UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
CENTRO TECNOLÓGICO
DEPARTAMENTO DE ENGENHARIA ELÉTRICA
PROJETO DE GRADUAÇÃO

RENAN BOTAN

IMPLEMENTAÇÃO DE ALGORITMO PARA CRIAÇÃO DE MOSAICO EM IMAGENS DE MICROSCOPIA DIGITAL DE BAIXO CUSTO

VITÓRIA – ES
OUTUBRO/2016

RENAN BOTAN

IMPLEMENTAÇÃO DE ALGORITMO PARA CRIAÇÃO DE MOSAICO EM IMAGENS DE MICROSCOPIA DIGITAL DE BAIXO CUSTO

Parte manuscrita do Projeto de Graduação do aluno **Renan Botan**, apresentada ao Departamento de Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para aprovação na disciplina —"ELE08553– Projeto de Graduação II".

Prof. Dr. Klaus Fabian Côco
Orientador

Eng. Matheus Vieira Lessa Ribeiro
Banca

Prof. Dr. Jorge Leonid Aching Samatelo
Banca

Renan Botan
Aluno

VITÓRIA – ES
OUTUBRO/2016

DEDICATÓRIA

Dedico este trabalho a meus mais que sempre com muito carinho, esforço e dedicação criaram as condições para que eu chegasse até aqui.

AGRADECIMENTOS

Agradeço, primeiramente a empresa Mogai Tecnologia, em especial ao Franco Machado, pela oportunidade de trabalhar com o MicroScanner e utilizá-lo como motivador da necessidade de desenvolvimento deste trabalho.

O meu muito obrigado a Lucas de Paula Veronese e Rômulo Ramos Radaelli pelas incontáveis horas me ajudando a corrigir códigos, me ensinando vários conceitos tanto de processamento de imagens quanto de programação.

Agradeço também a Lucio André Amorim, pelas ideias e experiências trocadas ao longo do processo devido a semelhança em nossos trabalhos.

Um agradecimento especial a Marilia Arantes Lima, minha namorada, por todo apoio moral, votos de motivação e companheirismo nesse período conturbado e estressante.

“Aqueles que se sentem satisfeitos sentam-se e nada fazem. Os insatisfeitos são os únicos benfeitores do mundo”

Walter S. Landor

RESUMO

Este trabalho apresenta um algoritmo de junção de imagens para a criação de mosaicos, com imagens provenientes de um microscópio digital de baixo custo, criado pela empresa Mogai Tecnologia da Informação, o qual é chamado de MicroScanner. Este programa resolve o problema entre campo de visão e resolução. Os códigos foram desenvolvidos em linguagem C++, e fez-se o uso das funções de processamento de imagens da biblioteca *open source* openCV. As fotos são capturadas sequencialmente, e essa ordem é usada pelo programa para reduzir custo de processamento e aumentar a confiabilidade da imagem final criada.

ABSTRACT

This work presents an algorithm that stitch images to create mosaics, which images comes form of a low cost digital microscope, created by Mogai Tecnologia da Informação, such device is called MicroScanner. This program solve the trade-off problem between field of view and resolution. These codes were developed in C++ language, and were used the open source penCV library as well. Pictures are sequentially taken, and this sequence is used by the program to reduce computing costs and increase reliability of the resulting image.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

OpenCV	*Open Source Computer Vision Library*
RANSAC	*Random sample consensus*
SIFT	*Scale-invariant feature transform*
SURF	*Speeded Up Robust Features*

SUMÁRIO

1 INTRODUÇÃO

Desde sua invenção no final do século XVI, o microscópio ótico vem tendo papel fundamental para a ciência, na descoberta de doenças, comprovação de métodos científicos e estudo celular, dentre muitas outras aplicações, contribuindo para a evolução da ciência em pesquisas de diferentes áreas do conhecimento como na biologia, medicina e ciências exatas (SCIENCELEARN, 2016).

Nas últimas décadas a evolução do microscópio ótico começou a decrescer devido ao fato de que os limites óticos foram atingidos. Foi descoberto que a resolução máxima de um microscópio ótico é a metade do comprimento de onda da luz que ilumina a amostra (a menos que sejam utilizadas técnicas de fluorescência). Por exemplo, se a luz que em questão tem $400\eta m$ (luz azul) a resolução máxima do microscópio será de $200\,\eta m$, mas esse limite dificilmente é obtido devido a qualidade das lentes (LEICA, 2016; SCIENCELEARN, 2016).

Em 1931 o físico alemão Ernst Ruska desenvolveu o microscópio eletrônico ou de elétrons que não usa luz para focar a imagem, mas sim feixe de elétrons. Com isso conseguiu-se uma maior resolução, superando os limites óticos. Hoje existem microscópios que atingem resoluções de até $0.1\eta m$ como ilustra a Figura 1.1. Entretanto esses microscópios ainda são relativamente caros, estando à ordem de R$ 100.000,00 (LEICA, 2016).

A partir da segunda metade do século XX os microscópios óticos passaram a ser também utilizados na montagem de microssistemas elétricos, eletrônicos e mecânicos e continuará sendo ferramenta indispensável para o crescimento e desenvolvimento dessas áreas (POTSAID; BELLOUARD; WEN, 2005).

Porém na maioria das vezes os microscópios óticos se deparam com o problema de se ter que optar por uma alta resolução, com lentes capazes de magnificar a escala em milhares de vezes, ou necessitar possuir um campo de visão maior com lentes que ampliam menos (POTSAID et al., 2005).

Nas últimas décadas, entretanto, com o advento da computação, processadores com melhor desempenho e memórias cada vez maiores, permitiram a criação de microscópios digitais e de algoritmos de processamento de imagens, tornando essa disputa entre resolução e campo de visão capaz de ser resolvidas com a junção de imagens formando grandes mosaicos (POTSAID et al., 2005).

Figura 1.1- Comparação do poder de resolução do olho humano com microscópios óticos e eletrônicos.

Fonte: Adaptado de SCIENCELEARN (2016)

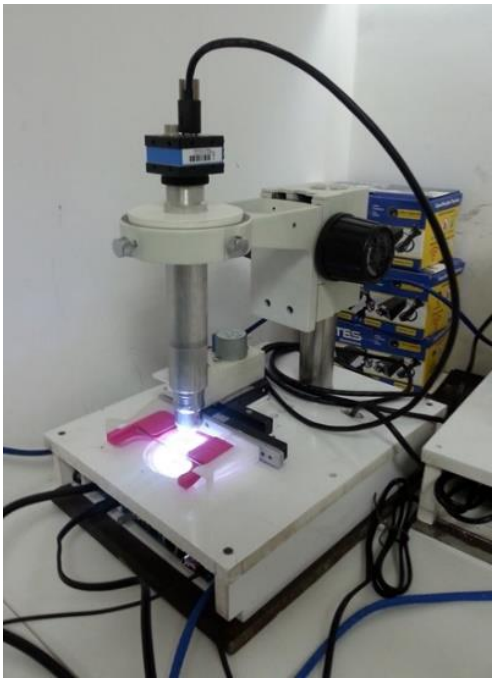
Além da microscopia algoritmos para construção de panoramas vem sendo usados em diversas outras áreas, como panorâmicas utilizadas em câmeras fotográficas convencionais e smartphones, mapeamento de território para operações de segurança e resgate como mostrado em AMORIN et al. (2015), produção de filmes como é o caso de BROWN, MATTHEW; LOWE (2007) .

1.1 Motivações

O que levou a escolha deste tema de trabalho para este projeto de graduação foi o fato de existir un interesse no estudo e desenvolvimento de algoritmos de construção de mosaicos para uso em sistemas microscópicos óticos/digitais de baixo custo. Para esta pesquisa, este projeto será referenciado como “MicroScanner”*, e o *hardware* utilizado foi o mostrado na Figura 1.1.1, o qual pertencente à empresa MOGAI Tecnologia da Informação.

*Comercialmente microscanner também pode se referir a um equipamento de verificação de cabos de rede.

Figura 1.1.1- MicroScanner desenvolvido pela MOGAI Tecnologia da Informação.



Fonte: Próprio autor.

A proposta do MicroScanner consiste em ter um microscópio ótico/digital conectado em um sistema de comunicação e de armazenamento de dados em nuvem (*cloud*). Em que amostras podem ser coletadas, escaneadas e fotografadas em uma estação de coleta, que muitas vezes pode estar em áreas remotas, como por exemplo no interior da Amazônia. Após a captura, é feito o *upload* automático dessas imagens para uma "nuvem", o que permite que as mesmas possam ser acessadas via navegador em qualquer lugar do mundo por um computador, *tablet* ou *smart phone*, desde que conectado à internet.

1.2 Importância deste trabalho

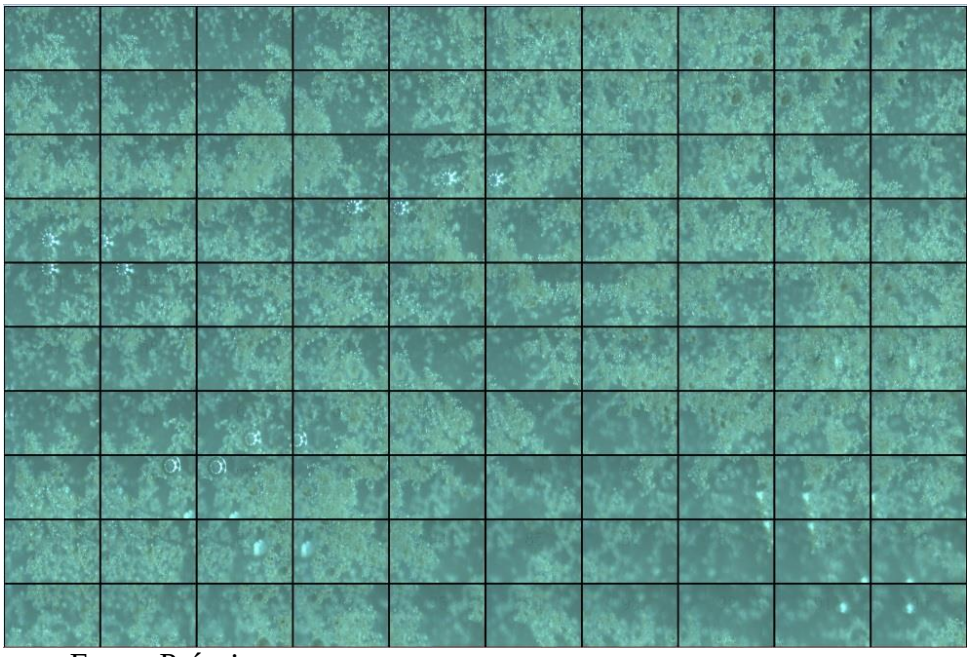
Este conceito de microscópio visa propor uma solução de baixo custo, quando comparada aos microscópios digitais disponíveis no mercado (LEICA, 2016; POTSAID et al., 2005), bem como, dentre outros benefícios, resolver o problema de alto custo com deslocamento de mão de obra especializada e de materiais para a análise das amostras coletadas.

