

Comparação de imagens em máquinas de self-checkout

Miguel Moreira¹ e Piedade Carvalho²

¹ Instituto Superior de Engenharia do Porto, Portugal
1171280@isep.ipp.pt

² Instituto Superior de Engenharia do Porto, Portugal
pbc@isep.ipp.pt

Resumo: Este trabalho surgiu na tentativa de solucionar um problema surgido pelas grandes mudanças nos últimos anos, verificadas no mercado do retalho, com grandes retalhistas a adotarem serviços de self-checkout. Neste tipo de serviço, os consumidores processam as suas próprias compras e realizam os pagamentos sem o auxílio dos operadores de caixa. No entanto, a segurança com que os produtos passam por esses serviços não é ideal. Alguns serviços como os de self-checkout deparam-se com problemas como fraudes no momento de *scanning* dos produtos. Como resultado desse problema, surgiu a necessidade de uma solução, que recorrendo à comparação dos produtos com imagens do suposto produto, se verifique se o produto corresponde ao faturado.

Este artigo visa uma introdução à computação visual para a resolução deste problema.

Palavras-chave: Reconhecimento de Imagens, Comparação de Imagens, Self-Checkout, NodeJS.

1 Introdução

Com o constante avanço tecnológico, começaram a surgir alternativas no funcionamento das operações no mercado do retalho. Por exemplo, foram adotados serviços de *self-checkout* em vários retalhistas, de modo a que fosse possível agilizar o processo da compra de produtos.

No entanto, a adoção desta alternativa introduziu também problemas, como a existência de fraudes no momento do processamento da compra ao realizar uma operação de *scanning* e o produto não ser compatível com o que foi faturado.

Assim, de modo a tentar resolver os problemas associados aos serviços de *self-checkout*, foi desenvolvida uma aplicação que tem por base a comparação de imagens.

O presente capítulo faz uma abordagem ao estado atual de um ponto de vista tecnológico, sendo realizada uma introdução à visão computacional e como esta tecnologia pode impactar e solucionar alguns problemas no mercado do retalho.

1.1 Visão computacional

Visão computacional é um campo da inteligência artificial que tem como objetivo treinar computadores para que estes interpretem e entendam o mundo visual (Brownlee, J. 2019).

Usando todo o tipo de imagens de câmaras e vídeos, em conjunto com modelos de deep-learning, as máquinas podem identificar e classificar objetos corretamente — e, então, reagir ao que elas “veem” (SAS, 2020). A aplicação da visão computacional não é de todo desconhecida, existindo constantemente exemplos do uso no nosso quotidiano. Alguns exemplos de aplicações são os veículos autónomos, o reconhecimento facial, a realidade aumentada e na área da saúde, para encontrar regularidades (Mihajlovic, 2020).

O uso da visão computacional pode criar soluções a outros problemas, além dos referidos anteriormente, e foi vista como a melhor abordagem para solucionar o problema apresentado.

1.2 OpenCV na comparação de imagens

Inicialmente desenvolvida pela Intel, o OpenCV (Open Source Computer Vision) é uma biblioteca open source para o desenvolvimento de aplicações na área de visão computacional e machine learning para o processamento de imagens em tempo real (What is OpenCV? 2020).

Um dos objetivos do OpenCV é de proporcionar uma infraestrutura sólida de simples uso de visão computacional a todos que trabalham nestas áreas de modo a ajudar a criar aplicações de visão mais sofisticadas, avançar na pesquisa, conhecimentos e melhorar tecnologias relacionadas com a visão computacional e inteligência artificial.

Atualmente dispõe de mais de 2500 algoritmos otimizados avançados de visão computacional (Billingsley e Brett, 2015), entre eles algoritmos que podem ser usados para detetar e reconhecer faces, identificar objetos, ou encontrar imagens semelhantes a partir de uma base de dados, tornando-se assim uma opção válida na resolução do problema.

2 Proposta de solução

Em virtude das mudanças apresentadas na descrição do problema no setor do retalho e a falta de segurança que estes disponibilizam atualmente, pretendeu-se desenvolver uma solução que compare imagens de um produto que está a ser processado em *self-checkout* para confirmar se corresponde ao produto que o cliente comprou, com base em comparações com imagens do mesmo produto de uma base de dados.

