### Registro Global de Nuvens de Pontos RGB-D em Tempo Real Usando Fluxo Óptico e Marcadores

#### Bruno Marques Ferreira da Silva

Orientador: Prof. Dr. Luiz Marcos Garcia Gonçalves

**Tese de Doutorado** apresentada ao Programa de Pós-Graduação em Engenharia Elétrica e de Computação da UFRN (área de concentração: Engenharia de Computação) como parte dos requisitos para obtenção do título de Doutor em Ciências.

Número de ordem PPgEEC: D147 Natal, RN, julho de 2015

#### Seção de Informação e Referência Catalogação da Publicação na Fonte. UFRN / Biblioteca Central Zila Mamede

Silva, Bruno Marques Ferreira da.

Registro global de nuvens de pontos RGB-D em tempo real usando fluxo óptico e marcadores / Bruno Marques Ferreira da Silva. – Natal, RN, 2015. 108 f.

Orientador: Luiz Marcos Garcia Gonçalves.

Tese (Doutorado em Engenharia Elétrica e de Computação) — Universidade Federal do Rio Grande do Norte. Centro de Tecnologia — Programa de Pós-Graduação em Engenharia Elétrica e de Computação.

1. Visão computacional - Tese. 2. SLAM - Tese. 3. Sensores RGB-D - Tese. 4. Fluxo óptico - Tese. I. Gonçalves, Luiz Marcos Garcia. II. Título.

RN/UF/BCZM CDU 004.923

### Registro Global de Nuvens de Pontos RGB-D em Tempo Real Usando Fluxo Óptico e Marcadores

### Bruno Marques Ferreira da Silva

Tese de Doutorado aprovada em 31 de julho de 2015 pela banca examinadora composta pelos seguintes membros:

figh o	
Prof. Dr. Luiz Marcos Garcia Gonçalves (orientador) UFR	N
Suito E. Strang	
Prof. Dr. Justo Emílio Alvarez Jácobo	N
Rosiery Silva Mara	
Profa. Dra. Rosiery da Silva Maia UER	N
Bur Moch by house	
Prof. Dr. Bruno Motta de Carvalho	N
Rafael Bosena Games	
Prof. Dr. Rafael Beserra Gomes UFR	N



# Agradecimentos

Aos meus pais, meu irmão, minha namorada, meus sogros, meus tios e meus amigos, que souberam compreender meus objetivos e me apoiar durante esta jornada.

Ao meu orientador, professor Luiz Marcos, por confiar no meu trabalho mesmo nos momentos mais difíceis.

Aos colegas do Laboratório Natalnet, sempre muito prestativos e dispostos, seja a ajudar na realização de algum experimento ou em engajar em alguma discussão visando enriquecer o conhecimento.

À CAPES, pelo apoio financeiro.

### Resumo

O registro de nuvens de pontos capturadas por sensores de profundidade é uma importante etapa em aplicações de reconstrução 3D baseadas em visão computacional. Em diversas aplicações com requisitos estritos de desempenho, o registro deve ser realizado não só com precisão, como também na frequência de dados de aquisição do sensor. Com o objetivo de registrar nuvens de pontos de sensores RGB-D (e.g. Microsoft Kinect) em tempo real, é proposto nesta tese o uso do algoritmo de fluxo óptico piramidal esparso para registro incremental a partir de dados de aparência e profundidade. O erro acumulado inerente ao processo incremental é posteriormente reduzido, através do uso de um marcador e minimização de erro por otimização em grafo de poses. Resultados experimentais obtidos após o processamento de diversos conjuntos de dados RGB-D validam o sistema proposto por esta tese para aplicações de odometria visual e localização e mapeamento simultâneos (SLAM).

Palavras-chave: Visão computacional, SLAM, Sensores RGB-D, Fluxo óptico.

### **Abstract**

Registration of point clouds captured by depth sensors is an important task in 3D reconstruction applications based on computer vision. In many applications with strict performance requirements, the registration should be executed not only with precision, but also in the same frequency as data is acquired by the sensor. This thesis proposes the use of the pyramidal sparse optical flow algorithm to incrementally register point clouds captured by RGB-D sensors (e.g. Microsoft Kinect) in real time. The accumulated error inherent to the process is posteriorly minimized by utilizing a marker and pose graph optimization. Experimental results gathered by processing several RGB-D datasets validate the system proposed by this thesis in visual odometry and simultaneous localization and mapping (SLAM) applications.

**Keywords**: Computer vision, SLAM, RGB-D sensors, Optical flow.

# Sumário

St	ımári	0		i
Li	sta de	Figura	as	v
Li	sta de	Tabela	as	vii
Li	sta de	Símbo	olos e Abreviaturas	ix
1	Intr	odução		1
	1.1	Metod	lologia	. 2
	1.2	Tese		. 4
	1.3	Contri	ibuições e Aplicações	. 4
	1.4	Organ	ização do Texto	. 5
2	Fun	dament	tos Teóricos	7
	2.1	Image	ens e Sistemas de Coordenadas	. 7
	2.2	Featur	es Visuais	. 9
		2.2.1	Extratores de Cantos	. 9
		2.2.2	Extratores de Pontos Invariantes à Escala	. 10
	2.3	Rastre	eamento de Features	. 11
		2.3.1	Descritores de Features	. 11
		2.3.2	Casamento entre Descritores	. 12
		2.3.3	Rastreamento por Fluxo Óptico	. 12
	2.4	Visão	Estéreo e Nuvens de Pontos RGB-D	. 14
		2.4.1	Estéreo por Luz Estruturada	. 15
		2.4.2	Nuvens de Pontos RGB-D	. 17
		2.4.3	Features e Descritores em Nuvens de Pontos	. 18
	2.5	Regist	tro de Pares de Nuvens de Pontos	. 19
		2.5.1	Registro com Correspondências Conhecidas	. 19
		2.5.2	Registro com Correspondências Desconhecidas	. 19
		2.5.3	Registro Visual Robusto a Falsas Correspondências	. 20
	2.6	Regist	tro Incremental de Nuvens de Pontos	. 20
	27	Locali	ização e Maneamento Simultâneos	21

3	Estado da Arte					
	3.1	Recons	strução 3D por SLAM	25		
	3.2	Recons	strução 3D Baseada em Imagens	26		
	3.3	Reduç	ão do Erro Acumulado no Mapeamento	27		
		3.3.1	Detecção de Fechamento de Laços			
		3.3.2	Otimização em Grafo de Poses			
		3.3.3	Minimização do Erro de Reprojeção			
	3.4	Recons	strução 3D com Sensores RGB-D			
4	Regi	istro de	Nuvens de Pontos RGB-D por SLAM Visual	35		
•	4.1		etria Visual			
		4.1.1	Rastreamento por Fluxo Óptico Piramidal KLT			
		4.1.2	Registro entre Pares de Nuvens de Pontos com Features Visuais .			
	4.2		I por Minimização do Erro em Grafo de Pose			
	7.2	4.2.1	Grafo de Pose			
		4.2.2	Minimização Não-Linear			
		4.2.3	Estrutura Esparsa das Matrizes do Sistema Linearizado			
		4.2.4	Parametrização Mínima da Pose			
_	_	_				
5	<b>Imp</b> 5.1		ação do Algoritmo de Registro de Nuvens de Pontos RGB-D etria Visual	<b>51</b> 52		
	3.1	5.1.1	Detecção de Features Visuais			
		5.1.2	Algoritmo KLT de Rastreamento de Features			
		5.1.2	Algoritmo KLTW de Rastreamento de Features			
		5.1.4	Estimação de Pose Robusta a Falsas Correspondências Visuais			
		5.1.5	,			
	5.2		Registro Incremental a Partir de Registro por Pares			
	3.2		ização do Erro Acumulado			
		5.2.1	Fechamento de Laço com Marcador Artificial			
		5.2.2	Uso de Keyframes			
		5.2.3	Construção do Grafo			
		5.2.4	Resolução Iterativa do Sistema Linearizado	67		
6	_		tos e Resultados	69		
	6.1	3	ntos de Dados e <i>Benchmark</i> TUM RGB-D	69		
	6.2		aração de Rastreamento para Registro de Nuvens de Pontos			
	6.3		as de Erro em Sistemas de SLAM	74		
	6.4		etrização do Rastreamento com Fluxo Óptico			
	6.5		etria Visual RGB-D	81		
		6.5.1	Precisão			
		6.5.2	Desempenho Computacional	84		
	6.6		I RGB-D com Minimização em Grafo de Poses	86		
	6.7	Registr	ro RGB-D: Resultados Qualitativos	91		
	6.8	Discus	são e Limitações	93		
7	Con	clusão		97		

# Lista de Figuras

1.1	SLAM Visual: Módulos Envolvidos
2.1	Transformação entre Sistemas de Coordenadas
2.2	Geometria Estéreo entre Duas Câmeras
2.3	Sensor RGB-D Microsoft Kinect
2.4	Estéreo por Luz Estruturada
2.5	Imagem RGB-D
2.6	SLAM Baseado em Grafos
4.1	Registro Incremental de Nuvens de Pontos
4.2	SLAM por Minimização em Grafo de Poses
5.1	Diagrama do Sistema de Registro de Nuvens RGB-D
5.2	Intuição do Rastreamento KLTTW
5.3	Rastreamento KLT vs. Rastreamento KLTTW
5.4	Erro no Registro Incremental de Nuvens de Pontos RGB-D 61
5.5	Marcador Artificial de Realidade Aumentada
5.6	Grafo: Odometria Visual vs. SLAM Visual 65
5.7	Nuvens RGB-D e Construção do Grafo
6.1	Rastreamento de imagem para imagem: KLT vs. SURF
6.2	Erro Absoluto de Trajetória (ATE)
6.3	Erro de Pose Relativa (RPE)
6.4	Parametrização do Rastreamento
6.5	Distribuição de Features em fr1/xyz
6.6	Distribuição de Features em fr2/xyz
6.7	Desempenho Computacional do KLTTW
6.8	SLAM: Trajetória e Erro ATE por Sequência 1
6.9	SLAM: Trajetória e Erro ATE por Sequência 2
6.10	Registro de Nuvens de Pontos RGB-D: fr1/floor
6.11	Registro de Nuvens de Pontos RGB-D: Laboratório Natalnet
6.12	Registro de Nuvens de Pontos RGB-D: nn/Cafeteira
6.13	Registro de Nuvens de Pontos RGB-D: nn/Capacete

# Lista de Tabelas

3.1	Registro de Nuvens RGB-D: Trabalhos Relacionados	33
6.1	Conjuntos de Dados TUM RGB-D	70
6.2	Configuração dos Parâmetros da Odometria Visual	77
6.3	Parametrização do Rastreamento: Resultados	82
6.4	Odometria Visual: Erro de Pose Relativa Translacional	83
6.5	Odometria Visual: Erro de Pose Relativa Rotacional	84
6.6	Odometria Visual: Desempenho Computacional	85
6.7	Conjuntos de Dados locais RGB-D	86
6.8	SLAM: Erro Absoluto de Trajetória	88

### Lista de Símbolos e Abreviaturas

ATE: Erro Absoluto de Trajetória

BA: Bundle Adjustment

BRIEF: Binary Robust Independent Elementary Features

FAST: Features from Accelerated Segment Test

FOVIS: Fast Odometry from VISion

G2O: General Graph Optimization

GPU: Unidade de Processamento Gráfico

ICP: Iterative Closest Point

IR: Infravermelho

KLT: Fluxo Óptico Kanade-Lucas-Tomasi

KLTTW: Fluxo Óptico Kanade-Lucas-Tomasi com Janelas de Rastreamento

ORB: Oriented FAST and Rotated BRIEF

RANSAC: RAndom SAmple Consensus

RMSE: Raíz do Erro Médio Quadrático

RPE: Erro de Pose Relativa

SFM: Structure from Motion

SIFT: Scale Invariant Feature Transform

SLAM: Simultaneous Localization and Mapping

SURF: Speeded-Up Robust Features

TORO: Tree based netwORk Optimizer

TUM: Technische Universität München

# Capítulo 1

### Introdução

Visão computacional é a área de pesquisa na qual soluções baseadas em hardware e software buscam resolver problemas relacionados à extração de informações a partir de imagens. Os dados de aparência presentes em imagens são processados por técnicas diversas advindas de outras áreas, como processamento digital de sinais e imagens, estatística, estruturas de dados e inteligência artificial, para extrair informações de baixo nível (por exemplo, elementos geométricos como retas ou círculos) ou informações de alto nível (objetos específicos como pessoas ou faces) [Forsyth e Ponce 2002, Szeliski 2010, Bradski e Kaehler 2008]. Posteriormente, estas informações podem servir como dados de entrada em outras tarefas, dependendo do contexto em que as soluções são aplicadas.

Dentre as informações que podem ser extraídas de imagens, a geometria tridimensional [Trucco e Verri 1998, Bradski e Kaehler 2008, Hartley e Zisserman 2004] da cena visualizada é hoje amplamente explorada em aplicações científicas e industriais. Por exemplo, mapas podem ser computados a partir de imagens aéreas [Irschara et al. 2011] e mosaicos a partir de imagens urbanas, como implementado no Google Street View [Google 2015, Anguelov et al. 2010]. Modelos tridimensionais podem ser ser empregados em aplicações de reconhecimento de formas geométricas e objetos 3D [Aldoma et al. 2012]. No ramo do entretenimento, isto também é realizado, seja auxiliando na aplicação de regras em esportes [Hawk-Eye 2015] ou detectando regiões sobre as quais devem ser renderizados gráficos 3D [Metaio 2015].

Comum a todas as aplicações mencionadas está o requisito de se determinar a profundidade de objetos em relação à câmera. Apesar de ser considerado um problema bem estudado em visão computacional, o cálculo de profundidade a partir de uma ou mais câmeras [Hartley e Zisserman 2004, Trucco e Verri 1998] ainda não possui uma solução aplicável a todos os domínios de aplicações, devido a fatores como as características que variam de sensor para sensor.

Sensores RGB-D, introduzidos no mercado voltado ao consumidor final no início da década de 2010, computam uma solução em hardware para este problema. Especificamente, câmeras como o Microsoft Kinect [MSDN 2015, Konolige e Mihelich 2011] oferecem imagens coloridas onde cada pixel está associado a uma medida de profundidade, com imagens possuindo resoluções de 640 × 480 e capturadas a uma frequência de 30 Hz. Inicialmente voltados ao mercado de entretenimento digital, estes sensores logo passaram a ser explorados pela comunidade científica [Han et al. 2013, Guo et al. 2014]. Proble-

mas resolvidos tradicionalmente a partir de dados 2D podem então ser solucionados pelo processamento de um novo tipo de dados 3D capturado por estes sensores, denominados nuvens de pontos RGB-D [Rusu e Cousins 2011, Aldoma et al. 2012]. Devido ao baixo custo, câmeras RGB-D despertam grande interesse na robótica, principalmente por causa de aplicações como reconhecimento de objetos [Gomes et al. 2013] e reconstrução 3D [Newcombe et al. 2011].

Em particular, estes dois domínios de aplicações compartilham de vários fundamentos matemáticos em comum, principalmente quando nuvens de pontos RGB-D são usadas como dados de entrada. Extrair características geométricas ou visuais presentes nestes dados para computar uma transformação de registro entre um par de nuvens de pontos estão entre os passos básicos que formam soluções mais completas para ambos os problemas. Através desta transformação, que alinha os sistemas de coordenadas cartesianas dos dois conjuntos de dados, pode-se determinar a presença de um objeto em uma cena [Gomes et al. 2013], como também pode-se obter os dados de entrada necessários para uma reconstrução 3D em forma de malha poligonal dos objetos visualizados. A nuvem de pontos resultante do registro pode ser pós-processada visando aplicações específicas, como por exemplo renderização gráfica [Marton et al. 2009, Foley et al. 1990] ou mapeamento de ambientes para robótica [Souza e Gonçalves 2015, Hornung et al. 2013].

O objeto de estudo desta tese se concentra no desenvolvimento de um método computacional em software que supere os desafios encontrados no problema de registro de nuvens de pontos RGB-D visando reconstrução 3D. Este processo deve ser realizado com o máximo de precisão possível, levando-se em consideração características do sensor, como ruídos presentes nos dados de aparência e profundidade e a taxa em que os dados são adquiridos.

#### 1.1 Metodologia

O registro entre pares de nuvens de pontos assume que elas possuem um certo grau de sobreposição de dados. Correspondências entre os dados comuns às duas nuvens devem ser computadas, de maneira que a partir delas, seja possível estimar a transformação de alinhamento. Considerando que o registro de nuvens RGB-D pode ser realizado por dados de aparência [Hu et al. 2012], dados de profundidade [Newcombe et al. 2011] ou por ambos [Endres et al. 2012], o rastreamento de características visuais por fluxo óptico [Lucas e Kanade 1981, Shi e Tomasi 1994, Bouguet 2000] se apresenta como uma possibilidade promissora para obter correspondências entre pares de imagens adquiridas entre instantes de tempo consecutivos. A partir de subconjuntos de pontos das duas nuvens em questão, a transformação de registro pode ser computada, originando assim um processo incremental (realizado para cada par de imagens consecutivas) de registro de nuvens de pontos por odometria visual [Scaramuzza e Fraundorfer 2011, Fraundorfer e Scaramuzza 2012].

Este processo incremental de registro está sujeito a acúmulos de erro, uma vez que as transformações computadas possuem somente consistência local. Em alternativa a estabelecer correspondências entre pares de nuvens de pontos, associações de dados podem ser estimadas entre uma nuvem de pontos de entrada e um mapa. Estes tipos de abordagens, com teorias bem fundamentadas no tópico de pesquisa de SLAM (*Simultaneous Localiza-*

3

tion and Mapping) [Durrant-Whyte e Bailey 2006, Bailey e Durrant-Whyte 2006, Thrun et al. 2005, Thrun e Leonard 2008], realizam o registro de forma global, acumulando o mínimo de erros caso as associações de dados e medições sensoriais estejam corretas. O uso de dados de aparência em SLAM, no chamado SLAM visual, faz com que a associação de dados seja estabelecida com menor ambiguidade [Newman et al. 2006, Cummins e Newman 2011], seja no rastreamento realizado de imagem para imagem ou na obtenção de correspondências em relação a um mapa.

Nesta tese, o registro de nuvens de pontos é abordado como um problema de SLAM visual, visando um registro computado com consistência global. O problema é modelado utilizando como mapa um grafo de poses [Lu e Milios 1997, Grisetti et al. 2010, Kümmerle et al. 2011]. As transformações rígidas do registro incremental obtido por odometria visual constituem as arestas do grafo, as quais unem vértices formados pela posição e orientação (com 6 graus de liberdade) do sensor no momento de captura de dados. Após várias hipóteses sobre os parâmetros de posição e orientação de um mesmo vértice serem adicionadas ao mapa, a trajetória do sensor pode ser otimizada, em um processo que computa os valores mais prováveis para estes parâmetros dadas as hipóteses. Com SLAM em grafos, o problema pode ser resolvido com menos limitações em comparação com algoritmos de SLAM com filtragens probabilísticas [Davison 2003], fazendo com que ele seja uma solução adequada considerando a densidade dos dados de nuvens RGB-D.

Devido ao fato do fluxo óptico não associar dados entre imagens capturadas de pontos de vista consideravelmente diferentes, é utilizado um marcador artificial de realidade aumentada [Garrido-Jurado et al. 2014] para este fim. O detector do marcador artificial age exatamente como um detector de fechamento de laços, isto é, detectando locais previamente mapeados e adicionando restrições ao grafo de forma que ele possa ter o erro de mapeamento acumulado minimizado. Apesar de necessitar intervenção humana, isto é implementado de forma mínima, bastando que um único marcador seja inserido na cena a ser mapeada para que o registro com consistência global seja computado. A Figura 1.1 ilustra um diagrama com os componentes de SLAM visual envolvidos na solução proposta pela tese.

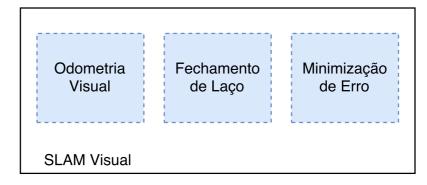


Figura 1.1: Módulos envolvidos no sistema de SLAM visual proposto.

#### **1.2** Tese

Idealmente, o método proposto deve ser executado em uma janela de tempo de processamento por nuvem RGB-D que seja suficiente para manter o sistema de registro em funcionamento na frequência de aquisição de dados do sensor (30 Hz). Isto torna o sistema atrativo para aplicações com requisitos de processamento em tempo real, como em entretenimento com realidade aumentada [Klein e Murray 2007] ou mapeamento para sistemas robóticos terrestres [Endres et al. 2012] ou aéreos [Huang et al. 2011]. Motivado por essas aplicações e tendo como base a problemática apresentada, a seguinte tese é proposta.

**Tese:** É possível registrar nuvens de pontos RGB-D de forma incremental em tempo real por correspondências obtidas pelo fluxo óptico e otimizar este registro inicial de forma global com o uso de marcadores.

A tese é validada sob um ferramental experimental presente na literatura [Sturm et al. 2012] desenvolvido para avaliar algoritmos de odometria e SLAM visual baseados em sensores RGB-D. Especificamente, conjuntos de dados compostos por nuvens RGB-D associadas a um *ground truth* são empregadas nos experimentos. Por causa de ruídos presentes em sensores, não é possível comparar diretamente a qualidade das nuvens de pontos obtidas pelo registro. Por isso, as transformações de registro são avaliadas por diferentes métricas, a partir das quais pode-se comprovar a precisão do sistema proposto em odometria e SLAM visual com o uso de marcadores. Com a finalidade de clarificar como se comporta a capacidade de registro incremental do sistema proposto, o seu resultado é comparado com os obtidos por um sistema de odometria visual [Huang et al. 2011] e SLAM [Endres et al. 2012] para sensores RGB-D. Para avaliar o desempenho do registro global, conjuntos de dados RGB-D foram coletados fazendo uso de um marcador de realidade aumentada. Por sua vez, este desempenho é aferido tanto de forma quantitativa, através do erro de registro computado antes e depois da otimização, como também de forma qualitativa, com registros globais sendo disponibilizados para avaliação visual.

O escopo do método nesta tese se limita ao registro de pontos realizado em ambientes internos e estáticos, com leituras de sensor ricas tanto em dados de aparência quanto de profundidade. A presença do marcador de realidade aumentada é estritamente necessária para minimizar o erro acumulado na odometria visual, sendo imposta ao sistema a execução neste último modo caso o marcador não esteja presente.

#### 1.3 Contribuições e Aplicações

São três as contribuições alcançadas em consequência desta tese. Na primeira delas, é desenvolvida uma estratégia de rastreamento de características visuais construída sobre o fluxo óptico piramidal esparso [Lucas e Kanade 1981, Shi e Tomasi 1994, Bouguet 2000]. Neste método [Silva e Gonçalves 2014a, Silva e Gonçalves 2014b, Silva e Gonçalves 2015c], é proposto o uso de janelas de rastreamento associadas a cada característica visual, de maneira que as posições destes pontos sejam mais uniformemente distribuídas

pela imagem em comparação com o fluxo óptico padrão. Como mostrado nos experimentos, este algoritmo possui desempenho de odometria visual similar ao estado da arte em sistemas RGB-D.

Uma segunda contribuição propõe um algoritmo de SLAM visual [Silva e Gonçalves 2015b] com o uso de um marcador artificial de realidade aumentada para realizar o registro de pontos [Gomes et al. 2013] de forma global. Com isto, é computada a solução mais provável dadas várias imagens em que o marcador é detectado, minimizando o erro acumulado no registro inicial.

Na terceira contribuição, é realizado um estudo [Silva e Gonçalves 2015a] comparando a precisão do fluxo óptico com um algoritmo de detecção e extração de descritores [Bay et al. 2008] no cenário de rastreamento de características visuais com sensores RGB-D. As duas formas de rastreamento são avaliadas em relação à quantidade de falsas correspondências obtidas para uma mesma fração de correspondências estabelecidas, o que é feito utilizando os mesmos conjuntos de dados RGB-D empregados no experimento de odometria visual.

Além das aplicações de odometria e SLAM visual implementadas para avaliar o desempenho do sistema proposto, o método pode ser utilizado para fornecer nuvens de pontos registradas globalmente a serem empregadas na digitalização de objetos 3D ou de ambientes internos. Outras aplicações podem potencialmente se beneficiar do registro computado, como reconhecimento de objetos em nuvens de pontos RGB-D e realidade aumentada.

#### 1.4 Organização do Texto

O restante deste trabalho está organizado como segue. No Capítulo 2, os fundamentos teóricos relacionados ao problema de registro de nuvens de pontos são apresentados. No Capítulo 3, os trabalhos relacionados com registro de nuvens de pontos para reconstrução 3D são enumerados, com atenção especial aos métodos que se baseiam em sensores RGB-D. A formalização matemática adotada para modelar o problema e a sua solução é apresentada no Capítulo 4, enquanto no Capítulo 5 são discutidos os algoritmos e detalhes de implementação. O ferramental de análise, a parametrização do método de rastreamento proposto e os experimentos conduzidos para avaliar as capacidades de SLAM e odometria visual são exibidos no Capítulo 6, com as conclusões sobre o trabalho fechando o documento no Capítulo 7.

## Capítulo 2

### **Fundamentos Teóricos**

Neste capítulo, são apresentados os fundamentos teóricos necessários para melhor compreender o problema abordado nesta tese. Em particular, são exibidos quais são os modelos geométricos envolvidos e como eles podem ser estimados a partir de elementos extraídos de imagens. Além disso, é mostrado como os sensores RGB-D obtêm dados de profundidade e como são geradas nuvens de pontos RGB-D, o tipo de dado explorado pelo método de registro proposto. Por fim, são discutidos os modelos probabilísticos de SLAM que podem ser utilizados como mecanismo para forçar consistência global.

#### 2.1 Imagens e Sistemas de Coordenadas

Uma imagem digital de aparência I(u,v) pode ser definida como uma função bidimensional discreta que mapeia um par de coordenadas u,v em um nível de cinza denotando a intensidade na posição correspondente,

$$I: \mathbb{N}^2 \to \mathbb{N}, \quad (u, v) \mapsto I(u, v).$$
 (2.1)

Para uma imagem com  $L \times A$  amostras,  $u \in [0, L-1]$  é a coordenada horizontal,  $v \in [0, A-1]$  é a coordenada vertical e  $I(u,v) \in [0,255]$  são os valores de intensidade em cada posição após amostragem e discretização realizada pelo sensor de captura [Trucco e Verri 1998]. A imagem monocromática I(u,v) pode ser obtida a partir de imagens coloridas pela conversão  $I(u,v) = (I^r(u,v) + I^g(u,v) + I^b(u,v))/3$ , onde  $I^r(u,v)$ ,  $I^g(u,v)$  e  $I^b(u,v)$  são imagens com os canais vermelho, verde e azul respectivamente.

A posição na imagem  $\mathbf{p} = [u, v]^t \in \mathbb{R}^2$  na qual um ponto  $\mathbf{P} = [x, y, z]^t \in \mathbb{R}^3$  referenciado no sistema de coordenadas de câmera é projetado pode ser determinada por projeção perspectiva. Sendo os parâmetros de uma matriz de projeção  $\mathbf{K}$  dados por  $f_x$  e  $f_y$  (distâncias focais) e  $c_x$  e  $c_y$  (coordenadas de imagem do centro de projeção da câmera), as coordenadas homogêneas  $\tilde{\mathbf{p}}$  relativas ao ponto são dadas por

$$\tilde{\mathbf{p}} = \begin{bmatrix} uz \\ vz \\ z \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \tag{2.2}$$

com **p** igual às duas primeiras coordenadas de  $\tilde{\mathbf{p}}/z$ ,

$$\left[\begin{array}{c} u\\v\end{array}\right] = \left[\begin{array}{c} f_x(x/z) + c_x\\f_y(y/z) + c_y\end{array}\right].$$

Os parâmetros de projeção são determinados por um processo de calibração [Zhang 1999, Bradski e Kaehler 2008] realizado com um objeto conhecido.

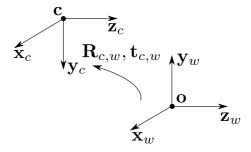


Figura 2.1: Transformação rígida entre um sistema de coordenadas centrado na câmera e um sistema de coordenadas de mundo.

Para um ponto  $\mathbf{P}_w$  referenciado em um sistema de coordenadas arbitrário fixo no mundo, a sua projeção pode ser obtida aplicando-se uma transformação de corpo rígido dada pela matriz de rotação  $\mathbf{R}_{c,w} \in SO(3)$  (grupo das matrizes de rotação de  $\mathbb{R}^3$ ) de dimensão  $3 \times 3$  e vetor de translação  $\mathbf{t}_{c,w} \in \mathbb{R}^3$  de dimensão  $3 \times 1$ ,

$$\mathbf{P}_{c} = \mathbf{R}_{cw} \mathbf{P}_{w} + \mathbf{t}_{cw}$$

Como mostra a Figura 2.1,  $\mathbf{R}_{c,w}$  e  $\mathbf{t}_{c,w}$  relacionam o sistema de coordenadas centrado na câmera, com posição  $\mathbf{c}$  e orientação dada pelos eixos  $\mathbf{x}_c, \mathbf{y}_c, \mathbf{z}_c$ , com um referencial fixo no mundo centrado na posição  $\mathbf{o}$  com orientação dada pelos eixos  $\mathbf{x}_w, \mathbf{y}_w, \mathbf{z}_w$ . Utilizando transformações homogêneas, esta relação pode ser generalizada por

$$\tilde{\mathbf{P}}_b = \mathbf{T}_{b,a} \tilde{\mathbf{P}}_a, \tag{2.3}$$

onde  $\tilde{\mathbf{P}}_a = [x_a, y_a, z_a, 1]^t$  é o vetor  $\mathbf{P}_a$  em coordenadas homogêneas (o mesmo para  $\tilde{\mathbf{P}}_b$ ) e  $\mathbf{T}_{b,a}$  é a matriz de transformação homogênea  $4 \times 4$  dada por

$$\mathbf{T}_{b,a} = \left[ \begin{array}{cc} \mathbf{R}_{b,a} & \mathbf{t}_{b,a} \\ \mathbf{0} & 1 \end{array} \right],$$

que transforma um ponto  $P_a$  em um sistema de coordenadas a em um ponto  $P_b$  em um sistema de coordenadas b.

#### 2.2 Features Visuais

Algoritmos de visão computacional constantemente baseiam-se na extração de características — ou como são comumente chamadas, features — em imagens. Estas características aparecem na imagem como projeções de objetos com propriedades distintas presentes na cena visualizada, se apresentando como pontos, geometrias (linhas ou círculos por exemplo) ou regiões que se sobressaem em relação à sua vizinhança.

O sistema proposto baseia-se na extração de pontos distintivos em imagens que, em utilização com outros algoritmos, possibilitam um desempenho eficiente em aplicações que necessitam de processamento na taxa de aquisição de vídeo (tipicamente 30 Hz). No restante deste documento, o termo features irá se referir a características de imagem em forma de pontos distintivos.

#### 2.2.1 Extratores de Cantos

Os extratores de pontos mais elementares classificam um ponto  $\mathbf{p} = [u, v]^t$  como ponto característico analisando variações em I(u, v) ao longo das direções horizontal e vertical da sua intensidade. Por exemplo, o algoritmo de Harris [Harris e Stephens 1988] extrai um ponto  $\mathbf{p}$  caso a condição

$$\det(\mathbf{C}(\mathbf{p})) + K \operatorname{tr}^2(\mathbf{C}(\mathbf{p})) > \tau,$$

sobre a matriz  $C(\mathbf{p})$  seja verdadeira, onde K é um parâmetro do algoritmo, tr(.) é o traço de uma matriz e  $\tau$  é um limiar especificado. Sendo  $I_u$  e  $I_v$  os gradientes da intensidade da imagem na direção horizontal e vertical (respectivamente) dados por

$$I_u = \frac{\partial I}{\partial u}, \quad I_v = \frac{\partial I}{\partial v},$$
 (2.4)

a matriz  $\mathbf{C}(\mathbf{p})$  é formada por somatórios destas grandezas computados em uma janela  $W = W_x \times W_y$ , como mostra a Equação 2.5

$$\mathbf{C}(\mathbf{p}) = \begin{bmatrix} \sum_{W} I_u^2 & \sum_{W} I_u I_v \\ \sum_{W} I_u I_v & \sum_{W} I_v^2 \end{bmatrix}.$$
 (2.5)

Devido à forma particular do padrão de intensidade das posições **p** que atendem a esta restrição, o operador de Harris também é comumente chamado de detector ou extrator de cantos (*corners*).

Posteriormente, Shi e Tomasi (1994) estenderam o algoritmo proposto por Harris observando que um ponto  $\mathbf{p}$  é um ponto característico caso o menor autovalor de  $\mathbf{C}(\mathbf{p})$  seja suficientemente grande. Matematicamente, um ponto  $\mathbf{p} = [u,v]^t$  é uma feature caso a condição

$$\min(\lambda_1, \lambda_2) > \tau \tag{2.6}$$

seja verdadeira, onde  $\lambda_1$  e  $\lambda_2$  são autovalores de  $\mathbf{C}(\mathbf{p})$  e  $\tau$  é um limiar especificado. Este extrator de pontos característicos ficou conhecido como ponto algoritmo de Shi-Tomasi.

#### 2.2.2 Extratores de Pontos Invariantes à Escala

Os extratores de cantos de Harris e Shi-Tomasi não são invariantes a transformações de escala na imagem. Na prática, a porção da cena ou objeto que em algum instante teve a sua projeção na imagem classificada como ponto distintivo pode não ser detectada como tal com o sensor óptico se afastando ou se aproximando da cena/objeto em questão. Extratores de features invariantes à escala são então propostos, sendo o SIFT (*Scale Invariant Feature Transform*) [Lowe 2004] e o SURF (*Speeded-Up Robust Features*) [Bay et al. 2008] os algoritmos mais utilizados.

Para detectar pontos distintivos invariantes à escala, a análise de cada ponto candidato é realizada no chamado espaço de escalas. O espaço de escalas de uma imagem I(u,v) é uma função  $L(u,v,\sigma)$  gerada a partir de operações de convolução entre a imagem e uma função Gaussiana bidimensional  $G(u,v,\sigma)$ , ou seja,

$$L(u, v, \sigma) = I(u, v) * G(u, v, \sigma),$$

onde

$$G(u, v, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(u^2+v^2)/2\sigma^2}.$$

De acordo com a teoria do espaço de escalas, o valor do parâmetro  $\sigma$  determina uma escala a ser examinada, sendo este variado entre valores e intervalos estipulados previamente. Realizar convoluções com vários valores para  $\sigma$  é um processo extremamente custoso, o que faz com que o SIFT realize a detecção de pontos característicos em uma aproximação do Laplaciano do Gaussiano dada pela Diferença de Gaussianos  $D(u, v, \sigma)$ ,

$$D(u, v, \sigma) = L(u, v, K\sigma) - L(u, v, \sigma), \tag{2.7}$$

onde K é um parâmetro que denota o intervalo entre escalas sucessivas. Uma coordenada na imagem  $\mathbf{p}$  é considerada um ponto característico SIFT na escala  $\sigma$  caso o valor de  $D(u, v, \sigma)$  seja um mínimo ou máximo na região que considera a vizinhança espacial e vizinhança entre escalas adjacentes do ponto.

O SURF procede de forma similar ao SIFT com análise no espaço de escalas para detectar pontos invariantes à esta propriedade. Entretanto, o algoritmo SURF foi desenvolvido pensando em eficiência computacional, em contraste com o SIFT que não foi projetado tendo isto como prioridade. Ao invés de utilizar a aproximação dada pela Diferença de Gaussianos  $D(u, v, \sigma)$  como feito pelo SIFT, o SURF classifica um ponto  $\mathbf{p}$  como distintivo realizando uma análise do seu Hessiano  $\mathbf{H}(u, v, \sigma)$ , dado por

$$\mathbf{H}(u,v,\sigma) = \begin{bmatrix} L_{uu}(u,v,\sigma) & L_{uv}(u,v,\sigma) \\ L_{uv}(u,v,\sigma) & L_{vv}(u,v,\sigma) \end{bmatrix},$$
(2.8)

onde

$$L_{uu}(u, v, \sigma) = \frac{\partial^2}{\partial u^2} G(u, v, \sigma),$$

$$L_{vv}(u, v, \sigma) = \frac{\partial^2}{\partial v^2} G(u, v, \sigma)$$

e

$$L_{uv}(u, v, \sigma) = \frac{\partial}{\partial u \partial v} G(u, v, \sigma)$$

são o Laplaciano do Gaussiano na imagem ao longo das direções correspondentes.

Com o uso de filtros bidimensionais em formas de caixa implementados por imagens integrais, o SURF é capaz de acelerar a geração do espaço de escala. Estes filtros podem ser computados rapidamente gerando aproximações  $D_{uu}(u,v,\sigma)$ ,  $D_{vv}(u,v,\sigma)$  e  $D_{uv}(u,v,\sigma)$  do Laplaciano do Gaussiano, que são por sua vez utilizadas para computar uma aproximação do Hessiano  $\mathbf{H}_{ap}(u,v,\sigma)$ . Assim, o SURF classifica um ponto como característico caso o determinante do Hessiano aproximado seja maior do que um limiar dado, ou seja se

$$\det(\mathbf{H}_{ap}(u,v,\sigma)) = D_{uu}(u,v,\sigma)D_{vv}(u,v,\sigma) - (0,9D_{uv}(u,v,\sigma))^2 > \tau,$$

considerando (assim como o SIFT) a vizinhança espacial e ao longo de escalas adjacentes.

