ESCOLA POLITÉCNICA DA
UNIVERSIDADE DE SÃO PAULO

Nota final
9,5 (nove e cinco)

PAULO ROBERTO GODOI DE OLIVEIRA

Construção de Mapa de Ambiente Baseado em Visão Omnidirecional Estéreo com Espelho Duplo de Perfil Hiperbólico

São Paulo
2005

PAULO ROBERTO GODOI DE OLIVEIRA

Construção de Mapa de Ambiente Baseado em Visão Omnidirecional Estéreo com Espelho Duplo de Perfil Hiperbólico

Monografia apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do título de Engenheiro

Orientador: Prof. Dr. Eduardo Lobo Lustosa Cabral

São Paulo
2005

FICHA CATALOGRÁFICA

Oliveira, Paulo Roberto Godoi de
Construção de mapas de ambiente baseado em visão omnidirecional estéreo com espelho duplo de perfil hiperbólico / P.R.G. de Oliveira. -- São Paulo, 2005.
118 p.

Trabalho de Formatura - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia Mecatrônica e de Sistemas Mecânicos.

1.Visão omnidirecional 2.Visão estéreo 3.Espelho duplo 4.Espelho de perfil hiperbólico I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia Mecatrônica e de Sistemas Mecânicos II.t.

RESUMO

Este trabalho consistiu no desenvolvimento de um algoritmo para construção de mapas de ambiente para robôs móveis em um ambiente estruturado. O mapa é construído a partir de imagens adquiridas por um sistema de visão omnidirecional estéreo baseado em um espelho duplo de perfil hiperbólico. A partir de uma única imagem obtida, utilizando-se algoritmos de visão estéreo, obtêm-se as distâncias de objetos presentes no ambiente ao sistema de visão. A partir da correspondência de várias imagens tomadas em diferentes posições, cria-se o mapa do ambiente. Levando em conta as três etapas de construção de mapas de ambiente, sendo estas, alinhamento, correspondência e integração, este trabalho emprega o Trimmed Iterative Closest Point para a correspondência e um caso especial para pontos coplanares do quaternio unitário para o alinhamento. O resultado é um mapa de ambiente de alta qualidade que traz ainda informação da auto-localização do robô no mesmo.

Prefácio

Este projeto consiste no desenvolvimento de um algoritmo para construção de mapas de ambiente para robôs móveis em um ambiente estruturado, ou seja, que pode ser descrito através de primitivas geométricas. O mapa é construído a partir de imagens adquiridas por um sistema de visão omnidirecional estéreo baseado em um espelho duplo de perfil hiperbólico. A partir de uma única imagem obtida, utilizando-se algoritmos de visão estéreo, obtém-se as distâncias de objetos presentes no ambiente ao sistema de visão. A partir da correspondência de várias imagens tomadas em diferentes posições cria-se o mapa do ambiente.

Esse relatório consiste em um relatório científico, que apresenta os desenvolvimentos realizados durante um ano de vigência do projeto.

Existem três etapas para a construção de mapas de ambiente, o alinhamento, que visa obter a transformação de duas imagens diferentes em um único sistema de coordenadas, a correspondência, que visa corresponder os pontos das duas imagens e a integração, que incorpora os pontos correspondidos em um único sistema de coordenadas para a construção de um modelo final.

Na etapa do alinhamento a transformação entre dois sistemas de coordenadas cartesianas pode ser dada como o resultado de um movimento de corpo rígido e pode, portanto, ser decomposto em uma rotação e uma translação. Foi implementado um caso especial, para pontos co-planares, utilizando-se o quaternio unitário , descrito por Horn (1987).

Na etapa de correspondência foi implementada uma versão mais robusta do ICP (Iterative Closest Point) descrita nos trabalhos de Chetverikov et al. (2002a) e Chetverikov e Stepanov (2002b), sendo chamada de Trimmed ICP (TrICP) que utiliza o método dos mínimos quadrados (Least Trimmed Squares - LTS) para cada iteração do ICP. O alinhamento é realizado considerando um fator de sobreposição entre o conjunto de dados, ou seja não é mais considerado que todos os pontos da nova imagem terão um correspondente na imagem anterior. O ICP original é um caso particular do TrICP para quando a taxa de sobreposição é de 100%. Duas alterações adicionais foram adicionadas ao algoritmo original, A primeira mudança é que o número de iteração máximo é definido pelo tamanho do maior conjunto de pontos, do mapa ou da amostra. A segunda mudança, e também a mais importante, é que agora o algoritmo não necessariamente considera a convergência monotônica, sendo guardado um registro do mínimo global durante as iterações até que algum dos critérios de parada seja

atingido. Quando isso ocorre, os dois erros quadráticos médios são comparados, o da amostra mínima global e a da amostra que atingiu o critério de parada. A amostra considerada como saída do algoritmo é a do mínimo global se o erro quadrático médio da amostra do critério de parada for a partir de 10% superior ao da amostra do mínimo global, caso contrário a saída será a amostra do critério de parada.

É implementado um outro método de selecionar os possíveis pontos correspondentes no algoritmo do TrICP, utilizando um algoritmo de programação dinâmica, porém os resultados não são bons e a alteração é descartada.

A etapa de integração é realizada através da inserção dos novos pontos correspondidos, seguida de duas filtragens, uma que eclode pontos muito próximos uns dos outros e os substitui por um ponto com o valor médio e uma filtragem de ruído.

Uma sala virtual é criada usando-se o programa POV RAY, para testar o algoritmo de correspondência. Nesta sala são criadas as seqüências de imagens utilizadas para validar o algoritmo de mapa de ambientes.

Um primeiro algoritmo é desenvolvido e testado, chegando a resultados de qualidade média. Neste algoritmo seis tentativas de correspondência são realizadas.

Posteriormente, aproveitando a experiência ganha com o primeiro algoritmo, um segundo algoritmo foi desenvolvido, onde se obteve resultados de alta qualidade. Neste algoritmo dez tentativas de correspondência são realizadas.

Nota-se que além do mapa global o algoritmo também realiza um cálculo da auto-localização do robô no mesmo. Ressalta-se que nenhuma informação adicional é fornecida ao algoritmo além da seqüência de imagens, como por exemplo, odômetria. As distâncias aos objetos e a auto-localização do robô é feita diretamente das informações provenientes da reconstrução tridimensional e da correspondência da imagens fotografadas. Contudo se observa, que o tempo computacional exigido pelo algoritmo impossibilita que ele seja usado em uma aplicação que necessite da geração do mapa em tempo real.

Esse relatório está organizado como se segue. Na seção 1 é apresentado um resumo das atividades que foram realizadas nestes dois semestres, seguidas de um detalhamento das mesmas e na seção 2 temos as conclusões do trabalho. Os algoritmos desenvolvidos no MATLAB estão apresentados com detalhes no Anexo I, as imagens fotografadas do ambiente simulado estão no Anexo I.

Índice

Lista de Figuras

Lista de Tabelas

CONSTRUÇÃO DE MAPAS DE AMBIENTE BASEADO EM VISÃO OMNIDIRECIONAL ESTÉREO COM ESPELHO DUPLO DE PERFIL HIBERBÓLICO

1. Introdução

Para a navegação de robôs móveis é necessário extrair informações do ambiente de maneira que essas forneçam subsídios para a navegação. A aquisição de informação é feita através de sensores. A escolha do sensor mais adequado depende do tipo de tarefa de navegação que se pretende implementar e da autonomia desejada.

Inúmeros sensores podem ser utilizados, dentre esses, citam-se: ultra-som, infravermelho, GPS, Lasers e visão computacional. Sensores de ultra-som e lasers são sensíveis a superfícies refletoras, tornando-se úteis para medirem distâncias. Sensor de infravermelho possui boa resolução angular, tornando-se útil para detectar a presença ou não de objetos. GPS podem ser utilizados para determinar a localização do robô. Sensores baseados em câmeras são capazes de capturar imagens de forma contínua e fornecer a posição de objetos.

Em certas aplicações de visão computacional é interessante ter um campo visual mais abrangente do que o obtido com uma câmera convencional. Imagens panorâmicas contém informação de todo o ambiente em uma única imagem. Um sistema de visão panorâmica ou omnidirecional fornece uma imagem de 360° do ambiente. Devido a essa característica, visão panorâmica tem se tornado uma ferramenta importante, especialmente na área de robôs móveis.

A visão omnidirecional (do latim "*omnis*", em português "oni", significa tudo ou todo) tem sido muito utilizada em inúmeras aplicações, dentre elas citam-se: vigilância remota, reconstrução tridimensional de ambientes, rastreamento visual e navegação de robôs móveis.

Robôs móveis são programados para realizarem tarefas de maneira autônoma. A autonomia de um robô fica fortemente condicionada por sua capacidade de perceber o ambiente de navegação. Um robô deve ser capaz de interagir coerentemente com o seu mundo, sendo capaz de recuperar descrições úteis, usando informação adquirida pelos sensores e utilizando eficientemente estas descrições com o objetivo de realizar uma tarefa especifica. A

tarefa mais importante que um robô móvel deve realizar autonomamente é a navegação, sem colisão com obstáculos.

Uma outra tarefa que proporciona uma maior autonomia a um robô móvel é a construção de mapas de ambiente. Um mapa é uma representação da estrutura do ambiente que pode ser utilizado para diversas tarefas, inclusive navegação. Nas tarefas de exploração de ambientes o robô navega adquirindo informações para a construção de um mapa. O robô pode ainda, baseando-se em um mapa existente, navegar resolvendo os problemas de localização, planejamento de trajetória e verificar se a posição final foi atingida.

1.1 Revisão Bibliográfica

1.1.1 Sistema de Visão Estéreo Omnidirecional

Existem inúmeras formas de aquisição de imagens omnidirecionais. Câmeras giratórias, composição de um conjunto de câmeras dispostas em diferentes posições do ambiente; lentes especiais como "olho de peixe" e a conciliação de meios refletores e refratores (espelhos e câmeras) são alguns exemplos.

No caso de câmeras giratórias, a câmera gira com velocidade de rotação constante em torno do seu eixo vertical. A resolução horizontal da imagem panorâmica não depende da resolução angular da câmera e sim da resolução angular da rotação. A desvantagem da utilização de câmeras giratórias está relacionada ao longo tempo de aquisição de imagens, tornando-as inaplicáveis para problemas de tempo real e para ambientes dinâmicos, Yagi (1999).

A composição de câmeras no ambiente pode ser feita de tal maneira a obter uma imagem panorâmica. No entanto, é difícil obter um sistema compacto em razão da existência de inúmeras câmeras.

Lentes especiais, tais como "olho de peixe", fornecem a visão de um hemisfério, no entanto a imagem obtida tem boa resolução no centro e baixa resolução na periferia. Assim, as projeções dos objetos presentes no mundo se tornam distorcidas pela projeção na parte periférica da imagem.

Uma outra maneira de se obter visão omnidirecional é com a utilização de meios catadióptricos. Conciliando uma câmera e um espelho convexo, sendo este alinhado com o

eixo óptico da câmera, obtém-se um sensor potencialmente útil, por exemplo, para aplicação em tarefas de navegação de robôs móveis. Dentre os vários tipos de sistemas de visão omnidirecional, este é o mais atrativo devido à sua simplicidade e compacticidade. Inúmeros pesquisadores têm investigado e utilizado esse tipo de sistema nos últimos anos para as variadas aplicações. Entre esses inúmeros autores pode-se citar alguns, como por exemplo, Baker e Nayar (1998), Chahl e Srinivasan (1997), Conroy e Moore (1999), Gaspar et al (2002), Geyer e Daniilidis (2000), Hicks e Bajcsy (1999a), Ollis et al. (1999), Svoboda et al. (1998) e Yamazawa et al (1999).

Além da consideração da estrutura utilizada para obter um sistema omnidirecional uma outra propriedade que qualifica esse tipo de sistema é a existência ou não de um centro único de projeção. O centro único de projeção é uma característica que dependendo da aplicação pode ser importante em um sistema de visão omnidirecional. O centro único de projeção garante a unicidade entre um ponto do espaço tridimensional e a sua projeção em um único ponto da imagem omnidirecional.

O formato do espelho determina a formação da imagem de um sistema de visão catadióptrico. Muitas formas diferentes de espelhos têm sido utilizadas e testadas. Os primeiros espelhos a serem utilizados foram os espelhos de formatos simples, tais como, planos, cônicos, hiperbólicos, parabólicos, elípticos e esféricos, devido à sua simplicidade geométrica e de fabricação.

Diversos autores apresentam estudos detalhados da utilização de espelhos em formato de hipérboles, parábolas, elipses e esferas em sistemas de visão omnidirecional, entre esses pode-se citar Svoboda et al. (1998) e Yamazawa et al. (1999). Os espelhos em formato de cone têm boa resolução na parte periférica, enquanto os espelhos esféricos têm boa resolução na parte central. O espelho hiperbólico tem a vantagem de boa resolução em ambas às partes central e periférica, sendo que a abrangência do seu campo visual é definida no projeto do perfil do espelho. O espelho parabólico tem resolução intermediária entre a dos espelhos esférico e hiperbólico, porém contém o maior campo visual obtido dentre esses tipos de espelhos convexos.

Uma das grandes vantagens de um sistema catadióptrico é que o espelho pode ser projetado com um formato especial de modo a preservar certas propriedades do espaço tridimensional na imagem. A escolha das propriedades que devem ser preservadas na

formação da imagem está naturalmente relacionada com a aplicação desejada. Na literatura são apresentadas inúmeras possibilidades para a propriedade a ser preservada.

Uma outra ferramenta muito importante em visão computacional é a visão estéreo. Com visão estéreo é possível determinar a distância de objetos dentro do campo visual do sistema de visão. Com a medida de distância dos objetos ao sistema pode-se pensar em realizar diversas tarefas, como por exemplo: planejamento de trajetória para navegação de robôs móveis; localização de objetos no espaço para manipulação por robôs; reconstrução tridimensional de ambientes para interação virtual; metrologia dimensional de peças; inspeção visual de equipamentos; etc.

A visão estéreo baseia-se no princípio de comparação das imagens de uma mesma cena obtidas por duas câmeras (ou uma única com movimento) posicionadas em locais diferentes mas próximos entre si. O problema fundamental em visão estéreo é determinar onde pixels, pontos ou outras características correspondentes estão presentes nas duas imagens; esse problema é chamado de correspondência. A procura por pontos correspondentes de forma eficiente, em geral, é realizada ao longo da chamada curva epipolar. A procura na linha epipolar restringe o problema de busca bidimensional para um problema de busca unidimensional. A forma da curva epipolar depende da geometria do sistema de visão.