A proposta de solução foi uma API em NodeJS baseado na comparação de imagens, tal como referido anteriormente. A imagem enviada à API corresponde a um produto de um retalhista captado pela camera da máquina de self-checkout, e deve ser comparada com uma base de dados de imagens desse mesmo produto de modo a retornar se o produto é, ou não, o mesmo.

A aplicação desenvolvida recebe por pedido *http* a imagem captada pela camera e respetivo código de barras lido pelo scanner. De seguida é estabelecida uma

comparação entre a imagem captada com todas as imagens do produto associadas ao código de barras.

Para que fosse possível realizar a comparação foi preciso recorrer aos seguintes algoritmos do OpenCV:

SIFT (Scale-Invariant Feature Transform): algoritmo necessário para detetar os pontos chave das imagens a serem comparadas (Tyagi, D. 2020) Foi selecionado este algoritmo em deterioramento de outros algoritmos de detecção de pontos-chave devido a este ser considerado o mais preciso entre os restantes, respondendo melhor a todos os tipos de transformações geométricas, tais como a rotação ou translação da imagem (Tareen e Saleem, 2018).

BFMatcher: algoritmo necessário para realizar o *matching* dos pontos chave realçados pelo algoritmo anterior que tentará todas as possibilidades (que é o significado de "Brute Force") e, portanto, encontrará as melhores correspondências ("BFMatcher Class Reference," 2020).

Resultado da execução dos algoritmos será retornado um valor correspondente à distância existente entre as duas imagens, onde quanto mais baixo o valor maior será a similaridade entre elas. Esse resultado foi convertido numa percentagem de modo a que posteriormente, na máquina de self-checkout, seja mostrada uma mensagem de alerta caso o método de comparação não considere o produto o mesmo.

Se o produto retornar mais de 75% de similaridade, valor arbitrário para que se considere o produto igual com base nos testes realizados, onde acima desse valor foi possível afirmar com certezas que a imagem captada corresponde ao mesmo produto, a imagem é guardada na base de dados sendo associada ao produto para aperfeiçoar resultados de futuras comparações.

Além da funcionalidade principal de comparar imagens, foi implementado na aplicação um método para apagar carregadas na base de dados de modo a não sobrecarregar a base de dados com imagens mal captadas que não consigam estabelecer qualquer comparação com as restantes associadas aos produtos devido a qualquer problema na captura, como por exemplo, uma má focagem.

3 Resultados

Após realização da aplicação foram realizados testes com imagens captadas com a camera de uma máquina, demonstração de resultados será usada uma comparação entre duas imagens do mesmo produto e uma comparação com duas imagens de produtos diferentes.

Nesta secção serão apresentadas duas das comparações realizadas e o resultado gerado pela aplicação.

A primeira comparação foi realizada com dois produtos diferentes, enviados à aplicação com o mesmo código de barras. A comparação entre a imagem da Fig. 1 e Fig. 2 resultou em 48% de similaridade.



Fig. 1. Imagem 1 da comparação

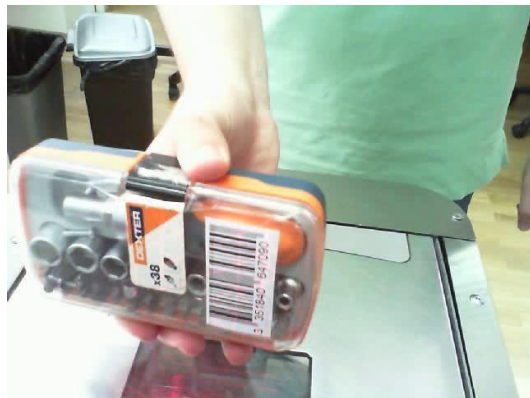


Fig 2. Imagem 2 da comparação

A segunda comparação foi realizada com dois produtos iguais, enviados à aplicação com o mesmo código de barras. A comparação entre a imagem da Fig. 3 e Fig. 4 resultou em 70% de similaridade.



