Mosaicagem de imagens para aplicações industriais

Cádson Alexandre Alves de Oliveira
Universidade Federal de Minas Gerais
Visão Computacional
cadsonalexandre@gmail.com

Resumo

A Mosaic is an grouping of small parts forming a bigger object. Image Mosaic is used when the scene is bigger than camera can capture. This article describe a simple implementation in C++, using the OpenCV correlation, capable to construct to an controlled mosaic with images from four cameras, like an 2×2 matrix , aiming at an industrial application.

1. Resumo

Um mosaico é o agrupamento de pequenas peças formando um objeto maior. Em visão computacional, mosaicagem de imagens é o processo usado quando a cena a ser imagiada é maior que a imagem fornecida por uma única câmera, de tal forma que duas ou mais imagens são unidas recompondo a cena.

Este artigo descreve uma implementação simples em C++, usando basicamente correlação da OpenCV, capaz de construir um mosaico controlado com imagens provenientes de quatro câmeras dispostas na forma de matriz 2×2, visando uma aplicação industrial.

2. Introdução

Um mosaico, segundo [10], é um embutido de pequenas peças formando determinado desenho. O objetivo do mosaico é preencher algum tipo de plano.

Os primeiros mosaicos de imagens eram construídos manualmente a partir de fotografias impressas, por meio de superposição de trechos da cena, como comenta [1]. Atualmente com o baixo custo de câmeras digitais e dos computadores convencionais, é possível construir mosaicos computadorizados sem quaisquer equipamentos ou hardware dedicado.

Em visão computacional, mosaicagem de imagens é a técnica empregada quando a cena a ser imagiada é maior que o campo de visão da câmera. Ela une duas ou mais de partes (imagens) de uma mesma cena (que podem ser ou não da mesma câmera) de forma a produzir uma única imagem maior.

Diversos pacotes computacionais de processamento de imagens estão disponíveis no mercado. Ferramentas consagradas realizam com destreza tais tarefas, como é o caso do Matlab [7]. JAI (Java Advanced Imaging) [8] é uma API Java, gratuita e robusta para o processamento de imagens, mas não possui o menor tempo de execução dos algoritmos.

O projeto utiliza a OpenCV (Intel Open Source Computer Vision Library) [6], que permite produzir algoritmos complexos de processamento de imagens digitais com uma excelente eficiência, em alto a médio nível de programação C++.

2.1. Mosaicagem

A mosaicagem [2] é dividida em duas etapas fundamentais:

1. Captura de imagens − este trabalho presupõe-se que a captura imageie partes da mesma cena por mais de uma imagem, de forma a possibilitar a fusão das imagens em uma maior por meio dos trechos redundantes nas imagens. De acordo com as imagens capturadas, o mosaico pode ser construído com mais ou menos etapas de processamento de imagens.
   Para imagens aéreas por exemplo, é quase sempre necessário uma etapa de retificação de imagens.
2. Blending − do inglês, mistura. É o processo de se combinar imagens na área de sobreposição de forma que sua emenda fique imperceptível.

Segundo [2], mesmo para bons registros, é comum haver diferenças nos níveis de das imagens a serem concate-

nadas, o que gera a necessidade de suavizar a área de transição entre elas, o método mais utilizado para suavizar as regiões de transição é o da média ponderada, mas ainda existem casos que a transição pode ser perceptível ao se utilizar a interpolação de média ponderada, devido à por exemplo, diferenças de textura ou pequena área de superposição das imagens.

Uma observação importante para o bleding, é que o sistema visual humano possui maior sensibilidade para detectar mudanças de baixa freqüência, logo, uma linha de corte em regiões de alta freqüência deve ser menos perceptível, como comenta [4].

Quando a percepção da área de transição persiste, alguns autores como [5], [2] e [9] sugerem fortemente a multiresolução (wavelet). Esta abordagem possibilita que diferentes freqüências sejam analisadas separadamente em diferentes resoluções. Credita-se também o benefício desta abordagem ao fato de pode se escolher o tamanho da região de transição em função do nível de resolução escolhido.

