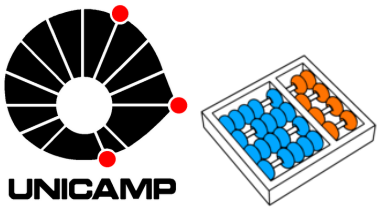

Universidade Estadual de Campinas
Instituto de Computação

MC949 - Visão Computacional

Projeto 1: Panorama

Alunos
Daniel Henriques Pamplona - RA: 260401
Gabriela Martins Jacob - RA: 186087
Igor Engelmann Batista - RA: 260511
Raphael Salles Vitor de Souza - RA: 223641
Renan Matheus Da Silva Florencio - RA: 244808

Professor
Prof. Anderson Rocha

Campinas, 3 de setembro de 2025

1 Introdução

Este projeto tem como objetivo a construção de panoramas. A biblioteca *OpenCV* foi utilizada em *python* para a execução de todo o processo de detecção e extração de características, emparelhamento de características, estimação de homografia e alinhamento, composição e *blending*.

2 Coleta das Imagens

A coleta de dados para a composição da imagem panorâmica foi realizada com o auxílio de um iPhone acoplado a um tripé. O aparelho foi rotacionado em seu próprio eixo, mantendo-o em um plano horizontal constante, a fim de capturar uma sequência de imagens sobrepostas. Durante o processo, buscou-se minimizar a presença de objetos em movimento na cena, garantindo assim uma maior consistência entre os quadros. Além disso, foi dada atenção especial para que as fotografias contivessem um número suficiente de pontos de interesse e texturas, assegurando a presença de descritores visuais adequados para as etapas subsequentes de alinhamento e montagem do panorama.

Para cada um dos três cenários obtidos, foram tiradas mais fotos do que a quantidade de fato utilizada nas atividades. Isso porque, ao utilizar um número muito alto de fotografias, é observado a presença de distorções no panorama final, causando um efeito de "gravata borboleta", em que o panorama apresenta deformações indesejadas devido ao excesso de sobreposição e ao acúmulo de erros de alinhamento. Dessa forma, para o cenário utilizado neste relatório, foram utilizadas 7 imagens, e nos outros dois cenários foram utilizadas 6 imagens para o panorama na Figura 4a e 8 imagens para o panorama na Figura 4b.

3 Detecção e Extração de Características

A primeira etapa de processamento das imagens é a detecção e extração de características. Nessa etapa, busca-se encontrar pontos chave e suas descrições em cada uma das imagens para que seja possível uni-las em panorama.

Os pontos chaves de uma imagem são aqueles que tendem a resistir a transformações e, portanto, podem ser encontrados em diversas posições da imagem. É importante notar que diferentes extratores de *features* podem ser invariantes a diferentes transformações – como escala, rotação, luminosidade e etc – e que as imagens coletadas foram pensadas para evitar ao máximo rotações. A partir desses pontos são extraídas características que serão utilizadas para encontrar o mesmo ponto em outras imagens, como um traqueamento de objetos. Dessa forma, pode-se inferir onde estão os pontos em comum entre duas imagens tiradas com ângulos diferentes, permitindo sua união e reconstrução como uma única imagem.

Os dois algoritmos utilizados e comparados neste trabalho foram *Scale Invariant Feature Transform* (SIFT) e *Oriented FAST and Rotated BRIEF* (ORB).

3.1 SIFT

Proposto por David Lowe, é um dos algoritmos mais utilizados atualmente para realizar detecção e extração de características, sendo invariante a escala e a rotação. O detector funciona a

partir da criação de uma pilha de imagens às quais filtros gaussianos são aplicados progressivamente, com diferentes valores de desvio padrão σ, tornando-as progressivamente mais borradas. Para cada par de duas imagens consecutivas, aplica-se uma diferença de Gaussianas, obtendo-se uma pilha de diferenças de Gaussianas. A nova pilha passa por um processo de busca de extremos, fornecendo um conjunto de candidatos a pontos de interesse para cada uma das diferenças de gaussianas, que serão novamente filtrados para manter somente aqueles mais prominentes. Esse processo em uma única escala é chamado de *octave* e será repetido para a quantidade de *octaves* desejada, diminuindo as escalas das imagens em cada uma das *octaves*.

