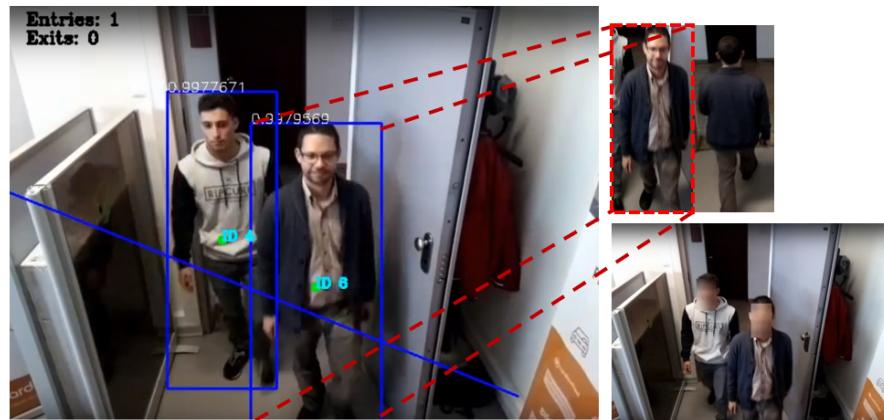




ISEL – INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA
ADEETC – ÁREA DEPARTAMENTAL DE ENGENHARIA DE ELECTRÓNICA E
TELECOMUNICAÇÕES E DE COMPUTADORES

LEIM
LICENCIATURA EM ENGENHARIA INFORMÁTICA E MULTIMÉDIA
UNIDADE CURRICULAR DE PROJETO

câmera User Tracking with Obfuscation



Gonçalo Ferreira nº43779

Gonçalo Adolfo nº43802

Frederico Costa nº44094

Orientador(es)

Professor engenheiro Pedro Jorge

Professor engenheiro André Lourenço

20 de Junho de 2019

Resumo

Neste projeto abordamos metodologias de deteção de pessoas, faces, métodos de tracking, de contagem de pessoas em entradas e saídas e técnicas de correspondência de pessoas, todos estas parte de um sistema que permite a contagem de pessoas que entram e saiam de um local e consequentemente a sua correspondência. Foram aplicados métodos de deteção de pessoas utilizando regiões resultantes da subtração de imagens consecutivas, princípio estudado em unidades curriculares anteriores como Processamento de Imagem e Visão, e em termos de tecnologias novas tirámos partido de tópicos de aprendizagem profunda como redes neurais convolucionais.

Para o sistema de tracking abordámos duas estratégias diferentes para descobrir a localização de uma pessoa em frames consecutivas, isto é, o seu tracking, estas estratégias são o tracking utilizando o centroide mais próximo das pessoas entre frames consecutivas e o tracking através da comparação das bounding boxes das pessoas detetadas através do algoritmo "Intersection over Union" ou "Jaccard Indice".

Para a correspondência, ou "matching", das pessoas foram estudadas algumas características que permitem-se um matching o mais robusto possível. Características como os histogramas de cor unidimensionais Hue e Saturação, bem como o histograma bidimensional HS, as áreas das pessoas, o descriptor estruturante de cor e o de contornos do padrão MPEG-7.

Devido ao Regulamento de Proteção de dados desenvolvemos um sistema capaz de ocultar as identidades das pessoas que possam ser capturadas em vídeo. Este sistema utiliza deteção de faces através de métodos com Haar Cascades ou com uma rede neuronal convolucional, bem como a deteção de pessoa que permite obter um sistema de ofuscação mais preciso e robusto.

Os sistemas desenvolvidos são combinados em uma aplicação a ser executada no computador de placa única Raspberry Pi que tira partido de um co-processador Edge TPU com a função de suportar e realizar a inferência da redes neurais convolucionais pré-treinadas. Esta aplicação permite a instalação do sistema utilizando uma câmara para realizar a contagem e matching de pessoas que essa câmara captura, sendo os resultados seja de contagem, de matching e vídeos resultantes da ofuscação de faces armazenados localmente ou remotamente.

Os sistemas de ofuscação de faces e o de contagem de pessoas são fornecidos com "serviços" a partir de uma aplicação Web implementada que permite um utilizador enviar vídeos para um servidor web Tornado que irá processá-lo e devolver as contagens de entradas e saídas de pessoas, ou um vídeo em que as faces estejam ofuscadas, conforme o pedido de um utilizador.

Abstract

In this project, we address methods for detecting people, faces, as well as tracking methods, counting people on entry and exit and people matching techniques, all part of a system that allows counting people entering and leaving a location and their subsequent matching.

We applied methods of detecting people using subtraction of consecutive images, a principle studied in previous curricular units such as Image and Vision Processing, and in terms of new technologies we took advantage of deep learning methods such as convolutional neuronal networks.

For the tracking system we used two different strategies to find the location of a person in consecutive frames that is, their tracking, also strategies of tracking by using the centroid closest to persons between consecutive frames and tracking by comparing the bounding boxes of people detected using the algorithm "Intersection over Union" or "Jaccard Index".

For the correspondence, or "matching" of people we studied which characteristics would allow a more robust matching. Characteristics such as the one-dimensional color histograms of Hue and Saturation, as well as the two-dimensional histogram Hue-Saturation, the area of a person, the color structure descriptor and edges histogram from the MPEG-7 standard.

Due to the Data Protection Regulation we developed a system capable of hiding the identities of people who can be caught on video. This system uses face detection by applying Haar Cascades methods or by convolutional neural networks, as well as taking advantage of person detection to allow a more accurate and robust obfuscation system.

The developed systems are combined into an application to running on the Raspberry Pi single board computer that takes advantage of an Edge TPU co-processor that supports and performs inference on pre-trained convolutional neural networks for object detection. This application allows the installation of the system using a câmara to perform the counting and matching of people on footage from the câmara, and the results of people counting, matching and videos with face obfuscation are stored locally or remotely.

Face obfuscation and people counting systems are provided as "services" trough an implemented web application that allows a user to send videos to a Tornado web server that will process it and return the counting of people entries and exits, or a video where the faces are blurred, according to the user's request.

Agradecimentos

O desenvolvimento do projeto não seria possível sem a ajuda e disponibilidade de um conjunto de pessoas. Em primeiro lugar, gostaríamos de agradecer aos nossos orientadores engenheiro Pedro Jorge e engenheiro André Lourenço, pelas ideias e ensinamentos ao longo de todo o trabalho. Para além disso, agradecemos também a disponibilidade para nos ajudarem em tudo o que fosse necessário. Agradecemos também ao professor engenheiro Paulo Trigo pelo ensinamento relativo às etapas de desenvolvimento de um projeto e outras temáticas importantes, assim como pelas suas opiniões e ideias relativas ao trabalho. Agradecemos também a toda a equipa da *CardioID*, por toda a paciência e ajuda nos testes realizados durante o desenvolvimento do projeto, assim como por todo o material disponibilizado para tal. Um grande obrigado a todos, por nos possibilitarem o crescimento, enquanto estudantes mas também como pessoas e trabalho de equipa.

Também não podíamos deixar de agradecer às nossas famílias e amigos, que nos dão diariamente um apoio incondicional a nível emocional, que nos proporcionam a motivação necessária para cada dia de trabalho.

Conteúdo

Resumo	i
Abstract	iii
Agradecimentos	v
0.1 Introdução	1
0.2 Trabalho Relacionado	2
0.3 Arquitetura	3
0.3.1 Arquitetura Global - Projeto	3
0.3.2 Arquitetura Global - Servidor HTTP	4
0.4 Sistema de Contagem	5
0.4.1 Deteção de pessoas	8
0.4.2 Tracking	20
0.4.3 Algoritmos de Tracking	20
0.5 Modelo de Dados	22
0.5.1 Modelo Entidade - Associação	22
0.5.2 Modelo Relacional	23
0.5.3 Diagrama de classes-Base de dados/FMS	24
0.6 Extração de Características	26
0.6.1 Características da Região	26
0.6.2 Características da Rede Neuronal	30
0.7 Algoritmo de correspondência	33
0.7.1 Primeira Abordagem	34
0.7.2 Algoritmo Brute Force	36
0.7.3 Implementação	38
0.7.4 Algoritmo em tempo real	40
0.7.5 Visualização de correspondências	40
0.8 Sistema de Ofuscação	42
0.8.1 Deteção de Faces - Haar Cascades	43
0.8.2 Deteção de Faces - Res10 Net	45
0.8.3 Sistema de Ofuscação - Abordagem Alternativa	46
0.9 Implementação no Raspberry pi	47
0.9.1 Edge TPU	47
0.9.2 Aplicação Geral	48
0.10 Disponibilização dos algoritmos	49
0.11 Métricas de Desempenho	52
0.12 Validação e Testes	54
0.13 Conclusões	56
0.14 Trabalho Futuro	57
Bibliografia	59

Lista de Tabelas

1	Entradas e saídas do módulo	9
2	Tabela de quantificação para os 5 tipos de contorno	30
3	Pedidos HTTP	50
4	Resultados da correspondência(video de 19min)	54

Listas de Figuras

1	Exemplo do Sistema AXIS: People Counter (retirada da página web da AXIS)	2
2	Arquitetura	4
3	Arquitetura-Serviços	4
4	Diagrama de classes do sistema	5
5	Processamento de uma frame	5
7	Relacional	8
8	Relacional	8
9	Módulo deteção de pessoas	9
10	Diagrama classes - Deteção de Pessoas	9
11	Comportamento deteção pessoas metodologia clássica	10
12	Imagens originais	11
13	Pixéis de movimento	11
14	Cálculo imagem de fundo	11
15	Comparação de métodos	12
16	Melhoramento de imagem	13
17	Imagen melhorada	13
18	Classificação de pessoas	14
19	Rede Neuronal - Arquitetura Básica	15
20	Arquitetura Básica de Rede Neuronal Convolucional	16
21	Estimação de movimento - Fluxograma	18
23	Short caption	23
24	Relacional	24
25	Relacional	25
26	Histogramas de cor Hue	26
27	Histogramas de Saturação	27
28	Áreas das pessoas em instantes diferentes	28
29	Elemento estruturante do CSD	28
30	Quantificação dos espaços de cor HMMD	29
31	Descritores estruturantes de cor	29
32	Tipos de contornos no EHD	30
33	Histogramas do EHD	30
34	Histogramas de contornos	31
35	Output da última camada da rede MobileNet	31
36	Grafo de correspondências	33
37	Algoritmo de correspondência	34
38	Fluxo MyMatching	35
39	Cenários de colisão	36
40	Iteração algoritmo BruteForce	37
41	Problema algoritmo iterativo	38
42	Diagrama de classes - Algoritmos de correspondência	39
43	Diagrama de blocos - correspondência em tempo real	40
44	Correspondência em tempo real	40
45	Informações adicionais das correspondências	41

46	Diagrama de classes - Sistema de ofuscação	42
47	Camadas de atributos Haar	43
48	Atributos Haar aplicados a uma imagem	44
49	Deteção de face com Haar Cascades	45
50	Deteção de face com rede neuronal convolucional	45
51	Resultados de Sistema de Ofuscação	46
52	Coral USB Accelerator	47
53	Arquitetura	49
54	Diagrama de classes - Fase de análise	51
55	Inicialização do servidor	51
56	Relacional	52
57	Classe MethodValidation	53
58	Função de custo para 2500 iterações, k=10, t=0.5	54
59	Matriz de confusão entradas/saidas	55

0.1 Introdução

A monitorização de entradas e saídas de pessoas é um problema da atualidade, em diversos cenários: edifícios, transportes públicos, tráfego em avenidas, entre outros. Com a evolução da tecnologia, é possível recorrer a sistemas de processamento de imagem e inteligência computacional para solucionar o problema. O projeto consiste no desenvolvimento de um sistema, com a capacidade de detetar entradas e saídas de pessoas, em um determinado ambiente, e fazer a sua correspondência de identidade abstrata, não sendo exatamente necessário o conhecimento de que pessoa se trata. Em uma outra vertente, devido a restrições impostas pelo Regulamento Geral de Proteção de Dados, o armazenamento de informação não pode incluir imagens de pessoas que não estejam ofuscadas. Deste modo, impõe-se como objetivo o desenvolvimento de um sistema de ofuscação de faces.

Em suma, pretende-se o desenvolvimento de um sistema capaz de corresponder entradas e saídas de pessoas, assim como um sistema de ofuscação, das suas identidades, para possível armazenamento de informação. Para além do armazenamento de vídeos, visa-se o armazenamento de outras informações úteis como a hora de entrada e hora de saída de cada pessoa. Como tal, é possível identificar o seguinte conjunto de finalidades:

- Contagem de pessoas;
- Predição de tempos de espera (edifícios);
- Fluxo diário;
- Horas de "ponta";
- Tempo de viagem médio (transportes públicos);
- Entre outros.

O documento desenvolvido, apresenta uma estrutura que tem como objetivo o seguimento do desenvolvimento dos objetivos finais mencionados. Proporciona, ao leitor, o acompanhamento em diversos tópicos tais como: trabalho relacionado, arquitetura inerente, deteção de pessoas, algoritmos de tracking, sistema de contagem, modelo de dados, características extraídas da pessoa, algoritmo de correspondência de pessoas, algoritmo de ofuscação, integração com o Raspberry Pi e por fim a disponibilização de algoritmos desenvolvidos a outros utilizadores/aplicações.

0.2 Trabalho Relacionado

Quando se aborda a contagem de pessoas que entram e saem de um local, neste momento existem tecnologias desenvolvidas e dirigidas exclusivamente para este propósito, tecnologias à base de sensores de movimento, proximidade ou como no caso do nosso projeto, através de câmaras de vigilância. Através dos coordenadores deste projeto tomamos conhecimento da empresa AXIS que fornece e desenvolve sistemas de vigilância através de vigilância por vídeo e áudio, em termos da contagem de pessoas a AXIS tem diferentes abordagens em termos de contagem seja por câmaras instaladas a um ângulo de 90 graus com o solo ou através de sensores estéreos que permitem obter imagens com profundidade tridimensional para melhor precisão na deteção de pessoas, ou por exemplo na existência de sombras, ou luz solar direta.

A solução de contagem da AXIS estudada foi o sistema "AXIS: People Counter" que possibilita a contagem em ambos os sentidos em tempo real, bem como é possível a escalar o sistema em termos de números de equipamentos que se encontram em comunicação para sincronizar a contagem. A instalação deste equipamento não necessita de um computador dedicado visto que o processamento é todo feito localmente nesse mesmo equipamento, as imagens obtidas pela câmera não são armazenadas sendo apenas os dados de contagem guardados no equipamento durante 90 dias. Por fim os dados resultantes da contagem depois poderam ser utilizados para verificar o fluxo de entradas e saídas de certo local em certos dias.

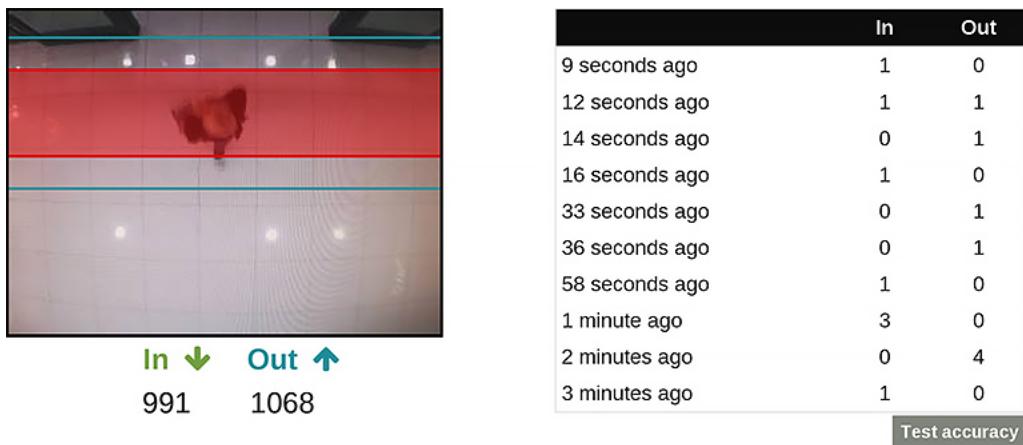


Figura 1: Exemplo do Sistema AXIS: People Counter (retirada da página web da AXIS)

0.3 Arquitetura

A arquitetura global do projeto foi concebida tendo um conjunto de fatores e restrições. Visto que o produto desenvolvido será implantando num autocarro, através de um mini computador designado por Raspberry Pi, existem algumas limitações relacionadas com o processamento e a capacidade de memória. Uma das propostas para combater estes problemas é a utilização dos serviços da Amazon AWS que disponibilizam serviços como: Amazon s3 e Amazon RDS. Uma vez que serão gravados vídeos de rotas ao longo do dia, e a memória do Raspberry pi depende de um cartão SD decidiu-se alojar os vídeos gravados na Amazon S3 bucket. O mesmo se aplica na base de dados, ou seja, existe uma base de dados local, mas o ideal é enviar para os servidores da Amazon depois de n rotas. Dado que o nosso algoritmo de Tracking e Ofuscação pode ser uma mais-valia, não só para o nosso projeto, mas também para outras pessoas que o possam querer usar dividimos a nossa arquitetura em duas partes: a primeira parte será destinada ao nosso projeto, e a segunda será a disponibilização dos nossos algoritmos através de um servidor HTTP a partir da tecnologia Node js a clientes interessados.

0.3.1 Arquitetura Global - Projeto

A primeira parte da arquitetura divide-se em dois sistemas: o sistema local(Raspberry pi) e o sistema de serviços da Amazon (Amazon AWS). Relativamente ao banco de dados(Fms e base de dados) e visto que as capacidades de armazenamento do raspberry estão limitadas a um cartão SD, decidiu-se usar serviços externos para armazenar toda a informação caso houvesse ligação com tais serviços, caso contrário seria armazenado localmente, em que as tecnologias a usar seriam as mesmas, e sincronizar quando fosse detetada nova conexão. As tecnologias que estão a ser usadas são as seguintes: Python(algoritmos), MariaDB(base de dados) e o Coral(co-processador para redes neurais convolucionais). Em relação ao sistema local este é dividido nos seguintes módulos:

- Main App
- Tracking algorithm
- Matching algorithm
- Obfuscation algorithm
- Local Database
- Edge-TPU

A aplicação Main terá como objetivo ligar e gerir a interação entre os vários módulos para posteriormente estabelecer comunicação com os serviços da Amazon. Relativamente à parte dos serviços da Amazon vão ser usados os seguintes:

- Amazon Simple Storage Service
- Amazon Relational Database Service

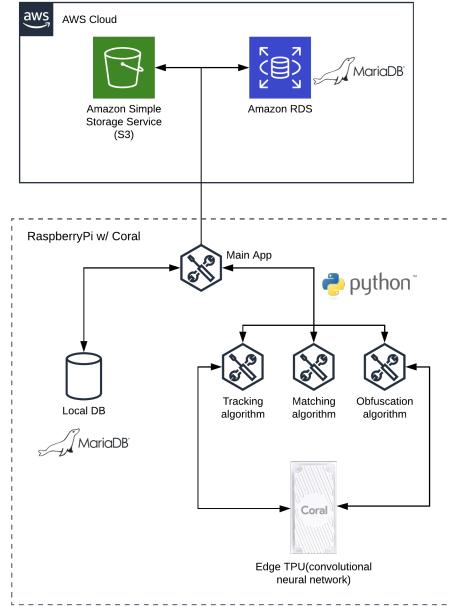


Figura 2: Arquitetura

0.3.2 Arquitetura Global - Servidor HTTP

Uma vez que os módulos anteriormente referidos funcionam independentemente é possível disponibilizar serviços a clientes através de um servidor HTTP. Um exemplo do que foi descrito seria uma cliente enviar um vídeo e ser-lhe-ia devolvido o mesmo vídeo, mas por exemplo com as faces ofuscadas(Obfuscation Algorithm). Para tal decidiu-se a utilização de uma ferramenta desenvolvida para o python designada por Tornado. Resumidamente é uma biblioteca assíncrona que permite criar servidores HTTP de uma forma simples através de Python. Esta tecnologia pode ser dimensionada para dezenas de milhares de conexões abertas, tornando-a ideal para pesquisas longas, WebSockets e outras aplicações que exijam uma conexão de longa duração com cada cliente.

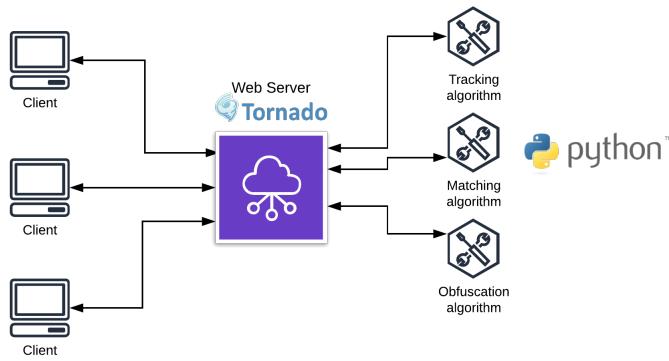


Figura 3: Arquitetura-Serviços

0.4 Sistema de Contagem

A Figura 4 ilustra o diagrama de classes inerente ao sistema de contagem. O sistema necessita de um mecanismo de deteção de pessoas assim como um objeto para o seu armazenamento permitindo a distinção entre entradas e saídas. O objeto *ClassicMethod* implementa uma metodologia clássica de deteção de pessoas com base na deteção de movimento. No entanto, outros mecanismos poderão ser abordados desde que respeitando o contrato desenvolvido com a interface *PersonDetection*.

