http://journal.unoeste.br/index.php/ce/index
DOI: 10.5747/ce.2021.v13.n2.e357
ISSN on-line 2178-8332

Colloquium Exactarum

Submetido: 13/03/2021 Revisado: 10/05/2021 Aceito: 20/07/2021

CONSTRUÇÃO DE IMAGENS PANORÂMICAS EM MÚLTIPLAS FAIXAS DE ALTURA E LARGURA USANDO ALGORITMOS WATERSHED E GRAPH-CUT

Construction of Panoramic Images in multiple height and width ranges using Watershed and Graph-cut algorithms

Caio Chizzolini Silva[1], Francisco Assis da Silva[1], Leandro Luiz de Almeida[1], Danillo Roberto Pereira[1], Almir Olivette Artero[2], Marco Antônio Piteri[2]

[1]Faculdade de Informática de Presidente Prudente, Unoeste - Universidade do Oeste Paulista, Presidente Prudente
caiochizzolini@hotmail.com, chico@unoeste.br, llalmeida@unoeste.br, danilopereira@unoeste.br
[2]Faculdade de Ciências e Tecnologia, UNESP - Universidade Estadual Paulista
Departamento de Matemática e Computação, Presidente Prudente
almir.artero@unesp.br, marco.piteri@unesp.br

RESUMO – Neste trabalho foi desenvolvido um algoritmo para construção de panoramas de imagens com múltiplas faixas de altura e largura. Para a costura multilinear, as imagens foram inicialmente colocadas em uma matriz e foram gerados panoramas parciais com imagens da mesma coluna. Para completar o panorama final, as colunas foram divididas com o auxílio de pontos de apoio e as colunas vizinhas foram costuradas, sendo remontadas ao final do processo. A costura foi realizada com um algoritmo de corte de grafo em conjunto com o algoritmo Watershed.
Palavras-chave: Panorama de imagens, Costura de imagens, Graph-cut, Watershed, SIFT, RANSAC.

ABSTRACT – In this work we developed an algorithm for building panoramas with multiple height and width ranges. For multilinear stitching, images were initially placed in a matrix and partial panoramas with images from the same column are generated. To complete the final panorama, the columns were divided with the help of support points and the neighboring columns were stitched, being reassembled at the end of the process. The stitching was performed with a graph cut algorithm in conjunction with the Watershed algorithm.
Keywords: Image Panorama, Stitching, Graph-cut, Watershed, SIFT, RANSAC.

1. INTRODUÇÃO

Uma das atividades para a construção de um panorama é a costura de imagens, que consiste em unir duas ou mais imagens (NYMAN, 2010). A geração de um panorama simples, com apenas uma faixa de extensão já foi muito estudada e diversos algoritmos que apresentam bons resultados já foram propostos (NYMAN, 2010; SZELISKI, 2006; ZHENG *et al*., 2018).

A construção de um panorama pode ser dividida em três etapas, sendo elas, o posicionamento de imagens, a costura e a correção (NYMAN, 2010).

A etapa de posicionamento tem como objetivo determinar a posição de cada imagem em relação as demais, para isso são fornecidas as imagens iniciais, e ao final do processo obtêm-se as transformações necessárias para que cada imagem seja posicionada corretamente no panorama final (NYMAN, 2010).

A etapa de costura consiste em identificar quais pixels de cada imagem serão mantidos ou removidos e realizar a junção entre as imagens (ZHENG *et al*., 2018). Um dos métodos existentes para realizar a costura é a sobreposição simples, onde ao realizar a junção, os pixels da primeira imagem serão substituídos pelos da segunda, em áreas onde ocorra a sobreposição das duas (NYMAN, 2010).

Embora o método da sobreposição seja rápido, ele possui grandes chances de gerar ruídos, objetos cortados ou regiões desproporcionais. A fim de minimizar esses erros, foram desenvolvidos algoritmos que calculam a melhor rota possível para que seja feita a costura por sobreposição (SZELISKI, 2006).

Como imagens capturadas de ângulos ou perspectivas diferentes podem possuir diferença de cor e iluminação, a etapa de correção tenta remover ou tornar imperceptível essas diferenças, de forma que no panorama final não seja possível perceber a costura e as diferentes imagens utilizadas em sua construção. Embora sejam etapas separadas, a correção pode ser feita durante o processo de costura para diminuir as taxas de erros e gerar resultados melhores (ZHENG *et al*., 2018).

Os panoramas mais simples, como os obtidos por câmeras de dispositivos móveis, se limitam a construir uma imagem maior em apenas uma faixa determinada, sendo vertical ou horizontal. Embora existam algoritmos para construção de panoramas com mais de uma faixa, poucos apresentam resultados satisfatórios, tornando possível o estudo de novos algoritmos que realizem a costura em múltiplas faixas (ZHENG *et al*., 2018; RU *et al.*, 2016).

Esse trabalho busca contribuir com um algoritmo para geração de panoramas em múltiplas faixas de altura e largura. A construção do panorama é realizada a partir de imagens de entrada, que são comparadas e costuradas em faixas verticais utilizando-se da correspondência entre imagens e corte de grafos. Esses panoramas intermediários são então costurados resultando em um panorama com múltiplas faixas.

Após esta seção de introdução, o trabalho está organizado da seguinte maneira. Na Seção 2 são descritos os conceitos fundamentais contendo os métodos utilizados no desenvolvimento deste trabalho. Na Seção 3 é apresentado o método proposto, bem coma a avaliação dos resultados obtidos. Por fim, na Seção 4 encontram-se as conclusões e propostas de trabalhos futuros.

