FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Ricardo Dutra Rosado Pinto de Almeida

Pintura Robotizada com Reconhecimento Automático de Forma e Dimensões

Tese realizada para a Conclusão do Mestrado
Integrado em Engenharia Electrotécnica e de Computadores
Ramo de Telecomunicações, Electrónica e Computadores pela
Faculdade de Engenharia da Universidade do Porto

Presidente do Júri

Projecto realizado sob orientação do
Professor Doutor António Paulo Gomes Mendes Moreira
Professor Doutor Paulo José Cerqueira Gomes da Costa

Faculdade de Engenharia da Universidade do Porto
Departamento de Engenharia Electrotécnica e de Computadores
Março de 2007

Resumo

A pintura robotizada de pequenas séries é pouco frequente, devido ao tempo de programação ser muito elevado face ao tempo de produção. Nos sistemas actuais, quando se muda de peça é necessário reprogramar o robot com todos os custos associados a essa operação.

Neste projecto selecciona-se um problema-tipo e, em colaboração com uma empresa de aplicação de superfícies anti-aderentes (FLUPOL), pretende-se desenvolver métodos baseados em visão artificial, que possam tornar o sistema de pintura robotizada automaticamente adaptável a formas e dimensões diferentes sem necessidade de reprogramação.

Para que este sistema possa funcionar será criado um algoritmo com a função de criar trajectórias para pintar o interior de simples objectos. Pretende-se que estas tenham a capacidade de se modificar consoante a informação fornecida pelo sistema de visão artificial.

Abstract

The robotic painting systems on small series are uncommon because of the amount of time spent programming the system being very high compared to production time. In current systems, when there is an exchange of a piece, it is necessary to reprogram the robot with all the costs associated with this operation.

In this project we chosen a problem proposed by an industrial partner to automate the application of anti-adherent (Flupol). The aim was to develop methods based on artificial vision, which could make a robotic painting system automatically adaptable to different shapes and sizes without the need for reprogram.

To make this project work there were created some algorithms to generate trajectories capable of painting the interior of simple objects. These trajectories can change depending on the information received from the artificial vision system.

Nomenclatura Empregue

O UDP, *User Datagram Protocol*, é um dos protocolos do conjunto de protocolos da Internet (habitualmente designado por TCP/IP). Corresponde ao nível 4 do modelo OSI, visto ser um protocolo de transporte, sem ligação. Uma vez que é um protocolo orientado à transacção, é utilizado quando a entrega fiável de pacotes não é necessária, por exemplo em *streams* de áudio e vídeo. Os pacotes UDP têm pouco processamento, definindo apenas o número das portas e um protocolo de verificação de erros. Está definido no RFC 768.

O TCP, *Transmission Control Protocol*, é um protocolo de transporte orientado à conexão diferente do UDP (que não o é), porque verifica se os dados são enviados de forma correcta, na sequência apropriada e sem erros, pela rede. Está definido no RFC 793.

Um pacote é um conjunto de dados a serem transmitidos encapsulados numa mensagem maior.

O RFC, *Request for Comments*, são documentos que definem normas e protocolos para a Internet e onde se fazem as discussões de nível técnico para a definição de novos protocolos.

O RS-232 é um standard para a interligação série de dados binários entre um DTE (*Data terminal equipment*) e um DCE (*Data circuit-terminating equipment*).

O BSC, também conhecido como *Binary Synchronous Communications*, é um protocolo de comunicação introduzido pela IBM muito utilizado não só em computadores como em máquinas de levantamento automático de dinheiro e sistema de radar.

O paradigma cliente-servidor é usado em processos em que a aplicação servidora aguarda conexões, executa serviços e retorna resultados. Já a aplicação cliente é quem estabelece a conexão com o servidor, envia mensagens para o mesmo e aguarda pelas mensagens de resposta.

O código ASCII, *American Standard Code for Information Interchange*, é um código binário para representar caracteres em computadores. Os primeiros trinta e dois caracteres são caracteres de controlo para comunicação.

O BCC, *Block check character*, designa caracter(es) de controlo para verificar se houve erro de transmissão.

Um *Job*, tarefa, é o programa que o controlador executa para dar as instruções de movimentações ao manipulador.

O espaço de trabalho de um manipulador é a região do mundo que o robot pode alcançar através dos seus movimentos, onde pode levar a cabo as tarefas programadas.

A junta é a interligação entre dois segmentos de um manipulador, que permite o movimento relativo entre os mesmos numa única dimensão ou grau de liberdade.

O TCP, ponto central da ferramenta, corresponde à origem do sistema de coordenadas da ferramenta.

Índice

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Robótica é um assunto que não escapa ao interesse de ninguém, na medida em que se trata de um sistema usado em praticamente todo o tipo de ambientes, quer seja nas indústrias, em casa ou apenas para recrear.

De tal forma que já há algum tempo, o Homem começou a ser substituído por manipuladores, fazendo com que, em grandes empresas, máquinas e trabalhadores sejam colegas de trabalho.

Porém, o problema originado pela complexa programação e morosa implementação usada pelos manipuladores robóticos industriais tem vindo a ser debatido com frequência.

Nos dias que correm, os altos custos das tecnologias apenas sustentados em mercados de produção com largos volumes de produção e intensos capitais, não permitem um forte alastramento de aplicações robóticas nas pequenas e médias empresas.

Com o abandono das industrias com grandes produções em série dos países desenvolvidos, atraídas pelos baixos custos de produção oferecidos por países como a China, a Índia e alguns da Europa de Leste, as industrias que não migraram acabaram por ter de diversificar o seu produto.

Assim, nestas indústrias predominam pequenas séries de produção onde são fabricados produtos muito específicos, normalmente encomendados por clientes que pretendem algo único, o que na maioria dos casos significa baixo número de exemplares. Este aspecto, na maioria dos casos, não possibilita o uso de manipuladores robóticos na medida em que o tempo de programação destes torna-se muito significativo face ao tempo de produção.

Isto fez com que grande parte da industria portuguesa e europeia tenha parado de adquirir manipuladores chegando mesmo a parar de usar os que já tem, pois as empresas não estão interessadas em perder um dia de produção para programar o manipulador, para que esta ocorra durante um ou dois dias, que é o que acontece na maioria dos casos.

Para que os manipuladores se possam tornar numa vantagem para as pequenas séries de produção, é necessário desenvolver novos métodos e técnicas de programação para os manipuladores e dotá-los de ferramentas auxiliares que melhorem a sua percepção, tais como sensores baseados em visão artificial, sensores de força e pressão, entre outros.

Neste âmbito, pretende-se desenvolver um algoritmo para um sistema de pintura robotizada, baseado em visão artificial que possibilite que este se adapte automaticamente a diferentes formas e tamanhos, sem que seja necessário recorrer à reprogramação.

O sistema de visão artificial que se vai usar obtém imagens 3D através de um sistema de triangulação Laser/ Câmara (Ferreira *et al*., 2007), que fornece as dimensões de qualquer objecto que o atravesse.

Este trabalho tem o intuito de fornecer aos sistemas de pintura robotizados a capacidade de poderem ser utilizados em vários tipos diferentes de linhas de produção, uma vez que hoje em dia os sistemas de pintura robotizados quase só são usados na indústria automóvel, onde o manipulador robótico pode pintar a mesma forma de chassis durante anos.

O algoritmo a desenvolver vai ser criado para um sistema de pintura robotizado, mas com o intuito de ser genérico, ou seja, poder ser aplicado a outro tipo de tarefas

como polir, lixar e outras mais e, se ainda houver tempo, este algoritmo será convertido para uma linguagem genérica que possibilite que vários tipos diferentes de manipuladores consigam interpretar a informação enviada.

Tem vindo a crescer a importância de desenvolver interfaces intuitivas, que durante o processo normal de uso do sistema de controlo, possam esconder a complexa programação, de maneira a que o controlo do manipulador não necessite de operadores com altos conhecimentos para cada sistema específico, sem recorrer a grande dispêndio de esforço e verbas em treino de pessoal. Em resultado disto, a função deste trabalho também engloba a criação de um interface onde se possa introduzir o algoritmo criado.

1.1 Descrição do problema

As novas exigências do mercado forçaram a que as industrias se tornassem mais versáteis e por conseguinte aumentassem a variedade dos seus produtos. Esta variedade de produtos é necessária, já que cada vez mais os potenciais clientes necessitam de produtos na maioria das situações muito específicos onde o número de exemplares não é muito significativo.

As industrias que usam manipuladores robóticos perdem sempre algum tempo a reprogramar os Controladores para cada produto a ser fabricado e na maioria das vezes o tempo gasto é muito significativo face ao tempo reduzido de produção. Isto levou a que os manipuladores começassem a ser substituídos por mão-de-obra humana para realizar as tarefas para os quais estes foram adquiridos.

A causa que faz com que a programação dos manipuladores seja tão demorada, resume-se ao facto de esta ser feita através de um método do tipo aprender fazendo, ou seja, o programador faz o manipulador percorrer o trajecto pretendido, registando as várias posições onde o manipulador muda de direcção de maneira a que este posteriormente consiga refazer a trajectória passando por estas posições guardadas.

Esta paragem não é nada vantajosa para o fabricante, principalmente nos casos em que a encomenda é feita para um determinado produto, mas para o qual se pretende vários tamanhos. Isto implica várias reprogramações do controlador, o que é totalmente inadmissível.

Este problema sucedeu à pintura robotizada, questão em foco nesta tese, o que fez com que passasse a ser usada predominantemente na industria automóvel, onde uma trajectória de pintura implementada num controlador é capaz de estar em constante utilização durante anos até que o modelo do chassis que é pintado seja substituído por uma nova versão.

Actualmente quase todos os produtos saem de fábrica pintados. As indústrias que realizam a pintura destes produtos apenas o conseguem fazer manualmente, o que normalmente exige mão-de-obra qualificada, que em muitos casos seria mais rentável noutros departamentos. Já para não falar nos casos em que o manipulador consegue executar a tarefa muito mais rapidamente e com maior precisão quando programado devidamente, visto este nunca perder a concentração nem ficar exausto de realizar a mesma tarefa horas a fio.

Assim, o aumento da gama de aplicações dos manipuladores robóticos é um passo necessário para maximizar a utilização do mesmo e assim rentabilizá-lo. A introdução de sensores nos manipuladores e nas tarefas que eles efectuam é um passo neste sentido. No entanto, este facto tem de ser acompanhado pela aquisição e manipulação computacional dos resultados dos sensores de maneira a utilizá-los no processo industrial de fabrico.

Por conseguinte, o algoritmo de pintura a desenvolver tem que ser versátil ao ponto de se adaptar a diferentes formas e tamanhos. Também deve ser genérico de maneira a poder ser utilizado nas diversas tarefas que são similares e nos diversos controladores de cada marca.

No final será criada uma aplicação que servirá de interface com o controlador, permitindo a um utilizador controlar o manipulador e atribuir-lhe tarefas com enorme facilidade, pois só assim se conseguirá combinar o algoritmo desenvolvido e o sistema de visão artificial que fornecerá as dimensões dos objectos a pintar.

O sistema de visão artificial baseia-se num sistema de triangulação Laser/Câmara. Este sistema obtém uma imagem 3D do objecto que o atravessa e através da análise de imagem consegue obter as suas dimensões.

É esperado que a aplicação possibilite o ajuste dos manipuladores para os diferentes produtos e que o algoritmo permita que os manipuladores comecem a ser usados em todas as tarefas semelhantes à pintura.

1.2 Estado da arte

Actualmente, existe grande interesse em expandir a utilização dos manipuladores robóticos industriais a todas as áreas de produção, pois estes são capazes de realizar tarefas muito mais complexas e exigentes que um ser humano, principalmente quando combinados com sensores que possibilitem a percepção do mundo que os rodeia.

O grande problema é que cada tarefa é abordada de uma maneira individualista, e para complicar mais a situação, as plataformas usadas para programar os robots usam linguagens estruturadas e poderosas que resultam em processos muito complexos e morosos (Pires, *et al*., 2007a). Este processo exige muito tempo, preparação e um conhecimento minucioso de todos os detalhes relacionados com as operações de movimento e respectivas parametrizações, interacção com outros equipamentos, etc. Para que tal processo seja simplificado, vários autores têm proposto a utilização de vários equipamentos e processos de implementação que permitiriam uma maior proximidade entre os operadores e os equipamentos (Pires, *et al*., 2007a, Pires, *et al*., 2006a, Pires, *et al*., 2006b , Pires, *et al*., 2005b).

A grande maioria das soluções passa por interfaces que, em certa parte, escondem a linguagem complexa dos manipuladores e proporcionam ao utilizador um controlo fácil sem necessitar de ter conhecimentos muito aprofundados sobre o sistema de controlo que se está a usar. Como exemplo deste tipo de soluções, é muito frequente a utilização de ambientes de projecto e simulação em 3 dimensões para programar sistemas robotizados (Pires *et al*., 2004, Pires *et al*., 2005a, Pires *et al*., 2006a, Pires *et al*., 2007b), onde se pode utilizar o CAD do objecto para gerar as trajectórias que o manipulador deve executar ou o próprio operador criar estas trajectórias.

Outra das soluções muito comum é o uso de sistemas de visão artificial (Ferreira *et al*., 2007, Haddadin *et al*., 2007, Sepp *et al*., 2005, Suppa *et al*., 2007), que permitem que a informação obtida através deste sistema, ajude a programar o manipulador e até mesmo salvaguardar a integridade física dos operadores. Este tipo de sistema é o mais usado como auxiliar de controlo dos manipuladores, devido à sua grande capacidade de fornecer informação do exterior e devido aos algoritmos já existentes o que torna esta informação muito exacta.

Mas também existem outras propostas bastante interessantes mas que não muito utilizadas, como é o caso do uso da voz ou de um joystick. A voz não disponibiliza grande capacidade de controlo, mas não deixa de ser interessante (Pires *et al*., 2007b). Por seu lado, o joystick consegue simular com bastante exactidão as trajectórias que o manipulador deve percorrer, pois os dispositivos disponibilizados pelas consolas não fornecem uma sensibilidade muito elevada em termos de movimentação do manípulo e portanto, estes joysticks conseguem com grande facilidade fazer o manipulador seguir com grande exactidão qualquer tipo de superfície, mesmo aquelas que apresentam curvaturas (Colombo *et al*., 2006, Fusaomi Nagata *et al*.).

1.3 Estrutura do relatório

Este relatório foi estruturado em capítulos mediante a área abordada, estando

estes organizados segundo a ordem pela qual foram analisados e trabalhados.

Deste modo, no capítulo seguinte será feita uma introdução ao manipulador a utilizar, abordando os aspectos mais importantes que permitem a sua manipulação e movimentação.

O terceiro capítulo será dedicado à apresentação do sistema de comunicação disponibilizado pelo manipulador e o protocolo usado para que tal comunicação seja possível com um computador.

No quarto capítulo será apresentada a aplicação criada para controlar o manipulador robótico. Aqui também será apresentado o software utilizado para criar a aplicação e a linguagem que este usa e como está estruturado o código da aplicação, os algoritmos criados para a geração de trajectórias de pintura e por último, teremos expostos quais os atributos da aplicação dando-se indicações de como esta funciona.

No quinto capítulo é demonstrado como foi criado um job para executar todo o tipo de instruções de movimento mas não sem antes dar uma introdução à Linguagem INFORM.

No Sexto capítulo apresenta-se a Conclusão e nos últimos dois capítulos apresenta-se a Bibliografia e os Anexos.

Capítulo 2

Manipulador Industrial

Neste trabalho foi usado um robot industrial que, como o nome indica, é constituído por um robot (ou manipulador) e por um sistema de controlo que controla o robot, ao qual normalmente se atribui o nome de controlador.

O modelo do manipulador robótico que se vai usar é o YR-HP6-B10, fabricado pela Motoman. Este modelo é constituído por 6 eixos controlados e tem a capacidade de movimentar cargas até 6kg.

O controlador a usar é o modelo NX100 que também é fabricado pela Motoman. Este modelo permite o controlo de múltiplos manipuladores (até 4) e para que tal seja possível traz de origem, como interface homem-máquina, uma consola táctil.

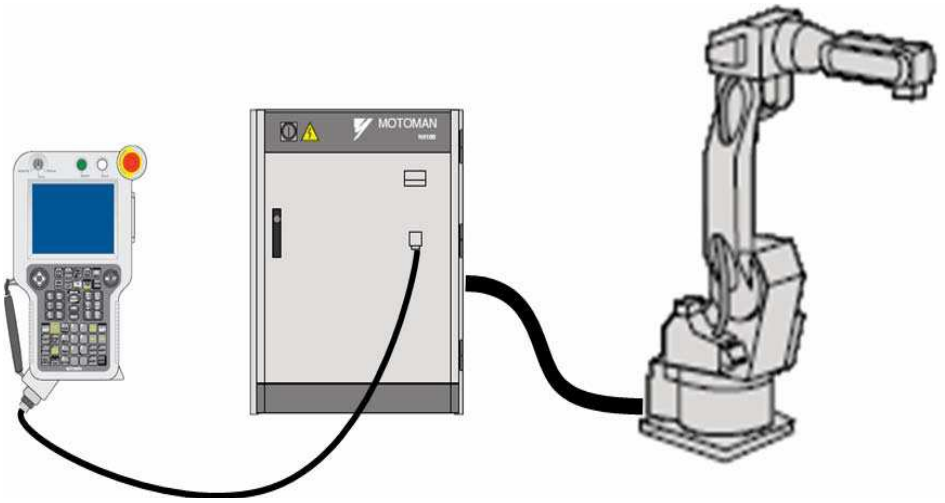


Figura 1: Ligações do controlador

Esta consola táctil possibilita a realização de inúmeras operações, sendo uma das mais relevantes a capacidade de movimentar manualmente o robot e posteriormente guardar as posições por onde o manipulador passou, para que o robot consiga sozinho recriar essa trajectória. A consola também torna possível a definição do TCP, a escolha do sistema de coordenadas a usar e a incorporação e controlo de ferramentas (tools) no manípulo.

Outra grande funcionalidade é a possibilidade de se poder programar e guardar trabalhos (jobs) usando para isso a linguagem criada pela marca, designada por INFORM, que já se encontra na versão 3, ou seja, INFORM III. Como não podia deixar de ser, visto que é a consola que normalmente controla o manipulador, esta possui dois botões de paragem de emergência.

Portanto o controlador é um computador dedicado que interpreta instruções e executa tarefas programadas através de algoritmos de controlo das juntas do manipulador.

Como nos dias que correm todos os computadores estão dotados de capacidade de comunicação via Ethernet, este controlador não fica atrás e disponibiliza um protocolo simples que permite que haja uma comunicação com um computador. O

computador pode enviar ficheiros de trabalhos (jobs), variáveis ou mesmo de comandos directos para o robot e pode receber todo o tipo de informação proveniente do controlador.

O controlador também está dotado de comunicação via protocolo RS-232, contudo nos dias que correm já não é nada atractivo, visto o melhor desempenho da Ethernet.

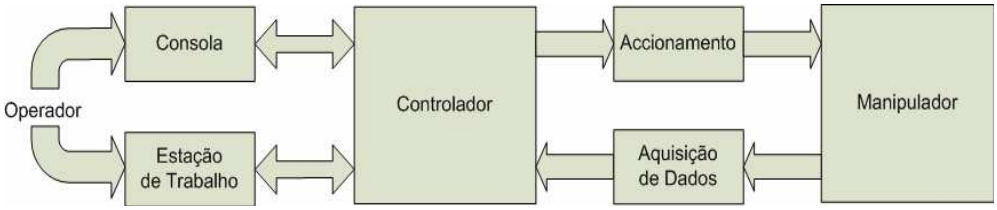


Figura 2: Esquema de interfaces do controlador

Neste capítulo serão apresentadas as funcionalidades disponibilizadas pelo controlador NX100 que poderão facilitar o desenvolvimento do trabalho a realizar.

2.1 Sistemas de coordenadas

Como já foi dito anteriormente, o manipulador usado é composto por 6 eixos, sendo o corpo (body) composto pelos três primeiros eixos, designados por S, L e U, e os restantes, que compõem a unidade do pulso, designados por R, B e T.

Conjugando esta informação com o facto de o controlador disponibilizar um leque variado de sistemas de coordenadas ou frames, o manipulador consegue executar movimentos de 360 graus, ou seja, é possível posicionar a ponta deste em qualquer direcção que seja pretendida.

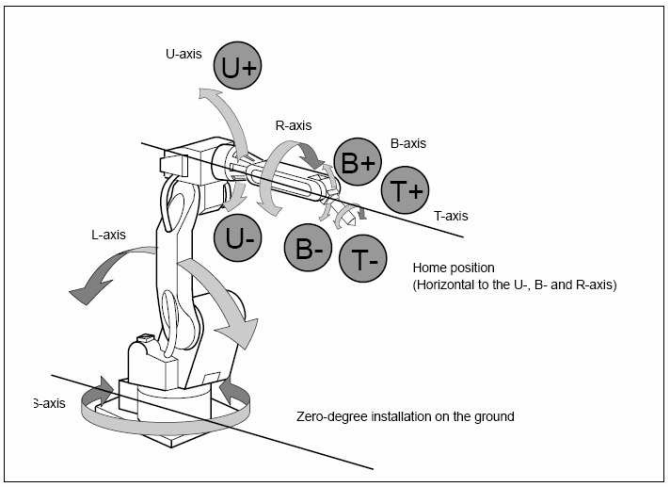


Figura 3: Representação dos eixos do manipulador

O sistema de coordenadas divide-se em dois grupos, sendo estes designados por referenciais externos e referenciais do manipulador.

Os referenciais externos são constituídos pelo referencial da estação de trabalho e pelo referencial da base. Cada um destes referenciais representa a direcção e o sentido que a estação de trabalho e o manipulador podem executar, respectivamente.

Os referenciais do manipulador dividem-se em cinco tipos:

1. Coordenadas das Juntas – “Joint Coordinates”

Neste tipo de coordenadas, a posição de uma junta é definida por um eixo de rotação que está associado a cada uma destas, cuja direcção é perpendicular ao plano de actuação. Como cada junta tem o seu eixo, é possível actuar sobre cada um individualmente, o que possibilita o movimento de cada junta de forma independente.

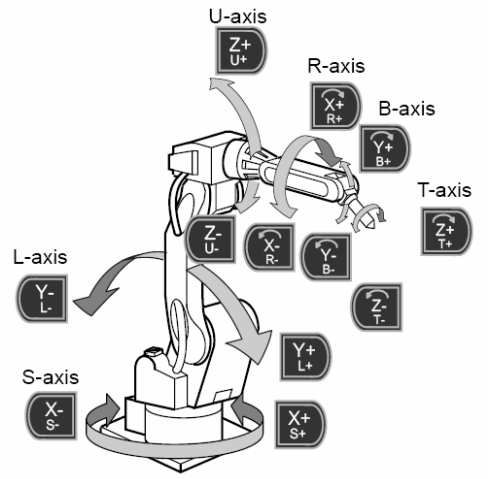


Figura 4: Sistema de coordenadas das juntas

2. Coordenadas Cartesianas – "Cartesian Coordinates"

Usando este sistema de coordenadas, o manipulador, independentemente da sua posição, move-se paralelamente aos eixos do referencial de base. Nos botões que realizam a rotação do pulso, que não é nada mais que a ponta do manipulador, pode-se realizar um movimento de reorientação em torno do TCP, segundo os eixos X, Y e Z.