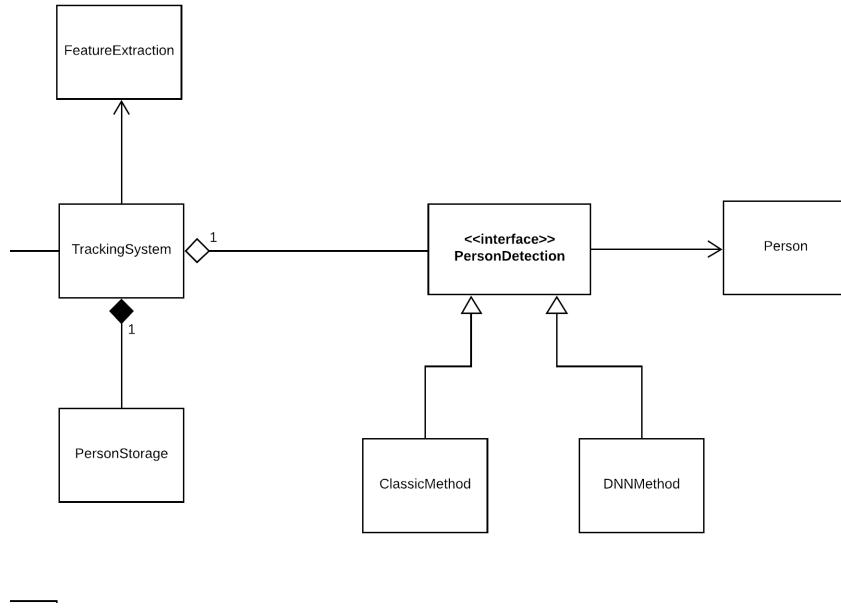


Figura 4: Diagrama de classes do sistema

O sistema efetua o processamento frame a frame. A Figura 5 desenha a sequência de blocos intrínsecos à sua computação:

- detetar as pessoas na frame com base num sistema de deteção de pessoas;
- efetuar a correspondência de pessoas com base na frame anterior;
- verificar se houve alguma pessoa no qual ultrapassou a linha de contagem classificando o evento como entrada ou saída;
- armazenar as pessoas no qual originaram um evento na respetiva estrutura de dados(entrada/saídas).

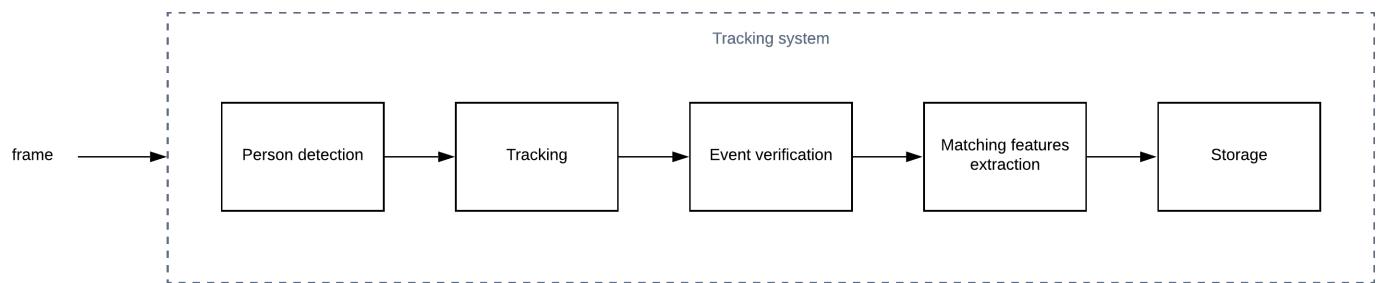


Figura 5: Processamento de uma frame

Uma vez apresentado o fluxo em que o algoritmo se efetua será explicado cada bloco de uma forma mais detalhada e profundando mais a parte da engenharia em relação à estrutura de classes e expressões matemáticas.

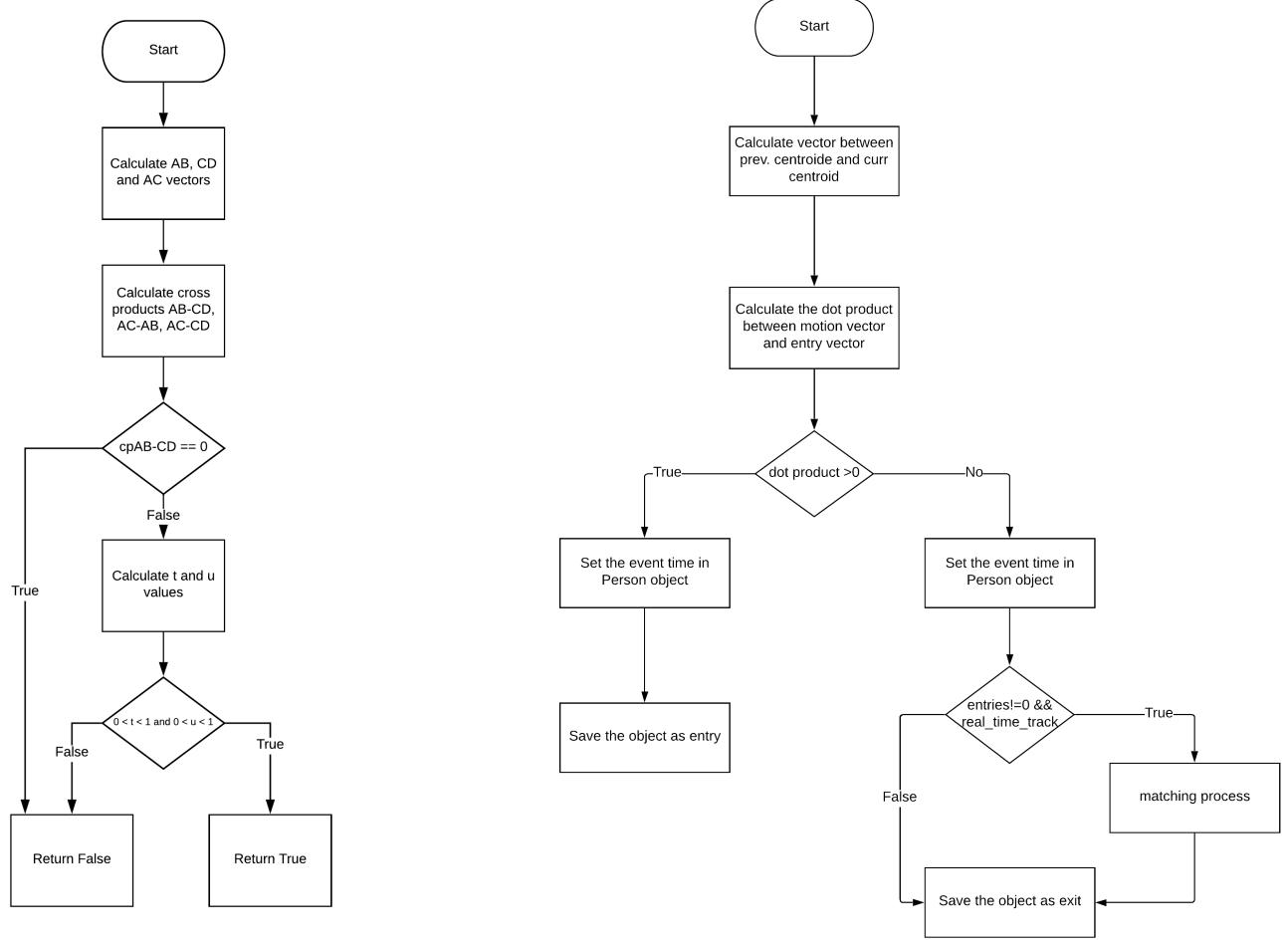
A intersecção da pessoa com a linha de contagem é calculada através do segmento de reta da pessoa(a partir de dois centroides), com o segmento de reta da linha virtual. A expressão da intersecção é dada pelas seguintes etapas e equações matemáticas (Figura 6a):

1. Calcular o segmento de reta da linha de contagem, dada pela expressão: $A + tAB$
2. Calcular o segmento de reta que une os centroides da pessoa, dada pela expressão: $A + C + uCD$
3. Intersecção entre as duas expressões: $A + tAB = C + uCD$
4. Verificar nível de perpendicularidade entre os vetores AB e CD . Se for 0 são paralelos e não se intersetam
5. Calcular t e u . Se ambos os valores estiverem entre 0 e 1 então os segmentos intersetam-se

Nota: os escalares t e u são calculados através da relação entre produtos vetoriais.

No caso da pessoa ter transposto a linha de contagem é necessário identificar se foi entrada ou saída, para tal foi concebido um algoritmo que faz a comparação entre o vetor dos centroides que intersetaram a linha de contagem e um vetor definido anteriormente. Como se pode observar na Figura 6b, o algoritmo é descrito nos seguintes passos:

1. Calcular o vetor dos centroides em que foi dada a intersecção;
2. Calcular o valor do produto escalar entre o vetor calculado anteriormente com um vetor predefinido(para saber se é entrada ou saída);
3. Se o valor for maior que 0(dado o vetor predefinido) consideramos que foi uma entrada, caso contrário contamos como saída;
4. Em ambos os casos armazenar-se as entradas/saídas num objeto que irá ser explicado posteriormente;
5. Em ambos os casos guardada-se a hora em que foi dada a intersecção;
6. No caso de ser uma saída, se o algoritmo de matching estiver a correr em direto e houverem entradas anteriormente será feito a correspondência desta pessoa;



Relativamente à extração de características, estas são extraídas no momento que uma pessoa interseque a linha virtual, seja esta uma entrada ou saída. É a partir das características extraídas que será feita a correspondência de pessoas de entrada com pessoas de saída. Os algoritmos de correspondência serão explicados posteriormente, mas resumidamente, as características que estão a ser usadas, são as seguintes:

- Área;
- Hue(cor);
- Saturação;
- Hue/Saturação(bi-dimensional);
- MPEG-7 Descriptor estruturante de cor;
- MPEG-7 Descriptor de contornos;

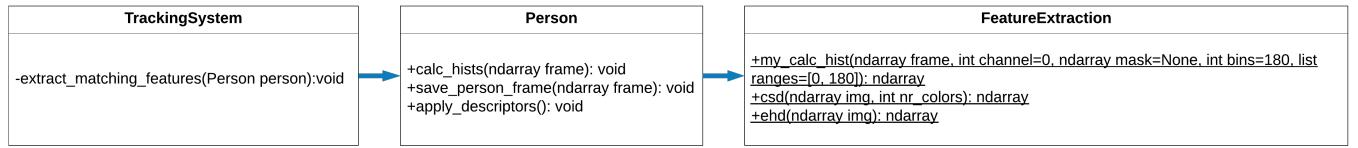
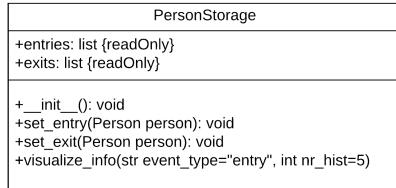


Figura 7: Diagrama de fluxo da extração de características

Todo o processo de extração de características é descrito pelo diagrama de fluxo da Figura 7. No momento em que a pessoa transpõe a linha de contagem, é despertado um evento e começa todo o processo de extração de características, sendo que o método *extract_matching_features* é o responsável pela tarefa. Posteriormente são calculados os histogramas de cor, saturação e a junção dos dois planos, assim como os descriptores de cor e contornos da norma MPEG-7. É de salientar que neste processo é armazenada a bounding-box da pessoa de maneira a visualizar os resultados da correspondência de uma maneira mais simples e amigável.

Finalmente para armazenar a contagem de pessoas em forma de entradas/saídas, foi concebida a classe *PersonStorage*, constituída por duas listas para armazenar as pessoas de entrada e saída. Visto que estas listas são formadas por objetos *Person* é facilitado todo o processo de criação de gráficos e imagens com informações de cada pessoa(caracteristicas,id).

Figura 8: Diagrama da classe *PersonStorage*

0.4.1 Deteção de pessoas

O módulo de deteção de pessoas, tal como mencionado na secção anterior, é um dos módulos necessários para o funcionamento do sistema de contagem. Recorrendo a métodos de visão por computador, permite a deteção de pessoas numa dada imagem. A figura 9 ilustra as entradas e saídas inerentes assim como as metodologias suportadas. O módulo implementa dois tipos de métodos:

1. Metodologia tradicional: deteção de pessoas baseada em classificação de regiões de movimento;
2. Aprendizagem profunda: deteção de pessoas com uma rede neuronal convolucional.

O número e tipo de entradas/saídas do módulo são independentes do tipo de metodologia adotada. No entanto, destaca-se a informação presente na tabela 1. Dependendo do método que se pretende utilizar, a entrada *frame* e a saída *debug frame* diferem quanto ao seu formato e quanto à sua interpretação. A saída *persons* não tem alterações consoante a metodologia, sendo uma lista de instâncias da entidade *Person* no qual foram detetadas na imagem de entrada.

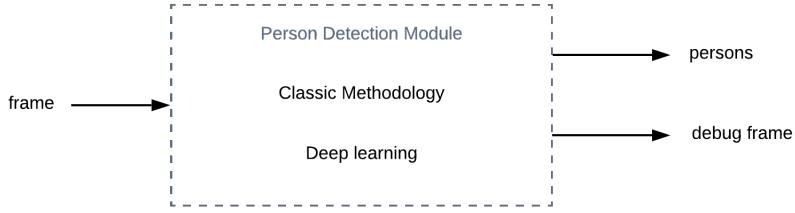


Figura 9: Módulo deteção de pessoas

Metodologia	Entrada/Saída	Formato	Interpretação
Tradicional	frame	ndarray-2D	Imagem em tons de cinzento
	debug frame	ndarray-2D	Imagen binária com as regiões de movimento detetadas
Aprendizagem Profunda	frame	ndarray-3D	Imagen blue-green-red
	debug frame	ndarray-3D	Imagen original com bounding boxes nas pessoas detetadas, assim como o valor associado à sua classificação (nível de confiança)

Tabela 1: Entradas e saídas do módulo

A figura 10 desenha o diagrama de classes, inerente à estrutura de *software*, com o objetivo de implementar os requisitos do módulo de deteção de pessoas. A interface *PersonDetection* define o "contrato" necessário para cada um dos métodos de deteção:

- Um método *getPersons(frame)* para obter uma lista com as pessoas detetadas;
- Propriedade *debugFrame* de modo a obter uma frame para efeitos de *debug*.

Tal como mencionado anteriormente, uma pessoa é representada através da entidade *Person*.

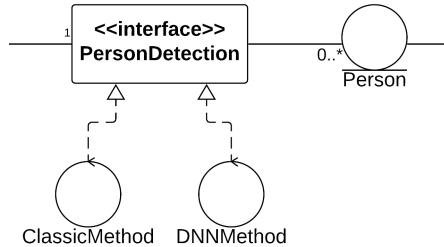


Figura 10: Diagrama classes - Detecção de Pessoas

Metodologia Clássica

A deteção de pessoas baseada em uma ideia tradicional, ou seja, sem a intervenção de modelos de aprendizagem automática, tem como base a classificação de regiões de movimento. A figura 11 representa o diagrama de blocos inerente ao comportamento. Tal como observado na figura, é composto pela seguinte sequência de atividades:

1. Detecção de movimento: identificação de pixéis de movimento;

2. Melhoramento de imagem: operações com o objetivo de melhorar a imagem, no sentido de eliminar ruído e enaltecer os objetos presentes;
3. Extração de regiões conexas: método, de visão por computador, para identificação de conjuntos de pixels de movimento conectados (regiões de movimento);
4. Extração de características: representação de cada região com um vetor de características;
5. Classificação: utilização de um modelo de classificação com o objetivo de atribuir uma classe a cada região;
6. Atualização da imagem auxiliar: atualização da imagem utilizada para a deteção de movimento (imagem do instante anterior ou imagem de fundo).

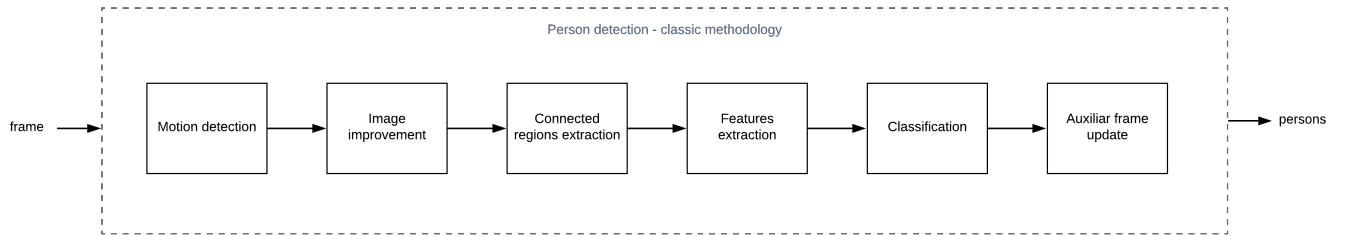


Figura 11: Comportamento deteção pessoas metodologia clássica

O comportamento *ClassicMethod*, objeto que implementa a metodologia tradicional, suporta métodos de subtração de imagem para detetar movimento, ou seja, baseiam-se na alteração de intensidade dos pixels (derivada temporal): subtração com a imagem anterior, subtração com imagem de fundo. As equações 1 e 2 representam as fórmulas matemáticas, de cada um dos métodos, para determinar os pixels ativos. A equação 1 é relativa à subtração com a imagem anterior, sendo que I_n é a imagem no instante atual n e T_n o valor do limiar de decisão. Segundo o mesmo raciocínio, na fórmula 2, a única diferença deve-se ao facto da subtração ser entre a imagem no instante atual e uma imagem de fundo B_n . Caracterizando a imagem no instante anterior e a imagem de fundo como "imagem auxiliar", é possível descrever as equações como: um dado pixel, na posição (r, c) , é pixel de movimento se o módulo da diferença com o pixel na posição (r, c) na imagem auxiliar for superior a um dado limiar de decisão. A saída deste bloco trata-se de uma imagem binária, onde os pixels de movimento apresentam valor 1.

$$|I_n(r, c) - I_{n-1}(r, c)| > T_n(r, c) \quad (1)$$

$$|I_n(r, c) - B_n(r, c)| > T_n(r, c) \quad (2)$$

O objeto implementado permite a parametrização do limiar de decisão. Tal como se pode observar pelas equações, este é responsável por controlar a sensibilidade a diferenças de intensidade. Quanto menor for este valor, mais sensível irá ser na determinação de pixels ativos, tendo tendência a gerar uma imagem binária de saída mais ruidosa. Por outro lado, se este valor for bastante elevado, apenas se pretende obter regiões de elevada diferença de intensidade. As figuras 51 e 13 representam um caso exemplificativo do impacto do valor de decisão. Note-se que, com valor $T_n = 50$, foram obtidos menos pixels de movimento ruidosos. No entanto, a região que compõe a pessoa a detetar também obteve menos pixels ativos. Em suma, o valor ideal a utilizar depende do nível de movimento que se pretende detetar.



Figura 12: Imagens originais

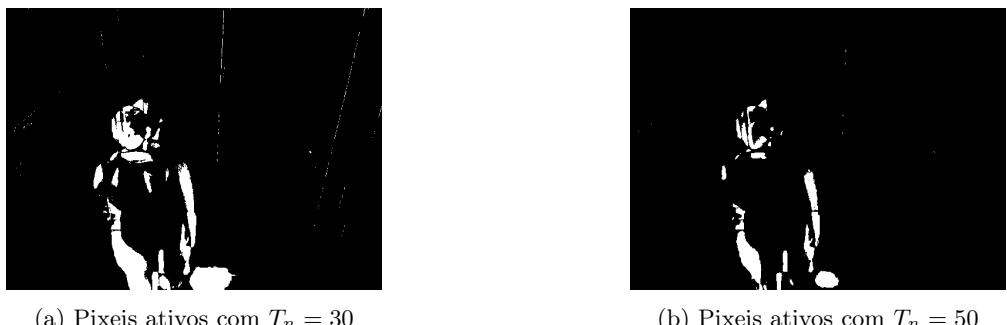


Figura 13: Pixel de movimento

No exemplo anterior que compõem as figuras, foi utilizado o método de subtração de imagens consecutivas. No entanto, tal como já mencionado, é suportado a subtração com imagem de fundo (a decisão de qual a metodologia a utilizar é definida pelo parâmetro *method*). Perante o caso de utilização de imagem de fundo, o comportamento *ClassicMethod* disponibiliza um método para o seu cálculo cujas entradas, saídas e sequência de atividades são descritas pela figura 14. A imagem de fundo, de um determinado vídeo, trata-se de uma mediana temporal das primeiras *frames_thr* imagens. Após a aplicação da mediana temporal, a imagem é escrita na diretoria passada como parâmetro.

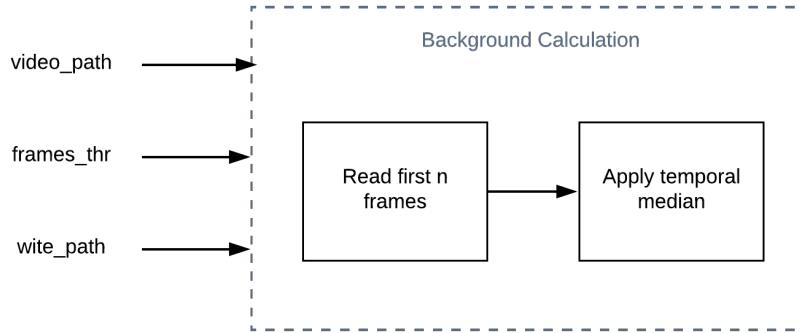


Figura 14: Cálculo imagem de fundo

Perante um dado caso de utilização, a imagem de fundo poderá não ser constante nas várias alturas do dia, devido a situações como a luminosidade ou até mesmo "efeitos naturais" como o símbolo de uma farmácia. Para contemplar estes cenários, existem várias abordagens possíveis:

- Utilização de mais do que uma imagem de fundo: aplicação da equação 2 para várias imagens de fundo. Apenas é considerado pixel ativo se for pixel ativo para todas as imagens de fundo. Abordagem mais utilizada para mudanças significativas de fundo;
- Atualização da imagem de fundo: em cada instante, é atualizada a imagem de fundo a utilizar na próxima iteração. Abordagem mais utilizada para pequenas mudanças no fundo como o caso de alterações de luminosidade.

