M 2019

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

DESENVOLVIMENTO DE UM MÉTODO DE PROGRAMAÇÃO DE ROBÔS INDUSTRIAIS COM RECURSO A UM BRAÇO DE METROLOGIA

MARIA CATARINA PEREIRA DE CASTRO NEVES
DISSERTAÇÃO REALIZADA NO ÂMBITO DO MESTRADO INTEGRADO EM ENGENHARIA METALÚRGICA E DE MATERIAIS

ORIENTADOR
PROFESSOR VITOR MANUEL BRANCO MARTINS AUGUSTO
PROFESSOR NO DEPARTAMENTO DE ENGENHARIA METALÚRGICA E DE MATERIAIS

U.PORTO

FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

CANDIDATO	MARIA CATARINA PEREIRA DE CASTRO NEVES	***CÓDIGO***	201404670
TÍTULO	DESENVOLVIMENTO DE UM MÉTODO DE PROGRAMAÇÃO DE ROBÔS INDUSTRIAIS COM RECURSO A UM BRAÇO DE METROLOGIA		
DATA	22 DE JULHO DE 2019		
LOCAL	F106 – DEMM - FEUP		

JÚRI

PRESIDENTE	MANUEL FERNANDO GONÇALVES VIEIRA	DEMM - FEUP
ARGUENTE	CARLOS ALBERTO MOURA RELVAS	DEM - UA
ORIENTADOR	PROFESSOR VITOR MANUEL BRANCO MARTINS AUGUSTO	DEMM - FEUP

Alguns homens vêem as coisas como são, e dizem 'Por quê?' Eu sonho com as coisas que nunca foram e digo 'Por que não?'.

- George Bernard Shaw

RESUMO

O atual mercado tecnológico impele as empresas na procura pela melhoria dos seus processos e inovação dos seus métodos de trabalho. Neste contexto, a presente dissertação tem como objetivo o desenvolvimento de uma nova metodologia de programação de robôs, recorrendo a um braço de metrologia, de forma a otimizar todo o processo industrial.

Para tal, foi necessário estudar, *à priori*, as metodologias utilizadas atualmente no mercado, programação *online* e *offline*, e o modo como estas potenciam o fluxo da produção. Nesse âmbito, foi recriada uma operação de perfilagem, de forma a estabelecer a comparação entre as duas metodologias. Foi possível constatar que ambos se revelaram morosos por diferentes razões: a programação *online* exige a concentração do colaborador por grandes períodos de tempo e a programação *offline* exige um colaborador qualificado no *software* CAM em utilização, o que se traduz num maior investimento. Contudo, o último apresenta um programa final com maior qualidade.

O novo método de programação, *teach & learn offline*, foi desenvolvido com base nos pontos fortes de cada uma das metodologias anteriormente estudadas. A adição de um braço de metrologia à metodologia da programação permitiu simplificar o processo e diminuir o tempo despendido na operação.

No futuro, perspetiva-se a maior autonomia deste novo processo, assim como a minimização dos custos associados.

PALAVRAS-CHAVE

Robot Programming; Online Programming; Offline Programming; NC program; PowerMill

ABSTRACT

Today's technological market stimulates companies to look for improvement in their own procedures, together with innovative working methods. Taking into account this context, this thesis aims to develop a new robot programming methodology, by using a metrology arm, in order to optimize the whole industrial procedure.

To achieve this outcome, it was necessary to study, theoretically, the methodologies available on the market, the online and offline programs, and the way these increase the production flow. Within this context, a trim operation was set, so as to establish a comparison between these two methodologies. It was possible to realize that both are time consuming, due to different causes: online programming demands the operator's attention span for large periods of time and offline programming demands a software CAM qualified operator, which means a higher investment. However, this last solution allows a final product with better quality The new programming method teach & learn offline was developed, bearing in mind the strengths of each previously studied methodology.

Adding up a metrology arm to the programming methodology allowed the simplification of the process and the reduction of time used in this operation.

In the future, it is predictable that this process will have higher autonomy as well as decreasing related costs.

KEY- WORDS

Robot Programming; Online Programming; Offline Programming; NC program; PowerMill

AGRADECIMENTOS

A elaboração desta dissertação não seria possível sem o apoio e incentivo de certas pessoas às quais me sinto extremamente grata, pela ajuda que me deram, ao longo deste semestre.

Ao Professor Vitor Martins Augusto, pela sua orientação, amabilidade e disponibilidade.

Ao Engenheiro Bruno Couto, por toda a paciência e dedicação.

Aos restantes Engenheiros da Norcam, por me terem recebido tão bem; especialmente ao Engenheiro Pedro Castro, pela oportunidade que me facultou, tornando possível a realização da minha dissertação em ambiente empresarial.

Aos meus pais, por me porem sempre com “os pés na terra”.

Aos meus amigos, que sem dúvida foram o meu maior apoio, nos últimos tempos, e me fizeram trabalhar com convicção.

ÍNDICE

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

CAD – *Computer Aided Design*

CAM – *Computer Aided Manufacturing*

NC – *Numeric Control*

PRI – *PowerMill Robot Interface*

1. INTRODUÇÃO

Este documento foi redigido no âmbito da dissertação em ambiente empresarial do Mestrado Integrado em Engenharia Metalúrgica e de Materiais da Faculdade de Engenharia da Universidade do Porto em parceria com a empresa Norcam – Engenharia e Design Industrial.

1.1. NORCAM

A Norcam – Engenharia e Design Industrial é uma empresa criada em 1991 que desenvolve e implementa soluções industriais, assim como sistemas CAD/CAM/CAE, impressão 3D, digitalização tridimensional, células robóticas e outros meios tecnológicos avançados de uma forma integrada, de forma a desenhar soluções para os problemas mais complexos [1].

1.2. MOTIVAÇÃO E OBJETIVOS DO PROJETO

Globalmente, a sociedade de consumo está em permanente e muito rápida mutação e a indústria transformadora está a enfrentar grandes desafios para dar resposta à escassez de recursos, à falta de qualificação da mão de obra, ao envelhecimento social e à cada vez maior procura por produções locais. Atendendo à procura de certos requisitos do consumidor, desde produtos mais baratos com melhor qualidade, em maiores quantidades, as diferentes indústrias de produção viram-se obrigadas a apostar mais na evolução tecnológica, nomeadamente no que diz respeito a equipamentos e *software* [2].

A resposta a estes desafios está a fazer-se essencialmente com o recurso à automação flexível de robôs industriais, através do aumento significativo no número de robôs colocados na indústria, com um número crescente em novas áreas de aplicação. Esta adaptabilidade permite que as diferentes indústrias apresentem um ciclo de desenvolvimento do produto rápido e eficaz, tornando-as competitivas e atuais [2].

A indústria da robótica está em pleno crescimento, sendo que em 2017 atingiu o nível mais alto do volume de vendas anual alguma vez registado. Aproximadamente 381.000 robôs industriais foram vendidos mundialmente, mais 30% de unidades em comparação com 2016 [2].

A indústria de transformação global ultrapassou o limiar de dois milhões de robôs em serviço pela primeira vez. Em 2017 foram utilizados cerca de 2.1 mil robôs industriais em diversas operações, mais 15% unidades em comparação com 2016, sendo que o principal impulsionador deste fenómeno foi a China [2].

Estima-se que este crescimento continue e que, entre 2018 e 2021, mais de dois milhões de novos robôs industriais sejam instalados em fábricas em todo o mundo. As vendas anuais globais totais atingirão mais de 600 mil unidades em 2021, com a Ásia ainda no topo, seguida da Europa e da América do Norte [2].

No futuro, novas aplicações potenciais serão desbloqueadas e as possibilidades de aplicação de robôs industriais aumentarão ainda mais. Os desenvolvimentos tecnológicos em robótica estão a progredir a um ritmo acelerado [2].

Desta forma, o objetivo deste trabalho assentou na necessidade de facilitar o próprio processo de programação de robôs, para que novas indústrias vejam potencialidade de executar os seus procedimentos com recurso a este tipo de equipamentos. Para tal, pretendeu-se desenvolver um novo método que exija menos dos seus colaboradores mantendo ou melhorando a qualidade implementada na estratégia a utilizar.

Tendo em conta o supracitado, surge a motivação para a realização deste projeto, o qual consistiu no desenvolvimento de um novo *software* para programação de robôs através da utilização de um braço de metrologia. A concretização deste compreendeu também o estudo da aprendizagem dos métodos de programação de robôs atualmente utilizados no mercado, *online* e *offline*, de modo a avaliar os pontos em falha e melhorar este processo na cadeia de valor.

1.3. ESTRUTURA DA DISSERTAÇÃO

A presente dissertação encontra-se estruturada em sete capítulos. No primeiro capítulo, de carácter introdutório, foi realizado um enquadramento da empresa e definido as diretrizes para a motivação e objetivos do trabalho.

O segundo capítulo pretendeu-se explanar o enquadramento teórico dos conceitos e ferramentas utilizadas na prossecução do objetivo proposto.

O terceiro capítulo foi dedicado à contextualização da programação *teach & learn offline*, com o pressuposto de otimizar todo o processo.

O quarto capítulo defineu o procedimento experimental adotado, baseando-se na comparação entre dois métodos de programação.

O quinto capítulo apresenta uma análise e discussão dos resultados repercutidos pela metodologia descrita no capítulo anterior.

O sexto capítulo focou-se na prossecução da solução desenhada do desenvolvimento da programação *teach & learn offline*, através da definição de uma estratégia inovadora. Adicionalmente, são definidos os trabalhos futuros.

O sétimo capítulo apresenta as conclusões do desenvolvimento desta dissertação.

2. ENQUADRAMENTO TEÓRICO

Este capítulo tem como objetivo contextualizar as ferramentas utilizadas na investigação e desenvolvimento do novo método de programação de robôs.

2.1. ROBÔS INDUSTRIAIS

Segundo a *Robotic Industries Association* (RIA), organização sem fins lucrativos, um robô industrial é definido como um "manipulador multifuncional reprogramável projetado para movimentar materiais, partes, ferramentas ou peças especiais, através de diversos movimentos programados, para o desempenho de uma variedade de tarefas" [3].

Uma definição mais completa é apresentada pela *International Organization for Standardization*, na ISO 8373:2012. Esta define um robô como "uma máquina manipuladora com vários graus de liberdade controlada automaticamente, reprogramável, multifuncional, que pode ter base fixa ou móvel para utilização em aplicações de automação industrial" [3].

Os robôs são, geralmente, projetados para executar funções repetitivas, desagradáveis ou perigosas para o utilizador. A sua implementação na indústria tem como objetivo complementar o trabalho das máquinas CNC, devido à grande amplitude de movimentos que definem os robôs [4, 5].

As vantagens de implementação de robôs em ambientes industriais podem ser divididas em três categorias [6]:

1. Técnicas: no que diz respeito à versatilidade de estratégias que podem vir a ser programadas, superando as capacidades de execução do ser humano, uma vez que garantem maior flexibilidade;
2. Económicas: um único robô consegue desempenhar diferentes funções/estratégias, o que pode permitir um aumento da produtividade que, por sua vez, pode vir a trazer benefícios de retorno do investimento efetuado;
3. Sociológicas: ao utilizar robôs vai-se retirar funcionários de certos ambientes perigosos e permitir que estes deixem de executar tarefas pesadas e/ou desagradáveis, que por sua vez levará à redução do risco de acidentes.

2.1.1. COMPONENTES

Um robô industrial é uma estrutura mecânica formada pela integração dos seguintes componentes [7]:

1. Órgão terminal: dispositivo afixado ao último eixo do robô de forma a anexar elementos para serem manipulados;

2. Manipulador: conjunto de corpos (braços) ligados entre si por juntas que permitem a cinemática do robô e definem a estrutura mecânica do robô;
3. Sensores: componentes responsáveis por avaliar a atuação do robô. Existem sensores internos, responsáveis por fornecer informação sobre a posição, velocidade e aceleração do robô, e sensores externos, como por exemplo câmaras de vídeo para deteção de colisões, responsáveis por fornecer informações sobre o ambiente em redor do robô;
4. Controlador: dispositivo que controla todos os movimentos do manipulador;
5. Unidade de potência: elemento que tem como objetivo fornecer energia aos vários componentes do robô;

2.1.2. CLASSIFICAÇÃO

Pode definir-se um robô, também, como um equipamento composto por um conjunto de corpos conectados em série por juntas, estabelecendo os seus eixos. Geralmente, um manipulador tem entre cinco a seis eixos, formando duas extremidades: uma extremidade fixa ao chão (a base) e uma extremidade livre (o atuador), onde podem ser aplicados vários tipos de ferramentas, consoante a tarefa pretendida, como se pode observar na Figura 1 [8, 9].

As três primeiras juntas de um robô manipulador são associadas a um cotovelo humano, tendo em conta a geometria e posicionamento das mesmas. O objetivo das três juntas referidas são o de posicionar e orientar o resto da estrutura do robô [10].

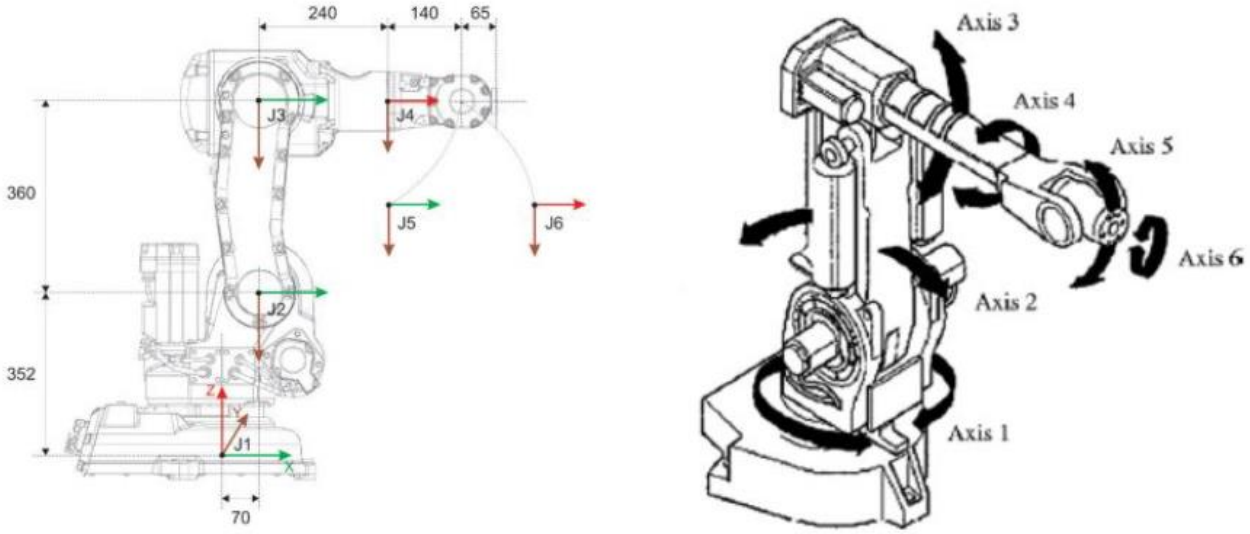

Figura 1. Sistema de coordenadas e robô industrial de 6 eixos [3].

Os robôs industriais são classificados relativamente à configuração das três primeiras juntas. Desta forma, surgem cinco tipos de robôs diferentes, tendo em conta a distribuição dos seus eixos e o tipo de coordenadas utilizado. Os cinco tipos estão representados na Figura 2 e são os seguintes [9, 11]:

1. Cartesiano: este tipo de robô apresenta três juntas prismáticas (PPP) perpendiculares entre si. Estas resultam num movimento composto por três translações, cujos movimento são descritos com base num sistema de coordenadas cartesiano;
2. Cilíndrico: este tipo de robô apresenta a uma junta rotacional e duas são prismáticas (PPR). Estas resultam num movimento de duas translações e uma rotação, sendo que os seus movimentos são descritos com base no sistema de coordenadas cilíndrico. Apresenta velocidades elevadas e grande capacidade de carga, no entanto o seu alcance é limitado;
3. Esférico: este tipo de robô apresenta duas juntas rotacionais e perpendiculares entre si e uma prismática (PRR). Estas resultam num movimento de uma translação e duas rotações. Apresenta grande alcance, com precisão mediana, aliando a alta velocidade à capacidade de carga. Contudo o seu desempenho pode ficar comprometido;
4. SCARA: este tipo de robô é semelhante ao tipo esférico (PRR), sendo que apenas as suas duas primeiras juntas é que são paralelas entre si. Estas resultam num movimento também de uma translação e duas rotações. A principal diferença entre estes dois tipos de robôs está na maior precisão que este tipo apresenta. Para além disso este também é caraterizado pela sua elevada velocidade e repetibilidade, no entanto o seu alcance é limitado;
5. Antropomórfico: também conhecido como robô articulado. Este tipo de robô apresenta três juntas de revolução, sendo a primeira perpendicular às duas seguintes. Devido à sua configuração, este tipo apresenta a maior mobilidade, podendo executar uma maior variedade de trabalhos. Este é utilizando quando necessário substituir trabalho humano. Apresenta alta precisão e velocidade.

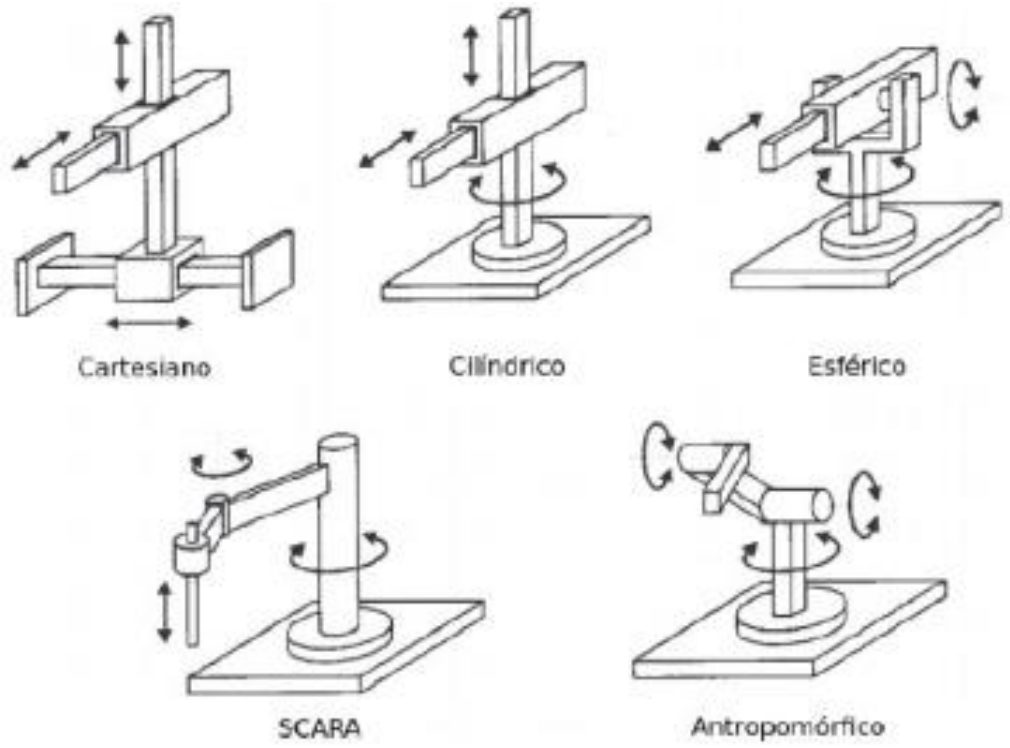

Figura 2. Tipos de robôs industriais [7].

