatizados e flexíveis de montagem e na área da montagem robotizada.

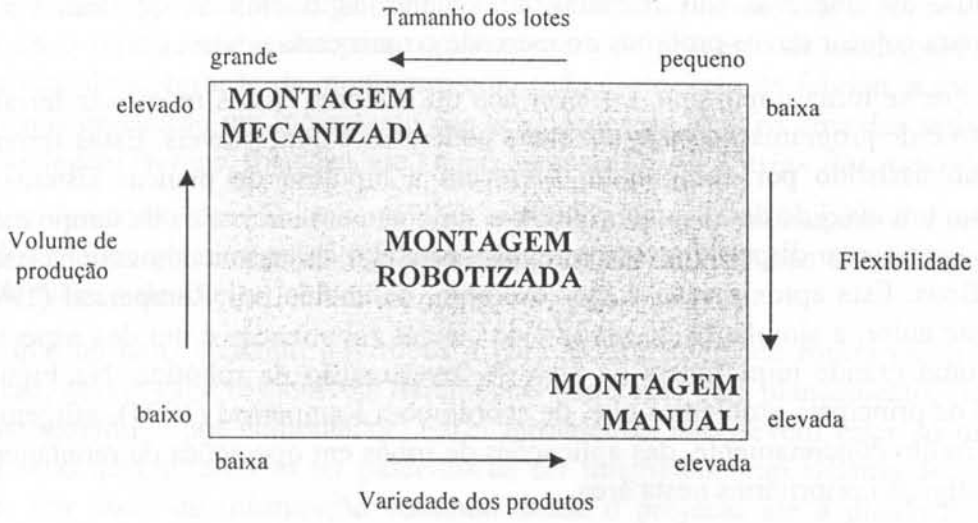


Figura 1.2: Métodos de montagem vs. características da produção [Rampersad, 1994]

Conclui-se assim, rapidamente, que todas estas alterações da situação do mercado pedem um maior grau de automação e integração dos processos de fabrico e, especialmente, dos processos de montagem. Para se alcançar uma amortização interessante dos investimentos requeridos, apesar da diminuição dos ciclos de vida dos produtos, esta automação tem que se caracterizar por uma maior produtividade e flexibilidade, levando à utilização de sistemas de fabrico e de montagem flexíveis.

Flexibilidade, ou o seu sinónimo adaptabilidade, é aqui visto em termos da capacidade do sistema suportar modificações em todos os estádios do processo produtivo, nomeadamente alterações nos requisitos dos produtos, alterações nas tecnologias e nos processos, e certa capacidade de recuperação de erros (anomalias). Outro ponto de vista da flexibilidade está relacionado com tempos de *setup* curtos, no sentido da redução do esforço de programação.

Neste contexto pode-se afirmar que os robôs jogam um papel importante como componentes de automação mais flexíveis. Isto também se pode concluir analisando o enorme aumento que se tem verificado, e continua a verificar, nas aplicações de robôs [Lloyd, 1994] e, mais concretamente, nos robôs afectos a tarefas de montagem de produtos pequenos [Philips, 1994] [Bekey, 1996]. Concretamente nos E.U.A., o aumento das vendas de robôs foi de 30% ao ano, nos últimos dois anos, devendo-se registar um aumento ainda superior este ano [Bekey, 1996].

2 Implicações da robotização dos processos de montagem

No entanto, a extensão da *performance* destes sistemas requer que um grande número de equipamentos, componentes e ferramentas estejam disponíveis. Daí que os futuros sistemas de fabrico venham a ser caracterizados por um elevado grau de complexidade.

Este aumento da complexidade dos dispositivos, do fluxo de materiais e das comunicações entre dispositivos, aumenta os esforços de planeamento e de programação necessários, o tempo necessário para a sua realização e os custos da operação dos robôs. Por outro lado,

como já vimos, as empresas são forçadas a executar as tarefas de planeamento mais rapidamente, para colocar novos produtos no mercado o mais cedo possível.

Vemos assim que se torna importante fornecer aos utilizadores destas máquinas ferramentas de planeamento e de programação cada vez mais poderosas e confortáveis. Estas ferramentas de planeamento assistido por computador oferecem a hipótese de planear sistemas mais complexos, com um nível de detalhe mais elevado, tudo isto em intervalos de tempo menores. De facto, estão-se a tornar disponíveis aproximações interactivas, essencialmente baseadas em interfaces gráficos. Esta aproximação é, por exemplo, defendida por Rampersad (1994): de acordo com este autor, a simulação de robôs e de células robotizadas é um dos aspectos que deve receber uma grande importância na área de investigação da robótica. Na Figura 1.3, apresentam-se os principais problemas que, de acordo com Rampersad (1994), afligem a área da robótica e, muito concretamente, das aplicações de robôs em operações de montagem, e as linhas de investigação prioritárias nesta área.

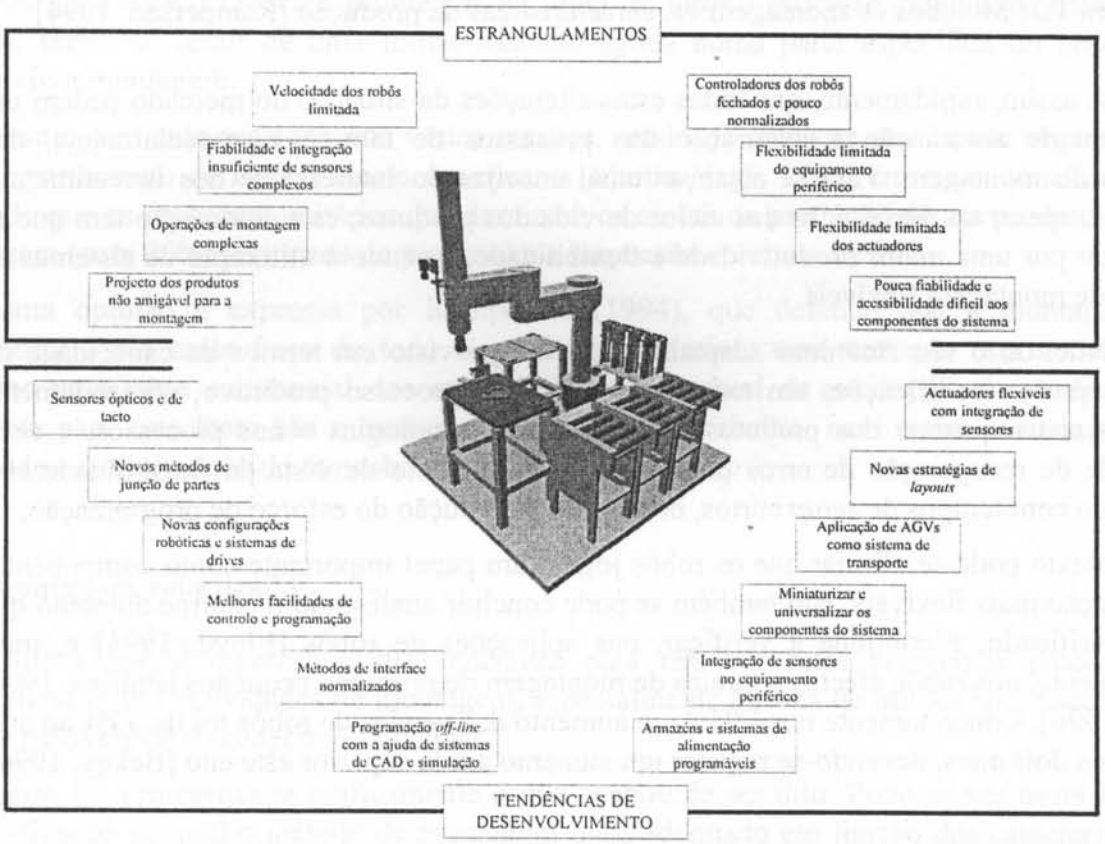


Figura 1.3: Principais estrangulamentos e tendências de desenvolvimento na robótica de montagem (Adaptado de [Rampersad, 1994])

O mesmo defende Caracciolo (1995), segundo o qual uma flexibilidade completa requer a evolução dos sistemas de programação *off-line*, bem como a realização eficiente e fiável de sistemas de planeamento ao nível da tarefa, directamente relacionados com as fases do projecto e da programação.

No estudo efectuado por Martin-Vega (1995), referido anteriormente, este autor sugere que os esforços de I&D sejam canalizados para o planeamento do processo de montagem e também para os aspectos de integração da montagem com todo o processo de fabrico, a montante e a jusante. É ainda defendido um incremento das actividades de I&D na área das actividades de suporte à montagem (*setups*, fixações, etc.) e nas áreas de DFA - *Design For Assembly*.

Por último, e de acordo com Gutsche (1995), o planeamento da montagem e a subsequente execução dos planos gerados pelos robôs vai ser uma das tecnologias chave dos processos modernos de fabrico flexível.

Conclui-se que no caso concreto dos robôs e para se utilizarem de forma efectiva as suas características, devem estar disponíveis ferramentas poderosas para planeamento, simulação e programação assistidos por computador. Estas ferramentas não devem estar só disponíveis para apoiar, mas devem também ser passíveis de ser integradas num sistema de informação que garanta um fluxo de informação contínuo desde o projecto até à planta fabril (*Shop Floor*). Isto é por vezes designado por integração da informação e é provavelmente o aspecto mais importante coberto pelo termo CIM (*Computer Integrated Manufacturing*).

Os benefícios que resultam de um sistema integrado para o planeamento da montagem, para a simulação da execução das tarefas de montagem a ser efectuadas e para a geração do código para os robôs, são menores custos de desenvolvimento de aplicações do robô, menores riscos de acidentes e uma melhor exploração da tecnologia dos robôs [Bernhardt, 1992].

3 A programação de robôs

Tal como para qualquer outro equipamento programável, o objectivo de um sistema de programação de robôs é a geração de um programa de controlo desse robô de uma forma simples e amigável para o utilizador. No entanto, quando comparado com a programação de outras máquinas controladas numericamente, a programação de robôs apresenta certas particularidades [Bernhardt, 1992]:

- diz respeito à geração de movimentos, na maioria das vezes bastante complexos. Estes movimentos são muito difíceis de imaginar para os seres humanos, sem a ajuda do próprio robô ou de um sistema de simulação e animação gráficas;
- os dados geométricos do mundo real (por exemplo, posições e orientações dos componentes a serem manipulados) apresentam desvios significativos dos dados usados no seu modelo para programação *off-line*; daí que muitas vezes sejam usados sensores para medir tais quantidades geométricas (os erros). Estas leituras são a base de um programa condicional para compensação dos desvios;
- o robô tem que estar sincronizado com dispositivos periféricos, tais como: sensores, actuadores finais, alimentadores, fixações, etc.

Obviamente, esta lista implica requisitos consideráveis para um sistema de programação de robôs. Adicionalmente, o programador do robô deve ser capaz de especificar as operações espaciais do robô de uma forma simples. As técnicas de programação de robôs devem ser adaptadas à maneira de pensar do utilizador, para facilitar a descrição das operações espaciais.

O utilizador pretende programar o robô de uma forma orientada ao problema, sem ter necessidade de medir as coordenadas na célula de trabalho e sem ter de as explicitar especificamente nos programas que desenvolve. Além do mais, o sistema tem que suportar a capacidade de manusear interrupções e permitir a interacção entre o controlador do robô e as máquinas periféricas.

Para sumariar, pode-se dizer que os métodos de programação de robôs devem apresentar as seguintes características:

- suficientemente simples para serem facilmente apreendidos e manipulados;
- orientados ao problema: o método de programação deve ser adaptado à imaginação do utilizador e à aplicação particular;
- suportar a programação *off-line* do robô, de forma a evitar o mais possível o recurso à célula do robô (por norma, extremamente cara) para efectuar a sua programação.

4 Objectivo do trabalho

No contexto anterior foi escolhido, como problema a resolver, a geração de programas para um robô que se encontra a realizar tarefas de montagem.

Para o efeito, fixou-se como objectivo do trabalho o desenvolvimento de uma metodologia, baseada num conjunto de pacotes de *software* de CAD/CAE (*Computer Aided Design/Computer Aided Engineering*), para efectuar a programação *off-line* de um robô SCARA (*Selective Compliance Assembly Robot Arm*) utilizado em tarefas de montagem, cujo funcionamento é caracterizado pela montagem de pequenas séries de produtos com características bastante diferentes.

Sem prejuízo de se pretender que o trabalho desenvolvido comporte um conjunto de ideias base com aplicação mais geral, para efeitos da sua implementação utilizou-se o caso concreto da Célula de Montagem pertencente à plataforma CIM disponível no CCP - Centro de CIM do Porto.

5 Solução proposta

O facto de o robô a programar ser caracterizado por trabalhar em montagens de séries de pequenas quantidades de produtos leva à necessidade de frequentes paragens do processo produtivo, para reprogramações do robô, podendo estas paragens do processo produtivo ser bastante demoradas. Como existe a necessidade de frequentes reprogramações, o facto de estas reprogramações serem *on-line* leva também à ocorrência de maiores riscos potenciais para o robô em causa, riscos estes provocados pela possível ocorrência de acidentes (com o próprio robô ou com o operador que se encontra responsável pela sua reprogramação).

No caso estudado optou-se por uma solução em que o robô não necessita de ser utilizado durante a sua programação: esta é efectuada *off-line*, recorrendo a um pacote de *software* com capacidade de simulação da célula em que o robô se encontra inserido. Optou-se por uma

programação recorrendo a macro-instruções, de forma a que o utilizador responsável por esta tarefa não necessite de conhecer a linguagem de programação do robô em questão para gerar o código do programa requerido por uma determinada operação de montagem.

Para alcançar este objectivo propõe-se uma solução baseada num pacote de *software* de simulação e programação *off-line* de robôs, sendo a informação relativa à montagem a efectuar proveniente, em parte, do sistema de CAD disponível no CCP - Centro de CIM do Porto (produto Pro/Engineer da Parametric Technology Corporation, Inc.).

6 Organização da Tese

Esta Tese de Dissertação está organizada de acordo com a estrutura seguida no desenvolvimento do trabalho realizado.

Assim, no Capítulo 2 abordam-se os princípios da simulação, concretamente na sua vertente de aplicação à programação *off-line* de robôs. Vão ser descritas as principais características e módulos que constituem este tipo de pacotes de *software*, sendo ainda feita uma apresentação das características principais de alguns destes pacotes de *software* disponíveis comercialmente.

No capítulo seguinte, o Capítulo 3, vamos apresentar os principais passos a seguir no desenvolvimento de um programa para um robô, qualquer que seja a aplicação em vista, e a forma como podem ser automatizados alguns desses passos. Dar-se-á ainda algum relevo ao estado de desenvolvimento dos interfaces neutros para troca de informação entre passos subsequentes do desenvolvimento das aplicações.

Seguidamente, no Capítulo 4, vamos concretizar alguns dos aspectos referidos no Capítulo 3, nomeadamente ao nível da integração do planeamento das tarefas com o desenvolvimento do programa para o robô, sendo este problema delimitado ao caso dos robôs que realizam tarefas de montagem. Neste capítulo vão ser também referidas algumas abordagens que têm sido seguidas na resolução deste problema.

Chegados a este ponto, no Capítulo 5 passa-se a descrever a Célula de Montagem disponível na Oficina Piloto do CCP - Centro de CIM do Porto, na qual foi realizado o trabalho de implementação da solução adoptada. Esta Célula vai ser aqui modelizada com vista a uma fácil compreensão do seu funcionamento e das trocas de informação existentes entre esta e as outras células constituintes da Oficina Piloto do CCP.

No Capítulo 6 especifica-se o problema que nos foi colocado e descreve-se a solução que foi por nós adoptada.

Após este passo passa-se à descrição da implementação de tal solução, o que é feito no Capítulo 7.

Esta Tese de Dissertação termina com um conjunto de conclusões retiradas do trabalho realizado, onde se apresentam quer as vantagens quer as limitações do mesmo, e onde se deixam ainda algumas reflexões sobre possíveis desenvolvimentos futuros deste trabalho. Tais conclusões são apresentadas no Capítulo 8.

Capítulo 2

Simulação aplicada à programação *off-line* de robôs

Neste capítulo procura-se dar resposta às seguintes questões:

- Como evoluiu a programação de robôs;
- Princípios da simulação;
- Quais as capacidades dos pacotes de *software* de simulação de robôs;
- Qual a sua arquitectura genérica;
- Breve comparativo dos pacotes de simulação para a programação *off-line* de robôs disponíveis comercialmente.

Tal como foi visto no capítulo anterior, são de importância essencial para qualquer sistema integrado de produção métodos poderosos e versáteis de programação de robôs.

O desenvolvimento histórico dos robôs, começando com simples dispositivos de *pick-and-place* e indo até aos dispositivos actuais bastante mais sofisticados, foi acompanhado pelo desenvolvimento concorrente de métodos de programação com potência e complexidade cada vez maiores. No entanto, os métodos de programação dos robôs actuais cumprem apenas alguns dos requisitos descritos no Capítulo 1, pelo que o seu desenvolvimento continua.

Nos nossos dias começam a ser utilizados, com frequência cada vez maior, métodos de simulação e animação gráfica para se efectuar a programação *off-line* de robôs. Este é o método que apresenta maiores potencialidades de crescimento, capaz de se impor como o método corrente de programação de robôs. No entanto, só nos últimos anos é que este tipo de métodos começa a ser desenvolvido e utilizado, tendo a programação de robôs sido efectuada de modo muito diferente durante os primeiros anos da robótica.

1 Evolução da programação de robôs

1.1 Programação *on-line*

A programação dos robôs começou por ser efectuada na linha de produção, utilizando para o efeito o próprio robô para efectuar a sua programação. Daí que o termo programação *on-line* seja utilizado quando o método de programação envolve directamente o próprio robô. A actividade principal da programação *on-line* é a definição de trajectórias. Este método de programação pode ainda ser dividido em manual ou automático.

Na programação *on-line*, o utilizador leva o robô às posições que pretende memorizar, quer manualmente (costuma também ser designado de *leadthrough programming* ou programação manual), quer utilizando um dispositivo do tipo *joy-stick*, denominado *teach-pendant* (programação automática).

A programação manual costuma ser utilizada quando se pretende definir uma sequência de pontos no espaço ou uma trajectória complexa. Esta forma de programação tem sido utilizada, por exemplo, em robôs que estão afectos a tarefas de pintura, onde a função do programador é mover o punho do robô ao longo da trajectória de pintura, de forma a que este a "aprenda".

A programação automática é mais utilizada quando se pretende definir alguns pontos discretos e específicos no espaço, por exemplo em aplicações de *pick-and-place*.

Por vezes, além da memorização do ponto, o utilizador associa a cada localização no espaço a velocidade com que esta deve ser alcançada e o tipo de movimento que deve ser efectuado para se alcançar esta posição (movimento ponto-a-ponto, linear ou circular).

A vantagem destes métodos de programação é o facto de serem bastante simples de realizar e de aprender. Como desvantagem principal apresenta-se o facto de o robô ser necessário para realizar a programação, o que implica a paragem da célula em que se encontra o robô durante esta fase, com as consequentes perdas de produtividade que dai advêm; além disso, podem surgir colisões do robô com outros equipamentos da célula em que este se encontra, devido a erros de programação (como, por exemplo, erros ou distracções do operador).

1.2 Programação *off-line*

Para ultrapassar os inconvenientes da programação *on-line* acabados de referir, especialmente o facto de a célula ter de estar parada durante a programação do robô, tem-se evoluído para a programação *off-line* que, tal como o seu nome indica, se refere à possibilidade de programar o robô sem ser necessária a sua utilização durante esta tarefa.

Este método de programação é cada vez mais utilizado, especialmente para aplicações complexas que exigem longos tempos para desenvolvimento dos programas. A programação *off-line* implica a construção do texto de um programa, que é editado sem a presença efectiva do robô e é depois traduzido ou interpretado.

1.2.1 Programação *off-line* usando editor e compilador de programas para o robô

Inicialmente começou-se por efectuar a programação *off-line* recorrendo a uma solução mista de editor e compilador de programas, solução esta na qual o código do programa para o robô era gerado recorrendo a um PC, sendo a sequência de operações da trajectória de trabalho do robô, e do seu actuador final, escrita com instruções textuais, tal como se se estivesse a desenvolver um programa em linguagem Pascal ou C.

No entanto, existe um problema com este método de programação textual: como especificar os pontos de movimento sem o robô.

Uma aproximação possível para este problema passa por o operador programar também os pontos para onde o robô se deve deslocar, utilizando para o efeito as instruções de programação próprias do robô e indicando as coordenadas de tais pontos. Esta aproximação é bastante complicada, pois é extremamente difícil ao operador conhecer com precisão suficiente as coordenadas dos pontos para onde o robô se deve deslocar para efectuar uma dada operação. Alternativamente a esta aproximação, o operador pode desenvolver todo o código do programa e, depois de esta tarefa estar concluída, ensinar ao robô as posições necessárias para ele cumprir as suas tarefas através de um método de programação *on-line*. Portanto, quase todos os sistemas de programação textual *off-line* são também fornecidos com um método de aprendizagem (*teach-in*).

Conclui-se que neste método de programação, por vezes designado de programação textual, não é necessária a utilização do robô para a tarefa de programação, sendo no entanto necessário para testar o programa desenvolvido e, por vezes, para memorizar as posições do trajecto que tem que seguir durante a execução do programa.

1.2.2 Programação *off-line* usando sistemas de simulação

Para ultrapassar esta situação e para simplificar a programação *off-line*, diminuindo o tempo necessário ao desenvolvimento dos programas, têm sido aplicadas ferramentas gráficas computorizadas sob a forma de sistemas de simulação e animação.

A principal tarefa de um sistema de simulação num sistema de programação *off-line* é o teste e verificação dos programas de aplicação que foram criados. Portanto, o simulador deve permitir a detecção de erros nestes programas.

Inicialmente, estes sistemas apenas permitiam testar os programas desenvolvidos para os robôs. Os gráficos computorizados eram usados, por exemplo, para simular num ecrã gráfico os efeitos de um programa escrito *off-line*.

Ainda mais amigáveis para o utilizador são os sistemas de programação gráfica interactivos. Estes sistemas permitem actualmente desenvolver os programas no ambiente simulado e ver imediatamente como o robô executa cada instrução de uma forma interactiva. Estes sistemas de simulação de última geração já permitem, também, a definição das trajectórias do robô e a descarga para o robô, depois de testados, quer do código do programas, quer das trajectórias, podendo os programas ser executados no robô sem necessitarem de correcções.

Pode-se ver na figura seguinte (Figura 2.1) o interface com o utilizador de um pacote de *software* deste tipo.

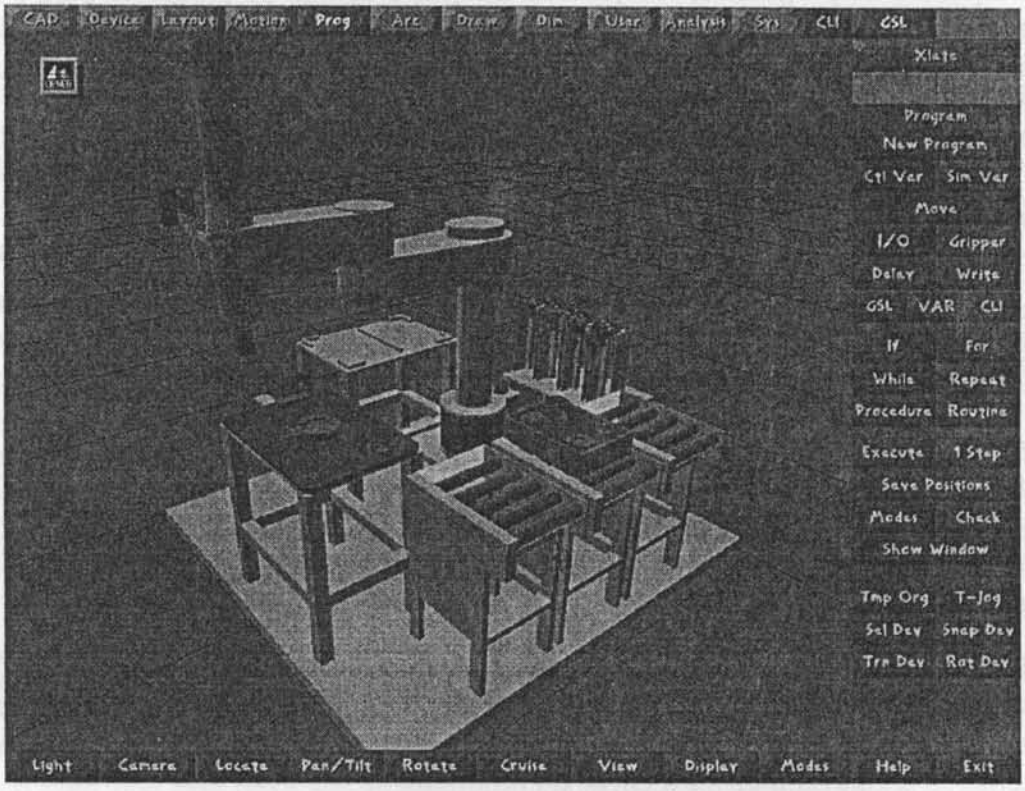


Figura 2.1: Interface do *software* IGRIP (Deneb Robotics, Inc.)

Após termos visto como evoluiu a programação de robôs, vamos agora dedicar uma atenção especial aos sistemas gráficos de simulação e animação. Neste caso concreto vamos apresentar uma arquitectura genérica destes produtos, vendo como funcionam os seus módulos principais e quais as suas reais capacidades, não sem antes fazer uma pequena introdução aos métodos de estudo de sistemas reais, dos quais a simulação é apenas um deles.

A título de conclusão apresentam-se as vantagens que estes produtos podem trazer no desenvolvimento de aplicações robotizadas.

2 A simulação de processos produtivos

2.1 Fundamentos de simulação

Em determinados momentos da vida de muitos sistemas, existe a necessidade de os estudar de forma a tentar obter algum conhecimento sobre as relações entre os vários componentes desses sistemas ou de forma a prever a sua *performance* debaixo de novas condições em consideração.

A Figura 2.2 apresenta as várias formas segundo as quais os sistemas podem ser estudados.

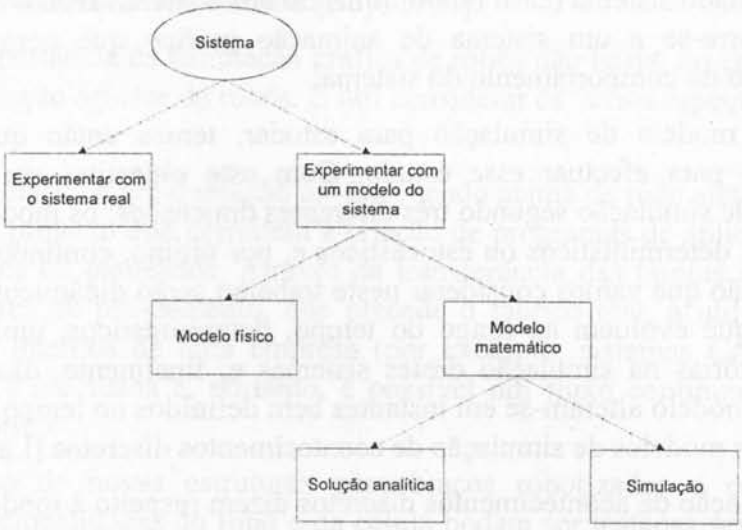


Figura 2.2: Formas de estudar um sistema [Law, 1991]

O objecto ou processo de interesse que se pretende estudar costuma ser designado por sistema e de forma a poder ser simulado temos muitas vezes que fazer um conjunto de pressupostos acerca do seu funcionamento. Estes pressupostos constituem o modelo que é utilizado para tentar obter uma melhor compreensão do funcionamento do sistema.

Se for possível alterar o sistema fisicamente e depois operá-lo debaixo das novas condições, é provavelmente desejável fazê-lo, uma vez que neste caso não se levantam nenhumas questões sobre se o que está em estudo é ou não relevante. No entanto, raramente esta aproximação é realizável, uma vez que uma tal experiência seria na maioria das vezes demasiado custosa. Além de que por vezes o sistema que se pretende estudar ainda nem sequer existe fisicamente. Por estes motivos, é usualmente necessário construir um modelo, como uma representação do sistema, e efectuar os estudos pretendidos sobre tal modelo.

Quando se opta pelo estudo de um modelo do sistema temos duas opções: estudar um modelo físico ou efectuar o estudo sobre um modelo matemático. Um modelo físico é uma representação palpável do sistema, muitas vezes com dimensões mais reduzidas, contrariamente ao modelo matemático que representa o sistema em termos de relações lógicas e quantitativas, que são manipuladas e alteradas de forma a ver como é que o modelo reage e, portanto, como é que o sistema responde.

Uma vez construído um modelo matemático, este deve então ser examinado para ver de que forma pode ser utilizado para responder às questões de interesse acerca do sistema que é suposto representar. Se o modelo for suficientemente simples, pode ser possível trabalhar com as suas relações e quantidades de forma a obter uma solução exacta ou analítica. Se, pelo contrário, for impossível obter uma solução analítica do modelo matemático porque o sistema é demasiado complexo, o modelo deve então ser estudado recorrendo à simulação, i.e., exercitar o modelo numericamente para as entradas em questão de forma a ver como elas afectam as medidas de *performance* da saída.

A simulação pode ser uma simulação matemática ou uma simulação gráfica. No primeiro caso, costuma-se recorrer a expressões matemáticas com vista a tentar determinar o comportamento de um dado sistema (caso muito utilizado nos sistemas físicos e económicos); no segundo caso recorre-se a um sistema de animação gráfico que permite visualizar, virtualmente, a evolução do comportamento do sistema.

Dado que temos um modelo de simulação para estudar, temos então que procurar as ferramentas adequadas para efectuar esse estudo. Com este objectivo em mente, é útil classificar os modelos de simulação segundo três diferentes dimensões: os modelos podem ser estáticos ou dinâmicos, determinísticos ou estocásticos e, por último, contínuos ou discretos. Os modelos de simulação que vamos considerar neste trabalho serão dinâmicos, uma vez que representam sistemas que evoluem ao longo do tempo, determinísticos, uma vez que não existem variáveis aleatórias na simulação destes sistemas e, finalmente, discretos, pois as variáveis de estado do modelo alteram-se em instantes bem definidos no tempo. Estes tipos de modelos são designados modelos de simulação de acontecimentos discretos [Law, 1991].

Estes modelos de simulação de acontecimentos discretos dizem respeito à modelização de um sistema que evolui ao longo do tempo através de uma representação na qual as variáveis de estado se alteram automaticamente em determinados instantes (em termos matemáticos, poderíamos dizer que o sistema só pode mudar num número finito de instantes de tempo). Estes instantes são aqueles nos quais ocorre um acontecimento, em que um acontecimento é definido como uma ocorrência instantânea que pode alterar o estado do sistema.

Apesar de a simulação de acontecimentos discretos poder ser conceptualmente efectuada através da realização de cálculos manuais, a quantidade de dados que têm que ser armazenados, e manipulados, para a maioria dos sistemas do mundo real obrigam a que este tipo de simulações tenha que ser efectuado recorrendo a ferramentas computacionais.

2.2 Simulação gráfica de robôs

As áreas de aplicação da simulação são cada vez mais numerosas [Law, 1991][Banks, 1996]. Nos dias de hoje, por exemplo, a indústria automóvel "destroi" centenas de carros por semana de forma a tentar melhorar os seus projectos - e tudo através de simulações em computador [Wozny, 1996]. Um tal nível de teste e avaliação não seria possível utilizando carros físicos. Um caso particular da utilização da simulação gráfica é na programação de robôs. De agora em diante, vamos passar a utilizar o termo simuladores, querendo com este termo referir os sistemas de simulação gráfica de robôs.

Uma vantagem fundamental destes simuladores nas aplicações industriais é a sua utilização para o desenvolvimento de programas *off-line*. Utilizando estas ferramentas não é necessário interromper o trabalho dos robôs reais na célula de trabalho, geralmente integradas numa linha de produção, podendo esta ser mais ou menos complexa. Isto permite poupanças de dinheiro, porque a produção da planta fabril não necessita de ser interrompida. Permite também diminuir o tempo de desenvolvimento dos produtos, através da realização das tarefas de desenvolvimento dos programas para os equipamentos das células em paralelo com outras actividades de projecto (Engenharia Concorrente).

Além do mais, no sistema de programação *off-line* é geralmente possível efectuar previamente os cálculos necessários antes de gerar o programa do robô. Estes cálculos prévios reduzem a complexidade do programa de movimento [Bey, 1994].

Para perceber a importância da simulação gráfica de robôs não basta, no entanto, considerar o aspecto da programação *off-line* de robôs. É útil considerar os vários aspectos onde a mesma é utilizada e porquê.

Os sistemas de simulação e programação *off-line*, sendo acima de tudo sistemas de simulação, são ferramentas de projecto que permitem a criação de programas de aplicação para sistemas de fabrico existentes ou planeados. Através da transferência das tarefas de programação da aplicação para a área de planeamento, que precede o fabrico real, a utilização de todos os sistemas de dados internos de uma empresa (por exemplo, sistemas CAD, base de dados tecnológicos, etc.) é efectuada e, portanto, é possível um fluxo contínuo de dados desde o Projecto até ao Fabrico.

Durante o projecto de novas estruturas para braços robotizados e células de trabalho robotizadas, as funcionalidades do robô e da célula podem ser testadas sem a necessidade de se construírem protótipos, que na maioria das vezes são extremamente caros [Laloni, 1995]. Adicionalmente, o *layout* das células de trabalho robotizadas pode ser optimizado facilmente, podendo por exemplo determinar-se a posição óptima do braço do robô, das ferramentas ou das peças a trabalhar, dentro da célula de trabalho simulada.

A utilização destes métodos no desenvolvimento das tarefas acima referidas permite reduzir o tempo de desenvolvimento das células robotizadas, quer através da realização de testes para escolher os melhores equipamentos para uma dada tarefa, quer na determinação do melhor *layout* para os equipamentos que vão realizar essa tarefa.

De igual forma, os simuladores fornecem uma segurança maior ao permitirem a verificação prévia das trajectórias dos robôs e ao tornarem possível a detecção de colisões com outros objectos, assim se evitando acidentes. Além do mais, as técnicas de simulação e programação *off-line* protegem o operador de eventuais acidentes provocados quer por erros de programação, quer mesmo por distracção, riscos estes em que o operador incorre ao trabalhar com sistemas de programação *on-line* mais convencionais, devido ao facto de ter que trabalhar na proximidade física do robô. Estas facilidades são ainda mais importantes no caso da sincronização do funcionamento de dois ou mais robôs que tenham de trabalhar em colaboração e cujos volumes de trabalho se intersectem. Esta coordenação de múltiplos robôs, que geralmente requer um planeamento mais cuidadoso, pode ser testada sem perigo, quer para as máquinas quer para o homem.

Por fim, a utilização de sistemas de simulação de robôs não deve ser subestimada nos campos da investigação e da educação [Vendrell, 1995]. Novos algoritmos de controlo podem, por exemplo, ser desenvolvidos e testados muito antes de o correspondente *hardware* estar disponível [Laloni, 1995] e, como aplicação em novas metodologias de ensino, os estudantes podem aprender os princípios da robótica em sistemas de simulação de uma forma barata e segura [Tenreiro, 1995].

2.3 A variável tempo na simulação

O tempo é uma variável muito importante dentro dos sistemas gráficos de simulação de robôs. Especialmente se o tempo da simulação não for idêntico ao tempo simulado, é necessário obter informação correcta acerca do tempo consumido, caso as acções simuladas devam ser executadas num ambiente real.

Devido à natureza dinâmica dos modelos de simulação que estamos a analisar, à medida que a simulação decorre é necessário manter um registo do valor corrente do tempo simulado, sendo também necessário um mecanismo para fazer avançar o tempo simulado de um valor para o seguinte. Num modelo de simulação, chama-se à variável que indica o valor corrente do tempo simulado o relógio da simulação [Law, 1991]. Por norma, não existe nenhuma relação entre o tempo simulado e o tempo necessário para correr uma simulação no computador.

Para efectuar o avanço do relógio da simulação existem basicamente duas aproximações: o avanço do tempo para o próximo acontecimento e o avanço do tempo em incrementos fixos, também conhecidas por simulação orientada à operação e simulação orientada ao tempo, respectivamente.

2.3.1 Simulação orientada à operação

Com a primeira aproximação, (avanço do tempo para o próximo acontecimento), o relógio da simulação é inicializado no valor zero e são determinados os instantes de tempo em que vão ocorrer os próximos acontecimentos. O relógio da simulação é então avançado para o instante de tempo da ocorrência do acontecimento mais iminente (primeiro acontecimento a ocorrer) destes acontecimentos futuros e, neste ponto, o estado do sistema é actualizado de forma a considerar que um acontecimento ocorreu; adicionalmente, é também actualizado o nosso conhecimento relativamente aos tempos de ocorrência dos acontecimentos futuros. O relógio da simulação é então avançado para o instante de tempo do próximo acontecimento iminente (o novo), o estado do sistema é actualizado e os instantes de ocorrência dos futuros acontecimentos são determinados, etc. Este processo de avançar o relógio da simulação de um acontecimento para o outro é continuado até que, eventualmente, seja satisfeita alguma condição de paragem pré-especificada. Uma vez que todas as mudanças de estado só ocorrem nos instantes dos acontecimentos, para um modelo de simulação de acontecimentos discretos os períodos de inactividade são ignorados, fazendo o relógio saltar do instante de ocorrência de um acontecimento para o instante de ocorrência do próximo acontecimento. Deve ficar bem claro que os saltos sucessivos do relógio da simulação são geralmente de tamanho variável ou desiguais, como se pode observar na Figura 2.3.

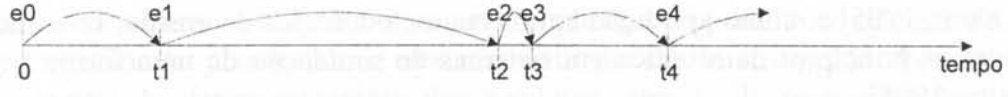


Figura 2.3: Aproximação de avanço do relógio da simulação para o próximo acontecimento [Law, 1991]

Os acontecimentos a considerar podem ser, por exemplo, a conclusão da execução de uma instrução, a ocorrência de uma colisão, entre outras. Este método apresenta como vantagem o facto de o tempo da simulação poder ser muito inferior ao tempo necessário para a execução do programa na planta fabril. Como desvantagem assinala-se o facto de a simulação avançar por "saltos" sem que o utilizador tenha perfeita consciência do que ocorre entre dois instantes de visualização sucessivos.

2.3.2 Simulação orientada ao tempo

A segunda aproximação para efectuar o avanço do relógio da simulação, num modelo de simulação de acontecimentos discretos é, como se disse, designado por avanço do tempo em incrementos fixos. Com esta aproximação, o relógio da simulação é avançado por incrementos fixos de exactamente Δt unidades de tempo, para alguma escolha apropriada de Δt. Após cada actualização do relógio é efectuada uma verificação para determinar se alguns acontecimentos deveriam ter ocorrido durante o intervalo prévio de duração Δt. Se um ou mais acontecimentos tinham sido escalonados para ocorrer durante tal intervalo, estes acontecimentos são considerados como ocorrendo no final do mesmo e o estado do sistema (e os contadores estatísticos) são actualizados em conformidade. A aproximação de avanço do tempo por incrementos fixos é apresentada na Figura 2.4, em que as setas curvas representam o avanço do relógio da simulação e e_i (i = 1, 2, 3, ...) representam o instante de tempo real de ocorrência do i-ésimo acontecimento de qualquer tipo (e não o i-ésimo valor do relógio de simulação).

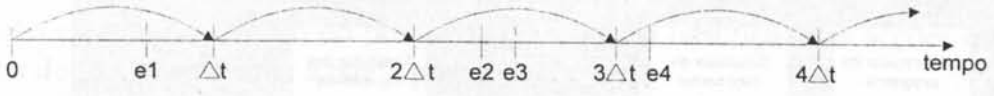


Figura 2.4: Aproximação de avanço do relógio da simulação em incrementos fixos [Law, 1991]

As duas desvantagens mais evidentes que resultam desta aproximação são o facto de os erros introduzidos pelo processamento dos acontecimentos só ser efectuado no fim do intervalo em que eles ocorreram e a necessidade de decidir qual o acontecimento a processar primeiro, já que os acontecimentos não são simultâneos na realidade, sendo tratados como tal pelo modelo.

Estes problemas podem ser tornados menos severos tornando Δt o menor possível. Isto tem como contrapartida o aumento de verificações das ocorrências dos acontecimentos, que têm que ser efectuadas no fim de cada intervalo Δt, daí resultando um enorme aumento do tempo de execução da simulação. Devido a estas considerações, o método de avanço do tempo por incrementos fixos não é geralmente utilizado em modelos de simulação de acontecimentos discretos quando os tempos entre acontecimentos sucessivos podem variar significativamente, o que não é o caso da simulação gráfica de robôs.

Finalmente, é de notar que o método de avanço do tempo por incrementos fixos pode ser efectuado quando se utiliza a aproximação de avanço do tempo pelo método do próximo acontecimento, através dum escalonamento artificial de "acontecimentos" para ocorrerem a cada Δt unidades de tempo.

3 Arquitectura dos sistemas de simulação e programação *off-line* de robôs

A Figura 2.5 apresenta a arquitectura genérica de um sistema gráfico de simulação para efectuar a programação *off-line* de robôs, com as suas várias bases de dados de entrada e saída e módulos associados.

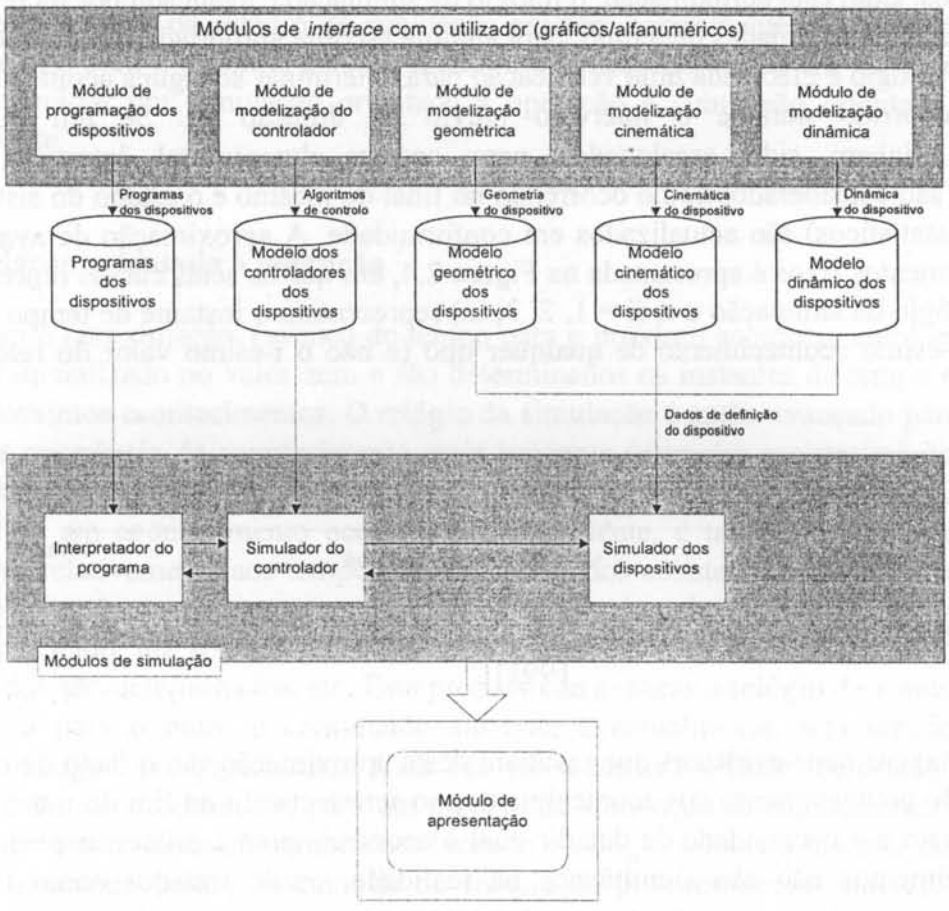


Figura 2.5: Estrutura básica de um sistema gráfico de simulação de robôs (Adaptado de [Bruhm, 1994])

Esta arquitectura identifica os componentes que podem existir num simulador destes, apresentando, no entanto, mais funcionalidades do que as que são estritamente necessárias. Cada um dos componentes identificados pode ser encontrado em vários sistemas comerciais de simulação de robôs, no entanto, em vários níveis diferentes de implementação [Nielsen,1994].

Vamos neste trabalho adoptar a notação de Trostmann (1994) e classificar os componentes de um sistema de simulação e programação *off-line* em modelos e módulos. Os modelos, tal como o seu nome indica, possuem a informação relativa ao sistema do robô, inclusivamente o programa do robô. Por sua vez, os módulos executam os necessários processamentos de dados. A figura anterior apresenta os modelos ("cilindros"), os módulos ("rectângulos"), os interfaces entre os módulos ("setas horizontais") e o fluxo de informação entre os modelos e os módulos ("setas verticais"). A seta vertical mais larga representa o fluxo de informação dos três módulos de simulação (interpretador do programa, simulador do controlador e simulador dos dispositivos) para o módulo de apresentação.

De seguida passam-se a descrever os modelos e os módulos individualmente. De notar que, numa implementação prática de um simulador, os módulos apresentados na figura não são necessariamente separados fisicamente. A implementação destes módulos pode variar entre subrotinas únicas (contidas num sistema de *software* maior), programas *stand-alone* e mesmo módulos de *hardware* (como é, por exemplo, o caso do módulo gráfico).

3.1 Modelos

Como pode ser visto na Figura 2.5, num simulador existem dois modelos diferentes: um modelo do programa do robô e um modelo do sistema do robô, este último constituído por vários sub-modelos.

3.1.1 Modelo do programa do robô

A informação contida no modelo do programa do robô é a descrição das acções que o robô deve realizar de forma a executar a tarefa pretendida.

As instruções de uma linguagem de programação de um robô podem ser divididas em duas categorias: uma parte trata da manipulação dos dados e, a outra parte, trata do controlo do robô. As instruções de manipulação de dados implementam a estrutura lógica do programa. As instruções relacionadas com o robô implementam o controlo do robô, incluindo-se nesta categoria instruções para o *feedback* da informação de sensores externos.

3.1.2 Modelo do sistema do robô

Como ilustrado na mesma figura (Figura 2.5), o modelo do sistema do robô pode ser dividido num modelo de controlo e num modelo do manipulador (dispositivo).

3.1.2.1 Modelo de controlo

O modelo de controlo contém uma descrição do controlador do robô e tem que ser capaz de descrever o controlo do movimento. Este modelo costuma integrar seis funções [Trostmann, 1994]: o planeamento da trajectória cartesiano; o interpolador cartesiano; a transformação da

cinemática inversa; o planeador de trajectórias junta-a-junta; o interpolador de juntas e o controlador de juntas.

O modelo de controlo também especifica uma interconexão específica destas seis funções, estabelecendo o fluxo de informação entre elas. A única liberdade quando se está a criar um modelo de controlo para um robô especifico, reside na definição dos algoritmos de controlo utilizados nas seis funções. Isto faz sentido, uma vez que as seis funções e a interconexão especificada são encontradas em quase todos os controladores de robôs industriais.

Este modelo fornece informação para o módulo de controlo da simulação.

3.1.2.2 Modelo do manipulador

O modelo do manipulador deve conter informação acerca da geometria, cinemática e dinâmica do manipulador robótico. Como resultado, este modelo deve incluir informação acerca da forma geométrica, da estrutura cinemática e das relações dinâmicas do manipulador.

A forma geométrica descreve as ligações individuais no manipulador. Esta informação geométrica do manipulador é essencial para efectuar testes de colisões e para visualização dos movimentos do robô, utilizando a animação gráfica tridimensional.

A estrutura cinemática descreve as relações entre as ligações e as juntas do manipulador e define os possíveis movimentos do manipulador, sendo o núcleo essencial para a simulação do movimento do manipulador.

Quanto às relações dinâmicas, estas podem ser descritas através de propriedades de massa (parâmetros concentrados definindo as distribuições da massa) das ligações do manipulador e da função de transferência dos *drives* dos motores que fazem mover o robô. A relação dinâmica é essencial para a simulação dos movimentos do manipulador sujeito à acção de forças/binários.

3.2 Módulos

3.2.1 Módulo de programação

A função do módulo de programação é transformar a descrição de uma dada tarefa do robô num programa do robô. Este módulo corresponde normalmente a um editor de texto, no qual é possível, ao utilizador, editar e compilar o programa do robô.

3.2.2 Módulo de interpretação do programa

O módulo de interpretação do programa pode ser visto como uma unidade funcional. A entrada é um programa do robô descrevendo as acções que este deve realizar de forma a alcançar a tarefa pretendida. Este módulo realiza a interpretação do programa do robô

baseando-se no modelo sintáctico e semântico da linguagem de programação utilizada. A interpretação "pega" numa instrução de cada vez e realiza as acções semânticas da expressão (por exemplo, modifica o valor de uma variável ou envia uma instrução de movimento para o simulador do controlador do robô). As acções vão depender do estado interno do interpretador (por exemplo, do valor das variáveis). A saída deste módulo é um fluxo de instruções de movimento passadas ao módulo de simulação do controlador do robô.

Uma vez que algumas tarefas do robô necessitam de informação respeitante ao estado actual do sistema do robô de forma a realizar as acções seguintes, quando necessário podem ser realimentados vários tipos de informações do módulo de simulação do controlador. Estas relações de entrada/saída e a interpretação do programa estão ilustradas na Figura 2.6.

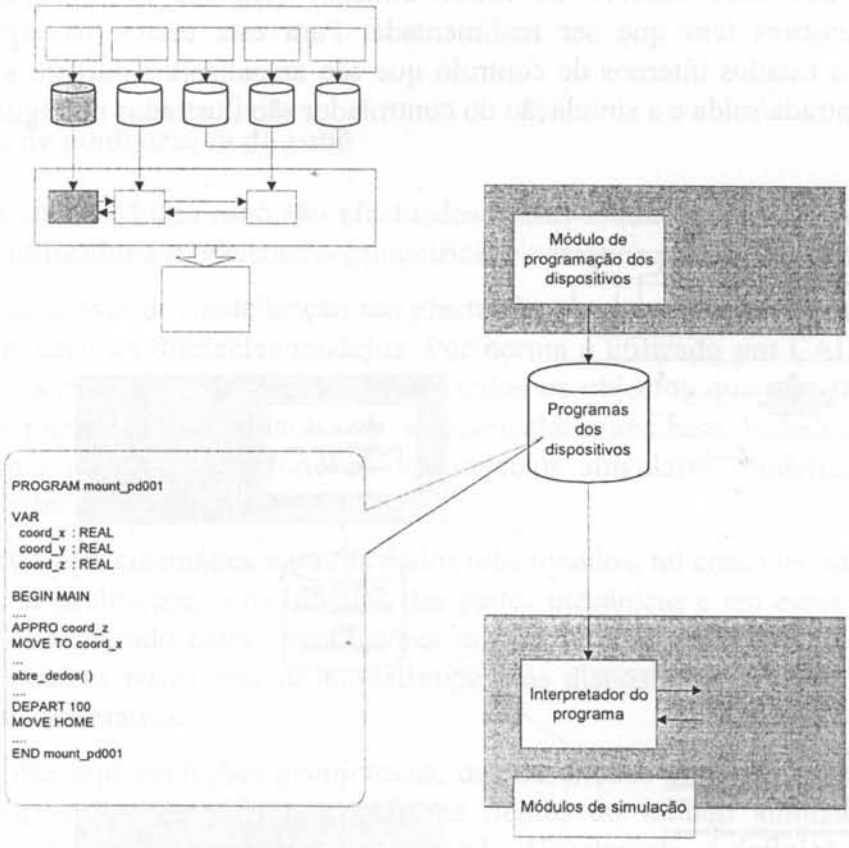


Figura 2.6: Interfaces do módulo de interpretação do programa (Adaptado de [Bruhm, 1994])

3.2.3 Módulo de modelização do controlador

O módulo de modelização do controlador, tal como o módulo de programação, é por norma um editor e compilador, mas neste caso de uma linguagem de programação de alto nível, por exemplo Pascal ou C, no qual são desenvolvidos os programas que implementam os algoritmos de controlo do robô. Nalguns simuladores ultimamente, têm sido também

utilizados produtos como o MATLAB e outros do género para implementar este módulo [Trostmann, 1993].

3.2.4 Módulo de simulação do controlador

O módulo de simulação do controlador calcula os sinais de controlo dos motores do robô, baseando-se no modelo de controlo. A entrada para o simulador do controlador é o fluxo de instruções de movimento do robô geradas pelo módulo de interpretação de programas. O modelo do controlador inclui algoritmos que podem efectuar as instruções de movimento e, em consonância, como saída gerar sinais de controlo dos *drives* dos motores.

Como a maioria dos controladores de robôs utilizam sensores no manipulador, esta informação dos sensores tem que ser realimentada. Para este efeito, os algoritmos do controlador incluem estados internos de controlo que são actualizados durante a simulação. Estas relações de entrada/saída e a simulação do controlador são ilustradas na Figura 2.7.

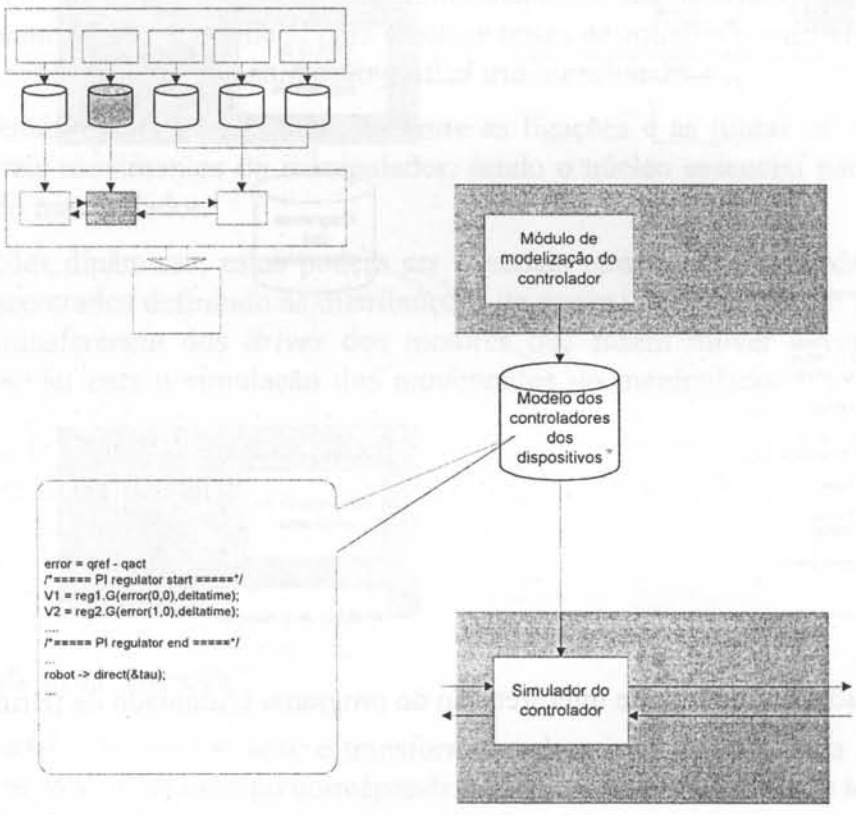


Figura 2.7: Interfaces do módulo de simulação do controlador (Adaptado de [Bruhm, 1994])

A precisão da simulação do controlador está fortemente dependente do modelo do controlador. Uma vez que a simulação do controlo deve reflectir o verdadeiro comportamento do controlador do robô a trabalhar no mundo real, os mesmos algoritmos utilizados para o controlo em tempo real do robô devem ser aplicados à simulação.

A definição de um tal modelo de controlo nem sempre é possível para um utilizador comum e as empresas fabricantes de robôs só muito excepcionalmente fornecem o modelo de controlo implementado no controlador dos seus robôs. Daí que seja extremamente complicado para um utilizador final conseguir determinar o modelo de controlo de um robô, que neste momento deve ser implementado pelo utilizador numa linguagem de programação de alto nível, presumivelmente utilizando o modelo estruturado genérico do controlador. Um exemplo de uma tentativa para ultrapassar este problema é o simulador ROPSIM (*Robot Off-line Programming and real-time SIMulation system*), que foi preparado para uma futura definição de um interface neutro para modelos de controlo [Trostmann, 1993]. Tal interface neutro permitiria aos utilizadores receberem um modelo de controlo exacto do vendedor do robô e aplicar este modelo para propósitos de simulação. Estes modelos de controlo também poderiam ser trocados com sistemas de *software* de outras origens ou diferentes funcionalidades.

3.2.5 Módulos de modelização do robô

Os passos da modelização do robô são efectuados nestes módulos. São eles os responsáveis por permitir ao utilizador a modelização geométrica, cinemática e dinâmica do robô.

Geralmente, estes passos de modelização são efectuados recorrendo a módulos especializados de *software* para criar os diferentes modelos. Por norma é utilizado um CAD para definir a estrutura e criar a representação geométrica de todos os objectos que vão ser utilizados na simulação. As representações resultantes são armazenadas numa base de dados de geometrias. Isto permite uma integração confortável dos objectos singulares modelizados dentro de múltiplos ambientes de simulação.

Para definir a estrutura cinemática e outros dados relacionados, tal como os limites das juntas, as velocidades, as acelerações e os binários das partes mecânicas e em especial do braço do robô, encontra-se integrado nestes simuladores um módulo de modelização cinemática. As descrições cinemáticas resultantes da modelização dos dispositivos são armazenadas numa base de dados de cinemática.

A combinação das representações geométricas, das descrições cinemáticas e dinâmicas e o arranjo e a disposição dos diferentes objectos dentro do mundo simulado é efectuada recorrendo a um sistema de modelização do mundo. Usualmente, a definição de uma célula robotizada é um procedimento iterativo, incluindo os módulos indicados anteriormente, de forma a alcançar um *layout* da célula óptimo. A saída deste sistema de modelização pode ser quer os modelos completos de células robotizadas, quer os modelos de partes de células, ambos armazenados numa base de dados de células.

3.2.6 Módulo de simulação do manipulador

Este é o módulo principal do simulador. O módulo de simulação do manipulador calcula a cinemática do manipulador tendo por base o modelo do manipulador. Uma vez que o modelo do manipulador pode descrever as propriedades de massa do sistema do braço do robô e as

propriedades dinâmicas dos *drives* dos seus motores, a simulação do manipulador pode, além do seu movimento, reflectir também o comportamento dinâmico deste.

A entrada para o módulo de simulação do robô são os sinais de controlo dos motores gerados pela simulação do controlador. Estes sinais de controlo vão afectar o manipulador através dos *drives* dos motores (em binário), e os estados internos do manipulador (posição, velocidade e aceleração das juntas do robô) vão ser actualizados. O resultado da simulação do manipulador é um modelo actualizado do mesmo. Estas relações de entrada/saída e a simulação do manipulador encontram-se ilustradas na Figura 2.8.