O uso de imagens integrais permite que o espaço de escala seja gerado de forma eficiente alterando-se o tamanho dos filtros ao invés de serem realizadas filtragens com diferentes valores de  $\sigma$  e versões subamostradas da imagem, como feito pelo SIFT.

#### 2.3 Rastreamento de Features

As informações extraídas de nuvens de pontos RGB-D precisam ser mantidas pelo algoritmo à medida em que novos dados são adquiridos pelo sensor. O registro de nuvens de pontos RGB-D proposto nesta tese envolve o uso extensivo de características visuais, embora também seja possível utilizar características puramente geométricas para este fim [Aldoma et al. 2012, Gomes et al. 2013].

#### 2.3.1 Descritores de Features

Um descritor  $\mathbf{d}$  é associado a um ponto característico  $\mathbf{p}$  para possibilitar que correspondências entre pontos  $\mathbf{p}_a^i$  e  $\mathbf{p}_b^j$  pertencentes a diferentes imagens  $I_a(u,v)$  e  $I_b(u,v)$  sejam estabelecidas. Uma possível opção de descritor poderia ser um vetor formado pelos valores das intensidades na imagem das posições contidas em uma janela em torno do ponto distintivo. Apesar deste descritor ser utilizado para computar correspondências por correlação [Bradski e Kaehler 2008], estes então chamados *patches* possuem limitações como não serem invariantes à rotação ou escala.

Visando obter invariância a estes fatores, algoritmos como SIFT e SURF também computam um vetor descritor **d** para cada ponto característico **p**, descrevendo assim a vizinhança em torno de cada ponto de forma numérica. Após ser determinada a escala de cada ponto **p**, a orientação do ponto é computada por estatísticas do gradiente da imagem na sua vizinhança. Uma janela orientada de acordo com o ângulo e escalas do ponto distintivo é construída no entorno do ponto, com um histograma de orientação computado para blocos correspondentes a subdivisões desta vizinhança. Estes valores são inseridos em um vetor de *N* dimensões, com a posterior normalização do vetor para diminuir a influência de mudanças de intensidade nas imagens. Este processo descrito de forma

resumida é aplicado tanto ao SIFT quanto ao SURF, nos quais os descritores dos pontos tem dimensões iguais a 128 e 64 respectivamente. O SURF novamente faz uso de imagens integrais, computando descritores com menos custo computacional.

#### 2.3.2 Casamento entre Descritores

Com vetores descritores invariantes à rotação e escala relacionados a pontos característicos de imagens, casamentos entre pontos podem ser estabelecidos por buscas de vizinhos mais próximos no espaço N-dimensional. Formalmente, para conjuntos de pontos extraídos de imagens diferentes  $I_a(u,v)$  e  $I_b(u,v)$  dados por  $\mathcal{P}_a = \{\mathbf{p}_a^0,...,\mathbf{p}_a^{M_a-1}\}$  e  $\mathcal{P}_b = \{\mathbf{p}_b^0,...,\mathbf{p}_b^{M_b-1}\}$ , com conjuntos de descritores associados  $\mathcal{D}_a = \{\mathbf{d}_a^0,...,\mathbf{d}_a^{M_a-1}\}$  e  $\mathcal{D}_b = \{\mathbf{d}_b^0,...,\mathbf{d}_b^{M_b-1}\}$ , uma correspondência é estabelecida entre  $\mathbf{p}_a^i$  e  $\mathbf{p}_b^j$  caso o descritor  $\mathbf{d}_b^j$  seja o mais próximo de  $\mathbf{d}_a^i$  entre todos os vetores de  $\mathcal{D}_b$ , de acordo com a distância Euclidiana.

A grande vantagem desta forma de obtenção de casamentos é que mesmo com imagens de uma mesma cena  $I_a(u,v)$  e  $I_b(u,v)$  capturadas de pontos de vista bem distintos, um número considerável de casamentos pode ser estabelecido, suportando assim o casamento de features por linha de base larga (*wide baseline matching*) [Duan et al. 2009]. Por outro lado, este método é extremamente custoso, uma vez que exige a detecção de pontos invariantes à escala, a computação de descritores a eles associados e a busca de vizinhos mais próximos no espaço N-dimensional, ainda que esta última etapa possa ser acelerada com buscas aproximadas [Muja e Lowe 2014].

Em aplicações que exigem casamentos entre features sendo mantidos de imagem para imagem em um processo contínuo de rastreamento, esta forma de obter correspondências, chamada de rastreamento por detecção, pode ser empregada, embora produza uma fração de falsos casamentos (associações incorretas) considerável entre features (como mostrado nos experimentos da Seção 6.2). Note que, neste trabalho, o termo rastreamento é usado sempre em relação a features, não possuindo significado relacionado a rastreamento de objetos conhecidos ou rastreamento de câmera (como é frequente na literatura [Lepetit e Fua 2005]).

#### 2.3.3 Rastreamento por Fluxo Óptico

Rastreamento por linha de base curta (*short* ou *narrow baseline matching*) [Duan et al. 2009] é ideal para aplicações de processamento de vídeo (sequências de imagens a uma frequência de 30 Hz), onde geralmente os movimentos capturados são curtos no intervalo entre as imagens. Além de correlação entre *patches* de imagens [Bradski e Kaehler 2008], uma alternativa de rastreamento deste tipo pode ser obtida pelo fluxo óptico, definido formalmente como movimento aparente no padrão de intensidade de imagens ([Trucco e Verri 1998, Bradski e Kaehler 2008]). Por movimento aparente, entenda-se o movimento da câmera em relação à cena ou de forma recíproca, o movimento de objetos em relação à câmera. Determinar o fluxo óptico consiste em computar o vetor deslocamento para um determinado pixel entre duas imagens. Isto pode ser feito de forma densa, onde cada posição na imagem tem o seu fluxo determinado, ou de forma esparsa, onde o fluxo é

calculado para algumas posições pré-determinadas (as quais podem ser features) [Bruhn et al. 2005].

Soluções dadas por métodos densos são mais custosas computacionalmente e mais sensíveis a perturbações, enquanto métodos esparsos são construídos para manter o erro mínimo em uma vizinhança local de pixels e por isso, são mais eficientes. A partir do fluxo óptico esparso, pode-se generalizar por interpolação os vetores deslocamento para pixels vizinhos que não tenham uma estimativa calculada, ao passo em que isto é feito naturalmente para os métodos densos.

Algumas premissas são assumidas na elaboração de um modelo matemático cuja resolução ofereça uma solução ao fluxo óptico. Por exemplo, assume-se que o intervalo de tempo entre as imagens é pequeno em relação ao movimento aparente. Além disso, assume-se que o padrão de intensidade de um determinado pixel ao longo de uma sequência de imagens se mantém constante, esteja este pixel em movimento ou não. Utilizando a representação de imagem I(u,v,t) para denotar a intensidade em uma posição u,v em função do tempo, estas premissas são matematicamente equivalentes a

$$\frac{dI(u,v,t)}{dt} = 0,$$

a qual, sendo explicitadas as derivadas parciais, origina a equação de brilho constante

$$\frac{dI(u,v,t)}{dt} = \frac{\partial I}{\partial u}\frac{du}{dt} + \frac{\partial I}{\partial v}\frac{dv}{dt} + \frac{\partial I}{\partial t} = 0.$$
 (2.9)

Diversos algoritmos estão presentes na literatura para computar o fluxo óptico a partir desta equação como ponto de partida [Trucco e Verri 1998, Bradski e Kaehler 2008, Bruhn et al. 2005, Szeliski 2010, Lucas e Kanade 1981, Bouguet 2000]. Um algoritmo eficiente proposto inicialmente para computar o fluxo óptico denso é o algoritmo de Lucas e Kanade (1981). Posteriormente, foi provado [Shi e Tomasi 1994] que este algoritmo não é capaz de determinar o deslocamento para toda posição de imagem, sendo isto possível somente para pixels em que a condição dada pela Equação 2.5 é satisfeita. Por esta razão, o algoritmo de Lucas e Kanade tem uso prático na determinação de fluxo óptico esparso, principalmente quando utilizado em conjunto com as features Shi-Tomasi, originando assim o termo rastreamento KLT (Kanade-Lucas-Tomasi) [Lucas e Kanade 1981, Shi e Tomasi 1994].

No rastreamento KLT, a equação de brilho constante pode ser reescrita como

$$\nabla I^t \mathbf{v} - \Delta I = 0, \tag{2.10}$$

onde

$$\mathbf{v} = \left[\frac{du}{dt}, \frac{dv}{dt}\right]^t$$

é o vetor deslocamento de cada ponto p,

$$\nabla I = \left[ \frac{\partial I}{\partial u}, \frac{\partial I}{\partial v} \right]^t$$

é o vetor gradiente da intensidade da imagem (conforme Equação 2.4) e  $\Delta I$  é uma imagem composta pela diferença de intensidade para cada pixel entre o instante de tempo t e um instante de tempo infinitesimalmente adiante t+dt, isto é,  $\Delta I(u,v)=I(u,v,t+dt)-I(u,v,t)$ . A Equação 2.10 possui duas variáveis a serem determinadas (os dois componentes do vetor  $\mathbf{v}$  de fluxo óptico), sendo portanto, subdeterminada. Para superar este obstáculo, o algoritmo KLT assume que as vizinhanças de cada pixel da imagem fazem parte de uma mesma vizinhança presente em superfícies da cena visualizada. Isto é equivalente a calcular a Equação 2.10 para todo pixel pertencente a uma janela de tamanho especificado, de maneira a resultar em um sistema sobredeterminado cuja solução é o vetor fluxo óptico  $\mathbf{v}$  do pixel central da vizinhança.

Explorar este artifício possui uma importante consequência em relação à escolha do tamanho da vizinhança. Utilizar uma janela de pixels muito grande pode fazer com que o vetor deslocamento resultante seja suavizado em excesso, ao passo que utilizar um tamanho muito pequeno pode não ser suficiente em relação à magnitude do deslocamento. Em consequência, o fluxo óptico computado possui precisão deteriorada no primeiro caso e múltiplas soluções no segundo. Este problema é conhecido na literatura como problema da abertura [Trucco e Verri 1998, Bradski e Kaehler 2008, Szeliski 2010].

Uma implementação eficiente do algoritmo KLT com uso de pirâmides de imagens [Bouguet 2000] ameniza os efeitos do problema da abertura, com soluções que obtêm precisão suficiente no registro de nuvens de pontos.

#### 2.4 Visão Estéreo e Nuvens de Pontos RGB-D

Devido à característica projetiva inerente a sensores ópticos, a recuperação de informações de profundidade de pontos visualizados pela câmera utilizando uma única imagem é um processo não-trivial [Trucco e Verri 1998, Hartley e Zisserman 2004]. A obtenção de coordenadas 3D da cena visualizada a partir de suas projeções 2D tem uma possível solução nos casos em que:

- 1. As regiões da cena possuam projeções em mais de uma imagem
- 2. As correspondências entre estas projeções tenham sido estabelecidas
- 3. A geometria que relaciona as câmeras relacionadas às imagens seja conhecida

Determinar correspondências entre uma ou mais imagens e a geometria que as relacionam constituem os problemas a serem resolvidos para se computar a profundidade de pontos visualizados, no que se chama de visão estéreo [Trucco e Verri 1998]. Em contraste com a recuperação de estrutura e movimento (*Structure from Motion* – SFM) [Hartley e Zisserman 2004], na qual a geometria entre as imagens é variável, em um sistema estéreo tem-se duas ou mais câmeras capturando imagens paralelamente de uma mesma cena, as quais estão relacionadas por uma geometria fixa. Esta relação, denominada de geometria estéreo ou geometria epipolar [Trucco e Verri 1998, Hartley e Zisserman 2004], restringe duas imagens de uma mesma cena. Como mostrado na Figura 2.2, a posição  $\mathbf{P}_w$  de um ponto pode ser obtida a partir de projeções  $\mathbf{p}$  e  $\mathbf{p}'$  presentes em mais de uma imagem, caso

seja conhecida a transformação rígida  $\mathbf{R}$ ,  $\mathbf{t}$  que alinha os sistemas de coordenadas centrados em  $\mathbf{c}$  e  $\mathbf{c}'$ . Além disto, o conhecimento da geometria estéreo faz com que a busca por correspondências entre pontos seja reduzida às linhas  $\overline{\mathbf{ep}}$  e  $\overline{\mathbf{e'p'}}$ .

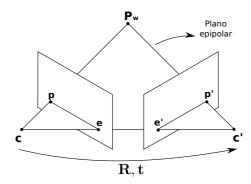


Figura 2.2: Geometria estéreo entre duas câmeras que visualizam uma mesma cena.

Métodos de calibração [Trucco e Verri 1998, Bradski e Kaehler 2008] são empregados para se determinar os parâmetros de projeção (Equação 2.2) de cada câmera e a transformação rígida entre elas. As correspondências podem ser obtidas pelos métodos apresentados na Seção 2.3, sendo em geral utilizadas medidas de correlação entre *patches* de imagens [Bradski e Kaehler 2008] para este fim. A partir das correspondências, computa-se a disparidade, isto é, diferença no valor da coordenada ao longo da dimensão em que foi realizada a busca. A disparidade é finalmente convertida em medida de profundidade fazendo-se uso dos parâmetros de calibração da geometria estéreo.

Devido à dependência de presença de textura na cena para se obter disparidade, a profundidade de um sistema de visão estéreo é muitas vezes de baixa qualidade. Isto pode ocorrer por fatores como escassez no número de correspondências estabelecidas ou pela imprecisão da profundidade computada em consequência de falsas correspondências ou ruídos na imagem. Ainda, como esta rotina geralmente é implementada em software, boa parte do processamento é dedicada à obtenção de correspondências, resultando em uma janela de tempo restante insuficiente para processar imagens na taxa de vídeo (30 Hz).

#### 2.4.1 Estéreo por Luz Estruturada

Ao invés de inferirem a profundidade a partir de dados de aparência, câmeras RGB-D estimam esta grandeza com sensores especificamente projetados para este fim. No caso de sensores RGB-D comerciais como o Microsoft Kinect [MSDN 2015, Konolige e Mihelich 2011], a tecnologia empregada é denominada estéreo por luz estruturada. O sensor é capaz de prover dados de aparência (imagens coloridas – canais RGB) e profundidade (canal D) para cada um dos  $640 \times 480$  pixels da câmera a uma taxa de 30 quadros por segundo explorando este mecanismo.

No Microsoft Kinect, a geometria estéreo relaciona um projetor de infravermelho (IR) e uma câmera sensível a estas frequências para estimar a profundidade de objetos visualizados. Na Figura 2.3a, é mostrado o Kinect sem o seu envoltório de plástico, o que

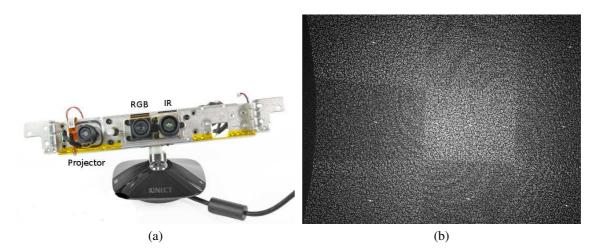


Figura 2.3: Sensor RGB-D Microsoft Kinect. (a) Sensor de cores junto com sensor e projetor infravermelho e (b) Padrão infravermelho utilizado para estimar a disparidade. Fonte: [Konolige e Mihelich 2011].

permite visualizar o projetor e sensor de IR, além da câmera RGB. Um padrão de luz IR em forma de grade (Figura 2.3b) é projetado na cena e utilizado para computar a disparidade através da correspondência estéreo com uma imagem de referência armazenada na memória do dispositivo. Neste processo, a disparidade é computada por correlação entre a imagem de IR e a imagem de referência.

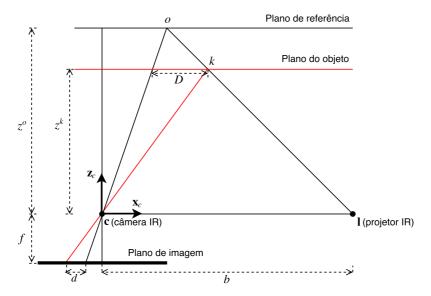


Figura 2.4: Visualização a partir do topo do processo de obtenção de dados de profundidade com estéreo por luz estruturada.

Mais especificamente, sendo  $z^o$  o valor de profundidade de referência (obtido pela projeção da grade de IR em um plano), a profundidade  $z^k$  de um objeto  $\mathbf{P}^k$  pode ser calculada aplicando-se semelhança de triângulos, como exibido na Figura 2.4 [Khoshelham

e Elberink 2012].

A partir do esquema exibido, tem-se as proporções

$$\frac{D}{b} = \frac{z^o - z^k}{z^o} \tag{2.11}$$

e

$$\frac{d}{f} = \frac{D}{z^k},\tag{2.12}$$

nas quais b é a distância entre o projetor e sensor IR, d é a disparidade, f é a distância focal da câmera de IR e D é a distância no espaço do objeto entre o raio emitido pelo projetor e o capturado pela câmera.

Substituindo-se a Equação 2.12 na 2.11 para D e isolando-se  $z^k$ , tem-se a determinação da profundidade a partir da geometria e disparidade

$$z^k = \frac{z^o}{1 + \frac{z^o}{fb}d}. (2.13)$$

O cálculo de disparidade seguido pelo de profundidade é feito para cada pixel da imagem, originando assim a imagem de profundidade densa  $I^d(u,v)$ , que mapeia um par u,v em um valor denotando a profundidade na posição u,v,

$$I^d: \mathbb{N}^2 \to \mathbb{Z}^*, \quad (u, v) \mapsto I^d(u, v).$$
 (2.14)

Note que pixels podem não ter medida de profundidade associada, devido a problemas como oclusão ou interferência de fontes de iluminação externas no padrão projetado.

#### 2.4.2 Nuvens de Pontos RGB-D

Com a geometria estéreo que relaciona a câmera dos dados de aparência com a câmera de IR, as duas matrizes de pixels são alinhadas em um mesmo sistema de referência, originando as chamadas imagens RGB-D. Esta calibração é realizada pelo fabricante, com a rotina de alinhamento realizada em hardware. As matrizes de dados de aparência I(u,v) e dados de profundidade  $I^d(u,v)$  passam a ter uma correspondência de um para um, ou seja, as coordenadas de imagem u e v acessam a cor RGB (ou intensidade) e a profundidade na posição correspondente.

A partir destas matrizes, um conjunto de pontos 3D pode ser computado, onde cada ponto possui um dado de aparência e as coordenadas 3D que o referencia em relação a um sistema de coordenadas fixo na câmera IR. Este sistema de mão direita possui o eixo local  $\mathbf{z}_c$  apontando no sentido do eixo óptico, o eixo  $\mathbf{x}_c$  apontando para a direita e o eixo  $\mathbf{y}_c$  apontando para baixo.

As coordenadas 3D  $\mathbf{P}^k = [x^k, y^k, z^k]^t$  de um ponto podem ser computadas em um processo inverso ao da projeção aplicado às suas coordenadas de imagem  $\mathbf{p}^k = [u^k, v^k]^t$ , ou

$$\begin{bmatrix} x^k \\ y^k \\ z^k \end{bmatrix} = \begin{bmatrix} z^k (u^k - c_x)/f_x \\ z^k (v^k - c_y)/f_y \\ z^k \end{bmatrix}, \tag{2.15}$$

onde  $z^k$  é obtido da imagem de profundidade por  $I^d(u^k, v^k)$ .

Realizar este processo para cada uma das L coordenadas horizontais e A coordenadas verticais resulta na construção de um conjunto  $\mathcal N$  denominado nuvem de pontos RGB-D, no qual cada elemento é composto por uma posição 3D referenciada no sistema de coordenadas de câmera. Cada nuvem de pontos  $\mathcal N$  possui tamanho igual a dimensão  $L \times A$  da imagem (640 × 480 no caso do sensor Kinect). Com esta terminologia, é possível tanto a) obter o ponto  $\mathbf p^k$  e a intensidade na imagem onde é projetado um ponto  $\mathbf P^k$  da nuvem, quanto b) obter  $\mathbf P^k$  a partir das coordenadas de imagem  $\mathbf p^k$  (que por sua vez acessam I(u,v) e  $I^d(u,v)$ ). A imagem RGB (Figura 2.5a) e a nuvem de pontos (Figura 2.5c) obtida da imagem de profundidade (Figura 2.5b) a ela associada são utilizadas como os únicos dados de entrada nos algoritmos propostos por esta tese.

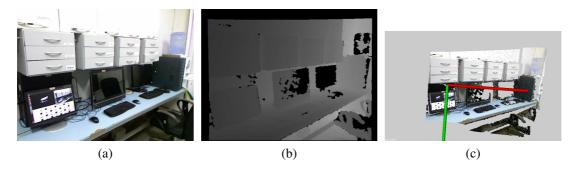


Figura 2.5: Dados de sensores RGB-D: (a) Imagem de aparência (b) Imagem de profundidade (codificada em níveis de cinza para visualização) e (c) Nuvem de pontos RGB-D com sistema de coordenadas de câmera associado, visualizada em 3D com os valores RGB associados a cada ponto.

#### 2.4.3 Features e Descritores em Nuvens de Pontos

Utilizando nuvens de pontos, é possível computar features e descritores no espaço tridimensional [Rusu e Cousins 2011, Aldoma et al. 2012]. Estes elementos representam pontos distintivos levando em consideração a geometria local de cada um deles. Em geral, são computados histogramas de vetores normais estimados em torno de uma vizinhança [Aldoma et al. 2012], visando invariância a transformações de rotação e translação. Como este processo é computacionalmente custoso, features e descritores extraídos do espaço 3D não são adequados a aplicações de registro de nuvens de pontos em tempo real.

### 2.5 Registro de Pares de Nuvens de Pontos

A rotina responsável por computar a transformação rígida que alinha os sistemas de coordenadas relativos a dois conjuntos de pontos é denominada registro de pares de nuvens de pontos. Este processo pode ser útil para alinhar pontos em um referencial comum que foram obtidos por um sensor em movimento capturando dados de uma cena ou reciprocamente, que foram obtidos de um sensor estático capturando dados de objetos que compartilham um movimento (transformação rígida) comum.

#### 2.5.1 Registro com Correspondências Conhecidas

Uma solução para este problema pode ser encontrada assumindo que correspondências entre pontos das duas nuvens são conhecidas. Considerando que os pontos que compõem ambas as nuvens estão livres de ruído, os parâmetros de uma transformação  $\mathbf{T}_{a,b}$ , composta pela matriz de rotação  $\mathbf{R}_{a,b}$  e vetor de translação  $\mathbf{t}_{a,b}$  (conforme a Equação 2.3) que registra os pontos de uma nuvem  $\mathcal{N}_b$  no sistema de referência de uma nuvem  $\mathcal{N}_a$ , podem ser obtidos diretamente.

Devido ao ruído inerente a sensores de profundidade, esta situação ideal não ocorre na prática, implicando no fato de que os parâmetros têm de ser estimados pela solução que minimiza o erro de alinhamento entre as nuvens por mínimos quadrados,

$$\mathbf{R}_{a,b}, \mathbf{t}_{a,b} = \underset{\mathbf{R}, \mathbf{t}}{\operatorname{argmin}} \sum_{k=0}^{M-1} ||\mathbf{P}_{a}^{k} - \left(\mathbf{R}\mathbf{P}_{b}^{k} + \mathbf{t}\right)||^{2},$$
(2.16)

considerando M correspondências entre os pontos  $\mathbf{P}_a^0, \mathbf{P}_a^1, ..., \mathbf{P}_a^{M-1}$  de  $\mathcal{N}_a$  e os pontos  $\mathbf{P}_b^0, \mathbf{P}_b^1, ..., \mathbf{P}_b^{M-1}$  de  $\mathcal{N}_b$ .

Algoritmos eficientes presentes na literatura computam uma solução com o uso de quatérnions [Horn 1987] ou por decomposição em valores singulares [Umeyama 1991].

#### 2.5.2 Registro com Correspondências Desconhecidas

Obter correspondências entre dados puramente geométricos (sem correspondências visuais) é um processo ainda mais ambíguo do que o equivalente utilizando dados de aparência obtidos de imagens. O algoritmo ICP – *Iterative Closest Point* [Besl e McKay 1992] pode ser utilizado para registrar um par de entidades geométricas quaisquer, tais como funções, curvas ou malhas poligonais. Nuvens de pontos 2D ou 3D de diferentes tamanhos e sem correspondências conhecidas estão entre estas entidades geométricas.

De forma iterativa, o algoritmo realiza uma sequência de passos dada por:

- 1. Selecionar uma subamostra dentre todos os pontos das nuvens  $\mathcal{N}_a$  e  $\mathcal{N}_b$
- 2. Estabelecer correspondências entre pontos das duas subamostras, através de buscas por vizinhos mais próximos

- 3. Estimar a transformação de alinhamento que minimiza uma medida de distância. Esta medida pode ser a distância entre pontos da Equação 2.16 ou a distância entre planos [Pomerleau et al. 2013]
- 4. Após aplicar a transformação estimada aos pontos da nuvem  $\mathcal{N}_b$  e medir o erro em relação a  $\mathcal{N}_a$ , voltar ao passo 1 caso este seja maior que um limiar

A transformação de rotação e translação acumulada após convergência das iterações registra as duas nuvens de pontos.

#### 2.5.3 Registro Visual Robusto a Falsas Correspondências

Realizar o registro de nuvens de pontos com ICP possui alguns fatores importantes, tais como, a forma de subamostrar os conjuntos de pontos, como realizar busca por vizinhos mais próximos de forma eficiente e que métrica de erro deve ser minimizada. Considerando os sensores RGB-D, estes fatores se tornam ainda mais relevantes, visto que os dados de profundidade capturados por estas câmeras possuem ruído considerável [Khoshelham e Elberink 2012] e que as nuvens podem possuir até  $640 \times 480 = 307.200$  pontos.

Desta forma, uma alternativa viável para sensores RGB-D é obter correspondências entre features visuais e, utilizando os pontos 3D relacionados aos dados de aparência, realizar o registro pela solução da Equação 2.16. Ainda assim, uma única correspondência incorreta pode implicar em uma estimativa completamente imprecisa para a transformação de alinhamento, já que dados que não seguem distribuição de erro Gaussiana desrespeitam as premissas assumidas em soluções por mínimos quadrados.

Para tornar estimativas computadas por mínimos quadrados robustas a falsas correspondências em imagens, pesquisas em visão computacional propõem o algoritmo RAN-SAC (*RAndom SAmple Consensus*) [Fischler e Bolles 1981]. Assim como o ICP, o RAN-SAC também funciona de forma iterativa, onde em cada uma delas, um número mínimo de dados é selecionado para computar uma solução hipótese para o modelo a ser estimado. Cada solução hipótese é avaliada segundo alguma métrica de erro utilizando todo o conjunto de dados. Após várias iterações, a solução hipótese que possuir a menor soma do erro é eleita a solução consenso. Posto desta forma, o RANSAC pode ser utilizado para computar diversas transformações (modelo) a partir de correspondências disponíveis, podendo ser portanto aplicado para estimar uma transformação de registro entre nuvens de pontos.

# 2.6 Registro Incremental de Nuvens de Pontos

Pode-se pensar que, ao realizar o registro de pares de nuvens de pontos e transformálas em relação a um sistema de referência fixo de forma encadeada, obtém-se uma solução para o fundir ou combinar múltiplas nuvens de pontos em um referencial comum. Por exemplo, ao se obter  $T_{0,1}$  pelo registro da nuvem  $\mathcal{N}_0$  e  $\mathcal{N}_1$  e  $T_{1,2}$  pelo alinhamento de  $\mathcal{N}_1$ e  $\mathcal{N}_2$ , pode-se concatenar  $T_{0,1}$  com  $T_{1,2}$  pelo produto das transformações, resultando no alinhamento  $T_{0,2}$  que registra  $\mathcal{N}_2$  no sistema de referência de  $\mathcal{N}_0$ . Este processo incremental está sujeito a erros devido à existência de ruído nas medições dos sensores, o que faz com que as transformações de registro calculadas passem a divergir por conta de erro acumulado. Mecanismos adicionais são necessários para forçar a consistência global no registro de todas as nuvens.

# 2.7 Localização e Mapeamento Simultâneos

O registro de nuvens de pontos pode ser modelado como um problema de localização e mapeamento simultâneos, mais conhecido na literatura como SLAM [Durrant-Whyte e Bailey 2006, Bailey e Durrant-Whyte 2006, Thrun et al. 2005, Thrun e Leonard 2008]. No SLAM, modelos probabilísticos são propostos de forma que um agente de posse de um sensor computa um mapa ao mesmo tempo em que se localiza em relação a ele. Grande parte da literatura disponível foi desenvolvida considerando um robô como o agente, já que localizar robôs em relação a um mapa é um problema extensivamente estudado e ainda considerado em aberto.

Ao modelar medições sensoriais e mudanças de posição/orientação do sensor ao longo do tempo com incertezas, um mecanismo para forçar consistência global no registro de nuvens pode ser alcançado, resultando em um sistema de uso prático. Estas incertezas são modeladas como probabilidades que podem ser computadas fundamentadas no teorema de Bayes. Formalmente, define-se a pose (posição e orientação) de um robô no instante de tempo t como  $\mathbf{x}_t$ , que deve ser computada à medida em que o robô se desloca por comandos de entrada  $\mathbf{u}_t$  e coleta medições sensoriais  $\mathbf{z}_t$  em um ambiente. Os vetores  $\mathbf{u}_t$  são determinados pelas entradas de controle ou por odometria, ao passo que  $\mathbf{z}_t$  são observações de pontos de referência (landmarks) presentes no ambiente.

Modelos matemáticos são especificados para descrever a relação da odometria  $\mathbf{u}_t$  com as poses  $\mathbf{x}_t$  e  $\mathbf{x}_{t-1}$  e também a relação das medições  $\mathbf{z}_t$  com um mapa  $\mathcal{M}$  e a pose  $\mathbf{x}_t$ . Estas duas relações são dadas em formas de funções de distribuição de probabilidade. A primeira relação, chamada modelo de movimento, é dada por

$$p(\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{u}_t) \tag{2.17}$$

e denota a distribuição de probabilidade do robô se encontrar com pose  $\mathbf{x}_t$  supondo um comando de entrada  $\mathbf{u}_t$  e pose  $\mathbf{x}_{t-1}$  no instante de tempo anterior. Este modelo assume um processo Markoviano [Durrant-Whyte e Bailey 2006], onde o estado anterior  $\mathbf{x}_{t-1}$  contém todo o histórico de poses passadas para se computar  $\mathbf{x}_t$  juntamente com  $\mathbf{u}_t$ . O modelo de medição (segunda relação)

$$p(\mathbf{z}_t|\mathbf{x}_t,\mathcal{M}) \tag{2.18}$$

caracteriza a probabilidade de ser obtida uma medição  $\mathbf{z}_t$  condicionada ao robô se encontrar no estado  $\mathbf{x}_t$  e o mapa estar configurado como  $\mathcal{M}$ . O mecanismo utilizado para associação de dados em SLAM, isto é, relacionar cada uma das medições  $\mathbf{z}_t$  com as grandezas armazenadas no mapa  $\mathcal{M}$ , é de suma importância, já que somente com múltiplas observações de um mesmo ponto de referência pode-se diminuir as incertezas relacionadas aos modelos apresentados. A partir deste mecanismo, é possível detectar fechamento

de laços, ou seja, perceber quando uma parte do ambiente sendo mapeado está sendo revisitada e com isto, reduzir as incertezas envolvidas no processo.

Os modelos de movimento e medição permitem formular soluções para o SLAM de duas maneiras: SLAM *online* e *full* SLAM [Thrun e Leonard 2008].

No SLAM online, deseja-se computar

$$p(\mathbf{x}_t, \mathcal{M}|\mathbf{z}_{0:t}, \mathbf{u}_{0:t}), \tag{2.19}$$

isto é, a densidade de probabilidade do robô se encontrar no estado  $\mathbf{x}_t$  com mapa  $\mathcal{M}$ , considerando que o tempo t é o instante de tempo mais recente. Os algoritmos para resolver esta classe de problema são conhecidos como filtros e funcionam estimando a Equação 2.19 através da aplicação do modelo de movimento e modelo de medição de forma incremental (sempre que o robô se move e obtém uma leitura sensorial). As incertezas são propagadas e corrigidas considerando variáveis aleatórias Gaussianas, nas versões com filtros de Kalman, e também distribuições não-paramétricas, nos algoritmos com filtros de partículas [Thrun e Leonard 2008].

No *full* SLAM, a trajetória completa  $\mathbf{x}_{0:N} = {\mathbf{x}_0,...,\mathbf{x}_N}$ , para um instante de tempo final N, é estimada computando-se a probabilidade dada por

$$p(\mathbf{x}_{0:N}, \mathcal{M}|\mathbf{z}_{0:N}, \mathbf{u}_{0:N}), \tag{2.20}$$

supondo medições  $\mathbf{z}_{0:N}$  e odometrias  $\mathbf{u}_{0:N}$  obtidas nos respectivos instantes de tempo. A rotina de cálculo é executada em lote, tão logo  $\mathbf{x}_N$ ,  $\mathbf{z}_N$  e  $\mathbf{u}_N$  encontrem-se disponíveis, com a solução pela resolução de problemas de minimização não-linear de mínimos quadrados.

Implementações práticas de *full* SLAM [Lu e Milios 1997, Thrun e Montemerlo 2006, Grisetti et al. 2010] baseiam-se no uso de estruturas matemáticas do tipo grafo [Bondy e Murty 1976]. Nestas soluções, o conjunto de vértices  $\mathcal{V}$  é formado pelas poses  $\mathbf{x}_t$  e também por pontos de referência  $\mathbf{P}^k \in \mathcal{M}$ , enquanto o conjunto de arestas  $\mathcal{E}$  é formado pelas relações entre vértices (odometrias) e entre vértices e pontos de referência. A Figura 2.6 ilustra um exemplo de grafo.

O processo de filtragem comum ao SLAM *online* possui limitações quanto à escalabilidade do mapeamento. Isto acontece porque é necessário armazenar a covariância conjunta entre a pose atual  $\mathbf{x}_t$  e todos os pontos de referência do mapa  $\mathcal{M}$ , implicando em rotinas computacionalmente intensivas de atualização destas matrizes à medida em que uma quantidade maior de pontos de referência é adicionada ao mapa [Durrant-Whyte e Bailey 2006]. Considerando a densidade dos dados de nuvens RGB-D e a quantidade de marcos visuais neles presentes, isto se torna um fator limitante. Esta desvantagem não atinge o *full* SLAM, que é considerado o método que melhor une eficiência e precisão nos dias de hoje [Grisetti et al. 2010]. Por outro lado, a otimização da trajetória pode divergir caso as soluções iniciais  $\mathbf{x}_{0:N}$  sejam estimadas de forma incorreta.

Como é demonstrado nesta tese, o uso de rastreamento de features por fluxo óptico se apresenta como uma forma eficiente de associação de dados. Em última análise, trajetórias  $\mathbf{x}_{0:N}$  computadas por um método de registro incremental entre pares de nuvens de pontos por odometria visual [Scaramuzza e Fraundorfer 2011, Fraundorfer e Scaramuzza 2012], servem como solução inicial para a obtenção de registros de nuvens de

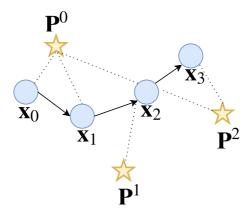


Figura 2.6: SLAM baseado em grafos. Os vértices do grafo são os círculos em azul e as arestas são as setas sólidas (odometria) e retas pontilhadas (medições sensoriais de pontos de referência).

pontos RGB-D com erro minimizado por *full* SLAM. A literatura demonstra que otimizar estimativas iniciais se mostra como alternativa mais eficiente [Strasdat et al. 2010] do que aplicar algoritmos como filtro de Kalman ou filtro de partículas [Thrun e Leonard 2008] no SLAM visual.

# Capítulo 3

# Estado da Arte

A obtenção de modelos geométricos de cenas e objetos do mundo é um tópico de pesquisa ativamente estudado por áreas como computação gráfica, visão computacional e robótica. Apesar de possuírem motivações distintas, o objeto de estudo comum aos três domínios visa obter modelos computacionais a partir de dados sintéticos, imagens ou dados de sensores. Neste trabalho, o interesse é em computar modelos geométricos em formas de nuvens de pontos 3D registradas em um sistema de coordenadas comum, o que pode ser considerado como reconstrução 3D de baixo nível. Outras tarefas podem ser realizadas a partir dos dados de nuvens de pontos, como por exemplo a texturização e geração de malhas geométricas para modelagem 3D [Foley et al. 1990, Marton et al. 2009] ou a obtenção de áreas navegáveis para robôs [Souza e Gonçalves 2015, Hornung et al. 2013].

