

Análise do RANSAC sobre algoritmo de odometria visual com imagens estéreo

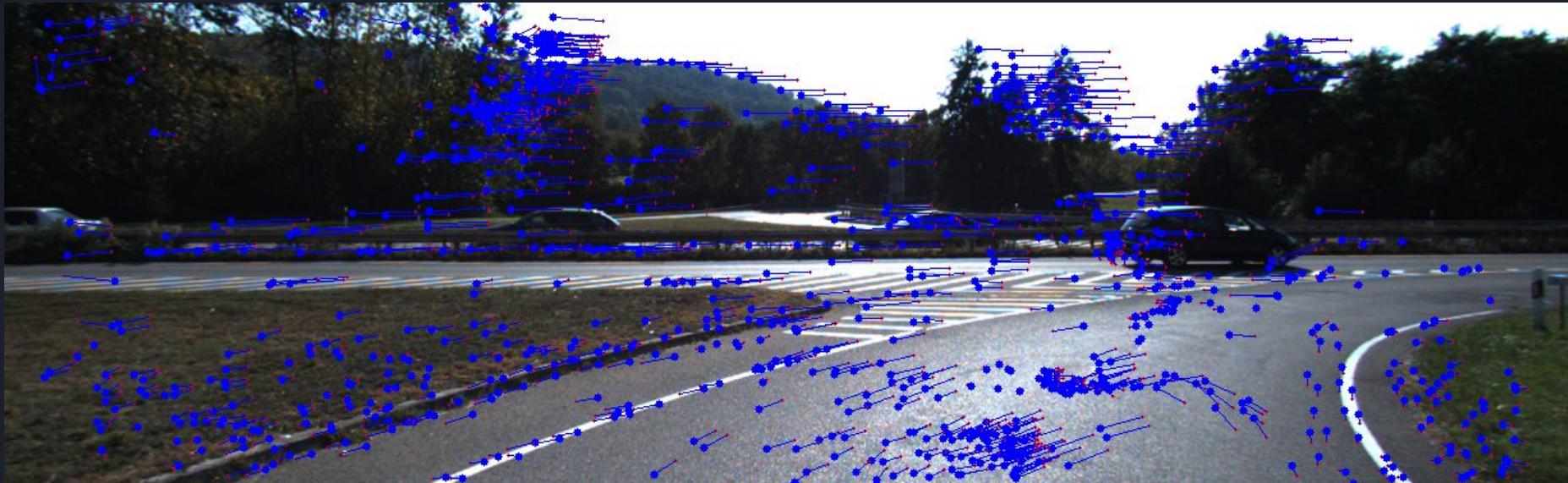
Vanessa Dantas de Souto Costa
vanessa.dantas796@gmail.com



INTRODUÇÃO

Contextualização com o mundo atual

- A odometria visual é um dos métodos mais usados para sistemas de navegação autônoma [Guedes Maidana 2017]
- Introduzida pela primeira vez para os robôs planetários operando em Marte - Moravec 1980, vide [Visual Odometry (VO) n.d.].





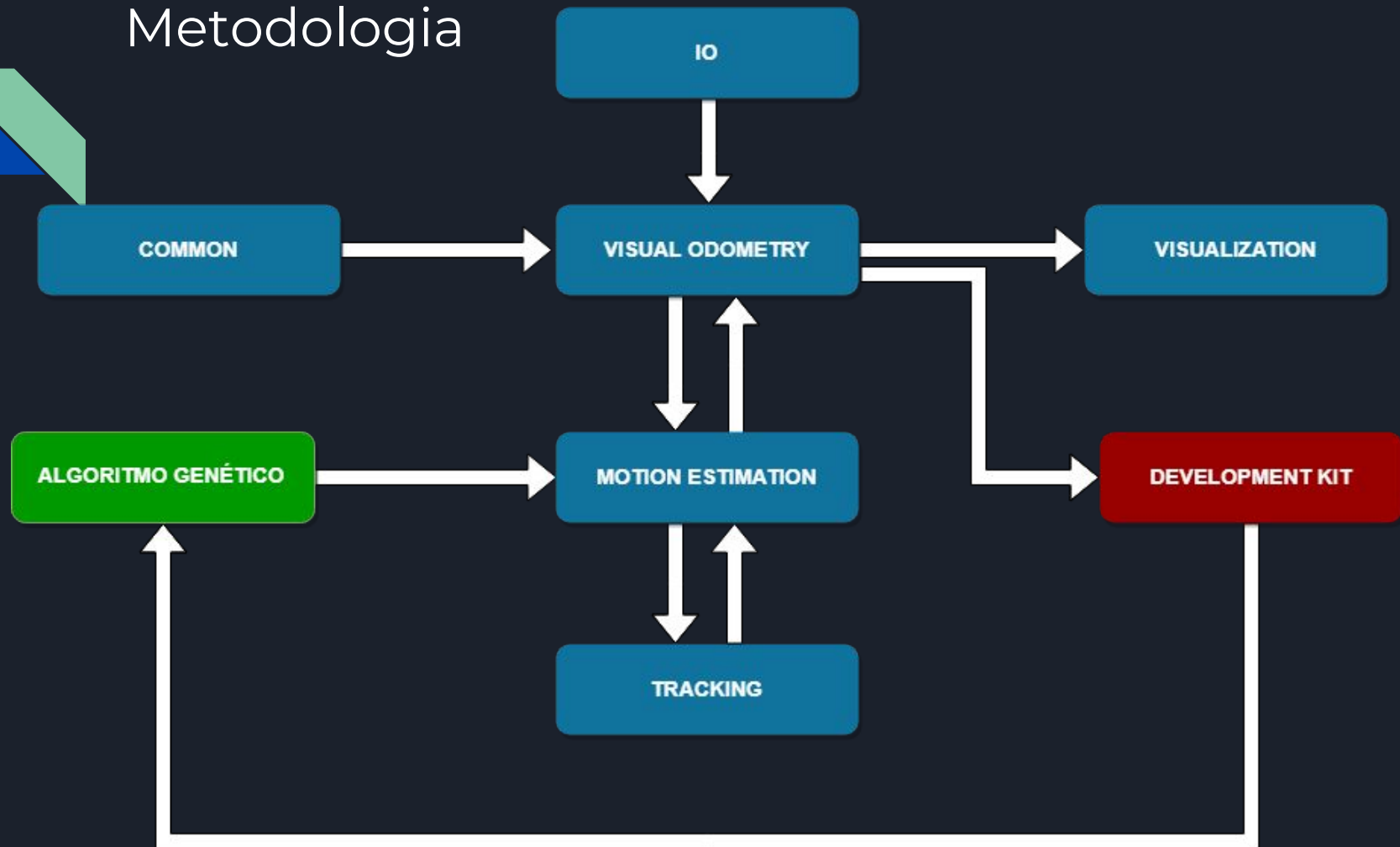
Motivação

- Pesquisa do Prof. Dr. Bruno Marques Ferreira da Silva.
- Ressalva: os algoritmos desenvolvidos não são robustos a ambientes dinâmicos.

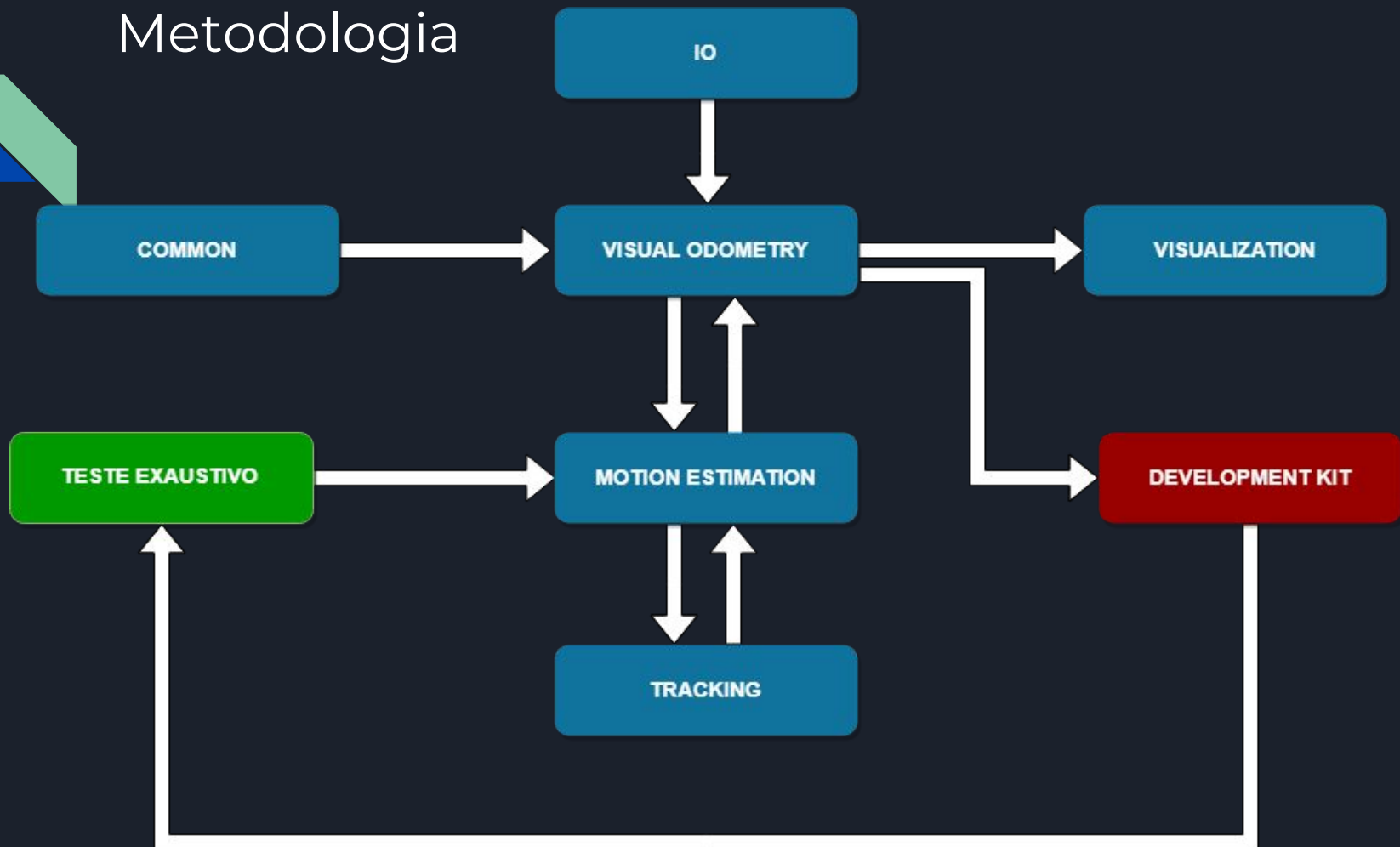
Objetivo

- 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].

Metodologia



Metodologia

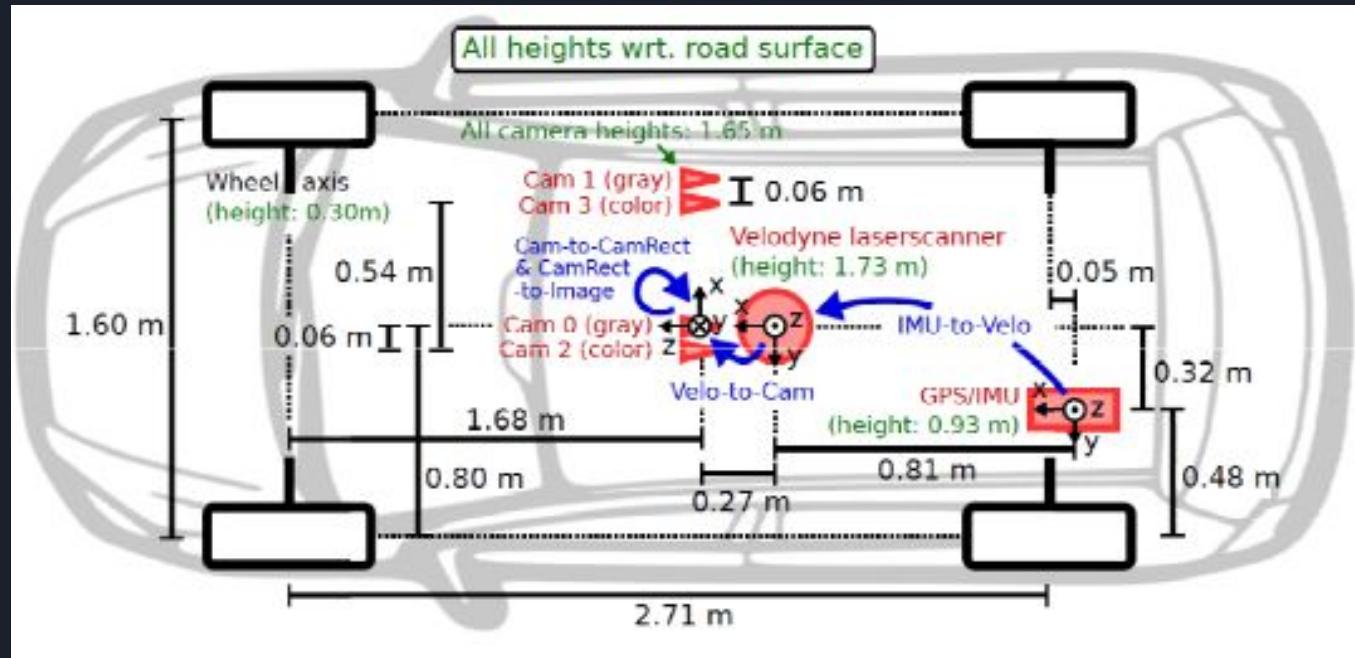




TEORIA

Sistema de Referência

- Disposição dos sensores no carro usado para construção do dataset KITTI.