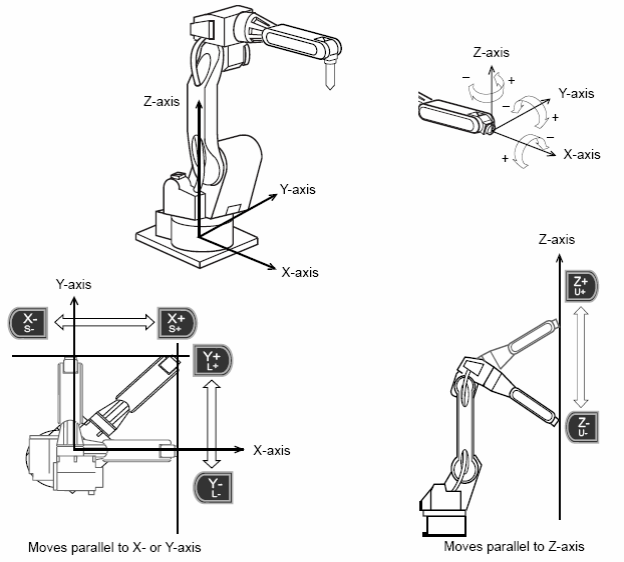


Figura 5: Sistema de coordenadas Cartesiano

.

3. Coordenadas Cilíndricas – "Cylindrical Coordinates"

Este sistema de coordenadas é composto pelos eixos r, θ e z. O eixo θ move-se em torno do eixo S do manipulador e o eixo r move-se paralelamente ao eixo L. Para os movimentos verticais, o manipulador move-se paralelamente ao eixo Z.

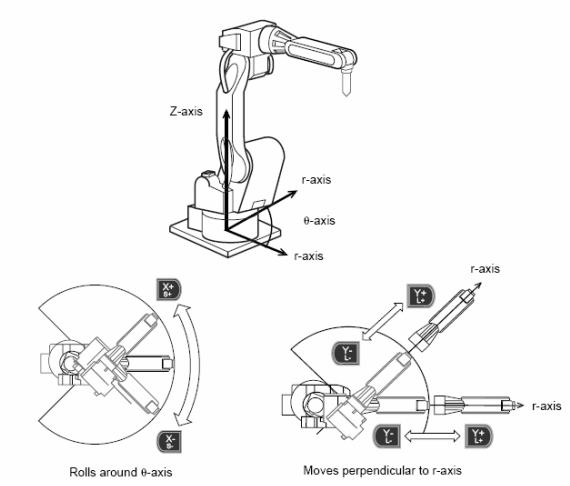


Figura 6: Sistema de coordenadas Cilíndricas

4. Coordenadas da Ferramenta – "Tool Coordinates"

Este sistema permite controlar as ferramentas de trabalho que são acopladas ao manipulador, mais propriamente à ponta do manipulador designada por punho.

Para que a ferramenta funcione na perfeição, é necessário definir o ponto de trabalho, que se designa por TCP. Este ponto, dependendo da característica do trabalho a realizar, poderá ser um ponto da própria ferramenta, ou então um ponto exterior a esta. No caso deste trabalho, pode-se considerar o ponto como estando localizado na ponta da pistola, mais propriamente o ponto por onde sai o spray de tinta, ou na ponta do spray, ou seja, no ponto de alcance do spray de tinta.

Pode-se então dizer que este sistema de coordenadas é o que controla as coordenadas do ponto de trabalho da ferramenta. A direcção da ferramenta montada no punho do manipulador é definida como o eixo Z. A partir daqui é possível realizar rotações do TCP em torno dos eixos da ferramenta.

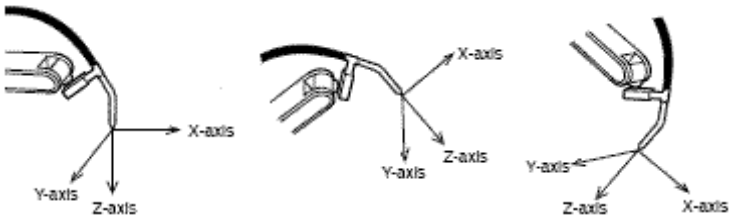


Figura 7: Definição do sistema de coordenadas da ferramenta

A definição do ponto de trabalho pode ser feita manualmente, definindo-se os parâmetros X, Y, Z e respectivas rotações da ferramenta em relação ao pulso, mas a melhor maneira para calibrar a ferramenta é através da funcionalidade de calibração do TCP que é disponibilizado pela consola. Para este efeito basta rodar, sobre o mesmo ponto o TCP, para as várias posições o mais ortogonais possíveis, como as que estão representadas na figura 2.8 e vai se gravando os valores dos eixos em cada uma das posições.

É muito importante que o ponto de trabalho seja bem definido, pois caso contrário podem acontecer falhas na execução de tarefas. No caso de uma ferramenta não estar bem definida e, por exemplo, ao se executar uma circunferência, quando o ponto de trabalho volta ao ponto de inicio, este ponto pode já não ser o ponto de inicio, pois como os vários eixos não coincidem, o ponto final pode estar mais ao lado do que o inicial e neste caso não se executou uma circunferência perfeita.

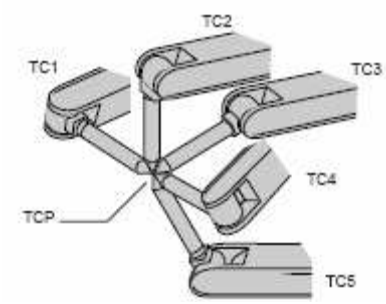


Figura 8: Calibração do TCP

5. Coordenadas de Utilizador – "User Coordinates"

Neste sistema, o manipulador movimenta-se, actuando na consola, paralelamente aos eixos X, Y e Z definidos pelo utilizador. Aqui pode-se criar um determinado plano diferente do convencional, onde as posições de actuação do manipulador são memorizadas neste sistema de coordenadas.

A utilidade torna-se visível, quando é preciso mover ou reorientar o plano onde estão localizadas posições de actuação, pois basta para isso, fazer a alteração no sistema de coordenadas criado. Desta forma não é necessário realizar o processo de obtenção de todas as posições de actuação para um novo plano, ou seja, evita-se o dobro do trabalho.

Outras operações onde este sistema simplifica a utilização do manipulador são, por exemplo, a utilização de várias estações de trabalho pelo manipulador sendo, nesse caso, definido um sistema de coordenadas para cada estação de trabalho ou, ao fazer operações de arranjos ou empilhamentos de paletes onde pode ser programado previamente o valor do incremento do desvio definindo o sistema de coordenadas na palete.

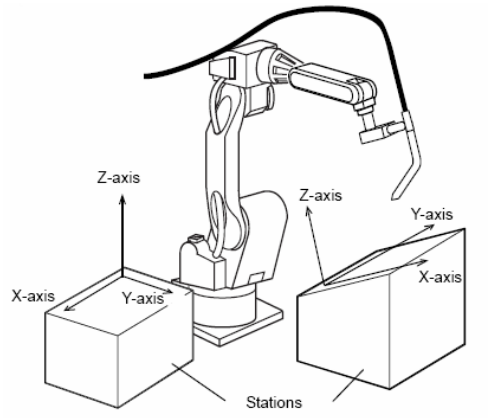


Figura 9: Exemplo do uso de vários sistemas de coordenadas de utilizador

As coordenadas do utilizador são definidas por três pontos que precisam de ser previamente ensinados ao manipulador através de outras operações de movimentação que são de seguida guardados no ficheiro de coordenadas de utilizador. Os três pontos que definem este sistema de coordenadas são ORG, XX e XY.

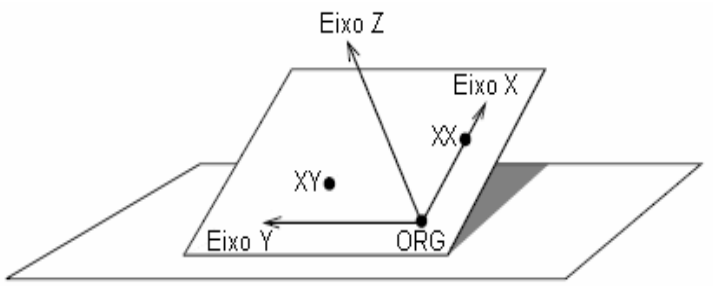


Figura 10: Definição das coordenadas do utilizador

Como é possível observar, ORG é a origem do referencial, sendo o ponto XX definido no eixo dos X. O ponto XY é definido no lado do eixo do Y do sistema de coordenadas do utilizador. A direcção dos eixos Y e Z são determinadas por este ponto.

2.2 Modo de Controlo

Na consola existe um botão que permite, ao utilizar, escolher o modo de controlo que quer usar. Existem 3 modos de Control:

1. Teach mode – neste modo é possível fazer operações sobre os eixos e editar programas através da consola;
2. Play mode – aqui é permitido executar os programas que acabaram de ser editados ou os que já estão guardados;
3. Remote mode – neste último modo, todas as operações são activadas por sinais externos provenientes, por exemplo, de um computador;

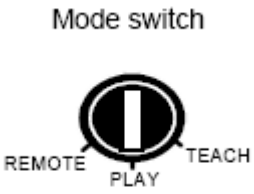


Figura 11: Botão para mudar o modo de controlo

2.3 Comandos de Movimento

Como podem observar na lista de comandos, que está exposta no anexo 1, quase todos os comandos de movimento podem ser usados por via remota, ficando fora deste grupo todos os comandos que envolvam curvaturas.

O MOVC é um dos comandos que não é possível executar remotamente, o que é uma pena, pois permite executar circunferências com uma enorme facilidade. A falta deste comando vai tornar certos movimentos mais complexos e difíceis de realizar aquando da realização da pintura. A resolução deste contratempo será exposta no capítulo 4.

Existem dois tipos de movimentos, sendo estes designados por Interpolação da Junta ("Joint Interpolation") e por Interpolação Linear ("Linear Interpolation"). A escolha do tipo de interpolação a usar é que determina a trajectória segundo a qual o manipulador se irá movimentar entre dois pontos a uma dada velocidade.

O primeiro tipo de interpolação é usado nos casos em que o manipulador não necessita de passar por um caminho específico para chegar à posição seguinte. Para se utilizar este tipo de interpolação, faz uma chamada da instrução MOVJ. Nesta instrução pode-se definir a velocidade VJ como percentagem da velocidade máxima definida.

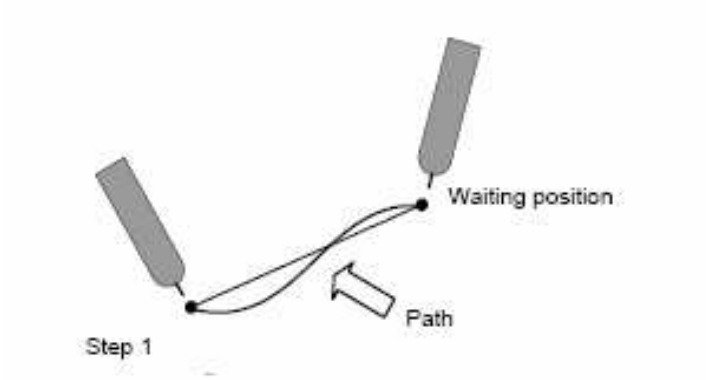


Figura 12: Trajecto entre dois pontos usando o movimento por interpolação da junta

No segundo tipo, o manipulador move-se segundo uma trajectória linear desde uma dada posição até à seguinte. Para que a execução do movimento seja perfeita o manipulador, enquanto se move, vai alterando automaticamente a posição do seu punho, como é mostrado na figura seguinte. Assim, consegue seguir a trajectória de uma maneira rigorosa.

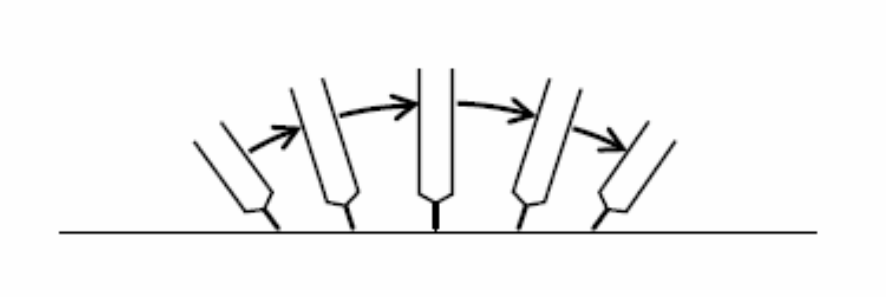


Figura 13: Trajecto entre dois pontos usando o movimento por interpolação da linear

Este tipo de interpolação é utilizado quando se usa a instrução MOVL ou IMOV. A velocidade linear é definida com V=xx em mm/s ou cm/min.

Pode-se então chegar à conclusão que é a interpolação linear que oferece melhores garantias de exactidão, quando se trata da execução de trajectórias pela parte do manipulador. Agora falta saber qual dos movimentos lineares deve ser usado.

Como as dimensões do objecto são conhecidas *a priori* através do processamento da informação fornecida pelo sistema laser/câmara, uma das opção passa por apenas posicionar a ponta da ferramenta num dado ponto inicial e daí ir dando instruções de incremento da ferramenta, para que toda a superfície do objecto seja percorrida. Outra opção consiste em obter pontos no espaço que representem a superfície do objecto e fazer com que a ponta do manipulador percorra esses pontos.

A instrução que permite o incremento da posição da ponta é o IMOV e a instrução que permite que a ponta consiga percorrer os pontos no espaço com um movimento linear é o MOVL.

Como inicialmente se tinha decidido que o sistema laser/câmara transmitiria apenas as dimensões do objecto a ser pintado, descartou-se a opção de usar a instrução MOVL, pois como não se tinha pontos no espaço que servissem como referência para a localização do objecto no espaço, não se conseguia determinar as coordenadas de todos os pontos por onde a ferramenta teria de passar.

Posteriormente, chegou-se à conclusão que o uso do IMOV apenas iria complicar as trajectórias de pintura e então decidiu-se usar a opção com MOVL,

alterando-se de seguida a informação fornecida pelo sistema laser/câmara que passou a ser quatro ou três pontos no espaço consoante o tipo de caixa.

O processo usado na opção IMOV, consiste em usar a instrução MOVJ para posicionar a ferramenta na posição inicial da trajectória e a partir dai apenas se usa o IMOV para incrementar a posição da ferramenta.

Na opção que usa o MOVL, usa-se também a instrução MOVJ para se atingir o ponto inicial e depois apenas se emprega a instrução MOVL para fazer com que a ferramenta percorra os pontos no espaço determinados pelo algoritmo.

No anexo 1 é possível ver que variáveis são necessárias manusear para que as várias instruções de movimento possam ser executadas correctamente.

2.4 Comandos

Para além das instruções mencionadas no capítulo anterior, existem muitas outras instruções que são bastante interessantes e que devem ser conhecidas. A maioria destas instruções será introduzida na aplicação, visto estar planeado que a aplicação seja capaz de controlar o manipulador, sem que seja preciso ter por perto a consola. Portanto, para acabar este capítulo 2, vai-se falar um pouco sobre algumas instruções para que se tenha uma noção do que cada uma é capaz de realizar.

- RPOSJ – lê a posição actual em que se encontra o ponto central da ferramenta no sistema de coordenadas das juntas;
- RPOSC – lê a posição actual em que se encontra o ponto central da ferramenta no sistema de coordenadas especificado no Comando;
- SAVEV – enviando esta instrução, o controlador irá responder com o valor da variável pedida pelo computador externo;
- SVON – permite ligar/desligar a alimentação dos motores das juntas do manipulador;

- START – a execução do programa (job) é iniciada do ponto onde foi encerrado na última utilização;
- MOVJ, MOVL, IMOV, PMOVJ, PMOVL – envia o comando para que uma determinada instrução de movimento seja realizada;
- LOADV – o controlador recebe e grava o valor de uma variável que foi enviado por um computador externo;
- JSEQ – abre um programa pretendido na linha especificada;
- HOLD – permite parar e voltar a arrancar do mesmo ponto uma tarefa que esteja a ser realizada pelo manipulador;
- RESET – faz um reset a um alarme que tenha ocorrido no controlador;
- CANCEL – cancela um erro que tenha ocorrido;
- MODE – permite seleccionar os modos de controlo descritos na secção 2.2;
- DELETE – permite apagar um determinado programa (job);

Capítulo 3

Comunicação

Neste capítulo, o assunto abordado é a comunicação via Ethernet, capacidade indiscutivelmente indispensável nos dias de hoje e que é disponibilizada pelo controlador NX100. Este é o protocolo usado neste trabalho.

Como já foi referido no capítulo 2, onde se expõe o manipulador industrial, o controlador também possibilita a comunicação via protocolo RS-232, mas como é óbvio este protocolo já está ultrapassado.

A grande vantagem da comunicação Ethernet sobre a porta série (RS-232) é a rapidez de transmissão, visto que o controlador robótico permite apenas uma velocidade máxima de transmissão de 9600 b/s, no caso da comunicação série, enquanto na comunicação por Ethernet é possível uma transmissão de 10Mb/s.

A necessidade de comunicação com o manipulador é evidente, visto que a utilização da consola para funções como a programação do manipulador pode tornar-se monótona e muito morosa, enquanto a programação num computador evolui a um ritmo mais acelerado e possibilita que haja alterações no código com mais facilidade.

O protocolo de comunicação entre o computador e o controlador já foi abordado noutro projecto (António *et al.*, 2007), o que nos deixa a tarefa de o adaptar para a nossa aplicação.

Assim, neste capítulo serão apresentados os modos de comunicação com o manipulador e explicada a implementação a efectuar para que tal aconteça.

3.1 Modos de comunicação

O protocolo Ethernet que o controlador do manipulador disponibiliza, baseia-se num protocolo do tipo BSC, onde todas as transmissões são efectuadas no modo ASCII.

Existem três modos de transmissão de dados:

1. O modo DCI (Data Communication by Instruction) permite que, quando um Job está a ser executado, certas instruções possam efectuar transmissão de dados para um computador, como por exemplo a transmissão de variáveis para posterior análise.
2. O modo Stand-alone serve para se comunicar com um computador através da utilização da consola de programação do controlador. Com esta função, é possível transmitir jobs e dados de condicionamento do manipulador (descrição de ferramentas, frames de utilizador, etc.).
3. O modo Host Control serve para transmissão de jobs, pedir informações sobre o controlador e manipulador e controlar o sistema através de envio de comandos de um computador.

Este último modo é o que iremos usar no nosso trabalho, pois este oferece-nos a capacidade de controlar o manipulador, enviando simples comandos que já foram abordados no Capitulo 2.

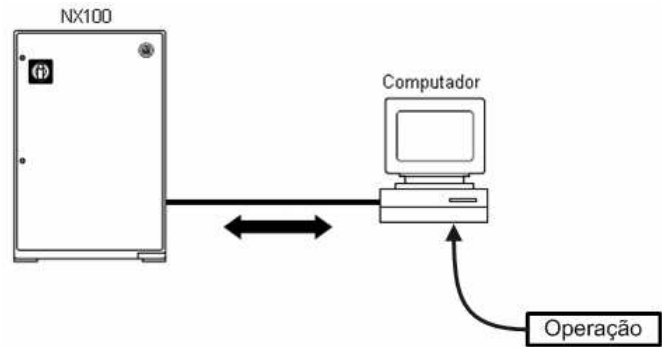


Figura 14: Modo de comunicação Host Control

Para que o computador possa comunicar com o controlador, na consola é necessário rodar o botão Mode Switch até que se atinja o modo remoto [REMOTE].

3.2 Configuração dos dispositivos

Para que se possa implementar a comunicação via Ethernet é necessário configurar o computador e o controlador. Visto não haver necessidade de maior expansão, foi instalada uma rede ponto a ponto.

Para a configuração do computador recorreu-se às propriedades TCP/IP da ligação da Área Local, mas na configuração do controlador, é necessário aceder ao modo de manutenção deste. Para tal que seja possível, é necessário entrar em modo de administrador e em seguida entrar no menu de configuração da ethernet.

As configurações efectuadas estão ilustradas na imagem seguinte.

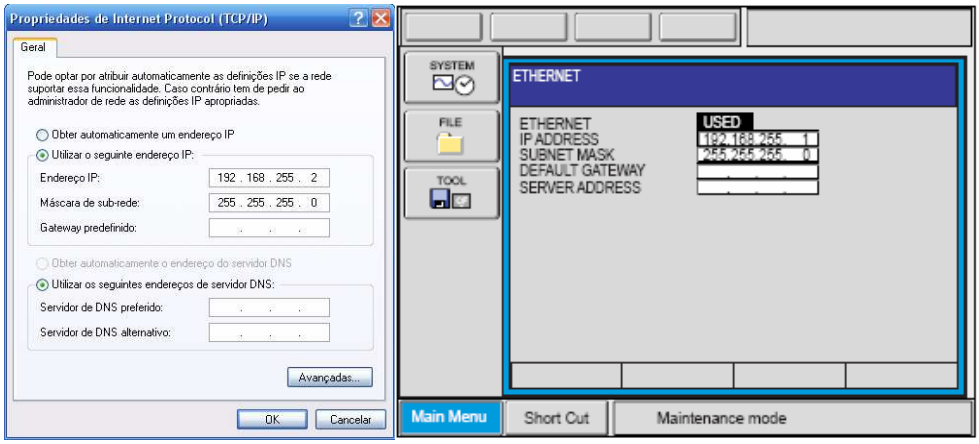


Figura 15: Configuração dos dispositivos

3.3 Comunicação Ethernet

Para se efectuar a comunicação via ethernet, pode-se usar um leque variado de protocolos.

O protocolo que foi usado na implementação da ligação entre o computador e o controlador, já disponível, foi o protocolo UDP.

Uma escolha destas não é surpreendente, pois este protocolo é o que oferece uma maior rapidez de comunicação, apesar de ter algumas falhas relevantes como a possível perda de pacotes ou mesmo a possível corrupção destes, originando consequentemente o descartar destes pacotes.

Algo que também ajudou na escolha não só foi o facto de a rede onde o controlador e o computador se encontram inseridos ser de pequena dimensão, o que minimiza as perdas que possam ocorrer, mas também o facto de o protocolo ser fácil de implementar, pois a estrutura dos pacotes criados são simples de criar e necessitam de menos tramas para efectuar a comunicação.

A arquitectura de ligação utilizada para a comunicação é ponto a ponto, pois ambos os dispositivos fornecem informação.

Para que a comunicação consiga ser efectuada, o controlador disponibiliza dois serviços, sendo estes, o serviço de supervisão e o serviço de transmissão de pacotes.

No primeiro serviço procede-se à ligação entre computador e controlador e posteriormente procede-se à quebra desta e no segundo serviço faz-se a transferência de pacotes BSC, onde se inclui a mensagem a transmitir.

Portanto, sempre que se envia um comando é estabelecida uma ligação, o envio deste e a desconexão.