2. CONCEITOS FUNDAMENTAIS

Nesta seção é apresentada a fundamentação teórica sobre os métodos utilizados para o desenvolvimento deste trabalho. O algoritmo SIFT foi utilizado para detecção e descrição de pontos chave nas imagens. O algoritmo RANSAC foi utilizado para calcular matrizes homográficas a serem aplicadas nas imagens para a correção de perspectiva. Para o cálculo da costura, foi utilizado o algoritmo *Min-Cut*/*Max-Flow* em conjunto com o algoritmo de cálculo de Imagem de Diferença e o algoritmo *Watershed*. Para auxiliar a costura em múltiplas faixas, *Apoints* e *Bpoints* foram calculados.

2.1. SIFT

O algoritmo SIFT (*Scale Invariant Feature Transform*) é um método muito eficiente para identificar e descrever pontos chave em imagens, resultando a posição *x*, *y*, escala, orientação θ e em um vetor com 128 valores, descrevendo cada ponto chave da imagem (LOWE, 2004).

O algoritmo pode ser dividido em duas partes, o detector e o descritor. O detector baseia-se em cálculos de diferença de filtros Gaussianos, enquanto o descritor utiliza-se de histogramas de gradientes orientados para descrever a vizinhança local dos pontos de interesse (SILVA *et al*., 2017).

Seu funcionamento é dividido em quatro

etapas principais, sendo as duas primeiras a parte do detector dos pontos chave, e as duas últimas a do descritor.

A primeira etapa, detecção de extremos no espaço escala, busca pontos que são invariáveis à escala e orientação ao se utilizar de diferença de filtros Gaussianos. O espaço escala de uma imagem pode ser definido pela Equação 1, onde *G*(*x*, *y*, *σ*) representa uma convolução de uma Gaussiana de escala variável e *I*(*x*, *y*) representa a imagem de entrada (KOENDERINK, 1984) (LINDEBERG, 1994).

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \quad (1)$$

A gaussiana *G*(*x*, *y*, *σ*) pode ser definida pela Equação 2.

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2} \quad (2)$$

Para a detecção de pontos chave estáveis no espaço escala Lowe (1999) propõe o uso de extremos do espaço escala com o uso de uma função de diferença de Gaussianas (DoG) (LOWE, 1999), que é calculada por pela Equação 3, onde *k* é uma constante multiplicadora. O DoG é uma aproximação do Laplaciano normalizado em escala da Gaussiana $\sigma^2\nabla^2 G^2$ (MIKOLAJCZYK; SCHMID, 2005).

$$D(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) \quad (3)$$

Na segunda etapa, localização de pontos chave, a partir de $D(x, y, \sigma)$, o algoritmo percorre áreas nas quais foram detectados extremos (máximos e mínimos), e detecta pontos chave comparando cada pixel com seus oito vizinhos da escala atual e nove vizinhos das escalas superior e inferior, totalizando 26 vizinhos. O SIFT garante que os pontos chave são localizados em regiões e escalas com alta variação, o que os torna estáveis para caracterizar a imagem (LOWE, 2004).

Na etapa de determinação da orientação, utiliza-se o gradiente local da imagem para determinar uma ou mais orientações para cada ponto chave, que serão utilizados para prover a invariância em orientação, escala e localização. O gradiente pode ser calculado pela Equação 4.

$$m(x, y) = \sqrt{\Delta x^2 + \Delta y^2} \quad (4)$$

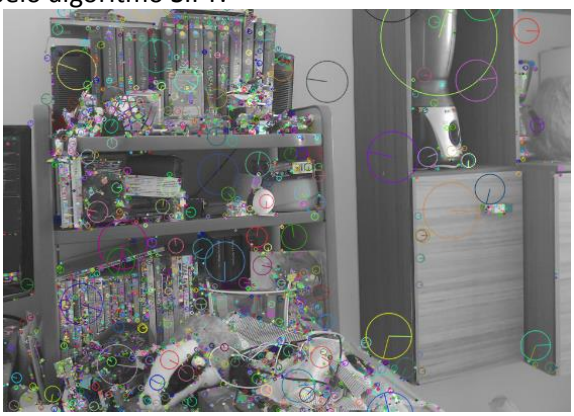
onde $\Delta x = L(x + 1, y) - L(x - 1, y)$ e $\Delta y = L(x, y + 1) - L(z, y - 1)$. A orientação é calculada pela Equação 5.

$$\theta(x, y) = tan^{-1}(\Delta y / \Delta x) \quad (5)$$

A última etapa do algoritmo SIFT, calcula os descritores com uso de histogramas de orientação. Estes são gerados ao medir gradientes locais da imagem em uma escala selecionada e então transformar essas medidas para uma representação que permita observar níveis de distorção e mudanças de iluminação.

Os pontos descritos pelo SIFT, por serem razoavelmente invariantes a mudanças de iluminação, ruídos, escala, rotação e pontos de vista 3D, podem servir como base para comparação entre imagens ou detecção de objetos. Na Figura 1 é apresentado um exemplo de pontos chave detectados e descritos pelo algoritmo SIFT.

Figura 1. Pontos chave detectados e descritos pelo algoritmo SIFT.



Fonte: (Autor, 2020).

