RECONSTRUÇÃO TRIDIMENSIONAL DE UMA CENA A PARTIR DE MÚLTIPLOS PONTOS DE VISTA

Eduardo Schumacher[1], Hadryan Salles[1], Leandra Ulbricht[1], Fabiano G. S. Magrin[1], Francisco Ganacim[1].

[1]Universidade Tecnológica Federal do Paraná, Curitiba, Brasil
(eduardo.schumacher99@gmail.com)

Resumo: O objetivo deste trabalho foi desenvolver uma reconstrução geométrica tridimensional através de ferramentas e *softwares* livres. Como metodologia foram utilizadas as linguagens *Python e JavaScript* e as bibliotecas *OpenCV, Django e Three.js*. Como resultados é apresentado o sistema desenvolvido que é capaz de realizar a reconstrução tridimensional a partir de duas imagens, além de dispor os dados gerados em uma interface gráfica produzida.

Palavras-chave: Reconstrução Tridimensional; Computação Gráfica; Visualização de Dados

INTRODUÇÃO

A reconstrução da geometria tridimensional de uma cena a partir de múltiplos pontos de vista apresenta aplicações em diferentes áreas, como a saúde (Costa, 2021), o georreferenciamento (Baldivieso, 2020), a robótica (Santos, 2010), a geolocalização (Klingner et al., 2013), entre outras. A diversidade de áreas aponta que este procedimento de reconstrução pode ser aplicado quando existe a necessidade da visualização tridimensional de uma cena capturada de forma bidimensional, independente da aplicação final.

As ferramentas e métodos para a resolução dos problemas de reconstrução variam conforme as variáveis que compõem a cena são conhecidas ou não (Solem, 2012). No caso do desenvolvimento da geometria tridimensional de uma cena através apenas de imagens bidimensionais, é essencial o conhecimento dos processos de calibração de câmeras, extração de pontos-chave e descritores das imagens, estimativa de pose de cada uma das câmeras e triangulação de pontos tridimensionais (Hartley e Zisserman, 2004). Para que sejam interpretados, todos estes pontos estimados necessitam de um ambiente tridimensional para que possam ser localizados e analisados.

Os algoritmos apresentados nas aplicações citadas anteriormente variam em termos de funcionalidade e complexidade, se adequando a cada caso explorado. Na maioria destes trabalhos, as técnicas de *Structure from Motion* (Ozyesil et al., 2017) (Schonberger et al., 2016) (McCann, 2008) e *Bundle Adjustment* (Indelman et al., 2015) (Triggs et al., 2000) estão presentes como referência para a construção dos algoritmos de reconstrução tridimensional. Nestes trabalhos, pouco é explorado sobre os procedimentos matemáticos e numéricos que compõem estas técnicas, tornando o processo de reprodução destes algoritmos uma extensa atividade de pesquisa. Além disso, não é apresentada uma forma de organização dos dados gerados pelos algoritmos.

Assim, este trabalho busca, através de ferramentas e *softwares* livres, realizar a reconstrução geométrica tridimensional a partir de múltiplas imagens de uma forma construtiva e contínua e, além disso, propor um ambiente tridimensional de visualização acompanhado de uma estrutura de banco de dados para a organização dos dados gerados.

MATERIAL E MÉTODOS

Foi realizado um estudo exploratório com o foco na reconstrução da geometria tridimensional de uma cena, onde foram selecionados livros e artigos publicados sobre este tema. A partir destes, foram selecionados os algoritmos computacionais que serviriam de base para a programação apresentada neste assunto.

Os algoritmos de computação e visão computacional foram implementados a partir das funções e processos da biblioteca *OpenCV*, utilizada através da linguagem de programação *Python*.

Para checar os resultados obtidos neste estudo, foram utilizados como referência os resultados produzidos pelo *COLMAP* (Schonberger et al., 2016). Este *software* permite reconstruir uma cena tridimensional a partir de um conjunto de imagens, e estimar parâmetros intrínsecos da câmera utilizada na captura dessas imagens.