Com o MicroScanner a coleta de amostras pode ser feita por um técnico, a um custo menor, e as análises dessas amostras poderão ser feitas de qualquer lugar do mundo, a custos menores.

1.3 Objetivos

Esta pesquisa propõe implementar um programa que realize a junção (*stitch*) das imagens capturadas pelo MicroScanner, como mostrado na Figura 1.3.1, para criar um mosaico sem perda de resolução e qualidade das imagens capturadas para uso em aplicações diversas a baixo custo operacional. Objetiva-se resolver o problema entre o grau de ampliação de lentes e campo de visão, pois área vista sem diminuir a magnificação nem a resolução.

Figura 1.3.1- Matriz com 10x10 imagens ordenadas capturadas pelo MicroScanner



Fonte: Próprio autor

Pretende-se, posteriormente, que esse código seja integrado ao sistema do servidor e assim, ser utilizado por processadores com menor custo no MicroScanner e centralizar o processamento da criação dos mosaicos em um único computador com maior capacidade, reduzindo-se assim, custos com *hardware*.

1.4 Metodologia para a solução do problema

Em posse das imagens capturadas e com o auxílio de ferramentas *open source*, foram utilizadas técnicas de detecção e correspondência de características (*features matching*), de ajuste de distorção (*warping*) para bordas e projeção das imagens no mesmo plano, compensação de ganhos para suavizar as diferenças de cor entre as imagens e de composição

(*seam correction* e*blending*) para fazer a construção da imagem final com alta resolução, pouca distorção das características presente nas imagens e transição suave de uma imagem para outra no mosaico final.

1.5 Correlação com Trabalhos Existentes

Há um extenso número de trabalhos na literatura como BROWN, M.; LOWE (2003), MILGRAM (1975), CHALFOUN et al. (2014) e SZELISKI (2004), além de aplicações comerciais tais como, CHEN, S. E. (1995)e BROWN, MATTHEW; LOWE (2007). Porém, pretende-se criar uma imagem final com uma maior precisão e transições mais suaves que o programa proposto por AMORIN et al. (2015), que não usa algoritmos para a suavizar marcas de junção, correção de tempos de exposição, e nem *blending*. Objetiva-se obter resultados os mais próximos possíveis de BROWN, MATTHEW; LOWE (2007), que usa algoritmos para suavizar as regiões de ligação, porém também se quer eliminar a necessidade de detectar qual imagem se ligará com qual, pois como o MicroScanner segue sempre o mesmo caminho para tirar as fotos, utilizou-se essa ordem para reduzir tempo de processamento do programa.

Um outro ponto que deve ser ressaltado, é o fato de já existir *softwares* comerciais que fazem isso no mercado com uma qualidade aceitável, como os citados acima, porém com valores de licença elevados. Apesar de se haver uma versão gratuita, para BROWN, MATTHEW; LOWE (2007), esta entretanto, está limitada a utilização de apenas 2GB de memória para o processamento das imagens, o que reduz a quantidade de imagens a serem processadas ou a qualidade do resultado final. E mesmo se comprado algum desses *softwares*, não seria possível integra-lo ao servidor do MicroScanner visto que não se teria acesso aos códigos fonte desses programas, apenas a seus executáveis.

1.6 Estrutura do Trabalho

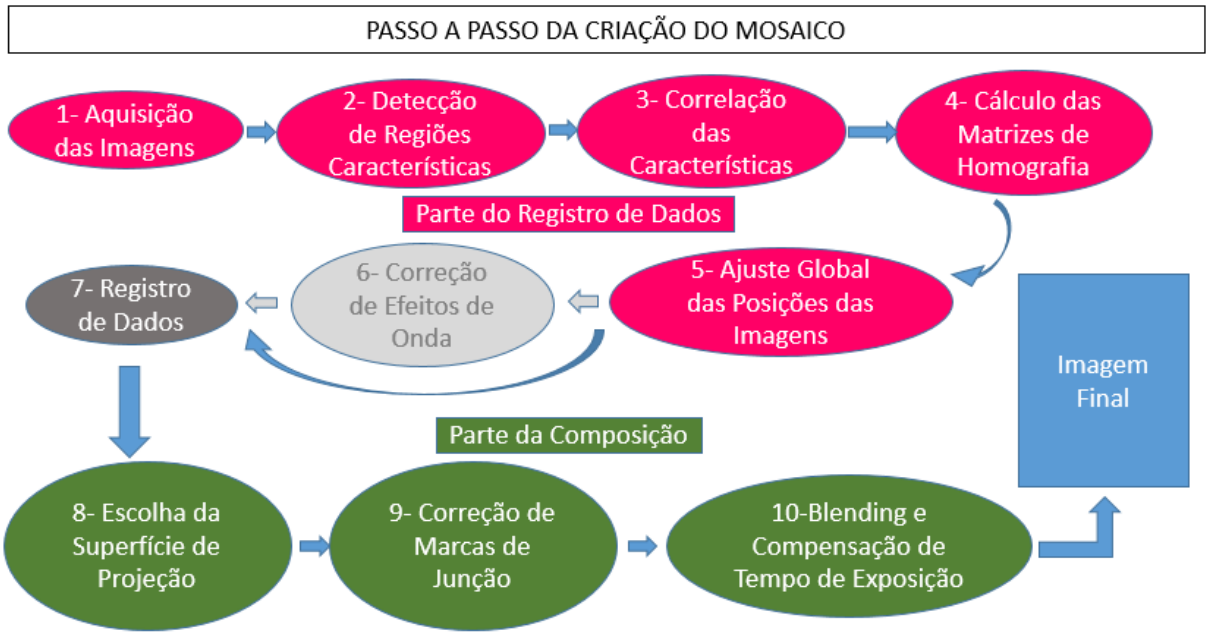
Este trabalho está organizado da seguinte forma:

No capítulo 2 será apresentada a sequência de passos que o programa proposto ira seguir para criar o mosaico, um pouco da teoria envolvida em cada passo e as vantagens e desvantagens de cada método utilizado. O capitulo 3 apresentará os resultados obtidos, a discutirá influência dos principais passos do programa.Por fim é apresentada uma conclusão geral e são feitas sugestões para trabalhos futuros.

2 ESTRUTURA DO CÓDIGO

O algoritmo em questão segue os passos como mostrado na Figura 2.1:

Figura 2.1- Passo a passo aplicado para se obter o mosaico final



Fonte: Próprio autor

Cada etapa indicada na Figura 2.1 será discutida em detalhes nas próximas seções. O passo 6 também pertence parte do registro de dados, entretanto este não foi utilizado na implementação deste código pois sua função não se fez necessária. Mas sua a título de complemento da teria de construção de mosaicos ele será comentado brevemente.

2.1 Aquisição das imagens

Inicialmente, os nomes das imagens são importados para o programa e ordenados de acordo com a ordem de ligação das imagens. Em seguida são carregadas as imagens. É importante ressaltar que nesta etapa cada imagem será uma matriz tridimensional RGB, portanto, esse processo consome bastante memoria, dependendo da quantidade de imagens usadas na composição final do mosaico.

2.2 Detecção de regiões características

Este trabalho baseia-se no método de detecção e correlação de regiões características (*features*) para alinhar as imagens no mosaico. Entretanto antes de começar a descrever como essa abordagem funciona é válido ressaltar que existe outro tipo de abordagem que pode ser utilizada para a solução deste problema.

Este segundo método é conhecido como Alinhamento direto de imagens, que pode funcionar bem no alinhamento sequencial de quadros (*frames*) na criação de mosaicos para vídeo, entretanto, para um número elevado de imagens, esta abordagem sofre o problema de desalinhamento das fotos na composição final, o que favorece ao aparecimento de espaços não preenchidos entre as imagens (SZELISKI; SHUM, 1997).

Por sua vez, o método baseado na detecção de *features* apresenta significativa robustez. O trabalho apresentado por BROWN, MATTHEW; SZELISKI; WINDER (2005) aponta que quando há *features* suficientemente distribuídas nas imagens, é possível encontrar alinhamento entre as imagens e conseguir um *stitch* bem sucedido. Porém esse método tem a desvantagem de se tornar impreciso em casos com poucas *features*, ou onde há muitas dessas mesmas, ou ainda, onde essas estão distribuídas de maneira desigual, levando-o à ineficácia (SZELISKI, 2010).

