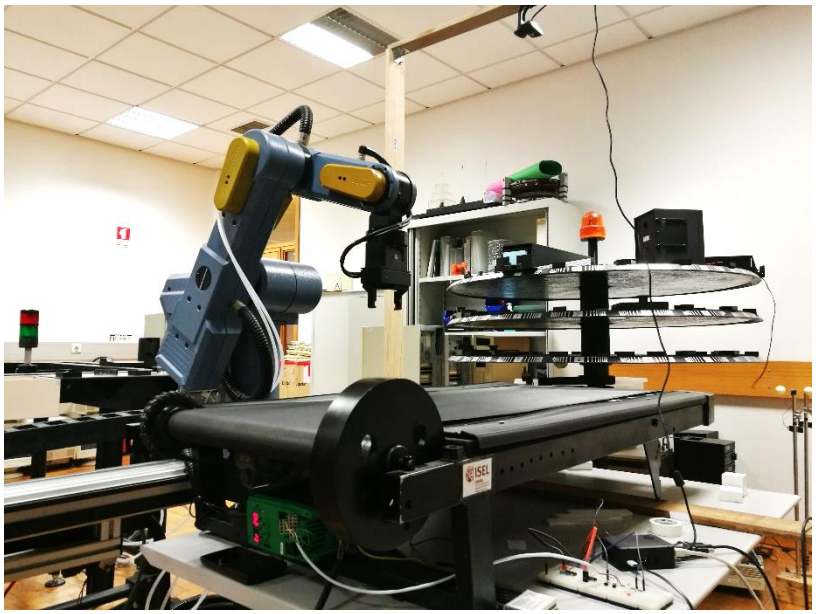
INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Área Departamental de Engenharia Eletrotécnica Energia e Automação



Automatização de sistema de separação de peças com recurso a um tapete rolante e um robot e processamento de imagem

JORGE MANUEL DE FIGUEIREDO COSTA
(Licenciado em Engenharia Eletrotécnica)

Trabalho Final de Mestrado para obtenção do grau de Mestre
em Engenharia Eletrotécnica – ramo de Automação e Eletrónica industrial

Orientadores:
Professora Doutora Maria da Graça Vieira de Brito Almeida (ISEL)
Professor Doutor Armando José Leitão Cordeiro (ISEL)

Júri:
Presidente: Professor Doutor Luís Manuel dos Santos Redondo (ISEL)
Vogais:
Professora Doutora Mafalda Maria Morais Seixas (ISEL)
Professora Doutora Maria da Graça Vieira de Brito Almeida (ISEL)

Abril de 2019

Abstract

The current Master Thesis was carried out during the development of a typical industrial automation solution, which consists of the task of the image detection and capture by an industrial robot, of objects being transported along a conveyor, with the information of its position taken through a video camera installed above the same conveyor.

This project relied on several disciplines such as robotics, artificial vision, and electromechanical drives, all of which are linked to a controller in a centralized control topology. This led to the study of some of the cutting-edge topics that have been under significant evolutions in the last decades, and that aim to be all over the industrial environment.

Some image processing techniques were analyzed with the purpose of developing an algorithm that allowed to detect some characteristics of the object, in order to capture it in the desired orientation.

The characterization of the robot used, passed through the study of the various existing models, in order to better understand their applications.

Both point above are quite present in two quite current topics: Industry 4.0 and Artificial Intelligence. Thus, a small reference was made to its current industry framework.

A variable-frequency-drive was also used to control the conveyor so as to be able to adapt the transportation of the object to its capture.

With several isolated systems applied, with different characteristics and connected to the same controller, it was also necessary to develop interfaces that allowed to integrate all of them in a single solution.

Finally, a complete software was developed to run on the controller, putting the entire system in operation as described. A calibration program was also created, which allowed to adapt the operation of the first program to the surrounding environment.

Keywords: Automation, Robotics, Image Processing, Electromechanical Drives, Industrial Electronics, Industrial Control

Resumo

A presente dissertação de mestrado foi realizada no seguimento do desenvolvimento de uma solução típica de automação industrial a qual consiste na tarefa de deteção por imagem e captura por um robot industrial, de objetos a serem transportados ao longo de um transportador, sendo a informação do seu posicionamento e orientação recolhida através de uma câmara instalada sobre o transportador.

O projeto recorreu a diversas disciplinas tais como a robótica, visão artificial, e acionamentos eletromecânicos, todos estes a funcionar ligados a um controlador numa topologia de controlo centralizado. Isto levou a um estudo de alguns dos temas que mais têm sofrido evoluções significativas nas últimas décadas e que se propõem a dominar o meio industrial.

Foram analisadas algumas técnicas de processamento de imagem com o objetivo de desenvolver um algoritmo que permitisse detetar algumas características da peça, de modo a capturar a peça na orientação pretendida.

A caracterização do robot utilizado passou pelo estudo dos vários tipos de modelos existentes, de modo a entender melhor os seus meios de aplicação.

Ambos os pontos acima estão bastante presentes em dois temas atuais: Indústria 4.0 e Inteligência artificial. Assim foi feita uma pequena referência ao seu enquadramento na indústria atual.

Utilizou-se também um variador eletrónico de velocidade para controlar o transportador de modo a poder-se adaptar o meio de transporte das peças à captura dos objetos.

Existindo assim vários sistemas isolados com características diferentes e ligados a um mesmo controlador, foi necessário desenvolver interfaces que permitissem a integração de todos numa única solução.

Por último, foi desenvolvido um software completo para ser executado no controlador, colocando todo o sistema em funcionamento conforme descrito. Foi também criado um programa de calibração do sistema de visão o qual permitiu adaptar o funcionamento do anterior ao meio envolvente.

Palavras chave: Automação, Robótica, Processamento de Imagem, Acionamentos Eletromecânicos, Eletrónica Industrial, Controlo Industrial.

Agradecimentos

O presente trabalho só foi possível com o contributo de algumas pessoas que de alguma forma me ajudaram neste percurso académico.

Em primeiro lugar agradeço aos meus orientadores, Doutora Graça Almeida e Doutor Armando Cordeiro por toda a orientação, apoio, disponibilidade e disponibilização de recursos em todas as fases do desenvolvimento desta dissertação.

Ao departamento de Engenharia Mecânica pela disponibilização do robot utilizado no projeto sem o qual não teria sido possível a sua implementação prática.

Aos meus pais Custódio Costa e Carminda Costa, por todo o apoio, amor e incentivo ao longo desta difícil etapa.

Por último e não menos importante, a todos os amigos e colegas que me foram acompanhando ao longo desta jornada académica e sem os quais chegar aqui teria sido muito mais difícil.

Índice Geral

Índice de figuras

Índice de tabelas

Acrónimos

AC – Corrente Alternada

API – *Application* Programming *Interface*

BCM – *Broadcom SOC channel*

CC – Corrente Contínua

CP – *Continuous Path* (Trajetória Contínua)

CFA – *Color Filter Array*

DAC – *Digital-Analog Converter* (Conversor Digital-Analógico)

FPGA's – *Field-Programmable Gate Array*

GPIO – *General Purpose Input/Output*

HDR – *High Dynamic Range*

IEC – *International Electrotechnical Commission*

IEA – *International Energy Agency*

I/O – *Input / Output* (Entrada / Saída)

ISO – *International Organization for Standardization*

MOS – *Metal-Oxide-Semiconductor*

OCR - *Optical Character Recognition*

OSI – *Open System Interconnection*

OPC – *Open Platform Communications*

PLC – *Programmable Logic Controller*

PTP – *Point-To-Point* (Ponto-a-Ponto)

PWM – *Pulse Width Modulation*

QoS – *Quality of Service* (Qualidade de Serviço)

RIA – *Robotic Industries Association*

ROI – *Region Of Interest* (Região De Interesse)

SoC – *System-On-a-Chip*

Thresholding – Técnica de processamento de imagem (Limiarização)

VEV – Variador Eletrónico de Velocidade

SPI – *Serial Peripheral Interface*

Simbologia

A – Ampere

cd – candela

cm - centímetro

kB - kilobyte

Hz – Hertz

m - metro

mm – milímetro

Ω – Ohm

px - *pixel*

rpm – rotações por minuto

s – segundo

V – Volt

W – Watt

1. Introdução

1.1. Motivação

Durante o processo criativo de uma solução de automação industrial, os vários sistemas constituintes são ainda desenvolvidos com o objetivo de desempenhar uma determinada função específica. Um exemplo disso são os robôs antropomórficos numa linha de montagem de veículos, que embora sejam equipamentos altamentes dinâmicos, são maioritariamente colocados durante todo o seu ciclo de vida numa determinada posição, para fazer determinada tarefa, com determinada ferramenta, fazendo movimentos pré-definidos. Quando essa linha de montagem é alterada para um novo veículo, o projeto de desenvolvimento volta ao ponto inicial.

Embora o termo “Automação” derive do conceito “automático”, atualmente a maioria dos processos industriais tanto por motivos de fiabilidade, como de necessidade de rápidos tempos de comissionamentos, continuam a estar limitados a determinadas premissas estabelecidas inicialmente. São automáticos no sentido de terem pouca ou nenhuma necessidade de intervenção humana, mas não no âmbito da tomada de decisões. No entanto nos anos mais recentes o paradigma tem começado a mudar.

Embora tenha acabado por se tornar também numa ferramenta publicitária da própria evolução económica mundial, o conceito de “Indústria 4.0” fomentou o desenvolvimento de plataformas que têm aproximado a criatividade social, da tecnologia que existe em meios de produção industrial. O resultado é uma interligação de inúmeros conceitos e tecnologias com o objetivo de resolver as necessidades fundamentais de qualquer meio industrial: Adquirir soluções fiáveis a longo prazo e eficazes nos objetivos de produzir cada vez mais com menos.

É a partir desta necessidade de rentabilizar recursos técnicos e otimizar os processos, que têm surgido novas soluções adaptáveis a vários processos em tempo operacional. O tratamento da informação captada pelos mais variados métodos de aquisição e a interação entre toda a cadeia de componentes de processo, torna-se assim uma forma de reduzir os tempos de paragens por falhas e manutenções e de otimizar métodos previamente definidos. Como resultado, surge uma dinâmica enorme entre diversas áreas da engenharia, as quais já me despertavam bastante interesse antes de ingressar no Mestrado em Engenharia Eletrotécnica.

A Automação Industrial, como elo de ligação entre diferentes campos de desenvolvimento tecnológico, tem permitido que exista constantemente uma necessidade de adquirir novos conhecimentos de diversas áreas científicas. Não se trata apenas de eletrónica (comando e potência), nem de informática, nem de máquinas elétricas, nem de robótica, nem de mecânica, mas sim de isto tudo junto e muito mais em simultâneo.

A realização deste projeto, permitiu assim focar-me numa solução multidisciplinar dentro do que foi estudado no curso, e de aproveitar o conhecimento adquirido para o desenvolvimento de novas soluções mais eficazes.

O facto do presente documento resultar de um projeto de implementação prática, proporcionou que novos desafios surgissem num ambiente experimental. O aparecimento de questões físicas (por exemplo variações de iluminação natural) e de interligação entre diferentes componentes do sistema, tornou o projeto mais complexo, mas ao mesmo tempo mais rico em conhecimento.

1.2. Objetivos

O objetivo principal do estudo realizado, é o projeto de desenvolvimento de um sistema automático, de identificação de múltiplos objetos com vista à posterior captura por um braço robotizado.

Este sistema é composto por um tapete transportador de objetos de formato conhecido e dimensões diferentes (paralelepípedos), que surgem numa extremidade do transportador e deslocam-se até uma nova zona onde são capturados por um braço robótico. Sobre o tapete encontra-se uma câmara de captura de imagens, ligada a um controlador central que comanda em simultâneo o robot e o tapete sendo este acionado por um variador eletrónico de velocidade que alimenta um motor de indução. No decorrer do funcionamento, o controlador central, realiza a deteção dos objetos e das suas características com recurso ao processamento de imagem, permitindo assim fazer o acompanhamento das peças ao longo de todo o deslocamento.

É frequente neste tipo de soluções, as imagens capturadas serem trabalhadas na sua totalidade, resultando no processamento de dados que não adicionam valor ao processo. Tendo em conta que os algoritmos de processamento de imagem como descritos à frente tornam-se facilmente pesados a nível computacional, isto significa maiores tempos de processamento resultando em perdas de dados que podem acabar por tornar a deteção falível. Se tivermos em conta as possíveis acumulações de erros ao longo do sistema até chegar ao movimento de captura, pode o mesmo ter um resultado indesejável.

Assim foi desenvolvida uma solução de acompanhamento e captura de objetos que resumida na Figura 1 tem com os objetivos:

- Adquirir características físicas dos objetos através do processamento de imagens adquiridas por uma câmara.
- Controlar um transportador a partir de acionamentos disponíveis para permitir o deslocamento dos objetos detetados numa posição aleatória numa extremidade do tapete até um novo ponto.

- Capturar os objetos previamente detetados.
- Utilizar uma plataforma com flexibilidade suficiente para desenvolver o software de controlo necessário ao funcionamento do sistema
- Desenvolver as interfaces necessárias de modo a permitir a interligação de todos os sistemas.

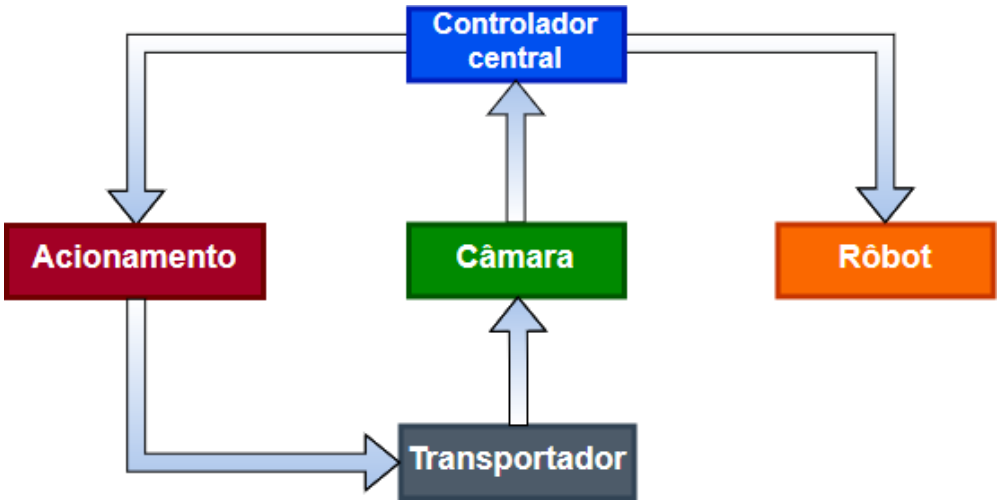


Figura 1. Modelo proposto para desenvolvimento do projeto

1.3. Estrutura da dissertação

O presente documento encontra-se dividido em 5 capítulos:

- O capítulo 1, serve como nota introdutória e enquadramento do leitor no tema da dissertação.
- O capítulo 2, refere-se às várias temáticas-chave que serviram de base para a elaboração da dissertação e desenvolvimento da solução. O mesmo está dividido nos 4 temas principais que inicialmente colocam o leitor no centro do sistema de controlo, e o guiam à posteriori para cada um dos pontos individuais das soluções utilizadas.
- O capítulo 3, refere-se à elaboração do projeto, começando pelos sistemas de hardware fundamentais à implementação mecânica do projeto e passando depois para o desenvolvimento do software.
- O capítulo 4 apresenta os resultados do projeto.
- No capítulo 5 são apresentadas as conclusões gerais do projeto com algumas reflecções sobre possíveis trabalhos futuros.

2. Estado de Arte

2.1. Introdução

Neste capítulo são apresentados os temas principais que serviram de base para a elaboração da presente dissertação a fim de permitir ao leitor um melhor enquadramento com os sistemas implementados no projeto.

Inicialmente é feita uma pequena viagem pelo mundo da automação industrial, passando por alguns dos pontos mais marcantes da história da evolução industrial, e terminando em temas bastante atuais tais como a Indústria 4.0 e a Inteligência Artificial.

Seguindo o tema automação, é feita uma apresentação da robótica industrial, fazendo referência às principais características destes equipamentos e as suas aplicações no meio.

De seguida é apresentado o conceito de processamento de imagem fazendo referência a algumas soluções existentes, e descrevendo as principais técnicas que foram utilizadas no projeto. Alguns dos exemplos utilizados para a demonstração dos conceitos referidos, serão imagens capturadas durante o desenvolvimento do projeto.

Por fim é feita referência aos motores elétricos e respetivos conversores industriais, equipamentos que também foram utilizados no desenvolvimento do projeto.

2.2. Automação industrial

Ao longo de milénios de evolução o ser humano foi desenvolvendo diversos mecanismos para analisar e interagir com o seu ecossistema, adaptando-se autonomamente aos problemas à medida que surgem. Ao mesmo tempo que foi melhorando o seu desempenho foi tentando descobrir novos métodos de automatizar as tarefas mais simples de modo a poder focar-se em novos objetivos. Tendo já alguns métodos definidos, a forma mais óbvia de criar sistemas automáticos foi começar por reproduzir as suas próprias operações e utilizar o conhecimento de si próprio transportando o seu desempenho para as ferramentas que foi construindo. Atualmente esta evolução histórica encontra-se relativamente bem documentada e definida por marcos relacionados com a capacidade dos humanos utilizarem cada uma das tecnologias que foram desenvolvendo. Se verificarmos, todo o período pré-histórico encontra-se subdividido pelos tipos de materiais (Idades da Pedra, Cobre, Bronze e Ferro) utilizados para a construção de ferramentas.

À medida que novas tecnologias foram surgindo, tornou-se mais fácil atingir novas metas que pudessem ser consideradas marcantes para a evolução do ser humano. Especialmente durante o último milénio, diversas invenções são reconhecidas como de extrema importância para a evolução da civilização tais como o moinho de vento, a máquina a vapor, o telégrafo, a lâmpada, a máquina elétrica, o computador, entre muitos outros. Existindo uma clara evolução desde a utilização exclusiva da energia mecânica até à

aplicação da energia elétrica nos dias de hoje (entre outras), diversas áreas tecnológicas foram surgindo possibilitando a execução de diferentes tarefas recorrendo à utilização dos mais variados recursos disponíveis.

Grande parte desta evolução deveu-se também à criação de métodos de controlar a forma como estes recursos eram utilizados. Durante os últimos dois séculos, à medida que a eletrificação da indústria foi tendo lugar, esta foi-se tornando cada vez mais difícil de controlar, tanto pela complexidade dos elementos utilizados como pelas suas velocidades de funcionamento. Isto levou a que o funcionamento das máquinas utilizadas começasse a ser ainda mais independente do ser humano tal como se pretendia. Se a invenção do relé eletromagnético (1830) foi um grande passo para a autonomia das máquinas, o transístor permitiu a evolução do controlo existente até à altura do seu aparecimento graças ao Modicon (1968), o primeiro PLC (*Programmable Logical Controller*). O seu tamanho muito inferior aos longos painéis de lógica cablada existentes até à altura, aliado à possibilidade de se poder alterar a sua programação com recurso a diferentes métodos de controlo, permitiu que a indústria evoluísse a uma velocidade superior com uma melhor e mais fácil interligação entre os diferentes equipamentos existentes.

Figura 2. Fotografia da equipa criadora do Modicon (primeiro PLC criado)

Nesta altura o controlo moderno já estivava bastante difundido principalmente devido à necessidade de desenvolver armamento durante os conflitos mundiais. Existem inclusive documentos datados da década de 60 que já falavam sobre a aplicação de computadores digitais na realização de operações de controlo industrial [1]. Isto levou a que já existisse a necessidade de incluir funções de controlo PID no equipamento. Entretanto à data do aparecimento do PLC, já a robótica estava presente no meio industrial à mais de meia década (ver capítulo 2.3). As linhas de produção da *General Motors* já incluíam diversos sistemas sincronizados na produção de automóveis.

Com o avanço nas tecnologias de comunicação começaram a ser desenvolvidos também redes de campo (*fieldbus*) para comunicação industrial. Com características diferentes das transmissões já existentes, estes tinham como objetivo a possibilidade de efetuar o controlo de sistemas de produção garantindo sempre uma elevada qualidade de servido (QoS). Redes tais como Modbus (1979), CAN Bus e HART (1986), e Profibus (1989) foram surgindo [2] possibilitando à indústria ter linhas de produção completas a trabalhar devidamente organizadas, maximizando a sua produtividade. Um passo importante para o avanço no desenvolvimento não só das redes industriais, mas de todos os tipos de comunicação foi a criação pela ISO (*International Organization for Standardization*) em 1970 do Modelo OSI (*Open System Interconnection*) [3]. Ainda servindo de referência nos dias de hoje, é uma hierarquia que define por camadas a organização e o funcionamento dos componentes existentes em redes de comunicação por forma a manter a interoperacionalidade entre diferentes sistemas.

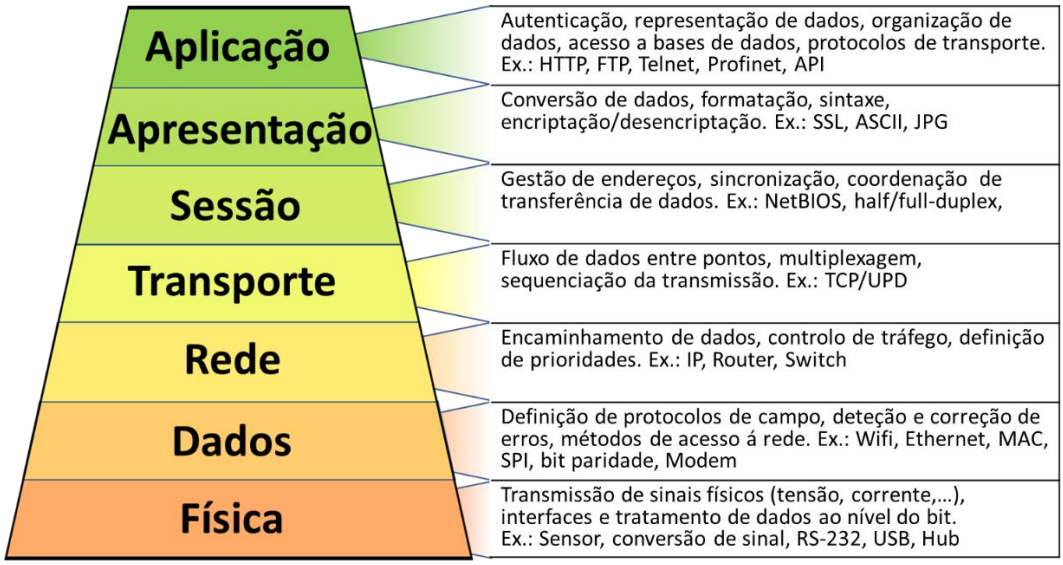


Figura 3. Modelo OSI

Com a evolução da eletrónica nas décadas seguintes foi criado um ciclo exponencial em que as evoluções da tecnologia de hardware e software foram-se alimentando dos seus próprios resultados para evoluírem até à realidade presente, onde a informática passou a ser algo banal para a generalidade do mundo civil. Atualmente é impossível fazer-se parte de uma sociedade moderna sem ter qualquer contacto com aparelhos que estabelecem algum tipo de comunicação. E o mesmo se passou no meio industrial. Com o aparecimento de sistemas mais robustos, começou-se a explorar novos conceitos de produção com vista a melhorar tanto o produto final como o próprio processo de fabrico. E aqui entra a Indústria 4.0.

Quando em 2011 surgiu pela primeira vez este conceito, o foco era em expor uma realidade em que fosse possível não só existir uma melhor comunicação homem-máquina disponibilizando mais informação sobre o estado dos processos, mas também máquina-máquina. Mas para tal era necessário que fossem criados canais de comunicação mais simples e menos complexos de implementar, e que o equipamento não dependesse de grandes estruturas para transmitir informações simples. Soluções como o Modelo OSI, e os sistemas OPC, foram grandes impulsionadores de uma indústria mais flexível, organizada e com menos barreiras. Passou-se a ter linhas de produção com máquinas com maiores capacidades de interligação entre si, possibilitando que os processos se tornassem mais unificados, mais fluidos, e com menos paragens.

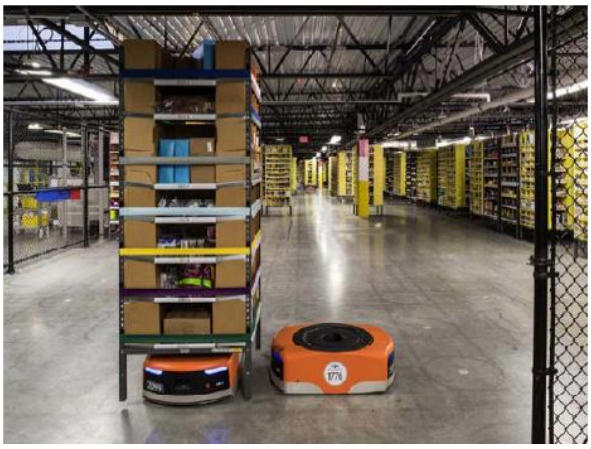


Figura 4. Veículos autónomos (AGV) a circular em ambiente industrial [4]

Outro grande passo foi o surgimento da chamada *Internet Of Things* (IoT). Começando por ser um conceito meramente civil, onde pequenos aparelhos banais do dia-a-dia como relógios, interruptores, começaram a ganhar características de troca de dados com rápidos e simples sistemas de comunicação, depressa a indústria viu uma oportunidade. Explorando tecnologias como o Wireless, Bluetooth ou o RFID, passou-se a ter trocas de dados diretas entre equipamentos sem a necessidade de existirem quaisquer intermediários nas comunicações. O resultado foi não termos somente equipamentos dedicados à gestão de redes a comunicar entre si, mas também diversos aparelhos com múltiplas funções que têm nas redes um meio de adquirir novas funcionalidades. Como exemplo industrial temos os veículos autónomos (AGV) a circular em fábricas (Figura 4) e a comunicar entre si para se auxiliarem nos movimentos evitando colisões e definindo os melhores percursos perante as condições existentes no momento e realizando uma gestão automática de frotas.

Sendo a inteligência artificial (IA) um tema também tão presente em todas as empresas tecnológicas atuais, facilmente surgiu a discussão sobre a sua aplicação em ambiente industrial. Se analisarmos o conceito no sentido em que pequenas partes do processo podem ganhar autonomia para realizar melhorias ao longo do tempo, acabando por

acelerar os ciclos produtivos, pode à partida parecer uma boa melhoria na indústria. No entanto a segurança é uma necessidade constante em qualquer indústria, e o facto de existir um fator de incerteza em algum ponto do processo, pode apresentar um perigo imediato. Utilizando o exemplo da Figura 4, pode facilmente acontecer o veículo autónomo com carga decidir acelerar para evitar colidir com o veículo que se encontra vazio (devido a alguma avaria no vazio por exemplo). Numa situação destas em que possa estar um operador fora da área de alcance dos sensores do aparelho pode à partida não ser um risco detetado pelo equipamento, no entanto a aceleração excessiva pode acabar por provocar uma queda da carga em cima do operador. Embora exemplos destes sejam facilmente detetados em situações onde podem ser encontradas mais valias, muitas outras situações podem ser mais difíceis de prever.

2.3. Robótica industrial

Embora os robôs industriais sejam hoje em dia um equipamento bastante proliferado pela indústria mundial, e bem conhecido pelo público geral, a sua definição deve ser sempre tida em conta para não existir qualquer confusão com o simples termo “robot” que tem um conceito muito mais abrangente no mundo das máquinas. Diferentes organizações ligadas a normas utilizam diferentes definições apesar de na maior parte dos casos variarem pouco. Como exemplo temos a ISO [5] e a RIA [6] que dão uma definição semelhante a um robot industrial, embora a ISO também diferencie especificamente um “manipulador” de um “robot industrial”:

Um “manipulador” é assim uma “máquina, normalmente com vários segmentos, ligados através de juntas rotativas ou lineares com o objetivo de agarrar ou mover objetos (peças ou ferramentas) com diversos graus de liberdade”.

Já um robot industrial, ambos tratam por “manipulador automaticamente controlado, reprogramável, polivalente, programável em 3 ou mais eixos, que pode funcionar fixo ou em aplicações móveis em aplicações industriais”.

O primeiro robot industrial surgiu em 1961 criado por George Charles Devol (considerado o Pai dos robôs industriais) [7]. O *Unimate* tratava-se de um equipamento electro-hidráulico que veio auxiliar o fabricante americano de automóveis GM (General Motors) nas tarefas de movimentação de pequenas peças metálicas fundidas, e de soldadura por pontos, funções que à data eram consideradas perigosas para os operadores. Embora tenha trazido ganhos significativos para o fabricante, na altura o equipamento ocupava bastante espaço e o controlador, que era baseado em memórias de discos sequenciais, era limitado e de difícil configuração, e apenas podia ser utilizado com pequenas cargas.