A linguagem *Python* também foi utilizada para modelar o banco de dados para armazenar todas as imagens e suas respectivas informações. Esse banco de dados foi gerado utilizando a biblioteca *Django*, que disponibiliza funções para criação de tabelas e relações entre diferentes tabelas.

Por fim, para construir uma cena tridimensional com indicadores gráficos para os objetos obtidos, foi utilizada a linguagem *JavaScript* e a biblioteca *Three.js,* que possui funções para renderização de primitivas geométricas com diferentes cores, posições e orientações em um ambiente interativo.

Os softwares e bibliotecas utilizados foram selecionados por serem amplamente utilizados e possuírem grande quantidade de material de referência. Além disso, todas essas ferramentas são de código aberto, com licenças permissivas para a aplicação desenvolvida.

RESULTADOS E DISCUSSÃO

Antes de realizar os processos computacionais de reconstrução com as imagens capturadas, é necessário modelar o principal componente físico da cena que será reconstruída, as câmeras. Neste trabalho apenas um modelo de câmera foi utilizado para a captura de todas as imagens e, inicialmente, o processo de reconstrução foi baseado nas duas imagens apresentadas na Figura 1.



Figura 1. Fotos utilizadas para a reconstrução da cena (autoria própria).

O primeiro passo é o de calibração de câmera. Utilizando fotos do padrão *ChessBoard* que a biblioteca *OpenCV* disponibiliza, foi possível através da função *calibrateCamera* e de 112 fotografias do padrão, como as apresentadas na Figura 2, encontrar a matriz *K* de calibração da câmera, esta que contém os parâmetros intrínsecos do equipamento, como a distância focal e o ponto principal. Além disso, foram recuperados os coeficientes de distorção da lente que são necessários para realizar a correção de distorções físicas aplicadas pela lente da câmera. Vale ressaltar que, para a calibração devolver uma boa estimativa dos parâmetros, as fotografias utilizadas no processo devem fazer o padrão *ChessBoard* percorrer todo o sensor da câmera. A função que retorna a calibração de câmera usa o algoritmo de Zhang (1998) como referência.



Figura 2. Fotos do padrão *Chessboard* utilizadas para a calibração (autoria própria).

O padrão *ChessBoard* é de fácil reconhecimento e como conhecemos suas características métricas é possível estimar como deve ser sua representação em uma imagem produzida pela câmera. Desta forma, ao comparar essa estimativa com as imagens capturadas, é possível realizar este ajuste.

São então retiradas as distorções radiais das imagens que serão utilizadas para a reconstrução da cena utilizando os parâmetros de calibração encontrados e através da função *undistort.* A figura 3 ilustra quais os efeitos deste processo.

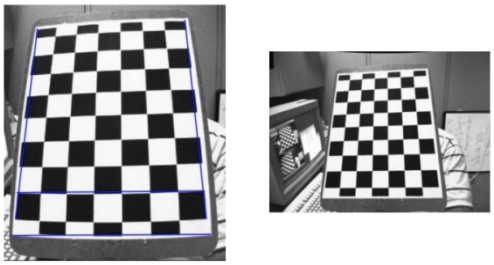

Figura 3. Imagem original (à esquerda) e imagem com as distorções retiradas através da função *undistort* (à direita) (Fonte: *Bradski, 2000)*.

Após a calibração de câmera, já é possível partir para o algoritmo de reconstrução tridimensional. O primeiro passo é encontrar quais pontos bidimensionais das imagens serão triangulados e, para isto, foi utilizado o algoritmo *SIFT* (Lowe, 2004), que está disponível no *OpenCV*.

Essencialmente, através da construção do objeto do SIFT, este é capaz de retirar os chamados pontos-chave e seus descritores das imagens de entrada. Os pontos-chave são pontos que se destacam na imagem, como por exemplo as bordas dos objetos, já que estes pontos contém uma variação de cor em sua vizinhança. Os descritores são responsáveis por caracterizar o ponto-chave e são utilizados para realizar a correspondência de pontos entre as imagens. O SIFT foi escolhido devido à qualidade dos resultados obtidos em relação aos outros tipos de descritores disponíveis na biblioteca. A Figura 4 mostra os pontos-chave das imagens de entrada destacadas.