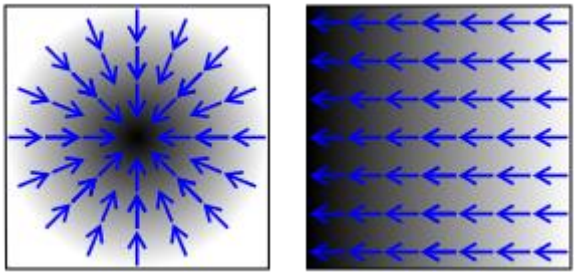
Ainda segundo SZELISKI (2010) o método de reconhecimento de *features* pode der dividido em três partes, que serão descritos nas três próximas subseções:

- Detecção de *patches* e *keypoints*;
- Extração de descritores;
- Pré-correlação dos descritores;

2.2.1 Detecção de *patches* e *keypoints*

Patch é conhecido como a região em torno de um *keypoint*, e de maneira básica este último é encontrado em regiões com grandes contrastes e/ou com uma mudança direcional de intensidade ou cor, ou seja, regiões que apresentam gradientes de valores mais elevados. A Figura 2.2.1.1 ilustra regiões para onde os gradientes crescem.

Figura 2.2.1.1- Variação do gradiente em 2 regiões diferentes.



Fonte: SHAPIRO; STOCKMAN (2001)

Por definição, o valor do gradiente é um vetor de dois componentes, calculado para cada pixel da imagem. Tal vetor aponta para a maior mudança de intensidade ou cor. Esse é um dos jeitos mais fáceis de se encontrar *keypoints*, porém, há casos mais complicados, que fogem ao escopo deste trabalho.

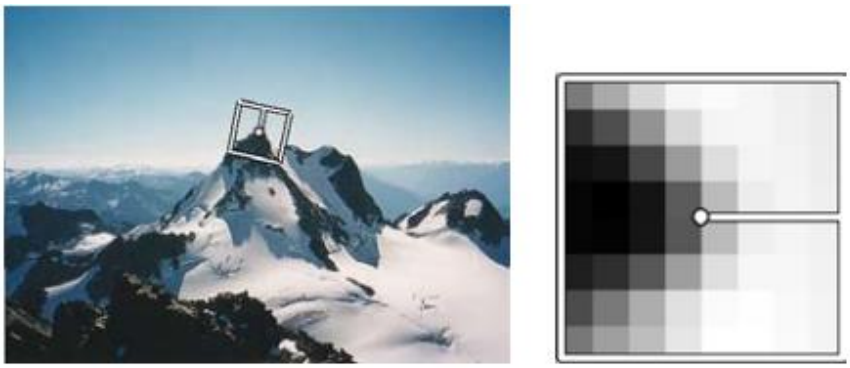
2.2.2 Extração de descritores

Nesta etapa os *patches* já extraídos são convertidos e agrupados em formas mais compactas, estáveis e invariantes, ou seja, são transformados em descritores.

Este processo consiste basicamente em orientar os *pathces*, rotacioná-los e escaloná-los, podendo-se também aplicar outras transformações afins (ou lineares de primeiro grau). A Figura 2.2.2.1 ilustra melhor de como é esse processo.

É importante destacar que esses descritores devem ser distintos, robustos a ruídos, a detecção de erros e deformações, tanto geométricas quanto fotométricas (distorções de lentes, movimento de pessoas/objetos).

Figura 2.2.2.1- Extração e orientação de um descritor



Fonte: BROWN, MATTHEW et al. (2005)

2.2.3 Pré-correlação dos descritores

Neste ponto é feita uma pré-seleção dos descritores eliminando os candidatos ruins para correlação com as outras imagens. Isso pode ser feito utilizando-se um *threshold* para o valor da distância euclidiana entre os descritores de imagens diferentes, que como são invariantes a rotação e mudança de escala, a alteração do tamanho das imagens não influencia o valor da distância encontrada.

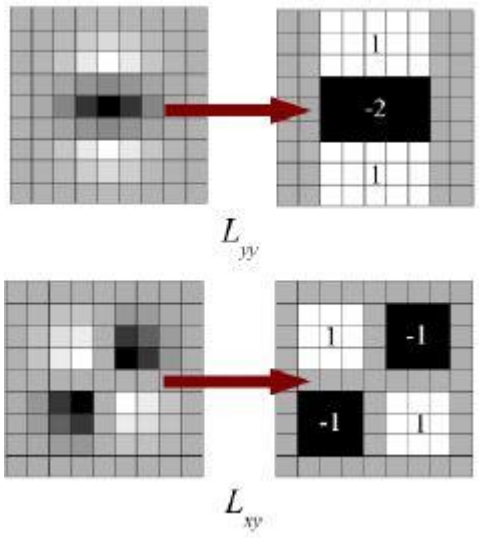
2.2.4 Comparação entre SIFT, SURF e o que foi implementado

Os três passos brevemente descritos nas seções anteriores, explicam como funciona o processo genérico de detecção de *features*. Agora será abordado um pouco mais detalhadamente duas abordagens comumente usados em diferentes aplicações da visão computacional para se encontrar *features*, o SURF (BAY; TUYTELAARS; VAN GOOL, 2006), que do inglês significa *Speeded-Up Robust Features*, onde, neste trabalho foram utilizadas, as funções de detecção de *keypoints* e extração de descritores, e quando possível serão feitas comparações com o SIFT (LOWE, 2004), *Scale-Invariant Feature Transform* e seus variantes, pois apresentam alta robustez e relativa velocidade quando comparado ao SURF.

Assim como o SIFT, o SURF é invariante a mudança de escala pois baseia-se no processo de detecção de *keypoints* em pirâmide (com diferentes resoluções). O SIFT utiliza uma aproximação do Laplaciano do Gaussiano para encontrar os *keypoints*; o Laplaciano para encontrar regiões de mudanças bruscas de gradiente (detector de bordas), e o Gaussiano como filtro, já que o Laplaciano é sensível a ruídos. Essa aproximação é chamada de Diferença de Gaussianos (BAY et al., 2006).

Por sua vez o SURF é mais ousado e baseia-se no determinante da matriz de Hessian para computar tanto escala quanto a localização dos *keypoints*, ele é chamado de *fast* Hessian detector. Esse método proposto é computacionalmente mais simples, menos sensível a ruídos, e com robustez equivalente à Diferença de Gaussianos. Ele é implementado com o auxílio de filtros caixa (*box filters*) convoluidos com a imagem original, como mostrado na Figura 2.2.4.1 (BAY et al., 2006).

Figura 2.2.4.1- Processo de convolução de filtros caixa com a imagem original.



Fonte: Adaptado de BAY et al. (2006)

No SIFT, o processo de descrição das *features* é feito considerando uma vizinhança de 16x16 no entorno do *keypoint*. O passo seguinte é dividi-la em 16 sub-regiões com 4x4 pixels e para cada sub-região um vetor de 128 dimensões para cada *keypoint* é criado esse vetor pode ser interpretado como um histograma. Enquanto o SURF utiliza uma vizinhança de 20x20 no entorno do *keypoint*, e os divide em 25 sub-regiões de 4x4 pixels, e para cada sub-região é aplicada uma transformação 2D Haar Wavelet, que pode retomar um vetor descritor de dimensão 64 ou 128. Este último é chamado de SURF-128, que apresenta maior robustez, porém é um pouco mais custoso que o SURF tradicional (SURF-64). Mesmo assim o SURF-128 é mais rápido que o SIFT. A Tabela 2.2.4.1 mostra a comparação de tempo gasto para executar os diferentes algoritmos para um mesmo grupo de imagens (BAY et al., 2006).

Para imagens de entrada, onde se tem certeza que elas não estão rotacionadas, pode ser usada a versão U-SURF, que não computa vetor de orientação (*Up-right vector*, por isso U-SURF) para os descritores, tornando o código ainda mais rápido. Entretanto, o SURF não apresenta grande robustez a mudanças de iluminação e de ponto de captura, quando comparado ao SIFT (BAY et al., 2006).

Tabela 2.2.4.1- Teste feito com SIFT e diferentes versões do SURF para um mesmo grupo de imagens

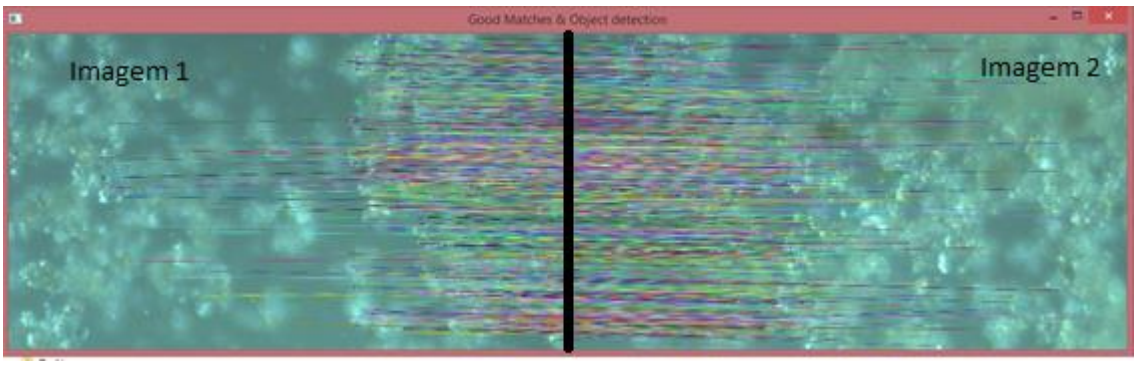
	U-SURF	SURF	SURF-128	SIFT
Tempo(ms):	255	354	391	1036

Fonte: Adaptado de BAY et al. (2006)

A pré-correlação dos descritores em ambos os casos é feita calculando-se a distância euclidiana dos pontos de interesse de imagens diferentes. Entretanto o SURF também usa o Laplaciano (traço da matriz de Hessian) para diferenciar regiões claras de regiões escuras, aumentando a qualidade das correspondências. Isso não aumenta o custo computacional, visto que o Laplaciano já foi calculado na detecção dos *keypoints*.

Neste trabalho, a pré-correlação é calculada em duas etapas: a primeira, é obrigatória, onde as posições dos descritores previamente extraídos de uma imagem são comparados com os de suas vizinhas horizontais. É aplicado um valor de corte (*treshold*) para verificar se os descritores se encontram na região acima ou abaixo desse valor, que empiricamente foi estabelecido 50, mas que pode sofrer alterações a depender do grupo de imagens, assim, se o descritor da imagem vizinha estiver dentro da região de corte ele será considerado um bom candidato para correspondência. A Figura 2.2.4.2 ilustra esse processo.