2.2. *Matching*

Com os pontos chave descritos pelo SIFT é possível encontrar correspondência entre imagens (*matching*). A melhor correspondência para cada ponto chave é encontrada identificando o vizinho mais próximo, que é definido ao minimizar a distância euclidiana dos vetores de características (LOWE, 2004). Para evitar a busca exaustiva, é sugerido utilizar uma estrutura de dados *k-d tree* (BEIS; LOWE, 1997) que se utiliza de busca binária balanceada para encontrar os vizinhos próximos

2.3. RANSAC

O algoritmo RANSAC (*RANdom SAmple Consensus*) é um método de estimação robusto amplamente utilizado no reconhecimento de objetos em imagens (OKABE; SATO, 2003). Ele tem como função a extração de dados relevantes

para um determinado modelo desejado (*inliers*), bem como estimar uma matriz fundamental *M* entre duas imagens.

Ao contrário das técnicas convencionais que utilizam muitos dados para obter uma solução inicial e então eliminam os dados que não se ajustam ao modelo (*outliers*), o RANSAC utiliza apenas um conjunto com número mínimo e suficiente de pontos necessários para a primeira estimativa e então procede aumentando o conjunto com dados consistentes (FISCHLER; BOLLES, 1981).

O algoritmo seleciona aleatoriamente um conjunto mínimo de pontos, que será o modelo inicial. Os demais pontos que estiverem dentro de uma distância *t* do modelo serão classificados como *inliers*. Se o total de *inliers* for menor que o limite definido, o modelo é descartado e um novo é gerado. Caso o número de *inliers* ultrapasse o limite, um novo modelo é gerado com base nos *inliers* encontrados e o algoritmo é repetido *N* vezes. Ao fim das iterações, o grupo com maior número de *inliers* é escolhido.

O número de iterações *N* deve ser o suficiente para garantir pelo menos um grupo sem a presença de *outliers*, com uma probabilidade *p*. *N* pode ser calculado pelas Equações 6 e 7, onde *w* é a probabilidade de qualquer dado ser um *inlier.*

$$(1 - w^s)^N = 1 - p \quad (6)$$

$$N = \frac{\log(1-p)}{\log(1-(1-\varepsilon)^s)} \quad (7)$$

Na correspondência entre imagens, o RANSAC possibilita encontrar as correspondências características geometricamente consistentes usadas para resolver a homografia, ou matriz fundamental, entre pares de imagens, possibilitando a construção de panoramas de imagens (BROWN; LOWE, 2007).

2.4. Imagem de diferença

Uma imagem de diferença (*DImage*) é gerada a partir da área sobreposta entre duas imagens, onde cada pixel da *DImage* representa o inverso do pixel corresponde na área sobreposta (NYMAN, 2010).

Cada pixel da *DImage* pode ser calculado pela Equação 8, onde I_n representa os pixels das imagens na área sobreposta.

$$P(x,y) = 255 - max\big(I_1(x,y) - I_2(x,y)\big) \quad (8)$$

O algoritmo tem maior eficácia em imagens em tons de cinza, tendo como resultado final uma imagem onde se é possível distinguir o contorno geral de objetos maiores e objetos menores são removidos. Na Figura 2 é possível reconhecer objetos grandes como teclado, mouse e canto de mesa, na parte inferior da imagem, enquanto não é possível distinguir outros objetos menores ao centro da imagem.

Figura 2. *DImage* gerada em uma sobreposição de duas imagens.

Fonte: (Autor, 2020).

Ao gerar uma *DImage*, é possível eliminar ruídos e agregar regiões semelhantes dentro da área sobreposta, simplificando a utilização de outros algoritmos que possam utilizar-se da mesma área (NYMAN, 2010).

2.5. Algoritmo *Watershed*

Também conhecido como algoritmo da Barragem, o *Watershed* é um algoritmo utilizado para detecção de bordas e objetos em imagens. Seu funcionamento consiste em inundar a imagem de forma que a mesma seja segmentada em regiões diferentes (GRACIAS *et al*., 2009).

Toda imagem em tons de cinza pode ser vista como uma imagem topológica, onde regiões de maior intensidade de cor são ditas como picos

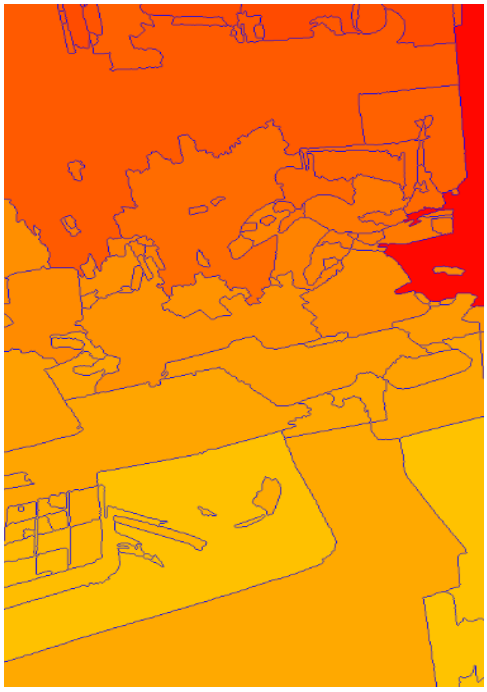
e regiões com menor intensidade são ditos vales (VINCENT; SOILLE, 1991).

Conforme os vales vão sendo inundados com valores diferentes em cada um, é possível que vales diferentes se conectem, e para evitar isso são feitas barragens entre eles. A inundação persiste até que todos os picos estejam inundados. O nome do algoritmo deriva desse conceito (BEUCHER, 2020).