Estéreo

- Visão estéreo:
- percepção de profundidade
- calculada por triangulação.
- ex.:



- Visão monocular:
- julgamento de profundidade:
- monoscópico.
- ex.:



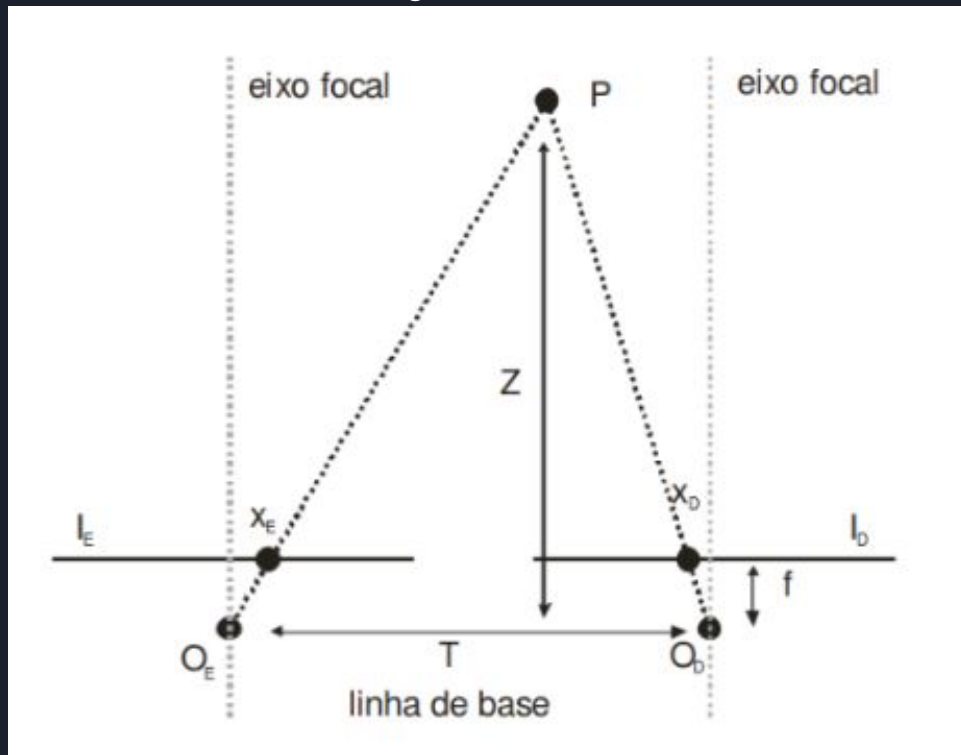
Estéreo

- a odometria visual estéreo, 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.

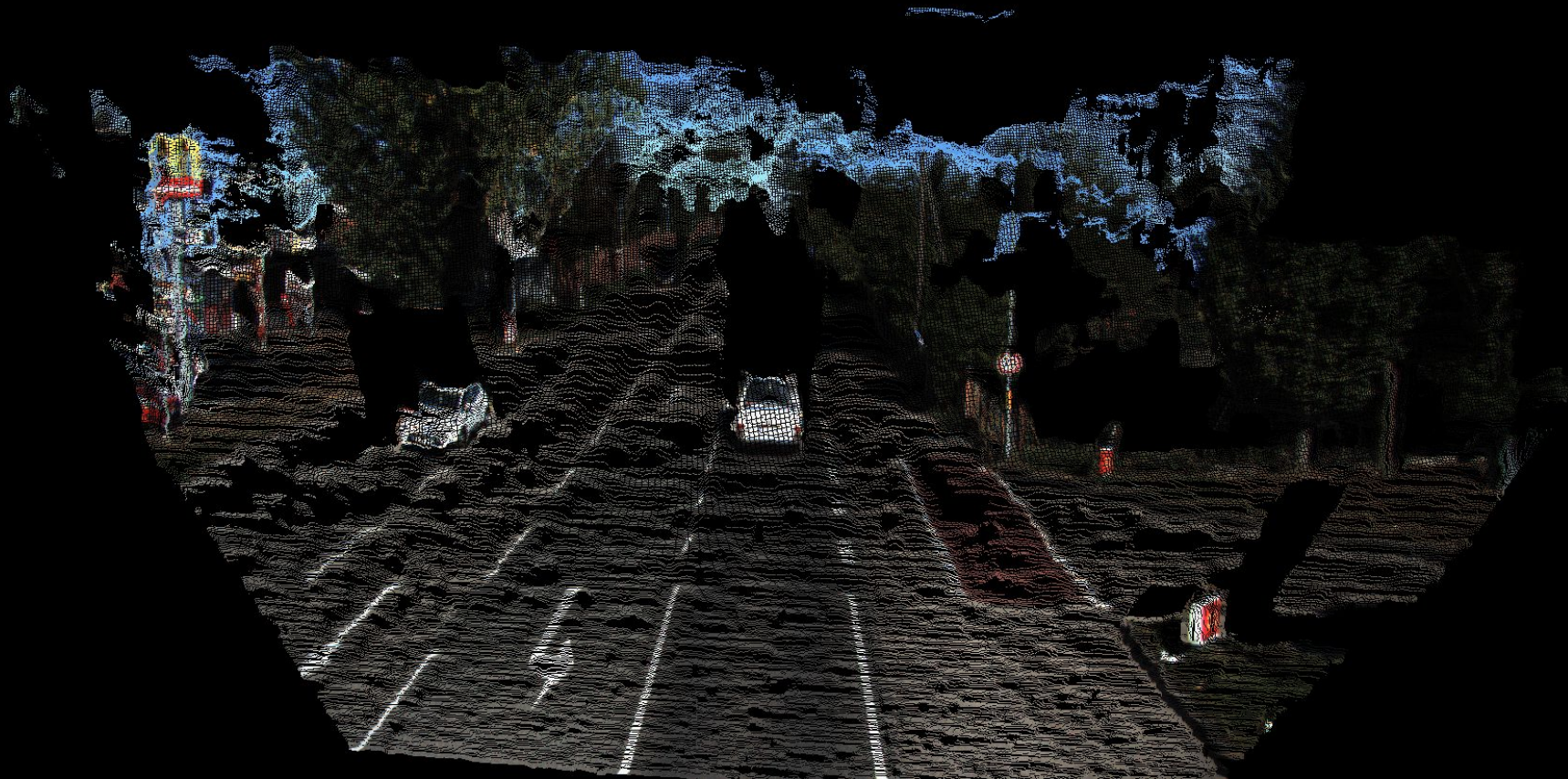


Correspondência

- A correspondência ou retificação da imagem consiste em projetá-la, segundo seu próprio feixe perspectivo para um plano horizontal. Em outras palavras, podemos modificar de modo a tentar eliminar os ângulos de atitude da câmera em relação a um dado referencial, bem como a distância focal da imagem,



Reconstrução (Projeção e Visão Tridimensional)

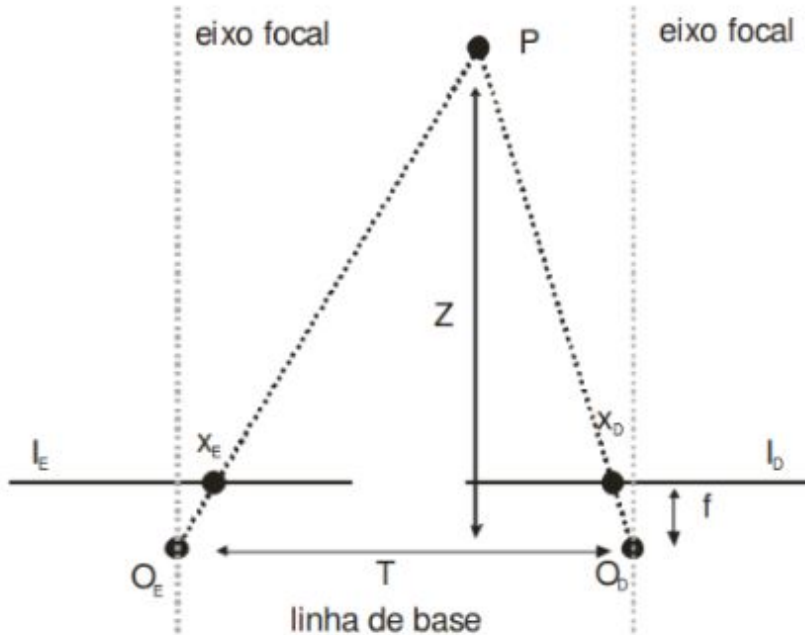


Reconstrução (Projeção e Visão Tridimensional)



Reconstrução (Projeção e Visão Tridimensional)

- Cálculo da Profundidade por Triangulação



$$\frac{T - disp}{Z - f} = \frac{T}{Z} \implies Z = \frac{f \cdot T}{disp}$$

- $disp$ 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 e Z é a profundidade.

Reconstrução (Projeção e Visão Tridimensional)

- Cálculo da Nuvem de Pontos

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

$$\begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix} = Q \cdot \begin{bmatrix} x \\ y \\ \text{disparity}(x,y) \\ 1 \end{bmatrix}$$

$$_3dImage(x,y) = (X/W, Y/W, Z/W)$$

- 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 a distância X,Y e Z no plano tridimensional da nuvem de pontos. Para esse cálculo devemos utilizamos os valores intrínsecos da câmera já mencionados (f , T e as coordenadas x e y dos pontos principais da câmera esquerda).

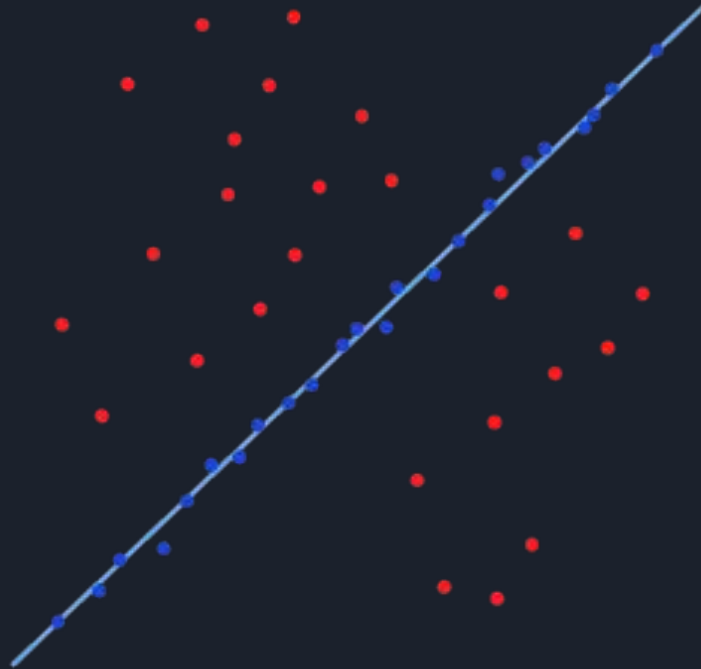
Odometria Visual

- A odometria visual é o método para determinar a posição e a orientação de um robô, analisando as imagens da câmera associadas.

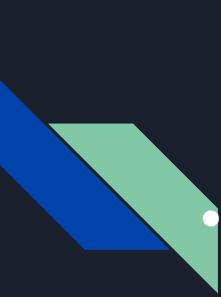


RANSAC

- Random sample consensus (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.



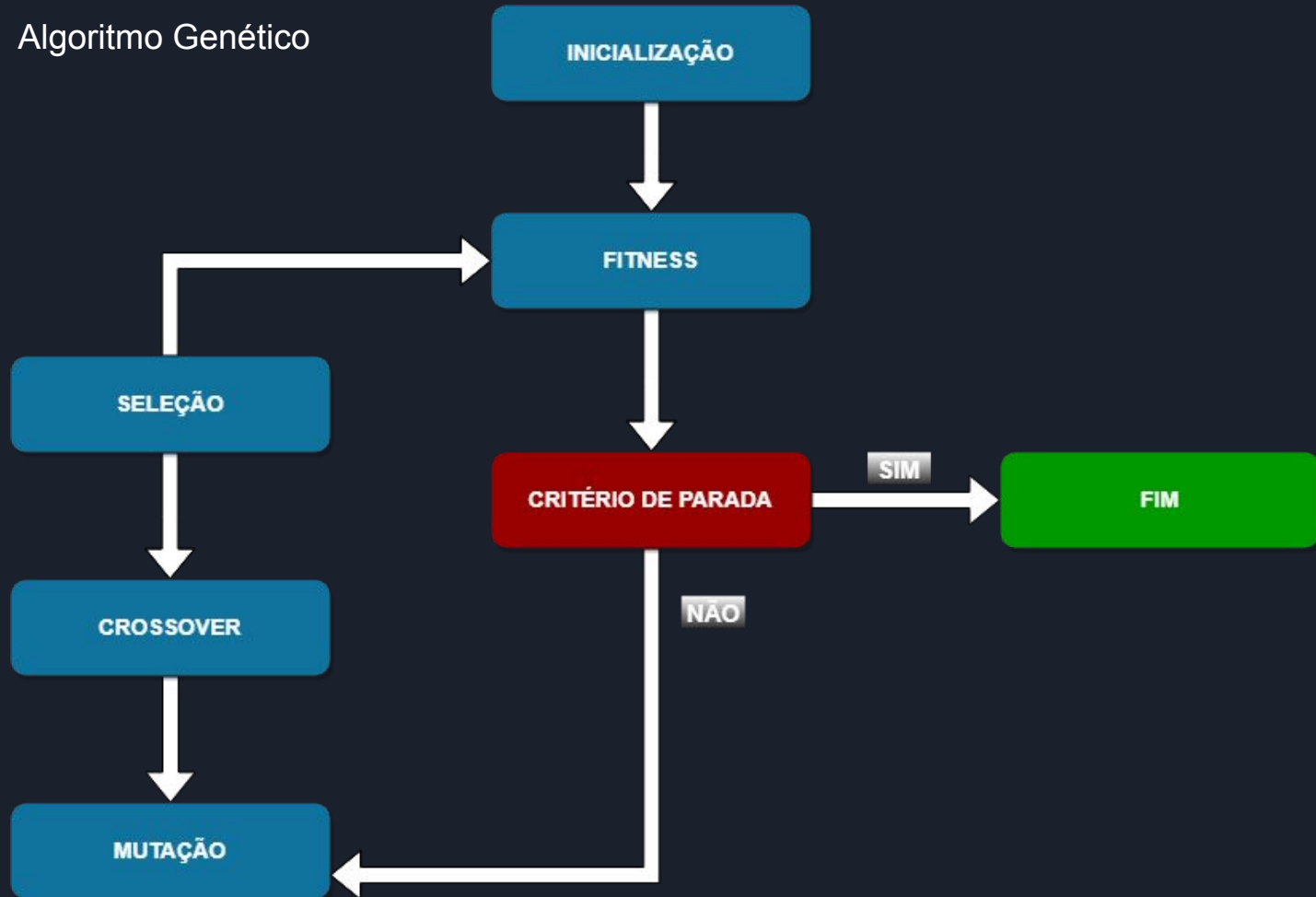
Parametrização do RANSAC



- 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.

