ISEP

Instituto Superior de Engenharia do Porto

Estrutura a partir do Movimento em Sistemas Autónomos

Artur Jorge Martins Junqueira

Tese submetida no âmbito do
Mestrado em Engenharia Electrotécnica e de Computadores
Ramo de Sistemas Autónomos

Orientador: José Miguel Soares de Almeida

Novembro de 2014

Agradecimentos

Agradeço ao Engenheiro José Miguel Soares de Almeida, pela orientação, disponibilidade e ajuda no decorrer deste trabalho.

Agradeço ao Engenheiro André Dias pela ajuda e disponibilização de conjuntos dados, para a realização dos testes experimentais.

Agradeço a todos os Engenheiros do Instituto Superior de Engenharia do Porto, destacando os docentes do Laboratório de Sistemas Autónomos pelo conhecimento transmitido ao longo do mestrado.

Agradeço aos familiares e amigos por todo o apoio e ajuda constante ao longo desta dissertação.

Resumo

A estimação da estrutura do espaço e do movimento das câmaras, a partir de um conjunto de pontos correspondentes entre as imagens, é um tópico de pesquisa na comunidade da visão computacional, conhecido como *Structure from Motion* (SfM).

Nesta dissertação pretende-se explorar uma ferramenta SfM e analisar o seu desempenho, tendo em consideração a sua aplicabilidade em sistemas autónomos. Para a análise são considerados dois cenários de aplicação: pós-processamento de imagens adquiridas durante operações robóticas (por exemplo para a criação de modelos tridimensionais) e aplicações em tempo real como a perceção do ambiente de operação.

A ferramenta em estudo é o projeto Bundler de Noah Snavely. Este projeto estima os parâmetros das câmaras e os pontos 3D do ambiente, a partir de um conjunto de imagens e um lista de pontos de interesse, correspondentes entre as imagens.

Na análise do Bundler pretende-se evidenciar os principais blocos que exigem maior esforço computacional, o *matching* de imagens e a estimação iterativa dos parâmetros das câmaras com base na diminuição do erro de reprojeção, através da biblioteca *Sparse Bundle Adjustment* (SBA).

Para o processo de deteção e *matching* de pontos de interesse, efetuou-se a comparação entre a aplicação *Scale Invariant Feature Transform* (SIFT) e uma adaptação deste para a *Graphics Processing Unit* (GPU), o SiftGPU. Os resultados da comparação demonstraram uma melhoria significativa na utilização do SiftGPU, com uma placa gráfica Nvidia com *Compute Unified Device Architecture* (CUDA).

Na exploração do Bundler para aplicações de pós-processamento, realizaram-se testes com diversos conjuntos de imagens e as nuvens de pontos geradas, permitem observar o modelo tridimensional dos objetos no espaço. Neste tipo de aplicações, a restrição do tempo de processamento não é uma condicionante tão grave como numa aplicação em tempo real. Então para tentar optimizar o desempenho do processo de estimação dos parâmetros das câmaras, apresentaram-se duas propostas. A primeira, considerando que se utiliza a mesma câmara e que esta tem uma distância focal fixa, consiste em efetuar uma calibração prévia dos parâmetros intrínsecos e fornecer ao Bundler, evitando assim a estimação de três parâmetros, e consequentemente redução do tempo de processamento do *Bundle Adjustment* (BA).

A segunda proposta, implica a utilização de dados sensoriais de IMU/GPS para fornecer uma óptima inicialização dos parâmetros extrínsecos das câmaras. Com os parâmetros extrínsecos provenientes do IMU/GPS, em conjunto com os parâmetros intrínsecos de calibração, pretende-se reduzir o tempo de estimação dos parâmetros das câmaras no processo iterativo BA. Através da utilização destes dados, também se pretende evitar o desvio da estimação, devido à acumulação de erros ao longo do processo. Uma das vantagens da utilização do IMU/GPS, é facto de os resultados serem gerados em relação a um sistema de coordenadas, o que não acontece no Bundler, em que as posições são relativas e com ambiguidade na escala.

Para adaptar o Bundler ao contexto de uma aplicação em tempo real, procedeu-se à alteração da abordagem atual, para enormes conjuntos desordenados de imagens, para um método sequencial e ordenado de acordo com o processo de aquisição de imagens. Este é apenas um dos primeiros passos para modificar o Bundler de acordo com as exigências das aplicações robóticas.

Palavras Chave: Bundler, Geo-Bundler, *Bundle Adjustment*, Reconstrução 3D, SIFT, SiftGPU, Sistemas Autónomos

Abstract

The estimation of the structure and cameras movement, from a set of corresponding points between images, is a research topic in the computer vision community, known as Structure from Motion (SFM).

This dissertation aims to explore an SfM tool and analyze its performance, taking into account their applicability in autonomous systems. Two application scenarios are considered for the analysis: post-processing of images acquired during robotic operations (for instance, the creation of three-dimensional models) and real-time applications, such as the perception of the operating environment.

The tool in question is the Noah Snavely's Bundler project. This project estimates the parameters of the cameras and the 3D environment points, from a set of images, image features and image matches.

With the analysis of Bundler it's intended to highlight the main blocks that require higher computational effort, the image pairwise matching and the iterative estimation of the cameras' parameters based on the reduction of the reprojection error, through the library Sparse Bundle Adjustment (SBA).

For the process of keypoint detection and matching, it was carried out a comparison between the Scale Invariant Feature Transform (SIFT) and an adaptation of this application for the Graphics Processing Unit (GPU), the SiftGPU. The comparison results showed a significant improvement in the use of SiftGPU, with an Nvidia graphics card with Compute Unified Device Architecture (CUDA).

During the exploration of Bundler for post-processing applications, tests were conducted with different image sets and the point clouds generated allowed us to observe the three-dimensional model of the objects in space. In this kind of applications, the restriction of processing time is not a serious condition such as it is in a real-time application. When trying to optimize the estimation process performance of the cameras' parameters, two proposals were presented. The first one, considering that the same camera is used, and it has a fixed focal distance, consists in performing a prior calibration of the intrinsic parameters, and provide them to the Bundler, thereby preventing the estimation of three parameters, and thus reducing the processing time of the Bundle Adjustment (BA).

The second proposal involves the use of sensory data from the IMU/GPS to provide an optimal initialization of the cameras' extrinsic parameters. The extrinsic parameters from the IMU/GPS, together with the intrinsic calibration parameters, are intended to reduce the estimation of the cameras' parameters during the BA iterative process. Through the use of this data, it's also intended to avoid the deviation of the estimation due to accumulation of errors along the process. One of the advantages of using the IMU/GPS, is that the results are generated relative to a coordinate system, which does not happen in Bundler, in which the positions are relative and ambiguous in scale.

To adapt the Bundler to the context of a real-time application, the current approach was modified, from large unordered collections of images, to a sequential and ordered method, according with the process of image acquisition. This is just one of the first steps to modify Bundler in accordance with the requirements of robotic applications.

Keywords: Bundler, Geo-Bundler, Bundle Adjustment, 3D Reconstruction, SIFT, SiftGPU, Autonomous Systems

Conteúdo

Lista de Figuras

Lista de Tabelas

Acrónimos

2D	Bidimensional
3D	Tridimensional
ANN	*Approximate Nearest Neighbors*
BA	*Bundle Adjustment*
CCD	*Charge-Coupled Device*
CMVS	*Clustering Views for Multi-view Stereo*
CUDA	*Compute Unified Device Architecture*
DLT	*Direct Linear Transformation*
DoG	*Difference of Gaussian*
EXIF	*Exchangeable Image File Format*
fps	*frames per second*
GPS	*Global Positioning System*
GPU	*Graphics Processing Unit*
IMU	*Inertial Measurement Unit*
ISEP	Instituto Superior de Engenharia do Porto
LSA	Laboratório de Sistemas Autónomos
MVS	*Multi-view Stereo*
PMVS	*Patch-based Multi-view Stereo*
RANSAC	*RANdom SAmple Consensus*
SBA	*Sparse Bundle Adjustment*
SfM	*Structure from Motion*
SIFT	*Scale Invariant Feature Transform*
SLAM	*Simultaneous Localization And Mapping*
SVD	*Singular Value Decomposition*

1 Introdução

Uma das principais propriedades para considerar um robô como autónomo é a capacidade de perceção, localização e navegação num ambiente desconhecido, usando a informação recolhida através dos seus sensores. A navegação é considerada como uma das áreas de investigação com maior relevo nos sistemas autónomos [1].
Em geral, os sistemas autónomos estão equipados com sistemas de navegação constituídos por sensores como o *Inertial Measurement Unit* (IMU) e *Global Positioning System* (GPS). A fusão sensorial da informação proveniente destes sensores, é implementada recorrendo a filtros probabilísticos como o filtro de Kalman. Em cenários onde não é possível utilizar o GPS, como situações subaquáticas, subterrâneas ou ambientes *indoor*, os sistemas de navegação apenas podem utilizar sensores inerciais. Os IMUs de elevada qualidade podem fornecer uma solução de navegação com precisão durante um determinado período de tempo, mas estes dispositivos têm um custo elevado e podem ser demasiado pesados, no caso de sistemas de dimensões reduzidas. Alem disso, estes sensores acumulam sempre erros. Os sensores inerciais construídos com tecnologias *Microelectromechanical Systems* (MEMS) são de baixo custo e têm dimensões reduzidas, sendo ideais para sistemas compactos. No entanto, estes sensores têm menor qualidade nas medidas e permitem apenas operações confiáveis para curtos períodos de tempo. Então, a integração de visão computacional em sistemas de navegação surge como uma das alternativas, para colmatar estas limitações.

Na comunidade robótica, o processo incremental de estimação e localização (*egomotion*), a partir de um sistema de visão, é denominado como odometria visual. A

odometria visual tem um grande potencial: à medida que o poder computacional aumenta, uma câmara de baixo custo pode ser utilizada para visualização e medição do movimento, substituindo ou complementado os típicos sensores de medição baseados em *lasers* e sonares, para determinação do movimento relativo a objetos no ambiente, e os *encoders* para o *dead reckoning*.
Abordagens como *Simultaneous Localization And Mapping* (SLAM) e *Structure from Motion* (SfM) focam os problemas de estimação do movimento da câmara, e simultaneamente calculam a geometria do espaço. A reconstrução eficiente da geometria do espaço é crucial para o sucesso da navegação de um sistema autónomo, em ambientes desconhecidos ou variáveis ao longo do tempo.

Um dos tópicos mais explorados na comunidade robótica é o problema do SLAM. O SLAM consiste na estimação do movimento e mapeamento em simultâneo, a partir de informação sensorial. Consoante a visão computacional foi evoluindo, a visão foi se tornando uma técnica sensorial predominante nos problemas de localização e mapeamento, e também devido ao facto de as câmaras serem dispositivos compactos, precisos e de baixo custo comparativamente aos sistemas de medição de distância baseados em *lasers*.
A introdução da visão nos problemas de localização e mapeamento, originou um novo conceito que se designa de SLAM Monocular. O SLAM Monocular consiste na utilização de uma câmara como sensor principal para a localização e mapeamento.

Na visão computacional, um dos tópicos de interesse é o problema SfM. O SfM consiste na reconstrução da geometria do espaço, assim como a estimação da posição relativa das câmaras, a partir de uma sequência de imagens. Este processo pode ser realizado a partir de um sistema com uma câmara, ou com múltiplas câmaras. A vantagem de utilizar múltiplas câmaras é que o movimento e a estrutura podem ser calculados diretamente numa escala absoluta, no caso da distância entre as câmaras ser conhecida. Quando se utiliza um câmara a distância absoluta pode ser calculada de outra forma, através da medida do movimento da *baseline* ou do tamanho de um elemento no espaço, ou usando outros sensores como IMU e GPS.

Apesar de oriundos de diferentes comunidades científicas, existe uma relação próxima entre o SfM e o SLAM Monocular. Ambos utilizam uma sequência de imagens para obter a estrutura do espaço (no SLAM refere-se ao mapa) e a posição do sistema. Apesar das similaridades, o foco do problema é um pouco diferente. No SLAM Monocular, normalmente, apenas a posição da câmara no momento de aquisição da última imagem, é que tem interesse para tomar decisões de navegação. As posições anteriores podem ser descartadas reduzindo o número de estados a serem estimados. O SLAM tem uma abordagem probabilística e permite integrar de uma

forma simples múltiplos sensores no modelo probabilístico de movimento. Outra característica do problema SLAM é que a estimação do movimento ocorre em tempo real, enquanto observa e mapeia o ambiente desconhecido.

1.1 Motivação

O Laboratório de Sistemas Autónomos (LSA) é uma unidade de pesquisa e desenvolvimento do Instituto Superior de Engenharia do Porto (ISEP). Este laboratório desenvolve sistemas autónomos e sub-sistemas com estes relacionados, para operar em diferentes ambientes e aplicações como monitorização, segurança, busca e salvamento. Neste sentido, no LSA desenvolvem-se sistemas sensoriais que permitem o aumento das capacidades de perceção, navegação e controlo dos sistemas autónomos.

O ROAZ, FALCOS e o TIGRE são alguns exemplos de projetos de sistemas autónomos desenvolvidos no LSA. O projeto ROAZ é um veículo autónomo marítimo desenvolvido para monitorização do ambiente, batimetria, recolha de dados marítimos, suporte em busca e salvamento e missões de segurança. O FALCOS é um UAV concebido para baixas altitudes e principalmente para aplicações civis como prevenção de fogos florestais, segurança, monitorização do ambiente e aquisição de imagens aéreas. O TIGRE é um veículo autónomo terrestre para exploração e operação em ambientes não estruturados e para aplicações como transporte em *outdoor*, segurança e busca e salvamento.

No LSA existe um esforço contínuo para melhorar os sistemas de perceção do ambiente, para a deteção e desvio de obstáculos, navegação em ambientes desconhecidos, localização em ambientes em que não estão disponíveis sistemas GPS e para mapeamento tridimensional. Durante as operações robóticas, os sistemas autónomos, adquirem informação sobre o ambiente de operação através de sistemas de visão e/ou sistemas baseados em *laser* com IMU e GPS. No sentido de continuidade da otimização dos sistemas de perceção, nesta dissertação pretende-se explorar a informação proveniente de sistemas de visão em conjunto com GPS/IMU, utilizando técnicas de SfM, para a estimação do movimento e da estrutura tridimensional do espaço.

Com base na contextualização e cenários de aplicação, a dissertação consiste em estudar o problema SfM, nomeadamente, estudar o estado da arte deste tópico e de ferramentas que possam ser utilizadas no contexto da robótica, para aplicações em tempo real e/ou para pós-processamento de conjuntos de dados obtidos durante as operações robóticas.

1.2 Objetivos

O tema da dissertação aborda um problema de interesse para a visão computacional, o *Strucutre from Motion*. O SfM tem sido uma das áreas mais exploradas nas últimas décadas, atingindo um determinado estado de maturidade. Alguns algoritmos SfM foram aplicados no setor comercial em aplicações como [2, 3, 4].

O objetivo principal deste trabalho consiste em estudar o estado da arte de projetos para problemas SfM e explorar a sua aplicabilidade no contexto da robótica. Durante o processo de exploração, pretende-se caracterizar o projeto SfM tendo em consideração dois cenários de aplicação: operações em tempo real e pós-processamento de conjuntos de dados, provenientes de operações robóticas, como por exemplo, tarefas de mapeamento e monitorização. Nesta caracterização, pretende-se analisar os resultados obtidos em função do tempo de processamento assim como verificar a qualidade das nuvens de pontos tridimensionais, e se possível, comparar os resultados com informação de referência (*ground truth*). Se o processo de caracterização evidenciar que o desempenho não corresponde aos requisitos da robótica (tempo processamento reduzido e precisão na reconstrução 3D), ambiciona-se apresentar soluções de otimização e, se necessário, efetuar algumas modificações nesse sentido.

1.3 Organização da dissertação

Esta dissertação está organizada em 9 capítulos. No Capítulo 1 realiza-se o enquadramento do tema da dissertação e explicam-se as motivações e os objetivos.

No Capítulo 2 apresentam-se algumas abordagens e ferramentas para resolver os problemas de reconstrução 3D para enormes conjuntos de imagens.

No Capítulo 3 abordam-se os fundamentos teóricos, necessários para o desenvolvimento do trabalho, como o modelo da câmara, geometria epipolar, reconstrução 3D a partir de duas imagens e alguns algoritmos utilizados no SfM.

No Capítulo 4 realiza-se a descrição do algoritmo de estimação da estrutura e do movimento, do projeto Bundler.

O Capítulo 5 refere-se à descrição do Bundler e procede-se à análise da sua aplicabilidade no contexto da robótica.

No Capítulo 6 apresentam-se os resultados da exploração do Bundler, para diversos conjuntos de imagens, adquiridos por sistemas autónomos. Neste capítulo são apresentadas as nuvens de pontos 3D geradas pelo Bundler e, realiza-se uma análise do seu desempenho. A ferramenta utilizada para deteção de pontos de interesse nas

imagens é o SIFT. Como existe uma adaptação do SIFT para a GPU, efetuou-se um teste comparativo, para analisar o desempenho de ambas as aplicações.

No Capítulo 7 realiza-se uma análise dos resultados obtidos da exploração do Bundler e, apresentam-se propostas para tentar otimizar o desempenho do processo de reconstrução. As propostas consistem na inicialização dos parâmetros intrínsecos e extrínsecos das câmaras e, na alteração da abordagem atual do Bundler.

O Capítulo 8 descreve a implementação das propostas definidas para otimizar o Bundler, assim como os resultados obtidos.

O Capítulo 9 refere-se às considerações finais perante os resultados obtidos, e são apresentadas propostas para trabalhos futuros.

2

Estado da Arte

O objetivo global do SfM é a determinação das posições dos pontos 3D do espaço (estrutura) e as posições das câmaras que capturam o espaço (movimento, calibração externa da câmara) através da correspondência de características entre as imagens (pontos, cantos, linhas, entre outros). Atualmente, algumas aplicações de SfM estimam os pontos esparsos do espaço tridimensional como um sub-produto do cálculo da posição da câmara [5]. Sabendo as posições das câmaras, podem-se aplicar métodos de reconstrução *Multi-View Stereo* (MVS) para construir o modelo denso 3D do espaço [6]. A reconstrução da nuvem esparsa de pontos 3D pode ser utilizada em várias aplicações como o mapeamento 3D do ambiente, localização de robôs através do modelo do espaço ou simplesmente para desvio de obstáculos.

Uma das possíveis aplicações do SfM consiste na combinação de imagens/fotografias em diferentes condições e perspetivas, a partir do solo ou da atmosfera (satélite ou imagens aéreas). A enorme diversidade de conjuntos de imagens disponíveis na Internet, sendo estas imagens desorganizadas, não calibradas, com diferentes resoluções, qualidade e iluminação despertou o interesse nos problemas SfM de grande escala. Os problemas SfM de grande escala têm ganho mais atenção nos últimos tempos, consoante o SfM foi se tornando uma das principais tecnologias em aplicações como a reconstrução tridimensional de cidades [7].

O *Bundle Adjustment* (BA) é o aperfeiçoamento não linear dos parâmetros das câmaras e das posições dos pontos. Este algoritmo é o elemento chave da maioria dos sistemas SfM, e também a componente que mais consome tempo de processamento

para conjuntos de imagens de grande escala. Conforme os conjuntos de imagens crescem até as centenas de milhar ou até milhões, a escalabilidade do BA torna-se um problema crítico, exigindo novas abordagens para solucionar este problema [7, 8, 9, 10, 11].

Atualmente, existem disponíveis no mercado várias ferramentas para problemas SfM. Em [12] realiza-se uma comparação de algumas ferramentas SfM que aceitam conjuntos de imagens desorganizadas e sem dados de calibração, e procedem à reconstrução do modelo 3D. O processo de reconstrução não exige que o utilizador tenha conhecimento na área de visão computacional, apenas tem que fornecer as imagens. Das ferramentas analisadas destaca-se o Bundler, apesar de não ter obtidos os melhores resultados, é uma ferramenta gratuita e de código aberto [5]. O código SfM do Bundler foi utilizado no projeto Photo Tourism apoiado pela Microsoft e pelo Laboratório Gráfico e de Imagem da Universidade de Washington [13, 14]. A nível de ferramentas gráficas, o VisualSFM é um projeto gratuito e interessante para uso pessoal ou académico mas não permite efetuar modificações. Este é um projeto de processamento mais rápido devido à exploração do paralelismo em vários processadores, para a deteção e correspondência de pontos de interesse (SiftGPU) e para o BA (*Multicore Bundle Adjustment*) [15, 16, 17, 18].

Nos últimos tempos, a visão tem sido utilizada para a mapeamento e navegação de sistemas autónomos. Na robótica móvel este é um problema conhecido como SLAM, ou SLAM Monocular quando se utiliza uma câmara como sensor principal. O SLAM monocular utiliza uma sequência de imagens para simultaneamente localizar e mapear o ambiente desconhecido, em tempo real [19, 20]. Em [1] é detalhado a técnica de SLAM Monocular em tempo real, para ambientes estruturados, usando apenas uma câmara e com base na informação visual estima trajetórias simples com seis graus de liberdade. A abordagem do problema SLAM tem algumas similaridades com o SfM. Em [21] realiza-se a estimação da posição da câmara e construção do mapa 3D a partir das correspondências entre as características das imagens, e a reconstrução da estrutura é aperfeiçoada através do *Bundle Adjustment* (BA).

O SfM em tempo real tem diversas aplicações que variam desde realidade aumentada e jogos de vídeo até ao cinema, e desde navegação de sistemas autónomos até aplicações na área da medicina. A fusão sensorial de sensores inerciais e GPS com o SfM pode fornecer um aumento na robustez da estimação da posição da câmara e na reconstrução do espaço. Em [22] apresentam um método SfM baseado na fusão de informação visual e inercial com um filtro de Kalman para a reconstrução precisa e aplicável no contexto cirúrgico. Em [23] apresentam uma abordagem que processa um *stream* em tempo real e o resultado é um modelo 3D com textura detalhada. A

velocidade de processamento é alcançada através da utilização do processador gráfico (GPU) e da informação proveniente dos sensores de GPS e inercial. Em [24] apresentam um abordagem que combina a procura de correspondências entres as imagens com a reconstrução da geometria, em vez de executar como passos separados. Esta abordagem aumenta a precisão e robustez de cada passo e se eficientemente implementada possibilita um desempenho de 5 a 10 fps num *hardware standard* com precisão relativamente alta. Esta implementação exige maior esforço computacional mas reduz significativamente o erro de reprojeção.

