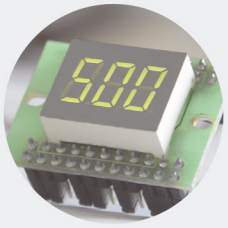
INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES






Robô para Emulação de Tarefas de Pintura Manual

TIAGO FERRAZ LARANJA PONTES
novembro de 2020

POLITÉCNICO
DO PORTO

Robô para Emulação de Tarefas de Pintura Manual

Tiago Ferraz Laranja Pontes

Mestrado em Engenharia Eletrotécnica e de Computadores
Área de Especialização de Automação e Sistemas
Departamento de Engenharia Eletrotécnica
Instituto Superior de Engenharia do Porto
2020

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: Tiago Ferraz Laranja Pontes, Nº 1141158, 1141158@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
12 de Novembro de 2020

Agradecimentos

Os meus sinceros agradecimentos a todos aqueles que me apoiaram e me motivaram para conseguir finalizar o meu projeto. O resultado final não se deve só a mim mas sim a todos aqueles que rodearam no decorrer deste ano.

Primeiro quero agradecer ao Engenheiro Manuel Silva, pela qualidade que demonstrou em toda a orientação da presente tese, pela sua disponibilidade, e sobretudo pelo elevado critério de exigência que aplicou na ajuda e na correção deste projeto.

Manifesto também um agradecimento muito especial a todos os docentes desta instituição pelo elevado grau de profissionalismo que sempre me demonstraram. Aos meus colegas gostaria de agradecer por todo o companheirismo e amizade manifestados no decorrer de toda a minha vida académica.

Por fim, gostaria de agradecer de uma forma muito especial à minha família. À minha Mãe e ao meu Pai por todo o esforço que fizeram para me proporcionar um percurso escolar de qualidade. Aos meus irmãos Rita e Bernardo e à minha namorada Teresa que me motivaram a nunca desistir, e sempre acreditaram no meu sucesso.

Resumo

A robótica já não faz parte do futuro mas sim do presente. Em pouco tempo esta tecnologia demonstrou ser a solução mais viável em muitos problemas. Os principais objetivos da robotização de um processo são a redução de custos e o aumento da produtividade sendo estes os benefícios que estão enraizados em qualquer tipo de negócio. Contudo com o aumento na procura destas soluções a robótica industrial tem vindo a tornar-se mais acessível e o número de projetos, nesta área, tem vindo a aumentar exponencialmente. Hoje em dia, uma máquina que possua tecnologia como Visão ou Inteligência Artificial aumenta o seu leque de soluções. Por exemplo, é capaz de se exprimir artisticamente ou aprender com o seu próprio trabalho sem a intervenção do ser humano, automatizando assim a tarefa que está a desempenhar.

Outras ferramentas capazes de se apresentarem como grandes soluções para o aumento da produtividade e automatização de processos são os programas *Computer Aided Design* (CAD). Esta dissertação foca-se numa solução que interliga a robótica e as ferramentas CAD com o principal objetivo de comprovar o que estas duas tecncologias são capazes. Assim, o presente projeto pretende demonstrar uma solução que visa aplicar uma pintura num objeto, de forma automática, tendo por base o desenho importado de um *software* CAD. Para tal foi desenvolvido um programa, em linguagem Python, que converte o desenho CAD em instruções para o robô executar a sua tarefa de pintura.

Em suma, foi necessário fazer uma pesquisa aprofundada dos diferentes tipos de ficheiro e técnicas de importação de ficheiros CAD, e analisadas e testadas possíveis formas de extração de informação relevante do ficheiro em questão.

Com a realização deste projeto, verificou-se a potencialidade destas duas tecnologias juntas, e espera-se que no futuro aparecerão inúmeros projetos baseados na robótica e nas ferramentas CAD.

Palavras-Chave: Robótica, visão e inteligência artificial, CAD, Python, pintura manual.

Abstract

Robotics is no longer part of the future, but of the present. In a short time this technology will adjust to be the most viable solution in many problems. The main objectives of robotizing a process are to reduce costs and increase productivity, these benefits being rooted in any type of business. However, with the increase in the demand for solutions, industrial robotics has become more accessible and the number of projects in this area has increased exponentially. Nowadays, a machine that has technology like Computer Vision or Artificial Intelligence increases its range of solutions. For example, its able to express artistically or learn from its own work without human intervention.

Other tools capable of presenting themselves as solutions for increasing productivity and automating processes are Computer Aided Design (CAD) programs. This dissertation focuses on a solution that links robotics and CAD tools with the main objective of proving what these two technologies are capable of. Thus, the present project intends to demonstrate a solution that aims to apply a painting to an object, automatically, based on the drawing imported from a CAD software. For this purpose, a program was developed, in Python language, that converts the CAD drawing into instructions for the robot to perform its painting task.

In short, it was necessary to do an intensive search of the different types of CAD files, and possible ways of extracting relevant information from the file.

With the realization of this project, the potential of these two technologies was verified together, and it is expected that in the newer future will appear many projects based on robotics and CAD tools.

Keywords: robotics, computer vision, artificial intelligence, CAD, Python, manual painting.

Índice

Índice de Figuras

Índice de Tabelas

Acrónimos

Acrónimo	Descrição	Página
ANSI	*American National Standards Institute*	21
CAD	*Computer Aided Design*	2
IDE	*Integrated Development Enviornment*	25
IGES	*The Initial Graphics Exchange Specification*	20
IoT	*Internet of Things*	7
ISO	*International Organization for Standardization*	21
PDF	*Portable Document Format*	47
RAE	Real Academia Espanhola	15
RGB	*Red Blue Green*	28
R.U.R	*Rossum's Universal Robots*	5
SCARA	*Selective Compliant Articulated Robotic Arm*	11
STEP	*Standard for the Exchange of Product Data*	20
TEDI	Tese / Dissertação	1

Capítulo 1

Introdução

Neste capítulo pretende-se apresentar o presente projeto, introduzindo o problema que o mesmo visa solucionar e enquadrando-o, igualmente, na área de estudos em que o mesmo surge. Para além disso, apresentar-se-á a metodologia adotada, bem como, a calendarização e a organização desta dissertação.

1.1 Contextualização

Este projeto surge do interesse pela área de robótica e programação e foi elaborado no âmbito da unidade curricular de Tese e Dissertação (TEDI), que se insere no Mestrado de Engenharia Electrotécnica e de Computadores (MEEC).

O atual contexto económico exige da indústria um constante esforço de otimização da produção através da redução de custos e do aumento da produtividade. Acresce que, a globalização da economia determina, igualmente, que a indústria se destaque e diferencie dos seus pares, através da criação de produtos novos e, sobretudo, de modelos personalizados. Por outro lado, a crescente procura e dinamização da utilização da robótica determina a incorporação destes mecanismos, não só em contextos industriais, mas também noutros ambientes, como é o caso da arte e do *design*.

1.2 Objetivos

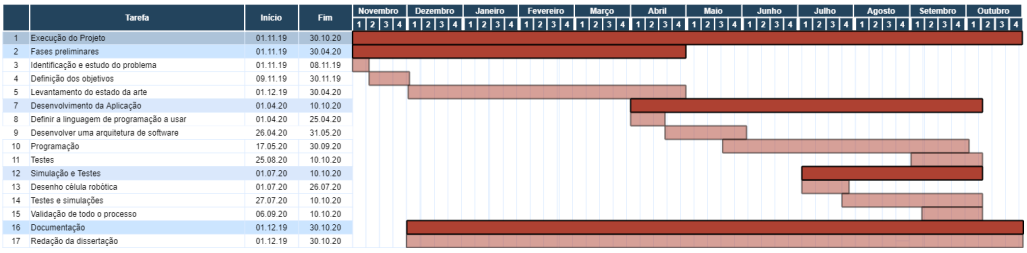
O projeto apresentado na presente dissertação visa flexibilizar e facilitar os processos de pintura e personalização de objetos, garantindo a sua execução de forma automática. Para tal, foi criada uma aplicação capaz de converter um desenho de formato *Drawing Exchange Format* (DXF), num ficheiro com linguagem entendida pelo robô, que executará a pintura pretendida no objeto.

1.3 Metodologia

Tendo em vista a concretização dos objetivos supra enunciados, foi necessário recolher informação relativa à robótica e aos procedimentos de pintura automática já existentes, por forma a criar um sistema totalmente capaz de cumprir com os requisitos pretendidos. Foi, igualmente, necessário analisar e estudar os diversos formatos de exportação de desenho *Computer Aided Design* (CAD), por forma a selecionar o tipo de ficheiro que melhor se adequa ao presente projeto. A fase preliminar de estudo e reflexão acima descrita permitiu desenvolver o projeto de execução que serviu de base à criação da solução para o problema identificado. Assim, definiram-se todas as etapas necessárias à concretização do projeto, designadamente a criação de uma aplicação que permitisse converter um desenho CAD numa linguagem entendida pelo robô; a criação da célula robótica composta por todos os componentes necessários à demonstração do resultado final; o ensaio do comportamento do programa gerado e, por último, a execução da simulação do normal desempenho de todo o processo.

1.4 Calendarização

O diagrama de Gantt ilustrado na Figura 1.1 sumariza o plano de trabalhos para este projeto.



	Tarefa	Início	Fim
1	Execução do Projeto	01.11.19	30.10.20
2	Fases preliminares	01.11.19	30.04.20
3	Identificação e estudo do problema	01.11.19	08.11.19
4	Definição dos objetivos	09.11.19	30.11.19
5	Levantamento do estado da arte	01.12.19	30.04.20
7	Desenvolvimento da Aplicação	01.04.20	10.10.20
8	Definir a linguagem de programação a usar	01.04.20	25.04.20
9	Desenvolver uma arquitetura de software	26.04.20	31.05.20
10	Programação	17.05.20	30.09.20
11	Testes	25.08.20	10.10.20
12	Simulação e Testes	01.07.20	10.10.20
13	Desenho célula robótica	01.07.20	26.07.20
14	Testes e simulações	27.07.20	10.10.20
15	Validação de todo o processo	06.09.20	10.10.20
16	Documentação	01.12.19	30.10.20
17	Redação da dissertação	01.12.19	30.10.20

Figura 1.1: Diagrama de Gantt com a calendarização do projeto

1.5 Organização do Relatório

O presente relatório encontra-se dividido em seis capítulos. O primeiro capítulo designa-se 'Introdução' e tem em vista a apresentação do projecto bem como, do problema que o mesmo visa resolver. Neste capítulo são ainda partilhados os objetivos a que se propõe a presente dissertação bem como, a identificação da metodologia adotada para o desenvolvimento do projeto. Segue-se o segundo capítulo, intitulado 'Revisão Bibliográfica', através do qual são definidos e contextualizados os conceitos em causa na presente dissertação, sendo ainda feita referência aos projetos já existentes na área. No terceiro capítulo, designado 'Ar-

quitetura Proposta para a Solução' é apresentada a estrutura da possível solução ao problema identificado. Posteriormente, no quarto capítulo, intitulado 'Implementação da Solução' é evidenciado todo o processo de criação e implementação da solução encontrada. Posteriormente, no quinto capítulo, denominado 'Testes da Solução e Resultados Obtidos', são exibidos os ensaios e testes realizados à solução proposta bem como, os resultados dos mesmos. Por fim, é encerrada a presente dissertação através da 'Conclusões', que se insere no sexto capítulo, e que visa apresentar, de forma detalhada, os problemas encontrados ao longo de todo o processo de criação do presente projeto bem como, a apresentação de eventuais e possíveis desenvolvimentos ao mesmo com vista à sua melhoria.

Capítulo 2

Revisão Bibliográfica

"One machine can do the work of fifty ordinary men. No machine can do the work of one extraordinary man"

-Elbert Hubbard

2.1 Robótica

O termo robô, originário da palavra checa 'robota', foi introduzido pelo dramaturgo Karel Capek, de nacionalidade checa, numa peça denominada de *Rossum's Universal Robots* (R.U.R), onde os robôs idealizados pelo artista não eram mecânicos, mas sim criados através de processos químicos, onde estes eram substitutos da mão de obra humana. O conceito de robótica foi pela primeira vez enunciado em 1942, pelo cientista e escritor Isaac Asimov, que se refere ao estudo e utilização de robôs nas mais diversas áreas. O desenvolvimento destas máquinas teve início em meados do século vinte. Os robôs mais 'modernos', produzidos nessa altura, foram utilizados para a teleoperação de materiais radioativos e para aplicações prostéticas.

O primeiro robô industrial foi desenvolvido por George Devol, em meados de 1950, designado de Unimate, apresentado na Figura 2.1, tinha a fisionomia de um braço que era usado para o transporte de peças fundidas na fábrica da General Motors em Nova Jersey [1].

Inicialmente a robótica era apenas curiosidade que interligava a ciência e tecnologia de última geração. Contudo, com as descobertas nas áreas da matemática, da informática e das ciências, começou-se a empregar robôs nas mais diversas áreas, protegendo assim os trabalhadores de tarefas monótonas ou em ambientes hostis.

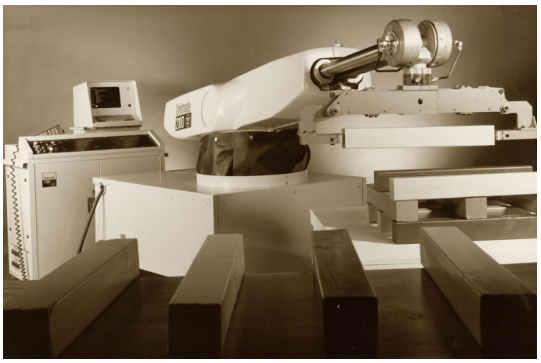


Figura 2.1: Robot Unimate - O primeiro robô industrial [1]

2.1.1 Robótica Industrial

A indústria surgiu nos finais do século XVIII e início do século XIX, esta emergiu no período de proto-industrialização onde o mercado era gerido através de produtos fabricados em ambientes artesanais e domésticos. Este aparecimento denominou-se de Primeira Revolução Industrial. Esta revolução trouxe mudanças ao nível da mecanização, uso de máquinas ou ferramentas para substituir ou auxiliar o trabalho humano. A preferência por esta técnica foi o que originou a substituição da agricultura pela indústria como o grande suporte da economia social. Na Figura 2.2 observa-se o surgimento das primeiras fábricas.

