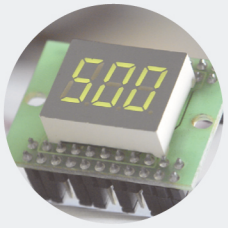
INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES






Robô para emulação de tarefas de pintura manual

CLÁUDIA MANUELA COUTO RIBEIRO
novembro de 2018

POLITÉCNICO
DO PORTO

Robô para emulação de tarefas de pintura manual

Cláudia Manuela Couto Ribeiro

Mestrado em Engenharia Eletrotécnica e de Computadores
Área de Especialização de Automação e Sistemas
Departamento de Engenharia Eletrotécnica
Instituto Superior de Engenharia do Porto
2018

Este relatório satisfaz, parcialmente, os requisitos que constam da Ficha de Unidade Curricular de Tese/Dissertação, do 2º ano, do Mestrado em Engenharia Eletrotécnica e de Computadores

Candidato: Cláudia Manuela Couto Ribeiro, Nº 1130445, 1130445@isep.ipp.pt
Orientação científica: Manuel Fernando dos Santos Silva, mss@isep.ipp.pt

Mestrado em Engenharia Eletrotécnica e de Computadores
Área de Especialização de Automação e Sistemas
Departamento de Engenharia Eletrotécnica
Instituto Superior de Engenharia do Porto
Outubro 2018

Para os meus pais.

Agradecimentos

As metas e ambições que temos na vida são suportadas pelo trabalho, pela motivação, mas também por quem caminha ao nosso lado.

Em primeiro lugar quero agradecer ao Professor Manuel Silva, pela oportunidade de execução deste projeto, assim como a sua disponibilidade e orientação ao longo da dissertação.

Agradeço ao Professor Miguel Silva, do Laboratório de Automação e Robótica Industrial do Departamento de Engenharia Mecânica, pelo apoio e auxílio.

Um agradecimento ao Nelson Martins pela sua prestabilidade e ajuda indispensável.

Quero agradecer à minha irmã Márcia por todos os momentos de confidência e amizade.

Ao meu namorado Ricardo, por todo amor e cumplicidade. Por percorreres a meu lado este percurso e, nos momentos mais difíceis, seres o meu apoio. Obrigada pelas palavras doces, pelos abraços calorosos e, sobretudo, por fazeres os meus dias mais felizes.

Um agradecimento especial aos meus pais, por todo o apoio, carinho e confiança ao longo destes anos. Por serem a principal razão de chegar onde estou. Sem vocês nada seria possível. Obrigada!

Resumo

A procura por soluções robóticas na indústria tem vindo a crescer nos últimos anos, devido à necessidade de desenvolvimento e construção de soluções que permitam competir com o mercado global. Uma vez que o mercado é caracterizado: pela sua globalização e concorrência, alta variedade de produtos e ciclos de vida curtos, surge a necessidade de criar sistemas mais flexíveis e ágeis. Esta flexibilidade carece de facilidade de programação dos sistemas, melhoria das interfaces homem-máquina, comunicações mais rápidas e descentralizadas e ferramentas de programação e protocolos *standard*.

As ferramentas *Computer Aided Design* (CAD) são cada vez mais utilizadas na indústria, e desta forma têm sido exploradas as suas capacidades na área da robótica. Em vista disso, esta dissertação aborda o conceito de pintura artística robotizada de desenhos importados a partir de um *software* CAD. O objetivo centra-se na forma como os robôs manipuladores podem ser utilizados na pintura manual e como deve ser a programação destes face à importação dos desenhos CAD. Desta forma, o propósito principal consiste no estudo e desenvolvimento de um programa que efetue a conversão do ficheiro CAD para a linguagem de programação do robô (RAPID), assim como analisar quais os parâmetros da pintura mais significativos para procedimento.

Portanto, foi necessário realizar um estudo teórico da robótica aplicada à pintura artística, dos formatos de exportação de ficheiro CAD e da estrutura da linguagem de programação RAPID. Sendo durante a implementação feita a análise das características da pintura e dos parâmetros a ter em conta nomeadamente, a variação de forma do pincel, a técnica de limpeza deste e a forma de aplicação da tinta.

Com a realização deste projeto constata-se que as ferramentas CAD têm um relevo a nível industrial devido à sua facilidade de desenvolvimento, começando a ser utilizadas como interface homem-máquina. É expetável que continuem a existir avanços nesta área devido às vantagens inerentes dos robôs.

Palavras-Chave

Robô manipulador, Robô e pintura a pincel, CAD, AutoCAD, DXF, RobotStudio, Linguagem programação RAPID

Abstract

Demand for robotic solutions in the industry has grown in recent years because of the ability to develop and build solutions that compete with the global marketplace. As the market is characterized by its globalization and competition, the high quality of products and life cycles, there is a need to create more flexible and agile systems. This concept needs: ease of system configuration, faster, decentralized man-machine machine interfaces, and standard programming tools and protocols.

CAD tools are increasingly used in industry, and has been explored their capabilities in robotics. Thus, this dissertation approaches the concept of robotized artistic painting of CAD software imported drawings. The goal is to focus in on how manipulator robots can be used in manual paintingand what should be the programming of the robot against the import of CAD drawings.

Therefore, the main purpose is to study and develop a program to converts the CAD file to the robot programming language (RAPID) as well as to analyze which painting parameters are most significant for the procedure. Therefore it was necessary to carry out a theoretical study of robotics applied to artistic painting, the CAD file export formats and the structure of the RAPID programming language. During the implementation, was made the analysis of characteristics of the painting and the required parameters, namely, the brush shape variation, the brush cleaning technique and the way of applying the ink.

In the realization of tihs project it was verified that the CAD tools are importat at industrial level due to, ease od development, starting to be used as human-computer interface. It is expected the there are still advances in this area because of the inherent advantages of robots.

Keywords

Robot manipulator, Robot and brush painting, CAD, AutoCAD, DXF, RobotStudio, RAPID programming language

Conteúdo

Agradecimentos **i**

Resumo **iii**

Abstract **v**

Índice **vii**

Índice de Figuras **ix**

Índice de Tabelas **xi**

Acrónimos **xiii**

1 Introdução **1**

1.1 Contextualização 1

1.2 Objetivos 2

1.3 Calendarização 2

1.4 Organização da Dissertação 3

2 Conceitos Teóricos **5**

2.1 Robótica 5

2.1.1 História e Evolução 5

2.1.2 Aplicações da robótica na pintura 8

2.2 Robôs Manipuladores 10

2.2.1 Estrutura mecânica 11

2.2.2 Atuador final 14

2.2.3 Controlador 15

2.2.4 Atuadores 16

2.2.5 Sensores 16

2.2.6 Especificações do Robô Manipulador 16

2.3 Tipos de programação 17

2.4 Robótica e Pintura Artística 17
2.4.1 Caso de estudo: robô de pintura a pincel através de *feedback* visual 20
2.5 Comunicação da Informação CAD 22
2.5.1 Componentes de um sistema CAD 24
2.5.2 Formatos 24

3 Arquitetura do Sistema **29**
3.1 Especificação do problema 29
3.1.1 *Software* CAD 30
3.1.2 Programa de conversão 30
3.1.3 Robô 30
3.2 Proposta para o Sistema 31
3.2.1 Formato DXF 31
3.2.2 Linguagem programação RAPID 35

4 Implementação Prática **39**
4.1 Programa de conversão de código DXF para linguagem RAPID . . . 39
4.2 Simulação do funcionamento do sistema RobotStudio 42
4.2.1 Definição da ferramenta de trabalho 42
4.2.2 Definição dos objetos de trabalho 44
4.3 Modelização geométrica no AutoCAD 46

5 Testes Efetuados e Resultados Obtidos **49**
5.1 Testes Efetuados 49
5.2 Resultados Obtidos 53

6 Conclusões **57**
6.1 Conclusões do Trabalho 57
6.2 Ideias para Desenvolvimentos Futuros 58

Referências Bibliográficas **59**

A *Datasheet* do robô ABB IRB120 **63**

B Programação C **67**

C Fluxogramas da extração de dados do ficheiro DXF **77**

Lista de Figuras

1.1 Calendarização do projeto 3

2.1 Estudo de um robô antropomórfico da autoria de Leonardo Da Vinci 6
2.2 Positioning or Manipulating Apparatus 7
2.3 Unimate 8
2.4 Patente do aparelho de controlo de posição de Pollard 9
2.5 Primeiro robô de pintura a pistola 10
2.6 Diagrama de blocos de um sistema robótico 11
2.7 Estrutura de um robô industrial 12
2.8 Representação gráfica do tipo de juntas 12
2.9 Configurações físicas do robôs 13
2.10 Garra mecânica 14
2.11 Controlador IRC5 da ABB e a consola 15
2.12 Volume de trabalho do robô ABB IRB120 16
2.13 Graus de liberdade do braço humano 17
2.14 Harold Cohen e a AARON 18
2.15 ArtSBot 19
2.16 Robô bitPaintr 20
2.17 Etapas do sistema 21
2.18 Garra e pincel 21
2.19 Variação dos parâmetros do pincel 22
2.20 Influência na pintura de diferentes parâmetros do pincel 22
2.21 Diagrama de Causa e Efeito 23
2.22 Componentes de um sistema CAD 24
2.23 Formato ficheiro CAD 25
2.24 Formato exemplo do ficheiro IGES 26
2.25 Facilidade de transferência de formatos CAD 27

3.1 Objetivo geral 29
3.2 ABB IRB120 e controlador IRC5 30

3.3 Arquitetura 31
3.4 Exemplo da representação de uma linha em formato DXF 32
3.5 Exemplo da representação de um arco em formato DXF 33
3.6 Exemplo da representação de um ponto em formato DXF 33
3.7 Exemplo da representação do parâmetro cor em formato DXF 34
3.8 Exemplo de identificadores no RAPID 35
3.9 Divisão de um programa RAPID 36
3.10 Exemplo da estrutura de um *target* na linguagem RAPID 36
3.11 Sintaxe de um MoveL na linguagem RAPID 37
3.12 Sintaxe de um MoveC na linguagem RAPID 37

4.1 Fluxograma do programa de conversão do código DXF para linguagem RAPID 40
4.2 Transformação do arco do formato DXF para RAPID 41
4.3 Célula modelada em RobotStudio 42
4.4 Criação do TCP da ferramenta 43
4.5 Medição das coordenadas do TCP da ferramenta 43
4.6 Sistema de coordenadas do utilizador e sistema de coordendas do objeto 44
4.7 *WorkObjects* do sistema 45
4.8 *Targets* da tinta 45
4.9 Definição do limite do espaço de trabalho do AutoCAD 46
4.10 Índice de cores do AutoCAD 47
4.11 Frascos de tinta 47

5.1 Teste *targets* da tinta e recipiente da água 49
5.2 Objetos de pintura 50
5.3 Fluxograma de reorganização do vetor das cores 50
5.4 Teste de diferentes tintas 51
5.5 Fluxograma de recarga de tinta 51
5.6 Teste de recarga de tinta 52
5.7 Teste pintura de letra 53
5.8 Teste pintura de logótipo 54
5.9 Teste pintura da estrela do norte 55

Lista de Tabelas

2.1 Tipo de juntas 12

3.1 Secções do ficheiro DXF 32
3.2 Código de cores DXF 34

Acrónimos

ABB	Asea Brown Boveri
ARLA	ASEA *Robot Language*
ASCII	American Standard Code for Information Interchange
CAD	*Computer Aided Design*
CAM	*Computer Aided Manufacturing*
CN	Controlo Numérico
DOF	*Dregrees Of Freedom*
DXF	*Drawing Exchange Format*
HCI	*Human-Computer Interaction*
IDE	*Integrated Development Environment*
IGES	*Initial Graphics Exchange Specifications*
ISO	International Organization for Standardization
PUMA	*Programmable Universal Machine for Assembly*
RGB	*Red Blue and Green*
SCARA	*Selective Compliance Assembly Robot Arm*
STEP	*STandard for External representation of Produt data*
TCP	*Tool Center Point*

1 Introdução

"Art has always existed in complex, symbiotic and continually evolving relationship with the tecnological capabilities of a culture." [1]

1.1 Contextualização

Desde o final da Segunda Guerra Mundial que houve uma crescente evolução de técnicas e heurísticas para a produção industrial. Estes procedimentos eram a consequência da mudança de mercado e elevada concorrência, que exigiam elevada variedade de produtos, tempos de produção cada vez menores e baixos custos. Face a isto, tornou-se fundamental a adaptação e desenvolvimento de sistemas de produção automatizados e robotizados.

A partir dos anos 80 o interesse e procura por robôs industriais e sistemas de aplicação aumentou constantemente. Pela sua fiabilidade, flexibilidade e custo justificado, continuou, e estima-se que continuará, a crescer. Por conseguinte, e de forma a dar resposta à alta exigência da indústria, torna-se primordial que a interação entre o homem e o robô seja mais simples e rápida. O objetivo é que o homem possa ensinar o robô de forma intuitiva, isto é, imitando a forma de comunicação dos humanos, por exemplo, utilizando sistemas *Computer Aided Design* (CAD), gestos ou explicação verbal.

Nas primeiras aplicações da robótica as tarefas eram simples e com um pequeno número de pontos. Porém, aquando à chegada dos robôs de pintura e soldadura o processo tornou-se mais complexo e demorado, surgindo a necessidade de procura por métodos mais eficientes de marcar trajetórias. Ultimamente, estudos feitos, revelam que os *softwares* CAD podem ser grandes aliados dos robôs na preparação dos programas [2, 3].

Apesar das aplicações mais faladas serem na área da indústria, a robótica e os sistemas inerentes têm ampliado para outros domínios distintos, melhor dizendo, agricultura, construção, medicina, educação e mesmo na arte. A integração entre arte e tecnologia não é um conceito novo, uma vez que a arte é feita das tecnologias do seu tempo, porém é um assunto controverso, pois associar arte com robótica desafia toda a história da arte, uma vez que os robôs podem imitar a criatividade humana mas as emoções são algo intrínsecas ao artista e fazem com que determine o que vai criar.

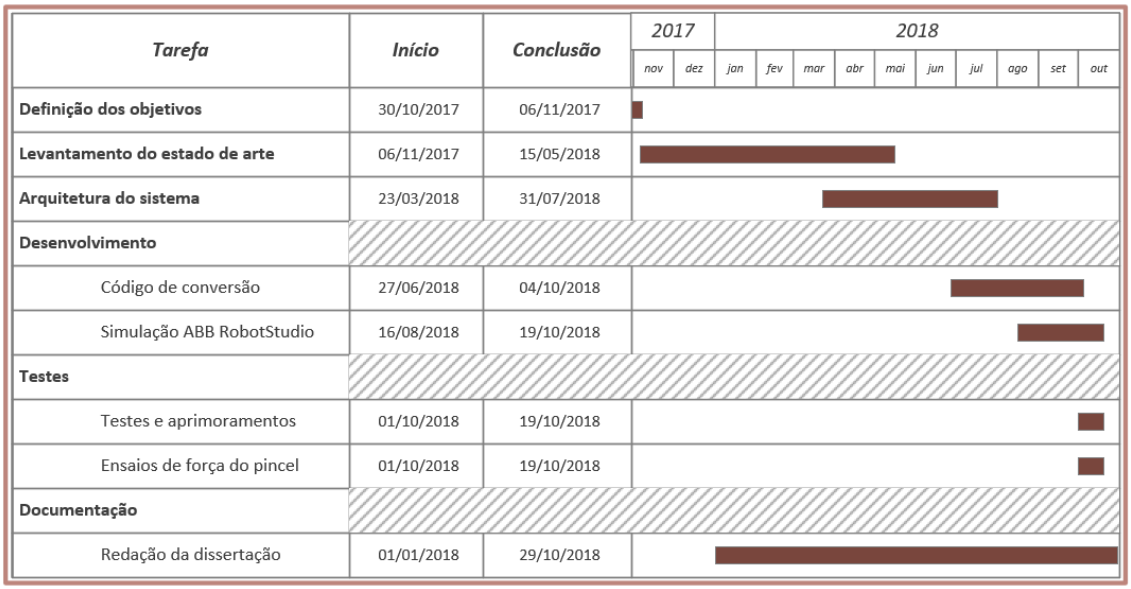
1.2 Objetivos

O objetivo principal deste trabalho consiste no estudo e desenvolvimento de um sistema robótico capaz de pintar a pincel, desenhos produzidos em CAD, de diferentes geometrias. Dado isto, para a realização deste objetivo, subdividiu-se o mesmo em várias tarefas, tais como:

- levantamento de aplicações similares já desenvolvidas nesta área;
- geração de um ficheiro, do desenho a pintar, recorrendo a um *softwares* de CAD a escolher;
- passagem da informação constante no ficheiro CAD para o controlador do robô;
- analisar a forma como deve ser posicionado o pincel, e a "força" a ser exercida sobre este, de forma a ser possível a execução de traços com diferentes caraterísticas;
- extração da informação do ficheiro CAD, de forma a converter o desenho a pintar num conjunto de linhas/curvas que possam ser interpretadas pelo controlador do robô em termos de *targets* e *paths* ;
- geração do programa do robô de forma a pintar a peça em questão, sendo possível o posicionamento da peça em posições arbitrárias, dentro de certos limites relacionados com o volume de trabalho do robô.

1.3 Calendarização

Para a realização do presente projeto foi estabelecido um conjunto de etapas a desempenhar. A definição dos objetivos e organização do plano de tarefas resultou na calendarização apresentada na Figura 1.1.



Tarefa	Início	Conclusão
Definição dos objetivos	30/10/2017	06/11/2017
Levantamento do estado de arte	06/11/2017	15/05/2018
Arquitetura do sistema	23/03/2018	31/07/2018
Desenvolvimento		
Código de conversão	27/06/2018	04/10/2018
Simulação ABB RobotStudio	16/08/2018	19/10/2018
Testes		
Testes e aprimoramentos	01/10/2018	19/10/2018
Ensaios de força do pincel	01/10/2018	19/10/2018
Documentação		
Redação da dissertação	01/01/2018	29/10/2018

Figura 1.1: *Calendarização do projeto*

1.4 Organização da Dissertação

A presente dissertação está dividida em 6 capítulos. No Capítulo 1 é feita uma breve introdução e contextualização do projeto.

O Capítulo 2 centra-se nos conceitos teóricos, onde são apresentados conceitos sobre a robótica, desde, história e evolução, características dos robôs manipuladores e aplicações da robótica na pintura. Sendo também abordado o sistema CAD e os seus formatos de troca de dados.

No terceiro Capítulo é descrita a especificação do problema bem como, a proposta de solução. Nesta é feita uma explicação do formato de arquivo de dados *Drawing Exchange Format* (DXF) e, sobre a linguagem de programação RAPID e a sua estrutura.

É no Capítulo 4 onde é abordada a implementação prática, sendo descrita a programação realizada, a simulação no *software* RobotStudio e algumas configurações necessárias no AutoCAD.

No capítulo seguinte, Capítulo 5, são apresentados testes efetuadas e quais os resultados obtidos.

Por fim, no Capítulo 6 são abordadas as conclusões finais e ainda algumas sugestões para trabalhos futuros.

2
Conceitos Teóricos

Neste capítulo são invocados aspetos da teoria da robótica, como a evolução histórica, noções e configurações dos robôs. Aborda-se, amplamente, o conceito da robótica na pintura e a ligação com sistemas CAD.

2.1 Robótica

"(...) a robótica é uma ciência de mecanismos engenhosos genéricos, de precisão, movidos por uma fonte permanente de energia e flexíveis, isto é, abertos às ideias, um estímulo à imaginação."[4]

2.1.1 História e Evolução

A história da robótica teve destaque num mundo de ficção e inspiração, que de alguma forma passou para a realidade dos nossos dias.

Inicialmente, Karel Capek trouxe-nos a palavra "robô", em 1921, no romance "*Rossum's Universal Robots*". Capek criou substitutos automatizados para os trabalhadores humanos, dandos-lhe aspeto humano com capacidades avançadas.

Posteriormente Isaac Asimov, escritor e cientista americano, contribuiu para a criação de um pensamento criativo no que poderia vir a ser a robótica. Foi na sua obra de ficção científica, "*Runaround*", que introduziu a palavra "robótica", onde era descrita a arte e ciência dos robôs. Foi também em diversas obras que Asimov introduziu as três leis da robótica. No entanto, em 1985, modificou a lista incluindo a lei zero [3, 5]:

- Lei zero: Um robô não pode fazer mal à humanidade e nem, por inação, permitir que esta sofra algum mal.
- 1ª lei: Um robô não pode fazer mal a um humano e nem, por inação, permitir que algum mal lhe aconteça.

- 2ª lei: Um robô deve obedecer às ordens dadas pelos humanos, exceto quando estas contrariarem outra lei de ordem superior.

- 3ªlei: Um robô deve proteger a sua existência, desde que a sua proteção não contrarie alguma outra lei de ordem superior.

O sonho do homem por máquinas que executem tarefas humanas vem de tempos mais remotos. Grandes pensadores da nossa história projetaram e construiram as suas ideias de mecanismos automatizados. Como Arquitas de Tarento, matemático grego, que criou um pombo em madeira movido a vapor, no ano de 400 a.C.

Leonardo Da Vinci (1452-1519) dedicou-se profundamente ao estudo da robótica, projetando mecanismos com movimentos e funções humanas. Grande parte dos seus estudos encontram-se no seu livro *Codex Atlanticus*, desde a anatomia humana a vários mecanismos mecânicos. Leonardo realizou vários estudos, muitos ficando incompletos, projetou um soldado robô com movimentos feitos por engrenagens (Figura 2.1) e até um leão autómato.

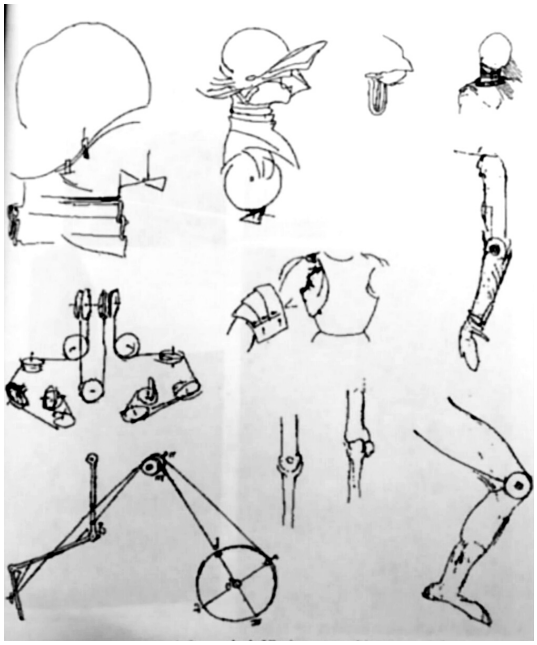


Figura 2.1: *Estudo de um robô antropomórfico da autoria de Leonardo Da Vinci [4]*

Nicola Tesla (1845-1943), também foi conhecido pela sua contribuição na evolução de mecanismos, inventou um submarino telecomandado (1898), usando impulsos hertzianos codificados.

Apesar dos diversos estudos e invenções da história, o desenvolvimento e criação de vários mecanismos era irreal e impraticável, uma vez que era fundamental o acesso a fontes de alimentação, materiais e procedimentos mais precisos. Com a evolução tecnológica, ideológica e disponibilidade de recursos, cresceram o número de descobertas, codificações e reproduções mecânicas e eletrónicas. O surgimento

do controlo numérico (CN) e do controlo remoto deram um novo rumo ao desenvolvimento das máquinas programáveis. O controlo numérico, desenvolvido na década de 40 e atualmente em desuso, era uma linguagem padronizada numa fita perfurada, em que a presença ou ausência de furo correspondia a um determinado dado ou posição. O controlo remoto veio trazer a possibilidade de manipular e controlar através de uma consola os movimentos especificados pelo operador. A sua grande utilidade vem com o manuseamento de substâncias perigosas tais como, substâncias tóxicas, radioativas e mesmo a altas temperaturas. Esta combinação de tecnologias formam a base do robô moderno [6].