Uma vez obtidos os dois pontos correspondentes nas duas imagens, pode-se calcular a distância do ponto físico no espaço ao sistema de visão através da técnica de triangulação, ou seja, dados os pontos correspondentes nas duas imagens e a localização relativa dos dois sistemas de visão, a partir de considerações geométricas calcula-se à distância do ponto no espaço.

Devido ao ângulo de visão de 360^{o} sistemas de visão estéreo omnidirecional são ideais para aplicações de navegação de robôs e reconstrução de ambientes. Nessas aplicações de visão estéreo, múltiplas câmeras ou câmeras móveis são freqüentemente usadas para compensar o pequeno campo de visão de sistemas de visão estéreo convencionais. Dois espelhos substituem diversas câmeras na visão estéreo omnidirecional.

Nene e Nayar (1998) apresentam diversas configurações de sistemas estéreos omnidirecionais que utilizam espelhos planos, parabólicos, elípticos e hiperbólicos, todos assegurando centro único de projeção.

Visão estéreo omnidirecional baseada em um par de espelhos axialmente coaxiais (ou espelho duplo com duas seções coaxiais) com campos de visão que se intersectam, como

mostra a Figura 1, foi sugerido inicialmente por Southwell et al. (1996). Nesse sistema os centros dos dois espelhos são colineares com o eixo da câmera e os espelhos têm um formato radialmente simétrico em torno do seu eixo.

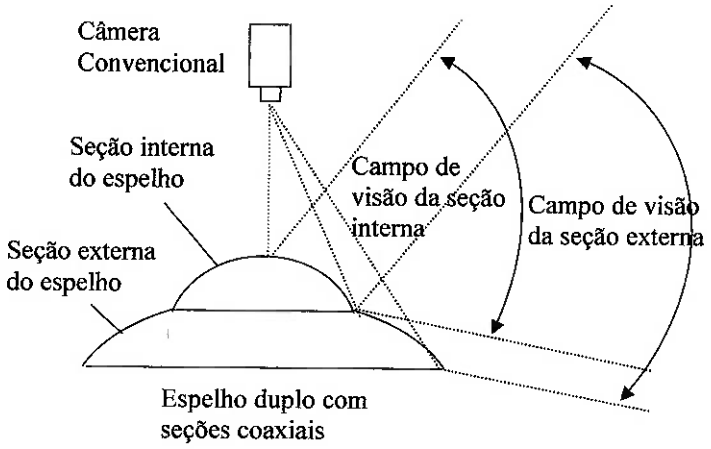

Figura 1- Sistema de visão estéreo omnidirecional baseada em um par de espelhos coaxiais radialmente simétricos.

Existem inúmeras formas de se implementar um sistema de visão estéreo omnidirecional. Ollis et al. (1999) apresentam um estudo sobre configurações de sistemas que proporcionam visão estéreo omnidirecional. As diferentes configurações utilizam um ou mais espelhos, e também uma ou mais câmeras, de forma a obter um campo abrangente. Uma análise do campo visual, da resolução da imagem e da precisão na distância obtidos em cada configuração é realizada.

Cabral et al. (2004), desenvolveram um sistema de visão omnidirecional estéreo baseado em um par de espelhos coaxiais com perfil parabólico capaz de realizar reconstrução tridimensional em tempo real.

1.1.2 Navegação de Robôs

Existem basicamente dois métodos para navegação de robôs: a arquitetura deliberativa e a arquitetura reativa. Na navegação de robôs baseada na arquitetura deliberativa, todas as tarefas requeridas para a navegação são planejadas. De acordo com esta arquitetura,

algoritmos de planejamento de trajetória, auto-localização e detecção de obstáculos, são implementados para a realização destas tarefas.

Na arquitetura reativa, Brooks (1986), o robô navega com base em comportamentos, que são atitudes do robô face à dinâmica do ambiente. Com este tipo de arquitetura o robô navega sem que seja sabido qual é a possível trajetória do robô a ser realizada durante a navegação, pois esta depende de quais comportamentos irão atuar.

Minguez et al. (2001) implementam uma arquitetura híbrida combinando as duas arquiteturas de navegação. Os comandos de desvios de obstáculos são provenientes do comportamento reativo para uma navegação livre de colisões, entretanto o robô planeja a trajetória constrói um mapa de grades de ocupação com as informações provenientes do sensor laser.

Utilizando-se visão computacional pode-se empregar diferentes metodologias de navegação, tais como: fluxo óptico (Giachetti et al. (1998) e Santos-Victor e Sandini (1997)) e reconhecimento de marcos no ambiente para extração de informações relevantes para a navegação (Gaspar e Santos-Victor (1999)).

Uma técnica muito empregada em trabalhos de navegação é a de visão estéreo. A partir da informação de distancias de robô a objetos presentes no ambiente, trajetórias são planejadas e tarefas de navegação, como a detecção de obstáculos, são executadas.

Diferentes configurações do sistema estéreo podem ser utilizadas, como por exemplo: um sistema estéreo divergente, obtido com um par de câmeras dispostas lateralmente, Santos-Victor et al (1995); um sistema estéreo trinocular, obtido com três câmeras, Little e Murray (1998), Murray e Jennings (1997) e Murray e Little (1998); Um sistema estéreo convencional, obtido com as câmeras dispostas horizontalmente, Kumano et al. (2000); sistemas formados por câmeras e espelhos, podendo ser dispostos verticalmente, Koyasu et al. (2002), Deccó (2004) e Cabral et al.(2004), ou horizontalmente, Svoboda et al. (1998); e outras.

Inúmeros trabalhos de navegação de robôs móveis utilizam sistemas compostos por câmeras e espelhos para obterem um campo visual abrangente. Utilizando um sistema composto por uma só câmera e um espelho hiperbólico, além de também utilizar a informação de fluxo óptico, Matsumoto et al. (1999) implementam uma navegação de robôs em corredores.

Gaspar e Santos-Victor (1999) utilizam um sistema catadióptrico para realizar uma navegação de um robô móvel, baseado no rastreamento de marcos no ambiente. /as imagens são obtidas com um espelho esférico.

Gaspar et al. (2000), utilizando o método das aparências sobre imagens de um mapa topológico do ambiente, estimam a posição do robô ao longo do corredor. O sistema omnidirecional utilizado é formado por uma câmera e uma espelho esférico.

Resumidamente, uma maior autonomia pode ser fornecida ao robô através de um conjunto de fatores, que estão relacionados com a arquitetura implementada, com o sensor utilizado e com a tarefa realizada.

1.1.3 Construção de mapas de ambiente

Robôs móveis navegam em ambientes e realizam tarefas com certo grau de autonomia. Uma tarefa elementar de navegação consiste no robô partir de uma posição inicial e atingir uma posição final. Outras tarefas que o robô pode realizar para facilitar a navegação consistem no planejamento da trajetória, auto-localização, rastreamento visual de marcos do ambiente, detecção e desvios de obstáculos e construção de mapas de ambiente.

Para a construção de mapas de ambiente, pode-se classificar o ambiente segundo a sua estrutura, podendo ser estruturado ou não estruturado. Desouza e Kak (2002) propõem que um ambiente estruturado contenha alguma regularidade na informação, que possa ser extraída para fornecer subsídios para a navegação, como retas e planos. Um ambiente não estruturado não contém nenhuma regularidade na informação, não existem restrições geométricas que possam ser parametrizadas ou utilizadas para navegação.

Outra classificação no tipo de ambiente é se ele é interno ou externo, podendo ser estruturado ou não. Giachetti et al. (1998), como exemplo de navegação em ambientes externos estruturados estudaram a navegação de carros autônomos que navegam em estradas. Como exemplo de navegação em ambientes externos não-estruturados tem-se os trabalhos de navegação em planetas, como Marte, por Goldberg et al. (2002). Ambientes internos como salas, corredores e laboratórios, podem ser considerados como estruturados, pois em sua maioria contêm retas e planos, assim sendo a navegação pode ser auxiliada por mapa que descreve a estrutura ou a posição dos objetos no ambiente.

Tipicamente, existem duas abordagens para a representação de mapas de ambiente. São os mapas geométricos ou topológicos. Mapas geométricos contêm informação da geometria do ambiente, da posição dos objetos e distâncias entre esses. Como exemplos, citam-se as grades de ocupação, conexão de polígonos e gráficos de Voronoi. Os mapas topológicos não possuem qualquer informação sobre a geometria do ambiente, eles são representados por elos conectados a nós. Nos nós o robô pode tomar certas ações, como virar à direita ou à esquerda, e reconhecer marcos para se localizar no ambiente. O reconhecimento dos marcos visuais dá ao robô uma localização qualitativa no ambiente, obtendo sua posição em termos do quanto está mais próximo ou mais distante do alvo. Na navegação com mapas geométricos a localização é quantitativa, sabendo o robô qual sua exata posição no ambiente.

Segundo Manessis (2003), a construção de mapas para navegação consiste de três etapas:

- Alinhamento: processo que visa obter a transformação de duas medições distintas em um único sistema de coordenadas através da informação que relaciona as duas medições;
- Correspondência: Processo realizado após o alinhamento dos pontos, consiste em corresponder os pontos das duas medições;
- Integração: processo que incorpora os pontos correspondidos no único sistema de coordenadas para a construção do modelo final.

O processo de alinhamento consiste em determinar qual é a transformação que levou um conjunto de pontos P para um conjunto de pontos Q. O conhecimento dessa transformação alinha os dois conjuntos de dados permitindo a descrição dos pontos em um único sistema de coordenadas. Com as informações descritas no mesmo sistema de coordenadas utiliza-se algum critério para corresponder o conjunto de pontos P ao conjunto de Q.

A maioria dos métodos de correspondência se baseiam no algoritmo Iterative Closest Point (ICP), originalmente desenvolvido por Besl e Mckay (1992), que utiliza o critério da mínima distancia entre os pontos. O ICP, calcula iterativamente a transformação rígida e a correspondência entre os pontos, convergindo para a solução da correspondência que minimiza a somatória das distâncias ao quadrado entre os pontos de dois conjuntos de dados.

Uma versão robusta do ICP é descrita nos trabalhos de Chetverikov et al. (2002a) e Chetverikov e Stepanov (2002b) na qual o Trimmed ICP (TrICP) utiliza a informação dos mínimos quadrados para cada iteração do ICP. O alinhamento é realizado considerando um

fator de sobreposição entre o conjunto de dados. O ICP original é um caso particular do TrICP para quando a taxa de sobreposição é de 100%.

Um método diferente de correspondência é proposto por Yagi et al. (1995) e Deccó (2004) nos quais a métrica utilizada é o mínimo ângulo entre os pontos. O método é proposto para a correspondência de mapas adquiridos com um sistema de visão omnidirecional composto por uma câmera e um espelho hiperbólico. Como o ângulo azimutal do ponto plano da imagem descreve a orientação do ponto no ambiente, pode-se utilizá-lo como medida para encontrar a correspondência entre um par de mapas. Nesse método, para cada novo mapa adquirido, os valores dos ângulos azimutais dos pontos dos mapas são dispostos em ordem crescente e são correspondidos pela mínima diferença angular entre eles.

Um outro método de correspondência é proposto por Nakamura e Ishiguro (2002), que projetam um espelho que mapeia pontos do chão linearmente com o aumento dos pixels no plano da imagem e realizam a construção de um mapa plano do ambiente para a navegação. Primeiramente são extraídas retas de um par de imagens. Para cada par é calculado o histograma angular das retas nas direções x e y. Pela máxima correlação cruzada entre os histogramas de um par de imagens encontra-se a transformação rígida que alinha o par.

Resumidamente, para a construção de mapas de ambiente, podemos classificar o ambiente como Estruturado ou Desestruturado e Externo ou Externo, representá-lo de uma forma Geométrica e/ou Topológica, e dentre as metodologias utilizadas para a construção de mapas utiliza-se normalmente o ICP, Proximidade Angular e Histograma Angular.

1.2 Justificativas

Robôs móveis são programados para realizarem tarefas de maneira autônoma. A autonomia de um robô fica fortemente condicionada por sua capacidade de perceber o ambiente de navegação. Um robô deve ser capaz de interagir coerentemente com o seu mundo, sendo capaz de recuperar descrições úteis, usando informação adquirida pelos sensores e utilizando eficientemente estas descrições com o objetivo de realizar uma tarefa especifica. A mais importante tarefa que um robô móvel deve realizar autonomamente é a navegação dentro de ambientes, sem colisão com obstáculos.

Os mapas de ambiente são construídos para serem utilizados em tarefas de navegação, fornecendo ao robô subsídios para o reconhecimento do ambiente de navegação, planejamento de trajetória, auto-localização e a realização de demais tarefas.

A definição do ambiente com sendo estruturado cobre os principais campos de aplicação dos robôs moveis, uma vez que, em geral, os ambientes construídos pelo homem são estruturados, contendo alguma regularidade na informação, que possa ser extraída para fornecer subsídios para a navegação, como retas e planos. Ambientes internos como salas, corredores e laboratórios, podem ser considerados como estruturados, pois em sua maioria contêm retas e planos.

Um sistema de visão estéreo omnidirecional baseado em um espelho duplo com ambas as seções com perfil hiperbólico, possui um centro único de projeção e garante o alinhamento perfeito das duas imagens. Além disso, esse arranjo garante naturalmente que as linhas epipolares são linhas radiais na imagem omnidirecional e linhas paralelas nas imagens perspectivas obtidas através da projeção da imagem omnidirecional em um cilindro. Estas características fazem esse sistema ser ideal para recuperação tridimensional em tempo real e, assim, para fornecer informação para a construção de um mapa detalhado do ambiente, de forma a facilitar a navegação do robô.

2. Algoritmo de construção de mapas de ambiente

Existem três etapas para a construção de mapas de ambiente, o alinhamento, que visa obter a transformação de duas imagens diferentes em um único sistema de coordenadas, a correspondência, que visa corresponder os pontos das duas imagens e a integração, que incorpora os pontos correspondidos em um único sistema de coordenadas para a construção de um modelo final.

Na etapa do alinhamento a transformação entre dois sistemas de coordenadas cartesianas pode ser dada como o resultado de um movimento de corpo rígido e pode, portanto, ser decomposto em uma rotação e uma translação. Foi implementado um caso especial, para pontos co-planares, utilizando-se o quaternio unitário , descrito por Horn (1987).