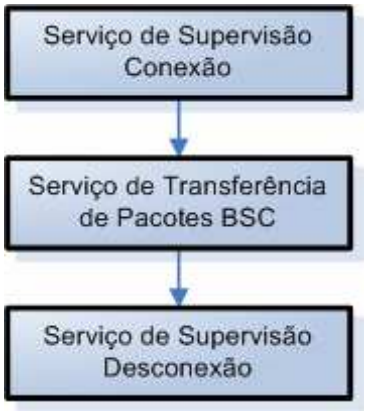


Figura 16: Serviços para comunicação ethernet

Os serviços utilizados estão localizados em portas diferentes. Assim, o serviço de supervisão está disponível na porta 10000 e o serviço de transferência BSC está localizado na porta 10006.

Como já foi mencionado, o serviço de supervisão é usado para iniciar e fechar a ligação. Através dele, o cliente faz o pedido de ligação e de início de transmissão de pacotes BSC, sendo que o servidor lhe dará as respostas a esses pedidos. Se as respostas a ambas as perguntas forem positivas, é iniciada a transmissão BSC.

Após a transmissão de pacotes BSC, é efectuado um processo semelhante para fechar a ligação, com o cliente a fazer o pedido de fim de serviço e de fim de ligação, sendo a resposta dada pelo servidor.

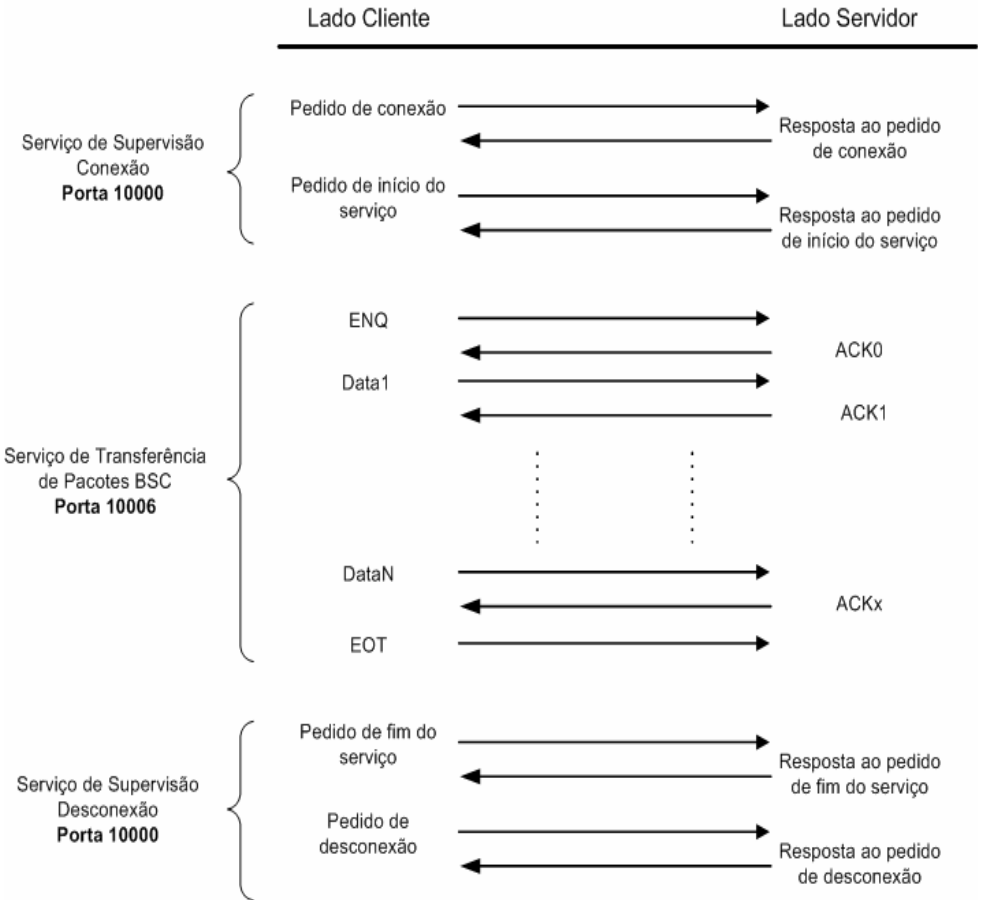


Figura 17: Detalhe da utilização dos serviços

Na figura anterior, é possível observar em maior detalhe o modo como as tramas são enviadas.

Todas as tramas, independentemente do serviço têm o mesmo formato, começando com um cabeçalho onde está incluído o tipo de serviço, 0x0000 para supervisão e 0x0006 para transmissão de pacotes BSC, e o comprimento da trama.

A figura seguinte ilustra o formato geral das tramas.

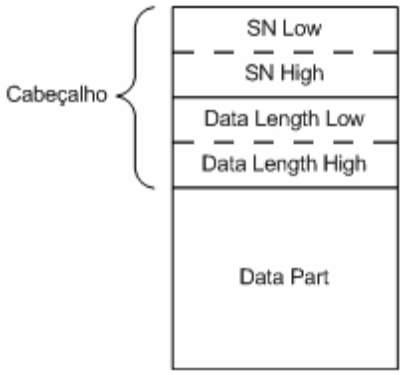


Figura 18: Formato das tramas

O bloco Data Part é a parte da trama que vai ser diferente, dependendo do serviço e do objectivo da trama, por exemplo, os pacotes BSC para serem transmitidos são encapsulados na trama dentro deste bloco.

Como seria de esperar, na transmissão de tramas do serviço de supervisão, a constituição do bloco Data Part varia dependendo do comando a enviar. De seguida, vai-se exemplificar a constituição do bloco para as diferentes tramas.

Tipo de Trama	***Bloco Data Part***
Pedido de conexão	Comando (0x0001)
Resposta a pedido de conexão	Comando (0x0001), Estado da ligação (0x0000/0xFFFF)
Pedido de início de serviço	Comando (0x0003), Serviço a iniciar (0x0006)

Resposta a pedido de início de serviço	Comando (0x8003), Serviço a iniciar (0x0006), Estado do serviço a iniciar (0x0000/0xFFFF)
Pedido de fim de serviço	Comando (0x0005), Serviço a parar (0x0006)
Resposta a pedido de fim de serviço	Comando (0x8005), Serviço a parar (0x0006), Estado do serviço a parar (0x0000/0xFFFF)
Pedido de desconexão	Comando (0x0002)
Resposta a pedido de desconexão	Comando (0x8002), Estado da ligação (0x0000 /0xFFFF)

Tabela 3.1: Constituição das tramas de supervisão

3.4 Serviço de pacotes BSC

Como já foi mencionado, a informação é transmitida usando um serviço de pacotes BSC. Este dispõe de vários tipos de tramas e estas variam dependendo da etapa em que se encontra a comunicação, ou seja, para que um comando seja enviado, é necessário que um encadeamento organizado de mensagens sejam enviadas, pois só assim se consegue uma autêntica comunicação, portanto sem erros nem falhas.

A mensagem que o computador usa para iniciar a comunicação é o ENQ e por seu lado a mensagem que o controlador robótico usa para terminar a comunicação é EOT. Como já foi referido, muitas mais mensagens são enviadas entretanto, uma delas contendo o comando que se pretende que seja executado pelo controlador.

Control Character	Code (hexadecimal)	Meanings of Control Character
DLE	10	Data Link Escape
SOH	01	Start of Heading
STX	02	Start of Text
ETX	03	End of Text
EOT	04	End of Transmission
ENQ	05	Enquiry
NAK	15	Negative Acknowledgment
ETB	17	End of Text Block
ACK0	10, 30	Even Affirmative Acknowledgment
ACK1	10, 31	Odd Affirmative Acknowledgment

Tabela 3.2: Caracteres de controlo de transmissão e o seu código em hexadecimal

A resposta aos pacotes será sempre ACKx, onde x será 0 ou 1 alternando sucessivamente. No caso de erro ou falha de transmissão, é enviado a mensagem NAK para pedir a retransmissão do pacote.

Como se pode observar na figura seguinte, a organização do pacote consiste num Cabeçalho antecedido por um sinal SOH, um bloco de Texto onde as mensagens são introduzidas e onde temos um sinal STX a anteceder e o bloco BCC que é antecedido pelos sinais ETX ou ETB consoante a mensagem necessitar de ser enviada em apenas um pacote ou em vários pacotes, respectivamente.

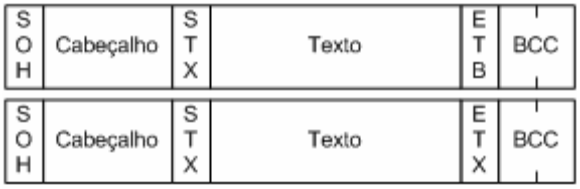


Figura 19: Pacotes de transmissão com cabeçalho

Na próxima figura é possível observar a constituição dos pacotes que são enviados, após a primeira trama ter sido enviada, para que as mensagens que necessitam de mais que uma trama possam ser enviadas na totalidade.

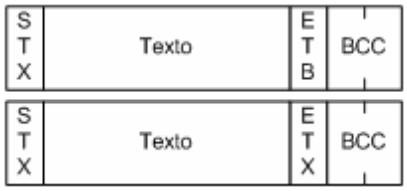


Figura 20: Pacotes de transmissão

O cabeçalho é usado para se introduzir que tipo de mensagem se está a enviar. Para que tal aconteça, este está dividido em duas partes que estão separadas por uma vírgula. A primeira parte deve conter o número do cabeçalho, que serve para indicar o que se pretende com a transmissão (Tabela 3.2), e a segunda parte deve conter o código do tipo de mensagem que está a ser enviado.

Número de Cabeçalho	Objectivo
01	Comando por computador
02	Transferência de ficheiro (Job, dados de condicionamento, informação do sistema)
03	Manipulação de variável (p. ex.: string, real, posição do manipulador)
04	Manipulação de sinais de entrada/saída

90	Resposta

Tabela 3.3: Propósito do número de cabeçalho

Como se depreende, para cada Número de Cabeçalho, podem ser pedidos vários tipos de informação. Por exemplo, pode-se querer transferir um job para o manipulador ou vice-versa e pode-se querer ler ou escrever uma variável. Para fazer essa distinção, é então utilizado o número de código. A lista de todos os cabeçalhos possíveis pode ser consultada no anexo 2.

O bloco de texto tem uma restrição imposta pelo protocolo que o limita a um máximo de 256 caracteres ASCII.

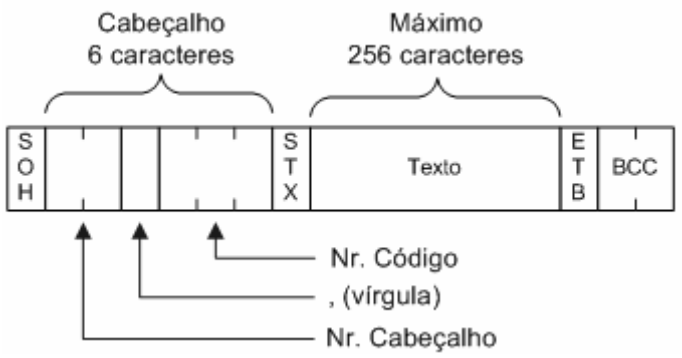


Figura 21: Esquema da trama

Para se fazer o controlo de erros de transmissão, calcula-se o BCC. Este cálculo cria um valor, que é representado por um conjunto de caracteres, que no acto de entrega do pacote é comparado com valor do BBC calculado no dispositivo de destino. Se ambos os valores forem diferentes o pacote apresenta um erro e tem que ser deitado fora.

Este cálculo é efectuado através da soma dos caracteres da trama. No caso de termos uma trama com cabeçalho, o cálculo efectua-se desde STX até ETX/ETB. Se a trama não tiver cabeçalho, STX não se inclui no cálculo.

Os caracteres BCC são formados colocando primeiro a metade mais baixa do byte BCC seguido da metade mais alta do byte BCC.

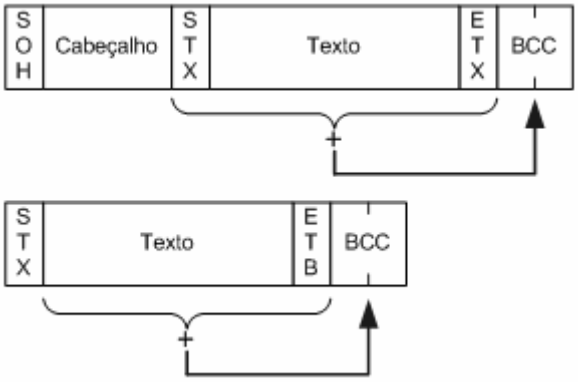


Figura 22: Cálculo do BCC

Para fazer o controlo da transmissão, existem dois temporizadores para proteger o sistema contra falhas na resposta. O primeiro monitoriza o tempo que demora a resposta, enquanto o segundo monitoriza o tempo que decorre entre a comunicação do controlador até à recepção da trama seguinte.

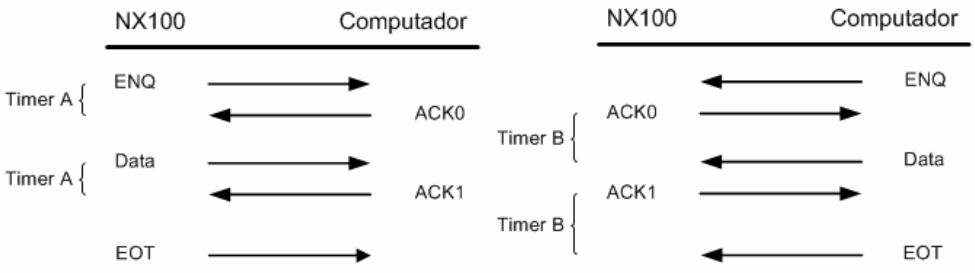


Figura 23: Temporizadores do controlador

Estão ainda implementados dois parâmetros para controlar o número máximo de reenvios dos pacotes no caso de ocorrerem falhas na transmissão, sendo que um controla o número de reenvios de NAK quando ocorre erro no cálculo do BCC e outro controla o número de reenvios de ACKx quando não se obtém resposta ou esta é inválida.

3.5 Implementação da comunicação

O módulo desenvolvido para comunicação foi dividido em duas partes, de acordo com os serviços disponibilizados para a transmissão. Assim, foi criado um módulo para fazer a comunicação do serviço de supervisão e outro com o serviço de pacotes BSC.

O serviço de supervisão é implementado de acordo com o modelo apresentado na figura seguinte.

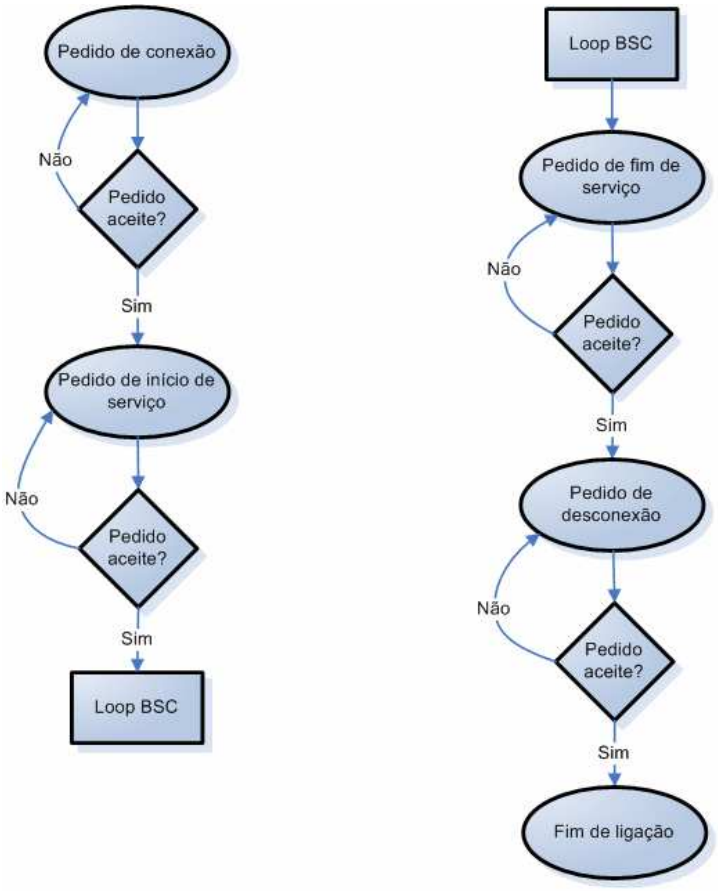


Figura 24: Implementação do serviço de supervisão

Na figura seguinte pode-se observar o protocolo usado para a transmissão de comandos de um computador para o controlador. O sistema para enviar ficheiros (jobs), que pode ser observado no anexo 3, é semelhante ao do envio de um comando, só que nessa situação, primeiro envia o nome do ficheiro e só depois envia o resto da informação nas outras tramas, dai usar o ETB mencionado anteriormente e o Número do Cabeçalho também ser diferente.

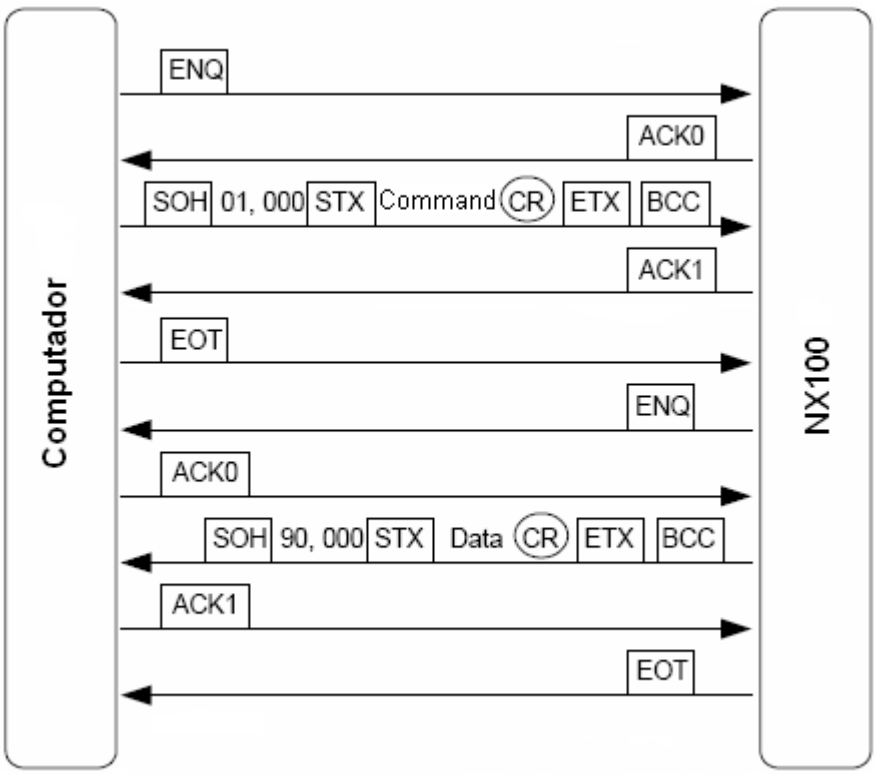


Figura 25: Serviço de pacotes para envio de comandos

Como se pode observar, o computador inicialmente envia três pacotes contendo as mensagens ENQ, o Comando a enviar e EOT. Após enviar toda a informação fica á espera de receber a resposta do controlador, que usa o mesmo sistema de três pacotes, para informar se o comando foi bem executado ou caso contrário qual o erro que ocorreu.

Se o comando foi bem executado o computador recebe uma mensagem 0000 em ASCII, caso o comando não peça nenhum tipo de informação ao controlador, mas se pedir, a resposta será a informação pretendida, também em ASCII. No caso em que o comando não é bem executado, o computador recebe uma mensagem que não 0000, podendo esta mensagem ser um número qualquer que equivale ao erro que ocorreu. A lista destes erros está disponível no anexo 4.

Como se pode observar, as mensagens EOQ e EOT servem para iniciar e terminar a comunicação de dados e as mensagens ACK0 e ACK1 servem para informar que as mensagens foram bem recebidas. Caso ocorra um erro na transmissão, as mensagens de resposta são substituídas por NAK, que induz o sistema a retransmitir a ultima mensagem.

Capitulo 4

Aplicação

Uma aplicação muito simples foi criada para controlar o manipulador robótico. A grande utilidade da interface apresentada pela aplicação consiste em fornecer informação aos utilizadores sobre as caixas que vão ser pintadas. É claro que também oferece a possibilidade ao utilizador de mandar o manipulador executar um leque variado de instruções, mas não é essa a ideia fundamental.

O interesse é sim fornecer informação ao utilizador de tudo o que se está a passar e apenas permitir que este possa realizar tarefas simples, como escolher que faces se quer pintar, ou a que velocidade a pintura deve ser executada ou que comando pretende enviar, sem que se tenha que preocupar com os detalhes que estão por trás das tarefas que serão realizadas.

Neste capítulo será então apresentada a aplicação desenvolvida e por conseguinte haverá uma breve introdução ao compilador que foi usado para a sua criação e posteriormente a descrição de como o código está organizado. Depois, será apresentada a aplicação onde se poderá ter a noção de como esta pode ser manuseada, do que é capaz de fazer e do tipo de informação que esta fornece aos utilizadores. Para finalizar será explicado os algoritmos de pintura criados para tal efeito.

4.1 Software de Desenvolvimento

O Software que disponibilizou a linguagem de programação usada para desenvolver a aplicação que controla todo o processo de pintura é o Delphi. Este software, produzido pela Borland Software corporation e um Ambiente de Desenvolvimento Integrado (IDE) para o desenvolvimento de softwares.

O nome Delphi é inspirado na cidade de Delfos, o único local na Grécia antiga em que era possível consultar o Oráculo de Delfos. O nome deve-se ao facto de os criadores do compilador procurarem uma ferramenta capaz de aceder ao banco de dados Oracle - daí o gracejo: "a única maneira de aceder ao oráculo é usando o Delphi".

A linguagem utilizada pelo Delphi é o Object Pascal, ou seja, Pascal com extensões orientadas a objectos. A partir da última versão do Delphi, que é a versão 7, passou a chamar-se Linguagem Delphi (“Delphi Language”).

O Delphi, apesar de ter sido criado para a plataforma Windows, posteriormente também começou a ser usado no desenvolvimento de aplicações nativas para o Linux e Mac Os, através do Kylix conhecido como Delphi para Linux, e para o Framework Microsoft Net nas versões mais recentes do Delphi.

O Delphi tem sido muito utilizado em quase todas as áreas mas com mais incidência na indústria, mais especificamente na área da robótica, da visão, entre outros. Esta escolha deve-se à maioria dos seus compiladores possibilitar programação orientada a objectos, mas também em grande parte devido à grande partilha de informação sobre a linguagem Delphi que existe na internet, onde é possível obter um leque variado de documentação e também de excertos de código, o que facilita imenso a rápida progressão no desenvolvimento de novas aplicações.

4.2 Estrutura do código

O compilador usado para desenvolver esta aplicação foi o Delphi 5. Apesar de esta versão já ser bastante antiga, já disponibiliza um leque bastante variado de bibliotecas, que permitem a criadores de aplicações a possibilidade de introduzirem vários tipos de funcionalidades às suas interfaces.