Para o emparelhamento, é necessário também obter descrições dos pontos de interesse. Em cada região de interesse, subdivide-se o círculo em quatro quadrantes e calcula-se o histograma das direções dos gradientes do quadrante, concatenando os quatro histogramas obtidos em um único histograma. Este será o descritor fornecido pelo SIFT para o emparelhamento de características.

Dentre os parâmetros ajustáveis a partir da biblioteca *OpenCV*, estão o número máximo de pontos chave a serem mantidos; o número máximo de camadas em cada *octave* (o número de *octaves* é calculado automaticamente) e o *smoothing* (σ) aplicado na primeira *octave*. Eles foram ajustados empiricamente a partir dos melhores resultados obtidos.

3.2 ORB

O ORB combina o detector *FAST*, que identifica pontos de interesse avaliando a intensidade dos vizinhos, e o descritor *BRIEF*, que gera descritores binários comparando pares de pixels em uma região da imagem. Essa combinação garante alta eficiência e baixo uso de memória em relação a métodos como SIFT.

No ORB, o *FAST* é usado para detectar pontos, aplicando uma pirâmide de escala para invariância a tamanho e calculando a orientação pelo centroide para invariância a rotação. O *BRIEF* é então orientado conforme essa rotação, e um algoritmo guloso seleciona as melhores *features*.

Na *OpenCV*, parâmetros como número máximo de *features*, fator e níveis da pirâmide, nível inicial e tipo de pontuação podem ser ajustados empiricamente para otimizar os resultados.

3.3 Comparação

O algoritmo ORB foi introduzido como uma alternativa mais rápida ao SIFT, embora menos robusta. No entanto, a quantidade de imagens que serão comparadas neste trabalho é relativamente pequena e o custo computacional dos dois algoritmos é similar e pequeno. Entendemos, portanto, que a maior vantagem do ORB não será bem aproveitada e escolhemos utilizar o algoritmo SIFT, por sua maior robustez e por continuar sendo estado-da-arte em termos de resultados.

4 Detecção automática das imagens

Para montar o panorama, o processo começa detectando pontos-chave e descritores em cada imagem com o SIFT. Esses pontos são características únicas, como cantos ou detalhes de textura, que permitem encontrar correspondências entre imagens. O algoritmo então compara todas as imagens, identificando pares de pontos que coincidem geometricamente usando o RANSAC. Os

pontos confiáveis, chamados inliers, são aqueles que fazem sentido ao alinhar as imagens considerando rotações ou translações. A partir disso, constrói-se uma matriz indicando, para cada par de imagens, quantos inliers existem. Para simplificar, o algoritmo mantém apenas as conexões mais fortes, formando um grafo onde cada imagem se liga às vizinhas mais prováveis. Por fim, define-se a ordem das imagens: começa-se pelas extremidades — com menos conexões — e segue-se para as vizinhas com mais correspondências, respeitando a sobreposição real das imagens.

5 Emparelhamento de Características

5.1 Algoritmos de emparelhamento

Emparelhamento trata-se de buscar, a partir dos descritores dos keypoints encontrados em cenas consecutivas, pontos correspondentes. Para encontrar esses casamentos, podemos utilizar diferentes algoritmos. O utlizado nesse trabalho será o *Brute Force Matcher*.

O algoritmo *Brute Force Matcher*, como o próprio nome diz, se baseia na comparação exaustiva. Basicamente, ele compara todos os descritores de uma imagem com cada um dos descritores de uma segunda imagem, calculando as distâncias. Ele pode ser bastante lento, dependendo do número de pontos comparados.