Na etapa de correspondência foi implementada uma versão mais robusta do ICP (Iterative Closest Point) descrita nos trabalhos de Chetverikov et al. (2002a) e Chetverikov e Stepanov (2002b), sendo chamada de Trimmed ICP (TrICP) que utiliza o método dos mínimos quadrados (Least Trimmed Squares - LTS) para cada iteração do ICP. O alinhamento é realizado considerando um fator de sobreposição entre o conjunto de dados, ou seja, não é mais considerado que todos os pontos da nova imagem terão um correspondente na imagem anterior. O ICP original é um caso particular do TrICP para quando a taxa de sobreposição é de 100%. Duas alterações adicionais foram adicionadas ao algoritmo original, A primeira mudança é que o número de iteração máximo é definido pelo tamanho do maior conjunto de pontos, do mapa ou da amostra. A segunda mudança, e também a mais importante, é que agora o algoritmo não necessariamente considera a convergência monotônica, sendo guardado um registro do mínimo global durante as iterações até que algum dos critérios de parada seja atingido. Quando isso ocorre, os dois erros quadráticos médios são comparados, o da amostra mínima global e a da amostra que atingiu o critério de parada. A amostra considerada como saída do algoritmo é a do mínimo global se o erro quadrático médio da amostra do critério de parada for a partir de 10% superior ao da amostra do mínimo global, caso contrário a saída será a amostra do critério de parada.

É implementado um outro método de selecionar os possíveis pontos correspondentes no algoritmo do TrICP, utilizando um algoritmo de programação dinâmica, porém os resultados não são bons e a alteração é descartada.

A etapa de integração é realizada através da inserção dos novos pontos correspondidos, seguida de duas filtragens, uma que eclode pontos muito próximos uns dos outros e os substitui por um ponto com o valor médio e uma filtragem de ruído.

2.1 Etapa de alinhamento

Eggert et al. (1997) fazem uma comparação entre quatro métodos para estimar a transformação de corpo rígido que leva a imagem nova para a imagem antiga. Os métodos estudados diferem na forma como as componentes de transformação são representadas e também no modo como eles minimizam a função de erro quadrático médio. Uma breve descrição desses métodos é apresentada a seguir.

Arun et al. (1987) desenvolveram uma solução baseada no cálculo da decomposição do valor singular (Singular Value Decomposition, SVD) da matriz derivada da representação convencional da rotação e translação. Nesta abordagem, a rotação é representada usando-se uma matriz 3x3 convencional. A translação é um vetor tridimencional. Resumidamente, calcula-se a decomposição do valor singular da matriz de correlação, $\mathbf{H}$, cujas componentes são dadas por:

$$h_{ij} = \sum_{i=1}^{N} \sum_{j=1}^{N} d_{ci} m_{cj} , \tag{1}$$

entre os dois conjuntos de pontos d e m, onde respectivamente d_{ci} e m_{ci} correspondem as coordenadas dos pontos dos conjuntos d e m com relação ao seu centróide. A decomposição é dada na forma,

$$\mathbf{H} = \mathbf{U} \wedge \mathbf{V}^{\mathrm{T}} . \tag{2}$$

A rotação, $\hat{\mathbf{R}}$, é então calculada por:

$$\hat{\mathbf{R}} = \mathbf{V}\mathbf{U}^{\mathrm{T}} , \tag{3}$$

e a translação, $\hat{\mathbf{T}}$, calculada na forma,

$$\hat{\mathbf{T}} = \overline{m} - \hat{\mathbf{R}}\overline{d} , \tag{4}$$

onde $\overline{m}$ e $\overline{d}$ são respectivamente as posições médias dos conjuntos de pontos m e d.

Uma abordagem similar, porém baseada em explorar as propriedades de ortogonormalidade da matriz de rotação, calcula os autovalores e autovetores (orthonormal eigensystem, OM) de uma matriz derivada, é apresentada por Horn et al. (1988). Como no algoritmo anterior, calcula-se a matriz de correlação $\mathbf{H}$, porém agora defini-se uma matriz $\mathbf{M} = \mathbf{H}^{\mathrm{T}}$. Assumindo que a matriz $\mathbf{M}$ seja não-singular, pode-se decompô-la como $\mathbf{M} = \mathbf{U}.\mathbf{S}$, onde:

$$\mathbf{U} = \mathbf{M}\mathbf{S}^{-1}, \qquad \mathbf{S} = \left(\mathbf{M}^{\mathrm{T}}\mathbf{M}\right), \tag{5}$$

sendo $\mathbf{U}$ uma matriz ortonormal. A rotação, $\hat{\mathbf{R}}$, pode então ser expressa na forma:

$$\hat{\mathbf{R}} = \mathbf{M}\left(\frac{u_1 u_1^{\mathrm{T}}}{\sqrt{\lambda_1}} + \frac{u_2 u_2^{\mathrm{T}}}{\sqrt{\lambda_2}} + \frac{u_3 u_3^{\mathrm{T}}}{\sqrt{\lambda_3}}\right), \tag{6}$$

onde $\{\lambda_j\}$ e $\{u_j\}$ são respectivamente os auto-valores e auto-vetores correspondentes a matriz $\mathbf{S}$. A translação é calculada como na equação (4).

O terceiro algoritmo, também desenvolvido por Horn (1987), envolve o cálculo dos autovalores e autovetores de uma matriz que representa as componentes rotacionais como quaternios unitários (unit quaternions, UQ). Ao invés de usar a matriz 3x3 ortonormal padrão para representar a rotação, o quaternio unitário é empregado. Se a rotação total for representada por um ângulo θ em relação a um eixo $\mathbf{a} = [a_x, a_y, a_z]$ que passa pela origem, com norma unitária, o quaternio unitário equivalente é um vetor tetradimencional definido como,

$$\mathbf{q} = [\cos(\theta/2), \mathrm{sen}(\theta/2)a_x, \mathrm{sen}(\theta/2)a_y, \mathrm{sen}(\theta/2)a_z]. \tag{7}$$

Usando propriedades conhecidas dos quaternios, monta-se uma matriz P, 4x4, que é composta por combinações de somas de produtos de pontos coordenados dos dois conjuntos de pontos, similar à $\mathbf{H}$. Define-se $\mathbf{P}$ como sendo:

$$\mathbf{P} = \begin{bmatrix} S_{xx}+S_{yy}+S_{zz} & S_{yz}-S_{zy} & S_{zx}-S_{xz} & S_{xy}-S_{yx} \\ S_{yz}-S_{zy} & S_{xx}-S_{yy}-S_{zz} & S_{xy}+S_{yx} & S_{zx}+S_{xz} \\ S_{zx}-S_{xz} & S_{xy}+S_{yx} & -S_{xx}+S_{yy}-S_{zz} & S_{yz}+S_{zy} \\ S_{xy}-S_{yx} & S_{zx}+S_{xz} & S_{yz}+S_{zy} & -S_{xx}-S_{yy}+S_{zz} \end{bmatrix}, \tag{8}$$

onde, $S_{ab} = \sum_{i=1}^{N} m_{ci_a} d_{ci_b}$, análoga a h_{ij}.

Calcula-se então o vetor $\hat{\mathbf{q}} = [\hat{q}_0, \hat{q}_1, \hat{q}_2, \hat{q}_3]$, definido por um quaternio unitário que representa a rotação, $\hat{\mathbf{R}}$. O vetor $\hat{\mathbf{q}}$ é definido como o auto-vetor associado ao maior auto-valor positivo da matriz $\mathbf{P}$. Calcula-se então a rotação, $\hat{\mathbf{R}}$, na forma:

$$\hat{\mathbf{R}} = \begin{bmatrix} \left(\hat{q}_0^2 + \hat{q}_1^2 - \hat{q}_2^2 - \hat{q}_3^2\right) & 2\left(\hat{q}_1\hat{q}_2 - \hat{q}_0\hat{q}_3\right) & 2\left(\hat{q}_1\hat{q}_3 + \hat{q}_0\hat{q}_2\right) \\ 2\left(\hat{q}_2\hat{q}_1 + \hat{q}_0\hat{q}_3\right) & \left(\hat{q}_0^2 - \hat{q}_1^2 + \hat{q}_2^2 - \hat{q}_3^2\right) & 2\left(\hat{q}_2\hat{q}_3 - \hat{q}_0\hat{q}_1\right) \\ 2\left(\hat{q}_3\hat{q}_1 - \hat{q}_0\hat{q}_2\right) & 2\left(\hat{q}_3\hat{q}_2 + \hat{q}_0\hat{q}_1\right) & \left(\hat{q}_0^2 - \hat{q}_1^2 - \hat{q}_2^2 + \hat{q}_3^2\right) \end{bmatrix}. \tag{9}$$

A translação é calculada como na equação (4).

Ainda outros autovalores e autovetores são analisados quando as componentes da translação e rotação são representadas usando dual quaternios (dual quaternions, DQ), como apresentado na quarta técnica por Walker et al. (1991). Neste algoritmo, a função de erro a ser minimizada é:

$$\Sigma^2 = \sum_{i=1}^{L} \alpha_i \left\| n_{1i} - \hat{\mathbf{R}} n_{2i} \right\|^2 + \sum_{i=1}^{L} \beta_i \left\| d_i - \hat{\mathbf{R}} m_i - \hat{\mathbf{T}} \right\|^2, \tag{10}$$

onde, $\{n_{1i}\}$ e $\{n_{2i}\}$ são dois conjuntos de dimensão L correspondendo a vetores unitários normais, e $(\{\alpha_i\}, \{\beta_i\})$ são fatores de ponderação refletindo a confiabilidade dos dados obtidos. Este método incorpora ambas informações da posição e direção na minimização. Neste método a rotação e a translação são representadas juntas usando um quaternio dual. Como o nome sugere, é um quaternio que consiste de duas partes, $\mathbf{q}_d = [r, s]$. Ao invés de descrever o movimento como uma rotação em torno da origem e depois uma translação, ele pode ser modelado simultaneamente como uma rotação em torno de uma linha particular no espaço tridimensional e uma translação ao longo da mesma.

Dado que a linha tenha direção $\mathbf{n} = [n_x, n_y, n_z]$, com norma unitária, e passe por um ponto $\mathbf{p} = [p_x, p_y, p_z]$, e o movimento total seja definido por um ângulo θ e uma distância t, então as duas componentes tetradimensionais do quaternio dual são:

$$\mathbf{r} = \begin{bmatrix} \text{sen}(\theta/2)\mathbf{n} \\ \cos(\theta/2) \end{bmatrix},$$
$$\mathbf{s} = \begin{bmatrix} t/2\cos(\theta/2)\mathbf{n} + \text{sen}(\theta/2)(\mathbf{p}\hat{}\,\mathbf{n}) \\ -t/2\,\text{sen}(\theta/2) \end{bmatrix}. \tag{11}$$

Estes dois componentes tem as propriedades $\mathbf{r}^{\mathrm{T}}\mathbf{r} = 1$ e $\mathbf{s}^{\mathrm{T}}\mathbf{r} = 0$.
Pode-se então reescrever o erro na seguinte forma:

$$\sum^2 = \hat{\mathbf{r}}^{\mathrm{T}}\mathbf{C}_1\hat{\mathbf{r}} + \hat{\mathbf{s}}^{\mathrm{T}}\mathbf{C}_2\hat{\mathbf{s}} + \hat{\mathbf{s}}^{\mathrm{T}}\mathbf{C}_3\hat{\mathbf{r}} + \mathbf{C}_4, \tag{12}$$

onde:

$$\mathbf{C}_1 = \sum_{i=1}^{N} \mathbf{Q}(\tilde{\mathbf{d}}_i)\mathbf{W}(\tilde{\mathbf{m}}_i),\ \mathbf{C}_2 = N\mathbf{I},\ \mathbf{C}_3 = 2\sum_{i=1}^{N}\left(\mathbf{W}(\tilde{\mathbf{m}}_i) - \mathbf{Q}(\tilde{\mathbf{d}}_i)\right),\ \mathbf{C}_4 = \sum_{i=1}^{N}\left(\tilde{\mathbf{m}}_i^{\mathrm{T}}\tilde{\mathbf{m}}_i + \tilde{\mathbf{d}}_i^{\mathrm{T}}\mathbf{d}_i\right), \tag{13}$$

sendo, $\tilde{\mathbf{m}}_i$ e $\tilde{\mathbf{d}}_i$ são vetores 4-D definidos com os três primeiros componentes iguais a metade do valor original na coordenada do ponto e o quarto componente igual a zero, e $\mathbf{Q}(\mathbf{v})$ e $\mathbf{W}(\mathbf{v})$, matrizes 4x4, definidas como:

$$\mathbf{Q}(\mathbf{v}) = \begin{bmatrix} v_3\mathbf{I} + \mathbf{K}(\mathbf{v}_{0..2}) & \mathbf{v}_{0..2} \\ -\mathbf{v}^{\mathrm{T}}{}_{0..2} & v_3 \end{bmatrix},\ \mathbf{K}(\mathbf{v}) = \begin{bmatrix} 0 & -v_2 & v_1 \\ v_2 & 0 & -v_0 \\ -v_1 & v_0 & 0 \end{bmatrix},$$
$$\mathbf{W}(\mathbf{v}) = \begin{bmatrix} v_3\mathbf{I} - \mathbf{K}(\mathbf{v}_{0..2}) & \mathbf{v}_{0..2} \\ -\mathbf{v}^{\mathrm{T}}{}_{0..2} & v_3 \end{bmatrix} \tag{14}$$

onde $\mathbf{v}_{0..2} = [v_0, v_1, v_2]$ representa os primeiros três elementos de um vetor $\mathbf{v} = [v_0, v_1, v_2, v_3]$, qualquer.

Minimizando a equação (12), obtemos uma matriz definida como:

$$\mathbf{A} = \frac{1}{4N}\mathbf{C}_3^{\mathrm{T}}\mathbf{C}_3 - \mathbf{C}_1. \tag{15}$$

Define-se então $\hat{\mathbf{r}} = [\hat{r}_0, \hat{r}_1, \hat{r}_2, \hat{r}_3]$ como sendo o auto-vetor correspondente ao maior auto-valor positivo da matriz $\mathbf{A}$. Usa-se o vetor $\hat{\mathbf{r}}$ para calcular a matriz de rotação $\hat{\mathbf{R}}$ como se segue:

$$\hat{\mathbf{R}} = \left(\hat{r}_3^2 - \hat{\mathbf{r}}_{0..2}^{\mathrm{T}}\hat{\mathbf{r}}_{0..2}\right)\mathbf{I} + 2\hat{\mathbf{r}}_{0..2}^{\mathrm{T}}\hat{\mathbf{r}}_{0..2} + 2\hat{r}_3\,\mathbf{K}\left(\hat{\mathbf{r}}_{0..2}\right). \tag{16}$$

Calcula-se a translação baseado no vetor $\hat{\mathbf{r}}$. Define-se $\hat{\mathbf{s}}$:

$$\hat{\mathbf{s}} = -\left(\mathbf{C}_2 + \mathbf{C}_2^{\mathrm{T}}\right)^{-1}\mathbf{C}_3 = -\frac{1}{2N}\mathbf{C}_3\hat{\mathbf{r}}, \tag{17}$$

para então definir a translação $\hat{\mathbf{T}}$, como:

$$\hat{\mathbf{T}} = \mathbf{W}\left(\hat{\mathbf{r}}\right)^{\mathrm{T}}\hat{\mathbf{s}}, \tag{18}$$

As performances dos algoritmos são classificadas, segundo Eggert et al. (1997), qualitativamente de 1 a 4, sendo 1 melhor e 4 pior. As colocações são baseadas no conjunto de respostas à diferentes níveis de ruído e tamanho do conjunto de pontos. São dadas comparações para a acurácia/robustez usando conjuntos de dados em 3-D com e sem ruído, estabilidade da resposta em conjuntos de dados degenerados corrompidos sem ruído, ruído isotrópico (ruído-i) e ruído anisotrópico (ruído-a), tempo de execução total para tamanhos grande e pequenos de conjuntos de dados. O resultado da comparação é mostrado na tabela 1.

