Luís Pedro Nabais Gonçalves

Licenciado em Ciências da Engenharia Eletrotécnica e de Computadores

Reconhecimento de Gestos para Interação com Robô Móvel em Ambiente Pediátrico

Dissertação para obtenção do Grau de Mestre em
Engenharia Eletrotécnica e de Computadores

Orientador: André Teixeira Bento Damas Mora
Professor Auxiliar, FCT-UNL
Coorientador: José Manuel Matos Ribeiro da Fonseca
Professor Auxiliar com Agregação, FCT-UNL

Presidente: Prof. Doutor João Almeida das Rosas
Arguente: Prof. Doutor Filipe de Carvalho Moutinho
Vogal: Prof. Doutor André Teixeira Bento Damas Mora

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Reconhecimento de Gestos para Interação com Robô Móvel em Ambiente Pediátrico

Resumo

Nos dias de hoje o desenvolvimento de aplicações que visam o bem-estar e o divertimento de crianças que se encontram com a saúde debilitada e que passam grande parte do seu tempo em hospitais, tem vindo a aumentar. O projeto retratado neste documento surge de uma colaboração com o Hospital Fernando Fonseca (Amadora/Sintra), com vista à dinamização de uma plataforma móvel autónoma já existente, a tartaruga Nando, e que será utilizada para animação e interação com crianças que se encontrem no hospital. O objetivo é, assim, desenvolver uma aplicação de reconhecimento de gestos baseado em visão por computador integrado nesse sistema robótico.

A plataforma *hardware* que suporta este projeto é um Raspberry Pi. Através deste dispositivo e com a utilização da linguagem de programação Python, é possível interpretar e descodificar movimentos captados por uma câmara (*Camera module V2)*, através de processamento de imagem, tendo como base a biblioteca OpenCV.

A escolha do Raspberry Pi teve por base o seu baixo custo e reduzida dimensão, o que faz com que seja possível transportar e instalar na plataforma robótica (tartaruga). Esta tartaruga é um robô que interage com as crianças através de sons, reproduzidos consoante o movimento executado pela criança que foi detetada pela câmara.

Através de Histogramas de Gradientes Orientados (HOG), de classificadores em cascata e posteriormente através de *background subtraction*, a aplicação consegue identificar uma pessoa e posteriormente reconhecer os movimentos por ela executados.

O sistema desenvolvido está preparado para reconhecer seis tipos de movimentos, com os membros superiores, depois de identificada uma pessoa. Para cada movimento, ascendente ou descendente, à direita ou à esquerda, bem como acima do ombro, à direita ou à esquerda, é emitido um som pré-definido diferente dependendo do mesmo.

Palavras-chave: Raspberry Pi, Reconhecimento de gestos, Python, OpenCV, Processamento de imagem, Sistema Robótico

Abstract

Nowadays, the development of applications aimed the well-being and fun of children who find themselves in complicated situations and spend much of their time in hospitals, has been increasing. The project presented arises from a collaboration with Hospital Professor Doutor Fernando Fonseca (Amadora/Sintra), in Lisbon, aiming to stimulate an existing mobile platform, a turtle named Nando, which is used for animation and interaction with children who are in the hospital. Therefore, the main goal is to develop an application of gesture recognition, based on computer vision, integrated in this robotic system.

The computing system that supports this project is Raspberry Pi. Through this device and the use of the Python programming language, it is possible to interpret and decode movements captured by a camera (camera module V2), through image processing, based on the OpenCV library.

Raspberry Pi choice is based on its low cost and small size, which makes it possible to transport and install it into a previously constructed robotic platform, in the form of a turtle. This turtle is the robot that interacts with the children, through sounds, reproducing different sound depending on the movement performed by the child that is detected in the camera.

Image processing is very important for the success of this application. By using Histograms of Oriented Gradients (HOG), cascaded classifiers, and then through the background subtraction, this application can identify a person to begin the interaction and subsequently the movements performed by the same person.

In this paper, the system is prepared to recognize six types of movements, with the upper limbs, after identified a person. To each movement, upper or downer, to the right or left, as well as above the shoulder (right or left), is emitted a pre-defined sound depending on the movement.

Keywords: Raspberry Pi, Gesture Recognition, Python, OpenCV, Image Processing, Robotic System.

Índice Geral

Lista de Siglas e Acrónimos

3D	Três Dimensões
2D	Duas Dimensões
HD	*High Definition* (Alta Definição)
UDF	*U-type Depth Feature* (Característica de Profundidade)
CRF	*Convex Region Feature* (Característica de Região Convexa)
CDF	*Convex Degree Feature* (Característica de Grau Convexo)
CDI	*Convex Degree Image* (Imagem de Grau Convexo)
DVI	*Digital Visual Interface* (Interface Digital de Vídeo)
RPI	*Raspberry Pi*
OpenCV	*Open Source Computer Vision Library*
IPP	*Integrated Performance Primitives*
HOG	*Histogram of Oriented Gradients* (Histograma de Gradientes Orientados)
ROI	*Region of Interest* (Região de Interesse)
XML	*Extensible Markup Language* (Linguagem Extensível de Marcação)
MOG	*Mixture Of Gaussians*
FPS	*Frames Per Second* (Imagens Por Segundo)

Índice de Tabelas

Índice de Figuras

1 Introdução

A animação pediátrica em hospitais é um tema que tem vindo a ser estudado e aprofundado nos dias de hoje. A interação com as crianças em ambiente de hospital apresenta uma solução para o melhoramento e desenvolvimento cognitivo das mesmas. O ato de brincar, constitui uma prática que leva ao bem-estar e à socialização das crianças (Carmo, 2013),(Carvalho & Begnis, 2006).

O desenvolvimento de soluções lúdicas para interação com robôs é uma forma de ajudar as crianças hospitalizadas, fazendo com que o tempo que estas passam em hospitais não seja notório e aborrecido para elas. Assim, aquilo que, à partida seria desagradável, poder-se-á tornar em algo divertido, fazendo esquecer um pouco o sentimento de medo ou preocupação habituais nestes locais.

Os gestos são uma das partes mais importantes da comunicação humana, onde a forma das mãos ou os movimentos e orientação dos braços contém uma vasta quantidade de informação que pode originar muitos tipos de interpretações. A capacidade de reconhecer gestos representa uma forma de interpretação e descodificação das informações transmitidas pelos humanos (Pok, 2004). Esta capacidade é uma das principais motivações para a resolução do problema abordado neste projeto.

O Reconhecimento de Gestos é um tema complexo que envolve muitos e diferentes tipos de abordagens, tais como: a modelação de movimentos, o reconhecimento de padrões, "*Machine Learning"* e por vezes alguns estudos psicolinguísticos (Wu & Huang, 1999).

Em relação à abordagem de modelação de movimentos, uma possível aplicação seria a classificação de movimentos em vídeos desportivos. Por exemplo, no ténis pode querer-se consultar um arquivo de vídeo para contar todos os casos em que um jogador foi até à rede e voltou. Esta abordagem eliminaria a necessidade de uma análise por parte de um humano num grande conjunto de dados que esse vídeo teria (Gavrila, 1999). Basicamente é uma abordagem que serve para identificar facilmente movimentos básicos numa grande quantidade de diferentes dados, numa imagem ou vídeo. A modelação de movimentos é a abordagem que se enquadra no tema abordado neste documento.

O reconhecimento por gestos pode ser aplicado em várias situações, como por exemplo (Lisetti & Schiano, 2000):

- Conceção de técnicas de identificação forense;
- Sistemas de auxílio para deficientes auditivos;
- Sistemas de deteção de mentiras;
- Monitorização de pacientes e dos seus níveis de *stress*;
- Comunicação por videoconferência;
- Vigilância de estado de alerta/sonolência dos condutores de automóveis;
- Ensino à distância;
- Navegação ou manipulação em ambientes virtuais;
- Reconhecimento de linguagem de sinais;

De forma a fazer o reconhecimento de gestos através de uma câmara, é necessário o processamento e tratamento da imagem proveniente da mesma. Para tal, existem duas formas de abordar este problema, o processamento da imagem em 3D ou em 2D.

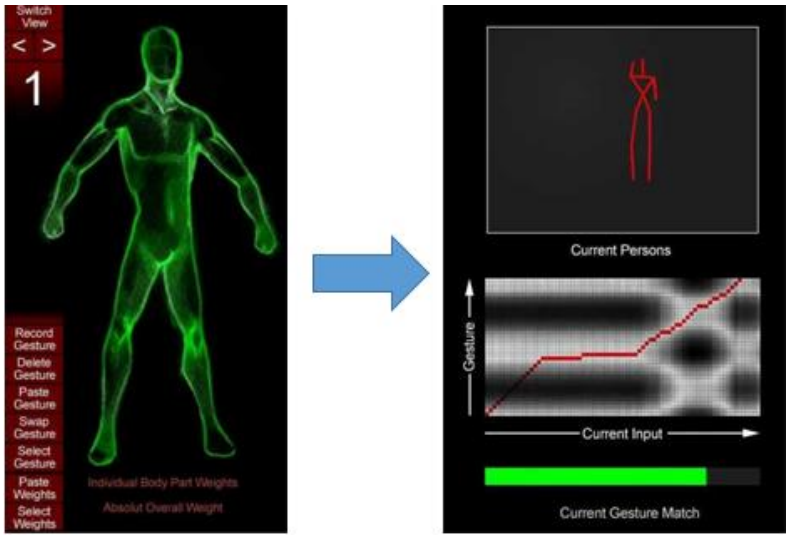

Figura 1.1 - Exemplo de análise de uma imagem em 3D através do *software* Kinetic Space 2.0, onde são utilizadas câmaras de profundidade, adaptado de [Adaptado de ("Google Code Archive - Kinetic Space 2.0 (Open source)," 2011).]

Quando se analisa a imagem em 3D (Figura 1.1), esta requer a utilização de sensores ou câmaras de profundidade (Suarez & Murphy, 2012). Um dos exemplos mais conhecidos é o da utilização de câmaras *kinect* que utilizam vetores ou matrizes para poder adquirir os dados da imagem e detetar o objeto pretendido através dos valores de três dimensões (Patsadu et al., 2012). Nestes casos, utilizam-se, geralmente, métodos de classificação de *data mining* para descodificar ou classificar determinados tipos de gestos humanos (Gorunescu, 2011; Patsadu et al., 2012), ou seja, são exploradas grandes quantidades de dados provenientes das imagens que formam o movimento, procurando padrões consistentes, para detetar relacionamentos sistemáticos entre variáveis. Assim, é

possível formar novos subconjuntos de dados para facilitar o reconhecimento do gesto executado pela pessoa.

Quando se trata uma imagem em 2D, existem estratégias que permitem retirar as características pretendidas. Uma das estratégias passa por desenvolver um algoritmo em que o objetivo é extrair campos de movimento bidimensionais de objetos, através de uma sequência de imagens baseada em trajetórias de movimento. Este método consiste em fazer uma comparação de determinadas regiões de imagens consecutivas, e através das diferenças entre as imagens nessas zonas, adquirir e classificar a que tipo de movimento correspondem. Estas comparações passam por correspondências de pixeis em duas *frames* consecutivas do vídeo captado (Yang et al., 2002).

No que diz respeito à identificação do sujeito que executa os gestos em estudo, a utilização de um sensor ou filtro de infravermelhos de onda longa (8 – 12 µm) acoplado à câmara, pode tornar o processo mais fácil, na medida em que não tem em conta cores ou sombras que muitas vezes dificultam a diferenciação entre o que é fundo e não interessa, e o dito sujeito. Com este sensor, a identificação é feita através da radiação eletromagnética emitida pelos objetos captados, onde os valores dos respetivos pixeis representam a sua temperatura (Arlowe, 1992; Han & Bhanu, 2005).

O caso de estudo deste projeto consistiu na captação de um indivíduo que se encontre num ambiente com sombras, podendo estar rodeado de vários objetos e por vezes num espaço com pouca luminosidade. Deste modo, o indivíduo poderia ser distinguível da restante imagem através da sua temperatura corporal, pela utilização de um sensor de radiação eletromagnética ou filtro de infravermelhos, conforme referido.

Com o aparecimento de vários computadores de placas únicas, tais como, Raspberry Pi, Arduino, BeagleBoard, CubieBoard, OLinuXino entre outros, a facilidade de desenvolver projetos fidedignos aumentou, devido, em grande parte, ao baixo consumo de energia associado a estas placas, que permite operar durante muito tempo recorrendo a uma bateria de baixo custo (Kochláň et al., 2014).

Neste projeto o Raspberry Pi (Figura 1.2) foi a plataforma escolhida como opção não intrusiva de baixo custo para tratar do reconhecimento de gestos através de uma câmara HD acoplada, que permite capturar e transmitir toda a informação da imagem para o Raspberry Pi com qualidade juntando ao baixo custo energético desta plataforma.

Figura 1.2 - Raspberry Pi 3 Model B, plataforma escolhida para o projeto desenvolvido (Raspberry Pi Foundation, 2016b).

1.1 Objetivos

Inserido numa questão social e com alguma relevância para a recuperação rápida e eficaz de algumas crianças no Hospital Professor Doutor Fernando Fonseca, o projeto a desenvolver tem como objetivo a interação entre uma criança e um robô. Esta interação é feita através de gestos executados pela criança, capturados pela câmara e descodificados pelo Raspberry Pi inserido no robô para que este execute uma determinada ação, neste caso emitir sons (frases) consoante cada gesto executado.

Os gestos por parte do humano servem como comandos para as ações do robô, para que cada criança que passar por este hospital possa participar num momento lúdico e abstrair-se um pouco do local e da situação em que se encontra.

