Universidade da Beira Interior
Departamento de Informática

Nº 93 - 2020: *Sistema de detecção e tracking de objectos em movimentocom base em análise de imagem*

Elaborado por:

David Miguel Martins Pires

Orientador:

Professor Doutor Pedro Domingues de Almeida

26 de Fevereiro de 2020

Agradecimentos

A conclusão deste trabalho, bem como da grande maior parte da minha vida académica não seria possível sem a ajuda de ...
- do professor Pedro Domingues de Almeida por toda a orientação e ajuda provenciada durante a elaboração deste projeto
- todos os meus docentes, colegas e amigos que me acompanharam durante este percurso académico
- Ao professor Pedro Inácio na provenidencia das bases para a realização deste Relatório e de um Apresentação a ser realizada no final deste Trabalho.

Conteúdo

Lista de Figuras

Lista de Tabelas

Acrónimos

CNN *Convolutional Neural Network*

IoU *Intersection over union*

SSD *Single Shot Detector*

SVM *Support Vector Machine*

ROI *Region of Interest*

R-CNN *Region-based Convolutional Neural Network*

R-FCN *Region-based Fully Convolutional Network*

YOLO *You Only Live Once*

Capítulo 1

Introdução

1.1 Enquadramento

Sistema de detecção e tracking de objectos em movimentocom base em análise de imagem

- Visão Computacional

Computer vision is a field of artificial intelligence that trains computers to interpret and understand the visual world. Machines can accurately identify and locate objects then react to what they "see" using digital images from cameras, videos, and deep learning models.

As computer vision evolved, programming algorithms were created to solve individual challenges. Machines became better at doing the job of vision recognition with repetition. Over the years, there has been a huge improvement of deep learning techniques and technology. We now have the ability to program supercomputers to train themselves, self-improve over time and provide capabilities to businesses as online applications.

- Identificação de objetos em imagem (2D, 3D) ou em um video sobre um ambiente de luz natural. Desafio para os sistemas de Visão Computacional, sendo que já foram implementadas muitas abordagens para estes sistemas ao longo dos anos. - O Tranking de objetos é tambem um desafio para estes sistemas de Visão Computacional.

Neste projeto todo o codigo é desenvolvido usando um Raspberry Pi, com capacidades limitadas, o que torna este trabalho ainda um maior defafio ao nivel da Complexidade Computacional.

- Complexidade Computacional

Usaremos tambem uma camera comum e conectada ao Raspberry Pi para obter as imagens para analisar em tempo real. Usarei uma camera da marca Logitech que tenho em casa, conectada via USB.

Para isto iremos usar irei usar um Raspnerry Pi 4, Modulo B de 4GB RAM que tenho em casa e um cartão de memória de 64 Gb.

1.2 Motivação

O tema deste projeto é um tema atual e muito importante para as áreas de Visão Computacional e da Robotica. Daí ser uma grande motivação minha na escolha deste trabalho.

O reconhecimento de objetos será sempre uma area imprencindivel e de grande interesse para a robotica. Neste caso iremos reconhecer alguns objetos e rastrea-los em um fluxo de video. Isto ainda em um ambiente de desenvolvimento com capacidades bastante limitadas, que é o Raspberry Pi, o que torna esta tarefa um desafio.

Apesar de nós humanos, conseguirmos reconhecer múltiplos objetos em diferentes imagens, e em diferentes disposições em segundos sem qualquer dificuldade. Muitas vezes mesmo Objetos obtruidos por outros objetos. Isto para a Visão Computacional é ainda um desafio, e é muito mais complexo reconhecer um objeto em uma imagem do que pensamos à primeira vista.

- Lab - Prof - Raspberry Pi

—— Old:

A motivação da escolha deste projeto como projeto deve-se ao meu interesse neste tema de projeto. Tenho também muito interesse no aprendizado das ferramentas utilizadas como por exemplo a programação em um dispositivo com capacidades reduzidas, como um Raspberry.