2.2. CINEMÁTICA

O conjunto de movimentos que o robô executa designa-se por cinemática. O estudo desta está relacionado com as ferramentas matemáticas necessárias à movimentação das juntas e elos, de forma a que uma estratégia seja executada.

2.2.1. ÂNGULOS EULER

Os ângulos Euler são responsáveis pela orientação dos diferentes elos presentes no robô. Estes têm como principal objetivo descrever a orientação de um corpo rígido através de três ângulos.

Os ângulos Euler correspondem à seguinte sequência de rotações, tal como demonstra na Figura 3:

1. Rotação de um ângulo ∅ em torno do eixo z $(R_{z,\emptyset})$;
2. Rotação de um ângulo θ em torno do eixo y $(R_{y,\theta})$;
3. Rotação de um ângulo φ em torno do eixo z $(R_{z,\varphi})$.

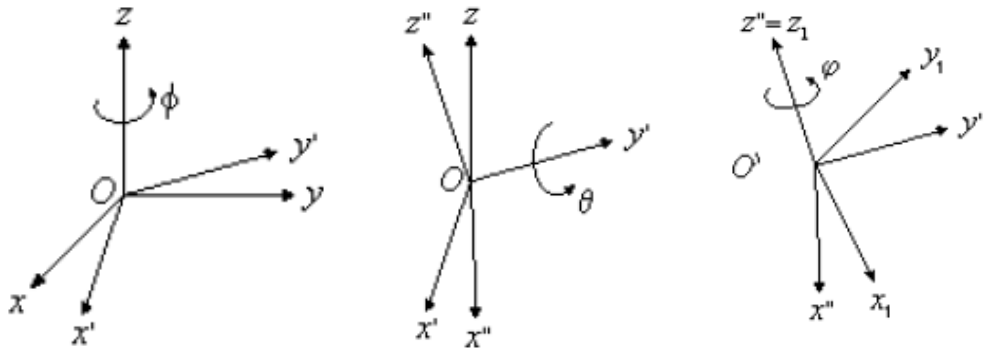

Figura 3. As três rotações consecutivas utilizadas para definir os ângulos de Euler.

Considerando as três rotações no sistema O_{xyz} esquematizadas na Figura 3. Primeiramente, o sistema sofre uma rotação no eixo z de um ângulo $\emptyset$. Em seguida, o novo sistema de coordenadas $O_{x'y'z}$ sofre uma rotação no eixo y' de um ângulo θ. Finalmente, o novo sistema $O_{x''y''z'}$ sofre uma rotação no eixo z'' de um angulo φ, resultando no sistema de coordenadas O_{x1y1z1}. Os três ângulos, $\emptyset, \theta$ e φ, determinam a orientação do novo sistema de coordenadas de forma única, e são os ângulos Euler. É importante que a sequência apresentada seja mantida, pois os resultados serão diferentes dos desejados, caso tal não se verificar.

Estas rotações representam uma transformação de coordenadas relacionando as coordenadas de um ponto P em dois sistemas com origem coincidente em um mesmo ponto (Figura 4).

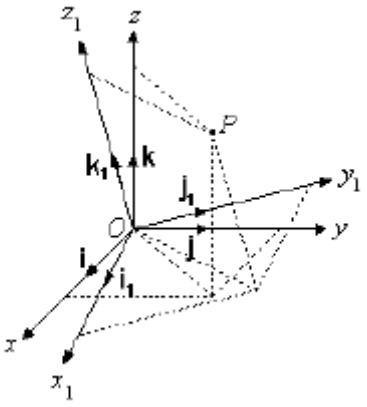

Figura 4. Transformação de rotação.

Para obter as rotações (os ângulos $\emptyset$, θ e φ) desejadas matematicamente é necessário conhecer os vetores i, j e k pré rotação e os vetores i, j e k pós rotação.

2.2.2. MOVIMENTOS REALIZADOS PELO ROBÔ

Considerando um robô antropomórfico que tem uma maior flexibilidade e um maior espaço operacional, é possível verificar que este executa quatro tipos de movimento diferentes: movimento de junta, movimento linear, movimento circular e movimento em parábola. Estes movimentos serão explicados nos seguintes aspetos.

O movimento de junta (MOVJ) é utilizado quando o manipulador não necessita de se mover num caminho específico até à próxima posição a executar podendo, desta forma, evitar singularidades. Este é, normalmente, o primeiro movimento a ser ensinado e executado pelo robô, por razões de segurança e que serão explicadas mais abaixo [12].

O movimento linear (MOVL) executa o trajeto mais curto entre posições, no entanto, pode apresentar singularidades. O robô desloca-se automaticamente, mudando a posição do pulso, como demonstra na Figura 5 [12].

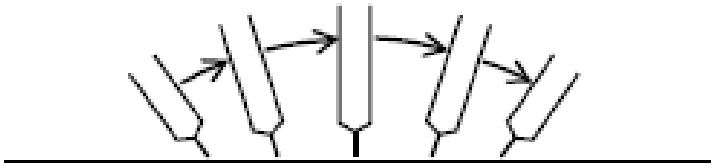

Figura 5. Movimento linear, MOVL [12].

O movimento circular (MOVC) é utilizado quando um movimento circular é necessário entre três pontos; e o movimento em parábola (MOVS) é utilizado quando um movimento em oval é necessário entre três pontos. Este último tipo de movimento é particularmente interessante ao realizar certas operações como, por exemplo, soldadura ou corte, devido às geometrias irregulares que as peças a trabalhar podem vir a apresentar [12].

2.2.3. SINGULARIDADES

As singularidades acontecem quando o posicionamento dos eixos do robô leva a que o mesmo não consiga atingir uma posição específica e, consequentemente, há uma paragem (fim de percurso). Ou quando existem demasiadas possibilidades para a nova posição ou, ainda, quando o próximo movimento requer que o robô se mova a velocidade infinita.

Existem três tipos principais de singularidades, tal como demonstra a Figura 6 [7, 13]:

1. Tipo A – singularidade de cotovelo que acontece quando existe alinhamento entre os eixos 2, 3, 4 e 5;
2. Tipo B – singularidade de alinhamento quando o eixo 1 está em linha com o eixo 6;

3. Tipo C – singularidade de pulso quando existe alinhamento entre os eixos 4 e 6.

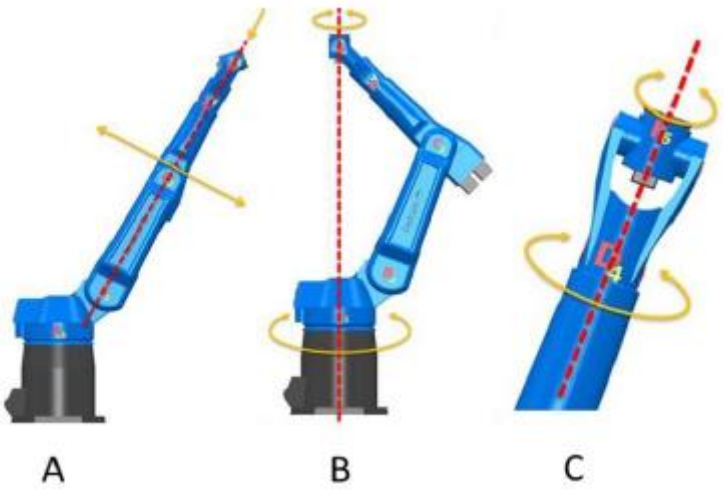

Figura 6. Tipos de singularidades dos robôs industriais [7].

2.3. APLICAÇÕES INDUSTRIAIS

Por mais que a robótica esteja a ser aplicada cada vez mais em diferentes industrias, é possível agrupar robôs industriais em três categorias [14]:

1. Manuseio, transporte e operações de carregamento e descarregamento: o braço do robô vai movimentar algo entre dois ou mais pontos;
2. Aplicação de processamento: o braço do robô, com a ajuda de ferramentas especiais, vai aplicar transformações numa peça;
3. Montagem e inspeção: o braço do robô realiza montagem mecânica ou eletrónica, além de inspeção de peças através de sensores especiais. Em ambos os casos este vai permitir a garantia de qualidade.

Dentro da categoria "Aplicação de processamento", é possível definir que as suas aplicações consistem em: soldadura por pontos, soldadura por arco elétrico, pintura por spray, corte jato de água, colas e vedantes, maquinagem leve, furação e perfilagem.

2.4. PROGRAMAÇÃO DE ROBÔS

Tal como mencionado no capítulo "2.1 Robôs Industriais", um robô industrial é considerado uma máquina automática programável. A sua programação está diretamente relacionada com a capacidade de o robô funcionar sem intervenção humana direta. Esta

implica a utilização de sistemas mecânicos, elétricos e/ou computadorizados para o seu comando e controlo [3].

A programação de robôs tem como objetivo criar um programa NC (*Numeric Control*) de maneira simples, de forma que o robô consiga interpretar a estratégia pensada. Nesta vão estar presentes as localizações que o robô terá de percorrer no espaço com uma determinada orientação, velocidade e aceleração, de modo a concluir a trajetória [6].

Atualmente, os robôs podem ser programados por dois métodos diferentes. A escolha de um destes é efetuada tendo em conta a aplicação para a qual o robô é utilizado. Para caminhos simples, o robô é, normalmente, programado *online*, enquanto que para caminhos mais complexos, o robô é programado *offline* [6].

2.4.1. PROGRAMAÇÃO *ONLINE*

A programação *online* de robôs é o método de programação mais simples de aprender e executar.

A aprendizagem de um novo programa é realizada diretamente no robô, pelo que também, poderá ser chamada de programação t*each & learn*. O utilizador através dos botões presentes no comando do controlador, vai deslocar o robô até que este esteja posicionado no ponto desejado, ao atingir esse ponto vai registá-lo e avançar para o seguinte [15].

Na Figura 7, é possível observar o comando do controlador do robô *Yaskawa Motoman* GP25, onde este método de programação é realizado. No capítulo "4.1.3 Programação *Online*" será realizado um planeamento e análise mais detalhada de uma trajetória criada.

Este é um método muito moroso que exige muita concentração e experiência por parte do utilizador. O programador tem de estar atento a muitos aspetos inerentes à programação, desde o conhecimento do comando do controlador do robô, até à verificação da calibração da ferramenta a utilizar.

Normalmente, este método costuma ser utilizado para percursos simples e pequenos de modo a evitar fadiga por parte do utilizador, nomeadamente em soldaduras.

Figura 7. Comando do controlador do robô *Yaskawa Motoman* GP25.

Na Tabela 1 apresentou-se uma análise SWOT deste tipo de programação.

Uma análise SWOT é um sistema simples para posicionar ou verificar a posição estratégica deste tipo programação, no ambiente em questão. O termo SWOT é um acrónimo de forças (*strengths*), fraquezas (*weaknesses*), oportunidades (*opportunities*) e ameaças (*threats*).

Tabela 1. Análise SWOT programação *online*.

FORÇAS (S)	**FRAQUEZAS (W)**
• Programar caminhos simples (linhas retas com um número reduzido de pontos); • Não exige *software* CAM; • Não exige desenho CAD; • É possível orientar a ferramenta da maneira pretendida para a operação.	• Programar caminhos complexos (zonas com geometrias complexas que exijam muitos pontos) – Pode vir a consumir muito tempo; • Potenciais desvios de trajetória – imperfeições; • Exige operador com conhecimento para manipular o robô; • Exige operador atento para que o robô não embata na peça a maquinar (o programa a realizar pode exigir pontos de aproximação); • Exige operador atento para que a ferramenta seja colocada da melhor forma; • O robô não está a produzir valor enquanto o operador cria o programa; • O programa obtido é muito simples (contem informação do plano de trabalho, da ferramenta, tipo de movimento e velocidade); • Caso seja preciso alterar um parâmetro no programa (por exemplo velocidade) é necessário alterá-lo ponto a ponto; • Não apresenta precisão, pelo que é necessário acrescentar depois de criar o programa; • Grande investimento no robô.
OPORTUNIDADES (O)	**AMEAÇAS (T)**
• Utilização crescente de robôs pela indústria; • Novas indústrias e aplicações a explorar; • Maior facilidade de formar técnicos.	• Melhores tipos de programação (por exemplo *offline*) que não obrigam o robô a parar de produzir valor e que programam o robô em menor tempo; • A crescente procura por produtos pode não ser correspondida; • Evolução de equipamentos (ex: de maquinagem) que não exijam tanto investimento ou formação; • Trabalho manual ou com recurso a outras máquinas ferramenta (por exemplo CNC) pode ficar menos dispendioso.

2.4.2. PROGRAMAÇÃO *OFFLINE*

Atendendo à procura de certos requisitos do consumidor e à necessidade de produzir peças com geometrias complexas, as diferentes indústrias viram-se obrigadas a evoluir de forma a apresentar um ciclo de desenvolvimento do produto rápido e eficaz. Esta evolução está associada à criação de *software* CAD e CAM (*Computer Aided Design* e *Computer Aided Manufacturing*), que aliam a capacidade de desenvolver e processar produtos, independentemente da complexidade que estes possam apresentar [4, 16]. Estes tipos de *software* fazem ligação de toda a cadeia de valor de produção de um produto [7].

A programação *offline* de robôs é realizada com o auxílio de um *software* CAM tridimensional que através de um modelo CAD 3D da peça a trabalhar, cria a trajetória desejada. Após o planeamento desta (demonstrado na Figura 8) o programa é exportado em formatos padronizados ou específicos em função do robô a utilizar, de modo a poder ser realizado [17].

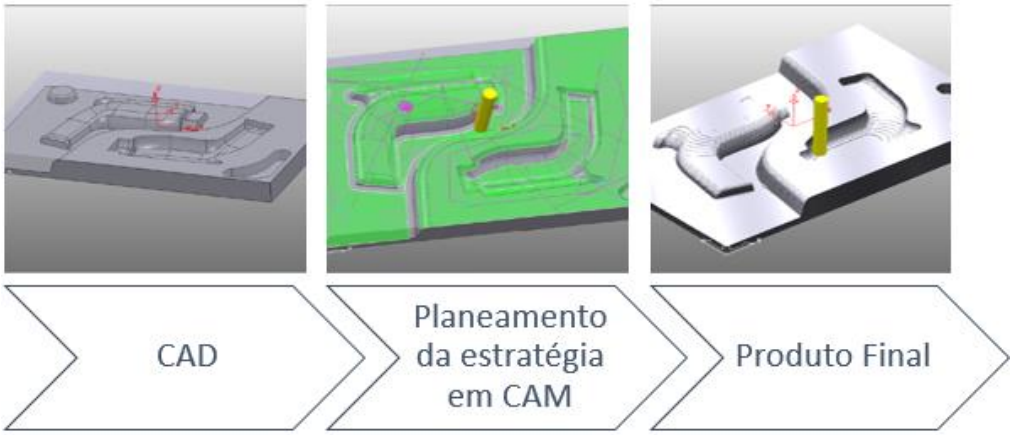

Figura 8. Processo em cadeia CAD-CAM [18].

Esta metodologia revela-se interessante, uma vez que o robô não necessita de interromper o seu funcionamento, para que seja criada uma nova programação do mesmo. Como esta é criada através de um computador pessoal, o robô continua a produzir.

Os custos associados à energia que o robô consome durante a sua utilização passam a estar todos relacionados com custos de operação e não de programação, o que não acontece na programação *online*.

Em contrapartida, este método exige a compra e subscrição de um *software* CAM, bem como a formação dos seus utilizadores, o que faz com que seja mais dispendioso. De salientar, que a formação dos utilizadores apresenta um caráter relevante na escolha deste método de programação. Estes tipos de *software* contêm muita informação de forma a dar resposta a todas as possíveis estratégias atualmente presentes no mercado. Desta forma, é necessário que o utilizador conheça o caminho a efetuar, isto é, saber como criar

e que estratégias deve implementar para a operação requerida, de modo a realizá-las no menor tempo possível.

A minimização de tempo de planeamento, programação e execução de estratégias é essencial, devido à alta competitividade presente nas indústrias atuais. Os *software* CAM permitem planear estratégias mais suaves e que com tempos de operação reduzidos [17].

Muitos dos benefícios da escolha deste método de programação provêm da simulação que os *software* CAM fornecem. Tendo em conta a escolha do local a colocar a peça a trabalhar, o tipo e durabilidade das ferramentas, as estratégias escolhidas e a otimização de parâmetros de operação, a simulação permitirá otimizar ao máximo a trajetória sem a necessidade de se produzir testes no robô [19].

Durante a simulação, é possível observar a trajetória que o robô irá realizar. Isto porque os *software* CAM presentes no mercado fornecem uma representação a três dimensões das operações a realizar, tal como é possível observar na Figura 9, onde é observada uma extensão do PowerMill, o PowerMill *Robot Interface*.

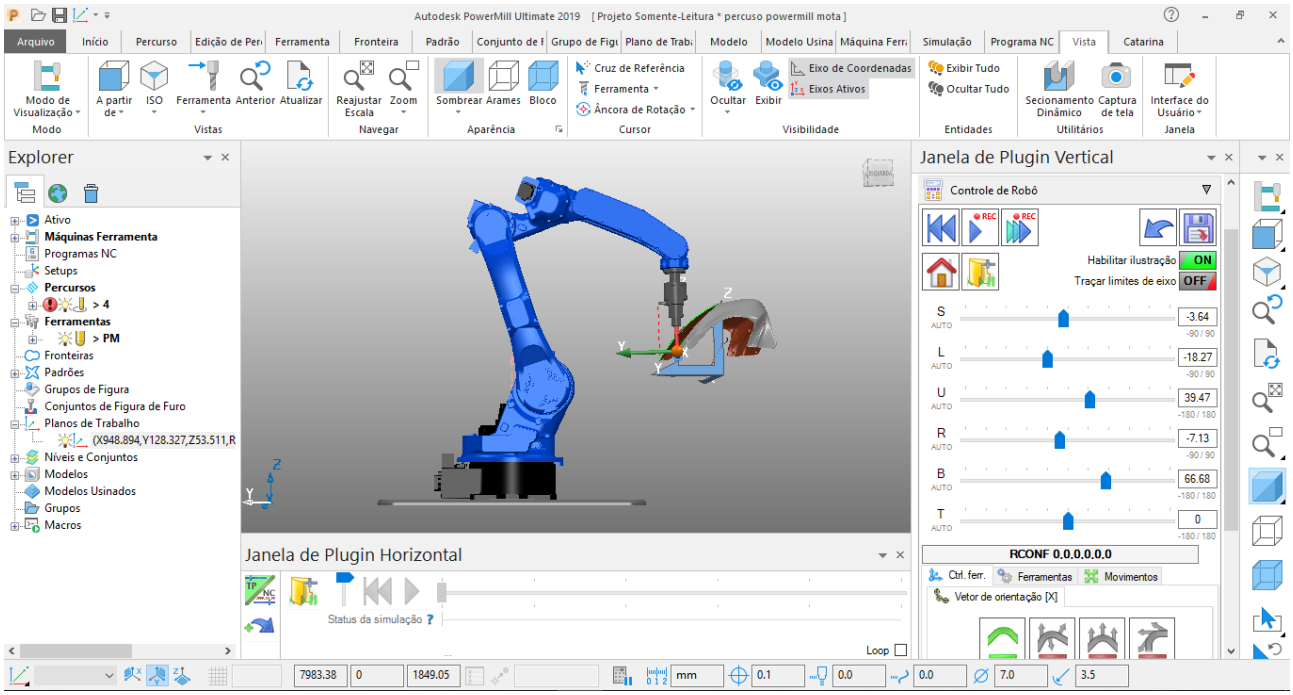

Figura 9. Exemplo de uma simulação realizada no *software* PowerMill.