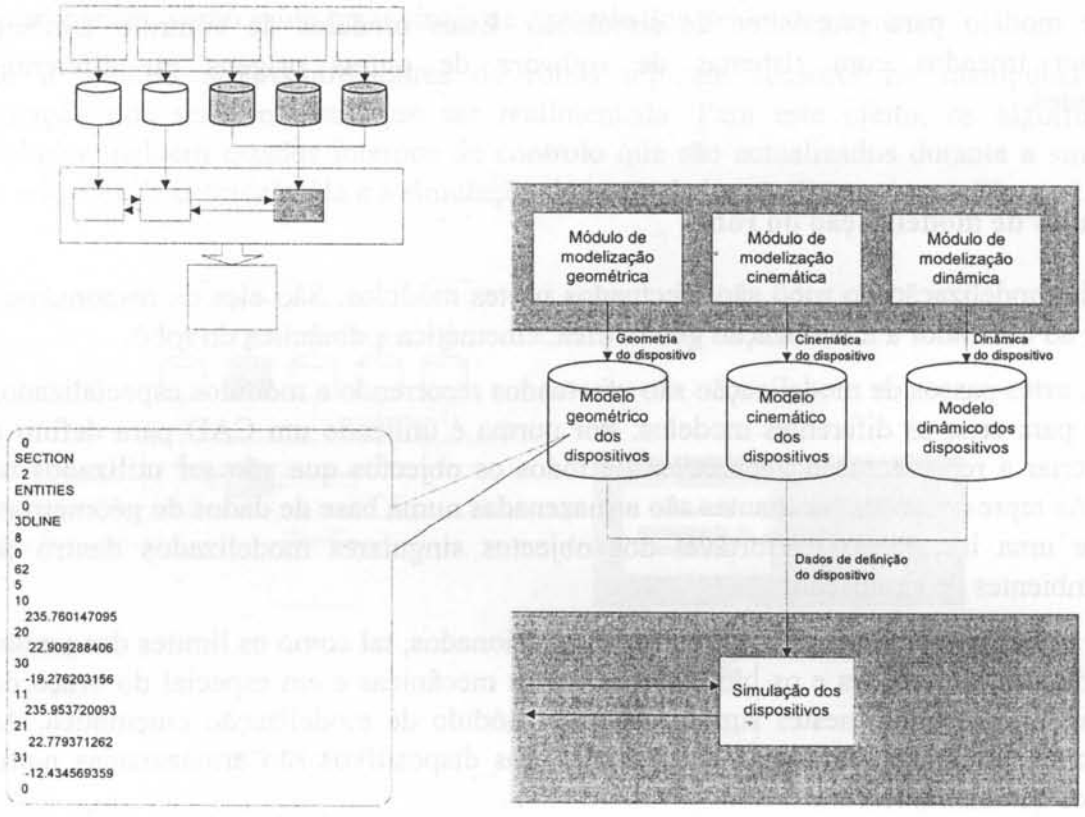


Figura 2.8: Interfaces do módulo de simulação do manipulador (Adaptado de [Bruhm, 1994])

3.2.7 Módulo de apresentação

O módulo de apresentação é o módulo onde os estados seleccionados do modelo do sistema (programa do robô e sistema do robô) são apresentados e é constituído, na esmagadora maioria dos casos, por monitores. Os estados do sistema global são gerados e actualizados pelos módulos de interpretação do programa, de simulação do controlador e de simulação do manipulador. A apresentação dos estados correntes pode tomar diferentes formas, dependendo da aplicação. Isto encontra-se ilustrado na Figura 2.9.

A visualização através da animação tem-se mostrado como sendo uma forma muito eficaz de ilustrar as relações geométricas entre objectos. A animação é muito útil para ilustrar os movimentos de um robô numa célula de trabalho, que poderá ainda incluir outros objectos em movimento.

Têm também sido aplicados diagramas para apresentação das relações entre variáveis de diversos domínios. Na simulação de robôs existem muitas relações que podem ter interesse e os gráficos e os diagramas são geralmente uma forma útil de apresentar tais relações. Os gráficos evolutivos podem ser utilizados para avaliar o estado de variáveis do sistema do robô ao longo do tempo.

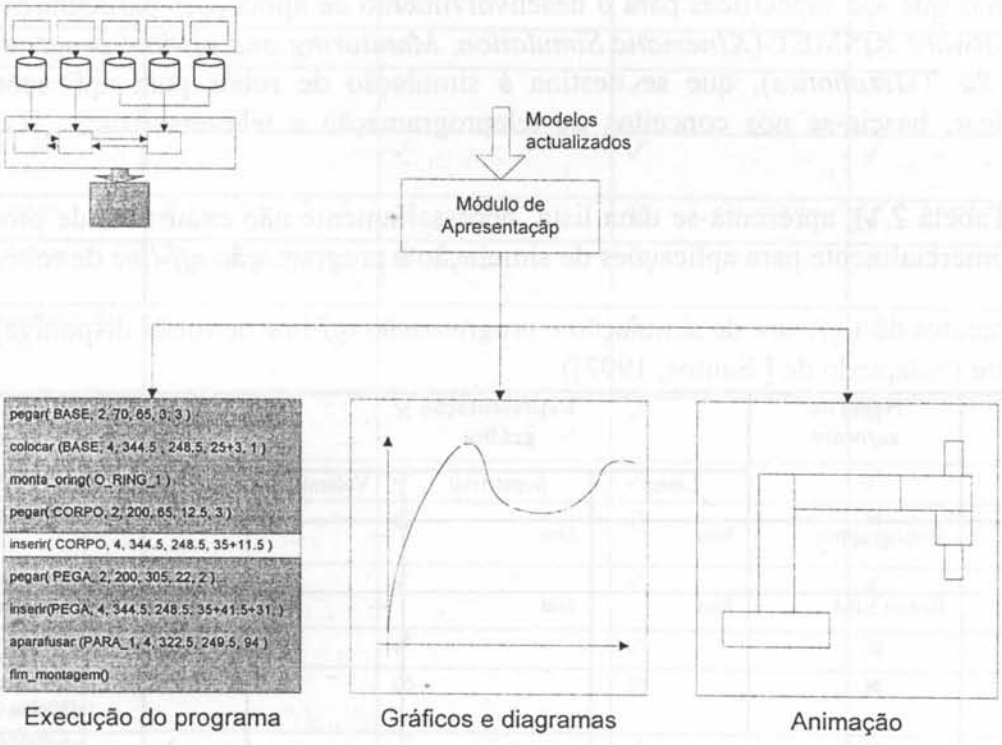


Figura 2.9: Interfaces do módulo de apresentação (Adaptado de [Bruhm, 1994])

Para indicar o estado de uma variável ou para indicar o estado do programa, aplica-se por vezes o texto da execução do programa. Um apontador para a instrução actual do programa é geralmente utilizado para mostrar o decurso da execução do programa.

4 Produtos disponíveis comercialmente

Está-se a assistir neste momento a uma explosão do número de produtos de *software* disponíveis comercialmente para efectuar a simulação e programação *off-line* de robôs.

O lançamento comercial destes produtos começou por ser efectuado por empresas que já dispunham de outros produtos de CAD tradicionais, tendo-se depois alargado a novas empresas que desenvolveram produtos de simulação de raiz. Além disso, surgem agora

algumas empresas, que desenvolvem e comercializam robôs, a lançar os seus próprios produtos no mercado; tais produtos são geralmente optimizados para os robôs e aplicações nas quais essas empresas se encontram a trabalhar. A título de exemplo, a firma Alemã KUKA Schweiβanlagen + Roboter GmbH lançou, no seguimento de um projecto Europeu em que participou, o projecto ESPRIT 623, um produto para simulação e programação *off-line* de células robotizadas chamado KUSIM.

Adicionalmente estão disponíveis comercialmente várias ferramentas que permitem efectuar a simulação de dispositivos programáveis, nomeadamente robôs e componentes de células robotizadas. Muitas outras ferramentas estão ainda em fase de desenvolvimento (em Escolas e Universidades, por exemplo as referidas por [Tenreiro, 1995] e [Vendrell, 1995]), assim como existem algumas que são especificas para o desenvolvimento de aplicações particulares - por exemplo, o *software* KISMET (*KInematic Simulation, Monitoring and off-line programming Environment for Telerobotics*), que se destina à simulação de robôs para aplicações na indústria nuclear, baseia-se nos conceitos de teleprogramação e telesensorização [Lorenz, 1994].

De seguida (Tabela 2.1), apresenta-se uma lista, necessariamente não exaustiva, de produtos disponíveis comercialmente para aplicações de simulação e programação *off-line* de robôs.

Tabela 2.1: Pacotes de *software* de simulação e programação *off-line* de robôs disponíveis comercialmente (Adaptado de [Santos, 1992])

Fornecedor	Nome do *software*	Representação gráfica			Introdução no mercado	Estado actual
		Linear	Superficial	Volúmica		
ComputerVision Corporation, Inc.	Robographix	Sim	Sim	---	*	*
General Electric / CALMA	Robot SIM	Sim	Sim	---	*	Vendeu esta linha de produtos à Computer-Vision Corporation, Inc.
Intergraph	Robotics	Sim	Sim	---	*	Vende os produtos da DENEB Robotics
SILMA	CimStation Robotics	Sim	Sim	Sim	*	*
SILMA	AdeptRapid	Sim	Sim	Sim	*	*
Mc Donnel Douglas	Robotics	Sim	---	---	*	*
Tecnomatix	Robcad	Sim	Sim	Sim	1983	*
BYG	Grasp	Sim	Sim	Sim	1986	*
Robot Simulations	Workspace	Sim	Sim	Sim	1989	Versão 3.4
Deneb Robotics	Igrip	Sim	Sim	Sim	1985	Versão 2.4
Francisco Fernandes Software	Robot3D	Sim	---	---	1991	Versão 3.0

* - Dados não disponíveis

Na tabela seguinte (Tabela 2.2), por sua vez, apresenta-se uma comparação das características de alguns destes produtos.

Tabela 2.2: Comparação das principais características de alguns pacotes de *software* de simulação e programação *off-line* de robôs

	Workspace	Robcad	Igrip	Grasp
Editor de CAD 3D	✓	✓	✓	✓
Tradutores *off-line* para as linguagens de programação de robôs	✓	∅	∅	∅
Comunicação entre tarefas	✓	✓	✓	✓
Cálculo do tempo de ciclo	✓	✓	✓	✓
Detecção de colisões	✓	✓	✓	✓
Análise do alcance dos robôs	✓	✓	✓	✓
Editor integrado de programas do robô	✓	✓	✓	✓
Calibração de robôs	✓	∅	∅	✓
Calibração de células robotizadas	✓	∅	∅	✓
Filtros para importação de ficheiros DXF do AutoCAD	✓	∅	∅	∅
Constructive Solid Geometry	✓	✗	✓	✗
Dimensionamento	✓	✓	✓	✓
Análise dinâmica do movimento	✓	∅	∅	✗
Cinemática definida pelo utilizador	✓	✓	✓	✓
Eixos auxiliares dos robôs	✓	✓	✓	✗
Linkagem de programas definidos pelo utilizador	✓	∅	∅	✗
Gerador automático da solução cinemática inversa	✓	✓	✓	✗
Execução em PC	✓	✗	✓	✗
Os programas admitem rotinas e procedimentos?	✓	✓	✓	✓
Suporta STEP?	✗	∅	∅	✓
Suporta IRL?	✗	✗	✗	✓
Suporta ICR?	✗	✗	✗	✓
Aplicações dedicadas de que dispõe (pacotes verticais) e se possui algum para montagem	✗	Soldadura por pontos; Soldadura por arco eléctrico; Pintura; Teleoperação.	Soldadura por pontos; Soldadura por arco eléctrico; Pintura; Acabamento.	Soldadura por arco eléctrico; Paletização; Pintura.
✓ disponível ∅ disponível como extra ✗ não disponível				

Apesar de não pretendermos que esta tabela sirva de base para a escolha de um produto deste tipo, tarefa sempre complicada e para a qual se têm desenvolvido algumas metodologias [Hlupic, 1996], apresentamos a título comparativo algumas características dos produtos de *software* deste tipo mais divulgados no mercado.

Como se pode ver na tabela apresentada, dentro da plêiade de produtos para simulação e programação *off-line* de robôs existem alguns que já possuem módulos verticais preparados para aplicações especiais, tais como pintura, soldadura por arco, soldadura por pontos e polimento. Estas aplicações verticais contam já com algoritmos próprios para as aplicações em causa que, por exemplo no caso dos pacotes vocacionados para as aplicações de pintura, permitem que a camada de tinta fique uniforme ao minimizar as passagens por cada local da superfície a pintar.

5 Aplicações da simulação à programação *off-line* de robôs e seus benefícios

Tal como referido no Capítulo 1, a integração óptima de robôs industriais em sistemas de fabrico complexos requer técnicos altamente qualificados, ferramentas de planeamento assistido por computador, quer para o planeamento do sistema de fabrico, quer para a programação do próprio robô, sendo um factor decisivo para a operação eficiente do sistema.

Este trabalho de Engenharia pode e deve ser suportado por ferramentas auxiliadas por computador. A simulação conjugada com a programação *off-line* alarga a área de aplicação dos robôs e abre novas possibilidades em domínios como o corte por laser e outras operações perigosas.

A título de sistematização, através dos sistemas de planeamento e simulação deve-se procurar dar resposta aos seguintes problemas:

- qual é a selecção óptima dos componentes?
- qual é o arranjo óptimo dos componentes dentro da célula?
- podem todas as posições-alvo do robô ser alcançadas por este?
- o manuseamento do robô faz a peça colidir com outros componentes da célula?

Utilizando este tipo de ferramentas de simulação para efectuar a programação *off-line* pode-se ainda efectuar o planeamento detalhado das tarefas a serem executadas pelo robô antes da célula real ser construída e implementada na planta fabril. Isto inclui o planeamento da trajectória geométrica do robô, a determinação dos parâmetros tecnológicos e de movimento, o controlo das interacções com os periféricos e a geração do código dos programas para o robô (programas das aplicações).

Segundo Bernhardt (1992), as categorias identificadas, onde o uso das técnicas de programação *off-line* são mais vantajosas para a indústria, são as seguintes:

- tarefas com especificações de tempo críticas;
- operações de robôs complexas, com actuadores finais multi-propósito;
- estruturas cinemáticas complexas com mais de seis juntas envolvidas;

- programação de peças regulares (por exemplo, simétricas) através da aplicação de funções de geração ou manipulação de programas (por exemplo, funções de espelho);
- aplicações de elevada precisão (por exemplo, corte por laser), requerendo programas com optimização dos valores das juntas;
- transferência e adaptação de programas existentes para aplicações similares (por exemplo, adaptação de uma linha de transferência para um novo modelo de automóvel).

Entre os benefícios reportados por Bernhardt (1992) obtidos com a simulação e programação *off-line* de robôs, incluem-se os seguintes:

- reduções de custos significativas, porque a reprogramação elimina o tempo de paragem do robô, além de que são reduzidos, ou mesmo eliminados, os custos de pessoal para projecto, optimização e programação; adicionalmente, são também reduzidos os custos para testes com células reais e alterações subsequentes do sistema;
- ciclos de produção mais rápidos, porque passa a estar disponível informação muito precisa, por exemplo em relação à configuração do *layout,* numa fase muito inicial do projecto;
- poupança de tempo, em alguns casos atingindo poupanças de tempo até 85% (ver Figura 2.10), pois as soluções do utilizador são desenvolvidas mais rapidamente, também dentro da fase de preparação;
- e, finalmente, a engenharia óptima dos produtos com qualidade melhorada, uma vez que podem ser analisadas e comparadas diversas soluções alternativas, daí resultando uma melhoria decisiva da qualidade da solução.

A título de exemplo, no gráfico seguinte (Figura 2.10) apresentam-se as poupanças de tempo declaradas pela KUKA, usando o pacote KUSIM de simulação e programação *off-line* de robôs, em aplicações de vários tipos.

6 A necessária evolução dos simuladores de robôs

Apesar de já estarem disponíveis há alguns anos, estes sistemas gráficos de simulação e programação *off-line* não têm tido o sucesso esperado no mercado. Conhecendo as vantagens óbvias deste tipo de programação *off-line* é natural perguntarmos o motivo pelo qual poucas empresas a usam.

Um estudo realizado pelo PSI (*Gesellschaft für Prozeβsteuerungsund Informationssysteme mbH*) ao mercado Alemão, na segunda metade de 1988, e apresentado por [Bernhardt,1992], resultou nas seguintes conclusões:

1. não existe uma linguagem de programação comummente aceite para os robôs industriais;
2. utilizadores com diferentes robôs têm diferentes linguagens de programação;
3. o mundo do CAD difere do mundo real;
4. não há formatos de dados normalizados para troca de dados geométricos e cinemáticos;

5. o desenvolvimento de um sistema de programação *off-line* que seja suficientemente poderoso para ser de utilidade prática é demasiado caro para um único controlador de robô.

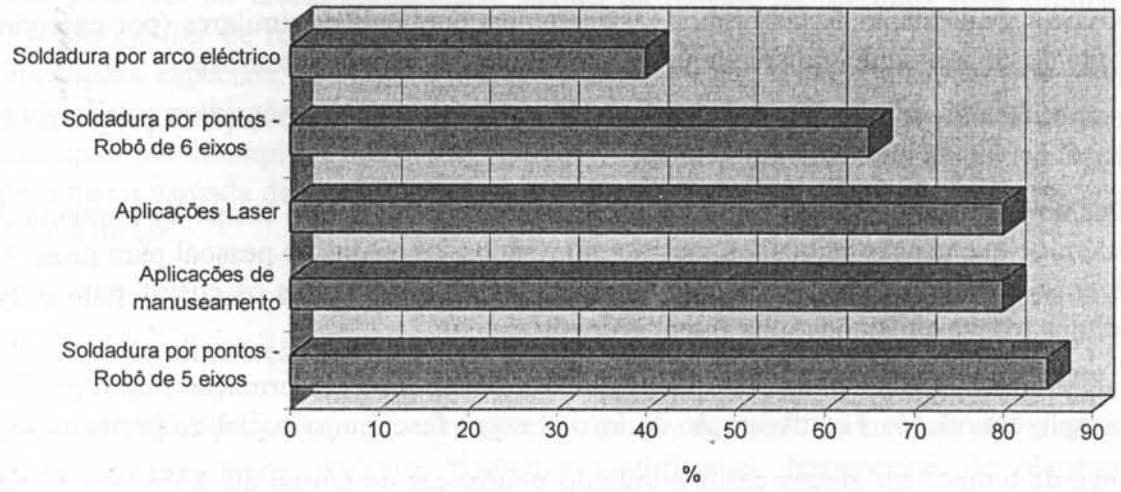


Figura 2.10: Poupanças de tempo conseguidas pela KUKA através da utilização da simulação e programação *off-line* (Adaptado de [Bernhardt, 1992])

Todos os pontos referidos neste estudo continuam a apresentar bastante actualidade e são estes alguns dos motivos pelos quais não se verifica uma introdução mais generalizada da programação *off-line*. A isto há que acrescentar o preço de alguns destes produtos, acrescidos das despesas de manutenção de mais este equipamento.

6.1 Normalização de formatos de troca de dados

Os fornecedores de sistemas de tecnologias de informação e de sistemas de comunicação podem ter um interesse limitado nos interfaces normalizados, o que pode ser deduzido da sua penetração no mercado; os utilizadores de tais sistemas, pelo contrário, apresentam a tendência para promoverem fortemente esta aproximação de forma a evitarem dependências de uma única fonte de equipamento e *software* [Bey, 1994].

Para empresas em crescimento e empresas a trabalhar em áreas com necessidade de rápidas mudanças, a possibilidade de trocar rapidamente os elementos dos seus sistemas é vital. Isto fornece o benefício dos ambientes integrados de projecto e produção, sem limitar a capacidade de responder a alterações.

Deve portanto ser possível fazer o interface entre o sistema de programação *off-line* com sistemas de CAD existentes, assim como com os robôs sem a necessidade de duas linguagens de programação diferentes. A progressiva introdução de interfaces neutros, espera-se, vai reduzir o preço das ferramentas de programação *off-line* e melhorar as suas capacidades. Isto

vai tornar a programação *off-line* numa técnica comum que também poderá ser usada em PMEs (Pequenas e Médias Empresas).

Adicionalmente, e como já referido, tem-se feito um grande esforço de normalização das linguagens de programação, julgando-se que este trabalho será continuado com vista ao desenvolvimento de uma norma que permita a troca transparente de programas entre os simuladores e os controladores de robôs, independentemente do fabricante de cada um destes equipamentos.

6.2 Calibração de robôs e de células robotizadas

Também ao nível da calibração de robôs se tem assistido a um esforço na tentativa de desenvolvimento de novas metodologias, mais precisas, fiáveis e fáceis de implementar, de forma a permitir uma rápida e eficaz calibração das células robotizadas simuladas de forma a corresponderem o mais aproximadamente possível à realidade existente na planta fabril

Estes problemas referidos aqui de uma forma sucinta serão aprofundados no capítulo seguinte, pois vão surgindo à medida que se for referindo os passos necessários para desenvolver uma aplicação robotizada usando um simulador de robôs.

Capítulo 3

Desenvolvimento *off-line* de uma aplicação robotizada

Neste capítulo procura-se dar resposta às seguintes questões:

- Quais os passos para o desenvolvimento *off-line* de uma aplicação robotizada;
- Que ferramentas existem para auxiliar a concretizar cada um desses passos.

Sintetizando o que foi dito no capítulo anterior, podemos dizer que os sistemas de simulação e programação *off-line* de células robotizadas são ferramentas indicadas para as seguintes tarefas de planeamento *off-line*:

- selecção de robôs e actuadores finais para uma dada tarefa;
- planeamento do *layout* de células de trabalho robotizadas;
- geração de programas para robôs;
- simulação gráfica da execução dos programas;
- detecção visual de colisões;
- optimização do *layout* da célula e dos programas dos dispositivos;
- transferência do código para os robôs;
- monitorização do comportamento dos equipamentos na planta fabril.

Neste capítulo vamos ver quais são os passos necessários para se desenvolver uma simulação de uma célula de trabalho robotizada com vista ao desenvolvimento *off-line* de programas para os robôs dessa célula. Simultaneamente, veremos como é que estas funcionalidades e vantagens, oferecidas pela utilização destes simuladores, podem ser alcançadas durante a realização deste trabalho.

A metodologia para o desenvolvimento de programas *off-line* utilizando sistemas de simulação e animação gráfica é, genericamente, a seguinte:

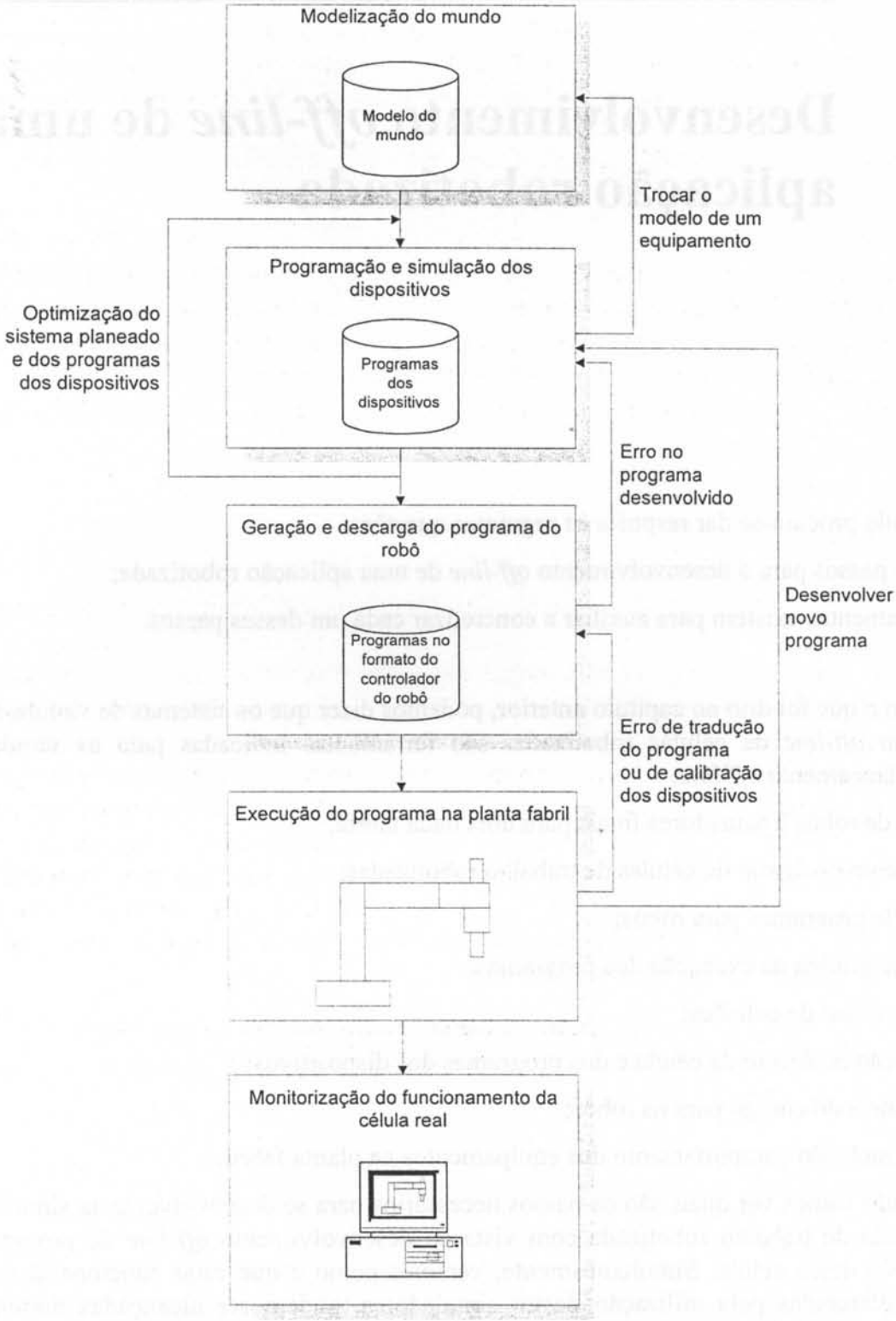


Figura 3.1: Metodologia para o desenvolvimento de simulações

1 Modelização do mundo

Como está representado na Figura 3.1, o desenvolvimento de uma aplicação inicia-se geralmente com o projecto da célula (modelo do mundo). Esta fase não só inclui a escolha dos componentes mais adequados a cada função particular (robôs, actuadores finais, fixações, *paletes*, etc.), como também o estudo do *layout* (ou seja, a disposição dos vários equipamentos na célula de forma a optimizar um ou mais parâmetros, por exemplo o tempo de ciclo de uma dada operação). Podem-se ainda realizar estudos geométricos, cinemáticos ou dinâmicos que conduzam ao desenvolvimento de um novo dispositivo ou componente para inclusão na célula em estudo.

Para modelizar o ambiente da célula de uma forma independente da tarefa é necessária a representação da sua funcionalidade global. Esta representação funcional tem que incluir as respectivas propriedades estáticas e dinâmicas da célula: as propriedades estáticas reflectem a descrição geométrica dos componentes, o *layout* e as relações entre esses mesmos componentes, enquanto que as propriedades dinâmicas representam a sincronização e a concorrência dos processos. Na descrição da sincronização têm que ser também definidas as condições da monitorização.

1.1 Modelização dos componentes

Se considerarmos uma célula robotizada típica, podemos distinguir diferentes tipos de componentes dentro do sistema de simulação de robôs. Geralmente, o principal componente é o próprio braço do robô. Adicionalmente, existem as garras e as ferramentas, tais como chaves aparafusadoras e tochas de soldadura, que são utilizadas pelo robô. Por oposição a estes componentes, que pertencem directamente ao robô, existem os componentes que constituem a envolvente do robô. Estes componentes podem ser divididos em duas classes principais - componentes dinâmicos e estáticos da envolvente [Laloni, 1995]. A classe dos componentes dinâmicos da envolvente é composto por todos os objectos que são capazes de realizar acções autónomas ou que podem ser activados por forças aplicadas externamente. Tipicamente, pertencem a esta classe os alimentadores, os tapetes rolantes, etc. Os restantes objectos da envolvente, tais como paredes, mesas, controladores, fixações, *paletes*, etc., são os constituintes da classe dos componentes estáticos.

Cada um destes componentes tem que ser modelizado. De acordo com Laloni (1995) e Bernhardt (1992), a modelização consiste nos seguintes pontos básicos:

- modelização geométrica;
- modelização cinemática;
- modelização dinâmica;
- modelização funcional.

Dependendo do tipo do objecto, têm que ser realizadas diferentes etapas de modelização que poderão ser todas ou só algumas das acima referidas.

1.1.1 Modelização geométrica

A primeira etapa da modelização é a definição do modelo geométrico. Este modelo é necessário para a visualização da célula de trabalho durante a sua simulação.

Para construir esta representação gráfica, a maioria dos pacotes de *software* deste tipo vêem equipados com uma aplicação de CAD que pode ser mais ou menos elaborada. Adicionalmente, alguns destes sistemas permitem também a importação de modelos construídos em sistemas de CAD de uso genérico.

A forma de um objecto ou o seu modelo geométrico podem ser representados recorrendo a várias técnicas [Ramos, 1993]. As formas mais frequentemente usadas nos produtos de simulação são a representação em modelo de arame (*wireframe*), a representação por superfícies ou fronteiras (*B-rep*) e a representação por modelos de sólidos ou geometria sólida construtiva (CSG - *Constructive Solid Geometry*).

A definição dos modelos ou representações dos objectos pode ser realizada quer através de uma especificação textual dos parâmetros dos objectos, por exemplo especificando as coordenadas dos vértices, a ligação destes com as arestas, etc., ou interactivamente recorrendo a um editor gráfico. Devido à facilidade de modelização e ao *feedback* visual directo, os editores gráficos são geralmente os preferidos. Dependendo da representação seleccionada para os objectos, as técnicas de modelização utilizadas em tais editores gráficos são diferentes: geralmente distinguem-se sistemas de modelização orientados às superfícies ou aos volumes, a que correspondem, respectivamente, as técnicas de representação de fronteira ou de sólidos.

1.1.1.1 Representação em modelo de arame

Na representação em estrutura de arame um objecto é representado unicamente pelos seus vértices e pelas arestas ligando os vértices (Figura 3.2).

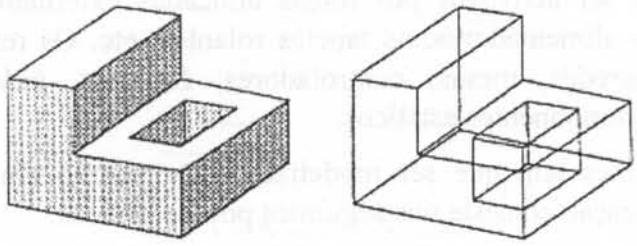


Figura 3.2: Representação de um objecto em estrutura de arame

A principal vantagem da representação em estrutura de arame resulta do facto de a sua criação e pós-processamento, tal como a computação de uma rotação ou translacção, ser efectuada muito rapidamente sem ser necessário recorrer a *hardware* gráfico especializado, devido à forma de representação ser simples.

Infelizmente, esta representação apresenta várias desvantagens. Devido à falta de informação relativa às superfícies dos objectos representados, os modelos em estrutura de arame oferecem

apenas uma pobre capacidade de visualização e uma interpretação não única; uma caixa aberta, por exemplo, não pode ser distinguida de uma fechada, uma vez que ambos os objectos são representados pelo mesmo modelo em estrutura de arame. Outra das dificuldades apontadas aos modelos em estrutura de arame é a dificuldade na representação de silhuetas [Ramos, 1993].

Este tipo de representação é geralmente o único disponível em produtos da gama baixa, sendo também muitas vezes o utilizado nas animações por permitir uma actualização dos gráficos de uma forma mais rápida.

1.1.1.2 Representação por superfícies ou fronteiras

A representação por superfícies ou fronteiras (*B-rep*) de um objecto consiste num conjunto de curvas planares ou equações de planos formando uma rede de polígonos da forma do objecto (Figura 3.3).

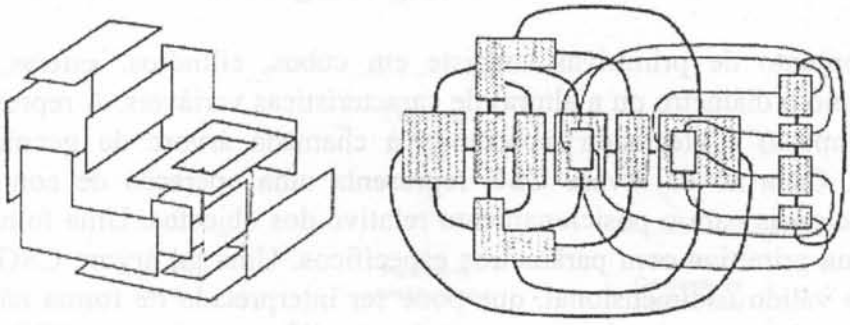


Figura 3.3: Representação de um objecto por superfícies

Se adicionalmente forem permitidas representações de curvas, utilizando aproximações polinomiais (*splines*) para definir as fronteiras de um objecto, podem-se mesmo representar objectos com uma forma bastante complexa. Dependendo da complexidade da rede de polígonos resultante, tem que ser tomado em consideração que a complexidade de processamento dos algoritmos envolvidos aumenta drasticamente. Em contraste com a representação em estrutura de arame mencionada acima, a representação explícita das fronteiras dos objectos leva a boas possibilidades de visualização; podem ser aplicados algoritmos de linhas ou superfícies invisíveis e sombreamento de superfícies para melhorar a qualidade da saída gráfica. No entanto, a informação acerca do lado interior e lado exterior do objecto, que é muito importante por exemplo nos algoritmos de detecção de colisões, não pode ser determinada directamente a partir do modelo do objecto, mas tem que ser computada explicitamente, o que constitui uma desvantagem deste método de representação.

1.1.1.3 Representação por modelos de sólidos

Na representação por modelos de sólidos, o objecto modelizado é representado através de sólidos primitivos ou sólidos previamente modelizados, que são combinados utilizando operações de conjuntos, tais como a união ou a intersecção, para formarem sólidos mais complexos. Na Figura 3.4 apresenta-se a forma como poderia ser construída a representação por modelos de sólidos do objecto apresentado na Figura 3.2.

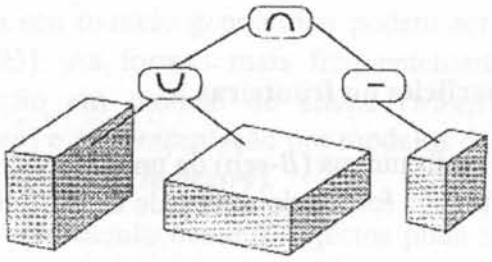


Figura 3.4: Representação de um objecto por modelos de sólidos

Tipicamente, o conjunto de primitivas consiste em cubos, cilindros, esferas, etc., com parâmetros (tais como o diâmetro ou a altura) de características variáveis. A representação do objecto sólido composto é efectuada recorrendo à chamada árvore da geometria sólida construtiva (CSG). Cada nó da árvore CSG representa uma operação de conjuntos, com transformações adicionais para o posicionamento relativo dos objectos. Uma folha da árvore CSG representa uma primitiva com parâmetros específicos. Uma tal árvore CSG representa sempre um objecto válido tridimensional, que pode ser interpretado de forma não ambígua, sem prejuízo de um objecto poder ser representado por diferentes árvores CSG. Devido ao facto dos objectos sólidos serem modelizados, é fácil distinguir entre o interior e o exterior dos objectos, o que é uma grande vantagem no que concerne à implementação dos algoritmos de detecção de colisões [Purgathofer, 1995].

Como desvantagem deste tipo de representação, há que referir o facto de a árvore CSG ter que ser avaliada de cada vez que o objecto é processado, o que leva a uma complexidade computacional muito elevada; portanto, é geralmente necessário *hardware* especializado para tratar objectos modelizados com sólidos, tais como placas aceleradoras gráficas e *Z-buffers*, o que encarece extremamente o *hardware* necessário para correr simuladores que utilizem este tipo de representação.

1.1.2 Modelização cinemática

Adicionalmente ao modelo geométrico é necessário uma representação cinemática para todos os dispositivos mecânicos capazes de realizar algum tipo de movimento. Esta descrição cinemática é necessária, independentemente do facto de o movimento ser causado por actuadores pertencentes ao dispositivo (como é o caso do braço do robô) ou do movimento ser provocado por forças aplicadas ao dispositivo a partir do exterior.

A cinemática não trata as massas, forças e/ou momentos que provocam o movimento. A cinemática trata apenas do estudo e representação da geometria do movimento de um dispositivo mecânico com uma ou mais juntas e ligações. Portanto, a cinemática do braço do robô só diz respeito aos parâmetros das juntas que definem a estrutura geométrica das ligações e os ângulos das juntas e velocidades descrevendo o deslocamento espacial do actuador final do robô em relação ao tempo. Isto é efectuado em relação a um sistema fixo de referência de coordenadas, que é geralmente denominado por sistema de coordenadas da base ou do mundo.

No campo da cinemática do braço do robô, as duas questões seguintes têm um papel vital:

- qual é a posição e orientação do actuador final relativamente ao sistema de coordenadas da base, para um dado manipulador e dados os valores dos ângulos das juntas?
- pode o braço do robô alcançar uma dada posição cartesiana com uma dada orientação e quais os valores dos ângulos das juntas que têm que ser utilizados? Existem diferentes configurações das juntas que satisfaçam estas condições?

A primeira questão é conhecida como o problema da cinemática directa, ao passo que a segunda é denominada por problema da cinemática inversa. A relação entre a cinemática directa e a inversa é apresentada na figura seguinte.

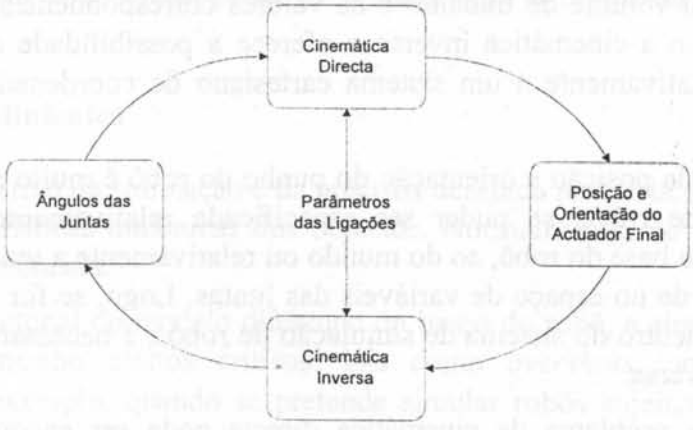


Figura 3.5: Relação entre a cinemática directa e a cinemática inversa [Laloni, 1995]

A cinemática directa calcula, para uns dados valores dos ângulos das juntas e parâmetros das ligações, a posição e orientação do actuador final, enquanto que a cinemática inversa fornece as variáveis das juntas para um dado conjunto de parâmetros das ligações e para uma posição do actuador final [Koivo, 1989].

A técnica geralmente utilizada para determinar e descrever a cinemática de um robô baseia-se na notação de Denavit-Hartenberg [Laloni, 1995].

1.1.2.1 Cinemática Directa

Se considerarmos um braço do robô, podemos afirmar que muitas vezes um tal manipulador mecânico pode ser modelizado como uma cadeia em malha aberta, consistindo em vários

corpos rígidos (ligações) que são ligados por juntas. Estas, quer sejam prismáticas ou de revolução, são conduzidas por actuadores e cada junta constitui um grau de liberdade mecânica. Um dos extremos desta cadeia em malha aberta encontra-se fixo à base do robô, enquanto que o outro se encontra geralmente livre e ligado ao actuador final.

Para descrever esta estrutura, Denavit e Hartenberg (1955) propuseram uma aproximação sistemática e generalizada utilizando matrizes de transformação homogéneas; esta aproximação levou à notação de Denavit-Hartenberg.

Cada matriz descreve as relações espaciais ou as transformações entre duas ligações adjacentes. A aplicação sucessiva destas transformações a todas as ligações do braço do robô leva a uma matriz de transformação homogénea equivalente, que descreve o posicionamento espacial do actuador final relativamente ao sistema de coordenadas da base do braço do robô. Portanto, esta matriz de transformação constitui a solução do problema da cinemática directa.

1.1.2.2 Cinemática inversa

Adicionalmente à cinemática directa do braço do robô, é muito útil durante o desenvolvimento *off-line* do programa do robô possuir a relação entre as coordenadas cartesianas dentro do seu volume de trabalho e os valores correspondentes das juntas. Esta relação é conhecida como a cinemática inversa e oferece a possibilidade de especificar os movimentos do robô relativamente a um sistema cartesiano de coordenadas definido pelo utilizador.

Usualmente, a definição da posição e orientação do punho do robô é muito mais conveniente para um programador de robôs se puder ser especificada relativamente ao sistema de coordenadas cartesiano da base do robô, ao do mundo ou relativamente a um sistema definido pelo utilizador, ao invés de no espaço de variáveis das juntas. Logo, se for desejada uma tal programação cartesiana dentro do sistema de simulação de robôs, é necessário implementar a solução da cinemática inversa.

A solução simbólica do problema da cinemática directa pode ser encontrada facilmente usando a notação de Denavit-Hartenberg. Pelo contrário, a computação da solução simbólica do problema da cinemática inversa é mais complicada e, por vezes, mesmo impossível. Este é, como vimos, o segundo problema no campo da cinemática do braço do robô.

Para a sua resolução têm sido desenvolvidas algumas aproximações numéricas. Em contraste com as aproximações analíticas, as técnicas numéricas são sempre capazes de computar iterativamente as variáveis das juntas correspondendo a uma dada posição e orientação do sistema de coordenadas do punho do robô, para braços de robôs com estruturas cinemáticas arbitrárias. Mas como as aproximações numéricas geralmente requerem longos tempos computacionais e uma vez que têm de se confrontar com problemas de convergência na vizinhança de singularidades ou no caso de cinemáticas degeneradas de braços, é desejável utilizar a solução analítica, se esta puder ser determinada.

Muitas vezes a resolução do problema da cinemática inversa leva a mais do que uma solução. Na Figura 3.6 apresentam-se as duas soluções da cinemática inversa de um robô SCARA para uma mesma localização no espaço cartesiano.

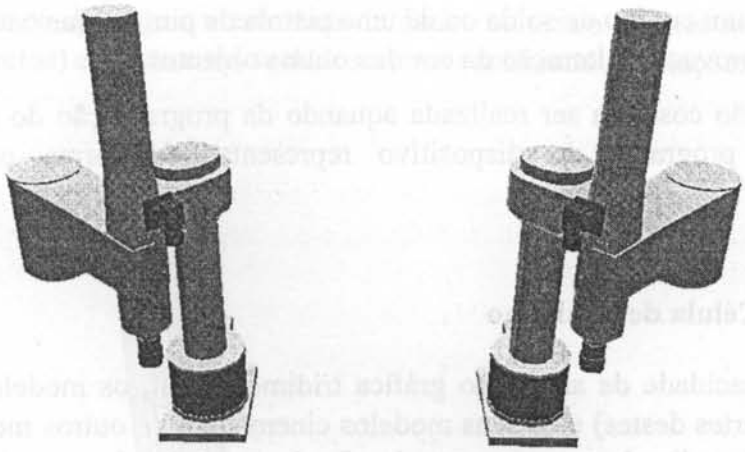


Figura 3.6: Soluções do problema da cinemática inversa para uma mesma localização

As múltiplas soluções do problema da cinemática inversa são muito importantes durante a programação do robô no mundo real, bem como no simulado: vamos por exemplo assumir que uma posição desejada do actuador final não pode ser alcançada devido a uma colisão do braço do robô com os objectos da sua envolvente. Nestes casos, talvez outra configuração do braço do robô, isenta de colisões, possa ser encontrada entre as múltiplas soluções.

1.1.3 Modelização dinâmica

Dependendo do objectivo da simulação e da precisão desejada para esta, poderá ser necessário modelizar as características dinâmicas dos objectos. Normalmente, no entanto, este tipo de modelização não é efectuada.

Com a utilização adicional do modelo dinâmico do braço do robô, a simulação do robô pode ser melhorada, e mesmo efeitos críticos, tais como *overshots*, podem ser simulados correctamente. Por exemplo, quando se pretende simular robôs sujeitos a grandes cargas e solicitações dinâmicas ou quando é necessário simular um problema de controlo, esta modelização tem que ser efectuada, pois só desta forma os resultados da simulação vão apresentar resultados realistas.

O modelo dinâmico consiste num conjunto de equações dinâmicas que podem ser determinadas a partir das leis físicas conhecidas e que estabelecem a relação entre forças/binários e posições, velocidades e acelerações das juntas [Koivo, 1989].

1.1.4 Modelização das funções dos dispositivos

Por fim, existe a necessidade de modelizar as funções específicas dos dispositivos em questão. Por exemplo, se um sensor de ultrasons, ou mesmo um simples interruptor, for incluído na simulação, as funções destes dispositivos, tais como os sinais de saída, têm de ser modelizados. Além do mais, o próprio efeito que é causado pela utilização de um objecto da célula de trabalho tem de ser modelizado: por exemplo, será o caso de um equipamento de

soldadura que provoca um cordão de solda ou de uma pistola de pintura que causa um feixe de tinta no ar e, por isso, provoca a alteração da cor dos outros objectos.

Porém, esta modelização costuma ser realizada aquando da programação do dispositivo em questão, devendo o programa do dispositivo representar de forma correcta a sua funcionalidade.

1.2 Modelização da Célula de Trabalho

Para alcançar uma capacidade de animação gráfica tridimensional, os modelos geométricos dos dispositivos (ou partes destes) e os seus modelos cinemáticos (e outros modelos), têm de ser combinados; isto é realizado nos sistemas de simulação de robôs, usando as chamadas estruturas de dados de ligação (*affixment data-structures*).

Adicionalmente, os componentes da célula de trabalho do robô têm de ser arranjados dentro do mundo da simulação e as suas relações, tais como "o componente A encontra-se sobre a mesa" ou " a parte A está ligada (*attached*) ao objecto B" têm que ser modelizadas. Com esse propósito, de forma semelhante à especificação da cinemática do braço do robô, são usadas matrizes de transformação homogéneas dentro das estruturas de dados de ligação.

Uma ligação (*affixment*) define [Laloni, 1995] :

- uma ligação entre dois objectos. Portanto, um objecto A que se encontre ligado a um objecto B tem que se mover se a posição de B mudar;
- a posição e orientação relativa entre os sistemas de coordenadas de referência (referenciais) de dois objectos, através de matrizes de transformação homogéneas. Portanto, a posição relativa de duas ligações subsequentes de um robô podem, por exemplo, ser expressas por uma ligação com uma matriz de transformação variável (a matriz A_i correspondente).

Na Figura 3.7 podem-se ver os sistemas de coordenadas utilizados para definir as ligações entre os componentes de um robô SCARA.

Seguidamente, apresentam-se as principais estruturas de dados utilizadas dentro do conceito de ligação, tal como definidas por Laloni (1995):

```
struct OBJECT{ nome       nome;          /*nome do objecto*/
               flag       actualizado;   /*para processamento*/
               frame      objectframe;   /*posição absoluta do objecto*/
               affix      *affixment;    /*apontador para lista de ligação*/
               geometria  *represent;    /*apontador para representação*/
               objecto    *próximo;      /*apontador para próximo objecto*/
               }
struct AFFIX{  objecto       *ligado;  /*apontador para objecto ligado*/
               transformação relativo; /*transformação para objecto*/
               affix         *próximo; /*apontador para próxima ligação*/
            }
```

Basicamente podemos distinguir a estrutura de dados OBJECTO (contendo toda a informação especifica dos objectos) e a estrutura de dados AFFIX (definindo as relações entre objectos).

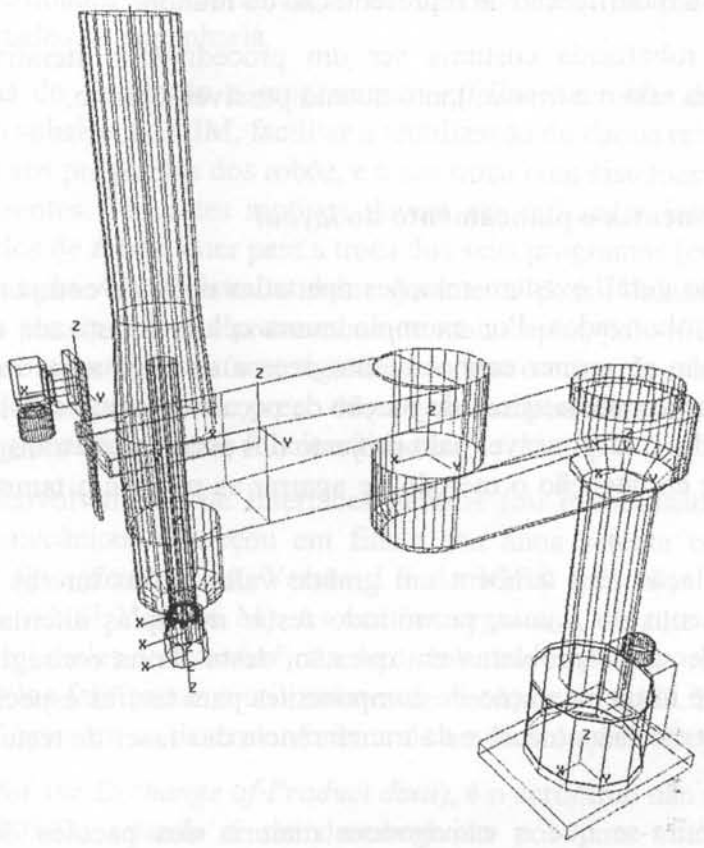


Figura 3.7: Referenciais utilizados na definição das ligações entre os componentes de um robô SCARA

A estrutura de dados OBJECT fornece geralmente um apontador para a representação geométrica do objecto (`geometria`) e contém a posição e orientação do objecto (`frame`) relativamente ao sistema de coordenadas do mundo do sistema simulado. Os campos adicionais, tais como o nome do objecto (`nome`) e uma *flag* adicional (`flag`) são utilizados durante os procedimentos de modificação e actualização. Para concatenar todos os objectos do mundo modelizado, a estrutura OBJECT contém um apontador correspondente ao próximo objecto (`objecto`). As relações para os outros objectos são definidas por uma lista de estruturas de ligação, que são concatenadas à estrutura OBJECT pelo apontador `affix`.

A estrutura de dados AFFIX contém um apontador para o objecto que se encontra ligado (`objecto`) e uma transformação (`transformação`) que define a posição e orientação do objecto ligado relativamente ao objecto corrente. Como um objecto pode ter relações com múltiplos outros objectos, as estruturas AFFIX correspondentes podem ser concatenadas dentro de uma lista utilizando o apontador de ligação `affix` (`proximo`).

Para combinar o modelo gráfico e a sua descrição cinemática, e para efectuar a disposição dos objectos no mundo simulado, utiliza-se geralmente um editor gráfico. A definição das

posições e orientações dos objectos pode ser efectuada relativamente aos outros objectos, relativamente a sistemas de coordenadas definidos previamente ou relativamente ao sistema de coordenadas do mundo. Além disso, costumam existir comandos de modelização adicionais que permitem a modificação da representação do mundo.

A definição da célula robotizada costuma ser um procedimento iterativo, de forma a conseguir-se alcançar para esta um *layout*, tanto quanto possível, óptimo.

1.3 Selecção dos componentes e planeamento do *layout*

Como é do conhecimento geral, existem relações apertadas entre os componentes de tarefas específicas de sistemas robotizados. Por exemplo numa célula robotizada de montagem, o projecto de uma garra não só requer conhecimento acerca da peça a ser manuseada, como também acerca das condições geométricas de junção da peça e de como tirá-la do *buffer* onde se encontra. Por outro lado, o responsável pelo projecto dos sistemas de transporte das peças e das fixações tem que ter em atenção o método de agarrar as peças e o tamanho e geometria das garras.

As ferramentas de simulação têm também um grande valor nestas tarefas de selecção dos componentes e planeamento do *layout*, permitindo testar múltiplas alternativas e escolher aquela que melhor se adequa ao problema em questão, desta forma conseguindo diminuir o período de planeamento e implementação de componentes para tarefas específicas, através da visualização das suas interacções no ecrã e da transferência das fases de teste e de verificação para a fase conceptual.

A título de exemplo refira-se que a esmagadora maioria dos pacotes de *software* para simulação e programação *off-line* de robôs trazem já incluídas bibliotecas com os dispositivos mais comuns utilizados nas simulações. De entre estas, são de realçar as bibliotecas de robôs, que são constituídas pelos robôs mais comuns no mercado (ou por todos aqueles que a empresa já tenha tido necessidade de modelizar), e as bibliotecas de ferramentas e garras, de que um exemplo bastante comum são as bibliotecas com pinças de soldadura para tarefas de soldadura por pontos e mesmo tochas para soldadura por arco eléctrico. A existência destas bibliotecas permite poupar bastante tempo ao utilizador, uma vez que já não necessita de efectuar a modelização destes componentes e pode experimentar (simular) vários componentes na realização de uma tarefa, de forma a determinar qual o que responde melhor aos requisitos da mesma.

1.4 O STEP - STandard for the Exchange of Product data

Actualmente é largamente aceite que as empresas industriais que alcancem uma integração em larga escala (mas não necessariamente uma automação total) das suas operações serão competitivas numa escala muito maior do que aquelas que só conseguirem a automação e integração dos seus processos de fabrico em áreas isoladas. A integração, neste contexto, é vista como a ligação dos diferentes componentes do sistema num sistema completo, que cumpra os objectivos e especificações do sistema de fabrico integrado.

Um exemplo de um problema típico de integração é encontrado numa empresa onde os projectos gerados num sistema de CAD não podem ser imediatamente transferidos para o departamento de produção (utilizando sistemas de CAM - *Computer Aided Manufacturing*) devido a incompatibilidades de comunicação de dados e falta de produtos compatíveis e sistemas de base de dados de engenharia.

Também os sistemas de simulação e programação *off-line* de robôs devem, de maneira a serem um verdadeiro subsistema CIM, facilitar a reutilização de dados relativos à definição do sistema robotizado e aos programas dos robôs, e a sua troca com sistemas de outras origens ou funcionalidades diferentes. Por estes motivos devem ser utilizados interfaces neutros quer para a troca de modelos de robôs, quer para a troca dos seus programas (este aspecto particular será abordado na secção 4.2.3). Isto também garante a possibilidade de trocar um dos componentes antigos por um novo, e provavelmente mais capaz, pelo menor preço e com um menor número de problemas de interface, significando que o custo de manutenção do sistema global é quase reduzido unicamente ao preço dos novos componentes, o mesmo se aplicando à extensão do sistema global com novas funcionalidades.

A "história" do desenvolvimento de interfaces neutros (ou normalizados) para a troca de dados de produtos mecânicos começou em finais dos anos setenta com o IGES (*Initial Graphics Exchange Specification*). A Versão 1.0 do IGES tornou-se uma Norma Norte-Americana em 1981 (ANSI Y 14.26 M). As versões seguintes, durante a década de oitenta, melhoraram as possibilidades da transferência de dados através de uma relativamente elevada aceitação do IGES pelos fabricantes e utilizadores de sistemas de CAD. No entanto, as várias deficiências do IGES levaram ao desenvolvimento de novas soluções [Bey, 1994].

O STEP (*STandard for the Exchange of Product data*), é o acrónimo não oficial para a Norma Internacional ISO 10303, estando a ser desenvolvido por uma comunidade de peritos internacionais dentro do ISO/TC184/SC4, desde 1984 [Kroszynski, 1994]. Este é um formato normalizado para troca de dados entre diferentes sistemas de CAD/CAM/CAE, em que se inclui, para além dos dados geométricos, a possibilidade de trocar dados tecnológicos respeitantes aos produtos e aos processos de fabrico incluindo, nomeadamente, uma extensão para a robótica (ver Figura 3.8).

A Europa tem estado activamente envolvida neste desenvolvimento principalmente através da contribuição de vários projectos ESPRIT (*European Strategic Program for the Research in Information Technologies*). O primeiro destes projectos que teve uma influência significativa no STEP, no período de tempo de 1985 a 1989, foi o Projecto ESPRIT 322, CAD*I (*CAD Interfaces*). Este projecto teve como objectivos desenvolver o interface entre sistemas de CAD, bem como destes para outros sistemas, ao nível da informação geométrica. O Projecto CAD*I influenciou o desenvolvimento da norma STEP numa grande extensão, mas não desenvolveu soluções neutras para a transferência de informação cinemática. A este seguiu-se o Projecto NIRO (*Neutral Interfaces for RObotics*), que continuou as contribuições ao STEP. As suas partes de definição de geometria, topologia e forma concordam quase a 90% com o modelo de referência desenvolvido no Projecto CAD*I. O mesmo se verifica para o formato físico do ficheiro. O grande progresso deste projecto, no entanto, foi ter adicionado ao STEP um novo aspecto: a cinemática [Schlechtendahl, 1994].

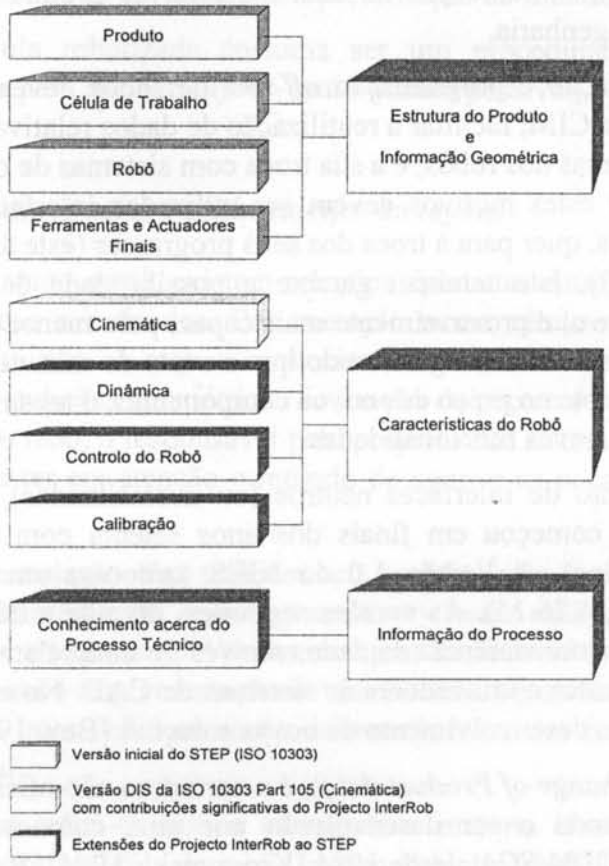


Figura 3.8: O STEP e as suas extensões para a robótica [Mikosch, 1995]

Neste projecto foi desenvolvido um modelo de informação para a cinemática para ser incluído no STEP. Esta proposta inclui o seguinte tipo de informação:

- topologia cinemática;
- associação geométrica;
- sequência das configurações cinemáticas e trajectos.

Actualmente, o STEP apresenta-se como a futura norma internacional para a definição de dados do produto [Trostmann, 1994]. Mas esta aproximação também apresenta problemas. A disparidade de capacidades em diferentes sistemas de CAD/CAM/CAE leva a que só seja permitida a transferência correcta daquelas características que têm uma correspondência exacta no formato neutro. Por vezes, o sistema que origina os dados tem capacidades não previstas no modelo de informação STEP. Estas são irreversivelmente perdidas ou substituídas por outras representações mais elementares, que resultam num modelo equivalente, mas degradado. Outras vezes, as características representadas na descrição neutra não têm correspondência no sistema que vai receber os dados. Nestes casos, a recuperação da informação é parcial.

Apesar de todos estes problemas, não exclusivos deste formato neutro, existem grandes expectativas em torno do STEP, tendo já sido investido um grande esforço nesta norma internacional de forma a fornecer ferramentas utilizáveis para a indústria.