A decisão de utilizar o *Brute Force Matcher* baseia no fato de que o volume de descritores gerado em cada cena é relativamente baixo. Nessas condições, o ganho de velocidade de algoritmos mais complexos, como o *FLANN*, seria marginal, enquanto o custo computacional da busca exaustiva do *Brute Force* se mantém em um nível totalmente aceitável.

5.2 Aplicação de Filtros

Após a aplicação de cada um dos algoritmos, aplicamos o filtro *ratio test* do David Lowe. Este filtro elimina pares ruins, por meio de uma comparação. Ele escolhe os dois *matches* mais próximos e só aceita caso um deles seja significativamente melhor do que o outro. Testamos diferentes *thresholds*, buscando uma melhoria nos casamentos, por meio de um equilíbrio entre os falsos negativos e falsos positivos.

5.3 Desafios

Grande parte dos nossos cenários eram paisagens naturais, com grande predominância de tons de verde. Percebemos que, nesses cenários, houve uma quantidade considerável de falsos casamentos, e acreditamos que a razão seja a homogeneidade das imagens.

Também utilizamos um cenário de uma sala de aula, que possui muitas cadeiras, padrões e objetos repetidos. Isso levou a alguns falsos casamentos. Para solucionar esse problema, podemos diminuir o *threshold* do *ratio test* de Lowe, mas percebemos que, ao fazer isso, geramos um alto número de falsos negativos.

Portanto, foi necessário testar alguns valores de *threshold* para ajustar a imagem final, considerando tanto os casos de homogeneidade quanto repetição de padrões. Encontrando o ideal global em **0.6**, mas obtivemos resultados melhores ajustando individualmente para cada cenário.

6 Estimação de Homografia e Alinhamento

Neste trabalho implementamos uma abordagem de *stitching* baseado em *features*. Para o alinhamento, utilizamos a técnica de encadeamento de homografias, onde calculamos as transformações entre pares de imagens adjacentes e, em seguida, as compomos para criar um alinhamento global.

6.1 Estimação de Homografia

A estimação de homografia é o processo fundamental que permite encontrar a transformação geométrica entre duas imagens. A abordagem conta que as imagens foram ordenadas da esquerda para a direita. Com essa suposição, na parte da detecção e correspondência de pontos de interesse, utilizamos como entrada o resultado do processo SIFT aplicado entre as imagens adjacentes. A função utilizada para encontrar as correspondências emprega um `BFMatcher` [5.1] e aplica o teste de razão para filtrar as correspondências.

Com os pares com boa qualidade, podemos finalmente calcular as matrize de homografia (H) com uma função que internamente emprega o algoritmo RANSAC (`cv2.findHomography`), no qual relacionamos cada uma das imagens em pares de vizinhos, criando uma "corrente"de matrizes $H_{i \to i+1}$, armazenadas em uma lista. A utilização do RANSAC é crucial para estimar H de forma robusta para filtrar *outliers*.

6.2 Alinhamento

A estratégia do alinhamento é utilizar a composição das matrizes H entre as imagens. Primeiramente, utilizamos a imagem central da nossa lista de imagens ordenadas como o pivô das transformações. Subsequentemente, calculamos as homografias entre as imagens $I_i, I_j, I_k : i < j < k$ na ordenação das imagens, e sendo a imagem I_j o nosso pivô, fazemos a composição das homografias da seguinte forma:

- Imagens à esquerda do pivô: $H_{i \to j} = \prod_{p=i}^{j-1} H_{p \to p+1}$
- Imagens à direita do pivô: $H_{k \to j} = \prod_{m=j}^{k-1} (H_{m \to m+1})^{-1}$

Com essas novas matrizes que relacionam diretamente qualquer imagem com o pivô, podemos renderizar o panorama utilizando `cv2.warpPerspective` para projetar em um canvas final, finalizando o processo de alinhamento.

(a) Panorama Progressivo com 2 imagens

(b) Panorama Progressivo com 3 imagens

(c) Panorama Progressivo com 5 imagens

Figura 1: Panorama progressivo.