É de 1954 a data da patente do primeiro dispositivo robótico, criado pelo inventor britânico Cyril W. Kenward (Figura 2.2). Era um dispositivo hidráulico de seis eixos, com braço duplo montado em pórticos, mecanizado por pinças. Uma característica deste dispositivo foi a condução interna hidráulica, que deveria ter sido mais estudada, mesmo para os robôs hidráulicos modernos [7, 8].

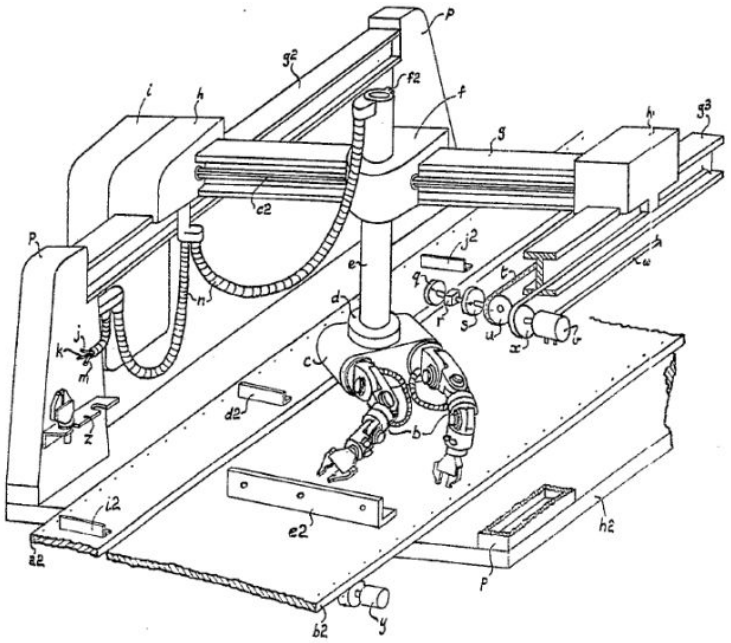


Figura 2.2: *Positioning or Manipulating Apparatus [9]*

Apesar de Cyril W. Kenward ter sido o primeiro inventor a patentear um dispositivo robotizado, o conceito moderno de robô foi desenvolvido por George Devol e Joseph Engelberger. Em 1954 Devol patenteou o primeiro braço robótico programável, *Programmed article tranfer*. Juntamente com Engelberger, em 1956, criaram a primeira empresa dedicada à robótica, Unimation (*Universal Automation*), sendo em 1961 desenvolvido o primeiro robô industrial, o Unimate, que começou a fazer descarregamento de uma máquina de fundição na Ford Motor Company (Figura 2.3).

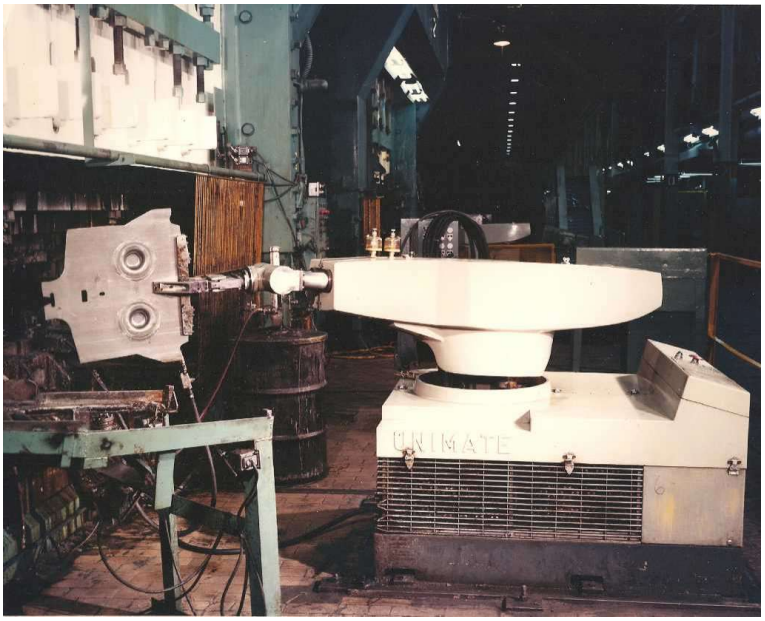


Figura 2.3: *Unimate [10]*

Em 1978 foi criado o PUMA - Máquina Programável Universal de Montagem (*Programmable Universal Machine for Assembly*) pela Unimation com o suporte da General Motors. O PUMA é uma braço articulado que rapidamente se tornou popular a nível industrial [8].

A partir de 1980 a indústria da robótica entrou numa fase de elevada expansão. Até hoje, o desenvolvimento das aplicações da robótica ficaram subjugadas ao avanço tecnológico, como a utilização de novos sensores, novas formas de programação, a utilização de sistemas computorizados e a *internet*, sistemas de visão, novas otimizações nas estruturas mecânicas e sistemas de fabrico, tudo isso permitindo produzir sistemas mais ágeis e eficientes.

2.1.2 Aplicações da robótica na pintura

O processo da pintura na história da indústria cresceu com o progresso tecnológico e com isto, as suas áreas de aplicação também. A pintura robotizada é um conceito conhecido pela sua repetibilidade e precisão no revestimento de peças. Os primeiros mecanismos de pintura foram desenvolvidos no final da década de 1930, data das primeiras patentes dos EUA no assunto. O *Position Controlling Apparatus* patenteado em 1938 por Willard Pollard, consiste num robô paralelo de três ramificações, sendo os três braços atuados por motores rotativos que através de juntas universais a posição da ferramenta é determinada (Figura 2.4). A patente de Pollard era um robô destinado a pintura por pulverização, no entanto, nunca foi construído [11].

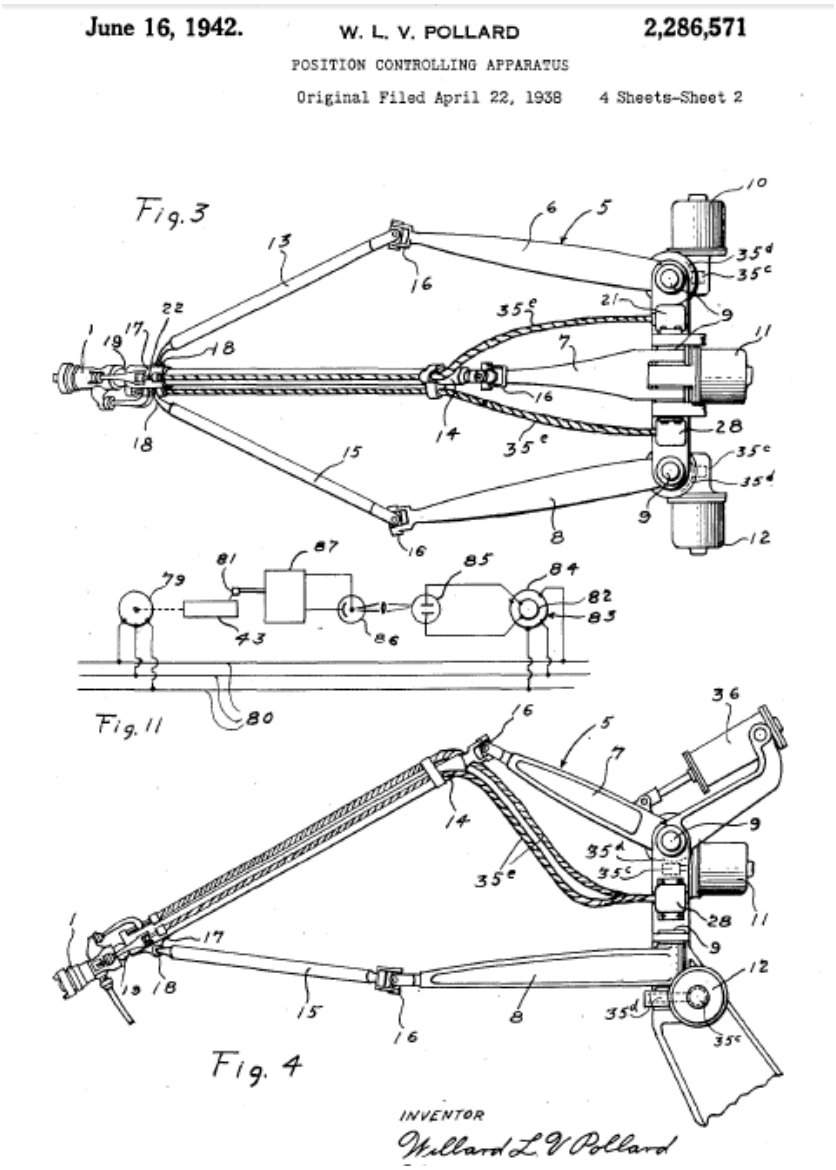


Figura 2.4: *Patente do aparelho de controlo de posição de Pollard [11]*

Foi em 1944 que Harold Roselund patenteou, juntamente com a empresa DeVilbiss, um robô de pintura a pistola. Não era um robô paralelo e usava o sistema de controlo proposto por Pollard.

Contudo, apesar de já existirem patentes para mecanismos e robôs de pintura, o primeiro robô de pintura a pistola foi desenvolvido por Ole Molaug na empresa Trallfa (atual Asea Brown Boveri (ABB)), entre 1965 e 1967.

A Trallfa era uma empresa de carrinhos de mão e equipamentos de transporte fundada em 1941 por Nils Underhaug. A fábrica foi crescendo de forma constante, surgindo novos projetos, porém, os carrinhos eram pintados à mão e, apesar de haver vários trabalhadores por turnos e equipamentos modernos, a pintura era o ponto de

estrangulamento do processo. Ole Molaug, engenheiro da Trallfa, desenvolveu a ideia de um robô de pintura a pistola, sendo aprovada por Underhaug em 1964. Foi em 1967 que o robô foi testado na fábrica pintando as caixas dos carrinhos de mão, obtendo um excelente resultado (Figura 2.5).

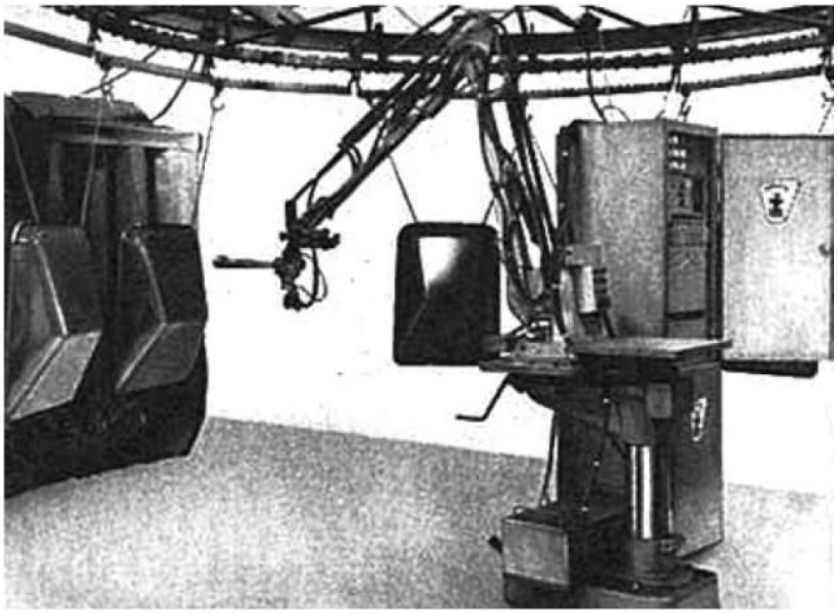


Figura 2.5: *Primeiro robô de pintura a pistola [12]*

Posteriormente, a Trallfa começou a entrar em produção com o robô, e em 1969, entregou na Suécia o robô industrial de pintura a pistola para pintar banheiras. Passando a ser a principal empresa fornecedora de robôs para aplicações de pintura pulverizada, conhecida hoje em dia por ABB. [12]

2.2 Robôs Manipuladores

Os robôs manipuladores são sistemas muito utilizados na indústria, devido à capacidade de se posicionarem e orientarem no espaço com a ferramenta ou garra. Estes podem tomar diferentes tamanhos e configurações, porém a estrutura de um sistema robótico industrial é normalmente constituída por:

- estrutura mecânica/manipulador;
- atuador final;
- controlador;
- atuadores;
- sensores.

Na Figura 2.6 é possível verificar as conexões dos diferentes elementos de um sistema robótico. Trata-se de um sistema de controlo em malha fechada, uma vez que compara o estado do sistema com os valores/movimentos desejados.

O sistema processa os sinais de entrada e converte estes numa ação (atuadores), gerando o movimento do manipulador. Aquando do funcionamento e através dos dados dos sensores, o sistema verifica o estado atual, sendo esta medida comparada com o valor desejado, resultando num erro. Com esta comparação o controlador fará ajustes de modo a que o erro seja nulo e o robô se movimente até à posição desejada.

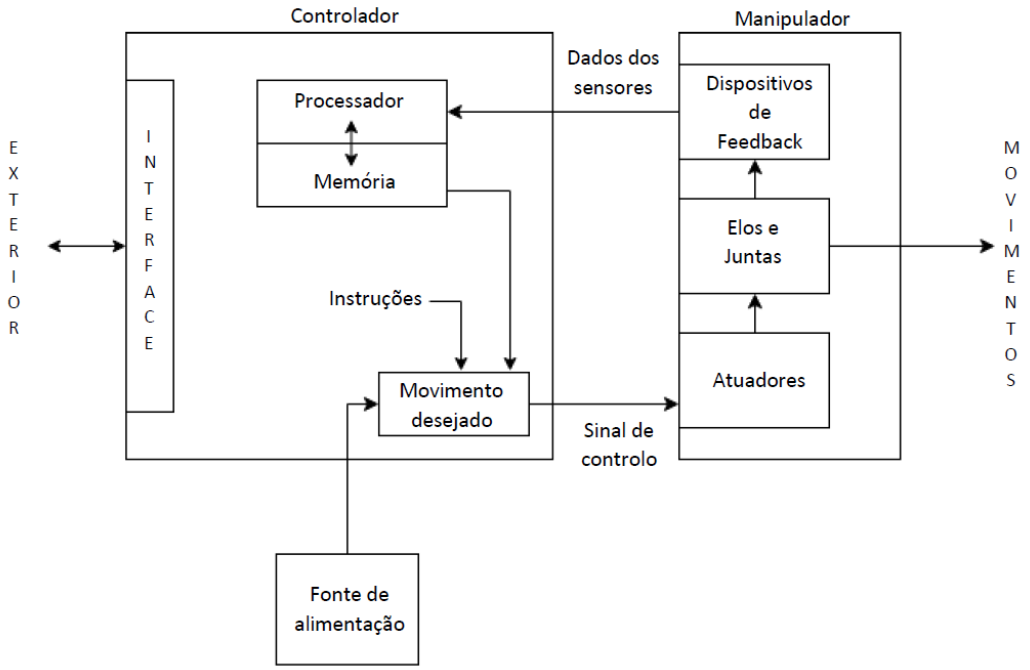


Figura 2.6: *Diagrama de blocos de um sistema robótico [13]*

2.2.1 Estrutura mecânica

A estrutura mecânica é a parte do manipulador associada ao posicionamento espaço físico cartesiano. Este é constituído por elementos rígidos, elos (*links*), ligados entre si pelas juntas (*joints*) (Figura 2.7). Os elos podem ter diversos tamanhos e formas, dependendo da aplicação do robô. As juntas permitem realizar o movimento, com o acionamento do sistema de controlo [14].

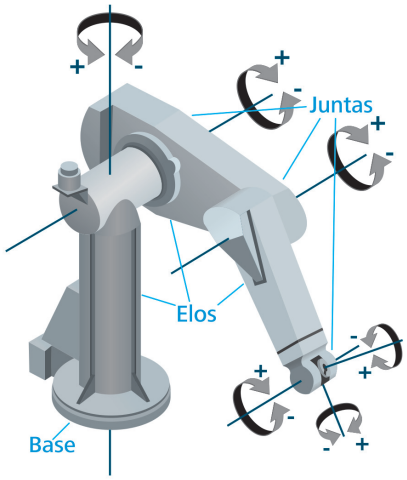


Figura 2.7: *Estrutura de um robô industrial [11]*

Os possíveis movimentos de um elo relativamente ao outro, deriva do tipo de junta que os une. O mesmo manipulador pode ter tipos de juntas distintos e, conforme as formas e tamanhos, determinam a anatomia do robô. As juntas podem ser divididas em diferentes tipos, como está apresentado na Tabela 2.1 e na Figura 2.8.

Tabela 2.1: *Tipo de juntas [15]*

Junta Linear ou Prismática (L ou P)	Junta Rotacional (R)	Junta de Torção (T)	Junta Revolvente (R)
Movimento linear	Rotação com eixo perpendicular aos eixos das duas ligações	Rotação com eixo paralelo aos eixos das duas ligações	Rotação com eixo paralelo à ligação de entrada e perpendicular à de saída

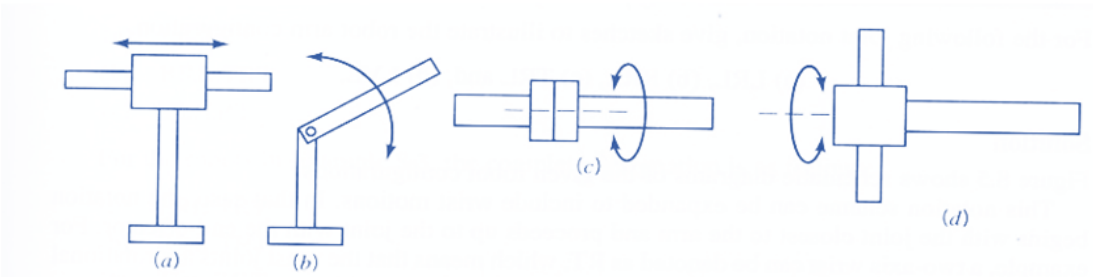


Figura 2.8: *Representação gráfica do tipo de juntas [16]*

Posto isto, a configuração física de um robô é determinada pelos diferentes tipos de juntas, existindo diferentes tipos de configurações de modo a conseguir corresponder às necessidades da indústria [17].

A configuração cartesiana realiza movimentos lineares num sistema de eixos cartesianos, sendo vantajosa para sua elevada precisão, grande volume de trabalho e grande capacidade de carga.

A configuração cilíndrica é a combinação de dois eixos ortogonais de movimento linear com um eixo rotativo, tendo uma estrutura simples e boa precisão de trabalho, possibilitando ainda, velocidade de operação elevadas.

A configuração *Selective Compliance Assembly Robot Arm* (SCARA) é uma combinação de eixos de movimento rotativo num plano horizontal com um movimento vertical; esta configuração é mais vocacionada para operações de montagem, devido à sua estrutura rígida, elevada velocidade de trabalho, precisão e repetibilidade.

A configuração articulada ou revoluta utiliza exclusivamente eixos de movimento rotativo, e é utilizada para tarefas de programação mais complexas. Possui grande volume de trabalho aliado com a sua elevada manobrabilidade.

A configuração esférica combina dois movimentos de rotação com um movimento linear, sendo diferenciada pelo enorme volume de trabalho, velocidades elevadas, assim como precisão e repetibilidade.

A configuração paralela é formado por duas plataformas, a base, que se encontra fixa, e a móvel, ligadas entre si por dois ou mais eixos.

Na Figura 2.9 é possível observar os tipos de configurações dos robôs manipuladores.

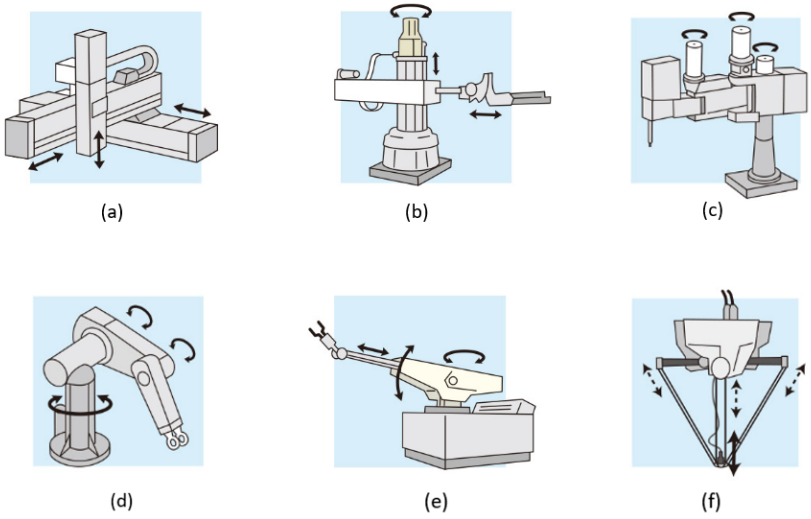


Figura 2.9: *Configurações físicas do robôs: a) cartesiana, b) cilíndrica, c) SCARA, d) articulada, e) esférica, f) paralela [18]*

2.2.2 Atuador final

O atuador final está ligado à extremidade do punho de forma a executar uma determinada função. Dependendo da operação a realizar é necessário um determinado atuador final, ou seja, consoante o tipo de objeto a ser manipulado e o tipo de cuidado a ter no seu manuseamento. Podendo ser tipo garra (*gripper*) para manusear e manipular objetos ou tipo ferramenta (*tool*).

As garras podem ser de diferentes tipos [15]:

- mecânicas

  As garras mecânicas são constituídas por dois ou mais dedos, sendo capaz de abrir e fechar os dedos e, quando fechados, conseguir exercer a força suficiente para segurar os objetos. Normalmente as garras são geometricamente adaptadas à forma geométrica aproximada do objeto a manipular, evitando possíveis deslocações destes, como é ilustrado na Figura 2.10.

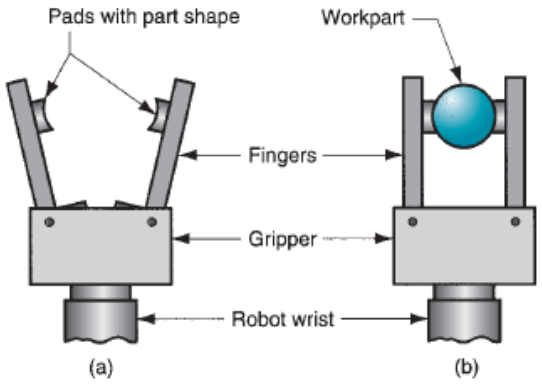


Figura 2.10: *Garra mecânica [19]*

- vácuo

  As garras de vácuo ou sucção são utilizadas para manipular objetos com superfícies lisas e planas, sendo utilizadas ventosas, que podem ser de diferentes materiais dependendo da superfície do material a manipular. Quando o objeto a manipular é constituído por um material mais rígido, as ventosas são fabricadas em material elástico (borracha ou plástico) de forma a se adaptarem melhor ao objeto. Se o objeto a manipular for composto por um material mais macio, normalmente as ventosas são feitas com um material mais rígido, sendo o objeto a manipular a adaptar-se às ventosas.