Como é possível existir muitos vales superficiais por causa de ruídos na imagem, é necessário direcionar a inundação do algoritmo. Para isso é utilizada uma matriz marcador (*marker*) que indica quais regiões serão vales a serem inundados e quais serão ignorados. A matriz *marker* possui o mesmo tamanho da imagem, porém, o valor de seus pixels é representado por 0 em regiões a serem ignoradas e valores inteiros positivos para as regiões inundadas.

Ao aplicar o algoritmo na imagem, os valores dos pixels da matriz *marker* serão atualizados, alterando para o valor -1 os pixels que representam a borda das regiões encontradas. A matriz *marker* pode então ser aplicada à imagem para que as regiões sejam visíveis. Na Figura 3 é apresentado um exemplo de imagem segmentada, onde as bordas são representadas por pixels azuis enquanto as diferentes regiões foram coloridas em tons de amarelo e vermelho.

Figura 3. Exemplo de imagem segmentada pelo *Watershed*.



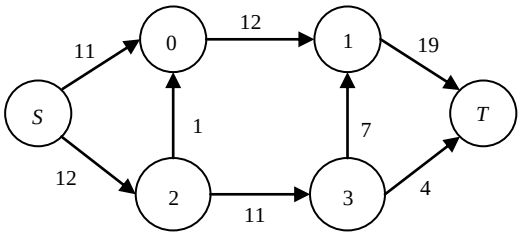
Fonte: (Autor, 2020).

2.6. *Min-Cut/Max-Flow*

O problema do corte mínimo (*Min-Cut*) ou fluxo máximo *(Max-Flow*) de grafos se encaixa no estudo de redes de transporte (DANTZIG; FULKERSON, 1964). Dado um grafo ponderado com um nó de fonte (*Source*) *S*, um nó de sumidouro (*Sink*) *T*, e outros nós intermediários (Figura 4), o problema do *Min-Cut* busca encontrar a menor soma total de pesos de forma que ao se remover as arestas correspondentes, *S* seja desconectado de *T,* enquanto o problema do *Max-Flow,* busca encontrar a maior soma possível do fluxo que leva de *S* à *T.* É possível observar um exemplo de *Min-Cut* na Figura 5, onde o corte encontrado é representado pela linha tracejada em vermelho e a soma das arestas retiradas abaixo da imagem é o valor do corte mínimo.

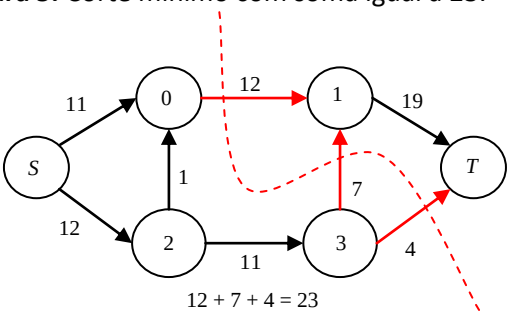
Na Figura 6 encontra-se um exemplo de *Max-Flow*, onde os diferentes fluxos são representados nas cores vermelho e azul e o fluxo máximo da rede, apresentado abaixo da imagem é dado pela soma do valor resultante em cada fluxo.

Figura 4. Grafo ponderado com *Source S* e *Sink T*.



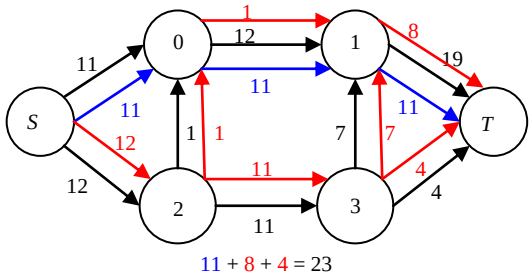
Fonte: (Autor, 2020).

Figura 5. Corte mínimo com soma igual a 23.



Fonte: (Autor, 2020).

Figura 6. Fluxo Máximo com soma igual a 23.



Fonte: (Autor, 2020).

O teorema "*Min-Cut Max-Flow*" proposto por Ford e Fulkerson (DANTZIG; FULKERSON, 1964) afirma que, o valor do fluxo máximo de *S* à *T* para qualquer rede de grafos é igual à soma mínima de pesos para se constituir um corte na mesma rede, ou seja, ao encontrar o *Max-Flow*, também é possível determinar o *Min-Cut* para a mesma rede.

2.7. Pontos de base

Ao dividir uma imagem em duas ou mais partes, é possível armazenar as coordenadas da divisão realizada, em pontos de base (*Bpoints*) (ZHENG *et al.*, 2018). Na construção de panoramas em múltiplas faixas, os *Bpoints* auxiliam a unir as imagens sem que as costuras anteriores sejam afetadas por transformações ou deformações aplicadas anteriormente.

O método mais simples para o cálculo dos *Bpoints* é armazenar os pixels da lateral criada ao dividir a imagem em intervalos regulares usando a Equação 9, onde *x* e *y* representam as coordenadas do primeiro pixel da borda criada, *k* representa o intervalo entre cada ponto e *i* e *j* controlam a variação horizontal e vertical respectivamente.

$$B = P(x + k * i, y + k * j) \quad (9)$$

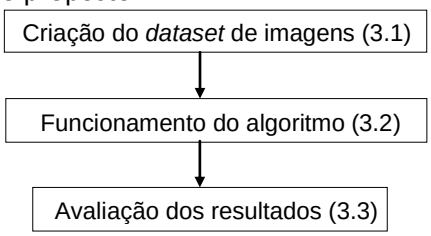
2.8. Pontos de apoio

Quando uma imagem sofre uma transformação, suas extremidades tendem a se deformar, causando aos objetos presentes nela a mesma deformação (ZHENG *et al.*, 2018). Para diminuir essa deformação causada, é possível utilizar os pontos de apoio (*Apoints*), que indicam a posição de uma região da imagem que não pode ser alterada ao se aplicar transformações. Para a construção de panoramas, os *Apoints* auxiliam a etapa de costura, para que as transformações ocorram apenas na área desejada e as deformações nas bordas sejam diminuídas.