Figura 2.2.4.2- Busca de possíveis boas correspondências com a imagem vizinha horizontal



Fonte: Próprio autor

A segunda etapa por sua vez é opcional, ela compara os descritores de uma imagem com sua vizinha inferior. Desta vez foi utilizada a distância euclidiana como variável de corte. Um valor razoável para corte é 200, ou seja, todas os descritores menos distantes que esse valor é considerado uma boa correspondência. Essa etapa serve como um refinamento do processo pois

aumenta o número de pares de imagens vizinhas auxiliando diretamente no processo de alinhamento global das imagens, processo que será explicado mais à frente.

2.3 Correlação das características

Na etapa final de correspondência dos descritores foi utilizado o algoritmo RANSAC (FISCHLER; BOLLES, 1981) que do inglês significa *RANdon SAmple Consensus*, devido a sua robustez e baixo custo computacional. Esse algoritmo funciona de maneira oposta aos tradicionais algoritmos de refinamento de descritores, onde quando mais descritores maior a chance de uma correspondência ser boa.

Ao invés disso, o RANSAC utiliza uma pequena amostra k de dados considerados inicialmente como "mais confiáveis", se faz do uso de S tentativas para obter um probabilidade p de sucesso em cada tentativa para obter uma probabilidade P total de sucesso.

Para representar isso matematicamente considerou-se primeiro a probabilidade de S tentativas falharem. Como mostra a equação 2.3.1

$$1 - P = (1 - p^k)^S \tag{2.3.1}$$

Então o número mínimo de tentativas é mostrado na equação 2.3.2:

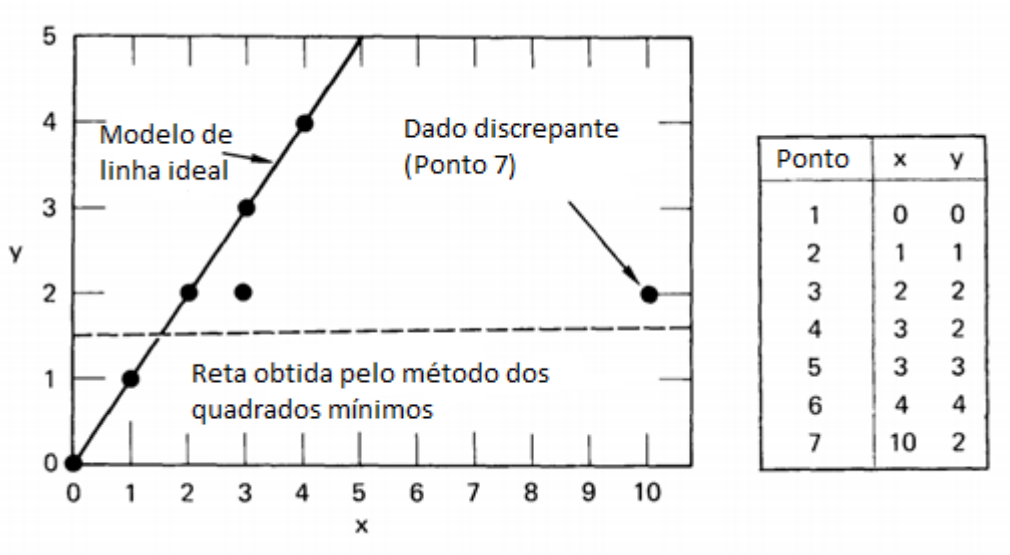
$$S = \frac{\log(1 - P)}{log(1 - p^k)} \tag{2.3.2}$$

O trabalho de STEWART (1999) dá exemplos de heurísticas e do número necessário de tentativas S para se obter 99% de probabilidade P de sucesso.

Aplicada a heurística S vezes, são então selecionados os descritores que pertencem ao conjunto de interesse. Esses descritores são chamados de *inliers*. Os descritores que representam correspondências ruins são chamados de *outliers*.

Um exemplo de heurística simples explica melhor a ideia de como o RANSAC funciona. Suponha que se queira encontrar uma reta dado um conjunto de pontos como mostra a Figura 2.3.1, onde tem-se sete pontos distintos, um deles é um dado discrepante da amostra. Pode-se então utilizar a heurística de se pegar S amostras dois pontos "de confiança"(pontos que se tem certeza que pertencem à amostra) e se traçar uma reta entre eles e se aplicar um valor de corte pré-definido para se determinar os *inliers* e os *outliers*.

Figura 2.3.1- Exemplo de heurística do RANSAC para encontrar quais pontos pertencem a uma reta.



Ponto	x	y
1	0	0
2	1	1
3	2	2
4	3	2
5	3	3
6	4	4
7	10	2

Fonte: Adaptado de (FISCHLER; BOLLES, 1981)

2.4 Cálculo das matrizes de homografia

O nome matriz de homografia serve para denominar a matriz de transformação de pontos em 3D, nela pode haver rotações, e transformações projetivas(SZELISKI, 2010). Quando representada em coordenadas não homogêneas se apresenta como mostra a Equação 2.4.1.

$$\begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} = \begin{bmatrix} H_{11} & H_{12} & H_{13} \\ H_{21} & H_{22} & H_{23} \\ H_{31} & H_{32} & H_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} \Leftrightarrow x_2 = Hx_1 \qquad 2.4.1$$

O que se quer é determinar a matriz de transformação para podermos projetar cada imagem diferente num mesmo plano de composição com sobreposição compatível com as regiões capturadas nas fotos. Para isso precisamos projetar a matriz de homografia no plano (x,y), ou seja, em 2D. Resolvendo a Equação 2.4.1 em coordenadas homogêneas ($x_2' = \frac{x_2}{z_2} e\ y_2' = \frac{y_2}{z_2}$) e obtemos a Equação 2.4.2 e 2.4.3.

$$x_2' = \frac{H_{11}x_1 + H_{12}y_1 + H_{13}z_1}{H_{31}x_1 + H_{32}y_1 + H_{33}z_1} \quad 2.4.2$$

$$y_2' = \frac{H_{21}x_1 + H_{22}y_1 + H_{23}z_1}{H_{31}x_1 + H_{32}y_1 + H_{33}z_1} \quad 2.4.3$$

Como pode-se perceber tem-se 9 incógnitas, portanto é preciso de pelo menos 3 pontos para calcular a matriz de homografia. Esses pontos são as posições dos descritores encontrados pelo SURF, e selecionados pelo RANSAC previamente discutidos. A OpenCV utiliza métodos iterativos para descobrir os valores dos Hs que retornam uma matriz em 2D expressa em coordenadas homogêneas e com as translações também computadas. A Equação 2.4.4 expressa o formato da matriz de homografia retornada pelo método *cvfindHomograpy*.

$$H = \begin{bmatrix} rot + proj & rot + proj & Tx \\ rot + proj & rot + proj & Ty \\ 0 & 0 & 1 \end{bmatrix} \quad 2.4.4$$

Onde Tx e Ty são translações em x e y respectivamente, e rot+proj são valores equivalentes a rotação mais projeções computadas juntas.

Analisando um output do código como mostrado na matriz 2.4.5, pode-se comprovar isso:

$$\begin{bmatrix} 1.007259901429307 & -0.003132591628213404 & -1002.115671807583 \\ 0.001401256245885265 & 1.00156905077643 & 23.08991730022312 \\ 4.045351565727439e-006 & -3.427694546171485e-006 & 1 \end{bmatrix} \quad 2.4.5$$

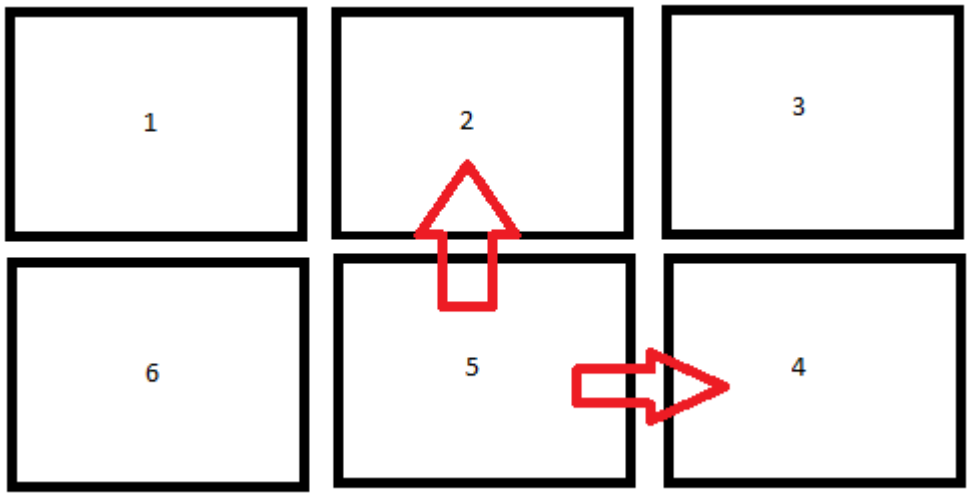
Podemos perceber que os 2 primeiros valores da última linha da matriz são muito pequenos podendo ser considerados nulos. Esse erro se dá devidos as aproximações numéricas utilizadas.

É importante também apontar que para cada imagem vizinha pode-se ter uma matriz de homografia correspondente, isso significa dizer que as imagens são correlacionadas em pares. A Figura 2.4.1 ilustra esse processo. Dela pode-se perceber que a imagem 5 por exemplo, pode ter uma matriz de homografia para determinar sua posição em relação a imagem 4, outra em relação a 3, outra em relação a 2 e ainda mais uma em relação a 1 (a imagem 6 não está na

análise neste caso, pois neste trabalho, as homografias foram estimadas com relação às posições das imagens anteriores, e não faz sentido achar uma matriz de homografia para a imagem 5 em relação 6, mas o oposto é válido).