O comportamento desenvolvido, implementa a atualização da imagem de fundo. A equação 3 descreve a sua atualização. O objetivo trata-se de pesar a imagem de fundo anterior com o valor dos pixels, na imagem atual, onde não foi detetado grandes alterações. O valor α controla o peso da imagem de fundo anterior. Consequentemente, $(1 - \alpha)$ será o peso a adotar na imagem atual. Quanto mais elevado for este valor, menor será o peso da imagem atual e por sua vez menor a alteração entre imagens de fundo entre instantes consecutivos. Relembre-se que apenas são tidos em conta os pixels não ativos na nova imagem de fundo.

$$B_{n+1}(r, c) = \begin{cases} B_n(r, c) & \text{se } (r, c) \text{ é pixel ativo} \\ \alpha B_n(r, c) + (1 - \alpha) I_n(r, c), & \text{caso contrário} \end{cases} \quad (3)$$

A figura 15 reflete as vantagens e desvantagens de cada um dos métodos de deteção de movimento. Dada uma imagem num instante t (figura 15a), foram aplicadas as equações 2 e 1 obtendo como saída os pixels ativos que se pode observar nas figuras 15b e 15c respetivamente. Pela observação das figuras, é possível afirmar que o método de subtração com fundo possibilita regiões "target" mais sólidas. No entanto, é totalmente dependente da imagem de fundo, sendo bastante sensível a movimentos que ocorram em objetos no fundo como o abrir de uma porta/janela ou pequenos movimentos na câmara mesmo não sendo visível a olho humano. A subtração com a imagem anterior, apesar de possuir menos pixels ruído não produz regiões "target" muito compactas devido à uniformidade dos pixels da região. Novamente, o método ideal a utilizar depende do caso de utilização. Tendo em conta que a câmara foi instalada na entrada de uma organização (*CardioID*), seria preferível a utilização de subtração de imagens consecutivas, devido ao impacto do movimento de abertura e fecho da porta na imagem de fundo. Note-se também que, até ao momento, os exemplos descritos não englobaram os restantes blocos do comportamento.

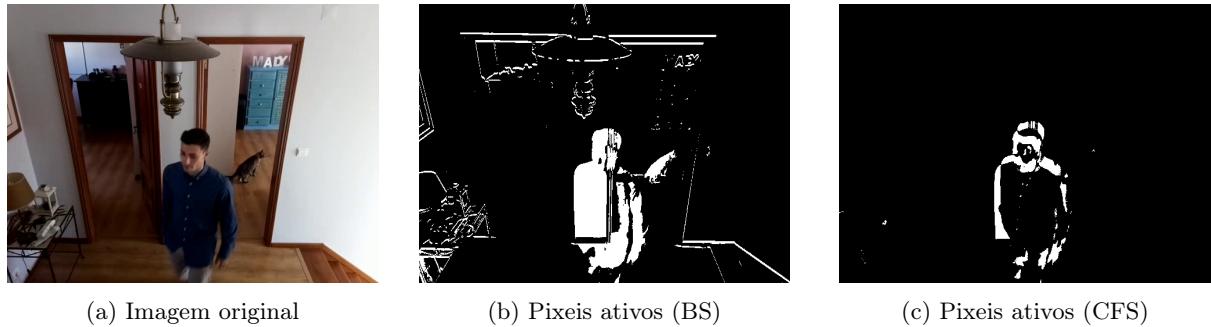


Figura 15: Comparaçao de métodos

O bloco de melhoramento de imagem tem um impacto significativo nos resultados obtidos. É possível observar na figura 15b que existe um ruído significativo nos pixels ativos provenientes do bloco anterior. Por outro lado, como já referido, os pixels ativos ilustrados na figura 15c não formam um região suficientemente compacta. Para resolução dos problemas mencionados, utilizam-se operações morfológicas. Neste tipo de operações, percorre-se os pixels da imagem com uma dada máscara, e consoante o tipo de operação obtém-se um valor para o pixel de saída. Destacam-se o seguinte conjunto de operações:

- Dilatação: o valor do pixel de saída é o máximo de todos os pixels inseridos na máscara;

- Erosão: o valor do pixel de saída é o valor mínimo de todos os pixels inseridos na máscara;
- Fecho: dilatação seguido de uma erosão;
- Abertura: erosão seguido de uma dilatação.

O conjunto de operações a aplicar depende do caso de utilização. Com esse intuito, fazendo a ligação com o software desenvolvido, ao instanciar um comportamento do tipo *ClassicMethod* é necessário passar como argumento uma função que segue o bloco descrito na figura 16. Desta forma, o utilizador do comportamento pode definir as operações a aplicar de modo a satisfazer os seus requisitos ficando, do ponto de vista do comportamento, o bloco de melhoramento de imagem a ser utilizado como uma caixa preta.

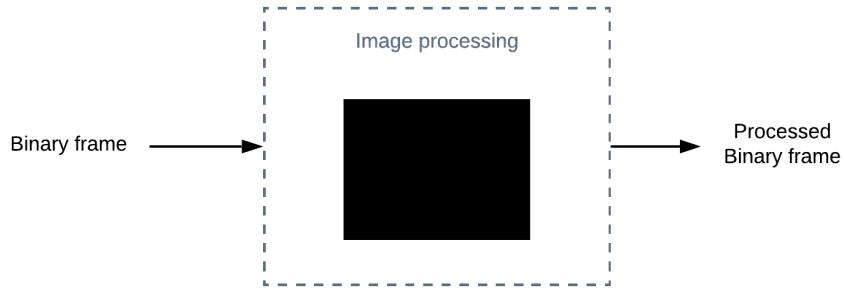


Figura 16: Melhoramento de imagem

A figura 17 ilustra um exemplo de aplicabilidade do bloco de melhoramento de imagem. Trata-se da imagem de saída após aplicação de uma dilatação, com um disco (10, 10), na imagem binária da figura 15c. Pela observação da imagem, é notável que a fronteira da região pessoa possui uma estrutura arredondada devido ao elemento estruturante disco e à sua dimensão. A operação dilatação deriva do problema identificado: a região pessoa não apresentava uma região sólida. Note-se que, logicamente, os pixels ruído também sofreram uma dilatação. No entanto, não será um problema para os blocos seguintes do módulo.



Figura 17: Imagem melhorada

A imagem de saída, após operações morfológicas, ainda não possui informação dos objetos detetados. Para tal, é necessário aplicar um algoritmo capaz de etiquetar regiões conexas, ou seja, conjunto de pixels ativos interligados entre si. Recorre-se à biblioteca *OpenCV* para aplicar o algoritmo união-procura implementado no método *connectedComponents*. O algoritmo percorre todos os pixels na imagem e segue os seguintes passos:

1. Verificar o valor da etiqueta dos pixels vizinhos (apenas os possíveis já etiquetados);
2. Se possuir algum vizinho já etiquetado, o pixel assume essa etiqueta;

3. Se não possuir nenhum vizinho etiquetado e o valor do pixel for igual a 1, o pixel assume a etiqueta nº etiquetas existentes + 1;
4. Se houver vizinhos com etiquetas diferentes atualiza a tabela de equivalência entre etiquetas e assume o valor de uma delas.

Em conclusão, restam os blocos de extração de características e classificação das regiões etiquetadas. Nestas etapas do módulo, não foi adotado um rigor significativo na identificação de pessoas devido aos casos de utilização do sistema. Transportes públicos e edifícios são o principal cenário de utilização do sistema desenvolvido. Neste tipo de cenários, raro é o movimento de área significativa não proveniente de um ser humano. Desta forma, apenas é extraída a área de cada região (nº de pixels) e são excluídas as regiões com área inferior a um dado limiar. As restantes regiões de movimento são classificadas como pessoa e representadas através da entidade *Person* (figura 18). Como já afirmado anteriormente, o módulo retorna uma lista de instâncias desta entidade no qual foram detetadas dada a imagem de entrada.

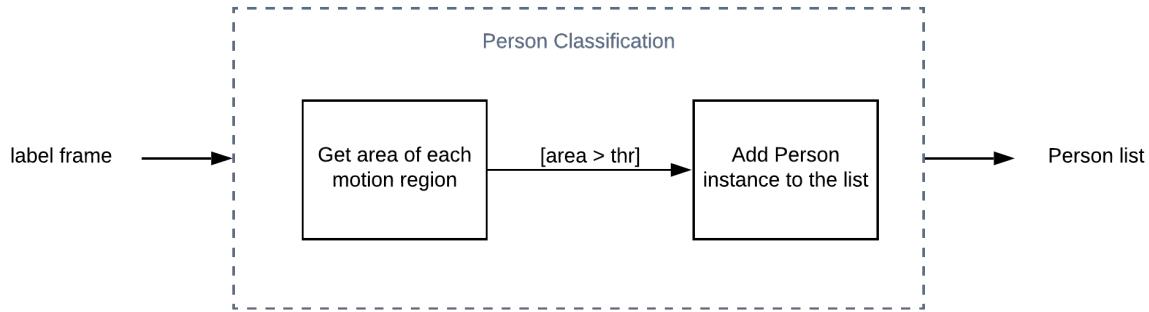


Figura 18: Classificação de pessoas

Em ambos os casos de utilização a câmara é colocada em uma posição fixa. No entanto, no caso de implantação em autocarros, o fundo encontra-se em constante alteração derivado do movimento do veículo. Para a *aplicação main*, é utilizada uma abordagem que utiliza redes neurais convolucionais para deteção de pessoas devido ao facto de possuir maior robustez a alterações no fundo. No entanto, com a utilização deste tipo de modelos, a complexidade temporal é significativamente superior. Para além disso, apresenta a desvantagem de não retornar os pixels ativos da pessoa, mas sim a sua *bounding box* que consequentemente influencia a extração de características (vetores a utilizar no algoritmo de correspondência).

Redes Neuronais

A inteligência artificial nos últimos anos tem vindo a evoluir de uma forma exponencial, continuando a crescer de tal forma que esta tecnologia será uma das maiores apostas em termos de investimento. A aprendizagem (Machine Learning) é um subtipo de inteligência artificial que consiste no estudo de algoritmos e modelos utilizados por certa máquina com o intento desta tomar decisões sem depender de um conjunto de instruções definidas, ou seja, esta máquina irá tomar decisões de acordo com padrões anteriormente "aprendidos".

Neste projeto demos um maior ênfase ao tipo de modelo de aprendizagem automática denominada de Redes Neuronais, visto que a explicação detalhada deste tópico seria muito extensa irá ser explicado em termos básicos em que consiste uma rede neuronal e posteriormente entraremos em detalhe qual o tipo de rede neuronal utilizada e os modelos correspondentes. As redes neurais são uma tecnologia com origens em meados de 1940, sendo que nas últimas décadas tornaram-se em um tema importante no campo da inteligência artificial, estas como o seu nome indica baseiam-se no cérebro humano, sendo que tentam replicar a forma como os humanos aprendem.

Basicamente uma rede neuronal é constituída por camadas de "input", camadas de "output" e normalmente camadas "escondidas" que irão converter o que entra na rede neuronal em algo que a camada de "output" consegue utilizar. Dando um exemplo que se aplica ao nosso projeto pode-se imaginar que certa imagem passa por uma rede neuronal esta contém uma camada que possibilita a deteção de contornos de quaisquer objetos, outra permite reconhecer texturas e quanto mais profunda a camada irá tornar-se identificar características cada vez mais específicas como olhos e bocas por exemplo.

Quando se aborda redes neurais é necessário considerar qual o seu tipo, tendo em consideração que neste projeto apenas foi estudado e utilizado apenas um tipo apenas iremos entrar em detalhe e explicar o seu conceito, bem como os diferentes modelos deste tipo de rede neuronal que tirámos partido.

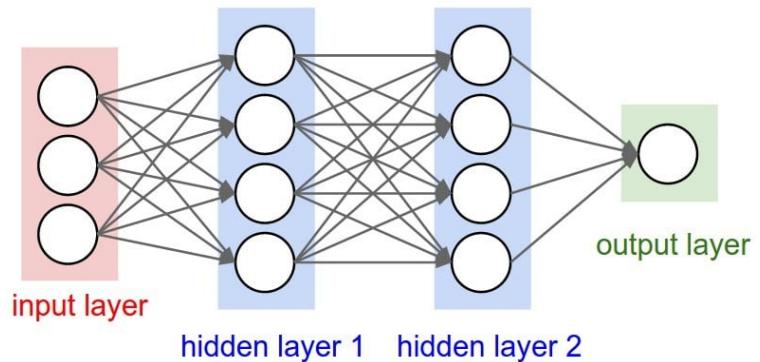


Figura 19: Rede Neuronal - Arquitetura Básica

Redes Neuronais Convolucionais

Uma rede neuronal é considerada "profunda" quando esta tem um nível maior de complexidade em comparação com rede neuronal básica, isto é, considera-se como rede neuronal profunda quando existe pelo menos uma camada escondida entre a camada de input e a camada de output. Para o objetivo de deteção seja de pessoas ou faces, após pesquisa, foi concluído as redes neuronais convolucionais seriam as mais indicadas para o nosso objetivo.

Este tipo de rede é utilizada principalmente no processamento de imagem, sendo que a lógica básica é que esta rede irá receber uma imagem, atribuir pesos a características de objetos na imagem e daí diferenciá-los e por fim classificá-los. Para compreensão do funcionamento deste tipo de rede é necessário entender o conceito de convolução, pooling, camada "fully connected" e classificação softmax, sendo o processo de convolução o que distingue esta rede de outros tipos de redes neuronais profundas.

De um resumido e básico uma imagem quando é processada por uma rede neuronal convolucional, irá passar por diferentes tipos de camadas (Fig.20), onde são realizadas diferentes operações:

- Convolução - Operação onde um kernel de $n \times n$ desloca-se pela imagem pixel a pixel e extraí características da imagem por multiplicação de matriz.
- Pooling - Reduz a dimensão espacial das características resultantes da convolução aplicando um filtro de média n por n a cada pixel (average pooling) ou um filtro do valor máximo dos pixels (max pooling).
- Camada Fully Connected - Nesta camada, as diferentes características locais extraídas (contornos, textura, formas, etc.) são combinadas num vetor de características que será utilizado na classificação pela última camada.
- Softmax - Atribui probabilidades decimais de pertencer a certa classe num problema multi-classe, sendo que a soma desses valores resulta em 1.

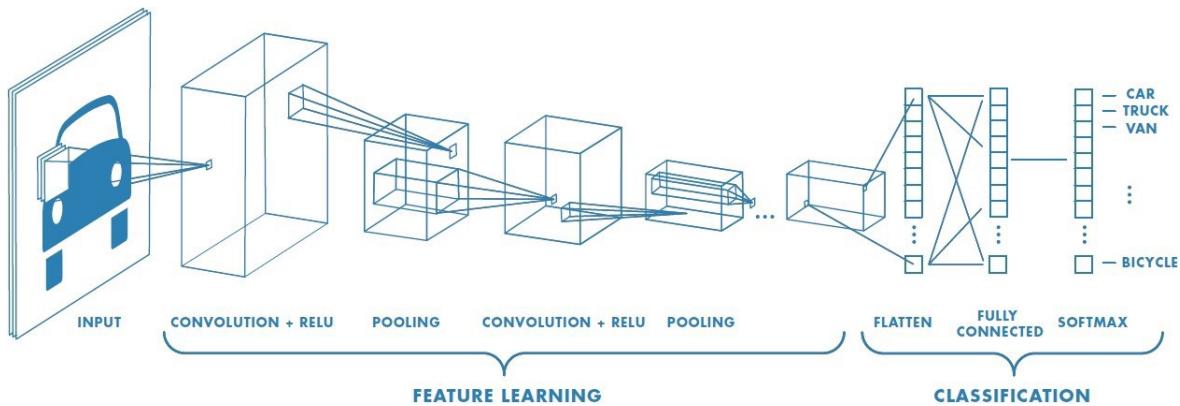


Figura 20: Arquitetura Básica de Rede Neuronal Convolutinal

Deteção de Pessoas - VGG Net

Na primeira abordagem, a implementação deste método de deteção foi realizada sem ter em consideração as limitações em termos de hardware no Raspberry Pi, ou seja, foi considerada como uma fase de testes com o objetivo de entender a aplicação de redes neuronais convolucionais na deteção de pessoas. Após o desenvolvimento do método de deteção de pessoas através de subtração de frames, conclui-se que este não seria a solução ideal para o objetivo do projeto, logo pesquisámos soluções alternativas, sendo a utilização de redes neuronais convolucionais a mais indicada para deteção de objetos capturados em frame.

Constatámos que a melhor forma que teria melhores resultados seria o treino de uma rede neuronal com a intenção de especificar a deteção para um determinado local, no entanto reconhecemos que para tal seria necessário um maior poder computacional, bem como uma grande dataset de pessoas. Chegámos à conclusão que a melhor solução seria tirar partido de redes neuronais pré-treinadas para a deteção de pessoas ou outros objetos, sendo a primeira encontrada a VGG Net em conjunto com o detetor SSD(Single Shot Multibox Detector).

Para a utilização deste modelo pré-treinado tirámos partido da framework "Caffe" de aprendizagem profunda que permitiu de uma forma facilitada e otimizada de realizar a deteção de pessoas em uma imagem, bem como a biblioteca OpenCV para realizar o pré-processamento necessário. A lógica desta implementação baseia-se em:

- Os ficheiros correspondentes a esta rede são lidos por método da biblioteca do "Caffe", um contendo a estrutura da rede em si e outro com os pesos utilizados em cada camada resultantes do processo de treino realizado.
- Aplicar um escalamento da imagem para 512 x 512, que é a resolução das imagens que modelo usou para a sua fase de treino.
- Realizada a subtração de média R/G/B à imagem.
- O que resulta do processamento é introduzido na rede e é realizado o "forward pass", processo de cálculo dos valores de output da última camada através dos valores de input da primeira.
- O output da deteção é uma lista de dimensão (1, 1, número de deteções, 7), sendo um exemplo de uma deteção [0. ,1. ,0.0604972 ,0.76823086 ,0.94985545 ,0.88575333 ,1.0422581] em que o terceiro valor corresponde á probabilidade esta deteção ser de uma pessoa, e os últimos 4 valores correspondem ás coordenadas da bounding box em percentagem relativa à imagem que foi escalada.
- Filtra-se as deteções indesejadas utilizando um limiar designado.
- Por fim as coordenadas são escaladas para o tamanho original da imagem, bem como são limitadas visto que ocorria casos em que estas coordenadas estavam fora da resolução da imagem.

Chegámos a rápida conclusão que este modelo não seria o ideal visto que o tempo de inferência (tempo de predição) para uma imagem era cerca de 2.6 segundos o que iria certamente aumentar quando implementado no Raspberry Pi e impossibilitar o processamento do sistema de contagem em tempo real. Numa tentativa de aumentar a velocidade de processamento foi implementado uma combinação da deteção de pessoas utilizando esta rede com a predição de movimento através do método de Optical Flow de Lucas-Kanade.

Predição de movimento - Optical Flow

Com o objetivo de diminuir o tempo de processamento de um certo vídeo ou conjunto de imagens capturadas, o método de deteção foi complementado com a lógica de predição de movimento. Esta predição baseia-se no conceito de Optical Flow utilizando o método de Lucas-Kanade, este basicamente consiste em utilizar a intensidade de um pixel ou vizinhos de frames consecutivas de modo a estimar onde esse valor de intensidade irá mover-se.

Neste caso utilizámos o centróide calculado da pessoa detetada de modo a calcular onde é que este irá estar na próxima frame, tendo em conta o valor da sua intensidade. A lógica seria não utilizar constantemente a deteção de pessoa, seria complementá-la com a predição de movimento, ou seja, de n em n frames era realizada a deteção, guardado o centróide dessa pessoa e nos próximos frames era calculado o centróide estimado da pessoa substituindo o anterior, lógica representada na Fig.46.

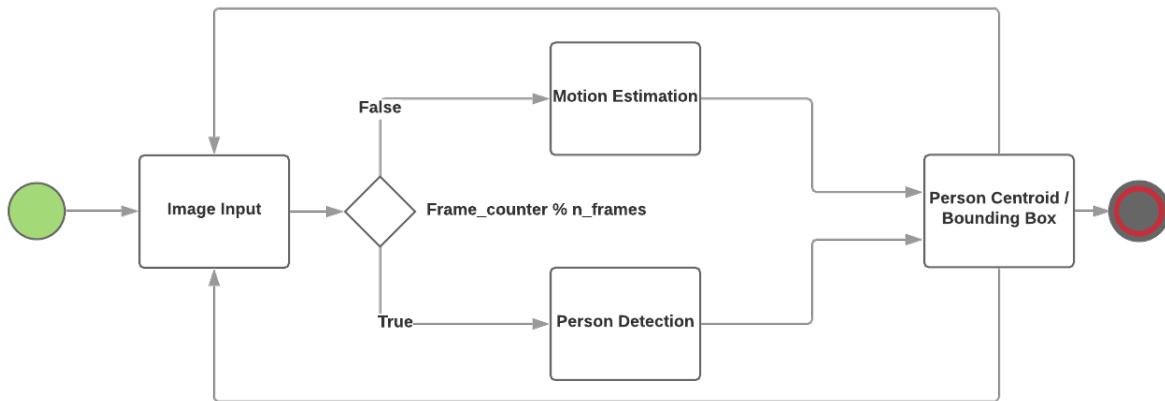


Figura 21: Estimação de movimento - Fluxograma

Alguns resultados demonstraram que seria possível seguir uma pessoa detetada nesses frames em que o seu movimento é estimado, no entanto as desvantagens desta solução prevaleceram sobre as vantagens, desvantagens que iriam dificultar posteriormente o processo de contagem e extração de características:

- O cálculo do centróide dependia da intensidade dos pixels vizinhos, logo se certa pessoa realizasse algum movimento brusco o centróide estimado iria ser mal calculado e esta situação só seria resolvida na próxima deteção.
- A bounding box da pessoa seria constante durante o processo de estimativa de movimento, visto que apenas é calculado onde este centróide irá estar no próximo frame.
- Se o intervalo de frames onde é realizada a estimativa de movimento for muito alto, uma pessoa pode aparecer em cena e não ser detetada durante esse intervalo, sendo possível esta pessoa não ser contada na entrada ou saída.