[3] é um projeto capaz de realizar mosaicos a partir de imagens quaisquer, com irradiâncias diferentes, recortes, etc. Empregando "bag of features", são analisadas as características das imagens. Entretanto, é um algoritmo lento e que demo disponível distorce ligeiramente as imagens, impedindo análises dimensionais.

2.2. O problema

O cenário proposto compreende um ambiente de inspeção industrial com quatro câmeras posicionadas em forma de uma matriz 2×2, de acordo com a figura 1.

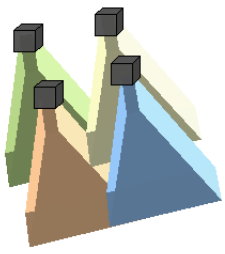


Figura 1. Representação de um sistema de inspeção industrial dotado de quatro câmeras dispostas na forma de matriz 2×2.

Desta forma, existem regiões da cena que são imagiadas por uma ou mais câmeras, como pode ser melhor visualizado na figura 2. Nela, pode ser observado que existe uma região comum entre as imagens da região 1 e da 2. Uma imagem resultante da agregação destas duas imagens terá outra região em comum com a região 3, assim como a região 4.

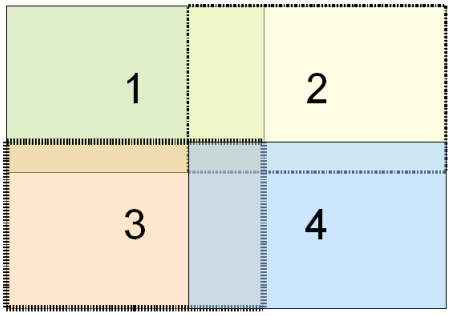


Figura 2. Vista superior, com as regiões imagiadas por quatro câmeras dispostas na forma de matriz 2×2, de forma a imagiar toda a cena

Desta forma, as imagens das regiões 1, 2, 3 e 4 podem ser agrupadas gerando um mosaico.

2.3. Objetivos

São objetivos deste artigo:

- Contruir mosaicos a partir de imagens que obedeçam o cenário descrito em 2.2;
- O algoritmo precisa ser executado no menor tempo possível;
- Não deve haver deformações relevantes na imagens, pois isto impossibilitaria uma análise dimensional (atividade comum na indústria);

2.4. Metodologia

O problema foi definido com as seguinte considerações:

1. Utilização da OpenCV em C++, de forma a privilegiar a performance do algoritmo;
2. O aplicativo sempre carregará quatro imagens de uma pasta, onda a primeira corresponde à região 1 , a segunda pertence à região 2, e assim sucessivamente até a região 4. Dessa forma, temos uma analogia com o cenário descrito em 2.2.
3. O tamanho das regiões comuns entre uma imagem e outra pode variar, mas sempre precisa existir;

4. a diferença de irradiância é desprezível, e não será necessário realizar suavizar as regiões de interseção;
5. as câmeras estão no mesmo plano, com a mesma distância da cena, suficientemente distantes dela que não será necessário realizar retificação de imagens;

A metodologia define as seguintes etapas para o desenvolvimento do aplicativo:

1. utilização de imagens sintéticas, recortadas simulando o ambiente descrito em 2.2;
2. inicialmente, todas as imagens devem estar em mesma escala;
3. o algoritmo deve ser empregado para imagens reais;
4. o aplicativo também deve realizar mosaicagem para imagens em diferentes escalas;
5. imagens rotacionadas também devem ser entradas do aplicativo;
6. imagens com alteração de escala e rotação devem ser entradas do aplicativo;
7. terminados todos estes itens, o aplicativo deve ser capaz de construir um mosaico com imagens recebidas de quatro câmeras descritas em 2.2;

2.5. Implementação do aplicativo

Dadas as necessidades expressas em 2.3, escolheu-se arbitrariamente que o aplicativo use como método a correlação de imagens. Naturalmente esta pode não ser a técnica mais indicada para criar mosaico a partir de quaisquer imagens, mas é uma técnica extremamente eficiente se utilizada adequadamente.