Isso significa que a mesma imagem, para o caso deste trabalho, poderá ter sua posição definida por até 4 matrizes de homografia diferentes. Entretanto, como os resultados obtidos foram satisfatórios, nesta pesquisa foram apenas utilizadas 2 matrizes de homografia, uma com sua vizinha horizontal e outra com sua vizinha vertical superior. Isso reduz o tempo total de alinhamento global da composição.

Figura 2.4.1- Esboço de como as homografias são estimadas nesse trabalho. Para a imagem 5 é estimada uma homografia em relação a imagem 2 e outra em relação a imagem 4.



Fonte: Próprio autor

Diante desse fato precisa-se de um processo que de posicionamento global dessas imagens (*bundle adjustment*) que será discutido na próxima seção.

2.5 Ajuste global das posições das imagens

Diante do fato de que agora tem-se mais de que uma matriz de homografia para determinar a posição de uma mesma imagem, precisa-se, portanto de um processo que possa alinha-las globalmente. Isso evita acúmulo de erros quando o número de imagens a serem inseridas no mosaico cresce. Esse processo de se ajustar a posição de projeção de imagens de forma global é denominado *bundle adjustment* (TRIGGS et al., 2000).

A função de *bundle adjustment* disponível na biblioteca Opencv faz com que o tempo de processamento para esta etapa cresce quase exponencialmente em função do aumento do número de matrizes de homografia. O algoritmo de *bundle adjustment* é executado de maneira iterativa e a openCV usa um método especifico e privado que estima um erro supostamente aceitável e não oferece parâmetros de entrada para altera-los manualmente. Uma opção para melhorar isso e sugestão de trabalho futuro poderia ser a utilização da biblioteca Ceres(CERES, 2016) para implementar essa etapa do processo, pois esta possui métodos que possuem a estimativa do erro como parâmetros de entrada. Colocando-se valores de erros maiores, entretanto, dentro de intervalões que ainda permitam resultados aceitáveis, se poderia reduzir o tempo de convergência, tornando o algoritmo mais veloz.

A função de *bundle adjustment* trabalha com matrizes de parâmetros intrínsecos da câmera. Essa matriz tem o formato mostrado na Equação 2.5.1. Onde f_xe f_y são as distancias focais das imagens nos eixos x e y respectivamente, enquanto c_x e c_y são os centros focais da imagem.

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 0 \end{bmatrix} \qquad 2.5.1$$

Neste ponto, é importante observar duas coisas, a primeira é que as posições das imagens foram definidas por matrizes de homografia. A openCV possui o método *estimator* para estimar as matrizes de parâmetros intrínsecos das câmeras dada as matrizes de homografias. A segunda é que função de *bundle adjustment* da openCV considera que todas imagens foram tiradas por uma mesma câmera virtual e estima as distancias focais e centros focais baseadas na posição única câmera, ou seja, só haverá rotação da câmera e não translação da mesma.

Apenas de formar ilustrativa, a Figura 2.5.1 mostra a situação em que o alinhamento global foi feito considerando-se a posição de várias câmeras (ou translação de uma única virtual, além da rotação), o que é uma outra possibilidade de implementação de função, que poderia reduzir o tempo de convergência do algoritmo.

Figura 2.5.1- Alinhamento global feito considerando-se várias câmeras.

Fonte: CORNELL (2016)

Nesta etapa a estimativa da posição global é feita reprojetando-se as matrizes de parâmetros intrínsecos da câmera para traz da câmera, ou seja, tornando suas distancias focais negativas, e afastando-as da câmera para reduzir os valores dos ângulos entre as imagens e consequentemente os erros de projeção. Assim teremos uma matriz K com f_xe f_y negativos e multiplicado por uma constante a e b respectivamente, como mostra a Equação 2.5.2.

$$K = \begin{bmatrix} -af_x & 0 & c'_x \\ 0 & -bf_y & c'_y \\ 0 & 0 & 1 \end{bmatrix} \quad 2.5.2$$

Segue um exemplo de saída do programa:

Antes da reprojeção, matriz K estimada a partir da matriz e homografia, como mostrado na matriz 2.5.3.

$$K = \begin{bmatrix} 10951.9406928294 & 0 & 960 \\ 0 & 10951.9406928294 & 540 \\ 0 & 0 & 1 \end{bmatrix} \quad 2.5.3$$

Depois da reprojeção obtemos a matriz 2.5.4:

$$K = \begin{bmatrix} -3985.78774653595294 & 0 & 5590.273722412106 \\ 0 & -6063.467220792676 & 1770.336352627874 \\ 0 & 0 & 1 \end{bmatrix} \quad 2.5.4$$

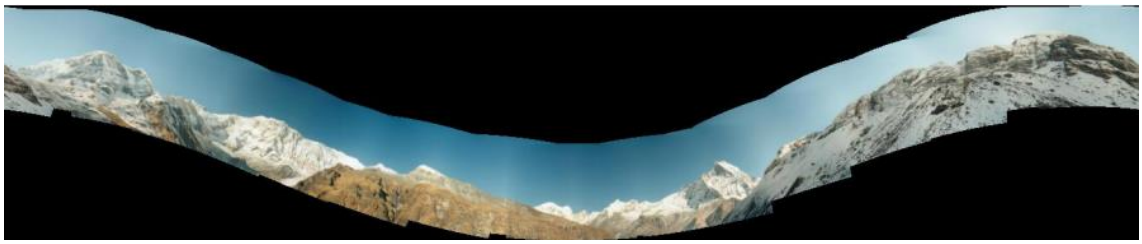
A técnica de *bundle adjustment* utilizada como processo posterior a detecção de *features* sofre dois problemas fundamentais, o primeiro é que a presença de *outliers* pode dificultar ou

impossibilitar a convergência do algoritmo. O segundo é que a mesma *feature* pode estar influenciado diretamente o cálculo de mais de uma matriz de homografia, tornando essa *feature* superestimada $\binom{m}{n}$ vezes, onde m é número de vezes que a *feature* é contada e n é o número de matrizes de homografia em que ela foi usada para ser estimada. Portanto um *outlier* que aparece $\binom{m}{n}$ vezes pode atrasar consideravelmente a convergência do algoritmo de *bundle adjustment* (SZELISKI, 2010).

2.6 Correção de efeitos de onda

Na construção de mosaicos onde as imagens são capturadas manualmente, isto é, sem ajuda de um suporte ou superfície fixa, o efeito de ondulação pode ocorrer na formação do panorama final, como ilustrado na Figura 2.6.1.

Figura 2.6.1- Ilustração de como pode ficar um panorama quando há rotação da câmera no eixo x ou y.



Fonte:BROWN, MATTHEW; LOWE (2007)

Com o objetivo de resolver esse problema BROWN, MATTHEW; LOWE (2007) propuseram um algoritmo para alinhamento de panoramas. A Figura 2.6.2 mostra como fica o alinhamento do panorama após a correção do efeito de onda.

Entretanto o neste trabalho não se fez o uso desta ferramenta pois a superfície de captura é fixa e o deslocamento entre uma foto e outra é constante.

Figura 2.6.2- Panorama após correção do efeito de ondulação da câmera.

Fonte:BROWN, MATTHEW; LOWE (2007)

2.7 Registro de dados

Até o momento foram apenas extraídas e computadas informações das imagens que formarão o panorama final. Informações sobre correspondência de *features* das vizinhanças, sobre posições locais das imagens e alinhamento global das mesmas. A openCV usa estrutura de dados próprias para guarda-los e envia-los para os passos seguintes da etapa de composição.

A primeira etapa da composição que precisa ser feita é escolher a superfície em que o panorama final vai ser projetado e aplicar as transformações de coordenadas e remapeamentos (*warping*) necessários para produzir a projeção final.

Em seguida, deve-se tratar as regiões de junção das imagens para produzir um mosaico com transições suaves, ou seja, sem marcas de junção e sem borrões devido ao deslocamento de objetos nas imagens (*seam correction*), selecionando-se os pixels que pertencem ao panorama e excluindo os que não pertencem.

Por fim utiliza-se um processo conhecido na literatura como *blending*, para compensar problemas de diferença de tempos de exposição na captura das imagens e outros problemas de alinhamento.

Nas próximas seções serão discutidos esses três tópicos da etapa de composição acima citados mais a fundo.

2.8 Escolha da superfície de projeção

Na escolha da superfície de projeção deve-se optar por um balanço entre manter a aparência local de cada imagem não distorcida e ao mesmo tempo obter uma composição uniforme.

Neste trabalho, como sua aplicação é destinada à microscopia, optou-se por utilizar uma superfície de projeção plana pois é a que melhor retrata a realidade do ambiente fotografado.

Entretanto, quando o número de imagens começa a ficar grande, torna-se difícil manter a aparência do mosaico plana, sem esticar muito as imagens mais externas, pois o erro acumulado das distorções das imagens é propagado para as bordas da foto. A Figura 3.8 mostra um caso em que foram coladas um conjunto de 40 imagens (4x10) e observa-se claramente a distorção das imagens mais da esquerda. É valido ressaltar que quando não há *features* correspondentes suficientes, as imagens também são esticadas com o intuito de cobrir espaços vazios entre as imagens coladas (SZELISKI, 2010).

Essa expansão do tamanho das imagens pode levar ao preenchimento total da memória do computador e levar ao encerramento do programa. Para essas situações talvez fosse interessante utilizar superfícies de projeção cilíndricas (CHEN, S., 1995; SZELISKI, 1996) ou esféricas (SZELISKI; SHUM, 1997).

Para a projeção em uma superfície plana precisa-se apenas fazer uma projeção perspectiva das imagens no plano. Para as projeções em superfícies cilíndricas e esféricas por sua vez precisa-se redefinir suas distancias focais calculadas na etapa de *bundle adjustment* e apenas após isso projetar as imagens na superfície desejada.

