MODELAGEM DE TERRENOS NATURAIS ATRAVÉS DE MALHAS DE TRIÂNGULOS

Anderson Patury Sangreman

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia Elétrica, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia Elétrica.

Orientador: Ramon Romankevicius Costa

Rio de Janeiro
Abril de 2014

MODELAGEM DE TERRENOS NATURAIS ATRAVÉS DE MALHAS DE TRIÂNGULOS

Anderson Patury Sangreman

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA ELÉTRICA.

Examinada por:

Prof. Ramon Romankevicius Costa, D.Sc.

Prof. Fernando Cesar Lizarralde, D.Sc.

Prof. Marco Antonio Meggiolaro, Ph.D.

RIO DE JANEIRO, RJ – BRASIL
ABRIL DE 2014

Sangreman, Anderson Patury

Modelagem de Terrenos Naturais Através de Malhas de Triângulos/Anderson Patury Sangreman. – Rio de Janeiro: UFRJ/COPPE, 2014.

XI, 86 p.: il.; 29, 7cm.

Orientador: Ramon Romankevicius Costa

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia Elétrica, 2014.

Referências Bibliográficas: p. 66 – 70.

1. Modelagem Digital de Terreno (MDT). 2. Triangulação de Delaunay. 3. Robótica de Campo. I. Romankevicius Costa, Ramon. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia Elétrica. III. Título.

Agradecimentos

Quero expressar agradecimentos a todos que me ajudaram a chegar ao final desse trabalho de mestrado. Em especial, agradeço aos professores da graduação e do mestrado, principalmente a Ramon Romankevicius Costa, Fernando Cesar Lizarralde e Afonso Celso Del Nero Gomes.

Gostaria de agradecer a Gustavo Medeiros Freitas por toda ajuda ao longo desse trabalho, e ao projeto CASC liderado pela Carnegie Mellon University (CMU) pelos dados experimentais.

E não menos importante, agradeço a Daniele Cristina Oliveira da Silva pela boa vontade na solução de todos os problemas relativos à secretaria do PEE, e a Robeto Calvet pelo gerenciamento das questões relacionadas ao LABCON.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

MODELAGEM DE TERRENOS NATURAIS ATRAVÉS DE MALHAS DE TRIÂNGULOS

Anderson Patury Sangreman

Abril/2014

Orientador: Ramon Romankevicius Costa

Programa: Engenharia Elétrica

Nesse trabalho é apresentado um método de modelagem de terrenos naturais cobertos por vegetação. A construção do modelo é feita a partir da triangulação de uma nuvem de pontos 3D, neste caso obtidos por um laser, gerando um superfície de duas dimensões e meia. Os pontos são previamente filtrados para que sejam excluídos os dados que representam a vegetação. Dessa forma, é possivel obter um modelo da superfície suportadora de carga (*load bearing surface*). Testes realizados em um ambiente natural ilustram o funcionamento do método proposto através dos resultados obtidos. Os mapas gerados representam o terreno com detalhes, de acordo com a resolução definida, que deve ser escolhida levando em conta o compromisso entre nível de detalhamento desejado e processamento computacional exigido.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

MODELING OF NATURAL TERRAIN THROUGH TRIANGULAR MESHES

Anderson Patury Sangreman

April/2014

Advisor: Ramon Romankevicius Costa

Department: Electrical Engineering

In this work we present a method for modeling natural terrain covered by vegetation. The model construction is made from triangulation of a 3D point cloud, in this case obtained by a laser, producing a two and a half dimension surface. The points are previously filtered, so that the data representing the vegetation are excluded. This way, is built a model of the load bearing surface. Tests performed in natural terrain illustrate the performance of the method. The obtained maps represent the terrain profile in detail according to the defined resolution, which should be chosen taking into account a tradeoff between level of detail and computational processing power required.

Sumário

Lista de Figuras

Capítulo 1

Introdução

A modelagem de um terreno é uma forma de representar a distribuição da elevação de sua superfície (Plagemann et al., 2009), simplificando infinitos pontos através de um modelo finito.

A modelagem de um determinado terreno é útil em várias áreas como engenharias cartográfica, civil, ambiental e militar, fornecendo informações indispensáveis para, por exemplo, viabilizar o planejamento urbano (projetos de construção de estradas, túneis, barragens e dimensionamento dos respectivos deslocamentos de terras), extração de curvas de níveis e redes de drenagem, estudos de impacto ambiental como o cálculo de áreas alagadas na construção de hidroelétricas (Piteri et al., 2007), pesquisas geológicas, treinamentos em simulações de voo e automação de robôs móveis. Mais aplicações, são descritas em Barbosa et al. (2003), Li et al. (2010) e El-Sheimy et al. (2005).

Nesta dissertação são analisadas questões pertinentes à robótica móvel, com foco na automação de veículos de campo. A atuação de veículos autônomos em terrenos irregulares possibilita redução da necessidade de operadores treinados, socorro de pessoas em situação de risco e alta capacidade de explorar regiões remotas na terra e em outros planetas.

Mesmo que um veículo não seja totalmente autônomo, existem vantagens consideráveis em se utilizar um sistema que pode tirar conclusões sobre o ambiente e tomar decisões. Essa capacidade pode ser usada para reduzir a probabilidade de capotamento na operação tanto de veículos controlados remotamente quanto de plataformas autônomas.

A modelagem de terrenos e a construção de mapas são tarefas centrais na robótica, e podem ser utilizadas para planejamento de trajetória, simulação do trajeto, ajuste da configuração do robô, identificação de pontos de contato, análise de estabilidade, entre outras. Contudo, a exploração de ambientes naturais por robôs ainda apresenta desafios além do estado da arte atual (Hamner et al., 2008). A estrutura irregular do terreno aumenta o risco de tombamento de veículos e robôs móveis, exi-

gindo maior atenção a questões relativas à estabilidade do sistema. A presença de vegetação e a possibilidade da existência de obstáculos cobertos são algumas dentre as dificuldades de se locomover através desse tipo de terreno.

1.1 Motivação

Terrenos naturais (em inglês também denominados *outdoor* ou *off-road*) tendem a compor superfícies irregulares que não são facilmente representadas por poucas figuras geométricas básicas. Sua modelagem se torna não trivial e problemas surgem à tona, como a capacidade necessária de processamento e a memória disponível, principalmente para robôs móveis com sistema de processamento embarcado.

Esses terrenos podem apresentar grande parte da superfície coberta por vegetação, pedras e outros obstáculos (como por exemplo o terreno apresentado na Figura 1.1). Isso gera dificuldade na identificação da superfície coberta do solo, e torna necessária a utilização de um método de filtragem da vegetação do terreno modelado.

Figura 1.1: Terreno natural parcialmente coberto por vegetação.

Para que seja possível a locomoção contínua de veículos autônomos em ambientes *outdoor*, é necessário uma representação desses terrenos. Por exemplo, Batavia et al. (2002) propõem uma estratégia para navegação autônoma, mas precisa utilizar um modelo do terreno previamente construído. Quanto maior for a velocidade do veículo, mais precisa deve ser a representação do terreno (Broten and Collier, 2006).

1.2 Modelagem de terrenos na robótica móvel

Atualmente na robótica móvel, um dos sensores mais utilizados na coleta de dados do ambiente é o *laser range finder* (Figura 1.2). Isso se deve ao fato da sua razão

eficiência/custo ser relativamente alta, se comparada a de outros sensores. Além do laser, merecem destaque outros dois sensores populares de baixo custo: o sensor de ultrassom e a câmera. O sensor de ultrassom possui custo menor se comparado à tecnologia a laser, mas seu alcance, tempo de resposta e precisão são consideravelmente inferiores. A câmera, diferente dos anteriores, é um sensor passivo e suas medições são altamente sensíveis a variações na iluminação do ambiente. Além disso, o processamento de imagens pode exigir alta capacidade computacional. Contudo, um sensor de visão oferece informações adicionais do ambiente, como cores e texturas.

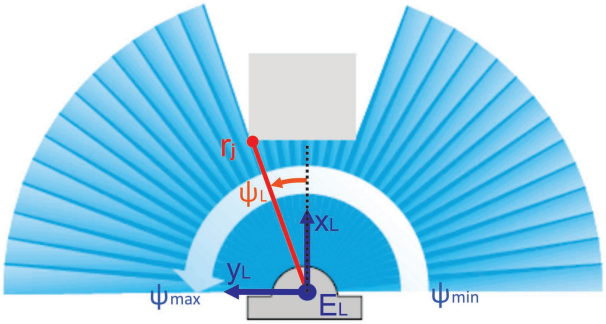


Figura 1.2: Diagrama de funcionamento do sensor *laser range finder*.

O sensor a laser, também chamado de *LIDAR* (*Light Detection And Ranging*), mede a distância entre o sensor e o objeto medido através da a diferença de tempo entre a emissão de um pulso de laser e a detecção do sinal refletido.

O pulso laser pode ser refletido em um espelho que rotaciona em torno de um eixo, e o tempo de trânsito de cada pulso pode ser associado com o ângulo do espelho no momento de sua emissão e detecção. Assim é o funcionamento de um *laser range finder* 2D, onde o laser é refletido em várias angulações no mesmo plano, e as distâncias são medidas.

No sensor 3D, o espelho pode girar em torno de dois eixos, realizando medições de distância em vários planos, como pode ser visto na Figura 1.3. Outro sensor a laser 3D conhecido é o Velodyne, que funciona de maneira semelhante ao LIDAR; a principal diferença é que, ao inves de utilizar apenas um feixe de laser, este sensor emprega 32 ou 64 feixes, permitindo a medição de múltiplos planos.

A varredura de um terreno por um *laser range finder* gera uma nuvem de pontos no espaço 3D. A representação de um terreno por pontos desconexos no espaço tridimensional dificulta a visualização da superfície real, e não oferece nenhuma informação adicional de natureza topológica (incidência, adjacência, conectividade), propriedades de natureza geométrica (declividade e orientação do terreno) e outras características que auxiliam nas diferentes aplicações de um modelo. Dessa forma, deve ser realizada algum tipo de aproximação permitindo a criação de uma superfície contínua no espaço tridimensional.

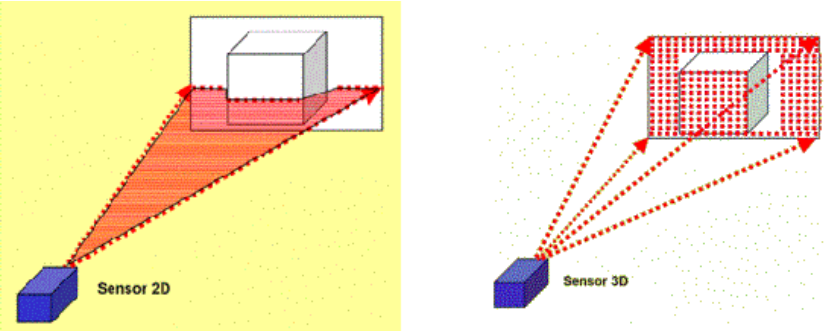


Figura 1.3: Comparação do funcionamento dos lasers 2D e 3D.

A forma mais comum de representação de terrenos *outdoor* é o 2 $\frac{1}{2}$ D *grid map* (Broten and Collier, 2006), (Hebert and Krotkov, 1993), (Bellutta et al., 2000), (Goldberg et al., 2002). Um mapa 2 $\frac{1}{2}$ D pode ser gerado pela seguinte regra de formação: $z = f(x, y)$, supondo z a elevação do terreno. Representações de terrenos em mapas 2 $\frac{1}{2}$ D são comumente referidos por Modelo Digital de Terreno (MDT), Mapa de Elevação Digital ou simplesmente Mapa de Terreno.

Uma ampla análise dos métodos usados para a modelagem de terrenos é apresentada em Hugentobler (2004). Mapas de elevação foram usados como uma eficiente estrutura de dados para representar terrenos complexos (Bares et al., 1989), (Pfaff and Burgard, 2006), (Hygounenc et al., 2004), (Parra et al., 1999).

Para aplicações relacionadas à robótica móvel, a formação de uma superfície contínua a partir da nuvem de pontos 3D também apresenta utilidades. Uma superfície representativa é útil na identificação de pontos de contato entre as rodas do robô e o terreno, auxiliando a análise de configuração de estabilidade do mesmo. A criação da superfície também pode facilitar a identificação de características do ambiente, que podem ser usadas em tomadas de decisão de robôs móveis, como não se aventurar em uma região onde foi identificada a presença de vegetação densa, ou então utilizar algum objeto (como uma árvore) como ponto de referência na localização (Siegwart and Nourbakhsh, 2004).

Uma superfície pode ser representada por um conjunto de planos adjacentes, e geometricamente, um plano pode ser definido por três pontos. Assim, um dos métodos mais utilizados na modelagem de terrenos a partir da amostragem de pontos é a triangulação de pontos, como exemplificado na Figura 1.4.

A criação de uma malha de triângulos não é um processo único, e permite soluções distintas. Uma das formas de triangulação mais utilizadas é a de Delaunay, devido à sua propriedade de maximizar o menor ângulo de cada triângulo. Isto significa que esse processo garante uma malha triangular o mais equiangular possível (Sibson, 1978).

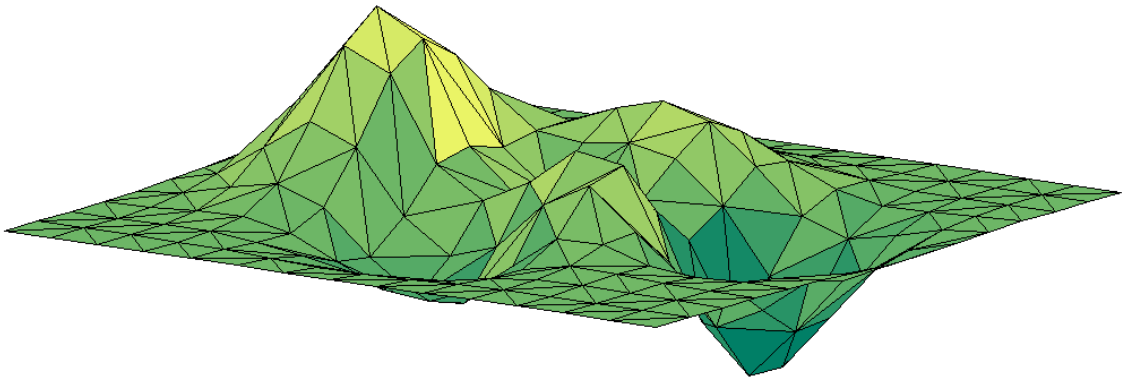


Figura 1.4: Exemplo de uma superfície triangulada.

1.3 Objetivo

O objetivo desse trabalho é apresentar uma metodologia para modelagem de terrenos naturais, de modo a gerar uma superfície 2 $\frac{1}{2}$ D. Para tal são utilizados dados coletados por um veículo autônomo equipado com *laser range finder* 2D e sensor de posicionamento de alta precisão. Estes dados experimentais foram obtidos pelo veículo ao percorrer um terreno natural de uma região agrícola, gerando uma nuvem de pontos no espaço 3D.

Deseja-se criar um modelo que represente apenas a superfície suportadora de carga (em inglês, *load bearing surface*) de um terreno natural. Nesse trabalho, denotaremos por superfície suportadora de carga o solo propriamente dito, visto que o foco dessa modelagem é possibilitar a análise da interação entre veículo e a superfície que suporta o seu peso.

Dessa forma, os pontos que não representam essa superfície devem ser excluídos no processo de modelagem.

No terreno percorrido, assim como em muitos terrenos naturais, a presença da vegetação evita que a superfície suportadora de carga seja lida diretamente pelo sensor a laser, e pontos que representam grama alta e arbustos são incorporados à nuvem de pontos. Logo, esses pontos devem ser filtrados adequadamente.

A obtenção de um modelo da superfície suportadora de carga viabiliza determinadas tarefas na robótica de campo, como a análise de estabilidade e o planejamento de trajetória. Essas aplicações do modelo criado são exemplificadas de maneira simplificada.

A estabilidade de um veículo está relacionada com o tombamento, um evento geralmente indesejável na robótica móvel. A análise realizada nesse trabalho objetiva calcular e prever a inclinação de um veículo em determinada posição de um terreno, para que possam ser evitadas configurações que ameacem o equilíbrio do sistema.

O planejamento de trajetória busca definir um percurso viável entre o ponto de partida e um ponto final, a partir de um mapa do ambiente. Nessa dissertação, a

aplicabilidade do método de modelagem de terrenos é exemplificada através de um método de ajuste de trajetória, a partir de um percurso realizado por um operador humano.

1.4 Metodologia empregada

A modelagem do terreno percorrido é realizada através da triangulação de Delaunay da projeção da nuvem de pontos em um plano horizontal, gerando uma malha triangulada no plano, que posteriormente é elevada e gera uma superfície 2 $\frac{1}{2}$ D. Essa modelagem foi implementada neste trabalho em um processo *offline* utilizando o software *Matlab*.

Após o devido tratamento dos dados, a nuvem de pontos é filtrada de modo a excluir do modelo os pontos que representam arbustos e grama alta. Essa filtragem é realizada a partir da análise de cada ponto individualmente, que é classificado como vegetação ou como terreno propriamente dito. O método de filtragem utilizado foi proposto em Vandapel et al. (2006). Nessa dissertação é proposta uma variação desse método, em que a incerteza dessa classificação é quantificada.

Os métodos utilizados na filtragem da vegetação e na triangulação dos pontos são detalhados no capítulo 4. A metodologia deste trabalho foi parcialmente apresentada em Sangreman et al. (2013).

1.5 Aplicabilidade do método

A técnica de modelagem pode ser adaptada para que seja executada em tempo real afim de auxiliar a navegação autônoma de um veículo. O filtro de vegetação faz a classificação entre vegetação e terreno comparando cada ponto com os demais. Dessa forma, a inclusão de novos pontos à nuvem de pontos original torna necessária uma nova análise de alguns pontos da nuvem original. A solução mais simples para esse problema é a subdivisão da nuvem de pontos em *clusters*, onde cada ponto só é comparado com pontos do mesmo *cluster*. *Cluster* é um conjunto de dados agrupados por alguma característica em comum. Nessa aplicação, as características analisadas são as coordenadas x e y dos pontos.

Essa estratégia pode reduzir sensivelmente o tempo de processamento no algoritmo de triangulação, o que favorece a sua aplicação para sistemas com aplicação em tempo real.

A separação em *clusters* pode gerar descontinuidades em suas fronteiras. Um procedimento que pode suavizar esse efeito é a criação de uma região em comum entre os *clusters*, de forma que os pontos na região de fronteira pertençam simultâneamente a dois *clusters*. Por exemplo: Se a coordenada x de P_i está entre 0 e 5.3 e a

coordenada y está entre 0 e 5.2, o ponto P_i fará parte do *cluster* k. Para haver uma superposição de *clusters*, o *cluster* $k+1$ localizado nas mesmas coordenadas em y (entre 0 e 5.2) deve compreender os valores de x entre 4.7 e 9.7, por exemplo. A região de interseção entre *clusters* deve ser proporcional à distância esperada entre pontos da amostragem.

No caso do filtro de vegetação, caso um ponto seja classificado como vegetação por algum dos dois *clusters*, ele deve ser considerado como parte da vegetação.

Caso a capacidade de processamento seja uma limitação, a separação em *clusters* também pode ser utilizada. A quantidade de pontos na nuvem pode ser reduzida ignorando varreduras do laser. Por exemplo, a cada dez varreduras, apenas uma é utilizada para a composição da nuvem de pontos. Contudo, esse procedimento reduz a resolução da amostragem.

1.6 Estrutura do trabalho

Essa dissertação é composta por 7 capítulos.

Este capítulo introduz o assunto abordado durante a dissertação, incluindo a motivação da pesquisa, os objetivos desejados, a metodologia proposta e comentários sobre sua aplicabilidade.

No capítulo 2 é apresentada uma revisão bibliográfica da modelagem de terrenos na robótica de campo. São apresentados trabalhos que realizam o mapeamento de ambientes através de diferentes tipos de sensores.

No capítulo 3 a triangulação de pontos é discutida, com destaque para a triangulação de Delaunay. Métodos e algoritmos de triangulação são apresentados.

Em seguida a metodologia utilizada é apresentada no capítulo 4, que inclui o tratamento inicial dos dados, transformações de coordenadas, filtragem de vegetação e triangulação dos pontos.

No capítulo 5 o experimento realizado é detalhado, e os resultados obtidos são descritos.

No capítulo 6 são apresentadas aplicações do modelo criado na robótica móvel. São apresentados procedimentos de análise da configuração do veículo e ajuste de trajetória a partir do percurso realizado pelo operador do veículo. Ambos procedimentos fazem uso do modelo de terreno criado a partir da metodologia proposta.

As conclusões desta dissertação são feitas no capítulo 7, apontando propostas de trabalhos futuros.

Capítulo 2

Levantamento Bibliográfico

Nesse capítulo é apresentada uma visão geral do mapeamento de terrenos realizado por robôs móveis, e em sequência alguns trabalhos são citados, onde são aplicados diferentes métodos de modelagem de ambientes por robôs móveis. Foram selecionados trabalhos representativos, com metodologias alternativas ao método utilizado neste trabalho. O objetivo desse levantamento bibliográfico é apresentar técnicas viáveis de modelagem de terrenos naturais empregadas por robôs móveis.

O mapeamento e localização simultâneos (SLAM) é um desafio atual da robótica móvel, e é tema de diversas pesquisas recentes. Dois artigos nessa área são apresentados; tais trabalhos foram selecionadas por apresentarem técnicas robustas e de publicações recentes.

O sensor utilizado para a coleta de dados nesta dissertação e na maioria das publicações apresentadas nesse capítulo é o laser. Contudo, também é comum a utilização da visão na modelagem de terrenos aplicada à robótica móvel. Por isso, uma seção desse capítulo aborda esse tema.

A modelagem de terrenos a partir da leitura por laser é uma técnica bastante utilizada no escaneamento aéreo. Diversas pesquisas já foram desenvolvidas nessa área, e na seção 2.4 é apresentado um trabalho representativo nessa área.

Em seguida, é abordado um tema central neste trabalho: a filtragem de vegetação na modelagem de terrenos. São apresentados trabalhos que realizam essa tarefa utilizando diversos sensores e criando diferentes representações do terreno. Dentre os sensores são utilizados para essa atividade, podem ser destacados o *laser range finder* 2D, o *laser scanner* 3D, a câmera digital e a visão estéreo.

Por fim, é realizada uma comparação de metodologias apresentadas neste capítulo com o método utilizado neste trabalho.

2.1 Mapeamento de terrenos na robótica móvel

Um dos problemas centrais na navegação de robôs móveis é a forma de sair do ponto inicial e atingir o ponto final. A localização torna-se uma questão relevante, assim como a trajetória a ser percorrida. A criação de um mapa que represente fidedignamente o ambiente é realizada para auxiliar nessas questões. No caso em que o ambiente é um terreno irregular, como em campos naturais, o mapeamento da superfície que suporta a carga pode se tornar necessária, visto que certas características do terreno podem facilitar, dificultar ou até inviabilizar a travessia do robô móvel.

O planejamento de trajetória de um robô submetido à restrições corresponde a tarefa de encontrar um trajeto admissível entre a configuração inicial e a configuração objetivo, de forma a não colidir com obstáculos. Para isso, algum tipo de mapa do ambiente é necessário.

Uma solução para o problema do planejamento de trajetória é dividir a tarefa de locomoção em sub-problemas menores, de forma a satisfazer as restrições do movimento e possibilitar sua otimização em algum aspecto. Na robótica móvel é comum que a trajetória escolhida tente minimizar não só o tempo de trajeto, mas também evitar regiões que podem ocasionar colisões ou capotamento.

2.2 Mapeamento e localização simultâneos

O mapeamento e localização simultâneos (SLAM - *simultaneous localization and mapping*) é uma técnica que consiste no mapeamento de um terreno desconhecido por um robô, ao mesmo tempo que esse realiza a sua própria localização. Essa tarefa ainda é considerada um desafio para a robótica atual, e em alguns trabalhos o SLAM é realizado com o auxílio de um operador humano através de comandos remotos. O operador, através de uma câmera embarcada, identifica características do ambiente a serem utilizadas como pontos de referência na localização e criação do mapa local. Esse é o caso dos dois trabalhos que são apresentados a seguir.

Os dois trabalhos apresentaram maneiras diferentes de representar o ambiente onde o robô se locomove. O mapa contínuo permite uma representação mais fidedigna à realidade, enquanto o mapa discreto reduz a precisão do modelo em troca de uma redução do esforço computacional.