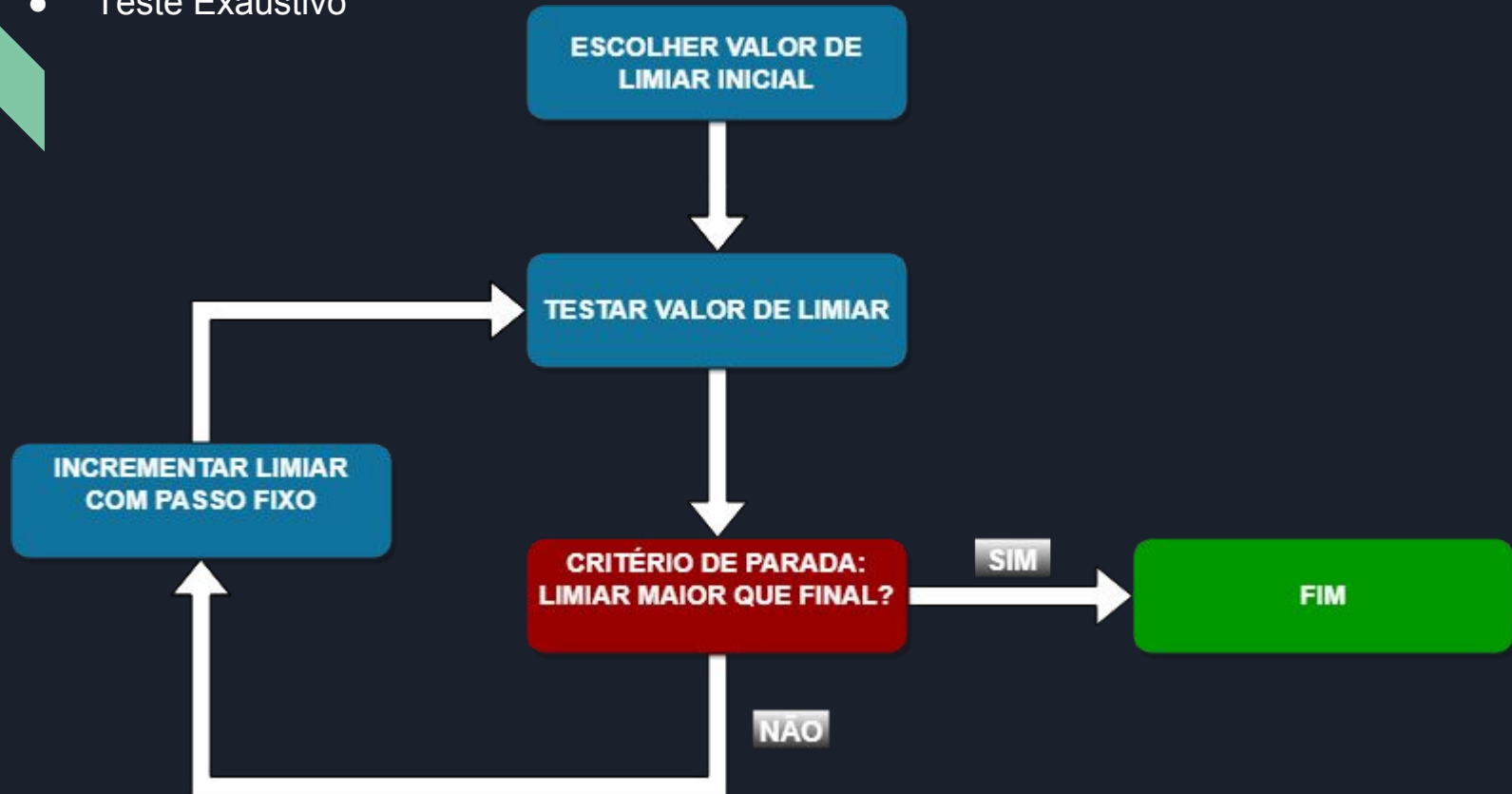
Parametrização do RANSAC

- Algoritmo Genético



Parametrização do RANSAC

- Teste Exaustivo






TRABALHOS RELACIONADOS

Trabalhos Relacionados

- Conforme discutido 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.



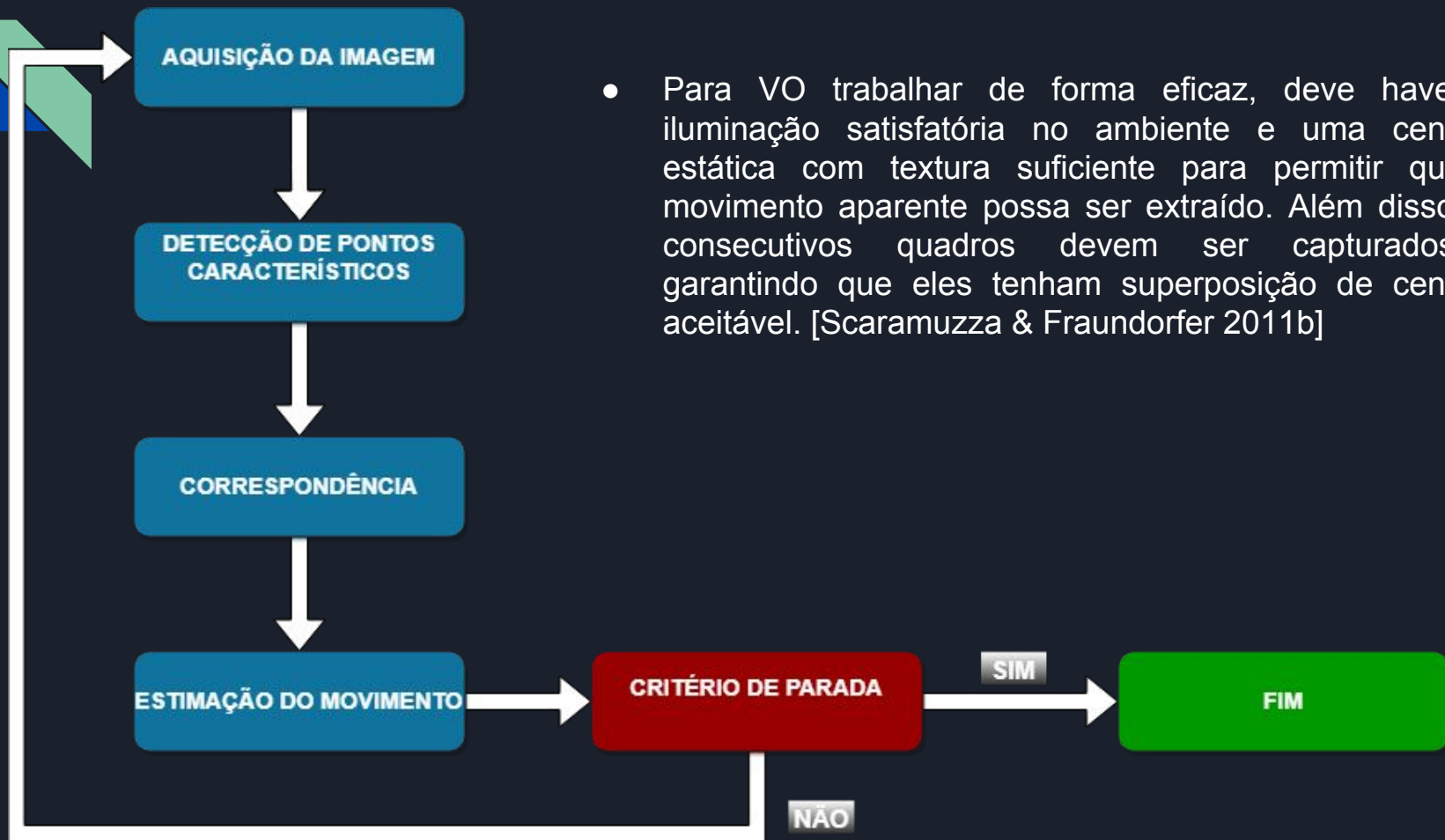
Trabalhos Relacionados

- 
- Trabalho de [Scaramuzza 2011] faz uma análise do RANSAC sob um sistema de odometria visual monocular.
 - Trabalho de conclusão de curso, foi [Hosseini-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.