Como a versão usada já disponibiliza a programação orientada a objectos foi possível, com relativa facilidade, criar uma interface gráfica muito simples e prática para a aplicação que controla o manipulador robótico.

Outra grande vantagem é já disponibilizar uma variedade muito grande de Objectos, o que tornou possível implementar o protocolo UDP com bastante facilidade, visto que a classe TNMUDP possibilita que o uso deste protocolo seja adaptado com enorme comodidade a vários tipos de projectos.

Para o Protocolo de comunicação entre o computador e o controlador robótico, foi criado uma classe TPcToCr, onde todas as funções relacionadas com este processo foram introduzidas. É esta a Classe que contém o protocolo de comunicação exposto no capítulo 3 e portanto é esta que possibilita que a aplicação consiga enviar ficheiros e comandos e receber ficheiros.

Esta classe tem uma variável pública do tipo string, designada por MessToSend, que é o único parâmetro que lhe é necessário fornecer para que as tarefas para as quais foi feita sejam executas. A função da variável MessToSend consiste em indicar o nome do ficheiro ou o comando com os seus parâmetros à classe para posterior envio. O conteúdo do ficheiro que se queira enviar ou se queira receber, tanto é obtido através da caixa de texto MemoFile ou exposto nesta.

Toda a informação relacionada com esta comunicação, mesmo a resposta por parte do controlador robótico aos pedidos feitos pelo computador, é apresentada na caixa de texto MemoLog.

O algoritmo de pintura está contido na classe TPaintTrajectories e por conseguinte todas as funções que estão envolvidas no cálculo e tratamento da informação das trajectórias estão dentro desta classe. Esta função não apresenta nenhuma variável publica o que significa que todos os valores que a Classe necessita são fornecidos através dos parâmetros das funções aquando da invocação desta.

Por fim temos a Classe TFormPaint que contém todas as funções que estão relacionadas com a interface da aplicação. É esta a classe que contem todas as funções de controlo do manipulador robótico.

4.3 Apresentação da Interface Gráfica

Como já foi referido, a interface gráfica é muito simples e tem como função principal fornecer informação sobre a caixa que vai ser pintada. Na figura seguinte pode-se observar a janela principal da aplicação.

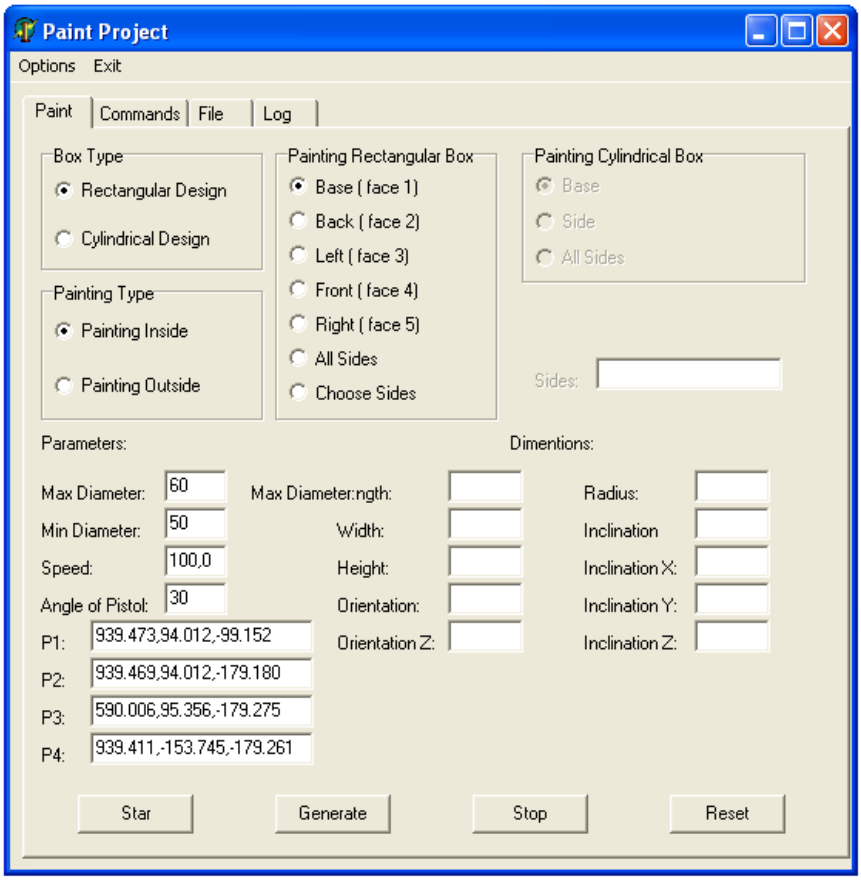


Figura 26: Janela Principal da Aplicação

Como se pode constatar, pode-se escolher que tipo de caixa se pretende pintar e depois disso, consoante a escolha efectuada, as opções de escolha aparecem. Neste caso, a pintura para caixa rectangular está seleccionada e portanto sobressaíram as opções para pintar caixa rectangular enquanto as opções para pintar a caixas cilíndricas praticamente que desapareceram e não podem ser seleccionadas. O mesmo acontece à informação que é mostrada pela aplicação, onde apenas as dimensões relacionadas com o tipo de caixa são apresentadas.

Visto que, para se pintar a caixa rectangular são precisos 4 pontos no espaço e para a caixa cilíndrica são precisos 3, fez-se com que a aplicação seleccionasse automaticamente o tipo de caixa a pintar aquando da recepção dos pontos provenientes do sistema laser/câmara, consoante o número de pontos fosse 3 ou 4.

Uma curiosidade que foi adicionada ao trabalho é a opção de se poder seleccionar se eventualmente se pretende fazer a pintura da parte externa da caixa ou da parte interna da caixa. Esta opção pode ser seleccionada nas opções “Painting Type”. Para que a pintura externa possa ser realizada, a caixa tem que se encontrar com a base virada para cima para que a pintura possa ocorrer correctamente.

É possível, para qualquer dos dois tipos de caixa, identificar que faces é que se pretende pintar. Quando se pretende pintar duas ou três faces de uma caixa rectangular, basta para isso seleccionar a opção “Choose Sides” e indicar na caixa de texto “Sides” o número das faces separadas por ‘;’.

Também é possível, em vez de realizar a tarefa de pintar a peça, que é iniciada ao se carregar no botão “Start”, gerar apenas os pontos no espaço e obter informações sobre as dimensões da peça, através de pressionar o botão “Generate”. O botão de “Stop” tem a funcionalidade de parar a execução de qualquer pintura em qualquer momento desta. O botão “Reset” tem a finalidade de fazer reset aos erros de movimentos apresentados pelo controlador. Em princípio não deve haver necessidade de recorrer a este botão, mas como o braço tem espaço de manobra limitado, nunca se sabe se o movimento pedido não vai exceder o permitido e o erro ocorrer.

Os valores que aparecem como parâmetros podem ser todos alterados, mesmo os pontos fornecidos pelo sistema laser/câmara. O “diameter” refere-se ao diâmetro do spray, o “speed” refere-se à velocidade e “Angle of Pistol” refere-se ao ângulo que a pistola de tinta usa para pintar as faces laterais das caixas.

Os pontos recebidos pelo sistema laser/câmara são expostos nas caixas de texto correspondentes a “P1”, “P2”, “P3” e “P4”.

No caso de se pretender enviar instruções para o controlador robótico, basta seleccionar o separador “Commands” que disponibiliza um leque variado de instruções. Caso não exista a instrução pretendida pode-se seleccionar “Other Command” e na caixa de texto introduzir a instrução que se pretende enviar. A seguir basta pressionar o botão “Send Command” para que a instrução seja enviada.

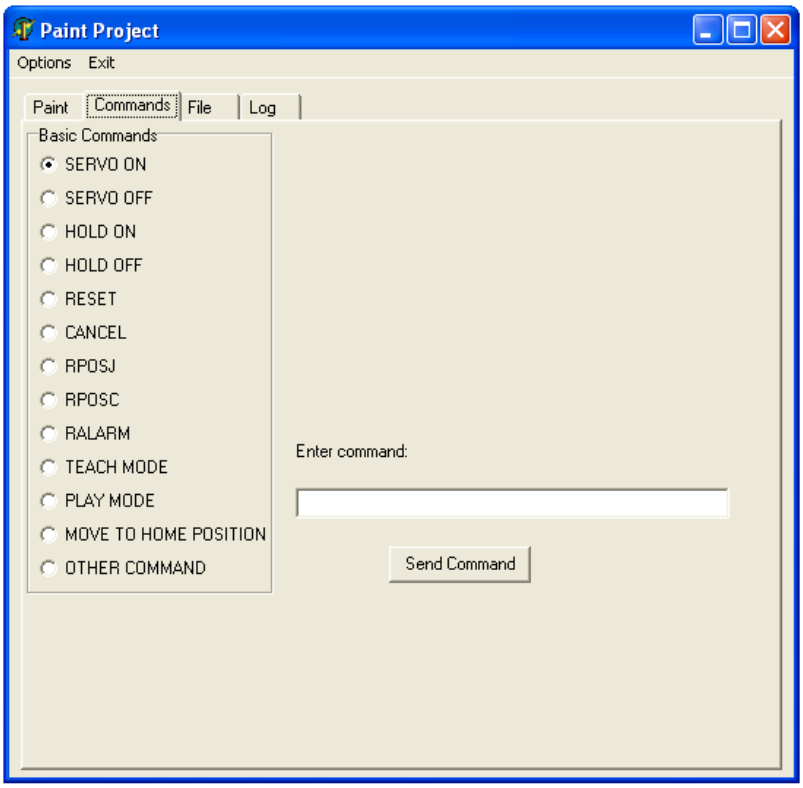


Figura 27: Janela de Comandos

Como para o segundo algoritmo foi criado e testado um job específico para executar, entre outros comandos, o comando MOVC, decidiu-se criar um separador "File" onde foi introduzido uma pequena janela de texto para se trabalhar os jobs que se pretende enviar para o controlador robótico. O botão "Load File" abre uma caixa de diálogo que permite seleccionar o ficheiro que se pretende trabalhar. O botão "Save File" permite guardar as alterações que foram feitas num ficheiro e o botão "Send File" permite enviar todo o texto que se encontra na caixa de texto para o controlador robótico.

Deve-se ter a noção que a aplicação apenas permite manusear ficheiros com extensão JBI, que são as extensões usadas pelos jobs, e portanto as únicas que são executadas pelo controlador robótico.

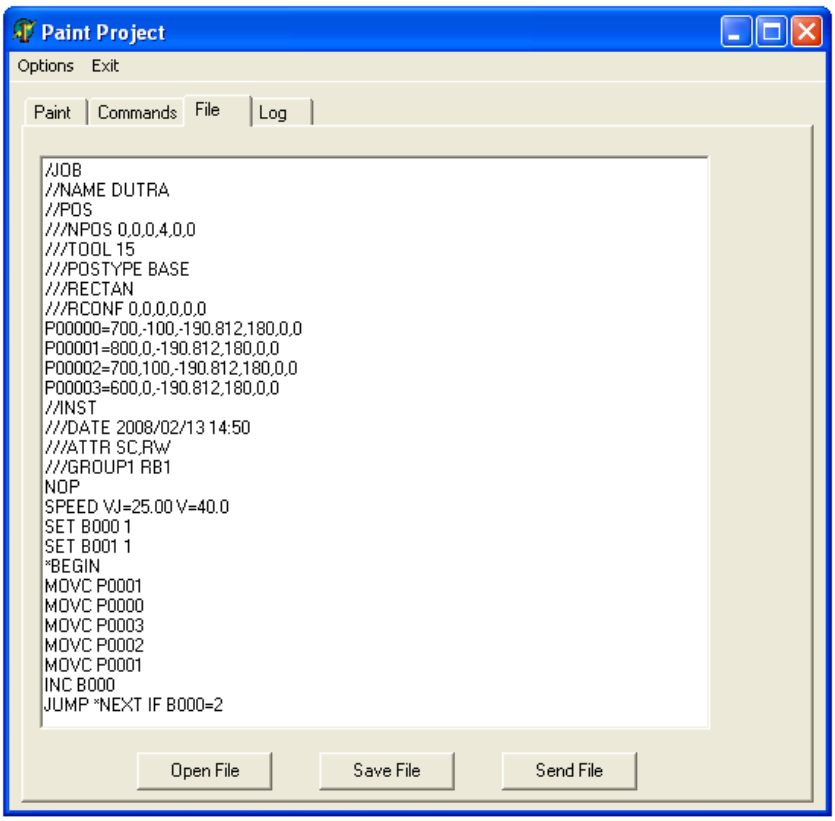


Figura 28: Janela de Ficheiros

No último separador denominado por “Log”, foi introduzida uma caixa de texto que apresenta toda a informação processada pela aplicação. Aqui pode-se ter a noção de que instruções foram enviadas para o controlador, que pontos no espaço foram criados para as trajectórias e que mensagens são recebidas por parte do controlador. Portanto é o local onde se pode procurar informação suplementar, principalmente nos casos em que por algum motivo a tarefa que se está a pedir para ser realizada não está a ser feita.

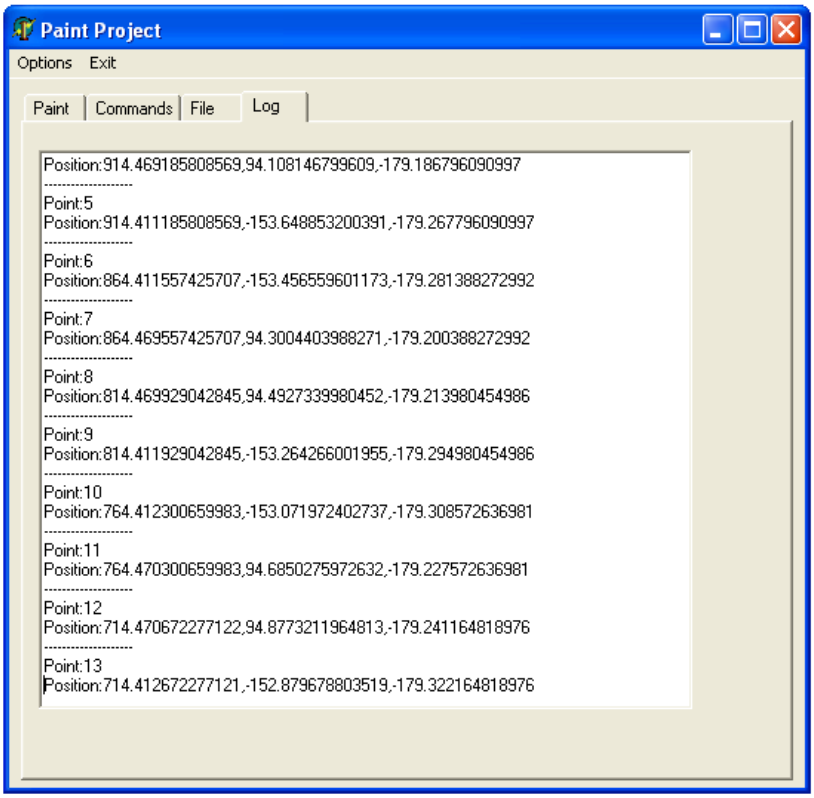


Figura 29: Janela de Registo

4.4 Trajectórias de Pintura

Para que o braço mecânico possa executar a tarefa de pintar, é necessário que lhe seja indicado que movimentos executar para que a tarefa seja concretizada.

Como neste trabalho a ideia principal é pintar o interior de alguns tipos de caixas, as trajectórias criadas foram todas pensadas com esse objectivo. Não se pretende pintar caixas com feitios muito sofisticados, como por exemplo o interior de uma caixa onde a base é quadrada e as faces laterais são semi-circulares.

As trajectórias desenvolvidas servem a necessidade de pintar caixas com formas paralelepipédicas ou cilíndricas. As características destas trajectórias serão agora apresentadas.

Como já foi mencionado anteriormente, inicialmente foram criadas trajectórias que usavam incrementos de posição como meio de percorrer toda a superfície a pintar. Inicialmente pareceu ser a opção mais favorável, mas os seus resultados vieram a mostrar-se decepcionantes devido à fraca capacidade de generalizar as trajectórias e à complexidade que se gerou no algoritmo.

Foi então que se decidiu abordar uma nova estratégia, que consistia em usar pontos no espaço que representariam os pontos por onde a ferramenta teria que passar para que a superfície a pintar fosse toda percorrida. Estas novas trajectórias mostraram-se muito mais vantajosas na medida em que adicionaram a capacidade à aplicação de poder pintar caixas que estivessem inclinadas e a capacidade de se poder escolher que faces é que se pretende pintar, características que não existiam com o algoritmo anterior e, apresentando uma complexidade de algoritmo muito baixo.

A ideia de criar os pontos no espaço é bastante interessante uma vez que pode permitir que esta aplicação possa ser adaptada a outros controladores robóticos, que não os da Motoman, com alguma facilidade, bastando que para isso seja alterado o protocolo de comunicação.

O controlo do manipulador aquando do uso do primeiro algoritmo consistia em enviar o comando IMOV para que este pudesse executar o incremento de posição da ferramenta. O controlador depois respondia com uma mensagem de confirmação que o comando tinha sido executado ou com um erro associado ao comando, caso este estivesse errado. Este processo era repetido sem pausas, pois o controlador realiza o incremento enquanto processa novos comandos que tenha recebido, até que a superfície fosse toda percorrida, quer tivesse a forma de um paralelepípedo ou de um cilindro. É claro que o incremento não passava de um movimento rectilíneo, o que levou a que fosse necessário adaptar as trajectórias para as formas cilíndricas a este problema.

No segundo Algoritmo manteve-se o mesmo sistema para as formas paralelepipédicas, mas alterando o comando para o MOVL pois agora movimentávamos a ferramenta para um ponto no espaço específico.

A grande alteração surgiu no caso em que as caixas são cilíndricas, pois como a instrução MOVC não estava disponível para o uso remoto, decidiu-se criar um job muito simples, que correria no controlador robótico, e que pediria os pontos no espaço à aplicação para posteriormente executar os MOVCs.

4.4.1 Base Rectangular

Para se pintar uma área quadrada existem vários tipos de trajectórias que podem ser consideradas. Mas as duas trajectórias que fazem mais sentido em usar são as expostas na figura seguinte.

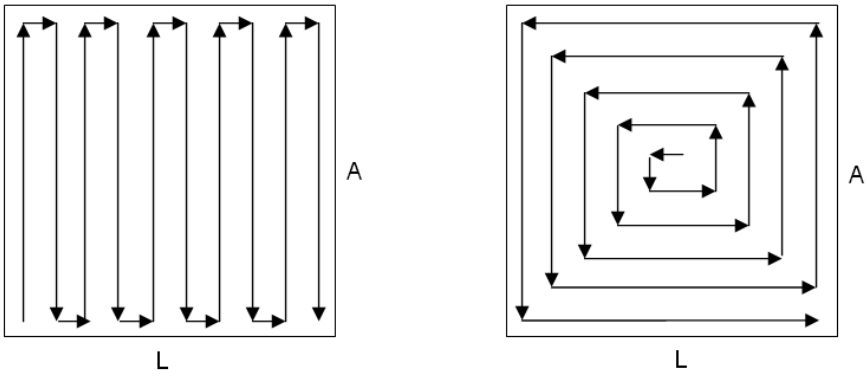


Figura 30: Exemplo dos tipos de trajectórias da base

O número de pontos (n) a serem percorridos pelo manipulador podem ser calculados para cada um das trajectórias da seguinte maneira:

1. Primeiro caso:

$$n = \left(\frac{L}{diâmetro\ do\ spray}\right) \times 2 \qquad (1)$$

2. Segundo caso:

$$n = \left(\frac{L + A}{diâmetro\ do\ spray}\right) \qquad (2)$$

Onde o L representa o comprimento da superfície e A a largura da superfície.

Sabendo que, por exemplo, a área é um quadrado com comprimento e largura de 10 cm, temos um número de pontos para ambos os casos com valor igual a 20 se o diâmetro do spray for igual a 1 cm. Portanto, à primeira vista ambas as trajectórias parecem estar em pé de igualdade, pois o número de pontos é igual.

Mas o problema no uso de uma das trajectórias mencionadas surge quando se considera uma área rectangular. Este problema pode ser observado na figura seguinte.

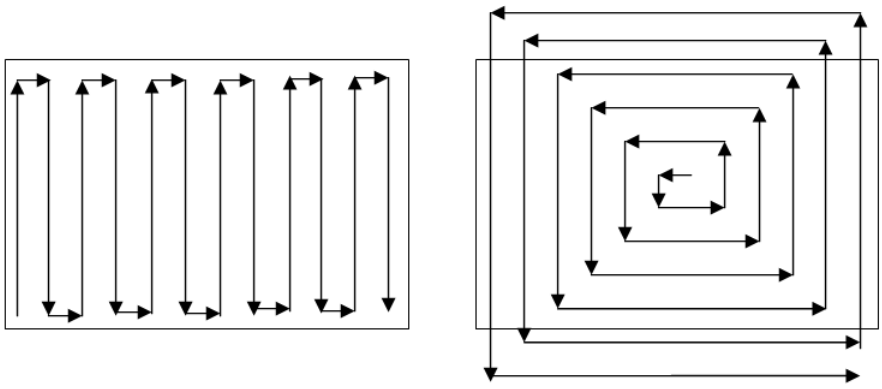


Figura 31: Resultado da trajectória quando a base é um rectângulo

A trajectória que vamos usar daqui para a frente em áreas rectangulares e quadradas vai ser a trajectória que representa o primeiro caso, visto que o segundo apresenta problemas quando a área da superfície é rectangular.

Seria possível usar a trajectória do segundo caso em situações em que a área é quadrada, mas ao se incorporar esta capacidade, o código a criar tornar-se-ia muito complexo, não trazendo nenhum tipo de benefícios.

O primeiro caso também tem uma característica muito valiosa, que é o facto de a trajectória poder ser alterada consoante o comprimento da superfície ser maior que largura ou vice-versa. Esta alteração é muito vantajosa uma vez que diminui o número de passagens que a pistola de tinta terá de efectuar sobre a superfície para que esta seja toda pintada.

Como se pode observar neste exemplo em que a área da superfície tem comprimento de 20 cm e 10 cm de largura e que o diâmetro do spray é 1 cm, a diferença do número de pontos é abismal. Com a alteração da direcção da trajectória conseguiu-se diminuir o número de pontos para metade. Esta possibilidade de alterar a direcção aumenta substancialmente a velocidade de cobrimento de superfícies rectangulares.

