



UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
CENTRO DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA DE COMPUTAÇÃO E
AUTOMAÇÃO



Análise do RANSAC sobre algoritmo de odometria visual com imagens estéreo

Vanessa Dantas de Souto Costa

Orientador: Professor Dr. Bruno Marques Ferreira da Silva

Trabalho de conclusão de curso apresentada a Coordenação do Curso de Engenharia de Computação e Automação da UFRN (Área de concentração: Engenharia de Computação) como parte dos requisitos para obtenção do título de Bacharel em Engenharia de Computação.

Natal, RN, dezembro de 2018

Análise do RANSAC sobre algoritmo de odometria visual com imagens estéreo

Vanessa Dantas de Souto Costa

À minha família, por me apoiar em quaisquer decisões em minha vida acadêmica. Ao meu Orientador e colegas de equipe do NatalNet, por me incentivarem a perseguir meus objetivos e acreditarem em mim.

Agradecimentos

Ao meu orientador, professor Dr. Bruno Marques Ferreira da Silva, sou grato pela orientação, paciência e incentivo em toda a jornada desse trabalho.

Ao doutorando Luis Ortiz pela ajuda com o cálculo da disparidade e geração da nuvem de pontos para imagens estéreo, ainda em passos iniciais do trabalho.

Aos colegas Felipe Ferreira Barbosa, Viviana Cabrera, pelo conhecimento compartilhado, críticas e sugestões de melhoria.

Aos demais colegas de graduação, por todos os anos que passamos compartilhando experiências e conhecimento durante nossa formação.

À minha família pelo apoio durante esta jornada.

À UFRN, pelo apoio financeiro com a bolsa de iniciação científica.

Resumo

A motivação para esse projeto surgiu embasada em algoritmo de odometria visual, presente na literatura [da Silva & Gonçalves 2015], desenvolvido pelo Prof. Dr. Bruno Marques Ferreira da Silva. Os algoritmos desenvolvidos até então percebiam o movimento de objetos móveis do ambiente como sendo movimento realizado pela câmera. Tal problema é considerado relevante pela comunidade científica. Uma vez que o algoritmo RANSAC (*Random Sample Consensus*) é utilizado no algoritmo de odometria visual, consistindo em método para estimar os parâmetros de um determinado modelo a partir de um conjunto de dados contaminados por grandes quantidades de outliers. O objetivo do projeto é estudar o efeito da parametrização do RANSAC no algoritmo de odometria visual com imagens estéreo, estas foram provenientes do *dataset* KITTI. Outrossim, os algoritmos desenvolvidos foram implementados em C++ fazendo uso de bibliotecas e softwares consolidados nas pesquisas em robótica, tais como *OpenCV*, *Point Cloud Library* e ROS. Um estudo sobre características visuais, geometria estéreo e RANSAC foi feito para elaboração desse projeto. Parametrizamos o limiar do RANSAC por teste exaustivo e por algoritmo genético. Com isso, através dos experimentos realizados, esperamos ver o impacto que o RANSAC tem sobre o sistema (tanto em ambientes estáticos quanto dinâmicos).

Palavras-chave: Odometria Visual, RANSAC, Parametrização, Algoritmo Genético, Mapa de Disparidade, Imagem de Profundidade.

Abstract

The motivation for this project was based on visual odometry algorithm, presented in the literature [da Silva & Gonçalves 2015], developed by Prof. Dr. Bruno Marques Ferreira da Silva. The algorithms developed until then perceived the movement of moving objects of the environment as being motion performed by the camera. Such a problem is considered relevant by the scientific community. Since the RANSAC (Random Sample Consensus) algorithm is used in the visual odometry algorithm, consisting in a method to estimate the parameters of a given model from a set of data contaminated by large amounts of outliers. The objective of the project is to study the effect of the parameterization of the RANSAC in the visual odometry algorithm with stereo images, provided by KITTI dataset. In addition, the algorithms developed were implemented in C ++ using libraries and software consolidated in robotic research, such as OpenCV, Point Cloud Library and ROS. A study about visual characteristics, stereo geometry and RANSAC was done to elaborate this project. We parameterize the RANSAC threshold by exhaustive test and by genetic algorithm. With this, through the experiments carried out, we hope to see the impact that RANSAC has on the system (in both static and dynamic environments).

Keywords: Visual Odometry, RANSAC, Parametrization, Genetic Algorithm, Disparity Map, Depth Image.

Sumário

Sumário	i
Lista de Figuras	iii
Lista de Tabelas	iv
1 Introdução	1
1.1 Motivação	2
1.2 Contribuições	2
1.3 Metodologia	3
Lista de Símbolos e Abreviaturas	1
2 Teoria	5
2.1 Sistema de Referência	5
2.2 Estéreo	6
2.2.1 Retificação	7
2.2.2 Reconstrução (Projeção e Visão Tridimensional)	7
2.3 Odometria Visual	10
2.3.1 RANSAC	10
2.4 Parametrização do RANSAC	11
2.4.1 Teste Exaustivo	11
2.4.2 Algoritmo Genético	11
3 Trabalhos Relacionados	13
4 O Problema	15
4.1 Contexto do Problema	15
4.2 Parametrização	15
4.3 Odometria e variações no ambiente	16
4.4 Cálculo da Disparidade em imagens estéreo para pontos distantes	17
5 Algoritmo e Abordagem do Problema	19
5.1 Estrutura do Algoritmo de Odometria Visual	19
5.2 Medição do Erro	21
5.3 Algoritmo para parametrização do RANSAC	21
5.3.1 Estrutura do Algoritmo de Teste Exaustivo	23

5.3.2	Estrutura do Algoritmo Genético	23
6	Experimentos	26
6.1	Método para parametrização: teste exaustivo	26
6.2	Método para parametrização: algoritmo genético	30
6.3	Comparação de entre algoritmo genético e teste exaustivo	33
6.4	Análise de Resultados	35
6.5	Explição sobre cada <i>Dataset</i>	40
7	Conclusão	41
Referências Bibliográficas		42

Lista de Figuras

5		
7		
2.3	Reconstrução de Imagens Estéreo	8
2.4	Nuvem de pontos em dado instante da sequência 04	9
11		
2.6	Estrutura de um Algoritmo Genético	12
4.1	Etapas de uma VO	16
17		
5.1	Módulos do projeto <i>rgbd_rtk</i>	19
5.2	Nuvem de pontos	20
5.3	Rastreamento de <i>features</i>	21
5.4	Acoplamento do Algoritmo Genético com o RGBD_RTK	22
5.5	Acoplamento do algoritmo de Teste Exaustivo com o RGBD_RTK	22
5.6	Estrutura de um Algoritmo Genético	23
5.7	Estrutura de um Algoritmo Genético	24
6.1	Experimentos Realizados	26
6.2	Teste Exaustivo Gráfico 01	29
6.3	Teste Exaustivo Gráfico 06	29
6.4	Teste Exaustivo Gráfico 03	30
6.5	Teste Exaustivo Gráfico 04	30
6.6	Algoritmo Genético Gráfico 01	31
6.7	Algoritmo Genético Gráfico 06	32
6.8	Algoritmo Genético Gráfico 03	32
6.9	Algoritmo Genético Gráfico 04	33
6.10	Algoritmo Genético vs Teste Exaustivo: Erro sequência 01	33
6.11	Algoritmo Genético vs Teste Exaustivo: Erro sequência 03	34
6.12	Algoritmo Genético vs Teste Exaustivo: Erro sequência 04	34
6.13	Algoritmo Genético vs Teste Exaustivo: Erro sequência 06	34
6.14	Algoritmo Genético vs <i>Threshold</i> de maior erro: Erro sequência 01	36
6.15	Algoritmo Genético vs <i>Threshold</i> de maior erro: Erro sequência 03	37
6.16	Algoritmo Genético vs <i>Threshold</i> de maior erro: Erro sequência 04	38
6.17	Algoritmo Genético vs <i>Threshold</i> de maior erro: Erro sequência 06	39

Lista de Tabelas

6.1	Resultados <i>Dataset 01</i> - Teste Exaustivo	27
6.2	Resultados <i>Dataset 06</i> - Teste Exaustivo	27
6.3	Resultados <i>Dataset 03</i> - Teste Exaustivo	28
6.4	Resultados <i>Dataset 04</i> - Teste Exaustivo	28
6.5	Resultados <i>Dataset 01</i> - Algoritmo Genético	31
6.6	Resultados <i>Dataset 06</i> - Algoritmo Genético	31
6.7	Resultados <i>Dataset 03</i> - Algoritmo Genético	31
6.8	Resultados <i>Dataset 04</i> - Algoritmo Genético	31
6.9	Resultado <i>Dataset 01</i> - Nova faixa para Teste Exaustivo	35

Capítulo 1

Introdução

A odometria visual é um dos métodos mais usados para sistemas de navegação autônoma [Guedes Maidana 2017]. Isso, pois um veículo que utilize esse tipo de navegação precisa ser capaz de conhecer sua localização em realação ao meio em que está inserido. Em outras palavras, o robô precisa discernir sua própria localização, a definição da meta e o planejamento do trajeto à meta, conforme conceitos de navegação definidos por [Leonard 1991].

Em breve contextualização, a odometria visual (do inglês, *Visual Odometry*, VO) foi introduzida pela primeira vez para os robôs planetários operando em Marte - Moravec 1980, vide [*Visual Odometry (VO)* 2016]. Ela consiste em estimar a postura do robô por meio de imagens de uma ou mais câmeras [Scaramuzza & Fraundorfer 2011a], odometria vem do grego e significa "medir rota", enquanto visual se refere ao uso de imagens para tanto. Ela proporciona uma precisão suficiente para várias aplicações em movimentos curtos e permite taxas de amostragem muito altas. A odometria visual assim como a odometria clássica tem como fundamento a integração de informação incremental do movimento ao longo do tempo, o que implica no acúmulo de erros. No entanto, ao contrário da odometria convencional, a VO permite estimar a posição do robô mesmo quando acontece derrapagem ou rotação, visto que os dados fornecidos pela visão não são sensíveis a esse tipo de problema [Drews-Jr 2008].

Outrossim, existem diversos modelos de odometria visual, os quais se distinguem pela configuração de tipos de câmeras [Scaramuzza & Fraundorfer 2011b], por exemplo, podemos encontrar na literatura sistemas do tipo passivo, que utilizam apenas imagens para a VO, bem como, sistemas ativos, que usam sensores como LIDAR, GPS, sensores Ópticos (para outros veja [Eli S. dos Santos 2006]), por fim, podemos ter também configurações de odometria visual híbridas que usam múltiplos sensores.

Para a elaboração desse projeto, optamos por um sistema de VO passivo com uso de imagens estéreo. Isso, pois dentro da categoria de imagens existem: monocular, estéreo e omnidirecional. A monocular recebe esse nome por ser constituída por apenas uma câmera, o problema é que o uso de apenas uma imagem dificulta a percepção de profundidade sendo necessário incluir outros sensores ou usar estratégias de cálculo de profundidade com vários frames, conforme [Guedes Maidana 2017]. Já sistemas de visão omnidirecional produzem imagens de 360º do ambiente podendo ser utilizados em navegação, vide [Grassi Júnior 2002]. Por último, a odometria visual estéreo, consiste no

uso de duas ou mais câmeras cuja calibração deve ser conhecida (seja por especificação do produto ou por outros métodos de calibração, como alguns descritos em [Kaehler & Bradski 2016]). Este modelo permite resolver o problema de escala da visão monocular, pois através de duas imagens é possível determinar a profundidade.

1.1 Motivação

A motivação para esse projeto surgiu embasada na pesquisa de odometria visual com imagens estéreo desenvolvida pela equipe do NatalNet, coordenada pelo Prof. Dr. Bruno Marques Ferreira da Silva.

Até o presente momento, os algoritmos de rastreamento de características visuais desenvolvidos para o projeto possuem a deficiência de não serem robustos a ambientes dinâmicos. Esse fato ocorre, pois a odometria visual é dependente do ambiente que o robô está inserido. O projeto implementado pela equipe citada [da Silva & Gonçalves 2015] e [da Silva et al. 2017], usava imagens advindas de sensores Kinect em ambiente cuja iluminação e *features* visuais foram inseridas de forma tal que o robô pudesse determinar como estava se locomovendo. Ainda, o ambiente era estático ou com uma dinâmica muito lenta em relação ao mesmo.

Por causa dessa configuração, os algoritmos desenvolvidos percebiam o movimento de objetos móveis do ambiente como sendo movimento realizado pela câmera. Tal problema é considerado relevante pela comunidade científica [CADENA 2016]. Portanto, é proposto o estudo e desenvolvimento de algoritmos de rastreamento de características visuais que sejam robustos às dinamicidades presentes em ambientes dinâmicos.

Para que isso possa ocorrer, é de suma importância fazer uma análise do comportamento do projeto em ambientes externos, tanto estáticos quanto dinâmicos. Para esse trabalho, optamos por analisar a influência que a parametrização do RANSAC tem sobre o sistema e mostrar que somente isso não é suficiente para que o algoritmo fique robusto a objetos dinâmicos.

De acordo com [Zuliani 2012], o algoritmo RANSAC (*Random Sample Consensus*) foi introduzido pela primeira vez por Fischler e Bolles em 1981, como método para estimar os parâmetros de um determinado modelo a partir de um conjunto de dados contaminados por grandes quantidades de ruído.

Além disso, o RANSAC é um método iterativo e o número de iterações necessárias para encontrar uma solução correta é a exponencial do número mínimo de pontos de dados necessários para estimar o modelo. Portanto, é de suma importância encontrar a parametrização mínima do modelo para estimar com maior precisão os pontos que constituem *outliers*.

1.2 Contribuições

O objetivo do projeto é estudar o efeito da parametrização do RANSAC no projeto de odometria visual em ambientes dinâmicos, usando imagens estéreo provenientes do *dataset* KITTI [Geiger et al. 2012].

Especificamente a este trabalho, os algoritmos implementados devem ser testados quanto a sua robustez a ambientes dinâmicos (i.e. que contêm objetos móveis), utilizando diferentes valores para threshold do RANSAC.

Com isso, implementamos um algoritmo de rastreamento de características visuais que usa dataset com imagens estéreo em ambiente dinâmico [Geiger et al. 2012] e permite a parametrização do RANSAC em C++ e OpenCV. Assim, abordando a mesma problemática que [Hossein-Nejad & Nasri 2016], propusemos um método para escolha do limiar do RANSAC. Isso, pois esse método é altamente utilizado no campo da odometria visual. No entanto, seu valor limiar é fixo e é empiricamente escolhido pela maioria dos projetos que utilizam o RANSAC.

A parametrização do RANSAC foi feita através de duas estratégias: teste exaustivo, que consiste em testar valores de limiar desde um valor inicial até um final com incremento (passo) fixo, e algoritmo genético. Com isso, foi visto o impacto que o RANSAC tem sobre o sistema, como também, a importância de mecanizar a parametrização desse sistema. No Capítulo 6, devemos, através dos experimentos realizados, provamos que buscar soluções de aprendizado de máquinas (sobretudo expandindo o problema para mais de uma variável) é essencial como próximo passo nessa vasta área que constitui a odometria visual.

1.3 Metodologia

O algoritmo de rastreamento de características visuais a ser desenvolvido deve ser compatível com as plataformas robóticas presentes no Laboratório Natalnet (Departamento de Engenharia de Computação e Automação): dois robôs do tipo *Turtlebot* e um robô do tipo *Pioneer 3-AT*.