Com o aumento da procura por equipamentos deste tipo, juntamente com uma visão de possível expansão de aplicações no mercado, em 1974 a ASEA (hoje em dia ABB) desenvolveu o primeiro robot totalmente elétrico com controladores baseados no recém-chegado microprocessador Intel 8008. Tratava-se do *IRB 6*, embora continuasse a ser limitado nas tarefas a executar (apenas possuía 8 kB de memória disponível). Tal limitação foi ultrapassada quando no mesmo ano surgiu o primeiro robot programável por computador: o *The Tomorow Tool* desenvolvido pela Cincinnati Milacron (mais tarde comprada pela ABB).

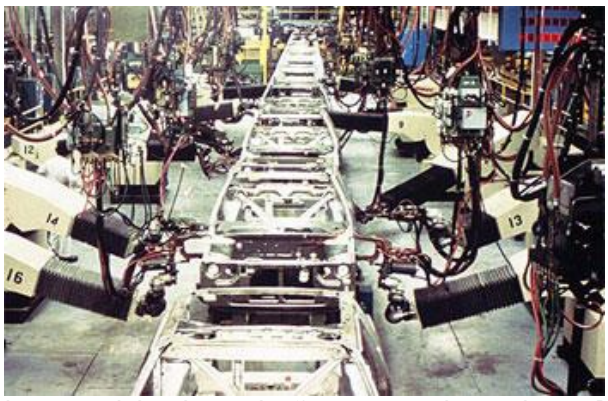


Figura 5. Linha de produção com o Robot Unimate [8]

Figura 6. Robot ASEA IRB 6 [9]

Dentro da definição dos chamados robôs industriais existem várias categorias que são atribuídas mediante a composição dos seus elementos e consequentemente movimentos possíveis de executar. Mecanicamente, um robot consiste num equipamento formado por Elos (estruturas rígidas) e Juntas (permitem a mobilidade entre elos) que conferem uma flexibilidade de movimentos que não se encontra em outros tipos de máquinas, principalmente se considerarmos que a sua área de trabalho é relativamente grande tendo em conta as dimensões físicas da sua estrutura. As juntas definem-se por 4 tipos:

- Lineares (tipo L): A orientação do movimento do elo de saída é igual à orientação do elo de entrada
- Rotacionais (tipo R): O elo de saída realiza uma rotação em torno da junta, perpendicular a ambos os elos
- De Torção (tipo T): O eixo de rotação é paralelo tanto ao elo de entrada como ao de saída
- Revolventes (tipo V): Semelhante ao anterior, no entanto o eixo de rotação é paralelo apenas ao elo de entrada

Dependendo da aplicação de um robot industrial, este é escolhido principalmente mediante a sua dimensão e capacidade de carga (incluindo ferramentas), precisão e tipo de

movimentos. Segundo a ISO, estes encontram-se assim divididos mecanicamente nas seguintes categorias [5]:

- **Lineares:** Este tipo de robôs possui apenas juntas lineares. São utilizados principalmente em aplicações de pick and place devido à sua repetibilidade e simplicidade. Dentro deste grupo normalmente são subdivididos em:
    - **Cartesianos:** Versão mais comum, programável utilizando o espaço cartesiano.
    - **Gantry:** Semelhantes aos anteriores, mas mecanicamente com maior apoio das juntas. Suportam maiores cargas e por vezes encontram-se associados a robôs articulados na última junta.
- **SCARA:** (*Selective Compliance Assembly Robot Arm*) Possuem 2 juntas rotacionais e 1 linear, e como o próprio nome indica são utilizados principalmente em operações de montagem de equipamentos. São característicos pela sua alta velocidade nos movimentos, mantendo uma boa precisão.
- **Articulados:** Constituídos normalmente por diversas juntas rotacionais, com uma junta de torção na base, são o tipo de robot mais flexível de todos os referidos. A sua utilização passa tanto pelo movimento de objetos, como pela utilização de diversas ferramentas sequencialmente ou em simultâneo aplicadas na última junta. Por vezes a base funciona sobre uma junta linear.
- **Paralelos:** Utilizados principalmente quando se pretende rapidez de movimentos num espaço curto. São constituídos por 3 ou mais pares de elos movidos por juntas lineares ou rotacionais.
- **Cilíndricos**: Constituídos por uma junta resolvente ou de torção e 2 juntas lineares. Utilizados principalmente em aplicações de empilhamento ou movimentação de cargas, caíram em desuso com o avanço dos articulados e o aparecimento dos SCARA.
- **Outros:** É frequente por especificidades da aplicação, serem criados robôs à medida que não se enquadram em nenhuma das aplicações anteriores.
- **Polares:** Não tendo atualmente uma categoria dedicada atribuída pela ISO, faz sentido fazer referência a este visto ter sido o primeiro tipo de robôs industriais a surgir (Unimate) e durante muitos anos o mais utilizado em todo o tipo de aplicações, utilizando controlo por coordenadas esféricas. No entanto com o avanço no desenvolvimento e redução de custos dos articulados, estes caíram em desuso. Eram constituídos por uma junta de torção, uma rotacional e uma linear.

A posição e os movimentos de um robot podem ser referenciados tanto em relação à posição dos seus elos e do órgão terminal relativamente ao seu sistema de coordenadas,

como em relação à posição das juntas sejam estas angulares ou lineares. Havendo uma relação direta entre ambos, é possível abordar o problema da posição e dos movimentos do robot de ambas as perspetivas, tendo ambas a sua importância. Para definirmos esta relação recorremos ao estudo da cinemática do robot.

A cinemática é uma ciência pertencente à física clássica com aplicação em diversas áreas. Esta resume-se ao estudo dos movimentos de corpos ignorando as forças que os geram. [10] Na robótica, permite a partir das características geométricas dos corpos, relacionar a interação no espaço e no tempo entre os diferentes componentes do robot.

- **Cinemática Direta**: Para conhecer a posição do robot no espaço de coordenadas, a partir da posição das juntas é realizada uma modelação do robot onde é estabelecida a relação entre todas as juntas partindo do referencial inicial do robot (normalmente na base – 1ª junta). O método de Denavit–Hartenberg criado em 1955, é utilizado para resolver este problema estabelecendo uma série de regras de orientação dos referenciais das juntas ao longo esqueleto do robot, de modo a poder-se adquirir um conjunto de 4 parâmetros por junta, parâmetros esses que contêm a informação dos ângulos e distâncias entre os respetivos referenciais (tabela de parâmetros D-H). Com esses parâmetros é possível então gerar matrizes homogéneas que permitem relacionar facilmente a posição do órgão terminal com a posição das juntas. Uma matriz é equivalente a um conjunto de matrizes-transformação referentes às rotações e translações executadas para ir de um referencial até ao seguinte. Multiplicando as diversas matrizes homogéneas, obtém-se uma relação entre o referencial inicial e o referencial associado ao órgão terminal. Esta análise é igualmente importante para evitar que haja impactos entre os elos do robot e outros objetos durante os vários movimentos das juntas.

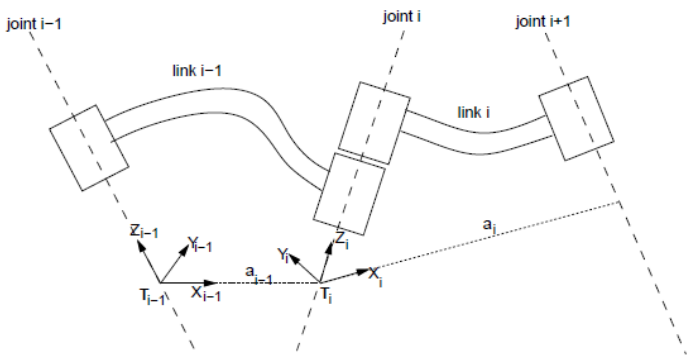


Figura 7. Exemplo de diagrama para determinação dos parâmetros D-H

- **Cinemática Inversa**: Para enviar o robot para um determinado conjunto de coordenadas, é necessário calcular a posição em que as juntas vão estar quando atingirem

a posição final. Embora este problema pareça ser resolvido simplesmente invertendo o procedimento da cinemática direta, neste caso existem mais variantes. Em primeiro lugar, dependendo do robot em causa, é relativamente fácil existirem múltiplas posições de juntas para o mesmo ponto do órgão terminal (singularidade). A quantidade de casos possíveis aumenta rapidamente ao aumentarmos o número de juntas existentes, tornando o cálculo bastante complexo. Existe também a possibilidade de os resultados colocarem o robot em posições impossíveis, por exemplo com elos sobrepostos, ou fora da área de trabalho. As soluções são várias. É possível simplesmente partir da análise da matriz de juntas, ou realizar uma análise geométrica do robot, embora ambos os casos exijam métodos adicionais para evitar os problemas referidos. Outras soluções passam por definir as posições finais relacionando-as com as atuais, o que permite indicar os movimentos mais curtos que o robot necessita de fazer.

No entanto corre-se o risco à mesma de o robot entrar em estados de alinhamento de juntas onde existem várias soluções para o mesmo movimento. A solução passa assim por entrar em conta também com as velocidades do robot, utilizando equações diferenciais (através do Jacobiano). A isto chama-se de **Cinemática Diferencial**.

Embora como já referido, a cinemática não lide com as forças inerentes ao sistema, é também importante para o estudo das mesmas. Dependendo da posição em que o robot se encontra a cada momento, o centro de massa tanto do robot completo como do agrupamento dos vários elos, vai variando. Isto resume-se a uma variação constante das forças que são necessárias exercer em cada uma das juntas pelos acionamentos. Existem diversos métodos desenvolvidos para analisar esta dinâmica do robot. Estes dividem-se também em **Dinâmica Direta** e **Dinâmica Inversa** [10], em que o primeiro aborda a resposta em movimentos resultante da aplicação de forças nos eixos do robot, e o segundo determina as forças necessárias para a realização de determinado movimento.

Por fim, para realizar o controlo de trajetórias do robot com diversos pontos, recorre-se a um dos 3 métodos:

- **Ponto-a-Ponto (PTP)**: Recorrendo diretamente à cinemática inversa, neste tipo de trajetória, o robot tenta mover o órgão terminal para o ponto-destino utilizando a trajetória mais curta possível. É no entanto, de realçar que para o utilizador, existe alguma aleatoriedade perigosa neste procedimento, visto que nem sempre é previsível o movimento que o robot vai executar.
- **Trajetória Contínua (CP)**: Aplicando este modo, são gerados diversos pontos intermédios que depois são associados em sequência. Embora o movimento acabe por se tornar mais lento do que no PTP, isto traz várias vantagens sendo a principal a possibilidade de definir trajetórias complexas. Desta forma consegue-se forçar caminhos para os movimentos do robot, e inclusive realizar outras operações (por

exemplo em ferramentas) durante um movimento. Por vezes os controladores dos robôs permitem interpolar curvas sobre os pontos, tornando os movimentos mais fluidos e rápidos.

- **Trajetória Controlada**: Desta forma o robot segue uma trajetória geométrica (linear, circular, …), controlada entre dois pontos. Embora os movimentos sejam mais lentos do que no caso do PTP, a trajetória é completamente previsível o que torna os movimentos mais seguros. No entanto, existe o risco de surgirem problemas com singularidades durante os movimentos, ou com movimentos de juntas impossíveis de executar em série. Este é um tipo de controlo bastante utilizado por exemplo em robôs com funções de pintura visto que nestes casos a trajetória e a velocidade constante são fatores cruciais.

2.4. Processamento de imagem

O facto de um sistema de automação receber dados de múltiplos sensores dos mais variados tipos, apenas significa alguma coisa se o equipamento estiver preparado para analisar essa informação. Já o ser humano fá-lo de forma bastante eficiente com todos os seus "elementos sensores". Um deles, o olho humano, de nada serve isolado. No entanto, ligado a uma rede de comunicação baseada em neurotransmissores, permite que o cérebro possa cruzar a informação capturada com o conhecimento de milhares de anos de evolução, para que saiba que a peça está com uma determinada orientação, numa localização que pode até ter múltiplos pontos de referência. Um sistema "automático" não possui esse conhecimento todo de raiz, pelo que para funcionar é necessário definir técnicas muito mais simples que permitam à tecnologia atual tomar decisões baseadas nos objetivos definidos para esse mesmo sistema de acordo com a informação recebida.

É frequente fazerem-se comparações diretas entre o olho humano (Figura 8) e uma câmara de captura de imagem, e de facto as suas estruturas assentam em princípios de funcionamento semelhantes. A luz ao chegar ao olho atravessa em primeiro lugar a córnea que se comporta como uma lente fixa, orientando a luz para o Cristalino (em inglês *lens* = lente) que tem a função de focagem da imagem (lente móvel). A pupila que se localiza sobre o cristalino, é adaptada pela íris de acordo com a intensidade de luz recebida, trabalhando como a abertura de uma "lente". Por fim surge a retina que funciona como elemento sensor, dividida em dois tipos de células: os "bastonetes" que funcionam como recetores de luz monocromática, e os "cones" que são responsáveis pela identificação das cores.

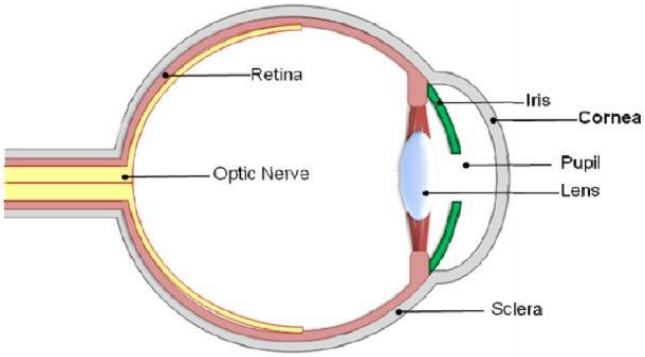


Figura 8. Anatomia do olho humano [11]

A digitalização das câmaras a partir da década de 80, criou novas áreas de desenvolvimento ao permitir que as imagens capturadas fossem armazenadas e modificadas digitalmente a qualquer momento, o que trouxe grandes vantagens relativamente a todos os elementos sensores existentes até à data. Com o aparecimento das técnicas de processamento de imagem passou a ser também possível determinar novas características do meio físico através de métodos não intrusivos, trazendo novas aplicações às câmaras tais como:

- Indústria:
  - Controlo de qualidade
  - Identificação de códigos
  - Deteção de objetos
  - Guiamento de robôs
- Automobilismo:
  - Controlo de trajetórias (auto-piloto)
  - Previsão de colisões
  - Transporte autónomo
- Segurança:
  - Deteção de pessoas
  - Controlo de acessos
  - Previsão de movimentos
  - Análise facial
- Agricultura:
  - Deteção de pragas
  - Deteção de alimentos estragados
  - Apoio na programação de cultivos

- Medicina:
  - Identificação de células
  - Análise de tumores
  - Modelação 3D de órgãos

Sendo a visão computacional uma tecnologia que habitualmente requer um nível elevado de processamento para atingir os desempenhos esperados, a sua evolução tem estado diretamente relacionada com o avanço da eletrónica e das tecnologias de computação. Ainda assim, embora grande parte da investigação sobre o tema resulte em algoritmos complexos, muitas vezes com uma componente relativamente grande de matemática, difícil de processar, têm-se conseguido desenvolver métodos que simplificam o esforço dos controladores ao aplicar estes mesmos algoritmos. As abordagens de tratamento matricial das imagens, trabalhando-as como agrupamentos lógicos de dados, foi um passo importante neste campo, visto facilitar a relação com a organização das memórias computacionais convencionais. O processamento de imagem envolve o tratamento de grandes quantidades de dados a alta velocidade (aplicação de inúmeros processos por cada *frame* de vídeo), e sendo esta uma necessidade comum à inteligência artificial, a evolução de algoritmos mais complexos tem passado por tirar o melhor partido de cada uma das duas áreas, de modo permitir abordagens mais preditivas, e rápidas no momento de trabalhar e analisar os dados.

No mercado existem diversas soluções comerciais tanto para captura como para processamento de imagem. As empresas direcionadas para a produção de elementos sensores, focam os esforços no desenvolvimento de elementos cada vez mais compactos, mas em simultâneo com melhores resultados, tentando ir de encontro às necessidades de cada público. Uma solução industrial por norma não tem os mesmos requisitos que um fotógrafo profissional. Enquanto que numa máquina com um ambiente relativamente controlado, são valorizados a capacidade de atingir grandes detalhes geométricos a grande velocidade, uma fotografia num ambiente aberto é favorecida por uma melhor composição das cores e maior variação de contrastes, e embora o tempo de captura seja sempre importante, comparativamente ao primeiro caso, o tempo de pós-processamento pode até ser alongado em favor de um melhor resultado. As soluções mais típicas na indústria passam por:

- **Câmaras com funções integradas:**

Para funções em que é necessário realizar a deteção de cores, realização de medições, deteção de formas simples, ou apenas a leitura de códigos de barras, é comum recorrer-se a câmaras que já incorporam algumas funções básicas (Figura 9), configuráveis

por software dedicado. Por norma estes aparelhos trabalham com baixas resoluções e por isso a distâncias curtas.

Figura 9. Exemplo de câmara com funções dedicadas de processamento de imagem [12]

- **Controladores:**

Aplicações mais complexas em que se exija maiores performances e maior quantidade de ferramentas obrigam à utilização de controladores ou computadores dedicados (Figura 10). Estes permitem também a ligação de múltiplas câmaras a fim de trabalhar com soluções mais complexas. Quando a velocidade é crítica encontram-se também sistemas com módulos de processamento de imagem independentes e dedicados, com processadores ou FPGA's (*Field-Programmable Gate Array*).

Figura 10. Exemplo de controlador dedicado para funções de processamento imagem [13]

- **Software:**

Determinadas situações mais específicas exigem o desenvolvimento de algoritmos dedicados e flexíveis. Realçar e identificar grupos de células na medicina (Figura 11), ou detetar grupos de pessoas em movimento em ambientes abertos, são alguns exemplos em que existem soluções mais complexas devido tanto às dinâmicas dos objetos a analisar como ao ambiente em que se encontram. Nestes casos é possível também recorrer-se a livrarias de imagem dedicadas quer para pequenas filtragens de imagem, quer para utilização de ferramentas mais complexas.

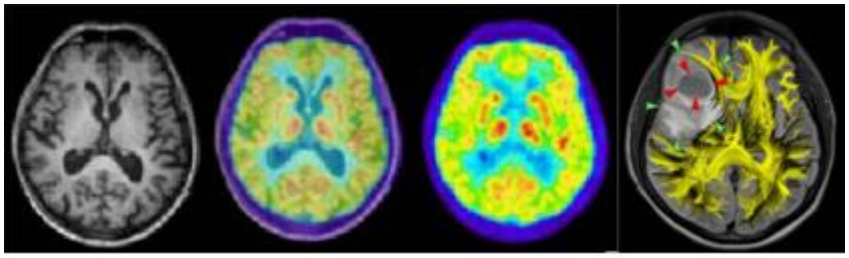


Figura 11. Exemplo de utilização de processamento de imagem na medicina [14]

Uma das ferramentas mais relevantes que surgiram relativamente à visão computacional foi o acompanhamento de objetos, o qual consiste na deteção de objetos e consequente acompanhamento dos seus movimentos ao longo do tempo, ou seja, ao longo das várias *frames* de um vídeo, seja este gravado ou em tempo real. Sendo uma tarefa importante em visão, também é uma das mais difíceis de realizar. Existem diferentes variáveis que dependendo da aplicação podem levar à necessidade de aplicar técnicas diferentes com níveis de complexidade que têm de se manter robustas ao longo do tempo. Podemos identificar as seguintes situações (entre outras):

- Alterações de luminosidade
- Movimentos não lineares
- Fundo variável
- Deslocamento da câmara
- Existência de objetos semelhantes
- Proximidade de objetos
- Ocultação temporária
- Rotação dos objetos (não uniformes)
- Deformação dos objetos

Alguns destes problemas são de relativamente fácil resolução, como por exemplo a luminosidade, onde a sua compensação pode ser feita tanto por sistemas externos (iluminação forçada) como por software/hardware através de algoritmos de compensação automática de brilho e exposição de captura de imagem. Ambas as soluções reduzem o esforço do sistema de acompanhamento ao fazer o papel dele, mas podem também criar outras dificuldades. Um vídeo que esteja a ser capturado numa zona onde existe uma grande influência da luz solar tem de ter também em conta a variação da cor, e uma compensação da imagem completa para colmatar esta variação, pode influenciar também os dados que se pretendem adquirir dos objetos. Já uma aquisição de imagem numa zona ampla, mas onde se pretende ter uma boa resolução de imagem com poucos recursos, pode levar à utilização de câmaras com sistemas de orientação motorizados para evitar alargar a área de captura afastando a camara, no entanto, tal vai resultar também na alteração do fundo da imagem.

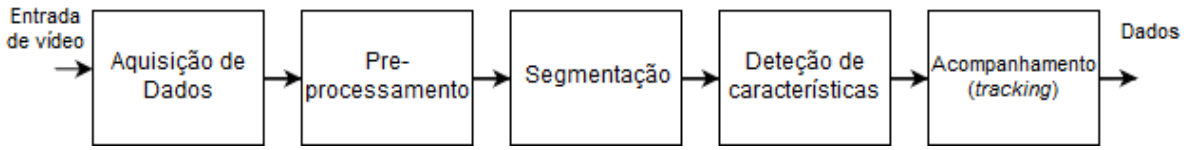


Figura 12. Arquitetura de um sistema típico de acompanhamento de objetos [15]

Observando a Figura 12 onde temos o modelo de uma aplicação típica de acompanhamento de objetos, podemos observar que o acompanhamento (*tracking*) é apenas a última fase de uma cadeia de processamento de informação. Atualmente, cada um dos blocos referidos, pode possuir diversos algoritmos que visam passar a informação para o passo seguinte rapidamente e com a menor quantidade possível de dados não relevantes (ruído). Durante a aquisição de imagem e pré-processamento, o foco principal é obter o máximo de dados sobre o mundo físico, sendo estes tratados geralmente de modo automático para compensar fenómenos previsíveis. Já nas restantes fases o processamento é orientado à aplicação, necessitando assim de calibrações para que os algoritmos utilizados sejam o mais fiéis possíveis aos objetivos pretendidos.

2.4.1. Aquisição de imagem

A luz visível, ou seja, identificável pelo olho humano, por definição, trata-se da radiação eletromagnética que se encontra no respetivo espectro, na gama de largura de bandas entre 400nm e 700nm [16], indo desde o violeta até ao vermelho respetivamente (Figura 13). É frequente fazer-se referência também à "luz ultravioleta" e à "luz infravermelha". Embora a retina não receba a radiação que se encontra na gama ultravioleta (10nm a 400nm) visto ser filtrada pela córnea, é conhecido que entre os 310nm e os 400nm a radiação reage com diversos materiais gerando luz visível (violeta). Já na gama infravermelha, apenas é visível a luz vermelha no limiar dos 700nm, largura de banda a partir da qual são utilizados por exemplo, equipamentos para deteção de radiação térmica.

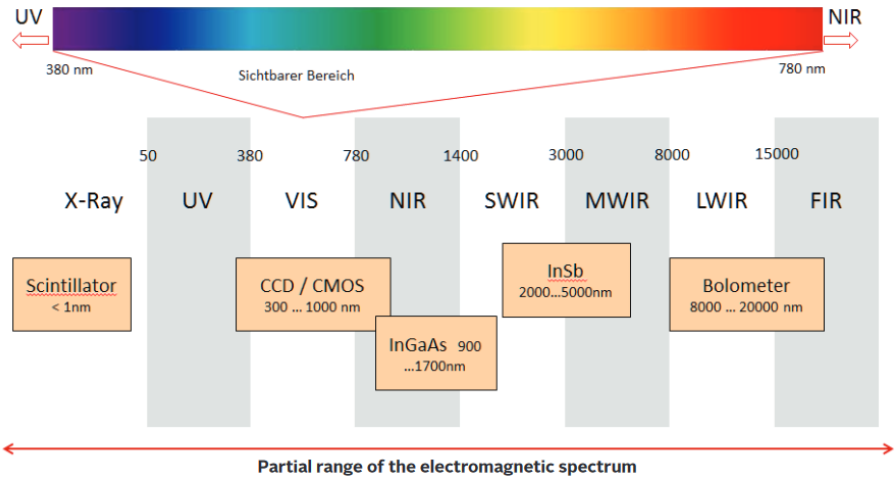


Figura 13. Diversas tecnologias de sensores para captação do espectro luminoso [17]

Enquanto que o olho humano tem as limitações referidas, já numa câmara digital é possível ir além do espectro de luz visível (Figura 13). Tal deve-se aos elementos sensores capturarem apenas a energia emitida pelas fontes de luz sem qualquer diferenciação.

Normalmente o elemento sensor de uma câmara, é formado por uma matriz de condensadores MOS (M*etal-Oxide-Semiconductor*), que armazenam carga ao serem projetados pela radiação luminosa. No entanto toda a matriz é uniforme, sendo necessário isolar cada uma das cores para obter imagens dentro do espectro acima referido. Esta operação pode ser feita tanto através de matrizes de filtros de cores visíveis ou não visíveis (CFA - *Color Filter Array*), colocadas sobre a matriz de condensadores, como através de lentes prismáticas que separem naturalmente a luz e a encaminhem para elementos sensores separados (Figura 14**).**

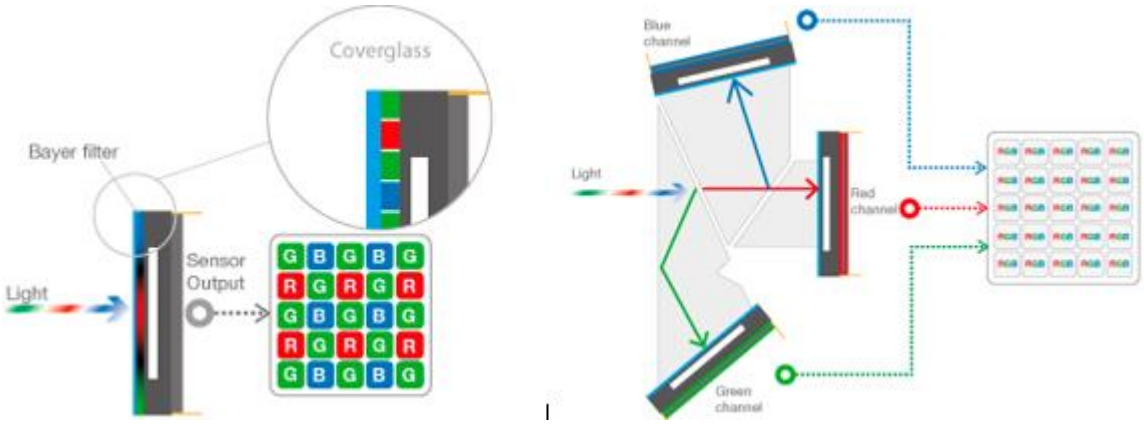


Figura 14. Métodos de separação de cores: Filtro matricial (esquerda), Lente prismática (direita)

A forma como essa carga armazenada é tratada depende da tecnologia em uso, sendo as duas principais a CCD e a CMOS:

- **CCD (*Charge-Coupled Device*):** Visto os condensadores estarem ligados entre si e organizados em linhas verticais (Figura 15), ao aplicar-se uma tensão entre os mesmos, é possível transmitir a carga acumulada ao longo dessas linhas. Após a captura da luz por estes elementos, esta operação é realizada várias vezes até se transmitir toda a carga existente na matriz até a uma zona responsável pela conversão A/D (Analógico-Digital) dos sinais recebidos, transformando-os depois em dados. A grande vantagem dos sensores CCD está na qualidade de imagem devido ao facto de a carga ser capturada de forma uniforme ao longo de toda a matriz não existindo variações na captura de carga entre cada célula.
- **CMOS (*Complementary Metal-Oxide-Semiconductor*):** Ao contrário da CCD neste caso a carga de cada célula da matriz é convertida localmente, acelerando o processo de captura o que também reduz o custo energético do sensor. É também possível realizar algumas correções em pré-processamento recorrendo à eletrónica para limitar a quantidade de carga tratada em cada célula como no caso do processamento HDR (*High Dynamic Range*) [18]. No entanto pode provocar disparidades entre a carga capturada e o sinal de saída do conversor A/D devido a pequenas diferenças existentes entre os componentes ao

longo da matriz. Outro inconveniente está na menor área de captura de imagem devido à matriz ser também ocupada pelos elementos conversores (Figura 15).