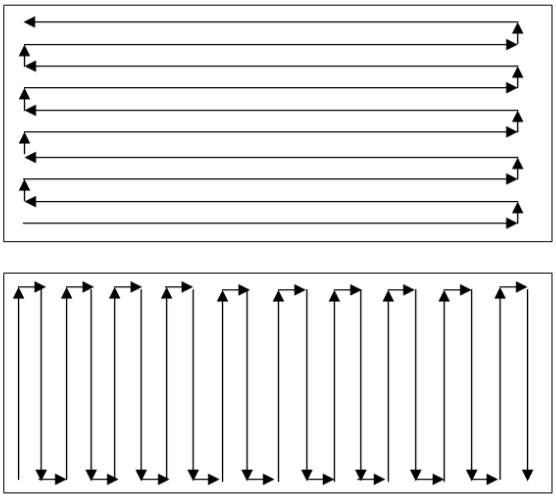


Figura 32: Exemplo de ajuste da trajectória no caso de o comprimento ser maior que a largura

Na primeira fase do trabalho tinha-se optado por usar incrementos para percorrer a superfície do objecto. Este sistema apenas precisava das dimensões do objecto, mas os resultados não foram os melhores. Isto porque em primeiro lugar era necessário inventar

um ponto inicial, visto que o sistema de visão artificial não forneceria nenhum tipo de referência que possibilita-se o cálculo de um. A falta de referências inviabilizou a possibilidade de apenas se pintar certas faces escolhidas pelo utilizador, visto que a pintura tinha que ser iniciada no ponto inicial e tinha que percorrer todas as faces até que a superfície fosse toda pintada. Isto até poderia ser feito se fosse indicada à pistola que não pinta-se certas partes do percurso, mas não seria propriamente prático. Outra desvantagem incidia no facto de não se conseguir pintar caixas com qualquer tipo de orientação nem qualquer tipo de inclinação, ou seja, seria necessário haver um sistema que posiciona-se as caixas para que o manipulador as pudesse pintar. A estas desvantagens juntou-se o facto de o algoritmo ter começado a ficar bastante complexo.

Consequentemente houve a necessidade de fazer alterações ao sistema todo. Então decidiu-se alterar a informação fornecida pelo sistema laser/câmara para quatros pontos no espaço, o que possibilitou que um algoritmo para criação das trajectórias muito mais simples fosse criado. Este algoritmo possibilitou também a fácil obtenção da posição de todos os pontos do percurso da trajectória no espaço, o que é bastante interessante. Criou-se portanto, um simples algoritmo usando a estrutura do anterior mas ao invés de gerar incrementos de posição para percorrer a superfície passou-se a criar os pontos no espaço para que estes fossem percorridos.

Este algoritmo conseguiu suplantar as falhas apresentadas pelo algoritmo anterior. Para além de conseguir pintar caixas com qualquer tipo de orientação e de inclinação, consegue pintar faces laterais escolhidas ao caso, gerando apenas os pontos necessários para pintar essas faces. Também consegue-se criar um ponto inicial por cima da caixa para onde o manipulador se deve deslocar no inicio de cada pintura, pois assim evita-se que este ao deslocar-se do ponto de repouso para pintar a caixa colida com esta. Para além disto consegue recriar formas circulares exactas, o que não era possível com o algoritmo anterior.

Para que se perceba as diferenças e onde está a vantagem em usar o segundo algoritmo, será primeiro mostrado como eram as trajectórias do algoritmo que usa incrementos de posição e depois as trajectórias do algoritmo que usa os pontos no espaço.

4.4.2 Caixa Paralelepipédica - Trajectórias com Incremento de Posição

Para este algoritmo duas trajectórias foram estudadas. Os dois primeiros passos das trajectórias são idênticos. Portanto ambas as trajectórias começam por pintar a base da maneira que foi demonstrada na secção 4.4.1.

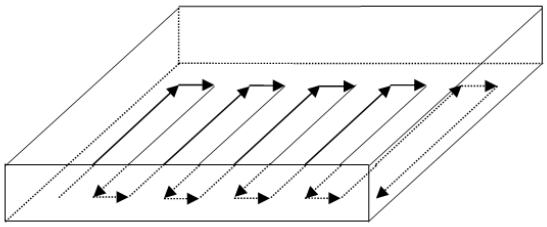


Figura 33: Pintura da base usando o 1º algoritmo

Depois de a base estar pintada, sobe-se a ponta da pistola de tinta, o equivalente ao raio do spray e roda-se esta para um ângulo suficiente, de modo a que o spray da pistola esteja apontado para as faces laterais. Este ângulo deverá variar de caixa para caixa pois o espaço de manobra que a pistola vai ter dentro destas pode variar consideravelmente.

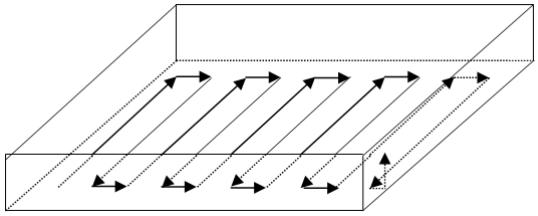


Figura 34: Posicionamento da pistola de tinta para a pintura das faces laterais no 1º algoritmo

A partir deste ponto, as duas trajectórias seguem caminhos diferentes. Seguiremos os dois casos separadamente começando por aquele que apresenta uma complexidade de movimentos mais elevado, pois necessita de rodar a pistola de tinta com mais frequência.

Após a pistola estar posicionada para as pinturas laterais, a pistola vai percorrer todas as faces laterais. Na passagem de face para face, a pistola executa dois movimentos num só, ou seja, enquanto se desloca em direcção ao canto esta vai rodando a sua ponta o equivalente a um ângulo de 45 graus. A mesma rotação é efectuada enquanto a pistola se afasta do canto. Assim a pistola executa uma rotação equivalente a 90 graus, o que permite que a ponta da pistola siga com bastante exactidão os cantos do objecto.

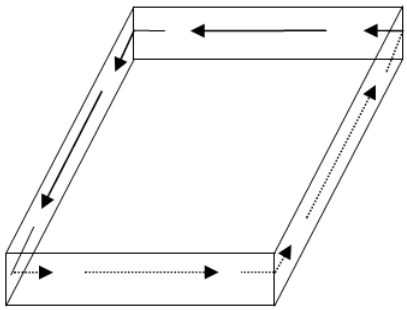


Figura 35: Movimento de pintura das faces laterais da primeira trajectória do 1º algoritmo

Após percorrer as 4 faces, a pistola sobe no eixo dos Zs o equivalente ao seu diâmetro do spray e percorre as arestas todas mas agora em sentido contrário. Uma sucessão de subidas e inversões de sentido é efectuado até que a pistola acabe de pintar todo o objecto.

O número de vezes que a pistola tem que executar o movimento ascendente n_{altura} é calculado dividindo a altura da caixa pelo diâmetro do spray da pistola:

$$n_{altura} = \left(\frac{altura\ da\ caixa}{diâmetro\ do\ spray} \right) \qquad (3)$$

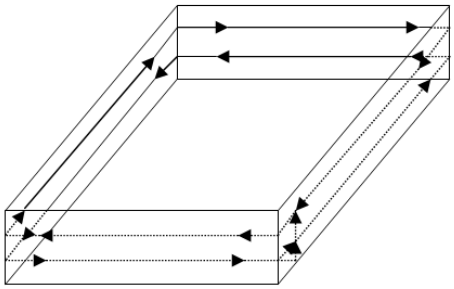


Figura 36: Pintura lateral usando a 1ª trajectória do 1º algoritmo

Agora passando para a trajectória seguinte, que foi suspensa na parte em que a pistola esta posicionada para pintar as faces laterais, prosseguimos a pintura executando uma trajectória igual à que se utiliza na base.

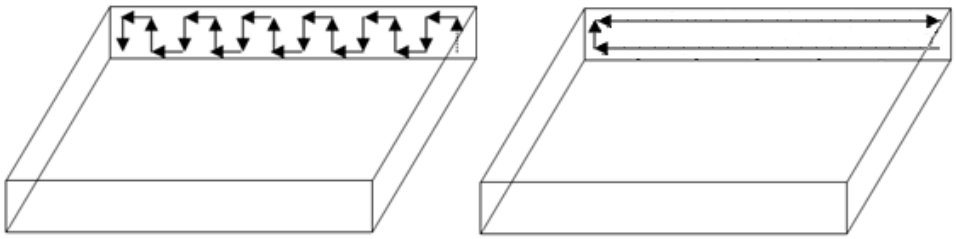


Figura 37: Pintura lateral usando a 2ª trajectória do 1º algoritmo

É possível observar na pintura lateral do lado direito da Figura 4.7, que minimiza o número de mudanças de direcção, que existe um problema. Pode haver situações como nesta que a haverá necessidade de se percorrer duas vezes o mesmo caminho, apesar de ser em direcções opostas, para se poder continuar a pintura. È claro que nestas situações se desligaria a pistola para não haver desperdício de tinta e para evitar que a caixa fica-se ensopada.

De seguida, roda-se 45 graus a ponta da pistola e executa-se um movimento ascendente ou descendente, dependendo da posição da pistola, para pintar o canto e volta-se a rodar 45 graus para posicionar a pistola na outra face. Aí executa-se a pintura como foi efectuada na face anterior.

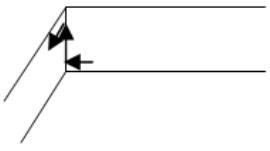


Figura 38: Pintura do canto usando a 2ª trajectória do 1º algoritmo

O processo de pintura lateral repete-se até que todas as faces estejam pintadas.

Esta trajectória é mais rápida a executar a pintura lateral do que o trajectória anterior. Isto é devido ao facto dos movimentos de rotação executados pela pistola, que nesta trajectória são pouco usados, serem feitos com uma velocidade inferior aos movimentos lineares. Esta lentidão é imposta pelo manipulador e não há nada que se possa fazer, a não ser, estipular uma velocidade maior para esse tipo de movimentos. Portanto a constante rotação em 45 graus executada pela pistola no primeiro método torna esse mesmo método mais lento que o segundo.

Para que todos os tipos de caixas possam ser pintados foi necessário adaptar estas trajectórias. Foram então criadas algumas vertentes dessas trajectórias de maneira a que estas se conseguissem adaptar a todas as dimensões.

As várias vertentes começam por ser distinguidas pelas várias posições em que a pistola se pode encontrar quando termina a pintura de uma base rectangular. Estas posições podem ser observadas na figura seguinte.

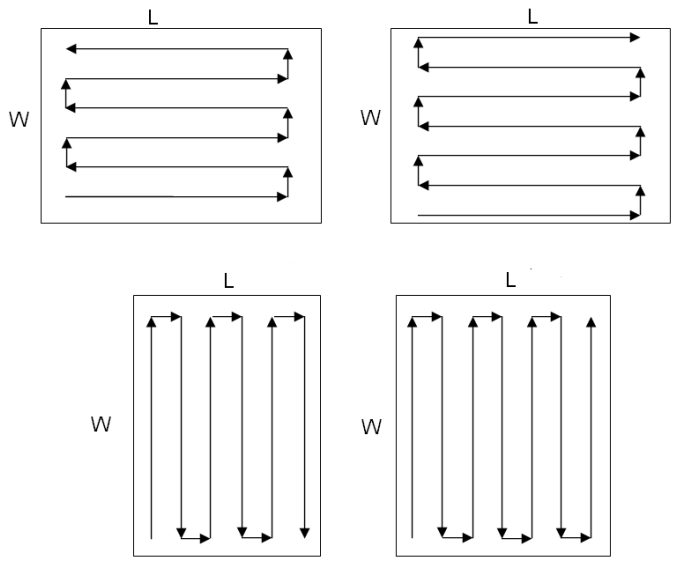


Figura 39: Trajectórias possíveis para a base

Como se pode observar, de todas as possíveis combinações, a pistola pode acabar a primeira fase da pintura numa de três posições. Consoante cada uma das posições, a trajectória a seguir por cada um dos métodos, mencionados anteriormente, tem que ser adaptada. Para a primeira trajectória, as várias vertentes da trajectória são as seguintes.

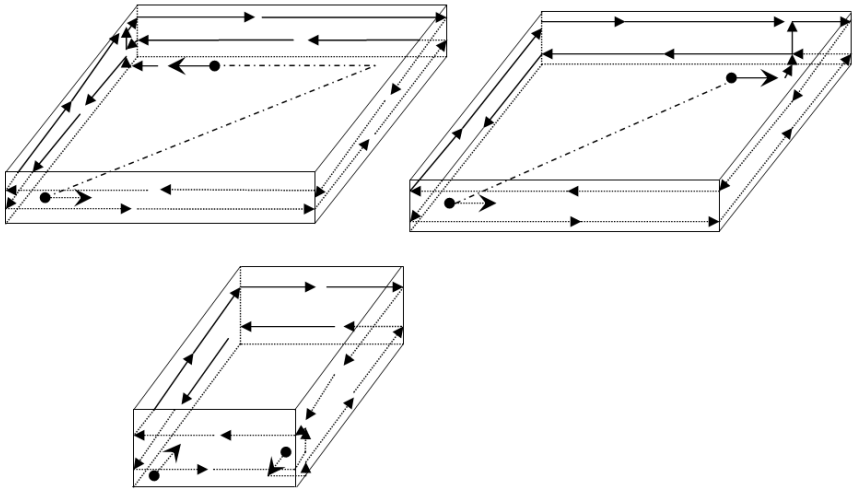


Figura 40: Possíveis caminhos percorridos na 1ª trajectória do 1º algoritmo

Estas várias versões, que têm de ser incluídas, tornam a implementação desta trajectória muito complexa, mas não tanto quando comparando com a segunda trajectória, pois neste caso, para além de a posição da pistola poder variar na base, também pode variar nas faces laterais. As várias vertentes podem ser observadas na figura seguinte.

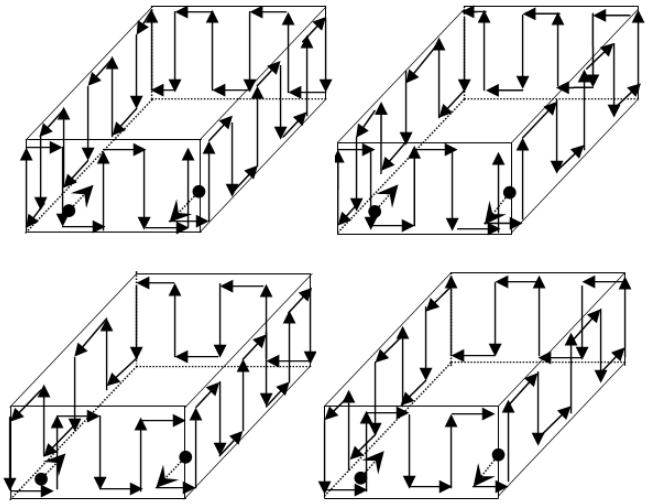


Figura 41: Possíveis caminhos percorridos na 2ª trajectória do 1º algoritmo

Consegue-se observar que a complexidade deste primeiro algoritmo é bastante elevada e nem sequer estão todas as possibilidades expostas. Isto levou a que este algoritmo fosse posto de lado e que fosse necessário encontrar uma nova solução para o trabalho em mãos.

4.4.3 Caixa Paralelepipédica -Trajectórias com Pontos no Espaço

Ao usar-se os quatro pontos no espaço para determinar as trajectórias, o sistema tornou-se muito mais simples. A primeira simplificação foi o facto de apenas se usar a trajectória de pintura da base para pintar todas as faces da caixa. Como se conhece os pontos no espaço, é fácil indicar quais são os pontos que pertencem a uma dada face lateral e ordenar, para que esses pontos sejam percorridos pela pistola de pintura. Por conseguinte este sistema permite que se possa escolher que faces se pretendem pintar.

Este atributo permite a execução de tarefas interessantes, como a de pintar caixas nas quais faltem algumas faces ou mesmo pintar caixas com várias cores, se for possível alterar a cor da pistola de tinta. Este algoritmo ainda possibilita a opção de se poder pintar a parte interna da caixa ou a parte externa da caixa o que é bastante interessante e pintar caixas que estejam inclinadas e com qualquer orientação.

A única parte que é um pouco complexa, é a obtenção dos pontos no espaço, que englobam uma combinação de cálculos geométricos que vão ser expostos de seguida.

Na figura seguinte pode-se entender qual é a posição no espaço dos quatro pontos fornecidos pelo sistema laser.

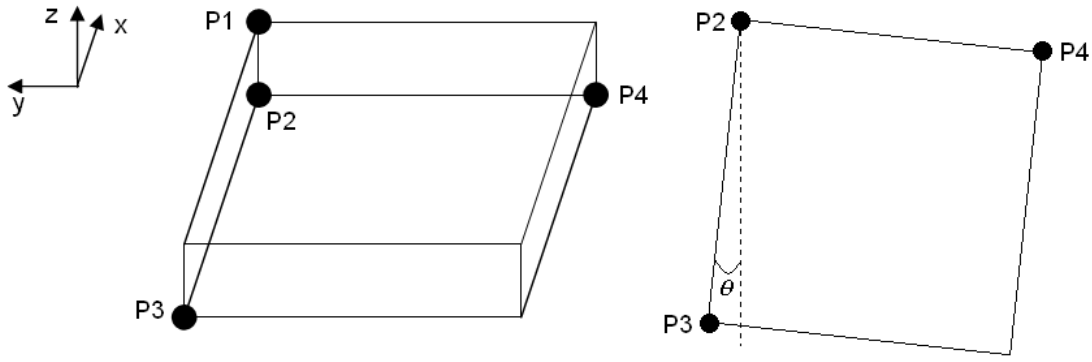


Figura 42: Localização dos pontos no espaço fornecidos pelo sistema laser/câmara para uma caixa com forma de paralelepípedo

Estes quatro pontos são suficientes para determinar as dimensões da caixa pois, como se sabe as coordenadas dos pontos, consegue-se calcular as dimensões da caixa apenas calculando a distância entre pontos num sistema tridimensional. A fórmula seguinte apresenta o cálculo da distancia entre dois pontos a e b .

$$D_{ab} = \sqrt{(x_b - x_a)^2 + (y_b - y_a)^2 + (z_b - z_a)^2} \qquad (4)$$

Então através da figura 4.11 pode-se indicar que:

- Comprimento (C):

$$C = \sqrt{(P_{2z} - P_{4z})^2 + [(P]_{2y} - P_{4y})^2 + [(P]_{2X} - P_{4X})^2} \qquad (5)$$

- Largura (L):

$$L = \sqrt{(P_{2z} - P_{3z})^2 + [(P]_{2y} - P_{3y})^2 + [(P]_{2X} - P_{3X})^2} \qquad (6)$$

- Altura (A):

$$A = \sqrt{(P_{2z} - P_{1z})^2 + [(P]_{2y} - P_{1y})^2 + [(P]_{2X} - P_{1X})^2} \qquad (7)$$

Também é necessário o cálculo de alguns ângulos, sendo estes a orientação (θ) que representa o desvio que a caixa apresenta, a inclinação (φ_x) resultante da diferença de altura entre P3 e P2 e a inclinação (φ_y) resultante da diferença de altura P4 e P2.

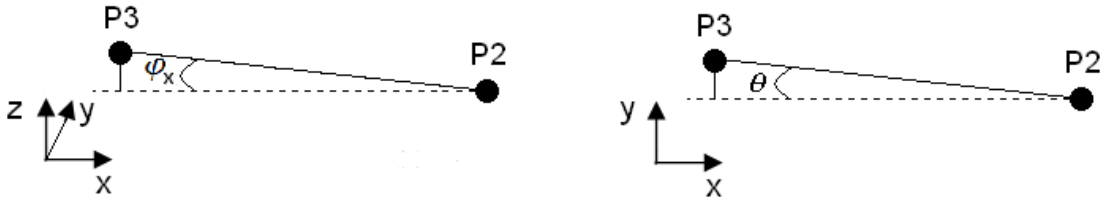


Figura 43: Exemplo elucidativo da relação entre os ângulos e os eixos

Através de algumas noções básicas de geometria e usando a figura 4.12 como referência pode-se afirmar que:

$$P_{3y} - P_{2y} = L * \cos\theta \qquad (8)$$

Depois de algumas manipulações, obtêm-se:

$$\theta = \text{sen}^{-1}\left(\frac{P_{3y} - P_{2y}}{L}\right) \qquad (9)$$

Usando a equação da orientação consegue-se obter a inclinação (φ_x) :

$$\varphi_x = \text{sen}^{-1}\left(\frac{P_{3z} - P_{2z}}{L}\right) \qquad (10)$$

E a inclinação (φ_y):

$$\varphi_y = \text{sen}^{-1}\left(\frac{P_{4z} - P_{2z}}{C}\right) \qquad (11)$$

Agora que já se conhece as dimensões e os ângulos apresentados pela caixa, pode-se calcular os pontos no espaço que o manipulador irá ter que percorrer. A posição dos pontos deve estar no centro do spray e por conseguinte a distância do primeiro ponto ao Ponto 2 tem o valor equivalente ao raio do spray.

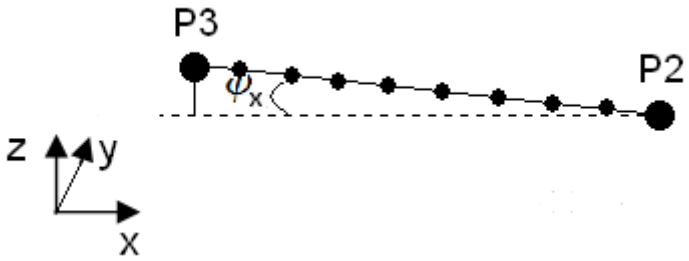


Figura 44: localização dos pontos entre p Ponto 2 e o Ponto 3

Para se calcular a coordenada Z dos pontos usa-se a equação:

$$Z = r * (1 + l) * \sin(\varphi_x) \qquad (12)$$

Onde r representa o raio do spray e i representa uma variável de incrementação. Para o primeiro ponto $i = 0$, pois como já se referiu anteriormente, este encontra-se a uma distância de P2 igual ao raio do spray. Como os pontos que o spray vai percorrer estão separados por uma distância igual ao diâmetro do spray, após o primeiro ponto, i será incrementado com um valor igual a dois para cada ponto a mais que é calculado. Assim teremos para o segundo ponto uma distância de P2 equivalente três raios do spray e para o terceiro ponto uma distância equivalente a 5 raios do spray, etc.

Para se calcular as coordenadas X e Y é necessário primeiro calcular a distancia (d) que o raio do spray (r) representa no plano xy.

$$d = r * (1 + i) * \cos(\varphi_x) \qquad (13)$$

$$X = d * (1 + i) * \cos(\theta) \qquad (14)$$

$$Y = d * (1 + i) * \sin(\theta) \qquad (15)$$

O ponto é então obtido por simples adição e subtracção dos valores com P2:

$$P = (P_{2x} - X, P_{2y} + Y, P_{2z} + Z) \qquad (16)$$

O processo usado para calcular os pontos de P2 para P4 é idêntico ao usado anteriormente.

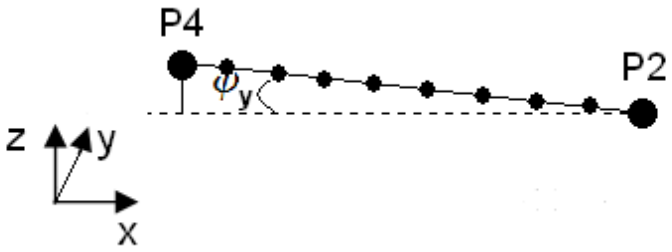


Figura 45: localização dos pontos entre p Ponto 2 e o Ponto 4

As Equações são as seguintes:

$$Z = r * (1 + i) * \sin(\varphi_y) \quad (17)$$

$$d = r * (1 + i) * \cos(\varphi_y) \quad (18)$$

$$X = d * (1 + i) * \sin(\theta) \quad (19)$$

$$Y = d * (1 + i) * \cos(\theta) \quad (20)$$

$$P = (P_{zx} + X, P_{zy} - Y, P_{zz} + Z) \quad (21)$$

A diferença na obtenção destes pontos com o processo anterior é pouca. Apenas se altera o ângulo de inclinação que passa a ser o φ_y, e ao invés de se subtrair o valor de X passa-se a somar e o contrário acontece com o valor de Y. Estas diferenças na soma e subtracção dos valores de X e Y advêm da disposição apresentada pelos eixos base do manipulador Motoman que podem ser observados na figura 4.11.