Tabela 1 - Comparação qualitativa da performance dos algoritmos, segundo Eggert et al. (1997).

Método	acurácia 3D		estabilidade 2D			estabilidade 1D			estabilidade 0D			tempo de exec.	
	ideal	ruído	ideal	ruído-i	ruído-a	ideal	ruído-i	ruído-a	ideal	ruído-i	ruído-a	Peq. N	Gd. N
SVD	1	1	1	1	1	2	2	2	3	1	1	2	2
OM	3	1	4	4	4	1	1	1	1	1	1	1	4
UQ	2	1	2	1	1	3	3	3	1	1	1	2	3
DQ	4	1	3	1	1	4	4	4	4	4	4	4	1

Dentre os algoritmos estudados, escolheu-se a implementação de um caso especial, para pontos co-planares, utilizando-se o quaternio unitário (UQ), descrito por Horn (1987).

A transformação entre dois sistemas de coordenadas cartesianas pode ser dada como o resultado de um movimento de corpo rígido e pode, portanto, ser decomposto em uma rotação e uma translação. Na prática, as medições não são exatas e então maior exatidão em determinar os parâmetros da transformação será obtida usando-se mais de três pontos.
A solução encontrada também apresenta simetria, ou seja, quando aplicado o problema de encontrar-se a melhor transformação de um sistema de coordenadas A para um outro B, ela fornece a inversa exata da melhor transformação de B para A.

2.1.1 Equações de transformação de corpo rígido

Dentre os algoritmos estudados, escolheu-se a implementação de um caso especial, para pontos co-planares, do quaternio unitário(UQ) descrito por Horn (1987).

Considera-se dois conjuntos de dados m e p. Os pontos destes conjuntos são representados pelos vetores de posição $\mathbf{r}_{m,i}$ e $\mathbf{r}_{p,j}$ respectivamente. O algoritmo tenta levar a amostra p para a m. No caso estudado, m será o mapa e p a nova amostra.

Define-se todas as medidas com relação ao centro das amostras casadas na fase de correspondência do TrICP, resultando em $Npo = Np.\lambda$ pontos casados considerados, sendo Np o número de pontos da amostra e Nm o número de pontos do mapa e λ o grau de encobrimento da amostra com relação ao mapa. Obtém-se portanto $\mathbf{r}'_{m,i}$ e $\mathbf{r}'_{p,i}$ respectivamente as posições dos pontos do mapa relativo ao centro dos pontos casados do mapa e os pontos da amostra relativo ao centro dos pontos casados da amostra.

$$\mathbf{r}'_{m,i} = \mathbf{r}_{m,i} - \bar{\mathbf{r}}_m, \qquad \mathbf{r}'_{p,i} = \mathbf{r}_{p,i} - \bar{\mathbf{r}}_p, \tag{19}$$

tal que $\bar{\mathbf{r}}_m$ e $\bar{\mathbf{r}}_p$ são as médias dos pontos casados respectivamente do modelo e da amostra.

$$\bar{\mathbf{r}}_p = \frac{1}{Npo}\sum_{i=1}^{Npo}\mathbf{r}_{p,i}, \qquad \bar{\mathbf{r}}_m = \frac{1}{Npo}\sum_{i=1}^{Npo}\mathbf{r}_{m,i}. \tag{20}$$

2.1.1.1 Encontrando a rotação

Uma vez que os dois conjuntos de pontos estão em um mesmo plano, deve-se encontrar a rotação no conjunto da amostra que minimiza a soma das distâncias ao quadrado entre os pontos correspondentes, ou seja, deseja-se minimizar:

$$\sum_{i=1}^{Npo} \left\| \mathbf{r}'_{m,i} - \mathbf{r}'_{p,i} \right\|^2 . \tag{21}$$

Considerando que a amostra esteja rotacionada de um ângulo θ com relação ao mapa, então para minimizar (3), deve-se maximizar:

$$\sum_{i=1}^{Npo} \mathbf{r}'_{m,i} \mathbf{r}'_{p,i} \cos\theta \tag{22}$$

ou

$$C\cos\theta + S\,\mathrm{sen}\,\theta , \tag{23}$$

onde:

$$\begin{aligned} C &= \sum_{i=1}^{Npo} \left(\mathbf{r}'_{m,i} . \mathbf{r}'_{p,i} \right); \\ S &= \sum_{i=1}^{Npo} \left(\mathbf{r}'_{m,i} \wedge \mathbf{r}'_{p,i} \right). \end{aligned} \tag{24}$$

Assim,

$$\mathrm{sen}\,\theta = \pm \frac{S}{\sqrt{S^2 + C^2}} \text{ e } \cos\theta = \pm \frac{C}{\sqrt{S^2 + C^2}} . \tag{25}$$

2.1.1.2 Encontrando a translação

Procura-se por uma transformação na forma:

$$\mathbf{r}_m = s\mathbf{R}(\mathbf{r}_p) + \mathbf{r}_o , \tag{26}$$

do sistema de coordenadas da amostra para o mapa, onde s é o fator de escala, $\mathbf{r}_o$ é a translação de referência e $\mathbf{R}(\mathbf{r}_p)$ representa o conjunto de pontos da amostra rotacionados.

Usando-se o fato de que a rotação é uma operação linear e mantêm os comprimentos dos vetores, têm-se:

$$\left\|\mathbf{R}\left(\mathbf{r}_p\right)\right\|^2 = \left\|\mathbf{r}_p\right\|^2, \tag{27}$$

onde $\|\mathbf{r}\|^2 = \mathbf{r}.\mathbf{r}$ é o comprimento ao quadrado do vetor $\mathbf{r}$.

A não ser que os conjuntos de dados sejam perfeitos, não será possível encontrar uma escala, uma rotação e uma translação que respeite a equação a baixo em todos os pontos, ao invés disso, encontra-se um erro residual.

$$\mathbf{e}_i = \mathbf{r}_{m,i} - s\mathbf{R}\left(\mathbf{r}_{p,i}\right) - \mathbf{r}_o. \tag{28}$$

O algoritmo minimiza a soma destes erros ao quadrado:

$$\sum_{i=1}^{Npo} \left\|\mathbf{e}_i\right\|^2. \tag{29}$$

Notando-se que:

$$\sum_{i=1}^{Npo} \mathbf{r}'_{m,i} = 0, \qquad \sum_{i=1}^{Npo} \mathbf{r}'_{p,i} = 0, \tag{30}$$

reescreve-se o erro residual na forma:

$$\mathbf{e}_i = \mathbf{r}'_{m,i} - s\mathbf{R}\left(\mathbf{r}'_{p,i}\right) - \mathbf{r}'_o, \tag{31}$$

onde:

$$\mathbf{r}'_o = \mathbf{r}_o - \bar{\mathbf{r}}_m + s\mathbf{R}\left(\bar{\mathbf{r}}_p\right). \tag{32}$$

A somatória dos erros ao quadrado se torna:

$$\sum_{i=1}^{Npo} \left\| \mathbf{r}'_{m,i} - s\mathbf{R}\left(\mathbf{r}'_{p,i}\right) - \mathbf{r}'_o \right\|^2 \tag{33}$$

ou

$$\sum_{i=1}^{Npo} \left\| \mathbf{r}'_{m,i} - s\mathbf{R}\left(\mathbf{r}'_{p,i}\right) \right\|^2 - 2\mathbf{r}'_o \cdot \sum_{i=1}^{Npo} \left\| \mathbf{r}'_{m,i} - s\mathbf{R}\left(\mathbf{r}'_{p,i}\right) \right\| + Npo \left\| \mathbf{r}'_o \right\|^2 . \tag{34}$$

Agora a somatória do termo do meio é zero, uma vez que as distâncias são medidas com relação ao centro das amostras. O primeiro termo independe de $\mathbf{r}'_o$ e o terceiro termo é sempre maior ou igual a zero. O erro total é obviamente minimizado com $\mathbf{r}'_o = 0$ ou:

$$\mathbf{r}_o = \bar{\mathbf{r}}_m - s\mathbf{R}\left(\bar{\mathbf{r}}_p\right), \tag{35}$$

onde $\mathbf{R}\left(\bar{\mathbf{r}}_p\right)$ é a média dos pontos casados da amostra rotacionados, ou seja, a translação é apenas a diferença entre o centróide do mapa e a centróide rotacionada da amostra vezes o fator de escala.

2.1.1.3 Encontrando o fator de escala

A este ponto notamos que o termo do erro pode ser rescrito na forma:

$$\mathbf{e}_i = \mathbf{r}'_{m,i} - s\mathbf{R}\left(\mathbf{r}'_{p,i}\right), \tag{36}$$

pois $r'_o = 0$. Então o erro total a ser minimizado é:

$$\sum_{i=1}^{Npo} \left\| \mathbf{r}'_{m,i} - s\mathbf{R}\left(\mathbf{r}'_{p,i}\right) \right\|^2 . \tag{37}$$

Expandindo o erro total:

$$\sum_{i=1}^{Npo}\left\|\mathbf{r}_{m,i}^{'}\right\|^2 - 2s\sum_{i=1}^{Npo}\mathbf{r}_{m,i}^{'}\mathbf{R}\left(\mathbf{r}_{p,i}^{'}\right) + s^2\sum_{i=1}^{Npo}\left\|\mathbf{r}_{p,i}^{'}\right\|^2, \tag{38}$$

que pode ser rescrito na forma:

$$S_m - 2sD + s^2 S_p, \tag{39}$$

onde S_m e S_p são somas dos comprimentos dos pontos referente ao centróide ao quadrado, enquanto D é o produto escalar das coordenadas referentes ao conjunto do mapa com as coordenadas rotacionadas da amostra. Completando o quadrado em s, têm-se:

$$\left(s\sqrt{S_p} - D/\sqrt{S_p}\right)^2 + \frac{\left(S_m S_p - D^2\right)}{S_p}. \tag{40}$$

Minimizando com respeito a s, onde o primeiro termo é zero, $s = D/S_p$, têm-se:

$$s = \frac{\sum_{i=1}^{n}\mathbf{r}_{m,i}^{'}.\mathbf{R}\left(\mathbf{r}_{p,i}^{'}\right)}{\sum_{i=1}^{n}\left\|\mathbf{r}_{p,i}^{'}\right\|^2}, \tag{41}$$

onde $\mathbf{R}\left(\mathbf{r}_{p,i}^{'}\right)$ indica os pontos da amostra relativos ao centro dos pontos casados da amostra rotacionada.

2.2 Etapa de correspondência

Tendo em vista as três etapas para a construção de mapas de ambiente, a parte mais importante, correspondência dos pontos das duas imagens fotografadas, foi implementada e

testada. O TrICP, desenvolvido por Chetverikov et al. (2002a) foi o método escolhido. O algoritmo é composto por um processo iterativo de seis passos:

1. Para cada ponto da nova imagem calculam-se as distâncias ao quadrado dos pontos da imagem anterior, d_i^2;
2. Ordena-se as distâncias ao quadrado (d_i^2) em ordem crescente e seleciona-se os pontos correspondentes. O número de pontos correspondidos na nova imagem, Npo, é dado por: $Npo = Np.\lambda$, sendo λ o grau de encobrimento da nova imagem e Np o número de pontos da mesma;
3. Calcula-se a soma das distâncias ao quadrado S'_{LTS} dos pontos correspondentes das duas imagens;
4. Verifica a condição de parada do processo iterativo. Se a condição de parada for satisfeita, sair; se não, faz-se $S_{LTS} = S'_{LTS}$ e executa o passo 5;
5. Calcula-se para os pares selecionados das possíveis correspondências Npo, o movimento ótimo, ou seja, rotação e translação (R, t) que minimiza S'_{LTS};
6. Aplica a transformação na imagem adquirida de acordo com (R, t) e volta para o passo 1.

No passo 1, dado um ponto no espaço da nova imagem, usa-se uma região em torno do ponto considerado, de modo que apenas os pontos presentes dentro deste bloco na imagem anterior sejam considerados durante a busca. O tamanho do bloco é atualizado com o andar da iterações. Uma ordenação comum é usada para ordenar as distâncias no passo 2. As condições de parada no passo 4 são: número máximo de iterações atingido, ou o valor do erro quadrático médio (EQM), definido por S'_{LTS}/Npo for suficientemente pequeno, ou a mudança de uma iteração para a outra do EQM for positiva. A transformação de rotação calculada no passo 5 é descrita por meio de arco-seno e arco-coseno.

Uma sala virtual é criada usando-se o programa POV RAY, para testar o algoritmo de correspondência.

2.2.1 Programação dinâmica

Após os testes do TrICP no ambiente simulado, mostrados na seção 3.2, decidiu-se fazer uma modificação no algoritmo, incorporando um algoritmo de programação dinâmica na correspondência dos pontos, definido pelos passos 1 e 2 do algoritmo original do TrICP.

Programação dinâmica é um método de organizar um problema de otimização de forma a explorar a estrutura recursiva dos cálculos necessários. Muitos trabalhos têm usado esta abordagem de correspondência para correspondência estéreo. O algoritmo de programação dinâmica, descrito a seguir, foi retirado dos trabalhos de Fielding e Kam (2000).

O principal de se usar programação dinâmica é a restrição de ordenação monotônica que estabelece que em uma linha epipolar, se um determinado ponto na imagem A de coordenada x_{ri} corresponde um determinado ponto na imagem B de coordenada x_{lj}, então x_{ri+1} só pode corresponder x_{lj+k} se $k \geq 1$. Sem a restrição de ordenação monotônica, é necessária que seja feita uma busca exaustiva.

A figura 2 mostra um gráfico no qual o caminho de menor custo entre os vértices S e T pode ser calculado usando programação dinâmica. As linhas pontilhadas permitem que ocorra oclusão, seus custos são dados pelo custo de oclusão *Co*. Associa-se o peso da matriz de custo epipolar, $\mathbf{c}(x_l, x_r)$, com as linhas diagonais sólidas.