Os movimentos dos membros superiores são os que entram em consideração como comandos para interagir com o robô, sendo considerados para reconhecimento, seis movimentos:

- Movimento ascendente do braço esquerdo;
- Movimento descendente do braço esquerdo;
- Movimento ascendente do braço direito;
- Movimento descendente do braço esquerdo;
- Movimento acima do antebraço acima do ombro esquerdo;
- Movimento acima do antebraço acima do ombro direito.

Estes movimentos servem de comandos para o robô, e este emite um som, pré-definido, associado a cada um destes movimentos. Neste processo o robô não se encontra e não tem possibilidade de estar em movimento, visto que não pode perder o contato visual com o indivíduo que está a executar os movimentos. Deste modo a câmara tem que permanecer estática.

A experiência de trabalhar com estas novas plataformas, como neste caso o Raspberry Pi, torna-se enriquecedora para uma conclusão de Mestrado da área de Engenharia Eletrotécnica e de Computadores. Um dos objetivos principais é também o de desenvolver este projeto neste tipo de plataforma de baixo custo, mantendo uma boa relação qualidade-preço.

1.2 Estrutura e Organização do Documento

Tendo em conta o que foi anteriormente exposto, este documento encontra-se organizado da seguinte forma:

O primeiro e presente capítulo, contém uma breve introdução e enquadramento ao tema abordado neste projeto, através da indicação de possíveis técnicas a utilizar em trabalhos deste tipo.

O segundo capítulo, diz respeito à descrição de diferentes projetos já desenvolvidos que se enquadram em aplicações com objetivos semelhantes ao abordado neste documento.

No terceiro capítulo, é feita uma descrição global do projeto, onde estão descritos tanto o *hardware* como o *software* utilizados e desenvolvidos neste projeto, com vista a demonstrar todo o material utilizado e o método de reconhecimento de pessoas e gestos.

O quarto capítulo descreve os testes e validações realizadas, os resultados obtidos pela aplicação desenvolvida, a discussão sobre problemas encontrados e as soluções desenvolvidas para os resolver.

As conclusões do trabalho realizado, retratando o que correu bem e mal ao longo da execução do projeto, encontram-se descritas no quinto capítulo.

O sexto e último capítulo apresenta o que poderá ser acrescentado e melhorado num trabalho futuro de continuação do que está feito, deixando assim as últimas impressões sobre o que está por melhorar e o que pode ser alterado.

Para perceber melhor como foi desenvolvida a aplicação correspondente ao *software* deste projeto, existe ainda um capítulo de Anexos, onde se encontra descriminado todo o código de porgramação em *Python*.

2 Estado da Arte

O tema do reconhecimento de gestos ou movimentos através de câmaras, tem vindo a ser muito aprofundado e trabalhado com o intuito de que os sistemas já existentes sejam melhorados, arranjando melhores respostas para problemas existentes na sociedade.

As abordagens de classificação de gestos ou movimentos são geralmente organizadas em duas categorias: abordagens baseadas em regras, nas quais os gestos são classificados a partir de regras previamente codificadas manualmente, e abordagens baseadas em *"machine learning"* que utilizam um conjunto de exemplos para deduzir modelos de gestos (Hassanpour et al., 2008).

Nesta secção são apresentadas algumas soluções existentes, inseridas na mesma área de aplicação da solução proposta neste documento. A pesquisa feita para encontrar aplicações relacionadas com o tema de visão por computador e reconhecimento de gestos é exposta neste capítulo.

2.1 Abordagens com base em exemplos dedutivos e/ou exemplos indutivos

O trabalho de Lementec e Bajcsy (Lementec & Bajcsy, 2004) está inserido na categoria de classificação de gestos baseados em regras, onde o foco está no reconhecimento de gestos dos braços utilizando sensores de orientação múltipla. Este trabalho foi estruturado da seguinte maneira: primeiro executaram uma modelação gestual utilizando ângulos de Euler (*roll, pitch* e *yaw),* depois caracterizaram os gestos de baixo nível, com a junção dos ângulos de Euler com a variação do referencial "*yaw",* e por fim utilizaram algoritmos de classificação de gestos baseados em modelos previamente codificados para posteriormente enviar os comandos ao robô, como está representado na Figura 2.1.

Figura 2.1 - Diagrama de fluxo do sistema de classificação de gestos baseados em regras para controlo do robô, de Lementec e Bajcsy [Adaptado de (Lementec & Bajcsy, 2004)].

A classificação de gestos é baseada nas características de baixo nível do fluxo de dados angular individual. Isto na prática significa que, quando são detetadas novas características de baixo nível, o classificador fica ativo. Utilizando a posição angular atual é-lhe atribuído um rótulo de gesto com base no modelo de gestos previamente embutido no sistema. Esta abordagem faz com que não seja preciso detetar as transições de um movimento para o outro (Lementec & Bajcsy, 2004).

Uma questão, que pode ser uma das limitações destes sistemas com este tipo de abordagens, é a delimitação de regras definidas e codificadas *a priori*, dando pouca autonomia ao sistema para se moldar a novas formas gestuais. No trabalho de Lementec e Bajcsy esta pouca autonomia acontece, pois só poderiam ser reconhecidos até 20 gestos que estavam codificados anteriormente para se proceder às comparações.

Por exemplo Katerina Nguyen (Nguyen, 2000) optou por desenvolver um sistema onde estão representadas as duas abordagens acima referidas, tanto um sistema baseado em regras, como um sistema baseado em *"machine learning"*. Esta última abordagem dá a hipótese de o sistema ser treinado para poder reconhecer novos gestos. Na Figura 2.2 pode-se observar o diagrama de funcionamento deste sistema.

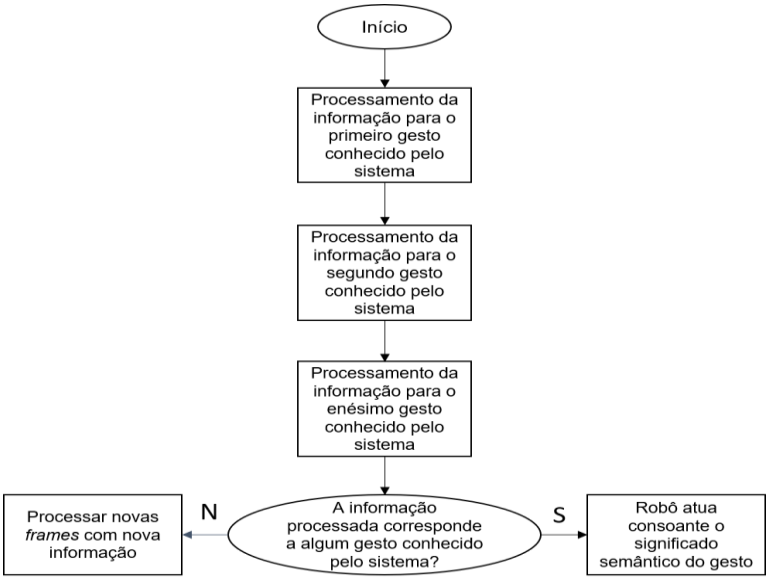

Figura 2.2 - Diagrama de funcionamento do sistema de reconhecimento de gestos baseado em regras e *machine learning*, de Nguyen [Adaptado de (Nguyen, 2000)].

Neste trabalho de Nguyen, é apresentado um sistema e um método de reconhecimento de gestos para realizar uma operação com base no significado semântico de gesto. Este sistema consiste, basicamente, no reconhecimento de um gesto executado por um sujeito através de uma câmara ligada a um computador, seguidamente este gesto é examinado pelo sistema, uma *frame* de cada vez. Estas *frames* são comparadas com dados que representam gestos já conhecidos pelo sistema, sendo estas comparações feitas em tempo real. Contudo, o sistema pode ser também treinado para melhorar o reconhecimento de gestos que já são conhecidos pelo sistema ou até novos gestos que ainda não estão registados no mesmo (Nguyen, 2000).

2.2 Sistemas de Reconhecimento através de Imagens 2D

Venetsky e Tieman (Venetsky & Tieman, 2008) demonstram um sistema de reconhecimento de gestos que permite o controlo de um robô através de comandos definidos por esses mesmos gestos executados por um utilizador.

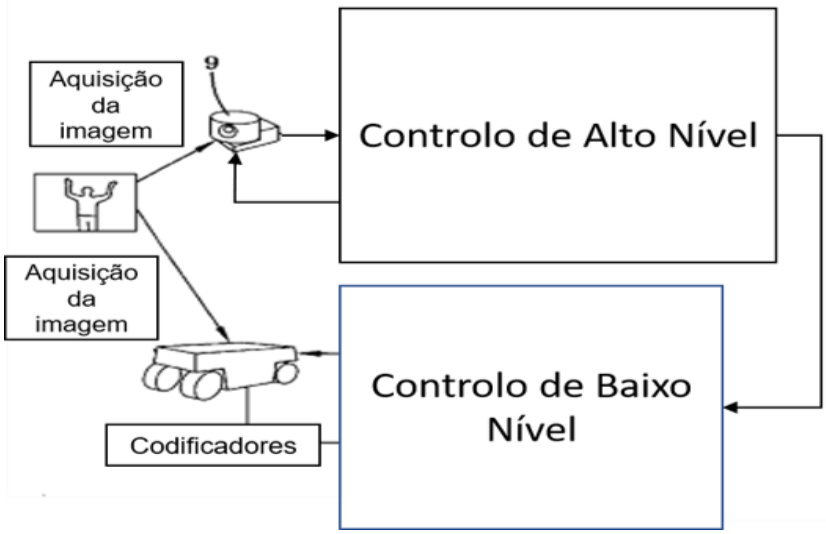

Figura 2.3 - Constituição do sistema de reconhecimento de gestos de Venetsky e Tieman [Adaptado de (Venetsky & Tieman, 2008)].

O sistema deste projeto, tal como se pode observar na Figura 2.3, é constituído por um robô, uma câmara, um computador, alto (*software* para processamento de imagem) e baixo (controladores) nível de controlo de reconhecimento de gestos. Este sistema permite localizar pontos pertencentes à mão esquerda, mão direita, tronco superior e tronco inferior da pessoa que está na imagem através das suas diferentes coordenadas, convertendo estes pontos para dados em forma de onda. Por sua vez, de seguida, relaciona estes dados com os dados de comandos do utilizador e posteriormente, dependendo da relação, é enviada a informação para o controlador de tensão que produz a tensão de corrente elétrica a aplicar a um ou mais motores elétricos ou atuadores pertencentes ao robô, para desta forma poder controlar o mesmo (Venetsky & Tieman, 2008).

Para detetar seres humanos e estimar as suas formas 2D a partir de uma única imagem, Poppe e Poel (Poppe & Poel, 2008) sugerem um método de deteção humana e recuperação da forma 2D a partir de imagens monoculares na presença de oclusões. Neste trabalho não existe uma modelação do ambiente capturado (*background*), pois os autores desenvolveram um algoritmo que funciona em ambientes desordenados e dinâmicos. Além disso, não se baseiam no movimento, o que adequa o trabalho para a estimativa a partir de uma única imagem (Poppe & Poel, 2008).

Dado um movimento de locomoção de um humano, este sistema pode prever com precisão a localização do pé esquerdo, observando apenas a perna esquerda. Para isto, o corpo humano é dividido em cinco partes (2 braços, 2 pernas e tronco), cada uma das quais tem bordas associadas e modelos de aparência. Dado um *match* entre o molde e a imagem do corpo, não se trata apenas as posições das junções dentro da parte do corpo mas sim de todas as posições comuns (Poppe & Poel, 2008). Esta abordagem permite recuperar a localização das articulações que estão ocluídas, como se pode observar na Figura 2.4.

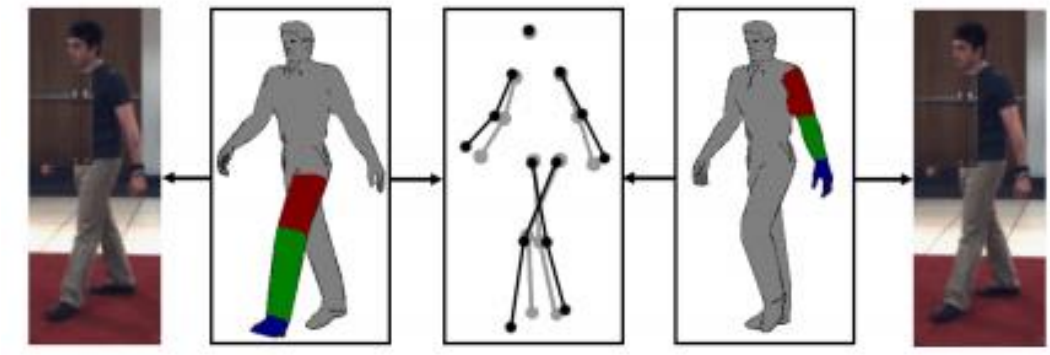

Figura 2.4 - Visão geral conceitual do método de deteção humana e recuperação da forma 2D, a partir de imagens monoculares na presença de oclusões, de Poppe e Poel [Adaptado de (Poppe & Poel, 2008)].

2.3 Sistemas de Reconhecimento através de Imagens 3D

No que diz respeito ao tratamento da imagem, para descobrir qual a posição real dos braços de uma pessoa em 3D a partir de imagens de profundidade, o trabalho de Hu et al. (Hu et al., 2010) representa uma boa solução para o problema de algumas posições que podem complicar a deteção dos braços da pessoa capturada, como por exemplo os braços junto ao corpo. A solução passa pelo uso de três caraterísticas convexas diferentes.

O principal foco neste projeto, está na deteção do braço em duas etapas, primeiramente e em paralelo deteta-se o antebraço e o braço, e seguidamente a união destes dois para se alcançar a totalidade do braço (Hu et al., 2010).

