Concepção de um Sistema Robótico para Prototipagem Rápida por Maquinagem

Ricardo Sérgio Martins Pires Afonso

Dissertação

Orientadores:

Professor Doutor Paulo Augusto Ferreira de Abreu

Professor Doutor António Mendes Lopes

Faculdade de Engenharia da Universidade do Porto

Mestrado Integrado em Engenharia Mecânica

Opção de Automação

Julho de 2010

Aos meus pais

Resumo

Hoje em dia cada vez mais o mercado é impulsionado por um aumento da competitividade global na indústria da manufactura. Sendo influenciado pela economia, existe cada vez mais a necessidade do deslocamento das indústrias de manufactura para mercados com baixos custos de mão-de-obra.

Numa tentativa de evitar esta tendência, os sistemas de prototipagem rápida vieram a oferecer uma solução com aumento de produtividade e redução de custos, com melhoria de qualidade devido a uma melhor avaliação da fase de planeamento do produto.

No entanto os sistemas convencionais de prototipagem rápida também possuem as suas limitações quer na escolha de materiais, quer no volume do protótipo produzido. Nesse sentido as máquinas-ferramenta CNC desempenham um papel fundamental ultrapassando estas limitações.

Nos últimos anos o desenvolvimento de equipamento e *software* de programação, permitiu o surgimento da tecnologia robótica na área da prototipagem rápida por maquinagem.

Este trabalho propõe o desenvolvimento de uma solução integrada para a criação de protótipos por maquinagem usando um robô industrial equipado com uma ferramenta de corte.

Em primeira instância foi efectuada uma pesquisa dos diferentes tipos de soluções existentes na integração dos recursos dos sistemas CAD/CAM com os *softwares* de programação *off-line* dos vários fornecedores.

Seguidamente através da utilização de recursos de CAD/CAM gerou-se o código de programação para máquinas-ferramenta CNC e fez-se a posterior adaptação, para a aplicação num sistema robótico, através de uma interface computacional desenvolvida na linguagem de programação VBA (Visual Basic for Applications).

Por fim a viabilidade da solução proposta é confirmada através de ensaios realizados em modelos de superfícies complexas, onde o objectivo do referido trabalho foi alcançado.

Conception of a Robotic System for Rapid Prototyping by Machining

Abstract

Nowadays the market is driven by increased global competition in the manufacturing industry sector. Being influenced by the economy, the manufacturing industry feels tempted to move away to markets with significantly lower labor costs.

In an attempt to prevent this tendency, the rapid prototyping systems created a solution with an increase in productivity and cost reduction combined with better quality of the final product due to the process planning that provides a better understanding of the product.

However the conventional rapid prototyping systems also have their limitations on the materials available and the volume of the prototype produced. In this matter CNC machines play a key role surpassing these limitations.

In the last years with the development of equipment and programming software, allowed the emergence of robot technology in the field of rapid prototyping by machining.

This project proposes the development of an integrated solution for the creation of prototypes by machining using an industrial robot equipped with a cutting tool.

In the first instance was made a research of different types of existing solutions in the integration of the resources from CAD/CAM systems with off-line programming software from multiple vendors.

Then through the use of CAD/CAM resources the code for programming CNC machine tools is generated and the later adjustment to the application in a robotic system is made through a computer interface developed in VBA (Visual Basic for Applications) programming language.

Finally the possibility of the proposed solution is confirmed through tests applied in complex surface models, where the objective of this project was achieved.

Agradecimentos

Primeiramente gostaria de agradecer aos meus orientadores, Professor Doutor Paulo Abreu e Professor Doutor António Mendes Lopes pela disponibilidade, confiança e o apoio sempre presente durante todas as fases da dissertação.

Agradeço ao coordenador da opção de Automação, o Professor Doutor Francisco Freitas, pelo acompanhamento efectuado durante todo o semestre.

Quero ainda agradecer à Sra. Engenheira Célia do INEGI pela sua disponibilidade e amabilidade pela cedência de materiais para a realização dos ensaios finais.

A todos os meus colegas finalistas na opção de automação, que realizaram também a dissertação neste semestre, agradeço pela amizade e bons momentos.

Aos amigos e família, pelos momentos agradáveis de descontração, pelo carinho e atenção e pelo interesse no andamento do mestrado, que foram importantes para recuperar o ânimo e continuar as actividades.

Por último, gostaria de agradecer aos meus pais, pelos conselhos, pela preocupação com o meu presente e futuro e por todos os valores que acrescentaram em mim.

Índice de Conteúdos

Índice de Figuras

1 Introdução

A indústria de hoje está a tornar-se cada vez mais competitiva, e para sobreviver muitas empresas têm que fazer os ciclos de desenvolvimento dos seus produtos o mais curtos possível. Em anos recentes, a introdução das tecnologias da prototipagem rápida como a estereolitografia (SLA), fabricação de objectos por camadas (LOM) e sinterização selectiva por lazer (SLS), reduziu drasticamente o tempo de desenvolvimento do produto e os custos associados. Esta redução permite a fácil visualização da peça (protótipo) nas fases iniciais de design, o que contribui para que possam ser detectados e corrigidos erros de design da peça antes de se passar à fase de produção. Existe assim a possibilidade de fabricar peças complexas a partir de um desenho em computador, acelerando o desenvolvimento de novos produtos e permitindo que peças únicas ou em pequenos lotes sejam fabricadas rapidamente e com um custo muito baixo.

No entanto, a indústria vê os sistemas de prototipagem rápida como sendo não totalmente satisfatórios devido à limitação da escolha de materiais, do volume do protótipo produzido (Schaaf, 2000) e à baixa precisão no acabamento superficial. Consequentemente as máquinas ferramenta de controlo numérico computadorizado (CNC) mantêm-se como a melhor alternativa face às tecnologias de prototipagem rápida, revelando-se mais eficientes e económicas na produção de protótipos de maiores dimensões ou protótipos de superfícies complexas como peças da indústria aeroespacial, carroçaria automóvel, cascos de navios e outros bens de consumo (Vergeest & Tangelder, 1996).

Contudo nos últimos anos, os avanços da tecnologia robótica em equipamento e respectivo *software* de programação, permitiram o surgimento dos robôs industriais, em aplicações de remoção de material usando ferramentas de corte. Isto constitui uma nova fase

de inovação da tecnologia robótica aplicada a uma área que era previamente exclusiva das máquinas ferramenta.

1.1 Aplicações de Maquinagem usando Robôs Industriais

As aplicações de robôs[1] em maquinagem envolvem, normalmente, a utilização de uma ferramenta rotativa (de accionamento pneumático ou eléctrico) que o robô manipula. O objectivo é o robô posicionar a ferramenta de modo a ser possível executar a operação com a referida ferramenta. Existem ainda alguns casos em que o robô transporta a peça e a ferramenta está fixa (Abreu, 2001).

A utilização de robôs industriais tem vindo a ser alargada a aplicações de maquinagem em várias áreas tais como as que estão referidas na Tabela 1.

Indústria	Processos	Produto
Aeroespacial	Lixagem, polimento, furação	Pás de turbinas, fuselagem
Automóvel	Lixagem, furação, corte, fresagem	Blocos de motor, carroçaria, painéis
Fundição	Rebarbagem, fresagem, furação, acabamento	Moldes e peças fundidas
Médica	Polimento, lixagem	Próteses
Entretenimento	Fresagem	Cenários e figuras de parques de diversão, esculturas
Madeireira	Fresagem	Mobiliário, corrimões, moldes de banheiras
Plásticos	Fresagem	Moldes, capacetes

Tabela 1 – Exemplos de aplicações de maquinagem usando robôs industriais

[1] É necessário ter em consideração que ao longo deste relatório a palavra robô sempre que surgir será como referência à expressão robô industrial

De seguida são ilustrados exemplos concretos que evidenciam as capacidades dos robôs industriais no fabrico de protótipos, sendo feita uma breve referência às respectivas empresas que implementaram com sucesso soluções robóticas de maquinagem: a *Garner Holt Productions, S.N.B.R., Fabricamachinale* e a *Simtech Systems*.

Garner Holt Productions

Empresa americana que está focada na indústria do entretenimento, produzindo cenários e figuras animadas para parques de diversão, museus e casinos. No seu processo de produção, utiliza digitalizadores tridimensionais a laser. O ficheiro gerado pode ser manipulado de modo a aumentar ou diminuir o tamanho da imagem do objecto digitalizado. Assim com uma maqueta de tamanho reduzido pode-se criar um objecto de grandes dimensões. Para finalizar o ficheiro é utilizado para programar o robô industrial da *Kuka*, que efectua a maquinagem do objecto pretendido (Figura 1).

Figura 1 – Célula robótica da *Garner Holt Productions* constituída por um robô *Kuka* e uma mesa rotativa (http://www.sme.org)

S.N.B.R.

A empresa francesa *S.N.B.R* desenvolve a sua actividade no restauro de arte contemporânea e preservação do património arquitectónico. Tem vindo a desenvolver uma nova tecnologia robótica, permitindo o processamento de grandes objectos de arte nos seus materiais originais. Esta companhia participou num projecto conjuntamente com uma equipa científica, para recriar a esfinge de Naxos, em mármore, partindo de uma réplica em gesso da

esfinge. O primeiro passo consistiu num mapeamento digital do modelo da esfinge usando uma câmara digital 3D e um computador. De seguida procedeu-se a uma tarefa árdua de refinamento e organização de centenas de digitalizações tridimensionais numa sequência de imagens que foram introduzidas num *software* para assim se gerarem as trajectórias de maquinagem para o robô (Figura 2).

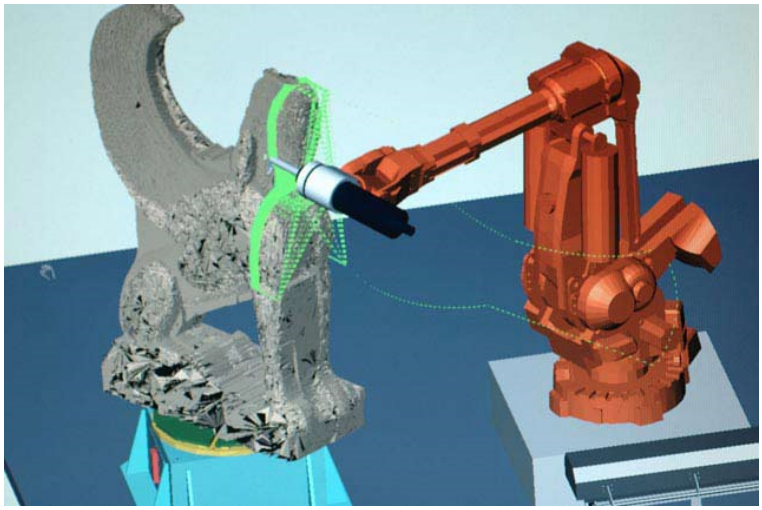


Figura 2 – Programação e simulação das trajectórias realizadas pelo robô da *S.N.B.R.* (http://www.snbr-stone.com/Le-Sphinx-des-Naxiens.html)

O resultado foi um sucesso, o robô da *S.N.B.R.* revelou-se à altura do desafio conseguindo esculpir um bloco de mármore de 6 toneladas, transformando-o numa peça de arte. A réplica da esfinge de Naxos final possui duas toneladas (Figura 3).

Figura 3– Robô escultor da *S.N.B.R.* executando o acabamento da réplica da esfinge de Naxos em mármore (http://www.snbr-stone.com/Le-Sphinx-des-Naxiens.html)

Fabricamachinale

A *Fabricamachinale* é uma empresa italiana do grupo *grupposcienziamachinale* que implementa soluções robóticas em diversas áreas, uma das quais é a área médica, produzindo modelos para a criação de órteses e próteses.

A solução robótica denominada *ORTIS* que a empresa comercializa consiste numa célula robotizada constituída por um robô industrial *Kuka* ou *ABB*, uma mesa rotativa (Figura 4) e um *software* próprio para programar o robô.

Figura 4 – Célula robotizada *ORTIS* da empresa *Fabricamachinale* (http://www.fabricamachinale.it/prodotti/ortis)

Simtech Systems

Esta empresa finlandesa desenvolve e distribui *software* para controlar robôs industriais. Trabalhando conjuntamente com a companhia alemã que fornece soluções robóticas integradas, a *Mühlbauer Maschinenbau*, e o fabricante automóvel *Audi*, a *Simtech systems* desenvolveu um método para a produção de moldes em areia. Os moldes são criados directamente a partir de um modelo em CAD (desenho assistido por computador) usando um robô para maquinar directamente o molde em areia sem ser necessário recorrer ao uso de modelos para criar o molde. A empresa designa este método por *patternless casting* (Sirviö, 2008)

Através do uso de dois *softwares* essenciais: um para a simulação de todo o processo de vazamento do metal no molde (*ConiferCast*) e o outro necessário para optimizar as trajectórias de maquinagem efectuadas pelo robô industrial no molde em areia (*ConiferRob*), tal processo tornou-se possível. A tecnologia da *Simtech* foi utilizada na produção de um protótipo do *Audi A5* (Figura 5).

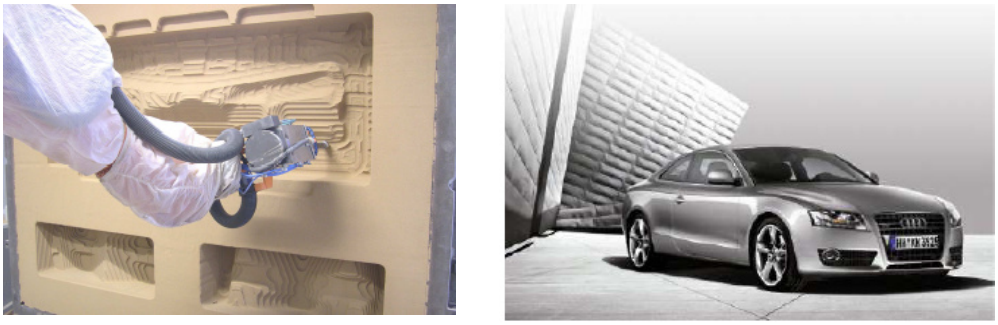


Figura 5 – Maquinagem robótica de moldes em areia para produção de protótipos na célula da *Audi* em Ingolstadt Alemanha (à esquerda) e visualização do *Audi A5* (à direita) (Sirviö & Wos, 2009)

1.2 Prototipagem Rápida por Maquinagem: Máquinas Ferramenta CNC e Robôs Industriais

Ao contrário do que acontece nos métodos tradicionais de prototipagem rápida em que o protótipo é feito por adição de sucessivas camadas de material até se obter a forma do protótipo, na prototipagem rápida por maquinagem o protótipo é produzido por remoção de material usando uma ferramenta.

É de facto na prototipagem rápida que os robôs têm vindo a mostrar grande potencial. Estes podem ser vistos de certo modo como máquinas CNC de 3 a 5 eixos na prototipagem de peças com superfícies complexas. Apesar da sua semelhança com os métodos de maquinagem CNC, a tecnologia de maquinagem com robôs tem as suas vantagens, incluindo um menor investimento no equipamento, maior flexibilidade associada aos seus 6 eixos (6 graus de liberdade) e ao seu grande espaço de trabalho em relação ao seu atravancamento, que garante uma enorme facilidade de acesso às superfícies do objecto a ser trabalhado.

No entanto algumas aplicações requerem tolerâncias muito apertadas que só as máquinas CNC conseguem assegurar. Esta é uma das limitações actuais da tecnologia

robótica sendo outra a baixa rigidez do braço robótico comparativamente ao CNC. Estas limitações restringem as aplicações robóticas de maquinagem a materiais de menor dureza, em que as forças de corte são pequenas, como por exemplo plásticos, espumas, madeira, fibra de vidro e até mesmo alumínio.

Todavia a precisão destes robôs situa-se entre as máquinas ferramenta CNC e as tecnologias de prototipagem rápida. Além disso, na produção de protótipos de grandes dimensões e quando a precisão dimensional não é muito crítica o CNC revela-se como sendo uma solução complexa e dispendiosa. Por vezes é necessário cortar a peça em várias partes para que esta possa ser introduzida na máquina, dado o seu reduzido espaço de trabalho. A solução robótica é mais flexível podendo o robô ser montado num eixo linear, o que permite ter um elevado espaço de trabalho (Figura 6). Consegue-se assim uma solução de maquinagem mais económica e rápida comparativamente com um centro de maquinagem CNC.

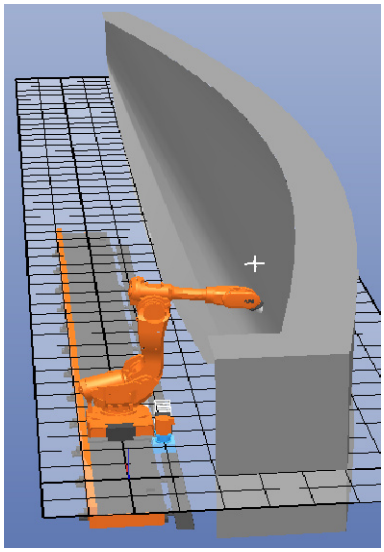


Figura 6 – Visualização em ambiente virtual de um robô montado num eixo linear (http://www.irbcam.com/IRBCAMTraining.pdf)

1.3 Pré-Maquinagem com Robôs Industriais

Outra área onde a maquinagem com robôs tem impacto é a pré-maquinagem de peças de materiais de maior dureza. Um exemplo, na indústria de fundição, é o acabamento de

blocos de motor (DePree & Gesswein, 2008), em que este passa por uma variedade de processos de pré-maquinagem desde que é retirado do molde em areia até chegar ao acabamento final que é feito pela máquina CNC. A prática comum aqui é enviar o bloco de motor da fundição para um determinado local onde se vai proceder aos vários processos de remoção de material. Toda a logística envolvida no processo de envio do produto para esse local resulta em custos acrescidos de operações e de transporte. Uma solução para a redução desses custos, neste caso, é executar operações de pré-maquinagem usando uma célula robótica junto à fundição, o que permite diminuir o peso do bloco motor quando sai da fundição. Devido à remoção de metal causada pela operação resultam não só menores custos no transporte, mas também uma redução do tempo em que a peça fica no CNC a ser trabalhada.

Em suma, a tecnologia da maquinagem robótica actualmente não deve ser entendida como substituição directa à tecnologia das máquinas ferramenta CNC mas sim como um complemento a esta. Existem aplicações em que o CNC apresenta características e capacidades muito superiores às necessárias, acarretando com isso um aumento de custos e diminuição de flexibilidade, quando comparado com soluções de maquinagem robotizada.

1.4 Robôs Recentes Optimizados para Maquinagem

Tal como já foi referido a rigidez do braço robótico é um factor limitativo para a maquinagem de materiais com maior dureza. Numa tentativa de contornar este problema os fabricantes de robôs utilizam duas abordagens: robôs de estrutura série e robôs de estrutura paralela. A maioria dos robôs industriais série apresenta configuração articulada, em que os eixos de rotação estão dispostos em série. No caso dos robôs de estrutura paralela os eixos estão dispostos em paralelo, suportando o elemento terminal (Figura 7).

A estrutura articulada possui um maior espaço de trabalho com grande acessibilidade à peça a ser maquinada, mas com rigidez limitada quando comparada com um robô de estrutura paralela. A elevada rigidez dos robôs de estrutura paralela permite uma elevada repetitibilidade de movimentos e estabilidade, em situações de carga típicas nas aplicações de maquinagem. No entanto a configuração dos seus eixos não permite ter grandes espaços de trabalho.

De seguida apresenta-se uma lista de robôs recentes utilizados para aplicações de maquinagem:

- *FANUC F-200iB* – estrutura paralela – para aplicações que requerem maior rigidez e repetitibilidade do braço robótico (Figura 7);
- *ABB IRB 6660* – estrutura série – dedicado a operações de pré-maquinagem na indústria da fundição (Figura 8);
- *Kuka KR 500* – estrutura série – com uma capacidade de carga até 500kg (Figura 9);
- *Motoman DX1350N* - estrutura série – compacto e com boa rigidez, bastante usado em operações de rebarbagem (Figura 10);
- *Stäubli RX170HSM* – estrutura série – integrando um motor de grande velocidade para aplicações de maquinagem em alta velocidade (Figura 11).

Figura 7 – Robô de estrutura paralela *FANUC F-200iB* (http://www.robots.com/ fanuc.php?robot=f-200ib)

Figura 8 – Robô de estrutura articulada *ABB IRB 6660* (http://www.abb.com/robotics)

Figura 9 – Robô de estrutura articulada *Kuka KR 500* (http://www.kuka-robotics.com/usa/en/products/industrial_robots)

Figura 10 - Robô de estrutura articulada *Motoman DX1350N* (http://www.robots.com/motoman.php?robot=dx1350n)

Figura 11 - Robô de estrutura articulada *Stäubli RX170HSM* (http://www.staubli.com/en/robotics)

1.5 Objectivos para o Trabalho a Realizar

O objectivo deste trabalho centra-se no desenvolvimento de uma solução integrada para criação de protótipos por maquinagem, baseada num robô industrial equipado com uma ferramenta de corte.

O progresso do trabalho irá compreender vários aspectos:

- Levantamento das soluções existentes no mercado;
- Definição da arquitectura da solução a implementar;
- Simulação do estudo de exemplos de maquinagem;
- Implementação de ensaios;
- Discussão de resultados/conclusões.

Na implementação de ensaios de maquinagem será utilizada a célula robótica disponível no Laboratório de Robótica do Departamento de Engenharia Mecânica da Faculdade de Engenharia da Universidade do Porto (Figura 12).

Figura 12 – Célula robotizada disponível no Laboratório de Robótica do Departamento de Engenharia Mecânica da Faculdade de Engenharia da Universidade do Porto

Os elementos constituintes da célula robotizada são:

- Robô industrial antropomórfico de seis eixos, *ABB IRB 2400*, com 1,5m de alcance máximo e 16kg de capacidade de carga;
- Mesa rotativa *IRBP*;
- Ferramenta de corte com accionamento pneumático;
- Controlador *IRC5*.

1.6 Estrutura do Relatório

Tendo em conta os objectivos atrás mencionados, o presente relatório está organizado em 7 capítulos e 1 anexo. Assim, no capítulo 2 denominado *Programação e Simulação Gráfica de Robôs* descreve-se os diferentes tipos de programação de robôs, assim como as diversas soluções de *software* dos vários fornecedores, onde é apresentado o *software* de programação *off-line*, o *RobotStudio* e o seu módulo de CAM o *Machining PowerPac*.

No capítulo 3, *Análise de Softwares de CAM* é feita uma breve referência a alguns dos softwares de CAM mais conhecidos, bem como analisadas as características dos *softwares* que serviram como ferramentas de suporte na realização do presente trabalho.

Seguidamente no capítulo 4, intitulado *Desenvolvimento de um Sistema Pós-Processador* é apresentado a solução proposta para o processo de concepção e simulação de um sistema robótico para prototipagem rápida por maquinagem, que assenta num sistema pós-processador desenvolvido na linguagem de programação VBA. Neste capítulo encontra-se também descrito o funcionamento da interface gráfica criada para o pós-processador.

O capítulo 5, denominado *Implementação da Solução na Célula Real*, retrata o procedimento experimental para a maquinagem dos protótipos desde a modelação em CAD, até aos ensaios experimentais elaborados na célula real.

No capítulo 6 são discutidos os resultados dos ensaios e são tiradas conclusões sobre a aplicabilidade da solução, evidenciando as dificuldades encontradas durante o desenvolvimento deste trabalho. Posteriormente são efectuadas sugestões para a realização de trabalhos futuros.

Por fim no capítulo 7 são listadas as referências bibliográficas que serviram como base para a realização do presente relatório.

No Anexo A disponibiliza-se o código de programação do sistema pós-processador desenvolvido.

2 Programação e Simulação Gráfica de Robôs

Neste capítulo é feita uma breve referência aos diferentes tipos de programação de robôs, bem como listadas soluções de *software* de vários fornecedores existentes no mercado. Será ainda dedicada especial atenção ao *software* de programação e simulação *off-line* dos robôs da *ABB*, o *RobotStudio*, visto ser uma ferramenta necessária para a programação *off-line* do robô *ABB IRB 2400* presente no Laboratório de Robótica do Departamento de Engenharia Mecânica da Faculdade de Engenharia da Universidade do Porto.

2.1 Programação de Robôs

Nos seus primórdios, a programação de robôs era efectuada na linha de produção, requerendo a utilização do robô. Sempre que houvesse necessidade de modificar o programa do robô, era necessário parar a produção da célula robotizada. Este tipo de programação designa-se como programação *on-line*.

Na actualidade devido à evolução tecnológica, existem *softwares* de simulação para computadores pessoais, que permitem programar os robôs, sem fazer uso directo do robô. Esta programação é denominada por programação *off-line*.

Sendo assim, quando se pretende programar um robô, tal pode ser feito usando dois métodos, um é a programação *on-line*, sendo o outro a programação *off-line*.

2.1.1 Programação *On-line*

Neste tipo de programação é necessário o uso directo do robô por parte do programador. Este movimenta o robô até às posições pretendidas e pode depois gravar cada uma das posições e instruções que criou na memória do controlador do robô, construindo assim o programa desejado.

O uso directo do robô por parte do programador pode ser feito de dois modos: *manual leadthrough* ou por *teach-pendant*. No primeiro o programador desloca manualmente o punho do robô ensinando os vários pontos ao longo da trajectória (Figura 13).

Figura 13 – Programação *on-line* através da deslocação manual do punho do robô (*manual leadthrough*)

Este tipo de programação é utilizado quando existem trajectórias complexas como é o caso de aplicação de tinta por spray, em que a trajectória deve garantir uma aplicação da tinta de um modo uniforme. No segundo, o termo técnico *teach-pendant* diz respeito ao uso da consola de interface (Figura 14) para movimentar o robô.

A forma de programação *on-line* traduz-se num método fácil de executar. No entanto o uso directo do robô, implica uma paragem forçada da célula robotizada, originando tempos não produtivos que se traduzem em quebras de produtividade.

Outro problema do uso directo do robô diz respeito ao risco de poderem ocorrer colisões entre o programador e o robô presente na célula. Nem sempre é possível ao programador manter-se fora do alcance do robô durante a fase de programação.

Figura 14 – Consola de interface do robô *ABB* (http://www.robots.com/abb.php?controller=irc5)

2.1.2 Programação *Off-line*

Ao contrário do que acontece com a programação *on-line*, a programação *off-line* não necessita do uso directo do robô, ou seja é possível o desenvolvimento de um programa sem que se tenha que parar a célula robotizada. Esta metodologia de programação do robô consiste na inserção de sucessivas linhas de comando numa linguagem específica do robô, recorrendo ao uso de *software* específico. Porém este método revela-se complicado de implementar, pois os pontos programados em linhas de comando não estão associados a uma visualização gráfica dos mesmos. Assim a única forma de verificar a existência de erros no programa é executando-o *on-line* ou seja recorrendo ao uso directo do robô.

Actualmente, graças aos avanços no *software* de programação dos robôs é possível a programação *off-line* com ferramentas de simulação gráfica. Pode-se assim movimentar o modelo do robô num ambiente virtual, gerar o programa, e verificar logo de seguida as trajectórias programadas, minimizando os erros de programação. Por fim os programas efectuados podem ser transferidos para o controlador do robô.

De facto tais são as potencialidades deste tipo de programação que se tornou comum alguns fabricantes de robôs possuírem a sua própria aplicação.

Na Tabela 2 é apresentada uma listagem de algumas soluções existentes no mercado para *software* de simulação e programação *off-line*, sendo feita uma distinção entre *software* proprietário (*software* fornecido pela empresa para a programação dos seus próprios robôs) e *software* genérico (*software* desenvolvido para a programação de robôs de vários fabricantes).

Software* de Simulação e programação *off-line	
***Software* proprietário**	
Empresa	***Software***
ABB	*RobotStudio*
Fanuc	*ROBOGUIDE*
Kuka	*KUKA SIM*
Motoman	*MotoSim EG*
***Software* genérico**	
Empresa	***Software***
Camelot Robotics	*Ropsim*
Carat robotic	*FAMOS*
Compucraft Ltd.	*RobotWorks*
Dassault Systems	*DELMIA*
EASY-ROB	*EASY-ROB*
W.A.T. Solutions	*Workspace 5*

Tabela 2 – Exemplos de algumas soluções de *software* para programação *off-line* e simulação de robôs

2.2 Software CAD/CAM e Programação *Off-line* de Robôs

A indústria das máquinas CNC desde há muito viu a necessidade do uso de padrões e adoptou o padrão DIN/ISO 66025 (mais conhecido como código G) como linguagem de programação padrão para comandar as máquinas, o que resultou no aparecimento de uma variedade de programas CAD/CAM[2] relativamente fáceis de usar para os utilizadores das máquinas CNC.

Contudo com os avanços técnicos em termos de *software* permitiu-se uma certa interacção entre os programas de CAD/CAM e o software de programação *off-line* de robôs. Deste modo a complexidade das peças a produzir deixa de ser uma barreira para a programação das trajectórias do robô. De facto tal interacção revela um enorme potencial, possibilitando usar programas CAD/CAM para criar o modelo em espaço virtual, definir todo

[2] CAD/CAM designa desenho assistido por computador (*Computer-aided design*) e manufactura assistida por computador (*computer-aided manufacturing*). Sendo usado no desenvolvimento do produto permitindo projectar e construir.

o plano de maquinagem para a sua execução e criar um programa de NC (controlo numérico) que pode ser convertido para a linguagem de programação do robô. Deste modo, programar o robô através do uso de *software* de CAD/CAM torna-se muito semelhante ao acto de programar uma máquina CNC, em que o mesmo software de CAD/CAM pode ser usado quer para programar uma máquina CNC quer para programar um robô. Neste último é apenas necessário converter o programa NC gerado pelo CAM, para um formato passível de ser lido pelo controlador do robô.

De seguida apresentam-se na tabela 3 alguns exemplos de *software* para o pós-processamento[3] dos programas de CAM em linguagem de programação de robôs

***Software* de pós-processamento**	
***Software* proprietário**	
Empresa	***Software***
IRBCAM GmbH	*IRBCAM*
Kuka	*Kuka CAMRob*
Motoman	*G-Code Converter EG*
***Software* genérico**	
Empresa	***Software***
Jabez Technologies	*Mastercam + Robotmaster*
Roboris	*Eureka*
Scienzia Machinale	*ARPP*

Tabela 3 – Exemplos de *software* de pós-processamento

É importante ter em conta que actualmente no mercado, os fabricantes de *software* para os robôs industriais adoptam diferentes abordagens na interacção dos seus programas de simulação e programação *off-line* com os programas de CAM existentes.

Foram assim identificadas três abordagens distintas. A primeira utiliza uma aplicação de CAD/CAM genérica, que incorpora pós-processadores para robôs de distintos fabricantes. Para executar este pós-processamento, define-se previamente qual o modelo do robô que irá ser utilizado, para assim se gerar o programa do robô na linguagem de programação pretendida.

[3] A expressão “pós-processamento” aqui usada diz respeito à conversão/tradução dos programas gerados pelo CAM em linguagem de programação de robôs.

Todo este processo é realizado no interior de um software de CAD/CAM genérico (Figura 15), que possui um módulo que permite fazer o pós-processamento e a respectiva simulação gráfica do programa criado. O programa do robô pode depois ser transferido para o controlador para ser implementado no robô real.