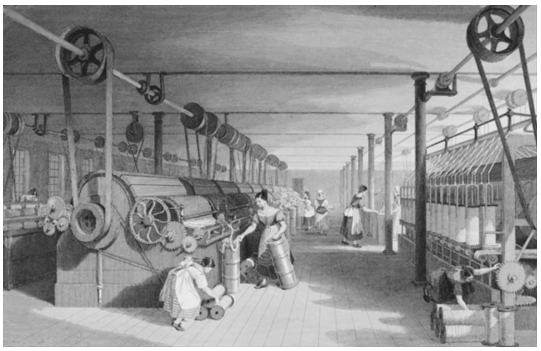


Figura 2.2: Primeira revolução industrial [2]

Um século depois, o mundo começa-se a aproximar da segunda revolução industrial, iniciando-se esta nos finais do século XIX. Graças aos avanços tecnológicos vivenciados naquela época surgiram novas fontes de energia como a eletricidade, o gás e o petróleo. Como resultado desta revolução houve o aparecimento do motor de combustão , começando este a demonstrar todo o seu potencial nas mais diversas áreas. Com esta revolução surgiu o aparecimento do mercado do aço, síntese química e novos meios de comunicação como o telégrafo e o telefone. O desenvolvimento do motor de combustão possibilitou a criação de

dois elementos essenciais que mudaram o mundo, o automóvel e o avião. Na Figura 2.3 observa-se uma linha de produção de automóveis criada por Henry Ford. Estas são as principais razões pelas quais esta revolução é considerada ainda nos dias de hoje a mais importante.

Figura 2.3: Linha de produção de automóveis

Na segunda metade do século XX assistiu-se à terceira revolução industrial, surgindo uma nova fonte de energia com um elevado potencial, a energia nuclear. Neste período, houve o aparecimentos da eletrónica, das telecomunicações e dos computadores. Estas tecnologias tiveram um papel muito importante, abrindo portas à exploração espacial, investigações e às biotecnologias. No mundo da indústria, surgiram duas invenções indispensáveis para a automação dos processos produtivos nas fábricas, os *Programmable Logic Controllers* (PLC) e os robôs [2].

Pensa-se que no presente momento, a indústria está a atravessar a quarta revolução industrial, mais comummente designada de indústria 4.0. Este conceito foi criado por Klaus Schwab. Para o economista e engenheiro a industrialização atingiu uma quarta fase, que "transformará fundamentalmente a forma como vivemos, trabalhamos e nos relacionamos"[7]. Assim, esta transformação é uma mudança de paradigma e não apenas mais uma etapa de desenvolvimento tecnológico. A *Internet of Things* (IoT), a inteligência artificial e a robótica são alguns dos campos onde é possível verificar uma grande mudança no quotidiano das pessoas, mas sobretudo das empresas. No Figura 2.4 é apresentado um gráfico elaborado por Kagermann [8], que ilustra as quatro revoluções, evidenciando que à medida que se passou pelas revoluções a complexidade dos sistemas na indústria foi evoluindo de uma forma exponencial.

2.1.1.1 Exemplo do uso da robótica industrial na pintura

A produção de automóveis em massa é utilizada por diversas marcas de automóveis. A Ford Motor Company, sob a liderança de Henry Ford, revolucionou esta indústria, criando uma linha de montagem em movimento. Assim o veículo é

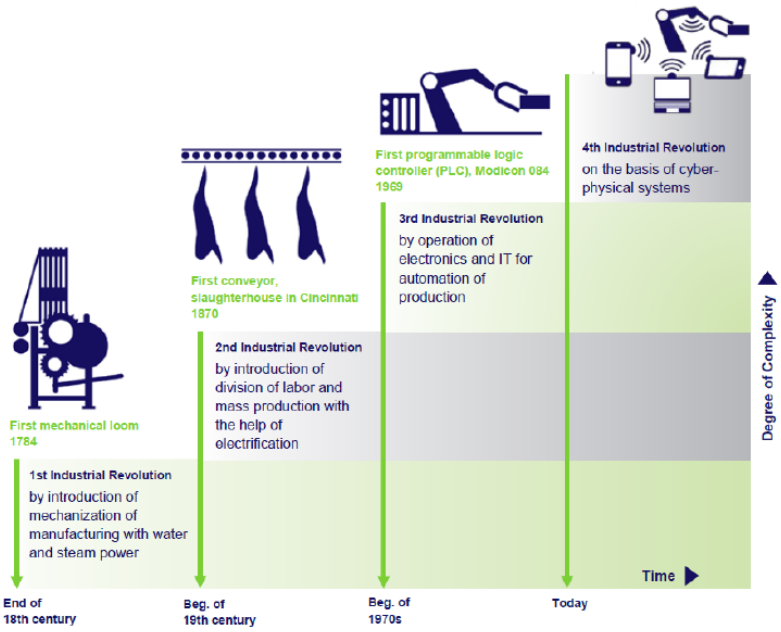


Figura 2.4: Gráfico ilustrativo explicitando as quatro revoluções industriais e apresentando acontecimentos e datas marcantes

produzido em série e à medida que atravessa a linha vão-lhe sendo acrescentadas propriedades que fazem com que este no fim se torne num produto pronto para ser comercializado. Porém, para que uma linha de produção em série obtenha bons resultados em termos de cadência, é necessário garantir que todas as etapas, Figura 2.5, trabalhem ao mesmo ritmo, evitando assim estrangulamentos na linha. Assim, para aumentar a eficiência recorre-se a robôs diminuindo o tempo de ciclo de cada etapa, aumentando assim a eficiência total da linha [9].

Figura 2.5: Etapas de uma linha de montagem em massa de automóveis

Uma das fases onde é exigida mais eficiência é a pintura, pois é uma tarefa precisa, morosa e torna o ambiente de trabalho hostil. Desde muito cedo que o uso de robôs nesta etapa é adotada pelas grandes fábricas de automóveis. O local onde é feita a pintura é designado por *paint shop*. Um *paint shop* robotizado é constituído por um ou mais robôs que são dispostos num carril, fazendo com que

estes se movam em parceria com o carro, um sistema de pintura, capaz de fazer a alteração da cor em segundos, e ventiladores, deixando o ambiente de pintura ventilado e livre de poeiras e sujidade. A tipologia dos robôs utilizados nesta área são os robôs manipuladores.

2.1.2 Robô Manipulador

Desde o seu aparecimento procura-se uma definição que espelhe todas as aplicações e funcionalidades desta máquina, contudo essa definição ainda não existe. Porém, de uma forma ampla, define-se robô qualquer máquina automática que execute funções atribuídas a seres humanos [10].

O manipulador robótico é constituído por diversos elementos essenciais para o seu correto funcionamento, como se pode observar no esquema da Figura 2.6. O manipulador, a unidade de potência, o controlador, sensores e atuadores.

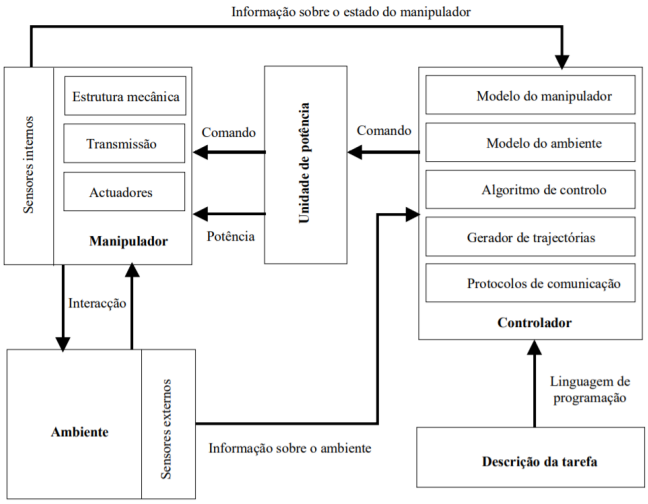


Figura 2.6: Esquema de um manipulador robótico

2.1.2.1 Estrutura Mecânica

Um manipulador é mecânicamente concebido para posicionar no espaço o seu órgão terminal: garra ou ferramenta. Este é composto pela base, pelo braço e pelo punho que são constituidos por partes rígidas, elos, interligadas entre si através das juntas, Figura 2.7.

A base é o apoio do robô, esta poderá estar fixa ou acoplada a um carril que permite ao robô deslocar-se linearmente para executar várias tarefas ao longo de uma linha de produção. O braço é normalmente associado ao posicionamento no espaço operacional do robô. Por fim, o punho permite orientar a garra de modo a facilitar o robô na execução de uma determinada tarefa.

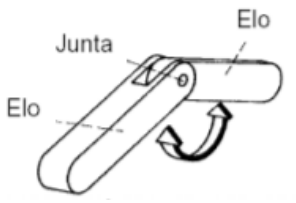


Figura 2.7: Elos e juntas

As juntas de ligação podem ser de quatro tipos diferentes, rotacionais, prismáticas (lineares), de torção ou revolventes. As juntas prismáticas apresentam um movimento linear em um só eixo. As rotacionais rodam no eixo prependicular ao eixo da ligação dos elos. Contrariamente às rotacionais as de torção rodam no eixo paralelo ao eixo de ligação. Por último as de revolução há rotação no eixo paralelo à ligação de entrada e prependicular ao de saída. O tipo de junta define a configuração do robô.

Um robô que possua apenas movimentos lineares (LLL) é designado de cartesiano, Figura 2.8. Esta configuração apresenta vantagens ao nível do volume de trabalho, permite que seja montado numa posição superior, libertando espaço no chão. Os sistemas de controlo destes robôs são simples.

Figura 2.8: Robô cartesiano

A configuração Cilíndrica, contém dois eixos com movimento linear e um com movimento rotacional (LLR). O robô com esta configuração descreve um volume de trabalho correspondente a um cilíndro, este manipulador, apresentado na Figura 2.9, permite aceder horizontalmente às máquinas. A sua estrutura vertical poupa espaço e concede boa rigidez, elevada capacidade de carga, sendo este robô aconselhado para trabalhos repetitivos.

O robô esférico, também conhecido com polar, possui um volume de trabalho na forma de uma esfera. Pode ser montado numa máquina para carregar/descarregar peças. Esta configuração deixou de ser utilizada devido ao aparecimento de robôs com 4 e 6 eixos. Contudo este apresenta vantagens pois permite a rotação 360 graus e tem um ótimo alcance horizontal. Na Figura 2.10 é apresentado o Unimate, um robô com configuração esférica, que foi o primeiro robô a ser

Figura 2.9: Robô cilíndrico

instalado numa linha de produção.

Figura 2.10: Robô esférico

Por último, o robô mais utilizado na indústria nos dias de hoje é o articulado (RRR) que apresenta três juntas rotacionais. Este robô está disponível em duas variantes. Uma vertical, designado de braço robótico, e outra horizontal, chamado de *Selective Compliant Articulated Robotic Arm* (SCARA). Este manipulador apresenta vantagens pois ocupa pouco espaço, tem um ótimo alcance para o seu tamanho, e permite que o atuador final tenha um elevado alcance e se consiga desviar de obstáculos. Na Figura 2.11 pode-se observar as duas vertentes desta configuração.

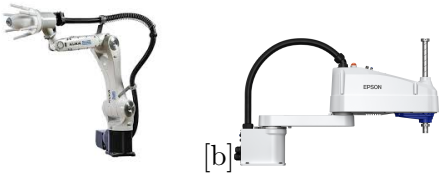


Figura 2.11: Robô articulado nas duas vertentes

2.1.2.2 Unidade de Potência

A unidade de potência é a responsável pelo fornecimento da potência necessária à movimentação dos atuadores. A bomba hidraúlica, o compressor, e a fonte

elétrica são os principais componentes de uma unidade de potência.

2.1.2.3 Controlador

O controlador, apresentado na Figura 2.12 faz a gestão e monitorização dos parâmetros operacionais necessários para a realização das tarefas do robô. Os comandos de movimento enviados através deste dispositivo para os atuadores, são originados neste módulo de processamento, baseando-se nas informações obtidas pelos sensores.

Figura 2.12: Controlador IRC 5 da ABB

2.1.2.4 Sensores

Por último, os sensores são dispositivos usados para recolher e proporcionar informações ao controlador sobre o estado do manipulador e do ambiente. Estes podem ser internos ou externos. Sâo internos quando fornecem informação sobre o estado do manipulador, isto é dados de posição, velocidade, aceleração, entre outros. Por outro lado, os sensores externos fornecem detalhes acerca do ambiente, estes podem ser sensores de força, de movimento ou até câmaras de video para a deteção de obstáculos.

2.1.2.5 Atuador Final

O atuador final é um dispositivo que é acoplado à extremidade do robô, punho, que permite que este execute uma determinada tarefa. Se a tarefa a realizar for o manuseamento de um objeto, o atuador mais indicado será a garra. As garras não são apenas dispostivos mecânicos, hoje em dia existem garras das mais avançadas tecnologias que permitem manusear qualquer tipo de objeto. As garras podem ser mecânicas, magnéticas, de sucção, adesivas, de agulhas, entre outras.

Se a a tarefa a realizar pelo robô for uma operação em um determinado objeto, o atuador mais indicado será uma ferramenta especifica para a tarefa a realizar. Essas ferramentas poderão ser rotativas, tochas de soldadura por arco elétrico, pistolas de pintura, unidades coaxiais inserção, entre outras.

2.2 Programação de Robôs

Nos capítulos iniciais, descreveu-se o aparecimento e a evolução dos robôs até aos dias de hoje. Na verdade estas máquinas são capazes de executar tarefas de uma forma eficiente e eficaz, contudo é necessário haver uma boa programação do robô. A programação de um robô é muito importante e tem que se realizar com muita precaução pois um robô mal programado, em vez de se apresentar como uma solução, pode-se transformar num problema. Esta tarefa pode ser realizada através de diversos métodos contudo os mais utilizados são a programação *online*, a programação *offline* ou por simulação e a programação por demonstração.

2.2.1 Programação Online

Esta programação é feita com o auxílio de uma consola, normalmente designada de *Teach Pendant*, representada na Figura 2.13. Este método é o mais popular e de acordo com a *British Automation and Robot Association* mais de 90% dos robôs são programados através desta forma [11]. As consolas de programação têm vindo a evoluir e cada vez mais tornam o processo rápido e fácil. As primeiras consolas eram grandes caixas cinzentas, que utilizavam para armazenamento fita magnética, hoje em dia as *Teach Pendants* são semelhantes a um *tablet* com ecrã tátil e o operador para programar o robô move-o ponto a ponto, usando a consola, e armazena a posição do braço até que este crie a rota pretendida. Depois de completa a programação, o operador poderá correr o programa utilizando uma velocidade maior [11].

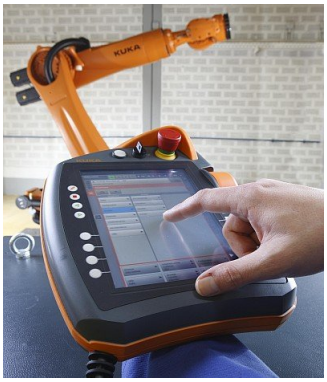


Figura 2.13: Consola de programação de um robô

Este método é bastante utilizado pois atualmente mesmo os robôs mais tradicionais já possuem uma consola de programação, é relativamente fácil e preciso pois esta técnica utiliza coordenadas numéricas, tornando esta programação eficiente quando é necessário executar movimentos simples. Contudo, durante todo o processo de programação o robô estará inoperacional não podendo ser utilizado para outra tarefa, todo o processo requer algum treino e aprendizagem da consola de programação e é um método difícil para quem não está familiarizado com tarefas de programação [11].