Uma vez que também é objetivo do projeto aplicar os algoritmos em cenários de uso relacionados a carros autônomos e drones, optamos pelo uso de conjuntos de dados disponíveis publicamente [Geiger et al. 2012]. O KITTI consiste em um conjunto de dados e *benchmarks* para pesquisa em visão computacional no contexto de direção autônoma. O conjunto de dados foi gravado na cidade de Karlsruhe, na Alemanha, usando a plataforma móvel AnnieWay (*VW station wagon*) que foi equipada com várias câmeras RGB e monocromáticas, um escâner a laser *Velodyne HDL 64*, bem como um GPS preciso corrigido com RTK. O conjunto de dados foi criado para pesquisa em visão computacional e aprendizado de máquina em estéreo, fluxo óptico, odometria visual, segmentação semântica, segmentação de instância semântica, segmentação de via, previsão de profundidade de imagem única, preenchimento de mapa de profundidade, detecção de objetos 2D e 3D e rastreamento de objetos. Além disso, várias gravações de dados brutos são fornecidas. Os conjuntos de dados são capturados pela condução em torno da cidade de tamanho médio de Karlsruhe, em áreas rurais e em rodovias. Até 15 carros e 30 pedestres são visíveis por imagem.

Sendo assim, os algoritmos foram implementados em C++ fazendo uso de bibliotecas e softwares consolidados nas pesquisas em robótica, tais como *OpenCV* e *Point Cloud Library*. Um estudo sobre características visuais, geometria estéreo e RANSAC foi feito para elaboração desse projeto. Em seguida, um sistema protótipo que parametriza o RAN-

SAC e alimenta o projeto de rastreamento de características visuais já citado foi implementado. Para fazer o estudo proposto (de parametrização do *threshold* do RANSAC), implementamos também scripts em python 2 que fazem uso de duas metodologias distintas: teste exaustivo e algoritmo genético. Os resultados foram catalogados de acordo com o erro gerado pela própria ferramenta de desenvolvimento do KITTI (Development Kit) [*Summary for IJRR 2013 paper 2013*]. Desse modo, pudemos verificar as vantagens e desvantagens do sistema desenvolvido.

Capítulo 2

Teoria

Neste capítulo, discutiremos os conceitos que embasaram a pesquisa. Dentre os principais tópicos, pode-se citar o sistema de referência do *dataset* desenvolvido pela KIT em parceria com o Instituto de Tecnologia Toyota [Geiger et al. 2012], a geometria estéreo, na qual iremos discorrer sobre a correspondência (retificação) de imagens e reconstrução (projeção e visão tridimensional) e o conceito de odometria visual com foco no algoritmo do *Random Sample Consensus*.

2.1 Sistema de Referência

Uma vez que o utilizamos a geometria estéreo é necessário tratar o problema a partir de um mesmo sistema de referências. Fazendo isso, será possível calcular a disparidade e, a posteriori, a profundidade dos pontos do par de imagens. Para que isso seja possível o sistema deve estar previamente calibrado. A Figura 2.1 abaixo retirada de [Summary for IJRR 2013 paper 2013] ilustra a disposição dos sensores no carro usado para construção do dataset KITTI.

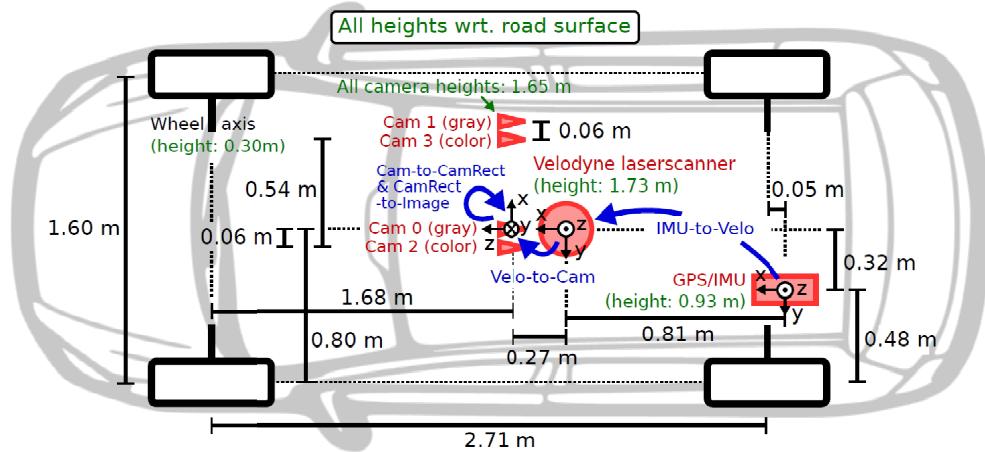


Figura 2.1: Configuração dos Sensores do KITTI, retirada de [*Summary for IJRR 2013 paper 2013*]

Na Seção 2.2, é explicado o processo de retificação (o par estéreo é processado a partir de mesmo sistema de coordenadas) e de reconstrução da imagem.

2.2 Estéreo

Os métodos de estimativa de profundidade podem ser classificados como estereoscópico ou monoscópico. Essa classificação pode ser facilmente exemplificada ao observar a própria fauna.

Predadores como leões possuem dois globos oculares na face posicionados paralelamente sob mesmo eixo horizontal (mesmo plano), estes, capazes de ver com ambos os olhos simultaneamente, são ditos com visão binocular e a percepção de profundidade desta forma é denominada de visão estereoscópica [Siscoutto 2004] e pode ser calculada por triangulação. Os predadores costumam ter esse tipo de visão, por ele ser mais preciso quanto ao cálculo de profundidade (funcionando perfeitamente para alvos a curta distância, já a longa distância, a visão estereoscópica se comporta de maneira análoga à visão monocular, fato esse, a ser abordado no Capítulo 4).

Por sua vez as presas, por exemplo cervos, priorizam observar o máximo do ambiente possível, não sendo necessária tanta precisão quanto à profundidade, porque caso uma possível ameaça seja detectada, basta correr na direção oposta. Assim, possuem dois olhos em planos paralelos (observam o ambiente quase em 360º), o que constitui uma visão monocular. Esse termo é aplicado para a observação com apenas um dos olhos e, o método de julgamento de distância é denominado monoscópico, vide [Guedes Maidana 2017].

Retomando o conceito de imagem estéreo, vide Capítulo 1, que consiste no uso de duas câmeras cuja calibração deve ser conhecida (seja por especificação do produto ou por outros métodos de calibração, como alguns descritos em [Kaehler & Bradski 2016]), este modelo permite resolver o problema de escala da visão monocular, pois através de duas imagens é possível determinar a profundidade. Podemos fazer um paralelo com o sistema visual humano [Siscoutto 2004], uma vez que a base para a percepção estereoscópica é a disparidade binocular do sistema visual humano, que gera duas imagens ligeiramente diferentes quando uma cena é projetada nas retinas dos olhos. As duas perspectivas diferentes das imagens são fundidas no córtex visual do cérebro, de forma a compor uma simples visão estereoscópica (tridimensional). Esse processo pode ser simulado através de duas câmeras organizadas com a mesma distância interocular dos olhos humanos. Logo, colocando-se as câmeras separadas uma da outra com base nessa distância, simula-se o sistema visual humano. Quando cada imagem das câmeras for apresentada ao seu olho correspondente, as duas imagens serão fundidas em uma única imagem pelo cérebro, produzindo a ilusão de visão estereoscópica.

Refraseando o conceito acima, pode-se afirmar que estereoscopia é a propriedade que estuda os métodos e técnicas que permitem a visão em perspectiva, quer dizer, a percepção de objetos com todas as modificações aparentes, ou com os diversos aspectos que a sua posição e situação determinam com relação à figura e à luz [Temba 2000].

2.2.1 Retificação

A retificação da imagem consiste em projetá-la, segundo seu próprio feixe perspectivo para um plano horizontal, segundo [Andrade 1998]. Em outras palavras, podemos modificar a imagem de modo a tentar eliminar os ângulos da câmera em relação a um dado referencial, bem como a distância focal da imagem, vide Figura 2.2 retirada de [Visão Estéreo 2001].

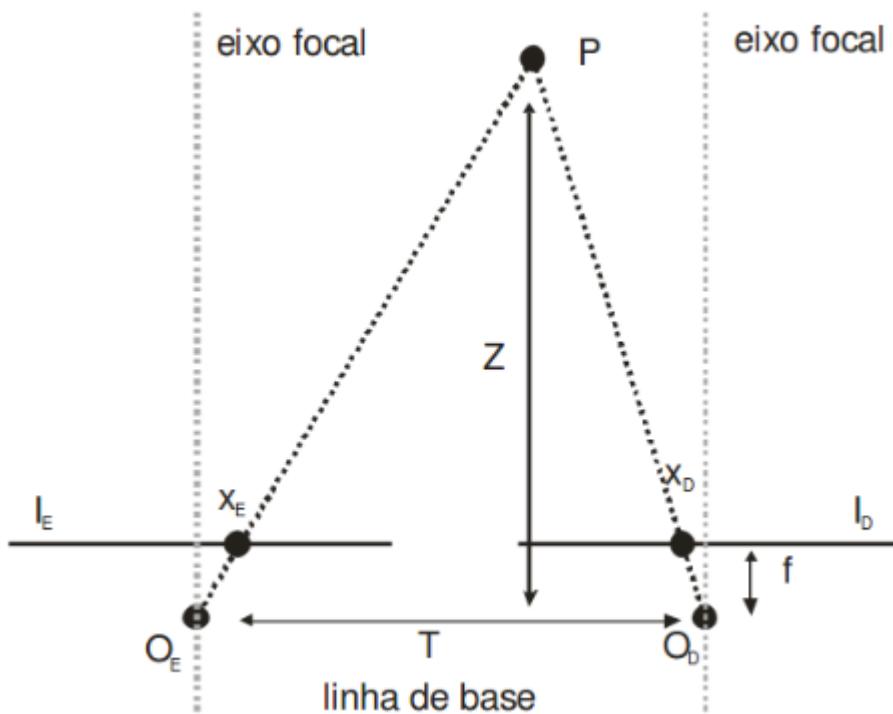


Figura 2.2: Conceito da retificação de um par de imagens, retirada de [Visão Estéreo 2001]

Na Figura 2.2 acima, o P se refere ao ponto observado, O_e e O_d correspondem ao observador (câmera) esquerda e direita, respectivamente, f é a distância focal, T é a linha de base.

No caso do KITTI [Geiger et al. 2012], as imagens disponíveis já estavam retificadas, conforme o arquivo *readme.txt* do *Devkit* do próprio *dataset*. Logo, pudemos prosseguir para o processo de cálculo de profundidade e reconstrução.

2.2.2 Reconstrução (Projeção e Visão Tridimensional)

O objetivo da reconstrução é gerar uma nuvem de pontos (do inglês, *point cloud*) para mapeamento de características visuais, reconhecimento de objetos, vigilância, entre outras aplicações. O importante é que, para gerar um modelo 3D nesse formato (nuvem de pontos), precisamos calcular a disparidade e a profundidade do par de imagem estéreo. Os passos para a reconstrução de um par de imagens estéreo está ilustrada na Figura 2.3:



Figura 2.3: Reconstrução de Imagens Estéreo

Dadas duas imagens capturadas a partir de diferentes pontos de vista, o problema é, conforme apresentado na seção anterior, a correspondência estéreo. A ideia é determinar a diferença de localização de pontos correspondentes em ambas as imagens. Para o caso de imagens já retificadas, essa diferença é calculada pelo deslocamento no eixo x de um pixel na imagem direita em relação à imagem esquerda. Esse deslocamento é chamado de disparidade. Através do mapa de disparidades podemos utilizar regras de triangulação para obter o mapa de profundidade.

Uma imagem de profundidade é uma imagem digital onde cada pixel armazena uma informação de profundidade. Essa informação é um valor de distância de um ponto visível na cena a um sistema de referência conhecido do sensor. Dessa maneira, no projeto desenvolvido, a profundidade do par de imagens estéreo foi calculada por triangulação, dado que as imagens já estavam retificadas, o cálculo da disparidade foi feito como descrito acima.

Métodos de triangulação, tanto passivos quanto ativos, exploram relações de proporção trigonométricas básicas existentes em triângulos. Pela lei dos senos, se o comprimento de um lado e dois ângulos interiores de um triângulo são conhecidos, essas informações são suficientes para se determinar os comprimentos dos outros dois lados e o ângulo restante [Conrad. D.J. e Sampson 1990].

Assim, podemos calcular a profundidade Z , através das relações descritas matematicamente conforme:

$$\frac{T-d}{Z-f} = \frac{T}{Z} \implies Z = \frac{f \cdot T}{d} \quad (2.1)$$

Na qual d representa a disparidade e é calculada pela diferença entre os pontos correspondentes da imagem da direita em relação à esquerda, T corresponde à linha de base, f é a distância focal. Esse processo é minunciosamente descrito por [Kaehler & Bradski 2016] como uma semelhança de triângulos.

Para que possamos calcular a nuvem de pontos, é necessário uma matriz de reprojeção Q para projetar os pontos bidimensionais da imagem no plano tridimensional da nuvem de pontos, ou seja, os pontos citados contêm as coordenadas X, Y e Z no plano tridimensional da nuvem de pontos. Para esse cálculo utilizamos os valores intríscos da câmera já mencionados (f, T e as coordenadas x e y dos pontos principais da câmera esquerda).

Abaixo, descrevemos a matriz Q utilizada pela *OpenCV*:

$$Q = \begin{bmatrix} 1 & 0 & 0 & -c_x \\ 0 & 1 & 0 & -c_y \\ 0 & 0 & 0 & f \\ 0 & 0 & -\frac{1}{T_x} & \frac{c_x - c'_x}{T_x} \end{bmatrix}$$

$$\begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix} = Q \cdot \begin{bmatrix} x \\ y \\ \text{disparidade} \\ 1 \end{bmatrix}$$

Para ilustrar os conceitos apresentados acima, na Figura 2.4 abaixo podemos ver a nuvem de pontos calculada em um dado instante da sequência 04 da KITTI.

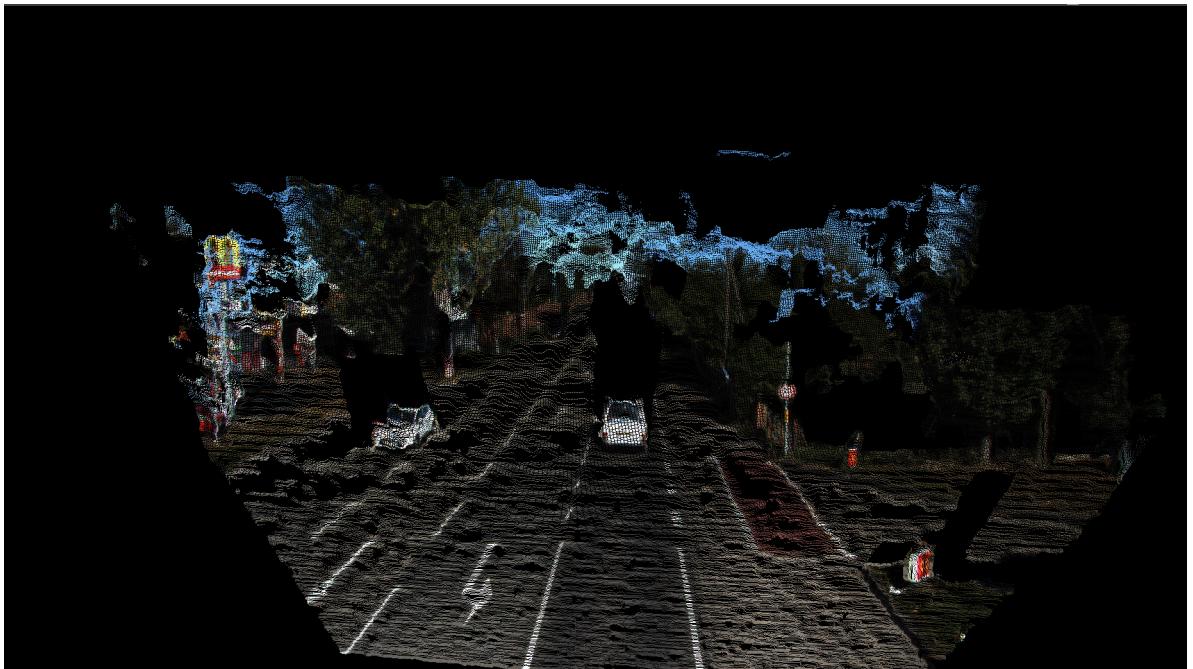


Figura 2.4: Nuvem de pontos em dado instante da sequência 04

2.3 Odometria Visual

Conforme discutido no Capítulo 1, a odometria visual é bastante utilizada como método para determinar a posição e a orientação de um robô, analisando as imagens da câmera associadas. As principais etapas para um algoritmo de odometria visual podem ser descritos conforme o próximo parágrafo.