Deteção de Pessoas - MobileNets

Nesta fase do projeto através de melhor pesquisa foi encontrado o modelo de rede neuronal convolucional MobileNets lançada pela Google e desenvolvida com o objetivo de ser possível realizar previsões e classificações num dispositivo móveis com menor capacidade de processamento. O modelo que utilizámos para o nosso projeto é mais uma vez um conjunto de duas redes, a MobileNets e a SSD, sendo que a primeira utilizada como extrator de características de alto nível e a segunda utilizada como detetor e classificador.

O modelo de raiz permitia a deteção de 20 classes diferentes (carro, mota, pessoa, etc.), no entanto para o nosso projeto apenas era necessária a deteção de pessoas, logo teve de ser feita uma pequena "filtragem" nos resultados de deteção. A única diferença em termos de implementação da primeira tentativa com a rede VGG Net foi que a primeira apenas estava treinada para detetar pessoas e a MobileNet permitia a deteção de outros objetos, no entanto a diferença em relação ao tempo de inferência e precisão na deteção foi notável, diminuindo o tempo de inferência para cerca de 0.1 segundos, demorando o VGG Net 26 vezes mais.

0.4.2 Tracking

Video Tracking é o processo de detetar um objeto em movimento, no caso do projeto pessoas, ao longo do tempo através de uma câmara. Todo este processo, tem como objetivo associar as várias pessoas em cena ao mesmo identificador ao longo das imagens do vídeo. Existem vários algoritmos capazes de produzir o que foi descrito, sendo que dois foram implementados:

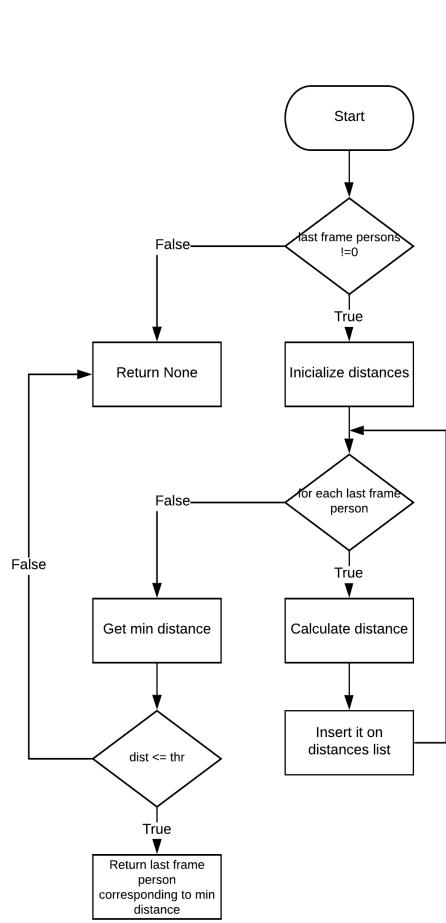
- Centroide mais próximo;
- Indice de Jaccard(IoU);

No caso do projeto além de se identificar as pessoas ao longo do vídeo, também se pretende registar as entradas e as saídas, através de uma linha de contagem. Todo este procedimento será explicado em detalhe posteriormente.

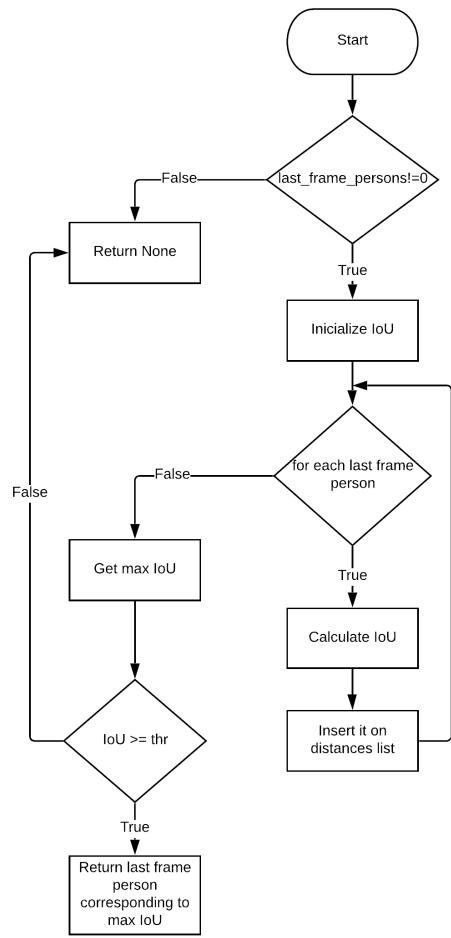
0.4.3 Algoritmos de Tracking

Relativamente ao algoritmo do centroide mais próximo, como se pode observar pelo fluxo representado na Figura 22a, é descrito pelos seguintes passos:

1. Calcular todas as distâncias do centroide da pessoa a identificar com todos os centroides das pessoas na última frame;
2. Obter o menor valor de todas as distâncias calculadas;
3. Se o valor for inferior a um certo limite, devolve-se a pessoa associada a esse centroide;
4. Se esse valor estiver acima do limite definido devolve-se None e é atribuído um novo identificador à pessoa em questão;



(a) Algoritmo do centroide mais próximo



(b) Algoritmo Intersection over Union

Em relação ao outro algoritmo de tracking "Intersection over Union" ou "Jaccard Indice" é uma métrica de avaliação usada para medir a precisão(ground-truth) de um detetor de objetos num conjunto de dados particular, ou seja , mesmo não sendo um algoritmo de tracking, o processo implementando consiste em comparar as "bounding-boxes" da pessoa a identificar com as das pessoas da frame anterior. O coeficiente de Jaccard consiste na divisão entre a intersecção e a união das duas "bounding-boxes". O raciocínio que foi descrito pode ser visualizado na Figura 22b e o seu procedimento é dado por:

1. Estimar o valor do IoU(Intersection over Union) da "bonding-box" da pessoa a identificar com todas as pessoas da frame anterior;
2. Obter o maior valor;
3. Se esse valor for maior que um certo limite, devolve-se a pessoa associada a essa "bounding-box";
4. Se esse valor estiver acima do limite definido devolve-se None e é atribuído um novo identificador à pessoa em questão;

0.5 Modelo de Dados

A informação gerada através da contagem/matching de pessoas pode ser uma ferramenta fundamental para estudos estatísticos e perceber alguns aspectos importantes, tais como: o tempo médio que uma pessoa passa no autocarro, quais são as paragens mais usadas e por exemplo calcular o melhor trajeto com tal de satisfazer as necessidades das pessoas. Toda a informação que é produzida no nosso algoritmo de Tracking é introduzida numa base de dados local (relacional), que posteriormente será reencaminhada para uma cloud alocada nos servidores da Amazon. Todo o processo de inserção na base de dados será facilitada através de uma api desenvolvida pelos elementos do grupo através da biblioteca MySQLdb, um modulo que simplifica o acesso a base de dados ODBC(banco de dados). Esta base de dados suporta diversos tipos de informação:

- Identificador da pessoa ao entrar;
- Identificador da pessoa ao sair;
- Hora de entrada;
- Hora de saída;
- Características de entrada;
- Características de saída;

Para além dos mencionados ainda existe uma coluna com o caminho para o vídeo gravado, para no futuro este também ser guardado nos servidores da Amazon.

A criação de uma base de dados relacional é dada pelas seguintes etapas:

1. Modelo Entidade-Associação
2. Modelo Relacional
3. Criação de tabelas(MariaDB)
4. Drop das tabelas(MariaDB)
5. Criação de vistas(MariaDB)*opcional

0.5.1 Modelo Entidade - Associação

Como se pode verificar na Figura 23, existem três entidades: Rota, Utilizador e Característica. Na entidade *Rota* temos os atributos *idRota* e *videoPath* para distinguir as várias rotas e guardar o caminho do vídeo respetivamente. A entidade *Utilizador* conta com quatro atributos capazes de identificar o matching de Id's (identificador e hora entrada/saída). Finalmente a entidade *Característica* para armazenar todo o tipo de características que são extraídas aos utilizadores quer à entrada ou saída do autocarro. É de referir que a entidade *Utilizador* é um entidade fraca de *Rota* porque só faz sentido existir utilizadores quando é realizada uma rota, o mesmo se aplica na entidade *Característica* em que só faz sentido haver características quando existem utilizadores.

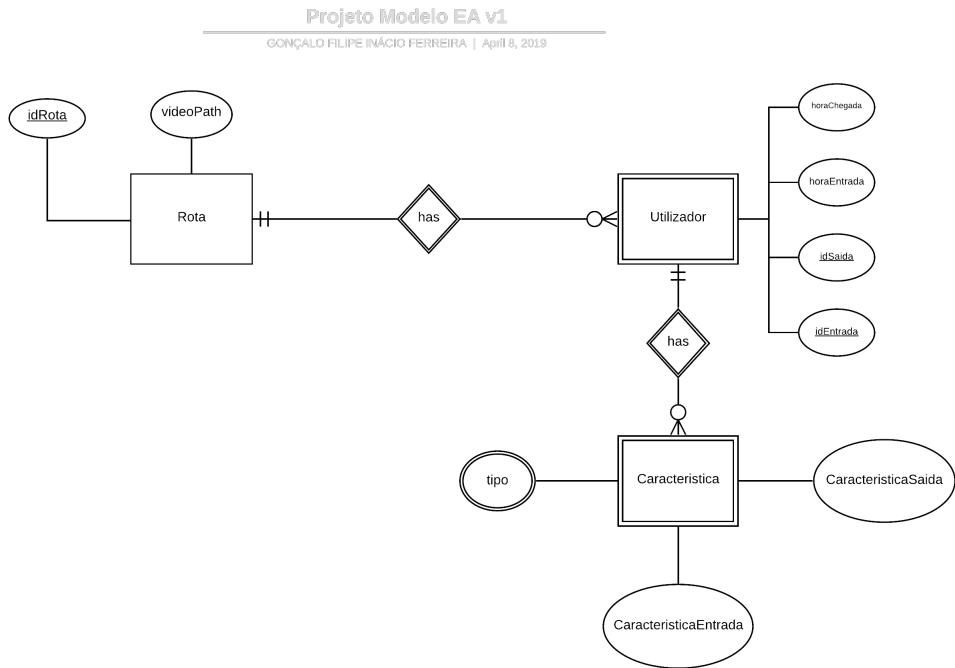


Figura 23: Modelo Entidade-Associação

0.5.2 Modelo Relacional

A criação do Modelo Relacional teve origem na aplicação de um método sistemático de passagem do Modelo Entidade-Associação para esquemas de relação, em que se verifica se os esquemas se encontram na forma normal desejada e caso não estejam, normalizar até à forma normal desejada. Assim sendo temos as seguintes regras:

- Cada Entidade representa um Esquema de Relação com os mesmos atributos e chaves;
- As associações podem ser descritas por um Esquema de relação próprio, tendo como chave a junção das chaves dos esquemas associados;
- Otimizações;

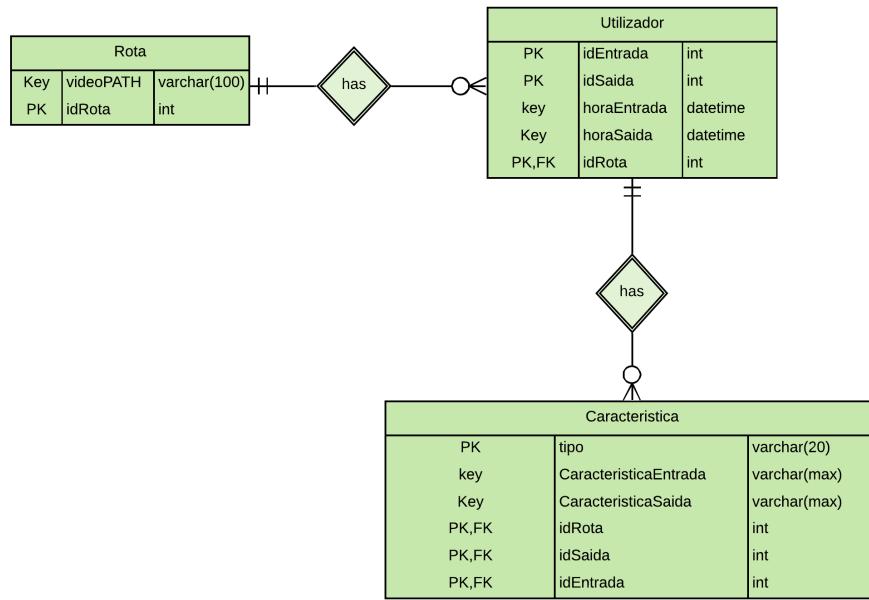


Figura 24: Modelo Relacional

Cada coluna representada nas tabelas da Figura 24 corresponde respetivamente a: chave, campo e tipo da variável. As entidades fracas são modeladas com um Esquema Relacional que conterá as suas chaves e a chaves primárias do Esquema Relacional de qual depende. Esta regra é aplicada à Entidade *Utilizador* e *Característica*, no qual como podemos observar na Figura 24 a chave *idRota* é uma chave primária e estrangeira na Entidade *Utilizador* e as chaves *idRota.idEntrada* e *idSaida* são chaves primárias e estrangeiras da Entidade *Característica*. Ainda assim as entidades fracas podem conter chaves parciais para distinguir qualquer tipo de colisão , no caso do EsqRel anteriormente referido a chave *tipo* é um atributo multi valor que representa os vários tipos de características extraídas (área,histograma,altura,etc..).

0.5.3 Diagrama de classes-Base de dados/FMS

De modo a fazer a ligação entre os algoritmos implementados e as estruturas de dados a utilizar, foram desenvolvidas duas API's (Figura 25) de forma a comunicar com a base de dados e o sistema de gestão de ficheiros, tanto remotamente como localmente.

A classe *Database* tem todos os métodos capazes de cumprir com todas as funcionalidades para o bom funcionamento da base de dados, além disso a partir do seu construtor é possível estabelecer conexão com uma base de dados relacional. Para todo o processo de ligação aos serviços da Amazon implemento-se a classe "AmazonAPI", preparada para ligar a um dos dois serviços(S3 e Rds) e também para o envio de um ficheiro, no nosso caso videos com as caras ofuscadas, para a estrutura de dados. É de referir, que mesmo fazendo a conexão com a base de dados remota, usa-se ainda assim a classe *Database* para todo o processo ligado à comunicação entre os módulos.

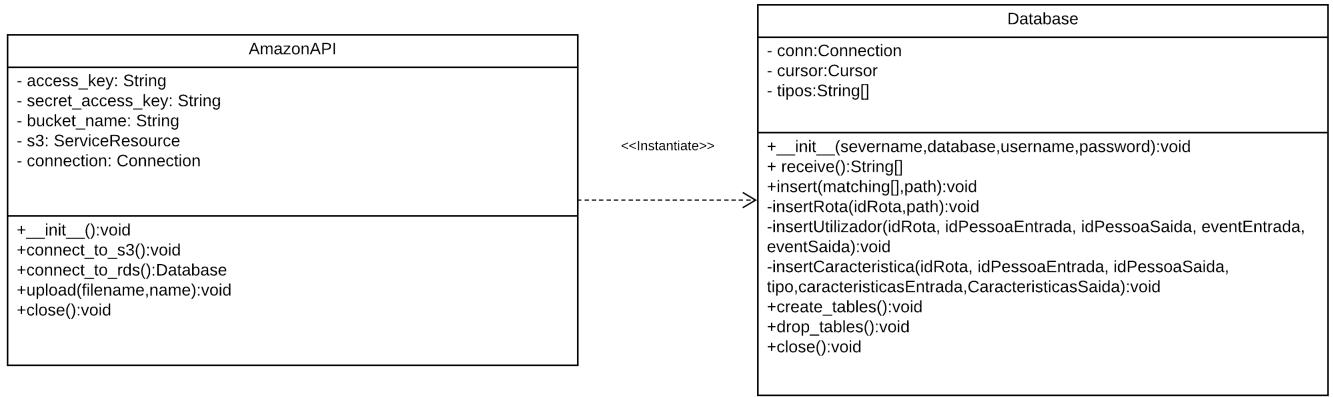


Figura 25: Diagrama de Classes

A utilização destas duas classes será feita pela *MainApp* que como foi referido anteriormente, tem como objetivo "juntar o puzzle" de modo a sincronizar todos os algoritmos desenvolvidos. A utilização de ditas bibliotecas é bastante simples no caso de não haver internet ou a ligação não puder ser estabelecida basta com instanciar a classe *Database* e inserir os resultados da correspondência apenas usando o método *insert*. Todo o processo de inserção será tratado por parte da classe sem que o programador tenha que fazer mais alguma coisa. No caso de haver internet e pretenda-se enviar informações obtidas e vídeos gravados para a cloud basta estabelecer ligação com os dois serviços através dos métodos *connect_to_s3* e *connect_to_rds* e fazer o upload do vídeo após ofuscado.

0.6 Extração de Características

De maneira a corresponder as pessoas que entram e saíam de um local é essencial definir aspectos que permitem discriminar pessoas diferentes, ou seja, características que possibilitam distinguir pessoas diferentes. Um ser humano consegue facilmente diferenciar pessoas a partir de traços como a face, cabelo, altura, composição e postura, no entanto para uma máquina torna-se este é um processo bastante complicado.

Tendo em consideração que o desafio deste projeto é a correspondência de pessoas em diferentes direções em relação à câmaras é preciso ter em consideração o facto que existem fatores que alteram certas características. Estes fatores podem ser mudanças na luminosidade ao longo do dia, a oclusão das pessoas quando em grupo, a mudança de cor e até o facto de a resolução das câmaras de vigilância não ser as mais ideais.

0.6.1 Características da Região

Na primeira abordagem identificamos que poderíamos utilizar a cor como elemento identificativo de diferentes pessoas, visto que normalmente a cor da roupa das pessoas costuma ser suficientemente distinta para poder identificar como uma pessoa diferente. Para tal considerámos os histogramas de cor úteis para diferenciar, bem como para corresponder pessoas, no entanto é preciso ter em conta qual o modelo de cor utilizado. No processamento de imagem o espaço de cor mais utilizado é o HSV, que corresponde à matiz, saturação e brilho de uma imagem, porque este separa a luminosidade da crominância o que é útil visto que separa a cor do brilho que varia regularmente e remove as sombras que possam ocorrer na imagem.

Quando mencionamos a extração de características de uma pessoa, não podemos afirmar que estas características são explicitamente da pessoa, ou seja, a imagem utilizada para extrair esta informação não corresponde à pessoa em si, à sua região, mas corresponde à sua região em conjunto com o fundo. O método deteção devolve-nos a bounding box da pessoa, esta é utilizada para retirar quaisquer características da imagem delimitada por esta, sendo que o ideal seria a extração apenas na região correspondente à pessoa.

Histogramas de cor HSV

A primeira característica extraída das pessoas foi os histogramas uni-dimensionais de Hue, os resultados desta extração nas Figs.26a e 26b onde é representado em cada gráfico os histogramas de Hue das entradas e saídas de cada pessoa num vídeo de teste. Nestes gráficos, considerando cada identidade de pessoa como classes distintas, é possível constatar uma ligeira variância intra-classe (mesma pessoa) e uma maior variância inter-classe (diferentes pessoas). Tendo em consideração que os valores de Hue estão entre 0-180 e os valores de contagem foram normalizados para representar um função densidade probabilidade.

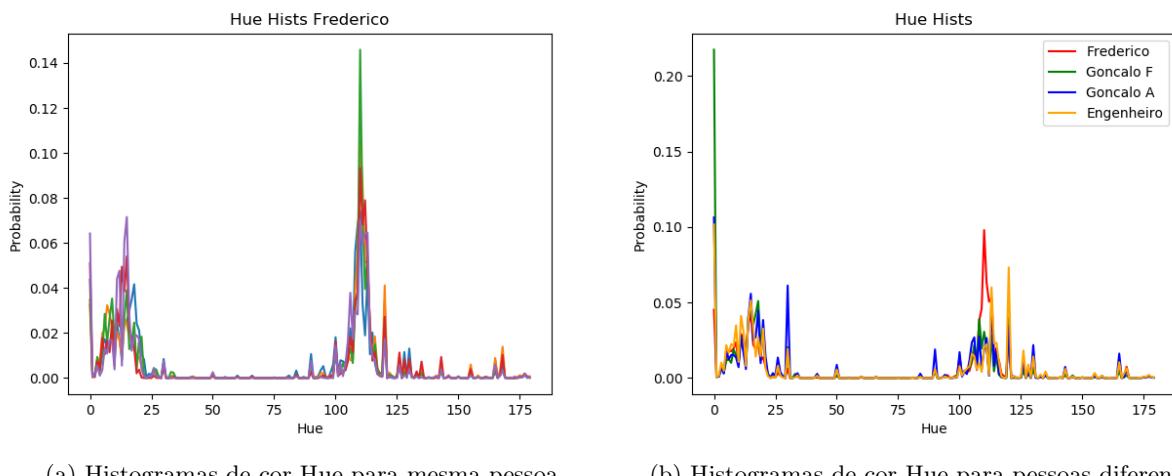


Figura 26: Histogramas de cor Hue

O histograma de saturação das pessoas foi outra característica que tivemos em consideração, nestes histogramas das Figs.27a e 27b verificou-se em relação aos histogramas de cor Hue que a variância entre pessoas diferentes era mais acentuada. No caso dos histogramas de Hue existem duas pessoas (Gonçalo F, Engenheiro André) em que a sua variância pode ser considerada semelhante, enquanto nos histogramas de saturação os seus valores normalizados revelam uma maior diferença, tendo valores máximos (picos do histogramas) mais distintos.