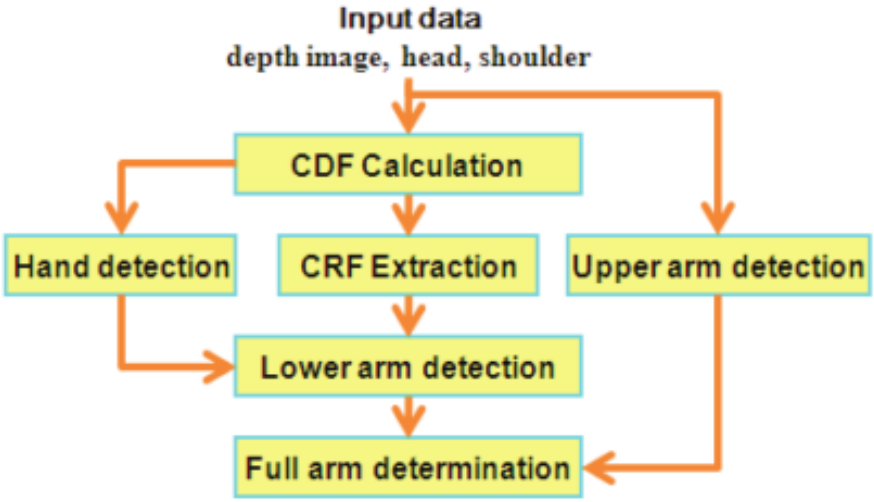

Figura 2.5 - Diagrama representativo do método que pretende descobrir qual a posição real dos braços de uma pessoa em 3D a partir de imagens de profundidade, utilizado em [Adaptado de (Hu et al., 2010)].

Os candidatos a braço são gerados a partir do ombro dado como *input*, utilizando UDF (*U-type depth feature*), que representa o contraste de profundidade entre o braço e a área na sua vizinhança.

O antebraço pode ser encontrado de duas maneiras diferentes, sendo o mais difícil de encontrar. Este pode ser extraído por CRF (*Convex Region Feature*) que é uma região ligada com alto grau de convexidade ou, se este falhar, o antebraço pode ser encontrado a partir da deteção das mãos da mesma maneira que o braço é detetado a partir do ombro.

Tanto os candidatos a mão, como a região ligada com alto grau de convexidade, são gerados a partir de CDF (*Convex Degree Feature)*, que descreve o gradiente de profundidade da área circundante centrada em cada *pixel*, como se pode ver na Figura 2.6, onde R representa a zona interior, mais próxima da câmara, e C a zona exterior, mais profunda.

Depois da utilização destas abordagens de caraterísticas convexas eficazes, é estimado o braço completo.

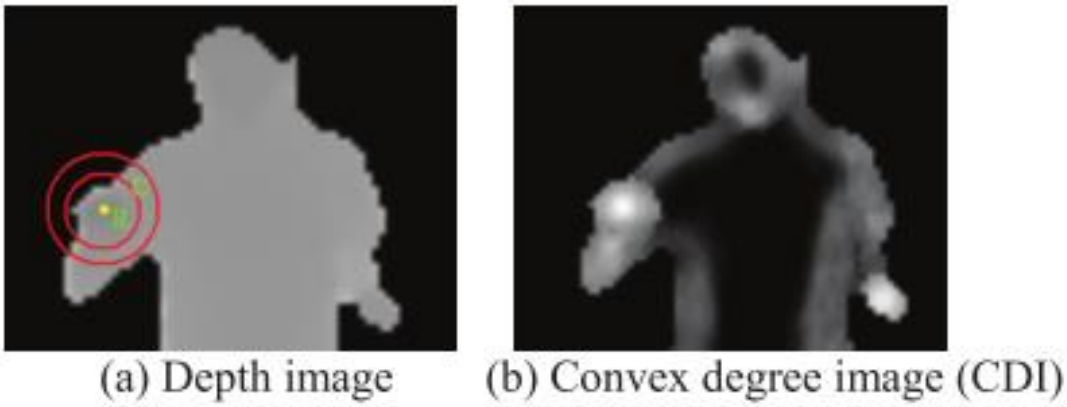

Figura 2.6 - Ilustração para cálculo de CDF (*Convex Degree Feature)* que descreve o gradiente de profundidade da área circundante centrada em cada *pixel* [Adaptado de (Hu et al., 2010)].

2.4 Aplicações com Sistemas Embutidos

Em relação aos sistemas embutidos, Senthilkumar et al. (Senthilkumar G et al., 2014) propõem uma técnica de captura de imagens através de um sistema embutido baseado em Raspberry Pi. Este sistema é constituído por uma placa Raspberry Pi, um monitor com porta DVI (*Digital Visual Interface)* para se poder observar as imagens capturadas, e uma câmara (*RPI Noir Camera Board*) sem filtro de infravermelho, o que permite capturar imagens com boa qualidade mesmo em ambientes com pouca luminosidade.

O sistema proposto por Senthilkumar et al encontra-se descrito no diagrama da Figura 2.7.

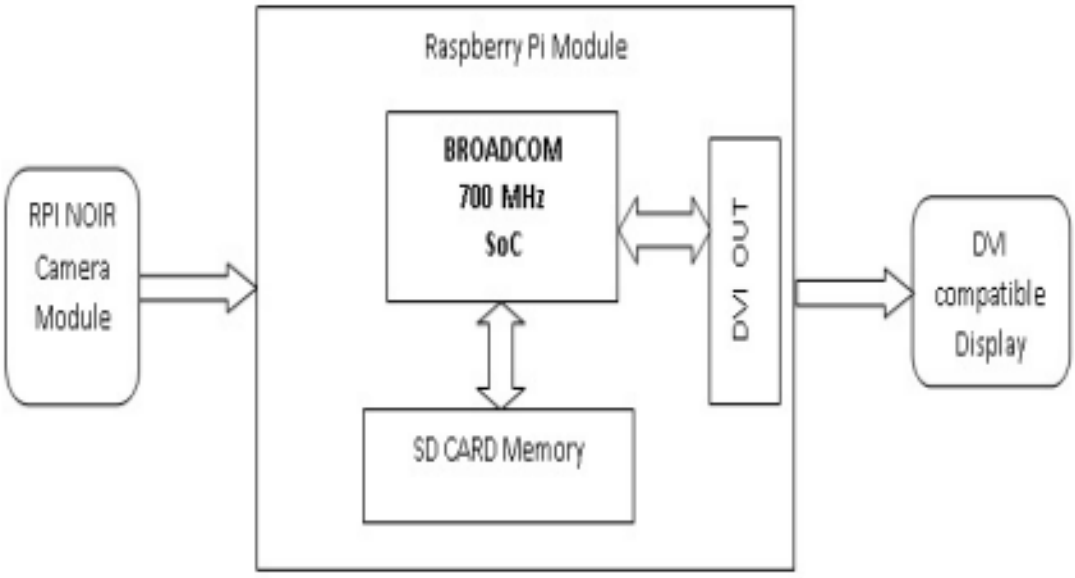

Figura 2.7 - Diagrama de blocos do sistema de captura de imagens através de um sistema embutido baseado em Raspberry Pi, proposto por (Senthilkumar G et al., 2014)

Este sistema foi concebido para operar em duas etapas diferentes. A primeira etapa serve para capturar a imagem e criar uma base dados. Na segunda etapa é capturada novamente uma imagem e esta é utilizada para comparar ou identificar imagens que estejam já presentes na base de dados, previamente criada.

O sistema aqui representado apresenta uma dimensão reduzida, mais leve e com menor consumo de energia, sendo por isso mais conveniente do que o sistema de reconhecimento de objetos baseado em *desktop*.

Devido ao fato de o código ser *open source* (código aberto), torna-se mais livre para fazer o desenvolvimento de *software* em Linux, o que facilita um melhor desempenho de todo o sistema (Senthilkumar G et al., 2014).

O fato deste projeto conter um sistema embutido, com o Raspberry Pi no centro desse sistema, faz com que o orçamento para o desenvolvimento de uma aplicação de reconhecimento de gestos, seja reduzido, sem perder a fiabilidade e com uma *performance* razoavelmente boa, em comparação com outros sistemas que utilizam computadores com outras capacidades.

A evolução deste tipo de sistemas tem vindo a ser cada vez maior, com a disponibilização de *software* sofisticado e com muitas bibliotecas de processamento de imagem como lili2, PIL, etc., especialmente desenvolvidas para o Raspberry Pi, que permitem desenvolver projetos deste tipo.

3 Descrição Global do Sistema Desenvolvido

Este capítulo contém toda a estrutura do projeto, tanto em termos de *hardware* como de *software*. É onde se apresenta a descrição de cada componente do sistema desenvolvido e onde se retratam algumas vantagens para a escolha dos mesmos.

3.1 *Hardware*

O sistema é composto por três grandes componentes de *hardware*. São estes o Raspberry Pi, a câmara V2 e a plataforma robótica onde vai ser embutido o Raspberry Pi e a câmara, a tartaruga Nando. Cada um destes componentes tem o seu papel no sistema que será retratado nesta secção.

Na Figura 3.1 está representado o sistema proposto, por blocos, em termos de *hardware*.

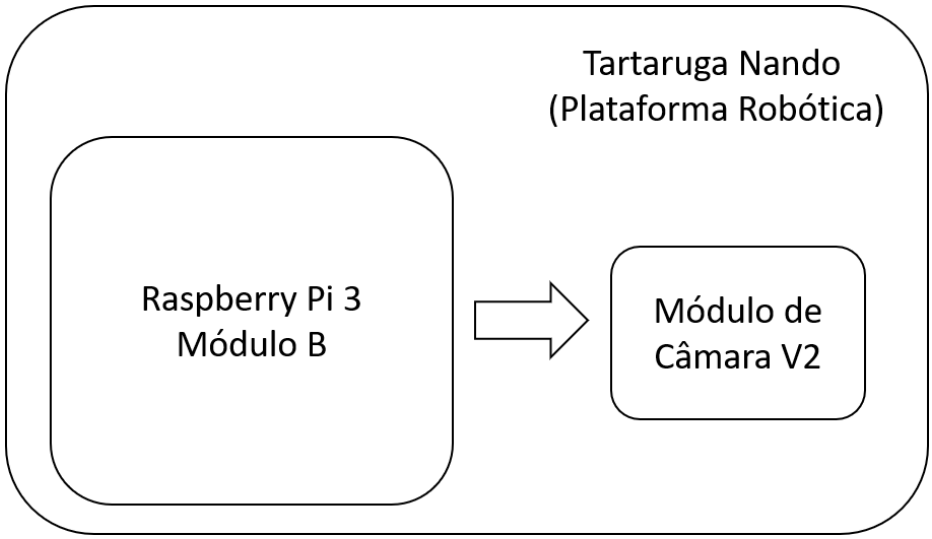

Figura 3.1 - Sistema de reconhecimento de gestos proposto neste documento, diagrama de blocos do *hardware*.

3.1.1 Raspberry Pi

O Raspberry Pi (Raspberry Pi Foundation, 2016b) é a plataforma escolhida para a implementação deste projeto, apresentando algumas características chave, que serão abordadas posteriormente nesta secção.

Este dispositivo é um sistema que integra todo o *hardware* de um computador numa só placa com o tamanho (largura e comprimento) de um cartão de crédito. A Fundação Raspberry Pi foi criada por Eben Upton, um professor da Universidade de Cambridge no Reino Unido, de modo a tornar este dispositivo num produto real e não apenas um projeto de estudo. O seu lançamento no mercado ocorreu em 2012 e gerou uma revolução no mundo da tecnologia. Apesar de o objetivo principal não ser a substituição de computadores, fixos ou portáteis, o produto teve resposta imediata por parte dos consumidores, nomeadamente como complemento aos mesmos. Uma das grandes vantagens, que também está na origem desta popularidade, é o seu baixo custo, sendo capaz de fazer o mesmo que computadores de elevado custo (Sachdeva & Katchii, 2014).

A Tabela 1 retrata as características dos vários tipos de Raspberry Pi (Raspberry Pi Foundation, 2015, 2016b):

Tabela 1 - Características de todos os Raspberry Pi existentes, com destaque para o Raspberry Pi 3 Modelo B, utilizado no âmbito deste projeto.

	RASPBERRY PI					
	1		**2**	**Zero**	**Zero W**	**3**
	A+	**Modelo B**	**Modelo B**			**Modelo B**
Lançamento	10/11/2014	15/02/2012	01/02/2015	30/11/2015	28/02/2017	**29/02/2016**
Preço (lançamento) €	18 €	30 €	30 €	5,50 €	11 €	**33 €**
Processador	BCM2835 700 MHz	BCM2835 700 MHz	BCM2836 900 MHz	BCM2835 1 GHz	BCM2835 1 GHz	**BCM2837 1.2 GHz**
GPU	Video-Core4	Video-Core4	Video-Core4	Video-Core4	Video-Core4	**Video-Core4**
RAM	256 MB	512 MB	1 GB	512 MB	512 MB	**1 GB**
Wireless	-	-	-	-	802.11n	**802.11n**
Bluetooth	-	-	-	-	4.1	**4.1**
Consumo	200 mA	700 mA	800 mA	160 mA	180 mA	**800 mA**

A escolha do Raspberry Pi 3 Modelo B, representado na Figura 3.2, é justificada devido à sua velocidade de processamento ser maior em relação a outros modelos Raspberry Pi, e também devido à sua reduzida dimensão que facilita todo o desenvolvimento do projeto em termos de acoplamento num trabalho previamente desenvolvido. Sendo este um projeto desenvolvido para complementar algumas características de um

robô previamente construído para outro tipo de projetos, a utilização desta plataforma de baixo custo é facilmente integrável em qualquer tipo de sistema robótico.

