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

Conteúdo

Lista de Figuras

Lista de Tabelas

Acrónimos

Capítulo 1

Estado da Arte

1.1 Introdução

O reconhecimento e rastreamento de objetos é um dos campos mais atuais da visão computacional.
É possível reconhecer diferentes objetos em uma cena através das suas formas, cor e diferentes características do objeto. É também possível rastrear um objeto em um fluxo de vídeo através do calculo de distâncias, comparação de formas, e outras características.
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 - **Algoritmos de reconhecimento de objetos** - Esta secção apresenta vários algoritmos estudados sobre a deteção/reconhecimento de objetos e uma breve explicação do seu funcionamento.

2. Secção 2.2 - **Algoritmos de rastreamento de objetos** - Esta secção apresenta vários algoritmos estudados sobre o rastreamento de objetos, e uma breve explicação do seu funcionamento.

3. Secção 2.3 - **Conclusões** - Esta secção apresentará uma breve conclusão sobre este capitulo.

1.2 Algoritmos de reconhecimento de objetos

A deteção de objetos combina as tarefas de localização e classificação de objetos. Corresponde a localizar um objeto em uma imagem e adivinhar que objeto é esse, como vemos na figura **??**. Em vez de prevermos a classe de objeto a partir de uma imagem, agora temos de prever a classe e também um rectângulo, chamada caixa delimitadora, contendo esse objeto.

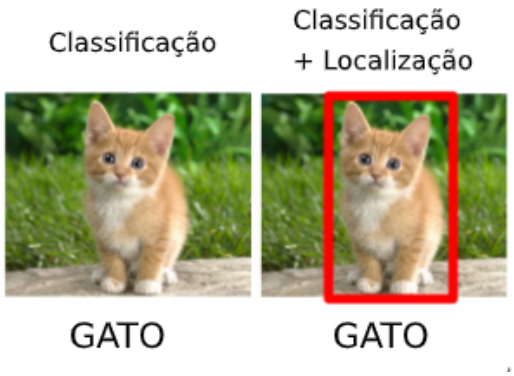

Figura 1.1: Ilustração de classificação e localização de um objeto/pessoa em uma imagem.

São necessárias 4 variáveis para identificar exclusivamente um retângulo. Portanto, para cada instância do objeto na imagem, preveremos as seguintes variáveis:

- nome_da_classe
- caixa_delimitadora_topo_esquerda_coordenada_x
- caixa_delimitadora_topo_esquerda_coordenada_y
- caixa_delimitadora_altura
- caixa_delimitadora_largura

Nas subsecções abaixo abordarei alguns algoritmos de deteção de objetos populares. Historicamente, tem havido muitas abordagens para deteção de objetos a partir de Haar cascades, proposta por Viola e Jones em 2001. No entanto, irei focar nos métodos de ponta que usam Redes Neurais e Aprendizagem Profunda.

Os detectores 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 **R-CNN!** (**R-CNN!**) é uma das mais famosas.
- Redes que prevêem caixas delimitadoras e classificações de classes de uma só vez, onde **YOLO!** (**YOLO!**) e **SSD!** (**SSD!**) são famosas arquiteturas.

1.2.1 Overfeat

A primeira rede neural profunda para deteção de objetos foi **Overfeat**. Eles introduziram uma abordagem de janela deslizante de várias proporções, como mostra na figura **??**, e usando uma **CNN!** (**CNN!**), mostraram que a deteção de objetos também melhorou a classificação de imagens.

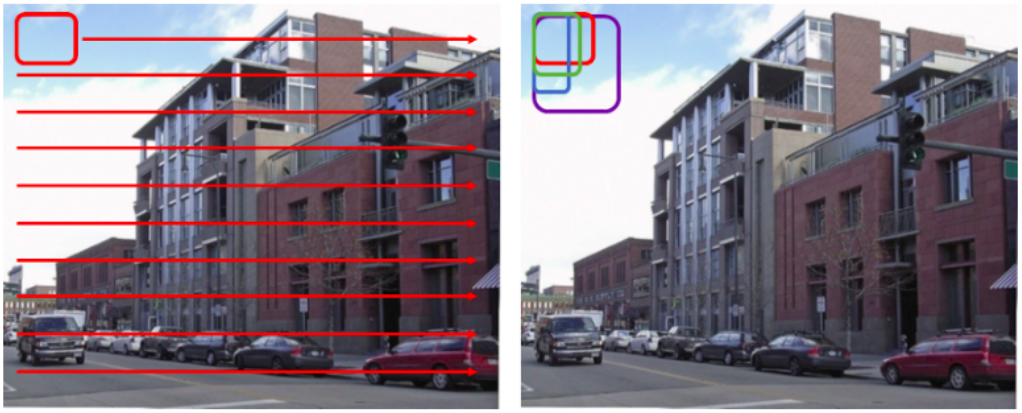

Figura 1.2: Ilustração de uma janela deslizante (direita) com diferentes proporções (direita).

