Localização do eixo principal de resistores em uma cena

Mailson Rodrigues de Medeiros Guimarães*

Resumo

O presente trabalho tem como objetivo a implementação de um algoritmo de Processamento Digital de Imagens capaz de localizar um resistor elétrico em uma cena previamente montada, sendo capaz de localizar seu centroide e sua orientação. Com estas informações, o eixo principal do resistor, que passa por todas as faixas de cores que indicam o valor do componente, deve ser encontrado. Com isto, trabalhos futuros podem ser realizados para o processo de informar o valor do resistor baseado nos resultados obtidos no presente relatório.

Palavras-chave: algoritmo, Processamento Digital de Imagens, resistor, eixo

1 Introdução

Em tempos hodiernos, a tecnologia tem ganhado cada vez mais espaço na sociedade. Algumas dessas tecnologias surgem como forma de facilitar a vida humana. Nesta âmbito, pode-se citar as tecnologias associadas às técnicas de Processamento Digital de Imagens (PDI).

O PDI ocorre quando, em um sistema, se tem como entrada uma imagem, onde esta é tratada e tem-se como saída uma outra imagem [1]. As aplicações deste tipo de processo são amplas, desde simples utilidades casuais à utilização em grandes indústrias.

Dessa forma, o presente trabalho tem como objetivo apresentar uma aplicação do PDI, voltado para a identificação de resistores elétricos em uma cena montada. Resistores são componentes cuja grandeza (resistência) é dada por faixas de cores específicas presentes na superfícia do resistor. A ideia é localizar o eixo central dos componentes que passa por todas as faixas de cores e observá-las. Todo o processo deve ser feito de forma dinâmica e em tempo real, sendo capaz de identificar múltiplos resistores.

2 Materiais e métodos

Para tornar possível a realização do processo proposto, para a captura da imagem, foi utilizado um celular posicionado a uma certa altura em relação a uma folha de ofício

*Departamento de Engenharia Elétrica, UFRN. Email: rodriguez.1997@hotmail.com.

branca, sobre a qual eram colocados resistores de diversos valores. A figura abaixo ilustra o *setup* descrito:

FIGURA 1: *Setup* de aquisição de imagens.

As imagens coletadas pelo celular eram então transmitidas para um computador via *Wi-Fi* através do aplicativo *DroidCamX* para celulares com sistema operacional *Android*. Esta opção é uma alternativa na ausência de uma *WebCam* conectada diretamente ao computador.

Já no computador, para a realização do processamento das imagens capturadas pela câmera, foi utiliza a linguagem de programação *Python*, com o auxílio de bibliotecas como o *OpenCV* [3], voltada especificamente para o processamento de imagens, o *numpy*, para todo o tratamento numérico e o *matplotlib* para a exibição de gráficos com os resultados.

3 Resultados e discussões

Inicialmente, o desenvolvimento do algoritmo foi feito sobre uma imagem fixa de forma a se obter um passo a passo definido partindo da imagem recebida até a saída dos dados. O processo pode ser resumido no seguinte diagrama de blocos:

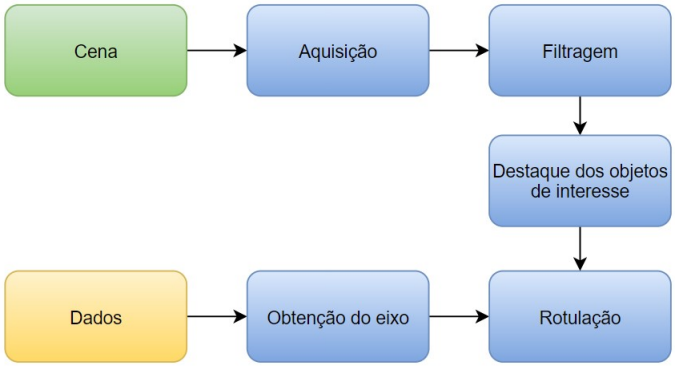

FIGURA 2: Diagrama de blocos do processo.