Figura 3.2 – Raspberry Pi 3 Modelo B utilizado, tendo sido escolhido por apresentar superior velocidade de processamento, em relação aos restantes modelos, exibindo reduzidas dimensões.

3.1.2 *Camera Module*

A aquisição de imagens, para posteriormente ser feita a interpretação dos gestos pretendidos, é feita através de uma câmara ligada ao Raspberry Pi, com o objetivo de captar os movimentos, trabalhando sempre com uma imagem de duas dimensões (2D). Através desta imagem, é possível obter características humanas, tais como o rosto de uma pessoa, juntamente com o reconhecimento do corpo na totalidade. Posteriormente, é feita a captação de movimentos dessa mesma pessoa, sendo que neste projeto não se consegue obter níveis de profundidade, pois a câmara utilizada não contém a característica necessária para obter tais dados.

A câmara utilizada neste projeto é a *Camera Module V2*, construída para ser utilizada com o Raspberry Pi. Este módulo veio substituir o módulo original (*Camera Module V1)*, em 2016, sendo que a diferença principal é a alteração do sensor OmniVision OV5647 de 5 megapixeis para um sensor Sony IMX219 de 8 megapixeis. Esta diferença veio aumentar a qualidade de imagem captada pela câmara.

Este módulo consegue gravar vídeos de alta definição e captar imagens com muita qualidade. A facilidade da sua utilização para iniciantes faz com que este tipo de câmaras sejam muito requisitadas para aprendizagem, mas estas também contêm capacidades para utilizadores avançados e projetos complexos (Raspberry Pi Foundation, 2016a). Na Figura 3.3 pode observar-se o módulo retratado neste capítulo.

Figura 3.3 - *Camera Module V2* igual à aplicada neste projeto, muito utilizado por iniciantes, no entanto, com capacidade para utilização no âmbito de projetos complexos, adaptado de (Raspberry Pi Foundation, 2016a).

Esta câmara pode ser utilizada nas versões 1, 2 e 3 do Raspberry Pi. A biblioteca *Picamera*, da versão de Python utilizada neste projeto, contém todas as funcionalidades possíveis desta câmara, como por exemplo, a captura de imagem, a alteração da resolução da mesma, entre outras.

Na Figura 3.4 pode-se observar a ligação deste módulo ao Raspberry Pi utilizado neste projeto:

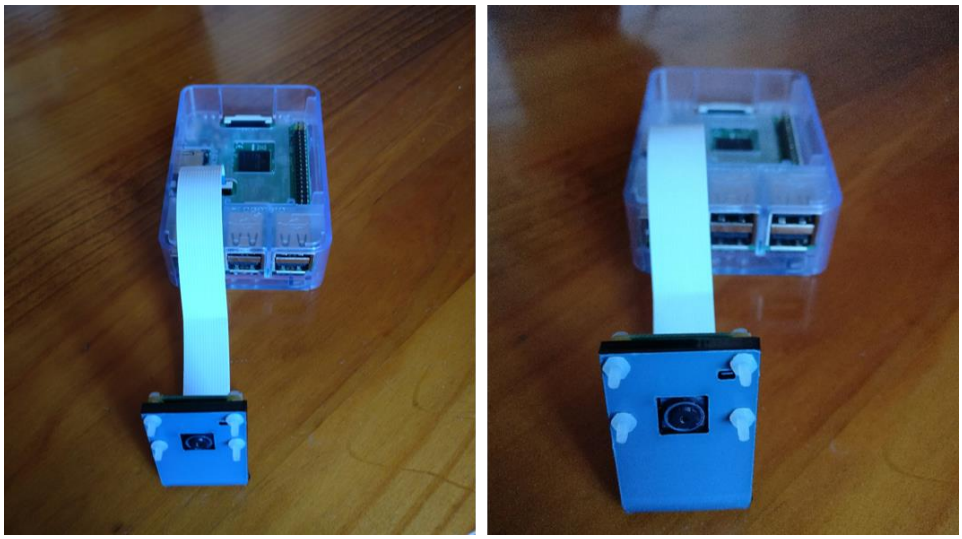

Figura 3.4 – Câmara V2 conectada com o Raspberry Pi 3, sistema utilizado neste trabalho.

Para concluir a constituição física deste projeto, é apresentada na Figura 3.5 a fotografia do sistema robótico, Tartaruga Nando, onde vai ser embutido o sistema desenvolvido, pretendido para utilização e aplicação neste projeto de reconhecimento de gestos.

Figura 3.5 – Visualização do *design* da Tartaruga Nando, elaborado no âmbito de um projeto anterior, igualmente em parceria com o Hospital Doutor Fernando Fonseca.

3.2 Software

Na plataforma Raspberry Pi podem ser instalados vários sistemas operativos, tais como: *Raspbian, Arch Linux, Fedora, Kano OS, Windows IoT Core, OSMC, OpenELEC, Pi MusicBox, RetroPie, Pipplware,* entre outros. No caso deste projeto utiliza-se o *Raspbian*. A linguagem de programação utilizada neste projeto foi o *Python*, que será falado mais à frente neste documento.

O processamento e tratamento de imagem é feito através da biblioteca do *OpenCV*, que é a mais utilizada quando se pretende trabalhar com processamento de imagem. Esta biblioteca é *open source*, pelo que é gratuita tanto para uso académico como uso comercial (Abaya et al., 2014), esta biblioteca terá nesta secção uma área onde é abordada.

Através de alguns dos métodos presentes na biblioteca acima referida, é possível obter resultados muito positivos no que diz respeito ao tratamento de imagem, sendo que, para este projeto, foram utilizados algoritmos de *background subtraction* e de *object detection*.

3.2.1 *Raspbian* e Python

O *Raspbian* é um sistema operativo (distribuição *Linux)*, gratuito baseado no *Debian* que foi desenvolvido e otimizado para Raspberry Pi, o que permite obter o desempenho máximo desta plataforma. Este sistema operativo vem com mais de 35 000 pacotes *.deb* que estão pré-compilados e prontos a serem facilmente instalados no sistema. Foi criado por um pequeno grupo de fãs de Raspberry Pi, mas que não tem ligação à Fundação Raspberry Pi. O *Raspbian* continua em desenvolvimento com o foco na melhoria de estabilidade e desempenho do máximo número de pacotes *Debian* possível (Dyllon et al., 2016),(Raspberrypi.org, 2016).

Podem-se observar na Figura 3.6 algumas ferramentas, tais como IDE's (*Integrated Development Environments*) de Python e outras funcionalidades, que podem ser utilizadas neste sistema operativo.

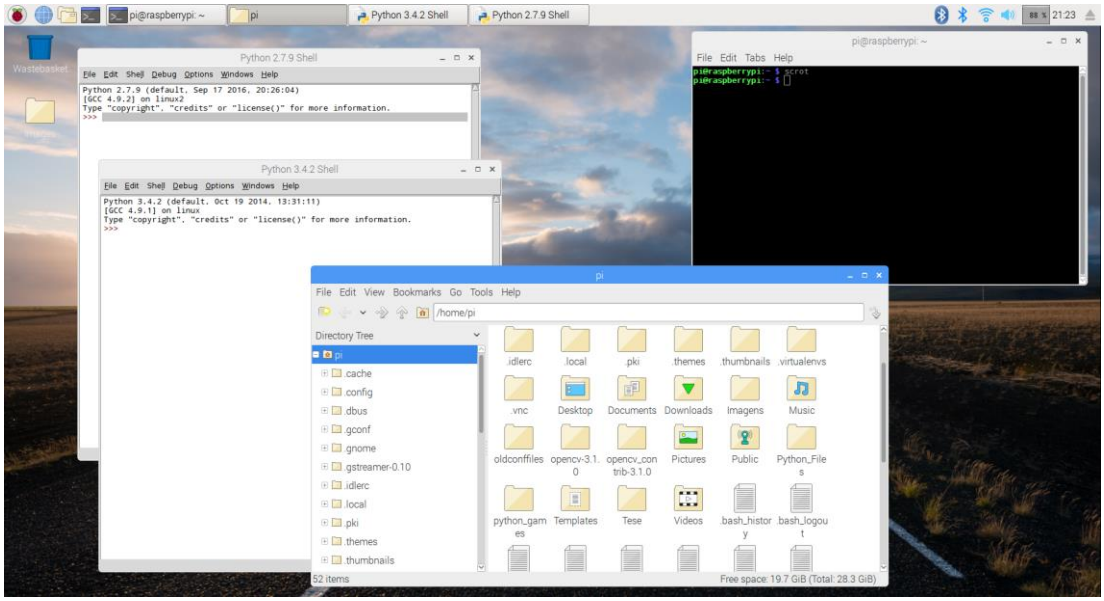

Figura 3.6 – Ambiente de Trabalho no *Raspbian*, assim como algumas ferramentas disponíveis com a utilização deste sistema operativo.

Em relação ao Python, esta é uma linguagem de programação quefacilmente se adapta ao sistema operativo *Raspbian*, uma vez que este contém as ferramentas essenciais para programar nesta linguagem.

Esta é uma linguagem de programação de fácil aprendizagem e por isso algo utilizada a nível de ensino, tendo também uma grande utilização em aplicações modernas, tanto a nível de aplicações *web* como de utilitários de *desktop* (Irani, 2016). Devido à sua sintaxe simples, facilidade de leitura e experiência global de fácil utilização, esta linguagem é recomendada para os aprendizes e iniciantes da programação.

Na Tabela 2 é possível observar algumas das vantagens que levam à preferência na utilização desta linguagem de programação.

Tabela 2 - Vantagens de utilização da linguagem de programação Python.

MULTI-PLATAFORMA	MULTI-PARADIGMA	INTEGRAÇÃO E COMUNICAÇÃO COM OUTRAS LINGUAGENS
Windows Mac Linux Android Maemo	Programação Estruturada Programação Orientada a Objetos Programação Funcional	Cython - C/C# Jython – Java IronPython – .NET Python for Dolphin – Dolphin Lunatic Python – Lua

3.2.2 Biblioteca OpenCV

O OpenCV (*Open Source Computer Vision*) (G. Bradski, 2000) é uma biblioteca de visão por computador que foi desenvolvida pela Intel no ano de 1999, mais propriamente por Gary Bradski, sendo que em 2006 saiu a versão 1.0 e em 2008 a Willow Garage (Laboratório de investigação de robótica) lançou a versão 1.1 desta biblioteca, tendo vindo a ajudar a aprimorar a mesma.

Esta biblioteca surgiu com o objetivo principal de acelerar o desenvolvimento de aplicações no que diz respeito ao processamento de imagens ou vídeos em tempo real. A versão original do OpenCV está escrita em C++ com possível aproveitamento de processadores *multicore*, porém as versões mais recentes contêm interfaces de programação para C, C++, Python, Java e Android. Uma das grandes vantagens da utilização desta biblioteca é o facto de ser multi-plataforma, na medida em que suporta Windows, Linux, e, recentemente, Android, MacOS e iOS (Wagner, 2012), (Gary Bradski & Kaehler, 2008; Emami & Suciu, 2012).

Na Figura 3.7 está representada a interface de OpenCV em Linux, mais propriamente em *Raspbian* (Sistema operativo implementado no Raspberry Pi).

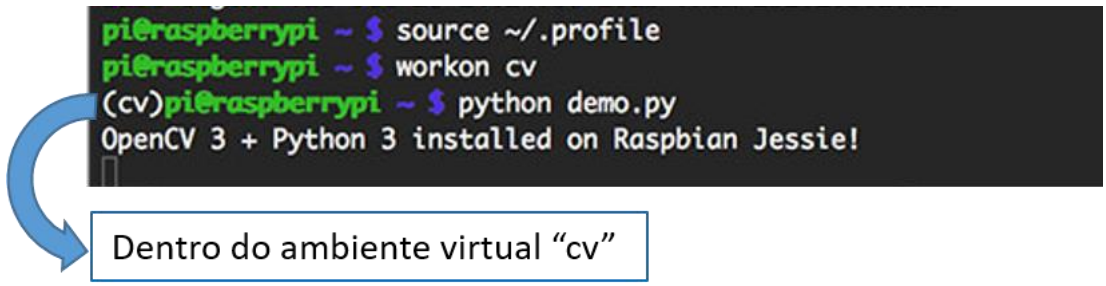

Figura 3.7 - Interface OpenCV em utilização no sistema operativo *Raspbian*

Ser *Open Source* significa que é uma biblioteca de código aberto, ou seja, contém implementações de algoritmos de visão por computador de patente livre, simplificando muito a programação para os seus utilizadores. Inclui funções avançadas para deteção e reconhecimento de rosto, reconhecimento de gestos, filtragem de Kalman e alguns métodos de inteligência artificial, que se encontram prontos a utilizar. Se os mesmos utilizadores desejarem uma otimização adicional, podem comprar as bibliotecas IPP (*Integrated Performance Primitives*) no *site* da Intel, que consistem em rotinas otimizadas de baixo nível em alguns algoritmos para diferentes e variadas aplicações. Sendo um dos objetivos do OpenCV fornecer uma infraestrutura de *computer vision* simples de usar, esta concede aos seus utilizadores uma grande ajuda para o desenvolvimento de aplicações de visão sofisticadas rapidamente.

Juntamente com empresas bem estabelecidas como Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota que utilizam a biblioteca OpenCV, existem muitas *startups*, como Applied Minds, VideoSurf e Zeitera, que fazem uso extensivo do OpenCV (G. Bradski, 2000).Contendo mais de 500 funções, o OpenCV abrange várias áreas tais como, imagens médicas, segurança, calibração de câmara, visão de estéreo e robótica (Gary Bradski & Kaehler, 2008).