Definidas as imagens de entrada (monocular, estéreo, omnidirecional), é necessário aplicar técnicas de processamento de imagem para remoção de distorções (p. ex. distorção de lente), em consequente deve-se construir um campo de fluxo óptico (fluxo óptico é apenas uma implementação possível, estamos citando-a pois foi a utilizada para o projeto) a partir dos operadores de interesse. A partir desse ponto, é preciso calcular a correlação para estabelecer correspondência entre duas imagens. Dessa maneira, podemos realizar a etapa de extração e correlação de recursos. Assim, constrói-se o campo de fluxo óptico. Feito isso, verificamos vetores de campo de fluxo para possíveis erros de rastreamento e removemos os valores discrepantes (no projeto desenvolvido isso é feito por desconsiderar os outliers determinados pelo RANSAC). Por último, há repopulação periódica de trackpoints para manter a cobertura em toda a imagem.

2.3.1 RANSAC

O algoritmo RANSAC constitui um método para estimar os parâmetros de um determinado modelo a partir de um conjunto de dados contaminados por grandes quantidades de *outliers* vide Capítulo 1 e [Zuliani 2012]. *Outliers* são valores atípicos, pois não atendem o modelo “verdadeiro” instanciado pelo conjunto “verdadeiro” de parâmetros dentro de algum limite de erro (*threshold*) que define o desvio máximo atribuível aos efeitos do ruído.

Ainda, de acordo com mesma fonte, apesar de muitas modificações, o algoritmo RANSAC é essencialmente composto de duas etapas que são repetidas de maneira iterativa (estrutura de hipótese e teste).

O algoritmo RANSAC é um método de detecção atípica, ou seja, tem o objetivo de diminuir o erro do sistema, escolhendo um grupo com número mínimo pontos dependendo do modelo matemático (reta, plano, círculo, matriz de rotação e translação, entre outros) para criar uma “hipótese”, ou seja, um modelo que calcule a rotação e translação feita pela câmera. Um pressuposto básico é que os dados consistem em “*inliers*”, isto é, dados cuja distribuição pode ser explicada por algum conjunto de parâmetros de modelo, embora possam estar sujeitos a ruído, e “*outliers*” que são dados que não se encaixam no modelo. A hipótese com maior número de *inliers* é escolhida como a verdadeira transformação de posição da câmera.

Veja exemplo da Figura 2.5 retirada de [Zuliani 2012] de otimização do cálculo de uma reta dado os pontos em verde.

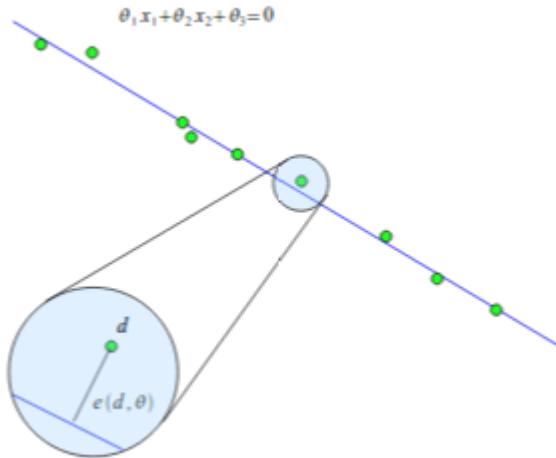


Figura 2.5: Exemplo de *line fitting*, retirada de [Zuliani 2012]

Por fim, é feito registro da nuvem de pontos atual na anterior, usando a transformação obtida pelo RANSAC para o cálculo da nova posição do objeto.

2.4 Parametrização do RANSAC

Conforme discutido, o que define o conceito de *outliers* no *Random Sample Consensus* é um modelo “verdadeiro” instanciado pelo conjunto “verdadeiro” de parâmetros dentro de algum limite de erro (*threshold*). Este limite de erro corresponde ao desvio máximo atribuível aos efeitos do ruído.

Além disso, no Capítulo 1, foi explicitado que o objetivo do trabalho é analisar o RANSAC através da parametrização do mesmo.

2.4.1 Teste Exaustivo

O teste exaustivo consiste em testar todos os valores, a partir de um valor inicial até um final, com incremento fixo. Tanto o início, o fim e o incremento são definidos pelo usuário. Essa metodologia testa uma faixa de valores discretos para que, com base nos resultados, possamos concluir qual o melhor valor para determinado problema.

2.4.2 Algoritmo Genético

Em 1975, John Holland propôs um modelo de algoritmo baseado nos conceitos de seleção natural da espécie e na teoria da evolução. A este modelo, deu-se o nome de algoritmo genético.

Vide [*algoritmos genéticos: conceitos básicos* 2017], uma definição de algoritmos genéticos poderia ser: um conjunto predeterminado e bem definido de regras e processos com operações finitas, destinados à busca estocástica polarizada da solução de um

problema, com um número finito de etapas. Os algoritmos genéticos seguem as leis biológicas da transmissão dos caracteres hereditários nos indivíduos e os mecanismos que asseguram essa transmissão.

A estrutura básica de um algoritmo genético pode ser vista na Figura 2.6 abaixo:

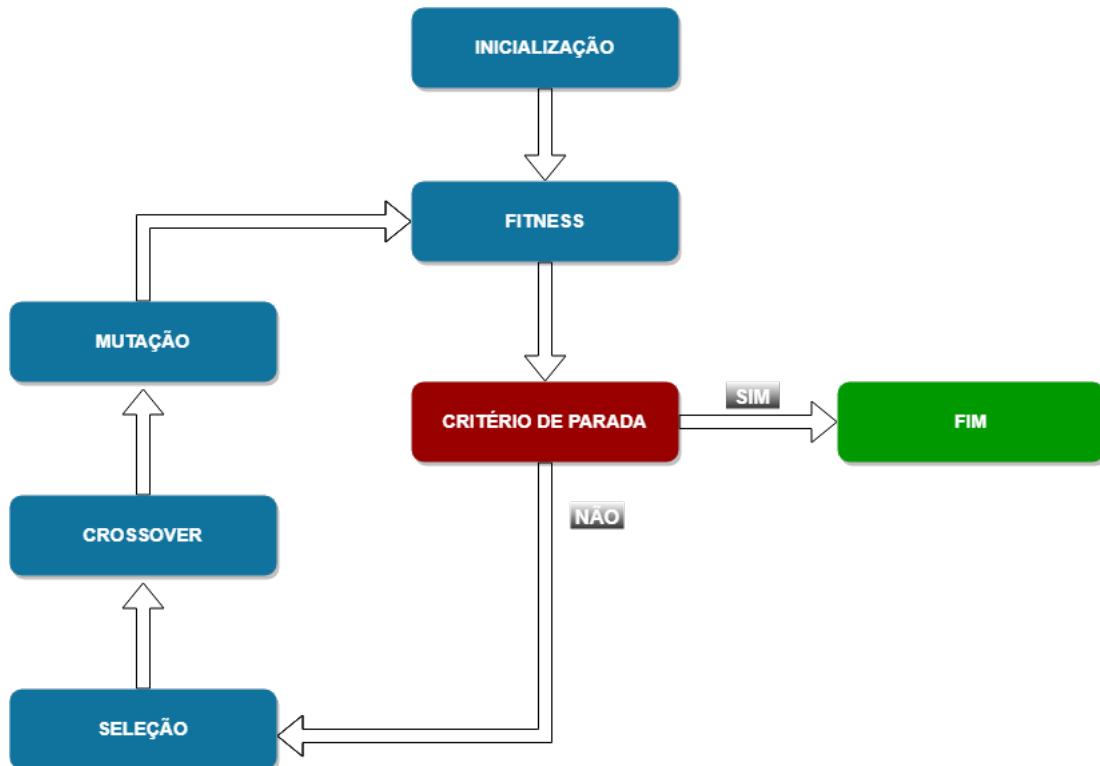


Figura 2.6: Estrutura de um Algoritmo Genético

Capítulo 3

Trabalhos Relacionados

Conforme discorrido no Capítulo 1, esse projeto surgiu embasado em [Scaramuzza & Fraundorfer 2011b] e na pesquisa de odometria visual com imagens estéreo desenvolvida pela equipe do NatalNet, coordenada pelo Prof. Dr. Bruno Marques Ferreira da Silva. O mesmo, possui diversos trabalhos na área, como os trabalhos de [da Silva & Gonçalves 2015] e de [da Silva et al. 2017] que fazem parte do nosso estudo.

Os projetos originais foram adaptados para utilizar o dataset disponível publicamente KITTI do KIT em parceria com o Instituto de Tecnologia da Toyota [Geiger et al. 2012]. Além disso, parametrizamos o threshold do RANSAC utilizando scripts em python 2.

Verificamos que o trabalho de [Scaramuzza 2011] faz uma análise do RANSAC sob um sistema de odometria visual monociliar. Apresentando experimentos em dados sintéticos e reais com o intuito de comparar o desempenho com abordagens de última geração e com diferentes veículos, tanto interiores como exteriores. Esse trabalho é interessante, pois a maior parte dos trabalhos na área de odometria visual foi produzido a partir de câmeras estéreo.

Outro trabalho que serviu de inspiração, ao cogitarmos o tema desse trabalho de conclusão de curso, foi [Hossein-Nejad & Nasri 2016]. Nele, é proposto um método para escolha do limiar do RANSAC, no qual o valor limite é calculado com base na variação entre as classes de correspondência correta e incompatibilidade. Isso, pois o RANdom SAmple Consensus (RANSAC) é altamente utilizado no campo da odometria visual. No entanto, seu valor limiar é fixo e é empiricamente escolhido pela maioria dos projetos que utilizam esse método. Ainda, em [Hossein-Nejad & Nasri 2016], vemos que os resultados da simulação obtidos pelos autores confirmam a superioridade do limiar escolhido em diferentes situações em comparação com os algoritmos clássicos RANSAC.

Em virtude disso, fazendo um paralelo com o trabalho de [Scaramuzza 2011], pensamos em analisar o impacto do RANSAC no projeto desenvolvido pela nossa equipe, a partir da mudança do valor de limiar (threshold). Para tanto, assim como [Hossein-Nejad & Nasri 2016] resolvemos automatizar o processo de definição desse limiar.

Uma vez que técnicas de aprendizado de máquina são amplamente utilizadas para parametrização de modelos matemáticos, resolvemos utilizar um algoritmo genético para tentar minimizar o erro de translação do sistema. Nesse aspecto, podemos citar outros trabalhos, como [Pedrosa 2014]. Nele vemos o uso de um algoritmo genético para minimizar a função Peaks do MATLAB. No nosso projeto optamos pela utilização de python 2, apenas por questões de facilidade para implementação.

Além disso, é válido salientar que o algoritmo genético foi desenvolvido a partir de uma estrutura básica proposta por trabalho desenvolvido por Troy Squillaci, projeto disponível em seu github [*Sample code for an incredibly simple genetic algorithm* 2018].

Capítulo 4

O Problema

Neste capítulo, será apresentado o problema de pesquisa. Nele serão levantadas questões para serem respondidas através da hipótese desse trabalho de conclusão de curso. A abordagem do problema que irá tratar da hipótese será discutida no Capítulo 5 e, no Capítulo 6 serão apresentadas as experiências realizadas para validação da mesma.

4.1 Contexto do Problema

A robustez a ambientes dinâmicos e diferentes iluminações em algoritmos de odometria visual é objetivado pela comunidade científica [CADENA 2016]. Seja pela crescente pressão industrial para geração de carros autônomos [*Carros autônomos já começam a virar realidade* 2018] ou pela tentativa de solucionar os problemas científicos considerados como estado da arte no campo de odometria visual.

Sendo assim, como forma de contribuição para com a pesquisa, e pensando no fato de que a maioria dos algoritmos estéreo de VO se baseam em testes empíricos para parametrizar os algoritmos (como é o caso do RANSAC), propomos como problema uma análise do RANSAC a partir de métodos para parametrizar o limiar que separa os *outliers* dos *inliers*.

Esta problemática já foi abordada por diferentes autores, citados no Capítulo 3 de trabalhos relacionados.

4.2 Parametrização

No campo de controle inteligente, existe o sistema especialista. Estes são programas que tentam simular o raciocínio de um profissional “*expert*” (normalmente com anos de experiência) em alguma área de conhecimento bem específica.

Estes sistemas costumam ser suficientes para parametrizar inúmeros problemas das plantas industriais. Eles aprendem o funcionamento do sistema básico e atuam substituindo a mão de obra humana para tarefas mais básicas (por exemplo inserir valor de tensão para alimentar um sistema de bombas). Contudo, contradizendo o pensamento leigo, esse sistema, por si, não é capaz de substituir o especialista. Outrossim, além de requererem a presença de profissional especialista, atendem apenas aplicações limitadas ao conhecimento humano [Sistemas Especialistas 1996].

Em virtude disso, não podemos aplicar essa solução para a parametrização do RANSAC. O *dataset* escolhido contém muitas variações de iluminação e movimento, o que dificulta a escolha de um limiar fixo para o *threshold* do RANSAC.

4.3 Odometria e variações no ambiente

De acordo com [Scaramuzza & Fraundorfer 2011b], Odometria visual (VO) é o processo de estimar o *egomotion* (termo em inglês que se refere a estimar o movimento de uma câmera em relação a uma cena rígida) de um agente usando apenas a entrada de uma única ou várias câmeras anexadas a ele. Domínios de aplicação incluem robótica, realidade aumentada, automotivo e outros. O termo VO foi cunhado em 2004 por Nister. O termo foi escolhido por sua semelhança com a odometria das rodas, que estima o movimento de um veículo, integrando o número de voltas de suas rodas ao longo do tempo. Da mesma forma, a odometria visual opera incrementalmente estimando a posição do veículo através do exame das mudanças que o movimento induz na imagens de suas câmeras a bordo. Para VO trabalhar de forma eficaz, deve haver iluminação satisfatória no ambiente e uma cena estática com textura suficiente para permitir que movimento aparente possa ser extraído. Além disso, consecutivos quadros devem ser capturados, garantindo que eles tenham superposição de cena aceitável.

Abaixo, vemos os passos básicos da odometria visual, pela Figura 4.1.

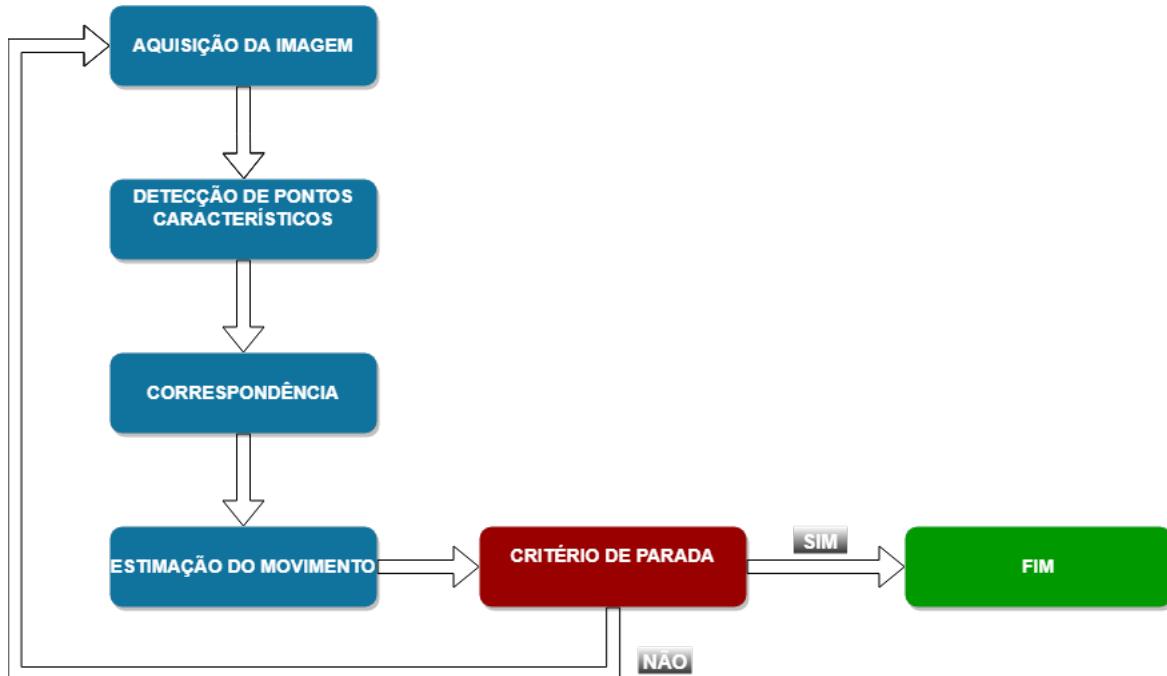


Figura 4.1: Etapas de uma VO

Embora esse modelo pareça simples, podemos citar alguns empecilhos ao aplicar as técnicas de VO. Dependendo do modelo que for escolhido para sistema, a odometria visual pode necessitar de um alto poder computacional (como, por exemplo, uma GPU). O

projeto desenvolvido pela equipe do NatalNet é pensado para ser acessível tanto em custo, por não utilizar uma configuração híbrida (híbrida diz respeito ao uso de vários tipos distintos de sensores e alguns sensores podem ter custo elevado), quanto em processamento, podendo ser rodado em um computador com CPU comum.