Quando o SfM é aplicado em enormes conjuntos de imagens, podem surgir dois problemas: elevado tempo de processamento e desvio na estimação devido à acumulação de erros. Em [25] é apresentada uma abordagem com integração da informação de GPS, para aumentar o desempenho do processo de reconstrução assim como evitar o desvio devido aos erros acumulados. Em [26] é utilizada a informação proveniente dos sensores GPS/IMU, para aumentar o desempenho do SfM e para aplicar um critério de seleção no processo de *matching*, baseado na informação da posição/orientação. Para a reconstrução 3D de cidades a partir de enormes conjuntos de imagens, em [27] demonstram que a incorporação da informação de GPS torna viável o processo de reconstrução e são utilizadas as *geo-tags* para o *matching* das imagens. Neste método, a informação de GPS é utilizada para dividir as imagens em sub-conjuntos, baseados na sua localização, para que possam ser eficientemente reconstruídos.

O projeto Geo-3d [28] consiste na criação de nuvens de pontos 3D densas a partir de imagens aéreas em conjunto com informação de IMU/GPS. A informação dos sensores é utilizada como inicialização dos parâmetros extrínsecos das câmaras para o *Bundle Adjustment*. Neste projeto é utilizada uma adaptação do SIFT para a GPU (SiftGPU), para aumentar consideravelmente o desempenho, nos processos de deteção e correspondência de pontos de interesse entre as imagens. O Geo-3d foi desenvolvido através da modificação do projeto Bundler. Estes projetos são gratuitos e o seu código está disponível, sendo estas duas vantagens, em comparação com os projetos descritos anteriormente, para iniciar a exploração do SfM em aplicações robóticas.

3
Fundamentos

A visão computacional é um processo que tem como objetivo determinar as propriedades geométricas, dimensões e posições dos objetos no espaço, a partir de imagens.

As câmaras são os sensores que permitem mapear o espaço 3D em imagens 2D. Este mapeamento pode ser representado por matrizes com propriedades específicas e que definem o modelo da câmara.

3.1 Modelo da Câmara

O modelo de câmara mais simples é o *pinhole* [29]. Este modelo considera o centro de projeção dos pontos no espaço num plano, onde a abertura da câmara é descrita por um ponto e não existe lente para focar a luz.

O centro de projeção é a origem do sistema de coordenadas Euclideano e considera-se o plano $Z = f$, que é denominado como plano da imagem ou plano focal.

Segundo o modelo *pinhole*, um ponto no espaço com coordenadas $\mathbf{X} = (X, Y, Z)^T$ é mapeado num ponto no plano da imagem. Na Figura 3.1 o ponto no plano da imagem, $\mathbf{x}$, corresponde ao ponto do plano pelo qual passa a linha que une o ponto $\mathbf{X}$ e o centro de projeção. Observando a Figura 3.1 e pela similaridade de triângulos ($x/f = X/Z$ e $y/f = Y/Z$) o ponto no espaço 3D $(X, Y, Z)^T$ é mapeado no ponto $(fX/Z, fY/Z, f)^T$ do plano da imagem.

O centro de projeção é denominado como o centro da câmara ou como centro

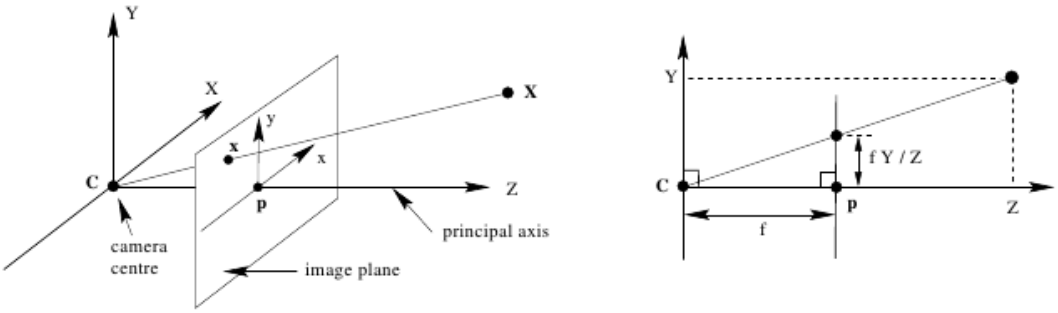

Figura 3.1: *Geometria do modelo pinhole [29]*

ótico. A linha perpendicular desde do centro da câmara, C, até ao plano da imagem é denominada como eixo principal ou raio principal da câmara, e o ponto onde este eixo intersecta o plano da imagem é designado de ponto principal, $\mathbf{p}$. O mapeamento do centro de projeção dos pontos do mundo para coordenadas na imagem, é definido pela Equação 3.1

$$\mathbf{x} = P\mathbf{X} \tag{3.1}$$

onde $\mathbf{x}$ é o ponto no plano da imagem, $\mathbf{X}$ o ponto 3D e P a matriz de projeção.

Em geral, os pontos no espaço são expressos no sistema de coordenadas do Mundo. Os sistemas de coordenadas da câmara e do Mundo estão relacionados por uma rotação e translação. Então a matriz de projeção é definida pela Equação 3.2.

$$P = K[R|t] \tag{3.2}$$

onde K é a matriz dos parâmetros intrínsecos da câmara, R (matriz 3×3 de rotação) e t (vetor de translação) correspondem aos parâmetros extrínsecos.

Normalmente os pontos no espaço são expressados num sistema de coordenadas, conhecido como sistema de coordenadas do Mundo. O sistema de coordenadas do Mundo e o sistema de coordenadas da câmara, estão relacionados por uma rotação e translação. Os parâmetros extrínsecos da câmara (R e t) transformam um ponto do referencial do Mundo para o referencial da câmara, aplicando uma rotação e translação, Equação 3.3.

$$\mathbf{X}_{cam} = R(\mathbf{X} - C) \tag{3.3}$$

onde $\mathbf{X}_{cam}$ representa o ponto do Mundo ($\mathbf{X}$) no referencial da câmara, e C representa as coordenadas do centro da câmara no referencial do Mundo.

O modelo *pinhole* assume que as coordenadas da imagem são Eucledianas e que as escalas em ambos os eixos são iguais. No caso dos sensores *Charge-Coupled Device* (CCD) das câmaras, existe a possibilidade de os *pixels* não serem quadrados.

Considerando que as coordenadas são medidas em *pixels* e que estes podem não ser quadrados, então é necessário introduzir um factor de escalar para cada eixo. Se o número de *pixels* por unidade de distância em coordenadas da imagem for mx e my para o eixo x e y, respetivamente, então a transformação de um ponto no referencial da câmara para coordenadas no plano da imagem é obtida pela matriz de calibração K, definida pela Equação 3.4.

$$K = \begin{bmatrix} \alpha_x & s & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.4}$$

onde $\alpha_x = fm_x$ e $\alpha_y = fm_y$ representam a distância focal da câmara em *pixels*. O ponto principal tem coordenadas (p_x, p_y) e, na realidade, a origem das coordenadas do plano da imagem não corresponde ao ponto principal, como pode ser observado na Figura 3.2. As coordenadas do ponto principal, em *pixels*, são representadas por

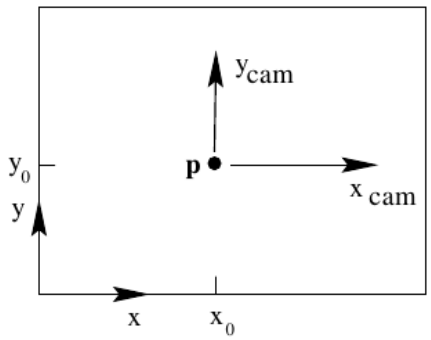

Figura 3.2: *Sistema de coordenadas do plano da imagem e da câmara [29]*

$x_0 = m_x p_x$ e $y_0 = m_y p_y$. O parâmetro s é o *skew*, apresentando em casos comuns, o valor zero. Se o *skew* for diferente de zero significa que os eixos x e y não são perpendiculares. Portanto, nesses casos, o modelo da câmara CCD tem 11 graus de liberdade, 5 da matriz K, 3 da matriz R e 3 do vetor t.

3.2 Geometria Epipolar

Quando duas câmaras capturam o mesmo espaço, cada uma com o seu centro de projeção, sendo estes não coincidentes, cada par de imagens capturado representa duas perspectivas diferentes do mesmo espaço estático. A geometria epipolar é a geometria projetiva intrínseca que estabelece a relação entre as duas perspetivas [29]. É independente da estrutura do espaço, e apenas depende dos parâmetros intrínsecos e extrínsecos (posição e rotação relativa) das câmaras.

A geometria epipolar entre duas perspetivas é essencialmente a geometria de intersecção dos planos de imagem com um conjunto de planos, tendo a *baseline* como eixo (a *baseline* é a linha que une os centros das câmaras). Esta geometria tem aplicabilidade na procura de pontos correspondentes em *stereo matching*.

Na Figura 3.3 são apresentadas duas projeções $\mathbf{x}$ e $\mathbf{x}'$ referentes ao ponto tridimensional $\mathbf{X}$. A Figura 3.3 (a) demonstra que tanto os pontos da imagem $\mathbf{x}$ e $\mathbf{x}'$, como respetivo centro das câmaras (C e C') e o ponto 3D $\mathbf{X}$ são coplanares. Os raios projetados para trás de $\mathbf{x}$ e $\mathbf{x}'$ intersectam no ponto $\mathbf{X}$, e estes raios são coplanares, Figura 3.3 (b). A *baseline* interseta cada plano da imagem nos epipolos e e e'. O

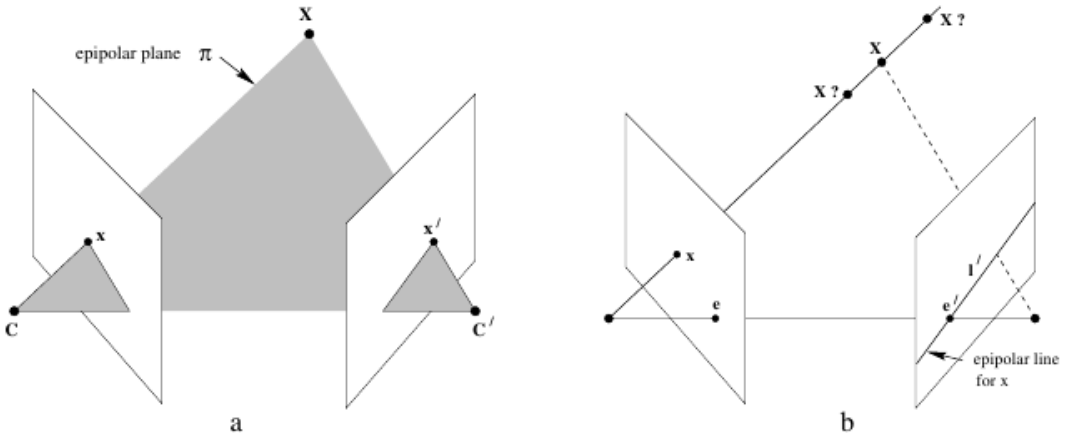

Figura 3.3: *Geometria para correspondência de pontos [29]*

epipolo é o ponto de interseção entre a *baseline* e o plano da imagem. De forma equivalente, o epipolo é a imagem numa perspetiva do centro da câmara da outra perspetiva. Cada plano π contendo a *baseline* é um plano epipolar, que intersecta os planos de imagem nas linhas correspondentes l e l', Figura 3.4 (a). Como a posição 3D do ponto $\mathbf{X}$ varia, os planos epipolares rodam em torno da *baseline*, Figura 3.4 (b). A linha epipolar é a interseção do plano epipolar com o plano da imagem. Todas as linhas epipolares intersetam o epipolo. O plano epipolar interseta o plano da imagem esquerda e direita nas linhas epipolares, e define a correspondência entre as linhas.

Supondo que apenas é conhecido o ponto $\mathbf{x}$, o objetivo passa por determinar o ponto $\mathbf{x}'$. Sabendo que o plano π é determinado pela *baseline* e pelo raio definido por $\mathbf{x}$, então o raio correspondente ao ponto $\mathbf{x}'$ está contido no plano π e o ponto $\mathbf{x}'$ está contido na linha l'. A linha l' corresponde à interseção no plano π com o plano da segunda imagem. Esta linha corresponde à imagem do raio projetado para trás do ponto $\mathbf{x}$, no plano a segunda imagem. Resumindo, l' é a linha epipolar correspondente ao ponto $\mathbf{x}$. A geometria epipolar estabelece a relação de correspondência entre um

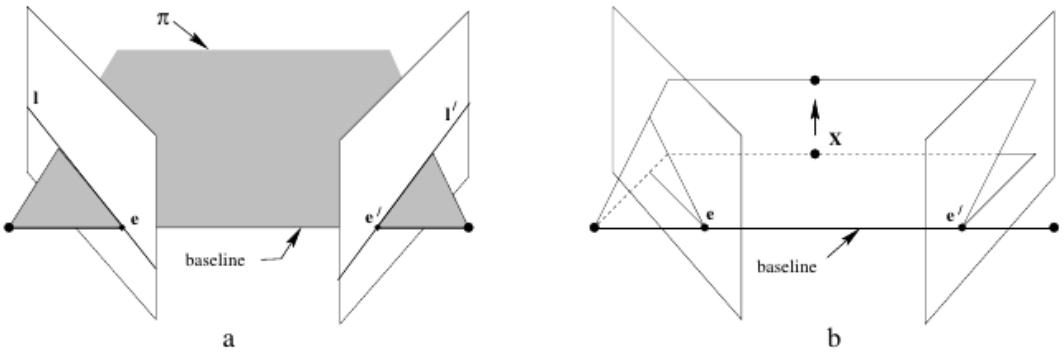

Figura 3.4: *Geometria epipolar [29]*

ponto de uma imagem e uma linha em outra imagem. A vantagem desta restrição está na redução da procura do ponto correspondente apenas a uma linha, em vez de procurar em todo o plano de imagem.

3.3 Matriz Fundamental

A geometria epipolar pode ser representada por uma matriz de dimensão 3×3, denominada como matriz fundamental F [29]. Se um ponto no espaço tridimensional $\mathbf{X}$ é observado na primeira imagem como $\mathbf{x}$ e na segunda como $\mathbf{x}'$, então estes pontos são relacionados pela Equação 3.5.

$$\mathbf{x}'^T F \mathbf{x} = 0 \tag{3.5}$$

A matriz fundamental também é independente da estrutura do espaço. Contudo, pode ser determinada a partir das correspondências de pontos, sem requerer conhecimento sobre os parâmetros intrínsecos e extrínsecos de cada câmara.

A matriz F é definida pela Equação 3.5, para qualquer par de pontos correspondentes, $\mathbf{x}$ e $\mathbf{x}'$, entre duas imagens. Sabendo pontos correspondentes suficientes, $\mathbf{x}_i \rightarrow \mathbf{x}'_i$, (no mínimo 7), com a Equação 3.5 é possível determinar a matriz F. Cada ponto correspondente $\mathbf{x} = (x, y, 1)^T$ e $\mathbf{x}' = (x', y', 1)^T$ permite criar uma equação linear para as incógnitas de F. A Equação 3.6 corresponde aos pontos conhecidos $\mathbf{x}$ e $\mathbf{x}'$ para determinar F.

$$x'xf_{11} + x'yf_{12} + x'f_{13} + y'xf_{21} + y'yf_{22} + y'f_{23} + xf_{31} + yf_{32} + f_{33} = 0 \tag{3.6}$$

Considerando f como um vetor de comprimento 9 e com as incógnitas de F, então para um conjunto de n pontos correspondentes, obtém-se um conjunto de equações

lineares, Equação 3.7.

$$Af = \begin{bmatrix} x'_1x_1 & x'_1y_1 & x'_1 & y'_1x_1 & y'_1y_1 & y'_1 & x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \\ x'_nx_n & x'_ny_n & x'_n & y'_nx_n & y'_ny_n & y'_n & x_n & y_n & 1 \end{bmatrix} f = 0 \qquad (3.7)$$

Este é um conjunto de equações homogéneas, sendo que f pode ser apenas determinado a menos de um fator de escala. Para existir uma solução, a matriz A deve ter no máximo *rank* 8, e se o *rank* for exactamente 8 então a solução é única (a menos de um fator de escala) e pode ser determinada por métodos lineares.

3.4 Algoritmo 8-point

O algoritmo *8-point* é o método mais simples para calcular a matriz fundamental, envolvendo a construção e solução de um conjunto linear de equações [29]. O ponto chave para o sucesso com o algoritmo 8-point, é o cuidado adequado de normalizar os dados antes de construir as equações a resolver. Uma simples transformação (translação e escala) dos pontos da imagem, antes de formular as equações lineares, permite uma melhoria significativa no condicionamento do problema e, consequentemente na estabilidade do resultado. Esta transformação introduz uma complexidade insignificante ao algoritmo. A normalização referida consiste numa translação e escala de cada imagem, para que o centroíde dos pontos de referência seja na origem das coordenadas, e a distância média dos pontos a partir da origem seja igual a $\sqrt{2}$.

Algoritmo: 8-point

Objetivo
Sabendo $n \geq 8$ pontos correspondentes entre imagens $\{\mathbf{x}_i \leftrightarrow \mathbf{x}'_i\}$, determinar a matriz fundamental F tal que ${\mathbf{x}'_i}^T F \mathbf{x}_i = 0$.

Algoritmo
(1) Normalização: transformação das coordenadas da imagem de acordo com $\hat{\mathbf{x}}_i = T\mathbf{x}_i$ e $\hat{\mathbf{x}}'_i = T\mathbf{x}'_i$, onde T e T' são as transformações de translação e escala

(2) Determinar a matriz fundamental $\hat{F}'$ correspondente aos pontos $\hat{\mathbf{x}}_i \leftrightarrow \hat{\mathbf{x}}'_i$

(a) Solução linear: Determinar $\hat{F}$ a partir do vetor singular correspondente ao menor valor singular de $\hat{A}$, onde $\hat{A}$ é composto a partir das correspondências entre $\hat{\mathbf{x}}_i \leftrightarrow \hat{\mathbf{x}}'_i$.

(b) *Constraint enforcement*: Substitui $\hat{F}$ por $\hat{F}'$ tal que o $det(\hat{F}') = 0$, usando o método de *Singular Value Decomposition* (SVD).

(3) *Denormalization*: Define $F = T'^T \hat{F}' T$. A matriz F é a matriz fundamental correspondente aos dados originais $\mathbf{x}_i \leftrightarrow \mathbf{x}'_i$

3.5 RANSAC

O *RANdom Sample Consensus* (RANSAC) é um algoritmo de estimação robusto, que consiste num método iterativo para estimar os parâmetros de um modelo matemático [30, 29]. Este algoritmo permite estimar os parâmetros de um modelo com dados contaminados por grandes quantidades de *ouliers*. A sua abordagem assenta na definição de três limites: t, T e N.

Algoritmo: RANSAC

Objetivo
Ajuste robusto de um modelo, para um conjunto de dados S que contém *outliers*

Algoritmo
(1) Seleciona aleatoriamente uma amostra de pontos s, a partir de S, e cria uma instância para o modelo a partir deste subconjunto s.

(2) Determina o conjunto de pontos S_i que estão dentro da distância de limite t do modelo. O conjunto S_i é o conjunto consenso da amostra e define os *inliers* de S.

(3) Se o tamanho de S_i (número de *inliers*) é maior que o limite T, re-estima o modelo usando todos os pontos em S_i e termina.

(4) Se o tamanho de S_i é menor que T, seleciona um novo subconjunto e repete os passos anteriores.

(5) Após N iterações, o maior conjunto consenso S_i é selecionado, e o modelo é re-estimado usando todos os pontos do conjunto S_i.

3.5.1 Limite t

O t é um limite de distância escolhido, de tal forma que com uma probabilidade α, o ponto é um *inlier*. Na prática o limite de distância é escolhido empiricamente. No entanto, se o erro de medida for Gaussiano, com média zero e desvio padrão σ, então o valor para t pode ser calculado.

Normalmente o valor de α é definido como 0.95, por isso existe 95% de probabilidade de um ponto ser um inlier e 5% de rejeição. Na Tabela 3.1 apresentam-se os valores para t para modelos de interesse para este documento e considerando $\alpha = 0.95$.

Tabela 3.1: *Limite de distância t para uma probabilidade de $\alpha = 0.95$ de que um ponto é um inlier*

m	**Modelo**	t^2
1	linha, matriz fundamental	3.84 σ^2
2	homografia, matriz da câmara	5.99 σ^2
3	tensor trifocal	7.81 σ^2
m é o número de graus de liberdade		

3.5.2 Limite N

Em geral, é computacionalmente inviável e desnecessário tentar todas as amostras possíveis. Em vez disso, é escolhido um número de amostras elevado N, para garantir com uma probabilidade, p, que pelo menos uma da amostras aleatórias de pontos s, não contém *outliers*.
Normalmente p é definido como 0.99. Considerando w como a probabilidade que um ponto selecionado seja um *inlier*, $\epsilon = 1 - w$ a probabilidade de ser um *outlier*, e $(1 - w^s)^N = 1 - p$, então pode-se determinar o número de amostras pela Equação 3.8.

$$N = \frac{\log(1-p)}{\log(1-(1-\epsilon)^s)} \tag{3.8}$$

A Tabela 3.2 apresenta exemplos de N para $p = 0.99$ e para um determinado valor de s e ϵ.

Tabela 3.2: *O número de amostras necessárias N para garantir, com probabilidade $p = 0.99$, que pelo menos uma amostra não tem outliers para um determinado conjunto de amostras, s, e proporção de outliers, ϵ*

[1]	**Proporção de *outliers* ϵ**						
s	5%	10%	20%	25%	30%	40%	50%
2	2	3	5	6	7	11	17
3	3	4	7	9	11	19	35
4	3	5	9	13	17	34	72
5	4	6	12	17	26	57	146
6	4	7	16	24	37	97	293
7	4	8	20	33	54	163	588
8	5	9	26	44	78	272	1177

[1] Dimensão da amostra

3.5.3 Limite T

Os *inliers* constituem o conjunto consenso. A regra geral consiste em verificar se o tamanho do conjunto de consenso é semelhante ao número de *inliers*, considerados no conjunto de dados, tendo em conta os *outliers* assumidos, ou seja, para n pontos $T = (1 - \epsilon)n$.

É frequente acontecer que o ϵ, a fração de dados que consistem em *outliers*, é desconhecido. Nestes casos, o algoritmo é inicializado, usando o pior caso de estimação de ϵ. Esta estimação pode ser atualizada conforme se vão encontrando maiores conjuntos consistentes. Por exemplo, se o pior caso é $\epsilon = 0.5$ e o conjunto consenso com 80% de pontos encontrados como *inliers*, então o valor de ϵ é atualizado para 0.2.