# 3.1 Reconstrução 3D por SLAM

Pesquisas em robótica podem ser consideradas responsáveis por boa parte do embasamento teórico relacionado com registro de pontos em tempo real, principalmente por causa da formulação probabilística dos algoritmos de SLAM [Durrant-Whyte e Bailey 2006, Bailey e Durrant-Whyte 2006, Thrun et al. 2005, Thrun e Leonard 2008]. Ao mapearem o espaço de trabalho de um robô ao mesmo tempo em que o localizam neste mapa, estes algoritmos podem ser usados para computar reconstruções bidimensionais ou tridimensionais de ambientes internos ou externos, o que é conhecido na robótica como mapeamento de ambientes (environment mapping). Utilizando dados de sensores [Siegwart e Nourbakhsh 2004] proprioceptivos (giroscópios, encoders óptico ou acelerômetros) e exteroceptivos (sonares, lasers, câmeras Time of Flight, ou GPS), a transformação de pose de um robô pode ser estimada alinhando-se o sistema de referência de duas leituras sucessivas de sensores. Com nuvens de pontos 2D ou 3D originárias de sensores de profundidade, este processo pode ser realizado com o algoritmo ICP. De forma iterativa, o algoritmo obtém alinhamentos entre leituras de sensores cuja transformação é a mudança de pose entre os instantes de tempo relacionados. Por confiarem em sensores de profundidade de alto custo e consumo energético, vários sistemas de SLAM [Lu e Milios 1997, Gutmann e Konolige 1999, Montemerlo et al. 2003, Newman et al. 2006] podem ser considerados de difícil acesso e implantação, mesmo nos dias de hoje.

# 3.2 Reconstrução 3D Baseada em Imagens

O uso de sensores ópticos também possibilita computar e registrar nuvens de pontos por meio de diferentes classes de algoritmos. Com a recuperação de estrutura e movimento (SFM) [Hartley e Zisserman 2004], é possível obter soluções por meio de imagens de câmeras de vídeo [Pollefeys et al. 2004, Akbarzadeh et al. 2006], imagens aéreas [Irschara et al. 2011] ou imagens da internet [Snavely et al. 2008, Agarwal et al. 2009]. A introdução do RANSAC [Fischler e Bolles 1981] permite que os modelos matemáticos de SFM possam ser estimados de forma robusta a falsas correspondências entre imagens, o que faz este algoritmo uma importante contribuição em visão computacional. Ainda assim, devido à característica de processarem imagens em lote de forma não-causal, o ferramental de SFM não é adequado para sistemas que necessitam de soluções computáveis em tempo real, tais como sistemas de localização e mapeamento de robôs ou sistemas de gráficos interativos em vídeo (aplicações de realidade virtual e aumentada). Com esta motivação, sistemas de reconstrução 3D que usam câmeras são propostos pela comunidade de robótica com o avanço em estudos em duas outras classes de soluções: SLAM visual e odometria visual.

Localização e mapeamento simultâneos a partir de câmeras (SLAM visual) é um objeto de estudo desde o surgimento das primeiras pesquisas em visão computacional [Moravec 1977]. Esta classe de algoritmos funciona extraindo features como pontos [Davison 2003, Goncalves et al. 2005] ou bordas [Tomono 2010] de uma [Davison 2003] ou mais [Se et al. 2001, Goncalves et al. 2005] câmeras e relacionando estas informações com um mapa. Extratores de pontos característicos como o operador de cantos de Harris e Stephens (1988) ou de Shi e Tomasi (1994) e extratores de linhas como a transformada Hough [Trucco e Verri 1998] são geralmente utilizados para processar a imagem e extrair tais características. Ao se modelar SLAM visual com mecanismos de filtragens probabilísticas, os mapas são constituídos pela média e matrizes de covariância das posições 3D correspondentes às features observadas, sendo estes computados com o mesmo formalismo matemático desenvolvido em SLAM com sensores de profundidade. O filtro de Kalman ou o filtro de partículas, presentes nos sistemas MonoSLAM [Davison 2003] e vSLAM [Karlsson et al. 2005], são exemplos das primeiras abordagens com êxito em SLAM visual da literatura. Todavia, o custo computacional envolvido na atualização das matrizes de covariância destes filtros limitam o processo de reconstrução a poucas (algumas centenas) de features.

Uma terceira frente de algoritmos foi proposta com a adaptação de métodos de SFM para processamento causal e sequencial de imagens, no que passou a ser denominado de odometria visual [Nistér et al. 2004]. Estes algoritmos são implementados minimizandose incrementalmente funções custo modeladas na geometria projetiva [Hartley e Zisserman 2004] que relacionam parâmetros de câmera de duas ou mais imagens. Avanços como o algoritmo dos 5 pontos [Nistér 2003a] e a versão preemptiva [Nistér 2003b] do RAN-SAC são responsáveis por permitir que mudanças incrementais na pose de câmera sejam computadas em tempo real a partir da triangulação 3D de features 2D extraídas das imagens. Implementações de sistemas estéreo [Se et al. 2001, Nistér et al. 2004, Konolige e Agrawal 2007, Konolige e Agrawal 2008, Howard 2008] possuem soluções mais

facilmente implantáveis, uma vez que os dados de profundidade obtidos da correspondência estéreo podem ser usados diretamente para computar a pose de câmera sem que seja necessário estimar as matrizes com restrições da geometria epipolar [Scaramuzza e Fraundorfer 2011, Fraundorfer e Scaramuzza 2012].

# 3.3 Redução do Erro Acumulado no Mapeamento

O registro incremental realizado ao se calcular a mudança de pose entre sucessivas leituras de sensores é susceptível a acúmulos de erro, originários de ruídos existentes nas medições. Este problema é inerente a todos os sistemas incrementais, independente do tipo de sensor empregado. O erro acumulado pode ser minimizado caso haja diversas observações de um mesmo ponto de referência (*landmark*) sendo mapeado, as quais fornecem restrições a um problema de otimização não-linear de mínimos quadrados a ser resolvido de forma iterativa.

#### 3.3.1 Detecção de Fechamento de Laços

Para que este processo funcione corretamente, é necessário detectar locais mapeados previamente, subproblema de SLAM conhecido na literatura como detecção de fechamento de laço (loop closing) [Durrant-Whyte e Bailey 2006]. Uma maior quantidade de fechamentos de laço fornece mais restrições ao sistema a ser minimizado, o que resulta em uma solução com menor erro global. Utilizando imagens, é possível executar esta tarefa com menor ambiguidade, graças à maturidade alcançada na área de reconhecimento de objetos baseado em features visuais [Nistér e Stewenius 2006]. Neste arcabouço, recorrese ao uso de features mais elaboradas como SIFT [Lowe 2004] e SURF [Bay et al. 2008] para computar pontos característicos e descritores associados invariantes à rotação e escala. Correspondências são estabelecidas através de buscas por vizinhos mais próximos [Muja e Lowe 2014] no espaço dos descritores, o que possui a propriedade de estabelecer associações entre imagens obtidas de pontos de vista consideravelmente diferentes. Apesar de possuir um custo computacional mais elevado em relação a features mais simples de serem extraídas, esta estratégia de casamento por linha de base larga é comumente utilizada para detectar fechamento de laços, mesmo em sistemas com sensores de profundidade [Galvez-Lopez e Tardos 2011, Liu e Zhang 2012, Newman et al. 2006, Cummins e Newman 2011].

#### 3.3.2 Otimização em Grafo de Poses

Modelando-se as poses do sensor como vértices de um grafo e as transformações relativas entre as poses como arestas, a minimização do erro pode ser resolvida em grafos, onde se busca encontrar o melhor valor para os vértices dadas as restrições entre eles [Lu e Milios 1997]. Ao detectar fechamento de laços, uma relação entre vértices correspondentes a poses não consecutivas é estabelecida, adicionando ao sistema informações importantes à respeito da configuração mais provável dos vértices no grafo. Esta forma

de forçar consistência global na reconstrução é conhecida como SLAM baseada em grafos [Lu e Milios 1997, Thrun e Leonard 2008, Grisetti et al. 2010], possuindo formulações diversas de mínimos quadrados não-linear na literatura, como *graphSLAM* [Thrun e Montemerlo 2006], TORO (*Tree based netwORk Optimizer*) [Grisetti et al. 2009] ou bibliotecas com algoritmos configuráveis, como a G2O (*General Graph Optimization*) [Kümmerle et al. 2011].

#### 3.3.3 Minimização do Erro de Reprojeção

Nos sistemas visuais, a solução que minimiza o erro acumulado conjunto entre posições de features e parâmetros de poses de câmeras recebe o nome de *bundle adjustment* (BA) [Triggs et al. 2000, Lourakis e Argyros 2009]. De maneira não-linear, o somatório do erro de reprojeção das features em cada câmera em que ela é visível é minimizado. Explorando a estrutura esparsa do sistema, é possível resolvê-lo em tempo tratável, mantendo assim a consistência global da reconstrução 3D obtida, como comumente empregado em sistemas de SFM. Aplicações de tempo real requerem que a minimização seja executada para um subconjunto das imagens adquiridas. Também, demonstra-se experimentalmente que BA desempenha o papel equivalente aos executados por filtros probabilísticos de forma mais eficiente [Strasdat et al. 2010], o que motivou o seu uso em diversas aplicações de SLAM visual [Mouragnon et al. 2006, Klein e Murray 2007, Konolige e Agrawal 2007, Konolige e Agrawal 2008, Strasdat et al. 2011].

Embora SLAM baseado em grafos possa ser aplicável a sistemas visuais, o BA pode minimizar o erro na estimativa da profundidade dos pontos e também de escala relativa entre reconstruções (comum em sistemas com uma única câmera), sendo assim mais adequado a métodos que possui incerteza considerável nas coordenadas 3D dos pontos. Por sua vez, SLAM com grafos possui custo computacional menor devido ao menor número de parâmetros a serem otimizados.

#### 3.4 Reconstrução 3D com Sensores RGB-D

Com o advento no mercado das chamadas câmeras RGB-D, algoritmos passaram a ser propostos para explorar as capacidades oferecidas por estes sensores, detalhados em estudos sobre precisão e modelagem probabilística do erro de cálculo de profundidade [Konolige e Mihelich 2011, Khoshelham e Elberink 2012]. O principal atrativo destes dispositivos é a captura de imagens e mapas de profundidade densos com resolução de 640 × 480 a uma taxa de aquisição de 30 Hz. Desta forma, features e descritores no espaço de imagens e também no espaço de nuvens de pontos [Aldoma et al. 2012] podem ser computados para serem usados em reconhecimento de objetos [Gomes et al. 2013], análise de atividade humana [Han et al. 2013] e mapeamento 3D.

Várias abordagens estão presentes na literatura para obtenção de registro de nuvens de pontos com câmeras RGB-D. Os algoritmos propostos podem ser divididos em soluções densas, que computam poses de câmera levando em consideração cada pixel RGB-D, ou soluções esparsas, que computam poses de câmera com base em features detectadas na imagem.

Odometria visual densa com sensores RGB-D pode ser computada pela otimização de uma função custo que maximiza a foto-consistência entre duas imagens RGB-D adjacentes [Steinbruecker et al. 2011], no chamado método direto. Esta formulação é posteriormente estendida em duas outras versões: uma primeira [Kerl et al. 2013b], para comportar modelos probabilísticos de movimento de câmera, tornando-o robusto a cenas não estáticas e uma segunda [Kerl et al. 2013a], para unir a odometria visual proposta com a minimização de erro de mapeamento em grafos de poses com resolução do problema de SLAM utilizando a biblioteca G2O. No algoritmo chamado KinectFusion [Newcombe et al. 2011], a leitura atual do sensor é registrada com um modelo volumétrico da cena armazenada em uma representação chamada de Truncated Signed Distance Function (TSDF). A pose que alinha os dois sistemas de coordenadas é computada entre a nuvem de pontos atual e uma nuvem extraída da TSDF por traçado de raios (raycasting) via ICP, ou seja, dados de aparência não são usados. A extensão do KinectFusion chamada Kintinuous [Whelan et al. 2012] visa aumentar a área mapeada, fazendo com que o volume correspondente à TSDF seja móvel. Em outro estudo relacionado ao KinectFusion [Whelan et al. 2013], são demonstrados os benefícios em se usar dados de aparência no registro das nuvens de pontos. São experimentadas combinações entre ICP denso, registro com odometria visual baseada em features [Huang et al. 2011] e uma versão própria do alinhamento baseado na maximização da foto-consistência [Steinbruecker et al. 2011]. Para manter desempenho em tempo real, métodos derivados do KinectFusion dependem de processamento paralelo massivo e por isso, são implementados em GPU (Graphics Processing Unit). No trabalho de Osteen et al. (2012), correlações entre representações no domínio da frequência dos vetores normais às superfícies são usadas para computar a rotação que alinha nuvens de pontos adjacentes, sendo esta estimativa inicial refinada com ICP. Stückler e Behnke (2012) propõem que distribuições de dados de aparência e da superfície sejam amostrados em uma estrutura de dados do tipo octree para obter uma representação em multirresolução da cena visualizada. Com uma variante do ICP, o algoritmo é capaz registrar nuvens de pontos com a representação da cena.

Os trabalhos que computam estimativas de pose de forma esparsa seguem uma sequência básica bem definida, variando os algoritmos utilizados em cada etapa. Este esquema geral está divido em:

- 1. Utilizar um extrator de características visuais e uma forma de estabelecer correspondências entre features de diferentes imagens
- 2. Computar a pose de câmera entre um par de imagens
- 3. Detectar fechamento de laços com as features visuais
- 4. Minimizar o erro de mapeamento no grafo de poses formado por um subconjunto de imagens RGB-D, chamadas de *keyframes*

Features SIFT extraídas em GPU [Wu 2007] são utilizadas no trabalho de Henry et al. (2010) para computar uma estimativa de pose inicial com RANSAC e refiná-la minimizando uma função custo baseada no ICP dividida em uma parte relacionada aos dados

de aparência e outra relacionada às coordenadas 3D. Fechamento de laços são detectados comparando os descritores da imagem atual com os de um subconjunto de keyframes, adicionando assim restrições a um grafo de poses com erro minimizado com o algoritmo TORO. Estudos complementares [Henry et al. 2012] foram feitos alterando a feature detectada para o operador de cantos FAST (Features from Accelerated Segment Test) [Rosten e Drummond 2006] em conjunto com descritores treinados previamente [Calonder et al. 2008]. Além disso, extraindo dados de profundidade do ICP, uma formulação para BA é proposta para minimizar o erro de reprojeção de pontos que não possuem uma feature visual associada. O trabalho de Hu et al. (2012) propõe o uso do algoritmo dos 8 pontos [Hartley e Zisserman 2004] da geometria projetiva para, em conjunto com as nuvens de pontos 3D, estimar movimentos de câmera com RANSAC em situações onde a profundidade dos pontos não está disponível (fato que pode acontecer em cenas com profundidade maiores do que o alcance do sensor). Features FAST e minimização de erro com BA são partes do processo, que também une submapas com diferentes escalas [Zhao et al. 2010]. Paton e Kosecka (2012) implementam odometria visual com uma heurística para escolha do algoritmo de alinhamento (ICP, RANSAC ou os dois) usando features SIFT. O algoritmo é de odometria visual e portanto, não aplica minimização de erro acumulado. Dryanovski et al. (2013) seguem uma abordagem um pouco diferente do esquema básico, na qual a nuvem de pontos atual é alinhada com um mapa (e não com outra nuvem). Para isso, as matrizes de covariância relativas às incertezas das posições 3D de features Shi-Tomasi são obtidas considerando estudos anteriores sobre a precisão do sensor [Khoshelham e Elberink 2012] e armazenadas em uma árvore do tipo KD tree. O registro acontece com uma variante do ICP que minimiza a distância de Mahalanobis. O algoritmo assume que erros não acumulam no mapa e por isso, não aplica minimização global.

No sistema RGB-D SLAM proposto por Endres et al. (2012), são experimentados vários tipos de features e descritores para obter correspondências visuais entre nuvens de pontos adjacentes. Uma versão do algoritmo extrai features ORB (Oriented FAST and Rotated BRIEF) [Rublee et al. 2011], que combina o extrator de pontos FAST com o descritor BRIEF (Binary Robust Independent Elementary Features) [Calonder et al. 2010], e duas outras versões são implementadas com SURF e SIFT em GPU (sendo esta a escolhida para funcionamento em tempo real). Henry et al. observam em seus trabalhos [Henry et al. 2010, Henry et al. 2012] que o ICP tem custo computacional considerável e nem sempre é necessário, visto que a estimativa inicial do movimento de câmera computada com RANSAC é muitas vezes de boa qualidade. Assim, RGB-D SLAM emprega apenas RANSAC no alinhamento das nuvens de pontos, detectando fechamento de laços por correspondências entre a imagem atual e um conjunto de keyframes. O grafo de poses composto pelos keyframes é submetido a um processo final de minimização de erro modelado por full SLAM e resolvido com a biblioteca G2O. Huang et al. (2011) propõem um sistema de odometria visual denominado FOVIS (Fast Odometry from VISion), que é usado em conjunto com o módulo de mapeamento do sistema de Henry et al. de forma assíncrona. FOVIS funciona extraindo features FAST e obtendo correspondências entre imagens adjacentes por correlação entre patches de imagens. Alternativamente ao uso de RANSAC e ICP como realizado pelas outras abordagens, FOVIS registra nuvens de pontos adjacentes estimando a rotação e translação de forma direta [Horn 1987], após uma estimativa inicial da rotação ser computada. O algoritmo é robusto a falsas correspondências computando o clique maximal em um grafo onde os vértices são os pontos 3D e arestas são formadas entre dois pontos se as distâncias Euclidianas entre os pontos originais e suas versões transformadas são menores do que um limiar (já que o movimento de corpo rígido de rotação e translação preserva distâncias entre pontos). O clique maximal é então dado pelo maior subconjunto de vértices conectados que são adjacentes no grafo e pode ser obtido com um algoritmo guloso [Howard 2008]. Novamente, *keyframes* são selecionados e as poses de câmera associadas a cada um deles otimizadas incrementalmente com TORO.

A Tabela 3.1 sumariza as abordagens de registro de pontos com sensores RGB-D. Nela, são mostrados os algoritmos discutidos, se requerem GPU para funcionamento em tempo real, qual tipo de entrada do algoritmo (dados de cor, profundidade ou ambos), como cada um computa a pose de câmera e como mantêm consistência global, se for o caso. Os trabalhos estão divididos entre densos e esparsos. Note que o trabalho de Whelan et al. (2013) possui diversas combinações para realizar odometria visual e portanto, aparece duas vezes na tabela (como denso e esparso). Alguns trabalhos citados (como o de Steinbruecker et al. (2011) e o de Whelan et al. (2012)) não costam na tabela, sendo mostradas apenas as versões atualizadas dos mesmos (implementação de Kerl et al. (2013*b*) e Whelan et al. (2013) respectivamente).

Analisando os requerimentos e propriedades dos trabalhos levantados, pode-se perceber que:

- Vários métodos apresentados [Newcombe et al. 2011, Whelan et al. 2013, Henry et al. 2010, Endres et al. 2012] dependem de uso de hardware especializado para executarem em tempo real. Isto acontece até mesmo com alguns métodos esparsos, que empregam versões em GPU de extratores de features, como o SIFT.
- Dentre os métodos esparsos, apenas o FOVIS [Huang et al. 2011] utiliza rastreamento por linha de base curta ao aplicar correlação entre janelas de pixels para busca de correspondências entre features.

Rastreamento por linha de base larga é um processo computacionalmente custoso, mesmo quando realizado em GPU. Isto se deve ao fato de features serem detectadas e respectivos descritores extraídos em cada nova imagem, em um processo de rastreamento por detecção. Alternativamente, as posições na imagem de features podem ser implicitamente determinadas por rastreamento, sem ser necessário que elas sejam detectadas novamente para isso. Ainda, rastreamento por detecção é sujeito a um número maior de falsas correspondências, o que implica em alinhamentos entre nuvens de pontos de pior qualidade e maiores tempos de processamento no RANSAC, já que erros de casamento entre features deterioram a busca pela transformação de interesse.

Com o objetivo de registrar nuvens de pontos de sensores RGB-D de forma eficiente, esta tese propõe o uso de rastreamento por linha de base curta, inserindo-se assim entre os trabalhos de abordagem esparsa. Mais especificamente, são exploradas propriedades presentes no KLT em uma estratégia de rastreamento de alto nível, resultando em um sistema de odometria visual capaz de executar na taxa de aquisição de dados do sensor

(30 Hz) com desempenho equivalente ao estado da arte. Apesar do rastreamento por linha de base curta já estar presente na literatura [Huang et al. 2011], não há registro de trabalhos que apliquem fluxo óptico para realizar esta tarefa. Em uma última etapa do algoritmo, a odometria visual computada pelo rastreador de features proposto é usada para construir um grafo de poses de *keyframes* a ser otimizado com G2O, minimizando o erro de mapeamento acumulado.

O fluxo óptico KLT é o rastreamento empregado em diversos sistemas visuais, como odometria visual estéreo [Moreno et al. 2007, Tamura et al. 2009], reconstrução 3D de cenas urbanas a partir de vídeo [Akbarzadeh et al. 2006], controle servo visual [Choi et al. 2008], realidade aumentada [Lee e Hollerer 2008], construção de mosaicos [DiVerdi et al. 2008] e estimação de movimento de usuários em realidade mista [DiVerdi e Hollerer 2008].

Uma vez que esta forma de rastreamento é incapaz de realizar casamentos por linha de base larga, associações de dados entre imagens de entrada e um mapa são realizadas pela detecção de um marcador artificial de realidade aumentada [Garrido-Jurado et al. 2014]. Um trabalho presente na literatura [Mihalyi et al. 2013] utiliza diversos marcadores artificiais no registro de nuvens de pontos RGB-D. Entretanto, o algoritmo não foca nos requisitos de aplicações em tempo real e por isso, implementa o processo em uma etapa de calibração entre os marcadores seguida pelo registro com consistência global, minimizando o erro também com algoritmos fornecidos pela biblioteca G2O.

Tabela 3.1: Trabalhos relacionados a registro de nuvens de pontos com sensores RGB-D. As colunas exibem o método, se requerem GPU para funcionamento em tempo real, qual a entrada do algoritmo, como cada um computa a pose de câmera e qual a biblioteca/algoritmo utilizado para manter consistência global. A tabela está dividida entre algoritmos que registram imagens utilizando os dados de todos os pixels presentes (abordagens densas) e algoritmos que registram imagens utilizando features extraídas das imagens (abordagens esparsas).

Trabalho	GPU	Entrada	Cálc. Pose	Otimização
Abordagens Densas:				
[Newcombe et al. 2011]	Sim	Profundidade	ICP+TSDF	<u>—</u>
[Kerl et al. 2013 <i>b</i> ]	Não	Cor/prof.	Direto	_
[Kerl et al. 2013 <i>a</i> ]	Não	Cor/prof.	Direto	G2O
[Stückler e Behnke 2012]	Não	Cor/prof.	ICP+Mapa	G2O
[Osteen et al. 2012]	Não	Normais	ICP	
[Whelan et al. 2013]	Sim	Cor/prof.	(Direto/ICP)+TSDF	_
Abordagens Esparsas:				
[Hu et al. 2012]	Não	FAST+Não inf.	RANSAC	BA
[Henry et al. 2010]	Sim	SIFT	RANSAC+ICP	TORO
[Henry et al. 2012]	Não	FAST+Calonder	RANSAC+ICP	BA/TORO
[Paton e Kosecka 2012]	Não	SIFT	RANSAC+ICP	_
[Dryanovski et al. 2013]	Não	Shi-Tomasi	ICP+Mapa	_
[Whelan et al. 2013]	Sim	FAST+Correlação	Clique/ICP	_
[Endres et al. 2012]	Sim	SIFT	RANSAC	G2O
[Huang et al. 2011]	Não	FAST+Correlação	Clique	TORO
Proposto	Não	KLT	RANSAC	G2O

# Capítulo 4

# Registro de Nuvens de Pontos RGB-D por SLAM Visual

Nesta tese, nuvens de pontos RGB-D são incrementalmente registradas utilizando correspondências estabelecidas entre features visuais. Após o rastreamento das features extraídas em um instante de tempo anterior nas suas correspondências na imagem atual, o registro pode ser computado entre o par de nuvens composto pela nuvem atual e a nuvem imediatamente anterior utilizando os casamentos estabelecidos em um esquema robusto a falsas correspondências. As transformações rígidas obtidas são acumuladas, resultando na pose de câmera para o instante de tempo atual e também no registro de pontos no sistema de coordenadas global definido pela câmera relacionada à primeira imagem RGB-D. Poses de câmera são calculadas com 6 graus de liberdade, dadas pela translação e rotação em torno dos eixos do referencial global. Este processo conjunto de registro de nuvens de pontos e computação de pose de câmera recebe o nome de odometria visual [Nistér et al. 2004, Scaramuzza e Fraundorfer 2011, Fraundorfer e Scaramuzza 2012]. As nuvens de pontos RGB-D constituem os únicos dados sensoriais utilizados no processo.

Algoritmos de odometria visual estão sujeitos a acúmulos de erro (conhecidos na literatura como *drift* [Scaramuzza e Fraundorfer 2011]), que ocorrem devido ao fato das mudanças de pose possuírem somente consistência local. Uma possível solução para este problema é minimizar o erro acumulado, seja usando uma janela temporal maior (minimizando erro entre um número maior de imagens adjacentes) ou em um processo de minimização global, no qual toda a trajetória computada pela odometria visual é otimizada em um método de full SLAM. Esta última alternativa somente é possível quando a associação de dados é realizada entre a imagem atual e um mapa por meio de um mecanismo de fechamento de laço. Com isto, restrições são adicionadas à estimativa inicial para que ela possa ser otimizada. Ainda, caso a trajetória inicial seja de má qualidade, a otimização pode convergir para um mínimo local com erro ainda considerável ou no pior caso, pode até mesmo não convergir.

Na implementação atual (Capítulo 5), um marcador artificial de realidade aumentada é utilizado para adicionar restrições adicionais que possibilitem a otimização. Entretanto, a solução de minimização de erro em grafo de poses [Lu e Milios 1997, Grisetti et al. 2010, Kümmerle et al. 2011] é apresentada aqui de forma genérica, visando restrições que possam ser obtidas por outras fontes.

#### 4.1 Odometria Visual

Deseja-se realizar o registro incremental de nuvens de pontos em um sistema de coordenadas global, onde cada nuvem de pontos é estimada a partir dos dados capturados por um sensor RGB-D. Em cada instante de tempo t, a imagem de aparência  $I_t(u,v)$  e de profundidade  $I_t^d(u,v)$  são utilizadas para computar a nuvem de pontos densa correspondente  $\mathcal{N}_t$  através da Equação 2.15.

Um conjunto  $\mathcal{F}_t = \{\mathbf{p}_t^0, \mathbf{p}_t^2, ..., \mathbf{p}_t^{M_t-1}\}$  de  $M_t$  pontos bidimensionais composto por features é extraído de cada imagem de aparência  $I_t(u,v)$ . Para cada ponto  $\mathbf{p}_{t-1}^k$  da imagem anterior, é obtido o seu par  $\mathbf{p}_t^k$  por fluxo óptico na imagem atual, originando  $C_{t-1,t}$  correspondências entre as características visuais. A partir destas, nuvens de pontos esparsas  $\mathcal{N}_{t-1}^f$  e  $\mathcal{N}_t^f$  são construídas pela Equação 2.15 aplicada à posição de cada feature  $\mathbf{p}_{t-1}^k$  e  $\mathbf{p}_t^k$ . Os pares de pontos 3D  $(\mathbf{P}_{t-1}^k, \mathbf{P}_t^k)$ , para  $k=0,...,C_{t-1,t}-1$  são utilizados para estimar a transformação rígida de registro. Como mostrado na Figura 4.1, este processo somente é possível quando há sobreposição entre as nuvens de pontos, isto é, o conjunto de pontos compartilhados  $\mathcal{N}_{t-1,t}$  pelas duas imagens  $I_{t-1}(u,v)$  e  $I_t(u,v)$  possui pelo menos três pontos. A transformação rígida  $\mathbf{T}_{t,t-1}$  resultante possui dois usos: a denotar a mudança de pose de câmera entre os instantes de tempo t-1 e t e b) registrar a nuvem de pontos  $\mathcal{N}_t$  no sistema de referência da nuvem  $\mathcal{N}_{t-1}$  utilizando a sua inversa,  $\mathbf{T}_{t,t-1}^{-1} = \mathbf{T}_{t-1,t}$ .

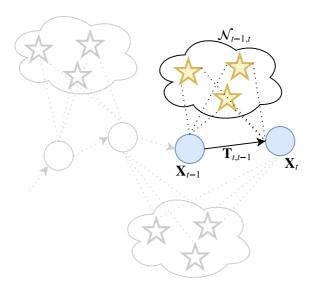


Figura 4.1: Registro incremental de nuvens de pontos. Em cada instante de tempo t, é estimada a transformação de registro  $\mathbf{T}_{t,t-1}$ , que alinha o sistema de referência em coordenadas globais da pose de câmera  $\mathbf{X}_{t-1}$  no referencial da pose  $\mathbf{X}_t$ . Este processo é possível porque um subconjunto de pontos está presente em medições sensoriais consecutivas.

Para qualquer instante de tempo específico t, a matriz de transformação homogênea

37

 $\mathbf{X}_t$  com a pose atual da câmera no sistema de coordenadas global pode ser obtida por

$$\mathbf{X}_t = \prod_{k=1}^t \mathbf{T}_{k,k-1},\tag{4.1}$$

considerando que o sistema de coordenadas global é arbitrariamente definido como o sistema de coordenadas de câmera referente à primeira imagem  $I_0(u,v)$ , inicializado com rotação igual à matriz identidade  $I_{3x3}$  e vetor translação igual a  $\mathbf{0}$ .

Também por meio da Equação 4.1, é possível obter a transformação  $\mathbf{T}_{0,t}$  que registra a nuvem de pontos densa  $\mathcal{N}_t$  em uma nuvem  $\mathcal{N}_t^w$  formada pela concatenação de todas as nuvens anteriores no sistema de referência global. Isto é possível por causa do produtório

$$\mathbf{T}_{0,t} = \mathbf{T}_{0,1}...\mathbf{T}_{t-2,t-1}\mathbf{T}_{t-1,t}.$$

A saída da odometria visual é a nuvem de pontos RGB-D total  $\mathcal{N}_t^w$ , resultante do registro incremental de todas as nuvens processadas, e a pose atual  $\mathbf{X}_t$ . Caso estejam disponíveis associações de dados adicionais, a etapa de otimização pode ser executada. Estas associações são dadas por transformações  $\mathbf{T}_{j,i}$ , onde  $i \neq j-1$ , ou seja, transformações entre poses de câmeras não consecutivas. A trajetória completa  $\mathbf{X}_{0:N} = \{\mathbf{X}_0, ..., \mathbf{X}_N\}$  é usada como entrada no processo de minimização de erro por SLAM.

# 4.1.1 Rastreamento por Fluxo Óptico Piramidal KLT

Sejam duas imagens quaisquer  $I_{t-1}(u,v)$  e  $I_t(u,v)$  adquiridas sequencialmente, a partir das quais deseja-se realizar o rastreamento. Dentro do contexto do problema (Seção 2.3.3), as duas imagens são tratadas como função do tempo,  $I_{t-1}(u,v) = I(u,v,t)$  e  $I_t(u,v) = I(u,v,t+dt)$ , ou seja, a primeira imagem foi capturada no instante de tempo t e a segunda imagem foi adquirida em t+dt, onde dt é um intervalo de tempo infinitesimal. Para facilitar a notação, define-se A(u,v) e B(u,v) como

$$A(u,v) := I_{t-1}(u,v), \quad B(u,v) := I_t(u,v).$$

O objetivo do fluxo óptico é determinar o vetor deslocamento

$$\mathbf{v} = \left[\frac{du}{dt}, \frac{dv}{dt}\right]^t := [\delta_u, \delta_v]$$

para cada posição dada por uma feature de Shi-Tomasi  $\mathbf{p} = [p,q]^t$  extraída de A(u,v), de maneira que  $\mathbf{p}'$ , as coordenadas de  $\mathbf{p}$  na imagem B(u,v), são dadas por

$$\mathbf{p}' = \mathbf{p} + \mathbf{v}$$
.

O desenvolvimento de uma solução para o fluxo óptico parte da Equação 2.9, convenientemente reescrita aqui

$$\frac{dI(u,v,t)}{dt} = \frac{\partial I}{\partial u}\frac{du}{dt} + \frac{\partial I}{\partial v}\frac{dv}{dt} + \frac{\partial I}{\partial t} = 0.$$

É suposto que a intensidade da imagem A(u, v) na posição  $\mathbf{p}$  deve ser igual à intensidade da imagem B(u, v) na posição  $\mathbf{p}'$ , ou seja,  $A(\mathbf{p})$  e  $B(\mathbf{p}')$  devem ser similares. Como discutido na Seção 2.3.3, pode-se achar uma solução para a equação em questão supondo também que o fluxo óptico é constante para uma dada vizinhança na imagem, como proposto pelo algoritmo KLT [Lucas e Kanade 1981, Shi e Tomasi 1994].

O fluxo óptico pode ser determinado para  $\mathbf{p}$ , localizado em torno de uma janela  $W = W_u \times W_v$ , por um processo iterativo de minimização de

$$\varepsilon(\mathbf{v}) = \varepsilon(\delta_u, \delta_v) = \sum_{u=p-W_u}^{p+W_u} \sum_{v=q-W_v}^{q+W_v} (A(u, v) - B(u + \delta_u, v + \delta_v))^2. \tag{4.2}$$

A solução é obtida derivando-se  $\varepsilon(\mathbf{v})$  em relação a  $\mathbf{v}$ ,

$$\frac{\partial \varepsilon}{\partial \mathbf{v}} = -2\sum \sum \left(A(u,v) - B(u + \delta_u, v + \delta_v)\right) \begin{bmatrix} \frac{\partial B}{\partial u}, & \frac{\partial B}{\partial v} \end{bmatrix},$$

onde todos os somatórios são computados nos mesmos índices da Equação 4.2 e portanto são omitidos (assim como feito nas demais equações desta derivação). Expandindo o termo  $B(u + \delta_u, v + \delta_v)$  pela sua aproximação de primeira ordem em série de Taylor na vizinhança de  $\mathbf{v} = [0,0]^t$ , obtém-se

$$\frac{\partial \mathbf{\varepsilon}}{\partial \mathbf{v}} = -2\sum \sum \left( A(u, v) - B(u, v) - \begin{bmatrix} \frac{\partial B}{\partial u}, & \frac{\partial B}{\partial v} \end{bmatrix} \mathbf{v} \right) \begin{bmatrix} \frac{\partial B}{\partial u}, & \frac{\partial B}{\partial v} \end{bmatrix}. \tag{4.3}$$

Observando-se que  $A(u,v) - B(u,v) = \Delta I$  (conforme Equação 2.10) e que o vetor gradiente da imagem B(u,v) é dado por

$$\nabla B = \left[ \frac{\partial B}{\partial u}, \frac{\partial B}{\partial v} \right]^t,$$

a Equação 4.3 pode ser reescrita como

$$\frac{\partial \mathbf{\varepsilon}}{\partial \mathbf{v}} = -2\sum \sum (\Delta I - \nabla B^t \mathbf{v}) \nabla B^t,$$

que é igual a

$$\frac{1}{2}\frac{\partial \mathbf{\varepsilon}}{\partial \mathbf{v}} = \sum \sum (\nabla B^t \mathbf{v} - \Delta I) \nabla B^t$$

após os termos serem reagrupados. O vetor linha

$$\left[\frac{\partial \mathbf{\varepsilon}}{\partial \mathbf{v}}\right]^t = \left[\begin{array}{cc} \frac{\partial \mathbf{\varepsilon}}{\partial \delta_u}, & \frac{\partial \mathbf{\varepsilon}}{\partial \delta_v} \end{array}\right]$$

é então dado por

$$\frac{1}{2} \begin{bmatrix} \frac{\partial \mathbf{\epsilon}}{\partial \mathbf{v}} \end{bmatrix}^{t} = \sum \sum \left( \begin{bmatrix} B_{u}^{2} & B_{u}B_{v} \\ B_{u}B_{v} & B_{v}^{2} \end{bmatrix} \mathbf{v} - \begin{bmatrix} \Delta I B_{u} \\ \Delta I B_{v} \end{bmatrix} \right). \tag{4.4}$$

Definindo

$$\mathbf{G} \coloneqq \sum \sum \left[ egin{array}{ccc} B_u^2 & B_u B_v \ B_u B_v & B_v^2 \end{array} 
ight] \quad \mathbf{e} \quad \mathbf{b} \coloneqq \sum \sum \left[ egin{array}{c} \Delta I \ B_u \ \Delta I \ B_v \end{array} 
ight],$$

a Equação 4.4 se torna

$$\frac{1}{2} \left[ \frac{\partial \varepsilon}{\partial \mathbf{v}} \right]^t = \mathbf{G} \mathbf{v} - \mathbf{b},\tag{4.5}$$

a qual informa que a solução v correspondente ao fluxo óptico de p é dada por

$$\mathbf{v} = \mathbf{G}^{-1}\mathbf{b}$$
.

ou seja, **v** é o vetor fluxo óptico para as coordenadas centrais **p** computado  $\forall (u, v) \in [p - W_u - 1, p + W_u + 1] \times [q - W_v - 1, q + W_v + 1].$ 

A matriz **G** possui inversa quando a intensidade da imagem em torno de **p** varia consideravelmente ao longo da direção vertical e horizontal. O uso de features de Shi-Tomasi como pontos onde devem ser computados o fluxo óptico pode garantir esta condição, desde que um tamanho de janela  $W = W_u \times W_v$  adequado tenha sido escolhido. Com o uso de pirâmides de imagens em um esquema de multirresolução, os efeitos deste problema podem ser reduzidos.