1.3 Objetivos

Como objetivo deste trabalho temos a elaboração de um Sistema de deteção e traking de objetos em movimento com base em análise de imagem. Este sistema deve ser desenvolvido para posteriormente ser usado em um Raspberry Pi, com capacidade de processamento limitada. Devemos também usar uma webcam e ferremntas de programação como Tensorflow, Opencv e Yolo.

1.4 Organização do Documento

De modo a refletir o trabalho que foi feito, este documento encontra-se estruturado da seguinte forma:

1. O primeiro capítulo – **Introdução** – apresenta o projeto, a motivação para a sua escolha, o enquadramento para o mesmo, os seus objetivos e a respetiva organização do documento.

2. O segundo capítulo – **Tecnologias Utilizadas** – descreve os conceitos mais importantes no âmbito deste projeto, bem como as tecnologias utilizadas durante do desenvolvimento da aplicação.

3. ...

1.5 Algumas Dicas – [RETIRAR DA VERSÃO FINAL]

Os relatórios de projeto são individuais e preparados em LaTeX, seguindo o formato disponível na página da unidade curricular. Deve ser prestada especial atenção aos seguintes pontos:

1. O relatório deve ter um capítulo Introdução e Conclusões e Trabalho Futuro (ou só Conclusões);

2. A última secção do primeiro capítulo deve descrever suscintamente a organização do documento;

3. O relatório pode ser escrito em Língua Portuguesa ou Inglesa;

4. Todas as imagens ou tabelas devem ter legendas e ser referidas no texto (usando comando \ref{}).

Capítulo 2

Estado da Arte

2.1 Introdução

O reconhecimento e rastreamento de objetos é um dos campos mais atuais da visão computacional.
É possivel reconhecer diferentes objetos em uma cena através das suas formas, cor e diferentes carateristicas do objeto. É também possivel rastrear um objeto em um fluxo de video através do calculo de distâncias, comparação de formas, e outras carateristicas.
Neste capitulo será abordado o estado atual do reconhecimento de objetos e o estado atual do rastreamento de objetos, e diferentes algoritmos que existem para o fazer. Isto com a especial atenção ao meio de desenvolvimento que será um Raspberry Pi, um pequeno computador com capacidades limitadas.

Este capítulo está estruturado da seguinte forma:

1. Secção 2.1 - **Estado atual do reconhecimento e rastreamento de objetos** - Esta secção apresenta uma introdução ao estado atual do reconhecimento e rastreamento de objetos, e uma explicação sobre algumas arquiteturas.

2. Secção 2.2 - **Algoritmos de reconhecimento de objetos** - Esta secção apresenta vários algoritmos estudados sobre a deteção/reconhecimento de objetos e uma breve explicação do seu funcionamento.

3. Secção 2.3 - **Algoritmos de rastreamento de objetos** - Esta secção apresenta vários algoritmos estudados sobre o rastreamento de objetos, e uma breve explicação do seu funcionamento.

4. Secção 2.4 - **Conclusões** - Esta secção apresentará uma breve conclusão sobre este capitulo.

2.2 Estado atual do reconhecimento e rastreamento de objetos

Para entendermos o estado atual do reconhecimento de objetos, é preciso ver uma breve história do reconhecimento de objetos até ao estado atual e entender algumas definições.

A deteção de objetos combina as tarefas de localização e classificação de objetos. Os detetores de objetos atuais podem ser divididos em duas categorias:
- Redes que separam as tarefas de determinar a localização dos objetos e sua classificação, onde Faster *Region-based Convolutional Neural Network* (R-CNN) é uma das mais famosas.
- Redes que prevêm caixas delimitadoras e classificações de classes de uma só vez, onde as redes *You Only Live Once* (YOLO) e *Single Shot Detector* (SSD) são famosas arquiteturas.

