

000

001 Projeto Demonstrativo 4 - Segmentação de 002 Imagens Aéreas

003

004 Natalia Oliveira Borges

005 natioliveira97@hotmail.com

006

Matrícula:

007 16/0015863

008

Lívia Fonseca

009 liviagcf@gmail.com

010

Matrícula:

011 16/0034078

012

013

014

Departamento de Ciência da

015 Computação

016 Universidade de Brasília

017 Campus Darcy Ribeiro, Asa Norte

018 Brasília-DF, CEP 70910-900, Brazil,

019

020 0 1

021

022

023

024 Abstract

025

026 Visão computacional é a ciência que trata de retirar informações de imagens para que possam ser interpretadas por máquinas. Nesse contexto, a segmentação de certas características é extremamente importante. Esse projeto tem como objetivo apresentar métodos de segmentação para classificar características de imagens aéreas. Implementamos e relatamos os resultados obtidos através do método dos K-vizinhos mais próximos, do método Support Vector Machine e limiares de cor.

027

028 1 Introdução

029

030 Grande parte das aplicações de visão computacional começam por uma segmentação de características em imagens. A segmentação ajuda a identificar e reduzir a região de interesse, diminuindo o esforço computacional para realizar o estudo das características em questão.

031

032 A segmentação de imagens já é usada em várias aplicações, como: identificação de tumores e patologias em exames de imagem, identificação de pessoas em câmeras de vigilância, estudo de poluição em lagos e rios [1] e outros.

033

034 Nesse projeto, trabalharemos com segmentação de prédios em imagens aéreas. Para isso usaremos o banco de imagens Inria Aerial Image Labeling [2] que possui fotos aéreas de várias cidades e ground truth de onde efetivamente há prédios na imagens.

035

036

Implementamos alguns métodos para fazer essa segmentação.

037

038 1.1 Método dos K-Vizinhos Mais Próximos

039

040 Esse método de aprendizagem é especialmente focado em classificação de características [3]. Primeiramente, um certo número de amostras com sua devida classificação é escolhido para treinamento.

041

042 © 2018. The copyright of this document resides with its authors.

043 It may be distributed unchanged freely in print or electronic forms.

044

045 ⁰Lívia: Código(Limiar de cor) Relatório(Introdução, Metodologia, Resultados e Discussões e Conclusão)

¹Natalia: Código (Método knn, método svm); Relatório(Abstract, Introdução, Metodologia, Resultados)

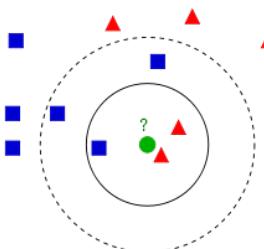


Figure 1: Método dos K-Vizinhos Mais Próximos

Depois do treinamento, é feito o teste usando uma outra amostra. Para cada ponto da amostra de teste é calculada a distância entre esse ponto e todos os pontos do treinamento. Esconhendo um número k , de preferência ímpar, vemos quais são os k pontos do treinamento que mais se aproximam do ponto de teste. A classificação mais frequente nesses k pontos será a classificação do ponto de teste.

Esse método possui um treinamento relativamente rápido, já que é basicamente um armazenamento de dados. Porém para cada teste é necessário calcular a distância do ponto estudado para todos os pontos de treinamento e encontrar os k vizinhos mais próximos, tornando a etapa de teste bastante lenta.

1.2 Método Support Vector Machine

Support Vector Machine [■] é um método de aprendizado de máquina para fazer classificação. O objetivo desse método é dividir o espaço de dados por meio de hiperplano em que de um lado ficam os dados de uma classe e do outro os da outra classe. Esse hiperplano pode ser uma reta que segmenta dois semiplanos em um espaço 2D, como mostrado na figura 2

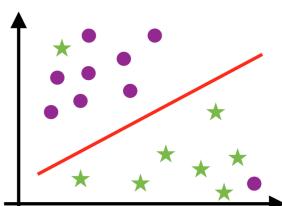


Figure 2: Método Support Vector Machine

Quando os dados não podem ser classificados de forma linear, realizamos um "kernel trick" para moldar a forma dos dados e conseguir segmentar através de um hiperplano.