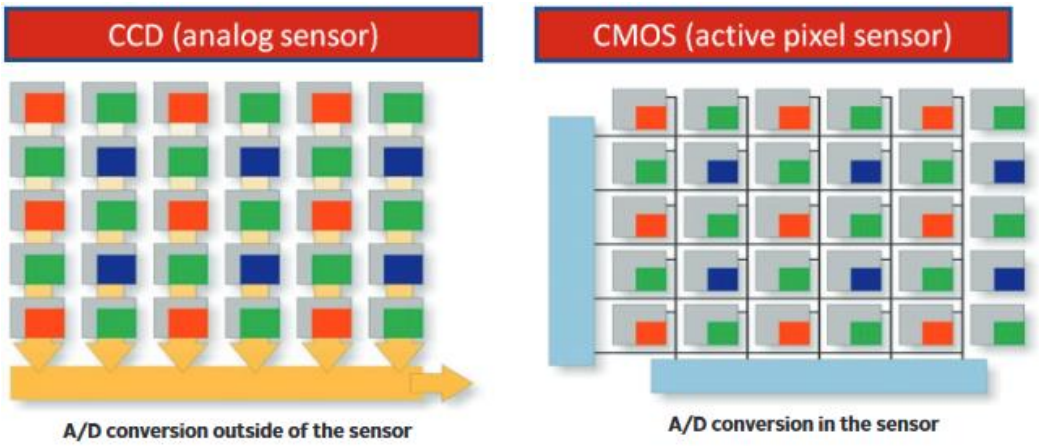


Figura 15. Comparação entre as tecnologias de sensores CCD e CMOS [17]

2.4.2. Processamento digital

Após a captura de sinais por parte do sensor e o respetivo acondicionamento de sinal, é formada uma imagem digital que, no entanto, ainda não está pronta para ser entregue pela câmara. Em primeiro lugar, a imagem capturada pelo sensor, vem dividida em camadas de cores diferentes resultantes da CFA (*Color Filter Array*), com cores isoladas em pontos vizinhos. Assim, é necessário utilizar um algoritmo de *demosaicing* que como o próprio nome indica, vai relacionar todos os pontos existentes de forma a eliminar o "efeito de mosaico". Outra das correções essenciais a realizar passa por compensações na imagem devido aos efeitos das curvaturas das lentes, e filtragens quer por ruido quer devido a variações na matriz do sensor CCD/CMOS (Figura 16). Após este passo, a informação é transmitida para armazenamento ou posterior tratamento de dados.

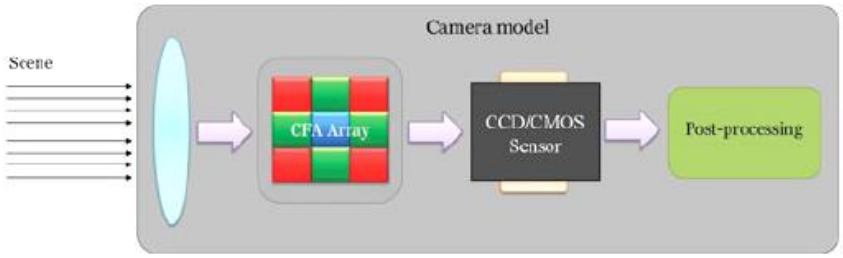


Figura 16. Esquema simplificado da cadeia de aquisição de imagem de uma câmara CCD/CMOS [19]

Uma imagem digital não é mais do que uma matriz de pontos denominados de “Pixel” (do inglês *Picture [Pix] Element*). A dimensão de cada pixel define a divisão cromática dentro de cada cor, sendo as mais usuais 1 bit (Preto e Branco) e 8 bits (Escala de Cinza) de forma a facilitar a computação de imagem sem se perder a perceção da variação linear das cores.

De acordo com a forma como as cores são visualizadas e processadas, é então possível multiplicar essa matriz pelo número de cores básicas a utilizar (espaços de cores). O espaço de cores mais utilizado é o RGB (*Red, Green, Blue*) devido à facilidade com que as cores são representadas em ecrãs, e à facilidade de relacionar as mesmas com o espectro luminoso e com a visão humana. No entanto existem outros espaços os quais são utilizados mediante a especificidade da aplicação. O CMYK (*Cyan, Magenta, Yellow, Key*), é utilizado principalmente quando o objetivo é a utilização de equipamento de impressão visto ser possível obter-se variações de cores adicionando qualquer um dos componentes em papel branco. Já o HSV (*Hue, Saturation, Value*) é utilizado em aplicações de imagem para ajudar na diferenciação das cores tal como o ser humano as identifica (variando o *Hue* = Tonalidade da cor).

A conversão de um espectro de cores para escala de cinza embora possa ser considerada direta, tem em conta uma série de parâmetros perceptuais da maneira como o olho humano visualiza o espectro de cores e de como as mesmas são reproduzidas por equipamentos de visualização. Existindo assim várias variáveis a serem consideradas, a expressão que mais se aproxima da realidade, e que diversos sistemas adotam por defeito, resulta da recomendação ITU-R BT.601, da *International Telecommunication Union – United Nations* [20], que é a seguinte:

$$Y' = 0.299\mathrm{R}' + 0.587\mathrm{G}' + 0.114\mathrm{B}' \qquad (1)$$

sendo Y’ a quantidade de luz monocromática (luminância em $\frac{cd}{m^2}$) por pixel e R’, G’, B’ a mesma intensidade nas cores Vermelho, Verde e Azul respetivamente. De forma bastante intuitiva, conseguimos aperceber-nos na Figura 17 da diferente dificuldade de visualizar os números em preto e em branco em cada uma das cores RGB. A intensidade luminosa é igual em cada uma das cores representadas.

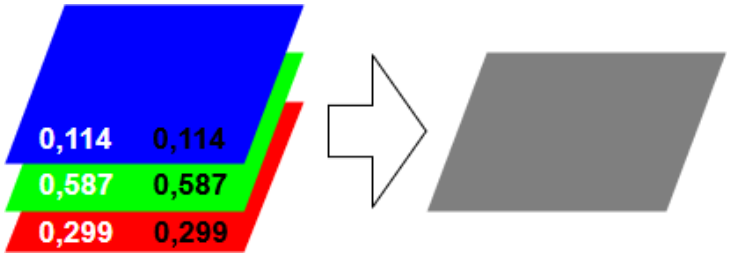


Figura 17. Relação na conversão RGB para Escala de Cinzas segundo ITU-R BT.601

Trabalhar com uma grandeza que indica a quantidade de luz, faz com que a escala de cinza (Figura 19) tenha como limites o preto a 0% e o branco a 100%. Passando por uma conversão de 8-bit, surge uma escala que vai de 0 a 255 tons de cinza respetivamente (Figura 19).

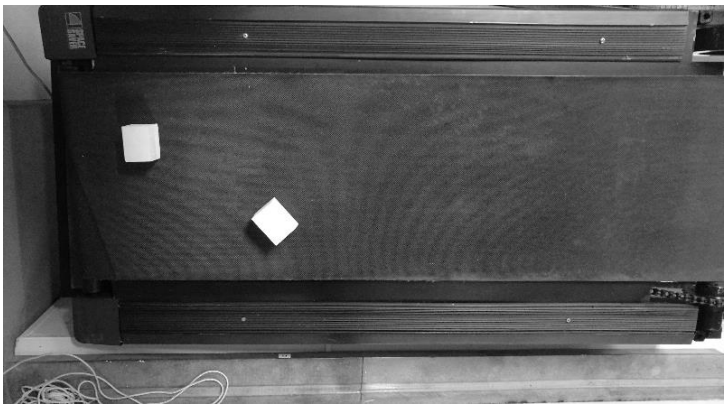


Figura 18. Peças sobre transportador (conversão RGB para Escala-Cinza)

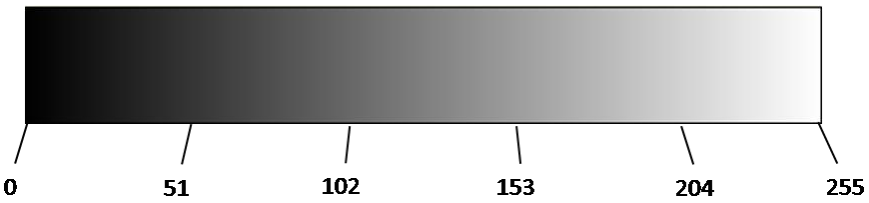


Figura 19. Escala de cinzas a 8-bit

O facto da informação recebida pelas câmaras ser tratada de forma matricial (Figura 20) permite que sejam utilizados algoritmos computacionais quer sequenciais, quer matemáticos, para processar a informação contida nas imagens com um elevado desempenho. No entanto, o desenvolvimento desses mesmos algoritmos tem por base na maior parte dos casos a análise da informação do ponto de vista do processamento de sinais contínuos, o que permite trabalhar essa mesma informação no sentido de a aproximar do espaço físico utilizando ferramentas já conhecidas.

Nesta fase, a operação de convolução tem uma enorme importância, pois utilizando matrizes de convolução (tipicamente 3x3, $n = 3$), também conhecidas na bibliografia por "máscaras" conseguimos facilmente aplicar os mais diversos filtros ao longo de toda a imagem $h(x, y)$ utilizando a mesma matriz. A convolução sendo uma operação que quando convertida para sinais discretos trata-se de um conjunto de somatórios (dependente do número de variáveis espaciais utilizadas), é facilmente implementada computacionalmente com recurso a ciclos [21]:

$$y(x,y) = h * w = \sum_k \sum_l \; h(x,y)w(x-k, y-l) \qquad (2)$$

As máscaras são compostas por um pixel central $W_{(x,y)}$ e por uma vizinhança $W_{(x\pm n, y\pm n)}$ tal como se observa na Figura 20.

W_1 $(x-1, y-1)$	W_2 $(x, y-1)$	W_3 $(x+1, y-1)$
W_4 $(x-1, y)$	W_5 (x, y)	W_6 $(x+1, y)$
W_7 $(x-1, y+1)$	W_8 $(x, y+1)$	W_9 $(x+1, y+1)$

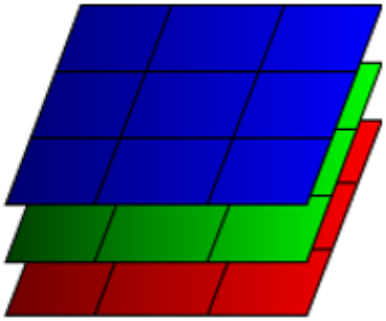


Figura 20. Matriz de pixéis (esquerda) e agrupamento de matrizes RGB (direita)

2.4.3. Filtros digitais

De modo geral, o conceito de filtragem, implica a redução ou eliminação de informação que não é necessária num sistema. Numa aplicação que envolva a deteção de objetos a informação mais comum de ser utilizada pelos métodos existentes para identificação dos mesmos é a dos contornos [22] (ver ponto 2.4.4), sendo analisada apenas uma matriz de dados com informação monocromática. Embora em determinados casos a análise de espaços policromáticos seja imperativa, geralmente consegue-se ótimos resultados simplificando. No entanto, é necessário que tal informação seja entregue o mais limpa possível, e que essa filtragem de conteúdos seja determinística o suficiente para que eventuais ruídos isolados não possam amplificar negativamente quaisquer variações na imagem.

O conjunto de toda a cadeia de aquisição de imagem é imperativa neste ponto. Quanto maior for a resolução da câmara, mais informação de imagem se recebe, o que não significa propriamente que seja informação com boa qualidade. Não sendo tema a desenvolver nesta dissertação, se todos os equipamentos desde a lente da câmara até ao algoritmo de geração dos dados de imagem, não forem devidamente adequados entre si, o resultado de aumento de resolução é apenas mais ruido amplificado em cada etapa do processo da captura.

Após a apresentação do conceito de filtragem de imagem, seguem-se assim alguns dos filtros mais comuns:

- **Mediana**

A aplicação da mediana como método de filtragem (Figura 21, Figura 22 e Figura 23), tem especial utilidade quando nos deparamos com ruido gaussiano ou estatístico, i.e., ruido presente ao longo de toda a imagem de forma não linear, mas que acompanha a evolução do conteúdo da mesma. A vantagem da aplicação, está na relação entre pixéis em que se

consegue eliminar pixéis fora do contexto a partir da evolução da vizinhança ao longo de toda a imagem. Temos, no entanto, a desvantagem de perder detalhes da imagem quanto maior for o índice *K* da mediana, o qual corresponde ao tamanho da vizinhança.

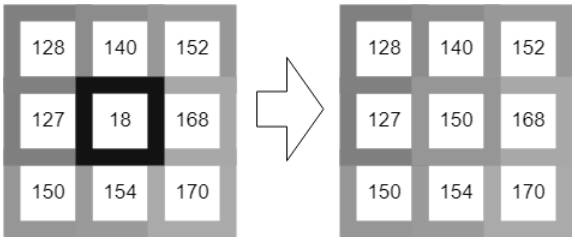


Figura 21. Cálculo matricial da mediana

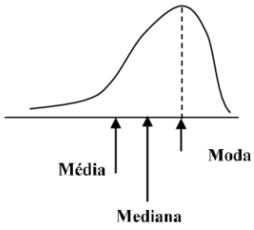


Figura 22. Curva de Gauss

Figura 23. Cálculo matricial da mediana (agrupado)

Para a determinação do *K* a aplicar temos de conseguir definir a dimensão do que vai ser considerado indesejado na nossa imagem. Olhando atentamente para as figuras (Figura Figura 24 e Figura Figura 25) conseguimos ver com mais detalhe as variações que existem ao longo do tapete e que podem ser consideradas ruído podendo ser reduzidas com a aplicação da mediana como se observa na Figura 26.

Aplicando o filtro mediana (K = 21):

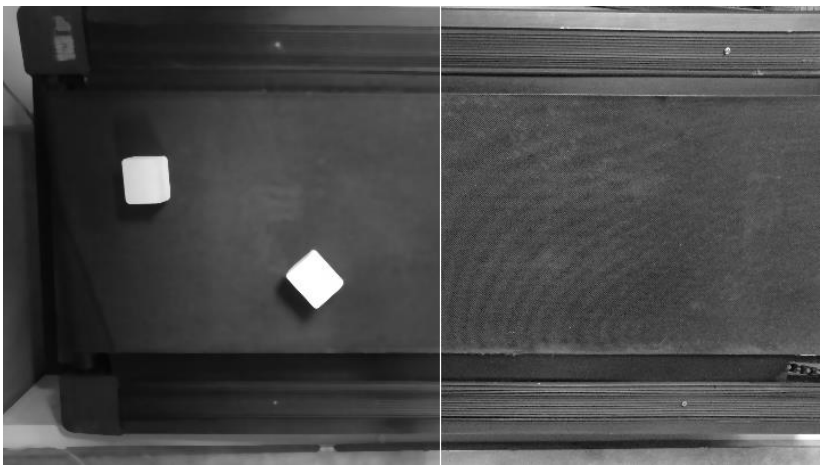


Figura 24. Imagem antes (direita) e após (esquerda) a aplicação do filtro mediana

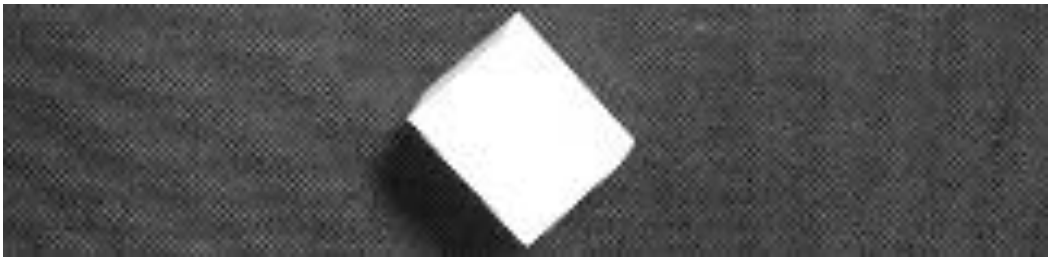


Figura 25. Imagem antes do filtro mediana (recorte)

Figura 26. Imagem após o filtro mediana (recorte)

- **Operações morfológicas**

As operações morfológicas em processamento de imagem consistem no estudo matemático de formas presentes em imagens de modo a modifica-las com recurso a operações algébricas. A principal vantagem desta abordagem, é a possibilidade de quantificar os resultados obtidos destas modificações permitindo posteriores análises da informação contida na imagem. A aplicação da morfologia matemática em processamento de imagem é de extrema utilidade para, entre outros:

- Simplificação de formas
- Filtragem
- Reconhecimento de caracteres (*OCR - Optical Character Recognition*)

Como exemplo, tal como representado na Figura 27, consideremos uma imagem binária pertencente a um espaço euclidiano H^2, na qual existe um agrupamento de pixéis vizinhos $p(x,y)$ de valor igual, o qual consideramos um subconjunto $G \subseteq H$. Se colocarmos uma nova forma $B \subseteq H$ (chamada de prova) a interagir com G (operação), conseguimos ter como resultado uma forma S a qual nos pode permitir extrair novas informações sobre a forma inicial G. A dimensão e o formato da nossa *prova* vai depender da forma como pretendermos modificar e/ou analisar a informação contida na imagem.

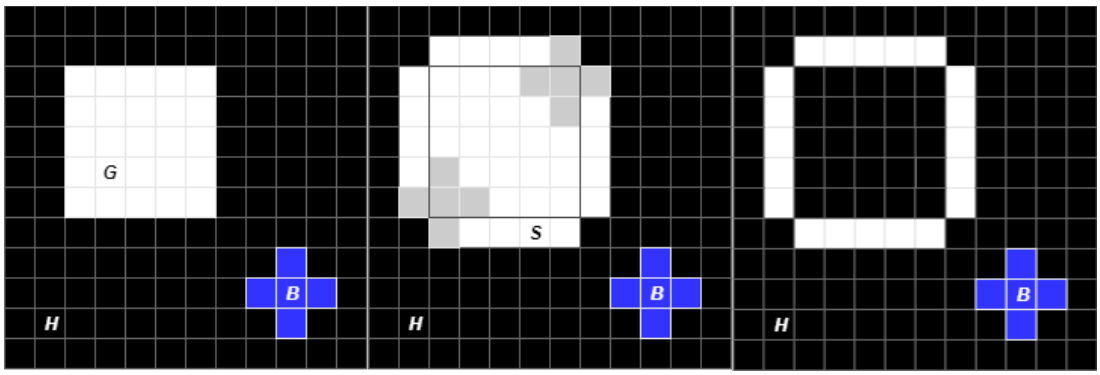


Figura 27. Exemplo de extração de contornos exteriores com recurso a operação morfológica (dilatação seguido de subtração da imagem original)

A aplicação das operações morfológicas, implica aumentos significativos do tempo de processamento dependendo do tamanho/formato da prova. No entanto a aplicação de outras técnicas de processamento numa imagem não tratada adequadamente pode provocar o aparecimento de resultados imprevistos e falsamente válidos.

- **Limiarização (*Thresholding*)**

O processo de Limiarização, consiste na conversão do um valor de um pixel de uma escala de 0 a 255 para um valor binário, i.e. verdadeiro ou falso / branco ou preto como se pode observar na Figura 28. A aplicação desta técnica embora parta de um conceito bastante simples é amplamente utilizada nas aplicações de processamento de imagem, e serve mesmo de base para várias aplicações que implicam segmentação de dados (agrupamento de dados) tais como reconhecimento de caracteres e de objetos, compressão de dados, pesquisa de imagens, entre outros. Por exigir um baixo nível computação, é facilmente aplicada no seu modo mais simples:

$$\boldsymbol{B}(\boldsymbol{x},\boldsymbol{y}) = \begin{cases} max, & I(x,y) \geq T \\ 0, & I(x,y) < T \end{cases} \quad (3)$$

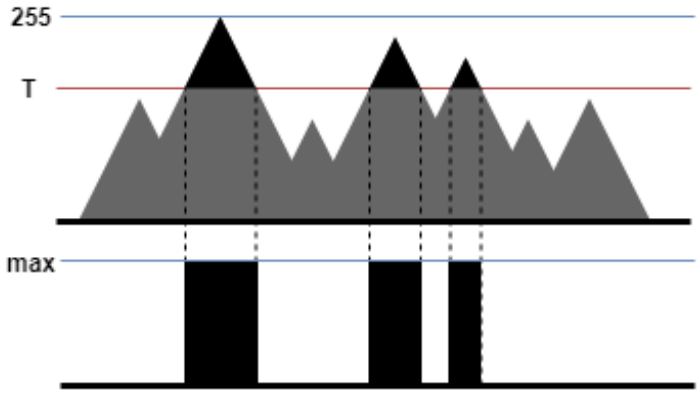


Figura 28. Exemplo de Limiarização

Existem, no entanto, diversas técnicas de Limiarização mais avançadas que aplicando métodos estatísticos, desenvolvem uma alteração do nível de *threshold* ao longo de toda imagem, compensando evoluções ao longo da imagem completa (por exemplo, no fundo). Chama-se a isto Limiarização Adaptativa.

A melhor maneira para determinar o índice T a aplicar neste caso é fazendo uma análise ao histograma aplicado à Figura 29. Devido ao exemplo em causa referir-se ao projeto, a análise irá ser apenas feita à zona do tapete, pois é a única área que nos interessa analisar durante a execução do programa.

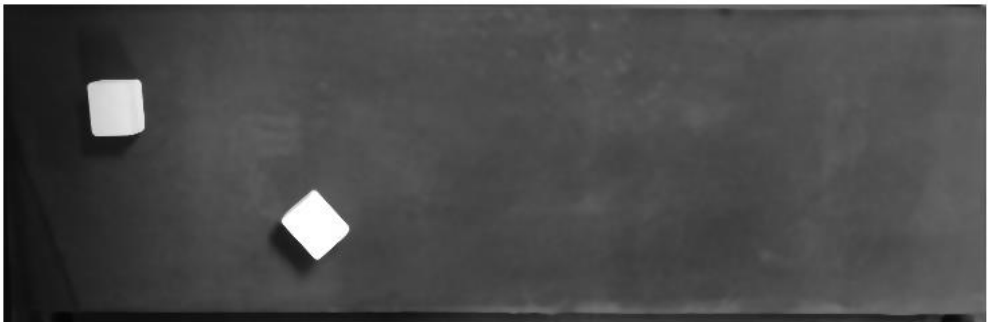


Figura 29. Recorte da zona do tapete

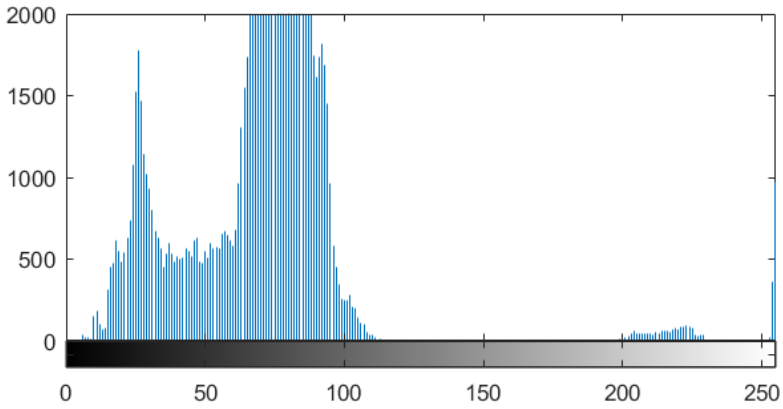


Figura 30. Histograma da figura 28

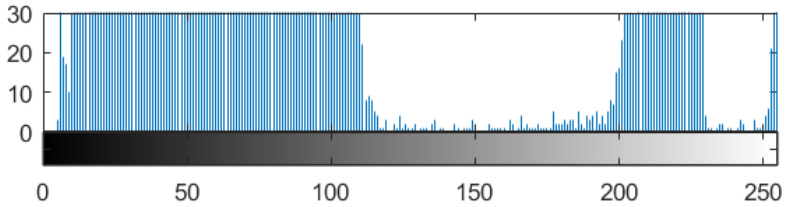


Figura 31. Histograma da figura 26. Pormenor com escala reduzida

A partir da Figura 30, é possível perceber que a maior parte dos "brancos" da imagem, que por sua vez correspondem aos objetos a serem detetados, encontram-se em valores da escala de cinza acima de 200. No entanto na Figura 31 verificamos também que embora em amplitudes muito inferiores, não deixam de existir valores de cinza na escala de cinzentos entre 120 e 200. Embora à primeira vista as peças sobre o tapete sejam "brancas", as sombras nas restantes faces, e o efeito do arredondamento de valores na conversão da imagem devido à desfocagem em escalas perto do pixel, faz com que, as arestas das peças possam acabar por ficar cortadas ao aplicar a Limiarização. Na Figura 32 podemos verificar essa diferença para dois valores diferentes de threshold. Esta diferença de resultados, embora não pareça grande nesta fase, pode forçar diferenças significativas quando se estiver a analisar as características dos objetos. Por exemplo, facilmente a aresta superior do objeto 1 com $T = 200$, pode gerar múltiplas linhas de contorno (ver capítulo 2.4.4 – *Transformada de Hough*) com várias direções que resultem em ângulos diferentes do objeto. Por outro lado,

com $T = 150$, a aresta superior surge muito mais linear. É por estes motivos que em processamento de imagem é comum não se aplicar apenas um filtro, mas sim uma cadeia de procedimentos contínuos de modo a corresponder aos objetivos da aplicação.

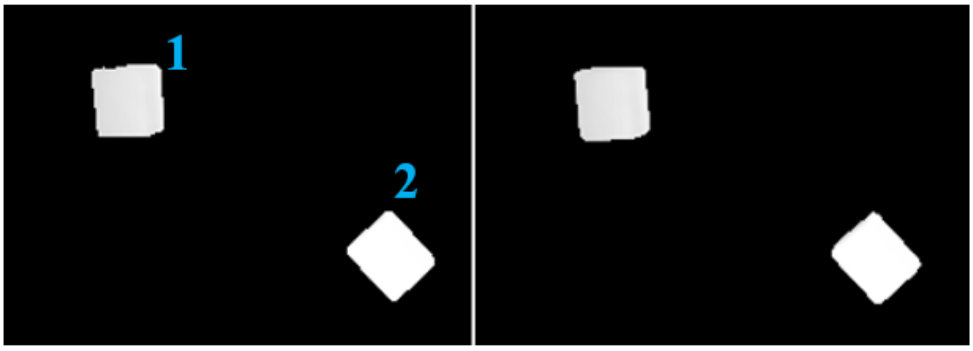


Figura 32. Resultado da Limiarização para valores de T: 200 (esquerda) e 150 (direita)

- ***Detetor de arestas***

Uma das características para identificação de objetos mais usadas em sistemas de visão artificial, são as arestas. Estes traduzem-se em pontos vizinhos com valores de intensidade luminosa semelhantes, próximos de outros pontos com valores bastante diferentes, os quais definem os limiares de formas numa imagem. No entanto, é necessário ter em conta que por vezes estes mesmos podem ser confundidos com outras variações não desejáveis de ser identificadas existentes por exemplo no fundo da imagem, ou simplesmente com ruido. Isto pode resultar no aparecimento de arestas não desejáveis ou com a sua mistura com outros pontos com níveis de gradiente inferiores levando à sua não-existência.

Para resolver este problema foram desenvolvidos vários métodos entre os quais se destacam:

- ***Sobel***

Resulta da convolução da imagem original $\boldsymbol{A}$ com duas matrizes 3x3 $\boldsymbol{G}$, que vão resultar na geração de dois gradientes no sentido do eixo X e do eixo Y. Estes gradientes tratam-se de variações bruscas de luminosidade existentes ao longo da imagem.

$$G_X = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} * A, \; G_Y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * A \qquad (3)\ (4)$$

Para obter o resultado da amplitude aproximada do gradiente em cada pixel temos:

$$G = \sqrt{G_X^2 + G_Y^2} \qquad (5)$$

E para determinar o ângulo do gradiente em cada pixel:

$$\theta = \operatorname{atan}\left(\frac{G_Y}{G_X}\right) \qquad (6)$$

Esta é uma solução relativamente simples de implementar e leve computacionalmente, no entanto de resultados limitados dependendo da sua aplicação. O

facto de se utilizar uma matriz 3x3 aplicada em 2 direções distintas (esquerda-direita e cima--baixo) faz com que a imagem não seja analisada toda simultaneamente e não sejam consideradas variações de luminosidade com tendências de áreas maiores.