Algoritmo: Abordagem adaptativa

- $N = 1$, $sample\_count = 0$
- Enquanto $N > sample\_count$ Repete
  - Escolhe uma amostra e conta o número de *inliers*
  - Define ϵ = 1- (número de *inliers*)/(número total de pontos)
  - Define N a partir de ϵ e equação acima com $p = 0.99$
  - Incrementa o $sample\_count$ com mais uma unidade
- Termina

3.6 DLT

O *Direct Linear Transformation* (DLT) [29] é um algoritmo que permite determinar uma matriz de homografia, a partir de um conjunto de pontos 2D correspondentes entre dois planos.

A matriz de homografia consiste numa transformação projetiva que mapeia pontos de um plano para outro plano. Este mapeamento linear de pontos pode ser descrito pela equação $\mathbf{x}'_i = H\mathbf{x}_i$, onde H é a matriz de homografia que define o mapeamento dos pontos correspondentes $\mathbf{x}_i \leftrightarrow \mathbf{x}'_i$ entre dois planos.

Sabendo um conjunto de pontos $\mathbf{x}_i = (x_i, y_i, w_i)^T$ sobre um plano π e outro conjunto de pontos correspondentes $\mathbf{x}'_i = (x'_i, y'_i, w'_i)^T$ num plano π', pretende-se determinar a transformação projetiva H que mapeia cada ponto $\mathbf{x}_i$ para $\mathbf{x}'_i$. Este

mapeamento pode ser descrito pela Equação 3.9.

$$\begin{bmatrix} x'_i \\ y'_i \\ w'_i \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ w_i \end{bmatrix} \tag{3.9}$$

Na Equação 3.9, os pontos 2D são expressos em coordenadas homogéneas. Embora $\mathbf{x}'_i$ e $H\mathbf{x}_i$ representem o mesmo ponto no plano 2D, analisando $\mathbf{x}'_i$ e $H\mathbf{x}_i$ como vetores 3D, eles não são iguais, pois, embora tenham a mesma direção, podem ter uma magnitude diferente para um fator de escala homogêneo $\lambda \neq 0$. Então a Equação 3.9 pode ser reescrita como a Equação 3.10.

$$\lambda \mathbf{x}'_i = H\mathbf{x}_i \tag{3.10}$$

Para eliminar o fator de escala, a Equação 3.10 pode ser expressa em termos de produto vetorial, como demonstrado na Equação 3.11.

$$\mathbf{x}'_i \times (\lambda \mathbf{x}'_i) = \mathbf{x}'_i \times H\mathbf{x}_i \tag{3.11}$$

que pode ser reescrita como a Equação 3.12.

$$\mathbf{x}'_i \times H\mathbf{x}_i = 0 \tag{3.12}$$

A equação 3.12 permite obter um solução linear para H. Definindo a j-ésima linha da matriz H como h^{jT}, o produto $H\mathbf{x}_i$ pode ser descrito pela Equação 3.13.

$$H\mathbf{x}_i = \begin{bmatrix} h^{1T}\mathbf{x}_i \\ h^{2T}\mathbf{x}_i \\ h^{3T}\mathbf{x}_i \end{bmatrix} \tag{3.13}$$

Utilizando a Equação 3.13, o produto vetorial da Equação 3.12 pode ser reformulado como

$$\mathbf{x}'_i \times H\mathbf{x}_i = \begin{bmatrix} y'_i h^{3T}\mathbf{x}_i - w'_i h^{2T}\mathbf{x}_i \\ w'_i h^{1T}\mathbf{x}_i - x'_i h^{3T}\mathbf{x}_i \\ x'_i h^{2T}\mathbf{x}_i - y'_i h^{1T}\mathbf{x}_i \end{bmatrix} = 0 \tag{3.14}$$

Sendo $h^{jT}\mathbf{x}_i = \mathbf{x}_i^T h^j$, a Equação 3.14 pode ser reescrita como

$$\begin{bmatrix} 0^T & -w'_i\mathbf{x}_i^T & y'_i\mathbf{x}_i^T \\ w'_i\mathbf{x}_i^T & 0^T & -x'_i\mathbf{x}_i^T \\ -y'_i\mathbf{x}_i^T & x'_i\mathbf{x}_i^T & 0^T \end{bmatrix} \begin{bmatrix} h^1 \\ h^2 \\ h^3 \end{bmatrix} = A_i h = 0 \tag{3.15}$$

onde A_i é uma matriz 3×9, tal que as suas entradas são formadas pelas coordenadas de um par de pontos correspondentes conhecidos e h é um vetor coluna contendo as entradas não conhecidas de H, ou seja, $h = (h_{11}, h_{12}, h_{13}, h_{21}, h_{22}, h_{23}, h_{31}, h_{32}, h_{33})^T$.

Das três linhas da matriz A_i na Equação 3.15, apenas duas são linearmente independentes. A terceira linha é obtida pela soma de $-x'_i$ vezes a primeira linha e $-y'_i$ vezes a segunda. Segundo [29], pode-se assumir que $w'_i = 1$ e a terceira linha pode ser omitida para a solução de H. Assim a Equação 3.15 pode ser reescrita como

$$\begin{bmatrix} 0^T & -w'_i\mathbf{x}_i^T & y'_i\mathbf{x}_i^T \\ w'_i\mathbf{x}_i^T & 0^T & -x'_i\mathbf{x}_i^T \end{bmatrix} \begin{bmatrix} h^1 \\ h^2 \\ h^3 \end{bmatrix} = A_i h = 0 \tag{3.16}$$

onde A_i é uma matriz 2×9.

A matriz H possui 9 entradas e 8 graus de liberdade. Baseado nos graus de liberdade de H, pode-se estabelecer um limite inferior n, que determina a quantidade de pontos correspondentes $\mathbf{x}_i \leftrightarrow \mathbf{x}'_i$ necessários para calcular a transformação projetiva H.

Cada par de pontos correspondentes tem dois graus de liberdade (x e y), pois as coordenadas de um ponto $\mathbf{x}_i$ são determinadas por dois elementos x e y (o fator de escala homogéneo é arbitrário) e os dois graus de liberdade do ponto $\mathbf{x}i$ devem corresponder ao ponto mapeado $H\mathbf{x}i$. Assim, pelo menos quatro pontos correspondentes são necessários ($n = 4$), sendo que, para cada par de pontos correspondentes $\mathbf{x}_i \leftrightarrow \mathbf{x}'_i$, pode-se calcular uma matriz A_i de dimensão 2×9. Então, a matriz A de dimensão $2n \times 9$, composta por cada matriz A_i, estabelece o sistema linear $Ah = 0$, cuja a solução resolve H.

Quando o número de pontos correspondentes é maior do que 4, o sistema é sobredeterminado, ou seja, o número de equações é maior que o número de incógnitas e duas situações devem ser consideradas. Na primeira, assume-se que a posição dos pontos é exata e o *rank* da matriz A é 8, então, a solução também é determinada pelo espaço nulo de A. Na segunda hipótese, a posição dos pontos não é exata (geralmente devido ao ruído) e a solução exata para o sistema $Ah = 0$ é inexistente e por isso deve-se calcular uma solução aproximada.

3.7 Deteção de Pontos de Interesse

O reconhecimento de objetos em imagens reais requer características que não sejam afetadas pela proximidade desorganizada ou pela oclusão parcial. As características devem ser no mínimo invariantes à iluminação e às transformações projetivas

3D. Por outro lado, as características devem ser suficientemente distintas, para permitir identificar um objeto específico num espaço com diversos objetos. A dificuldade do problema de reconhecimento de objetos, deve-se maioritariamente à falta de sucesso em encontrar as características numa imagem. No processo SfM, a extração de pontos de interesse é um passo crucial para a reconstrução tridimensional do espaço.

3.7.1 SIFT

O *Scale Invariante Feature Transform* (SIFT) [31] transforma uma imagem num enorme grupo de vetores locais de características (*features*), em que cada um é invariante à translação, escala, rotação da imagem e parcialmente invariante às alterações na luminosidade e a projeções tridimensionais. Os vetores resultantes são designados de pontos SIFT. Normalmente são gerados 1000 pontos SIFT por imagem, em menos de 1 segundo.

Os pontos SIFT obtidos de uma imagem são utilizados num metodologia do tipo vizinho mais próximo, de modo a identificar os objetos candidatos. Quando três ou mais pontos sejam concordantes com os parâmetros do modelo, existe uma forte indicação que o objeto pretendido foi encontrado. Como deverão existir dezenas de pontos SIFT na imagem de um objeto comum, é possível ter um nível considerável de oclusão na imagem, e mesmo assim obter altos níveis de confiança.

O primeiro passo no SIFT consiste em desfocar a imagem (*Gaussian Blur*) para reduzir o detalhe e evitar falsas *features*, destacando-se apenas a forma do objeto. Além de desfocar a imagem original, esta é reduzida na sua dimensão para metade. O processo de redução da imagem é designado como oitava. O criador do SIFT sugere que quatro oitavas e cinco níveis de desfoque são o ideal para o algoritmo. O processo de desfoque é realizado pela convolução entre uma função Gaussiana e uma imagem, Equação 3.17.

$$L(x, y, \delta) = G(x, y, \delta) * I(x, y) \tag{3.17}$$

onde

$$G(x, y, \delta) = \frac{1}{2\pi\delta^2} e^{-\frac{x^2+y^2}{2\delta^2}} \tag{3.18}$$

onde L é a imagem desfocada, G é a função Gaussiana, I a imagem, x e y as coordenadas da localização do *pixel* e δ o parâmetro de escala.

Numa imagem, as fronteiras e os cantos são locais propensos a encontrar pontos SIFT. Para extrair as fronteiras e os cantos, desfoca-se a imagem e calculam-se as derivadas de segunda ordem (Laplaciano). A derivada de segunda ordem é extremamente sensível ao ruído, e embora o desfoque reduza o ruído, o cálculo de todas

as derivadas de segunda ordem exige esforço computacional. Por isso, para detectar eficazmente a localização dos pontos chave no espaço de escala, Lowe [31] propõe a utilização da diferença de duas escalas separadas por um fator de multiplicação k, obtendo-se assim uma função denominada *Difference of Gaussian* (DoG), a convolver com a imagem, Equação 3.20.

$$D(x, y, \delta) = (G(x, y, k\delta) - G(x, y, \delta)) * I(x, y) \tag{3.19}$$

$$= L(x, y, k\delta) - L(x, y, \delta) \tag{3.20}$$

O DoG é uma adaptação do Laplaciano de Gaussiano, e consiste na subtração entre imagens consecutivas, Figura 3.5. As imagens obtidas através da DoG são aproxi-

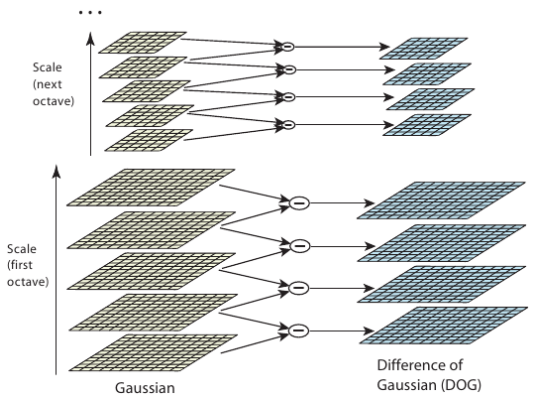

Figura 3.5: *Diferença dos Gaussianos [31]*

madamente equivalentes às que seriam obtidas usando o Laplaciano de Gaussiano, exigindo menos tempo computacional e tornando o processo que seria intensivo num processo mais leve, rápido e eficiente.

O método para encontrar os pontos SIFT, resume-se em dois passos: encontrar os pontos mínimos e máximos nas imagens obtidas pela DoG e encontrar o *subpixel* mínimo e máximo.

O primeiro passo é simples, consiste em percorrer todos os *pixels* e verificar a sua vizinhança. A vizinhança engloba os *pixels* circundantes das imagens anterior e seguinte, Figura 3.6. O mínimo e máximo da DoG é obtido pela comparação de um *pixel* (marcado com X) com os 26 vizinhos nas regiões 3×3 das escalas corrente e adjacentes (marcadas com círculos verdes). Os pontos selecionados são considerados apenas uma aproximação dos máximos e mínimos, isto porque estes não se encontram exatamente num *pixel*, situam-se algures entre cada pixel. Neste caso, é necessário encontrar matematicamente o *subpixel*. O *subpixel* é determinado pela expansão de Taylor da imagem, em torno do ponto SIFT aproximado.

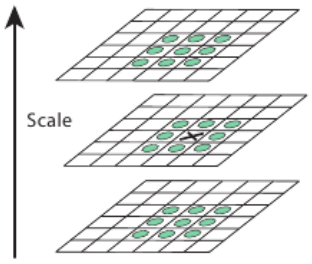

Figura 3.6: *Pixels mínimo e máximo [31]*

Sabendo os pontos SIFT, procede-se a uma eliminação de pontos que se encontram ao longo de uma extremidade ou que não possuam contraste suficiente. Em ambos os casos, estes pontos não são úteis para serem utilizados como *features*, e por isso são eliminados. O método de remoção de *features* que estão nas extremidades, é equivalente ao *Harris Corner Detector* [32].

A invariância na rotação é obtida através da atribuição da orientação a cada ponto SIFT encontrado na imagem. A ideia consiste em perceber a direção dos gradientes e magnitudes em torno de cada ponto. Desta forma, são definidas as orientações mais proeminentes na região em estudo e, seguidamente aos pontos SIFT correspondentes. A magnitude e orientação são calculadas para todos os *pixels* em torno de um ponto chave SIFT, usando as Equações 3.21 e 3.22, respetivamente.

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2} \quad (3.21)$$

$$\theta(x,y) = \tan^{-1}((L(x,y+1) - L(x,y-1))/(L(x+1,y) - L(x-1,y))) \quad (3.22)$$

Posteriormente é criado um histograma, onde os 360 graus de orientação são divididos em 36 intervalos com 10 graus. Este procedimento é aplicado para todos os *pixels* em torno do ponto SIFT, e a orientação é determinada pelo ponto mais alto do histograma.

No final, é necessário criar uma identificação única para cada ponto SIFT. Sendo assim, é necessário gerar uma janela 16×16 em torno do ponto SIFT, sendo que cada janela é subdividida em dezasseis janelas de 4×4. Dentro de cada janela 4×4 é calculado o gradiente de magnitudes e orientações. Estas orientações são colocadas num histograma de 8 intervalos. Os resultados das orientações são normalizados, e colocam-se em forma de vetores. Este conjunto de vetores identificam o ponto SIFT. Para tornar a identificação independente da rotação, a rotação dos pontos é subtraída a todas as orientações, e assim os gradientes de orientação ficam relativos à orientação do ponto SIFT.

3.7.2 SiftGPU

O SiftGPU é uma implementação do SIFT para a *Graphics Processing Unit* (GPU) [18]. Os passos enunciados podem utilizar a GPU para processar os *pixels* paralelamente:

- conversão de cor para intensidade, e aumento ou diminuição da amostragem (resolução) das imagens de entrada;
- construção de pirâmides Gaussianas (intensidade, gradiente, DoG);
- deteção de pontos SIFT (localização do *sub-pixel* e da sub-escala);
- criação de listas compactas das *features* com uma redução do histograma no GPU;
- cálculo das orientações e descritores dos pontos.

A implementação do SIFT para GPU pode alcançar uma grande melhoria na velocidade de processamento, em relação ao CPU, através da utilização das vantagens do processamento das placas gráficas modernas.

A biblioteca SiftGPU também inclui a funcionalidade de procurar pontos SIFT correspondentes entre as imagens (*matches*), através da GPU (SiftMatchGPU).

Nem todo o tipo de processamento é mais rápido na GPU, por isso, esta biblioteca procura encontrar a melhor opção para cada passo. A execução do SiftGPU exige bastante memória na placa gráfica para armazenar informação temporária de forma a garantir o processamento eficientemente das novas imagens.

O SiftGPU disponibiliza implementações para *Compute Unified Device Architecture* (CUDA) e *OpenGL Shading Language* (GLSL). O CUDA é uma plataforma de computação paralela e um modelo de programação desenvolvida pela Nvidia, exclusivamente, para as suas GPUs. O GLSL é uma extensão da linguagem de programação OpenGL e possibilita aos programadores mais controlo direto da placa gráfica. O GLSL funciona em placas gráficas ATI e Nvidia.

3.8 Reconstrução 3D a partir de duas imagens

A reconstrução 3D do espaço pode ser realizada através de duas imagens de perspetivas diferentes, desde que exista um número suficiente de pontos correspondentes (*matches*). Estes *matches* provém de um conjunto de pontos 3D, com posição desconhecida e a posição e orientação das câmaras também desconhecida. Então

o processo de reconstrução consiste em encontrar a matrizes de projeção (P) das câmaras assim como a posição dos pontos 3D, Equações 3.23.

$$\mathbf{x}_i = P\mathbf{X}_i, \tag{3.23}$$

$$\mathbf{x}'_i = P'\mathbf{X}_i$$

onde i é o índice do ponto, $\mathbf{x}_i$ ponto no plano da imagem de uma câmara e $\mathbf{x}'_i$ o ponto correspondente na outra imagem. As matrizes de projeção das respetivas câmaras são definidas como P e P'. Se existir um número suficiente de *matches* entre as imagens, torna-se possível determinar a matriz fundamental, e consequentemente reconstruir o espaço a menos de uma ambiguidade projetiva. A ambiguidade da reconstrução pode ser reduzida, se for fornecida informação adicional sobre as câmaras ou espaço.

O método de reconstrução tridimensional, para duas imagens, pode ser resumido da seguinte forma:

1. Determinação da matriz fundamental a partir dos pontos correspondentes.

2. Determinação das matrizes das câmaras a partir da matriz fundamental.

3. Para cada par de pontos correspondentes, determinação do ponto 3D que originou os pontos projetados no plano da imagem.

Este método pode ter diversas variantes. Por exemplo, no caso de câmaras calibradas, é determinada a matriz essencial em vez da matriz fundamental. Nesses casos, a informação da calibração pode ser utilizada para melhorar a reconstrução.

3.8.1 Matriz de Projeção

A determinação das matrizes de projeção é realizada com base num conjunto de pontos correspondentes $\mathbf{x}_i \rightarrow \mathbf{x}'_i$ entre duas imagens, permitindo que a matriz fundamental F satisfaça a condição ${\mathbf{x}_i}'^T F\mathbf{x}_i = 0$. Considerando que os pontos dos planos de imagem $\mathbf{x}_i$ e $\mathbf{x}'_i$ são conhecidos, então torna-se possível construir uma equação linear para determinar os parâmetros desconhecidos da matriz F. Cada ponto correspondente gera uma equação linear, e por isso são necessários no mínimo 8 pontos correspondentes para resolver linearmente as equações dos parâmetros de F. Com mais de 8 equações, a solução é encontrada com o método dos mínimos quadrados. Este método é uma abordagem comum em sistemas sobre-determinados (maior número de equações do que incógnitas), onde a solução é encontrada através da minimização da soma dos quadrados dos erros de cada equação.

As matrizes das câmaras P e P' correspondentes à matriz fundamental, são facilmente determinadas aplicando a Equação 3.24.

$$P = [I|0] \qquad P' = [[e'] \times F|e'] \tag{3.24}$$

3.8.2 Triangulação

Sabendo as matrizes das câmaras P e P', e os pontos $\mathbf{x}$ e $\mathbf{x}'$ nas duas imagens que satisfazem a restrição epipolar $\mathbf{x}'^T F\mathbf{x} = 0$, torna-se possível determinar o ponto 3D $\mathbf{X}$ através da triangulação. Considerando o que foi explicado sobre a geometria epipolar e analisando a Figura 3.7, o ponto 3D pode ser determinado através da intersecção dos raios projetados para trás dos pontos $\mathbf{x}$ e $\mathbf{x}'$.

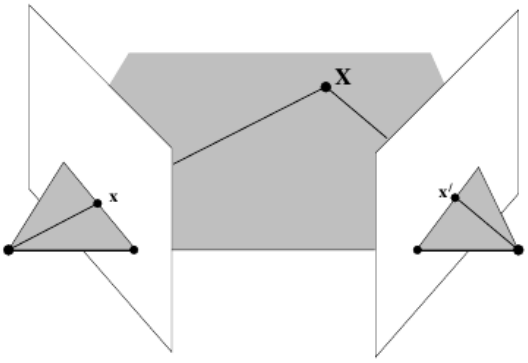

Figura 3.7: *Triangulação*

Os únicos pontos 3D que não podem ser determinados, são os pontos que se encontram na *baseline* entre as duas câmaras. Neste caso, os raios projetados para trás são colineares (iguais à *baseline*) e intersectam-se ao longo de todo o comprimento, por isso o ponto 3D não pode ser unicamente determinado.

Em alguns casos, as coordenadas dos pontos correspondentes $\mathbf{x}$ e $\mathbf{x}'$ podem conter algum tipo de ruído, por exemplo, gerado por erros de estimação, fazendo com que o ponto de intersecção $\mathbf{X}$, relativo aos raios que projetam $\mathbf{x}$ e $\mathbf{x}'$, não possa ser estabelecido, ou seja, $\mathbf{X}$ pode não satisfazer ambas as equações $\mathbf{x} = P\mathbf{X}$ e $\mathbf{x}' = P\mathbf{X}$. Nestas situações, uma solução aproximada para $\mathbf{X}$ deve ser determinada por estimação.

4

Structure from Motion

O *Strucuture from Motion* (SfM) é um processo de estimação simultânea da geometria 3D do espaço e do movimento (posição das câmaras) através de uma sequência de imagens. Considerando a existência de diversas abordagens para algoritmos de SfM, nesta dissertação, será descrita a abordagem do algoritmo implementado no projeto Bundler. O algoritmo deste projeto foi utilizado no projeto Photo Tourism. Devido à inexistência de publicações acerca do Bundler, o seu algoritmo será descrito através das publicações referentes ao projeto Photo Tourism.