7 Composição e Blending

Para o processo de composição do panorama, utilizamos, inicialmente, uma abordagem mais simples, que envolvia a colagem direta dos pixels da imagem que estava sendo adicionada, com posterior tratamento das imperfeições. Essa abordagem permitiu resultados preliminares, mas não atingiu a qualidade desejada. Dessa forma, buscamos transições mais suaves e um processo de *blending* propriamente dito, através da técnica de *feathering*.

7.1 Composição por colagem direta

Para essa abordagem, fizemos uma máscara simples que retornava *True* para os pixels com valores acima de $[0, 0, 0]$. Ou seja, que indicava apenas os pixels válidos da imagem transformada, e não o fundo preto que surgiu após a transformação. Inicialmente, através dessa máscara, nós colamos no panorama apenas os pixels que foram indicados como válidos pela máscara. O resultado obtido (Figura 2a) mostra que surgiram pixels pretos nas linhas de transição das imagens.

Para resolver tal problema, recorremos inicialmente a ferramentas de filtragem, preferencialmente afetando apenas as linhas de transição. Como primeiro passo, realizamos uma filtragem para detecção de bordas (utilizando um filtro Laplaciano). Na sequência, aplicamos um filtro de mediana apenas nos pixels detectados como borda de costura (idealmente). Realizamos esse processo duas vezes em sequência, o resultado é mostrado na Figura 2b. Entretanto, o resultado final não ficou dentro do esperado, e nosso processo de detecção de costuras trouxe também outras bordas da imagem, e a posterior aplicação do filtro de mediana gerou perda de detalhes em diversos pontos da imagem.

(a) Panorama inicial.

(b) Resultado final, após duas filtragens.

Figura 2: Etapas da correção via filtragem

Como alternativa, aplicamos uma máscara mais estrita, garantindo que não houvesse pixels pretos de borda. Aplicamos operações de morfologia matemática (dilatação com *kernel* 5x5, para preencher possíveis buracos internos e posterior erosão com kernel 9x9, para termos 2 pixels de margem) na máscara de pixels válidos. Para diminuir um pouco os efeitos de diferença de iluminação, aplicamos também correções no histograma da imagem, corrigindo o valor médio da iluminação dos pixels. O resultado, mostrado na Figura 3a, mostra uma melhora em relação à tentativa anterior, além de um menor custo de processamento. De qualquer forma, ainda são visíveis as transições súbitas na imagem e o *ghosting*.

7.2 Composição com Blending via Feathering

Para suavizar as transições entre imagens, utilizamos a técnica de *feathering*, em que uma imagem vai gradualmente perdendo influência perto de suas bordas. Escolhemos um gradiente linear e obtivemos ótimos resultados, como mostrado na Figura 3b, em que não mais se observa o *ghosting*, majoritariamente devido à combinação de ajuste de máscara de colagem, normalização dos histogramas e a técnica de *feathering*.

(a) Panorama obtido após colagem direta dos pixels após operações morfológicas na máscara.

(b) Panorama obtido após colagem através do Feathering Blending.

Figura 3: Etapas da correção via filtragem

A execução da técnica se dá pela criação de uma matriz de pesos que corresponde à distância do pixel à borda mais próxima e posterior normalização ao dividir pela maior distância; assim, os pixels centrais possuem valores próximos de 1, enquanto os marginais possuem valores próximos de zero. Na sequência, é feita a composição de fato, somando-se os valores ponderados pelos seus pesos em todas as imagens que contribuem para um pixel, e por fim, é feita uma divisão pelos pesos acumulados, com fins de normalização. A solução é relativamente barata, simples computacionalmente e resolve o problema de *ghosting*, não sendo necessária a aplicação de técnicas mais intensas, como o *Multiband*.

8 Outros resultados

Como mencionado anteriormente, fizemos a captura de 3 cenários diferentes. Mostramos acima o processo feito para um dos cenários e, abaixo, encontram-se os outros dois panoramas resultantes do mesmo processo para os outros dois cenários.

(a) Panorama obtido no segundo cenário.

(b) Panorama obtido no terceiro cenário.

Figura 4: Outros cenários utilizados.