Projeto Final - Relatório Final

SCC 0251 - Processamento de Imagens

GRUPO 4
Kelvin Guilherme de Oliveira - 9293286
Mauricio Caetano da Silva - 9040996

1. OBJETIVO PRINCIPAL

Este projeto tem como objetivo a construção de um software que seja capaz de realizar o reconhecimento do preço de um produto em um estabelecimento comercial, a partir de uma fotografia tirada por um celular.

A principal função desse software é facilitar o desenvolvimento de outras aplicações que necessitem deste serviço, como, por exemplo, uma aplicação que realize a soma total da compra de um cliente em um supermercado a partir de fotografias de tais preços, ou uma aplicação que diga ao cliente se o produto ao qual a foto foi tirada está caro ou não, entre outras aplicações.

2. IMAGENS DE ENTRADA

As imagens que utilizamos como entrada para nosso software são imagens fotografadas das etiquetas que contém o preço de um produto em um estabelecimento comercial.

Colocamos algumas restrições sobre as imagens para facilitar o desenvolvimento do software, são elas:

- A imagem deve conter apenas UMA etiqueta que corresponde ao preço de um produto. Imagens com mais de uma etiqueta ou preço não resultarão em valores válidos.
- A etiqueta do produto deve ser do formato retangular, e não deve estar sobreposta por outros objetos ou etiquetas na fotografia.
- As etiquetas devem ser das cores verde ou amarela. Quaisquer etiquetas de cores diferentes não serão encontradas.
- A foto deve ser tirada com o celular em um plano paralelo à etiqueta, ou seja, a etiqueta não pode ter apresentar uma noção perspectiva (Mais detalhes na seção 3.2.4).

Para a realização de testes e para o funcionamento correto da aplicação utilizamos imagens retiradas das prateleiras de alguns supermercados da cidade de São Carlos (São Paulo, Brasil), utilizando as câmeras dos celulares Asus Zenfone 5 e Motorola Moto G2.

As imagens foram retiradas em diferentes horários com diferentes condições de iluminação e angulação de foto em vários supermercados, com o objetivo de proporcionar melhor adaptação aos ambientes variados nos quais o software será submetido. A seguir temos alguns exemplos de imagens capturadas.

3. DESCRIÇÃO DOS PASSOS E DOS MÉTODOS UTILIZADOS

Como este projeto possui duas partes muito bem definidas e independentes, iremos explicar os passos realizados e explorar os resultados obtidos de forma separada, para depois, na seção 4 (Resultados Finais) apresentar a união das partes e a solução final deste projeto.

As duas partes são:

- Extração da etiqueta de preço a partir da imagem original.
- Extração do preço em si a partir de uma etiqueta.

3.1. EXTRAÇÃO DA ETIQUETA

A primeira parte do projeto tem como objetivo a localização da etiqueta de preço de produto em uma foto, para que, posteriormente, possamos realizar a extração de seu valor numérico utilizando o software de reconhecimento Tesseract.

O fluxo da extração da etiqueta de preço é formado pelos seguintes passos: Pré-processar a foto da prateleira com os produtos, realçando os contornos da etiqueta e excluindo ao máximo os contornos dos demais objetos, Extrair os objetos com maior probabilidade de serem a etiqueta, Extrair a etiqueta, copiando seu conteúdo para uma segunda imagem que será utilizada adiante. A seguir será explicado o passo a passo, as ideias e métodos utilizados, bem como os problemas encontrados em cada passo. O fluxo do processo é mostrado na imagem abaixo.

3.1.1. Pré-processamento da imagem da prateleira

Para realçar os contornos das etiquetas de preço a primeira etapa realizada foi a conversão do sistema de cores. Por padrão, a leitura de uma imagem na biblioteca OpenCv é realizada no sistema RGB, então convertemos seu sistema de RGB para HSV. Essa conversão é realizada com objetivo de facilitar a diferenciação da etiqueta do restante da imagem por meio de sua intensidade de cor (verde e amarela).

Após isso, uma máscara binária (preta e branca) de objetos com as cores interessadas é gerada utilizando valores de mínimo e máximo de intensidades. Utilizamos então uma outra função da OpenCv chamada “canny”, para a extração de arestas resultando em uma imagem de linhas de contorno. Essa imagem ainda passou por algumas operações morfológicas com objetivo de melhorar as linhas e dimunuir os ruídos.