- magnéticas

  As garras magnéticas são uma boa solução no manuseamento de materiais ferrosos. Estas podem conter eletroímans ou ímans permanentes. As garras magnéticas com eletroímans são mais fáceis de controlar, pois podem anular o magnetismo durante o manuseamento; já as garras com ímans permanentes,

apesar de não necessitarem de uma fonte permanente de energia, apresentam menor facilidade de controlo.

- adesivas

  As garras adesivas utilizam substâncias adesivas de forma a segurar o objeto, tendo principal aplicação com tecidos e materiais leves.

- agulhas

  As garras por agulhas são constituídas por agulhas mecânicas e são indicadas para manusear objetos macios e que possam ser perfurados, como por exemplo, têxteis, borrachas, plásticos, entre outros.

As ferramentas são atuadores finais cuja função é realizar um determinado trabalho sobre uma peça. Existe uma variedade de ferramentas na indústria, nomeadamente para maquinação e rebarbagem, soldadura, pistolas de pintura, corte a jato de água, entre outras.

2.2.3 Controlador

O controlador é o dispositivo que controla todos os movimentos do manipulador. Processa os dados fornecidos pelo manipulador, pelos sensores e também do operador, efetuando os cálculos algébricos necessários e envia os sinais de controlo aos atuadores de forma a posicionar o manipulador no local desejado.

Juntamente com o controlador existe uma consola portátil, *teach pendant*. Permitindo uma interação mais acessível entre o utilizar e o robô. Caracteriza-se pelo seu design agradável no ecrã tátil e um "*joystick*"3D.

Figura 2.11: *Controlador IRC5 da ABB e a consola [20]*

2.2.4 Atuadores

Os atuadores são os dispositivos que geram o movimentos das partes mecânicas do manipulador, através da conversão de energia em forças e binários. Estes podem ser elétricos, hidráulicos ou pneumáticos.

2.2.5 Sensores

São dispositivos utilizados para recolher dados do sistema e do manipulador, enviando ao controlador de modo a este efetuar as correções necessárias. Os sensores podem ser internos ou externos, isto é, os sensores internos fornecem informação sobre o estado do manipulador, nomeadamente posição, velocidade ou aceleração. Enquanto que o sensores externos comunicam sobre o ambiente, como sensores de força ou câmaras para deteção de objetos.

2.2.6 Especificações do Robô Manipulador

Os robôs manipuladores possuem características específicas, que variam consoante o tipo de manipulador e a sua aplicação. Características como volume de trabalho, graus de liberdade, precisão e repetibilidade são alguns dos pontos que a indústria tem em consideração quando efetua a aquisição de um robô.

O volume de trabalho, ou *workspace*, de um robô consiste no conjunto de pontos que podem ser alcançados pelo robô, sendo então, o espaço de todas as posições que o robô alcança no espaço tridimensional. Este conjunto de pontos depende do tipo de juntas do robô, do tamanho dos elos e da estrutura do robô. Assim, o alcance do robô corresponde à distância máxima da garra face à base [15].

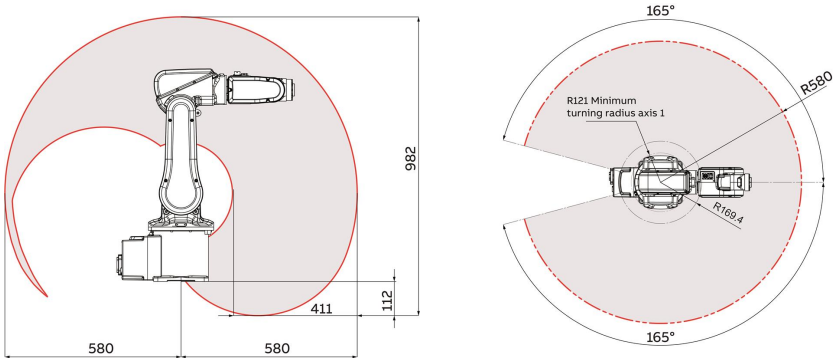
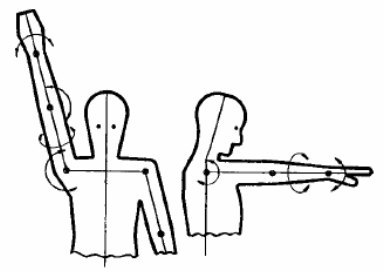


Figura 2.12: *Volume de trabalho do robô ABB IRB120 [21]*

O número total de movimentos independentes que o robô pode efetuar é denominado por graus de liberdade, *Degrees of Freedom* (DOF). Como é visível na Figura 2.13 cada braço humano dispõe de sete graus de liberdade.



Junta	*Tipo*	*Graus de liberdade (DOF)*
Ombro (*Shoulder*)	Esférica	3
Cotovelo (*Elbow*)	Rotacional	1
Pulso (*Wrist*)	Esférica	3

Figura 2.13: *Graus de liberdade do braço humano [14]*

A precisão é a capacidade do robô se posicionar no mesmo ponto dentro do volume de trabalho.

A repetibilidade traduz-se pela diferença de posição com que o robô volta a recolocar-se num ponto alcançado anteriormente. Esta é afetada tanto pela variabilidade do comportamento mecânico como pelos materiais a serem trabalhados.

Todos este conceitos são inerentes à escolha do tipo de robô. Atualmente, e com a competitividade do mercado, os robôs industriais devem deter elevada precisão, repetibilidade e velocidade, permitindo uma fácil integração com as restantes estruturas e, dispor de um controlo simplificado [15].

2.3 Tipos de programação

Ao longo do crescimento e desenvolvimento do sistemas robóticos, tornou-se essencial que a programação destes gerasse um código de controlo de uma forma simples e apelativa ao utilizador.

A programação dos robôs pode ser dividida em dois métodos distintos [15]:

- Programação *online*: envolve diretamente o robô e, apesar de ser um método simples de programar, possui algumas desvantagens como necessidade de interromper a célula do robô aquando a programação é realizada e grande possibilidade de erros.

- Programação *offline*: consiste em programar o robô sem ser necessário a sua utilização. Utilizando um sistema de computação para simular o programa e, somente quando isento erros, o programa é enviado para o robô. A grande vantagem deste método é não ser necessário interromper a célula para construir o programa e consequentemente, permitir a segurança do programador e reduzir de custos.

2.4 Robótica e Pintura Artística

A simbiose entre arte e tecnologia não é uma ideia recente. Esta área foi crescendo em diversos domínios, desenho e pintura, dança, música, teatro e cinema. São várias as áreas em que artistas e investigadores decidiram explorar o conceito das máquinas inteligentes.

A pintura é algo que nos acompanha desde a pré-história e é interpretada como uma manifestação artística, uma expressão/pensamento específico, contemporâneo ou abstrato. Contudo com o progresso tecnológico cresce o leque de áreas em estudo e a possibilidade de junção de determinadas ciências, nomeadamente, a robótica na pintura artística.

O pioneiro na arte de ensinar robôs a pintar é Harold Cohen, desde 1973, que vem desenvolvendo o sistema AARON [22], visivel na Figura 2.14. Cohen desenvolveu um *software* de pintura com base em inteligência artificial e construiu a máquina física. O objetivo era ensinar robôs a pintar como seres humanos, pelo que Cohen desenvolveu o modelo com uma lista de geometrias e a relação entre elas, nunca tendo mostrado uma imagem a AARON. Enquanto o sistema produz pinturas aleatoriamente, o estilo e a estética do desenho foram previamente programadas pelo homem, ou seja, o robô é inteiramente autónomo, na definição do tema da pintura, palete de cores, e composição da pintura. Com o tempo as suas pinturas foram melhorando e muitas vezes, sendo expostas em museus.

Figura 2.14: *Harold Cohen e a AARON [23]*

Muitos artistas seguiram as pisadas de Cohen e introduziram-se na pesquisa e desenvolvimento de sistemas tecnológicos para a arte. Benjamin Grosser, desenvolveu um sistema, denominado *Interactive Robotic Painting Machine* [24], em 2011. Este sistema utiliza inteligência artificial para tomar as suas decisões durante o decorrer de toda a pintura. Assim, enquanto pinta, o sistema ouve o próprio ambiente, e os sons captados funcionam como uma entrada no processo que influenciam as decisões de pintura.

Outro sistema de arte robótica é o ArtSBot [25] (Figura 2.15), desenvolvido por Leonel Moura. Este sistema é caracterizado por um grupo de robôs programados para serem autónomos. O sistema não tem uma meta definida, apenas se gere

pela aleatoriedade, porém a obra não tem menos mérito pois não é um resultado meramente aleatório, uma vez que o sistema é programado com lógica difusa, isto é, tudo o que foi pintado faz parte do processo de decisão do que será e como será a próxima decisão de pintura. O sistema é constituído por sensores, controlador e atuadores, e responde com uma série de comportamentos consoante a leitura do sensor perante a tela, desde a cor ou planos de cores e, também, obstáculos. As criações dos robôs resultam da interpretação da tela e do que nela contém, sendo então, um modelo artístico autónomo que tenta fazer a imitação da arte humana, um pensamento autêntico que muitas vezes resulta de um cadeia de ideias até à obra final.

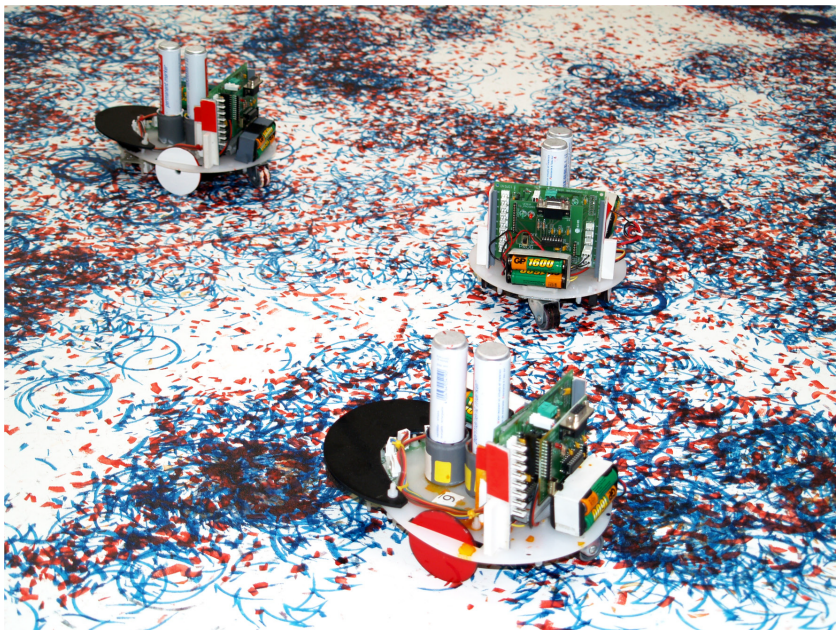


Figura 2.15: *ArtSBot [25]*

Um outro sistema que tenta imitar o processo de pintura humana é o e-David, desenvolvido por professores da Universidade de Konstanz, na Alemanha. O robô utiliza *feedback* visual para tentar imitar um pintor humano; então, a partir de uma imagem de entrada o sistema calcula as pinceladas necessárias para replicar o desenho e, posteriormente, pinta um conjunto de traços e verifica se estes representam com alguma precisão a imagem de entrada. Caso seja necessário, é gerado mais um conjunto de pinceladas até a pintura conseguir corresponder à imagem inicial. O sistema é constituído por um robô articulado com uma ferramenta no punho em que a extremidade contém um pincel, e este tem disponível uma palete de 24 cores e cinco pincéis distintos.

Pindar Van Arman, equitativamente desenvolveu um robô para pintura, o bitPaintr (Figura 2.16). Todo o sistema por detrás do bitPaintr é um algoritmo baseado em inteligência artificial que demorou mais de 10 anos a ser desenvolvido, onde é feito o *upload* de uma imgem e o robô pinta de forma autónoma. Pindar Arman defende que o futuro destes sistemas não é substituir o pintor mas, de certa forma,

ser um auxiliar do artista.

Figura 2.16: *Robô bitPaintr [26]*

Neste conceito surgem algumas discussões, nomeadamente a própria definição de robótica. Se, por um lado, há tradições culturais de arte realizadas tradicionalmente pelo homem, por outro lado, encontram-se tradições literárias e cinematográficas associadas a autómatos e robôs. Apesar das controvérsias que têm existido, muitos artistas defendem a introdução da robótica na arte, como ampliação dos seus limites de trabalho, uma vez que, cada artista desenvolve estratégias específicas ao explorar a robótica.

No entanto, tal como as tecnologias passadas, alguns artistas sentirão a necessidade de utilizar a inteligência das máquinas como parceiro, assim como outros só irão utilizar mecanismos e técnicas tradicionais. O simples fato de explorar as tecnologias na arte trará uma experiência positiva ou negativa, que, de certa forma, trará conclusões importantes quanto à interação homem-máquina, *human-computer interaction* (HCI). A robótica também cresceu, e tem vindo a ser desenvolvida, cada vez mais, com base nas técnicas HCI, isto é, na modelação cognitiva [1, 27, 28].

2.4.1 Caso de estudo: robô de pintura a pincel através de *feedback* visual

Neste caso é descrito o estudo do método de manipulação de um pincel num robô para pintura, onde o maior desafio é o facto da ponta do pincel ser deformável e ser necessário determinar o controlo adequado para a pintura. Para isso, é necessário que os métodos sejam de controlo fechado, usando visão computacional e sensores de força [29].

Deste modo, o sistema de pintura é dividido em três partes:

- obtenção do modelo 3D;
- construção da figura modelo;
- robô executa a pintura.

O processo inicia-se com a reconstrução da forma da imagem captada pelas câmaras; após este reconhecimento segue-se a interpretação do modelo, isto é, a extração da forma do objeto, de forma a criar uma silhueta possível para o robô pintar.

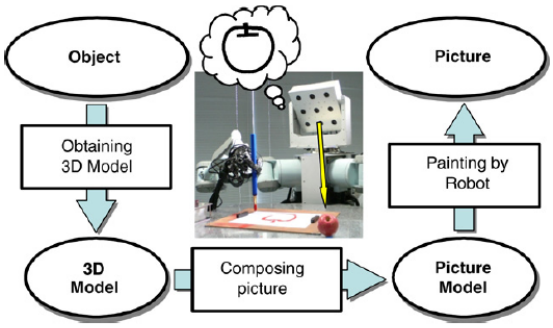


Figura 2.17: *Etapas do sistema [29]*

Um dos desafios deste caso é a utilização de uma garra universal (*multi-fingered hand)* surgindo questões quanto à forma como a garra segura no pincel e a força necessário para que este não se mova durante o trabalho.

O pincel é suportado através de quatro pontos (Figura 2.18), três dedos da garra e a raiz do polegar, de forma a imitar a mão humana. Mas, como tal, tem algumas limitações, como por exemplo, na direção de pintura, o pincel fica instável se este se mover na diagonal, mas se for na horizontal este é estável; desta forma, é necessário ter em consideração estes factos para evitar a instabilidade da garra.

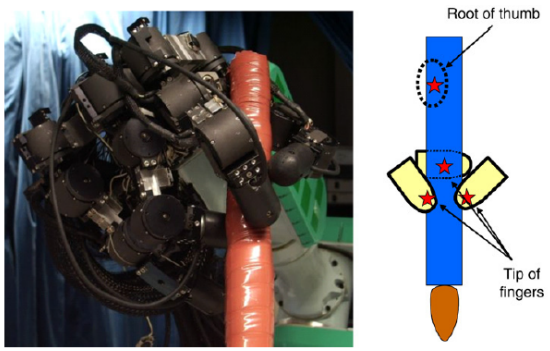


Figura 2.18: *Garra e pincel [29]*

De forma a criar diferentes tipos de pinceladas, e consequentemente pinturas, consideram-se três técnicas (Figura 2.19):

- estudar o ângulo do pincel;
- pressionar o pincel contra a tela;
- levantar gradualmente o pincel da tela.

O primeiro parâmetro analisa o ângulo de inclinação do pincel, pois influencia no resultado, principalmente se se tratar de uma curva. Seguidamente, os parâmetros seguintes verificam a pressão exercida - estes intervêm no tipo de linha do desenho.

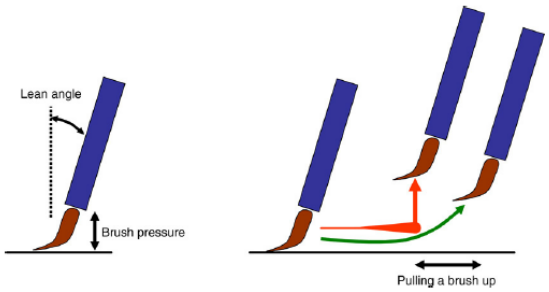


Figura 2.19: *Variação dos parâmetros do pincel [29]*

A implementação do sistema, como referido anteriormente, é dividida em fases: inicialmente múltiplos pontos da imagem são selecionados, e ao mesmo tempo é verificada a cor. Posteriormente, as coordenadas da figura modelo são calibradas, em que as curvas são divididas num conjunto de pequenas linhas.

Na Figura 2.20 é apresentado o resultado de várias pinturas relativamente à mesma imagem modelo. No canto superior esquerdo encontra-se a imagem em 3D, posteriormente à extração da silhueta, a figura modelo é gerada, desenho superior central. As restantes pinturas são os resultados da pintura realizada pelo robô. Como é notável, das quatro pinturas, estas têm diferenças visíveis.

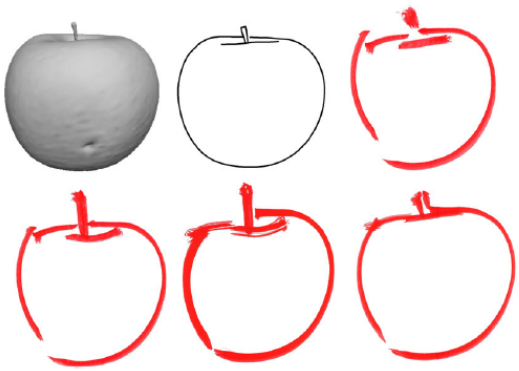


Figura 2.20: *Influência na pintura de diferentes parâmetros do pincel [29]*

Em termo de conclusão, é possível analisar neste caso que o pincel é uma ferramenta sensível, tendo variações na pintura consoante a direção e ângulo, uma vez que este não se move na mesma direção durante a pintura.

2.5 Comunicação da Informação CAD

Nos nossos dias, a indústria vem reinventando os sistemas de produção, muito pela competição do mercado global. Os sistemas tradicionais de automação física, cada

vez mais, são substituídos por sistemas mais flexíveis e inteligentes, permitindo às empresas produzir mais e melhor e garantir a qualidade e o preço competitivo. Contudo, devido à flexibilidade, os robôs industriais necessitam de rápida e eficiente atualização da programação, trazendo ideias de tornar a programação do robô mais simples e intuitiva. Com isto a programação *offline* tornou-se indispensável, não parando a produção do robô e também garantindo a segurança do operador. A tecnologia CAD toma um papel importante nesta matéria, pela maneira fácil e económica de trabalhar.

Com o triunfo dos robôs e os seus sistemas na indústria, nos nossos dias, o propósito passa pela realização de tarefas de forma mais rápida, flexível e intuitiva. Para isso a robótica ainda levanta algumas questões e desafios, como é representado no diagrama de causa e efeito da Figura 2.21.

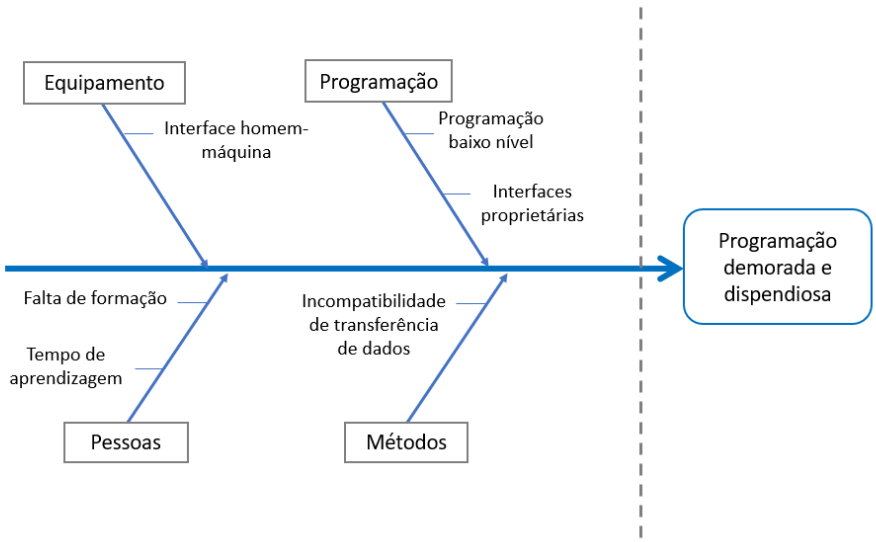


Figura 2.21: *Diagrama de Causa e Efeito*

A programação hoje em dia ainda tem algumas limitações, que tornam o processo demorado e dispendioso e que requer especialização técnica. O objetivo é desenvolver metodologias que ajudem os utilizadores a programar o robô com alto nível de abstração da linguagem específica do robô. Para isso, existe a necessidade de criar interfaces homem-máquina mais intuitivas. Isto pressupõe desenvolvimentos na facilidade de programação e utilização e em ferramentas e linguagens de programação *standard* [4].

A indústria automóvel esteve na origem do surgimento da tecnologia CAD, desenvolvida para ajudar na modelização e otimização do processo de criação. Esta foi pensada por Ivan Sutherland, em 1963, que formou a base teórica para o sistema computacional, nomeadamente, a interatividade gráfica.

2.5.1 Componentes de um sistema CAD

Segundo Mikell Groover [30], um sistema CAD é "Um sistema computorizado para ajuda na criação, modificação, análise e optimização da actividade de concepção (projecto)".

Como apresentado na Figura 2.22, um sistema CAD é constituído essencialmente por [30]:

- *hardware*: Computador, terminal gráfico, periféricos de entrada e saída de dados;
- *software* aplicativo: aplicação que executa as funcionalidades CAD (Desenho 2D e 3D, análise e documentação);
- sistema operativo: interface entre o *hardware* e *software* aplicativo.

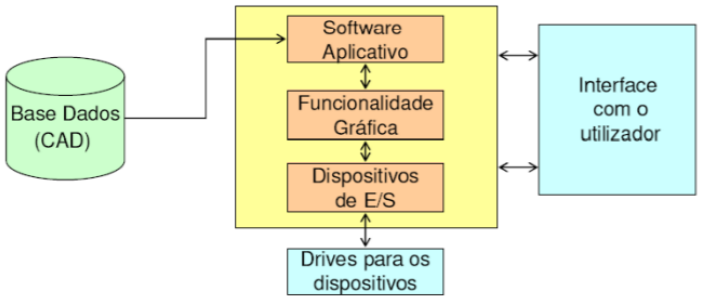


Figura 2.22: *Componentes de um sistema CAD [30]*

Os sistemas CAD estão associados a diversas vantagens, trazendo um aumento da produtividade, sendo um auxílio para o projetista seja a visualizar o produto ou a fazer as correções necessárias, sínteses e documentação, desta forma, reduzindo o tempo de modelação. Sendo uma ferramenta que permite fazer análise completa do esboço, melhora a qualidade do desenho e a sua documentação. Outro aspeto a realçar é a possibilidade de criação de base de dados de geometrias específicas, reduzindo, mais uma vez, o tempo de modelação e consequentemente do processo inerente.