Esta permite observar os movimentos que o robô irá realizar e avaliá-los de forma a evitar problemas de segurança. Para além disso, também permitirá saber o tempo total real de operação, de uma forma rápida. Em alguns segundos, é possível verificar a veracidade da trajetória gerada.

Atualmente, é possível encontrar no mercado inúmeros *software* CAM que satisfaçam as necessidades dos seus utilizadores, como por exemplo o Mastercam,

HSMWorks (Non Fusion 360) e o PowerMill. Estes destacam-se por serem os *software*s CAM mais vendidos no ano de 2016 [20].

Mais sobre este tipo de programação encontra-se na Tabela 2 onde foi realizada uma análise SWOT deste tipo de programação.

Tabela 2. Análise SWOT programação *offline*.

FORÇAS (S)	**FRAQUEZAS (W)**
• Programa completo (contem informação do plano de trabalho, da ferramenta, tipo de movimento, velocidade, precisão, aceleração e desaceleração); • É possível simular no *software* de forma a perceber se há algum erro (por exemplo colisões do robô na peça ou singularidade); • Programa caminhos complexos variados (por exemplo desbaste em Z constante, 3D offset, entre outros); • Permite escolher o vetor orientação da ferramenta; • Permite aumentar a precisão em certas áreas do produto a operar.	• Exige *software* CAM; • Exige modelo CAD 3D; • Programa completo exige conhecimento específico; • Exige utilizadores com formação no *software* utilizado; • Exige utilizadores com conhecimento para manipular o robô; • Exige um grande investimento: robô, *software* (compra e subscrição) e formação de utilizadores.
OPORTUNIDADES (O)	**AMEAÇAS (T)**
• Utilização crescente de robôs pela indústria; • Novas indústrias e aplicações a explorar; • Com a Indústria 4.0 os sistemas tendem a vir a ser totalmente integrados.	• Evolução de máquinas-ferramentas que não exijam tanto investimento ou formação; • Criação de novos tipos de programação (por exemplo que não obriguem a formação em *software*); • A crescente procura por produtos pode não ser correspondida; • Trabalho manual ou com recurso a outras máquinas ferramenta (por exemplo CNC) pode ficar mais em conta.

3. PROGRAMAÇÃO *TEACH & LEARN OFFLINE*

O presente capítulo pretende apresentar um projeto de melhoria dos métodos de programação atualmente utilizados. A programação *teach & learn offline* surge na tentativa de revolucionar a programação de robôs industriais, tendo como objetivo facilitar e diminuir os custos associados ao processo de programação.

Este processo foi designado de *teach & learn offline*, pois utiliza as vantagens características de cada método de programação atualmente presentes no mercado.

Uma estratégia é criada tendo em conta três parâmetros:

1. As coordenadas dos pontos-chave, tendo em conta um sistema de coordenadas global;
2. A velocidade de operação;
3. Orientação que a ferramenta deve apresentar.

No caso da programação de robôs, os pontos-chave de uma trajetória são memorizados, tendo em conta o espaço de atuação do robô. No método *online* estes são captados diretamente no robô e no método *offline* são captados através do desenho CAD e um *software* CAM que seja capaz de produzir o programa para o robô em questão (que contenha o pós processador do robô a utilizar). Desta forma, para realizar a programação de uma trajetória, apenas é necessário um mecanismo que seja capaz de memorizar coordenadas.

Assim, surgiu a ideia da utilização de um braço de metrologia, para a captura dos pontos da trajetória, que permita criar um percurso real para a peça em questão, sem a necessidade de se recorrer ao desenho CAD. Para além disso, estaria a ser realizada uma programação de aprendizagem, sem necessidade de se utilizar o robô.

Deste modo, este novo método permitirá que utilizadores com menos experiência sejam capazes de produzir um programa NC, facilitando todo o processo, uma vez que não é necessário conhecer intrinsecamente um *software* CAM para o criar.

Um braço de metrologia articulado, como exemplificado na Figura 10, permite que a medição das coordenadas do percurso seja realizado quando a sua sonda toca no objeto. As coordenadas angulares são adquiridas segundo um sistema de coordenadas cartesiano (x, y, z) [21].

Este tipo de braços têm como principal vantagem a sua mobilidade, pelo que podem ser colocados em qualquer tipo de lugar [21]. Desta forma, não há necessidade de o colocar junto do robô, o que poderia dificultar outras estratégias pré-existentes. Porém, é de salientar, que a localização de colocação do braço tem de ser cuidada. Como é um braço de metrologia, este é muito sensível à vibração, pelo que o ideal seria criar a estratégia num ambiente controlado [22].

Figura 10. Braço de metrologia - RPS Metrology EVO7 [22].

Através do braço, o utilizador deve ser capaz de memorizar a localização e a orientação em relação à qual o ponto é memorizado, ou seja, as coordenadas e o vetor ijk correspondente.

De forma a criar um programa que o robô consiga ler, é necessário correlacionar as coordenadas obtidas com o braço de metrologia com as do robô. Para tal, deve-se desenvolver um *software* capaz de correlacionar as coordenadas do braço com as do robô e transformar os vetores ijk obtidos em ângulos Euler.

Mais sobre este tipo de programação encontra-se na Tabela 3 onde foi realizada uma análise SWOT deste tipo de programação.

Tabela 3. Análise SWOT programação *Teach & learn offline*.

FORÇAS (S)	**FRAQUEZAS (W)**
• Não exige desenho CAD; • Facilidade em definir o percurso de maquinagem através do braço; • *Software* de interface simples (userfriendly).	• Exige *software* CAM específico; • Exige operador com conhecimento para manipular o robô; • Exige um grande investimento (robô + *software* + braço); • Não tem em conta a precisão dos movimentos a utilizar; • Dificuldade em localizar o plano de trabalho; • O programa obtido é muito simples (contem informação do plano de trabalho, da ferramenta, tipo de movimento e velocidade); • Exige operador atento para que o movimento que realiza com o braço, ao ser replicado pelo robô, não colida com a peça a maquinar (pontos a retirar podem exigir que sejam pontos de aproximação); • Visualização do trajeto criado no *software* em 2D.
OPORTUNIDADES (O)	**AMEAÇAS (T)**
• Utilização crescente de robôs pela indústria; • Novas indústrias e aplicações a explorar; • Permite lidar com alterações à geometria inicial da peça.	• Evolução de máquinas-ferramentas que não exijam tanto investimento; • A crescente procura por produtos pode não ser correspondida; • Trabalho manual ou com recurso a outras máquinas ferramenta (por exemplo CNC) pode ficar menos dispendioso.

4. PROCEDIMENTO EXPERIMENTAL

Para perceber o que seria necessário desenvolver no método de programação *teach & learn offline*, primeiro foi preciso compreender a nível pratico, como funcionam os métodos utilizados pela indústria.

Desta forma, o trabalho experimental realizado teve como objetivo inicial a comparação dos dois métodos de programação de robôs, tradicionalmente utilizados (*online* e *offline*). Para tal, foi escolhida uma estratégia a realizar em ambos os métodos, com o intuito de comparar as dificuldades de programação, tempos, desvantagens e benefícios de cada um.

4.1.1. PEÇA E ESTRATÉGIA

A peça em estudo é a carenagem frontal de uma mota, como é possível observar na Figura 11. Esta é uma peça polimérica (ABS) produzida por injeção. Após fabrico a carenagem apresenta um bom acabamento superficial, com a exceção das extremidades (bordos). Estas necessitam de ser submetidas à operação de perfilagem, ou seja, submetidas a uma operação de corte que permitirá a remoção da rebarba de injeção, de forma a obter a geometria final especificada pelo cliente.

Figura 11. Objeto de estudo - carenagem frontal de uma mota.

A estratégia a programar para realizar a operação de perfilagem está representada a vermelho na Figura 12. Tal como já foi mencionado, esta foi replicada nos dois métodos de programação.

Figura 12. Estratégia de perfilagem criada para comparação entre os métodos de programação.

A estratégia foi pensada de forma a abranger duas partes distintas do percurso, que apresentam diferentes complexidades de programação, nomeadamente no que diz respeito à posição e orientação da ferramenta. Isto permitirá retirar informações importantes que serão úteis para o desenvolvimento do método de programação *teach & learn offline*.

4.1.2. ROBÔ

De modo a executar a trajetória programada, foi utilizado um robô antropomórfico Motoman GP25 de seis eixos, Figura 13. Este é coordenado por um controlador Motoman YRC1000 e o seu respetivo comando.

Figura 13. Robô Yaskawa Motoman GP25 [12].

É importante conhecer o comando de controlo do robô e as suas funcionalidades, independentemente do método de programação a utilizar. No caso do método de programação *online*, a sua programação é efetuada diretamente no comando. Já na programação *offline* é necessário colocar o programa gerado no comando, através de uma pen drive ou via rede, caso o comando esteja equipado para tal.

Na Figura 14 está presente um esquema das funcionalidades do comando de controlo.

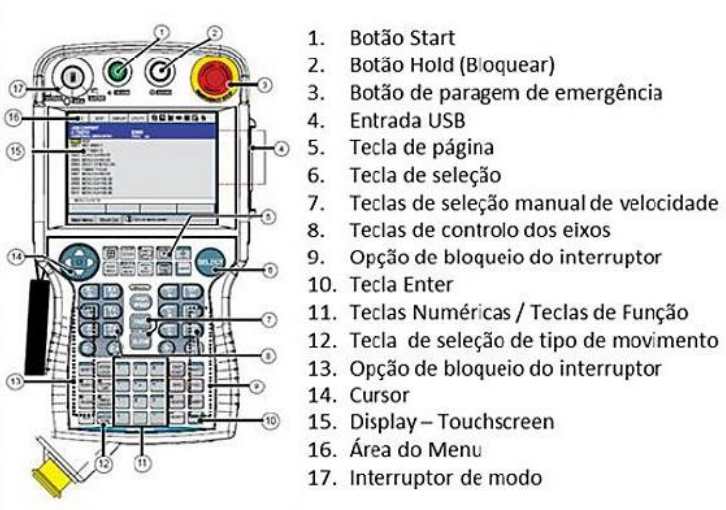

Figura 14. Controlador Motoman YRC1000 e comando, com as respetivas funcionalidades [4].

4.1.2.1. Calibração da ferramenta

Antes de começar a "ensinar" o robô um novo programa, ou colocar um novo programa no robô, é necessário garantir que a ferramenta a utilizar já se encontra calibrada. Este passo é essencial, especialmente quando vão ser realizadas estratégias com 3+2 eixos ou 5 eixos, nos quais uma má calibração irá refletir-se em defeitos no acabamento da peça.

4.1.3. PROGRAMAÇÃO *ONLINE*

Como já foi explicado anteriormente, o método de programação *online* é realizado diretamente no robô. Isto é, através do comando do controlador o utilizador move o braço do robô, de forma a memorizar pontos necessários para criar a trajetória desejada.

Com o intuito de programar a trajetória demonstrada na Figura 12, a peça foi posicionada primeiro junto ao robô, tal como é possível observar na Figura 15.

Figura 15. Robô Yaskawa Motoman GP25, spingle e peça.

Tendo em conta a complexidade da trajetória, foram criados dois programas diferentes, um para a zona A e outro para a zona B. Os programas foram criados com o maior número de pontos possível, para que a estratégia obtivesse uma maior precisão. De

forma a avaliar a importância do número de pontos, foi criado um terceiro programa, também, na zona A, mas com um menor número de pontos. Podemos observar na

Tabela 4 um resumo dos programas criados com este método.

Tabela 4. Programas criados com o método *online*: zonas, números de pontos memorizados e tempo de programação, correspondentemente.

Zona	Número de pontos	Tempo (minutos)
A (1)	208	45
A (2)	103	33
B	197	67

Neste âmbito, também foi criado um terceiro programa, na zona A, para estudar a hipótese de utilizar um programa número menor de pontos e como este se comporta em termos de desempenho.

As seguintes etapas foram realizadas no comando do controlador do robô para realizar a programação da estratégia pensada:

1. Colocou-se o interruptor de modo, representado na Figura 14 pelo número 17, no modo *Teach* (este modo permite controlar o robô manualmente);
2. Selecionou-se o número da ferramenta a utilizar no programa;
3. Selecionaram-se as coordenadas cartesianas (desta forma o robô comporta-se como um todo, em vez de se mover junta a junta);
4. Selecionou-se, no menu vertical presente no ecrã, *Job* e neste selecionou-se *Create new job*, no qual se deu um nome ao novo programa;
5. Um novo programa foi criado apenas com instruções *NOP* e *END*;

Para memorizar um ponto foi necessário, primeiro, colocar a ferramenta no local desejado. Para mover o braço do robô foi necessário selecionar, em simultâneo, o botão *Server On* e o comando ativador (que se encontra na parte de trás do comando). Só depois foi possível mover o robô utilizando os botões dos eixos.

Na Figura 16 é possível observar a ordem pelos quais os pontos foram memorizados e como a ferramenta se encontra quando memorizados.

6. Para memorizar cada ponto foi necessário selecionar os botões *Insert* e *Enter*. Caso o ponto seja mal memorizado basta selecionar o ponto em questão e selecionar os botões *Modidy* e *Enter*. Cada ponto memorizado vai representar uma linha de código.

Figura 16. Exemplo de alguns pontos memorizados na programação *online* realizada na zona A. (A) Pontos 1, 104, e 208 (B) Ponto 2.

Ao criar os programas acima mencionados foi necessário acrescentar pontos de aproximação, para que o robô não colidisse com a peça antes de começar a realizar percurso de perfilagem.

4.1.4. PROGRAMAÇÃO *OFFLINE*

Para programar uma trajetória de uma peça, no método *offline*, são necessários o desenho CAD da peça e um *software* CAM.

Como, no caso deste trabalho experimental, a peça foi fornecida sem desenho CAD, foi necessário digitalizá-la primeiro. Desta forma, foi utilizado um digitalizador *HP 3D Scanner Pro 3D*, o qual permitiu exportar o ficheiro digitalizado em *STL*[1]. Esta etapa demorou cerca de 15 minutos a ser completada.

Para que seja possível programar e simular uma estratégia num *software* CAM é necessário que o desenho CAD esteja arquivado num ficheiro dmt (*decalm machining triangles*). Assim, o ficheiro *STL* obtido foi convertido num ficheiro *dmt*, através do *software* CAD *PowerShape* da *AutoDesk*.

Para programar e simular a trajetória escolhida foi utilizado o *software* CAM PowerMill da *Autodesk*. Este permite o desenho 2D e 3D e o processamento de trajetórias

[1] Um formato de arquivo *STL* utiliza uma série de triângulos vinculados para recriar a geometria de superfície de um modelo sólido.

de maquinagem em sistemas multi-eixos, pois fornece estratégias para máquinas CNC de 3 a 5 eixos. A *Autodesk* descreve-o como um *software* especialista em maquinagem de alta velocidade e de 5 eixos.

Este *software* permite estudar a possibilidade de estratégias mais curtas, de forma a reduzir os tempos de maquinagem e desgaste das ferramentas utilizadas. De modo a obter as estratégias pretendidas, é necessário, primeiro, especificar as ferramentas de corte a utilizar e as dimensões do bloco a trabalhar. Através de simulações realizadas no próprio *software*, é possível verificar se existe ou não colisão entre a ferramenta e a peça e, também, se é possível aumentar a eficiência da estratégia escolhida.

De forma a criar um novo programa para o robô, primeiro, é necessário definir as estratégias no PowerMill, sendo que, depois, é necessário recorrer ao *plug-in* PowerMill *Robot Interface* (PRI) para concluir a programação do robô.

Tendo em conta a trajetória pensada, presente no capítulo "4.1.1 Peça e estratégia", foram realizadas as seguintes etapas:

1. Importou-se o ficheiro *dmt* para o PowerMill - Figura 17;

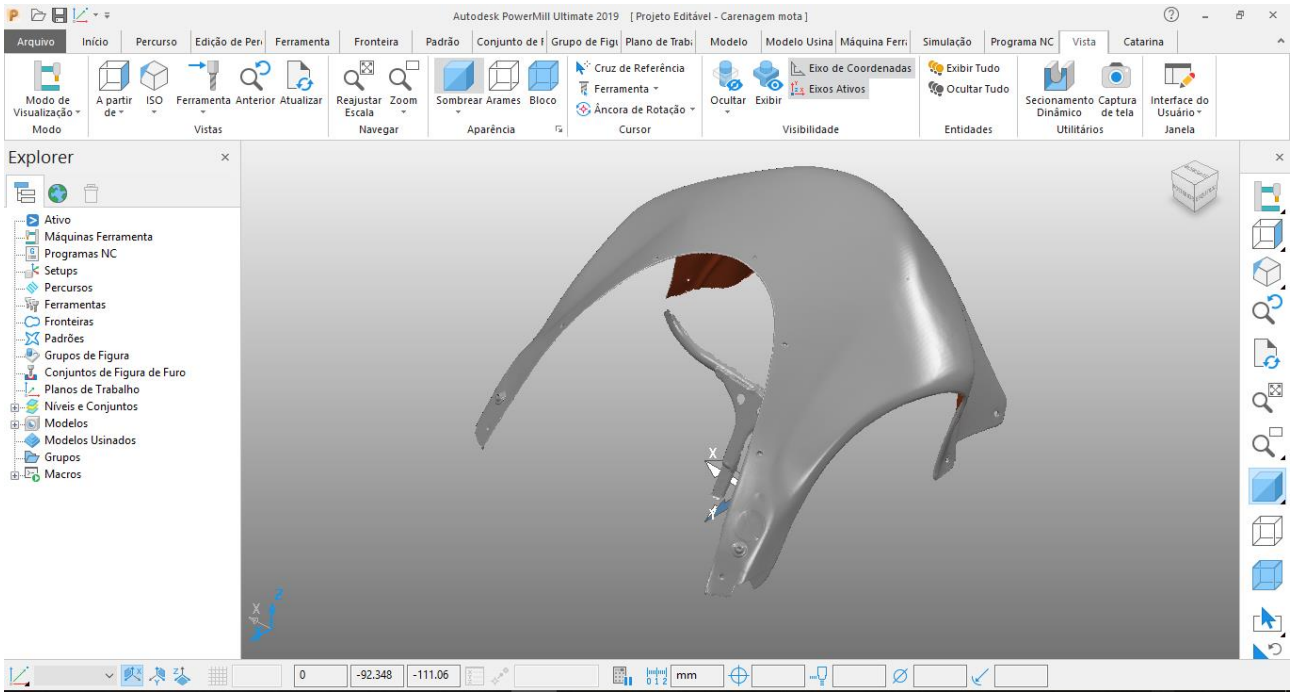

Figura 17. Interface PowerMill e desenho CAD da carenagem.

2. Criou-se uma fronteira personalizada, tendo em conta o modelo;

A fronteira personalizada é criada tendo em conta todas as extremidades do modelo. Desta forma, a fronteira obtida resulta numa linha que faz a delineação do modelo, como é possível observar na Figura 18 a amarelo, a circundar o modelo.