Respeitando a aproximação em série de Taylor da Equação 4.3, o vetor  $\mathbf{v}$  pode ser obtido de forma iterativa resolvendo-se a Equação 4.5 por uma solução atual e usando esta solução para transformar a imagem B(u,v) até que um critério de parada seja satisfeito. Isto é equivalente a linearizar o erro  $\varepsilon(\mathbf{v})$  em torno da solução atual e aproximar a sua derivada com melhor precisão.

Na k-ésima iteração, o ponto  $\mathbf{p}$  em B(u,v) é transladado pelo fluxo  $\mathbf{v}^{k-1} = \left[\delta_u^{k-1}, \delta_v^{k-1}\right]$  computado na iteração anterior, resultando em

$$B^{k}(u, v) = B(u + \delta_{u}^{k-1}, v + \delta_{v}^{k-1}).$$

O vetor resíduo  $\mathbf{r}^k = \left[r_u^k, r_v^k\right]^t$  entre as posições A(u, v) e  $B^k(u, v)$  deve ser computado pela minimização de

$$\varepsilon^{k}(\mathbf{r}^{k}) = \varepsilon(r_{u}^{k}, r_{v}^{k}) = \sum_{u=p-w_{u}}^{p+w_{u}} \sum_{v=q-w_{v}}^{q+w_{v}} \left( A(u, v) - B^{k}(u + r_{u}^{k}, v + r_{v}^{k}) \right)^{2}, \tag{4.6}$$

o que é alcançado pela resolução direta por mínimos quadrados de

$$\mathbf{r}^k = \mathbf{G}^{-1}\mathbf{b}^k.$$

onde

$$\mathbf{b}^k = \sum \sum \left[ \begin{array}{c} \Delta I^k \ B_u \\ \Delta I^k \ B_v \end{array} \right]$$

e  $\Delta I^k = A(u,v) - B^k(u,v)$ . A estimativa do fluxo óptico para a iteração k é então dada por

$$\mathbf{v}^k = \mathbf{v}^{k-1} + \mathbf{r}^k,$$

com  $\mathbf{v}^0 = [0,0]^t$  como solução inicial.

Como critérios de parada, utiliza-se um número máximo de iterações ou um limiar mínimo para o resíduo  $\mathbf{r}^k$ , encerrando assim o cálculo iterativo quando algum dos dois quesitos é satisfeito.

Uma implementação eficiente calcula o fluxo óptico esparso KLT com o uso de imagens em multirresolução [Bouguet 2000] para amenizar os efeitos do problema da abertura (i.e. os problemas relacionados ao tamanho da janela W). Mais especificamente, uma representação em forma de pirâmide com L níveis é construída recursivamente para uma imagem qualquer I(u,v), sendo o nível 0 da pirâmide dada pela imagem original  $I^0(u,v) := I(u,v)$  e um nível l da pirâmide formado pela imagem  $I^{l-1}(u,v)$  subamostrada ao longo de coordenadas pares de u e v. A cada nível, a largura e altura do nível anterior são reduzidas pela metade.

Para computar o fluxo óptico em multirresolução, as pirâmides são construídas para ambas as imagens A(u,v) e B(u,v). O vetor deslocamento é computado para as features no nível de menor resolução das imagems A(u,v) e B(u,v), isto é, nas imagens  $A^{L-1}(u,v)$  e  $B^{L-1}(u,v)$ , com o resultado sendo utilizado de forma recursiva como estimativa inicial do fluxo em níveis sucessivos da pirâmide,  $A^{L-2}(u,v)$ .... e  $B^{L-2}(u,v)$ ...., até o primeiro nível  $A^0(u,v)$  e  $B^0(u,v)$ . O vetor deslocamento acumulado no primeiro nível é o fluxo óptico do ponto  $\bf p$ , observando-se que o fluxo é estimado de forma iterativa em cada nível. Com o uso do esquema de multirresolução para computar o fluxo óptico, movimentos aparentes maiores são suportados pelo algoritmo, ao mesmo tempo em que é mantida a precisão do cálculo do vetor deslocamento relacionado a cada feature.

### 4.1.2 Registro entre Pares de Nuvens de Pontos com Features Visuais

Com o rastreamento das features visuais computado pelo fluxo óptico, as  $C_{t-1,t}$  correspondências entre a nuvem de pontos esparsa do instante de tempo anterior  $\mathcal{F}_{t-1}$  e a nuvem esparsa atual  $\mathcal{F}_t$  são empregadas no cálculo da transformação rígida de registro. Considerando que  $\mathcal{F}_{t-1} = \{\mathbf{P}_{t-1}^0,...,\mathbf{P}_{t-1}^{C_{t-1,t}-1}\}$  e  $\mathcal{F}_t = \{\mathbf{P}_t^0,...,\mathbf{P}_t^{C_{t-1,t}-1}\}$  possuem mesmo tamanho  $C_{t-1,t}$ , e que um ponto  $\mathbf{P}_{t-1}^i$  de  $\mathcal{F}_{t-1}$  tem a sua correspondência no conjunto  $\mathcal{F}_t$  dada pelo ponto de mesmo índice  $\mathbf{P}_t^i$ , o registro entre os pares de nuvens de pontos pode

ser computado resolvendo a Equação 2.16,

$$\mathbf{R}_{a,b}, \mathbf{t}_{a,b} = \underset{\mathbf{R}, \mathbf{t}}{\operatorname{argmin}} \sum_{k}^{M} ||\mathbf{P}_{a}^{k} - \left(\mathbf{R}\mathbf{P}_{b}^{k} + \mathbf{t}\right)||^{2}, \tag{4.7}$$

em uma solução por mínimos quadrados [Umeyama 1991].

Para simplificar a notação,  $M := C_{t-1,t}$ ,  $\mathcal{N}_a := \mathcal{F}_{t-1}$  e  $\mathcal{N}_b := \mathcal{F}_t$ . O vetor média (centróides de cada nuvem) são computados em um primeiro passo por

$$\mu_a = \frac{1}{M} \sum_{k=0}^{M-1} \mathbf{P}_a^k$$

e

$$\mu_b = \frac{1}{M} \sum_{k=0}^{M-1} \mathbf{P}_b^k.$$

Em seguida, a matriz de covariância  $\Sigma_{a,b}$  das duas nuvens de pontos é calculada por

$$\Sigma_{a,b} = \frac{1}{M} \sum_{k=0}^{M-1} (\mathbf{P}_a^k - \mu_a) (\mathbf{P}_b^k - \mu_b)^t,$$

e utilizada para computar a rotação  $\mathbf{R}_{a,b}$  que alinha os pontos da nuvem  $\mathcal{N}_b$  na nuvem  $\mathcal{N}_a$ . Sendo o produto de matrizes  $\mathbf{UDV}^t$  uma decomposição em valores singulares [Hartley e Zisserman 2004] de  $\Sigma_{a,b}$ , a rotação  $\mathbf{R}_{a,b}$  que minimiza a Equação 2.16 é dada por

$$\mathbf{R}_{ab} = \mathbf{U}\mathbf{S}\mathbf{V}^t$$
,

onde **S** é a matriz identidade se o determinante de  $\Sigma_{a,b}$  for maior ou igual a 0 ou **S** é diag(1,1,-1), com diag(.) sendo uma matriz diagonal, caso contrário. O vetor solução do registro  $\mathbf{t}_{a,b}$  é dado por

$$\mathbf{t}_{a,b} = \mu_a - \mathbf{R}_{a,b}\mu_b$$

com a transformação rígida em forma de matriz homogênea  $\mathbf{T}_{t-1,t}$  construída a partir de  $\mathbf{R}_{t-1,t} := \mathbf{R}_{a,b}$  e  $\mathbf{t}_{t-1,t} := \mathbf{t}_{a,b}$ .

# 4.2 SLAM por Minimização do Erro em Grafo de Pose

Forçar consistência global às nuvens de pontos registradas incrementalmente consiste da etapa final do algoritmo, realizada após toda a trajetória  $\mathbf{X}_0,...,\mathbf{X}_N$  ter sido estimada pela odometria visual e restrições adicionais estarem disponíveis na cena mapeada. A trajetória  $\mathbf{X}_0,...,\mathbf{X}_N$  é parametrizada por vetores  $\mathbf{x}_0,...,\mathbf{x}_N$ , com a pose referente à primeira imagem  $\mathbf{x}_0$  sendo a origem do sistema de referência global e por isso, mantida fixa ao longo da otimização.

Ao abordar o problema sob a teoria de SLAM, um vasto ferramental matemático bem fundamentado em modelos probabilísticos [Durrant-Whyte e Bailey 2006, Bailey

e Durrant-Whyte 2006, Thrun et al. 2005, Thrun e Leonard 2008, Lu e Milios 1997, Grisetti et al. 2010, Kümmerle et al. 2011] pode ser empregado, resultando em soluções robustas a ruídos de diferentes origens presentes no processo. Com o *full* SLAM, o problema pode ser resolvido explorando-se a estrutura esparsa intrínseca às matrizes a serem otimizadas [Grisetti et al. 2010], de forma que soluções precisas podem ser computadas eficientemente por rotinas numéricas [Marquardt 1963, Davis 2006].

Para resolver o problema de *full* SLAM, os parâmetros que descrevem a densidade de probabilidade da Equação 2.20,

$$p(\mathbf{x}_{1:N}, \mathcal{M}|\mathbf{z}_{1:N}, \mathbf{u}_{1:N}),$$

devem ser computados. Considerando que no caso desta tese o método não é restrito à aplicação em robótica, não há portanto dependência dos comandos de controle  $\mathbf{u}_{1:N}$  do robô. Também, não há um mapa  $\mathcal{M}$  no qual pontos 3D são mantidos explicitamente como pontos de referência com os quais devem ser associados os dados da imagem atual. O papel semântico de mapa é desempenhado aqui por um grafo, que por sua vez leva indiretamente em consideração marcos visuais associados visualmente pelo sensor, abstraindo estes dados em forma de restrições entre as poses. Não manter um mapa formado por marcos visuais é uma das razões pelas quais o algoritmo alcança desempenho eficiente [Grisetti et al. 2010]. É assumido que na sequência de observações  $\mathbf{z}_{1:N}$  obtida ao longo da trajetória do sensor, mais de uma observação pode ter sido coletada para um único instante de tempo.

A densidade de probabilidade de interesse passa a ser então

$$p(\mathbf{x}_{1:N}|\mathbf{z}_{1:N}),\tag{4.8}$$

a partir da qual deve ser computada a sua média, dados os vetores denotando todas as poses ao longo da trajetória  $\mathbf{x}_{1:N}$  e observações sensoriais  $\mathbf{z}_{1:N}$ . Sem perda de generalidade, os índices 1:N serão omitidos de  $\mathbf{x}_{1:N}$  e  $\mathbf{z}_{1:N}$  no que segue. Pela regra de Bayes,

$$p(\mathbf{x}|\mathbf{z}) = \frac{p(\mathbf{z}|\mathbf{x})p(\mathbf{x})}{p(\mathbf{z})},$$

onde o modelo de medição  $p(\mathbf{z}|\mathbf{x})$  (Equação 2.18) é também a verossimilhança (*like-lihood*) das observações condicionada à trajetória. Nenhuma informação à respeito de  $p(\mathbf{x})$  é assumida, fazendo com que esta probabilidade assuma um valor constante qualquer;  $p(\mathbf{z})$  também assume um valor constante, já que para o nosso propósito, esta grandeza tem efeito somente de normalização. Portanto,

$$p(\mathbf{x}|\mathbf{z}) \propto p(\mathbf{z}|\mathbf{x}),$$

o que indica que pode-se trabalhar em cima do modelo de sensor para computar a média da probabilidade  $p(\mathbf{x}|\mathbf{z})$ .

Assume-se que cada uma das medições obtidas  $\mathbf{z}_k$ , com k = 1, ..., N são independentes,

implicando em

$$p(\mathbf{z}|\mathbf{x}) = \prod_{k=1}^{N} p(\mathbf{z}_k|\mathbf{x}). \tag{4.9}$$

Assume-se também que estas medições estão corrompidas por um ruído com média igual a 0 e distribuído normalmente, ou seja, a distribuição  $p(\mathbf{z}_k|\mathbf{x})$  é dada por

$$p(\mathbf{z}_k|\mathbf{x}) \propto \exp(-(\mathbf{\hat{z}}_k - \mathbf{z}_k)^t \Omega_k(\mathbf{\hat{z}}_k - \mathbf{z}_k)),$$

onde  $\hat{\mathbf{z}}_k = h_k(\mathbf{x})$  é o valor previsto para a observação  $\mathbf{z}_k$  a partir da trajetória  $\mathbf{x}$  e  $\Omega_k$  é a matriz de informação obtida pelo inverso da matriz de covariância  $\Sigma_k$  da distribuição, isto é,  $\Omega_k = \Sigma_k^{-1}$ . Assim, a verossimilhança da Equação 4.9 também é uma distribuição normal,

$$p(\mathbf{z}|\mathbf{x}) = \prod_{k=1}^{N} \exp(-(\hat{\mathbf{z}}_k - \mathbf{z}_k)^t \Omega_k(\hat{\mathbf{z}}_k - \mathbf{z}_k)). \tag{4.10}$$

A maximização da verossimilhança  $p(\mathbf{z}|\mathbf{x})$ , resulta no conjunto de parâmetros  $\mathbf{x}_{1:N}^*$ , que melhor se ajustam às observações  $\mathbf{z}_{1:N}$ . Por causa da relação pela regra de Bayes, a trajetória  $\mathbf{x}_{1:N}^*$  é também a média da probabilidade *a posteriori*  $p(\mathbf{x}|\mathbf{z})$  [Grisetti et al. 2010], que é exatamente o objetivo do *full* SLAM. Assim, aplicando o logaritmo natural na Equação 4.10, obtém-se a equação

$$p(\mathbf{z}|\mathbf{x}) = \sum_{k=1}^{N} (-(\mathbf{\hat{z}}_k - \mathbf{z}_k)^t \Omega_k(\mathbf{\hat{z}}_k - \mathbf{z}_k)),$$

cuja maximização é equivalente à minimização de

$$F(\mathbf{x}) = \sum_{k=1}^{N} (\hat{\mathbf{z}}_k - \mathbf{z}_k)^t \Omega_k (\hat{\mathbf{z}}_k - \mathbf{z}_k). \tag{4.11}$$

Posta a dedução da verossimilhança a ser otimizada com *full* SLAM, procede-se em relacionar as grandezas estabelecidas nesta formalização com o seu equivalente na mode-lagem do problema como otimização em grafo de poses. Isto é importante para esclarecer tanto como é construído o grafo, quanto para explicar como a minimização não-linear é resolvida de forma eficiente.

#### 4.2.1 Grafo de Pose

Um grafo [Bondy e Murty 1976] é uma estrutura matemática formada por um conjunto de vértices  $\mathcal{V}$  e um conjunto de arestas  $\mathcal{E}$ . No contexto de SLAM [Lu e Milios 1997, Grisetti et al. 2010, Kümmerle et al. 2011], grafos são construídos a partir dos dados obtidos pela trajetória inicial  $\mathbf{x}_{1:N}$  e observações  $\mathbf{z}_{1:N}$ , com cada vértice de  $\mathcal{V}$  sendo uma pose de câmera  $\mathbf{x}_t$ , para t=1,...,N, e cada aresta sendo uma restrição entre dois vértices do grafo em forma de uma função densidade de probabilidade. Especificamente, uma aresta conecta os vértices  $\mathbf{x}_i$  e  $\mathbf{x}_j$  por uma densidade de probabilidade normal com média dada pelo vetor  $\mathbf{z}_{i,j}$  e covariância dada pela matriz de informação  $\Omega_{i,j}$  caso exista uma rela-

ção entre os dois vértices. As transformações  $\mathbf{T}_{t,t-1}$ , para t=1,...,N, obtidas no registro incremental, constituem um tipo possível de relação, que também pode ser formada por qualquer  $\mathbf{T}_{j,i}$  com  $i \neq j-1$  (transformações entre vértices não sequenciais), caso algum mecanismo de associação de dados seja capaz de determinar correspondências entre os marcos visuais presentes nas nuvens de pontos  $\mathcal{N}_i$  e  $\mathcal{N}_j$ , possibilitando assim estimar  $\mathbf{T}_{j,i}$ . O arcabouço também é genérico o suficiente para suportar restrições adicionais formadas por dados de outros sensores [Kümmerle et al. 2011], embora esta tese explore somente sensores RGB-D.

Arestas entre vértices não sequenciais são estritamente necessárias para o processo de otimização ser realizado. As arestas do grafo são direcionadas, denotando que para uma restrição dada por  $\mathbf{z}_{i,j}$  e  $\Omega_{i,j}$ , uma observação do vértice  $\mathbf{x}_j$  é obtida a partir do referencial de  $\mathbf{x}_i$ . Intuitivamente, múltiplas arestas incidindo em um mesmo vértice indica que há mais de uma hipótese para a pose do mesmo, ou seja, existe mais de uma observação à respeito da grandeza a ser otimizada. Um grafo exemplo é mostrado na Figura 4.2.

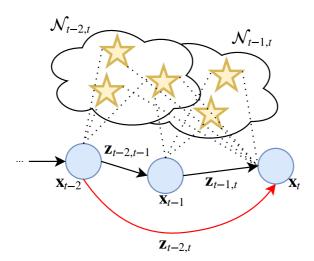


Figura 4.2: No SLAM formulado por minimização em grafo de poses, a estrutura semântica do mapa armazena um conjunto de vértices  $\mathcal{V}$ , formado pelas poses do sensor  $\mathbf{x}_t$ , e um conjunto de arestas  $\mathcal{E}$ , formado por restrições  $\mathbf{z}_{i,j}$  entre um par de poses. A imagem mostra três vértices, com duas arestas entre vértices sequenciais  $\mathbf{z}_{t-2,t-1}$  e  $\mathbf{z}_{t-1,t}$ , obtidas por odometria visual, e uma aresta  $\mathbf{z}_{t-2,t}$  obtida por algum mecanismo de associação de dados entre entre vértices não sequenciais.

Para facilitar a notação entre as relações dos vértices, é introduzido o operador de composição de movimento utilizado na formulação de Lu e Milios (1997), dado por

$$\mathbf{x}_{i} = \mathbf{x}_{i} \oplus \mathbf{T}_{i,i} \tag{4.12}$$

e o seu operador inverso,

$$\mathbf{T}_{i,i} = \mathbf{x}_i \ominus \mathbf{x}_i \tag{4.13}$$

Sendo  $T_{j,i}$  uma transformação relativa de referencial, o operador  $\oplus$  informa o sistema de coordenadas resultante, enquanto o operador inverso  $\ominus$  denota qual a transformação

relativa que existe entre dois dados sistemas de coordenadas. A transformação rígida

$$\mathbf{X}_j = \mathbf{T}_{j,i} \mathbf{X}_i,$$

é equivalente ao operador ⊕ e

$$\mathbf{T}_{j,i} = \mathbf{X}_j^{-1} \mathbf{X}_i$$

realiza a mesma transformação que  $\ominus$ , caso os vértices sejam parametrizados com matrizes homogêneas.

Na dedução do *full* SLAM, um único  $\mathbf{z}_k$  pode representar sem perda de generalidade múltiplas observações, já que em geral várias medições sensoriais são obtidas em um mesmo instante de tempo. Com a notação  $\mathbf{z}_{i,j}$  utilizada no grafo, o passo de tempo é removido, sendo explicitados os índices sobre os quais existe uma observação. O mesmo vale para o valor esperado para a observação de acordo com a configuração da trajetória  $\hat{\mathbf{z}}_{i,j} = h_{i,j}(\mathbf{x})$ , que pode ser agora explicitado por

$$\hat{\mathbf{z}}_{i,j} = h_{i,j}(\mathbf{x}) = \mathbf{x}_j \ominus \mathbf{x}_i.$$

Um conjunto C armazena todos os pares de índices i e j para os quais existem uma aresta no grafo.

A Equação 4.11 a ser minimizada no *full* SLAM é então desenvolvida com grafos de pose. O erro  $\mathbf{e}_{i,j}$  entre o valor esperado  $\mathbf{\hat{z}}_{i,j}$  e uma observação  $\mathbf{z}_{i,j}$  entre dois vértices  $\mathbf{x}_i$  e  $\mathbf{x}_j$  é definido por

$$\mathbf{e}_{i,j}(\mathbf{x}_i,\mathbf{x}_j) = \mathbf{\hat{z}}_{i,j} - \mathbf{z}_{i,j}. \tag{4.14}$$

Usando esta nova medida na Equação 4.11 e substituindo-se os índices genéricos k pelos índices i, j, a função a ser minimizada  $F(\mathbf{x})$  pode ser reescrita de forma adequada à modelagem com grafos por

$$F(\mathbf{x}) = \sum_{\langle i,j \rangle \in \mathcal{C}} \mathbf{e}_{i,j}^t \Omega_{i,j} \mathbf{e}_{i,j}, \tag{4.15}$$

onde cada termo do somatório pode ser escrito como

$$F_{i,j}(\mathbf{x}) = \mathbf{e}_{i,j}^t \Omega_{i,j} \mathbf{e}_{i,j}. \tag{4.16}$$

Note que caso uma única observação  $\mathbf{z}_{i,q}$  envolvendo um vértice específico  $\mathbf{x}_q$  esteja disponível, o somatório do erro para o vértice se resume a uma única parcela  $F_{i,q}$ , com  $\mathbf{e}_{i,q}(\mathbf{x}_i,\mathbf{x}_q) = \hat{\mathbf{z}}_{i,q} - \mathbf{z}_{i,q}$  na qual o valor previsto  $\hat{\mathbf{z}}_{i,q}$  é igual à observação  $\mathbf{z}_{i,q}$ , ou seja, o vetor erro para vértice é o vetor nulo. Este é exatamente o motivo pelo qual a otimização em grafo não pode ser aplicada à trajetória obtida pela odometria visual sem restrições adicionais.

#### 4.2.2 Minimização Não-Linear

A Equação 4.15 encontra-se na forma de mínimos quadrados, o que implica no fato de que o vetor  $\mathbf{x}^*$  que a minimiza pode ser calculado pela sua derivada em relação a  $\mathbf{x}$ , ou

seja,

$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmin}} F(\mathbf{x}). \tag{4.17}$$

Devido a cada termo  $\mathbf{e}_{i,j}$  na Equação 4.15 depender de  $\mathbf{\hat{z}}_{i,j} = h_{i,j}(\mathbf{x})$ , com cada  $h_{i,j}$  sendo uma função não-linear, procede-se com uma solução de minimização iterativa [Grisetti et al. 2010, Kümmerle et al. 2011]. Para uma solução inicial da trajetória  $\mathbf{\check{x}}$ , o erro  $\mathbf{e}_{i,j}$  na restrição associada pode ser aproximado pela sua expansão em série de Taylor de primeira ordem na vizinhança de  $\mathbf{\check{x}}$  por

$$\mathbf{e}_{i,j}(\check{\mathbf{x}}_i + \Delta \mathbf{x}_i, \check{\mathbf{x}}_j + \Delta \mathbf{x}_j) = \mathbf{e}_{i,j}(\check{\mathbf{x}} + \Delta \mathbf{x}) \simeq \mathbf{e}_{i,j} + \mathbf{J}_{i,j}\Delta \mathbf{x},$$

onde  $\mathbf{e}_{i,j} := \mathbf{e}(\mathbf{x})$  é o erro da solução inicial  $\mathbf{x}$  e  $\mathbf{J}_{i,j}$  é o Jacobiano de  $\mathbf{e}_{i,j}(\mathbf{x})$  computado também em  $\mathbf{x}$ ,

$$\mathbf{J}_{i,j} = \frac{\partial \mathbf{e}_{i,j}(\mathbf{x})}{\partial \mathbf{x}} \bigg|_{\mathbf{x} = \check{\mathbf{x}}}.$$
(4.18)

Substituindo-se  $\mathbf{e}_{i,j}(\mathbf{x}_i + \Delta \mathbf{x}_i, \mathbf{x}_j + \Delta \mathbf{x}_j)$  em cada termo  $F_{i,j}(\mathbf{x})$  do somatório na Equação 4.16, resulta em

$$F_{i,j}(\mathbf{\breve{x}} + \Delta \mathbf{x}) = \mathbf{e}_{i,j}(\mathbf{\breve{x}} + \Delta \mathbf{x})^t \Omega_{i,j} \mathbf{e}_{i,j}(\mathbf{\breve{x}} + \Delta \mathbf{x})$$

$$= (\mathbf{e}_{i,j} + \mathbf{J}_{i,j} \Delta \mathbf{x})^t \Omega_{i,j} (\mathbf{e}_{i,j} + \mathbf{J}_{i,j} \Delta \mathbf{x})$$

$$= \mathbf{e}_{i,j}^t \Omega_{i,j} \mathbf{e}_{i,j} + 2\mathbf{e}_{i,j}^t \Omega_{i,j} \mathbf{J}_{i,j} \Delta \mathbf{x} + \Delta \mathbf{x}^t \mathbf{J}_{i,j}^t \Omega_{i,j} \mathbf{J}_{i,j} \Delta \mathbf{x},$$

onde

$$c_{i,j} = \mathbf{e}_{i,j}^t \Omega_{i,j} \mathbf{e}_{i,j},$$
  
$$\mathbf{b}_{i,j} = \mathbf{e}_{i,j}^t \Omega_{i,j} \mathbf{J}_{i,j},$$
  
$$\mathbf{H}_{i,j} = \mathbf{J}_{i,j}^t \Omega_{i,j} \mathbf{J}_{i,j}.$$

A aproximação local de cada termo  $F_{i,j}(\mathbf{x})$  pode ser então substituída no somatório da Equação 4.15, como dado por

$$F(\check{\mathbf{x}} + \Delta \mathbf{x}) = \sum_{\langle i,j \rangle \in \mathcal{C}} F_{ij}(\check{\mathbf{x}} + \Delta \mathbf{x})$$

$$= \sum_{\langle i,j \rangle \in \mathcal{C}} c_{ij} + 2\mathbf{b}_{ij}\Delta \mathbf{x} + \Delta \mathbf{x}^t \mathbf{H}_{ij}\Delta \mathbf{x}$$

$$= \mathbf{c} + 2\mathbf{b}^t \Delta \mathbf{x} + \Delta \mathbf{x}^t \mathbf{H} \Delta \mathbf{x},$$

na qual

$$\mathbf{c} = \sum_{\langle i,j \rangle \in \mathcal{C}} c_{i,j},\tag{4.19}$$

$$\mathbf{b} = \sum_{\langle i,j \rangle \in \mathcal{C}} \mathbf{b}_{i,j},\tag{4.20}$$

$$\mathbf{H} = \sum_{\langle i,j\rangle \in \mathcal{C}} \mathbf{H}_{i,j}. \tag{4.21}$$

Derivando-se  $F(\mathbf{x} + \Delta \mathbf{x})$  em relação a  $\Delta \mathbf{x}$ , chega-se a

$$\frac{\partial F(\breve{\mathbf{x}} + \Delta \mathbf{x})}{\partial \Delta \mathbf{x}} = 2\mathbf{b} + 2\mathbf{H}\Delta \mathbf{x},$$

cujo valor mínimo pode ser obtido pela resolução do sistema linearizado

$$\mathbf{H}\Delta\mathbf{x} = -\mathbf{b}.\tag{4.22}$$

O processo de obtenção de  $\mathbf{x}^*$  é iterativo, onde cada iteração computa o incremento  $\Delta \mathbf{x}$ , acumula este valor em  $\mathbf{x}$ , e lineariza o erro em torno da nova solução  $\mathbf{x} + \Delta \mathbf{x}$  com expansão em série de Taylor. A solução  $\mathbf{x}^*$  é dada por  $\mathbf{x}$  quando um critério de parada é atingido.

#### 4.2.3 Estrutura Esparsa das Matrizes do Sistema Linearizado

Uma vez que o erro  $\mathbf{e}_{i,j}(\mathbf{x})$  depende apenas da restrição entre dois vértices  $\mathbf{x}_i$  e  $\mathbf{x}_j$ , o Jacobiano  $\mathbf{J}_{i,j}$  possui uma estrutura matricial peculiar. Mais precisamente, a derivada parcial em relação a  $\mathbf{x}$  será igual ao vetor  $\mathbf{0}$  para todos os vértices do grafo, exceto para os vértices  $\mathbf{x}_i$  e  $\mathbf{x}_j$ , ou seja,

$$\mathbf{J}_{i,j} = \left( \begin{array}{cccc} \mathbf{0}...\mathbf{0} & \mathbf{A}_{i,j} & \mathbf{0}...\mathbf{0} & \mathbf{B}_{i,j} & \mathbf{0}...\mathbf{0} \end{array} \right), \tag{4.23}$$

onde

$$\mathbf{A}_{i,j} = \frac{\partial \mathbf{e}_{i,j}(\mathbf{x})}{\partial \mathbf{x}_i} \bigg|_{\mathbf{x} = \check{\mathbf{x}}} \quad \mathbf{e} \quad \mathbf{B}_{i,j} = \frac{\partial \mathbf{e}_{i,j}(\mathbf{x})}{\partial \mathbf{x}_j} \bigg|_{\mathbf{x} = \check{\mathbf{x}}}.$$

Consequentemente, todas as matrizes envolvidas na resolução do sistema linearizado (Equação 4.22) são esparsas (uma grande quantidade dos elementos são iguais a 0), já que elas são originárias de somas e produtos entre o vetor  $\mathbf{e}_{i,j}(\mathbf{x})$  e  $\mathbf{J}_{i,j}$ . A matriz  $\mathbf{H}_{i,j}$ 

constitui as parcelas da matriz de informação do sistema completo H, assumindo a forma

$$\mathbf{H}_{i,j} = \begin{pmatrix} \ddots & & & \\ & \mathbf{A}_{i,j}^t \Omega_{i,j} \mathbf{A}_{i,j} & \dots & \mathbf{A}_{i,j}^t \Omega_{i,j} \mathbf{B}_{i,j} \\ & \vdots & \ddots & \vdots \\ & \mathbf{B}_{i,j}^t \Omega_{i,j} \mathbf{A}_{i,j} & \dots & \mathbf{B}_{i,j}^t \Omega_{i,j} \mathbf{B}_{i,j} \end{pmatrix}$$
(4.24)

assim como o vetor  $\mathbf{b}_{i,j}$ ,

$$\mathbf{b}_{i,j} = \begin{pmatrix} \vdots \\ \mathbf{A}_{i,j}^t \mathbf{\Omega}_{i,j} \mathbf{e}_{i,j} \\ \vdots \\ \mathbf{B}_{i,j}^t \mathbf{\Omega}_{i,j} \mathbf{e}_{i,j} \\ \vdots \end{pmatrix}, \tag{4.25}$$

mostrados com os elementos nulos omitidos.

A relação de vizinhança entre os vértices no grafo indica correlação estatística entre os vetores erro correspondentes [Grisetti et al. 2010], ou seja, vértices adjacentes a qualquer outro com várias observações, tendem a ter uma melhor estimativa final após a minimização. Esta relação de adjacência é exatamente o que se reflete na estrutura esparsa da matriz de informação **H** do sistema completo, ou seja, quanto mais denso é o grafo, menos esparsas são as matrizes do sistema, sendo esta característica explorada na resolução do sistema linearizado da Equação 4.22 para fornecer soluções eficientes [Davis 2006].

#### 4.2.4 Parametrização Mínima da Pose

O vetor de pose para um vértice qualquer  $\mathbf{x}_i$  é assumido parametrizado com uma parte denotando a sua posição e outra parte denotando a sua orientação, ambas em relação à origem. Um vetor de translação  $\mathbf{t}_i \in \mathbb{R}^3$  pode ser usado para a parte da posição, enquanto as entradas de uma matriz de rotação podem ser alinhadas em um vetor de 9 dimensões, compondo assim cada  $\mathbf{x}_i$  com dimensão igual a 12. Tal parametrização possui consequências em relação à minimização, uma vez que as soluções computadas para cada  $\mathbf{x}_i^*$  pode fazer com que a parte relativa à orientação deixe de representar de fato uma orientação. Em outras palavras, o vetor de 9 dimensões computado para a orientação pode deixar de formar uma matriz de rotação, já que para  $\mathbf{R}_i$  ser uma matriz de rotação, esta precisa ser ortonormal, o que não é explicitamente garantido na resolução do sistema linearizado.

Isto acontece porque a parte vetorial correspondente à orientação não é um vetor Euclidiano: variações pequenas nos valores dos seus componentes podem corresponder a variações bruscas na orientação resultante e vice-versa. Ao parametrizar a parte de orientação com ângulos de Euler, este problema é evitado, embora isto resulte em soluções sujeitas às singularidades desta representação [Grisetti et al. 2010].

Como solução, a minimização é conduzida considerando um espaço formado por variedades diferenciáveis (*smooth manifolds*) [Lima 2011] ao invés do espaço Euclidiano.

Nestas superfícies, garante-se que uma variação pequena no vetor correspondente à orientação resulta em uma orientação válida [Grisetti et al. 2010]. O vetor  $\mathbf{x}_i$  é então parametrizado de forma mínima, com a sua parte relativa à orientação formada por três números correspondentes à parte vetorial de um quatérnion unitário. Em última análise, a condução da minimização no espaço de variedades diferenciáveis possui o mesmo formalismo discutido nesta seção, sendo necessário apenas uma função que converte da representação sobreparametrizada (por ex. com matrizes de rotação) para quatérnions unitários e viceversa. Os Jacobianos são computados em  $\Delta \mathbf{\tilde{x}}$  (com  $\mathbf{\tilde{x}}$  sendo uma representação mínima de  $\mathbf{x}$ ), obtidos pelo uso de um operador que converte as representações [Grisetti et al. 2010].

# Capítulo 5

# Implementação do Algoritmo de Registro de Nuvens de Pontos RGB-D

Os detalhes envolvidos na implementação do método de registro de nuvens de pontos RGB-D proposto nesta tese são apresentados neste capítulo, objetivando concretizar o formalismo discutido no Capítulo 4. Toda a implementação foi concebida em um ambiente C++/Linux, utilizando as bibliotecas OpenCV (www.opencv.org), Point Cloud Library (www.pointclouds.org), G2O [Kümmerle et al. 2011] e ARUCO [Garrido-Jurado et al. 2014].

O registro é implementado em dois processos: registro incremental, obtido por odometria visual e registro global, implementado pela minimização do erro acumulado da odometria visual por minimização em grafo de poses. A estimação de odometria visual, detecção de fechamento de laço e a construção do grafo são realizados por um *front-end*, enquanto a linearização e resolução do sistema (Equação 4.22) por mínimos quadrados é realizada por um *back-end* [Thrun e Leonard 2008]. Na prática, isto implica em uma modularização da implementação, na qual cada parte constituinte possui uma entrada e saída associadas, como exibe a Figura 5.1.

As contribuições principais desta tese se concentram em implementações relacionadas ao *front-end*. A Seção 5.1 apresenta o módulo de odometria visual, responsável por computar o registro incremental. A detecção de um marcador artificial é discutida na Seção 5.2.1, enquanto a Seção 5.2.2 exibe uma solução para que o método proposto reduza a quantidade de dados a serem processados. Na Seção 5.2.3, é mostrado o algoritmo relacionado à construção do grafo. Observe que, a partir da solução adotada, o processo de minimização do erro acumulado acontece tanto no *front-end* quanto no *back-end*, já que isto envolve detectar fechamento de laços (detectar o marcador), construir o grafo de poses a partir das nuvens de pontos, computar os erros e Jacobianos de cada aresta do grafo (ou seja, linearizar o sistema) e obter uma solução por mínimos quadrados para a trajetória otimizada. O trabalho do *back-end* (Seção 5.2.4) é executado por rotinas da biblioteca G2O.

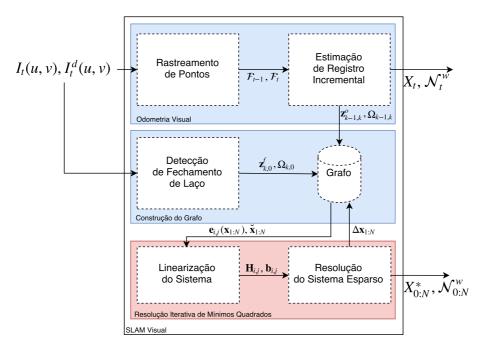


Figura 5.1: Diagrama exibindo os módulos envolvidos no algoritmo de registro de nuvens de pontos RGB-D proposto e o fluxo de dados entre eles. O *front-end* do sistema é mostrado em azul e o *back-end* é mostrado em vermelho.