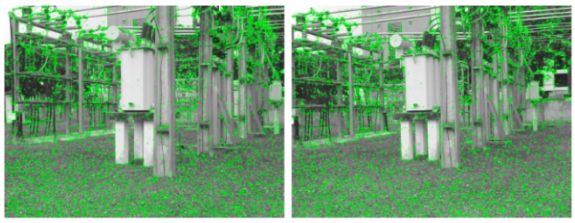

Figura 4. Imagens de entrada com seus pontos-chave destacados (autoria própria).

Agora, com os pontos que possivelmente serão reconstruídos, é encontrada a correspondência entre eles, ou seja, quais pontos da primeira imagem representam os mesmos pontos na segunda e, para isto, foi utilizado o algoritmo KNN (Cover e Hart, 1967).

Para cada ponto-chave em uma das imagens, são selecionados *n pontos* correspondentes na segunda imagem, no caso deste projeto, *n=2*. É realizado um teste de distâncias (*Lowe's ratio test (*Lowe, 2004)) para decidir se esta correspondência é consistente ou não. As correspondências de pontos caracterizados como consistentes serão utilizadas no processo de reconstrução.

O nome da teoria capaz de capturar a geometria projetiva entre dois pontos de vista é a geometria epipolar e a entidade capaz de encapsular essa geometria é a matriz fundamental F (Hartley e Zisserman, 2004). Dado um par de pontos correspondentes x na primeira imagem e x' na segunda, a matriz F cumpre a igualdade exposta na Equação 1.

$$x'Fx = 0 \quad (1)$$

É possível reconstruir a cena utilizando a matriz fundamental, porém o resultado é dado ao menos de uma transformação projetiva arbitrária (Hartley e Zisserman, 2004). Isso significa que, em relação a cena real, o produto da reconstrução pode estar rotacionado, transladado, cisalhado, entre outras distorções.

O processo de calibração de câmera é capaz de retirar esta ambiguidade, deixando esta arbitrariedade apenas em relação a escala da reconstrução.

Como o primeiro passo deste projeto foi realizar a calibração de câmera, a matriz fundamental F é substituída pela matriz essencial E, onde a relação entre estas é tal qual a Equação 2.

$$E = K^t F K \quad (2)$$

Utilizando os pontos encontrados pelo *SIFT* e a matriz de calibração *K*, a matriz *E* correspondente ao par de imagens utilizada neste estudo é encontrada através da função *findEssentialMat*, que utiliza o método RANSAC para encontrar a matriz *E* que melhor se ajusta a Equação 2 com as informações fornecidas.

A informação da matriz essencial é suficiente para que se possa extrair os parâmetros extrínsecos das câmeras, estes que se referem à posição destes equipamentos na cena. Se estabeleceu uma das câmeras como a origem do sistema de coordenadas global e, através da função *recoverPose,* foi encontrada a rotação R_2 e a translação t_2 que se deve aplicar à posição da outra câmera para transportá-la para a origem. Essencialmente, se encontrou a posição da segunda câmera. Como a primeira câmera está na origem, sua rotação é a identidade (R_1=I) e a translação é o vetor nulo (t_1=0).

Com os parâmetros intrínsecos e extrínsecos, foram construídas as matrizes de câmera P_1 e P_2 que representam os equipamentos utilizados para registrar as duas imagens que compõem a reconstrução nesse trabalho. Estas matrizes são decompostas tal qual a Equação 3.

$$P_i = K[R_i|t_i], i = 1, 2 \quad (3)$$

Em termos de reconstrução, cada matriz P de câmera cumpre a relação apresentada na Equação 4.

$$PX_i = \lambda_i x_i \quad (4)$$

Onde X_i é um ponto tridimensional, que é projetado no plano de projeção da câmera no ponto x_i e λ_i é um fator de escala arbitrário proveniente da reconstrução tridimensional. Este fator de escala pode ser eliminado caso sejam conhecidas as distâncias reais entre objetos que compõem a cena.