- ***LoG*** (Laplacian of Gaussian)

Este método baseia-se mais uma vez na aplicação de um filtro de convolução na imagem, mas gerado a partir da aplicação da transformada de Laplace de modo a obtermos os máximos de gradiente:

$$para\ uma\ vizinhança\ em\ \mathbb{R}^2: \begin{bmatrix} a_0 & a_1 & a_2 \\ a_7 & [i,j] & a_3 \\ a_6 & a_5 & a_4 \end{bmatrix} \quad (7)$$

$$aplicando\ as\ derivadas\ parciais\ \Delta f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \quad (8)$$

$$resultamos\ em,\ \frac{\partial^2 f}{\partial x^2} = f(i,j+1) - 2f(i,j) + f(i,j-1) \quad (9)$$

$$e,\ \frac{\partial^2 f}{\partial y^2} = f(i+1,j) - 2f(i,j) + f(i-1,j) \quad (10)$$

$$obtendo\ assim\ um\ operador: \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad (11)$$

Devido a este método trabalhar com derivadas de 2ª ordem (quadráticas), trata-se de um detetor bastante sensível e que facilmente realça as arestas da imagem, no entanto esta mesma vantagem torna-o altamente sensível a ruído. A forma de ajudar a ultrapassar esta limitação é aplicando um filtro Gausseano em primeiro lugar.

- ***Canny***

Proposto por John Canny em 1986 [23], apesar de já antigo, continua a ser um dos métodos mais utilizados. Trata-se de uma solução completa que procura a resposta ótima ao problema da localização das arestas de uma imagem. Canny abordou em primeiro lugar o problema a 1 dimensão estudando a resposta de um filtro a uma aresta influenciada por ruido gaussiano, e depois alargando para uma imagem bidimensional, tendo como objetivos:

1. Boa deteção: pretendia que as arestas detetadas fossem o mais próximos possível da realidade com uma baixa probabilidade de erros
2. Boa localização: as localizações dos pontos resultantes, deveriam ser o mais próximas possível do centro das arestas
3. Resposta única: não deveriam existir múltiplos resultados para a mesma aresta

1. Dado um filtro com uma resposta $f(x)$ a um impulso, tenhamos $G(x)$ como a aresta da imagem dentro do espaço $[-W, +W]$.

Temos assim a resposta de sinal ideal do sistema:

$$H_G = \int_{-W}^{+W} G(x_0 - x) f(x) dx \qquad (12)$$

, com $x_0 = 0$ (impulso de entrada iniciado na origem / aresta)

Já a resposta do ruído vai-se traduzir em:

$$H_r = \sqrt{n_0^2 \int_{-W}^{+W} f^2(x) dx} = n_0 \sqrt{\int_{-W}^{+W} f^2(x) dx} \qquad (13)$$

,com n_0^2 sendo o valor de pico do ruído por ciclo de sinal.

Temos assim uma razão sinal-ruido (SNR – signal to noise ratio) de:

$$\boldsymbol{SNR} = \frac{\left| \int_{-W}^{+W} G(-x) f(x) dx \right|}{n_0 \sqrt{\int_{-W}^{+W} f^2(x) dx}} \qquad (14)$$

2. Sabendo que quando a primeira derivada de uma função passa por zero temos o máximo da função, podemos deduzir a equação anterior para obter a localização do centro da aresta:

$$\boldsymbol{Localização} = \frac{\left| \int_{-W}^{+W} G'(-x) f'(x) dx \right|}{n_0 \sqrt{\int_{-W}^{+W} f'^2(x) dx}} \qquad (15)$$

3. A fim de evitar a influencia de ruídos que gerassem múltiplos máximos na mesma aresta concluiu que:

$$x_{máx}(f) = k.W \qquad (16)$$

, onde $x_{máx}$ é a distância entre deteções de arestas próximas, e k uma constante que permite limitar a quantidade de falsos contornos. Analisando os resultados de várias respostas do sistema, quanto mais longo fosse o gradiente da aresta maior era $x_{máx}$.

A implementação analógica do detetor passaria pela multiplicação das equações do *SNR* com a *Localização*. No entanto o cálculo era demasiado complexo para ser implementado diretamente de forma analítica. Assim criou um algoritmo alternativo que em imagens 2D resume-se a:

1. Aplicação de um filtro Gaussiano para redução de ruído
2. Aplicação do filtro de Sobel em X e Y para determinar a localização e orientação de gradientes (limites)
3. Arredondamento dos ângulos dos gradientes em 45º e posterior utilização do *"Non-Maximum Surpression"* (comparação dos pontos adjacentes em 180º e manter apenas o que tiver uma amplitude maior)

4. Aplicar um threshold duplo. Pixéis com peso superior ao threshold alto são marcados como “forte”, inferiores ao threshold baixo são eliminados, e os intermédios marcados como “fraco”
5. Análise por histerese: Os pontos marcados como “forte” são considerados como arestas certas. Os marcados como “fracos” são considerados parte das arestas apenas se tiverem em contacto com as anteriores, caso contrário são eliminados.

Finalizando a aplicação de um ou de vários filtros destes, pode-se começar a analisar a informação resultante, do ponto de vista da extração das características dos objetos, com vista à sua futura identificação.

2.4.4. Identificação das características de objetos

Ao longo do tempo, o ser humano foi adquirindo inconscientemente a perceção das características dos objetos que o rodeiam e com que interage e desenvolveu métodos cognitivos de o fazer. Facilmente olhamos para o objeto do dia-a-dia e, identificamo-lo e isolamo-lo do meio em que se encontra. Ao olhar para um determinado plano com apenas um olho sem nos movermos (para nos focarmos apenas no âmbito deste projeto) por intuição percebemos que padrões, com organizações diferentes entre si, luminosidades diferentes devido a posições diferentes em relação ao(s) mesmo(s) foco(s) de luz, e tonalidades de cores agrupadas, permitem-nos diferenciar os objetos à nossa frente. Juntamos a isto o nosso histórico de tipos objetos semelhantes guardados na nossa memória (podemos comparar com as ‘provas’ referidas nas operações morfológicas), e temos grande parte da fórmula necessária para interagirmos com o meio mediante toda esta informação que temos organizada.

Embora anteriormente apenas tenham sido referidas algumas características físicas de um objeto, estas são na maioria das aplicações, as que servem de base para identificação dos mesmos em sistemas digitais. Como exemplo, ao olharmos para uma imagem de um horizonte, facilmente reconhecemos o limite que separa o céu e a terra. No entanto esse limite é apenas parte de um contorno existente na imagem da terra completa que apenas seria possível ver na totalidade olhando a 360º para o horizonte.

Num sistema de processamento de imagem, é necessário escolher o que é que são as informações/características que nos permitem identificar o objeto e extrair informação sobre si. Grande parte das aplicações de imagem que envolvem estas técnicas não têm como objetivo final a alteração da imagem, mas sim a extração de determinadas informações sobre a mesma. Para tal, da mesma forma que o ser humano utiliza inconscientemente determinadas características dos objetos para os identificar do espaço envolvente,

matricialmente é identificada informação semelhante em cada ponto adjacente. Seguem-se assim algumas técnicas que permitem adquirir essas informações.

- **Identificação de contornos**

Quando os detetores atrás referidos localizam uma aresta numa imagem apenas devolvem o resultado de uma análise a variações que ocorrem ao longo da imagem sem se preocuparem com a relação entre estas variações. Ao procurarmos pelo contorno de um objeto numa imagem, ter acesso apenas às arestas do objeto não nos dá informação suficiente sobre a sua estrutura, até porque pode resultar num contorno aberto (embora em certos casos possa até ajudar até a identificar elementos que não sejam importantes para as aplicações em causa). Uma imagem composta por arestas apenas contém informação sobre a localização das mesmas e não sobre a sua organização. Para tal, é necessário agrupá-las e identificá-las estruturalmente por meio de "etiquetas". Assim pixéis que façam parte do mesmo contorno podem ser identificados como os limites físicos de formas tais como objetos, permitindo calcular características espaciais tais como a área (interior e envolvente), centro de massa, diâmetro, entre outras.

Para chegar a esta informação existem vários algoritmos que de forma geral seguem quase todos a mesma topologia de funcionamento: detetar um ponto de uma aresta, e de modo iterativo seguir a vizinhança até terminar a associação de arestas, fechando ou não o contorno. [24] Alguns dos algoritmos mais utilizados são:

- **Seguimento de pixéis:**

  Tendo uma imagem binária constituída apenas por arestas, é verificada toda a imagem começando por uma das pontas da imagem. Assim que é detetado um pixel com o valor 1, é realizada uma pesquisa da vizinhança à volta do pixel seguindo um conjunto de ordens específicas. A ordem da sequência de pesquisa da vizinhança varia entre algoritmos.

- **Varrimento de imagem:**

  De forma semelhante ao primeiro caso, é feito o acompanhamento dos pixéis da vizinhança, no entanto cada pixel é analisado no contexto da imagem completa. Um exemplo deste tipo de algoritmo é o proposto por *Suzuki e Abe* em 1985. [25] Neste caso a análise é feita aos pares ao longo de cada linha. Dependendo da vizinhança anterior ou posterior ter o valor 0, assim é definido o espaço à volta do contorno (exterior do contorno, ou buraco). Como exemplo, na Figura 33 verifica-se que no início do varrimento de uma linha, o par vermelho identifica $f_{i,j-1}$ como a zona externa do contorno, e o par verde identifica $f_{i,j+1}$ como o interior do contorno. Desta forma é possível inclusive detetar se existem formas dentro de outras formas através

de uma hierarquia de contornos onde são identificados contornos interiores e exteriores de objetos.

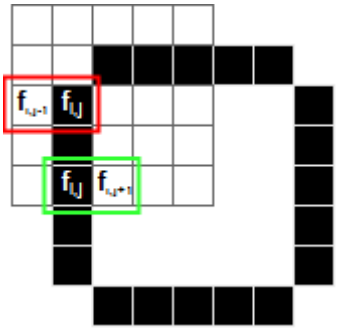


Figura 33. Exemplo do funcionamento do algoritmo de Suzuki e Abe

- **Momentos invariáveis**

Da mesma forma que em campos como a física clássica foi importante definir ferramentas matemáticas para representar diferentes características dos objetos relativamente ao seu espaço envolvente, na análise de imagens a mesma necessidade surgiu e desde cedo considerou-se aplicar abordagens semelhantes que até ao presente continuam a ter grande implementação. Em 1962, Hu [26] apresentou o estudo sobre a técnica de aquisição de dados em imagens a que chamou de "Reconhecimento de padrões por momentos invariáveis". Indo buscar à álgebra e à estatística, métodos de relacionar a exposição dos dados adquiridos de forma agrupada, conseguiu assim extrair informação que pudesse ajudar a interpretar as características que identificassem os objetos a estudar de forma relativamente simples. O conceito parte do momento matemático onde μ_n define uma característica de uma função de distribuição de probabilidade de ordem n.

$$\mu_n = \int_{-\infty}^{+\infty}(x-c)^n \rho(x)dx \qquad (17)$$

Como na estatística, a partir de uma determinada função distribuição definida por $\rho(x)$, temos a probabilidade total (0º momento), a média (1º momento) e a variância (2º momento). Já na física temos respetivamente, a massa de um determinado corpo, o seu centro de massa e inércia.

Aplicando a equação 17 a uma imagem bidimensional, Hu utilizou apenas os momentos centrais μ_{pq} de ordem $k = (p+q)$ definindo o seu cálculo relativamente ao centro de massa do objeto:

$$\mu_{pq} = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}(x-\bar{x})^p(y-\bar{y})^q \rho(x,y)dxdy\,,\ p,q = 0,1,2,\ldots \qquad (18)$$

onde,

$$\bar{x} = \frac{m_{10}}{m_{00}},\ \bar{y} = \frac{m_{01}}{m_{00}} \qquad (19)$$

são identificados como os **centros de massa**, sendo:

$$m_{pq} = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} x^p y^q \, \rho(x,y) dx dy, \qquad (20)$$

$$\mu_{00} = m_{00} = \mu \left(\frac{\boldsymbol{massa}}{\boldsymbol{area}} \boldsymbol{do\ objeto} \right) \qquad (21)$$

O facto, de segundo Hu, os momentos serem invariáveis, tem como fonte a teoria das invariantes algébricas, e deve-se à definição de que estes momentos não se alteram com a translação, rotação e alteração de escala dos corpos. Uma invariante é assim uma propriedade matemática de um objeto que não se altera perante determinadas operações. Hu definiu 7 invariantes que permitem determinar várias características de um objeto tais como o momento de inércia ou a assimetria.

Não ficando o problema totalmente resolvido, houve uma continuação do trabalho desenvolvido por outros investigadores tais como Flusser e Suk [27] que resolveram alguns erros no trabalho de Hu, expandiram a aplicação para objetos deformados e generalizaram o número de variantes possíveis.

- **Transformada de Hough**

Ter a informação da localização sobre os contornos pode ser suficiente para termos informação espacial sobre os objetos, mas não para informação geométrica. Em certas aplicações tais como controlo de movimentos por visão ou captura de objetos, é importante saber também a orientação das formas identificadas. Esta característica pode ser obtida através da orientação das linhas que formam as arestas das formas a analisar.

A transformada de Hough trata-se de uma ferramenta de deteção de linhas retas em imagens que embora tenha sido criado por Paul Hough em 1962 [28], apenas uma década mais tarde foi possível realizar a sua implementação num computador. Esta consiste na transformação das coordenadas cartesianas dos pontos da imagem em coordenadas polares, e consequente agrupamento da informação em linhas.

Pegando no exemplo da Figura 34 comecemos por identificar no plano XY um ponto $P_n(x_n, y_n)$ pertencente a uma aresta de um quadrado o qual é atravessado por reta $l_k(m_k, b_k)$ dada por

$$y = mx + b \qquad (22)$$

,onde m corresponde à inclinação da reta e b corresponde à sua interceção com o eixo Y em relação à origem do referencial Figura 35.

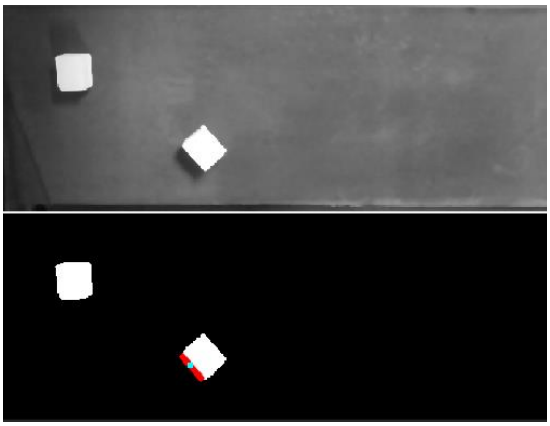


Figura 34. Imagem a analisar (transformada de Hough)

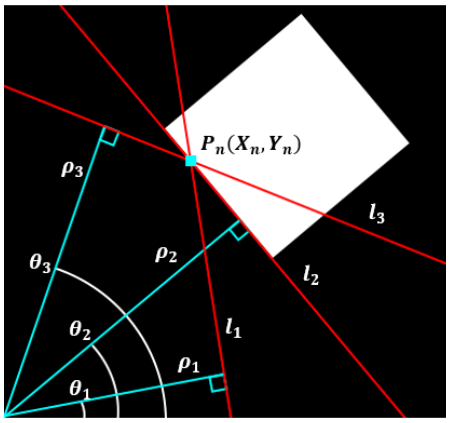


Figura 35. Transformada de Hough - Linhas

A mesma reta l pode ser descrita em coordenadas polares (θ_k, ρ_k) por:

$$\rho = x.cos\theta + y.cos\theta \qquad (23)$$

,onde θ corresponde ao angulo relativamente à origem de uma linha perpendicular à reta l, e ρ corresponde à distancia da reta à origem.

Consideremos agora um novo plano (plano de Hough) onde cada ponto tem uma localização correspondente ao conjunto das coordenadas polares (θ, ρ) de cada linha do plano cartesiano.

Se gerarmos múltiplas linhas em torno do mesmo ponto P_n, vamos ter como resultado uma curva nova no plano de Hough gerada a partir dos vários pontos correspondentes às várias linhas agora geradas.

A aplicação deste procedimento a cada um dos pontos P_n existentes na imagem, e a incrementação de um peso K pela passagem de cada curva por cada ponto no plano de Hough, vai resultar numa matriz $\theta\rho$ em que os pontos onde todas as linhas se cruzam (Figura 36) correspondem às coordenadas polares de cada linha de contorno do objeto (Figura 37).

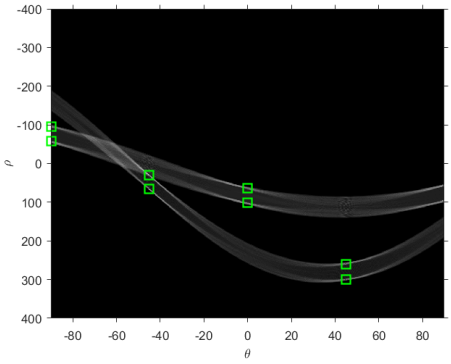


Figura 36. Plano de Hough $\theta\rho$ – localização dos pontos máximos

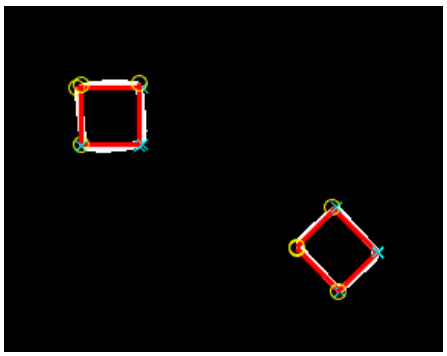


Figura 37. Plano XY - Resultado da Transformada de Hough (retas a vermelho)

2.5. Acionamentos elétricos

Em 2014 a IEA (*International Energy Agency*) estimava que 53% do consumo mundial de energia elétrica estaria relacionado com motores elétricos [29]. Se considerarmos que atualmente a maior parte das máquinas industriais têm motores elétricos algures na sua composição, estamos a falar de um dos equipamentos que mais peso tem no setor industrial, e esta presença não é difícil de perceber. O desenvolvimento do motor elétrico esteve relacionado desde cedo com o desenvolvimento das redes elétricas [30], e de modo a podermos tirar o melhor proveito deles, ao longo dos anos diversas tecnologias têm vindo a ser desenvolvidas tanto nos próprios motores como nos sistemas que se dedicam à sua alimentação e controlo.

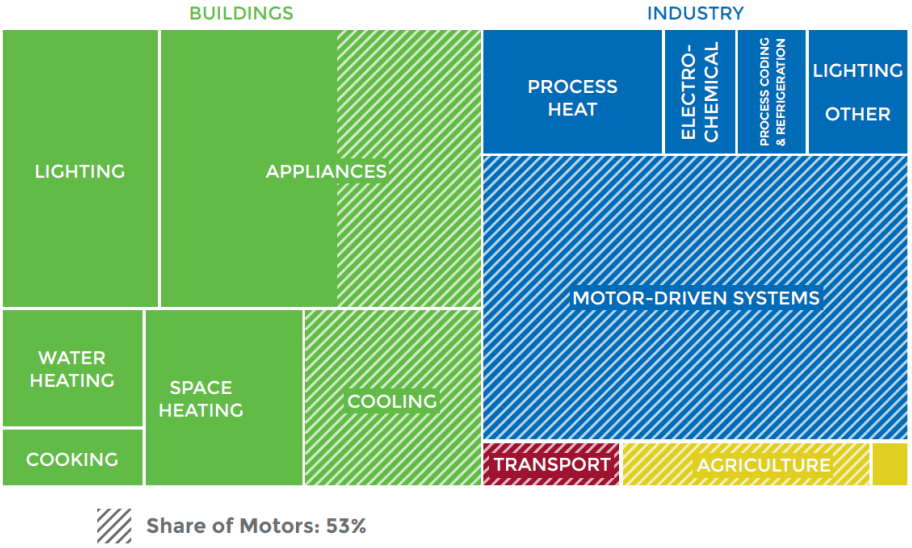


Figura 38. Consumo mundial de energia elétrica por sector em 2014 [29]

Os motores elétricos subdividem-se quanto à sua alimentação entre motores de corrente contínua (DC) e de corrente alternada (AC). Tendo sido os primeiros a surgir, os **motores de corrente contínua** exploram o fenómeno da indução eletromagnética (lei de Faraday – 1831), onde uma corrente elétrica ao percorrer um fio condutor que esteja sobre o efeito de um campo magnético gera uma força com uma direção perpendicular a ambos. São relativamente fáceis de controlar e têm como principal vantagem relativamente aos motores AC a possibilidade de produzir binário elevado desde velocidades muito baixas. Entretanto desde o aparecimento dos motores AC e o posterior avanço nas suas técnicas de controlo, a utilização de motores de corrente contínua começou a cair. O facto de os motores DC utilizarem anéis de escovas para alimentar o rotor, acabou por tornar-se um ponto negativo devido aos elevados custos de manutenção na troca das escovas (dinheiro e tempo principalmente em motores de grandes dimensões). A solução encontrada para este

problema passou por utilizar ímanes permanentes no rotor e alimentar o motor pelo estator, solução no entanto bastante dispendiosa devido ao custo dos ímanes. A alimentação ser em corrente contínua (ao contrário das redes elétricas) também tornou-se um fator limitativo relativamente aos motores AC devido a obrigar sempre à utilização de conversores/controladores. No entanto, apesar destas limitações a indústria continuou a utilizar em determinadas situações os motores DC. Casos em que o espaço seja um fator limitativo, ou em que se pretenda trabalhar com grandes variações de velocidade com baixas perdas de binário continuam a ter como solução os motores DC.

Já os **motores de corrente alternada**, para além do referido, tiveram a sua ascensão principalmente por poderem funcionar ligados diretamente à rede elétrica. Tornaram-se uma solução simples de implementar e com uma manutenção reduzida. A dificuldade surgiu quando se tentou controlar a velocidade dos motores. Devido à velocidade dos motores de indução estar diretamente dependente da frequência de alimentação, o seu controlo obriga à utilização de conversores com eletrónica de potência que funcionem com altas frequências de comutação (atualmente na casa dos kHz), algo que demorou a surgir. Aliando esta dificuldade a um princípio de funcionamento mais complexo do que nos motores DC, fez com que apenas na década de 20 surgissem os primeiros documentos relativos à variação de frequência (controlo V/F) e na década de 80 com o *boom* da eletrónica se começasse a falar no controlo por fluxo variável (também conhecido por controlo vetorial).

Figura 39. Exemplo de motor DC com respetivo controlador

Figura 40. Exemplo de motor de indução

Dependendo do tipo de motor em causa e da aplicação, o seu funcionamento pode obrigar à utilização de controladores dedicados para poderem ser alimentados e/ou controlados. Por exemplo, numa aplicação em que o motor esteja sempre a rodar à mesma velocidade, sem grandes oscilações na carga, facilmente um motor de indução (Figura 40) traz vantagens visto poder ser alimentado diretamente da rede. No entanto se essa mesma

carga necessitar de variações na velocidade ou no binário, seja qual for o tipo de motor a usar, vai ser necessário instalar equipamento que permita realizar algum tipo de controlo do motor (Figura 39).

Atualmente para realizar este controlo em motores de indução, são utilizados conversores estáticos controlados, também conhecidos por variadores eletrónicos de velocidade (VEV), que utilizam uma topologia em ponte de conversão AC-DC-AC.

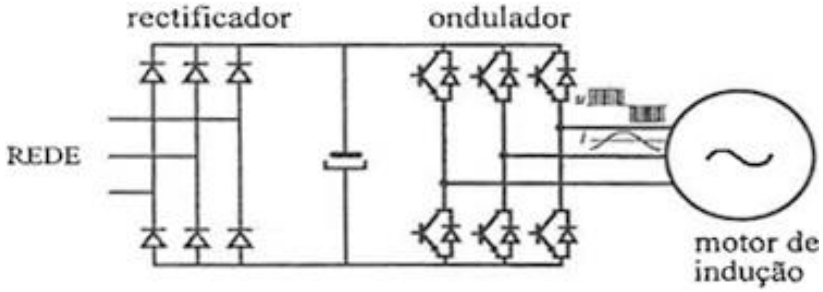


Figura 41. Topologia típica de um conversor estático para controlo do motor de indução [31]

Na Figura 41 pode-se ver um esquema desta topologia que pode ser dividida em três blocos. O primeiro (retificador) faz a conversão da alimentação da rede de corrente alternada para corrente contínua, alimentando o chamado "barramento DC". O terceiro (ondulador) transforma a tensão do barramento DC em AC através do comando de uma ponte de semicondutores com a técnica PWM (*Pulse Width Modulation*). Assim um motor é alimentado com uma tensão pulsada com um sinal médio semelhante a uma onda sinusoidal, o qual resulta também num sinal de corrente de forma análoga. É comum (embora mais dispendioso) o retificador ser formado também por semicondutores comandados, permitindo que em aplicações em que o motor funcione em regime de gerador, seja fornecida energia à rede. A partir desta topologia é depois possível aplicar técnicas de controlo tanto de velocidade como de binário como por exemplo:

- **Controlo V/F**: No motor de indução, o fluxo do motor necessário para existir produção de binário, está relacionado diretamente com a tensão e a frequência do estator. Mantendo constante a relação da equação 24 conseguimos assim manter o binário do motor dentro dos valores nominais de binário Figura 42 durante uma grande gama de velocidades (a baixas velocidades tem de ser compensada a tensão, devido à queda de tensão resistiva do motor sobrepor-se à tensão necessária à magnetização da máquina, e a velocidades acima da nominal não é possível aumentar a tensão sob o risco de danificar os isolamentos dos enrolamentos).

$$\Psi_m \approx \frac{U_s}{\omega_s} = \frac{U_s}{f*2\pi} \qquad (24)$$

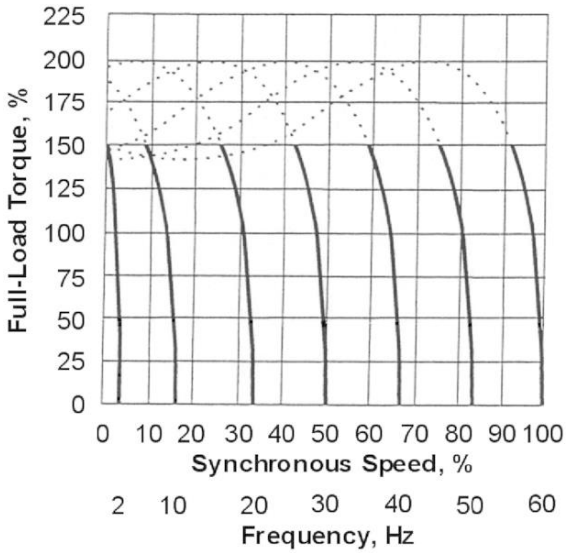


Figura 42. Curvas de binário de motor de indução com controlo V/F

- **Controlo por Orientação de Campo:** Também conhecido por "controlo vetorial" este método recorre à utilização de transformações de *Clarke* ($\alpha\beta$) e *Park* (dq) dos modelos da máquina de indução (aplicável também a outras máquinas elétricas), para traduzir separadamente o fluxo do motor e o binário como dependentes de componentes vetoriais das correntes do motor [32]. Com esta técnica é possível controlar independentemente o binário do motor desde a velocidade 0. No entanto a aplicação deste controlo obriga ao conhecimento da posição do rotor a cada momento de modo a poder-se orientar convenientemente as componentes estatóricas e rotóricas no referencial *dq* que as relaciona. Isto pode ser realizado com a utilização de encoders acoplados ao veio do motor, realizando-se um controlo em malha fechada, ou utilizando modelos equivalentes do motor passando a ser um controlo em malha aberta (menos preciso).

Figura 43. Aplicação típica de controlo por orientação de campo

3. Desenvolvimento do Projeto

3.1. Introdução

Estando apresentados nos capítulos anteriores alguns conceitos fundamentais relacionados com aspetos de desenvolvimento desta dissertação de mestrado, serão abordados neste capítulo a solução adotada para a execução do projeto completo e explicada a relação entre os diversos subsistemas integrantes. De seguida irão ser apresentados cada um desses subsistemas começando pelos mecânicos e percorrendo os seguintes de modo a se obter o sistema completo.

O sistema de acompanhamento dos objetos baseia-se essencialmente em técnicas de aquisição e processamento de imagem. Sendo o foco do projeto prático a interação com objetos a serem detetados, faz todo o sentido afirmar que o funcionamento do sistema depende do resultado da solução de imagem implementada. Desta forma será explicada a arquitetura da solução de imagem utilizada e o algoritmo implementado para permitir o funcionamento do sistema.

Seguidamente é apresentada a interligação de todos os subsistemas numa só solução integrada. Sendo o controlo baseado num Raspberry Pi serão em primeiro lugar abordadas as interfaces disponíveis, e depois as criadas para servir de ponte de comunicação com os diferentes componentes.