#### 5.1 Odometria Visual

O módulo de odometria visual é responsável por detectar e rastrear features visuais de imagem para imagem e em seguida, estimar a transformação de registro entre  $I_{t-1}(u,v)$  e  $I_t(u,v)$  de forma robusta a falsas correspondências. Estas features são gerenciadas pelo rastreador de pontos de acordo com uma estratégia específica dada por um algoritmo. A partir do fluxo óptico KLT, duas estratégias são implementadas, discutidas nas Seções 5.1.2 e 5.1.3.

#### 5.1.1 Detecção de Features Visuais

Ao detectar features visuais em uma imagem I(u,v), a aplicação direta das Equações 2.5 e 2.6 resulta em pontos de Shi-Tomasi classificadas de acordo com o valor do limiar  $\tau$ . Para remover a dependência deste parâmetro, um processo de detecção é implementado onde cada coordenada de imagem dada por p,q é classificada como uma feature de Shi-Tomasi de acordo com a comparação do menor autovalor  $\lambda$  da matriz de covariância  $\mathbf{C}$  com o maior autovalor  $\lambda_M$  computado dentre todos pixels. Assim, caso  $\lambda > f\lambda_M$ , onde f é um fator em forma de porcentagem, uma feature é detectada na posição p,q. Para evitar que features sejam detectadas em posições próximas umas das outras, pontos separados por uma distância menor do que  $D_{\min}$  são também removidos. Os  $N_f$  pontos com maior valor de  $\lambda$  são selecionados como features e adicionados a um conjunto  $\mathcal{S}_t$ . Para facilitar o processamento, cada valor computado para  $\lambda$  é adicionado a um mapa de qualidade em forma de imagem M(u,v). O processo encontra-se em forma de pseudo-código no

Algoritmo 1.

```
Algoritmo 1 Detecção de Features
```

```
Entrada: I(u,v)
Saída: S_t
 1: compute os gradientes da imagem I_u(u,v) e I_v(u,v)
 2: inicialize o mapa de qualidade das features M(u, v) como a imagem nula, de mesma
    dimensão que I(u, v)
 3: para todo pixel p, q de I(u, v) faça
       compute a matriz C na vizinhança 3 \times 3 de p,q usando os gradientes I_u(u,v) e
       I_{\nu}(u,v)
       aplique a Equação 2.5 e 2.6 para computar \lambda_1 e \lambda_2 da matriz C
 5:
       guarde \lambda = \min(\lambda_1, \lambda_2) no mapa de qualidade M(u, v)
 6:
       guarde o maior entre todos os autovalores em \lambda_M
 7:
 8: fim para
 9: para todo valor do mapa M(u,v) faça
       se M(u,v) > f\lambda_M então
10:
         adicione M(u, v) a um conjunto S'_t de features visuais
11:
       fim se
12:
13: fim para
14: ordene o conjunto S'_t em ordem decrescente de \lambda
15: descarte os pontos de S'_t que estão separados por uma distância menor que D_{\min}
16: adicione os N_f primeiros pontos de S'_t em S_t
```

### **5.1.2** Algoritmo KLT de Rastreamento de Features

Após serem detectadas, as features de Shi-Tomasi do conjunto  $S_t$  são rastreadas de imagem para imagem com o fluxo óptico piramidal esparso discutido na Seção 4.1.1. Estas posições na imagem constituem os pontos com as melhores características para o rastreamento com o fluxo óptico esparso, fazendo com que o registro incremental obtenha resultados precisos de forma eficiente. Em alternativa a detectar pontos característicos a cada nova imagem  $I_t(u,v)$ , as features são rastreadas até que o seu número se torne menor do que um limiar especificado previamente, quando só então é disparada uma nova detecção.

Especificamente, o conjunto  $S_t$  de features detectadas é atribuído ao conjunto de features mantidas pelo rastreador ao longo de cada instante de tempo,  $\mathcal{P}_t$ . Para o passo de tempo seguinte, as features de  $\mathcal{P}_t$  são diretamente atribuídas a um conjunto de pontos  $\mathcal{P}_{t-1}$ , sendo este conjunto usado como entrada para o cálculo do fluxo óptico piramidal esparso segundo a Equação 4.2. No cálculo do fluxo óptico, vários pontos de  $\mathcal{P}_{t-1}$  podem não ter a sua correspondência estabelecida, o que pode ocorrer por uma diferença muito grande entre os valores da janela de pixels em torno do ponto nas imagens  $I_{t-1}(u,v)$  e  $I_t(u,v)$  (um ponto é removido por causa de erro maior do que um limiar) ou porque o fluxo óptico não pôde ser determinado para ele (situação de oclusão). Os pontos de  $\mathcal{P}_{t-1}$ 

com vetor deslocamento determinado são rastreados nas suas correspondências na imagem atual, dadas pelo conjunto  $\mathcal{P}_t$ . Logo após o fluxo óptico ser computado, o número de pontos do rastreador  $|\mathcal{P}_t|$  é comparado com um limiar para o mínimo de pontos  $N_{\min}$ , com novas features detectadas e adicionadas ao rastreador caso  $|\mathcal{P}_t| \leq N_{\min}$ . Assume-se que a computação do fluxo óptico pela resolução da Equação 4.2 é realizada em uma rotina fluxo\_optico, que calcula as derivadas de intensidade da imagem, constrói as pirâmides em multirresolução e computam a minimização em um processo iterativo. Um par de nuvens de pontos RGB-D esparsas  $\mathcal{F}_{t-1}$  e  $\mathcal{F}_t$  são obtidas de correspondências de pontos entre  $\mathcal{P}_{t-1}$  e  $\mathcal{P}_t$ , possuindo estas um mesmo tamanho, como discutido na Seção 4.1.2. Este algoritmo, denominado rastreamento KLT, produz como saída as nuvens de pontos esparsas usadas para computar a transformação de registro. O pseudo-código do rastreamento KLT é mostrado no Algoritmo 2.

#### Algoritmo 2 Rastreamento KLT

```
Entrada: I_{t-1}(u,v), I_t(u,v)
Saída: \mathcal{F}_{t-1}, \mathcal{F}_t, \mathcal{P}_{t-1}, \mathcal{P}_t
  1: se |\mathcal{P}_{t-1}| = 0 então
         detecte o conjunto S_t de features iniciais com o Algoritmo 1
  3:
         \mathcal{P}_t \leftarrow \mathcal{S}_t
  4: senão
         rastreie os pontos de \mathcal{P}_{t-1} nos pontos atuais \mathcal{P}_t pela resolução da Equação 4.2 em
  5:
         uma rotina fluxo_optico(I_{t-1}(u,v), I_t(u,v), \mathcal{P}_{t-1}, \mathcal{P}_t)
         adicione as correspondências de pontos obtidos nas nuvens de pontos RGB-D es-
  6:
         parsas \mathcal{F}_{t-1} e \mathcal{F}_t
  7:
         se |\mathcal{P}_t| \leq N_{\min} então
             detecte o conjunto S_t de novas features com o Algoritmo 1
  8:
  9:
             adicione S_t ao conjunto de pontos rastreados \mathcal{P}_t
         fim se
 10:
11: fim se
12: \mathcal{P}_{t-1} \leftarrow \mathcal{P}_t
```

## **5.1.3** Algoritmo KLTTW de Rastreamento de Features

Um efeito indesejável do rastreamento KLT é a possível concentração de pontos rastreados em uma região comum. Isto pode acontecer quando, ao realizar o cálculo do fluxo óptico em uma sequência de imagens, o número de features do rastreador  $|\mathcal{P}_t|$  se mantém constante e acima de  $N_{\min}$ , ao mesmo tempo em que todos os  $|\mathcal{P}_t|$  pontos rastreados se encontram aglomerados em uma mesma região da imagem. Portanto, a detecção de novas features, que poderia ser executada para resolver esta situação, é inibida, limitando o rastreamento a uma porção de pontos com propriedades comuns. Para uma melhor qualidade na estimativa da transformação de registro, é importante que as features encontrem-se de forma espalhada por todas as posições da imagem, evitando assim configurações singulares ou ambíguas na determinação da transformação entre as nuvens de pontos RGB-D.

55

Com o objetivo de realizar o rastreamento de features por fluxo óptico com os pontos característicos visuais dispostos de uma maneira uniforme pelas imagens, é proposto um esquema de rastreamento denominado de rastreamento KLTTW (KLT with *Tracking Windows*). Nesta estratégia de rastreamento, novas features são detectadas em cada nova imagem, com cada ponto adicionado ao rastreador caso 1. novas features sejam necessárias de acordo com um limiar e 2. este ponto não faça parte da região da imagem coberta pelos pontos do rastreador.

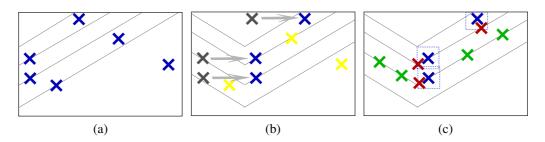


Figura 5.2: Rastreamento KLTTW (KLT com janelas de rastreamento): (a) Na imagem  $I_0(u,v)$ , o conjunto de features iniciais  $S_0$  (em azul) é extraído e usado para alimentar o conjunto  $\mathcal{P}_0$  de pontos do rastreador. (b) Na imagem  $I_t(u,v)$  com t>0, os pontos de  $\mathcal{P}_{t-1}$  (em cinza) são rastreados em  $\mathcal{P}_t$  (azul). Features que não puderam ser rastreadas pelo fluxo óptico piramidal esparso são mostradas em amarelo. (c) Um conjunto de features candidatas  $S_t$  é computado, com os pontos candidatos que foram adicionados ao conjunto  $\mathcal{P}_t$  mostrados em verde e os pontos rejeitados (dentro da região retangular dos pontos rastreados) mostrados em vermelho.

O esquema é similar ao rastreamento KLT (Algoritmo 2), sendo inicializado da mesma forma e também baseado no fluxo óptico piramidal esparso para obtenção de correspondências entre os conjuntos de pontos. Em específico ao algoritmo KLTTW, à cada ponto  $\mathbf{p}_t^j$  do conjunto de features do rastreador  $\mathcal{P}_t$ , é atribuída uma janela retangular  $W_j$  de tamanho fixo  $S_l \times S_a$ , denotando a região coberta pelo ponto. Um conjunto de features candidatas  $\mathcal{S}_t$  é detectado em toda nova imagem  $I_t(u,v)$ , com cada ponto  $\mathbf{q}^i$  deste conjunto adicionado ao rastreador se o número de pontos rastreados  $|\mathcal{P}_t|$  for menor do que um limiar especificado  $N_{\max}$  e também se  $\mathbf{q}^i$  não se encontrar no interior de nenhuma janela  $W_j$  correspondente a cada  $\mathbf{p}_t^j$ . Note que cada ponto candidato  $\mathbf{q}^i$  adicionado ao rastreador somente pode ser utilizado para computar a transformação de registro na imagem seguinte, uma vez que ele não possui uma correspondência associada no conjunto  $\mathcal{P}_{t-1}$ . A Figura 5.2 exibe a intuição para o rastreamento KLTTW, com os passos do processo mostrados no Algoritmo 3.

Os parâmetros correspondentes ao tamanho da janela  $S_l \times S_a$  governa o desempenho do algoritmo. À medida em que este tamanho é pequeno, uma quantidade maior de features detectadas é adicionada ao rastreador por imagem. Reciprocamente, menos pontos característicos são adicionados quando o tamanho da janela  $S_l \times S_a$  é grande, uma vez que a região de cobertura de cada feature se torna maior. Algoritmos mais rebuscados envolvendo por exemplo janelas de tamanho adaptativo podem ser desenvolvidos, embora não sejam considerados neste trabalho. Apesar de se tratar de uma heurística simples, o

#### Algoritmo 3 Rastreamento KLTTW

```
Entrada: I_{t-1}(u,v), I_t(u,v)
Saída: \mathcal{F}_{t-1}, \mathcal{F}_t, \mathcal{P}_{t-1}, \mathcal{P}_t
  1: se |\mathcal{P}_{t-1}| = 0 então
         detecte o conjunto S_t de features iniciais com o Algoritmo 1
 3:
         \mathcal{P}_t \leftarrow \mathcal{S}_t
 4: senão
         rastreie os pontos de \mathcal{P}_{t-1} nos pontos atuais \mathcal{P}_t pela resolução da Equação 4.2 em
         uma rotina fluxo_optico(I_{t-1}(u,v), I_t(u,v), \mathcal{P}_{t-1}, \mathcal{P}_t)
         adicione as correspondências de pontos obtidos nas nuvens de pontos RGB-D es-
 6:
         parsas \mathcal{F}_{t-1} e \mathcal{F}_t
         detecte o conjunto S_t de features candidatas com o Algoritmo 1
 7:
         enquanto |\mathcal{P}_t| < N_{\text{max}} faça
 8:
             tome um ponto candidato \mathbf{q}^i de \mathcal{S}_t
 9:
             para todo \mathbf{p}_t^j de \mathcal{P}_t faça
10:
                se \mathbf{q}^i estiver dentro de alguma janela de rastreamento W_i de \mathbf{p}_t^j então
11:
                    rejeite \mathbf{q}^i e saia do laço
12:
                fim se
13:
             fim para
14:
             se q<sup>i</sup> não foi rejeitado então
15:
                adicione \mathbf{q}^i ao conjunto de pontos rastreados \mathcal{P}_t
16:
                remova \mathbf{q}^i de \mathcal{S}_t
17:
18:
             fim se
19:
         fim enquanto
20: fim se
21: \mathcal{P}_{t-1} \leftarrow \mathcal{P}_t
```

rastreamento KLTTW é responsável por computar um registro visual de nuvens de pontos de forma precisa e eficiente.

A união de todas as janelas de rastreamento formam regiões de exclusão, ou seja, porções na imagem onde novas features são rejeitadas. Este comportamento vai de encontro com abordagens presentes na literatura que delimitam regiões de interesse, nas quais devem ser buscadas as correspondências de cada feature [Davison 2003]. De qualquer forma, os objetivos das abordagens são diferentes, visto que o algoritmo proposto visa manter um número constante de features espalhadas pela imagem no rastreador e não obter correspondências com maior precisão.

Na Figura 5.3 é mostrado o comportamento do rastreamento KLTTW em relação ao rastreamento KLT, onde é possível se notar uma maior disponibilidade espacial de pontos rastreados de imagem para imagem.

#### 5.1.4 Estimação de Pose Robusta a Falsas Correspondências Visuais

Nuvens de pontos esparsas  $\mathcal{F}_{t-1}$  e  $\mathcal{F}_t$  são obtidas a partir dos pontos rastreados  $\mathcal{P}_{t-1}$ e  $\mathcal{P}_t$  para servirem como dados de entrada no cálculo da transformação de registro. É assumido que pontos do rastreador sem medida de profundidade associada são removidos logo após o rastreamento por fluxo óptico, uma vez que este dado é estritamente necessário nesta etapa. A partir de  $\mathcal{F}_{t-1}$  e  $\mathcal{F}_t$ , a transformação de registro  $\mathbf{T}_{t,t-1}$  que resulta na mudança de pose de câmera entre as imagens  $I_{t-1}(u,v)$  e  $I_{t-1}(u,v)$  pode ser computada como discutido na Seção 4.1.2. Entretanto, por se tratar de um problema de minimização por mínimos quadrados, a aplicação direta da Equação 4.7 assume que o erro em relação às coordenadas verdadeiras de cada ponto 3D segue uma distribuição Gaussiana, o que geralmente não acontece com o uso de correspondências obtidas a partir de imagens. Esta propriedade é desrespeitada devido à existência de falsos casamentos entre pontos das nuvens esparsas, que são considerados outliers (que não fazem parte) em relação à distribuição Gaussiana. Uma única correspondência falsa é capaz de inutilizar completamente uma solução obtida por mínimos quadrados. É necessário portanto um mecanismo que detecte correspondências inliers (pertencentes) à esta distribuição e que posteriormente utilize estes dados no processo de obtenção da transformação. Para isto, utiliza-se o algoritmo RANSAC [Fischler e Bolles 1981, Hartley e Zisserman 2004] de estimação de modelos robusta a falsas correspondências visuais, como discutido na Seção 2.5.3.

Para computar a transformação de registro de forma robusta, em cada iteração do RANSAC é selecionada aleatoriamente uma fração mínima dos dados disponíveis, de forma que a partir deles, uma hipótese para o modelo estimado possa ser computada. No caso do modelo consistir de uma transformação de corpo rígido (rotação + translação) tridimensional, três casamentos entre pontos das duas nuvens são necessárias. Cada hipótese para a transformação  $\mathbf{T}_h$  é avaliada no restante dos dados disponíveis, com cada par  $\mathbf{P}_{t-1}^k$ ,  $\mathbf{P}_t^k$  concordando ou não com a solução hipótese, de acordo com uma medida de erro dada pela distância Euclidiana entre o ponto original e o ponto transformação pela transformação hipótese. A correspondência que concordar (possuir erro inferior a um limiar  $\tau_R$ ) com o modelo é adicionada ao conjunto suporte da solução  $\mathcal{M}_h$ . Após um certo número de iterações, a hipótese com maior tamanho do conjunto  $\mathcal{M}_h$  é eleita a vencedora, com

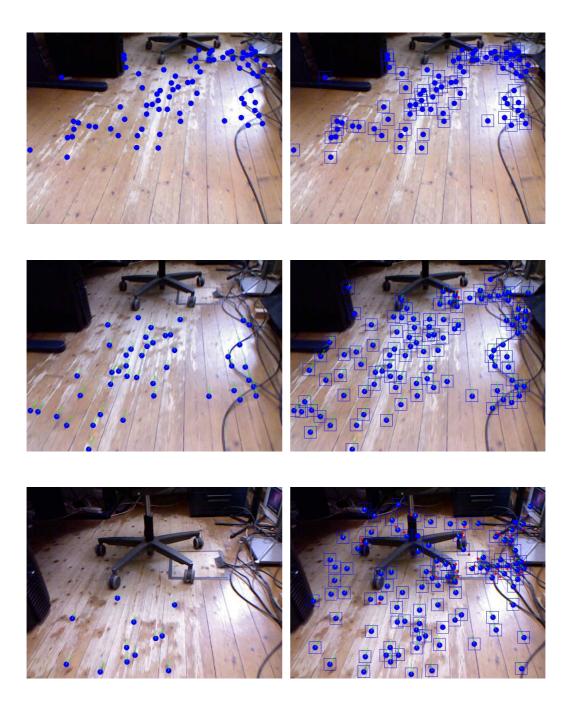


Figura 5.3: Comparação entre o rastreamento KLT (coluna esquerda) e o rastreamento KLTTW (coluna direita), ao longo de três imagens de uma mesma sequência. Com exceção da primeira imagem (primeira linha) que corresponde à imagem inicial  $I_0(u,v)$  da sequência, pode-se notar que as features rastreadas pelo algoritmo KLTTW se encontram dispostas de maneira uniforme ao longo das coordenadas de imagem em relação ao algoritmo KLT. A figura também exibe as janelas retangulares de cada ponto rastreado e features extraídas que foram rejeitadas (em vermelho).

os casamentos nelas contidos utilizados para computar uma transformação de registro por mínimos quadrados.

Uma heurística simples pode ser utilizada para computar o número de iterações para o encerramento do RANSAC [Hartley e Zisserman 2004]. Este cálculo leva em consideração o número total de iterações  $N_R$  necessárias para garantir com probabilidade igual a  $\alpha$  que as três correspondências entre pontos sejam constituídas de casamentos *inliers*. Supondo que uma fração  $\varepsilon$  de pares *outliers* está presente entre todas as correspondências,  $N_R$  pode ser obtido conforme

$$N_R = log(1 - \alpha)/log(1 - (1 - \epsilon)^3).$$
 (5.1)

Como exemplo, supondo 99% de probabilidade de seleção aleatória de casamentos *inliers* e uma fração de pares *outliers* igual a 92,5% (valores para  $\alpha$  e  $\epsilon$  respectivamente), o número de iterações  $N_R$  resultante é superior a 10000 iterações. Apesar de ser um número elevado,  $N_R$  pode ser atualizado durante as iterações de acordo com o novo valor de  $\epsilon$  obtido pela razão entre o tamanho do conjunto suporte correspondente à melhor solução até o momento em relação ao número total de correspondências. O passo a passo em pseudo-código para obter a transformação de registro com o RANSAC é mostrado no Algoritmo 4.

# 5.1.5 Registro Incremental a Partir de Registro por Pares de Nuvens RGB-D

Após ser computada a transformação  $\mathbf{T}_{t,t-1}$  de mudança de pose de câmera entre as imagens  $I_{t-1}(u,v)$  e  $I_t(u,v)$ , a pose de câmera atual  $\mathbf{X}_t$  é obtida conforme a Equação 4.1. Esta concatenação é realizada também para a transformação de registro  $\mathbf{T}_{t-1,t} = \mathbf{T}_{t,t-1}^{-1}$ , fazendo com que a transformação acumulada registre a nuvem de pontos densa atual  $\mathcal{N}_t$  no sistema de coordenadas global, resultando na nuvem  $\mathcal{N}_t^w$  composta pela sobreposição de todas as nuvens  $\mathcal{N}_{0:t}$  no mesmo sistema de coordenadas.

Apesar de poder ser considerado um mapa de pontos com propriedades que possibilitam o seu uso posterior em algumas aplicações, este processo de registro incremental está sujeito a acúmulos de erro inerente ao processo. Este fato é exibido no exemplo da Figura 5.4, que mostra a evolução da nuvem total  $\mathcal{N}_t^w$  à medida em que uma sala é mapeada. Nota-se claramente que o registro não possui consistência global quando a última nuvem de pontos não se alinha com a região correspondente à primeira nuvem processada.

## 5.2 Minimização do Erro Acumulado

A detecção de fechamento de laços constitui um módulo importante do sistema proposto, uma vez que sem ele, não é possível obter um registro de nuvens de pontos RGB-D que minimize o erro acumulado de forma global. Isto se deve ao fato de o fluxo óptico não ser capaz de obter correspondências por linha de base larga, o que poderia ser realizado entre a imagem atual e imagens anteriores, com o objetivo de detectar locais mapeados previamente. Desta forma, o sistema implementa como detector de fechamento de laço

#### Algoritmo 4 Estimação de Transformação de Registro com RANSAC

```
Entrada: \mathcal{F}_{t-1}, \mathcal{F}_t, \alpha, \epsilon, \tau_R
Saída: T_{t,t-1}, \mathcal{M}
    calcule o número de iterações necessárias N_R de acordo com a Equação 5.1
    inicialize o melhor conjunto de correspondências corretas \mathcal{M} como o conjunto vazio
    inicialize a melhor transformação T como a matriz identidade I_{4\times4}
   i \leftarrow 0
   enquanto i < N_R faça
        inicialize o conjunto de correspondências inliers \mathcal{M}_h como o conjunto vazio
        selecione aleatoriamente 3 pontos \mathbf{P}_{t-1}^i, \mathbf{P}_{t-1}^j, \mathbf{P}_{t-1}^k de \mathcal{F}_{t-1}
        selecione as correspondências dos pontos \mathbf{P}_{t}^{i}, \mathbf{P}_{t}^{j}, \mathbf{P}_{t}^{k} em \mathcal{F}_{t}
        obtenha uma solução hipótese T_h para a transformação de registro (Equação 4.7),
        usando \{\mathbf{P}_{t-1}^i, \mathbf{P}_{t-1}^j, \mathbf{P}_{t-1}^k\} e \{\mathbf{P}_t^i, \mathbf{P}_t^j, \mathbf{P}_t^k\} como entrada
        para todo par \mathbf{P}_{t-1}^m, \mathbf{P}_t^m de \mathcal{F}_{t-1} e \mathcal{F}_t faça
            \mathbf{P}' \leftarrow \mathbf{R}_h \mathbf{P}_{t-1}^m + \mathbf{t}_h, onde \mathbf{R}_h e \mathbf{t}_h são tirados de \mathbf{T}_h
            d^m \leftarrow d(\mathbf{P}', \mathbf{P}_t^m), onde d(.) é a distância Euclidiana
            se d^m < \tau_R então
                adicione a correspondência \mathbf{P}_{t-1}^m, \mathbf{P}_t^m ao conjunto \mathcal{M}_h
            fim se
        fim para
        se |\mathcal{M}_h| > |\mathcal{M}| então
            \mathcal{M} \leftarrow \mathcal{M}_h
            \mathbf{T} \leftarrow \mathbf{T}_h
            se |\mathcal{M}_h|/|\mathcal{F}_t| > (1-\epsilon)| então
               \varepsilon \leftarrow 1 - |\mathcal{M}_h|/|\mathcal{F}_t|
                atualize N_R utilizando a Equação 4
            fim se
       fim se
        i \leftarrow i + 1
```

#### fim enquanto

Obtenha a solução  $\mathbf{T}_{t,t-1}$  pela resolução da Equação 4.7 usando o conjunto  $\mathcal{M}$  de correspondências classificadas como verdadeiras

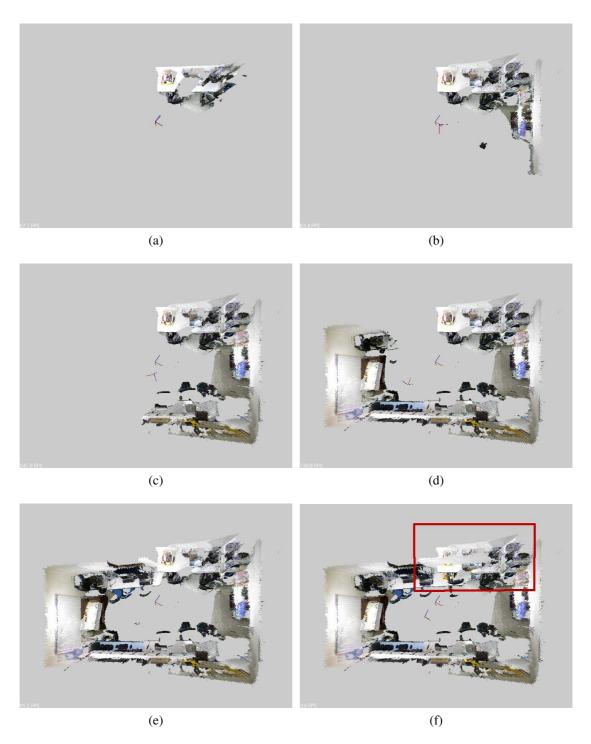


Figura 5.4: Erro inerente ao registro incremental de nuvens de pontos RGB-D. Em (a-e), uma sala está sendo mapeada, com a nuvem  $\mathcal{N}_t$  relativa a cada instante de tempo acumulada em uma nuvem total  $\mathcal{N}_t^w$  no referencial global. Em (f), é evidenciado o acúmulo de erro, devido ao qual a nuvem atual não se alinha após o registro com a nuvem total.

a busca em cada imagem de um marcador artificial de realidade aumentada da biblioteca ARUCO [Garrido-Jurado et al. 2014]. A partir deste mecanismo, são estabelecidas relações entre vértices não-sequenciais do grafo, dadas pela transformação relativa entre a pose de câmera na qual um marcador é detectado e a pose de vértices do grafo.

# **5.2.1** Fechamento de Laço com Detecção de Marcador Artificial de Realidade Aumentada

Define-se como marcador artificial de realidade aumentada um objeto planar, com padrão e dimensões conhecidas previamente. Estes objetos se tornaram popular na comunidade de realidade aumentada pois, a partir deles, é possível sobrepor gráficos renderizados em imagens de vídeo [Garrido-Jurado et al. 2014]. Na biblioteca ARUCO, um marcador é formado por uma borda preta e interior dividido em blocos iguais, os quais podem possuir cor preta ou cor branca. Cada combinação possível de valores para os blocos forma uma cadeia de dígitos binários, com todas as possibilidades de cadeias constituindo um dicionário. Ao realizar a detecção de marcadores na cena, os padrões em potencial na imagem são checados se pertencem ou não a este dicionário, reduzindo assim a ocorrência de falsas detecções [Garrido-Jurado et al. 2014].

Para que um marcador seja detectado, cada imagem é processada por uma sequência de operações. Após limiarização no nível de intensidade de cada pixel, contornos são extraídos, sendo rejeitados aqueles que não se aproximarem de um polígono de quatro lados. A projeção perspectiva é removida de regiões candidatas pelo cálculo da homografia [Hartley e Zisserman 2004], transformação que relaciona o plano do objeto com o plano da imagem. Cada região candidata é checada em relação ao dicionário de padrões, sendo descartada aquela cujo padrão não se encaixa com os armazenados. Por último, a pose da câmera em relação a um referencial fixo no marcador é computada por minimização iterativa do erro da homografia [Garrido-Jurado et al. 2014, Hartley e Zisserman 2004] com o algoritmo de Levenberg-Marquardt [Marquardt 1963], fornecendo a posição e orientação com a qual possíveis gráficos podem ser renderizados sobre a cena. A Figura 5.5 mostra o padrão usado como marcador, o marcador detectado em uma imagem e a visualização tridimensional da nuvem de pontos RGB-D referenciada no sistema de coordenadas fixo no marcador.

Em termos práticos, o uso de marcadores artificiais permite que uma pose de câmera em relação ao marcador seja computada de forma precisa e eficiente. De acordo com os autores do sistema ARUCO [Garrido-Jurado et al. 2014], a pose para um marcador é computada em cerca de 10 milissegundos e sem ocorrências de falsos positivos, o que a torna aplicável em um sistema de detecção de fechamento de laço para SLAM visual em tempo real.

A biblioteca ARUCO permite o uso de um único marcador ou de vários marcadores compartilhando uma mesma transformação rígida, no que se denomina de prancha de marcadores (Figura 5.5c). Para os efeitos desta tese, uma prancha de marcadores é denominada sem perda de generalidade como um marcador. Desta forma, um marcador com vários padrões quadrados é empregado na detecção de fechamentos de laço, fornecendo assim uma quantidade maior de informação para o cálculo da homografia.

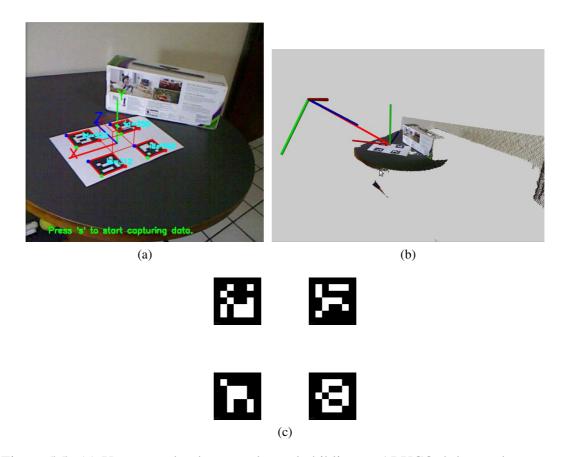


Figura 5.5: (a) Uma prancha de marcadores da biblioteca ARUCO é detectada em uma imagem de aparência. (b) Visualização tridimensional de uma nuvem de pontos RGB-D referenciada com a pose computada em relação ao marcador. (c) Padrão binário de intensidade de imagens relativo a uma prancha ARUCO. Para os efeitos deste trabalho, uma prancha é considerada um marcador.

Note que a pose do marcador não é usada em substituição à pose da odometria visual, isto é, o marcador é empregado com o único propósito de fornecer relações adicionais necessárias ao fechamento de laço no grafo de poses. Note também que é possível utilizar mais de um marcador (mais de uma prancha) no processo de SLAM. Entretanto, isto não é abordado nesta tese, uma vez que este processo requer calibração extrínseca entre cada marcador envolvido. Apesar de necessitar intervenção humana, o uso de um único marcador no processo de minimização de erro é implementado sem maiores problemas, bastando que um único marcador seja inserido na cena a ser mapeada para que o registro com consistência global seja computado.

### 5.2.2 Uso de Keyframes

Considerando que um sensor RGB-D captura imagens a uma taxa de aquisição de 30 Hz, adicionar a pose referente a cada imagem como vértice no grafo pode fazer com que este cresça rapidamente em número de vértices e arestas. Para contornar este contra-

tempo, procede-se com um artifício comumente utilizado em abordagens de SLAM visual [Mouragnon et al. 2006, Klein e Murray 2007, Kerl et al. 2013a, Endres et al. 2012] denominado seleção de *keyframes*. *Keyframes* são subamostras de imagens potencialmente significativas em relação às outras imagens de uma sequência. Critérios de seleção podem envolver a porção da cena visualizada ou o intervalo de tempo compartilhados entre *keyframes* adjacentes.

Com subamostragem, os vértices do grafo a ser otimizado passam a ser compostos apenas por *keyframes*, reduzindo assim a quantidade de informação redundante no mapa. O ferramental de minimização de erro em grafos até então discutido é diretamente aplicável ao uso de *keyframes*.

Resta portanto apresentar a heurística de seleção de *keyframes* implementada pelo método proposto, a qual está intrinsecamente relacionada com a forma de rastreamento realizado. No rastreamento KLT, um *keyframe* pode ser selecionado de forma trivial, baseado na lógica de que, enquanto a detecção de novos pontos não for disparada, uma mesma porção da cena está sendo visualizada. Assim, um *keyframe* é selecionado sempre que novos pontos são detectados. Por sua vez, o rastreamento KLTTW necessita de outra heurística, uma vez que novos pontos são adicionados em cada nova imagem. É desejável que a sobreposição de pontos visualizados seja a maior possível entre *keyframes*, assim como a distância geométrica, embora maximizar uma destas condições implique automaticamente na minimização da outra. Considerando a quantidade de pontos em comum entre as imagens, é possível estabelecer uma métrica que alcance um equilíbrio na maximização de ambos os critérios. Sendo  $M_k$  o número de features presentes no último *keyframe* e  $M_t$  o número de pontos no rastreador no instante de tempo atual, um *keyframe* é selecionado no rastreamento KLTTW sempre que a condição

$$|M_k - M_t| > f_{\rm kf} M_k \tag{5.2}$$

é verdadeira, ou seja, sempre que a diferença em módulo na quantidade de features entre o último keyframe e a imagem atual for superior a uma porcentagem  $f_{kf}$  das features presentes no último keyframe. A primeira imagem é sempre selecionada como o primeiro keyframe.

### 5.2.3 Construção do Grafo

No sistema de SLAM visual com marcadores artificiais, o grafo é construído à medida em que a odometria visual é computada de imagem para a imagem e *keyframes* são selecionados. Isto implica em, sempre que uma imagem  $I_t(u,v)$  é selecionada como *keyframe*, o sistema de referência da câmera correspondente, dado pela pose de câmera  $\mathbf{x}_t$ , é adicionado como vértice no grafo como  $\mathbf{x}_k$ . Restrições são adicionadas através de dois tipos diferentes de arestas:

- 1. Aresta de odometria, composta pela observação  $\mathbf{z}_{k-1,k}^o$  e matriz de informação  $\Omega_{k-1,k}$
- 2. Aresta de fechamento de laço, composta pela observação  $\mathbf{z}_{k,0}^f$  e matriz de informação  $\Omega_{k,0}$

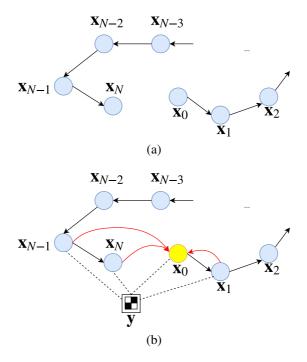


Figura 5.6: (a) Grafo obtido por odometria visual e (b) grafo obtido por SLAM visual com o uso de marcadores artificiais. Em (a), o grafo é composto apenas por arestas de odometria  $\mathbf{z}_{k-1,k}^o$  (setas pretas) que relacionam vértices de *keyframes* sequenciais  $\mathbf{x}_{k-1}$  e  $\mathbf{x}_k$ . O grafo não possui restrições suficientes para ser otimizado caso possua somente este tipo de aresta. Em (b), sempre que o marcador com pose  $\mathbf{y}$  no referencial global é detectado (linhas tracejadas) em alguma imagem  $I_t(u,v)$ , uma aresta  $\mathbf{z}_{k,0}^f$  de fechamento de laço (setas vermelhas) que relaciona o vértice  $\mathbf{x}_k$  com a origem  $\mathbf{x}_0$  é adicionada ao grafo, permitindo que o erro de mapeamento acumulado seja minimizado.

As arestas de odometria (Figura 5.6a) são compostas por relações entre o último key-frame selecionado (índice k) e o imediatamente anterior (índice k-1). A distribuição de probabilidade da observação relacionada a cada aresta deste tipo possui média  $\mathbf{z}_{k-1,k}^{o}$ , obtida pela transformação relativa  $\mathbf{T}_{k,k-1}$  (Equação 4.13), e matriz de informação  $\Omega_{k-1,k}$ . Apesar de ser possível estimar esta matriz a partir da modelagem de erro do sensor [Khoshelham e Elberink 2012] e da propagação da incerteza na nuvem de pontos para o movimento relativo, todas as matrizes de informação  $\Omega_{i,j}$  utilizadas na implementação atual são dadas pela matriz identidade  $\mathbf{I}$ .