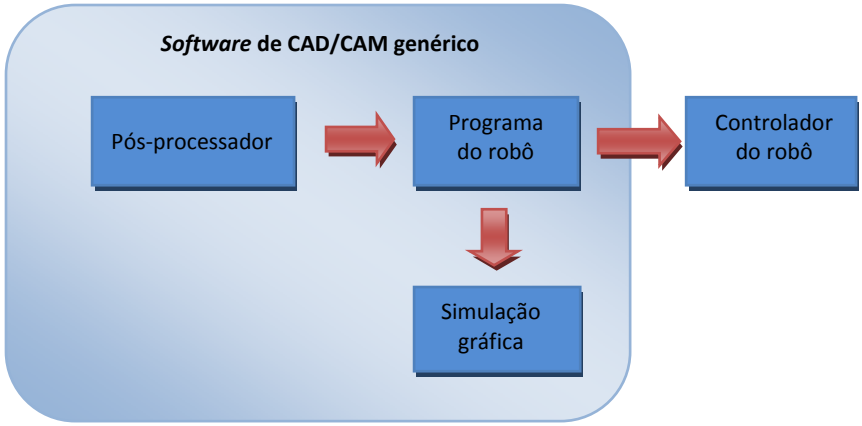


Figura 15 – Esquema ilustrativo da solução baseada num *software* de CAD/CAM genérico

Exemplos de *software* de CAD/CAM genérico:

- *Mastercam+Robotmaster*;
- *Delcam Powermill*.

A segunda abordagem envolve o uso de dois *softwares* distintos, um *software* de CAD/CAM e um *software* genérico de simulação robótica. O programa de CAD/CAM é usado para a criação do percurso da ferramenta na peça, gerando um programa em linguagem APT ou ISO (código G) com o auxílio de um pós-processador interno do programa de CAM.

O programa em linguagem APT/ISO é importado para um *software* genérico de simulação, o qual possui um pós-processador, em que escolhendo o modelo do robô, se faz a conversão do programa gerado pelo CAM em linguagem de programação do robô, gerando-se o programa para o robô pretendido (Figura 16).

Para verificar a aplicabilidade do programa antes de o transferir para o controlador do robô, é possível efectuar a sua simulação gráfica.

Existe ainda a possibilidade de instalação de um módulo de CAM integrado em alguns *softwares* genéricos de simulação. Assim através de um *software* de CAD é definida a geometria do objecto pretendido, que pode ser guardado em ficheiro de formato neutro, sendo

depois importado para o software genérico de simulação. A geração de trajectórias é feita automaticamente nas superfícies do modelo de CAD importado.

Este módulo de CAM é normalmente mais pequeno e mais limitado que um programa de CAM normal, em contrapartida possui a vantagem de permitir obter programas criados já em linguagem de programação de robôs, sem necessidade de recorrer a um pós-processador.

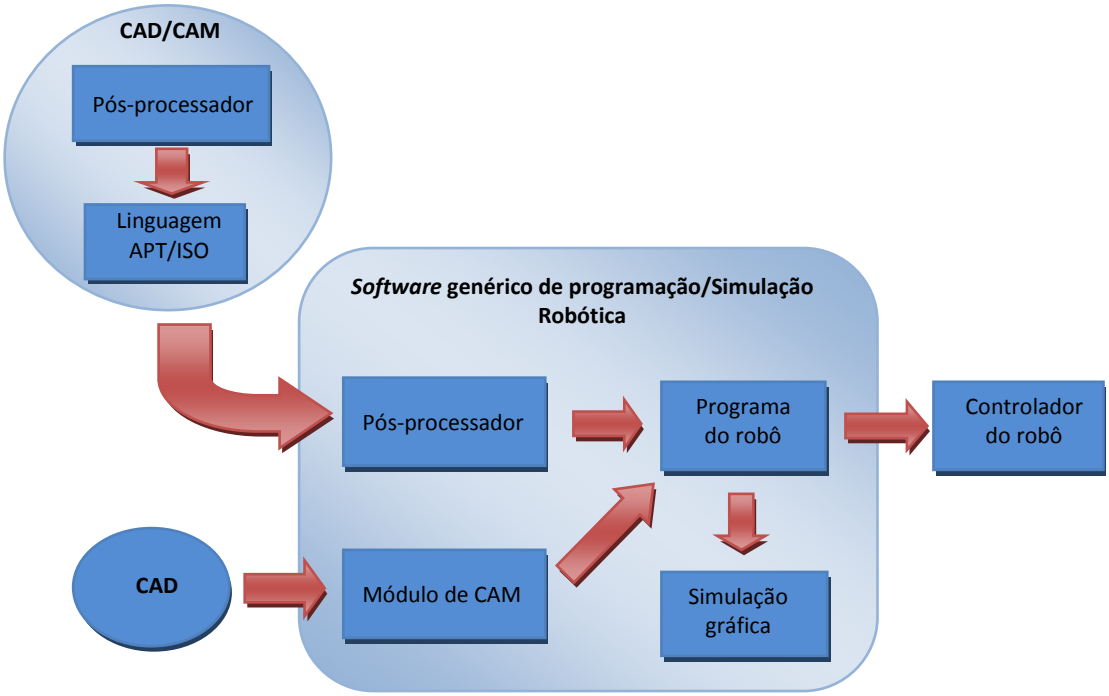


Figura 16 – Esquema ilustrativo da solução baseada num *software* genérico de programação/simulação robótica

Exemplos de *software* genérico de programação/simulação robótica:

- *Eureka*;
- *ARPP*.

Exemplo de módulo de CAM integrado num *software* genérico de simulação:

- *ARPPCAM.*

A terceira abordagem centra-se no uso de *software* proprietário de programação *off-line* de robôs para gerar o programa do robô. Tal como acontece no caso anterior, é utilizado um *software* de CAD/CAM para gerar o programa em linguagem APT/ISO, que depois é pós-processado para gerar o programa do robô. No entanto o pós-processador, como está instalado num software proprietário de programação *off-line*, apenas gera os programas na linguagem de programação dos robôs do fabricante.

A disponibilização de um módulo de CAM integrado é uma possibilidade em alguns *softwares* proprietários de programação *off-line*. O processo de geração do programa do robô é semelhante ao que acontecia na abordagem anterior, em que um modelo criado em software de CAD é importado para o software de programação *off-line* de robôs, gerando-se as trajectórias automaticamente nas superfícies do modelo de CAD importado.

Em alguns destes softwares também é possível a instalação de um conversor (pós-processador de baixo nível) no *software* de programação *off-line*, que utiliza os dados geométricos dos ficheiros criados em CAD/CAM ou por edição manual, convertendo-os em pontos de programação para o robô. Os pontos são usados para criar o programa do robô, que pode ser simulado no *software* de programação *off-line* e transferido para o controlador do robô (Figura 17).

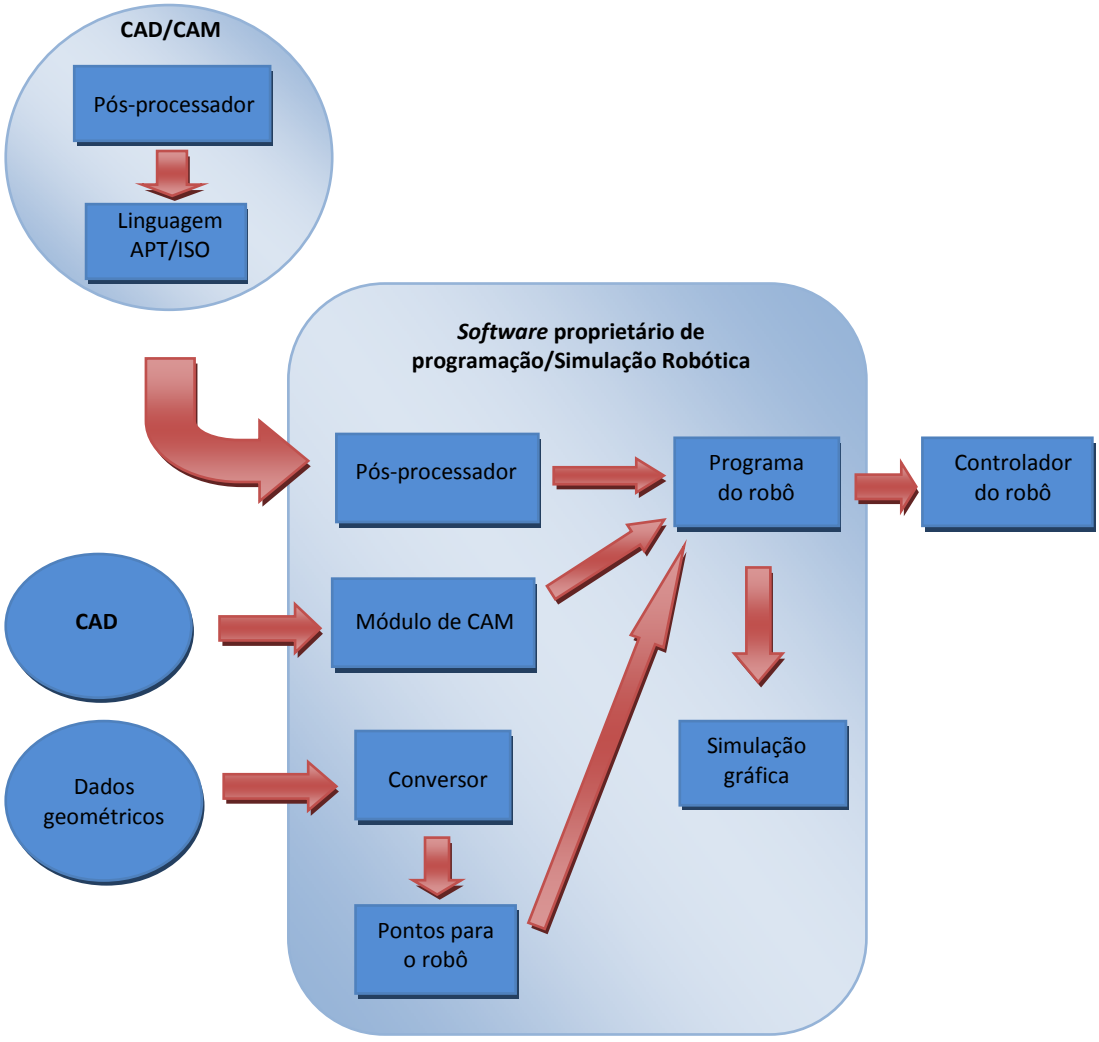


Figura 17 – Esquema ilustrativo da solução baseada num *software* proprietário de programação/simulação robótica

Exemplos de *software* proprietário de programação/simulação robótica:

- *RobotStudio*;
- *KUKA SIM*;
- *MotoSim EG*.

Exemplos de *software* proprietário para o pós-processamento que funcionam no interior de um *software* proprietário de programação/simulação robótica:

- *IRBCAM*;
- *Kuka CAMRob*;
- *Motoman G code converter*.

Exemplo de Módulo de CAM integrado em *software* proprietário de programação *off-line:*

- *Machining PowerPac* para o *RobotStudio* da *ABB*.

Exemplo de conversores que podem ser instalados em *software* proprietário de programação *off-line*:

- *Motoman Points Importer EG*;
- *Coordinate File Import Add-in* para o *RobotStudio*.

Analisando as diferentes soluções existentes no mercado verifica-se que os *softwares* genéricos apresentam uma flexibilidade bastante interessante, são capazes de gerar uma solução compatível com vários robôs de diferentes fabricantes, isto é os seus pós-processadores permitem gerar programas para os robôs dos diferentes fabricantes.

Outra vantagem reside no facto de alguns destes *softwares* genéricos estarem completamente integrados num programa de CAD/CAM (caso de Mastercam+Robotmaster e Delcam Powermill), o que permite usufruir ao máximo de todas as ferramentas de CAM disponíveis sem ser necessário a conversão do programa de CAD/CAM em linguagem ATP/ISO, gerando os programas mais rapidamente.

Contudo há que ter em consideração que as soluções de *software* genérico usam no seu ambiente de simulação controladores virtuais que são genéricos. Isto pode constituir uma enorme desvantagem já que os controladores virtuais genéricos, não possuem o modelo cinemático proprietário do robô real que descreve a cinemática real do robô. Sendo assim, as

simulações realizadas podem acarretar algumas imprecisões quando comparadas com implementação na célula real.

Os softwares proprietários, ao contrário dos softwares genéricos, possuem controladores virtuais no ambiente de simulação, que são idênticos aos controladores reais. Assim os programas gerados utilizam o modelo de cinemática real do robô, o que permite a execução de simulações bastante realistas.

Tendo em consideração todos estes aspectos, a escolha da solução a implementar para o presente projecto incidiu num *software* de programação *off-line* proprietário.

Feita uma pesquisa sobre os vários softwares proprietários existentes, conclui-se que a empresa da *ABB* fornece uma solução bastante completa possuindo:

- Variedade de softwares de CAD/CAM com os quais é compatível;
- Módulo de CAM que pode ser instalado sem custos adicionais;
- Um modelo de cinemática real do robô no *software* de simulação;
- Capacidade de transferência directa do programa do robô para o controlador do robô sem necessidade de qualquer pós-processamento;
- Disponibilidade total de experimentação do *software* por um período de 30 dias com funcionalidade completa.

Por todas estas razões o *software* de programação *off-line* da *ABB* o *RobotStudio* foi escolhido como ferramenta de apoio à realização do presente projecto.

2.3 *RobotStudio* da *ABB*

No âmbito deste trabalho, é utilizado o *software* de programação *off-line* da *ABB* o *RobotStudio* que utiliza a linguagem de programação RAPID da *ABB*.

Neste tópico é feita uma breve abordagem às capacidades do *RobotStudio* e da aplicação *Machining PowerPac* (*Add-in* do *RobotStudio*).

2.3.1 Potencialidades do *RobotStudio*

A empresa *ABB* introduziu no mercado um conceito de programação para os seus robôs que designou por *true Off-line Programming*. Tal nome surge do facto do *RobotStudio* utilizar um controlador virtual que é uma cópia exacta do controlador real que corre nos robôs utilizados na produção, permitindo executar simulações bastante realistas.

Numa visão geral sobre o desenvolvimento de programas, o *RobotStudio* apresenta ferramentas muito úteis, as quais permitem que seja possível:

- Importação de ficheiros CAD;
- Geração de trajectórias automaticamente a partir de um modelo CAD (através do *add-in Machining PowerPac)*;
- Optimização de trajectórias (indicando possíveis pontos de singularidade);
- Verificação de alcance;
- Verificação tridimensional das trajectórias programadas;
- Detecção de colisões;
- Adição de aplicações extra (soldadura, maquinagem etc.);
- Verificação da aplicabilidade do programa na célula real[4];
- Possibilidade de exportar os programas criados em *RAPID* ou em *RW Machining FC*[5].

Outra ferramenta que o *RobotStudio* possui através do *add-in Machining PowerPac* é a possibilidade de usufruir de 3 modos de operação para a maquinagem: o *Normal Process*, o *FC SpeedChange* e o *FC Pressure.* O *Normal Process* é baseado no comando do robô através do controlo de posição tendo como objectivo fazer o robô cumprir cada ponto da sua trajectória programada.

[4] Graças ao controlador virtual que garante de certa forma que se a simulação em ambiente virtual for bem sucedida, esta funcionará na célula real.
[5] *RW Machining FC* designa *RobotWare Machining Force Control* que é um *software* da ABB para o controlo de força.

O *FC Pressure* faz com que o robô se adapte às superfícies de trabalho, tornando-o sensível a forças de contacto, sendo para isso necessário ensinar uma trajectória de referência, definindo a direcção sobre a qual o robô aplicará uma dada força. Sendo assim o robô poderá modificar cada ponto da sua trajectória de referência, de modo a cumprir o valor especificado para a força.

O *FC SpeedChange* tem como principal objectivo controlar a velocidade da trajectória em função da força de contacto limite especificada. Deste modo, quando é excedido o valor para a força, a velocidade irá ser automaticamente reduzida, contrariamente se houver um decremento da força a velocidade será automaticamente aumentada. Ao contrário do que acontece no *FC Pressure*, o *FC SpeedChange* não altera a trajectória do robô, mas sim a velocidade da trajectória.

Ambos os modos de operação o *FC Pressure* e o *FC SpeedChange* requerem que o robô esteja equipado com um sensor de força e que o controlador disponha do *software* adequado.

O sistema de comando do robô que envolve o uso do controlo de força apesar de ser inovador, é uma tecnologia muito recente e ainda em expansão. Na área da prototipagem rápida em que se pretende gerar uma forma específica para o protótipo a maquinar, não é utilizado controlo de força, mas sim o controlo de posição e velocidade do braço robótico, de modo a definir as trajectórias pretendidas.

2.3.2 Exemplo de Aplicação

Para a familiarização com o *software RobotStudio* executou-se um programa de maquinagem para permitir a gravação de letras numa peça. Partindo da modelação da peça num *software* de CAD 3D, o *SolidWorks*, procedeu-se então à importação da referida peça para o *software* de programação de robôs (Figura 18).

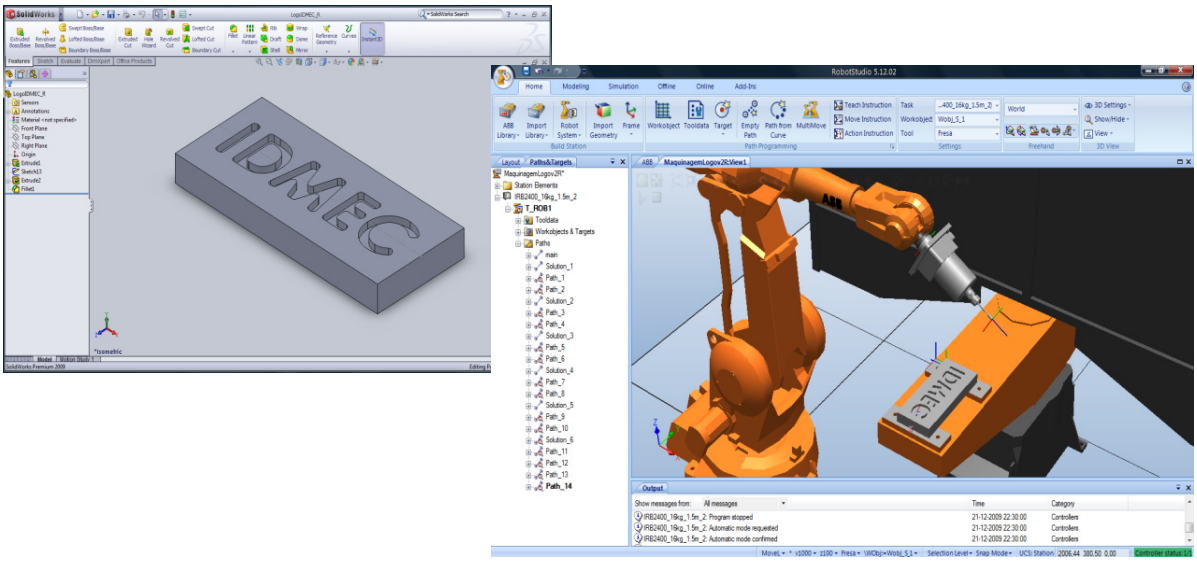


Figura 18 – Modelação da peça no SolidWorks (à esquerda) e importação do ficheiro de CAD para o ambiente de trabalho do RobotStudio (à direita)

Importa salientar que o *RobotStudio* também permite a modelação tridimensional de geometrias, através do uso da opção *modelling*, contudo este *software* não é específico para criação de formas complexas, contendo apenas algumas ferramentas básicas, para a criação de formas mais simples. Por essa razão recorreu-se ao uso do *SolidWorks* em que a peça criada foi modelada e guardada em ficheiro de extensão *SAT* (associado ao formato neutro *ACIS*) de modo a ser passível a sua importação para o *RobotStudio*.

De seguida procedeu-se à execução da solução de maquinagem recorrendo à aplicação *Machinning PowerPac*, que é um módulo de CAM disponível no *RobotStudio* para gerar automaticamente as trajectórias sobre as superfícies da peça. O resultado pode ser visto na Figura 19.

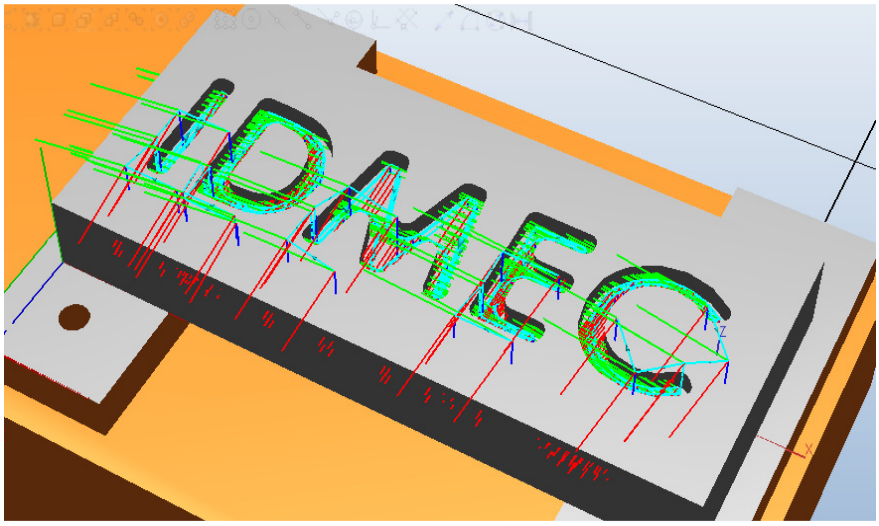


Figura 19 - Visualização das trajectórias geradas

De seguida foi feita a sincronização com o controlador virtual e gerado o código RAPID, bem como executada a simulação das trajectórias feitas pelo robô.

Este trabalho embora se tenha revelado uma tarefa simples envolvendo apenas maquinagem em 3 eixos, foi suficiente para tirar algumas conclusões acerca deste tipo de abordagem para a criação de programas de maquinagem.

De facto o *Machining PowerPac* do *RobotStudio* revela potencialidades apresentando uma solução rápida e fácil na execução de trajectórias podendo-se manipular diversos parâmetros como compensação de raio de ferramenta, velocidades de maquinagem, criação de pontos de aproximação à peça, etc. No entanto apresenta limitações, no que diz respeito à maquinagem de superfícies interiores. O *Machining PowerPac* ao gerar as trajectórias de maquinagem também cria pontos de aproximação e de saída na peça a maquinar. Estes pontos não têm em conta os limites físicos da peça, originando trajectórias incorrectas que provocam colisões da ferramenta com a peça no acto de aproximação e saída da ferramenta à peça a maquinar (Figura 20).

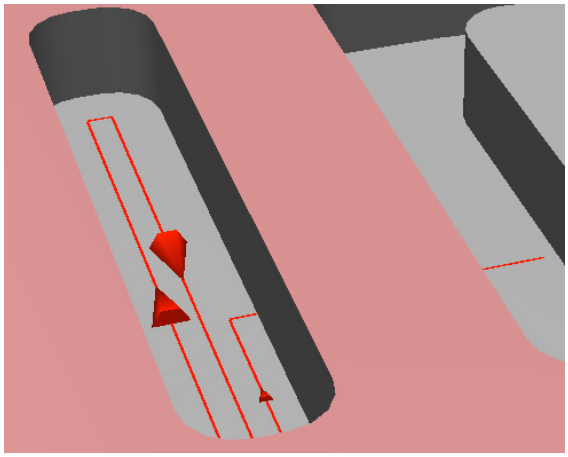


Figura 20 – Trajectórias incorrectas que causam colisões da ferramenta com a peça

No sentido de resolver este problema o utilizador pode sempre editar manualmente estes pontos problemáticos da trajectória, no entanto tal solução é inviável em peças mais complexas.

Deste modo abandonou-se a solução do módulo de CAM do *RobotStudio*, o *Machining PowerPac*, em alternativa procurou-se explorar outras soluções, tendo em vista o alcance dos objectivos do presente trabalho.

3 Análise de Softwares de CAM

Após uma fase inicial de familiarização com o *software* de programação *off-line* do *RobotStudio*, procedeu-se à análise de alguns softwares de CAM, que irão ser utilizados como ferramentas de suporte para o processo de criação das trajectórias sobre a peça.

No desenvolvimento deste trabalho foram utilizados como softwares de CAM, o *MeshCAM* e o *GSimple*. A escolha destes é justificada pela sua disponibilidade e funcionalidade total e também por se enquadrarem nas necessidades do presente projecto.

Será também feita uma breve referência aos softwares de CAM existentes no mercado como o *PowerMill* da *Delcam* e o *MasterCAM*.

3.1 *PowerMill* da *Delcam*

O *PowerMill* é um *software* de CAD/CAM com capacidade para a execução de trajectórias de maquinagem em 5 eixos. A sua interface é bastante avançada (Figura 21) permitindo simular todo o percurso da ferramenta, bem como a possibilidade de visualização da remoção de material durante o processo.

Sendo um *software* de CAD/CAM também possui ferramentas de CAD 3D permitindo a criação de objectos tridimensionais (embora não possuindo um CAD tridimensional tão avançado como um *software* inteiramente dedicado ao CAD, como por exemplo o *Solidworks*).

Quanto à sua funcionalidade o *PowerMill* é de facto uma solução de *software* CAD/CAM bastante completa, oferecendo estratégias de maquinagem bastante eficientes em 5 eixos, em que é possível maquinar superfícies complexas de forma eficiente, tirando partido de ferramentas automáticas para detecção de colisões entre a ferramenta e a peça durante a operação de maquinagem.

O pós-processador do *PowerMill* gera o programa NC para variadas máquinas CNC existentes no mercado. Existem ainda empresas que se dedicam à elaboração de pós-processadores para robôs industriais usando o *PowerMill*. Um caso concreto é o da empresa *Programming Plus Inc* que criou um pós-processador para os Robôs da *Kuka*.

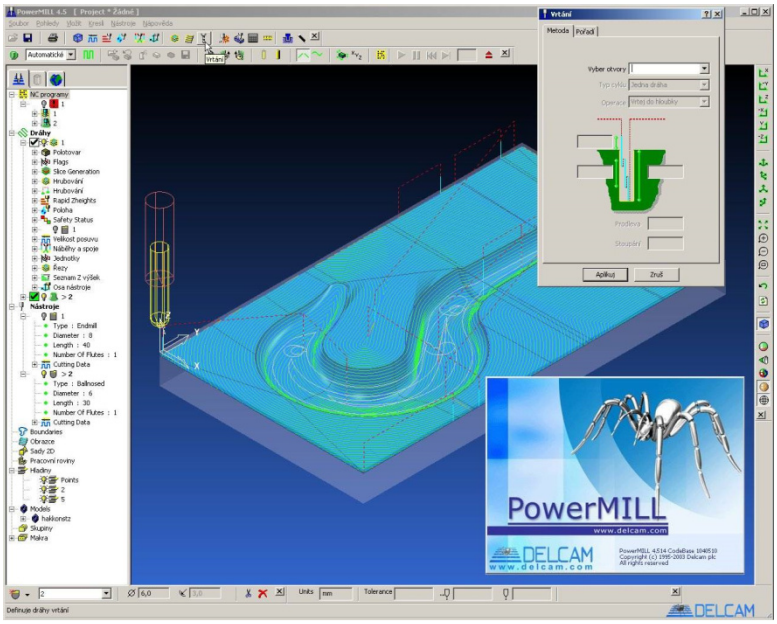


Figura 21 – Visualização da interface gráfica do software de CAD/CAM PowerMill (http://u12134.fsid.cvut.cz/?udaj=link&list=skup_technologie_obrabeni)

3.2 *MasterCAM*

O *MasterCAM* é um software de CAD/CAM para maquinagem em 5 eixos, e tal como o *PowerMill* possui uma interface e funcionalidades bastante avançadas para a geração das trajectórias de maquinagem.

O pós-processamento do programa gerado no MasterCAM produz programas NC para diferentes máquinas CNC à escolha. O MasterCAM permite ainda o pós-processamento dos

programas gerados para robôs industriais dos mais conhecidos fabricantes, como a *FANUC, Stäubli, ABB e Kuka* através de um módulo de programação, simulação e pós-processamento no interior do *MasterCAM*, designado por *RobotMaster*, constituindo assim uma solução integrada.

Na Figura 22 é possível visualizar o *software* de CAD/CAM MasterCAM bem como o seu módulo de programação e simulação de robôs o Robotmaster.

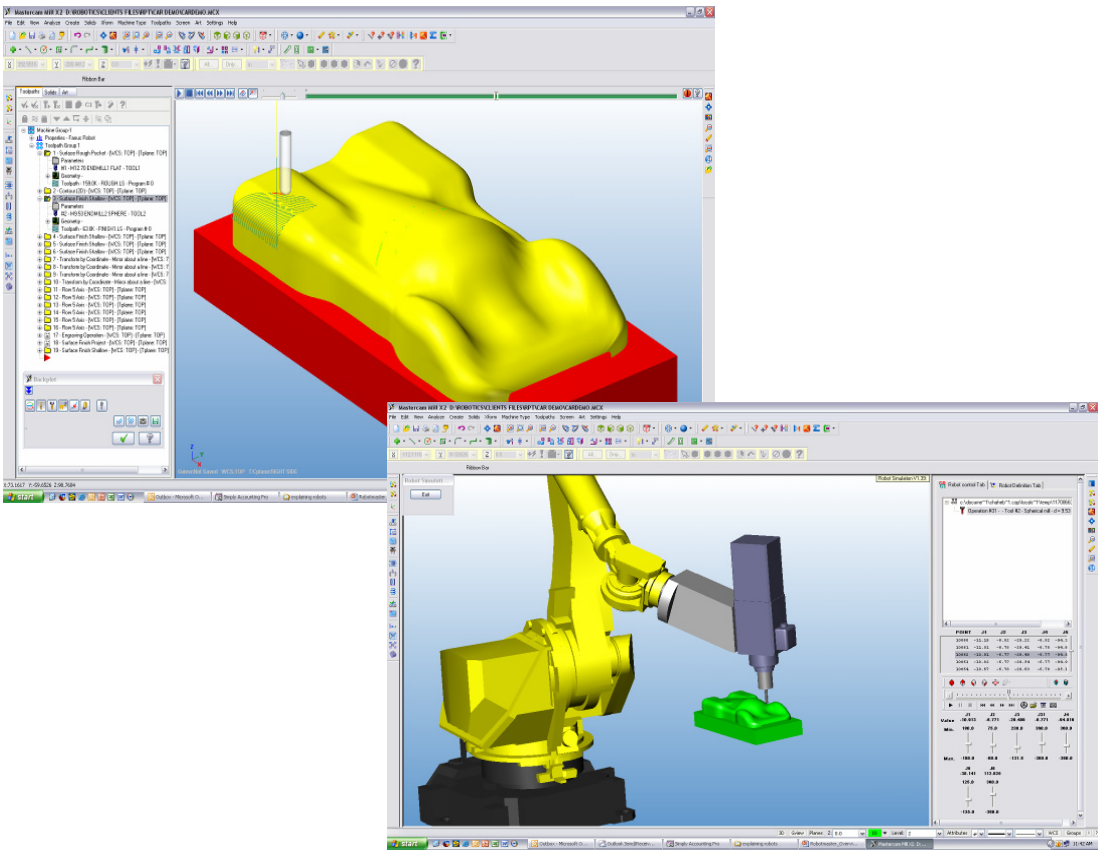


Figura 22 – Visualização da interface gráfica do software de CAD/CAM MasterCAM (em cima à esquerda) e do modulo de programação e simulação robótica, o Robotmaster (em baixo à direita) (http://www.inhousesolutions.com/products/robotmaster/mastercam_robotmaster.php)

Apesar destes dois softwares, o *PowerMill* e o *MasterCAM*, se revelarem como extremamente interessantes no que diz respeito às funcionalidades, estes não estão directamente disponíveis ao utilizador. A utilização destes programas requer o uso de licenças de utilização, o que constitui uma solução economicamente não muito viável no que toca à realização do presente projecto.