Por fim é fechado o tema com a apresentação da arquitetura e dos pontos principais do software desenvolvido no controlador, que juntam tudo o que foi referido até à altura, sendo apresentadas as principais características dos mesmos e os modos de funcionamento existentes e os utilizados. O software está dividido em 2 aplicações. A primeira trata-se de uma ferramenta de calibração que serve de configuração do programa principal e que será descrita no desenvolvimento de cada ponto com o qual se relacione. A segunda trata-se do programa principal que realiza todo o processo que foi discutido e que é apresentado no final do capítulo.

3.2. Solução adotada

A elaboração do projeto descrito nesta dissertação teve como objetivo a criação de um sistema de captura de diferentes objetos que se deslocassem sobre um transportador, sem qualquer intervenção humana. O transportador tendo como acionamento um motor de indução e um variador eletrónico de velocidade, dá flexibilidade suficiente para ser realizado o arranque e paragem, e alterar a velocidade de deslocamento dos objetos a qualquer altura através de controlo remoto. Para realizar a captura foi utilizado um robot articulado de 5 juntas, este também controlável remotamente. Para conhecer a posição do objeto a cada momento, recorreu-se à utilização como elemento sensor, de uma câmara de captura de vídeo, e posteriormente ao processamento por um controlador principal. O controlo de todo o conjunto de subsistemas de forma a colocar o projeto em funcionamento foi realizado de modo centralizado partilhando informação diversa o mais próximo possível do tempo real, de modo a realizar o correto acompanhamento e captura. Foi assim crítico ter em conta o tempo de ciclo da execução de cada tarefa de modo a que o sistema cumprisse os pressupostos fluidamente.

Outra alternativa à topologia escolhida seria dividir a mesma em vários subsistemas onde cada um iria dedicar-se a assegurar uma tarefa específica. No caso da câmara esta iria estar incluída num sistema independente que iria encarregar-se de todo o processamento de imagem e até do acompanhamento dos objetos ao longo do tapete. Embora não sendo um requisito neste caso, um variador eletrónico de velocidade (VEV) poderia receber feedback de um encoder instalado no tapete fazendo um controlo direto da velocidade linear. Com estas implementações, o controlador central iria apenas dedicar-se a manter o sincronismo e a relação entre todos os subsistemas. No entanto, seria uma solução muito mais dispendiosa devido à complexidade dos equipamentos associados a cada uma destas funções isoladas que nativamente têm funcionamentos bem mais simples.

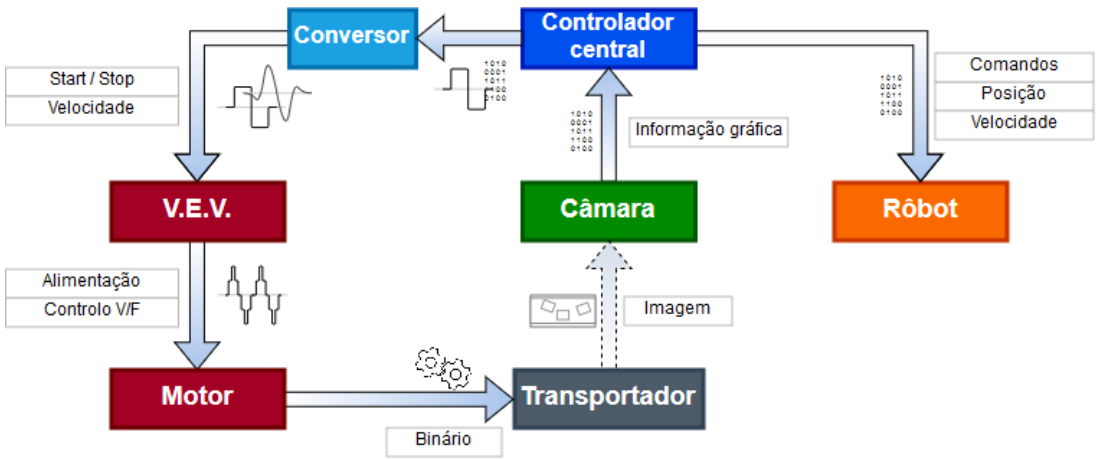


Figura 44. Modelo físico do sistema completo

Na configuração desenvolvida (Figura 44), o controlador recebe ciclicamente as imagens capturadas pela câmara, realizando todo o processamento de imagem necessário, e nos momentos em que estiverem reunidas as condições para a captura dos objetos, comanda a paragem do transportador, e posteriormente envia a sequência de captura dos comandos do robot. Durante o restante tempo, o transportador encontra-se sempre em movimento. Embora idealmente a captura dos objetos devesse ser feita com o transportador em movimento, sem ou com redução de velocidade, o mesmo não foi possível fazer devido às limitações de controlo do mesmo. Dentro dos comandos documentados, e depois de vários testes revelou-se que não era possível ter vários comandos seguidos em buffer a fim de realizar movimentos em algum tipo de modo contínuo (*CP – Continuous Path*), nem comandar o terminal de captura em simultâneo com os movimentos do robot. Com esta capacidade seria possível o robot realizar um movimento de acompanhamento da peça sobre o tapete e durante o mesmo apanhar a peça.

É também possível verificar no modelo da Figura 44 que os vários componentes têm, pela sua natureza, diferentes grandezas a serem transmitidas entre si. Isto obriga à criação de interfaces para que seja possível converter e transferir estas grandezas entre diferentes subsistemas. O facto do controlador utilizado possuir apenas entradas e saídas de sinais discretas, cria uma barreira com o VEV a ser ultrapassada visto que se pretende enviar um sinal de velocidade. Os sinais de comando também têm características diferentes. Já no caso da câmara e do robot é feita sempre uma troca de sinais digitais, mas que no entanto utiliza protocolos de comunicação diferentes. A câmara utiliza uma interface USB que está também presente no controlador, já o robot por outro lado comunica por RS-232. Por fim existem também as grandezas físicas características de cada subsistema como descritas no capítulo 2.

3.3. Sistemas mecânicos

3.3.1. Transportador

O transportador das peças a capturar trata-se de um tapete rolante de 1170mm x 500mm (comprimento x largura) acionado por um motor assíncrono através de uma caixa redutora com relação de transmissão desconhecida e de um carreto com corrente mecânica. A alimentação do motor é feita a partir de um variador eletrónico de velocidade *Commander SK* [33] de alimentação monofásica com controlo V/F, cuja seleção embora obviamente sobredimensionada recaiu também sobre a disponibilidade de material dentro do tipo de variador necessário para o projeto. Foi assim utilizado o variador SKA1200075:

Model Number	Nominal motor power		Maximum input fuse rating	Typical full load input current	100 % RMS output current	150 % overload current for 60 s
	kW	hp	A	A	A	A
					Heavy Duty	
SKA1200025	0.25	0.33	6	4.3	1.7	2.55
SKA1200037	0.37	0.5	10	5.8	2.2	3.3
SKA1200055	0.55	0.75	10	8.1	3.0	4.5
SKA1200075	0.75	1.0	16	10.5	4.0	6.0

Figura 45. Características principais do variador (fonte: [33])

Já o motor utilizado tem como valores nominais:

- $P_n = 220\ W$
- $I_n = 1.04\ A$
- $U_n = 220\ V\ (\Delta)$
- $N_n = 2780\ rpm$
- $cos\varphi = 0{,}85$ A
- $f_n = 50\ Hz$

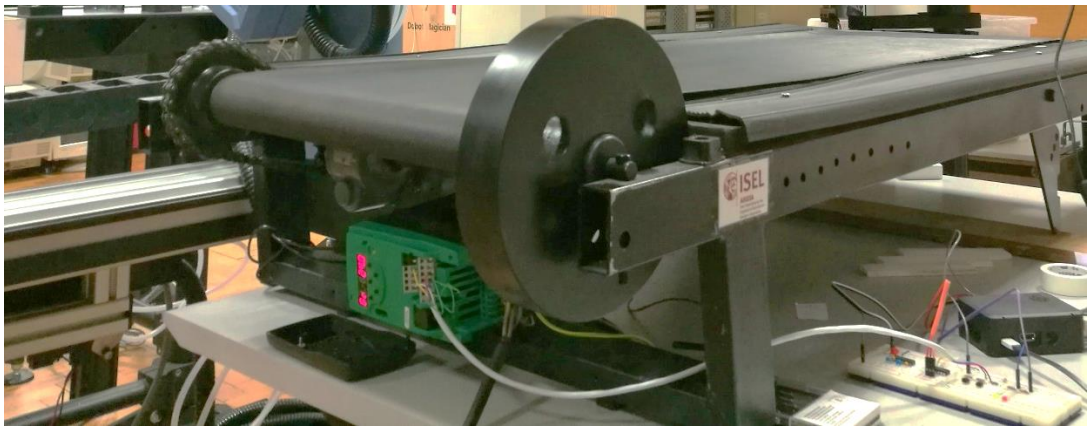


Figura 46. Fotografia do Transportador, Motor e VEV

Embora inicialmente o objetivo fosse a velocidade do transportador ser adaptada durante a captura dos objetos, tal não foi possível devido à limitação do robot já referida. Passando por isso o controlo a ser mais simples, optou-se por manter a funcionalidade do transportador sendo o variador controlado pelo comando de arranque e paragem do motor através de um sinal digital de $0 - 24V_{CC}$ e o comando de variação da velocidade através de um sinal analógico $0 - 10V_{CC}$. Durante o funcionamento do sistema, o tapete irá iniciar o movimento com o arranque do sistema a uma velocidade definida e parar nos momentos de captura de peças (Figura 46). Embora vá estar sempre a deslocar-se à mesma velocidade,

caso necessário poderia variar a velocidade a qualquer altura durante o seu funcionamento (Figura 47, Figura 48 e Figura 49).

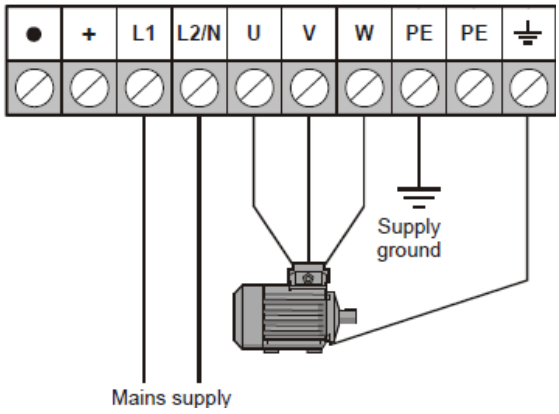


Figura 47. Esquema de circuito de potência do VEV-Motor [33]

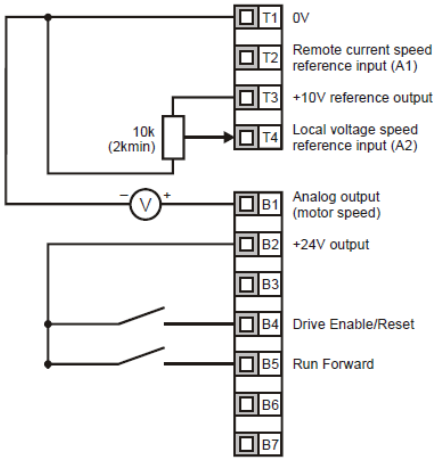


Figura 48. Esquema do circuito de comando do VEV [33]

T1	**0V common**

T4	**Analog input 2 (A2), either voltage or digital input**
Voltage: Digital input	0 to +10 V / 0 to +24 V
Scaling (as voltage input)	Input range automatically scaled to Pr **01** *Minimum set speed* / Pr **02** *Maximum set speed*
Resolution	0.1 %
Input impedance	100 kΩ (voltage) /6 k8 (digital input)
Normal threshold voltage (as digital input)	+10 V (positive logic only)

B2	**+24 V output**
Maximum output current	100 mA

B4	**Digital Input - Enable/Reset*/****
B5	**Digital Input - Run Forward****
B6	**Digital Input - Run Reverse****
B7	**Digital Input - Local/Remote speed reference select (A1/A2)**
Logic	Positive logic only
Voltage range	0 to +24 V
Nominal threshold voltage	+10 V

Figura 49. Lista de características dos Inputs de comando do VEV [33]

3.3.2. Robot

O equipamento utilizado trata-se do modelo Scorbot-ER9 (Figura 50 e Figura 51). É um robot elétrico articulado, com 5 juntas de rotação e uma garra elétrica como órgão terminal. As juntas encontram-se, 1 na base do corpo do robot (*base)*, 3 ao longo do braço (*shoulder, elbow* e *pitch)*, e a última na garra (r*oll*) permitindo a rotação desta. O corpo do robot está assente numa 6ª junta linear que embora não fosse possível controla-la remotamente permitiu posicionar manualmente o robot paralelamente ao tapete.

Figura 50. Robot Scorbot ER9

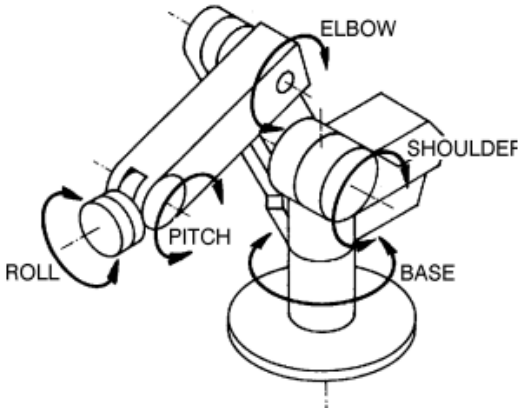


Figura 51. Movimentos das juntas do robot

As juntas deste robot são acionadas por servomotores C.C., posicionadas com recurso ao feedback de encoders óticos incrementais. Visto tratarem-se de sensores incrementais, sempre que o robot é alimentado, é necessário proceder-se a uma auto-calibração. Durante este procedimento, o robot move cada um dos eixos sequencialmente até à posição *HOME*, servindo esta como referência para os movimentos do robot.

Mecanicamente, o robot tem um raio máximo paralelo à base de 691mm (excluindo a garra), e o centro da segunda junta (*shoulder*) está a uma altura de 388mm relativamente à base (Figura 52 e Figura 53).

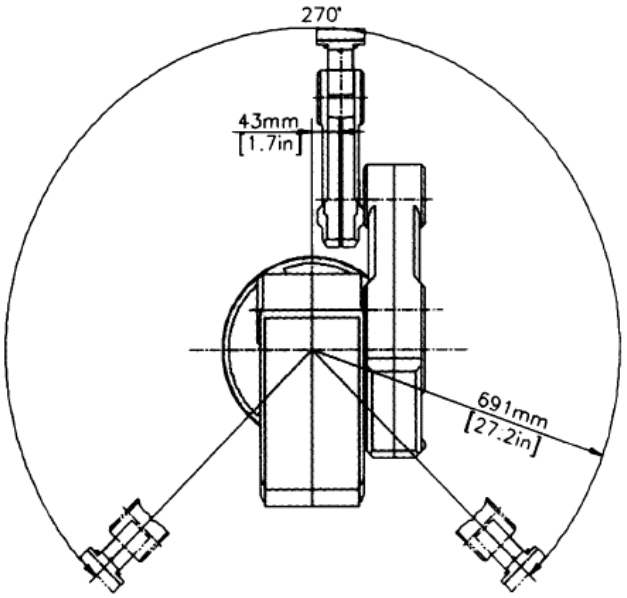


Figura 52. Área de trabalho do robot (vista topo)

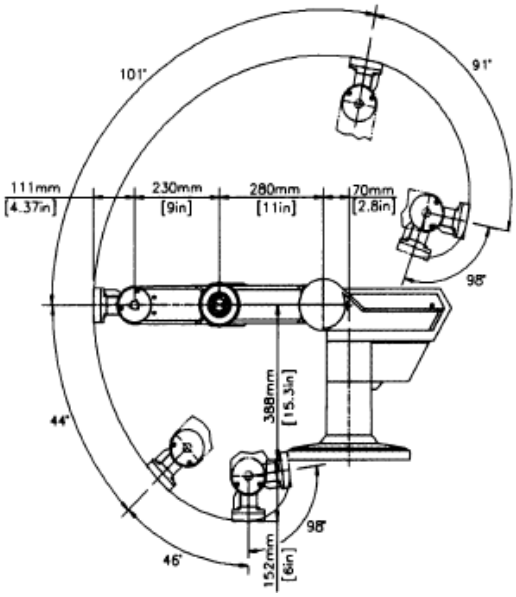


Figura 53. Área de trabalho do robot (vista lateral)

O robot tem um controlador externo o qual permite alimentar os servomotores, e estabelecer comunicações com outros dipositivos tais como uma consola móvel dedicada (*Teach Pendant*), um computador com software dedicado, ou qualquer outro sistema independente a partir de comunicação série com comandos dedicados através de uma porta RS-232 (Figura 54).

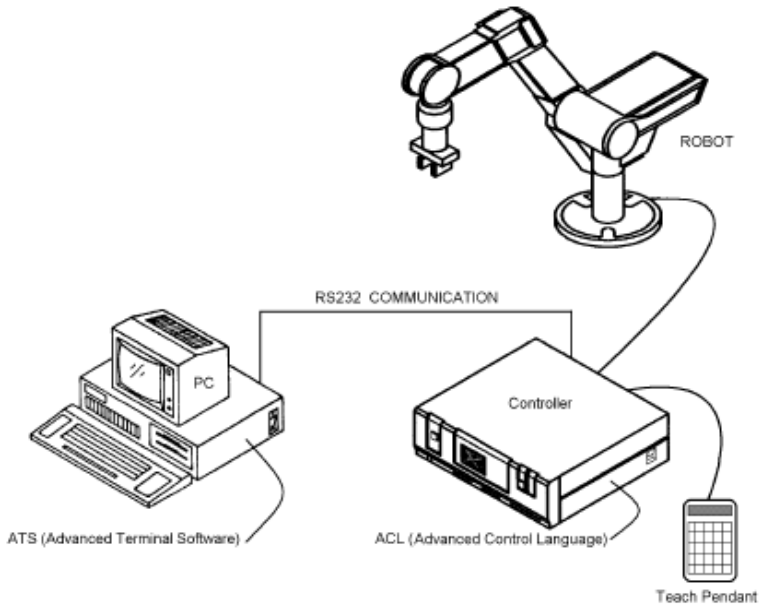


Figura 54. Diagrama de ligações elétricas do robot (fonte: [34])

Devido às dimensões tanto da estrutura do tapete como do próprio robot, foi necessário determinar a largura disponível a cruzar com a área de trabalho do robot de forma

a realizar a captura. Desta forma, objetos que surgissem fora da área útil de trabalho (zona a verde na Figura 55) seriam à partida ignorados pelo programa.

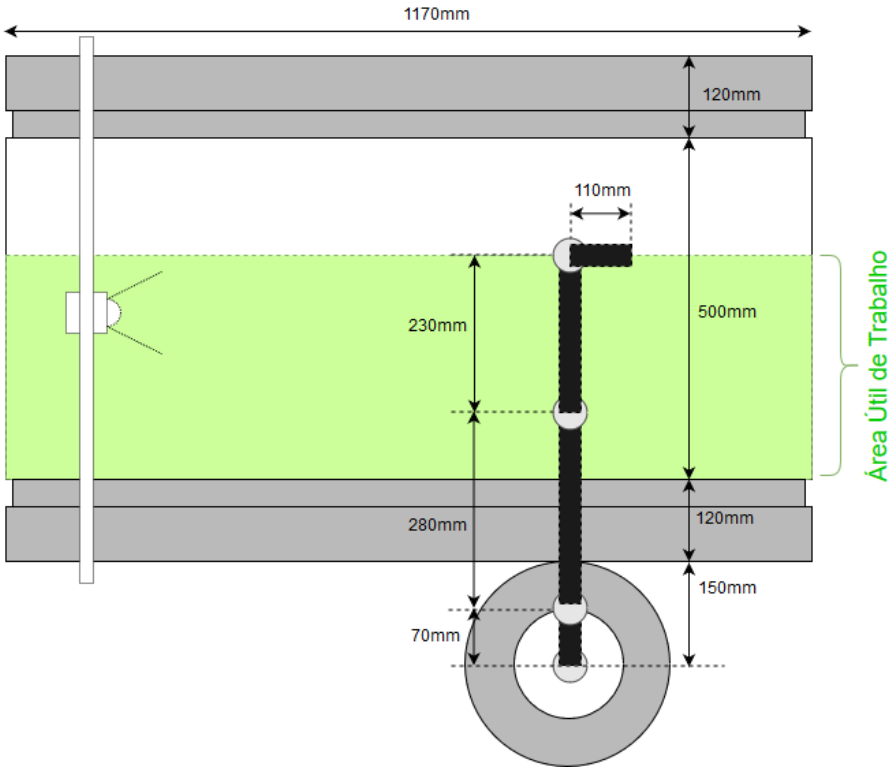


Figura 55. Posição do robot relativamente ao transportador (vista de cima)

É importante referir que a distância tem de ter em conta a posição do órgão terminal no momento da captura. Na Figura 56 observamos o esquema do manipulador na posição de captura de um objeto à distância máxima da sua área de trabalho. Conhecendo as dimensões do robot podemos então calcular a largura útil de trabalho do transportador, l:

$$l = \sqrt{(280 + 230)^2 - (388 - 311 - 50)^2} - (150 + 120 - 70) = \sim 309mm \qquad (25)$$

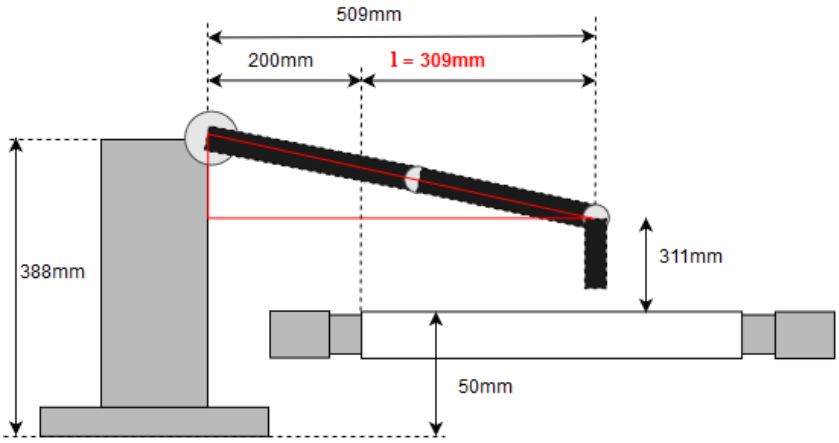


Figura 56. Posição do robot relativamente ao transportador (vista lateral)

3.3.3. Orientação dos eixos e captura

Para fazer a ligação entre o espaço físico, o espaço de movimentos do robot e da câmara, é essencial relacionar a orientação de ambos. Com o robot já com o seu referencial predefinido, optou-se por manter a orientação do mesmo utilizando o seu sistema de coordenadas cartesianas. Relativamente à posição da câmara, sendo um equipamento fácil de posicionar, foi optado por localizá-la de forma a que o movimento do tapete ficasse paralelo ao seu eixo X e ao do referencial do robot. É importante fazer referência à orientação do eixo Z da câmara. O facto da lente estar virada para baixo na direção do tapete faz com que o seu eixo Z fique nesta mesma direção, mas com o sentido contrário ao do robot que tem a sua área de trabalho orientada para cima (Figura 57). A câmara ficou posicionada a uma altura de 90 cm relativamente à superfície do tapete.

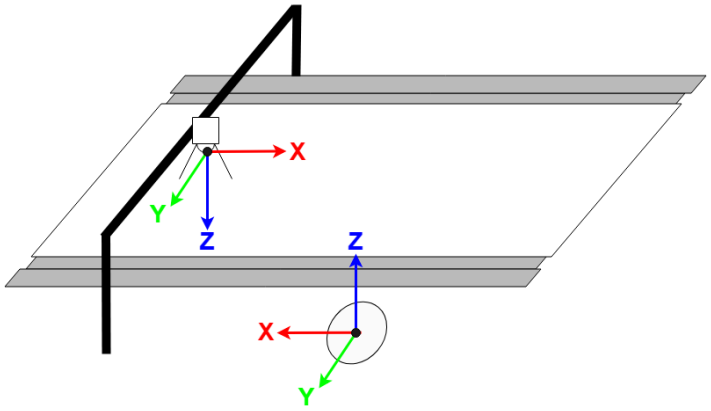


Figura 57. Sistemas de eixos de coordenadas do robot e da câmara

Entre os 2 subsistemas existe também uma diferença de grandezas que é necessária ter em consideração. O robot por defeito utiliza instruções em milímetros (mm) enquanto que a câmara trabalha com pixéis (px). No entanto não tendo a câmara resolução suficiente à distância utilizada (necessária para apanhar uma região comprida de tapete) não faria sentido trabalhar com uma escala inferior ao centímetro (cm) visto que desvios inferiores não iriam ser totalmente percetíveis. Assim foi essencial criar variáveis que relacionassem todas estas grandezas resultando assim nas seguintes equações para as posições de captura (*pickX e pickY* – Figura 58).

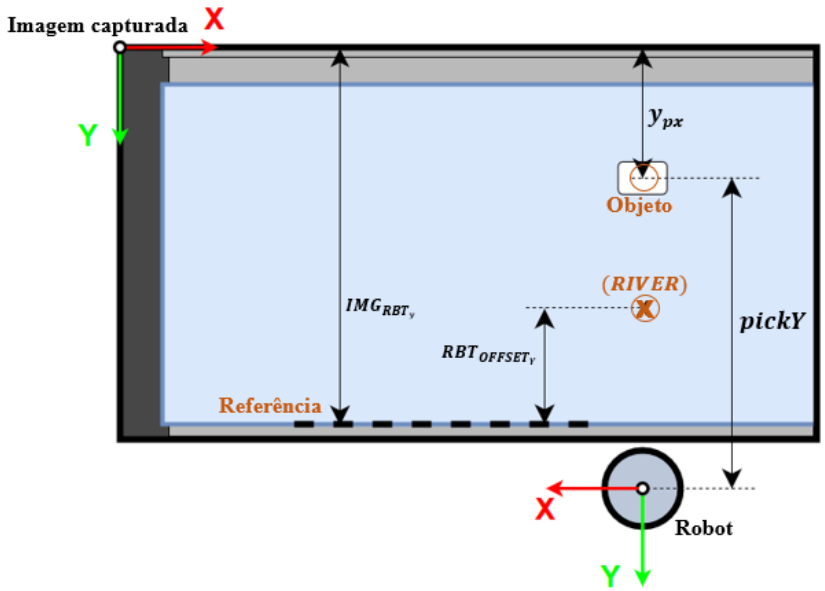


Figura 58. Esquema com variáveis utilizadas para cálculo de captura

Sendo y_{px} a posição da peça em pixéis e $CALIB_Y$ a relação pixel/cm da imagem capturada pela câmara, temos assim y_{mm} como a posição da peça sobre o tapete em milímetros relativamente à origem do referencial da imagem. Conhecida também a posição da garra do robot ($RBT_{OFFSET_{Ymm}}$) relativamente a um ponto comum com a imagem (Referência), e a posição do referencial da imagem ($IMG_{RBT_{Ypx}}$) relativamente a esse mesmo ponto, é passado este último também para milímetros. A partir daqui chega-se às posições enviadas pelo controlador para o robot para este executar a captura dos objetos. Este valor vai ser a soma da posição de referência do robot $RBT_{RIVER_{Ymm}}$, com a distância em Y deste último ponto à extremidade de baixo do tapete (segundo a Figura 58).

O ponto denominado *RIVER* trata-se de um ponto presente nas configurações do robot que serve de ponto inicial para este sistema. A sua localização é nas coordenadas *X, Y, Z, P, R:* (*8.937, -427.888, 357.933, -90.689, 88.683)*.

$$y_{mm} = \frac{y}{CALIB_Y} * 10 \qquad (26)$$

$$IMG_{RBT_{Ymm}} = \frac{IMG_{RBT_{Ypx}}}{CALIB_Y} * 10 \qquad (27)$$

$$pickY_{mm} = \left(RBT_{RIVER_{Ymm}} + RBT_{OFFSET_{Ymm}}\right) - \left(IMG_{RBT_{Ymm}} - y_{mm}\right) \qquad (28)$$

$$pickX_{mm} = RBT_{RIVER_{Xmm}} \qquad (29)$$

Estando definida a relação entre os dois sistemas de coordenadas, falta definir o método de estabelecer a relação entre as grandezas utilizadas pelo robot (mm) e pela câmara (pixel). Não sendo esta relação fixa, visto depender da posição física de todos os componentes, foi necessário criar uma ferramenta para se definir esta relação em qualquer

momento. Para tal foi criada uma rotina acessível pelo programa de calibração que tem como saída as variáveis $CALIB_X$ e $CALIB_Y$ que correspondem à quantidade de pixéis de imagem por centímetro no transportador, segundo os eixos x e y respetivamente.