4.1 Projeto Photo Tourism

O Photo Tourism é um sistema para navegar e explorar enormes conjuntos desordenados de imagens de um espaço, usando uma interface 3D [13, 14]. Este projecto é baseado no trabalho sobre SfM de Hartley e Zisserman [29] para obter parâmetros da câmara, estimação da posição e geometria esparsa 3D através de sequências de imagens.
A abordagem do Photo Tourism é semelhante à abordagem de Brown and Lowe [33], com várias modificações de forma a aumentar a robustez para uma grande diversidade de conjuntos de imagens. A descrição da abordagem de Brown and Lowe é seguinte:

Algoritmo: Reconstrução 3D

Entrada: n imagens desordenadas

(1) Extrair pontos SIFT de todas as n imagens

(2) Procurar k vizinhos próximos para cada ponto SIFT usando uma *k-d tree*

(3) Para cada imagem:
(a) Selecionar m imagens candidatas correspondentes (com o máximo número de *matches* SIFT com esta imagem)

(b) Procurar *matches* SIFT geometricamente consistentes usando o RANSAC, para calcular a matriz fundamental entre pares de imagens

(c) (Trabalho futuro) Verificar as correspondências entre as imagens usando um modelo probabilístico

(4) Procurar componentes ligados dos *matches* de imagens (lista dos *matches* SIFT ao longo de múltiplas imagens, cada componente ligado corresponde a um ponto 3D)

(5) Para cada componente ligado
(a) Executar o *Sparse Bundle Adjustment* para determinar a rotação θ_1, θ_2 e θ_3, translação t_1, t_2 e t_3 e a distancia focal f de todas as câmaras, e os pontos 3D

(b) (Trabalho futuro) Estimar a profundidade densa, triangulação, mapa de textura, etc.

Saída: Modelo 3D

Em relação à abordagem anterior, das modificações implementadas no algoritmo do projeto *Photo Tourism*, destacam-se as seguintes:

- Inicialização das novas câmaras usando a estimação da posição, para ajudar a evitar os mínimos locais.

- Seleção do par inicial de imagens com um método diferente.
- Verificação do bom condicionamento dos pontos reconstruídos, antes de adicionar ao espaço.
- Utilização da informação da distância focal, extraída das *Exchangeable Image File Format* (EXIF) *tags* das imagens, para inicialização.

Os projectos no qual se basearam para desenvolver o Photo Tourism, foram testados em conjuntos de imagens mais simples, enquanto que este foi o primeiro projecto SfM a ser aplicado com sucesso, em diferentes tipos de imagens reais provenientes do Google e Flickr. Os conjuntos de imagens usados contêm fotos provenientes de centenas de diferentes câmaras, níveis de zoom, resoluções, diferentes momentos do dia e estações, luminosidade, tempo e bastantes oclusões.

As ferramentas de visualização e navegação do Photo Tourism, requerem informação precisa acerca da localização relativa, orientação, e parâmetros intrínsecos tais como a distância focal para cada imagem, assim como a geometria esparsa 3D. Alguma desta informação pode ser fornecida por dispositivos GPS ou sensores inerciais, mas a grande maioria das fotografias existentes na Internet não contêm este tipo de informação. Muitas câmaras digitais, incluem a distância focal e outras informações nas EXIF *tags* das imagens. Estes dados são úteis para a inicialização, mas por vezes não têm precisão. Este sistema, não confia na câmara ou em outro dispositivo para fornecer a localização, orientação ou geometria. Em vez disso, esta informação é obtida através das imagens usando técnicas da visão computacional. Primeiro, detetam-se pontos de interesse em cada imagem, depois estabelece-se a correspondência dos pontos de interesse entre pares de imagens e, finalmente executa-se um método SfM iterativo e robusto para obter os parâmetros das câmaras.

4.1.1 Deteção e correspondência de pontos SIFT entre as imagens

O primeiro passo consiste em encontrar pontos de interesse usando o SIFT. A utilização do SIFT deve-se ao facto de ser invariante às transformações das imagens. De seguida, estabelece-se a correspondência entre os pontos extraídos para cada par de imagens, usando uma *kd-tree* da biblioteca *Approximate Nearest Neighbors* (ANN) [34]. Para procurar pontos correspondentes entre as imagens I e J, é criada uma *kd-tree* dos descritores dos pontos SIFT em J, depois, para cada ponto SIFT em I, procura-se o vizinho mais próximo em J usando a *kd-tree*. Para tornar mais eficiente, utiliza-se o algoritmo de procura com prioridade da biblioteca ANN, limitando cada pedido a visitar no máximo 200 *bins* da *tree*. Em vez de classificar as falsas correspondências por um limite de distância para o vizinho mais próximo, utiliza-se o

ratio test descrito por Lowe [31]. Para um descritor SIFT em I, procuram-se os dois vizinhos mais próximos em J, com distâncias d_1 e d_2, e aceita-se o correspondente (*match*) se $\frac{d_1}{d_2} < 0.6$. Se mais de que um ponto SIFT em I corresponde o mesmo ponto em J, então removem-se todos estes pontos correspondentes, visto que alguns deles podem ser falsos.

O passo seguinte é a estimação da matriz fundamental para cada par, usando o RANSAC [30]. Durante cada iteração do RANSAC, é calculada uma matriz fundamental usando o algoritmo *8-point* [29], seguido de um refinamento não linear. O limite do RANSAC para *outliers* é 0.6% da máxima dimensão da imagem, ou seja, $0.006max(imagewidth, imageheight)$ (cerca de seis *pixels* para uma imagem com 1024×768). A matriz F retornada é refinada através do algoritmo Levenberg-Marquardt [35] nos oito parâmetros da matriz F, minimizando os erros para todos os *inliers* de F. No final, são removidos os *matches outliers* (segundo o limite referido) para obter a matriz fundamental. Se o número restante de *matches* é inferior a 20, todas os *matches* considerados são removidos.

Depois de encontrar um conjunto consistente de *matches*, entre cada par de imagens, procede-se à organização dos mesmos em *tracks*. O *track* é um conjunto conectado de *matching keypoints* ao longo de múltiplas imagens, ou seja, cada *track* corresponde a um ponto 3D. Se um *track* contém mais do que um *keypoint* na mesma imagem, é considerado inconsistente. Os *tracks* consistentes são mantidos contendo no mínimo dois *keypoints*, para a próxima fase do processo de reconstrução.

4.1.2 Estimação dos parâmetros das câmaras e dos pontos 3D

No processo de reconstrução determina-se um conjunto de parâmetros da câmara (rotação, translação e distância focal) e a posição 3D para cada *track* (ponto 3D). Os parâmetros obtidos devem ser consistentes, em que o erro de reprojeção (a soma das distâncias entre as projeções de cada *track* e o seus pontos de interesse correspondentes) é minimizado. O problema de minimização é formulado como um problema não linear dos minímos quadrados, e resolvido com algoritmos tais como *Bundle Adjustment* de Levenberg-Marquardt [35]. Estes algoritmos apenas garantem encontrar os mínimos locais, e os problemas SfM em grande escala tendem a ficar bloqueados em maus mínimos locais, sendo por isso importante fornecer uma boa estimação inicial dos parâmetros. Em vez de estimar os parâmetros para todas as câmaras e *tracks* de uma vez, é apresentada uma abordagem incremental, adicionando uma câmara por iteração.

O processo é iniciado pela estimação dos parâmetros de um par de câmaras. O par inicial deve ter um grande número de pontos SIFT correspondentes (*matches*),

mas também uma grande *baseline*, para que a localização dos pontos 3D seja bem condicionada. Então é escolhido o par de imagens com mais pontos correspondentes, sujeito à condição que esses pontos não podem ser bem condicionados por uma única homografia. Desta forma são evitados casos corrompidos. Neste caso, procura-se uma homografia entre cada par de imagens correspondentes usando o RANSAC com um limite *outlier* de 0.4% do $max(imagewidth, imageheight)$, e guarda-se a percentagem de *matches*, que são os *inliers* para a homografia estimada.
O par inicial é selecionado como aquele que tem a menor percentagem de *inliers*, para a homografia obtida, mas com um número mínimo de 100 *matches*. Os parâmetros das câmaras para este par são estimados usando a implementação de Nistér do algoritmo de cinco pontos [36], e depois as *tracks* visíveis nas duas imagens são trianguladas. Finalmente, é realizado o *Bundle Adjustment* (BA) para as duas imagens, a partir da inicialização dos parâmetros das câmaras, estimados previamente.

De seguida, é adicionada outra câmara à otimização. É selecionada a câmara que observa o maior número de *tracks* dos pontos 3D, que já foram estimados até ao momento. A inicialização dos parâmetros extrínsecos da nova câmara é realizada usando a técnica de transformação linear direta (DLT) [29] dentro um procedimento RANSAC. Para este passo RANSAC, é utilizado um limite de 0.4% do $max(imagewidth, imageheight)$. O DLT também é utilizado para estimar a matriz dos parâmetros intrínsecos, K. É usada a matriz K e a distância focal estimada a partir das EXIF *tags* da imagem, para inicializar a distância focal da nova câmara. A partir deste conjunto inicial de parâmetros, é executado o BA, permitindo apenas modificação da nova câmara e os pontos que esta observa, mantendo o restante modelo inalterado.

Finalmente, adicionam-se os pontos observados pela nova câmara no processo de optimização. O ponto é adicionado, se for observado pelo menos por uma outra câmara estimada até ao momento, e se a triangulação do ponto permitir uma estimação bem condicionada da sua localização. A estimação do condicionamento é obtida pela consideração de todos os pares de raios que podem ser usados para triangular esse ponto, e procurando o par de raios com o máximo ângulo de separação. Se este ângulo é maior que o limite definido (2.0 graus), então o ponto é triangulado. Esta verificação tende a rejeitar pontos no infinito. Os pontos no infinito podem ser bastante úteis para estimar com precisão as rotações das câmaras, no entanto, verificou-se que algumas vezes, estes pontos causam problemas (como utilizar parâmetros das câmaras com ruído para estimar pontos no infinito, podem resultar em pontos errados).

No final dos novos pontos terem sido adicionados, é executado um BA global

para refinar o modelo completo. A solução da minimização do erro de reprojeção é obtida com a biblioteca *Sparse Bundle Adjustment* (SBA) de Loukaris e Argyros [37]. Este processo é repetido uma câmara por iteração, enquanto existirem câmaras que observem um número mínimo de pontos 3D reconstruídos (20 pontos) para se proceder à reconstrução de novos pontos. Por isso, é comum em alguns casos, que apenas um sub-conjunto de imagens seja reconstruído.

Para aumentar a robustez e velocidade, foram efetuadas algumas modificações na abordagem descrita. Primeiro, após cada iteração da otimização, detetam-se as *tracks outliers* que contém pelo menos um ponto SIFT com elevado erro de reprojeção, e removem-se estas *tracks* da otimização. O limite de *outlier* para uma determinada imagem adapta-se, conforme a distribuição corrente dos erros de reprojeção da própria imagem. Neste caso, para uma imagem I, calcula-se d_{80} (80-ésimo dos erros de reprojeção) e utiliza-se $clamp(2.4 \times d_{80}, 4.0, 16.0)$ como o limite de *outlier* ($clamp(x, a, b) = min(max(x, a), b)$). Este método permite que todos os pontos como erro de reprojeção acima de 16.0 *pixels* sejam rejeitados como *outliers*, e que todos os pontos com erro de reprojeção abaixo de 4.0 sejam guardados como *inliers*, através da definição de um limite entre estes dois valores. Depois de rejeitar os *outliers*, procede-se a uma nova otimização, rejeitando os *outliers* no final de cada passo, até que não sejam detetados mais *outliers*.
Segundo, em vez de adicionar uma única câmara por cada passo de otimização, são adicionadas múltiplas câmaras. Neste processo de seleção de câmaras, primeiro procura-se a câmara com maior número de *matches*, M, em relação aos pontos 3D reconstruídos até ao momento, e adiciona-se qualquer câmara com o mínimo de $0.75M$ *matches*.

A estimação dos parâmetros de distorção radial para cada câmara podem ter um grande impacto na precisão da reconstrução, porque várias câmaras e lentes produzem imagens com distorção perceptível. Por isso, são estimados dois parâmetros de distorção, k_1 e k_2, para cada câmara. O mapeamento de um ponto 2D $p = (p_x, p_y)$ para um ponto distorcido $p' = (x', y')$ é descrito pelas Equações 4.1, 4.2 e 4.3.

$$\rho^2 = \frac{p_x}{f}^2 + \frac{p_y}{f}^2 \tag{4.1}$$

$$\alpha = k_1\rho^2 + k_2\rho^4 \tag{4.2}$$

$$p' = \alpha p \tag{4.3}$$

Onde f é a distância focal estimada (é assumido que o centro da distorção é o centro da imagem, e o centro da imagem é a origem do sistema de coordenadas

da imagem). Quando novas câmaras são inicializadas, defini-se $k_1 = 0$ e $k_2 = 0$, mas estes parâmetros são estimados durante o BA. Para evitar valores enormes indesejáveis para estes parâmetros, é adicionado o termo $\lambda(k_1^2 + k_2^2)$ ($\lambda = 10.0$) na função objetiva de cada câmara.

Como exemplo, do tempo total de execução para o processo SfM para conjuntos de imagens mencionados em [14], este varia desde algumas horas (Great Wall, aproximadamente 3 horas para 120 fotografias) até duas semanas (Notre Dame, 2635 fotografias). A maior parte do tempo é despendido no *matching* dos pontos SIFT entre as imagens e nas iterações do BA, que se torna mais lento consoante se vão adicionando novas imagens. A complexidade da fase de *matching* é quadrática em relação ao número, mas cada par de imagens pode ser comparado independentemente, por isso o tempo de processamento pode ser reduzido através do paralelismo. A velocidade do BA depende de vários fatores, como o número de imagens e de pontos, e a interligação das imagens (quando muitas câmaras observam os mesmos conjuntos de pontos, o BA tende a tornar-se mais lento).

4.1.3 Bundle Adjustment

A perspetiva da câmara pode ser parametrizada por onze elementos da matriz de projeção. Assumindo que os *pixels* são quadrados e que o centro de projeção é coincidente com o centro da imagem, o número de parâmetros é reduzido para sete: a orientação 3D (3 parâmetros), o centro da câmara (3 parâmetros) e a distância focal (1 parâmetro). O sistema do Bundler também calcula dois parâmetros de distorção radial, k_1 e k_2, por isso o número total de parâmetros por câmara é nove.

Para parametrizar as rotações, é utilizada uma rotação incremental, ω, onde

$$R(\theta, \hat{n}) = I + \sin\theta[\hat{n}]_\times + (1 - \cos\theta)[\hat{n}]_\times^2, \qquad \omega = \theta\hat{n} \tag{4.4}$$

é a matriz de rotação incremental aplicada para uma rotação inicial, e

$$[\hat{n}]_\times = \begin{bmatrix} 0 & -\hat{n}_z & \hat{n}_y \\ \hat{n}_z & 0 & -\hat{n}_x \\ -\hat{n}_y & \hat{n}_x & 0 \end{bmatrix} \tag{4.5}$$

Os nove parâmetros são agrupados num vetor, $\Theta = [\omega, c, f, k_1, k_2]$. Cada ponto é parametrizado por uma posição 3D, $\mathbf{X}$.

A obtenção dos parâmetros pode ser formulada como um problema de otimização. Considerando um conjunto de n câmaras parametrizadas por Θ_i, um conjunto de m *tracks* parametrizadas por $\mathbf{X}_i$ e um conjunto de projeções 2D parametrizados por

q_{ij}, onde q_{ij} é a projeção do ponto observado na *track* $j-th$ da câmara $i-th$.

Sendo $P(\Theta, X)$ a matriz que mapeia um ponto 3D num ponto 2D de uma câmara com parâmetros Θ. A matriz P transforma $\mathbf{X}$ em coordenadas homogéneas da imagem, aplica a divisão de perspetiva e a distorção radial:

$$\mathbf{X}' = R(\mathbf{X} - c) \tag{4.6}$$

$$P_0 = [-fX'_x/X'_z - fX'_y/X'_z]^T \tag{4.7}$$

$$P = g_{rd}(P_0) \tag{4.8}$$

onde g_{rd} é a equação de distorção radial apresentada na Secção 4.1.2.

A optimização tem como objetivo a minimização da soma dos erros de reprojeção, Equação 4.9.

$$\sum_{i=1}^{n}\sum_{j=1}^{m}\omega_{ij}\|q_{ij} - P(\Theta_i, \mathbf{X}_j)\| \tag{4.9}$$

O ω_{ij} é usado como um indicador de variável, onde $\omega_{ij} = 1$ se a câmara i observa o ponto j, e $\omega_{ij} = 0$ caso contrário.

Na inicialização da nova câmara, existem uma ou duas estimações independentes da distância focal. Uma estimação, f_1, é $\frac{1}{2}K_{11} + K_{22}$, onde K é a estimação da matriz dos parâmetros intrínsecos obtida usando o DLT. A outra, f_2, é calculada a partir das EXIF *tags* da imagem, e pode ser indefinida se não existirem as EXIF *tags*. Em geral, esta estimação é boa, mas ocasionalmente pode não existir. Preferencialmente, utiliza-se f_2 como inicialização quando existe, mas é efetuada uma verificação ($0.7f_1 < f_2 < 1.4f_1$) para garantir que f_2 não discorda consideravelmente da estimação obtida com o DLT. Se este teste falhar, utiliza-se f_1. Caso contrário, utiliza-se f_2 e adiciona-se o termo $\gamma(f - f_2)^2$ ($\gamma = 0.001$) para a função objetiva, de forma a manter a distância focal próxima da estimação inicial.

4.1.4 Critério de seleção de imagens

Na determinação de como uma imagem I_j representa um conjunto de pontos S, durante a seleção de um objeto, a pontuação é calculada pela soma de três termos, Equação 4.10.

$$E_{visible} + \alpha E_{angle} + \beta E_{detail} \tag{4.10}$$

Onde $\alpha = \frac{1}{3}$ e $\beta = \frac{2}{3}$.

Para calcular o termo da visibilidade, primeiro verifica-se se $S \cap Pontos(C_j)$ é vazio. Se a interseção for vazia, o objeto não é visível para C_j, e $E_{visible} = -\infty$. Caso contrário, $E_{visible} = \frac{n_{inside}}{|S|}$, onde n_{inside} representa o número de pontos em S

que são projetados dentro da fronteira da imagem I_j.

O termo E_{angle} é usado para favorecer vistas frontais, de um conjunto de pontos, sobre vistas obliquas. Para calcular E_{angle}, primeiro ajusta-se um plano de pontos em S usando o RANSAC. Se a percentagem de pontos em S, que são *inliers* para calcular o plano, for superior ao limite de 50%, então as câmaras que observam o objeto com vista frontal são favorecidas, definindo $E_{angle} = V(C_j) \cdot \hat{n}$. O parâmetro V indica a direção da perspetiva e $\hat{n}$ a normal do plano calculado. Se a percentagem for inferior que 50% dos pontos que ajustam o plano, então $E_{angle} = 0$.

Finalmente, E_{detail} favorece vistas do objeto com alta resolução. E_{detail} é definido como a área, em *pixels*, da caixa delimitadora das projeções de S na imagem I_j (considerando apenas os pontos que são projetados dentro da fronteira de I_j). E_{detail} é normalizado pela área da maior caixa delimitadora, por isso, a vista com a maior resolução terá uma pontuação de 1.0.

4.2 Projeto Bundler

O Bundler é um software SfM desenvolvido em C/C++ por Noah Snavely, para conjuntos de imagens desordenados (por exemplo, imagens existentes na Internet) [5]. Uma versão recente deste software foi utilizada no projeto Photo Tourism. Este é um projeto Microsoft, que consiste num sistema para navegar enormes conjuntos de fotos em 3D.

O Bundler aceita um conjunto de imagens, pontos de interesse (*features*), e correspondências de imagens (*matches*), e produz a reconstrução 3D da câmara e da geometria do espaço. O sistema reconstroi o espaço incrementalmente, algumas imagens por instante de tempo, usando uma versão modificada da biblioteca *Sparse Bundle Adjustment* (SBA).

O SBA é um pacote de *software* desenvolvido por Lourakis e Argyros, como um sistema de otimização [37]. Este *software* é utilizado como o último passo em muitos algoritmos de reconstrução a partir de múltiplas imagens, para obter os parâmetros estimados da estrutura 3D e movimento (matriz da câmara). Sabendo a estimação inicial, o BA corrige, simultaneamente, o movimento e a estrutura através da minimização do erro de reprojeção entre os pontos observados e os estimados nas imagens.

O Bundler contém implementados diversos algoritmos de visão computacional, destacando-se os seguintes:

- Estimação da matriz fundamental.

- Estimação da posição relativa para duas câmaras calibradas, a partir de cinco pontos correspondentes entre as imagens (algoritmo *5-point*).
- Triangulação.

Os resultados do Bundler são nuvens esparsas de pontos. Para pontos densos, existe um pacote de *software* denominado *Patch-based Multi-view Stereo* (PMVS2) [6, 38], desenvolvido por Yasutaka Furukawa, para executar um *Multi-view Stereo* (MVS) denso. Para realizar a reconstrução 3D e visualizar a nuvem de pontos densa, deve-se executar a seguinte sequência de passos, apresentada na Figura 4.1.

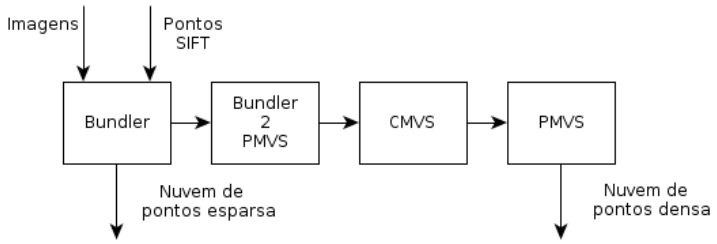

Figura 4.1: *Sequência de passos para construir uma nuvem de pontos densa*

O Bundler2PMVS é um programa para converter os resultados do Bundler no formato adequado, para introduzir como parâmetros de entrada para o PMVS.

O *Clustering Views for Multi-view Stereo* (CMVS) [6, 39] é uma ferramenta de pré-processamento para aumentar a velocidade e precisão do PMVS. A execução do CMVS é opcional.

4.2.1 PMVS

O *Patch-based Multi-view Stereo* (PMVS), desenvolvido por Yasutaka Furukawa, é um algoritmo MVS para reconstrução de nuvens densas de pontos [6, 38]. Fornecendo um conjunto de imagens e parâmetros das câmaras, o PMVS reconstroi a estrutura 3D de um objeto ou espaço, visível nas imagens. Este algoritmo utiliza propriedades da geometria epipolar, para gerar conjuntos densos de correspondências entre duas imagens. A estimação da relação epipolar (matriz fundamental) advém dos parâmetros das câmaras fornecidos previamente, por exemplo pelo Bundler. As duas matrizes de projeção são utilizadas para determinar a matriz fundamental entre as duas câmaras. O PMVS divide cada imagem em pequenas partes (*patches*) e aplica o algoritmo de Harris para deteção de pontos de interesse e para procurar

pontos comuns entre as imagens (*matching*). Quando um ponto de interesse é detectado num *patch*, o *patch* é utilizado para *matching*. A procura da correspondência para cada *patch*, é realizada ao longo da sua linha epipolar. Se for encontrada uma correspondência, procede-se à reconstrução tridimensional do *patch*.