3.3 Métodos de processamento de imagem

Nesta secção encontram-se descritos os métodos/algoritmos de processamento de imagem que deram origem aos resultados expostos na próxima secção. A utilização dos seguintes métodos, em conjunto, vem dar resposta ao pretendido para este projeto, com o principal foco no reconhecimento e interpretação dos gestos executados pelo utilizador. Sendo o OpenCV a biblioteca utilizada neste trabalho, todos os seguintes métodos estão relacionados com funções disponibilizadas pela mesma.

3.3.1 HOG (*Histogram of Oriented Gradients*)

É um método de descrição de características, muito utilizado nas áreas de *computer vision* e de processamento de imagem. Esta técnica de descrição trabalha com as diferentes orientações de gradientes localizados numa imagem ou numa região de interesse (ROI – *Region of Interest*). Para a deteção de humanos existem algumas estratégias dentro das várias formas de utilização do método, que variam a sua *performance*. Neste caso, tanto a amostragem espacial bastante grosseira como a amostragem de orientação fina e a forte normalização fotométrica local, são a melhor estratégia para descrever humanos, na medida em que permite que membros e segmentos corporais mudem de aspeto e se movam de várias maneiras, desde que mantenham aproximadamente uma orientação vertical (Dalal & Triggs, 2005).

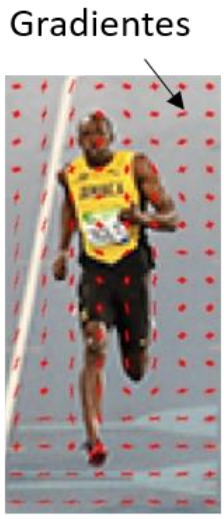

Figura 3.8 - Observação da variação da orientação dos gradientes numa determinada imagem, o que vai permitir a descrição de características da mesma, para posterior processamento e identificação de um humano (Mallick, 2016).

Na Figura 3.8 pode-se observar a variação na orientação dos gradientes que conseguem ser indicadores de objetos em determinadas posições. O descritor HOG baseia-se neste tipo de gradientes para encontrar um formato que se aproxime das orientações de gradientes num humano. Ou seja, através da variação de orientação dos gradientes (vetores) que estão distribuídos pela imagem, esta função consegue determinar a forma do objeto em função das diferentes orientações desses mesmos gradientes.

A quantidade de falsos positivos deste método é reduzida, sendo dependente dos parâmetros utilizados na função que está presente na aplicação desenvolvida.

3.3.2 Classificador *Haar Cascade*

Este classificador serve para fazer uma deteção rápida e com alguma precisão de certas partes do corpo humano. É utilizado neste trabalho para fazer a deteção facial frontal do utilizador da aplicação, através da função *cv2.CascadeClassifier*, também ela fornecida pelo OpenCV.

Dada uma imagem, neste caso uma *frame* de um vídeo em direto ou gravado, este detetor facial examina cada local da imagem e classifica-o como sendo um rosto ou não.

O classificador usa dados armazenados num ficheiro XML, que contém um conjunto de árvores de decisão que usam características *haar-like* (simetrias) para fazer a classificação. O algoritmo utilizado é o Viola-Jones (P. Viola & Jones, 2001; Paul Viola & Jones, 2001). Este algoritmo executa várias comparações de simetria através de imagens de rostos que são utilizadas para treinar e melhorar a sua eficácia.

O ficheiro XML utilizado neste trabalho, *haarcascade_frontalface_default.xml*, é obtido através do *download* do pacote OpenCV, que já contém alguns ficheiros XML no mesmo.

O processo de deteção do rosto é apresentado no diagrama de bloco da Figura 3.9.

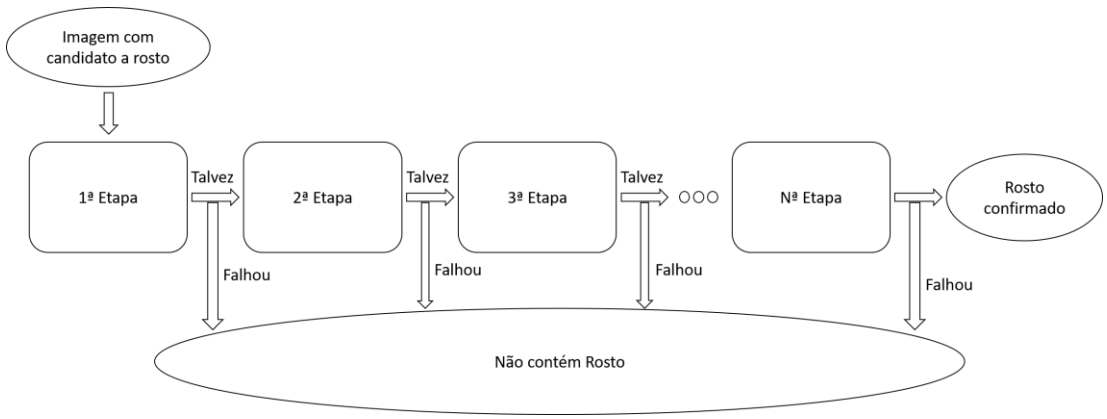

Figura 3.9 - Diagrama de blocos da classificação *Haar Cascade*, representada com as respetivas etapas, para deteção do rosto humano. [Adaptado de (Krishna & Srinivasulu, 2012)]

O diagrama apresenta, de uma forma simples, como se comporta o algoritmo de classificação (Krishna & Srinivasulu, 2012). Em cada etapa é comparada a *frame* com as imagens de rostos predefinidas no ficheiro XML, divididas por níveis, onde a última etapa é o nível mais conclusivo.

Esta classificação é feita rapidamente, apesar do processo parecer um pouco pesado em termos computacionais, pois a resolução utilizada é baixa, o que reduz o tempo de processamento (Emami & Suciu, 2012).

Na Figura 3.10 está representada uma deteção facial aplicando o tipo de classificador *Haar Cascade* a partir do ficheiro XML acima referenciado.

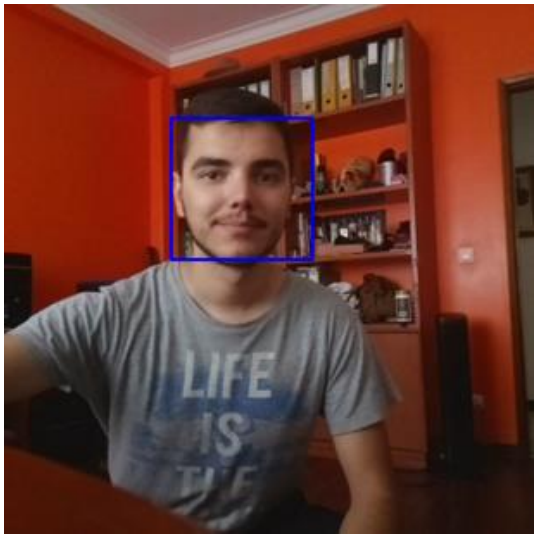

Figura 3.10 - Exemplo de deteção facial através de *Haar Cascade*. Depois de analisada a imagem num todo, este detetor facial examina cada local da mesma e classifica-o como sendo um rosto (retângulo azul) ou não.

3.3.3 MOG (*Mixture Of Gaussians*)

Este é um método de *background subtraction* utilizado essencialmente para detetar objetos em movimento num vídeo (Kochláň et al., 2014). MOG é uma técnica recursiva que não guarda um *buffer* de imagens para estimar o fundo, como outras técnicas não recursivas, mas atualiza um único plano de fundo baseado numa *frame* (Cheung, 2017). Um dos problemas de ser recursivo é o fato de poder manter um erro durante muito tempo, arrastado de uma *frame* de algum tempo atrás (Mohamed et al., 2010). Este erro pode causar uma imperfeição no fundo identificado. Uma vez que atualiza o fundo em cada *frame*, esta técnica utiliza pouca memória o que aumenta o desempenho do processo, pois não vai guardando todas as *frames* em memória.

No diagrama de blocos da Figura 3.11, é possível observar-se o processo de identificação de movimento numa imagem, através de *background subtraction*.

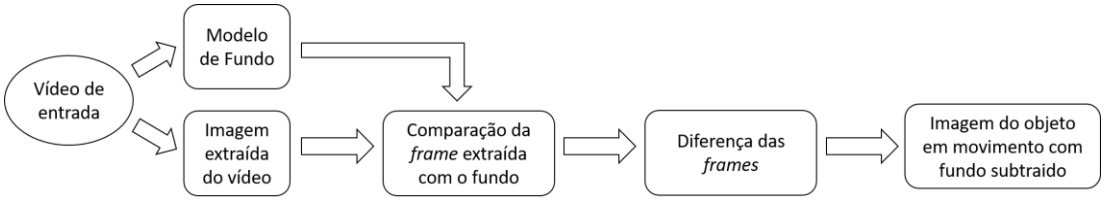

Figura 3.11 - Diagrama de blocos do processo de *background subtraction*, a partir do vídeo dado à entrada, obtém-se a imagem do objeto ou pessoa em movimento, já com a subtração do fundo. [Adaptado de (Tyapi & Sowmya, 2015)]

Os grandes problemas associados a esta técnica são as definições de alguns parâmetros, pois cada caso tem de ser tratado de maneira diferente. O tempo de atualização é, por exemplo, um dos grandes parâmetros deste método, pois pode ser determinante na deliberação da relevância que se dá às diferenças em cada *pixel* nas diferentes *frames* de um vídeo (Zivkovic, 2004). No caso deste trabalho, como o que se pretende é o reconhecimento do movimento executado por uma pessoa, a sensibilidade deste método é um dos fatores determinantes no seu sucesso.

Para a utilização do MOG e de qualquer técnica de *background subtraction*, é necessário que a câmara que capta o vídeo se encontre estática para que não existam alterações no fundo que condicionem a utilização desta função.

Pode observar-se na Figura 3.12 um exemplo de deteção do movimento de um humano recorrendo à aplicação do algoritmo MOG.

Figura 3.12 - Exemplo de deteção do movimento de um humano através do algoritmo MOG, sendo de realçar que o fundo não foi alterado e a câmara encontrava-se estática.

3.4 Funcionamento Global do Sistema

Numa visão geral, todo o projeto está pensado de uma forma simplificada, para que seja alcançado o objetivo de interação com o robô através do resultado final de um reconhecimento adequado de movimentos/gestos. Para esta interação resultar em boas condições, a câmara, ou seja, a plataforma robótica, deverá permanecer estática para que não existam alterações bruscas de fundo, pois esse é um problema difícil de lidar.

A forma como se desenrola todo o processo de reconhecimento está dividida em três etapas: a deteção de uma possível pessoa, a identificação e confirmação dessa possível pessoa através da deteção do seu rosto e por fim a pesquisa e reconhecimento de movimentos executados pela mesma.

Cada gesto efetuado com o braço, identificado e considerado por esta aplicação, está associado a um som diferente feito pelo computador/robô, o que serve de interação do robô com o utilizador. Na Tabela 3 é possível verificar quais os sons emitidos pelo robô aquando da execução de cada um dos gestos.

Tabela 3 - Identificação sumariada de todos os movimentos passíveis de deteção pela aplicação, aos quais está associada a emissão de um som, respetivamente.

MOVIMENTO	FALA / SOM
Esquerda ascendente	"Vamos brincar!"
Esquerda descendente	"Eu sou uma tartaruga."
Direita ascendente	"Gosto muito da minha carapaça!"
Direita descendente	"Sei contar até 3: 1,2,3!"
Esquerda em cima	"Olá outra vez!"
Direita em cima*	"Então adeus, até breve!"

*Movimento que termina o programa

Encontra-se representado na Figura 3.13 o fluxograma que melhor descreve o funcionamento global desta aplicação, com foco nas três etapas acima referidas e nas respetivas decisões em cada uma das etapas.

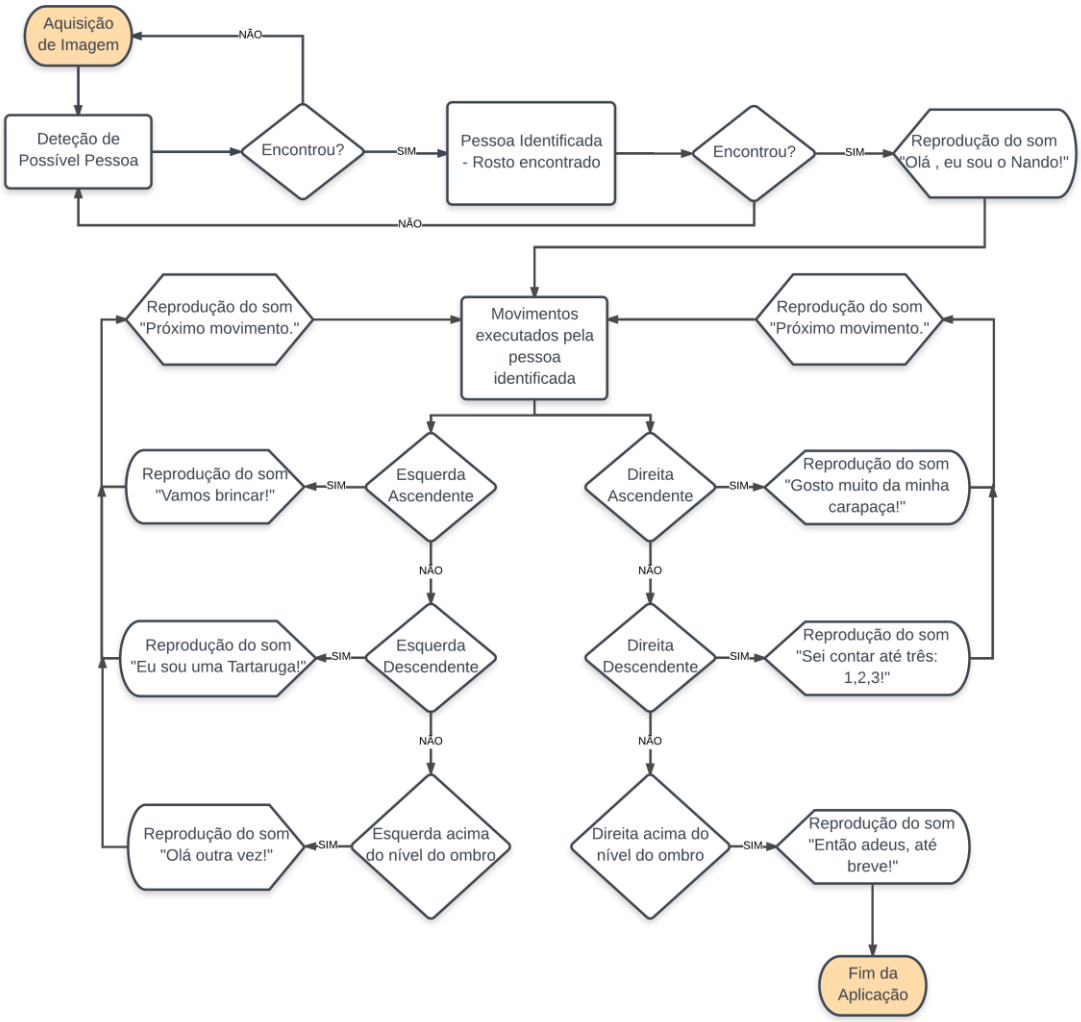

Figura 3.13 - Fluxograma representativo do projeto retratado neste documento, iniciando-se pela “Aquisição da Imagem” e finalizando com a Reprodução do som “Então adeus, até breve!”.