Para cada correspondência x_i e x_i' de pontos entre as duas imagens, e suas respectivas matrizes de câmera, utilizou-se a Equação 5 para encontrar os pontos tridimensionais relacionados a esta correspondência bidimensional.

$$\begin{bmatrix} P_1 & -x & 0 \\ P_2 & 0 & -x' \end{bmatrix} \begin{bmatrix} X \\ \lambda \\ \lambda' \end{bmatrix} = 0 \quad (5)$$

Esse sistema de equações é resolvido através da função *triangulatePoints* e, desta forma, são computados os pontos tridimensionais.

Uma vez obtidas a posição e orientação da câmera e o conjunto de pontos-chave para cada imagem processada, além dos pontos tridimensionais triangulados, é possível visualizar a cena. Para isso, foi criado um banco de dados capaz de acomodar estas informações. O conteúdo armazenado por cada tipo de objeto pode ser visualizado na Tabela 1.

O objeto Imagem armazena o arquivo (JPG) da imagem e a posição e orientação na cena da câmera referente à imagem em questão. A tabela Ponto-Chave guarda qual imagem contém o ponto, a posição (2D) do ponto-chave nessa imagem, a cor do pixel da imagem na posição do ponto chave, seus descritores e o ponto da cena com o qual é

relacionado. Por fim, a tabela de Pontos da Cena deverá possuir apenas sua posição (3D).

Tabela 1. Modelo do banco de dados

Objeto	Dados
Imagem	id, imagem, posição e orientação (3D) da câmera
Ponto-Chave	id, id da imagem, posição (2D), cor do pixel, descritores e id do ponto da cena
Ponto da Cena	id, posição (3D)

Após preencher este banco com os dados obtidos pelo processamento, foi feita a visualização da cena tridimensional utilizando a biblioteca *Three.js* para renderização de primitivas. Cada ponto da cena foi representado com um esfera de raio pequeno (ajustável), com a cor determinada pela média entre as cores dos pontos chaves relacionados ao ponto da cena. A opacidade destes pontos é ajustável. As imagens foram representadas por pirâmides de base retangular, com tamanhos, posição e orientação correspondentes aos dados pelas imagens. A altura dessa pirâmide é calculada de acordo com a distância focal da câmera utilizada para capturar as imagens. Também foi aplicada a textura da imagem na base dessa pirâmide retangular. A Figura 5 mostra a cena reconstruída na interface gráfica projetada.

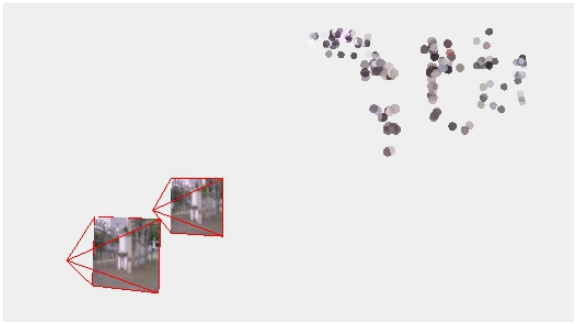

Figura 5. Cena tridimensional reconstruída com todos os elementos estimados (autoria própria).

Agora com a reconstrução realizada, é possível adicionar novos elementos, os quais podem ser novas câmeras e novos pontos tridimensionais.

O processo de adicionar novas câmeras a cena é tratado individualmente, ou seja, para cada nova imagem de entrada, o processo a ser realizado será o mesmo, onde os únicos parâmetros necessários são os elementos já presentes na cena reconstruída.

O problema de encontrar a posição de uma nova câmera em relação à uma cena reconstruída é chamado de *Point-n-Perspective*. A Equação 4 é utilizada novamente, desta vez com os pontos bidimensionais da imagem que será inserida em conjunto com os pontos tridimensionais já conhecidos, desta forma é possível estimar a matriz *P* da nova câmera.