2.2.1 SLAM com mapa contínuo

Ohno et al. (2010) realizaram a construção de um mapa 3-D em Disaster City (Figura 2.1), um local construído para treinamento que pertence a Agência Federal de Gerenciamento de Emergências dos Estados Unidos (FEMA). Foi utilizado o robô de resgate Kenaf, que se movimenta a partir de esteiras, e foram simuladas situações

de emergência. O robô é capaz de percorrer terrenos *indoor* e *outdoor*, e é dotado de um laser scanner 3-D.

A posição do robô é estimada através de odometria e de um giroscópio. Como o erro de medida da posição se acumula com o tempo, um sistema de correção do erro evita que ele cresça indefinidamente. A posição é ajustada quando o robô recebe um comando remoto designando essa tarefa. Através de verificações com o mapa formado pela nuvem de pontos 3-D, a leitura de algum formato característico é comparada com a sua leitura prévia, e o erro de posição pode ser dimensionado.

O mapa 3D é construído baseado na comparação de formatos característicos (*feature matching*). Quando o mapa 3D é construído manualmente, o operador escolhe o local onde é feita essa comparação baseado nas imagens transmitidas a ele. No experimento, ao atravessar passagens estreitas, o operador frequentemente enviava o comando para a realização dessa verificação, pois era uma região crítica para a locomoção do robô.

No experimento, o robô foi controlado por um operador remoto, e o sistema de localização era utilizado para que o operador identificasse a posição do robô no ambiente. Em situações nas quais o controle é feito autonomamente, o problema se torna mais complexo.

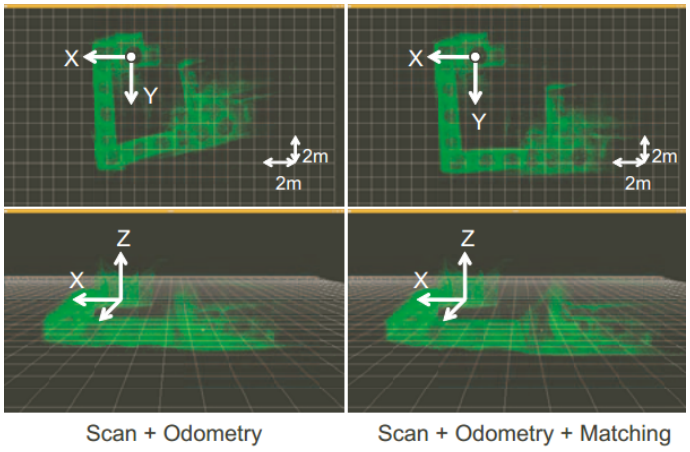


Figura 2.1: Mapa contínuo criado por Ohno et al. (2010), onde o robô realiza o SLAM.

2.2.2 SLAM com mapa discretizado em grid 2D

Também em Disaster City, Pellenz et al. (2010) realizaram a classificação de trechos do terreno como transitável ou não transitável, criaram um mapa 3D do local (Figura 2.2) e um sistema de mapeamento e localização simultânea foi realizado. O robô utilizado se movimenta através de esteiras, e possui sensores laser scanner 3D, GPS e sensor inercial (IMU). Para aquisição de dados, o robô foi controlado remotamente por um operador humano.

A classificação de terrenos foi feita utilizando o laser scanner 3D. Este gera uma nuvem de pontos 3D que é subdividida através de um grid horizontal 2D. O espaço é dividido em células 2D no plano horizontal e cada célula é classificada como transitável ou não transitável.

Se o tamanho das células utilizado for muito pequeno, haverá muitas células vazias, pois nenhum laser chegará a medi-las. Por outro lado, se o tamanho das células for muito grande e um obstáculo for detectado em seu interior, uma grande região é classificada como não transitável, o que pode obrigar o robô a realizar um desvio muito maior do que o necessário. Caso exista uma passagem estreita que necessite de um sistema de navegação preciso, o tamanho da célula pode classificar a passagem como não transitável.

O sistema de mapeamento proposto nesse trabalho tenta combinar as vantagens de células grandes e de células pequenas. O algoritmo verifica se na célula em questão existe algum ponto possa colidir com o robô de forma a tornar a célula uma região não transitável. Caso exista tal ponto, a célula é dividida em dois, e recursivamente repete a tarefa até que um tamanho mínimo de célula seja atingido. A célula é dividida no meio de sua maior aresta. Quando o tamanho mínimo é atingido, essa célula é classificada como obstáculo.

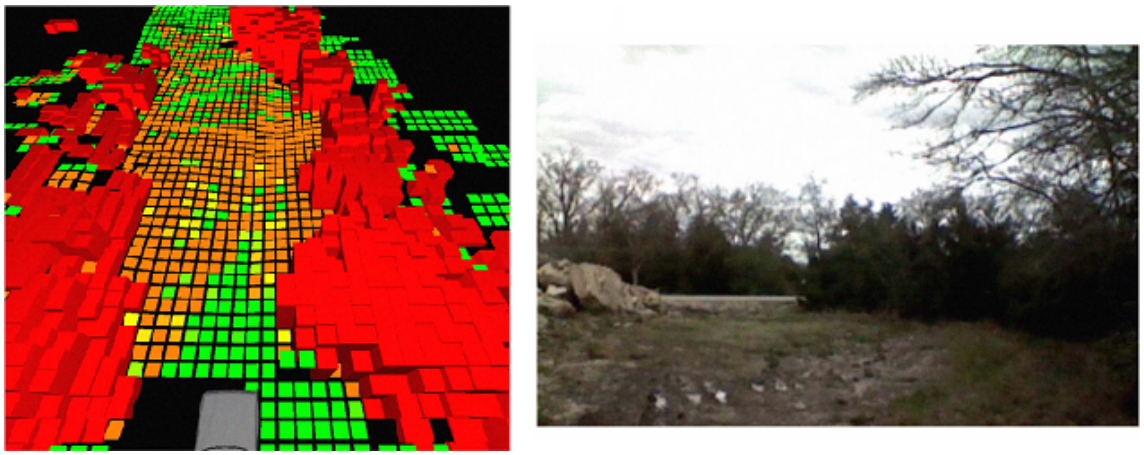


Figura 2.2: Mapeamento apresentado em Pellenz et al. (2010). É possível observar à direita a região mapeada, e à esquerda o mapa gerado dessa região

Além disso, a rugosidade do terreno também é avaliada, pois mesmo que uma região seja classificada como transitável, ela pode ser não ser ideal. A rugosidade aqui é indicada através da variância da altura local.

O laser scanner 3D gera 1,5 milhões de pontos por segundo. Para a realização do *feature matching* do SLAM, apenas pontos de áreas com obstáculos foram usadas, reduzindo o tempo de processamento.

O erro de posição é cumulativo, até que o robô volte a um ponto previamente mapeado. Nesse momento, o erro de posição é calculado, e seu valor é utilizado para

aprimorar o SLAM.

Um ponto importante observado nos experimentos é que com o sistema de SLAM utilizado (6D SLAM) não foi possível criar o mapa 3D completo em tempo real, por falta de capacidade de processamento do robô e por haver a necessidade da otimização de parâmetros que dependiam do ambiente em questão. Contudo, segundo os autores, a classificação de terrenos funcionou bem em tempo real, mesmo em áreas com alta rugosidade.

2.3 Mapeamento utilizando sensores de visão

A visão é uma ferramenta muito utilizada na modelagem de terrenos por robôs móveis. A utilização de imagens pode disponibilizar muitas informações sobre o ambiente, como iluminação, cor, tamanho e formato dos objetos, dando aos robôs mais informações para atuar com eficiência.

Nessa seção, são apresentados trabalhos que realizam a modelagem de terrenos em ambientes *outdoor*, utilizando duas abordagens muito comuns na robótica de campo: a leitura do ambiente por câmera simples e por visão estéreo.

São exemplificadas duas aplicações da modelagem de terrenos a partir de sensores de visão, e algumas das dificuldades e limitações inerentes a esse tipo de sensor podem ser observadas. A utilização da visão na modelagem de terrenos permite a obtenção de novas categorias de dados: as cores e as texturas. Esse tipo de sensor pode ser utilizado de forma complementar ao laser, como apresentado na seção 2.5.1.

Vale ressaltar que o trabalho apresentado na seção 2.3.2 também cria um modelo do terreno na forma de uma superfície 2 $\frac{1}{2}$ D.

2.3.1 Mapeamento utilizando câmera de vídeo digital

Shirkhodaie et al. (2005) utilizaram imagens do ambiente para extrair características (*features*) através da análise de texturas, e assim classificar o terreno (Figura 2.3). O foco do trabalho é a exploração planetária. Três técnicas são comparadas para analisar se o terreno é transitável ou não, e é apresentado um método de caracterização de texturas do terreno. Como o terreno possui muitos padrões de textura cuja definição matemática é complicada, optou-se por uma análise estatística para a classificação do terreno. Esta estratégia consiste em mapear as rugosidades do terreno em preto e branco e, através da variação da tonalidade, indicar os locais onde há rugosidade.

Muitos fatores contribuem para a limitação da observabilidade, como as condições ambientes (iluminação, sombras, umidade), características da câmera digital (resolução, sensibilidade, lentes e distorções), posição da câmera com relação ao

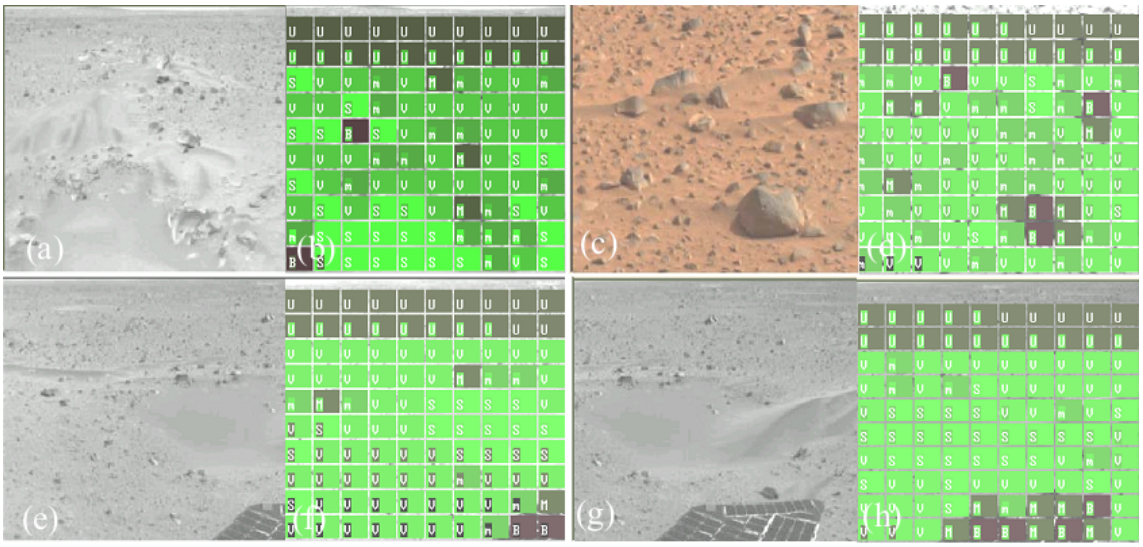


Figura 2.3: Classificação de terreno realizada por Shirkhodaie et al. (2005). Em cada par (a,b), (c,d), (e,f), (g,h) é possível comparar a imagem da região e o mapa de obstáculos criado.

plano do solo e irregularidades do terreno. É difícil estabelecer um modelo matemático conciso que ligue as características do terreno com as informações contidas nas imagens. Para lidar com esse problema, as propriedades da textura do terreno classificadas foram ajustadas com um fator que pesa a distância da imagem, asserindo mais incerteza a pontos mais distantes do observador. Esse método reduz as possibilidades de trajeto do robô, mas é uma abordagem mais confiável e conservadora.

2.3.2 Mapeamento utilizando visão estéreo

Sarkar et al. (2013) usam um conjunto de robôs para construir um mapa de elevação do terreno (superfície 2 $\frac{1}{2}$ D) usando visão estéreo, como ilustra a Figura 2.4. O sistema utiliza odometria em tempo real, computação em nuvem e algoritmos de recuperação de situações em que ocorra perda de comunicação. O sistema também foi desenvolvido com foco na exploração planetária, e foi implementado um sistema de odometria visual para a navegação. O sistema foi testado com apenas dois robôs.

O conjunto de robôs tem o comportamento análogo ao de um enxame (*swarm*). Esse comportamento foca na criação de uma rede sem fio onde os robôs podem colaborar com a computação em nuvem e construir o mapa mais facilmente. Esse sistema de mapeamento tem as vantagens de ser robusto, capaz de realizar processamento em paralelo e ser escalonável, ou seja, facilita a adição de mais robôs ao enxame.

poder ser aumentado facilmente adicionando mais robôs ao enxame.

Um dos robôs possui um sistema de processamento central de alta capacidade,

possibilitando o processamento de imagens em tempo real e sendo capaz de implementar complexos algoritmos de enxame através de uma rede sem fio. Essa tarefa necessita de muito mais processamento computacional do que o microcontrolador de cada robô pode realizar. Os robôs sempre seguem os comandos do controlador central, exceto quando ocorre a perda de comunicação.

Dois fatores principais que afetam a eficiência desse sistema de mapeamento autônomo são a velocidade de renderização do mapa pelo sistema central e a velocidade com que cada robô consegue se comunicar com o processador central enviando imagens estéreo e recebendo comandos de locomoção.

O conjunto de robôs consegue mapear qualquer terreno desconhecido. Os robôs *scout* são usados para a obtenção de imagens estéreo e seu posterior envio ao robô *skipper*, que carrega o processador central. O *skipper* cria então um mapa de disparidades (para a odometria visual) com todas as imagens estéreo recebidas dos *scouts*, armazenando as informações em uma arquitetura *octree* e gerando um mapa de elevação do terreno. Baseado nesse mapa criado, são designados novos comandos aos *scouts* para que esses continuem a exploração de forma a otimizar a velocidade de mapeamento.

O sistema também tenta fazer com que o *skipper* percorra o caminho mais seguro possível. Os *scouts* têm um sistema para se manter sempre dentro do alcance de comunicação com o *skipper*.

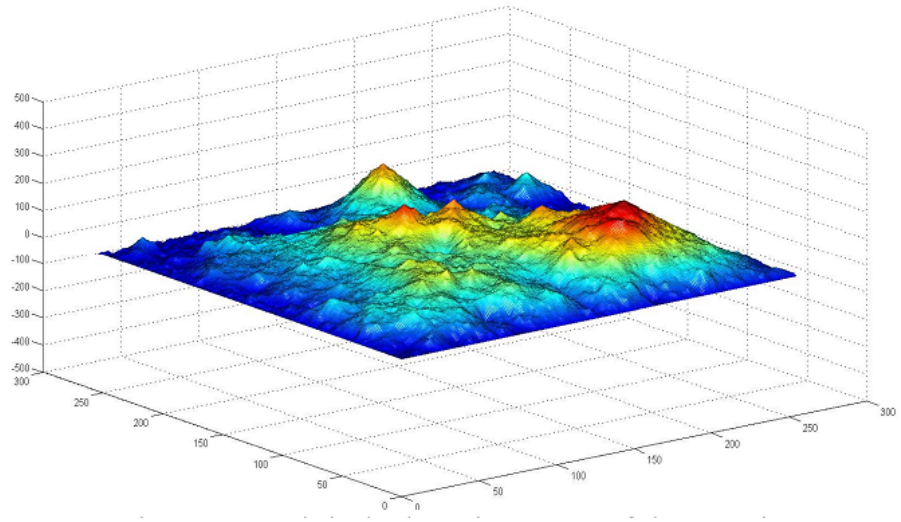


Figura 2.4: Superfície 2 $\frac{1}{2}$ D criada para representar o terreno no trabalho de Sarkar et al. (2013).

2.4 Escaneamento aéreo por laser

Uma área da modelagem de terrenos merece destaque: o escaneamento aéreo por laser (*Airborne Laser Scanning* - ALS), onde é feita uma varredura aérea de uma

região utilizando sensor a laser. Muitas pesquisas já foram desenvolvidas nessa área, e mesmo não estando diretamente ligadas à automação de veículos terrestres, podem ser aplicadas na robótica de campo. A análise da leitura de sensores em um terreno feita por robôs móveis apresenta semelhanças ao procedimento feito na classificação de terrenos por escaneamento aéreo.

Sithole and Vosselman (2003) focam o mapeamento de terrenos por ALS. Neste relatório, uma comparação é feita entre métodos de processamento dos dados dessa varredura, onde o objetivo é criar um modelo do terreno percorrido e classificar os objetos varridos. A classificação distingue entre terreno propriamente dito e objetos acima dele, como casas e árvores (Figura 2.5).

Muitos algoritmos de filtragem foram desenvolvidos nessa atividade. Para avaliar a performance dos algoritmos foram utilizados oito diferentes conjuntos de dados. Além de determinar a performance de cada filtro, o estudo também analisa a influência da resolução dos pontos na filtragem e faz um panorama das pesquisas na área.

No teste, os resultados de cada filtragem foram comparados com dados de referência gerados a partir da filtragem manual dos dados de escaneamento aéreo por laser, sendo feita a distinção entre solo (superfície suportadora de carga) e objetos. Posteriormente, os objetos são classificados entre soltos ou presos ao solo.

Figura 2.5: Filtragem de vegetação por escaneamento aéreo realizado por Sithole and Vosselman (2003).

Existem vários outros trabalhos nessa área, e muitas analogias podem ser feitas entre eles e os desafios da modelagem de terreno na robótica de campo. A transposição de metodologias do ALS para a robótica móvel é uma tarefa complexa e merece pesquisa e experimentos dedicados a essa questão.

2.5 Mapeamento de terrenos com vegetação

Em terrenos naturais, é comum a presença de vegetação. Evitar áreas com vegetação pode tornar o trajeto mais longo, ou até inviabilizá-lo, visto que o caminho através da vegetação pode ser o único percurso viável.

A locomoção de robôs autônomos em terrenos com vegetação é um tema recorrente em artigos científicos na área da robótica de campo. A principal dificuldade

reside em capturar, identificar e representar as diferentes formas e características desse tipo de ambiente. A modelagem de um solo irregular sem vegetação já é uma tarefa não trivial. Quando esse solo está coberto por grama e arbustos, a complexidade da tarefa aumenta consideravelmente. Esses elementos da vegetação formam superfícies porosas, prejudicando a leitura do solo ou de objetos que estejam cobertos por eles. Sensores como câmeras estéreo ou laser range finders não medem diretamente a superfície suportadora de carga. Além de obstáculos, uma região com grama alta e arbustos pode esconder um terreno que exceda o limite de inclinação do veículo.

Uma tarefa útil para lidar com essa questão é a classificação das regiões medidas pelos sensores. A classificação pode ser simples, e separar as regiões do terreno como transitável ou não transitável. Uma classificação mais complexa pode avaliar se o ponto medido pertence à superfície suportadora de carga, a algum obstáculo ou à vegetação.

2.5.1 Filtragem de vegetação com laser e câmera

Nguyen et al. (2011) propõem a utilização de uma câmera para a detecção da vegetação, utilizando a informação das cores do ambiente para facilitar a classificação do terreno. O sistema também utiliza leituras de laser range finder para a criação de uma nuvem de pontos 3D. Essa nuvem de pontos é segmentada em regiões, como ilustra a Figura 2.6. A distribuição de pontos local é então analisada para cada região, de forma que assim sejam detectados padrões de permeabilidade, visto que regiões com vegetação geralmente possuem essa característica. Em seguida, uma calibração é feita para que seja realizada a correspondência da região mapeada com a imagem colorida, e são criados padrões representativos de cores para cada região. A partir daí a vegetação é classificada.

O veículo utilizando nos experimentos apresentados possui dois *laser range finders*, um apontado para frente e outro inclinado, apontado para o solo (Figura 2.6). Uma câmera CMOS foi instalada apontada para frente, logo acima do laser. Os objetos analisados por esses dois sensores apontados para frente devem estar entre 3.8 metros e 15.8 metros de distância.

É feita uma fusão entre a nuvem de pontos 3D gerada pelo laser e as cores geradas pela câmera para realizar a localização da vegetação nas áreas mapeadas. O método possui grande robustez e pode ser utilizado em situações que apresentam dificuldades para a locomoção de robôs móveis.

Para classificar um conjunto de pontos no espaço 3D é realizada a decomposição em componentes principais (PCA) da matriz de covariância da distribuição espacial dos pontos, ordenada por autovalores de forma decrescente. Caso os pontos

representem um objeto de alta permeabilidade ao laser (como uma região com vegetação), não haverá uma direção dominante entre os pontos, então também não haverá grande diferença entre os autovalores. Caso os pontos representem uma estrutura linear, deverá haver apenas uma direção dominante, e o primeiro autovalor será consideravelmente maior do que os outros. Caso os pontos representem uma superfície sólida, o autovetor principal estará na direção normal à superfície, e os dois primeiros autovalores estarão próximos um do outro, mas distantes dos demais.

A dificuldade na aplicação desse método é o agrupamento dos pontos 3D, pois a nuvem de pontos que representa todo o ambiente deve ser dividida em *clusters*, onde cada *cluster* representa um objeto.

Para isso, as informações de cor da câmera são utilizadas. A nuvem de pontos 3D é projetada no plano de visão do veículo, e assim pode ser feita uma correlação entre os dados do laser e a imagem da câmera.

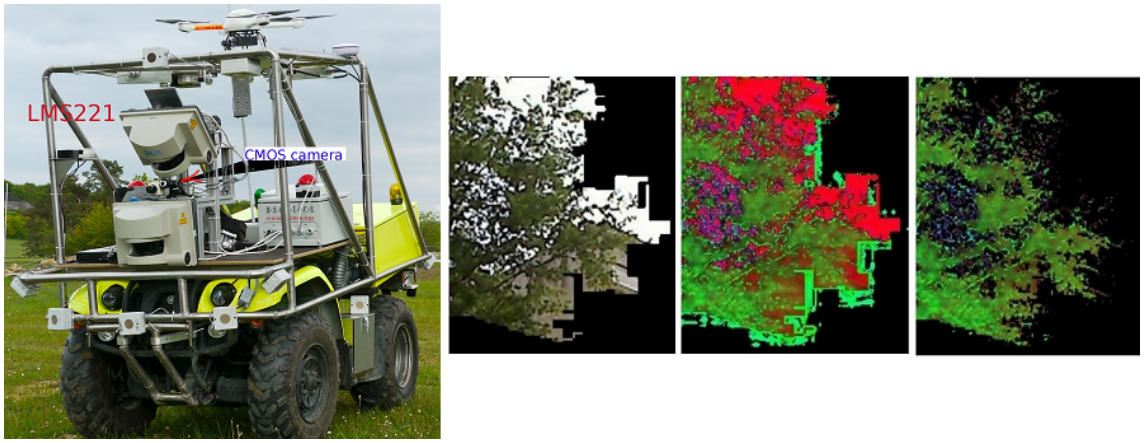


Figura 2.6: Filtragem de vegetação realizada por Nguyen et al. (2011). É possível observar à esquerda o veículo utilizado, e à direita o processo de segmentação de imagens.

2.5.2 Filtragem vegetação utilizando laser 3D com *clusters* de obstáculos

Schafer et al. (2008) apresentam uma metodologia de detecção de obstáculos para terrenos naturais com vegetação. A partir de um laser scanner 3-D, são identificadas características do terreno para construir a representação do ambiente. Também é apresentado uma abordagem para a cobertura de pontos cegos através do conceito de sensores virtuais.

Para testar a metodologia, foi utilizada a plataforma *off-road* Ravon, que possui uma variedade de sensores para a detecção de obstáculos. Dois laser range finders montados na horizontal (na frente e atrás) monitoram a segurança das proximidades. Um sistema de câmera estéreo e um laser scanner 3-D na parte frontal do robô são usados para a obtenção de informações do terreno até uma distância de 15

metros. Um sistema de pára-choques detectores de impacto foi usado para detectar obstáculos cobertos pela vegetação.

Se a região à frente do veículo tiver sido classificada como vegetação densa e flexível, o robô reduz a sua velocidade para o modo "caminhar suavemente", para evitar uma colisão em alta velocidade com possíveis obstáculos escondidos pela vegetação.

A defleção no pára-choque é constantemente monitorada como um indicador de obstáculos escondidos pela vegetação. Caso a colisão ocorra, o robô retrocede e tenta descobrir outro caminho.

O robô possui tração nas quatro rodas com quatro motores separados. Para a direção, as rodas da frente e as de trás giram, em dois pares independentes. Isso permite manobras, como realizar curvas com raio pequeno, ou então realizar uma translação lateral sem rotação. O veículo tem as dimensões de um automóvel urbano, pode subir e descer rampas de até 45 graus a uma velocidade máxima de 2 metros por segundo.

