

Universidade Federal do Paraná  
Departamento de Informática - DInf

## **Relatório TA1 - processamento de imagem básico**

Alunos: Anna Caroline Bozzi e Vinícius de Lima Golçalves

Abril  
2023

# **Conteúdo**

<b>1</b>	<b>Introdução</b>	<b>1</b>
<b>2</b>	<b>Desenvolvimento</b>	<b>3</b>
<b>3</b>	<b>Conclusão</b>	<b>9</b>
<b>4</b>	<b>Link para o código desenvolvido</b>	<b>10</b>
	<b>Bibliografia</b>	<b>10</b>

# 1 Introdução

Uma imagem é uma matriz de *pixel*, que é um valor numérico representando uma intensidade, normalmente com 8 bits, que vai de 0 a 255, nas imagens em tons de cinza. Em imagens coloridas, no padrão RGB, essa matriz se torna tridimensional, sendo cada canal (R, G, ou B) a intensidade de cada cor.

Com isso, é possível gerar uma representação gráfica da frequência de ocorrência de cada valor de intensidade presente na imagem utilizando um histograma, sendo importante para saber quais valores são mais frequentes, seja na escala de cinza ou nos canais de cor, direcionando um melhor uso das demais técnicas processamento de imagem.

*Thresholding* é uma técnica simples de segmentação, onde a imagem em tons de cinza é convertida em uma imagem binária, onde os pixels são classificados em dois grupos distintos - preto e branco - com base em um valor de limiar(*threshold*) (Gonzalez; E., 2010).

O *threshold* é o valor que separa os *pixels* que serão considerados pretos ou brancos. Os *pixels* na imagem original com valores de intensidade abaixo do *threshold* são atribuídos à categoria preta na imagem binária e os *pixels* com valores acima do *threshold* são atribuídos à categoria branca (Gonzalez; E., 2010).

Existem vários métodos de *thresholding*, os usados nesse trabalho foram *Normal Thresholding*, *Automatic Thresholding*, *Multilevel thresholding* e *Local Thresholding*.

O *Normal Thresholding* resulta em uma imagem binária, em que os *pixels* são brancos quando acima do limiar e pretos abaixo do limiar determinado. Já o *Automatic Thresholding* determina o valor de limiar automaticamente com base na distribuição dos níveis de intensidade dos *pixels* na imagem. No *Multilevel thresholding* valores de limiar são calculados para 3 níveis e as imagens binárias são geradas com base nesses valores de limiar. E por fim o *Local Thresholding* é quando diferentes valores de *threshold* são aplicados a diferentes regiões da imagem.

Já os filtros são operações matemáticas que quando aplicadas em uma imagem que tem como objetivo extrair ou realçar determinadas características (Crósta; A. P., 2002). Eles são usados para remover ruídos, suavizar, realçar, borrar, detectar bordas e linhas nas imagens, dentre outras aplicações.

Dentre os filtros de suavização ou borramento estão os filtros de média, mediana e o gaussiano. O filtro de média é um filtro linear que substitui o valor de cada *pixel* pela média dos valores dos *pixels* em uma janela deslizante ao redor dele, através de uma *kernel* que é uma matriz no qual se realiza a operação, no qual é usado para reduzir ruídos de alta frequência na imagem

(Crósta; A. P., 2002). Já o filtro de mediana é um filtro não linear que substitui o valor de cada *pixel* pela mediana dos *pixels* da janela deslizante, sendo útil para remover ruídos de impulso, como o ”sal e pimenta” (Crósta; A. P., 2002). Por fim, o filtro gaussiano é um filtro linear que suaviza a imagem usando uma função gaussiana para ponderar os valores dos *pixels* na janela deslizante (Crósta; A. P., 2002).

Os filtros de detecção de bordas identificam mudanças abruptas nos valores dos *pixels*, que indicam a presença de uma borda ou linha na imagem. Entre eles estão o filtro *Sobel*, *Laplace* e *Canny*. O filtro *Sobel* usa duas máscaras para calcular a magnitude do gradiente da imagem e identificar as mudanças abruptas nos valores dos *pixels* na horizontal e vertical (Crósta; A. P., 2002). Já o filtro de *Laplace* é um filtro de segunda ordem que calcula a diferença entre os valores dos *pixels* na sua vizinhança. Isso pode ser usado para detectar bordas de diferentes orientações (Crósta; A. P., 2002). O filtro *Canny* é um filtro mais complexo que usa múltiplas etapas detectando bordas com maior precisão, nas quais inclui a aplicação de um filtro gaussiano para suavizar a imagem e a identificação dos *pixels* de borda com base em um limiar adaptativo (Crósta; A. P., 2002).