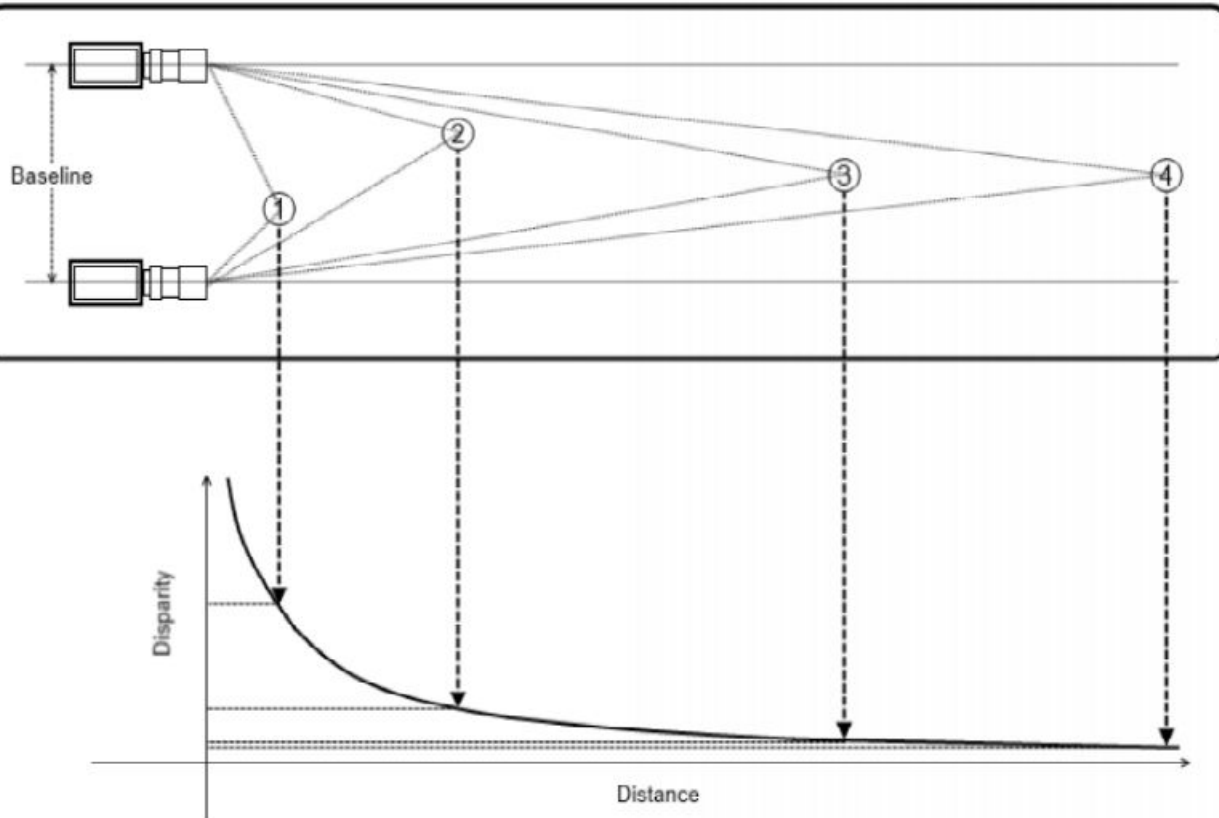


O PROBLEMA

Odometria e variações no ambiente



Cálculo da Disparidade em imagens estéreo para pontos distantes

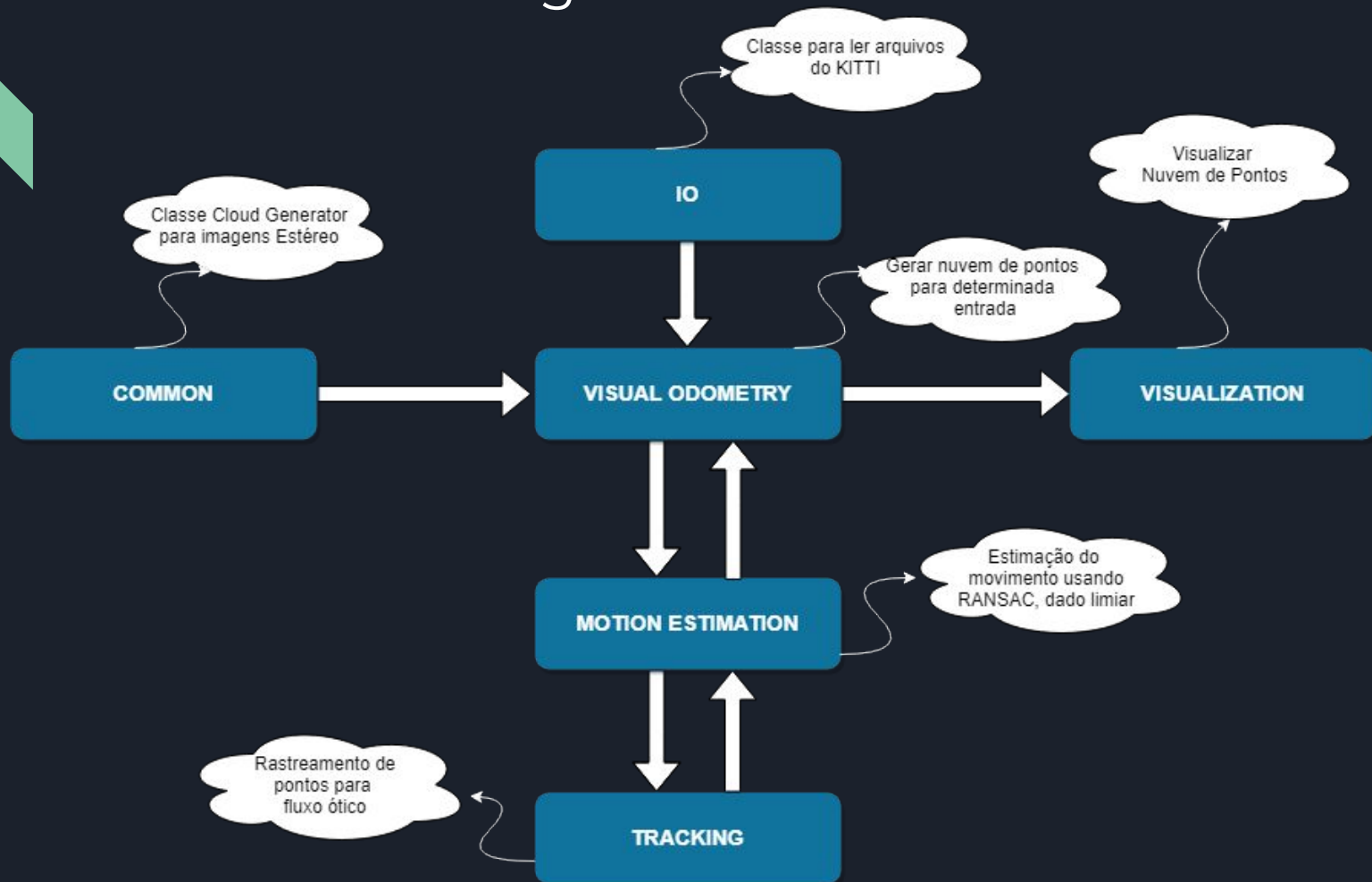


- o erro de correspondência de cada componente no cálculo da disparidade afeta a profundidade.
- Ao tratar de pontos distantes, o caso estéreo é degenerado para monocular.

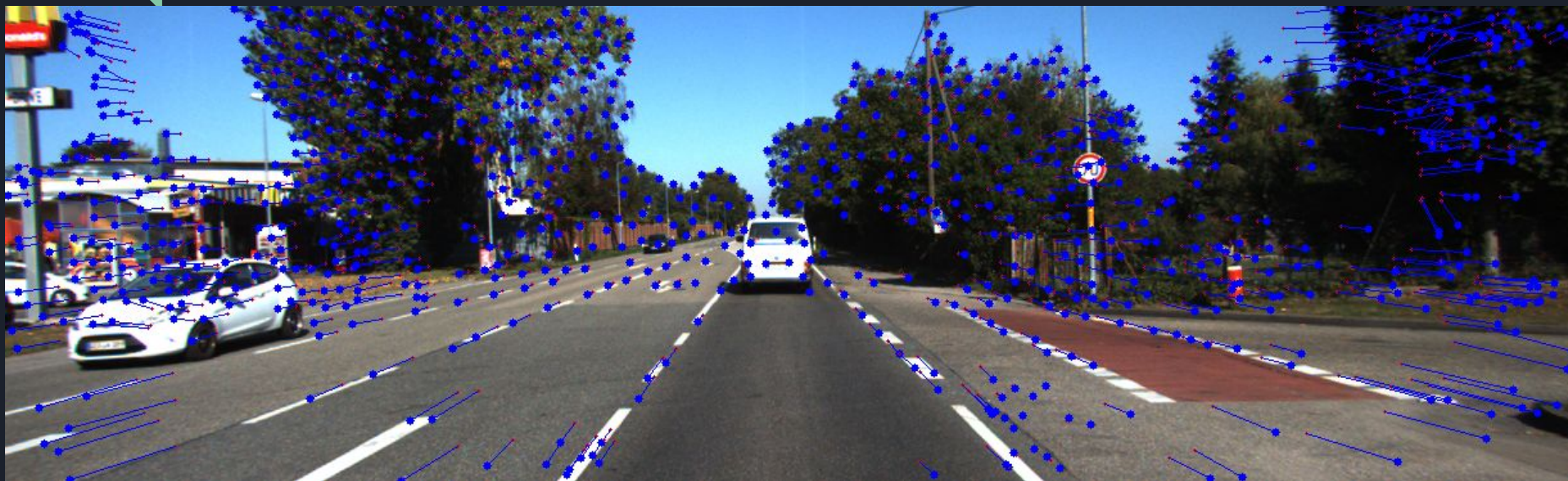


ALGORITMO E ABORDAGEM DO PROBLEMA

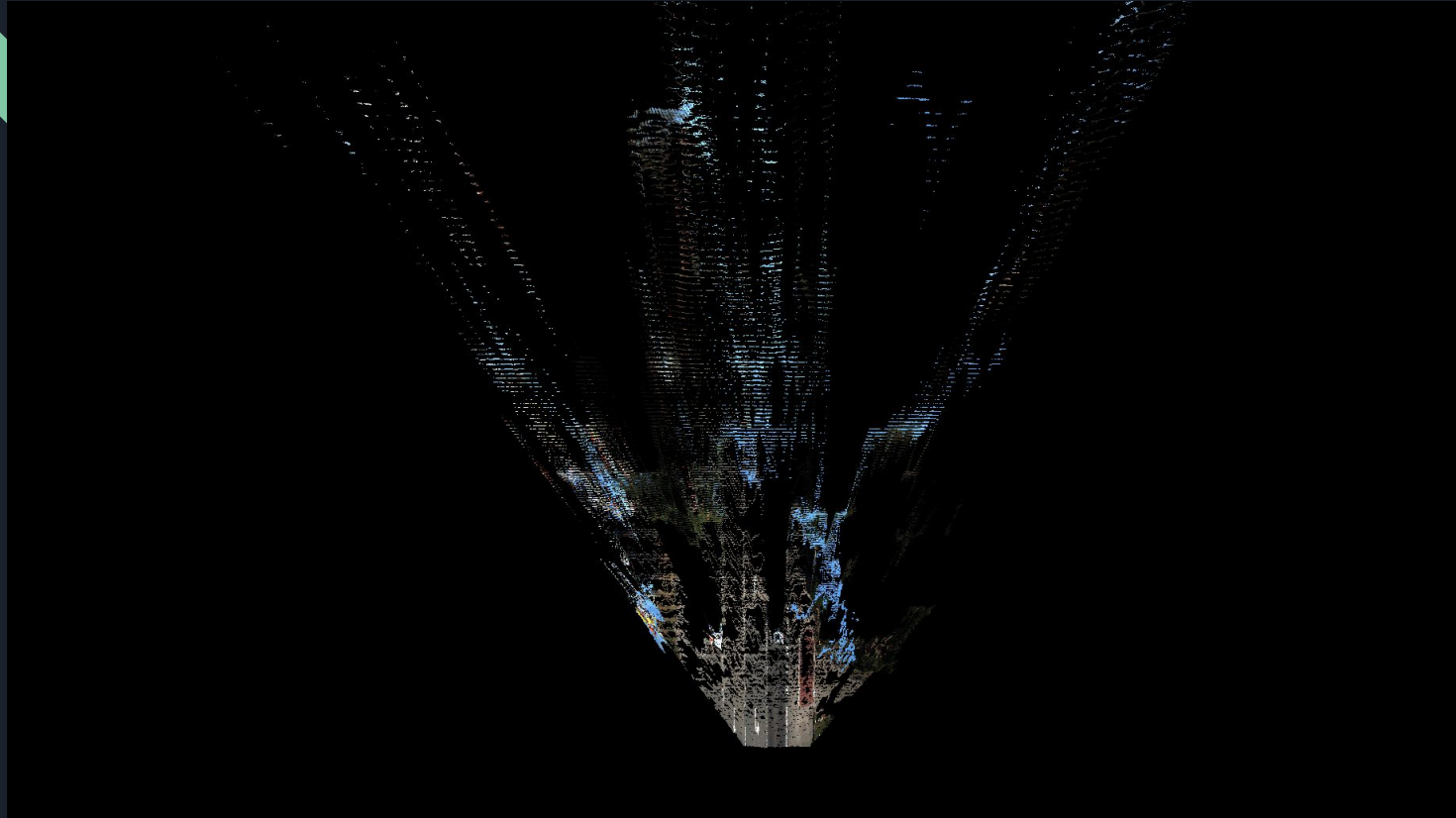
Estrutura do Algoritmo de Odometria Visual



Estrutura do Algoritmo de Odometria Visual



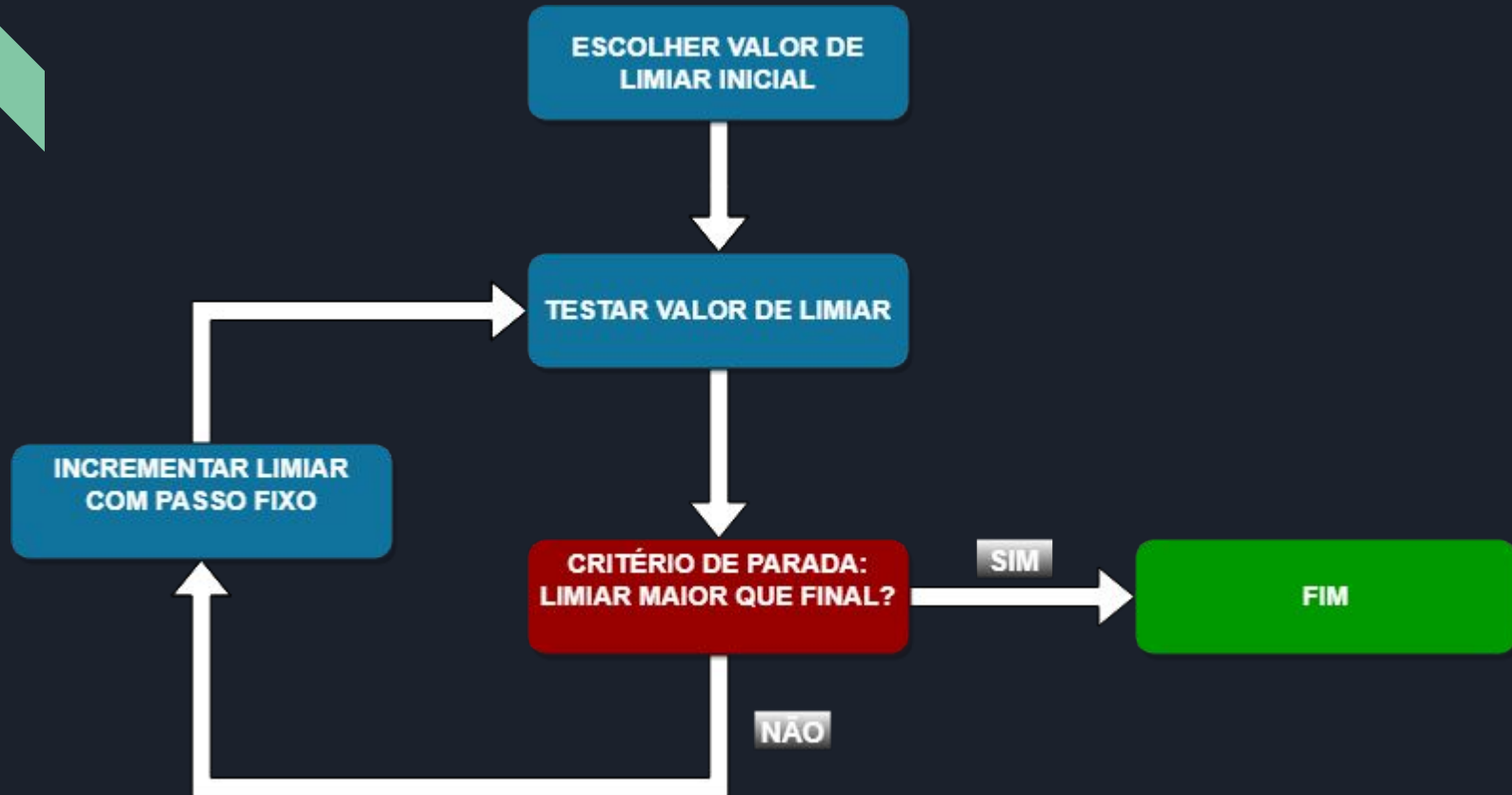
Estrutura do Algoritmo de Odometria Visual