2.5.2 Formatos

Com a existência de uma grande variedade de ferramentas CAD, surge a necessidade de importar/exportar ficheiros entre os diferentes *softwares*. Um ficheiro CAD pode divide-se em duas categorias distintas, (Figura 2.23) formato nativo ou ficheiro formato neutro. O formato nativo permite aceder ao ficheiro e mesmo fazer alterações, como formatos, .DWG, .JPG, .PNG, entre outros.

Os formatos neutros (*Standard*) ajudam na troca de ficheiros entre diferentes *softwares*. Para transferir um modelo de um sistema CAD para outro distinto é necessário converter o ficheiro do primeiro sistema para um formato neutro, sendo

depois, através de um pós-processador do segundo sistema CAD, tratado de forma a poder ser interpretado.

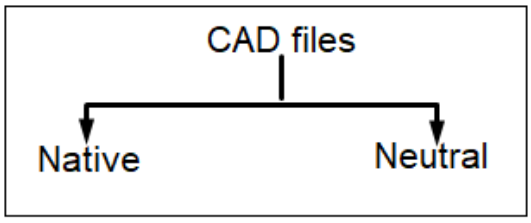


Figura 2.23: *Formato ficheiro CAD*

A troca de dados deve incluir a partilha de toda a informação necessária sobre o objeto. Existem quatro tipos de informação na base de dados do produto [30]:

- forma: informação topológica e geométrica, isto é, características da peça;
- auxiliar: informação gráfica (cores), informação de parâmetros da base de dados;
- *design*: informação gerada para fins de análise, como massa;
- fabrico: informação sobre ferramentas, maquinação, tolerâncias a ter, entre outros.

Embora haja melhores formatos, o *Standard for External representation of Produt data* (STEP) e o *Initial Graphics Exchange Specifications* (IGES) são os mais dominantes. O formato CAD recorrente, como DXF ainda não é padronizado.

O IGES foi desenvolvido em 1980, como resposta à necessidade da indústria para um formato neutro, que permitisse a troca entre sistemas CAD e CAM. É utilizado mundialmente para transferência de dados gráficos entre sistemas CAD idênticos ou não. Consiste num procedimento para o armazenamento e transferência de primitivas geométricas, tais como pontos, linhas e superfícies.

Os ficheiro IGES são constituídos por seis secções: *Flag*, *Start* (S), *Global* (G), *Directory Entry* (DE), *Parameter Data* (PD) e *Terminate* (T). As seções são identificadas através de uma letra colocada na coluna '73' de cada linha do ficheiro. Por exemplo, se for a letra 'G', trata-se da secção *Global*. É na secção *Directory Entry* que é feita a descrição de todas as entidades que constituem o desenho. A descrição de cada entidade utiliza duas linhas, sendo a segunda uma continuação da primeira e inclui informação sobre tipo de linha, cor, etc. Nesta secção existe um ponteiro de matrizes, ponteiro para a secção *Parameter Data*, onde são especificados os parâmetros/coordenadas de cada elemento, como é apresentado na Figura 2.24.

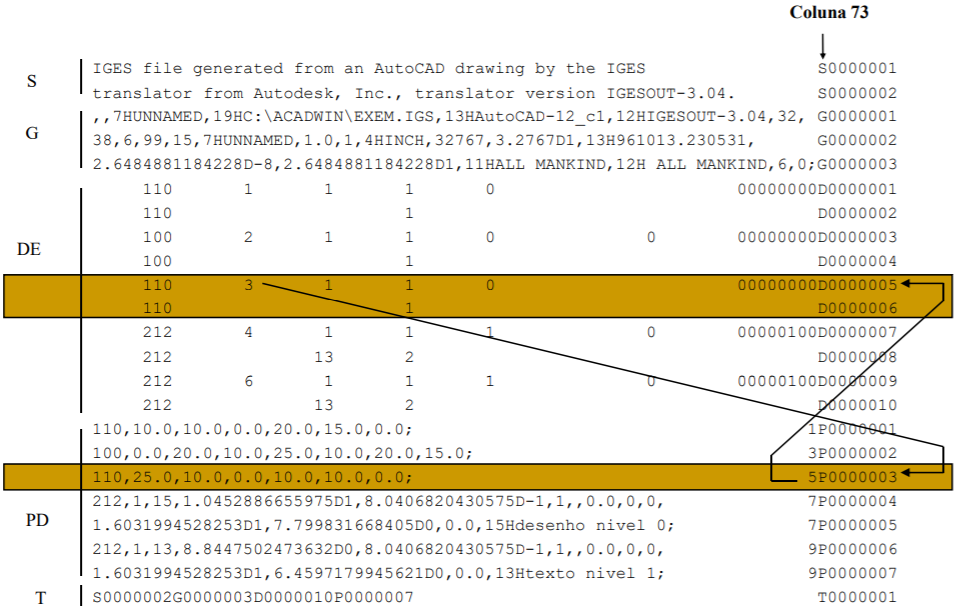


Figura 2.24: *Formato exemplo do ficheiro IGES [30]*

O STEP é uma norma internacional (ISO) para a representação e troca de informação de um produto. Na sua estrutura inclui informação não geométrica, como especificações de tolerâncias, especificações de materiais, entre outros.

Inclui uma linguagem específica de dados (EXPRESS) possuindo uma abordagem orientada a objetos, trata-se de uma linguagem de modelação de dados com gramática mais flexível [30].

A escolha o melhor formato CAD advém muitas vezes do tipo de informação a transferir, dos tipos de *softwares* envolvidos e das referidas compatibilidades, porém há características gerais que referentes ao tipo de formato que são cruciais na escolha.

Apesar do formato IGES ser o segundo formato neutro mais popular tem incoerências quanto ás superfícies de sólidos, deixando algumas superfícies com dados em falta. Além disso, tem uma leitura não linear tornando mais difícil a leitura de dados. O formato STEP apesar de se diferenciar relativamente ao IGES, raramente supera as estratégias de conversão do formato *kernel* ou do nativo. Uma vez que estes têm mais dados relativamente à geometria, tanto informação de produto como atributos definidos pelo utilizador. Como apresentado na Figura 2.25 o formato nativo é a melhor formato possível para utilizar quando se pretende uma perda mínima de informação [31].

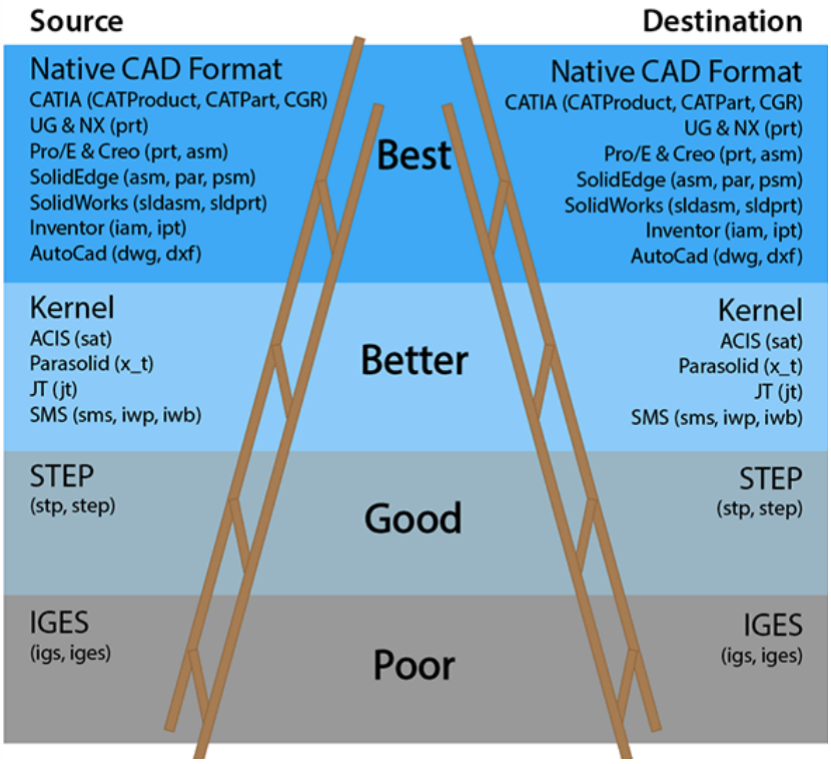


Figura 2.25: *Facilidade de transferência de formatos CAD [31]*

3
Arquitetura do Sistema

Neste capítulo é abordada a arquitetura do sistema. É especificado o cenário do problema e os requisitos propostos. Posteriormente é apresentada a arquitetura de uma possível solução.

3.1 Especificação do problema

Embora as aplicações de pintura a pistola na indústria sejam mais difundidas, existem aplicações de pintura manual a pincel, em áreas de cerâmica e decoração, em que os produtos são mais valorizados e, consequentemente, mais dispendiosos. Associar a automação a esta área pode ser um mercado a explorar e possivelmente com futuro associado.

O objetivo deste trabalho, como pode ser visualizado na Figura 3.1, consiste no desenvolvimento de um robô capaz de efetuar pintura artística a pincel, de desenhos importados a partir de *software* CAD. Para tal, é necessário o desenvolvimento de um programa de conversão do desenho para o formato de "leitura"do robô.

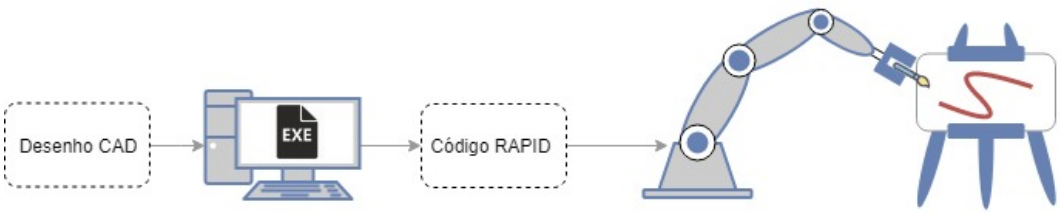


Figura 3.1: *Objetivo geral*

Para isso é essencial:

- *software* CAD;
- programa de conversão;

- robô;
- ferramenta de trabalho;
- objeto de pintura.

3.1.1 *Software* CAD

O *software* adotado para o presente trabalho é o AutoCAD 2017. Esta escolha prende-se com o facto de se tratar de uma ferramenta de trabalho em 2D e 3D e pelo formato de intercâmbio escolhido, DXF, ter sido desenvolvido pela Autodesk, e, consequentemente, poder encontrar-se mais documentação acerca do formato e das suas particularidades.

3.1.2 Programa de conversão

De forma a extrair informação dos desenhos CAD é necessário desenvolver um programa que faça esse processo. Com esta finalidade foi adotado o *Integrated Development Environment* (IDE) NetBeans, por ser do conhecimento do autor. O NetBeans permite desenvolver *software* em diversas linguagens, como Java, PHP, C/C++, entre outras. Por ser uma linguagem com portabilidade, de nível intermédio e muito popular, escolheu-se a linguagem C para a realização do programa.

3.1.3 Robô

O robô utilizado é o modelo ABB IRB120 e o controlador ABB IRC5 *Compact* (Figura 3.2) disponíveis no laboratório F104 do Departamento de Engenharia Mecânica. O IRB120 pode ser montado em qualquer ângulo, tendo um alcance de 580 mm e uma capacidade de carga de 3 kg. Todos os dados técnicos deste modelo podem ser visualizados no *datasheet* presente no Anexo A.

Figura 3.2: *Robô ABB IRB120 e controlador IRC5 Compact [32]*

3.2 Proposta para o Sistema

A arquitetura do sistema, representada na Figura 3.3, consiste, primeiramente, na elaboração de um desenho 2D em CAD, sendo este guardado no formato DXF. O ficheiro DXF será convertido para a linguagem específica do robô, RAPID, através do programa desenvolvido. Inicialmente será feita a passagem do código RAPID para o ambiente de simulação RobotStudio, onde serão feitos ajustes e previsões - por fim, feito o teste em ambiente real.

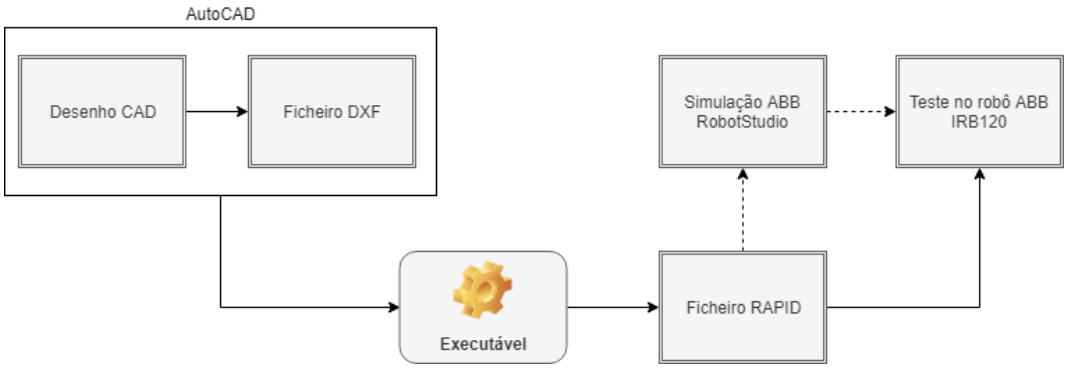


Figura 3.3: *Arquitetura*

3.2.1 Formato DXF

O formato DXF é um tipo de ficheiro comum na troca de desenhos entre diferentes *softwares*. São ficheiros de texto ASCII divididos em quatro secções (*SECTION*): *HEADER*, *TABLES*, *BLOCKS* e *ENTITIES*. Na Tabela 3.1 são apresentados os parâmetros de cada segmento.

Aquando da extração da informação gráfica necessária, muitas partes do ficheiro podem ser ignoradas. Cada elemento gráfico é gravado num formato fixo, permitindo uma extração mais simples da informação. Os parâmetros de geometria necessários encontram-se na secção ENTITIES, portanto basta fazer a leitura desta secção.

Como apresentado na Figura 3.4, após identificar a palavra "ENTITIES"é necessário identificar o tipo de elemento gráfico (LINE, POINT, ARC) e os correspondentes valores de coordenadas. As coordenadas do tipo LINE encontram-se dentro do bloco "AcDbLine", onde é apresentado o código e o respetivo valor, ou seja, o valor "10"representa o código para o valor de X inicial, sendo o valor apresentado na linha seguinte. Assim, o valor de Y inicial encontra-se abaixo do "20", o valor de Z inicial abaixo do "30"e, os valores de X, Y e Z finais, abaixo de "11", "21"e "31", respetivamente.

Tabela 3.1: *Secções do ficheiro DXF [33]*

HEADER	Contém parâmetros do desenho, que podem ser visualizados com o comando STATUS no AutoCAD. Apresentando parâmetros como tamanho do ficheiro, estilo da camada do desenho, etc.
TABLES	Especificadas todos os tipos de tabelas que compõem o desenho AutoCAD, tais como, tabela tipos de linha (*linetypes*), tabela de níveis (*layer*), de estilo (*style*) e de vistas (*(view*).
BLOCKS	Contém a definição das entidades *block* descrevendo as entidades que as compõem
ENTITIES	Contém a geometria 2D ou 3D, especificado em pontos, linhas, círculos e, fazendo a conexão à informação dos blocos. É a secção principal da identificação do formato DXF.

```
ENTITIES          Secção
 0
LINE              Tipo de elemento gráfico
...
AcDbLine
 10
35.26858487543496      Valor de X inicial
 20
169.637801389579       Valor de Y inicial
 30
0.0                    Valor de Z inicial
 11
59.97938134410765      Valor de X final
 21
169.637801389579       Valor de Y final
 31
0.0                    Valor de Z final
 0
```

Figura 3.4: *Exemplo da representação de uma linha em formato DXF*

Se o elemento gráfico for tipo ARC, os parâmetros apresentados dentro da secção

ENTITIES são diferentes. Como apresentado na Figura 3.5, dentro do bloco "AcDb-Circle"é apresentado o valor das coordenadas do ponto central do arco, em que os códigos "10", "20"e "30"representam as coordenadas X, Y e Z do centro do arco, respetivamente. Também é apresentado o valor do raio, no código "40", e os valores do ângulos inicial e final, nos códigos "50"e "51", respetivamente.

```
ENTITIES
 0
ARC        Tipo de elemento gráfico
...
AcDbCircle
 10
147.6887736793939      Coordenada X do centro do arco
 20
88.55705918098536      Coordenada Y do centro do arco
 30
0.0                    Coordenada Z do centro do arco
 40
27.69130492820283      Raio
100
AcDbArc
 50
275.0580492524776      Ângulo inicial
 51
354.9419507875411      Ângulo final
 0
```

Figura 3.5: *Exemplo da representação de um arco em formato DXF*

No caso do elemento gráfico ser tipo POINT são apresentados apenas os valores das coordenadas X, Y e Z, nos códigos "10", "20"e "30", respetivamente, dentro do bloco "AcDbPoint, apresentado um exemplo na Figura 3.6.

```
ENTITIES
 0
POINT          Tipo de elemento gráfico
...
AcDbPoint
 10
187.2748929910803      Valor de X
 20
72.0439870715586       Valor de Y
 30
0.0                    Valor de Z
 0
```

Figura 3.6: *Exemplo da representação de um ponto em formato DXF*

O ficheiro DXF também armazena as especificações das entidades, nomeadamente, cor, espessura de linha, entre outros.

No código DXF as cores são tabelas conforme o sistema de combinação de cores *Red Blue and Green* (RGB). Sendo a cor combinada na escala de 0 a 255, é atribuído um número a ser utilizado no *software* CAD. Na Tabela 3.2 são apresentados alguns dos códigos cor mais utilizados.

Tabela 3.2: *Código de cores DXF*

Código DXF/AutoCAD	Cor
1	Vermelho
2	Amarelo
3	Verde
4	Ciano
5	Azul
6	Magenta
250	Preto

O código '62' é o código associado ao número da cor, sendo apresentado o "valor"da cor na linha seguinte. Estes parâmetros são apresentados na subclasse 'AcDbEntity' dentro da secção 'Entities'.

Na Figura 3.7 é possível visualizar um exemplo. Imediatamente abaixo do código que representa a cor, '62', encontra-se o valor '2', que corresponde à cor amarelo.

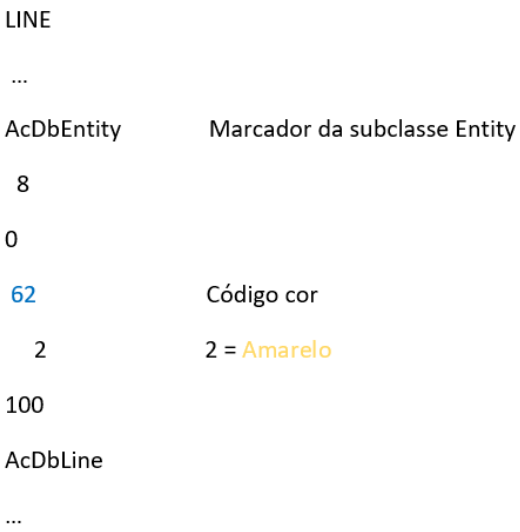


Figura 3.7: *Exemplo da representação do parâmetro cor em formato DXF*

3.2.2 Linguagem programação RAPID

A linguagem RAPID é uma linguagem de programação de alto nível para programação de robôs ABB. O RAPID foi introduzido em 1992 substituindo a ASEA programming Robot Language (ARLA). Um programa em RAPID é composto por um conjunto de instruções estruturadas seguindo uma lógica hierárquica que controlam o robô, existindo instruções específicas para os movimentos, havendo então três tipos de rotinas [34]:

- *procedures*: são utilizados como subprogramas;
- *functions*: retornam valores de um tipo específico que é utilizado como argumento de instrução;
- *trap routines*: providenciam meios de resposta a interrupções no programa, podendo ser associadas a uma interrupção específica.

A informação pode ser armazenada em dados, sendo entes agrupados em diferentes tipos, cada um descrevendo um tipo de informação distinta. No RAPID não há limite na criação de dados e estes, podem existir globalmente ou localmente dentro de uma rotina. Existem três tipos de dados [34]:

- *constants*: representa um valor estático e apenas podem ser alterados modificando o código-fonte;
- *variables*: podem ser alterados durante a execução do programa;
- *persistents*: são referenciadas como variáveis "persistentes", ou seja, o seu valor assume o último valor adquirido, mesmo caso haja uma reinicialização do programa.

De forma a nomear instruções em RAPID, são utilizados identificadores, utilizados para denominar módulos, rotinas e dados, como é possível visualizar na figura 3.8.

```
MODULE module_name
PROC routine_name()
VAR pos data_name;
label_name:
```

Figura 3.8: *Exemplo de identificadores no RAPID [34]*

Um módulo de programa pode conter diferentes dados e rotinas. Um dos módulos contém o procedimento de entrada, um procedimento global, denominado por *main*. Executar o programa significa a execução do procedimento (*procedure*) principal, sendo que este pode referenciar rotinas e/ou dados contidos e outros módulos, visível na figura 3.9.

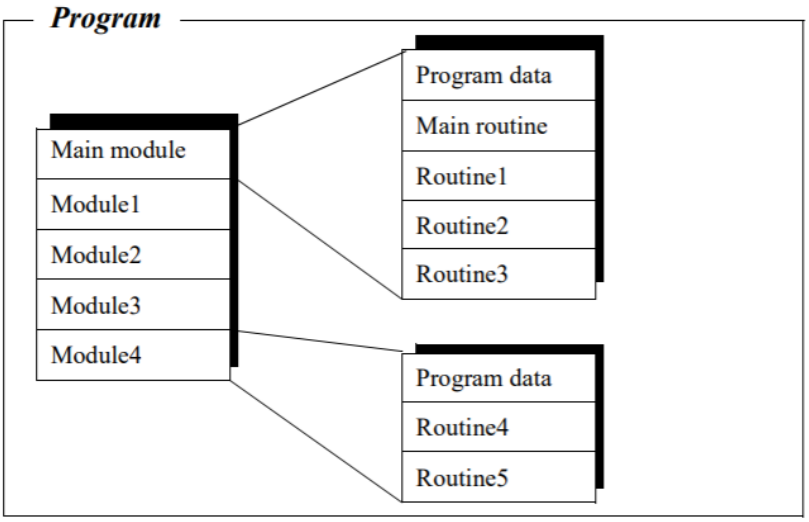


Figura 3.9: *Divisão de um programa RAPID [34]*

Durante a execução do programa, as instruções de posicionamento controlam os movimentos do robô. A principal tarefa da instrução é fornecer [34]:

- ponto de destino do movimento (definido com a posição, orientação, configuração e posição dos eixos externos);
- método de interpolação utilizado (interpolação das juntas, interpolação linear e interpolação circular);
- a velocidade do robô;
- os dados de *zone*;
- sistema de coordenadas (*tool, user* ou *object*) utilizada no movimento.

Em RAPID definição dos pontos (*targets*) é constituída por, nome, valores das coordenadas, orientação, valores das configurações do robô e posição dos eixos externos [35], como é visível na Figura 3.10.

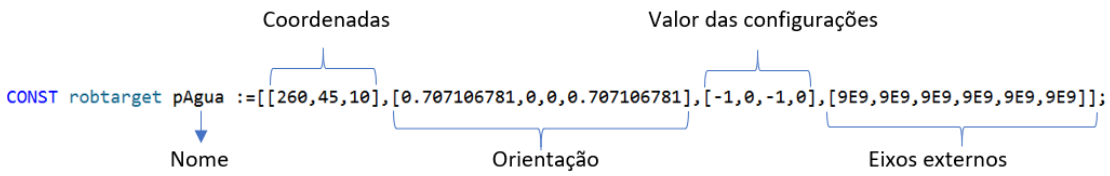


Figura 3.10: *Exemplo da estrutura de um target na linguagem RAPID*

Existem várias instruções de movimento em RAPID, porém as mais comuns são *MoveL*, *MoveJ* e *MoveC*:

- *MoveL*: instrução que move o robô de forma linear, desde a posição atual para a posição seguinte;
- *MoveJ*: intrução que move rapidamente o robô de uma ponto para outro, sendo que este movimento não tem que ser em linha reta:
- *MoveC*: instrução que move o robô ao longo do arco de uma circunferência.