2.2.2 Programação Offline

A programação *Offline* ou por simulação é utilizada para assegurar que os algoritmos de controlo do movimento robô são os corretos, antes de executar o programa no robô. Este método é uma mais valia quando o mesmo robô é usado em tarefas distintas. Esta técnica permite que o robô a ser programado esteja no seu funcionamento normal pois a programação deste é feita através de um *software* de simulação. As vantagens deste tipo de programação é o reduzido tempo de inoperacionalidade do robô, é um método intuitivo e permite realizar diversos testes em ambientes e condições diferentes sem ter que as implementar fisicamente. Contudo os ambientes virtuais, exemplo apresentado na Figura 2.14 não são capazes de representar identicamente o mundo real e a curva de aprendizagem de alguns *softwares* de programação é lenta [11].

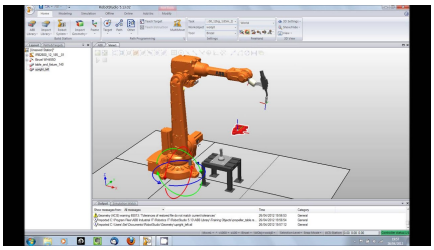


Figura 2.14: *Software* utilizado para programação de robôs

2.2.3 Programação por demonstração

Programar através de demonstração, Figura 2.15, é um método muito intuitivo que permite ensinar o robô guardando as posições pretendidas através do manuseamento do braço envolvendo um sensor de força ou um *joystick* acoplado no pulso do robô. Esta técnica é a mais intuitiva dos métodos anteriormente apresentados apresentando uma grande eficiência em tarefas mais detalhadas e precisas como é o caso da soldadura ou da pintura. Porém, para a programação este método requer que o robô esteja livre, aumentando assim o tempo de inoperacionalidade.

Este tipo de programação não é ideal quando é necessário programar através de coordenadas específicas [11].

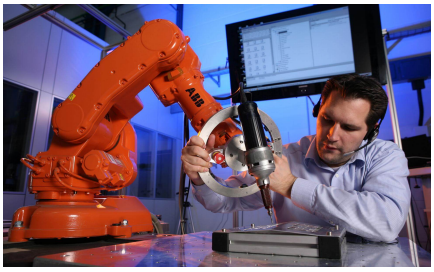


Figura 2.15: Programação por demonstração

2.3 Robótica na arte

A arte, de acordo com a Real Academia Espanhola (RAE), é a ”manifestação da atividade humana por meio da qual o real é interpretado ou o imaginado é representado com recursos físicos, linguísticos ou sonoros”[12]. Até à pouco tempo acreditava-se que a arte era uma capacidade inerente ao ser humano. Contudo, com o aparecimentos da inteligência artificial, os robôs obrigaram os cientistas, empresários, artistas a repensar se essa técnica continuava no património exclusivo da humanidade. Desde muito cedo que a arte e a tecnologia mostraram ser duas áreas com bastante simbiose. Estas envolvem-se em vários domínios como desenho e pintura, dança, música, teatro e cinema. Em todas estas áreas os artistas e investigadores patrocinaram diversas explorações onde as máquinas inteligentes provaram trazer imensos benefícios em projetos artísticos.

O uso da robótica nestas áreas torna-se cada vez mais imprescindível, aumentando a qualidade dos projetos e facilitando muitas tarefas, que no passado eram feitas pelo ser humano. No cinema e no teatro o uso da câmara robótica é bastante utilizado, para tal é acoplada uma câmara ao punho do robô permitindo que este execute a filmagem das cenas. Em título de exemplo, o filme ”Gravity”, realizado por Cuarón, que conta com a participação da atriz Sandra Bullock, muito elogiado e premiado pelas cenas que o compõem é um caso onde o uso da robótica permite a criação de filmagens que no passado eram muito mais complexas de as realizar [13].

A pintura, é uma das artes mais antigas, é uma forma de o artista expressar ideias e emoções, numa linguagem visual a duas ou mais dimensões. Os elementos desta linguagem podem ser formas, linhas, cores, tonalidade e texturas.

2.3.1 Aplicações da robótica na pintura

Tal como as restantes formas de arte, a pintura tem vindo a mostrar ser uma área onde o uso da robótica, aliada a tecnologias como Inteligência Artificial, pode trazer grandes avanços na forma de se desenvolver ou interpretar esta forma de arte.

O pioneiro na arte de ensinar robôs a pintar é o artista Harold Cohen, que desde 1973 tem vindo a desenvolver o sistema AARON, apresentado na Figura 2.16. O AARON é o primeiro robô capaz de desenvolver pinturas originais, aleatórias com base em elementos conhecidos pelo sistema. Assim a pintura e os elementos a pintar foram programados pelo artista, o robô com base nessa informação decide qual o tema a pintar, quais as tonalidades a usar e qual a composição do seu desenho [14].

Figura 2.16: Sistema AARON e o seu criador Harold Cohen

Porém, desde cedo que o ser humano apresenta interesse em replicar num robô as suas características, atribuindo-lhes capacidades e funcionalidades que permitam ao humanóide executar tarefas que até então só alguns seres humanos com vocação e talento eram capazes de as executar.

Foi com essa ambição que apareceu o AI-DA, concluído em 2019, é o primeiro robô humanóide artista, desenvolvido pela universidade de Oxford, capaz de desenhar, pintar ou criar esculturas, baseando-se apenas num *input* fornecido pelo utilizador. Este sistema é composto por um braço, semelhante ao braço humano, e uma câmara que o robô utiliza para analisar imagens, paisagens ou até pessoas e objetos, que alimenta o algoritmo, por forma a movimentar o braço fazendo com que o humanóide seja capaz de realizar desenhos ou pinturas. Toda a estrutura apresenta características físicas inerentes ao ser humano [15].

Todas as obras desenvolvidas pelo AI-DA são caracterizadas pela sua criatividade. Na figura seguinte, observa-se o robô a expressar, através de uma pintura, a sua concepção de uma árvore [3].

Figura 2.17: Robô AI-DA e uma pintura de uma árvore [3]

Assim o AI-DA não é apresentada como substituição dos artistas, mas sim como uma ferramenta que permite aos artistas conhecer novas formas de criatividade nunca antes exploradas.

Numa vertente menos criativa e mais realista, existe o E-David que é um manipulador industrial, tendo este um pincel acoplado ao seu punho e uma câmara. O sistema é totalmente independente pois captura imagens, do modelo a copiar, à medida que está a desenvolver a obra e consoante o algoritmo decide em tempo real onde irá pintar. Todo o sistema é composto por cinco pincéis diferentes e vinte e quatro cores diferentes. É o sistema que consoante o *input* (imagens capturadas) escolhe o pincel, a cor e se necessário executa a limpeza da ferramenta de pintura [4].

Este robô é capaz de desenvolver retratos ou réplicas perfeitas, sem que o programador tenha que intervir em todo o processo. Na Figura 2.18 estão algumas obras desenvolvidas pelo E-David [4].

Num lado mais comercial e interativo, surge o Cloud Painter, criado pelo artista Pindar Van Arman, que consiste numa plataforma *online* onde o artista promove todas as obras de arte produzidas pela sua equipa de robôs. Todos os robôs são elaborados e programados pelo artista. Este desenvolve todo o *software* baseando-se em redes neuronais artificiais. Deste modo, cada obra de arte é única e cada pessoa pode adquiri-la *online*. Na Figura 2.19 surge o artista e criador da plataforma com um dos seus robôs a pintar uma obra [5].

2.4 Sistemas Computer Aided Design (CAD)

A sigla CAD surge do inglês, *Computer Aided Design*, e entende-se como o uso do computador para a criação, modificação, análise, ou otimização de um determinado modelo. Com o auxílio de uma aplicação CAD é possível construir um modelo, num espaço imaginário, podendo depois visualizar as propriedades do

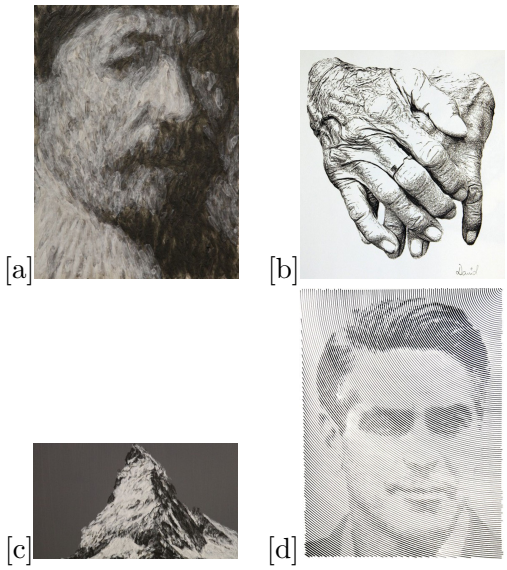


Figura 2.18: Quatro exemplos de obras desenvolvidas pelo E-david [4]

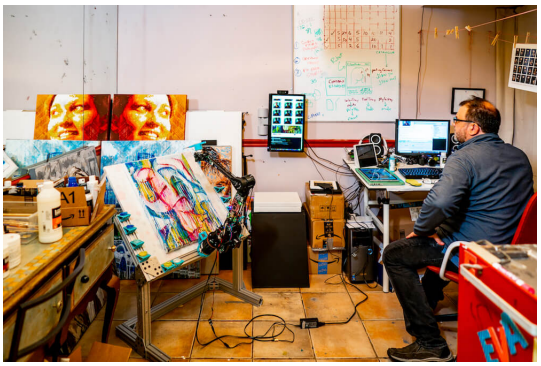


Figura 2.19: Pindar Van Arman com um dos seus robôs [5]

objeto, como peso, dimensões geométricas, material ou cor, antes do modelo ser implementado para uma aplicação em particular.

Foi em 1962 que Ivan Sutherland criou o primeiro programa de computação gráfica, Figura 2.20, denominado de "SketchPad", que permitia ao utilizador escrever ou desenhar figuras simples num ecrã com o auxílio de uma caneta para o efeito. Este desenvolvimento marcou o início do futuro de todas as aplicações CAD [16]. Inicialmente, estas ferramentas eram unicamente utilizadas para investigações, contudo na década de setenta as grandes organizações pertencentes aos sectores automóvel e aeroespacial começaram a desenvolver as suas próprias aplicações CAD, expandindo-as para outras indústrias em meados de 1980. Antes da década de noventa, *softwares* como CATIA e AutoCAD surgiram, possibilitando o uso deste tipo de aplicações noutras indústrias [17].

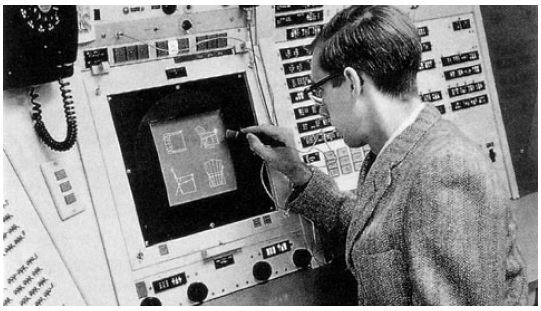


Figura 2.20: Ivan Sutherland criador da primeira aplicação de computação gráfica

Atualmente estas aplicações são indispensáveis em diversas áreas, como arquitetura, *design* do produto e gráfico, engenharia, entre outras. Hoje em dia existem aplicações CAD para diversas plataformas, incluindo Microsoft Windows, Linux, Unix e Mac OS X. Nas últimas décadas houve um grande desenvolvimento de ferramentas CAD. Os *softwares* mais utilizados são o AutoCAD, o SolidWorks, o CATIA e o Pro/ENGINEER [18]. A arquitetura, apresentada na Figura 2.21, de um sistema CAD é genérica para as aplicações CAD abordadas acima. Um sistema CAD apresenta inúmeras vantagens no desenvolvimento de um determinado modelo, redução dos custos no desenvolvimento do produto, aumento da produtividade, melhoria na qualidade do produto, melhor visualização do modelo final e redução dos custos associados ao desenvolvimento do protótipo físico. Contudo também apresenta desvantagens, o preço para a aquisição de um *software* CAD é elevado, o custo de um *hardware* compatível com esta aplicação é elevado e por fim o custo de formação para os futuros utilizadores também representa uma grande despesa para a organização.

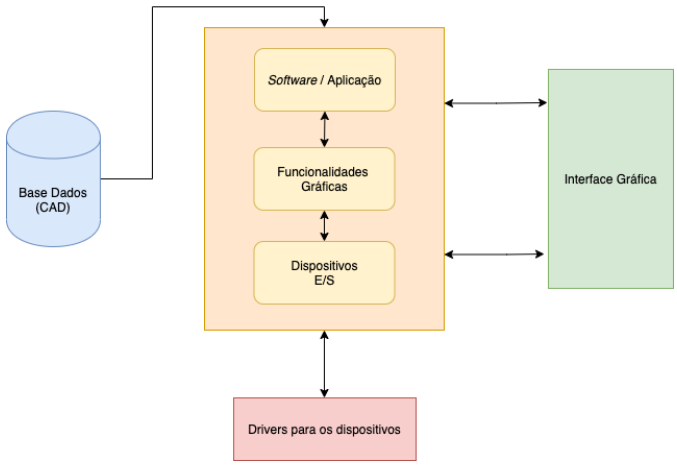


Figura 2.21: Arquitetura de um sistema CAD

Nos dias de hoje, as organizações recorrem a estas aplicações frequentemente

na sua atividade para desenvolver produtos ou para modificar/otimizar sistemas já existentes. Quando esse modelo necessita de ser partilhado com outro departamento ou outra organização é necessário recorrer à distribuição do ficheiro CAD. Um ficheiro CAD, contém a informação detalhada de todas as etapas, desde o *design* à produção de um determinado modelo. Assim, é importante ter em atenção que não haja perda de informação do modelo durante a sua partilha. Para tal é necessário haver uma conversão para um formato de ficheiro conhecido e interpretado por todas as aplicações CAD no mercado. Atualmente, esta conversão pode ser feita de duas formas distintas, através de conversores dedicados ou através de formatos *standard*. Estes conversores dedicados são muito dispendiosos e requerem aquisições de licenças. Assim, o uso de formatos neutros/*standard* ou formatos nativos são muito utilizados quando é necessário haver partilha de um ficheiro com conteúdo CAD. Na Figura 2.22 apresenta-se como exemplo a partilha do ficheiro CAD pelos diversos departamentos constituintes de uma organização.