Medição do Erro

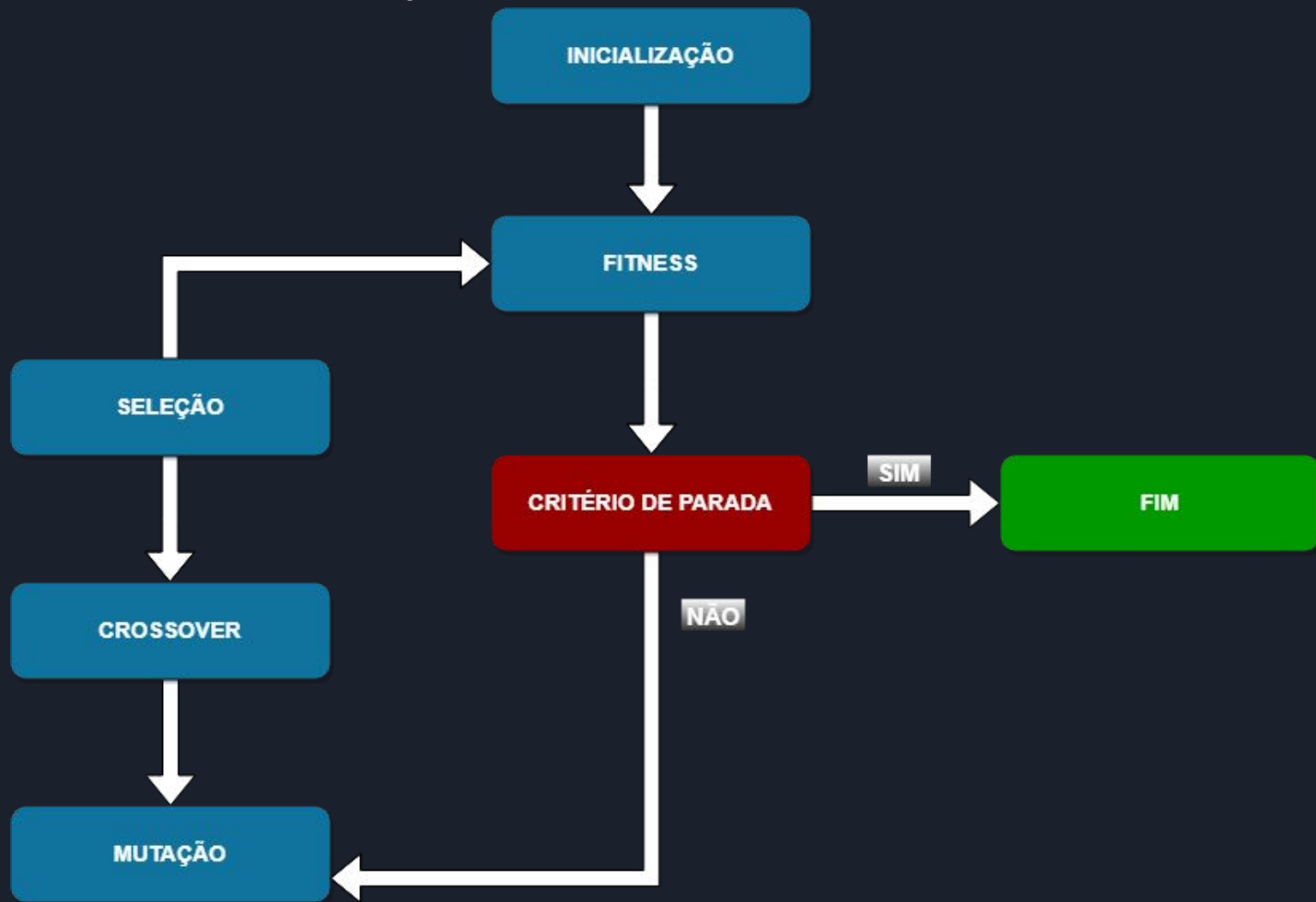
- A métrica utilizada para determinar se o threshold melhorou ou não o desempenho do projeto foi a média do erro de translação calculado pelo development kit do dataset [Geiger et al. 2012].
- Tem base no erro de rotação que usa como métrica o erro de posição relativa (erro entre todas as correspondências).

Parametrização do RANSAC: Teste Exaustivo




```
#Passo 1: Executar o rgbd_rtk com o threshold desejado para a sequência desejada
s="./rgbd_rtk/build/applications/bin/stereo_optical_flow_visual_odometry_test ./KITTI_color/SubColor/ "+str(numeroSequencia)+" "+str(threshold)
os.system(s)
#Passo 2: Mover o arquivo gerado em txt para ./devkit/cpp/results/stereo/data/01.txt
s="mv 0"+str(numeroSequencia)+".txt ./devkit/cpp/results/stereo/data/"
os.system(s)
#Passo 3: rodar o devkit
os.system("cd ./devkit/cpp/ && ./evaluate_odometry stereo")
#Passo 4: ler o arquivo de erro e salvar a media do
#erro translacional: 3 coluna do arquivo gerado em ./devkit/cpp/results/stereo/errors/01.txt
s="./devkit/cpp/results/stereo/errors/0"+str(numeroSequencia)+".txt"
arq = open(s,'r')
erroTranslacional=0
numeroLinhas=0
for linha in arq:
    valores = linha.split()
    erroTranslacional=erroTranslacional+float(valores[2])
    numeroLinhas=numeroLinhas+1
erroTranslacional=erroTranslacional/numeroLinhas
arq.close()
#Passo 5: Salvar o threshold e o erro em texto e incrementar o threshold
texto.append(str(threshold)+" "+str(erroTranslacional)+"\n")
threshold=threshold+passo
#Passo 6: escrever no arquivo de saída o threshold e o erro respectivamente
s="/resultadosErro/testeExaustivo0"+str(numeroSequencia)+".txt"
arq = open(s,'w')
arq.writelines(texto)
arq.close();
#Passo 7: apagar arquivos para podermos sobrescrever
os.system("rm ./devkit/cpp/results/stereo/data/* && rm ./devkit/cpp/results/stereo/errors/* && rm ./devkit/cpp/results/stereo/plot error/* &&
```

Parametrização do RANSAC: Algoritmo Genético



Parametrização do RANSAC: Algoritmo Genético

```
# -*- Coding: UTF-8 -*-
#coding: utf-8
import os
import commands

import numpy
import random

class Agent:
    def __init__(self, valor):
        self.string = str(valor)
        self.fitness = 1
    def __str__(self):
        return 'String: ' + str(self.string) + ' Fitness: ' + str(self.fitness)

numeroSequencia=6
population = 10 #a população deve ser sempre >= a 5
media= 22
desvio = 5

generations = 10
```

Parametrização do RANSAC: Algoritmo Genético

```
def algoritmoGenetico():
    texto=[]
    agents = init_agents(population, media, desvio)
    for generation in xrange(generations):
        print 'Generation: ' + str(generation)
        agents = fitness(agents)
        agents = ordena(agents)
        print("Agentes depois de fitness e ordenação")
        imprime(agents)
        agents = selection(agents)
        print("Agentes depois de selection")
        imprime(agents)
        agents = crossover(agents)
        print("Agentes depois de crossover")
        imprime(agents)
        agents = mutation(agents)
        print("Agentes depois de mutação")
        imprime(agents)
        agents = ordena(agents)
        print("Agentes depois de ordenação")
        imprime(agents)
        agente = agents[0]
        print("Melhor")
        print(agente.string)
        print(agente.fitness)
        texto.append(agente.string+" "+str(agente.fitness)+"\n")
        s="./resultadosErro/algoritmoGenetico0"+str(numeroSequencia)+".txt"
        arq = open(s,'w')
        arq.writelines(texto)
        arq.close()
```

Parametrização do RANSAC: Algoritmo Genético

```
def imprime(agents):
    for agent in agents:
        print(agent.string)
        print(agent.fitness)

def init_agents(population, media, desvio):
    agents=[]
    contador=0
    valoresThreshold=numpy.random.normal(media,desvio,population)
    for valor in (valoresThreshold):
        agents.append(Agent(abs(valor)))
        contador=contador+1
    return agents

def fitness(agents):
    for agent in agents:
        if(agent.fitness==1):
            agent.fitness = erro_devKit(agent.string)
            print(agent.string)
            print(agent.fitness)
    return agents

def ordena(agents):
    return sorted(agents, key=lambda agent: agent.fitness, reverse=False)

def selection(agents):
    agents = agents[:int(0.4 * len(agents))]
    return agents
```

Parametrização do RANSAC: Algoritmo Genético

```
def crossover(agents):
    offspring = []
    for _ in xrange((population - len(agents)) / 2):
        parent1 = random.choice(agents)
        parent2 = random.choice(agents)
        child1 = Agent(float(parent1.string))
        child2 = Agent(float(parent2.string))
        dif=abs(float(parent1.string)-float(parent2.string))
        influencia = float(parent1.string) +dif
        child1.string=str(influencia)
        influencia = abs(float(parent2.string) -dif)
        child2.string=str(influencia)
        offspring.append(child1)
        offspring.append(child2)
    agents.extend(offspring)
    return agents

def mutation(agents):
    for agent in agents:
        if random.uniform(0.0, 1.0) <= 0.1:
            print("mutação de")
            print(agent.string)
            print(agent.fitness)
            agent.string = str(abs(2*desvio*random.random()+ (media-desvio)))
            agent.fitness=1
    return agents
```


Parametrização do RANSAC: Algoritmo Genético

```
def erro_devKit(threshold):
    s="./rgbd_rtk/build/applications/bin/stereo_optical_flow_visual_odometry_test ./KITTI_color/SubColor/ "+str(numeroSequencia)+" "+threshold
    os.system(s)
    s="mv 0"+str(numeroSequencia)+".txt ./devkit/cpp/results/stereo/data/"
    os.system(s)
    os.system("cd ./devkit/cpp/ && ./evaluate_odometry stereo")
    s="./devkit/cpp/results/stereo/errors/0"+str(numeroSequencia)+".txt"
    arqR = open(s,'r')
    erroTranslacional=0
    numeroLinhas=0
    for linha in arqR:
        valores = linha.split()
        erroTranslacional=erroTranslacional+float(valores[2])
        numeroLinhas=numeroLinhas+1
    erroTranslacional=erroTranslacional/numeroLinhas
    arqR.close()
    os.system("rm ./devkit/cpp/results/stereo/errors/* && rm ./devkit/cpp/results/stereo/plot_error/* && rm ./devkit/cpp/results/stereo/plot_pa")
    return erroTranslacional

if __name__ == '__main__':
    algoritmoGenetico()
```



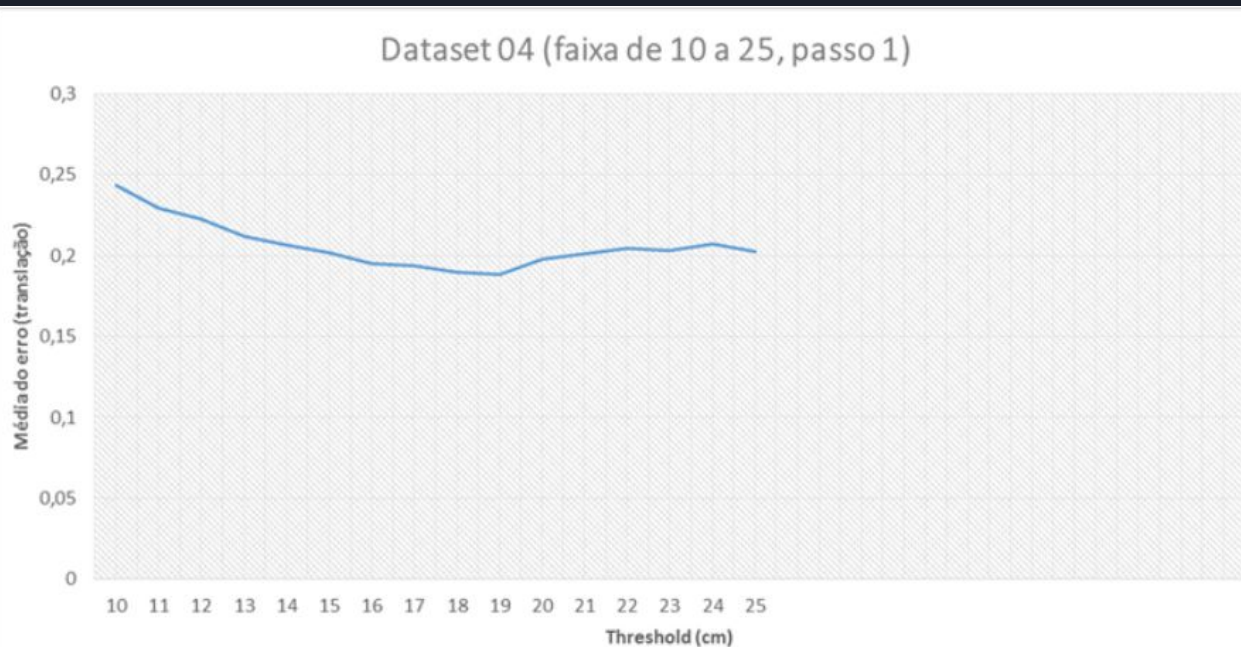
EXPERIMENTOS

Parametrização do RANSAC: 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.