3.1.2. Extração da etiqueta de preço

Tendo a imagem com as arestas dos objetos, uma função chamada “findContours” foi então utilizada para que os contornos dos objetos fossem destacados. Esses contornos foram então armazenados na memória, ordenados por área e então, retiramos uma amostra com os 200 maiores contornos.

Tendo essa amostra de contornos, utilizamos uma outra função chamada “approxPolyDP” para realizar a aproximação desses contornos por polígonos que o preencham com a menor área possível. Desse modo pudemos eliminar os polígonos que tenham mais do que 4 lados e que não estejam dentro de uma faixa de valores de área pré determinada por nós. A faixa de valores foi definida por meio da análise individual das etiquetas em cada imagem.

Nesse ponto a grande maioria de falsos positivos já foram eliminados e, se a etiqueta foi encontrada, esta será mostrada na tela.

3.1.3. Problemas na extração da etiqueta de preço

Nosso software não teve uma precisão tão alta na extração da etiqueta de preço devido, principalmente, a problemas na extração das arestas da etiqueta, pois são as arestas da imagem, a principal forma de se encontrar a etiqueta.

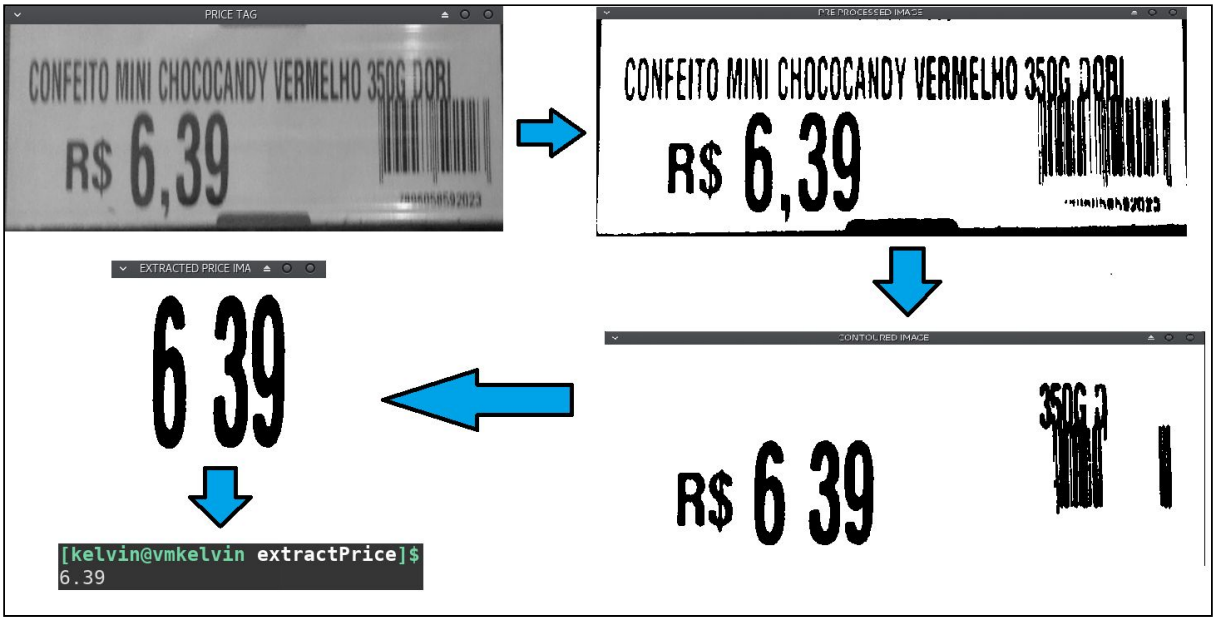
A abordagem de cores no sistema HSV trouxe uma boa melhoria para a precisão, porém erros acontecem quando objetivos que estão logo abaixo do limite inferior da etiqueta tem cores próximas do verde e amarelo, pois, desse modo, as linhas desse objeto se confundem com a etiqueta.