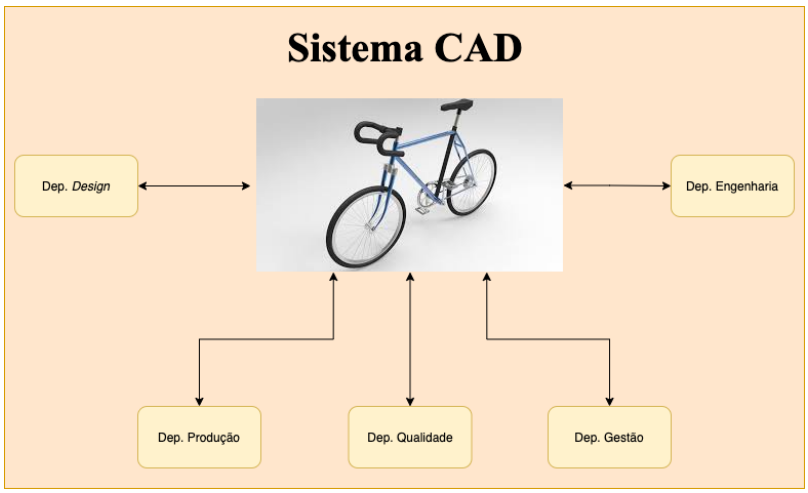


Figura 2.22: Sistema CAD numa fábrica de bicicletas

2.4.1 Formatos Neutros e Nativos

Um formato CAD neutro, é um tipo de ficheiro que armazena toda a informação do modelo CAD e pode ser usado em qualquer aplicação de desenho gráfico, pois é gerido por entidades externas que normalizam e regem regras de estrutura para estes ficheiros. Exemplos de formatos neutros são o *The Initial Graphics Exchange Specification* (IGES), o *Stereolithography* (STL), o *Standard for the Exchange of Product Data* (STEP), o Parasolid, entre outros. Por outro lado, um formato nativo é desenvolvido por uma organização que possui aplicações CAD visando que este tipo de ficheiro só apresente compatibilidade nos *softwares* geridos pela organização em questão.

O primeiro formato neutro de distribuição de ficheiros CAD foi o IGES, foi criado em 1980 e a última atualização foi a 1996, este formato é gerido pela *American National Standards Institute* (ANSI). Este formato é usado unicamente para partilhar informações relativas às superfícies de um modelo, uma vez que este tipo de ficheiro não suporta sólidos geométricos.

O exemplo mais popular de um ficheiro nativo é o *Drawing Exchange Format* (DXF), criado pela AutoDesk, proprietária de diversas aplicações CAD como AutoCAD, Inventor, Fusion 360, entre outros. Este tipo de ficheiro foi apresentado em 1982, com o intuito de promover uma boa partilha de conteúdo CAD entre as aplicações geridas pela empresa.

Hoje em dia, o ficheiro neutro mais utilizado e mais aceite para a partilha de conteúdo CAD é o STEP. Todas as aplicações CAD têm a capacidade de importar/exportar ficheiros STEP, permitindo que haja boa interação entre aplicações pertencentes ao ecossistema CAD. Este formato é gerido pela *International Organization for Standardization* (ISO) tendo sido criado em 1994 e tendo vindo a sofrer atualizações até aos dias de hoje [19].

2.5 Conclusão do Capítulo 2

Desde o aparecimento da máquina que o ser humano começou a explorar novos mundos e a desenvolver tecnologias que eram impensáveis até à data. O robô é o melhor exemplo de como a sabedoria humana e a tecnologia são capazes de desenvolver ferramentas que permitem à indústria responder à altura ao mercado atual, conseguindo níveis de produtividade e de qualidade de produto que superam todas as expectativas do cliente. Contudo, esta máquina munida com tecnologias como inteligência artificial é capaz de trazer ao mundo da arte novos ideais e perspetivas totalmente diferentes do já conhecido. A pintura é um exemplo, ainda explorado no dia de hoje, que relata a forma como um robô é capaz de expressar o que entende de um determinado objeto ou paisagem, permitindo ao artista explorar novas perspetivas.

Capítulo 3

Arquitetura Proposta para a Solução

Neste capítulo é abordado o contexto do problema e são especificados todos os requisitos e cenários para a sua resolução. Por fim é apresentada a arquitetura da solução encontrada para o problema em questão.

3.1 Problema

Entende-se que a área onde se utiliza mais a robótica é na indústria, contudo tem-se vindo a explorar o uso destas máquinas noutras áreas, onde o seu uso pode ser apresentado como uma mais valia. Na pintura industrial a utilização dos manipuladores é uma solução, ao invés da pintura manual, aumentando a poupança da matéria prima, a tinta, entre 15% a 30% [20].

Porém, estes sistemas podem ser adaptados a outro tipo de atividades, como por exemplo a pintura a pincel de peças cerâmicas ou personalizada de objetos, entre outras.

Assim o objetivo deste trabalho consiste no desenvolvimento de um sistema, apresentado na Figura 3.1, capaz de efetuar pintura a pincel de um objeto a partir de desenhos criados por um *software* CAD. Em suma, é obrigatório desenvolver um programa que converta o desenho CAD em linguagem própria do robô, instruindo este para executar a pintura do objeto em questão.

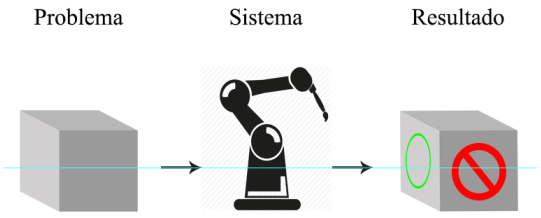


Figura 3.1: Explicação do problema

3.2 Solução Encontrada

Com vista a resolver o problema apresentado na secção acima, desenvolveu-se um sistema, como pode ser visualizado na Figura 3.2, constituído por um robô capaz de efetuar a pintura, desenhos importados a partir de um *software* CAD e também um programa com a função de converter o desenho em linguagem do robô para que este seja capaz de realizar a pintura do objeto pretendido.

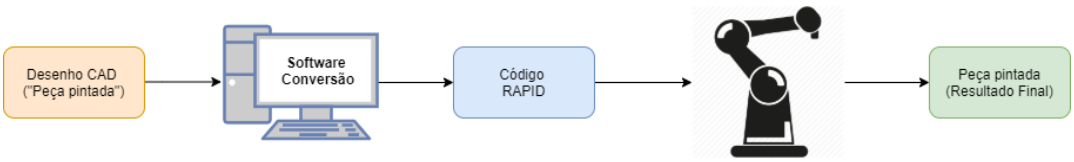


Figura 3.2: Sistema desenvolvido para a resolução do problema

Para a elaboração deste sistema foram necessários os componentes:

- *software* CAD;
- programa de conversão;
- robô;
- ferramenta de trabalho do robô;
- peça a pintar.

3.2.1 Software CAD

O programa CAD utilizado para o presente trabalho foi o AutoCAD 2021. A razão para a preferência deste é o facto de o formato de intercâmbio DXF e o *software* pertencerem à AutoDesk, reduzindo assim o risco de incompatibilidades. A principal razão da escolha do formato DXF foi o facto de este tipo de ficheiro apresentar uma vasta documentação técnica, garantindo assim um excelente suporte.

3.2.2 Programa de Conversão

Por forma a extrair a informação dos desenhos CAD e gerar um ficheiro RAPID (linguagem entendida pelo robô), foi desenvolvido um programa com uma interface de fácil interação para o utilizador que permite ao utilizador escolher o ficheiro DXF para que de seguida seja gerado automaticamente o ficheiro pretendido com as instruções do robô. Para a elaboração do programa foi utilizada a linguagem de programação Python, por ser uma linguagem muito popular hoje em dia e por possuir muito suporte. O *Integrated Development Enviornment* (IDE) foi o PyCharm.

3.2.3 Robô

O robô utilizado neste projeto foi o IRB 120 e o controlador IRC5 da ABB, Figura 3.3. Este é caracterizado por ser um manipulador industrial, com capacidade de carga de 3 kg e com seis eixos de movimento independentes entre si. Este manipulador pode ser montado em qualquer ângulo obtendo um alcance de até 580 mm.

Figura 3.3: Robô IRB 120 e controlador IRC5 Compact da ABB

3.2.4 Ferramenta de trabalho do robô

Para que a pintura manual do objeto seja efetuada, a ferramenta que mais se adequa a tal tarefa é o pincel convencional.

3.2.5 Peça

A peça utilizada, para efeitos de testes, foi um cubo com as dimensões 100x100x100 mm. Contudo poderá ser qualquer objeto que o utilizador do sistema pretenda

aplicar-lhe uma pintura artística.

3.3 Arquitetura do Sistema

A arquitetura do sistema, Figura 3.4, assenta em três partes essenciais: ficheiro DXF, programa de conversão e ficheiro RAPID. Primeiramente, é necessário criar ou importar a peça/objeto que se pretende aplicar uma pintura, para que posteriormente com a ajuda da ferramenta de CAD seja possível desenhar os conteúdos pretendidos nas faces da mesma. De seguida, já com o desenho exportado no tipo DXF, procede-se à conversão para uma linguagem que seja entendida pelo robô. Por fim, através de teste e simulações computacionais valida-se o programa RAPID gerado para posteriormente testar em ambiente de simulação.

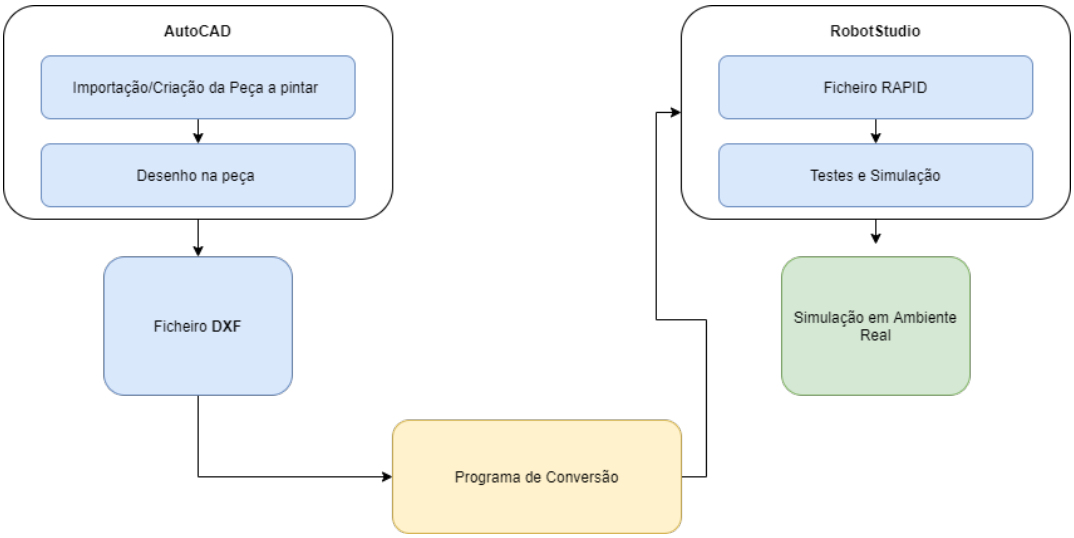


Figura 3.4: Arquitetura da solução

3.3.1 Ficheiro DXF

Um tipo de ficheiro DXF é utilizado para o intercâmbio de informação CAD entre os diversos programas de desenho. Estes ficheiros são codificados em ASCII e estão divididos em quatro secções: cabeçalho (*HEADER*), tabelas (*TABLES*), blocos (*BLOCKS*) e entidades (*ENTITIES*). A secção *HEADER* apresenta informação geral acerca do desenho, como tolerâncias, padrões, detalhes dos *layers*, entre outros. Contudo como este tipo de ficheiro tem como principal objetivo a transferência de informações geométricas entre programas, sendo assim esta secção não é crucial para o problema em questão. De seguida, encontrar-se-á as tabelas, onde são especificadas algumas constantes que podem conter informações não geométricas importantes do desenho, como por exemplo estilos de linhas e

texto, os *layers*, tabelas de visualização ou até detalhes acerca do sistema de coordenadas do utilizador. Porém todas as informações apresentadas nas *TABLES* não são essenciais para a definição geométrica do desenho. A secção *BLOCKS* contém a definição das entidades *block* descrevendo as entidades que as compõem. Por fim, é apresentada a secção mais importante para a extração de informação geométrica do desenho. É nas entidades que são exibidas todas as características do desenho. Todas as geometrias contidas no desenho são expostas nesta secção, como, por exemplo, linhas, pontos, círculos, entre outros [21].

Os dados geométricos serão apresentados de forma diferente nesta secção consoante os desenhos contidos no modelo. No caso da Figura 3.5, observa-se que no modelo, neste caso um cubo, são desenhadas duas entidades, uma linha e um círculo. Assim no ficheiro DXF a secção *ENTITIES* só irá apresentar informações geométricas destes dois objetos.

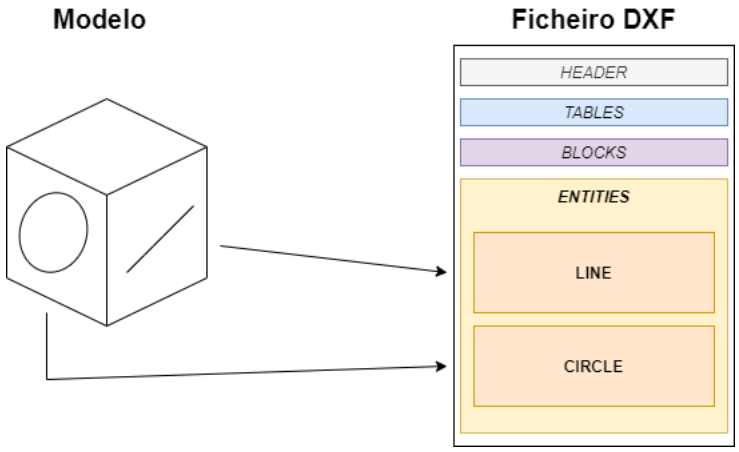


Figura 3.5: Estrutura de um ficheiro do tipo DXF

As entidades apresentam diferentes formas de serem definidas. Tendo como exemplo as entidades linha, círculo e ponto, nas figuras apresentadas a seguir fica explícito a forma de definição das geometrias nestas três entidades.

Na Figura 3.6, observa-se que a linha é definida tendo em conta dois pontos. Assim os parâmetros 10, 20, 30, 11, 21, 31 são responsáveis pela atribuição de coordenadas ao ponto A e ao ponto B.

A geometria seguinte, Figura 3.7, o círculo necessita de um ponto, que neste caso é o centro de circunferência, e do raio para a definição da sua entidade.