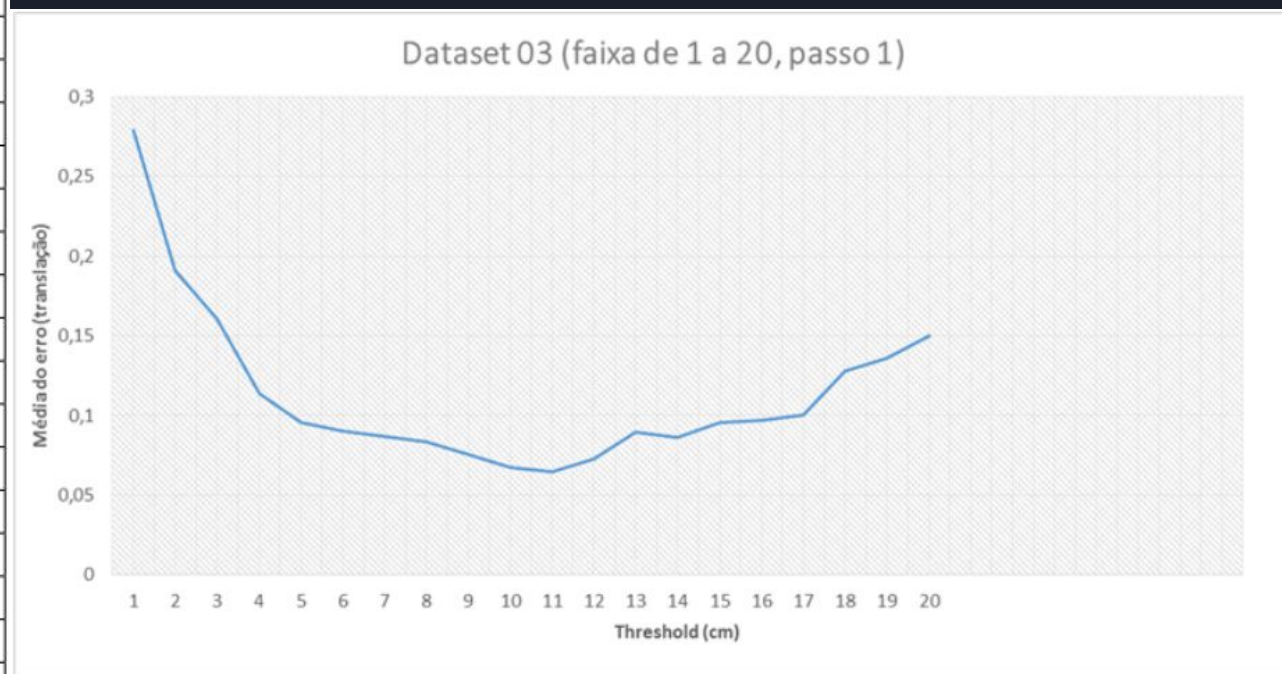
Teste Exaustivo: Sequência 04

Dataset 04	faixa de 10 a 25, passo 1
Threshold cm	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



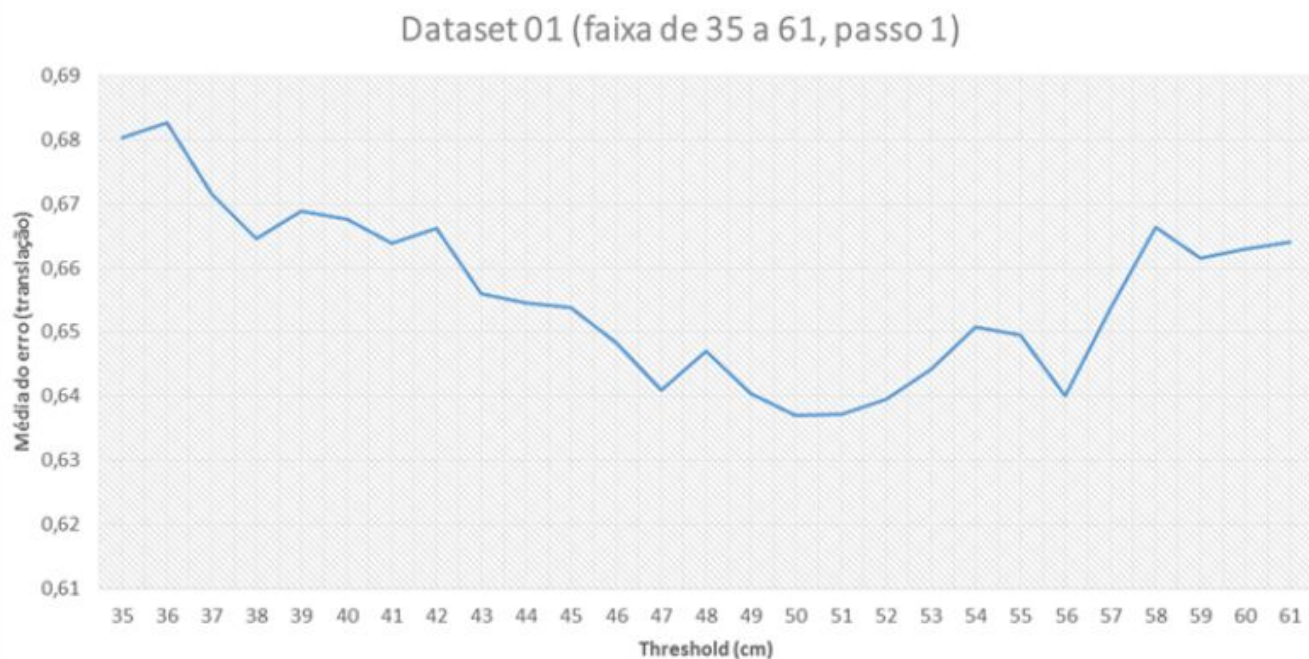
Teste Exaustivo: Sequência 03

Dataset 03	faixa de 1 a 20, passo 1
Threshold cm	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



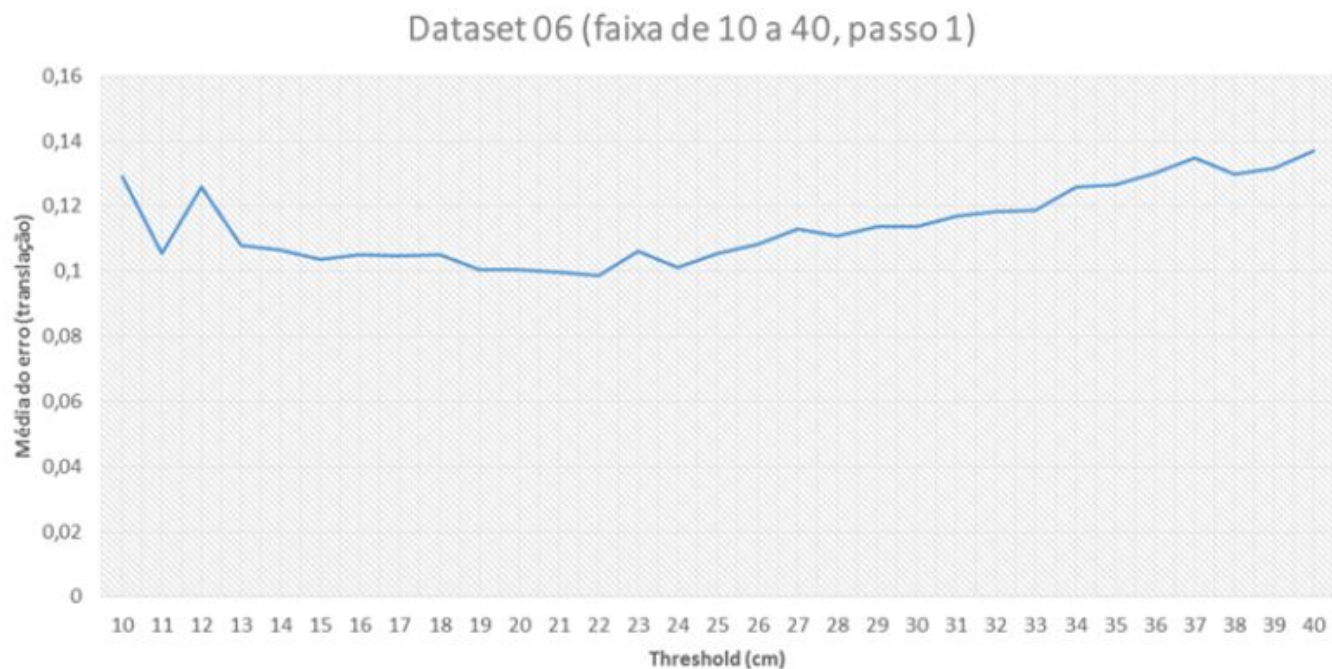
Dataset 01	faixa de 35 a 61, passo 1
Threshold cm	Média do erro(translação)
35	0.680360492707
36	0.682681625608
37	0.67153423825
38	0.664578220421
39	0.668938773096
40	0.667645364668
41	0.663899418152
42	0.666132756888
43	0.656060771475
44	0.654452009724
45	0.653825667747
46	0.648346435981
47	0.64083771637
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

Teste Exaustivo: Sequência 01



Dataset 06	faixa de 10 a 40, passo 1
Threshold cm	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

Teste Exaustivo: Sequência 06



Parametrização do RANSAC: 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.

Dataset 03	media 5, desvio 5
Threshold (cm)	Média do erro (translação)
6.79347974199	0.0823417065217
8.50717421481	0.075938625
8.50717421481	0.075938625
10.2208686876	0.0652939021739
10.2208686876	0.0652939021739

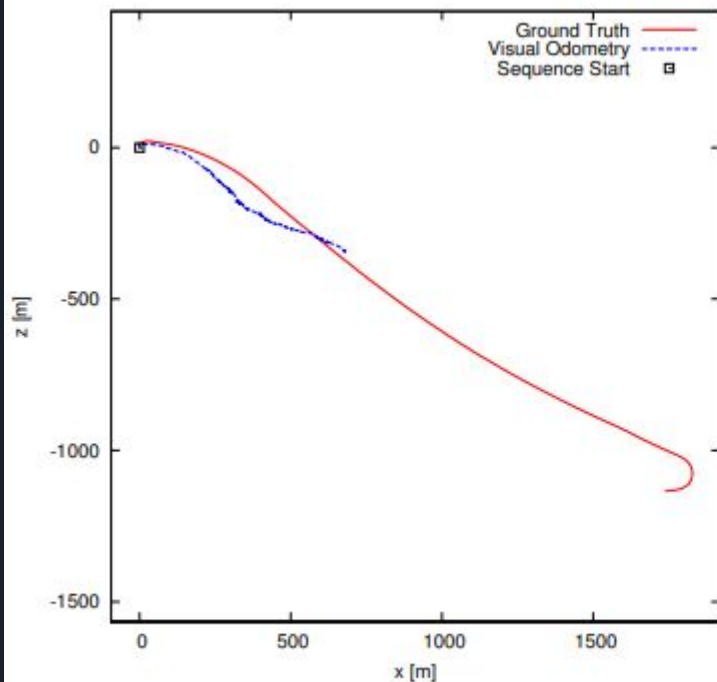
Dataset 01	media 52, desvio 10
Threshold (cm)	Média do erro (translação)
49.8855209687	0.635082867099
49.8855209687	0.635082867099
49.8855209687	0.635082867099
49.8118231238	0.632428748784
49.8118231238	0.632428748784

Dataset 04	media 20, desvio 5
Threshold (cm)	Média do erro (translação)
18.2945042908	0.189460418605
18.2656038554	0.188787069767
18.23670342	0.187898093023
18.23670342	0.187898093023
18.23670342	0.187898093023

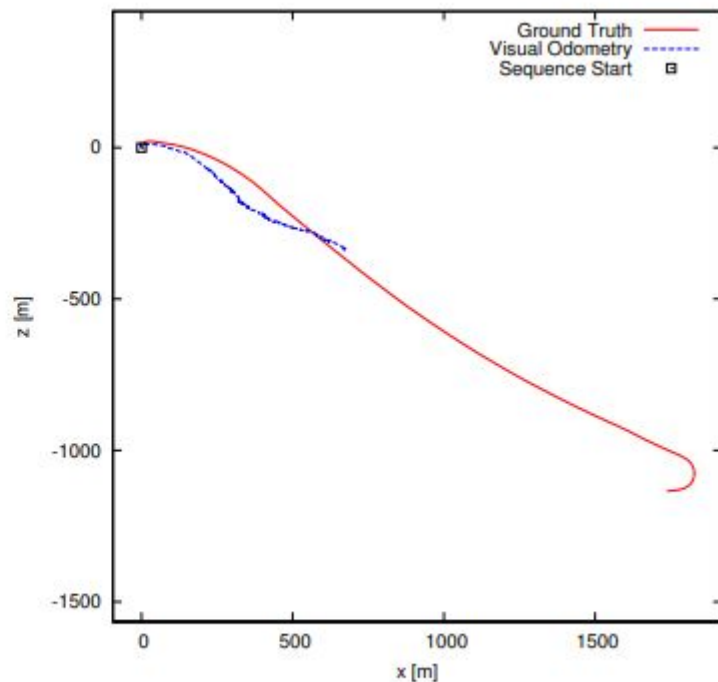
Dataset 06	media 23, desvio 10
Threshold (cm)	Média do erro (translação)
21.0026484588	0.0996869736842
21.0026484588	0.0996869736842
21.0026484588	0.0996869736842
20.5362424457	0.0986120947368
20.5362424457	0.0986120947368