As arestas de fechamento de laço (Figura 5.6b) são adicionadas ao grafo conforme o marcador é detectado em cada imagem de entrada. Especificamente, caso o marcador seja detectado em uma imagem  $I_t(u,v)$ , esta é automaticamente selecionada como *keyframe* e uma aresta  $\mathbf{z}_{k,0}^f$  de fechamento de laço é adicionada ao grafo. A restrição associada a esta aresta é dada pelo sistema de coordenadas referente à origem ( $\mathbf{x}_0$ ) observado a partir do referencial da pose  $\mathbf{x}_k$ . Para obter a média da distribuição de probabilidade referente à observação, computa-se a pose do último *keyframe*  $\mathbf{x}_k^m$  no referencial do marcador e a partir desta pose, obtém-se a transformação adicional  $\mathbf{T}_{0,k}'$  que registra o *keyframe* atual

com a origem de acordo com

$$\mathbf{T}'_{0,k} = \mathbf{x}_k^m \ominus \mathbf{x}_0.$$

Para que a minimização possa ser realizada, é necessário que o sistema de coordenadas global da odometria visual passe a ser o mesmo que o referencial do marcador artificial, denotado por  $\mathbf{y}$ . Assim, a odometria visual não é computada até que o marcador seja detectado pela primeira vez, quando é determinado o sistema de coordenadas global dada pela pose de câmera  $\mathbf{x}_i^m$  em relação ao marcador em um instante de tempo  $i \geq 0$  qualquer. No grafo, o vértice  $\mathbf{x}_0$  é mantido fixo, ou seja, os seus parâmetros não são modificados durante a otimização. O processo descrito é exibido em forma de pseudo-código no Algoritmo 5, cuja estratégia de rastreamento de features na odometria visual é o rastreamento KLTTW.

#### Algoritmo 5 Construção do Grafo de Poses

```
Entrada: sequência de imagens RGB-D I_0(u,v), I_1(u,v),...
Saída: conjunto de vértices \mathcal{V} e conjunto de arestas \mathcal{E} do grafo
 1: inicializado ← falso
 2: para todo I_t(u, v) faça
        se não inicializado então
 3:
 4:
           detecte o marcador na imagem I_t(u, v)
           se marcador detectado então
 5:
              compute a pose \mathbf{x}_0^m da câmera atual em relação ao marcador
 6:
 7:
              inicialize a pose inicial da odometria visual \mathbf{x}_0 como \mathbf{x}_0^m
              adicione o vértice \mathbf{x}_0 ao grafo e o mantenha como fixo
 8:
              inicializado ← verdadeiro
 9:
           fim se
10:
        senão
11:
           realize o rastreamento com o Algoritmo 3
12:
13:
           compute a mudança de pose \mathbf{T}_{t,t-1} com o Algoritmo 4
           compute a pose atual \mathbf{x}_t com a Equação 4.1
14:
           se \mathbf{x}_t selecionado como keyframe de acordo com a Equação 5.2 então
15:
16:
              adicione o vértice \mathbf{x}_t ao grafo como \mathbf{x}_k
              adicione ao grafo a aresta de odometria \mathbf{z}_{k-1,k}^o = \mathbf{T}_{k,k-1}, \Omega_{k-1,k} = \mathbf{I}
17:
           fim se
18:
19:
           detecte o marcador na imagem I_t(u,v)
20:
           se marcador detectado então
21:
              selecione \mathbf{x}_t como keyframe e o adicione ao grafo como \mathbf{x}_k
              compute a pose \mathbf{x}_k^m da câmera atual em relação ao marcador
22:
              compute a transformação de fechamento de laço \mathbf{T}'_{0,k} = \mathbf{x}_k^m \ominus \mathbf{x}_0
23:
              adicione ao grafo a aresta de fechamento de laço \mathbf{z}_{k,0}^f = \mathbf{T}_{0,k}', \Omega_{k,0} = \mathbf{I}
24:
           fim se
25:
        fim se
26:
27: fim para
```

O registro incremental é realizado normalmente a cada nova imagem  $I_t(u,v)$ , ou seja,  $\mathbf{T}_{t,t-1}$  é computada para todo par de imagens formado por  $I_{t-1}(u,v)$  e  $I_t(u,v)$ . Isto possibilita por exemplo mapear uma área distante do marcador, sendo adicionadas novas restrições de fechamento de laço somente quando o mesmo for detectado novamente. Ou seja, a pose do marcador  $\mathbf{x}_k^m$  é utilizada somente para adicionar restrições visando a correção do erro, e não em substituição às (possivelmente incorretas)  $\mathbf{x}_k$  obtidas pela odometria visual. Reciprocamente, caso a pose  $\mathbf{x}_k$  não possua erros significativos,  $\mathbf{x}_k \simeq \mathbf{x}_k^m$ , o que é comum com o marcador visível no início da trajetória. Na Figura 5.7, é possível visualizar o grafo de poses para um exemplo de mapeamento.

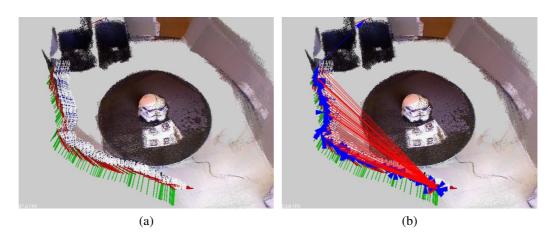


Figura 5.7: Exemplo de grafo construído a partir de nuvens de pontos RGB-D. Em (a), são exibidos os sistemas de coordenadas correspondentes à pose de cada *keyframe*, ou seja, a pose de cada vértice no grafo de poses. Em (b), arestas de odometria  $\mathbf{z}_{k-1,k}^o$ , que conectam *keyframes* sequenciais são exibidas em azul, enquanto arestas de fechamento de laço  $\mathbf{z}_{k,0}^f$ , que conectam o *keyframe* mais recente com o vértice correspondente à origem, são exibidas em vermelho.

# 5.2.4 Resolução Iterativa do Sistema Linearizado por Mínimos Quadrados

Após serem obtidos toda a trajetória  $\mathbf{x}_{0:N}$ , o conjunto de vértices  $\mathcal{V}$  e o conjunto de arestas  $\mathcal{E}$  do grafo, o sistema não-linear pode ser otimizado, visando uma configuração dos vértices do grafo que sejam o mais consistente possíveis com todas as observações [Grisetti et al. 2010]. Como discutido na Seção 4.2.2, o processo de minimização do erro em grafo de poses é realizado de forma iterativa. Em cada iteração, é computado um refinamento  $\Delta \mathbf{x}_{0:N}$  da estimativa inicial  $\mathbf{x}_{0:N}$  baseado na derivada do erro relacionado a cada aresta. Este refinamento é acumulado na solução inicial, com este processo repetido até que um critério de parada seja atingido.

A trajetória  $\mathbf{x}_{0:N}$  computada pela odometria visual é usada como estimativa inicial  $\mathbf{x}_{0:N}$ . Consequentemente, caso esta trajetória seja de má qualidade, o processo de minimização pode não convergir, o que reflete a importância do algoritmo de registro incremental

de nuvem de pontos. Cada aresta de  $\mathcal{E}$  é processada separadamente, sendo computado o erro  $\mathbf{e}_{i,j}$  de acordo com a Equação 4.14, e também o Jacobiano  $\mathbf{J}_{i,j}$  do erro em relação aos vértices  $\mathbf{x}_i$  e  $\mathbf{x}_j$ , como mostra a Equação 4.18 e 4.23. As contribuições  $\mathbf{H}_{i,j}$  e  $\mathbf{b}_{i,j}$  decorrentes da linearização do sistema são computadas e somadas conforme as Equações 4.21 e 4.20, formando a matriz  $\mathbf{H}$  e o vetor  $\mathbf{b}$ .

Ao invés de resolver diretamente o sistema linearizado  $\mathbf{H}\Delta\mathbf{x}=-\mathbf{b}$ , procede-se com uma solução dada pelo algoritmo Levenberg-Marquardt [Marquardt 1963], que por sua vez computa a solução  $\Delta\mathbf{x}$  do sistema linearizado dado por  $(\mathbf{H}-\lambda\mathbf{I})\Delta\mathbf{x}=-\mathbf{b}$ . O parâmetro  $\lambda$  introduz um fator de amortecimento, para evitar que a minimização não convirja. À medida em que soluções são computadas que melhore a soma dos erros  $F(\mathbf{x}_{0:N})$  (Equação 4.15) em relação à iteração anterior,  $\lambda$  tem o seu valor decrementado, sendo este incrementado caso aconteça o contrário. Quando uma solução pior é computada, esta é ignorada, com uma nova estimativa para o incremento sendo obtida para o novo valor de  $\lambda$ . A fatorização esparsa de Cholesky [Davis 2006], que explora a estrutura esparsa das matrizes envolvidas, é aplicada para resolver cada iteração no algoritmo de Levenberg-Marquardt. O pseudo-código para o processo é mostrado no Algoritmo 6, onde a rotina compute\_LM checa se a solução melhora ou piora a solução anterior e restaura o valor de  $\Delta\mathbf{x}$  caso uma solução pior seja obtida.

#### Algoritmo 6 Resolução Iterativa de Mínimos Quadrados

```
Entrada: estimativa inicial \check{\mathbf{x}}, conjunto de vértices \mathcal{V} e arestas \mathcal{E}
Saída: estimativa ótima x*
  1: enquanto não convergiu faça
  2:
            b \leftarrow 0
  3:
            H \leftarrow 0
            para todo aresta de \mathcal{E} que relaciona os vértices \mathbf{x}_i e \mathbf{x}_j faça
  4:
  5:
                compute o erro \mathbf{e}_{i,j} de acordo com a Equação 4.14
                compute o Jacobiano de \mathbf{e}_{i,j} com as Equações 4.18 e 4.23
  6:
                insira as contribuições da restrição na matriz H, de acordo com:
  7:
                \mathbf{H}_{i,i} + = \mathbf{A}_{i,j}^t \Omega_{i,j} \mathbf{A}_{i,j}
  8:
                \mathbf{H}_{i,j} + = \mathbf{A}_{i,j}^t \mathbf{\Omega}_{i,j} \mathbf{B}_{i,j}
  9:
                \mathbf{H}_{j,i} + = \mathbf{B}_{i,j}^t \Omega_{i,j} \mathbf{A}_{i,j}
 10:
                \mathbf{H}_{i,j} + = \mathbf{B}_{i,j}^t \Omega_{i,j} \mathbf{B}_{i,j}
 11:
                compute o vetor de coeficientes b, de acordo com:
 12:
                \mathbf{b}_i + = \mathbf{A}_{i,j}^t \Omega_{i,j} \mathbf{e}_{i,j}
13:
                oldsymbol{b}_j + = \mathbf{B}_{i,j}^{f^*} \Omega_{i,j} \mathbf{e}_{i,j}
 14:
 15:
            fim para
            \Delta \mathbf{x} \leftarrow \text{compute\_LM}((\mathbf{H} + \lambda \mathbf{I})\Delta \mathbf{x} = -\mathbf{b})
 16:
            atualize a solução inicial com \mathbf{x} + \Delta \mathbf{x}
 17:
 18: fim enquanto
19: \mathbf{x}^* \leftarrow \mathbf{\breve{x}}
```

# Capítulo 6

# Experimentos e Resultados

As capacidades e o desempenho do método de registro de nuvens de pontos proposto são avaliados em uma série de experimentos com objetivos distintos. Nesta seção, são apresentados o ferramental sob o qual o sistema é aferido, a metodologia empregada e os objetivos específicos de cada um deles.

O primeiro experimento tem como objetivo justificar a investigação do rastreamento por fluxo óptico piramidal esparso como alternativa ao rastreamento por detecção, comumente encontrado em algoritmos do estado da arte em registros de nuvens de pontos RGB-D. Em seguida, são detalhadas algumas características das estratégias KLT e KLTTW de rastreamento por fluxo óptico e as escolhas pelos parâmetros usados na implementação. Uma terceira série de experimentos avalia o desempenho (em termos de precisão e tempo de computação) do método proposto no registro de nuvens de pontos para odometria visual, ou seja, sem minimização global de erro. A minimização global do erro implementada pela versão *full* SLAM do sistema proposto é avaliada no último experimento.

Todos os experimentos foram executados em um PC desktop com processador de quatro núcleos do tipo Intel Core i5 de 3.2 Ghz e 8 GB de memória RAM.

### 6.1 Conjuntos de Dados e Benchmark TUM RGB-D

Através de ferramentas de avaliação (benchmark) e conjuntos de dados disponibilizados publicamente pela Technische Universität München (TUM) [Sturm et al. 2012], um padrão de medidas de erro em dados comuns é estabelecido, facilitando a comparação entre algoritmos de SLAM baseados em sensores RGB-D. O sistema proposto é avaliado fazendo uso extensivo destes conjuntos de dados, justificado pelo fato de as imagens RGB-D coletadas serem disponibilizadas juntamente com um ground truth (medições tidas como verdadeiras, ou seja, um gabarito), obtidas por câmeras e marcadores infravermelhos de sistemas de captura de movimentos (motion capture). Desta forma, para cada imagem RGB-D processada por um algoritmo, é possível comparar a pose computada com a pose verdadeira, e a partir disto, determinar a precisão do sistema sob avaliação.

Um total de doze conjuntos de dados capturados a 30 Hz em ambientes distintos e com diferentes características é empregado nos experimentos. A Tabela 6.1 sumariza as diferentes características encontradas nas imagens, tais como distância total percorrida,

total de imagens e velocidades linear e angular. Para facilitar a avaliação do algoritmo, as sequências de imagens estão divididas em 1. sequências de depuração, onde a câmera realiza movimentos exclusivamente de translação ou rotação ao longo/em torno de cada eixo, 2. sequências de SLAM manual, onde a câmera é manualmente operada de forma irrestrita (6 graus de liberdade) para mapear ambientes formados por objetos de escritório (cadeiras, mesas, livros, computadores, etc.) com várias features visuais e 3. sequências de reconstrução 3D de objetos, nas quais o sensor é manuseado com o objetivo de escanear um objeto específico, obtendo assim uma representação digital do mesmo.

Tabela 6.1: Detalhes dos conjuntos de dados TUM RGB-D empregados nos experimentos de odometria visual. Cada coluna mostra a distância percorrida, o total de imagens e as médias das velocidades linear e angular para cada sequência, agrupadas em sequências de depuração, SLAM manual e reconstrução 3D de objetos.

Seq.	Dist. Perc.	Imagens	Vel. Lin. Méd.	Vel. Ang. Méd.		
Depuração:						
fr1/xyz	7,112 m	792	0,244 m/s	8,920°/s		
fr1/rpy	1,664 m	694	0,062 m/s	50,147°/s		
fr2/xyz	7,029 m	3615	0,058 m/s	$1,716^{\circ}/s$		
fr2/rpy	1,506 m	3221	0,014 m/s	5,774°/s		
SLAM Manual:						
fr1/room	15,989 m	1352	0,334 m/s	29,882°/s		
fr1/floor	12,569 m	979	0,258 m/s	15,071°/s		
fr1/desk	9,236 m	573	0,413 m/s	23,327°/s		
fr1/desk2	10,161 m	620	0,426 m/s	29,308°/s		
fr1/360	5,818 m	744	0,210 m/s	41,600°/s		
fr2/desk	18,880 m	2893	0,193 m/s	6,338°/s		
Reconstrução 3D de Objetos:						
fr1/plant	14,795 m	1126	0,365 m/s	27,891°/s		
fr1/teddy	15,709 m	1401	0,315 m/s	21,320°/s		

# 6.2 Comparação de Rastreamento para Registro de Nuvens de Pontos

Desejamos justificar as razões pelas quais investir em um algoritmo bem conhecido da literatura como o fluxo óptico esparso piramidal pode se mostrar como uma boa escolha para compor o *front-end* de SLAM visual. Para isso, um experimento de rastreamento de features para registrar pares de nuvens de pontos RGB-D é conduzido. Especificamente, o objetivo é mostrar que o rastreamento por linha de base curta via fluxo óptico resulta em uma fração considerável de pontos rastreados, ao mesmo tempo em que é mantida uma fração muito pequena de falsas correspondências, principalmente se comparado com

rastreamento por linha de base larga encontrados em sistemas de registro de nuvens de pontos RGB-D [Henry et al. 2010, Endres et al. 2012, Paton e Kosecka 2012].

Recorre-se a uma metodologia experimental [Ke e Sukthankar 2004, Mikolajczyk e Schmid 2005, Nascimento et al. 2012] frequentemente utilizada na avaliação de desempenho de pontos característicos e descritores. Nessa abordagem, a obtenção de correspondências entre features de imagem para imagem é tratado como um problema de classificação, ou seja, uma determinada feature pode ter o seu casamento:

- Estabelecido corretamente (verdadeiro positivo)
- Estabelecido incorretamente (falso positivo)
- Corretamente não estabelecido (verdadeiro negativo)
- Incorretamente não estabelecido (falso negativo)

Com o auxílio do *ground truth* presente nos conjuntos de dados TUM RGB-D, é possível determinar se uma correspondência entre uma feature  $\mathbf{p}_t^i$  e o seu par segundo o rastreamento  $\mathbf{p}_{t+1}^j$  é um casamento estabelecido corretamente. Para isso, é extraído o conjunto  $\mathcal{P}$  formado por um número constante M de pontos característicos em cada imagem  $I_t(u,v)$  e estabelecidos os casamentos para cada um deles na imagem  $I_{t+1}(u,v)$ , de acordo com o algoritmo de rastreamento. Cada ponto  $\mathbf{p}_t^i$  é convertido no seu ponto 3D  $\mathbf{P}_t^i$  da nuvem de pontos (Equação 2.15) e transformado de acordo com  $\mathbf{G}_{t+1,t}$ , a transformação rígida de rotação e translação obtida a partir do *ground truth* que relaciona as imagens  $I_t(u,v)$  e  $I_{t+1}(u,v)$ . Após projetar o ponto transformado  $\hat{\mathbf{P}}^i$  nas coordenadas  $\hat{\mathbf{p}}^i$  da imagem  $I_{t+1}(u,v)$  usando a Equação 2.2, a distância Euclidiana d(.) entre  $\mathbf{p}_{t+1}^j$  e  $\hat{\mathbf{p}}^i$  informa se a correspondência está correta ou não, de acordo com um limiar  $\gamma$ . Matematicamente,  $\mathbf{p}_t^i$  e  $\mathbf{p}_{t+1}^j$  é considerado um casamento correto se

$$d(\mathbf{p}_{t+1}^j, \hat{\mathbf{p}}^i) \le \gamma, \tag{6.1}$$

onde

$$\hat{\mathbf{P}}^i = \mathbf{R}_{t+1,t}^g \mathbf{P}_t^i + \mathbf{t}_{t+1,t}^g$$

e o par  $\mathbf{R}_{t+1,t}^g$ ,  $\mathbf{t}_{t+1,t}^g$  é obtido de  $\mathbf{G}_{t+1,t}$ .

A avaliação de desempenho de algoritmos de rastreamento pode ser analisada pela visualização de curvas 2D do tipo 1-precisão vs. sensibilidade, onde 1-precisão informa a proporção de casamentos falsos dentre os casamentos estabelecidos, ou seja,

$$1\text{-precisão} = \frac{\text{núm. de casamentos falsos (falsos positivos)}}{\text{núm. total de casamentos (corretos ou falsos)}},$$

e a sensibilidade (*recall*) informa a fração do total M de pontos detectados em  $I_t(u,v)$  que teve um casamento corretamente estabelecido na imagem  $I_{t+1}(u,v)$ ,

$$sensibilidade = \frac{\text{n\'um. de casamentos corretos (verdadeiros positivos)}}{\text{n\'um. total de pontos a serem rastreados}}.$$

As médias obtidas após processar uma sequência de imagens dão origem a um ponto no plano formado pelo eixo horizontal sendo a medida 1-precisão e o vertical sendo a sensibilidade. Para gerar os demais pontos da curva, varia-se o limiar que estabelece correspondências entre features usando diversos valores possíveis, de acordo com o algoritmo de rastreamento em específico.

O rastreamento por fluxo óptico piramidal esparso de features Shi-Tomasi (rastreamento KLT) é comparado com o rastreamento por detecção e casamento por vizinhos mais próximos de descritores de features SURF. Este último método foi escolhido por unir desempenho em termos de precisão de rastreamento e também em termos de tempo de computação.

No KLT, varia-se o limiar para a distância de Manhattan  $d_1(.)$  entre vetores formados pelos valores das intensidades nas imagens na vizinhança de  $\mathbf{p}_t^i$  e  $\mathbf{p}_{t+1}^j$ , sendo esta distância normalizada pelo número de pixels na janela, isto é

$$d_1(\mathbf{s}^i, \mathbf{s}^j) = \frac{1}{W \times W} \sum_{k=1}^{k=W \times W} |s_k^i - s_k^j|,$$

onde  $\mathbf{s}^i$  e  $\mathbf{s}^j$  são os *patches* de tamanho  $W \times W$  em torno de  $\mathbf{p}_t^i$  e  $\mathbf{p}_{t+1}^j$  respectivamente. Caso esta distância seja menor ou igual a um valor  $\gamma_K$ , uma correspondência entre  $\mathbf{p}_t^i$  e  $\mathbf{p}_{t+1}^j$  é estabelecida. No SURF, o limiar para a distância Euclidiana d(.) entre o descritor  $\mathbf{d}^i$  de  $\mathbf{p}_t^i$  e o seu vizinho mais próximo no espaço de 64 dimensões dos descritores SURF é variado, sendo uma correspondência estabelecida entre  $\mathbf{p}_t^i$  e  $\mathbf{p}_{t+1}^j$  se  $d(\mathbf{d}^i, \mathbf{d}^j)$  for menor ou igual a um valor  $\gamma_S$ . Note que uma correspondência estabelecida não é necessariamente uma correspondência verdadeira, daí a importância de analisar os gráficos em questão.

Frações das distâncias máximas foram utilizadas para variar os valores dos limiares  $\gamma_K$  e  $\gamma_S$  para geração das curvas, com valores escolhidos iguais a 2%, 4%, 6%, 8%, 10%, 25%, 35%, 50%, 75% e 100% de 255 (valor máximo para  $\mathbf{d}_1(.)$  entre *patches* de imagens) e 2 (valor máximo para  $\mathbf{d}(.)$  entre descritores unitários). Quanto maior é este valor, mais liberal se torna o classificador, no sentido de que este abre mão da certeza dos casamentos para obter correspondências para mais pontos. Um casamento é considerado correto caso a condição na Equação 6.1 seja verdadeira para o valor  $\gamma$  igual a 5 pixels.

Os resultados do experimento em seis conjuntos de dados selecionados são mostrados com as curvas 1-precisão vs. sensibilidade computadas na Figura 6.1.

Ao realizar a análise das curvas resultantes para estudar o desempenho dos dois tipos de rastreamento empregados, o melhor algoritmo tem um comportamento ótimo determinado teoricamente: 100% de sensibilidade com 0% de 1-precisão, ou seja, o algoritmo acha correspondências corretas para todos os pontos característicos de cada imagem  $I_t(u,v)$ . Obter este desempenho ideal é impossível na prática, sendo considerado um bom desempenho obter taxas elevadas de sensibilidade ao mesmo tempo em que falsas correspondências são baixas. Tal meta é alcançada pelo rastreamento por fluxo óptico esparso KLT, como evidenciado com os pontos presentes no canto superior esquerdo dos gráficos. Em comparação com o rastreamento por detecção realizado pelo SURF, o KLT possui uma medida de falsas correspondências consideravelmente menor para uma mesma medida de sensibilidade. Esta propriedade do rastreamento é explorada tanto para garantir

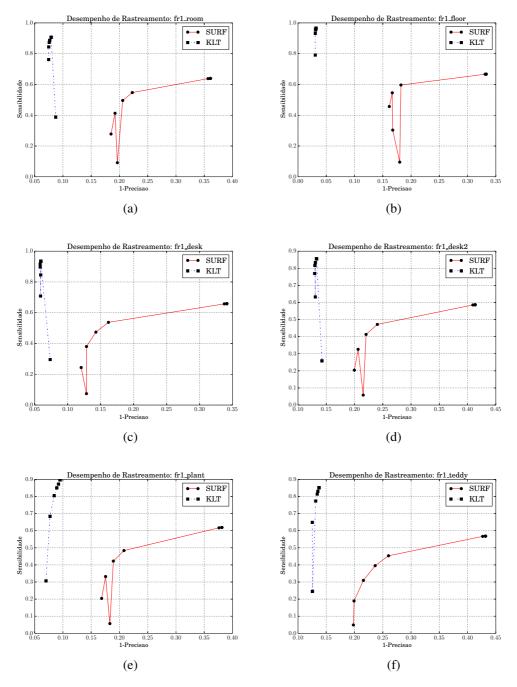


Figura 6.1: Rastreamento de imagem para imagem. O desempenho do rastreamento por fluxo óptico KLT é comparado com o desempenho do rastreamento por detecção de features SURF. O eixo horizontal mostra a porcentagem de falsas correspondências, enquanto o eixo vertical mostra a porcentagem dos pontos em  $I_t(u,v)$  que foram rastreados em  $I_{t+1}(u,v)$  (sensibilidade). Observe que para uma mesma sensibilidade, o rastreamento por fluxo óptico possui uma quantidade consideravelmente menor de falsas correspondências.

uma maior precisão na estimação do movimento de imagem para imagem quanto para acelerar este mesmo processo, já que o RANSAC se beneficia de uma maior porção de casamentos corretos. Os resultados do experimento condizem com os obtidos em um experimento similar [Klippenstein e Zhang 2007], no qual o rastreamento KLT foi analisado com a transformação de imagem para imagem modelada pela matriz fundamental [Hartley e Zisserman 2004].

#### 6.3 Medidas de Erro em Sistemas de SLAM

O ferramental disponibilizado pelo *benchmark* TUM RGB-D oferece duas medidas de erro diferentes para aferir a precisão de sistemas de localização e mapeamento. Como investigado em trabalhos relacionados [Kümmerle et al. 2009, Sturm et al. 2012], as métricas podem fornecer informações a respeito de diferentes capacidades em SLAM.

A medida de erro denominada Erro Absoluto de Trajetória (*Absolute Trajectory Error* – ATE) compara diretamente cada pose de sensor computada  $\mathbf{x}_t$  com a pose dada pelo *ground truth*  $\mathbf{g}_t$  associada ao mesmo instante de tempo da imagem  $I_t(u,v)$ , após as duas trajetórias  $\mathbf{x}_{1:N}$  e  $\mathbf{g}_{1:N}$  serem alinhadas em um sistema de referência comum. O erro ATE  $\mathbf{E}_t^A$  para uma imagem  $I_t(u,v)$  é dado pela pose

$$\mathbf{E}_t^A = \mathbf{g}_t \ominus \mathbf{x}_t.$$

Uma estatística é comumente computada sobre ele, como a média  $med(\mathbf{E}_{1:N}^A)$ , dada por

$$med(\mathbf{E}_{1:N}^{A}) = \frac{1}{N} \sum_{t=1}^{N} ||trans(\mathbf{E}_{t}^{A})||,$$
(6.2)

onde trans(.) retorna o vetor de translação da pose de câmera e ||.|| é a norma Euclidiana do vetor.

A Figura 6.2 exibe o motivo pelo qual o erro ATE é inadequado para aferir a precisão de sistemas de odometria visual (que não aplicam correção de erro). Nos dois exemplos de SLAM unidimensional ilustrados, há apenas um erro de estimação, com valor igual a K, ao longo de toda a trajetória  $\mathbf{x}_{1:N}$ . No primeiro caso, o erro para todos os instantes de tempo até t = N - 1 é nulo, sendo igual a K no instante t = N, o que faz com que o somatório do erro seja igual a K. No segundo caso, o erro de estimação acontece em t = 2, fazendo com que este erro seja indevidamente propagado para as poses restantes até t = N e consequentemente, resultando em um somatório igual a  $K \times (N - 1)$ .

O Erro de Pose Relativa (*Relative Pose Error* – *RPE*) foi proposto para fornecer uma forma mais adequada e abrangente na avaliação de sistemas de SLAM [Kümmerle et al. 2009]. Ao invés de poses  $\mathbf{x}_t$  serem comparadas diretamente ao *ground truth* associado  $\mathbf{g}_t$ , a estimativa de mudança de pose  $\mathbf{x}_i \ominus \mathbf{x}_j$  entre dois instantes de tempo quaisquer i e j é comparada com a mesma transformação dada pelo *ground truth*  $\mathbf{g}_i \ominus \mathbf{g}_j$  (Figura 6.3). Matematicamente, a medida RPE  $\mathbf{E}_{i,j}^R$  que calcula o erro de transformação relativa para a

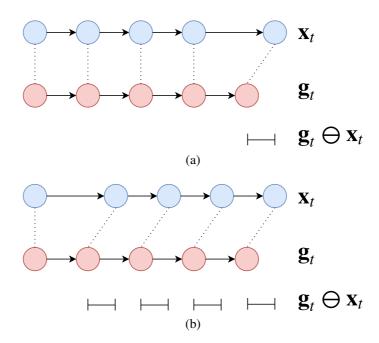


Figura 6.2: Erro Absoluto de Trajetória (ATE) e o motivo pelo qual não se deve usá-lo em odometria visual. Ambas as imagens mostram uma trajetória unidimensional estimada para o sensor (com poses  $\mathbf{x}_t$ , em azul) e as poses reais ( $\mathbf{g}_t$ , em vermelho), dadas por um ground truth. O erro  $\mathbf{E}_t^A$  deve ser igual a 0 quando as medições forem feitas corretamente. Em (a) e (b), ocorre apenas uma medição incorreta, entretanto, como em (a) esta medição ocorreu no último instante de tempo, o erro de alinhamento entre as trajetórias será consideravelmente menor do que em (b).

imagem  $I_i(u,v)$  em relação à imagem  $I_i(u,v)$  é dada pela transformação

$$\mathbf{E}_{i,j}^R = \left[ (\mathbf{x}_i \ominus \mathbf{x}_j) \ominus (\mathbf{g}_i \ominus \mathbf{g}_j) \right].$$

Ao variar os índices i e j, medidas estatísticas podem ser computadas para transformações relativas envolvendo diferentes poses da trajetória  $\mathbf{x}_{1:N}$ . Isto pode ser feito com o uso de um parâmetro  $\Delta$ , especificado para fornecer medidas de erro em um intervalo fixo de tempo, como dado por

$$\mathbf{E}_{j}^{R} = \left[ (\mathbf{x}_{j+\Delta} \ominus \mathbf{x}_{j}) \ominus (\mathbf{g}_{j+\Delta} \ominus \mathbf{g}_{j}) \right].$$

Desta forma, a raíz do erro médio quadrático (*Root Mean Squared Error* – RMSE) pode ser computada para  $N' = N - \Delta$  intervalos, fornecendo a estatística rmse( $\mathbf{E}_{1:N}^R$ ) conforme mostra a Equação 6.3.

$$\operatorname{rmse}(\mathbf{E}_{1:N}^{R}) = \left[\frac{1}{N'} \sum_{j=1}^{N'} ||\operatorname{trans}(\mathbf{E}_{j}^{R})||\right]^{1/2}$$
(6.3)

Ao usar a função rot(.) na Equação 6.3, o RMSE do erro rotacional pode ser compu-

tado, o qual fornece uma estatística robusta do erro para o ângulo associado a uma rotação em torno de um eixo arbitrário ao longo de uma trajetória.

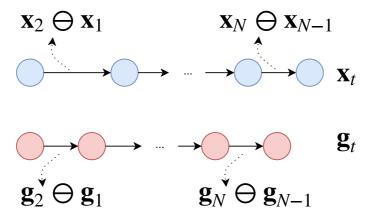


Figura 6.3: Erro de Pose Relativa (RPE). Ao invés de comparar diretamente as poses do sensor  $\mathbf{x}_t$  com o *ground truth*  $\mathbf{g}_t$  como calculado no erro ATE, o erro é computado sobre as transformações relativas para a trajetória estimada  $\mathbf{x}_2 \ominus \mathbf{x}_1$ ,  $\mathbf{x}_3 \ominus \mathbf{x}_2$ ,...,  $\mathbf{x}_N \ominus \mathbf{x}_{N-1}$  e as transformações relativas para o *ground truth*  $\mathbf{g}_2 \ominus \mathbf{g}_1$ ,  $\mathbf{g}_3 \ominus \mathbf{g}_2$ ,...,  $\mathbf{x}_N \ominus \mathbf{g}_{N-1}$ . Caso uma transformação relativa qualquer entre  $\mathbf{x}_{t-1}$  e  $\mathbf{x}_t$  não contenha erro,  $[(\mathbf{x}_t \ominus \mathbf{x}_{t-1}) \ominus (\mathbf{g}_t \ominus \mathbf{g}_{t-1}) = \mathbf{I}]$ .

Uma medida comum em trabalhos relacionados a SLAM com sensores RGB-D é usar  $\Delta=1$  segundo, resultando em um  $\mathbf{E}_{j}^{R}$  que indica o quanto de erro foi acumulado a cada segundo pelo sistema avaliado, se caracterizando assim como uma medida mais adequada para aferir odometria visual do que o erro ATE. Este último ainda é uma medida muito utilizada para avaliar sistemas de SLAM e por isso, é empregado nos experimentos que medem esta capacidade do método proposto.

## 6.4 Parametrização do Rastreamento com Fluxo Óptico

As duas estratégias de rastreamento de features visuais por fluxo óptico piramidal esparso são parametrizadas e comparadas nesta seção, com o objetivo de estabelecer qual o melhor desempenho em odometria visual. Os algoritmos KLT e KLTTW são avaliados em uma série de experimentos nas sequências de imagens do tipo depuração nos quais é investigado o comportamento de cada um deles de acordo com alterações nos seus parâmetros. A Tabela 6.2 mostra os parâmetros dos algoritmos KLT e KLTTW e os respectivos valores usados nos experimentos, com a observação que vários parâmetros são compartilhados por ambos os algoritmos.

O principal parâmetro do algoritmo KLT é o limiar  $N_{\min}$  para o número mínimo de features. Novas features são detectadas e adicionadas ao rastreador sempre que a quantidade de features se torna inferior a  $N_{\min}$ , o que pode acontecer por oclusão ou pelo fluxo óptico não ser capaz de determinar o rastreamento dos pontos. Para o KLTTW, o parâmetro que governa o desempenho do algoritmo é o tamanho  $S = S_l = S_a$  da janela de rastreamento. Caso este valor seja muito grande, um número menor de features é aceito

Param.	Descrição	Valor
$\overline{f}$	Fator de qualidade de pontos Shi-Tomasi	0,01
$D_{\min}$	Distância mínima na detecção de pontos em pixels	10
$N_f$	Nr. máximo de pontos adicionados	5000
L	Nr. de níveis na pirâmide do fluxo óptico	4
$N_{ m max}$	Nr. máximo de pontos no rastreador	5000
$N_{\min}$	Nr. mínimo de pontos no rastreador (KLT)	Variável
S	Tamanho da janela de rastreamento em pixels (KLTTW)	Variável
$N_R$	Nr. inicial de iterações do RANSAC	10000
α	Prob. de escolha de casamento inlier no RANSAC	99%
$ au_R$	Limiar do RANSAC para correspondência inlier	0,008 m

Tabela 6.2: Lista de parâmetros envolvidos na odometria visual por fluxo óptico.

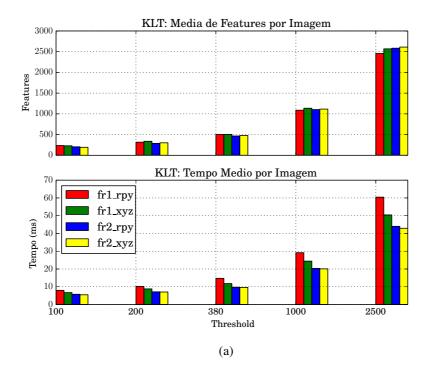
em comparação com o uso de um valor pequeno. De certa forma, parametrizar o KLTTW em função do valor da janela permite que ele seja configurado para regular a quantidade desejada de features no rastreador em cada imagem, tornando-o mais preciso (mais features) ou mais rápido (menos features), supondo obviamente que features de boa qualidade estarão disponíveis de forma constante.

Com este raciocínio, foram estabelecidas cinco configurações equivalentes para os dois algoritmos, de forma que eles mantenham a mesma quantidade média de features rastreadas por imagem. Esta média e a média do tempo total computando cada imagem das sequências do tipo depuração de acordo com diferentes valores dos parâmetros  $N_{\min}$  e S são mostradas na Figura 6.4.

As cinco configurações possuem os pares de valores  $N_{\rm min}/S$  dados por 100/50, 200/30, 380/20, 1000/10 e 2500/5. Como mostrado na Figura 6.4, estas configurações são capazes de manter uma média em torno de (respectivamente) 200, 300, 500, 1000 e 2500 pontos rastreados por imagem, exceto pela última configuração na sequência fr2/xyz (na qual a média do KLTTW chega a quase 3500 features por imagem).