CNN é uma rede neural que consiste em várias camadas diferentes, como a camada de entrada (input), pelo menos uma camada escondida e uma camada de saída (output). Esta é usada para reconhecer padrões como pontas, vertical/horizontal, formas, cores e texturas.

1.2.2 R-CNN

O modelo anterior foi logo seguindo pelo **R-CNN!**: Regiões com características **CNN!**. As **CNN!**s são demasiado lentas e dispendiosas, por isso é impossível correr uma **CNN!** em tantos caminhos gerados por um detector de janela deslizante. Então a **R-CNN!** resolve este problema usando um algoritmo de propostas de objetos chamado de **Pesquisa seletiva** que reduz o numero de caixas delimitadoras que são alimentadas no classificador para fechar em 2000 propostas de região. A Pesquisa seletiva usa dicas locais como textura, intensidade, cor e/ou a medida de incidências, etc, para gerar todas as propostas de localização do objeto. Cada região é alimentada em uma **CNN!**, que produziu um vetor de característica de grande dimensão. Este vetor é depois usado para a classificação final e regressão da caixa delimitadora, como mostra na figura **??**.

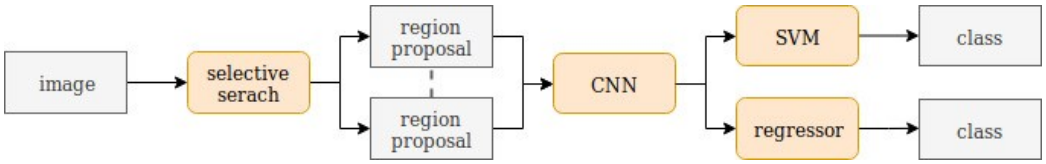

Figura 1.3: A arquitetura padrão de uma rede **R-CNN!** consiste em um método de proporção regional. A classificação final é feita com uma **SVM!** e uma regressão.

SVM! é um conjunto de métodos de aprendizado supervisionado que analisam os dados e reconhecem padrões, usando classificação e análise de regressão.

Então, existem 3 importantes etapas da **R-CNN!**:

1. Realizar pesquisa seletiva para gerar prováveis objetos.
2. Alimentar esses caminhos em uma **CNN!**, seguidos de uma **SVM!** para prever a classe de cada caminho.
3. Optimizar os caminhos, treinando as caixas delimitadoras separadamente.

1.2.3 Fast R-CNN

Uma abordagem mais sofisticada, o **Fast R-CNN!** também gera proporções regionais com pesquisa seletiva, 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 directamente em um mapa característico por **RoI!** (**RoI!**) pooling. Os vetores característicos agrupados são alimentados em uma rede totalmente conectada para classificação e regressão, como retratado na figura **??**. Similar ao **R-CNN!**, o Fast **R-CNN!** gera as proporções regionais usando pesquisa seletiva.

RoI! pooling, ou agrupamento 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 próxima etapa de deteção do objeto.

1.2.4 Faster R-CNN

A parte mais lenta da arquitetura Fast **R-CNN!** é a pesquisa selectiva e as caixas de borda. Então a arquitetura **Faster R-CNN!** substitui a pesquisa seletiva por

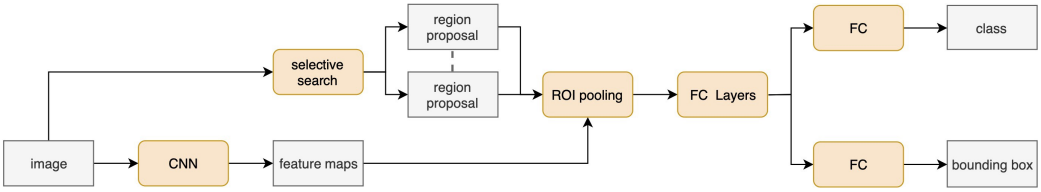

Figura 1.4: A arquitetura padrão da rede Fast-**R-CNN!**. A proporção regional é gerada usando pesquisa seletiva, mas agrupada diretamente no mapa de características, seguida de várias camadas FC para a classificação final e regressão da caixa delimitadora.