Algoritmo Genético vs Teste Exaustivo

Sequência 01



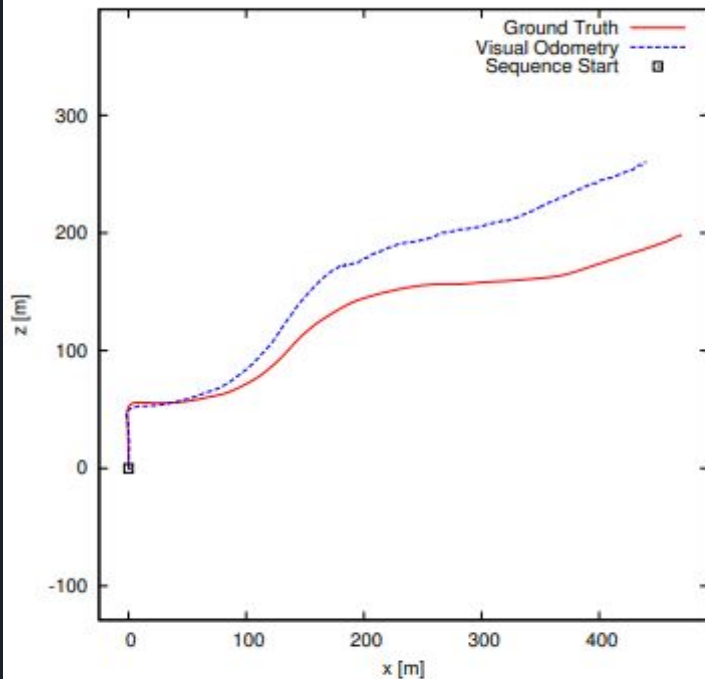
(a) Algoritmo Genético 01



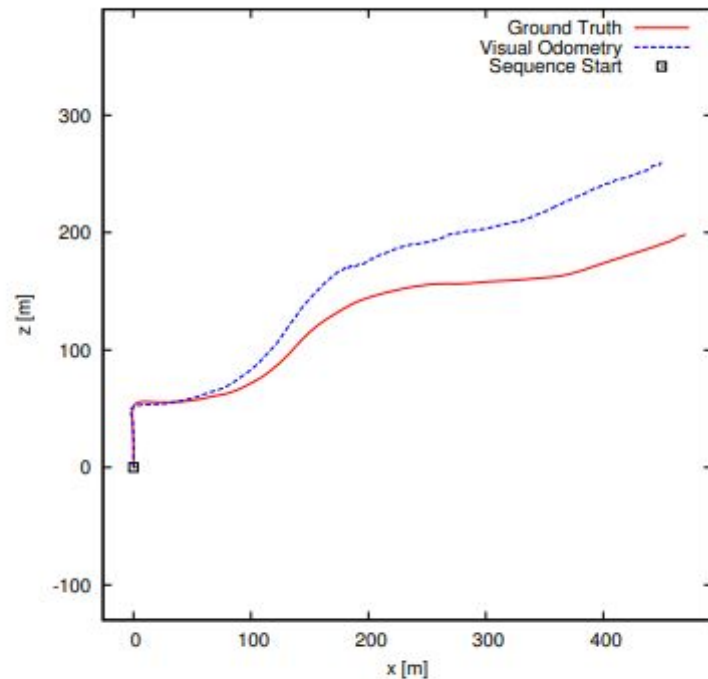
(b) Teste Exaustivo 01

Algoritmo Genético vs Teste Exaustivo

Sequência 03



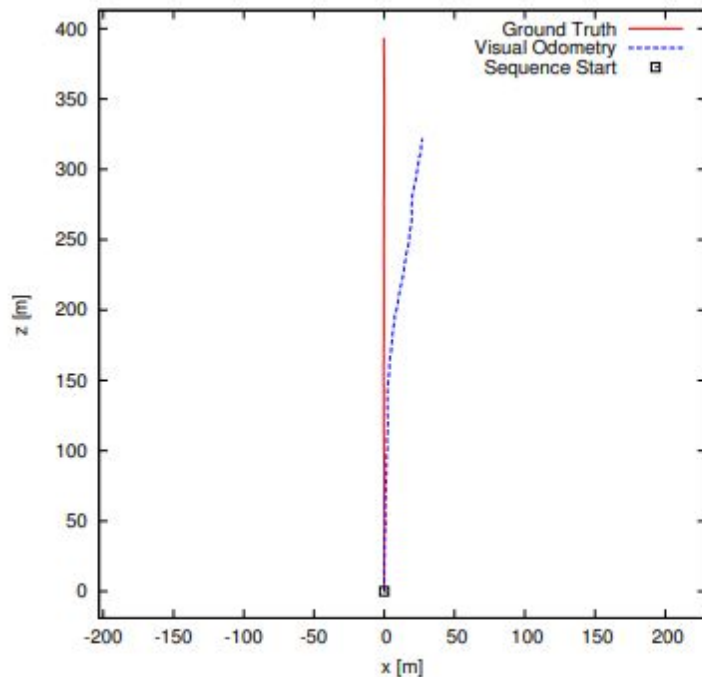
(a) Algoritmo Genético 03



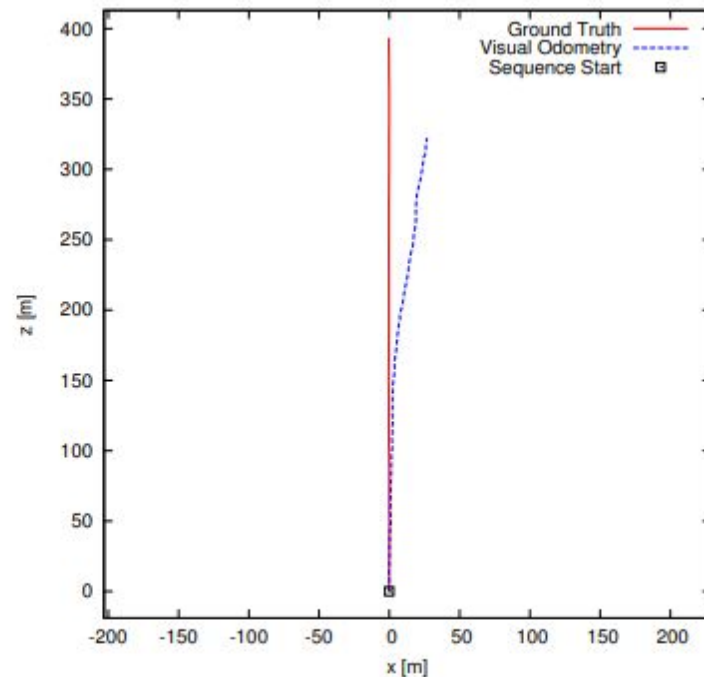
(b) Teste Exaustivo 03

Algoritmo Genético vs Teste Exaustivo

Sequência 04



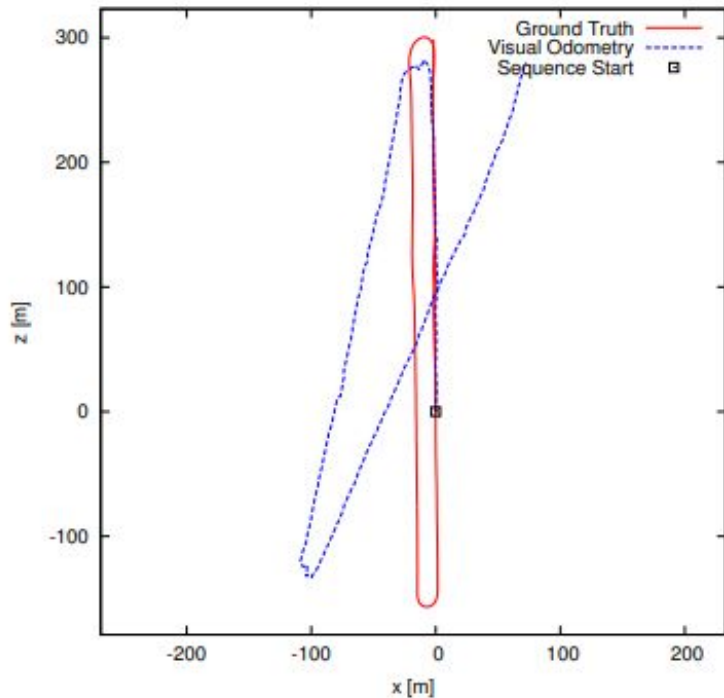
(a) Algoritmo Genético 04



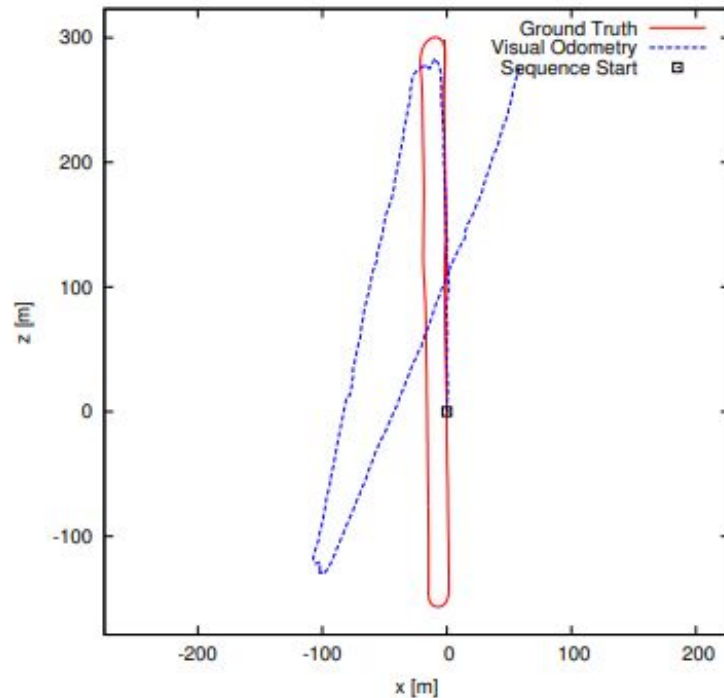
(b) Teste Exaustivo 04

Algoritmo Genético vs Teste Exaustivo

Sequência 06



(a) Algoritmo Genético 06



(b) Teste Exaustivo 06

Desvantagem Teste Exaustivo

- Dependente da faixa de valores testada

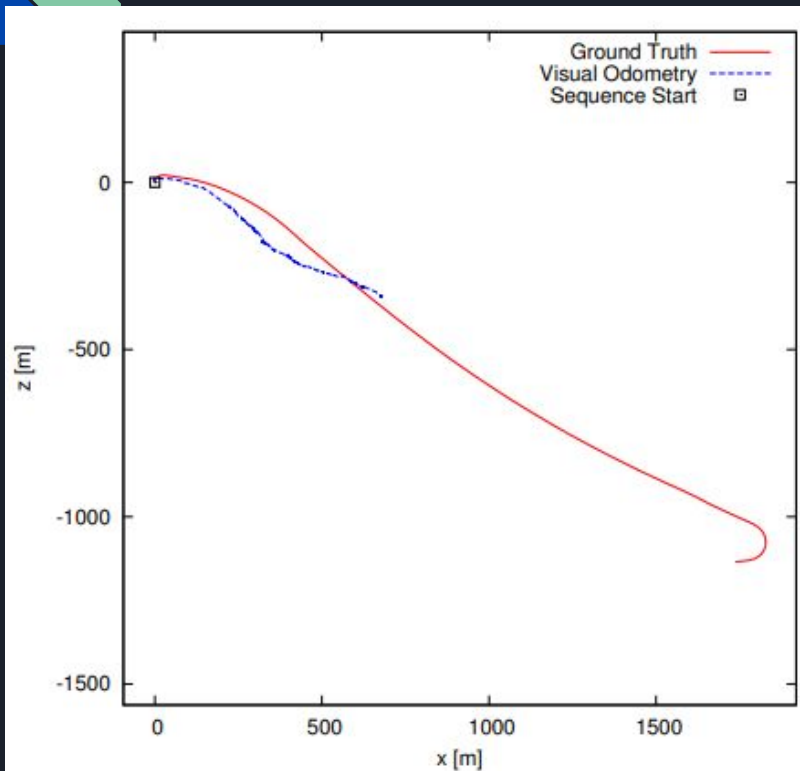
Dataset 01	faixa de 0.5 a 10.5, passo 1
Threshold cm	Média do erro (translação)
0.5	0.926811753647
1.5	0.873590789303
2.5	0.843967126418
3.5	0.819378688817
4.5	0.803934641815
5.5	0.797938066451
6.5	0.794576824959
7.5	0.790058658023
8.5	0.781211175041
9.5	0.775006259319
10.5	0.76917580389

Melhor
solução?