4 Testes e Resultados

Neste capítulo estão presentes alguns testes a todos os métodos abordados, bem como a descriminação do que é feito em cada procedimento. São também apresentados problemas encontrados ao longo do desenvolvimento da aplicação, a escolha da resolução de imagem e a justificação dessa mesma escolha.

Com este capítulo pretende-se esclarecer os resultados que foram obtidos ao longo de todo o processo de desenvolvimento desta aplicação de reconhecimento de gestos.

4.1 Teste e Descrição de Todo o Procedimento

No início deste processo de reconhecimento de gestos, o primeiro desafio a ser proposto é o de identificação de uma pessoa em frente da câmara. Para isso, é através de Histogramas de gradientes orientados (HOG) que se procura, nas imagens capturadas pela câmara, por um objeto com o formato de um humano em corpo inteiro.

Como se pode observar na Figura 4.1, quando é encontrado esse objeto um retângulo a verde é desenhado um retângulo a verde, delineando a pessoa possivelmente identificada.

Figura 4.1 - Diferentes deteções de possíveis pessoas, encontrando-se identificadas com recurso a um retângulo verde. Quando o rosto não é identificado, o programa não identifica como sendo uma pessoa.

Depois deste processo de deteção através da função HOG, que pode obter alguns falsos positivos, para se proceder à identificação e confirmação de que é realmente uma pessoa, procede-se de seguida à deteção facial através de *Haar Cascade.*

Com a junção da função baseada em *Haar Cascade* e a função HOG, consegue-se obter uma confirmação, demonstrado no retângulo azul que vai delinear a face da pessoa identificada, como se pode observar na Figura 4.2.

Figura 4.2 - Observação de uma pessoa identificada, encontrando-se o corpo identificado com recurso a um retângulo verde, através da função HOG, e o rosto identificado com recurso a um retângulo azul, pela utilização da função *Haar Cascade*.

Quando a pessoa é identificada, o robô reproduz a seguinte frase: "Olá, eu sou o Nando!", para que haja uma interação inicial com o utilizador, dando assim sinal de que o mesmo se encontra visível na câmara.

Posteriormente à determinação da pessoa o processo de identificação de movimentos ou gestos é o próximo desafio, sendo também o último.

Assim que é feita a identificação do corpo e da face, o programa fica à espera de um comando para que se comece o reconhecimento de gestos ou então basta esperar cerca 15 segundos. Isto para que a pessoa só depois de estabilizada em frente da câmara, ou seja, passado poucos segundos parada, possa dar início ao processo de reconhecimento de gestos sem que esteja a executar gestos que possam ser feitos por qualquer outra parte do seu corpo que ainda estivesse em movimento.

Depois de executado o comando, neste caso premindo a tecla "i" do teclado ou esperando 15 segundos (Figura 4.3) a função MOG fica ativa para qualquer tipo de movimento executado dentro do retângulo verde, que é o delimitador da pessoa em questão. No caso de aparecer outra pessoa na imagem, o programa reconhece um erro de identificação e deixa de reconhecer movimentos até que fique só uma pessoa na imagem.

Figura 4.3 - Pessoa identificada com reconhecimento de gestos ativo. De salientar que apenas os movimentos realizados na zona interna do retângulo verde são contabilizados.

Para detetar movimentos na imagem utiliza-se uma função de *background subtraction* que neste caso é a função MOG, que vai atualizando o fundo da imagem e identificando assim o que está em movimento e o que está parado. Neste caso, o que está parado não interessa e é considerado fundo.

A identificação do movimento por si só não consegue dar resposta ao tipo de movimento que é feito para ser considerado um gesto ou indicação de algo. Para dar algum significado ao movimento executado entra-se em consideração com a posição desse mesmo movimento em relação a uma parte do corpo da pessoa previamente identificada, que neste caso é o seu rosto.

Na Figura 4.4 é possível verificar que, com a comparação da posição do retângulo vermelho (corpo em movimento) com o retângulo azul (rosto), e comparando também com a posição do braço no momento anterior, já é possível dar uma resposta ao tipo de movimento executado, tanto em termos do lado como da direção para os quais é feito o movimento. Podemos também observar nessa mesma figura a extração da parte do corpo em movimento feita pela função MOG (*background subtraction*).

Figura 4.4 – Identificação do "Movimento Direita Ascendente", bem como da imagem do algoritmo MOG, em que o braço em movimento (retângulo vermelho) é comparado com o rosto (retângulo azul), de modo a identificar o lado e direção para os quais o movimento é feito.

O retângulo vermelho é obtido através da aglomeração dos pixeis a branco na imagem do algoritmo MOG. Está definido previamente um tamanho mínimo para se considerar um braço na imagem, para que os pequenos pontos, que se conseguem observar na imagem da direita na Figura 4.4, não sejam identificados como movimento. Este parâmetro de tamanho mínimo é definido tendo em conta a resolução em que se está a trabalhar. No caso deste trabalho onde a resolução é 320x240 pixeis, ou seja uma área total de 76800 pixeis, está definido um número de área total igual ou superior a 300 para se considerar o movimento. Assim, o ruído que se observa, não entra em conta na formação do retângulo vermelho.

Existem alguns gestos reconhecidos sempre tendo em conta o referencial da posição do rosto, sendo assim pode-se obter uma variedade de movimentos que podem ser interessantes para algumas aplicações.

Na Figura 4.5 são observados todos os movimentos reconhecidos neste projeto. Os movimentos à direita e à esquerda, ascendentes ou descendentes, que são identificados consoante a posição do membro em movimento esteja à direita ou à esquerda do rosto, e consoante esse movimento seja feito para cima ou para baixo. Existem também os movimentos em cima, que são identificados se o gesto for executado acima da parte inferior do rosto identificado, ou seja acima do ombro, e o lado correspondente.

Figura 4.5 - Observação de todos os gestos/movimentos reconhecidos neste projeto

4.2 Identificação e Resolução de Problemas

Nesta secção são apresentados alguns problemas encontrados ao longo do desenvolvimento do projeto.

O problema mais importante é a questão da velocidade de processamento ser baixa, ou seja, processar poucas imagens por segundo (FPS - *Frames Per Second*). Isto acontece devido ao processador do Raspberry Pi 3 não ter capacidade de realizar as várias etapas de processamento de uma forma rápida numa imagem com grande resolução. Os processos de *background subtraction* sobcarregam o processador, pois existe uma grande quantidade de pixeis para o algoritmo percorrer.

Em relação ao problema da velocidade de processamento, os cálculos efetuados para verificar as imagens por segundo que o processador consegue tratar em tempo real, é feito variando a resolução até se obter um número de FPS suficiente para ter uma resposta em tempo real.

O primeiro teste apresentado é com a resolução de 800 x 600 pixeis. O valor de FPS obtido pode verificar-se na Figura 4.6. Neste tipo de imagens, com esta resolução, o número total de pixeis é de 480 000, o que é bastante elevado para obter velocidades de processamento razoáveis.

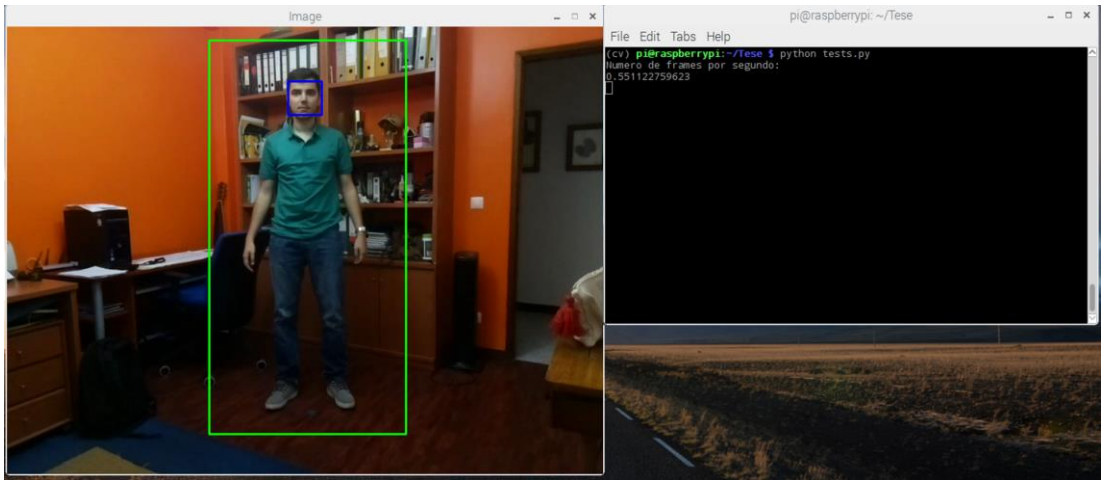

Figura 4.6 - Teste de deteção e obtenção de número de FPS (*Frames Per Second*) na captura de vídeo para reconhecimento de gestos, utilizando a resolução de 800x600 pixeis.

O segundo teste contém a resolução de 640 x 480 pixeis. O valor de FPS obtido corresponde ao valor apresentado na Figura 4.7. Esta resolução contém imagens com 307 200 pixeis o que ainda é um valor elevado para se obter uma velocidade que viabilize uma interação sem atrasos de processamento que afetem o desempenho do reconhecimento do movimento efetuado pelo utilizador.

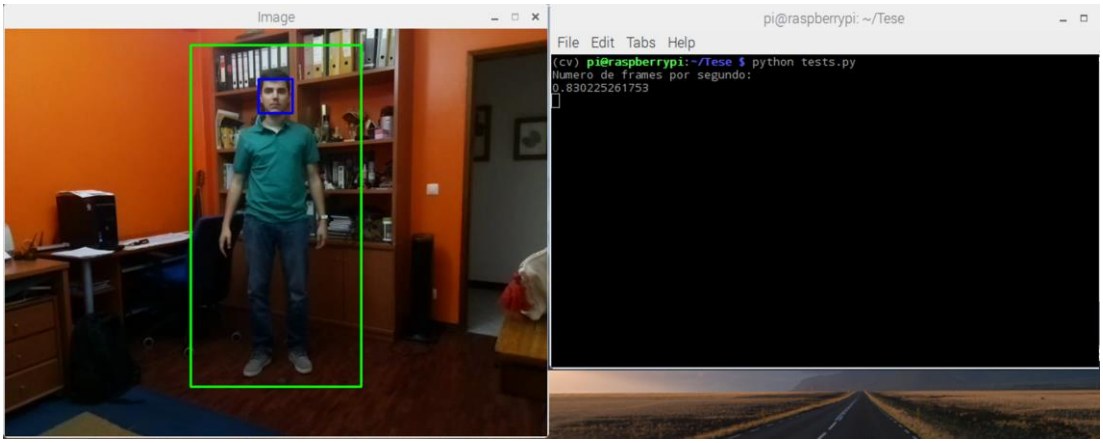

Figura 4.7 - Teste de deteção e obtenção de número de FPS (*Frames Per Second*) na captura de vídeo para reconhecimento de gestos, utilizando a resolução de 640x480 pixeis.

O terceiro teste executado, com a resolução 320 x 240 pixeis, já apresenta um bom valor de FPS para este tipo de processamento, como se pode observar na Figura 4.8. O número de pixeis nesta resolução é de 76 800, valor que permite uma maior velocidade de processamento em relação aos anteriores.