Os pontos são caracterizados apenas pelas suas coordenadas, como se pode comprovar pela Figura 3.8.

Deste modo, além das geometrias, a secção *ENTITIES* do ficheiro DXF armazena a cor aplicada no desenho para determinada entidade. Assim, as cores

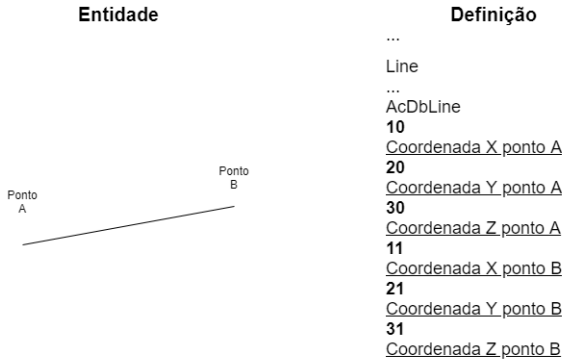


Figura 3.6: Definição de uma linha

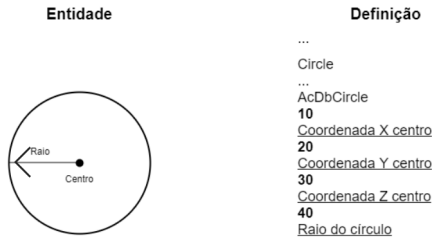


Figura 3.7: Definição de um círculo

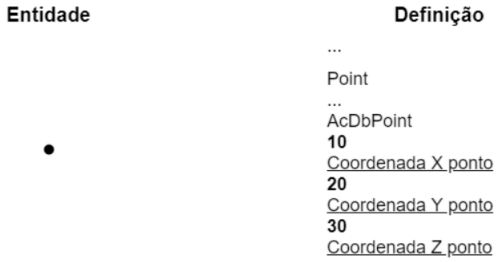


Figura 3.8: Definição de um ponto

são definidas no ficheiro DXF baseando-se no sistema de cores aditivas *Red Blue Green* (RGB), sendo a cor uma combinação destas três componentes com uma escala de 0 a 255. Nos *softwares* CAD são atribuídos números a cada cor, como se ilustra na Tabela 3.1.

Estes códigos de cor estão dispostos para cada entidade dentro da subclasse 'AcDbEntity' após a linha de identificação de cor que é caracterizada pelo número '62'. Na Figura 3.9 observa-se um exemplo, onde a seguir ao código de identificação de cor surge a cor, neste caso azul (código DXF = 5) da entidade em questão (linha).

Tabela 3.1: Código de cores utilizados pelos *softwares* CAD

Cor	Código DXF
Vermelho	1
Amarelo	2
Verde	3
Ciano	4
Azul	5
Magenta	6
Branco	7

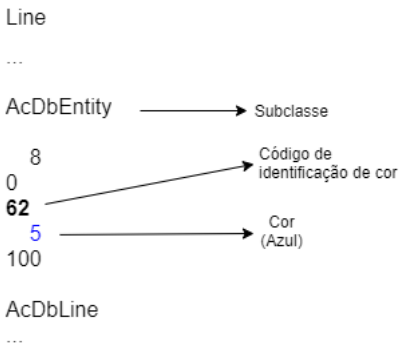


Figura 3.9: Identificação da cor na secção *Entities*

3.3.2 Linguagem RAPID para programação de Robôs

A linguagem de programação RAPID foi desenvolvida pela ABB em 1994 e veio substituir a linguagem ARLA, que era utilizada, até à data, para a programação dos robôs da marca. Esta linguagem é de alto nível pelo que através dela é possível, com alguma facilidade, controlar o fluxo do programa a ser desenvolvido para o robô. É possível também a partir de alguns parâmetros específicos controlar movimentos, velocidades do robô em determinados pontos e até controlar saídas e entradas aumentado assim a integração da máquina numa célula por exemplo.

Um programa RAPID está organizado por três tipos diferentes de rotinas:

- *Procedures*: são chamados também de subprogramas, isto é são pequenos programas que são utilizados dentro do programa principal.
- *Functions*: funções que retornam um valor de um determinado tipo que será usado como argumento de uma instrução.
- *Trap Routines*: servem para detetar anomalias no programa e podem também ser associadas a interrupções.

Estas rotinas são compostas por instruções que podem ser de vários tipos. As mais comuns são instruções para o controlo do fluxo do programa, para o controlo do movimento do robô e para a manipulação de entradas/saídas do robô.

A informação ao longo de todo o programa é armazenada na forma de variáveis, que podem ser globais ou locais. Variáveis globais é quando o dado em questão pode ser acessível em qualquer rotina do programa. Pelo contrário, as variáveis locais só são acedidas nas rotinas onde foram declaradas. Os dados podem assumir três tipos diferentes:

- constantes: dados que são fixos e só poderão ser alterados modificando o código fonte do programa.
- variáveis: dados que podem ser alterados no decorrer do código.
- *persistent*: como o próprio nome indica são variáveis 'persistentes', assim esta variável assume sempre o último valor adquirido, mesmo que ocorra reinicialização do programa.

Um programa RAPID é dividido em módulos, rotinas e variáveis. Um módulo poderá conter inúmeras rotinas e nessas poderão existir muitas variáveis. Executar o programa RAPID significa correr o procedimento principal (*main*) que por sua vez poderá aceder a rotinas dentro do mesmo módulo ou de outros módulos. A Figura 3.10 ajudará a perceber a estrutura da programação RAPID.

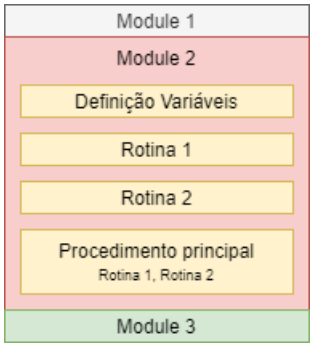


Figura 3.10: Ilustração da estrutura de um programa RAPID

O principal objetivo de um programa RAPID é o controlo do movimento da ferramenta de um robô para a execução de uma determinada tarefa, seja ela soldadura, pintura, movimentação de objetos, paletização, entre outras. Na movimentação de um robô utiliza-se *targets* e *paths*. O *target* é um ponto no espaço, definido pelas suas coordenadas, que o robô deve alcançar. O *path* é uma sequência de instruções de movimento, usados para fazer com que o robô se mova ao longo de uma série de *targets*.

Os *targets* contêm informação acerca das coordenadas, da orientação e da configuração dos eixos do robô para que este seja capaz de atingir determinada posição. Estes dados são gravados numa variável do tipo constante, Figura 3.11.

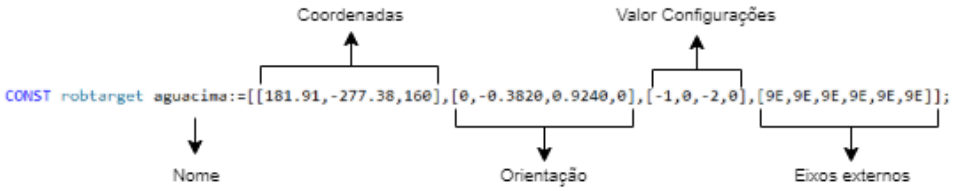


Figura 3.11: Criação de um *target*

Os *paths* são definidos através de um conjunto de instruções de movimento, instruções estas que têm como objetivo controlar o robô para que este seja capaz de atingir um determinado ponto (*target*). Para tal, existem três tipos principais de instruções que podem ser utilizadas para esse efeito, que são denominadas de *MoveL*, *MoveJ* e *MoveC*.

A instrução *MoveL*, Figura 3.12 obriga o robô a mover-se de forma linear entre dois pontos.

```
MoveL p10, v1000, fine, tool0;
```

Figura 3.12: Instrução *MoveL*

- **p10**: especifica a posição (*target*) para onde o robô se deverá movimentar.
- **v1000**: velocidade do manipulador para aquela instrução, neste caso 1000 mm/s.
- **fine**: indica qual o raio de desvio tolerado para diminuir a distância percorrida pelo robô entre dois ou mais pontos. Na instrução apresentada utilizou-se o *fine* que garante que o robô ao se deslocar para o ponto pretendido não encurta caminho.
- **tool0**: determina qual a ferramenta acoplada na ponta do robô que irá atingir um ponto específico.

Caso se pretenda movimentar o robô para um determinado *target* de uma forma rápida sem que essa trajetória seja em linha reta, poderá recorrer-se à instrução *MoveJ*. A sintaxe desta instrução é análoga à instrução abordada anteriormente.

Por último, se a trajetória pretendida for circular, Figura 3.13, poderá utilizar-se a instrução *MoveC*, Figura 3.14.

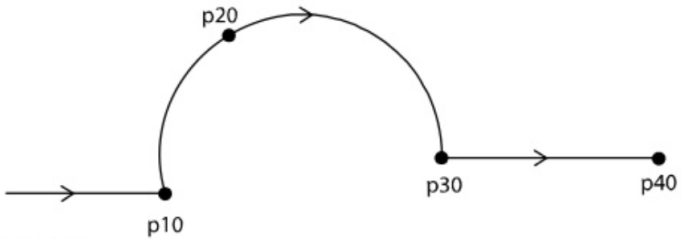


Figura 3.13: Trajetória circular

```
MoveL p10, v1000, fine, tool0;
MoveC p20,p30, v500, fine, tool0;
MoveL p40, v500, fine, tool0;
```

Figura 3.14: Instruções para descrição da trajetória pretendida

3.3.3 Robot Studio

No uso da robótica as simulações são indispensáveis contribuindo assim para a diminuição de erros e também para redução do tempo de programação. Desenvolvido pela ABB, o *Robot Studio* apresenta-se como um programa de simulação de células robóticas onde é possível executar os programas e testar diversos casos antes de exportar o programa para o ambiente real.

Esta aplicação possui diversas ferramentas que permitem ao utilizador criar uma réplica da célula podendo posteriormente simular e criar o próprio programa de uma forma extremamente intuitiva. O *Robot Studio* também disponibiliza uma bilbioteca de *add-ins* tornando assim a simulação mais real e colocando em teste as operações que o robô irá realizar em ambiente real, como por exemplo *add-ins* de pintura, de soldadura, de paletização, entre outros.

A interface do utilizador neste *software* é muito intuitiva e a aplicação encontra-se muito bem dividida consoante as etapas de programação de uma célula robótica. Na Figura 3.15 está apresentado o ambiente de trabalho do *Robot Studio*.

Para o desenvolvimento da célula, este programa possui bibliotecas com alguns modelos de objetos previamente elaborados. Contudo a importação de modelos CAD para esta aplicação é possível, tornando assim a célula mais real. Por fim, com a ajuda das ferramentas disponibilizadas por esta aplicação é possível realizar testes e simulações verificando assim se o programa está pronto para ser testado em ambiente real.

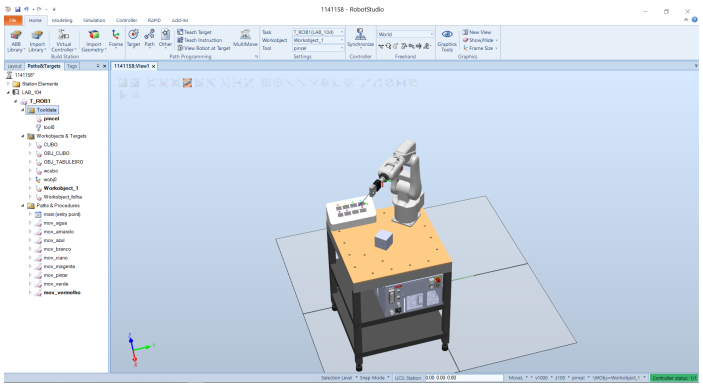


Figura 3.15: Ambiente de trabalho do Robot Studio

3.3.4 Python

O *software* desenvolvido no âmbito desta dissertação foi programada em linguagem Python. O Python é uma linguagem de programação de alto-nível orientada por objetos, com uma semântica bastante simples e intuitiva. Esta linguagem suporta módulos e pacotes que levam as aplicações a serem bastante modulares e com grande capacidade de reutilização.

O Python apresenta um enorme crescimento em muito pouco tempo comparativamente com outras linguagem de programação bastante utilizadas, como se pode comprovar pelo gráfico na Figura 3.16.

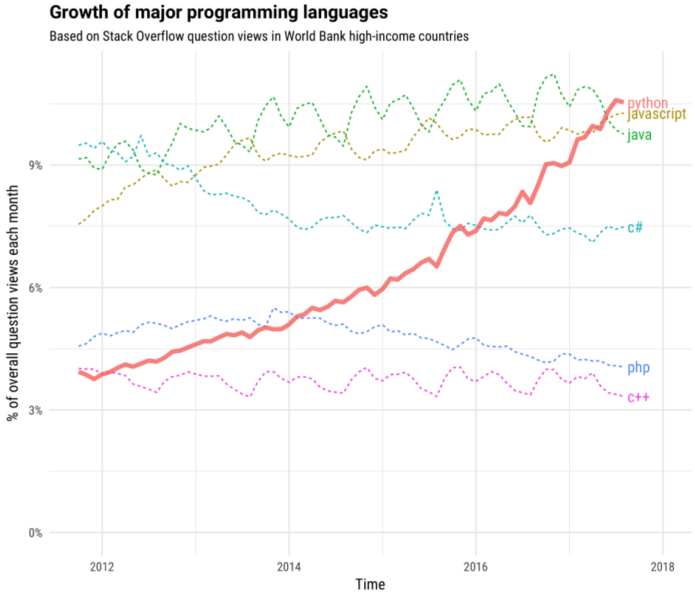


Figura 3.16: Crescimento da linguagem Python [6]

3.3.5 Integrated Development Environment (IDE)

O ambiente de programção utilizado foi o PyCharm que é um IDE focado na linguagem Python. Este possuí inúmeros módulos e pacotes que permitem a verificação de erros no código ou correção da sintaxe. Este ambiente é bastante intuitivo e a versão *Community*, apresentada na Figura 3.17 é totalmente gratuita.