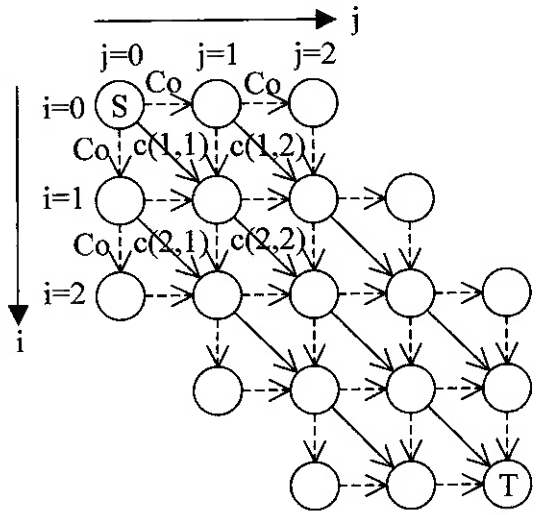

Figura 2 - Representação gráfica do problema de programação dinâmica.

Cada vértice é associado com o seu antecessor usando uma matriz $\mathbf{P}(i, j)$, permitindo que se possa rastrear, do vértice $\mathbf{T}(N,N)$, quais nós estão no caminho de menor custo. Uma

linha horizontal ou vertical corresponde a uma oclusão no mapa de disparidade *A* e uma linha vertical representa uma correspondência no mapa de disparidade.

Uma vez que a programação dinâmica esta efetivamente procurando por um caminho de menor custo através do gráfico, qualquer correspondência feita na linha epipolar afeta correspondências subseqüentes. Um fator de importância crítica na programação dinâmica na solução de problemas de correspondência estéreo é a escolha do custo de oclusão, *Co*. Definindo *Co* com um valor muito grande resulta em ausência de oclusões e definindo com um valor muito pequeno resulta na seleção de oclusões apenas.

A restrição de ordenação monotônica efetivamente mantém as disparidades locais perto umas das outras sem usar uma regularização da estrutura ou uma técnica de relaxação. Uma desvantagem da restrição de ordenação monotônica é que ela não é satisfeita onde os objetos oclusos são estreitos. O uso da programação dinâmica nestas condições normalmente resulta em um mapa de disparidade que ou detecta parcialmente ou ignora completamente objetos estreitos.

O algoritmo usado é mostrado a baixo.

```
for i = 0 to N do P(i,0) = 2; V(i,0) = i*Co; //inicializa a primeira coluna de vértices.
for j = 0 to N do P(0,j) = 3; V(0,j) = j*Co; //inicializa a primeira linha de vértices.
for i = 1 to N do
    for j = 1 to N do
        v1 = V(i-1,j-1) + c(i,j); //custo da linha de correspondência
        v2 = V(i-1,j) + Co; //oclusão na linha vertical
        v3 = V(i,j-1) + Co; //oclusão na linha horizontal
        V(i,j) = min(v1,v2,v3); //escolhe o caminho de menor custo
        P(i,j) = argmin(v1,v2,v3); //registra o antecessor (1, 2 ou 3)
i = N; j = N;
while (j > 0) //rastreia a partir do vértice T
    if P(i,j = = 1) then RightMate[j] = i; i = i-1; j = j -1;
    else if P(i,j) = = 2 do i = i -1;
    else RightMate[j] = UNMATCHED; j = j -1;
```

2.2.2 Algoritmo do Trimmed Interative Closest Point modificado

Após os teste do TrICP com programação dinâmica, mostrados no próximo capítulo, foi necessária a continuidade dos trabalhos com o TrICP convencional. Para a integração deste algoritmo junto ao de construção de mapa duas mudanças importantes foram realizadas em relação ao algoritmo inicial. A primeira mudança é que o número de iteração máximo é definido pelo tamanho do maior conjunto de pontos, do mapa ou da amostra. A segunda mudança, e também a mais importante, é que agora o algoritmo não necessariamente considera a convergência monotônica, sendo guardado um registro do mínimo global durante as iterações até que algum dos critérios de parada seja atingido. Quando isso ocorre, os dois erros quadráticos médios são comparados, o da amostra mínima global e a da amostra que atingiu o critério de parada. A amostra considerada como saída do algoritmo é a do mínimo global se o erro quadrático médio da amostra do critério de parada for a partir de 10% superior ao da amostra do mínimo global, caso contrário a saída será a amostra do critério de parada.

Foi realizado um estudo mais detalhado deste algoritmo com o conjunto de dez imagens omnidirecionais mostradas no Anexo B. O objetivo desta análise foi mostrar a utilização do mínimo global e também estudar o comportamento do algoritmo quando ocorre inversão na ordem das amostras, ou seja, ao invés de tentar levar a amostra ao mapa, levar o mapa à amostra. Outro fator estudado foi a seleção de pontos mais próximos ao centro da amostra baseada do incremento angular de 1°. Os resultados encontrados são apresentados no próximo capítulo.

2.3 Etapa de integração

Quando um processo de correspondência é bem sucedido, uma fusão dos pontos próximos entre si é realizada com o objetivo de diminuir o número de pontos no mapa final.

A fusão se dá da seguinte forma, para cada ponto da amostra transformada pelo TrICP é procurado no mapa os pontos que se enquadram dentro do quadrado de erro considerado e um único ponto médio substitui os pontos encontrados. Após este processo de fusão ocorre a integração final entre o mapa e a amostra gerando um único conjunto de pontos, sendo agora o mapa global do ambiente.

2.4. Integração das três etapas do algoritmo

Um primeiro algoritmo foi desenvolvido e testado, chegando a resultados de qualidade média. Neste algoritmo seis tentativas de correspondência são realizadas.

Posteriormente, aproveitando a experiência ganha com o primeiro algoritmo, um segundo algoritmo foi desenvolvido, onde se obteve resultados de alta qualidade. Neste algoritmo dez tentativas de correspondência são realizadas.

Segue agora a primeira implementação da integração dos pontos alinhados e correspondidos nas etapas anteriores. Para visualizar a lógica utilizada no algoritmo de construção de mapa de ambiente é apresentado um diagrama lógico com todos os passos principais do mesmo na figura 3.

Inicialmente uma imagem inicial é adquirida pelo sistema de visão omnidirecional, a imagem é processada e sua reconstrução tridimensional é realizada, sendo a partir de então considerado o mapa inicial para o algoritmo de construção de mapa de ambiente. A partir desta primeira etapa o algoritmo entra em um ciclo até que não haja mais nenhuma imagem para ser integrada ao mapa global.

A seguir é descrito o ciclo principal do algoritmo de construção de mapa de ambiente. Uma nova imagem é adquirida e será considerada uma amostra do ambiente em relação ao mapa global. A imagem é processada e reconstruída tridimensionalmente utilizando o algoritmo desenvolvido por Souza Jr. (2004). De modo a tentar agilizar o processo de correspondência para o TrICP, tanto o mapa quando a amostra são filtrados de modo a ser considerado apenas o ponto mais próximos ao centro da imagem por incremento angular de 1°, desta forma cada mapa filtrado e amostra filtrada tem no máximo 360 pontos. Em seguida seis tentativas de correspondência são feitas. O algoritmo do TrICP leva a amostra em direção ao mapa, sendo considerado aqui que o primeiro conjunto de pontos citado corresponde ao mapa para o TrICP e o segundo conjunto a amostra, independente do algoritmo de construção de mapa. As tentativas são as seguintes: mapa filtrado - amostra filtrada, amostra filtrada - mapa filtrado, mapa filtrado - amostra, amostra - mapa filtrado, mapa - amostra, amostra - mapa.

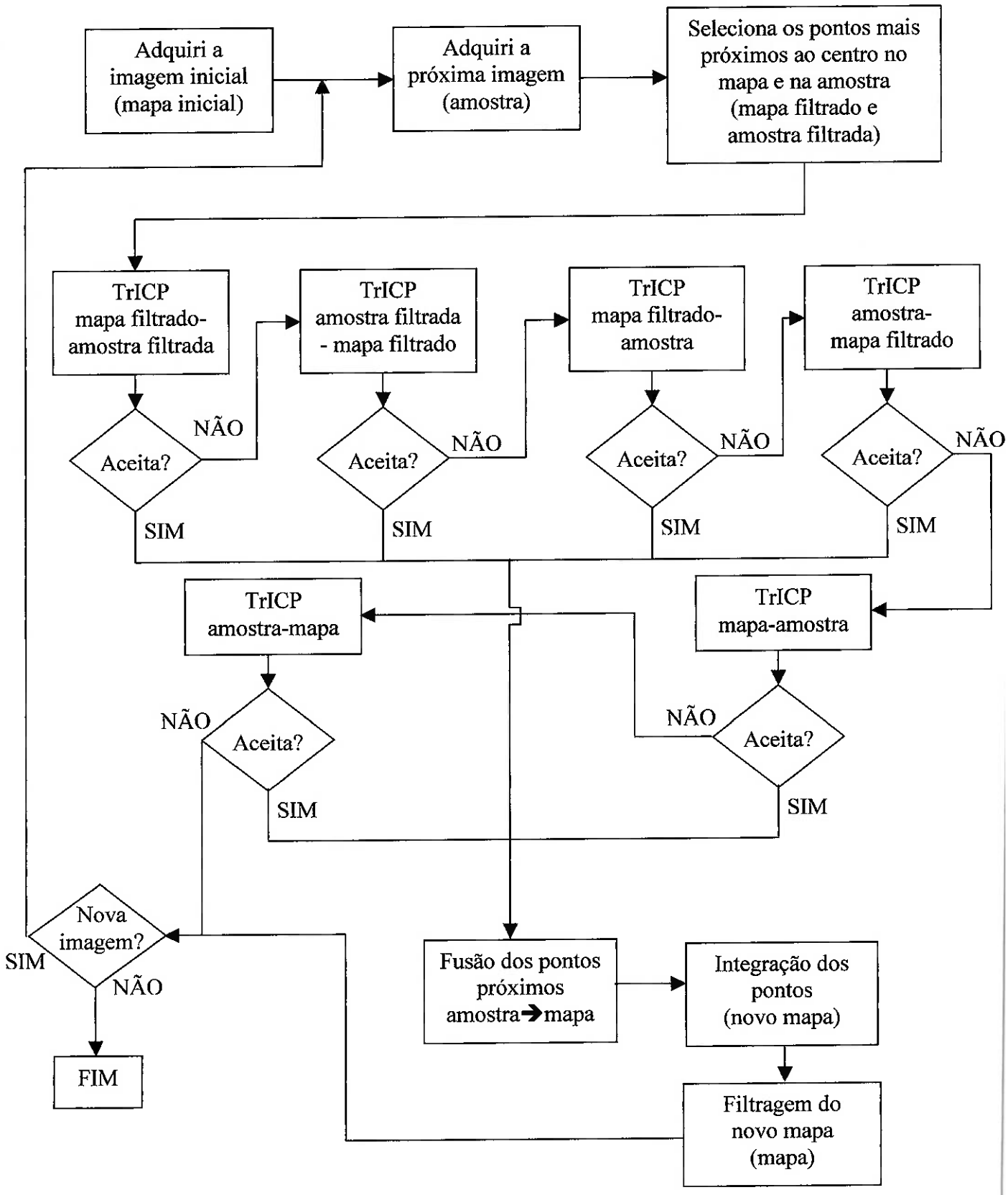

Figura 3 - Diagrama lógico do primeiro algoritmo de construção de mapa de ambiente

Para determinar se o processo de correspondência foi bem sucedido, duas condições são checadas, a mudança relativa do erro quadrático médio inicial e final da amostra e o ângulo de rotação final, de modo que para ser aceita o valor do erro quadrático médio relativo deve ser superior a 50% e o ângulo total da rotação não pode ser superior a 5°, tendo em vista que o robô desenvolvido para este projeto tem a capacidade de girar as rodas em torno do próprio eixo para realizar mudanças de direção, mantendo sempre a frente alinhada.

Se o processo de correspondência falhar em todas as tentativas a imagem é descartada e se houver mais alguma imagem o ciclo se reinicia, se não, o algoritmo termina. Se algum dos processos for bem sucedido, uma fusão dos pontos próximos entre si é realizada com o objetivo de diminuir o número de pontos no mapa final.

A fusão se dá da seguinte forma, para cada ponto da amostra transformada pelo TrICP é procurado no mapa os pontos que se enquadram dentro do quadrado de erro considerado e um único ponto médio substitui os pontos encontrados. Após este processo de fusão ocorre a integração final entre o mapa e a amostra gerando um único conjunto de pontos, sendo agora o mapa global do ambiente. Uma filtragem é realizada com o objetivo de eliminar pontos espúrios e erros da reconstrução tridimensional. Se houver mais alguma imagem disponível para a integração o processo se reinicia, se não o algoritmo termina.

Após os testes no ambiente simulado usando o algoritmo de mapa de ambiente descrito acima, um novo algoritmo foi desenvolvido. O diagrama lógico deste novo algoritmo é mostrado na figura 4.

A diferença principal entre o primeiro e o segundo algoritmo é que o último mantém um registro das últimas três imagens adquiridas pelo sistema de visão estéreo omnidirecional com o objetivo de expandir as possibilidades de correspondência entre as imagens. Inicialmente uma imagem inicial é adquirida pelo sistema de visão omnidirecional, a imagem é processada e sua reconstrução tridimensional é realizada, sendo a partir de então considerado o mapa inicial para o algoritmo de construção de mapa de ambiente, também esta imagem fica registrada como última imagem adquirida. De modo a tentar agilizar o processo de correspondência para o TrICP, tanto o mapa é filtrado de modo a ser considerado apenas o ponto mais próximos ao centro da imagem por incremento angular de 1°, desta forma o mapa filtrado tem no máximo 360 pontos. A partir desta primeira etapa o algoritmo entra em um ciclo até que não haja mais nenhuma imagem para ser integrada ao mapa global.

A seguir é descrito o ciclo principal do segundo algoritmo de construção de mapa de ambiente.