Para proceder à calibração, o software começa por capturar uma fotografia ao tapete (Figura 59) com a câmara e inicia uma sequência de passos onde o utilizador tem de selecionar com o rato diversos pontos na imagem (Figura 60) seguindo as instruções dadas na imagem de referência, e inserir a distância entre esses pontos em centímetros. Idealmente estes pontos deverão ser marcas colocadas sobre o tapete a distâncias já conhecidas.

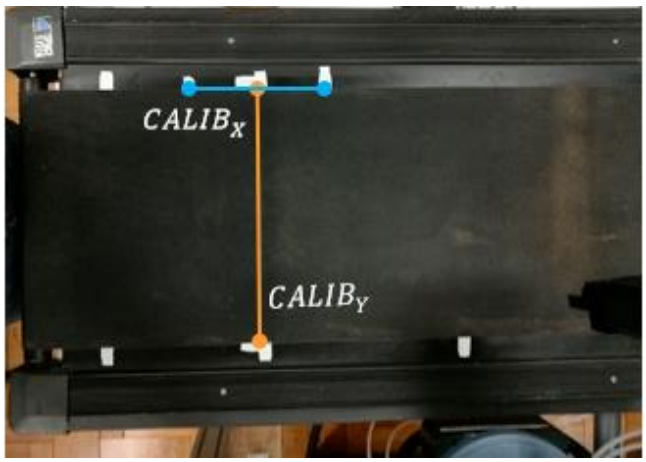


Figura 59. Marcação de pontos no transportador para calibração

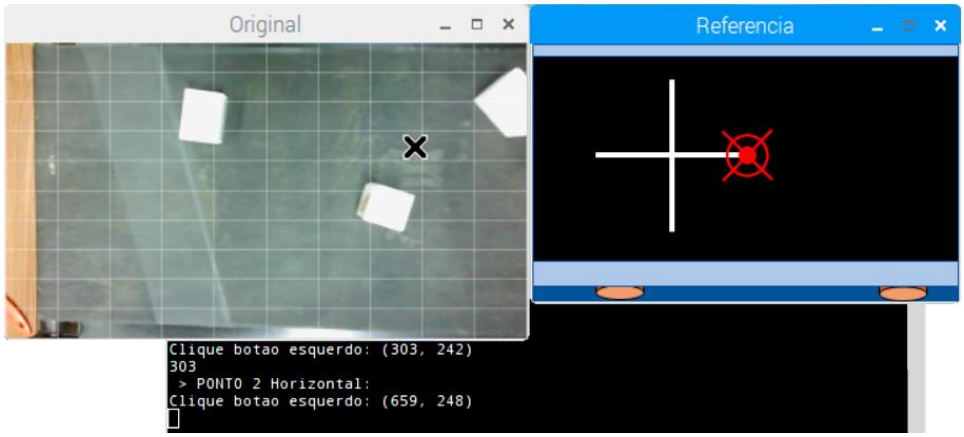


Figura 60. Processo de calibração de imagem

3.4. Arquitetura da solução de imagem

Para realizar o acompanhamento dos objetos com os quais o sistema vai interagir, em primeiro lugar é necessário que seja definido o que é que vai ser detetado. Uma imagem resultante da aquisição de um sensor ótico, após passar por toda a cadeia de processamento analógico/digital, nada mais é do que um matriz *n* por *m* com um ou mais níveis dependendo do espaço de cores, com informação em formato numérico. Depende da aplicação é então

necessário definir o que se considera como sendo informação de interesse e como a mesma está organizada. Assim sendo, para desenvolver a solução em questão foi tido como entradas:

- Objetos a detetar (características)
- Ambiente onde objetos se inserem
- Posicionamento dos componentes do sistema
- Deslocamento previsto

Os objetos a serem detetados tratam-se de paralelepípedos de faces lisas, com diferentes orientações e cores que se deslocam numa determinada direção fixa sobre um transportador. Na Figura 61 verificamos uma foto inicial dos objetos onde conseguimos desde já identificar várias características específicas da aplicação sobre o transportador.

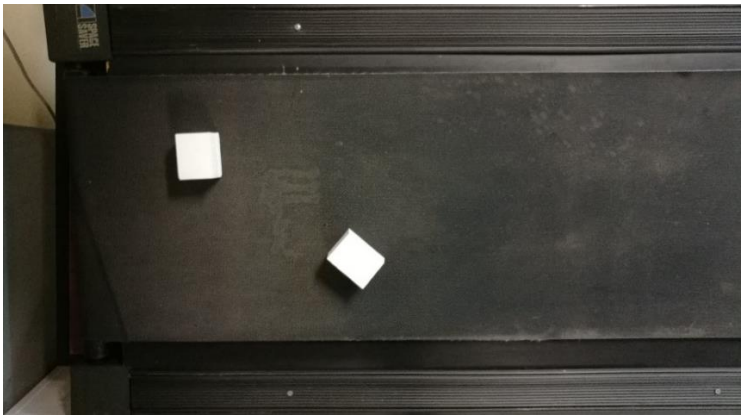


Figura 61. Captura de imagem de peças sobre o transportador

Devido a estarmos a utilizar apenas um ponto de aquisição de imagem num plano amplo, deparamo-nos com o efeito da perspetiva, o que faz com que identifiquemos várias faces em simultâneo do objeto dependendo da posição. A localização da câmara, é então um fator bastante influenciador que foi tido em conta.

A iluminação é sempre um dos fatores que mais afetam qualquer aplicação que recorra a processamento de imagem. Embora no caso da Figura 61 as condições estivessem minimamente controladas (ambiente fechado com luz artificial), foram realizados testes com sucesso com diferentes condições (variações na luz natural ao longo do dia). Para tal foi essencial a criação de uma ferramenta de calibração das várias operações de processamento de imagem (capítulo 3.7) para ser possível chegar a uma solução que se adaptasse às condições reais. Apesar de a luz incidir de forma mais ou menos uniforme sobre o transportador, diferentes materiais resultam em diferentes reflexões da luz e a própria diferença na cor da luz pode tornar ainda mais difícil a identificação das características dos

objetos. As sombras causadas pela posição da iluminação podem também constituir um problema.

Apesar das peças serem facilmente identificáveis, existiam mais formatos de imperfeições verificadas ao longo de todo o tapete (manchas, riscas, etc) que podem gerar falsos objetos durante a deteção das formas. Em último caso, dependendo da posição dos objetos sobre o tapete, podem mesmo ser associados as formas aleatórios aos objetos, e provocar erros na identificação das suas características.

Perante estas condições temos assim em primeiro lugar de fazer um tratamento prévio da imagem adquirida (Figura 62). Todo este processamento inicial vai ter influência na restante cadeia do algoritmo de deteção de objetos e por isso será sempre uma variável a ser adaptada às necessidades dos restantes pontos. Com uma filtragem inicial conseguimos por um lado ajudar a compensar os fatores externos ao sistema (iluminação e fundo) e por outro excluir o processamento de informação que não é necessária atingindo menores tempos de ciclo e maior exatidão durante o acompanhamento dos objetos. A nível de software (Figuras Figura 62 e Figura 63), após diversos testes foi então criada uma função que executa em sequência as seguintes tarefas:

- Filtro de escala de cinza
- Mediana
- Operação morfológica (abertura)
- Limiarização

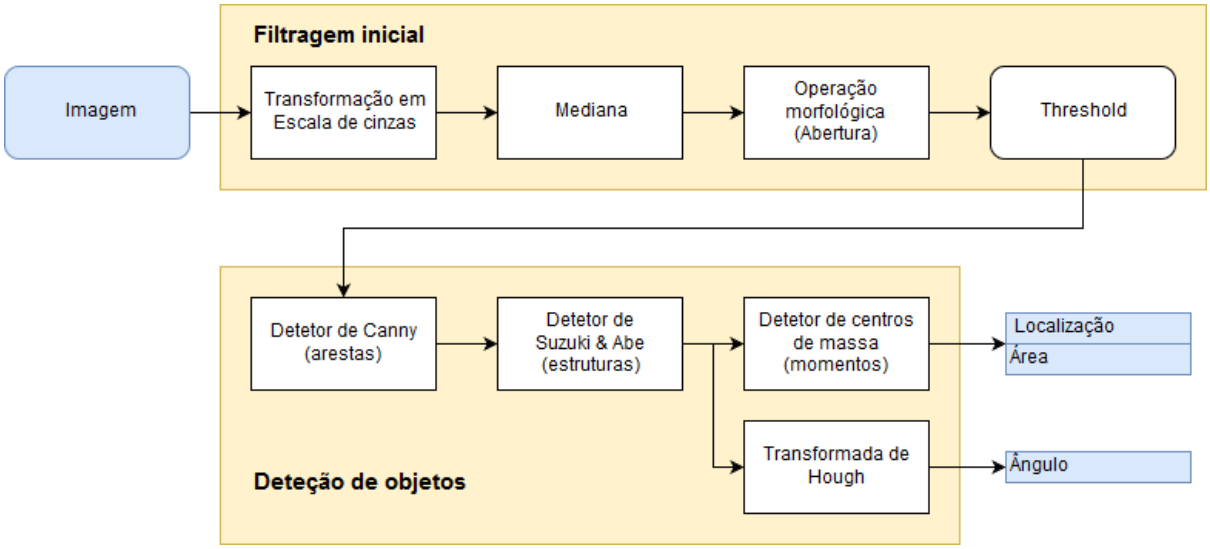


Figura 62. Modelo do algoritmo de processamento de imagem utilizado

```
cvtColor( frameA, frameB, COLOR_BGR2GRAY );
medianBlur( frameB, frameC, KSIZE );
morphologyEx( frameC, frameD, MORPH_OPEN, MORPH_ELEM );
threshold( frameD, frameE, THRESHOLD_VALUE, THRESHOLD_MAX_BINARY_VALUE,THRESHOLD_TYPE );
```

Figura 63. Código com lista de funções de filtragem aplicadas

Já tendo sido descritos as características de todas as técnicas acima referidas, em capítulos anteriores, serão apenas descritos os parâmetros de entrada de cada uma das funções:

Os dois primeiros parâmetros de cada função indicam a matriz de entrada e a matriz de saída resultante da filtragem aplicada. De seguida temos:

- *COLOR_BGR2GRAY*: Indica que se pretende passar do espaço de cores RGB para o espaço de escala de cinza. Esta função utiliza a recomendação ITU-R BT.601 referida no capítulo 2.
- *KSIZE*: índice K da mediana correspondente ao tamanho da vizinhança
- *MORPH_*SIZE: Tamanho da forma de prova a utilizar na abertura morfológica
- *MORPH_ELEM*: Forma de prova a utilizar na operação morfológica (foi utilizada a elipse)
- *THRESHOLD_*VALUE: Valor de *threshold* utilizado na limiarização
- *THRESHOLD_*MAX_BINARY_VALUE: Valor atribuído aos pixéis acima do valor de *threshold* (normalmente cor branca).
- *THRESHOLD_*TYPE: Tipo de resultado de threshold utilizado (binário direto, binário invertido, …)

Após essa filtragem inicial, é realizada a identificação no espaço dos objetos que nos interessam e a determinação das suas características (Figura 64). Esta informação irá servir para determinar a evolução do objeto ao longo do tempo e do espaço, para determinar as condições de captura por parte do robot. Esta identificação utiliza as seguintes técnicas:

- Detetor de arestas de Canny
- Detetor de contornos de Suzuki e Abe
- Cálculo de momentos invariáveis
- Transformada de Hough

```
Canny( frame, proc_canny_output, CANNY_THRESHOLD1, CANNY_THRESHOLD2, CANNY_APERTURE_SIZE);
findContours(proc_canny_output, proc_contornos, proc_hierarchy, CV_RETR_EXTERNAL, CV_CHAIN_APPROX_SIMPLE, Point(0,0));
mMassa[i] = moments( proc_contornos[i], false);
HoughLinesP(proc_canny_output, lines, 1, CV_PI/180, HOUGH_THRESHOLD, HOUGH_LINE_LENGHT, HOUGH_LINE_GAP );
```

Figura 64. Código com lista de funções de deteção de objetos

Como parâmetros de entrada para estas funções temos:

- *CANNY_THRESHOLD1* e *CANNY_TRESHOLD2*: Filtragem de pesos do algoritmo de Canny
- *CANNY_APERTURE_SIZE*: Dimensão da matriz de utilizada no filtro de Sobel

- *CV_RETR_EXTERNAL*: Instrução para devolver apenas os contornos externos dos objetos segundo o algoritmo de Suzuki & Abe
- *CV_CHAIN_APPROX_SIMPLE*: Instrução para devolver apenas os pontos limites dos contornos detetados
- *OFFSET - Point(0,0)*: Utilizado caso se pretenda utilizar um ROI dentro da matriz a completa (neste caso o ROI já foi previamente definido)
- *BINARY_IMAGE = false (moments())*: parâmetro para o caso de se utilizar matrizes binárias
- *RHO = 1 (Hough)*: Distancia máxima entre linhas no varrimento de ρ
- *THETA = CV_PI/180*: Resolução do varrimento do ângulo θ
- *HOUGH_THRESHOLD*: Após aplicar a Transformada de Hough apenas linhas com peso K acima deste valor são consideradas válidas
- *HOUGH_LINE_LENGHT*: Comprimento mínimo das linhas detetadas para serem consideradas válidas
- *HOUGH_LINE_GAP*: Espaço máximo aceite entre dois pontos para serem considerados pertencentes à mesma linha

Embora alguns dos parâmetros tanto de filtragem como da determinação das características sejam fixos do ponto de vista do funcionamento da abordagem utilizada de processamento de imagem, outros podem ser adaptados às condições de captura. Para tal, foram adicionadas duas opções ao programa de calibração, uma dedicada à filtragem, e outra dedicada à transformada de Hough.

Para os parâmetros de filtragem, como mostrado na Figura 65, foram disponibilizadas as variáveis correspondentes ao tamanho da vizinhança na mediana (*KSIZE*), ao tamanho da forma do operador morfológico (*MORPH_SIZE*), e aos limites da limiarização (*THRESHOLD_VALUE* e *THRESHOLD_MAX_BINARY_VALUE*). Para alterar os valores o utilizador apenas tem de inserir os valores pela ordem indicada, tal como no exemplo. De seguida o programa captura novas imagens e utiliza a nova configuração para apresentar os resultados.

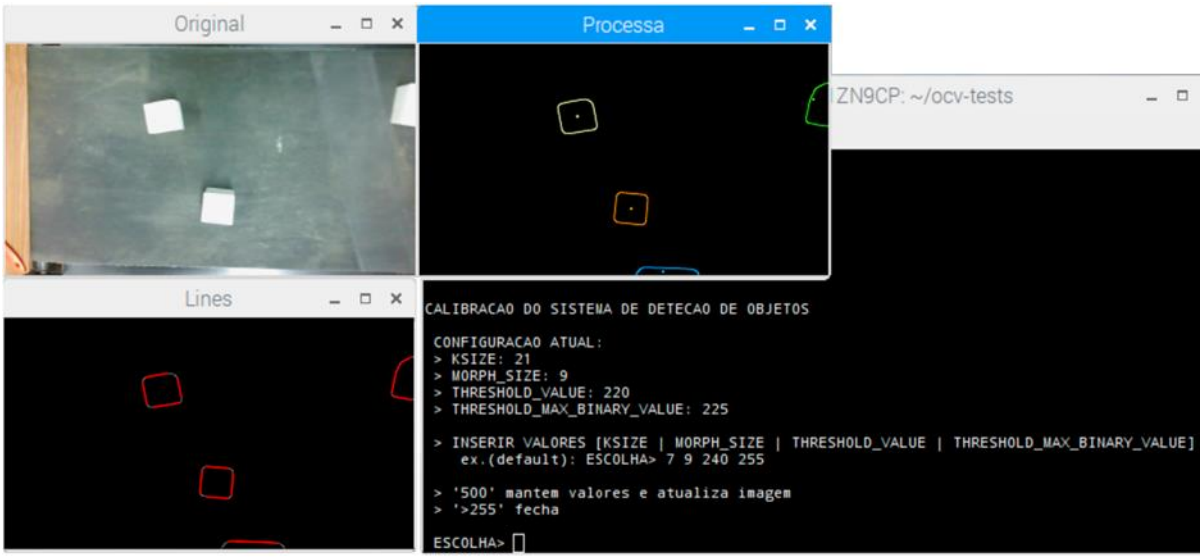


Figura 65. Menu de calibração de imagem (filtragem)

A calibração da deteção de objetos a partir da transformada de Hough utiliza um menu semelhante, desta vez com os parâmetros de entrada *HOUGH_THRESHOLD, HOUGH_LINE_LENGHT*, e *HOUGH_LINE_GAP* (Figura 66). Os resultados são apresentados em janelas separadas da mesma forma que no menu anterior.

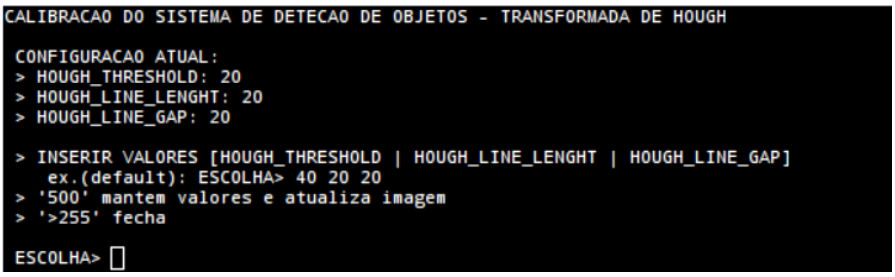


Figura 66. Menu de calibração da transformada de Hough

3.5. Algoritmo de acompanhamento de objetos

Embora no nosso caso possam existir movimentos em ambos os eixos X e Y, devido ao posicionamento da câmara em relação ao tapete e ao efeito da perspetiva, o movimento dos objetos é linear e retilíneo ao longo da imagem, o qual não exige a aplicação de técnicas para compensar/prever desvios de trajetórias. Embora se tenha optado por adquirir a velocidade do tapete a partir da imagem como consequência da velocidade do motor previamente definida, o facto de termos controlo sobre a velocidade do motor, poderia ajudar-nos a tornar o sistema ainda mais dinâmico caso fosse necessário. No entanto como referido antes, é desconhecida a relação de transmissão entre o motor e o tapete, pelo que optar por este método resolve este problema.

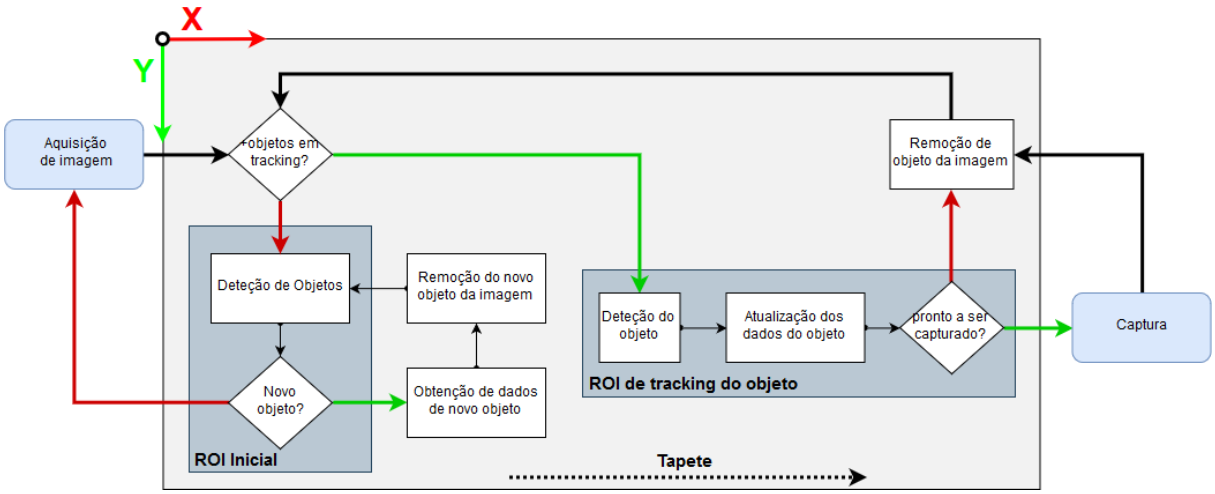


Figura 67. Modelo do algoritmo de acompanhamento de objetos

Na (Figura 67) podemos observar o modelo do algoritmo utilizado para realizar o acompanhamento (*tracking)* e consequente captura dos objetos. Começando pela ROI Inicial (*Region of Interest*), esta trata-se de uma zona no início do tapete que cobre a sua largura visível (segundo o eixo Y), tendo em consideração também a largura da área de trabalho do robot na zona de captura. Esta zona é reservada à primeira deteção dos objetos. Sempre que um novo objeto completo é detetado nesta zona, o mesmo é adicionado a um *buffer* juntamente com os dados adquiridos relativos à sua posição, centro de massa e orientação. De seguida o objeto é eliminado da imagem adquirida e é de novo feito o varrimento em busca de um novo objeto, permitindo assim realizar a deteção e acompanhamento de vários objetos em simultâneo. No entanto pode acontecer ser capturada uma nova imagem quando um objeto anteriormente detetado pela primeira vez ainda não saiu da *ROI Inicial*. Por esta razão são analisados em primeiro lugar todos os objetos que já se encontrarem em *buffer* e eliminados da imagem.

Esta região é definida manualmente durante o procedimento de calibração, no respetivo programa (Figura 68). Para tal, o utilizador deve selecionar três pontos na imagem capturada do tapete de forma análoga à imagem de referência apresentada, a fim de selecionar a zona de deteção inicial de objetos. Esta zona deve ter dimensão suficiente para serem detetados os objetos em qualquer orientação possível e analisar a maior largura de tapete possível dentro da área de trabalho disponível. De forma semelhante para definir a coordenada em Y a partir da qual se considera que o objeto está na área de trabalho do robot pronto a ser capturado, o utilizador através de uma rotina dedicada, tem de selecionar um ponto indicativo dessa mesma posição. Mais uma vez devido a serem utilizadas medidas físicas, é recomendado a utilização de marcas no próprio tapete com vista a obter resultados mais exatos como mostrado na Figura 69.

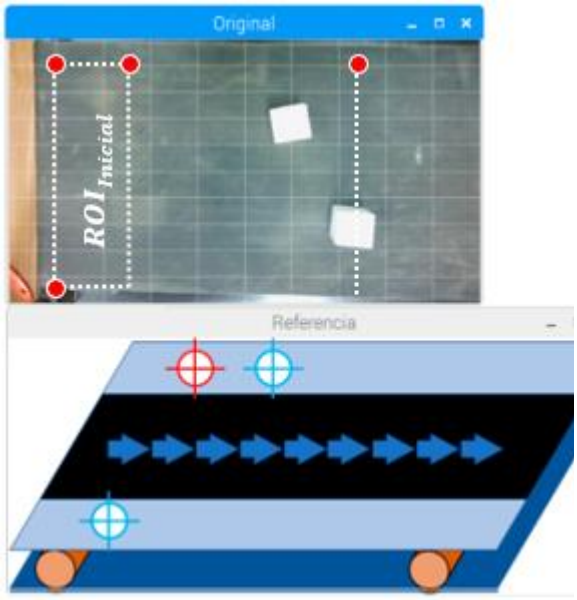


Figura 68. Procedimento de definição do ROI inicial e da zona de captura

Figura 69. Marcações para definição do ROI inicial e zona de captura

Durante a primeira deteção, além das características físicas, é criada também uma nova *ROI de Tracking* do objeto (Figura 67), que é atualizado a cada nova deteção. Esta nova região é gerada tendo em conta as dimensões do objeto e alongada para a zona onde é previsto que o objeto esteja na imagem seguinte. Esta previsão é feita a partir da equação do movimento retilíneo aplicada aos centros de massa dos objetos e tendo em conta que a aceleração é nula visto que a velocidade do tapete é constante (equações 30 a 33). A velocidade do objeto é calculada após um determinado número *n* de deteções, as quais acontecem inicialmente ainda durante o tempo em que o objeto se encontra dentro da *ROI Inicial*. Ao longo de todo o avanço do tapete um *buffer* com os tempos e posições do objeto vai sendo atualizado permitindo fazer pequenas correções da informação sobre a velocidade. Ao atingir um limite de captura previamente definido (Figura 69), o tapete é parado, e os comandos com toda a sequência de movimentos de captura são enviados para o robot. Após a captura ser realizada o tapete arranca de novo, e o ciclo recomeça. É de relembrar que embora os cálculos sejam realizados em pixéis (imagem) e centímetros (conversão para espaço físico), o robot recebe os comandos de posicionamento em milímetros (mm) pelo que a conversão é realizada utilizando as relações imagem-espaço $CALIB_X$ e $CALIB_Y$ em centímetros resultantes da calibração e depois alterada a escala.

$$X_n = X_{n-1} + v_x * t + a * t \quad [px] \qquad (30)$$

$$v_x = \frac{\Delta X}{\Delta t} = \frac{X_n - X_{n-1}}{t_n - t_{n-1}}, \quad v_y = \frac{Y_n - Y_{n-1}}{t_n - t_{n-1}} \; [px] \qquad (31)$$

$$vel_x = \frac{v_X}{CALIB_X} \quad [cm/s], \; vel_y = \frac{v_Y}{CALIB_Y} \quad [\mathrm{cm/s}] \qquad (32)$$

$$X_{n_{mm}} = \frac{10 * X_n}{CALIB_X} \quad [mm], \; Y_{n_{mm}} = 10 * \frac{Y_n}{CALIB_Y} \quad [mm] \qquad (33)$$

Embora o modelo utilizado acabe por ser simples do ponto de vista funcional, a principal vantagem está no facto de se analisar quantidades muito menores de informação do que em outras soluções padrão, visto que apenas é processada informação de pequenas áreas da imagem. Com isto é possível vir a reduzir bastante o tempo de processamento, aproximando-o mais do tempo mínimo de receção de uma nova imagem.

3.6. Integração dos sistemas

3.6.1. Raspberry Pi

Lançado pela primeira vez em 2012 por Eben Upton, Rob Mullins, Jack Lang e Alan Mycroft na *Universidade de Cambridge*, o *Raspberry Pi* (RPi) trata-se de um microcomputador do tamanho de um cartão de crédito (Figura 70), que surgiu inicialmente com o objetivo de criar um módulo de prototipagem, flexível, e com fortes possibilidades de conectividade com diversos sistemas externos. Trata-se de uma plataforma de computação de baixo custo que permite ao utilizador ter acesso a um hardware de processamento de alto nível, programável em ambiente *Linux*, e que ao contrário de um computador comum, tem disponível uma grande variedade de entradas e saídas discretas que podem ser utilizadas com grande flexibilidade (GPIO – *General Purpose Input/Output*).

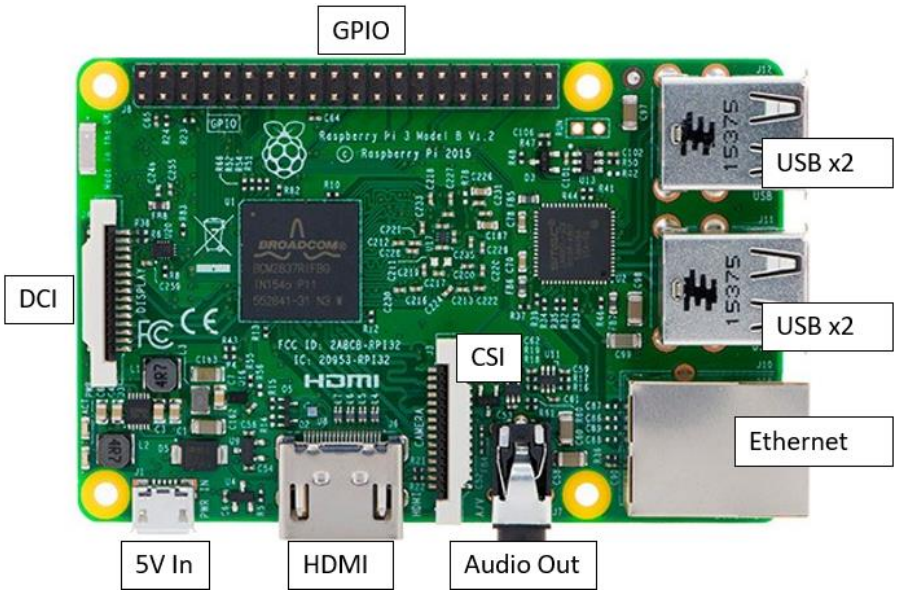


Figura 70. Placa Raspbery Pi 3 Model B e respetivas interfaces

A arquitectura do RPi está centrada no SoC (*System-On-a-Chip*) da Boardcom, que dependendo dos modelos tem como principais características (RPi 3 Model B) [35]:

- **CPU:** 4xARM Cortex-A53, 1,2Ghz
- **GPU:** Boardcom VideoCore IV
- **RAM:** 1GB LPDDR2 (900Hz)
- **Rede:** 10/100 Ethernet 2,4Ghz 802.11n
- **Armazenamento**: microSD
- **Portas**:
  - HDMI
  - Áudio via conector 3,5mm
  - 4xUSB 2.0
  - CSI - Módulo *Raspberry Pi Câmara* (modelo dedicado)
  - DSI – Interface para monitor
  - GPIO

A variedade de interfaces disponíveis, e a flexibilidade que esta plataforma disponibiliza para trabalhar em diferentes temáticas juntamente com toda a disponibilidade de livrarias disponíveis em ambiente Linux, foram os pontos chaves que tornaram este "micro-PC" na escolha para este projeto. Desde cedo foram realizados testes para testar o comportamento do Raspberry Pi à exigência computacional da aplicação, saindo-se com sucesso e mostrando ser uma ferramenta eficaz na realização das tarefas pretendidas. A versão disponível do hardware foi a *Raspberry Pi 3 Model B v1.2.* A linguagem de programação recaiu sobre o C++ devido à sua flexibilidade e estabilidade já reconhecidas, devido à sua portabilidade para posteriores desenvolvimentos ao projeto, e para caso de haver necessidade de partir para outra plataforma por razões de falta de processamento durante o desenvolvimento na plataforma escolhida. O facto de se trabalhar "mais perto" do hardware, isto é, sem a necessidade de utilizar demasiadas interfaces e *frameworks* para suportar toda a aplicação contribuiu para uma mais rapidez de processamento. Ainda assim foi utilizada a livraria *WiringPi*© que disponibiliza ferramentas para um acesso rápido e simples às GPIO do Raspberry Pi. Quanto ao processamento de imagem, recorreu-se à utilização da livraria *OpenCV*© em C++ que sendo flexível o suficiente, também contém as ferramentas necessárias ao desenvolvimento da solução.