Cada bloco, a partir da aquisição anteriormente descrita, será explicado nas seções a seguir no presente trabalho.

3.1 Filtragem

Devido a iluminação presente na cena e a qualidade da câmera utilizada, além dos algoritmos de compressão que devem ser empregados para o *stream* do vídeo através da rede sem fio faz com que muito ruído apareça apareça na imagem a ser processada, além do problema de iluminação.

Para o processo em questão, o filtro que apresentou melhor resultado foi o Filtro da Mediana [2], com uma máscara de 5x5. O impacto do filtro pode ser visto abaixo:

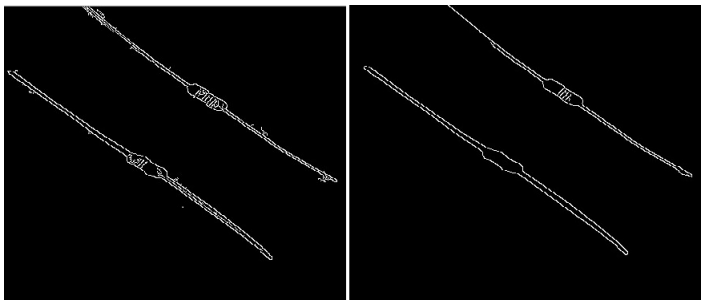

FIGURA 3: Atuação do filtro.

Na imagem da esquerda, percebe-se a presença de ruído na detecção de bordas quando o filtro está desligado. Após a utilização do filtro, este ruído some e tem-se uma borda mais definida. Vale ressaltar ainda que após o processo de filtragem, a imagem é normalizada para atenuar os problemas relacionados à iluminação. Outro impacto direto do processo de filtragem será mostrado em seções adiante.

3.2 Destaque dos objetos de interesse

Para se destacar apenas os objetos de interesse na cena, foi utilizado o algoritmo de *Canny* para a detecção das bordas dos resistores (Figura 3). Para uma calibração dinâmica dos resultados em tempo real, foram implementadas *trackbars* para configuração dos limiares do algoritmo de *Canny* e reduzir também o efeito do ruído e bordas falsas que podem surgir.

Com a obtenção das bordas, foi utilizado o algoritmo de dilatação, onde o *kernel* utilizado possuía dimensão 5x5 e era completamente preenchido com números 1. Além disso, este processo de passagem do *kernel* sobre a imagem foi realizado 3 vezes, tendo-se como resultado o seguinte:

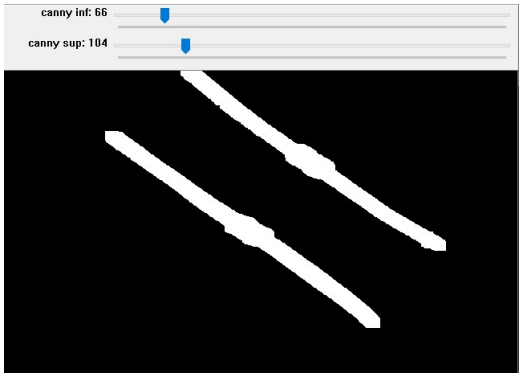

FIGURA 4: Dilatação das bordas.

Dessa forma, tem-se apenas os dois resistores destacados em toda a cena, facilitando assim os processos posteriores.

3.3 Rotulação

O processo de rotulação foi utilizado para separar os resistores da cena em diferentes categorias. O método utilizado para este processo foi realizado através de funções disponibilizadas pela biblioteca *OpenCV*. O primeiro passo foi encontrar, novamente, os contornos da imagem dilatada através de uma função chamada *findContours*.

Em seguida, fazendo o uso da função *moments*, é possível calcular a posição do centroide de todos os grupos de contornos que foram encontrados. Objetos com uma quantidade de pontos de contorno abaixo de um limiar definido são descartados para evitar que qualquer ruído que surja seja considerado um objeto.