A primeira rede neural profunda para deteção de objetos foi Overfeat. Eles introduziram uma abordagem de janela deslizante de várias escalas usando *Convolutional Neural Network* (CNN) e mostraram que a deteção de objetos também melhorou a classificação de imagens. Foram logo seguindos por R-CNN: Regiões com carateristicas CNN. Os autores proposerão um modelo que usa pesquisa seletiva (selective-search) para gerar as proporções da região, mesclando pixels semelhantes em regiões. Cada região foi alimentada em uma CNN, que produziu um vetor de carateristica de grande dimenção. Este vetor é depois usado para a classificação final e regressão da caixa delimitadora, como mostra na figura 2.2.

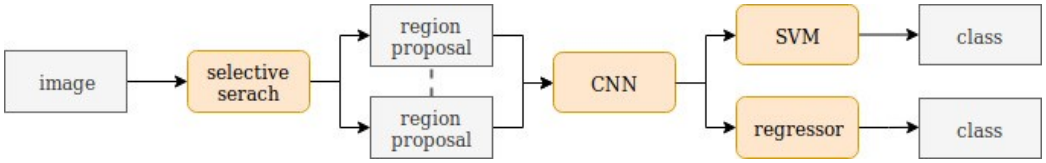


Figura 2.1: A arquitetura padrão de uma rede R-CNN consiste em um metodo de proporção regional. A classificação final é feita com uma SVM e um regressor.

SVM é um conjunto de métodos de aprendizado supervizionado que analizam os dados e reconhecem padrões, usando classificação e análise de regressão.

Uma abordagem mais sotisficada, o Fast R-CNN também gera proporções regionais com pesquisa selectiva, mas alimenta a imagem inteira através de uma

CNN. Ela superou o desempenho da rede Overfeat por uma grande margem, mas ainda assim é muito lenta, porque a geração proposicional usando pesquisa seletiva é muito demorada, tal como a necessidade de alimentar cada proposta através de uma CNN. As proporções regionais são agrupadas diretamente em um mapa carateristico por *Region of Interest* (ROI) pooling. Os vetores carateristicos agrupados são alimentados em uma rede totalmente conectada para classificação e regressão, como retratado na figura 2.2. Similar ao R-CNN, o Fast R-CNN gera as proporções regionais usando pesquisa seletiva.

ROI pooling, ou agrupemento da Região de Interesse é o processo de converter uma região de interesse da imagem original em uma imagem de tamanho fixo para que se possa passar à proxima etapa de deteção do objeto.

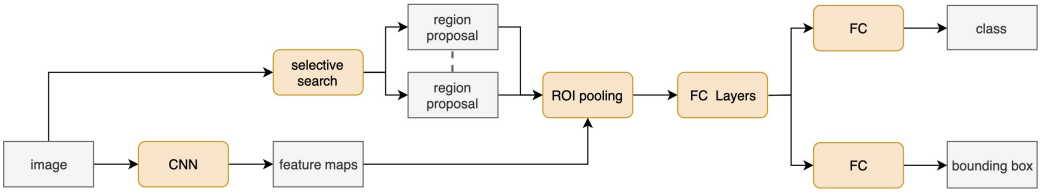


Figura 2.2: A arquitetura padrão da rede Fast-R-CNN. A proporção regional é gerada usando pequisa seletiva, mas agrupada diretamente no mapa de caraterísticas, seguida de várias camadas FC para a classificação final e regressão da caixa delimitadora.

Faster R-CNN abordou esta questão propondo uma nova rede de proporção regional, que foi unida com a arquitetura Fast R-CNN para drasticamente aumentar a velocidade do processo. Outra abordagem para a deteção de objetos em imagens é o *Region-based Fully Convolutional Network* (R-FCN), uma rede totalmente convolucional da região, que usa mapas de pontuação sensiveis à posição em vez de uma sub-rede pré-região.