À exceção da câmara que utilizou comunicação USB para comunicar com o Raspberry Pi, os restantes componentes utilizaram a interface GPIO do controlador para comunicar. Na Figura 71 pode-se ver a organização dos pinos disponíveis. Para a alimentação dos diversos circuitos, existem ligações a 5V e a 3,3V com ponto equipotencial comum, e diversos pinos (marcados a verde) para utilizar com quaisquer funções pretendidas de entrada e saída. Para

comunicação estão disponíveis alguns pinos de ligação para utilizar com protocolos como o SPI (comunicação série síncrona) e UART (comunicação série assíncrona).

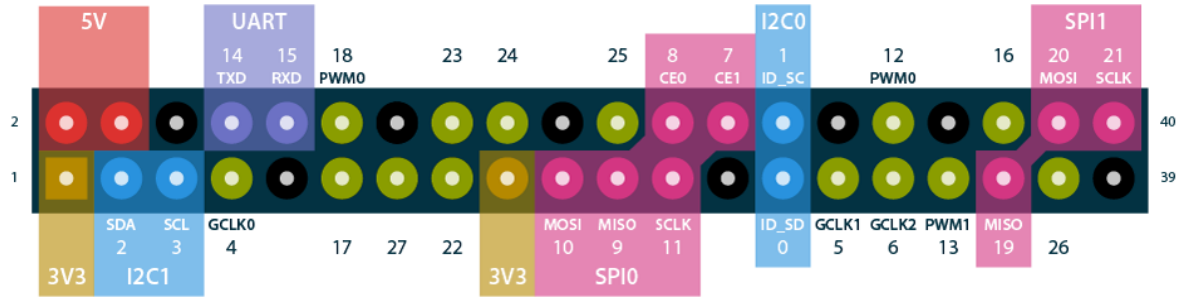


Figura 71. Pinout do Raspberry Pi 3 Model B V1.2 [36]

Durante o desenvolvimento do projeto, foi utilizado um computador externo ligado em rede *Ethernet* ao Raspberry Pi para auxiliar na programação do mesmo e na análise de dados. A comunicação foi feita com recurso a um software de acesso via linha de comandos pelo protocolo *SSH* (*Secure Shell*) e com um software de acesso *VNC* (*Virtual Network Computing*) para acesso direto ao ambiente de trabalho.

3.6.2. Câmara

Para a captura de imagem foi utilizada uma câmara da marca *Trust* com comunicação USB (Figura 72). O sensor de captura de imagem tem uma resolução de 1280x720 pixéis, possibilitando velocidades de captura de imagens a cores até 30 fps (frame per second).

Figura 72. Câmara de captura de imagem

3.6.3. Robot

Embora tanto o robot como o controlador partilhem o método de comunicação série, o robot utiliza a interface RS232, enquanto que do lado do controlador temos o GPIO como interface com níveis de tensão CMOS que se situam fora da gama admissível pela norma anterior:

Valor lógico	R.Pi GPIO (V)	Robot RS-232 (V)
0	0	+3 a +15
1	3,3	-3 a -15

Tabela 1. Comparação de tensões de sinais de comunicação entre GPIO e o Robot

Para fazer a conversão entre estes 2 sinais recorreu-se à utilização de um conversor em circuito integrado, o MAX3232 (Figura 74), passando os sinais de saída de GPIO para valores de $\pm 5V$ com a lógica corrigida podendo o circuito ser alimentado à mesma por $3{,}3V$.

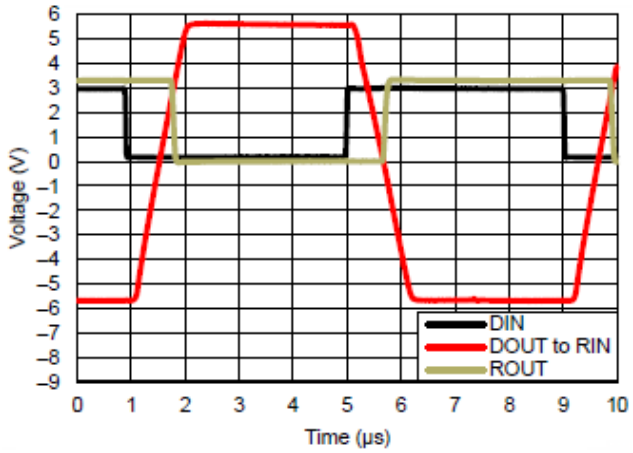


Figura 73. Sinais de entrada e saída do MAX3232 [37]

Necessitando este dispositivo de alguma eletrónica auxiliar (Figura 74) para gerar as tensões necessárias, foram utilizados os valores de componentes sugeridos pelo fabricante (Tabela 2).

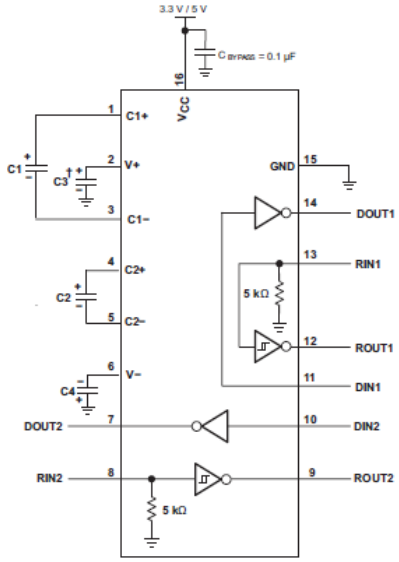


Figura 74. Esquema elétrico do MAX3232 [37]

V_{CC}	C1	C2, C3, C4
3.3 V to 0.3 V	0.1 µF	0.1 µF
5 V to 0.5 V	0.047 µF	0.33 µF
3 V to 5.5 V	0.1 µF	0.47 µF

Tabela 2. Lista de condensadores recomendados [37]

Por fim, para enviar os comandos para o robot, foi necessário configurar os parâmetros da comunicação série. Para este tipo de comunicação, os sistemas baseados em UNIX [38] utilizam a API (*Application Programming Interface*) *termios* [39] que fornece um conjunto de ferramentas para estabelecer comunicações em portas de comunicação assíncronas. As configurações da porta série para comunicar com o robot encontram-se assim nas funções de comunicação *abreCom()* e no final da função *enviaCom() (*Figura 75*)*:

```
        int abreCom(){
            // Porta GPIO: /dev/ttyS0
            // Baudrate: 9600 bits/s
            // Databits: 8
            // FlowControl: None
            // Parity: None
            // Stop bits: 1
            // Terminador: Carriage Return = 0x0D
            int baudrate = 9600;
            int fd;
            if ((fd = serialOpen("/dev/ttyS0", baudrate)) < 0) {
            //(...) handling de erros
            else {
                struct termios options;             // TERMIOS: Definições
                tcgetattr (fd, &options);           // Lê parametros actuais
                options.c_cflag &= ~CSIZE;          // Mascara da mensagem
                options.c_cflag |= CS8;
                options.c_cflag |= CSTOPB;
                tcsetattr (fd, TCSANOW, &options);  // Escreve novos parametros
                cout << "# Com. aberta: fd> " << fd << endl;
                return fd;
            }
        }

int enviaCom(int fd, const string& msg, int tempo){
    const char *arrayMsg = msg.c_str();
    std::cout << "# enviaCom() msg> '" << arrayMsg << "'" << std::endl;
    serialPrintf(fd,arrayMsg);
    serialPrintf(fd,"%c ",0x0D); // Terminador - Carriage Return
}
```

Figura 75. Código de comunicação série com as configurações da comunicação

3.6.4. Variador

Durante o funcionamento do sistema, o VEV responsável pelo movimento do transportador receberá 2 tipos de instruções:

- Arranque / Paragem (sinal único)
- Referência de Velocidade

Para a indicação de início de marcha será utilizado o sinal de output BCM 18 (*Broadcom SOC channel*), no entanto mais uma vez temos uma diferença de tensões nos sinais:

Valor lógico	R.Pi GPIO (V)	VEV (V)
0	0	0
1	3,3	24

Tabela 3. Comparação de tensões de sinais de comando entre GPIO e o VEV

A solução adotada para a resolução deste problema foi a montagem de um circuito de transístores NPN-PNP. Analisando os transístores 2N3904 (NPN) [40] e 2N3906 (PNP) [41] conclui-se que cumprem os nossos requisitos e tratando-se de 2 modelos relativamente simples de adquirir, foram os escolhidos:

- **2N3906**

Não existindo documentação que indique a corrente necessária para a entrada digital do VEV funcionar, sabe-se pelo menos que se situa abaixo dos 100mA (alimentação máxima disponibilizada pelo VEV para sinais digitais de comando [33]). Partindo deste pressuposto e realizando algumas medições chegou-se então à conclusão que o consumo era abaixo dos 5mA. Analisando o datasheet do 2N3906, percebemos que a corrente está bem abaixo do máximo suportado pelo transístor, e a tensão também se situa dentro dos limites admissíveis para o semicondutor.

- **2N3904**

Para comandar o transístor PNP dentro dos valores de carga pretendidos é necessária uma corrente na base de pelo menos 0.1mA. Os outputs do Raspberry Pi admitem uma corrente máxima de 16mA por saída, e 50mA distribuído por todas as saídas do *Bus* [35]. Pela análise do datasheet percebemos que uma corrente na gate do 2N3904 de $0.5mA$ é mais do que suficiente para permitir a passagem da corrente ($> 30mA$) que o 2N3906 necessita na gate ($< 10mA$).

Aplicando uma resistência de $10k\Omega$ na gate do 2N3906 limitou-se a corrente a $\sim 2.5mA$, que se mostrou, com boa margem, ser o suficiente para a entrada digital funcionar. Quanto à gate do 2N3904, com uma resistência de $4.7k\Omega$ ficou com uma corrente de $\sim 0.5mA$ na gate. Os valores das resistências foram escolhidos para permitir dentro dos valores pretendidos, simplificar o esquema utilizando material facilmente acessível no mercado.

$$\frac{24-0.85\ (V)}{2.5*10^{-3}\ (A)} = 9260\ (\Omega),\ \frac{3.3-0.85\ (V)}{0.5*10^{-3}\ (A)} = 4900\ (\Omega) \qquad (34)$$

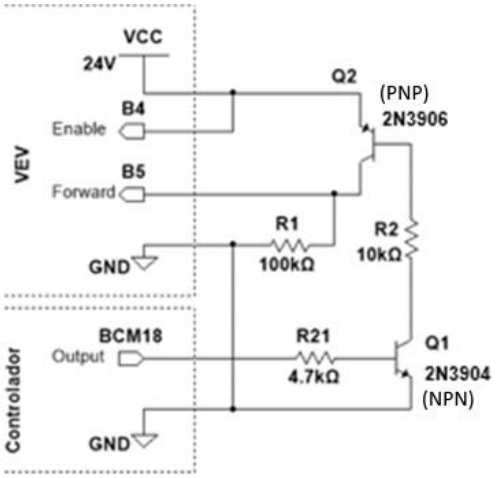


Figura 76. Circuito de acionamento do VEV

O VEV utilizado dispõe de uma entrada analógica de 0V a 10V que pode ser utilizada como sinal de controlo da velocidade do motor. Já o Raspberry Pi apesar de não dispor de qualquer sinal analógico de saída, tem sim uma saída de comunicação SPI (Figura 77) que pode ser utilizada para comunicar em curtas distâncias com outros periféricos como Conversores Digital-Analógico (DAC) em circuitos integrados que partilhem a mesma interface. É o caso do MCP4922 (Figura 78) utilizado que tem como principais características:

- 2 canais de 12bit
- Interface SPI
- Tensão de alimentação (V_{DD}): $2.7V$ a $5.2V$
- Corrente de alimentação (I_{DD}): $50mA$
- Sinal de Referência: $0V$ a V_{DD}

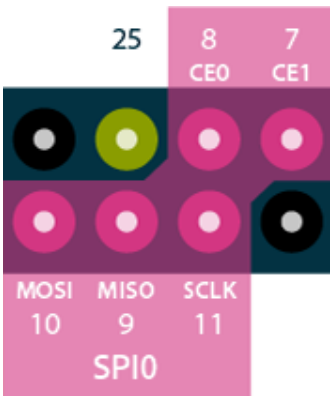


Figura 77. Pinout do Raspberry Pi (SPI) [36]

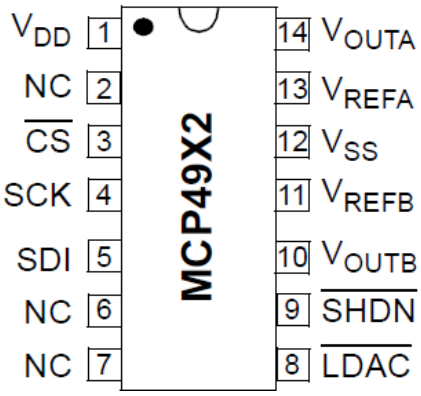


Figura 78. Pinout do MCP4922 [42]

Pin	Descrição – Sinais de comunicação
MOSI	*Master Out Slave In* – Sinal de saída série
$MISO$	*Master In Slave Out* – Sinal de entrada série
$SCLK$	*Serial Clock* – Sinal de sincronização da comunicação
CE0 / CE1	*Chip Enable* – Sinal de seleção de Slave

Tabela 4. Lista de pin's do Raspberry Pi dedicados a SPI

Pin	Descrição – Alimentação e sinais analógicos
V_{DD}	Tensão de alimentação
V_{SS}	Ground
$V_{OUT\#}$	Saída do sinal analógico
$V_{REF\#}$	Entrada do sinal analógico de referência
Pin	**Descrição – Sinais de comunicação**
$\overline{CS}$	Chip-Select
SCK	Serial Clock Input
SDI	Serial Data Input

Tabela 5. Lista de pin's do MCP4922 dedicados a SPI

Uma das primeiras conclusões a tirar a partir das características indicadas, é que o DAC apenas consegue atingir nas suas saídas cerca de metade da tensão indicada no VEV. Uma das soluções poderia passar por adicionar um circuito amplificador à saída do MCP4922. Outra seria configurar o VEV para utilizar metade da escala da entrada analógica para a velocidade máxima. Por motivos de simplicidade de montagem optou-se por utilizar a segunda opção.

Para controlar o DAC, são transmitidas mensagens de 16bit (Tabela 6 e Tabela 7) para os seus registos com a seguinte configuração:

$\overline{A}/B$	BUF	$\overline{GA}$	$\overline{SHDN}$	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
bit 15															bit 0

Tabela 6. Trama da mensagem de comando do DAC via SPI

bit	Descrição
$\overline{A}/B$	Seleção de saída a que se refere a mensagem (‘0’ para saída A)
BUF	Utilizar o buffer para manter o sinal à saída
$\overline{GA}$	Selecionar ganho na saída (‘0’ para $V_{out_{máx}} = 2 * V_{ref}$, ‘1’ para $V_{out_{máx}} = V_{ref}$)
$\overline{SHDN}$	“Shutdown” desativa a saída o DAC com o valor ‘0’
$D0 - D11$	Valor em 12bit ($n = 2^{16} = 4096$)

Tabela 7. Descrição dos bits da mensagem da DAC via SPI

Embora o protocolo SPI permita comunicação em *Full-Duplex*, ou seja, possibilita o envio e receção de dados em simultâneo, devido a dispor de dois canais de comunicação *MISO* e *MOSI*, o MCP4922 apenas suporta a receção de dados, não tendo nenhuma instrução de feedback. Assim, a comunicação procede-se segundo a trama que se pode observar na Figura 79:

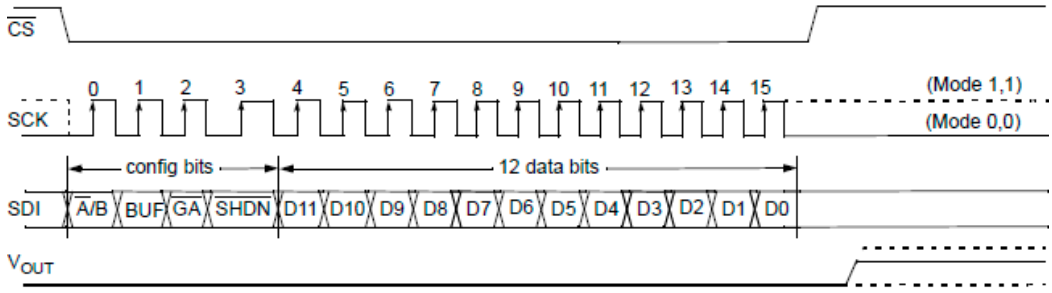


Figura 79. Diagrama temporal da comunicação com o MCP4922 (fonte: [42])

Utilizou-se assim a seguinte ordem de mensagem por defeito – velocidade nominal (Tabela 8):

$\overline{A}/B$	BUF	$\overline{GA}$	$\overline{SHDN}$	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x3				0xFFF = '4095' = V_{ref}											

Tabela 8. Mensagem enviada do Raspberry Pi para o DAC via SPI

Convertendo para hexadecimal: 0x3FA0

$$V_{out} = \frac{V_{ref} * D_n * GA}{n} = \frac{5 * 2048 * 1}{4096} = 2.50\ V \qquad (35)$$

A implementação torna-se assim bastante simples como se pode ver na Figura 80:

```
// ========================================================
// Configurações para comunicação SPI
// ========================================================
static const char *spiDevice = "/dev/spidev0.0";
static uint8_t spiMode = 0;
static uint8_t spiBits = 8;
static uint32_t spiSpeed = 500000; //Hz
static uint16_t spiDelay;
// ************ ARRANQUE TAPETE ************
fd_spi = startSPI();
if (fd_spi == -1){return -1;}
uint8_t msg_spi[] = {0x3F,0xA0};
wiringPiSPIDataRW(fd_spi, 0, msg_spi, ARRAY_SIZE(msg_spi));
digitalWrite(1, HIGH);
```

Figura 80. Código de comunicação via SPI com o DAC

3.6.5. Sinalizadores

Para ajudar no desenvolvimento da solução foram utilizados 2 leds para indicar:

- Captura de imagem / início de ciclo: Led Verde (alternado)
- Transportador ligado: Led Vermelho

Para os leds *TLLR4400* e *TLLG4400* temos as seguintes características (Tabela 9):

PARTS TABLE

PART	COLOR	LUMINOUS INTENSITY (mcd)			at I_F (mA)	WAVELENGTH (nm)			at I_F (mA)	FORWARD VOLTAGE (V)			at I_F (mA)	TECHNOLOGY
		MIN.	TYP.	MAX.		MIN.	TYP.	MAX.		MIN.	TYP.	MAX.		
TLLR4400	Red	0.63	1.2	-	2	612	-	625	2	-	1.9	2.4	2	GaAsP on GaP
TLLG4400	Green	0.63	1.2	-	2	562	-	575	2	-	1.9	2.4	2	GaP on GaP

ABSOLUTE MAXIMUM RATINGS (T_{amb} = 25 °C, unless otherwise specified)
TLLG440., TLLR440., TLLY440.

PARAMETER	TEST CONDITION	SYMBOL	VALUE	UNIT
Reverse voltage		V_R	6	V
DC forward current		I_F	7	mA

Tabela 9. Tabela de características dos leds (fonte: [43])

Utilizando uma resistência de 330Ω mantemo-nos dentro dos limites de corrente tanto do led como das GPIO:

$$V_{out} = U_R + V_{F_{LED}} \Leftrightarrow V_R = 3.3 - 1.9 = 1.4\,V \qquad (36)$$

$$I_R = \frac{V_R}{R} = \frac{1.9}{330} \approx 4.2\,mA \qquad (37)$$

3.6.6. Esquema de comando

Estando definidas todas as interfaces que permitem que o controlador comande os diferentes sistemas pode-se então apresentar o esquema completo:

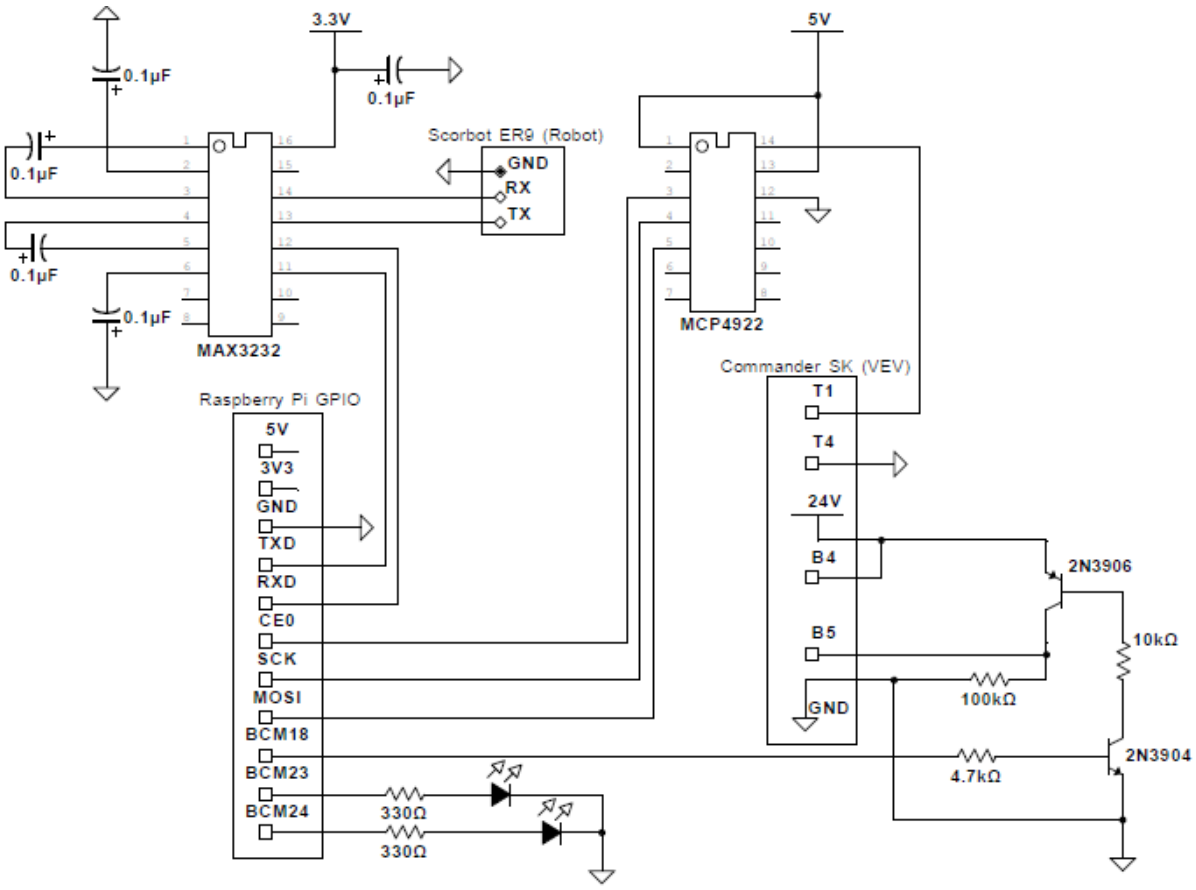


Figura 81. Esquema completo de interligação de interfaces

3.7. Software

Estando já descritos todos os subsistemas do projeto, é possível incluí-los no contexto do programa principal implementado no controlador. O software criado utilizando a linguagem de programação C++ foi dividido em dois programas distintos, sendo um o programa principal que coloca em funcionamento todo o sistema desenvolvido, e o outro uma ferramenta de configuração / calibração dos sistemas de processamento de imagem implementados. As calibrações uma vez finalizadas podem ser guardadas pelo programa num ficheiro de texto de configuração devidamente identificadas, o qual é lido no arranque do sistema.

O programa principal (Figura 82) é iniciado pelo utilizador com a hipótese de executar a inicialização do robot (Figura 84). Esta opção é importante pois sempre que o robot é alimentado necessita que esta operação seja realizada. Antes de se poder efetuar qualquer ligação aos subsistemas, é carregada a livraria responsável pela gestão dos IO's e comunicações (WiringPi©) e são então abertas as ligações.

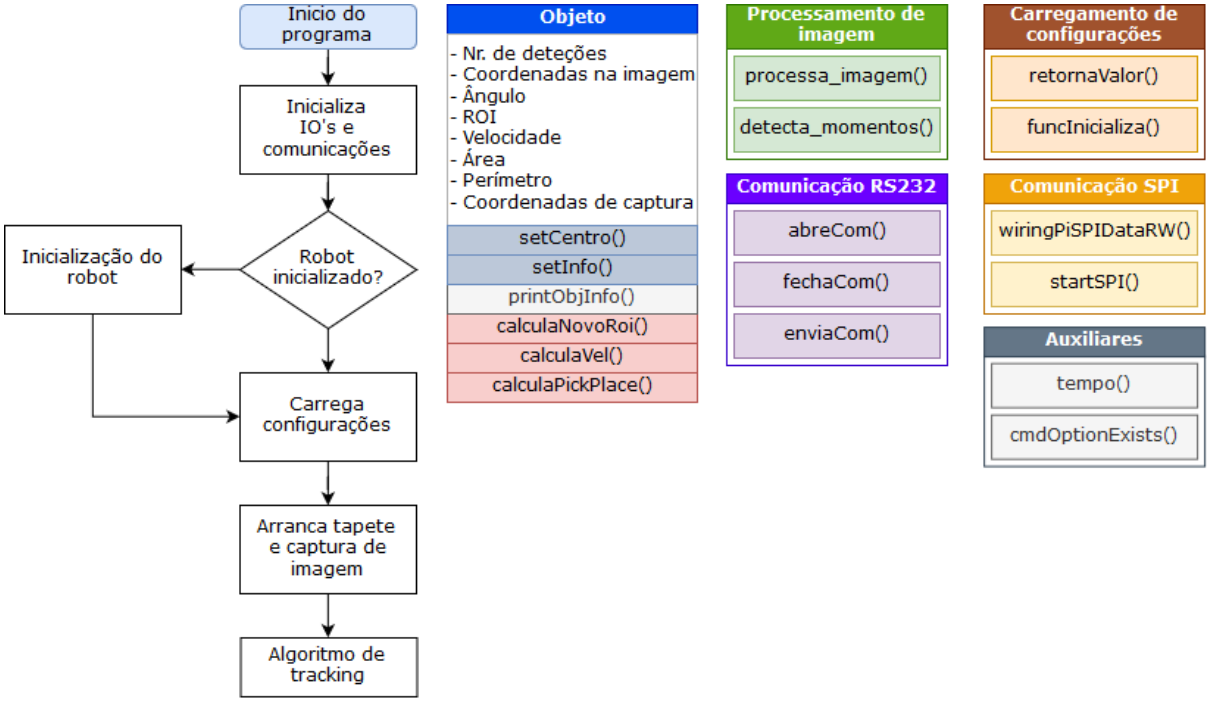


Figura 82. Organização do programa principal

Funcionando em "tempo real", o programa corre de forma cíclica o algoritmo de acompanhamento de objetos descrito no capítulo 3.5. Não havendo nenhuma interação entre o utilizador e o controlador durante o decorrer do programa, os resultados mais significativos em formato de texto relativos ao funcionamento do mesmo vão sendo apresentados (Figura 83) em lista durante a execução (tempos, coordenadas, localizações, etc), e os resultados gráficos que mostram as deteções (na *ROI Inicial* e na *ROI de tracking*).