O escaneamento por laser é feito em um plano perpendicular ao solo, montado em um mecanismo que pode rotacionar 65 graus para a esquerda ou para a direita. As leituras do laser são armazenadas em uma memória ligada a um ângulo de visão ligado a sensores virtuais. Assim é feita uma interface genérica para o sistema de controle do robô baseado em comportamento (*behavior-based control*).

A estimativa da superfície suportadora de carga é feita levando-se em conta a capacidade do veículo de subir ou descer rampas. Assim, o terreno pode ser classificado como potencialmente transitável, e os possíveis obstáculos podem ser analisados quando o veículo estiver próximo o suficiente deles.

Mesmo coberta por vegetação, a superfície suportadora de carga pode ser medida desde que a vegetação não seja muito densa, de forma que alguns feixes do laser passem por ela e atinjam o solo (Figura 2.7). Os pontos de contato das rodas da frente com o solo são usados como pontos de referência na formação da superfície, juntamente com os pontos medidos pelo laser que atravessaram a vegetação.

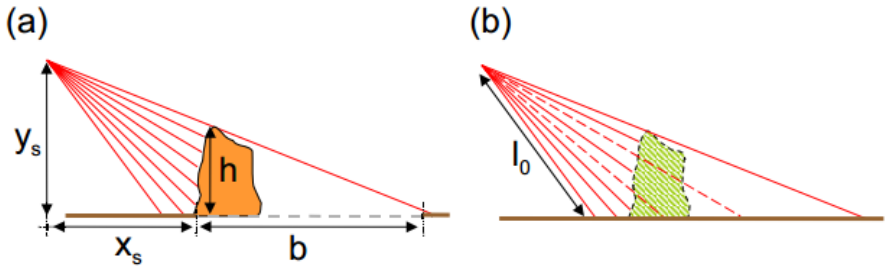


Figura 2.7: Filtragem de vegetação realizado em Schafer et al. (2008). A figura à esquerda (a) representa a leitura de objetos sólidos, e a figura à direita (b) representa a leitura de objetos permeáveis.

Pontos classificados como solo são considerados pontos seguros, e são utilizados para a medição da rugosidade do terreno. Pontos abaixo do nível do solo podem ser classificados como obstáculos negativos.

Para estimar a rugosidade do terreno, é realizada uma regressão linear com pontos que representam o solo, em uma região com o comprimento da ordem de grandeza do veículo. As retas obtidas não são concomitantes com o terreno modelado. Elas são utilizadas para se computar a variância de cada reta, como uma medida da rugosidade do terreno.

Pontos classificados como abaixo do solo são considerados obstáculos negativos se a sua profundidade exceder o comprimento do raio da roda do veículo. Se o ângulo de inclinação do solo exceder a capacidade máxima do robô, essa região também é registrada como obstáculo negativo. O análogo é feito com regiões de inclinação positiva acima da capacidade do robô.

Os obstáculos são agrupados em *clusters*. Na Figura 2.8 pode-se observar os círculos cinza que representam solo seguro, e os pontos coloridos representam vários tipos de obstáculos

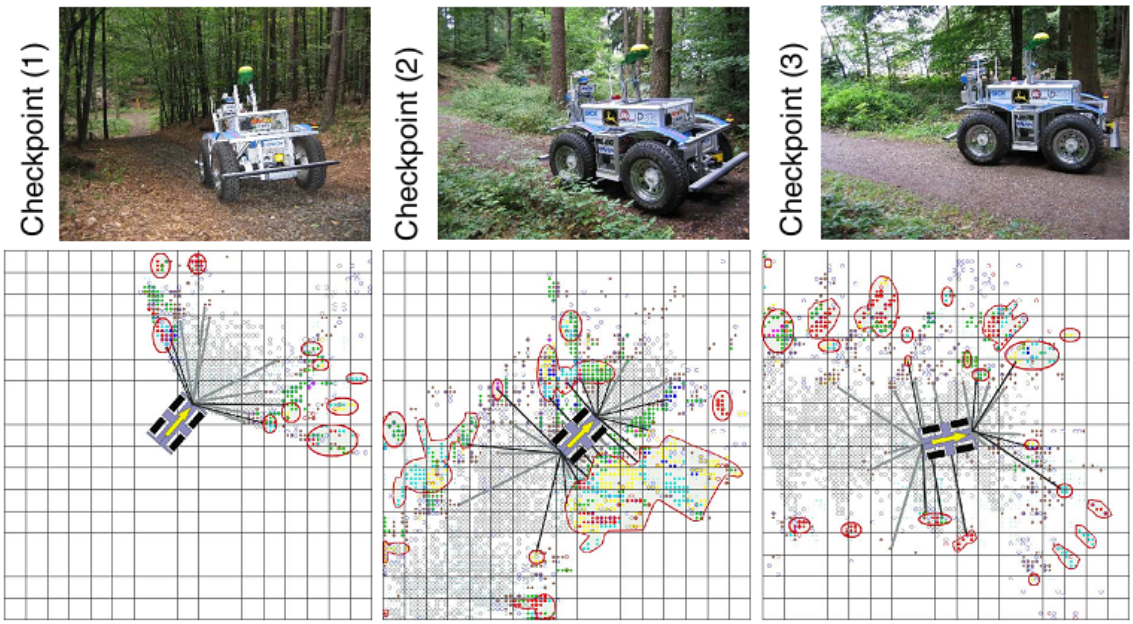


Figura 2.8: Filtragem de vegetação realizado por Schafer et al. (2008). É possível observar nas figuras acima as imagens de três regiões mapeadas, e nas figuras abaixo as suas respectivas representações geradas pelo veículo.

2.5.3 Filtragem de vegetação com laser 3D através de modelagem por *voxels*

Lacaze et al. (2002) descrevem um controlador autônomo do veículo XUV patrocinado pelo exército dos Estados Unidos. O controle foi criado para evitar obstáculos

na travessia de ambientes outoor não estruturados. Comandos de alto nível enviados pelos soldados são automaticamente refinados pelo sistema de controle e localização do veículo.

O principal sensor do veículo é o laser range finder. Ele também possui visão estéreo, unidade de navegação, GPS, detector de colisão e sensores na suspensão.

Através do laser range finder, é criado um mapa de *voxels* (unidades de volume no espaço 3D de conceito análogo ao de *pixels* no plano 2D). Os *voxels* armazenam informações de transparência ou opacidade. A transparência pode ter um valor intermediário, visto que em um determinado tempo, um *voxel* pode ser atravessado pelo laser, mas em outro determinado tempo de medição, o laser pode ter sido refletido no mesmo *voxel*. Através do mapa de *voxels* e suas respectivas transparências, é determinado qual é a superfície de suporte de carga.

O nível de transparência de um *voxel* é utilizado também para determinar a sua densidade. É assumido que um *voxel* menos transparente tem mais chances de ser sólido do que um mais transparente. Essa suposição é falha, pois uma porta de vidro ou uma cerca de arame provavelmente não seriam classificados como sólidos. Dessa forma, para melhorar o algoritmo de classificação de uma região como transitável ou não transitável, são utilizadas câmeras para a obtenção da cor de cada *voxel*.

A superfície de suporte de carga é usada para determinar a estabilidade e a rugosidade do terreno. Também utilizando o laser, foi feito um sistema de detecção de obstáculos à distância (até 60 metros).

Funções de custo são associadas a cada trajetória potencial, levando em consideração, por exemplo, a distância e as acelerações laterais. Um custo também é associado ao terreno a ser atravessado, que inclui a superfície de suporte de carga e a densidade do terreno a ser percorrido para seguir cada trajetória. Assim, são calculados os ângulos *pitch* e o *roll* que o terreno causa no veículo ao longo de cada trajetória. Os termos em inglês *roll*, *pitch* e *yaw* são rotações com respeito aos eixos *x, y e z* respectivamente, conforme ilustrado na Figura 2.9:

O modelo também é usado para calcular a quantidade de densidade nos *voxels* que o veículo precisa atravessar. Outros termos na função custo incluem a rugosidade que cada roda terá que atravessar no caminho, além da verificação de objetos com altura suficiente que possam colidir com o veículo.

Outro fator importante para a função custo é se a superfície suportadora de carga a entrar em contato com a roda já foi medido pelo laser. Caso esse terreno esteja coberto pela vegetação, a avaliação de custos do planejamento de trajetória associará a esse um custo mais elevado.

Considere um veículo em movimento com um laser 3D embarcado e uma região coberta por vegetação, em que o sistema de mapeamento não é capaz de medir sua superfície a uma certa distância. Quanto mais o veículo se aproxima da região,

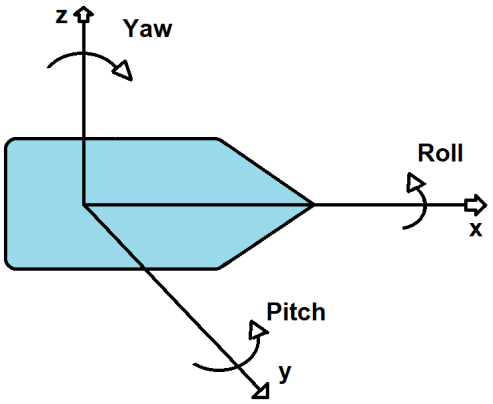


Figura 2.9: *Roll*, *pitch* e *yaw*: ângulos de rotação com respeito aos eixos x, y e z respectivamente.

menos tempo o robô terá para medir a sua superfície. Dessa forma, a probabilidade de que regiões venham a ser medidas antes de entrar em contato com o robô diminui com a aproximação do veículo. Por isso, ao se aproximarem do veículo as regiões não observadas têm seus custos elevados de uma forma não linear com a distância.

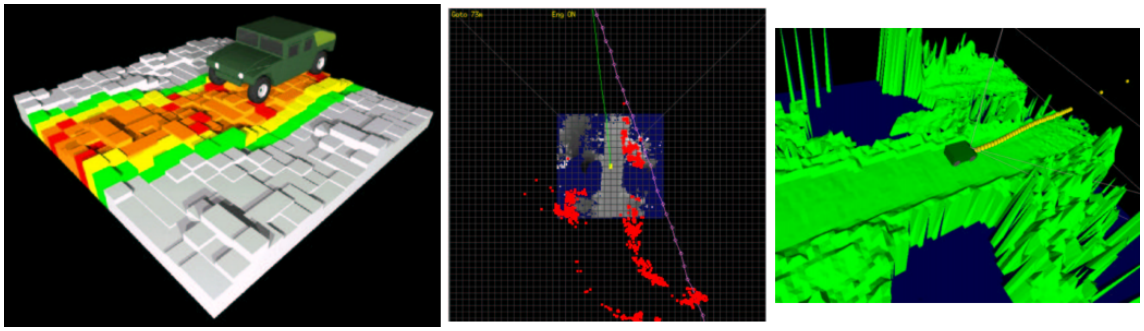


Figura 2.10: Geração de mapas e planejamento de trajetória com função custo realizado por Lacaze et al. (2002)

As trajetórias são inicialmente formadas por segmentos de reta, e subseqüentemente têm seu formato suavizado.

Na Figura 2.10 é mostrado o veículo atravessando uma ponte numa área florestada. A figura central mostra o planejamento de trajetória inicial em roxo, que não leva em conta a existência da ponte. Um segundo sistema de planejamento de trajetória faz o refinamento do percurso.

Ao final dos experimentos, pôde ser observado que, em relação ao sistema antigo de navegação, houve ganho de desempenho em regiões com vegetação.

2.5.4 Classificação de vegetação e identificação de obstáculos cobertos através de laser 2D e 3D

No trabalho apresentado por Vandapel et al. (2004), o foco foi a segmentação de uma nuvem de pontos 3D gerada por um laser em três categorias: pontos espalhados que representam grama e folhagem, pontos que formam estruturas lineares como arames ou galhos de árvores e pontos ditos como superfícies sólidas, como pedras, troncos de árvores e a superfície do terreno. Dados similares são agrupados em regiões que concentram pontos de mesma classificação, que correspondem a grandes trechos de superfície. O método apresentado não depende da forma específica de leitura do sensor. A representação do terreno é atualizada sempre que um novo dado é adiquirido pelo sensor, independente da ordem de aquisição. São utilizadas técnicas estatísticas de classificação, e a maioria dos parâmetros são definidos manualmente. Foram utilizados laser range finder 2D e 3D.

O mapa é formado por *voxels* de 10 centímetros cúbicos, e a sua transparência é definida através dos pontos contidos em seu interior. Nesse sistema, nenhum dado é descartado, mas ao mesmo tempo foi atingida uma considerável redução na quantidade de dados armazenados na memória.

A implementação desse sistema é feita por dois processos assíncronos. O primeiro é a atualização, que consiste em incorporar um novo ponto ao modelo, e é feita contínuamente. O segundo é a classificação dos dados, e é feito em intervalos regulares, ou quando solicitado.

Essa abordagem permite que a incorporação de dados seja feita em tempo real no experimento, tão logo eles sejam enviados pelo laser range finder. A classificação é realizada a cada três segundos ou a cada 7 metros percorridos.

A classificação entre superfícies porosas, planas e lineares é feita de forma análoga a apresentada na seção 2.5.1. Pontos que representam uma superfície plana de espessura não nula foram misturados a pontos distribuídos aleatoriamente, para avaliar a capacidade do algoritmo de identificar corretamente a superfície sólida em meio à nuvem de pontos. A superfície imersa nos pontos pôde ser identificada pelo algoritmo (Figura 2.11).

2.5.5 Classificação terreno com vegetação utilizando laser 2D

Utilizar um laser range finder 2D apresenta uma grande desvantagem se comparado ao uso do laser com capacidade de escaneamento 3D: a quantidade de informação disponível é muito menor. Contudo, o custo, o peso, o consumo de energia e a necessidade de processamento podem justificar a utilização do laser range finder 2D.

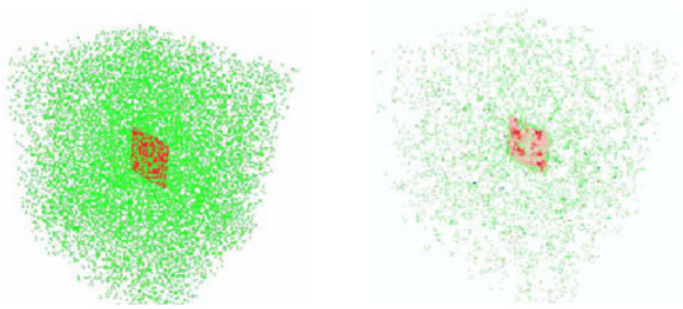


Figura 2.11: Classificação de pontos realizada por Vandapel et al. (2004). Pontos que representavam uma superfície plana foram imersos em uma nuvem de pontos aleatórios. O algoritmo foi capaz de distinguir a superfície.

O trabalho realizado por Andersen et al. (2006) propõe um algoritmo de classificação que discrimina entre terreno transitável e terreno não transitável através de dados obtidos por um laser range finder 2D. Os experimentos foram realizados em estrada de terra e estrada de asfalto, com vegetação nas extremidades.

O sensor a laser é montado na parte superior do veículo e direcionado para o solo poucos metros à frente do veículo. Os pontos gerados são agrupados de acordo com sete características: altura, rugosidade, tamanho do degrau, curvatura, inclinação, largura e dados inválidos. Esses parâmetros são usados para a identificação dos limites da estrada por onde o veículo anda, e discernir o terreno transitável e os obstáculos.

No trabalho é discutido qual a inclinação ótima do sensor a laser 2D, tendo em vista o tamanho mínimo do obstáculo a ser detectado, a frequência de leitura do sensor, a velocidade máxima que ele irá se locomover e capacidade de frenagem do veículo. Para o veículo utilizado, foi encontrado o valor ótimo para o ângulo de inclinação com a horizontal de 9°, com o sensor a 41 centímetros de altura, com uma velocidade máxima de 2,6 m/s. Nessa configuração, o robô faz a leitura do terreno a 2,6 metros de distância do veículo.

A classificação é feita utilizando os pontos de uma única varredura 2D, conforme ilustrado na Figura 2.12. Para classificá-los, o algoritmo submete-os a sete testes. Primeiramente é verificado se o ponto é válido, pois algumas superfícies (como a água por exemplo) não são detectadas pelo laser. Para agrupar os pontos válidos em segmentos transitáveis, os pontos são separados inicialmente em grupos homogêneos, com rugosidade e alturas semelhantes. Esses grupos são combinados através dos classificadores de tamanho de degrau e de curvatura. Após isso, os grupos restantes são filtrados com base na inclinação e na amplitude de cada um, e finalmente formam o conjunto de segmentos transitáveis.

Para passar no teste da altura, um ponto deve estar no máximo a 20 centímetros acima e no mínimo a 50 centímetros abaixo do veículo. Em seguida, a rugosidade

é verificada, que foi definida como o desvio padrão local das distâncias de reflexão ao longo de uma leitura. Um algoritmo de decomposição em valores singulares não foi usado por ser considerado computacionalmente pesado. O cálculo da rugosidade é feito de forma que perfis de terreno que formam curvas suaves com relação ao tamanho do robô gerem valores baixos de rugosidade.

Após o agrupamento de pontos em seções de rugosidade e alturas homogêneas, é verificado o tamanho do degrau de cada grupo. Esse teste assegura que dois segmentos não sejam agrupados juntos caso estejam separados por um obstáculo, ou possuam alturas diferentes. Esse seria o teste responsável por não combinar a calçada com o asfalto de uma rua, mesmo os dois sendo classificados como transitáveis.

O teste de curvatura verifica a inclinação de uma estrada com perfil convexo ou côncavo. Caso seja côncavo, o ângulo máximo de inclinação lateral (*roll*) é de 3°. Caso seja convexo, esse limite passa para 10°.

Para finalmente ser classificado como transitável, um grupo de pontos deve possuir uma inclinação longitudinal (*pitch*) menor do que 10° e largura maior do que a largura do robô.

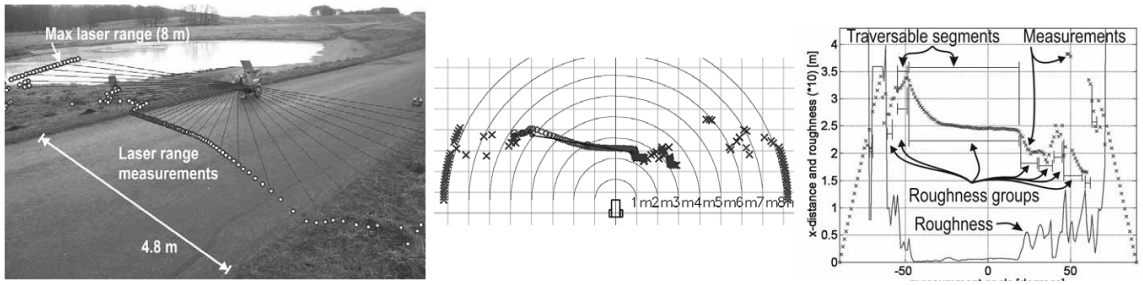


Figura 2.12: Classificação de dados do Laser 2D realizado por Andersen et al. (2006). É possível se observar à esquerda o robô realizando a medição, ao centro os pontos medidos e à direita um gráfico com a rugosidade da mesma medição.

2.6 Comparações entre mtodologias apresentadas e adotada neste trabalho

Há uma variedade de sensores viáveis para a modelagem de terreno, e diversos tipos de modelo de terreno aplicáveis à robótica móvel. Os trabalhos apresentados nesse capítulo demonstram metodologias alternativas de modelagem de terrenos.

Um método de modelagem que pode ser observado em alguns trabalhos apresentados neste capítulo é a modelagem discreta. Esta possui a vantagem de necessitar de menor processamento computacional, mas apresenta uma sensível perda de precisão quando utilizada para a análise de estabilidade, tornando a abordagem contínua mais indicada para essa atividade. Além disso, a abordagem discreta pode gerar

descontinuidades na modelagem de terrenos com grande inclinação.

O método proposto nessa dissertação cria uma superfície 2 $\frac{1}{2}$ D contínua. Ela se apresenta compatível com a varredura do terreno por um *laser range finder* 2D apontado para o solo, visto que detalhes de objetos que não compõem a superfície suportadora de carga não são modelados.

A utilização do laser 2D para a coleta de dados apresenta vantagens se comparado a outros sensores, como:

- Robustez a variações de condições do ambiente como iluminação e sombras
- Necessidade de processamento de dados menor do que de outros sensores
- Relativo baixo custo, se comparado ao *laser scanner* 3D

A montagem do sensor na parte superior do veículo apontada para o solo à sua frente permite que o mesmo sensor faça a varredura do terreno e ao mesmo tempo possa ser usado para a detecção de obstáculos.

Capítulo 3

Triangulação de Pontos

3.1 Modelagem através de triangulação

A reconstrução de um terreno por triangulação de pontos não é uma ideia nova. Malhas triangulares têm sido aplicadas na modelagem de terrenos naturais para reduzir a informação geométrica disponível à sua forma mais básica, procedimento comumente utilizado na aerofotogrametria (Fayek and Wong, 1996).

Uma vantagem de malhas triangulares, como ilustrado pelos dados modelados em Bakambu et al. (2006), é que a resolução pode variar com a quantidade de informação, podendo levar em conta a complexidade do terreno.

É recomendado que o método de triangulação escolhido seja aquele em que a superfície obtida seja a mais próxima possível do terreno original. Foi observado que triangulações que levam a boas interpolações tendem a evitar triângulos longos e finos (Barnhill, 1977). Quanto maior for a aresta de um triângulo, maior a distância entre os pontos que estão sendo conectados. Na ausência de informações extras, conectar pontos mais próximos tende a gerar interpolações melhores. Dessa forma, a triangulação de Delaunay se apresenta como uma eficiente forma de representação, visto que possui a propriedade de maximizar o menor ângulo interno dos triângulos.

A abordagem proposta aqui não leva em conta a altura de cada ponto no ato da triangulação, por isso também é chamada de abordagem independente dos dados. Uma boa motivação para essa abordagem é apresentada em Rippa (1990), que prova que a triangulação de Delaunay minimiza a rugosidade do modelo do terreno resultante, independente da altura dos pontos.

Aqui a rugosidade G está sendo defininda como a raiz quadrada da média aritmética dos quadrados dos desvios e de cada ponto a partir do plano médio

$$G = \sqrt{\frac{\sum_{i=1}^{n} e_i^2}{n}} \quad . \tag{3.1}$$

Ao analisar a rugosidade de uma superfície contínua através dessa definição,

faz-se necessário a discretização dessa superfície em um número finito de pontos.

Existem pesquisas que levam em consideração a altura dos pontos para tentar descobrir triangulações melhores. Essa abordagem dependente dos dados foi proposta pela primeira vez em Dyn et al. (1990), onde são sugeridos diferentes critérios de custo para triangulações que dependem da altura dos dados. Vale observar que esse processo é iniciado a partir de uma triangulação de Delaunay e realiza iterativamente a inversão de arestas. A mesma abordagem é utilizada por Quak and Schumaker (1990), que observaram que suas triangulações geram pequenas melhoras se comparadas à triangulação de Delaunay nos casos em que tentam aproximar superfícies suaves, mas para superfícies não-suaves o método se mostrou menos eficiente.

Alguns trabalhos na área da robótica móvel que utilizaram a triangulação na modelagem de ambientes devem ser destacados. Wettergreen et al. (2010) construiram uma malha de triângulos para criar um modelo 3D do terreno a ser percorrido por um robô móvel que usa *laser range finders* 2D (Figura 3.1). A malha de triângulos é formada utilizando a triangulação de Delaunay 2D com restrições, uma variação da triangulação de Delaunay planar. Esse modelo é utilizado para duas tarefas: planejamento de trajetória e ajuste de configuração do robô.

No planejamento de trajetória, um algoritmo de minimização de custo é utilizado para calcular a trajetória ótima. Essa trajetória é posteriormente suavizada visando uma maior eficiência energética e menor tempo de trajeto.

No ajuste de configuração do robô, uma simulação da interação robô-terreno é realizada e, caso necessário, modificações na configuração são feitas para que não ocorra capotamento devido ao mal condicionamento do terreno a ser percorrido.

Figura 3.1: Modelagem de terreno por robô móvel através da triangulação de Delaunay (Wettergreen et al., 2010). O robô possui um laser range finder e constrói um modelo em tempo real.

Outra aplicação na área de robótica móvel é o trabalho apresentado por Hähnel et al. (2003), onde é realizada a reconstrução em 3D de ambientes antrópicos (afetados pela ação humana) *indoor* e *outdoor* através de *laser range finders* instalados em robôs móveis. Os pontos escaneados são ligados formando malhas de triângulos

que formam um modelo do ambiente.