O design das redes de deteção de objetos foi revolucionado pela rede YOLO. Este segue uma abordagem completamente diferente dos modelos mencionados e é capaz de prever classificações de classes e caixas delimitadoras de uma vez. O modelo proposto divide a imagem em uma grelha, onde cada celula preve classificações de confiança de um objeto estar presente com as coordenadas da caixa delimitadora correspondente. Isto permitiu previsões em tempo real com o YOLO. Os autores libertaram mais três versões, YOLO9000, YOLOv2 e YOLOv3, onde o primeiro é capaz de prever até 9000 categorias, o segundo é capaz de processar imagens maiores e o terceiro é mais preciso. Outra rede que preve classes e caixas delimitadoras de uma vez é o SSD. É comparavel com o YOLO, mas usa várias

proporções por célula da grelha e mais camadas convolucionais para melhorar a precisão.

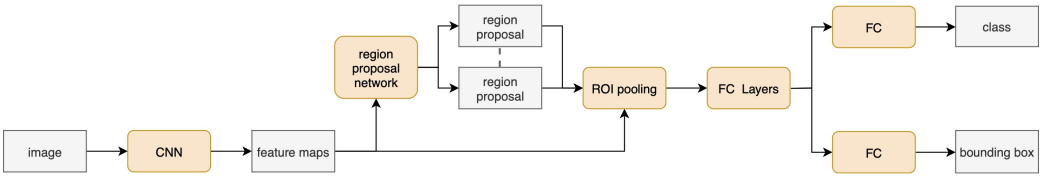


Figura 2.3: A arquitetura padrão da rede Faster-R-CNN, onde a proposta de região é gerada usando uma RPN, que trabalha diretamente com o mapa de carateristicas. As ultimas camadas estão totalmente connectadas para classificação e regressão das caixas delimitadoras.

https://towardsdatascience.com/understanding-object-detection-9ba089154df8

2.3 Algortimos de Deteção de Objetos

2.3.1 Haar cascades

Este método foi proposto por P. Viola e M. Jones em 2001. EM resumo, é um método de aprendizagem de máquina (Machina Learning) onde uma função chamada cascata é treinada em uma grande quantidade de imagens positivas e negativas (positivas significam que contêm o objeto especificado e negativas que não contem), que por sua vez podem ser usadas para a deteção de objetos.

Para isto vou explicar o conceito dos recursos Haar, estes são obtidas pelas caixas a preto e branco abaixo que atuam como Kernels convolucionais. Os recursos são mais expecificamente valores unicos recebidos através da subtração da soma dos pixels dos retangulos brancos da soma dos pixels dos retangulos pretos.

imagem 1

Se estiver desconfortável com a palavra convolução, esta é essencialmente o resultado de duas funções afetando uma à outra. Portanto, no nosso caso, isto significa, como a soma dos pixels na parte das caixas a preto e branco interagem, ou seja, são diferenciados para produzir um unico valor. Já existe um algoritmo para calcular a soma dos pixels eficientemente dentro de uma area, é chamado tabela de àrea resumida e foi publicado por F. Crow em 1984. Este foi posteriormente popularizado no dominio do processamento de imagens e fica sobre o nome de imagem integral.

imagem 2

Tentamos diferentes recursos Haar e vemos qual produz o maior valor para a diferença entre a soma dos pixels entre os retangulos pretos e brancos. Temos

abaixo um exemplo onde recursos Haar otimizados foram encontrados. Os olhos são normalmente um pouco mais escuros, onde a area abaixo é provavelmente mais clara, e assim um retangulo horizontal com preto no topo e branco em baixo é adequado. Em segundo lugar a ponte do nariz geralmente é mais clara que os olhos e então, um recurso de Haar com uma caixa branca vertical no meio é mais adequada.

imagem 3

....

https://towardsdatascience.com/object-detection-with-haar-cascades-in-python-ad9e70ed50aa