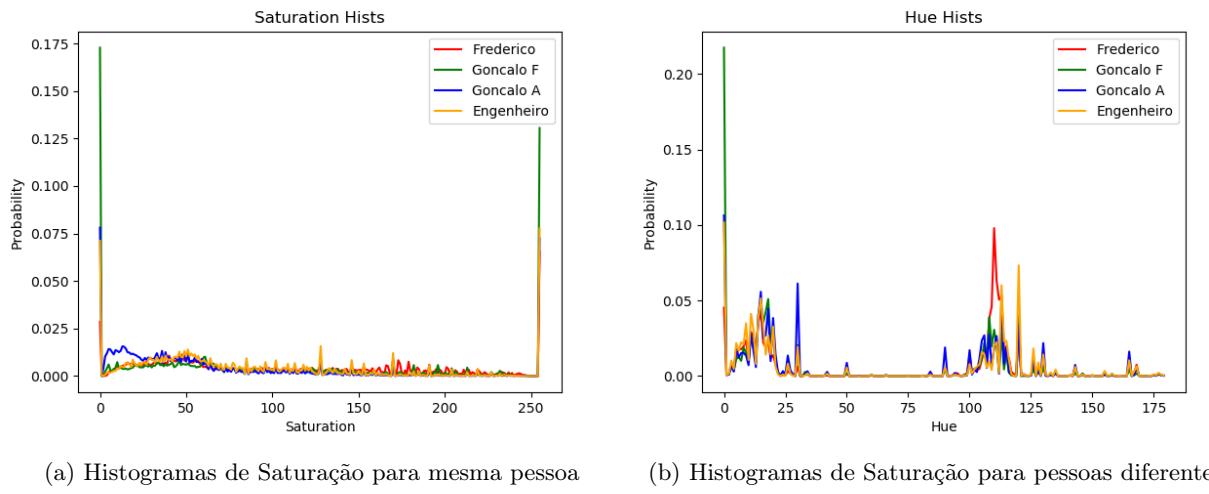


Figura 27: Histogramas de Saturação

Tivemos em consideração o facto de que estes histogramas poderiam ser combinados num histograma bidimensional HS (Hue Saturation), utilizando dois planos de cor em vez de apenas um. Os resultados desta característica foram utilizados em testes preliminares e verificou-se que o desempenho na correspondência das pessoas era menos satisfatório quando usado individualmente os histogramas de Matiz e de Saturação, portanto este não foi considerado para a versão "final" das características extraídas.

A área das bounding boxes das pessoas também foi outra característica considerada, em que no vídeo de teste que utilizámos observou-se que a área das pessoas nos instantes de entrada ou saída, os valores para a mesma pessoa ainda tinha uma certa variação (pouco constante), no entanto para pessoas diferentes as suas áreas ainda divergiam de uma forma significativa. As Figs.28a e 28b demonstram em cada instante em que uma pessoa entra ou sai, a área capturada pela bounding box, nestes gráficos consegue-se observar a diferença entre pessoas anteriormente referida.

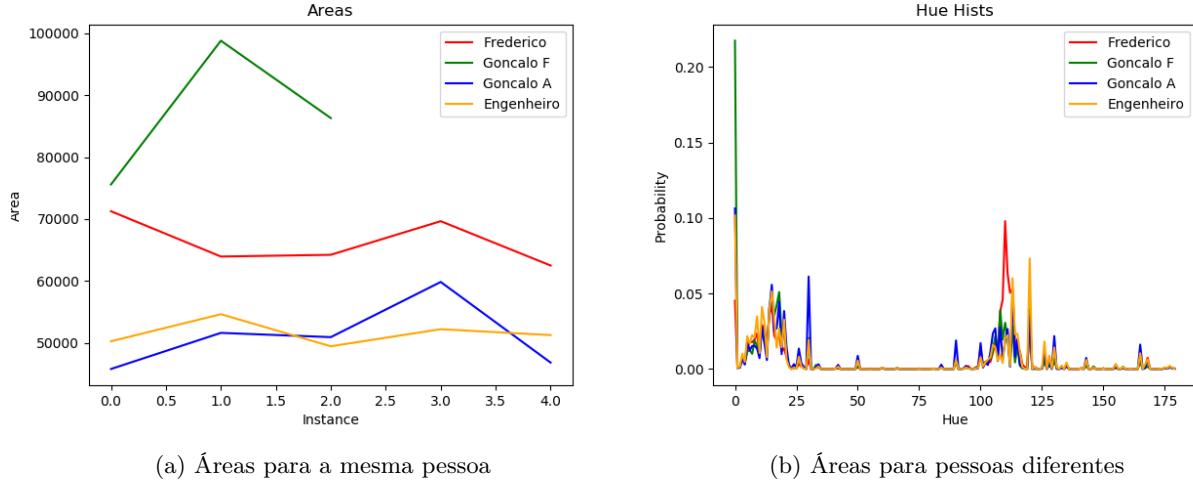


Figura 28: Áreas das pessoas em instantes diferentes

Descriptor estruturante de cor

O descriptor estruturante de cor é baseado em histogramas de cor, no entanto, tem como objetivo proporcionar uma descrição mais precisa ao identificar localizações de distribuição de cor utilizando uma janela de pesquisa(8x8). O espaço de cor utilizado neste descriptor especificamente é o HMMD(Hue-Min-Max-Difference), após a conversão de RGB para HMMD existe um total de 5 componentes identificados no espaço de cor, os 4 presentes no nome HMMD mais um componente definido pela soma entre o máximo e o mínimo na conversão de RGB para HMMD a dividir por 2. Como foi dito anteriormente, o CSD é calculado através de várias janelas de pesquisa(8x8 pixels) ao longo da imagem (Figura 29), em que para cada pixel é incrementado o contador(bin) associado as cores C{0-7}(no caso do projeto usaram-se 64).

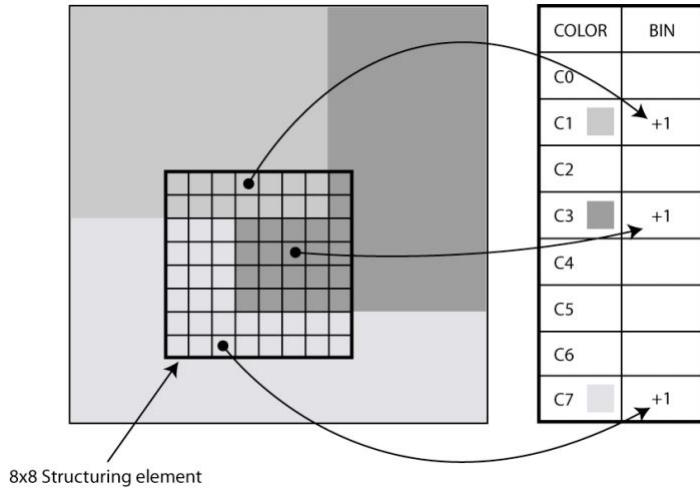


Figura 29: Elemento estruturante do CSD
[12]

Como foi dito anteriormente o espaço de cores usado no descriptor estruturante de cor é o HMMD, este é definido usando 4 pontos de quantificação: 256, 128, 64 e 32 bins. A quantificação, como se pode observar pela Figura 30, é feita nos seguintes passos:

1. O espaço de cor HMMD é dividido em 5 sub espaços(0-4).

2. A divisão dos sub espaços é dependente do valor diff no HMMD resultando nos seguintes intervalos: [0,6],[6-20],[20,60],[60,110] e [110,255].
3. Posteriormente cada sub espaço é quantificado uniformemente ao longo das componentes *Hue* e *Sum*.
4. Finalmente os valores obtidos os valores que contem o histograma de cor estruturante são normalizados.

Component	Subspace	Number of quantization levels for different numbers of histogramm bins			
		256	128	64	32
Hue	0	1	1	1	1
	1	4	4		
	2	16	8	4	4
	3	16	8	8	4
	4				
Sum	0	32	16	8	8
	1	8		4	4
	2	4		4	
	3	4	4	2	
	4			1	1

Figura 30: Quantificação dos espaços de cor HMMD
[12]

Na representação gráfica nas Figs.31a e 31b deste descriptor foi possível notar que esta característica demonstrava uma maior semelhança entre as mesmas pessoas, no caso deste teste este facto também se observou entre pessoas diferentes mas não de uma forma tão acentuada.

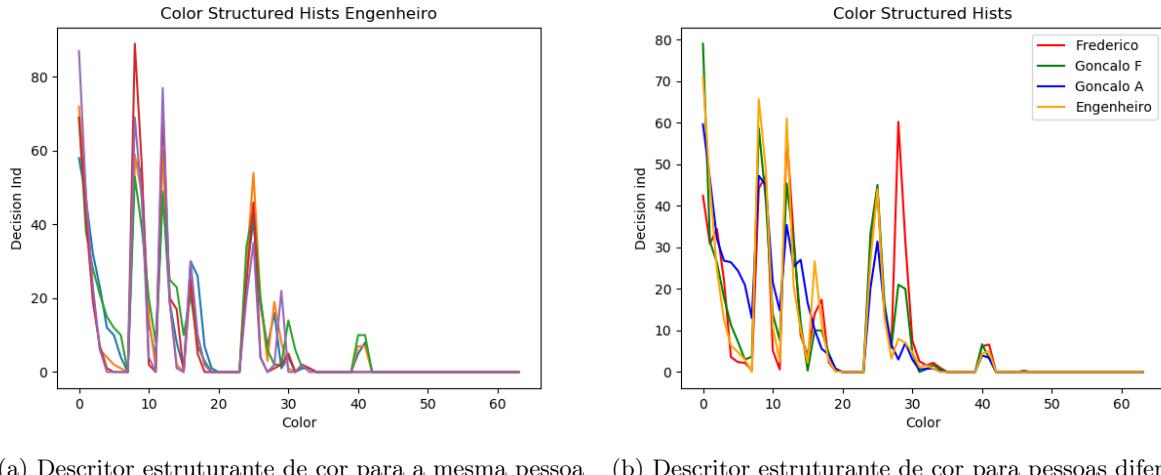


Figura 31: Descritores estruturantes de cor

Descriptor do histograma de contornos

O descriptor do histograma de contornos(EHD) é um dos métodos mais usados na deteção de formas. Este descriptor representa a frequência relativa para cada um dos 5 tipos de contornos em cada local, designado por bloco. Este bloco(também designado por sub-imagem) é definido pela partição da imagem em blocos de 4x4 não sobrepostos entre eles, querendo isto dizer que a imagem será dividida em 16 blocos iguais independentemente do tamanho da imagem. Para definir as características do bloco, são gerados histogramas da distribuição de contornos para cada bloco. Como se pode observar na Figura 32, os contornos no bloco(sub-imagem) são classificados em 5 tipos: vertical, horizontal, 45-graus diagonal, 135-graus diagonal e

contorno sem direção. Resumidamente, o histograma para cada bloco representa a distribuição relativa dos 5 tipos de contornos na sub-imagem correspondente.

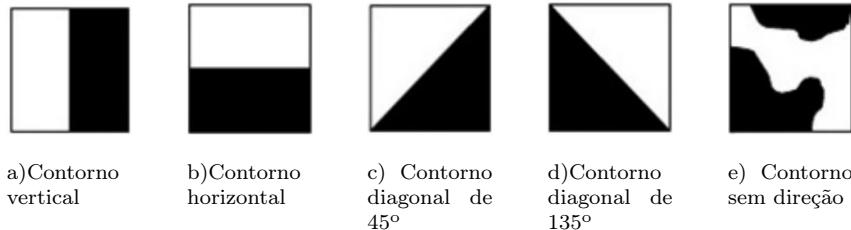


Figura 32: Tipos de contornos no EHD

[10]

Visto que a imagem é composta por 16 blocos de semelhantes dimensões e existem 5 tipos de contornos possíveis, são necessários $5 * 16 = 80$ histogramas, Figura 33. É de referir que cada histograma representa uma semântica diferente sendo que são caracterizados por local(localização na imagem) e tipo de contorno. A posição de cada bloco é dada por (x,y) sendo que o primeiro bloco começa no $(0,0)$ e o ultimo no $(3,3)$.

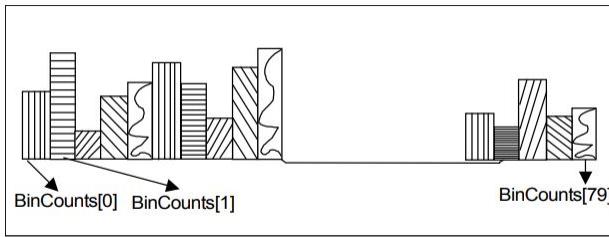


Figura 33: Histogramas do EHD

[10]

Finalmente os valores normalizados do histogramas são quantificados em representação binária. Esta quantificação é feita não linearmente devido a que a gama em que os valores estão situados é de pequena dimensão. A tabela 2 representa o valor binário para cada uma das quantificações de cada tipo de contorno.

Representação binária (3bits/bin)	Valor representativo para contornos verticais	Valor representativo para contornos horizontais	Valor representativo para contornos diagonais 45°	Valor representativo para contornos diagonais 135°	Valor representativo para contornos sem direção
000	0.010867	0.012266	0.004193	0.004174	0.006778
001	0.057915	0.069934	0.025852	0.025924	0.051667
010	0.099526	0.125879	0.046860	0.046232	0.108650
011	0.144849	0.182307	0.068519	0.067163	0.166257
100	0.195573	0.243396	0.093286	0.089655	0.224226
101	0.260504	0.314563	0.123349	0.115391	0.285691
110	0.358031	0.411728	0.161505	0.151904	0.356375
111	0.530128	0.564319	0.228960	0.217745	0.450972

Tabela 2: Tabela de quantificação para os 5 tipos de contorno

[10]

A partir da Fig.34 chega-se rapidamente à conclusão que esta característica não é a mais indicada para procedemos à correspondência das pessoas, uma vez que a variância para a mesma pessoa é demasiado dispersa, não sendo possível identificar semelhanças algumas entre descriptores da mesma pessoa.

0.6.2 Características da Rede Neuronal

Sendo que um dos aspetos das redes neuronais convolucionais é o facto de existirem camadas exclusivamente para a extração de características das imagens, através da rede MobileNets + SSD e utilizando o módulo de

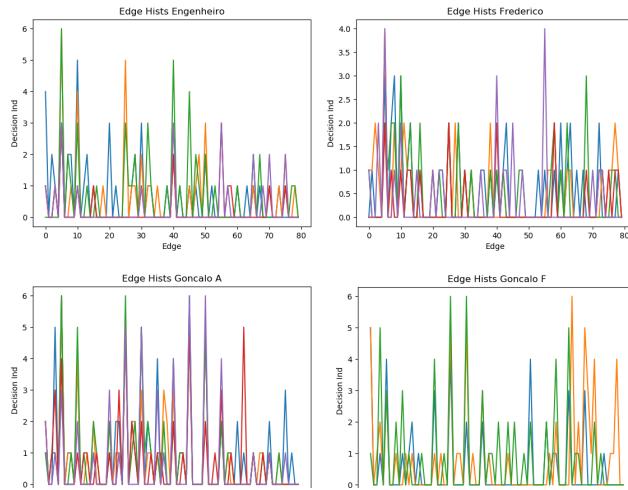


Figura 34: Histogramas de contornos

redes neurais "dnn" do OpenCV foi possível obtermos o output de cada camada desta rede. A rede MobileNet é utilizada como extratora de características que depois seriam utilizadas pelo detetor SSD MultiBox, logo a lógica foi extraer o output da última camada do MobileNet antes de passar para a classificação e deteção.

Com a intenção de verificar se as características resultantes iriam ser úteis na correspondência de pessoas foram armazenadas as imagens das pessoas nas entradas e saídas, estas imagens iriam passar pela rede até uma certa camada(parâmetro controlável) e o output resultante foi representado num gráfico para identificar se estes eram discerníveis entre as diferentes pessoas. Na Fig.35 é possível observar que para cada pessoa das entradas os outputs têm uma distribuição semelhante entre estes, esta semelhança pode ser justificada devido ao facto que a rede que utilizamos está treinada para identificar 20 classes diferentes, logo o output para uma pessoa seria parecido de modo a distingui-los de outros objetos que possam ser detetados. Através destes testes pudemos concluir que as características extraídas desta rede não seriam úteis como métrica para diferenciar as pessoas detetadas, portanto estas características não foram utilizadas na versão final do sistema de matching.

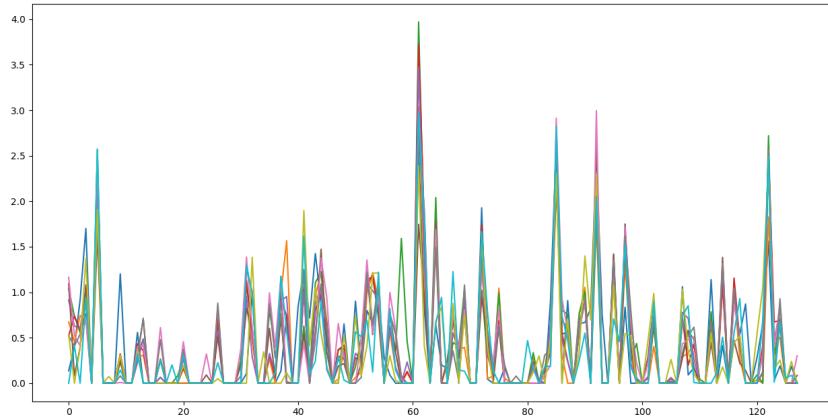


Figura 35: Output da última camada da rede MobileNet

Após realizado a extração destas características e observar a representação gráfica dos resultados seja para

a mesma pessoa seja para diferentes pessoas, chegámos à conclusão que as características que seriam mais úteis na correspondência das pessoas seriam os histogramas de cor Hue, Saturação, as áreas das pessoas e os histogramas estruturantes de cor.

0.7 Algoritmo de correspondência

O algoritmo de correspondência é um dos módulos prioritários do sistema desenvolvido. Durante a análise de requisitos do projeto, foi definido a correspondência de pessoas como um dos principais objetivos. O problema de correspondência pode ser observado como um grafo (figura 36), no qual os seus nós são as pessoas detetadas e as suas adjacências são caracterizadas por um valor de semelhança. O objetivo é corresponder as pessoas que entraram com as pessoas que saíram, de modo a minimizar a soma dos valores das adjacências (pensamento lógico de custo).

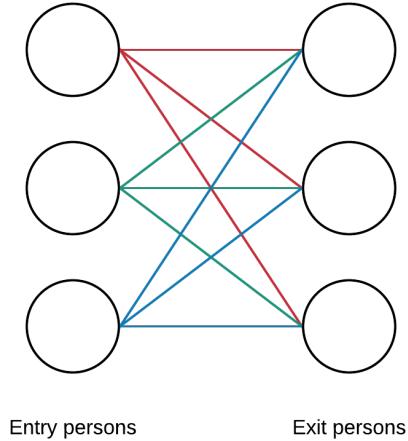


Figura 36: Grafo de correspondências

O algoritmo de correspondência, dados os casos de utilização, não impõe restrições quanto à sua utilização em tempo real. No entanto, o sistema de contagem incorpora mecanismos para que essa opção também seja possível, caso se pretenda observar as correspondências em tempo real. Sem essa restrição, o algoritmo é aplicado no final do processo de contagem. Nessa fase do processo, já são conhecidas as entradas e saídas detetadas.

A figura 37 ilustra as entradas e saídas do algoritmo de correspondência, assim como a sua organização em termos de software em uma fase de análise. Tal como se pode observar, recebe o conjunto de pessoas detetadas a entrar e a sair e devolve uma estrutura de dados que permite o armazenamento de informação das correspondências atribuídas. Os resultados são armazenados sob a forma de uma lista que apresenta como elementos três outras listas:

1. Primeira lista: conjunto de características utilizadas na correspondência;
2. Segunda lista: pessoas que entraram;
3. Terceira lista: pessoas que saíram.

As entidades *Person*, contidas na segunda e terceira lista, vêm ordenadas de acordo com os resultados das correspondências, sendo que uma pessoa cujo resultado não tenha uma correspondência atribuída vem com o valor *None* na lista contrária (na mesma posição).

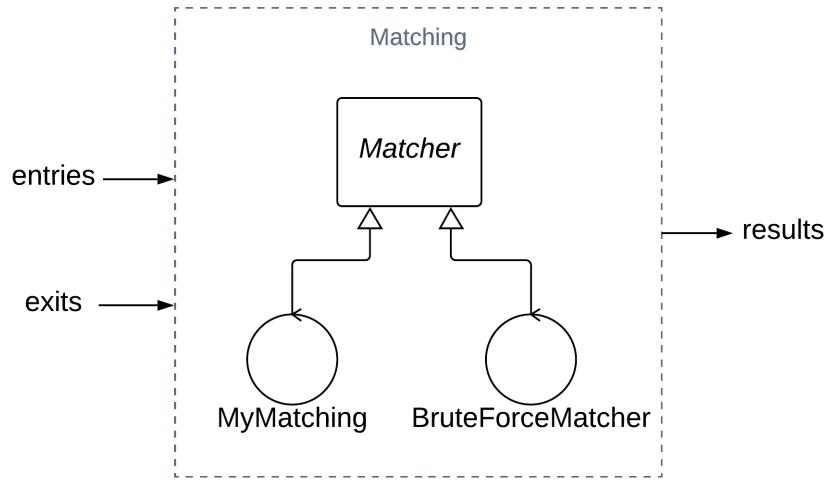


Figura 37: Algoritmo de correspondência

0.7.1 Primeira Abordagem

Uma das primeiras fases do desenvolvimento de um sistema é a definição da arquitetura inerente. Dada a arquitetura, foi optado por primeiramente fundamentar a cobertura do sistema e em uma fase posterior a sua precisão. Esta seção destina-se ao primeiro algoritmo desenvolvido para a correspondência de pessoas, sendo que na fase de desenvolvimento deste algoritmo cada pessoa apenas era caracterizada por um histograma de cor. Deste modo, e fazendo a ligação ao grafo representado na figura 36, cada nó (pessoa) é representado por um histograma de cor e o valor da semelhança entre pessoas é dado pelo valor da medida *match* entre os seus histogramas (equação 4).

$$match = \frac{\sum \min(hist1, hist2)}{\sum hist2} \quad (4)$$

Os passos do algoritmo estão representados sob a forma de fluxo, através do diagrama na figura 38. Fazendo a leitura do diagrama, percorre-se cada pessoa que conste na lista de pessoas de saída e segue-se os seguintes passos:

1. obtém-se as correspondências válidas: pessoas que entraram antes da hora de saída da pessoa a percorrer;
2. calcula-se o valor de *histogram match* entre o histograma da pessoa a percorrer e o histograma das pessoas válidas;
3. obtém-se a pessoa inerente ao valor mais elevado;
4. se a pessoa inerente não tiver sido correspondida anteriormente, insere-se a correspondência à estrutura dados de resultados. Caso contrário, inicia-se o processo que evita colisões de correspondências.