Tendo isso em consideração, foram explorados outros softwares de CAM que são apresentados a seguir.

3.3 *MeshCAM* da *GRZ Software*

O *MeshCAM* é um programa de CAM produzido pela *GRZ Software*, que permite gerar trajectórias de maquinagem para máquinas CNC de 3 eixos.

A sua interface permite que modelos 3D sejam importados de um programa de CAD (Figura 23), sendo capaz de gerar as trajectórias sobre as superfícies do modelo importado.

O *MeshCAM* é essencialmente um programa de CAM 3D para criação de trajectórias para maquinagem em três dimensões o que significa que os percursos de movimentação da ferramenta são efectuados nos 3 eixos coordenados X, Y e Z, sendo também capaz de gerar percursos de movimentação da ferramenta nos três eixos em simultâneo.

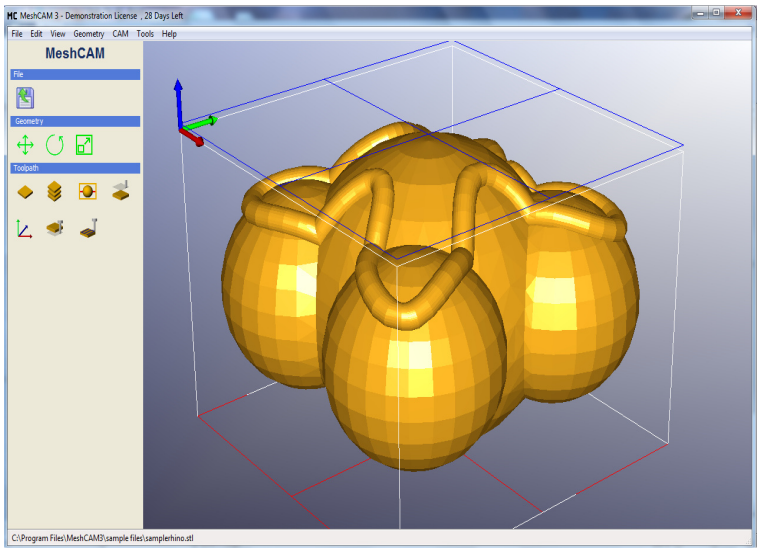


Figura 23 – Ilustração da interface gráfica do *software* de CAM *MeshCAM* e de um exemplo de um modelo CAD importado em formato STL

O *software* permite importar ficheiros de CAD em formato STL (*Stereolithography*). Este é um formato neutro muito usado nos sistemas de prototipagem rápida por estereolitografia e também no processo de impressão tridimensional utilizando impressoras 3D, permitindo compatibilidade com uma variedade de programas de CAD.

O *software MeshCAM* possui uma interface gráfica bastante intuitiva, permitindo uma rápida aprendizagem por parte do utilizador, mesmo que este não possua experiência em softwares de CAM.

A geração de trajectórias é feita automaticamente a partir de um modelo CAD importado, sendo para isso necessário definir alguns parâmetros de maquinagem necessários tais como:

- Dimensões do bloco inicial a ser maquinado (Figura 24);
- O zero do programa (o código G gerado é sempre relativo ao zero do programa);
- A máxima profundidade de corte;
- A distância da ferramenta à face de topo do bloco nos movimentos rápidos de posicionamento;
- O tipo de ferramenta, dimensões da ferramenta, velocidade de accionamento (Figura 25).

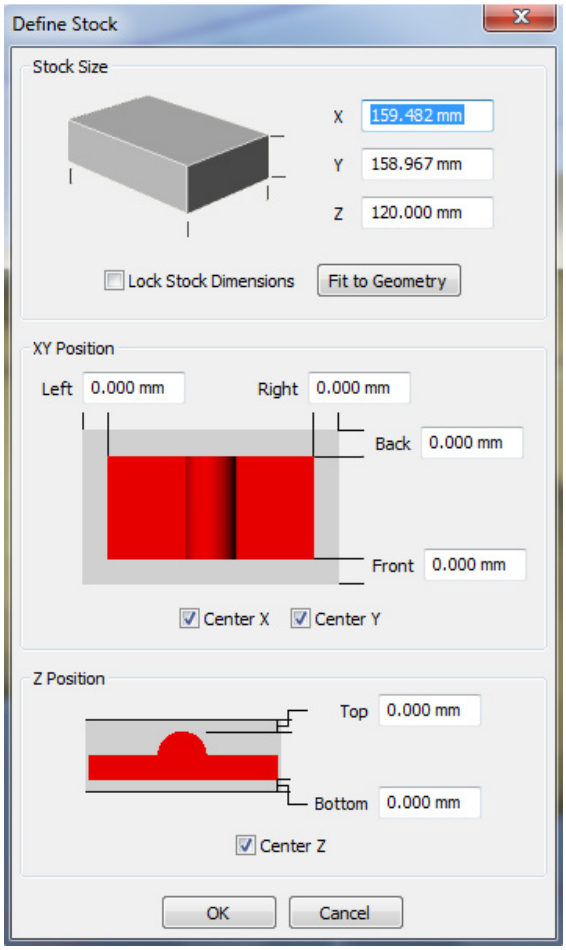


Figura 24 – Janela de diálogo do *MeshCAM* para a definição das dimensões do bloco inicial a ser maquinado

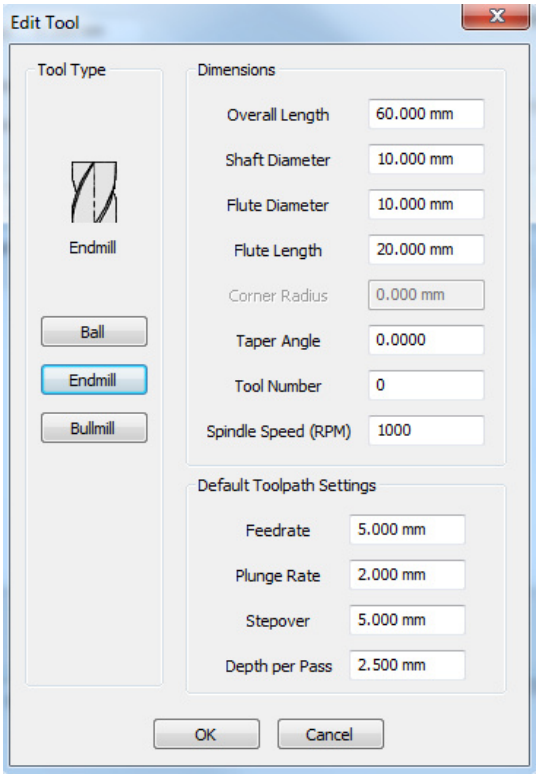


Figura 25 – Janela de diálogo do *MeshCAM* para a definição dos parâmetros da ferramenta

Existem ainda outros parâmetros que o *MeshCAM* necessita para criar as trajectórias da ferramenta:

- Tolerância – Define a precisão com que as trajectórias são calculadas. Uma maior tolerância significa um menor número de pontos utilizados para descrever a trajectória ao longo da superfície a maquinar. Este parâmetro também influencia o tempo de cálculo necessário à produção das trajectórias, sendo que quanto maior a tolerância, menor o tempo de cálculo;
- Profundidade de corte (*depth per pass*) – Define a profundidade de cada passagem da ferramenta;
- *Stepover* – Define a distância entre as passagens paralelas da ferramenta no ciclo de corte para uma cota Z constante;
- *Feedrate* – velocidade com que a ferramenta se movimenta em relação à peça;
- *Plunge Rate* – Velocidade com que a ferramenta avança em profundidade na cota Z;
- *Stock to Leave* – Representa a quantidade de material que vai ser deixada para a operação de acabamento.

Há também a possibilidade de executar o acabamento definindo uma segunda ferramenta e uma direcção para a execução do processo (Figura 26).

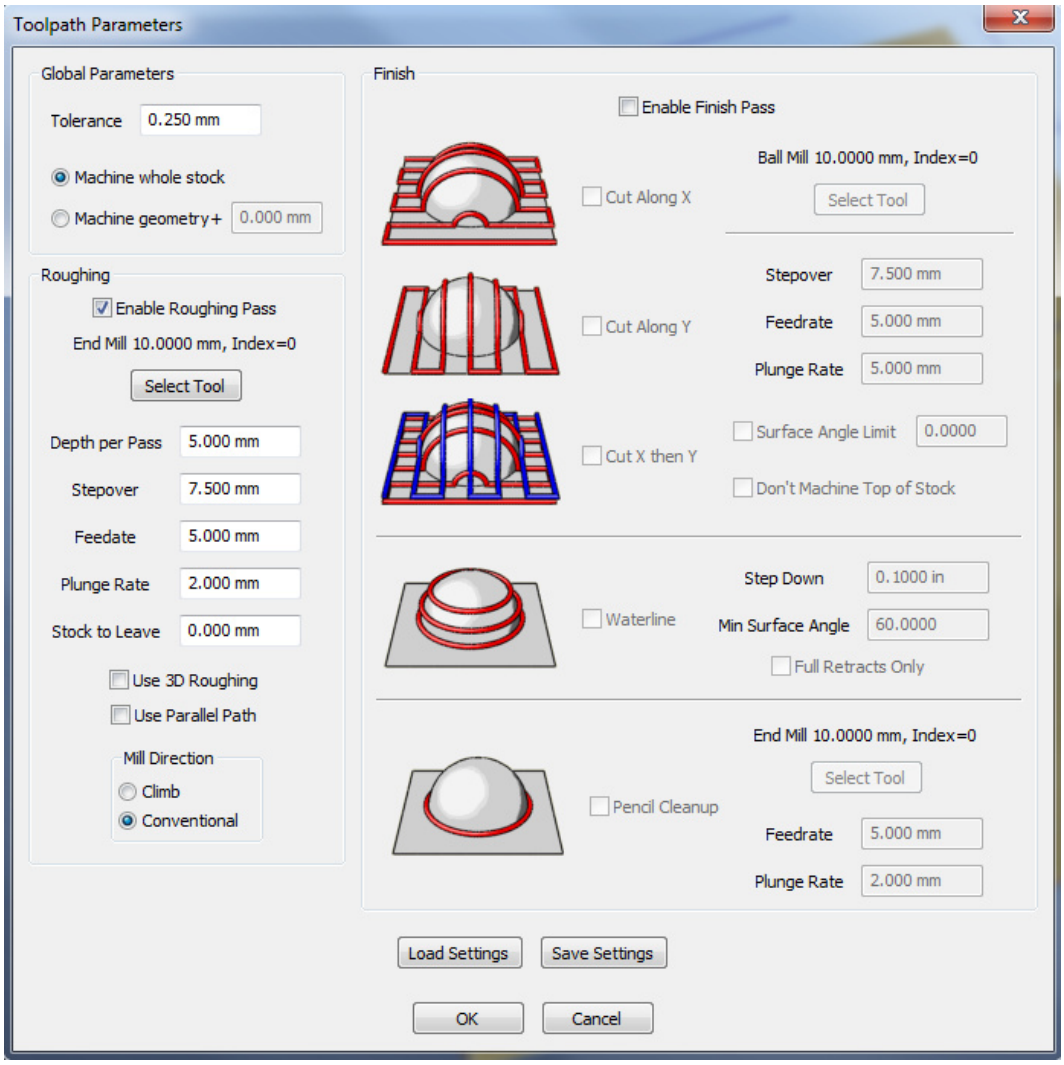


Figura 26 – Definição de alguns dos parâmetros necessários para gerar as trajectórias da ferramenta no *MeshCAM*

Os parâmetros que são descritos a seguir são opcionais, devendo ser utilizados sempre que as trajectórias criadas não sejam satisfatórias.

- *Use 3D Roughing* – Seleccionando esta opção, permite que a ferramenta siga os contornos verticais da peça. Assim cada passagem durante a sua execução pode variar na cota Z;

- *Use parallel path* – Seleccionando esta opção, as passagens são criadas usando uma série de linhas paralelas ao eixo X e ao mesmo tempo garantindo que a trajectória segue o contorno da superfície da peça.

Assim que os parâmetros de maquinagem estão definidos, as trajectórias da ferramenta na peça tornam-se visíveis. Pode-se então recorrer a um pós-processador para gerar o código G num ficheiro de extensão ".nc". Visto que cada máquina interpreta um formato específico do código G, o pós-processador presente no *MeshCAM* permite a criação de código G para diferentes máquinas CNC. Este pós-processador requer um ficheiro de configuração para poder descrever qual o formato de saída do código G presente no ficheiro do programa NC. Estes ficheiros de configuração têm a extensão ".con"e podem ser encontrados na pasta "Posts" de instalação do *MeshCAM* (Figura 27). Caso o pós-processador não gere o ficheiro do programa NC com o formato do código G que o utilizador pretende para a sua máquina CNC, há sempre a possibilidade de editar o ficheiro de configuração com um editor de texto (os ficheiros de configuração não são mais do que ficheiros de texto) (Figura 28) de modo a criar um programa NC com o código G no formato desejado pelo utilizador.

O *MeshCAM* é de facto uma alternativa interessante aos *softwares* de CAM proprietários e genéricos mais conhecidos no mercado, tendo como principais vantagens o facto de ser bastante expedito, fácil de utilizar e ser gratuito e perfeitamente funcional num período de 30 dias. Depois desse período o utilizador terá que adquirir uma licença para continuar a usar o programa. O *MeshCAM* disponibiliza as suas licenças a um preço muito mais apelativo que os habituais programas de CAM disponíveis no mercado.

No entanto o *MeshCAM* também possui limitações, principalmente no que toca ao programa gerado, de facto o *MeshCAM* não faz uso de interpolações circulares no cálculo das trajectórias da ferramenta, apenas usa interpolações lineares. Sendo assim ao gerar uma trajectória sobre uma superfície curva é feita uma aproximação por trajectórias lineares, em pequenos segmentos, sendo o comprimento desses segmentos definido de acordo com a tolerância especificada.

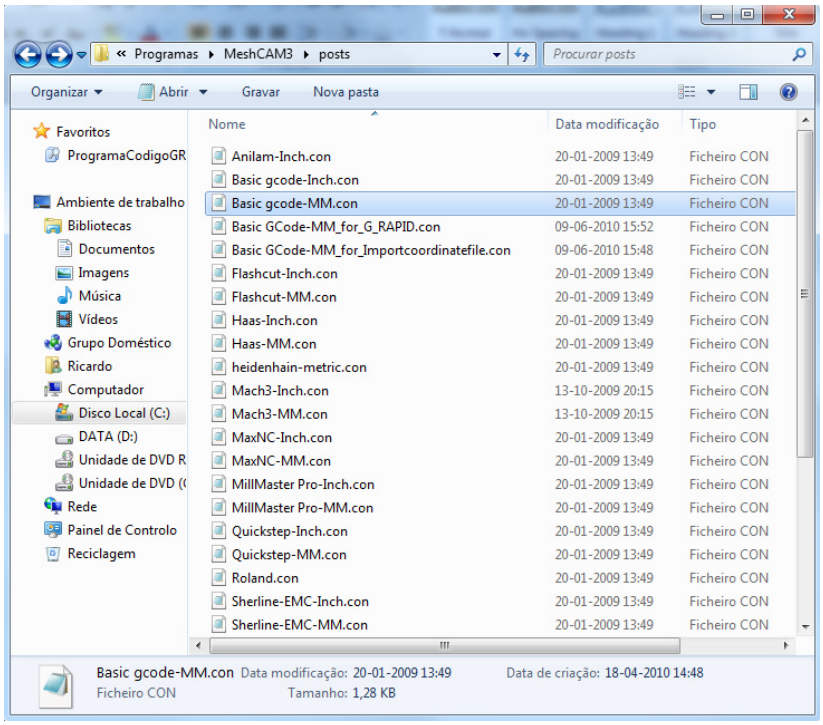


Figura 27 – Ficheiros de configuração para a criação do código G num formato específico para uma determinada máquina CNC

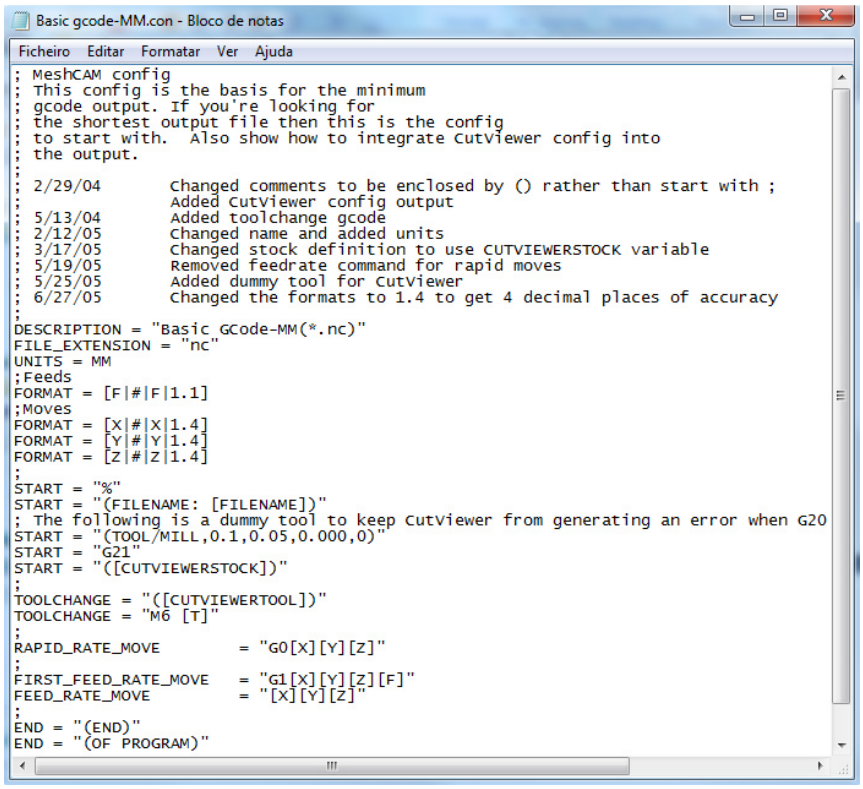


Figura 28 – Exemplo de um ficheiro de configuração aberto com um editor de texto

3.4 *G-SIMPLE*

Tal como o *MeshCAM*, o *G-SIMPLE* é um *software* de CAD/CAM que permite gerar trajectórias de maquinagem para máquinas CNC de 3 eixos. Este *software* possui ferramentas básicas de desenho de CAD 2D, permitindo a visualização de qualquer objecto criado numa perspectiva isométrica (embora a edição dos objectos de CAD seja apenas possível numa perspectiva bidimensional).

O *G-SIMPLE* é essencialmente um *software* CAD/CAM para maquinagem em 2,5D (duas dimensões e meia), isto é, permite a criação de trajectórias da ferramenta nos 3 eixos coordenados X, Y e Z, no entanto apenas gera as trajectórias para movimentação da ferramenta em dois eixos em simultâneo.

Para a geração de trajectórias de maquinagem, o utilizador tem de definir um bloco inicial (Figura 29), seleccionar as entidades que pretende maquinar e escolher a operação de maquinagem pretendida. O programa tem disponível uma biblioteca de ferramentas que o utilizador pode escolher (Figura 30), podendo também criar a sua própria ferramenta se for necessário.

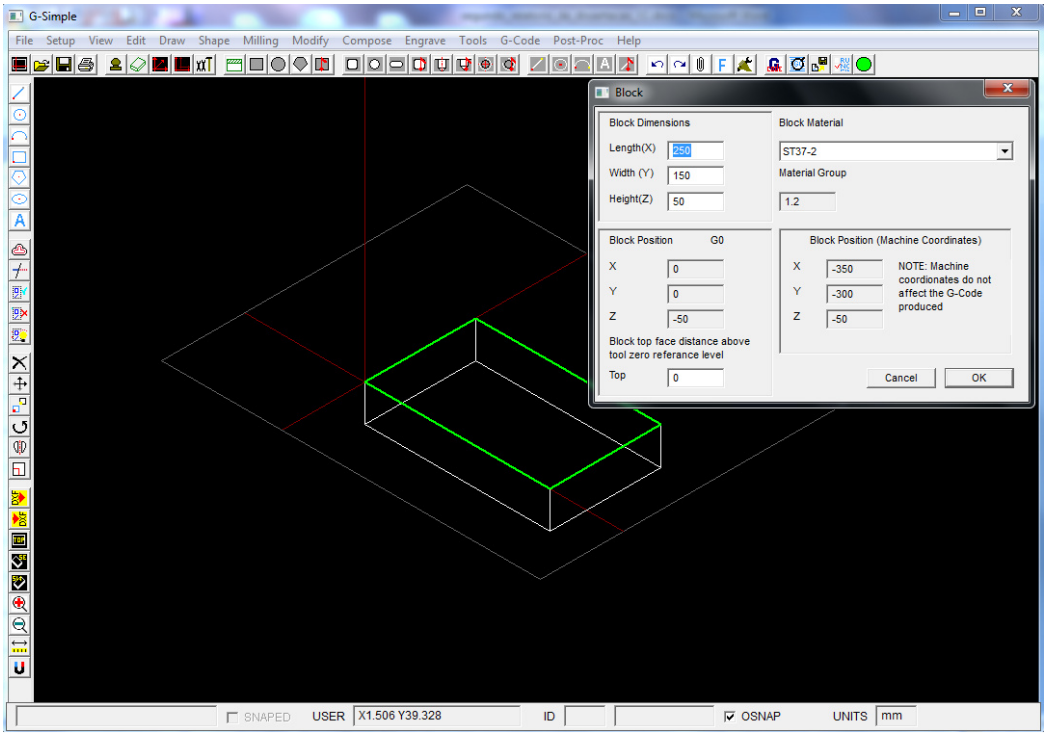


Figura 29 – Exemplo de definição das dimensões do bloco inicial no *G-SIMPLE*

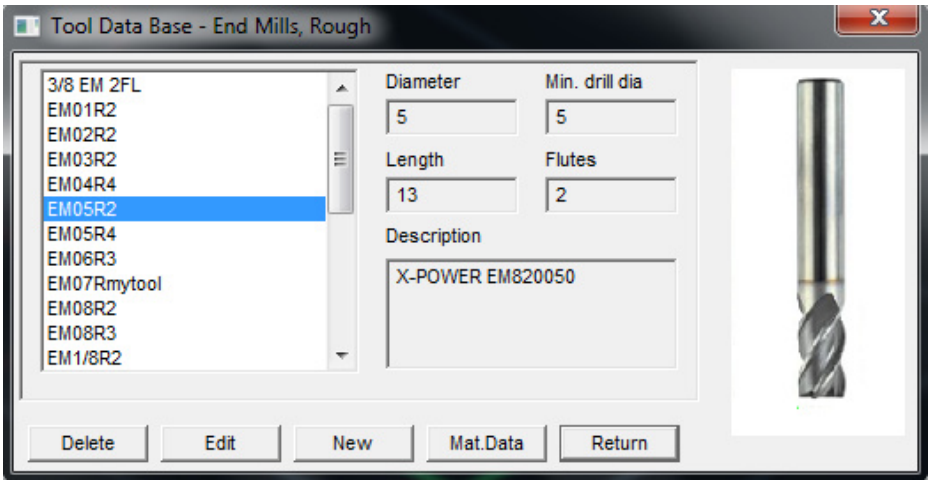


Figura 30 – Biblioteca de ferramentas de corte disponível no *G-SIMPLE*

O *G-SIMPLE* permite a importação de ficheiros em extensão DXF criados num software de CAD/CAM. O programa permite gerar e visualizar o Código G criado, sendo possível fazer a simulação do percurso da ferramenta, bem como ver quais as etapas do código G que estão a ser executadas (Figura 31), podendo-se guardar no final o programa NC.

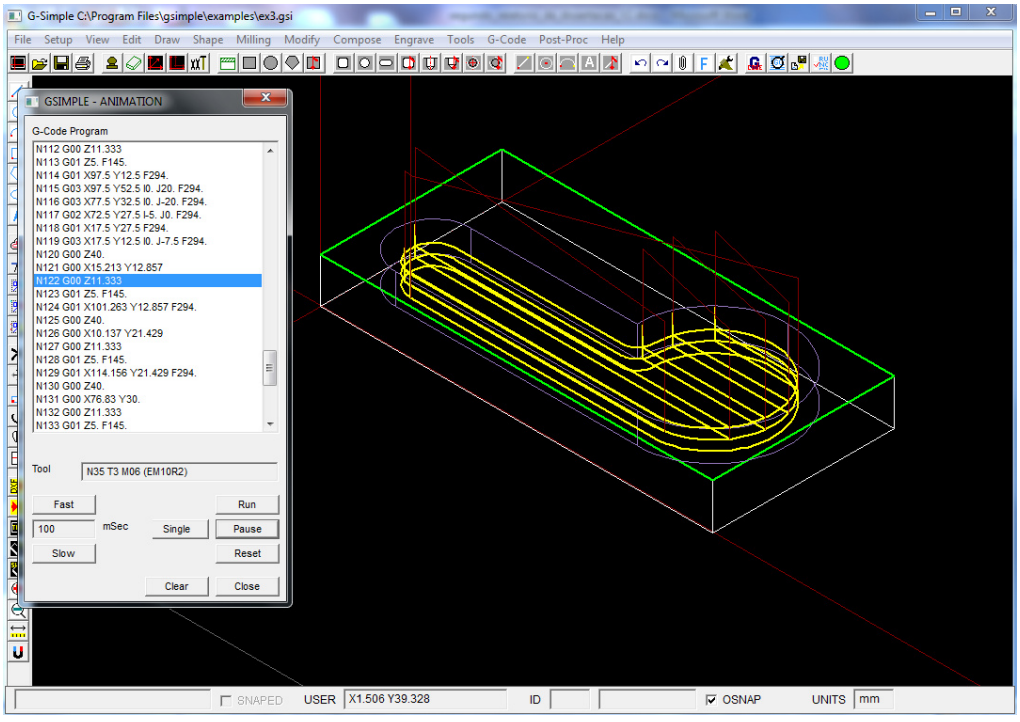


Figura 31 – Visualização do percurso da ferramenta e das etapas do código G em execução

Porém, o pós-processador do *G-SIMPLE* gera o programa NC para um número reduzido de máquinas CNC. Por isso é bastante provável que o utilizador não consiga obter o programa NC pretendido para a sua máquina. No entanto, tal como o *MeshCAM*, é possível a edição do ficheiro de configuração do pós-processador, de modo a que o utilizador possa gerar o seu programa NC com o código G em formato personalizado requerido para ser usado numa determinada máquina.

Contudo para esse efeito o utilizador precisa de executar um programa designado *GSPOST* que é fornecido juntamente com o *G-SIMPLE* e que pode ser encontrado na sua pasta de instalação. Isto é necessário pois o *G-SIMPLE* não permite o uso de um ficheiro de configuração editado do pós-processador no seu interior. Sendo assim o utilizador necessita de executar o programa *GSPOST* utilizando o ficheiro NC gerado pelo *G-SIMPLE* como "input file" e o ficheiro de configuração editado do pós-processado como "rule file", gerando como "output file" o programa NC com o código G no formato pretendido (Figura 32).

O *GSPOST* é um programa independente do software de CAD/CAM *G-SIMPLE*, no sentido em que não necessita que o *G-SIMPLE* esteja aberto para poder ser executado.

Embora o *G-SIMPLE* não possua as mesmas funcionalidades dos softwares de CAM mais dominantes no mercado, este oferece uma solução para maquinagem de 3 eixos completamente gratuita para o utilizador.

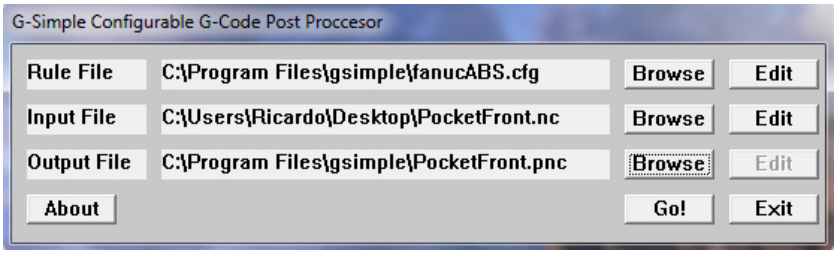


Figura 32 – Utilização do programa *GSPOST* para obtenção de um ficheiro NC com código G em formato personalizado

4 Desenvolvimento de um Sistema Pós-Processador

O presente capítulo apresenta o desenvolvimento de um sistema pós-processador para a utilização de robôs em operações de maquinagem, mais precisamente fresagem. A sua estrutura utiliza os recursos de integração entre sistemas para proporcionar flexibilidade no uso de sistemas robóticos, além da automação do processo de programação de robôs para a execução de trajectórias e adequação de parâmetros de maquinagem.

4.1 Estrutura Geral da Solução Adoptada

A estrutura do pós-processador implementado e sua integração aos restantes sistemas utilizados é ilustrado na Figura 33.

O processo de automatização da programação de trajectórias para maquinagem através de um robô inicia-se num *software* de CAD onde a informação geométrica do modelo gerado é representada analiticamente através de entidades geométricas como pontos, linhas, planos e superfícies num arquivo de dados.

Os arquivos gráficos gerados no sistema CAD são transferidos para os sistemas de CAM através de um formato neutro de transferência de arquivos gráficos como IGES, STEP ou ACIS. O *software* de CAM interpreta as informações deste arquivo e disponibiliza, através da manipulação do mesmo, ferramentas de auxílio à geração de trajectórias e programas de operação de máquinas-ferramenta CNC.

