Composição Automática de Imagens Dispersas

Eliel Marcos Rocha Romancini

Departamento de Computação – Universidade Federal de Santa Catarina (UFSC)
Araranguá, SC – Brazil

elielmarcos@hotmail.com

Abstract. *The algorithm, developed in Python programming linguage with the aid of the OpenCV tool, consistis of rearranging in a cohesive and automatic way, images that are scattered and disorganized, as well as in a Puzzle game where the ultimate goal is to go back a totalized figure that is divided into subfigures (pieces). This problem is similar in reassembling importante torn documents, satellite images or even to Puzzles games.*

Resumo. *O algoritmo, desenvolvido em linguagem de programação Python, com auxílio da ferramenta OpenCV, consiste em rearranjar de forma coesa e automática, imagens que estão dispersas e desorganizadas, assim como em um jogo de "quebra-cabeça", onde o objetivo final é remontar uma figura totalizada que está dividida em subfiguras (peças). Este problema é semelhante na remontagem de documentos importantes rasgados, imagens de satélite ou até mesmo para jogos Puzzles.*

1. Introdução

A aplicação de visão computacional no processamento de imagens digitais, abrange um campo muito grande na resolução de problemas que simulam o comportamento humano com base na visão artificial. Um desses problemas está relacionado a resolução automática da ordenação de figuras, onde estas juntas compõe uma imagem maior e completa.

Para o algoritmo proposto, este tem a função de encontrar, a partir de uma imagem fornecida pelo usuário, as peças que ali estão dispersas e desorganizadas. Com as peças, uma a uma identificada, é iniciado o processo de combinação, onde é verificado entre elas quais possuem um certo grau de similaridade em suas bordas/lados, para que assim possam ser agrupadas de forma coesa resultando em uma única figura final que possua alguma regularidade. É possível imaginar como a resolução de um jogo de puzzle, onde sobre uma mesa estão as peças dispersas e o objetivo final é junta-las para forma uma figura.

Os principais passos do algoritmo são, identificar as peças na imagem fornecida, comparar entre elas a similaridade de seus lados e, por último, reorganiza-las lado a lado para formar a imagem como um todo.

Como forma de simplificar o problema, as figuras dispersas devem possuir apenas quarto lados no formato retangular, estarem posicionadas na vertical ou horizontal e com *background* de cor preta.

2. Identificando as peças

O primeiro passo do algoritmo é identificar cada peça fornecida na imagem pelo usuário, sendo ela uma imagem de três canais (RGB) é convertida para apenas um canal (GRAY) utilizando a função *cvtColor* da biblioteca OpenCV. Em seguida é aplicado um limiar *Threshold* para diferenciar as figuras do *background*, onde em suas posições prevalecerá a cor branca e o fundo a cor preta.

Nesta etapa é aplicada a função *findContours* que retornará os pontos dos cantos de cada peça, assim com essas coordenadas é possível construir máscaras individuais para extrair cada figura e armazena-las em uma lista de peças.

Na figura 1 é ilustrado um exemplar da imagem de entrada e as peças identificadas.