4.2.2 CMVS

O *Clustering Views for Multi-view Stereo* (CMVS) [6, 39] é uma ferramenta de pré-processamento para o PMVS. O seu objetivo consiste em remover imagens redundantes que são desnecessárias para as reconstruções MVS. Este pré-processamento permite aumentar a velocidade e precisão de reconstrução do PMVS.

Em geral, os algoritmos MVS apresentam dificuldades quando são utilizados com um grande número de imagens, devido à falta de recursos de memória e capacidade computacional. Os dados de entrada para o CMVS são os resultados obtidos por um *software* SfM como o Bundler. O CMVS decompõe as imagens em grupos de imagens (*clusters*) de tamanho gerenciável. Um *software* MVS consegue processar cada *cluster* independentemente e em paralelo, onde a união das reconstrução de todos os *clusters* não deve perder qualquer detalhe que possa ser obtido através do processamento do conjunto completo.

4.2.3 Projeto Geo-Bundler

O projeto *Geo-Accurate Dense Point Cloud Generation* (Geo-Bundler) [28] consiste numa versão modificada do Bundler de Noah Snavely.

O objetivo do projeto é criação de nuvens de pontos densas, referenciadas a um sistema de coordenadas, a partir de imagens aérias e de um sistema IMU/GPS. Este *software* foi desenvolvido a partir de uma modificação do Bundler, para utilizar informação proveniente de sensores IMU/GPS. Esta informação é utilizada como a estimação inicial no processo do *Bundle Adjustment*, evitando o uso de técnicas de estimação da posição.
Outra característica interessante do projeto é a alteração do processo de deteção e correspondência de pontos de interesse entre as imagens. Este projeto utiliza uma versão do SIFT adaptada para a GPU, o SiftGPU [18]. A utilização da GPU apresenta vantagens significativas no desempenho do SIFT, para deteção e correspondência de pontos de interesse.

Neste projeto, foi desenvolvida uma aplicação para extrair e estabelecer a correspondência dos pontos de interesse entre as imagens, com base na biblioteca SiftGPU. A biblioteca SiftGPU disponibiliza configurações para GPUs baseadas em CUDA (GPU da NVIDIA) ou GLSL (baseado em OpenGL). A aplicação do Geo-Bundler

foi desenvolvida e testada apenas com uma GPU da Nvidia com suporte CUDA. Nesta dissertação, a GPU utilizada não suporta o CUDA, e por isso foi necessário modificar a aplicação para utilizar o GLSL. Através da análise do código da aplicação e em conjunto com o manual do SiftGPU, efetuaram-se algumas alterações para adaptar a aplicação de forma a ser executada numa GPU que suporte o GLSL.

Na Figura A.2 do Anexo A, encontra-se um diagrama de blocos que apresenta as modificações do Geo-Bundler em relação ao projeto Bundler.

5 Bundler

O Bundler é um projeto de reconstrução tridimensional para enormes conjuntos desordenados de imagens. Este projeto é gratuito e de código aberto, por isso é a ferramenta escolhida para análise e realização de testes que serão descritos ao longo deste documento. Neste capítulo será realizada uma descrição técnica dos principais blocos do Bundler, assim como uma análise da sua aplicabilidade na robótica.

5.1 Descrição do Bundler

O Bundler é uma ferramenta de reconstrução 3D incremental e utiliza um pacote de software modificado do SBA de Lourakis e Argyros [37]. Nesta dissertação é explorada a versão 0.4 de 2010 para Linux. Para executar o Bundler é necessário instalar algumas ferramentas externas, que são essenciais para os passos iniciais. Destas ferramentas destacam-se o ImageMagic, o SIFT e a biblioteca ANN. O ImageMagic é utilizado para converter as imagens do formato '.jpg' em '.pgm'. Esta conversão é necessária antes de executar o SIFT. O SIFT é uma ferramenta para extrair pontos de interesse nas imagens. Este passo é crucial no processo de reconstrução. A biblioteca ANN é utilizada, por exemplo na procura de pontos correspondentes entre as imagens.

O Bundler pode ser descrito essencialmente em quatro passos principais (diagrama de blocos da Figura 5.1):

1. Criação de uma lista com o nome e localização das imagens e a respetiva

distância focal. A distância focal é extraída das EXIF *tags* das imagens usando um *script*.

2. Extração dos pontos de interesse de cada imagem através do SIFT. Antes de executar o SIFT as imagens são convertidas de '.jpg' para o formato '.pgm'.

3. Procura de pontos correspondentes entre os pares de imagens e escrita dos resultados num ficheiro.

4. Execução do Bundler. Neste passo são fornecidos três ficheiros: a lista com as imagens e a respetiva distância focal; os resultados dos pontos correspondentes entres as imagens e um ficheiro com a lista de opções disponíveis para executar o Bundler.

No Anexo A encontra-se uma descrição detalhada do procedimento para utilizar o Bundler.

A Figura 5.1 representa um diagrama de blocos com os passos principais do procedimento completo da reconstrução 3D. Os resultados do processamento do Bundler

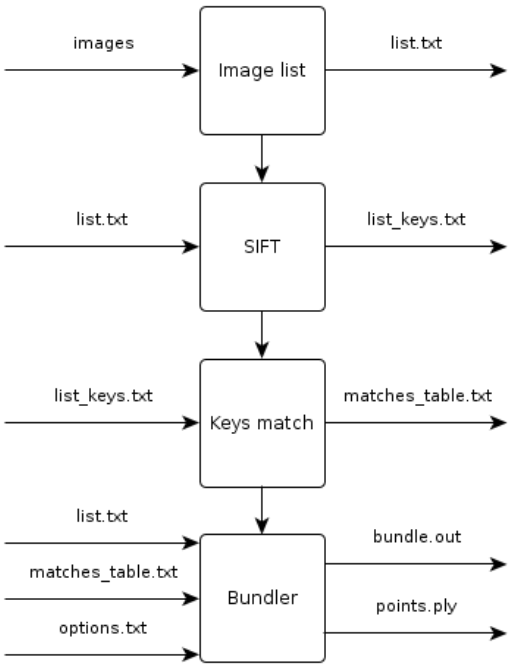

Figura 5.1: *Sequência de passos do processo de reconstrução 3D*

são escritos em dois tipos ficheiros: *bundle.out* e *points.ply*. O *bundle.out* é o ficheiro com os resultados finais da reconstrução. Este ficheiro contém o número de câmaras,

número de pontos, os parâmetros intrínsecos e extrínsecos de cada câmara e a posição, cor e lista dos pontos 3D reconstruídos. O ficheiro *points.ply* contém as câmaras e os pontos reconstruídos. Este tipo de ficheiro pode ser utilizado por aplicações como o MeshLab, para visualizar a nuvem de pontos 3D.

A estrutura do ficheiro com os resultados finais é a seguinte:

<número de câmaras/imagens> <número de pontos>
<câmara 1>
<câmara 2>
...
<câmara N>
<ponto 1>
<ponto 2>
...
<ponto N>

Estrutura da câmara:

$<f>$ $<k_1>$ $<k_2>$	[distância focal e dois coeficientes de distorção radial]
$<R>$	[matriz de rotação 3x3]
$<t>$	[vetor 3 de translação]

Estrutura do ponto:

<posição>	[vetor 3 de posição 3D do ponto]
<cor>	[vetor 3 de cor RGB do ponto]
<lista>	[lista de imagens onde o ponto é observado]

Estrutura da lista onde o ponto é observado:

<comprimento> <câmara> <key> $<x>$ $<y>$
[dimensão da lista, índice da câmara, índice da key SIFT e posição x e y do ponto]

Como foi referido anteriormente, o Bundler permite fornecer uma lista de opções para configurar a sua execução. Das opções disponíveis destacam-se as seguintes:

- *match_ table* : serve para indicar o ficheiro com os pontos correspondentes entre as imagens.

- *output* : utiliza-se para definir o nome do ficheiro do tipo '.out' com os resultados finais.
- *output_all* : define o prefixo do nome para os ficheiros com resultados intermédios.
- *output_dir* : nome do diretório onde são guardados os resultados.
- *variable_focal_length* : indica que a distância focal das imagens é variável.
- *use_focal_estimate* : indica que deve ser usada a distância focal extraída das EXIF *tags*.
- *constrain_focal* : adiciona uma restrição nas distâncias focais para se manterem próximas dos seus valores estimados.
- *constrain_focal_weight* : peso nas restrições da distância focal.
- *estimate_distortion* : indica para estimar a distorção radial.
- *run_bundle* : comando de execução do Bundler.

O Bundler disponibiliza mais opções, no entanto, estas destacam-se como as *standard*, para situações em que se pretende efetuar a reconstrução 3D de um conjunto desordenado de imagens, que podem ser provenientes de diferentes câmaras.

5.2 Modelo da Câmara

O modelo da câmara utilizado no Bundler é o *pinhole*. Os parâmetros estimados para cada câmara são: distância focal (f), dois coeficientes de distorção radial (k_1 e k_2), rotação (R) e translação (t). A projeção de um ponto 3D do Mundo, X, para as coordenadas da câmara, P, é definida pelas equações seguintes.
Conversão de um ponto do referencial do Mundo para o referencial da câmara, Equação 5.1.

$$P = R \cdot X + t \tag{5.1}$$

Divisão de perspectiva, Equação 5.2.

$$p = \frac{-P}{P.z} \tag{5.2}$$

Onde $P.z$ é a coordenada Z do ponto P.
Conversão para coordenadas em *pixels*, Equação 5.3.

$$p' = f \cdot r(p) \cdot p \tag{5.3}$$

A função $r(p)$ calcula o fator de escala para remover a distorção radial, Equação 5.4.

$$r(p) = 1.0 + k_1 \cdot |p|^2 + k_2 \cdot |p|^4 \tag{5.4}$$

Estas equações definem a projeção em *pixels*, onde a origem da imagem é o seu centro. O eixo positivo do x aponta para a direita e o eixo positivo de y aponta para cima. No sistema de coordenadas da câmara, o eixo positivo de z aponta para trás, sendo que a perspetiva da câmara está direcionada segundo o eixo negativo de Z, como no OpenGL, Figura 5.2.

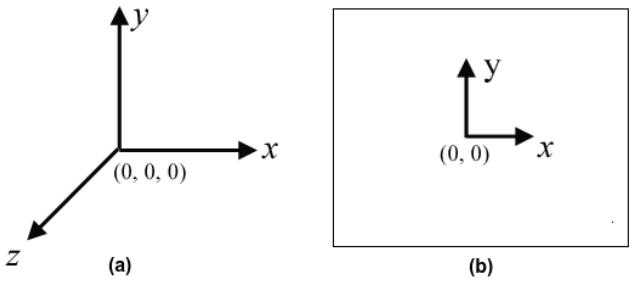

Figura 5.2: *Sistema de coordenadas do Bundler. a) Sistema de coordenadas da câmara segundo o OpenGL. b) Sistema de coordenadas do plano da imagem do Bundler*

5.3 Aplicação do Bundler na Robótica

O SfM consiste na estimação da estrutura do espaço e do movimento em simultâneo, a partir de um sequência de imagens. Em geral, esta reconstrução resulta em nuvens de pontos esparsas ou densas, dependendo do *software* utilizado. As nuvens de pontos 3D têm interesse em aplicações no domínio da robótica, por exemplo na navegação, localização, percepção e mapeamento do ambiente. Atualmente com dispositivos laser, GPS e IMU, podem-se obter nuvens de pontos com elevada qualidade, mas estas soluções acarretam um custo elevado. A utilização de câmaras para reconstrução tridimensional é uma solução mais económica que a anterior, e permite extrair mais informação sobre o espaço como por exemplo a cor e textura dos objetos. Neste sentido, esta dissertação pretende explorar as potencialidades de uma ferramenta SfM, o Bundler, para aplicações robóticas. A análise do Bundler tem em consideração duas possibilidades: (1) avaliar os resultados obtidos em função do tempo de processamento e perceber a sua aplicabilidade para aplicações em tempo real e/ou (2) analisar a qualidade da nuvem de pontos resultante, e se possível comparar com uma nuvem de pontos de referência, para explorar a hipótese de utilização em pós-processamento para tarefas de mapeamento e monitorização.

No caso de um aplicação em tempo real, o fator determinante para validar a viabilidade do Bundler, é o desempenho de processamento. Em situações de pós-processamento, o tempo de processamento não apresenta uma restrição. Neste caso, o essencial é a qualidade da reconstrução tridimensional. O facto do Bundler ter sido desenvolvido para conjuntos desordenados de imagens e estimar os parâmetros das câmaras sem intervenção por parte do utilizador, são duas características interessantes para pós-processamento.

6

Caracterização do Bundler

Numa fase inicial, executou-se o Bundler com alguns conjuntos de imagens para verificar visualmente as nuvens de pontos 3D, e também para analisar o tempo de processamento. Os conjuntos de imagens podem ser desordenados e provenientes de diferentes câmaras. O Bundler define a ordem do processamento das imagens de acordo com número de pontos de interesse entre as imagens, ou seja, escolhe apenas as imagens que têm maior número de pontos de interesse em comum. Estas características são interessantes, por exemplo para aplicações de pós-processamento de operações de mapeamento aéreo através de um UAV. Durante a operação, o UAV vai adquirindo imagens sequencialmente, mas, em algumas situações, nem todas as imagens têm interesse para o processo de reconstrução e por isso, a selecção de imagens do Bundler é uma mais valia para o sucesso da reconstrução 3D.

Durante o processo de testes e caracterização, pretende-se analisar os blocos principais que exigem maior esforço computacional, a deteção dos pontos SIFT e o *Bundle Adjustment* (BA). Esta análise tem como objetivo, perceber o desempenho do Bundler em função da dimensão dos conjuntos de imagens e tentar encontrar, se possível, soluções de optimização.

6.1 Exploração do projeto

O Bundler estima a estrutura do espaço e o movimento, utilizando apenas o conjunto de imagens. Para o primeiro teste utilizou-se o conjunto de imagens obtido pelo

Rochester Institute of Technology's Digital Imaging e pela plataforma sensorial do *Remote Sensing Laboratory's WASP*. As imagens foram adquiridas sobre *Rochester* na cidade *New York*. Este conjunto de dados denomina-se Rocgeo e é fornecido como exemplo para o projeto Geo-Bundler [28]. O conjunto tem 28 imagens, adquiridas de uma perspectiva aérea, com resolução 4000×2672.

No primeiro teste realizado com o Bundler, pretende-se estimar os parâmetros das câmaras e os pontos 3D, assim como observar a nuvem esparsa de pontos 3D. A Figura 6.1 (a) e (b) apresenta exemplos de imagens do conjunto e a nuvem de pontos esparsa gerada pelo Bundler, respetivamente. As nuvens de pontos foram observadas através do MeshLab. Os pontos coloridos no topo da Figura 6.1 (b),

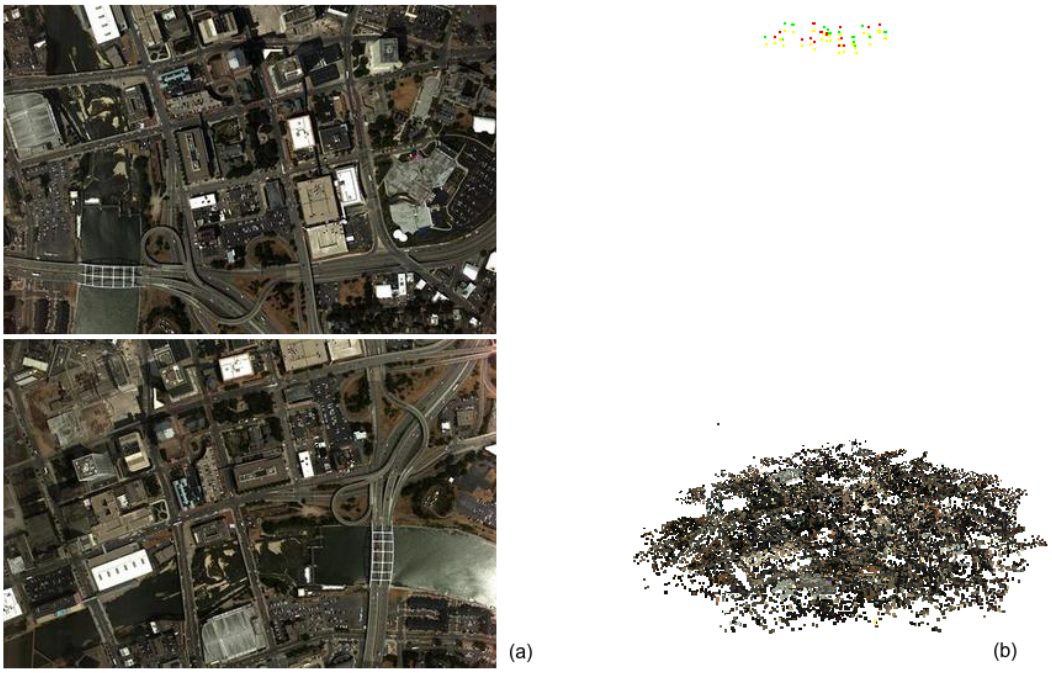

Figura 6.1: *Resultado do Bundler para o conjunto de imagens Rocgeo. a) Exemplos de imagens do conjunto. b) Nuvem de pontos esparsa resultante do Bundler.*

representam as posições das câmaras para cada imagem. A nuvem de pontos gerada pelo Bundler não é densa. Os resultados do Bundler são nuvens de pontos esparsas e os parâmetros intrínsecos e extrínsecos das câmaras. Estes resultados são obtidos sem fator de escala. Para visualizar uma nuvem de pontos densa, pode-se utilizar os programas CMVS e PMVS com os resultados do Bundler. Para o conjunto de imagens anterior, a nuvem de pontos densa encontra-se na Figura 6.2.

A estimação das posições dos pontos 3D e dos parâmetros das câmaras são obtidas através de um método denominado como *Bundle Adjustment* (BA). O BA consiste num processo iterativo de optimização, para estimar em simultâneo, as coordenadas

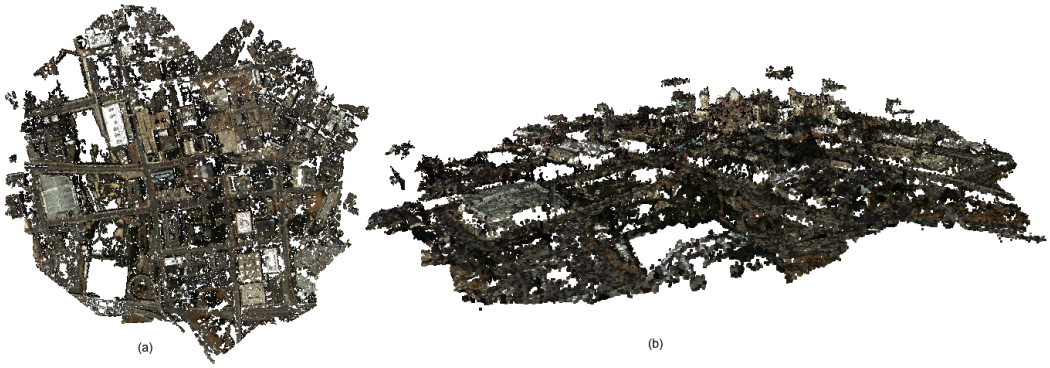

Figura 6.2: *Resultado do CMVS e PMVS para o conjunto de imagens Rocgeo. a) Perspectiva de topo. b) Perspectiva lateral.*

3D dos pontos observados nas imagens e os parâmetros intrínsecos e extrínsecos das câmaras. Fornecendo uma estimação inicial dos parâmetros, este processo de optimização estima novos parâmetros de forma a diminuir o erro de reprojeção entre os pontos observados nas imagens e os pontos estimados. A diminuição do erro de reprojeção é alcançada usando o algoritmo não linear dos mínimos quadrados. O BA exige um elevado esforço computacional, que aumenta consoante a adição de novas imagens ao processo de estimação.

O Bundler disponibiliza dois métodos de operação que apresentam um desempenho no tempo de processamento muito diferente, Tabela 6.1. No método II, o

Tabela 6.1: *Métodos de adição de novas imagens no BA*

Método	Observações
I	adiciona N imagens por iteração no BA
II	adiciona apenas uma imagem por iteração no BA

Bundler adiciona apenas uma imagem por iteração no processo de optimização. Para aumentar o desempenho e a robustez, uma das modificações foi a adição de múltiplas imagens por iteração (método I), no processo de optimização. A seleção das imagens é realizada com base no número de *matches* com os pontos reconstruídos até ao momento.

A Figura 6.3 apresenta o gráfico que relaciona o tempo despendido no BA com o número de imagens, para o conjunto Rocgeo. No final de cada passo de estimação, é realizada um verificação do erro de reprojeção dos pontos para identificar e remover os *outliers*. Após a eliminação dos *outliers*, os parâmetros das câmaras são estimados novamente. Este processo repete-se até que não existam *outliers* ou um número mínimo de pontos. Por isso, para cada iteração do BA é apresentado o tempo para

a primeira estimação dos parâmetros das câmaras e dos pontos 3D, e o tempo total que também inclui as iterações realizadas pelo BA, após a eliminação de *outliers*. Analisando a Figura 6.3, verifica-se, como previsto, que o tempo no processo de

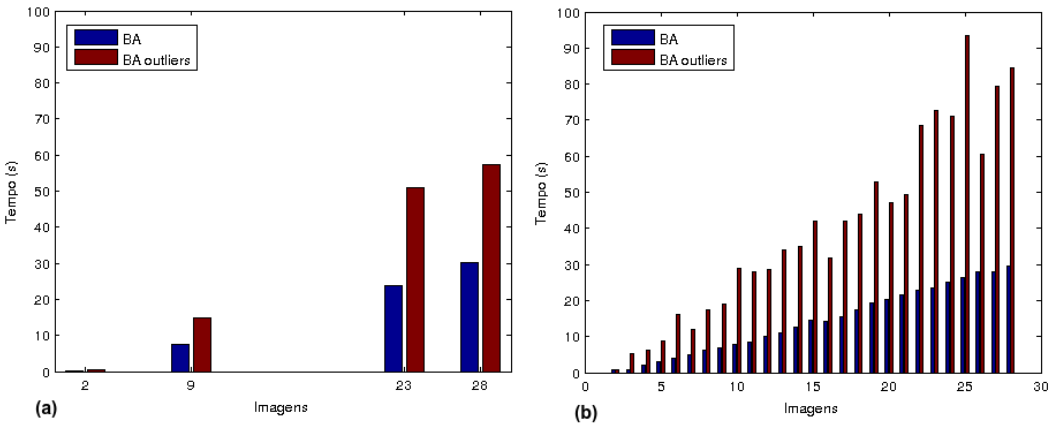

Figura 6.3: *Análise da evolução do tempo no BA consoante o número de imagens. a) Método I. b) Método II*

estimação aumenta consideravelmente conforme o aumento do número de imagens. Comparando as duas situações, confirma-se que a abordagem com múltiplas imagens por iteração, apresenta uma melhoria significativa no desempenho do Bundler. Os registos de tempo apresentados na figura, referem-se apenas ao tempo despendido na estimação de parâmetros usando a biblioteca SBA. O registo total do processo de reconstrução pode ser observado na Tabela 6.2. Este registo engloba todos os passos do processo de reconstrução, como o cálculo da geometria entre todos os pares de imagens, seleção do par inicial, inicialização das câmaras, verificação e remoção de *outliers*, entre outros.