No entanto, a maior problemática da estratégia de odometria visual consiste na falta de robustez do projeto a movimentos e variação de luminosidade. Os algoritmos desenvolvidos percebem o movimento de objetos móveis do ambiente como sendo movimento realizado pela câmera.

4.4 Cálculo da Disparidade em imagens estéreo para pontos distantes

Com base no trabalho [Chang & Chatterjee 1992], podemos ver matematicamente a relação entre o cálculo da disparidade e a geração da imagem de profundidade. Nele, é possível ver como o erro de correspondência de cada componente no cálculo da disparidade afeta a profundidade, o erro absoluto de profundidade é igual à soma da derivada parcial da profundidade para cada componente.

Além disso, utilizando tanto o estudo citado como o próprio livro do opencv [Kaehler & Bradski 2016], vemos que ao tratar de pontos distantes, o caso estéreo é degenerado para monocular. Para explicar esse conceito, primeiramente observe a Figura 4.2 abaixo retirada do livro de *Opencv*:

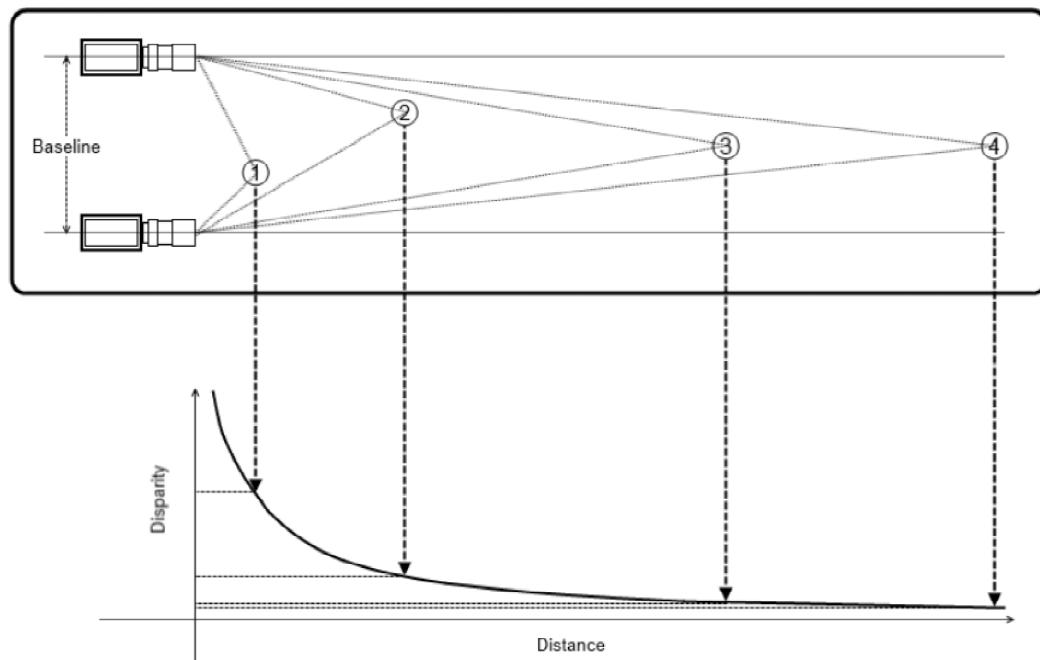


Figura 4.2: Relação entre disparidade e o ponto principal, retirada de [Kaehler & Bradski 2016]

Nela, vemos que pontos distantes resultam em linhas quase perfeitamente paralelas, o que implica em uma disparidade tende a zero. Por isso, o cálculo da profundidade é comprometido. Conforme [Kaehler & Bradski 2016], a profundidade é inversamente proporcional à disparidade, há obviamente uma relação não-linear relação entre esses dois termos. Quando a disparidade é quase zero, pequenas mudanças na disparidade causam grandes diferenças na profundidade. Quando a disparidade é grande, pequenas diferenças no cálculo do mapa de disparidade não afetam de maneira significativa a imagem de profundidade.

Capítulo 5

Algoritmo e Abordagem do Problema

Neste capítulo será discutido o software desenvolvido, bem como a abordagem pensada para parametrização do RANSAC.

5.1 Estrutura do Algoritmo de Odometria Visual

Conforme Figura 5.1, ilustramos os módulos do sistema implementado:

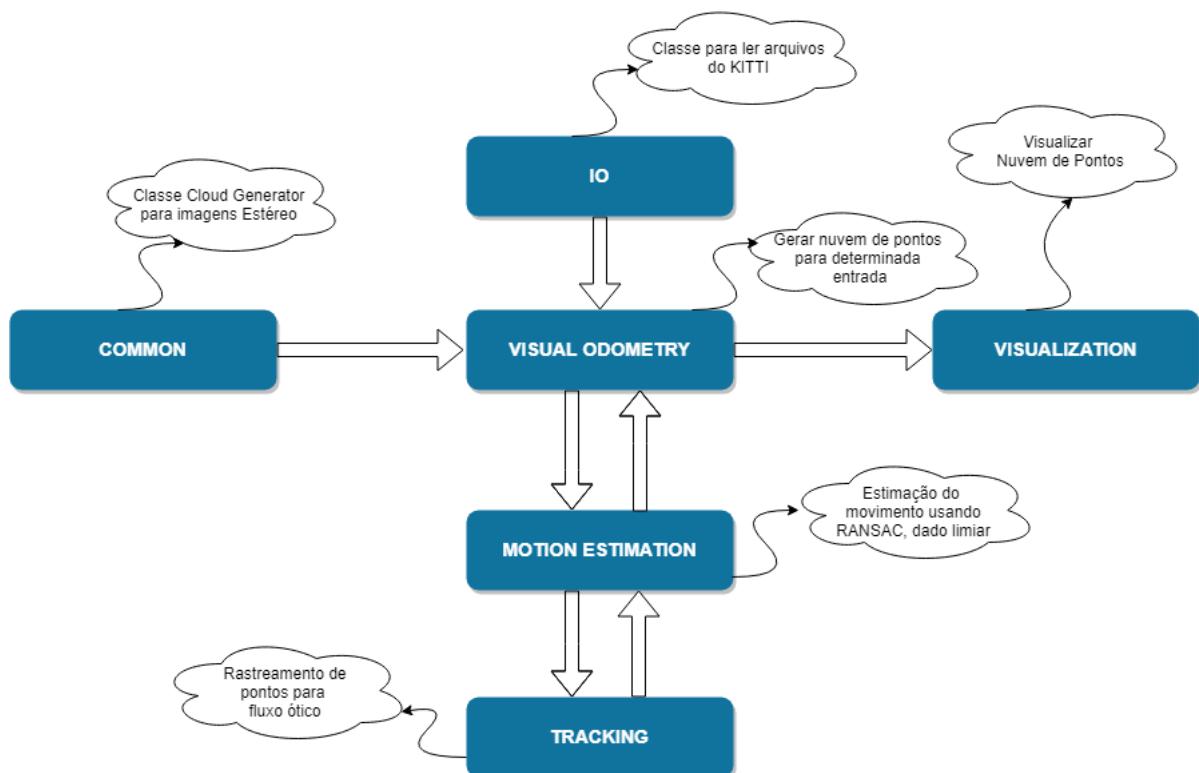


Figura 5.1: Módulos do projeto *rgbd_rtk*

O algoritmo de odometria visual construído pela equipe do NatalNet, chamado *rgbd_rtk*, está estruturado em módulos. **IO** é o módulo responsável pela entrada e saída do sistema. Nele, fizemos uma classe própria para ler os arquivos da KITTI. O módulo **Common**, possui classes que não entraram em outras categorias, nele, implementamos a classe do cloud generator para a KITTI. Em **Motion Estimator**, há a implementação de uma classe para estimar o movimento usando RANSAC. Essa classe foi modificada para receber como entrada o *threshold*, objeto alvo de análise desse trabalho de conclusão de curso. No módulo **Tracking**, é feito o rastreamento de pontos para fluxo óptico. O módulo **Visualization** serve para visualização da nuvem de pontos. Por fim, temos o **Visual Odometry** feito para relacionar os demais módulos com a finalidade de gerar a nuvem de pontos para determinada entrada e sua visualização.

Além disso, as Figuras 5.2 e 5.3 ilustram a nuvem de pontos e a visualização do rastreamento de *features*, respectivamente, geradas ao executar o projeto. Vale salientar que essas imagens são provenientes do nosso projeto executando a sequência 04 do *dataset* da KITTI.

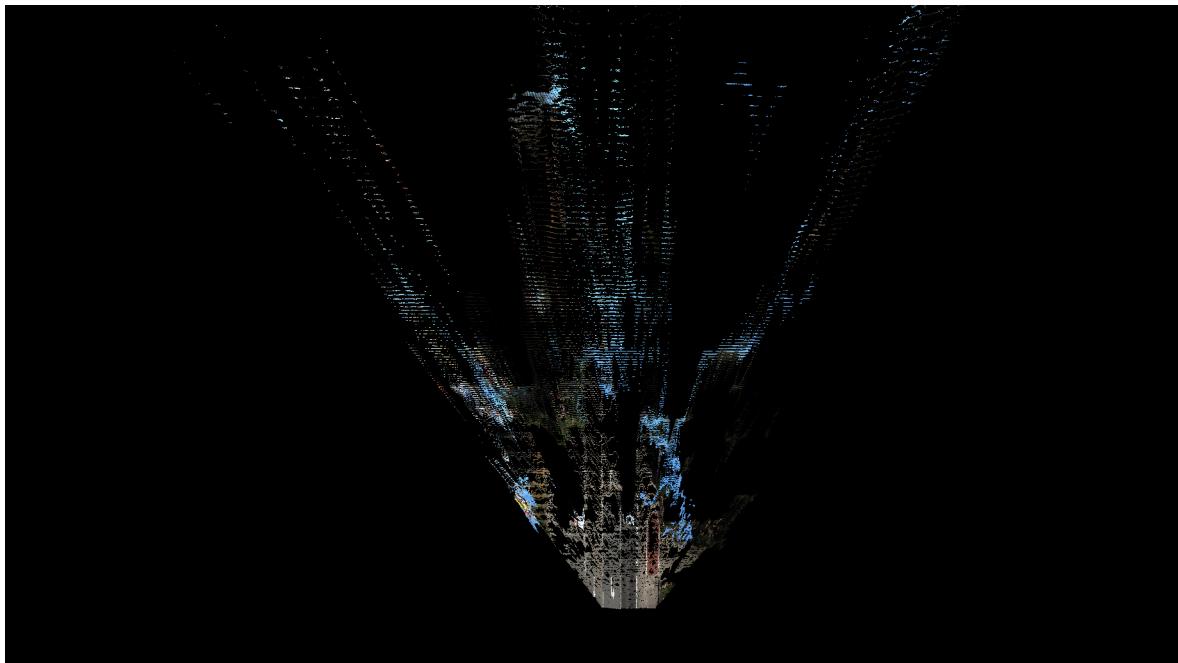


Figura 5.2: Nuvem de pontos

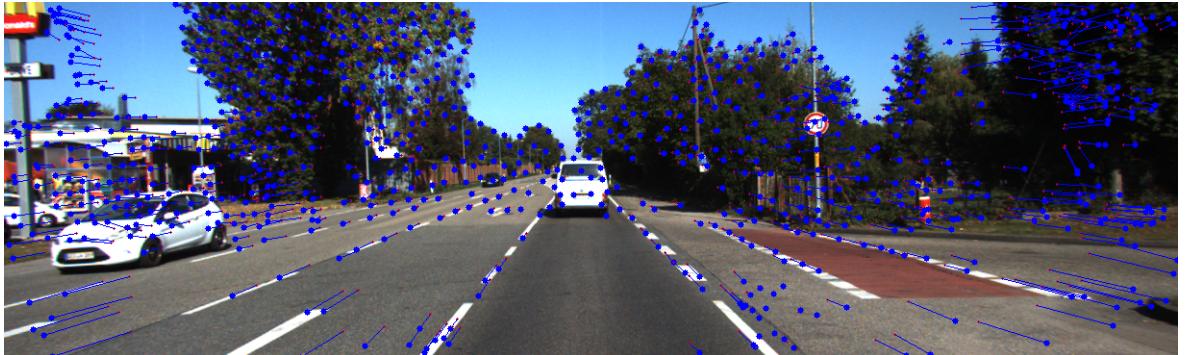


Figura 5.3: Rastreamento de *features*

5.2 Medição do Erro

A métrica utilizada para determinar se o *threshold* melhorou ou não o desempenho do projeto foi o erro de translação calculado pelo *development kit* do *dataset* [Geiger et al. 2012].

O erro de translação de um sistema é calculado com base no erro de rotação que usa como métrica o erro de posição relativa (do inglês, *Relative Pose Error* ou RPE). O relative pose error é calculado entre as correspondências. Ele calcula o erro entre todos os pares de pontos da trajetória estimada, ou seja, para todas as imagens de uma sequência especificada. Por fim, para analisarmos um único valor de erro, fazemos a média do erro gerado para cada par de pontos correspondentes. Um script que exemplifica este conceito pode ser visto em [Useful tools for the RGB-D benchmark 2012]. Ainda, o trabalho [Martinelli 2017] propicia mais detalhes sobre o conceito de como calcular o erro de um sistema de odometria.

5.3 Algoritmo para parametrização do RANSAC

Conforme discutido em capítulos anteriores, o objetivo do projeto é estudar o efeito da parametrização do *Random Sample Consensus* no projeto de odometria visual em ambientes dinâmicos, usando imagens estéreo provenientes do dataset KITTI [Geiger et al. 2012].

Para tanto, a parametrização do RANSAC foi feita por teste exaustivo e por algoritmo genético. Sendo assim, discutiremos ambas as implementações. Além disso, para ilustrar melhor como esses algoritmos se acoplam com o projeto implementado, veja as Figuras 5.4 e 5.5 abaixo.