Além dessas três superfícies de projeção há inúmeras outras que podem sem utilizadas a depender da aplicação. Um bom exemplo é o mapeamento das 6 faces de um cubo para fazê-lo parecer uma esfera (GREENE, 1986; SZELISKI; SHUM, 1997), método o qual tem grande aplicação da definição de ambientes em computação gráfica.

Por fim, é importante ressaltar duas observações, a primeira é que o centro de projeção da composição final já foi definido na etapa de *bundle adjustment*, pois como dito anteriormente, as distancias focais de todas as imagens são estimadas em relação a uma única câmera virtual, e a segunda é que caso a qualidade das imagens sejam reduzidas para a composição final, para se economizar memória e tempo de processamento, deve-se nesta etapa utilizar um pré-filtro para remover distorções (*aliasing*). Neste trabalho, entretanto o uso de tal filtro não se faz necessário visto que a qualidade das imagens não é reduzida, pois objetiva-se uma imagem final com resolução maior que as imagens de entrada.

2.9 Correção de marcas de junção

Para suavizar a diferença entre as regiões de junção a openCV estima máscaras onde as imagens são sobrepostas para aplicar o algoritmo apenas nessa região. Esse algoritmo baseado em UYTTENDAELE, M.; EDEN; SKELISKI (2001), calcula as diferenças de pixels entre as regiões e decide o que manter e o que apagar nas imagens para o caso de coisas se movimentando. Logo após, aplica-se uma média ponderada na região para dar maior importância aos pixels de maior intensidade, que geralmente é onde as imagens estão se ligando.

Na Figura 2.9.1 pode-se perceber que da imagem a para b, que pessoas foram removidas da parte direita e centro da imagem, resolvendo-se assim problemas fantasmas e borrões causados por movimento. Percebe-se ainda que mesmo com essa suavização as imagens ainda possuem erros de tempo de exposição que precisam ser tratados, como pode-se ver a diferença na cor do céu em cada imagem.

Figura 2.9.1- Ilustração de como objetos são removidos da composição e as marcas de ligação são suavizadas.

(a) Sem suavização

(b) Com suavização

Fonte: (UYTTENDAELE, M. T.; SZELISKI, 2004)

Essa etapa ajuda a corrigir problemas de falta de correspondência no mosaico final e também lida com questões de bactérias se movendo na lamínula. Todavia, a parte negativa deste método é que partículas de interesse presentes na amostra podem ser removidas por causa desse algoritmo.

2.10 *Blending e* compensação de tempo de exposição

Depois de corrigidos os problemas devidos a movimento e marcas de junção, ainda se precisa corrigir os efeitos de diferença do tempo de exposição e pequenos problemas de alinhamento para o caso em que há poucas *features* correspondentes.

Nesta etapa optou-se por utilizar um processo conhecido como *mult-blending* proposto em BURT; ADELSON (1983) pois utiliza-se uma pirâmide Laplaciana onde em cada nível é selecionada uma banda de frequência para que sejam suavizadas as diferenças entre as imagens.

Esse processo se assemelhar muito ao passo anterior, entretanto ele é aplicado em todas as imagens ao invés de apenas em regiões (máscaras), e o tratamento em várias bandas de frequência permite suavizar diferenças tanto em altas como em baixas frequências. BROWN, MATTHEW; LOWE (2007) propõem uma solução com apenas 2 bandas de frequência. Todavia neste trabalho a função utilizada da openCV seleciona automaticamente quantas bandas de frequência selecionar. A Figura 2.10.1 mostra um exemplo de *mult-blendig* usado na composição de montagens.

Figura 2.10.1- exemplo de como o *mult-blendig* pode ser usado em montagens. (a) Imagens de entrada, (b) Imagens montadas sem *mult-blendig* (c) Montagem com *mult-blendig*.

(a) (b) (c)

Fonte:(PEREZ; GANGNET; BLAKE, 2003)

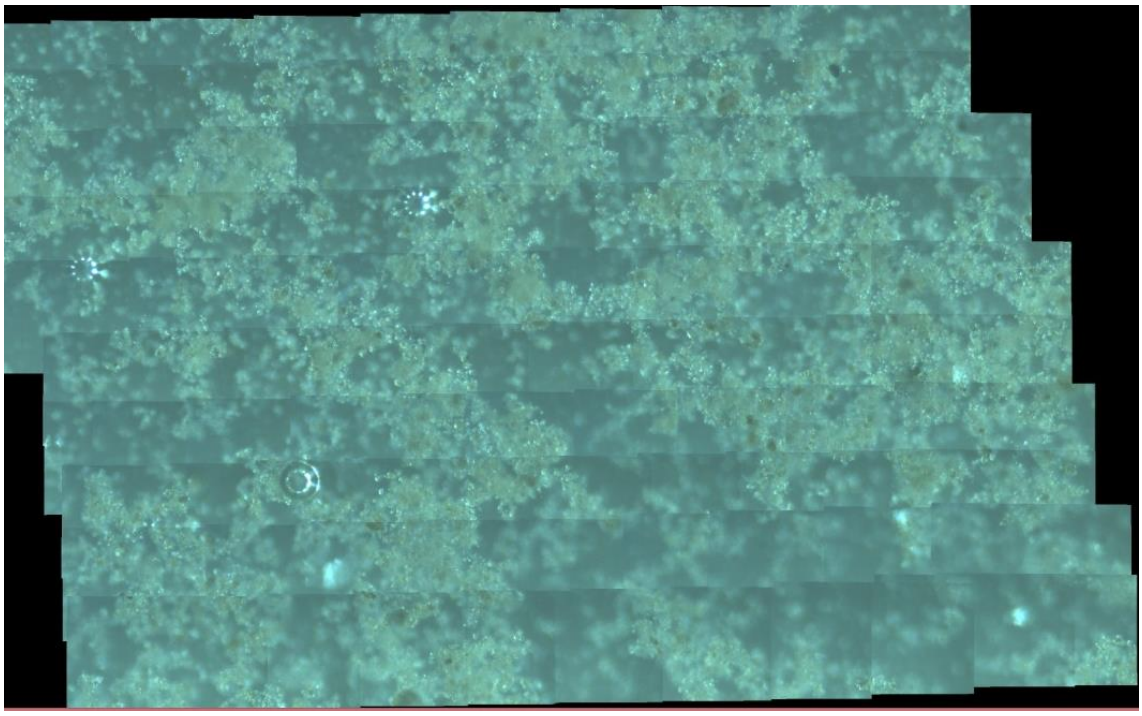
É importante indicar que caso a diferença de tempo de exposição seja muito grande pode-se ainda fazer uso de algoritmos como os propostos em (UYTTENDAELE, M. et al., 2001), que uniformiza os ganhos das imagens, tornado suas diferenças de iluminação menos discrepantes. Mas não foi necessário o uso de tais algoritmos pois a iluminação e tempo de captura das imagens do MicroScanner são constantes.

3 RESULTADOS

Neste capítulo serão apresentados e discutidos os resultados obtidos, também serão apresentados os problemas e limitações ainda existentes para a composição do mosaico final. Inicialmente é apresentado na Figura 3.1 uma projeção das 100 imagens anteriormente mostradas na Figura 1.3.1. Nessa projeção não foi feito ajuste global das posições das imagens, ou qualquer processo de composição descrito nas três últimas seções do capitulo 2.

O processo utilizado para obtê-la foi parecido com o utilizado em AMORIN et al. (2015). Dessa projeção é possível observar que (i) a correspondência das *features* não é boa devido a falta de alinhamento global e da não remoção de marcas de junção (ii) os motores do MicoroScanner precisam ser melhor calibrados.

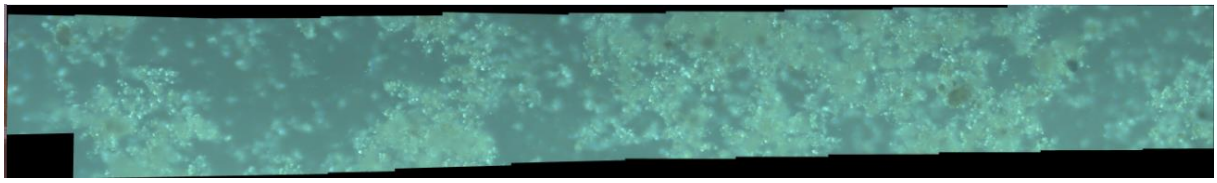
Figura 3.1- Projeção de 100 imagens mostradas na Figura 1.3.1, feita apenas baseada nas posições das matrizes de homografia de suas vizinhas horizontais.



Fonte: Próprio autor

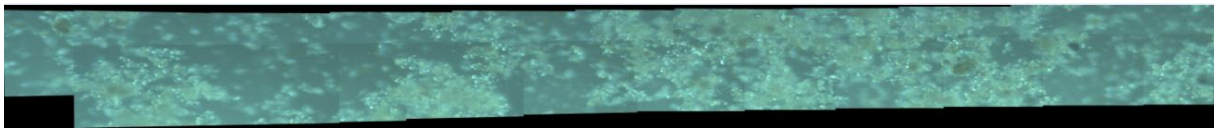
Finalmente a Figura 3.2 mostra uma composição com 20 imagens com alinhamento global, correção de marcas de junção e *blending*. Os resultados foram obtidos utilizando-se um computador com processado intel i5 2GHz 3MB de *cache* e 8GB de memória RAM. A Figura 3.3 mostra um mosaico feito com o mesmo conjunto de imagens utilizado para formar a Figura 3.2 porém desta vez sem correção de marcas de junção e sem *blending*.

Figura 3.2- Mosaico de 2x10 imagens utilizando-se alinhamento global, correção de marcas de junção e *blending*. Tempo total: 32.48 minutos, tempo alinhamento global: 25.77 minutos.