3.2 Triangulação de Delaunay

A triangulação de um conjunto de pontos estabelece uma conexão entre objetos de dimensão 0 (os pontos), formando objetos de dimensão 1 e 2 (as arestas e as faces triangulares). Dessa forma, é criada uma estrutura topológica onde é possível obter informações úteis sobre o terreno, representado aqui por uma superfície 2 $\frac{1}{2}$ D.

Uma forma de garantir que a triangulação de pontos gere uma superfície desse tipo é realizar uma triangulação de Delaunay 2D, utilizando as projeções dos pontos originais em uma plano horizontal. Após a triangulação, as projeções dos pontos são elevados às suas alturas originais (Figura 3.2).

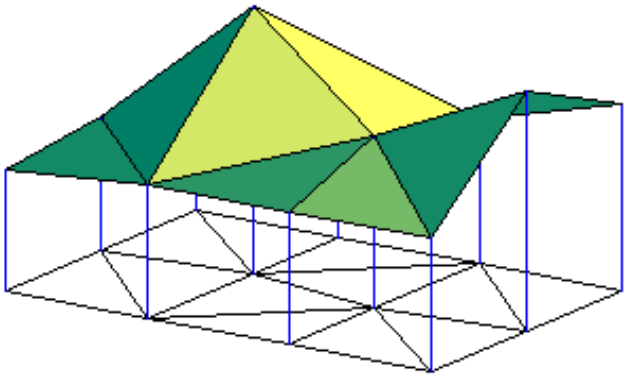


Figura 3.2: Triangulação gerando uma superfície 2 $\frac{1}{2}$ D.

A triangulação de Delaunay 2D não garante que o triângulo obtido, quando reprojetado no espaço tridimensional, seja necessariamente o mais equiangular possível. Ainda assim, esta variação do método possui consideráveis vantagens em comparação à triangulação tridimensional.

Segundo Rippa (1990), a triangulação 2D minimiza a rugosidade do modelo do terreno resultante, independentemente da altura dos pontos e da sua quantidade. Esta pode ser uma característica desejada durante a interpolação de pontos. Outras vantagens são a simplicidade de implementação, menor processamento computacional associado e maior literatura disponível. De fato, os trabalhos citados no início desse capítulo empregam a triangulação de Delaunay 2D.

A principal vantagem da triangulação de Delaunay 2D em relação à 3D é o custo computacional. Essa triangulação gera um mapa de elevação do terreno, uma superfície 2 $\frac{1}{2}$ D. Essa representação é compatível com a coleta de dados a partir de um laser range finder 2D direcionado para o solo, pois esta forma de leitura não possui a capacidade de distinguir objetos tridimensionais no espaço da mesma forma que o laser scanner 3D.

Um método alternativo é apresentado em Dyn et al. (1990), combinando triangulação 2D com otimização de uma função de custo calculada levando em conta a altura de cada ponto. Uma abordagem semelhante é utilizada em Quak and Schumaker (1990), podendo resultar em representações mais realistas com respeito à triangulação de Delaunay no caso de superfícies suaves.

3.2.1 Propriedades da triangulação de Delaunay

Além de maximizar o menor ângulo interno, a triangulação de Delaunay apresenta outras características fundamentais (Lee and Schachter, 1980):

- Se três pontos são vértices de um mesmo triângulo, o círculo que passa por esses três pontos não contém nenhum outro ponto em seu interior.
- Se dois pontos formam uma aresta, existe um círculo que passa por esses dois pontos e não contém mais nenhum ponto.

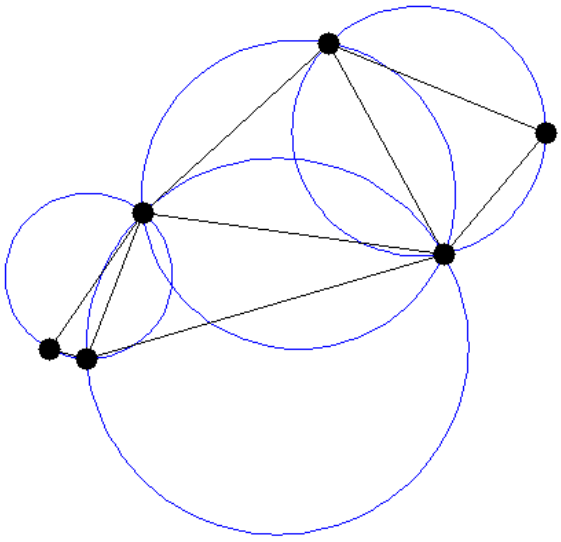


Figura 3.3: Triangulação de Delaunay de 6 pontos no plano. Vale observar que o círculo circunscrito a cada triângulo não contém nenhum outro ponto.

Essas propriedades são utilizadas na construção da malha triangular (Figura 3.3). Triângulos e arestas que não obedeçam às características fundamentais são chamadas de ilegais. Caso uma triangulação possua somente triângulos e arestas legais, trata-se de uma triangulação de Delaunay.

A triangulação de Delaunay não é necessariamente única. Caso no conjunto de pontos existam quatro ou mais pontos pertencentes a um mesmo círculo, mais de uma triangulação pode ser obtida. Caso contrário, a solução é única.

Dado um conjunto de pontos no plano, a triangulação de Delaunay corresponde ao grafo dual do diagrama de Voronoi do mesmo conjunto de pontos (Figura 3.4).

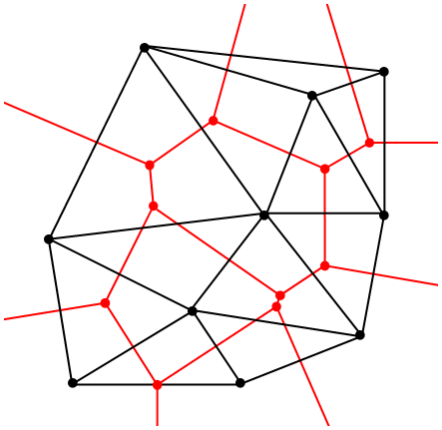


Figura 3.4: Analogia entre a triangulação de Delaunay e o diagrama de Voronoi. As arestas pretas pertencem à triangulação de Delaunay, e as arestas vermelhas formam o diagrama de Voronoi.

A relação entre a triangulação de Delaunay e o diagrama de Voronoi é assunto de algumas publicações, e aplicações de suas propriedades podem ser vistas em Watson (1981).

3.2.2 Construção da triangulação de Delaunay

Um quadrilátero convexo pode ser triangulado de duas maneiras invertendo-se a aresta interior conforme ilustrado na Figura 3.5.

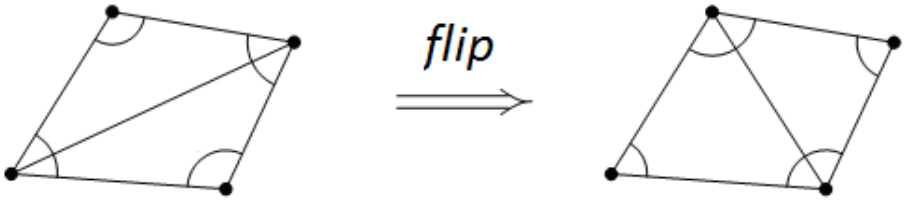


Figura 3.5: Flip de arestas em um quadrilátero.

A triangulação apresentada na figura 3.4 é obtida utilizando a técnica de Delaunay. É possível perceber que ela maximiza o menor ângulo dos triângulos. A inversão de arestas é chamada em inglês de *flip*. Uma implementação simples da técnica parte de uma triangulação qualquer dos pontos. Em seguida realiza-se o *flip* de arestas ilegais até que só existam arestas legais. Contudo, esta abordagem é pouco eficiente.

Métodos para a construção da triangulação de Delaunay já foram amplamente divulgados na literatura e uma grande quantidade de algoritmos para sua obtenção estão disponíveis. O método utilizado nesse trabalho para a triangulação foi o algoritmo incremental aleatório (*randomized incremental algorithm*) conforme proposto em Guibas et al. (1992). Esse é um algoritmo eficiente e sua implementação

é simples, com convergência na ordem de $n\ log(n)$, sendo n o número de pontos.

A modelagem começa pela criação de um primeiro triângulo auxiliar que irá englobar todos os pontos do modelo. Os pontos são ordenados de forma aleatória e adicionados ao triângulo auxiliar um de cada vez, gerando novos triângulos. Quando um ponto é adicionado, primeiramente é identificado a qual triângulo ele pertence. O novo ponto é ligado aos vértices do triângulo em que está contido, criando três novos triângulos. Em seguida, é verificado se as arestas dos novos triângulos são legais. Em caso negativo, é realizado um *flip* e as arestas envolvidas são novamente verificadas. Quando restarem apenas arestas legais, o próximo ponto é adicionado. O processo é repetido até que não existam mais pontos a se adicionar, quando então é concluída a triangulação.

O algoritmo implementado nesse trabalho utiliza uma estrutura de armazenamento de dados do tipo árvore, conforme sugerido em De Berg et al. (2008), composta de um elemento principal chamado *raiz* ligado a outros elementos chamados de *ramos*. Cada ramo pode levar a outros elementos, gerando novos ramos, até que atinja um elemento que não se ramifica denominado de *folha*. Dessa forma, é armazenado o histórico de criação dos triângulos e dos *flips* realizados, de maneira que o elemento da árvore que armazena cada triângulo também armazena um indicador para os triângulos que o substituíram. Para identificar o triângulo que contém um determinado ponto, parte-se da raiz da estrutura proposta, indo para os níveis seguintes até chegar ao último nível (folha) correspondente.

Exemplo de aplicação do algoritmo

Nesta seção, o algoritmo apresentado é aplicado a um conjunto de 10 pontos no plano, afim de exemplificar seu funcionamento.

O passo inicial é a adição de 3 novos pontos, de forma que esses formem um triângulo inicial que contém os 10 pontos originais.

Em seguida, é adicionado o primeiro ponto ao triângulo inicial. Ele é conectado aos vértices do triângulo inicial, gerando três novos triângulos.

A triangulação continua com a inclusão do segundo ponto. Agora, é necessário localizar qual dos três triângulos existentes contém o novo ponto. Após essa localização, o segundo ponto é inserido, e ele é conectado aos vértices desse novo triângulo, formando três novos triângulos. Após isso é verificado se as novas arestas formadas são arestas legais. Caso alguma não seja, é realizado o *flip* da mesma, e é verificado se alguma aresta que era legal passou a ser ilegal.

Na Figura 3.6 pode ser observado o procedimento de verificação de legalidade e *flip* de aresta realizado na inserção do segundo ponto da triangulação.

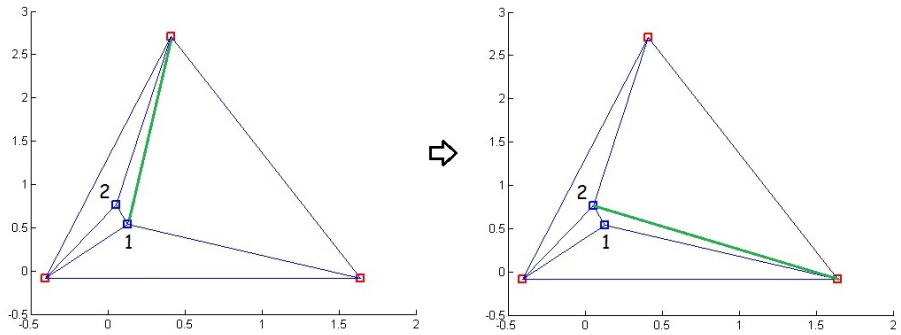


Figura 3.6: Legalização da aresta destacada em verde após a inclusão do segundo ponto na triangulação.

Com a adição do segundo ponto e a legalização de todas as arestas ilegais, os pontos seguintes são adicionados, e o procedimento realizado com o segundo ponto é repetido para todos os pontos subsequentes.

Ao final da inclusão de todos os pontos na triangulação, os 3 pontos originais e todas arestas ligadas a eles são excluídas, restando a triangulação de Delaunay 2D dos 10 pontos originais.

O processo completo de triangulação dos 10 pontos é ilustrado na Figura 3.7.

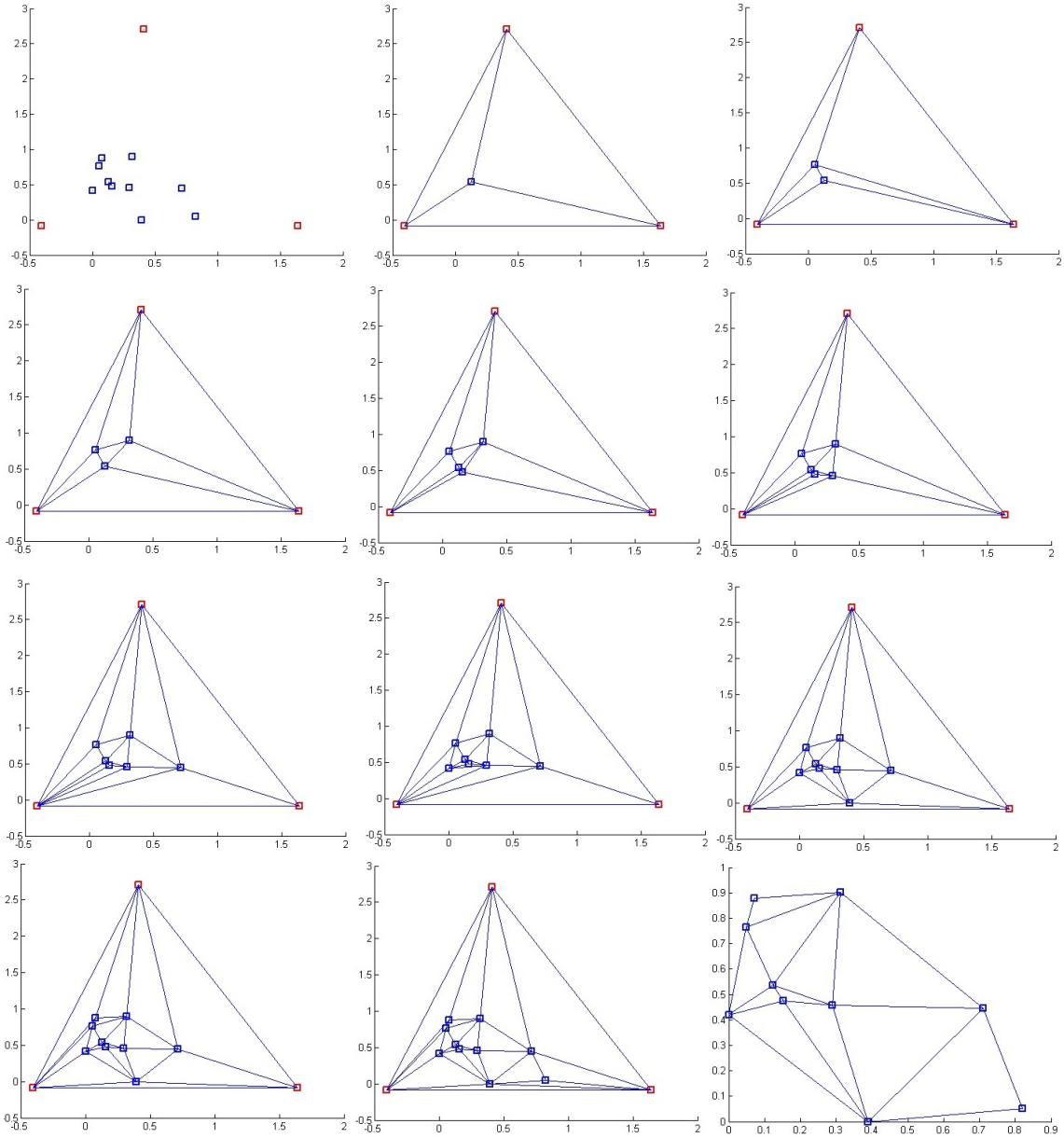


Figura 3.7: Etapas da triangulação de Delaunay de um conjunto de 10 pontos.

Após a triangulação plana, os pontos devem ser levados a suas alturas originais, mantendo a triangulação. Assim, é formada uma superfície $2\frac{1}{2}$ D, conforme ilustrado na Figura 3.8.

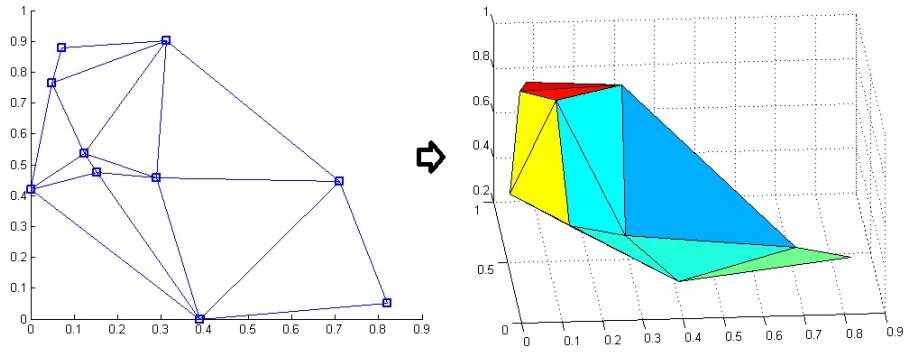


Figura 3.8: Verificação e legalização da aresta verde em destaque no triângulo verde.

3.3 Triangulação de Delaunay com restrições

A triangulação de Delaunay com restrições, também chamada de *Constrained Delaunay Triangulation* (CDT) em inglês, é uma variação da triangulação de Delaunay padrão.

Além do conjunto de pontos a ser triangulado, a CDT também recebe como entrada algumas arestas que deverão estar presentes na triangulação final. É comum que uma CDT contenha arestas que não satisfazem as propriedades da triangulação de Delaunay, fazendo com que a CDT não possa ser considerada como uma triangulação de Delaunay do conjunto de pontos.

Um popular algoritmo para a construção da CDT podem ser encontrado em Chew (1989).

A CDT 2D é um processo utilizado na robótica para a construção de mapas onde a presença de algumas arestas devem ser respeitadas. Por exemplo, na triangulação de um ambiente natural onde é conhecida a existência de um rio, pode ser interesante que as arestas que formam as margens do rio estejam presentes na triangulação final.

Wettergreen et al. (2010) utilizam a CDT na construção de mapas para localização robótica móvel. O robô móvel está constantemente fazendo a leitura de novos pontos do terreno através do laser range finder, e deseja incorporar novas áreas ao mapa antigo. Nesse caso, as bordas da triangulação antiga podem ser utilizazdas como restrições nas triangulaçoes novas.

Outras aplicações são apresentadas em Baker (1989) e Ren et al. (2005).

Capítulo 4

Metodologia para Modelagem de Terrenos Naturais

Nesse capítulo é apresentada a metodologia de modelagem de terrenos naturais utilizada nesse trabalho. Essa modelagem é feita a partir de uma nuvem de pontos 3D. Neste trabalho, a nuvem de pontos foi obtida por um veículo que possui um *laser range finder* 2D e um sensor de posição e é apresentado na seção 5.1. O laser foi instalado na parte superior frontal, inclinado em relação à horizontal, de maneira que o seu plano de medição intercepte o solo em uma linha a frente do veículo. Quando o veículo se move, o laser mede diferentes linhas do solo transversais à sua trajetória, fornecendo uma nuvem de pontos que representa o perfil do terreno.

A aplicabilidade desse método não é restrita à forma de coleta de dados descrita acima. Contudo, caso essa metodologia seja aplicada a partir de outro método de coleta de dados, algumas adaptações podem ser necessárias.

A modelagem é feita em quatro etapas (Figura 4.1): tratamento inicial dos dados, transformação de coordenadas dos pontos do laser para um referencial inercial, filtragem da vegetação e triangulação dos pontos. Cada uma dessas quatro etapas é detalhada nas respectivas seções a seguir.

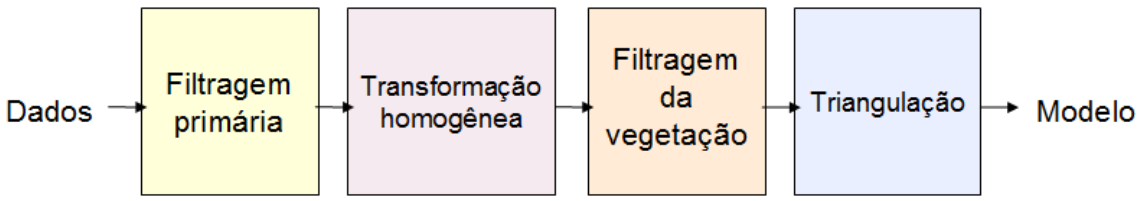


Figura 4.1: Diagrama da metodologia proposta para mapeamento de terrenos naturais.

O método de filtragem de vegetação utilizado foi sugerido em Vandapel et al. (2006) e, posteriormente, uma variação desse método é proposta em que é levada em conta a incerteza da classificação do filtro, possibilitando uma análise estatística dos pontos do modelo.

No final do capítulo, é apresentado um método otimizado de busca de pontos no modelo criado que utiliza os triângulos auxiliares gerados na triangulação, e são feitos comentários sobre o modelo criado.

4.1 Tratamento inicial dos dados

O primeiro passo consiste em filtrar as medições obtidas pela instrumentação embarcada: um *laser range finder* e um sensor de posição. No presente caso, foi utilizado um sensor Applanix que, devido à sua acurácia na medida de pose, induz pouco erro de modelagem e por isso dispensa o uso de filtros (Freitas, 2014).

O principal ruído do sensor laser é conhecido em inglês como *salt and pepper* (Lingemann et al., 2005), e consiste em medidas que não acompanham a geometria da superfície local (Sotoodeh, 2006). Ruídos desse tipo são causados por limites de oclusão, refletância de superfícies e múltiplos caminhos de reflexão. Para eliminar tais ruídos, foi adotada uma técnica de identificação e rejeição de *outliers* análoga à aplicada em Underwood et al. (2007). O termo *outlier* denota um ponto que, através de algum parâmetro, destoa do seu conjunto de pontos.

Um *laser range finder* realiza varreduras num ângulo de visão $\Delta(\psi_L)$, fornecendo a cada medição um *array* contendo distâncias a_j (Figura 1.2). O número de elementos do *array* ($size(a_j)$) é dado pela resolução s_l do laser

$$\begin{aligned} size(a_j) &= \frac{\Delta(\psi_L)}{s_l} \quad , \\ \Delta(\psi_L) &= \psi_{L\max} - \psi_{L\min} \quad . \end{aligned} \tag{4.1}$$

O laser Sick LMS 291 tem campo de visão de $\Delta(\psi_L) = 180°$ e resolução $s_l = 1°$. A cada medição, o sensor fornece o *array* $a_{j,\ j=0,\ldots,180}$.

Afim de filtrar as medições fornecidas pelo *laser range finder*, cada *array* a_j é modelado como uma função contínua com média μ_r e desvio padrão σ_r. Um ponto é classificado como *outlier* caso a diferença entre o valor médio e ele seja maior maior que o desvio padrão vezes uma constante k

$$|a_j - \mu_r| > k\ \sigma_r \ \Rightarrow a_j \text{ representa um outlier.} \tag{4.2}$$

Nesse trabalho foi utilizado $k = 2$ (Freitas, 2014).

Os pontos identificados são substituídos pela média aritmética dos pontos vizinhos

$$a_j = \frac{a_{j-1} + a_{j+1}}{2} \quad . \tag{4.3}$$

Vale lembrar que esse método não elimina erros de dados que se encontram

próximos à média. Contudo, a finalidade esse procedimento é eliminar pontos que estão claramente fora do padrão do ambiente local.

A região do terreno a ser modelada com maior aplicação na automação de veículos é a estrada a ser percorrida. Como a medição é feita por um veículo percorrendo uma estrada de terra, essa é o foco da medição.

O erro de medição do *laser range finder* é proporcional à distância (Siegwart and Nourbakhsh, 2004). No caso da medição de um ambiente *outdoor*, a leitura do laser é mais precisa em pontos diretamente a frente do veículo. Nessa região, a densidade de pontos também é maior. Assim, só são utilizados pontos mais próximos à frente do veículo em cada leitura, o que gera uma sequência de pontos que engloba a estrada percorrida pelo veículo e trechos adjacentes a ela.

Dessa forma, ao invés de utilizar todos os pontos do *array* entre 0° e 180°, são utilizados para a modelagem os pontos entre 45° e 135°.

4.2 Transformações de coordenadas

O laser faz a leitura de pontos do terreno em frente ao veículo, fornecendo as distâncias e seus respectivos ângulos de medição no plano de visão do sensor. Contudo, os pontos devem ser registrados com respeito a um sistema de coordenadas inercial, tornando necessária transformações de coordenadas dos dados.