Neste ponto, jé é possível saber a posição de cada resistor na cena, como mostrado abaixo:

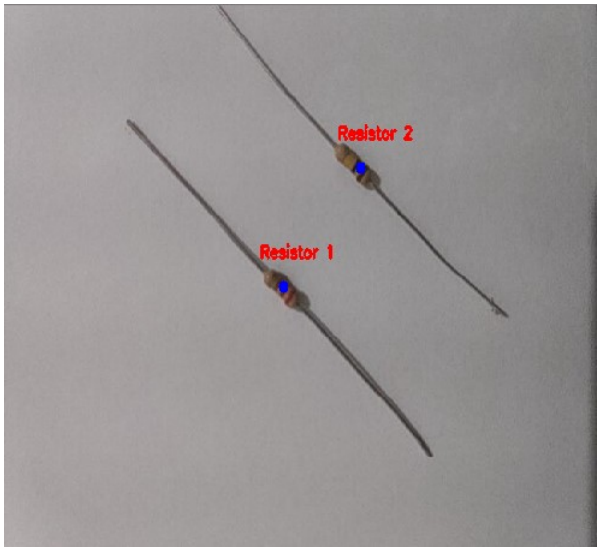

FIGURA 5: Rotulação.

O ponto azul nos resistores consiste no centroide encontrado anteriormente. Estes pontos serão de suma importância para o desenvolvimento da etapa seguinte.

3.4 Obtenção do eixo

Esta etapa será responsável pela obtenção do eixo principal do resistor para posterior acesso aos dados das cores nesse eixo. Com os centroides encontrados anteriormente, foi criada uma lógica para se descobrir o eixo mais longo do resistor que passa por todas as faixas de cores.

A ideia do algoritmo criado é, a partir do centro, partir com quatro retas posicionadas a 90º umas das outras até que seja encontrada a cor de fundo da figura, como mostrado abaixo:

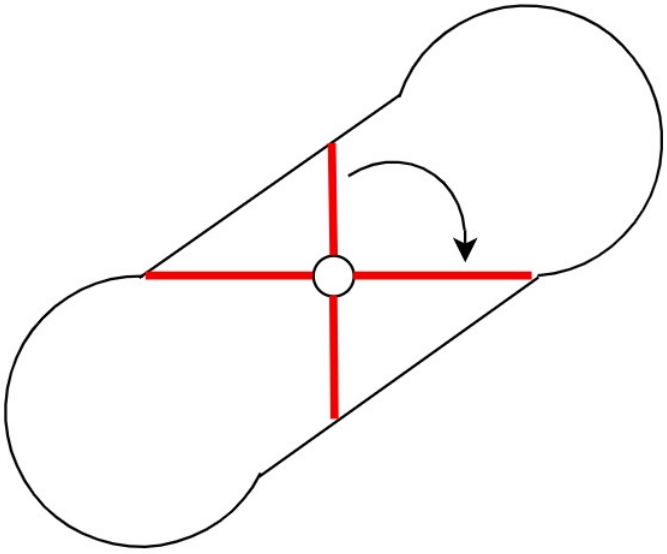

FIGURA 6: Primeiro passo para a obtenção do eixo.

Numa primeira iteração, são encontrados os pontos em que as quatro retas tocam a borda do objeto. Estes pontos são então adicionados a uma lista e o ângulo das retas é incrementado em um determinado passo, tendo como valor de parada 90º, que é o espaçamento entre as retas. A cada iteração, novos pontos são acrescentados nas listas.

Um dos grandes desafios foi a criação do algoritmo responsável por percorrer os *pixels* da matriz de acordo com a equação da reta:

$$y = mx \tag{1}$$

onde m consiste no coeficiente angular da reta e é dado pela tangente do ângulo que a reta forma com o eixo x. Uma função foi criada para, de acordo com o ângulo proposto para a reta, percorrer o eixo x com um determinado passo e calcular o valor correspondente de y. Vale ressaltar que todos os valores resultantes devem ser inteiros, visto que se trata de

índices da matriz da imagem. O algoritmo pode ser representado de forma simplificada da seguinte maneira:

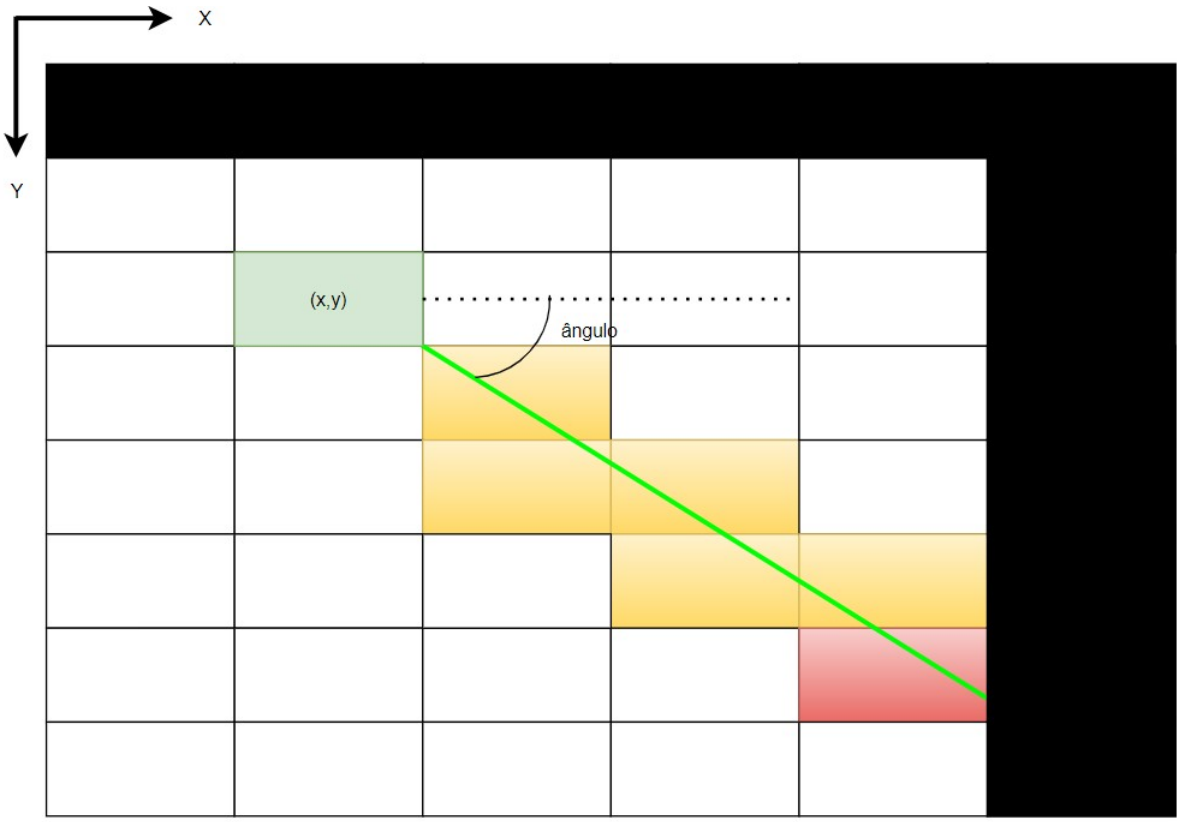

FIGURA 7: Equação da reta sobre a matriz de *pixels*.

O ponto em verde corresponde ao pixel de partida. Com um determinado ângulo fornecido, o algoritmo irá passar pelos *pixels* em amarelo e a posição do pixel em vermelho será adicionada à lista. Como citado anteriormente, esta representação é simplificada, visto que diversos problemas precisaram ser contornados, como por exemplo o tratamento da direção das retas em ângulos específicos e o cálculo do passo utilizado no eixo x que depende do coeficiente angular da reta, além da elaboração dos critérios de parada do algoritmo.