Fig. 3. Imagem 1 da comparação

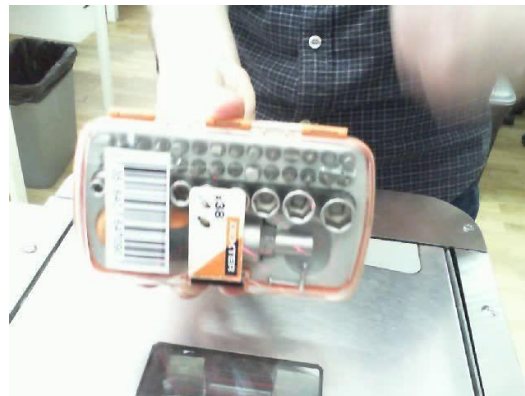


Fig. 4. Imagem 2 da comparação

Com base nos resultados destas duas comparações é possível verificar que ambas ainda são muito inflacionadas pelo ruído da imagem, pois na primeira comparação, onde os produtos eram diferentes a aplicação apresentou um resultado de 48% de confiabilidade e na segunda comparação apresentada, a aplicação retornou um valor de 70%.

A aplicação retornará um melhor resultado de comparação caso, por exemplo, a camera seja melhor, pois qualquer fator como o desfoque ou um movimento rápido pode afetar a captura de imagem e por consequente afetar a comparação.

4 Conclusão

A aplicação de comparação de imagens utilizando algoritmos de visão computacional desenvolvida pode apresentar-se como uma solução viável para o problema, no entanto devido ao curto tempo para a realização da aplicação visto existir complexidade no problema apresentado, este ainda demonstra algumas limitações, como o foco da captura no produto comprado, ou caso a embalagem do produto seja alterada. Limitações estas que serão possíveis de ultrapassar com os seguintes melhoramentos, que poderão ser realizados num trabalho futuro com base nesta aplicação.

Um exemplo desses melhorias, tal como referido anteriormente, pode ser o tratamento da imagem para focar apenas na comparação do produto, pois a imagem captada pelas cameras nas maquinas de self-checkout ainda é bastante influenciada pelo ruído da imagem, ou seja tudo o que não se queira comparar, e será necessário trabalhar na aplicação de comparação de modo a que esta ignore o máximo de ruído possível numa imagem.

Outro melhoramento passa pelo desenvolvimento de um método para detetar se um produto alterou a embalagem, que acontece várias vezes no mercado do retalho, e ao realizar essa alteração da embalagem afeta a comparação e por consequente a percentagem da confiabilidade da comparação que é retornada pela aplicação.

Referências

- [1] BFMatcher Class Reference. (2020). Retrieved from https://docs.opencv.org/3.4/d3/da1/classcv_1_1BFMatcher.html
- [2] Billingsley, J. e Brett, P. (2015). Machine vision and mechatronics in practice: Springer.
- [3] Mihajlovic, I. (2020). Everything You Ever Wanted To Know About Computer Vision. Retrieved from <https://towardsdatascience.com/everything-you-ever-wanted-to-know-about-computer-vision-heres-a-look-why-it-s-so-awesome-e8a58dfb641e>
- [4] SAS. (2020). Visão Computacional. Retrieved from https://www.sas.com/pt_br/insights/analytics/computer-vision.html
- [5] Tareen, S. A. K. e Saleem, Z. (2018). A comparative analysis of SIFT, SURF, KAZE, AKAZE, ORB, and BRISK.
- [6] What is OpenCV? (2020). Retrieved November 20, 2020, from <https://software.intel.com/content/www/us/en/develop/articles/what-is-opencv.html>
- [7] Tyagi, D. (2020). Introduction to SIFT(Scale Invariant Feature Transform). Retrieved November 20, 2020, from <https://medium.com/data-breach/introduction-to-sift-scale-invariant-feature-transform-65d7f3a72d40>
- [8] Brownlee, J. (2019). A Gentle Introduction to Computer Vision. Retrieved November 26, 2020, from <https://machinelearningmastery.com/what-is-computer-vision/>