Fonte: Próprio autor

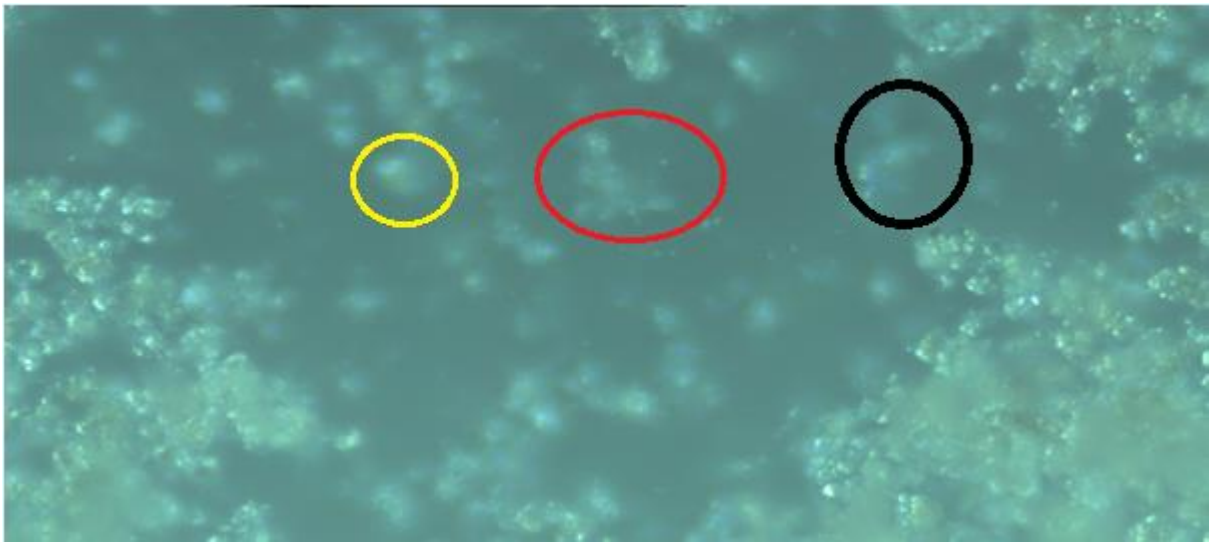
Figura 3.3- Mosaico de 2x10 imagens utilizando-se alinhamento global, sem correção de marcas de junção e sem *blending*. Tempo total: 31.34 minutos.



Fonte: Próprio autor

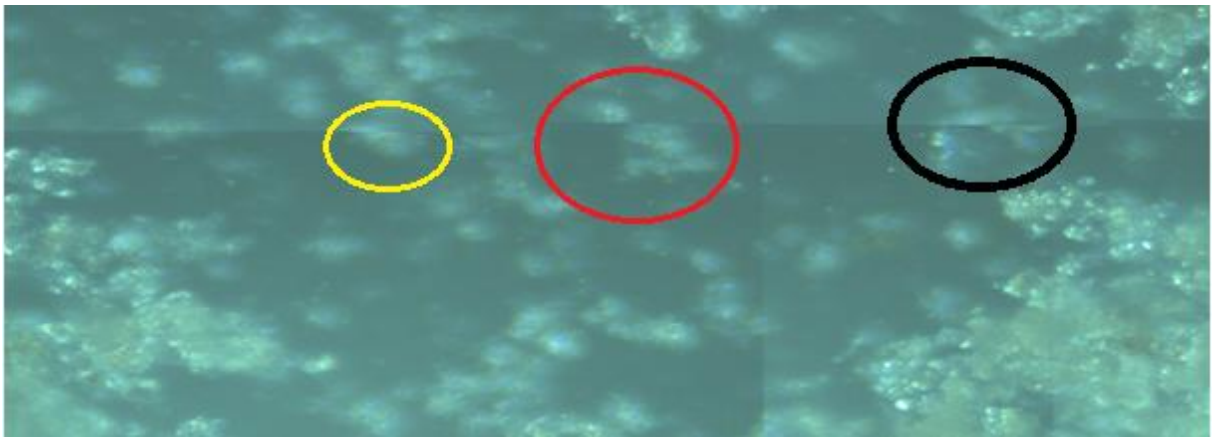
Ampliando a área esquerda das Figuras 3.2 e 3.3 como mostrado nas Figuras 3.4 e 3.5 respectivamente e analisando o círculo em preto podemos ver como o algoritmo de *seam corretiom* funciona removendo partes que ele entende como movimento de objetos. Os círculos em vermelho e amarelo ilustram como o algoritmo de *blending* funciona na correção de alinhamento de pequenas áreas onde não houve uma boa correlação de *features* e consequentemente falta de alinhamento global.

Figura 3.4- Zoon dado na Figura 3.2.



Fonte: Próprio autor

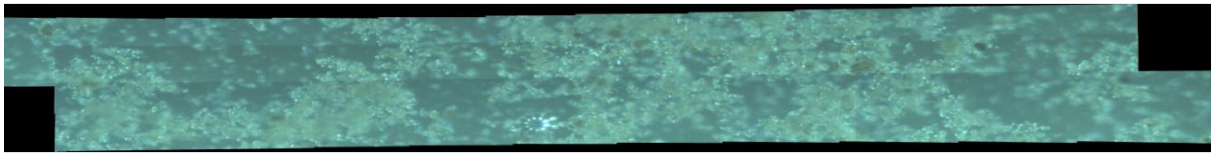
Figura 3.5- Zoon dado na Figura 3.3.



Fonte: Próprio autor

Agora serão discutidos os efeitos do aumento do número imagens na composição final do mosaico. A Figura 3.6 mostra um mosaico criado com 30 imagens. Apesar da falta de alinhamento dos motores do MicroScaner, não se percebe significativas distorções das imagens no mosaico.

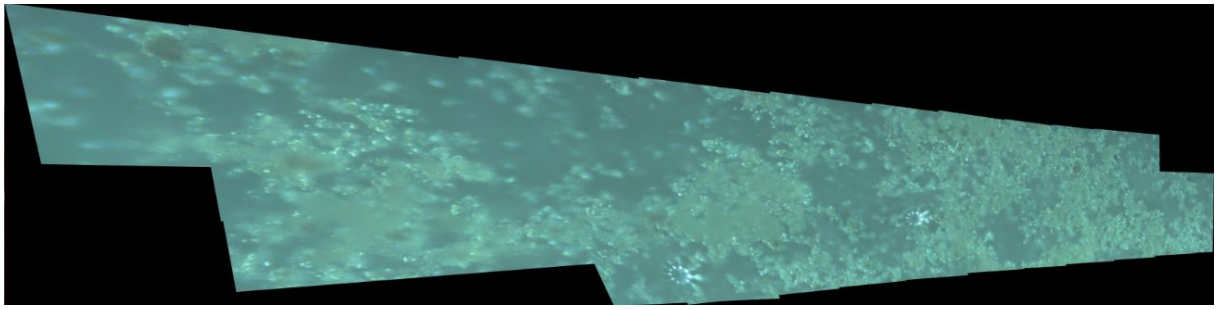
Figura 3.6- Mosaico de 3x10 imagens utilizando-se alinhamento global, sem correção de marcas de junção e sem *blending*. Tempo total: 31.34 minutos.



Fonte: Próprio autor

Entretanto quando se faz o *stitch* de 40 imagens percebe-se claramente a distorção sobretudo das imagens das bordas, como mostrado nas Figuras 3.7 e 3.8. Esse efeito se dá por três motivos, (i) falta de boas correspondências de *features* na vertical (ii) o fato de o algoritmo de *warping* esticar as imagens no intuito de resolver o problema de espaços vazios entre elas, e (iii) a superfície de projeção utilizada é um plano, problema que poderia ser amenizado utilizando uma superfície cilíndrica ou esférica, como dito na seção 2.8.

Figura 3.7- Mosaico de 4x10 imagens utilizando-se alinhamento global, correção de marcas de junção e *blending*. Tempo total: 162.80 minutos, tempo alinhamento global: 153.49 minutos.



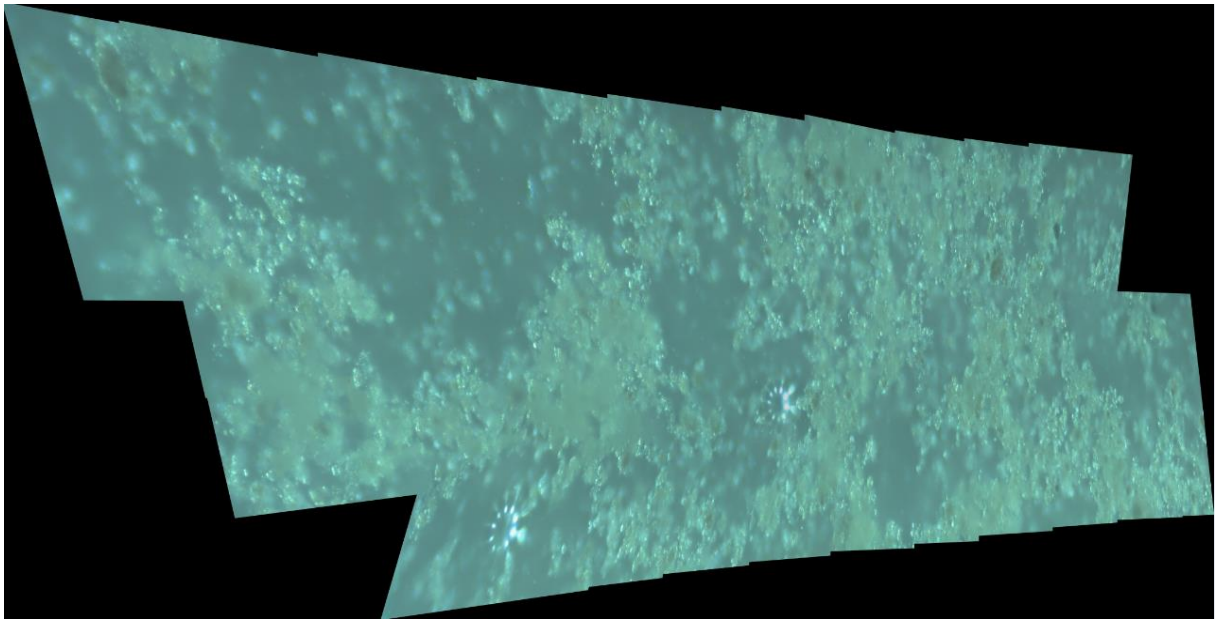
Fonte: Próprio autor

Também é possível amenizar esse problema de alta distorção das imagens melhorando-se as correspondências de *features*. Quando foi diminuído o valor de corte da distância euclidiana de 200 (resultado na Figura 3.7) para 170 (resultado na Figura 3.8) na etapa de pré-correlação de *features*, nesse caso, aumentou-se a amostragem inicial de supostas boas correspondências para o RANSAC e consequentemente este algoritmo retornou um número maior de *inliers*, o que permitiu que a imagem final da Figura 3.8 ficasse menos distorcida que a composição da Figura 3.7.