2 Programação e simulação dos dispositivos

Terminado o projecto da célula há que passar à modelização do seu controlo, ao desenvolvimento dos programas para os vários equipamentos (programáveis) que vão fazer parte da mesma e, finalmente, à simulação. Estas tarefas são realizadas em passos subsequentes.

No que diz respeito aos programas dos dispositivos, estes devem ser verificados no respeitante a colisões (entre o robô ou qualquer outro equipamento que possua movimento e o restante equipamento da célula) e quanto à necessidade de todas as localizações onde se pretende que o robô efectue operações se encontrarem dentro do seu volume de trabalho, limite este fixado pelas respectivas juntas. Depois de desenvolvidos, os programas podem ainda ser optimizados quer em relação a tempos de ciclo, quer a *stress* mínimo nas juntas dos robôs, quer ainda no respeitante a outros parâmetros eventualmente fixados pelo utilizador.

2.1 Programação dos dispositivos

A partir de uma descrição da tarefa que se pretende realizar (que usualmente forma a base de informação para o programador) são gerados os programas de aplicação.

Para alcançar a tarefa, são executados no sistema um certo número de operações. Uma operação é uma sequência de elementos de trabalho e resulta em progresso em direcção à tarefa a realizar. Os elementos são definidos como constituintes de uma operação, que nem pode ser decomposta em termos da sua descrição, nem da sua medição temporal.

O objectivo da programação de um dispositivo é determinar todas as operações necessárias para este realizar uma determinada tarefa e impor as restrições necessárias a estas operações.

Logo, um programa do robô é gerado através da discretização da tarefa global numa sequência de operações que representam as diferentes subtarefas do trabalho completo. Este processo é repetido até que as operações tenham sido suficientemente detalhadas para serem expressas directamente nas primitivas de operação dos componentes da célula e, especificamente para o caso do robô, nos seus elementos de programação (ver Figura 3.9). Nos programas dos robôs correntes, as entidades programáveis mais pequenas (elementos de programação) são determinados pelas capacidades do robô (do seu controlador e da sua linguagem de programação) e não pela aplicação na qual o robô é utilizado. Pelo contrário, os parâmetros dos movimentos elementares, assim como as operações, a sua sequência e a estrutura do programa da tarefa são determinados pela aplicação especifica.

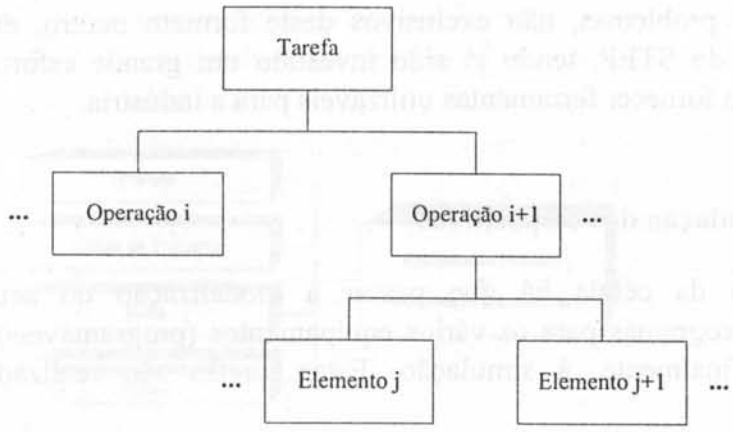


Figura 3.9: Decomposição de uma tarefa de montagem

Independentemente do tipo de aplicação em que se está a utilizar o robô, existem genericamente dois métodos alternativos para efectuar a sua programação: a programação explícita e a programação implícita.

2.1.1 Programação explícita

A programação explícita é caracterizada pela indicação, de forma explícita, de todas as instruções que o robô tem que executar; ou seja, o programador da aplicação do robô tem que o programar com as instruções que lhe dizem o que fazer numa determinada aplicação, utilizando a linguagem de programação do robô para esse efeito.

Caso se opte por este método de programação, o programador de robô tem que arranjar uma sequência de acções únicas dos equipamentos da célula do robô. Tem também que pensar e tomar em consideração os muitos problemas geométricos (por exemplo, no caso de uma aplicação de montagem, como agarrar uma peça de forma a ser capaz de realizar uma determinada operação de junção de partes, como mover uma peça de forma a que não ocorra nenhuma colisão durante o movimento) ou como utilizar sensores para colocar certas indeterminações debaixo do controlo.

Caso se pretenda trabalhar a um nível superior, poder-se-ia definir um conjunto de operações e programar o robô utilizando chamadas a essas operações como se fossem instruções de programação do robô. Por exemplo, no caso de um robô a executar tarefas de montagem, poderíamos definir as seguintes operações: Transferir, Aproximar, Agarrar, Afastar, Largar, Inserir. Estas operações seriam depois utilizadas pelo programador para realizar a programação do robô, instruindo-o a executar cada uma destas operações para cumprir o seu objectivo final, que será a montagem de um determinado produto dados os seus componentes.

2.1.2 Programação implícita

Esta técnica de programação é a oposta da programação explícita. O termo implícita refere-se ao facto de estes sistemas não especificarem as acções do robô de uma maneira explícita. Pelo

contrário, a especificação é dada implicitamente através da definição do objectivo das operações, ou pelos seus efeitos nos objectos, ao invés das próprias operações.

Esta técnica de programação é também designada de método de programação ao nível da tarefa (*task level*). Um sistema de programação ao nível da tarefa tem que determinar, de forma automática, uma sequência de acções de forma a resolver a tarefa especificada para o robô de uma célula de trabalho, resolvendo problemas de coordenação e evitando colisões entre os robôs, sendo a sequência resultante de movimentos do robô (o programa do robô) gerada automaticamente pelo sistema. Resumindo, estes planos têm que ser transformados num conjunto de operações, para poderem ser executados pelo robô.

Conclui-se que neste método de programação o robô é programado recebendo instruções de mais alto nível (que podem ser vistas como ordens), sem que o utilizador tenha que se preocupar com, por exemplo, a definição da estratégia de aproximação a um dado ponto ou objecto. Podemos dizer, por oposição à programação explícita, que enquanto naquele tipo de programação o operador tem que indicar ao robô como é que ele tem que efectuar uma dada operação, na programação implícita o operador tem que indicar o que é que o robô tem que fazer.

O desenvolvimento tecnológico a que se assiste nesta área aponta o método de programação implícita como referência futura na programação de robôs.

No capítulo seguinte abordaremos com maior detalhe a programação implícita de robôs afectos a tarefas de montagem, nomeadamente nas vertentes da integração do planeamento de tarefas com a execução das acções resultantes.

Independentemente da metodologia de programação adoptada, a verificação do programa desenvolvido pode ser sempre efectuada recorrendo a um *software* de simulação e programação *off-line* de robôs.

2.2 Ferramentas de suporte à programação de movimentos

Grande parte da programação de robôs diz respeito à geração dos trajectos que estes devem repetir. As ferramentas para suportar a programação de movimentos têm acesso directo ao interpretador de programas e ao modelo de simulação. Podem-se distinguir três tipos de ferramentas [Bernhardt, 1992]:

- ferramentas de programação;
- ferramentas de análise;
- ferramentas de cálculo e optimização.

Todas as ferramentas se encontram integradas num ambiente de operação homogéneo, como se pode ver Figura 3.10.

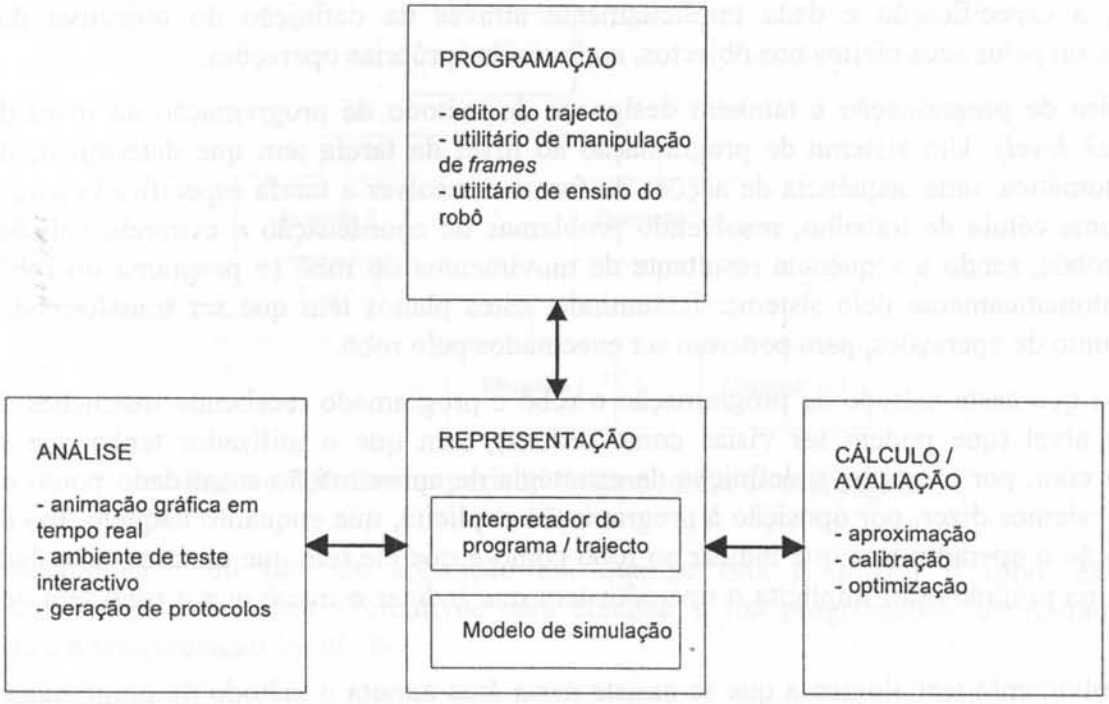


Figura 3.10: Ferramentas que suportam o planeamento do movimento [Bernhardt, 1992]

2.2.1 Planeamento do movimento

Durante a fase da programação deve-se efectuar um planeamento das trajectórias de forma a evitar colisões do robô com os outros dispositivos da célula. O objectivo do planeamento da execução de movimentos é planear os trajectos de movimento em relação a todos os factores de influência (apresentados na Figura 3.11). Esta tarefa de planeamento é executada por um planeador de trajectórias no caso da programação implícita e pelo utilizador no caso da programação ser explícita.

Um planeador de trajectórias é um sistema capaz de determinar, de forma automática, um trajecto isento de colisões entre duas posições dadas (geralmente o ponto inicial em que o robô se encontra e o ponto final do movimento, ponto este que se pretende alcançar com o movimento do robô), num ambiente com obstáculos. Geralmente, o trajecto é definido através de uma sequência de posições intermédias. Os resultados são trajectos de movimentos executáveis, que servem directamente como entrada para compiladores ou módulos de geração de programas.

Em particular, os seguintes aspectos têm que ser definidos para cada trajecto do robô:

- tipo de movimento (junta-a-junta, ou linear);
- velocidade;
- aceleração;
- tipo de aproximação às posições intermédias.

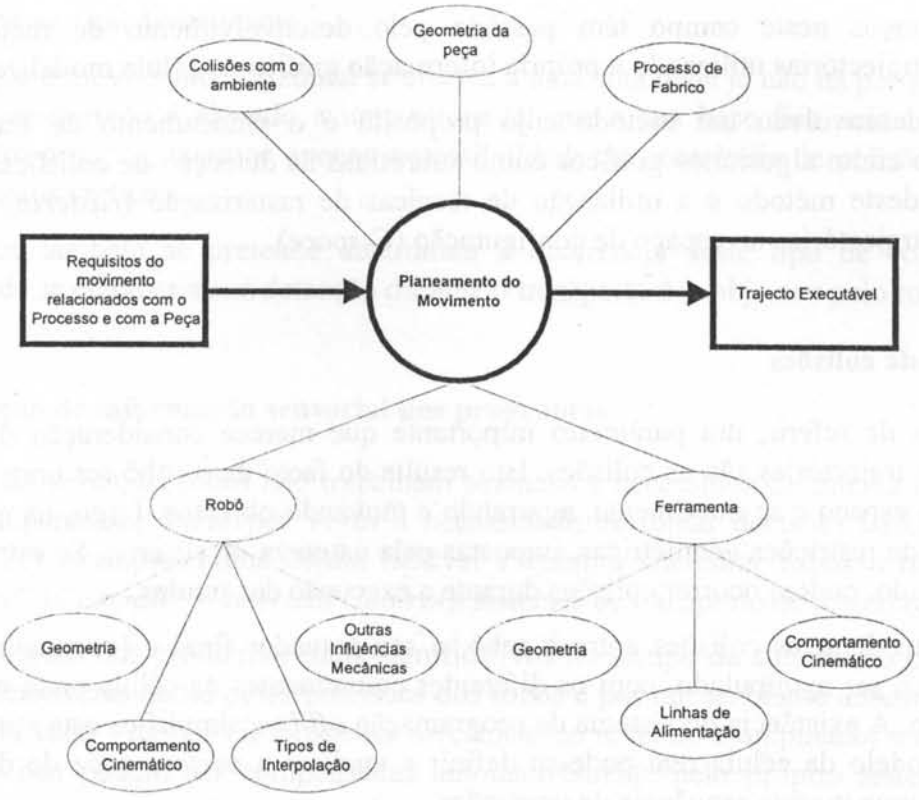


Figura 3.11: Factores determinantes para o planeamento do movimento

O planeamento das trajectórias inclui o planeamento da trajectória grosseira (a determinação de uma trajectória isenta de colisões para os movimentos de transferência de maior amplitude) e o planeamento da trajectória fina (a determinação do movimento para aproximar e abandonar as posições onde o robô tem que realizar as suas acções).

Este planeamento requer o modelo da cinemática inversa, mapeando as trajectórias planeadas no espaço das juntas [Müller, 1995]. De facto, as trajectórias isentas de colisões são obtidas definindo regiões de colisões no espaço das juntas. As verificações das colisões são realizadas quer para o robô, quer para a sua carga, no espaço cartesiano usando modelos de fronteira.

Para operações simples é suficiente modelizar os movimentos finos como movimentos lineares do robô, com um comprimento pré-definido. Em situações mais complexas estes movimentos finos, no entanto, não podem ser modelizados como movimentos lineares, sendo este o chamado problema do planeamento dos movimentos finos (*fine motion problem*).

O sucesso do planeamento dos movimentos finos depende fortemente do objecto geométrico para o qual é realizado o planeamento da trajectória, o chamado objecto do planeamento da trajectória (*path planning object*). Frequentemente, não é realizável executar um planeamento das trajectórias rápido para os movimentos finos se todo o robô for tomado em consideração, porque as restrições importantes resultam da geometria do objecto. Este não deve estar limitado ao objecto a ser manipulado, mas deve também incluir toda a garra na posição de agarrar (ou no estado aberto, dependendo do tipo de movimento fino), em adição da última parte da cadeia cinemática do robô (do punho do robô) [Reinhart, 1995].

Os últimos avanços neste campo têm passado pelo desenvolvimento de métodos de planeamento de trajectórias utilizando a própria informação gráfica da célula modelizada.

Müller (1995) desenvolveu um método cujo propósito é o planeamento de trajectórias utilizando para o efeito algoritmos gráficos como subrotinas na detecção de colisões. A ideia básica por trás deste método é a utilização de técnicas de rasterização (*rastering*) para o planeamento de trajectórias no espaço de configuração (*C-space*).

2.2.2 Detecção de colisões

Como acabamos de referir, um parâmetro importante que merece consideração durante o planeamento das trajectórias são as colisões. Isto resulta do facto de o robô ser uma entidade física que ocupa espaço e se movimenta, agarrando e movendo objectos. Logo, há que tratar todo o conjunto de restrições geométricas, impostas pela natureza do sistema. Se este aspecto não for considerado, podem ocorrer colisões durante a execução das tarefas.

É necessário considerar as colisões entre o robô, o seu actuador final e (eventualmente) o objecto que está a ser manipulado, com os diferentes componentes da célula onde o robô se encontra inserido. A existência do sistema de programação *off-line* simplifica esta verificação. Aqui, com o modelo da célula real pode-se definir e analisar a *performance* de diferentes trajectórias para uma mesma sequência de operações.

As possíveis colisões podem ocorrer entre o robô (principalmente o seu actuador final) ou entre os objectos transportados (durante o seu transporte) e os da célula.

Nas operações que tenham por objectivo agarrar ou largar um dado objecto, a garra do robô pode colidir com objectos que estejam na vizinhança daquele que se pretende agarrar ou na vizinhança do local onde se pretende largar o objecto. Para evitar este tipo de colisões, denominadas por colisões do tipo "garra-objecto" [Ramos, 1993], há que proceder à detecção de colisões sempre que o robô vai apanhar ou pousar um objecto. Também devem ser tomadas em consideração as colisões "garra-objecto", que envolvem a garra e o próprio objecto que esta vai apanhar, durante as manobras de aproximação e fecho da garra. Consegue-se, dessa forma, determinar se a garra é a indicada para apanhar um dado objecto.

Para além das colisões "garra-objecto", existe ainda o caso das colisões que podem ocorrer entre um objecto transportado e outros objectos. Uma vez agarrado pela garra do robô, um objecto efectua um movimento no qual acompanha o movimento associado ao referencial da garra do robô. Durante este movimento, pode dar-se uma colisão entre o objecto transportado e outros objectos. Essa situação que, de acordo com Ramos (1993) é apelidada por colisões do tipo "objecto-objecto", é de todo indesejável.

Por norma, o movimento efectuado pelo robô após agarrar um objecto (por exemplo, tendo em vista proceder a uma operação de manipulação) é composto pelos seguintes movimentos, antes de largar o objecto na sua posição final:

- movimento vertical ascendente;
- movimento horizontal paralelo ao plano XY do mundo;

- movimento vertical descendente.

Admitindo que o movimento horizontal se efectua a uma cota onde já não há perigo de colisão (o objecto transportado é elevado a uma altura tal que a sua base fica acima do topo de qualquer outro objecto), teremos apenas a possibilidade de ocorrência de colisão durante os dois movimentos verticais.

Uma vez que também se pretende determinar a ocorrência deste tipo de colisões, há a necessidade de se efectuar a sua detecção durante o transporte dos objectos pelo robô.

2.3 Integração de informação sensorial nos programas

Na maioria das vezes, os robôs não trabalham sozinhos e necessitam de interagir com outras peças de equipamento, tendo por vezes a necessidade de tomar decisões baseadas na sua envolvente. Por exemplo, numa célula flexível é comum encontrar robôs a interagir com máquinas CNC (*Computer Numerical Control*), sistemas de transporte de materiais, etc.

Além disto, alguns dos problemas mais significativos no campo da automação e da robótica residem nas incertezas dadas pelas precisões dos robôs e por um ambiente desconhecido, não estruturado ou vago. Algumas imprecisões mecânicas ao nível do manipulador e as incertezas no ambiente não podem ser compensadas automaticamente pelo próprio sistema. Mesmo pequenos desvios na posição, orientação e forma de todos os objectos que têm que ser manipulados não são permitidos sem informação sensorial, porque a execução da tarefa vai falhar num ambiente de incerteza.

Daqui surgem problemas ao nível do interface entre o programa desenvolvido *off-line* e estes dispositivos quando em funcionamento real, pelo que a inexistência da simulação de informação sensorial tem colocado restrições na utilização da programação *off-line* em robôs cujo comportamento é influenciado pela aquisição desta informação.

Conclui-se que para alcançar autonomia local ao nível dos movimentos finos, a simulação gráfica do robô dentro do seu ambiente deve ser estendida, tendo o sistema de simulação a capacidade de emular todas as funções dos diferentes tipos de sensores disponíveis no ambiente real, tais como sensores de distância, força/binário, e visão artificial. Só desta forma é possível alcançar uma cópia correcta do funcionamento do sistema real.

Além da sua utilização na programação gráfica *off-line* destas aplicações, a gama de aplicações da simulação da informação sensorial alarga-se também ao campo da teleoperação com grandes atrasos temporais [Brunner, 1995] e [Reinhart, 1995].

De acordo com Brunner (1995), deve-se utilizar percepção sensorial para trazer autonomia local ao nível do manipulador. Esta aproximação é aplicável quer no mundo real do robô, quer no mundo simulado. O processamento dos dados dos sensores é o requisito mais importante para alcançar um comportamento autónomo.

Por exemplo, Brunner (1995) introduz o conceito de TeleSensorProgramming apresentando um sistema no qual o ensino do robô é efectuado utilizando um sistema de simulação e programação *off-line* de robôs. Na programação do robô incluem-se a definição dos parâmetros que os sensores devem retornar quando em funcionamento real e a forma como o

robô deve responder a esses valores retornados pelos sensores. Depois da programação ser efectuada, o robô recorre aos dados dos sensores para executar o programa, não necessitando nesta fase de intervenção humana.

Existem três aproximações possíveis para a resolução destes problemas [Bernhardt, 1992]:

1. chamada a rotinas que fazem o tratamento da informação do interface;
2. comandos que não são interpretados na simulação e são depois traduzidos directamente para a linguagem do robô;
3. simulação do efeito destes dispositivos.

Relativamente a esta última possibilidade, Brunner (1995) distingue entre os sensores tácteis e os não tácteis. Os últimos podem ser emulados através do cálculo de simples funções de projecção ou interferência, mas os sensores tácteis, como sensores de força/binário, de acordo com este autor provocam dependência entre a sensibilidade do sensor e o posicionamento da garra, tendo estes factores que ser considerados ao emular o comportamento do sensor real.

2.3.1 Simulação de sensores de proximidade

A emulação do comportamento de um sensor de proximidade passa pela determinação da distância a que se encontra o sensor do objecto que se encontra mais próximo deste, na direcção para a qual o sensor se encontra orientado [Chen, 1992].

Este autor apresenta uma solução para efectuar a simulação de dois tipos diferentes de sensores ópticos de proximidade, baseados num LED (*Light Emitting Diode*) que emite um feixe de luz infravermelha e num fotodíodo que actua como receptor. O primeiro destes sensores de proximidade detecta a luz reflectida por um objecto que passe dentro do alcance do sensor, enquanto que o outro tipo de sensores detecta se o objecto interrompe o feixe de luz emitido pelo LED em direcção ao fotodíodo.

Chen (1992) simula o alcance destes sensores como sendo um segmento de recta no espaço tridimensional. A simulação do comportamento dos sensores consiste na verificação da intersecção do segmento de recta, representando o alcance efectivo dos sensores, com todos os objectos do mundo simulado que se encontram em frente do LED. Este problema é resolvido utilizando um algoritmo de *ray-trace*.

Por sua vez, Brunner (1995) apresenta um exemplo da emulação do comportamento de um sensor de proximidade LASER (*Light Amplification by Stimulated Emission of Radiation*) - *Range finder sensor*. Segundo este autor, é necessário calcular a interferência entre o feixe do sensor e as superfícies do objecto mais próximo. Assume-se, neste caso, que as características ópticas das superfícies dos objectos são desprezáveis, não se considerando as propriedades de reflexão dos objectos, mas calculando a interferência geométrica simples entre uma linha recta direccionada do sensor para o objecto, e o plano do objecto mais próximo do sensor. Chen (1992) utiliza este mesmo princípio para simular o seu *Point LASER Range Sensor* (sensor de proximidade baseado num feixe emitido por um LASER) e o *Imaging LASER Range Sensor* (sensor de proximidade baseado no varrimento de uma superfície por um feixe LASER)

2.3.2 Simulação de sensores de força/binário

De acordo com Brunner (1995), a simulação deste tipo de sensores pode ser considerada como uma aplicação de detecção de colisões, isto é, a interacção entre o sensor de força/binário e o objecto tocado pode ser reconhecida como um problema de detecção de uma colisão e quantificação da força exercida nessa colisão. Este autor refere os trabalhos de Goldenberg (1989), utilizando este último um método de simulação de sensores de força/binário baseado no princípio de d'Alembert. Este método requer um modelo dinâmico completo da célula simulada, modelo este que na maioria dos casos não se encontra disponível. Adicionalmente, o processo de cálculo das equações de movimento necessárias é muito demorado.

Daí que Brunner (1995) proponha uma aproximação baseada na deformação de um elemento flexível. Este elemento flexível emula um sensor de força/binário utilizando as deformações artificiais que podem ser deduzidas da violação de certas restrições. Este método converte a informação acerca do tipo de contacto geométrico em equações de restrições de movimento e força. Estes dados, em conjunto com a descrição da sensibilidade do sensor e a assumpção de um equilíbrio força/binário, contêm informação suficiente para calcular as forças de contacto virtuais.

2.3.3 Simulação de sistemas de visão artificial

No caso da simulação de *feedback* de sensores de visão, e contrariamente ao que se pode imaginar, nem sempre é necessário um modelo sofisticado de uma câmara de visão artificial para a emulação dos dados sensoriais retornados.

Num modelo detalhado, todos os parâmetros da câmara, quer internos quer externos, precisam de ser conhecidos, assim como as propriedades ópticas das superfícies e a descrição geométrica das fontes de iluminação. Chen (1992) utilizou um conjunto de parâmetros que lhe permitem simular o comportamento de uma câmara real, conseguindo obter imagens quer em tons de cinzento, quer coloridas.

Lu (1996) adoptou uma solução idêntica para efectuar o treino de um sistema de visão artificial *off-line*. Os objectos que o sistema de visão artificial poderia ter que identificar foram modelizados num sistema de CAD e os parâmetros da câmara do sistema de visão artificial foram determinados e testados de forma a reflectirem o funcionamento da câmara real. Desta forma é possível ao sistema de simulação determinar as perspectivas do objecto a identificar como imagens em tons de cinzento, que são similares às obtidas com a câmara real. Estas imagens podem ser depois processadas pelo sistema de visão para gerarem dados úteis para o reconhecimento dos objectos, antes que o sistema entre em funcionamento real.

Quando o importante não é o comportamento físico, nem os valores calculados exactos de *pixels*, mas só a sua interpretação, o sistema só necessita de emular os dados de saída do sistema real [Brunner, 1995]. Se o sistema real de processamento de imagem funciona a um nível baseado em características, a simulação tem que retornar os valores apropriados para estas características.

Sendo assim, os dados sensoriais do sistema de visão podem ser obtidos a partir do modelo de dados geométrico, sem entrar em linha de conta com as fontes de luz e com as propriedades ópticas das superfícies, tais como a reflectância e a refracção.

Portanto consideram-se pontos, linhas e superfícies como características dos objectos, segundo o modelo geométrico do mundo. A projecção no plano da câmara do espaço tridimensional é efectuado em relação aos parâmetros pré-definidos da câmara. O resultado é a característica correspondente no plano de projecção.

2.4 Simulação da execução do processo

O objectivo último destes simuladores, tal como o seu nome indica, é o teste dos programas de aplicação desenvolvidos usando a simulação. Como já vimos, o planeamento e programação de uma célula robotizada necessitam da simulação como uma parte importante de todo o sistema, sendo um requisito importante para os sistemas de programação *off-line* o teste da exequibilidade e praticabilidade, ao nível da planta fabril, dos programas desenvolvidos para as aplicações dos robôs.

Chegados a este ponto, e depois de o *layout* da célula ter sido estabelecido e terem sido desenvolvidos os programas para os dispositivos programáveis desta, o simulador está configurado para simular a tarefa de produção descrita e apresentar o efeito que o programa de aplicação vai ter no ambiente de produção, pelo que deve efectuar-se a simulação do conjunto. Tal como para a programação, também para a configuração do simulador a descrição da tarefa é a principal fonte de informação.

Baseado na descrição da tarefa, o utilizador do simulador tem que detectar erros no programa de aplicação relacionados com a praticabilidade e encadeamento da execução da tarefa. Isto inclui a detecção de colisões (para este efeito, as câmaras virtuais existentes na maioria dos simuladores, especialmente através do seu *zoom*, podem ser utilizadas para melhorar a saída visual e para fornecer ao utilizador um *feedback* visual óptimo durante a realização de tarefas de posicionamento fino) e a verificação da velocidade e da aceleração das juntas dos dispositivos. Além disto, têm que ser testadas as interacções com os periféricos. O tempo de ciclo do programa de aplicação simulado é também um resultado importante, que permite uma optimização na direcção de uma célula de fabrico totalmente sincronizada.

Além do mais, dentro do simulador têm que ser disponibilizadas ao utilizador algumas características de planeamento e programação, de forma a ser possível testar e validar o programa da célula planeada, procedendo a algumas alterações caso se justifique.

O sistema é baseado num interpretador conduzido pelo tempo e é testado correntemente num modo conduzido pela operação. Deve permitir a simulação do fluxo do programa da célula e oferecer várias possibilidades de análise. Em relação à *performance* temporal, os resultados melhoram a qualidade do programa gerado anteriormente. O sistema também deve permitir a detecção de *deadlock's* devidos a situações de conflito irresolúveis ou motivados por problemas relacionados com o fluxo de materiais. O utilizador deve ainda ser capaz de gerar interrupções na simulação da execução (para verificar o estado do sistema ou alterar um

conjunto de parâmetros) e verificar o efeito dessas alterações após reinicialização da simulação.

O simulador tem que se encontrar numa forma adequada para preencher estes diferentes requisitos, para isso sendo necessário distinguir as diferentes formas de simulação: por um lado, ao nível de célula é necessária uma simulação mais orientada à comunicação num nível lógico; por outro lado, ao nível dos equipamentos ou componentes das células (por exemplo, robô), é necessária uma simulação mais orientada ao movimento.

A simulação da execução dos programas dos dispositivos, no ambiente distribuído da célula robotizada, é realizada num nível lógico. A modelização do ambiente da célula de uma forma independente da tarefa requer a representação da globalidade das funcionalidades da mesma. Esta representação funcional tem que incluir quer as propriedades estáticas, quer as propriedades dinâmicas da célula. Os dois modelos são:

- o modelo estático, descrevendo o *layout*, a funcionalidade dos componentes e as relações entre os componentes;
- o modelo dinâmico, descrevendo a sincronização entre os componentes (incluindo as condições de monitorização) e a concorrência dos processos.

Estas duas representações, em conjunto com um mecanismo de disparo, formam o modelo de controlo da célula: o controlador interpreta o escalonamento planeado da célula e dispara as operações dos componentes, de acordo com o estado actual e as condições de monitorização permitidas.

O teste de um programa de aplicação para um componente activo da célula, tal como um robô, vai ser executado ao nível do movimento. Este teste de praticabilidade contém o teste das trajectórias definidas, relacionadas com posições, orientações, velocidades e acelerações. Além do mais, têm que ser testados e verificados os comandos dos actuadores finais e as interacções com os periféricos.

Como visto no capítulo anterior, o simulador requer uma representação de todos os componentes da célula e do seu comportamento. Estas representações são chamadas modelos de simulação e são estruturadas da seguinte forma:

- modelos de controlo, que descrevem o comportamento dos movimentos dos componentes;
- modelos cinemáticos, que contêm as relações dos referenciais das diferentes ligações;
- modelos geométricos, que descrevem as representações gráficas dos componentes.

A Figura 3.12 apresenta a conexão funcional destes modelos para o exemplo da simulação do movimento de um robô.

O programa de aplicação do robô é carregado no seu modelo de controlo. Este modelo de controlo interpreta o programa de aplicação e fornece ao modelo cinemático os valores das juntas das ligações. Dentro do modelo cinemático, são calculados os referenciais descrevendo a posição e a orientação de cada ligação do robô. A conexão destes referenciais aos modelos geométricos relevantes permite a visualização do movimento do robô num sistema gráfico.

Figura 3.12: Simulação da execução do movimento [Bernhardt, 1992]

2.5 Tratamento de excepções

Apesar de se proceder à simulação da execução dos programas desenvolvidos e da integração neste programa da informação sensorial (e sua simulação), por vezes costumam surgir alguns problemas aquando da execução na planta fabril dos programas desenvolvidos.

Um dos maiores problemas associados à programação *off-line* é o facto de os modelos simulados do robô e do seu ambiente estarem muitas vezes incompletos. Na maioria dos casos, este modelo é estático e os objectos são supostos permanecerem nas suas localizações a menos que sejam explicitamente movidos. Na realidade, as influências no ambiente do robô são muito mais complexas. Desvios nos tamanhos das partes, colisões com objectos não representados no modelo, ou o escorregamento de partes, são exemplos das influências que podem levar a uma quebra no processo produtivo.

Isto deve-se ao facto de nos sistemas de programação *off-line* ser, na maioria das vezes, muito difícil modelizar o ambiente de trabalho de um robô ou prever o valor de variáveis ambientais relevantes. Muitas variáveis ambientais têm uma natureza discreta e apresentam relações complexas entre elas. Como resultado, o modelo interno do ambiente do robô, muitas vezes só parcialmente reflecte o estado actual da variável ambiental.

O modelo existente no sistema de simulação e programação *off-line* deve, apesar disto, corresponder de forma o mais aproximada possível à situação real para assegurar, tanto quanto possível, que os programas do robô resultantes funcionem correctamente.

Para evitar estes problemas deve-se tentar prever os erros que podem acontecer e tentar planear com avanço uma forma de recuperação.

Esta área de investigação, que trata dos problemas de tratamento de excepções (ou recuperação automática de erros), envolve o planeamento e a execução das operações de um sistema robotizado que não foram previstas a priori durante a execução de um programa do robô. Tradicionalmente, estas situações levariam a uma paragem completa do robô e da sua actividade e necessitariam da intervenção do operador.

A forma mais eficaz de efectuar o tratamento das excepções é recorrendo à sensorização dos equipamentos da célula robotizada e tratando a informação recolhida pelos sensores durante a execução do programa na planta fabril. O programa gerado *off-line* e descarregado para o robô já deve, no entanto, ter previstas as acções a tomar em função da informação possível de ser retornada pelos sensores [Meijer, 1992].

3 Optimização do sistema planeado e dos programas dos dispositivos

Após a obtenção dos primeiros resultados da fase de simulação passa-se, geralmente, por uma fase de optimização da solução desenvolvida, o que não só envolve a optimização do *layout*, como dos próprios programas dos dispositivos constituintes.

3.1 Optimização do *layout*

A optimização do *layout* é um requisito importante da tarefa de planeamento do sistema, acabando por ser o seu último passo. O objectivo global é a melhoria das características e da *performance* do sistema em relação à tarefa a ser executada.

Nesta fase verifica-se se os dispositivos acabados de programar conseguem (ou não) alcançar todos os pontos pretendidos, sem se excederem os limites das juntas e sem que se verifiquem colisões. Caso ocorra algum destes problemas pode-se recorrer a duas soluções: por um lado, tentar alterar o programa do robô de forma a que este alcance a posição pretendida com outra configuração do punho e, por outro lado, alterar o posicionamento do dispositivo dentro da célula, isto é, efectuar um pequeno ajuste do *layout*.

Caso nenhum destes expedientes solucione o problema, pode ser necessário trocar o dispositivo (que causa o erro) por um outro com características diferentes. Neste caso, os pacotes de *software* que apresentam bibliotecas de dispositivos já modelizados são de grande utilidade, uma vez que é possível, por exemplo no caso de um robô, substituir o que estava a ser utilizado por um outro, também existente na biblioteca, e determinar se com este último já se conseguem alcançar os objectivos pretendidos. Esta é uma funcionalidade oferecida pelos produtos de simulação de robôs que é de grande utilidade durante o projecto de novas células robotizadas. Obviamente, após a célula estar instalada, esta funcionalidade serve, por exemplo, para testar as ferramentas e os actuadores finais a utilizar para uma dada tarefa do robô.

De acordo com Bernhardt (1992) existem duas aproximações utilizadas usualmente para a optimização dos problemas de fabrico:

- técnicas de simulação;
- o conhecimento e a experiência das pessoas (peritos na matéria).

Dao (1995) defende no entanto que, isoladamente, nenhuma das duas aproximações tem sido totalmente coroada de sucesso na maioria dos casos, particularmente no domínio da optimização do *layout*, considerando o aspecto dinâmico e a complexidade destes problemas.

Daí que comecem a surgir sistemas de simulação baseado em técnicas de Inteligência Artificial, tais como os propostos por Kóvacs (1994) e Dao (1995). Estes sistemas são uma combinação das duas aproximações referidas acima, sendo esta combinação considerada poderosa e adequada para a resolução optimizada de problemas deste tipo.

Nestes sistemas, existe um interface com o utilizador para que este possa escolher os parâmetros de entrada e os objectivos a alcançar, efectuando o sistema a optimização do *layout* em função dos parâmetros e objectivos escolhidos, baseado na simulação de diversos cenários alternativos. Os resultados são então apresentados sob a forma gráfica.

3.2 Optimização dos programas dos dispositivos

Após uma primeira optimização do *layout* da célula passa-se à fase da optimização dos programas dos dispositivos. Estas duas fases são muitas vezes executadas em ciclo, com iterações sucessivas.

Durante esta fase é habitual optimizarem-se os seguintes aspectos da programação:

- tempos de ciclo;
- sincronização de processos;
- trajectos dos dispositivos após optimização do *layout* das células.

A optimização dos tempos de ciclo está intimamente relacionada com a sincronização de processos. Na maioria das vezes os robôs não se encontram a trabalhar sozinhos nas células de trabalho; pelo contrário, têm que cooperar com outros dispositivos mecânicos, programáveis ou não, e que também têm os seus próprios movimentos. Surge então a necessidade de sincronizar o funcionamento desses vários dispositivos. Este é, por norma, um trabalho demorado e sujeito a grandes riscos de colisão entre dispositivos, motivados por erros de programação ou distracção do operador, o que pode ter consequências desastrosas.

É aqui que a simulação entra em acção, permitindo fazer múltiplos testes de sincronização e testar várias soluções alternativas, assim evitando a paragem demorada dos equipamentos em questão e eliminando as consequências desastrosas dos erros de programação.

Também se costuma optimizar as trajectórias dos robôs, quer eliminando alguns pontos do trajecto que provem ser desnecessários (o que contribui para diminuir o tempo de ciclo), quer incluindo outros (com vista a evitar colisões e aproximações perigosas a outros componentes da envolvente do robô). Pode-se também optar por variar a velocidade com que os

dispositivos executam os programas e observar as consequências que isso provoca (quer em tempo de execução das operações, quer em alterações nas trajectórias dos robôs).

4 Geração e descarga do programa do robô

O objectivo último da programação *off-line* é a geração de programas de aplicações do robô que sejam executáveis na sua globalidade na planta fabril, sem haver a necessidade de ajustes. Isto significa que as interacções manuais para correr esses programas no robô real devem ser minimizadas. Para alcançar este objectivo devem ser considerados os seguintes aspectos:

- os desvios entre os dados nominais (usados no planeamento do *layout* e na programação), e os dados reais da célula de trabalho (e do robô): problema vulgarmente designado por calibração;
- a geração dos programas na linguagem nativa dos controladores dos robôs;
- e, por último, a ligação das comunicações entre a área de *off-line* e a planta fabril.

4.1 Calibração do modelo do robô e da célula

A validade de uma simulação depende, em boa medida, da forma como o modelo está de acordo com a realidade.

Quando se concebe *off-line* um programa do robô, surge o problema da adaptação desse programa ao ambiente de trabalho real. Tal problema resulta de pequenos desvios, sempre existentes, entre o modelo da célula de trabalho e a realidade, agravados pela precisão de posicionamento do robô. Para ultrapassar esta dificuldade têm sido desenvolvidos procedimentos de calibração de robôs bastante poderosos [Schröer, 1992] [Deneb, 1992] [Owens, 1994] [Owens, 1995].

Complementarmente, um robô devidamente calibrado pode desempenhar um papel importante em novas aplicações de programação *off-line*: tal robô pode ser usado como um dispositivo de medida para detectar os desvios entre a célula de trabalho real, a sua envolvente e os modelos de CAD nos quais se baseia a programação *off-line*.

Uma alternativa à calibração das células robotizadas passa pelo desenvolvimento *off-line* dos programas para os equipamentos, os desvios sendo depois compensados *on-line* recorrendo a informação sensorial. Um exemplo desta solução foi implementada por Lu (1996).

No sistema descrito por este autor, os programas para um robô que se encontra a efectuar a carga/descarga de máquinas CNC são gerados *off-line*, recorrendo a um sistema de simulação de robôs. Quando em funcionamento real, o robô executa os programas gerados *off-line* até se encontrar perto das peças, e daí para a frente passa a recorrer à informação proveniente dos sensores (de força/binário e sistema de visão artificial) com que se encontra equipado para as apanhar, o mesmo se passando quando o objectivo é carregar uma máquina. Recorrendo a uma solução deste tipo deixa de ser necessário recorrer à calibração do modelo da célula existente no *software* de simulação.

4.2 Geração do código para o robô

Depois de os programas dos dispositivos terem sido desenvolvidos, simulados e optimizados, é necessário passá-los para os controladores dos robôs.

Existem duas filosofias que os simuladores de robôs costumam adoptar em relação à programação: por um lado, existem simuladores que permitem que o desenvolvimento dos programas dos dispositivos seja efectuado na linguagem nativa dos controladores desses mesmos dispositivos; por outro, existem outros produtos de simulação em que a programação dos dispositivos é efectuada numa linguagem dita neutra, sendo depois necessário traduzir os programas gerados para a linguagem nativa dos controladores dos dispositivos. Quer uma solução, quer a outra, têm os seus pontos fortes e os seus pontos fracos.

4.2.1 Programas desenvolvidos na linguagem nativa do controlador do robô

Caso o programa tenha sido desenvolvido na linguagem nativa do controlador do dispositivo que se pretende programar, a única coisa a fazer é descarregar o programa para o controlador do robô, onde este correrá (em princípio) sem problemas.

Esta solução apresenta a grande vantagem de o utilizador do robô só necessitar de conhecer a linguagem de programação do robô que pretende programar. É uma solução muito boa no caso de o utilizador dos robôs só dispor de equipamentos com uma linguagem de programação, visto que não necessita de aprender mais nenhuma.

No entanto, são muito poucos os produtos de simulação que permitem fazer a programação dos dispositivos na linguagem nativa dos seus controladores, sendo no entanto uma solução muito utilizada quando o *software* de simulação é desenvolvido por uma empresa que já comercializa robôs para apoio à sua programação. Nestes casos, costuma também ser comum que a biblioteca de dispositivos, incluída no pacote de *software*, seja constituída unicamente por equipamentos comercializados por esse mesmo fabricante.

Por oposição, conclui-se facilmente que esta solução apresenta graves inconvenientes quando o utilizador do *software* dispõe de equipamentos de vários fabricantes, com diferentes linguagens de programação, o que o obrigaria a aprender tantas linguagens de programação quantas as dos seus equipamentos.

4.2.2 Programas desenvolvidos em linguagem neutra

Uma exigência importante normalmente requerida pela programação *off-line* de robôs é a da facilidade da sua adaptação a uma grande variedade de diferentes controladores de robôs.

Uma situação que ocorre cada vez com maior frequência é a necessidade de transportar uma aplicação para um novo tipo de robô. Nestas situações é desejável reutilizar o programa antigo. No entanto, e uma vez que a programação de robôs é feita (na maioria dos casos) com uma linguagem específica de cada fabricante de robôs, isto não é normalmente possível devido às incompatibilidades entre as diferentes linguagens associadas aos controladores de

robôs. Esta situação encontra-se em franco contraste com a situação da programação genérica, em que a programação estruturada e a independência da máquina têm sido o estado da arte já há algumas décadas.

Como forma de ultrapassar os problemas acabados de referir é necessária a geração de um código flexível. Para esse efeito têm sido adoptadas duas soluções.

Por um lado, alguns pacotes de *software* de simulação vêm equipados com uma linguagem neutra, na qual são desenvolvidos os programas para os robôs independentemente do seu controlador, sendo estes programas traduzidos para a linguagem nativa do controlador do robô recorrendo a um tradutor ou *cross compiler*, após os programas do robô serem finalizados, sendo depois descarregado para o controlador do robô.

Um tradutor é um programa que é capaz de traduzir de uma linguagem para a outra. Esta solução é muitas vezes a preferida porque oferece uma solução directa para o problema: "eu tenho uma linguagem do sistema de programação *off-line* e eu quero usar esta linguagem num robô que percebe/entende outra linguagem". O maior problema de compilar o programa em linguagem neutra para o formato nativo é que o formato nativo pode não ser capaz de expressar tudo o que a linguagem neutra é capaz de expressar, ou vice-versa.

Outra solução passa pelos esforços de normalização das linguagens de programação de robôs, aspecto que tem merecido uma grande atenção nos últimos tempos e de que os trabalhos mais palpáveis (a nível Europeu) são as propostas de normalização das linguagens ICR (*Intermediate Code for Robots*) e IRL (*Industrial Robot Language*). Desta forma o programa pode ser executado num interpretador que passa os comandos de movimento ao planeador de trajectórias do controlador do robô.

Um interpretador é um programa que é capaz de ler outro programa linha a linha e executar o seu conteúdo passo a passo. O controlador de um robô inclui sempre pelo menos um interpretador: o que lê a linguagem nativa do robô.

A longo prazo a solução do interpretador é a aproximação mais vantajosa, uma vez que não é necessário traduzir o programa neutro do robô antes de este ser executado [Bey, 1994]. No entanto, os robôs industriais actuais não são projectados para serem usados com interpretadores (uma vez que o acesso em tempo real ao controlador não está disponível) e portanto a aproximação do compilador é ainda relevante.

Na programação de robôs, as principais vantagens que resultam da utilização dos interfaces neutros são as seguintes:

- não é necessário nenhum esforço de programação extra quando a aplicação é transferida de um sistema para outro, nomeadamente se o fabricante for diferente;
- para os vendedores de simuladores, a principal vantagem residirá na necessidade de um único interface de linguagem na saída dos seus sistemas, não existindo mais a necessidade de desenvolver um pós-processador específico para cada sistema-alvo.

Convém realçar que estas linguagens não são totalmente neutras no que se refere à execução prática de um programa num robô. Em teoria, deveria ser possível gerar um programa *off-line* e, depois, transferir o programa para um robô arbitrário suportando a linguagem neutra e executar o programa nesse robô. No entanto isto é difícil de realizar na prática, porque os

robôs diferem muito nas suas capacidades físicas. A título de exemplo podemos referir os seguintes problemas:

- o volume de trabalho de diferentes robôs não é igual. Um programa incluindo trajectos com dois metros de comprimento não pode ser executado por um robô em miniatura;
- um robô de cinco eixos não pode resolver um problema que requer seis eixos.

É necessário ter estas restrições gerais em mente. Estas linguagens não são totalmente neutras no sentido que um mesmo programa pode ser executado em qualquer robô que as suporte. Neutro significa aqui que são independentes do meio do sistema e da máquina, isto é, garantem que o programa do sistema gerador representa informação com significado para o sistema utilizador (ou receptor).

Outra faceta deste problema é a necessidade de a linguagem neutra necessitar de dispor de instruções específicas da aplicação, caso por exemplo, de instruções específicas para aplicações de robôs em tarefas de soldadura.

A solução da linguagem neutra apresenta a vantagem de o utilizador do *software*, que disponha de vários robôs com diferentes linguagens de programação, só necessitar de aprender uma linguagem de programação (a do *software* de simulação em que vai desenvolver os programas para os robôs), ficando entregue a tradutores (*cross compilers*) a tarefa da geração do código em linguagem nativa dos controladores dos robôs.

4.2.3 Normalização das linguagens neutras de programação de robôs

4.2.3.1 A linguagem neutra IRL - Industrial Robot Language

A linguagem IRL, também conhecida por PLR (*Programming Language for Robots*) foi introduzida pela primeira vez no "21st International Symposium for Industrial Robots", em Outubro de 1990. No entanto, o trabalho de normalização de uma linguagem de programação de robôs iniciou-se em 1988 na Alemanha, a nível nacional pela secção NAM (*Standards for Machinery Tools*) da DIN, e em paralelo na ISO TC184/SC2/WG4 a um nível internacional, tendo o primeiro rascunho do IRL sido publicado em 1989 (DIN/89).

A ideia do IRL é introduzir no controlo de robôs as vantagens das linguagens de programação genéricas de alto nível, sendo a sua definição baseada nos conceitos modernos para programação estruturada.

Esta linguagem apresenta todas as características principais de uma linguagem de programação genérica de alto nível tal como o Pascal (a sintaxe é similar à do Pascal, sendo mesmo idênticas muitas construções), incluindo a programação modular, controlo de fluxo do programa, procedimentos, funções, variáveis e constantes locais e globais, conceito de bloco, recursividade e outros. Adicionalmente, contém um conjunto de construções específicas da robótica, essencialmente relacionadas com o facto de um braço de robô ser passível de um movimento a uma dada velocidade num espaço tridimensional, e que diferentes tipos de sensores podem ser utilizados para o controlar. Foi dada uma atenção especial para permitir o

ensino de localizações de robôs, para permitir o controlo de vários robôs de um mesmo programa e para permitir um meio estruturado de adaptação do sistema [Schmiedecke, 1994]. O IRL também virá a permitir, numa futura extensão, a execução paralela de partes de um programa.

As vantagens a obter com esta linguagem são programas independentes do robô, mais fáceis e rápidos de fazer, ler, entender e reestruturar. Como contrapartida, o tempo de execução das instruções é superior; no entanto, comparado com o tempo que leva um movimento do braço do robô, este aumento do tempo de execução das instruções é praticamente irrelevante.

A linguagem IRL encontra-se presentemente adoptada na Alemanha (Norma "Din 66312 Teil1").

No que diz respeito à ISO, encontra-se praticamente parado o trabalho de desenvolvimento da Norma PLR, equivalente e fortemente influenciada pela linguagem IRL.

Sobre o mesmo assunto, a posição Norte-Americana é a de adoptar uma linguagem baseada em C, ou nenhuma; esta tendência conjuga-se com a oposição Japonesa à linguagem PLR, já que o Japão prefere uma linguagem japonesa baseada no BASIC (chamada SLIM).

No entanto, conclui-se que todos os grandes produtores destas tecnologias estão atentos ao problema e a desenvolver trabalho no sentido de o ultrapassar.

4.2.3.2 A linguagem neutra ICR - Intermediate Code for Robots

Uma vez que a tendência actual na programação de robôs é a programação *off-line*, é importante focar na funcionalidade do interface entre o sistema de programação do robô e do próprio robô.

A maioria das linguagens dos robôs são construídas com base na mesma filosofia: serem fáceis de utilizar pelos programadores. No entanto, a programação *off-line* recorrendo a ferramentas computacionais pede outras qualidades da linguagem do robô. Não é mais uma necessidade torná-la fácil para os humanos a entenderem, uma vez que os programas vão ser trocados entre computadores, devendo antes ser optimizada com respeito a outros requisitos. Deve ser capaz de expressar tarefas gerais do robô, ser fácil de gerar e fácil de ler e executar por máquinas.

O CLDATA (*Cutter Location DATA*) foi um dos primeiros códigos para a programação de máquinas CNC, e é basicamente utilizado para definir o trajecto que a ferramenta deve seguir reflectindo a funcionalidade básica de uma máquina CNC típica.

O VDI alemão adoptou esta ideia no código de programação de robôs IRDATA (*Industrial Robot DATA*), mas estendeu o IRDATA para incluir um largo conjunto de instruções para operações aritméticas, controlo de fluxo de programas e operações de entrada/saída. Este foi um dos primeiros passos para a normalização de um interface entre a programação de robôs e o seu controlo, mas faltavam a esta norma Alemã (DIN 66 313, Part 1) muitas funções necessárias para a programação de robôs.

No final dos anos oitenta, a ISO também reconheceu esta necessidade, tendo sido iniciado o desenvolvimento do ICR, tendo sido publicados pela ISO em 1989 os primeiros documentos respeitantes a este formato neutro.

O ICR também tem as suas raízes no código CLDATA, de programação de CNC, e no código IRDATA. Uma vez que o objectivo era o mesmo do IRDATA, este foi utilizado como uma base importante para o trabalho de desenvolvimento do ICR, sendo semelhantes os conceitos básicos e elementos principais dos dois códigos.

O objectivo do ICR é ter uma forma normalizada de transferir programas de sistemas de programação *off-line* para controladores de robôs, que suporte todas as características que um controlador de robô geralmente realiza e todos os elementos para descrever a tarefa, e não uma linguagem que o utilizador usa na programação do robô. O código simplesmente não é apropriado para este propósito, uma vez que se encontra optimizado com respeito a uma rápida execução na máquina, tendo sido projectado para o nível de abstracção intermédio (o que é um pré-requisito para uma execução rápida da máquina [Clausen, 1994]). Adicionalmente, apresenta também como objectivos a redução do esforço de implementação na construção de controladores de robôs e a programação utilizando módulos de programas redefinidos.

Um código intermédio, neste contexto, é um código *assembly* universal, independente do sistema, num nível que se situa entre o código máquina e as linguagens de alto nível, tais como o Pascal e o C. O código descrito é chamado "código" porque é uma linguagem de baixo nível com semelhanças ao código *assembly*.

A independência do robô é assegurada através do projecto da linguagem ICR para um modelo de referência do robô que reflicta a funcionalidade geral de um robô industrial. As funções comuns à maioria dos robôs industriais estão incluídas directamente no conjunto das instruções do ICR. Exemplos são as instruções de movimento, as de controlo das entradas/saídas e as de controlo das garras. Funções mais específicas de determinados produtos não são suportadas directamente, estas funções sendo antes construídas através da combinação das instruções básicas.

O código ICR consiste numa sequência de registos, cada um deles representando uma função que é executada imediatamente pelo controlador do robô. As funções e os seus parâmetros, expressas pelo código, definem um super-conjunto das funções do controlador dos robôs. Isto significa, felizmente, que cada controlador de robô pode utilizar um menor ou maior subconjunto dos elementos de código do ICR para exprimir as suas funcionalidades. Mas o controlador de um robô não necessita de incluir todos os elementos do código ICR.

As funções incluem operações aritméticas e de *stack*, que permitem uma avaliação eficiente de expressões aritméticas. Partes do código, chamadas unidades ICR, podem ser um programa do utilizador, um módulo do utilizador ou ficheiros de transferência utilizados para troca de dados entre vários componentes de programação e controlo. Também é possível trocar dados de uma unidade de controlo de um robô para outra, por meio de transferência de ficheiros. Tais operações também podem ser definições de dados.

Apesar de a proposta de normalização do ICR apresentada à ISO em 1992 ter sido rejeitada, o trabalho de normalização tem continuado.

Na Alemanha, a DIN/NAM decidiu em 1992 tornar a recomendação IRDATA, anteriormente publicada pela VDI e que influenciou em grande medida o ICR, como Norma Alemã para o nível de código (DIN/90).

Por sua vez no Japão, o JIS (*Japan Industrial Standards*) efectuou em Fevereiro de 1992 trabalho preliminar numa outra proposta intitulada STROLIC (*STandard RObô Language in Intermediate Code*).

Estas são duas propostas de normalização que estão a ser preparadas a nível internacional. Existe uma grande competição entre as várias propostas e será necessário um esforço significativo para alcançar uma norma internacional.

Pode-se ver na Figura 3.13 como é prevista a utilização das linguagens neutras de programação de robôs, IRL e ICR, bem como a troca de dados em formato STEP.

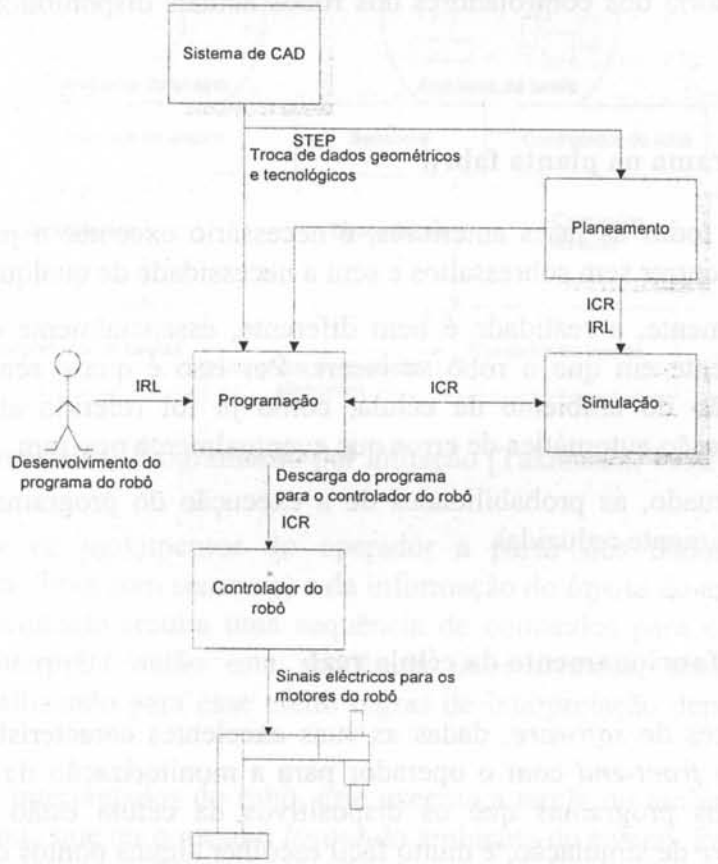


Figura 3.13: Utilização dos interfaces neutros no desenvolvimento de aplicações de robôs (Adaptado de [Bey, 1994])

4.3 Descarga do programa para o robô

Chegados aqui, o utilizador já dispõe do programa desenvolvido, testado, optimizado e gerado na linguagem nativa do controlador do robô, pelo que só falta descarregá-lo para o controlador do robô, de forma a ser possível executá-lo.

Para efectuar esta tarefa existem várias hipóteses. Caso exista uma ligação por rede (por exemplo, uma LAN fabril) entre o controlador do robô e o computador onde se situa o sistema de simulação e programação *off-line* do robô, o programa pode ser descarregado directamente através dessa rede. Esta será talvez a solução que se apresenta como mais fácil ao utilizador, pois não implica que ele tenha que se deslocar até ao controlador do robô. Alternativamente, e caso não exista rede, o programa pode ser descarregado por disquete (caso o controlador do robô disponha de um *drive* para disquetes, situação já bastante comum nos controladores dos robôs actuais) ou ainda através de uma qualquer porta de entrada de dados (por exemplo uma porta série, que a maioria dos controladores dos robôs actuais disponibiliza para a troca de dados com o exterior).

5 Execução do programa na planta fabril

Após a realização de todas as fases anteriores, é necessário executar o programa do robô: teoricamente, este vai correr sem sobressaltos e sem a necessidade de qualquer ajuste.

Muitas vezes, infelizmente, a realidade é bem diferente, essencialmente devido à falta de estruturação do ambiente em que o robô se insere. Por isso é quase sempre aconselhável recorrer à sensorização do ambiente da célula, como já foi referido atrás, e efectuar o tratamento e a recuperação automática de erros que eventualmente ocorram.

Se tudo isto for efectuado, as probabilidades de a execução do programa vir a apresentar problemas são extremamente reduzidas.

6 Monitorização do funcionamento da célula real

Por vezes estes pacotes de *software*, dadas as suas excelentes características gráficas, são ainda utilizados como *front-end* com o operador para a monitorização da célula robotizada real. Uma vez que os programas que os dispositivos da célula estão a executar estão disponíveis no *software* de simulação, é muito fácil escolher alguns pontos dos programas dos dispositivos de forma a que, nesses pontos, seja efectuada uma sincronização da simulação com o funcionamento dos dispositivos reais. Tais programas podem ainda ser utilizados para recolha e tratamento de dados estatísticos da célula.

7 O futuro da programação de robôs industriais

Pensamos não estar muito enganados se dissermos que o desenvolvimento da programação de robôs vai no sentido da progressiva utilização de métodos baseados em realidade virtual.