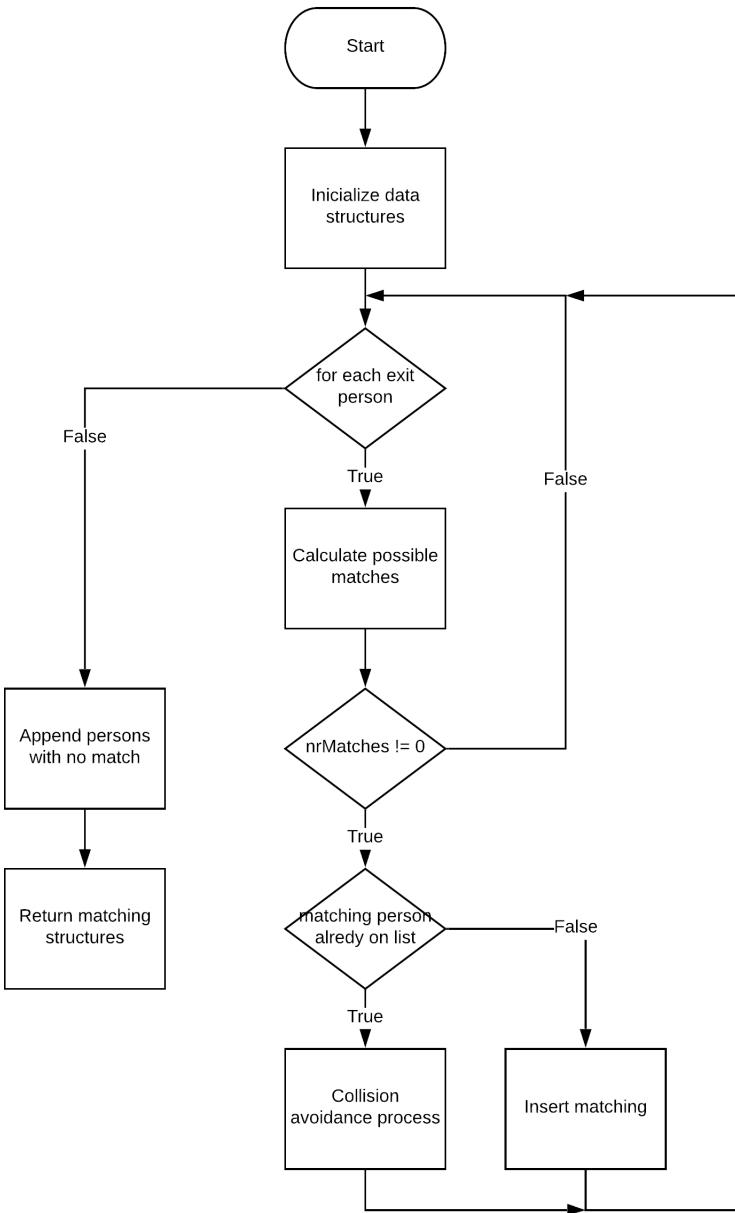


Figura 38: Fluxo MyMatching

Se duas pessoas que saíram, tiverem como melhor correspondência a mesma pessoa que entrou, está-se perante uma situação de colisão. O processo *Collision Avoidance*, tal como o nome o sugere, destina-se a evitar que duas pessoas sejam correspondidas a uma mesma pessoa. A solução implementada baseia-se na verificação das correspondências alternativas e avaliação da conjugação que providencie menor custo. A figura 39 contempla os vários cenários possíveis de colisão:

- ambas as pessoas não têm correspondências alternativas (figura 39a): a pessoa 3 é correspondida à pessoa cujo valor da adjacência é superior (pessoa 1). A pessoa 2 irá ficar sem correspondência;
- ambas as pessoas apresentam correspondências alternativas (figura 39b): avaliação da conjugação de correspondências que originam maior ganho. Tal como se pode verificar pelos cálculos, o ganho é

superior caso *person 1* tenha como correspondência *person 4*, apesar do valor da adjacência com *person 3* ser superior. Neste exemplo, as correspondências seriam P1-P4 e P2-P3.

Person 1 assume a alternativa: $P1-P4 (0.7) + P2-P3 (0.6) = 1.3$
 Person 2 assume a alternativa: $P2-P4 (0.1) + P1-P3 (0.8) = 0.9$

- apenas uma das pessoas tem correspondências alternativas (figura 39c): a pessoa que apresenta correspondências alternativas fica com essa associação, de modo a que os resultados apresentem o menor número de pessoas não correspondidas. Neste cenário, resultaria as correspondências P1-P3 e P2-P4 apesar do valor da adjacência de *Person 2* com *Person 3* ser superior.

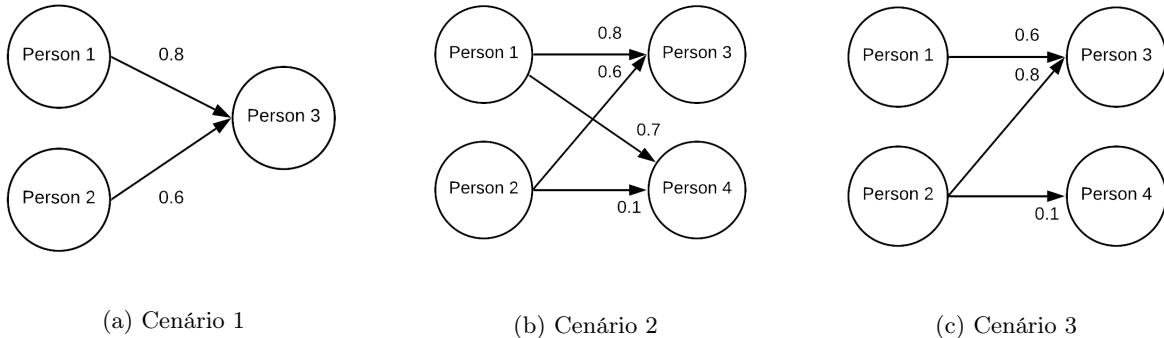


Figura 39: Cenários de colisão

Apesar de não ter sido possível devido a requisitos temporais, o grupo de trabalho tencionava realizar como próximos objetivos:

1. Generalizar o algoritmo para que cada pessoa seja representada por um vetor de características;
2. As adjacências, dado o primeiro passo, seriam caracterizadas pela diferença entre vetores ou outra medida de distância entre vetores;
3. Introduzir o conceito de recursividade, dado que a obtenção de alternativa de correspondência também pode originar colisão;
4. Comparar o desempenho com o algoritmo *Brute Force*.

0.7.2 Algoritmo Brute Force

A primeira abordagem utiliza procura exaustiva, de modo a determinar a pessoa que tem maior valor *histogram match* em relação a uma outra pessoa. A abordagem do algoritmo a descrever nesta secção segue a mesma metodologia, generalizando o facto de cada pessoa ser descrita por um vetor de características. Para tal, tira partido de objetos de correspondência entre vetores, disponibilizados pela biblioteca *OpenCV*, como por exemplo, o objeto *BFMatcher*. Em uma primeira instância, o algoritmo era apenas descrito pelos seguintes passos:

1. obter próxima pessoa na lista de pessoas de saída;
2. utilizar o objeto de correspondências entre vetores, de modo a verificar as distâncias absolutas para com as pessoas na lista de entradas;
3. a correspondência da pessoa de saída trata-se da primeira pessoa que entrou, cuja correspondência é válida, ou seja, caso a sua hora de entrada seja menor que a da saída e se não tiver sido correspondida anteriormente.

Posteriormente, o mecanismo de correspondência foi adaptado para uma algoritmo iterativo. As instruções, enumeradas anteriormente, são executadas n iterações. Para cada iteração, são armazenados os seus resultados assim como o custo associado (soma de todas as distâncias entre vetores das correspondências). Para que, logicamente, os resultados entre iterações não sejam constantes, a ordem de processamento das pessoas detetadas a sair é gerada de forma aleatória. O diagrama de fluxo ilustrado na figura 40 representa o fluxo associado a cada iteração. No final do processamento das n iterações, são retornados os resultados associados à iteração que obteve menor custo.

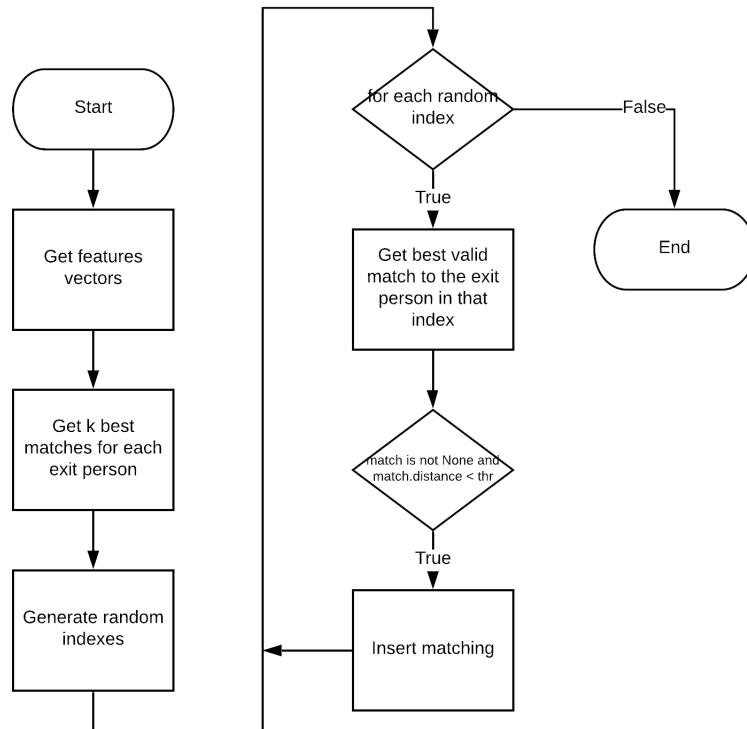


Figura 40: Iteração algoritmo BruteForce

O objetivo desta abordagem trata-se de introduzir alguma robustez a possíveis erros de correspondências que possam surgir. No entanto, também originou outros cenários no qual interferem com a performance do algoritmo tal como o exemplo ilustrado na figura 41. Como a lista de pessoas detetadas a sair passa a não ser percorrida ordenadamente, é possível que uma pessoa obtenha uma correspondência correta do ponto vista da pessoa mas incorreta do ponto vista da instância de pessoa (temporalmente incorreta), prejudicando outra correspondência.

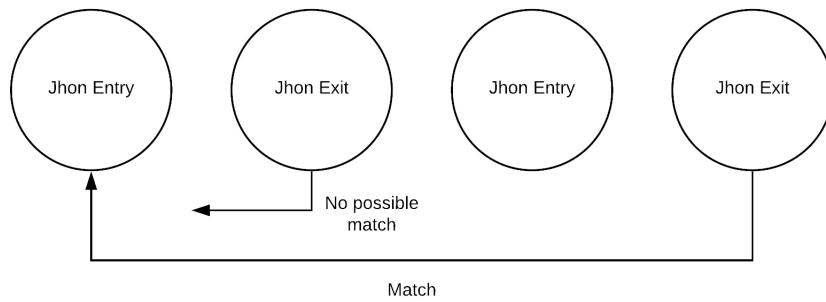


Figura 41: Problema algoritmo iterativo

Para atenuar o problema descrito, o custo de cada iteração deixou de ser apenas a soma das distâncias entre vetores. A distância de cada correspondência passou também a ser pesada pela diferença temporal entre as duas pessoas que compõem o match (equação 5). O valor α é o peso da diferença temporal enquanto que, por outro lado, $1 - \alpha$ o peso da distância vetorial. Desta forma, pessoas correspondidas entre intervalos de tempo distantes iriam aumentar o valor do custo da iteração. A utilização deste parâmetro requer algum estudo da correlação temporal entre correspondências, no qual seria um tópico futuro a explorar pelo grupo.

$$d = (1 - \alpha)vectorDistance + \alpha temporalDiff \quad (5)$$

0.7.3 Implementação

Um diagrama visa apresentar informação de uma dada perspetiva do sistema. Nas secções relativas aos algoritmos, foram ilustrados diagramas quanto ao seu fluxo (comportamento). A vista representada na figura 42 contém informação quanto à sua estrutura de implementação.

A classe *Matcher* é uma classe abstrata que contém um conjunto de métodos comuns a ambos os algoritmos tais como:

- **set_person_storage:** deve ser chamado antes de ser aplicado o algoritmo. Define o conjunto de pessoas a ser correspondidas;
- **insert_matching:** permite inserir uma correspondência à estrutura de dados;
- **write_matches:** permite a escrita dos resultados em uma dada diretoria;
- **print_results:** permite a visualização dos resultados e algumas medidas associadas;
- **apply_algorithm:** método abstrato, destinado à aplicação do respetivo algoritmo de correspondência produzindo um conjunto de resultados, com a estrutura definida na secção 0.7 na explicação do módulo de correspondência;
- entre outros.

Os objetos *MyMatching* e *BruteForceMatcher* implementam os algoritmos descritos anteriormente. Dependendo do algoritmo, tendo em conta a sua sequência de atividades, são necessários um conjunto de mecanismos que se podem visualizar, em parte, pelo seu conjunto de métodos. Note-se que o objeto *MyMatching* apresenta um método representativo do processo de colisão de correspondências. Por outro lado, para o algoritmo *BruteForce*, pode-se verificar que existe: geração índices aleatórios dada um determinada dimensão, cálculo do custo, processamento de uma iteração, verificação se um par de pessoas é uma correspondência válida, entre outros mecanismos necessários.

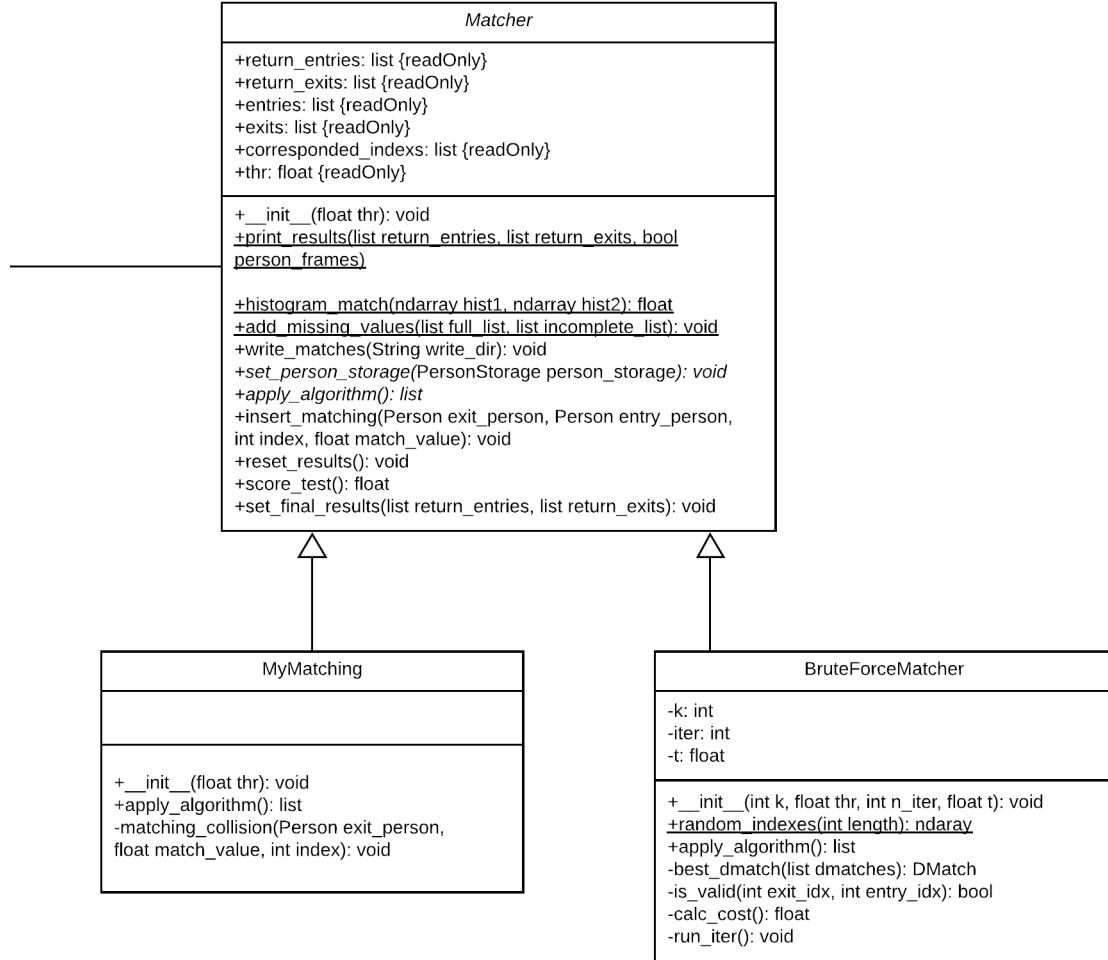


Figura 42: Diagrama de classes - Algoritmos de correspondência

Tendo em conta a secção anterior, para o processo de instanciação de um objeto para aplicação do algoritmo *BruteForce*, é necessário receber um conjunto de parâmetros. Os parâmetros k/thr definem a benevolência das correspondências significando, respetivamente, o número de matches a ser devolvido pelo *BFMatcher* e o valor máximo de distância vetorial para ser uma correspondência possível. Adicionalmente, o valor n define o número de iterações a processar e o valor t define o peso da diferença temporal no cálculo da distância de cada correspondência. No caso do algoritmo *MyMatching*, destaca-se o facto do valor limiar ter o pensamento inverso, uma vez que utiliza como medida de semelhança *histogram match*.

O código listado representa um exemplo de utilização do algoritmo *BruteForce*. Tal como se pode observar, pretende-se que não haja restrições ao nível do limiar de correspondência, não é aplicada o mecanismo de diferença temporal, o valor do k é de 20 e são pretendidas 100 iterações.

```

1 thr = sys.float_info.max
2 k = 20
3 n_iter = 100
4 t = 0.0

```

```

5 matcher = BruteForceMatcher(k, thr, n_iter, t)
6 matcher.set_person_storage(person_storage)
7 results = matcher.apply_algorithm()

```

0.7.4 Algoritmo em tempo real

O comportamento *TrackingSystem*, objeto que implementa o sistema de contagem, apresenta a possibilidade de parametrização de correspondência em tempo real. O mecanismo implementado não segue a estrutura apresentada na secção 0.7. Apenas possibilita a visualização da correspondência, caso fosse em tempo real, não sendo estas armazenadas. Os blocos inerentes estão representados na figura 43. Cada vez que uma pessoa é detetada a sair, são calculadas as distâncias entre o seu vetor de características e os vetores de características das pessoas detetadas a entrar até ao momento; obtem-se a pessoa que entrou inerente à menor distância e ilustra-se a sua *imagem* (figura 44). A imagem permanecerá visível e o sistema de contagem bloqueado até que o utilizador prima qualquer tecla. Esse evento, ao ser detetado, destrói a janela relativa à ilustração da correspondência e continua o ciclo de execução do sistema de contagem.

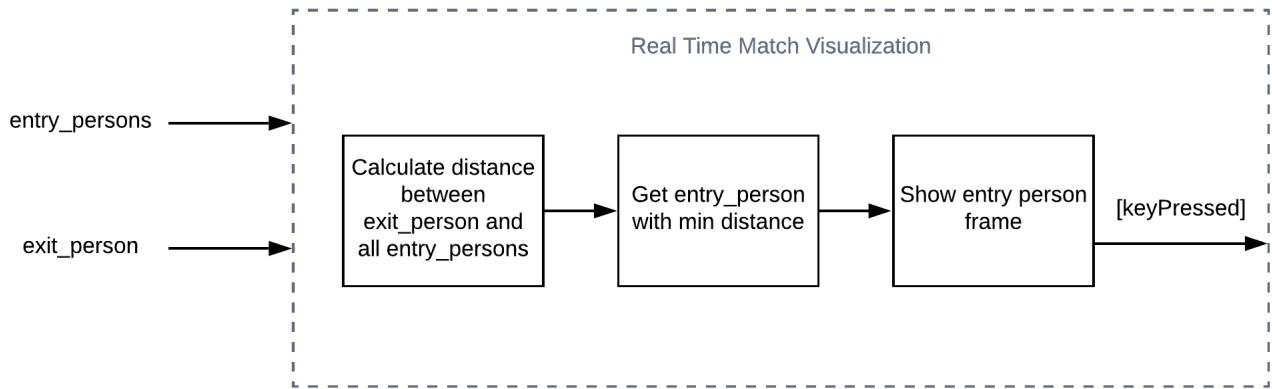


Figura 43: Diagrama de blocos - correspondência em tempo real



Figura 44: Correspondência em tempo real

0.7.5 Visualização de correspondências

A visualização de resultados é um bloco importante para a melhoria contínua dos algoritmos implementados, de modo a que a deteção de um problema seja um processo facilitado e rapidamente seja implementada uma solução. Numa fase inicial, os resultados das correspondências eram ilustrados na consola através dos ids das

pessoas correspondentes. Deste modo, não possibilita uma fácil verificação da veracidade da correspondência, e caso esta seja incorreta qual poderá ser a origem do erro.

Dados os motivos mencionados, foi adicionado um atributo à entidade *Person* com o objetivo de armazenar a imagem correspondente à pessoa. O valor deste atributo é atribuído no momento em que a pessoa transpõe a linha de contagem. Tal como já descrito em secções anteriores, os resultados do algoritmo utilizam instâncias desta mesma entidade. Como tal, é possível ilustrar cada correspondência através das imagens correspondentes às pessoas. As imagens, para cada pessoa, terão dimensões diferentes. É necessário aplicar uma interpolação a uma das imagens e aplicar a concatenação de *arrays*, de forma a que se possa ilustrar a correspondência em uma única janela.