3. MÉTODO PROPOSTO

Nesta seção é descrito o funcionamento do método proposto, sendo dividido em três etapas: criação do *dataset* de imagens utilizado nos experimentos, funcionamento do algoritmo e avaliação dos resultados (Figura 7).

Figura 7. Fluxograma representando as etapas do método proposto.



Fonte: (Autor, 2020).

3.1. Criação do dataset de imagens

O *dataset* utilizado neste trabalho é constituído por imagens com resolução Full HD (1920x1080) que foram capturadas a partir de uma câmera de um dispositivo móvel Xiaomi Redmi 8. Cada caso de teste (total de 100 casos) é composto por nove imagens. As imagens foram obtidas de pontos de vista diferentes, para que assim tenham diferentes ângulos, posicionamentos e iluminação, e favoreçam a análise da metodologia proposta, considerando a detecção e cálculo da costura sob diferentes circunstâncias. Alguns exemplos dos posicionamentos das câmeras utilizados na captura são mostrados na Figura 8. As imagens foram capturadas em ambientes internos de uma residência e em ambientes externos. O *dataset* de imagens, bem como os resultados obtidos estão disponíveis em: https://www.github.com/CaioChizzolini/dataset-panoramas.

Figura 8. Exemplo de caso de teste com nove imagens obtidas com a câmera utilizada, demonstrando os diferentes pontos de vista de captura.

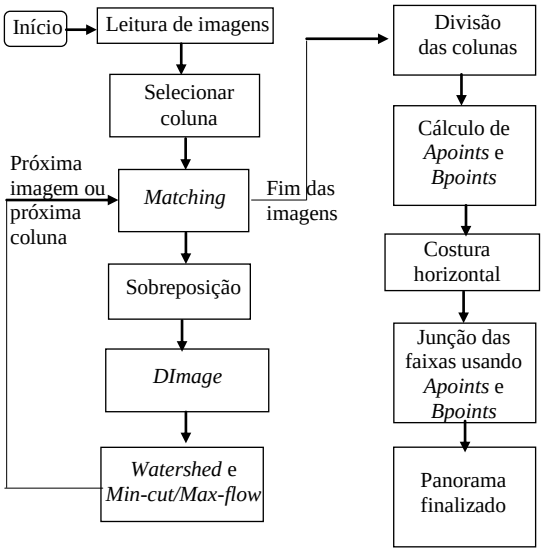
Fonte: (Autor, 2020).

3.2. Funcionamento do Algoritmo

O algoritmo desenvolvido é dividido em duas etapas principais, sendo elas a costura de panoramas verticais e a costura multilinear utilizando-se de *Bpoints* e *Apoints*. O fluxograma de funcionamento é mostrado na Figura 9.

O algoritmo foi desenvolvido na linguagem de programação Python utilizando a biblioteca de visão computacional OpenCV. O algoritmo limita-se na junção de nove imagens, dispostas em uma matriz 3x3. A primeira etapa é responsável pela geração de panoramas intermediários que fazem parte do panorama final.

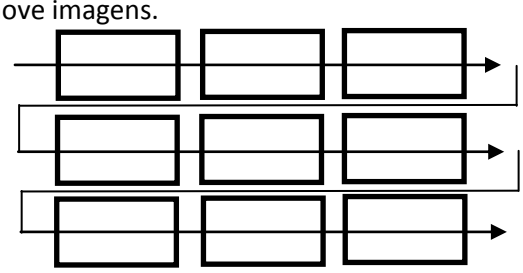
Figura 9. Fluxograma de funcionamento do algoritmo.



Fonte: (Autor, 2020).

Inicialmente, é feita a leitura das imagens de entrada. As imagens são lidas como uma matriz quadrada, linha por linha, conforme apresentando na Figura 10. Ao final da leitura, com a matriz preenchida, é possível identificar a vizinhança de cada imagem, eliminando a necessidade de comparar todas as imagens entre si para calcular a vizinhança individualmente.

Figura 10. Ordem de leitura para um conjunto de nove imagens.



Fonte: (Autor, 2020).

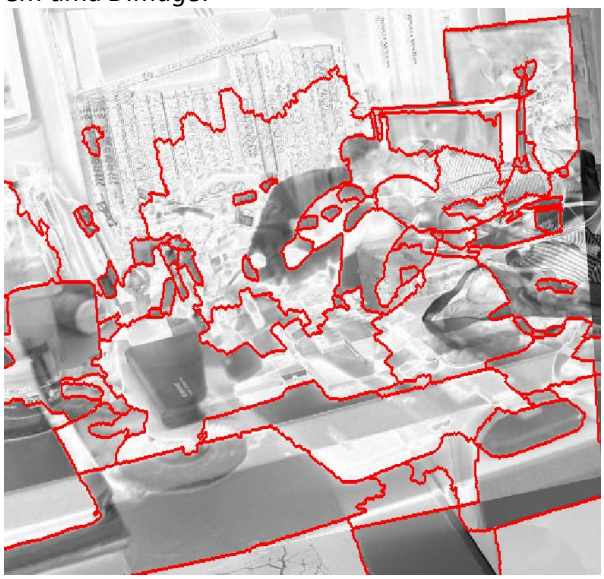
Os panoramas intermediários são gerados através da costura das imagens de uma mesma coluna na matriz, obtendo-se panoramas verticais. Para iniciar a costura, foi utilizado o algoritmo SIFT (LOWE, 2004) para detectar e descrever os pontos chave das imagens e a partir destes realizar a correspondência entre as imagens (*matching*). O algoritmo RANSAC (FISCHLER; BOLLES, 1981) é então utilizado para separar as informações relevantes e se obter a matriz homográfica, com as transformações necessárias a serem aplicadas nas imagens. Dessa forma é possível identificar a área de sobreposição entre as imagens. A costura utilizada é calculada através do problema do

corte mínimo, sendo necessário gerar um grafo a partir das imagens para que seja possível aplicar o mesmo.