Para o cálculo dos pontos que se vão encontrar entre P2 e P1, o processo altera-se um pouco. Aqui usou-se o sistema de coordenadas esféricas para se poder calcular os pontos. Isto é necessário porque existe a possibilidade de a caixa se encontrar inclinada e portanto, com este sistema consegue-se determinar com exactidão a orientação do ponto P1 em relação a P2 e o ângulo que o ponto P1 faz com o plano onde se encontra P2.

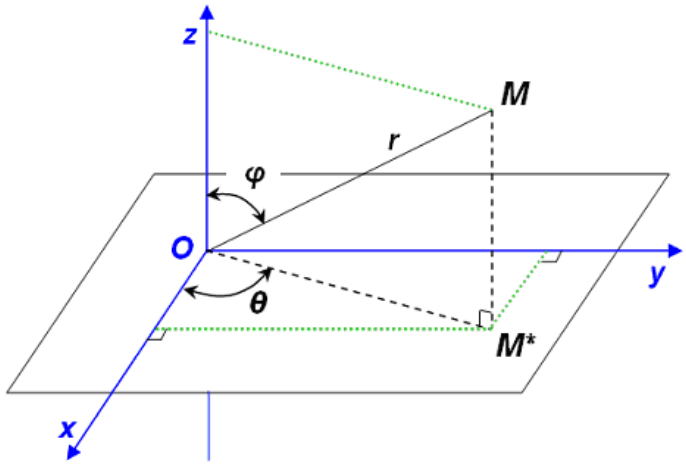


Figura 46: Sistema de coordenadas esférico

Sabendo que θ e φ são obtidos através das formulas:

$$\theta = \tan^{-1}\left(\frac{y}{x}\right) \qquad (22)$$

$$\varphi = \tan^{-1}\left(\frac{\sqrt{x^2+y^2}}{z}\right) \qquad (23)$$

Pode-se então obter as seguintes equações:

$$\theta_z = \tan^{-1}\left(\frac{y_1 - y_2}{x_1 - x_2}\right) \qquad (24)$$

$$\varphi_z = \tan^{-1}\left(\frac{\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}}{(z_1 - z_2)}\right) \qquad (25)$$

Assim consegue-se obter as seguintes equações:

$$Z = r * (1 + t) * \cos(\varphi_z) \qquad (26)$$

$$d = r * (1 + i) * \sin(\varphi_z) \qquad (27)$$

$$X = d * (1 + i) * \cos(\theta_z) \qquad (28)$$

$$Y = d * (1 + i) * \sin(\theta_z) \qquad (29)$$

$$P = (P_{zx} + X, P_{zy} + Y, P_{zz} + Z) \qquad (30)$$

Como se pode observar, todos os pontos criados até agora só dependeram de P2 e por conseguinte, apenas é necessário um ponto inicial para calcular os pontos restantes. Para já demonstrou-se como se calcula os pontos apresentados na figura seguinte.

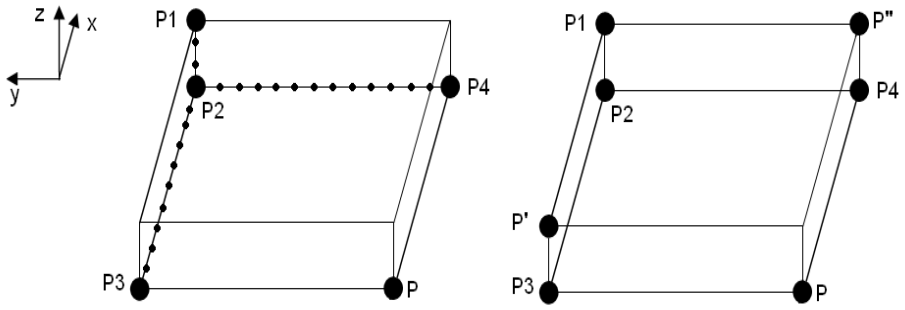


Figura 47: À esquerda apresenta-se os pontos calculados entre os quatro pontos fornecidos pelo sistema laser/câmara. À direita mostra-se os pontos dos vértices que são calculados.

Para se obter os pontos de P4 a P usa-se o mesmo processo usado na obtenção dos pontos entre P2 a P3 mas a única alteração é, utilizar o ponto P4 como referência. O mesmo se pode aplicar aos restantes pontos que estão por calcular, bastando para isso mudar o ponto de referência.

É claro que não se conhece todas as coordenadas dos vértices da caixa e portanto calcula-se os pontos P, P' e P'' da seguinte maneira:

$$P = (x_3 - C * \sin(\theta), y_3 - C * \cos(\theta), z_3 + C * \sin(\varphi_y)) \quad (31)$$

$$P' = (x_1 - L * \cos(\theta), y_1 + L * \sin(\theta), z_1 + L * \sin(\varphi_x)) \quad (32)$$

$$P'' = (x_1 - C * \sin(\theta), y_1 - C * \cos(\theta), z_1 + C * \sin(\varphi_y)) \quad (33)$$

Agora, sempre que se queira pintar uma face, a aplicação escolhe qual é a trajectória aplicável a esta e cria os pontos necessários. Esta escolha baseia-se na comparação entre o comprimento e a largura da caixa para que se possa optimizar a pintura. Relembro que as trajectórias possíveis para uma face estão expostas na figura 4.8.

Para se perceber melhor vai-se dar dois exemplos. No primeiro pretende-se pintar as faces 1, 2 e 3, e o comprimento da caixa é maior que a largura e que a altura, mas a altura é maior que a largura. No segundo pretende-se também pintar as faces 1, 2 e 3, e a largura da caixa é maior que o comprimento, que por sua vez é maior que a altura. Na figura seguinte pode-se observar os pontos que foram criados para cada exemplo.

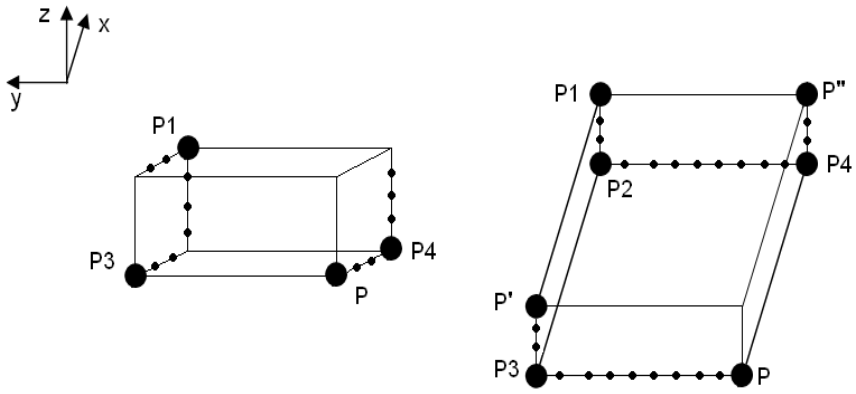


Figura 48: Exemplos de trajectórias de pintura caixas rectangulares com segundo algoritmo. À esquerda apresenta-se o primeiro exemplo e à direita apresenta-se o segundo.

Neste algoritmo, ao contrário do anterior, é possível determinar um ponto onde a pistola pode ser posicionada por cima da caixa. Esta posição é importante, pois é necessário posicionar a pistola de tinta por cima da caixa, pois caso contrário ao iniciar-se a pintura o manipulador pode colidir com as faces laterais com o movimento efectuado do seu ponto de repouso para o interior da caixa. As coordenadas X e Y do ponto são então obtidas da seguinte forma:

$$X = P_{1x} - L * \cos(\theta) - C * \sin(\theta) \qquad (34)$$

$$X = P_{1x} - L * \sin(\theta) - C * \cos(\theta) \qquad (35)$$

Para se obter a coordenada Z é necessário achar o valor do vértice mais alto. Para além da coordenada P_{1z} , que já se conhece, falta obter para as coordenadas para outros pontos dos vértices mais altos, sendo estas obtidas por:

$$P'_z = L * \sin(\varphi_x) \qquad (36)$$

$$P''_z = C * \sin(\varphi_y) \qquad (37)$$

$$P'''_z = C * \sin(\varphi_y) + L * \sin(\varphi_x) \qquad (38)$$

Ao valor da coordenada Z mais alta deve-se adicionar um valor extra para que se aumente o espaço de manobra.

Para se conseguir pintar o exterior da caixa também basta mudar o ponto de referência. Como todos os pontos que se localizam nos vértices da caixa podem ser obtidos facilmente, para se pintar a base que está virada para cima bastou usar os pontos P1, P' e P'' como pontos de referência, sendo obviamente a escolha destes efectuada consoante a largura é maior que o comprimento ou vice-versa. Para os pontos laterais, estes são obtidos da mesma maneira que os obtidos para pintar o interior da caixa.

4.4.4 Base Circular

Para se realizar a pintura de uma superfície circular apenas foram levadas em consideração duas hipóteses para trajectórias. Estas podem ser observadas na figura seguinte.

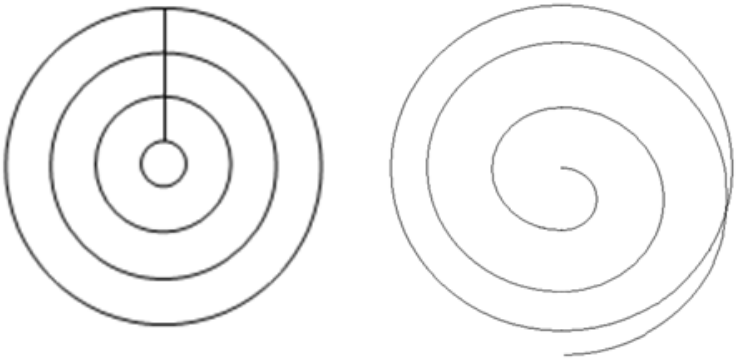


Figura 49: Hipóteses para pintar base circular.

A trajectória em forma de espiral é muito utilizada na indústria, mas é necessário fazer um pequeno ajuste á forma, para que esta possa ser utilizada para a pintura da base de um cilindro. Como se pode ver na figura em cima, a trajectória necessita de sair fora da superfície para poder pintar a parte que falta no final da pintura. Obviamente, se houver uma superfície perpendicular à superfície que está a ser pintada, esta ultima parte da trajectória não pode ser efectuada, pois a pistola acabará por colidir com a superfície perpendicular. Portanto, a alteração efectuada consistiu em finalizar a espiral em círculo.

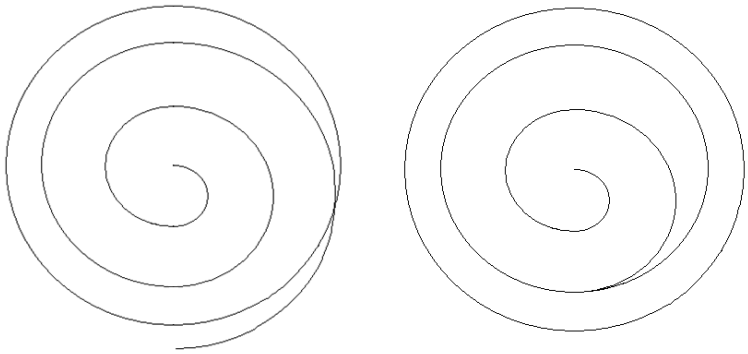


Figura 50: Alteração da trajectória em espiral

Como o comando MOVC não se encontra disponível para a solução em vista neste trabalho, que é controlar o manipulador remotamente, aquando da criação do primeiro algoritmo que usava incrementos, foi impossível usar a trajectória em espiral, pois o algoritmo tornar-se-ia muito complexo e portanto optou-se por usar a trajectória mostrada no lado esquerdo da figura 4.17. No segundo algoritmo como se passou a criar pontos no espaço, conseguiu-se implementar a trajectória em espiral modificada, o que apresenta uma melhoria significativa, pois os movimentos perpendiculares usados na outra trajectória eram um desperdício de movimento e tempo de execução da pintura.

4.4.5 Caixa Cilíndrica - Trajectórias com Incremento de Posição

No primeiro algoritmo levou-se em consideração que uma circunferência pode ser aproximada por um conjunto de rectas, mas surgiu um dilema. Qual seria o número de rectas que se aproximaria melhor a uma circunferência, mas que não fosse muito elevado. Foi então que se optou por usar uma trajectória com a forma de um dodecágono, pois é a forma que se aproxima mais de uma circunferência com o menor número de rectas. A trajectória para pintar a base de um cilindro passou a ter a forma apresentada na figura seguinte. O número de trajectórias dodecágonos aumenta

consoante o raio da base aumenta e a pistola de tinta passa de uma forma para a seguinte através de um movimento ascendente no eixo dos x.

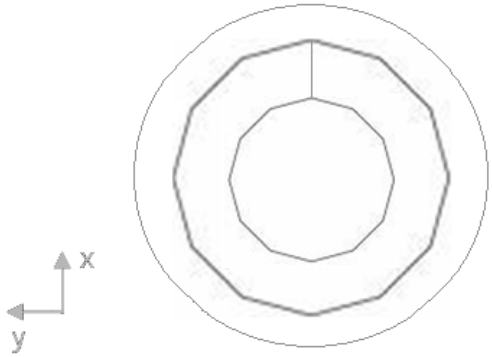


Figura 51: Trajectória com forma de um dodecágono

Para se pintar a face lateral, também é utilizada a forma do dodecágono, mas agora a pistola está inclinada de forma a poder pintar a parte lateral. Este ângulo de inclinação pode ser escolhido pelo utilizador de maneira a que a pistola possa se movimentar dentro da caixa sem colidir com a face lateral.

O processo de pintura lateral consiste, em primeiro lugar, no posicionamento da pistola num ponto da face lateral com uma inclinação escolhida. Este ponto encontra-se a uma altura da base igual ao raio do spray e por conseguinte, para este ser atingido é necessário executar um movimento da pistola no eixo dos z.

Figura 52: Posicionamento para pintura da face lateral de uma caixa cilíndrica

Agora pode-se iniciar a pintura lateral. Para isso a pistola irá executar um dodecágono, como os usados na pintura da base, mas em cada recta que a pistola realiza, esta gira sobre si um ângulo de 30º. Esta rotação de 30º foi escolhida levando em consideração a abertura entre os vários pontos do dodecágono. Esta rotação permite que a ponta da pistola esteja sempre apontada para a face lateral do cilindro. No final de um ciclo a pistola sobe no eixo dos zz o equivalente ao diâmetro do spray e inverte a rotação da pistola.

Para que esta trajectória em forma de dodecágono, que é usada nas pinturas circulares, fosse implementada na aplicação, foi necessário calcular o comprimento das rectas do hexágono, os ângulos que estas fazem com os vários eixos trigonométricos e quanto será o aumento destas à medida que o raio da suposta circunferência (dodecágono) aumenta.

Como é fácil de prever, o tamanho de todas as rectas será igual, portanto basta obter o comprimento para uma delas, que as outras têm o mesmo valor. Então em primeiro lugar vai-se mostrar os cálculos relativos à obtenção dos ângulos que estas rectas apresentam em relação aos eixos XX e YY, pois nestes cálculos acabamos por obter o comprimento das rectas, e depois vai se expor o cálculo efectuado com o intuito de obter o coeficiente que representa o aumento de comprimento destas. Começaremos então pela primeira recta que é mostrada na figura seguinte.

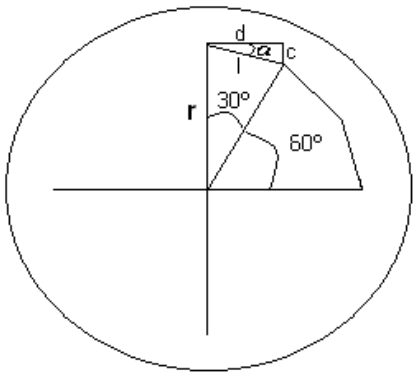


Figura 53: Representação dos ângulos da primeira recta do dodecágono

Usando a trigonometria podemos chegar facilmente às representações seguintes:

$$\begin{cases} d = r \times \cos 60 \\ c = r - r \times \sin 60 \end{cases} \qquad (39)$$

A figura mostra um triângulo rectângulo. Este tipo de triângulo cumpre o Teorema de Pitágoras, que afirma que a quadrado da hipotenusa é igual à soma do quadrado dos catetos. Os catetos são os lados d e c, sendo a hipotenusa o lado l. Portanto a fórmula do teorema fica representada da seguinte forma:

$$l^2 = d^2 + c^2 \qquad (40)$$

Substituindo as farias formulas, obtemos:

$$l^2 = (r \times \cos 60)^2 + (r - r \times \sin 60)^2 \qquad (41)$$

$$l^2 = r^2 \times \cos 60^2 + r^2 - r^2 \sin 60 + r^2 \sin 60^2 \qquad (42)$$

Após algumas manipulações obtem-se o resultado, que é:

$$l = r \times \sqrt{2 - 2 \sin 60} = r \times \sqrt{(2 - \sqrt{3})} \qquad (43)$$

Assim conseguimos obter o valor do comprimento das rectas, sendo este:

$$l = r \times \sqrt{(2 - \sqrt{3})} \qquad (44)$$

Agora podemos começar a calcular os ângulos. Sabendo que:

$$d = l \times \cos \alpha \qquad (45)$$

Pode-se alterar a fórmula de forma a se poder calcular α :

$$\cos\alpha = \frac{d}{l} = \frac{r \times \cos 60}{r \times \sqrt{2 - 2\sin 60}} \qquad (46)$$

Depois de algumas manipulações, obtem-se finalmente o valor de α :

$$\alpha = \cos^{-1}\left(\frac{\frac{1}{2}}{\sqrt{(2-\sqrt{3})}}\right) = 15 \qquad (47)$$

Temos então que o ângulo α é igual a 15º.

Sabendo que a soma dos ângulos internos de um triângulo é igual a 180º e que o triângulo é recto, o valor do ângulo β é:

$$\beta = 180 - 90 - 15 = 75 \qquad (48)$$

A segunda recta pode ser observada na figura seguinte

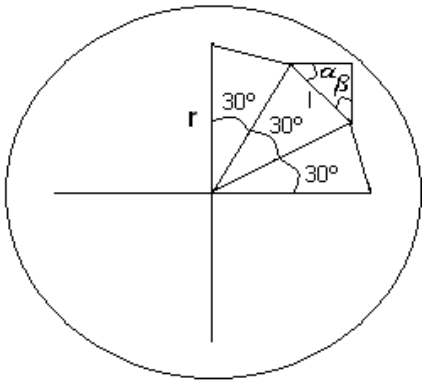


Figura 54: Representação dos ângulos da segunda recta do dodecágono

As equações que representam cada um dos lados do triângulo são:

$$\begin{cases} d = r \times \cos 30 - r \times \sin 60 = r\left(\frac{\sqrt{3}}{2} - \frac{1}{2}\right) = \frac{r}{2\left(\sqrt{3}-1\right)} \\ c = r \times \sin 60 - r \times \cos 30 = r\left(\frac{\sqrt{3}}{2} - \frac{1}{2}\right) = \frac{r}{2\left(\sqrt{3}-1\right)} \\ l^2 = d^2 + c^2 \end{cases} \quad (49)$$

Como o comprimento das rectas c e d são iguais, é obvio que este triângulo rectângulo tem os ângulos α e β iguais, tendo estes um valor igual a 45º.

Agora em relação à última recta, os ângulos são fáceis de obter pois se se reparar bem na figura seguinte, é possível notar que a recta é semelhante à que foi calculada em primeiro lugar, bastando para isso rodar a imagem.

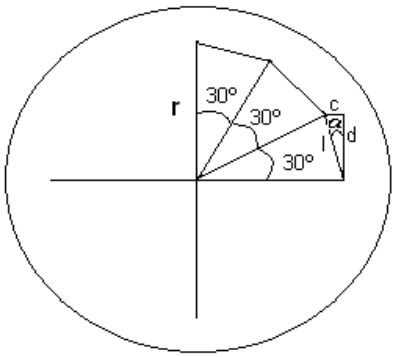


Figura 55: Representação dos ângulos da terceira recta do dodecágono

Por isso o valor de α é 75º e o de β é 15º.

Por último é mostrado como é conseguido o coeficiente que relaciona o aumento do raio do anel com o aumento necessário para que as rectas possam percorrer o perímetro do anel.

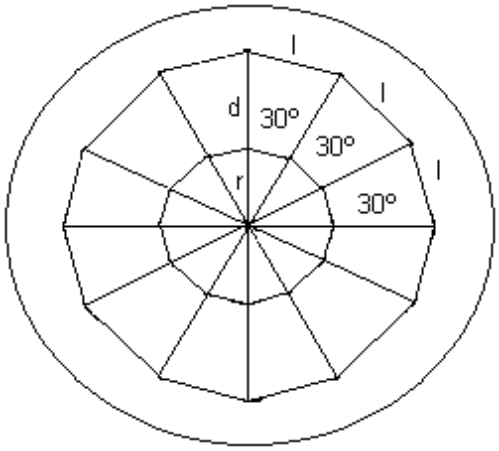


Figura 56: Representação da relação entre o tamanho do raio do dodecágono e as suas rectas

Como se pode observar na figura acima, inicialmente o centro do spray da pistola precisa de estar centrada a uma distância, igual ao raio do spray, do centro do círculo que irá ser pintado.

A partir do primeiro ciclo de pintura, para cada vez que se passa para um ciclo seguinte, o raio, a que se deve encontrar o centro do spray da pistola com o centro do círculo a pintar, deve ser sempre incrementado com um valor igual ao diâmetro do spray.

Portanto podemos mostrar a nova equação que representa todos comprimentos para as rectas em qualquer ciclo, onde x representa o ciclo em que se encontra a pintura:

$$l = r * (1 + i)\sqrt{(2 - \sqrt{3})} \qquad (50)$$

Onde i inicialmente tem o valor zero, mas com o aumentar do número de ciclos este é incrementado por cada um com um valor igual a 2.

4.5 Caixa Cilíndrica -Trajectórias com Pontos no Espaço

Neste algoritmo como já foi dito anteriormente, usou-se pontos no espaço. O sistema laser para este tipo de caixas fornece 3 pontos à aplicação. O primeiro ponto está localizado no centro da face superior da caixa, o segundo está localizado no centro da base da caixa e o terceiro ponto está localizado no perímetro da base no ponto onde a base está mais alta.