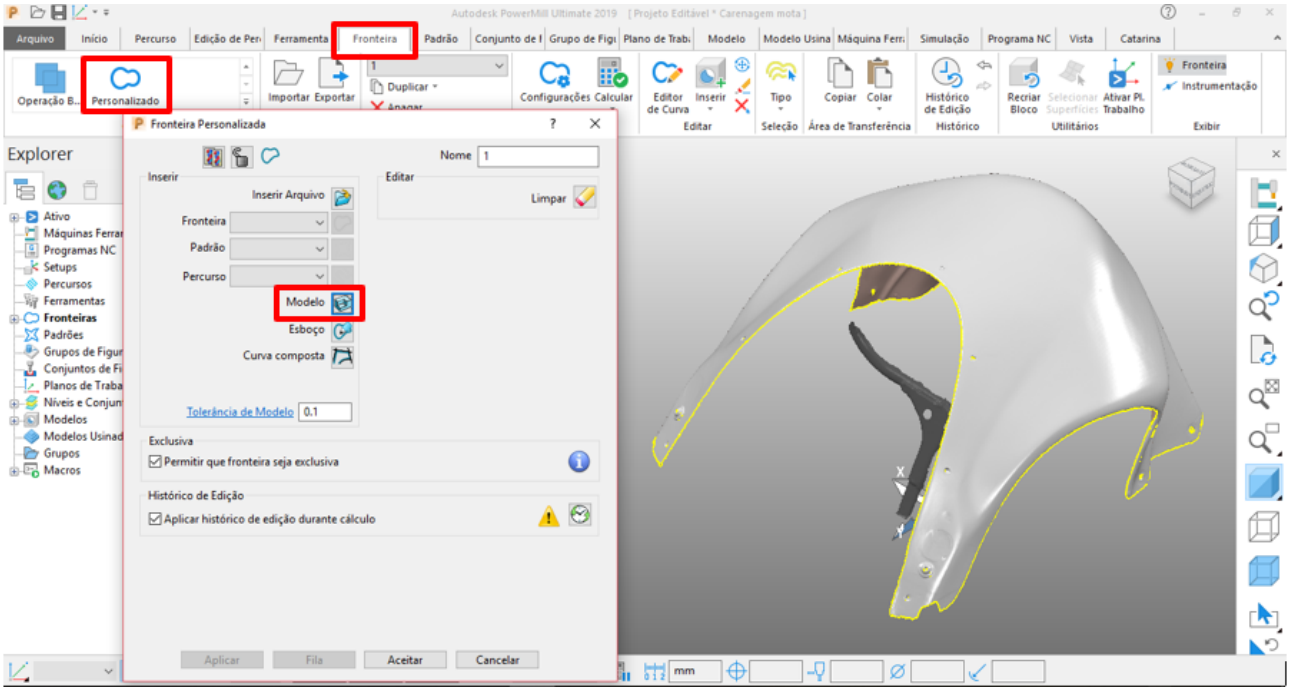

Figura 18. Fronteira personalizada.

Como o modelo CAD 3D foi obtido por digitalização da peça, é normal que esta apresente imperfeições que são designadas por "ruído", ou seja, rugosidades e informação adicional não presente na peça real.

3. Editou-se a fronteira, de forma a suavizá-la e a eliminar as zonas desnecessárias;

Como a trajetória planeada só abrange duas faces do modelo (zona A e B) foi necessário editá-la, ou seja, eliminar as zonas que não pertencem a esta.

Também foi necessário suavizar a trajetória. Tendo em conta que o ficheiro CAD foi obtido por digitalização, este vai apresentar alguma rugosidade nas suas extremidades, como é possível observar na Figura 19.

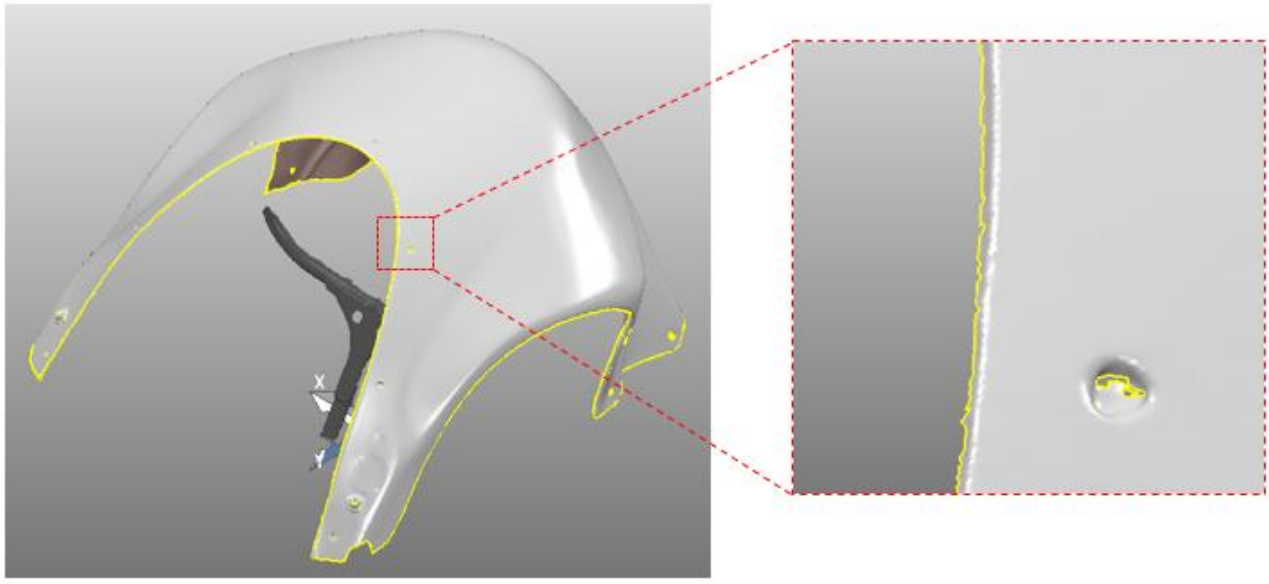

Figura 19. Rugosidade observada ao gerar fronteira.

Quanto mais suave for a trajetória, maior será a qualidade aquando da sua realização e a operação será realizada em menor tempo.

Foi também necessário criar duas fronteiras, uma para cada zona A e B, devido à complexidade de planeamento dos percursos e da programação.

4. Gerou-se um padrão a partir da fronteira criada no ponto anterior;

De forma a criar uma estratégia utilizando o PowerMill, é obrigatório criar um padrão, uma vez que as estratégias possíveis de se programar, apenas são construídas através de padrões e não fronteiras.

5. Criou-se um bloco (o *software* exige sempre a criação de um bloco, independentemente da estratégia criada ou do estado inicial da peça a maquinar);
6. Definiu-se uma ferramenta igual à que será utilizada na estratégia (esta permitirá retirar conclusões reais, aquando da simulação no *software* do estado final da peça, após a estratégia ter sido realizada);
7. Definiu-se um plano de trabalho no modelo;
8. Criou-se um percurso de "Usinagem de Perfil em Arames", tal como se demonstra na Figura 20;

De modo a criar a trajetória pretendida, através do padrão criado no ponto 4, seria espectável que o percurso utilizado fosse "Acabamento por Padrão". No entanto, este percurso requer que o utilizador prepare um padrão 2D, que será projetado sobre o modelo CAD [18].

Como o padrão criado para a trajetória de perfilagem é tridimensional, foi necessário utilizar a opção de percurso "Usinagem de Perfil em Arames", de forma a contornar o problema.

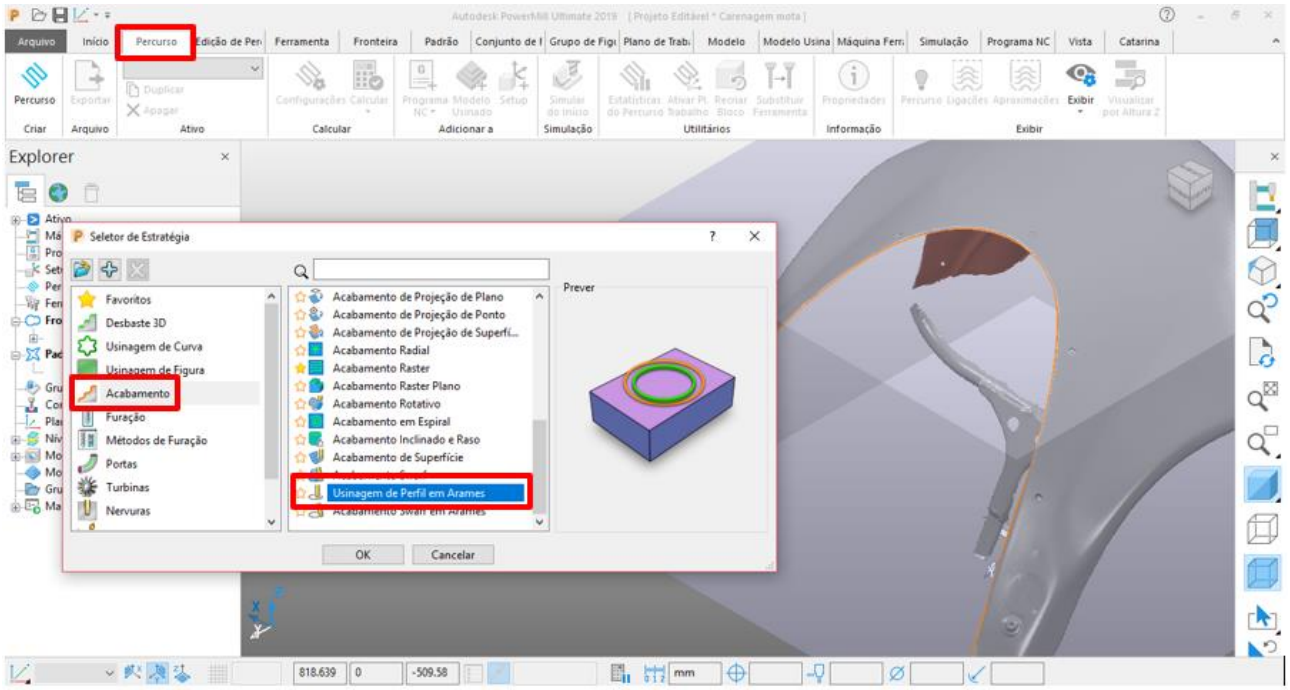

Figura 20. Criação de percurso de usinagem em perfil em arames.

9. Selecionou-se a fronteira criada como curva diretriz do percurso.

Concluindo estas etapas, foi gerado um percurso que vai seguir o padrão criado.

O percurso, presente na Figura 21, é representado a cor distintas. A cor verde indica os movimento de corte, a cor azul representa os movimentos de entrada/saída da ferramenta e a cor vermelha representa movimentos rápidos realizados acima do plano de segurança [18].

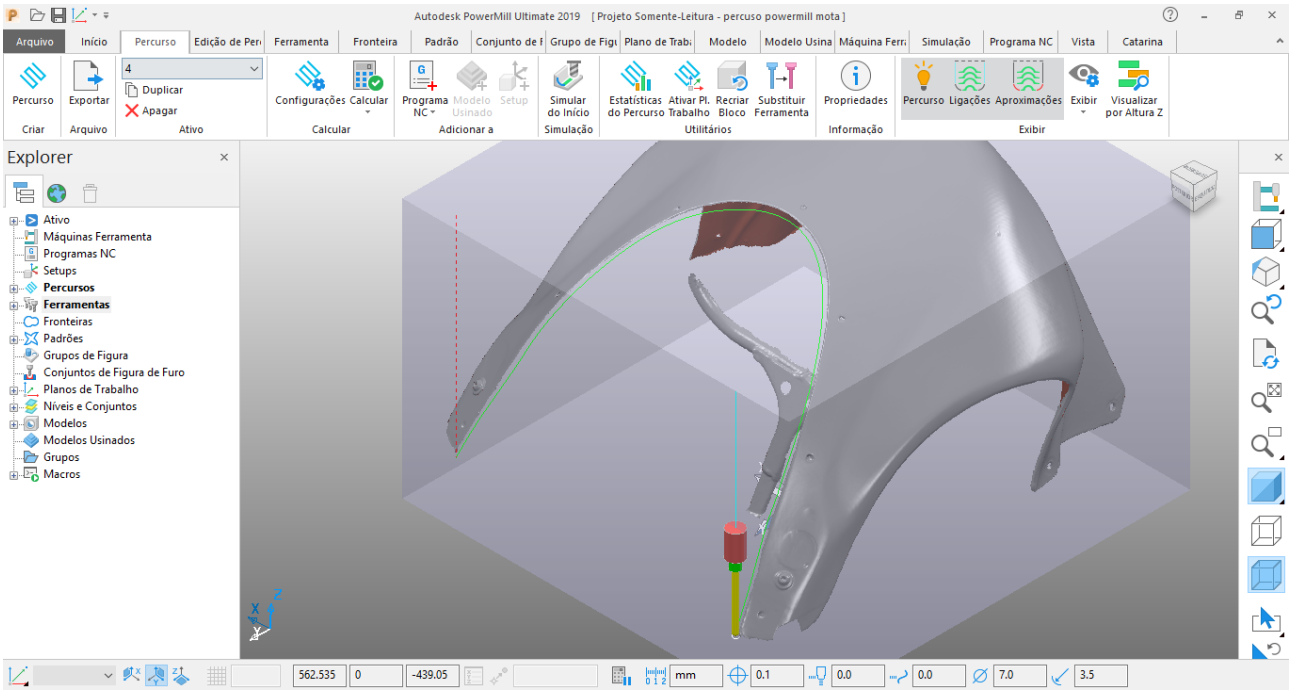

Figura 21. Percurso criado por usinagem em perfil em arames.

Estando o percurso criado, foi necessário recorrer ao *plug-in* PRI, presente na Figura 23. Este permitiu realizar a programação *offline* de robôs, da mesma forma como seria programada uma máquina CNC.

Ao utilizar o PRI é necessário recorrer aos quatro menus que este dispõe; sendo esses: Biblioteca de Robô, Célula de Robô, Controle de Robô e Programa de Robô [7].

10. Abriu-se o *plug-in* PRI;
11. No menu "Biblioteca de Robô", Figura 22, selecionou-se o robô *Norcam;*

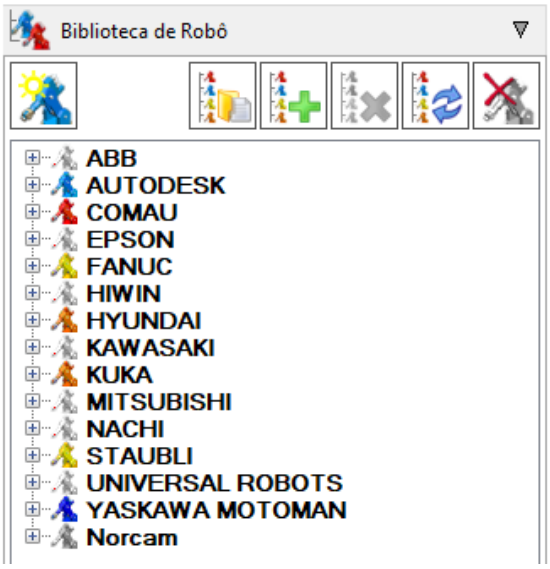

Figura 22. Menu "Biblioteca de Robô" presente no PRI.

Neste menu estão presentes os atalhos para os ficheiros *mtd* (*machine tool design*) e ficheiros *XML*, que ativaram o robô no PowerMill. Estes contêm os ficheiros *dmt* correspondentes aos diferentes componentes, eixos do robô e coordenadas relativas ao eixos das ferramentas [7].

Desta forma, ficou disponível para visualização, o robô que foi utilizado para simular a estratégia, como é possível observar na Figura 23.

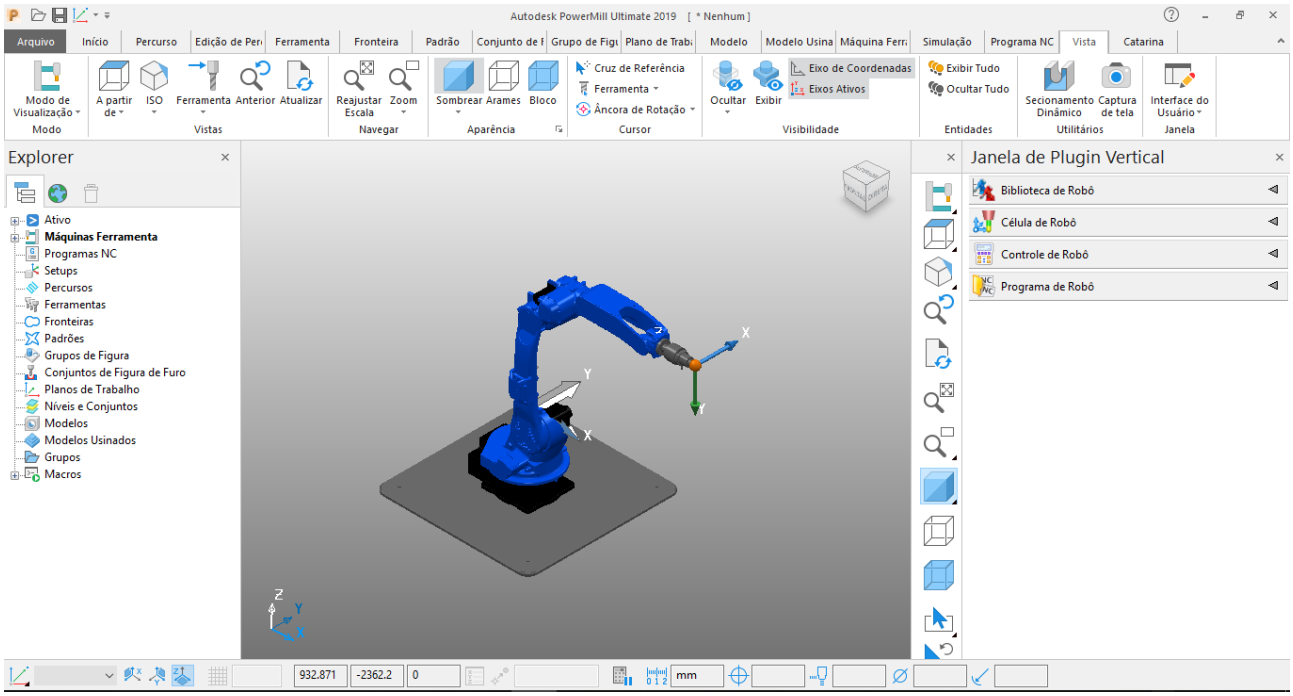

Figura 23. Interface PowerMill e PowerMill Robot Interface e robô para simular a estratégia.

12. No menu "Célula de robô", Figura 24, foi ajustado o plano de trabalho real ao plano de trabalho de simulação do modelo CAD, tal como mostra a Figura 26;

Este menu é responsável por todos os parâmetros relativos ao posicionamento de todas as partes constituintes da célula robótica. Ou seja, neste menu são ajustadas as posições da ferramenta, da mesa de trabalho e da peça a trabalhar (modelo CAD), em relação ao robô [7].

O botão assinalado a vermelho é dos mais utilizados neste menu, pois permite alterar a posição do modelo CAD, em relação ao eixo principal do robô, e ajustá-la à realidade.

Este método de programação não exclui a utilização do robô durante a sua programação, devido aos ajustes que são necessários realizar, de forma a que a simulação seja feita com a maior precisão e veracidade possível.

Foi necessário recorrer ao controlador do robô, para conhecer as coordenadas de origem do *user coordinate* a utilizar para a estratégia em questão e as rotações a ela correspondentes (Rx, Ry, Rz). Conhecer as rotações é importante, uma vez que a peça pode estar ligeiramente inclinada, relativamente ao plano de trabalho pré-definido.