O grafo é obtido pelo uso conjunto de dois algoritmos, sendo eles, o algoritmo de geração de imagem de diferença, que produz uma *DImage* a partir da área de sobreposição entre as duas imagens a serem costuradas e o algoritmo *Watershed* (VINCENT; SOILLE, 1991; BEUCHER, 2020) que segmenta a *DImage* em regiões distintas. Na Figura 11 são apresentadas as regiões encontradas, delimitadas pela cor vermelha.

Com as regiões identificadas na *DImage*, o grafo é construído, onde cada região encontrada representa um nó. As bordas entre as diferentes regiões são as arestas, e a quantidade total de pixels nas bordas são os pesos das arestas. Calculando o corte mínimo da rede, é possível armazenar as arestas que fazem parte do corte, sendo estas arestas, o caminho de pixels que representa a costura do panorama vertical.

Figura 11. Regiões encontradas pelo *Watershed* em uma *DImage*.



Fonte: (Autor, 2020).

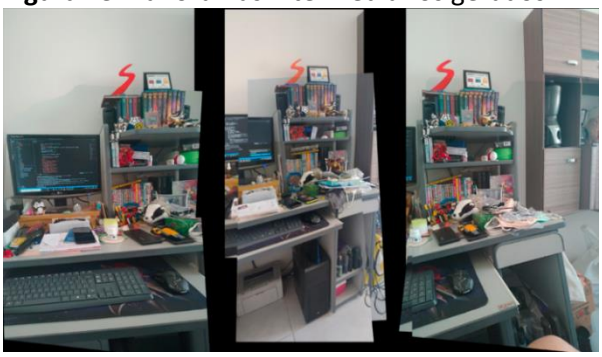
Na Figura 12 é apresentada a costura ideal em vermelho, encontrada ao aplicar o corte mínimo no grafo gerado. Por ser a costura com menor comprimento em pixels, a chance de se obter objetos cortados ou ruídos é menor (NYMAN, 2010).

Figura 12. Costura ideal calculada utilizando corte de grafos.

Fonte: (Autor, 2020).

Na Figura 13 são mostrados os panoramas intermediários construídos pelo algoritmo proposto.

Figura 13. Panoramas intermediários gerados.

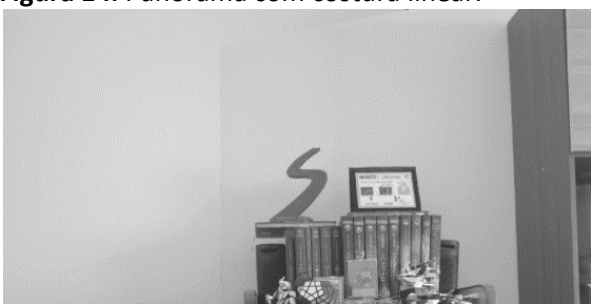


Fonte: (Autor, 2020).

Um problema encontrado no cálculo da costura foi no caminho de pixels calculado em regiões que apresentam simultaneamente cores iguais, poucos detalhes e superfícies extensas, como paredes, mesas, ruas de asfalto e céu. Para esses casos, principalmente quando essas regiões se localizam nas bordas das imagens de entrada, a costura tende a ser linear e diferenças de cor ou iluminação entre as imagens são ressaltados, tornando a costura visível no resultado final. Com isso, foi necessário um algoritmo de correção para suavizar essa diferença. O algoritmo utilizado foi a interpolação dos pixels na região da costura (HIRAGA *et al*., 2012). Na Figura 14 é possível observar a diferença de iluminação gerada na costura em uma parede, devido ao processo de costura linear calculada pelo

algoritmo.

Figura 14. Panorama com costura linear.



Fonte: (Autor, 2020).

Ao serem construídos todos os panoramas intermediários, o algoritmo passa para a próxima etapa, que consiste em unir as colunas e gerar o panorama final. Como cada imagem sofreu transformações diferentes para a geração das colunas, a chance de se encontrar uma transformação que possibilite a costura entre colunas é baixa. Para poder então realizar a costura, foi necessário repartir as colunas novamente em imagens menores e calcular individualmente a costura com a coluna vizinha. Na Figura 15 é apresentado um exemplo de divisão de colunas, onde ambas as colunas foram divididas em duas partes e serão costuradas horizontalmente entre si.