Para isso, são definidos três sistemas de coordenadas como pode ser visto na Figura 4.2: um sistema de coordenadas coincidente com o plano de medição do laser com origem O_L, um sistema de coordenadas do veículo cuja origem O_V está localizada na superfície abaixo do centro do eixo das rodas traseiras e um sistema de coordenadas inercial com origem O_I localizado no ponto de partida do veículo.

Ao escanear o terreno à frente do veículo, o laser fornece as distâncias a_j medidas no plano de visão do sensor.

Os pontos medidos são registrados em coordenadas polares, onde um ângulo $(\psi_{L\min} + j\ s_l)$ é associado a uma distância a_j de acordo com sua posição (j) no *array*. Esses pontos são medidos com relação ao sistema de coordenadas do laser E^L por

$$P_{lj}^L = \left[a_j \cos\left(\psi_{L\min} + j\ s_l\right), a_j \sin\left(\psi_{L\min} + j\ s_l\right), 0\right]^T \quad . \tag{4.4}$$

Em seguida, os pontos lidos pelo laser precisam ser levados ao sistema de coordenadas inercial. Para isso, duas transformações de coordenadas são necessárias. A primeira transformada leva os pontos do sistema de coordenadas do laser P_{lj}^L para o sistema de coordenadas do veículo E^R, e a segunda leva do sistema do veículo para o sistema de coordenadas inercial E^I.

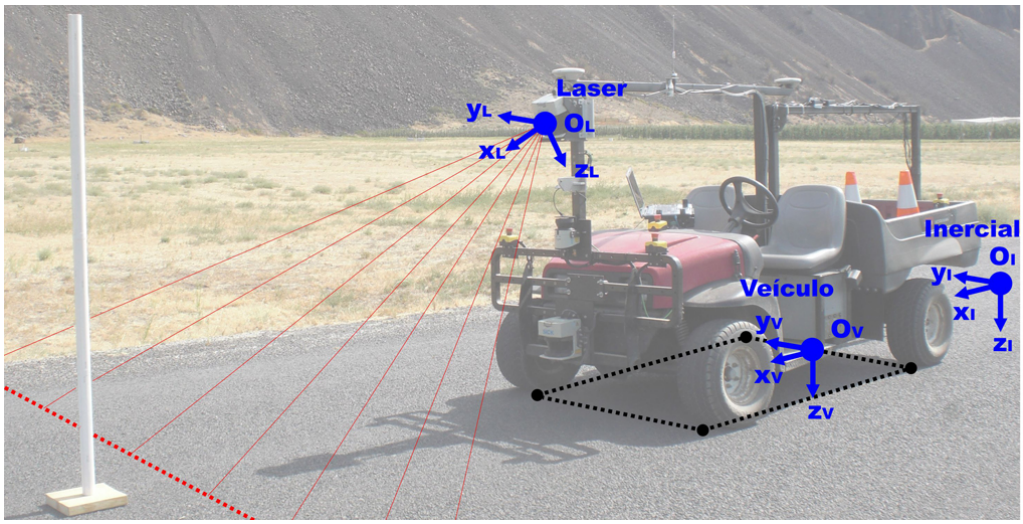


Figura 4.2: Veículo agrícola de pesquisa Laurel, com o laser instalado inclinado ao plano horizontal, medindo linhas do perfil do terreno 4 m a frente do veículo. Esta imagem ilustra também as origens dos sistemas de coordenadas do laser O^L, robô O^R e inercial O^I

A primeira transformação de coordenadas que deve ser realizada considera a pose do laser com relação ao centro de coordenadas do robô, denotada por $\mathbf{x}_L^R = [P_L^R, \boldsymbol{\varphi}_L^R]^T$. Ela foi obtida através do método de calibração apresentado em Underwood et al. (2007).

O procedimento de calibração realizado é descrito em Freitas (2014), e foi obtida a translação de $P_L^R = [2.61, 0.04, 1.05]^T$ m e rotação de $\boldsymbol{\varphi}_L = [0.05°, 21.25°, 0.56°]^T$. Os valores obtidos referem-se ao veículo apresentado na Figura 4.2.

A segunda transformação de coordenadas está relacionada à translação e rotação do veículo $\mathbf{x}_R^I$ com relação ao sistema de coordenadas inercial P_l^I. Esta transformação é obtida diretamente com os dados fornecidos pelo sensor de posição.

Os pontos medidos pelo laser são levados ao sistema inercial P_l^I através da equação

$$P_l^I = P_R^I + R_R^I \, P_l^R \quad . \tag{4.5}$$

4.3 Filtragem da vegetação

Essa dissertação propõe a realização da classificação dos dados obtidos pelo laser, separando pontos que representam a superfície suportadora de carga e a vegetação que cobre parte do terreno.

Muitas pesquisas implementaram essa tarefa utilizando sensor laser 3D, mas nesse trabalho objetiva-se realizar essa classificação utilizando apenas um *laser range finder* 2D, apontado para o solo com certa inclinação com a horizontal.

Essa configuração permite que o veículo faça a varredura do terreno com o laser,

ao mesmo tempo que pode detectar eventuais obstáculos que surjam à sua frente, utilizando apenas um único sensor a laser com escaneamento em duas dimensões.

Naturalmente, essa configuração impõe certas limitações, como a quantidade de dados disponíveis para a modelagem do terreno, a distância em que esses dados são obtidos e a velocidade de locomoção do veículo. Se o veículo estiver se locomovendo a alta velocidade, um sistema de detecção de obstáculos próximos pode não ser capaz de evitar a colisão com obstáculos que eventualmente estejam no trajeto do veículo.

Além disso, a velocidade está diretamente ligada à quantidade de dados obtidos em uma determinada região: quando mais lentamente o veículo se locomover, mais dados são obtidos da mesma região do terreno, aumentando a resolução do modelo gerado. Por esses fatores, essa configuração é indicada para a locomoção a baixas velocidades.

O objetivo da classificação implementada é filtrar arbustos e grama alta que podem ser discriminados pela varredura do laser, pois compõem volumes porosos. O método de filtragem utilizado é proposto em Vandapel et al. (2006). Este método classifica cada ponto como vegetação ou terreno analisando os pontos no interior de um cone com topo no ponto em questão. Caso exista algum ponto no interior desse cone, o ponto no topo é classificado como vegetação. Devido a erros característicos do laser, existe uma distância mínima entre o topo e um ponto no interior do cone para que o ponto no topo seja considerado vegetação (Figura 4.3). Os parâmetros deste algoritmo de classificação são o ângulo θ do cone e a distância mínima ρ entre o ponto sendo classificado e os pontos dentro do cone.

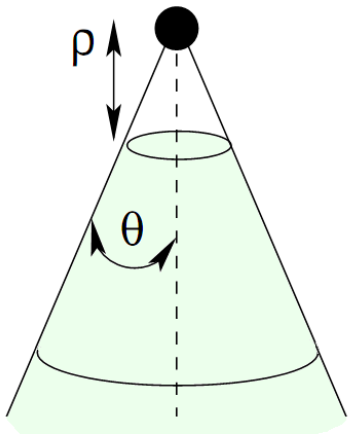


Figura 4.3: Parâmetros do algoritmo de classificação de vegetação.

Seja o ponto C sendo classificado, e suas coordenadas (x_c, y_c, z_c). No passo n, é verificado se o ponto N se encontra na região que fará o ponto C ser classificado como vegetação.

Primeiramente é verificado se

$$z_c - z_n > \rho \quad . \tag{4.6}$$

Caso a equação acima se verifique, é calculado o ângulo α que a semi-reta que liga C a N faz com a reta vertical que passa por C. Em seguida, é verificado se

$$\alpha < \theta \quad . \tag{4.7}$$

Caso essa equação também se verifique, o ponto C é classificado como vegetação e a iteração para, ou seja, não é nessessário verificar se mais nenhum ponto está no interior do cone abaixo de C. Caso a equação não se verifique, o processo é repetido para o ponto $N + 1$ e assim sucessivamente.

Os valores de ρ e θ dependem da aplicação. Como nesse trabalho deseja-se identificar grama alta foi utilizado $\rho = 0.2$, para que folhas de grama com comprimento maior do que 0.2 metros possam ser identificadas. O valor de θ depende não só da vegetação a ser filtrada, mas também da distância média entre as medições. Uma nuvem de pontos mais povoada de pontos permite a redução desse valor sem perdas consideráveis na qualidade da filtragem. É indicado que esses parâmetros sejam ajustados para cada conjunto de dados. No capítulo 5 é apresentada a modelagem completa do terreno com diferentes valores de ρ e θ para efeito de comparação.

Esse método de classificação não faz uma análise estatística dos pontos na sua classificação. Contudo, associar o grau de incerteza na classificação de um ponto pode ter utilidade na tomada de decisão de um robô móvel. Por isso, uma variação método é proposta a seguir.

É intuitivo que, quanto mais pontos existam no interior do cone de um ponto sendo classificado, mais certeza isso trará a afirmação de que ele pertence à vegetação. Contudo, para realizar essa análise, não se pode interromper a tarefa de classificação no primeiro resultado positivo do teste; este deve ser realizado analisando-se todos os pontos candidatos a estarem dentro do cone.

Para que todos os pontos da nuvem de pontos não precisem ser analisados na classificação de cada ponto, é natural que se restrinja essa análise a pontos próximos do ponto sendo classificado. Essa restrição pode ser feita por comparação das coordenadas x e y dos dois pontos, verificando se

$$x_1 - x_2 < d \quad . \tag{4.8}$$

Apenas no caso em que a equação acima se verifique, uma segunda análise é realizada:

$$y_1 - y_2 < d \quad . \tag{4.9}$$

Esse procedimento pode reduzir ou aumentar o trabalho computacional da filtragem, visto que essa análise está inclusa no teste do cone. Se o ponto está próximo o suficiente do ponto de referencia, utilizar esse método aumenta o trabalho computacional, pois o ponto passará por dois testes. Caso contrário, ele diminui a quantidade de processamento nessa iteração.

Na prática, existirá uma quantidade muito maior de pontos distantes do que está sendo classificado do que de pontos próximos. Logo, a tendência é que essa restrição reduza o custo computacional do método.

Também é possível quantificar o quão próximo das bordas do cone de referência um ponto está. Quanto mais próximo das bordas, maior a incerteza associada à classificação do ponto sendo analisado. Por exemplo: Se um ponto foi classificado como solo, significa que nenhum outro ponto está no interior de seu cone. Contudo, se existirem muitos pontos que quase infringiram essa regra, é intuitivo que esse ponto seja classificado como solo, mas com certo grau de incerteza.

Caso um ponto seja classificado como vegetação, deve se analisar não só a proximidade dos pontos que infringiram a regra do cone, mas também a quantidade de pontos no interior do cone. Quanto mais pontos no interior do cone, mais certeza é associada à classificação desse ponto como vegetação. Por outro lado, se os pontos no interior do cone estão próximos à borda, mais incerteza deve ser dada à classificação de vegetação.

Essa simples regra intuitiva permite associar um grau de incerteza à classificação de cada ponto. Vale lembrar que as coordenadas de cada ponto já estão associadas a um grau de incerteza que é proporcional à distância da medição. Para obter maior precisão, essa incerteza deve ser associada a um peso adequado.

Quanto maior for a precisão na determinação da incerteza, maior é o processamento computacional exigido. A determinação de tal precisão depende das condições de operação. No caso de uma tarefa *online*, deve-se limitar o processamento de acordo com a capacidade computacional do robô, permitindo a modelagem em tempo real. Entretanto, para realizar uma modelagem *offline* do terreno, é possível aumentar a precisão da incerteza, o que demanda mais processamento.

4.4 Triangulação da nuvem de pontos

O conjunto de pontos filtrados é usado como entrada no processo de triangulação. Para isso são consideradas apenas as coordenadas x e y dos pontos, realizando uma triangulação 2D. Aqui é utilizada triangulação de Delaunay, através do algoritmo incremental aleatório. O pseudocódigo 4.4.1 descreve a implementação do algoritmo

utilizando o software *Matlab*.

Algorithm 4.4.1: TRIANGULAÇÃO DE DELAUNAY($Pontos$)

---

procedure LEGALIZARARESTA(P_r, P_i, P_j, T)

if $aresta(P_i, P_j)$ é ilegal

$$\textbf{then} \begin{cases} Find(P_k) \\ Replace(aresta(P_i, P_j), aresta(P_r, P_k)) \\ LegalizarAresta(P_r, P_i, P_k, T) \\ LegalizarAresta(P_r, P_k, P_j, T) \end{cases}$$

main

$CreateBigTriangle(Pontos)$

$RandomSort(Pontos)$

for $r \leftarrow 1$ **to** n

$$\textbf{do} \begin{cases} T(P_i, P_j, P_l) \leftarrow FindTriangle(P_r) \\ Insert(P_r, T) \\ LegalizarAresta(P_r, P_i, P_j, T) \\ LegalizarAresta(P_r, P_i, P_l, T) \\ LegalizarAresta(P_r, P_j, P_l, T) \end{cases}$$

$DeleteBigTriangle$

return ($MalhaDeTriangulos$)

Vale comentar algumas funções presentes no código:

- $LegalizarAresta(P_r, P_i, P_j, T)$: O ponto sendo inserido é P_r, e a $aresta(P_i,P_j)$ pertence ao triângulo $T(P_r, P_i, P_j)$ que pode precisar da realização de um *flip* para tornar-se uma *aresta legal* (Figura 3.5). A $aresta(P_i,P_j)$ é considerada ilegal caso o círculo circunscrito ao triângulo $T(P_r, P_i, P_j)$ contenha algum outro ponto;

- $Find(P_k)$: encontra P_k (caso exista) que forma o triângulo $T(P_i,P_j,P_k)$, que é adjacente ao triângulo $T(P_r,P_i,P_j)$ e compartilha com esse a $aresta(P_i,P_j)$;

- $Replace(aresta(P_i, P_j), aresta(P_r, P_k))$: Substitui a primeira aresta pela segunda (*flip*);

- $CreateBigTriangle(Pontos)$: Cria um triângulo auxiliar que envolve todos os pontos;

- $FindTriangle(P_r)$: Encontra o triângulo $T(P_i, P_j, P_l)$ que contém P_r;

- $Insert(P_r, T)$: Insere P_r em T, criando 3 triângulos ao conectar P_r a P_i, P_j e P_l;
- $DeleteBigTriangle$: Apaga o triângulo auxiliar, seus 3 pontos e as arestas ligadas a eles.

Legalização

Nesta seção é exemplificado o processo de legalização de uma aresta. Considere a triangulação sendo realizada na Figura 4.4. O ponto R está sendo inserido na malha de triângulos existente, substituindo o triângulo QIJ pelos triângulos QIR, RIJ e QRJ.

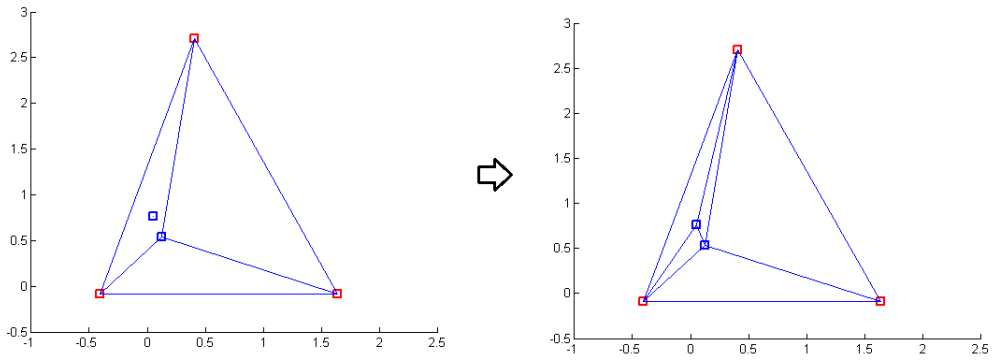


Figura 4.4: Processo de triangulação: o ponto R está sendo inserido na malha de triângulos, substituindo o triângulo QIJ pelos triângulos QIR, RIJ e QRJ.

É necessário verificar se os 3 novos triângulos são legais. Considere o quadrilátero convexo $RIKJ$, que contém o triângulo RIJ. Como visto na seção 3.2.2, um quadrilátero convexo pode ser triangulado de duas formas, mas apenas uma será uma triangulação de Delaunay (exceto no caso em que seja possível traçar um círculo que passe pelos 4 vértices do quadrilátero).

Por isso, deve ser verificado se o triângulo RIJ é legal. Isso é feito traçando o círculo que passa pelos vértices de RIJ. Caso o ponto K esteja no interior desse círculo, a triangulação não é legal, e a aresta IJ deve ser substituída pela aresta RK.

No algoritmo implementado, é verificado se o ponto K está no interior do círculo circunscrito ao triângulo RIJ através da seguinte equação matricial:

$$\begin{bmatrix} x_R & y_R & x_R^2+y_R^2 & 1 \\ x_I & y_I & x_I^2+y_I^2 & 1 \\ x_J & y_J & x_J^2+y_J^2 & 1 \\ x_K & y_K & x_K^2+y_K^2 & 1 \end{bmatrix} = \begin{bmatrix} x_R-x_K & y_R-y_K & (x_R^2-x_K^2)+(y_R^2-y_K^2) \\ x_I-x_K & y_I-y_K & (x_I^2-x_K^2)+(y_I^2-y_K^2) \\ x_J-x_K & y_J-y_K & (x_J^2-x_K^2)+(y_J^2-y_K^2) \end{bmatrix} > 0$$

Caso a equação acima se verifique, a aresta IJ deve ser substituída pela aresta RK (Figura 4.5).

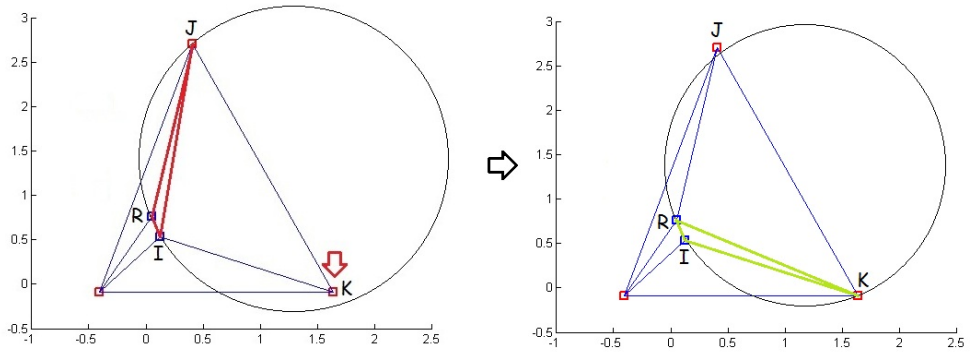


Figura 4.5: O círculo circunscrito ao triângulo RIJ contém o ponto K, por isso aponta uma triangulação ilegal. Já o círculo circunscrito ao triângulo RIK não contém nenhum outro ponto em seu interior, denotando um triângulo e arestas legais. Assim, o flip da aresta IJ gerando a aresta RK é um processo de legalização.

Na equação matricial, os vértices do triângulo R, I e J devem estar ordenados em sentido anti-horário. Para fazer essa verificação, basta realizar o produto vetorial de dois vetores

$$v_{12} = \begin{bmatrix} x_R - x_I \\ y_R - y_I \\ 0 \end{bmatrix} \quad , \tag{4.10}$$

$$v_{13} = \begin{bmatrix} x_R - x_J \\ y_R - y_J \\ 0 \end{bmatrix} \quad , \tag{4.11}$$

$$v_4 = v_{12} \times v_{13} \quad . \tag{4.12}$$

Se a coordenada z de v_4 for maior que 0, os pontos J, I e R estão organizados no sentido anti-horário.

Concluída a triangulação no plano, é agregada novamente a componente z de cada ponto, preservando as conexões entre eles (Figura 3.8). Dessa forma, é gerada uma superfície 2 $\frac{1}{2}$ D.

4.5 Otimização da busca de pontos no modelo

O modelo gerado pode ser utilizado para que a altura em um ponto não medido seja estimada. Para isso, é necessário a realização de uma busca no modelo, para encontrar o triângulo que contém o ponto.

Durante o processo, os triângulos são armazenados em uma estrutura de dados do tipo árvore. Quando um triângulo dá origem a novos triângulos, o triângulo original não é apagado da memória, mas é indicado que esse não é mais um triângulo final (folha), e é informada a localização dos triângulos novos originados por ele. Essa árvore é útil mesmo após a conclusão da triangulação.

Por exemplo, caso se deseje saber a aproximação da altura de um ponto não amostrado a partir da triangulação, deve-se localizar o triângulo final que contém o ponto. Uma forma eficiente de busca é partir dos triângulos iniciais e seguir nos ramos correspondentes na árvore de triângulos, até que se chegue a um ramo final da árvore onde é obtido o triângulo final que contém o ponto.

4.6 Comentários sobre o modelo gerado

O modelo do terreno obtido corresponde a uma superfície 2 $\frac{1}{2}$ D. O método de triangulação utilizado cria necessariamente uma superfície cuja projeção no plano horizontal forma um polígono convexo. Essa característica pode gerar uma aproximação grosseira em regiões distantes dos pontos medidos. Caso esse efeito não seja desejado, a nuvem de pontos pode ser separada em regiões com pontos de coordenadas (x,y) próximas. Esse procedimento tende a reduzir o custo computacional da triangulação quanto maior for a quantidade de pontos a ser triangulada.

Para que não se forme uma camada não modelada entre as regiões trianguladas, é sugerido que existam pequenas regiões de intercessão entre as duas regiões trianguladas. Esse procedimento foi testado no modelo criado a partir dos pontos medidos, e foi observado que a altura de um ponto na intercessão era a mesma nas duas regiões. Dessa forma, a busca de um ponto na região de intercessão pôde ser feita em qualquer uma das duas regiões sem que seja observada perda de qualidade na aproximação.

Capítulo 5

Validação Experimental

Para demonstrar a aplicabilidade do método proposto nessa dissertação, a técnica foi verificada com dados coletados em terreno natural - uma estrada de terra. Para a coleta de dados foi utilizado o veículo agrícola Laurel. Como descrito no capítulo 4, foram utilizados sensor *laser range finder* e sensor de posição de alta precisão.

Experimentos com o veículo foram realizados para o desenvolvimento de um sistema de detecção de obstáculos (Freitas et al., 2012), e dados foram coletados em terrenos naurais. Parte dos dados são utilizados aqui para ilustrar a modelagem de terrenos. A coleta de dados foi realizada no pomar *Soegel*, no estado da Pensilvânia - EUA, em 24/08/2011.

Os resultados da modelagem do terreno a partir dos dados dos experimentos são apresentados na seção 5.2.

5.1 Plataforma para a coleta de dados

Laurel é um veículo de pesquisa da família APM da Carnegie Mellon University (CMU) utilizado no desenvolvimento de métodos e técnicas para a realização de operações autônomas, a serem implementadas nos demais robôs da família APM. Ele é baseado no veículo elétrico Toro MDE eWorkman e pode operar tanto manualmente como autonomamente, e é utilizado no desenvolvimento de métodos e técnicas para a realização de operações autônomas. O veículo é apresentado na Figura 5.1.

A plataforma é equipada com *laser range finder* modelo Sick LMS 291 (Figura 5.2. O sensor possui um ângulo de visão de 180° com resolução de 1° e alcance máximo de leitura de 80 m. A resolução para medir distâncias entre 1 e 20 metros é de 10 mm, com 35 mm de precisão e 10 mm de desvio padrão.

Laurel se locomove numa velocidade máxima de 2 m/s, e leva aproximadamente 2 s para parar. Dessa maneira, o laser foi posicionado de forma a fazer a leitura do terreno a aproximadamente 4 metros do veículo. O sensor foi instalado na posição mais alta do veículo, a 1.45 m de altura do solo, buscando diminuir variações no

Figura 5.1: Veículo de pesquisa agrícola utilizado na coleta de dados de campo.

Figura 5.2: Sensor *laser range finder* modelo SICK LMS 291

plano de medição do laser causadas por irregularidades do terreno (Figura 4.2). Assim, a inclinação do laser com respeito ao plano horizontal é de aproximadamente $\theta_L = 20°$.

O veículo também é equipado com um sensor de posicionamento de alta precisão Applanix POS 220 LV INS/GPS, que fornece medições com precisão de 3 cm para posição e 0.05° para orientação.

5.2 Descrição dos experimentos e resultados

O veículo percorreu aproximadamente 100 metros de uma estrada de terra coberta por grama em grande parte de sua superfície (Figura 5.3), localizada em uma região agrícola. A velocidade média do veículo foi de aproximadamente 1 m/s.