Nos mesmos experimentos, calcula-se que o rastreamento por fluxo óptico piramidal esparso consome cerca de 0,03 ms para determinar a posição de cada ponto característico, ao passo que o Algoritmo 1 de detecção de features Shi-Tomasi tem tempo total de computação entre 8 ms e 12 ms, dependendo do número de features extraídas (já que o algoritmo tem uma etapa final de ordenação). O algoritmo KLTTW detecta features em cada nova imagem, tornando-o mais computacionalmente intensivo do que o KLT. Entretanto, a estratégia de rastreamento com janelas possui a importante propriedade de manter features mais uniformemente espalhadas pelas imagens. Para visualizar esta característica, a distribuição das coordenadas 2D dos pontos rastreados ao longo de cada sequência de imagens é armazenada em histogramas de tamanho 32 × 32, mostrados em forma de mapas de calor na Figura 6.5 e Figura 6.6, para as sequências fr1/xyz e fr2/xyz respectivamente.

O benefício do KLTTW sobre o KLT pode ser quantificado computando-se o erro RPE para as quatro sequências de depuração. Os resultados para cada uma das cinco



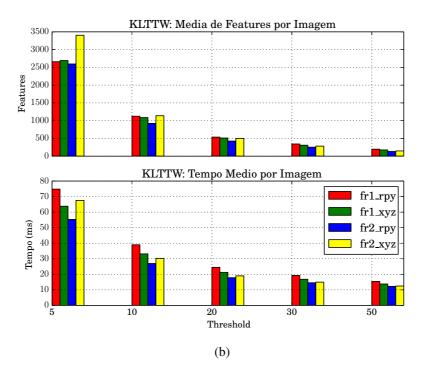


Figura 6.4: Número de features rastreadas (acima) e tempo total processando cada imagem (abaixo) para (a) Rastreamento KLT e (b) Rastreamento KLTTW, nas quatro sequências do tipo depuração. Os gráficos mostram como se comportam os valores variando-se o parâmetro  $N_{\min}$  para o KLT e S para o KLTTW.

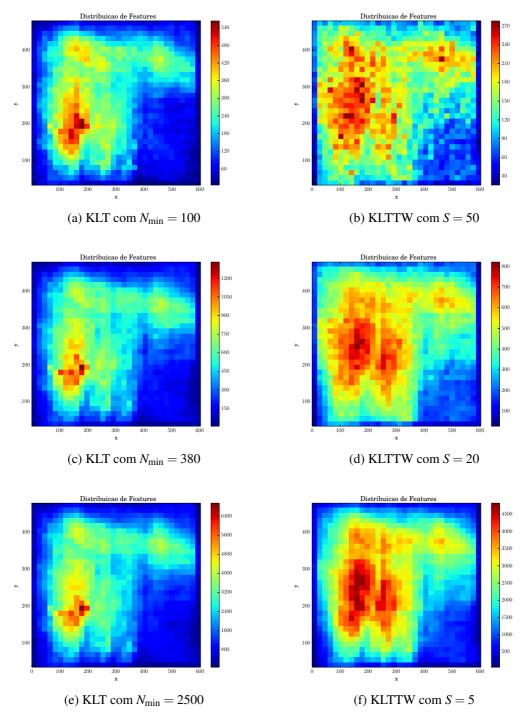


Figura 6.5: Mapa de calor representando a distribuição de features em todas as imagens da sequência fr1/xyz, no qual cada célula tem tamanho 32 × 32 pixels. A coluna da esquerda exibe configurações para o KLT, enquanto a coluna da direita exibe configurações para o KLTTW. Configurações equivalentes KLT/KLTTW estão agrupadas por linha. Observe que para uma mesma configuração, o KLTTW resulta em features mais uniformemente distribuídas.

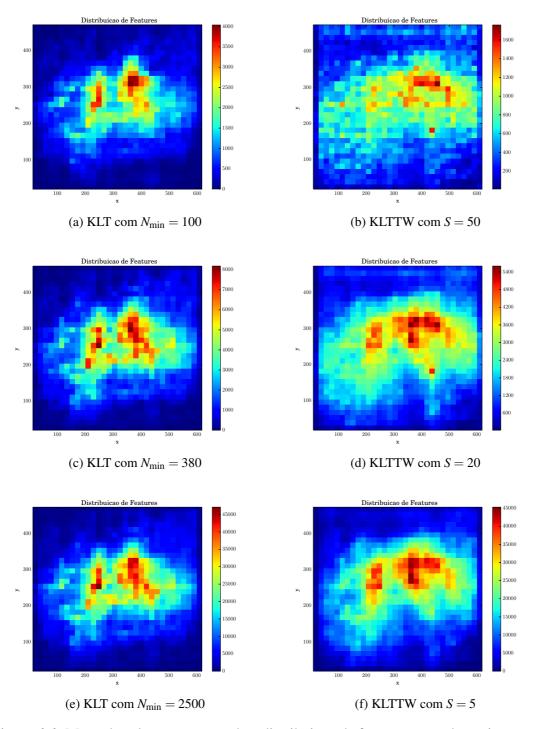


Figura 6.6: Mapa de calor representando a distribuição de features em todas as imagens da sequência fr2/xyz, no qual cada célula tem tamanho 32 × 32 pixels. A coluna da esquerda exibe configurações para o KLT, enquanto a coluna da direita exibe configurações para o KLTTW. Configurações equivalentes KLT/KLTTW estão agrupadas por linha. Observe que para uma mesma configuração, o KLTTW resulta em features mais uniformemente distribuídas.

configurações diferentes são mostrados na Tabela 6.3, a qual detalha o erro RPE translacional e rotacional de cada uma das configurações e o quanto que o KLTTW melhora as estimativas de pose de câmera do KLT. O uso de janelas de rastreamento pode resultar em 32,1% menos erro acumulado na translação e 28,7% na rotação, com médias de redução de aproximadamente 15%. Apenas na sequência de imagem fr1/rpy o KLTTW não obteve melhores resultados, o que aconteceu na configuração que mantém a média de 1000 features por imagem.

Também é importante notar que diminuir o tamanho da janela de rastreamento não só contribui com um maior tempo de processamento como também pode resultar em estimativas de pose de pior precisão, provavelmente por features de pior qualidade serem adicionadas ao rastreador. Com base nos resultados apresentados e levando em consideração que algumas configurações são inadequadas para computar pose de câmera na taxa de aquisição de imagens do sensor (30 Hz), os demais resultados apresentados, relativos ao desempenho de odometria visual e SLAM, foram obtidos configurados com o KLTTW com tamanho de janela S igual a 30 pixels.

#### 6.5 Odometria Visual RGB-D

Com o objetivo de esclarecer como a atual implementação do sistema proposto se compara com o estado da arte em registro de nuvens de pontos com sensores RGB-D, os resultados dos experimentos em odometria visual são mostrados juntamente aos estimados por dois sistemas pertencentes à classe de soluções esparsas (como discutido na Seção 3.4): o sistema FOVIS [Huang et al. 2011] e o sistema RGB-D SLAM [Endres et al. 2012].

O sistema FOVIS é, segundo o levantamento bibliográfico realizado, o único método de registro de nuvens de pontos implementado para sensores RGB-D que emprega rastreamento de linha de base curta de features visuais. Além disso, FOVIS é completamente implementado sem uso de GPU e é um algoritmo de odometria visual (sem minimização de erro). Em um estudo experimental de estimação de odometria com sensores RGB-D [Fang e Scherer 2014], FOVIS se destacou pelo seu desempenho, além de ter sido implementado como *front-end* no registro denso realizado por Whelan et al. (2013). Já o sistema RGB-D SLAM utiliza GPU para a extração de features SIFT, que requerem hardware especializado para que o sistema seja executado com o desempenho informado. Ao contrário do sistema proposto e do FOVIS, RGB-D SLAM detecta fechamento de laços e minimiza o erro de mapeamento acumulado na odometria visual. Apesar de a princípio parecer injusta, a comparação da odometria visual pura com RGB-D SLAM fornece uma noção da precisão desta classe de sistemas, principalmente em relação ao rastreamento de imagem para imagem por linha de base curta.

A implementação de código aberta do FOVIS<sup>1</sup> foi instalada e executada com a sua configuração padrão no mesmo hardware e software usado nos experimentos do sistema proposto, enquanto os resultados publicamente disponíveis<sup>2</sup> do RGB-D SLAM foram

<sup>&</sup>lt;sup>1</sup>https://github.com/srv/libfovis

<sup>&</sup>lt;sup>2</sup>https://svncvpr.in.tum.de/cvpr-ros-pkg/trunk/rgbd\_benchmark/rgbd\_benchmark\_tools/data/rgbdslam/

Tabela 6.3: Resultados da parametrização do rastreamento KLT e KLTTW. São mostrados o erro (RPE) translacional e rotacional, assim como o quanto (em %) o KLTTW melhora sobre o KLT. Os resultados estão agrupados por configurações equivalentes em termos de números de features rastreadas, no formato KLT/ $N_{\rm min}$  e KLTTW/S. Os resultados em vermelho mostram o quanto o resultado do KLTTW foi pior (e não melhor) em relação ao KLT.

Erro de Posicionamento Relativo					
	KLT		KLTTW		
Seq.	Transl. (m/s)	Rot. (°/s)	Transl. (m/s)	Rot. (°/s)	Melhoria (%)
	C	onfiguração 1:	KLT/100 e KLT	TW/50	
fr1/xyz	0,030	1,850	0,022	1,161	26,7/37,2
fr1/rpy	0,090	4,414	0,066	3,147	26,7/28,7
fr2/xyz	0,028	0,918	0,020	0,711	28,5/22,5
fr2/rpy	0,020	0,646	0,018	0,604	10,0/6,5
	C	onfiguração 2:	KLT/200 e KLT	TW/30	
fr1/xyz	0,028	1,642	0,022	1,188	21,4/27,6
fr1/rpy	0,076	4,071	0,053	3,169	30,2/22,1
fr2/xyz	0,028	0,882	0,019	0,640	32,1/27,4
fr2/rpy	0,021	0,665	0,019	0,631	9,5/5,1
	C	onfiguração 3:	KLT/380 e KLT	TW/20	
fr1/xyz	0,027	1,561	0,022	1,167	18,5/25,2
fr1/rpy	0,064	3,665	0,056	3,030	12,5/17,3
fr2/xyz	0,025	0,816	0,021	0,719	16,0/11,8
fr2/rpy	0,022	0,667	0,018	0,596	18,1/10,6
	Co	onfiguração 4:	KLT/1000 e KL7	ΓTW/10	
fr1/xyz	0,025	1,521	0,022	1,221	12,0/19,7
fr1/rpy	0,068	3,599	0,082	3,765	20,5/4,6
fr2/xyz	0,025	0,830	0,022	0,742	12,0/10,6
fr2/rpy	0,020	0,622	0,017	0,604	15,0/2,8
		• •	KLT/2500 e KL	TTW/5	
fr1/xyz	0,026	1,491	0,022	1,263	15,3/15,2
fr1/rpy	0,067	3,690	0,070	3,589	<b>4,4</b> /2,7
fr2/xyz	0,024	0,796	0,021	0,745	12,5/6,4
fr2/rpy	0,020	0,612	0,017	0,584	15,0/4,5
Média:					15,3/15,0

utilizados para computar as mesmas medidas de erro (RPE translacional e rotacional) extraídas dos outros dois sistemas.

83

#### 6.5.1 Precisão

A precisão da odometria visual com fluxo óptico KLTTW é exibida em comparação com os sistemas FOVIS e RGB-D SLAM nas Tabelas 6.4 e 6.5. São mostrados o RMSE do RPE translacional e rotacional, bem como o erro máximo de estimativa para as 12 sequências do conjunto de dados TUM RGB-D.

Tabela 6.4: Precisão da odometria visual. São mostrados o RMSE do erro de pose relativa (RPE) e erro máximo para a translação (entre parênteses) em metros por segundo.

RPE Translacional – RMSE (Máx.) em m/s						
Seq.	KLTTW		FOVIS		RGB-D SLAM	
fr1/xyz	0,022	(0,057)	0,029	(0,089)	0,021	(0,048)
fr1/rpy	0,053	(0,174)	0,059	(0,165)	0,045	(0,144)
fr2/xyz	0,019	(0,076)	0,004	(0,016)	0,006	(0,019)
fr2/rpy	0,019	(0,049)	0,004	(0,015)	0,009	(0,036)
fr1/room	0,067	(0,364)	0,056	(0,224)	0,095	(0,558)
fr1/floor	0,017	(0,038)	0,040	(0,133)	0,017	(0,043)
fr1/desk	0,040	(0,105)	0,057	(0,260)	0,033	(0,097)
fr1/desk2	0,047	(0,097)	0,060	(0,148)	0,054	(0,247)
fr1/360	0,100	(0,267)	0,080	(0,204)	0,088	(0,242)
fr2/desk	0,046	(0,106)	0,013	(0,034)	0,016	(0,092)
fr1/plant	0,055	(0,171)	0,061	(0,278)	0,061	(0,193)
fr1/teddy	0,114	(0,624)	0,071	(0,255)	0,088	(0,357)
Média:	0,049	· · · · · ·	0,044		0,044	· · · · · · ·

Uma observação importante a ser feita é que os resultados mostram que não há superioridade absoluta de nenhum método específico, ao contrário do que poderia ser suposto à respeito do RGB-D SLAM. Em grande parte, isto se deve ao rastreamento por linha de base curta, principalmente em relação ao KLTTW, que possui resultados próximos ou superiores ao FOVIS e ao RGB-D SLAM, com exceção das sequências com prefixo fr2 (tratadas como um caso à parte).

O erro RPE translacional é reduzido em pelo menos 9,8% e em até 57,5% em relação ao FOVIS (como acontece nas sequências fr1/plant e fr1/floor). Quando o RGB-D SLAM computa melhores estimativas do que o FOVIS, a melhoria máxima chega a 12,9% (sequência fr1/desk2). O pior resultado possui RMSE 60,5% maior do que o FOVIS, o que acontece devido à sequência de imagens fr1/teddy possuir um intervalo de imagens sem features visuais de boa qualidade. Processando apenas as primeiras 1000 nuvens de pontos do total de 1401 da sequência, o RMSE obtido é de 0,074 m/s, mais próximo do valor igual a 0,059 m/s para o FOVIS nas mesmas imagens. Nas sequências fr1/room e fr1/360, o KLTTW computa estimativas entre 19% e 25% piores. Em média, o KLTTW acumula erro translacional a uma taxa de 0,049 m/s, enquanto estes valores são iguais a 0,044 m/s, tanto para o FOVIS quanto para o RGB-D SLAM.

Estimativas para o erro RPE rotacional são ainda mais competitivas, nas quais a pre-

		RPE Rotacion	nal – RMSE	(Máx.) em °/	S	
Seq.	KLTTW		KLTTW FOVIS		RGB-D SLAM	
fr1/xyz	1,188	(2,837)	1,824	(5,403)	0,934	(2,295)
fr1/rpy	3,169	(8,553)	18,499	(56,706)	2,735	(8,350)
fr2/xyz	0,640	(2,032)	0,331	(1,232)	0,359	(1,544)
fr2/rpy	0,631	(1,979)	0,368	(1,606)	0,522	(1,941)
fr1/room	2,523	(7,870)	2,656	(7,565)	3,156	(13,987)
fr1/floor	0,954	(3,350)	2,207	(10,198)	0,900	(3,434)
fr1/desk	2,430	(6,058)	6,570	(29,460)	1,599	(6,486)
fr1/desk2	2,814	(7,261)	4,305	(12,869)	2,622	(11,207)
fr1/360	2,768	(6,312)	3,102	(6,528)	3,415	(8,458)
fr2/desk	1,909	(3,531)	0,599	(2,623)	0,660	(3,010)
fr1/plant	1,961	(4,656)	2,347	(11,091)	3,053	(9,037)
fr1/teddy	3,268	(16,194)	2,981	(14,363)	5,006	(23,495)
Média:	2,021		3,815		2,080	

Tabela 6.5: Precisão da odometria visual. São mostrados o RMSE do erro de pose relativa (RPE) e erro máximo para a rotação (entre parênteses) em graus por segundo.

cisão da odometria visual pode ir de 5% (fr1/room) até 82,8% (fr1/rpy) menor do que as obtidas pelo FOVIS. Em relação ao RGB-D SLAM quando este tem melhor desempenho do que o FOVIS, há uma redução na estimativa do RPE em 20% (sequência fr1/desk2). A única sequência (fora as sequências fr2) em que o KLTTW obtém piores estimativas é novamente fr1/teddy, sendo esta 9,6% maior do que o FOVIS. Assim como acontece com o erro translacional, caso sejam processadas imagens com quantidades razoáveis de features visuais (primeiras 1000 nuvens de pontos), o RMSE medido para RPE rotacional é igual a 2,285°/s, mais próximo do RMSE computado para o FOVIS de 2,046 nas mesmas imagens. Chama atenção o fato de o FOVIS ter obtido a pior média do RMSE para o RPE rotacional (3,815°/s), mesmo com o artifício implementado de computar uma estimativa inicial para a rotação separada da translação [Huang et al. 2011]. O KLTTW obteve média do RMSE igual a 2,021°/s, valor similar aos computados pelo RGB-D SLAM (2,080°/s).

As sequências com prefixo fr2 (fr2/xyz, fr2/rpy e fr2/desk) possuem estimativas de erro de translação e rotação bem superiores em relação às fornecidas pelo FOVIS e RGB-D SLAM. Acredita-se que o sensor utilizado (Asus Xtion Pro ao invés de Microsoft Kinect [Sturm et al. 2012]) ou as características da cena (mesa com objetos isolada em um galpão onde a profundidade média da cena é alta) possa necessitar de uma configuração específica dos parâmetros.

#### 6.5.2 Desempenho Computacional

As médias de tempo de processamento de cada imagem para a odometria visual com o algoritmo KLTTW são mostradas na Tabela 6.6, junto com o desvio padrão e tempo máximo gasto por nuvem de pontos RGB-D. Novamente, para fornecer uma ideia de como se

85

compara o algoritmo com outras implementações, são mostrados os mesmos dados para o FOVIS. Os números para o RGB-D SLAM são de, segundo o artigo original [Endres et al. 2012], 330 ms por imagem sem otimização e 335 ms com otimização (minimização em grafo de poses com G2O [Kümmerle et al. 2011]). Estes tempos foram obtidos utilizando a implementação em GPU do SIFT [Wu 2007] em um hardware similar ao empregado nos experimentos com KLTTW e FOVIS. Com isso, percebe-se claramente a importância da associação de dados por linha de base curta, que implementado pelo FOVIS e pelo algoritmo proposto, são mais rápidos do que o RGB-D SLAM por mais de 10 ordens de magnitude.

Tabela 6.6: Desempenho computacional da odometria visual implementada com rastreamento KLTTW. São mostrados o tempo médio, o desvio padrão e o tempo máximo (entre parênteses) para cada conjunto de dados utilizado nos experimentos.

-	Desempenho C	Computacional	em ms/imagem	
Seq.	KLTTW	-	FOVIS	
fr1/xyz	$18,577 \pm 1,956$	(28,086)	$11,307 \pm 2,025$	(18,339)
fr1/rpy	$19,195 \pm 4,227$	(84,422)	$11,309 \pm 2,166$	(22,516)
fr2/xyz	$14,982 \pm 1,300$	(21,919)	$10,710 \pm 1,345$	(16,285)
fr2/rpy	$14,526 \pm 1,120$	(19,520)	$9,951 \pm 1,609$	(17,478)
fr1/room	$13,355 \pm 1,435$	(22,320)	$11,949 \pm 2,103$	(19,863)
fr1/floor	$26,475 \pm 7,203$	(85,277)	$14,810 \pm 2,732$	(23,930)
fr1/desk	$18,528 \pm 3,955$	(77,519)	$11,462 \pm 2,535$	(22,968)
fr1/desk2	$18,971 \pm 7,401$	(106,582)	11,381 $\pm 2,366$	(19,839)
fr1/360	$14,185 \pm 4,053$	(63,195)	$12,180 \pm 2,767$	(25,568)
fr2/desk	$17,071 \pm 1,476$	(30,246)	$11,244 \pm 1,436$	(18,315)
fr1/plant	$19,618 \pm 4,972$	(102,179)	11,315 $\pm$ 2,201	(18,314)
fr1/teddy	$17,634 \pm 7,421$	(68,990)	11,768 $\pm$ 2,850	(22,830)
Média:	17,759		11,615	

O algoritmo FOVIS é cerca de 60% mais rápido do que o KLTTW. Em grande parte, isto se deve às features FAST [Rosten e Drummond 2006], propostas para fornecer uma alternativa com baixo custo computacional para algoritmos que dependem da extração de features visuais. As features propostas por Shi e Tomasi (1994) são, por construção, as mais adequadas para o fluxo óptico esparso, sendo necessário que se gaste um considerável esforço computacional (em relação ao total por imagem) para detecção destas características. Ainda, um fator que pode contribuir com um maior tempo de processamento é a qualidade das correspondências entre pontos 3D utilizadas no RANSAC. Como o RANSAC trabalha de forma combinatória, este funciona consideravelmente mais rápido caso haja menos ambiguidade (configurações singulares ou falsos casamentos).

A fração de tempo das etapas de processamento de detecção e RANSAC podem ser visualizadas no gráfico da Figura 6.7, mostrada ao longo de cada sequência usada nos experimentos. Pode-se perceber que os picos no tempo de processamento total de cada imagem se devem aos dois fatores mencionados: detecção de um número maior de fe-

atures, implicando em tempos maiores do que os típicos  $\sim 10$  ms, ou maior tempo de processamento do RANSAC para achar a solução consenso, o que resulta em tempos maiores do que os usualmente gastos ( $\sim 2$  ms) por imagem. De qualquer forma, a média de processamento obtida sugere que o algoritmo é capaz de computar poses de câmera em odometria visual a uma taxa de mais de 55 Hz, o que é suficiente para processar os 30 quadros fornecidos a cada segundo por sensores RGB-D.

## 6.6 SLAM RGB-D com Minimização em Grafo de Poses

As capacidades de SLAM do algoritmo de registro de nuvens de pontos proposto são analisadas nesta seção. O principal objeto investigado é determinar se o método de minimização em grafo de poses com marcador artificial é capaz de reduzir o erro da trajetória inicial computada pela odometria visual. Também, é quantificada a fração do erro minimizado e determinado empiricamente em que situações o método proposto não é capaz de realizar a minimização.

Uma vez que no método proposto de SLAM visual é necessário o uso de um marcador artificial, foram coletados conjuntos de dados RGB-D nos quais um marcador do tipo ARUCO [Garrido-Jurado et al. 2014] encontra-se visível na cena. Com um sensor Microsoft Kinect conduzido manualmente, foram colhidos 6 conjuntos de dados, divididos em 1. sequências do tipo objetos de mesa, nas quais o sensor visualiza um ou mais objetos de interesse de médio porte por movimentos em torno de uma mesa e 2. sequências do tipo digitalização de ambientes, nas quais o sensor captura dados de ambientes internos constituídos por cômodos de tamanho médio ( $\sim 8~\text{m}^2$ ). As sequências coletadas variam algumas características importantes para o processo, como o número de imagens em que o marcador encontra-se visível, o intervalo de reobservação do marcador artificial, a quantidade de textura presente em cada imagem e o total de imagens. A Tabela 6.7 mostra o total de imagens e o tipo de cada sequência capturada.

Tabela 6.7: Detalhes dos conjuntos de dados RGB-D empregados nos experimentos de SLAM visual. Para cada sequência, é mostrado o total de imagens e o seu tipo.

Seq.	Imagens	Tipo
nn/Cafeteira	498	Objetos de mesa
nn/Capacete	516	Objetos de mesa
nn/Objetos	318	Objetos de mesa
nn/Quarto	953	Ambientes internos
nn/Escritório	887	Ambientes internos
nn/Prateleira	800	Ambientes internos

A precisão do algoritmo é aferida com a média do erro ATE considerando a pose de câmera computada pelo sistema de realidade aumentada ARUCO como *ground truth*. Esta metodologia é adotada devido à indisponibilidade de um sistema de captura de movimentos, como utilizado nos conjuntos de dados TUM RGB-D. De qualquer forma, supõe-se

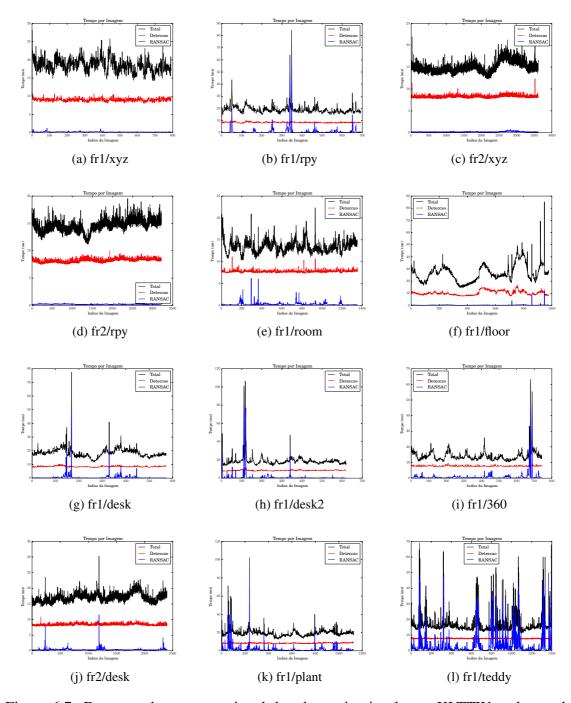


Figura 6.7: Desempenho computacional da odometria visual com KLTTW ao longo de cada sequência de imagens. Os gráficos exibem o tempo de processamento (em milissegundos) por imagem, separados em tempo total, tempo de estimação de pose com RANSAC e tempo de detecção de features. A imagem pode ser melhor visualizada com *zoom*, disponível na versão digital.

que a pose do marcador é computada com precisão, com base no fato de que a mesma

função (erro de reprojeção por homografia) é minimizada na estimação de parâmetros de calibração [Zhang 1999]. Embora seja uma medida absoluta, o *ground truth* é restrito às imagens onde o marcador é detectado.

A minimização não-linear do erro acumulado por SLAM em grafo de poses foi configurada para se encerrar em 10 iterações, com a observação de que em cada uma delas, várias iterações do algoritmo de Levenberg-Marquardt são executadas, de acordo com o fator de amortecimento.

A média do erro ATE computada para a trajetória inicial  $\mathbf{x}_{0:N}$  e também para a trajetória otimizada  $\mathbf{x}_{0:N}^*$  são exibidas na Tabela 6.8 para cada uma das sequências. Também são mostrados o número de vértices no grafo (*keyframes* selecionados), a quantidade de arestas de fechamento de laço, a melhoria entre a estimativa inicial e após minimização e o tempo total do processo de minimização.

Tabela 6.8: Minimização do erro em grafo de pose (*full* SLAM) para registro global de nuvens de pontos RGB-D. Para cada sequência, a tabela mostra, nesta ordem: o número de vértices do grafo, o número de arestas de fechamento de laço, o erro ATE médio para a trajetória inicial, o erro ATE médio para a trajetória otimizada, a porcentagem de redução do erro em comparação com o *ground truth* e o tempo total da minimização.

Seq.	Vértices	Arestas FL	ATE (Ini.)	ATE (Otim.)	Melhoria	Tempo
nn/Cafeteira	162	154	121,9 mm	6,2 mm	95,0%	8 ms
nn/Capacete	183	176	130,1 mm	6,4 mm	95,3%	10 ms
nn/Objetos	299	298	13,8 mm	2,1 mm	84,7%	25 ms
nn/Quarto	346	331	30,2 mm	1,6 mm	94,7%	23 ms
nn/Escritório	134	73	183,4 mm	3,4 mm	98,1%	10 ms
nn/Prateleira	76	45	58,1 mm	3,8 mm	93,4%	5 ms

Os resultados mostram que a minimização aplicada pode reduzir o erro acumulado na odometria visual em porcentagens que vão de 84,7% até 98,1% da média do erro ATE para os *keyframes* em que o marcador foi detectado. Isto acontece tanto em situações em que o marcador está presente em uma fração considerável das imagens, como por exemplo na sequência nn/Objetos, quanto em sequências onde as arestas de fechamento de laço são mais escassas em comparação com o total de imagens, como é o caso do resultado para nn/Escritório. A menor melhoria foi obtida para uma estimativa inicial do registro incremental cujo erro médio já era considerado pequeno. Reciprocamente, a melhor trajetória otimizada foi computada em uma sequência de imagens onde o erro ATE antes da otimização foi o maior, devido à característica da sequência na qual a câmera mapeia regiões distantes geometricamente do marcador ao longo de várias imagens.

Com o propósito de esclarecer melhor o comportamento do erro ao longo de cada sequência dos experimentos, são plotados o erro ATE para cada *keyframe* (equivalente ao erro para cada vértice) e a trajetória percorrida pelo sensor nas Figuras 6.8 e 6.9. O movimento de volta completa em torno da mesa onde se localiza os objetos visualizados realizado nas trajetórias das sequências nn/Cafeteira e nn/Capacete é notável nos respectivos gráficos. É notável também que todas as trajetórias referentes à estimação da

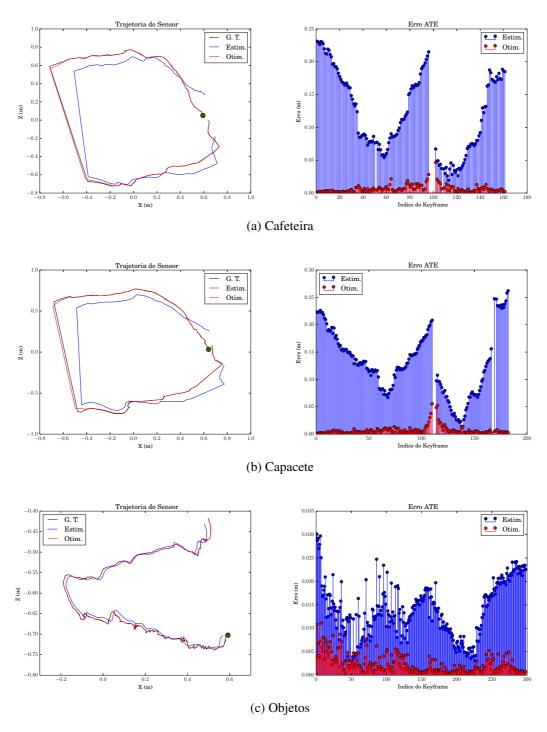


Figura 6.8: Trajetória (coluna esquerda) e erros por *keyframe* (coluna direita) para cada uma das sequências de imagens do tipo objetos de mesa utilizadas nos experimentos de SLAM visual. São mostradas a trajetória relativa ao *ground truth*, relativa à estimativa inicial (odometria visual) e relativa à estimativa otimizada (após minimização do erro acumulado). O círculo verde denota o início do percurso realizado pela câmera. O erro ATE por *keyframe* é mostrado para a estimativa inicial e estimativa otimizada. Observe que esta medida não é computada para os *keyframes* onde o marcador artificial não está visível ou não foi detectado.

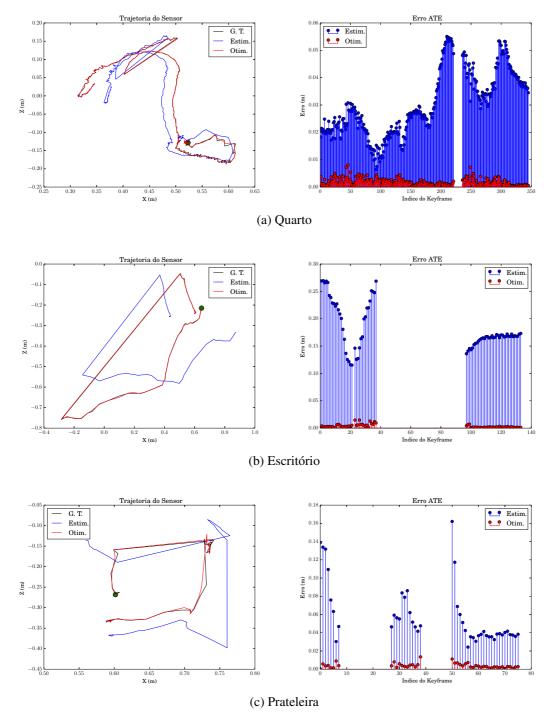


Figura 6.9: Trajetória (coluna esquerda) e erros por *keyframe* (coluna direita) para cada uma das sequências de imagens do tipo ambientes internos utilizadas nos experimentos de SLAM visual. São mostradas a trajetória relativa ao *ground truth*, relativa à estimativa inicial (odometria visual) e relativa à estimativa otimizada (após minimização do erro acumulado). O círculo verde denota o início do percurso realizado pela câmera. O erro ATE por *keyframe* é mostrado para a estimativa inicial e estimativa otimizada. Observe que esta medida não é computada para os *keyframes* onde o marcador artificial não está visível ou não foi detectado.

odometria visual nas sequências de ambientes internos encontram-se consideravelmente desalinhadas do *ground truth*, o que é drasticamente corrigido após a otimização. Este fato acontece em menor escala nas sequências de objetos de mesa.

Observe que nos gráficos referentes ao erro ATE computado em cada imagem da sequência, existem vários *keyframes* em que a medida de erro não está disponível. Obviamente, isto acontece para os *keyframes* cujo marcador não está visível ou não foi detectado. Também, picos no erro acontecem com maior frequência nestes mesmos instantes de tempo. Em todos os gráficos, o erro da trajetória inicial encontra-se limitado, não sendo maior do que ~ 30 cm. Apesar de não ser possível de se avaliar quantitativamente, os erros em *keyframes* sem arestas de fechamento de laço associadas são também reduzidos, uma vez que erros corrigidos se propagam conforme a relação de adjacência no grafo. Este fato pode ser notado nos resultados qualitativos.

O desempenho computacional do processo de *full* SLAM por minimização do erro em grafo de poses varia de 5 ms a 25 ms. Esta variação depende principalmente da densidade do grafo, ou seja, quanto mais arestas de fechamento de laço, maior será o tempo do processo. Outro fator que contribui neste ponto é a quantidade de iterações do algoritmo de Levenberg-Marquardt, que funciona restaurando estimativas anteriores caso seja necessário. Os tempos de otimização levantados apontam que uma execução incremental do algoritmo de minimização, possivelmente realizada a cada detecção de fechamento de laço, pode ser empregada, embora isto não seja investigado nesta tese.

#### 6.7 Registro RGB-D: Resultados Qualitativos

Alguns resultados qualitativos do sistema de registro de nuvens de pontos proposto nesta tese são mostrados no que segue. Na Figura 6.10, é exibida a nuvem de pontos resultante do processo de registro incremental por odometria visual e algumas imagens de aparência para a sequência de imagens fr1/floor. Nesta sequência, imagens são coletadas com a câmera levemente inclinada em direção ao chão, visualizando a textura de madeira presente no ambiente. Esta situação é facilmente encontrada em aplicações para robótica, nas quais as câmeras são frequentemente posicionadas desta maneira. A característica repetitiva do padrão de textura influencia positivamente no resultado do algoritmo, uma vez que o fluxo óptico KLT se beneficia destas situações. A vista superior da nuvem de pontos RGB-D resultante não apresenta erros notáveis visualmente. Outro resultado do registro incremental pode ser encontrado na Figura 6.11, que mostra uma perspectiva 3D do Laboratório Natalnet juntamente com amostras de imagens utilizadas no processo.

Dois exemplos de registro global obtidos com o método proposto e uso do marcador artificial de realidade aumentada são mostrados nas Figuras 6.12 e 6.13. Pensando em uma aplicação direta do método proposto para digitalização de objetos de médio porte, foram colocados sobre uma mesa uma cafeteira (sequência nn/Cafeteira) e um capacete (sequência nn/Capacete), com o sensor RGB-D capturando imagens destes objetos ao ser realizada uma trajetória circular em torno dos mesmos. Neste cenário, os objetos precisam estar dispostos isolados de outros elementos da cena que possam perturbar o processo de aquisição de um objeto de interesse, resultando assim em menos textura e consequentemente, em um número pequeno de pontos rastreados ao longo das imagens. Por conta

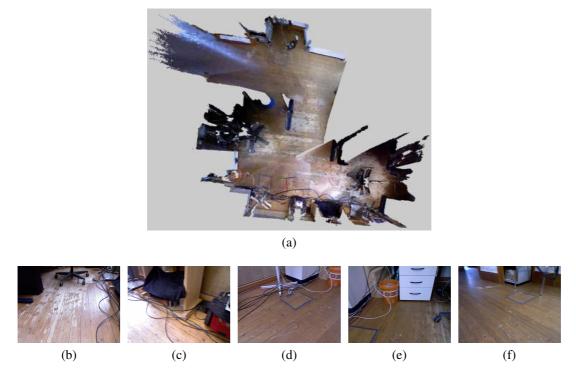


Figura 6.10: (a) Vista superior da nuvem de pontos total computada após realizar o registro incremental por odometria visual na sequência de imagens fr1/floor. (b-f) Algumas das imagens utilizadas no processo.