uma pequena rede convolucional chamada rede de propostas por região (**RPN!** (**RPN!**)) para gerar regiões de interesse.
Para lidar com as variâncias nas proporções e escala dos objetos, Faster **R-CNN!** introduz a ideia de **caixas âncora**. Em cada localização, o documento original usa 3 tipos de caixas âncora para as escalas 128x128, 256x256 e 512x512. Da mesma forma, para as proporções de 1:1, 1:2 e 2:1. Portanto, em cada local, temos 9 caixas nas quais a **RPN!** prevê a probabilidade de ser segundo plano ou primeiro plano. Aplicamos a regressão da caixa delimitadora para melhorar as caixas âncora em cada local. Portanto, a **RPN!** fornece-nos caixas delimitadoras de vários tamanhos com as probabilidades correspondentes de cada classe. A rede restante é semelhante à arquitetura Fast **R-CNN!**.

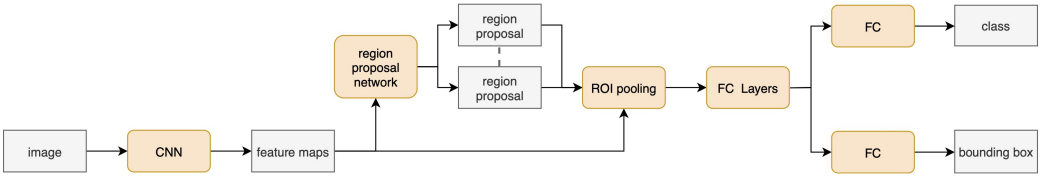

Figura 1.5: A arquitetura padrão da rede Faster-**R-CNN!**, onde a proposta por região é gerada usando uma **RPN!**, que trabalha directamente com o mapa de características. As ultimas camadas estão totalmente conectadas para classificação e regressão das caixas delimitadoras.

1.2.5 YOLO

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.
Este modelo divide a imagem em uma grelha S x S, e cada célula prevê N caixas delimitadoras e confiança. A confiança reflete a precisão da caixa delimitadora e

se a caixa delimitadora realmente contem o objeto (independentemente da classe). **YOLO!** também prevê a pontuação da classificação para cada caixa de cada classe de treinamento. Você pode combinar as duas classes para calcular a probabilidade de cada classe estar presente em uma caixa prevista.

Portanto, são previstas S x S x N caixas. No entanto, muitas dessas caixas tem pontuações baixas de confiança e, se definirmos um limite, digamos que a 30% de confiança, podemos remover a maioria deles, como mostrado no exemplo **??**.

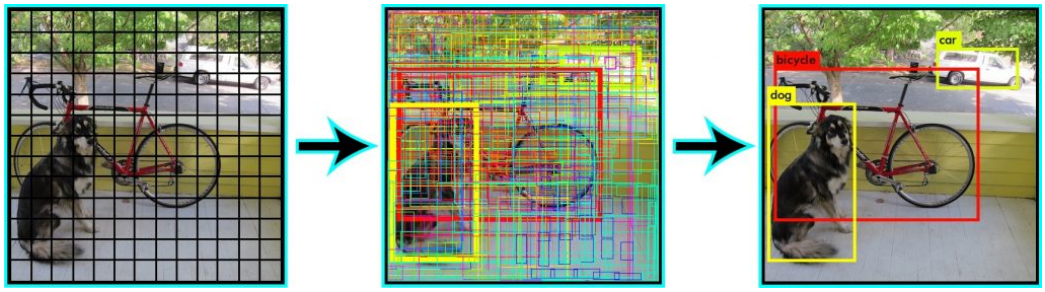

Figura 1.6: Exemplo do funcionamento da arquitetura **YOLO!**.

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.