Os dados foram obtidos em dois experimentos de campo. No primeiro, o veículo partiu do ponto mais baixo da estrada em direção ao mais alto, enquanto no segundo

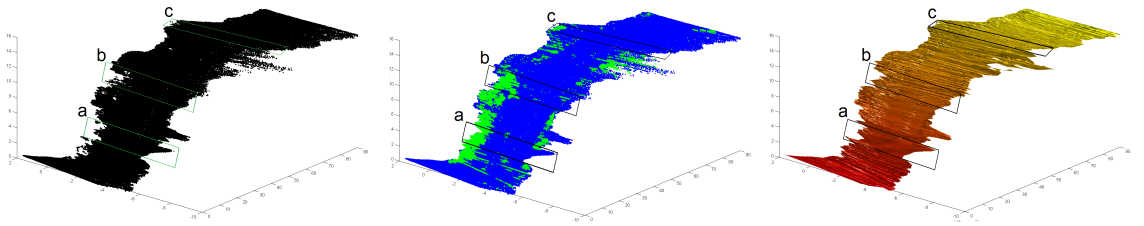
a estrada foi percorrida no sentido descendente. O modelo do terreno foi gerado a partir dos dados do primeiro experimento, enquanto os dados coletados no segundo experimento são utilizados para verificar a qualidade do modelo gerado.

Figura 5.3: Imagem da estrada de terra percorrida, obtida através da interpolação de imagens geradas por uma câmera embarcada.

Um operador conduziu o veículo ao longo da estrada, possibilitando que o *laser range finder* fizesse a varredura do terreno percorrido. Após os experimentos, os dados do laser e do sensor de posição foram sincronizados, de forma a fornecer um par de letura do laser e do sensor de posição em um mesmo *timestep*.

Timestep denota uma marcação temporal que representa o passo de uma simulação, ou otimização ou qualquer algoritmo de caráter recursivo.

Uma visão geral do processo é ilustrada na Figura 5.4. Na imagem (1) encontra-se a nuvem de pontos após a filtragem de *outliers* e transformação para referencial inercial. Na imagem (2) os pontos foram classificados como vegetação ou terreno (verde e azul respectivamente), e na imagem (3) é possível ver o resultado final da modelagem do terreno. Os pontos classificados como vegetação foram removidos do modelo final.



(1) (2) (3)

Figura 5.4: Etapas do processo de modelagem de terreno: (1) Nuvem de pontos 3D; (2) Classificação dos pontos como vegetação ou terreno; (3) Superfície criada pela triangulação dos pontos.

5.2.1 Identificação da vegetação

Devido a dificuldade da avaliação do desempenho do filtro de vegetação, não foi realizada uma análise quantitativa do erro da filtragem. Contudo, é possível realizar uma análise qualitativa através da comparação visual do modelo com imagens do terreno real.

Na Figura 5.5 são exibidas imagens obtidas por câmera de três trechos do terreno, e podem ser comparadas com as seções (a), (b) e (c) destacadas na Figura 5.4. O

filtro de vegetação consegue identificar claramente as regiões do terreno cobertas por grama alta, conforme ilustrado na Figura 5.6. A mesma figura apresenta também declividades e buracos do modelo que estão presentes no terreno.

Figura 5.5: Imagens do terreno percorrido que podem ser comparadas com o modelo criado nas Figuras 5.4 e 5.6.

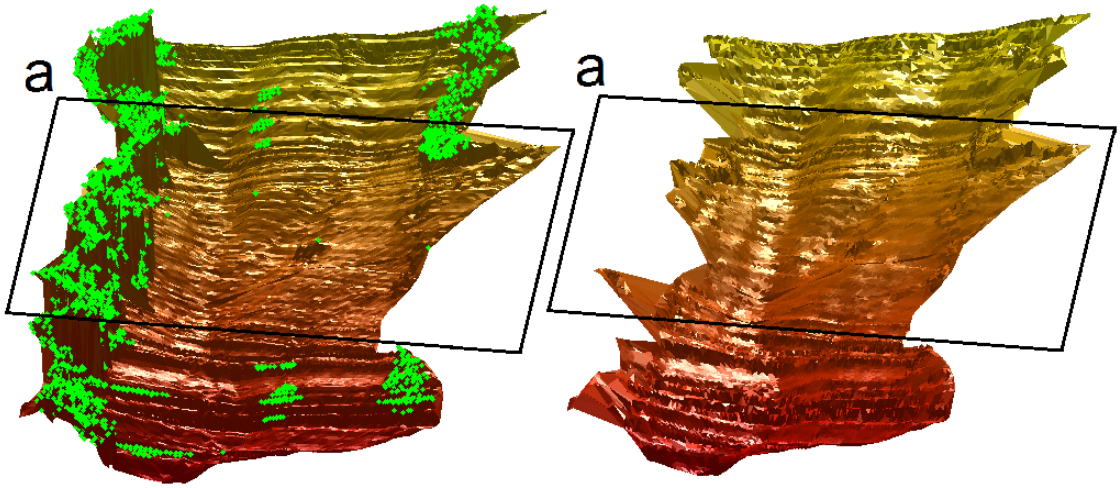


Figura 5.6: Trecho do terreno modelado que contém a região ilustrada na Figura 5.5(a). À esquerda o modelo com vegetação e à direita o modelo após a filtragem da vegetação.

Em uma análise qualitativa das imagens, é possível inferir que a vegetação foi corretamente identificada e removida do modelo da superfície suportadora de carga.

Variação de parâmetros do filtro de vegetação

Com base nos dados experimentais, várias regiões com grama que não foram detectadas pelo filtro de vegetação. Contudo, isso já era esperado, visto que o filtro só possui a capacidade de identificar vegetação alta.

Outra característica da filtragem que pode ser observada é a presença de falsos positivos (pontos classificados como vegetação que não pertencem à vegetação). A presença de pontos verdes formando linhas transversais à estrada é devida a classificação errônea. Esses pontos tem origem na sobreposição de linhas de pontos na nuvem de pontos e, provavelmente, se devem a erros de sincronização dos dados do laser com o sensor de posição. De qualquer forma, o algoritmo retira esses pontos do modelo, o que pode ser benéfico para a verossemelhança do mesmo.

Os parâmetros do filtro de vegetação influem diretamente na presença de falsos positivos no modelo. Os valores dos parâmetros utilizados na modelagem são de $\theta = 60°$ e $\rho = 20$ cm. Os dados coletados no primeiro experimento foram utilizados para testar diferentes valores desses parâmetros, como pode ser observado nas Figuras 5.7 e 5.8.

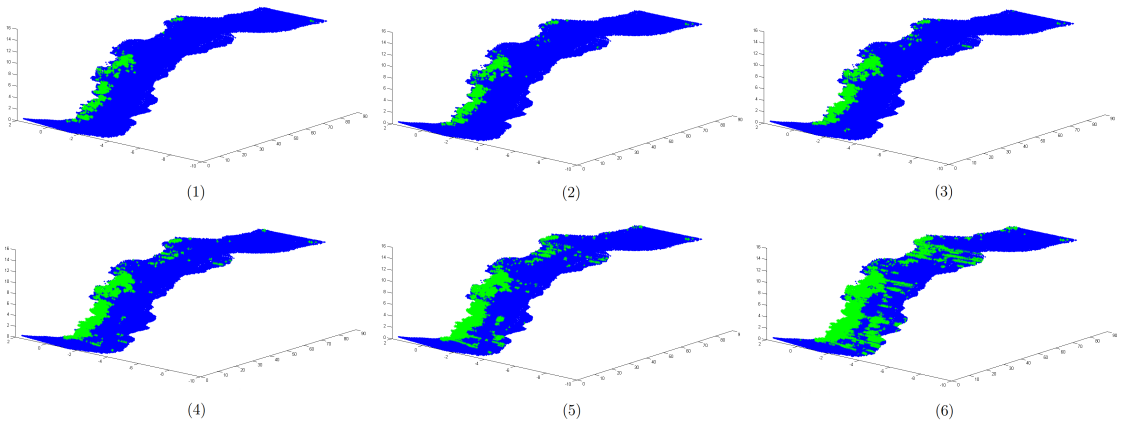


Figura 5.7: Classificação da vegetação com $\rho = 20$ cm e diferentes valores do ângulo θ: (1) $\theta = 30°$; (2) $\theta = 40°$; (3) $\theta = 50°$; (4) $\theta = 60°$; (5) $\theta = 65°$; (6) $\theta = 70°$.

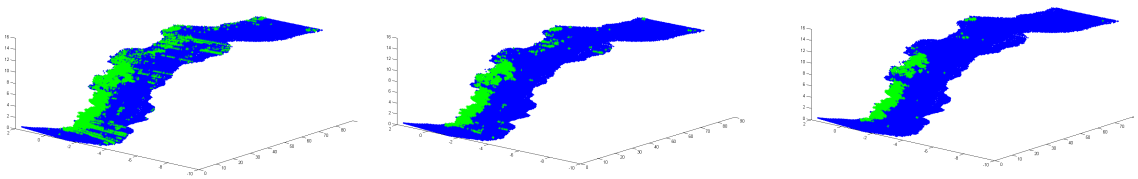


Figura 5.8: Classificação da vegetação com $\theta = 60°$ e diferentes valores da distância mínima ρ: (1) $\rho = 10$ cm; (2) $\rho = 20$ cm; (3) $\rho = 30$ cm.

5.2.2 Superfície obtida

Os pontos medidos pelo laser passaram por filtragem de *outliers* e transformação para referencial inercial através da leitura do sensor de posição. A nuvem de pontos resultante passou pelo processo de filtragem de vegetação, onde cada ponto foi

classificado como parte da superfície do terreno ou como vegetação. Por fim, foi feita a triangulação de Delaunay plana dos pontos através do algoritmo incremental aleatório e os pontos foram levados à sua altura original formando uma superfície 2 $\frac{1}{2}$ D (Figura 5.4).

Análise do erro do modelo

A modelagem de terrenos a partir da triangulação de medições gera uma estimativa da elevação do terreno em pontos que se encontram entre dados amostrados. Afim de analisar o erro dessa aproximação, foi realizada uma comparação entre o modelo gerado a partir de dados do primeiro experimento com pontos coletados no segundo experimento.

Para tal, foi criado um algoritmo para a seleção de um subconjunto de dados do segundo experimento. Esse algoritmo tem a finalidade de selecionar pontos aleatoriamente, de forma a obter pontos com alguma uniformidade no espaçamento. O algoritmo busca obter pontos em sequência, onde o a distância d_k entre o ponto p_k e o ponto p_{k+1} deve ficar entre um valor mínimo e um valor máximo

$$d_{min} < d_k < d_{max} \quad . \tag{5.1}$$

O algoritmo busca um ponto aleatoriamente entre os 3000 pontos à frente do ponto p_k no *array* de pontos. Caso o novo ponto obedeça à restrição da equação acima, esse é tomado como p_{k+1} e a iteração é repetida para o passo seguinte. Caso a restrição não seja satisfeita, o ponto é descartado e um novo ponto é escolhido aleatoriamente.

A densidade de pontos é muito maior em determinadas regiões, e uma seleção puramente aleatória tenderia a obter mais pontos dessas regiões do que de regiões com menos dados. Dessa forma, a utilização do algoritmo favorece a seleção de pontos com espaçamento mais regular.

Foram selecionados 150 pontos coletados no segundo experimento. A superfície cinza mostrada na Figura 5.9 foi gerada pelo procedimento de modelagem a partir de pontos obtidos no primeiro experimento, e os pontos em azul foram coletados no segundo experimento.

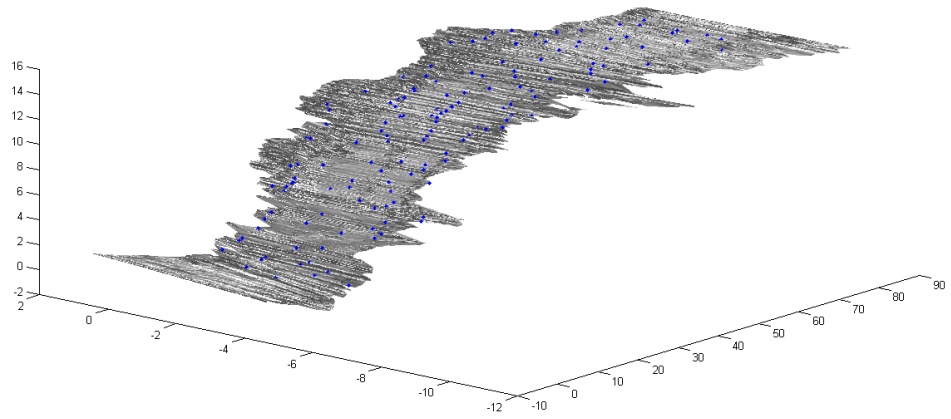


Figura 5.9: A superfície cinza foi gerada pelo procedimento de modelagem a partir de pontos obtidos no primeiro experimento, e os pontos em azul foram coletados no segundo experimento.

O erro do modelo foi estimado através da diferença entre a coordenada z^k_{exp2} dos pontos do segundo experimento e a coordenada z^k_{modelo} da projeção das coordenadas x^k_2 e y^k_2 na superfície modelada no primeiro experimento

$$\mu = \frac{\sum_{k=1}^{n} \left| z^k_{modelo} - z^k_{exp2} \right|}{n} \quad . \tag{5.2}$$

A amostragem de pontos obtidos no segundo experimento gerou uma média do módulo do erro μ de 7.3 centímetros, e desvio padrão de 9.8 centímetros.

Módulo do Erro	Valor (metros)
μ	0.0727
max	0.7161
min	0.0032
σ^2	0.0096
σ	0.0982

Foi observado que os 6 pontos com maior erro se encontravam nas extremidades do modelo.

Os pontos do primeiro experimento utilizados para a geração do modelo passaram pelo filtro de vegetação, assim como os pontos do segundo experimento. O filtro atua eliminando alguns pontos pertencentes à vegetação, mas não todos.

O modelo real do terreno medido não é facilmente obtido. Dessa forma, os erros relativos à modelagem são estimados através de comparacão entre modelo gerado a partir de um experimento e medições de outro experimento, ambos realizados no mesmo ambiente pelo mesmo robô com os mesmos sensores.

O erro calculado não leva em consideração as incertezas de medição dos sensores, nem a sincronização entre *laser range finder* e sensor de posição. Esses erros estão presentes tanto nos pontos do segundo experimento quanto nos pontos do primeiro experimento que geraram o modelo.

Dessa forma, não é possível inferir com precisão quais erros se devem à medição e quais erros se devem ao processo de modelagem através de triangulação.

O limite aceitável do erro de um modelo depende da sua aplicação. Em aplicações referentes a robôs móveis, as dimensões do veículo devem ser consideradas.

Nos experimentos apresentados neste capítulo, foi utilizado o veículo Laurel, que possui 215 cm de comprimento e 150 cm de largura. O diâmetro das rodas do veículo utilizado é de aproximadamente 56 cm (22").

Assumindo certa complacência entre roda e terreno, podemos considerar que esses erros não invalidam o modelo obtido para aplicações com o veículo utilizado no experimento.

Capítulo 6

Aplicações na Robótica de Campo

Para realizar tarefas com segurança em ambientes naturais, um veículo autônomo deve ser capaz de reconhecer quais interações podem apresentar risco à sua integridade. Esse é um problema de difícil solução, pois existem interações complexas entre o robô e o terreno, que pode ser desconhecido, e que possivelmente muda ao longo do tempo.

Em aplicações na agricultura, é possível modelar grande parte do ambiente, mas mudanças podem ocorrer devido a condições meteorológicas ou a objetos deixados no terreno. Além disso, é comum que o veículo necessite atravessar regiões com vegetação que muda ao longo do ano. Dessa forma, a modelagem do terreno é tarefa central no controle de robôs móveis que atuam em ambientes naturais.

Nesta seção, são apresentados exemplos de aplicações da modelagem de terrenos com filtragem de vegetação em aplicações relacionadas à robótica de campo: um método simples de análise de estabilidade e um método de ajuste de trajetória, ambos aplicados à automação agrícola. Essas técnicas utilizam o modelo do terreno gerado com a metodologia apresentada nas seções anteriores.

6.1 Análise de estabilidade

A estimação de estabilidade é um tema importante no controle de veículos que operam em superfícies irregulares. A capacidade de se prever a interação entre veículo e terreno em uma determinada região significa a possibilidade de planejar trajetórias que ofereçam menos risco de tombamento ao robô.

A estabilidade é uma propriedade do estado de equilíbrio ou ponto de equilíbrio. Um sistema pode ter vários pontos de equilíbrio estáveis e instáveis. O equilíbrio é dito estável se, após o sistema ser submetido a um distúrbio limitado, ele permanece próximo do estado original de equilíbrio. Quanto maior a magnitude do distúrbio necessário para retirar o sistema do estado original, maior é a região de estabilidade daquele equilíbrio. Caso contrário, se não for possível determinar uma região de

estabilidade para o equilíbrio, por menor que seja a perturbação, este equilíbrio é dito instável. Em outras palavras, qualquer que seja a perturbação, mesmo infinitesimal, irá afastar o sistema do seu estado original instável. Esta é a noção de estabilidade introduzida por Lyapunov.

Os robôs móveis são exemplos de sistemas mecânicos onde podem ser identificados vários pontos de equilíbrio estáveis e instáveis alternados. O estado de equilíbrio estável de interesse é caracterizado pela posição em que todas as suas rodas tocam o terreno. Uma perturbação suficientemente grande no robô pode causar o seu tombamento, isto é, o sistema pode ir para outro estado de equilíbrio estável onde não as rodas, mas o seu chassi está em contato com o terreno. Uma perturbação ainda maior poderia fazer o robô tombar várias vezes e voltar a ficar com todas as rodas em contato com o terreno novamente. No entanto, neste estudo, nenhum tipo de tombamento é aceitável. Os estados de equilíbrio instáveis do robô móvel estão localizados na fronteira entre duas regiões de estabilidade adjacentes.

A análise de estabilidade realizada nesse trabalho objetiva fazer uma estimativa da configuração do robô em dada posição do terreno, e inferir sua inclinação, para que possam ser evitadas configurações não desejadas, como o tombamento.

De acordo com Sreenivasan (1994), o terreno provoca alterações na altura e orientação do robô. Assim, a seção 6.1.1 apresenta dois métodos simples de cálculo das coordenadas z dos pontos de contato das rodas do veículo com o terreno a partir das coordenadas (x,y) das rodas do veículo, e do modelo da superfície suportadora de carga. A partir dessa análise, é possível calcular a inclinação do veículo nessa posição, realizando uma análise estática da pose do robô.

Em alguns casos, o cálculo da direção do contato entre a roda e o solo também é útil, pois a partir dele é possível calcular a força resultante nesse ponto. A seção 6.1.2 apresenta dois métodos simples de cálculo da direção da força normal entre a roda do veículo e o terreno.

6.1.1 Cálculo dos pontos de contato entre as rodas do veículo e o terreno

Uma solução simples para verificar a viabilidade de dada trajetória é a discretização do percurso, e em seguida a análise da configuração do veículo em cada um dos pontos gerados na discretização. Supondo que a trajetória seja dada em função da origem do sistema de coordenadas do robô (nesse trabalho, foi usado como referência o centro do eixo das rodas traseiras do veículo), com o conhecimento da geometria e das dimensões do seu chassi, é possível aproximar as coordenadas (x,y) das rodas a partir das coordenadas (x,y) do ponto de referência do robô.

Utilizando um modelo da superfície suportadora de carga, é possível estimar a

coordenada z das rodas do veículo em cada ponto, obtendo assim as coordenadas (x,y,z) dos 4 possíveis pontos de contato do veículo com o solo em determinado ponto.

A análise estática dos pontos de contato do veículo com o solo é simples, mas fornece informações úteis, como o *roll* e o *pitch* que o terreno gera no robô a partir do seu *yaw* em determinada posição.

Nesta seção são apresentados dois métodos simples para a definição dos pontos de contato entre as rodas e o terreno. O primeiro utiliza apenas a nuvem de pontos e o segundo utiliza a malha triangulada.

Em ambos os métodos, deve-se levar em consideração as dimensões do veículo e das rodas (raio e largura), a superfície de contato esperada dos pneus com o solo, a posição e orientação do veículo em cada ponto da trajetória (coordenadas (x, y) e *yaw*) e a posição relativa ao veículo do ponto de referência (neste trabalho, foi utilizado o ponto central do eixo das rodas traseiras do veículo).

As coordenadas (x_{ri}, y_{ri}) da roda i podem ser obtidas a partir das coordenadas (x_c, y_c) do ponto de referência do robô e da translação $[x_{ri}, y_{ri}]^T$ da roda i com relação à referência do robô

$$\begin{bmatrix} x_{ri} \\ y_{ri} \end{bmatrix} = \begin{bmatrix} x_c \\ y_c \end{bmatrix} + \begin{bmatrix} \Delta x_{ri} \\ \Delta y_{ri} \end{bmatrix} \quad . \tag{6.1}$$

Cálculo dos pontos de contato: método 1

No primeiro método, faz-se uso da área da superfície de contato do pneu com o solo para se definir círculo de raio R ao redor do centro do ponto de contato. São verificados quais pontos da nuvem de pontos que representam o terreno (após a filtragem da vegetação) que se encontram nessa área. O ponto P com coordenadas (x_P, y_P, z_P) à distância d_P do centro da roda i está próximo o suficiente do ponto de contato caso a seguinte equação se verifique

$$d_P = \sqrt{(x_P - x_{ri})^2 + (y_P - y_{ri})^2} < R \quad . \tag{6.2}$$

Após a obtenção de um vetor com os n pontos contidos nessa área, é calculada a média ponderada das coordenadas z dos pontos, onde cada ponto é ponderado de acordo com a sua distância d_P do centro do ponto de contato

$$z_{\mu i} = \sqrt{\frac{\sum_{p=1}^{n} d_P^{-1} \times z_P}{\sum_{p=1}^{n} d_P^{-1}}} \quad . \tag{6.3}$$

Caso se deseje usar o grau de incerteza associado a cada ponto, este também pode ser levado em consideração no cálculo dos pesos. A variância dessa média pode ser utilizada como o grau de incerteza do ponto de contato da roda do veículo com o

solo. Dessa forma, as coordenadas do ponto de contato da roda i são aproximados por $(x_{ri}, y_{ri}, z_{\mu i})$.

Um problema associado a esse método é a distribuição irregular dos pontos medidos. No experimento realizado, era comum que apenas um ou dois pontos estivéssem contidos no círculo de raio R. Quando nenhum ponto é encontrado nessa área, deve-se aumentar recursivamente o raio do círculo (por exemplo: $R_{i+1} = 1,5 \times R_i$), até que pelo menos um ponto seja encontrado. O aumento do raio de investigação reduz a qualidade da aproximação do ponto de contato, aumentando assim a incerteza do valor encontrado.

Cálculo dos pontos de contato: método 2

O cálculo dos pontos de contato também pode ser realizado utilizando a superfície triangulada, onde a coordenada z_P de um ponto P é dada a partir da projeção de suas coordenadas (x_P, y_P) na malha de triângulos. Nesse método, é verificado qual triângulo T pertencente à malha contém o ponto com coordenadas (x_P, y_P). Essa busca pode ser otimizada como foi proposto na seção 4.5.

Um método de cálculo da a coordenada z_P da projeção do ponto (x_P, y_P) no plano de um triângulo é apresentada a seguir.

Considere o triângulo T formado pelos pontos A, B e C e as matrizes M e N

$$M = \begin{bmatrix} x_A & y_A & 1 \\ x_B & y_B & 1 \\ x_C & y_C & 1 \end{bmatrix} \quad , \tag{6.4}$$

$$N = \begin{bmatrix} z_A \\ z_B \\ z_C \end{bmatrix} \quad . \tag{6.5}$$

Seja a matriz auxiliar $O = [o_1, o_2, o_3]^T$ definida por

$$\begin{bmatrix} o_1 \\ o_2 \\ o_3 \end{bmatrix} = M^{-1} * N \quad . \tag{6.6}$$

A coordenada z_{ri} do ponto de contato entre a roda i e o terreno pode ser aproximado por

$$z_{ri} = o_1 \times x_{ri} + o_2 \times y_{ri} + o_3 \quad . \tag{6.7}$$

A incerteza desse valor pode ser associada à incerteza do triângulo, à área do triângulo (quanto maior a área do triângulo, maior a incerteza) e à incerteza dos

pontos formadores do triângulo.

Desde que o ponto esteja dentro do terreno modelado, sempre haverá um triângulo que o contenha (o método de triangulação cria um polígono convexo). E, teoricamente, esse triângulo define o plano local que melhor representa o terreno nessa região, a partir dos dados disponíveis.