O primeiro passo é organizar a ordem das imagens adquiridas anteriormente, de modo a descartar a amostra mais antiga e deixar um espaço livre para a que a próxima imagem a ser adquirida não ocupe o lugar de uma imagem mais antiga. As três amostras são nomeadas segundo a ordem de aquisição sendo esta última, penúltima e antepenúltima imagem a ser adquirida, deste modo, a penúltima imagem se torna a antepenúltima e a última se torna e penúltima. Nota-se que não somente a amostra original é armazenada, mas também sua versão de pontos filtrados segundo o incremento angular como descrito acima, de modo que cada amostra é filtrada apenas uma vez durante todo o processo, economizando tempo computacional. Uma nova imagem é adquirida e será considerada a última amostra do ambiente em relação ao mapa global. A imagem é processada e reconstruída tridimensionalmente e filtrada segundo o incremento angular tendo no máximo 360 pontos. O mapa global é filtrado segundo o incremento angular. Em seguida dez tentativas de correspondência são feitas. O algoritmo do TrICP leva a amostra em direção ao mapa, sendo considerado aqui que o primeiro conjunto de pontos citado corresponde ao mapa para o TrICP e o segundo conjunto a amostra, independente do algoritmo de construção de mapa. As tentativas são as seguintes: mapa filtrado - última amostra filtrada, última amostra filtrada - mapa filtrado, mapa - última amostra, última amostra - mapa, penúltima amostra filtrada - última amostra filtrada, última amostra filtrada - penúltima amostra filtrada, penúltima amostra - última amostra, última amostra - penúltima amostra, antepenúltima amostra - última amostra, última amostra - antepenúltima amostra. Para determinar se o processo de correspondência foi bem sucedido, duas condições são checadas, a mudança relativa do erro quadrático médio inicial e final da amostra e o ângulo de rotação final, de modo que para ser aceita o valor do erro quadrático médio relativo deve ser superior a 50% e o ângulo total da rotação não pode ser superior a 5°.

Se o processo de correspondência falhar em todas as tentativas a imagem não é integrada ao mapa e se houver mais alguma imagem o ciclo se reinicia, se não, o algoritmo termina. Se algum dos processos for bem sucedido, uma fusão dos pontos próximos entre si é realizada com o objetivo de diminuir o número de pontos no mapa final.

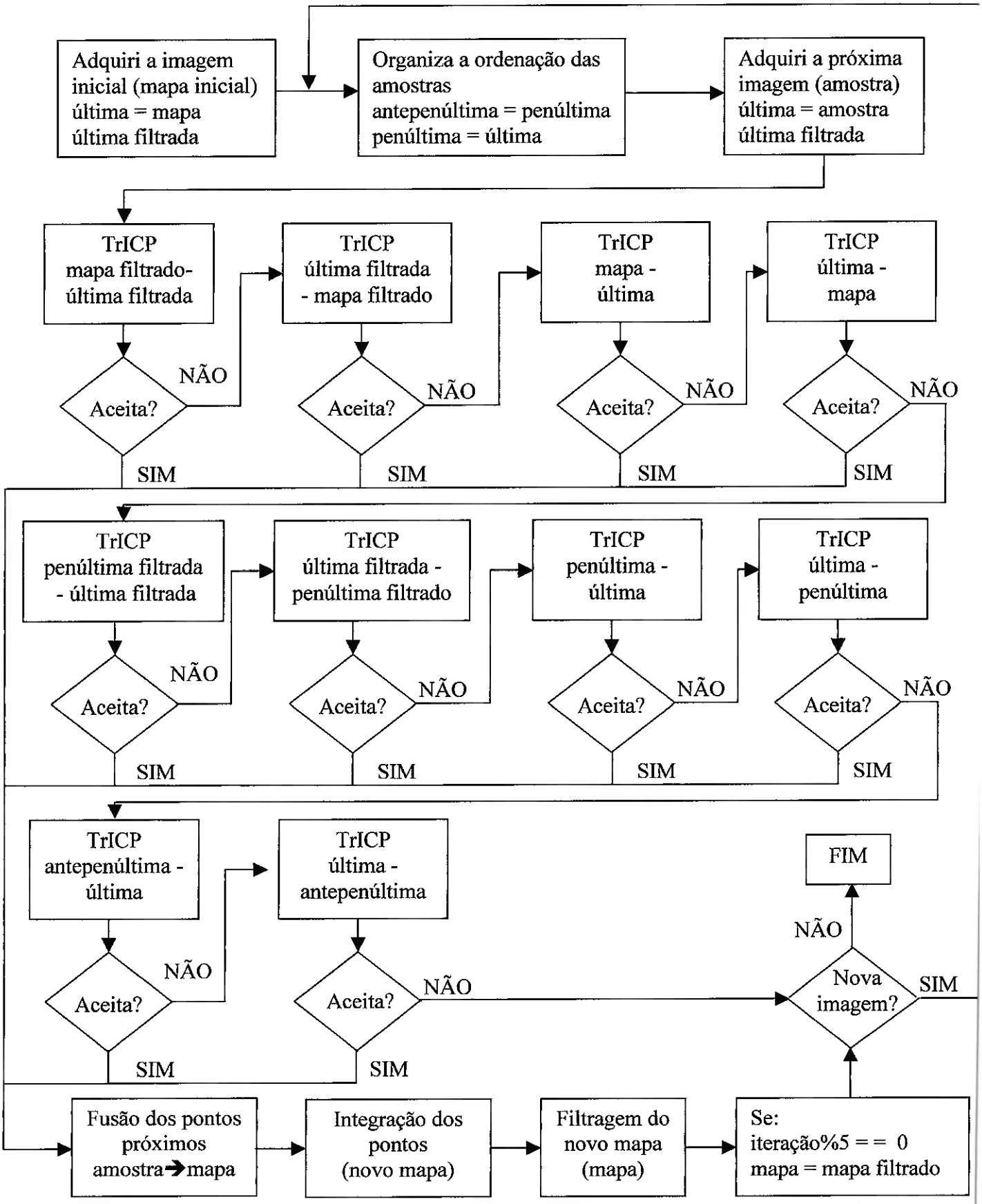

Figura 4 - Diagrama lógico do segundo algoritmo de construção de mapa de ambiente.

. A fusão se dá da seguinte forma, para cada ponto da amostra transformada pelo TrICP é procurado no mapa os pontos que se enquadram dentro do quadrado de erro considerado e um único ponto médio substitui os pontos encontrados. Após este processo de fusão ocorre a integração final entre o mapa e a amostra gerando um único conjunto de pontos, sendo agora o mapa global do ambiente. Uma filtragem é realizada com o objetivo de eliminar pontos espúrios e erros da reconstrução tridimensional. A cada iteração a partir do mapa global é gerado um mapa global apenas com pontos que tenham tido certo numero de correspondências.

A cada cinco iterações a mapa global é substituído por este mapa filtrado de modo a eliminar pontos desnecessários para as futuras iterações. Se houver mais alguma imagem disponível para a integração o processo se reinicia, se não o algoritmo termina.

3. Testes do sistema

3.1 Teste preliminar dos algoritmos TrICP e ICP

Inicialmente, foi feito um programa no ambiente de programação do MatLab que gera randomicamente um conjunto de pontos no espaço tridimensional. A partir destes pontos gerados, o programa gera um subespaço, com uma dada rotação em torno do eixo z, translação nos eixos x e y e um certo grau de encobrimento (λ). O código do programa esta incluso no Anexo A.

Vários testes foram feitos com o TrICP e ICP convencional, variando se: a rotação, as translações, o número de pontos gerados e o grau de recobrimento do subespaço. Os resultados encontram-se na tabela 2. Nas colunas dos métodos mostram se os números de iterações médias para 5 testes nas condições iniciais da linha. Observa-se que para todos os testes em que houve convergência, os resultados obtidos para a rotação e translação foram satisfatórios.

Observa-se que o ICP convencional funciona bem, necessitando, contudo, um número de iterações maior do que o do TrICP, para diferentes graus de dificuldade, desde que não haja encobrimento entre o espaço original e o subespaço. Esse fato já era de se esperar, entretanto, no último teste (teste 16) ele não foi capaz de reconstruir a translação e rotação.

Tabela 2 - Teste de desempenho comparativo entre o TrICP e o ICP

Teste	TrICP	ICP	rotação (°)	x (mm)	y (mm)	nº pontos	λ
1	5,8	11,6	10	0	0	30	1
2	5,0	10,6	0	100	100	30	1
3	5,6	12,2	10	100	100	30	1
4	5,4	-	10	0	0	30	0,7
5	4.6	-	0	100	100	30	0,7
6	5,6	-	10	100	100	30	0,7
7	10,2	11,8	10	1000	500	40	1
8	10,4	-	10	1000	500	40	0,7
9	-	-	20	500	500	40	0,7
10	-	-	20	500	500	50	0,7
11	-	-	20	500	500	60	0,7
12	-	-	20	500	500	70	0,7
13	-	-	20	500	500	80	0,7
14	-	-	20	500	500	90	0,7
15	-	-	20	500	500	100	0,7
16	-	-	20	500	500	100	1

O TrICP se mostrou bastante robusto até o teste 8, mostrando eficiência maior que o próprio ICP com relação ao número de iterações nos casos em que o ICP funciona. A partir do teste 9, o TrICP não foi mas capaz de reconstruir a rotação e a translaçãos.

Quando há convergência, a diminuição do valor da soma das distâncias ao quadrado é monotônico. Quando há flutuações neste valor o algoritmo não é capaz de convergir para um mínimo e fica flutuando em torno de um valor.

A figura 5 mostra um conjunto de pontos gerados no programa e posteriormente processados pelo método TrICP para o teste 9. Os centros dos círculos azuis representam os pontos gerados inicialmente. Os pontos pretos representam o subespaço gerado a partir do espaço inicial, transladados e rotacionados. Os pontos vermelhos representam os pontos excluídos do espaço inicial, também transladados e rotacionados.

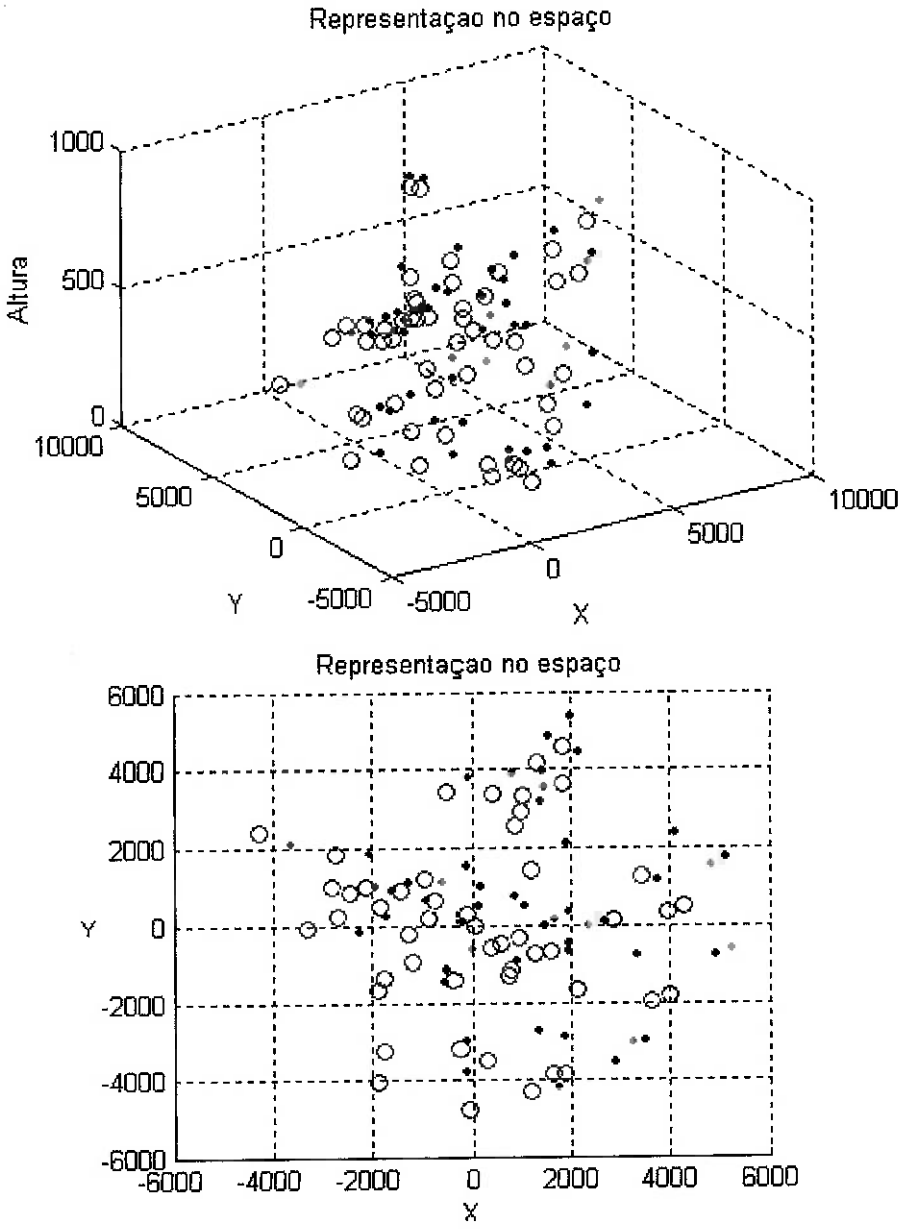

Figura 5 - Pontos gerados pelo programa gerador de pontos.

Na figura 6 vê se o resultado após a passagem pelo TrICP.

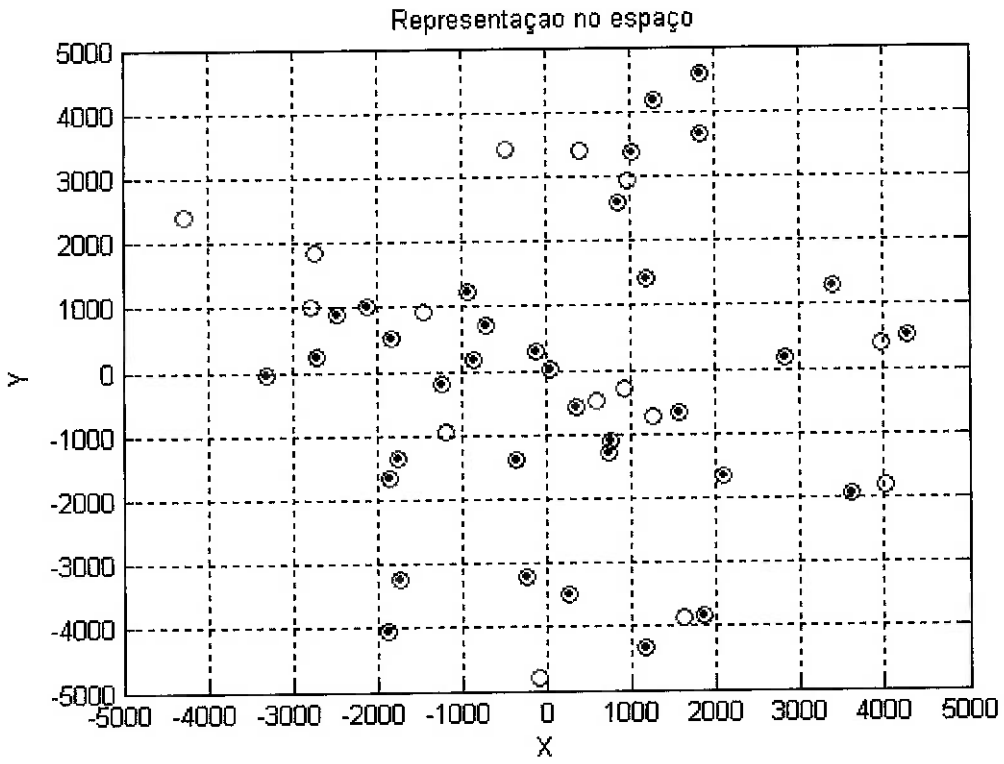

Figura 6 - Reconstrução da translação e rotação pelo TrICP.

