UNIOESTE

Universidade Estadual
do Oeste do Paraná

Relatório de Processamento de Imagens Digitais

Professor:
André Luiz Brun
Adair Santa Catarina

Aluna:
Milena Lucas dos Santos

21 de dezembro de 2020

Conteúdo

1 Introdução

Segmentar a imagem é dividir esta em regiões ou objetos de interesse. Existem duas maneiras para segmentar uma imagem, sendo a Descontinuidade e a Similaridade. A Descontinuidade divide a imagem com base nas mudanças bruscas de intensidade, visíveis nas bordas dos objetos, a Similaridade divide a imagem em regiões que sejam semelhantes de acordo com um conjunto de critérios pré-definidos. Neste trabalho, a técnica de Descontinuidade foi a abordagem utilizada, onde são aplicadas uma série de técnicas para a extração e a segmentação das imagens. Para facilitar este processo, as imagens estão em um ambiente controlado, onde fatores externos não prejudicam o processamento das folhas. Após a segmentação da imagem, são necessários extrair informações importantes e salva-las, tal abordagem é chamada de Representação e Descrição. A descrição de uma região da imagem pode ser feita por suas características externas (fronteira) ou por características internas (pixels que constituem a região). A técnica Momentos Invariantes descreve uma imagem segmentada independente de sua escala, rotação ou translação. Neste trabalho, o objetivo é desenvolver um algoritmo para a segmentação das imagens e a descrição dessas imagens por meio da técnica Momentos Invariantes.

2 Segmentação e Descrição

Para a resolução do trabalho, a linguagem Python foi escolhida para a implementação, pois trás muitos benefícios e tutoriais para o aprendizado de processamento de imagens, principalmente a biblioteca OpenCV que disponibiliza vários tutoriais de implementação e teoria como *Image Thresholding*, *Image Gradients*, *Morphological Transformations* e outros. O programa foi implementado dentro de um ambiente virtual, para não haver problemas com diferentes versões do Python. Ao rodar o programa é apresentado para que o usuário digite o nome da imagem como apresentado na figura 1.

Após a imagem ser escolhida, é aplicada algumas técnicas para ser segmentada. A imagem primeiramente é convertida para tons de cinza, em seguida é removido os ruídos da imagem de tons de cinza aplicando a função

Figura 1: Interface

Figura 2: Imagem original segmentada.

Figura 3: Perímetro da imagem.

GaussianBlur. Então é aplicado a função chamada *Canny()*, considerado o melhor dos detectores de borda, baseado em três objetivos básicos: Baixa taxa de erros; Pontos de borda bem localizados; Resposta de borda bem localizados.

Através desses passos obtemos os contornos da imagem, demarcamos as folhas contornadas com a função *cv2.rectangle(imagem, (x, y), (x + w, y + h), (0,0,255), 2)*. Essa função pega a imagem e demarca as regiões de interesse por meio do *x,y,w,h*, também podemos escolher a cor do *bounding box* como também sua espessura. Para realizar a demarcação, primeiramente usamos uma função para procurar os contornos e após utilizamos um *for* para demarcar as folhas e corta-las. Nas figuras 2 e 3 são apresentadas um exemplo do resultado desse algoritmo. Contudo, apesar das técnicas aplicadas, algumas folhas não tiveram um bom contorno, causando problemas na segmentação como a divisão de várias regiões de uma folha ou o agrupamento de duas folhas ou mais em uma imagem segmentada. Para tentar resolver o problema, foi adicionado um *if* para que regiões menores que certo tamanho não fossem considerado, entretanto não funcionou em todas as imagens.

O número das folhas é apresentado no terminal ao término da execução e o perímetro é armazenado no arquivo *SegmentacaoMomentosInvariantes.csv*.

Figura 4: Parte da implementação da descrição Momentos Invariantes.

```
5   def calculaMediaX(fImage):
6       mediaX =0
7       for i in range(len(fImage)):
8           for j in range(len(fImage[0])):
9               mediaX += (j+1)*fImage[i,j]
10      mediaX = mediaX/m00
11      return mediaX
12
13  def calculaMediaY(fImage):
14      mediaY =0
15      for i in range(len(fImage)):
16          for j in range(len(fImage[0])):
17              mediaY += (i+1)*fImage[i,j]
18      mediaY = mediaY/m00
19      return mediaY
20
21  def mediaInvariancia(p, q, fImage):
22          count = 0
23      for i in range(len(fImage)):
24          for j in range(len(fImage[0])):
25              count += ((i+1)-m10m00)**p * ((j+1)-m01m00)**q * fImage[i,j]
26      return count
27
28  def moment(p, q, fImage):
29      lamda = ((p+q)/2) + 1
30      media = mediaInvariancia (p, q, fImage)
31      return media / ((m00) **lamda)
```

A técnica Momentos Invariantes foi implementada descrevendo as fórmulas apresentadas pelo professor em aula, sendo somente o primeiro e segundo momento, na qual beneficiou o tempo de execução do programa. Na figura 4 é apresentado uma parte do código desenvolvido da descrição, sendo calculado a média de x e y, a soma do momento central e os momentos centrais normalizados. Os dados são salvos no mesmo arquivo *.csv*.

3 Conclusão

Apesar do programa não estar funcionando 100% do esperado, o aprendizado foi agregado, acredita-se que o resultado da descrição Momentos Invariantes está de acordo com o que foi proposto. Contudo, ainda é necessário investigar técnicas e maneiras para melhorar a segmentação das imagens.