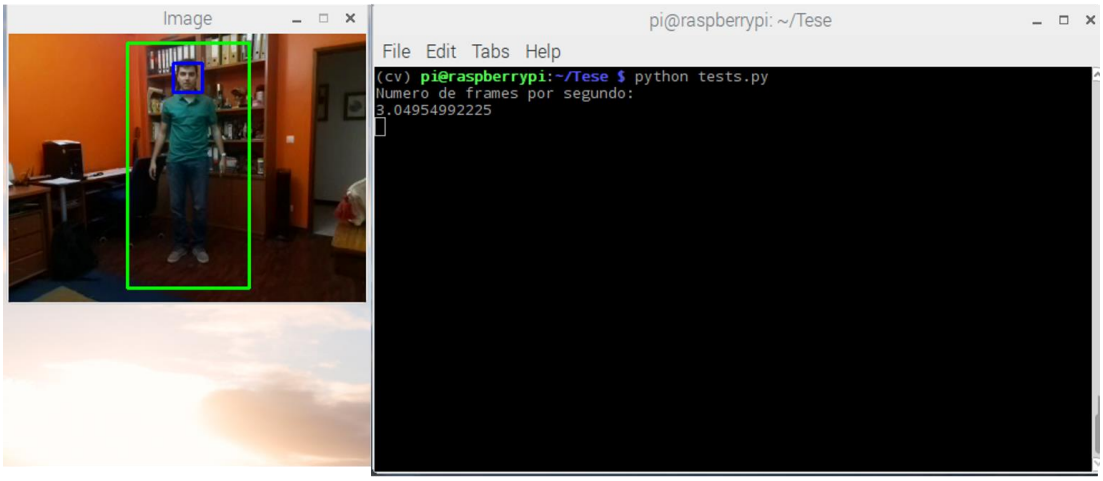

Figura 4.8 – Teste de deteção e obtenção de número de FPS (*Frames Per Second*) na captura de vídeo para reconhecimento de gestos, utilizando a resolução de 320x240 pixeis.

Quanto mais baixa for a resolução utilizada, piores serão as deteções dos utilizadores e a respetiva deteção do rosto, consequentemente menores serão os movimentos detetados. Isto acontece devido ao fato desta aplicação não conseguir através dos algoritmos utilizados, que contém um mínimo de dimensão exigida para validar algumas operações, identificar a pessoa que está em frente da câmara a executar os movimentos. A uma baixa resolução vem associada uma menor definição, o que faz com que deixem de ser percetíveis algumas características identificadas, como por exemplo o rosto.

O quarto teste, com resolução 160x120, pode observar-se na Figura 4.9. Os resultados da utilização desta resolução não são fidedignos, e por isso a sua utilização pode comprometer os resultados da aplicação, apesar de apresentar uma maior velocidade de resposta. Isto acontece, pois a imagem não contém tamanho suficiente para se identificar a pessoa e o rosto da mesma.

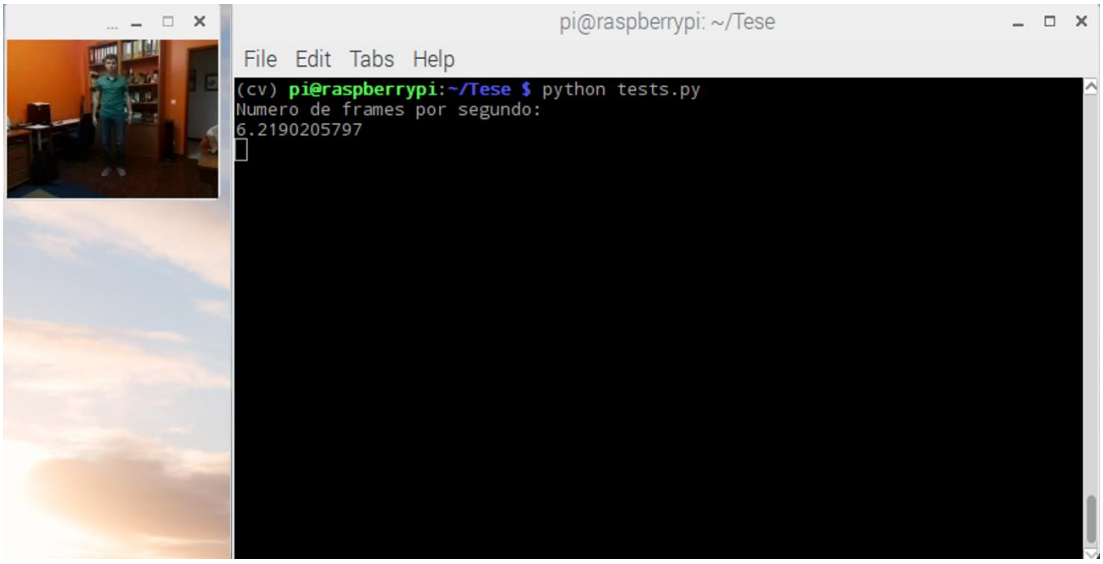

Figura 4.9 - Teste de deteção e obtenção de número de FPS (*Frames Per Second*) na captura de vídeo para reconhecimento de gestos, utilizando a resolução de 160x120.

No teste da Figura 4.9, pode verificar-se que, nas mesmas condições que os testes anteriores, não se consegue obter a deteção do utilizador que é pretendida, eliminando desde logo a utilização desta última resolução.

Na Tabela 4 pode observar-se uma descrição sumarizada da utilização das diferentes resoluções e também uma avaliação qualitativa dos diferentes comportamentos. A verde encontra-se a resolução escolhida neste projeto. Esta escolha deve-se a uma ponderação de todas as variáveis que se apresenta na seguinte tabela.

Tabela 4 - Descrição de FPS associadas ao uso das diferentes resoluções e a respetiva avaliação qualitativa, tanto em termos de velocidade de processamento e de qualidade de deteção.

RESOLUÇÕES	FPS (*FRAMES PER SECOND*)	VELOCIDADE DE PROCESSAMENTO	QUALIDADE DE DETEÇÃO
800x600	**0.55**	**BAIXA**	**ELEVADA**
640x480	**0.83**	**BAIXA**	**ELEVADA**
320x240	**3.05**	**ELEVADA**	**MÉDIA**
160x120	**6.22**	**ELEVADA**	**BAIXA**

Em relação às avaliações, a definição de velocidade de processamento **Elevada** significa uma resposta praticamente imediata ao movimento e **Baixa** quer dizer que a resposta ao movimento tem uma demora significativa e que dificulta o processo de interação com o robô.

A definição de qualidade de deteção **Elevada** significa que a deteção do utilizador e a sua confirmação através da deteção do seu rosto, é garantida onde quer que o utilizador se coloque em frente da câmara, a **Média** quer dizer que existem algumas áreas mais afastadas da câmara onde já não se garante deteção e **Baixa** significa que a deteção não é garantida em nenhuma área de captação da câmara.

A limitação da área de captação da câmara é um dos fatores que se pode tornar também um problema se não forem cumpridas certas regras de distâncias.

Uma vez que a altura do utilizador é variável, o campo onde é captada a imagem total, necessária para identificação da pessoa, é dependente dessa mesma altura. Esta variável torna-se importante pois sem a devida captação da pessoa que executa os movimentos, estes não são reconhecidos.

Na Figura 4.10 estão presentes as distâncias que delimitam a área de captação da câmara.

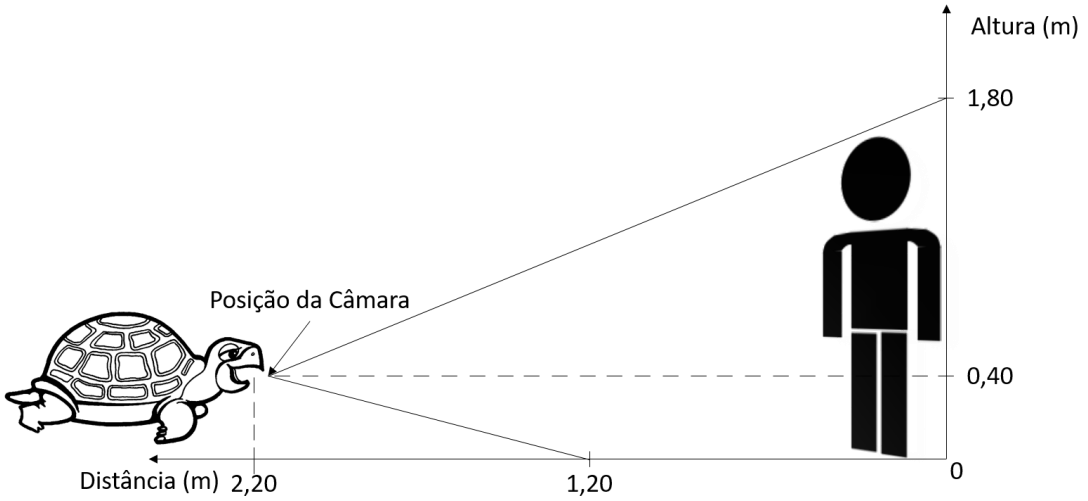

Figura 4.10 – Gráfico de distâncias requeridas para uma captação que permita o reconhecimento de movimentos, da câmara até à pessoa, para que esta possa ser captada na totalidade.

Não é recomendado o afastamento da câmara a uma distância muito superior à demonstrada na Figura 4.10, pois a partir de uma certa distância a pessoa deixa de ser percetível na imagem capturada. São apresentados na Figura 4.11 alguns resultados de testes feitos para outras distâncias onde não é percetível o utilizador em frente da câmara.

Figura 4.11 - Posições onde o sistema não consegue identificar o utilizador devido à distância estar fora do recomendado, fazendo com que a deteção do corpo ou por vezes do rosto, não aconteça.

As distâncias testadas indicam que demasiado longe (4 metros da câmara) ou demasiado perto (1,20 metros da câmara), não é possível reconhecer o utilizador que pretende executar os gestos. Estas distâncias foram testadas para um utilizador com altura de 1,80 metros, demonstrando assim que para os utilizadores mais baixos, como as crianças a que se destina, esta aplicação tem maior área de alcance, aumentando assim os casos de sucesso.

5 Conclusões

O objetivo do projeto é o desenvolvimento de uma aplicação, de baixo custo, de reconhecimento de gestos para interação com um robô móvel, através da implementação numa plataforma de computação, Raspberry Pi, com a vertente social de interação em ambiente pediátrico.

Ao longo da execução do projeto, foram muitas as estratégias pensadas para adotar, como por exemplo o uso de câmaras *Kinect* que devido a muitos fatores como por exemplo o tamanho, que não permite a integração da plataforma robótica Nando. Muitas das estratégias foram repensadas para que os custos associados ao projeto não aumentassem, mantendo sempre uma perspetiva conservadora, mas que desse origem a resultados positivos e fidedignos.

Associado a esta plataforma de computação utilizada (Raspberry Pi 3), vem o baixo consumo de energia, o que hoje em dia representa um ponto muito importante neste tipo de projetos. Também a sua reduzida dimensão e custo reduzido são pontos positivos que eram requisitos essenciais para este projeto. Sendo que apesar do seu reduzido tamanho, contém características de processador razoáveis para este tipo de dispositivos. Porém, alguns problemas foram aparecendo ao longo do projeto, tais como a velocidade de processamento quando se trabalha com resoluções altas, sendo que o processador deste tipo de equipamentos apresenta algumas dificuldades para dar respostas em tempo real.

Apesar de alguns problemas a nível de capacidade de processamento, o custo total do projeto é reduzido e a solução apresentada cumpre com o pretendido inicialmente, apresentando também resultados muito positivos.

Este sistema será, posteriormente, acoplada ao robô móvel construído noutro projeto, adicionando-lhe uma outra vertente de animação. Este projeto está ligado ao Hospital Professor Doutor Fernando Fonseca, EPE (Amadora Sintra).

6 Trabalho Futuro

O projeto aqui retratado, uma vez tratar-se de uma dissertação de mestrado, teve um prazo limitado, o que condicionou algumas estratégias que inicialmente foram pensadas como boa opção para este tipo de aplicações, fazendo assim com que fosse necessário optar por outras soluções de abordagem mais simples.

Embora a abordagem utilizada neste trabalho satisfaça o objetivo pretendido, existem alguns pontos que poderão ser repensados para uma melhor precisão e robustez da aplicação desenvolvida.

Um dos pontos que poderá melhorar a precisão na identificação do movimento/gesto efetuado, será a construção do esqueleto do utilizador, para que os cálculos sobre a determinação do movimento sejam feitos de uma forma menos "peculiar" e para que os resultados sejam mais próximos do real valor, diminuindo assim alguns falsos positivos, que são um dos problemas mais comuns nestes projetos.

Em relação à robustez, a utilização de uma câmara que contenha características para obter profundidade nas imagens capturadas, permite que, através dessa profundidade, se possam eliminar facilmente os pontos que não precisam de ser processados, diminuindo assim a sobrecarga de processamento e aumentando possivelmente a velocidade do processo de identificação, não só do utilizador, como também dos gestos executados por este. Esta estratégia poderá ser conseguida através de visão estereoscópica, nomeadamente usando imagens de duas câmaras e algoritmos de triangulação, para obter o nível de profundidade de cada ponto numa imagem, trabalhando assim com uma imagem 3D.

Como trabalho futuro poderá ser adicionado o reconhecimento de outros tipos de movimentos, gestos ou comandos de voz. Estes novos comandos podem fazer com que o robô se mova para encaminhar os utilizadores para certos locais no Hospital, e assim a aplicação fique mais interessante para todo o tipo de utilizadores.

Referências

Abaya, W. F., Basa, J., Sy, M., Abad, A. C., & Dadios, E. P. (2014). Low Cost Smart Security Camera with Night Vision Capability Using Raspberry Pi and OpenCV, (November).

Arlowe, H. D. (1992). Thermal Detection Contrast of Human Targets. *Proc. IEEE International Carnahan Conference on Security Technology*, 27–33.

Bradski, G. (2000). OpenCV library. Retrieved from http://opencv.org/

Bradski, G., & Kaehler, A. (2008). *Learning OpenCV: Computer Vision with the OpenCV Library*. *OReilly Media Inc* (Vol. 1). doi:10.1109/MRA.2009.933612

Carmo, M. R. costa do. (2013). *O Brincar no Hospital: Possibilidade de Recuperação da Saúde da Criança e do Adolescente*. Universidade Federal do Recôncavo da Bahia-UFRB.