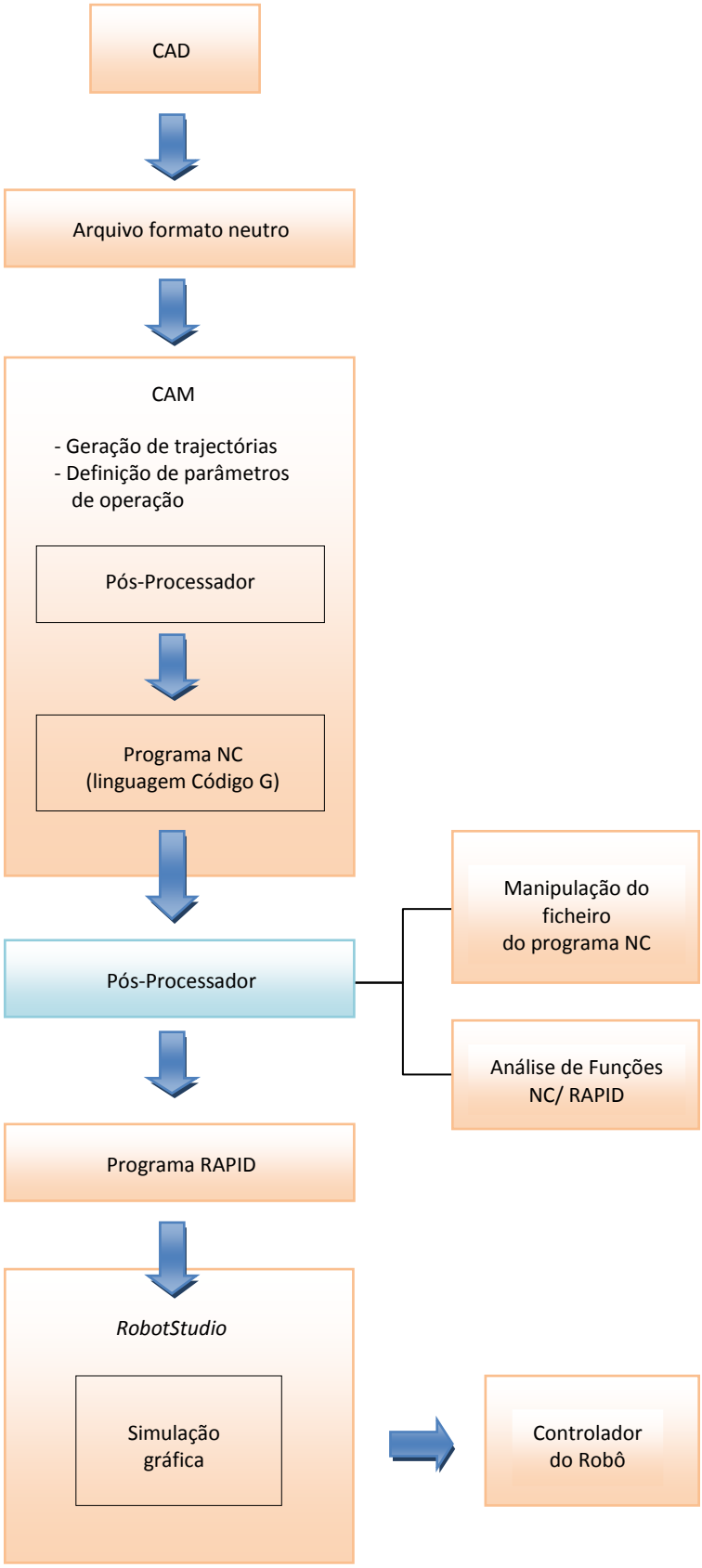


Figura 33 – Esquema ilustrativo da integração do pós-processador implementado com os restantes sistemas

Os parâmetros de operação de maquinagem, como velocidade e profundidade de corte e avanço, são determinados pelo módulo de programação de comando numérico do sistema CAM. As trajectórias a serem percorridas são geradas automaticamente sobre as entidades geométricas seleccionadas através da interface gráfica do *software* de CAM de acordo com as estratégias de maquinagem seleccionadas. O processo de geração das trajectórias e a interpolação dos movimentos lineares e circulares são realizados pelos algoritmos matemáticos presentes no programa de CAM.

Após a criação das trajectórias, o pós-processador do programa de CAM gera o programa de controlo numérico, que possui instruções e comandos para o controlo de máquinas-ferramenta CNC no formato de linguagem ISO habitualmente conhecido como código G, com características específicas requeridas pelas variadas máquinas CNC existentes.

O *software* de CAM utilizado admite a manipulação do arquivo de pós-processamento, permitindo a personalização de um pós-processador qualquer e o desenvolvimento de novos pós-processadores a partir de um padrão que contém a maioria das funções necessárias às operações de maquinagem. Assim, através da manipulação deste pós-processador é possível modificar o formato e o padrão do arquivo de texto do programa NC resultante do pós-processamento. A opção pelo desenvolvimento de um novo pós-processador ao invés da possibilidade de personalização de um já existente, é justificada pelas limitações no processo, principalmente na adaptação da sintaxe da linguagem robótica (RAPID) para o editor de pós-processador.

O pós-processador desenvolvido utiliza como dado de entrada o ficheiro do programa NC gerado pelo CAM. No entanto é necessário que o ficheiro do programa NC seja manipulado primeiro num editor de texto, para que possa ser reconhecido pelo pós-processador. O algoritmo do pós-processador permite a conversão automática de trajectórias, adequação das funções de movimento e parâmetros de operação, obtendo-se assim como resultado a automação do processo de programação de robôs.

O pós-processador desenvolvido consiste em funções e subrotinas elaboradas na linguagem de programação VBA (*Visual Basic for Applications*) que são executadas no interior do Microsoft Excel. O algoritmo implementado identifica funções e parâmetros da operação descritos para o comando numérico de uma máquina-ferramenta e relaciona-os com o sistema robótico.

Após a geração do programa do robô, o mesmo é transferido para o *software* de programação *off-line*, o *RobotStudio*, onde são simuladas as trajectórias a executar pelo robô. Por fim o programa pode ser transferido para o controlador do robô de modo a ser possível testar o programa gerado na célula real.

Para um melhor entendimento da metodologia seguida, será feita uma análise das características das linguagens padrão DIN/ISO 66025 (código G) e RAPID, evidenciando as limitações e equivalências impostas para permitir a tradução entre as diferentes linguagens.

4.2 Linguagem padrão DIN/ISO 66025 (código G)

Um programa na linguagem padrão DIN/ISO 66025 mais conhecido por código G é utilizado pelo controlador de uma máquina CNC. A sua estrutura compreende dois grandes grupos de funções, as funções G e as funções M. As funções G estão associadas ao posicionamento e movimentação da ferramenta, enquanto que as funções M estão ligadas aos parâmetros de controlo da máquina CNC. O código G usa também variáveis reservadas para a definição do tipo de operação, parâmetros auxiliares de operação, coordenada de movimentação, etc.

Algumas das funções e variáveis utilizadas no código G são apresentados na tabela 4 e tabela 5.

Variável	Descrição
N	Número da linha do programa
G, M	Funções
X, Y, Z	Deslocamento nos eixos cartesianos
I, J, K	Coordenadas de centro do arco em relação aos eixos de coordenadas
F	Avanço
T	Selecção da ferramenta
R	Raio de interpolação circular
S	Velocidade de rotação do eixo da ferramenta

Tabela 4 – Exemplo de algumas variáveis utilizadas no código G e sua descrição

Funções G	**Descrição**
G00	Função de posicionamento através de movimento rápido onde a ferramenta se desloca em linha recta até o ponto especificado pelos parâmetros da coordenada.
G01	Movimento de interpolação linear
G02	Movimento de interpolação circular no sentido horário
G03	Movimento de interpolação circular no sentido anti-horário
G17	Selecção de plano de trabalho XY
G18	Selecção de plano de trabalho XZ
G19	Selecção de plano de trabalho YZ
G41	Compensação do raio da ferramenta à esquerda
G42	Compensação do raio da ferramenta à direita
G53	Selecção do sistema de coordenadas máquina
G54 a G59	Selecção do sistema de coordenadas da peça
G70	Selecciona o sistema de unidades inglês (polegada)
G71	Selecciona o sistema de unidades métrico (milímetro)
G90	Define sistema de coordenadas absoluto
G91	Define sistema de coordenadas incremental
Funções M	**Descrição**
M03	Rotação do eixo da ferramenta no sentido horário
M04	Rotação do eixo da ferramenta no sentido anti-horário
M05	Desliga o accionamento do eixo da ferramenta
M06	Troca da ferramenta
M08	Liga o sistema de refrigeração
M09	Desliga o sistema de refrigeração
M30	Fim do programa

Tabela 5 – Exemplo de algumas funções utilizadas no código G e sua descrição

Um aspecto do código G que vale a pena referir é o conceito de funções modais: o código G pode possuir funções modais ou não modais. As funções modais são aquelas que quando definidas permanecem activas em blocos subsequentes do código até que uma nova função que a cancele seja executada. As funções não modais, apenas permanecem activas nas linhas de código onde foram definidas. Convém tomar nota que o pós-processador desenvolvido foi concebido para converter código G que utilize apenas funções não modais.

Exemplo de um programa com funções G00 e G01 modais no código G:

```
G00 Z2.
X38.1 Y3.528
G01 Z-1.F100
X37.394 Y4.586 F1000
X38.1 Y3.528
G00 Z2.
X26.576 Y26.223
G01 Z-1. F100
X26.694 Y25.988 F1000
X26.576 Y26.223
G0 Z2.
```

Exemplo de um programa com funções G00 e G01 não modais no código G:

```
G00 Z2.
G00 X38.1 Y3.528
G01 Z-1. F100
G01 X37.394 Y4.586 F1000
G01 X38.1 Y3.528
G00 Z2.
G00 X26.576 Y26.223
G01 Z-1. F100
G01 X26.694 Y25.988 F1000
G01 X26.576 Y26.223
G00 Z2.
```

De todas as funções presentes no código G as que sem dúvida requerem mais atenção são as funções de movimentação G00, G01, G02 e G03. Estas funções concentram informação à cerca de como é feita a interpolação dos movimentos para a geração das trajectórias da ferramenta.

A sintaxe utilizada para este grupo de funções é apresentada no exemplo descrito a seguir e inclui além da definição da função, as coordenadas cartesianas do ponto de destino em relação ao ponto de referência da peça (zero da peça) e a velocidade de movimentação quando necessário. Para a função G00 a velocidade é previamente definida no controlador CNC, por essa razão não aparece no bloco de instruções do código G. Quanto às coordenadas de destino, estas não necessitam de ser descritas se não forem alteradas em relação à linha anterior. Assim, elimina-se alguns caracteres da linha de instrução, mantendo a quantidade de

informação transmitida. Sendo assim, mantêm-se as coordenadas anteriores para os eixos que não estão a ser descritos no bloco de instruções.

Exemplo de sintaxe para a instrução de movimento linear G01 (Figura 34):

G01 X15 Y25. Z2.5 F500

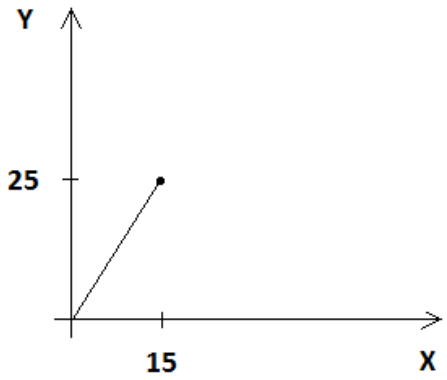


Figura 34 - Visualização de um exemplo de instrução de movimento linear G01

Para as funções de movimento circular a sintaxe é definida pelo sentido de movimentação, as coordenadas do ponto final e do centro do arco descrito, não esquecendo que o ponto inicial para a definição do arco se encontra nas coordenadas do ponto da instrução anterior. No caso de uma circunferência completa, o bloco de instruções apenas define o sentido de rotação e as coordenadas do centro da circunferência descrita, visto os pontos inicial e final coincidirem e estarem definidos na instrução anterior.

Exemplo da sintaxe para a instrução de movimento circular no sentido horário (Figura 35):

G02 X35 Y25 I25 J25

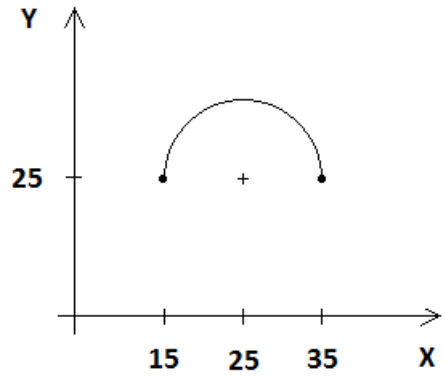


Figura 35 - Visualização de um exemplo de instrução de movimento circular G02

Para uma circunferência completa, descrita no sentido horário (Figura 36):

G02 I25 J25

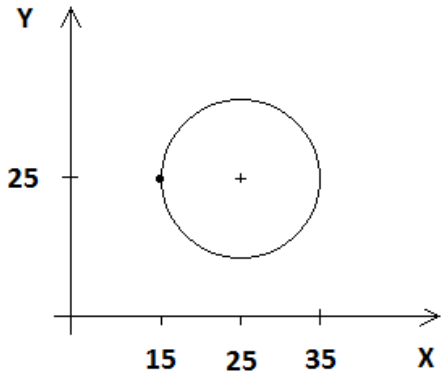


Figura 36 - Visualização de um exemplo de instrução de movimento circular G02 para uma circunferência completa

A sintaxe da instrução de movimento circular também pode ser definida com as coordenadas do ponto final do arco e o respectivo raio (Figura 37).

Exemplo:

G03 X35 Y25 R10

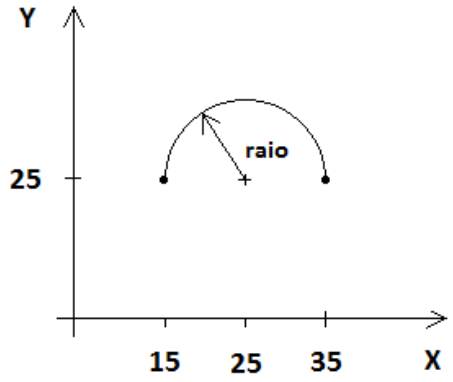


Figura 37 - Visualização de um exemplo de instrução de movimento circular G03

4.3 Linguagem RAPID e o Sistema Robótico

A linguagem de programação de robôs denominada RAPID foi desenvolvida pela *ABB* para utilização nos seus próprios robôs. Esta linguagem está disponível nos vários controladores de robôs da *ABB* (IRC5 incluído) e no *software* de programação *offline RobotStudio*.

RAPID é uma linguagem de programação de alto nível, possuindo funções, procedimentos, rotinas, expressões lógicas e aritméticas, etc.

Na linguagem RAPID uma trajectória é programada como uma sequência de movimentos entre os pontos programados, no entanto estes pontos podem ser uma posição exacta que o robô deve alcançar ou ponto de passagem onde o robô muda de direcção próximo do ponto programado. O parâmetro que mede este desvio à posição exacta a alcançar é designado por zona. É dado em milímetros sendo definido como uma distância radial (Figura 38). Para o caso do ponto programado constituir uma posição exacta que o robô deve alcançar, o parâmetro de zona é definido como *fine*.

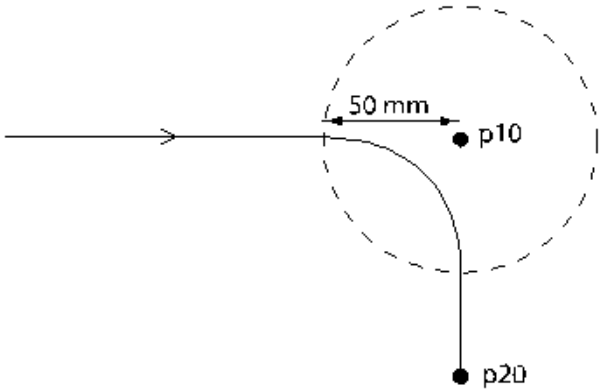


Figura 38 – Identificação do parâmetro de zona, neste caso o parâmetro de zona equivale a z50

Assim, como nas máquinas-ferramenta, a posição do robô e os seus movimentos estão relacionados com um sistema de coordenadas. Num sistema robótico existem dois sistemas de coordenadas bem definidos. O primeiro, denominado sistema de coordenadas base, corresponde a um sistema fixo localizado na base do robô (Figura 39). O segundo é um sistema de coordenadas da ferramenta, por vezes referido como sistema de coordenadas do TCP (*Tool Center Point*). A origem deste referencial está associada ao ponto central da ferramenta (TCP).

Na Figura 39 estão apresentados os diferentes sistemas de coordenadas definidos.

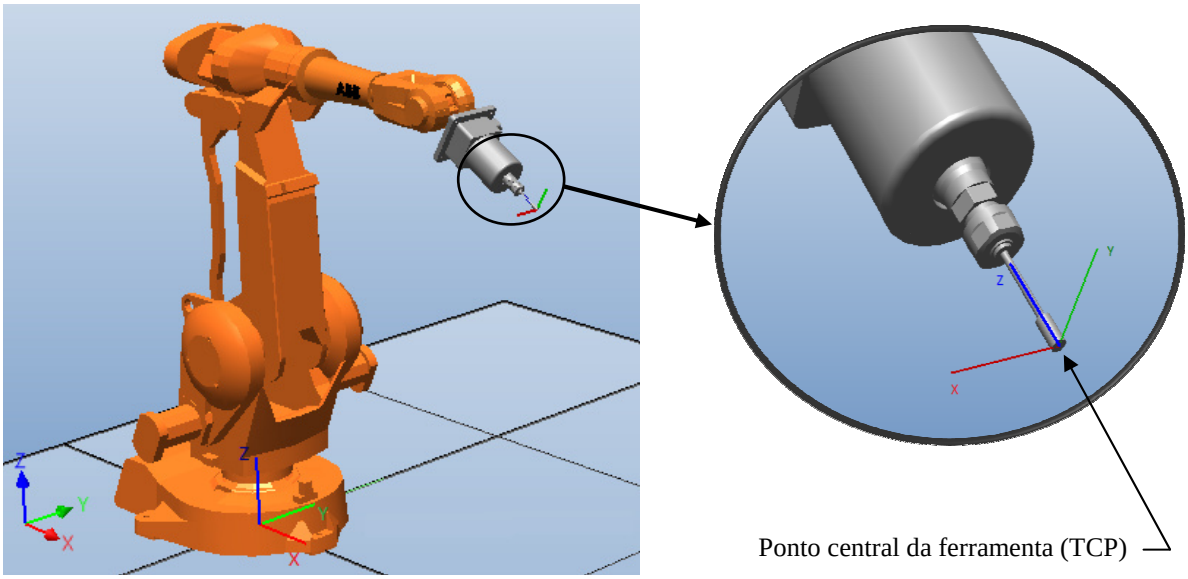


Figura 39 – Ilustração da posição do sistema de coordenadas base (à esquerda) e do sistema de coordenadas do TCP (pormenor à direita)

Quando um programa é executado, o robô movimenta o seu TCP para as posições programadas, descrevendo uma determinada trajectória (linear ou circular) à velocidade que tiver sido programada.

Relativamente às instruções utilizadas para definir os movimentos do robô, a linguagem RAPID disponibiliza três funções base:

- MoveJ - para trajectória livre com movimentação definida pelas juntas do robô de modo a que, quando a posição de destino é atingida, a paragem do movimento dos eixos ocorra em simultâneo;
- MoveL - para um movimento do TCP descrevendo uma trajectória linear;
- MoveC - para um movimento do TCP descrevendo uma trajectória circular.

O parâmetro de velocidade define, em milímetros por segundo, a velocidade de movimentação do TCP.

Um outro conceito importante no *RobotStudio*, e que é usado na linguagem RAPID, é o *workobject*. Um *workobject* representa a localização física de um objecto na estação de trabalho. O *workobject* está pois associado a um sistema de coordenadas (*workobject coordinate system*). Para especificar um *workobject* é necessário utilizar dois referenciais: o *user frame* e o *object frame*, sendo o último dependente do primeiro (estrutura hierárquica) (Figura 40). Quando se programa um robô os pontos estão relacionados com o *object frame* do objecto. Se nenhum outro *workobject* é especificado, os pontos programados estarão relacionados com o *workobject* de defeito, *Wobj0*, que coincide sempre com o referencial do robô (sistema de coordenadas base do robô).

A utilização de *workobjects* permite um fácil ajuste da programação do robô, recorrendo a um offset, se a posição da peça necessita de ser modificada. Assim, os *workobjects* podem ser utilizados para calibrar os programas realizados por um processo *offline*. Se a localização da peça/posto de trabalho relativamente ao robô real não coincide com a localização na programação *off-line*, basta simplesmente ajustar a localização do *workobject*.

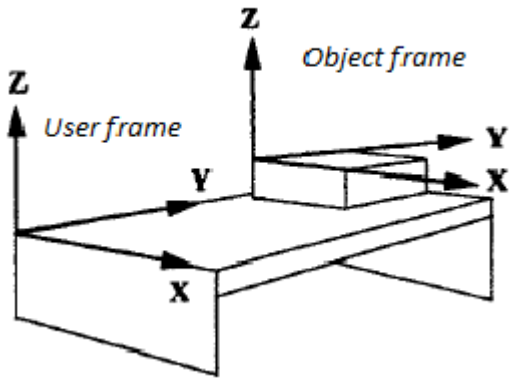


Figura 40 – Sistema de coordenadas *workobject* constituído pelo *user frame* e *object frame*

Um programa de um robô na linguagem RAPID está organizado numa estrutura constituída por módulos. Assim, existem dois tipos de módulos, os módulos de programa e os módulos de sistema. Os primeiros são constituídos por um módulo principal e módulos secundários (Figura 41).

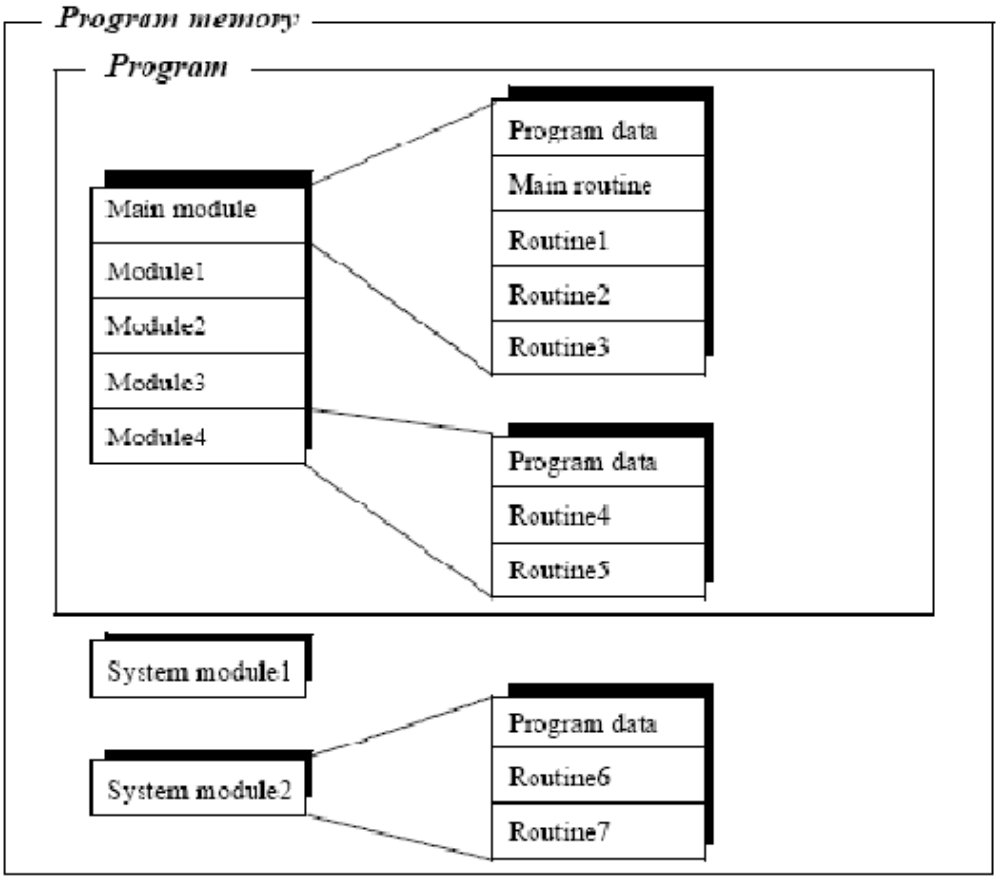


Figura 41 – Estrutura de um programa escrito na linguagem RAPID

Módulos de programa

Cada módulo de programa é constituído por rotinas (*routines*) e dados do programa (*program data*). Nos módulos de programa tem sempre que existir um módulo principal, i.e., aquele que contém a rotina principal (*main routine*) que possui o código de início do programa.

Em pequenas aplicações, a programação pode estar contido em apenas um módulo de programa, enquanto que em aplicações mais complexas, poderá ser necessário ter o módulo principal a referenciar rotinas ou dados contidos num outro ou outros módulos de programa secundários.

Um módulo de programa pode incluir:

- Definição da interface com equipamento externo;
- Rotinas gerais;
- Dados geométricos obtidos por sistemas CAD;
- Dados geométricos obtidos por movimentação directa do robô.

Todos os módulos de programa são removidos quando é apagado o programa da memória do controlador. Tipicamente, os módulos de programa são escritos pelo utilizador.

Módulos de sistema

Os módulos de sistema são utilizados para definir dados comuns e específicos de ferramentas ou de rotinas. Estes dados não ficam incluídos no programa quando este é guardado, mas sim armazenados na memória principal do controlador. Os módulos de sistema dizem normalmente respeito mais a equipamentos do que ao programa em particular e são normalmente escritos pelo fabricante do robô ou integrador da célula robótica.

Dados do programa (*program data*)

Dados do programa dizem respeito a valores e definições utilizados nos módulos de programa e módulos de sistema. Os dados podem ser acedidos por instruções do mesmo módulo ou por instruções de outros módulos (dependendo do tipo de dados).

Rotina

Uma rotina contém um conjunto de instruções, i.e., define as tarefas que o robô deve executar. Uma rotina pode também conter dados. Um tipo particular de rotina é a rotina principal (*main routine*), que define o ponto inicial de execução do programa. Cada programa tem que ter esta rotina, caso contrário, não será possível executar o programa.

Apresenta-se de seguida um exemplo de um programa em linguagem RAPID para descrever o percurso representado na Figura 42.

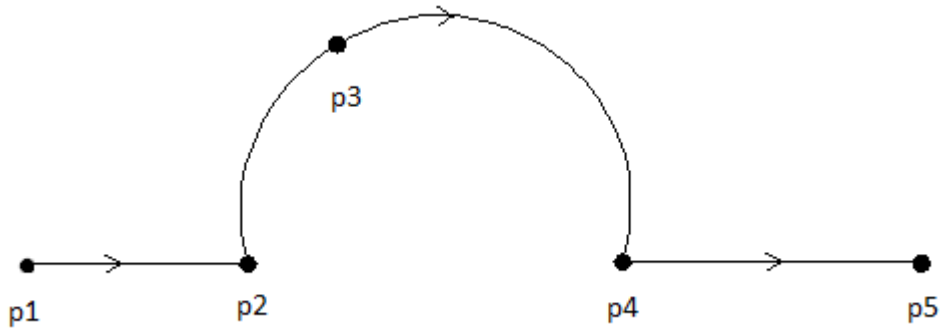


Figura 42 - Pontos programados para descrever um determinado percurso

```
MODULE Module1
        CONST robtarget p1:=[[140,49.9999999999998,30],
        [1,0,0,0],[0,0,0,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
        CONST robtarget p2:=[[185,49.9999999999996,30],
        [1,0,0,0],[0,0,0,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
        CONST robtarget p3:=[[220,84.9999999999997,30],
        [1,0,0,0],[0,0,0,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
        CONST robtarget p4:=[[255,49.9999999999994,30],
        [1,0,0,0],[0,0,0,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
        CONST robtarget p5:=[[300,49.9999999999994,30],
        [1,0,0,0],[0,0,0,0],[9E9,9E9,9E9,9E9,9E9,9E9]];

PROC Path_10()
        MoveJ p1,v100,fine,Fresa\WObj:=Workobject_1;
        MoveL p2,v100,fine,Fresa\WObj:=Workobject_1;
        MoveC p3,p4,v50,fine,Fresa\WObj:=Workobject_1;
        MoveL p5,v100,fine,Fresa\WObj:=Workobject_1;
ENDPROC


ENDMODULE
```

Dados do programa (*program data*)

Sub-rotina

Módulo Secundário

```
MODULE mMain
      PERS tooldata Fresa:=[TRUE,
[[2.22044604925031E-13,7.88860905221012E-28,
346.03],[8.02659911E-17,-0.707106781186548,
0.707106781186548,5.88784672006416E-17]],
[3,[-3.76412843303929E-06,2.66335457553803,
105.594972825955],[1,0,0,0],2,2,2]];
      TASK PERS wobjdata Workobject_1:=[
FALSE,TRUE,"",[[965,150,806],[0.70710678118
,-0,0,-0.70710678118]],[[0,1.1102230E-13,0],
[1,0,0,0]]];

PROC main()
      Path_10;
ENDPROC


ENDMODULE
```

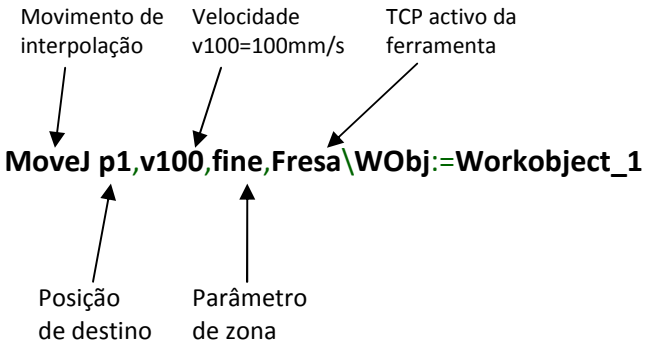
Dados do programa (*program data*)

Rotina Principal

Módulo Primário

Este exemplo em concreto é constituído por um módulo primário e por um módulo secundário. No módulo secundário a informação contida nos dados do programa diz respeito à posição e orientação dos pontos que o robô deverá atingir, bem como a respectiva configuração do braço robótico para alcançar os respectivos pontos. A sub-rotina presente no

módulo secundário contém as instruções de movimentação para a realização do percurso. Cada instrução de movimentação possui a posição de destino que o robô deve atingir, o tipo de interpolação do movimento (MoveJ, MoveL e MoveC), a velocidade de execução do movimento, o parâmetro de zona, o TCP activo da ferramenta e o respectivo *workobject* no qual os pontos programados estão definidos.



O módulo principal possui dados à cerca da ferramenta e informações sobre o TCP e os *workobjects* definidos. A rotina principal como esperado encontra-se no módulo principal e define o ponto inicial de execução do programa, evocando a sub-rotina que contém as instruções de movimentação.

4.4 Conversão Código G para Linguagem RAPID

Em primeiro lugar é importante frisar que os programas de CAM analisados geram programas em código G para 3 eixos. Por essa razão o pós-processador foi elaborado para executar o pós-processamento para 3 eixos apenas.

O pós-processador desenvolvido relaciona as funções do código G com as funções da linguagem RAPID de acordo a sua forma de execução. Algumas funções têm uma correspondência directa entre as linguagens, como as funções de movimento linear G00 e G01, que são interpretadas na linguagem RAPID como instruções MoveJ e MoveL, respectivamente.

As funções do código G, G02 e G03, são utilizados pelas máquinas CNC para gerar instruções de trajectórias circulares. A sintaxe destas instruções define para além da própria

função de movimento circular, as coordenadas do ponto final de movimentação no plano de trabalho e as coordenadas do centro da trajectória circular I, J (em coordenadas absolutas).

A instrução para a trajectória circular na linguagem RAPID diferencia-se não só na sua forma de execução como também na sua sintaxe. A sintaxe desta instrução define, além da própria função de trajectória circular (MoveC), as coordenadas de um ponto intermédio no arco descrito e as coordenadas do ponto final da trajectória. Isto constitui uma particularidade que impede uma conversão directa para código G. Para uma correcta conversão das trajectórias circulares foi usada uma estratégia baseada em noções de álgebra e geometria, para a obtenção do ponto médio de uma trajectória circular, necessária para a correcta utilização da função MoveC na linguagem RAPID.