A simulação com realidade virtual começa a alcançar um tal grau de perfeição que permite aos engenheiros o "luxo" da repetitibilidade, um factor crucial para a optimização de qualquer projecto ou processo de fabrico [CiME, 1995]. Começam já a ser desenvolvidos e testados métodos de programação de robôs baseados em luvas com capacidades sensoriais [Takahashi, 1992]. Utilizando estas luvas, o programador do robô efectua a tarefa pretendida com a sua própria mão, num ambiente virtual gerado pelo computador, chamado ambiente de ensino, estando o controlador do robô a recolher informação sensorial da luva sobre a forma como a tarefa foi efectuada, de forma a poder repeti-la à posteriori (ver Figura 3.14).

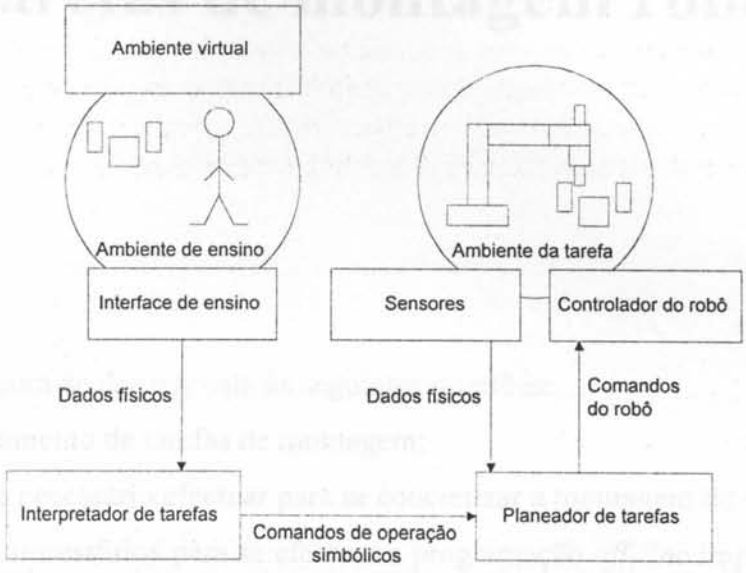


Figura 3.14: Programação por imitação [Takahashi, 1992]

O sistema reconhece os movimentos do operador a partir dos dados retornados pelo dispositivo de interface (luva com sensores) e da informação do *layout* do ambiente de ensino. Da tarefa de reconhecimento resulta uma sequência de comandos para o robô ao nível da tarefa. O sistema interpreta então esta sequência de comandos, transformando-os em comandos do robô, utilizando para esse efeito regras de interpretação dependentes da tarefa definidas à priori.

Usando os comandos interpretados do robô, este executa a tarefa no ambiente da tarefa, que não tem necessariamente que ter o mesmo *layout* do ambiente de ensino. Para ultrapassar este problema usam-se sensores que fornecem *feedback* ao planeador de tarefas, de forma a que este possa adaptar o plano que construiu a partir dos dados que recebeu do interpretador de tarefas.

Capítulo 4

Planeamento e programação de tarefas de montagem robotizadas

Neste capítulo procura-se dar resposta às seguintes questões:

- O que é o planeamento de tarefas de montagem;
- Que operações é necessário efectuar para se concretizar a montagem de um produto;
- Quais os passos necessários para se efectuar a programação *off-line* implícita de um robô a efectuar tarefas de montagem;
- Que alternativas existem para se concretizar cada um desses passos;
- Pacotes dedicados à montagem ou que apresentem funcionalidades próprias para esta tarefa.

Vimos no capítulo anterior que, de forma a que um equipamento programável realize uma tarefa, qualquer que ela seja, é necessário programar esse equipamento. Só assim ele realizará trabalho útil. Foi também visto que a estratégia de programação passa por decompor a tarefa que o robô tem que realizar sucessivamente em operações e refiná-las cada vez mais, até se obterem os comandos básicos de movimento, ou seja, a decomposição do processo completo de montagem em:

- tarefa;
- operação;
- elemento de programação.

Exemplificando para o caso concreto de uma tarefa de montagem, a divisão de uma tarefa do robô em operações, e destas nos elementos de programação correspondentes, é a ilustrada na Figura 4.1:

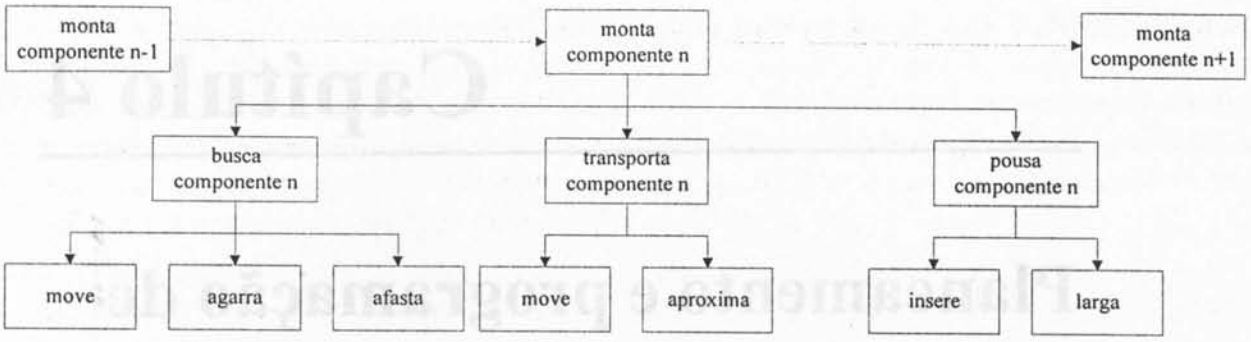


Figura 4.1: Operações e elementos de programação para uma tarefa de montagem

1 A integração do planeamento de tarefas com a sua execução

1.1 Planeamento da montagem

O planeamento pode ser definido como a determinação de um conjunto de operações necessárias e suficientes para alcançar um objectivo.

Uma montagem consiste num determinado número de subtarefas (operadores abstractos) que têm uma relação de dependência dada pela aplicação em causa. Esta relação de dependência indica a ordem pela qual duas tarefas têm que ser necessariamente executadas, função de restrições que podem ser de natureza geométrica (tal como o problema de empilhar vários componentes da montagem), que podem estar associadas às capacidades físicas do robô e do equipamento periférico necessário (e disponível), que podem ainda ser impostas pela disponibilidade de componentes ou com a sua manipulação.

Concretizando, o problema-chave no planeamento de montagem é o seguinte: encontrar um conjunto de operações de montagem/desmontagem válidas para passar do estado inicial (componentes) para o estado final (montagem completa), conhecidos o estado actual do sistema e um conjunto de operadores capazes de efectuarem a mudança de estado pretendida.

Uma representação das operações geradas e das restrições correspondentes é o chamado "plano de montagem".

Esta decomposição das tarefas em operações (planeamento da montagem) pode ser efectuada pelo programador, no caso da programação explícita, ou de forma automática (por um planeador de tarefas) no caso da programação implícita.

Em qualquer dos casos, pode-se facilmente imaginar a complexidade de determinar o plano de uma montagem grande, já que devem ser considerados todos os aspectos que dizem respeito às características da montagem, às características dos componentes, ao método de junção e ao processo de manuseamento pelo robô [Rampersad, 1994].

1.2 Automação do planeamento de tarefas de montagem

Como já foi referido no capítulo anterior, a programação tende a evoluir para uma programação implícita, tendo-se vindo a efectuar nos últimos tempos diversas tentativas de integrar a programação de equipamentos com o planeamento da tarefa que eles têm que realizar. Esta tendência também se tem verificado na programação de robôs afectos a tarefas de montagem.

Assim, no interface entre as ferramentas de planeamento de células robotizadas e de programação dos equipamentos, poderá existir uma ferramenta de planeamento do processo de montagem assistido por computador. A função desta ferramenta é compreender e transformar os dados do planeamento elaborado num formato apropriado para o processo de programação. Este método dá como resultado uma sequência de operações detalhadas, a definição das trajectórias do robô, assim como a determinação das condições lógicas e tecnológicas destas trajectórias (velocidades, aproximações aos pontos da trajectória, sinais para as garras, etc.).

Num sistema destes, e uma vez que o sistema de planeamento tem que fornecer uma descrição detalhada da tarefa de montagem para a programação dos equipamentos, é necessária a seguinte informação para que se possa proceder ao planeamento:

- componentes;
- sequência de operações;
- equipamento.

Como já referido no ponto anterior, o planeamento de montagem é uma tarefa computacional complexa porque têm que ser tomadas em consideração muitas restrições, como por exemplo a estabilidade das submontagens criadas e a exequibilidade geométrica e mecânica do processo de montagem das submontagens consideradas. Assim, apesar da automação do planeamento de tarefas de montagem, montagens mais difíceis podem necessitar de intervenção humana ou de um maior tempo de computação; as causas para as dificuldades são as seguintes:

- montagens com uma estrutura lógica muito complexa;
- montagens necessitando de técnicas especiais;
- montagens requerendo o planeamento de trajectórias muito complexas.

Por estes motivos Pezzinga (1992) argumenta que as ferramentas de planeamento de tarefas deverão ser usadas como modo de suporte à decisão, em vez de como um dispositivo que automatiza a tarefa de planeamento, já que considera poderem surgir situações nas quais as recomendações do sistema não sejam aceitáveis para o utilizador, que necessariamente as desprezará.

1.3 A integração do planeamento com a programação *off-line* de robôs de montagem

Todos estes passos podem ser integrados numa aplicação de simulação e programação *off-line* de robôs, com todas as vantagens que daí advêm. Assim, e na opinião de Pezzinga (1992), o processo de planeamento de uma tarefa de montagem, e a correspondente programação de um robô para a executar, seria realizado através dos seguintes passos:

- descrição interactiva da célula de trabalho, realizada por um operador numa estação de trabalho dotada com um interface gráfico amigável;
- especificação gráfica da tarefa de montagem a ser executada e inicialização do sistema de planeamento.

O sistema de planeamento geraria automaticamente uma sequência segura de acções/movimentos do robô necessários para realizar a tarefa. Seguidamente, traduziria estas instruções para comandos específicos e executáveis no controlador do robô e passá-los-ia para o sistema de simulação, onde visualizaria os movimentos do robô para essa tarefa de montagem específica. Portanto, a presença do robô ou da célula de trabalho não seriam necessárias na fase de programação do robô.

Um sistema integrado deste tipo requer os seguintes módulos [Pezzinga, 1992]:

- gerador do grafo de montagem;
- escalonador;
- gerador do programa e planeador do movimento;
- gerador do código;
- sistema de simulação.

Independentemente do método de programação adoptado, as operações que o robô tem que executar para concretizar as operações de montagem são as mesmas. Estas operações só variam de aplicação para aplicação (uma aplicação de montagem exige um conjunto de operações totalmente distintas de uma aplicação de soldadura por arco eléctrico, como facilmente se compreende).

Por este motivo, e antes de passarmos a ver como tem sido efectuada a integração da programação de robôs afectos a tarefas de montagem com o planeamento das mesmas, vamos ver quais são as operações de montagem geralmente necessárias para se proceder a uma tarefa de montagem robotizada.

2 Operações de montagem

As tarefas de montagem robotizada, de acordo com Bernhardt (1992), podem ser divididas nas seguintes operações:

- Transferir (*Transfer*);
- Aproximar (*Approach*);
- Agarrar (*Grasp*);

- Afastar (*Depart*);
- Largar (*Detach*);
- Inserir (*Insert*).

Por seu lado, Groover (1986) define essencialmente as mesmas operações que Bernhardt para efectuar as tarefas de montagem, dando no entanto especial relevo à operação Inserir (*Insert*).

Quanto a Ramos (1993), definiu as seguintes operações para proceder a tarefas de montagem:

- Agarrar (*Grasp_on*);
- Largar (*Ungrasp_on*);
- Agarrar lateralmente (*Grasp_by_side*).

Além destas operações, todos estes autores definem também algumas operações auxiliares de apoio de montagem. Meramente a título de exemplo, apresentamos as rotinas auxiliares que Ramos (1993) definiu para a boa execução das tarefas de montagem:

- Verificar (*Verify_if_free*);
- Desobstruir destino (*Clear_dest*);
- Aproximar (*Get_closer*);
- Identificar (*Map_unknown*);
- Agarrar por cima (*Get_from_top*).

No entanto, é Rampersad (1994) quem melhor define e organiza as operações necessárias à execução das tarefas de montagem.

Por esse facto, e também porque a solução adoptada no nosso caso foi inspirada em larga medida nesta divisão das operações de montagem, vamos passar a descrever a organização que este autor propõe para as operações de montagem.

Segundo ele a tarefa de um sistema de montagem é executar operações de montagem de forma a que a função do sistema seja cumprida, compreendendo o processo de montagem um ciclo de seis operações (ver Figura 4.2): alimentar, pegar, mover, compor, largar e mover.

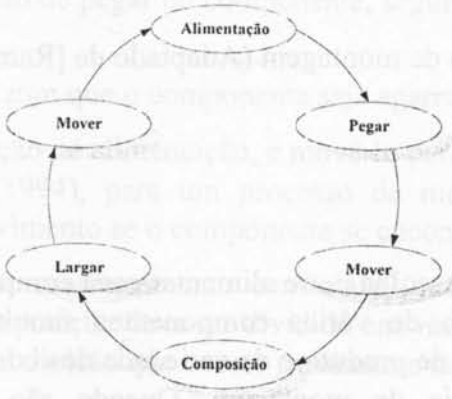


Figura 4.2: Ciclo das operações de montagem [Rampersad, 1994]

As operações de montagem foram previamente divididas em operações de alimentação (*feeding*), manuseamento (*handling*), composição (*composing*), verificação (*cheking*), ajuste (*adjusting*) e processos especiais (*special processes*), de acordo com o ilustrado na Figura 4.3.

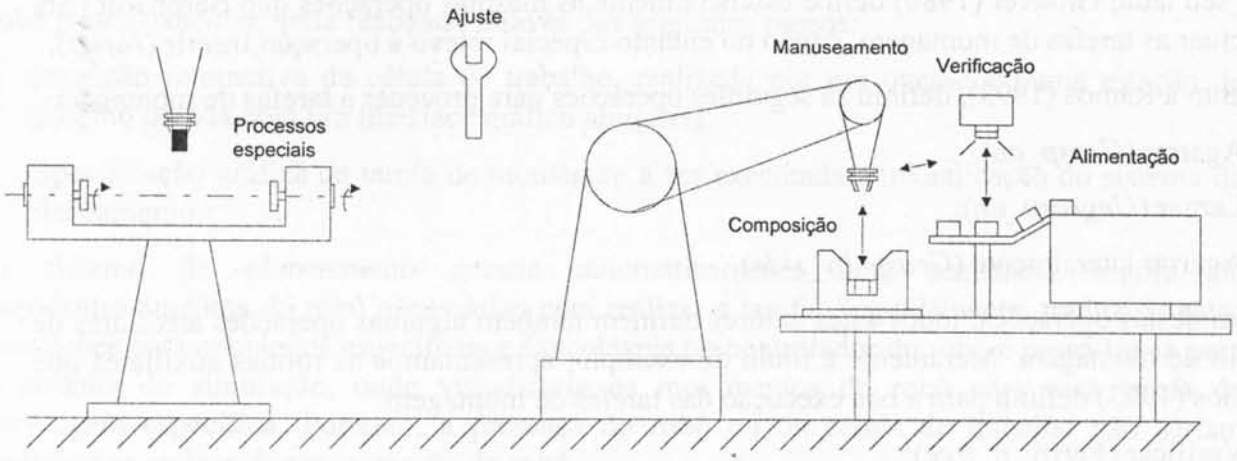


Figura 4.3: Operações de montagem num sistema de montagem robotizado [Rampersad, 1994]

Cada um destes conjuntos de operações, organizados desta forma, pode ainda ser dividido em suboperações. Essa divisão encontra-se representada na Figura 4.4.

Figura 4.4: Operações de montagem (Adaptado de [Rampersad, 1994])

2.1 Operações de alimentação

Na alimentação é efectuada uma escolha entre alimentar cada componente (elementar ou não) separadamente ou a alimentação de vários componentes simultaneamente. Esta escolha depende em larga medida do tipo de produto e da variedade dos componentes, sendo portanto efectuada ao nível da estratégia de montagem. Quando são utilizados alimentadores vibratórios e alimentadores com duplo tapete (alimentação de cada componente

separadamente) o armazenamento, separação e escolha dos componentes é efectuado automaticamente no sistema alimentador, enquanto que no caso da utilização de *paletes* (alimentação de vários componentes em simultâneo), estas operações são geralmente efectuadas manualmente fora do sistema de montagem.

Os componentes podem ser fornecidos ao robô quer numa posição de alimentação fixa (caso de um alimentador vibratório ou um alimentador de duplo tapete) ou num padrão fixo de posições de alimentação alternadas (caso da utilização de *paletes*). A alimentação de componentes numa orientação fixa (com *paletes*) também pode ser distinguida da utilização de sensores visuais para orientar os componentes.

Em função do que acabou de ser dito, o processo de alimentação encontra-se subdividido em armazenamento, separação e escolha de componentes, e no fornecimento de componentes posicionados e orientados ao robô. Isto corresponde às operações de armazenamento (*storage*), separação (*separating*), escolha (*sorting*), posicionamento (*positioning*) e orientação (*orientating*).

2.2 Operações de manuseamento

O manuseamento inclui pegar nos componentes na posição de alimentação e transferi-los para a posição de composição. Neste leque de operações incluem-se as de pegar (*picking up*), mover (*moving*) e pousar (*putting down*), esta última já relacionada com as operações de composição.

Na operação de pegar nos componentes podem ser distinguidas as seguintes fases:

- verificar (com sensores) se o componente está presente no local previsto e identificar o seu tipo;
- seleccionar a garra indicada para a operação de pegar;
- determinar a posição e orientação do componente; isto é geralmente efectuado recorrendo a sensores visuais, sendo o estado dos componentes (por exemplo, tamanhos, forma e simetria) automaticamente determinado e informado o controlador do robô que utiliza esta informação para determinar a posição correcta para pegar no componente;
- mover em direcção à posição de pegar no componente, seguindo uma trajectória óptima do movimento;
- actuação da garra, fazendo com que o componente seja agarrado;
- tirar o componente da posição de alimentação, e movê-lo para a posição de composição (de acordo com Rampersad (1994), para um processo de montagem ser fiável, deve ser verificado durante este movimento se o componente se encontra presente na garra).

Para efectuar um processo de manuseamento controlável e fiável, o movimento entre as posições de alimentação e composição deve ser dividido em várias trajectórias de movimento. A Figura 4.5 apresenta uma subdivisão que ocorre geralmente na prática.

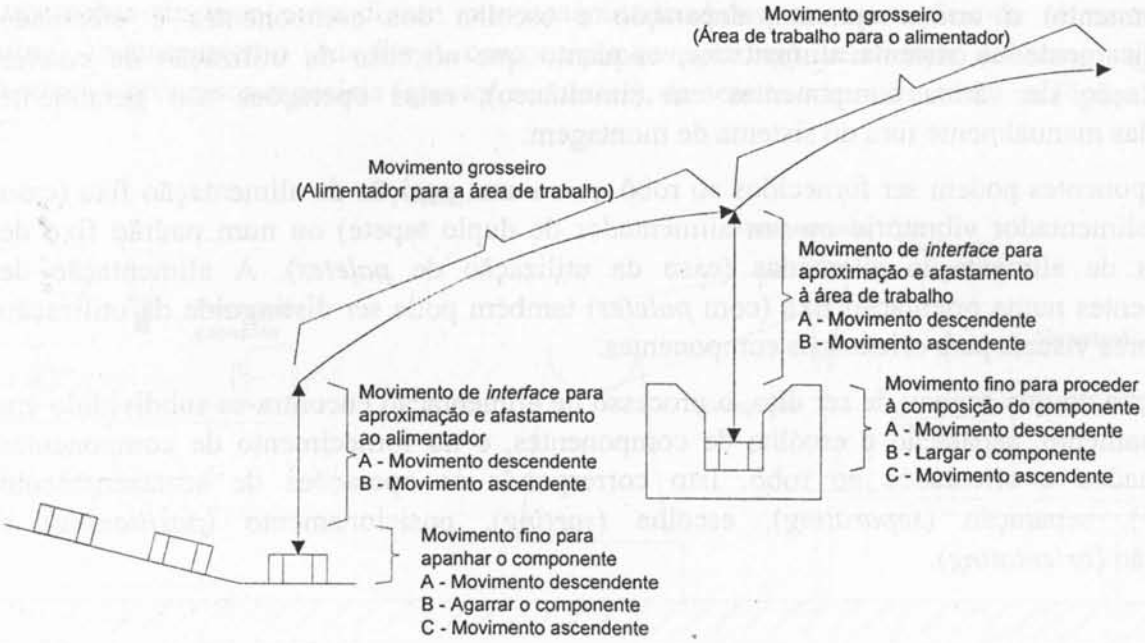


Figura 4.5: Trajectórias de movimento nas operações de manuseamento (Adaptado de [Rampersad, 1994])

Tal como é ilustrado nesta figura, a trajectória do movimento do robô encontra-se dividida em:

- movimento grosseiro: movimento rápido entre a posição de alimentação e a posição de composição, movimento este não muito preciso;
- movimento de interface: movimento rectilíneo que forma a transição para a próxima fase de movimento; durante este movimento, a posição final que se pretende alcançar com o movimento é aproximada até alguns centímetros, com a orientação e o estado da garra correctos, sendo este movimento mais lento e mais preciso do que o movimento grosseiro;
- movimento fino: consiste num movimento, por norma rectilíneo, muito preciso e lento, entre o fim do movimento de interface e a posição de alimentação ou composição; no final deste movimento é activada a garra.

Aproximadamente, um terço do tempo de ciclo de uma célula de montagem robotizada diz respeito ao movimento grosseiro e a parte restante é determinada em grande medida pelo movimento de interface e pelo movimento fino [Rampersad, 1994]. Portanto, de forma a obterem-se tempos de ciclo curtos é importante manter as distâncias de movimento tão pequenas quanto possível.

2.3 Operações de composição

As operações de composição executam as tarefas correspondentes à ligação de dois ou mais componentes, de tal maneira que o número de graus de liberdade diminua, através do estabelecimento de conexões que são mais ou menos fixas. Dentro destas, temos as operações

de inserção (*inserting*) e junção (*joining*). Estas operações estão fortemente relacionadas com a operação de pousar um componente.

A maneira como se realiza a composição é bastante influenciada pelo método de junção, sendo este determinado durante o projecto do produto, ao nível da sua estrutura.

Estas operações requerem um movimento que resulta num primeiro contacto entre os componentes, seguido da inserção ou junção propriamente dita. A sua complexidade é fortemente determinada pelos seguintes factores [Rampersad, 1994]:

- a maneira de aproximação: os componentes que devem seguir um trajecto complexo durante as operações de composição, causam geralmente problemas na montagem robotizada. Um movimento linear é portanto preferível a um movimento não linear;
- a direcção de composição: a composição vertical, de cima para baixo, é favorecida em relação às direcções laterais ou de baixo para cima (isto é ainda mais importante no caso dos robôs SCARA);
- a estabilidade do componente após ter sido pousado: o componente pode ser insuficientemente fixado após ter sido pousado, de forma que tem que ser seguro para o manter com, ou transferi-lo para, a posição e orientação correctas;
- falhas relativas nas dimensões e posições mútuas nos componentes a montar: as falhas de posição são divididas em falhas laterais e falhas angulares. Ambos os tipos de falhas podem causar encravamentos no processo de montagem. Este problema é resolvido através de um alinhamento correcto do componente a inserir com o buraco onde vai ser inserido (por exemplo, equipando o componente e o buraco com um chanfro) e segurando no componente com uma certa flexibilidade no plano horizontal para facilitar o alinhamento coaxial durante a montagem.

2.4 Operações de Ajuste

O ajuste envolve a realização de certas operações de ajuste na célula de montagem. Exemplos típicos são a operação calibrar (*calibrate*), cuja finalidade é efectuar a calibração de componentes do sistema, e a operação alinhar (*align*), que tem por finalidade a realização do ajuste de montagens compostas.

2.5 Processos Especiais

Estes referem-se a processos específicos que podem ser levados à prática na célula de montagem. Incluem-se aqui as operações de maquinagem (*machining*), lavagem (*washing*), impressão (*printing*), desmontagem (*disassembling*) e embalagem (*packing*).

Por vezes, alguns destes processos podem ser realizados antes ou depois da composição, nos componentes individuais ou na montagem completa.

2.6 Operações de verificação

A verificação, que inclui inspecções e testes, envolve a determinação da qualidade do processo de montagem realizado, através da execução de uma medida ou de um teste, por exemplo visual. Através da verificação da qualidade da montagem imediatamente após a operação de composição, as composições erradas não viajam para além do sistema de montagem. Além destas situações, é muitas vezes necessário verificar o funcionamento da montagem (através de um teste funcional) antes que ela abandone a célula de montagem.

Dentro das operações de verificação incluem-se as operações de medição (*measuring*), inspecção (*inspection*) e teste (*testing*).

A utilização de sensores é central para as operações automáticas de verificação de *performance*, tais como a verificação de presença através de um simples sensor indutivo ou sensores ópticos, e a determinação da posição e orientação dos componentes, assim como da sua identificação através de um sistema de visão artificial.

3 O planeamento automático de tarefas de montagem

3.1 Estratégias de planeamento e programação de tarefas de montagem

Como vimos, o problema que se pretende resolver com o planeamento da montagem é o da geração de um plano a alto nível, capaz de sequenciar um conjunto de operações de modo a transitar de um estado inicial para o estado que se pretende atingir (estado final).

De acordo com Ramos (1993), existem duas formas de proceder à geração do plano: o planeamento *off-line* e o planeamento *on-line*.

No primeiro caso é pretendida a geração de um plano que não necessitará de ser executado imediatamente. O planeamento *off-line* é mais aconselhável para situações industriais em ambientes muito estruturados, onde o que se pretende é gerar um plano que será executado repetidas vezes (será até talvez o único plano que um dado robô irá executar ao longo de toda a sua vida). Sendo assim, o principal requisito é a geração de um bom plano, se possível o melhor plano (normalmente aquele que permite executar a tarefa em menos tempo). A introdução de um gerador de planos capaz de fornecer uma boa solução (ou a melhor solução) é importante, pois essa solução tornará o processo de montagem (e de produção) mais eficiente. Outra característica do planeamento *off-line* é o facto de ter disponível, antes de se executar o planeamento e a execução do plano, toda a informação necessária para que a geração do plano tenha lugar.

Pelo contrário, no planeamento *on-line* o plano gerado necessita de ser executado logo de seguida, sendo uma das principais preocupações a minimização da soma do tempo necessário para gerar um plano e para executá-lo. Poder-se-á dizer que o objectivo a alcançar é um bom compromisso entre o tempo de geração do plano e o custo do plano gerado, não sendo fundamental gerar o melhor plano segundo os mesmos critérios do planeamento *off-line*. Se nos restringirmos ao processo de planeamento a alto nível, propriamente dito, uma boa

aproximação será a geração rápida de um plano procurando garantir que este não tenha um custo muito elevado do ponto de vista de execução. Este tipo de planeamento adapta-se melhor às situações nas quais não se espera que os estados iniciais e finais se repitam com grande frequência, sendo por natureza um planeamento que está mais adaptado a ambientes menos estruturados.

3.2 Informação necessária ao planeamento de tarefas de montagem

A estrutura do planeador de montagem depende do nível de definição pretendido para a sequência a obter. No caso mais simples, a sequência de montagem é um conjunto de ordens de manipulação, enquanto que, em casos mais complexos, é uma sequência ordenada de operações de variados tipos. O tipo de sequência gerada depende, em grande medida, da quantidade e do tipo de informação disponíveis para a tarefa de planeamento.

Numa visão moderna e a alto nível dos processos de fabrico podemos constatar que, no centro, se encontram representações ou modelos dos produtos, dos processos e dos equipamentos. Em redor deste núcleo central encontram-se as actividades responsáveis pela criação e utilização destes modelos. As representações a alto nível dos processos são criadas através de actividades, usualmente chamadas planeamento e escalonamento. Os planeadores raciocinam acerca da geometria (e outras propriedades) dos objectos, tentando encaixá-las nas capacidades dos processos físicos, resultando daí a geração de uma sequência de operações de produção. Os escalonadores raciocinam acerca dos processos e recursos, alocando operações ao equipamento apropriado e especificando a informação temporal associada.

O planeamento da montagem começa tradicionalmente com uma descrição geométrica dos componentes individuais e da sua posição e orientação no produto montado. São também necessários dados respeitantes às operações de montagem possíveis e informação relativa aos equipamentos que vão ser utilizados para a execução do plano gerado (por exemplo, a descrição dos dedos das garras). O resultado deverá ser um escalonamento com todas as instruções para os robôs e outros dispositivos na célula.

3.2.1 Representação da informação dos componentes da montagem

De acordo com Mäntylä (1996), a modelização do produto pode ser caracterizada como uma metodologia unificada para capturar e representar a informação relativa ao produto, tornando-a disponível para vários processos de projecto e produção. A criação de uma representação de um produto é designada por projecto. Na prática, o projecto de um componente mecânico ou montagem é especificado pelo seu modelo geométrico e por um conjunto de propriedades não geométricas (por exemplo, o seu material).

Considerando as diferentes funções no planeamento da montagem, a maioria dos problemas ocorre com a interrogação e interpretação do modelo dos componentes para questões como: o que é uma boa sequência de montagem para um produto, como pode um componente ser agarrado com uma garra, qual é o trajecto de aproximação e inserção adequado para adicionar um componente a uma submontagem?

É difícil obter respostas a estas questões usando unicamente um modelo geométrico, porque muita informação necessária para isto (por exemplo, relações entre componentes, tolerâncias e eixos de simetria) não é representada no modelo geométrico e também não pode ser facilmente computada a partir dele [van Holland, 1995].

Para ultrapassar este problema, Gutsche (1995) propõe um sistema de planeamento de montagem que trabalha com relações espaciais simbólicas. Este sistema suporta componentes rígidos, modelizados em dois níveis: a um nível abstracto, os componentes são representados por objectos sem corpo, consistindo num conjunto de referenciais matemáticos; as relações espaciais (sob a forma de sistemas de coordenadas de referência) são utilizadas para definir um grafo de montagem e para deduzir automaticamente as transformações homogéneas entre os componentes da montagem, a partir da informação simbólica. Num outro nível, os componentes são representados geometricamente como poliedros e, opcionalmente, por modelos de CSGs.

Por sua vez, van Holland (1995) afirma ser preferível a utilização de modelos de características (*features*), ao invés de modelos geométricos, uma vez que os primeiros podem ser de grande ajuda no processo de planeamento de montagem. O mesmo é defendido por Mäntylä (1996).

Uma característica é uma forma paramétrica que tem associado um conjunto de atributos como sejam os seus parâmetros geométricos intrínsecos - comprimento, largura e altura ou profundidade - bem como posição, orientação, tolerâncias geométricas, propriedades do material e referências a outras características.

No projecto, as características capturam atributos explícitos de engenharia e relações entre entidades de definição dos produtos - informação essencial para várias tarefas de projecto e de análise.

Na produção, as características podem ser ligadas a vários tipos de conhecimentos sobre a produção. Este conhecimento facilita o processo de planeamento e ajuda na geração das instruções detalhadas das operações requeridas pelos robôs [Mäntylä, 1996].

De acordo com este autor, as características são potencialmente atractivas para as tarefas de planeamento de montagem, uma vez que um modelo de características pode armazenar a disposição física dos componentes numa montagem, bem como informação respeitante aos atributos das ligações físicas entre os componentes ligados. Estes dados são fundamentais para a execução das operações de montagem necessárias e para o seu sequenciamento.

Concretamente para o planeamento de tarefas de montagem, van Holland (1995) divide as características em dois tipos: características de montagem (*connection features*) e características de manipulação (*handling features*). As primeiras estão relacionadas com a maneira como os componentes da montagem devem ser montados e, as segundas, estão relacionadas com a forma como devem ser agarrados e fixados. Com estas duas classes de características podem ser definidos dois modelos de características diferentes para cada produto. Cada modelo representa um diferente aspecto ou vista sobre o produto. Estas múltiplas vistas têm que ser mantidas simultaneamente e têm que ser consistentes.

Existem duas aproximações possíveis para a criação de modelos de características [Mäntylä, 1996]: o projecto por características, no qual o responsável pelo projecto desenvolve o

produto directamente como uma combinação de características, e o reconhecimento de características, que envolve a procura de características num modelo geométrico do produto.

No entanto o reconhecimento automático de características ainda é um problema [Requicha, 1996]. Uma aproximação possível consiste na definição das características como um padrão de faces e arestas estruturadas sob a forma de um grafo, efectuando depois uma procura destes padrões nas fronteiras dos objectos. Esta procura envolve a detecção de isomorfismos em subgrafos. Esta aproximação à procura em grafos apresenta algumas dificuldades relacionadas com características que interagem volumetricamente, uma vez que as interacções podem danificar os padrões, tornando impossível a sua identificação. Outra aproximação passa pelos métodos baseados em volumes, dois dos quais são apresentados por Mäntylä (1996).

Apesar da representação dos produtos baseada em características parecer promissora, apresenta ainda vários problemas, sendo o maior destes o problema de decidir quais as formas geométricas a utilizar como características básicas para uma dada operação ou processo. Para ajudar neste processo, Mäntylä (1996) apresenta uma metodologia de seis passos para a identificação das características mais adequadas em função da utilização para elas pretendida.

3.2.2 Representação da informação relativa ao processo de montagem

Além da informação respeitante ao produto a montar e aos seus componentes, é também necessária a informação relativa ao processo de montagem a ser utilizado, caso se pretenda gerar os programas para os equipamentos que vão executar a tarefa de montagem.

Assim, é necessária informação relativa ao conjunto de operações de montagem que poderão ser utilizadas para cumprir o plano de montagem. Estas, como já foi visto, deveriam incluir entre outras operações de montagem, manuseamento, ajuste dos componentes, identificação das submontagens.

É também necessária informação específica ao equipamento a utilizar. De forma a obter uma definição completa das operações é necessário o conhecimento acerca das ferramentas disponíveis, do robô, dos alimentadores e da célula física. Além do mais, a informação acerca da célula de trabalho é também necessária para a fase de planeamento da manipulação e de planeamento de trajectórias.

Segundo Caracciolo (1995) a descrição da célula em termos de sistemas de coordenadas de referência permite uma fácil integração do planeador de montagem com o planeador de trajectórias. As trajectórias podem ser descritas como conjuntos de sistemas de coordenadas de referência, representando as posições discretas do actuador final do robô. Desta forma, pode-se definir uma operação de agarrar nos termos de uma matriz, relacionando a ferramenta com o objecto a manipular. Logo, a saída do planeador de montagem pode ser directamente usada pelo planeador de trajectórias.

A informação geométrica acerca da montagem, em conjunto com a informação acerca da célula de trabalho e do robô, permitem obter de forma automática um programa ao nível de tarefa para a montagem, onde não só é especificada a ordem de montagem dos componentes, mas também as operações a serem realizadas.

4 Passos para o planeamento automático da montagem

Rocha (1995) divide o planeamento e execução de tarefas em duas fases: a primeira fase é a geração e representação automática de um plano de alto nível composto por operadores simbólicos (por exemplo, inserir A em B, juntar C com D). O planeamento de alto nível permite gerar um plano para alcançar o objectivo, considerando todas as restrições e dados o estado inicial e o objectivo da tarefa. É também possível gerar não apenas um, mas vários planos que podem ser activados dinamicamente, de acordo com a situação corrente.

De forma a ser possível definir os planos de montagem, é necessário e essencial a existência de um método para reunir toda a informação respeitante à sequência de operações de montagem.

No livro editado por Homem de Mello (1991), é possível encontrar uma diversidade de soluções para a representação de planos de montagem. As aproximações mais simples apenas têm em linha de conta a ligação entre componentes e fornecem uma informação gráfica das possíveis sequências de montagem. As técnicas de representação mais elaboradas relacionam as características dos componentes, tais como o tipo de superfície ou propriedades de simetria, com a sua ligação e com as trajectórias dos robôs durante o processo de montagem [Gutsche, 1995][van Holland, 1995].

Os estudos que têm sido realizados com o objectivo de definir e classificar planos de montagem agrupam-nos de acordo com três representações distintas [Caracciolo, 1995]:

- sequência de estados: representando planos de montagem como sequências de operações;
- árvore de montagem parcial: o plano de montagem é visto como uma decomposição recursiva em subconjuntos (a árvore E/OU é um exemplo);
- árvore de submontagens: representando os planos de montagem como sequências de operações, inserindo componentes ou submontagens numa fixação.

A informação mais importante produzida pelo sistema de planeamento é a descrição formal da tarefa a ser executada. Usando esta informação pode ser gerada uma representação explícita da solução (por exemplo, o código de um programa que um robô deve executar para cumprir o plano de montagem). Esta fase, a geração dos programas para os robôs, é a segunda fase em que Rocha (1995) dividiu o planeamento de tarefas de montagem.

Após se ter decomposto a tarefa numa sequência de operações, tem que se iniciar uma fase de planeamento de execução das operações, que engloba a determinação dos parâmetros detalhados destas, tais como coordenadas dos trajectos do robô e condições tecnológicas e lógicas (por exemplo, a especificação da velocidade ou a transmissão de sinais). Esta tarefa pode ser realizada por alguns módulos especializados, tais como um planeador de manipulação e um planeador de trajectórias, podendo este ser ainda dividido em planeador de movimentos finos e em planeador de movimentos grosseiros (aspecto já abordado no capítulo anterior).

Tanto Hörmann (1992) como van Holland (1995) apresentam a mesma opinião de Rocha (1995), tendo sistematizado o planeamento automático de tarefas de montagem como sendo constituído pelas seguintes funções:

- decomposição em submontagens;
- geração das sequências de montagem;
- escalonamento;
- planeamento da manipulação;
- planeamento do movimento.

Vamos de seguida passar a ver em que é que consiste cada um destes passos e como é que podem ser concretizados, ignorando contudo o aspecto do planeamento do movimento (já abordado no capítulo anterior).

4.1 Decomposição em submontagens

Na maioria dos casos não é de grande utilidade gerar todas as possíveis sequências de como um produto pode ser montado a partir de componentes individuais; de acordo com van Holland (1995), existem várias razões que apontam para a divisão da montagem em submontagens. Por exemplo, para se efectuar o sequenciamento das operações e o seu escalonamento é mais conveniente começar por montar componentes que apresentem as ligações mais complicadas entre eles, formando submontagens, e combinar depois estas submontagens de forma a dar origem ao produto final.

As submontagens podem ser determinadas quer automaticamente, que à mão, tendo por base uma decomposição funcional (componentes eléctricos vs. componentes hidráulicos, por exemplo), ou tendo por base as propriedades da conexão que descrevem como é que os componentes interagem e são ligados uns aos outros (soldados, aparafusados, colados, etc.).

Os componentes que se encontram fortemente ligados são candidatos a constituírem submontagens. Outras condições são a estabilidade geométrica e a uniformidade na direcção de montagem e tipo de ligação, que podem dar origem à distinção de grupos de componentes no produto. Também a disponibilidade de uma base adequada no componente, que possa ser fixada numa fixação disponível, é um importante factor para uma submontagem.

Uma estrutura de dados útil para representar as ligações entre componentes de uma montagem é o grafo de relação, também conhecido por grafo de conexões, contacto ou ligação que representa a forma como os vários componentes de uma montagem se encontram ligados. A decomposição em submontagens pode ser representada utilizando um grafo hierárquico.

4.2 Geração das sequências de montagem

Dada a decomposição em submontagens, podem ser geradas as sequências de montagem possíveis dentro de cada submontagem (e para toda a montagem). Mas estas submontagens não podem ser realizadas por qualquer ordem. Para a execução correcta de uma tarefa de montagem têm que se tomar em consideração várias regras de acordo com leis físicas. Geralmente, a execução de certas tarefas é impossível devido à existência de restrições. Rocha

(1995) refere que as principais se encontram relacionadas com as restrições do processo, as restrições de execução e as restrições geométricas.

De forma a evitar a explosão combinacional, os componentes podem ser agrupados em conjuntos que não apresentam uma ordem de montagem preferencial. Num conjunto de parafusos, por exemplo, estes podem muitas vezes ser montados sem que exista uma ordem preferencial na sua montagem.

Pode ser introduzida outra redução no número de sequências possíveis considerando a exequibilidade geométrica. A maioria das configurações geométricas impõem uma ordem na montagem dos seus componentes. Estas relações de precedência podem ser estabelecidas deixando que o utilizador imponha uma dada ordem, ou através da realização de testes de exequibilidade geométrica, quer aplicados localmente, quer globalmente. As restrições de precedência podem ter várias causas que resultam quer da própria tarefa, quer dos recursos (como o robô, a garra, etc.) que foram seleccionados para executar a tarefa. Se, por exemplo, existe um componente que deve ser aparafusado a outro, antes de se poder proceder ao aparafusamento, os dois componentes têm que se encontrar na posição final.

Assumindo que não estão presentes nenhumas forças internas ou externas no produto montado, o inverso de uma sequência de desmontagem válida é uma sequência de montagem válida. Portanto, as sequências de montagem válidas podem ser determinadas por uma aproximação de caminho inverso. Neste caso, o problema de gerar sequências de montagem é transformado no problema de gerar sequências de desmontagem. Esta transformação leva a uma aproximação de decomposição na qual o problema de desmontar uma montagem é decomposto em subproblemas distintos, cada um deles sendo a desmontagem das submontagens criadas.

As sequências de montagem exequíveis resultantes podem ser representadas utilizando um grafo E/OU, em que os nós E representam a combinação de dois componentes e os nós OU representam sequências alternativas para a mesma montagem.

4.3 Escalonamento

Cada uma das sequências de montagem encontradas no passo anterior oferece uma alternativa de planeamento. No entanto, cada alternativa apresenta um custo diferente em relação às mudanças de garras, transporte, montagem e requisitos das fixações, e difere na flexibilidade respeitante à recuperação de erros. Os custos são geralmente estimados com base nas estimativas do tempo necessário para a realização dos movimentos de montagem de componentes individuais, trocas de garras, etc.

O escalonamento é necessário para determinar uma sequência de operações particular para a execução do plano de montagem. A tarefa de escalonamento consiste da escolha de um trajecto, por exemplo através de um grafo E/OU, optimizando esse trajecto em relação aos custos, recursos disponíveis e flexibilidade. Muitas vezes não é atractivo montar um único produto de cada vez, sendo preferível montar um lote de produtos de forma a minimizar o número de mudanças de ferramentas e das actividades de transporte.

Dependendo das restrições temporais e do próprio escalonamento, podem ser utilizados muitos critérios de optimização para gerar esta sequência, cada um dependendo das características da aplicação do robô. Ramos (1993) apresenta várias heurísticas para a realização desta tarefa.

Os grafos de precedência são um método conveniente para representar uma ordem válida de operações de montagem, assim como para representar o fluxo de operações de montagem. Os nós destes grafos descrevem as várias operações de montagem que são necessárias para executar uma tarefa de montagem especificada. As arestas definem a ordem cronológica das operações. Elas expressam a relação "antes", o que significa que uma operação só pode ser realizada se outra operação estiver completa.

Chegados aqui, após se ter decomposto a tarefa numa sequência de acções, tem que se iniciar uma fase de planeamento de execução das acções, que deve ser realizada por alguns módulos especializados, tais como o planeador de manipulação e o planeador de trajectórias (planeador de movimento finos e planeador de movimentos grosseiros).

O objectivo destas últimas funções do planeamento é a geração de instruções para os robôs.

4.4 Planeamento da manipulação

Um processo de manipulação, de acordo com Reinhart (1995), consiste em oito sequências (ver Figura 4.6).

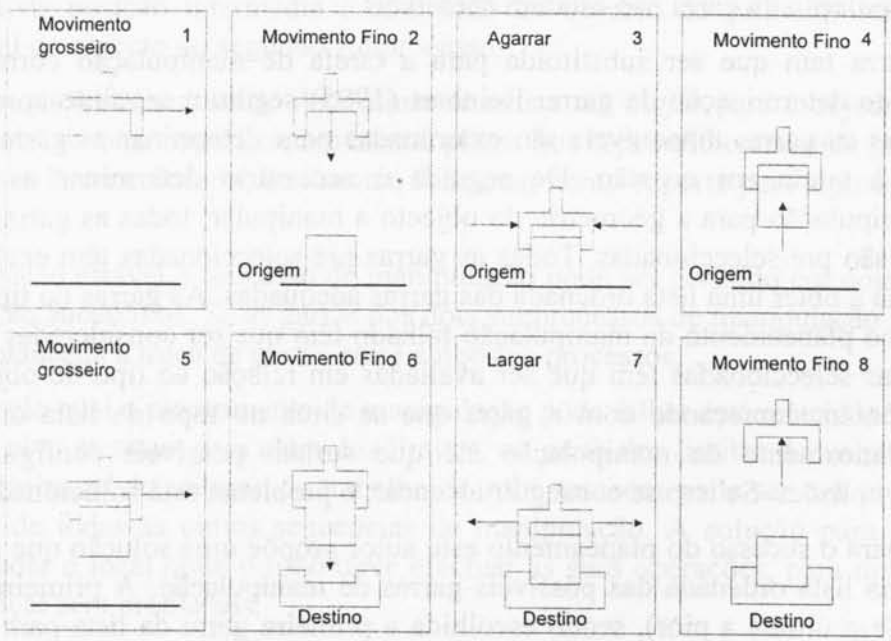


Figura 4.6: Sequência de um processo de manipulação [Reinhart, 1995]

O planeamento da manipulação trata do seguinte problema: como é que um componente pode ser agarrado pela garra do robô (ou com que garra do robô) de tal forma que possa ser inserido na sua posição final de montagem?

Reinhart (1995) dividiu este problema em dois subproblemas: o problema da escolha da garra para agarrar o objecto na sua posição de origem (*gripper problem*) e o problema de largar o objecto na sua posição destino (*release problem*). A estes há que acrescentar o problema da determinação do ponto de manipulação.

4.4.1 Determinação do ponto de manipulação

O processo de planeamento da manipulação começa normalmente com a determinação do ponto de manipulação de cada componente da montagem e da montagem completa. Isto pode ser efectuado manualmente pelo programador do robô, ou de forma automática recorrendo a um planeador automático de manipulação [van Holland, 1995] [Hörmann, 1992].

4.4.2 Escolha da garra para a tarefa de manipulação

Após ter sido determinado o ponto de manipulação, o robô deve tentar apanhar o objecto. O planeamento da manipulação pode no entanto falhar, devido a uma garra inadequada. A causa pode, por exemplo, ficar a dever-se ao facto de a garra ser demasiado grande em relação ao espaço disponível na área de agarrar ou ao facto de os lados do objecto que se adequam ao princípio de manipulação da garra não estarem acessíveis.

Neste caso, a garra tem que ser substituída para a tarefa de manipulação corrente. Para efectuar a tarefa de determinação da garra, Reinhart (1995) seguiu a seguinte aproximação: inicialmente, todas as garras disponíveis são examinadas para determinar as garras que são mais adequadas à tarefa em questão. De seguida é necessário determinar os possíveis princípios de manipulação para a geometria do objecto a manipular; todas as garras com um destes princípios são pré-seleccionadas. Todas as garras pré-seleccionadas têm então que ser avaliadas de forma a obter uma lista ordenada das garras adequadas. As garras do tipo das que foram utilizadas no planeamento da manipulação falhado têm que ser consideradas as piores. As restantes garras seleccionadas têm que ser avaliadas em relação ao tipo de objectos que podem agarrar. Então, começando com a garra que se situa no topo da lista ordenada, é efectuado um planeamento da manipulação até que surjam possíveis configurações de manipulação sem colisões. Se isso se conseguir alcançar, o problema está solucionado.

Conclui-se que, para o sucesso do planeamento este autor propõe uma solução que passa pela construção de uma lista ordenada das possíveis garras de manipulação. A primeira garra da lista é a melhor (e a última a pior), sendo escolhida a primeira garra da lista para realizar a tarefa de manipulação.

4.4.3 Largar o objecto no seu destino

O passo seguinte do planeamento do processo de manipulação é a determinação da posição do robô para largar o objecto que foi manipulado. A posição do robô onde o objecto é largado (Figura 4.6 - passo 7) resulta directamente da posição por onde o objecto foi agarrado e do seu destino.

Conhecendo a posição relativa da garra em relação ao objecto (a garra escolhida para manipular o objecto define a sua posição em relação a este) e o destino do objecto, a posição necessária da garra e, finalmente, a posição do robô para largar o objecto podem ser determinadas. As posições da garra e do robô têm que ser verificadas para detectar eventuais colisões com outros objectos do ambiente ou vizinhança. Caso seja detectada uma colisão, as posições-alvo têm que ser determinadas para a próxima garra da lista ordenada, determinada no ponto anterior, e assim sucessivamente.

É aqui que se poderá verificar o outro problema referido por Reinhart (1995): o problema de largar o objecto. Este problema inicia-se quando nenhuma, de todas as possíveis configurações da garra utilizada para manipular o objecto, permitem que este seja largado sem que ocorram colisões.

Este problema pode ser resolvido com outra posição da garra, ou mesmo com outra garra. O planeamento falha se nenhuma das garras determinada pelo algoritmo de planeamento de manipulação permitir largar o objecto sem que ocorra uma colisão.

Caso isto ocorra, Reinhart (1995) sugere que o processo de manipulação seja efectuado através da sua divisão em dois subprocessos de manipulação sucessivos, com uma posição intermédia. Esta posição intermédia é o destino do objecto do primeiro subprocesso e a posição inicial do objecto do segundo subprocesso.

Neste caso, passa a ser necessário determinar se existe um *buffer* para o objecto na célula de trabalho do robô ou uma superfície horizontal plana, que seja suficientemente grande para que o objecto possa lá ser colocado, de forma a funcionar como suporte para a posição intermédia do objecto.

Com esta posição estável o processo de manipulação pode ser dividido em dois subprocessos de manipulação sucessivos. Se as garras dos dois subprocessos de manipulação são diferentes, tem que ser planeada a troca de garras entre ambos os processos.

Outra razão pela qual o planeamento de manipulação pode falhar está relacionado com o facto de a posição-alvo se situar para além dos limites, ou perto dos limites do volume de trabalho do robô. A limitação do volume de trabalho do robô também pode ser um problema para o planeamento de todas as outras sequências de manipulação. A solução para este problema passa por mudar o local onde o robô deve efectuar as suas operações, para um local que ele consiga alcançar sem problemas.

4.5 Métodos de planeamento de montagem

Vamos agora passar a ver algumas abordagens que têm sido seguidas para se efectuar o planeamento de tarefas de montagem. Durante os últimos anos foram desenvolvidas várias

abordagens para solucionar este problema. Alguns dos sistemas de planeamento desenvolvidos não consideravam a existência de um plano de execução, ou seja, como programar os robôs (ou mais genericamente máquinas) para executarem o plano simbólico de alto nível gerado.

Contudo, outros sistemas já tratam esse problema. Quer Ramos (1993), quer Homem de Mello (1991) apresentam vários sistemas de planeamento de montagem inseridos nestes dois tipos. Gutsche (1995) divide a geração automática de planos de montagem em duas categorias diferentes.

Uma delas compreende os sistemas com interacção do utilizador para aquisição do conhecimento acerca de uma montagem (também conhecidos como sistemas de pergunta/resposta). Para montagens simples com um pequeno número de ligações, esta aproximação tem a vantagem de explorar o conhecimento intuitivo que um perito humano tem das relações dos componentes e da exequibilidade das operações, quando este responde a perguntas; a avaliação das respostas evita computações exaustivas e complexas para determinar todas as sequências de montagem através de raciocínio geométrico. Mas para montagens complexas pode ser difícil garantir a correcção das respostas ou ser fastidioso responder ao número total de perguntas (que pode aumentar exponencialmente). Um exemplo de um sistema deste tipo é o descrito por Bourjault (1984).

Por outro lado, podem ser construídos sistemas sem interacção do utilizador, tal como o sistema descrito por Homem de Mello (1991a). Neste caso, para gerar automaticamente todas as sequências de montagem válidas, é necessária uma representação precisa da montagem não sendo suficiente para este efeito a simples informação geométrica dos componentes a montar. Ele utiliza também um modelo relacional para a representação da montagem, que incorpora informação sobre a geometria dos contactos e das conexões.

Não vamos ser muito exaustivos nesta descrição, dando antes realce aos métodos de planeamento de montagem que têm sido aplicados em estreita colaboração com sistemas de CAD e em aplicações cujo objectivo é o planeamento e a programação *off-line* de robôs de montagem: uma vez que os sistemas de CAD são cada vez mais utilizados em empresas industriais para suportarem a actividade de projecto, a definição automática dos planos de montagem deverá começar pelo modelo dos componentes existentes no CAD.

No planeamento com base em modelos de CAD podem ser distinguidas duas aproximações [van Holland, 1995]: o planeamento genérico e o planeamento direccionado ao objectivo.

O planeamento genérico extrai toda a informação para o planeamento da montagem do modelo geométrico. Isto diz respeito à informação acerca da maneira como os componentes são conectados, graus de liberdade dos componentes, possível interferência dos componentes durante a montagem, como os componentes podem ser manipulados, etc. Um problema desta aproximação é que para responder a estas questões de forma óptima o modelo geométrico descreve o produto num nível demasiado baixo. Neste caso, o procedimento de combinar várias restrições é bastante simples. Mais difícil é a dedução automática do conjunto de restrições a partir de um modelo de CAD.

A alternativa, o planeamento direccionado ao objectivo, aplica mais uma aproximação *top-down*: o produto é inicialmente descrito a um nível funcional, com informação sobre as

principais intenções do projecto (como o produto é composto, como os componentes são ligados e como podem ou devem ser montados). O modelo geométrico é aqui usado unicamente para responder a questões detalhadas que surgem durante o planeamento.

A primeira aproximação foi a seguida por Lee (1992). O sistema descrito por este autor aceita como entrada a descrição de um produto efectuada por um sistema de CAD e gera um plano de montagem sujeito às restrições impostas pelos recursos existentes na célula de montagem. Existe uma base de dados do "mundo", representando os componentes, as relações entre eles, operações industriais e as máquinas e ferramentas da célula de montagem. O gerador de planos efectua inferências sobre um conjunto de regras de produção, do tipo "SE <pré-condição> ENTÃO <acções>", em que as acções correspondem à criação de uma lista de junção e de uma lista de remoção. O plano é obtido pela inversão da sequência de desmontagem.

Uma aproximação ao planeamento orientado ao objectivo é a especificação das propriedades de uma montagem através da utilização de características (*features*). Esta é a aproximação defendida por van Holland (1995). As características ligam conotações funcionais aos elementos geométricos podendo aquelas ser então directamente interpretadas pelo *software* de planeamento da montagem sem necessitarem de ser inferidas a partir da geometria.

5 Pacotes de software de simulação adaptados a tarefas de montagem

Comercialmente ainda não existem pacotes dedicados exclusivamente a tarefas de montagem ou cujo objectivo seja primordialmente esse. Mesmo assim existem alguns pacotes de *software* para simulação e programação *off-line* que já apresentam algumas características de planeamento com vista à execução de tarefas de montagem (caso do CimStation).

Onde estas aproximações têm tido algum desenvolvimento é no seguimento de projectos de I & D.

Um exemplo do que acaba de ser dito está a ser levado à prática no âmbito do Progetto Finalizzato ROBOTICA. Este projecto, que está a ser desenvolvido pelo CNR (*Consiglio Nazionale delle Ricerche* - organismo Italiano que tutela a I&D nesse país), tem como objectivo a realização de um sistema integrado para a programação *off-line* de robôs de montagem.

O sistema, capaz de trabalhar com vários sistemas de CAD, através de um interpretador de características de forma geométrica determina quais os componentes que têm que ser montados. Um planeador de montagem determina então a sequência de operações necessárias para efectuar uma montagem, conhecendo as relações geométricas de precedência e o contacto entre os componentes da montagem. Após isto, a célula é modelizada num ambiente visual fornecido por um simulador de robôs, de forma a efectuar uma simulação *off-line* do conjunto de trajectórias a serem seguidas pelo robô para realizar a tarefa de montagem, sem que ocorram colisões entre o manipulador e os objectos da célula. Este módulo gera assim, de uma forma totalmente automática, uma sequência de instruções que podem ser imediatamente executadas pelo robô.

Também o Projecto ESPRIT 623 - *Operational Control for Robot System Integration Into CIM, Systems Planning, Implicit and Explicit Programming* - que teve por tema, tal como o seu nome indica, a integração de robôs em ambientes fabris, desenvolveu uma aproximação deste tipo.

Este projecto iniciou-se no final da década de 80 e o seu objectivo principal foi o desenvolvimento de ferramentas que auxiliassem a implementação de células robotizadas, desde o seu projecto até à programação dos equipamentos instalados na célula. Para esse efeito utilizou-se uma aproximação baseada em ferramentas de simulação e programação *off-line* de robôs, auxiliadas por um sistema de planeamento de operações de robôs [Bernhardt, 1992]. Concretamente no que concerne à programação dos equipamentos, adoptou-se uma arquitectura modular que permite utilizar quer a programação explícita, quer a programação implícita dos equipamentos envolvidos.

Um resultado prático deste projecto foi o sistema IMPRES (*IMplicit PRogramming Expert System*), descrito por Pezzinga (1992), cuja arquitectura (modular) é apresentada na Figura 4.7. O objectivo deste sistema é diminuir o tempo de programação de células robotizadas para tarefas de montagem, permitindo executar a programação *off-line* através de capacidades avançadas de planeamento e escalonamento.

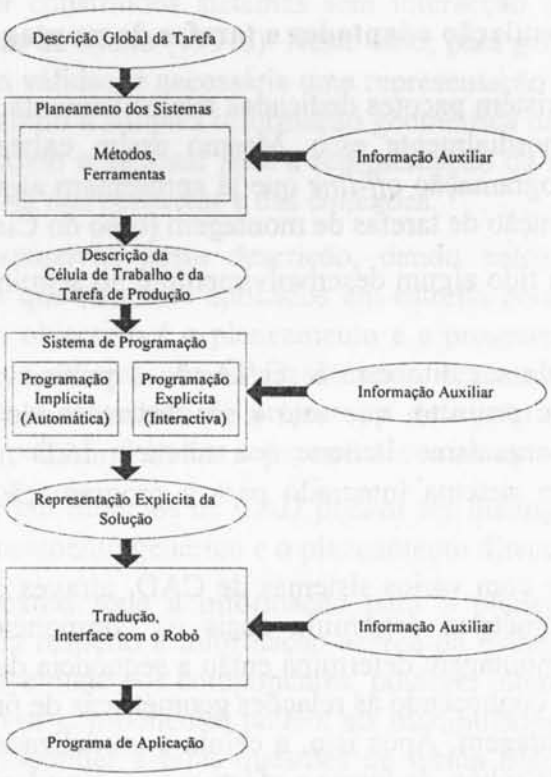


Figura 4.7: Principais interfaces de informação entre subsistemas do sistema IMPRES [Pezzinga, 1992]

Nesta aplicação o processo de planeamento da tarefa é realizado através dos seguintes passos:

- descrição interactiva da célula de trabalho, realizada por um operador numa estação de trabalho, dotada com um interface gráfico amigável;
- especificação gráfica da tarefa de montagem a ser executada e inicialização do sistema de planeamento.

O sistema recebe como entradas a descrição da tarefa, as posições inicial (antes de montagem) e final (após a montagem) de todos os componentes de montagem e a definição do contacto que especifica o tipo de contactos entre objectos na configuração montada (por exemplo, aparafusamento, inserção, etc.). Esta informação descreve completamente a tarefa de montagem, podendo ser obtida muito facilmente usando um sistema interactivo (do tipo CAD), capaz de definir as posições relativas aos objectos.