1.2.6 SSD

Outra rede que prevê classes e caixas delimitadoras de uma vez é a **SSD!**. Esta executa uma rede convolucional em uma imagem apenas de entrada e calcula um mapa de recursos. Aqui executamos um pequeno núcleo convolucional de tamanho 3 x 3 neste mapa de recursos para prever as caixas delimitadoras e a probabilidade de classificação.
A rede **SSD!** também usa caixas âncora em várias proporções semelhantes ao Faster **R-CNN!** e aprende o deslocamento em vez de aprender a caixa. Para lidar com a escala, a **SSD!** prevê caixas delimitadoras após várias camadas convolucionais. Como cada camada convolucional opera em uma escala diferente, ela é capaz de detectar objetos de várias escalas.

É comparável com o **YOLO!**, mas esta usa várias proporções por célula da grelha e mais camadas convolucionais para melhorar a precisão.

1.3 Algoritmos de rastreamento de objetos

O rastreamento de objetos é o processo de localizar objetos em movimento ao longo do tempo em vídeo. Este é um problema antigo e difícil da visão computacional. Existem várias técnicas e algoritmos que tentam resolver esse problema de várias maneiras diferentes. No entanto, a maioria baseia-se em duas coisas principais:

- **Modelo de movimento**
  Um dos principais componentes de uma bom rastreador é a capacidade de entender e modelar o movimento do objeto. Assim, é desenvolvido um modelo de movimento que captura o movimento dinâmico de um objeto. Ele prevê a posição potencial dos objetos nos quadros (frames) seguintes, reduzindo assim o espaço de pesquisa. No entanto, o modelo de movimento por si só pode falhar em cenários em que o movimento é causado por coisas que não estão em um vídeo ou direcção abrupta e mudança de velocidade.

- **Modelo de Aparência Visual**
  A maioria dos rastreadores altamente precisos precisa entender a aparência do objeto que eles estão rastreando. Mais importante, eles precisam aprender a discriminar o objeto do plano de fundo. Nos rastreadores de um único objeto, apenas a aparência visual pode ser suficiente para rastrear o objeto através dos quadros, enquanto nos rastreadores de vários objetos, apenas a aparência visual não é suficiente.

Existem várias características para classificar um rastreador de objetos, como por exemplo:

- **Rastreadores com base na deteção**
  Os quadros de vídeo consecutivos são dados a um detector de objetos pré treinado que fornece uma hipótese de deteção, que por sua vez, é usada para formar trajetórias de rastreamento. Novos objetos são detectados e objetos que desaparecem são finalizados automaticamente. Nesta abordagem, o rastreador é usado para os casos em que a deteção de objetos falha. Em outra abordagem, o detector de objetos é executado para nada N quadros e as previsões restantes são feitas usando o rastreador.

- **Rastreamento sem deteção**
  O rastreamento sem deteção requer inicialização manual de um número fixo de objetos no primeiro quadro. Em seguida, localiza esses objetos nos quadros subsequentes. Esta abordagem não pode lidar com o caso em que novos objetos aparecem nos quadros subsequentes.

- **Acompanhamento de um único objeto**
  Apenas um único objeto é rastreado, mesmo que o ambiente tenha vários objetos. O objeto a ser rastreado é determinado pela inicialização do primeiro quadro.

- **Rastreamento de Objetos múltiplos**
  Todos os objetos presentes são rastreados ao longo do tempo. Se um rastreador baseado em deteção for usado, ele poderá rastrear novos objetos que surjam no meio do vídeo.

- **Rastreamento offline**
  Rastreadores offline são usados quando você precisa rastrear um objeto em um fluxo de gravado. Neste caso, podemos não apenas usar os quadros passados, mas também os futuros quadros para fazer previsões de rastreamento mais precisas.

- **Rastreamento online**
  Rastreadores online são usados onde as previsões estão disponíveis imediatamente, e portanto, estes não podem usar quadros futuros para melhorar os resultados.

- **Rastreamento de aprendizagem online**
  Estes rastreadores geralmente aprendem sobre o objeto a ser rastreado usando o quadro de inicialização e alguns quadros subsequentes. Portanto, esses rastreadores são mais gerais, porque você pode simplesmente desenhar uma caixa delimitadora em torno de um objeto e rastreá-lo.