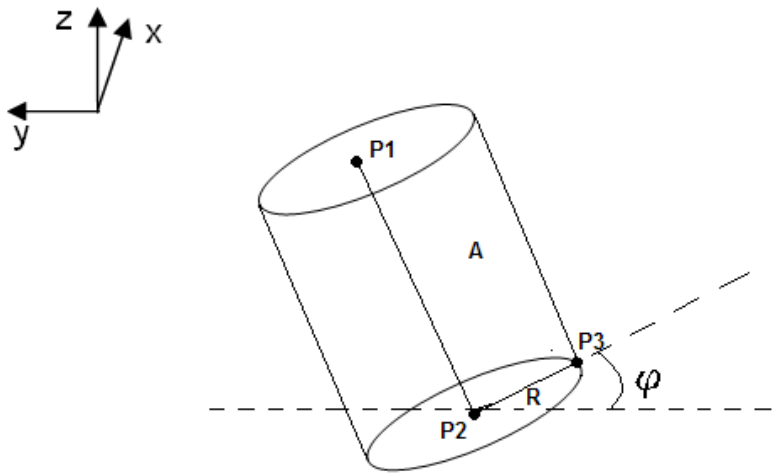


Figura 57: Representação dos pontos no espaço fornecidos pelo sistema laser/câmara para uma caixa cilíndrica

Para se obter as dimensões da caixa foi tomado em consideração a formula 4 e por conseguinte estas são obtidas usando as seguintes equações:

- Raio:

$$R = \sqrt{(P_{3z} - P_{2z})^2 + [(P]_{3y} - P_{2y})^2 + [(P]_{3X} - P_{2X})^2} \qquad (51)$$

- Altura:

$$A = \sqrt{(P_{1z} - P_{2z})^2 + [(P]_{1y} - P_{2y})^2 + [(P]_{1X} - P_{2X})^2} \qquad (52)$$

Os ângulos que precisam ser calculados são obtidos usando as formulas 22 e 23. Estes ângulos representam a orientação que P3 apresenta em relação a P2, a orientação que P1 apresenta em relação a P2 e a inclinação que P3 apresenta em relação a P2. As equações são:

$$\theta = \tan^{-1}\left(\frac{y_3 - y_2}{x_3 - x_2}\right) \qquad (53)$$

$$\theta_z = \tan^{-1}\left(\frac{y_1 - y_2}{x_1 - x_2}\right) \qquad (54)$$

$$\varphi = \sin^{-1}\left(\frac{P_{3z} - P_{2z}}{R}\right) \qquad (55)$$

Sabendo estes dados pode-se determinar quais são os pontos no espaço que serão necessários para que a caixa seja pintada. Aqui também os pontos da base e da face lateral são determinados separadamente o que possibilita que se possa pintar só a face lateral.

Neste algoritmo adoptou-se para a base da caixa uma trajectória em espiral o que melhora bastante a performance da pintura. Na figura seguinte pode-se observar como os pontos se dispõem na trajectória.

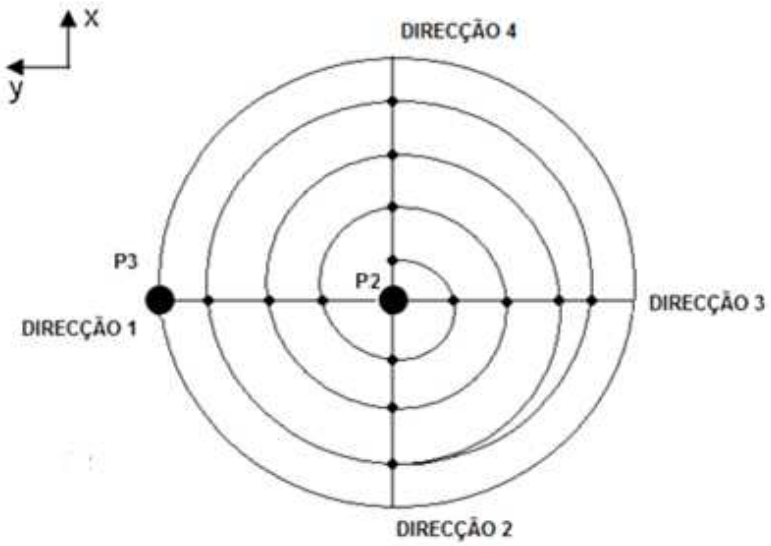


Figura 58: Representação dos pontos no espaço para a pintura de uma base circular no segundo algoritmo

Os pontos vão sendo determinados em cada uma das quatro direcções a partir do centro da base e por ordem de execução. Portanto os valores X, Y e Z vão sendo somados ou subtraídos ao ponto central consoante os pontos se vão afastando deste. Os valores de X, Y e Z são determinados da seguinte maneira:

$$X = r * (1 + i) * \cos(\theta + a) \qquad (56)$$

$$Y = r * (1 + i) * \sin(\theta + a) \qquad (57)$$

$$Z = r * (1 + i) * \sin(\varphi) \qquad (58)$$

Onde i inicialmente é zero e com a criação de cada 4 pontos é incrementado com um valor igual a 2, querendo isto dizer que esses pontos estão afastados do centro da base $(1 + i)$ vezes o raio sendo $i = 0,2,4,6, \ldots$. O a é um angulo que aumenta 90º

cada vez que se muda de direcção. Portanto inicialmente está em 0 e aumenta até aos 270 graus aquando da quarta direcção.

- Direcção 1:

$$\begin{cases} P = (P_{2x} + X, P_{2y} + Y, P_{2z} + Z) \\ a = 0 \end{cases} \quad (59)$$

- Direcção 2:

$$\begin{cases} P = (P_{2x} + X, P_{2y} + Y, P_{2z}) \\ a = 90 \end{cases} \quad (60)$$

- Direcção 3:

$$\begin{cases} P = (P_{2x} + X, P_{2y} + Y, P_{2z} - Z) \\ a = 180 \end{cases} \quad (61)$$

- Direcção 4:

$$\begin{cases} P = (P_{2x} + X, P_{2y} + Y, P_{2z}) \\ a = 270 \end{cases} \quad (62)$$

Como era de esperar os pontos nas direcções 2 e 4 estão à mesma altura que o ponto central da base da caixa, enquanto os pontos na direcção 1 vão subindo em altura pois estão na direcção do ponto mais alto que é P3 e como tal os pontos na direcção 3 vão diminuindo em altura.

Para que a espiral termine numa circunferência, adiciona-se no final um ponto igual ao último ponto obtido na direcção 1. Se isto não fosse feito a trajectória de pintura pararia no último ponto da direcção 4.

Agora para se determinar os pontos na face lateral usa-se um método parecido. Neste, calcula-se os pontos que ficam entre P2 e P1 e no entretanto usa-se esses pontos centrais para calcular os pontos na periferia da caixa estando estes localizados nas 4 direcções já explicadas em cima. Resulta então que os pontos centrais são determinados da seguinte maneira:

$$X = r * (1 + i) * \cos(\theta_z) \quad (63)$$

$$Y = r * (1 + i) * \sin(\theta_z) \qquad (64)$$

$$Z = r * (1 + i) * \cos(\varphi) \qquad (65)$$

$$P_n = (P_{zx} + X, P_{zy} + Y, P_{zz} + Z) \qquad (66)$$

Os pontos na periferia são determinados da mesma maneira que eram determinados os pontos que estavam dispostos nas 4 direcções da base. Mas neste caso o raio é igual a R, que relembro que é o raio da base. Resulta então que:

$$P = (P_n + R * \cos(\theta + a), P_n + R * \text{sen}(\theta + a), P_n + r * (1 + i) * \cos(\varphi)) \qquad (67)$$

Onde o a vai aumentando até se dar uma volta completa, ou seja 360 graus. Nesta trajectória é necessária a volta completa para que depois possa haver um movimento ascendente que consiga colocar a pistola de tinta na camada superior. Neste movimento ascendente a pistola não deve pintar para não haver desperdício de tinta.

Após ter ocorrido a subida, a pistola executa o trajecto em sentido oposto ao feito anteriormente. A figura 4.34 consegue mostrar como estão dispostos os pontos na face lateral. Pensou-se em usar uma trajectória como a forma de uma mola para realizar esta pintura lateral, só que é norma comum que após uma rotação a seguinte seja efectuada no sentido contrário e portanto se fosse usada a trajectória em forma de mola, essa mudança de trajectória não poderia ser feita.

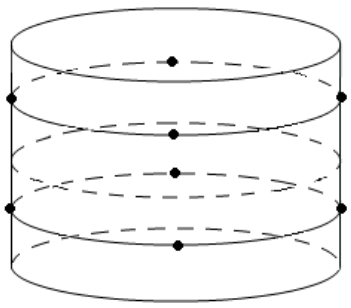


Figura 59: Representação dos pontos no espaço para a pintura de uma face lateral de uma caixa cilíndrica no segundo algoritmo

O ponto que se situa por cima da caixa para evitar colisões entre o manipulador e caixa no inicio de uma pintura é calculado da seguinte maneira:

$$P = (P_{1x}, P_{1y}, P_{1z} + R * \sin\varphi + b) \qquad (67)$$

O valor b é uma constante que tem o intuito de dar uma margem de valor ao ponto calculado.

Para se conseguir pintar o exterior da caixa cilíndrica basta mudar o ponto de referência da base deixando de ser P2 e passando a ser P1.Para que esta pintura ocorra com sucesso, é necessário que a caixa cilíndrica esteja com a base virada para cima, significando que P1 é o centro da base.

4.6 Limite

Com o intuito de garantir que não haveria sobreposição de trajectos, ou seja, para garantir que a mesma parte da superfície da caixa não seria pintada duas vezes, houve a necessidade uma ligeira alteração no algoritmo de pintura.

Decidiu-se assumir que é possível ajustar o diâmetro do spray entre um valor máximo e mínimo. Este mecanismo permite optimizar a trajectória, pois consegue-se com isso evitar que a mesma área seja pintada duas vezes e que ocorra colisões da pistola com as faces perpendiculares da que está a ser pintada. Como exemplo desta ultima situação, considerando que a base da caixa tem uma largura igual a 10.5 cm e o comprimento é 15 cm e se o diâmetro do spray for 2 cm, quer isto dizer que no último trajecto, estando os pontos de mudança de direcção posicionados ao longo da largura, o

ultimo ponto estará localizado em 11 cm, ou seja fora da caixa. É óbvio que existe colisão.

Então o processo é feito da seguinte maneira:

1. Compara-se a distâncias e descobre-se qual é a mais pequena. Por exemplo para o caso da base compara-se o comprimento com a largura.
2. A seguir é efectuado o calculo, que consiste em:

$$n = \frac{distancia}{diametro\ máximo} \qquad (68)$$

$$diametro\ escolhido = \frac{distancia}{valor\ de\ n\ truncado\ à\ unidade\ superior} \qquad (69)$$

Mesmo no caso em que o *diâmetro escolhido* seja menor que o valor do diâmetro mínimo, o valor do diâmetro a usar na trajectória continua a ser o *diâmetro escolhido*, mas o valor a ser usado como diâmetro do spray será o valor do diâmetro mínimo.

Capítulo 5

Job para Caixas Cilíndricas

Como o segundo algoritmo criado necessita do uso da instrução MOVC para a realização da pintura de caixas cilíndricas, houve a necessidade de abandonar a ideia de ser apenas o computador a realizar todas as tarefas e portanto teve que se criar um pequeno job onde se pudesse executar a instrução MOVC para que todas as trajectórias com forma cilíndrica possam ser realizadas.

Nesta fase do trabalho apenas as caixas cilíndricas poderão usar este job, mantendo assim a pintura de caixas rectangulares a serem completamente controladas pela aplicação. O algoritmo deste job consiste num simples case select onde se pode escolher executar algumas das instruções de movimento mencionadas no capítulo 2. Para que tal aconteça, este job após a sua activação, fica num ciclo à espera que o computador lhe informe que pode executar um movimento, fornecendo-lhe a posição para onde deve ser feito o movimento.

Antes de ser apresentado o raciocínio por trás do algoritmo do job irá ser apresentada a linguagem Inform desenvolvida pela Motoman, mas apenas referindo comandos, variáveis e certas características da linguagem.

5.1 INFORM

A linguagem de programação usada pelo controlador NX100 é o Inform II, apesar de esta linguagem já estar a ser comercializada na sua terceira versão, Inform III. Esta linguagem é de fácil compreensão, simples programação e é bastante eficaz a tirar proveito das capacidades do controlador. Mas como é uma linguagem máquina é bastante limitada ao nível da execução de cálculos e no manuseamento de informação, pois podem ser necessárias várias linhas de código para que se possa por exemplo fazer uma simples conta que envolva somas e multiplicações.

Esta linguagem, devido a ser uma linguagem máquina, não oferece nenhum tipo de ciclo For ou While o que torna a programação bastante maçadora. Para se conseguir criar algum destes tipos de ciclo é necessário alguma imaginação e um grupo de instruções como o Jump e o If.

O Inform II é composto por uma instrução e um item adicional, como uma tag ou uma informação numérica.

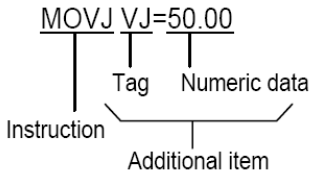


Figura 60: Exemplo de um comando

A instrução é usada para a execução de operações ou de processamento. O item adicional dependendo da instrução pode ser a velocidade, o tempo, etc. Na tabela seguinte pode-se ter uma noção de todos os tipos de instruções e em que situações são usadas.

Type	Content	Instruction Example
I/O Instruction	It is the instruction used to control the I/O.	DOUT, WAIT
Control Instruction	It is the instruction used to control the processing and operation.	JUMP, TIMER
Operating Instruction	It is the instruction by which the variables, etc. are used and operated.	ADD, SET
Move Instruction	It is an instruction concerning the movement and the speed.	MOVJ, REFP
Shift Instruction	It is an instruction used when a present teaching position is shifted.	SFTON, SFTOF
Instruction which adheres to instruction	It is an instruction which adheres to the instruction.	IF, UNTIL
Work Instruction	It is an instruction concerning work, such as arc welding and handling.	ARCON, WVON
Optional Instruction	It is an instruction concerning optional functions. It can only be used when the function is available.	-

Tabela 5.1: Tipos de instruções

Existem vários tipos de variáveis que podem ser utilizadas. As variáveis podem ser do tipo Inteiro, Real, Byte, Frases (Strings), Inteiros de dupla precisão (Doubles) e também podem ser uma estrutura que guarda a posição do manipulador no sistema de coordenadas que se pretender. Estas variáveis mencionadas podem ser do tipo variável local e do tipo variável do utilizador.

As variáveis locais são criadas durante a execução de um job e como pertencem a este job, mais nenhum job consegue alterá-las ou usá-las. Recordo que o controlador NX100 consegue controlar 4 manipuladores simultaneamente e daí haver a necessidade deste tipo de variáveis, para evitar que diferentes jobs usem as mesmas variáveis.

As variáveis do utilizador são variáveis já existentes no controlador e que podem ser alteradas e usadas por qualquer job que esteja a ser executado. Outra grande diferença é que o número de variáveis do utilizador tem um número fixo ao invés do número de variáveis locais que pode ser aumentado enquanto a memória do controlador permitir.

Na tabela seguinte pode-se observar as variáveis do utilizador, variáveis que são utilizadas pelo job que foi criado. A escolha recaiu sobre estas pois são as únicas que podem ser escritas ou lidas remotamente.

Data format		Variable No. (pcs)	Functions
Byte type		B000 to B099 (100)	Range of storable values is from 0 to 255. Can store I/O status. Can perform logical operations (AND, OR, etc.).
Integer type		I000 to I099 (100)	Range of storable values is from -32768 to 32767.
Double precision integer type		D000 to D099 (100)	Range of storable values is from -2147483648 to 2147483647.
Real type		R000 to R099 (100)	Range of storable values is from -3.4E+38 to 3.4E38. Accuracy: 1.18E-38 < x ≤ 3.4E38
Character type		S000 to S099 (100)	Maximum storable number of characters is 16.
Position type	Robot axes	P000 to P127 (128)	Can store position data in pulse form or in XYZ form. XYZ type variable can be used as target position data for move instructions, and as incremental values for parallel shift instructions.
	Base axes	BP000 to BP127 (128)	
	Station axes	EX000 to EX127 (128)	

Tabela 5.2: Variáveis do utilizador

A trajectória de um cilindro usa vários tipos de movimentos, todos mencionados no capitulo 2, menos o MOVC motivo pelo qual se criou este job e portanto vai-se apresentar o seu funcionamento.

A instrução MOVC, ao contrário de todas as outras instruções de movimento, necessita de três pontos para que possa ser executado, mais propriamente, necessita de três instruções para ser executada. Na figura seguinte, exemplifica-se o uso destas instruções para realizar uma semicircunferência. Para se obter a forma circular é necessário executar as seguintes instruções: MOVC P0, MOVC P1, MOVC P2.

Isto significa que a instrução necessita de três pontos para fazer a interpolação da trajectória que deve seguir e tem lógica, pois com dois pontos apenas se consegue obter uma recta.

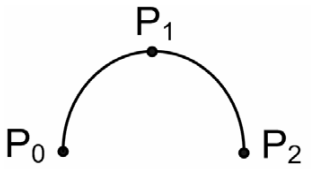


Figura 61: Movimentação com interpolação circular

5.2 JOB

Antes de se dar inicio à explicação do algoritmo do job é necessário indicar quais são as variáveis usadas. Estas são:

- BYTE:
  - B000 – Estado actual
    - 0 – Em espera
    - 1 – Executa MOVJ
    - 2 – Executa MOVL
    - 3 – Executa MOVC
    - 4 – Encerra job

- Position
  - P001 – Posição usada por todas as instruções de movimento
  - P002 – Posição usada apenas por MOVC
  - P003 - Posição usada apenas por MOVC

A seguir vão ser apresentadas as instruções que foram usadas na criação do job, usando para esse efeito alguns exemplos para que se consiga compreender melhor o que fazem.

- **SPEED VJ=**30.00 **V=**75 – Permite definir a velocidade de forma global. A velocidade da junção *VJ* é definida como percentagem da velocidade máxima e a velocidade linear *V* é definida em mm/s;
- **SET** I002 30 – Esta instrução atribui o valor do segundo parâmetro ao primeiro parâmetro (i.e. I002 = 30);

- **JUMP** ***Begin** **IF** B000=0 – Esta instrução salta "Jump" para a linha do programa com a label ***Begin** na eventualidade de se verificar a condição em "IF";
- **LOADV** P001 – Esta instrução pede o valor de uma variável ao computador remoto e guarda na variável indicada no comando.
- **SWAIT** – Instrução normalmente usada a seguir a um LOAD que obriga o job a esperar que o load seja concluído para poder continuar a ler instruções.

A título de exemplo pode-se observar no anexo 5 qual é a estrutura do código de um job. Existe uma data de parâmetros que necessitam ser escolhidos antes que se possa começar a introduzir linhas de código. Estas escolhas passam por ser a escolha da ferramenta a usar, o tipo de referencial a usar, o número de variáveis a usar e de que tipo, etc. As instruções podem ser introduzidas entre as palavras NOP e END.

O algoritmo do job usa a variável B000 como uma variável de controlo. A ideia é executar um ciclo usando a ideia de um case select. O job ao ser inicializado, inicia a variável B000 em zero e portanto é iniciado o ciclo de espera que dura enquanto o computador não mandar o job executar alguma instrução. O computador só manda executar uma instrução depois de o job tentar fazer o LOADV da variável B000, ou seja, lhe pedir para actualizar o valor da variável.

Quando o computador responde com o valor da variável que o controlador está a pedir, sendo nesta altura do tipo BYTE, o job consoante o valor prepara-se para executar a instrução de movimento ou finalizar o job no caso de o valor ser 4. Caso o valor corresponda a uma instrução de movimento, o job pede então as posições no espaço que a instrução de movimento necessita. O número de variáveis pedidas pode corresponder a 3 no caso do MOVC ou a 1 no caso das outras instruções. Após receber as variáveis de posição o job executa os movimentos e posteriormente coloca a variável de controlo a zero para iniciar o ciclo.

Conclusão

Conseguiu-se dar um grande avanço no trabalho pedido pela empresa Flupol. A parte de controlo do manipulador robótico foi implementada e os resultados finais foram acima do esperado. Não foi possível incorporar o sistema laser/câmara, devido à escassez de tempo.

A tarefa que tinha sido pedida e que consistia na criação de um algoritmo que conseguisse criar trajectórias para a pintura do interior de caixas com a forma paralelepipédica, foi concluída com sucesso e ainda foram adicionadas novas capacidades, tais como a possibilidade de se pintar caixas cilíndricas.

Os resultados apresentados pelo segundo algoritmo superaram em larga escala os resultados apresentados pelo primeiro. A primeira diferença consiste no facto de o primeiro algoritmo ser capaz de pintar caixas com diferentes orientações e diferentes inclinações, ao contrário do segundo. Isto significa que não é necessário nenhum tipo de mecanismo que alinhe a caixa para que esta seja pintada.

A segunda diferença consiste no facto de ser possível escolher que faces da caixa se pretendem pintar e ainda se poder escolher se se pretende pintar o interior da caixa ou o exterior, bastando para o segundo caso a caixa estar virada ao contrário.

A terceira diferença provem do facto de, como se calcula os pontos no espaço, um destes pontos serve como ponto de início da pintura, coisa que não é possível no primeiro algoritmo, onde é necessário indicar um ponto inicial que será usado em todos os tamanhos de um tipo de caixa.

A quarta vantagem passa pelo facto do código do primeiro algoritmo ser muito mais simples do que o do segundo, o que permitirá que a pessoa que continuar este trabalho, visto ainda faltar a implementação do sistema de visão artificial, possa aumentar o número de objectos que possam ser pintados usando a aplicação.

Esta simplificação do algoritmo teve como origem a modificação do tipo de informação escolhida para ser enviada do sistema laser/camâra para a aplicação. Isto prova que, com a ajuda de sensores que consigam fornecer bastante informação sobre a tarefa a realizar, é possível automatizar e aumentar o uso dos manipuladores robóticos para realizarem todo o tipo de tarefas.

Apesar de ter havido necessidade de programar o controlador robótico para que fosse possível realizar movimentos circulares, o controlo do manipulador continuou a ser todo feito no computador. Este job criado, como permite realizar todo o tipo de instruções de movimento, pode ser usado em trabalhos futuros para realizar os movimentos requeridos em tais trabalhos.

Como trabalho para o futuro deve-se concluir a adaptação do sistema laser/câmara para enviar pontos no espaço em substituto das dimensões das caixas. Também seria interessante criar uma pistola de tinta para que se possa testar o potencial da aplicação e continuar a criar trajectórias para novas formas de caixas.

Referências Bibliográficas

Ana S. Ferreira, A. Paulo Moreira, Paulo G. Costa: Low-Cost System for Object Positioning Through Laser–Camera Triangulation. *EPIA 2007, 13th Portuguese Conference on Artificial Intelligence, Guimarães, Portugal*, December 3-7, 2007.

António Azevedo, A.Paulo Moreira, Paulo G. Costa: Manipulador Industrial Aplicações Auto Ajustáveis. Projecto final de Curso, Julho de 2007.

Colombo, D., Dallefrate, D., Tosatti, L. Molinari: PC based control systems for compliance control and intuitive progamming of industrial robots. In: VDI-Wissensforum et al.: ISR 2006 -ROBOTIK 2006: *Proceedings of the Joint Conference on Robotics, Munich*, May 15-17, 2006.