O sistema também necessita de um modelo do mundo (definição da célula de trabalho que tem de executar a tarefa), construído a partir de informação fornecida pelo utilizador:

- descrição geométrica de todos os elementos envolvidos (componentes, fixações, ligações do robô, etc.), obtidos a partir de um interface normalizado fornecido por vários sistemas de CAD;
- modelo dos robôs (parâmetros cinemáticos, definição da linguagem do robô);
- *layout* da célula (posições dos robôs e das fixações).

O sistema procede então à geração automática de uma sequência segura de acções/ movimentos do robô, necessários para realizar a tarefa. Tais instruções são depois traduzidas para comandos específicos do controlador do robô (e executáveis no robô) e passadas para um sistema de simulação onde se podem visualizar os movimentos do robô.

Vemos que, a partir da informação de entrada, o sistema é capaz de determinar o programa executável para os robôs da célula, sem que a presença do robô real, ou da célula de trabalho, seja necessária.

6 O futuro da integração da simulação com o planeamento de tarefas

Além da evolução dos métodos de programação de robôs de uma programação explícita - na qual é necessário programar todas as instruções que o robô tem que executar - para uma programação implícita ao nível da tarefa - na qual se indica ao robô o objectivo que se pretende alcançar, começam também a surgir vários desenvolvimentos para integrar a simulação com o planeamento de tarefas e com a inteligência artificial, no sentido de propor simuladores com boas capacidades de optimização de *layouts* (tal como o sistema proposto por Dao (1995)) e com capacidades de escalonamento em tempo real (caso do sistema apresentado na Figura 4.8 [Kovács, 1994]).

Além destas evoluções de cariz mais genérico, na integração das tarefas de planeamento com a execução das tarefas (concretamente no que diz respeito à robótica de montagem e ao planeamento de tarefas de montagem) os tópicos de investigação importantes nesta área são, de acordo com Bekey (1996), os seguintes:

- planeamento eficiente de sequências de montagem para número elevado de componentes (no sentido de evitar as explosões combinacionais);
- combinação do planeamento com o escalonamento;
- integração das tolerâncias no planeamento da montagem;
- combinação do planeamento da montagem com o projecto e o reprojecto (para um melhor fabrico);
- e combinação do planeamento com inspecção.

A este respeito gostaríamos de citar aqui a opinião de Bekey (1996):

"Eu prevejo que os futuros sistemas de modelização de sólidos se encontrarão fortemente interligados com planeadores baseados em inteligência artificial, que fornecerão sequências de montagem eficientes (nalguns sentidos, talvez mesmo óptimas) para os produtos que se encontram representados. Além do mais, os palaneadores de montagem vão gerar instruções de montagem adequadas, quer para os humanos, quer para os robôs, conforme apropriado".

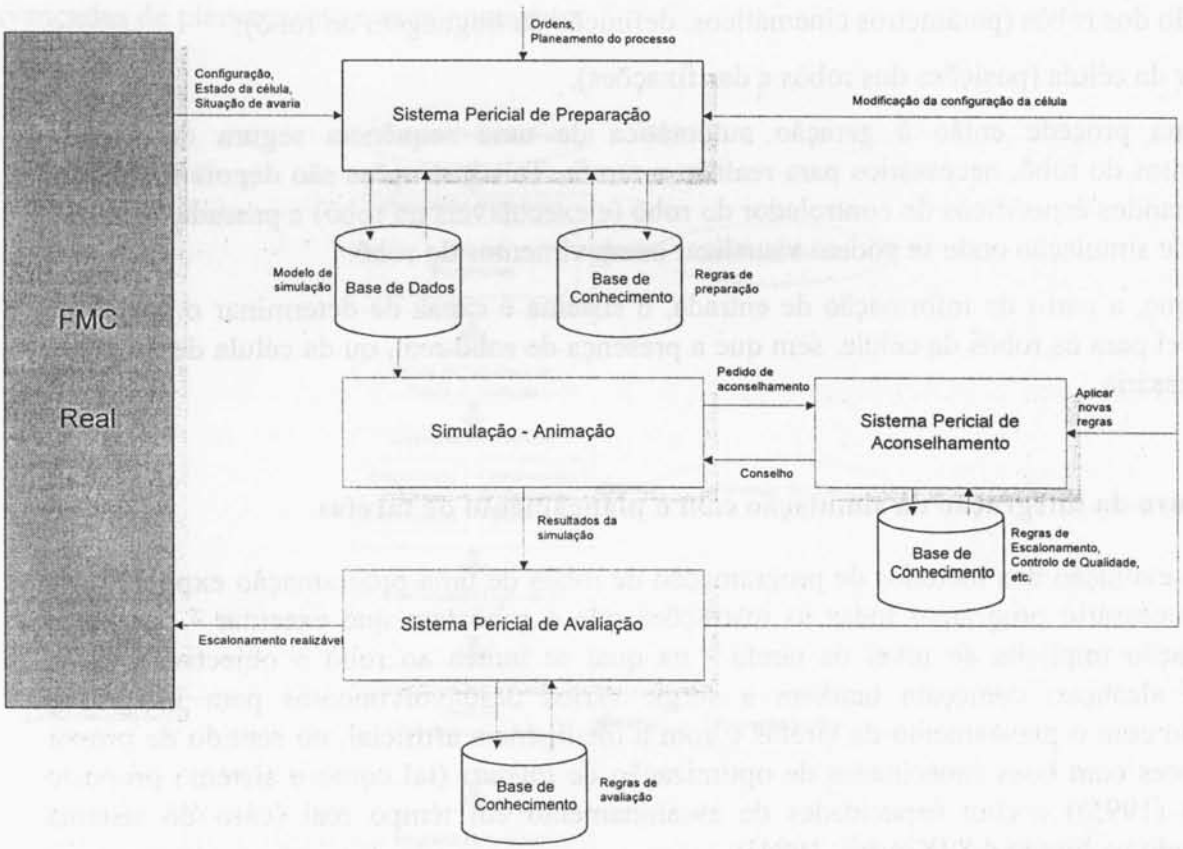


Figura 4.8: Arquitectura do sistema de simulação Sim Sched-Q [Kóvacs, 1994]

Capítulo 5

Modelização da Célula de Montagem do CCP

Neste capítulo procura-se dar resposta às seguintes questões:

- Qual é a arquitectura da Oficina Piloto do CCP;
- Como é efectuada a interligação entre a Célula de Montagem do CCP e as outras células relevantes para o seu funcionamento (Célula de Armazenamento e de Transporte de Materiais; Departamento de Projecto);
- Quais são os equipamentos disponíveis na Célula de Montagem do CCP;
- Como é que estes se encontram integrados;
- Qual a estrutura e como funciona a Célula de Montagem do CCP;
- Como foi modelizada a Célula de Montagem do CCP.

O CCP - Centro de CIM do Porto - apresenta como um dos seus objectivos principais demonstrar à indústria Portuguesa a possibilidade de se aplicar, com vantagem, o conceito CIM a uma empresa fabril, através da implementação distribuída de funções de administração, relações com o exterior (compras e vendas), projecto (de produtos e de processos de fabrico), planeamento, controlo e monitorização da produção. Este objectivo foi realizado através da implementação de uma Oficina Piloto de demonstração, onde é possível fazer formação em tecnologias CIM, desenvolver actividades de I&D e produzir peças metálicas em pequenas séries.

É nesta Oficina Piloto que se encontram os equipamentos que serviram de plataforma à implementação prática do trabalho apresentado nesta Tese, nomeadamente a respectiva Célula de Montagem, cuja estrutura e princípio de funcionamento se procuram modelizar neste capítulo.

Esta célula, bem como toda a Oficina Piloto de demonstração, foi desenvolvida durante o decurso do Projecto ESPRIT 5629, cuja conclusão ocorreu em Outubro de 1995. O funcionamento detalhado da Célula de Montagem e os princípios que nortearam a sua execução são descritos por Guedes (1995a e 1995b).

Este autor foi o responsável pelo trabalho de implementação desta célula, tendo ficado a seu cargo o desenvolvimento do controlador de célula assim como a integração dos vários subsistemas existentes dentro desta. Foi também o responsável pelo desenvolvimento das rotinas do Sistema de Visão Artificial, em conjunto com o Eng. Jorge Baptista. O autor da presente Tese participou também nos trabalhos de desenvolvimento desta célula, nomeadamente no que toca aos aspectos relacionados com o controlador do robô e à programação deste, tendo para o efeito tido a colaboração do Eng. Paulo Magalhães. Adicionalmente, desenvolveu todo o trabalho de simulação que possibilitou que a programação do robô existente nesta célula passasse a ser efectuado *off-line*, com todas as vantagens daí resultantes para o CCP.

Antes de avançar para a modelização da Célula de Montagem, passa-se a descrever de forma sucinta a Oficina Piloto de Fabrico do CCP e os equipamentos que a compõem (nomeadamente no que se refere à arquitectura de implementação e à comunicação entre células), dando especial relevo àqueles que interactuam com a Célula de Montagem, de forma a facilitar a compreensão do trabalho realizado e a sua integração com os outros equipamentos da fábrica.

1 A montagem como parte do processo produtivo

De acordo com Rampersad (1994), o fabrico e a montagem de partes constituintes de uma peça, em conjunto, formam subprocessos coerentes dentro do processo produtivo. No fabrico dessas peças, as matérias primas são processadas ou transformadas em partes de produtos. Durante o fabrico são alterados a forma, os tamanhos e/ou as propriedades dos materiais. Na montagem, as partes dos produtos são juntas em submontagens ou produtos finais.

A Figura 5.1 mostra as relações entre estes processos funcionais e os processos de controlo mais importantes de uma empresa. Tal figura mostra que a montagem, através do fluxo de materiais ou de produtos, se encontra ligada ao fabrico de partes e que, através do fluxo de informação, se encontra ligada com o *marketing*, com o planeamento do produto, com o desenvolvimento do produto, com o planeamento do processo e com o controlo da produção. A montagem é definida aqui como a junção de partes e/ou submontagens, de forma a que surja uma unidade - a peça final. Uma submontagem é a composição de partes numa unidade de produto. O processo de montagem é determinado pela maneira e sequência pela qual os componentes dos produtos são juntos de forma a darem origem a um produto completo. Este processo compreende uma série de submontagens, que podem ser divididas em pré-montagens e montagens finais. A saída da fase de pré-montagem consiste essencialmente em submontagens e a saída da fase de montagem final consiste nos produtos finais (Figura 5.1).

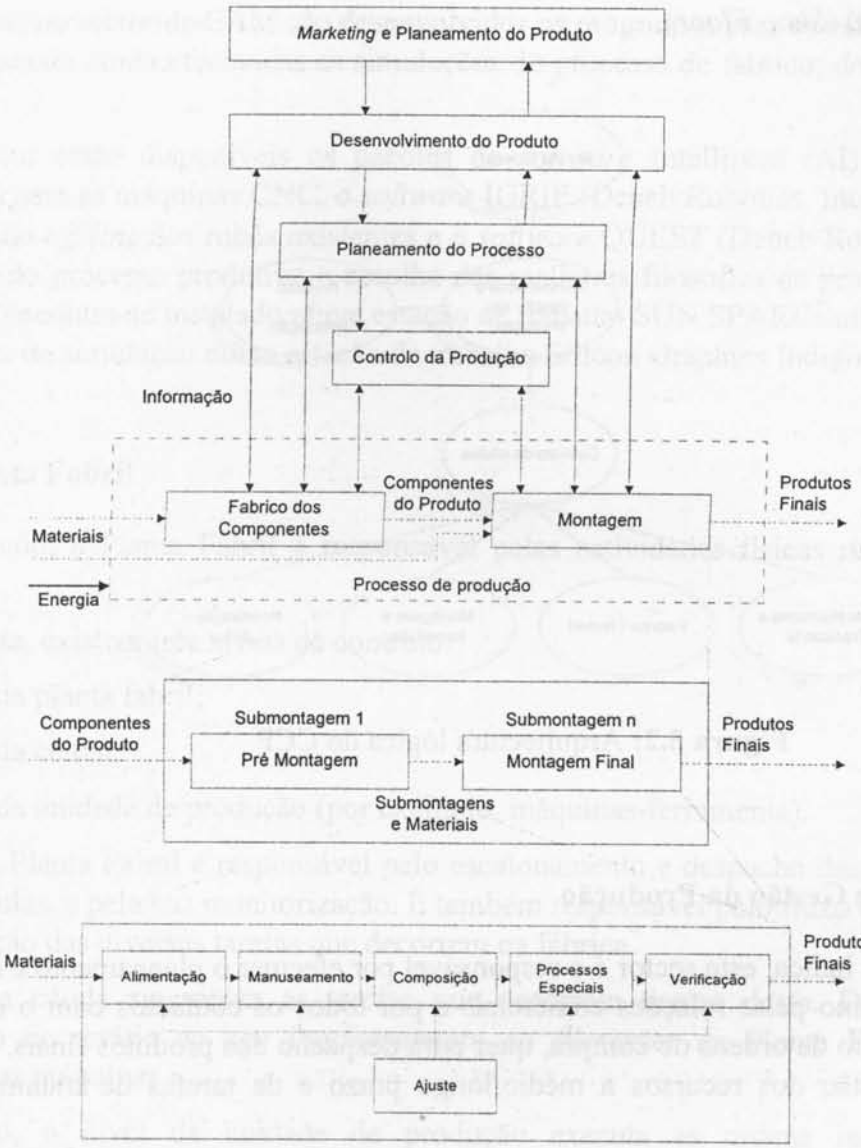


Figura 5.1: A montagem como parte do processo de produção [Rampersad, 1994]

2 Arquitectura da Oficina Piloto do CCP

2.1 Organização da Oficina Piloto do CCP

A Oficina Piloto do CCP segue, nas suas linhas mestras, a arquitectura acabada de descrever.

Como se pode observar na Figura 5.2, a Oficina Piloto do CCP está dividida em três grandes áreas, designadas por:

- área de Planeamento e Gestão de Produção;
- área de Projecto e Simulação;

- área da Planta Fabril (*Shop Floor*).

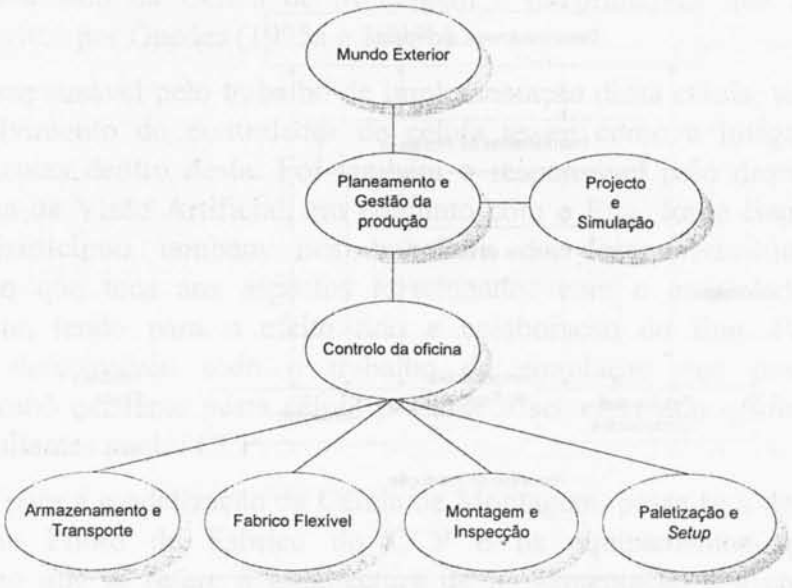


Figura 5.2: Arquitectura lógica do CCP

2.1.1 Planeamento e Gestão da Produção

Tal como o seu nome indica, este sector é o responsável por efectuar o planeamento e a gestão da produção, bem como pelas relações comerciais e por todos os contactos com o exterior, quer para o lançamento de ordens de compra, quer para despacho dos produtos finais. É ainda responsável pela gestão dos recursos a médio/longo prazo e de tarefas de tratamento de informação.

Este sector está equipado com uma estação de trabalho IBM, na qual se encontra a correr o *software* de planeamento e gestão da produção TRITON (Baan).

2.1.2 Projecto e Simulação

A área de Projecto e de Simulação é a responsável pela preparação da produção, envolvendo-se em campos que vão desde o desenvolvimento do produto até ao planeamento dos processos de fabrico e à simulação da produção, sendo constituída por um sector de CAD e um sector de CAM.

No sector de CAD são projectadas as peças que se pretendem fabricar na Oficina Piloto do CCP. Este sector dispõe de uma estação de trabalho SUN SPARCstation 20, na qual se encontra instalado o Pro/Engineer (PTC), que é o *software* utilizado no projecto das peças.

Por sua vez, no sector de CAM são desenvolvidos os programas para as máquinas CNC e para os robôs, sendo ainda efectuadas as simulações do processo de fabrico, de forma a optimizá-lo.

Neste sector estão disponíveis os pacotes de *software* Intellipost (AI) para geração dos programas para as máquinas CNC, o *software* IGRIP (Deneb Robotics, Inc.) para simulação e programação *off-line* dos robôs existentes e o *software* QUEST (Deneb Robotics, Inc.) para a simulação do processo produtivo e escolha das melhores filosofias de produção. O *software* Intellipost encontra-se instalado numa estação de trabalho SUN SPARCstation 10 e os pacotes de *software* de simulação numa estação de trabalho Silicon Graphics Indigo II.

2.1.3 Planta Fabril

Pelo seu lado, a Planta Fabril é responsável pelas actividades físicas relacionadas com a produção.

Dentro desta, existem três níveis de controlo:

- o nível da planta fabril;
- o nível da célula;
- o nível da unidade de produção (por exemplo, máquinas-ferramenta).

O nível da Planta Fabril é responsável pelo escalonamento e despacho das ordens de fabrico para as células, e pela sua monitorização. É também responsável pelo fluxo de materiais e pela sincronização das diversas tarefas que decorrem na fábrica.

O nível da célula sincroniza as tarefas que decorrem dentro desta. Deve ir recolher a informação necessária ao seu funcionamento ao *fileserver* da Planta Fabril, e recebe o *feedback* das máquinas.

Por último, o nível da unidade de produção executa as ordens recebidas do nível imediatamente superior e envia para este nível informação de *feedback* (para monitorização e controlo).

2.1.3.1 Constituição da Planta Fabril

A fábrica propriamente dita é constituída por quatro células:

- Célula de Paletização e de Aferição de Ferramentas (CPAF);
- Célula de Fabrico Flexível (CFF);
- Célula de Montagem (CM);
- Célula de Armazenamento e de Transporte de Materiais (CATM).

Na Figura 5.3 apresenta-se o *layout* da Planta Fabril do CCP, onde se pode ver a localização, dentro desta, das células acima referidas.

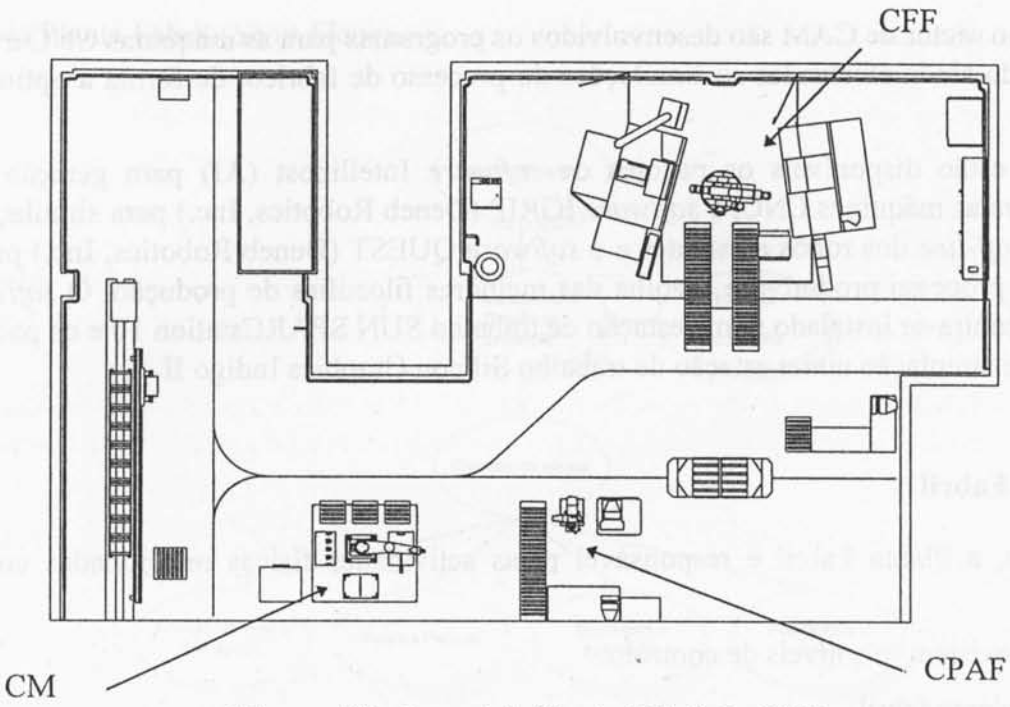


Figura 5.3: *Layout* da Planta Fabril do CCP

Cada uma destas células tem o seu próprio controlador (Controlador de Célula), que só comunica com o controlador de toda a Planta Fabril (*Shop Floor Controller*). O Controlador da Célula de Paletização e Aferição de Ferramentas não é um verdadeiro controlador, pois nesta célula o controlo é humano (apenas faz o interface entre o operador humano e o sistema).

2.1.3.1.1 Célula de Paletização e de Aferição de Ferramentas

Como acabou de ser dito, a Célula de Paletização e de Aferição de Ferramentas funciona em modo manual, pois existe um operador para realizar as várias tarefas da célula. A célula tem como objectivos:

- montar conjuntos de ferramentas para os centros de torneamento e de maquinagem;
- montar conjuntos de garras para o robô de carga/descarga e para o robô de montagem;
- aferir as ferramentas e as garras que foram montadas;
- paletizar e despaletizar os materiais para a Célula de Fabrico e para a Célula de Montagem.

Durante a realização destas operações, o operador tem um terminal que lhe permite o interface com o controlador da célula.

Os recursos físicos desta célula são os seguintes :

- mesa de paletização;
- máquina de aferição de ferramentas (marca ELBO CONTROLLI, modelo AR2000 GA);

- mesa de transferência motorizada (gerida pelo Controlador da Célula de Armazenamento e de Transporte de Materiais).

2.1.3.1.2 Célula de Fabrico Flexível

Tal como o seu nome indica, a função da Célula de Fabrico é a de produzir (fabricar) peças, usando os recursos nela disponíveis. Os recursos principais são um centro de torneamento, um centro de maquinagem e um robô de carga/descarga.

O centro de torneamento é da marca LEALDE, modelo TCN10, com controlador numérico SINUMERIK 880T (SIEMENS); o seu objectivo é o torneamento de peças com geometria cilíndrica.

Por sua vez, o centro de maquinagem é o modelo B500 da KONDIA, com um controlador numérico FANUC 16MA; tem como objectivo maquinar peças com qualquer forma.

Quanto ao robô, a sua função é a de efectuar a carga/descarga das duas máquinas CNC instaladas na Célula de Fabrico. A carga e a descarga dessas duas máquinas é efectuada tendo como origem/destino dos materiais, respectivamente, as duas mesas de transferência motorizadas (ver Figura 5.4).

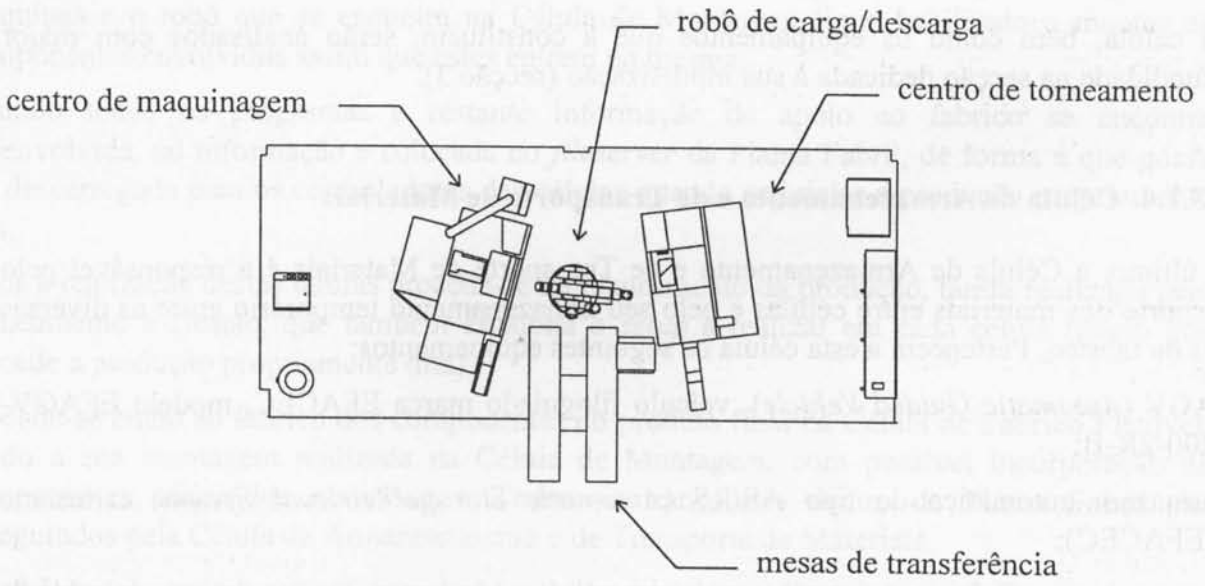


Figura 5.4: *Layout* da Célula de Fabrico Flexível do CCP

O robô utilizado é um robô KUKA IR163/30.1, com seis eixos. O controlador deste robô é o SIEMENS ACR20, com a linguagem de programação SIRL, de alto nível e baseada na linguagem de programação IRL, descrita no Capítulo 3.

De forma a que o robô seja capaz de manipular convenientemente as peças, como requerido pelos centros de maquinagem e de torneamento, foi constituído um *buffer* externo (onde o robô pode pousar, virar e voltar a pegar nas peças).

Por motivos de segurança, este robô encontra-se equipado com um sistema de detecção de sobrecargas e de colisões, capaz de parar o movimento do robô quando este sofre uma colisão ou está sujeito a uma situação de sobrecarga mecânica. Tal sistema é um OPS (*Overload Protection System*) da SHUNK.

Esta célula dispõe também de um armazém de garras, com capacidade para quatro garras, montadas sobre flanges SHUNK GWK80, encontrando-se o robô equipado com um sistema automático de troca de garras (SHUNK GWA80) de tipo pneumático. É neste armazém de garras que se encontram as diferentes garras para a carga/descarga de peças para as máquinas CNC.

Por fim, desta célula fazem ainda parte duas mesas de entrada/saída motorizadas, que são geridas pelo Controlador da Célula de Armazenamento e de Transporte de Materiais.

2.1.3.1.3 Célula de Montagem

A Célula de Montagem tem por função a montagem de peças ou componentes de um dado produto, produzidos internamente na Célula de Fabrico Flexível ou provenientes do exterior. O resultado das operações desta célula é o produto final, ou submontagens desse mesmo produto.

Esta célula, bem como os equipamentos que a constituem, serão analisados com maior profundidade na secção dedicada à sua modelização (secção 3).

2.1.3.1.4 Célula de Armazenamento e de Transporte de Materiais

Por último, a Célula de Armazenamento e de Transporte de Materiais é a responsável pelo transporte dos materiais entre células e pelo seu armazenamento temporário entre as diversas fases de fabrico. Pertencem a esta célula os seguintes equipamentos:

- AGV (*Automatic Guided Vehicle*), veículo filoguiado marca EFACEC, modelo EFAGV-200-2R-B;
- armazém automático do tipo AS/RS (*Automatic Storage/Retrieval System*) cartesiano (EFACEC);
- mesas de transferência, existentes nas várias células, bem como a mesa de entrada/saída de materiais da fábrica e a mesa de acesso ao armazém.

2.2 Realização de um produto na Oficina Piloto do CCP

Após esta breve descrição da estrutura da Oficina Piloto do CCP (e para tornar mais claro o seu princípio de funcionamento) vamos apresentar, também de uma forma sucinta, quais os passos normalmente necessários para nela se efectuar a produção de um produto, desde a sua encomenda até à expedição do produto final resultante.

Todo o processo se inicia com a encomenda de um determinado produto. Após a recepção da encomenda, inicia-se a fase de preparação da produção. As peças ou componentes desse produto são adquiridas no exterior ou começam a ser desenvolvidas pelo Sector Mecânico. Neste último caso, o Sector Mecânico apoia-se num sistema de CAD Pro/Engineer, no qual são desenhadas as peças e feitos os estudos de engenharia, sobre as peças em si e sobre o melhor projecto a adoptar, tendo em vista o fabrico e a montagem (poder-se-á dizer que o projecto de um determinado produto é baseado nos conceitos DFM - *Design For Manufacturing* - e DFA).

Neste sistema de CAD são também projectados todos os acessórios necessários (garras para robôs, bases de fixação, etc.) para a correcta execução dos produtos pedidos.

Estas tarefas são realizadas em estreita colaboração com a área da simulação de robôs, especialmente no que concerne aos testes dos acessórios desenvolvidos no ambiente simulado das Células de Fabrico e Montagem.

Após a fase de Projecto Mecânico ter sido dada por concluída, passa-se à fase de geração de programas para as máquinas CNC que vão fabricar os componentes do produto em questão (ou são fornecidos os desenhos para a sua execução no exterior - subcontratação).

Procede-se também à geração de programas para os robôs, de forma a que o robô que se encontra na Célula de Fabrico Flexível fique habilitado a carregar/descarregar as respectivas máquinas e o robô que se encontra na Célula de Montagem fique habilitado a montar os componentes envolvidos assim que estes entrem na mesma.

Quando todos os programas e restante informação de apoio ao fabrico se encontra desenvolvida, tal informação é colocada no *fileserver* da Planta Fabril, de forma a que possa ser descarregada para os controladores das células quando se iniciar a produção propriamente dita.

Após a realização destas tarefas procede-se ao planeamento da produção, tarefa realizada pelo Planeamento e Gestão, que também escalona o *setup* a realizar em cada célula (passo que precede a produção propriamente dita).

Procede-se então ao fabrico dos componentes do produto final na Célula de Fabrico Flexível, sendo a sua montagem realizada na Célula de Montagem, com possível incorporação de componentes adquiridos no exterior. Todos os transportes dentro da Planta Fabril são assegurados pela Célula de Armazenamento e de Transporte de Materiais.

O robô que se encontra na Célula de Montagem tem à sua disposição um sistema de visão artificial (que se encontra embarcado no próprio robô), ao qual recorre sempre que necessita de informação especifica para a realização de determinadas operações de montagem.

Após a montagem das peças ter sido dada por concluída, estas são armazenadas controladamente tendo em vista a sua futura expedição para o cliente.

3 A Célula de Montagem do CCP

Neste ponto, vamos efectuar uma descrição o mais completa possível dos equipamentos, da estrutura e do funcionamento da Célula de Montagem do CCP.

Como já foi dito, a principal função da Célula de Montagem, tal como o seu nome indica, é a montagem de itens produzidos na Oficina Piloto do CCP, concretamente na sua Célula de Fabrico Flexível, ou adquiridos no mercado. A saída desta célula são os produtos finais, ou submontagens de produtos finais.

3.1 Equipamentos da Célula de Montagem do CCP

3.1.1 Robô de montagem

O robô de montagem é um SCARA com 4 eixos, da Adept Technology Inc., modelo Adept Three. Este robô está equipado com um controlador Adept CC-A Series, que apresenta uma arquitectura Multi-bus, sendo o sistema operativo multi-tarefa. Possui quatro ligações série RS232 com o exterior e 16 entradas/saídas digitais. Apresenta uma linguagem de programação de alto nível, o V/V+, dispondo de uma consola de programação integrada no controlador. Como acessório, possui ainda um sistema automático de troca de garras (SHUNK GWA64).

3.1.2 Sistema de visão artificial

O sistema de visão artificial é da Cognex Corporation, modelo Cognex 4200EX, com arquitectura VME (*Virtual Machine Extension BUS bar*), sistema de processamento sequencial, dispondo de quatro ligações série RS232 com o exterior, 32 entradas/saídas digitais, programável usando a linguagem C. A programação é efectuada a partir de PC-DOS, através de uma linha série RS232 dedicada.

A câmara é uma CCD (*Charge Coupled Device*) a preto e branco, da PULNIX, Inc.

3.1.3 Mesas de transferência

Esta célula dispõe de três mesas de transferência motorizadas, que compõem o sistema de alimentação de materiais e componentes da célula, permitindo a sua entrada/saída. O controlo destas mesas não é da responsabilidade do controlador da Célula de Montagem, estando antes a cargo do Controlador da Célula de Armazenamento e de Transporte de Materiais.

3.1.4 *Buffers* internos de peças

Internamente, esta célula dispõe de dois *buffers* de peças, que são usados para o armazenamento de partes, componentes ou submontagens, dentro do perímetro da célula, assim permitindo uma maior flexibilidade para o processo de montagem. A subdivisão dos *buffers* internos de peças é devida à estrutura física da mesa existente. Além da divisão lógica, esta divisão corresponde a uma divisão física que foi realizada com o objectivo de se poderem

pousar contentores ou de se criarem pontos de referência em cada um deles, tal como será descrito posteriormente.

3.1.5 Mesa de operações

A Célula de Montagem possui ainda uma mesa de operações, que é o local físico onde as operações de montagem são executadas. Esta mesa dispõe de actuadores mecânicos cujo objectivo é a fixação das peças, mantendo os seus componentes em posição estável, de forma a que as operações de montagem possam ser efectuadas sem deslocamentos indesejáveis.

Actualmente, existem para o efeito dois cilindros pneumáticos montados na própria mesa e controlados por uma lógica digital.

3.1.6 Armazém de garras

Por último, esta célula também dispõe de um armazém de garras, com capacidade para quatro garras, como se referiu montadas sobre flanges SHUNK GWK64.

Este armazém é o suporte para a implementação do sistema de troca automática de garras e nele se encontram as várias garras e ferramentas necessárias para a concretização das montagens a efectuar pelo robô SCARA.

3.2 *Layout* da Célula de Montagem do CCP

A forma como os equipamentos descritos se encontram implantados na Célula de Montagem, ou seja o seu *layout*, é a que se encontra ilustrada na Figura 5.5.

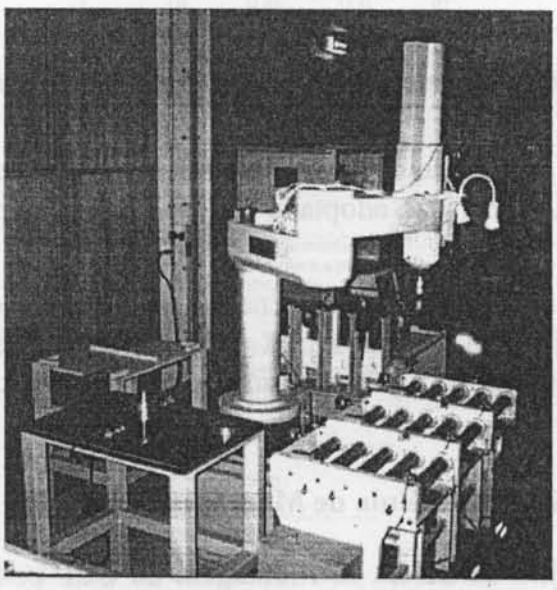


Figura 5.5: *Layout* da Célula de Montagem do CCP

Na mesma figura vêem-se claramente o robô de montagem, a câmara do sistema de visão artificial instalada no extremo da ligação 2 do robô e a plataforma sobre a qual estão montados quer o robô, quer os outros componentes da célula de montagem. Pode-se ainda ver, na retaguarda do robô de montagem, o sistema automático de troca de garras com que este se encontra equipado, assim como as garras necessárias para a montagem das peças PD001 e PD002, (referências de duas peças produzidas no CCP - as imagens destas peças são apresentadas no Anexo A), que se encontram neste armazém.

Também se podem ver, sobre a mesa que se encontra à frente do robô (mesa de operações), os dispositivos de fixação para estas mesmas peças (PD001 e PD002) e os armazéns de O-Rings e parafusos. Estes armazéns não passam de *paletes* adaptadas ao transporte destes componentes de montagem, mas que têm que ser alimentados manualmente por um operador (por motivos económicos, e já que ao nível de funcionalidades e requisitos - para o robô e para a montagem na sua globalidade - as duas soluções são bastante parecidas, o Sector Mecânico do CCP optou por esta solução, em detrimento de uma solução que utilizasse alimentadores vibratórios automáticos).

Esta célula encontra-se dividida em zonas físicas (e lógicas) baseadas na funcionalidade de cada zona (ver Figura 5.6).

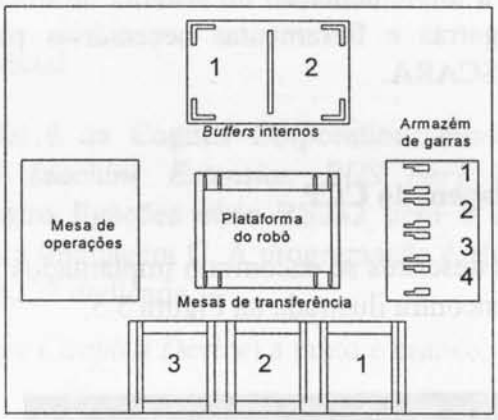


Figura 5.6: Divisão física e lógica da Célula de Montagem do CCP

À frente do robô encontra-se a mesa de operações e à sua retaguarda encontra-se o armazém de garras, onde se pode ver a numeração adoptada para cada posição das garras dentro deste armazém.

Dos lados do robô situam-se as mesas de transferência e os *buffers* internos, respectivamente do seu lado esquerdo e direito. Da mesma forma que com o armazém de garras, também nestes dois casos podemos observar na figura as numerações adoptadas.

3.3 Representação esquemática da Célula de Montagem do CCP

Para descrever o funcionamento da Célula de Montagem do CCP poder-se-ia, por exemplo, utilizar redes de Petri, diagramas de transição de estados ou mesmo recorrer a um fluxograma.

Dada a simplicidade com que se consegue efectuar essa modelização, recorremos a esta última aproximação. No fluxograma apresentado na Figura 5.7 encontram-se descritas as funções necessárias para se realizar uma montagem nesta célula.

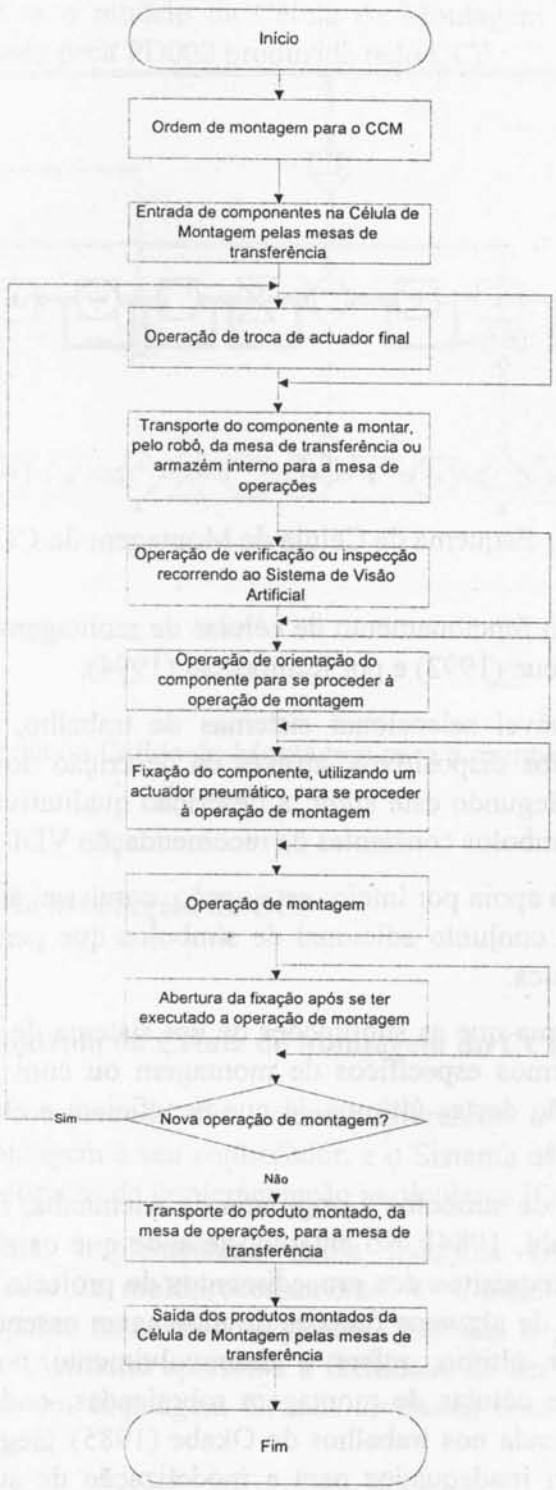


Figura 5.7: Representação do funcionamento da Célula de Montagem do CCP

No entanto, e uma vez que pensamos ser preferível efectuar tal esquematização recorrendo a símbolos normalizados, também apresentamos no presente texto uma esquematização da Célula de Montagem do CCP, em que se adoptou a simbologia desenvolvida por Rampersad (ver Figura 5.8).

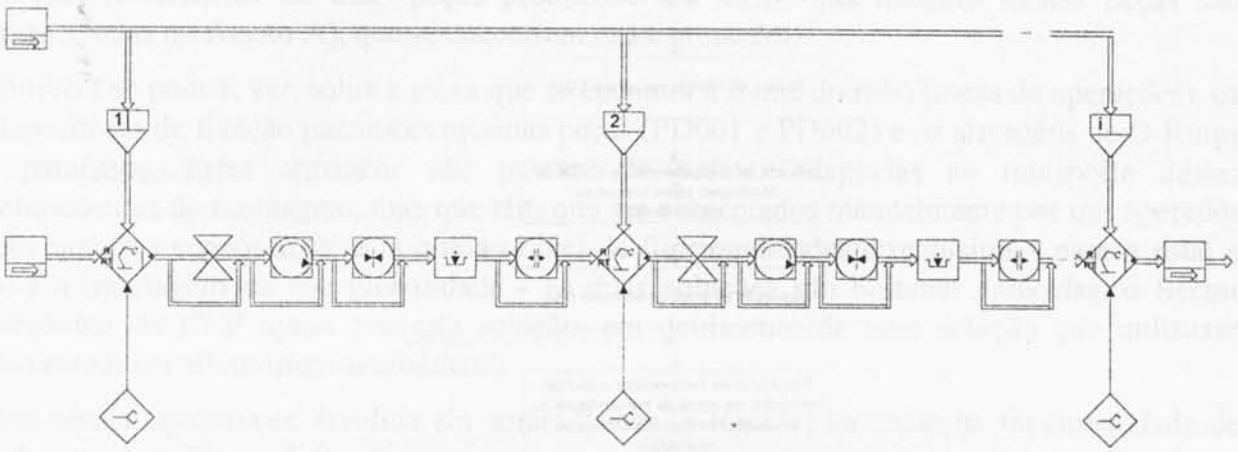


Figura 5.8: Esquema da Célula de Montagem do CCP

Esta forma de representação do funcionamento de células de montagem robotizadas tem sido defendida, entre outros, por Gleue (1992) e por Rampersad (1994).

Gleue afirma que é aconselhável seleccionar sistemas de trabalho, meios de transporte, alimentação, orientação e outros dispositivos, através da descrição do processo de trabalho com símbolos normalizados. Segundo este autor, a descrição qualitativa do procedimento de manuseamento pode usar os símbolos constantes da recomendação VDI-2860.

Por sua vez, Rampersad não só apoia por inteiro esta opção, como vai ainda mais longe ao ser ele próprio a desenvolver um conjunto adicional de símbolos que permitem estender ainda mais as capacidades de tal técnica.

A este respeito, este autor afirma que as subfunções de um sistema de montagem podem ser formuladas com recurso a termos específicos de montagem ou com a ajuda de símbolos. Defende no entanto a utilização destes últimos, já que beneficiam a clareza e simplificam o projecto funcional dos sistemas.

Afirma ainda que a utilização de símbolos é originária da Alemanha, fazendo referência aos trabalhos de Pahl & Beitz [Pahl, 1984]. No entanto, defende que os símbolos Alemães não cumprem suficientemente os requisitos dos procedimentos de projecto necessários, uma vez que lhes falta a representação de algumas funções de montagem essenciais e não prevêem a estratégia de montagem. Por último, refere o desenvolvimento no Japão de símbolos específicos para o projecto de células de montagem robotizadas, onde surgiu mesmo uma proposta de normalização baseada nos trabalhos de Okabe (1985). Segundo Rampersad, tais símbolos também se mostram inadequados para a modelização de sistemas de montagem robotizados tomados como um todo, uma vez que se encontram em falta, por exemplo, as funções de movimentação (princípios de garras) e as principais funções de transporte.

Pelos motivos acabados de apresentar, o próprio Rampersad desenvolveu um conjunto de símbolos (apresentados e descritos no Anexo B) orientados para os procedimentos de projecto de sistemas de montagem robotizados. Estes símbolos permitem desenvolver quer a estrutura do sistema, quer a funcionalidade de uma célula de montagem robotizada.

Na Figura 5.9 apresenta-se o modelo da Célula de Montagem para o caso específico da montagem da já mencionada peça PD002 produzida pelo CCP.

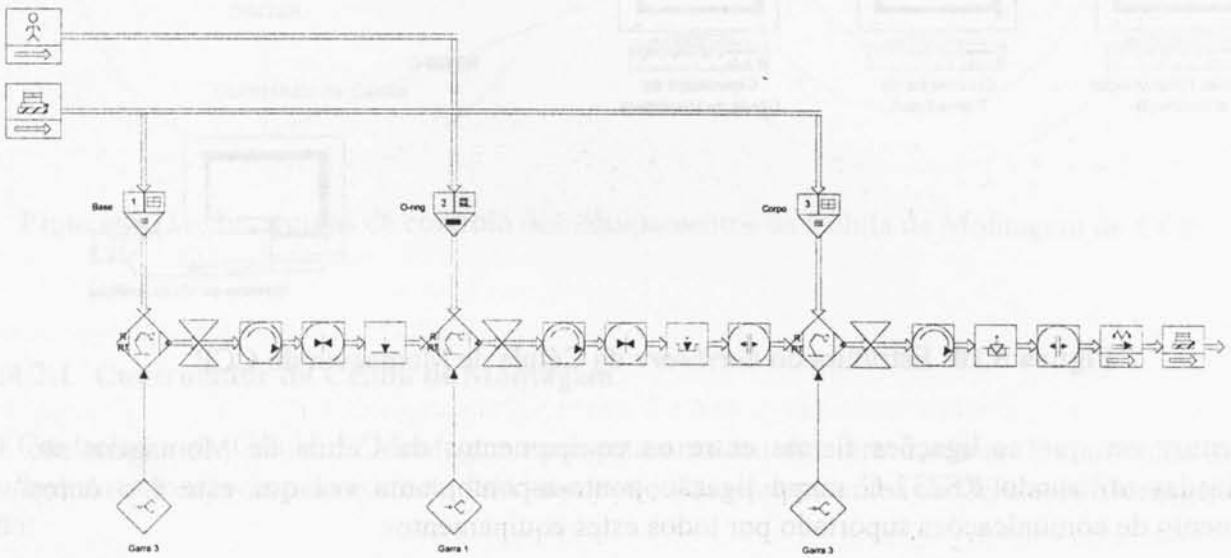


Figura 5.9: Esquema da Célula de Montagem para a montagem da peça PD002

3.4 Controlo da Célula de Montagem do CCP

3.4.1 Arquitectura de controlo da Célula de Montagem do CCP

A Célula de Montagem compreende três sistemas diferentes: o Controlador da Célula de Montagem, o robô de montagem e seu controlador, e o Sistema de Visão Artificial. Cada um destes sistemas apresenta tópicos de implementação particulares [Guedes, 1995b].

O Controlador de Célula, implementado numa máquina UNIX, aproveita todas as potencialidades do ambiente de multiprocessamento. O Controlador do robô permite um ambiente multitarefa do ponto de vista do *software*, mas não o permite para comandos do robô. O Sistema de Visão Artificial apresenta a facilidade de ser um sistema modular, num ambiente de programação em linguagem C, mas apresenta como desvantagem o facto de trabalhar de forma sequencial.

Os Controladores dos diversos equipamentos da Célula de Montagem encontram-se dispostos de acordo com a estrutura apresentada na Figura 5.10.

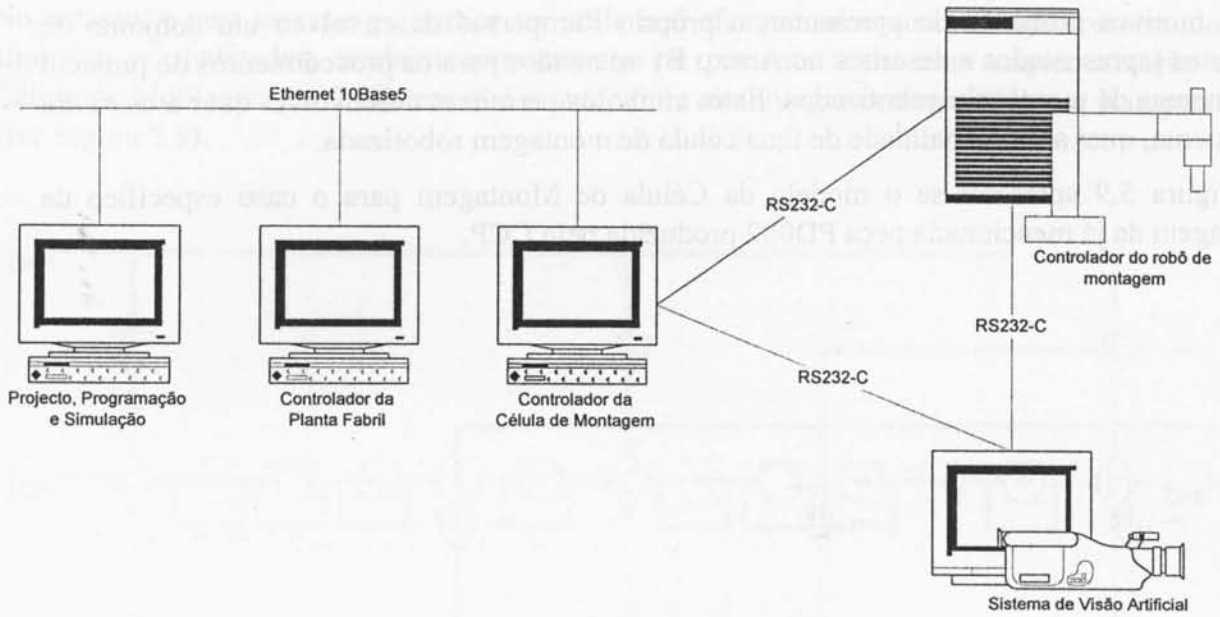


Figura 5.10: Estrutura do *hardware* da Célula de Montagem do CCP

Podemos ver que as ligações físicas entre os equipamentos da Célula de Montagem são realizadas utilizando RS232-C numa ligação ponto-a-ponto, uma vez que este é o único protocolo de comunicações suportado por todos estes equipamentos.

Pelo contrário, a ligação entre a área de Projecto (neste caso concreto representado pela estação de trabalho Silicon Graphics Indigo 2, que se encontra afecta à simulação e programação *off-line* de robôs, correndo o *software* IGRIP) e o Controlador da Célula de Montagem (representado pela estação de trabalho SUN) é efectuada recorrendo a uma ligação sobre Ethernet 10Base5, sobre a qual se encontra o protocolo de comunicações NFS (*Network File System*) para troca de ficheiros entre estas duas máquinas (programas de montagem e ficheiros com instruções de *setup* dos equipamentos da Célula de Montagem).

3.4.2 Hierarquia de controlo

Se olharmos para esta célula a partir do exterior, só vemos o controlador de célula. Cada ordem ou pedido do Controlador da Planta Fabril é processado pelo Controlador da Célula de Montagem. Temos assim, no topo da hierarquia de controlo, o Controlador da Célula de Montagem.

No interior da Célula podemos considerar duas situações (Figura 5.11): uma, em que o Controlador da Célula de Montagem tem o controlo total sobre todos os componentes da Célula de Montagem (esta situação ocorre quando a Célula se encontra à espera de uma ordem de montagem e durante a fase de *setup*); a outra situação ocorre durante a execução de uma ordem de montagem, situação esta na qual o Controlador de Célula comunica unicamente com o Controlador do robô de montagem e este, por sua vez, controla o Sistema de Visão Artificial.

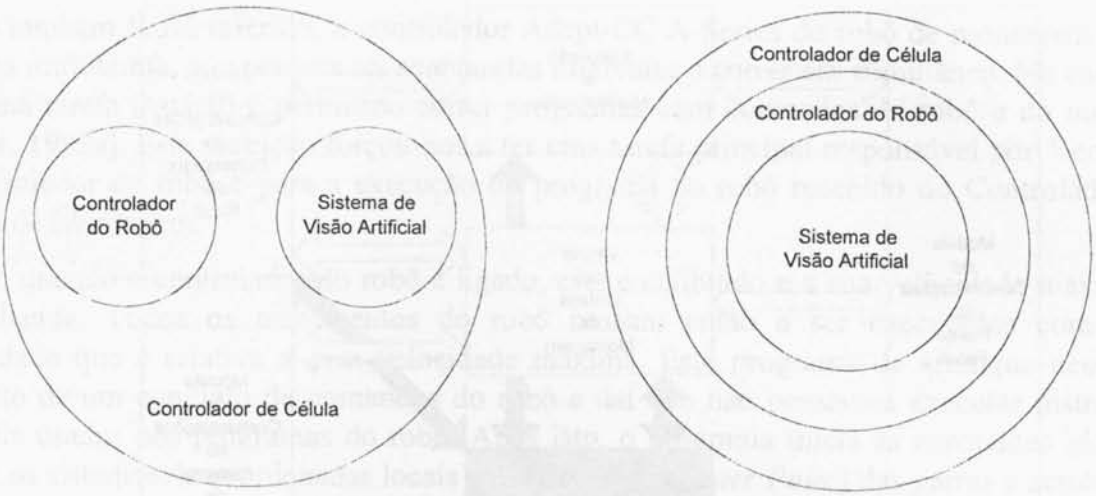


Figura 5.11: Hierarquias de controlo dos equipamentos da Célula de Montagem do CCP

3.4.2.1 Controlador da Célula de Montagem

O Controlador da Célula de Montagem apresenta uma arquitectura modular. Os módulos de *software* responsáveis pelo correcto funcionamento dos equipamentos da Célula de Montagem são:

- Módulo de Arranque: este módulo tem por função a inicialização de todos os outros módulos do Controlador de Célula, verificando o estado dos equipamentos, e arrancando as comunicações com o Controlador da Planta Fabril;
- Gestor das Ordens de Montagem: a principal função deste módulo é a aceitação ou rejeição das ordens de montagem. Ele é o responsável pelo processamento da informação que se encontra inserida nas ordens de montagem, passando-a para os equipamentos da Célula e para os módulos de execução e monitorização;
- Módulo de Execução e Monitorização: este módulo controla o processo de montagem. Como o controlo da execução e a monitorização estão interligados, estas funções foram colocadas no mesmo módulo. Outra funcionalidade deste módulo é a gestão das excepções;
- Módulos de Comunicações da Planta Fabril, do Controlador do Robô e do Sistema de Visão Artificial: estes módulos gerem as comunicações externas ao Controlador da Célula de Montagem. Formatam as mensagens para os diferentes equipamentos, inicializam as comunicações entre eles, etc.

A interligação entre os módulos acabados de descrever é apresentada na Figura 5.12.

Apesar de não indicado na figura anterior, existe ainda outro módulo, que é o Módulo de Monitorização. Este encontra-se ligado a todos os outros módulos de *software* da Célula, sendo o responsável pela apresentação do estado da Célula de Montagem no ecrã do Controlador da mesma e sendo o único que aceita comandos do operador, funcionando como módulo de comando manual da Célula de Montagem.

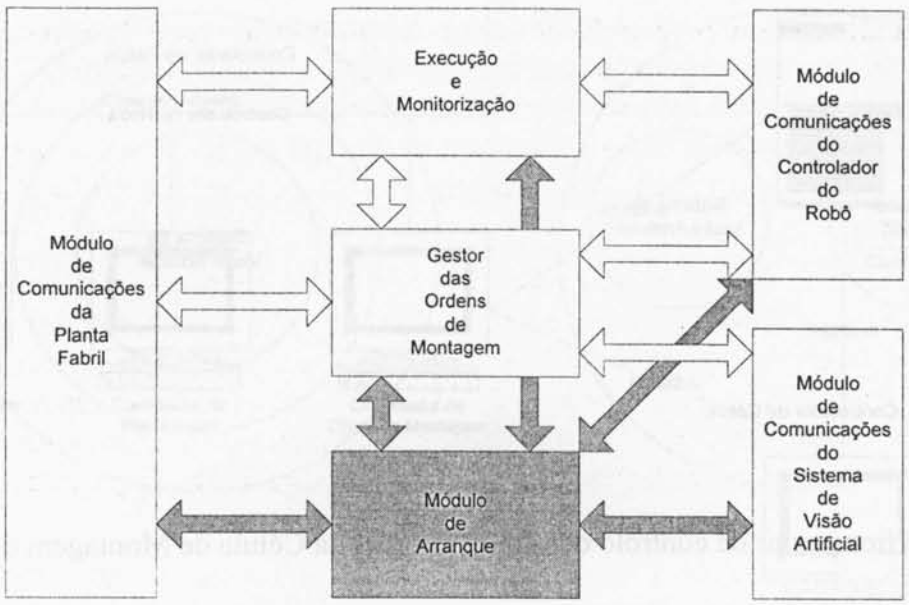


Figura 5.12: Módulos de *software* do Controlador da Célula de Montagem do CCP

Cada módulo de *software* do Controlador da Célula de Montagem é um processo UNIX [Guedes, 1995c]. Toda a informação é passada entre eles, das duas seguintes maneiras:

- *mailboxes*;
- memória partilhada.

É ainda utilizado um segmento de memória partilhada para manter a chamada imagem da célula. Toda a informação respeitante ao estado da célula (acontecimentos da célula, etc.), é guardada nesta zona de memória. Por exemplo, quando o robô move um objecto, informa o Módulo de Comunicações do Controlador do Robô desse facto, e este módulo, durante a fase de descodificação da mensagem do robô, altera as variáveis correspondentes da zona de memória partilhada para que estas reflictam este facto. Após isto, é enviada para a *mailbox* do Módulo de Execução e Monitorização uma pequena mensagem apenas com a identificação do objecto.

A principal vantagem deste tipo de implementação é a modularidade do Controlador de Célula. Cada mensagem, enviada de um módulo para outro, induz neste último um conjunto de operações. Através da alteração destes comportamentos pode-se alterar todo o comportamento do Controlador de Célula [Guedes, 1995c].

3.4.2.2 Controlador do robô de montagem

O controlador do robô pode comunicar com o Sistema de Visão Artificial ou com o Controlador da Célula de Montagem usando o protocolo de comunicações RS232-C, como se viu na Figura 5.10. A rotina para comunicação série implementa as funcionalidades que a seguir se descrevem.

Como também já foi referido, o controlador Adept-CC A-Series do robô de montagem é um sistema multitarefa, que permite ter sete tarefas diferentes a correr em simultâneo. No entanto, só numa tarefa (*task* 0) é permitido correr programas com comandos do robô e do monitor [Adept, 1993a]. Esta restrição forçou-nos a ter uma tarefa principal responsável por "acordar" o controlador do robô e para a execução do programa do robô recebido do Controlador da Célula de Montagem.