3.2. EXTRAÇÃO DO PREÇO

A segunda parte desse projeto consiste na extração do preço propriamente dito (ou seja, fazer com que o programa resulte em uma String contendo o valor do produto) a partir de uma imagem da etiqueta que contém o preço. Essa parte consiste em pegar as etiquetas extraídas das imagens originais (procedimento explicado na seção acima), aplicar alguns métodos de processamento de imagens para melhorar a visibilidade do preço de tal forma que o Tesseract (OCR, será explicado adiante) consiga reconhecê-lo e retornar a String desejada.

O fluxo da extração do preço é formado basicamente por: Pré-processar a imagem da etiqueta a fim de deixar o preço mais visível, Extrair os objetos com

maior probabilidade de pertencerem ao preço, Extrair os objetos que realmente compõem o preço, Enviar a imagem ao Tesseract OCR para receber a String desejada. A seguir será explicado o passo a passo, as ideias e métodos utilizados, bem como os problemas encontrados em cada passo. O fluxo do processo é mostrado na imagem abaixo.



3.2.1. Pré-processamento da imagem da etiqueta

A primeira etapa consistiu em preparar a imagem da etiqueta da melhor maneira possível para facilitar a extração do preço. Para tanto, a primeira ação realizada foi converter a imagem para escala de cinza (para aplicar os métodos desejados) e redimensionar todas as imagens para o tamanho 800x400, pois, devido ao fato que as etiquetas normalmente possuem um tamanho padrão, padronizamos o tamanho para posteriormente calcular medidas como tamanho dos objetos, distância entre objetos, etc.

Para realçar o preço e remover ruídos, algumas estratégias foram testadas e não tiveram o resultado desejado, tais como: Filtros para melhoramento da imagem (Gaussian Blur, Logaritmic), Filtro Adaptativo, entre outros. Mas o melhor método que causou a reação esperada foi o Threshold. Nesse método, temos diferentes abordagens que levam a resultados também diferentes. Foi tentado o THRESH_BINARY e o ADAPTIVE_THRESHOLDING, sem sucesso. A melhor abordagem encontrada foi o Threshold com THRESH_BINARY em conjunto com THRESH_OTSU (com parâmetros threshold = 127, max value = 255, Gerando uma imagem binária), cujo resultado é mostrado na imagem abaixo. Note que a abordagem realça muito bem o preço dentro da etiqueta.

3.2.2. Extração dos objetos mais prováveis

Essa etapa do processo tem como objetivo extrair os componentes da imagem que têm mais chances de pertencerem ao preço, ou seja, remover os componentes que certamente não são relacionados ao preço contido na etiqueta.

Para realizar tal objetivo, foi utilizada uma função fundamental no nosso projeto. A função “findContours” da biblioteca OpenCV. Tal função encontra os contornos em uma imagem binária e retorna as suas coordenadas. Isso foi extremamente útil para reconhecermos os formatos de cada componente da imagem e suas propriedades, como área, perímetro, entre outras.

Com posse de tais dados, pudemos extrair os componentes mais similares ao preço, descartando os demais. Tal seleção foi feita com base na área de cada componente (pois dados os contornos, podemos encontrar suas áreas internas com a função “contourArea” da biblioteca OpenCV). Analisando as imagens de entrada descritas na seção 2, pudemos notar que a área dos números que compõem o preço ficam em torno de 3000 a 7000 (inserimos uma margem de erro grande para evitar que números sejam descartados nessa etapa), sendo ‘1’ o numeral que possui a menor área, e alguns números como o ‘8’ e o ‘9’ sendo os de maiores áreas. Sendo assim, executamos um Threshold sobre os componentes, deixando apenas aqueles que possuam a área na faixa de valores que caracterizam os números.

Sendo assim, a imagem composta pelos elementos possíveis de serem o preço fica conforme a ilustração a seguir.

3.2.3. Extração dos objetos que compõem o preço

Tendo a imagem com os componentes possíveis de serem o preço, essa etapa tem como objetivo extrair, dentre estes componentes, os que mais são similares ao preço. Essa extração é realizada com base em novos filtros, que vão eliminando os componentes menos similares até chegar em um conjunto de componentes que possuam as características do preço.