Tabela 6.2: *Resultados do Bundler para o conjuntos de imagens Rocgeo*

Método	**I**	**II**
Imagens	28/28	28/28
Resolução	4000×2672	4000×2672
Pontos SIFT	288 300	288 300
BA (s)	341.940	1 309.950
Pontos 3D	34 094	34 890

6.2 Conjuntos de imagens adquiridos por sistemas robóticos do LSA

A exploração do Bundler tem como objetivo validar a sua aplicabilidade no contexto da robótica. Então, no sentido de explorar o desempenho do Bundler em ambientes diferentes, realizaram-se alguns testes utilizando conjuntos de imagens de operações realizadas por sistemas autónomos do LSA. Os testes foram realizados com imagens adquiridas por dois sistemas autónomos terrestres, o Fleximap e o Tigre, e por um veículo aéreo quadrotor, o Pelican.

O Tigre realizou uma operação num parque da cidade do Porto, em que se deslocava autonomamente em direção a um objecto de cor laranja pousado no solo. Este veículo autónomo incorpora duas câmaras localizadas nas laterais esquerda e direita. Foram realizados dois testes com 32 imagens de cada câmara, e um terceiro teste com as imagens de ambas as câmaras. As imagens tem uma resolução de 1278×958 *pixels*. Na Figura 6.4 (a) e (b), pode ser observado um exemplo de uma imagem do conjunto da câmara da direita, e a nuvem de pontos resultante do Bundler, respetivamente. Os pontos vermelhos e amarelos na nuvem de pontos correspondem às posições das câmaras para cada imagem.

Figura 6.4: *Teste do Bundler com o conjunto de imagens da câmara direita do Tigre. a) Exemplo de uma das imagens do conjunto. b) Nuvem de pontos do Bundler.*

Na Figura 6.5 (a) e (b), pode ser observado um exemplo de uma imagem do conjunto da câmara da esquerda, e a nuvem de pontos resultante do Bundler, respetivamente.

Na Figura 6.6, pode ser observada a nuvem de pontos resultante do Bundler com as imagens de ambas as câmaras.

Observando os resultados das nuvens de pontos obtidas, a partir do conjunto de imagens do Tigre, verifica-se que a reconstrução tridimensional do espaço não é ob-

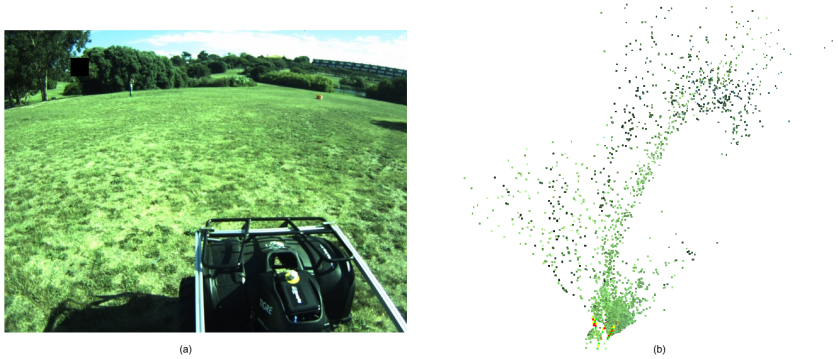

Figura 6.5: *Teste do Bundler com o conjunto de imagens da câmara esquerda do Tigre. a) Exemplo de uma das imagens do conjunto. b) Nuvem de pontos do Bundler.*

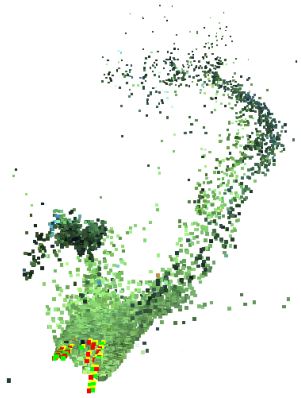

Figura 6.6: *Teste do Bundler com o conjunto de imagens de ambas as câmaras do Tigre.*

tida com sucesso. No caso da nuvem de pontos da Figura 6.4, a trajetória do Tigre é perceptível pela observação dos pontos vermelhos e amarelos na nuvem de pontos. Os resultados das Figuras 6.4 e 6.6 não permitem observar uma trajetória coerente com o movimento efetuado pelo veículo autónomo. Estes resultados questionam o desempenho do Bundler para sequências de imagens obtidas a partir de movimentos longitudinais com uma perspetiva em profundidade. Numa perspetiva em profundidade, existe pouca variação da posição dos objetos na imagem, variando com mais intensidade a dimensão do objeto (aumenta o número de *pixels* na imagem). No caso de uma perspetiva lateral, a variação da posição dos objetos nas imagens é mais perceptível comparativamente a uma perspetiva em profundidade.

No conjunto de imagens do Tigre, as câmaras estão numa posição fixa, e um fator que também pode influenciar os resultados é o facto da estrutura frontal do veículo estar presente em todas as imagens, e sempre na mesma posição. Este fator pode induzir em erro no processo de reconstrução, porque os pontos SIFT associados ao veículo indicam que não ocorreu movimento ao longo da sequência de imagens. Outro fator que também pode estar em causa do insucesso da reconstrução, é a possível existência de falsos *matches* de pontos de interesse detetados no relvado.

O Fleximap é um sistema móvel de mapeamento 3D através de laser e imagem, para plataformas rodoviárias e ferroviárias. O conjunto de imagens utilizado no teste foi obtido de uma operação de mapeamento de uma plataforma ferroviária. Este conjunto contêm imagens obtidas a partir de duas câmaras localizadas em posições diferentes. Utilizaram-se 44 imagens para cada câmara e com resolução 1032×778.

Na Figura 6.7 encontra-se um exemplo de uma imagem do conjunto e a nuvem de pontos gerada pelo Bundler para a câmara 1 do Fleximap.

Figura 6.7: *Teste do Bundler com o conjunto de imagens da câmara direita do Fleximap. a) Exemplo de uma das imagens do conjunto. b) Nuvem de pontos do Bundler.*

Na Figura 6.8 encontra-se um exemplo de uma imagem do conjunto e a nuvem de pontos gerada pelo Bundler para a câmara esquerda do Fleximap.

Figura 6.8: *Teste do Bundler com o conjunto de imagens da câmara esquerda do Fleximap. a) Exemplo de uma das imagens do conjunto. b) Nuvem de pontos do Bundler.*

Na Figura 6.9 encontra-se a nuvem de pontos do Bundler com o conjunto de imagens de ambas as câmaras. Analisando os resultados obtidos com estes conjuntos

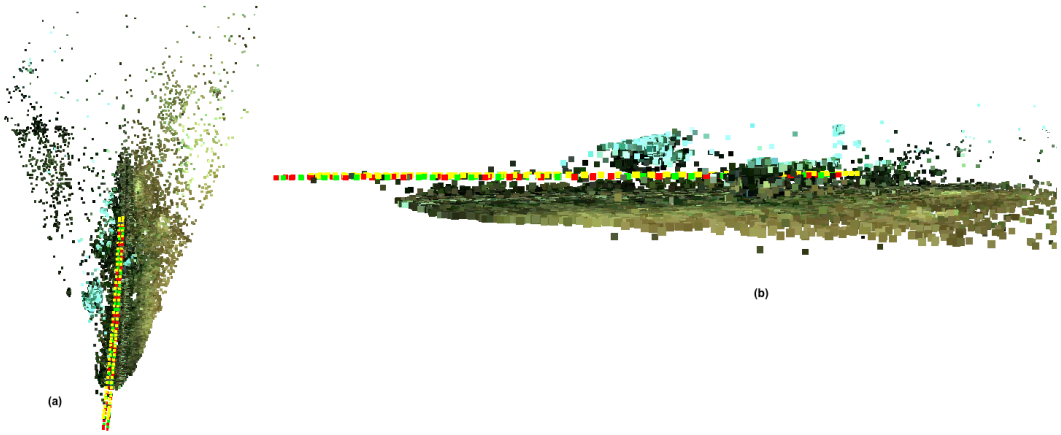

Figura 6.9: *Teste do Bundler com o conjunto de imagens de ambas as câmaras do Fleximap. a) Perspetiva de topo. b) Perspetiva lateral.*

de imagens, é possível identificar as posições das câmaras e confirmar que a estimação corresponde à trajetória realizada pelo Fleximap. Na Figura 6.9 (a) consegue-se visualizar que as câmaras estão em paralelo ao longo do percurso efetuado e em (b) observa-se a estimação da altura da posição das câmaras em relação ao solo. A nuvem de pontos é representada sem escala, mas visualmente a estimação é coerente com a localização real das câmaras.

O Pelican é um veículo aéreo quadrotor com uma câmara direcionada para baixo. O conjunto de imagens utilizado no teste foi obtido durante uma operação, em que o Pelican inspecionava um alvo laranja, pousado no solo do parque da cidade do Porto. O conjunto contém diversas imagens a curtas e longas distâncias do alvo, Figura 6.10 (a). Foram utilizadas 55 imagens com resolução 752×480. Na Figura 6.10 (b) encontra-se a nuvem de pontos obtida com o Bundler.

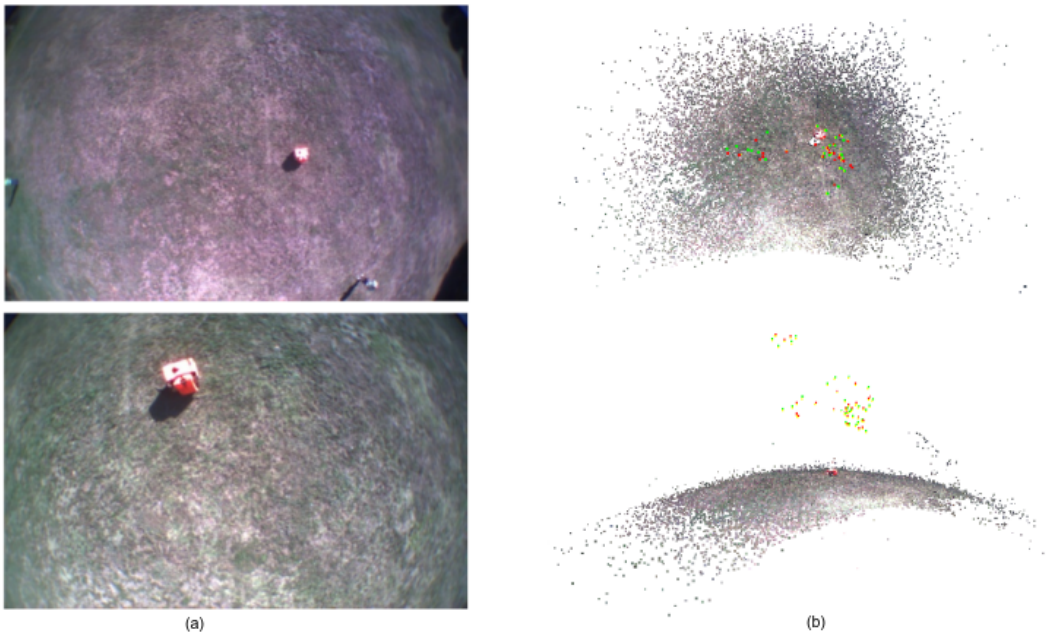

Figura 6.10: *Teste do Bundler com o conjunto de imagens do Pelican. a) Exemplo de imagens do conjunto. b) Nuvem de pontos do Bundler.*

Utilizando os resultados obtidos com o Bundler, procedeu-se a construção da nuvem de pontos densa através do PMVS, como pode ser visualizado na Figura 6.11. Comparando a nuvem de pontos densa com as imagens originais, é perceptível a sua

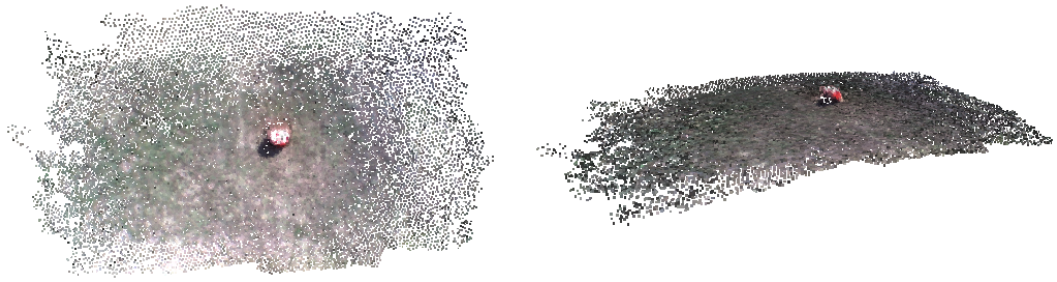

Figura 6.11: *Nuvem de pontos densa do Pelican, obtida com o PMVS a partir dos resultados do Bundler*

semelhança a olho nu. O alvo e o espaço estão reconhecíveis, e na nuvem de pontos

observa-se uma correção à distorção das imagens.

Na Tabela 6.3 estão apresentados o número de pontos 3D registados e o tempo do BA, para os testes efetuados com os conjuntos de imagens anteriores. Nos casos do Tigre e do Fleximap, apenas estão na tabela os dados para os testes realizados com as imagens das duas câmaras em simultâneo.

Tabela 6.3: *Informação sobre o processo de reconstrução para todos os conjuntos de imagens*

Conjunto de imagens	**Rocgeo**	**Tigre**	**Fleximap**	**Pelican**
Imagens	28/28	63/64	88/88	55/55
Resolução	4000×2672	1278×958	1032×778	752×480
Pontos SIFT	288 300	1 850 974	912 552	469 762
BA (s)	341.940	644.630	2 722.44	2 478.670
Pontos 3D	34 094	20 673	41 037	37 735

6.3 Geo-Bundler

O Geo-Bundler é uma adaptação do Bundler para estimar a estrutura do espaço e o movimento, utilizando um sequência de imagens em conjunto com a informação da posição, fornecida por sensores como IMU/GPS. As modificações do Geo-Bundler evitam a estimação inicial da posição das câmaras e utilizam a informação de sensores, para inicializar os parâmetros extrínsecos das câmaras (R e t). Com esta modificação, espera-se que o tempo de estimação dos parâmetros das câmaras no BA seja inferior, considerando que as câmaras são inicializadas corretamente.

A informação com os parâmetros extrínsecos deve ser colocada num ficheiro para cada imagem, com o mesmo nome e extensão '.pos'. A estrutura do ficheiro é a seguinte:

$<x>$ $<y>$ $<z>$ [posição da câmara (vetor t)]
$<r_1>$ ... $<r_9>$ [orientação da câmara (matriz R)]

Sabendo esta informação, a nuvem de pontos será gerada em relação a um sistema de coordenadas e, sem a ambiguidade do fator de escala.

Para analisar o impacto destas modificações, realizaram-se dois testes com o Bundler e o Geo-Bundler, para o conjunto de imagens Rocgeo. Este é o conjunto de imagens de exemplo do Geo-Bundler, e contém um ficheiro com a informação acerca da posição da câmara para cada imagem. Os resultados dos testes encontram-se na Tabela 6.4. Analisando os resultados da Tabela 6.4, verifica-se que o tempo de estimação dos parâmetros das câmaras aumentou para o caso em que se utiliza

Tabela 6.4: *Comparação dos resultados dos testes com os parâmetros intrínsecos e extrínsecos*

Projeto	**Bundler**	**Geo-Bundler**
Imagens	28/28	28/28
Resolução	4000×2672	4000×2672
Pontos SIFT	288 300	288 300
BA (s)	341.940	493.890
Pontos 3D	34 094	33 953

os dados do IMU para inicializar os parâmetros das câmaras. Estes resultados não correspondem ao esperado (redução do tempo do BA) e por isso, procedeu-se à análise dos dados fornecidos para inicialização e o seu impacto no processo de estimação.

Analisando os parâmetros extrínsecos fornecidos, observou-se que a matriz de rotação era igual para todas as imagens e os seus valores consideravam que a orientação da câmara, no momento de aquisição, era plana. Esta consideração da orientação como plana, não se verifica na realidade e o seu efeito traduz-se na triangulação dos pontos. No primeiro par de imagens, os pontos em comum são adicionados ao processo de reconstrução, se o erro de reprojeção da triangulação for reduzido. No caso em que o Bundler estima os parâmetros iniciais das câmaras, os pontos são adicionados se o erro de reprojeção for inferior a 4 *pixels*. No caso do Geo-Bundler, com os dados do IMU, observou-se que o erro de reprojeção para a maioria dos pontos era elevado e, para contornar esta situação o limite foi alterado para 300 *pixels*. O facto do erro de reprojeção inicial ser elevado provoca um aumento significativo no processo de estimação dos parâmetros das câmaras.

Na Figura 6.12 apresentam-se os registos de tempo do BA para o Bundler e Geo-Bundler, assim como o número de pontos, consoante as imagens são adicionadas ao processo de estimação.

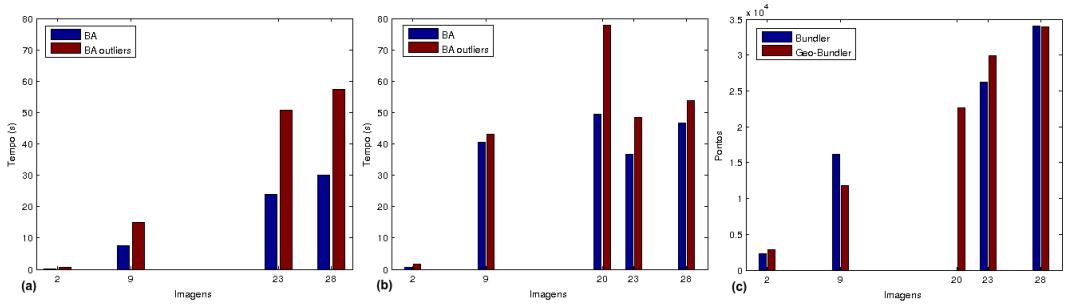

Figura 6.12: *Comparação do Bundler e Geo-Bundler para o conjunto de imagens Rocgeo. a) Registo do tempo do BA para o Bundler. b) Registo do tempo do BA para o Geo-Bundler. c) Registo no número de pontos para o Bundler e Geo-Bundler.*

Comparando a Figura 6.12 (a) e (b), observa-se que o Geo-Bundler despende mais tempo durante o processo de estimação dos parâmetros da câmaras.

6.4 SIFT e SiftGPU

A extração de pontos de interesse é um passo essencial para o processo de estimação da estrutura e do movimento. Sabendo os pontos de interesse, procede-se à correspondência dos pontos entre as imagens. Este processo (*matching*) é um dos passos que consome mais tempo de processamento. Por isso, uma optimização nesta tarefa permitirá reduzir consideravelmente o tempo da reconstrução 3D.

A versão original do Bundler utiliza o SIFT para extrair os pontos de interesse. O processo de *matching* é realizado para todas as imagens através de uma aplicação que utiliza a biblioteca ANN.

O projeto Geo-Bundler substituiu o SIFT pelo SiftGPU. O SiftGPU é uma versão modificada do SIFT, para usufruir da potencialidade da placa gráfica do computador (GPU). Um elevado número de imagens aumenta consideravelmente o esforço computacional do processo de *matching*, mas se as imagens contiverem muitos pontos em comum, aumenta a probabilidade do sucesso da reconstrução. A biblioteca do SiftGPU além de detetar pontos SIFT, também permite efetuar o *matching* usando a GPU.

No caso de conjuntos de imagens com resoluções diferentes, o facto de ter maior resolução não implica melhores resultados. As imagens com resolução elevada mostram pequenos detalhes do espaço, que não são observados com imagens de resolução inferior, e por vezes este tipo de detalhe em vez de aumentar a qualidade da reconstrução, reduz o desempenho devido à introdução de *outliers*. A versão do SIFT utilizada no Bundler, é uma versão de demonstração e tem um limite máximo de 1800 *pixels* na resolução, em qualquer dimensão. No caso do SiftGPU esta limitação da dimensão de resolução é configurável pelo utilizador. Contundo, segundos os testes realizados em [12], a resolução VGA (640×480) apresenta o melhor compromisso entre qualidade e velocidade na reconstrução.

Para comparar o desempenho do SIFT e do SiftGPU, realizou-se um teste com o mesmo conjunto de imagens para ambas as versões. O conjunto de imagens contém 55 imagens com resolução 720×480. Para estes testes utilizou-se um processador i7-2670QM e uma GPU AMD Radeon HD 7470M. A Figura 6.13 (a) e (b) apresentam a comparação do número de pontos SIFT extraídos para cada imagem e o tempo despendido no processo de *matching* entre as imagens, respetivamente. Analisando a Figura 6.13 (a), verifica-se que o SiftGPU extraí aproximadamente mais do dobro

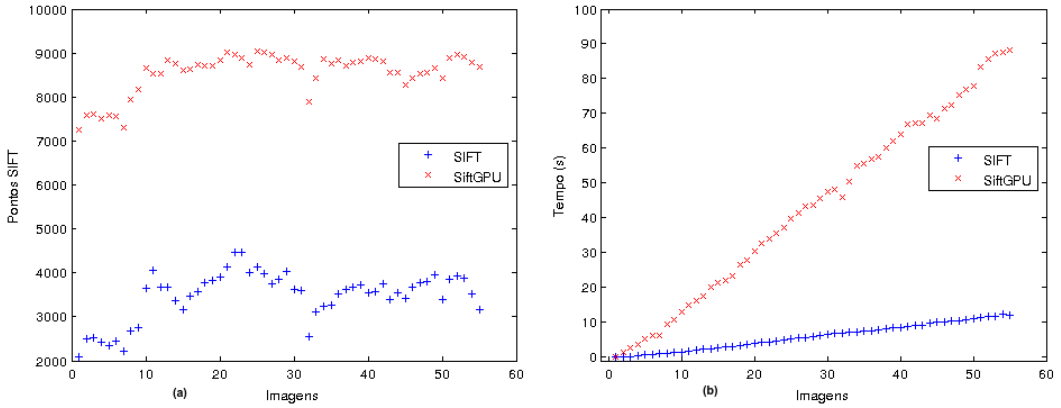

Figura 6.13: *Comparação entre SIFT e SiftGPU. a) Extração de pontos SIFT para cada imagem. b) Matching dos pontos SIFT entre as imagens.*

do número de pontos que a aplicação do SIFT no processador. Esta diferença no número de pontos reflete-e no processo de *matching*, como se pode observar na Figura 6.13 (b). Apesar do número de pontos com o SiftGPU ser muito elevado, esperava-se verificar um desempenho mais rápido do que o observado.