O Objetivo deste trabalho é realizar a implementação dessas técnicas, utilizando a biblioteca de processamento de imagens *OpenCV*, a fim de efetuar diversos testes para ambientação e propor uma aplicação real com as técnicas implementadas.

## 2 Desenvolvimento

Para entender melhor como as características da imagem escolhida poderia interferir nos demais processos, foram geradas imagens em cada canal de cor. Na figura 1 é possível visualizar na primeira linha os canais de cor ( Neste caso, preservou-se o canal da cor correspondente e zerou os demais, mantendo a imagem tridimensional). Na segunda linha, as imagens em tons de cinza representam a intensidade de cada cor, sendo a primeira imagem uma conversão que pondera as proporções de cada canal. Por conta disso, alguns tons que eram discriminantes na imagem colorida acabam ficando próximos, podendo atrapalhar futuramente.

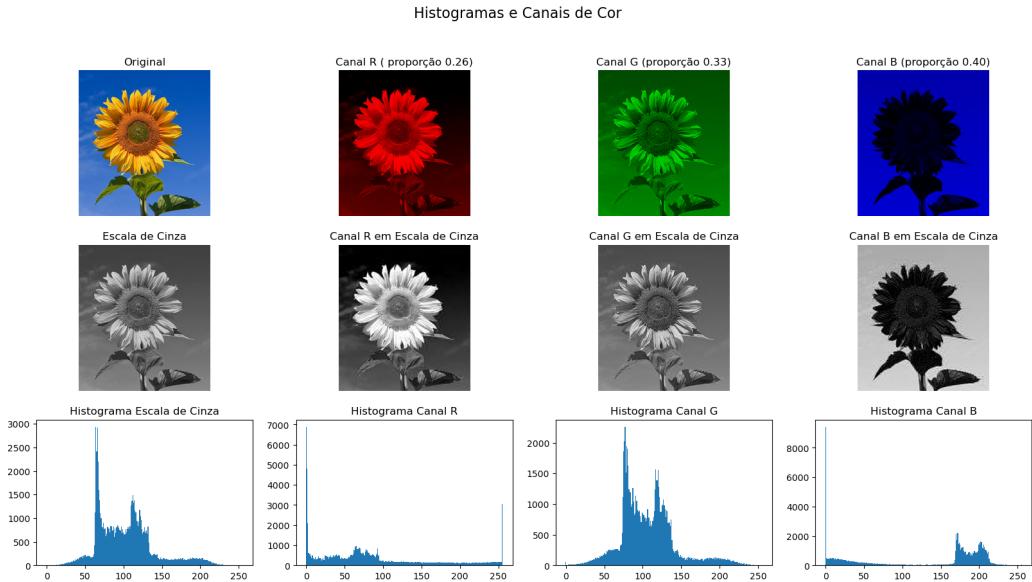


Figura 1: Representação dos canais de cores para a imagem do Girassol

Testando o comportamento de cada um dos canais com a aplicação do *thresholding* e dos filtros, foi possível ver que o canal vermelho seria o mais promissor, pois tinha mais intensidade desse canal no objeto. Entretanto, usar o canal azul e o inverso aparentou ser mais preciso, além de que seu histograma está mais disperso e concentrado em 2 polos, como mostra a figura 2 e 3.

Para a implementação da *Normal Thresholding*, *Automatic Thresholding*, *Multilevel thresholding* e *Local Thresholding* foi utilizado a imagem em tons de cinza considerando apenas o canal azul. As figuras 4, 5, 6 e 7 apresentam os resultados obtidos com esses processos. Como o *thresholding* torna a imagem binária, utilizando essa imagem proveniente do canal azul, cujo fundo está