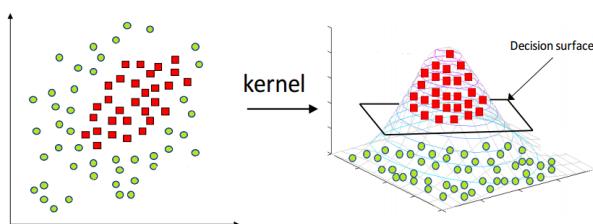


Figure 3: Kernel Trick

092 1.3 Limiar de cor

093 Aplicar um limiar de cor significa testar a condição de que o valor do pixel de uma imagem
094 está dentro de certo intervalo. Se a condição for atendida, o pixel na máscara correspondente
095 terá valor verdadeiro (255). Senão, terá valor falso (0).

097 1.3.1 Espaço YCbCr

099 O espaço YCbCr é muito usado em vídeos atualmente. Nesse espaço é formado por três
100 parâmetros base: Y, que indica a luminância da imagem, Cb que indica a diferença-azul e Cr
101 que indica a diferença-vermelho [1].

102 Para se aplicar um limiar nesse espaço de cores, é necessário testar se cada componente
103 se encontra dentro do intervalo especificado.

104 2 Metodologia

105 Esse trabalho foi feito em Linux Xubuntu versão 18.04 e em Linux Ubuntu versão 16.04,
106 ambos usando Opencv versão 3.2.0. O código foi feito python3.5.

111 2.1 Cálculo de acurácia e jaccard

112 Para avaliar os resultados temos dois parâmetros, a acurácia e o index de jaccard.

113 A acurácia mede a quantidade de pixels classificados corretamente, pelo número total de
114 pixels.

$$\frac{(results \cap ground_truth) + (results \cup ground_truth)}{total_pixels} \quad (1)$$

115 O index de jaccard é outra forma de medir os resultados e é dado pela interseção dividido
116 pela união:

$$\frac{results \cap ground_truth}{results \cup ground_truth} \quad (2)$$

124 2.2 Método dos K-vizinhos mais próximos

125 Para aplicar o método dos K-Vizinhos mais próximos na imagem, classificamos os pixels da
126 imagem de treinamento em duas categorias: telhado e não telhado. Cada pixel possui 3 bytes
127 referentes aos canais RGB da imagem. Esses valores são usados para situar o pixel como um
128 ponto em um espaço 3D.

129 Após aplicar o treinamento na primeira imagem, fazemos a etapa de teste. Para isso
130 escolhemos um valor de K=3, baseado em testes experimentais. Assim, o algoritmo levará
131 em conta os 3 pixels mais próximos do pixel de estudo da segunda imagem para fazer a
132 classificação.

133 Com o resultado da segmentação da segunda imagem fazemos a etapa de validação.
134 Nessa etapa há a verificação dos erros na imagem, os pontos marcados incorretamente são
135 separados para refazer treinamento. Depois desse segundo treinamento, aplicamos o teste na
136 terceira imagem e calculamos acurácia jaccard.

2.3 Método Support Vector Machine

Semelhante ao que fizemos no método anterior, usamos os três canais RGB do pixel para situar os todos os pixels da imagem em um espaço 3D. E treinamos o método com os dados da primeira imagem.

Aplicamos o teste na segunda imagem e verificamos os pontos em que houve erro, para esses pontos refizemos o treinamento e por fim aplicamos o esse treinamento na terceira imagem.

2.4 Limiar de cor

Foram utilizados loops iteradores para variar os limites do limiar para Cr e Cb, os limites de Y foram 100 e 255, e foram escolhidos os limiares que forneceram o melhor valor do index de jaccard.