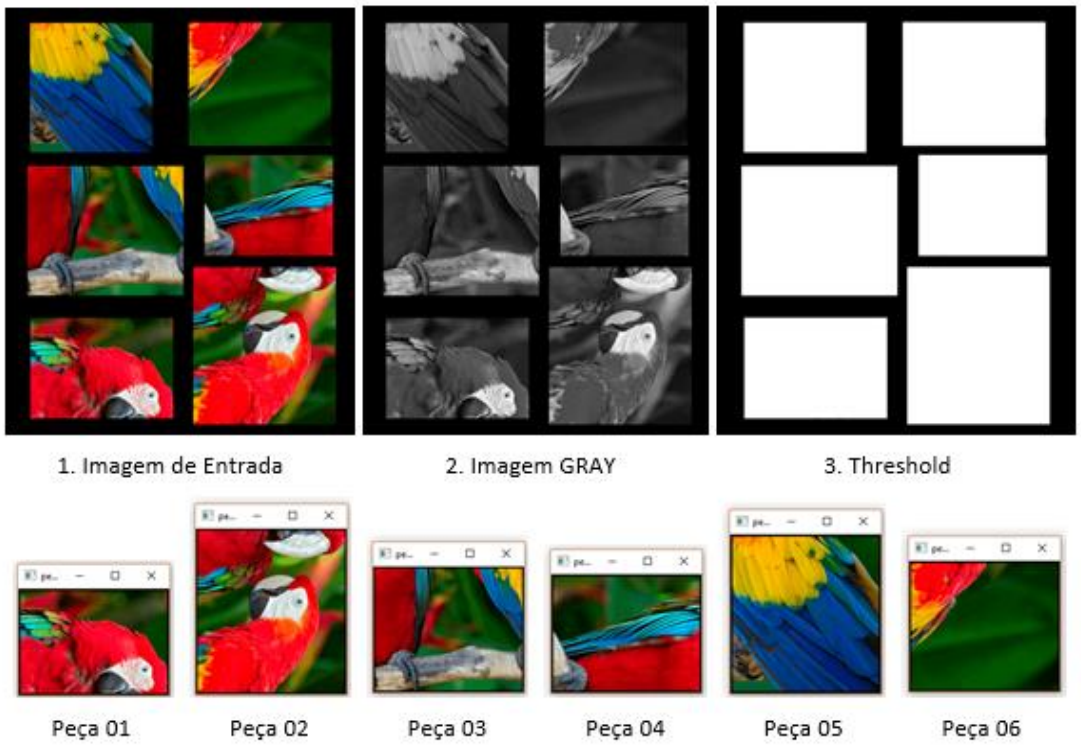

Figura 1. Primeira etapa, extraindo as peças

3. Verificando a similaridade

Com cada peça separada da imagem de entrada, a próxima etapa é verificar quais lados possuem um certo grau de semelhança. Para isso é necessário testar os lados de cada peça com os lados das peças que possuem o mesmo tamanho ou quase o mesmo, sendo considerado uma variação absoluta de até cinco *pixels* entre os dois lados em questão.

O processo é executado juntando duas peças por vez em uma única imagem, no entanto é necessário rotacionar cada peça de forma correta para unir os lados desejados, sendo identificados por Lado 1 a extremidade esquerda, Lado 2 a extremidade inferior, Lado 3 a extremidade direita e Lado 4 a extremidade superior.

Na união, uma das peças deve ser rotacionada para que o lado que se deseja avaliar fique no mesmo sentido que o vetor diretor x positivo, enquanto a outra peça deve ser rotacionada para que o lado fique no sentido do vetor diretor x negativo. Por exemplo, para analisar a similaridade da Peça 03, Lado 1 com a Peça 04, Lado 2, da figura 1, é necessário rotacionar a primeira 90º e a segunda 180º, conforme figura 2.