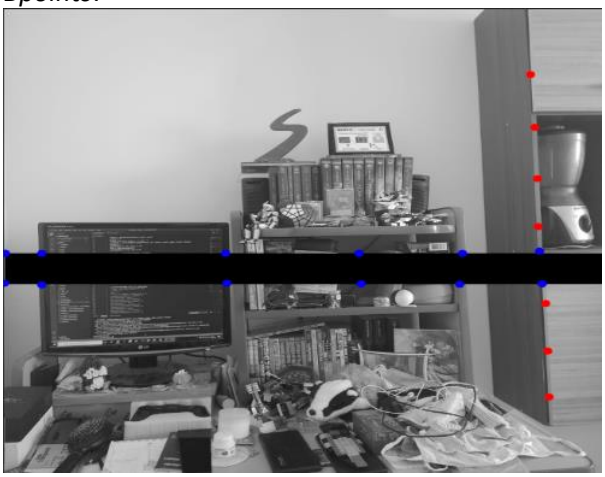
Ao realizar esse processo, caso a imagem seja deformada, a costura já efetuada pode ser perdida. Para evitar isso, são utilizados os *Apoints* e *Bpoints.* Os *Bpoints* são usados para demarcar a área onde as colunas foram divididas em ambas as imagens, de modo que ao realizar a costura horizontal, a correspondência com as imagens da mesma coluna seja mantida.

Figura 15. Panoramas intermediários divididos para costura horizontal.

Fonte: (Autor, 2020).

Os *Apoints* serão aplicados fora da área de sobreposição e as regiões da imagem com presença de *Apoints* não sofrerão deformações durante o processo de costura. Na Figura 16 são mostrados os *Apoints* em vermelho e os *Bpoints* em azul.

Figura 16. Representação visual de *Apoints* e *Bpoints*.



Fonte: (Autor, 2020).

O processo de costura entre as colunas é aplicado do centro da matriz para as bordas, ou seja, uma coluna central é definida e as demais são costuradas em relação a ela. Ao final dessa etapa é obtido o panorama com múltiplas faixas.

3.3. Avaliação dos Resultados

Para a avaliação dos resultados foi levado em consideração apenas o desempenho dos algoritmos desenvolvidos neste trabalho. Por se tratar de um trabalho com foco em costura de imagens, foi avaliada a taxa de acerto do algoritmo, que representa a quantidade de casos de teste que efetivamente geraram um panorama, a deformação visual apresentada nos panoramas, bem como o tempo de execução do algoritmo.

Os resultados dos testes da costura podem ser visualizados na Tabela 1. Para cada caso de teste foram analisadas as costuras do primeiro panorama intermediário gerado. Como cada panorama intermediário é composto por três imagens, cada caso de teste teve duas costuras analisadas, totalizando 200 costuras. A costura é considerada finalizada quando foi possível determinar o caminho e unir as imagens. A eficácia representa a porcentagem de costuras com aspecto visual natural, sem diferenças exageradas de cor e iluminação. O algoritmo apresenta uma taxa de acerto alta, porém sua

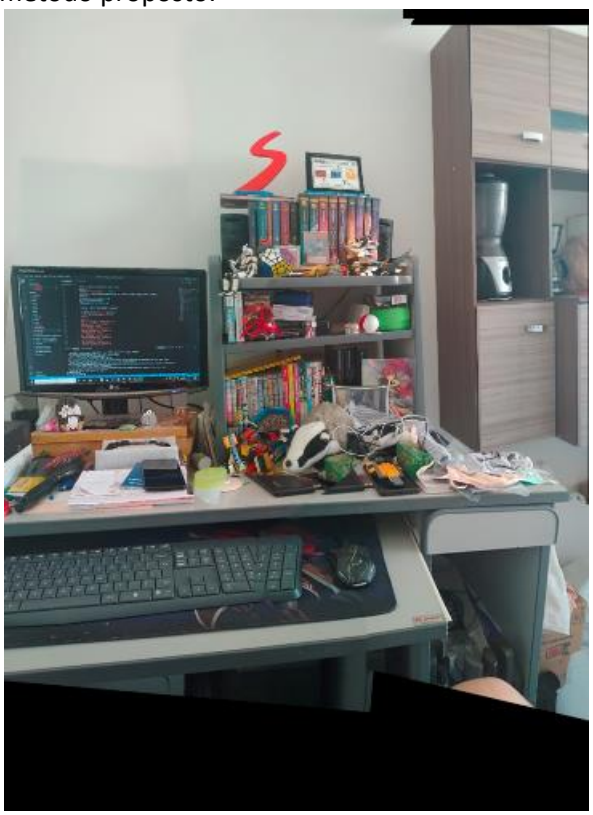
eficácia é fortemente ligada às características das imagens utilizadas.

Tabela 1. Resultado da Costura individual.

Costuras avaliadas	Costuras finalizadas	Taxa de acerto	Eficácia da costura
200	186	93%	84%

Para realizar a avaliação final dos panoramas construídos, foram utilizados 100 casos de teste, onde cada caso é composto de nove imagens a serem costuradas. Na Figura 17 é apresentado um panorama construído pelo algoritmo, onde é possível observar que as bordas do panorama obtido não possuem muitas deformações, porém apresenta algumas mudanças de iluminação.

Figura 17. Panorama final gerado utilizando o método proposto.



Fonte: (Autor, 2020).

O método proposto atingiu resultados satisfatórios na geração de panoramas com múltiplas faixas de altura e largura, com uma alta taxa de acerto e baixo número de casos onde a deformação é visível. Os resultados podem ser observados na Tabela 2. O tempo de execução será dependente das dimensões da imagem, bem como do comprimento da costura encontrada. Os experimentos foram realizados em um computador com processador AMD Ryzen 5 2400G, podendo ter uma diminuição de tempo de execução para processadores mais recentes.

Tabela 2. Resultados finais do algoritmo.