Em seguida, esse limiar foi aplicado na imagem de validação e foram feitas novas iterações para se encontrar os valores que forneciam o melhor jaccard. Por fim, os limites encontrados foram aplicados na imagem de teste.

3 Resultados e Análise

A fim de comparação, calculamos a acurácia e o jaccard para uma imagem de resultado totalmente preta (apenas fundo) e para uma imagem totalmente branca (apenas prédio).

Os resultados foram os seguintes:

- Apenas prédios:

Jaccard = 34,78%

Acurácia = 34,78%

- Apenas fundo:

Jaccard = 0%

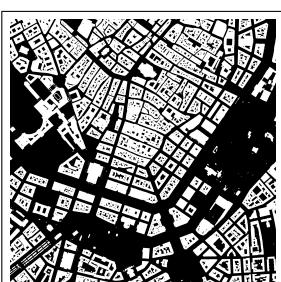
Acurácia = 65,22%

3.1 Método dos K-Vizinhos Mais Próximos

Treinamos o método com a imagem vienna16.tif, validamos na imagem vienna5.tif e finalmente avaliamos na imagem vienna22.tif.



(a)



(b)

Figure 4: (a) Imagem de treinamento (b) Ground truth

138

139

140

141

142

143

144

145

146

147

148

149

150

151

152

153

154

155

156

157

158

159

160

161

162

163

164

165

166

167

168

169

170

171

172

173

174

175

176

177

178

179

180

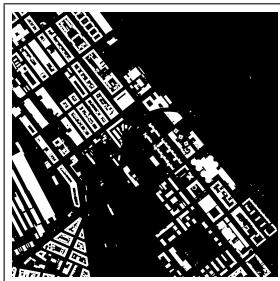
181

182

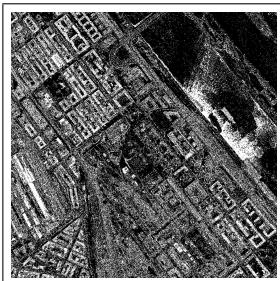
183



(a)



(b)



(c)

Figure 5: (a) Imagem de Validação (b) Ground Truth (c) Resultado knn

Jaccard = 20,83%

Acurácia = 57,22%



(a)



(b)



(c)

Figure 6: (a) Imagem de Validação (b) Ground Truth (c) Resultado knn

Jaccard = 38,88%

Acurácia = 69,75%

Vimos que o resultado entre a imagem de validação e de avaliação foi melhor, logo o método de retreinamento dos pixels errados foi efetivo. Porém o resultado final ficou muito ruidoso e não teve um jaccard nem acurácia muito alta. Isso se deve a diferença entre os pixels de telhado e não telhado da imagem de treinamento e de teste, mas principalmente ao fato de que os existem pixel de telhado e não telhado com cores muito semelhantes, e as vezes até iguais, sendo difícil até para humanos reconhecer alguns lugares.

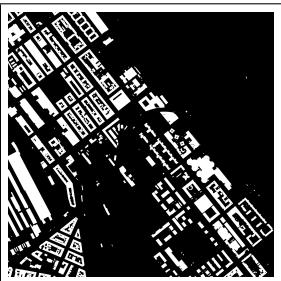
Tentamos usar outros espaços de cor, como HSL, YCrCb e HSV, mas o resultado não mudou muito, já que esses pixels iguais ou muito próximos em RGB também são iguais ou muito próximos nos outros espaços.

3.2 Método Support Vector Machine

Treinamos o método com a imagem vienna16.tif, validamos na imagem vienna5.tif e finalmente avaliamos na imagem vienna22.tif.