A instrução *MoveL*, é constituída pela sintaxe apresentada na Figura 3.11.

O parâmetro *ToPoint* específica o ponto de destino, por uma constante to tipo *robtarget*. Este é constituído por quatro partes, como referenciado anteriormente, as coordenadas x, y e z, orientação em quaterniões, configurações dos ângulos dos eixos do robô e especifica para até 6 eixos adicionais, quando definido por '9E9' não é utilizado nenhum eixo adicional.

O parâmetro *Speed* define a velocidade do movimento. Já o parâmetro *zone*, especifica a zona de canto aquando o movimento do robô, caso a *zone* seja definida como *fine* o robô vai exatamente para a posição especificada, podendo esta trajetória ser feito um corte nos "cantos"quando o robô passa por um *target*.

Por último, o parâmetro *Tool*, especifica a ferramenta que está a ser utilizada.

```
MoveL ToPoint Speed Zone Tool [\WObj];
```

Figura 3.11: *Sintaxe de um MoveL na linguagem RAPID [35]*

A instrução *MoveJ* tem a mesma sintaxe que a instrução *MoveL*. Na instrução *MoveC* é especificado um ponto do arco da circunferência. Na Figura 3.12 é apresentado um exemplo de movimento com a instrução *MoveC*.

```
MoveL p10, v500, fine, tPen;
MoveC p20, p30, v500, fine, tPen;
MoveL p40, v500, fine, tPen;
```

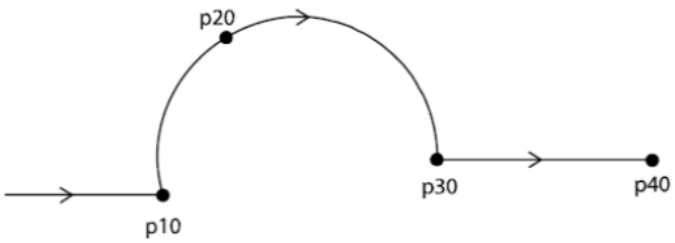


Figura 3.12: *Sintaxe de um MoveL na linguagem RAPID [35]*

Um módulo de programa é guardado com a extensão ".Mod", este é aberto no RAPID como um módulo. A utilização de diversos módulos no programa é apenas para tornar mais fácil a leitura e reutilização deste. Porém apenas um módulo pode conter o procedimento *main*.

4
Implementação Prática

Neste capítulo são descritos os métodos utilizados na implementação do trabalho. Sendo feita uma abordagem sequencial, pelas tarefas principais do trabalho, nomeadamente, executável referente à programação de conversão, simulação no *software* ABB RobotStudio e o espaço de trabalho real.

4.1 Programa de conversão de código DXF para linguagem RAPID

Para a conversão do código DXF para linguagem RAPID procedeu-se à elaboração de um programa. Este é incumbido de:

- ler ficheiro DXF;
- identificar tipo de desenho (linha, arco ou ponto);
- guardar respetivas coordenadas e as cores;
- criar ficheiro RAPID.

Para a extração e atribuição dos dados DXF é necessário seguir uma determinada lógica, de modo a guardar os valores necessários numa matriz. Lógica essa que pode ser vista no fluxograma da Figura **??**. O código do programa de conversão pode ser visualizado no Anexo B.

O programa inicia-se pela leitura linha a linha do ficheiro DXF, indicado pelo utilizador. Inicialmente corre o ficheiro para encontrar a palavra "ENTITIES", no momento que esta for encontrada o programa continua até encontrar um tipo de desenho (LINE, ARC ou POINT).

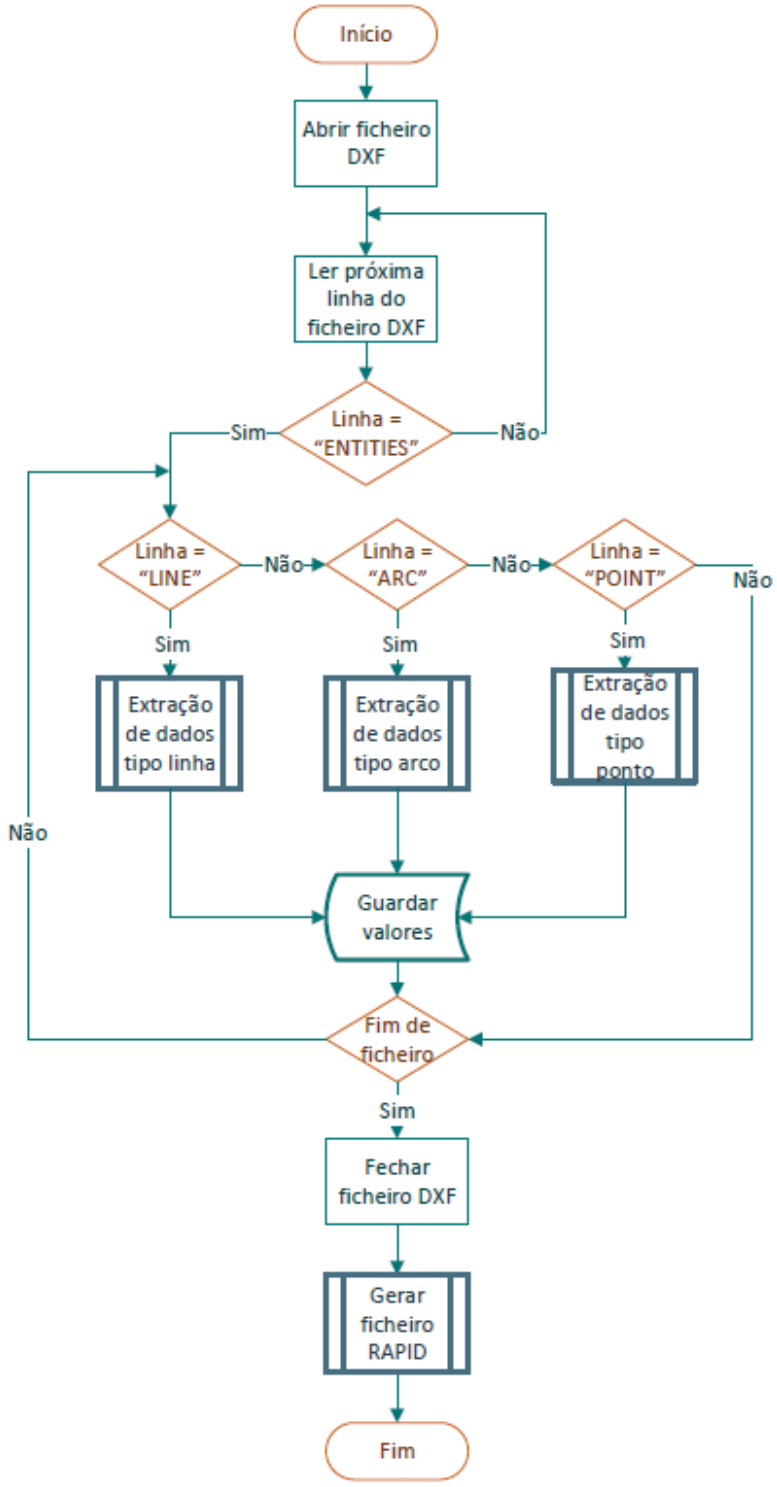


Figura 4.1: *Fluxograma do programa de conversão do código DXF para linguagem RAPID*

No caso do tipo de desenho ser linha (LINE), procede-se à leitura da cor correspondente e compara com o leque de cores de referência. De seguida efetua a leitura do ponto inicial e final da linha, verificando se este se encontra dentro da área de trabalho definida. Caso reúna as condições anteriores, acondiciona os valores numa matriz. O fluxograma de extração de dados encontra-se no Anexo C.

No caso do arco (ARC), procede-se novamente à leitura e verificação da cor, sendo posteriormente realizada a leitura dos dados do arco, que serão utilizados para determinar os três pontos necessário à criação de um arco em RAPID.

Uma vez que o ficheiro RAPID necessita de três pontos para a criação de um arco e, o ficheiro DXF não nos fornece esses pontos diretamente, é necessário efetuar a transformação dos valores dados para os pontos necessários (Figura 4.2). O fluxograma de extração de dados encontra-se no Anexo C.

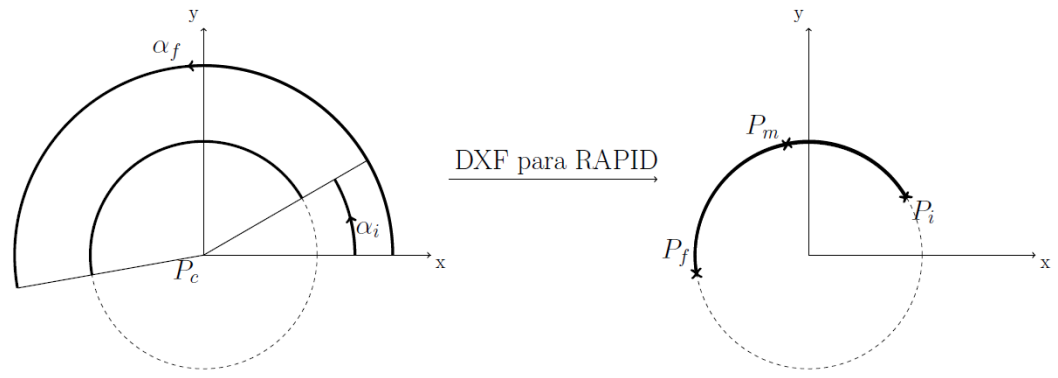


Figura 4.2: *Transformação do arco do formato DXF para RAPID*

Para criar o ponto inicial (P_i), tem-se em conta o ponto central, o raio e o ângulo inicial. Assim como o ponto final, que se calcula utilizando o ponto central, raio e o ângulo final:

$$P_i(\text{x,y}) = \left\{ \begin{array}{l} x = \text{P}_c x + \text{raio} * \cos(\alpha_i) \\ y = \text{P}_c y + \text{raio} * \sin(\alpha_i) \end{array} \right\}$$

$$P_f(\text{x,y}) = \left\{ \begin{array}{l} x = \text{P}_c x + \text{raio} * \cos(\alpha_f) \\ y = \text{P}_c y + \text{raio} * \sin(\alpha_f) \end{array} \right\}$$

Como no código RAPID é necessário um ponto auxiliar para realizar o movimento do arco, foi determinado um novo ângulo para calcular esse ponto:

$$\alpha_m = (\alpha_i + \alpha_f)/2$$

$$P_m(\text{x,y}) = \left\{ \begin{array}{l} x = \text{P}_c x + \text{raio} * \cos(\alpha_m) \\ y = \text{P}_c y + \text{raio} * \sin(\alpha_m) \end{array} \right\}$$

No caso do tipo de elemento gráfico ponto (POINT), é mais simples, uma vez que, apenas tem três valores a serem lidos, correspondentes a um ponto (X, Y e Z). O fluxograma de extração de dados encontra-se no Anexo C.

Concluindo a leitura do ficheiro DXF é feita uma reorganização da matriz relativamente ao parâmetro cor, no sentido de organizar a ordem de pintura do desenho, ou seja, para reduzir o número de vezes que o robô vai fazer a limpeza do pincel.

4.2 Simulação do funcionamento do sistema RobotStudio

Para uma programação mais assertiva realizou-se a programação *offline* do sistema no *software* de simulação RobotStudio da ABB.

Para a modelação da célula do robô, apresentada na Figura 4.3, é necessário determinar as dimensões da célula real, de forma a que a modelação seja o mais precisa possível. Sendo o suporte do robô um elemento significativo, uma vez que faz de base para o robô assim como, é o ambiente de trabalho deste.

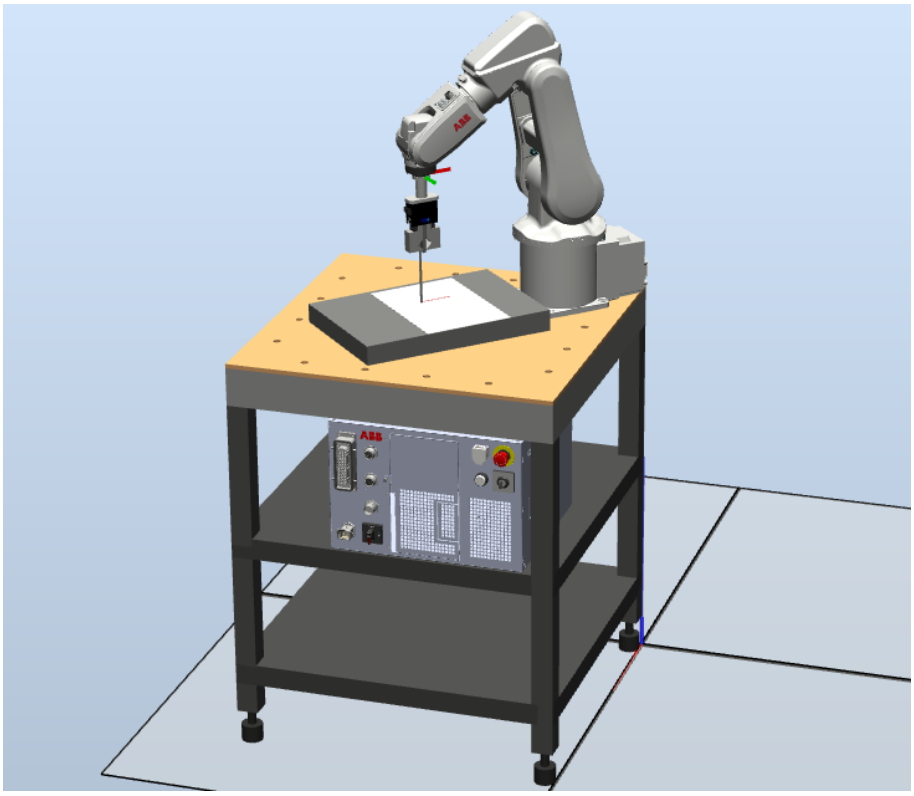


Figura 4.3: *Célula modelada em RobotStudio*

4.2.1 Definição da ferramenta de trabalho

A ferramenta é um objeto que pode ser montado na flange do robô, devendo ser definida com um *Tool Center Point* (TCP), ponto central da ferramenta. Como o robô e os seus movimentos estão sempre relacionados com o TCP da ferramenta, é importante que a definição deste seja feita de forma tão precisa quanto possível [35].

O sistema de coordenadas pode ser definido manualmente ou utilizando o robô como ferramenta de medição. Neste caso adota-se a definição de uma ferramenta utilizando o robô para medição, adaptando um pincel à garra do robô (Figura 4.4). Depois de criada a ferramenta é necessário colocar o robô em movimento, levando a ponta da ferramenta a um ponto fixo do espaço de trabalho, no mínimo com quatro orientações distintas, como apresentado na Figura 4.5.

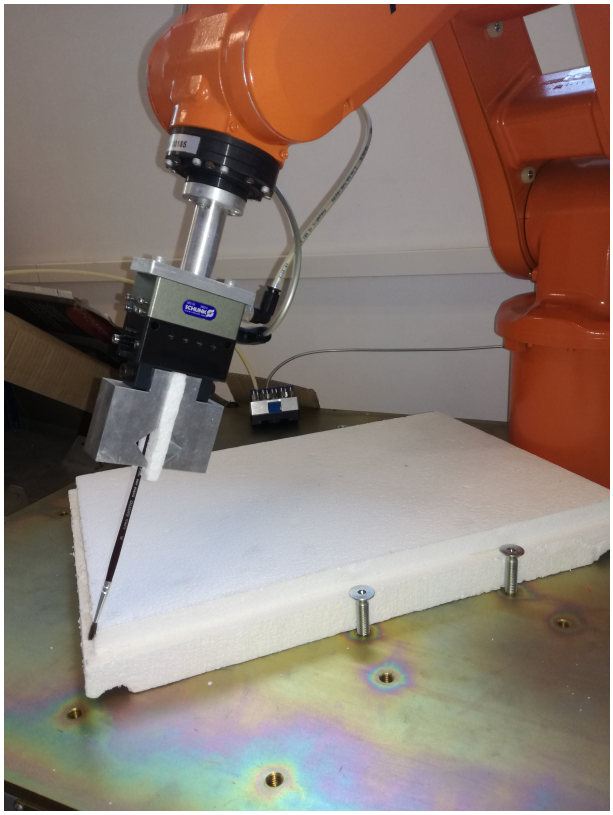


Figura 4.4: *Criação do TCP da ferramenta*

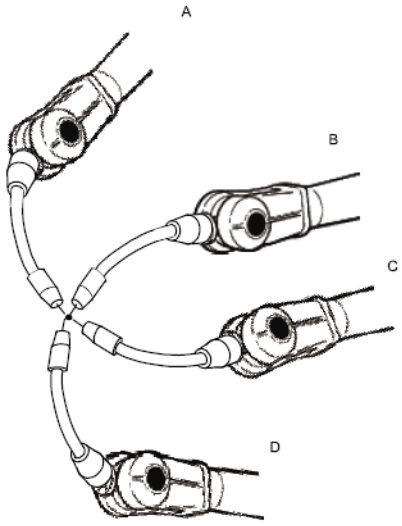


Figura 4.5: *Medição das coordenadas do TCP da ferramenta [35]*

Após programar o TCP da ferramenta são apresentados os valores dos erros máximo, médio e mínimo, no ecrã do *teach pendant*. O erro médio é a distância

média dos pontos de aproximações a partir do TCP calculado. Este valor depende da ferramenta, do tipo de robô e das condições da célula de trabalho.

Na primeira tentativa na criação do TCP da ferramenta, obteve-se um erro médio de 1,977 mm. Usualmente um erro médio de algumas décimas de milímetro é um bom resultado. Portanto, foi necessário voltar a realizar a definição do TCP da ferramenta, sendo este segundo valor de 0,707 mm, um valor considerado aceitável.

4.2.2 Definição dos objetos de trabalho

Um objeto de trabalho (*WorkObject*) é um sistema de coordenadas com propriedades específicas associadas e ele. É utilizado no sentido de simplificar a programação do robô, definindo uma origem para os *targets*.

Tal como o TCP, o objeto de trabalho pode ser definido manualmente ou utilizando o robô como medição. Neste trabalho adotou-se o robô para a programação do objeto de trabalho. Para tal, é necessário levar a ponta do robô a dois pontos situados no eixo dos XX, e um ponto no eixo dos YY, no sistema de coordenadas do objeto (Figura 4.6).

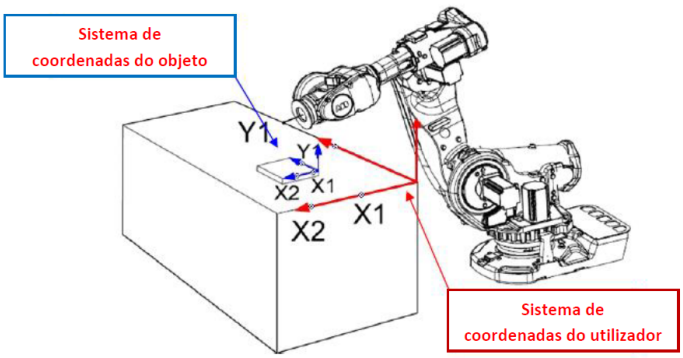


Figura 4.6: *Sistema de coordenadas do utilizador e sistema de coordendas do objeto*

Neste trabalho utiliza-se uma placa de esferovite como base de trabalho para ser mais fácil a adaptação dos componentes, tais como tintas e água. Desta forma, criou-se um *WorkObject* na esferovite e outro na folha de papel, apresentados na Figura 4.7.

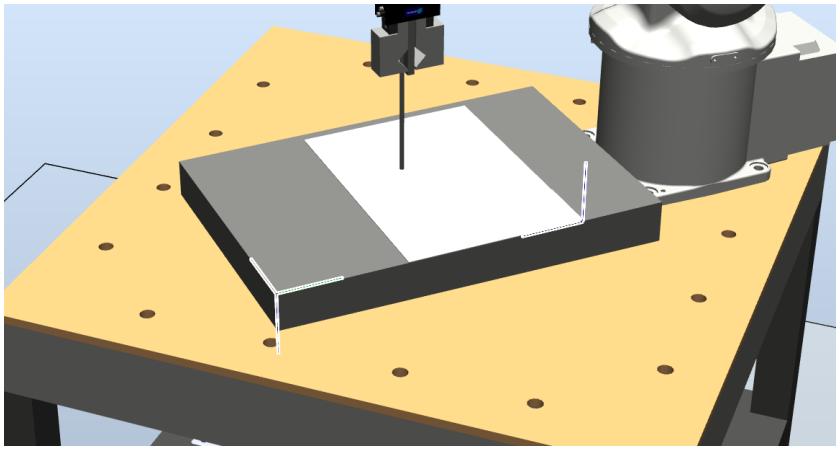


Figura 4.7: *WorkObjects do sistema [35]*

Definiram-se as posições das tintas no objeto de trabalho, assim como a posição do recipiente da água.

Devido às dimensões dos frascos da tinta (Figura 4.8), foi necessário criar três *targets* de forma a criar um caminho (*path*) sem colisões da ferramenta com o frasco. Devido ao volume de trabalho do robô foi necessário reorientar os *targets*.

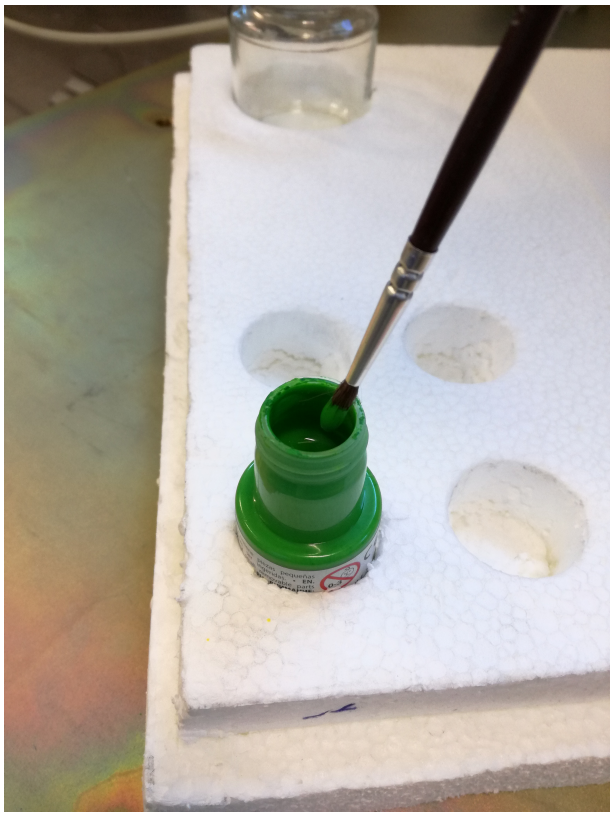


Figura 4.8: *Targets da tinta*

4.3 Modelização geométrica no AutoCAD

Para a realização deste trabalho, é necessário a criação de pequenos desenhos 2D, que posteriormente são gravados no formato DXF.