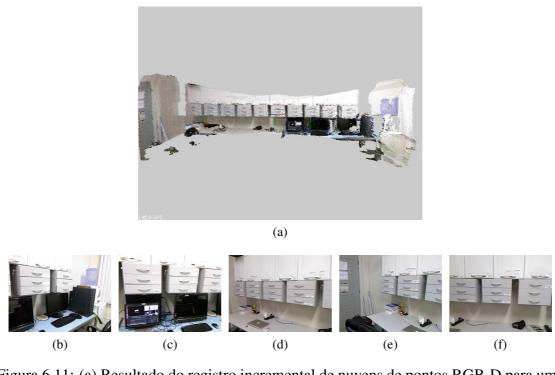


Figura 6.11: (a) Resultado do registro incremental de nuvens de pontos RGB-D para uma sequência de imagens exibindo o Laboratório Natalnet/DCA/UFRN. (b-f) Algumas das imagens utilizadas no processo.

disto, pode-se notar a diferença de qualidade visual do registro global (Figuras 6.12b e 6.13b) em relação ao registro computado pela odometria visual (Figuras 6.12a e 6.13a), evidenciando a importância da minimização do erro. Vídeos demonstrando o método em funcionamento na referidas sequências de imagens estão disponíveis na internet<sup>3</sup>.

#### 6.8 Discussão e Limitações

Os resultados apresentados para os experimentos em registro de nuvens de pontos RGB-D com o fluxo óptico esparso deixam claro que explorar esta estratégia para esta aplicação é completamente possível, principalmente por causa de características apresentadas, como um baixo número de falsas correspondências, distribuição uniforme nas posições 2D de features e baixo custo computacional. Mais ainda, aplicações diversas como mapeamento 3D para robótica e entretenimento, que possuem requisitos estritos de processamento a 30 Hz, não só são factíveis como também podem ser implementadas com precisão e desempenho computacional similar ao estado da arte. Salienta-se que a comparação realizada com o FOVIS e RGB-D SLAM não é de forma alguma exaustiva, uma vez que existem diversos outros trabalhos que podem ser classificados por diversas taxonomias (como mostrado na Seção 3.4 e em levantamentos recentes [Fang e Scherer 2014]) e por isso, podem se sair melhores ou piores em determinadas situações ou aplicações.

O rastreamento por linha de base curta requer que a distância dos centros de projeção entre imagens sucessivas seja limitada. Consequentemente, o método falharia em situações em que apenas uma subamostra das imagens coletadas a 30 Hz fosse processada com o sensor em movimento. Tanto o fluxo óptico quanto o rastreamento por correlação entre janela de pixels (como empregado pelo FOVIS) não seriam capazes de determinar o rastreamento das features. Por usar rastreamento por detecção, o RGB-D SLAM não possui este limite teórico, mas pode ter o rastreamento afetado por um baixo número de correspondências verdadeiras. Esta limitação também atinge algoritmos de registro de nuvens de pontos que se baseiam no ICP [Whelan et al. 2013] e também as abordagens densas para sensores RGB-D que registram nuvens de pontos de forma direta [Steinbruecker et al. 2011, Kerl et al. 2013*b*, Kerl et al. 2013*a*].

O uso do marcador artificial para detecção de fechamento de laços no algoritmo de minimização de erro por *full* SLAM é capaz de reduzir drasticamente o erro acumulado durante o registro incremental. Isto acontece mesmo com as arestas de fechamento de laço adicionando sempre uma restrição ao vértice correspondente ao inicial da trajetória. A minimização empregada mostra-se capaz de reduzir o erro da odometria visual supondo que este não cresce indefinidamente ao longo do processo. Do contrário, seria necessário uma forma de minimizar o erro por full SLAM de maneira incremental, tão logo uma aresta de fechamento de laço seja detectada, na qual possivelmente vários marcadores artificiais são empregados. Qualitativamente, é avaliado que a redução de erro é consideravelmente maior na região da cena próxima ao marcador artificial, o que pode apontar para o fato de que o mesmo deve ser posicionado em locais onde se deseja uma maior precisão.

<sup>&</sup>lt;sup>3</sup>https://www.youtube.com/user/brunomfs/videos

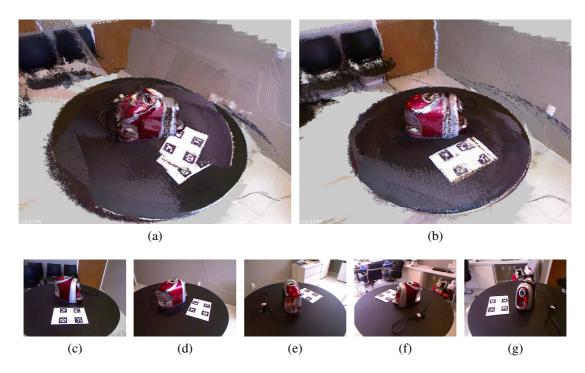


Figura 6.12: Registro de nuvens de pontos para a sequência nn/Cafeteira. Em (a) é mostrado o resultado do registro incremental, enquanto o registro global (após minimização do erro acumulado) é mostrado em (b). (c-g) Algumas das imagens utilizadas no processo.

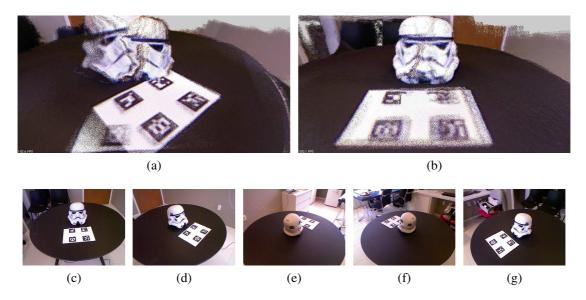


Figura 6.13: Registro de nuvens de pontos para a sequência nn/Capacete. Em (a) é mostrado o resultado do registro incremental, enquanto o registro global (após minimização do erro acumulado) é mostrado em (b). (c-g) Algumas das imagens utilizadas no processo.

95

O escopo do método apresentado se restringe ao mapeamento de ambientes estáticos e com vasta disponibilidade de features visuais com dados de profundidade associados. Possíveis estratégias para aplicar o sistema em ambientes dinâmicos podem contribuir com uma versão de uso mais geral do método proposto.

# Capítulo 7

### Conclusão

Nesta tese, é proposto um método para registro [Gomes et al. 2013] de nuvens de pontos capturadas por sensores RGB-D baseado em features visuais que obtém consistência global no registro utilizando marcadores na cena mapeada.

Especificamente, um algoritmo de registro incremental [Silva e Gonçalves 2014a, Silva e Gonçalves 2014b, Silva e Gonçalves 2015c] é proposto com o rastreamento de features visuais por fluxo óptico em conjunto com uma estratégia de alto nível em que são utilizadas janelas de rastreamento. Novas features são adicionadas ao rastreador em cada nova imagem, desde que elas não estejam inseridas dentro da janela de rastreamento de nenhuma feature rastreada. Este mecanismo, denominado KLTTW, resulta em características visuais espalhadas pela imagem de uma maneira mais uniforme do que se for utilizado o algoritmo KLT [Bouguet 2000]. A partir do rastreamento, correspondências entre pontos 3D pertencentes a duas nuvens RGB-D consecutivas são obtidas, de forma que a transformação de registro pode ser computada de maneira rápida e precisa.

Com o uso de um marcador artificial inserido manualmente na cena em uma posição e orientação arbitrária, o erro acumulado no registro incremental pode ser otimizado de forma global [Silva e Gonçalves 2015b]. O problema é modelado por uma abordagem de SLAM visual baseado em minimização de erro em grafo de poses, onde cada vértice do grafo é formado pela pose de keyframes (imagens chaves) extraídas automaticamente durante o mapeamento. Os vértices são conectados por arestas denotando a transformação de registro que relaciona as poses do sensor no instante de tempo da captura dos dados. O marcador artificial é utilizado para a detecção de locais mapeados previamente, sendo adicionadas restrições em forma de arestas de fechamento de laço ao grafo representando o mapa sempre que o mesmo é detectado. Após todas as imagens RGB-D da sequência serem processadas, a otimização do grafo é computada, solucionando o problema de full SLAM com os parâmetros de transformação mais prováveis dadas as observações coletadas ao longo de toda a trajetória do sensor. As nuvens de pontos relacionadas aos sistemas de referência de cada keyframe são então transformadas, de modo a refletir os resultados da otimização e resultar no registro de todas as nuvens processadas com consistência global.

Resultados experimentais obtidos com o uso de conjuntos de dados e *benchmark* presentes na literatura [Sturm et al. 2012] validam esta tese. Medidas estatísticas robustas são computadas sobre métricas de desempenho que avaliam o quanto de erro translacional e rotacional são acumulados pelo sistema, evidenciando que o método de registro

incremental proposto tem precisão comparável a sistemas de odometria visual [Huang et al. 2011] e de SLAM com sensores RGB-D [Endres et al. 2012] que fazem parte do estado da arte em sistemas esparsos de registro de nuvens de pontos. O sistema proposto tem desempenho computacional suficiente para realizar o processo considerando imagens adquiridas na taxa de vídeo (30 Hz).

Tanto as trajetórias finais computadas pela otimização em grafo de poses em relação à trajetória inicial quanto o erro absoluto computado em relação ao marcador evidenciam que o registro é realizado com consistência global. Apesar de somente ser aplicado após uma trajetória inicial ser estimada, o desempenho computacional do algoritmo de otimização indica que o processo pode ser implementado de forma online, tendo em vista o tempo máximo de minimização obtido nos experimentos ( $\sim 30 \text{ ms}$ ).

O algoritmo pode ser diretamente aplicado em odometria visual e SLAM visual, com o resultado do registro sendo a entrada para representações 3D mais complexas [Marton et al. 2009, Foley et al. 1990] ou visando mapeamento de ambientes para robótica [Souza e Gonçalves 2015, Hornung et al. 2013]. Dentre as vantagens oferecidas pelo método, está o fato da computação ser realizada totalmente em CPU, não necessitando de hardwares especializados como GPUs. Ainda, a implementação do método pode ser facilitada caso sejam utilizados componentes de software disponíveis em bibliotecas como OpenCV, Point Cloud Library, G2O e ARUCO.

Como contribuição secundária [Silva e Gonçalves 2015a], é mostrado experimentalmente que o rastreamento de imagem para imagem computado pelo fluxo óptico oferece um número drasticamente menor de falsas correspondências em relação ao rastreamento empregado ao detectar e extrair descritores SURF. Esta metodologia de avaliar algoritmos de rastreamento está presente na literatura com o uso de conjuntos de dados planares [Gauglitz et al. 2011], não tendo sido encontrados trabalhos comparativos considerando um conjunto de dados relacionados a SLAM visual.

Trabalhos futuros estão previstos para estender as capacidades do método proposto nesta tese. O primeiro deles pretende apontar qual é o descritor de imagem mais adequado às features visuais de Shi-Tomasi. O objetivo é elaborar uma estratégia de rastreamento híbrido [Choi et al. 2008], na qual as features são rastreadas de imagem para imagem com fluxo óptico e fechamentos de laço são detectados com base em busca de imagens por descritores. Os descritores SIFT e SURF devem ser avaliados, assim como descritores binários (e.g. BRIEF [Calonder et al. 2010] e ORB [Rublee et al. 2011]).

Versões online do algoritmo serão propostas futuramente, fazendo uso tanto de marcadores artificiais quanto de descritores de features, para realizar a otimização à medida em que fechamentos de laço são detectados. Para este fim, mecanismos de minimização do erro devem ser executados em grafos formados por submapas locais (vértices próximos) e também em grafos formados por *keyframes* resultantes da otimização destes submapas ao longo de toda a trajetória do sensor, de forma semelhante à realizada na literatura de SLAM monocular [Strasdat et al. 2011, Klein e Murray 2007].

# Referências Bibliográficas

- Agarwal, S., N. Snavely, I. Simon, S.M. Seitz e R. Szeliski (2009), Building rome in a day, *em* 'IEEE International Conference on Computer Vision (ICCV)', IEEE, pp. 72–79.
- Akbarzadeh, A., J.-M. Frahm, P. Mordohai, B. Clipp, C. Engels, D. Gallup, P. Merrell, M. Phelps, S. Sinha, B. Talton, L. Wang, Q. Yang, H. Stewenius, R. Yang, G. Welch, H. Towles, D. Nistér e M. Pollefeys (2006), Towards urban 3D reconstruction from video, *em* 'International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT)', IEEE, pp. 1–8.
- Aldoma, A., Z. Marton, F. Tombari, W. Wohlkinger, C. Potthast, B. Zeisl, R.B. Rusu, S. Gedikli e M. Vincze (2012), 'Tutorial: Point cloud library: Three-dimensional object recognition and 6 DOF pose estimation', *IEEE Robotics and Automation Magazine* **19**(3), 80–91.
- Anguelov, Dragomir, Carole Dulong, Daniel Filip, Christian Frueh, Stephane Lafon, Richard Lyon, Abhijit Ogale, Luc Vincent e Josh Weaver (2010), 'Google street view: Capturing the world at street level', *Computer* **43**(6), 32–38.
- Bailey, Tim e H. Durrant-Whyte (2006), 'Simultaneous localization and mapping (SLAM): part II', *IEEE Robotics and Automation Magazine* **13**(3), 108–117.
- Bay, Herbert, Andreas Ess, Tinne Tuytelaars e Luc Van Gool (2008), 'Speeded-up robust features (SURF)', *Computer Vision and Image Understanding* **110**(3), 346–359.
- Besl, P.J. e Neil D. McKay (1992), 'A method for registration of 3-D shapes', *IEEE Transactions on Pattern Analysis and Machine Intelligence* **14**(2), 239–256.
- Bondy, J. A. e U. S. R. Murty (1976), *Graph Theory With Applications*, Elsevier, New York.
- Bouguet, Jean-Yves (2000), 'Pyramidal implementation of the lucas kanade feature tracker', *Intel Corporation, Microprocessor Research Labs*.
- Bradski, Gary Rost e Adrian Kaehler (2008), *Learning OpenCV*, 1<sup>a</sup> edição, O'Reilly Media, Inc.
- Bruhn, Andrés, Joachim Weickert e Christoph Schnörr (2005), 'Lucas/kanade meets horn/schunck: Combining local and global optic flow methods', *International Journal of Computer Vision* **61**(3).

- Calonder, M., V. Lepetit e P. Fua (2008), Keypoint signatures for fast learning and recognition, *em* 'European Conference on Computer Vision (ECCV)', Springer Berlin Heidelberg, pp. 58–71.
- Calonder, Michael, Vincent Lepetit, Christoph Strecha e Pascal Fua (2010), BRIEF: Binary robust independent elementary features, *em* 'European Conference on Computer Vision (ECCV)', Springer Berlin Heidelberg, pp. 778–792.
- Choi, Changhyun, Seung-Min Baek e Sukhan Lee (2008), Real-time 3D object pose estimation and tracking for natural landmark based visual servo, *em* 'IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)', IEEE, pp. 3983–3989.
- Cummins, Mark e Paul Newman (2011), 'Appearance-only SLAM at large scale with FAB-MAP 2.0', *International Journal of Robotics Research* **30**(9), 1100–1123.
- Davis, T. (2006), *Direct Methods for Sparse Linear Systems*, Society for Industrial and Applied Mathematics.
- Davison, Andrew J. (2003), Real-time simultaneous localisation and mapping with a single camera, *em* 'IEEE International Conference on Computer Vision (ICCV)', IEEE, p. 1403.
- DiVerdi, S., J. Wither e T. Hollerei (2008), Envisor: Online environment map construction for mixed reality, *em* 'IEEE Virtual Reality Conference (VR)', IEEE, pp. 19–26.
- DiVerdi, S. e T. Hollerer (2008), 'Heads up and camera down: A vision-based tracking modality for mobile mixed reality', *IEEE Transactions on Visualization and Computer Graphics* **14**(3), 500–512.
- Dryanovski, I, R.G. Valenti e Jizhong Xiao (2013), Fast visual odometry and mapping from RGB-D data, *em* 'IEEE International Conference on Robotics and Automation (ICRA)', IEEE, pp. 2305–2310.
- Duan, Liya, Tao Guan e Bo Yang (2009), 'Registration combining wide and narrow baseline feature tracking techniques for markerless AR systems', *Sensors* **9**(12), 10097.
- Durrant-Whyte, H. e Tim Bailey (2006), 'Simultaneous localization and mapping: part I', *IEEE Robotics and Automation Magazine* **13**(2), 99–110.
- Endres, F., J. Hess, N. Engelhard, J. Sturm, D. Cremers e W. Burgard (2012), An evaluation of the RGB-D SLAM system, *em* 'IEEE International Conference on Robotics and Automation (ICRA)', IEEE, pp. 1691–1696.
- Fang, Zheng e S. Scherer (2014), Experimental study of odometry estimation methods using RGB-D cameras, *em* 'IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2014)', IEEE, pp. 680–687.

- Fischler, M.A. e R.C. Bolles (1981), 'Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography', *Communications of the ACM* pp. 381–395.
- Foley, James D., Andries van Dam, Steven K. Feiner e John F. Hughes (1990), *Computer Graphics: Principles and Practice (2nd Ed.)*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.
- Forsyth, David A. e Jean Ponce (2002), *Computer Vision: A Modern Approach*, Prentice Hall Professional Technical Reference.
- Fraundorfer, F. e D. Scaramuzza (2012), 'Visual odometry: Part II: Matching, robustness, optimization, and applications', *IEEE Robotics and Automation Magazine* **19**(2), 78–90.
- Galvez-Lopez, D. e J.D. Tardos (2011), Real-time loop detection with bags of binary words, *em* 'IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)', IEEE, pp. 51–58.
- Garrido-Jurado, S., R. Muñoz-Salinas, F.J. Madrid-Cuevas e M.J. Marín-Jiménez (2014), 'Automatic generation and detection of highly reliable fiducial markers under occlusion', *Pattern Recognition* **47**(6), 2280 – 2292.
- Gauglitz, Steffen, Tobias Höllerer e Matthew Turk (2011), 'Evaluation of interest point detectors and feature descriptors for visual tracking', *International Journal of Computer Vision* **94**(3), 335–360.
- Gomes, R.B., B.M.F. Silva, L.K.M. Rocha, R.V. Aroca, L.C.P.R. Velho e L.M.G. Gonçalves (2013), 'Efficient 3D object recognition using foveated point clouds', *Computers & Graphics* **37**(5), 496–508.
- Goncalves, L., E. Di Bernardo, D. Benson, M. Svedman, J. Ostrowski, N. Karlsson e P. Pirjanian (2005), A visual front-end for simultaneous localization and mapping, *em* 'IEEE International Conference on Robotics and Automation (ICRA)', IEEE, pp. 44–49.
- Google (2015), 'Google Street View', www.google.com/streetview. Acessado em 20 de maio de 2015.
- Grisetti, G., C. Stachniss e W. Burgard (2009), 'Nonlinear constraint network optimization for efficient map learning', *IEEE Transactions on Intelligent Transportation Systems* **10**(3), 428–439.
- Grisetti, G., R. Kümmerle, C. Stachniss e W. Burgard (2010), 'A tutorial on graph-based SLAM', *IEEE Intelligent Transportation Systems Magazine* **2**(4), 31–43.
- Guo, Yulan, M. Bennamoun, F. Sohel, Min Lu e Jianwei Wan (2014), '3d object recognition in cluttered scenes with local surface features: A survey', *IEEE Transactions on Pattern Analysis and Machine Intelligence* **36**(11), 2270–2287.

- Gutmann, J.-S. e K. Konolige (1999), Incremental mapping of large cyclic environments, *em* 'IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA)', IEEE, pp. 318–325.
- Han, Jungong, Ling Shao, Dong Xu e J. Shotton (2013), 'Enhanced computer vision with microsoft kinect sensor: A review', *IEEE Transactions on Cybernetics* **43**(5), 1318–1334.
- Harris, C. e M. Stephens (1988), A combined corner and edge detector, *em* 'Proceedings of the 4th Alvey Vision Conference', pp. 147–151.
- Hartley, R. I. e A. Zisserman (2004), *Multiple View Geometry in Computer Vision*, 2<sup>a</sup> edição, Cambridge University Press.
- Hawk-Eye (2015), 'Hawk-Eye Innovations', http://www.hawkeyeinnovations.co. uk/. Acessado em 20 de maio de 2015.
- Henry, P., M. Krainin, E. Herbst, X. Ren e D. Fox (2010), RGB-D mapping: Using depth cameras for dense 3D modeling of indoor environments, *em* 'International Symposium on Experimental Robotics (ISER)'.
- Henry, P., M. Krainin, E. Herbst, X. Ren e D. Fox (2012), 'RGB-D mapping: Using kinect-style depth cameras for dense 3D modeling of indoor environments', *International Journal of Robotics Research* **31**(5), 647–663.
- Horn, Berthold K. P. (1987), 'Closed-form solution of absolute orientation using unit quaternions', *Journal of the Optical Society of America A* **4**(4), 629–642.
- Hornung, Armin, KaiM. Wurm, Maren Bennewitz, Cyrill Stachniss e Wolfram Burgard (2013), 'OctoMap: An efficient probabilistic 3D mapping framework based on octrees', *Autonomous Robots* **34**(3), 189–206.
- Howard, A. (2008), Real-time stereo visual odometry for autonomous ground vehicles, *em* 'IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)', IEEE, pp. 3946–3952.
- Hu, G., S. Huang, L. Zhao, A. Alempijevic e G. Dissanayake (2012), A robust RGB-D SLAM algorithm, *em* 'IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)', IEEE, pp. 1714–1719.
- Huang, Albert S., Abraham Bachrach, Peter Henry, Michael Krainin, Daniel Maturana, Dieter Fox e Nicholas Roy (2011), Visual odometry and mapping for autonomous flight using an RGB-D camera, *em* 'Int. Symposium on Robotics Research (ISRR)'.
- Irschara, A., C. Hoppe, H. Bischof e S. Kluckner (2011), Efficient structure from motion with weak position and orientation priors, *em* 'IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)', IEEE, pp. 21–28.

- Karlsson, N., E. Di Bernardo, J. Ostrowski, L. Goncalves, P. Pirjanian e M.E. Munich (2005), The vSLAM algorithm for robust localization and mapping, *em* 'IEEE International Conference on Robotics and Automation (ICRA)', IEEE, pp. 24–29.
- Ke, Yan e R. Sukthankar (2004), PCA-SIFT: a more distinctive representation for local image descriptors, *em* 'IEEE Conference on Computer Vision and Pattern Recognition (CVPR)', Vol. 2, IEEE, pp. 506–513.
- Kerl, C., J. Sturm e D. Cremers (2013a), Dense visual slam for RGB-D cameras, *em* 'IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)', IEEE, pp. 2100–2106.
- Kerl, C., J. Sturm e D. Cremers (2013b), Robust odometry estimation for RGB-D cameras, *em* 'IEEE International Conference on Robotics and Automation (ICRA)', IEEE, pp. 3748–3754.
- Khoshelham, Kourosh e Sander Oude Elberink (2012), 'Accuracy and resolution of kinect depth data for indoor mapping applications', *Sensors* **12**(2), 1437–1454.
- Klein, Georg e David Murray (2007), Parallel tracking and mapping for small AR workspaces, *em* 'IEEE International Symposium on Mixed and Augmented Reality (ISMAR)', IEEE, pp. 1–10.
- Klippenstein, J. e Hong Zhang (2007), Quantitative evaluation of feature extractors for visual SLAM, *em* 'Conference on Computer and Robot Vision (CRV)', IEEE, pp. 157–164.
- Konolige, K. e M. Agrawal (2007), Frame-frame matching for realtime consistent visual mapping, *em* 'IEEE International Conference on Robotics and Automation (ICRA)', IEEE, pp. 2803–2810.
- Konolige, K. e M. Agrawal (2008), 'Frameslam: From bundle adjustment to real-time visual mapping', *IEEE Transactions on Robotics* **24**(5), 1066–1077.
- Konolige, K. e P. Mihelich (2011), 'Technical description of kinect calibration', http://wiki.ros.org/kinect\_calibration/technical. Acessado em 20 de maio de 2015.
- Kümmerle, R., G. Grisetti, H. Strasdat, K. Konolige e W. Burgard (2011), G2O: A general framework for graph optimization, *em* 'IEEE International Conference on Robotics and Automation (ICRA)', IEEE, pp. 3607–3613.
- Kümmerle, Rainer, Bastian Steder, Christian Dornhege, Michael Ruhnke, Giorgio Grisetti, Cyrill Stachniss e Alexander Kleiner (2009), 'On measuring the accuracy of SLAM algorithms', *Autonomous Robots* **27**(4), 387–407.
- Lee, Taehee e T. Hollerer (2008), Hybrid feature tracking and user interaction for markerless augmented reality, *em* 'IEEE Virtual Reality Conference (VR)', IEEE, pp. 145 –152.

- Lepetit, Vincent e Pascal Fua (2005), 'Monocular model-based 3D tracking of rigid objects', Foundations and Trends on Computer Graphics and Vision 1(1), 1–89.
- Lima, Elon Lages (2011), *Variedades Diferenciáveis*, IMPA. Disponível em http://www.impa.br/opencms/pt/biblioteca/pm/PM\_25.pdf.
- Liu, Yang e Hong Zhang (2012), Indexing visual features: Real-time loop closure detection using a tree structure, *em* 'IEEE International Conference on Robotics and Automation (ICRA)', IEEE, pp. 3613–3618.
- Lourakis, Manolis I. A. e Antonis A. Argyros (2009), 'SBA: A software package for generic sparse bundle adjustment', *ACM Trans. Math. Softw.* **36**(1), 1–30.
- Lowe, David G. (2004), 'Distinctive image features from scale-invariant keypoints', *International Journal of Computer Vision* **60**(2), 91–110.
- Lu, F. e E. Milios (1997), 'Globally consistent range scan alignment for environment mapping', *Autonomous Robots* **4**(4), 333–349.
- Lucas, Bruce D. e Takeo Kanade (1981), An iterative image registration technique with an application to stereo vision, *em* 'International Joint Conference on Artificial Intelligence (IJCAI)', pp. 674–679.
- Marquardt, Donald W. (1963), 'An algorithm for least-squares estimation of nonlinear parameters', *Journal of the Society for Industrial and Applied Mathematics* **11**(2), 431–441.
- Marton, Z.C., R.B. Rusu e M. Beetz (2009), On fast surface reconstruction methods for large and noisy point clouds, *em* 'IEEE International Conference on Robotics and Automation (ICRA)', IEEE, pp. 3218–3223.
- Metaio (2015), 'Metaio Augmented Reality', https://www.metaio.com/. Acessado em 20 de maio de 2015.
- Mihalyi, R.-G., K. Pathak, N. Vaskevicius e A. Birk (2013), Uncertainty estimation of ar-marker poses for graph-SLAM optimization in 3D object model generation with RGBD data, *em* 'IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)', IEEE, pp. 1807–1813.
- Mikolajczyk, K. e C. Schmid (2005), 'A performance evaluation of local descriptors', *IEEE Transactions on Pattern Analysis and Machine Intelligence* **27**(10), 1615–1630.
- Montemerlo, Michael, Sebastian Thrun, Daphne Roller e Ben Wegbreit (2003), FastS-LAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges, *em* 'International Joint Conference on Artificial Intelligence (IJCAI)', Morgan Kaufmann Publishers Inc., pp. 1151–1156.

- Moravec, Hans P. (1977), Towards automatic visual obstacle avoidance, *em* 'International Joint Conference on Artificial Intelligence (IJCAI)'.
- Moreno, F. A., J. L. Blanco e J. González (2007), An efficient closed-form solution to probabilistic 6D visual odometry for a stereo camera, *em* 'International Conference on Advanced Concepts for Intelligent Vision Systems (ACIVS)', Springer Berlin Heidelberg, pp. 932–942.
- Mouragnon, E., Maxime Lhuillier, M. Dhome, F. Dekeyser e P. Sayd (2006), Real time localization and 3D reconstruction, *em* 'IEEE Conference on Computer Vision and Pattern Recognition (CVPR)', IEEE, pp. 363–370.
- MSDN (2015), 'Kinect for Windows Sensor Components and Specifications', https://msdn.microsoft.com. Acessado em 20 de maio de 2015.
- Muja, Marius e David G. Lowe (2014), 'Scalable nearest neighbor algorithms for high dimensional data', *IEEE Transactions on Pattern Analysis and Machine Intelligence* **36**.
- Nascimento, E.R., G.L. Oliveira, M.F.M. Campos, A.W. Vieira e W.R. Schwartz (2012), BRAND: A robust appearance and depth descriptor for RGB-D images, *em* 'IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)', IEEE, pp. 1720–1726.
- Newcombe, Richard A., Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J. Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges e Andrew Fitzgibbon (2011), KinectFusion: Real-time dense surface mapping and tracking, *em* 'IEEE International Symposium on Mixed and Augmented Reality (ISMAR)', IEEE, pp. 127–136.
- Newman, P., D. Cole e K. Ho (2006), Outdoor SLAM using visual appearance and laser ranging, *em* 'IEEE International Conference on Robotics and Automation (ICRA)', IEEE, pp. 1180–1187.
- Nistér, D. (2003a), An efficient solution to the five-point relative pose problem, *em* 'IEEE Conference on Computer Vision and Pattern Recognition (CVPR)', Vol. 2, IEEE, pp. 195–202.
- Nistér, D. (2003*b*), Preemptive RANSAC for live structure and motion estimation, *em* 'IEEE International Conference on Computer Vision (ICCV)', IEEE, pp. 199–206 vol.1.
- Nistér, D. e H. Stewenius (2006), Scalable recognition with a vocabulary tree, *em* 'IEEE Conference on Computer Vision and Pattern Recognition (CVPR)', Vol. 2, IEEE, pp. 2161–2168.
- Nistér, David, Oleg Naroditsky e James Bergen (2004), Visual odometry, *em* 'IEEE Conference on Computer Vision and Pattern Recognition (CVPR)', IEEE, pp. 652–659.

- Osteen, P.R., J.L. Owens e C.C. Kessens (2012), Online egomotion estimation of RGB-D sensors using spherical harmonics, *em* 'IEEE International Conference on Robotics and Automation (ICRA)', IEEE, pp. 1679–1684.
- Paton, M. e J. Kosecka (2012), Adaptive RGB-D localization, *em* 'Conference on Computer and Robot Vision (CRV)', IEEE, pp. 24–31.
- Pollefeys, Marc, Luc Van Gool, Maarten Vergauwen, Frank Verbiest, Kurt Cornelis, Jan Tops e Reinhard Koch (2004), 'Visual modeling with a hand-held camera', *International Journal of Computer Vision* **59**(3), 207–232.
- Pomerleau, François, Francis Colas, Roland Siegwart e Stéphane Magnenat (2013), 'Comparing ICP variants on real-world data sets', *Autonomous Robots* **34**(3), 133–148.
- Rosten, E. e T. Drummond (2006), Machine learning for high-speed corner detection, *em* 'European Conference on Computer Vision (ECCV)', Springer Berlin Heidelberg, pp. 430–443.
- Rublee, E., V. Rabaud, K. Konolige e G. Bradski (2011), ORB: An efficient alternative to SIFT or SURF, *em* 'IEEE International Conference on Computer Vision (ICCV)', IEEE, pp. 2564–2571.
- Rusu, R.B. e S. Cousins (2011), 3D is here: Point cloud library (PCL), *em* 'IEEE International Conference on Robotics and Automation (ICRA)', IEEE, pp. 1–4.
- Scaramuzza, D. e F. Fraundorfer (2011), 'Visual odometry [tutorial]', *IEEE Robotics and Automation Magazine* **18**(4), 80–92.
- Se, S., D. Lowe e J. Little (2001), Vision-based mobile robot localization and mapping using scale-invariant features, *em* 'IEEE International Conference on Robotics and Automation (ICRA)', Vol. 2, IEEE, pp. 2051–2058 vol.2.
- Shi, J. e C. Tomasi (1994), Good features to track, *em* 'IEEE Conference on Computer Vision and Pattern Recognition (CVPR)', IEEE, pp. 593–600.
- Siegwart, Roland e Illah R. Nourbakhsh (2004), *Introduction to Autonomous Mobile Robots*, Bradford Company, Scituate, MA, USA.
- Silva, B.M.F. da e L.M.G. Gonçalves (2014*a*), A fast feature tracking algorithm for visual odometry and mapping based on RGB-D sensors, *em* 'SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)', IEEE, pp. 227–234.
- Silva, B.M.F. da e L.M.G. Gonçalves (2014*b*), A fast visual odometry and mapping system for RGB-D cameras, *em* 'Latin American Robotics Symposium (LARS)', IEEE, pp. 55–60.

- Silva, B.M.F. da e L.M.G. Gonçalves (2015*a*), 'Evaluation of visual tracking for pairwise registration of RGB-D point clouds'. Technical Report, July 2015 NatalNet Associated Laboratory (UFRN).
- Silva, B.M.F. da e L.M.G. Gonçalves (2015*b*), 'Real-time registration of RGB-D point clouds with artificial markers'. Technical Report, July 2015 NatalNet Associated Laboratory (UFRN).
- Silva, B.M.F. da e L.M.G. Gonçalves (2015c), 'Visual odometry and mapping for indoor environments using RGB-D cameras', *Communications in Computer and Information Science*. Aceito para publicação.
- Snavely, Noah, Steven M. Seitz e Richard Szeliski (2008), 'Modeling the world from internet photo collections', *International Journal of Computer Vision* **80**(2), 189–210.
- Souza, Anderson e Luiz M. G. Gonçalves (2015), 'Occupancy-elevation grid: an alternative approach for robotic mapping and navigation', *Robotica* **FirstView**, 1–18.
- Steinbruecker, F., J. Sturm e D. Cremers (2011), Real-time visual odometry from dense RGB-D images, *em* 'International Conference on Computer Vision Workshop on Live Dense Reconstruction with Moving Cameras'.
- Strasdat, H., A.J. Davison, J.M.M. Montiel e K. Konolige (2011), Double window optimisation for constant time visual SLAM, *em* 'IEEE International Conference on Computer Vision (ICCV)', IEEE, pp. 2352–2359.
- Strasdat, H., J.M.M. Montiel e A.J. Davison (2010), Real-time monocular SLAM: Why filter?, *em* 'IEEE International Conference on Robotics and Automation (ICRA)', IEEE, pp. 2657–2664.
- Stückler, J. e S. Behnke (2012), Integrating depth and color cues for dense multiresolution scene mapping using RGB-D cameras, *em* 'IEEE International Conference on Multisensor Fusion and Information Integration (MFI)', IEEE.
- Sturm, J., N. Engelhard, F. Endres, W. Burgard e D. Cremers (2012), A benchmark for the evaluation of RGB-D SLAM systems, *em* 'IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)', IEEE, pp. 573–580.
- Szeliski, Richard (2010), Computer Vision: Algorithms and Applications, Springer-Verlag New York, Inc., New York, NY, USA.
- Tamura, Y., M. Suzuki, A. Ishii e Y. Kuroda (2009), Visual odometry with effective feature sampling for untextured outdoor environment, *em* 'IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)', IEEE, pp. 3492–3497.
- Thrun, Sebastian e John J. Leonard (2008), Simultaneous localization and mapping, *em* B.Siciliano e O.Khatib, eds., 'Springer Handbook of Robotics', Springer Berlin Heidelberg, pp. 871–889.

- Thrun, Sebastian e Michael Montemerlo (2006), 'The Graph SLAM algorithm with applications to large-scale mapping of urban structures', *International Journal of Robotics Research* **25**(5-6), 403–429.
- Thrun, Sebastian, Wolfram Burgard e Dieter Fox (2005), *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*, The MIT Press.
- Tomono, M. (2010), 3D localization based on visual odometry and landmark recognition using image edge points, *em* 'IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)', IEEE, pp. 5953–5959.
- Triggs, B., P. Mclauchlan, R. Hartley e A. Fitzgibbon (2000), Bundle adjustment a modern synthesis, *em* 'Vision Algorithms: Theory and Practice, LNCS', Springer Berlin Heidelberg, pp. 298–375.
- Trucco, Emanuele e Alessandro Verri (1998), *Introductory Techniques for 3D Computer Vision*, Prentice Hall PTR, Upper Saddle River, NJ, USA.
- Umeyama, Shinji (1991), 'Least-squares estimation of transformation parameters between two point patterns', *IEEE Transactions on Pattern Analysis and Machine Intelligence* **13**(4), 376–380.
- Whelan, T., H. Johannsson, M. Kaess, J.J. Leonard e J. McDonald (2013), Robust real-time visual odometry for dense RGB-D mapping, *em* 'IEEE International Conference on Robotics and Automation (ICRA)', IEEE, pp. 5724–5731.
- Whelan, T., M. Kaess, M.F. Fallon, H. Johannsson, J.J. Leonard e J.B. McDonald (2012), Kintinuous: Spatially extended KinectFusion, *em* 'Robotics Science and Systems Conference Workshop on RGB-D: Advanced Reasoning with Depth Cameras', Sydney, Australia.
- Wu, Changchang (2007), 'SiftGPU: A GPU implementation of scale invariant feature transform (SIFT)', http://cs.unc.edu/~ccwu/siftgpu.
- Zhang, Zhengyou (1999), Flexible camera calibration by viewing a plane from unknown orientations, *em* 'IEEE International Conference on Computer Vision (ICCV)', Vol. 1, IEEE, pp. 666–673 vol.1.
- Zhao, Liang, Shoudong Huang, Lei Yan, J.J. Wang, G. Hu e G. Dissanayake (2010), Large-scale monocular SLAM by local bundle adjustment and map joining, *em* 'International Conference on Control Automation Robotics Vision (ICARCV)', IEEE, pp. 431–436.