(a)



(b)



(c)

Figure 7: (a) Imagem de Validação (b) Ground Truth (c) Resultado svm

230

231

232

233

234

235

236

237

238

239

240

241

242

243

244

245

246

247

248

249

250

251

252

253

254

255

256

257

258

259

260

261

262

263

264

265

266

267

268

269

270

271

272

273



(a)



(b)



(c)

Figure 8: (a) Imagem de Validação (b) Ground Truth (c) Resultado svm

255

256

257

258

259

260

261

262

263

264

265

266

267

268

269

270

271

272

273

4 Limiar de cor

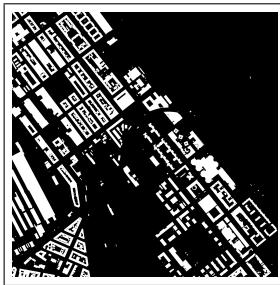
Treinamos o método com a imagem vienna16.tif, validamos na imagem vienna5.tif e finalmente avaliamos na imagem vienna22.tif.

274

275



(a)



(b)



(c)

Figure 9: (a) Imagem de Validação (b) Ground Truth (c) Resultado limiar de cor

Jaccard = 21,06%

Acurácia = 40,88%



(a)



(b)



(c)

Figure 10: (a) Imagem de Validação (b) Ground Truth (c) Resultado limiar de cor

Jaccard = 39,78%

Acurácia = 56,98%

O resultado desse método ficou razoável, mas não excelente, pois, nas imagens, prédios e não prédios possuíam cores muito parecidas, então o resultado ficou ruidoso e com muito branco ao segmentar a imagem por limiar de cor.

5 Discussão e Conclusões

Com esse projeto, foi possível conhecer e entender alguns métodos para segmentação de uma imagem, possibilizando classificar os pixels em duas categorias, prédios e não prédios.

Para os tipos de imagem utilizados, percebemos que métodos de aprendizado de máquina usando knn foi mais efetivos que o método de segmentação de cores. Isso aconteceu, pois, como deveríamos diferenciar prédios e não prédios e os prédios possuíam uma cor muito parecida com a cor das ruas, o que causou ambiguidade.

Além disso, os métodos de aprendizado de máquina (knn e svm) se mostraram muito dependentes da imagem utilizada para treinamento e a forma encontrada para melhorar o resultado foi fazer um segundo treinamento usando a imagem de validação.

Os resultados obtidos não foram ideais, contudo, esse projeto exigiu uma extensa pesquisa e, juntamente com as diferentes tentativas, contribuiu bastante para o aprendizado.

References

- [1] Guillermo Cámara-Chávez. Sistema de cores. <http://www.decom.ufop.br/guillermo/BCC326/slides/Processamento%20de%20Imagens%20-%20Sistema%20de%20Cores.pdf>. 322
323
324
325
326
- [2] Jamie Gray. Using computer vision to detect river pollution, January 31, 2018. <https://jamiegray.net/ux-ui/2018/river-color-analysis>. 327
328
329
- [3] Emmanuel Maggiori, Yuliya Tarabalka, Guillaume Charpiat, and Pierre Alliez. Can semantic labeling methods generalize to any city? the inria aerial image labeling benchmark. In *International Geoscience and Remote Sensing Symposium (IGARSS)*. IEEE, 2017. doi: 10.1109/IGARSS.2017.8127684. <https://project.inria.fr/aerialimagelabeling/>. 330
331
332
333
334
- [4] Wikipedia contributors. K-nearest neighbors algorithm — Wikipedia, the free encyclopedia, 2018. https://en.wikipedia.org/w/index.php?title=K-nearest_neighbors_algorithm&oldid=865557299;[Online; accessed 30-October-2018; Page Version ID: 865557299]. 335
336
337
338
339
- [5] Wikipedia contributors. Support vector machine — Wikipedia, the free encyclopedia, 2018. https://en.wikipedia.org/w/index.php?title=Support_vector_machine&oldid=865035110;[Online; accessed 30-October-2018; Page Version ID: 865035110]. 340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367