- **Rastreamento de aprendizagem offline**
  O treinamento destes rastreadores acontece apenas offline. Ao contrário dos rastreadores de aprendizado online, estes rastreadores não aprendem nada durante o tempo de execução. Estes rastreadores aprendem sobre os conceitos completos offline, ou seja, podemos treinar um rastreador para identificar pessoas. Estes rastreadores podem ser usados para rastrear continuamente todas as pessoas em um fluxo de vídeo.

Muitos algoritmos de rastreamento tradicionais (não baseados em aprendizagem profunda) estão integrados na **API!** (**API!**) de rastreamento do **OpenCV!** (**OpenCV!**). A maioria destes algoritmos não é muito precisa comparativamente. No entanto estes podem vir a ser úteis para executar em um ambiente de com restrições de recursos, como é o caso do Raspberry Pi. No entanto, os rastreadores de baseados em aprendizagem profunda estão agora muito à frente dos rastreadores tradicionais em termos de precisão. Então, nos pontos a seguir vou falar de

alguns dos algoritmos mais populares no rastreamento de objetos, baseados em **IA!** (**IA!**), como é o caso do **GOTURN!** (**GOTURN!**), **MDNet!** (**MDNet!**) e do **ROLO!** (**ROLO!**), bem como dos tradicionais, como é o caso do **KCF!** (**KCF!**).

1.3.1 GOTURN

A **GOTURN!** é um rastreador de aprendizado offline com base em uma **CNN!**. Este rastreador é treinado usando vídeos de rastreamento geral de objetos. Este rastreador pode ser usado para rastrear a maioria dos objetos sem nenhum problema, mesmo que esses objetos não fizessem parte do conjunto de treinamento. Este algoritmo está integrado na biblioteca **OpenCV!**.

1.3.2 MDNet

A **MDNet!** é um rastreador de treinamento online baseado em **CNN!**s. O objetivo da **MDNet!** é acelerar o treinamento para fornecer resultados em tempo real. A estratégia é dividir a rede em duas partes. A primeira parte atua como um extrator de recursos genéricos que treina em vários conjuntos de treinamento e aprende a distinguir objetos de seus antecedentes. A segunda parte é treinada em um conjunto de treinamento especifico e aprende a identificar objetos nos quadros de vídeo.

A **MDNet!** possibilita a modificação dos os pesos das ultimas camadas da CNN durante o treinamento, reduzindo significativamente o tempo de computação.

1.3.3 ROLO

O **ROLO!** é um algoritmo de rastreamento baseado em deteção, online, e com um único objeto. Este usa a rede YOLO para deteção de objetos e um rede **LSTM!** (**LSTM!**) para encontrar a trajectória do objeto de destino. Há duas razões pelas quais **LSTM!** com **CNN!** é uma combinação mortal:

- As redes **LSTM!** são particularmente boas em aprender padrões históricos, sendo particularmente adequadas para o rastreamento de objetos visuais.
- As redes **LSTM!** não são muito caras em termos computacionais, por isso é possível criar rastreadores do mundo real muito rápidos.

O diagrama **??** fornece-nos o seguinte entendimento. Os quadros de entrada passam pela rede **YOLO!**. Nesta rede, são obtidas duas saídas diferentes (Recursos de imagem e coordenadas da caixa delimitadora). Estas duas saídas são fornecidas à rede **LSTM!**. O **LSTM!** gera trajetórias, isto é, a caixa delimitadora