Cálculo dos ângulos *roll* e *pitch* do veículo

Após a obtenção das coordenadas (x_{ri}, y_{ri}, z_{ri}) do ponto de contato de cada roda, o *roll* e o *pitch* do veículo podem ser estimados. Nesse trabalho, por simplificação foi suposto que os quatro pontos de contato devem estar em um mesmo plano. Como a chance de que os quatro pontos calculados caiam exatamente no mesmo plano é pequena, é natural que se utilize um método de cálculo do plano do veículo.

O método utilizado nesse trabalho foi utilizado em Freitas (2014), e tem por objetivo fazer uma análise estática do pior caso para a estabilidade do sistema, ou seja, da situação em que a inclinação do robô indicada pelos ângulos de *roll* e *pitch* alcance valores extremos. Para isso, calcula-se a média das quatro coordenadas z e exclui-se o ponto com a coordenada z mais próximo da média. Em seguida, é calculado o plano que passa pelos três pontos, e apartir da normal desse plano são inferidos os ângulos de *pitch* e *roll* do veículo naquele ponto.

Uma abordagem alternativa para esse problema é levar em conta a incerteza de cada medição na definição do ponto a ser excluído.

6.1.2 Cálculo da direção da força normal no contato entre terreno e rodas do veículo

Algumas aplicações com robôs articulados objetivam a maximização da eficiência de tração, de forma a evitar o escorregamento das rodas e assim diminuir o consumo de energia do robô. A análise de tração de cada roda pode ser obtida através do cone de atrito, que depende da força normal aplicada nas rodas.

A análise de estabilidade pode ser feita utilizando uma métrica de estabilidade chamada transferência de carga, que considera a diferença entre forças normais aplicadas em lados opostos do robô. Em geral, é desejado que as forças normais aplicadas nos pontos de contato sejam uniforme.

Dessa forma, o cálculo da normal do contato entre roda e terreno é relevante e dois métodos são apresentados: o primeiro utiliza apenas a nuvem de pontos, e o segundo utiliza a malha de triângulos criada.

Cálculo da normal dos pontos de contato: método 1

No primeiro método, são verificados quais pontos da nuvem de pontos que se encontram na área da superfície de contato do pneu com o solo.

Assim como no método 1 de cálculo dos pontos de contato, o ponto P com coordenadas (x_P, y_P, z_P) fará parte do conjunto de pontos próximos o suficiente do ponto de contato caso a equação 6.2 se verifique. Em seguida, é definido um plano que represente adequadamente esse conjunto de pontos, através do método dos mínimos quadrados.

Seja a equação do plano

$$ax + by + c = z \quad . \tag{6.8}$$

A distância d_P^{plano} de um ponto P a esse plano é

$$d_P^{plano} = \frac{|ax_P + by_P - z_P + c|}{\sqrt{a^2 + b^2 + 1}} \quad . \tag{6.9}$$

Contudo, os valores de a, b e c que minimizam d_P^{plano} também minimizam a função

$$e_P = (ax_P + by_P - z_P + c)^2 \quad . \tag{6.10}$$

Afim de minimizar o erro quadrático médio entre o plano e o conjunto de pontos, basta minimizar a soma

$$S = \sum_{p=1}^{n} e_P \quad . \tag{6.11}$$

A derivada da soma S em relação a a, b e c geram 3 equações, que se igualadas a zero, formam o sistema

$$\frac{dS}{da} = 0, \quad \frac{dS}{db} = 0, \quad \frac{dS}{dc} = 0 \quad . \tag{6.12}$$

A solução desse sistema gera os valores de a, b e c que definem o plano que melhor descreve o conjunto de pontos em questão.

A normal do ponto de contato da roda do veículo com o solo é definida pela normal do plano encontrado, e a incerteza dessa normal pode ser associada à incerteza dos pontos formadores do plano e à média das distâncias dos pontos ao plano μ_d^{plano} calculada por

$$\mu_d^{plano} = \frac{\sum_{p=1}^{n} d_P^{plano}}{n} \quad . \tag{6.13}$$

A distribuição irregular dos pontos novamente se torna um problema para esse método. Com o agravante de que agora são necessários no mínimo três pontos para a definição de um plano. Caso sejam encontrados menos de três pontos nessa área,

o algoritmo aumenta recursivamente o raio do círculo R até que pelo menos três pontos sejam encontrados.

Cálculo da normal dos pontos de contato: método 2

No segundo método, utiliza-se a malha triangular préviamente contruída para o cálculo da normal do contato. Como já foi localizado o triângulo que contém o ponto na análise de estabilidade (supondo que essa análise tenha sido feita préviamente, o que é uma suposição bastante razoável), basta calcular a normal do triângulo, que pode ser obtida diretamente dos três pontos que o definem.

O vetor normal ao triângulo T definido pelos pontos A, B e C ordenados no sentido anti-horário pode ser calculado através dos seguintes vetores auxiliares

$$v_{12} = \begin{bmatrix} x_A - x_B \\ y_A - y_B \\ z_A - z_B \end{bmatrix} \quad , \tag{6.14}$$

$$v_{13} = \begin{bmatrix} x_A - x_C \\ y_A - y_C \\ z_A - z_C \end{bmatrix} \quad . \tag{6.15}$$

O vetor normal ao plano definido pelo triângulo T é dado pelo seguinte produto vetorial

$$n = v_{12} \times v_{13} \quad , \tag{6.16}$$

e sua normalização

$$\bar{n} = \frac{n}{|n|} \quad . \tag{6.17}$$

A incerteza desse valor também pode ser associado a incerteza do triângulo, proporcional à área do triângulo e à incerteza dos pontos que o compõe.

6.2 Ajuste de trajetória

Afim de ilustrar outra aplicação do modelo construído, esta seção propõe um método simples de ajuste de trajetória. Este é um método pouco robusto, onde os parâmetros devem ser ajustados em cada aplicação afim de que se obtenha um resultado satisfatório.

Os experimentos para a coleta de dados apresentados no capítulo 5 foram obtidos por meio da condução do veículo por um operador através da estrada de terra a ser

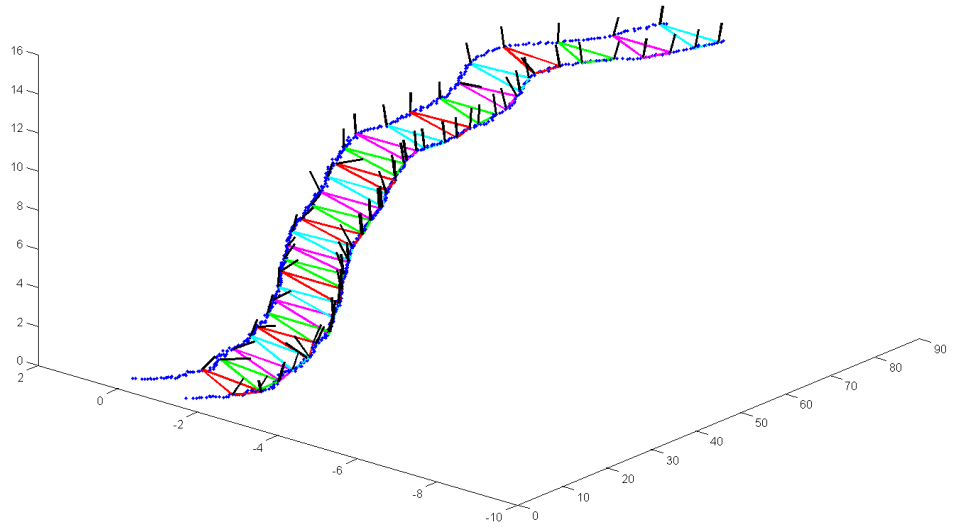


Figura 6.1: Exemplo do cálculo dos pontos de contato das rodas do veículo com o terreno e das respectivas retas normais ao solo ao longo de uma trajetória. Os triângulos ilustrados na figura indicam os três pontos de contato escolhidos para o cálculo do plano do veículo e sua inclinação.

medida. Naturalmente, o operador tentou evitar buracos e elevações que pudessem ameaçar o equilíbrio do sistema.

Esta seção propõe um ajuste da trajetória original percorrida pelo operador, e supõe que o trajeto descrito pelo veículo durante a coleta de dados é considerado aceitável. Esse percurso é adotado como uma trajetória base, e o algoritmo de ajuste irá propor pequenas variações desse trajeto.

Como descrito no capítulo 5, o experimento foi realizado por um veículo agrícola em uma região rural. A utilização do ajuste de trajetória na automação agrícola pode não se justificar apenas na otimização do percurso a partir da trajetória base definida por um operador humano, pois este caminho já é válido. Contudo, terrenos naturais são suceptíceis a intempéries da natureza (como chuvas) e a formação de buracos devido ao trânsito de veículos pesados. Por isso, com o passar do tempo, o trajeto definido pelo operador pode se tornar distante da trajetória ótima, e um ajuste de percurso pode se tornar desejável.

O ajuste da trajetória tem como entrada o caminho original gerado pelo operador. Esse percurso é posteriormente discretizado, gerando um número finito de posições estáticas do veículo. Essa discretização é feita dividindo a trajetória original em passos de comprimento l. Cada posição define um *timestep*, onde o algoritmo

apontará um deslocamento lateral ótimo em relação à posição do trajeto original. Esse deslocamento tem por objetivo a minimização dos ângulos de inclinação do veículo durante a navegação.

Para isso, foi criada uma função custo f_c a ser minimizada a cada *timestep*, onde são levados em conta 6 parâmetros: os ângulos *roll* e *pitch* do veículo ($\psi(t)$ e $\xi(t)$), o distanciamento do percurso do operador ($d_{operador}(t)$), a distância da posição lateral do *timestep* anterior ($d_{lateral}(t) = d(t) - d(t-1)$), a diferença entre o *roll* da iteração em questão e o *roll* da iteração anterior ($\Delta\psi(t) = \psi(t) - \psi(t-1)$), e a diferença entre o *pitch* da iteração em questão e o *pitch* da iteração anterior ($\Delta\xi(t) = \xi(t) - \xi(t-1)$). Cada parâmetro é ponderado por uma constante k_i.

$$f_c(t) = k_1\psi(t) + k_2\xi(t) + k_3[d_{operador}(t)]^2 + k_4[d_{lateral}(t)]^2 + k_5\Delta\psi(t) + k_6\Delta\xi(t) \quad (6.18)$$

A presença dos termos relativos ao *roll* e ao *pitch* na função custo são a base do ajuste de trajetória, visto que o objetivo desse ajuste é minimizar o risco de tombamento, fazendo uma análise estática da configuração do robô em cada *timestep*.

Os termos relacionados às variações do *roll* e *pitch* avaliam de forma grosseira a estabilidade em um aspecto cinético. É intuitivo que grandes variações do *roll* e *pitch* podem oferecer risco à estabilidade do sistema.

O termo relativo à distância entre o ponto atual e o ponto base pertencente à trajetória descrita pelo operador limita a solução do algoritmo a pontos próximos a trajetória descrita pelo operador.

O planejamento de trajetória deve obedecer as restrições de movimento de veículo não-holonômico. Contudo, esse método não verifica se essas restrições são obedecidas. Uma maneira simples de obedecer às restrições é limitar o deslocamento lateral $d_{lateral}$ do veículo de acordo com o comprimento do passo (l). Por isso também foi introduzido um termo relativo a esse deslocamento na função custo. O coeficiente correspondente (k_4) deve ser escolhido de forma que o algoritmo gere uma trajetória que respeite as restrições de movimento do veículo.

Os ângulos de *roll* e *pitch* do veículo são calculados a partir da malha de triângulos, através do cálculo dos pontos de contato do veículo com o solo.

O ponto de partida do veículo deve ser o mesmo ponto de partida do trajeto realizado pelo operador. A cada passo de l centímetros de deslocamento na trajetória original é definido um *timestep*, e o algoritmo irá calcular um deslocamento lateral que minimiza a função custo.

A minimização é feita calculando-se a função custo nesse ponto, em um ponto deslocado lateralmente $d_{lateral}$ para a direita e em outro para a esquerda. O ponto com a menor função custo é adotado, e é calculado mais um passo em sua direção.

O processo é repetido até que o novo ponto na direção tenha a função custo maior do que o ponto anterior. O ponto de mínimo local da função custo é tomado como a posição ótima naquele *timestep*.

No *timestep* seguinte, parte-se do mesmo deslocamento lateral do *timestep* anterior e o processo é repetido, com um detalhe: o ponto do percurso percorrido pelo operador sempre é analisado, pois sempre é um potencial candidato a mínimo.

Esse método de ajuste de trajetória pode ser refinado com a redução do passo l entre os pontos de dois *timesteps* consecutivos e com a redução da distância passo lateral $d_{lateral}$. Um outro método de refino é a redução recursiva do passo $d_{lateral}$ ao se chegar em um mínimo local, pois com a redução do passo, é possível que um valor menor da função custo seja encontrado.

Um exemplo da aplicação desse método de ajuste pode ser observado na Figura 6.2. Os pontos pretos fazem parte da trajetória original, e os pontos em rosa foram obtidos pelo ajuste da trajetória. O ponto de partida é marcado com a cruz vermelha, e o ponto de chegada com a cruz azul.

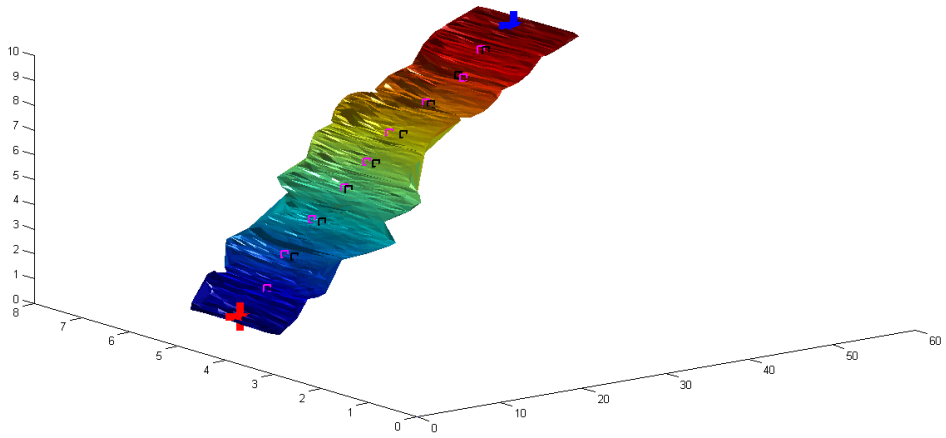


Figura 6.2: Exemplo de aplicação do método de ajuste de trajetória. Os pontos pretos fazem parte da trajetória original, e os pontos em rosa foram obtidos pelo ajuste da trajetória. O ponto de partida é marcado com a cruz vermelha, e o ponto de chegada com a cruz azul.

Capítulo 7

Conclusões

A modelagem de terrenos viabiliza diversas atividades na robótica móvel.

Nessa dissertação foi feito um levantamento bibliográfico de métodos de modelagem de terrenos. Foram apresentados trabalhos onde diferentes sensores são utilizados, e diversas técnicas para representação do ambiente são exemplificadas.

Com base no panorama geral estudado e nas técnicas apresentadas nos diversos trabalhos pesquisados, foi escolhida uma metodologia de modelagem de terrenos condizente com o método de coleta de dados utilizado.

As características da triangulação de Delaunay 2D foram apresentadas, visto que este foi o processo escolhido para a construção da superfície que representa o terreno. Foi apresentado o algoritmo incremental aleatório, e alguns detalhes da sua implementação foram discutidos.

A metodologia de modelagem de terrenos foi apresentada, considerando um veículo dotado de *laser range finder* 2D e e sensor de posição de alta precisão. O método inclui remoção de *outliers* e transformações de coordenadas para que seja gerada uma nuvem de pontos em um referencial inercial. Uma técnica de filtragem de vegetação foi apresentada, de forma a eliminar do modelo final pontos que foram obtidos a partir de arbustos e grama alta. O conjunto de pontos filtrados é utilizado como entrada em um algoritmo de triangulação de Delaunay 2D, e um pseudo-código de sua implementação foi apresentada.

A partir de experimentos para a coleta de dados em uma região agrícola, a metodologia proposta foi aplicada, gerando uma superfície 2 $\frac{1}{2}$ D que representa o terreno por debaixo de eventuais elementos da vegetação. O resultado obtido pôde ser observado e comparado com o terreno real através de imagens. Uma comparação qualitativa do filtro de vegetação pôde ser feita. A superfície gerada em um experimento foi comparada a pontos coletados em outro experimento, e uma estimativa do erro do modelo foi quantificada.

Foram ilustradas algumas das aplicações do modelo gerado, como cálculo dos pontos de contato do veículo com o terreno, cálculo dos ângulos de inclinação *roll*

e *pitch* do veículo em dada posição e cálculo da normal do contato entre as rodas do veículo e o solo. Um método simples de ajuste de trajetória também foi apresentado, de forma a exemplificar a aplicabilidade do modelo do terreno no problema de planejamento de trajetória.

Os resultados obtidos foram considerados satisfatórios. A aplicação do método na automação de veículos de campo deverá verificar a qualidade do modelo gerado, visto que grande parte dele é uma composição de técnicas já utilizadas na prática.

Propostas de trabalhos futuros

As etapas da modelagem foram implementadas *offline* utilizando *Matlab*. A implementação do método em tempo real consiste em uma direção natural do trabalho para o futuro. O processo *offline* não apresenta limitações de resolução, mas para o processamento em tempo real a resolução deve se adequar às limitações do *hardware* utilizado. A mudança na resolução pode ser feita com a seleção de medições com espaçamento maior, não necessitando demais alterações no algoritmo. Contudo, foi sugerido que a divisão dos dados em *clusters* pode apresentar ganhos significativos de desempenho no algoritmo.

Alguns parâmetros devem ser calibrados de acordo com a inclinação do terreno e a vegetação, como os parâmetros do filtro de vegetação.

Uma dificuldade associada à modelagem consiste em estimar os erros entre o modelo obtido e o terreno natural percorrido, devido principalmente à complexidade em medir ambientes naturais. Futuramente, este método será testado em uma superfície conhecida, para que uma comparação mais precisa possa ser feita.

Outra tendência de implementação futura é a combinação de imagens com pontos do laser para a filtragem da vegetação.

Referências Bibliográficas

Andersen, J., Blas, M., Ravn, O., Andersen, N., and Blanke, M. (2006). Traversable terrain classification for outdoor autonomous robots using single 2-D laser scans. *Integrated Computer-aided engineering*, 13(3):223–232.

Bakambu, J. N., Allard, P., and Dupuis, E. (2006). 3-D terrain modeling for rover localization and navigation. In *III Canadian Conference on Computer and Robot Vision*, pages 61–61. IEEE.

Baker, T. J. (1989). Automatic mesh generation for complex three-dimensional regions using a constrained delaunay triangulation. *Engineering with Computers*, 5(3-4):161–175.

Barbosa, R. L., Silva, J. F. C., Meneguette, J. M., and Gallis, R. B. A. (2003). Geração de modelo digital de terreno utilizando triangulação de delaunay e thin plate spline. *Anais do III Colóquio Brasileiro de Ciências Geodésicas*.

Bares, J., Hebert, M., Kanade, T., Krotkov, E., Mitchell, T., Simmons, R., and Whittaker, W. (1989). Ambler: An autonomous rover for planetary exploration. *The Computer Journal*, 22(6):18–26.

Barnhill, R. (1977). Representation and approximation of surfaces. *Mathematical Software Journal*, 3:69–120.

Batavia, P., Roth, S. A., and Singh, S. (2002). Autonomous coverage operations in semi-structured outdoor environments. In *International Conference on Intelligent Robots and Systems (IROS)*, volume 1, pages 743–749. IEEE.

Bellutta, P., Manduchi, R., Matthies, L., Owens, K., and Rankin, A. (2000). Terrain perception for demo iii. In *Intelligent Vehicles Symposium, 2000. IV 2000. Proc. of the IEEE*, pages 326–331. IEEE.

Broten, G. and Collier, J. (2006). Continuous motion, outdoor, 2 1/2-D grid map generation using an inexpensive nodding 2-D laser rangefinder. In *International Conference on Robotics and Automation (ICRA)*, pages 4240–4245. IEEE.

Chew, L. (1989). Constrained delaunay triangulations. *Algorithmica*, 4(1-4):97–108.

De Berg, M., Cheong, O., and Van Kreveld, M. (2008). *Computational geometry: algorithms and applications*. Springer.

Dyn, N., Levin, D., and Rippa, S. (1990). Data dependent triangulations for piecewise linear interpolation. *IMA Journal of numerical analysis*, 10(1):137–154.

El-Sheimy, N., Valeo, C., and Habib, A. (2005). *Digital terrain modeling: acquisition, manipulation, and applications*. Artech House Boston.

Fayek, R. and Wong, A. (1996). Using hypergraph knowledge representation for natural terrain robot navigation and path planning. In *International conference on Robotics and Automation*, volume 4, pages 3625–3630. IEEE.

Freitas, G. (2014). *Reconfiguração de robôs móveis com articulação ativa navegando em terrenos irregulares*. PhD thesis, Universidade Federal do Rio de Janeiro.

Freitas, G., Hamner, B., Bergerman, M., and Singh, S. (2012). A practical obstacle detection system for autonomous orchard vehicles. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Algarve, Portugal.

Goldberg, S. B., Maimone, M. W., and Matthies, L. (2002). Stereo vision and rover navigation software for planetary exploration. In *Aerospace Conf. Proc., 2002. IEEE*, volume 5, pages 5–2025. IEEE.

Guibas, L., Knuth, D., and Sharir, M. (1992). Randomized incremental construction of delaunay and voronoi diagrams. *Algorithmica Journal*, 7(1-6):381–413.

Hähnel, D., Burgard, W., and Thrun, S. (2003). Learning compact 3-D models of indoor and outdoor environments with a mobile robot. *Robotics and Autonomous Systems*, 44(1):15–27.

Hamner, B., Singh, S., Roth, S., and Takahashi, T. (2008). An efficient system for combined route traversal and collision avoidance. *Autonomous Robots Journal*, 24(4):365–385.

Hebert, M. and Krotkov, E. (1993). Local perception for mobile robot navigation in natural terrain: Two approaches. *INRIA, European Space Agency, Aerospatiale espace & defense*.

Hugentobler, M. (2004). *Terrain modelling with triangle based free-form surfaces*. PhD thesis, Citeseer.

Hygounenc, E., Jung, I., Soueres, P., and Lacroix, S. (2004). The autonomous blimp project of laas-cnrs: Achievements in flight control and terrain mapping. *International Journal of Robotics Research*, 23(4-5):473–511.

Lacaze, A., Murphy, K., and DelGiorno, M. (2002). Autonomous mobility for the demo iii experimental unmanned vehicles. In *in Assoc. for Unmanned Vehicle Systems Int. Conf. on Unmanned Vehicles (AUVSI 02*. Citeseer.

Lee, D. and Schachter, B. (1980). Two algorithms for constructing a delaunay triangulation. *International Journal of Computer and Information Sciences*, 9(3):219–242.

Li, Z., Zhu, Q., and Gold, C. (2010). *Digital terrain modeling: principles and methodology*. CRC press.

Lingemann, K., Nchter, a., Hertzberg, J., and Surmann, H. (2005). High-speed laser localization for mobile robots. *Robotics and Autonomous Systems*, 51(4):275–296.

Nguyen, D., Kuhnert, L., Jiang, T., Thamke, S., and Kuhnert, K. (2011). Vegetation detection for outdoor automobile guidance. In *International Conference on Industrial Technology (ICIT)*, pages 358–364. IEEE.

Ohno, K., Tadokoro, S., Nagatani, K., Koyanagi, E., and Yoshida, T. (2010). Trials of 3-D map construction using the tele-operated tracked vehicle kenaf at disaster city. In *International Conference on Robotics and Automation (ICRA)*, pages 2864–2870. IEEE.

Parra, C., Murrieta-Cid, R., Devy, M., and Briot, M. (1999). 3-D modelling and robot localization from visual and range data in natural scenes. In *Computer Vision Systems*, pages 450–468. Springer.

Pellenz, J., Lang, D., Neuhaus, F., and Paulus, D. (2010). Real-time 3-D mapping of rough terrain: a field report from disaster city. In *International Workshop on Safety Security and Rescue Robotics (SSRR)*, pages 1–6. IEEE.

Pfaff, P. and Burgard, W. (2006). An efficient extension of elevation maps for outdoor terrain mapping. In *Field and Service Robotics*, pages 195–206. Springer.