2.3.2 HOG + Linear SVM

Neste algortimo, conhecido como Single Shot Detector (Detector de com apenas uma foto), tiramos apenas uma foto para detetar multiplos objetos presentes em uma unica imagem usando multibox (multiplas caixas). É significamente mais rápido em velocidade e grande precisão em algoritmos de deteção de objetos.
A rápida velocidade e precisão do SSD usando relativamente imagens com baixa resolução é atribuida devido às seguintes razões:
- Remove caixas delimitadoras como as que são usadas em R-CNN's
- Inclui um filtro convulucional progressivamente decrescente para prever as categorias dos objetos e compensações no lugar das caixas delimitadoras.
A Alta presição de deteção no SSD é obtida usando multiplas caixas ou filtros com diferentes tamanhos, e proporção de tela para a deteção de objetos. Isto também aplica esses filtros a vários mapas de recursos dos estágios posteriores de uma rede. Isso ajuda a executar a deteção em várias escalas.

2.3.3 SSDs

2.3.4 Faster R-CNNs

2.3.5 Yolo

Yolo! *(**Yolo!**) (You Only Look Once)* é um algortimo de deteção de objetos que aceita um feed de video em tempo real. No Yolo, com uma unica CNN ??? ??? simultanemente prevemos multiplas caixas delimitadoras e probabilidades de classes para essa caixa. O Yolo treina em imagens completas e otimiza diretamente o desempenho da deteção. Este modelo afirma ter um numero de beneficios sobre outros métodos de deteção de objetos, como:
- YOLO encontra todos os objetos em uma imagem simultanemente.
- Usa uma unica rede convulcional para toda a imagem.

- Usa carateristicas da imagem inteira para prever cada caixa.delimitadora. Tambem preve todas as caixas delimitadoras de todas as classes para uma imagem simultanemanete. Prevê as caixas delimitadoras e as probabilidades de classe para essas caixas.
- YOLO vê a imagem inteira durante o treino e tempo de teste, portanto codifica implicitamente informações contextuais sobre classes tal como a sua aparencia.
- YOLO aprende representações generalizadas de objetos para que quando treinado em imagens naturais e testado em obras de arte, o algoritmo supera outros modelos de deteção.

Trabalho do YOLO

- YOLO pega em uma imagem e divide-a em uma rede SxS. Cada célula prevê apenas um objeto. - A clissificação e localização da imagem são aplicados em cada célula. - Se o centro de um objeto fica em uma célula, essa célula é responsavel por detetar esse objeto. - Cada célula da rede prevê B caixas delimitadoras com classificações de confiança para essas caixas.

1. Caixas de confiança refletem o quão confiante o modelo está sobre a caixa conter o objeto e quão preciso ele pensa a caixa ser o que ele pensa. Se o objeto não existir, então a classificação de confiança será zero.

A previsão de confiança reprensenta a IOU entre a caixa de previsão e qualquer caixa de verdade do solo. Pr(Object) * IOU verdade previsão - interseção de união (*Intersection over union* (IoU)) entre a caixa prevista e o verdadeiro solo.

The output of the algorithm is a list of bounding box, in format [class, x, y, w, h, confidence]. The class is an id related to a number in a txt file (0 for car , 1 for pedestrian, . . .). x, y, w and h represent the parameters of the bounding box. x and y are the coordinates of the center while w and h are its size (width and height). The confidence is a number expressed in

2.3.6 Tensorflow

Tensorflow Lite

2.3.7 OpenCV

Opencv é uma biblioteca *open source* criada em 2000 pela Intel e com Licença BSD Intel para a sua Distribuição. Utilizada para o desenvolvimento de aplicações na área de computação visual, seja para o uso acadêmico ou comercial. Possui módulos para processamento de imagens, estrutura de dados, àlgebra linear, bem como uma interface gráfica para usuário e mais de 350 algoritmos de visão

computacional, como filtros de imagem, calibração de câmera, reconhecimento de objetos, análise estrutural e outros.