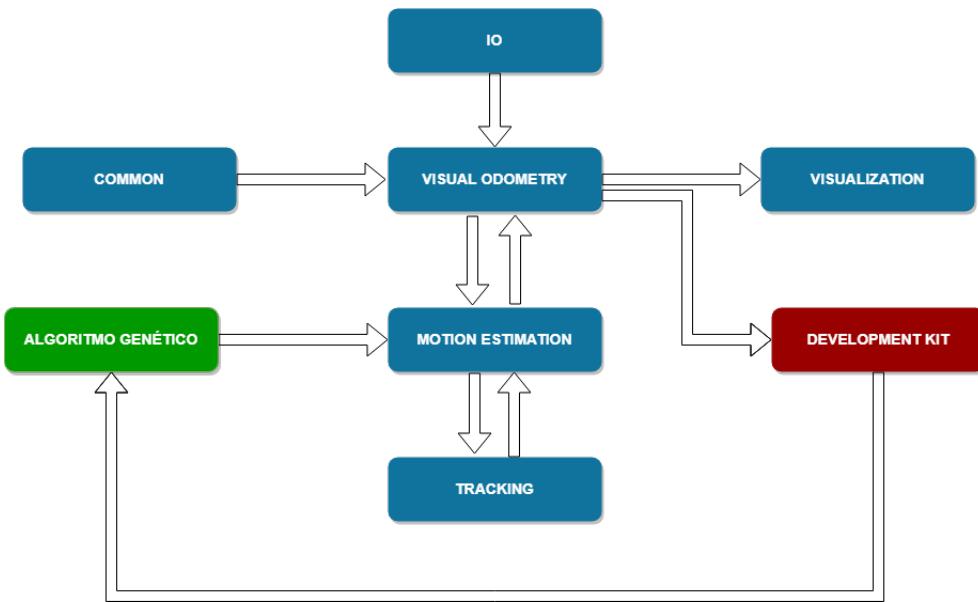


Figura 5.4: Acoplamento do Algoritmo Genético com o RGBD_RTK

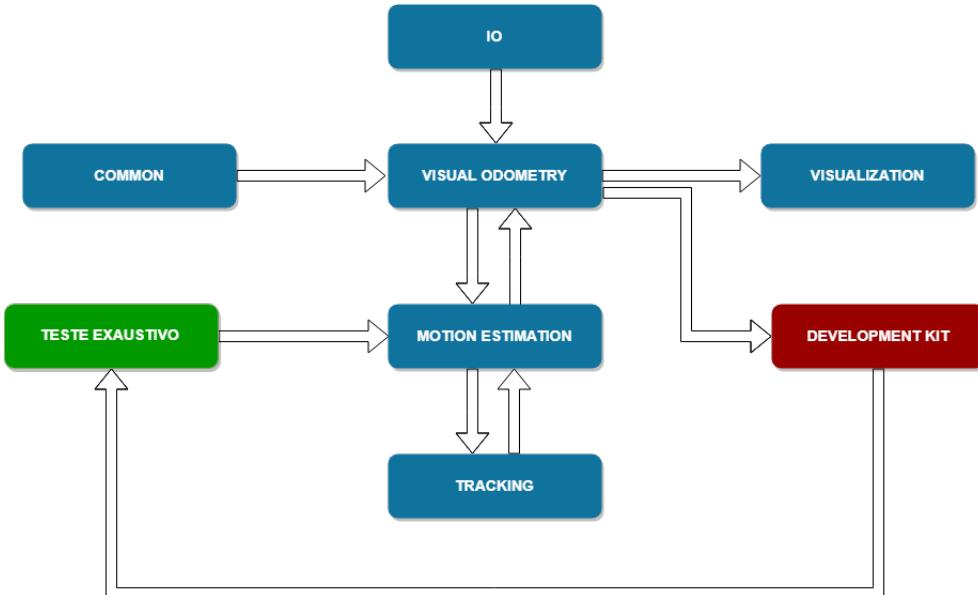


Figura 5.5: Acoplamento do algoritmo de Teste Exaustivo com o RGBD_RTK

5.3.1 Estrutura do Algoritmo de Teste Exaustivo

Para o teste exaustivo, a metodologia escolhida foi a partir de um valor de *threshold* inicial até um valor final, com incremento fixo, todos escolhidos pelo usuário, executar o projeto *rgbd_rtk* para cada valor de limiar e salvar em um arquivo de texto os resultados. Conforme Figura 5.6:



Figura 5.6: Estrutura de um Algoritmo Genético

5.3.2 Estrutura do Algoritmo Genético

Por sua vez, a estruturação do algoritmo genético seguiu os passos de geração de população inicial, *fitness*, *selection*, *crossover* e *mutation* conforme [algoritmos genéticos: conceitos básicos 2017]. Veja Figura 5.7 abaixo:

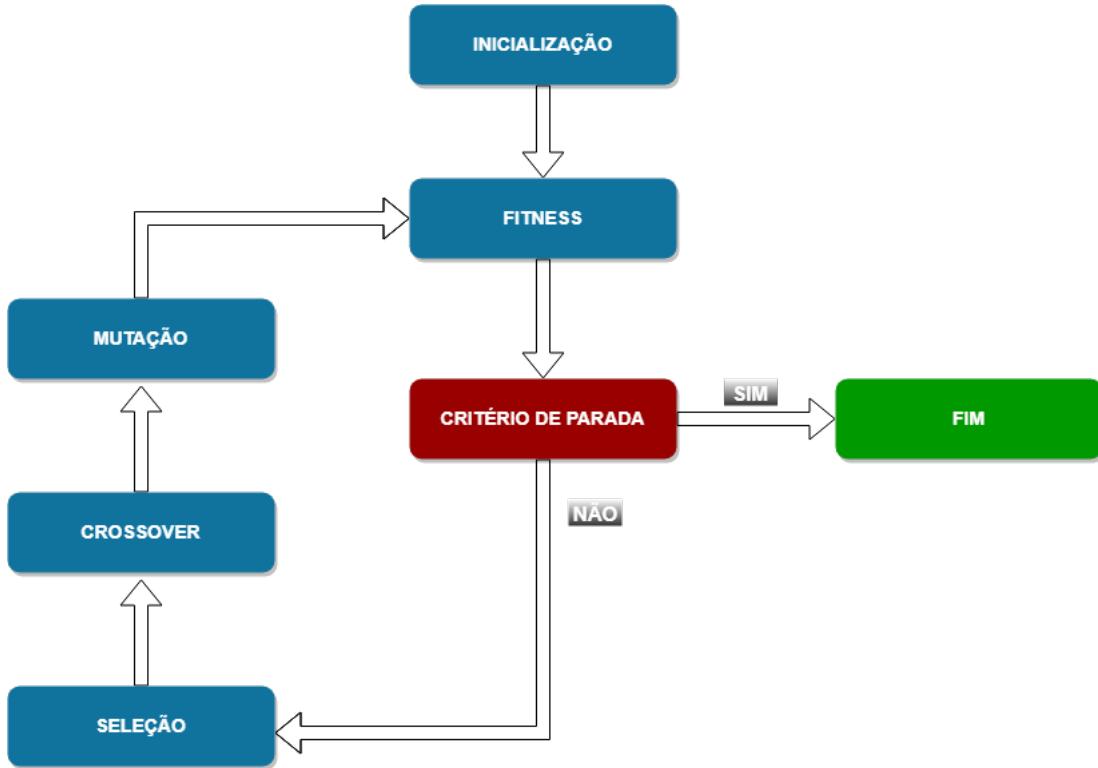


Figura 5.7: Estrutura de um Algoritmo Genético

A implementação do mesmo foi feita em python 2 seguindo o modelo de Troy Squillaci em seu *Github* [Sample code for an incredibly simple genetic algorithm 2018].

A população foi definida como a classe agent, esta continha um string que correspondia ao *threshold* e um valor *fitting*, gerado pela média do erro de translação ao executar o *rgbd_rtk* e o *development kit* do KITTI.

No passo de inicialização, foram escolhidos valores para a população baseados em uma distribuição normal gaussiana. Esse modelo foi escolhido devido à sua importância na estatística inferencial. Corresponde a um modelo de distribuição centrado em determinada média e possui frequência pequena de valores muito extremos (popularizada pelo alemão Karl Gauss).

O fitness da população consiste na etapa de executar o *rgbd_rtk* e calcular a média do erro de translação apenas para valores que não foram ainda classificados, ou seja, valores que ainda não têm a média do erro de translação calculada. Neste ponto, é essencial destacar que, na primeira geração, nenhum valor foi classificado, no entanto nas gerações seguintes, somente os filhos e agentes que sofreram mutação é que passarão novamente por esse processo de *fitness*.

A seleção representa o quanto da população será mantida, para nosso projeto escolhemos manter 40% (com base nos testes realizados concluímos que esse valor seria aceitável para diminuir o tempo do algoritmo e ainda assim convergir para a função de erro que tentamos minimizar).

O processo de *crossover* consiste em gerar novos membros da população de modo a manter o número de população inicial. Para nosso modelo 60% da população corresponde

aos filhos gerados. Cada par de agentes gera um par de filhos, esse algoritmo considera que a população é hermafrodita. O primeiro filho é gerado pela influência do primeiro agente escolhido (aleatoriamente) somado com a diferença entre os valores dos agentes pais. Já o segundo filho, é gerado pela influência do segundo agente escolhido (aleatoriamente) subtraída a diferença entre os valores dos agentes pais. O cálculo para cada um dos filhos foi pensada para que cada um fosse mais influenciado por um pai distinto, essa metodologia foi determinada empiricamente.

O passo seguinte, mutação, consiste em escolher um novo valor de *threshold* para uma parcela populacional (no caso, 10%), valor utilizado por questões empíricas.

Por fim, ordenamos a população criada e, para cada geração, escrevemos o resultado do indivíduo (valor de *threshold*) mais capacitado (gerou menor valor de erro médio de translação) em um arquivo de texto.

Capítulo 6

Experimentos

Neste capítulo faremos a descrição dos experimentos realizados, bem como a análise dos mesmos. Para ilustrar como está organizado este capítulo, veja a Figura 6.1 abaixo:

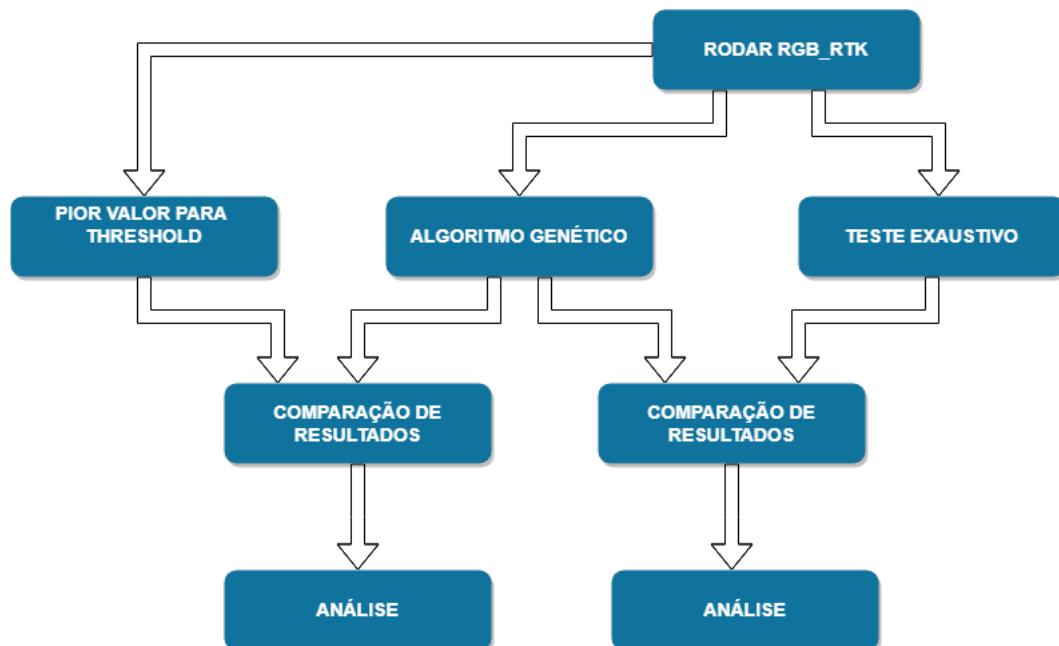


Figura 6.1: Experimentos Realizados

6.1 Método para parametrização: teste exaustivo

Para o teste exaustivo executamos o projeto *rgbd_rtk* com faixa de valores de *threshold* distintos para os *datasets* de odometria da KITTI 01, 03, 04 e 06.

Abaixo é possível ver Tabelas (6.1, 6.2, 6.3 e 6.4) com o valor do *threshold*, dado em centímetros (cm), testado e a média do erro de translação, as faixas para teste foram escolhidas empiricamente.

<i>Dataset 01</i>	faixa de 35 a 61, passo 1
<i>Threshold cm</i>	Média do erro(translação)
35	0,680360492707
36	0,682681625608
37	0,67153423825
38	0,664578220421
39	0,668938773096
40	0,667645364668
41	0,663899418152
42	0,666132756888
43	0,656060771475
44	0,654452009724
45	0,653825667747
46	0,648346435981
47	0,64083771637
48	0,64695128363
49	0,640318580227
50	0,6369213047
51	0,637095636953
52	0,639500155592
53	0,644220975689
54	0,650794424635
55	0,649462544571
56	0,639998823339
57	0,653786105348
58	0,666303252836
59	0,661459160454
60	0,663025562399
61	0,664080820097

Tabela 6.1: Resultados *Dataset 01* - Teste Exaustivo

<i>Dataset 06</i>	faixa de 10 a 40, passo 1
<i>Threshold cm</i>	Média do erro(translação)
10	0,128977715789
11	0,105469784211
12	0,125930736842
13	0,107874073684
14	0,106495317544
15	0,103674650877
16	0,105194866667
17	0,104834342105
18	0,105053150877
19	0,100506940351
20	0,100473026316
21	0,0997277157895
22	0,0987494298246
23	0,106112110526
24	0,101334078947
25	0,1056062
26	0,108457
27	0,112953864912
28	0,110752838596
29	0,113861338596
30	0,113531285965
31	0,116983989474
32	0,118218057895
33	0,118754107018
34	0,125803936842
35	0,126506966667
36	0,130194114035
37	0,134998324561
38	0,129784736842
39	0,131451392982
40	0,136954812281

Tabela 6.2: Resultados *Dataset 06* - Teste Exaustivo

<i>Dataset 03</i>	faixa de 1 a 20, passo 1
<i>Threshold cm</i>	Média do erro(translação)
1	0,279230423913
2	0,190776423913
3	0,160693570652
4	0,11382811413
5	0,0954085054348
6	0,0903409456522
7	0,0868536956522
8	0,0834769945652
9	0,0755402663043
10	0,067542076087
11	0,0646009347826
12	0,072557923913
13	0,0894150978261
14	0,0864954293478
15	0,0954452065217
16	0,0967900597826
17	0,100208358696
18	0,127892157609
19	0,136055831522
20	0,150038494565

<i>Dataset 04</i>	faixa de 10 a 25, passo 1
<i>Threshold cm</i>	Média do erro(translação)
10	0,243136930233
11	0,229404697674
12	0,222508790698
13	0,211602232558
14	0,20671244186
15	0,201895046512
16	0,195092348837
17	0,193545860465
18	0,189673930233
19	0,188641906977
20	0,197468930233
21	0,200829023256
22	0,204626627907
23	0,202774186047
24	0,206865930233
25	0,202600813953

Tabela 6.4: Resultados *Dataset 04* - Teste Exaustivo

Tabela 6.3: Resultados *Dataset 03* - Teste Exaustivo

Além disso, os mesmos dados foram representados por meio de Gráficos 6.2, 6.3, 6.4 e 6.5 (salientando que o *threshold* é dado em centímetros):