Em seguida, são calculadas as distância euclidianas entre entre o centroide e as extremidades das quatro retas encontradas para cada iteração do ângulo. O algoritmo então encontra a direção em que a reta foi mais longe, correspondendo assim ao eixo principal do resistor, como ilustra a figura a seguir:

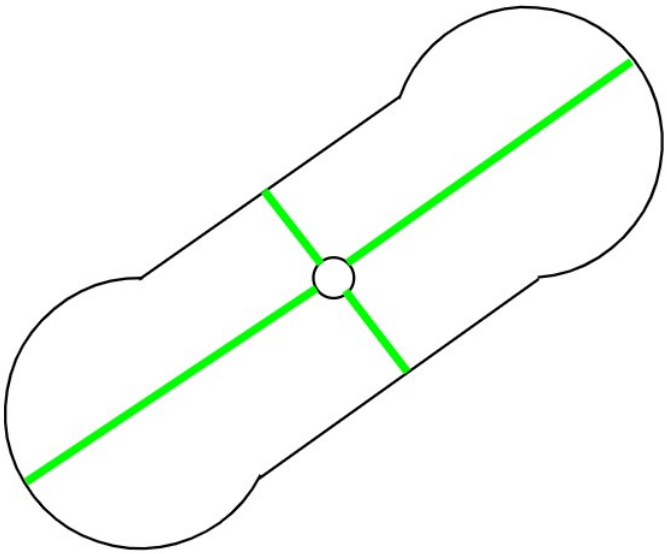

FIGURA 8: Obtenção do eixo principal.

Com esse resultado, tem-se os pontos que representam ambas as extremidades do eixo principal. Dessa forma, elaborou-se um algoritmo para poder obter as posições de todos os *pixels* presentes na reta que interconecta estes dois pontos. Para isso, foi utilizado novamente o algoritmo ilustrado na Figura 7, de forma que, em vez de ter como objetivo apenas o ponto final onde a reta intersecta a borda do objeto, todos os pontos foram acrescentados em uma lista.

A exibição dos pontos obtidos sobre a imagem obtida é mostrada abaixo:

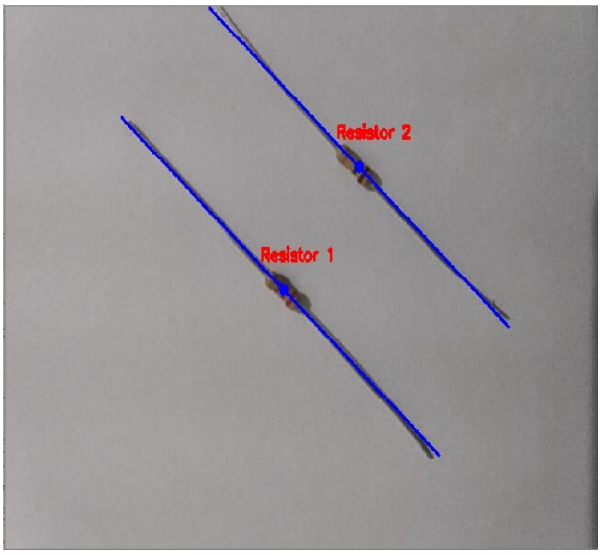

FIGURA 9: Representação do eixo principal.

Percebe-se que na prática, além do corpo principal do resistor, as pernas do componente também foram consideradas. Além disso, nesta etapa, o ajuste dos limiares do algoritmo

de *Canny* também se mostrou de suma importância, visto que em alguns momentos, a orientação da reta do eixo principal varia rapidamente, sendo que esta variação pode ser completamente sanada regulando-se os limiares.

3.5 Observação das cores nos eixos

Foram realizados testes com alguns resistores de valores diferentes. Os que obtiveram um resultado mais nítido em relação a observação dos padrões de cores no eixo principal foram os que possuíam faixas de cores que continham um maior contraste em relação à cor do resistor. A figura abaixo mostra o resultado das compontes RGB para um resistor de 10k Ohms:

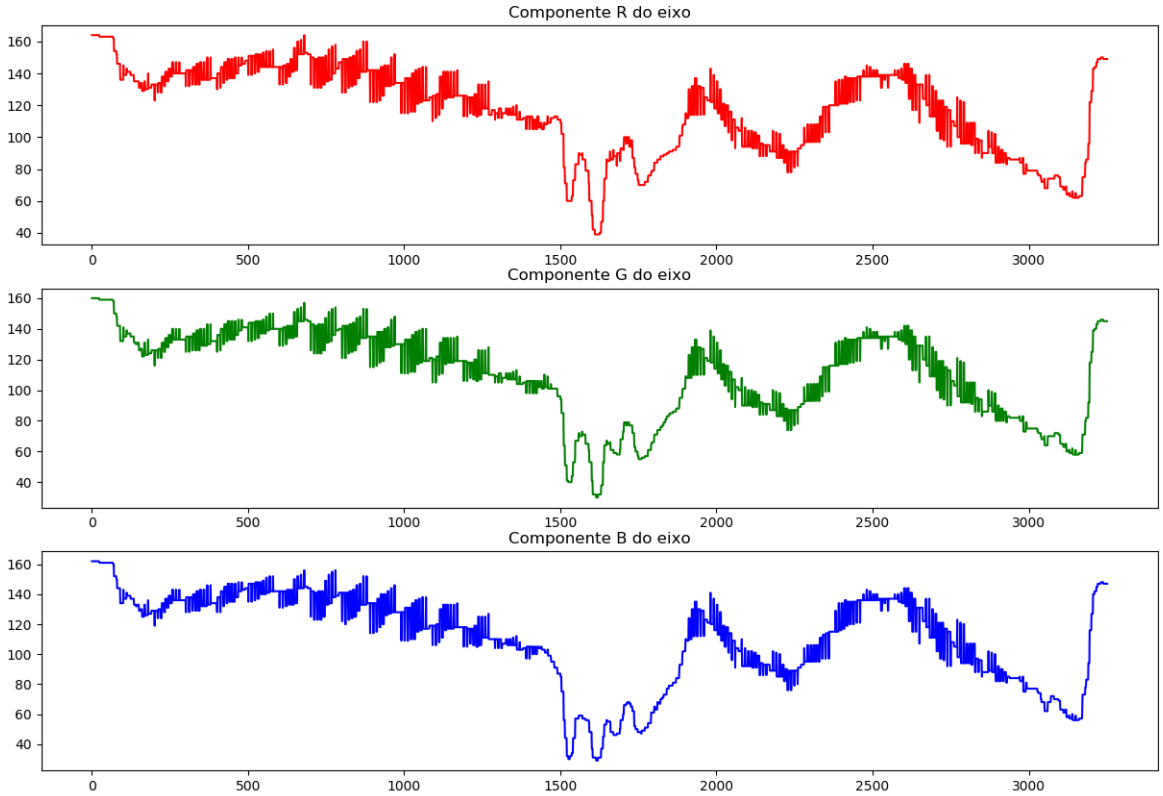

FIGURA 10: Cores do eixo central do resistor de 10k.

Neste gráfico, percebe-se a presença de muito ruído sobre o eixo. Porém, ao habilitar a filtragem descrita na Seção 3.1, o resultado foi o seguinte:

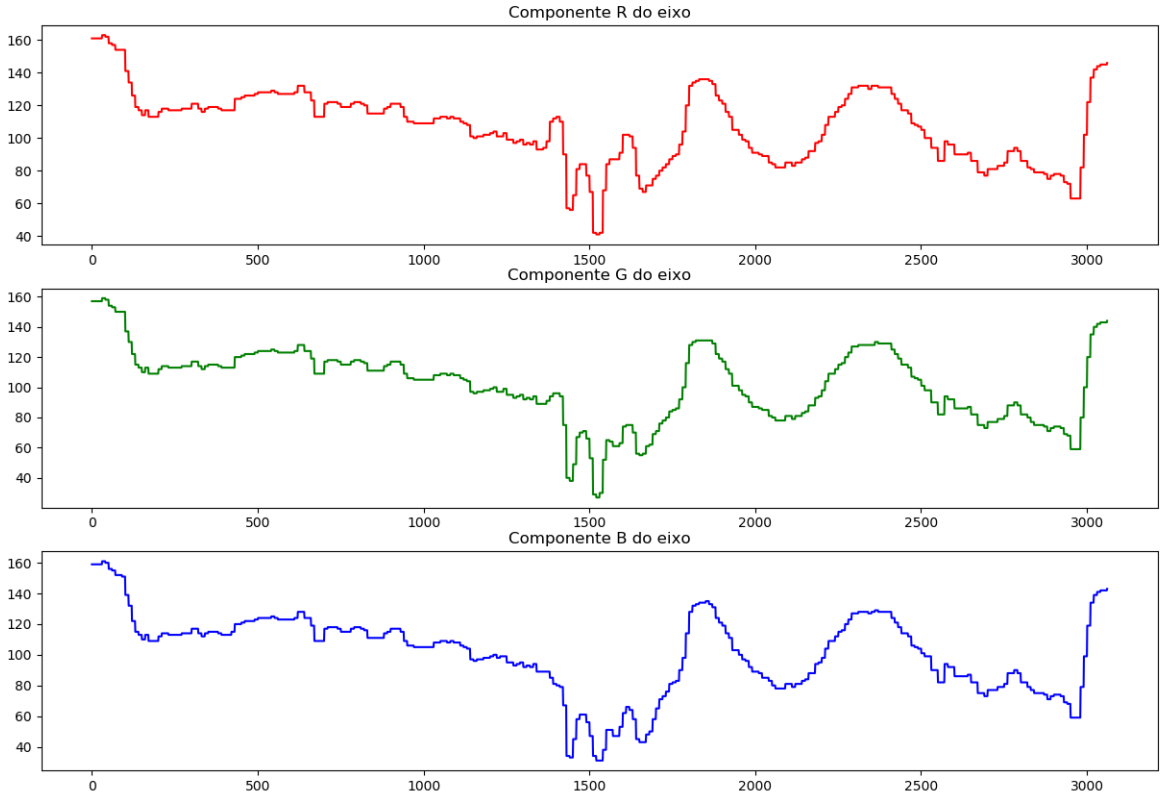

FIGURA 11: Cores do eixo central do resistor de 10k com processo de filtragem.

Neste caso, percebe-se nitidamente a melhoria no resultado. Em relação à observação das cores do resistor, está ocorre no vale em torno do centro do gráfico, visto que o meio da reta gerada pelo algoritmo passa pelo centroide do resistor. As cores do resistor de 10k são marrom, preto e vermelho. No centro do gráfico, percebe-se dois vales, correspondendo às duas primeiras faixas de cores escuras do resistor. A última faixa (vermelha) é observada justamente na componente em vermelho, onde há uma componente maior desta cor.

3.6 Limitações

Algumas limitações foram encontradas na implementação. Uma delas é a lentidão do algoritmo quando um resistor é encontrado na cena. O processamento dos quadros enviados do celular não acompanha a taxa de aquisição, de forma que a imagem de saída sai com um *delay* considerável e que depende da quantidade de objetos encontrados na cena.

Outra limitação é que nem todas as exceções que ocorrem foram tratadas, de forma que, às vezes, ao introduzir outro objeto na cena ou durante o reposicionamento dos resistores, o código pode travar.

Além disso, pode-se configurar o passo dos ângulos verificados para a obtenção da orientação do resistor. Porém, quanto menor o passo, mais lento o algoritmo se torna também, mas a mudança no resultado é perceptível, como mostrado abaixo:

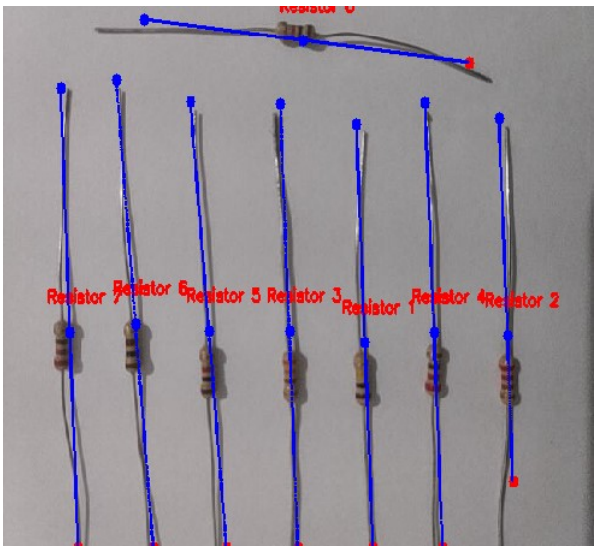

FIGURA 12: Alteração do passo do ângulo.