Assim, quando o controlador do robô é ligado, este é calibrado e a sua velocidade máxima é estabelecida. Todos os movimentos do robô passam então a ser executados com uma velocidade que é relativa a essa velocidade máxima. Este programa de arranque necessita portanto de um conjunto de comandos do robô e daí que não possamos executar instruções normais usadas nos programas do robô. Após isto, o programa inicia as constantes globais, define os sistemas de coordenadas locais e o TCP (*Tool Center Point*) das garras e actuadores finais.

Nesta fase, vários programas são lançados numa das tarefas disponíveis do controlador do robô. Um destes programas, provavelmente o mais importante de todos, é um programa que está sempre a monitorizar a linha série que liga o controlador do robô ao Controlador da Célula de Montagem. Este programa recebe e executa qualquer tipo de ordem que seja recebida do Controlador da Célula de Montagem. Conclui-se então, que o controlador do robô se encontra sempre à espera de receber mensagens (está sempre à escuta), e que deve executar operações diferentes consoante as mensagens que receber.

O programa de montagem é recebido pela linha série do Controlador da Célula de Montagem e é, desta forma, gravado pelo robô Adept num ficheiro em disco. Após todo o programa ter sido recebido, é carregado para memória e executado. A execução do programa pode ser completa, ou pode ser interrompida durante a execução por uma ordem recebida do Controlador da Célula de Montagem (ordem de interrupção da execução).

Também é enviada para o Controlador da Célula de Montagem a informação de monitorização respeitante à forma como está a decorrer a operação de montagem.

Mais simples é a comunicação que foi implementada com o Sistema de Visão Artificial. O controlador do robô encontra-se também ligado ao sistema de visão artificial e, quando necessário, pede-lhe dados sobre o processo de montagem em curso

Durante a execução de uma montagem, o controlador do robô pergunta várias vezes ao sistema de visão artificial como é que pode pegar nos componentes (elementares ou não) da montagem. Esta comunicação é extremamente simples porque o controlador do robô envia uma mensagem com um pedido de identificação de uma peça e o sistema de visão artificial responde-lhe indicando se detectou ou não essa peça. Caso a tenha detectado, retorna também as coordenadas onde a detectou e a rotação do referencial dessa peça em relação à origem dos referenciais.

Na Figura 5.13 podem-se ver os estados possíveis para o Controlador do Robô de Montagem.

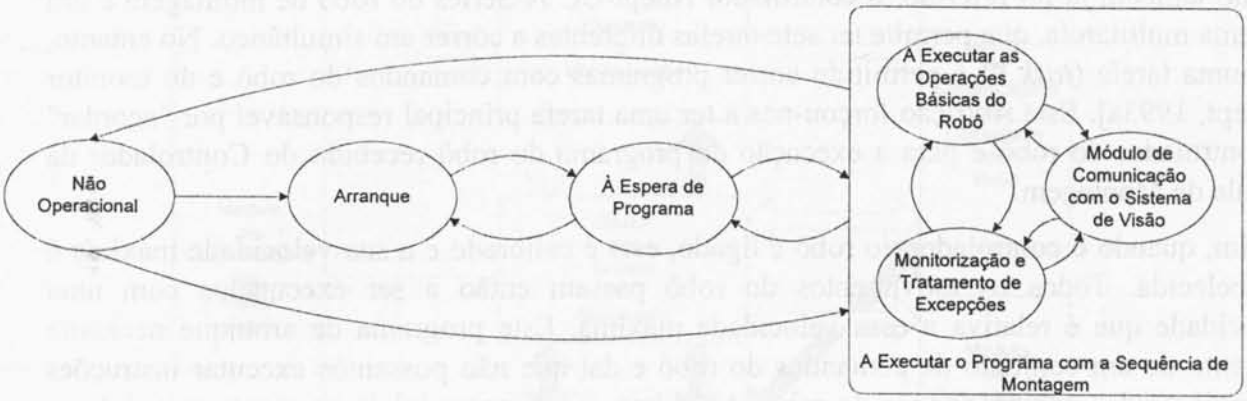


Figura 5.13: Estados do robô da Célula de Montagem do CCP

3.4.2.3 Implementação do Sistema de Visão Artificial

Finalmente, vou referir brevemente a forma como foi implementado o Sistema de Visão Artificial.

O Sistema de Visão Artificial trabalha como servidor, esperando por ordens do Controlador da Célula de Montagem e por perguntas do controlador do robô. Assim que o sistema de visão artificial recebe alguma ordem ou pergunta, em conjunto com os parâmetros necessários para poder compreender essas ordens ou perguntas executa as acções correspondentes e invoca o algoritmo necessário para enviar a resposta.

Basicamente, o sistema de visão artificial tem duas funções principais: detecção e orientação [Guedes, 1995b]. A primeira é executada, por exemplo, para detectar uma peça na mesa de operações ou um parafuso no armazém de parafusos. A orientação calcula o ângulo 3D desse componente. Apesar de só ter estas duas funções principais, o sistema de visão usa-as frequentemente. Daí que esteja a cargo do robô Adept e do Controlador da Célula de Montagem o fornecimento de todos os dados relacionados com os componentes que são necessários para que o sistema de visão artificial consiga atingir os seus propósitos.

Na Figura 5.14 são indicados os estados possíveis para o Sistema de Visão Artificial.

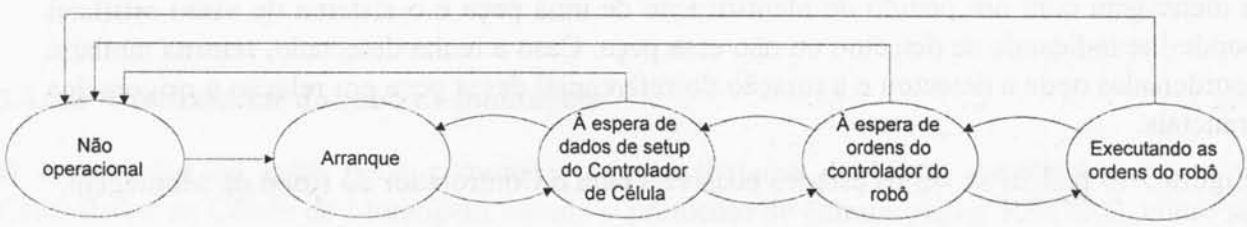


Figura 5.14: Estados do Sistema de Visão Artificial

3.4.3 Funcionamento da Célula de Montagem

Os estados do Controlador da Célula de Montagem são apresentados na Figura 5.15.

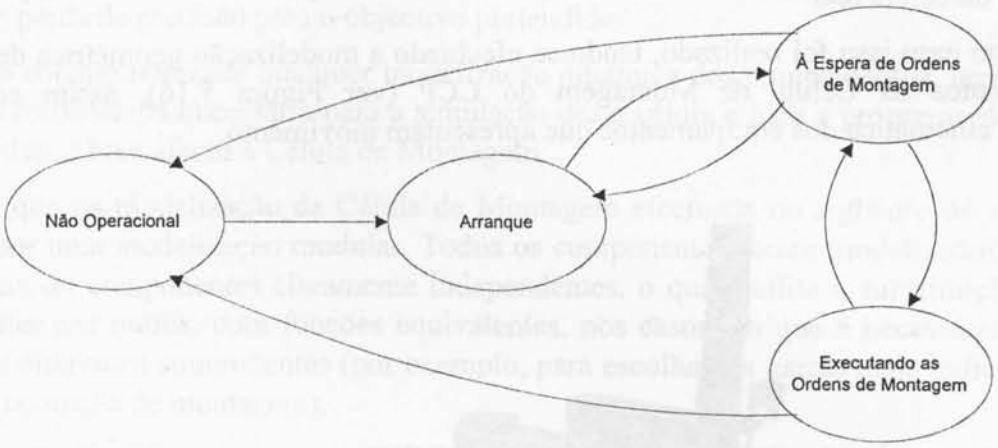


Figura 5.15: Estados do Controlador da Célula de Montagem do CCP

O princípio de funcionamento desta célula, sempre que surge uma ordem de montagem, é o seguinte:

- o Controlador da Planta Fabril envia uma ordem de montagem para a Célula de Montagem;
- o Controlador da Célula de Montagem (gestor das ordens de montagem) processa a ordem de montagem e procura a informação necessária para executar essa mesma ordem. Se encontrar toda a informação necessária, a ordem é aceite; caso contrário, é rejeitada;
- após obter a informação necessária, o Controlador da Célula de Montagem decide qual a informação que deve ser enviada a cada um dos componentes da Célula de Montagem (controlador do robô e sistema de visão artificial);
- quando esta fase de *setup* termina, dá-se uma mudança da hierarquia de controlo dos equipamentos da célula e o Controlador da Célula de Montagem encontra-se pronto para iniciar o processo de montagem. Neste momento o controlador do robô começa a executar o programa recebido e o sistema de visão artificial encontra-se à espera de ordens (pedidos) do robô;
- o Controlador da Célula de Montagem é informado de todas as ordens executadas pelo robô, informação essa que é processada e, se necessário, enviada para o Controlador da Planta Fabril. O Controlador da Célula de Montagem também envia para o controlador do robô a informação de que este necessita durante o processo de montagem (por exemplo, chegada de um contentor numa mesa de transferência);
- no final do processo de montagem, após informar o Controlador da Planta Fabril deste facto, o sistema fica pronto (e à espera) de receber nova ordem de montagem.

3.5 Modelização da Célula de Montagem efectuada no *software* IGRIP

Tal como foi referido no Capítulo 3, uma das fases do desenvolvimento da simulação de uma célula robotizada é a modelização dos equipamentos constituintes dessa célula, bem como a modelização da célula real.

Também neste caso isso foi realizado, tendo-se efectuado a modelização geométrica de todos os equipamentos da Célula de Montagem do CCP (ver Figura 5.16), assim como a modelização cinemática dos equipamentos que apresentam movimento.

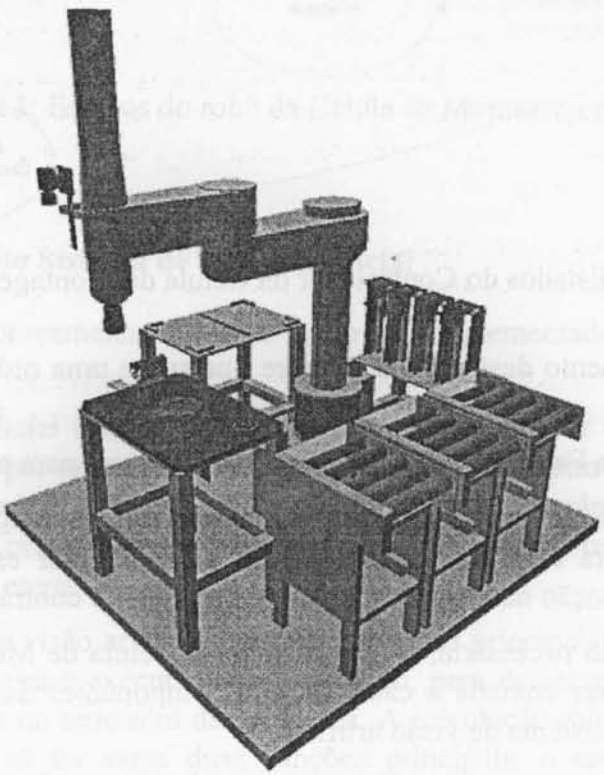


Figura 5.16: Imagem da Célula de Montagem do CCP modelizada no IGRIP

Em relação à modelização geométrica, esta foi realizada utilizando modelos de sólidos, uma vez o IGRIP suporta esta técnica de modelização e é a que apresenta melhores características gráficas e de visualização. Mesmo assim, dado o "peso" atingido pela simulação, foi necessário simplificar alguns pormenores sem relevância para o objectivo pretendido (a programação *off-line* do robô de montagem).

De entre estes, gostaria de salientar a simplificação da geometria das alhetas do motor que o robô Adept apresenta na ligação 3, que no modelo aparece como a conjugação de um cilindro e de um cone. Esta simplificação, que não apresenta implicações para a programação *off-line*, permitiu tornar a simulação muito mais rápida, apresentando como contrapartida uma representação menos fiel do robô Adept. No entanto e por comparação com a Figura 5.5, podemos ver que a célula modelizada representa, de uma forma muito aproximada, a célula real.

Quanto à cinemática deste dispositivo, ela foi herdada de outro robô SCARA da Adept, disponível na biblioteca de robôs fornecida com o *software*.

Em relação aos outros dispositivos, foi desenvolvido o seu modelo geométrico e cinemático, tendo sempre em atenção a simplificação do modelo geométrico, desde que isso não implicasse perda de precisão para o objectivo pretendido.

Não foi no entanto realizada qualquer modelização dinâmica dos equipamentos, uma vez que esta não é considerada necessária para a simulação desta célula e para a programação *off-line* do robô Adept Three afecto à Célula de Montagem.

De referir que na modelização da Célula de Montagem efectuada no *software* de simulação optou-se por uma modelização modular. Todos os componentes foram modelizados como se se tratassem de componentes claramente independentes, o que facilita a substituição de uns componentes por outros, com funções equivalentes, nos casos em que é necessário efectuar testes com diferentes componentes (por exemplo, para escolha das garras mais indicadas para uma dada operação de montagem).

Relativamente às diferentes garras (SHUNK) disponíveis no CCP, foi desenvolvida uma biblioteca, abrangendo também os respectivos dedos de sujeição entretanto produzidos no CCP.

Devido à modelização modular adoptada, passou a ser extremamente fácil, a um operador, testar diferentes garras (ou a mesma garra com diferentes dedos) para determinar qual a mais indicada para uma dada operação de montagem.

Capítulo 6

Especificação da solução adoptada

Após termos visto quais os passos para desenvolver um programa *off-line* para um robô afecto a tarefas de montagem, utilizando um *software* de simulação, e após se ter definido o modo de funcionamento da Célula de Montagem do CCP, neste capítulo procura-se dar resposta às seguintes questões:

- Qual a solução proposta para resolver o problema que se nos colocava;
- Que operações de montagem foram implementadas;
- Que informação deve ser trocada entre a área de Projecto (CAD) e o Controlador de Oficina (SFC), e entre este e o Controlador da Célula de Montagem (CCM).

Vimos, no Capítulo 1, que os métodos de programação de robôs deveriam ser:

- capazes de suportar a programação *off-line* do robô, de forma a que, para efectuar a sua programação se evite o mais possível o recurso à célula em que o mesmo se integra.
- orientados ao problema, isto é, o método de programação deve ser adaptado à imaginação do utilizador e à aplicação particular;
- suficientemente simples para serem facilmente apreendidos e manipulados.

Tendo exactamente estes objectivos em mente, foi pelo autor especificada a solução a adoptar para o caso da programação do robô de montagem existente no CCP.

Utilizou-se um pacote de *software* de simulação para efectuar a programação *off-line* do robô, sendo a programação baseada num conjunto de rotinas desenvolvidas com base nas operações de montagem definidas no Capítulo 4. Desta forma conseguimos um sistema de programação do robô Adept existente na Célula de Montagem do CCP muito intuitivo, e por isso, de fácil aprendizagem.

Pretendeu-se, acima de tudo, uma solução que permitisse testar facilmente a exequibilidade da montagem, desenvolvida no sistema de CAD Pro/Engineer, pelo robô Adept existente na Célula de Montagem do CCP, de forma a diminuir ao máximo o tempo gasto com as tarefas afectas ao projecto da montagem.

Após se ter concluído o projecto do produto, tendo a certeza que o robô Adept é capaz de efectuar a sua montagem, o programa para este robô surge de uma forma natural, sem existir a necessidade de esforço adicional.

Esta solução resulta do trabalho realizado durante o Projecto ESPRIT 5629, como tentativa de ultrapassar algumas das dificuldades detectadas na metodologia implementada durante esse Projecto para a realização da programação *off-line* do robô de montagem.

1 Especificação do problema

O problema que se pretendia resolver era o da programação de um robô de montagem Adept Three, existente na Célula de Montagem do CCP. As características principais deste robô, bem como a constituição e funcionamento da referida Célula de Montagem, foram descritas no capítulo anterior.

Esta célula, ou mais concretamente o robô Adept, deveria efectuar a montagem das peças PD001 e PD002 (descritas no Anexo A), para o que dispunha de um armazém com capacidade para quatro garras, das quais uma tinha por função a manipulação e montagem dos O-Rings e, outra, tinha por função realizar o aparafusamento da pega à tampa da peça PD001.

Para auxiliar na tarefa de montagem, estavam disponíveis, sobre a mesa de operações, duas fixações auxiliares cuja função era a fixação de alguns componentes numa posição estável durante a realização das operações de montagem.

2 Solução adoptada

A solução adoptada passa pela realização das tarefas de planeamento da montagem de uma forma iterativa, recorrendo às ferramentas do sistema de CAD existente no CCP e à simulação do processo de montagem no IGRIP. Pensamos desta forma conseguir um processo de desenvolvimento da solução mais rápido, baseando-se o projecto dos componentes da montagem na aplicação das técnicas de DFA e Engenharia Concorrente.

Quanto à programação do robô propriamente dita, optou-se pelo recurso à definição de um conjunto de rotinas, cada uma delas representando uma operação de montagem, de forma a facilitar eventuais tarefas futuras de reprogramação do robô. Estas rotinas são bastante genéricas e podem ser aplicáveis em montagens com características bastante diferentes. Dentro das rotinas é executado um conjunto de testes permitindo facilmente detectar se os componentes da montagem são correctamente agarrados com as garras pretendidas e se existem colisões durante o seu transporte e montagem.

As rotinas acabadas de referir recebem como parâmetros de entrada os códigos das peças a manipular e montar, de forma a permitir que na comunicação entre o robô de montagem e o

Sistema de Visão Artificial e Controlador da Célula de Montagem estes se entendam em relação ao componente que está a sofrer a operação de montagem. Recebem também como entrada as coordenadas locais das localizações onde devem ser realizadas as operações.

O facto de se ter optado por trabalhar com coordenadas locais aumentou a flexibilidade do processo de montagem e evita a necessidade de se proceder à calibração da célula de montagem de cada vez que se lhe adicionam equipamentos auxiliares (por exemplo, quando se muda o produto a montar).

O desenvolvimento dos programas para os equipamentos da célula de montagem, desde o projecto do produto e dos seus componentes, até à descarga dos programas para os equipamentos da célula real, implica uma colaboração estreita entre as diferentes áreas do departamento de projecto (área de CAD e área de simulação). O diagrama apresentado na Figura 6.1 descreve o fluxo de informação durante o processo de desenvolvimento.

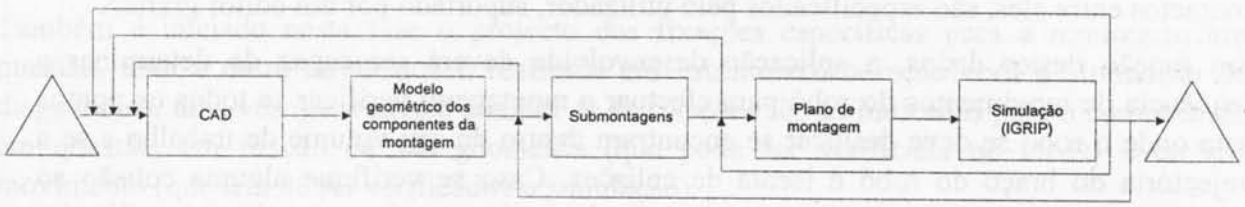


Figura 6.1: Desenvolvimento dos programas para os equipamentos da célula de montagem (Adaptado de [Guedes, 1995b])

A solução que foi adoptada parte dos pressupostos que a seguir se apresentam.

a) O desenvolvimento do produto, bem como o projecto de todos os seus componentes elementares, é realizado no sistema de CAD Pro/Engineer (PTC) existente no CCP.

b) Uma parte integrante do projecto de qualquer produto que tenha que sofrer operações de montagem é a fase de testes de montagem. No caso concreto em apreço, esses testes são efectuados recorrendo ao módulo de montagem Pro/Assembly existente no Pro/Engineer.

c) Após o projecto ter sido dado por concluído (incluindo a realização dos testes de montagem), exporta-se a informação relativa à geometria dos componentes da montagem para a área de simulação.

d) Aqui, são adicionados os referenciais de manipulação e de montagem. Idealmente, estes referenciais de manipulação e montagem deveriam ser incluídos nos componentes da montagem no sistema de CAD e serem exportados conjuntamente com o modelo geométrico. No entanto, uma vez que o único interface de troca de dados entre o CAD e a simulação, suportado por ambos os produtos, é o formato *Render* do Pro/Engineer (que só suporta dados geométricos), só é possível adicionar estes referenciais no *software* de simulação.

   Tais referenciais são a base de toda a programação e simulação: a todos os componentes de uma determinada montagem têm de ser atribuídos dois referenciais: um é o ponto por onde

a garra do robô o vai manusear; o outro é a base desse componente, ou seja, a sua origem espacial.

Todos os componentes que vão ter algum outro ligado a si têm que receber um referencial de montagem no local onde vai ser efectuada a ligação. Este é o referencial de montagem. Este referencial vai ser tornado coincidente com a base do componente a ligar durante o processo de montagem.

e) Depois desta tarefa estar concluída, o operador inicia a tarefa de programação do robô de montagem, escolhendo, de uma forma interactiva: o componente que quer montar; com que garra o vai manipular; por onde quer pegar nesse componente (referencial de manipulação) e onde o quer colocar (referencial de montagem). Resta acrescentar a esta informação a operação de montagem necessária para efectuar a ligação dos dois componentes.

f) Os referenciais por onde os objectos vão ser manipulados (referencial de manipulação), bem como as posições-objectivo de todos os componentes (referencial de montagem) e os contactos entre eles, são especificados pelo utilizador, suportado por um editor gráfico.

g) Em função destes dados, a aplicação desenvolvida deverá ser capaz de determinar a sequência de movimentos do robô para efectuar a montagem, verificar se todos os pontos para onde o robô se deve deslocar se encontram dentro do seu volume de trabalho e se a trajectória do braço do robô é isenta de colisões. Caso se verifique alguma colisão ao efectuar a montagem de um componente, a simulação deve ser interrompida e o utilizador deve ter conhecimento do local onde se verificou tal colisão. Neste caso, deve ser o utilizador a definir qual a trajectória precisa que o robô deve descrever para efectuar a montagem do componente que provoca a colisão.

h) A simulação também deve ser interrompida se o robô não conseguir alcançar alguma posição, devendo o operador, tal como no caso anterior, tomar medidas para ultrapassar este problema.

3 Desenvolvimento de uma montagem

3.1 Projecto dos componentes da montagem

O desenvolvimento de uma montagem inicia-se pelo projecto dos componentes da montagem, sendo esta tarefa realizada, hoje em dia quase exclusivamente, recorrendo a um *software* de CAD (como se disse, no CCP encontra-se disponível o pacote Pro/Engineer, da PTC).

O projecto para a montagem robotizada é um processo complexo e iterativo, uma vez que a forma de um objecto é essencial para a montagem automática. A necessidade de existência de pontos de manipulação bem definidos, especialmente para componentes complexos, introduz problemas quer para as rotinas do sistema de visão artificial, quer para as operações de manuseamento. A definição destes pontos é realizada durante a fase de projecto dos componentes.

Como se calcula, não é fácil criar restrições aos pontos de manuseamento, impondo alterações físicas aos componentes. Ainda mais difícil, ou mesmo impossível para montagens constituídas por um grande número de componentes, é dispor de uma garra para a manipulação de cada componente diferente. No entanto, o facto de as montagens realizadas serem efectuadas, na sua grande maioria, nas direcções vertical ou horizontal (por imposição do tipo de robô utilizado) reduz as manipulações dos componentes e, desta forma, os pontos de manipulação são mais facilmente definidos.

Mesmo assim, é muitas vezes necessário refazer o projecto de alguns componentes após ter sido efectuada a simulação da montagem no *software* de simulação. Esta necessidade pode surgir devido ao facto de as garras disponíveis não conseguirem manipular um dado componente da montagem, ou porque durante a montagem de um outro componente existe uma colisão da garra com os componentes que já se encontram na sua posição final. Na sua esmagadora maioria, estes problemas só são detectados durante a fase de simulação da execução da tarefa de montagem.

Também é iniciado nesta fase o projecto das fixações específicas para a montagem em questão. Esta é outra tarefa que é realizada em estreita colaboração com a simulação de dispositivos, uma vez que convém analisar se as fixações são as indicadas para o componente em questão, em função da sua geometria (que pode ser verificada no CAD) e do seu movimento (que tem de ser verificado na simulação).

Também são realizados nesta fase os testes de montagem, para ver se os componentes da montagem encaixam correctamente uns nos outros, tal como pretendido.

3.2 Divisão das montagens em submontagens

A divisão das montagens em submontagens é um problema bastante complexo. No caso do CCP, este é mais um problema de projecto do que propriamente um problema de montagem, sendo fundamental para o correcto funcionamento da própria Oficina Piloto.

Durante o processo de subdivisão da montagem devem ser respeitadas as seguintes regras empíricas: deve ser dado privilégio às montagens verticais; as possíveis submontagens devem ser constituídas pelo menor número de componentes possível; o desenvolvimento de novas garras e fixações deve ser reduzido ao mínimo exequível.

Duas das maiores limitações colocadas pela Célula de Montagem do CCP são:

- o respectivo armazém de garras só dispõe de quatro posições;
- só existem três mesas de transferência para efectuar a entrada/saída de materiais na célula.

Em relação ao primeiro ponto, se uma das garras tiver de ser especial (por exemplo, um aparafusador pneumático), só restam três (ou menos) posições no armazém para outras tantas garras diferentes, cada uma capaz de manipular um reduzido número de distintas formas geométricas. Daqui resulta que os componentes da montagem não podem ser muito diferentes, caso contrário ter-se-á que recorrer a submontagens.

O outro problema é ainda mais complexo. Se, por exemplo, uma montagem for constituída por mais de três tipos de componentes, que chegam à célula em diferentes contentores, o AGV tem que providenciar a troca dos contentores numa (ou mais) das mesas de transferência. Uma vez que o AGV também efectua o transporte de materiais para as outras células, isto pode sobrecarregar os transportes, causando atrasos em toda a Oficina. Outra solução, esta para montagens que compreendam um pequeno número de componentes (restrição imposta pelo tamanho físico dos contentores que o AGV pode transportar), passa pela sua produção em *kits*, situação na qual todos os componentes de uma montagem dão entrada na Célula de Montagem num único contentor, saindo a montagem final nesse mesmo contentor.

A divisão das montagens em submontagens, que são reenviadas para o armazém após a sua realização, é uma das soluções possíveis para estes problemas. Assim, no primeiro caso, poderia ser efectuado um *setup* à Célula de Montagem, enquanto as submontagens se encontravam no armazém, constituindo esse *setup* da troca das garras existentes no armazém de garras desta célula.

Finalmente, convém realçar que a realização de operações sucessivas do robô com a mesma garra permite consideráveis poupanças de tempo. Tal não se consegue normalmente incorporar no projecto da montagem, uma vez que geralmente diferentes componentes requerem diferentes garras, mas é um tópico importante na escolha das sequências de montagem.

3.3 Planeamento da montagem

Uma vez que o ambiente na Célula de Montagem é muito estruturado, não é necessário planear *on-line*. Uma outra razão para o planeamento da montagem ser realizado *off-line* resulta do facto de a Célula de Montagem do CCP não dispor de sensorização (excepção feita à câmara do Sistema de Visão Artificial), pelo que não é possível determinar o estado actual do sistema de forma a tentar planear *on-line*.

O primeiro componente da montagem é colocado numa superfície paralela ao plano de movimento do robô e é fixado nessa posição por um dispositivo (ou conjunto de dispositivos) dedicado (uma fixação). Esta imposição torna o processo de planeamento da montagem mais fácil. Tal como já foi visto no capítulo anterior, o robô existente na Célula de Montagem do CCP é um robô SCARA com quatro graus de liberdade. Uma vez que este tipo de robôs possui uma certa flexibilidade de alinhamento coaxial no plano horizontal e uma elevada rigidez nas outras direcções, as montagens efectuadas segundo a direcção de junção vertical são favorecidas.

No entanto, apesar de esta táctica de montagem ser a mais indicada para este robô, ele pode também realizar montagens cuja direcção de junção dos componentes seja horizontal e, mais dificilmente, realizar montagens em que essa direcção seja oblíqua, uma vez que o robô possui um controlo de movimento em trajectória contínua.

Durante o planeamento da montagem e devido às características deste robô, também não é possível modificar o estado estável dos componentes a montar (por exemplo, deitar os

componentes e pegar neles noutra posição), pelo que os mesmos têm que ser fornecidos ao robô no estado (estável) em que vão ser montados.

Tendo em conta estas características, um plano de montagem é desenvolvido pelo operador de forma a conseguir alcançar, da forma o mais eficiente possível, a montagem pretendida. Aqui, a simulação deve oferecer a possibilidade de facilmente experimentar múltiplos planos de montagem, verificando se o robô os consegue cumprir (alcance do robô ao longo dos vários pontos da sua trajectória e colisões) e determinar qual o tempo de execução envolvido em cada plano alternativo testado.

3.4 Exportação dos dados

Os dados referentes à geometria da montagem são exportados em formato *Render*, formato este proprietário do *software* de CAD Pro/Engineer, para o qual o *software* IGRIP dispõe de um pós-processador.

Os ficheiros que contêm o modelo geométrico de cada um dos componentes da montagem, desenvolvidos no CAD, são gravados numa directoria (directoria - *directory* - `/design`), no disco da estação de trabalho em que se encontra instalado o *software* de CAD (Sun SPARC20). Esta directoria encontra-se exportada, estando montada como directoria remota na estação de trabalho onde se encontra instalado o *software* de simulação (Silicon Graphics Indigo II).

Desta forma, sempre que é necessário desenvolver o programa *off-line* para a montagem de um produto, o operador do *software* de simulação recebe a indicação do produto sobre o qual o trabalho vai ser desenvolvido e procura, dentro da directoria `/design`, a directoria que corresponde ao nome do produto em questão.

Dentro desta directoria encontram-se quer os modelos geométricos de todos os componentes necessários para a montagem do produto em questão, quer um modelo geométrico explodido da mesma, de forma a que o operador do *software* de simulação perceba como deverá ser efectuada a montagem.

4 Tarefas a realizar no sector de simulação

4.1 Importação dos dados

Como se referiu no ponto anterior, para efectuar a simulação no *software* IGRIP, importam-se do CAD todos os modelos geométricos sólidos dos componentes da montagem.

Seguidamente, é geralmente necessário proceder-se (no *software* de simulação) à simplificação dessas geometrias, de forma a que os modelos geométricos não se tornem muito pesados durante o processo de animação gráfica. Esta tarefa é realizada normalmente pelo operador responsável pela tarefa de simulação e programação *off-line* do robô de montagem.

Após estas tarefas o operador deve acrescentar ao modelo geométrico dos componentes da montagem os respectivos referenciais de montagem e de manipulação. Optamos por definir os referenciais de manipulação e de montagem como sendo sistemas de referência em coordenadas locais.

4.2 Desenvolvimento do programa

De forma a diminuir a complexidade da tarefa de programação do robô e a facilitar a tarefa de teste da montagem robotizada, tornando ao mesmo tempo o interface mais amigável e intuitivo para o operador, optamos por definir um grupo de operações básicas de montagem, correspondendo cada uma destas operações a uma macro-instrução de programação, ou se preferirmos, a uma rotina de programação. As operações básicas de montagem implementadas foram inspiradas, em larga medida, nas operações em que Rampersad (1994) dividiu as tarefas de montagem (ver Capítulo 4, secção 1).

Utilizando esta filosofia de programação dos robôs de montagem, um programa de montagem é uma sequência de chamadas a estas rotinas. Funções auxiliares, tais como a troca de garras, as tarefas de inspecção com recurso ao Sistema de Visão Artificial, a abertura e o fecho dos actuadores presentes sobre a mesa de operações e mesmo a abertura e o fecho das garras do robô, são chamadas dentro destas rotinas, não tendo o programador do robô que se preocupar com tais problemas: tem unicamente que invocar as operações necessárias para realizar cada operação de montagem.

4.2.1 Operações de montagem implementadas

Uma vez que para efectuar a programação do robô se adoptou a solução de uma programação baseada em macro-instruções, colocou-se o problema de escolher as que deveriam ser implementadas (sendo cada macro-instrução correspondente a uma operação de montagem).

Para efectuar tal escolha baseámo-nos nas necessidades concretas colocadas pela montagem das peças PD001 e PD002 (apresentadas no Anexo A), bem como das outras peças já produzidas na Oficina Piloto do CCP. Como pode ser visto no Anexo A, concretamente para as peças PD001 e PD002, a sua montagem consta de um conjunto de encaixes de uns componentes noutros. A estas operações de encaixe deve ser acrescentada uma operação que permita a inserção de um O-Ring num dos componentes (BASE), de forma a que o componente que vai ser montado sobre este (CORPO) fique com um certo aperto, mas apresentando a possibilidade de ser desmontado. É também necessária uma operação que permita efectuar o aparafusamento do componente PEGA ao componente TAMPA, no caso da peça PD001. Adicionalmente, é necessária a implementação de instruções de manipulação, de forma a que o robô possa manipular todos os componentes das várias peças.

Daqui resultou a implementação das seguintes operações de montagem:

- pegar;
- pousar;

- juntar;
- inserir;
- aparafusar;
- montar O-Ring.

Apesar de as operações de montagem implementadas serem as necessárias para a programação do robô para o caso das peças PD001 e PD002, o trabalho desenvolvido reveste-se de alguma generalidade, estando por isso facilitado o desenvolvimento futuro de novos programas para este robô efectuar outras tarefas de montagem.

4.3 Especificação das operações implementadas

4.3.1 Operações de Manuseamento

Grupo de operações para movimentar as peças dentro da Célula de Montagem. Todas as outras operações recorrem, directa ou indirectamente, a estas operações.

4.3.1.1 Operação Pegar

Operação para pegar em componentes da montagem ou nos produtos montados. Estes podem estar numa das mesas da Célula de Montagem ou dentro de um contentor, colocado numa das mesas de transferência.

A sequência de movimentos que o robô necessita de efectuar para executar uma operação Pegar é a que se apresenta na Figura 6.2.

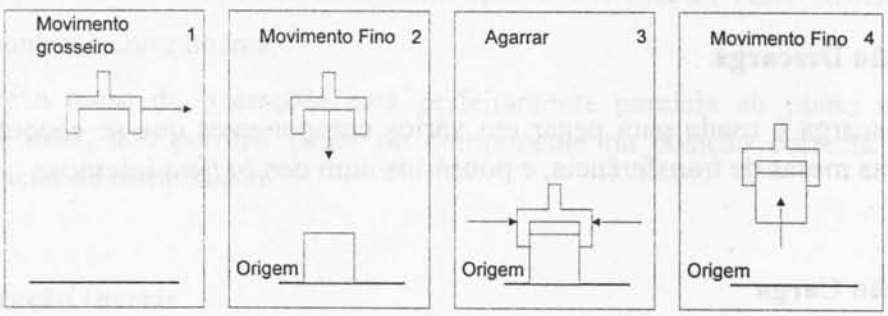


Figura 6.2: Sequência de movimentos da operação Pegar

4.3.1.2 Operação Pousar

Operação usada para pousar componentes da montagem ou produtos montados sobre uma das mesas da Célula de Montagem ou dentro de um contentor, colocado numa das mesas de transferência. Daí que o robô tenha que efectuar um movimento de aproximação ao ponto em que vai pousar o componente, depois efectuar um movimento fino para essa localização e, finalmente, abrir a garra (ver Figura 6.3).

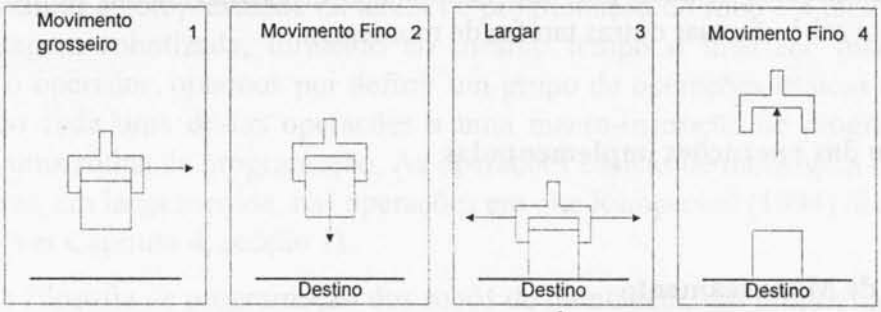


Figura 6.3: Sequência de movimentos da operação Pousar

4.3.1.3 Operações de Armazenamento

Estas operações podem ser vistas como um subgrupo das operações de manuseamento. São usadas para descarga e carga de um contentor que se encontre numa das mesas de transferência, recorrendo extensivamente às operações de manuseamento propriamente ditas, sendo mesmo traduzidas para a linguagem do robô como uma sequência de instruções Pegar e Pousar. Após uma destas operações, o contentor é sempre retirado da Célula de Montagem pelo AGV.

4.3.1.3.1 Operação Descarga

A operação de descarga é usada para pegar em vários componentes que se encontram num contentor, numa das mesas de transferência, e pousá-los num dos *buffers* internos.

4.3.1.3.2 Operação Carga

A operação de carga, tal como o seu nome indica, tem a função inversa da operação anterior. O seu objectivo é pegar num conjunto de peças, que se encontram num dos *buffers* internos, e colocá-las num contentor, numa das mesas de transferência.

4.3.2 Operações de Composição

As operações deste grupo são as usadas no processo de montagem propriamente dito, estando directamente relacionadas com a operação Pousar. Para uma destas operações ser realizada, o robô necessita de ter previamente apanhado o componente sobre o qual vai ser realizada uma destas operações.

4.3.2.1 Operação Juntar

Operação de composição cuja função, tal como o seu nome indica, é a junção de dois componentes (ver Figura 6.4): no início desta operação um desses componentes encontra-se na sua posição final.

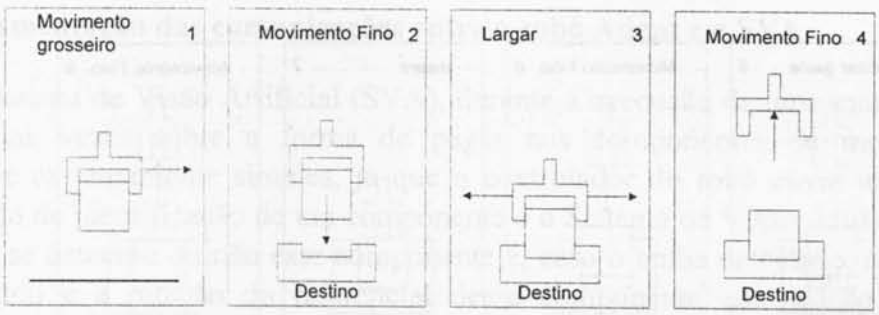


Figura 6.4: Sequência de movimentos da operação Juntar

Quer esta operação, quer a seguinte (operação Inserir), têm que:

- pousar o componente na mesa de operações;
- se for necessária uma identificação do componente ou uma verificação da sua orientação, estas operações fazem um pedido de identificação ao Sistema de Visão Artificial;
- voltar a apanhar o componente.

Uma vez que a mesa de operações está perfeitamente paralela ao plano de movimento horizontal do robô, isto permite pegar no componente na posição correcta para executar depois a operação de composição.

4.3.2.2 Operação Inserir

Operação utilizada para a inserção parcial ou total de um componente dentro de outro ou numa submontagem; só existe um grau de liberdade para a trajectória do robô durante a execução desta operação.

Apresenta-se na Figura 6.5 a sequência de movimentos que o robô necessita de efectuar para executar uma operação Inserir.

4.3.2.3 Operação Aparafusar

Esta operação de montagem é responsável pela montagem de parafusos (aparafusamento), sendo muito semelhante à execução sequencial das operações Pegar e Pousar tirando alguns pormenores que serão analisados no capítulo seguinte.

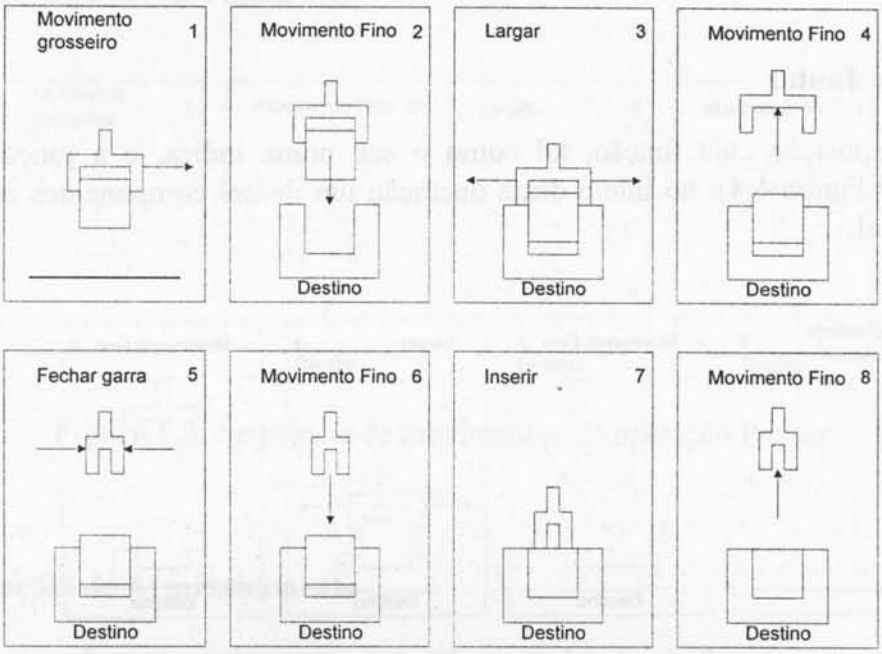


Figura 6.5: Sequência de movimentos da operação Inserir

4.3.2.4 Operação Montar O-Ring

Esta operação serve para efectuar a operação de montagem de O-Rings. Ela é responsável pela inserção dos O-Rings nas peças cilíndricas, como forma de ajuste da montagem de algumas peças.

4.3.3 Operações auxiliares

4.3.3.1 Operação de troca de garras

Esta operação implementa a troca de garras do robô, sendo responsável por pousar a garra que o robô tem no punho e ir buscar a garra pretendida, que se encontra no armazém de garras.

4.3.3.2 Operação de abertura/fecho das garras

Para que o robô possa agarrar nos objectos que tem que manipular é necessária a implementação de operações de abertura e fecho das garras. Estas operações podem também ter a função de activar/desactivar as funcionalidades dos actuadores finais com que o robô se encontra equipado, caso por exemplo da chave aparafusadora que o robô tem que manipular de forma a executar as operações de aparafusamento.

4.3.3.3 Operação de activação das fixações

A operação de activação das fixações resume-se a activar os sinais digitais correctos para fechar os actuadores pneumáticos que se encontram na mesa de operações.

4.3.3.4 Implementação das comunicações entre o robô Adept e o SVA

No caso do Sistema de Visão Artificial (SVA), durante a execução de uma montagem o robô inquire-o várias vezes sobre a forma de pegar nos componentes da montagem. Esta comunicação é extremamente simples, já que o controlador do robô envia uma mensagem com um pedido de identificação de um componente e o Sistema de Visão Artificial responde-lhe indicando se detectou ou não esse componente e, caso o tenha detectado, as coordenadas onde o detectou e a rotação do referencial desse componente em relação à origem de referência.

Sendo assim, esta operação é a responsável por inquirir o sistema de visão artificial, para que ele tente identificar um determinado componente de uma montagem ou verificar a sua posição e orientação.

Para simular a troca de informação entre o robô de montagem e o Sistema de Visão Artificial a ele associado, recorreu-se à implementação de uma rotina cujos parâmetros de entrada são os que o robô envia ao Sistema de Visão Artificial sempre que necessita de um serviço deste último; essa rotina faz o retorno da informação (parâmetros de saída) da mesma forma que o faria o Sistema de Visão Artificial.

Uma vez que esta rotina é depois executada dentro de uma operação de montagem, é transparente para o utilizador o seu funcionamento durante a simulação.

Consoante os parâmetros passados pelo controlador do robô ao Sistema de Visão Artificial, apresenta duas funções alternativas:

- identificação;
- localização.

A primeira destas operações retorna se detectou algum objecto no seu campo de visão e, caso o tenha conseguido identificar, de que componente se trata.

A segunda operação retorna a localização do componente identificado no referencial próprio da câmara, isto é, a posição do objecto é retornada em coordenadas da câmara do Sistema de Visão Artificial.

4.3.3.5 Operação de ajuste do Sistema de Visão Artificial

Esta operação auxiliar calcula os ângulos de cada junta do robô, de forma a que ele possa ficar com a câmara sobre a peça a detectar ou a identificar, e depois dá ordem ao robô para ele se deslocar para esse ponto. Os cálculos são efectuados iterativamente, tendo como ponto de partida o ponto onde o robô pousou a peça a detectar ou a identificar.

4.3.3.6 Operação de ajuste do ponto de pega

Esta operação de montagem é responsável por colocar a junta 4 do robô na posição correcta para apanhar as peças em função da rotação que elas apresentam, e que é a retornada pelo sistema de visão artificial.

4.3.3.7 Operação indicadora do fim da montagem

Esta operação é responsável por informar o Controlador da Célula de Montagem que a operação de montagem que estava a decorrer terminou com sucesso.

4.4 Verificação da exequibilidade de manipulação dos componentes da montagem

Esta tarefa destinada à verificação da adequação da garra escolhida pode ser efectuada recorrendo à análise de colisões.

Se uma garra aberta se aproxima de um objecto, sem que haja colisão entre a garra e esse objecto até ser dado o sinal para fechar a garra, passando depois a haver colisão entre ambos, isso significa que a garra em questão pode manipular o objecto, uma vez que esta situação significa que a garra o está a segurar.

Caso a garra nunca colida com o objecto, nem quando está aberta, nem quando está fechada, isso significa que essa garra não é a indicada para manipular o objecto em questão, uma vez que quando fechada não consegue agarrá-lo.

No extremo oposto situa-se a situação em que a garra não consegue apanhar o objecto por não ter abertura suficiente. Neste caso existe colisão entre a garra e o objecto, quer com a garra aberta, quer fechada.

Resta uma última situação, que é aquela em que a garra, quando aberta, colide com o objecto, mas depois de fechada deixa de colidir com ele. Isto significa que a garra em questão não é a indicada para manipular o objecto, verificando-se geralmente o caso de este apresentar uma cavidade no seu interior, devendo por isso ser manipulado por uma garra que se aproxime, no

estado de fechada, até à posição em que deve agarrar o objecto e, só aí, abrir os seus dedos. Outra solução será tentar apanhar o objecto recorrendo a uma garra com maior amplitude de movimento (vulgo, maior abertura).

4.5 Verificação da exequibilidade das trajectórias do robô

Para a realização de todas as trajectórias é necessário verificar a exequibilidade dos movimentos, quer no aspecto do alcance do robô, quer no aspecto de eventuais colisões entre o robô, ou o objecto transportado, e os outros equipamentos da Célula de Montagem.

O primeiro aspecto pode ser realizado no *software* IGRIP durante a programação *off-line* do robô. Isto é efectuado verificando se todos os pontos dessa trajectória do robô se mantêm dentro dos limites do volume de trabalho do robô em questão.

Outro parâmetro importante que merece consideração durante o planeamento das trajectórias são as colisões. Se este aspecto não for considerado podem ocorrer colisões durante a execução das tarefas inerentes à montagem.

É necessário considerar as colisões entre o robô, o seu actuador final e o objecto que está a ser manipulado, com os diferentes componentes da Célula de Montagem. A existência do modelo da Célula de Montagem no *software* IGRIP simplifica esta verificação. Aqui, com o modelo da célula real, podem-se analisar as diferentes trajectórias para uma sequência de montagem.

As possíveis colisões podem ocorrer entre o robô (principalmente o seu actuador final) ou entre os componentes da montagem (durante as operações de manipulação e montagem) e os da célula.

Como vimos no Capítulo 3 existem dois tipos de colisões: as colisões "garra-objecto" e as colisões "objecto-objecto".

Para detectar as colisões "garra-objecto" há que proceder à detecção de colisões sempre que o robô vai apanhar ou pousar um objecto. Por outro lado, e assumindo como também visto no Capítulo 3, que os movimentos grosseiros se efectuam a uma cota onde já não há perigo de colisão (o objecto transportado é elevado a uma altura tal que a sua base fica acima do topo de qualquer outro objecto) teremos apenas a possibilidade de colisões "objecto-objecto durante os dois movimentos verticais (correspondentes aos movimentos de interface e movimentos finos). Uma vez que também se pretende determinar a ocorrência deste tipo de colisões, há a necessidade de se efectuar a sua detecção durante o transporte dos objectos pelo robô.

5 Trocas de informação entre componentes do sistema

Do que vimos até aqui, neste capítulo, resulta claramente a noção de que existe um grande número de tarefas que devem ser realizadas no CAD e, as restantes, na simulação, sendo este um processo iterativo. Como resultado, um dos grandes problemas que se colocam durante o processo de montagem robotizada de uma peça é a troca de informação entre o sistema de CAD e o sistema de simulação, uma vez que existem múltiplas tarefas em cada um destes sistemas, e a informação de um deles, é absolutamente necessária para o outro.

Existe também a necessidade de enviar a informação respeitante aos programas de montagem para o Controlador da Célula de Montagem (e daqui para o robô Adept).

5.1 Informação trocada entre o sector de CAD e o sector de simulação

A informação proveniente do CAD é a respeitante à geometria dos diversos componentes que compõem a montagem, assim como a informação relativa às coordenadas específicas dos locais onde estes vão ser montados (referenciais de montagem).

A informação sobre o local por onde os componentes da montagem podem/devem ser manipulados (referenciais de manipulação) também faz parte do pacote fornecido pelo CAD à simulação, se bem que esta informação só seja definitivamente estabelecida durante a simulação da manipulação dos objectos.

5.2 Informação trocada entre o sector de simulação e o CCM

Para a realização de qualquer uma das operações de montagem, o controlador robô necessita de conhecer qual a trajectória do movimento que este deve efectuar e as localizações espaciais nas quais deve activar o actuador final. Esta informação encontra-se contida dentro do programa de montagem que é descarregado para o controlador do robô.

Necessita também de conhecer qual o actuador final necessário à realização de uma determinada operação (caso seja necessário algum) e qual deverá ser a função a activar, caso aquele possua mais do que uma função. Ainda em relação aos actuadores finais, é necessária a informação relativa ao TCP destes e à sua posição no armazém de garras (informação de *setup*).

Adicionalmente, é ainda necessária informação relativa aos actuadores externos e às fixações, cuja utilização poderá ser necessária para a montagem.

Toda esta informação é fornecida dentro do programa de montagem que, como se disse, é descarregado para o robô. O programa de montagem, no entanto, é gravado pelo *software* de simulação num *fileserver* e só é descarregado para o robô quando é dada a ordem, a este último, para iniciar a montagem.

Sintetizando, o Controlador da Célula de Montagem recebe do *software* de simulação a seguinte informação:

- programa de montagem;
- ficheiro com o TCP das garras (ferramentas) e suas posições no armazém de garras;
- ficheiro com a identificação dos componentes da montagem.

5.3 Informação trocada entre o SFC e o CCM

A estrutura das mensagens provenientes do Controlador de Oficina (SFC) é a seguinte:

tipo_mensagem	origem	destino	id_mensagem	id_diálogo	parâmetros

Os parâmetros que podem ser incluídos na mensagem são os seguintes:

tipo_base	id_contentor	posição	mesa	material_tipo

Antes de iniciar a montagem do produto pretendido, o Controlador da Célula de Montagem (CCM) deve receber do Controlador de Oficina uma ordem de montagem. Em função desta ordem de montagem é que o Controlador da Célula de Montagem descarrega o programa de montagem para o robô.

Adicionalmente, e uma vez que só quando um contentor chega à Célula de Montagem é que se sabe por que mesa é que este vai entrar na Célula de Montagem (esta informação é gerida pela Célula de Armazenamento e de Transporte de Materiais), esta informação é fornecida pelo Controlador de Oficina ao Controlador da Célula de Montagem, que posteriormente a passa ao respectivo robô. Assim, além da ordem de montagem, o Controlador da Célula de Montagem recebe uma mensagem do tipo *part_pos* para cada componente da montagem. O *part_pos* é um tipo de mensagem especifico, que identifica por que mesa de transferência entrou o contentor que traz o componente a montar. Quando esta informação é recebida, é gravada em disco de forma a que, quando o robô executar a operação de montagem correspondente a um dado componente, possa saber em que mesa de entrada (contentor) o mesmo se encontra.

Capítulo 7

Implementação da solução

Neste capítulo procura-se dar resposta às seguintes questões:

- Como foi implementada a solução adoptada;
- Como se efectua a integração e troca de informação;
- Que outros aspectos tiveram que ser solucionados;
- Que testes foram feitos para comprovar o funcionamento da solução desenvolvida.

Quando o Controlador da Célula de Montagem recebe uma ordem de montagem, descarrega para o robô Adept o programa correspondente. Por sua vez, este, à medida que o vai recebendo, grava-o num ficheiro em disco. Após todo o programa ter sido recebido, é carregado para a memória e executado.

O programa de montagem, que é gerado *off-line* e descarregado para o robô Adept quando este recebe uma ordem de montagem do Controlador da Célula de Montagem, consta de uma sequência de instruções de chamada a rotinas, cada uma delas representando uma operação de montagem (tais como pegar, pousar, inserir, etc.).

O recurso a rotinas apresenta vantagens sobre a programação explícita pura, no que diz respeito ao tempo de desenvolvimento de um novo programa para o robô (simplifica a programação); em contrapartida, é necessário um esforço acrescido na definição e desenvolvimento das rotinas.

De seguida passa-se a descrever a aplicação desenvolvida no IGRIP, *software* este utilizado para a simulação e programação *off-line* do robô Adept existente na Célula de Montagem do CCP. Descrevem-se também as rotinas implementadas para a correcta execução das tarefas de montagem, de forma a simplificar a tarefa de (re)programação do robô. Algumas destas rotinas implementam operações básicas de montagem robotizada, enquanto outras implementam funcionalidades próprias do controlador do robô. Todos os programas de montagem do robô se servem destas rotinas, ou pelo menos de parte delas.

1 Programação do robô de montagem

1.1 Definição dos referenciais de manipulação e montagem

Um dos problemas que foi necessário ultrapassar durante o desenvolvimento da solução foi o da inserção dos referenciais, para manipulação e montagem, nos modelos dos componentes da montagem importados do sistema de CAD.

Adoptamos a solução de utilizar sistemas de coordenadas para representar estes referenciais de manipulação e de montagem. Essa tarefa tem que ser realizada manualmente e o maior problema que se coloca é a definição da orientação dos sistemas de coordenadas.

No caso dos referenciais para a montagem, convencionou-se que os sistemas de coordenadas seriam colocados de forma a que o eixo Z definisse a forma como se iria proceder à inserção das peças a montar. Desta forma o utilizador escolhe o componente a montar e o componente no qual ele vai ser montado, tendo que indicar para este último qual o referencial onde o primeiro vai ficar após a montagem. Com estes dados, a aplicação desenvolvida determina automaticamente como é que o componente pode ser montado, de forma a que o seu referencial da base fique coincidente com o referencial de montagem do componente que o recebe (ver Figura 7.1) e quais são as coordenadas do ponto para onde o robô tem que se deslocar de forma a que, quando o componente for colocado no seu local, o robô esteja a pegar nele.

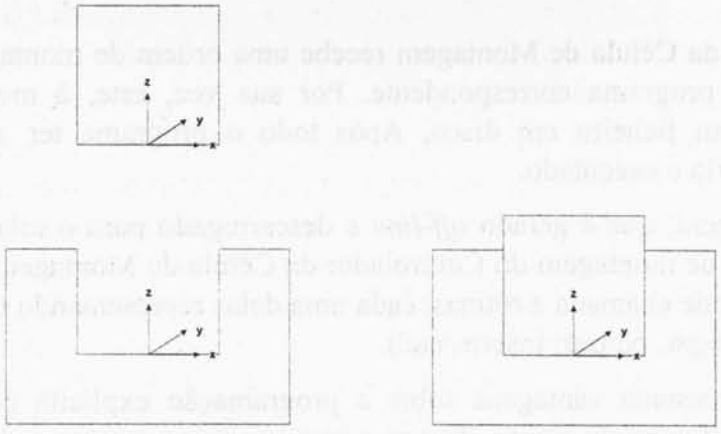


Figura 7.1: Referenciais de montagem

O mesmo se aplica para a manipulação de um determinado componente. Dado o seu referencial de manipulação e o do TCP da garra ou do actuador final do robô (indicado para a tarefa de manipulação do componente), a aplicação desenvolvida consegue determinar automaticamente as coordenadas do ponto para onde o robô tem que se deslocar, de forma a que o referencial do TCP da garra fique coincidente com o referencial de manipulação do componente a manipular (ver Figura 7.2). Este é o ponto para onde o robô se deve deslocar quando se pretende pegar, ou pousar, um determinado componente. Consegue-se igualmente

determinar a forma como a garra tem que efectuar a aproximação ou afastamento ao componente, a pegar ou largar respectivamente. Esta informação é extraída a partir da orientação do eixo Z do referencial de manipulação do componente.

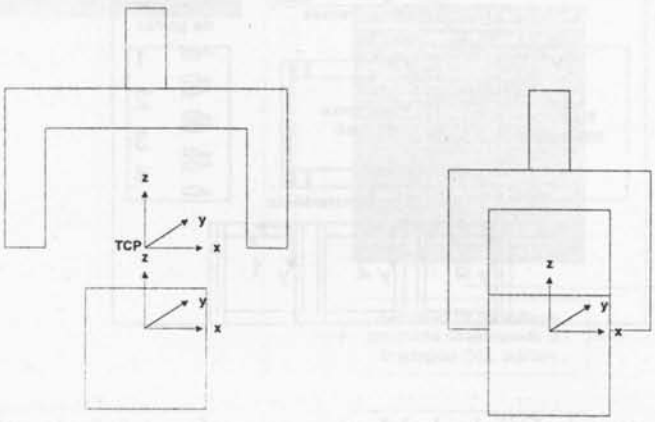


Figura 7.2: Referenciais de manipulação

1.2 Desenvolvimento do programa de montagem

As rotinas desenvolvidas para a programação *off-line* têm equivalência noutras (com o mesmo nome e função) que existem no robô real. Da primeira vez que um determinado robô é programado as rotinas existentes no robô real podem ser descarregadas a partir das suas equivalentes existentes no simulador.

O operador só necessita de conhecer as operações e a sequência com que é necessário executá-las para montar os diversos componentes da montagem, de forma a obter, como resultado, o produto final. Usando estas rotinas, o operador tem que as invocar pela ordem correcta, sendo os seus argumentos determinados automaticamente. Estes argumentos são, na sua maioria: as coordenadas locais das localizações onde o robô tem que executar as operações; os nomes dos componentes sobre os quais vão ser executadas essas operações; o nome do produto resultante da operação. De forma a que isto seja possível, todas as coordenadas das posições para onde o robô tem que se mover são relativas a sistemas de coordenadas de referência do utilizador, referenciais estes que foram definidos em locais estratégicos das diferentes mesas da Célula de Montagem sobre as quais têm que ser executadas as diversas operações (ver Figura 7.3).

Desta forma, quando os programas gerados no *software* de simulação são traduzidos para o robô, unicamente são descarregadas as instruções de chamada a essas rotinas e os respectivos argumentos.

Ainda graças à definição destes referenciais em locais estratégicos da Célula de Montagem, esta célula não necessitou de ser calibrada e, apesar deste facto, funciona de forma correcta, não havendo qualquer problema na descarga das localizações geradas *off-line* para o robô Adept.