Segundo (SZELISKI (2010)) o trabalho de se criar mosaicos envolve tanto um esforço artístico quanto computacional. Isso significa dizer que além de ser necessário utilizar

algoritmos rápidos e precisos, também precisa-se utilizar parâmetros condicentes com tipo de imagem utilizada, o que muitas vezes envolve tentativas empíricas.

Figura 3.8- Mosaico de 4x10 imagens utilizando-se alinhamento global, correção de marcas de junção e *blending*. Tempo total: 166.87 minutos, tempo alinhamento global: 156.91 minutos.



Fonte: Próprio autor

Por fim é importante ressaltar que a resultado obtido na Figura 3.8 é uma imagem de 165 Mega Pixels (MP), mostrando que a imagens final tem resolução e campo muito maiores que cada imagem por sozinha.

4 CONCLUSÕES E SUGESTÕES PARA TRABALHOS FUTUROS

Com base no dados e análises apresentados é possível afirmar que a criação de mosaicos baseado na abordagem de *features* é uma tarefa de media a alta complexidade, sobretudo quando o número de imagens a serem ligadas é grande.

Foram também apresentadas e discutidas várias técnicas de processamento de imagens e visão computacional utilizadas nos passos intermediários do processo, como detecção e correlação de *features*, estimativa de matrizes de homografia, parâmetros intrínsecos da câmera, *bundle adjustment*, *seam correction*, *warping* e *blendig*.

Como proposto inicialmente, os resultados obtidos apresentaram significativa qualidade, sobretudo para conjuntos menores de imagens. Além disso, o código produzido neste trabalho não tem limitação do uso de memória via software, apenas por hardware. Outra vantagem é que como todo o código fonte deste trabalho é acessível e editavel o que permite integra-lo facilmente às funcionalidades do MicroScanner.

Por fim pode-se afirmar que a experiência de estudar, programar e utilizar as técnicas apresentadas neste trabalho representa uma excelente forma de complemento técnico, visto que podem ser empregadas tanto na área da engenharia da computação, como da eletrônica, além da de controle e automação.

4.1 Sugestões de trabalhos futuros

Entre as melhorias que futuros trabalhos podem trazer a essa pesquisa encontram-se:

- Colocar o código para procurar correspondências com suas vizinhas superiores direita e esquerda para melhorar a qualidade do alinhamento global.
- Utilizar outras opções de bibliotecas diferentes para fazer o alinhamento global, como discutido na seção 2.5, para diminuir tempo de convergência
- Implementar uma etapa de validação dos resultados, já que é difícil perceber diferenças de detalhes em imagens microscópicas.
- Fazer melhorias no hardware do MicroScanner, sobretudo no movimento dos motores.

5 GLOSSÁRIO

Stitching: Processo de criação de mosaicos.

Stitch de Imagens: Mosaico de imagens.

Features: Regiões características de uma imagem que são utilizadas para futuras buscas de correlação.

Features Matching: Correlação de características.

Keypoints: Pontos com gradientes de valores elevados.

Patch: Região em torno de um *keypoint*.

Inliers: *Features* consideradas como boas correlações pelo RANSAC.

Outliers: *Features* consideradas como más correlações pelo RANSAC.

Threshold: Valor de corte.

Bundle Adjustment: Processo de alinhamento global das imagens para futura composição de um mosaico.

Warping: Processo de distorção de uma imagem para fazer com que suas *features* fiquem do mesmo tamanho que as da sua vizinha.

Seam Correction: Correção marcas de junção. Nessa etapa também são corrigidos problemas de movimento nas imagens

Blending: Processo para suavizar as diferenças de tonalidades entre as fotos.

Mult-blending: Utiliza várias bandas de frequência para suavizar as diferenças de tonalidades entre as fotos.

6 REFERÊNCIAS

AMORIN, L. A. et al. Construção de Mosaico Utilizandi Imagens Aéreas Adquiridas de Forma Autonoma. **XII Simpósio Brasileiro de Automaç˜ao Inteligente**, 25 a 28 de outubro de 2015.

BAY, H.; TUYTELAARS, T.; VAN GOOL, L. SURF: Speeded Up Robust Features. In: LEONARDIS, A.;BISCHOF, H., *et al* (Ed.). **Computer Vision – ECCV 2006: 9th European Conference on Computer Vision, Graz, Austria, May 7-13, 2006. Proceedings, Part I**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006. p.404-417. ISBN 978-3-540-33833-8.

BROWN, M.; LOWE, D. Recognising panoramas. **In Proceedings of the 9th International Conference on Computer Vision (ICCV03),** v. 2, p. 1218–1225, 2003.

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.

BROWN, M.; SZELISKI, R.; WINDER, S. **Multi-Image Matching Using Multi-Scale Oriented Patches**. Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1 - Volume 01: IEEE Computer Society**:** 510-517 p. 2005.

BURT, P. J.; ADELSON, E. H. A multiresolution spline with application to image mosaics. **ACM Trans. Graph.,** v. 2, n. 4, p. 217-236, 1983.

CERES. **Ceres Solver**. Disponivel em: <http://ceres-solver.org/>. Acesso em: 30 set. 2016.

CHALFOUN, J. et al. MIST: Microscopy Image Stitching Tool. Bioinformatics and Biomedicine (BIBM), 2015 IEEE International Conference on, 2014. 9-12 Nov. 2015. p.1757-1757.

CHEN, S. QuickTime VR – An image-based approach to virtual environment navigation. **SIGGRAPH'95,** v. 29, p. 29–38, 1995.

CHEN, S. E. **QuickTime VR: An image-based approach to virtual environment navigation**. Proceedings of the 22nd annual conference on Computer graphics and interactive techniques: ACM**:** 29-38 p. 1995.

CORNELL. Disponivel em: <http://www.cs.cornell.edu/~snavely/bundler/ >. Acesso em: 30 set. 2016.

FISCHLER, M. A.; BOLLES, R. C. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. **Commun. ACM,** v. 24, n. 6, p. 381-395, 1981.

GREENE, N. Environment Mapping and Other Applications of World Projections. **IEEE Computer Graphics and Applications,** v. 6, n. 11, p. 21-29, 1986.

LEICA. **Basics in Microscopy**. Disponivel em: <https://www.leica-microsystems.com/science-lab/>. Acesso em: 01 out. 2016.

LOWE, D. G. Distinctive Image Features from Scale-Invariant Keypoints. **International Journal of Computer Vision,** v. 60, n. 2, p. 91-110, 2004.

MILGRAM, D. Computer methods for creating photomosaics. **IEEE Transactions on Computers,** v. C-24, n. 11, 1975.

PEREZ, P.; GANGNET, M.; BLAKE, A. Poisson image editing. **ACM Trans. Graph.,** v. 22, n. 3, p. 313-318, 2003.

POTSAID, B.; BELLOUARD, Y.; WEN, J. T. Design of an Adaptive Scanning Optical Microscope for Simultaneous Large Field of View and High Resolution. Proceedings of the 2005 IEEE International Conference on Robotics and Automation, 2005. 18-22 April 2005. p.460-465.

SCIENCELEARN. **Resolving Power of Microscopes**. Disponivel em: <http://sciencelearn.org.nz/Contexts/Exploring-with-Microscopes/Sci-Media/Images/Resolving-power-of-microscopes>. Acesso em: 02 jun. 2016.

SHAPIRO, L.; STOCKMAN, G. Computer Vision. **Upper Saddle River, New Jersey: Prentice-Hall, Inc**, p. 157-158, 2001.

STEWART, C. V. Robust Parameter Estimation in Computer Vision. **SIAM Review,** v. 41, n. 3, p. 513-537, 1999.

SZELISKI, R. Video mosaics for virtual environments. **IEEE Computer Graphics and Applications,** v. 16, n. 2, p. 22-30, 1996.

_____. **Image alignment and stitching: A tutorial**. Technical Report MSR-TR-2004-92, Microsoft Research. 2004

SZELISKI, R. **Computer vision: algorithms and applications**. Springer Science & Business Media, 2010. ISBN 1848829353.

SZELISKI, R.; SHUM, H.-Y. **Creating full view panoramic image mosaics and environment maps**. Proceedings of the 24th annual conference on Computer graphics and interactive techniques: ACM Press/Addison-Wesley Publishing Co.**:** 251-258 p. 1997.

TRIGGS, B. et al. Bundle Adjustment — A Modern Synthesis. In: TRIGGS, B.;ZISSERMAN, A., *et al* (Ed.). **Vision Algorithms: Theory and Practice: International Workshop on Vision Algorithms Corfu, Greece, September 21–22, 1999 Proceedings**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000. p.298-372. ISBN 978-3-540-44480-0.

UYTTENDAELE, M.; EDEN, A.; SKELISKI, R. Eliminating ghosting and exposure artifacts in image mosaics. Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on, 2001. 2001. p.II-509-II-516 vol.2.

UYTTENDAELE, M. T.; SZELISKI, R. S. **System and method for exposure compensation**: Google Patents 2004.