Carvalho, A. M., & Begnis, J. G. (2006). Play in pediatric care units: Applications and perspectives . *Psicologia Em Estudo*, *11*(1), 109–117. doi:10.1016/j.aquaculture.2005.05.018

Cheung, S. S. (2017). Robust Background Subtraction with Foreground Validation for Urban Traffic Video Robust Background Subtraction with Foreground Validation for Urban Traffic Video, (July), 2330–2340. doi:10.1155/ASP.2005.2330

Dalal, N., & Triggs, B. (2005). Histograms of oriented gradients for human detection. In *Proceedings - 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2005* (Vol. I, pp. 886–893). doi:10.1109/CVPR.2005.177

Dyllon, D., Macedo, J. De, Araújo, S. R. F. De, & Moreno, E. D. (2016). Anais do WSCAD-WIC 2016 Workshop de Iniciação Científica Anais do WSCAD-WIC 2016.

Emami, S., & Suciu, V. P. (2012). Facial Recognition using OpenCV. *Journal of Mobile, Embedded and Distributed Systems*, *4*(1), 38–43. Retrieved from http://www.jmeds.eu/index.php/jmeds/article/view/57

Gavrila, D. . M. (1999). The Visual Analysis of Human Movement: A Survey. *Computer Vision and Image Understanding*, *73*(1), 82–98. doi:10.1006/cviu.1998.0716

Google Code Archive - Kinetic Space 2.0 (Open source). (2011). Retrieved from https://code.google.com/archive/p/kineticspace/

Gorunescu, F. (2011). *Data Mining*. *San Francisco, CA, itd: Morgan Kaufmann* (Vol. 12). doi:10.1007/978-3-642-19721-5

Han, J., & Bhanu, B. (2005). Human Activity Recognition in Thermal Infrared Imagery. *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.

Hassanpour, R., Wong, S., & Shahbahrami, A. (2008). Vision - Based Hand Gesture Recognition for Human Computer Interaction : A Review. *IADIS International Conference on Interfaces and Human Computer Interaction*, 25--27.

Hu, Z., Chen, M., Chu, R., & Lim, H. (2010). Human arm estimation using convex features in depth images. In *2010 IEEE International Conference on Image Processing* (pp. 3269–3272). IEEE. doi:10.1109/ICIP.2010.5651215

Irani, R. (2016). Start programming on Raspberry Pi with Python - Open Source For You. Retrieved from http://opensourceforu.com/2016/10/programming-raspberry-pi-with-python/

Kochláň, M., Hodoň, M., Čechovič, L., Kapitulík, J., & Jurecka, M. (2014). WSN for Traffic Monitoring using Raspberry Pi Board. *Federated Conference on Computer Science and Information Systems*, *2*, 1023–1026. doi:10.15439/2014F310

Krishna, Mg., & Srinivasulu, A. (2012). Face Detection System on AdaBoost Algorithm Using Haar Classifiers. *International Journal of Modern Engineering Research (IJMER)*, *2*(5), 3556–3560. Retrieved from www.ijmer.com

Lementec, J.-C., & Bajcsy, P. (2004). Recognition of arm gestures using multiple orientation sensors: gesture classification. *Proceedings. The 7th International IEEE*

Conference on Intelligent Transportation Systems (IEEE Cat. No.04TH8749), 965–970. doi:10.1109/ITSC.2004.1399037

Lisetti, C. L., & Schiano, D. J. (2000). Automatic facial expression interpretation: Where human-computer interaction, artificial intelligence and cognitive science intersect. *Pragmatics & Cognition*, *8*(1), 185–235. doi:10.1075/pc.8.1.09lis

Mallick, S. (2016). Histogram of Oriented Gradients | Learn OpenCV. Retrieved from http://www.learnopencv.com/histogram-of-oriented-gradients/

Mohamed, S. S., Tahir, N., Adnan, R., & Mara, U. T. (2010). Background Modelling and Background Subtraction Performance for Object Detection, 236–241.

Nguyen, K. H. (2000). Method And Apparatus For Real-Time Gesture Recognition, 11.

Patsadu, O., Nukoolkit, C., & Watanapa, B. (2012). Human gesture recognition using Kinect camera. *Computer Science and Software Engineering (JCSSE), 2012 International Joint Conference on*, 28–32. doi:10.1109/jcsse.2012.6261920

Pok, L. Y. (2004). Gesture Recognition Using Web Camera. *School of Graduate Studies. Universiti Putra Malaysia in Fulfillment of the Requirements for the Degree of Master of Science*.

Poppe, R., & Poel, M. (2008). Body-part templates for recovery of 2D human poses under occlusion. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *5098 LNCS*, 289–298. doi:10.1007/978-3-540-70517-8_28

Raspberry Pi Foundation. (2015). Raspberry Pi 2 Model B. Retrieved from https://www.raspberrypi.org/products/raspberry-pi-2-model-b/

Raspberry Pi Foundation. (2016a). Camera Module. *Product Data*, 10400. Retrieved from https://www.raspberrypi.org/products/camera-module-v2/

Raspberry Pi Foundation. (2016b). Raspberry Pi 3 Model B. Retrieved from https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

Raspberrypi.org. (2016). FrontPage - Raspbian. Retrieved from https://www.raspbian.org/

Sachdeva, P., & Katchii, S. (2014). Review Article A Review Paper on Raspberry Pi,

4(6), 3818–3819.

Senthilkumar G, Gopalakrishnan K, & Satish Kumar. (2014). Embedded image capturing system using raspberry pi system. *International Journal of Emerging Trends & Technology in Computer Science*, *3*(2), 213–215. Retrieved from http://www.hades.in/BasePapers/Embedded/Journal/General/HEM (46).pdf

Suarez, J., & Murphy, R. R. (2012). Hand gesture recognition with depth images: A review. *Proceedings - IEEE International Workshop on Robot and Human Interactive Communication*, (September 2012), 411–417. doi:10.1109/ROMAN.2012.6343787

Tyapi, L., & Sowmya, K. S. (2015). Real Time Human Detection from Video Surveillance. *International Journal of Innovative Research in Computer and Communication Engineering*, *3*(5), 4413–4417. doi:10.15680/ijircce.2015.0305059

Venetsky, L., & Tieman, J. W. (2008). Robotic Gesture Recognition System, (19).

Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. *Computer Vision and Pattern Recognition (CVPR)*, *1*, I--511--I--518. doi:10.1109/CVPR.2001.990517

Viola, P., & Jones, M. (2001). Robust Real-time Object Detection, 1–25.

Wagner, P. (2012). Face Recognition with OpenCV2. *Matlab Instructions*, 1–26. Retrieved from http://face-rec.org/source-codes/%5Cnpapers3://publication/uuid/8429B995-3218-4B1B-BA34-69A624D00FB6

Wu, Y., & Huang, T. S. (1999). Vision-Based Gesture Recognition: A Review. *Gesture-Based Communication in Human-Computer Interaction. International GestureWorkshop, GW'99 Gif-Sur-Yvette, France, March 17-19, 1999 Proceedings*, 103–115. doi:10.1007/3-540-46616-9_10

Yang, M., Ahuja, N., & Tabb, M. (2002). Extraction of 2D Motion Trajectories and Its Application to Hand Gesture Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *24*(8), 1061–1074.

Zivkovic, Z. (2004). Improved adaptive Gaussian mixture model for background

subtraction. *Proceedings of the 17th International Conference on Pattern Recognition*, *2*(2), 28–31 Vol.2. doi:10.1109/ICPR.2004.1333992

Anexos

```python
# import the necessary packages
from imutils.object_detection import non_max_suppression
import numpy as np
import argparse
import imutils
import time
import cv2
import os

# construct the argument parser and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-v", "--video", help="path to the video file")
ap.add_argument("-a", "--min-area", type=int, default=150, help="minimum area size")
ap.add_argument("-m", "--max-area", type=int, default=2500, help="maximum area size")
args = vars(ap.parse_args())

# if the video argument is None, read from webcam (live)
if args.get("video", None) is None:

    camera = cv2.VideoCapture(0)
    camera.set(3,370)
    camera.set(4,240)
    camera.set(5,3)
    time.sleep(2.0)

# otherwise, read from a video file
else:

    camera = cv2.VideoCapture(args["video"])
    time.sleep(2.0)


# initialize the HOG descriptor/person detector
hog = cv2.HOGDescriptor()
hog.setSVMDetector(cv2.HOGDescriptor_getDefaultPeopleDetector())

# initialize the Background Subtraction function
fgbg = cv2.createBackgroundSubtractorMOG2(10,15)

# initialize the Haar Cascade Classifier
face_cascade = cv2.CascadeClassifier('/home/pi/opencv-3.1.0/data/haarcascades/haarcascade_frontalface_default.xml')
firstFrame = None

# Some required flags and counters
person_flag = 0
arm_pos = 500
finish_move = 0
start_sound = 0
wait_loops = 0
sound = 0
c = 0
start = time.time()

# loop over the image paths
while True:

    (s, image) = camera.read()
    image = imutils.resize(image, width=320, height=240)
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    #looking for an objet with the format of a person
    (rects, weights) = hog.detectMultiScale(image, winStride=(8, 8), padding=(16, 16), scale=1.05)
    rects = np.array([[x, y, x + w, y + h] for (x, y, w, h) in rects])
    pick = non_max_suppression(rects, probs=None, overlapThresh=0.65)

    if firstFrame is None:
        firstFrame = image
        fgmask = image
        continue

    frameDelta = cv2.absdiff(firstFrame, image)
    fgmask = fgbg.apply(frameDelta,fgmask,0.03)

    (_, cnts, a) = cv2.findContours(fgmask.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

```

```
    # draw the bounding boxes
    for (xA, yA, xB, yB) in pick:
        roi_gray = gray[yA:yA+yB, xA:xA+xB]
        roi_color = image[yA:yA+yB, xA:xA+xB]

        #probably a person, looking for a face to confirm (object/person surrounded with a green rectangle)
        cv2.rectangle(image, (xA, yA), (xB, yB), (0, 255, 0), 2)

        #looking for a face inside of the rectangle of the possible person
        faces = face_cascade.detectMultiScale(gray, 1.03, 3, 0, (1,1), (40,40))

        for (ex,ey,ew,eh) in faces:

            #person confirmed through face found(face surrounded with a blue rectangle)
            cv2.rectangle(image,(ex,ey),(ex+ew,ey+eh),(255,0,0),2)

            if faces.shape[0] == 1:

                if person_flag == 0:

                    #person found and waiting for the first welcome robot sound
                    if start_sound == 0:
                        os.system('omxplayer -o alsa ola_nando.mp3 &')
                        start_sound = 1

                else:

                    if person_flag == 1:

                        #person found and robot looking for movements made by him/her
                        for c in cnts:

                            # if the contour is too small, ignore it
                            if cv2.contourArea(c) < args["min_area"]:
                                continue

                            if cv2.contourArea(c) > args["max_area"]:
                                continue

                            # compute the bounding box for the contour of movements, draw it on the frame (movement surrounded with a red rectangle)
                            (x, y, w, h) = cv2.boundingRect(c)

                            if (x > xA and x < xB) or (x+w > xA and x+w < xB):

                                rect = cv2.minAreaRect(c)
                                box = cv2.boxPoints(rect)
                                box = np.int0(box)
                                cv2.drawContours(image,[box],0,(0,0,255),2)

                                #movements above the bottom line defining the face
                                if y < ey+eh:

                                    #upper left side movement
                                    if x > ex:

                                        if sound == 0:
                                            os.system('omxplayer -o alsa ola_outra_vez.mp3 &')
                                            sound = 1

                                    #upper right side movement (finish move)
                                    else:

                                        if sound == 0:
                                            os.system('omxplayer -o alsa entao_adeus_1.mp3 &')
                                            sound = 2

                                #movements below the bottom line defining the face
                                else:

                                    #left side
                                    if x > ex and arm_pos != 500:

                                        #upward movement on the left side
                                        if rect[0][1] < arm_pos:

                                            if sound == 0:
                                                os.system('omxplayer -o alsa vamos_brincar.mp3 &')
                                                sound = 1

                                        #downward movement on the left side
                                        else:

                                            if sound == 0:
                                                os.system('omxplayer -o alsa tartaruga.mp3 &')
                                                sound = 1
```

```
                                    #right side
                                    else:

                                        if x < ex and arm_pos != 500:

                                            #upward movement on the right side
                                            if rect[0][1] > arm_pos:

                                                if sound == 0:
                                                    os.system('omxplayer -o alsa conta_ate_3.mp3 &')
                                                    sound = 1

                                            #downward movement on the right side
                                            else:

                                                if sound == 0:
                                                    os.system('omxplayer -o alsa carapaca.mp3 &')
                                                    sound = 1

                                arm_pos = rect[0][1]

            else:

                #person lost or error finding person
                person_flag = 0
                start_sound = 0
                continue

    # show the output images
    cv2.imshow("Image", image)
    cv2.imshow("MOG", fgmask)
    key = cv2.waitKey(1) & 0xFF

    #sound time control (when the sound = 1 means that the robot is emitting a sound and for a while does not emit another sound over that)
    if sound == 1:

            wait_loops += 1

            if wait_loops == 12:
                    os.system('omxplayer -o alsa proximo_movimento_1.mp3 &')
                    arm_pos = 500

            if wait_loops == 18:
                    sound = 0
                    wait_loops = 0

    #waiting for robot to emit the last sound before finish the program
    if sound == 2:

            wait_loops += 1

            if wait_loops == 5:
                finish_move = 1
```

```
    #waiting for robot to emit the welcome sound
    if start_sound == 1 and person_flag == 0:

            wait_loops += 1

            if wait_loops == 20:
                os.system('omxplayer -o alsa proximo_movimento_1.mp3 &')
                wait_loops = 0
                person_flag = 1


    # if the `q` key is pressed, break from the lop
    if key == ord("q") or finish_move==1:
            os.system('sudo shutdown')
            break
```