2.4 Algortimos de Rastreamento de Objetos

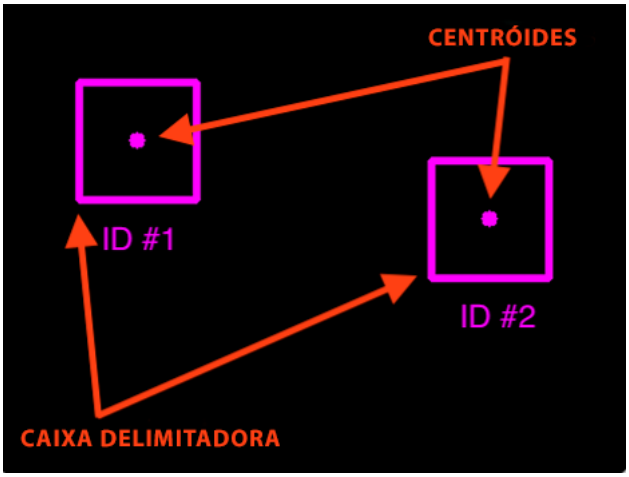
2.4.1 Centroid tracking

Centroid traking é um algoritmo presente na biblioteca OpenCV, que depende da distância euclidiana[1] entre centróides de objetos (ex. objetos que o rastreador de centróide já viu antes) e novos centróides de objetos entre quadros subsequentes em um video.

O algoritmo de rastreamento de centróides é um processo de multiplas etapas:

1- Aceitar as coordenadas da caixa delimitadora do objeto e calcular os centróides

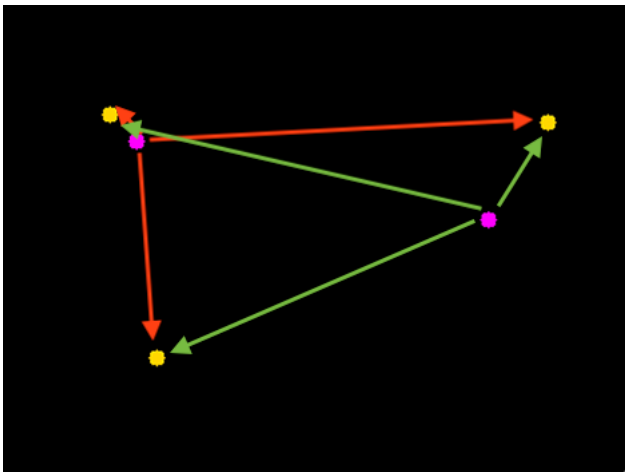
Este algoritmo assume que estamos a passar um conjunto delimitador de coordenadas 2D de cada objeto detectado em *every single frame* (cada frame de video). Estas caixas delimitadoras podem ser produzidas por qualquer tipo de detetor de objetos que mais gostemos (temos em este capitulo alguns algoritmos de delimitadores explicados).
Depois de obtermos as coordenadas das caixas delimitadoras, devemos determinar o centróide, ou simplesmente, o centro da caixa.
Visto que este é o conjunto inicial de caixas, devemos lhes atribuir IDs unicos.



[1] Dintância entre dois pontos

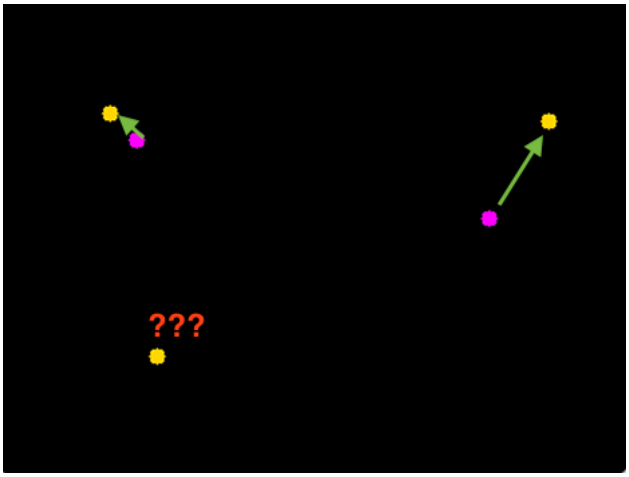
2- Calcular distância euclediana entre novas caixas delimitadoras e objetos existentes