A estratégia utilizada e que foi implementada no pós-processador desenvolvido é descrita a seguir.

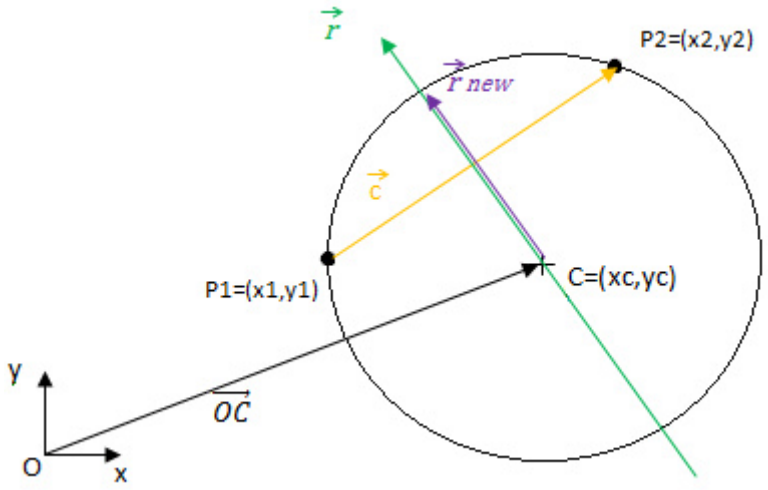


Figura 43 - Arco definido por 2 pontos e centro

Considerando um arco descrito pelos pontos P1 (ponto inicial) de coordenadas (x1, y1), P2 (ponto final) de coordenadas (x2, y2) e um ponto ao centro C de coordenadas (xc, yc) (Figura 43).

Traçando um segmento de recta entre os pontos P1 e P2 obtém-se o vector $\vec{c}$

$\vec{c}$ = (x2-x1 ; y2-y1) (Eq. 1)

De seguida determina-se um vector $\vec{r}$ que é perpendicular ao vector c

$$\vec{r} = (-(y2-y1) ; x2-x1) \quad \text{(Eq. 2)}$$

Determina-se um vector perpendicular ao vector $\vec{c}$ com comprimento igual ao raio

$$\text{raio} = \| (xc-x1 ; yc-y1) \| \quad \text{(Eq. 3)}$$

$$\vec{r}\text{new} = \frac{\vec{r}}{\|\vec{r}\|} \times \text{raio} \quad \text{(Eq. 4)}$$

O ponto médio do arco P1 P2 é o ponto P3 que é obtido da seguinte expressão:

$$\overrightarrow{P3} = \vec{r}_{new} + \overrightarrow{OC} \quad \text{(Eq. 5)}$$

Obtêm-se o vector P3 que possui as coordenadas do ponto P3 relativamente ao eixo de coordenadas do zero da peça.

No entanto existe outra solução para o ponto médio (ponto P4) que é o ponto médio do arco P2 P1 (Figura 44).

$$\overrightarrow{P4} = -\vec{r}_{new} + \overrightarrow{OC} \quad \text{(Eq. 6)}$$

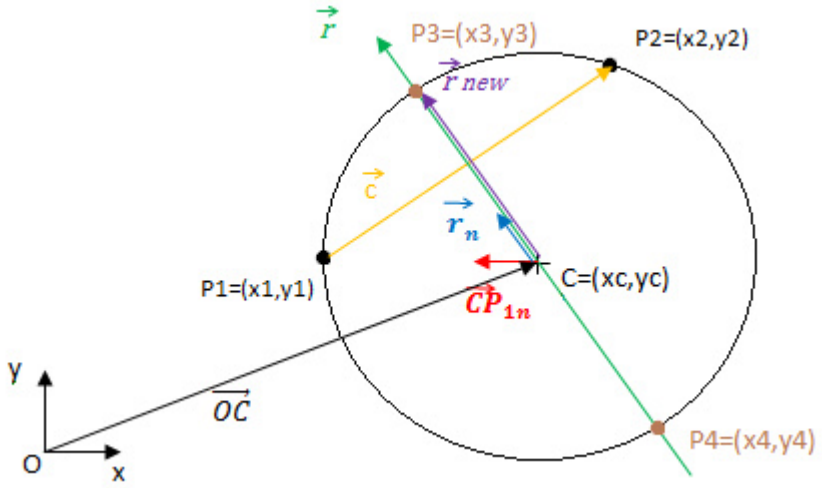


Figura 44 -Relações geométricas usadas para a determinação do ponto intermédio do arco

A próxima fase consiste na determinação do vector $\overrightarrow{CP_1}$ cujas coordenadas serão:

$\overrightarrow{CP_1}$ = (x1-xc ; y1-yc) (Eq. 7)

Dividindo este vector pela sua norma, obtêm-se um vector unitário com direcção e sentido de $\overrightarrow{CP_1}$, o vector $\overrightarrow{CP_{1n}}$:

$$\overrightarrow{CP_{1n}} = \frac{\overrightarrow{CP_1}}{\|\overrightarrow{CP_1}\|} \quad \text{(Eq. 8)}$$

Dividindo o vector $\vec{r}_{new}$ pela sua norma, obtemos um vector unitário com direcção e sentido de $\vec{r}_{new}$, o vector $\vec{r}_n$:

$$\vec{r}_n = \frac{\vec{r}_{new}}{\|\vec{r}_{new}\|} \quad \text{(Eq. 9)}$$

Utilizando estes dois vectores unitários, $\overrightarrow{CP_{1n}}$ e $\vec{r}_{new}$ e fazendo o produto vectorial entre eles $\overrightarrow{CP_{1n}} \times \vec{r}_{new}$ obtém-se um vector com direcção normal a estes dois, cujo sentido é dado pela regra da mão direita. O sentido deste produto vectorial é fundamental para determinar qual dos dois pontos, é efectivamente o ponto médio do arco pretendido.

Sendo assim, estrutura-se o algoritmo de modo correlacionar as instruções de movimento circular G02 e G03 do código G com o sentido do vector resultante do produto vectorial dos dois vectores unitários.

O pós-processador identifica qual a instrução de movimento do código G presente numa linha de código, se essa instrução for a instrução de movimento circular no sentido horário G02, então o algoritmo irá executar o produto vectorial dos vectores unitários, se dessa operação resultar um vector com sentido negativo então o ponto médio do arco será o ponto P3, caso contrário o ponto em questão é o ponto P4.

```
If G02
    If produto_Vectorial < 0
        Then Pmedio = P3
    Else
        Pmedio = P4
End
```

No caso de a instrução presente numa linha do código G ser a instrução de movimento circular no sentido anti-horário G03, o algoritmo presente no pós-processador irá executar o produto vectorial dos vectores unitários, se o resultado da operação for um vector com o sentido positivo então o ponto médio do arco será o ponto P3, caso contrário o ponto em questão é o ponto P4.

```
If G03
    If produto_Vectorial > 0
        Then Pmedio = P3
    Else
        Pmedio = P4
End
```

Para as funções que especificam o uso de coordenadas absolutas ou relativas (G90 ou G91), a especificação é feita no pós-processador do *software* de CAM. O pós-processador desenvolvido foi elaborado para somente operar com coordenadas absolutas.

Outro aspecto importante para uma correcta conversão de um movimento de interpolação circular no código G para a linguagem RAPID, é a identificação do formato utilizado para a definição deste tipo de movimentos. Como já se viu uma instrução de movimento de interpolação circular pode ser definida de diferentes formas:

- Através das coordenadas de um ponto inicial, coordenadas de um ponto final e de um ponto do centro do arco em coordenadas absolutas;
- Coordenadas de um ponto inicial, coordenadas de um ponto final e de um ponto do centro do arco em coordenas relativas;

- Coordenadas de um ponto inicial, coordenadas de um ponto final e definição do raio do arco.

O algoritmo presente no pós-processador desenvolvido identifica no código G instruções de movimento circular no formato que utiliza coordenadas de centro do arco (I, J) em coordenadas absolutas.

Relativamente ao tipo de funções utilizadas no código G, o pós-processador interpreta apenas as funções não modais, por forma a simplificar o algoritmo implementado.

Sendo assim, através do correcto pós-processamento dos programas gerados num CAM, obtém-se um programa em código G que reúne as características necessárias para a sua integração com o pós-processador desenvolvido.

As funções M que estão relacionadas com os parâmetros de controlo da máquina CNC envolvendo funções para troca e accionamento de ferramentas, não são processadas atendendo a que o objectivo principal do presente projecto centra-se no pós-processamento das trajectórias do CAM para o sistema robótico.

O código de programação para a implementação do pós-processador é disponibilizado para consulta no Anexo A.

4.5 Interface de Pós-Processamento

O sistema pós-processador desenvolvido designado por G-RAPID tem como finalidade facilitar o processo de programação de trajectórias para sistemas robóticos, proporcionando a sua adaptação em operações de maquinagem, mais precisamente fresagem. Sendo assim foi desenvolvido uma interface gráfica interactiva entre o utilizador e o processo de fabricação, capaz de se integrar aos sistemas CAD/CAM facilitando o processo de programação e operação de sistemas robóticos.

Quer a interface, quer o algoritmo de conversão foram elaborados na linguagem de programação VBA (*Visual Basic for Applications*). O VBA permite a criação de macros, e está integrado em todos os produtos da família de produtos Microsoft Office como: Word, Excel, Access, Outlook, PowerPoint e FrontPage. Sendo assim foi escolhido o Microsoft Excel como aplicação e executada toda a programação no seu interior recorrendo ao módulo

de programação VBA. Como o nome sugere o VBA é muito semelhante à linguagem de programação em Visual Basic e pode ser usado para controlar a totalidade dos aspectos da aplicação anfitriã, como também pode controlar uma aplicação a partir de outra (por exemplo abrir e escrever num editor de texto como o Notepad através de dados no Excel). No entanto para compilar um programa em VBA tal só é possível dentro da aplicação anfitriã (o Microsoft Excel neste caso).

Para iniciar a aplicação G-RAPID, o utilizador apenas necessita de abrir o ficheiro G-RAPID.xlsm que é um ficheiro elaborado na folha de cálculo Excel. Ao abrir o ficheiro o utilizador necessita de dar a permissão para que as macros presentes na folha de cálculo possam ser executadas. Após dada a permissão é executado automaticamente um formulário (*Form*) que constitui a interface gráfica do G-RAPID.

A interface gráfica desenvolvida para o pós-processador G-RAPID é apresentada na Figura 45. Esta é essencialmente composta por caixas de texto (*text box*), caixas de combinação (*combo box*) e botões de comando que executam macros.

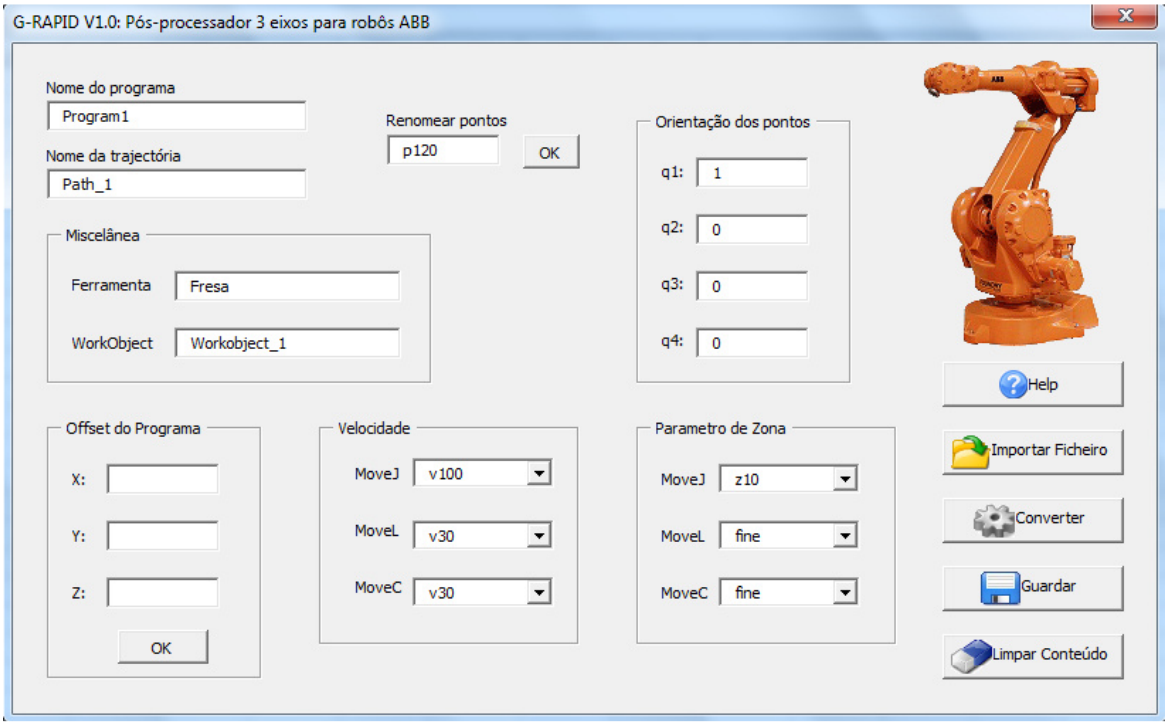


Figura 45 – Janela de interface do pós-processador G-RAPID

Para a correcta utilização do pós-processador, o utilizador deve primeiro certificar-se de que a folha de cálculo Excel está sem qualquer conteúdo. O botão “Limpar Conteúdo”

executa uma macro que deixa todas as células da folha de cálculo em branco. Isto é necessário para que o processo de conversão dos programas importados no G-RAPID se efectue sem erros.

A importação do ficheiro em código G para a folha do Excel é feita através do botão de comando "Importar Ficheiro" que abre uma janela de diálogo permitindo ao utilizador procurar a localização do ficheiro no seu computador pessoal e importá-lo (Figura 46).

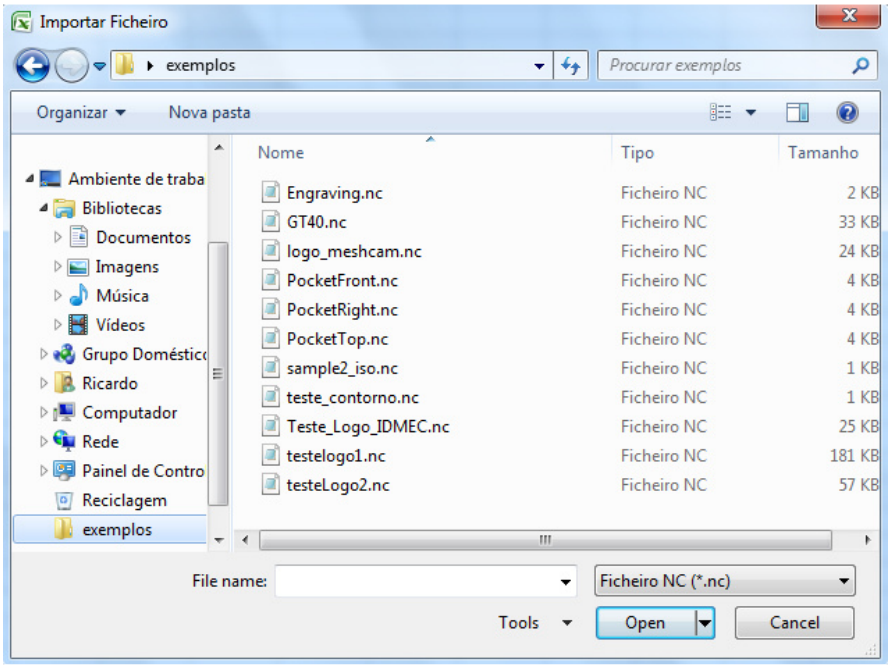


Figura 46 – Janela de diálogo do G-RAPID para importação do ficheiro NC (código G)

Ao importar o ficheiro de código G, determinadas células da folha cálculo são automaticamente preenchidas. Pressionando o botão "Converter", é iniciado o processo de conversão do programa para a linguagem RAPID. O tempo de execução do processo vai depender da extensão do programa importado.

Após a conversão obtém-se o programa RAPID pós-processado na folha de Excel. No entanto o nome do programa gerado, a ferramenta e o *workobject* a utilizar, são definidos por defeito. Através de caixas de texto na interface, o utilizador modifica o programa RAPID gerado podendo atribuir um nome a este e à trajectória criada.

Relativamente à definição da ferramenta a utilizar e ao *workobject*, podem ser atribuídos nomes de ferramentas e *workobject* já existentes no *software* de programação *off-line*. Assim o programa RAPID obtido fica associado a ferramentas e *workobjects* já criados.

É também possível a atribuição de uma orientação do conjunto de pontos do programa RAPID relativamente ao *workobject* definido. Se este campo não for preenchido na janela de interface do G-RAPID, todos os pontos irão possuir a orientação [1, 0, 0, 0], isto é vão ter a mesma orientação que o sistema de eixos coordenados do *workobject*. Ainda relativamente aos pontos que constituem a trajectória do programa gerado, é possível renomear estes de modo a evitar que surjam erros de ambiguidade no *software* de programação do *RobotStudio*, pois por defeito para cada programa convertido, o G-RAPID atribui a mesma numeração aos pontos da trajectória iniciando em "p1" e incrementando sucessivamente até ao último ponto da trajectória. Por essa razão é boa prática renomear os pontos sempre que se queira importar mais do que um programa para o *RobotStudio*, de modo a evitar que surjam pontos definidos com o mesmo nome em programas diferentes, causando os designados erros de ambiguidade.

Em relação aos parâmetros de velocidade e de zona, estes são definidos através de caixas de combinação para cada um dos diferentes tipos de movimento de interpolação.

No botão de comando "Guardar" o utilizador pode guardar o programa RAPID pós-processado. Uma janela de diálogo é aberta para nomear o ficheiro que se deseja gravar e seleccionar o directório para a sua armazenagem. É também possível escolher qual a extensão a dar ao ficheiro que se pretende guardar. O utilizador pode gravar como ficheiro de texto (*.txt) ou como ficheiro de módulo do programa RAPID (*.mod) (Figura 47) de modo a que possa ser importado para o *RobotStudio*.

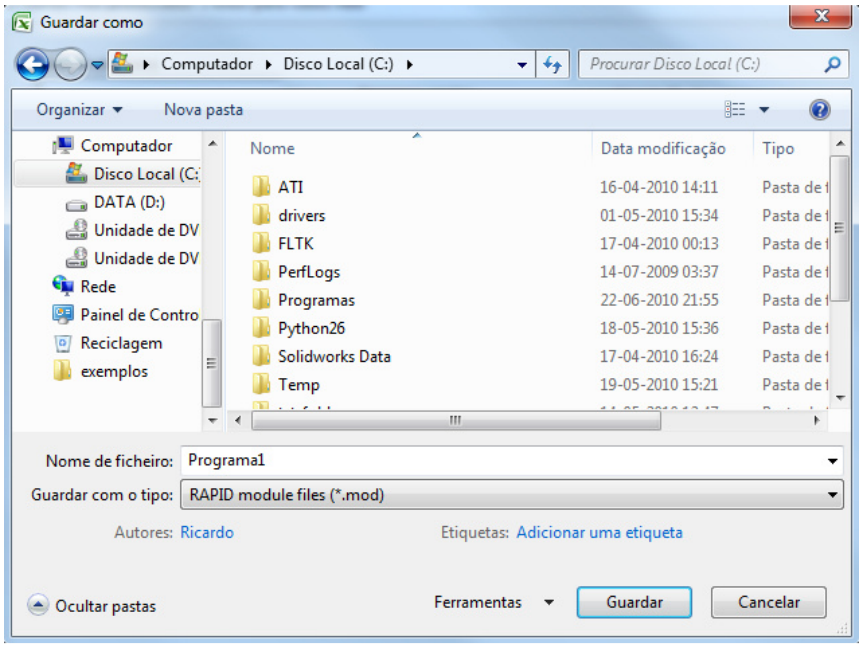


Figura 47 – Janela de diálogo do G-RAPID que permite guardar o ficheiro que contém o programa RAPID pós-processado

O G-RAPID possui ainda um tutorial que pode ser acedido através do botão “Help”, que abre um ficheiro de texto com informação sobre o correcto procedimento para a execução do pós-processamento.

5 Implementação da Solução na Célula Real

Com a finalidade de comprovar a viabilidade operacional do processo de conversão de trajectórias e adequação do sistema robótico a operações de maquinagem usando o sistema pós-processador G-RAPID, foram executados testes práticos usando a célula robótica disponível no Laboratório de Robótica do Departamento de Engenharia Mecânica da Faculdade de Engenharia da Universidade do Porto.

O presente capítulo irá abordar os ensaios efectuados na célula real que contém o robô antropomórfico ABB IRB 2400 de seis eixos.

5.1 Elaboração de Desenhos usando o Robô ABB

Numa primeira abordagem ao processo de conversão, geração do programa na linguagem do robô e sua implementação, decidiu-se desenvolver uma solução para a criação de desenhos, partindo de um ficheiro de imagem, através do uso do robô antropomórfico da ABB. O procedimento para a execução da solução envolve os seguintes passos:

- Utilização de um ficheiro de imagem (*.jpg, *.bmp, *.gif ,*.png, etc);
- Edição do ficheiro de imagem através de um *software* de edição gráfica (Photoshop, CorelDraw, etc.);
- Importação do ficheiro de imagem editado para um *software* de vectorização (WinTopo) para obtenção de um ficheiro de desenho vectorial em extensão dxf;

- Importação do ficheiro dxf para o *software* de CAD/CAM *G-SIMPLE* e geração do programa em código G;
- Pós-processamento do programa em código G para a Linguagem RAPID através do uso do pós-processador G-RAPID;
- Abertura do programa criado na linguagem RAPID com o *RobotStudio* e execução da simulação das trajectórias associadas;
- Transferência do programa para o controlador do robô e execução do desenho com a célula robótica.

Sendo assim foi utilizado um ficheiro de imagem em formato jpg (Figura 48) para a elaboração do desenho.

Figura 48 – Ficheiro de imagem em formato jpeg escolhido para a elaboração do desenho (http://campaignme.wordpress.com/2009/06/)

Para possibilitar a importação da informação contida no ficheiro de imagem para um *software* de CAD/CAM foi necessário recorrer a um *software* de vectorização capaz de transformar o ficheiro de imagem num ficheiro de desenho vectorial em formato dxf, passível de ser transferido para um *software* de CAD/CAM como o *G-SIMPLE*. O *software* de vectorização utilizado foi o WinTopo.

Após as primeiras tentativas de vectorização do ficheiro de imagem escolhido, depressa se verificou que o desenho vectorial produzido continha várias descontinuidades e alguns contornos que eram ignorados devido à existência de vários pixéis de cor escura nas

proximidades. O resultado era um desenho vectorial pobre em detalhe que não era identificável com a imagem original.

A edição da imagem original por via de um *software* de edição gráfica (Photoshop, CorelDraw etc), revela-se como uma solução para ultrapassar este problema. Com o uso destes *softwares* de edição é possível ao utilizador obter um ficheiro da imagem original em *Line Art*, isto é, um estilo de imagem essencialmente monocromático que dá ênfase à forma e aos contornos presentes na imagem sobre as cores, sombras e texturas. Atendendo ao facto de já se possuir uma imagem em *Line Art* da respectiva imagem original (Figura 49), este ficheiro foi processado pelo *software* WinTopo de modo a obter um ficheiro de desenho vectorial (Figura 50).

Figura 49 – Imagem original no estilo *Line Art* (http://101coloringpages.com/b/barack-obama-coloring-pages/)

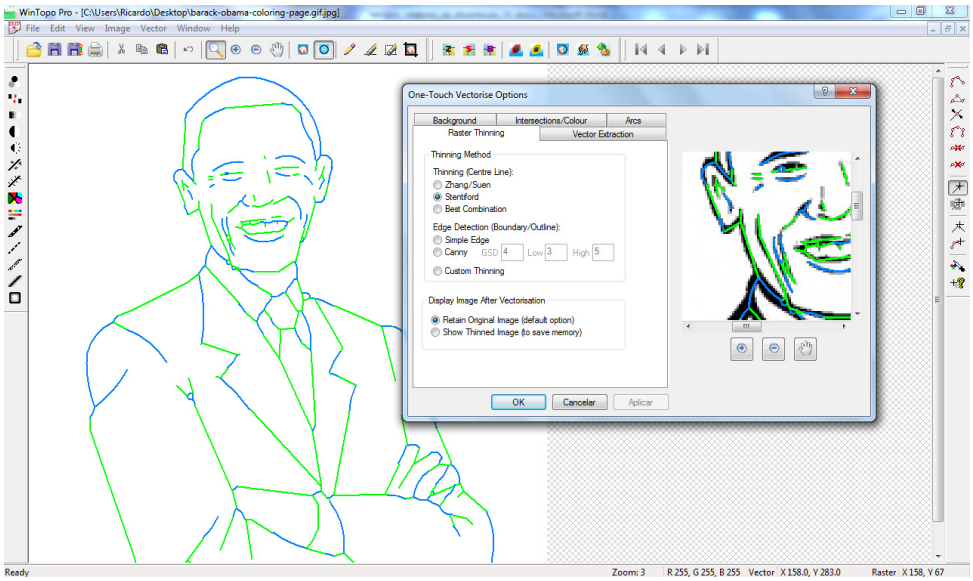


Figura 50 – Obtenção de um desenho vectorial da imagem em *Line Art* no *software* WinTopo

Após a criação do ficheiro de desenho vectorial, este foi guardado no formato dxf e foi posteriormente importado para o software de CAD/CAM *G-SIMPLE*, onde foram criadas as trajectórias e o programa correspondente em código G (ver Figura 51).

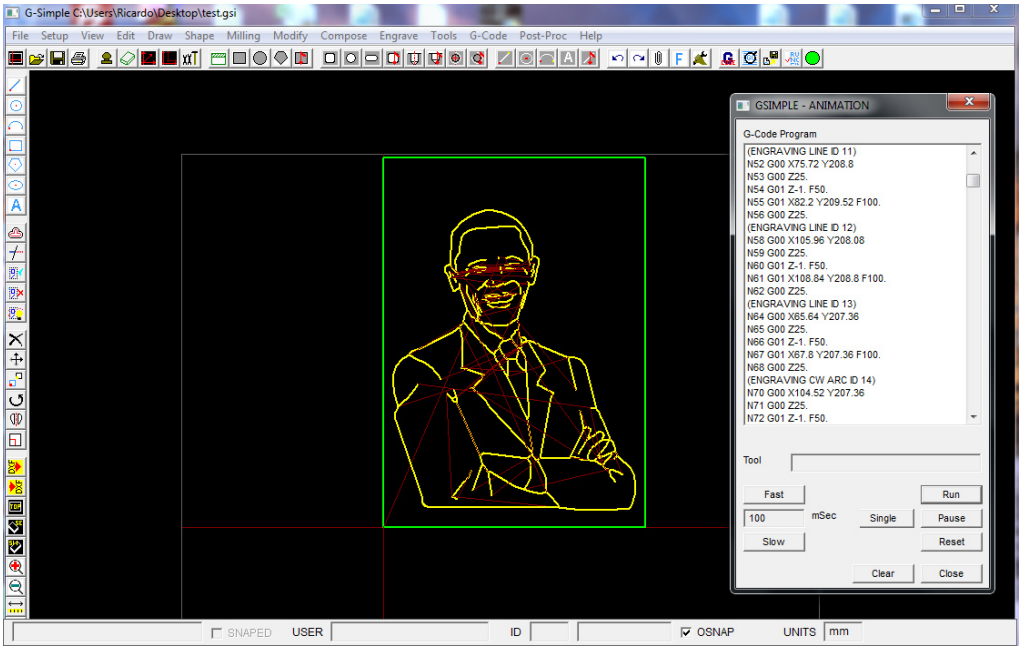


Figura 51 – Geração das trajectórias necessárias no *G-SIMPLE* e do programa em código G

No entanto, o *G-SIMPLE* não gera o código G num formato que possa ser directamente importado pelo G-RAPID. Foi necessário pós-processar o código G usando o programa *GSPOST* que é fornecido com o *G-SIMPLE* (ver capítulo 3.4).

Assim procedeu-se ao pós-processamento do programa de código G, usando o pós-processador G-RAPID. O programa gerado na linguagem RAPID foi importado para o *software* de programação *off-line* e simulação, o *RobotStudio*, onde foi feita uma sincronização com o controlador virtual e execução da simulação das trajectórias na célula virtual para a elaboração do desenho (Figura 52).

A simulação executada permitiu verificar que o robô cumpre, em ambiente virtual, as trajectórias que foram programadas, não tendo sido verificado qualquer erro durante a execução do programa RAPID.

O próximo passo consistiu na transferência do programa para o controlador IRC5 do robô ABB IRB 2400 de modo a se executar na célula real a operação para a realização do desenho.

No entanto, a posição do referencial do TCP da ferramenta e do referencial do *workobject* na simulação não coincide com a sua posição na célula real, deste modo foi necessário redefinir a posição destes na célula real, antes de se passar à execução do programa.

Para possibilitar a elaboração da operação adaptou-se uma garra pneumática no punho do robô, que segura uma caneta. A execução do desenho foi feita sobre uma folha A4.

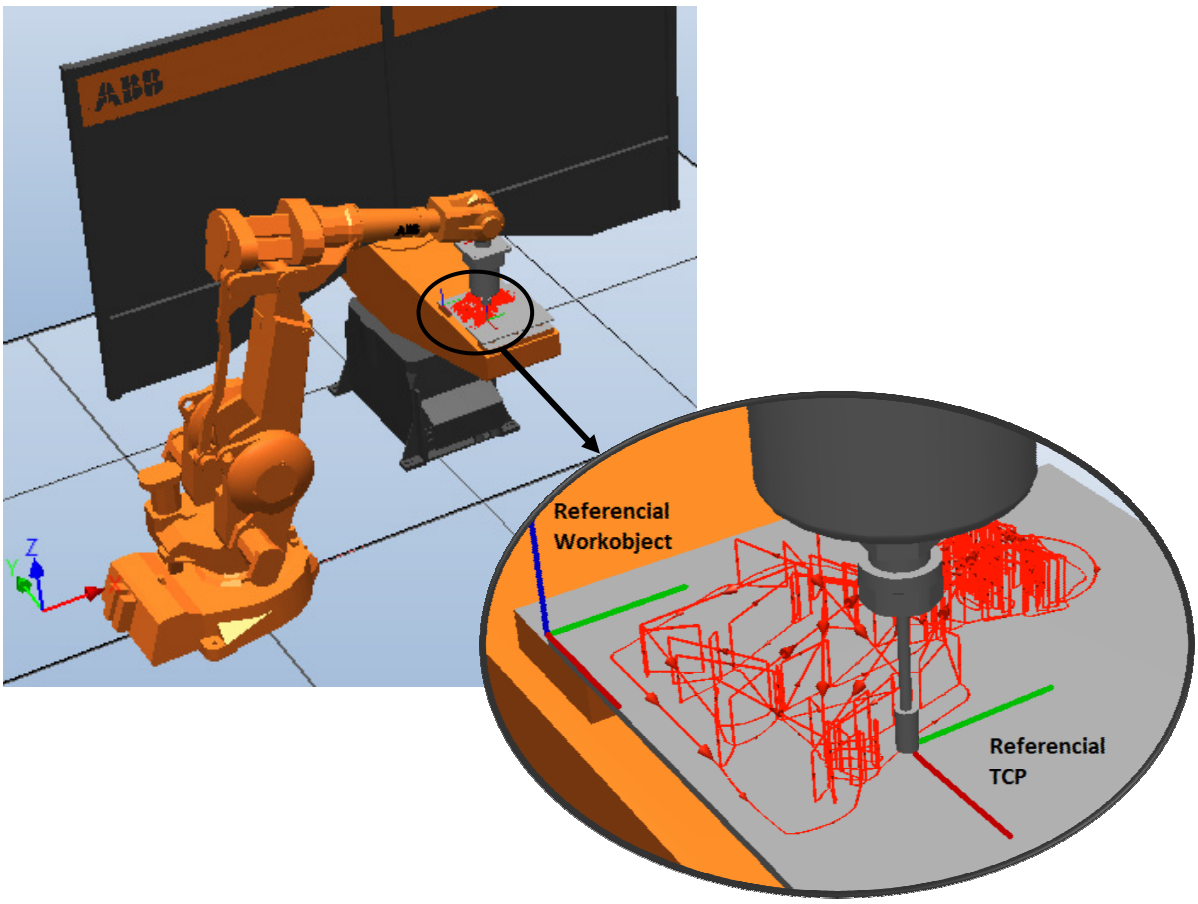


Figura 52 – Simulação das trajectórias na célula virtual do *RobotStudio*

Os primeiros ensaios foram efectuados com um marcador de traço grosso, estando a folha posicionada em cima de um material esponjoso de forma a limitar as forças de contacto aquando do contacto do marcador com a folha de desenho (Figura 53).