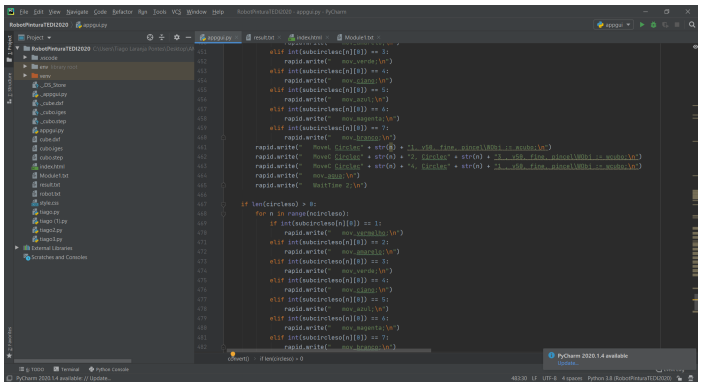


Figura 3.17: Ambiente de trabalho do PyCharm Community

3.4 Conclusão Capítulo 3

Em suma, a solução encontrada para a resolução do problema proposto foi encontrada, contemplando todos os requisitos e cumprindo todos os objetivos. Assim, foram escolhidos todas as ferramentas que irão fazer parte deste sistema, avaliando as principais características de cada uma. Por fim, elaborou-se a arquitetura do sistema que irá suportar toda a implementação da solução.

Capítulo 4

Implementação da Solução

O principal objetivo deste trabalho é o desenvolvimento de um programa capaz de converter um ficheiro DXF num programa RAPID com as instruções de movimento necessárias que permitam ao robô executar a pintura ou desenho importado para a aplicação. Assim, o presente capítulo tem como principal objetivo explicar todo o processo de programação da aplicação de conversão, criação da célula e todos os detalhes presentes nesta implementação.

4.1 Modelização AutoCAD

Para que seja possível a aplicação de um desenho no objecto, é necessário que o utilizador do sistema desenvolva uma pintura, utilizando para tal uma ferramenta CAD. Neste projeto utilizou-se o AutoCAD, sendo o DXF o formato de exportação escolhido. Na Figura 4.1 é apresentado um cubo onde nas suas faces foram desenhados alguns símbolos conhecidos, um sol e o sinal de proibido.

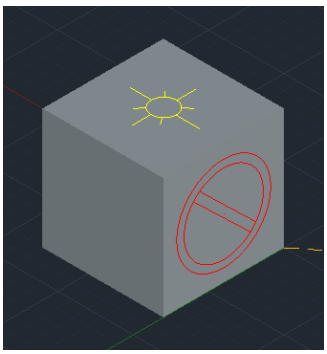


Figura 4.1: Modelização geométrica

4.2 Aplicação de Conversão

O desenvolvimento do programa de conversão foi o principal desafio na elaboração deste projeto.

O programa encontra-se dividido em funções de manipulação do ficheiro DXF, de processamento, de escrita e de *design* da interface gráfica. A aplicação é bastante intuitiva contendo uma interface simples e limpa, tornado o programa de fácil utilização.

4.2.1 Arquitetura de Software

A aplicação desenvolvida divide-se em quatro partes fundamentais.

Em primeiro lugar é necessário importar o ficheiro DXF, ficheiro este que foi anteriormente exportado do *software* CAD. Contudo este ficheiro, como se analisou anteriormente, apresenta secções que não são necessárias para a tarefa em questão. Assim é necessário extrair a informação necessária e guardá-la localmente, posteriormente analisa-se e processa-se toda a informação recolhida para que, no fim, seja possível gerar automaticamente o ficheiro RAPID com as instruções do robô para que este seja capaz de realizar a tarefa de pintura.

4.2.1.1 Leitura e Manipulação do Ficheiro DXF

Antes de mais, ter-se-á que extrair o ficheiro DXF de um qualquer *software* de desenho. O ficheiro DXF contém muita informação que não apresenta relevância para este projeto. Assim, a única secção do ficheiro que se pretende guardar é a *ENTITIES*, para tal, com a ajuda da função *split()* ir-se-á percorrer este ficheiro e guardar a secção pretendida num ficheiro de texto (*result.txt*).

4.2.1.2 Extração de informação

Tal como foi abordado no capítulo anterior, as entidades pretendidas para posteriormente utilizar na pintura do objeto são as linhas e os círculos. Assim, depois de toda a secção *ENTITIES* ter sido armazenada no ficheiro *result.txt*, é necessário recolher os detalhes destas duas entidades armazenando-os num vetor, facilitando assim as etapas de processamento seguintes.

Para tal foram criadas duas funções que retornam um vetor com as informações pretendidas de cada entidade. As funções funcionam de forma iterativa, isto é, para cada linha verifica-se o valor da mesma, decidindo assim se é relevante o seu armazenamento ou não.

A função *line()* extrai todos os detalhes necessários para a definição de uma linha. Para tal é armazenado o valor da sua cor, seguindo-se das coordenadas x, y e z do ponto inicial e por último a posição x, y e z do ponto final da reta.

Por outro lado, a função *circle()* é responsável pela extração e armazenamento de toda a informação necessária para a descrição de um círculo, a sua cor, as coordenadas x, y e z do centro e o raio do mesmo.

Os valores extraídos pelas duas funções são guardados da seguinte forma:

- **Line()**: [*cor linha 1, xinicial , yinicial, zinicial, xfinal, yfinal, zfinal, cor linha 2, ...*]
- **Circle()**: [*cor circulo 1, xcentro, ycentro, zcentro, raio circulo 1, cor circulo 2, ...*]

Em suma após a execução das duas funções pretende-se originar dois vetores, o que contém as linhas e o dos círculos.

O vetor das linhas apresenta outros vetores onde cada um desses contém as informações de cada forma. Na Figura 4.2 é apresentado um esquema que pretende demonstrar como é feita a divisão dos dados.

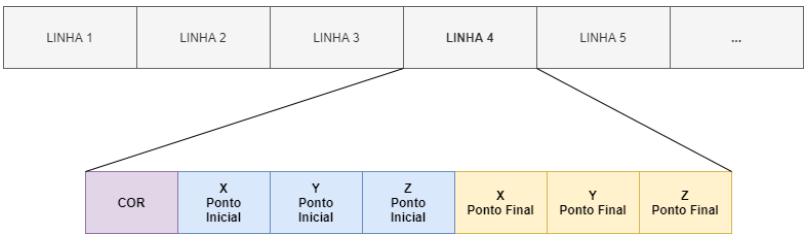


Figura 4.2: Esquema representativo da organização dos dados recolhidos da *ENTITIE* linha

Paralelamente ao vetor que contém as linhas, o vetor que armazena os círculos apresenta uma organização semelhante, como se demonstra na Figura 4.3.

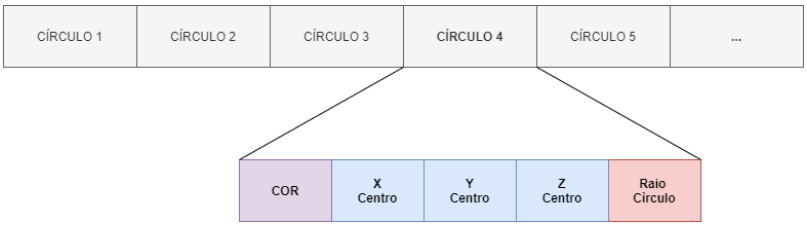


Figura 4.3: Esquema representativo do armazenamento da forma círculo

4.2.1.3 Análise e processamento da informação recolhida

Depois de armazenados os valores relevantes para a definição das duas formas, segue-se a análise e processamento dos dados, que visa simplificar todo o processo posterior de geração do ficheiro com linguagem entendida pelo robô.

Antes da escrita do ficheiro com linguagem RAPID é necessário analisar e processar os dados recolhidos retirando assim algumas informações necessárias.

Primeiramente é necessário definir com que orientação é que o robô irá alcançar os *targets* garantindo assim que o percurso a percorrer por este é feito sem nenhum imprevisto, e que a posição de pintura é a correta. Para tal, assumindo que o modelo, a ser pintado, é um cubo com as dimensões 100 mm x 100 mm x 100 mm, é importante determinar em que face se encontram as coordenadas x,y,z de cada forma recolhida.

Assim, para cada ponto testam-se os valores x,y,z , determinando em que face este se encontra. Nas Figura 4.4 está representado o objeto a pintar em duas perspetivas diferentes, onde estão assinaladas as numerações de cada face.

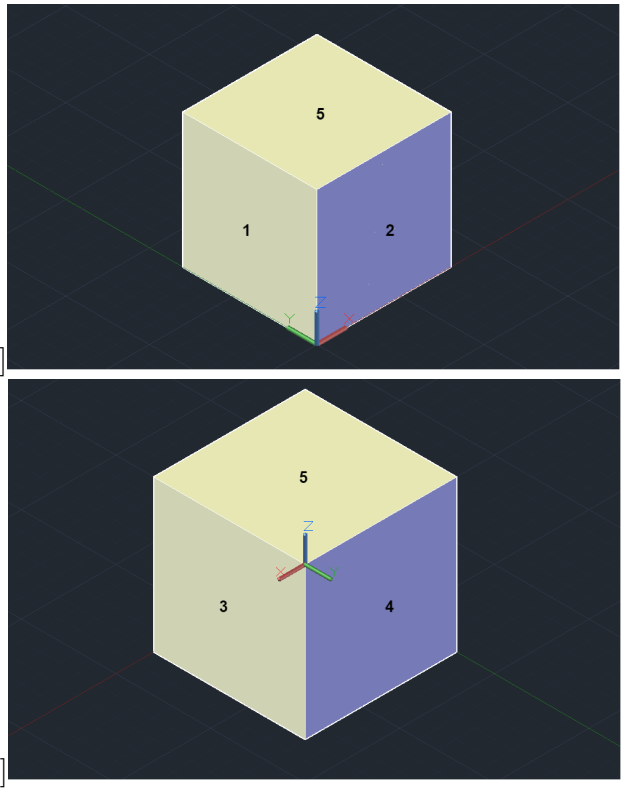


Figura 4.4: Cubo em duas perspetivas

Assumindo que o valor X do ponto é X, o valor Y do ponto é Y e o valor Z do ponto é Z, e que o cubo tem de dimensões 100 mm x 100 mm x 100 mm, as condições testadas que permitem descobrir em que face estão os pontos são:

- **Se X = 0** - Face 1
- **Se Y = 0** - Face 2

- **Se X = 100** - Face 3
- **Se Y = 100** - Face 4
- **Se Z = 100** - Face 5

Contudo, foi necessário descobrir quais as orientações a escolher para cada face. Assim, utilizou-se a ferramenta *free hand* do *Robot Studio* e determinou-se quais as melhores orientações para cada face.

Todas estas verificações foram feitas para ambas as entidades.

Depois das orientações dos *targets* determinados, o próximo passo é a geração do ficheiro RAPID.

4.2.1.4 Geração do ficheiro pretendido

Após o armazenamento das definições das entidades e das orientações determinadas para cada *target* é necessário iniciar a escrita do programa RAPID.

A conteúdo do ficheiro a gerar é dinâmico, isto é, depende do ficheiro DXF importado para a aplicação, contudo existem alguns *targets* e procedimentos que não mudam, como é o caso dos responsáveis pela coloração do desenho.

Primeiramente, é necessário recorrer a funções *In and Out* do Python para criar um ficheiro, que neste caso é designado de *Module1.mod*, extensão do ficheiro que contém a programação do robô.

Um ficheiro RAPID divide-se em três fases essenciais. Definição das variáveis, onde também são definidos os *targets*, procedimentos, pequenas funções com instruções de movimento do robô e, por fim, a função *main()*, função principal que controla toda a execução do programa, chamando os procedimentos necessários.

É nas definições das variáveis que são escritos os *targets*. Assim, consoante os valores das coordenadas armazenados e as orientações determinadas anteriormente, serão escritos todos os *targets* necessários para a operação de pintura no objeto pretendido. Como foi visto anteriormente, cada linha necessitará de dois pontos para a sua definição. Na Figura 4.5 é apresentada a definição de dois *targets* responsáveis pela descrição de uma linha.

```
CONST robtarget Line0i :=[[25.63,23.88,100.0],[0.12,-0.67,0.71,0.11],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
CONST robtarget Line0f :=[[79.18,68.73,100.0],[0.12,-0.67,0.71,0.11],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
```

Figura 4.5: Definição de dois *targets*

Os dados extraídos do círculo contêm as coordenadas do centro e o raio da circunferência. Contudo, com estes dados não será possível que o robô desenhe o círculo no objeto, pois este para descrever um círculo necessita de, pelo menos,

quatro *targets*. Para tal determinou-se quatro pontos baseando-se no centro e no raio. Com base na Figura 4.6 e assumindo que o ponto C é o centro do círculo com coordenadas x, y e um raio r, determinam-se as coordenadas dos pontos 1, 2, 3 e 4:

- **Ponto 1:** (x, y + R)
- **Ponto 2:** (x + R, y)
- **Ponto 3:** (x, y - R)
- **Ponto 4:** (x - R, y)

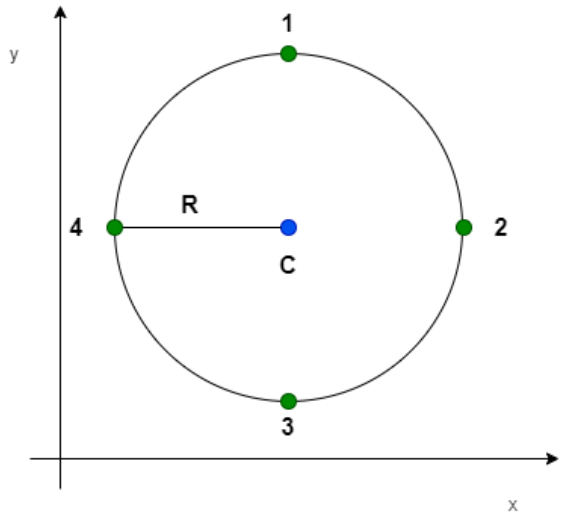


Figura 4.6: Demonstração para a extração de quatro pontos numa circunferência

Assim, são definidos e escritos os quatro *targets* no programa RAPID, capacitando o robô para a descrição de um círculo. Na Figura 4.7 são apresentados os quatro pontos que definem um círculo numa das faces do modelo.

```
CONST robtarget Circleo01 :=[[ 100,42.91,68],[0.422,-0.01,-0.90,-0.007],[0,-1,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
CONST robtarget Circleo02 :=[[ 100,24,50.59],[0.422,-0.01,-0.90,-0.007],[0,-1,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
CONST robtarget Circleo03 :=[[ 100,42.91,32],[0.422,-0.01,-0.90,-0.007],[0,-1,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
CONST robtarget Circleo04 :=[[ 100,60,50.59],[0.422,-0.01,-0.90,-0.007],[0,-1,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
```

Figura 4.7: Definição de quatro pontos para a descrição de um círculo

O passo seguinte para a escrita do programa são os procedimentos. Existem dois tipos diferentes de procedimentos. Os procedimentos de coloração e os procedimentos de desenho no objeto. Os procedimentos de coloração são responsáveis pelas instruções de movimento do robô até às diferentes cores da paleta. Os procedimentos de desenho descrevem todos os *targets* que o robô atravessa com

o objetivo de desenhar no modelo pretendido. Os procedimentos responsáveis pelo movimento do robô até à paleta são fixos, isto é não dependem do ficheiro DXF importado para a aplicação. Os restantes procedimentos são dinâmicos, dependendo assim do desenho CAD que o utilizador importa para o programa de conversão. Na Figura 4.8 é apresentado um procedimento que tem como função molhar o pincel na tinta de cor azul.

```
81 PROC mov_azul()
82     MoveL pontomedio,v500,fine,pincel\WObj:=Workobject_1;
83     MoveL azulcima,v50,fine,pincel\WObj:=Workobject_1;
84     MoveL azulbaixo,v50,fine,pincel\WObj:=Workobject_1;
85     MoveL azulcima,v50,fine,pincel\WObj:=Workobject_1;
86     MoveL pontomedio,v500,fine,pincel\WObj:=Workobject_1;
87 ENDPROC
```

Figura 4.8: Instrução de movimento para molhar o pincel na tinta azul

De seguida, é necessário escrever o procedimento main(), rotina esta que controla todo o fluxo do programa. Primeiramente, de acordo com a cor a utilizar no desenho, chamar-se-á o procedimento responsável pelo movimento do robô até à tinta pretendida. De seguida, é necessário inserir na main() as instruções responsáveis pela descrição de determinado desenho no objeto. Nas Figuras 4.9 e 4.10 são apresentados exemplos de instruções geradas automaticamente, para o desenho de um círculo e de uma linha, respetivamente.

```
MoveJ Circleo01 , v50, fine, pincel\WObj := wcubo;
MoveC Circleo02 , Circleo03 , v50, fine, pincel\WObj := wcubo;
MoveC Circleo04 , Circleo01 , v50, fine, pincel\WObj := wcubo;
```

Figura 4.9: Instrução de movimento para descrição de um círculo

```
MoveJ Line0i, v50, fine, pincel\WObj := wcubo;
MoveL Line0f, v50, fine, pincel\WObj := wcubo;
```

Figura 4.10: Instrução de movimento responsável pelo desenho de uma linha

Depois do robô executar as instruções de movimento a fim de desenhar no modelo, é chamada, no procedimento *main()*, um procedimento responsável por lavar o pincel em água, para que não ocorra mistura de cores.