3.2 Teste do sistema em um ambiente virtual

Terminados os testes iniciais, um ambiente virtual foi criado no programa PovRay©. Basicamente o ambiente é composto por dois armários idênticos de dois metros de comprimento por meio metro de profundidade e altura de um metro e meio, uma mesa com base cilíndrica de um metro de comprimento por oitenta centímetros de largura e cinco centímetros de espessura, uma janela, uma porta, um tablado de dois metros e meio por um metro e oitenta de largura e vinte centímetros de altura, um quadro de dois metros e meio de comprimento por um metro de altura com cinco centímetros de profundidade, uma esfera com vinte cinco centímetros de raio e quatro caixas idênticas de quarenta centímetros por quarenta centímetros com vinte cinco centímetros de altura afastadas simetricamente um metro do centro de coordenadas da sala. A sala tem três metros de altura, sendo a altura zero correspondente a altura da imagem gerada na câmera. Uma vista superior deste ambiente simulado pode ser vista na figura 7.

altura 3m, [-700,2700]

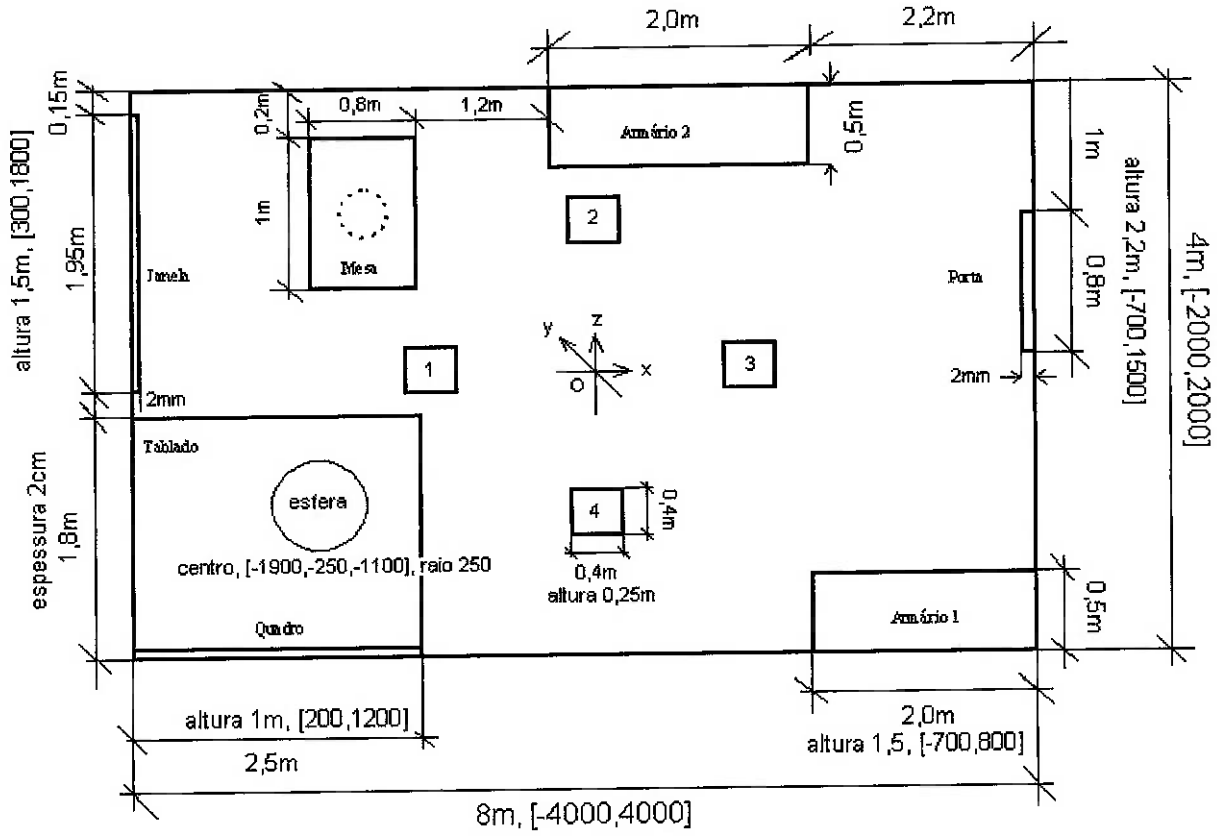

Figura 7 - Vista superior do ambiente simulado.

De modo a testar o algoritmo TrICP uma trajetória foi criada na sala, com pontos de tomada de imagens pelo sistema de visão omnidirecional. A figura 8 mostra esta trajetória, onde cada ponto vermelho marca a tomada das imagens. A tabela 3 mostra as coordenadas dos pontos no sistema de coordenadas do ambiente. As imagens estão inclusas no Anexo B.

Tabela 3 - Coordenadas da tomada de pontos da trajetória do robôs no ambiente simulado

Ponto	Coord. X	Coord. Z	Ponto	Coord. X	Coord. Z
1	3000	1500	6	0	0
2	3000	500	7	-1000	500
3	3000	-500	8	-2000	500
4	2000	-500	9	-3000	500
5	1000	-500	10	-3000	1500

altura 3m, [-700,2700]

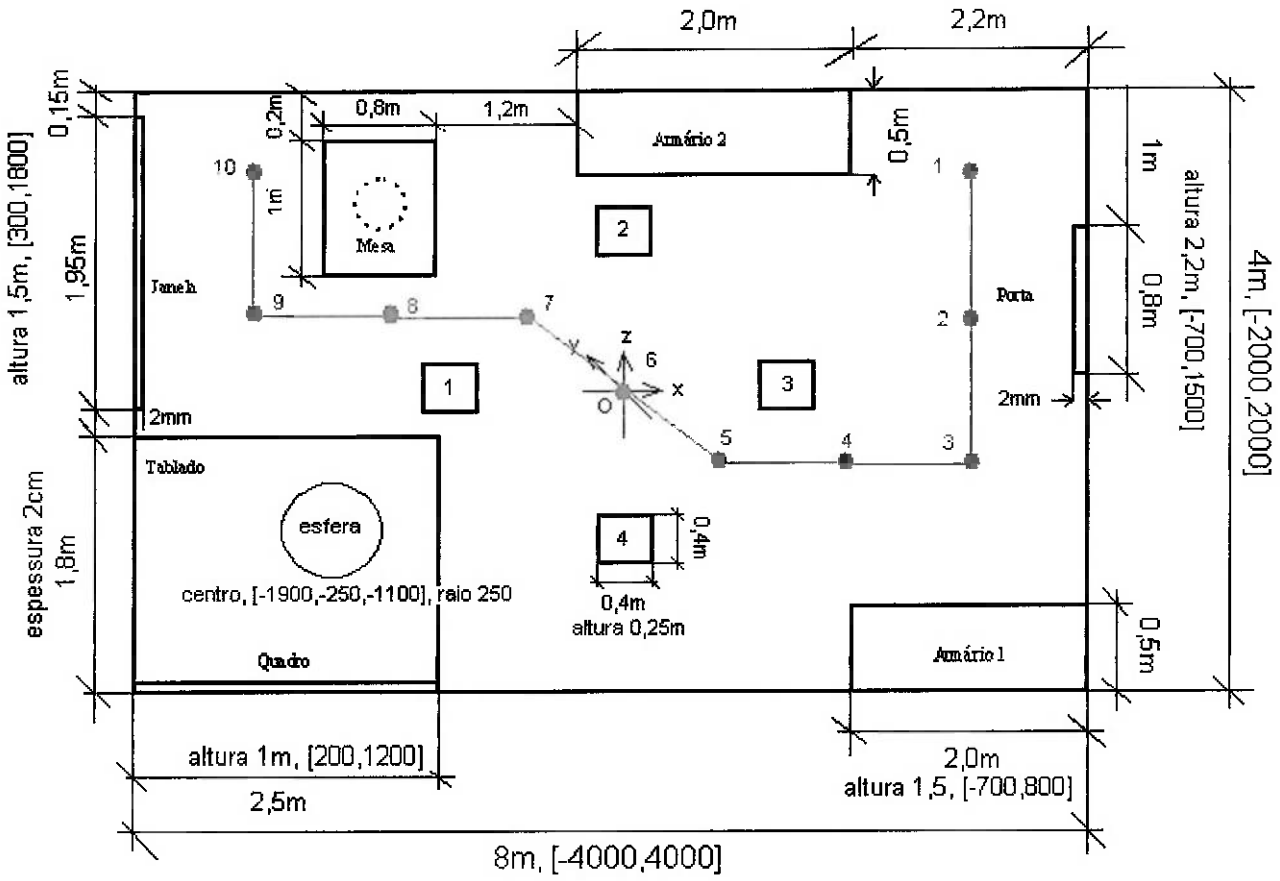

Figura 8 - Trajetória do robô pelo ambiente simulado com as posições das tomadas de imagem.

O programa que faz a reconstrução tridimensional das imagens obtidas no ambiente simulado foi desenvolvido por Souza Jr. (2004) em sua tese de doutorado, especificamente para o conjunto espelho-câmera usado neste trabalho. Entretanto os programas eram específicos para cada passo da reconstrução tridimensional, tendo sido neste trabalho simplificados, na medida do possível, e também unificados em um único programa. Este programa realiza, a partir de uma única imagem obtida do espelho duplo de perfil hiperbólico, a correspondência entre os pontos mais próximos do robô para cada intervalo de 1° nas imagens interna e externa do espelho e sua triangularização no espaço. Esse processo gera o conjuntos de pontos usado no TrICP de modo a montar o mapa do ambiente a partir das consecutivas imagens obtidas. Cabral et al.(2004) demonstra as equações usadas na triangularização para o espelho duplo de perfil hiperbólico usado neste trabalho.

A figura 9 mostra a primeira imagem obtida para a trajetória adotada anteriormente.

Figura 9 - Imagem do ponto 1 da trajetória vermelha.

A figura 10 mostra a imagem da figura 9 após a reconstrução tridimensional. O sistema de coordenadas da figura é o do robô.

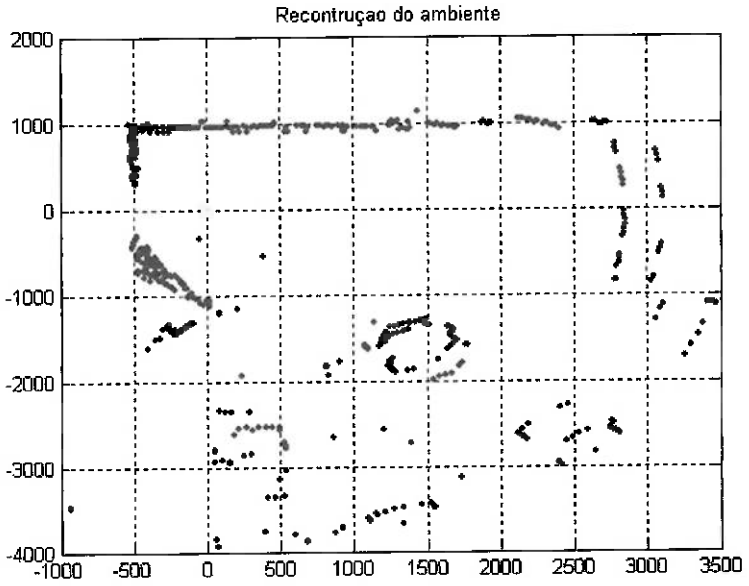

Figura 10 - Reconstrução do ambiente a partir do primeiro ponto da trajetória.

A figura 11 mostra a segunda imagem obtida para a trajetória indicada anteriormente.

Figura 11 - Imagem do ponto 2 da trajetória vermelha.

A figura 12 mostra a imagem da figura 11 após a reconstrução tridimensional. O sistema de coordenadas da figura é o do robô.

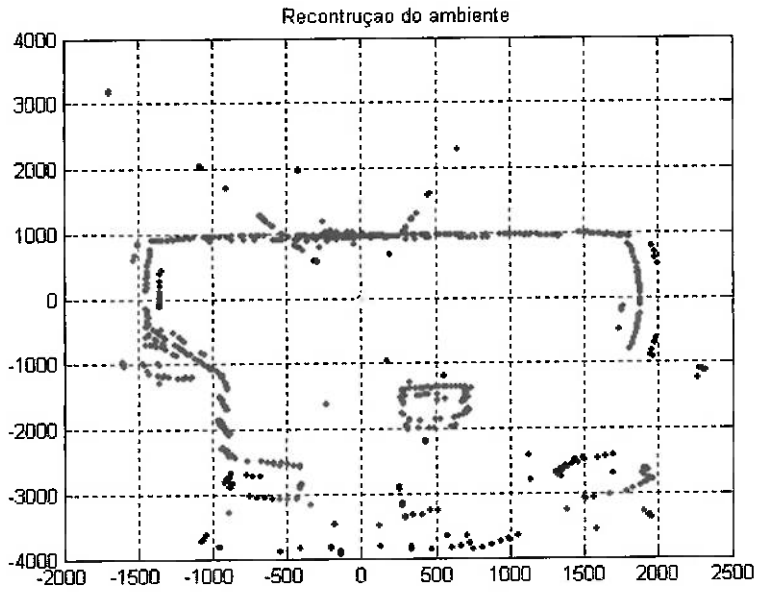

Figura 12 - Reconstrução do ambiente a partir do ponto 2 da trajetória vermelha.

A figura 13 mostra a fusão dos resultado das figuras 10 e 12 para a construção de um mapa de ambiente realizada pelo método TrICP.