Foram também realizados testes experimentais utilizando uma esferográfica de traço fino a qual foi fixada na garra pneumática. Tal como no caso do marcador, sentiu-se necessidade de limitar as forças de contacto presentes, tendo sido introduzida uma mola no interior da esferográfica para esse efeito.

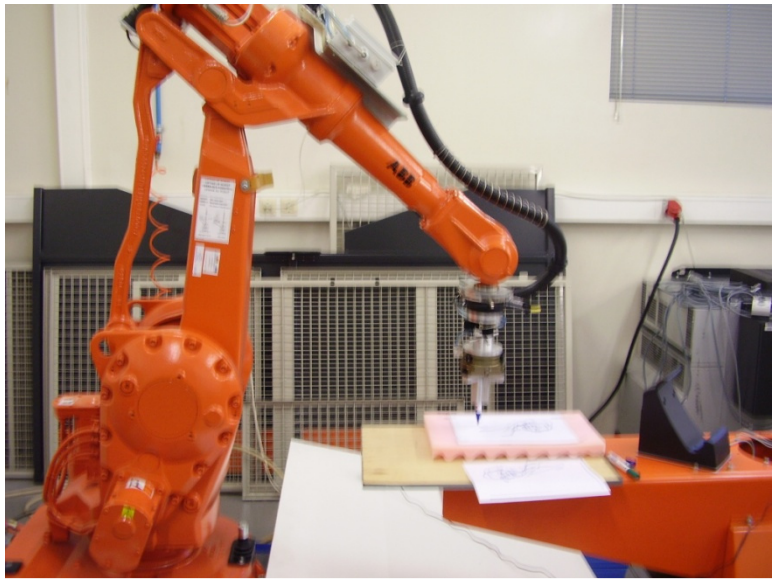


Figura 53 – Foto da célula real durante a elaboração do desenho utilizando um marcador grosso

Na Figura 54 podem ser vistas as trajectórias executadas pelo robô para a criação do desenho, utilizando a esferográfica de traço fino, fixa na garra pneumática e movimentada pelo braço robótico.

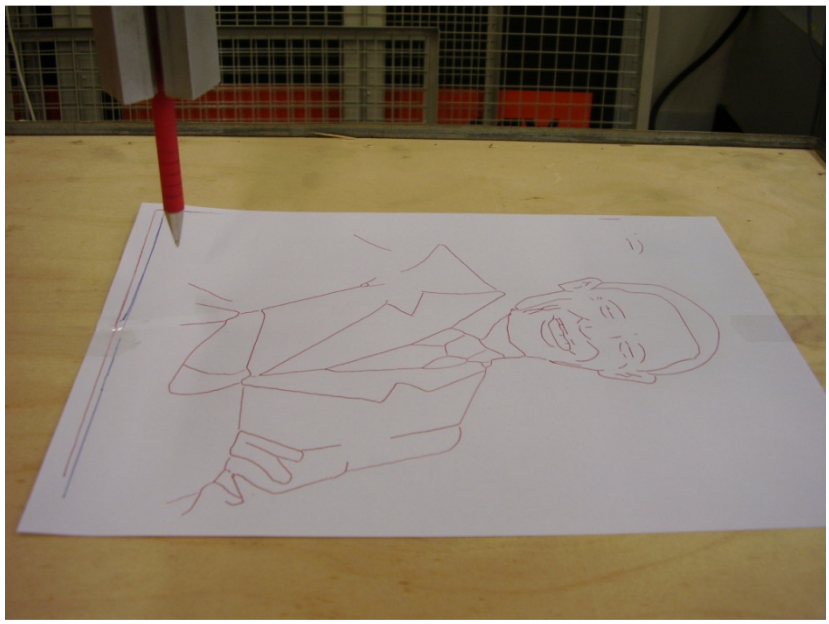


Figura 54 - Execução das trajectórias do robô utilizando a esferográfica de traço fino

Os resultados dos ensaios experimentais para a elaboração do desenho com o robô industrial permitiram avaliar a viabilidade do processo de conversão das trajectórias no pós-procesador G-RAPID.

Na Figura 55 é apresentado o desenho vectorial no *software* de vectorização e o desenho final elaborado com o robô antropomórfico na célula real.

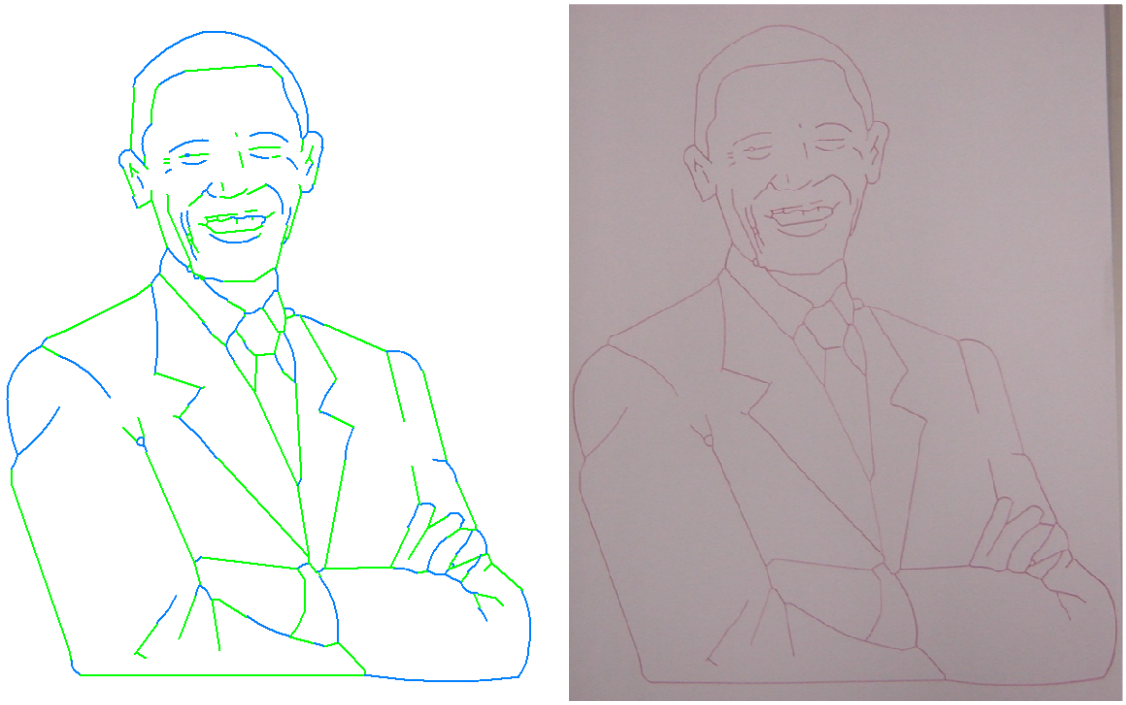


Figura 55 – Visualização do desenho vectorial (à esquerda) e do desenho final efectuado pelo robô na célula real (à direita)

Os resultados obtidos são bastante satisfatórios, sendo que o sistema robótico conseguiu descrever todos os contornos do desenho vectorial, o que significa que as trajectórias programadas no *software* de CAM foram fielmente reproduzidas, comprovando assim a viabilidade do pós-processador desenvolvido.

5.2 Maquinagem de Protótipos com Ferramenta de Corte Rotativa

Nos testes experimentais, para a geração de protótipos, foi utilizado um material de baixa resistência mecânica devido às limitações de rigidez impostas pelo braço robótico. O material escolhido incidiu em espumas de poliuretano (Figura 56) que devido às suas propriedades mecânicas possibilitam a maquinagem com bom acabamento superficial e a geração de baixas forças de corte, não superando as limitações de rigidez do sistema robótico.

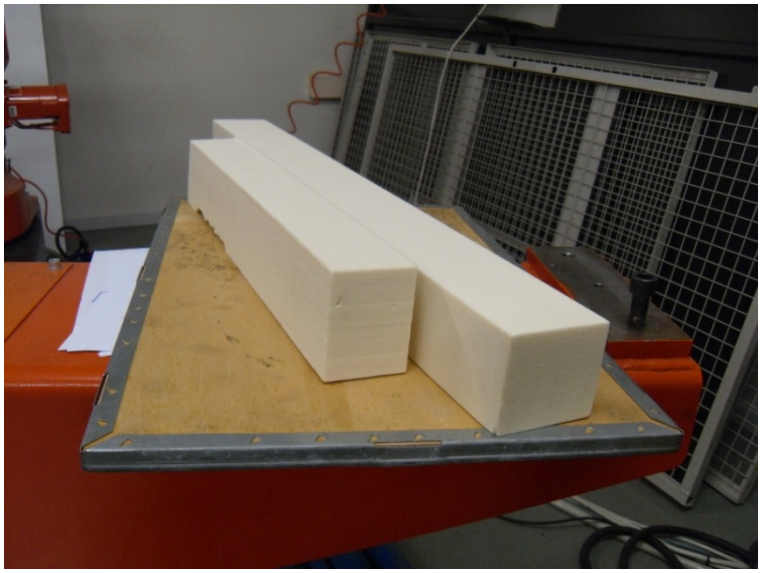


Figura 56 – Espumas de poliuretano usadas na maquinagem dos protótipos

A ferramenta de corte e o respectivo accionamento não foram objecto de estudo neste trabalho. No entanto, a finalidade de se testar a aplicabilidade da solução proposta, levou à utilização dos seguintes recursos disponíveis (Figura 57):

- Suporte da ferramenta – É adaptável no punho do robô antropomórfico da *ABB* por meio de uma interface de fixação;
- Ferramenta – Possui accionamento pneumático à pressão disponível na rede (de 6 bar aproximadamente), com rotação máxima de 22000 r.p.m;
- Ferramentas de corte:

- Fresa de ponta esférica de 4,8mm de diâmetro (Figura 58);
- Fresa cilíndrica de 7,8mm de diâmetro (Figura 58).

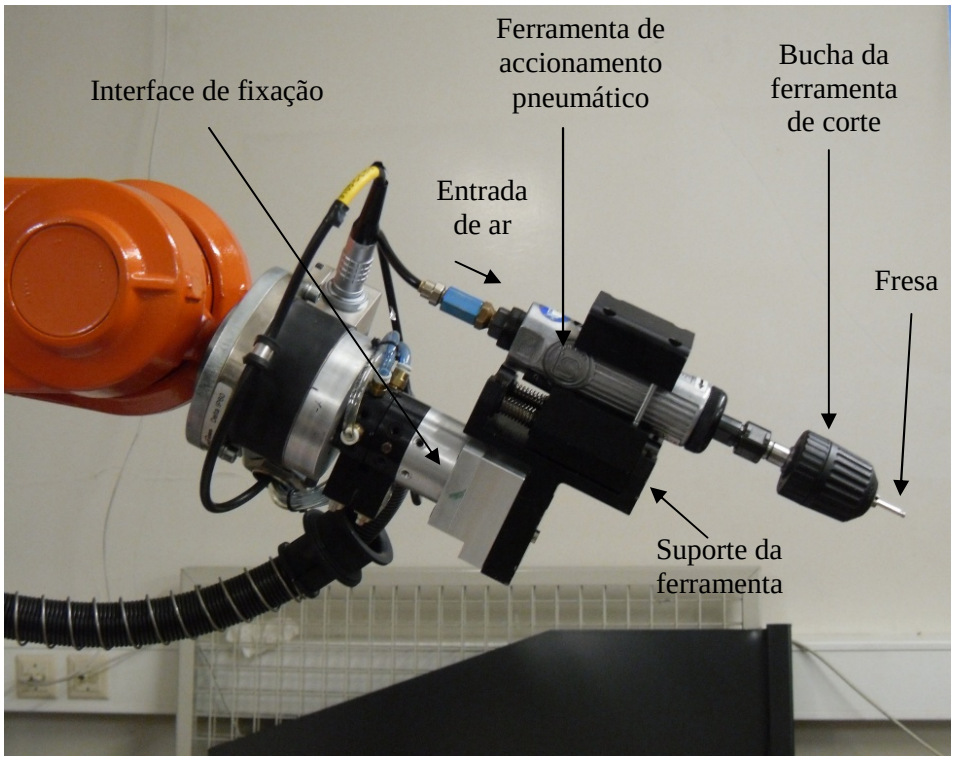


Figura 57 – Conjunto interface de fixação, suporte e ferramenta

Figura 58 – Ferramentas de corte utilizadas: Fresa de ponta esférica (à esquerda) e Fresa cilíndrica (à direita)

Para possibilitar o processo de maquinagem com o robô foi improvisado um sistema de fixação das espumas de poliuretano, constituído por grampos de fixação que mantêm a espuma a maquinar imóvel numa mesa de apoio (Figura 59).

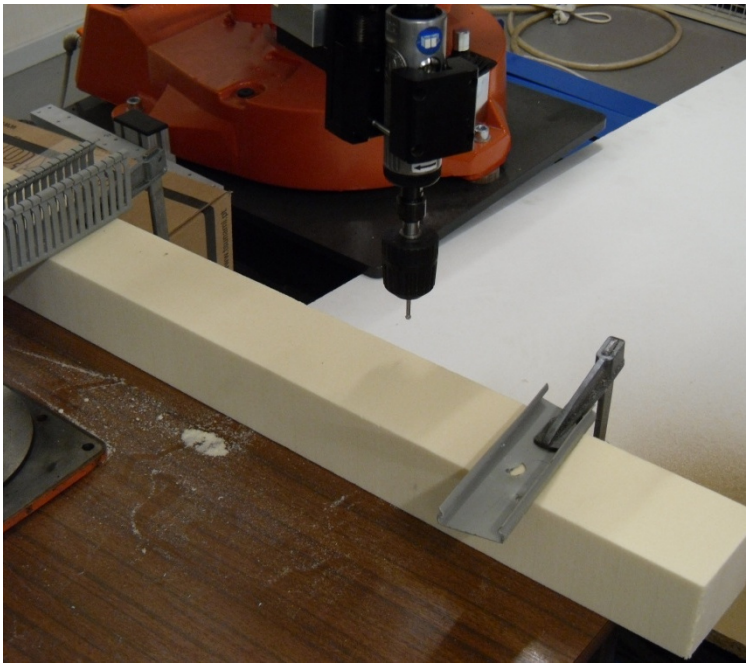


Figura 59 – Sistema de fixação improvisado

5.2.1 Procedimento

O procedimento experimental adoptado para a realização dos ensaios de maquinagem foi o mesmo para todos os ensaios e segue o esquema apresentado na Figura 60.

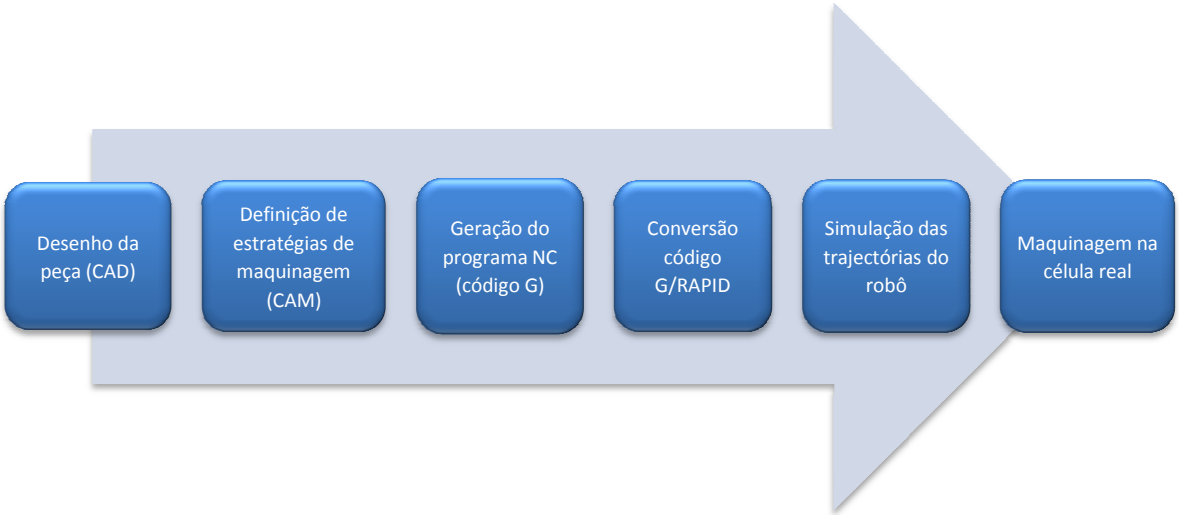


Figura 60 – Procedimento experimental adoptado

5.2.2 Parâmetros de Operação

Devido à inexistência de referências para os parâmetros de corte relativos a operações de maquinagem com a utilização de sistemas robóticos, procurou-se compatibilizar a rigidez do sistema utilizado, com o baixo binário da ferramenta utilizada e as propriedades do material a ser maquinado. Foi então realizado um programa no *RobotStudio* para a execução de testes práticos, tendo em vista a adequação dos parâmetros de operação, mais concretamente, das velocidades de avanço, a implementar no processo de maquinagem.

Os testes efectuados foram realizados com duas ferramentas de corte distintas: uma fresa cilíndrica de 7,8mm de diâmetro e uma fresa esférica de 4,8mm de diâmetro. Usaram-se diferentes velocidades de avanço, executando cortes lineares na superfície do material. As velocidades de avanço testadas foram de 30, 50, 80 e 100mm/s. A ferramenta pneumática foi operada a 6 bar. Os resultados podem ser vistos na Figura 61.

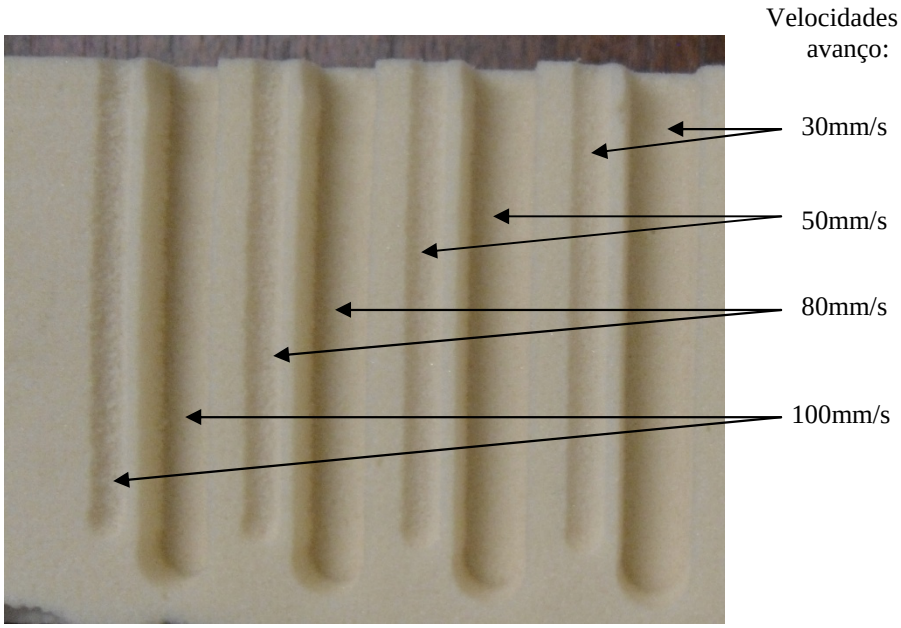


Figura 61 – Visualização da qualidade superficial das cavidades efectuadas com as duas fresas para velocidades de avanço diferentes

Após inspecção visual das cavidades deixadas por remoção de material, verificou-se que a melhor qualidade superficial ocorreu na operação com a velocidade mais baixa de 30mm/s. Sendo assim decidiu-se adoptar esta velocidade como velocidade de avanço durante a operação de corte para os restantes ensaios desenvolvidos.

5.2.3 Realização de Ensaios de Maquinagem

A seguir são apresentados os ensaios laboratoriais realizados, para a obtenção de protótipos seguindo o procedimento ilustrado na Figura 60.

Ensaio 1: Elaboração de uma cavidade esférica

A primeira etapa no processo de geração do programa da peça para maquinagem com a utilização do robô corresponde à modelação desta num *software* de CAD. O modelo foi então criado no *SolidWorks* (Figura 62).

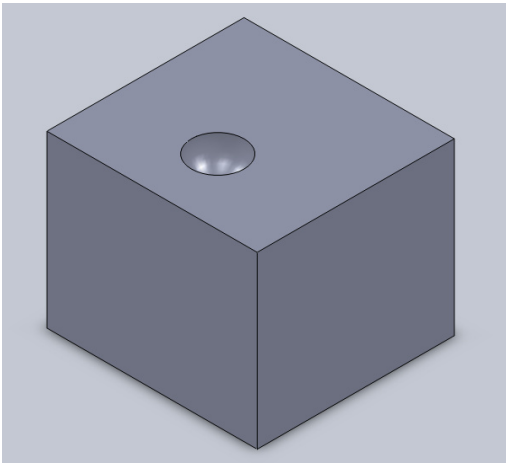


Figura 62 - Cavidade esférica - Modelação em CAD no *SolidWorks*

Após a criação do modelo, este foi transferido para o *software* de CAM 3D o *MeshCAM*, através da conversão do ficheiro do modelo de CAD em ficheiro de formato neutro STL para a geração das trajectórias e estratégias de operação.

Para a fabricação da cavidade esférica foi realizada uma operação de desbaste e uma operação de acabamento. Devido às reduzidas dimensões da cavidade esférica (diâmetro=20mm e profundidade=10mm), decidiu-se usar a mesma ferramenta de corte: uma fresa de ponta esférica de 4,8mm de diâmetro, para a operação de desbaste e acabamento.

Os percursos da ferramenta criados para a elaboração do processo de maquinagem podem ser visualizados na Figura 63, em que as trajectórias para o ciclo de desbaste estão representadas a verde, enquanto que a amarelo se identificam as trajectórias do ciclo de acabamento.

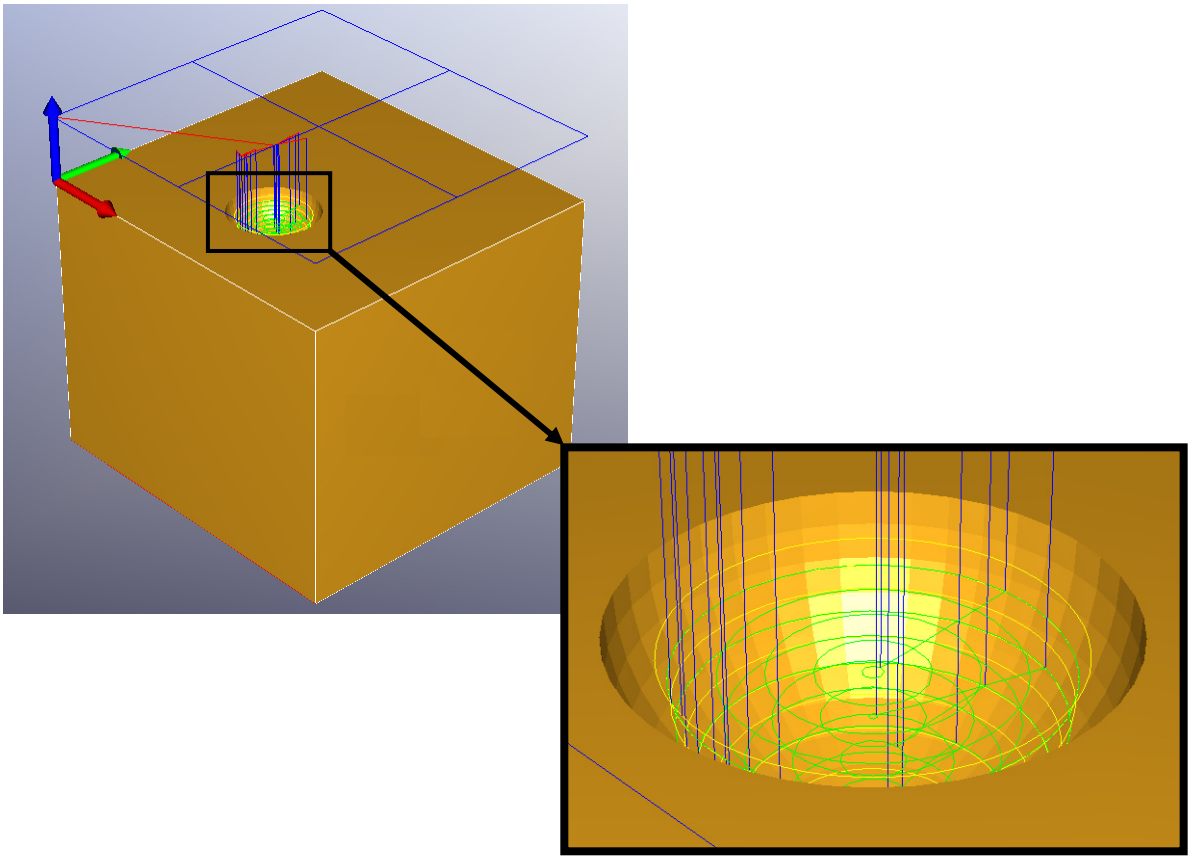


Figura 63 – Trajectórias da ferramenta para o ciclo de desbaste (a verde) e para o ciclo de acabamento (a amarelo) no *MeshCAM*.

Definidos todos os parâmetros necessários no CAM, foi gerado o programa em código G, através da personalização do pós-processador do CAM, que foi modificado de modo a permitir gerar o código G, já com os formatos necessários para a importação directa do programa para o pós-processador G-RAPID (Figura 64).

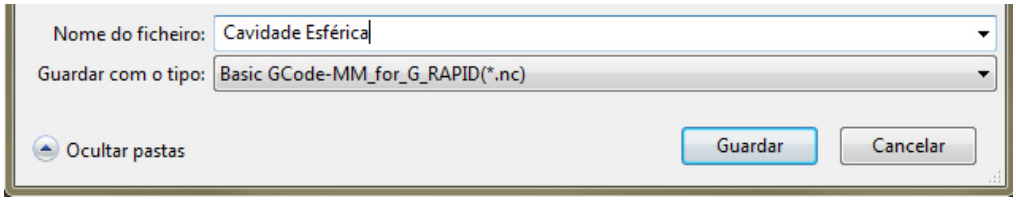


Figura 64 – Geração do programa em código G para importação directa com o G-RAPID

O ficheiro criado é utilizado como dado de entrada no G-RAPID, onde se executa a conversão em instruções de operação para o robô na linguagem RAPID.

A etapa seguinte consistiu na importação do programa na linguagem RAPID para o *RobotStudio*, onde foi feita a sincronização com o controlador virtual, com o objectivo de simular o programa (Figura 65) e fazer a sua transferência para o controlador real do robô.

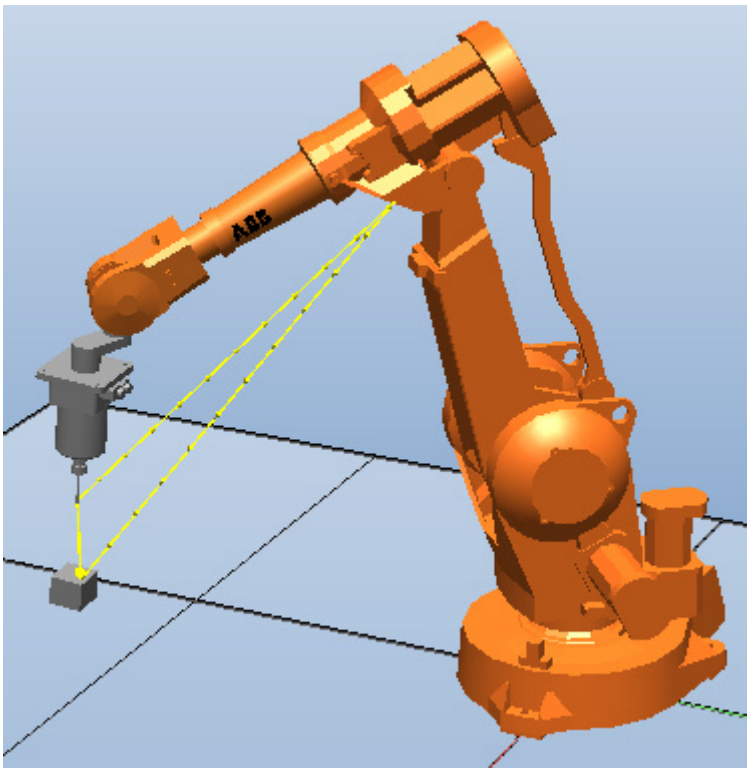


Figura 65 - Simulação do programa no *RobotStudio*

No entanto, como já foi referido a posição do referencial do TCP da ferramenta e do referencial do *workobject* na simulação não coincide com a sua posição na célula real, deste modo foi necessário redefinir a posição destes na célula real, antes de se passar à execução do programa (Figura 66). Só assim o robô saberá identificar a localização dos pontos programados bem como a localização do TCP da ferramenta pneumática, para assim poder cumprir as trajectórias a realizar.