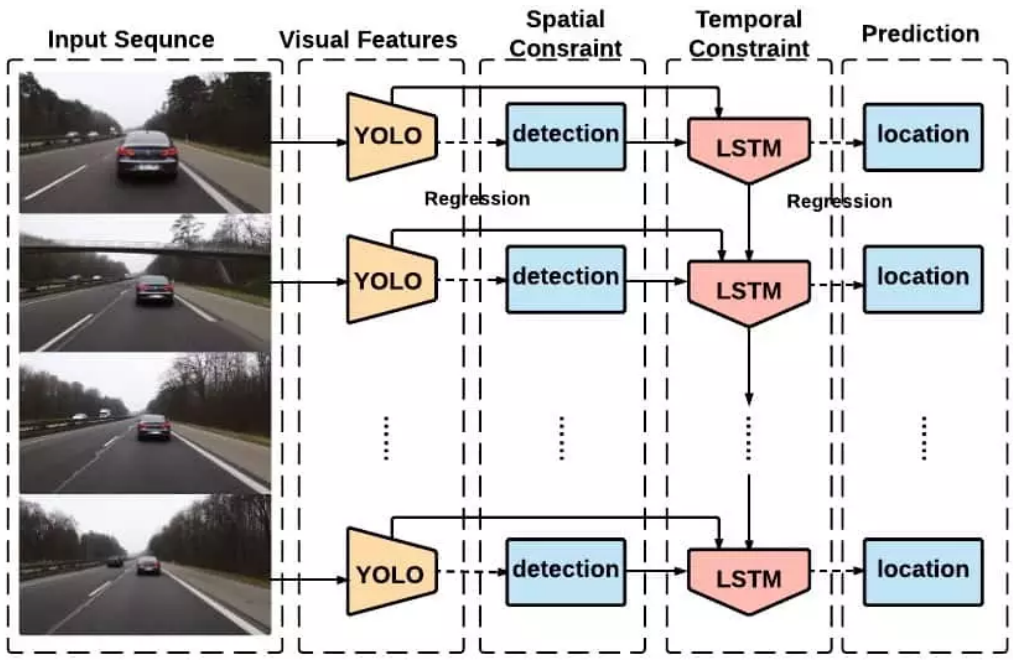

Figura 1.7:

- **YOLO! INPUT**: Quadros de entrada brutos.
- **YOLO! OUTPUT**: Vetor de recursos de quadros de entrada e coordenadas da caixa delimitadora.
- **LSTM! INPUT**: Junção dos recursos da imagem e coordenadas da caixa delimitadora.
- **LSTM! OUTPUT**: Coordenadas da caixa delimitadora do objeto a ser rastreado.

do objeto a ser rastreado. A inferência preliminar de localização (do **YOLO!**) ajuda o **LSTM!** a prestar atenção a certos elementos visuais. O **ROLO!** explora a história espaço-temporal, ou seja, juntamente com o histórico de localização, o **ROLO!** também explora a história dos recursos visuais. Mesmo quando a detecção do **YOLO!** falha devido a desfoque de movimento, o rastreamento de **ROLO!** permanece estável.

1.3.4 KCF

A ideia básica do **KCF!** é estimar um filtro de imagem ideal, de modo a que a filtragem com a imagem de entrada produza a resposta desejada. Dado um conjunto inicial de pontos, o **KCF!** tenta calcular o movimento destes pontos observando a direcção da mudança no próximo quadro. Por isso este é um rastreador offline, que apenas funciona em vídeos gravados. Em cada quadro consecutivo, tentamos procurar o mesmo conjunto de pontos na vizinhança. Depois que as novas posições desses pontos são identificadas, podemos mover a caixa delimitadora sobre o novo conjunto de pontos. Há matemática envolvida para tornar a pesquisa mais rápida e eficiente.

1.3.5 Centroid

O **rastreamento de centroides** é um algoritmo presente na biblioteca **OpenCV!**, que depende da distância euclidiana[1] entre centroides de objetos (objetos que o rastreador de centroide já viu antes) e novos centroides de objetos entre quadros subsequentes em um vídeo. Neste algoritmo temos de usar um detector de objetos em cada quadro de vídeo e determinar os centroides de cada objeto. Assumimos aqui que no próximo quadro o centroide do objeto estará a uma curta distância no centroide no quadro anterior, e assim assumimos que seja o mesmo objeto através da distância. Este algoritmo não é muito eficiente em ambientes com mais que um objeto.

1.4 Conclusões

Neste capitulo foram descritos alguns dos algoritmos de deteção e rastreamento mais populares e eficientes. Podemos concluir que apesar da deteção e rastreamento de objetos ser uma ser ainda uma temática bastante recente teve já um grande avanço. Apesar da sua dificuldade e do seu nível computacional, cada vez existem mais algoritmos para melhorarem e tornarem o reconhecimento e rastreamento de objetos mais preciso e rápido.
Existem inúmeros algoritmos de detectores de objetos em imagens e de rastreadores de objetos em imagem. Em este capitulo foquei me em alguns dos mais populares e eficientes. Estes possuem na maioria versões mais leves em termos de peso computacional deles mesmos, versões que terei um especial foco no futuro devido ao ambiente de desenvolvimento, que é um Raspberry Pi. Foquei-me também em versões mais atuais, sendo a maioria baseadas em Aprendizagem Profunda e Redes Neurais.

[1] Distância entre dois pontos