Para a imagem da esquerda, foi utilizado um passo de 5 graus. Já para a esquerda, o passo foi de 3 graus, onde perceb-se que quanto menor o passo, mais alinhado com o eixo do resistor a reta estará.

Além disso, a Figura 12 mostra outra limitação da implementação, que é a quantidade máxima de resistores observados na cena, devido à abertura da câmera a sua altura em relação ao plano da cena.

3.7 Dificuldades encontradas

Durante a implementação proposta, muitas dificuldades foram encontradas. A proposta inicial era de, através de erosão e dilatação, apagar as pernas dos resistores e deixar apenas o corpo principal. Esta ideia funcionou com uma imagem estática onde os resistores ocupavam um grande espaço na imagem. Na prática, os resistores eram muito pequenos em relação ao tamanho da cena e a presença das sombras dos objetos fez com que o processo de dilatação tornasse as pernas mais grossas do que realmente eram. Dessa forma, o processo foi impossibilitado mesmo após diversos testes.

Além disso, houveram problemas na implementação do algoritmo da equação da reta, devido à representação dos eixos como mostrou a Figura 7. Ainda neste trecho, muito tempo foi gasto para ser possível fazer com que as retas pudessem ser percorridas em todas as direções possível a partir de um ponto central.

Uma das ideias de implementação do código era plotar um gráfico com as cores do eixo, assim como os exibidos nas Figuras 10 e 11, porém, sendo atualizados em tempo real. Não

foi possível encontrar uma solução em tempo hábil para a execução deste processo e os gráficos passaram a ser gerados de forma estática após a finalização do programa. Ademais, o objetivo principal do trabalho era a obtenção dos valores do resistor, porém, não houve tempo hábil para a implementação desta função.

4 Conclusão

O presente trabalho foi muito importante para a elucidação dos conceitos envolvidos no Processamento Digital de Imagens, onde foi possível observar os problemas recorrentes, por exemplo, no processo de aquisição das imagens e como estas devem ser tratadas para se ter um processamento melhor. Além disso, pôde-se aprender mais funcionalidades relacionadas ao PDI através da linguagem *Python* e da biblioteca *OpenCV*.

Em relação ao que foi proposto pelo trabalho, apesar das dificuldades encontradas e problemas presentes no código, este conseguiu exercer o que foi proposto, tendo como entrada a imagem de um resistor em uma cena e sendo capaz de obter o eixo principal do resistor que passa por todas as faixas de cores.

Como trabalhos futuros, pretende-se organizar melhor a cena montada, com uma iluminação mais apropriada, além da utilização de uma câmera com melhor resolução. Com isso, a implementação do cálculo do valor do resistor baseado na imagem deve se tornar mais simples. Além disso, otimizações no código devem ser feitas tanto para evitar os travamentos quanto para diminuir o *delay* provocado pelo processamento.

Referências

[1] BRITO, Agostinho. Processamento digital de imagens. 2018. Disponível em: https://agostinhobritojr.github.io/curso/pdi/fundamentos.pdf. Acesso em: 14 dez. 2020.

[2] MORDVINTSEV, Alexander; REVISION, Abid K.. Smoothing Images. c2013. Disponível em: https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_filtering/py_filtering.html. Acesso em: 16 dez. 2020.

[3] MORDVINTSEV, Alexander; REVISION, Abid K.. Introduction to OpenCV-Python Tutorials. c2013. Disponível em: https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_setup/py_intro/py_intro.html#intro. Acesso em: 16 dez. 2020.