O AutoCAD permite a configuração do ambiente de trabalho, denominado espaço do modelo. De forma a evitar que os esboços realizados excedessem o espaço real de pintura do robô, nomeadamente uma folha A4, delimitou-se uma área retangular invisível na área de desenho (Figura 4.9). Para isso utilizou-se o comando *DEFLIMITE*, definindo o limite com as dimensões da folha (210x297).

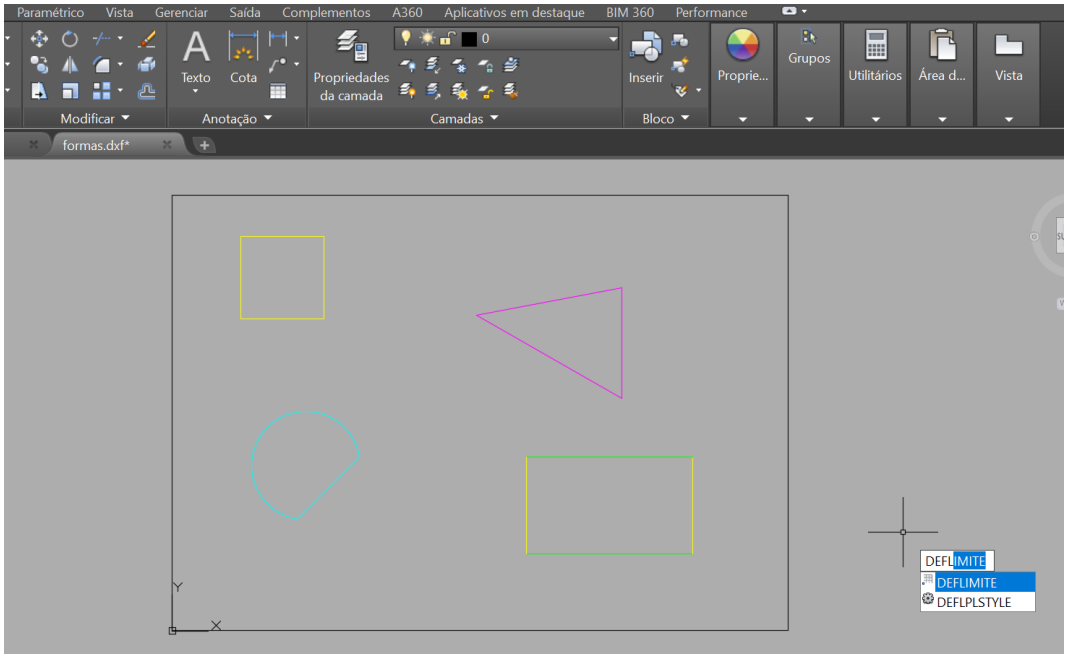


Figura 4.9: *Definição do limite do espaço de trabalho do AutoCAD*

A atribuição das cores a cada forma ou tipo de desenho, é feita tendo em conta os códigos de cor, visto no capítulo 3. No AutoCAD é exibido qual o número indexado a cada cor, como é possível visualizar na Figura 4.10. Esta escolha deve ter em conta quais as cores de tinta escolhidas, neste caso, foram selecionadas cinco cores para execução da pintura (Figura 4.11): verde, amarelo, magenta, ciano e preto. A escolha do tipo de tinta, guache, teve em conta o estudo e análise de aplicações semelhantes, optando então por estes tipo de tinta mais consistente, evitando assim a possibilidade de pingar durante o manuseamento do pincel.

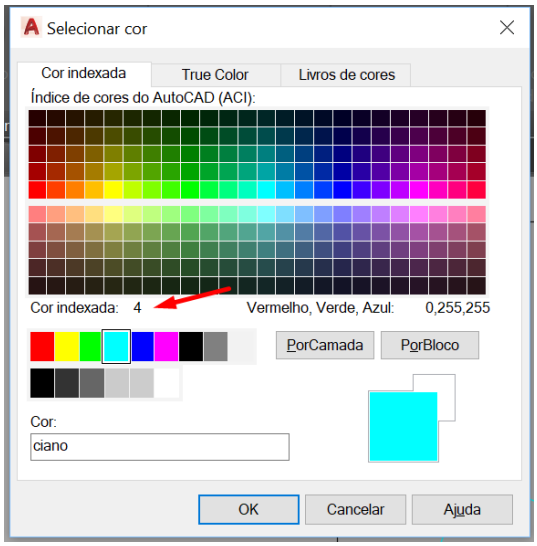


Figura 4.10: *Índice de cores do AutoCAD*

Figura 4.11: *Frascos de tinta*

5
Testes Efetuados e Resultados Obtidos

Neste capítulo serão apresentados os testes efetuados à solução descrita anteriormente e os resultados obtidos.

5.1 Testes Efetuados

Inicialmente realizaram-se testes relativamente às posições dos *targets* das tintas e da água em ambiente real (Figura 5.1). Sendo necessário fazer alguns ajustes.

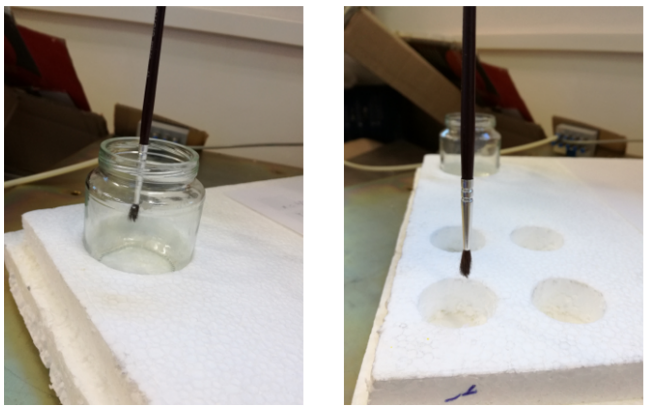


Figura 5.1: *Teste targets da tinta e recipiente da água*

Posteriormente, aquando a realização de teste com água, verificou-se a necessidade de colocação de uma esponja para retirar os restos de tinta do pincel, devido à sua consistência. Na esponja definiram-se quatro *targets* de modo a tentar abranger todos os lados do pincel (Figura 5.2).

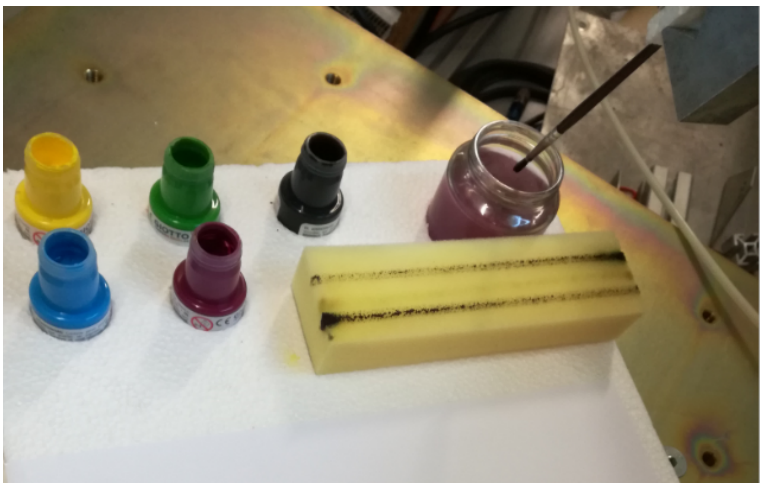


Figura 5.2: *Objetos de pintura*

Outro teste foi a verificação dos vários elementos gráficos utilizados, com diferentes cores, como apresentado na Figura 5.4. Neste processo verificou-se a similaridade relativamente ao desenho CAD - foi apurado que o robô efetuava muitas limpezas do pincel (sempre que muda de tinta), uma vez que segue a ordem de criação de desenho do AutoCAD. Para contornar esta repetição de mudança de tinta e, consequentemente, elevado tempo de pintura, alterou-se o código de forma a reorganizar os dados por cores da matriz (Figura 5.3) antes da criação do ficheiro RAPID, assim reduzindo a quantidade de vezes que efetua a limpeza do pincel.

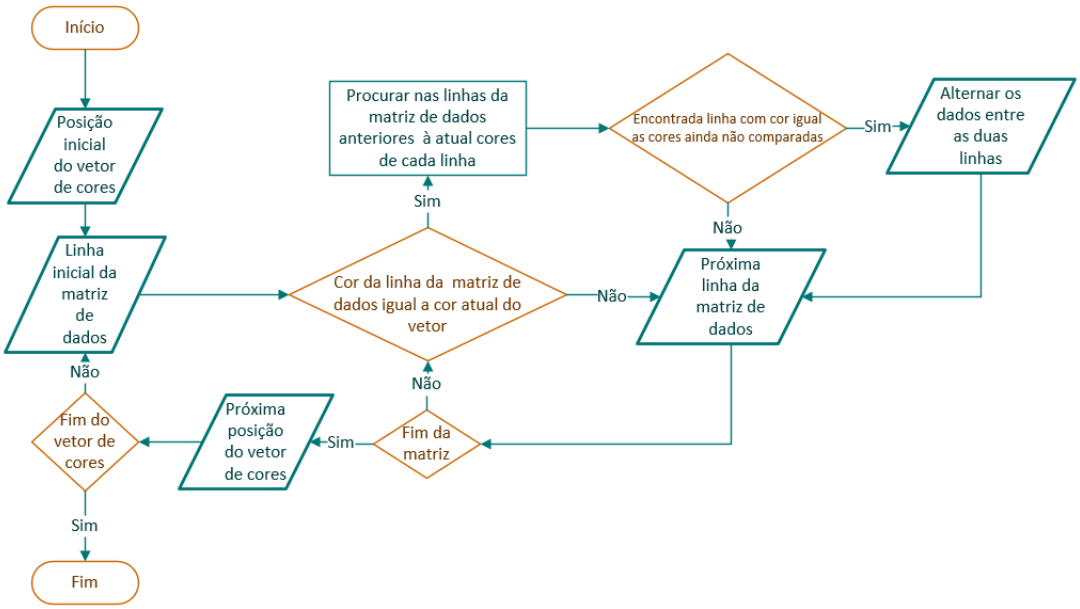


Figura 5.3: *Fluxograma de reorganização do vetor das cores*

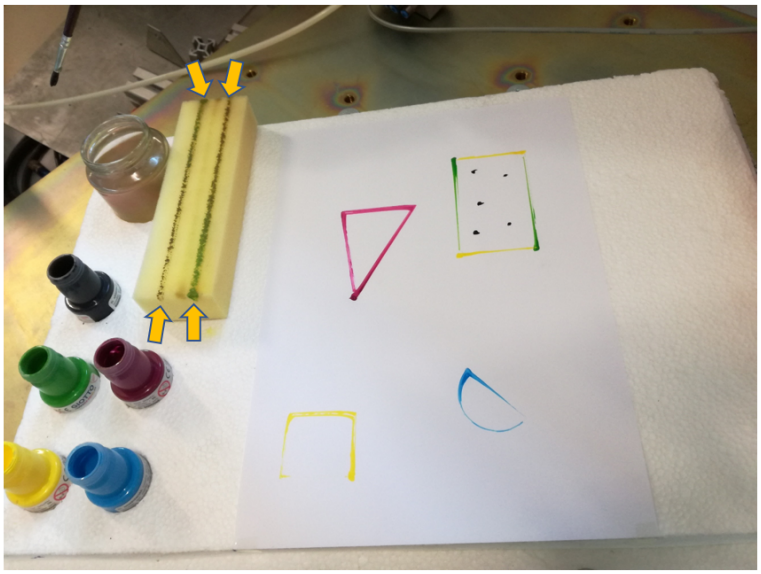


Figura 5.4: *Teste de diferentes tintas*

Depois de realizar testes com desenhos mais extensos, verificou-se o défice de tinta no pincel, tendo por isso, efetuado alterações no código, de modo a fazer recarga de tinta ao fim de 100 mm de desenho (Figura 5.5). Esta contabilização, sempre que uma forma é desenhada (linha, curva) é adicionado o valor do comprimento desse elemento a uma variável, sendo comparada com o valor 100 mm, de forma a que não fique sem tinta durante o desenho de um elemento.

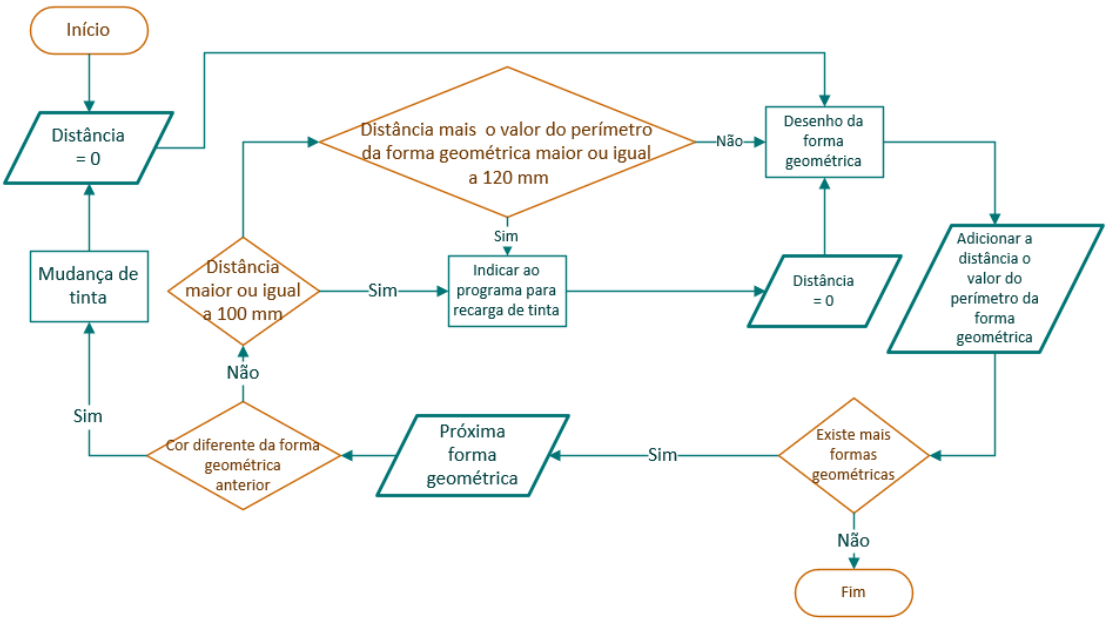


Figura 5.5: *Fluxograma de recarga de tinta*

Na Figura 5.6 é possível visualizar a pintura do mesmo desenho, com e sem recarga da tinta.

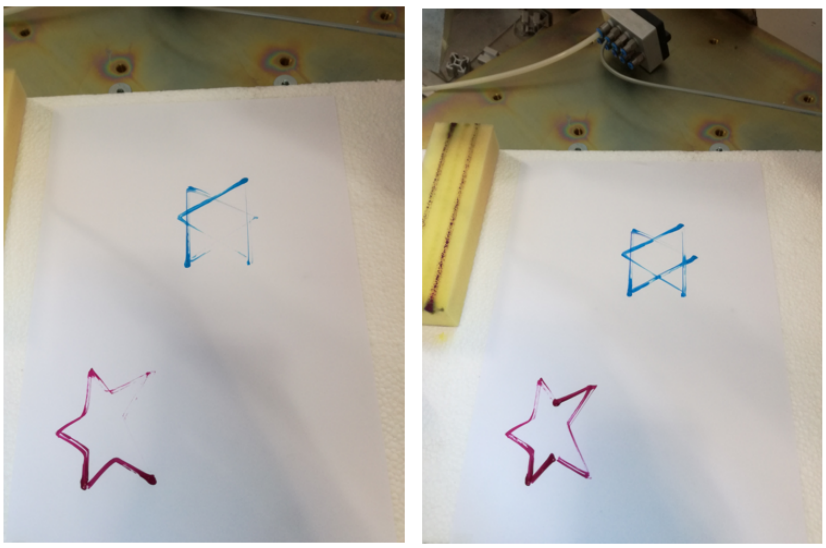


Figura 5.6: *Teste de recarga de tinta. Esquerda: Sem recarga de tinta; Direita: Com recarga de tinta*

5.2 Resultados Obtidos

Durante a realização de testes foram feitas algumas pinturas, tendo em atenção a diversidade de formas e cores. Desta forma na Figura 5.7, 5.8 e 5.9 são apresentados os resultados obtidos. Nestes, é possível verificar a similaridade de forma e cor com os desenhos realizados no *software* CAD, porém, verifica-se também, a irregularidade da pintura. Isto deve-se ao fato do pincel ser um elemento deformável e variável durante o trabalho de pintura.

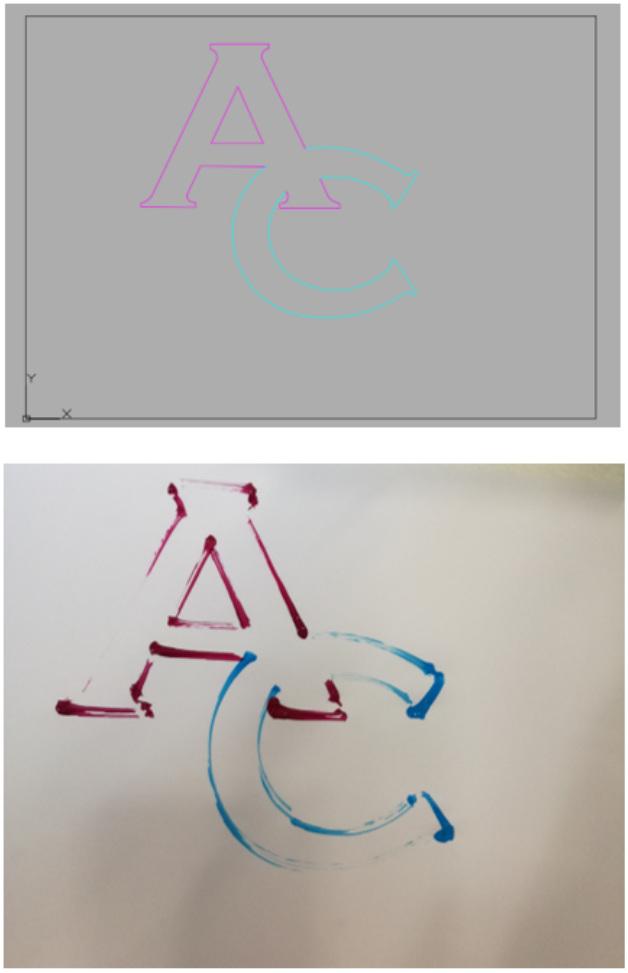


Figura 5.7: *Teste pintura de letras*

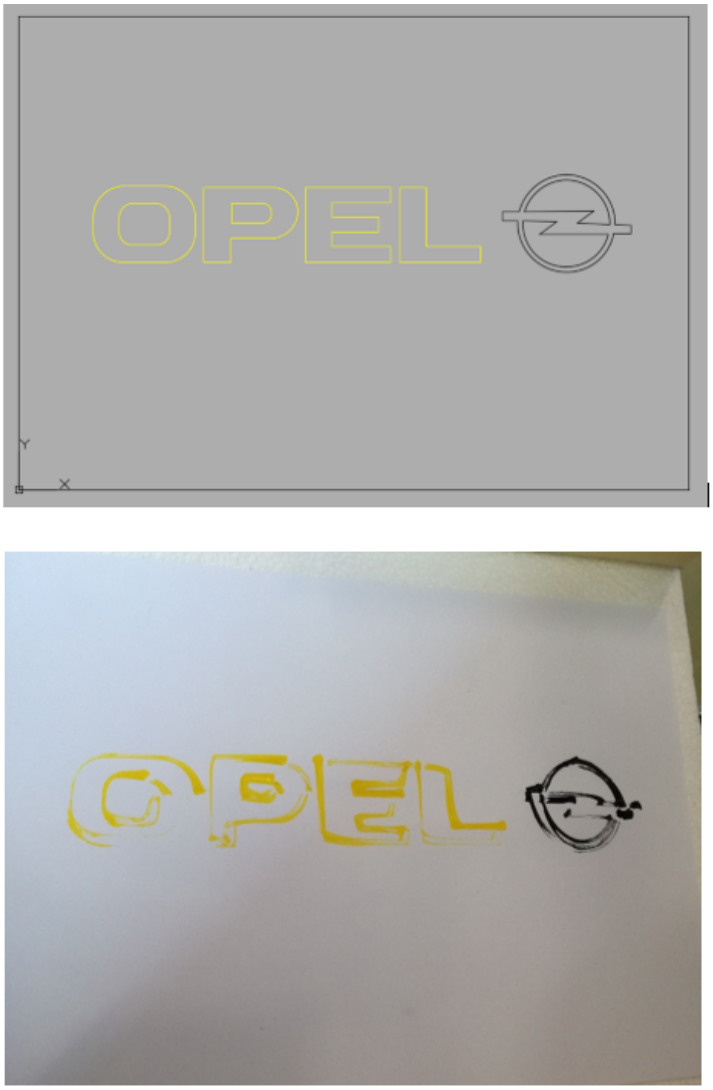


Figura 5.8: *Teste pintura de logótipo*

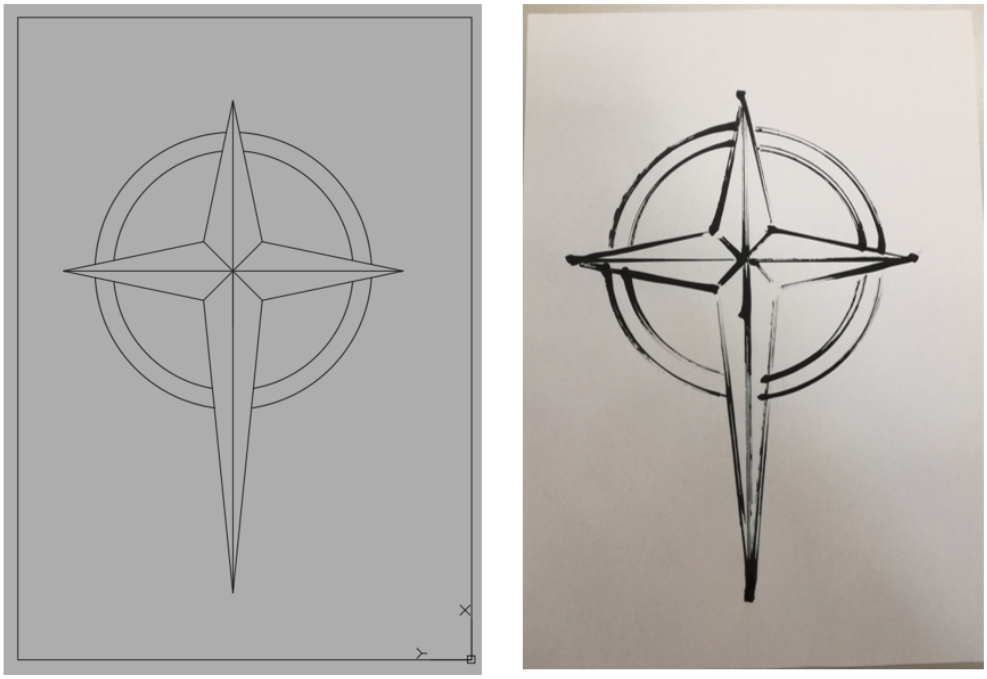


Figura 5.9: *Teste pintura da estrela do norte*

6 Conclusões

Neste capítulo será feito um balanço entre os objetivos propostos e os resultados obtidos. Sendo referidas as principais dificuldades, bem como propostas para trabalhos futuros.

6.1 Conclusões do Trabalho

A presente dissertação descreve o estudo e desenvolvimento de um robô capaz de efetuar pintura artística a pincel, de desenhos exportados a partir de *software* CAD. O processo inicia-se pelo estudo da estrutura do formato a converter, DXF, definindo quais os elementos chave para a extração da informação necessária, nomeadamente os parâmetros geométricos do desenho efetuado em CAD.