Fusaomi Nagata, Keigo Watanabe and Kazuo Kiguchi: *Joystick Teaching System for Industrial Robots Using Fuzzy Compliance Control*. Open Access Database www.i-techonline.com

Haddadin, S., Albu-Schäffer, A., Hirzinger, G.: Dummy Crashtests for Evaluation of Rigid Human-Robot Impacts. *International Workshop on Technical Challenges for dependable robots in Human Environments*, 2007.

JN Pires, Godinho, Tiago, and Ferreira, Pedro, "CAD interface for Automatic Robot Welding Programming ", **Volume 31**, nº1, *Industrial Robot, An International Journal, MCB University Press*, 2004.

JN Pires, Godinho, Tiago, and Ferreira, Pedro, Loureiro, Altino, "Industrial Robotic System Programmed from CAD files – An Update", **Volume 32**, nº4, *Industrial Robot, An International Journal, MCB University Press*, (2005, a).

Pires, JN "Robot-by-Voice: Experiments on Commanding an industrial robot using the human voice", Industrial Robot, An International Journal, Emerald Group Publishing Limited, **Volume 32**, Number. 6, (2005, b).

JN Pires, "Welding robots: new trends and developments" *Industrial Robot – An International Journal* **Vol. 32** Number. 4, (2005, b).

JN Pires, "'Robotics for Small and Medium Enterprises: control and programming challenges", *Industrial Robot, an International Journal*, **Volume 33**, Number 6, Emerald Publishing, (2006, a).

JN Pires, Caramelo, F, Brito, P, Santos, J, Botelho, MF, "Robotics in Implant Industry: Stress/Strain Analysis. Part I: System Overview and Experiments", *Industrial Robot, an International Journal*, **Volume 33**, Number 3, Emerald Publishing, (2006, b).

JN Pires, "*Industrial Robots Programming: Building Applications for the Factories of the Future*", Springer, New-York (2007, a).

JN Pires, Godinho, Tiago, and Araújo, Ricardo, "Using Digital Pens to Program Welding Tasks",*Industrial Robot, An International Journal, MCB University Press* , **Volume: 34**, Issue: 6, Page: 476 – 486, (2007, b).

Manuais Disponibilizados pela Motoman, especialmente o NX100, System Setup, Data Transmission, Ethernet, Inform III, Basic Programning e Relative Job.

Sepp, W.: A Direct Method for Real-Time Tracking in 3-D Under Variable Illumination. *In Proceedings of the 27th DAGM Symposium Pattern Recognition 2005*, page 246-253, Deutsche Arbeitsgemeinschaft für Mustererkennung e.V., Wien (Österreich), August 31-September 2, 2005; Lecture Notes in Computer Science 3663, Springer Verlag 2005, ISBN-10 3-540-28703-5

Suppa, M., Kielhoefer, S., Langwald, J., Hacker, F., Strobl, K.H., Hirzinger, G.: The 3D-Modeller: A Multi-Purpose Vision Platform. In: *Proceedings of the IEEE - International Conference on Robotics and Automation* (ICRA 2007), Rome, Italy, April 2007.

ANEXO 1

Neste anexo estão expostas as instruções de movimento que podem ser executados remotamente (marcadas a cor azul) e também como são estruturados esses comandos.

Command Name		Read/Write Enabled					Only Read Enabled	
		Non-alarm/Non-error				Alarm/ Error	Non-alarm/ Non-error	Alarm/ Error
		Teach Mode		Play Mode				
		Stop	Operat-ing	Stop	Operat-ing			
Read or Monitor	RALARM	○	○	○	○	○	○	○
	RPOSC	○	○	○	○	○	○	○
	RPOSJ	○	○	○	○	○	○	○
	RSTATS	○	○	○	○	○	○	○
	RJSEQ	○	○	○	○	○	○	○
	JWAIT	○	○	○	○	A	○	A
	RGROUP	○	○	○	○	○	○	○
Read or Data Access	RJDIR	○	○	○	○	○	C	C
	RUFRAME	○	○	○	○	○	C	C
	UPLOAD	○	○	○	○	○	C	C
	SAVEV	○	○	○	○	○	C	C
Operation	HOLD	○	○	○	○	○	C	C
	RESET	○	○	○	○	○	C	C
	CANCE	○	○	○	○	○	C	C
	MODE	○	○	○	○	○/A*3	C	C
	CYCLE	○	○	○	○	○/A*3	C	C
	SVON 0 (OFF)	○	○	○	○	○	C	C
	SVON 1 (ON)	○	○	○	○	A	C	C
	HLOCK	○	○	○	○	○	C	C
	MDSP	○	○	○	○	○	C	C
	CGROUP	○	○	○	○	○	C	C
	CTASK	○	○	○	○	○	C	C
Activation	START	M	M	○/H*1	MOVE/○*2	A	C	C
	MOVJ	M	M	○/H*1	MOVE/○*2	A	C	C
	MOVL	M	M	○/H*1	MOVE/○*2	A	C	C
	IMOV	M	M	○/H*1	MOVE/○*2	A	C	C
	PMOVJ	M	M	○/H*1	MOVE/○*2	A	C	C
	PMOVL	M	M	○/H*1	MOVE/○*2	A	C	C
Editing	DELETE	○	MOVE	M	M	A	C	C
	CVTRJ	○	MOVE	M	M	A	C	C
	CVTSJ	○	MOVE	M	M	A	C	C
	WUFRAME	○	MOVE	M	M	A	C	C
	DOWNLOAD	○	○/MOVE*4	○	○/MOVE*4	A	C	C
	LOADV	○	○	○	○	A	C	C
Job selection	SETMJ	○	MOVE	○	MOVE	A	C	C
	JSEQ	○	MOVE	○	MOVE	A	C	C

MOVJ

Moves a manipulator to a specified coordinate position in joint motion.

Command format : MOVJ Data-1, Data-2, ····, Data-16

Data-1 = Motion speed (0.01 to 100.0%)
Data-2 = Coordinate specification
0 : Base coordinate
1 : Robot coordinate
2 : User coordinate 1
⋮
25 : User coordinate 24

Data-3 = X coordinate value (unit : mm, significant 3 decimal points)
Data- 4 = Y coordinate value (unit : mm, significant 3 decimal points)
Data-5 = Z coordinate value (unit : mm, significant 3 decimal points)
Data-6 = Wrist angle TX (unit : degree (°), significant 2 decimal points)
Data-7 = Wrist angle TY (unit : degree (°), significant 2 decimal points)
Data-8 = Wrist angle TZ (unit : degree (°), significant 2 decimal points)
Data-9 = Type
Data-10 = Tool No. (0 to 23)
Data-11 = Number of 7th axis pulses (for travel axis, mm)
Data-12 = Number of 8th axis pulses (for travel axis, mm)
Data-13 = Number of 9th axis pulses (for travel axis, mm)
Data-14 = Number of 10th axis pulses
Data-15 = Number of 11th axis pulses
Data-16 = Number of 12th axis pulses

- In a system without external axis, Data-11 to Data-16 should be set to "0".
- If a specified user coordinate is not defined, an error occurs.

Response format : 0000 or Error code

<Example>
Command MOVJ 50.0, 2, 123.1, 50.34, 10.8, 180.0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
Response 0000

IMOV

Moves a manipulator from the current position for a specified coordinate incremental value ir linear motion.

Command format : IMOV Data-1, Data-2, ····, Data-17

Data-1 = Motion speed selection (0 : V (speed), 1 : VR (posture speed))
Data-2 = Motion speed (0.1 to □□□.□□ mm/s, 0.1 to □□□.□° /s)
Data-3 = Coordinate specification
0 : Base coordinate
1 : Robot coordinate
2 : User coordinate 1
⋮
25 : User coordinate 24
26 : Tool coordinate

Data-4 = X coordinate incremental value (unit : mm, significant 3 decimal points)
Data-5 = Y coordinate incremental value (unit : mm, significant 3 decimal points)
Data-6 = Z coordinate incremental value (unit : mm, significant 3 decimal points)
Data-7 = Wrist angle TX incremental value (unit : degree (°), significant 2 decimal points)
Data-8 = Wrist angle TY incremental value (unit : degree (°), significant 2 decimal points)
Data-9 = Wrist angle TZ incremental value (unit : degree (°), significant 2 decimal points)
Data-10 = Reserved
Data-11 = Tool No. (0 to 23)
Data-12 = Number of 7th axis pulses (for travel axis, mm)
Data-13 = Number of 8th axis pulses (for travel axis, mm)
Data-14 = Number of 9th axis pulses (for travel axis, mm)
Data-15 = Number of 10th axis pulses
Data-16 = Number of 11th axis pulses
Data-17 = Number of 12th axis pulses

- In a system without external axis, Data-12 to Data-17 should be set to "0".
- If a specified user coordinate is not defined, an error occurs.

Response format : 0000 or Error code

<Example>
Command IMOV 0, 100.0, 2, 10.0, 10.0, 10.0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
Response 0000

MOVL

Moves a manipulator to a specified coordinate position in linear motion.

Command format : MOVL Data-1, Data-2, ····, Data-17

Data-1 = Motion speed selection (0 : V (speed), 1 : VR (posture speed))
Data-2 = Motion speed (0.1 to □□□.□□ mm/s, 0.1 to □□□.□° /s)
Data-3 = Coordinate specification
0 : Base coordinate
1 : Robot coordinate
2 : User coordinate 1
⋮
25 : User coordinate 24

Data-4 = X coordinate value (unit : mm, significant 3 decimal points)
Data-5 = Y coordinate value (unit : mm, significant 3 decimal points)
Data-6 = Z coordinate value (unit : mm, significant 3 decimal points)
Data-7 = Wrist angle TX (unit : degree (°), significant 2 decimal points)
Data-8 = Wrist angle TY (unit : degree (°), significant 2 decimal points)
Data-9 = Wrist angle TZ (unit : degree (°), significant 2 decimal points)
Data-10 = Type
Data-11 = Tool No. (0 to 23)
Data-12 = Number of 7th axis pulses (for travel axis, mm)
Data-13 = Number of 8th axis pulses (for travel axis, mm)
Data-14 = Number of 9th axis pulses (for travel axis, mm)
Data-15 = Number of 10th axis pulses
Data-16 = Number of 11th axis pulses
Data-17 = Number of 12th axis pulses

- In a system without external axis, Data-12 to Data-17 should be set to “0”.
- If a specified user coordinate is not defined, an error occurs.

Response format : 0000 or Error code

<Example>
Command MOVL 0, 500.0, 2, 123.1, 50.34, 10.8, 180.0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
Response 0000

ANEXO 2

Lista de cabeçalhos possíveis para o envio de comandos.

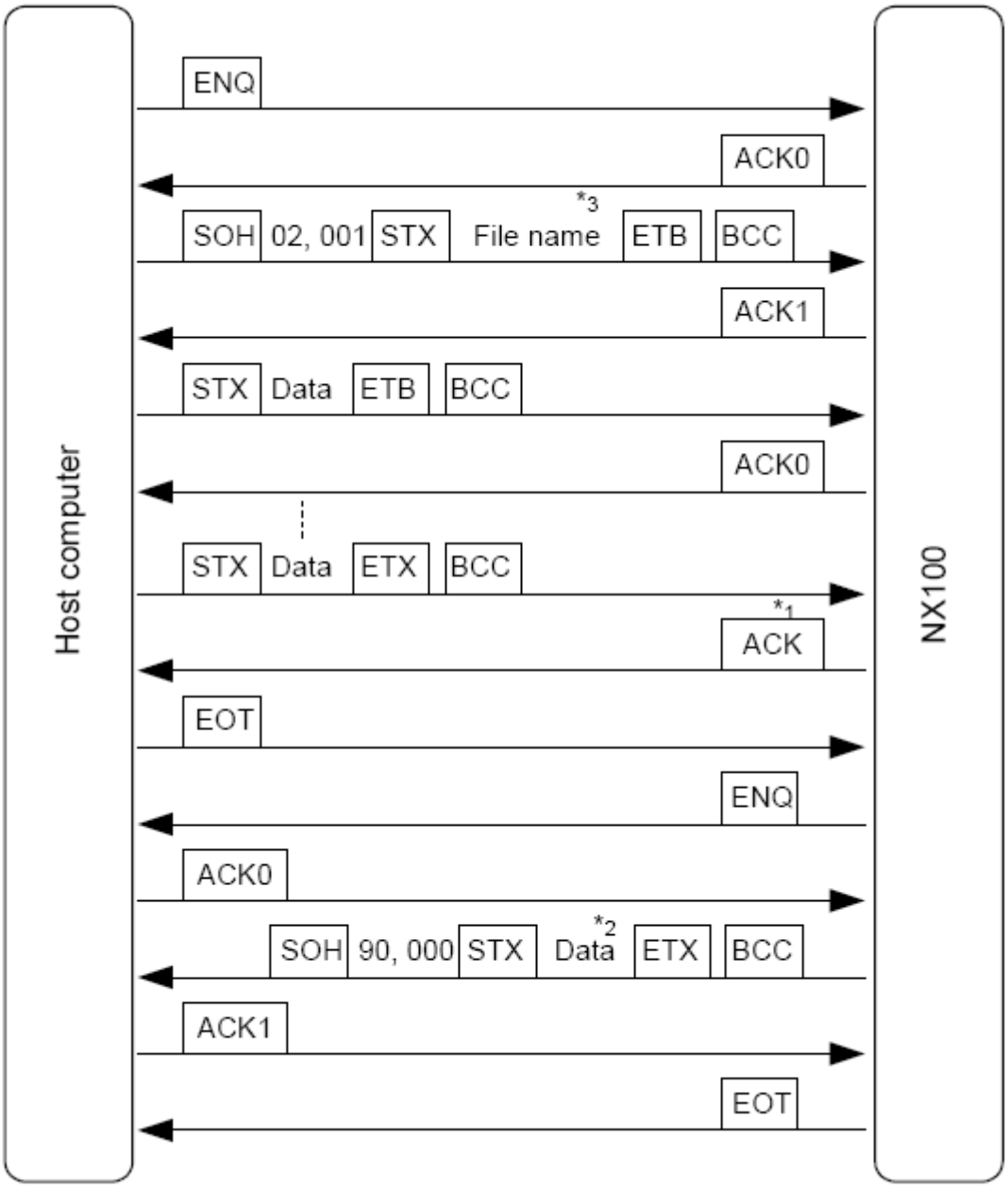
		Contents	File Name
01,	000	Command from a external computer	
02,	001	Single job data	xxxxxxxx. JBI
	002	Related job data	xxxxxxxx. JBR
02,	051	Request for single job data	xxxxxxxx. JBI
	052	Request for related job data	xxxxxxxx. JBR
02,	200	Tool data	TOOL. CND
	201	Weaving condition data	WEAV. CND
	202	User coordinate data	UFRAME. CND
	203	Welding start condition data	ARCSRT. CND
	204	Welding end condition data	ARCEND. CND
	218	Motor gun pressure data	SPRESS.CND
	219	Pressure data	SPRESSCL.CND
	220	Open/full open position data	STROKE.DAT
	221	Spot I/O allocation data	SPOTIO.DAT
	222	Air gun condition data	AIRGUN.DAT
	230	Air auxiliary condition	ARCSUP.DAT
	232	Variable data	VAR. DAT
	265	Spot welder condition data	SWELDER.DAT
	413	Clearance setting data	CLEARNCE.DAT
	240	System information	SYSTEM. SYS
	241	Alarm history data	ALMHIST. DAT
02,	300	Request for tool data	TOOL.CND
	301	Request for weaving condition data	WEAV.CND

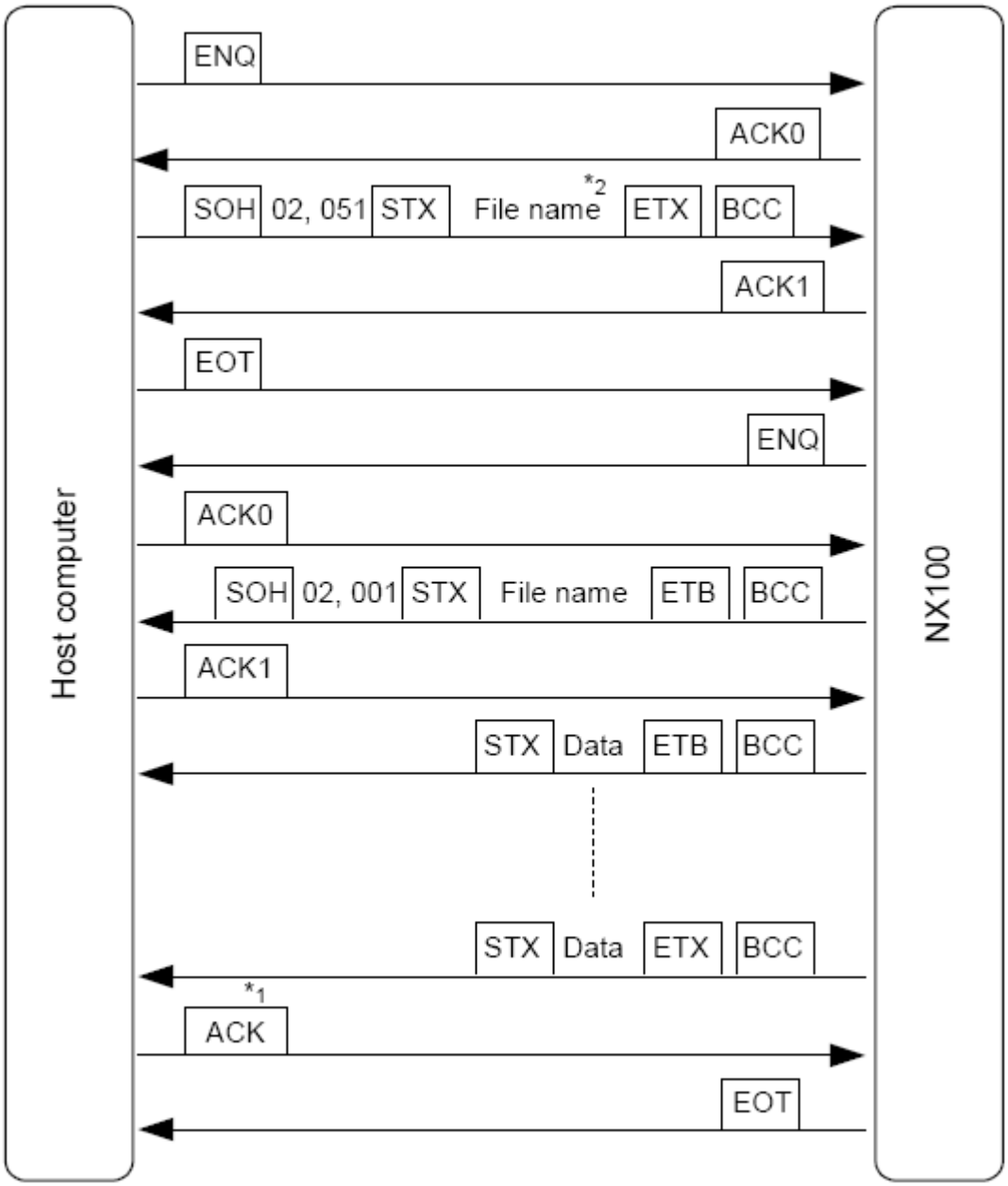
		Contents	File Name
	302	Request for user coordinate data	UFRAME.CND
	303	Request for welding start condition data	ARCSRT.CND
	304	Request for welding end condition data	ARCEND.CND
	318	Request for motor gun pressure data	SPRESS.CND
	319	Request for pressure data	SPRESSCL.CND
	320	Request for open/full open position data	STROKE.DAT
	321	Request for spot I/O allocation data	SPOTIO.DAT
	322	Request for air gun condition data	AIRGUN.DAT
	330	Request for weld auxiliary condition	ARCSUP.DAT
	332	Request for variable data	VAR.DAT
	344	Request for shock detection level	SHOCKLVL.CND
	350	Request for power source condition data	WELDER.DAT
	351	Request for power source user definition data	WELDUDEF.DAT
	364	Request for spot gun condition data	SGUN.DAT
	365	Request for spot welder condition data	SWELDER.DAT
	513	Request for clearance setting data	CLEARNCE.DAT
	340	Request for system information	SYSTEM.SYS
	341	Request for alarm history data	ALMHIST.DAT
03,	001	Byte type variable	
	002	Integer type variable	
	003	Double precision type variable	
	004	Real number type variable	
	005	Robot axis position type variable (pulse type)	

		Contents	File Name
	006	Robot axis position type variable (XYZ type)	
	007	External axis position type variable (pulse type)	
	008	External axis position type variable (XYZ type)	
03,	051	Request for byte type variable	
	052	Request for integer type variable	
	053	Request for double precision type variable	
	054	Request for real number type variable	
	055	Request for robot axis position type variable (pulse type)	
	056	Request for robot axis position type variable (XYZ type)	
	057	Request for external axis position type variable (pulse type)	
	058	Request for external axis position type variable (XYZ type)	
04,	001	Request for write-in of I/O signals	
	051	Request for read-out of I/O signals	
90,	000	Command or data response (normal/error)	
	001	Command or data response (data)	

ANEXO 3

Neste anexo é apresentado o protocolo que permite a troca de ficheiros como controlador robótico.




Host computer
NX100
ENQ
ACK0
SOH
02, 051
STX
File name
*2
ETX
BCC
ACK1
EOT
ENQ
ACK0
SOH
02, 001
STX
File name
ETB
BCC
ACK1
STX
Data
ETB
BCC
STX
Data
ETX
BCC
*1
ACK
EOT

ANEXO 4

Lista de erros que podem ser reportados pelo controlador robótico aquando de o envio de algo para o controlador.

Code	Contents
3070	Current value not made
3220	Panel lock ; mode/cycle prohibit signal is ON.
3230	Panel lock ; start prohibit signal is ON.
3350	User coordinate not taught
3360	User file destroyed
3370	Incorrect control group
3380	Incorrect base axis data
3390	Relative job conversion prohibit (at CVTRJ)
3400	Master call prohibit (parameter)
3410	Master call prohibit (lamp On during operation)
3420	Master call prohibit (teach lock)
3430	Robot calibration data not defined
3450	Servo power cannot be turned ON.
3460	Coordinate system cannot be set.
4010	Insufficient memory capacity (job registered memory)
3420	Master call prohibit (teach lock)
3430	Robot calibration data not defined
3450	Servo power cannot be turned ON.
3460	Coordinate system cannot be set.
4010	Insufficient memory capacity (job registered memory)
4012	Insufficient memory capacity (position data registered memory)
4020	Job editing prohibit
4030	Same job name exists

Code	Contents
4120	Position data destroyed
4130	Position data not exist
4140	Incorrect position variable type
4150	END instruction for job which is not master job
4170	Instruction data destroyed
4190	Invalid character in job name
4200	Invalid character in label name
4230	Invalid instruction in this system
4420	No step in job to be converted
4430	Already converted
4480	Teach user coordinate.
4490	Relative job/Independent control function not permitted
5110	Syntax error (syntax of instruction)
5120	Position data error
5130	No NOP or END instruction
5170	Format error (incorrect format)
5180	Incorrect number of data
5200	Data range over
5310	Syntax error (except instruction)
5340	Error in pseudo instruction specification
5370	Error in condition data record
5390	Error in job data record
5430	System not matched
5480	Incorrect welding function type