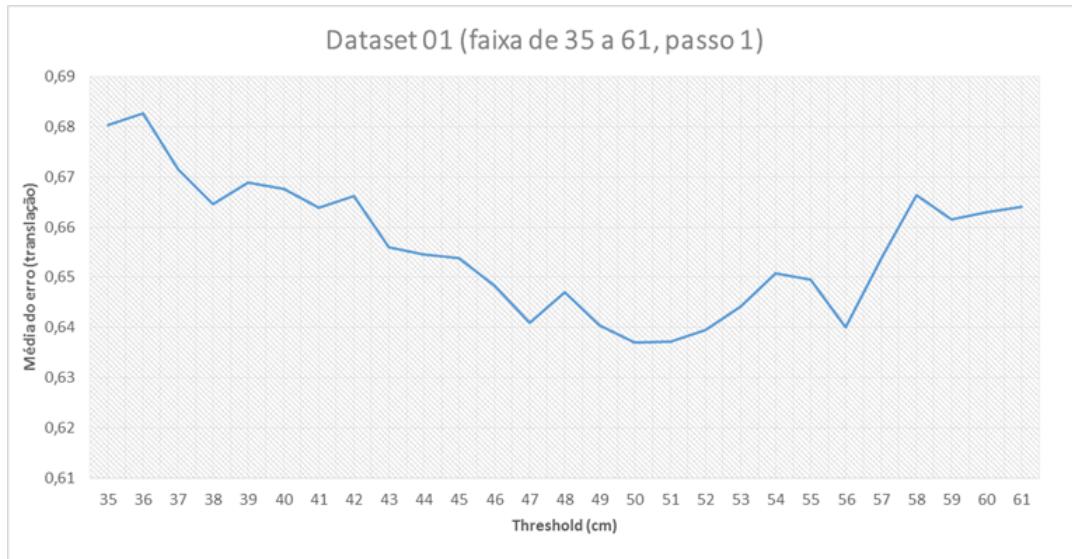


Figura 6.2: Gráfico de erro de trajetória para sequência 01, dado faixa de *threshold* de 35 a 61 (cm), passo 1

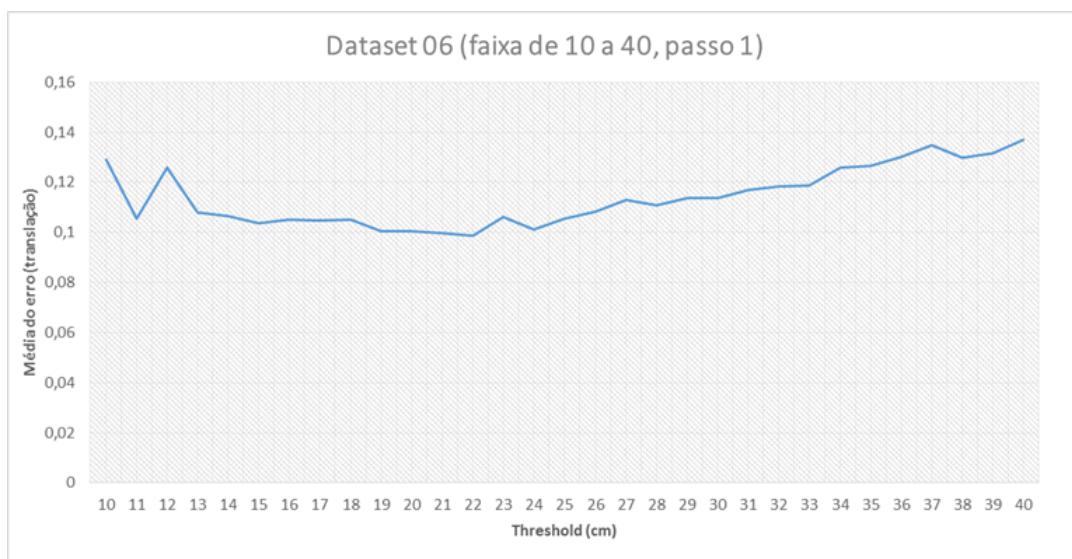


Figura 6.3: Gráfico de erro de trajetória para sequência 06, dado faixa de *threshold* de 10 a 40 (cm), passo 1

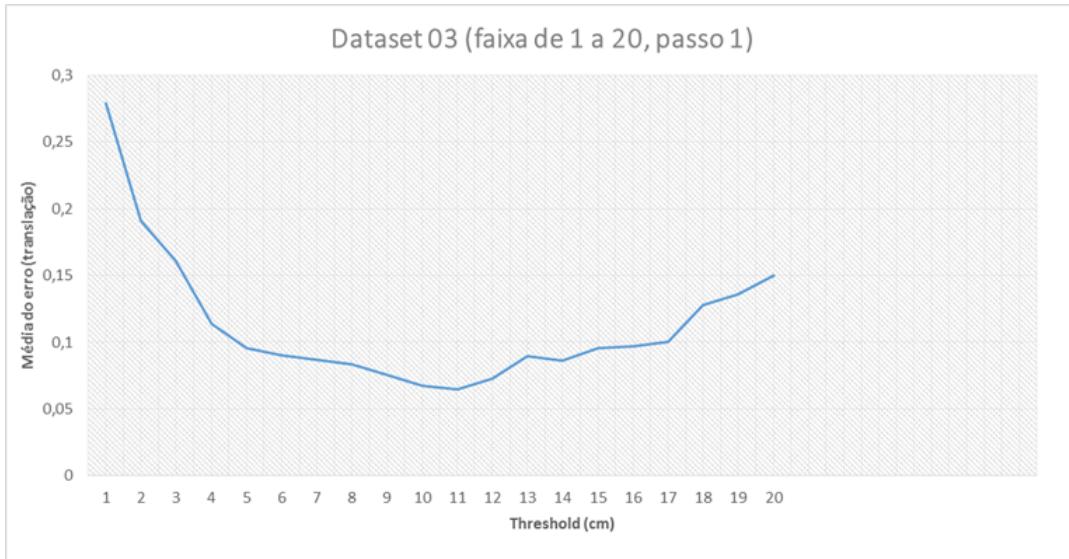


Figura 6.4: Gráfico de erro de trajetória para sequência 03, dado faixa de *threshold* de 1 a 20 (cm), passo 1

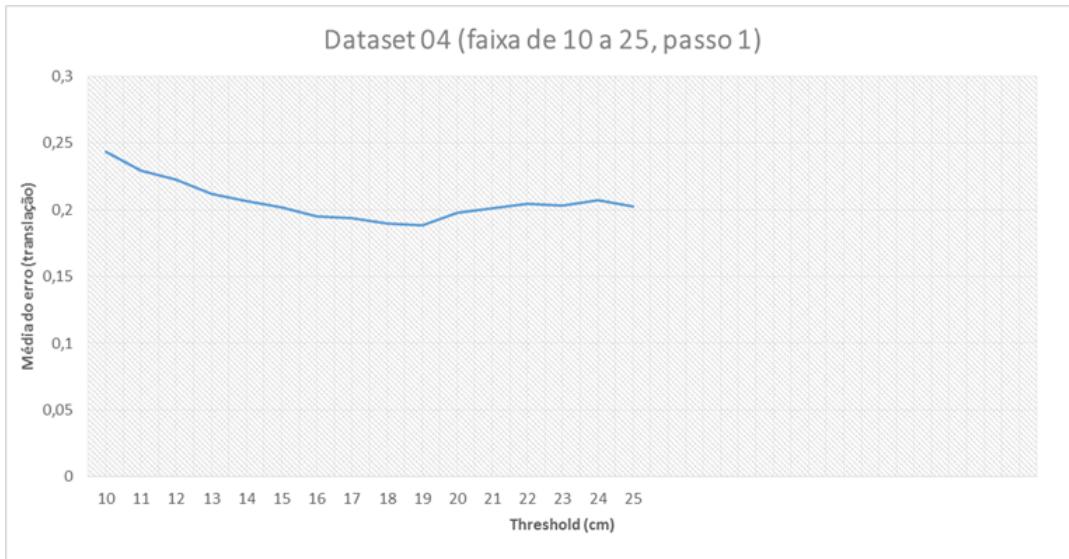


Figura 6.5: Gráfico de erro de trajetória para sequência 04, dado faixa de *threshold* de 10 a 25 (cm), passo 1

6.2 Método para parametrização: algoritmo genético

Para o teste em algoritmo genético executamos o projeto *rgbd_rtk* para os *datasets* de odometria da KITTI 01, 03, 04 e 06. A metodologia para tanto, discutida no Capítulo 5, consistiu em gerar uma população inicial de x indivíduos, com distribuição normal gaussiana em torno de média m e desvio padrão d . Para cada geração, salvamos em

um arquivo de texto, o indivíduo cujo *threshold* resultou na menor média do erro de translação.

Abaixo, é possível ver nas Tabelas 6.5, 6.6, 6.7 e 6.8 o indivíduo mais apto por geração, a população foi de 10 indivíduos por geração, os valores de média e desvio foram escolhidos empiricamente.

<i>Dataset 01</i>	média 52, desvio 10
<i>Threshold (cm)</i>	Média do erro (translação)
49,8855209687	0,635082867099
49,8855209687	0,635082867099
49,8855209687	0,635082867099
49,8118231238	0,632428748784
49,8118231238	0,632428748784

Tabela 6.5: Resultados *Dataset 01* - Algoritmo Genético

<i>Dataset 06</i>	média 23, desvio 10
<i>Threshold (cm)</i>	Média do erro (translação)
21,0026484588	0,0996869736842
21,0026484588	0,0996869736842
21,0026484588	0,0996869736842
20,5362424457	0,0986120947368
20,5362424457	0,0986120947368

Tabela 6.6: Resultados *Dataset 06* - Algoritmo Genético

<i>Dataset 03</i>	média 5, desvio 5
<i>Threshold (cm)</i>	Média do erro (translação)
6,79347974199	0,0823417065217
8,50717421481	0,075938625
8,50717421481	0,075938625
10,2208686876	0,0652939021739
10,2208686876	0,0652939021739

Tabela 6.7: Resultados *Dataset 03* - Algoritmo Genético

<i>Dataset 04</i>	média 20, desvio 5
<i>Threshold (cm)</i>	Média do erro (translação)
18,2945042908	0,189460418605
18,2656038554	0,188787069767
18,23670342	0,187898093023
18,23670342	0,187898093023
18,23670342	0,187898093023

Tabela 6.8: Resultados *Dataset 04* - Algoritmo Genético

Além disso, os mesmos dados foram representados por meio de Gráficos 6.6, 6.7, 6.8 e 6.9 (salientando que o *threshold* é dado em centímetros):

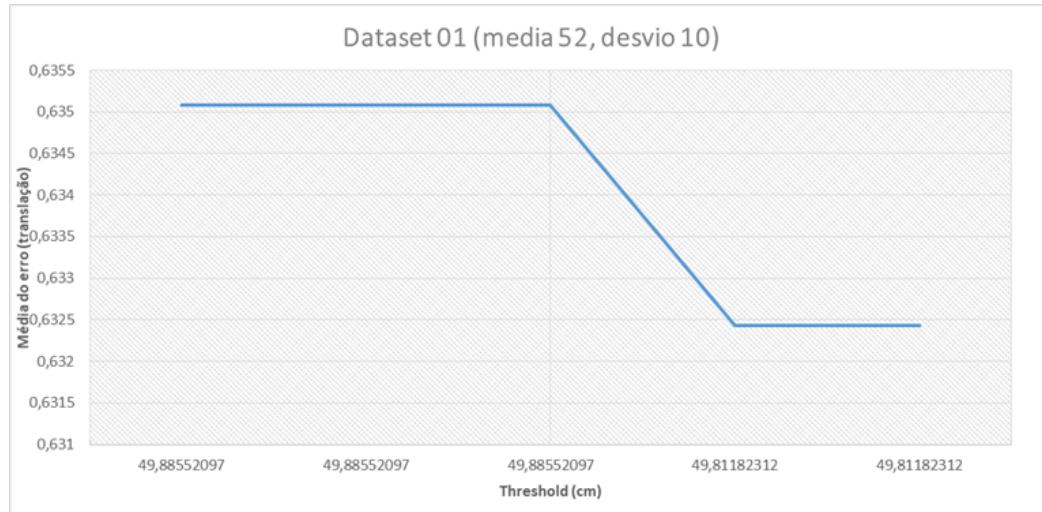


Figura 6.6: Gráfico de erro de trajetória para sequência 01, dado média 52 e desvio 10

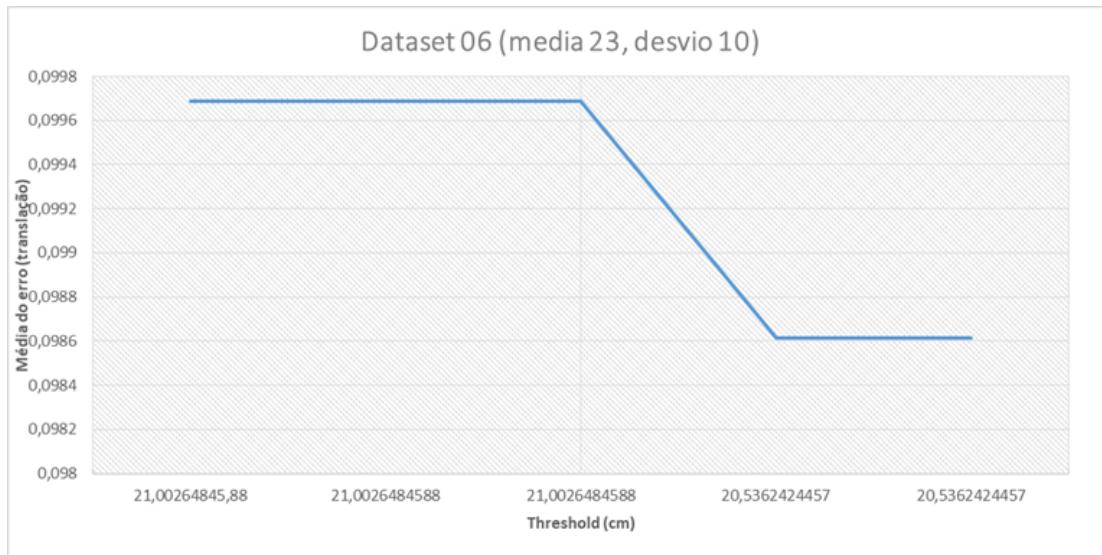


Figura 6.7: Gráfico de erro de trajetória para sequência 06, dado média 23 e desvio 10

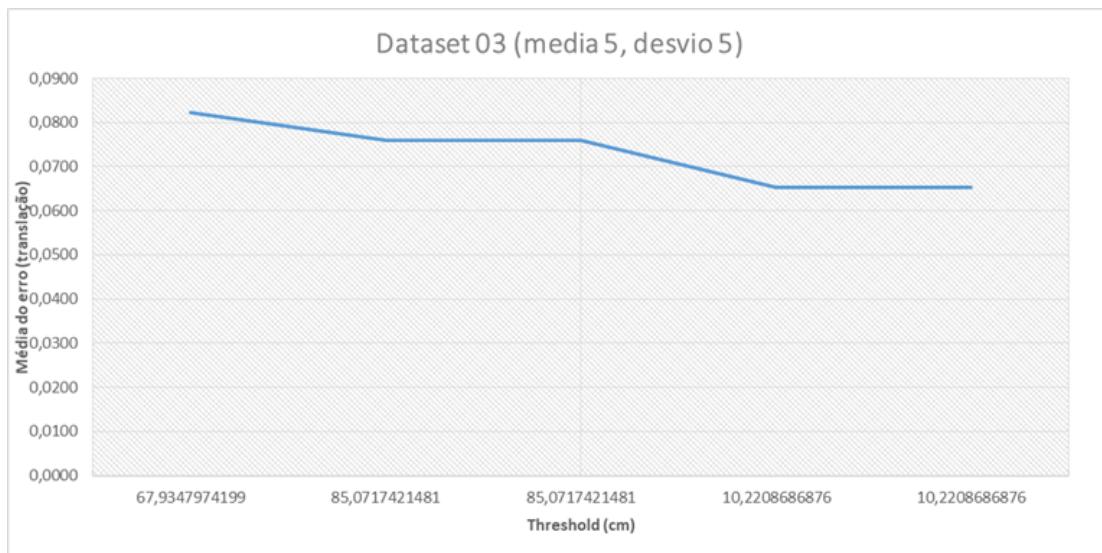


Figura 6.8: Gráfico de erro de trajetória para sequência 03, dado média 5 e desvio 5

6.3. COMPARAÇÃO DE ENTRE ALGORITMO GENÉTICO E TESTE EXAUSTIVO 33

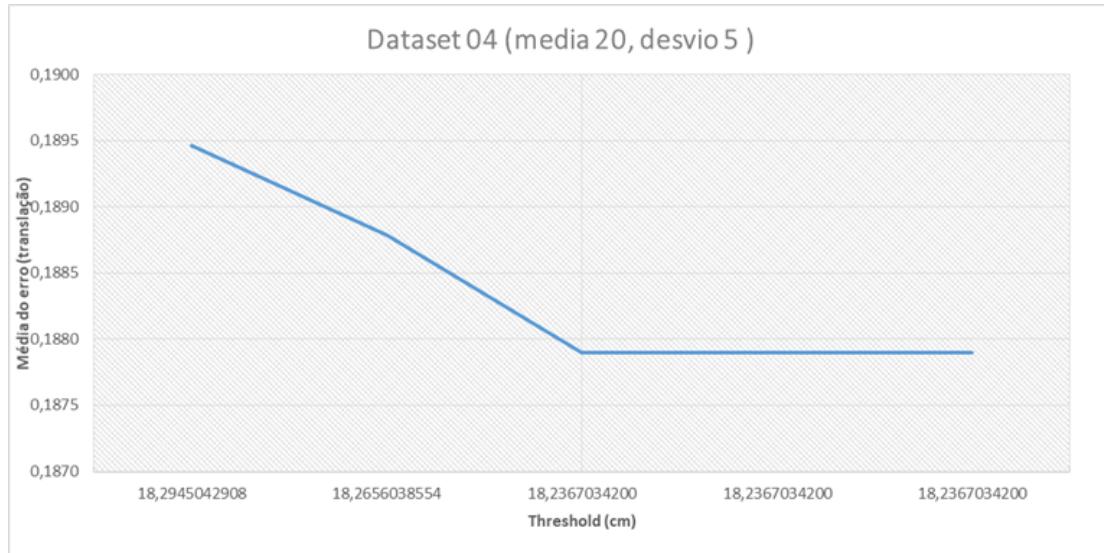


Figura 6.9: Gráfico de erro de trajetória para sequência 04, dado média 20 e desvio 5

6.3 Comparação de entre algoritmo genético e teste exaustivo

Nesta seção, apresentamos o caminho traçado pelo *rgbd_rtk* versus o caminho verdadeiro (do inglês, *ground truth*), em outras palavras, nas Figuras 6.10, 6.11, 6.12, 6.13, temos o erro de trajeto das sequências 01, 03, 04 e 06. Inserimos as imagens geradas pelo *threshold* que resultou no menor erro do algoritmo genético e do teste exaustivo, respectivamente.