O aplicativo desenvolvido para resolver as questões de 2.3, possui 3 etapas:

1. carrega configurações de um arquivo texto;
2. carrega imagens e montas as matrizes conforme 2.2;
3. recorta um retângulo na borda superior esquerda da imagen 2 (este será o template);
   (a) converte esse retângulo (tempalte) para tons de cinza (gray scale);
   (b) cria uma cópia da imagem a qual se fará a correlação com o template;
   (c) converte esse copia de imagem para tons de cinza (gray scale);
   (d) faz correlação do template (gray scale) com a imagem vizinha (em gray scale);
   (e) procura o ponto de máximo da função de correlação e considera este com o ponto de interseção das imagens;
   (f) cria uma imagem maior capaz de comportar o agrupamento das imagens
   (g) sobrepõe a segunda imagem sobre a primeira formando um mosaico
4. realiza o item 2 para a imagens 3;
5. realiza o item 2 para a imagem 4;
6. encerra o algoritmo.

As configurações citadas são devido à não generalidade do método implementado. O tamanho do retângulo recortado é parâmetro fundamental para o bom funcionamento do algoritmo, desta forma, suas dimensões estão contidas no arquivo de configurações.

A correlação é feita em imagens em tons de cinza, segundo o método de correlação normalizada, $CV\_TM\_CCORR\_NORMED$ na OpenCV. Veja a figura 3 onde é representada a correlação. A método empregado implementa a função:

$$R(x,y) = sum_{x',y'} \frac{\left[T\left(x',y'\right) \bullet I\left(x+x',y+y'\right)\right]}{\sqrt{sum_{x',y'} T\left(x',y'\right)^2 \bullet sum_{x',y'} I\left(x+x',y+y'\right)^2}}$$

onde:

$$I \rightarrow imagem$$
$$T \rightarrow template \text{ (retângulo recortado)}$$
$$R \rightarrow resultado$$

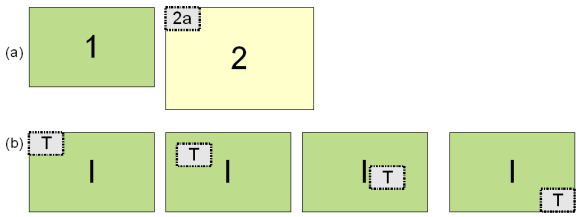


Figura 3. Representação da correlação. (a) É feito um recorte na borda superior esquerda da imagem 2, o 2a, ele será o template. Este template será convolvido na imagem 1, no caso genérico, o template T é convolvido pela imagem I

Apesar do grande número de operações apresentadas na função que implementa esta correlação, para templates relativamente pequenos, a OpenCV apresenta tempo de execução pequeno.

O resultado da função de correlação é uma matriz

com os valores da correlação para cada ponto da convolulção. Procura-se então o ponto de máximo, encontrado o ponto, as imagens são unidas partir de suas coordenadas, como pode ser visto na figura 4.

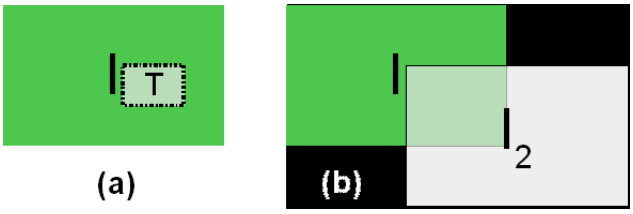


Figura 4. Blending (a) Representação do template que encontra o ponto de máximo na correlação (b) Utilizando este ponto de máximo, a imagem I_2 é unida à imagem I formando uma imagem maior

Em uma imagem colorida, cada pixel possui 3 canais, geralmente definidos por RGB (prefixo para Red, Green e Blue), ao converter uma imagens 3 canais para tons de cinza, cada pixel terá apenas 1 canal, e seu valor será a média dos seus canais em RGB. Desta forma, pixels com cores diferentes podem ter a mesma representação em tons de cinza, por exemplo:

Um pixel com valores RGB abaixo, terá como tom de cinza o valor 100:

$$R = 200$$
$$G = 100$$
$$B = 0$$

Já outro pixel pode ter também 100 como valor em tom de cinza e valor RGB de:

$$R = 150$$
$$G = 150$$
$$B = 200$$

Desta forma, a conversão perde informações e fará pixels diferentes serem considerados iguais pela correlação, o que prejudica os resultados do algoritmo.