Posto isto, é desenvolvido um programa em C que executa a leitura do ficheiro DXF, armazenamento dos parâmetros (coordenadas e cores) e, por último, escrita de um novo ficheiro (.MOD) em linguagem de programação RAPID, com as coordenadas necessárias. Ao mesmo tempo foi fundamental a modelação da célula de trabalho em ambiente de simulação, RobotStudio.

A realização do sistema foi concluída com sucesso, desde a modelação CAD, conversão para RAPID, e teste no robô. Porém, no que diz respeito às dificuldades encontradas, aquando da realização os testes em ambiente real, verificaram-se algumas situações:

- necessidade de colocar uma esponja para uma limpeza mais eficiente, uma vez que depois da limpeza na água o pincel ainda continha restos de tinta;
- a assimetria e irregularidade do pincel, não esboça uma linha de forma constante;
- a quantidade de tinta no pincel vai decrescendo, tornado o desenho com traços irregulares.

6.2 Ideias para Desenvolvimentos Futuros

Em termos de desenvolvimentos futuros é sugerido:

- o estudo da manipulação do pincel ao longo da pintura, usando diferentes inclinações;
- testar a espessura da linha, ou por meio da utilização de outros pincéis ou pela aplicação de *targets* mais abaixo da linha do papel, criando assim pressão no pincel;
- Baixar os *targets* da tinta à medida que vai sendo utilizada a tinta;
- testar o sistema com desenhos mais extensos.

Referências Bibliográficas

[1] Blaise Aguera y Arcas, "Art in the Age of Machine Intelligence,", https://medium.com/artists-and-machine-intelligence/what-is-ami-ccd936394a83, 2016.

[2] P. Neto et al., *Industrial Robot: An International Journal - High-level robot programming based on CAD* (ESMERALD, 2012).

[3] Shimon Y. Nof, *Handbook of Industrial Robotics*, 2 ed. (JOHN WILEY & SONS, INC., 1999).

[4] N. Pires, *Automação Industrial*, 5 ed. (ETEP, 2012).

[5] Thomas R. Kurfess, *Robotics And Automation Handbook* (CRC PRESS, 2005).

[6] Reza N. Jakar, *Theory od Applied Robotics*, 2st ed. (Springer, London, 2010).

[7] M. E. Rosheim, *Robot Evolution: The Development of Anthrobotics*, 1st ed. (John Wiley & Sons, Inc., New York, NY, USA, 1994).

[8] Fernando Pazos, *Automação de Sistemas & Robótica*, 1st ed. (AXCEL Editora, Rio de Janeiro, Brasil, 2002).

[9] Cyberneticzoo, "1954, March – "Positioning or Manipulating Apparatus" patent by Cyril Kenward (British),", http://cyberneticzoo.com/.

[10] Frank MacEachern, "Robotics pioneer George Devol, a former Greenwich resident, dies at 99,", https://www.ctpost.com/local/article/Robotics-pioneer-George-Devol-a-former-Greenwich-2076055.php#photo-1529510, 2011.

[11] Cyberneticzoo, "1934-78 – Spray-paint robot patents – Pollard Jr, Pollard, Roselund and DeVilbiss Comp. – (American),", http://cyberneticzoo.com/.

[12] ABB, "The world's first paint robot,", http://www02.abb.com/global/noabb/noabb071.nsf/bf177942f19f4a98c1257148003b7a0a/678557fdd98b69a8c125720b0031cb06.

[13] Industrial electronics, "Electrical and electronic drawing-Industrial Controls (part 2),", http://www.industrial-electronics.com/eed5th_9b.html.

[14] Vítor Santos, *Robótica Industrial - Apontamentos teóricos* (Universidade Aveiro, 2003).

[15] Manuel Silva, *ROBIN: Robótica Industrial - Atuadores Finais para robôs industriais*

[16] EngEasier, "Robôs Industriais,", http://engeasier.omb10.com/engblog/Robos-Industriais+29738, 2017.

[17] Paulo Abreu, *Robótica Industrial - Fundamentos da robótica e aspetos tecnológicos da robótica* (2001).

[18] Yaskawa, "What is "Robot",", https://www.yaskawa-global.com/product/robotics/about.

[19] Mikell P. Groover, *Fundamentals of Modern Manufacturing*, 4th ed. (Wiley, 2010).

[20] ABB, "Detailed information for: IRC5 Controller,", https://new.abb.com/products/3HAC020536-014/irc5-controller.

[21] ABB, "Technical data for the IRB 120 industrial robot,", https://new.abb.com/products/robotics/industrial-robots/irb-120/irb-120-data.

[22] Richard Moss, "Creative AI: The robots that would be painters,", https://newatlas.com/creative-ai-algorithmic-art-painting-fool-aaron/36106/, 2015.

[23] Martin Gayford, "Robot Art Raises Questions about Human Creativity,", https://www.technologyreview.com/s/600762/robot-art-raises-questions-about-human-creativity/, 2016.

[24] Benjamin Grosser, "Interactive Robotic Painting Machine,", https://bengrosser.com/projects/interactive-robotic-painting-machine/.

[25] Leonel Moura, Henrique Pereira, *A New Kind of Art - Dynamics of Data-Driven Design* (FOOTPRINT, 2014).

[26] Cloudpainter, "An artificially intelligent painting robot,", http://www.cloudpainter.com/.

[27] E. Kac, "Origin and development of robotic art,", http://www.ekac.org/roboticart.html.

[28] Myounghoon Jeon, *Robotic Arts: Current Practices, Potentials, and Implications* (Multimodal Technologies and Interaction, Michigan Technological University, 2017).

[29] Shunsuke Kudoh et al., *Painter Robot: Manipulator of Paintbrush by Force and Visual Feedback* (Elsevier, 2007).

[30] Lino Figueiredo, *Sistemas flexíveis de fabrico - CAD/CAE* (2008).

[31] Brad Strong, "The CAD Format Ladder Part 2,", `https://transmagic.com/cad-format-ladder-part-2/`, 2016.

[32] Cloudpainter, "Multipurpose Industrial Robot for Flexible and compact prodution - ABB Australia,", `http://www.ferret.com.au/`.

[33] Huibin Yang, *DXF File Identification with c for CNC Engraving Machine System* (Scientific Research Publishing, 2014).

[34] ABB, *RAPID Reference Manual - RAPID Overview On-line*

[35] Manuel Silva, *ROBIN:Programação em RAPID* (Instituto Superior de Engenharia do Porto, 2014).

A

Datasheet do robô ABB IRB120

ROBOTICS

IRB 120

ABB's 6 axis robot – for flexible and compact production

The IRB 120 robot is the latest addition to ABB's new fourth-generation of robotic technology. It is ideal for material handling and assembly applications and provides an agile, compact and lightweight solution with superior control and path accuracy.

Compact and lightweight
IRB 120's compact design enables it to be mounted virtually anywhere at any angle without any restriction - for example inside a cell, on top of a machine or close to other robots.

IRB 120 is also the most portable and easy to integrate on the market with its 25 kg weight. The smooth surfaces are easy to clean and the cables for air and customer signals are internally routed, all the way from the foot to the wrist, ensuring that integration is effortless.

Multipurpose
IRB 120 is ideal for a wide range of industries including the electronic, food and beverage, machinery, solar, pharmaceutical, medical and research sectors.

The Food Grade Lubrication (NSF H1) option includes Clean Room ISO Class 5, which ensures uncompromising safety and hygiene for food and beverage applications.

Optimized working range
IRB 120 has a horizontal reach of 580 mm, the best in class stroke, the ability to reach 112 mm below its base and a very compact turning radius.

Fast, accurate and agile
Designed with a light, aluminum structure, the motors ensure the robot is enabled with a fast acceleration, and can deliver accuracy and agility in any application. Using the IRB 120T variant, cycle-times can be reduced up to 25% where the work piece needs extensive reorientation and axis 4, 5 and 6 are predominantly used. This faster versions is well suited for pick and packing applications and guided operations together with PickMaster 3™.

IRC5 Compact controller – optimized for small robots
ABB's new IRC5 Compact controller presents the capabilities of the IRC5 controller in a compact format. It brings accuracy and motion control to applications which have been exclusive to large installations and enables easy commissioning through one phase power input, external connectors for all signals and a built-in expandable 16 in, 16 out, I/O system.

RobotStudio for offline programming enables manufacturers to simulate a production cell to find the optimal position for the robot, and provide offline programming to prevent costly downtime and delays to production.

Reduced footprint
The combination of the new lightweight architecture of the IRB 120 with the new IRC5 Compact controller introduces a significantly reduced footprint.

Specification

Robot version	Reach (m)	Payload (kg)	Armload (kg)
IRB 120-3/0.6	0.58	3*	0.30
Number of axes	6		
Protection	IP30		
Mounting	Any angle		
Controller	IRC5 Compact/IRC5 Single Cabinet		
Integrated signal supply	10 signals on wrist		
Integrated air supply	4 air on wrist (5 bar)		

* 4 with vertical wrist

Performance (according to ISO 9283)

	IRB 120	IRB 120T
1 kg picking cycle		
25 x 300 x 25 mm	0.58 s	0.52 s
25 x 300 x 25 with 180° axis 6 reorientation	0.92 s	0.69 s
Acceleration time 0-1 m/s	0.07 s	0.07 s
Position repeatability	0.01 mm	

Technical information

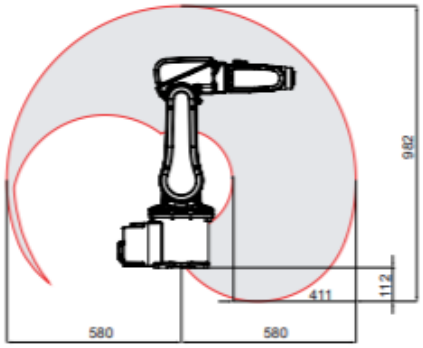
Electrical Connections	
Supply voltage	200-600 V, 50/60 Hz
Rated power transformer rating	3.0 kVA
Power consumption	0.25 kW
Physical	
Dimensions robot base	180 x 180 mm
Dimension robot height	700 mm
Weight	25 kg
Environment	
Ambient temperature for robot manipulator:	
During operation	+5°C (41°F) to +45°C (122°F)
During transportation and storage	-25°C (-13°F) to +55°C (131°F)
During short periods (max. 24 h)	up to +70°C (158°F)
Relative humidity	Max. 95%
Noise level	Max. 70 dB (A)
Safety	Safety and emergency stops 2-channel safety circuits supervision, 3-position enabling device
Emission	EMC/EMI-shielded
Options	Clean Room ISO class 5 (certified by IPA)**

** ISO class 4 can be reached under certain conditions.
Data and dimensions may be changed without notice.

Movement

Axis movement	Working range	Axis max speed IRB 120	Axis max speed IRB 120T
Axis 1 rotation	+165° to -165°	250 °/s	250 °/s
Axis 2 arm	+110° to -110°	250 °/s	250 °/s
Axis 3 arm	+70° to -110°	250 °/s	250 °/s
Axis 4 wrist	+160° to -160°	320 °/s	420 °/s
Axis 5 bend	+120° to -120°	320 °/s	590 °/s
Axis 6 turn	+400° to -400°	420 °/s	600 °/s

Working range



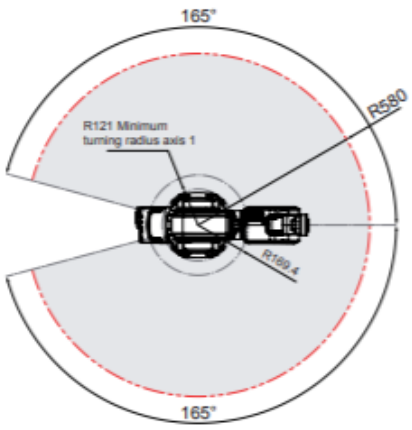


abb.com/robotics

ROBOT14986N_D Rev. H October 2018

B

Programação C

```c
// main.c Programa de conversao DXF para RAPID

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>

const char extjoint[] = "[9E9, 9E9, 9E9, 9E9, 9E9, 9E9]";
const char orient[] = "[0,-0.707106781,0.707106781,0]"; //orientacao dos pontos criados;

void targetiniciais(FILE *to)//targets fixos (agua,tinta,intermedios)
{
    fprintf(to,"\tCONST robtarget pBase :=
    [[154.083,80,-286.952],[0.659888103,0.261826974,-0.245404568,0.660129476],[0,-1,0,0],%s
    ];\n",extjoint);//ponto base
    fprintf(to,"\tCONST robtarget pAgua :=
    [[260,45,10],[0.707106781,0,0,0.707106781],[-1,0,-1,0],%s];\n",extjoint);//ponto agua
    fprintf(to,"\tCONST robtarget pAguamedio :=
    [[260,45,-20],[0.707106781,0,0,0.707106781],[-1,0,-1,0],%s];\n",extjoint);//ponto medio
    cor agua
    fprintf(to,"\tCONST robtarget pAguacima :=
    [[260,45,-70],[0.653281482,0.27059805,-0.27059805,0.653281482],[-1,0,-2,0],%s];\n",
    extjoint);//ponto acima da agua
    fprintf(to,"\tCONST robtarget pTinta_1 :=
    [[161,26,-10],[0.707106781,0,0,0.707106781],[-1,0,-1,0],%s];\n",extjoint);//ponto cor
    preto
    fprintf(to,"\tCONST robtarget pTintamedio_1 :=
    [[161,26,-40],[0.701057385,0.092295956,-0.092295956,0.701057385],[-1,0,-1,0],%s];\n",
    extjoint);//ponto medio cor preto
    fprintf(to,"\tCONST robtarget pTintacima_1 :=
    [[161,26,-70],[0.683012702,0.183012702,-0.183012702,0.683012702],[-1,0,-1,0],%s];\n",
    extjoint);//ponto acima cor preto
    fprintf(to,"\tCONST robtarget pTinta_2 :=
    [[40,25,-10],[0.707106781,0,0,0.707106781],[0,0,0,0],[9E9,9E9,9E9,9E9,9E9,9E9]]; \n"); //
    ponto cor amarelo
    fprintf(to,"\tCONST robtarget pTintamedio_2 :=
    [[40,25,-40],[0.701057385,0.092295956,-0.092295956,0.701057385],[0,-1,0,0],%s];\n",
    extjoint);//ponto medio cor amarelo
    fprintf(to,"\tCONST robtarget pTintacima_2 :=
    [[40.06,25,-70],[0.653281482,0.27059805,-0.27059805,0.653281482],[0,-1,1,0],%s];\n",
    extjoint);//ponto acima cor amarelo
    fprintf(to,"\tCONST robtarget pTinta_3 :=
    [[100,25,-10],[0.707106781,0,0,0.707106781],[0,0,0,0],%s];\n",extjoint);//ponto cor verde
    fprintf(to,"\tCONST robtarget pTintamedio_3 :=
    [[100,25,-40],[0.701057385,0.092295956,-0.092295956,0.701057385],[0,-1,0,0],%s];\n",
    extjoint);//ponto medio cor verde
    fprintf(to,"\tCONST robtarget pTintacima_3 :=
    [[100,25,-70],[0.653281482,0.27059805,-0.27059805,0.653281482],[0,-2,1,0],%s];\n",
    extjoint);//ponto acima cor verde
    fprintf(to,"\tCONST robtarget pTinta_4 :=
    [[40,75,-10],[0.707106781,0,0,0.707106781],[0,0,0,0],%s];\n",extjoint);//ponto cor ciano
    fprintf(to,"\tCONST robtarget pTintamedio_4 :=
    [[40,75,-40],[0.701057385,0.092295956,-0.092295956,0.701057385],[0,-1,0,0],%s];\n",
    extjoint);//ponto medio cor ciano
    fprintf(to,"\tCONST robtarget pTintacima_4 :=
    [[40,75,-70],[0.653281482,0.27059805,-0.27059805,0.653281482],[0,-1,1,0],%s];\n",extjoint
    );//ponto acima cor ciano
    fprintf(to,"\tCONST robtarget pTinta_6 :=
    [[100,75,-10],[0.707106781,0,0,0.707106781],[0,0,0,0],%s];\n",extjoint);//ponto cor
    magenta
    fprintf(to,"\tCONST robtarget pTintamedio_6 :=
    [[100,75,-40],[0.701057385,0.092295956,-0.092295956,0.701057385],[0,-1,0,0],%s];\n",
    extjoint);//ponto medio cor magenta
```

```
    fprintf(to,"\tCONST robtarget pTintacima_6 :=
    [[100,75,-70],[0.653281482,0.27059805,-0.27059805,0.653281482],[0,-2,1,0],[9E9,9E9,9E9,9
    E9,9E9,9E9]]; \n");//ponto acima cor magenta
    fprintf(to,"\tCONST robtarget pTintaesponja_1
    :=[[130,115,-22.5],[0.707106781,0,0,0.707106781],[0,0,0,0],%s];\n",extjoint);
    fprintf(to,"\tCONST robtarget pTintaesponja_2 :=
    [[320,115,-22.5],[0.707106781,0,0,0.707106781],[-1,-1,-1,0],%s];\n",extjoint);
    fprintf(to,"\tCONST robtarget pTintaesponja_3 :=
    [[320,95,-22.5],[0.707106781,0,0,0.707106781],[-1,-1,-1,0],%s];\n",extjoint);
    fprintf(to,"\tCONST robtarget pTintaesponja_4 :=
    [[130,95,-22.5],[0.707106781,0,0,0.707106781],[0,0,0,0],%s];\n",extjoint);
    fprintf(to,"\tCONST robtarget pBase_folha :=
    [[155,250,-100],[0.707106781,0,0,0.707106781],[-1,0,-1,0],%s];\n",extjoint);
}

int testecor(float cor)
{
    if(cor == 250 || cor == 2 || cor == 3 || cor == 4 || cor == 6)
        return 1;

    return 0;
}

int testeponto(float x, float y)
{
    if( x > 0 && x < 297 && y > 0 && y < 210)
        return 1;

    return 0;
}

void funcagua(FILE *to)//limpeza do pincel
{
    fprintf(to,"PROC mov_agua()\n");
    fprintf(to,"\tMOVEL pBase, v500, z100, pincel\\WObj:=Workobject_1;\n");
    fprintf(to,"\tMOVEL pAguacima, v500, fine, pincel\\WObj:=Workobject_1;\n");
    fprintf(to,"\tFOR i FROM 1 TO 10 DO\n");
    fprintf(to,"\t\tMOVEL pAgua, v500, fine, pincel\\WObj:=Workobject_1;\n");
    fprintf(to,"\t\tMOVEL pAguamedio, v500, fine, pincel\\WObj:=Workobject_1;\n");
    fprintf(to,"\tENDFOR ");
    fprintf(to,"\tMOVEL pAguacima, v50, fine, pincel\\WObj:=Workobject_1;\n");
    fprintf(to,"\tMOVEL pBase, v500, z200, pincel\\WObj:=Workobject_1;\n");
    fprintf(to,"\tFOR i FROM 1 TO 5 DO\n");
    fprintf(to,"\t\tMOVEL pTintaesponja_1, v100, fine, pincel\\WObj:=Workobject_1;\n");
    fprintf(to,"\t\tMOVEL pTintaesponja_2, v100, fine, pincel\\WObj:=Workobject_1;\n");
    fprintf(to,"\t\tMOVEL pTintaesponja_3, v100, fine, pincel\\WObj:=Workobject_1;\n");
    fprintf(to,"\t\tMOVEL pTintaesponja_4, v100, fine, pincel\\WObj:=Workobject_1;\n");
    fprintf(to,"\tENDFOR ");
    fprintf(to,"\tMOVEL pBase, v500, z100, pincel\\WObj:=Workobject_1;\n");
    fprintf(to,"ENDPROC\n");
}

void functinta(int tinta, FILE *to)//ir ao frasco de tinta
{
    fprintf(to,"\tMOVEL pTintacima_%d, v500, fine, pincel\\WObj:=Workobject_1;\n", tinta);
    fprintf(to,"\tMOVEL pTintamedio_%d, v50, fine, pincel\\WObj:=Workobject_1;\n", tinta);
    fprintf(to,"\tMOVEL pTinta_%d, v50, fine, pincel\\WObj:=Workobject_1;\n", tinta);
    fprintf(to,"\tWaitTime 2;\n");//esperar 2 segundos
    fprintf(to,"\tMOVEL pTintamedio_%d, v50, fine, pincel\\WObj:=Workobject_1;\n", tinta);
    fprintf(to,"\tMOVEL pTintacima_%d, v50, fine, pincel\\WObj:=Workobject_1;\n", tinta);
    fprintf(to,"\tMOVEL pBase, v500, z100, pincel\\WObj:=Workobject_1;\n");
}
```

```
void confdados(int *dados, float x, float y)//Dados de configuracao dos targets
{
    if(x <= 297/2)//do centro da folha para a esquerda
    {
        dados[0] = 0;
        dados[2] = 0;
    }
    else
    {
        dados[0] = -1;
        dados[2] = -1;
    }
    dados[1] = 0;
    dados[3] = 0;
}

int main(void)
{
    char line[4096]; //numero de carcteres por linha
    char file_name[25];
    char resp;
    FILE *from, *to; // declaracao de ponteiro para comunicacao com o ficheiro
    float num,dis;
    int i, j, tipo, flag, count;
    float **mat;//matriz
    float arco[8];
    float *distancia;
    int dados[4];
    int tintanterior, tintatual, target;
    int linhas = 0, lvector = 1;

    while(1)
    {
        printf("Qual o nome do ficheiro\n");
        gets(file_name);
        strcat(file_name, ".dxf");

        from = fopen(file_name, "r"); // Ler o ficheiro

        if(from != NULL)
        {
            printf("Ficheiro aberto\n");
            break;
        }
        else
        {
            printf("Ficheiro nao existe\n");
            printf("Sair do programa (s ou n)\n");
            gets(&resp);

            if(resp == 's' || resp == 'S')
                exit(0);
        }
    }

    if(( mat = malloc( sizeof( float* ))) == NULL )//linhas
        printf("erro1\n");

    if(( mat[0] = malloc( 11*sizeof( float ) )) == NULL )//colunas
        printf("erro2\n");

    if(( distancia = malloc( sizeof( float ) )) == NULL )//colunas
        printf("erro5\n");