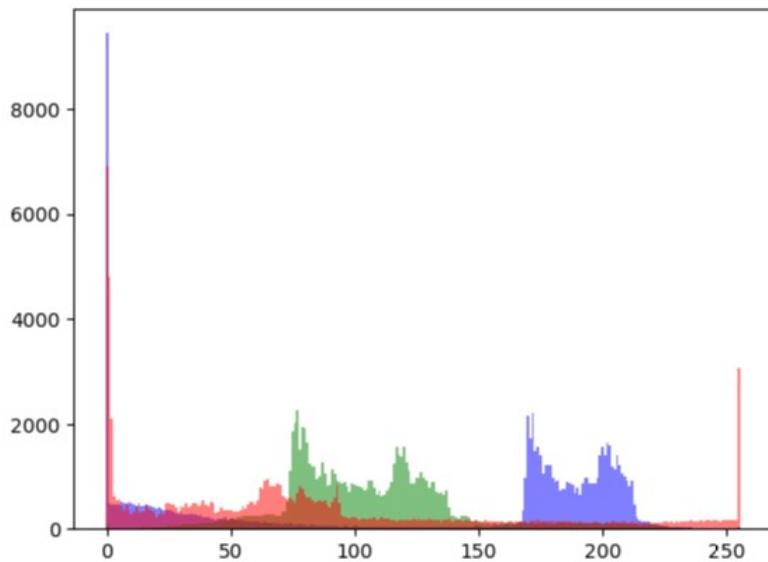


Figura 2: Histograma RGB

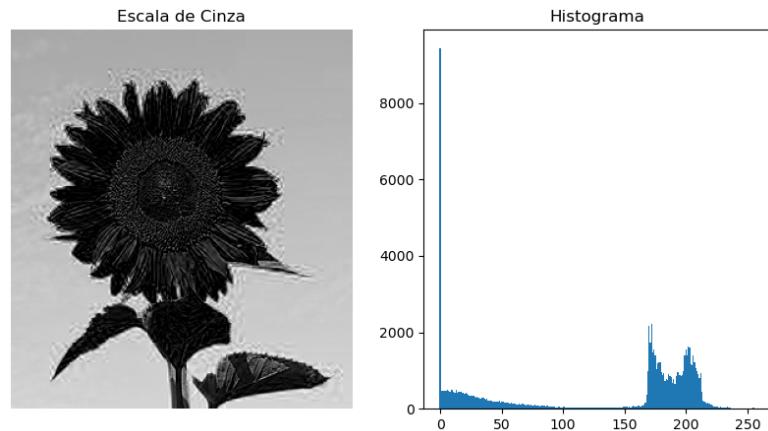


Figura 3: Imagem em escala de cinza com o seu histograma do canal azul

com uma maior intensidade do que o objeto, o resultado fica como se fosse *thresholding* invertido.

Os filtros de média, mediana e *Gauss* foram testados com a imagem original, alterando o tamanho da janela deslizante e gerando 5 imagens como mostrado nas figuras 8, 9 e 10.

Já filtros que detectam curvas como o *Sobel*, *Laplace* e *Canny* foram testados com a imagem em escala de cinza utilizando o canal azul, também alterando o tamanho da janela deslizante nos dois primeiros e o valor de *th-*



Figura 4: Resultado da técnica de *Normal thresholding*



Figura 5: Resultado da técnica de *Automatic Thresholding*

*resholding* no último, cada um gerando 5 imagens como mostrado nas figuras 11, 12 e 13.

Com todos esses processos testados, foi pensando em um *workflow* para extrair o fundo da imagem. A figura 14 demonstra o resultado obtido. Para realizar isso, foi utilizado inicialmente o filtro de *Sobel* com a janela deslizante