Para cada subsequente frame do nosso video, aplicamos a etapa 1 de calcular os centróides de cada objeto. Porem aqui, invês de atribuir um ID unico a cada objeto detectado (que iria contrariar o sentido de rastrear um objeto), precisamos determinar primeiro se podemos acociar novos centróides de objetos (pontos amarelos) a aos centróides antigos (pontos roxos). Para este processo, nós calculamos a distância euclediana entre cada par de objetos existente e novos objetos.
Mas como vamos usar as distâncias Eucledianas entre estes pontos para combiná-los e associá-los?



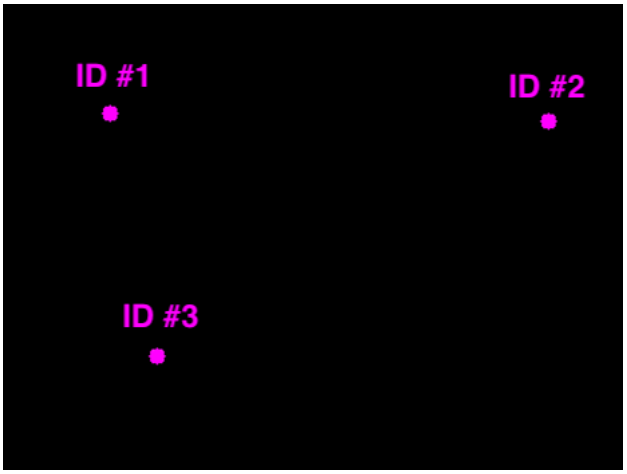
3- Atualizar coordenadas para objetos existentes

A supozição primária do algoritmo de rastreamento de centroides de objetos é de que o objeto irá se potencialemente mover entre frames subsequentes, porem a distância entre centróides será menor que outras distâncias entre objetos.
Portanto, se escolhermos associar centróides com a menor distância entre frames subsequentes, podes construir o nosso rastreador de objetos.
Mas e se surgir um novo objeto?



4- Registar novos objetos

No caso de obtermos mais objetos detetados do que objetos a ser rastreados, precisamos registar o novo objeto. Registar o novo objeto na lista de objetos a ser rastreados significa que:
- Atribuimos um novo ID
- Guardamos o centróide da caixa delimitadora para esse objeto
Podemos ir até a etapa 2 e repetir o processo de etapas em cada frame de video.



ref: https://www.pyimagesearch.com/2018/07/23/simple-object-tracking-with-opencv/

2.4.2 Hungarian Algorithm (Kuhn-Munkres)

A Hungarian algorithm can tell if an object in current frame is the same as the one in previous frame. It will be used for association and id attribution

https://towardsdatascience.com/computer-vision-for-tracking-8220759eee85

2.4.3 Kalman Filter

A Kalman Filter is an algorithm that can predict future positions based on current position. It can also estimate current position better than what the sensor is telling us. It will be used to have better association.

https://towardsdatascience.com/computer-vision-for-tracking-8220759eee85

2.5 Citações e Referências Cruzadas – [RETIRAR DA VERSÃO FINAL]

Para se referenciarem outras secções, usar \ref{label}, e.g., para citar a secção da Introdução deste capítulo, usar \ref{chap2:sec:intro}. O resultado é: a secção 2.1 contém a introdução deste capítulo.