```

```
    while (fgets(line, sizeof(line), from) != 0)
153 {
        if (strncmp(line, "ENTITIES", sizeof("ENTITIES")-1) == 0)
155     {
            flag = 0;
157         tipo = 0;//procura forma geometrica
            count = 0;
159         while (fgets(line, sizeof(line), from) != 0)
            {
161             if(strncmp(line, "LINE", sizeof("LINE") - 1) == 0 && tipo == 0)
                {
163                 tipo = 1;
                    linhas++;//nova linha de matriz
165             }

167             if(strncmp(line, "ARC", sizeof("ARC") - 1) == 0 && tipo == 0)
                {
169                 tipo = 2;
                    linhas++;
171             }

173             if(strncmp(line, "POINT", sizeof("POINT") - 1) == 0 && tipo == 0)
                {
175                 tipo = 3;
                    linhas++;
177             }

179             if(tipo != 4 && tipo != 0)//encontrada forma geometrica
                {
181                 if( lvector <  linhas) //n de linhas criadas menor do que necessarias
                    {
183                     if (( mat = realloc( mat, linhas * sizeof( float* ))) == NULL )//
criar uma nova linha
                            printf("erro3\n");
185                     if (( mat[linhas - 1] = malloc( 11 * sizeof( float ) )) == NULL )//
coluna nova
                            printf("erro4\n");
187                     if (( distancia = realloc( distancia, linhas * sizeof( float*))) ==
NULL )//criar uma nova linha
                            printf("erro5\n");
189                     lvector++;//numero de linhas de matriz criadas
                    }
191
                    mat[linhas - 1][0] = tipo;
193                 tipo = 4;//termina o ciclo de procura de tipo de linha
                }
195
                if(strncmp(line, "AcDbEntity", sizeof("AcDbEntity")-1) == 0 && tipo == 4 &&
flag == 0)
197                 flag = 1;//flag para ler cor

199             if(strncmp(line, " 62", sizeof(" 62")-1) == 0 && tipo == 4 && flag == 1)
                {
201                 fgets(line, sizeof(line), from); //10
                    mat[linhas - 1][1] = atof(line);  // matriz da ultima linha que esta, na
2 coluna. atof - float
203                 flag = 0;//termina leitura de cor

205                 if(testecor(mat[linhas - 1][1]) == 0)
                    {
207                     linhas--; //nao cria outra linha
                        tipo = 0;
209                 }
                }
```

```
211
                if(strncmp(line, "AcDbLine", sizeof("AcDbLine")-1) == 0 && tipo == 4)
213             {
                    for(i = 2; i < 8; i++)
215                 {
                        fgets(line, sizeof(line), from); //10,20,30,11,21,31
217                     fgets(line, sizeof(line), from);//valor a guardar
                        mat[linhas-1][i] = atof(line);
219                 }

221                 distancia[linhas - 1] = sqrt((mat[linhas - 1][2] - mat[linhas - 1][5]) *
    (mat[linhas - 1][2] - mat[linhas - 1][5]) + (mat[linhas - 1][3] - mat[linhas - 1][6]) * (
    mat[linhas - 1][3] - mat[linhas - 1][6]));//calculo do tamanho da linha

223                 if(testeponto( mat[linhas-1][2], mat[linhas-1][3]) == 0 && testeponto(
    mat[linhas-1][5], mat[linhas-1][6]) == 0 )
                    {
225                         linhas--;
                    }
227                 tipo = 0;//inicia procura forma geometrica
                }
229
                if(strncmp(line, "AcDbCircle", sizeof("AcDbCircle")-1) == 0 && tipo == 4)
231             {
                    for(i = 0; i < 6; i++)
233                 {
                        fgets(line, sizeof(line), from); //10,20,30,40,50,51
235                     fgets(line, sizeof(line), from);
                        arco[i] = atof(line);//vector auxiliar
237
                        if(i == 3)
239                     {
                            fgets(line, sizeof(line), from);
241                         fgets(line, sizeof(line), from); //passar  "AcDbArc" a frente
                        }
243                 }

245                 if(arco[5] < arco[4])//angulo final < angulo inicial
                        arco[5]+= 360;//para o calculo do angulo medio
247
                    arco[6] = (arco[4] + arco[5]) / 2;//angulo medio
249
                    for(i = 0; i < 3; i++ )
251                 {
                        mat[linhas-1][2 + 3 * i] = arco[0] + arco[3] * cos(arco[4 + i] * M_PI
     / 180); //x
253                     mat[linhas-1][3 + 3 * i] = arco[1] + arco[3] * sin(arco[4 + i] * M_PI
     / 180); //y
                        mat[linhas-1][4 + 3 * i] = arco[2];  //z
255                 }

257                 arco[7] = arco[5] - arco[4];

259                 distancia[linhas - 1] = (arco[7] * M_PI * arco[3])/180;

261                 if(testeponto( mat[linhas-1][2], mat[linhas-1][3]) == 0 && testeponto(
    mat[linhas-1][5], mat[linhas-1][6]) == 0 && testeponto( mat[linhas-1][8], mat[linhas
    -1][9]) == 0)
                    {
263                     linhas--;
                    }
265                 tipo = 0;
                }
267
```

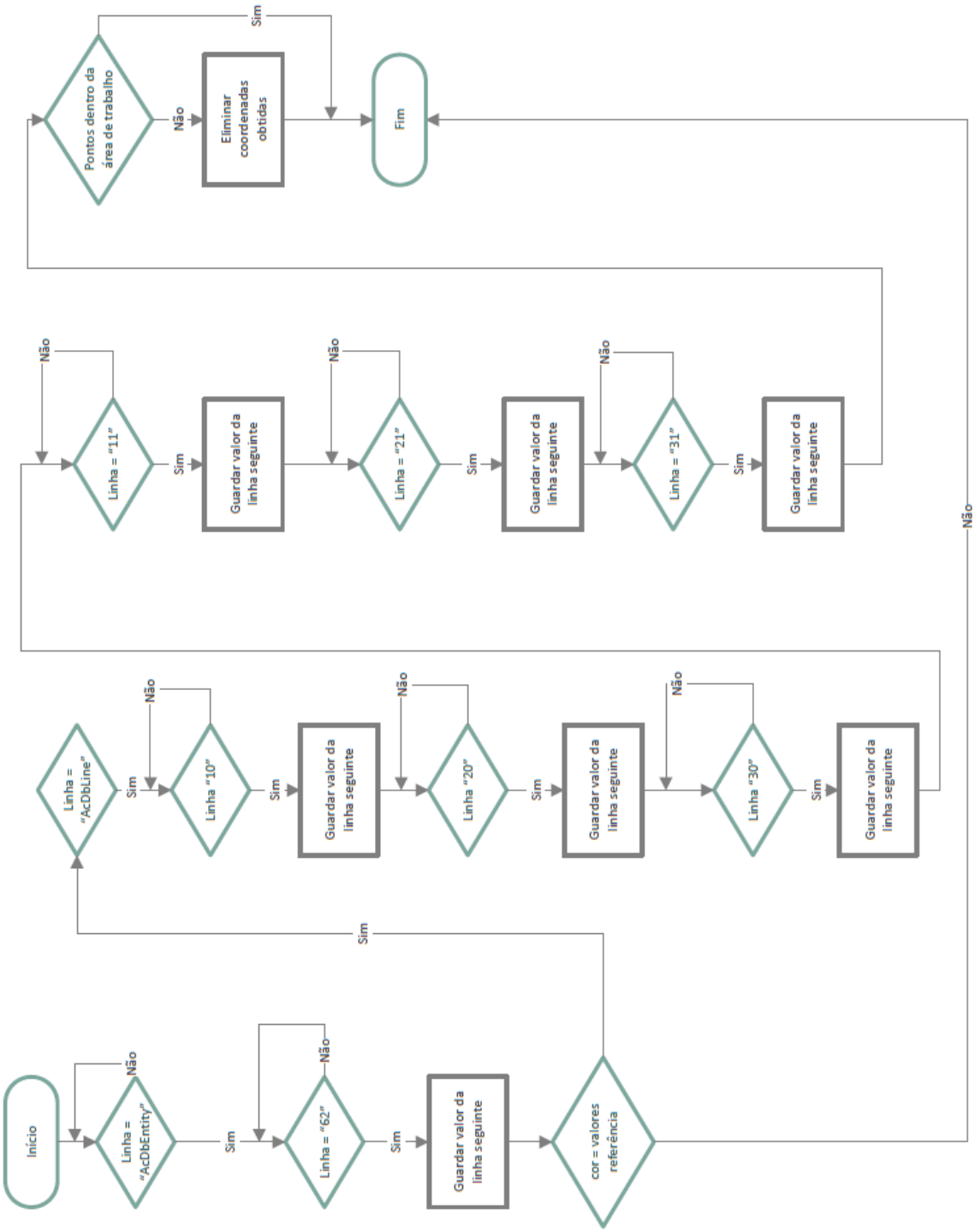
```
                if(strncmp(line, "AcDbPoint", sizeof("AcDbPoint")-1) == 0 && tipo == 4)
                {
                    for(i = 2; i < 5; i++)
                    {
                        fgets(line, sizeof(line), from); //10,20,30
                        fgets(line, sizeof(line), from);
                        mat[linhas-1][i] = atof(line);
                    }

                    distancia[linhas - 1] = 10;

                    if(testeponto( mat[linhas-1][2], mat[linhas-1][3]) == 0)
                        linhas--;

                    tipo = 0;
                }
            }
        }
    }
    fclose(from);//fechar ficheiro DXF

    //variaveis necessarias para a organizacao da matriz
    float codcor[] = { 2, 3, 4, 6, 250};
    int cor, l = 0;
    float *save;

    //organizar matriz em relacao a cor
    for(cor = 0; cor < 5; cor++)
    {
        for (i = 0 ; i < linhas; i++)//target
        {
            if(mat[i][1] == codcor[cor])
            {
                if(l == i)
                    l++;
                if(l < i)
                {
                    save = mat[l];
                    mat[l] = mat[i];
                    mat[i] = save;
                    l++;
                }
            }
        }
    }

    //indicacao do numero de formas a desenhar
    printf("Criado matrix com %d linhas\n",linhas);

    to = fopen("Module1.mod", "w+");

    fprintf(to,"MODULE Module1 \n");

    targetiniciais(to);//target definidos atraves do robotstudio e ambiente real, ex,tintas e
     agua

    for (i = 0 ; i < linhas; i++)//target
    {
        confdados(dados, mat[i][2], mat[i][3]);//obter a configuracao de dados do target

        fprintf(to,"\tCONST robtarget pInicial_%d := [ [%f, %f, %f], %s, [%d, %d, %d, %d], %s
     ]; \n", i + 1, mat[i][2], mat[i][3], mat[i][4] - 0.5, orient, dados[0], dados[1], dados
    [2], dados[3], extjoint);//ponto inicial
```

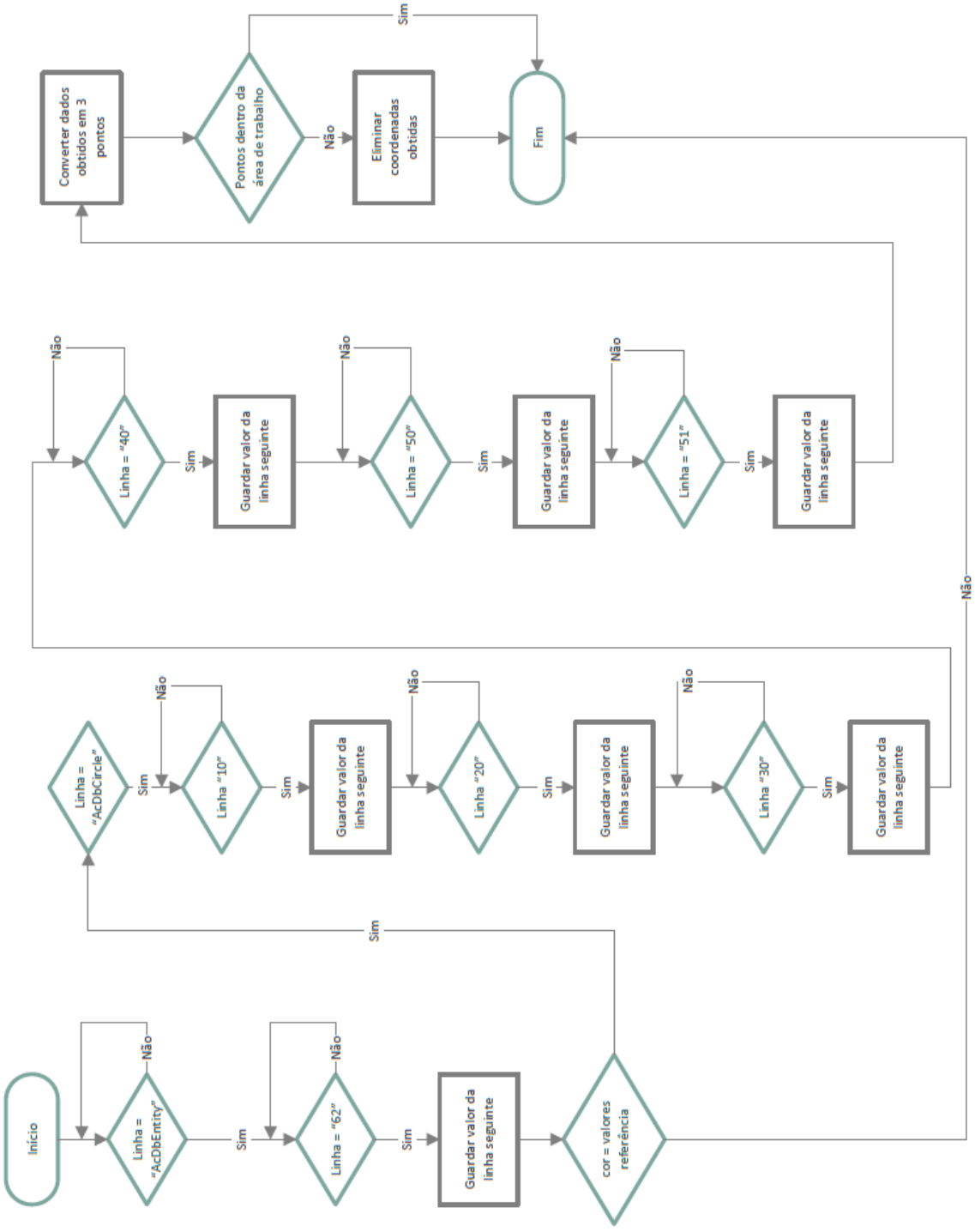
```
329         if(mat[i][0] != 3)//linha ou arco
            {
331             confdados(dados, mat[i][5], mat[i][6]);
                fprintf(to,"\tCONST robtarget pFinal_%d := [ [%f, %f, %f], %s, [%d, %d, %d, %d],
    %s ]; \n", i + 1, mat[i][5], mat[i][6], mat[i][7] - 0.5, orient, dados[0], dados[1],
    dados[2], dados[3], extjoint);//ponto final
333         }

335         if(mat[i][0] == 2)//arco
            {
337             confdados(dados, mat[i][8], mat[i][9]);
                fprintf(to,"\tCONST robtarget pAux_%d := [ [%f, %f, %f], %s, [%d, %d, %d, %d], %s
     ]; \n", i + 1, mat[i][8], mat[i][9], mat[i][10] - 0.5, orient, dados[0], dados[1], dados
    [2], dados[3], extjoint);//ponto auxarco
339         }
        }
341
        funcagua(to);//escrever a funcao agua no rapid
343     fprintf(to,"PROC mov_pincel()\n");//inicio da funcao de desenho

345     target = 100;//100 - base; entre 101 e 199 - ponto inicial da linha;  entre 200 e 299 -
    ponto final da linha; >300 - ponto   auxiliar da linha;
        tintatual = 0;
347     tintanterior = 0;
        dis = 0;
349
        for(i = 0; i < linhas; i++)
351     {
            tintatual = (int)mat[i][1];
353         if(tintatual == 250)
                tintatual = 1;
355
            if(tintanterior != tintatual)
357         {
                tintanterior = tintatual;
359
                if(target > 101 + i )//se target na folha
361                 fprintf(to,"\tMOVEL pBase_folha, v500, z100, pincel\\WObj:=Workobject_1;\n");
    //ponto base folha

363             fprintf(to,"\tmov_agua;\n");
                functinta(tintatual, to);
365             target = 100;
                dis = 0;
367
            }
369
            if(dis >= 100 || (dis + distancia[i]) > 120)//recarga de tinta
371         {
                fprintf(to,"\tMOVEL pBase_folha, v500, z100, pincel\\WObj:=Workobject_1;\n");//
    ponto base folha
373             fprintf(to,"\tMOVEL pBase , v500, z100, pincel\\WObj:=Workobject_1;\n");//ponto
    base
                functinta(tintatual, to);
375             dis = 0;
            }
377
            if(target != 101 + i )
379             fprintf(to,"\tMOVEL pBase_folha, v500, z100, pincel\\WObj:=Workobject_1;\n");//
    ponto base folha

381         fprintf(to,"\tMOVEL pInicial_%d, v50, fine, pincel\\WObj:=Workobject_folha;\n", i +
    1);//ponto inicial
```

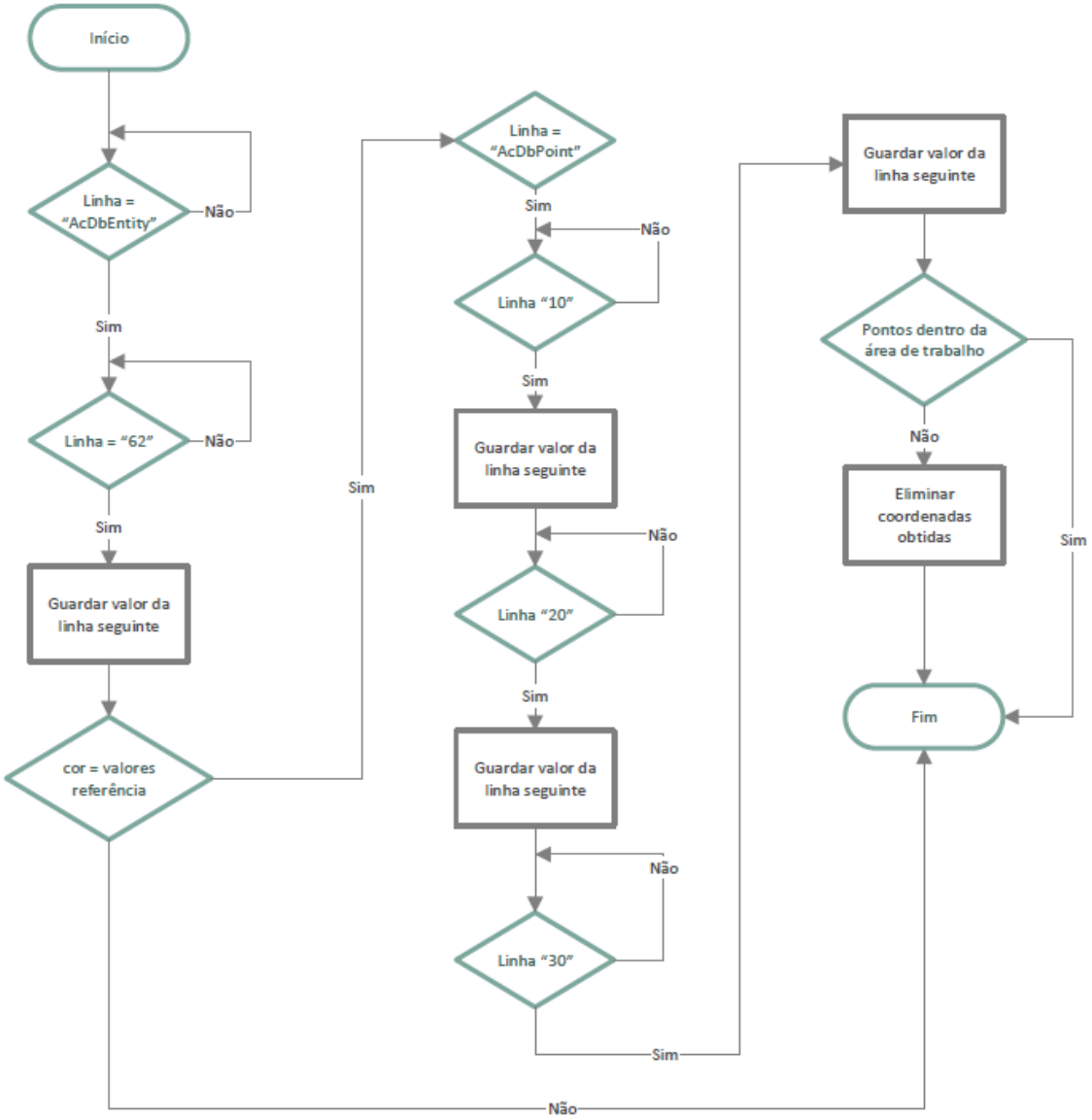
```
        if(mat[i][0] == 1)// linha
        {
            fprintf(to,"\tMOVEL pFinal_%d, v50, fine, pincel\\WObj:=Workobject_folha;\n", i +
 1);//ponto final
            dis += distancia[i];
        }

        if(mat[i][0] == 2)// arco
        {
            fprintf(to,"\tMOVEC pAux_%d, pFinal_%d, v50, fine, pincel\\WObj:=Workobject_folha
;\n", i + 1, i + 1);//ponto final
            dis += distancia[i];
        }

        if(mat[i][0]  == 3) // ponto
            j = 0;
        else
            j = 3;

        if(i < linhas -1 && abs(mat[i][2 + j] - mat[i+1][2]) < 1 && abs(mat[i][3 + j] - mat[i
+1][3]) < 1)//verificar se o ponto do desenho e proximo do ponto do proximo desenho
            target = 101 + i + 1;
        else
            target = 201 + i;
    }

    if(target != 101 + i )
        fprintf(to,"\tMOVEL pBase_folha, v500, z100, pincel\\WObj:=Workobject_1;\n");//ponto
    base folha

    fprintf(to,"\tmov_agua;\n");
    fprintf(to,"\tMOVEL pBase, v500, fine, pincel\\WObj:=Workobject_1;\n");//ponto base
    fprintf(to,"ENDPROC \n");
    fprintf(to,"PROC main()\n");
    fprintf(to,"\tmov_pincel;\n");
    fprintf(to,"ENDPROC \n");
    fprintf(to,"ENDMODULE \n");
    printf("Programa Rapid criado\n");

    fclose(to); //fechar ficheiro

    for (i = 0 ; i < lvector; i++)
     free(mat[i]);//libertar espaco usado pelas linhas

    free(mat);//libertar espaco usado pela colunas

    return 0;
}
```

C

Fluxogramas da extração de dados do ficheiro DXF


Início
Linha = "AcDbEntity"
Não
Sim
Linha = "62"
Não
Sim
Guardar valor da linha seguinte
cor = valores referência
Não
Sim
Linha = "AcDbLine"
Não
Sim
Linha "10"
Não
Sim
Guardar valor da linha seguinte
Linha "20"
Não
Sim
Guardar valor da linha seguinte
Linha "30"
Não
Sim
Guardar valor da linha seguinte
Não
Sim
Linha = "11"
Não
Sim
Guardar valor da linha seguinte
Linha = "21"
Não
Sim
Guardar valor da linha seguinte
Linha = "31"
Não
Sim
Guardar valor da linha seguinte
Pontos dentro da área de trabalho
Não
Eliminar coordenadas obtidas
Sim
Fim


Início
Linha = "AcDbEntity"
Não
Sim
Linha = "62"
Não
Sim
Guardar valor da linha seguinte
cor = valores referência
Não
Sim
Linha = "AcDbCircle"
Não
Sim
Linha "10"
Sim
Guardar valor da linha seguinte
Não
Linha "20"
Sim
Guardar valor da linha seguinte
Não
Linha "30"
Sim
Guardar valor da linha seguinte
Não
Linha = "40"
Sim
Guardar valor da linha seguinte
Não
Linha = "50"
Sim
Guardar valor da linha seguinte
Não
Linha = "51"
Sim
Guardar valor da linha seguinte
Converter dados obtidos em 3 pontos
Pontos dentro da área de trabalho
Não
Eliminar coordenadas obtidas
Sim
Fim


Início
Linha = "AcDbEntity"
Não
Sim
Linha = "62"
Não
Sim
Guardar valor da linha seguinte
cor = valores referência
Sim
Linha = "AcDbPoint"
Sim
Não
Linha "10"
Sim
Guardar valor da linha seguinte
Não
Linha "20"
Sim
Guardar valor da linha seguinte
Não
Linha "30"
Sim
Guardar valor da linha seguinte
Pontos dentro da área de trabalho
Não
Eliminar coordenadas obtidas
Sim
Fim
Não