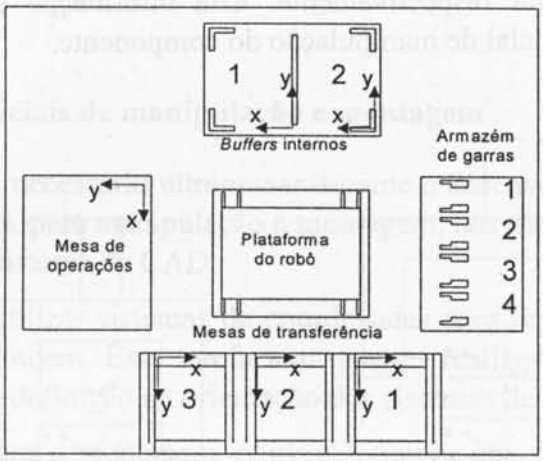


Figura 7.3: *Layout* da Célula de Montagem e referenciais do utilizador

Esta solução, além de tornar mais simples a simulação, a programação e a calibração do robô, também simplifica toda a (re)programação que seja necessária no futuro. Adicionalmente, a linha série de comunicação entre o robô Adept e o Controlador da Célula de Montagem não fica ocupada durante períodos de tempo muito longos, o que a acontecer, seria prejudicial, já que implicaria um estrangulamento nas comunicações. No entanto, existe um grande senão: estas rotinas levam mais tempo a programar, de forma a serem um espelho perfeito das rotinas que se encontram implementadas no robô real.

As operações que o robô necessita de implementar de forma a executar a montagem dos produtos (cada operação corresponde a uma rotina de montagem), foram divididas em vários grupos, de acordo com os resultados de cada uma dessas operações. Desta divisão resultou a organização que se descreve seguidamente.

2 Interface com o utilizador

Quando o utilizador entra na aplicação desenvolvida, surge-lhe um interface (ver Figura 7.4) que lhe permite múltiplas escolhas.

No menu correspondente, com o título "Programação Implícita Visual", o utilizador pode escolher entre efectuar o *setup* de equipamentos da Célula de Montagem, definir as operações necessárias para efectuar a montagem, gerar o programa correspondente às operações que o utilizador definiu como sendo as necessárias para efectuar a montagem ou, por último, traduzir o programa desenvolvido para a linguagem própria do robô Adept (V/V+). Neste menu o utilizador dispõe ainda de uma opção que lhe permite abandonar a aplicação.

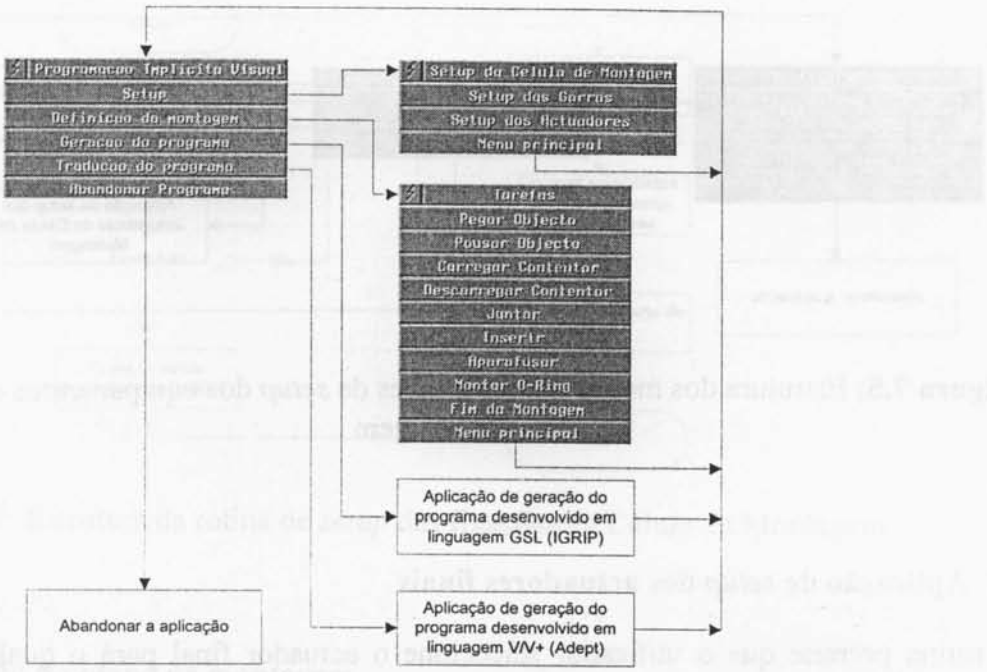


Figura 7.4: Estrutura dos menus da aplicação de programação do robô de montagem

2.1 *Setup* dos equipamentos da Célula de Montagem

Sempre que se inicia uma nova montagem é necessário efectuar algumas operações de *setup* na Célula de Montagem. Essas operações de *setup* incidem com maior frequência sobre as garras ou actuadores finais do robô, uma vez que cada montagem apresenta diferentes componentes, sendo necessárias garras com características diferentes para os manipular e, também, actuadores finais com funcionalidades diferentes em função das operações de montagem a realizar.

Pelos mesmos motivos, as fixações e os actuadores que, por exemplo, se encontram sobre a mesa de operações da Célula de Montagem, necessários para auxiliar o robô na execução das operações de montagem, na maioria das vezes têm de ser substituídos sempre que se inicia a montagem de um produto diferente.

Daí que se tenha implementado esta opção, que permite ao utilizador efectuar o *setup* de alguns parâmetros de funcionamento das garras e actuadores finais do robô de montagem e das fixações ou actuadores auxiliares existentes na Célula, como sejam os TCPs das garras e dos actuadores finais e os sinais digitais que, uns e outros, trocam com os controladores, quer do robô, quer da Célula de Montagem.

No caso de o utilizador escolher a opção de *setup* no menu "Programação Implícita Visual", salta para o menu "Setup da Célula de Montagem", que lhe permite efectuar o *setup* quer das garras do robô, quer dos actuadores da Célula de Montagem. A estrutura destes menus é a apresentada na Figura 7.5.

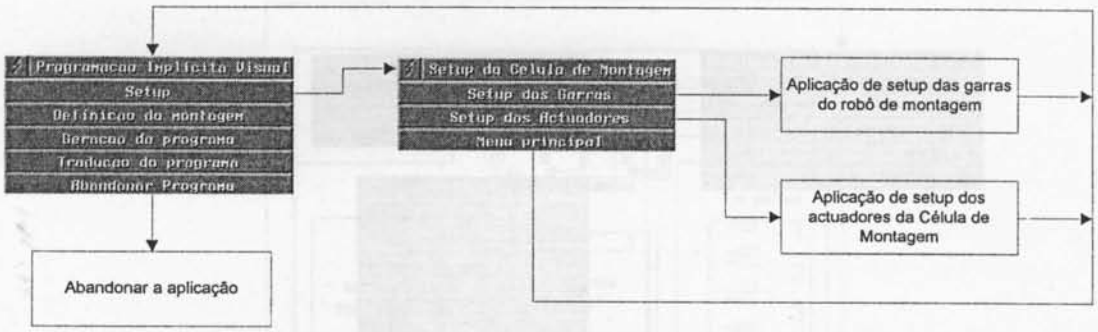


Figura 7.5: Estrutura dos menus das aplicações de *setup* dos equipamentos da Célula de Montagem

2.1.1 Aplicação de *setup* dos actuadores finais

Esta rotina permite que o utilizador seleccione o actuador final para o qual vai definir os parâmetros de *setup* e, logo de seguida, proceda à sua definição. Os valores dos parâmetros de *setup* (sinais digitais trocados entre o controlador do robô e o actuador final, e valores do TCP do actuador final) são escritos num ficheiro que contém esses dados, sendo este descarregado para o robô quando se inicia uma nova montagem.

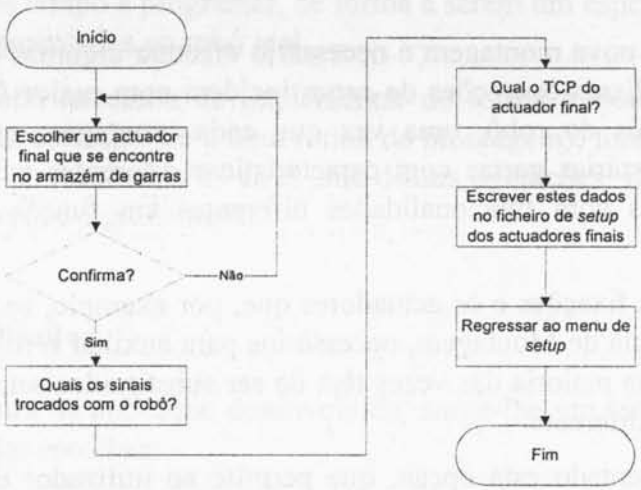


Figura 7.6: Estrutura da rotina de *setup* das garras do robô Adept

2.1.2 Aplicação de *setup* das fixações

O funcionamento desta rotina é exactamente igual ao da anterior, sendo neste caso definidos como parâmetros a inicializar, sempre que se muda para uma nova montagem, os sinais digitais que o actuador troca com o controlador do robô Adept e o número pelo qual é identificado este actuador.

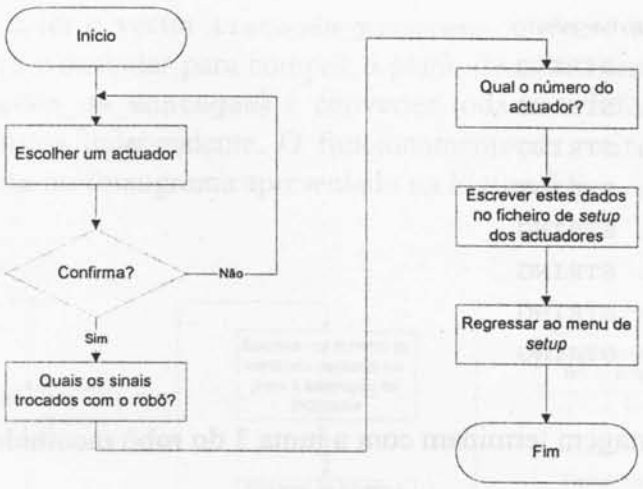


Figura 7.7: Estrutura da rotina de *setup* das fixações da Célula de Montagem

2.2 Implementação do plano de montagem

Caso o utilizador escolha a opção "Definição da Montagem" no menu "Programação Implícita Visual", salta para o menu "Tarefas", a partir do qual pode definir qual é a sequência de operações necessária para proceder à montagem do produto em questão (ver Figura 7.8).

Chegado a este menu, o operador pode programar a tarefa de montagem do robô de forma interactiva, seleccionando as operações que o robô deve executar para cumprir o plano de montagem definido em colaboração com o sector de CAD e respondendo às várias solicitações que o sistema que lhe vai colocando, por exemplo através da indicação (por meio da sua selecção com o rato) de componentes a manipular ou montar. Caso ocorram erros, o operador recebe no monitor as correspondentes mensagens de erro, já tratadas e com possíveis causas para a sua ocorrência, de forma a que possa tomar medidas para evitar que elas voltem a ocorrer.

A informação, interactivamente estipulada pelo operador durante a definição das operações necessárias para cumprir o plano de montagem, é escrita no fim da simulação de cada operação de montagem e, no caso de não se verificar nenhum erro dessa execução durante a simulação, na variável `instrucao`. Esta variável é do tipo:

```
instrucao                : operacoes_de_montagem
```

No fim de cada uma destas operações é também incrementada a variável `indice` (inicialmente com o valor nulo) de forma a ser possível escrever o conteúdo da variável `instrucao` no vector de estruturas `listagem_programa`.

Esta sequência de operações é armazenada num vector de estruturas de dados. O vector e a estrutura de dados têm a seguinte estrutura:

```
listagem_programa    : ARRAY [100] OF operacoes_de_montagem
```

em que `operações_de_montagem` é uma estrutura do tipo:

```
STRUCT operacoes_de_montagem
     operação          : STRING
     argumento_1       : STRING
     argumento_2       : STRING
     argumento_3       : STRING
     argumento_4       : STRING
     argumento_5       : STRING
     argumento_6       : STRING
     argumento_7       : STRING
ENDSTRUCT
```

Todas as operações de montagem terminam com a junta 3 do robô recolhida (na coordenada Z mínima).

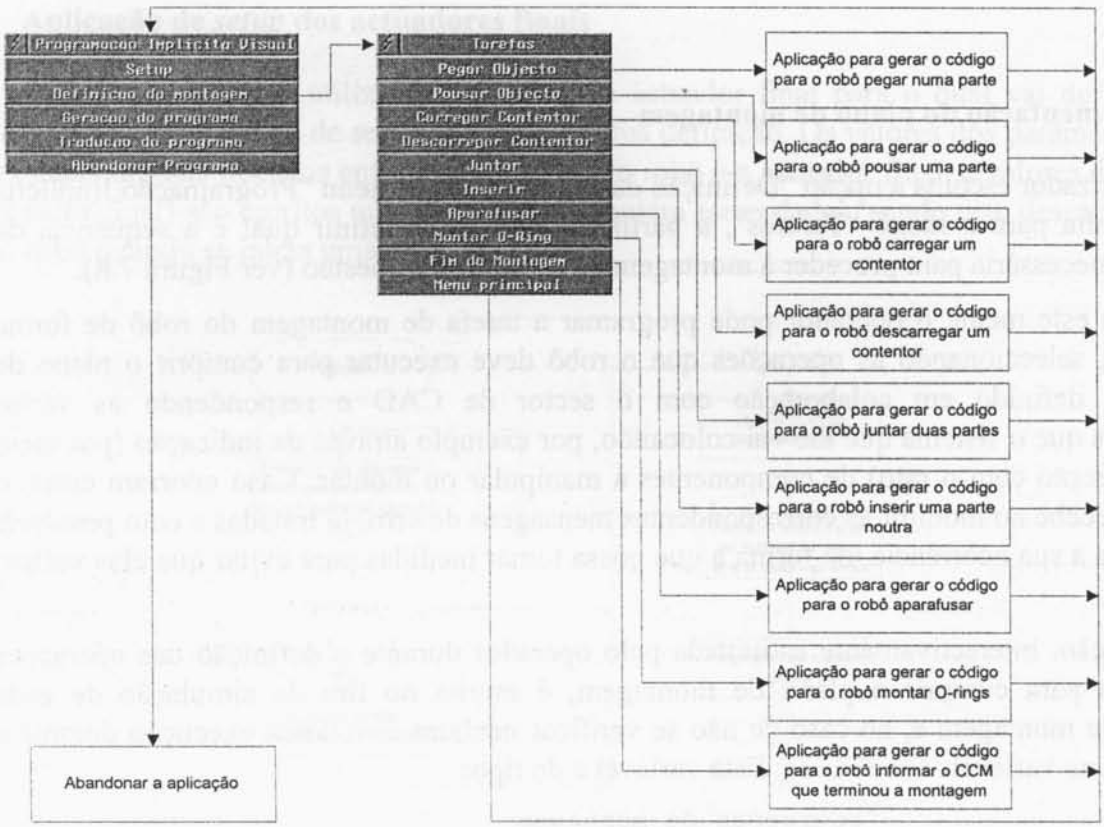


Figura 7.8: Estrutura dos menus para a implementação do plano de montagem

2.3 Aplicação de geração do programa desenvolvido em linguagem GSL

Após se ter implementado o plano de montagem, através da definição da sequência de operações necessárias para o cumprir, é necessário transformar toda a informação resultante num programa que possa ser executado, quer no *software* de simulação, quer no robô.

A função desta rotina é ler o vector `listagem_programa`, onde se encontram registadas as operações de montagem a executar para cumprir o plano de montagem, e os seus parâmetros (nas estruturas `operações_de_montagem`) e converter toda esta informação num programa GSL, executável de forma independente. O funcionamento da rotina que implementa esta funcionalidade é descrito no fluxograma apresentado na Figura 7.9.

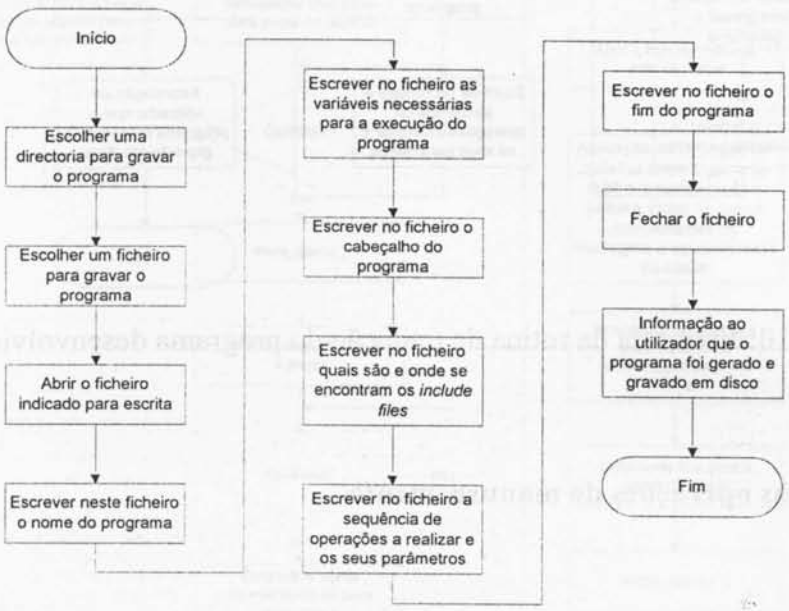


Figura 7.9: Estrutura da rotina de geração do programa desenvolvido

2.4 Aplicação de tradução do programa desenvolvido

Esta rotina apresenta a mesma funcionalidade da anterior, com a única diferença que o programa não é gerado em linguagem GSL, mas sim em linguagem V/V+, a linguagem de programação do robô Adept. O funcionamento desta rotina é descrito na Figura 7.10, sendo em tudo semelhante ao da rotina anterior.

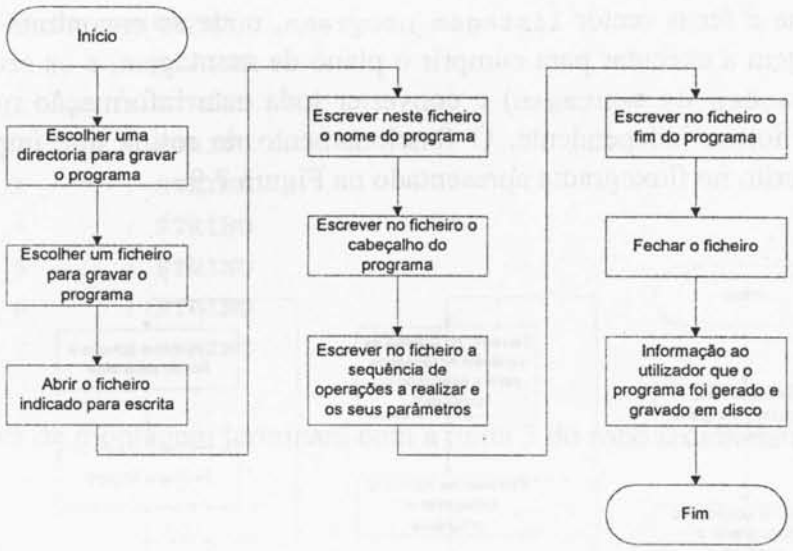


Figura 7.10: Estrutura da rotina de tradução do programa desenvolvido

3 Implementação das operações de manuseamento

3.1 Operação Pegar

Esta rotina tem por objectivo simular a execução da operação Pegar. Esta operação permite que o robô pegue num componente ou produto montado, de forma a poder manipulá-lo. O seu funcionamento é descrito no fluxograma da Figura 7.11.

Sempre que esta rotina é executada, invoca uma rotina de troca de garras. Após esta operação de troca de garras, o robô aproxima-se da localização do objecto a manipular. Este movimento de aproximação é efectuado com a garra recolhida (junta 3 no seu mínimo valor), para evitar colisões com outros objectos que se encontrem sobre as mesas ou contentores da Célula de Montagem. Quando esta aproximação está concluída (a garra do robô encontra-se sobre a vertical do objecto a apanhar), a garra desce, fecha e volta a subir, já com o objecto agarrado.

3.2 Operação Pousar

Esta rotina tem a função contrária à da rotina anterior. O seu objectivo é simular a operação que o robô executa para pousar um objecto sobre uma das mesas da Célula de Montagem, ou sobre um dos contentores que se encontram nas mesas de transferência da Célula. O seu funcionamento encontra-se descrito na Figura 7.12.

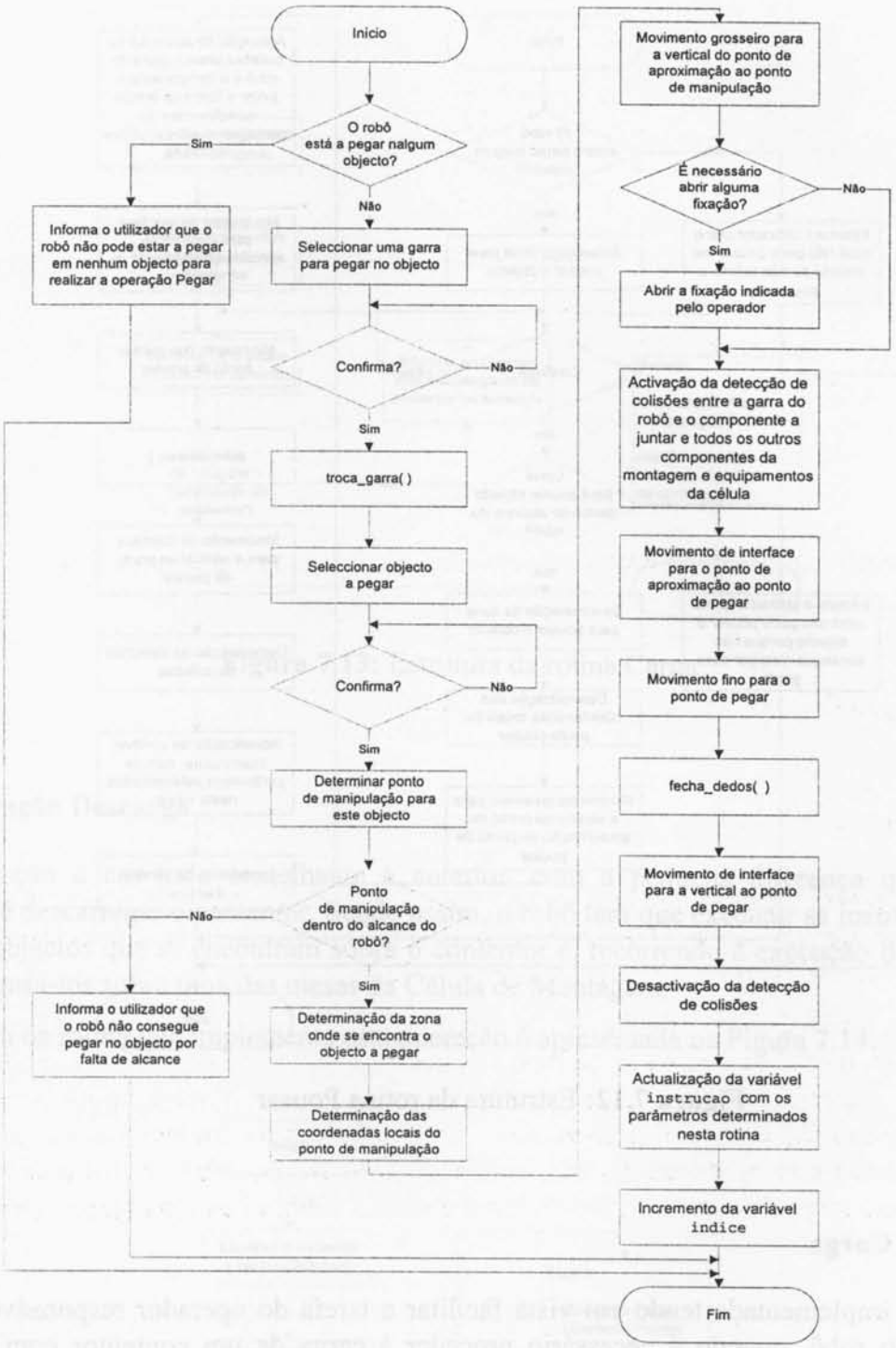


Figura 7.11: Estrutura da rotina Pegar

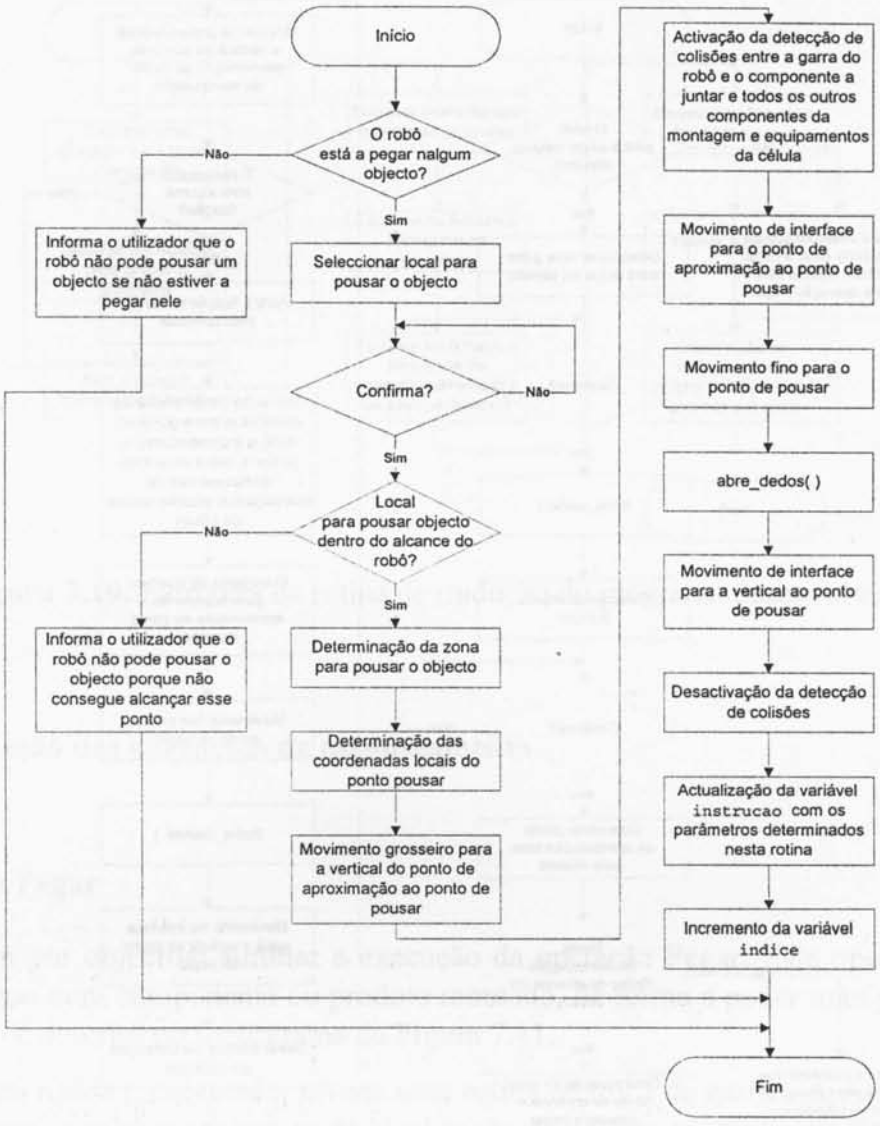


Figura 7.12: Estrutura da rotina Pousar

3.3 Operação Carga

Esta rotina foi implementada tendo em vista facilitar a tarefa do operador responsável pela programação do robô, quando é necessário proceder à carga de um contentor com vários componentes, ou produtos montados, que se encontram sobre uma das mesas da Célula de Montagem.

Como pode ser visto na Figura 7.13, que descreve a estrutura desta rotina, o operador só tem que indicar quais são as peças a carregar e o contentor onde devem ser colocadas, ficando esta rotina responsável pela tarefa de transformar esta ordem num conjunto de instruções Pegar e Pousar, que é como são traduzidas para o programa gerado.

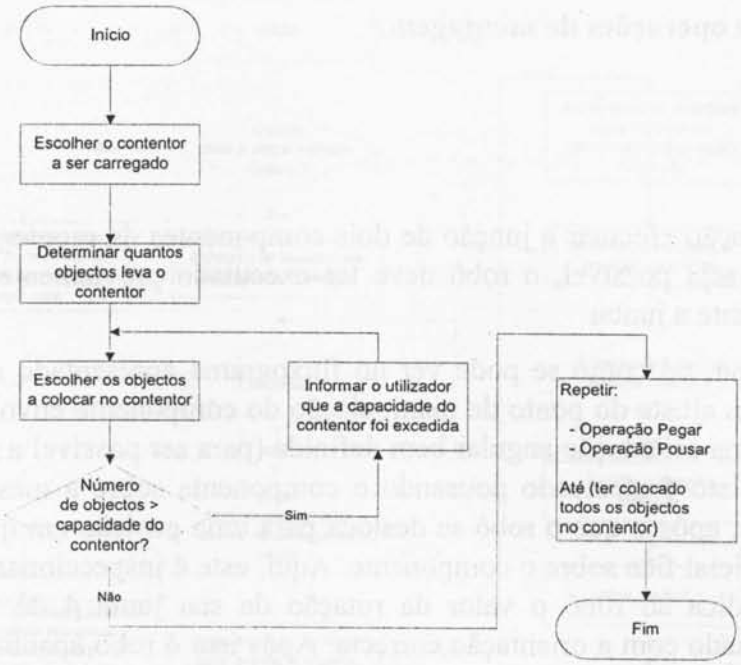


Figura 7.13: Estrutura da rotina Carga

3.4 Operação Descarga

Esta operação é em tudo semelhante à anterior, com a pequena diferença que agora o objectivo é descarregar o contentor. Sendo assim, o robô tem que executar as instruções Pegar sobre os objectos que se encontram sobre o contentor e, recorrendo à execução de operações Pousar, pousá-los sobre uma das mesas da Célula de Montagem.

A estrutura da rotina que implementa esta operação é apresentada na Figura 7.14.

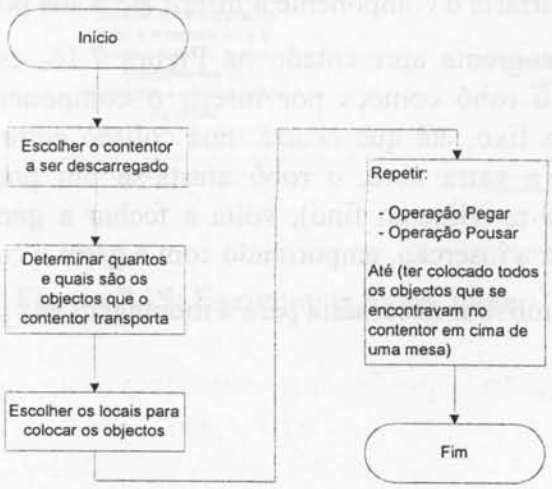


Figura 7.14: Estrutura da rotina Descarga

4 Implementação das operações de montagem

4.1 Operação Juntar

Esta rotina tem por função efectuar a junção de dois componentes da montagem. Para que a execução desta rotina seja possível, o robô deve ter executado previamente uma operação Pegar sobre o componente a juntar.

Basicamente, esta rotina, tal como se pode ver no fluxograma apresentado na Figura 7.15, começa por efectuar um ajuste do ponto de manipulação do componente envolvido, de forma a que este fique com uma orientação angular bem definida (para ser possível a sua junção com o outro componente). Isto é efectuado pousando o componente sobre a mesa de operações num local pré-definido, após o que o robô se desloca para uma posição em que a câmara do Sistema de Visão Artificial fica sobre o componente. Aqui, este é inspeccionando e o Sistema de Visão Artificial indica ao robô o valor da rotação da sua junta 4 de forma a que o componente seja apanhado com a orientação correcta. Após isto o robô apanha o componente e junta-o (operação muito semelhante à operação Pousar) ao outro componente da montagem.

4.2 Operação Inserir

Tal como a anterior, esta rotina tem como objectivo simular uma operação de montagem, neste caso concreto a operação de inserção. No entanto, apresenta a particularidade de ter sido pensada com o objectivo de se proceder à inserção completa de um componente dentro de outro, na situação em que só estivesse disponível uma garra comum, com dedos. Idealmente, para se proceder a esta operação deveria estar disponível uma garra equipada com um sistema de actuadores pneumáticos. Neste caso a garra seria a responsável por alinhar o componente a inserir com o outro e proceder ainda a uma ligeira inserção do primeiro no segundo, de forma a preparar a execução da tarefa do actuador. Após esta primeira inserção, o actuador existente na garra seria activado, e empurraria o componente a inserir até à sua posição final.

Como pode ser visto no fluxograma apresentado na Figura 7.16, esta operação difere da anterior porque, neste caso, o robô começa por inserir o componente que se encontra a manipular no que se encontra fixo, até que ocorra uma colisão entre a garra e este último componente. Nesse instante, a garra abre, o robô afasta-se um pouco (para o ponto de aproximação onde se inicia o movimento fino), volta a fechar a garra e desloca-se para a posição onde deverá completar a inserção, empurrando com a garra o componente a inserir.

Esta rotina acabou por não se mostrar necessária para a montagem das peças PD001 e PD002.

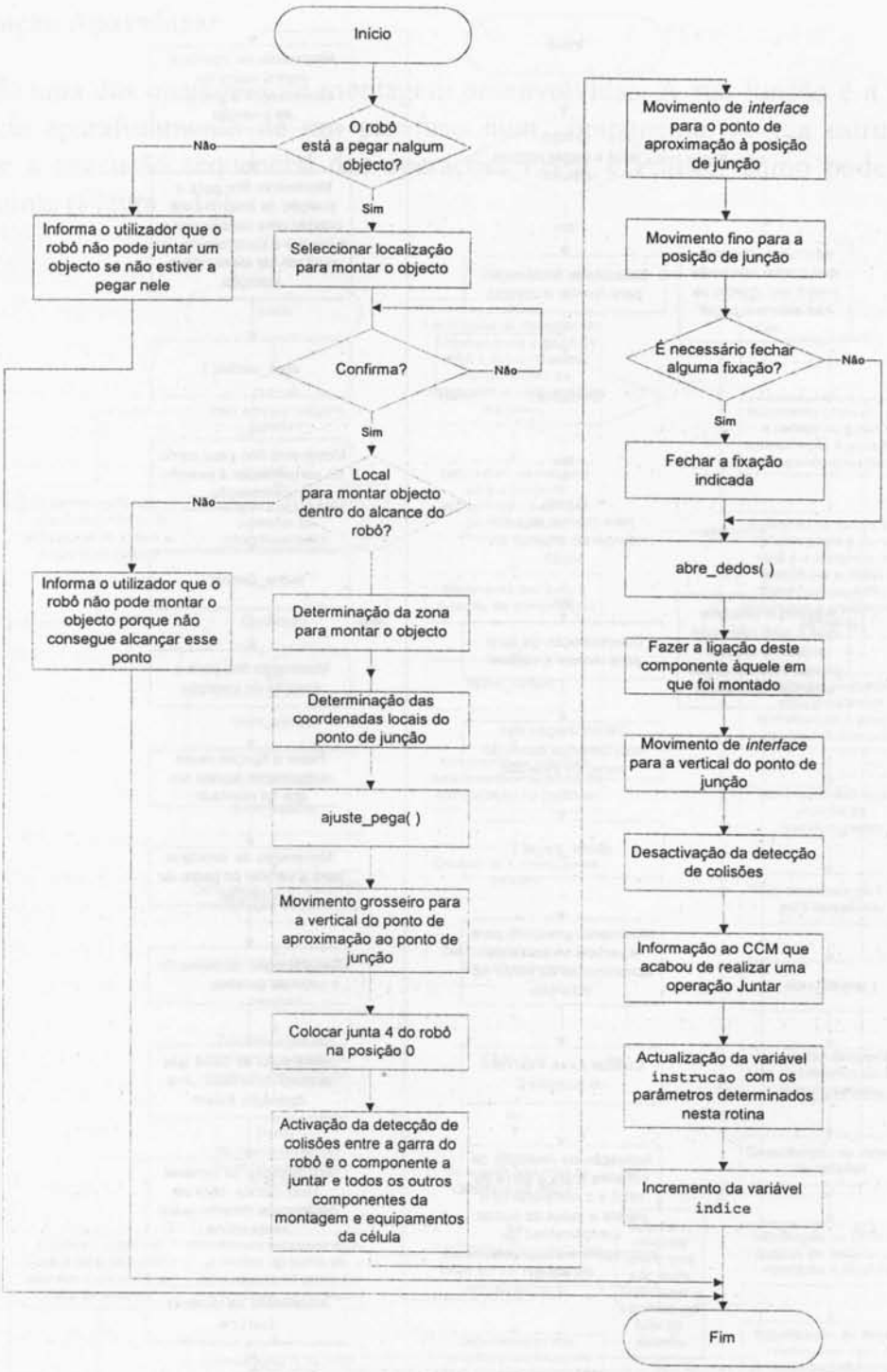


Figura 7.15: Estrutura da rotina Juntar

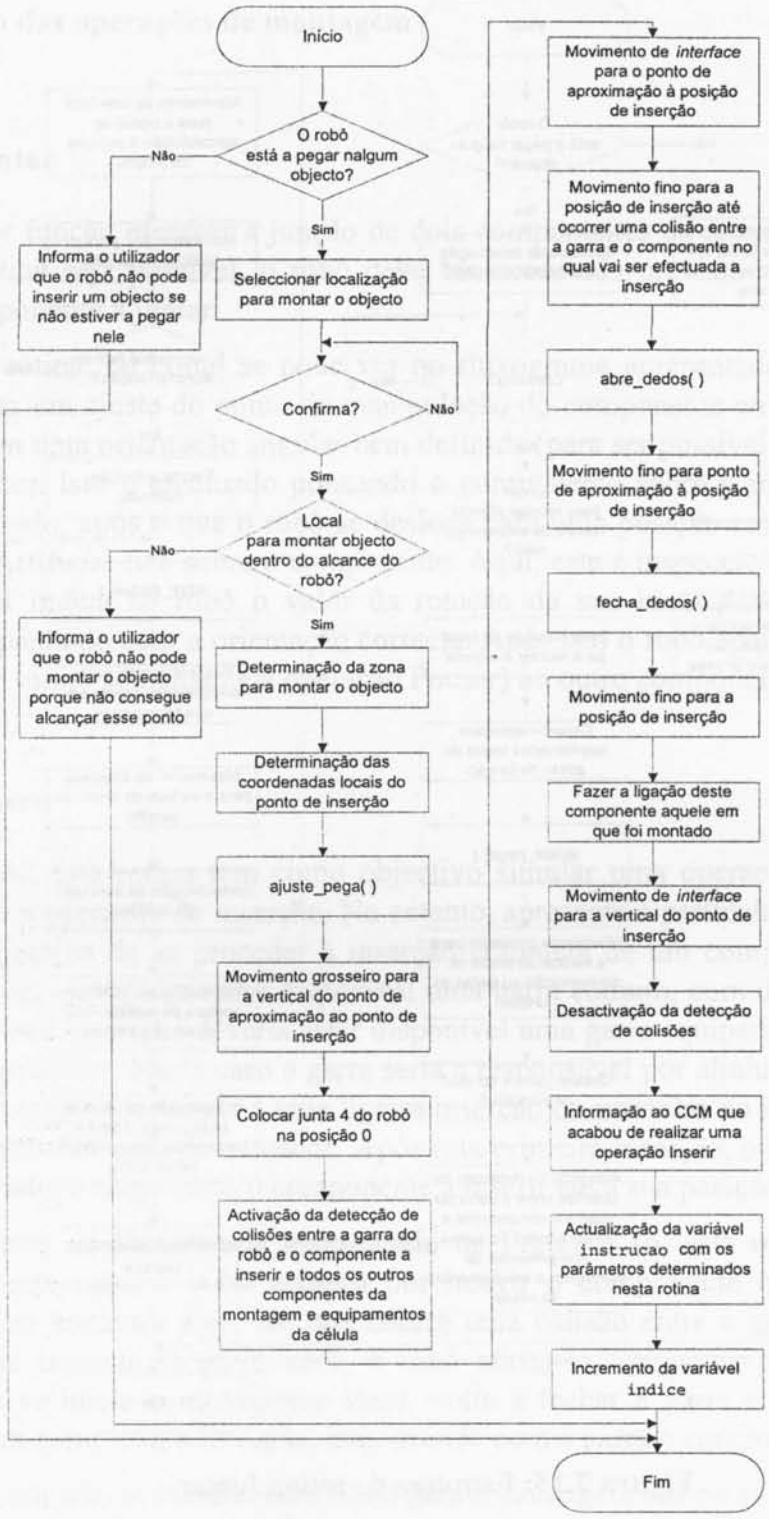


Figura 7.16: Estrutura da rotina Inserir

4.3 Operação Aparafusar

Esta é mais uma das operações de montagem desenvolvidas. A sua função é a simulação da operação de aparafusamento de um parafuso num componente. A sua estrutura é muito semelhante à execução sequencial das operações Pegar e Pousar, como pode ser visto na figura seguinte (Figura 7.17).

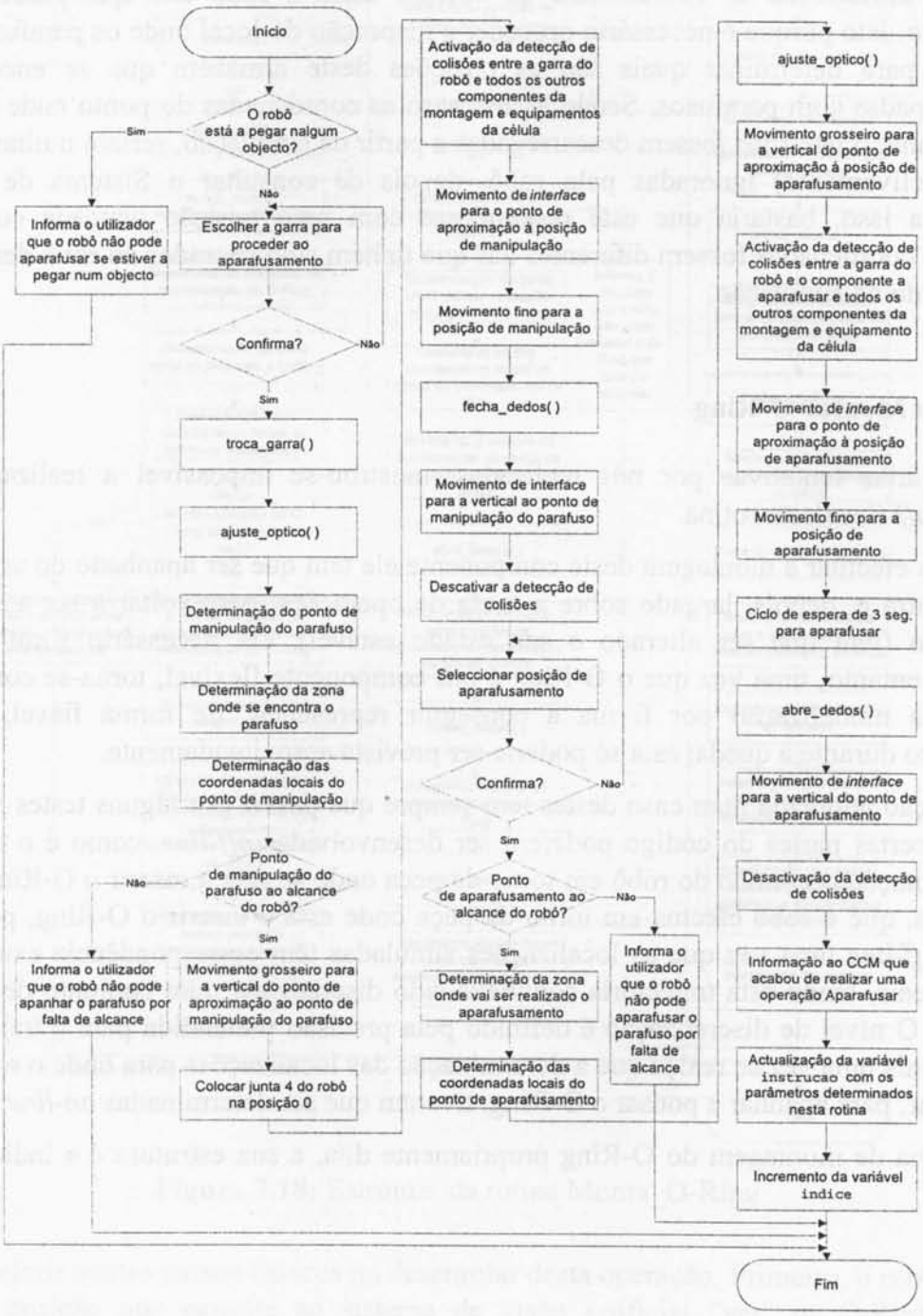


Figura 7.17: Estrutura da rotina Aparafusar

Relativamente às duas operações de montagem apresentadas anteriormente, a operação Aparafusar tem a particularidade de a operação Pegar (no parafuso) e Pousar (ou aparafusar o parafuso propriamente dito) serem realizadas dentro desta operação, não tendo o robô que estar previamente a segurar no parafuso.

No entanto, apesar de o robô ter que ir pegar num parafuso para executar esta operação, as coordenadas do local onde ele tem que ir apanhar o parafuso não são descarregadas, sendo descarregadas unicamente as coordenadas da posição onde o robô tem que proceder ao aparafusamento. Isto porque é necessário proceder à inspecção do local onde os parafusos são armazenados, para determinar quais são as posições deste armazém que se encontram realmente ocupadas com parafusos. Sendo assim, caso as coordenadas do ponto onde o robô deveria ir apanhar o parafuso fossem descarregadas a partir da simulação, seriam muitas vezes (e são-no efectivamente) ignoradas pelo robô, depois de consultar o Sistema de Visão Artificial; para isso, bastaria que este respondesse com uma posição ocupada com um parafuso cujas coordenadas fossem diferentes das que tinham sido passadas ao robô dentro do programa gerado na simulação.

4.4 Operação Montar O-Ring

Apesar das várias tentativas por nós realizadas, mostrou-se impossível a realização da programação *off-line* desta rotina.

Dado que para efectuar a montagem deste componente ele tem que ser apanhado do armazém onde se encontra e, depois, largado sobre a mesa de operações, para voltar a ser apanhado noutra posição (tem que ser alterado o seu estado estável), era necessário simular esta operação. No entanto, uma vez que o O-Ring é um componente flexível, torna-se complexo efectuar a sua modelização por forma a conseguir representar, de forma fiável, o seu comportamento durante a queda; esta só poderia ser prevista aproximadamente.

Assim, a solução adoptada num caso destes tem sempre que passar por alguns testes *on-line*, pese embora certas partes do código poderem ser desenvolvidas *off-line*, como é o caso da operação de rotação do punho do robô em torno da peça onde se está a inserir o O-Ring: toda esta trajectória, que o robô efectua em torno da peça onde está a inserir o O-Ring, pode ser programada *off-line* uma vez que as localizações simuladas têm correspondência exacta nas localizações reais, tendo esta trajectória complexa sido discretizada num conjunto de pontos de passagem. O nível de discretização é definido pela precisão pretendida para a trajectória. No entanto, mais uma vez se realça que a determinação das localizações para onde o robô tem que se deslocar, para apanhar e pousar o O-Ring, tiveram que ser determinadas *on-line*.

Quanto à rotina de montagem do O-Ring propriamente dita, a sua estrutura é a indicada na Figura 7.18.

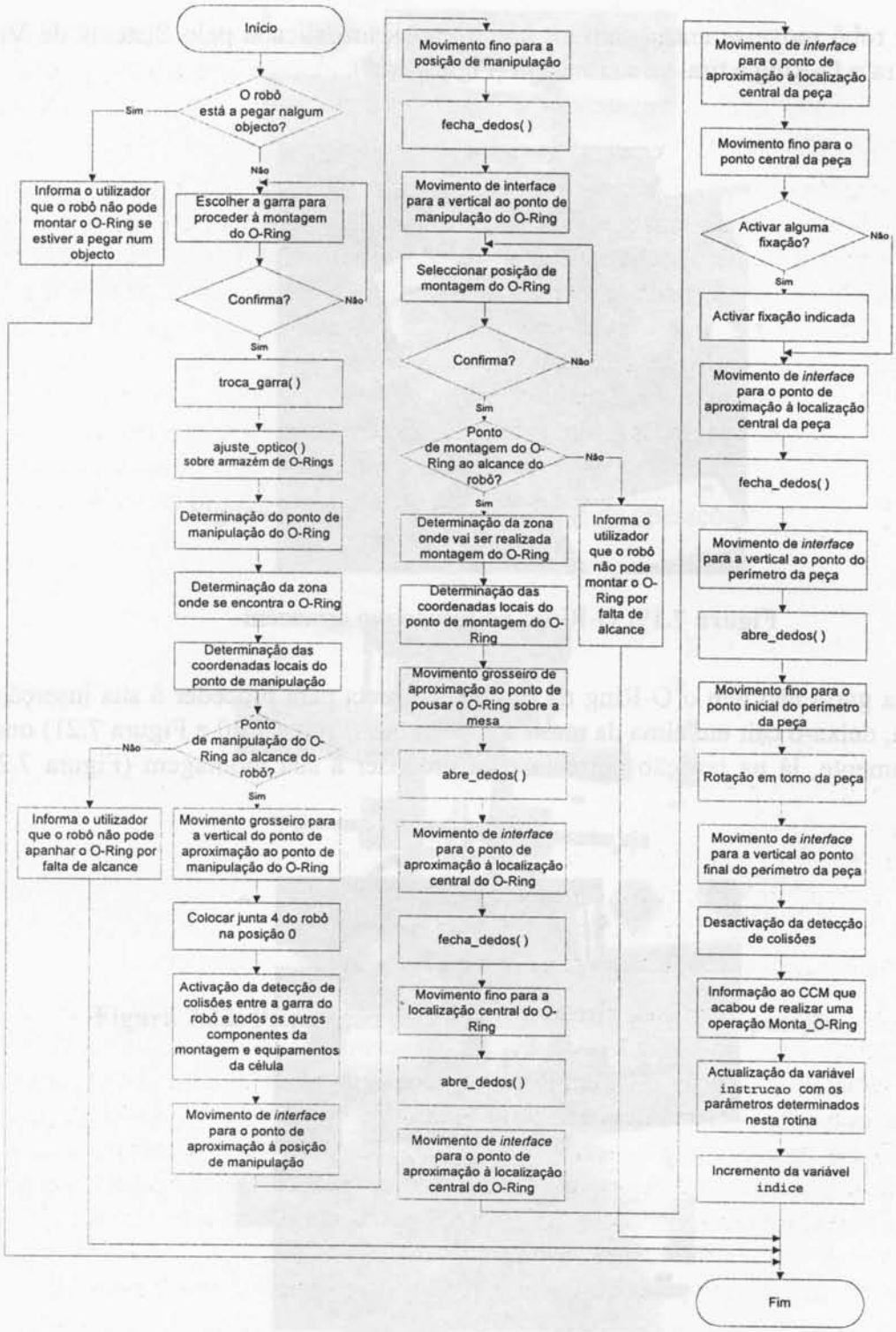


Figura 7.18: Estrutura da rotina Monta_O-Ring

Podemos referir quatro passos básicos no desenrolar desta operação. Primeiro, o robô move-se para uma posição que permite ao sistema de visão artificial "ver" os O-Rings que se encontram no respectivo armazém. Uma vez nesta posição, o sistema de visão artificial pode determinar a localização dos O-Rings dentro do respectivo armazém, e informar o robô da sua

localização. O robô move-se então para a primeira posição indicada pelo Sistema de Visão Artificial, agarra o O-Ring e tira-o do armazém (Figura 7.19).

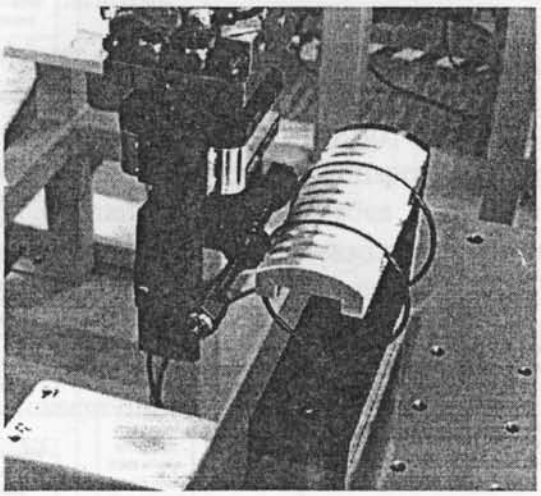


Figura 7.19: O-Rings no respectivo armazém

Uma vez que a garra não tem o O-Ring na posição correcta para proceder à sua inserção na peça cilíndrica, deixa-o cair em cima da mesa de operações (Figura 7.20 e Figura 7.21) onde é agarrado novamente, já na posição correcta para proceder à sua montagem (Figura 7.22 e Figura 7.23).

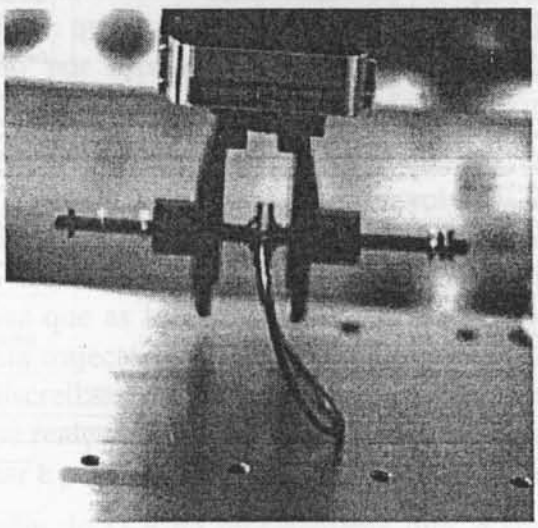


Figura 7.20: Deixar cair o O-Ring sobre a mesa de operações

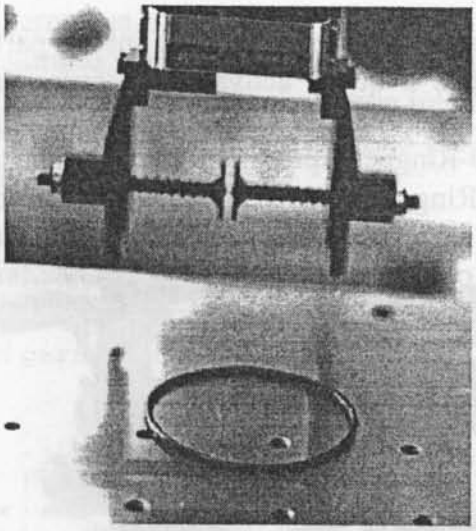


Figura 7.21: O-Ring sobre a mesa de operações

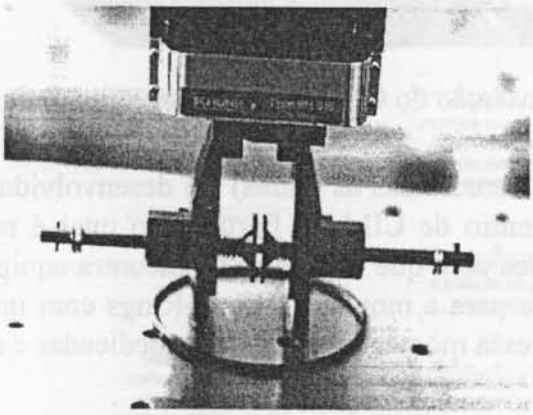


Figura 7.22: Preparação para agarrar o O-Ring na nova posição

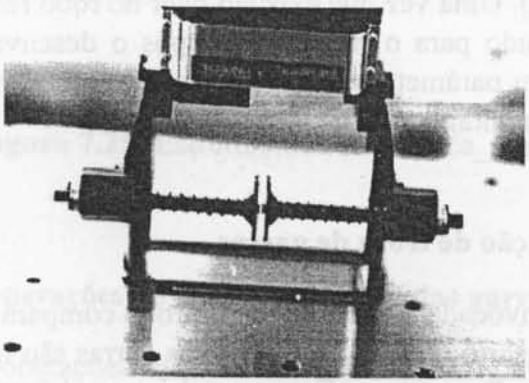


Figura 7.23: Agarrar o O-Ring na nova posição

O robô move-se então para uma localização onde pode inserir parte do O-Ring na peça da montagem (ver Figura 7.24), após o que é activado um outro actuador pneumático para segurar o O-Ring nessa posição. Após isto, a garra do robô efectua uma rotação à volta da peça onde se pretende inserir o O-Ring, comprimindo-o contra a peça até o inserir totalmente. Após terminada a inserção do O-Ring, o actuador pneumático é desactivado.

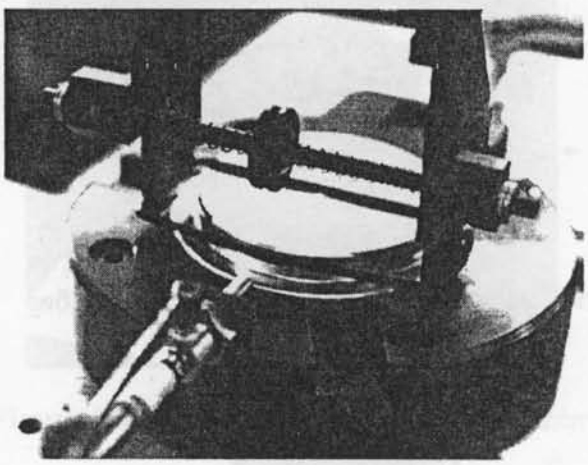


Figura 7.24: Inicio da inserção do O-Ring no componente onde vai ser montado

De referir que esta garra (bem como todas as outras) foi desenvolvida de raiz pelo Sector de Apoio Mecânico do CCP - Centro de CIM do Porto, pelo qual é responsável o Eng. Rui Fazenda. Em particular, os dedos com que esta garra se encontra equipada mostraram-se uma solução eficaz e de baixo custo para a montagem de O-Rings com uma garra de dois dedos (quando o normal é proceder a essa montagem com garras dedicadas e de elevado custo).

5 Rotinas auxiliares

Estas rotinas foram implementadas para auxiliar o cumprimento de outras operações (de montagem e de manuseamento). Uma vez que existem quer no robô real, quer na simulação, o seu código nunca é descarregado para o robô Adept após o desenvolvimento de um novo programa: estas rotinas não têm parâmetros de funcionamento, ou têm-nos embebidos dentro dos parâmetros das rotinas de montagem.

5.1 Implementação da operação de troca de garras

De cada vez que esta rotina é invocada, o controlador do robô compara a garra que este tem no punho com a garra que é necessário utilizar. Se estas duas garras são iguais, então a execução desta rotina termina sem que nada aconteça. Caso contrário, o robô pousa a garra que tem no punho no respectivo armazém e vai de seguida buscar a garra correcta. Isto mesmo se pode verificar na Figura 7.25, que apresenta a estrutura desta rotina.

A informação respeitante à garra que o robô deve ir buscar é passada a esta rotina como parâmetro de entrada. Pelo contrário, a variável `garra_actual` guarda o valor da garra que o robô tem em cada instante no punho (esta variável, do tipo inteiro, assume o valor da posição do armazém onde se encontrava a garra que o robô tem no punho, ou o valor nulo, caso ele não empunhe qualquer garra).

Antes de o robô pousar uma garra, executa sempre a rotina fecha_dedos( ), uma vez que as posições no armazém de garras são muito estreitas (optimização de espaços) e existem garras que, com os dedos abertos, entram em colisão com a estrutura de tal armazém. Após a troca de garras é actualizada a variável `garra_actual`.