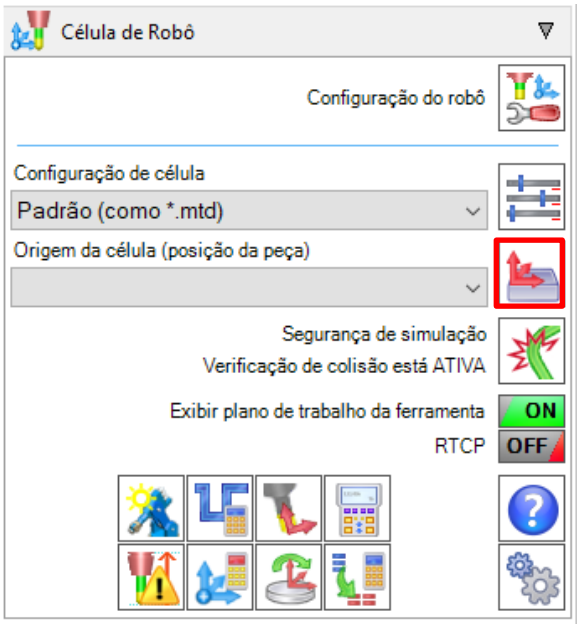

Figura 24. Menu "Célula de Robô" presente no PRI.

Foram necessários realizar, ainda, as seguintes etapas no robô, aquando da utilização do menu Célula de Robô:

- Colocaçou-se da peça na posição que esta assumiu durante a realização da estratégia;
- Criou-se um *user coordinate* igual ao plano de trabalho criado no ponto 7;
- No controlador retiraram-se as informações sobre as coordenadas de origem do *user coordinate* e respetiva rotação (Figura 25).

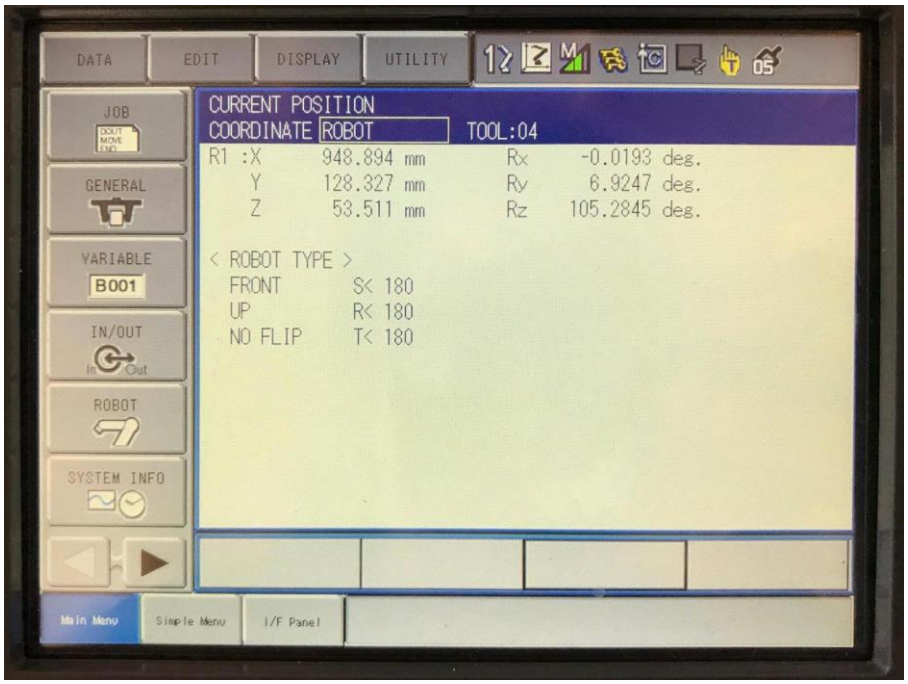

Figura 25. Interface do comando do controlador do robô - informações relativas ao *user coordinate* criado no robô e respetivas rotações.

Ao terminar estas três etapas, o utilizador não necessita de recorrer ao robô, nem ao seu controlador, até o programa estar completo.

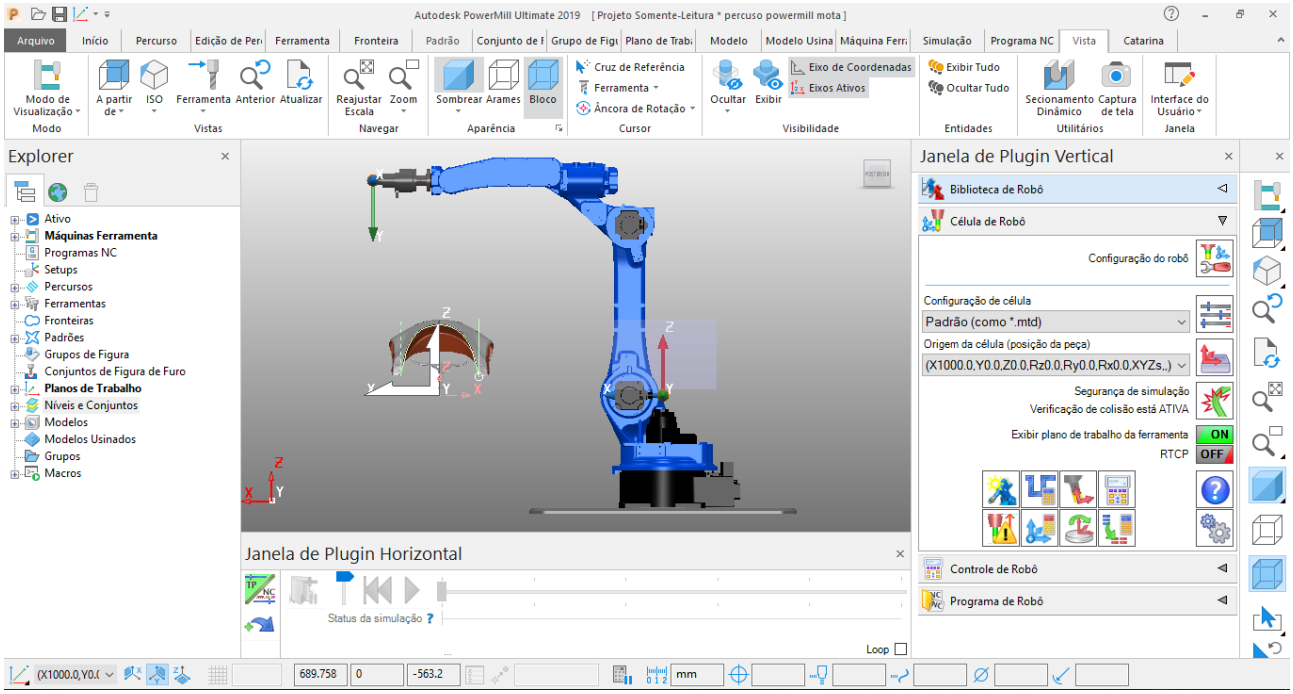

Figura 26. Posicionamento do modelo CAD em relação ao robô.

13. No menu “Controle de Robô”:
    a. Simulou-se a estratégia criada;
    b. Escolheu-se o vetor orientação da ferramenta;
    c. Guardaram-se os movimentos criados durante a simulação;

De modo a evitar singularidades e colisões ao utilizar o PRI, é necessário simular a estratégia criada até ao fim do percurso desta. Tal não é exigido quando são criadas estratégias para máquinas CNC [7].

Na Figura 27, podemos observar a interface deste menu. Nesta estão assinalados os comandos utilizados ao contruir o programa desta estratégia. De destacar o ponto b, vetor orientação da ferramenta, onde o utilizador controla a forma como a ferramenta executa o percurso, isto é, se a ferramenta executa a estratégia de forma livre ou seguindo vetores orientação.

A simulação correspondente ao percurso criado demorou cerca de 2 segundos a ser concluída.

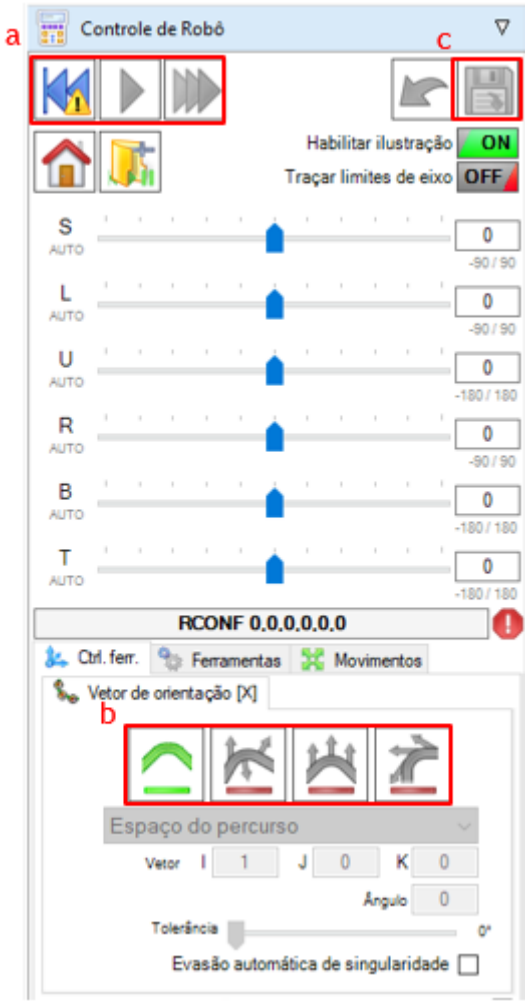

Figura 27. Menu "Controle de Robô" presente no PRI.

14. No menu "Programa de Robô" foi criado o programa NC de robô. Para tal foram realizados os seguintes passos:
    a. Selecionou-se o *user coordenate* criado no 12;
    b. Selecionou-se o pós-processador correto (*Motoman GP25);*
    c. Adicionaram-se os movimentos criados no ponto 13;
    d. Foram indicados os valores de velocidade, aceleração, precisão e *user coordinate,* no separador Parâmetros (Figura 28).

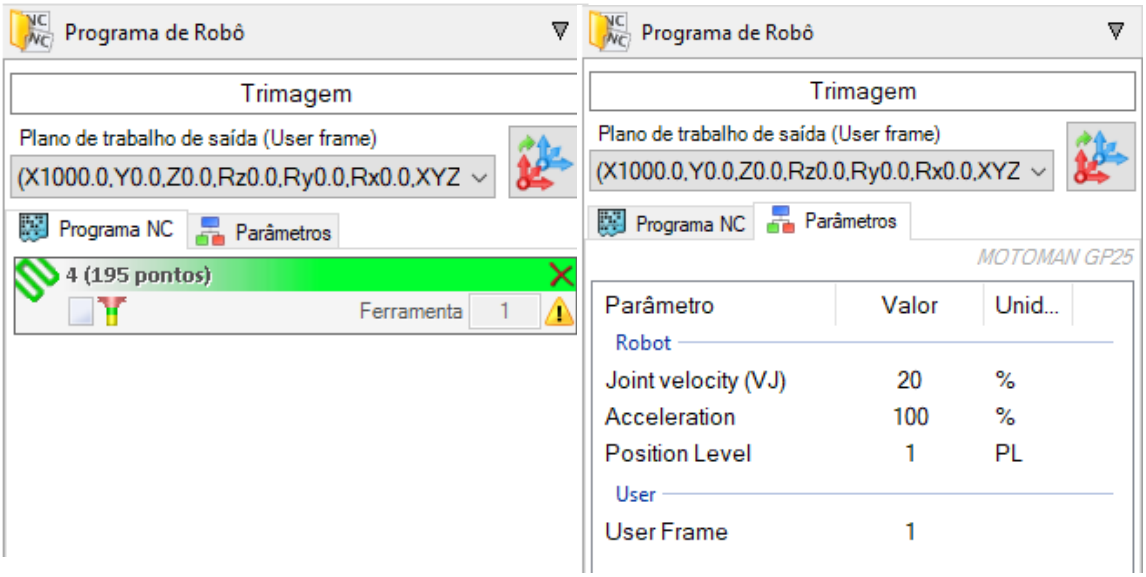

Figura 28. Menu "Programa de Robô" presente no PRI.

O programa foi criado com cento e noventa e cinco pontos chave. A criação do programa no PowerMill demorou cerca de vinte e cinco minutos.

No total, foram despendidos cerca de trinta e cinco minutos a realizar toda a sua programação, desde a digitalização, o planeamento do percurso e a criação do programa de perfilagem.

5. ANÁLISE E DISCUSSÃO DE RESULTADOS

A análise de resultados irá ser realizada tendo em conta os passos efetuados no procedimento. Primeiro será discutido o conceito do método de programação, depois será realizada uma análise das etapas de construção do percurso e, por fim, será discutido o programa gerado.

5.1. PROGRAMAÇÃO *ONLINE*

Os programas criados com este método foram todos realizados com o propósito de estudar a potencialidade deste método de programação. Como já mencionado no capítulo "4.1.3 Programação *Online*", foram realizados 3 programas distintos, presentes na Tabela 4.

Os programas A (1) e B, criados em zonas diferentes, foram alvo de estudo da orientação da ferramenta; pretendia-se saber, se através deste método o robô conseguiria interpretar as diferentes orientações que a ferramenta teria de assumir num determinado ponto. Dado que, uma localização no espaço pode estar associada a infinitas orientações, é necessário verificar se o ângulo ijk realizado com o robô, aquando da programação, está também gravado no programa criado, para que na execução da trajetória seja replicado e não cause problemas.

A ferramenta utilizada para os percursos não necessitou de mudar a sua orientação ao longo da zona A, devido à geometria que esta zona apresenta e ao fácil acesso que o robô consegue ter a esta. Tal não se verificou na zona B. Esta zona apresentou-se como uma zona de difícil acesso, pelo que a ferramenta para realizar a operação necessitou de assumir diferentes orientações. Desta forma, assumiu-se que o problema deste método está relacionado com a orientação da ferramenta e não tanto com a sua localização.

A programação da zona B foi a que consumiu mais tempo (setenta minutos), devido a todas as preocupações relacionadas com a localização dos pontos a memorizar e a orientação da ferramenta. Tendo em conta que a geometria desta zona é curva e inconstante, foi necessário memorizar mais pontos, de forma a verificar que toda a estratégia era cumprida. Também foi necessário alterar a orientação da ferramenta, quase em todos os pontos memorizados; isto é, como estava a ser programada uma estratégia de prefilagem, foi necessário verificar que a ferramenta estava na posição ideal para a realizar, pois nem todas as posições realizam esta operação com a máxima qualidade.

Estes problemas de orientação e posição da ferramenta estão associados à tridimensionalidade da estratégia, o que aumenta a complexidade de programação. E modo, é necessário garantir que o utilizador não só tem conhecimento do robô, desde movimentos a realizar, orientação da ferramenta e do comando do controlador, mas também, que tem experiência na operação de perfilagem.

Os programas criados na mesma zona, A (1) e A (2), foram alvo de estudo do número de pontos memorizados.

Ao criar-se um programa com poucos pontos, resultará que certas zonas da peça não sofrem qualquer alteração por parte da operação de perfilagem, tal como demonstra a Figura 29, e foi possível observar no programa A (2). Através dos resultados obtidos foi ainda possível verificar, que os dois programas criados apresentam precisões diferentes e, neste tipo de operações como perfilagem, é necessário que a precisão seja alta, uma vez que representa a última operação de pós processamento da peça em questão.

Deste modo, para este tipo de programação é interessante criar programas com um grande número de pontos, por forma a controlar este problema, uma vez que o número de pontos ideal pode ser um conceito abstrato para os programadores da estratégia.

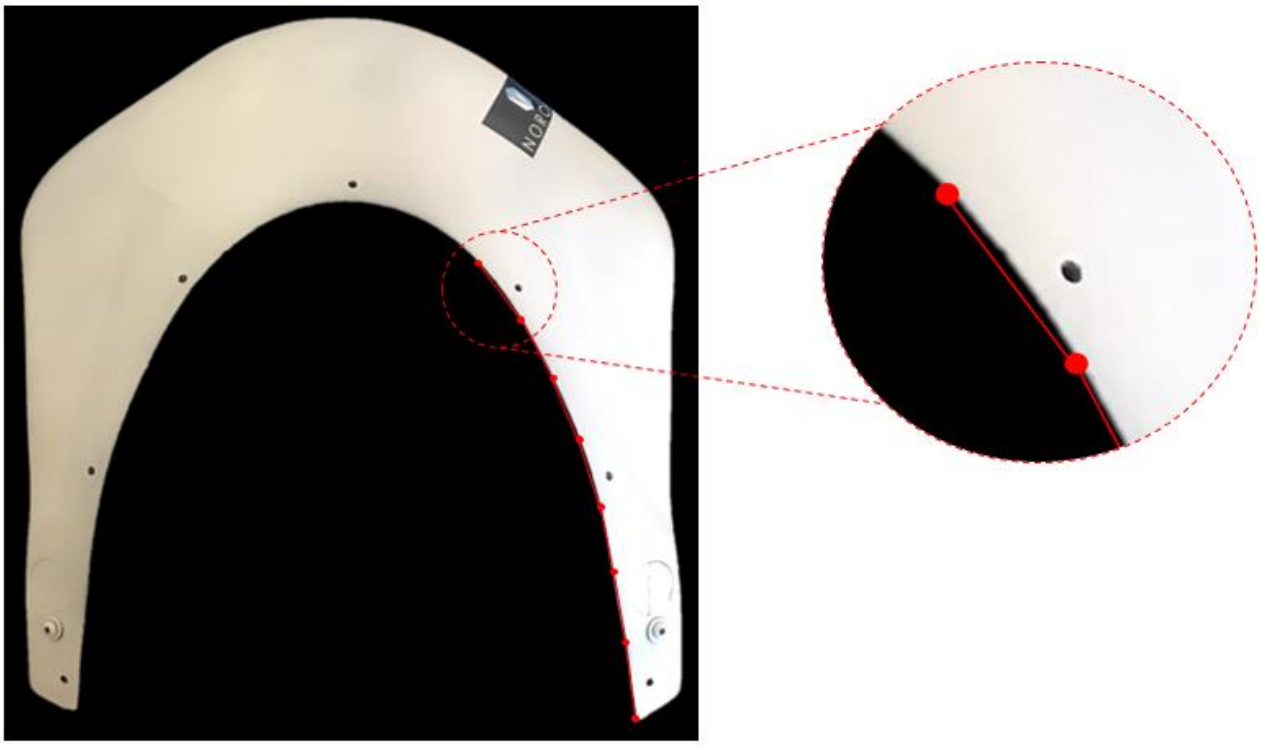

Figura 29. Movimento realizado entre pontos e erro associado.

Este método de programação é interessante para utilizadores que não possuam nem o desenho CAD da peça, nem o *software* CAM, o que acaba por minimizar bastante o custo associado à programação. Para além disso, é possível programar a peça no seu estado real; isto é, se a peça apresentar um defeito de fabrico, é possível contruir o programa à volta dele. Tal não se verificaria com a utilização de um *software* CAM, pois o desenho CAD não apresentaria nenhum defeito e a estratégia seria criada como se a peça se encontrasse em boas condições. Para contornar esse problema, a peça teria de ser *scanizada*, isto é, feito o levantamento de forma.

Contudo, um grande problema deste método está presente nos grandes tempos de programação. Estes vêm-se a refletir na fadiga do utilizador. Quanto mais tempo o

utilizador se estiver a dedicar a programar, mais cansado ficará e a qualidade da estratégia tenderá a diminuir.

O tempo total de criação desta estratégia foi de cento e doze minutos (quarenta e cinco da zona A (1) e sessenta e sete da zona B). Pelo que, enquanto se programou esta estratégia, o robô não foi utilizado para produzir valor de outra forma; o que representa custos de energia e diminuição do capital que a empresa teria a receber.

Quando todos os pontos desejados são memorizados, o programa fica completo e pronto para execução. Cada linha do programa contem a informação da localização do ponto memorizado correspondente, do tipo de movimento a realizar e da velocidade de deslocação até ao ponto, tal como é possível observar na Figura 30.