Por se tratar de correlação de uma parte da imagem, ela pode resultar em mosaicos absurdos, por exemplo, numa parede de tijolos, se o retângulo recordado para ser o template contiver um tijolo, muito provavelmente ele poderá encontrar como ponto de máximo um outro tijolo em uma posição que o agrupamento destas imagens não seja o ideal.

2.6. Alteração de escala e rotação das Imagens

Pequenas alterações na escala e rotações das imagens são situações esperadas para o cenário 2.2. A solução é:

1. rotaciona-se iterativamente a imagem do menor, ao maior ângulo especificado pelo arquivo de configurações;
2. para cada valor de rotação, altera-se a escala, do menor ao maior valor especificado pelo arquivo de configurações;
3. para cada imagem resultante deste par rotação/escala, calcula-se a correlação;
4. armazena-se os valores da correlação e os parâmetros rotação/escala em um vetor;
5. procura-se o o maior valor de correlação neste vetor;
6. utiliza-se os parâmetros rotação/escala correspondentes ao maior valor de correlação para corrigir a imagem a ser agrupada para gerar um mosaico;

Como o valor de maior correlação deve ser o que melhor corrige a rotação e escala das imagens, espera-se que o mosaico deva ser gerado corretamente.

2.7. Resultados

No estado atual, não foram implementados a rotação e alteração de escala, logo apenas imagens com alteração de escala e rotação desprezíveis serão consideradas nos testes.

O mosaico foi realizado para imagens sintéticas (sem rotação e em mesma escala). Utilizando as imagens da figura 5, temos o resultado dos mosaicos na figura 6;

2.7.1. (Imagem Real)

Para evitar problemas de rotação e perspectiva, foram utilizadas as imagens reais distintas, obtidas com a câmera na mesma posição, imagiando a mesma cena. Entretanto, essas imagens não obedecem o cenário de 2.2, desta forma, recortou-se partes distintas de quatro destas imagens. Agora, estas imagens menores resultantes, obedecem ao cenário 2.2. Veja a figura 7 onde são apresentados a mesma cena, em quatro imagens, e com a representação dos trechos recortados destas imagens, de forma a simular uma captura por quatro câmeras.

Veja as figura 8 e 9.

Figura 5. Imagens sintéticas posicionadas na forma de matriz 2×2

2.8. Conclusões e continuação deste trabalho

A partir das restrições impostas ao projeto em 2.3 e as considerações de 2.4,os testes se deram para imagens sintéticas e imagens reais de forma a simular a captura pelas quatro câmeras.

Os resultados dos mosaicos gerados foram excelentes visualmente, especialmente visto que não foi preciso suavizar as imagens.

Outro ponto interessante é o baixo tempo de execução do algoritmo, que se mostra totalmente viável para aplicações de tempo real, que é geralmente um anseio da indústria.

Entretanto, partes deste trabalho ficaram pendentes mas, com a metodologia para a implementação já descrita em 2.6. Estou pendentes:

1. finalizar a correção de escala das imagens
2. implementar a rotação de imagens
3. testes com imagens reais sujeitas a mudanças de escala e rotação

Referências

[1] E. R. d. A. J. Arruda. Mosaicagem de imagens digitais.

[2] F. L. M. G. Bagli, Vantier Veronezi. Emprego de análise em multiresolução para mosaicagem de imagens de sensoriamento remoto.