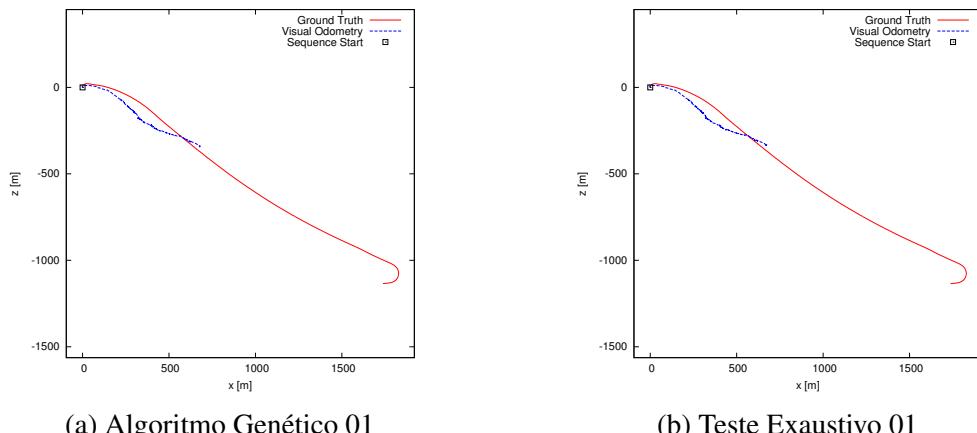


Figura 6.10: Erro de trajetória da sequência 01: Algoritmo Genético vs Teste Exaustivo

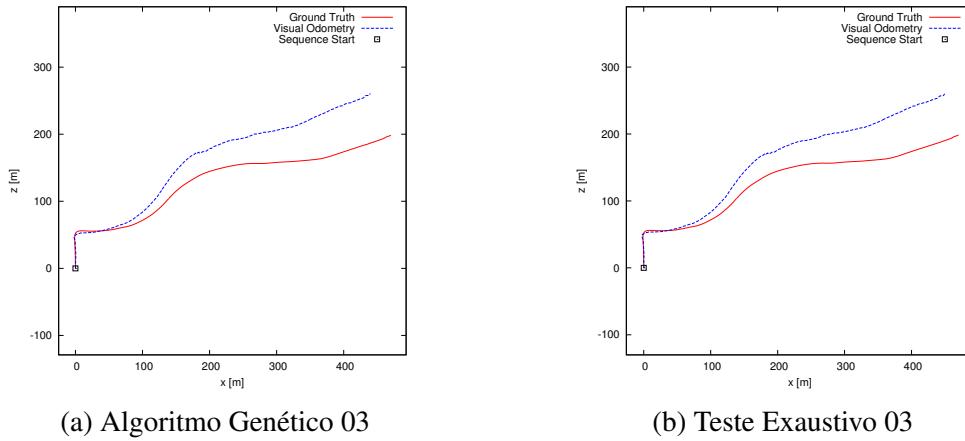


Figura 6.11: Erro de trajetória da sequência 03: Algoritmo Genético vs Teste Exaustivo

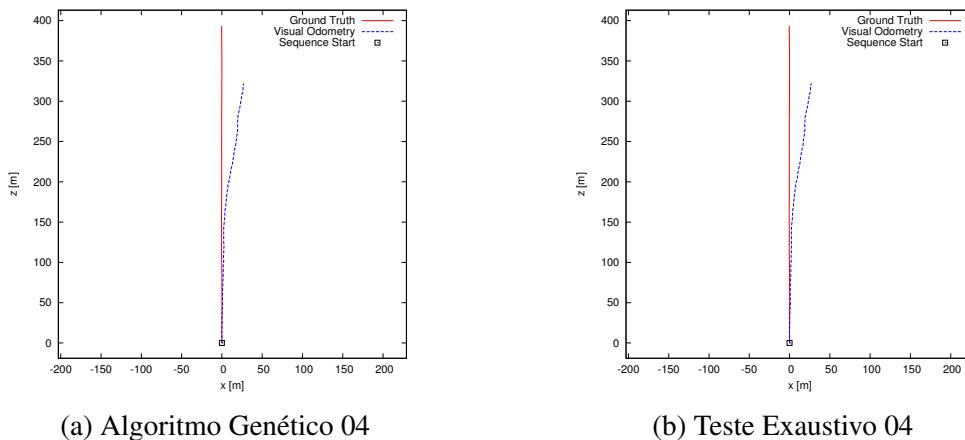


Figura 6.12: Erro de trajetória da sequência 04: Algoritmo Genético vs Teste Exaustivo

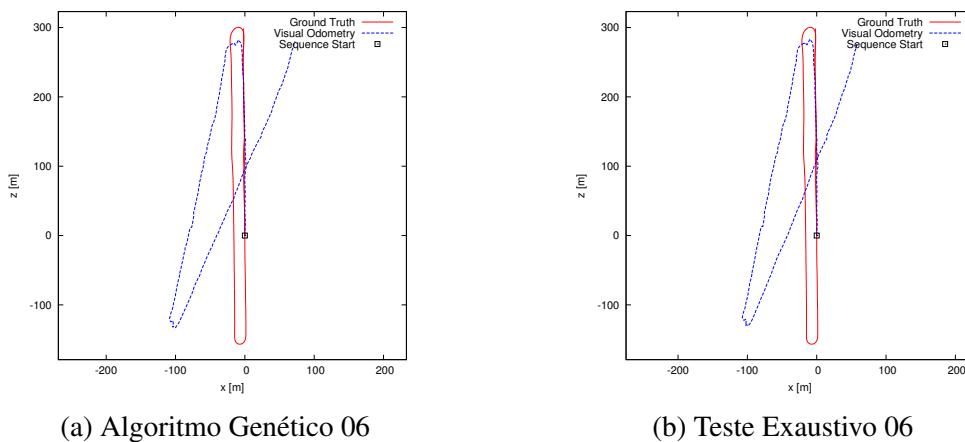


Figura 6.13: Erro de trajetória da sequência 06: Algoritmo Genético vs Teste Exaustivo

6.4 Análise de Resultados

Analizando os resultados apresentados, podemos perceber que não houve tanta diferença entre o erro gerado pelo *threshold* escolhido pelo teste exaustivo e pelo algoritmo genético.

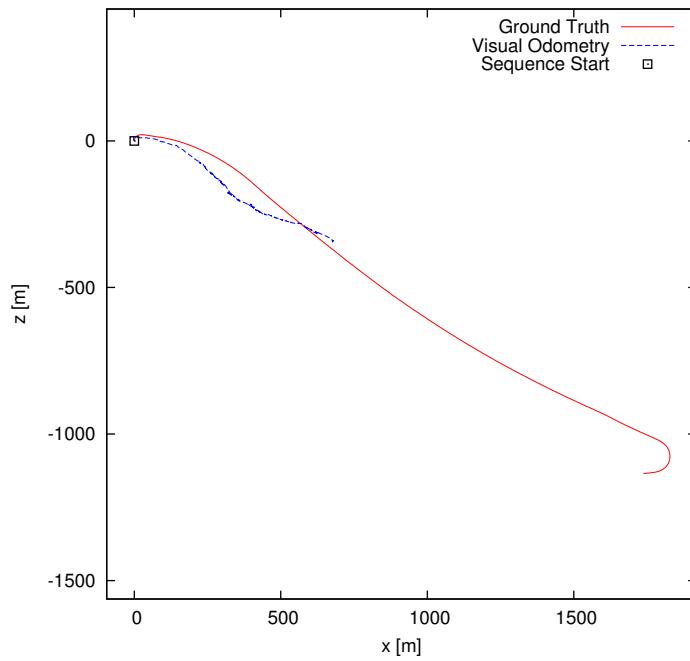
No entanto, cabe ressaltar que o teste exaustivo é restringido a uma faixa de valores específica, com passo fixo. Ou seja, caso o valor ótimo de *threshold* não esteja contido nessa faixa, não será possível encontrá-lo. Veja, por exemplo, o resultado dos erros calculados para faixa de 0.5 a 10.5 (cm) com passo 1 para a sequência 01 (Tabela 6.9):

<i>Dataset 01</i>	faixa de 0.5 a 10.5, passo 1
<i>Threshold cm</i>	Média do erro (translação)
0.5	0,926811753647
1.5	0,873590789303
2.5	0,843967126418
3.5	0,819378688817
4.5	0,803934641815
5.5	0,797938066451
6.5	0,794576824959
7.5	0,790058658023
8.5	0,781211175041
9.5	0,775006259319
10.5	0,76917580389

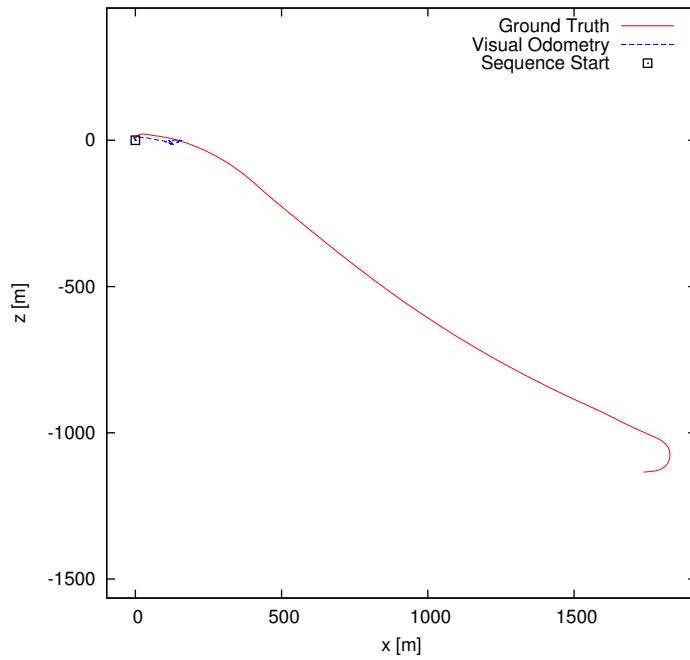
Tabela 6.9: Resultado *Dataset 01*: Faixas de 0.5 a 10.5 com passo 1 para a sequência 01

Caso, o teste feito tivesse compreendido essa faixa, não teríamos atingido o menor erro possível para essa sequência. Nesse aspecto, o algoritmo genético se destaca, pois, nele, é possível iterar até que atingido um critério de parada ou um número limite de iterações. Esse método não é restrito a um número fixo de valores imutáveis.

Outrossim, é importante observar o impacto que a escolha do *threshold* tem sobre o projeto de odometria visual com imagens estéreo. Isso pois, ao compararmos o erro de trajetória entre os valores de *threshold* que resultam no maior erro testado com os valores escolhidos pelos métodos (teste Exaustivo ou Algoritmo Genético) vemos que a diferença é bastante perceptível (podemos ver diferença a olho nu). Para exemplificação, ilustramos esse aspecto com as Figuras 6.14, 6.15, 6.16, 6.17. Nela vemos o erro de trajeto das sequências 01, 03, 04 e 06 das imagens geradas pelo *threshold* que resultou no menor erro do algoritmo genético *versus* o *threshold* que resultou no maior erro dentre todos os testes, respectivamente.

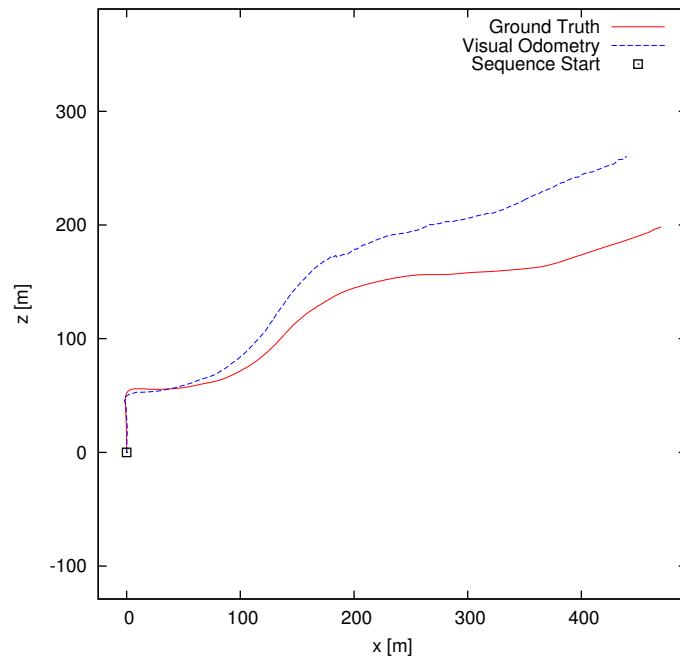


(a) Algoritmo Genético 01

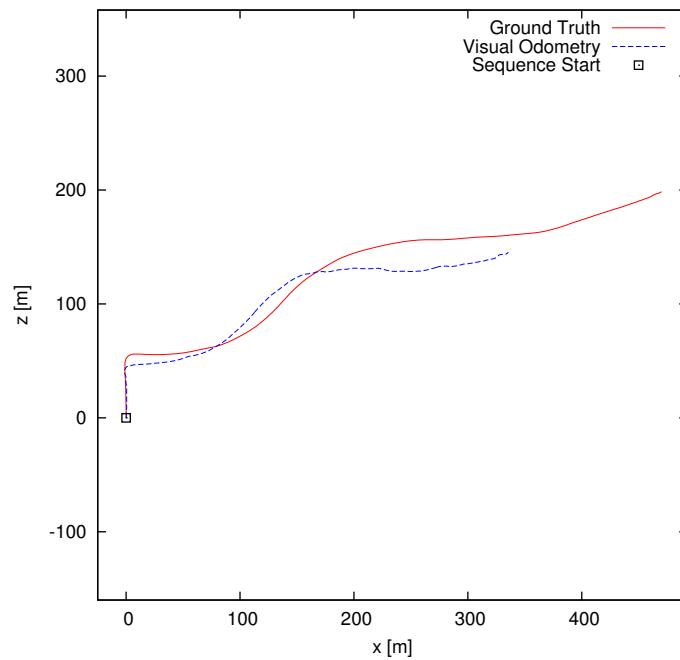


(b) Maior erro 01

Figura 6.14: Erro de trajetória da sequência 01: Algoritmo Genético vs *Threshold* de maior erro