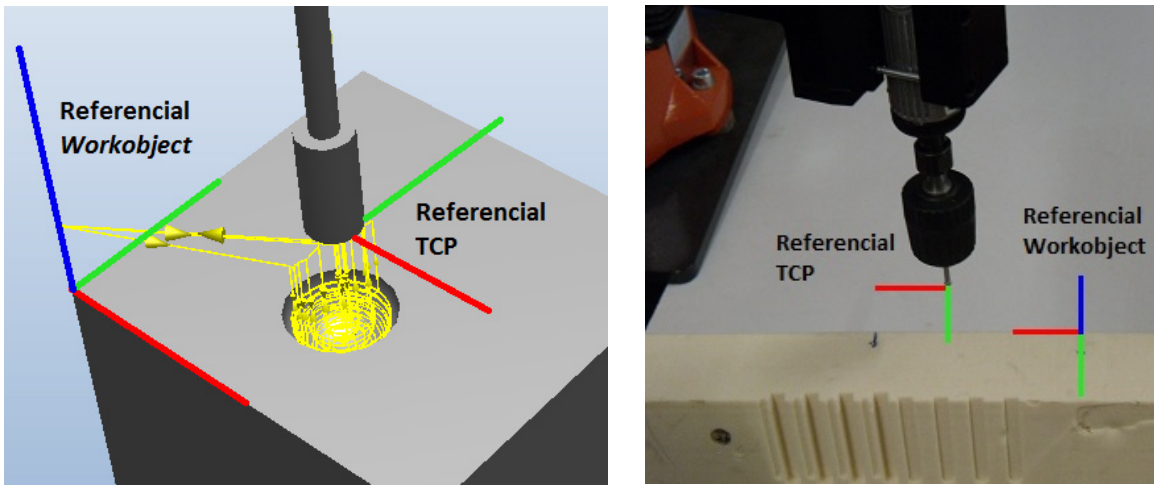


Figura 66 - Posição dos referenciais TCP e *workobject* na simulação (à esquerda) e na célula real (à direita)

Definida a posição dos referenciais TCP e *workobject*, procedeu-se à execução do programa de maquinagem. A Figura 67 apresenta o resultado final do ensaio realizado.

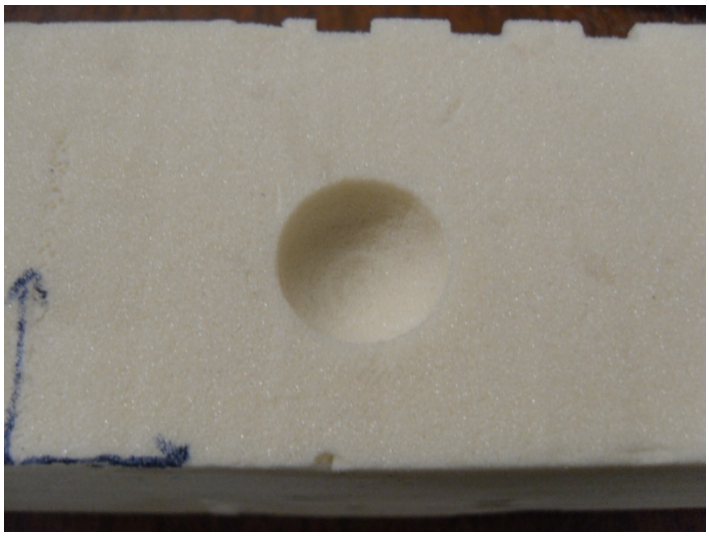


Figura 67 - Cavidade esférica concluída

Ensaio 2: Elaboração de um protótipo constituído por superfícies esféricas

As etapas do desenvolvimento para a criação do protótipo são semelhantes ao ensaio 1, diferenciando-se nas definições das trajectórias e estratégias de operação.

O modelo elaborado em CAD (*SolidWorks*) é apresentado na Figura 68.

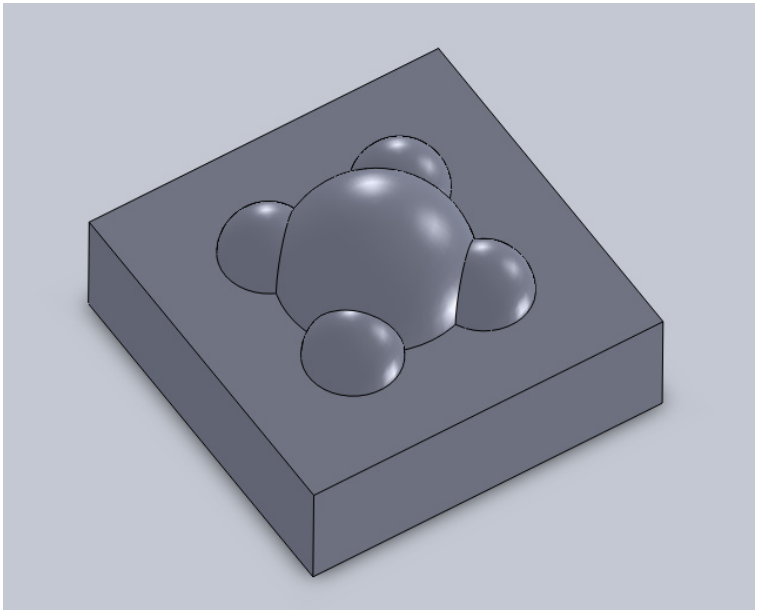


Figura 68 – Protótipo - Modelo em CAD

A estratégia de maquinagem definida no *software* de CAM *MeshCAM* consistiu numa operação de desbaste realizado com uma fresa cilíndrica de 7,8mm e duas operações de acabamento com uma fresa de ponta esférica de 4,8mm (Figuras 69, 70 e 71).

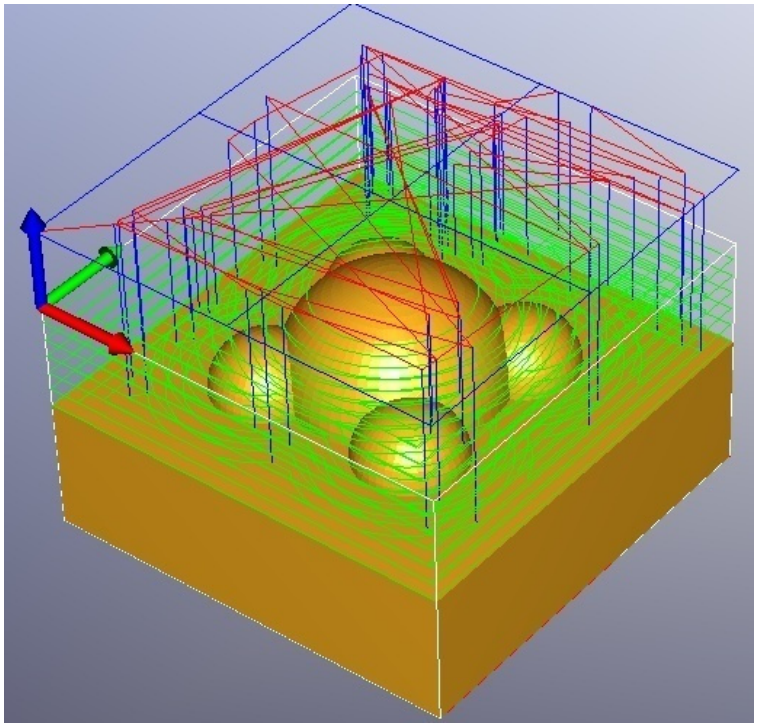


Figura 69 – Percursos da ferramenta para a operação de desbaste

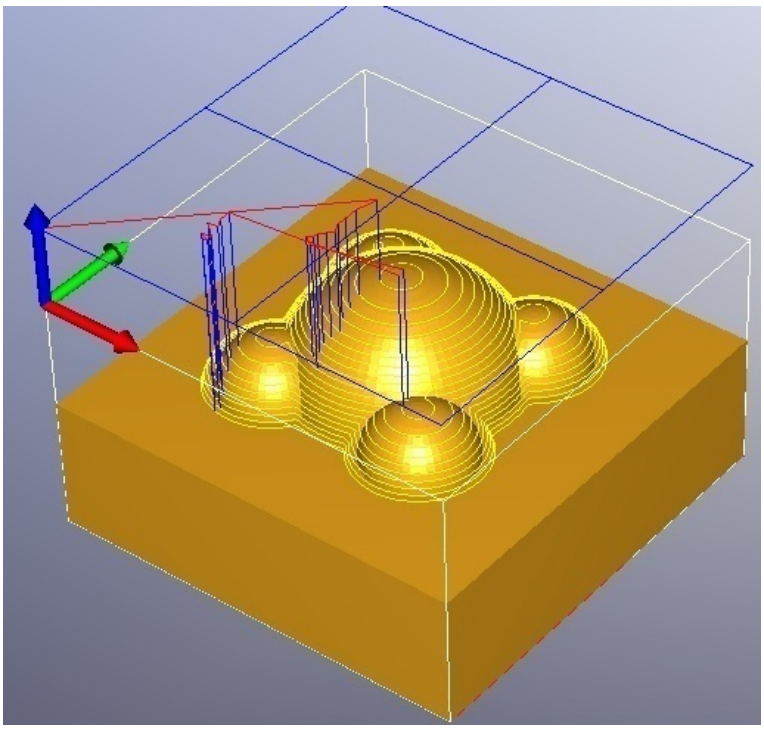


Figura 70 – Percursos da ferramenta para a primeira operação de acabamento

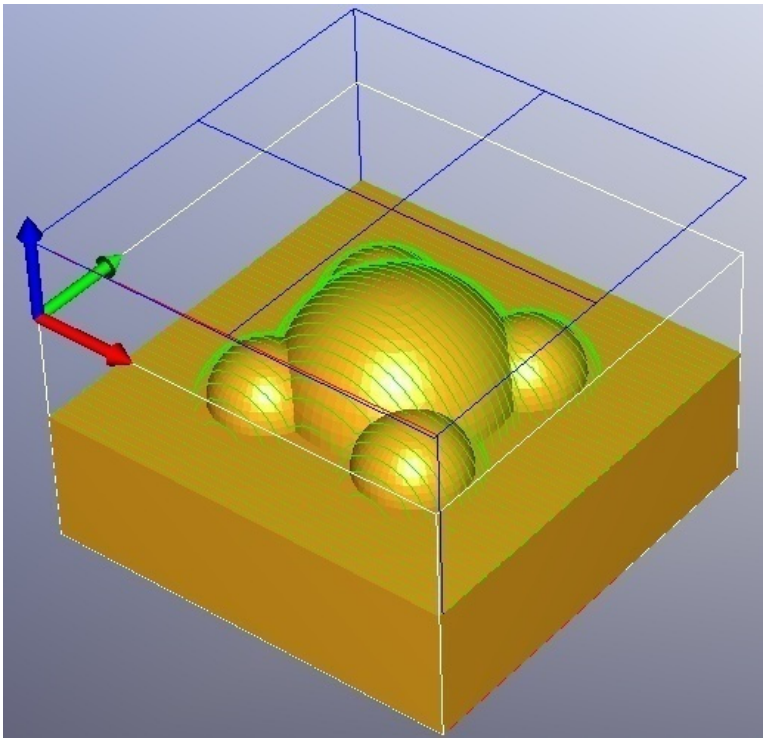


Figura 71 – Percursos da ferramenta para a segunda operação de acabamento

Depois de criado o programa em código G e feito o seu pós-processamento para a linguagem RAPID com o G-RAPID, procedeu-se à sua importação para o *RobotStudio*, onde foi feita a sincronização com o controlador virtual e simulado o programa de maquinagem (Figura 72).

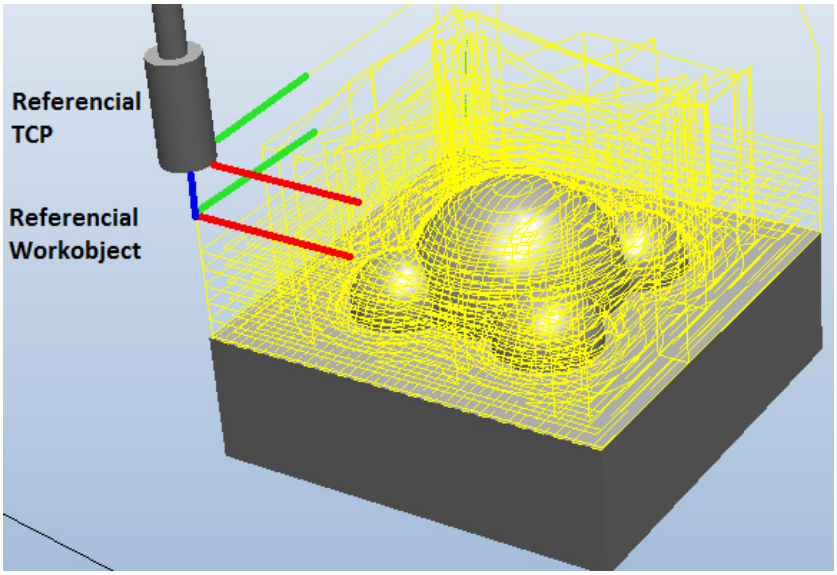


Figura 72 – Simulação do programa de maquinagem no *RobotStudio*

Após a simulação, o programa foi transferido para o controlador real, tendo-se efectuado o processo de execução de trajectórias na célula real.

Relativamente aos referenciais TCP e *workobject*, apenas foi necessário definir a posição do referencial *workobject*, visto a posição do referencial TCP ser a mesma que foi definida no ensaio 1.

Na Figura 73 é ilustrada a célula robotizada utilizada como todos os passos realizados para a execução do processo de maquinagem para a obtenção do protótipo pretendido (Figuras 74 a 80).

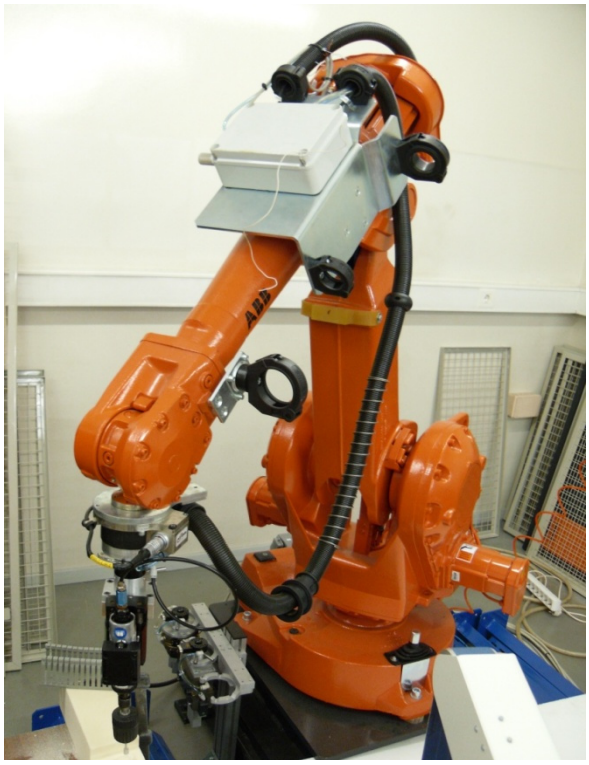


Figura 73 - Célula robotizada disponível no laboratório de robótica

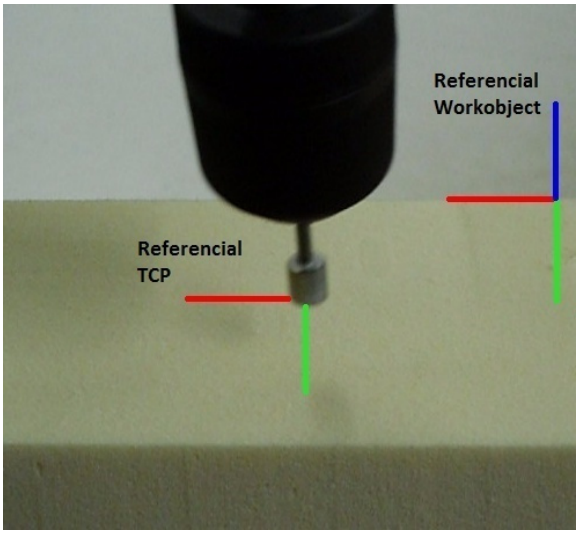


Figura 74 - Definição do referencial *workobject* e inicio da operação de desbaste

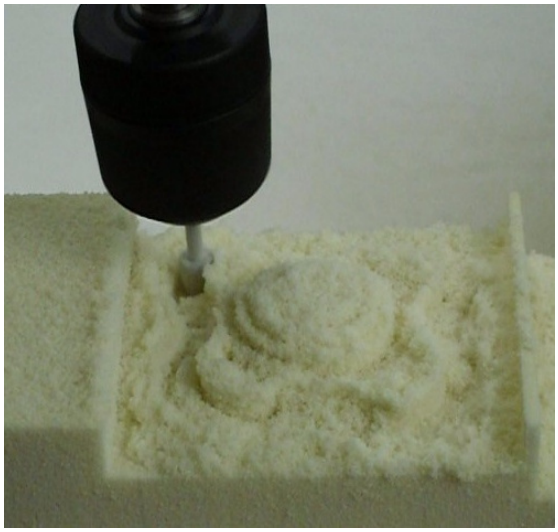


Figura 75 - Operação de desbaste

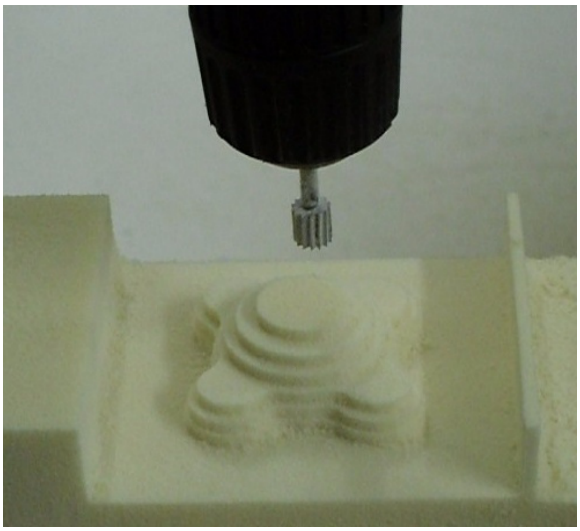


Figura 76 - Operação de desbaste concluída

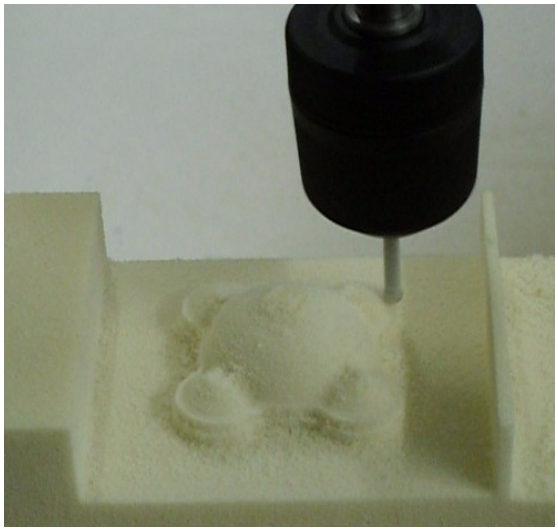


Figura 77 - Primeira operação de acabamento

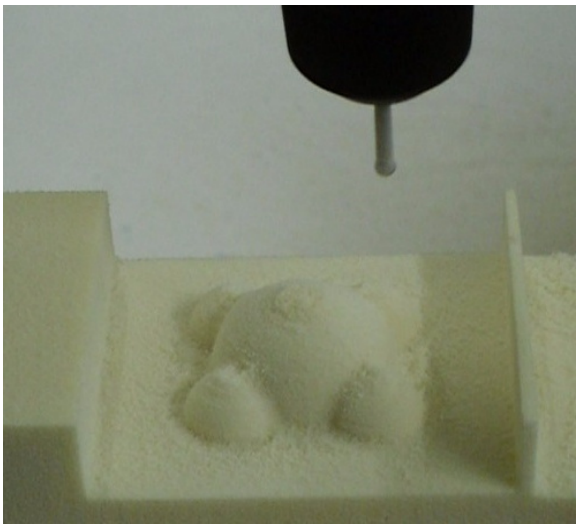


Figura 78 - Primeira operação de acabamento concluída

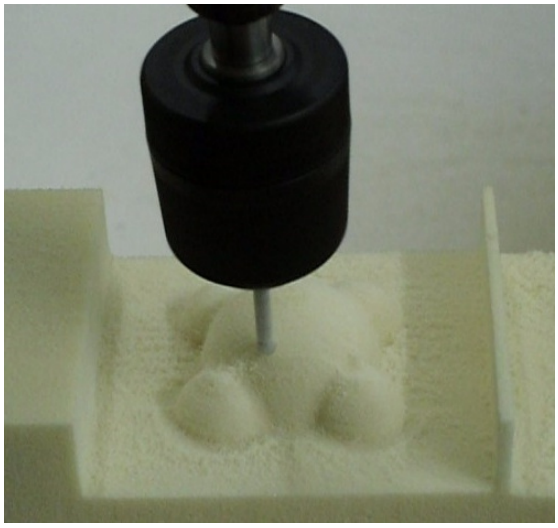


Figura 79 - Segunda operação de acabamento

Figura 80 - Aspecto final do protótipo produzido

6 Conclusões e Trabalhos Futuros

6.1 Resultados e discussões

Os procedimentos experimentais permitiram a fabricação de modelos geométricos através de operações de maquinagem usando um sistema robótico. O pós-processamento do ficheiro de programa NC (código G) gerou um programa com informações de trajectórias e parâmetros necessários para a fabricação dos modelos por maquinagem, através de um robô, a partir de informações geométricas provenientes dos sistemas CAD e da utilização de recursos e funções dos sistemas CAM, atingindo assim os objectivos para os quais o sistema se propôs.

Uma das dificuldades encontradas na execução dos procedimentos experimentais foi o elevado número de pontos criados no CAM para a elaboração do programa de maquinagem, especialmente no segundo ensaio, onde foram criados aproximadamente 6000 pontos.

Este elevado número de pontos do programa quando importado para o *software* de programação *off-line* de robôs o *RobotStudio*, provocou uma certa instabilidade, tornando o *RobotStudio* muito pesado e lento, sendo por vezes necessário reiniciar o *software*. No entanto, numa tentativa de contornar a situação organizaram-se os vários pontos constituintes do programa em percursos de 500 pontos cada. Isto facilitou o processo de sincronização dos pontos com o controlador virtual, permitindo assim a elaboração da simulação do programa de maquinagem no *RobotStudio*.

O *software* de CAM no qual se elaboraram os programas para os ensaios foi o *MeshCAM*. Como já foi referido no capitulo 3, este *software* apenas gera trajectórias com interpolação linear, o que não é a melhor forma de descrever trajectórias que acompanhem superfícies curvas, sendo necessário mais pontos para conseguir descrever uma superfície curva através de vários pequenos segmentos lineares do que no caso do uso de trajectórias com interpolações circulares. Mesmo assim o *MeshCAM* possui uma interface gráfica

bastante poderosa e interactiva constituindo uma boa solução de programa de CAM para maquinagem em 3D, revelando-se uma peça fundamental para o alcance do objectivo deste trabalho.

Relativamente à ferramenta pneumática utilizada e o seu suporte, estes não foram objectos de estudo, no entanto verificou-se um pequeno desalinhamento da ferramenta relativamente ao eixo do suporte. O suporte usado não pertencia à ferramenta pneumática, o que dificultou a definição da posição do referencial TCP da ferramenta.

Quanto à qualidade superficial das peças produzidas, o acabamento proporcionado pelo sistema robótico revelou-se como satisfatório. Porém, embora fosse interessante proceder à sua verificação dimensional, tal procedimento não foi abordado, pois o objectivo do presente trabalho está focado mais na capacidade de implementação de uma solução robótica de maquinagem, do que propriamente na análise dimensional das referidas peças produzidas.

6.2 Conclusões

Os resultados dos ensaios experimentais efectuados na célula real com o robô *ABB* vieram a comprovar a viabilidade da aplicação do pós-processador desenvolvido (G-RAPID) e a sua integração com os sistemas computacionais de auxílio à manufactura (sistemas CAD/CAM).

As vantagens do desenvolvimento de um pós-processador capaz de interpretar os códigos de programação das trajectórias das máquinas-ferramenta CNC e convertê-los em programas na linguagem de robôs, estão na versatilidade, flexibilidade e diminuição do tempo de programação do sistema robótico.

O algoritmo implementado no pós-processador G-RAPID permite que o utilizador possa programar trajectórias e parâmetros de operação de uma forma expedita sem necessidade de saber programar na linguagem RAPID. O utilizador pode programar através da linguagem de programação em código G utilizando um programa de CAM. Deste modo é possível usar o mesmo software de CAM quer para gerar o código G para uma máquina-ferramenta CNC quer para um robô industrial. Neste último caso o G-RAPID encarrega-se de adaptar os parâmetros de operação para o sistema robótico.

Numa perspectiva geral, embora os robôs industriais não estejam ao mesmo nível das máquinas ferramentas CNC no que diz respeito à precisão de posicionamento e rigidez do sistema, possuem as vantagens, da utilização de seis graus de liberdade para movimentação da ferramenta em conjunto com um elevado espaço de trabalho, o que lhes confere uma flexibilidade notável. Por estas razões a aplicação para prototipagem rápida por maquinagem revela um potencial bastante interessante para os robôs industriais, nomeadamente no que diz respeito à maquinagem de protótipos de grandes dimensões em materiais macios, onde a precisão dimensional não é crítica.

6.3 Trabalhos Futuros

Para complementar o estudo efectuado no presente trabalho, deverá ser explorada a utilização de diferentes materiais para a execução dos protótipos, como madeira e outros materiais compósitos, sendo no entanto necessário fazer um estudo prévio das ferramentas a utilizar e do seu accionamento.

De forma a tornar o processo mais expedito sugere-se a implementação das funções M do código G no pós-processador desenvolvido. Sabendo que as funções M estão intimamente ligadas com funções da máquina-ferramenta CNC, o G-RAPID necessitará de elaborar sub-rotinas na linguagem RAPID, para a execução destas mesmas funções no sistema robótico.

Por fim, igualmente interessante, seria o desenvolvimento do pós-processador G-RAPID para a execução de maquinagem em 5 eixos com o sistema robótico, usando para isso um CAM capaz de gerar o código G para máquinas CNC de 5 eixos.

7 Referências Bibliográficas

ABB. *ABB Robotics.* [Online] www.abb.com/robotics (consultado em 25/02/2010)
—. 2009. Machining PowerPac. *Operating Manual. Revision C,* 2009
—. 2002. *RobotStudio.* Maio de 2002.

Abreu, Paulo. (2001). "Aplicações industriais de robôs". Faculdade de Engenharia da Universidade do Porto. Porto: s.n., 2001. Textos de Apoio.

Alves, F. J. Lino, Braga, F. J. Sousa, S. Simão, Manuel, Neto, Rui J., e Duarte, Teresa M. (2001), "PROTOCLICK - Prototipagem Rápida", Porto, Fevereiro 2001.

Chua, C.K. e Leong, K.F. (1997). Rapid Prototyping: Principles & Applications in Manufacturing, New York, NY. 1997.

fabricamachinale. *fabricamachinale.* [online] www.fabricamachinale.it (consultado em 11/03/2010)

FANUC. *FANUC Robotics.* [Online] www.fanucrobotics.com (consultado em 17/03/2010)

Garner Holt Productions. *Garner Holt Productions Inc.* [Online] www.garnerholt.com (consultado em 12/03/2010)

Groover, Mikell P. (1996). "Industrial robotics : technology, programming, and applications". New York : McGraw - Hill Book Company, 1996.

Kuka. *Kuka Robot group.* [Online] www.kuka-robotics.com/en (consultado em 17/03/2010)

Lennings, A.F. (1997), "Advantages of using CNC milling for rapid prototyping", in Roller, D., Proceedings of the 30th International Symposium on Automotive Technology & Automation (ISATA), Florence, pp. 113-120.

Lye, S. W., Yeong, H. Y. e Lee, S. G. (1996). "An investigation into the rapid prototyping of moulds for expanded polystyrene foam". The International Journal of Advanced Manufacturing Technology .1996, Vol 12, No. 2, pp. 87-92.

Motoman. *Motoman*. [Online] www.motoman.com (consultado em 17/03/2010)

Ribeiro, A. F. M., Norrish, J. e Mcmaster R. (1994), "Practical case of Rapid Prototyping using Gas Metal Arc Welding", 5th International Conference on Computer Technology in Welding, Paris, France, 15-16 June 1994.

Schaaf, W. (2000). "Robotyping - new rapid prototyping processes for sand mould using industrial robots".Assembly Automation. 2000 Vol. 20 , No. 4, pp. 321-329.

Simtech Systems. *Simtech Systems* [Online] www.easysimulation.com/web/html/simtech_en (consultado em 13/03/2010)

Sirviö, M. (2008). "Patternless casting is ideal for prototyping". HiTech Finland. 2008.

Sirviö, M. e Wos, M. (2009). "Casting directly from a computer model by using advanced simulation software FLOW-3D Cast". ARCHIVES OF FOUNDRY ENGINEERING. 2009, Vol. 9, pp. 79-82.

S.N.B.R. *S.N.B.R.* [online] www.snbr-stone.com (consultado em 13/03/2010)

Stäubli. *Staubli Robotics* [Online] www.staubli.com/en/robotics (consultado em 19/03/2010)

Vergeest, J. S. e Tangelder, J. W. (1996)."Robot machines rapid prototype". Industrial Robot. 1996, Vol. 23, No. 5, pp. 17–20.

Walstra, W.H., Bronsvoort, W.F. and Vergeest, J.S.M. (1994),"Interactive simulation of robot milling for rapid shape prototyping", Computers and Graphics, Vol. 18 No. 6, 1994, pp. 861-871.