[3] M. Brown. Autostitch. Technical report, The University of British Columbia, http://www.cs.ubc.ca/ mbrown/autostitch/autostitch.html, 2007. acessado em 16/05/2007.

[4] W. R. E. Gonzalez, Rafael C. *Digital Imaging processing*. Editora Edgar Blucher Ltda, São Paulo, 2000.

[5] W. J. L. Hsu, C. T. Multiresolution mosaic.

[6] Intel. Intel open source computer vision library. Technical report, Intel, http://www.intel.com/technology/computing/opencv/, 2007. acessado em 16/05/2007.

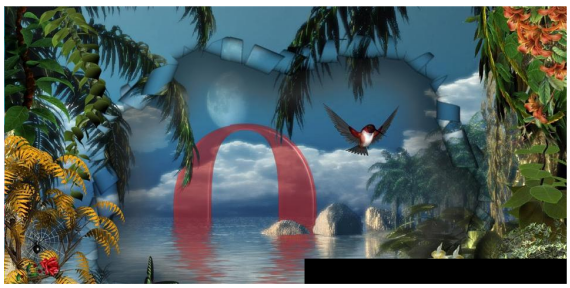
[7] T. MathWorks. Matlab. Technical report, The MathWorks, http://www.mathworks.com/, 2007. acessado em 16/05/2007.

[8] S. Microsystens. Jai java advanced imaging. Technical report, Sun Microsystens, http://java.sun.com/products/java-media/jai/iio.html, 2007. acessado em 16/05/2007.

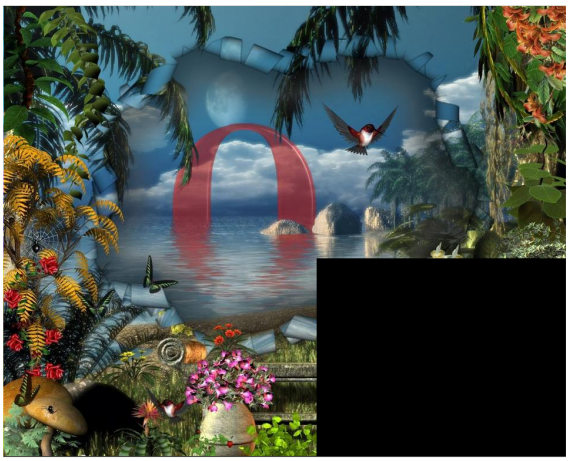
[9] H. W. L. C. K. Y. Su, M. S. Analysis on multiresolution mosaic images.

[10] Wikipedia. Mosaico. Technical report, Wikipedia, http://pt.wikipedia.org/wiki/Mosaico, 2007. acessado em 16/05/2007.

(a)



(b)



(c)

Figura 6. Mosaicagem: (a) Mosaico da imagem 1 com imagem 2 formando a primeira linha da matriz 2×2 (b) Adição da imagem 3 ao mosaico (c) Mosaico final para as imagem de 5

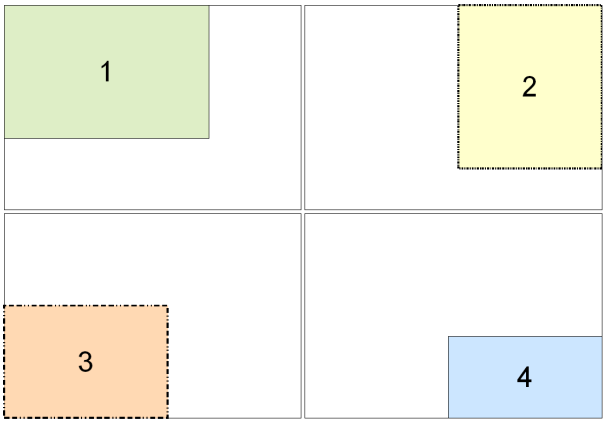


Figura 7. Representação da obtenção do cenário 2.2 a partir de quatro imagens distintas da mesma cena.

Figura 8. Imagens reais usadas no teste

Figura 9. Mosaico obtido para imagens reais