Piteri, M. A., Meneguete, M., Santos, A. D., and Oliveira, F. D. (2007). Triangulação de delaunay e o princípio de inserção randomizado. *II Simpósio Brasileiro de Geomática*, pages 655–663.

Plagemann, C., Mischke, S., Prentice, S., Kersting, K., Roy, N., and Burgard, W. (2009). A bayesian regression approach to terrain mapping and an application to legged robot locomotion. *Journal of Field Robotics*, 26(10):789–811.

Quak, E. and Schumaker, L. (1990). Cubic spline fitting using data dependent triangulations. *Computer Aided Geometric Design*, 7(1):293–301.

Ren, X., Berg, A. C., and Malik, J. (2005). Recovering human body configurations using pairwise constraints between parts. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 1, pages 824–831. IEEE.

Rippa, S. (1990). Minimal roughness property of the delaunay triangulation. *Computer Aided Geometric Design*, 7(6):489–497.

Sangreman, A., Freitas, G., and Costa, R. R. (2013). Modelagem de terrenos naturais através de malhas de triângulos. *II Simpósio Brasileiro de Brasileiro de Automação Inteligente*.

Sarkar, A., Srivastava, S., and Manoj, B. (2013). Elevation mapping using stereo vision enabled heterogenous multi-agent robotic network. In *Global Humanitarian Technology Conference: South Asia Satellite (GHTC-SAS)*, pages 340–345. IEEE.

Schafer, H., Hach, A., Proetzsch, M., and Berns, K. (2008). 3-D obstacle detection and avoidance in vegetated off-road terrain. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 923–928. IEEE.

Shirkhodaie, A., Amrani, R., and Tunstel, E. (2005). Visual terrain perception modeling of space planetary robotic systems based on soft computing classifiers. In *IEEE International Workshop on Robot and Human Interactive Communication*, pages 42–47. IEEE.

Sibson, R. (1978). Locally equiangular triangulations. *The Computer Journal*, 21(3):243–245.

Siegwart, R. and Nourbakhsh, I. (2004). *Intro to Autonomous Mobile Robots*. MIT press.

Sithole, G. and Vosselman, G. (2003). Report: Isprs comparison of filters. *ISPRS commission III, working group*, 3.

Sotoodeh, S. (2006). Outlier detection in laser scanner point clouds. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, XXXVI(5):297–302.

Sreenivasan, S. V. (1994). *Actively Coordinated Wheeled Vehicle Systems*. PhD thesis, Mechanical Eng. Dept., Ohio State University.

Underwood, J., Hill, A., and Scheding, S. (2007). Calibration of range sensor pose on mobile platforms. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 3866–3871. IEEE.

Vandapel, N., Donamukkala, R., and Hebert, M. (2006). Experimental results in using aerial ladar data for mobile robot navigation. In *Field and Service Robotics*, pages 103–112. Springer.

Vandapel, N., Huber, D., Kapuria, A., and Hebert, M. (2004). Natural terrain classification using 3-D ladar data. In *IEEE International Conference on Robotics and Automation (ICRA)*, volume 5, pages 5117–5122. IEEE.

Watson, D. F. (1981). Computing the n-dimensional delaunay tessellation with application to voronoi polytopes. *The computer journal*, 24(2):167–172.

Wettergreen, D., Moreland, S., Skonieczny, K., Jonak, D., Kohanbash, D., and Teza, J. (2010). Design and field experimentation of a prototype lunar prospector. *International Journal of Robotics Research*, 29(12):1550–1564.

Apêndice A

Códigos Implementados no Matlab

IncrementalF.m - Algoritmo incremental aleatório que realiza a triangulação de Delaunay 2D de um conjunto de pontos

```
1
2  function [Points,Triangles,tri] = IncrementalF(Points)
3
4
5  %Ordenar os pontos de forma aleatória
6  n=length(Points);
7  random= randperm(n);
8  Pointss=Points;
9  for i=1:n
10     Points(i,:)=Pointss(random(i),:);
11 end
12 Pointss=[];
13
14 %Translação dos dados
15 Translacao = min(Points);
16 for i=1:n
17     Points(i,:)=[Points(i,1)-Translacao(1) ...
           Points(i,2)-Translacao(2) Points(i,3)-Translacao(3)];
18 end
19
20
21
22
23 %% Criar triangulo inicial
24
25 Maximos = max(Points);
26 Minimos = min(Points);
27
28 d = Maximos(1)-Minimos(1);
```

```
29 h = Maximos(2)-Minimos(2);
30
31 Points(n+1,:) = [Minimos(1)+ d/2 (Maximos(2)+2*h) 0];
32 Points(n+2,:) = [Minimos(1)-(d/2) Minimos(2)-(h/10) 0];
33 Points(n+3,:) = [Maximos(1)+(d) Minimos(2)-(h/10) 0];
34
35
36 Triangles = [n+1 n+2 n+3 0 0 0];
37 Lines = [n+1 n+2; n+1 n+3; n+2 n+3];
38
39
40 % Verificar se o ponto p esta no triangulo T
41 T=1;
42 p=1;
43
44 d = Points(p,:);
45 a = Points(Triangles(T,1),:);
46 b = Points(Triangles(T,2),:);
47 c = Points(Triangles(T,3),:);
48 [Inside] = InsideTriangle(a, b, c, d);
49
50 if Inside == 1 % Caso o ponto esteja dentro do triangulo em ...
       questão, inicia-se a triangulação
51
52     %Criando 3 linhas: ligando o ponto p aos vertices do triangulo
53     Lines(length(Lines)+1,:) = [p Triangles(T,1)];
54     Lines(length(Lines)+1,:) = [p Triangles(T,2)];
55     Lines(length(Lines)+1,:) = [p Triangles(T,3)];
56
57     %Criando 3 novos triangulos e avisando que o triangulo ...
           original deu
58     %lugar a eles
59     Lenghtriangle = 1;
60
61     Triangles(Lenghtriangle+1,:) = [p Triangles(T,1) ...
           Triangles(T,2) 0 0 0];
62     Triangles(Lenghtriangle+2,:) = [p Triangles(T,1) ...
           Triangles(T,3) 0 0 0];
63     Triangles(Lenghtriangle+3,:) = [p Triangles(T,2) ...
           Triangles(T,3) 0 0 0];
64
65     Triangles(1,4) = Lenghtriangle+1;
66     Triangles(1,5) = Lenghtriangle+2;
67     Triangles(1,6) = Lenghtriangle+3;
68
69
70
```

```
71 end
72
73 p=2;
74
75 for T=1:4 %temos somente 4 triangulos ate aqui
76
77     if Triangles(T,4) == 0
78
79
80
81         d = Points(p,:);
82         a = Points(Triangles(T,1),:);
83         b = Points(Triangles(T,2),:);
84         c = Points(Triangles(T,3),:);
85         [Inside] = InsideTriangle(a, b, c, d);
86
87         if Inside == 1 % Caso o ponto esteja dentro do triangulo ...
               em questão, inicia-se a triangulação
88
89             %Criando 3 linhas: ligando o ponto p aos vertices do ...
                   triangulo
90             Lines(length(Lines)+1,:) = [p Triangles(T,1)];
91             Lines(length(Lines)+1,:) = [p Triangles(T,2)];
92             Lines(length(Lines)+1,:) = [p Triangles(T,3)];
93
94             %Criando 3 novos triangulos e avisando que o ...
                   triangulo original deu
95             %lugar a eles
96             Lenghtriangle = 4;
97
98             Triangles(Lenghtriangle+1,:) = [p Triangles(T,1) ...
                   Triangles(T,2) 0 0 0];
99             Triangles(Lenghtriangle+2,:) = [p Triangles(T,1) ...
                   Triangles(T,3) 0 0 0];
100            Triangles(Lenghtriangle+3,:) = [p Triangles(T,2) ...
                   Triangles(T,3) 0 0 0];
101
102            Triangles(T,4) = Lenghtriangle+1;
103            Triangles(T,5) = Lenghtriangle+2;
104            Triangles(T,6) = Lenghtriangle+3;
105
106
107            pi = Triangles(T,1);
108            pj = Triangles(T,2);
109            Tri_p_pi_pj = Lenghtriangle+1;
110            [Troca,Points,Lines,Triangles]=CorrigeAresta(p,pi,pj, ...
                   Tri_p_pi_pj,Points,Lines,Triangles);
```

```
111
112
113             pi = Triangles(T,1);
114             pj = Triangles(T,3);
115             Tri_p_pi_pj = Lenghtriangle+2;
116
117             [Troca,Points,Lines,Triangles]=CorrigeAresta(p,pi,pj, ...
                    Tri_p_pi_pj,Points,Lines,Triangles);
118
119
120
121
122             pi = Triangles(T,2);
123             pj = Triangles(T,3);
124             Tri_p_pi_pj = Lenghtriangle+3;
125             [Troca,Points,Lines,Triangles]=CorrigeAresta(p,pi,pj, ...
                    Tri_p_pi_pj,Points,Lines,Triangles);
126
127         end
128
129     end
130
131
132
133 end
134
135
136 %Finalmente, for geral
137
138 for p=3:n
139
140     Lenghtriangle = length(Triangles);
141     candidatos = [1 0 0];
142     d = Points(p,:);
143
144
145         while true
146
147             while true
148
149                 T=candidatos(1);
150                 a = Points(Triangles(T,1),:);
151                 b = Points(Triangles(T,2),:);
152                 c = Points(Triangles(T,3),:);
153                 [Inside] = InsideTriangle(a, b, c, d);
154                 if Inside == 1
155                     candidatos = [Triangles(T,4), ...
```

```
                        Triangles(T,5), Triangles(T,6)];
156                 break
157             end
158
159             T=candidatos(2);
160             a = Points(Triangles(T,1),:);
161             b = Points(Triangles(T,2),:);
162             c = Points(Triangles(T,3),:);
163             [Inside] = InsideTriangle(a, b, c, d);
164             if Inside == 1
165                 candidatos = [Triangles(T,4), ...
                        Triangles(T,5), Triangles(T,6)];
166                 break
167             end
168
169             T=candidatos(3);
170             a = Points(Triangles(T,1),:);
171             b = Points(Triangles(T,2),:);
172             c = Points(Triangles(T,3),:);
173             [Inside] = InsideTriangle(a, b, c, d);
174             if Inside == 1
175                 candidatos = [Triangles(T,4), ...
                        Triangles(T,5), Triangles(T,6)];
176                 break
177             end
178
179         end
180
181         if candidatos(1) == 0% Achamos o triangulo que ...
                contem o ponto. inicia-se a triangulação
182 %           %teste - debugando
183 %           %Triangulo=T
184 %
185             %Criando 3 linhas: ligando o ponto p aos ...
                    vertices do triangulo
186             Lines(length(Lines)+1,:) = [p Triangles(T,1)];
187             Lines(length(Lines)+1,:) = [p Triangles(T,2)];
188             Lines(length(Lines)+1,:) = [p Triangles(T,3)];
189
190             %Criando 3 novos triangulos e avisando que o ...
                    triangulo original deu
191             %lugar a eles
192
193
194             Triangles(Lenghtriangle+1,:) = [p Triangles(T,1) ...
                    Triangles(T,2) 0 0 0];
195             Triangles(Lenghtriangle+2,:) = [p Triangles(T,1) ...
```

```
                    Triangles(T,3) 0 0 0];
196                 Triangles(Lenghtriangle+3,:) = [p Triangles(T,2) ...
                    Triangles(T,3) 0 0 0];
197
198                 Triangles(T,4) = Lenghtriangle+1;
199                 Triangles(T,5) = Lenghtriangle+2;
200                 Triangles(T,6) = Lenghtriangle+3;
201
202
203
204
205                 pi = Triangles(T,1);
206                 pj = Triangles(T,2);
207                 Tri_p_pi_pj = Lenghtriangle+1;
208                 [Troca,Points,Lines,Triangles]=CorrigeAresta(p, ...
                    pi,pj,Tri_p_pi_pj,Points,Lines,Triangles);
209
210                 pi = Triangles(T,1);
211                 pj = Triangles(T,3);
212                 Tri_p_pi_pj = Lenghtriangle+2;
213
214                 [Troca,Points,Lines,Triangles]=CorrigeAresta(p, ...
                    pi,pj,Tri_p_pi_pj,Points,Lines,Triangles);
215
216
217
218                 pi = Triangles(T,2);
219                 pj = Triangles(T,3);
220                 Tri_p_pi_pj = Lenghtriangle+3;
221                 [Troca,Points,Lines,Triangles]=CorrigeAresta(p, ...
                    pi,pj,Tri_p_pi_pj,Points,Lines,Triangles);
222                 break
223             end
224
225         end
226
227
228
229 end
230
231
232
233  [Points,Lines,tri]=ApagaLinhaBase(Points,Lines,Triangles); % ...
       Apaga triângulo auxiliar e as arestas ligadas a ele
234
235
236 end
```

InsideTriangle.m - Verifica se o ponto d está contido no triângulo abc

```
function [Inside] = InsideTriangle(a, b, c, d)


Vab = [a(1)-b(1) a(2)-b(2) 0];
Vac = [a(1)-c(1) a(2)-c(2) 0];
Vbc = [b(1)-c(1) b(2)-c(2) 0];
Vad = [a(1)-d(1) a(2)-d(2) 0];
Vbd = [b(1)-d(1) b(2)-d(2) 0];

Inside = 0;
if sign(cross(Vab, Vad)) == sign(cross(Vab, Vac))
    if sign(cross(Vac, Vad)) == sign(cross(Vac, Vab))
        if sign(cross(Vbc, Vbd)) == sign(cross(Vbc,-Vab))
            Inside = 1;
        end
    end
end

end
```

FindAresta.m - Encontra a aresta [pi pj] na lista de arestas "Lines". Também encontra o triangulo [pi pj pk], que é adjacente ao triângulo [p pi pj] e contém a aresta [pi pj].

```

function ...
    [Aresta,TrianguloAdjacente,pk]=FindAresta(p,pi,pj,Points, ...
    Lines,Triangles)


% Essa função encontra a aresta [pi pj] na lista de arestas ...
    "Lines". Também
% encontra o triangulo [pi pj pk], que é adjacente ao triângulo ...
    [p pi pj] e
% contém a aresta [pi pj].



Aresta=0;
TrianguloAdjacente=0;


N = length(Lines);
```

```
16 i=1;
17
18 while true
19     if Lines(i,:) == [pi pj]
20         break
21     end
22
23     if Lines(i,:) == [pj pi]
24         break
25     end
26
27     i = i+1;
28     if i > length(Lines)
29         i = 0;
30         erroNaoAcheiLinha = 1
31         break
32     end
33 end
34
35
36
37 j=1;
38
39 while true
40
41     if Triangles(j,4) == 0
42
43         if Triangles(j,1) == pi;
44             if Triangles(j,2) == pj;
45                 if Triangles(j,3) == p;
46                 else
47                     pk = Triangles(j,3);
48                     break
49                 end
50             end
51             if Triangles(j,3) == pj;
52                 if Triangles(j,2) == p;
53                 else
54                     pk = Triangles(j,2);
55                     break
56                 end
57             end
58         end
59
60         if Triangles(j,2) == pi;
61             if Triangles(j,1) == pj;
62                 if Triangles(j,3) == p;
```

```
63                  else
64                      pk = Triangles(j,3);
65                      break
66                  end
67              end
68              if Triangles(j,3) == pj;
69                  if Triangles(j,1) == p;
70                  else
71                      pk = Triangles(j,1);
72                      break
73                  end
74              end
75          end
76
77          if Triangles(j,3) == pi;
78              if Triangles(j,1) == pj;
79                  if Triangles(j,2) == p;
80                  else
81                      pk = Triangles(j,2);
82                      break
83                  end
84              end
85              if Triangles(j,2) == pj;
86                  if Triangles(j,1) == p;
87                  else
88                      pk = Triangles(j,1);
89                      break
90                  end
91              end
92          end
93
94      end
95
96      j=j+1;
97      if j > length(Triangles)
98          pk = 0;
99          % j=0;
100         % NaoAcheiTriangulo = 1
101         break
102     end
103 end
104
105
106
107 Aresta=i;
108 TrianguloAdjacente=j;
109
```

```
110
111
112 end
```

cruza.m - Verifica se as retas [2 3] e [1 k] se cruzam

```
1
2  function [cruzou]=cruza(p1,p2,p3,pk,Points)
3
4
5
6  %ver se as retas [2 3] e [1 k] se cruzam ( Sao as diagonais do
7  %quadrilatero em questão)
8
9
10 c(1) = top(p1,p2,p3,Points);
11 c(2) = top(pk,p2,p3,Points); % c3 e c4 nao necessariamente ...
       precisam ser feitos. otimizar isso.
12 c(3) = top(p2,p1,pk,Points);
13 c(4) = top(p3,p1,pk,Points);
14
15 if c(1)==2 || c(2)==2 || c(3)==2 || c(4)==2
16     % d=2; % um dos pontos de uma reta pertence à outra reta
17     cruzou=2 % retas não concorrentes
18
19 else
20     if c(1) ≠ c(2) && c(3) ≠ c(4)
21         cruzou=1; % retas concorrentes
22     else
23         cruzou=0; % retas não concorrentes
24     end
25 end
26
27
28
29 end
```

CorrigeAresta.m - Verifica se uma aresta é ilegal, e realiza o *flip* caso positivo

```
1
2  function [Troca,Points,Lines,Triangles]=CorrigeAresta(p,pi,pj, ...
       Tri_p_pi_pj,Points,Lines,Triangles)
3
4
5
6  %Localiza a aresta e o Triângulo a ser trocado
```

```
7  [Aresta,TrianguloAdjacente,pk]=FindAresta(p,pi,pj,Points, ...
       Lines,Triangles);
8
9
10 if pk ≠ 0
11
12
13
14
15     % Primeiro é testado se a aresta [pi pj] é ilegal, através ...
           do critério do
16     % ponto dentro do círculo.
17
18     p1=p;
19     p2=pi;
20     p3=pj;
21     pk;
22
23     %Precisamos botar os pontos p1 p2 e p3 na ordem antihorária.
24
25     V12 = [Points(p1,1)-Points(p2,1) Points(p1,2)-Points(p2,2) 0];
26     V13 = [Points(p1,1)-Points(p3,1) Points(p1,2)-Points(p3,2) 0];
27     ProdutoVetorial=cross(V12, V13);
28     if ProdutoVetorial(3) > 0
29         %troquei=0
30     else
31         x=p2; %variavel auxiliar pra armazenar o valor
32         p2=p3;
33         p3=x;
34         %troquei=1
35     end
36
37     %agora que os pontos estao ordenados corretamente, ...
           verificamos se o ponto
38     %pk está dentro do círculo circunscrito ao triangulo p1 p2 p3
39
40
41     D = [Points(p1,1) Points(p1,2) ...
           (Points(p1,1)^2)+(Points(p1,2)^2) 1;
42         Points(p2,1) Points(p2,2) ...
               (Points(p2,1)^2)+(Points(p2,2)^2) 1
43         Points(p3,1) Points(p3,2) ...
               (Points(p3,1)^2)+(Points(p3,2)^2) 1
44         Points(pk,1) Points(pk,2) ...
               (Points(pk,1)^2)+(Points(pk,2)^2) 1 ];
45
46     if det(D) > 0
```

```
47
48          [cruzou]=cruza(p,pi,pj,pk,Points);
49          if cruzou == 1 % só podemos realizar o flip se as ...
                diagonais do quadrilátero se cruzam
50
51
52              Troca=1;
53
54              % Inicia-se o FLIP da aresta
55
56
57              Lines(Aresta,:) = [p pk];
58
59              Triangles(length(Triangles)+1,:) = [p pi pk 0 0 0];
60              TrianguloAverificar1 = length(Triangles);%variavel ...
                    auxiliar que servira de entrada da fase ...
                    recursiva, onde é feita a verificação desse ...
                    triangulo
61
62              Triangles(length(Triangles)+1,:) = [p pj pk 0 0 0];
63              TrianguloAverificar2 = length(Triangles);
64
65
66              %referenciando os novos triangulos nos seus ...
                    triangulos de origem
67
68              Triangles(Tri_p_pi_pj, 4) = length(Triangles)-1;
69              Triangles(Tri_p_pi_pj, 5) = length(Triangles);
70
71              Triangles(TrianguloAdjacente, 4) = length(Triangles)-1;
72              Triangles(TrianguloAdjacente, 5) = length(Triangles);
73
74
75              %%%%%%%%%% Função recursiva %%%%%
76
77              [Troca,Points,Lines,Triangles]=CorrigeAresta(p,pi, ...
                    pk,TrianguloAverificar1,Points,Lines,Triangles);
78              [Troca,Points,Lines,Triangles]=CorrigeAresta(p,pk, ...
                    pj,TrianguloAverificar2,Points,Lines,Triangles);
79          else
80              Troca=0;
81
82          end
83
84
85
86
```

```
87      else
88          Troca=0;
89      end
90  else
91      Troca = 0;
92  end
93
94
95  end
```

ApagaLinhaBase.m - Apaga as arestas ligadas aos pontos que formam o triângulo auxiliar

```
1
2   function [Points,Lines,tri]=ApagaLinhaBase(Points,Lines,Triangles);
3
4   n = length(Points);
5   n = n-3;
6
7
8   for i=1:length(Lines)
9
10      if (Lines(i,1)==n+1) || (Lines(i,2)==n+1)
11          Lines(i,:)=[1 1];
12      end
13
14      if (Lines(i,1)==n+2) || (Lines(i,2)==n+2)
15          Lines(i,:)=[1 1];
16      end
17
18      if (Lines(i,1)==n+3) || (Lines(i,2)==n+3)
19          Lines(i,:)=[1 1];
20      end
21
22  end
23
24
25  tri=[0 0 0];
26  j=1;
27
28  for i=1:length(Triangles)
29      if Triangles(i,4)==0
30
31          if (Triangles(i,1)≠n+1) && (Triangles(i,2)≠n+1) && ...
                (Triangles(i,3)≠n+1) && (Triangles(i,1)≠n+2) && ...
                (Triangles(i,2)≠n+2) && (Triangles(i,3)≠n+2) && ...
```

```
            (Triangles(i,1)≠n+3) && (Triangles(i,2)≠n+3) && ...
            (Triangles(i,3)≠n+3)
32              tri(j,1)=Triangles(i,1);
33              tri(j,2)=Triangles(i,2);
34              tri(j,3)=Triangles(i,3);
35              j=j+1;
36          end
37
38      end
39
40  end
41
42
43  end
```

top.m - Verifica se o ponto p está acima da reta [p1 p2]

```
1
2   function verdade=top(p,p1,p2,Points)
3
4
5
6   point = Points(p,:);
7   ponto1 = Points(p1,:);
8   ponto2 = Points(p2,:);
9
10
11
12
13  x0 = point(1);
14  y0 = point(2);
15
16  x1 = ponto1(1);
17  y1 = ponto1(2);
18  x2 = ponto2(1);
19  y2 = ponto2(2);
20
21  %coeficientes da reta ax+b=y
22
23  a = (y1-y2)/(x1-x2);
24  b = y1 - (a*x1);
25
26  yref = (a*x0)+ b;
27
28  if y0 > yref
29      top = 1;
```

```
30 else
31     if y0 < yref
32         top = 0;
33     else
34         top = 2;
35     end
36 end
37
38 verdade = top;
39 end
```

FiltragemDaVegetacao.m - Algoritmo que classifica cada ponto como vegetação ou terreno, e elimina os pontos classificados como vegetação do *array* de pontos

```
1
2
3  angulo = 60;
4
5  limite = 0.2;
6
7  n = length(Points);
8  V = zeros(n,4);
9  V(:,1) = Points(:,1);
10 V(:,2) = Points(:,2);
11 V(:,3) = Points(:,3);
12
13
14 InicioDaclassificacao=1
15
16 for i=1:n;
17     i
18
19
20     for j=2:n
21
22         if i ≠ j
23
24             d = (V(i,3)-V(j,3));
25
26             if d > limite
27
28
29
30
31                 teta = atan2(sqrt((V(i,1)-V(j,1))^2 + ...
                       (V(i,2)-V(j,2))^2),V(i,3)-V(j,3))*180/pi;
```

```
32
33                  if teta < angulo
34                      V(i,4) = 1;
35                      break
36                  end
37
38              end
39
40          end
41
42
43      end
44
45  end
46
47
48
49  InicioDafiltragem=1
50
51  Vf = V;
52
53  i=1;
54  while true
55
56      if i > length(Vf)
57          break
58      end
59
60
61      if Vf(i,4) == 1;
62          Vf(i,:) = [];
63          i=i-1;
64      end
65      i=i+1;
66  end
67
68
69  Vf(:,4) = [];
70
71  Points = Vf;
```