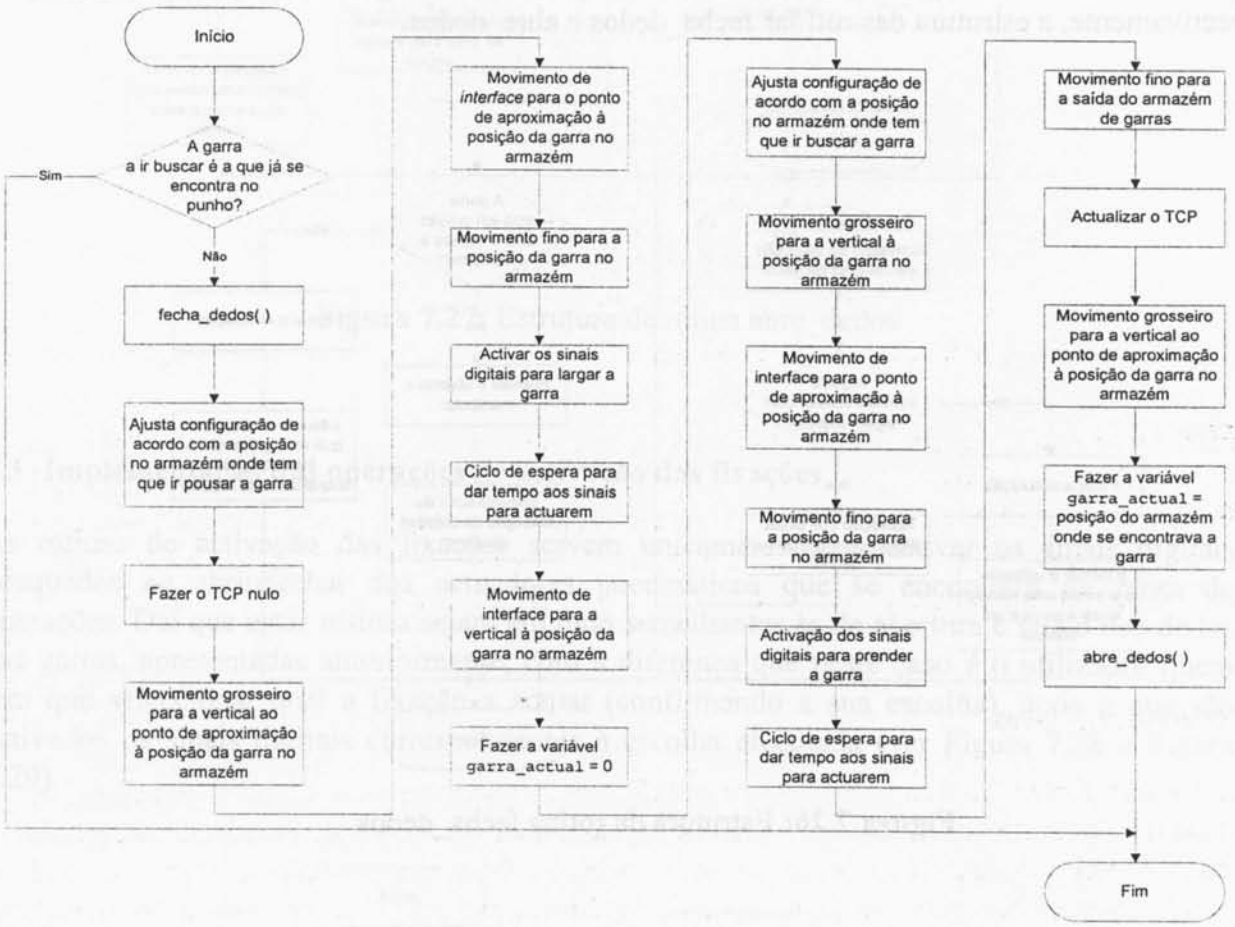


Figura 7.25: Estrutura da rotina troca_garra

5.2 Implementação das operações de abertura e fecho das garras

Estas rotinas simulam as operações de abertura e fecho das garras no robô. No robô real também foram implementadas estas rotinas, pois é muito mais intuitivo executar uma rotina cujo nome é abre_dedos ou fecha_dedos, do que colocar os sinais digitais do robô,

correspondentes às electroválvulas que controlam a abertura e fecho dos dedos das garras, no estado 1 ou 0, respectivamente.

A única particularidade destas rotinas reside na sua simulação, e situa-se no facto de ser necessário informar o robô que deve prender ou largar os componentes a manipular; para esse efeito não basta fechar ou abrir as garras, respectivamente, tal como no mundo real. Na simulação é necessário dar uma ordem de ligação de forma a que o componente a manipular fique com uma ligação à garra quando esta fecha, e essa ligação seja quebrada quando a garra abre. Isto corresponde ao estabelecimento das relações descritas pelas estruturas de dados de ligação.

O que acaba de ser dito pode ser constatado na Figura 7.26 e Figura 7.27 que apresentam, respectivamente, a estrutura das rotinas fecha_dedos e abre_dedos.

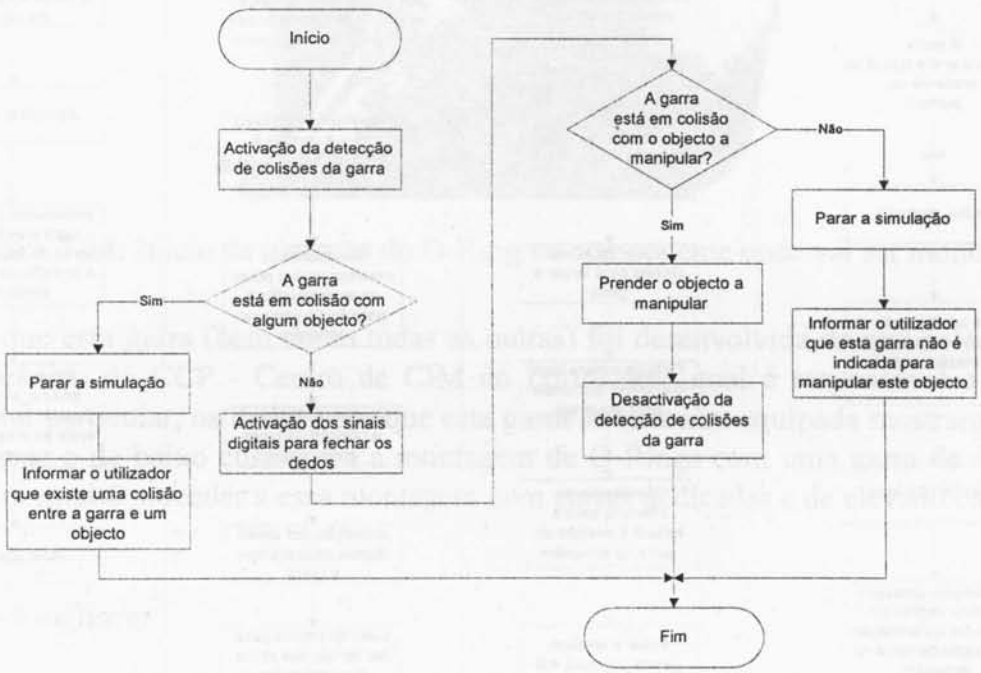


Figura 7.26: Estrutura da rotina fecha_dedos

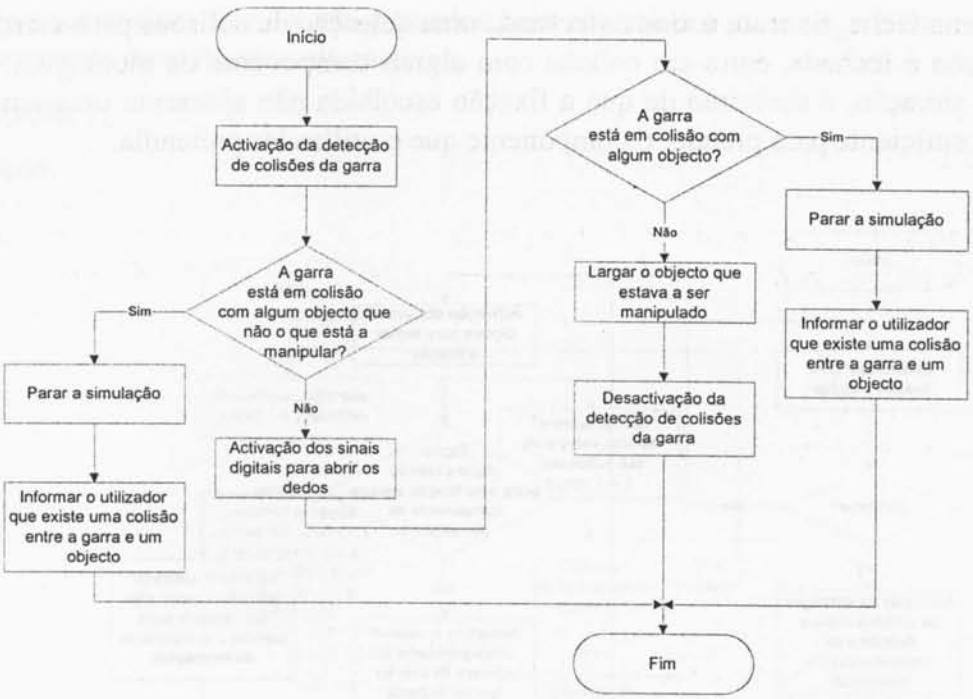


Figura 7.27: Estrutura da rotina abre_dedos

5.3 Implementação das operações de activação das fixações

As rotinas de activação das fixações servem unicamente para activar os sinais digitais adequados ao abrir/fechar dos actuadores pneumáticos que se encontram na mesa de operações. Daí que estas rotinas sejam em tudo semelhantes às de abertura e fecho dos dedos das garras, apresentadas anteriormente, com a diferença que neste caso é o utilizador quem tem que seleccionar qual a fixação a actuar (confirmando a sua escolha), após o que são activados os sinais digitais correspondentes à escolha efectuada (ver Figura 7.28 e Figura 7.29).

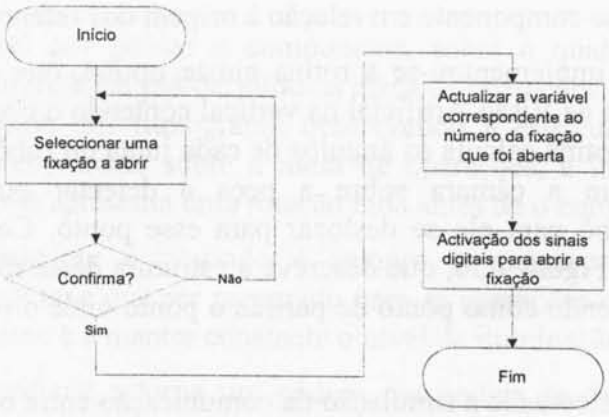


Figura 7.28: Estrutura da rotina abre_fixacao

No caso da rotina fecha_fixacao, é ainda efectuada uma detecção de colisões para verificar se, quando a fixação é fechada, entra em colisão com algum componente da montagem. A não acontecer esta situação, é sinónimo de que a fixação escolhida não apresenta uma amplitude de movimento suficiente para prender o componente que o utilizador pretendia.

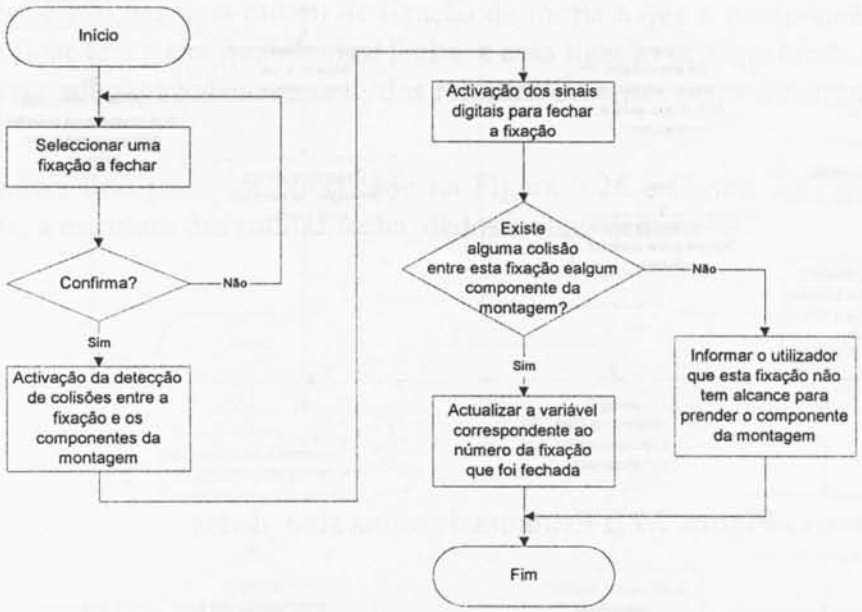


Figura 7.29: Estrutura da rotina fecha_fixacao

5.4 Implementação da operação de ajuste do ponto de visualização

Durante a execução de uma montagem, o robô interroga frequentemente o Sistema de Visão Artificial sobre a forma de pegar nos componentes da montagem. Esta comunicação é extremamente simples, porque o controlador do robô envia uma mensagem com um pedido de identificação de um componente e o Sistema de Visão Artificial responde-lhe indicando se detectou ou não esse componente e, caso o tenha detectado, as coordenadas onde o detectou e a rotação do referencial desse componente em relação à origem dos referenciais.

Para efeitos de simulação, implementou-se a rotina ajuste_optico, que é a responsável por colocar a câmara do Sistema de Visão Artificial na vertical contendo o eixo do componente da montagem a analisar. Esta rotina calcula os ângulos de cada junta do robô, de forma a que ele possa ser posicionado com a câmara sobre a peça a detectar ou identificar, dando seguidamente ordem ao robô para ele se deslocar para esse ponto. Como se pode ver no fluxograma apresentado na Figura 7.30, que descreve a estrutura desta rotina, os cálculos são efectuados iterativamente, tendo como ponto de partida o ponto onde o robô pousou a peça a detectar ou identificar.

É dentro desta rotina que é efectuada a simulação da comunicação entre o robô e o Sistema de Visão Artificial. Esta operação é implementada através de uma rotina de comunicação que envia um pedido (com parâmetros) e espera pela resposta do Sistema de Visão Artificial.

Consoante os parâmetros que lhe são passados pelo controlador do robô, o Sistema de Visão Artificial apresenta duas funções alternativas:

- identificação;
- localização.

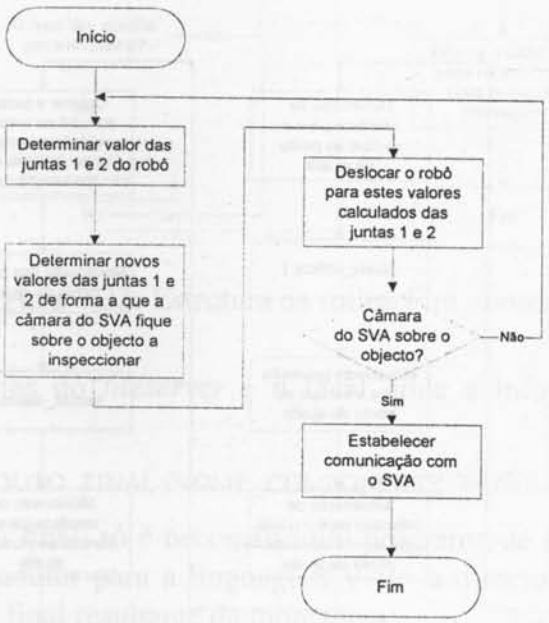


Figura 7.30: Estrutura da rotina ajuste_optico

A primeira destas operações de inspecção automática retorna se detectou algum objecto no seu campo de visão e de que componente se trata, caso o tenha conseguido identificar.

A segunda retorna a localização do componente identificado no referencial próprio da câmara, ou no referencial do robô, conforme o pretendido.

5.5 Implementação da operação de ajuste do ponto de manipulação

Esta rotina é responsável por pousar o componente, sobre o qual vai ser realizada uma operação de montagem, num local pré-definido da mesa de operações. Como pode ser visto na Figura 7.31, que apresenta um fluxograma descrevendo a estrutura desta rotina, o robô começa por pousar o componente sobre a mesa de operações, e invoca a rotina de visão artificial. A junta 4 do robô apresenta uma rotação nula antes de o componente ser pousado.

O local onde o componente é pousado é sempre o mesmo, independentemente do componente, uma vez que teve que ser preparado para as operações de inspecção de forma a evitar os reflexos luminosos e a manter constante o nível de iluminação.

O Sistema de Visão Artificial retorna um código que indica se detectou correctamente o componente da montagem e, neste caso, qual a rotação que este necessita de sofrer para ficar

na posição de rotação nula. O robô apanha então o componente, colocando previamente a sua junta 4 na posição correcta para o apanhar, em função da rotação que ele apresenta.

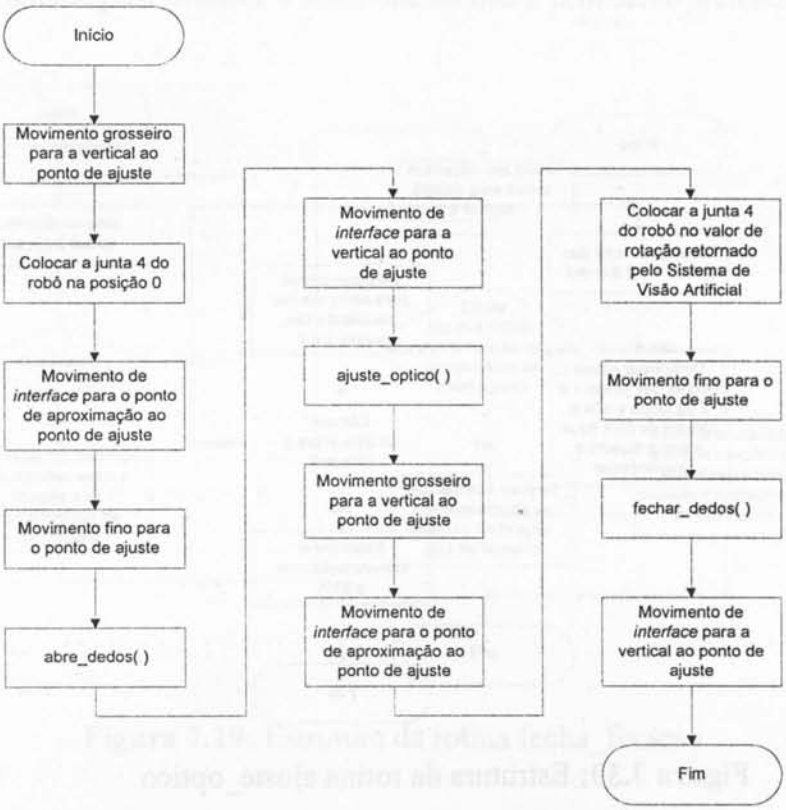


Figura 7.31: Estrutura da rotina ajuste_pega

5.6 Implementação da operação de informação ao controlador do fim da montagem

Esta rotina é responsável por informar o Controlador da Célula de Montagem que a operação de montagem que estava a decorrer terminou com sucesso. Após isto, e caso o robô ainda tenha alguma garra no punho, é dada uma ordem para a pousar como pode ser visto na Figura 7.32, que apresenta a estrutura desta rotina. Depois da conclusão desta operação é enviada uma nova mensagem ao Controlador da Célula de Montagem, informando-o que o robô se encontra pronto para receber uma nova ordem de montagem.

6 Organização da informação no *fileserver*

A informação relevante para o funcionamento da Célula de Montagem, originada no Sector de Projecto, é armazenada num *fileserver* que se encontra instalado numa estação de trabalho SUN SPARCstation 10. Isto inclui a informação que é gerada no *software* de simulação, mais especificamente os programas gerados *off-line* para o robô, e a informação necessária ao correcto funcionamento do Sistema de Visão Artificial.

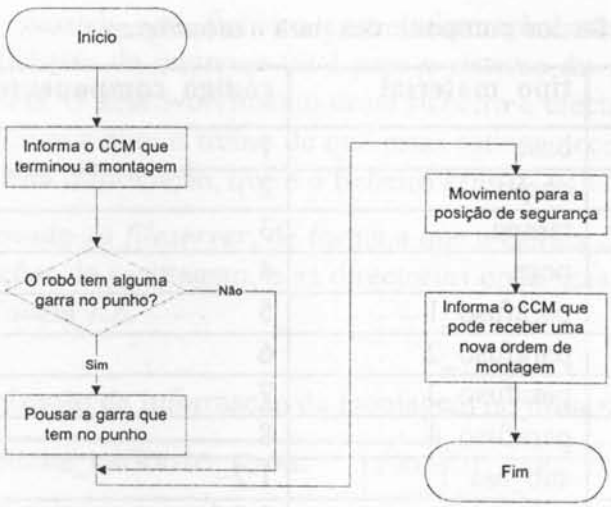


Figura 7.32: Estrutura da rotina Fim_montagem

A estrutura das directorias do *fileserver* e o local onde a informação é armazenada, é a seguinte:

/FILESERVER /NOME_PRODUTO_FINAL /NOME_COMPONENTE_PRODUTO

Como, para cada produto final, só é necessário um programa de montagem, definiu-se como directoria de saída do tradutor para a linguagem V/V+ a directoria do *fileserver* que tem o mesmo nome do produto final resultante da montagem:

/FILESERVER /NOME_PRODUTO_FINAL/

É a esta directoria que o controlador de célula irá buscar os programas para o robô quando receber uma ordem de montagem. O programa do robô tem o nome ROBOT_PROGRAM.

Adicionalmente, foi necessário criar uma tabela em que os nomes dos componentes do produto final são mapeados em números inteiros, de tal forma que o Controlador da Célula de Montagem seja capaz de fazer a descarga dessa informação para o controlador do robô de uma forma correcta e, ao mesmo tempo, seja simplificada a programação do robô no *software* de simulação, já que o operador não necessita de saber o código dos componentes, mas só seleccioná-los. Esta tabela encontra-se armazenada num ficheiro, que assume o nome ROBOT_PART_TABLE.

Para o caso de um produto de demonstração (o produto PD001 já referido atrás - apresentado no Anexo A), composto de quatro componentes e quatro parafusos, a informação é a contida na Tabela 7.1.

É também passado um ficheiro que contém informação relativa ao *setup* das garras que se encontram no respectivo armazém e outro com informação respeitante ao *setup* dos actuadores que se encontram sobre a mesa de operações. O nome do primeiro ficheiro é ROBOT_GARRAS_INFO (um exemplo de um ficheiro deste tipo, para a montagem da peça PD001, é apresentado na Figura 7.33) e, o do segundo, é ROBOT_ACTUADORES_INFO.

Tabela 7.1: Codificação dos componentes para a montagem

id_parte	tipo_material	código_componente
1	base	1
2	corpo	2
3	tampa	3
4	pega	4
5	parafuso_1	5
6	parafuso_2	6
7	parafuso_3	7
8	parafuso_4	8
9	sub_ass_1	1;2
10	sub_ass_2	1;2;3
11	sub_ass_3	1;2;3;4
12	sub_ass_4	1;2;3;4;5
13	sub_ass_5	1;2;3;4;5;6
14	sub_ass_6	1;2;3;4;5;6;7
15	peça_final	1;2;3;4;6;

```
1
RH918-ORINGS
0.0
0.0
87.0
0.0
0.0
80.0
2
PZN64-25
0.0
0.0
71.85
0.0
0.0
44.2
3
PZN64-105
0.0
0.0
71.85
0.0
0.0
64.6
4
GWA64-APARAFU
0.0
0.0
220.0
0.0
0.0
0.0
```

Figura 7.33: Ficheiro com a informação de *setup* dos actuadores finais

Por último, é também passado um ficheiro com informação sobre os componentes da montagem, sendo este ficheiro de interesse vital para o sistema de visão artificial. Este é o ficheiro VISION_PART_INFO. O desenvolvimento deste ficheiro é efectuado em paralelo com o das rotinas da visão artificial e com o treino de que estas rotinas necessitam. O robô também recebe um ficheiro com esta informação, que é o ficheiro ROBOT_PART_INFO.

A informação que é colocada no *fileserver*, de forma a que a Célula de Montagem execute de forma correcta as operações de montagem, e as directorias onde essa informação é colocada, são as apresentadas na Tabela 7.2.

Tabela 7.2: Organização da informação da montagem no *fileserver*

/FILESERVER	/NOME_PRODUTO_FINAL	/ ROBOT_PART_INFO
	ROBOT_GARRAS_INFO	
	ROBOT_ACTUADORES_INFO	
	ROBOT_PART_TABLE	
	ROBOT_PROGRAM	
	VISION_PART_INFO	

7 Teste da solução implementada

A peça PD001 é uma peça que tem sido produzida no CCP e que serviu de demonstração das capacidades da Oficina Piloto do CCP durante o Projecto ESPRIT 5629. Esta peça, bem como a sua "explosão" (de forma a ser possível ver os componentes que a compõem e a maneira como estes componentes devem ser montados), é apresentada no Anexo A.

Este exemplo de aplicação foi seleccionado, por várias razões:

- é uma tarefa de produção planeada e executada pelo CCP;
- o exemplo é suficientemente simples para ser usado como aplicação inicial, mas suficientemente complexo para produzir *feedback* significativo;
- estão envolvidas diferentes garras e componentes com tamanhos e formas variáveis, requerendo troca de garras e garras multifuncionais.

Para desenvolver esta aplicação foram seguidos os seguintes passos:

- planeamento do processo de montagem (desenvolvimento iterativo entre a área de CAD e a área de simulação);
- geração *off-line* dos programas de aplicação do robô Adept;
- simulação e teste da execução da tarefa de montagem dentro da célula de trabalho planeada;
- descarga do programa para o robô real e sua execução na planta fabril.

Para este produto concreto (peça PD001), o plano de montagem passa pela montagem inicial da base, seguida da montagem do O-Ring, do corpo, da pega e, por último, o aparafusamento dos quatro parafusos por uma ordem aleatória (mas sempre em cruz).

O autor desta Tese de Dissertação executou interactivamente este plano de montagem na aplicação desenvolvida, e a geração do programa originou a listagem das operações de montagem (e seus parâmetros) apresentada na Figura 7.34.

```
pegar( BASE, 2, 70, 65, 3, 3 )
juntar (BASE, 4, 344.5 , 248.5, 25+3, 1 )
monta_oring( O_RING_1 )
pegar( CORPO, 2, 200, 65, 12.5, 3 )
juntar( CORPO, 4, 344.5, 248.5, 25+10+11.5, BASE, SUB_ASS_1 )
pegar( TAMPA, 2, 70, 305, 9, 3 )
juntar( TAMPA, 4, 344.5, 248.5, 25+10+41.5+9, CORPO, SUB_ASS_2 )
pegar( PEGA, 2, 200, 305, 22, 2 )
juntar(PEGA, 4, 344.5, 248.5, 25+10+41.5+7+22, TAMPA, SUB_ASS_3 )
aparafusar (PARAFUSO_1, 4, 345-18.5, 249.5, 94 )
aparafusar (PARAFUSO_2, 4, 345+18.5, 249.5, 94 )
aparafusar (PARAFUSO_3, 4, 345, 249.5-18.5, 94 )
aparafusar (PARAFUSO_4, 4, 345, 249.5+18.5, 94 )
pegar(OUTPUT_PART, 4, 344.5, 248.5, 25+40, 3 )
pousar(OUTPUT_PART, 2, 70, 65, 40 )
fim_montagem()
```

Figura 7.34: Parte do código GSL do programa de montagem da peça PD001

Este programa, constituído pelas operações de montagem com os parâmetros indicados, foi executado no modelo do robô (existente na célula simulada no sistema de programação *off-line*), sem que tenham ocorrido colisões e sem que o robô apresentasse qualquer dificuldade em atingir os pontos pretendidos. A verificação da manipulação dos componentes da montagem também não apresentou nenhum problema, pelo que se procedeu à tradução do programa para a linguagem V/V+.

Posteriormente, este programa foi executado com total sucesso no robô Adept durante uma tarefa de montagem real da peça PD001 (cuja produção e montagem já foi repetida dezenas de vezes).

Capítulo 8

Conclusões

Neste capítulo procura-se dar resposta às seguintes questões:

- Qual era o problema que tínhamos que resolver?
- Que solução foi adoptada e implementada?
- Quais as vantagens da nossa solução?
- Quais as limitações da nossa solução?
- Quais os possíveis desenvolvimentos desta nossa solução?

O problema concreto que se nos punha, e que tentámos resolver com a realização deste trabalho, era o problema da programação de um robô que se encontra a realizar tarefas de montagem, especificamente a montagem das peças PD001 e PD002 (apresentadas no Anexo A).

Este robô de montagem encontra-se inserido na Célula de Montagem da Oficina Piloto do CCP, cujo funcionamento foi descrito no Capítulo 4, e necessita de múltiplas reprogramações dadas as características produtivas desta unidade: baixo volume de produção de peças com características bastante diferentes entre si.

A solução adoptada para resolver este problema passou, como foi visto no Capítulo 6, pela programação *off-line* do robô de montagem, recorrendo a um pacote de *software* de simulação gráfica.

A solução implementada passou pela descrição da tarefa de montagem a realizar pelo robô (implementação do plano de montagem) utilizando macro-instruções, o que simplifica e facilita em larga medida a tarefa de desenvolvimento do programa do robô.

O trabalho realizado foi testado efectuando-se a programação do robô Adept existente na Célula de Montagem do CCP - Centro de CIM do Porto, para realizar as tarefas de montagem das peças PD001 e PD002; constatou-se que a solução funciona sem qualquer problema.

Vamos de seguida passar a ver quais as vantagens e limitações da solução adoptada e, para terminar, fazer uma reflexão sobre qual a evolução possível deste sistema.

1 Características principais da solução implementada

1.1 Vantagens da aplicação desenvolvida no CCP

Entre as vantagens do sistema desenvolvido no CCP, para a programação do robô Adept, encontram-se as seguintes:

- não é necessário interromper uma tarefa em curso na Planta Fabril para se proceder ao desenvolvimento de um novo programa para o robô que vai executar as tarefas de montagem: graças à programação ser efectuada recorrendo à simulação, permite um desenvolvimento e teste exaustivo dos programas gerados *off-line*;
- a aplicação desenvolvida é adaptada à programação de tarefas de montagem, sendo fácil a aprendizagem do princípio de funcionamento do sistema de programação e sendo o interface com o utilizador bastante intuitivo;
- uma vez que as tarefas de montagem foram programadas no *software* de simulação podem, com pouco trabalho adicional, ser adaptadas para um novo controlador de robô, especialmente se este for um robô SCARA: dado que o código foi desenvolvido na linguagem neutra GSL e que o mesmo pode ser traduzido para várias linguagens de programação de robôs, basta utilizar os tradutores internos do IGRIP, ou desenvolver um novo tradutor para uma outra linguagem;
- as operações de montagem implementadas têm, embebidas no seu próprio código, instruções que permitem a realização de testes às operações, tais como detecção de colisões e verificação do alcance do robô; estas características facilitam a tarefa do programador do robô, uma vez que este não tem que se preocupar com tais aspectos da programação;
- o recurso à simulação permite, de forma fácil, a optimização do programa desenvolvido;
- a forma interactiva como a programação é realizada, usando o princípio de "apontar e seleccionar", evita que o programador do robô tenha que conhecer as coordenadas das localizações espaciais onde tem que pegar e pousar nos componentes da montagem.

O facto de termos conseguido todos estes pontos positivos, bem como o facto de a solução ter funcionado satisfatoriamente, leva-nos a pensar que a solução implementada é uma boa solução.

1.2 Limitações da aplicação desenvolvida no CCP

Mas este sistema, obviamente, também apresenta algumas limitações, algumas das quais já foram abordadas no capítulo anterior. Julgamos importante mencionar as seguintes:

- de momento, ainda não existe uma ligação directa entre o *software* de simulação e o sistema de CAD existente no CCP (Pro/Engineer) que não esteja limitada à troca de dados geométricos do produtos em formato *Render*, pelo que os referenciais de montagem e manipulação têm que ser inseridos manualmente, tendo ainda que ser realizada muitas vezes uma simplificação das geometrias importadas do sistema de CAD;
- o planeamento da montagem é efectuado manualmente, não se encontrando integrado nem no sistema de CAD, nem no *software* de simulação, levando a que a optimização dos possíveis planos de montagem determinados tenha que ser executado manualmente, através da sua programação e da determinação do tempo de ciclo de execução necessário;
- este sistema não cobre todas as possíveis operações de montagem (basta verificar no Capítulo 4 as operações de montagem que aí foram especificadas), pelo que pode ser necessário, em certos casos, desenvolver rotinas adicionais para realizar outras tarefas de montagem não previstas;
- os componentes da montagem têm que ser fornecidos ao robô no seu estado estável (são estados estáveis de um objecto todas as posições nas quais o objecto se mantenha em equilíbrio estável sobre uma superfície, independentemente das suas coordenadas de referência e orientação) em que vão ser montados;
- todos os componentes da montagem têm de possuir uma forma geométrica constante durante a operação de montagem; isto é, não são por exemplo permitidos componentes como molas (nestes casos, o programa ou rotina para executar essa operação específica de montagem terá que ser desenvolvido *on-line* no seu todo, ou pelo menos em parte - caso da rotina de montagem do O-Ring). Esta restrição surge porque, tal como foi referido a respeito da rotina que implementa a operação de montar o O-Ring, é muito difícil modelizar este tipo de componentes e, logo, prever por simulação o seu comportamento durante a montagem;
- cada componente da montagem só pode ser montado com um movimento linear. Movimentos combinados, consistindo em translações ou rotações, não são possíveis, uma vez que nas rotinas que implementam as operações postulámos que as trajectórias dos movimentos de interface e dos movimentos finos eram trajectórias rectas;
- durante uma operação de montagem, só um componente pode ser movimentado de cada vez. Para ser possível movimentar dois componentes em simultâneo durante a montagem seriam necessários, por exemplo, dois robôs (um para manipular cada um dos componentes a montar).

2 Desenvolvimentos futuros deste sistema

A solução desenvolvida, como vimos, ainda apresenta várias limitações passíveis de resolução futura ("a necessidade aguça o engenho").

No entanto, algumas há que, se satisfatoriamente resolvidas, tornam o sistema mais poderoso e de mais fácil utilização.

De entre essas, apontamos como prioritárias:

- desenvolver um interface normalizado entre o sistema de CAD existente no CCP - Centro de CIM do Porto e o *software* de simulação; este interface poderia, por exemplo, ser desenvolvido de acordo com a Norma STEP;
- desenvolver um sistema de planeamento de tarefas de montagem que, a partir da informação respeitante à montagem de um objecto proveniente do sistema de CAD, fosse capaz de desenvolver o plano de montagem para a tarefa correspondente a esse produto;
- implementar um planeador de trajectórias e um planeador de manipulação, de forma a complementar as acções do planeador de tarefas de montagem e a tornar possível o desenvolvimento, de forma automática, do programa de montagem para o robô Adept;
- proceder à sensorização da Célula de Montagem de forma a que fosse possível implementar uma estratégia de planeamento e recuperação de erros.

Referências

[Adept, 1993a] *Adept CC Controller User's Guide*; Adept Technology Inc.; (1993).

[Banks, 1996] Jerry Banks, John S. Carson, Barry L. Nelson; *Discrete-Event Sistem Simulation;* Second Edition; Prentice-Hall, Inc.; (1996).

[Bekey, 1996] George A. Bekey; *Trends in Robotics*; Communications of the ACM; Vol. 39/No. 2; February 1996; (1996).

[Bernhardt, 1992] R. Bernhardt, R. Dillman, K. Hormann, K. Tierney; *Integration of Robots into CIM*; Chapman & Hall; (1992).

[Bey, 1994] I. Bey, D. Ball, H. Bruhm, T. Clausen, W. Jacob, O. Knudsen, E. G. Schlechtendahl, T. Sørensen; *Neutral Interfaces in Design, Simulation and Programming for Robotics*; Springer-Verlag; (1994).

[Bourjault, 1984] A Bourjault; *Contribution a une approche méthodologique de l'assemblage automatisé: Elaboration automatique des séquences opératoires*; Thèse d'État, Univ. de Franche-Comté, Besançon, France; (1984).

[Bruhm, 1994] H. Bruhm; *Interface Between Robot Off-line Programming and Control Systems* ; Neutral Interfaces in Design, Simulation and Programming for Robotics; Edited by I. Bey, D. Ball, H. Bruhm, T. Clausen, W. Jacob, O. Knudsen, E. G. Schlechtendahl, T. Sørensen; Springer-Verlag; (1994); pp. 229-230.

[Brunner, 1995] Bernhardt Brunner, Klaus Arbter and Gerhard Hirzinger; *Graphical Robot Simulation Within the Framework of an Intelligent TeleSensor Programming System*; Graphics and Robotics - Edited by Wolfgang Straβer and Friedrich M. Wahl; Springer-Verlag; (1995); pp. 31-44.

[Caracciolo, 1995] R. Caracciolo, E. Ceresole, T. De Martino, F. Giannini; *From CAD Models to Assembly Planning*; Graphics and Robotics - Edited by Wolfgang Straβer and Friedrich M. Wahl; Springer-Verlag; (1995).

[Chen, 1992] ChuXin Chen, Mohan M. Trivedi, Clint R. Bidlack; *Simulation and Graphical Interface for Programming and Visualization of Sensor-based Robot Operation*; Proceedings of the 1992 IEEE International Conference on Robotics and Automation; Nice, France; May 1992; (1992); pp. 1095-1101.

[CiME, 1995] CiME; *Design & Engineering - The Virtues of Virtual NC*; CiME Computer Integrated Manufacture and Engineering; April/May 1995; (1995).

[Clausen, 1994] T. Clausen; *Intermediate Code for Robots (ICR)*; Neutral Interfaces in Design, Simulation and Programming for Robotics; Edited by I. Bey, D. Ball, H. Bruhm, T. Clausen, W. Jacob, O. Knudsen, E. G. Schlechtendahl, T. Sørensen; Springer-Verlag; (1994); pp. 106-125.

[Denavit-Hartenberg, 1955] J. Denavit, R. Hartenberg; *A Kinematic notation for lower pair mechanisms based on matrices*; Journal of Applied Mechanics; Vol. 22/N. 2; June1995; (1955); pp. 215-221.

[Deneb, 1995] Deneb Robotics, Inc; *IGRIP User Manual#2*; Deneb Robotics, Inc.; (1995).

[Dao, 1995] T. M. Dao, M. Galopin and Y. A. Youssef; *A Knowledge Based Simulation System for Optimum Design of Manufacturing Cell System*; Proceedings of the Third IASTED International Conference Robotics and Manufacturing; June 1995; Cancun-Mexico; (1995); pp. 328-331.

[Gleue, 1992] V. Gleue; *Introduction to the systems planning process*; Integration of Robots into CIM; Edited by R. Bernhardt, R. Dillman, K. Hormann, K. Tierney; Chapman & Hall; (1992); pp. 29-35.

[Gogg, 1992] T. J. Gogg and J. R. A. Mott; *Improve Quality and Productivity with Simulation*; JMI Consulting Group; (1992).

[Goldenberg, 1989] A. Goldenberg; *Modeling the interaction between robot and environment*; Sensor Devices and Systems for Robotics - Edited by A. Casals; NATO ASI Series F50; Springer; Berlin; (1989).

[Groover, 1986] Mikell P. Groover, Mitchell Weiss, Roger N. Nagel, Nicholas G. Odrey; *Industrial Robotics, technology, programming and applications*; McGraw-Hill International Editions; (1986).

[Guedes, 1995a] Pedro Guedes; *Assembly Sequences and Assembly Operations*; ESPRIT Project 5629 - Presentation Day Proceedings; 27th October 1995; (1995).

[Guedes, 1995b] Pedro Guedes, Manuel S. Silva, Paulo J. Magalhães, Jorge B. Silva; *The CCP Assembly Cell Inplementation*; ESPRIT Project 5629 - Presentation Day Proceedings; 27th October 1995; (1995).

[Gutsche, 1995] R. Gutsche, F. Röhrdoriz and F. M. Wahl; *Assembly Planning Using Symbolic Spatial Relationships*; Graphics and Robotics - Edited by Wolfgang Straβer and Friedrich M. Wahl; Springer-Verlag; (1995).

[Hlupic, 1996] Vlatka Hlupic, Ray J. Paul; *Methodological approach to manufacturing simulation software selection*; Computer Integrated Manufacturing Systems; Vol. 9; No. 1;Elsevier Science Ltd.; (1996); pp. 49-55.

[Holland, 1995] Winfried van Holland, Willem F. Bronsvoort and Frederick W. Jansen; *Feature Modelling for Assembly*; Graphics and Robotics - Edited by Wolfgang Straβer and Friedrich M. Wahl; Springer-Verlag; (1995).

[Homem de Mello, 1991] L. S. Homem de Mello and S. Lee; *Computer-Aided Mechanical Assembly Planning*; Kluwer Academic Publishers; (1991).

[Homem de Mello, 1991a] L. S. Homem de Mello, A. Sanderson; *A basic algorithm for the generation of mechanical assembly sequences*; Computer-Aided Mechanical Assembly Planning; edited by L. S. Homem de Mello and S. Lee; Kluwer Academic Publishers; (1991); pp. 163-190.

[Hörmann, 1992] K. Hörmann *et al.*; *Task-level Programming*; Integration of Robots into CIM; Edited by R. Bernhardt, R. Dillman, K. Hormann, K. Tierney,; Chapman & Hall; (1992); pp. 122-167.

[Koivo, 1989] Antti J. Koivo; *Fundamentals for Control of Robotic Manipulators*; John Wiley & Sons, Inc.; (1989).

[Kovács, 1994] George L. Kovács, István Mezgár, Sándor Kopácsi, Daniela Gavalcová, János Nacsa; *Application of artificial intelligence to problems in advanced manufacturing systems*; Computer Integrated Manufacturing Systems; Vol. 7/No. 3; (1994).

[Kroszynski, 1994] U. Kroszynski, T. Sørensen, E. G. Schlechtendahl; *STEP Specification for Kinematics and Robotics CAD Data Exchange. Final Release: L02V00*; Neutral Interfaces in Design, Simulation and Programming for Robotics; Edited by I. Bey, D. Ball, H. Bruhm, T. Clausen, W. Jacob, O. Knudsen, E. G. Schlechtendahl, T. Sørensen; Springer-Verlag; (1994); pp. 243-312.

[Laloni, 1995] C. Laloni and F. M.Wahl; *Principles of robot simulation and their application in a PC-based robot simulation system*; Graphics and Robotics - Edited by Wolfgang Straβer and Friedrich M. Wahl; Springer- Verlag; (1995).

[Law, 1991] Averill M. Law, W. David Kelton; *Simulation Modelling and Analysis*; Second Edition; McGraw-Hill, Inc.; (1991).

[Lee, 1992] C. Lee; *Proceedings of the first workshop on assembly planning: theory and implementation*; Workshop of IEEE International Conference on Robotics and Automation; (1992).

[Lloyd, 1994] Bob Lloyd; *Robots in CIM*; CiME Computer Integrated Manufacture and Engineering; August September 1994; (1994).

[Lorenz, 1994] H. P. Lorenz, S. Haas; *KISMET - System description*; Neutral Interfaces in Design, Simulation and Programming for Robotics; Edited by I. Bey, D. Ball, H. Bruhm, T. Clausen, W. Jacob, O. Knudsen, E. G. Schlechtendahl, T. Sørensen; Springer-Verlag; (1994); pp. 64-69.

[Lu, 1996] Tien-Fu Lu, Grier C. I. Lin; *CAD, vision and sensor based intelligent robot server*; Computer Integrated Manufacturing Systems; Vol. 9; N. 2; (1996); pp. 91-100.

[Machado, 1995] J. A. Tenreiro Machado, Alexandra M. S. F. Galhano; *WinRob: An Educational Program for Robotics*; Proceedings of the Advanced Summer Institute '95 on Life Cycle Approaches to Production Systems; June 1995; Lisbon-Portugal ; (1995).

[Mäntylä, 1996] Martti Mäntylä, Dana Nau, Jami Shah; *Challenges in Feature-Based Manufacturing Research*; Communications of the ACM; Vol. 39; No. 2; February 1996; (1996); pp. 77-85.

[Martin-Vega, 1995] Louis A. Martin-Vega, Harold K. Brown, Wade H. Shaw and Thomas J. Sanders; *Industrial Perspective on Research Needs and Opportunities in Manufacturing Assembly*; Journal of Manufacturing Systems; Vol. 14/No. 1; (1995)

[Meijer, 1992] G. R. Meijer, V. Caglioti, U. Negretto; *Exception Handling*; Integration of Robots into CIM; Edited by R. Bernhardt, R. Dillman, K. Hormann, K. Tierney; Chapman & Hall; (1992); pp. 168-207.

[Mikosch, 1995] Falk Mikosch; *Precise Manufacturing with Off-line Programed Robots - Integration in Manufacturing Enables Innovative Robotics Solutions*; Opening Productive Partnerships; K.-R. von Barisani *et al.* (Eds.); IOS Press; (1995); pp.331-340.

[Müller, 1995] Heinrich Müller; *Using Graphics Algorithms as Subroutines in Collision Detection*; Graphics and Robotics - Edited by Wolfgang Straβer and Friedrich M. Wahl; Springer-Verlag; (1995); pp. 45-57

[Nielsen, 1992] L. F. Nielsen, S. Trostmann, E. Trostmann, F. Conrad; *Robot Off-line Programming and Simulation As a True CIME-Subsystem*; Proceedings of the 1992 IEEE International Conference on Robotics and Automation; Nice - France; May 1992; (1992); pp. 1089-1094.

[Okabe, 1985] S. Okabe; *Symbols for Assembly Systems*; Kanazawa University; Japan; (1985)

[Owens, 1994] J. Owens; *Robot and Fixture Calibration for Offline Programming*; Robot Simulations Ltd.; Newcastle-Upon-Tyne; (1994)

[Owens, 1995] J. Owens; *Offline Programming and Robot Calibration - The Car Industry*; Robot Simulations Ltd.; Newcastle-Upon-Tyne; (1995)

[Pahl, 1984] G. Pahl, W. Beitz; *Engineering Design: A Systematic Approach*; Design Council; London; (1984)

[Pezzinga, 1992] A. Pezzinga; *Applications at FIAR*; Integration of Robots into CIM; Edited by R. Bernhardt, R. Dillman, K. Hormann, K. Tierney,; Chapman & Hall; (1992); pp. 293-301.

[Philips, 1994] Everette Philips; *System Integrators for Robot Systems*; CiME Computer Integrated Manufacture and Engineering; August/September 1994; (1994)

[Purgathofer, 1995] W. Purgathofer, M. Zeiller; *CSG Based Collision Detection*; Graphics and Robotics - Edited by Wolfgang Straβer and Friedrich M. Wahl; Springer-Verlag; (1995); pp. 59-72.

[Ramos, 1993] Carlos Fernando da Silva Ramos; *Planeamento e Execução Inteligente de Tarefas em Robótica de Manipulação e de Montagem*; Dissertação de Doutoramento Apresentada à FEUP, DEEC; (1993).

[Rampersad, 1994] Hubert K. Rampersad; *Integrated and Simultaneous Design for Robotic Assembly*; Jonh Wiley & Sons, Ltd; (1994).

[Reinhart, 1995] G. Reinhart, D. Kugelmann; *Robot Handling in Case of Disturbances With 3D Simulation*; Proceedings of the Third IASTED International Conference Robotics and Manufacturing; June 1995; Cancun-Mexico; (1995).

[Requicha, 1996] Aristides A. G. Requicha; *Geometric Reasoning for Intelligent Manufacturing*; Communications of the ACM; Vol. 39; No. 2; February 1996; (1996); pp. 71-76.

[Rocha, 1995] João Rocha, Carlos Ramos, Zita Vale; *Geração e Representação de Planos para o Planeamento do Processo*; Robótica e Automatização; N. 21/22; Outubro 1995; (1995); pp. 31-35.

[Robinson, 1994] Stewart Robinson; *Applying Simulation to Manufacturing*; CiME Computer Integrated Manufacture and Engineering; August/September 1994; (1994)

[Santos, 1992] J. F. Oliveira Santos, L. Quintino; *Automatização e Robotização em Soldadura*; I.S.Q. Edições Técnicas; (1992)

[Schlechtendahl, 1994] E. G. Schlechtendahl; *Results Related to the STEP Standard*; Neutral Interfaces in Design, Simulation and Programming for Robotics; Edited by I. Bey, D. Ball, H. Bruhm, T. Clausen, W. Jacob, O. Knudsen, E. G. Schlechtendahl, T. Sørensen; Springer-Verlag; (1994); pp. 17-24.

[Schmiedecke, 1994] U. Schmiedecke; *Industrial Robot Language (IRL)*; Neutral Interfaces in Design, Simulation and Programming for Robotics; Edited by I. Bey, D. Ball, H. Bruhm, T. Clausen, W. Jacob, O. Knudsen, E. G. Schlechtendahl, T. Sørensen; Springer-Verlag; (1994); pp. 85-106.

[Schröer, 1992] K. Schröer, R. Bernhardt; *Program Execution*; Integration of Robots into CIM; Edited by R. Bernhardt, R. Dillman, K. Hormann, K. Tierney,; Chapman & Hall; (1992); pp. 115-121.

[Trostmann, 1994] E. Trostmann; *Interface Between CAD, Robot Programming and Simulation Systems*; Neutral Interfaces in Design, Simulation and Programming for Robotics; Edited by I. Bey, D. Ball, H. Bruhm, T. Clausen, W. Jacob, O. Knudsen, E. G. Schlechtendahl, T. Sørensen; Springer-Verlag; (1994); pp. 226-228.

[Takahashi, 1992] Tomoichi Takahashi, Hiroyuki Ogata; *Robotic Assembly Operation Based on Task-Level Teaching in Virtual Reality*; Proceedings of the 1992 IEEE International Conference on Robotics and Automation; Nice - France; May 1992; (1992); pp. 1083-1088.

[Vendrell, 1995] E. Vendrell and M. Mellado; *Robotic Tools and Experiences in Manufacturing Systems*; Proceedings of the Advanced Summer Institute '95 on Life Cycle Approaches to Production Systems; June 1995; Linbon-Portugal ; (1995).

[Wozny, 1996] Michael J. Wozny, William C. Regli; *Computer Science in Manufacturing*; Communications of the ACM; Vol. 39; No. 2; February 1996; (1996); pp. 33.

Anexo A

Peças produzidas no CCP e seus componentes

Apresentam-se neste anexo as imagens de duas das peças produzidas na Oficina Piloto do CCP, bem como imagens em que podem ser vistos os componentes destas peças e a forma como eles se juntam para constituírem o produto final.

As peças apresentadas têm as referências PD001 e PD002. O projecto destas peças é da autoria do Sector Mecânico do CCP e foram estas as peças produzidas com o objectivo de se demonstrar a operacionalidade da Oficina Piloto de Produção instalada pelo CCP no decurso do Projecto ESPRIT 5629.

1 Peça PD001

Figura A.1: Peça PD001

2 Explosão da peça PD001

Figura A.2: Explosão da peça PD001

3 Peça PD002

Figura A.3: Peça PD002

4 Explosão da Peça PD002

Figura A.4: Explosão da peça PD002

Anexo B

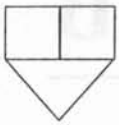
Símbolos normalizados para modelização de sistemas de montagem robotizados

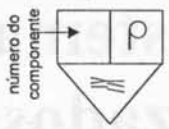
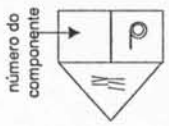
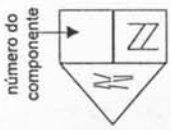
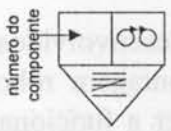
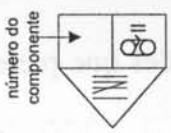
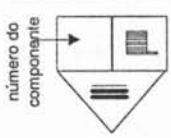
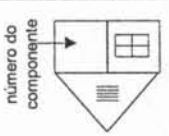
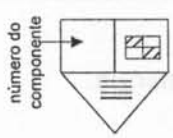
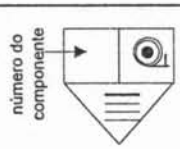
Apresentam-se neste anexo, o conjunto de símbolos desenvolvidos por Rampersad (1994) para os procedimentos de projecto de sistemas de montagem robotizados. Estes símbolos permitem esquematizar quer a estrutura do sistema, quer a funcionalidade de uma célula de montagem robotizada.

Estes símbolos foram divididos, de acordo com as funções que pretendem esquematizar, nas seguintes categorias:

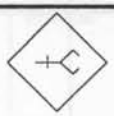
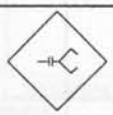
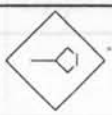
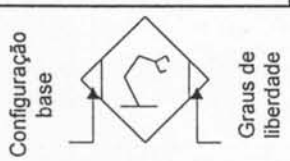
- Funções de alimentação;
- Funções de manuseamento;
- Funções de transporte;
- Funções de composição;
- Funções associadas com a alimentação e a composição;
- Funções de verificação e ajuste;
- Processos especiais;
- Funções associadas com o fluxo de informação e de energia.

1 Funções de alimentação

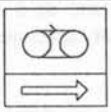
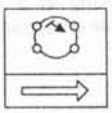
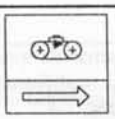
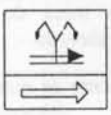
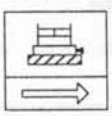


	Símbolo	Significado	Observações
Alimentação em volume		Alimentação com um alimentador de panela vibratória	Especifico do componente.
		Alimentação com alimentadores de panela vibratória empilhados	Especifico do componente.
		Alimentação com um alimentador vibratório linear	Especifico do componente.
Alimentação semi-organizada		Alimentação com um alimentador multifuncional de tapete duplo	As partes são alimentadas num tapete.
		Alimentação com uma correia transportadora equipada com um sistema de visão artificial	
Alimentação organizada		Alimentação com um armazém de empilhar	As partes são empilhadas num armazém.
		Alimentação com *paletes*	As partes são apresentadas em *paletes* e são indexadas em mais do que uma dimensão.
		Alimentação com *kits*	As partes são apresentadas num *kit* em que várias partes são armazenadas.
Fabrico *on-line*		Alimentação a partir de fita	Integrado com o fabrico das partes.

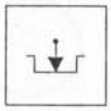
2 Funções de manuseamento

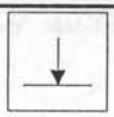
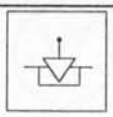
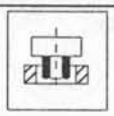
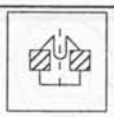
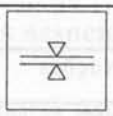
	Símbolo	Significado	Observações
Estratégia de manipulação 		Manuseamento com uma garra fixa	Manuseamento de um tipo de componente. Tempo de troca de garra nulo.
		Manuseamento com uma garra multifuncional	Manuseamento de 2 a 5 tipos diferentes de componentes.
		Manuseamento com um sistema multigarra	Manuseamento de 3 a 8 tipos diferentes de componentes.
		Troca de garras / ferramentas	Manuseamento de mais do que 8 tipos diferentes de componentes. Tempo de troca de garra / ferramenta elevado.
		Troca de dedos	Manuseamento de mais do que 8 tipos diferentes de componentes. Tempo de troca de dedos elevado.
		Manuseamento com uma garra universal	Manuseamento de mais do que 8 tipos diferentes de componentes. Tempo de troca de garra nulo. Em desenvolvimento.
		Complacência	A capacidade de complacência durante as operações de montagem. A = complacência activa P = complacência passiva
		Manuseamento com um tipo de robô particular	O tipo de robô é determinado pela configuração base e pelo número de graus de liberdade.

3 Funções de transporte

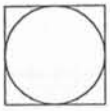
Símbolo	Significado	Observações
	Transporte manual	
	Transporte com um tapete rolante	Transporte não organizado.
	Transporte com um sistema de transferência indexável	Transferência sincronizada.
	Transporte com um sistema de transferência *power-and-free*	Transferência não sincronizada.
	Transporte com o robô	Micro transporte.
	Transporte com um sistema de AGV	Elevada flexibilidade no que concerne ao *routing* e à sequência de montagem.

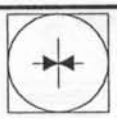
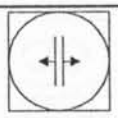
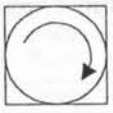
4 Funções de composição



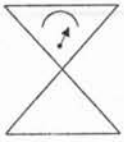
Símbolo	Significado	Observações
	Inserção	Junção através de inserção.
	Pressionar	Junção pela força.
	Aparafusar	Utilizando uma chave aparafusadora automática.
	Encaixes	
	Soldadura	
	Rebitagem	
	Ligação com adesivos	

5 Funções associadas com a alimentação e a composição

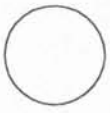


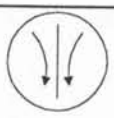
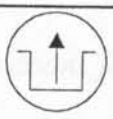
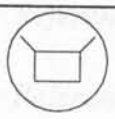
Símbolo	Significado	Observações
	Agarrar	Para fornecer estabilidade durante a operação de inserção.
	Largar	
	Virar	Alteração da orientação.

6 Funções de verificação e ajuste

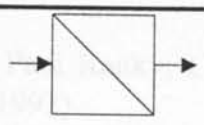
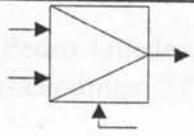
Símbolo	Significado	Observações
	Verificação	Verificação da qualidade do processo de montagem executado.
	Ajuste	Por exemplo, calibrar ou alinhar componentes do sistema.

7 Processos especiais



Símbolo	Significado	Observações
	Maquinação	Requer ferramentas especiais.
	Lavagem	Limpeza.
	Aquecimento	
	Desmontagem	
	Empacotamento	

8 Funções associadas com o fluxo de informação e de energia

Símbolo	Significado	Observações
	Transformação	Transformação da energia eléctrica em energia pneumática, hidráulica ou mecânica.
	Controlo do processo	Transformação de energia e informação.

Anexo C

Bibliografia

[1] - *Adept Three Robot User's Guide*; Adept Technology Inc.; (1993).

[2] - *Instructions for Adept Utility Programs, Version 10.3*; Adept Technology Inc.; (1993).

[3] - *V/V+ Reference Guide - Volume I*; Adept Technology Inc.; (1993).

[4] - *V/V+ Reference Guide - Volume II*; Adept Technology Inc.; (1993).

[5] - *Adept V/V+ Programming I Course Manual*; Adept Technology Inc.; (1993).

[6] - *Adept V/V+ Programming II Course Manual*; Adept Technology Inc.; (1993).

[7] - E. Appleton and D. J. Williams; *Industrial Robot Applications*; Open University Press Robotics Series; Edited by P. G. Davey; Halsted Press; (1987).

[8] - Elaine Rich; *Artificial Intelligence*; McGraw-Hill International Editions; (1983).

[9] - *IGRIP User Manual#1*; Deneb Robotics, Inc.; (1995).

[10] - Paul Ranky; *Computer Network for World Class, CIM Systems*, CIMWare Limited; (1992).

[11] - Pedro Guedes; *Assembly Cell Calibration*; ESPRIT Project 5629 - Presentation Day Proceedings; 27th October 1995; (1995).

[12] - *Indigo II WorkStation Owner's Guide*; Silicon Graphics Computer Systems; (1994).

[13] - *Iris Software Installation Guide*; Silicon Graphics Computer Systems; (1994).