Para se citarem fontes bibliográficas, colocar a entrada certa no ficheiro bibiografia.bib e usar o comando \cite{label-da-referencia}, ligando o comando com a palavra que o antecede com um til. Por exemplo, para citar a referência eletrónica *The Not So Short Introduction to LaTeX* [1], deve incluir-se o trecho seguinte no ficheiro bibiografia.bib e usar \cite{short} para a citação (citação incluída nesta mesma frase):

```
@MISC{short,
 author = {Tobias Oetiker and Hubert Partl and Irene Hyna and Elisabeth Schlegl},
 title  = "{The Not So Short Introduction to \LaTeX{}}",
 year   = 2018,
 note   = {[Online] \url{https://tobi.oetiker.ch/lshort/lshort.pdf}.
        Último acesso a 12 de Março de 2019}
}
```

2.6 Secções Intermédias

2.7 Conclusões

Cada capítulo intermédio deve referir o que demais importante se conclui desta parte do trabalho, de modo a fornecer a motivação para o capítulo ou passos se-

guintes.

Capítulo 3

Tecnologias e Ferramentas Utilizadas

3.1 Introdução

Cada capítulo intermédio deve começar com uma breve introdução onde é explicado com um pouco mais de detalhe qual é o tema deste capítulo, e como é que se encontra organizado (i.e., o que é que cada secção seguinte discute).

3.2 Python

Python é a linguagem de programação usada para o desenvolvimento deste projeto

3.3 Secções Intermédias

A tabela 3.1 serve apenas o propósito da exemplificação de como se fazem tabelas em LaTeX.

campo 1	**campo 2**	**campo 3**
14	15	16
13	13	13

Tabela 3.1: Esta é uma tabela de exemplo.

3.4 Conclusões

Cada capítulo intermédio deve referir o que demais importante se conclui desta parte do trabalho, de modo a fornecer a motivação para o capítulo ou passos seguintes.

Capítulo 4

Implementação e Testes

4.1 Introdução

Cada capítulo intermédio deve começar com uma breve introdução onde é explicado com um pouco mais de detalhe qual é o tema deste capítulo, e como é que se encontra organizado (i.e., o que é que cada secção seguinte discute).

4.2 Secções Intermédias

O trecho de código seguinte mostra a função main() e o seu funcionamento:

```
#include <stdio.h>

int main(){
  int i = 0;
  for(i = 0; i < 100; i++)
    printf("%d\n",i);
}
```

Excerto de Código 4.1: Trecho de código usado no projeto.

Se quiser definir a distribuição de Pareto, posso colocar a fórmula *inline*, da seguinte forma $P(x) = \frac{x_i^{1/\Lambda}}{2}$, ou numa linha em separada, como se mostra a seguir:

$$y^2 = \sum_{x=0}^{20} (x^3 - 2x + 3).$$

Outra maneira, mas numerada, é usar o ambiente equation, como se mostra na (4.1):

$$y^2 = \sum_{x=0}^{20} (x^3 - 2x + 3). \tag{4.1}$$

$$2+2+2+2+2+2+2+2+2+2+y^2 = \sum_{x=0}^{20}(x^3-2x+3); \tag{4.2}$$

$$= x^4 - 2. \tag{4.3}$$

4.3 Conclusões

Cada capítulo intermédio deve referir o que demais importante se conclui desta parte do trabalho, de modo a fornecer a motivação para o capítulo ou passos seguintes.

Capítulo 5

Conclusões e Trabalho Futuro

5.1 Conclusões Principais

Esta secção contém a resposta à questão:
Quais foram as conclusões princípais a que o(a) aluno(a) chegou no fim deste trabalho?

5.2 Trabalho Futuro

Esta secção responde a questões como:
O que é que ficou por fazer, e porque?
O que é que seria interessante fazer, mas não foi feito por não ser exatamente o objetivo deste trabalho?
Em que outros casos ou situações ou cenários – que não foram estudados no contexto deste projeto por não ser seu objetivo – é que o trabalho aqui descrito pode ter aplicações interessantes e porque?

Bibliografia

[1] Tobias Oetiker, Hubert Partl, Irene Hyna, and Elisabeth Schlegl. The Not So Short Introduction to LaTeX, 2018. [Online] https://tobi.oetiker.ch/lshort/lshort.pdf. Último acesso a 12 de Março de 2019.