Casos de teste	Taxa de acerto	Panoramas com deformações visíveis	Média do tempo de execução
100	87%	16	33.5s

4. CONCLUSÕES

Os resultados obtidos com este trabalho mostram que a metodologia desenvolvida pode ser aplicada para a geração de panoramas com múltiplas faixas com boa qualidade de imagem.

Este trabalho poderá ser utilizado em áreas que necessitem de imagens mais abrangentes, como geração de mapas aéreos ou mapeamento de rios, cobrindo uma área maior do que em panoramas lineares.

Como trabalhos futuros, melhorias podem ser realizadas, como a utilização de outros algoritmos para o cálculo da costura que possam otimizar o caminho encontrado bem como o tempo total de execução. Para as questões melhoria de desempenho poderia tornar o algoritmo paralelizável para aproveitar os recursos de processador com mais de um núcleo. Também é possível utilizar um método de leitura diferente para que seja possível adicionar novas faixas de imagens ao panorama final.

REFERÊNCIAS

NYMAN, P. **Image stitching using watersheds and graph cuts**. Centre for Mathematical Sciences, Lund University, Sweden, 2010.

SZELISKI, R. Image alignment and stitching: A tutorial. Foundations and Trends® **Computer Graphics and Vision**, v. 2, n. 1, p. 1-104, 2006. https://doi.org/10.1561/0600000009

ZHENG, J.; ZHANG, Z.; TAO, Q.; SHEN, K.; WANG, Y. An accurate multi-row panorama generation using multi-point joint stitching. Sequential Data Modeling and Its Emerging Applications, IEEE Access, v. 6, p. 27827-27839, 2018. https://doi.org/10.1109/ACCESS.2018.2829082

RU, D.; WANG, V.; HU, Q.; HU, W. The construction method of measurable aerial panorama based on panoramic image and multi-

view oblique images matching. *In*: 4th IEEE INTERNATIONAL WORKSHOP ON EARTH OBSERVATION AND REMOTE SENSING APPLICATIONS (EORSA) 4., 2016. **Anais** […]., Guangzhou, China. 2016.

LOWE, D. G. Distinctive image features from scale-invariant keypoints. **International Journal of Computer Vision**, v. 60, n.2, p. 91-110, 2004. https://doi.org/10.1023/B:VISI.0000029664.99615.94

SILVA, F. A.; PEREIRA, D. R.; SILVA, J. F. C.; ARTERO, A. O.; PITERI, M. A. TSRS - A new approach for traffic sign recognition using the SIFT algorithm. **Journal of Urban and Environmental Engineering**, v. 2, n. 2, p. 1-5, 2017.

KOENDERINK, J. J. The structure of images. **Biological Cybernetics**, v. 50, p. 363-396, 1984. https://doi.org/10.1007/BF00336961

LINDEBERG, T. "Scale-space theory: a basic tool for analyzing structures at different scales. **Journal of applied Statistics**, v. 21, n. 1-2, p. 225-270, 1994. https://doi.org/10.1080/757582976

LOWE, D. G. "Object recognition from local scale-invariant features", *In*: IEEE INTERNATIONAL CONFERENCE ON COMPUTER VISION, Kerkyra, Greece, 1999 p. 1150-1157. https://doi.org/10.1109/ICCV.1999.790410

MIKOLAJCZYK, K.; SCHMID, C. A performance evaluation of local descriptors. **IEEE Transaction on Pattern Analysis and Machine Intelligence**. v. 27, n. 10, p. 1615-1630, 2005. https://doi.org/10.1109/TPAMI.2005.188

BEIS, J.; LOWE, D. G. Shape indexing using approximate nearest-neighbour search in high dimensional spaces. *In*: IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, San Juan, Puerto Rico, USA,. 1997. p. 1000-1006

OKABE, T.; SATO, Y. Object recognition based on photometric alignment using RANSAC. *In*: IEEE COMPUTER SOCIETY CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, Madison, WI, USA,. 2003. p. 221-228.

FISCHLER, M. A.; BOLLES, R. C. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. **Communications of the ACM**, v. 24, n. 6, p. 381-395, 1981. https://doi.org/10.1145/358669.358692

BROWN, M.; LOWE, D. G. Automatic panoramic image stitching using invariant features. **International Journal of Computer Vision**, v. 74, n. 1, p 59-73, 2007. https://doi.org/10.1007/s11263-006-0002-3

GRACIAS, N.; MAHOOR, M. H.; NEGAHDARIPOUR, S.; GLEASON, A. Fast image blending using watersheds and graph cuts. **Image and Vision Computing**, v. 27, p. 597-607, 2009. https://doi.org/10.1016/j.imavis.2008.04.014

VINCENT, L.; SOILLE, P. Watersheds in digital spaces: an efficient algorithm based on immersion simulations. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 13, n. 6, p. 583-598, 1991. https://doi.org/10.1109/34.87344

BEUCHER, S. Image segmentation and mathematical morphology. the watershed transformation. 2010. Disponível em: http://www.cmm.mines-paristech.fr/~beucher/wtshed.html. Acessado em: 12 dez. 2020.

DANTZIG, G. B.; FULKERSON, D. R. On the max-flow min-cut theorem of networks. RAND corporation, v.. 13, 1964.

HIRAGA, A. K.; SILVA, F. A.; ARTERO, A. O.; PAIVA, M. S. V. Um novo algoritmo para a construção de imagens panorâmicas usando os algoritmos SIFT e RANSAC. *In*: SIMPÓSIO BRASILEIRO DE GEOMÁTICA, 3. Presidente Prudente-SP, 2012, v. 1. p. 151-156,.