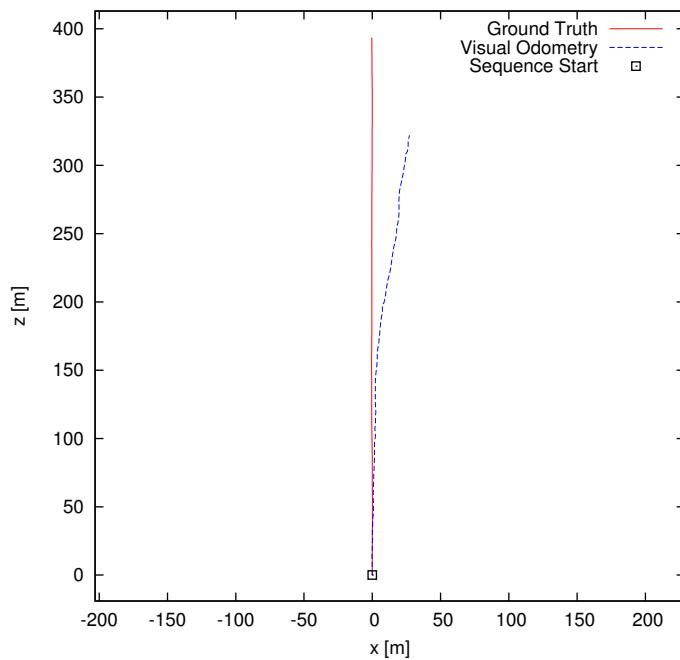


(a) Algoritmo Genético 03

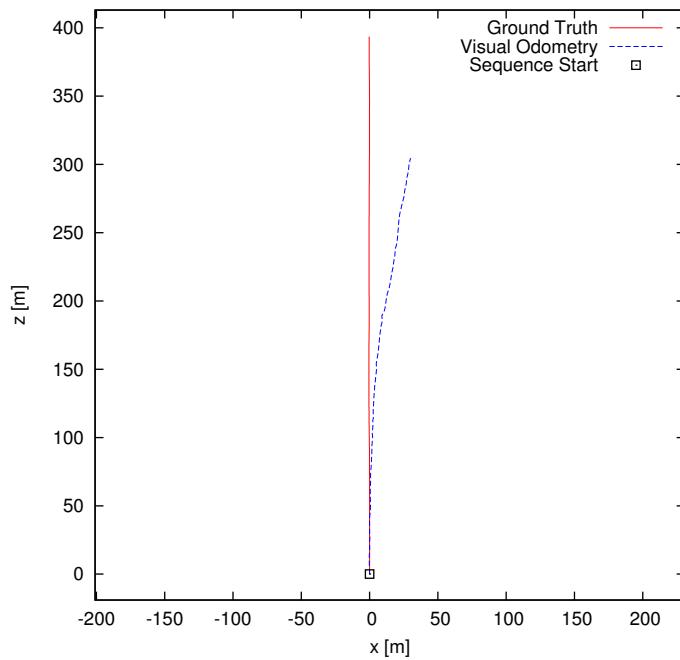


(b) Maior erro 03

Figura 6.15: Erro de trajetória da sequência 03: Algoritmo Genético vs *Threshold* de maior erro

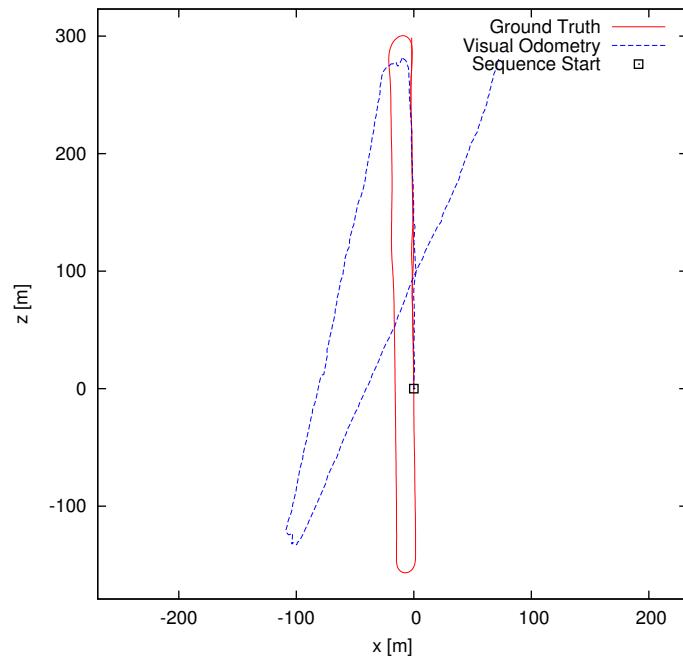


(a) Algoritmo Genético 04

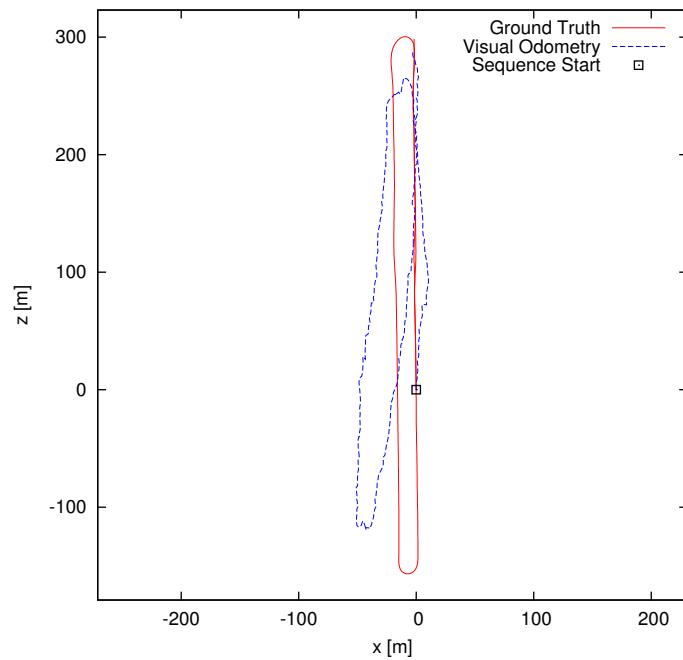


(b) Maior erro 04

Figura 6.16: Erro de trajetória da sequência 04: Algoritmo Genético vs *Threshold* de maior erro



(a) Algoritmo Genético 06



(b) Maior erro 06

Figura 6.17: Erro de trajetória da sequência 06: Algoritmo Genético vs *Threshold* de maior erro

6.5 Expliação sobre cada *Dataset*

Nesta seção, explicaremos o porquê do erro de algumas sequências serem maiores do que outras.

A sequência 01, possui o maior erro, pois é muito dinâmica, devido a passagem de ciclistas, carros, entre outros. Além de possuir altas variações de iluminação. Fatores para os quais, conforme especificado em capítulos anteriores, o projeto implementado não possui robustez. Além disso, é valido salientar que nas últimas 100 imagens há ofuscamento de uma das câmeras pelo sol, fato que resultou em erro ao gerar a disparidade. Por isso, removemos as 100 últimas imagens do *dataset* para teste.

A sequência 03 é pouco dinâmica, contudo, embora não haja movimentação brusca, o automóvel do KITTI segue um outro carro durante o início do trajeto e depois para de seguí-lo. Essa mudança gera um aumento de erro no ponto onde ela ocorre. Além disso, é perceptível o acúmulo do erro gerado pelo *rgbd_rtk*.

Na sequência 04 (pouco dinâmica) o automóvel do KITTI segue um outro carro durante todo o trajeto. Assim como em todas as outras sequências, há o acúmulo do erro gerado pelo *rgbd_rtk*.

A sequência 06 também com baixa dinamicidade, possui algumas curvas conforme podemos perceber ao observar a trajetória traçada.

Em alguns resultados apresentados, observamos que a trajetória estimada pelo nosso projeto não segue a linha da trajetória real. Esse fenômeno ocorre devido à falta de robustez do algoritmo a dinamicidades. Por vezes, o movimento externo é percebido como sendo o movimento do próprio robô, por isso, a trajetória aparece "encurtada".

Por fim, vemos que a escolha do *threshold* do RANSAC teve impacto significativo no erro de trajetória, mesmo com as limitações do nosso projeto para *datasets* dinâmicos.

Capítulo 7

Conclusão

Partindo do objetivo de analisar o efeito da parametrização do *Random Sample Consensus* no projeto de odometria visual em ambientes dinâmicos, usando imagens estéreo, verificou-se que um aspecto muito relevante é a escolha do limiar do RANSAC, no qual o valor limite foi determinado pelo uso de algoritmo genético e de teste exaustivo. Isso, pois o *RANdom SAmple Consensus* (RANSAC) é altamente utilizado no campo da odometria visual. No entanto, seu valor limiar é fixo e é empiricamente escolhido pela maioria dos projetos que utilizam esse método. Observou-se, com base nos resultados dos experimentos, que a escolha do *threshold* do RANSAC teve impacto significativo no erro de trajetória, mesmo com as limitações do nosso projeto para *datasets* dinâmicos.

Mesmo que a diferença entre o erro gerado pelo *threshold* escolhido pelo teste exaustivo e pelo algoritmo genético não seja grande. Pudemos verificar as deficiências que uma abordagem por teste exaustivo apresenta (restringe a uma faixa de valores específica, com passo fixo). Ou seja, caso o valor ótimo de *threshold* não esteja contido nessa faixa, não será possível encontrá-lo. Nesse aspecto, o algoritmo genético se destaca, pois, nele, é possível iterar até que seja atingido um critério de parada ou um número limite de iterações. Esse método, não é restrito a um número fixo de valores imutáveis.

Outrossim, vemos que a escolha do limiar tem suma importância sobre o projeto de odometria visual com imagens estéreo. Isso foi evidenciado pelas comparações feitas entre o erro de trajetória gerado pelos valores de *threshold* que resultam no maior erro testado e gerado pelos valores escolhidos pelos métodos (teste Exaustivo ou Algoritmo Genético).

Por isso, conclui-se que é altamente recomendável, automatizar a parametrização da odometria visual, escolhendo o limiar que minimize o erro, e buscar soluções de aprendizado de máquinas. No entanto, ainda há muito o que fazer para que nosso projeto atinja a robustez a ambientes dinâmicos com iluminação inconstante.

Referências Bibliográficas

algoritmos genéticos: conceitos básicos (2017), ftp://ftp.dca.fee.unicamp.br/pub/docs/vonzuben/theses/pvargas_mest/arq_11.pdf. Accessed: 2018-10-14.

Andrade, J. Bittencourt de. (1998), Ed. SBEE. Curitiba, Brasil.

CADENA, C.; CARLONE, L.; CARRILLO H.; LATIF Y.; SCARAMUZZA D.; NEIRA J.; REID I.; LEONARD J. J. (2016), Past, present and future of simultaneous localization and mapping: toward the robust perception age., In: IEEE Transactions on Robotics.

Carros autônomos já começam a virar realidade (2018), https://www.correiobrasiliense.com.br/app/noticia/economia/2018/02/02/internas_economia,657350/carros-autonomos-ja-comecam-a-virar-realidade.shtml. Accessed: 2018-10-20.

Chang, C. & S. Chatterjee (1992), Quantization error analysis in stereo vision, em '[1992] Conference Record of the Twenty-Sixth Asilomar Conference on Signals, Systems Computers', pp. 1037–1041 vol.2.

Conrad. D.J. e Sampson, R.E. (1990), 3d range imaging sensors, Henderson, T.C.,(ed.) Traditional and Non-Traditional Robotic Sensors. NATO ASI Series, Springer-Verlag.

da Silva, B. Marques F., L. F. Maciel Correia, K. de Araújo Bezerra & L. M. Garcia Gonçalves (2017), Tracking spatially distributed features in klt algorithms for rgb-d visual odometry, em '2017 Workshop of Computer Vision (WVC)', pp. 67–72.

da Silva, B. Marques F. & L. M. Garcia Gonçalves (2015), Visual odometry and mapping for indoor environments using rgb-d cameras, Vol. 507, pp. 16–31.

Drews-Jr, Paulo (2008), Odometria visual.

Eli S. dos Santos, J. (2006), Semana acadêmica de projetos e instalações elétricas: Minicurso de introdução à instrumentação.

Geiger, Andreas, Philip Lenz & Raquel Urtasun (2012), Are we ready for autonomous driving? the kitti vision benchmark suite, em 'Conference on Computer Vision and Pattern Recognition (CVPR)'.

- Grassi Júnior, Valdir. (2002), Sistema de visão omnidirecional aplicado no controle de robôs móveis.
- Guedes Maidana, R.; Tergolina Salton, A. e de Moraes Amory A. (2017), Odometria visual monocular para localizaÇÃo de robÔ terrestre, XIII Simpósio Brasileiro de Automação Inteligente Porto Alegre – RS, 1o – 4 de Outubro de 2017.
- Hosseini-Nejad, Zahra & Mehdi Nasri (2016), Image registration based on sift features and adaptive ransac transform.
- Kaehler, Adrian & Gary Bradski (2016), *Learning OpenCV 3: Computer Vision in C++ with the OpenCV Library*, 1st edio, O'Reilly Media, Inc.
- Leonard, J.; Durrant-Whyte, H. (1991), *Mobile robot localization by tracking geometric beacons*, Vol. 7, IEEE Transaction on.
- Martinelli, A. (2017), Modeling and estimating the odometry error of a mobile robot.
- Pedrosa, Antônio Ítalo Rodrigues (2014), Algoritmo genÉtico para identificaÇÃo do mÍnimo da funÇÃo peaks do matlab.
- Sample code for an incredibly simple genetic algorithm* (2018), <https://github.com/zivia/simple-ga>. Acesssed: 2018-10-20.
- Scaramuzza, D. (2011), Performance evaluation of 1-point-ransac visual odometry, Journal of Field Robotics.
- Scaramuzza, D. & F. Fraundorfer (2011a), Visual odometry: Part 1 - the first 30 years and fundamentals, em ‘Robotics and Automation’, IEEE Robotics and Automation Magazine 18(4).
- Scaramuzza, Davide & Friedrich Fraundorfer (2011b), ‘Visual odometry [tutorial]’, *IEEE Robot. Automat. Mag.* **18**, 80–92.
- Siscoutto, R. A., Szenberg F. Tori R. Raposo A. B. Celes W. Gattas M. (2004), Estereoscopia, C. Kirner e R. Tori(eds.), Realidade Virtual: Conceitos e Tendências - Livro do Pré-Simpósio SVR 2004, pp. 179–201.
- Sistemas Especialistas* (1996), <http://www.inf.ufsc.br/~alexandre.goncalves.silva/courses/14s2/ine5633/slides/aula1105.pdf>. Acesssed: 2018-7-26.
- Summary for IJRR 2013 paper* (2013), https://github.com/bostondiditeam/kitti/blob/master/Papers_Summary/Geiger2013IJRR/readme.md. Acesssed: 2018-09-20.
- Temba, P. (2000), Fundamentos da fotogrametria.
- Useful tools for the RGB-D benchmark* (2012), <https://vision.in.tum.de/data/datasets/rgbd-dataset/tools>. Accessed: 2018-05-17.

Visual Odometry (VO) (2016), http://www.cs.toronto.edu/~urtasun/courses/CSC2541/03_odometry.pdf. Acessed: 2018-06-04.

Visão Estéreo (2001), <http://www.comp.ita.br/~forster/CC-222/lecture/06-Visao-Estereo.pdf>. Acessed: 2018-08-07.

Zuliani, Marco (2012), Ransac for dummies.