ANEXOS

ANEXO A - Código de Programação do Pós-Processador Desenvolvido

```
Sub Organizar()

Sheets("Folha1").Activate
lastrow = Cells(Rows.Count, "A").End(xlUp).Row

'A primeira linha do código tem que possuir coordenada X coordenada Y e coordenada Z na posição respectiva
If Left(Cells(1, "B"), 1) = "Y" Then Cells(1, "C") = Cells(1, "B")
If Left(Cells(1, "B"), 1) = "Y" Then Cells(1, "B").ClearContents
If Left(Cells(1, "B"), 1) = "Z" Then Cells(1, "D") = Cells(1, "B")
If Left(Cells(1, "B"), 1) = "Z" Then Cells(1, "B").ClearContents
If Left(Cells(1, "C"), 1) = "Z" Then Cells(1, "D") = Cells(1, "C")
If Left(Cells(1, "C"), 1) = "Z" Then Cells(1, "C").ClearContents


If Left(Cells(1, "B"), 1) = "" Then Cells(1, "B") = "X0"
If Left(Cells(1, "C"), 1) = "" Then Cells(1, "C") = "Y0"
If Left(Cells(1, "D"), 1) = "" Then Cells(1, "D") = "Z0"


For i = 1 To lastrow
   For j = 1 To 7
      If Left(Cells(i, j), 1) = "F" Then Cells(i, 14) = Cells(i, j)
      If Left(Cells(i, j), 1) = "X" Then Cells(i, 9) = Cells(i, j)
      If Left(Cells(i, j), 1) = "Y" Then Cells(i, 10) = Cells(i, j)
      If Left(Cells(i, j), 1) = "Z" Then Cells(i, 11) = Cells(i, j)
      If Left(Cells(i, j), 1) = "I" Then Cells(i, 12) = Cells(i, j)
      If Left(Cells(i, j), 1) = "J" Then Cells(i, 13) = Cells(i, j)
      If Left(Cells(i, j), 1) = "G" Then Cells(i, 8) = Cells(i, j)

   Next j
Next i


'Resolve o problema de G01,G02,G03 sem instruções F
For b = 1 To lastrow
   If Cells(b, "N") = "" Then GoTo step1 Else GoTo step2
step1:
   If Cells(b, "A") = "G01" Or Cells(b, "A") = "G02" Or Cells(b, "A") = "G03" Then Cells(b, "N") =
Cells(b - 1, "N")
step2:
Next b


For C = 1 To lastrow
   For d = 9 To 11
      If Cells(C, d) = "" Then Cells(C, d) = Cells(C - 1, d)
   Next d
Next C

End Sub


Public Function radius(x1 As Double, y1 As Double, xc As Double, yc As Double) As Double
radius = Sqr((x1 - xc) ^ 2 + (y1 - yc) ^ 2)
End Function
```

```
Public Function rnorm(x1 As Double, x2 As Double, y1 As Double, y2 As Double)
rnorm = Sqr((-(y2 - y1)) ^ 2 + (x2 - x1) ^ 2)
End Function
Public Function vrnewx(y1 As Double, y2 As Double, rnorma As Double, raio As Double) As Double
vrnewx = (-(y2 - y1)) / (rnorma) * raio
End Function
Public Function vrnewy(x1 As Double, x2 As Double, rnorma As Double, raio As Double) As Double
vrnewy = (x2 - x1) / (rnorma) * raio
End Function
Public Function vCP3x(rnewx As Double, xc As Double) As Double
vCP3x = rnewx
End Function
Public Function vCP3y(rnewy As Double, yc As Double) As Double
vCP3y = rnewy
End Function
Public Function vOP3x(xc As Double, CP3x As Double) As Double
vOP3x = xc + CP3x
End Function
Public Function vOP3y(yc As Double, CP3y As Double) As Double
vOP3y = yc + CP3y
End Function
Public Function vOP4x(xc As Double, CP3x As Double) As Double
vOP4x = xc - CP3x
End Function
Public Function vOP4y(yc As Double, CP3y As Double) As Double
vOP4y = yc - CP3y
End Function
Public Function vCP1norma(x1 As Double, y1 As Double, xc As Double, yc As Double) As Double
vCP1norma = Sqr((x1 - xc) ^ 2 + (y1 - yc) ^ 2)
End Function
Public Function vCP1nx(x1 As Double, xc As Double) As Double
vCP1nx = x1 - xc
End Function
Public Function vCP1ny(y1 As Double, yc As Double) As Double
vCP1ny = y1 - yc
End Function
Public Function vrnewnorma(rnewx As Double, rnewy As Double) As Double
vrnewnorma = Sqr((rnewx) ^ 2 + (rnewy) ^ 2)
End Function
Public Function vrnx(rnewx As Double, rnewnorma As Double) As Double
vrnx = rnewx / rnewnorma
End Function
Public Function vrny(rnewy As Double, rnewnorma As Double) As Double
vrny = rnewy / rnewnorma
End Function
Public Function prodvect(CP1nx As Double, CP1ny As Double, rnx As Double, rny As Double) As
Double
prodvect = (CP1nx * rny) - (CP1ny * rnx)
End Function

Sub InsertRows()
 Sheets("Folha1").Activate
 lastrow = Cells(Rows.Count, "H").End(xlUp).Row
 'Sempre que se remove linhas o loop deve ser feito ao contrário
 For i = lastrow To 1 Step -1
    If Cells(i, "H") = "G02" Or Cells(i, "H") = "G03" Then Cells(i, "H").Select: Selection.EntireRow.Insert
, CopyOrigin:=xlFormatFromLeftOrAbove: i = i + 1
    lastrow = Cells(Rows.Count, "H").End(xlUp).Row
 Next i
```

```
End Sub


Sub Arco()

Dim x1 As Double
Dim y1 As Double
Dim x2 As Double
Dim y2 As Double
Dim xc As Double
Dim yc As Double
Dim rnorma As Double
Dim raio As Double
Dim rnewx As Double
Dim rnewy As Double
Dim CP3x As Double
Dim CP3y As Double
Dim x3 As Double
Dim y3 As Double
Dim x4 As Double
Dim y4 As Double
Dim rnewnorma As Double
Dim CP1nx As Double
Dim CP1ny As Double
Dim rnx As Double
Dim rny As Double
Dim pvect As Double


Sheets("Folha1").Activate


'Substituição dos separadores de casas décimais de "." para "," nas células da coluna I e J e colunas L e M
Columns("I:J").Select
Selection.Replace What:=".", Replacement:=",", LookAt:=xlPart, SearchOrder:=xlByRows,
MatchCase:=False, SearchFormat:=False, ReplaceFormat:=False

Columns("L:M").Select
Selection.Replace What:=".", Replacement:=",", LookAt:=xlPart, SearchOrder:=xlByRows,
MatchCase:=False, SearchFormat:=False, ReplaceFormat:=False

lastrow = Cells(Rows.Count, "H").End(xlUp).Row
For i = 1 To lastrow
   For j = 8 To 13
      If Cells(i, j) = "G02" Or Cells(i, j) = "G03" Then GoTo step1 Else GoTo step2
step1:
      x1 = Mid(Cells(i - 2, j + 1), 2, 20)
      x2 = Mid(Cells(i, j + 1), 2, 20)
      y1 = Mid(Cells(i - 2, j + 2), 2, 20)
      y2 = Mid(Cells(i, j + 2), 2, 20)
      xc = Mid(Cells(i, j + 4), 2, 20)
      yc = Mid(Cells(i, j + 5), 2, 20)
      raio = radius(x1, y1, xc, yc)
      rnorma = rnorm(x1, x2, y1, y2)
      rnewx = vrnewx(y1, y2, rnorma, raio)
      rnewy = vrnewy(x1, x2, rnorma, raio)
      CP1norma = vCP1norma(x1, y1, xc, yc)
      CP3x = vCP3x(rnewx, xc)
      CP3y = vCP3y(rnewy, yc)
      x3 = vOP3x(xc, CP3x)
```

```
        y3 = vOP3y(yc, CP3y)
        x4 = vOP4x(xc, CP3x)
        y4 = vOP4y(yc, CP3y)
        rnewnorma = vrnewnorma(rnewx, rnewy)
        CP1nx = vCP1nx(x1, xc)
        CP1ny = vCP1ny(y1, yc)
        rnx = vrnx(rnewx, rnewnorma)
        rny = vrny(rnewy, rnewnorma)
        pvect = prodvect(CP1nx, CP1ny, rnx, rny)

step2:

        If Cells(i, j) = "G02" Then GoTo step3 Else GoTo step4
step3:
        If pvect < 0 Then Cells(i - 1, j + 1) = x3 Else Cells(i - 1, j + 1) = x4
        If pvect < 0 Then Cells(i - 1, j + 2) = y3 Else Cells(i - 1, j + 2) = y4
step4:
        If Cells(i, j) = "G03" Then GoTo step5 Else GoTo step6
step5:
        If pvect > 0 Then Cells(i - 1, j + 1) = x3 Else Cells(i - 1, j + 1) = x4
        If pvect > 0 Then Cells(i - 1, j + 2) = y3 Else Cells(i - 1, j + 2) = y4
step6:
    Next j
Next i


'Substituição dos separadores de casas decimais de "," para "." nas células da coluna I e J e colunas L e M
Columns("I:J").Select
Selection.Replace What:=",", Replacement:=".", LookAt:=xlPart, SearchOrder:=xlByRows,
MatchCase:=False, SearchFormat:=False, ReplaceFormat:=False

Columns("L:M").Select
Selection.Replace What:=",", Replacement:=".", LookAt:=xlPart, SearchOrder:=xlByRows,
MatchCase:=False, SearchFormat:=False, ReplaceFormat:=False

'Vai adicionar a coordenada Z do ponto médio do arco que é igual à coordenada Z da instrução anterior
For k = 1 To lastrow
    If Cells(k, "H") = "" Then Cells(k, "K") = Cells(k - 1, "K")
Next k

'Organiza a coluna "N" em que cada instrução G de movimento tem a sua velocidade'
'É feito tambem a conversão de velocidade de avanço de mm/min em mm/seg
For l = 1 To lastrow
    If Cells(l, "H") = "G00" Then Cells(l, "N") = 100 '(100mm/seg) 'é a velocidade posicionamento

    If Cells(l, "H") = "G01" Or Cells(l, "H") = "G02" Or Cells(l, "H") = "G03" Then Cells(l, "N") =
Mid(Cells(l, "N"), 2, 20) / 60
    If Cells(l, "H") = "" Then Cells(l, "N") = Mid(Cells(l + 1, "N"), 2, 20) / 60
    'A função RoundUP faz o arredondamento, RoundUP(....,0) em que 0 indica arredondamento para o
inteiro mais proximo
    Cells(l, "N") = Application.RoundUp(Cells(l, "N"), 0)
Next l

End Sub
```

```
Sub ProgRAPID()

Sheets("Folha2").Activate
Cells(1, 1) = "MODULE"
Cells(1, 2) = "default"
'Abre a folha1
Sheets("Folha1").Activate
'Vai à Folha1 ver qual é a ultima linha
lastrow = Cells(Rows.Count, "H").End(xlUp).Row
'Abre a Folha2
Sheets("Folha2").Activate

'Vai selecionar a célula D2 e fazer autofill
   Cells(2, "C") = Cells(2, "C").Select
   ActiveCell.FormulaR1C1 = "p1"
   Selection.AutoFill Destination:=Range(Cells(2, "C"), Cells(lastrow + 1, "C")), Type:=xlFillDefault
For i = 2 To lastrow + 1
   Cells(i, "A") = "CONST"
   Cells(i, "B") = "robtarget"

'Troca separadores decimais de virgula para ponto
   With Application
   .DecimalSeparator = "."
   .UseSystemSeparators = False
   End With
'Copiar os valores das celulas com a coordenada X da folha1 para eliminando a string "X"
   Cells(i, "D") = Sheets("Folha1").Cells(i - 1, "I")
   If Left(Cells(i, "D"), 1) = "X" Then Cells(i, "D") = Mid(Cells(i, "D"), 2, 20)
'Copiar os valores das células com a coordenada Y da folha1 eliminando a string "Y"
   Cells(i, "E") = Sheets("Folha1").Cells(i - 1, "J")
   If Left(Cells(i, "E"), 1) = "Y" Then Cells(i, "E") = Mid(Cells(i, "E"), 2, 20)
'Copiar os valores das células com a coordenada Z da folha1 eliminando a string "Z"
   Cells(i, "F") = Sheets("Folha1").Cells(i - 1, "K")
   If Left(Cells(i, "F"), 1) = "Z" Then Cells(i, "F") = Mid(Cells(i, "F"), 2, 20)

'Orientação de cada um dos pontos, por defeito será [q1,q2,q3,q4]=[1,0,0,0]
   Cells(i, "I") = 1
   Cells(i, "J") = 0
   Cells(i, "K") = 0
   Cells(i, "L") = 0
'Todos os pontos têm configuração [0,0,0,0]
   Cells(i, "N") = "[0,0,0,0]"
'Nenhum eixo externo será usado logo:
   Cells(i, "O") = "[9E9,9E9,9E9,9E9,9E9,9E9]];"

Next i

'Neste loop na coluna C é adicionado uma string
For k = 2 To lastrow + 1
   Cells(k, "C") = Cells(k, "C") & ":=[["
   Cells(k, "G") = "]"
   Cells(k, "H") = "["
   Cells(k, "M") = "]"

Next k

'Escrita da subrotina
Cells(lastrow + 3, "A") = "PROC"
Cells(lastrow + 3, "B") = "Path_default()"
```

```
'Vai seleccionar a célula Cells(lastrow+4, "B") e fazer autofill
Cells(lastrow + 4, "B") = Cells(lastrow + 4, "B").Select
ActiveCell.FormulaR1C1 = "p1"
    Selection.AutoFill Destination:=Range(Cells(lastrow + 4, "B"), Cells(lastrow + lastrow + 3, "B")),
Type:=xlFillDefault

For m = lastrow + 4 To lastrow + lastrow + 3
'Vai ler o valor da celula H da Folha1 se for G00 a instrução é MoveJ
'Se for G01 a instrução é MoveL
'Se for G02 ou G03 ou vazio a instrução é MoveC
Cells(m, "A") = Sheets("Folha1").Cells(m - lastrow - 3, "H")
If Cells(m, "A") = "G00" Then Cells(m, "A") = "MoveJ"
If Cells(m, "A") = "G01" Then Cells(m, "A") = "MoveL"
If Cells(m, "A") = "G02" Or Cells(m, "A") = "G03" Then Cells(m, "A") = "MoveC"
If Cells(m, "A") = "MoveC" Then Cells(m, "C") = Cells(m, "B")
If Cells(m, "A") = "MoveC" Then Cells(m, "B") = Cells(m - 1, "B")
'Escrita do valor das velocidades nas instruções de movimento
If Cells(m, "A") = "MoveJ" Or Cells(m, "A") = "MoveL" Then Cells(m, "C") = "v" &
Sheets("Folha1").Cells(m - lastrow - 3, "N")
If Cells(m, "A") = "MoveC" Then Cells(m, "D") = "v" & Sheets("Folha1").Cells(m - lastrow - 3, "N")
'Escrita do valor z(zone data), z1 foi escolhido por defeito
If Cells(m, "A") = "MoveJ" Or Cells(m, "A") = "MoveL" Then Cells(m, "D") = "z1"
If Cells(m, "A") = "MoveC" Then Cells(m, "E") = "z1"
'Escrita do nome da ferramenta
If Cells(m, "A") = "MoveJ" Or Cells(m, "A") = "MoveL" Then Cells(m, "E") = "Fresa"
If Cells(m, "A") = "MoveC" Then Cells(m, "F") = "Fresa"
'Escrita do nome do workobject
If Cells(m, "A") = "MoveJ" Or Cells(m, "A") = "MoveL" Then Cells(m, "F") = "\WObj:=Workobject_3;"
If Cells(m, "A") = "MoveC" Then Cells(m, "G") = "\WObj:=Workobject_3;"

Next m
'Fim da procedure
Cells(lastrow + lastrow + 4, "A") = "ENDPROC"
'Fim do modulo
Cells(lastrow + lastrow + 6, "A") = "ENDMODULE"

'Remove as linhas que não são necessárias no programa
'Sempre que se removem linhas o loop deve ser feito ao contrário
For N = lastrow + lastrow + 3 To lastrow + 4 Step -1
    If Cells(N, "A") = "" Then GoTo step1 Else GoTo step2
step1:
    Range(Cells(N, "A"), Cells(N, "Z")).Select
    Selection.Delete Shift:=xlUp
    N = N - 1
step2:
Next N

End Sub


Sub Get_Data()
FileToOpen = Application.GetOpenFilename _
(Title:="Importar Ficheiro", _
fileFilter:="Ficheiro NC, *.nc, Excel Files, *.xls, All Files, *.*")
"
If FileToOpen = False Then
MsgBox "O ficheiro não foi encontrado.", vbExclamation, "Erro...!"
Exit Sub
Else
Workbooks.OpenText Filename:=FileToOpen, Origin:= _
```

```
        xlMSDOS, StartRow:=1, DataType:=xlDelimited, TextQualifier:=xlDoubleQuote _
        , ConsecutiveDelimiter:=True, Tab:=True, Semicolon:=False, Comma:=False _
        , Space:=True, Other:=False, FieldInfo:=Array(Array(1, 1), Array(2, 1), Array _
        (3, 1), Array(4, 1), Array(5, 1), Array(6, 1), Array(7, 1)), TrailingMinusNumbers:= _True
    Filename = ActiveWorkbook.Name

    Windows(Filename).Activate
    Cells.Select
    Selection.Copy
    Windows("G-RAPID.xlsm").Activate
    Sheets("Folha1").Select
    Cells(1, 1).Select
    ActiveSheet.Paste
    Application.CutCopyMode = False
    Range("A1").Select
    Windows(Filename).Activate

    ActiveWorkbook.Close Filename:=FileToOpen
    Windows("G-RAPID.xlsm").Activate
End If
End Sub



Sub Save_and_edit()
FileSaveName = Application.GetSaveAsFilename( _
    fileFilter:="Text Files (*.txt), *.txt, RAPID module files, *.mod,All Files, *.*")
  If FileSaveName = False Then
MsgBox "O ficheiro não foi guardado.", vbExclamation, "Erro...!"
Exit Sub
Else
    Application.DisplayAlerts = False
    Application.ScreenUpdating = False
    Sheets("Folha2").Select
    ActiveWorkbook.Save
    Cells.Select
    Selection.Copy
    Workbooks.Add
    ActiveSheet.Paste
    Application.CutCopyMode = False
    ActiveWorkbook.SaveAs Filename:=FileSaveName, FileFormat:=xlCSV, CreateBackup:=False
    ActiveWindow.Close
    Sheets("Folha2").Select
    Range("A1").Select

    Application.DisplayAlerts = True
    Application.ScreenUpdating = True

Dim lOpenFile As Long
Dim sFileText As String
Dim sFileName As String

sFileName = FileSaveName
'Abre o ficheiro e lê-o numa variavel
lOpenFile = FreeFile
Open sFileName For Input As lOpenFile
sFileText = Input(LOF(lOpenFile), lOpenFile)
Close lOpenFile

'sFileText = Replace(sFileText, "find", "replace")
```

```
sFileText = Replace(sFileText, "MODULE,", "MODULE ")
sFileText = Replace(sFileText, "CONST,", "CONST ")
sFileText = Replace(sFileText, "robtarget,", "robtarget ")
sFileText = Replace(sFileText, "[,", "[")
sFileText = Replace(sFileText, ",]", "]")
sFileText = Replace(sFileText, """", "")
sFileText = Replace(sFileText, ",,,", "")
sFileText = Replace(sFileText, ",,", "")
sFileText = Replace(sFileText, ";,", ";")
sFileText = Replace(sFileText, "MoveJ,", "MoveJ ")
sFileText = Replace(sFileText, "MoveL,", "MoveL ")
sFileText = Replace(sFileText, "MoveC,", "MoveC ")
sFileText = Replace(sFileText, ",\", "\")
sFileText = Replace(sFileText, "PROC,", "PROC ")
sFileText = Replace(sFileText, "ENDPROC,", "ENDPROC")
sFileText = Replace(sFileText, "ENDMODULE,", "ENDMODULE")
'Vai escrever no ficheiro
lOpenFile = FreeFile
Open sFileName For Output As lOpenFile
Print #lOpenFile, sFileText
Close lOpenFile

End If
End Sub



Private Sub UserForm_Initialize()

UserForm2.Show
ComboBox1.AddItem "v5"
ComboBox1.AddItem "v10"
ComboBox1.AddItem "v20"
ComboBox1.AddItem "v30"
ComboBox1.AddItem "v40"
ComboBox1.AddItem "v50"
ComboBox1.AddItem "v60"
ComboBox1.AddItem "v80"
ComboBox1.AddItem "v100"
ComboBox1.AddItem "v150"
ComboBox1.AddItem "v200"
ComboBox1.AddItem "v300"
ComboBox1.AddItem "v400"
ComboBox1.AddItem "v500"
ComboBox1.AddItem "v600"
ComboBox1.AddItem "v800"
ComboBox1.AddItem "v1000"
ComboBox1.AddItem "v1500"
ComboBox1.AddItem "v2000"
ComboBox1.AddItem "v2500"
ComboBox1.AddItem "v3000"
ComboBox1.AddItem "v4000"
ComboBox1.AddItem "v5000"
ComboBox1.AddItem "v6000"
ComboBox1.AddItem "v7000"
ComboBox1.AddItem "vmax"

ComboBox2.AddItem "v5"
ComboBox2.AddItem "v10"
ComboBox2.AddItem "v20"
```

```
ComboBox2.AddItem "v30"
ComboBox2.AddItem "v40"
ComboBox2.AddItem "v50"
ComboBox2.AddItem "v60"
ComboBox2.AddItem "v80"
ComboBox2.AddItem "v100"
ComboBox2.AddItem "v150"
ComboBox2.AddItem "v200"
ComboBox2.AddItem "v300"
ComboBox2.AddItem "v400"
ComboBox2.AddItem "v500"
ComboBox2.AddItem "v600"
ComboBox2.AddItem "v800"
ComboBox2.AddItem "v1000"
ComboBox2.AddItem "v1500"
ComboBox2.AddItem "v2000"
ComboBox2.AddItem "v2500"
ComboBox2.AddItem "v3000"
ComboBox2.AddItem "v4000"
ComboBox2.AddItem "v5000"
ComboBox2.AddItem "v6000"
ComboBox2.AddItem "v7000"
ComboBox2.AddItem "vmax"

ComboBox3.AddItem "v5"
ComboBox3.AddItem "v10"
ComboBox3.AddItem "v20"
ComboBox3.AddItem "v30"
ComboBox3.AddItem "v40"
ComboBox3.AddItem "v50"
ComboBox3.AddItem "v60"
ComboBox3.AddItem "v80"
ComboBox3.AddItem "v100"
ComboBox3.AddItem "v150"
ComboBox3.AddItem "v200"
ComboBox3.AddItem "v300"
ComboBox3.AddItem "v400"
ComboBox3.AddItem "v500"
ComboBox3.AddItem "v600"
ComboBox3.AddItem "v800"
ComboBox3.AddItem "v1000"
ComboBox3.AddItem "v1500"
ComboBox3.AddItem "v2000"
ComboBox3.AddItem "v2500"
ComboBox3.AddItem "v3000"
ComboBox3.AddItem "v4000"
ComboBox3.AddItem "v5000"
ComboBox3.AddItem "v6000"
ComboBox3.AddItem "v7000"
ComboBox3.AddItem "vmax"

ComboBox4.AddItem "fine"
ComboBox4.AddItem "z1"
ComboBox4.AddItem "z5"
ComboBox4.AddItem "z10"
ComboBox4.AddItem "z15"
ComboBox4.AddItem "z20"
ComboBox4.AddItem "z30"
ComboBox4.AddItem "z40"
ComboBox4.AddItem "z50"
```

```
ComboBox4.AddItem "z60"
ComboBox4.AddItem "z80"
ComboBox4.AddItem "z100"
ComboBox4.AddItem "z150"
ComboBox4.AddItem "z200"

ComboBox5.AddItem "fine"
ComboBox5.AddItem "z1"
ComboBox5.AddItem "z5"
ComboBox5.AddItem "z10"
ComboBox5.AddItem "z15"
ComboBox5.AddItem "z20"
ComboBox5.AddItem "z30"
ComboBox5.AddItem "z40"
ComboBox5.AddItem "z50"
ComboBox5.AddItem "z60"
ComboBox5.AddItem "z80"
ComboBox5.AddItem "z100"
ComboBox5.AddItem "z150"
ComboBox5.AddItem "z200"

ComboBox6.AddItem "fine"
ComboBox6.AddItem "z1"
ComboBox6.AddItem "z5"
ComboBox6.AddItem "z10"
ComboBox6.AddItem "z15"
ComboBox6.AddItem "z20"
ComboBox6.AddItem "z30"
ComboBox6.AddItem "z40"
ComboBox6.AddItem "z50"
ComboBox6.AddItem "z60"
ComboBox6.AddItem "z80"
ComboBox6.AddItem "z100"
ComboBox6.AddItem "z150"
ComboBox6.AddItem "z200"

End Sub

Private Sub TextBox1_Change()
Sheets("Folha2").Activate
Cells(1, 2) = TextBox1.Text

End Sub

Private Sub TextBox2_Change()
Sheets("Folha1").Activate
'Vai à Folha1 ver qual é a ultima linha
lastrow = Cells(Rows.Count, "H").End(xlUp).Row
Sheets("Folha2").Activate
Cells(lastrow + 3, "B") = TextBox2.Text & "()"
End Sub

Private Sub TextBox3_Change()
Sheets("Folha1").Activate
'Vai à Folha1 ver qual é a ultima linha
lastrow = Cells(Rows.Count, "H").End(xlUp).Row
Sheets("Folha2").Activate
For m = lastrow + 4 To lastrow + lastrow + 3
   If Cells(m, "A") = "MoveJ" Or Cells(m, "A") = "MoveL" Then Cells(m, "E") = TextBox3.Text
   If Cells(m, "A") = "MoveC" Then Cells(m, "F") = TextBox3.Text
```

```
Next m
End Sub

Private Sub TextBox4_Change()
Sheets("Folha1").Activate
'Vai à Folha1 ver qual é a ultima linha
lastrow = Cells(Rows.Count, "H").End(xlUp).Row
Sheets("Folha2").Activate
For m = lastrow + 4 To lastrow + lastrow + 3
   If Cells(m, "A") = "MoveJ" Or Cells(m, "A") = "MoveL" Then Cells(m, "F") = "\WObj:=" &
TextBox4.Text & ";"
   If Cells(m, "A") = "MoveC" Then Cells(m, "G") = "\WObj:=" & TextBox4.Text & ";"
Next m
End Sub

Private Sub TextBox8_Change()
Sheets("Folha1").Activate
'Vai à Folha1 ver qual é a ultima linha
lastrow = Cells(Rows.Count, "H").End(xlUp).Row
'Abre a Folha2
Sheets("Folha2").Activate
   For i = 2 To lastrow + 1
   Cells(i, "I") = TextBox8.Text
Next i
End Sub

Private Sub TextBox9_Change()
Sheets("Folha1").Activate
'Vai à Folha1 ver qual é a ultima linha
lastrow = Cells(Rows.Count, "H").End(xlUp).Row
'Abre a Folha2
Sheets("Folha2").Activate
For i = 2 To lastrow + 1
   Cells(i, "J") = TextBox9.Text
Next i
End Sub

Private Sub TextBox10_Change()
Sheets("Folha1").Activate
'Vai à Folha1 ver qual é a ultima linha
lastrow = Cells(Rows.Count, "H").End(xlUp).Row
'Abre a Folha2
Sheets("Folha2").Activate
For i = 2 To lastrow + 1
   Cells(i, "K") = TextBox10.Text
Next i
End Sub

Private Sub TextBox11_Change()
Sheets("Folha1").Activate
'Vai à Folha1 ver qual é a ultima linha
lastrow = Cells(Rows.Count, "H").End(xlUp).Row
'Abre a Folha2
Sheets("Folha2").Activate
For i = 2 To lastrow + 1
   Cells(i, "L") = TextBox11.Text
Next i
End Sub

Private Sub ComboBox1_Change()
```

```
Sheets("Folha1").Activate
'Vai à Folha1 ver qual é a ultima linha
lastrow = Cells(Rows.Count, "H").End(xlUp).Row
Sheets("Folha2").Activate
For m = lastrow + 4 To lastrow + lastrow + 3
   If Cells(m, "A") = "MoveJ" Then Cells(m, "C") = ComboBox1.Value
Next m
End Sub

Private Sub ComboBox2_Change()
Sheets("Folha1").Activate
'Vai à Folha1 ver qual é a ultima linha
lastrow = Cells(Rows.Count, "H").End(xlUp).Row
Sheets("Folha2").Activate
For m = lastrow + 4 To lastrow + lastrow + 3
   If Cells(m, "A") = "MoveL" Then Cells(m, "C") = ComboBox2.Value
Next m
End Sub

Private Sub ComboBox3_Change()
Sheets("Folha1").Activate
'Vai à Folha1 ver qual é a ultima linha
lastrow = Cells(Rows.Count, "H").End(xlUp).Row
Sheets("Folha2").Activate
For m = lastrow + 4 To lastrow + lastrow + 3
   If Cells(m, "A") = "MoveC" Then Cells(m, "D") = ComboBox3.Value
Next m
End Sub

Private Sub ComboBox4_Change()
Sheets("Folha1").Activate
'Vai à Folha1 ver qual é a ultima linha
lastrow = Cells(Rows.Count, "H").End(xlUp).Row
Sheets("Folha2").Activate
For m = lastrow + 4 To lastrow + lastrow + 3
   If Cells(m, "A") = "MoveJ" Then Cells(m, "D") = ComboBox4.Value
Next m
End Sub

Private Sub ComboBox5_Change()
Sheets("Folha1").Activate
'Vai à Folha1 ver qual é a ultima linha
lastrow = Cells(Rows.Count, "H").End(xlUp).Row
Sheets("Folha2").Activate
For m = lastrow + 4 To lastrow + lastrow + 3
   If Cells(m, "A") = "MoveL" Then Cells(m, "D") = ComboBox5.Value
Next m
End Sub

Private Sub ComboBox6_Change()
Sheets("Folha1").Activate
'Vai à Folha1 ver qual é a ultima linha
lastrow = Cells(Rows.Count, "H").End(xlUp).Row
Sheets("Folha2").Activate
For m = lastrow + 4 To lastrow + lastrow + 3
   If Cells(m, "A") = "MoveC" Then Cells(m, "E") = ComboBox6.Value
Next m
End Sub
```

```
Private Sub CommandButton1_Click()
Call Get_Data
End Sub

Private Sub CommandButton2_Click()
Call Organizar
Call InsertRows
Call Arco
Call ProgRAPID
End Sub

Private Sub CommandButton3_Click()
Sheets("Folha2").Select
Cells.Select
Selection.ClearContents
Range("A1").Select
Sheets("Folha1").Select
Cells.Select
Selection.ClearContents
Range("A1").Select
End Sub

Private Sub CommandButton4_Click()
Call Save_and_edit
End Sub

Private Sub CommandButton5_Click()
Sheets("Folha1").Activate
'Vai à Folha1 ver qual é a ultima linha
lastrow = Cells(Rows.Count, "H").End(xlUp).Row
Sheets("Folha2").Activate
For i = 2 To lastrow + 1
   Cells(i, "D") = Cells(i, "D") + TextBox5.Value
   Cells(i, "E") = Cells(i, "E") + TextBox6.Value
   Cells(i, "F") = Cells(i, "F") + TextBox7.Value
Next i
End Sub

Private Sub CommandButton6_Click()
Sheets("Folha2").Activate
Cells(2, "C") = Cells(2, "C").Select
Sheets("Folha1").Activate
'Vai à Folha1 ver qual é a ultima linha
lastrow = Cells(Rows.Count, "H").End(xlUp).Row
Sheets("Folha2").Activate
ActiveCell.FormulaR1C1 = TextBox12.Text
   Selection.AutoFill Destination:=Range(Cells(2, "C"), Cells(lastrow + 1, "C")), Type:=xlFillDefault
For k = 2 To lastrow + 1
   Cells(k, "C") = Cells(k, "C") & ":=[["

 Next k


l = 0
For m = lastrow + 4 To lastrow + lastrow + 3
   If Cells(m, "A") = "MoveJ" Then Cells(m, "B") = Left(Cells(m - lastrow - 2 + l, "C"), Len(Cells(m -
lastrow - 2, "C")) - 4)
   If Cells(m, "A") = "MoveL" Then Cells(m, "B") = Left(Cells(m - lastrow - 2 + l, "C"), Len(Cells(m -
lastrow - 2, "C")) - 4)
```

```
    If Cells(m, "A") = "MoveC" Then Cells(m, "B") = Left(Cells(m - lastrow - 2 + l, "C"), Len(Cells(m -
lastrow - 2, "C")) - 4)
    If Cells(m, "A") = "MoveC" Then Cells(m, "C") = Left(Cells(m - lastrow - 1 + l, "C"), Len(Cells(m -
lastrow - 2, "C")) - 4)
    If Cells(m, "A") = "MoveC" Then l = l + 1

Next m
End Sub

Private Sub CommandButton7_Click()

Dim RetVal
RetVal = Shell("C:\WINDOWS\notepad.exe C:\Users\Ricardo\Os Meus
Documentos\Dissertacao\ImportCoordinateFile\exemplos\Help_G-RAPID.txt", 1)

End Sub
```