O primeiro filtro é relacionado com o perímetro dos componentes (tal medida pode ser obtida por meio da função “arcLength” da biblioteca OpenCV). De forma similar à realizada na etapa anterior com a área, aqui observamos que o perímetro dos números fica entre os valores 300 e 700. Sendo assim, aplicamos um Threshold de forma que apenas os componentes que possuam perímetro nessa faixa de valores sejam selecionados, e os demais descartados.

O segundo filtro aplicado é relacionado com as dimensões dos componentes, mais precisamente com a altura e a largura. De forma similar, notamos que todos os

números têm altura maior que 80, e largura maior que 15. Sendo assim, aplicamos um threshold removendo os componentes que não se aplicam a regra. Depois disso, calculamos a variância de cada componente, com base nas alturas e larguras, pois, uma vez que os números possuem formato similar, essa medida é excepcional para nos mostrar quais são os componentes mais similares (aqueles cuja variância é mínima). Sendo assim, aplicamos o Threshold com base na variância, ficando apenas com os objetos que pertencem ao preço, e com outros que precisam ser similares, em formato, aos números.

Por fim, aplicamos um último filtro para extrair, de fato, o preço. Esse filtro está relacionado com a distância entre os elementos. Em primeiro lugar, calculamos as distâncias entre cada componente. A partir de um vetor de distâncias, verificamos a média das distâncias e eliminamos os componentes que mais fogem da média, até que a média dos componentes selecionados seja menor que 10000 (Valor calculado observando as imagens de entrada). Sendo assim, chegamos ao final do filtro, que em basicamente 95% das imagens testadas extrai perfeitamente apenas o preço, conforme ilustrado nas imagens a seguir.

3.2.4. Interação com o Tesseract OCR

Após todo o procedimento acima ter sido realizado, podemos então utilizar o Tesseract OCR para tentar obter a String contendo o preço. Mas essa parte do projeto teve alta dificuldade devido ao não entendimento sobre o funcionamento do Tesseract. Como mostrado nas imagens acima, os números estão perfeitamente visíveis e sem ruídos, fato este que implicaria no sucesso da interação com o OCR, o que nem sempre aconteceu.

Fizemos vários testes, e notamos que o Tesseract só reconhece números quando estes estão bem claros e com suas formas ideais. O que não conseguimos entender foi que para diversas imagens aparentemente perfeitas, o OCR não conseguiu reconhecer o número, não retornando nada na String.

Tentamos aplicar diversas abordagens para tentar fazer com que o Tesseract reconhecesse o preço, como por exemplo: Pegar número por número e enviar ao Tesseract, Aplicar operações morfológicas, Aplicar rotações, Filtros em geral, entre outras estratégias, sem o sucesso esperado.