Em termos de *software*, a classe *Matcher* (figura 42) fornece um conjunto de métodos úteis relativos a esta temática:

- **print_results**: permite a visualização das correspondências através das imagens concatenadas, utilizando a biblioteca *opencv*, assim como outras informações das características;
- **write_matches**: permite gravar as imagens concatenadas das correspondências em uma dada diretoria;
- **score_test**: implementado antes do mecanismo automático de *ground truth*. Permite obter o valor de percentagem de acertos para um vídeo previamente gravado para testes.

O gráfico, na figura 45, é ilustrado ao ser executada o método **print_results**. Tal como se pode verificar, representa graficamente um conjunto de informações adicionais: valor *histogram match* para cada histograma (cor, saturação, cor-saturação, cor estruturante, contornos) e diferença entre áreas. Estas informações, de certo modo, contemplam o estudo das características e possibilitam a verificação de quais as características no qual estão a ser prejudiciais no processo de correspondência.

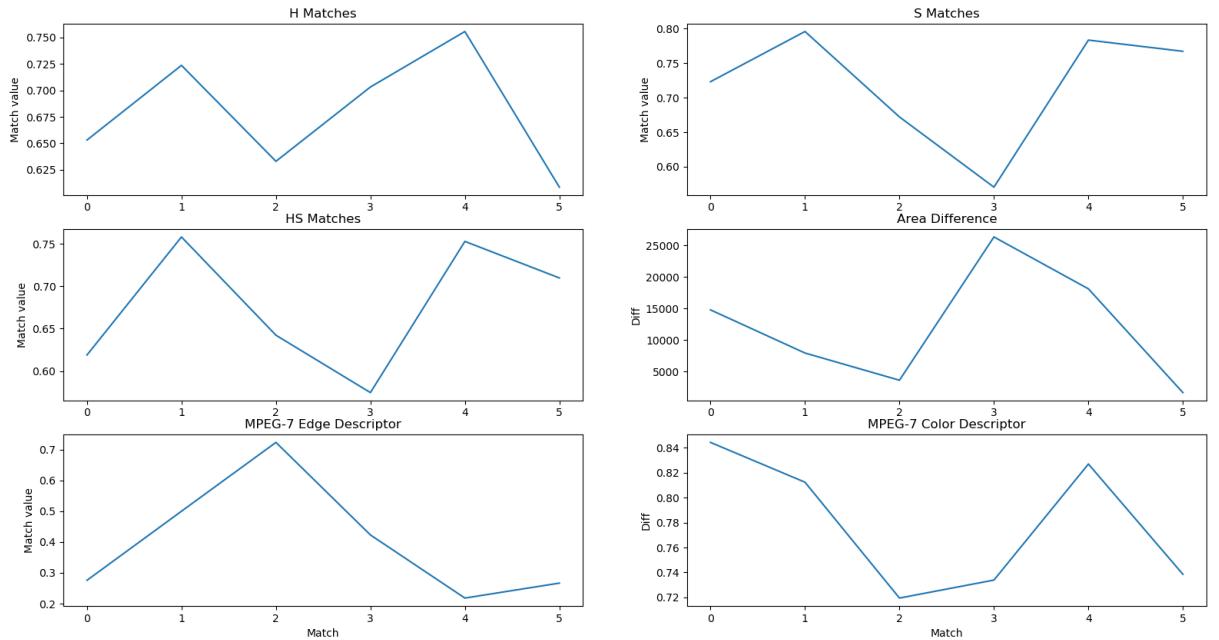


Figura 45: Informações adicionais das correspondências

0.8 Sistema de Ofuscação

Todas as aplicações/sistemas desenvolvidos para processamento de imagens ou vídeo são controladas pelo Regulamento de Proteção de Dados, sendo que este dita que não é permitido o armazenamento ou partilha de dados pessoais sem o consentimento de certa pessoa, que possam ser utilizados para identificar e criar perfil de um indivíduo. Tendo em consideração estas leis, para o nosso projeto foi necessário ter atenção como é que as imagens e vídeos capturados seriam tratados após o seu processamento, para tal implementou-se um sistema de capaz de ofuscar quaisquer faces detetadas nos dados tratados de modo a ocultar a identidade das pessoas e permitir-nos o armazenamento de vídeos ou imagens capturadas.

O sistema de ofuscação foi desenvolvido de modo a permitir a sua utilização sem depender seja do sistema de contagem ou o sistema de matching, de modo a manter uma certa modularidade. Este sistema recebe certo vídeo, as imagens do vídeo irão passar pelo método de deteção de faces ou pessoas (combinação justificada posteriormente) que devolve a caixa correspondente a uma face, à face é associada a sua "bounding box" que permitirá a ofuscação dentro dos limites dessa caixa aplicando um filtro de desfoque(blur). As imagens resultantes com as faces ofuscadas são armazenadas em vídeo, permitindo guardá-las permanentemente ou distribuí-las.

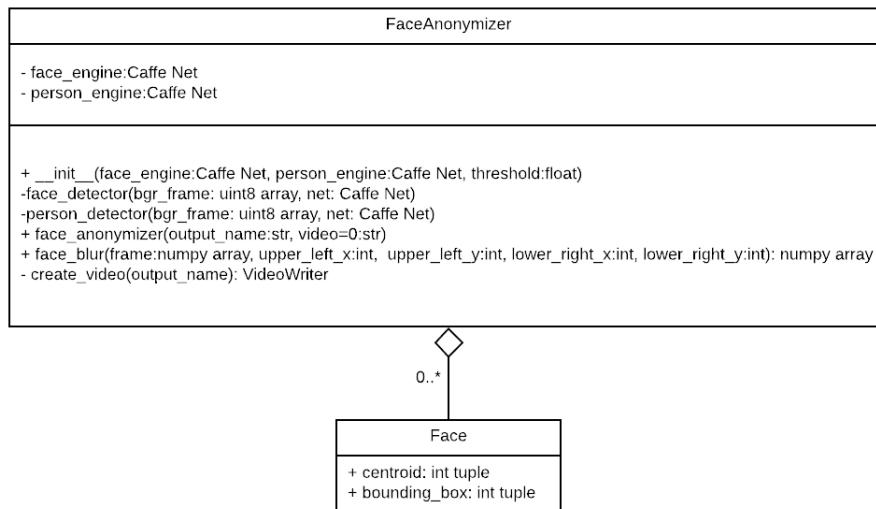


Figura 46: Diagrama de classes - Sistema de ofuscação

0.8.1 Deteção de Faces - Haar Cascades

Numa fase inicial do projeto foi utilizado um método clássico de deteção de objetos utilizando classificadores baseados em cascades, neste caso um classificador baseado em características Haar, que irá ser explicado posteriormente.

Com o objetivo de entender melhor este tipo de classificador, há que primeiro entender o conceito de de detetores baseados em cascades, nestes é treinada uma árvore de decisão em que em cada camada desta existe um classificador fraco que recebe o output de outro classificador fraco concatenado anteriormente e utiliza a informação que origina para o próximo cascade.

No contexto do nosso projeto em que o objetivo é detetar faces que possam ser filmadas em determinado evento ofuscando-as de maneira a esconder a identidade de certa pessoa. Na deteção de face, o classificador na fase treino recebe grandes conjunto de imagens positivas (imagens com faces) e imagens negativas (imagens sem faces), são extraídas características utilizando características Haar que baseiam-se na Fig.52, cada característica extraída é calculada subtraindo a soma dos pixéis na região branca com a soma dos pixéis da região preta.

As características extraídas são aplicadas a todas as imagens de treino e para cada característica é calculado o melhor limiar para classificar a face como positiva ou negativa, sendo as características com menor percentagem de erro selecionadas. Com classificações erradas, o peso destas é aumentado e o processo é repetido até diminuir ao máximo a percentagem de erro.

O que distingue o classificador por Haar cascades é o facto deste em vez de aplicar um número elevado de características, estas são divididas em camadas de classificadores em que se uma janela que percorre a imagem falha na primeira camada, esta é descartada. No caso de uma janela passar a todas as camadas esta é considerada uma face, como pode ser observado na Fig.48.

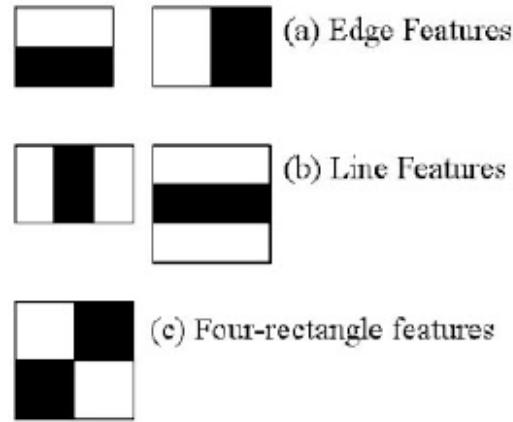


Figura 47: Camadas de atributos Haar



Figura 48: Atributos Haar aplicados a uma imagem

Para este projeto utilizámos classificadores pré-treinados fornecidos pela biblioteca OpenCV sob o formato XML, sendo cada classificador treinado para detetar certo objeto ou característica. Ao utilizarmos este método concluímos que não se adequava aos objetivos do nosso trabalho, visto que os resultados obtidos foram maioritariamente negativos, encontrando várias desvantagens na utilização de um classificador deste tipo, desvantagens como:

- A limitação de um classificador apenas estar treinado para identificar apenas certo objeto ou característica, como face, olhos, boca ou objetos como carros.
- Certa face não é detetada se o seu ângulo for irregular.
- Um classificador está treinado para detetar faces de perfil apenas para um certo lado, sendo necessário o espelhamento da imagem.
- O desempenho em termos de execução é lento.
- A necessidade de utilizarmos dois classificadores, um para deteção de faces frontais e outro para faces de perfil.

0.8.2 Deteção de Faces - Res10 Net

Após verificar os resultados que a deteção de objetos através de redes neurais convolucionais, reconhecemos que para desenvolver um sistema mais robusto e preciso teríamos de recorrer a aprendizagem profunda. Para tal tirámos partido novamente de um modelo pré-treinado para detetar apenas faces, este modelo consiste em na rede base ResNet em conjunto com a rede de deteção SSD. A implementação foi igual à realizada para deteção de pessoas com a rede VGG Net, sendo a única diferença o output da deteção, isto é, uma tinha como finalidade a deteção de uma pessoa outra foi designada para detetar apenas faces de pessoas. Os resultados da deteção com esta rede foram mais satisfatórios quando comparados com o método de deteção com classificadores baseados em Haar cascades, onde faces de perfil ou com um certo ângulo não eram detetadas, como se pode observar nas Figs.49 e 50.

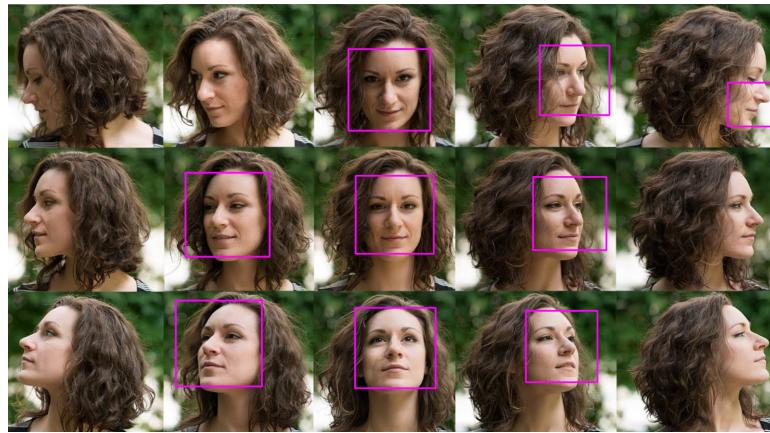


Figura 49: Deteção de face com Haar Cascades

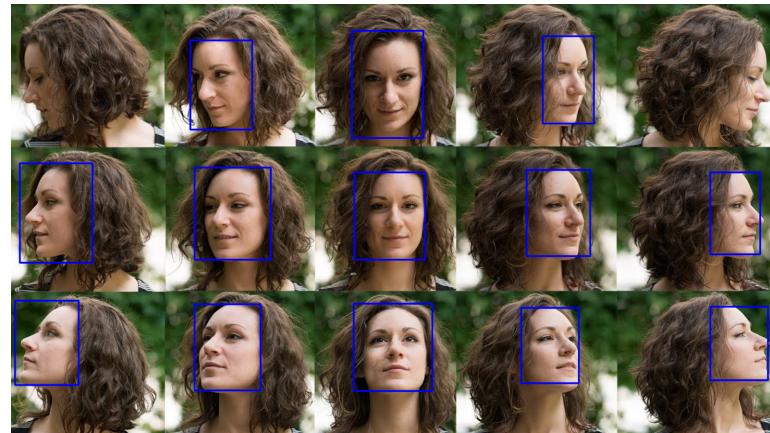


Figura 50: Deteção de face com rede neuronal convolucionarial

0.8.3 Sistema de Ofuscação - Abordagem Alternativa

Os resultados que obtivemos em testes preliminares revelaram que a deteção a partir de rede neuronal convolucional seria a mais indicada, visto que ocorriam menos falhas de deteção e tinha uma precisão satisfatória. No entanto como foi demonstrado na Fig.50, ainda ocorriam situações em que a face de uma pessoa não era detetada por exemplo quando certa face estava a um ângulo demasiado alto, seja vista de cima ou de baixo, a situação mais preocupante foi o facto de não ocorrer deteção devido à resolução da imagem. Com o objetivo de tornar este sistema o mais robusto possível com menos erros possíveis complementámos a deteção de face com a deteção de pessoa, isto é, observamos que em casos que uma face não fosse detetada sempre poderíamos utilizar a deteção da pessoa para compensar estes erros de deteção.

A lógica da complementação destas duas deteções consistiu em apenas aplicar a deteção de face a uma pessoa que já tenha sido detetada, ou seja, é utilizado apenas a imagem dentro dos limites da bounding box resultante da deteção da pessoa (Fig.). Este método permitiu em casos em que uma pessoa é detetada mas não a sua face, logo a partir do centroide calculado é realizada a ofuscação do centroide para cima, este processo proporcionou resultados com menos falhas de deteção, tendo em conta que apenas é necessário um frame em que uma face esteja visível para não estar de acordo com o Regulamento de Proteção de Dados.



(a) Ofuscação de pessoa

(b) Ofuscação de face

Figura 51: Resultados de Sistema de Ofuscação

0.9 Implementação no Raspberry pi

O desenvolvimento destes sistemas foi realizado inicialmente com o objetivo de funcionarem nas máquinas onde foram implementados(computadores) e numa próxima fase seriam adaptados para o funcionamento no Raspberry Pi. O objetivo principal da implementação neste equipamento foi permitir maior portabilidade dos sistemas de tracking, matching e ofuscação e combiná-los numa aplicação geral que iria correr no Raspberry Pi em tempo real.

O desempenho dos métodos de deteção através de aprendizagem profunda referidos anteriormente era demasiado baixo, sendo que o tempo de inferência na deteção de pessoa nos nossos equipamentos era cerca 0.1 segundos, enquanto no caso do Raspberry Pi demorava cerca 2 segundos a realizar o processo de inferência para apenas uma imagem. Esta situação impossibilitava a deteção e a contagem em tempo real que dependia do tempo de processamento de cada imagem capturada pela câmara, ou seja, durante esse tempo de deteção se uma pessoa estivesse a deslocar-se rapidamente quando a inferência acabasse para uma frame na próxima a ser processada essa pessoa poderia já não estar em cena. Uma solução que consideramos foi processar estas imagens de acordo com o buffer, isto é, enquanto a câmara capturasse essas imagens estas iriam ser processadas pela ordem em que eram armazenadas nesse buffer, garantindo assim o processamento de cada uma, no entanto com a recomendação dos nossos coordenadores, deram-nos conhecimento de uma tecnologia desenvolvida pela Google chamada Edge TPU que tinha como função correr a inferência de modelos de redes neuronais tirando o processamento da máquina principal onde é utilizado(Raspberry Pi).

0.9.1 Edge TPU

A solução da Google para tarefas onde é necessária a utilização de inteligência artificial, foi desenvolver um chip com a função de correr inferência de redes neuronais "on edge", isto é, não recorrer a serviços externos em cloud para permitir que o processamento seja totalmente local, evitando cenários em que é necessária a existência de uma ligação com a internet aumentando também o grau de privacidade dos dados tratados.

O equipamento utilizado para este projeto foi o Coral USB Accelerator, um "acessório" com ligação USB compatível com sistemas operativos baseados em Linux que age como coprocessador a um sistema removendo a necessidade de ser este que processa a inferência das redes neuronais. Tirando partido de uma API para Python fornecida pelo Coral no módulo "edgetpu", a utilização deste equipamento foi um processo facilitado que consistia em criar uma instância de "Detection Engine" com um modelo de uma rede neuronal convolucional e passar-lhe uma imagem para deteção, sendo que é devolvido uma lista de candidatos de deteção com a sua classe, grau de confiança e as suas coordenadas.



Figura 52: Coral USB Accelerator

0.9.2 Aplicação Geral

Como já foi referido a aplicação "main" é executada no Raspberry Pi tem como propósito utilizar os sistemas implementados de contagem, de matching e ofuscação e combiná-los numa aplicação capaz de utilizar uma captura de vídeo de uma câmara para realizar a contagem de pessoas que entram e saem de um certo local, fazer a sua correspondência no fim da captura e a ofuscação de faces do vídeo capturado para fins de armazenamento.

Para o desenvolvimento desta aplicação foi necessário alterarmos a estrutura dos sistemas de tracking como também de matching, esta necessidade proveio do facto que a extração de características como os histogramas estruturantes atrasava o processamento em cerca de 3 segundos no computador portátil enquanto no Raspberry Pi demorava perto de 30 segundos. A alteração mais significativa foi mudar a extração de características para que seja realizado apenas no fim de certa captura, adicionando uma flag para distinguir se é desejada a extração em tempo real, ou seja, quando é detetada uma entrada ou saída de pessoa, ou ser realizado no fim da contagem e da captura.

Com o objetivo de facilitar a primeira utilização desta aplicação, seja quando for necessário iniciá-la ou pará-la, esta foi desenvolvida de maneira que possa ser interrompida de forma segura através da verificação da existência de um ficheiro temporário que determina o fim da execução da aplicação. Este ficheiro que serve como flag é criado através de scripts que podem ser executados sem ter de recorrer á própria "main app", com a intenção de permitir uma execução remota da aplicação foi ligado o protocolo de rede SSH no Raspberry Pi de modo a permitir ligações remotas e controlo através do terminal deste.

Como já foi explicado brevemente na arquitetura deste projeto a aplicação geral tem a seguinte lógica:

- O sistema de tracking e contagem de entradas e saídas de pessoas é executado continuamente, em que a porção da imagem onde se encontra a pessoa(bounding box) na entrada ou saída é guardada para cada instância de pessoa.
- Através de deteção de movimento, apenas é armazenado temporariamente o vídeo que irá ser utilizado no sistema de ofuscação de faces.
- Após a interrupção da aplicação, através das imagens correspondentes a cada pessoa dentro da sua bounding box, são extraídas as suas características.
- Depois da extração de características o algoritmo de matching faz a correspondência das pessoas que entraram e saíram num determinado local.
- Os resultados de matching, isto é, a imagem das pessoas par entrada-saída são armazenados para fins de testes e verificações.
- O vídeo gravado utilizando deteção de movimento é passado pelo sistema de ofuscação de faces que guarda o vídeo novo permanentemente com a identidade das pessoas ocultadas, sendo que o vídeo original é apagado do equipamento.
- A ligação á internet é verificada através de uma tentativa de ligação a um servidor DNS da Google, no sucesso dessa conexão a base de dados e o vídeo com as faces ofuscadas são armazenados remotamente, no caso de não verificar-se conexão à internet a base de dados é armazenada localmente.

0.10 Disponibilização dos algoritmos

Durante o desenvolvimento do projeto, foram implementados um conjunto de algoritmos necessários para o cumprimento dos objetivos: tracking, deteção de pessoas, sistema de contagem, correspondência e ofuscação de faces. Sendo que cada módulo pode ser visualizado de forma independente, foi desenvolvida uma aplicação web de modo a que outros utilizadores possam tirar partido dos algoritmos.

A figura 53 ilustra a arquitetura inerente à aplicação. Tal como se pode observar, segue uma metodologia tradicional cliente-servidor que comunicam através do protocolo HTTP. Adicionalmente, apresenta informação dos blocos inerentes ao processamento de um pedido:

1. Validação dos parâmetros do pedido: validações dependem do tipo de pedido;
2. Aplicação do respetivo algoritmo: tal como afirmado anteriormente, tira partido dos algoritmos já implementados;
3. Formulação de uma resposta dados os resultados do algoritmo: apenas implementado com respostas no formato html. Para a construção de conteúdos dinâmicos, são utilizados *tornado templates*. Em uma fase futura, pretende-se suportar respostas no formato xml, de modo a que possa ser integrado com outras aplicações.

Relativamente às tecnologias adotadas, utilizou-se a biblioteca *tornado (Python Web Server)* para a implementação do servidor HTTP, uma vez que os algoritmos foram desenvolvidos em linguagem *Python*. Este servidor armazena os ficheiros que foram enviados por parte dos utilizadores, os resultados, *tornado templates* e um conjunto de ficheiros adicionais de grafismo (css) e dinamismo (js) associados aos conteúdos HTML.