O algoritmo *SIFT* é novamente utilizado para gravar os pontos-chave da imagem que será agregada à cena. A Figura 6 mostra a imagem que será adicionada.



Figura 6. Imagem que será adicionada à cena já reconstruída (autoria própria).

Para cada ponto 3D na cena reconstruída existem 2 pontos 2D, um para cada câmera utilizada no processo de reconstrução. Portanto, para encontrar as correspondências tridimensionais da nova câmera, é realizada a relação entre os pontos 2D da nova imagem com os pontos 2D das imagens já presentes na cena, desta forma, essa correspondência é estendida para os pontos 3D.

Assim, é utilizada a função *SolvePnPRansac*, que encontra a melhor matriz *P* que se adequa ao sistema da Equação 4, encontrando assim a rotação e a translação referentes a nova câmera que será adicionada à cena. Novamente, a posição da câmera na cena é encontrada. A Figura 7 mostra a cena da Figura 5 com a nova posição de câmera localizada.

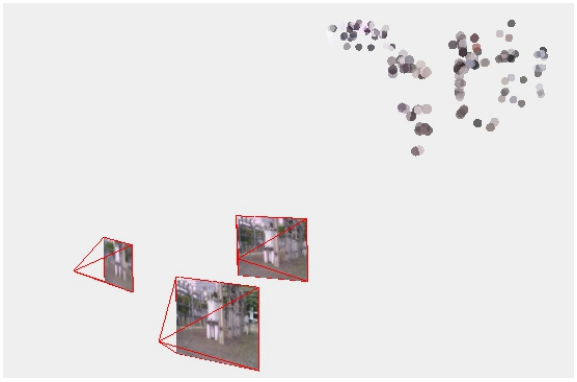

Figura 7. Cena tridimensional reconstruída com a adição de uma nova câmera (autoria própria).

Agora, retornando à Equação 4, como tem-se três matrizes de câmera, cada ponto-chave das imagens pode dar origem a pontos tridimensionais. Porém, como as câmeras tem seus parâmetros todos estimados, estes valores estão sujeitos a erros, assim como a própria obtenção de pontos-chave.

O algoritmo mais utilizado para resolver estes problemas numéricos oriundos dessas estimativas é o *Bundle Adjustment,* que se caracteriza pelo procedimento final nos algoritmos de reconstrução tridimensional (Lourakis e Argyros, 2009).

Segundo Triggs et al. (2000), o objetivo dos algoritmos de *Bundle Adjustment* é, através dos elementos presentes na cena, reestimar os parâmetros das câmeras de forma ótima e global. O nome *bundle* vem do conjunto de raios saindo de cada ponto tridimensional e convergindo ao centro das câmeras, estes que são ajustados em relação aos pontos-chave das imagens e aos centros das câmeras (Triggs, 2000).

O *software* COLMAP realiza o processo de *bundle adjustment* ao final de toda cena que é reconstruída por ele, além de oferecer uma opção de aplicar novamente este algoritmo em uma cena já construída. Os dados de uma cena podem ser exportados e reinterpretados pela interface gráfica construída para o projeto, como é apresentado pela Figura 8.

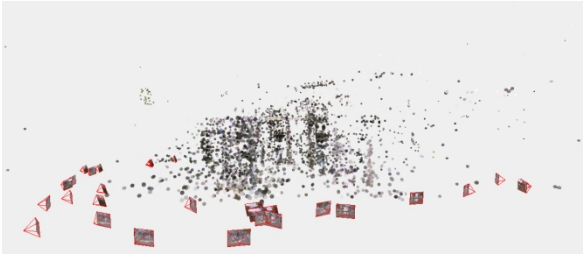

Figura 7. Cena tridimensional reconstruída pelo *COLMAP* visualizada na interface gráfica (autoria própria).

Além disso, é possível ler o banco de dados criado pelo *COLMAP*,e utilizar estes dados como ponto de referência para a alteração e expansão da cena que está sendo representada.