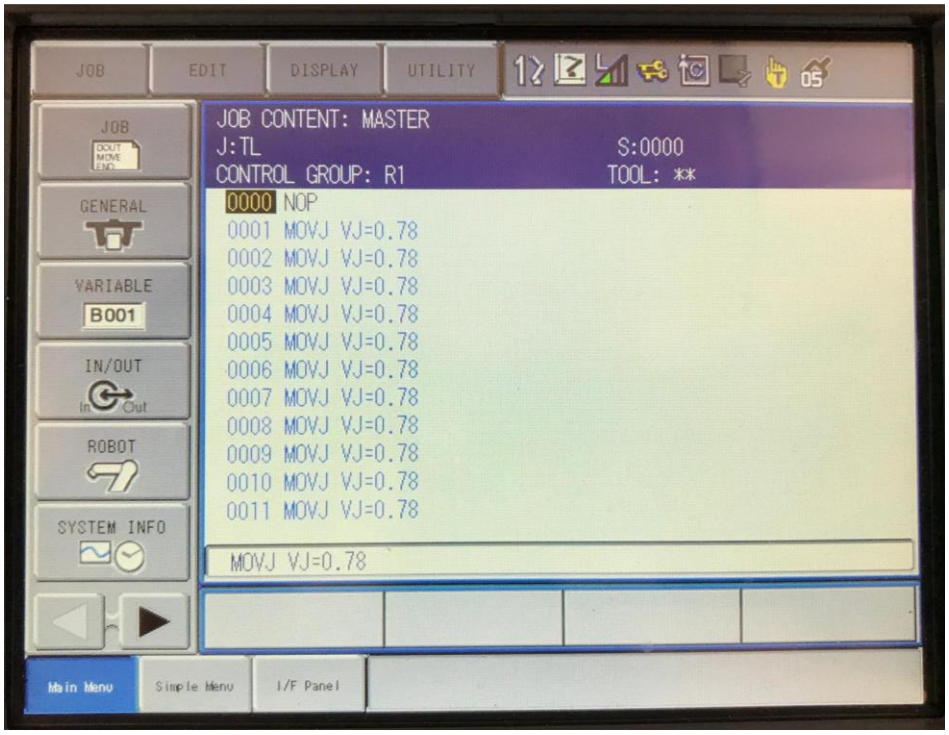

Figura 30. Programa obtido por Teach & Learn.

O novo programa é automaticamente criado com movimento de junta, por forma a evitar as singularidades, o que pode aumentar os tempos de execução da estratégia. A fim de modificar algum parâmetro no programa criado, é necessário fazê-lo linha a linha do código. Neste podemos modificar o tipo de movimento associado, a velocidade, a aceleração e a precisão.

5.2. PROGRAMAÇÃO *OFFLINE*

Antes de começar a analisar e discutir o método de programação, é necessário comentar os custos associados a um *software* CAM. Este poderá apresentar-se dispendioso para a empresa, uma vez que esta não só investiu num robô, como também num *software* CAM, que acarreta custos de compra e subscrição. Acresce ainda que, ao utilizar estes tipos de *software*, é necessário garantir que os seus utilizadores são especialistas nos mesmos. Ou a empresa investe em colaboradores já formados ou investe na formação dos seus atuais. Em ambos os casos a empresa necessita de investir capital.

Ao analisar todo o procedimento necessário para a realização deste método é possível verificar que uma grande desvantagem deste é necessitar obrigatoriamente, de um modelo CAD 3D da peça, de modo a criar a trajetória pretendida. No caso das estratégias planeadas, foi necessário recorrer a um digitalizador, para obter o modelo CAD da peça, o que aumentou o tempo de planeamento, em quinze minutos.

Tendo em conta o *software* utilizado e os passos realizados, presentes na seção "4.1.4. Programação *Offline*", é de notar que apenas um utilizador experiente, isto é, com formação, seria capaz de reproduzir toda a etapa de criação de percurso, no menor tempo possível. Para realizar uma estratégia no PowerMill, é necessário cumprir uma série de etapas já pré-estabelecidas pelo próprio. Isto é, para criar um novo programa é preciso: obter o modelo CAD 3D, definir um bloco, uma ferramenta, criar uma estratégia (esta poderá implicar mais ou menos etapas, dependendo da operação pretendida), definir os parâmetros da estratégia (desde velocidade, altura de segurança, entradas e saídas, tolerância, entre outros), realizar uma simulação e gerar o programa. Cada uma destas é realizada num local do *software* específico. Por mais que as diferentes opções se encontrem bem assinaladas, poderá ser complicado para um utilizador inexperiente saber como estas funcionam e de que maneira a conseguem implementar no programa a criar. Desta forma, é fundamental realçar a importância da formação.

Tendo em conta a estratégia criada para a operação de perfilagem no PowerMill, esta foi criada com cento e noventa e cinco pontos em cerca de vinte e cinco minutos. Ao utilizar o *software,* foi possível encontrar o número de pontos ideal para realizar a estratégia pensada. Ao contrário do método de programação *online*, que depende da experiência e conhecimento da operação por parte do utilizador, este método consegue arranjar de maneira rápida a otimização ideal para a criação do percurso. Tal revela-se uma vantagem, pois programas com muitas linhas de código exigem que o robô despenda mais tempo a ler as informações que tem de realizar. Por sua vez a execução desses mesmos programas será mais demorada. Desta forma, ao otimizar o número de pontos-chave do percurso, está-se a garantir um menor tempo de execução da estratégia, enquanto a qualidade desta se mantem.

A geração do programa NC utilizado no robô foi realizada no PRI onde foram consumidos cerca de dez minutos. Tendo em conta que a simulação realizada consumiu cerca de dois segundos, é possível concluir que maior parte do tempo foi despendido no

ajuste dos planos de trabalho real vs CAM. Mesmo utilizando um método que não necessita diretamente do robô para a criação da estratégia requerida, para a realização desta etapa, este não é independente do robô, o que aumenta os tempos de programação.

A simulação do percurso criado revela-se importante neste método de programação. Como toda a programação deste percurso é realizada num computador, é essencial garantir que este não trará problemas durante a sua execução, nomeadamente singularidades e colisões. Esta também é executada rapidamente e evita a realização de testes reais, que envolveriam diversos custos, desde energia, desgaste das ferramentas, sucata da peça teste, entre outros. Para além disso, simular o percurso no PRI permite que o robô execute outras tarefas que produzam valor para a empresa.

Sendo o percurso de perfilagem apresentado um percurso tridimensional, este envolve uma programação complexa, devido às translações e rotações do robô e da ferramenta, a ele associados. Ao utilizar um *software* CAM, é garantido que este será otimizado pela complexidade de informações que este tipo de *software* contém.

Na Figura 31, está presente o comando do controlador do robô com o respetivo programa NC criado. É possível observar na Figura 31 e na Figura 32 que o programa gerado por este método de programação se apresenta muito mais completo do que o gerado pelo método *online*.

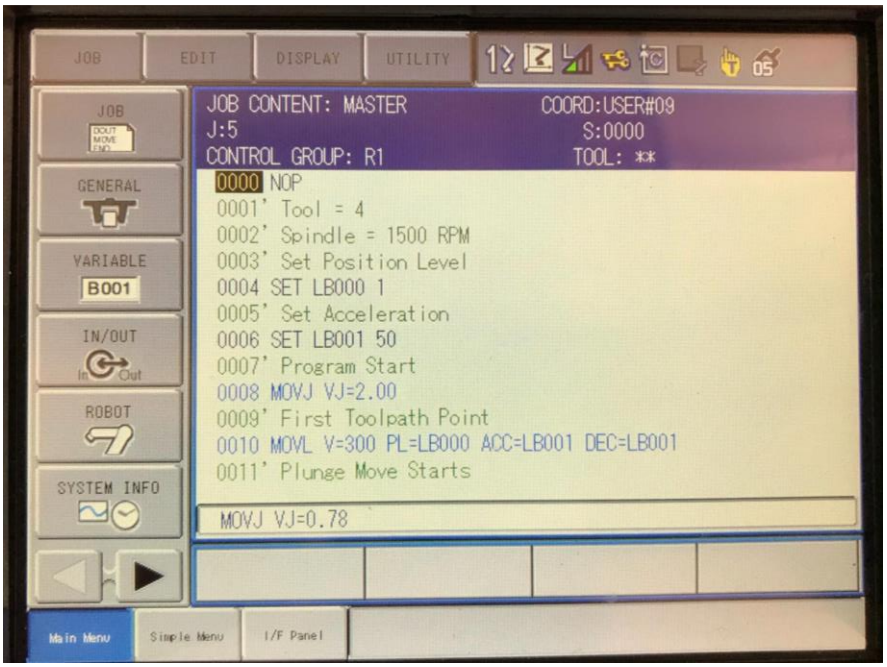

Figura 31. Programa obtido pelo *software* PowerMill.

Ao ler o programa no controlador do robô, é possível observar que este apresenta linhas com diferentes cores. As linhas de cor verde representam a informação que o *software* fornece ao utilizador para melhor interpretação do próprio código. Caso algum dado necessite de ser alterado, já depois do programa ter sido gerado, o utilizador não necessita de recorrer ao *software* novamente para o realizar. Por exemplo, na Figura 31,

podemos observar que a aceleração corresponde a cinquenta, mas, caso o utilizador queira alterar para vinte, basta utilizar o comando do controlador para o fazer, da mesma maneira que o faria na programação *online*. Desta forma, é possível concluir que independentemente do método de programação a utilizar, o utilizador necessita obrigatoriamente de conhecer o funcionamento do robô e do respetivo controlador e comando.

Cada linha de código apresenta informações relativas aos movimentos a realizar: o tipo de movimento (MOV), a velocidade (V), a precisão dos movimentos (PL), a aceleração (ACC) e a desaceleração (DEC), como é possível observar na Figura 32. Nem todas estas informações são necessárias, pelo que para certas estratégias podem apresentar inúmeras vantagens e para outras não.

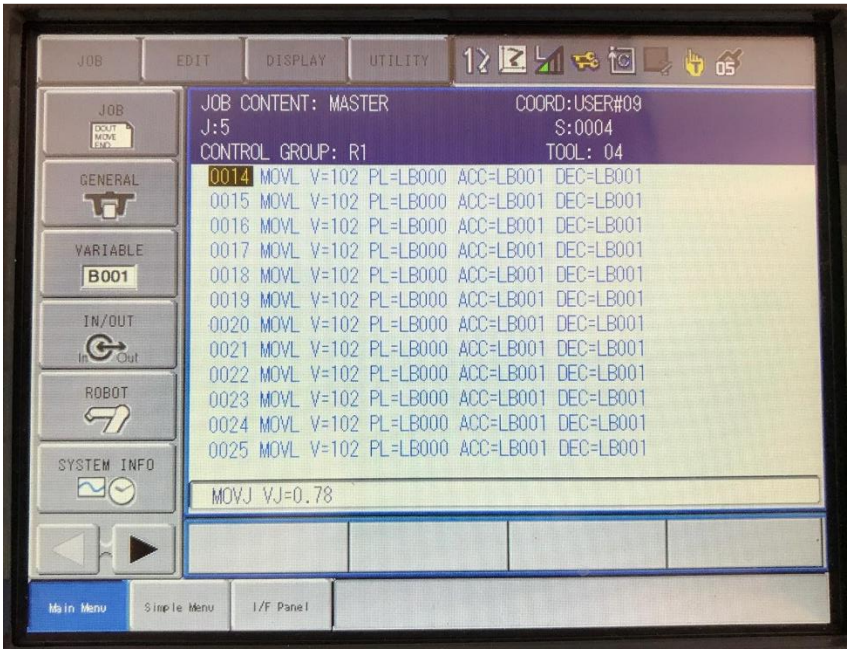

Figura 32. Programa obtido pelo *software* PowerMill - informação acoplada a cada ponto.

Quanto mais informações o código tiver presente, maior vai ser a sua qualidade pois é possível suavizar ao máximo a estratégia ao executá-la. Para além disso, para certas operações, certas informações revelam-se interessantes, de modo a utilizar toda a potencialidade que o robô consegue fornecer. Por exemplo, caso sejam utilizados pontos de aproximação, é interessante modificar a precisão (PL) destes para valores mais altos (quanto mais alto for o PL, menor será a precisão dada ao ponto), de forma a diminuir os tempos de operação.

Tendo em conta o programa gerado para a operação de perfilagem, é possível averiguar que este contem informação não necessária para a execução do programa, nomeadamente no que diz respeito à precisão. Quando este parâmetro não está escrito no programa, o controlador do robô assume automaticamente a precisão que este deve ter de acordo com a velocidade escolhida.

Estes tipos de informações desnecessárias apresentam-se como um problema, pois quanto maiores forem os programas (quantas mais linhas apresentarem) mais tempo vão consumir na sua execução, como já previamente mencionado.

Ao executar o programa criado, foi possível verificar que a sua execução necessita de ajustes. De forma a compreender o sucedido, foi criado um esquema exemplificativo presente na Figura 33, na qual o programa foi executado da direita para a esquerda (assumindo a posição real da peça em relação ao robô, que se encontraria no seu lado direito).

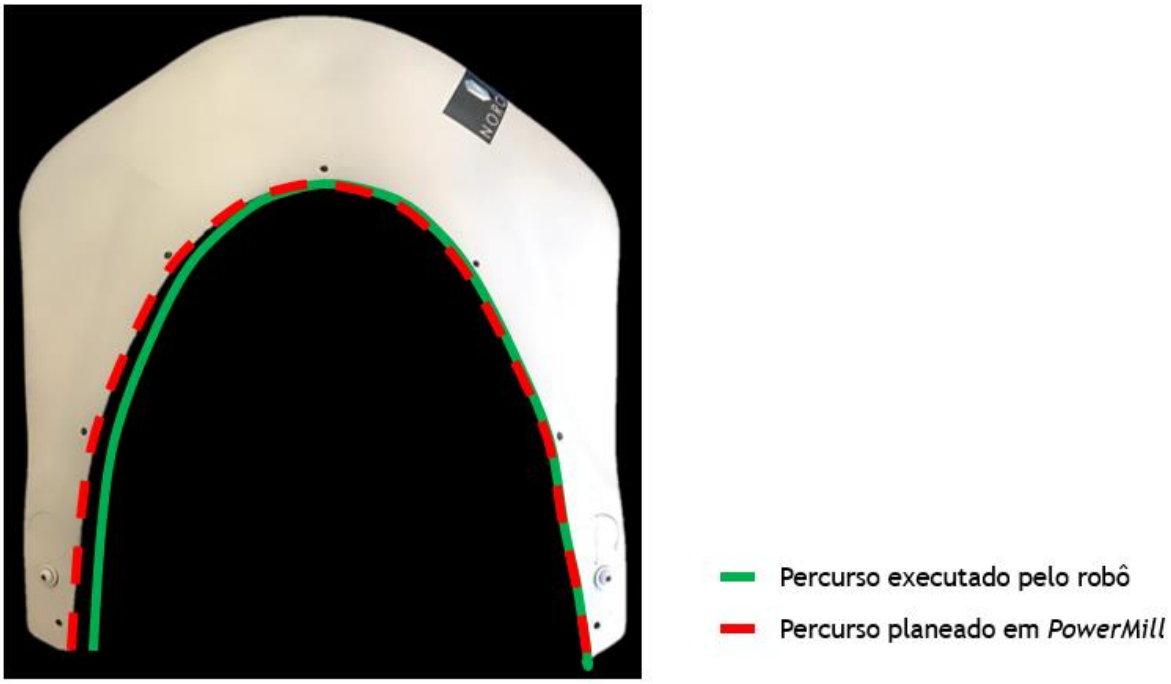

Figura 33. Percurso executado pelo robô vs percurso planeado em PowerMill.

Este problema está associado a uma incorreta digitalização da peça. A peça foi inicialmente digitalizada sem qualquer tipo de suporte. No entanto, para que fosse possível realizar e verificar a veracidade desta, foi necessário recorrer a um gabari[2], para que a carenagem não sofresse vibrações ou deslocações do seu plano de trabalho aquando da realização do programa. Este expandiu a morfologia da peça, pelo que a estratégia CAD deixou de executar o programa com a maior precisão possível. Para otimizar o percurso criado foi necessário realizar uma segunda digitalização, desta vez com o gabari e criar novamente um programa NC completo, seguindo as mesmas etapas.

Avaliando o sucedido, foi possível verificar um problema ao programar trajetórias em CAM através de modelos CAD. Normalmente, excluindo as peças digitalizadas, os

[2] Gabari: Peça que visa fornecer estabilidade a outra, de forma a garantir o controlo dimensional durante operações como maquinagem. Isto é, diminuir desvios na tolerância ou vibrações.

modelos CAD não apresentam defeitos. Os *software* CAM, por sua vez, vão realizar uma estratégia sem ter em conta os defeitos, o que pode vir a ser desnecessário, pois o percurso pode não apresentar qualquer resultado. O *software* não vai dar atenção ao defeitos, uma vez que não assume que existem, o que pode vir a invalidar certas peças na revisão de qualidade final. Sendo que tal foi verificado no caso desta estratégia, onde uma parte da carenagem sofreu a operação de perfilagem e a outra continuou ia como estava na pré-operação.

5.3. COMPARAÇÃO DOS MÉTODOS ATUALMENTE UTILIZADOS

Tendo em conta a análise de resultados realizada para cada método de programação, é possível verificar que aspetos de cada um se revelam interessantes para a programação de robôs.

Ao longo de cada seção anterior já foram realizadas algumas comparações entre os dois métodos. Para além das já efetuadas, ao longo deste capítulo, serão realizadas mais comparações e comentadas.

O primeiro aspeto relevante entre os dois métodos de programação está relacionado com o investimento que as empresas necessitam de realizar. Enquanto que na programação *online* é necessário apenas investir no robô e na formação de um utilizador sobre robô comprado, na programação *offline* é necessário investir: no robô, na formação de um colaborador no robô, na compra e subscrição de um *software* CAM e na formação de um colaborador no *software* CAM.

Enquanto que para programar *online* apenas são necessários a peça e o robô, para programar *offline* são necessários o *software* CAM e o modelo CAD da peça. Tal como já foi explicado, o modelo CAD da peça pode apresentar-se como um problema. Ou este é fornecido de forma a criar a estratégia ou então é necessário digitalizá-la. Independentemente de como é obtido o ficheiro CAD, este é necessário, o que aumenta o tempo de programação.

Também é preciso ter em consideração as conformidades do ficheiro CAD e da peça que se vai trabalhar. O modelo CAD vai fornecer informações do estado da peça ao *software* CAM, de forma a que este consiga construir um percurso. Caso a peça se encontre defeituosa, em relação a como está desenhada no seu modelo CAD, o programa obtido por programação *offline* não vai ser capaz de construir um programa que contorne o problema. Desta forma, a programação *online* ganha vantagens, uma vez que a personalização do percurso é realizada de uma forma mais eficaz.

Relativamente à programação realizada em ambos os métodos, estes demonstram-se ser cansativos e exigentes. É possível considerar que esta poderá ser considerada o *bottleneck*[3] de todo o processo de criação e execução de uma estratégia num robô.

A programação *online* exige que o utilizador esteja concentrado durante todo o seu processo. Este necessita de conhecer bem que estratégia está a programar, de forma a colocar a ferramenta na orientação correta. Este trabalho pode ser muito aborrecido uma vez que para deslocar a ferramenta, o utilizador necessita de pressionar os botões x, y e z (no caso de estar a trabalhar em coordenadas cartesianas) sempre que pretende deslocar a ferramenta ou reorientá-la. Sendo este um trabalho exaustivo, que leva rapidamente à fadiga do utilizador, é normalmente recomendado para percursos curtos. Ao realizar a programação por aprendizagem, é necessária especial atenção de forma a garantir que a qualidade da estratégia não é comprometida.