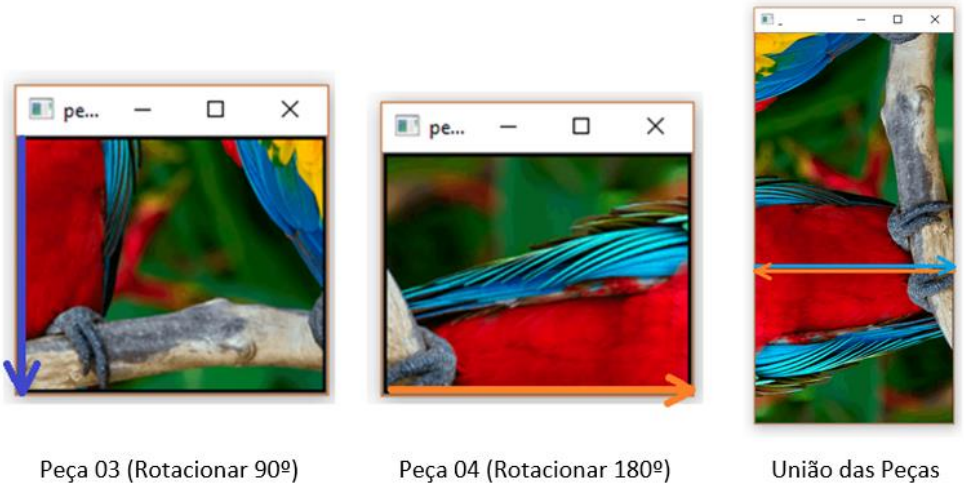

Figura 2. Segunda etapa, unindo as peças

Após a união de duas peças que se deseja testar, o próximo passo é aplicar o operador *Canny*, está função tem como objetivo encontrar através de uma varredura (x, y) transições abruptas na variação da escala de cores, logo este operador reage como uma derivada. O objetivo é encontrar a formação de uma linha na imagem, exatamente na junção das duas peças, quando elas possuírem uma baixa ou nenhuma similaridade e a não formação da linha quando possuírem uma alta similaridade, caracterizando que as duas partes "casam".

A figura 3 representa a união de dois lados que possuem baixa similaridade e outros dois que possuem alta similaridade. É evidente a formação de uma linha na região de união para o primeiro caso e não formação para o segundo caso, após aplicação do operador *Canny*.

Figura 3. Segunda etapa, verificando a similaridade através do operador *Canny*

4. Composição da imagem resultante

Após todos os lados terem suas similaridades avaliadas, aqueles que possuírem os maiores graus, são juntados em uma única imagem resultante final, através de uma função recursiva.

Esta etapa do algoritmo realiza a composição iniciando por uma das peças que obteve mais lados correspondentes. A recursão verifica se a peça atual, que está sendo inserida na imagem resultante, possui alguma peça que tem correspondência ao Lado 1 da mesma. Como a similaridade já foi analisada na etapa anterior é fácil saber quais lados das peças "casam", logo se existe alguma peça que encaixa, esta é inserida na composição, posicionando-a na coordenada exata e rotacionando-a se necessário, então o algoritmo passa a analisa-la como peça atual, caso contrário é verificado o próximo lado até chegar ao último, Lado 4. Assim que todos os lados de todas as peças forem examinados, o algoritmo finaliza a recursividade tento como resultado a composição da imagem final.

A fim de evitar a recursividade infinita, cada lado de cada peça, uma vez verificado não será visitado novamente e uma peça quando inserida na imagem resultante não será inserida novamente. A figura 4 ilustra a ordem da composição da imagem final iniciando pela Peça 03.

Figura 4. Última etapa, composição da imagem resultante

5. Conclusão

Os testes realizados em imagens com até 16 peças dispersas, demonstrou que o algoritmo detém uma confiabilidade muito boa, retornando o grau de similaridade entre 75 a 100% para os lados que existem correspondência. No entanto se mantém limitado a peças no formato de retângulos e com posicionamento sempre em eixos x e y, logo, não foram testadas peças com certo grau de rotação.

Diante desses resultados, fica evidente que o processamento de imagens digitais é de suma importância para acompanhar as tecnologias encontradas e as que estarão por vir, facilitando e simulando o comportamento humano real.

Para atualizações futuras, poderia ser implementado a composição de peças com 3 a mais lados, não limitando-se a estruturas retangulares, assim como utilizar outros métodos, em conjunto com operador *Canny*, para validação do *match*, dentre eles a análise das cores nos extremos de cada peça.

6. Referências

Albertazzi, R. (2018) "Solving Jigsaw puzzles with Python and OpenCV", https://towardsdatascience.com/solving-jigsaw-puzzles-with-python-and-opencv-d775ba730660, Julho.

Merry, J. (2014) "Algoritmo de resolución automática de puzzles", Departamento de Tecnologia Eletrônica, Universidade de Málaga, https://pt.slideshare.net/TFG_PFF_acg/algoritmo-de-resolucion-automatica-de-puzzles-jo-merry, Julho.

Tarallo, A.; Jorge, L.; Silva, F.; Hiraga, A.; Paiva, M. (2011) "Construção automática de mosaico de imagens agrícolas aéreas sequenciais", https://ainfo.cnptia.embrapa.br/digital/bitstream/item/83526/1/Proci-11.00253.PDF, Julho.