CONCLUSÃO

Através da metodologia adotada para este trabalho, onde os problemas foram estudados de forma isolada, para que o todo fosse composto gradualmente, foi possível realizar a construção de um conhecimento mais aprofundado acerca das estruturas matemáticas e lógicas do problema de reconstrução tridimensional.

A organização de referências bibliográficas de livros e trabalhos publicados traz um caminho direto para a resolução do problema, se preocupando com os pré-requisitos teóricos necessários para o avanço neste campo da computação gráfica e visão computacional.

Além disso, a organização proposta para o banco de dados propõe salientar as relações entre os objetos que compõem a cena.

Como um projeto em construção, os próximos passos se referem a compreender como ampliar a densidade de pontos tridimensionais da cena. Além disso, implementar o algoritmo de *bundle adjustment* de forma que as cenas construídas tenham o melhor resultado possível.

AGRADECIMENTOS

Os autores agradecem à UTFPR pelo apoio e infraestrutura disponibilizada para o desenvolvimento desta pesquisa e a COPEL - Distribuição pelo apoio e financiamento dos recursos para realização do projeto P&D ANEEL/COPEL Distribuição PD-2866-0528/2020 - DESENVOLVIMENTO DE METODOLOGIA PARA ANÁLISE AUTOMÁTICA DE IMAGENS TÉRMICAS.

REFERÊNCIAS

BALDIVIESO, T. J. M. **Um Estudo do Uso de VANTS para a Reconstrução de Cenas 3D.** Dissertação (Mestrado em Sistemas e Computação) - Instituto Militar de Engenharia, Rio de Janeiro, 2020.

BRADSKI, G. The OpenCV Library. **Dr. Dobb's Journal of Software Tools.** v. 25, n. 11, 2000.

COSTA, G. R. **Reconstrução de um modelo 3D a partir de imagens térmicas 2D de uma mama via câmera infravermelha**. Dissertação (Mestrado em Engenharia Mecânica) - Universidade Federal de Uberlândia, Uberlândia, 2021.

COVER, T. M. HART, P. E. Nearest Neighbor Pattern Classification. **IEEE Transactions on Information Theory**. v. 13, n. 1, 1967.

HARTLEY, R. ZISSERMAN, A. **Multiple View Geometry**, 2 ed. Cambridge University Press, Cambridge, 2004.

INDELMAN, V. ROBERTS, R. DELLAERT, F. Incremental Light Bundle Adjustment for structure from motion and robotics. **Robotics and Autonomous Systems**, v.70, p.63-82, 2015.

KLINGER, B. et al. Street View Motion-From-Structure-From-Motion. **International Conference on Computer Vision**, p. 953-960, 2013.

LOURAKIS, M. I. A. ARGYROS, A. A. SBA: A software package for generic sparse bundle adjustment. **ACM Transactions on Mathematical Software**, v. 36, n. 1, 2009

MCCAN, S. 3D Reconstruction from Multiple Images. **Foundations and Trends in Computer Graphics and Vision**, v. 4, p. 287-404, 2008.

OZYESIL, O. et al. A survey of Structure from Motion. **Acta Numerica**. v. 26, p. 305-364, 2017.

SANTOS, G. L. **Localização de Robôs Móveis Autônomos Utilizando Fusão Sensorial de Odometria e Visão Monocular**. Dissertação (Mestrado em Ciências) - Universidade Federal do Rio Grande do Norte, Natal, 2010.

SCHONBERGER, J. L. et al. Structure-From-Motion revisited. **IEEE Conference on Computer Vision and Pattern Recognition**, p. 4104-4113, 2016.

SOLEM, J. E. **Programming Computer Vision with Python**, 1 ed. O'Reilly Media, Sebastopol, 2012

TRIGGS, B. et al. Bundle Adjustment: A Modern Synthesis. **International Workshop on Vision Algorithms**, p. 298-372, 2000.

ZHANG, Z. A Flexible New Technique for Camera Calibration. **IEEE Transactions Pattern Analysis and Machine Intelligence**, v. 22, n. 11, p. 1330-1334, 2000.