A programação *offline,* por sua vez, exige muito conhecimento e perícia para utilizar o *software* CAM, de forma a criar o programa no menor tempo possível. Estes *software*, como o PowerMill, apresentam uma grande variedade de percursos a criar e o utilizador tem de ser capaz de conseguir filtrar qual a informação necessária, para criar a estratégia pretendida. Tendo escolhido o melhor percurso, o utilizador necessita de cumprir com um número vasto de etapas, para garantir a qualidade da operação que vai realizar. Como está a programar a estratégia utilizando um *software,* é garantido que está vai ser criada o mais otimizada possível.

Em comparação com a programação *online*, o utilizador não necessita de conhecer a operação a realizar, basta escolher a estratégia certa e o programa o ajudará como, por exemplo, a otimizar o número de pontos (caso de estudo demonstrado na análise de resultados dos dois métodos estudados). A qualidade da estratégia também se verifica superior na programação *offline*, uma vez que esta permite a suavização do percurso. Na programação *online*, o utilizador memoriza pontos com o robô e estes vão automaticamente ficar ligados entre si, sem qualquer tipo de suavização ajustada à estratégia requerida.

Ao contrário da programação *offline*, a programação *online* não permite que haja simulações, a não ser que o utilizador decida realizar um teste. Partindo do princípio de que, como a programação *online* é realizada diretamente com o robô e com a peça, a sua estratégia não apresentará problemas como singularidades e colisões. No entanto, o programador necessita de considerar se é ou não necessário memorizar pontos de aproximação, de forma a que não haja colisões peça vs. robô. Já para a programação *offline*, a simulação é essencial para garantir que não existem singularidades e colisões antes de se correr o programa no próprio robô, evitando problemas de segurança.

Através da análise dos programas obtidos é possível verificar uma grande diferença entre os seus códigos. Enquanto que o programa obtido por programação *online* é mais

[3] *Bottleneck*: Fator limitante, que impede que o sistema atinja o seu pleno potencial.

simples, o obtido por *offline* é mais completo. Apresentar um programa mais completo pode apresentar as suas vantagens ou não. Como já discutido, quantas mais informações o código contiver maior vai ser a sua qualidade, mas mais lenta será a sua execução.

Através da programação *online,* podem-se acrescentar mais informações do que as pré-estabelecidas, assim que concluída a criação do programa. No entanto, estas alterações são realizadas linha à linha, o que pode ser um trabalho considerado exaustivo. Por sua vez, a programação *offline* apresenta sempre o mesmo formato do código, sendo que por mais que não se preencham algumas informações ao criar o programa NC, estas são criadas na mesma, como por exemplo a informação relativa ao PL. Independentemente se o utilizador fornece ou não informação, o PowerMill vai fornecer ao PL um valor fixo.

Na Tabela 5 é possível observar um resumo de todos os pontos anteriormente mencionados.

Tabela 5. Comparação entre os dois métodos de programação.

	Programação *online*	**Programação *offline***
Preparação (Pré-Programação)	Necessita da peça a trabalhar e do comando do controlador do robô	Necessita de um modelo CAD 3D, de um de *software* CAM e do robô
Programação	Programa trajetórias simples Exige formação de colaboradores no robô	Programa qualquer tipo de trajetórias Exige formação de colaboradores no robô e *software*
Resultados	A qualidade pode vir a ser comprometida pelo colaborador	Qualidade da estratégia garantida
Tempo	112 minutos	35 minutos (com digitalização)

Desta forma, pode-se concluir que o ideal seria combinar partes dos dois métodos, de forma a retirar o melhor proveito do planeamento/programação e execução da estratégia pensada.

6. DESENVOLVIMENTO PROGRAMAÇÃO *TEACH & LEARN OFFLINE*

Concluído o mapeamento do estado atual através do qual os robôs são programados, o foco foi direcionado para a identificação de diferentes áreas de atuação, com o objetivo de melhorar e facilitar o próprio processo de programação.

Tal como já explicado no capítulo "3. Programação *teach & learn offline*", este novo método pensado procura integrar a parte de planeamento da estratégia, por aprendizagem com o braço de metrologia e a parte de criação do programa num *software* a desenvolver. Esta solução foi pensada de forma a reduzir os tempos totais de programação, diminuindo a necessidade de altos investimentos e formação, procurando uma melhoria de qualidade do percurso a executar.

Neste contexto, foi necessário desenvolver um *software* capaz de recolher todos os pontos memorizados, com o braço de metrologia e criar um percurso a partir destes, gerando um programa capaz de ser lido pelo robô.

O software foi desenvolvido tendo em conta o braço de metrologia e o robô disponíveis para testes. Isto é relevante, pois utilizando outro tipo de braços ou robô, o *software* não iria dar a resposta desejada, uma vez que apenas contém a informação dos pós-processadores para o qual foi projetado.

Desta forma, foi utilizado o braço de metrologia *EVO 7* da *RPS Metrology*. Programação *teach & learn offline"*. Este é um braço transportável com sete eixos. A sua manipulação é bastante intuitiva, pelo que não requer muita experiência por parte do utilizador. Para mexer no braço, basta desbloqueá-lo, deslocar o braço para o percurso pretendido e pressionar o botão sempre que for necessário memorizar um ponto novo, ou então pressionar continuamente. A sua manipulação está esquematizada na Figura 34.

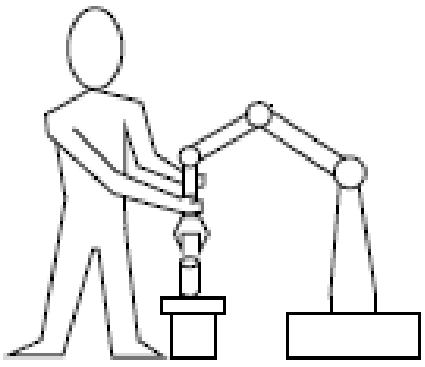

Figura 34. Esquema metodologia criação de estratégia pelo método de programação *teach & learn offline*, adaptado de [23].

De modo a obter a trajetória planeada com o braço, é necessário conectá-lo ao computador ou através de um cabo USB ou por Wi-Fi.

A interface do primeiro *software* desenvolvido está presente na Figura 35. Para se proceder à criação de um novo programa NC foi necessário, primeiro, conectar o braço ao robô e, de seguida, utilizar o braço para memorizar a estratégia desejada. Na interface do *software* foi possível observar a estratégia realizada em duas dimensões; isto é, cada um dos quadrados pretos representa um dos plano (x,y), (x,z), (y,z), o que poderia não facilitar a criação da estratégia. Por um lado, como a estratégia foi criada pelo utilizador, este conseguia ter uma noção da tridimensionalidade e de como esta se comportaria no espaço. Porém, caso o utilizador pretendesse alterar algum aspeto da estratégia, ou eliminar um ponto mal memorizado, o programa não permitiria de forma intuitiva descobrir onde estaria o problema.

Na janela a branco foi possível observar as coordenadas de todos os pontos memorizados, bem como o vetor ijk correspondente. Caso fosse necessário alterar algum dado do ponto tal era possível, bastava que o utilizador clicasse no ponto a modificar.

Figura 35. Interface do primeiro *software* desenvolvido para o método *teach & learn offline*.

Já com a estratégia desenhada, selecionou-se em "POST-PROCESS FOR MOTOMAN" de forma a criar o programa NC a ser lido pelo robô. Este comando abriu uma nova janela, presente na Figura 36, à qual o utilizador deveria indicar a velocidade que pretende que o programa seja executado no robô, o número da ferramenta e do *user coordinate*.

O número da ferramenta e do *user coordinate* foram informações diretamente retiradas do robô. O robô já tem presente e calibrada a ferramenta a utilizar e memorizado o plano de trabalho.

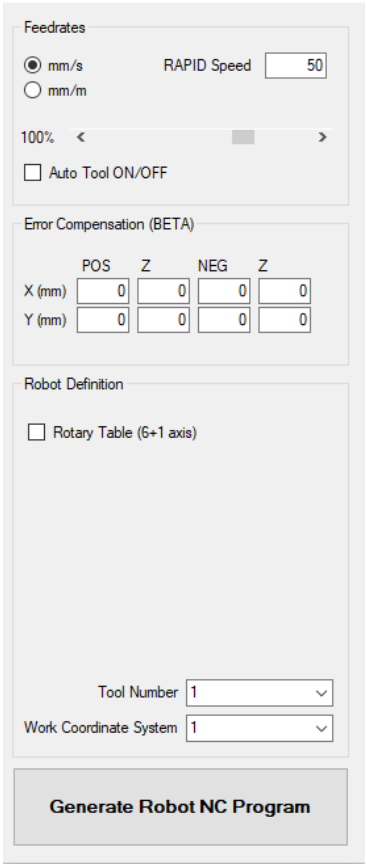

Figura 36. Criação de programa NC para o primeiro *software* desenvolvido para o método de programação *teach & learn offline*.

Infelizmente, este *software* não conseguiu resolver certos problemas, pelo que foi complicado descobrir onde colocar a peça a trabalhar no robô. Isto porque foi criado um plano de trabalho para a peça a trabalhar no robô, mas o *software* assumiu sempre o seu plano de trabalho geral como o plano a criar a estratégia. Para além disso, por mais que a trajetória fosse criada com diferentes inclinações da ponteira do braço, a ferramenta apresentava sempre a mesma posição vertical. Isto deve-se à falta de ferramentas matemáticas presentes no software capazes de realizar transformações de translação e rotação de forma a correlacionar os planos de trabalho criados.

Para que este *software* fosse independente, seria necessário que neste estivessem contidas as ferramentas matemáticas necessárias para:

1. Correlacionar os planos de trabalho criados no robô com os planos de trabalho criados a partir do braço de metrologia;
2. Converter os vetores ijk memorizados em ângulos Euler.

De forma a contornar este problema foi considerada a hipótese de, atualmente, utilizar este *software* como um *plug-in* do PowerMill. Este *software* CAM para além ser capaz de observar a trajetória criada com o braço tridimensionalmente já possui um conjunto de ferramentas matemáticas necessárias para realizar os diferentes ajustes

necessários à criação de uma estratégia. A criação do programa NC para o robô ficaria de total responsabilidade do PRI. A este novo *plug-in* desenvolvido foi chamado de *RPS*2PowerMill.

As etapas efetuadas para construir um programa NC correspondente à operação de perfilagem da carenagem, pelo método *teach & learn offline*, foram as seguintes:

1. Abriu-se o PowerMill;
2. Conectou-se o braço de metrologia EVO7 ao computador;

Após conectar o braço ao computador, este abre automaticamente uma janela na qual o utilizador deve realizar uma rápida calibração. Nesta, o utilizador deve rodar todos os eixos presentes no braço, de forma a ver se tudo se encontra operacional.

3. Abriu-se o *plug-in RPS*2PowerMill, que apresenta a interface presente na Figura 37;

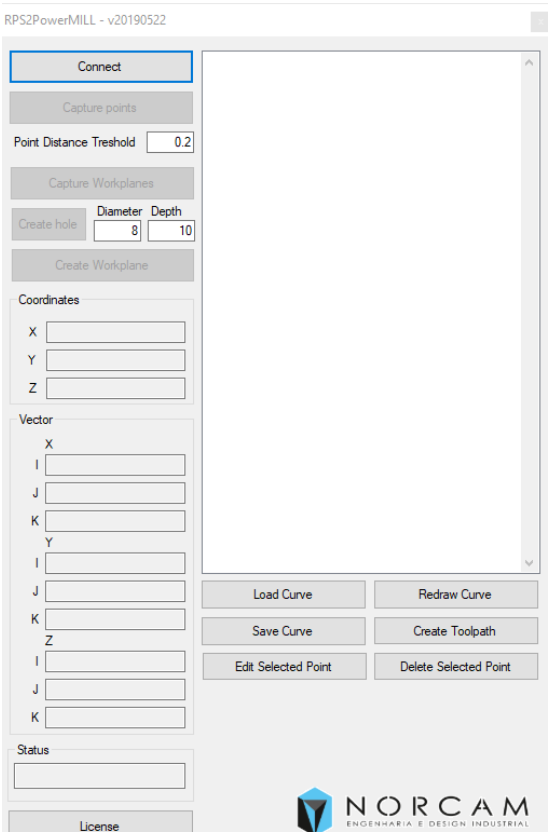

Figura 37. Interface *plug-in RPS*2PowerMill.

4. Selecionou-se em conectar e de seguida em capturar pontos. Ao mesmo tempo foi utilizado o braço EVO7 e foram retirados pontos continuamente;

Por mais que se tenha selecionado em capturar pontos no *plug-in*, estes só são capturados quando o utilizador clica no botão do braço.

À medida que os pontos vão sendo memorizados, estes começaram a ser descritos no mostrador do *plug-in*, onde são descritas as coordenadas de cada ponto e o respetivo vetor ijk, relativo à orientação da ponteira, como é possível observar na Figura 38.

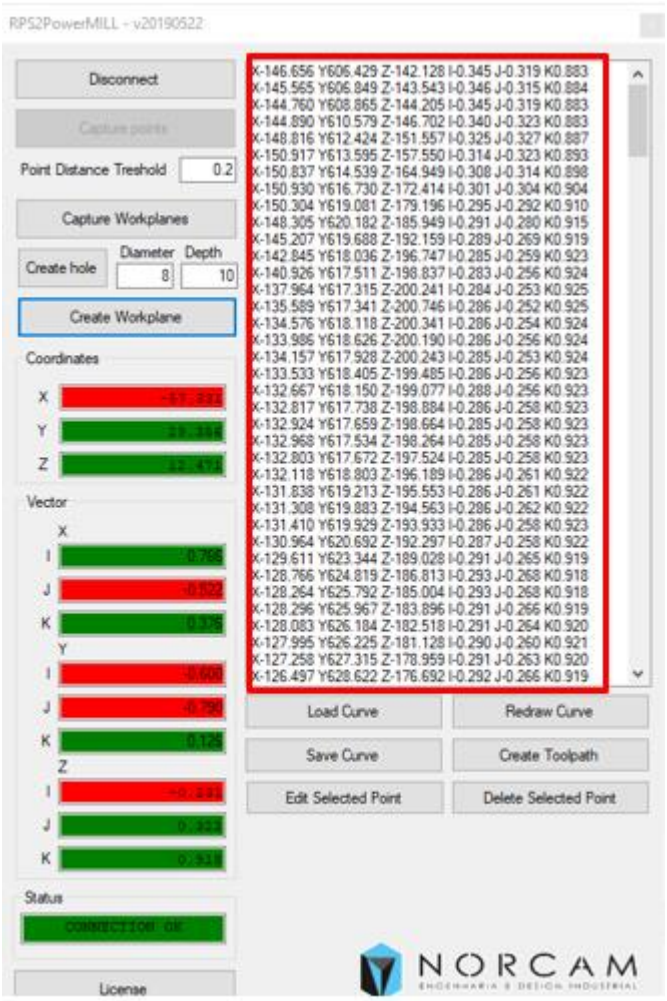

Figura 38. Pontos memorizados no *plug-in* RPS2PowerMill.

Foi, assim, criado um padrão, tendo em conta os pontos memorizados, como demonstra a Figura 39. Antes de prosseguir, foi necessário averiguar se este ficou de acordo com a estratégia desejada. O PowerMill permite, nesta etapa, alterar a geometria do padrão criado. Ao criar um padrão através de pontos contínuos, é possível que este apresente muita vibração, pelo que se pode suavizar a estratégia nesta etapa. De notar que esta edição do padrão é facultativa.

Estando o padrão de acordo com o desejado, foi selecionado "Aceitar", ficando o padrão criado completo e, assim, deste modo, foi possível prosseguir para a etapa seguinte.

Como este método não necessita do modelo CAD da peça para programar a estratégia, no *software* vai ser apenas possível observar o trajeto efetuado com o braço.

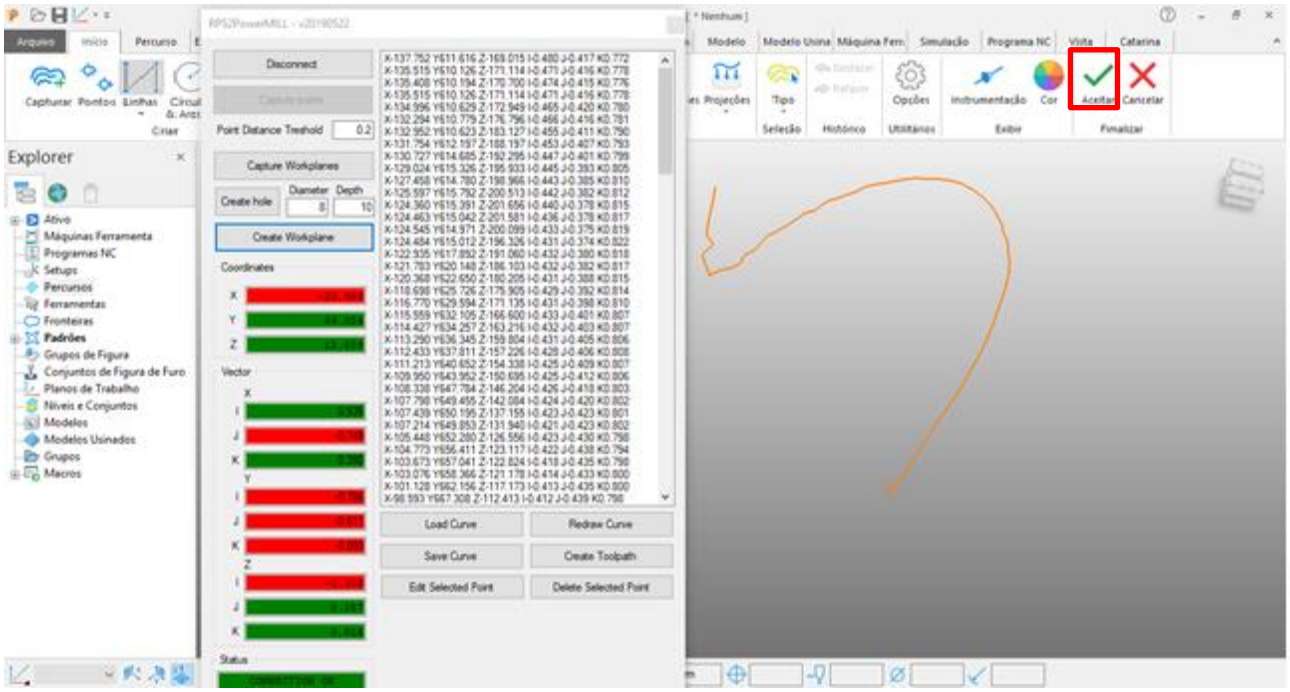

Figura 39. Criação de padrão através do pontos memorizados.

5. Criou-se um plano de trabalho através de três pontos, no *software*, e procedeu-se à criação de um novo *user coordinate* no robô;

Começou-se por selecionar a opção "Create Workplane" no *plug-in*. Esta abre automaticamente a opção do PowerMill de criação de "Plano de Trabalho por Três Pontos" (Figura 40).