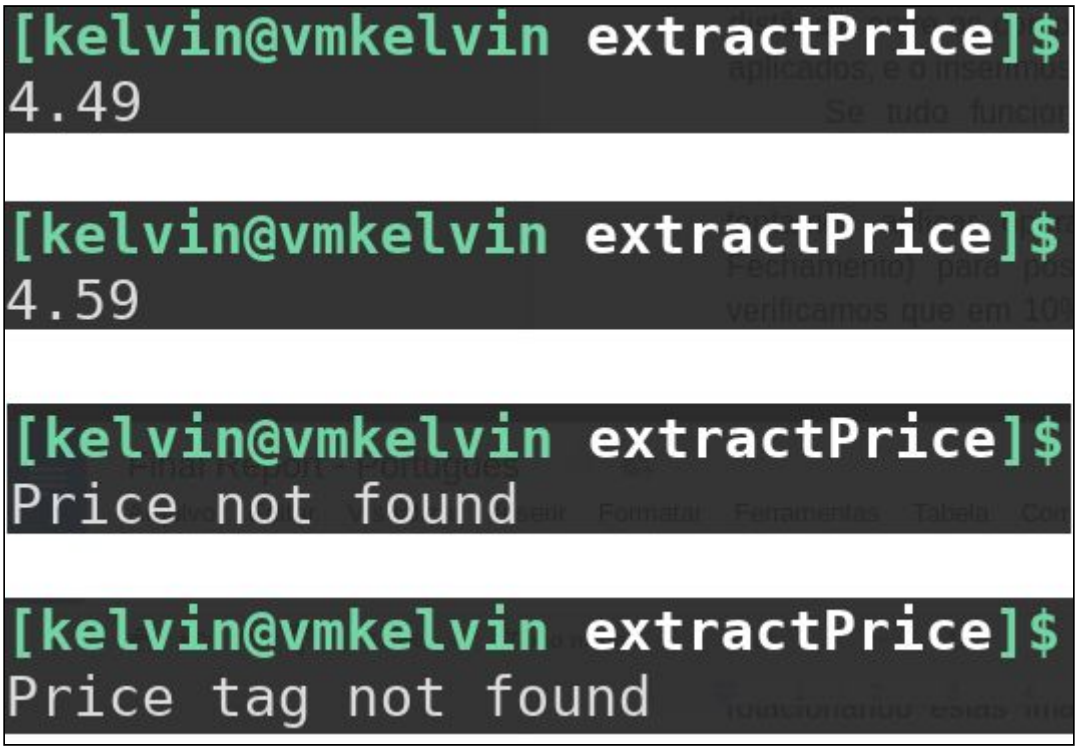
A principal semelhança entre a maioria das imagens que o Tesseract não reconheceu é que todas aparentam estar rotacionadas. Fizemos então um teste, rotacionando estas imagens de forma que ficassem alinhadas, sem sucesso, e depois pegamos as imagens que funcionam, e rotacionamos elas, para que ficassem com o ângulo de rotação similar ao das imagens que não funcionaram, e, para nossa surpresa, elas continuaram funcionando.

Supomos então, que a falha com o OCR se dá por conta da noção perspectiva da imagem, na qual os números ficam com uma noção tridimensional, uns mais distantes do plano da câmera do que os outros. Mas não conseguimos provar tal teoria e nem consertar as falhas nessas imagens.

Por fim, com as imagens que o Tesseract reconhecia, calculamos o lugar exato do ponto de separação do preço (parte inteira, parte decimal) por meio da distância entre os componentes do preço, uma vez que o ponto some com os filtros aplicados, e o inserimos no lugar correto da String.

Se tudo funcionou até aqui, temos a String desejada. Caso contrário, tentamos aplicar operadores morfológicos nas imagens (Erosão, Dilatação e Fechamento) para posteriormente enviar ao Tesseract. Com base em testes, verificamos que em 10% das imagens que o Tesseract não reconhecia, após essa aplicação, ele começa a reconhecer.

Sendo assim, finalizamos essa etapa, encontrando os resultados como mostrados nos exemplos abaixo.



4. RESULTADOS FINAIS

Como visto anteriormente, nosso software pode ser dividido em duas partes: uma para localização e extração da etiqueta e outra para extração do preço. A junção dessas duas partes obteve sucesso em pouco menos da metade das imagens retiradas (aproximadamente 40%). Isso ocorre por dois motivos basicamente: erros na localização da etiqueta de preço ou erros na extração do valor do preço (descritos anteriormente).

Esse valor de precisão pode ser melhorado principalmente no entendimento de como o software de reconhecimento Tesseract funciona, pois, muitas vezes, etiquetas de preço eram localizadas, porém os valores numéricos não. Além disso, traçar uma melhor estratégia de localização da etiqueta, como utilizar a ideia de regiões de interesse pode melhorar na precisão.

5. REFERÊNCIAS

Template matching
http://docs.opencv.org/2.4/doc/tutorials/imgproc/histograms/template_matching/template_matching.html

Feature matching
http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_matcher/py_matcher.html

Operações morfológicas
http://docs.opencv.org/2.4/doc/tutorials/imgproc/erosion_dilatation/erosion_dilatation.html

Detecção de objetos por cor
http://www.pyimagesearch.com/2014/08/04/opencv-python-color-detection/
http://www.pyimagesearch.com/2016/02/15/determining-object-color-with-opencv/

Tesseract
http://docs.opencv.org/trunk/d7/ddc/classcv_1_1text_1_1OCRTesseract.html
https://www.packtpub.com/mapt/book/application_development/9781785280948/10/ch10lvl1sec72/using-tesseract-ocr-library