4.3 Modelo de Simulação

Depois de gerado o ficheiro RAPID foi necessário recorrer-se ao *software* de simulação. Contudo, para que a simulção se aproxime da realidade, é fundamental recorrer a uma modelização da célula. Na Figura 4.11 é apresentado o modelo da

célula. Assim, foram identificados três objetos essenciais para a definição deste ambiente de simulação. O robô e a sua ferramenta, o objeto a desenhar e a paleta de cores.

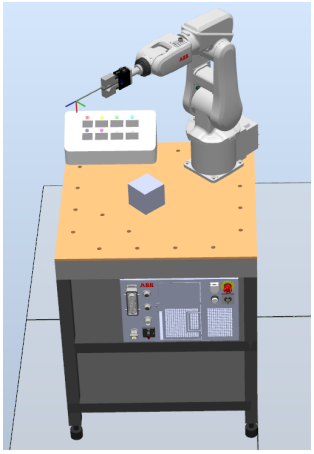


Figura 4.11: Célula Modelada

Para a modelação do robô e do controlador, utilizou-se os modelos incluídos na biblioteca do *Robot Studio*, que possuí modelos de robô já desenvolvidos. Nesta célula utilizou-se o modelo IRB 120, apresentado na Figura 4.12.

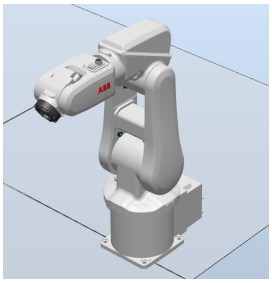


Figura 4.12: Robô utilizado na criação da célula

Tendo em conta o principal objetivo deste projeto, pintura de um objeto de forma automática, o atuador final teria que ser uma qualquer ferramenta que permitisse aplicar no modelo um desenho. Contudo, como uma das principais funcionalidades deste desenvolvimento é a inclusão de cores no desenho definiu-se que a ferramenta acoplada ao robô deveria ser um pincel de pintura tradicional. Para a fixação do pincel utilizou-se uma garra mecânica com atuação pneumática. Na Figura 4.13 é apresentado o conjunto da garra e pincel.

O objeto, apresentado na Figura 4.14, ao qual se pretende aplicar um desenho é o cubo. O cubo modelado é de cor branca para que depois de aplicado o desenho se destaque do modelo.

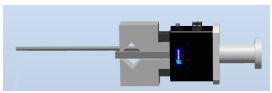


Figura 4.13: Modelo do atuador final

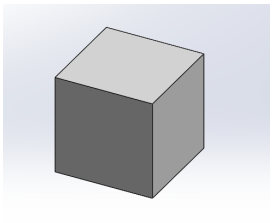


Figura 4.14: Objeto a ser a aplicado o desenho

Por fim, para disponibilizar todas as cores suportadas pelo formato DXF na coloração do desenho, foi desenvolvida uma paleta de cores, Figura 4.15, para que de uma forma simples o robô pudesse molhar o pincel na tinta pretendida ou então lavá-lo na água.

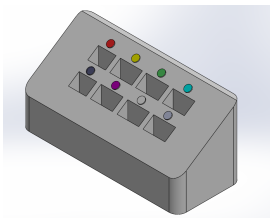


Figura 4.15: Paleta de cores para o modelo

4.3.1 WorkObjects Utilizados

Qualquer *target* criado terá que ser relacionado a um *workobject* que funciona como um sistema de coordenadas local. Neste projeto utilizaram-se dois *workobjects*, um responsável pelos *targets* do desenho do cubo, designado de *wcubo* e outro que referencia os restantes *targets* ao longo de todo o programa, chamado de *Workobject_1*.

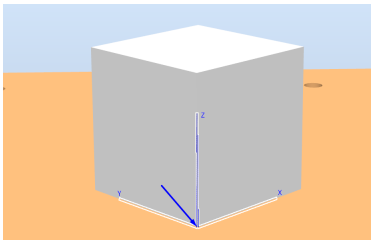
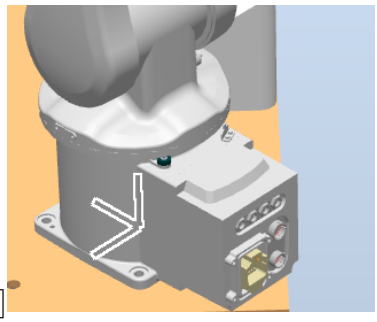
[a]  [b] 

Figura 4.16: Dois *WorkObjects* utilizados

4.4 Conclusão Capítulo 4

Deste modo, toda a implementação da solução foi executada com sucesso, chegando-se a um sistema capaz de cumprir todos os objetivos a que foi proposto. Assim, todas as ferramentas utilizadas apresentaram-se com uma mais valia para o desenvolvimento do projeto, apresentando um elevado nível de produtividade. O ambiente de programação provou ser bastante capaz e todas as ferramentas disponibilizadas pelo IDE facilitaram a programação da aplicação. Por fim, na simulação o *Robot Studio* permitiu uma modelização rápida e bastante completa.

Capítulo 5

Testes da Solução e Resultados Obtidos

Depois de terminada a fase de implementação da solução desenvolvida viu-se necessário a realização de testes e interpretação de resultados por forma a validar o sistema projetado. Assim, foram percorridas todas as etapas essenciais para o correto funcionamento do projeto com objetivo de verificar problemas e aplicar melhorias que podem apresentar-se como mais valias nesta dissertação. Ao longo deste capítulo, demonstrar-se-á o funcionamento do programa desenvolvido, através da perspetiva do utilizador.

5.1 Criação do Desenho na Peça

Primeiramente, abrira-se-á o ficheiro que contém o modelo da peça que se pretende pintar. Neste caso, o objeto utilizado foi o cubo. Utilizando as ferramentas de desenho do AutoCAD, aplicar-se-á uma pintura nas faces do cubo. O utilizador poderá desenhar o que pretender, usando as formas suportadas pela aplicação, que neste caso são as linhas e círculos. Através das propriedades de cada forma é possível escolher de entre sete cores para a colorir. Na Figura 5.1 é apresentado o modelo de um cubo onde o utilizador pretende pintar automaticamente um sol, a cor verde, na face superior e um sinal de proibido, em azul, numa das faces laterais do objeto.

5.1.1 Exportação para ficheiro DXF

O AutoCAD possui inúmeros formatos de exportação que são indicados com uma determinada finalidade. Contudo, para que seja possível importar o desenho para a aplicação desenvolvida, é necessário exportá-lo para o formato de DXF. Todo o

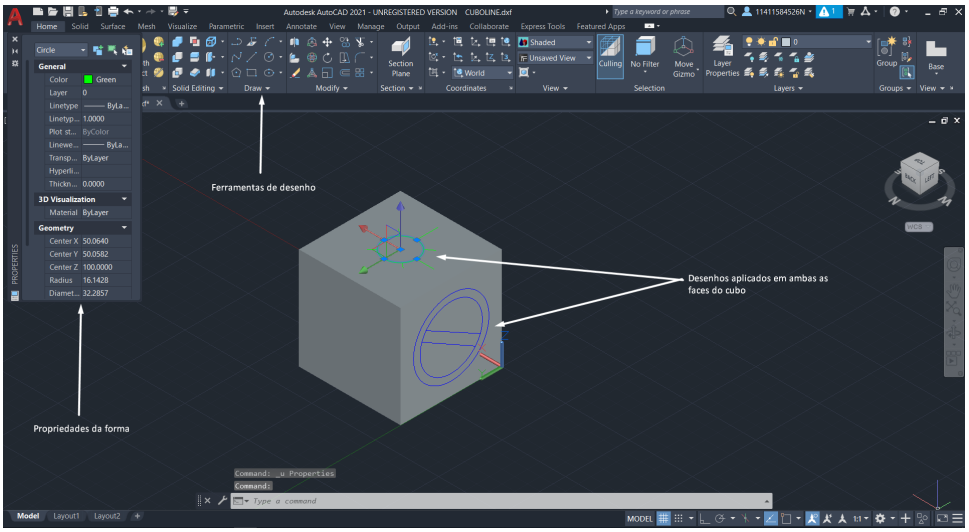


Figura 5.1: Ambiente de desenho na peça

mecanismo de exportação é bastante intuitivo, para tal basta guardar o desenho e escolher o tipo de ficheiro DXF, como demonstra a Figura 5.2

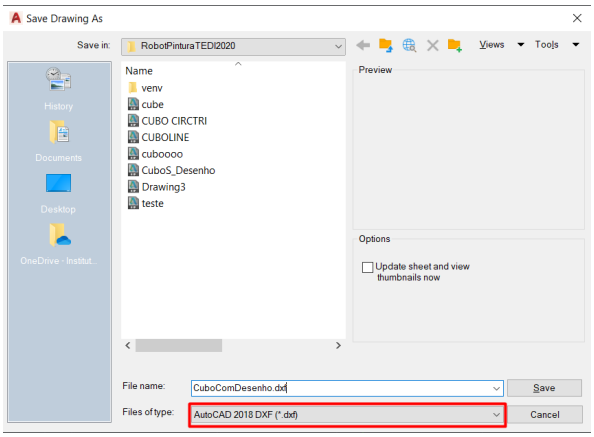


Figura 5.2: Exportação do desenho para o tipo DXF

5.1.2 Interface de utilizador

Hoje em dia, as aplicações tendem a apresentar um *design* simples, intuitivo que prima por disponibilizar ao utilizador uma experiência limpa mas, ao mesmo tempo, capaz. Assim, a aplicação foi desenvolvida por forma a possuir uma interface muito simples. O programa desenvolvido apresenta unicamente quatro botões, tal como demonstra a Figura 5.3

O botão 'Acerca do projeto', depois de pressionado, abre uma aba nova onde

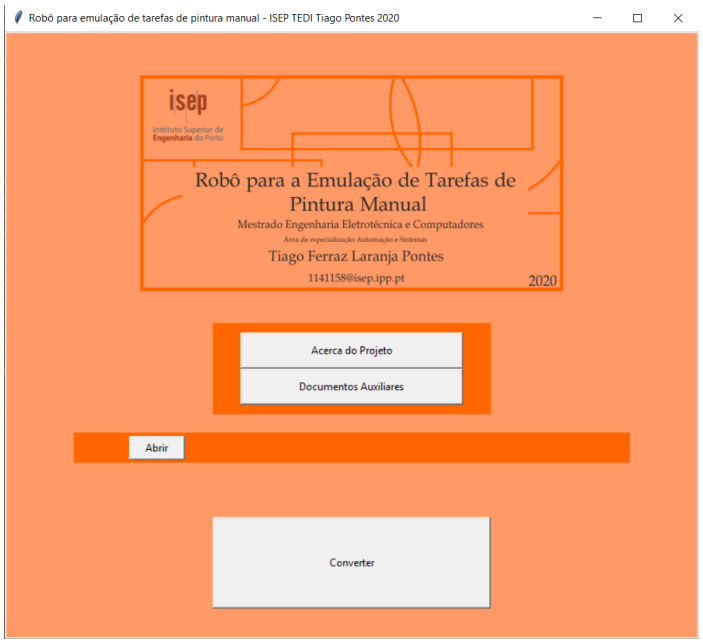


Figura 5.3: Interface de utilização da aplicação desenvolvida

se encontra o relatório em formato *Portable Document Format* (PDF). Caso o utilizador pretenda aceder ao código fonte da aplicação basta clicar no botão 'Documentos Auxiliares'.

Para proceder à importação do ficheiro DXF para a aplicação, é necessário clicar no botão 'Abrir' onde será disponibilizada uma janela com navegador de ficheiros que permite selecionar o ficheiro DXF que pretende exportar. O navegador de ficheiros, apresentado na Figura 5.4, apresenta um filtro e só permite a importação de ficheiros do tipo DXF, evitando assim possíveis falhas ou distrações do utilizador.

Caso o utilizador não selecione nenhum ficheiro, é apresentada uma mensagem de erro, conforme demonstra a Figura 5.5.

Depois do utilizador selecionar o ficheiro que pretende importar para a aplicação e clicar no botão 'Converter' surge uma mensagem de confirmação que indica onde se encontra o ficheiro gerado. Na Figura 5.6 é apresentada a mensagem após o utilizador pressionar o botão de conversão.

Após a geração do ficheiro, a próxima fase é a simulação do programa RAPID num *software*.