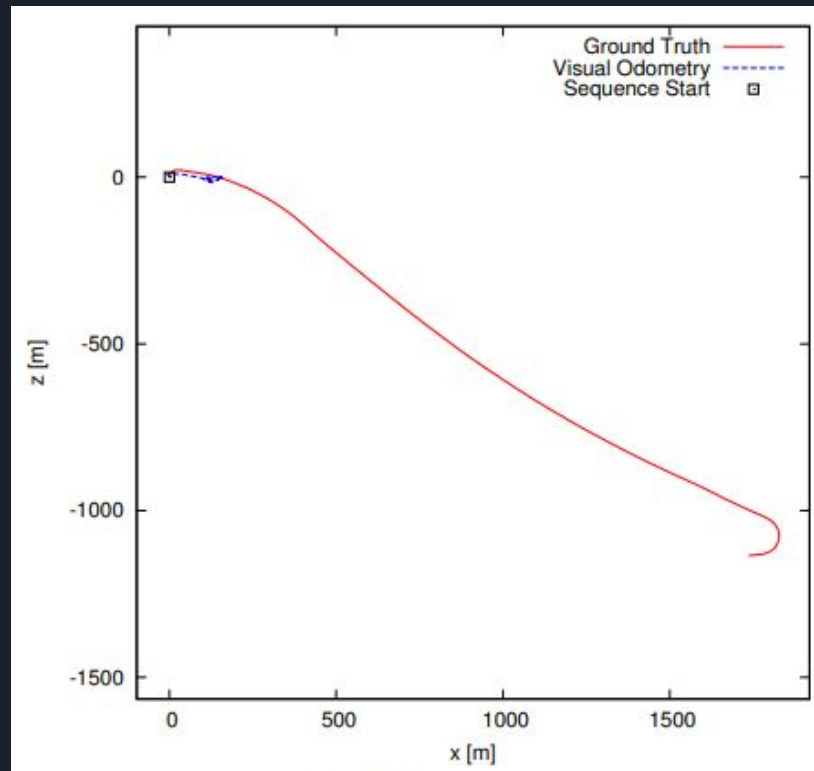


Importância da Escolha do limiar

Sequência 01



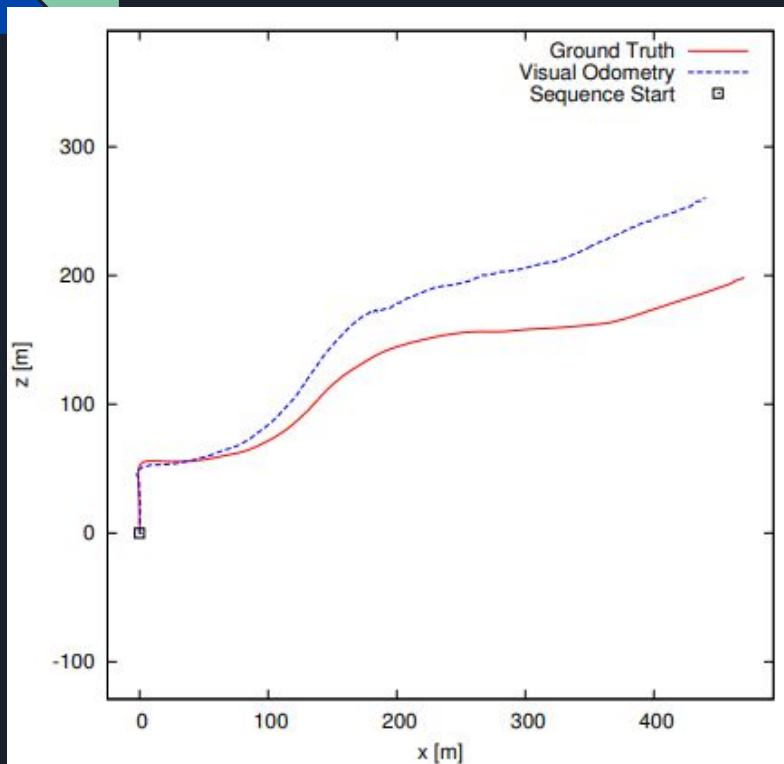
(a) Algoritmo Genético 01



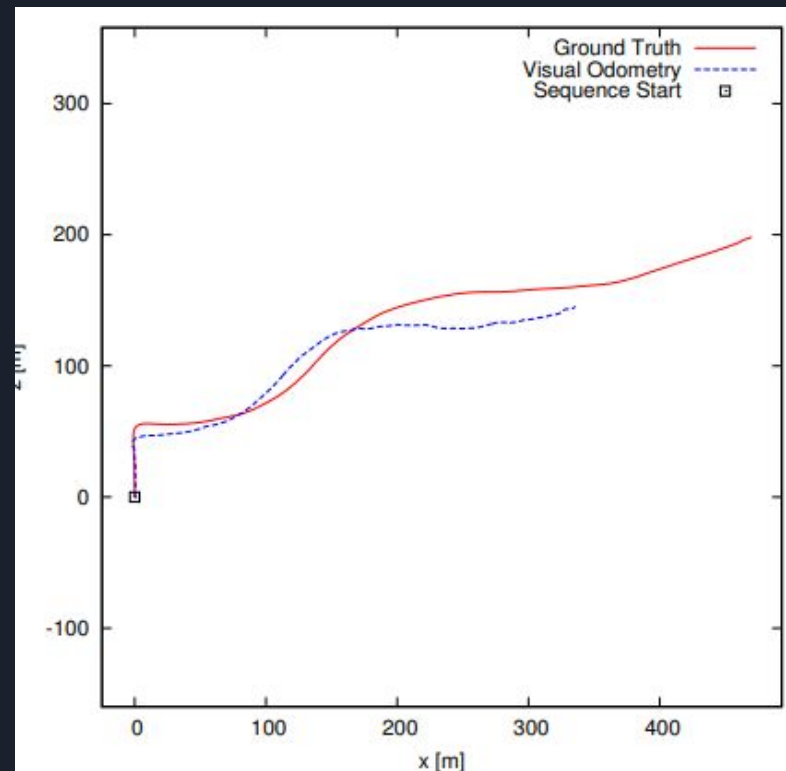
(b) Maior erro 01

Importância da Escolha do limiar

Sequência 03



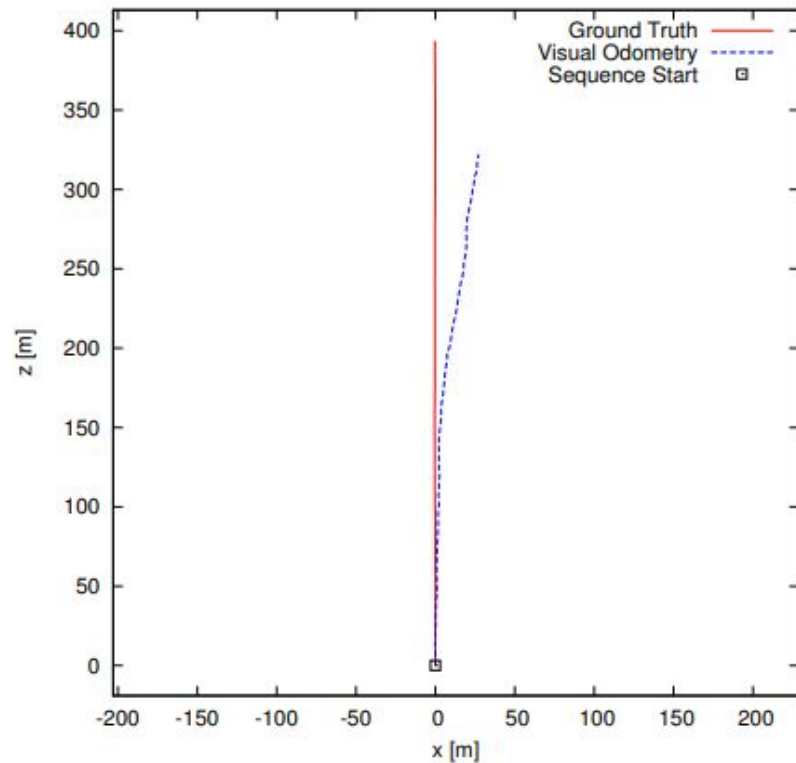
(a) Algoritmo Genético 03



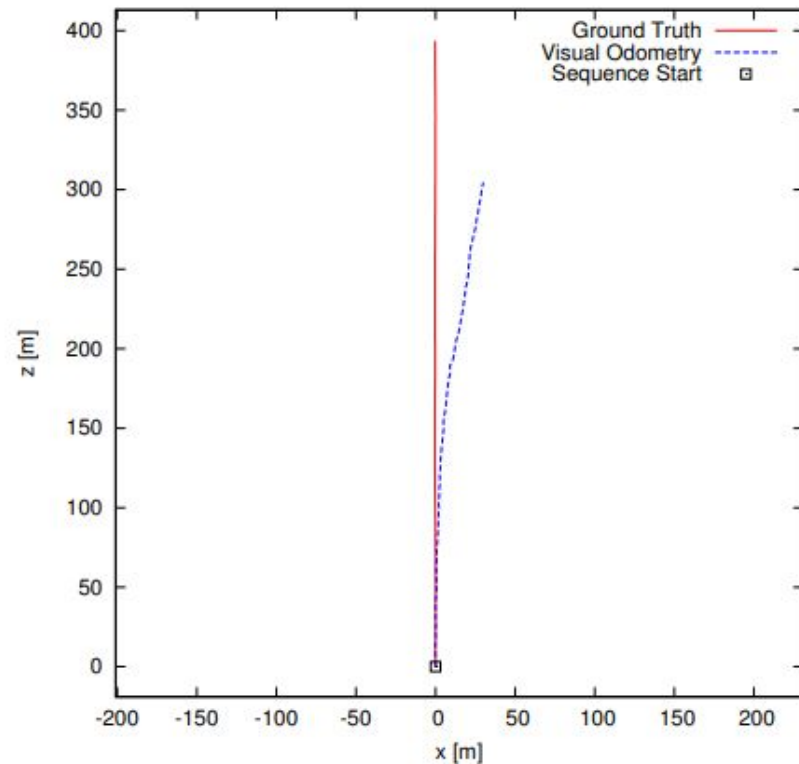
(b) Maior erro 03

Importância da Escolha do limiar

Sequência 04



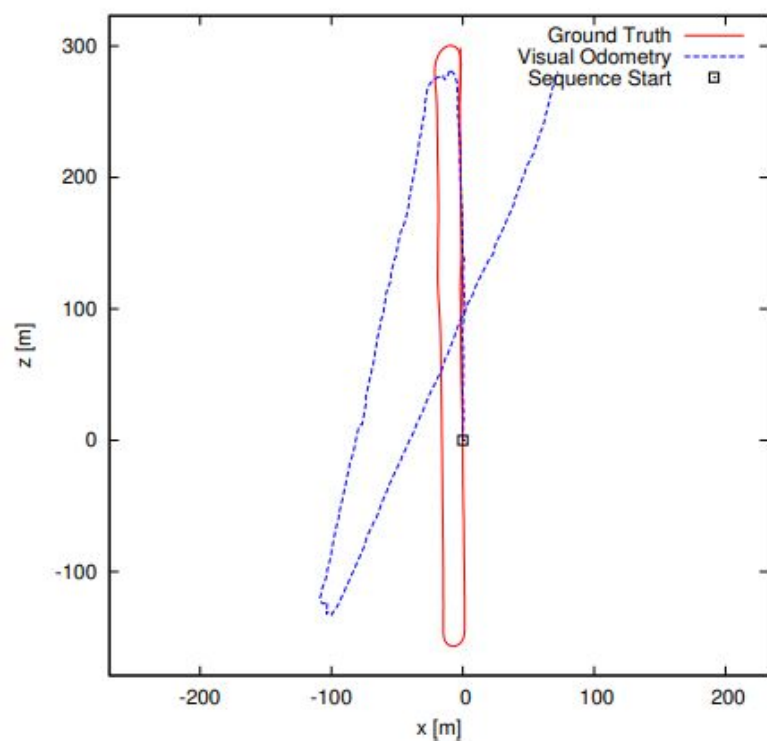
(a) Algoritmo Genético 04



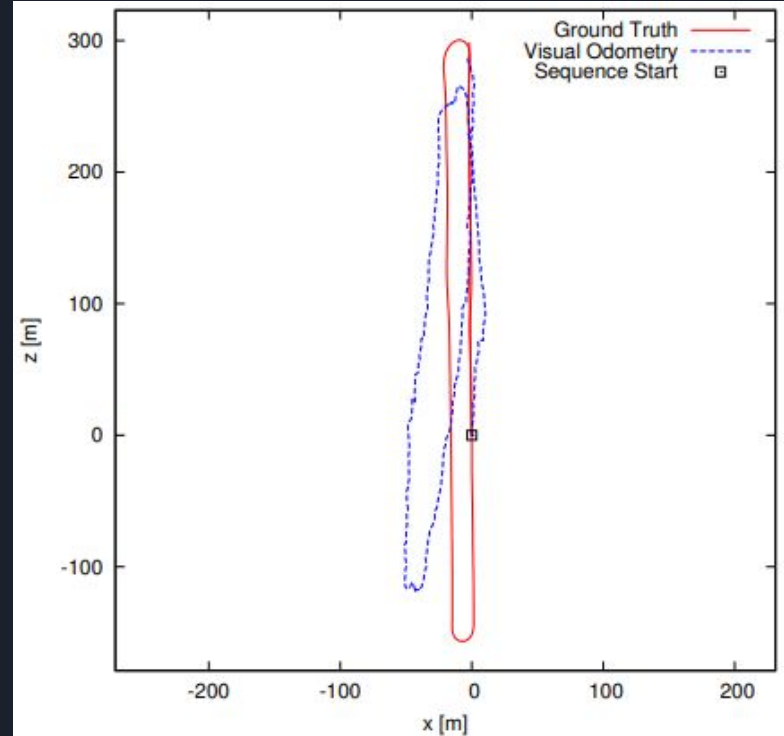
(b) Maior erro 04

Importância da Escolha do limiar

Sequência 06



(a) Algoritmo Genético 06



(b) Maior erro 06

Análise das Sequências Escolhidas

- 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 válido 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 segui-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 também com baixa dinamicidade, possui algumas curvas conforme podemos perceber ao observar a trajetória traçada. 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.



CONCLUSÃO

Conclusão

- 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.
- A escolha do limiar tem suma importância sobre o projeto de odometria visual com imagens estéreo.
- 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

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.

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.

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.



Referências

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.

Visual Odometry (VO) (n.d.), http://www.cs.toronto.edu/~urtasun/courses/CSC2541/03_odometry.pdf. Accessed: 2018-06-04. Visão Estéreo (n.d.), <http://www.comp.ita.br/~forster/CC-222/lecture/06-Visao-Estereo.pdf>. Accessed: 2018-08-07.

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.



FIM