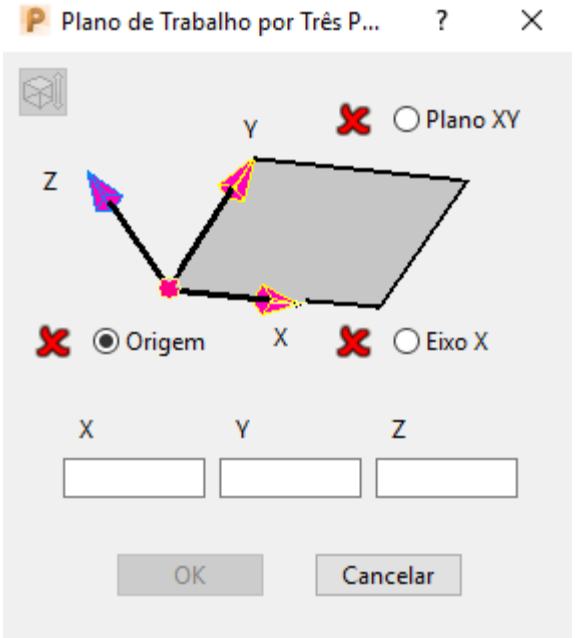

Figura 40. Criação de plano de trabalho por três pontos.

Estes três pontos foram retirados com a ordem demonstrada na Figura 41 através do braço de metrologia. Para registar cada ponto bastou colocar a ponteira do braço no local indicado e clicar no botão lá presente. O mesmo processo foi realizado no robô para criar um novo *user coordinate*, seguindo a mesma ordem de pontos.

Tendo em conta a regra da mão direita, o eixo de cordenadas Z ficará a apontar para baixo. Isto não se revelará como um obstáculo, uma vez que este problema pode ser ajustado nas etapas que utilizam o PRI.

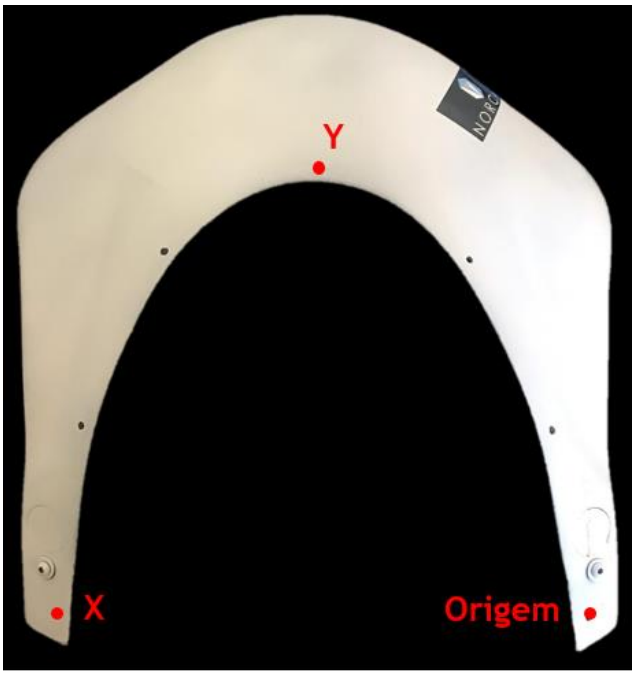

Figura 41. Criação de um plano de trabalho e respetivo *user coordinate*.

6. Selecionou-se em "Create Toolpath" de forma a criar, desde logo, um percurso;

Como já descrito no capítulo "4.1.4. Programação *Offline*", a criação de um percurso exige o cumprimento de um cumprimento de etapas. Ao utilizar este *plug-in,* o percurso foi automaticamente criado, pois o RSP2PowerMill criou de forma autónoma um bloco e uma ferramenta para o realizar através da estratégia planeada, como é possível observar na Figura 42.

Quando se escrever o programa NC, como é necessário indicar o número da ferramenta para o criar, não haverá problema, uma vez que esta se encontrará automaticamente calibrada (já foi previamente calibrada pelo robô e ele ajustará o programa criado tendo em conta a sua calibração).

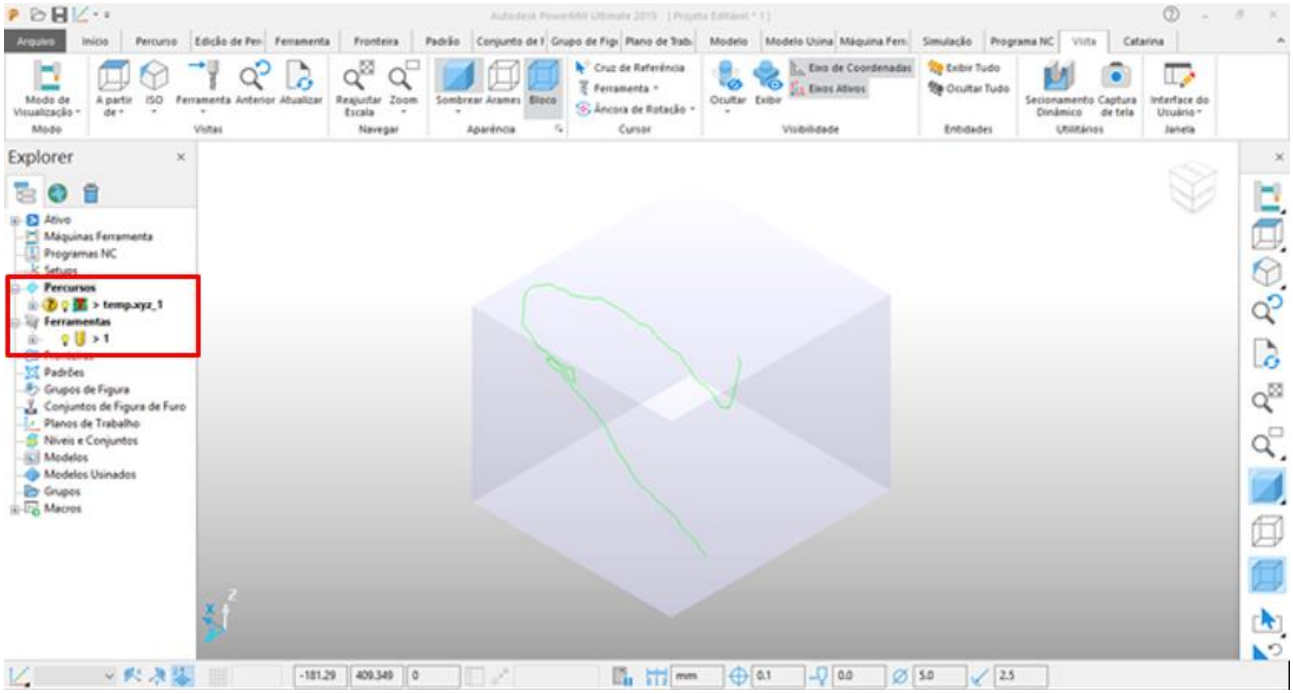

Figura 42. Criação do percurso.

Estando o percurso já planeado, deixa de ser necessário recorrer ao *plug-in* *RPS*2PowerMill. A criação do programa NC é realizada no PRI, seguindo as mesmas etapas já explicadas no capítulo "2.4.2. Programação *offline*" (da etapa dez à catorze).

Este método de programação torna-se muito mais simples e acessível aos seus utilizadores. Estes não ficam necessariamente cansados de manipular o braço, uma vez que o movimento que vão executar é contínuo e mais suave do que na programação *online*. Isto, tendo em conta que na programação *online* o utilizador desloca o braço do robô através do comando do manipulador acionando os botões x, y e z, sendo um processo moroso. Para além disso, as empresas não necessitam de investir em muitas horas na formação de colaboradores experientes no *software*, pois este método é bastante intuitivo, ao contrário da programação *offline*. Contudo, sabendo que este *software* presentemente funciona como *plug-in* de um *software* CAM, é necessário que ainda haja investimento na formação, nomeadamente PRI.

Ao retirar os pontos com o braço, é imperativo dar importância ao local onde a ponteira é colocada, quando os pontos são memorizados. Este local será onde a ferramenta a utilizar na estratégia programada terá a sua ponta. Desta forma, ou os pontos são retirados com o braço mais abaixo do local indicado, ou então terá de ser realizado um ajuste da estratégia, de modo a garantir que esta é executada.

Mesmo assim, a utilização do braço de metrologia para desenhar a estratégia pode vir a ser dificultada, tendo em conta a ponteira utilizada. Na criação da trajetória de perfilagem, utilizando este método de programação, foram empregues duas ponteiras diferentes, uma com uma esfera de rubi e uma em cone de aço (Figura 43).

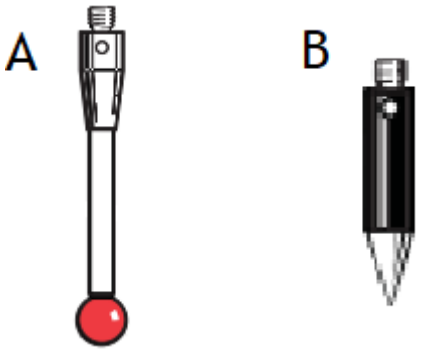

Figura 43. Ponteiras: A) esfera de rubi, B) cone de aço, adaptado de [24].

Ao utilizar a esfera de rubi como apalpador, foram sentidas inúmeras dificuldades. Uma vez que a esfera não apresenta nenhuma aderência à carenagem, esta desviou-se facilmente da trajetória requerida, quando o utilizador não apresentava uma mão mais firme, como é possível observar na Figura 44. No entanto, e devido à captura contínua de pontos, a trajetória apresentou-se com muita vibração, pelo que para aumentar a qualidade da estratégia foi necessário suavizá-la, editando-a no PowerMill. Este estado de vibração também foi observado com a ponteira em cone de aço.

Tendo em conta a operação de perfilagem realizada, a ponteira mais indicada foi a em cone de aço. Esta memorizou os pontos com uma maior precisão entre a peça e a rebarba de injeção, devido à facilidade de utilização do seu vértice.

Figura 44. Exemplo de erro ao criar o percurso devido à falta de atrito entre a ponteira e a carenagem.

Novas ponteiras podem ser idealizadas para facilitar a realização de mais estratégias com este método de programação, por exemplo uma ponteira cilíndrica. Desta forma, seria mais fácil manter o braço no local pretendido, sem receio que este saísse da sua estratégia.

Ao criar o programa NC no PRI, obteve-se um código com a estrutura igual ao código gerado para a programação *offline*. Este apresenta-se como um código completo, que permite garantir a qualidade das estratégias planeadas. No entanto, ao seguir o formato *standard* do *software* CAM para programas, o número de linhas de programa será maior e, por sua vez, a estratégia ao ser executada vai ser mais lenta do que poderia ser.

Infelizmente, ao utilizar este *software* como *plug-in* de outro pode ser muito dispendioso, devido a todos os investimentos que este apresenta. Este método, como está desenhado atualmente, exige a compra de um robô, a compra de um braço de metrologia, a compra e subscrição de um *software* CAM e a formação dos seus utilizadores.

Contudo, ao utilizar este método de programação, o próprio processo de utilização do *software* CAM é simplificado. Este não necessita de um modelo CAD, para que seja criada a estratégia e permitirá reduzir os tempos de programação. Não haverá necessidade de digitalizar a peça antes da programação, para obtenção de um modelo CAD, nem a preocupação de execução de um número elevado de etapas para construir a própria estratégia. Tal é observado na comparação dos tempos totais de programação. A estratégia planeada por *teach & learn offline* foi rapidamente desenhada no *software* e programada, sendo que utilizando este método, foram consumidos cerca de quinze minutos. Já no caso da programação *offline,* esta demorou cerca de vinte e cinco minutos e na *online* cerca de quarenta e cinco minutos (se considerarmos apenas a zona A da estratégia).

7. CONCLUSÕES

A presente dissertação teve como principal objetivo analisar os métodos de programação atualmente utilizados pela indústria, e propor a criação de um novo metodo com base na utilização de um braço de metrologia 3D.

Com o objetivo de compreender os passos essenciais para a construção de um programa de robô, desde o planeamento de estratégia à criação do programa NC, foram estudados os métodos de programação *online* e *offline*. Através deste estudo foi possível observar as vantagens e adversidades sentidas em cada método.

O método de programação *online* revelou-se interessante, na medida em que não são necessários grandes investimentos para a realizar. No entanto, este consiste num método moroso, pelo que apenas é utilizado na indústria para trajetórias pequenas.

O método de programação *offline* revelou-se mais interessante, não só pelos tempos reduzidos de criação de um novo programa, em comparação com a programação *online*, mas também pela qualidade que este acrescenta ao programa NC. Contudo, para utilizar este método é obrigatório possuir o modelo CAD da peça, onde a estratégia será realizada. Para além disso, este método obriga a que as empresas invistam continuamente nesta metodologia, não só na subscrição do *software* CAM a utilizar, mas também na formação dos seus utilizadores.

Ambas as metodologias se caracterizam por apresentarem períodos de criação da estratégia relativamente longos. A programação *online* exige muita concentração e perícia por parte do programador, para que o colaborador consiga memorizar ponto a ponto, utilizando o comando do controlador do robô. A programação *offline*, por sua vez, exige formação e experiência no *software* CAM a utilizar, para permitir ao programador cumprir todas as etapas exigidas para criação da estratégia, de forma rápida.

Através da observação dos programas gerados por ambas as metodologias, chegou-se à conclusão que para criar um programa NC são necessárias certas informações-chave. De forma a recriar a estratégia a realizar, é necessário ter em consideração: os pontos-chave, a velocidade de operação e a orientação da ferramenta, assim como, o número da ferramenta e o *user coordinate* a utilizar.

A metodologia *teach & learn offline* em desenvolvimento procurou ser uma solução para os problemas encontrados até ao momento, nos métodos utilizados pela indústria, focando-se nos tempos consumidos a programar a trajetória e na facilidade de criação desta.

Ao associar o *software* desenvolvido *RPS2*PowerMill com o PowerMill, foi possível encontrar uma solução rápida. Através da utilização do braço de metrologia foi possível capturar todos os pontos necessários à construção da estratégia e visualizá-los na interface

do PowerMill, o que permitiu realizar alterações nesta para a suavizar, garantindo, desta forma, um aumento de qualidade. Adicionalmente, ao utilizar o PowerMill *Robot Interface* foi possível criar um programa de robô completo, o qual permitiu correlacionar as coordenadas do braço com as coordenadas do robô, o vetor orientação da ferramenta e simular a estratégia.

No futuro, deve continuar-se a apostar no desenvolvimento deste *software,* de modo a que este consiga criar programas NC de forma autónoma. Estes passarão pela implementação de ferramentas matemáticas que permitirão correlacionar os sistemas de coordenadas do braço com o robô e converter os vetores *ijk* em ângulos Euler. Este novo *software* trará benefícios, não só a nível de tempos de programação, mas também a nível de custos associados a toda a logística.

8. TRABALHOS FUTUROS

Para atingir o máximo potencial deste novo método de programação, seria interessante desenvolver o *software* atual, de modo a maximizar a independência no seu *modus operandi*. Isto é, fazer com que este deixe de ser um *plug-in* de outros *software* CAM. Para além disso, seria interessante expandir a sua programação para outro tipo de robôs.

Desta forma, é importante conhecer as áreas de atuação que estão a falhar neste momento, para que este seja independente do PowerMill. Estas estão relacionadas com a programação de ferramentas matemáticas, desde matrizes de transformação e rotação, relacionadas com os planos de trabalho, a transformação de vetores ijk em ângulos Euler.

Para além disso, seria importante que o *software* fornecesse uma visualização tridimensional da estratégia criada, para além de criar simulação pré execução do programa no computador, por razões de segurança. Por mais que o braço de metrologia apresente sete eixos, mais um do que o robô, e seja possível prever algumas singularidades, estas numa serão completamente evitadas ao criar a estratégia.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] Norcam. (2019). *Sobre*. Available: https://www.norcam.pt/sobre/

[2] IFR, "Executive Summary World Robotics 2018 Industrial Robots," ed, 2018.

[3] P. Abreu, "Robótica Industrial - Fundamentos e Aspectos Tecnológicos da Robótica," ed. Faculdade de Engenharia da Universidade do Porto, 2002.

[4] B. A. T. Couto, "Aplicação e desenvolvimento em células de fabrico por maquinagem robotizada," Departamento de Engenharia Metalurgica e de Materiais, Faculdade de Engenharia da Universidade do Porto, 2012.

[5] J. Ferreira, "Programação de robôs para maquinagem," 2013.

[6] P. Almeida, "Programação de Robôs para Realização de Tarefas de Maquinação," Departamento de Engenharia Eletrotécnica, Politécnico do Porto, Instituto Superior de Engenharia do Porto, 2016.

[7] R. Costa, "Desenvolvimento de Célula Robótica para Polimento Automático de Moldes," Departamento de Engenharia Metalurgica e de Materiais Faculdade de Engenharia da Universidade do Porto, 2017.

[8] D. Angelino, "Conceção e Simulação de uma Célula Robótica para Operações de Acabamento de Guitarras feitas em Materiais Compósitos," Departamento de Engenharia Mecânica, Faculdade de Engenharia da Universidade do Porto, 2012.

[9] A. J. Alvares, J. S. Toquica, E. Jose Lima, and M. H. S. Bomfim, "Retrofitting of ASEA IRB2-S6 industrial robot using numeric control technologies based on LinuxCNC and MACH3-MatLab," pp. 2148-2153, 2017.

[10] V. Santos, "Robótica Industrial - Apontamentos Teóricos," in *Departamento de Engenharia Mecânica*, ed. Universidade de Aveiro, 2003.

[11] V. Romano and M. Dutra, "Capítulo 1 - Introdução à Robótica Industrial," in *Robótica Industrial - Aplicação na Indústria de Manufatura e de Processos*, E. E. Blucher, Ed., 2002.

[12] Yaskawa, "YRC1000 General Operator's Manual," ed, 2017.

[13] F. Campos, A. Filho, and A. Pin, "Estudo e Modelagem Computacional de um Robô Aplicado em Processos," presented at the Nono Simpósio de Mecânica Computacional, Universidade Federal de São João Del-Rei, 2010.

[14] V. Carrara, Introdução à robótica industrial, S. d. I. e. D. (SID), ed., São José dos Campos - Brasil: Instituto Nacional de Pesquisas Espaciais - INPE, 2015. [Online]. Available.

[15] A. Moraes, N. Júnior, and J. Pecegueiro, "Curso Técnico em Mecatrônica - Robótica," ed. SENAI-SP, 2003.

[16] N. Rodrigues, "Célula Robótica Industrial aplicação de ferramentas CAD CAM na programação de robôs," Departamento de Engenharia Mecânica, Faculdade de Engenharia da Universidade do Porto, 2011.

[17] A. Verl, A. Valente, S. Melkote, C. Brecher, E. Ozturk, and L. T. Tunc, "Robots in machining," *CIRP Annals,* 2019.

[18] Norcam, "PowerMILL - Curso de Formação Iniciado," ed, 2018.

[19] G. Ionut and G. Adrian, "Simulation Techniques in CAD-CAM Processing by Milling of Surfaces on NC Machine-Tools," 2013.
[20] CNCCookbook. (2016). *Results of the 2016 CNCCookbook CAM Survey*. Available: https://www.cnccookbook.com/results-2016-cnccookbook-cam-survey/?fbclid=IwAR1_BDZPDefROlLPDFsPcYs5HSdkhWpux067Fdv29u3vmDcWzkqdkQzajfE
[21] A. Gomes, "A metrologia em diferentes equipamentos," Mestrado em Engenharia Mecânica - Produção Industrial, Escola Superior de Tecnologia e Gestão - Instituto Politécnico de Leiria, 2017.
[22] RPS, "User Guide RPS EVO7 Articulated Measuring Arms," ed.
[23] A. Dourado, "Cinemática de Robôs Cooperativos," Engenharia Mecânica, Universidade Federal de Santa Catarina, Florianópolis, 2005.
[24] Renishaw, "Styli and accessories," ed, 2015.