5.1.3 Simulação Robot Studio

A primeira etapa para a simulação do programa é a abertura da célula já desenvolvida. De seguida copia-se o conteúdo do ficheiro gerado para a secção do

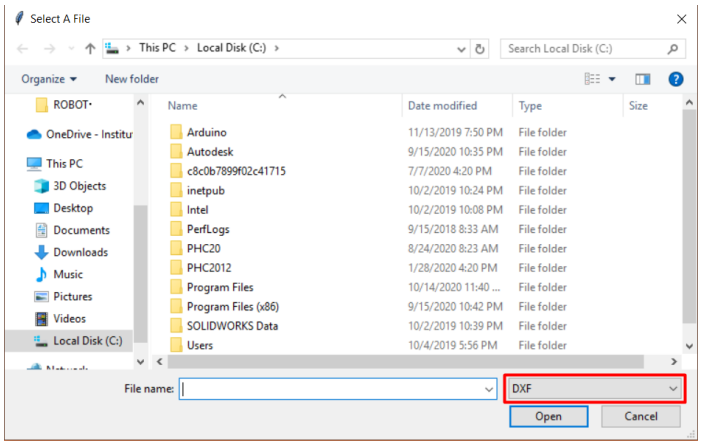


Figura 5.4: Navegador de ficheiros da aplicação

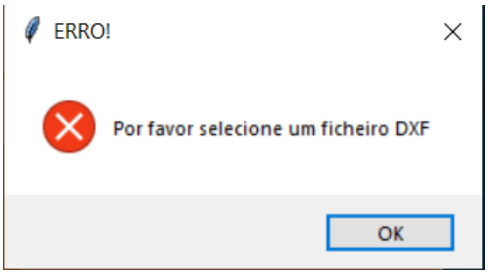


Figura 5.5: Erro caso o utilizador não selecione nenhum ficheiro DXF

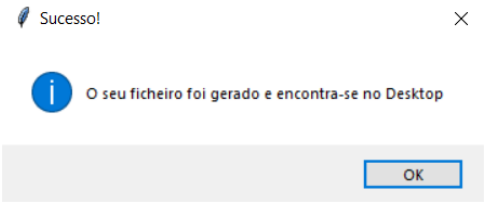


Figura 5.6: Janela de confirmação da geração do ficheiro

RAPID do *Robot Studio* responsável pela programação da célula.

Posteriormente terá que se sincronizar o programa RAPID com a célula robótica para que esta crie os *targets* e os *paths* para efeitos de simulação. De seguida inicia-se a simulação carregando no botão iniciar.

Na Figura 5.7 observa-se o programa pronto a iniciar, onde já se encontram representados todos os *targets* e *paths* que irão ser atravessados pelo atuador final do robô. Para efeitos de simulação, alterou-se a cor do cubo para preto para que se torne mais fácil a visualização dos desenhos nas faces, apresentados pelos *paths*, a amarelo.

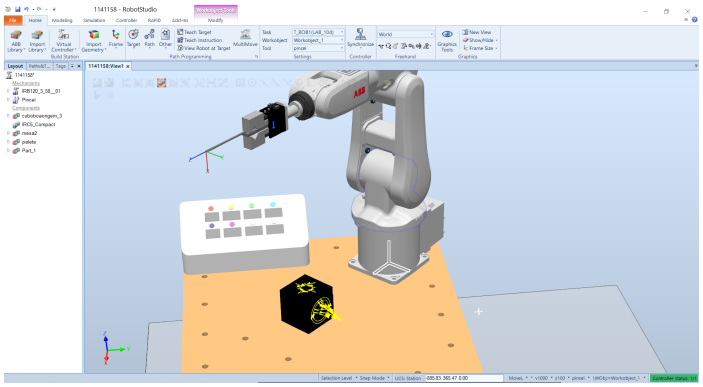


Figura 5.7: Ambiente de simulação já com ficheiro RAPID importado

5.2 Resultados Obtidos

Os resultados obtidos em todos os testes efetuados nas diversas fases do desenvolvimento do projeto, permitiram a melhoria contínua do mesmo.

Os principais ajustes efetuados foram relativos à posição dos diferentes elementos constituintes da célula. A posição do cubo foi sofrendo algumas alterações até que se determinou qual a posição que permite que o pincel atue em pelo menos três faces do objeto com facilidade. A paleta de cores foi também colocada especificamente num local que permita ao robô aceder a todas as cores com facilidade sem encontrar nenhum obstáculo.

Por último, através de alguns testes, achou-se essencial o uso de um *target* de transição, paralelo a cada uma das faces, para que o atuador não embata no cubo sempre que se movimenta para a paleta.

5.3 Conclusão Capítulo 5

Assim, todos os testes efetuados em ambiente de simulação mostraram-se fundamentais para o sucesso do sistema, uma vez que permitiram descobrir possíveis problemas e encontrar soluções para os mesmos. Em suma, toda a solução desenvolvida, mostrou-se robusta e bastante capaz, cumprindo assim todos os objetivos a que foi proposta.

Capítulo 6

Conclusões

A presente dissertação procurou apresentar uma solução robotizada capaz de garantir a pintura automática de qualquer produto, em contexto industrial ou artístico. Deste modo, com o objectivo de aumentar a flexibilização e otimização dos processos de personalização de produtos, foi criada uma aplicação que converte um desenho CAD em formato apreensível por um robô.

Depois de analisadas as propostas já existentes nesta matéria, foi selecionado o método mais eficaz e melhor se adapta aos objetivos pretendidos. Assim, em primeiro lugar, foi escolhida a linguagem de programação Python em detrimento de outras, pelo facto de esta ser uma linguagem de alto nível, intuitiva, com uma sintaxe bastante fluída e com uma curva de aprendizagem acentuada, sendo, portanto, o método mais adequado para o tipo de aplicação em causa. A escolha da linguagem Python garante, ainda, um tempo de processamento curto, o que favorece a utilização da aplicação. Por forma a fornecer uma utilização ágil da aplicação, foi criado um ambiente gráfico simples e intuitivo que permite ao utilizador uma fácil apreensão do *software* desenvolvido. Ao nível do formato de exportação do desenho, foi escolhido o DXF, uma vez que, em virtude da sua extensa aplicação, este apresenta um elevado suporte técnico, o que facilita a sua utilização. Por outro lado, a utilização do formato DXF foi, igualmente motivada, pela escolha prévia do *software* CAD, uma vez que ambos pertencem ao mesmo proprietário ou seja, a *AUTODESK*, pelo que ambos apresentam uma excelente compatibilidade.

Todavia, e apesar de todos os objetivos a que se propôs o presente projeto terem sido alcançados, de uma forma eficiente, foram identificadas alguns problemas no decurso do mesmo. O primeiro problema que surgiu prendeu-se com o facto de o formato DXF ser um pouco arcaico e não possuir atualizações recentes. Deste modo, poder-se-ia ter selecionado o tipo STEP, uma vez que este se

apresenta bastante adequado ao contexto industrial, sendo um formato alvo de constantes atualizações e que dispõe de diversas funcionalidades alheias ao DXF. Por fim, esta dissertação desenvolveu-se no decorrer da pandemia COVID-19, o que impediu a normal concretização do projeto, através de testes e ensaios em ambiente de laboratório.

6.1 Desenvolvimentos para Projetos Futuros

Ao nível dos desenvolvimentos futuros deste projeto, em primeiro lugar seria imprescindível a realização de testes e ensaios em contexto laboratorial, o que permitiria o aperfeiçoamento da técnica de pintura e o estudo da localização física de todos os objetos da célula. Por outro lado, e por forma a incrementar as funcionalidades do projeto criado, seria interessante permitir a importação para a aplicação, de uma imagem obtida através de uma câmara para que a mesma fosse pintada automaticamente no objeto.

Referências Bibliográficas

[1] 2011, "George devol: A life devoted to invention, and robots - ieee spectrum." https://spectrum.ieee.org/automaton/robotics/industrial-robots/george-devol-a-life-devoted-to-invention-and-robots. (Accessed on 06/28/2020). [cited on p. v, 5, 6]

[2] K. Pouspourika, "The 4 industrial revolutions - institute of entrepreneurship development." https://ied.eu/project-updates/the-4-industrial-revolutions/. (Accessed on 04/06/2020). [cited on p. v, 6, 7]

[3] "Ai robot ai-da presents original artworks in university of oxford exhibition." https://www.dezeen.com/2019/06/14/ai-robot-ai-da-artificial-intelligence-art-exhibition/. (Accessed on 06/30/2020). [cited on p. v, 16, 17]

[4] "edavid robot - pesquisa google." https://www.google.com/search?sxsrf=ALeKk02K1aRYc2-eTFJu080nmXYbvNrxHw%3A1593525481460&ei=6UT7XsPSG_WVjLsPkOaRuAc&q=edavid+robot&oq=edavid+robot&gs_lcp=CgZwc3ktYWIQA1AAWABgyChoAHAAeACAAQCIAQCSAQCYAQCqAQdnd3Mtd2l6&sclient=psy-ab&ved=0ahUKEwiD64yC2anqAhX1CmMBHRBzBHcQ4dUDCAw&uact=5. (Accessed on 06/30/2020). [cited on p. v, 17, 18]

[5] "cloudpainter - an artificially intelligent painting robot." https://www.cloudpainter.com/. (Accessed on 06/30/2020). [cited on p. v, 17, 18]

[6] "The incredible growth of python | stack overflow." https://stackoverflow.blog/2017/09/06/incredible-growth-python/. (Accessed on 10/11/2020). [cited on p. vi, 33]

[7] K. Schwab, "Indústria 4.0: entenda o que é a quarta revolução industrial — startse." https://www.startse.com/noticia/nova-economia/

industria-4-0-entenda-o-que-e-quarta-revolucao-industrial. (Accessed on 04/06/2020). [cited on p. 7]

[8] Kagermann, "(1) (pdf) a discussion of qualifications and skills in the factory of the future: A german and american perspective." https://www.researchgate.net/publication/279201790_A_Discussion_of_Qualifications_and_Skills_in_the_Factory_of_the_Future_A_German_and_American_Perspective, April 2015. (Accessed on 04/09/2020). [cited on p. 7]

[9] J. Lippert, "Humans vs. robots: The car production debate - bnn bloomberg." https://www.bnnbloomberg.ca/humans-vs-robots-the-car-production-debate-1.976017, January 2018. (Accessed on 04/09/2020). [cited on p. 8]

[10] "Sistemas robóticos de locomoção." https://moodle2.isep.ipp.pt/pluginfile.php/235067/mod_resource/content/1/ROBIN_Cap_1_Introducao_Robotica_2017_18.pdf. (Accessed on 10/13/2020). [cited on p. 9]

[11] A. Owen-Hill, "What are the different programming methods for robots?." https://blog.robotiq.com/what-are-the-different-programming-methods-for-robots, 2016 05. (Accessed on 06/15/2020). [cited on p. 13, 14, 15]

[12] "arte | diccionario panhispánico de dudas | rae - asale." https://www.rae.es/dpd/arte. (Accessed on 06/28/2020). [cited on p. 15]

[13] "Movie tech: How 'gravity' threw sandra bullock into zero gravity." https://www.nbcnews.com/sciencemain/movie-tech-how-gravity-threw-sandra-bullock-zero-gravity-8C11326787. (Accessed on 06/28/2020). [cited on p. 15]

[14] "Immutable object - wikipedia." https://en.wikipedia.org/wiki/Immutable_object. (Accessed on 06/30/2020). [cited on p. 16]

[15] "Ai-da is the world's first humanoid ai artist | dazed." https://www.dazeddigital.com/art-photography/article/43240/1/meet-ai-da-the-worlds-first-ai-robot-artist-aiden-meller. (Accessed on 06/30/2020). [cited on p. 16]

[16] "Sketchpad - complete history of the sketchpad computer program." https://history-computer.com/ModernComputer/Software/Sketchpad.html. (Accessed on 06/30/2020). [cited on p. 18]

[17] "Cad: A brief history." https://www.digitalschool.ca/cad-a-brief-history/. (Accessed on 06/30/2020). [cited on p. 18]

[18] "Top 10 best cad software for all levels - 3dnatives." https://www.3dnatives.com/en/top10-cad-software-180320194/#! (Accessed on 06/30/2020). [cited on p. 19]

[19] "Step-file, iso 10303-21." https://www.loc.gov/preservation/digital/formats/fdd/fdd000448.shtml. (Accessed on 06/30/2020). [cited on p. 21]

[20] "Robotics industry insights - robotic paint automation ...." https://www.robotics.org/content-detail.cfm/Industrial-Robotics-Industry-Insights/Robotic-Paint-Automation-for-Smaller-Industrial-Operations/content_id/983. (Accessed on 07/07/2020). [cited on p. 23]

[21] P. Bourke, "Minimal dxf." http://paulbourke.net/dataformats/dxf/min3d.html. (Accessed on 10/08/2020). [cited on p. 27]

Anexo A

Especificações Técnicas Robô IRB 120 da ABB

ABB

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.

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)	Handling capacity (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
1 kg picking cycle	
25 x 300 x 25 mm	0.58 s
25 x 300 x 25 with 180° axis 6 reorientation	0.92 s
Acceleration time 0-1 m/s	0.07 s
Position repeatability	0.01 mm

Technical information

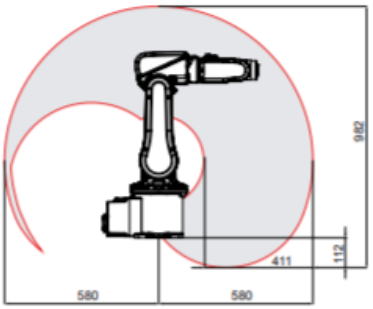
Electrical Connections	
Supply voltage	200-600 V, 50/60 Hz
Rated power transformer rating	3.0 kVA
Power consumption	0.24 kW
Physical	
Robot base	180 x 180 mm
Robot height	700 mm
Robot weight	25 kg
Environment	
Ambient temperature for robot manipulator:	
During operation	+5°C (41°F) to +45°C (113°F)
During 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)**

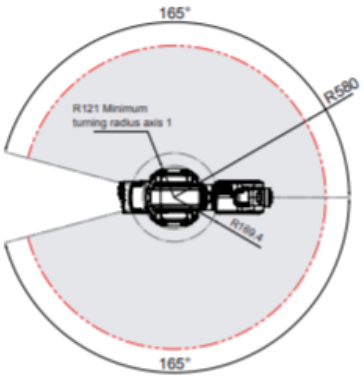
** ISO class 4 can be reached under certain conditions.
Data and dimensions may be changed without notice.

Movement

Axis movement	Working range	Velocity IRB 120
Axis 1 rotation	+165° to -165°	250°/s
Axis 2 arm	+110° to -110°	250°/s
Axis 3 arm	+70° to -110°	250°/s
Axis 4 wrist	+160° to -160°	320°/s
Axis 5 bend	+120° to -120°	320°/s
Axis 6 turn	Default: +400° to -400° Max. rev: +242 to -242	420°/s

Working range





iber 2019