Para realizar uma comparação mais equilibrada, parametrizou-se o SiftGPU com um limite máximo na deteção do número de pontos. Com este teste espera-se, que no mínimo, o desempenho do SiftGPU no processo de *matching* seja aproximadamente equivalente ao desempenho da aplicação desenvolvida usando a biblioteca ANN. Os resultados do teste podem ser observados na Figura 6.14. Analisando os resultados

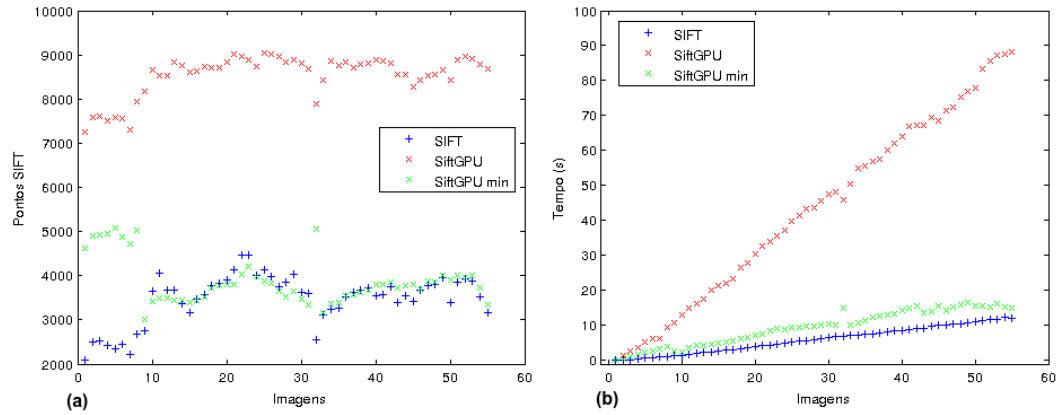

Figura 6.14: *Comparação entre SIFT e SiftGPU com limite máximo da deteção de pontos. a) Extração de pontos SIFT para cada imagem. b) Matching dos pontos SIFT entre as imagens.*

obtidos, verifica-se o impacto no número de pontos detectados no processo de *matching*. Para um número de pontos detetados aproximadamente equivalente, o tempo

no processo de *matching* no SiftGPU é relativamente semelhante ao tempo da aplicação criada para o SIFT no CPU, no entanto apresenta um pior desempenho. O facto da aplicação de *matching*, para o SIFT no CPU, utilizar funções da biblioteca ANN, pode ter um impacto significativo no desempenho deste processo.

Como era esperado um desempenho mais rápido com a GPU, e para despitar problemas de configuração do *driver* ou até alguma limitação da GPU utilizada nestes testes, decidiu-se repetir o teste com uma GPU Nvidia com suporte ao CUDA. O CUDA é um modelo de programação exclusivo da Nvidia, que concede aos programadores mais controlo direto da placa gráfica. Para placas ATI e Nvidia sem suporte CUDA, o SiftGPU utiliza o GLSL, que consiste numa extensão da linguagem de programação OpenGL.

Os testes anteriores foram repetidos com uma GPU Nvidia com suporte ao CUDA, e os resultados obtidos encontram-se na Figura 6.15. Os resultados são comparados com o SIFT para o CPU e com a aplicação de *matching* com recurso à biblioteca ANN. Analisando estes resultados, confirmam-se as vantagens da GPU

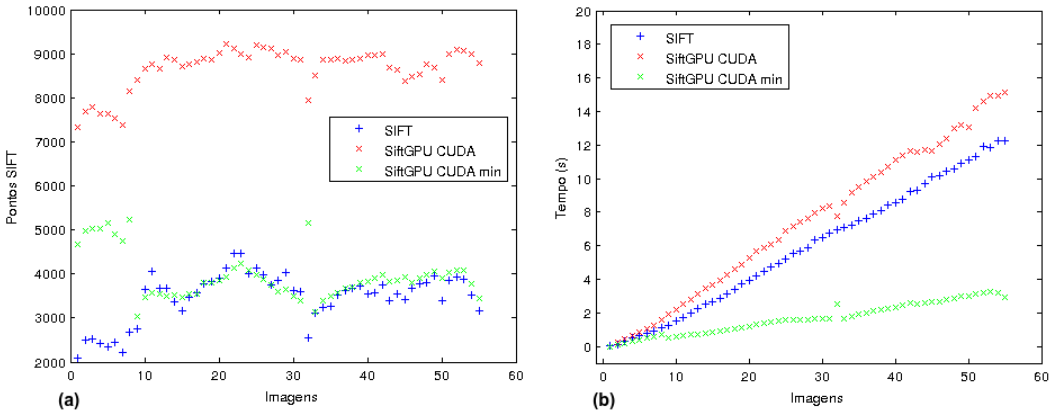

Figura 6.15: *Comparação entre SIFT e SiftGPU com um placa Nvidia. a) Extração de pontos SIFT para cada imagem. b) Matching dos pontos SIFT entre as imagens.*

Nvidia com CUDA, no processo de deteção e de *matching*. Para o mesmo número de pontos, o *matching* na GPU despende, aproximadamente, menos de metade do tempo que a aplicação para o SIFT no CPU. E para, aproximadamente, o dobro do número de pontos detetados, a GPU despende quase o mesmo tempo que a aplicação do CPU com metade do número de pontos.

Durante os testes anteriores, registou-se o tempo despendido na deteção de pontos SIFT. Na Tabela 6.5 encontram-se os valores do tempo total do processo de deteção de pontos para o SIFT no CPU, SiftGPU com placa gráfica ATI e SiftGPU com placa gráfica Nvidia. Analisando os resultados obtidos, verifica-se que o SiftGPU apresenta

Tabela 6.5: *Tempo de processamento na deteção de pontos SIFT*

Deteção	Pontos SIFT	Tempo (s)
SIFT CPU	191 067	100
SiftGPU ATI	469 762	18.11
SiftGPU ATI *	213 005	14.52
SiftGPU CUDA	475 964	10.07
SiftGPU CUDA *	216 951	6.81

* com limite máximo nos pontos SIFT

uma melhoria significativa no desempenho do processo de detecção de pontos. Com o SiftGPU são detetados, aproximadamente, o dobro do número de pontos e cinco vezes mais rápido, usando a placa gráfica ATI. Com a GPU da Nvidia obtém-se o melhor desempenho em todas as situações.

Os resultados destes testes permitem verificar as vantagens da utilização da GPU, no processo de deteção e *matching* dos pontos SIFT. O facto de utilizar a GPU para processos de elevado esforço computacional, é vantajoso para aplicações multi-tarefa, porque o processador fica disponível para realizar outras tarefas.

Para verificar o impacto das duas versões do SIFT no processo de reconstrução, procedeu-se à execução do Bundler para ambos os casos. A GPU utilizada neste comparação é a ATI e sem a restrição no número de pontos SIFT detetados. O objetivo consiste em comparar as duas situações, sem parametrização por parte do utilizador. A comparação dos resultados obtidos pode ser observada na Tabela 6.6. O SiftGPU detecta um número elevado de pontos comparativamente ao SIFT no

Tabela 6.6: *Comparação do desempenho do Bundler com o SIFT e o SiftGPU*

Aplicação	**SIFT**	**SiftGPU**
Pontos SIFT	191 067	469 762
BA (s)	1 018.280	1 870.847
Pontos 3D	13 661	34 639

CPU, e esta diferença significativa reflete-se na quantidade de pontos obtidos no processo de reconstrução 3D, assim como no tempo despendido durante o processo de estimação dos parâmetros das câmaras e dos pontos 3D (BA).

7

Análise da Caracterização do Bundler e Definição de Propostas

Nesta dissertação pretende-se explorar uma ferramenta SfM, o Bundler, para aplicações robóticas. Durante a análise do Bundler, teve-se em consideração dois tipos de aplicações diferentes: aplicações em tempo real e/ou pós-processamento.

O Bundler foi desenvolvido para enormes conjuntos desordenados de imagens. A sequência de imagens utilizada no processo de estimação, é definida pelo Bundler através de uma seleção efetuada com base num número mínimo de pontos comuns entre as imagens. As imagens que não respeitarem este critério, serão descartadas do processo de estimação. Para aplicações de pós-processamento, a seleção das melhores imagens é uma mais valia para o sucesso da reconstrução 3D, pelo facto de reduzir o número de iterações do BA durante a estimação dos parâmetros das câmaras e, consequentemente melhorar a qualidade da nuvem de pontos.

Numa aplicação em tempo real, por exemplo como a percepção do espaço para evitar obstáculos, a abordagem de selecção das imagens não tem aplicabilidade. Neste contexto, as imagens são adquiridas sequencialmente e de preferência a uma frequência elevada, por isso pode-se garantir a existência de pontos em comum entre imagens consecutivas. O factor determinante numa aplicação em tempo real é o tempo de processamento. Analisando o desempenho do Bundler, os principais blocos de elevado processamento são a correspondência de pontos SIFT entre as imagens (*matching*) e a estimação dos parâmetros das câmaras e das posições dos pontos 3D,

o BA.

Em relação ao *matching* propõe-se implementação de um filtro baseado num limite de distância entre câmaras. O objetivo consite em evitar o *matching* entre câmaras distantes, reduzindo o tempo de processamento e a introdução de falsos *matches* no processo de reconstrução.

O BA é um processo de estimação iterativo com base na redução do erro de reprojeção. Se os parâmetros das câmaras forem devidamente inicializados, consequentemente diminuirá o número de iterações de estimação, para alcançar o menor erro de reprojeção. Para alcançar esta optimização apresentam-se duas propostas, que têm como objetivo inicializar as câmaras com a maior precisão possível. A primeira aplica-se para sistemas autónomos que utilizem a mesma câmara para aquisição de imagens e sem focagem automática, ou seja, com uma distância focal fixa. Considerando estes requisitos, os parâmetros intrínsecos podem ser obtidos, uma única vez, por um processo prévio de calibração. Sabendo os parâmetros intrínsecos, pretende-se que sejam fornecidos ao Bundler e evitar que este proceda a estimação dos mesmos. O Bundler estima três parâmetros intrínsecos, a distância focal e dois coeficientes de distorção radial. Por isso, se o Bundler não precisar de estimar estes parâmetros, espera-se conseguir reduzir consideravelmente o tempo do BA e melhorar o processo de reconstrução, considerando que são fornecidos parâmetros intrínsecos corretos.

A segunda proposta, implica a utilização de dados sensoriais de IMU/GPS para inicializar os parâmetros extrínsecos das câmaras (posição e orientação). Através da utilização destes dados, também se pretende evitar o desvio da estimação, devido à acumulação de erros ao longo do processo de reconstrução. Uma das vantagens da utilização do IMU/GPS, é facto de os resultados serem gerados em relação a um sistema de coordenadas, o que não acontece no Bundler, em que as posições são relativas e com ambiguidade na escala.

Portanto, sabendo os parâmetros intrínsecos e extrínsecos das câmaras, no instante de aquisição, encontra-se reunida a informação necessária para inicializar as câmaras.

É importante referir que o Bundler foi desenvolvido para a reconstrução 3D, a partir de enormes conjuntos desordenados de imagens. Portanto, para considerar a possibilidade de utilizar o Bundler numa aplicação robótica, é necessário modificar a abordagem atual, para uma abordagem sequencial e ordenada, de acordo com o processo de aquisição.

Nesta dissertação pretende-se adaptar o Bundler com base nos seguintes tópicos:

- fornecer os parâmetros intrínsecos obtidos através de uma calibração realizada previamente;

- inicializar os parâmetros extrínsecos das câmaras, no instante de aquisição, através de informação sensorial de GPS/IMU;
- alterar a abordagem do Bundler para o processamento sequencial de um conjunto ordenado de imagens;
- aplicar um filtro baseado num limite de distância entre as câmaras, no processo de *matching*.

8

Implementação e Resultados

Os resultados apresentados referem-se às alterações efetuadas no projeto do Bundler, para adaptar às necessidades das aplicações robóticas. Foram realizadas duas alterações no Bundler. A primeira consiste numa correcção do projeto para a leitura dos parâmetros intrínsecos das câmaras. Esta opção tem aplicação para conjuntos de imagens que utilizam a mesma câmara e, quando os parâmetros de calibração são obtidos previamente. A segunda alteração, implicou a eliminação do critério de seleção de imagens do BA, para um abordagem sequencial e ordenada, em que é adicionada apenas uma imagem em cada passo.

Em relação à inicialização dos parâmetros extrínsecos das câmaras, podem-se utilizar as modificações que o projeto Geo-Bundler aplicou ao Bundler, para integrar os dados de sensores IMU/GPS como estimação inicial para a posição e orientação das câmaras.

8.1 Parâmetros intrínsecos das câmaras

Em geral, os robôs contém câmaras posicionadas numa estrutura rígida e fixa. Nestes casos, efetuando uma correta calibração, pretende-se fornecer os parâmetros intrínsecos ao Bundler, no sentido de reduzir o tempo de estimação. Segundo o *Frequently Asked Questions* (FAQ) do Bundler, existe uma opção escondida para fornecer os parâmetros intrínsecos através de um ficheiro segundo uma determinada estrutura. O problema é que esta opção explora partes de código que não foi testado.

Considerando que o Bundler foi concebido para realizar a estimação de todos os parâmetros, pressupõe-se que por este motivo, a opção de fornecer os intrínsecos foi irrelevante para o seu desenvolvimento. No entanto, para o contexto robótico, esta opção é uma mais valia.

Para fornecer os parâmetros intrínsecos, pode-se adicionar a opção "intrinsics" ao ficheiro de opções do Bundler (options.txt). A utilização desta opção implica a remoção das opções associadas à estimação dos parâmetros intrínsecos, devido à incompatibilidade. Juntamente com esta opção, é necessário colocar o nome do ficheiro que contém os parâmetros intrínsecos (–intrinsics <file_name>). O formato do ficheiro dos parâmetros intrínsecos, é compatível com o formato dos resultados da ferramenta de calibração do Bouguet [40]. A estrutura do ficheiro é a seguinte:

<número de conjuntos de parâmetros de calibração>

<parâmetros 1>

...

<parâmetros N>

Estrutura dos parâmetros:

<matriz K 3x3 com os parâmetros intrínsecos>

<5 coeficientes de distorção radial>

Se todas as câmaras têm os mesmos parâmetros, então basta apenas passar um conjunto de parâmetros intrínsecos. Se as câmaras têm parâmetros diferentes, o Bundler efetua a atribuição dos parâmetros à respetiva câmara através da comparação das distâncias focais fornecidas pelo ficheiro dos intrínsecos e pelo ficheiro com a lista de imagens.

O valor do ponto central deve ser definido, tendo em conta que a origem do sistema de coordenadas do plano da imagem corresponde ao centro da imagem, o x é positivo para a direita e o y é positivo para cima. O ponto central de acordo com o referencial do Bundler, pode ser determinado através das Equações 8.1 e 8.2.

$$C'_x = C_x - \frac{H}{2} \tag{8.1}$$

$$C'_y = \frac{V}{2} - C_y \tag{8.2}$$

Onde C'_x e C'_y são as coordenadas do ponto central de acordo com o referencial do plano da imagem do Bundler, H e V correspondem à resolução horizontal e vertical

da imagem e, C_x e C_y correspondem às coordenadas do ponto central obtidas da calibração.

Então, sabendo os parâmetros de calibração da câmara para um determinado conjunto de imagens, realizaram-se dois testes com o Bundler: o primeiro, sem os parâmetros intrínsecos e o segundo, fornecendo o ficheiro com os parâmetros intrínsecos.

Para o primeiro teste, no caso em que se fornece o ficheiro de calibração intrínseca da câmara, espera-se que o Bundler não proceda à estimação destes parâmetros e, por isso estes deveriam estar registados no ficheiro de resultados finais. No entanto, isto não se verificou, o Bundler apenas utilizava os parâmetros do ficheiro fornecido para o primeiro par de imagens, e para as outras câmaras a reconstrução falhava. Analisando estes resultados, confirma-se que o Bundler não estava corretamente desenvolvido para a utilização de parâmetros intrínsecos, fornecidos pelo utilizador.
Para resolver esta limitação do Bundler, procedeu-se à análise do código do projeto, para perceber a fluxo do programa e tentar corrigir o problema. Depois de uma análise do projeto, efetuaram-se algumas alterações para corrigir a limitação da leitura dos parâmetros intrínsecos.

O primeiro teste efetuou-se com a configuração de opções *standard*, em que o Bundler estima todos os parâmetros das câmaras. Este teste serve como referência de comparação após efetuadas as alterações no Bundler, para a leitura adequada do ficheiro de calibração intrínseca.

No segundo teste, os parâmetros intrínsecos são fornecidos num ficheiro usando a opção do Bundler para esse efeito. Os resultados obtidos estavam de acordo com o esperado, ou seja, os parâmetros intrínsecos de todas as câmaras registados no ficheiro de resultados, eram exactamente iguais aos parâmetros do ficheiro de calibração.

Os testes foram realizados com um conjunto de 55 imagens adquiridas pelo Pelican com resolução 752×480. Na Tabela 8.1 encontra-se uma comparação dos testes, relativamente ao número de imagens registadas, número de pontos 3D reconstruídos e o tempo total de processamento do BA. Com esta comparação pretende-se confir-

Tabela 8.1: *Comparação do Bundler com e sem calibração*

Teste	**Sem calibração**	**Com calibração**
Imagens	55/55	55/55
Resolução	752×480	752×480
Pontos SIFT	469 762	469 762
BA (s)	2 478.670	1 057.580
Pontos 3D	37 735	30 461

mar que, fornecendo previamente os dados de calibração ao Bundler, o processo de estimação será mais rápido devido à correcta inicialização dos parâmetros intrínsecos. Analisando o tempo do BA da Tabela 8.1, confirma-se a especulação anterior.

A Figura 8.1 apresenta o tempo despendido no BA e o número de pontos reconstruídos até ao momento, para o caso sem intrínsecos e com intrínsecos. Efetuando

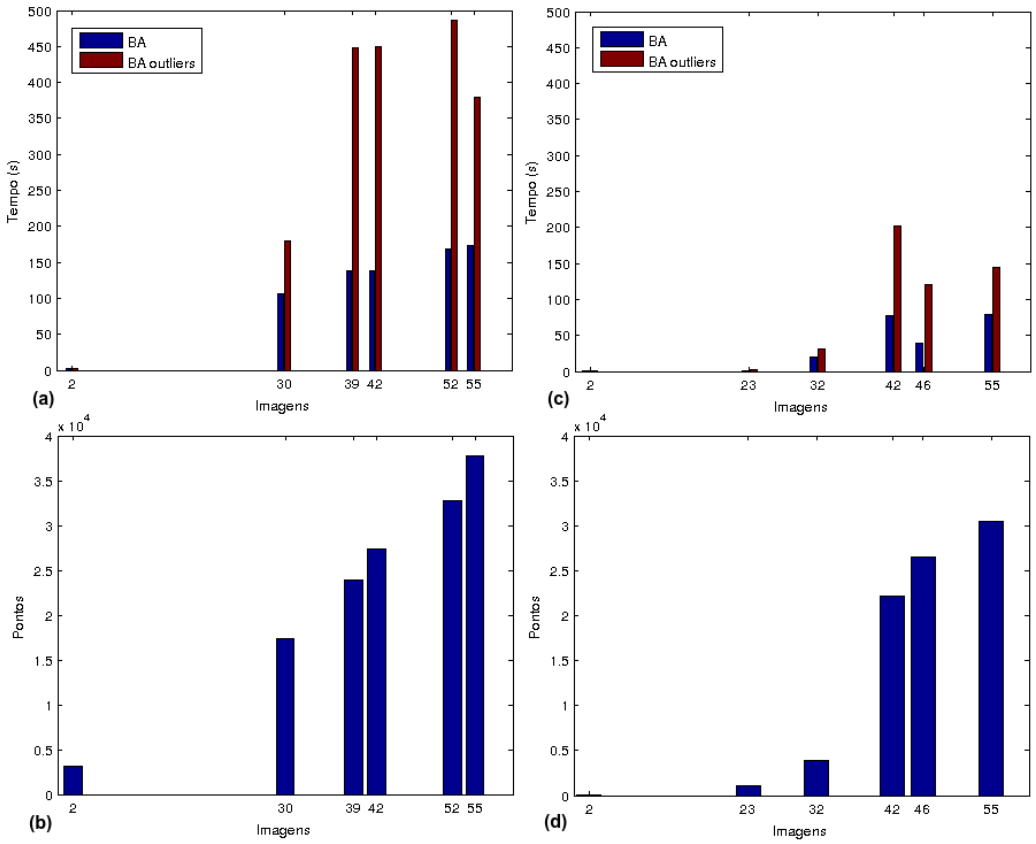

Figura 8.1: *BA e número de pontos 3D para o conjunto de imagens do Pelican. a) e b) Registo do tempo de processamento do BA e número de pontos 3D, para a situação de estimação dos parâmetros intrínsecos. b) Registo do tempo de processamento do BA e número de pontos 3D, para a situação em que os parâmetros intrínsecos não são estimados.*

a comparação entre as duas situações, verifica-se que na situação em que o Bundler estima os intrínsecos, uma grande parte do tempo é despendido em novas estimações após remover os *outliers*. Por isso, presume-se que a introdução dos intrínsecos corretos, evita a adição de *outliers* e, consequentemente permite alcançar mais rápido o menor erro de reprojeção.