Figura 6: Resultado da técnica de *Multilevel thresholding*

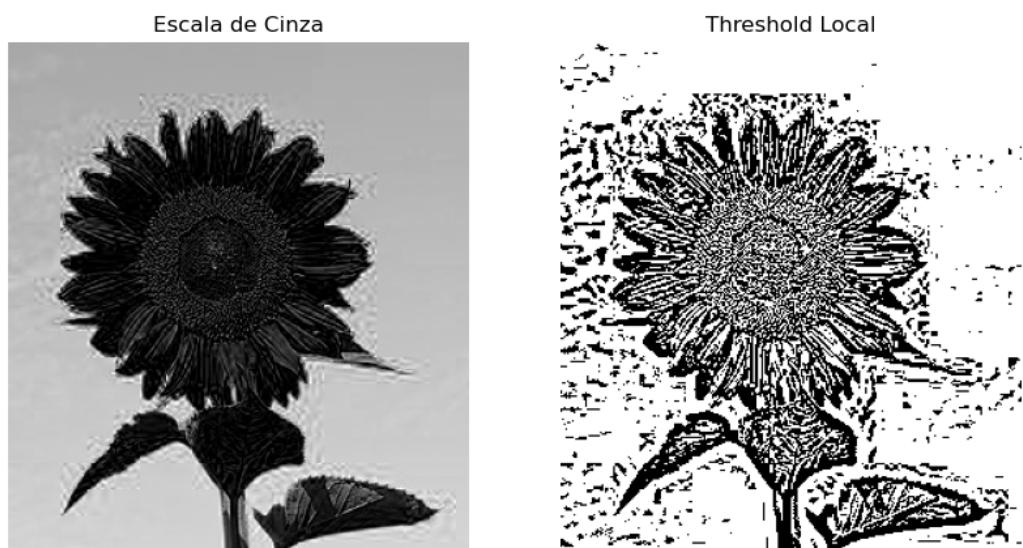


Figura 7: Resultado da técnica de *Local Thresholding*

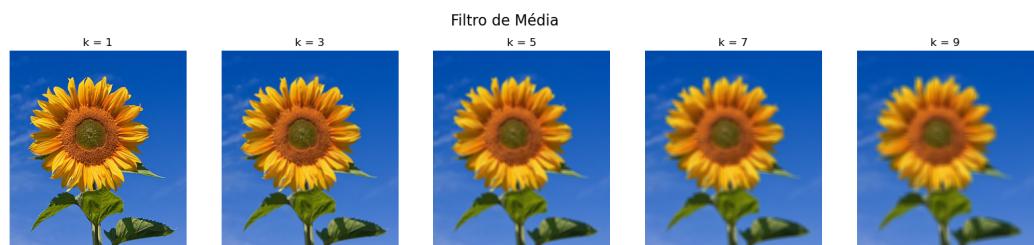


Figura 8: Resultados da aplicação do filtro da média

de  $5 \times 5$  na imagem com as intensidades do canal azul, resultando em uma imagem das bordas. Com essas bordas, foi feito o *thresholding* automático e invertido a binarização, com o intuito de mesclar as bordas com a imagem



Figura 9: Resultados da aplicação do filtro da mediana



Figura 10: Resultados da aplicação do filtro de *Gauss*



Figura 11: Resultados da aplicação do filtro de *Sobel*



Figura 12: Resultados da aplicação do filtro de *Laplace*

em tons de cinza, realçando ainda mais os limites entre o objeto e o fundo. A imagem resultante do *Sobel* está num espaço de 64 bits, e quando convertida para 8 bits os pixels "estouram". Entretanto isso realçou os traços na imagem cinza, que após ser mesclada foi feito um *thresholding* automático

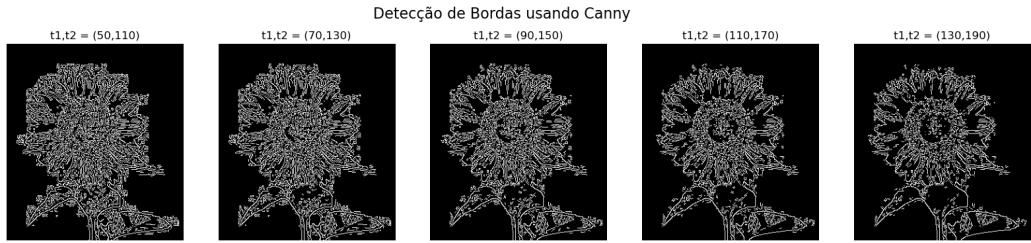


Figura 13: Resultados da aplicação do filtro de *Canny*

também para ter não só o contorno como também o preenchimento do objeto. Também foi invertido essa imagem resultante, com intuito de destacar a região de interesse. Percebeu-se que tinham ruídos muito parecidos com o ruído conhecido com sal e pimenta, então foi aplicado um filtro de mediana, com a *kernel* no tamanho de 5x5, para suavisá-los. Com isso, o *thresholding* do objeto ficou mais ajustado, então foi realizado a operação *bit a bit* chamada de *and*, essa operação zera na imagem original o que for correspondente zero na máscara, que é o *thresholding* ajustado, e mantém os *pixels* que forem 1. No fim, obteve uma imagem quase totalmente sem o fundo, permanecendo apenas uma pequena borda azul. Todo esse *workflow* está representado na figura 15.



Figura 14: Resultado da aplicação da extração do fundo da imagem



Figura 15: Passo a passo da aplicação de extração do fundo da imagem

### 3 Conclusão

Mesmo sendo técnicas básicas de processamento de imagens, o seu uso e entendimento podem ser cruciais para um futuro processamento. Existem muitas coisas que podem ser feitas utilizando essas técnicas, desde remoção de fundo como foi abordado nesse trabalho até extração de características utilizando *Sobel* por exemplo, que detalha muito bem as bordas das imagens, podendo ser utilizado como descritor para processo de aprendizado de máquina.

Entretanto, ficou evidente que algo que não se deve generalizar as técnicas, uma vez que cada imagem tem suas peculiaridades, devendo ajustar os métodos que mais podem ajudar na tomada de decisão. Esse processo muitas vezes é manual e através de testes.

## **4 Link para o código desenvolvido**

<https://github.com/ACBozzi/VisaoComputacionalPercepcao/tree/main/TA1>

## **Bibliografia**

GONZALEZ, R. C.; E., W. R. Processamento Digital de Imagens. [S.l.]: Prentice Hall, 2010.

CRÓSTA, A. P. Processamento Digital de Imagens de Sensoriamento Remoto. 2002.