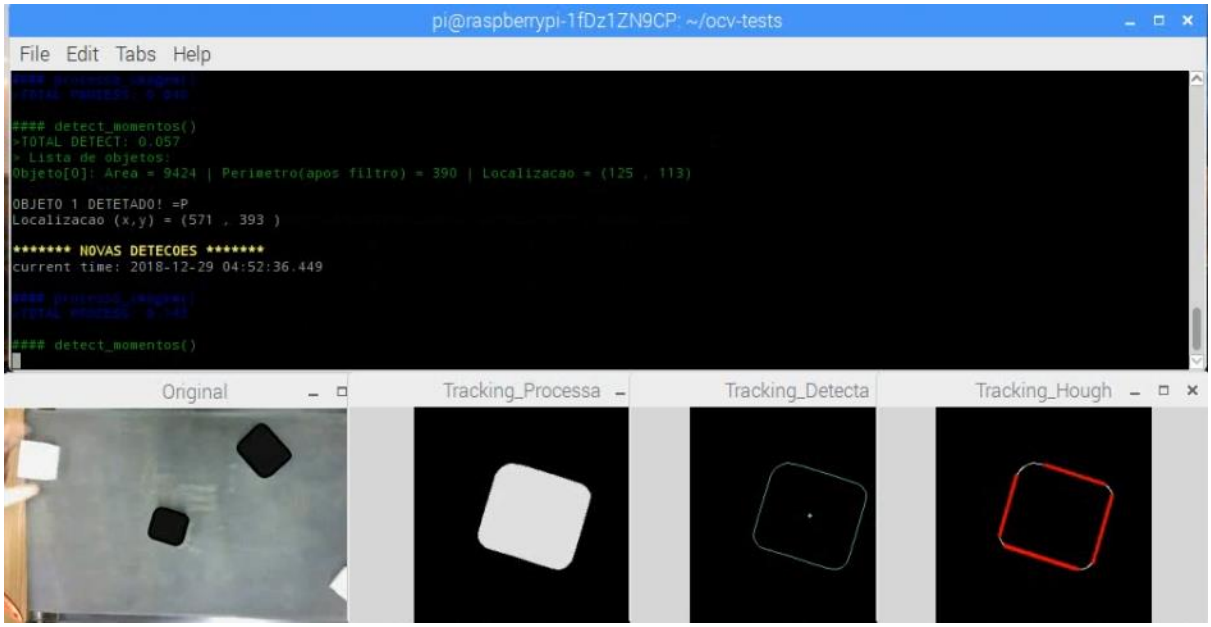


Figura 83. Resultados gráficos da execução do programa

As funções criadas no decorrer do desenvolvimento do projeto foram-se mantendo organizadas de acordo com os subsistemas utilizados. Estas estão organizadas em 6 grupos:

- **Processamento de imagem**: A solução de imagem está dividida em 2 subsistemas com 2 funções correspondentes. A primeira (*processa_imagem()*) é dedicada à filtragem inicial das imagens capturadas de forma a obtermos a informação o mais perto possível do necessário para tratamento de dados. De seguida a função *detecta_momentos()* é executada a fim de obter toda a informação necessária para trabalhar o acompanhamento dos objetos.

- **Comunicação RS232:** A comunicação entre o Raspberry Pi e o Robot é efetuada em comunicação série via RS232. Para tal foram criadas duas funções dedicadas a abrir e fechar a porta de comunicação com o robot (*abreCom()* e *fechaCom()*) e uma função para enviar os dados (*enviaCom()*).

- **Comunicação SPI:** A comunicação com o VEV que aciona o transportador, é feita com recurso a uma interface SPI. Para tal, foi utilizada a função *startSPI()* para estabelecer a comunicação e a função *wiringPiSPIDataRW()* para realizar a transferência de dados.

- **Carregamento de configurações:** A ferramenta de calibração gera um ficheiro de texto *config.txt*. Embora o programa arranque com variáveis por defeito (já com valores disponíveis), é necessário atualizar as mesmas para cada situação diferente de funcionamento do sistema. Assim, no inicio do programa a função *funcInicializa()* lista o carregamento dos valores do ficheiro para todas as variáveis necessárias enquanto a função *retornaValor()* trata do acesso ao ficheiro e leitura de cada uma das variáveis indicadas.

- **Auxiliares:** Embora durante o funcionamento do programa sejam passadas para o utilizador o tempo de ciclo do programa e de execução algumas das funções associadas, por vezes saber a hora atual de sistema, ajuda a ter uma perceção melhor dos tempos de execução do programa, visto que a própria chamada das funções de tempo gasta tempo de ciclo. Assim foi adicionada a função tempo() para devolver esta informação.

- Sempre que é ligada a alimentação do robot é necessário realizar a operação de referenciação das juntas do robot. Foi assim adicionado um comando (Figura 84) ao programa onde é possível:
    - Realizar a referenciação do robot e de seguida colocar na posição de inicial do sistema (posição '*river*')
    - Apenas enviar o robot para a posição inicial do sistema
    - Mover o robot para uma posição pré-definida manualmente (*PA*)

```
**********    INICIO DE PROGRAMA    **********
./progTrackAndPick42683 [ -i ] [ -river ] [ -PA ]
   [ -i ] Realizar Homing do robot ( +river )
   [ -river ] Mover robot para posicao 'river'
   [ -PA ] Mover robot para posicao PA
```

Figura 84. Lista de argumentos a utilizar para inicialização do robot

- A função *cmdOptionsExists()* auxilia na passagem dos argumentos para o programa.
- **Objeto:** Por fim, este grupo corresponde a um objeto de C++ que agrega todas as informações relativas aos objetos físicos a acompanhar pelo programa. Todas os dados como número de deteções ao longo do programa, localização do objeto, região de interesse (ROI), velocidade, características físicas, e ponto de captura são guardados neste objeto que tem também associado funções dedicadas a trabalhar os dados indicados. Ao longo do programa em várias ocasiões o algoritmo de acompanhamento e captura de objetos necessita de recorrer a este conjunto de dados e funções (Figura 85):
    - *setCentro()*: guarda dados do centro de massa do objeto
    - *setInfo()*: guarda dados de área, perímetro e ângulo do objeto
    - *printObjInfo*: mostra ao utilizador os dados guardados
    - *calculaNovoRoi()*: calcula a nova Região de Interesse do objeto
    - *calculaVel()*: calcula a velocidade dos objetos em cm/s
    - *calculaPickPlace()*: calcula as coordenadas de captura e verifica se estão dentro dos limites do espaço de trabalho do robot

```
Dados Item - ultima detecao [1]: Detecao nr.: 7
 | Area: 12947 px | Perimetro: 457 px | Angulo: -172 graus
Localizacao(x,y): ( 537 , 180 ) px
roiX: 427 px | roiY: 65 px | roiWidth: 242 px | roiHeight: 228 px
VelX: 3.34199 cm/s | VelY: 0.0821389 cm/s
```

Figura 85. Resultado do comando printObjInfo()

Ao longo do funcionamento do programa, diversas variáveis são tidas em conta, que podem ou não resultar de fatores externos: fala-se da iluminação, da posição de montagem da câmara e do robot, do estado do tapete transportador, da câmara utilizada, entre outros.

Todos estes fatores podendo ser alterados a qualquer altura, obrigam à alteração de diversos parâmetros de modo a que com alguma flexibilidade se deixe o sistema o mais próximo possível da solução ideal para cada situação. Assim foi criado um programa de calibração com o objetivo de fornecer ferramentas que possam facilitar na configuração do sistema sem que para isso seja preciso estar a editar o código do programa ou a alterar diretamente ficheiros de texto (Figura 86).

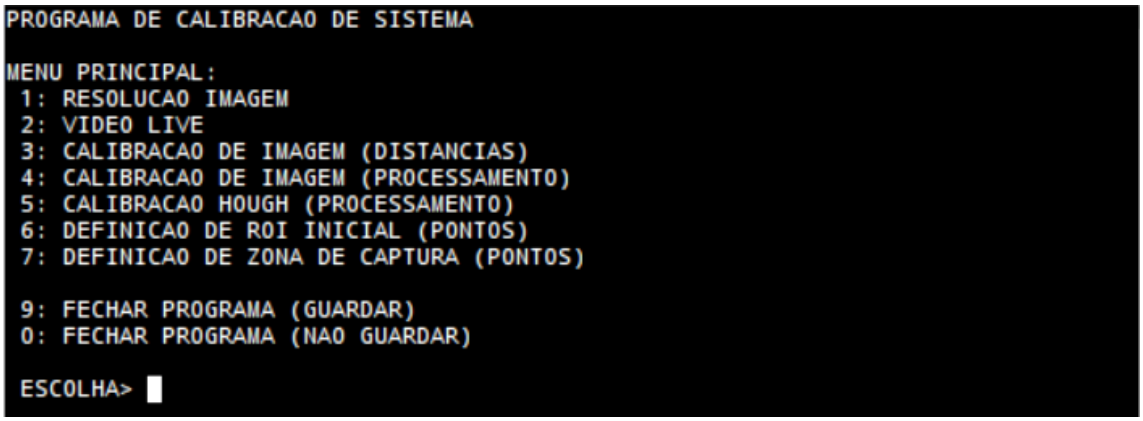


Figura 86. Menu inicial do programa de calibração

Este programa contém as mesmas funções de processamento de imagem utilizadas no programa principal e trabalha diretamente sobre os parâmetros destas. No entanto os valores apenas são guardados caso o utilizador assim o pretenda. Para além dos pontos já referidos ao longo do atual capítulo 3, ficou a faltar referir as opções que permitem definir a resolução de imagem da câmara de dentro das hipóteses disponíveis (opção 1), e que permitem ver em direto as imagens que a câmara está a capturar com o objetivo de auxiliar no posicionamento da mesma.

4. Resultados

Após fazer as respetivas calibrações do programa, através da aplicação auxiliar, foi possível obter resultados positivos relativamente aos objetivos propostos. Em primeiro lugar foi interessante analisar os tempos de processamento de todo o sistema. Estes foram verificados independentemente nas funções *processa_imagem()* (filtragem) e *detecta_momentos()* (deteção de objetos). Interessava em primeiro lugar quantificar a diferença entre realizar a filtragem de toda a imagem capturada e apenas a de uma zona do tapete (ROI inicial). O resultado foi um aumento aproximadamente proporcional à diferença das dimensões das áreas a analisar no tempo de processamento.

Nas Figura 87 e Figura 88 verifica-se que o tempo de filtragem da imagem passou de $0.695s$ para $0.060s$ numa proporção equivalente às áreas a analisar, 1280x720 e 432x181 pixeis respetivamente. Relativamente à deteção de objetos, não foi é possível comparar diretamente visto no primeiro caso serem sempre detetados objetos não válidos (em zonas com fundos não adequados à calibração).

$$\frac{1280 * 720 \text{ px}}{432 * 181 \text{ px}} = 11.786 \qquad \frac{0.695 \text{ s}}{0.060 \text{ s}} = 11.583$$

```
**************************     NOVO CICLO | n. 11    ******************************

******* NOVAS DETECOES *******
Tempo atual: 2018-12-28 20:58:50.842

#### processa_imagem()
>Tempo total: 0.695

#### detecta_momentos()
>TOTAL DETECT: 0.097
> Lista de objetos:
Objeto[0]: Area = 19 | Perimetro(apos filtro) = 508 | Localizacao = (759 , 709)
Objeto[1]: Area = 384 | Perimetro(apos filtro) = 85 | Localizacao = (26 , 664)
Objeto[2]: Area = 284 | Perimetro(apos filtro) = 68 | Localizacao = (55 , 319)

NR OBJETOS DETETADOS: 3  | ciclo: 12
>TOTAL CICLO: 0.810 s
Tempo atual: 2018-12-28 20:58:51.637
```

Figura 87. Tempos de filtragem com a imagem completa 1280x720 pixeis

```
**************************     NOVO CICLO | n. 53    ******************************

******* NOVAS DETECOES *******
Tempo atual: 2018-12-28 20:54:57.382

#### processa_imagem()
>Tempo total: 0.060

#### detecta_momentos()
>TOTAL DETECT: 0.002
> Lista de objetos:

NR OBJETOS DETETADOS: 0  | ciclo: 54
>TOTAL CICLO: 0.080 s
Tempo atual: 2018-12-28 20:54:57.446
```

Figura 88. Tempos de processamento com uma ROI Inicial de 432x181 pixeis

Analisar um objeto a ser acompanhado demora como se pode ver na Figura 89, 0.045 segundos $(0.036 + 0.009)$. Contabilizando um ciclo completo ficamos com um tempo total de 0.21 segundos.

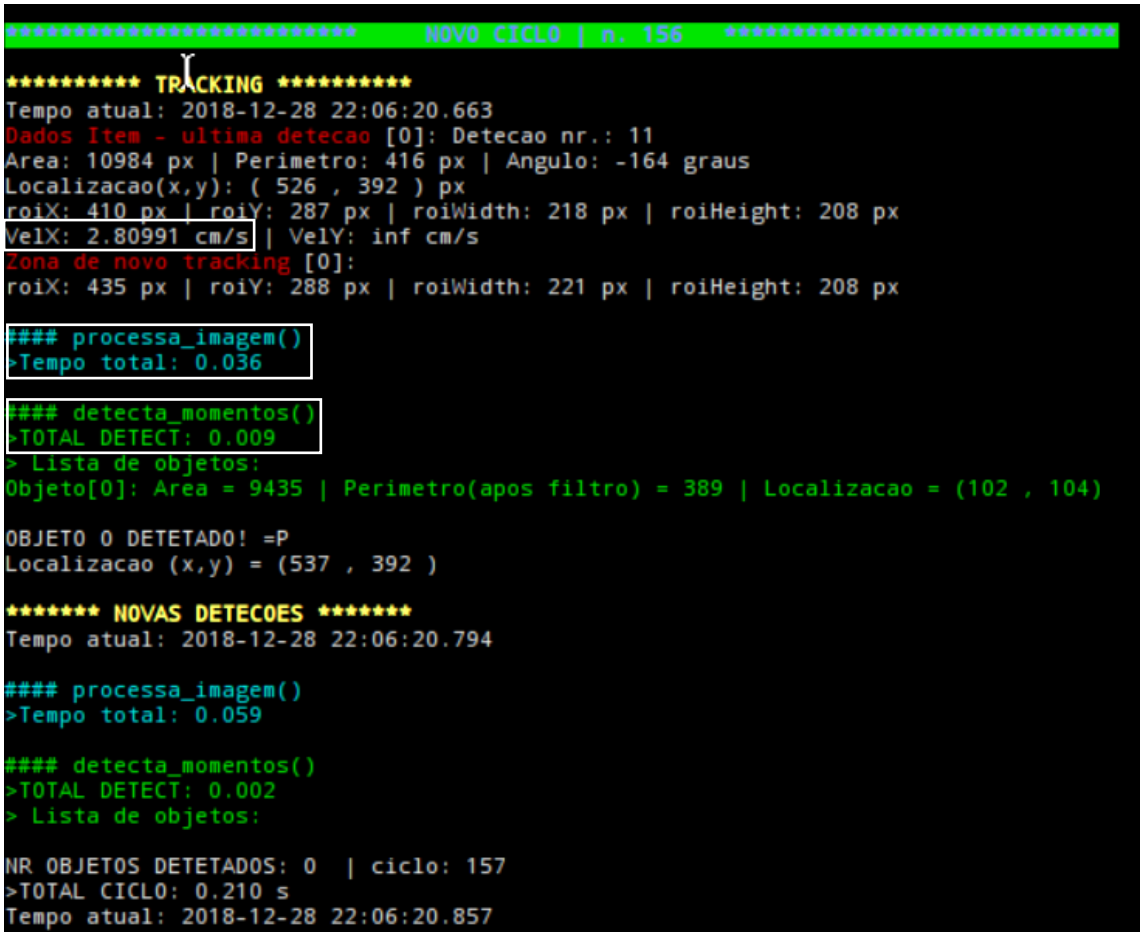


Figura 89. Ciclo com 1 peça em tracking

Por fim, foi possível realizar a captura dos objetos com sucesso. Na Figura 90 pode-se observar os comandos a serem enviados para o robot para efetuar a captura.

```
roiX: 450 px | roiY: 66 px | roiWidth: 246 px | roiHeight: 228 px

#### processa_imagem()
>Tempo total: 0.090

#### detecta_momentos()
>TOTAL DETECT: 0.016
> Lista de objetos:
Objeto[0]: Area = 12094 | Perimetro(apos filtro) = 442 | Localizacao = (114 , 114)

OBJETO 1 DETETADO!
Localizacao (x,y) = (564 , 180 )
OBJETO ATINGIU PONTO DE CAPTURA
Apagar objeto...
nr. do objeto a capturar: 0
# enviaCom() msg> 'SETPVC PA1 X 8'
# enviaCom() msg> 'SETPVC PA1 Y -427'
# enviaCom() msg> 'SETPVC PA1 Z 330'
# enviaCom() msg> 'SETPVC PA1 R 16'
# enviaCom() msg> 'MOVE PA1'
# enviaCom() msg> 'SETPVC PA1 Z 230'
# enviaCom() msg> 'MOVE PA1'
```

Figura 90. Sequência de envio de comandos de captura para o robot

Por fim, na Figura 91 podemos observar o objeto no inicio do tapete, com o robot na sua posição inicial, e na Figura 92 a realizar o movimento de captura do objeto devidamente alinhado com este.

Figura 91. Peça em deslocação e robot na posição inicial

Figura 92. Peça em posição final e robot a realizar o movimento de captura

5. Conclusões

5.1.1. Análise de resultados

Se considerarmos que a câmara captura imagens a uma velocidade máxima de 30fps o equivalente a 0.033 segundos ($\frac{1}{30}$) por frame, chega-se à conclusão de que por cada ciclo de processamento (0.21s) não são analisados 6 frames ($\frac{0.21}{0.033}$). Tendo em conta que a velocidade de deslocamento das peças rondava os 2.8cm/s (Figura 89) e que o programa realizava 4.76 ciclos de programa por segundo ($\frac{1}{0.21}$), significa que é possível atualizar a posição do objeto a cada 0.59cm $\left(\frac{2.8}{4.76}\right)$. Embora numa primeira análise se possa considerar que os valores não são muito animadores, se compararmos a resolução utilizada (1280x960) com o que é considerado a gama de entrada de soluções industriais (abaixo 800x600), podemos verificar que com um equipamento com um custo dezenas de vezes inferior ao que existe no mercado de soluções dedicadas, trabalhou-se com dados 2.56 vezes acima da referência. Adicionalmente o facto de se estar a vigiar a velocidade dos objetos permite que sejam capturados com uma precisão superior.

Relativamente ao processo de captura dos objetos, não foi possível avaliar efetivamente a velocidade de captura dos objetos por parte do robot (apesar de o mesmo continuar operacional), com o restante sistema em funcionamento. Durante os testes foram recebidas frequentemente indicações sonoras de erro por parte do robot quando os tempos entre mensagens eram inferiores a 1 segundo. Embora não tenham sido analisadas ao pormenor as razões para tal, algumas possíveis causas poderiam passar pela blindagem inexistente na cablagem entre o Raspberry Pi e a ficha DB9 do robot, ou pela utilização de uma outra interface de comunicação com o robot.

Quanto ao transportador, o funcionamento correspondeu ao espectável, com o DAC a funcionar com ambos os equipamentos como pretendido e com o variador a responder corretamente aos sinais de variação de velocidade e de arranque e paragem. O fundo do tapete que foi desde início detetado como um possível problema devido a não ser completamente preto, foi suprimido com sucesso pela filtragem de imagem.

5.1.2. Trabalho futuro

Apesar de não ser o principal objetivo a performance do sistema, os resultados obtidos do desenvolvimento do projeto foram animadores. Para além do referido no ponto anterior, foi possível realizar uma comparação entre soluções industriais já existentes, com produtos não industriais, mas mais flexíveis e com maior liberdade para desenvolvimento. O Raspberry Pi revelou-se uma boa ferramenta para o desenvolvimento laboratorial de soluções antes de

serem submetidas às necessidades e influencias agressivas do meio industrial. Foram assim estudadas algumas soluções já existentes e explorados alguns temas em específico que permitem dirigir diversos estudos em torno de uma solução corrente em sistemas industriais.

Do ponto de vista de redes industriais, como referido anteriormente existe a possibilidade de descentralizar cada um dos subsistemas ficando cada um com um controlador dedicado. A sequência natural passa então por ter um controlador central interligado a mais equipamentos de modo a se poder juntar à aplicação valências de diferentes soluções a desenvolver tais como o controlo de um processo de fabrico.

Existindo uma continuidade em futuros projetos do trabalho desenvolvido neste, e utilizando o mesmo robot, seria importante analisar as falhas de comunicação que foram detetadas de modo a criar um sistema mais robusto e que permitisse um controlo em “tempo real” do equipamento.

Em relação ao processamento de imagem, foram estudados vários temas que podem servir de base para o estudo e desenvolvimento de sistemas de visão artificial mais complexos. Numa altura em que por todo o mundo as câmaras de vídeo se tornam cada vez mais um equipamento banal, e o especializado torna-se comum, o mercado procura melhores soluções e mais fiáveis, tanto a nível de performance como de diversidade de funções. Foi visto que várias técnicas de processamento de imagem, apesar de já há muito desenvolvidas, continuam a ter grande aplicação pela sua simplicidade e otimização, e continuam a servir de referência na metodologia para o desenvolvimento de novos algoritmos.

Por fim, a inteligência artificial sendo uma realidade aos dias de hoje, tem aplicação na otimização de qualquer um dos tópicos acima referidos. Falando especificamente do projeto executado, tanto a *ROI Inicial* como a *ROI do objeto* poderiam ser alterados dinamicamente durante o funcionamento do programa a partir de um algoritmo que previsse o comportamento dos objetos mediante o seu formato. Isto iria permitir que o programa deteta-se não só objetos que alterassem as suas trajetórias (como no caso do filtro de *Kalman*) mas também objetos que pudessem sofrer alterações estruturais ao longo dos movimentos (resultado por exemplo da variação de reflexos luminosos). Com esta capacidade seria também possível otimizar os movimentos dos equipamentos de transporte (variar a/o velocidade/binário do motor mediante a carga prevista) e dos equipamentos de captura de objetos (trajetórias automaticamente otimizadas). Por fim seria possível otimizar a coordenação entre todos os equipamentos existentes numa aplicação expandida.

6. Bibliografia

[1] S. Bennett, “A Brief History of Automatic Control,” 1996.

[2] M. Wollschlaeger, T. Sauter e J. Jasperneite, “The Future of Industrial Communication,” 2017.

[3] ISO, “35.100 Open systems interconnection (OSI),” [Online]. Available: https://www.iso.org/ics/35.100/x/.

[4] “Rise of the logistics robots,” [Online]. Available: http://industrial.embedded-computing.com.

[5] ISO, “ISO 8373:2012 Robots and robotic devices — Vocabulary,” 2012.

[6] ANSI/RIA, “ANSI/RIA R15.06-2012 - Industrial Robots and Robot Systems —Safety Requirements,” 2012.

[7] J. Wallén, “The history of the industrial robot,” Linköpings universitet, 2008.

[8] International Federation of Robotics, “Robot History,” [Online]. Available: https://ifr.org/robot-history.

[9] ABB, “Historical Milestones,” [Online]. Available: https://new.abb.com/products/robotics/home/about-us/historical-milestones.

[10] P. I. Corke, Visual Control of Robots - high-performance visual servoing, 1996.

[11] S. L. Wilson, “Control of Scar Tissue Formation in the Cornea: Strategies in Clinical and Corneal Tissue Engineering,” [Online]. Available: https://www.researchgate.net/publication/236172788.

[12] Cognex, “Machine Vision,” [Online]. Available: https://www.cognex.com/.

[13] National Instruments, “Machine VIsion,” [Online]. Available: http://www.ni.com.

[14] National Center for Biotechnology Information, “Viewpoints on Medical Image Processing: From Science to Application,” [Online]. Available: https://www.ncbi.nlm.nih.gov.

[15] V. Sundari, S. Rani e S. Prabhu, Tracking of Moving Objects in Video Sequences, 2018.

[16] Wikipédia, “Light,” [Online]. Available: https://en.wikipedia.org/wiki/Light.

[17] Allied Vision, “CCD or CMOS: can CMOS sensors replace CCDs in all cases?”.

[18] J. N. Burghartz, H.-G. Graf e C. Hare, “HDR CMOS Imagers and Their Applications,” 2006.

[19] S. Milani, M. Fontani, P. Bestagini, M. Barni, A. Piva, M. Tagliasacchi e S. Tubaro, “An overview on video forensics,” 2012.

[20] ITU-R, “Recommendation ITU-R BT.601 - Studio encoding parameters of digital television for standard 4:3 and wide-screen 16:6 aspect ratios,” 2011.

[21] Fernando Oliveira Nunes e G. Almeida, “Sistemas Robóticos,” 2008.

[22] A. Koschan e M. Abidi, Digital Color Image Processing, 2008.

[23] J. Canny, “A Computational Approach to Edge Detection,” 1986.

[24] J. Seo, S. Chae, J. Shim, D. Kim, C. Cheong e T.-D. Han, “Fast Contour-Tracing Algorithm Based on a Pixel-Following Method for Image Sensors,” MDPI - Sensors, 2016.

[25] S. Suzuki e K. Abe, “Topological Structural Analysis of Digitized Binary Images by Border Following,” 1985.

[26] M.-K. Hu, “Visual Pattern Recognition by Moment Invariants,” 1962.

[27] J. Flusser e T. Suk, “Pattern recognition by affine moment invariants,” 1993.

[28] P. Hough, “Method and means for recognizing complex patterns,” 1962.

[29] United for Efficiency (UN), “U4E Policy Guide Series – Energy-Efficient Electric Motors and Motor Systems,” Energy-Efficient , 2017.

[30] M. Guarnieri, “The Beginning of Electric Energy Transmission: Part Two,” IEEE, 2013.

[31] J. C. P. Palma, Accionamentos electromecânicos de velocidade variável, 2008.

[32] J. C. P. Palma, Accionamentos de Velocidade Variável, 2008.

[33] Control Techniques, “Commander SK - Getting Started Guide,” 2013.

[34] Eshed Robotec, “ACL Reference Guide for Controller-A,” 1999.

[35] Raspberry Pi Foundation , “Raspberry Pi Documentation,” [Online]. Available: https://www.raspberrypi.org/documentation/faqs/#powerReqs.

[36] “Raspberry Pi Pinout,” [Online]. Available: https://pinout.xyz/.

[37] Texas Instruments, “MAX3232 3-V to 5.5-V Multichannel RS-232 Line Driver/Receiver,” 2017.

[38] The Open Group, “The UNIX® Standard,” [Online]. Available: https://www.opengroup.org/membership/forums/platform/unix.

[39] The Open Group, “<termios.h>,” 1997. [Online]. Available: http://pubs.opengroup.org/onlinepubs/7908799/xsh/termios.h.html.

[40] On Semiconductor, “2N3903, 2N3904 - General Propose Transistors NPN Silicon,” 2012.

[41] ON Semiconductor, “2N3906 - General Purpose Transistors PNP Silicon,” 2010.

[42] Microchip, “8/10/12-Bit Dual Voltage Output Digital-to-Analog Converter,” 2010.

[43] Vishay Semiconductors, “TLLG440., TLLR440., TTLY440. Low Current LED in d3 mm Tinted Diffused Package,” 2017.

[44] International Electrotechnical Commission, “IEC - Electromagnetic compatibility,” [Online]. Available: https://www.iec.ch/emc/.

[45] Schneider Electric, “50 Years of Modicon,” [Online]. Available: https://www.schneider-electric.co.in/en/about-us/events/modicon.jsp.

[46] Julabo, “Highly Dynamic Temperature Control Systems,” [Online]. Available: https://www.julabo.com.

[47] M.Hermann, T. Pentek e B. Otto, “Design Principles for Industrie 4.0 Scenarios,” 2016.

[48] PI North America, “A Beginner’s Guide to PROFINET,” [Online]. Available: https://us.profinet.com/beginners-guide-profinet/.

[49] S. Rani, V. Sundari e S. Prabhu, “Tracking of Moving Objects in Video Sequences”.