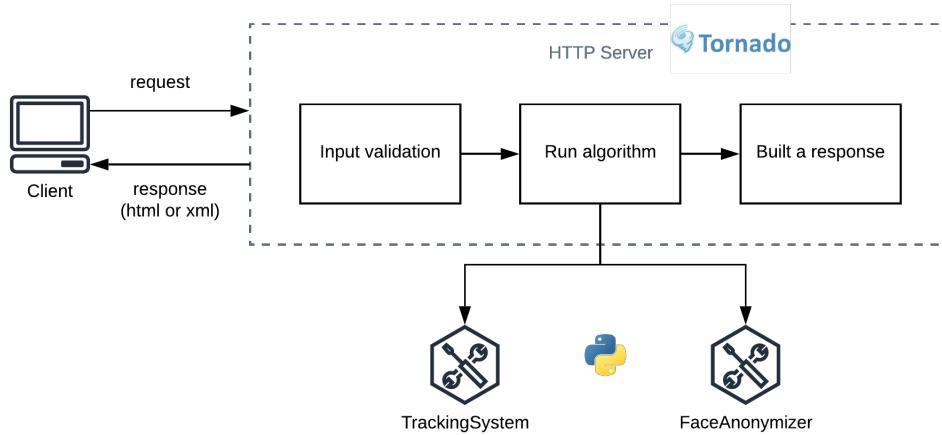


Figura 53: Arquitetura

A tabela 3 ilustra o formato dos pedidos HTTP suportados. Tendo como base os algoritmos de ofuscação e contagem, são necessários os seguintes parâmetros:

- algoritmo de ofuscação: vídeo a ofuscar, formato da resposta;
- sistema de contagem: ficheiro a aplicar linha virtual, direção de entrada, algoritmo de tracking, formato da resposta.

URL pedido	Tipo	Parâmetros	Finalidade
<IP servidor>:<Porto servidor>/obfuscation	POST	upload_file	ficheiro a aplicar o algoritmo de ofuscação
		output_format	formato em que se pretende a resposta
<IP servidor>:<Porto servidor>/counting	POST	upload_file	ficheiro a aplicar o algoritmo de contagem
		output_format	formato em que se pretende a resposta
		xInitial	posição x do ponto inicial da linha virtual de contagem
		yInitial	posição y do ponto inicial da linha virtual de contagem
		xEnd	posição x do destino da linha virtual de contagem
		yEnd	posição y do ponto destino da linha virtual de contagem
		xEntryDir	posição x da direção da entrada
		yEntryDir	posição y da direção da entrada
		trackingAlg	algoritmo de tracking: "nearest_centroid" ou "iou"
		trackingThr	valor do threshold a ser utilizado pelo algoritmo tracking

Tabela 3: Pedidos HTTP

As respostas, tal como afirmado anteriormente, apresentam html como *content-type* por omissão (único implementado). Consoante o tipo do pedido, a página html (*tornado template*) recebe como parâmetros um conjunto de objetos que permitam a demonstração dos resultados. Cada algoritmo apresenta a seguinte informação:

- Algoritmo de ofuscação: o vídeo ofuscado é demonstrado ao utilizador;
- Sistema de contagem: os resultados da contagem são demonstrados sob a forma de uma dashboard com três abas: informação geral, informação das pessoas detetadas, gráficos relativos às características. O número de entradas e saídas são enumerados na aba de informação geral, em conjunto com um vídeo que proporciona a verificação da contagem. Quanto à informação das pessoas, são ilustradas sob a forma de *cards* contendo informação da sua imagem, id e hora de deteção. Adicionalmente, na aba *plots*, são ilustrados um conjunto de gráficos para análise das características das pessoas detetadas (área, histograma de cor, histograma de saturação, histograma de contornos, histograma de cor estruturante).

A estrutura necessária, para o correto funcionamento do servidor, está representada na figura 54. O servidor, ao ser inicializado (figura 55), define um conjunto de pares URL-Handler de modo a que, por cada pedido recebido, contenha informação de qual objeto irá tratar do seu processamento. Deste modo, foi desenhado um comportamento para lidar com pedidos de ofuscação (*ObfuscationHandler*) e outro com pedidos de contagem (*CountingHandler*). A fronteira *GlobalLib* contém um conjunto de mecanismos comuns aos dois comportamentos: validação de extensão dos vídeos, geração de nomes para armazenamento de ficheiros com base na hora, upload de ficheiros e definições de configuração (porto, diretoria de upload, diretoria de resultados, mensagens de erros).

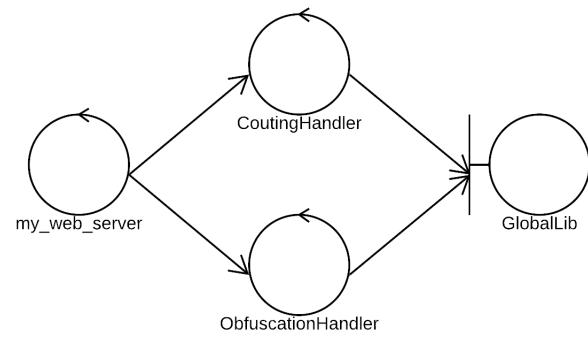


Figura 54: Diagrama de classes - Fase de análise

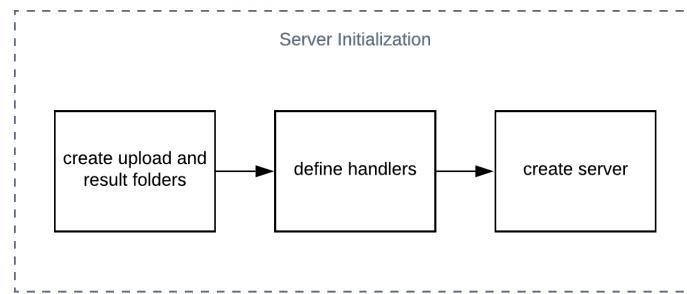


Figura 55: Inicialização do servidor

0.11 Métricas de Desempenho

De maneira a avaliar os algoritmos de correspondência e de contagem de pessoas foi necessário calcular a classe verdadeira do exemplo que foi testado. Especificamente, era feita a comparação dos resultados da correspondência e da contagem de pessoas(predicted class) com a classe verdadeira(true class). Para a criação da classe verdadeira(no caso da correspondência), à medida que as pessoas intersetavam a linha de contagem era introduzido um "alias" à pessoa de forma constante, ou seja, se ela passasse mais do que uma vez era atribuído o mesmo alias. Resumidamente uma pessoa podia ter varias correspondências no caso de passar várias vezes pela linha de contagem. Toda esta informação era introduzida no Objeto *True_class*(dicionário implementado), que pode visto na Figura 56, que herda do Objeto *dict* do Python, o conteúdo do dicionário podia seguir os seguintes casos:

- alias:[id,id]
- id:alias

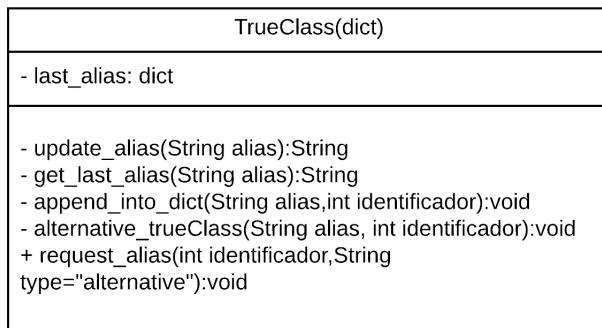


Figura 56: Dicionário da classe verdadeira

A razão pela qual o dicionário podia tomar ambas abordagens dependia da métrica pretendíamos obter o desempenho. No primeiro caso, e com menos acertos, o que pretendia-se era obter a correspondência absoluta em que efetivamente o identificador x correspondia ao identificador y. No segundo caso, e admitindo que uma pessoa tinha mais do que uma correspondência, a única restrição imposta para que o resultado desse certo era que a correspondência fosse constituída pela mesma pessoa. Relativamente a criação da classe verdadeira no caso da contagem de pessoas, inicialmente experimentamos um sensor localizado na porta que servia de "GroundTruth" para o algoritmo implementado e tirar algumas medidas de desempenho. No entanto, o sensor produzia erros na contagem. Quando a porta era aberta e fechada a contagem era incrementada de maneira errónea(o sensor contava a porta). Outra abordagem que foi feita(parecida com a da correspondência) foi analisar os vídeos gravados e tirar dados para fazer a avaliação posteriormente.

Uma vez criada a classe verdadeira é possível fazer a comparação com os resultados obtidos pelos algoritmos implementados, através do Objeto *MethodValidation* que foi concebido com uma série de métodos para obter resultados de avaliação(Figura. No caso da contagem de pessoa, foi visto como um problema binário em que a classe "pessoas de entrada"era a classe dos positivos e a classe "pessoas de saída"era a classe dos negativos. A partir do que foi dito, foram tiradas métricas tais como: precision, recall, F1-Score e a matriz de confusão de um vídeo exemplo. O objetivo desta avaliação era medir o desempenho da contagem no âmbito de ser entrada ou saída e verificar se o algoritmo que retornava entrada ou saída estava bem implementado. Para medir o algoritmo que tinha como objetivo determinar se a pessoa intersetava a linha virtual, calculou-se a percentagem de acertos do resultado obtido com a classe verdadeira criada anteriormente.

Relativamente à avaliação ao algoritmo de correspondência foram implementados dois métodos que calculavam a probabilidade de acertos para os dois casos que foram mencionados, em que o dicionário podia ser estruturado de duas maneiras.

MethodValidation
- indice : int
- matches_dict:dict
+ confusion_matrix(list trueClass, list predictedClass):void
+ score_alternative_matches(Dict trueClass, Dict predictedClass):String
+ score_matches(Dict trueClass, Dict predictedClass):String
+ metricasBin(list target, list result):void
+ match_to_array(list match):dict

Figura 57: Classe MethodValidation

0.12 Validação e Testes

Para os testes foram tomados alguns exemplos(em forma de vídeo) filmados nos escritórios da CardioID utilizando o Raspberry Pi. Para testes preliminares foram gravados alguns minutos, mas para testes mais realistas foram gravados dois vídeos durante 3 dias. No entanto estes dois,e devido as capacidades de armazenamento do Raspberry Pi, apenas aramzenavam as frames em que era detetado movimento, reduzindo assim o tamanho e a duração do vídeo. Como foi dito anteriormente para obter resultados dos algoritmos implementados foi necessária a construção da classe verdadeira para calcular o desempenho. Decidiu-se, criar o dicionário da classe verdadeira para o vídeo que foi gravado durante 1 dia(19min,132 entradas/saidas),e os resultados da correspondência podem ser visualizados na seguinte Tabela 4:

<i>Algoritmo de correspondência</i>	<i>k</i>	<i>t</i>	<i>Iterações</i>	<i>P(acertos)</i>
Brute Force	10	0	10000	60.976%
Brute Force	20	0	10000	59.055%
Brute Force	30	0	10000	59.174%
Brute Force	40	0	10000	59.807%
Brute Force	10	0.1	10000	59.899%
Brute Force	10	0.4	10000	60.0%
Brute Force	10	0	20000	60.249%
Brute Force	10	0.5	2500	61.772%

Tabela 4: Resultados da correspondência(video de 19min)

O algoritmo Brute Force é um algoritmo iterativo em que para cada iteração corresponde pessoas aleatoriamente. Para cada iteração é calculado um valor de custo, valor este que é a soma de distâncias dadas pela expressão:

$$d = (1 - t) * distancia + t * \Delta t$$

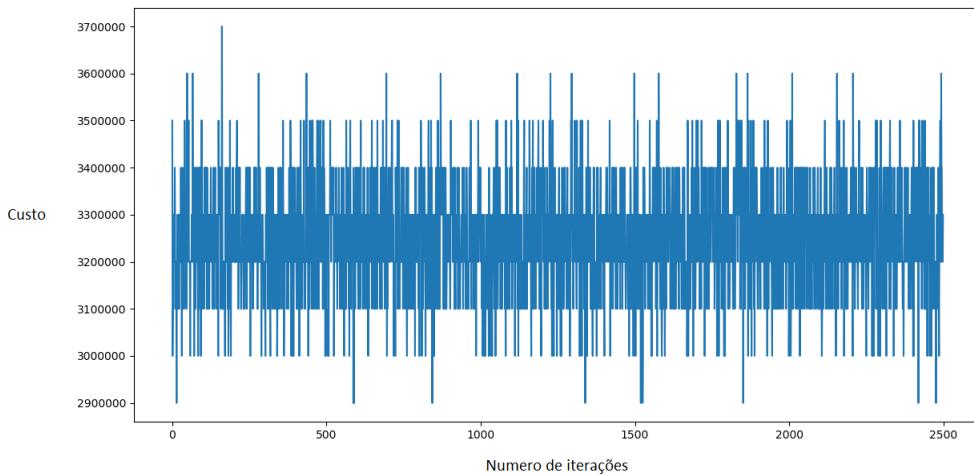


Figura 58: Função de custo para 2500 iterações, k=10, t=0.5

Como foi referido anteriormente, é um algoritmo iterativo em que corresponde pessoas aleatoriamente e é calculado um valor de custo para cada iteração. O resultado final será a iteração em que o custo foi o menor entre todos. Na figura 58 é possível observar alguns picos e oscilações no gráfico, devido ao peso que é atribuído caso a uma pessoa não lhe ser atribuída uma pessoa de correspondência, desta maneira "obrigamos" o algoritmo a escolher uma iteração em que hajam o maior número de correspondências possíveis.

Em relação aos resultados da contagem foram feitos dois testes: probabilidade de acerto(se a contagem estava correta) e verificação de entradas e saídas(caso binário). Os resultados dos testes foram obtidos com o mesmo vídeo utilizado para os outros testes antes referidos sendo que a probabilidade de acerto na contagem foi de 94.964%. Relativamente às entradas e saídas foi desenhada uma matriz de confusão, Figura 59, para serem observadas entradas ou saídas que tenham sido contadas de forma errónea.

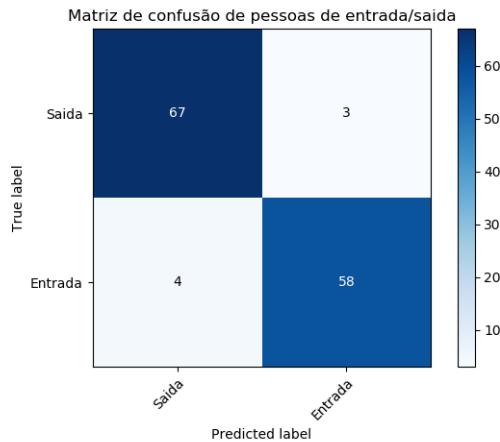


Figura 59: Matriz de confusão entradas/saidas

0.13 Conclusões

O projeto desenvolvido apresenta conclusões em diferentes setores. Relativamente ao sistema de contagem, no geral, apresentou bons resultados. No entanto, devido a certo ruído que possa haver nos centróides das pessoas detetadas, houve algumas instâncias no qual foram contadas com um sentido incorreto. Neste âmbito também foram feitas experimentações com um sensor de contagem, colocado em cima de uma porta, mas apresentava a desvantagem de efetuar a contagem do movimento da porta. Por outro lado, a linha virtual apresenta a desvantagem de ser uma linha 2D. Futuramente, seria interessante a exploração de um plano 3D virtual.

Em termos das características utilizadas para realizar a correspondência entre pessoas, poderíamos ter explorado em mais detalhe que conjunto de características é que daria melhores resultados. O desafio deste projeto era realizar a correspondência de pessoas utilizando apenas uma câmara, logo, a correspondência teria de ser feita tendo em consideração que uma pessoa estaria em direções opostas para a câmara, mesmo assim, através de deteção de face poderia-se ter extraído características das faces das pessoas. As características das faces são atualmente utilizadas para o reconhecimento facial e identificação de identidades, uma vez que permitem a distinção de pessoas diferentes e demonstram uma certa unicidade para uma pessoa.

O sistema de ofuscação implementado revelou resultados satisfatórios, sendo que talvez os métodos utilizados não são os mais adequados, métodos como realizar a deteção de face apenas a pessoas já detetadas causando também a ofuscação mesmo que certa pessoa esteja de costas para a câmara. Para aperfeiçoar este sistema poderiam ser utilizadas métodos capazes de identificar que certa face deixou de ser detetada e a partir daí é que se utilizava a deteção de pessoas de modo a corrigir erros de deteção.

A aplicação geral, que junta todos os módulos desenvolvidos, reflete os objetivos enumerados na introdução do trabalho. Permite a aplicação do sistema de contagem e na sua interrupção a aplicação do algoritmo de correspondência/algoritmo de ofuscação. Deste modo, são armazenadas informações que contribuem para as finalidades da aplicação (na sua maioria estatísticos). No entanto, tal como já mencionado ao longo do trabalho, realça-se que cada módulo funciona de forma autónoma, sendo possível a sua disponibilização a outros utilizadores/aplicações. Em uma fase final, foram implementados dois exemplos através de uma aplicação web (ofuscação de faces/sistema de contagem). Seria pretendido, em uma fase futura, o melhoramento da aplicação web e a generalização da disponibilização dos algoritmos através de *web services*, de modo a que possam ser utilizados por outras aplicações.

0.14 Trabalho Futuro

Para este projeto tirámos partido da aprendizagem profunda através da deteção de objetos utilizando redes neuronais convolucionais, no entanto estas redes pré-treinadas são capazes de detetar objetos que não sejam pessoas, são eficazes, mas talvez a sua precisão aumentasse se fossem treinadas exclusivamente para a deteção de pessoas. Após a gravação dos vídeos nos escritórios, poderíamos utilizar as suas imagens para criar um dataset com as pessoas capturadas em vídeo, esse dataset poderia consequentemente ser utilizado para realizar o treino de uma rede neuronal convolucional, seja treiná-la de "raiz" ou através do equipamento Coral Edge TPU que permite uma técnica de transfer learning. Esta técnica de transfer learning tira proveito de um modelo de rede pré-treinado para retreiná-lo evitando um maior tempo de computação que seria necessário para treinar uma rede neuronal convolucional desde o início.

A aplicação geral que é executada no Raspberry Pi pode ser lançada e terminada remotamente através de envios de comandos por ligação SSH, contudo, para um indivíduo sem conhecimentos programáticos esta não seria a melhor solução. Foi começado o desenvolvimento de uma aplicação capaz de controlar o sistema baseado mais uma vez em conexões SSH, mas iria facilitar a sua utilização, já que a parte dos envios dos comandos necessários seria ocultada para os utilizadores. Na primeira abordagem desta aplicação é possível lançar a aplicação geral no Raspberry Pi, no entanto, deparamo-nos com "bugs" no processamento dessa aplicação, portanto não achámos relevante entrar em detalhe neste relatório.

Foi desenvolvida a aplicação Web que permitia a utilização dos sistemas implementados, porém uma aplicação Web para visualizar os resultados obtidos pela aplicação geral do Raspberry Pi também seria útil para os utilizadores que instalassem este sistema. Esta aplicação Web permitia verificar a base de dados armazenada remotamente no serviço RDS da Amazon, bem como os vídeos com as faces ofuscadas gravados a partir da deteção de movimento.

Para que os testes e avaliações de todos os exemplos tomados fossem feitos de modo mais automático seria interessante tomar partido de sensores externos como medida de "GroundTruth". Como foi dito anteriormente foi experimentado um sensor de entradas e saídas, mas devido a sua localização havia alguns erros. Na parte da correspondência para que todo o processo de criação da classe verdadeira fosse mais amigável e automático, seria necessário optar por outro tipo de abordagem.

Bibliografia

- [1] Coral with Google. *Get started with the USB Accelerator.*
<https://coral.withgoogle.com/docs/accelerator/get-started/>
- [2] Axis Communications. *AXIS People Counter.*
<https://www.axis.com/pt-pt/products/axis-people-counter>
- [3] Viswarupan, Niruhan. *A Practical Guide to Person Re-Identification Using AlignedReID.* Medium, 25 de Junho de 2018,
[https://medium.com/@niruhan/a-practical-guide-to-person-re-identification-using-alignedreid-7683222da644.](https://medium.com/@niruhan/a-practical-guide-to-person-re-identification-using-alignedreid-7683222da644)
- [4] Rosebrock, Adrian. *Real-Time Object Detection with Deep Learning and OpenCV.* PyImageSearch, 18 de Setembro de 2017,
[https://www.pyimagesearch.com/2017/09/18/real-time-object-detection-with-deep-learning-and-opencv/.](https://www.pyimagesearch.com/2017/09/18/real-time-object-detection-with-deep-learning-and-opencv/)
- [5] Rosebrock, Adrian. *Object Detection with Deep Learning and OpenCV.* PyImageSearch, 11 de Setembro de 2017,
[https://www.pyimagesearch.com/2017/09/11/object-detection-with-deep-learning-and-opencv/.](https://www.pyimagesearch.com/2017/09/11/object-detection-with-deep-learning-and-opencv/)
- [6] Saha, Sumit. *A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way.* Towards Data Science, 15 de Dezembro de 2018,
[https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.](https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53)
- [7] Gao, Hao. *Understand Single Shot MultiBox Detector (SSD) and Implement It in Pytorch.* Medium, 6 de Junho de 2018,
[https://medium.com/@smallfishbigsea/understand-ssd-and-implement-your-own-caa3232cd6ad.](https://medium.com/@smallfishbigsea/understand-ssd-and-implement-your-own-caa3232cd6ad)
- [8] Rosebrock, Adrian. *Object Detection and Image Classification with Google Coral USB Accelerator.* PyImageSearch, 13 de Maio de 2019,
[https://www.pyimagesearch.com/2019/05/13/object-detection-and-image-classification-with-google-coral-usb-accelerato.](https://www.pyimagesearch.com/2019/05/13/object-detection-and-image-classification-with-google-coral-usb-accelerato)
- [9] OpenCV: Deep Neural Networks (dnn module).
[https://docs.opencv.org/master/d2/d58/tutorial_table_of_content_dnn.html.](https://docs.opencv.org/master/d2/d58/tutorial_table_of_content_dnn.html)
- [10] Won, Chee Sun Won, et al. “Efficient Use of MPEG-7 Edge Histogram Descriptor.” *ETRI Journal*, vol. 24, no. 1, Feb. 2002, pp. 23–30. DOI.org (Crossref), doi:10.4218/etrij.02.0102.0103.
- [11] H. Eidenberger, C. Breiteneder, “VizIR – a framework for visual information retrieval “— *jvlc2003.pdf*
- [12] B. S. Manjunath, “Color and Texture Descriptors” *IEEE Transactions on circuits and systems for video technology*, vol. 11, no.6, June 2001