Os dados de calibração também têm um impacto perceptível na nuvem de pontos gerada pelo Bundler. Comparando as duas nuvens de pontos, sem calibração e com calibração, verifica-se que a informação de calibração aplica uma melhor correcção

da distorção da câmara, Figura 8.2. Analisando a figura anterior, é perceptível a

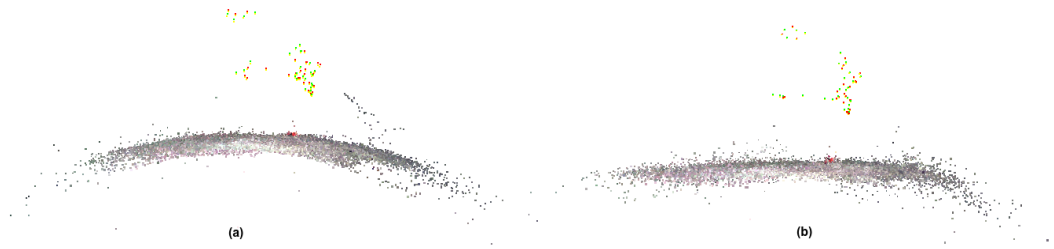

Figura 8.2: *Nuvens de pontos do Bundler com e sem dados prévios da calibração interna. a) O Bundler estima os parâmetros intrínsecos. b) Os parâmetros intrínsecos são fornecidos previamente ao Bundler*

diferença da distorção no conjunto de pontos que representam a superfície do solo. Na Figura 8.1 (a) o piso apresenta uma ligeira curvatura, e em (b) observa-se um piso plano. Portanto, para este caso, a introdução dos parâmetros intrínsecos apresenta vantagens para o tempo do BA e para a distorção da câmara.

8.2 Parâmetros intrínsecos e extrínsecos das câmaras

As modificações do projeto Geo-Bundler permitem inicializar os parâmetros das câmaras, a partir de informação sensorial obtida com sensores IMU/GPS. Esta informação deve ser o mais exacta possível, tendo em consideração os resultados obtidos com o conjunto de dados Rocgeo. Quando se realizou o teste para este conjunto, verificou-se que uma incorreta inicialização introduz um efeito negativo durante a triangulação dos pontos, aumentado consideravelmente o erro de reprojeção.

Realizaram-se alguns testes utilizando informação sensorial com ruído e como consequência obtiveram-se resultados desastrosos. Quando o erro de reprojeção durante a triangulação inicial era elevado, verificava-se que os pontos não eram adicionados para o processo de optimização e, por isso não era possível efetuar a estimação dos parâmetros das câmaras. Em alguns casos, observou-se que após a estimação dos parâmetros das câmaras para o primeiro par, o erro de reprojeção dos pontos era tão elevado, que estes eram considerados como *outliers* e, por isso eram removidos até não restarem quaisquer pontos e, consequentemente a reconstrução terminava. Então para evitar elevados erros de reprojeção devidos a uma incorreta inicialização, utilizou-se um conjunto de imagens obtidas com o Fleximap porque este sistema contém um IMU de elevada qualidade.

Realizaram-se três testes com 44 imagens do Fleximap com resolução 1032×778:

- Teste A: estimação dos parâmetros extrínsecos e intrínsecos das câmaras.

- Teste B: estimação apenas dos parâmetros extrínsecos das câmaras. Os parâmetros intrínsecos são obtidos previamente por um processo de calibração.
- Teste C: estimação apenas dos parâmetros extrínsecos das câmaras, usando como estimação inicial os dados do IMU. Os parâmetros intrínsecos são obtidos previamente por um processo de calibração.

Os resultados dos testes encontram-se na Tabela 8.2. Observando os resultados da

Tabela 8.2: *Resultados do Bundler através da inicialização externa dos parâmetros das câmaras*

Teste	**A**	**B**	**C**
Imagens	44/44	44/44	43/44
Resolução	1032×778	1032×778	1032×778
Pontos SIFT	460 736	460 736	460 736
BA (s)	744.570	637.460	884.110
Pontos 3D	17 852	18 058	17 101

Tabela 8.2 verifica-se que a utilização dos intrínsecos de calibração reduz o tempo de estimação dos parâmetros das câmaras, no entanto, o mesmo não se verifica quando se utilizam os parâmetros extrínsecos provenientes do IMU. Através de uma análise do processo de estimação no Teste C, verificou-se que o erro de reprojeção da triangulação dos pontos era superior em comparação com o método de estimação nos Testes A e B. Supõe-se que esta diferença, deve-se ao ruído nos parâmetros extrínsecos obtidos do IMU. Devido à falta de tempo e de novos conjuntos de dados, não foi possível realizar mais testes para diagnosticar esta diferença de resultados.

8.3 Modificação da abordagem do Bundler

O Bundler foi desenvolvido com uma abordagem para enormes conjuntos desordenados de imagens. A sequência de imagens para o processo de optimização (BA) é definida segundo um critério de seleção. Na primeira optimização, o Bundler procede à estimação dos parâmetros do melhor par de imagens. Este par de imagens deve ter um número elevado de pontos em comum e uma *baseline* grande, para que a localização dos pontos 3D observados seja bem condicionada. As restantes imagens são adicionas ao BA, se observarem os pontos reconstruídos até ao momento. A sequência das imagens é definida pelo número de *matches* com os pontos reconstruídos.

No caso de uma aplicação robótica em tempo real, a aquisição de imagens é sequencial e com uma elevada frequência, por isso a probabilidade de existirem pontos

em comum entre imagens consecutivas, é elevada. A abordagem atual do Bundler não se enquadra neste contexto e por isso, pretende-se alterar para uma abordagem sequencial e ordenada, de acordo com a aquisição das imagens. Com esta alteração não é esperado melhoria no desempenho do Bundler, porque será adicionada apenas uma imagem de cada vez ao BA.

Em relação ao tempo de processamento, para comparar a modificação da abordagem sequencial e ordenada, realizaram-se dois testes para o mesmo conjunto de imagens. O primeiro teste efetuou-se com o Bundler no método II, em que adiciona uma imagem de cada vez ao BA. O segundo teste realizou-se com a modificação da abordagem do Bundler, em que também se adiciona apenas uma imagem por iteração. Os resultados obtidos, certamente serão diferentes, porque no primeiro caso as imagens são selecionadas de acordo com o número de pontos em comum. No segundo caso, as imagens são utilizadas conforme a sequência de aquisição. O objetivo desta comparação, consiste em verificar se esta modificação apresenta resultados equivalentes à abordagem atual do Bundler, em relação ao tempo de processamento. Na Tabela 8.3 encontram-se os resultados dos dois testes. Foi utilizado um conjunto de 55 imagens adquiridas pelo Pelican, com resolução 752×480. Este conjunto

Tabela 8.3: *Resultados da abordagem sequencial e ordenada*

Método	II	III*
Imagens	55/55	55/55
Resolução	752×480	752×480
Pontos SIFT	469 762	469 762
BA (s)	1 643.135	1 360.213
Pontos 3D	37 831	38 637

* Método sequencial e ordenado

de imagens encontra-se ordenado de acordo com o tempo de aquisição. Como as imagens consecutivas tem bastantes pontos em comum, obteve-se uma reconstrução tridimensional com sucesso e com um tempo inferior.

A modificação da abordagem é apenas um dos primeiros passos de adaptação do Bundler para um aplicação em tempo real, no entanto ainda ficam pendentes bastantes alterações. Neste momento, também é possível fornecer os parâmetros intrínsecos previamente obtidos e utilizar informação de sensores GPS/IMU para inicializar os parâmetros extrínsecos das câmaras.

Os processos de deteção e *matching* de pontos, são realizados como um bloco independente do Bundler. Segundo uma lista de imagens, procede-se à deteção de pontos SIFT e procura-se a correspondência de pontos entre todas as imagens. No

caso de uma aplicação em tempo real, não é necessário efetuar o *matching* para todas as imagens adquiridas até ao momento. Para optimizar este processo, aplicou-se um filtro baseado nas posições das câmaras, ou seja, o *matching* entre imagens adquiridas em posições relativamente distantes, pode ser evitado segundo um limite configurável. Com introdução deste limite pretende-se reduzir o tempo processamento, evitando o *matching* entre imagens que não visualizam o mesmo espaço e, consequentemente melhorar o processo de reconstrução, assim como reduzir o tempo de estimação dos parâmetros das câmaras, devido à redução falsos pontos correspondentes.

Em relação à velocidade do processos de deteção e *matching* entre as imagens, verificou-se que a utilização de uma GPU Nvidia com CUDA, oferece um desempenho significativamente mais rápido em relação aos processos equivalentes, executados pelo CPU.

Existem outros blocos de processamento que também podem ser optimizados, como por exemplo, uns dos passos iniciais do Bundler é a determinação da relação geometria entre todas as imagens. Esta tarefa consome bastante tempo de processamento consoante o número de imagens, e uma possível sugestão seria a aplicação de um filtro, baseado num limite de distância entre imagens, equivalente ao aplicado no SIFT.

9

Conclusões e Propostas para Trabalho Futuro

O objetivo desta dissertação consistia em explorar e analisar o projeto Bundler para a estimação, em simultâneo, da estrutura e do movimento a partir de uma sequência de imagens.

Na análise do Bundler evidenciaram-se os blocos que exigem maior esforço computacional, o *matching* de pontos SIFT entre as imagens e o processo iterativo de estimação dos parâmetros das câmaras com base na diminuição erro de reprojeção, o *Bundle Adjustment*.

Para o processo de *matching* realizaram-se testes comparativos entre a deteção e *matching* de pontos SIFT no processador com as tarefas equivalentes utilizando uma adaptação do SIFT para a GPU, conhecido como SiftGPU. Os resultados mostraram que o SiftGPU apresenta melhor desempenho do que o processador, em ambas as tarefas, se for utilizada uma GPU Nvidia com CUDA. No sentido de reduzir o tempo do *matching*, aplicou-se um filtro com base num limite de distância, em que evita o *matching* se as imagens foram adquiridas em posições distantes. Com este filtro pretende-se evitar processamento desnecessário e também a evitar a introdução de falsos pontos, no processo de reconstrução.

No *Bundle Adjustment* exploraram-se duas propostas para tentar reduzir o tempo de estimação dos parâmetros das câmaras. Estas propostas foram definidas com base na aplicação em sistemas autónomos. A primeira aplica-se para os casos em que se utiliza a mesma câmara com uma distância focal fixa. Considerando estes requisitos, os parâmetros intrínsecos podem ser obtidos previamente por um processo de

calibração, e posteriormente fornecidos ao Bundler. Os testes realizados mostraram que a utilização de parâmetros intrínsecos fixos reduz consideravelmente o tempo de estimação.

A segunda proposta consiste na utilização de informação proveniente de sensores IMU/GPS para inicializar os parâmetros extrínsecos das câmaras. O objetivo consistia em fornecer uma inicialização para as câmaras que permitisse, em conjunto com os parâmetros intrínsecos de calibração, a redução do tempo no BA. Com a utilização dos dados do IMU/GPS, os resultados dos parâmetros das câmaras e as posições do pontos 3D estão referenciados ao sistema de coordenadas usado na inicialização. Para longos conjuntos de imagens, os dados do IMU/GPS também permitem evitar desvios devido a erros cumulativos, no entanto, os resultados dos testes realizados mostraram que se estes dados contiverem ruído considerável, o erro de reprojeção torna-se elevado e consequentemente o processo de reconstrução falha.

O Bundler foi desenvolvido para estimar a estrutura e o movimento a partir de enormes conjuntos desordenados de imagens. No sentido de adaptar o Bundler para o contexto de uma aplicação num sistema robótico, procedeu-se à modificação da abordagem atual, para uma abordagem sequencial e ordenada de acordo com o processo de aquisição. Este é um pequeno passo para uma adaptação projeto de acordo com os requisitos de uma aplicação em tempo real.

Como propostas para futuro sugere-se a reestruturação do Bundler, adaptando os blocos do projeto de acordo com as exigências do tempo de processamento. Por exemplo, pode-se explorar o paralelismo através da utilização da biblioteca *Multicore Bundle Adjustment* [17] para a processo iterativo de estimação dos parâmetros das câmaras. Atualmente, a última versão do Bundler contempla uma nova biblioteca, denominada *Ceres Solver* [41], para resolver o problema do BA. Esta biblioteca é gratuita e foi desenvolvida para resolver problemas não lineares dos mínimos quadrados. A Google utiliza a *Ceres Solver* para resolver problemas de estimação da posição, reconstrução 3D, entre outros. Então como sugestões para o futuro, propõe-se a modificação do Bundler e exploração destas novas bibliotecas, no sentido de aumentar o desempenho do processo de reconstrução.

A

Manual de Utilizador para o Bundler

Para executar o Bundler no Linux, usando o SiftGPU, devem-se executar os seguintes passos:

1. Criar um ficheiro com a lista de imagens <list>.txt.

2. Executar o SiftGPU para deteção e *matching* de pontos entre as imagens.
   ./siftGPU <list>.txt <matches_table>.txt> <max_resolution_dimension> <device_number>

3. Executar o Bundler.
   ./bundler <list>.txt --options_file <options>.txt

   Se preferir executar o Geo-Bundler, a sintaxe é equivalente ao Bundler.
   ./GeoBundler <list>.txt --options_file <options>.txt

   Para o GeoBundler é necessário criar um ficheiro para cada imagem, com o respetivo nome da imagem e extensão *.pos (<image_name>.pos).

O projeto Bundler utiliza o SIFT e uma aplicação própria para o *matching* (ambos utilizam o CPU). Como alternativa, recomenda-se o SiftGPU devido à melhoria significativa no desempenho, para a deteção e *matching*.

A.1 Estrutura dos ficheiros

A estrutura do ficheiro <list>.txt é a seguinte:
<image_path> 0 <focal_distance>

Para o modo de estimação geral, os parâmetros do <options>.txt são os seguintes:
--match_table <matches_table>.txt
--output bundle.out
--output_all bundle_
--output_dir bundle
--variable_focal_length
--use_focal_estimate
--constrain_focal
--constrain_focal_weight <value>
--estimate_distortion
--run_bundle

Quando se fornece os parâmetros intrínsecos ao Bundler, deve-se utilizar a seguinte combinação de parâmetros no ficheiro <options>.txt:
--match_table <matches_table>.txt
--output bundle.out
--output_all bundle_
--output_dir bundle
--intrinsics <intrinsics_file>.txt
--run_bundle

A estrutura do ficheiro com os parâmetros de calibração é a seguinte:
<n>
<fx> 0 <cx> 0 <fy> <cy> 0 0 1
<k1> <k2> <k3> <k4> <k5>
Onde <n> indica o número de conjuntos dos parâmetros intrínsecos. Quando existem câmaras diferentes, podem-se definir conjuntos de parâmetros para cada câmara. A atribuição do conjunto à respetiva câmara é realizada pela comparação da distância focal deste ficheiro com a distância focal do ficheiro <list>.txt.
O <cx> e <cy> correspondem às coordenadas do ponto central, mas devem ser definidas considerando que a origem do sistema de coordenadas é o centro da imagem.

O ficheiro <image_name>.pos para o Geo-Bundler, contém a posição das câmaras e a matriz de rotação. A estrutura do ficheiro é a seguinte:
<x> <y> <z>
<r1> <r2> <r3> <r4> <r5> <r6> <r7> <r8> <r9>

A.2 Diagrama de blocos

Na Figura A.1 está representado um diagrama de blocos sobre o algoritmo SfM do Bundler. Este diagrama descreve de uma forma muito simples a sequência de operações efetuadas pelo Bundler.

Figura A.1: *Diagrama de blocos do Bundler*

A Figura A.2 apresenta uma comparação de uma perspetiva funcional entre os projetos Bundler e Geo-Bundler.

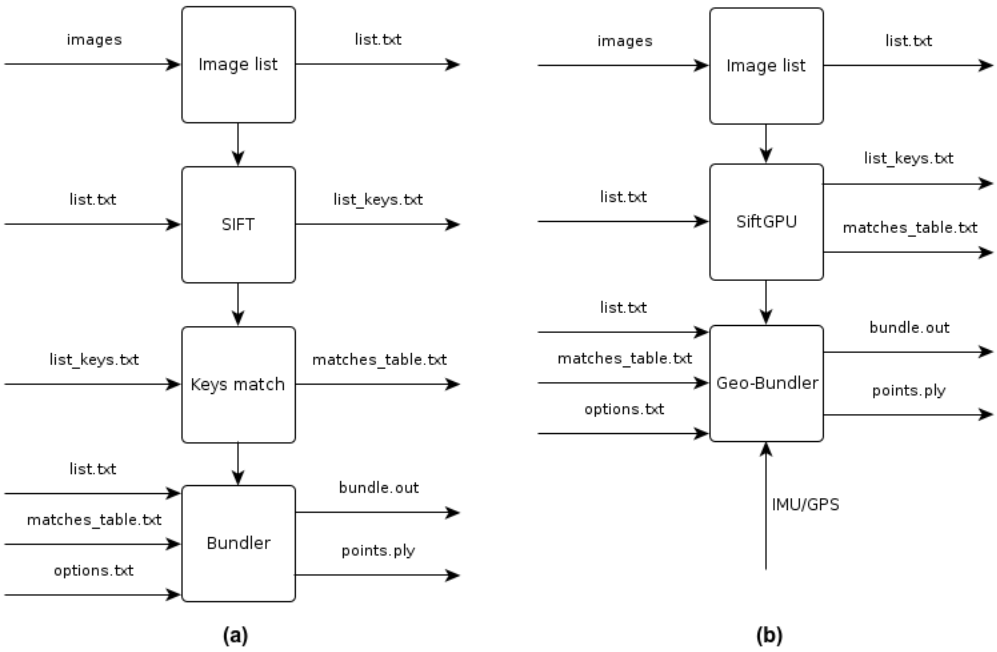

Figura A.2: *Comparação entre os projetos Bundler e Geo-Bundler. a) Projeto Bundler. b) Projeto Geo-Bundler*

Bibliografia

[1] A. Díaz, E. Caicedo, L. Paz, and P. Piniés, "A REAL TIME 6DOF VISUAL SLAM SYSTEM USING A MONOCULAR CAMERA," (2012).

[2] "ARC 3D Webservice,", `http://homes.esat.kuleuven.be/~visit3d/webservice/v2/index.php`, 2014.

[3] Autodesk, "123D Catch,", `http://www.123dapp.com/catch`, 2014.

[4] "2d3,", `http://www.2d3sensing.com/`, 2014.

[5] N. Snavely, "Bundler: Structure from Motion (SfM) for Unordered Image Collections,", `http://www.cs.cornell.edu/~snavely/bundler/`, 2014.

[6] Y. Furukawa and J. Ponce, "Accurate, Dense, and Robust Multi-View Stereopsis," (2008).

[7] S. Agarwal, N. Snavely, I. Simon, S. M. Seitz, and R. Szeliski, "Building Rome in a Day," (2009).

[8] N. Snavely, S. M. Seitz, and R. Szeliski, "Skeletal graphs for efficient structure from motion," (2008).

[9] S. Agarwal, N. Snavely, S. M. Seitz, and R. Szeliski, "Bundle Adjustment in the Large," (2010).

[10] D. Crandall, A. Owens, N. Snavely, and D. Huttenlocher, "Discrete-Continuous Optimization for Large-Scale Structure from Motion," (2011).

[11] K. Ni and F. Dellaert, "HyperSfM," (2012).

[12] Y.-F. Wang, "A Comparison Study of Five 3D Modeling Systems Based on the SfM Principles," (2011).

[13] N. Snavely, S. M. Seitz, and R. Szeliski, "Photo Tourism: Exploring Photo Collections in 3D," (2006).

[14] N. Snavely, S. M. Seitz, and R. Szeliski, "Modeling the World from Internet Photo Collections," (2007).

[15] C. Wu, "Towards Linear-time Incremental Structure From Motion," (2013).

[16] C. Wu, "VisualSFM: A Visual Structure from Motion System," (2011), `http://ccwu.me/vsfm/`.

[17] C. Wu, S. Agarwal, B. Curless, and S. M. Seitz, "Multicore Bundle Adjustment," (2011).

[18] C. Wu, "SiftGPU: A GPU implementation of Scale Invaraint Feature Transform (SIFT)," (2007), `http://cs.unc.edu/~ccwu/siftgpu`.

[19] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "MonoSLAM: Real-Time Single Camera SLAM," (2007).

[20] R. M. and, "Real-Time Structure from Motion for Monocular and Stereo Cameras," (2010).

[21] E. Mouragnon, M. Lhuillier, M. Dhome, F. Dekeyser, and P. Sayd, "Monocular Vision Based SLAM for Mobile Robots," (2006).

[22] S. Giannarou, Z. Zhang, and G.-Z. Yang, "Deformable Structure From Motion by Fusing Visual and Inertial Measurement Data," (2012).

[23] M. Pollefeys, J.-M. Frahm, F. Fraundorfer, C. Zach, C. Wu, B. Clipp, and D. Gallup, "Challenges in Wide-area Structure-from-motion," (2010).

[24] O. Kahler and J. Denzler, "Tracking and Reconstruction in a Combined Optimization Approach," (2012).

[25] M. Lhuillier, "Incremental Fusion of Structure-from-Motion and GPS Using Constrained Bundle Adjustments," (2012).

[26] A. Irschara, C. Hoppe, and H. Bischof, "Efficient Structure from Motion with Weak Position and Orientation Priors," (2011).

[27] C. Strecha, T. Pylvanainen, and P. Fua, "Dynamic and Scalable Large Scale Image Reconstruction," (2010).

[28] D. Nilosek, S. Sun, and C. Salvaggio, "Geo-Accurate Model Extraction from Three-Dimensional Image-Derived Point Clouds," (2012).

[29] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, second edition ed. (Cambridge University Press, 2003).

[30] M. Fishler and R. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," (1987).

[31] D. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," (2004).

[32] C. Harris and M. Stephens, "A combined corner and edge detector," (1988).

[33] M. Brown and D. G. Lowe, "Unsupervised 3D Object Recognition and Reconstruction in Unordered Datasets," (2005).

[34] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu, "An Optimal Algorithm for Approximate Nearest Neighbor Searching in Fixed Dimensions," (1998).

[35] J. Nocedal and S. J. Wright, *Numerical Optimization* (Springer, 1999).

[36] D. Nistér, "An efficient solution to the five-point relative pose problem," (2004).

[37] M. I. Lourakis and A. A. Argyros, "The design and implementation of a generic sparse bundle adjustment software package based on the Levenberg Marquardt algorithm," (2004).

[38] Y. Furukawa and J. Ponce, "Patch-based Multi-view Stereo Software (PMVS - Version 2),", `http://www.di.ens.fr/pmvs/`, 2014.

[39] Y. Furukawa, "Clustering Views for Multi-view Stereo (CMVS),", `http://www.di.ens.fr/cmvs/`, 2014.

[40] J.-Y. Bouguet, "Camera Calibration Toolbox for Matlab,", `http://www.vision.caltech.edu/bouguetj/calib_doc/`, 2014.

[41] S. Agarwal, K. Mierle, and Others, "Ceres Solver,", `http://ceres-solver.org`.