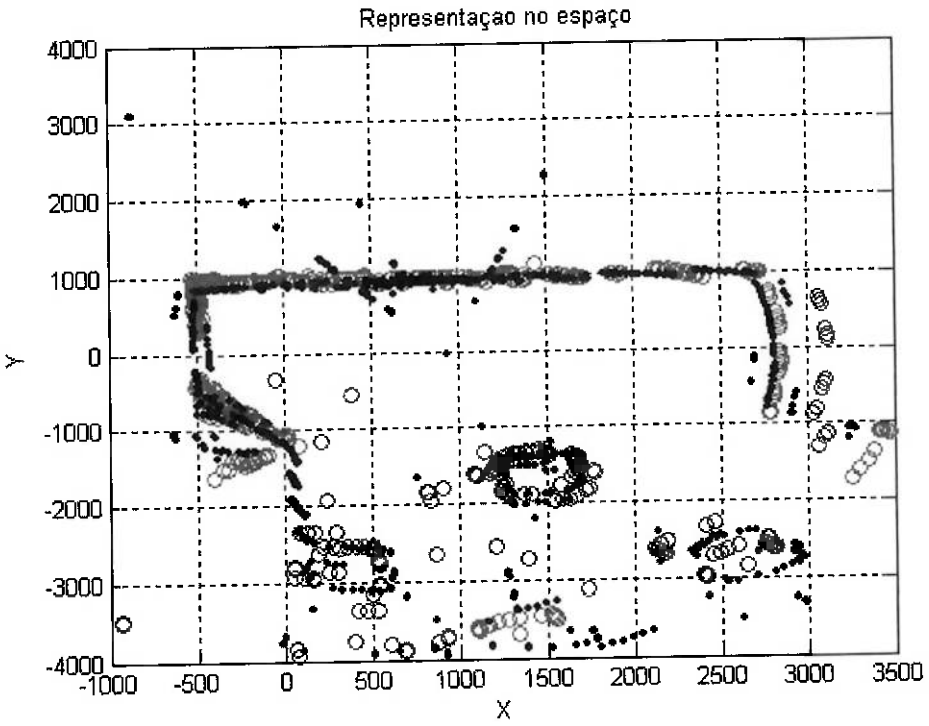

Figura 13 - Imagem dos primeiros e segundos pontos da trajetória reconstruídos e processados pelo TrICP.

Os centros dos círculos azuis mostram os pontos da imagem 1 da trajetória. Os pontos pretos mostram os pontos da imagem 2 reconstruída em relação à imagem 1. Os resultados esperados eram de translação em Z de 1000 mm (mil milímetros), translação em X de 0 mm (zero milímetros) e rotação de zero graus. Os resultados obtidos foram de translação de 927,87 mm em Z, -33,35 mm em X e uma rotação de 1,86° em 11 iteraçoes. Considerando os resultados, o erro foi de 7,15%.

Com o objetivo de tentar obter-se uma correspondência de melhor qualidade, um algoritmo de programação dinâmica foi acrescentado ao TrICP nos passos 1 e 2 do algoritmo original da correspondência dos pontos. O algoritmo é explicado em detalhes no capítulo anterior. Para simplificar a nomenclatura, chama-se o algoritmo de DPICP (Dynamic Programming Iterative Closest Point). Inicialmente, foram feitos testes com o gerador de pontos aleatórios.

A Condição inicial do teste 1 é mostrada na figura 14, sendo de 20 graus de rotação, 1000 mm em x, 1000 mm em y, sem recobrimento com relação ao conjunto de pontos original e 1000 pontos.

O resultado pode ser visto na figura 15, tendo sido necessárias 6 iterações. O erro quadrático total foi de 2.2172e-024.

Os círculos azuis correspondem ao conjunto de pontos iniciais e os pontos pretos correspondem aos pontos do subespaço gerado que sempre tiveram correspondência durante todo o processo. Os pontos vermelhos representam os pontos que durante alguma iteração não tiveram correspondência e foram excluídos das futuras iterações.

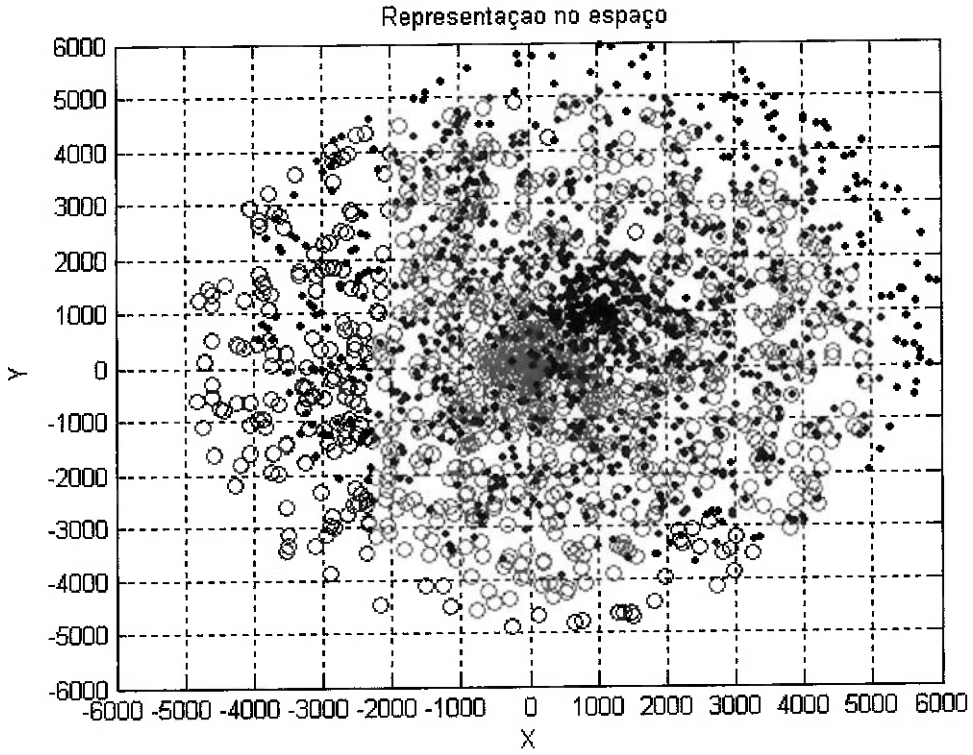

Figura 14 - Condição inicial do teste 1 para passar no DPICP

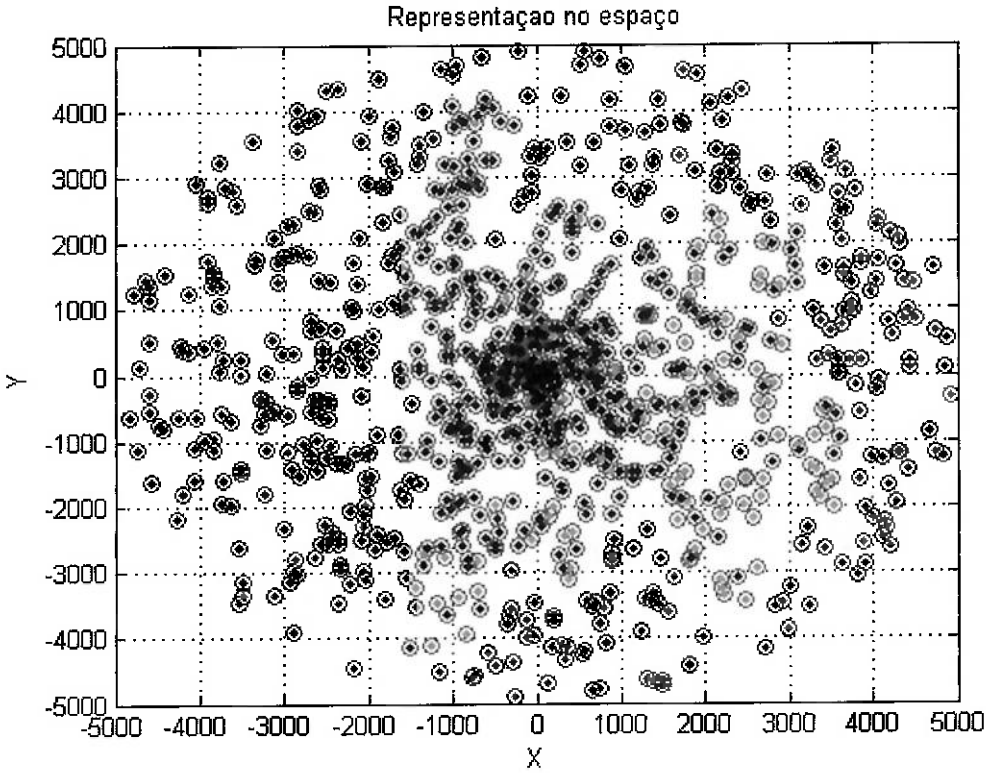

Figura 15 - Teste 1 após o processamento pelo DPICP.

A Condição inicial do teste 2 é mostrada na figura 16, sendo de 20 graus de rotação, 1000 mm em *x*, 1000 mm em *y*, recobrimento de 70% com relação ao conjunto de pontos original e 1000 pontos. Os pontos vermelhos representam os pontos encobertos.

O resultado pode ser visto na figura 17, tendo sido necessárias 6 iterações. O erro quadrático total foi de 1.7753e-024.

Os círculos azuis correspondem ao conjunto de pontos iniciais e os pontos pretos correspondem aos pontos do subespaço gerado que sempre tiveram correspondência durante todo o processo. Os pontos vermelhos representam os pontos que durante alguma iteração não tiveram correspondência e foram excluídos das futuras iterações. A ausência de pontos nos círculos azuis indica que este ponto estava encoberto.

Os resultados são realmente impressionantes, pois apesar do grande grau de dificuldade, o erro quadrático médio das duas reconstruções ficou na casa de 1.0e-24.

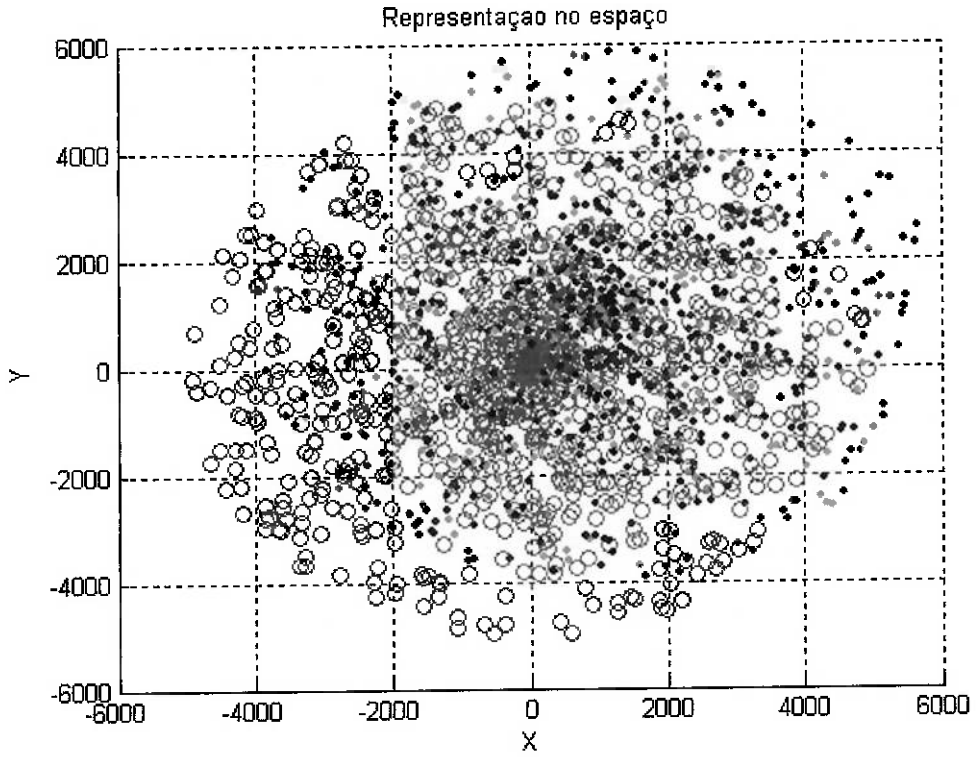

Figura 16 - Condição inicial do teste 2 para passar no DPICP

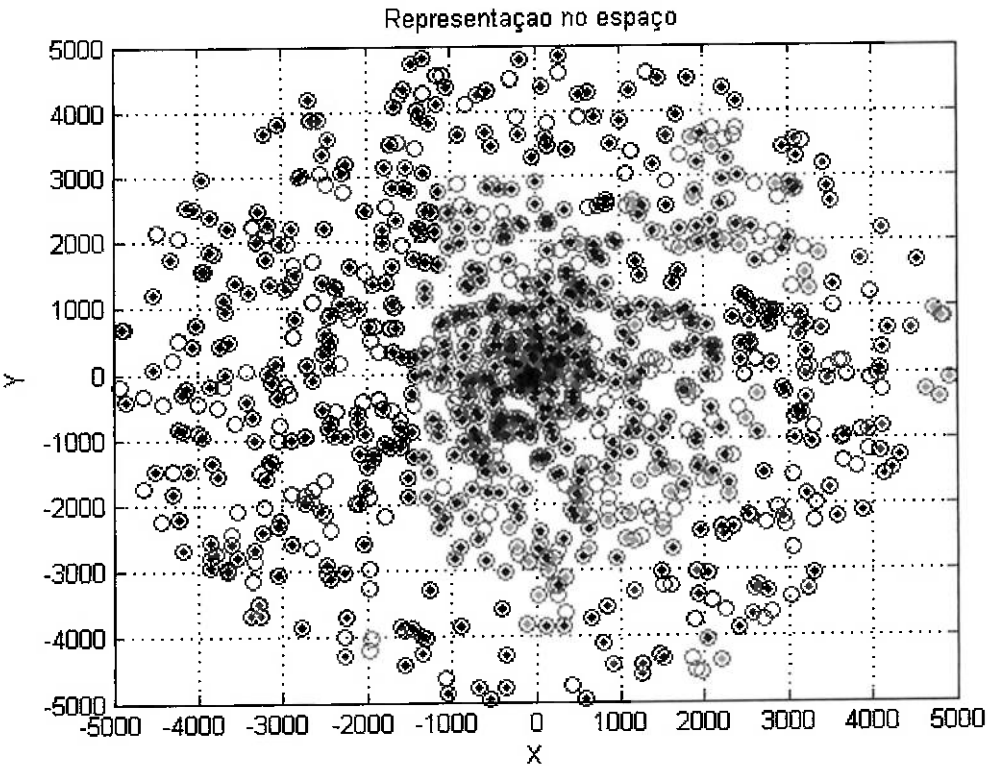

Figura 17 - Após o processamento pelo DPICP.

A seguir o DPICP foi testado na mesma condição do teste realizado com o TrICP. A tentativa é levar a imagem da figura 12 à imagem da figura 10. O resultado é mostrado na figura 18.

Em seguida tentou-se levar a imagem da figura 10 à imagem da figura 12. O resultado é mostrado na figura 19.

Os pontos vermelhos representam os pontos que durante alguma iteração não tiveram correspondência e foram excluídos das futuras iterações. A ausência de pontos nos círculos azuis indica que este ponto estava encoberto.

Os resultados mostram que o algoritmo fica preso em mínimos locais para ambos os casos em que se utilizam as imagens simuladas. Além deste problema que impossibilita a utilização deste algoritmo, nota-se que o tempo de processamento de cada reconstrução é extremamente alto, devido ao cálculo da matriz de custo da programação dinâmica.

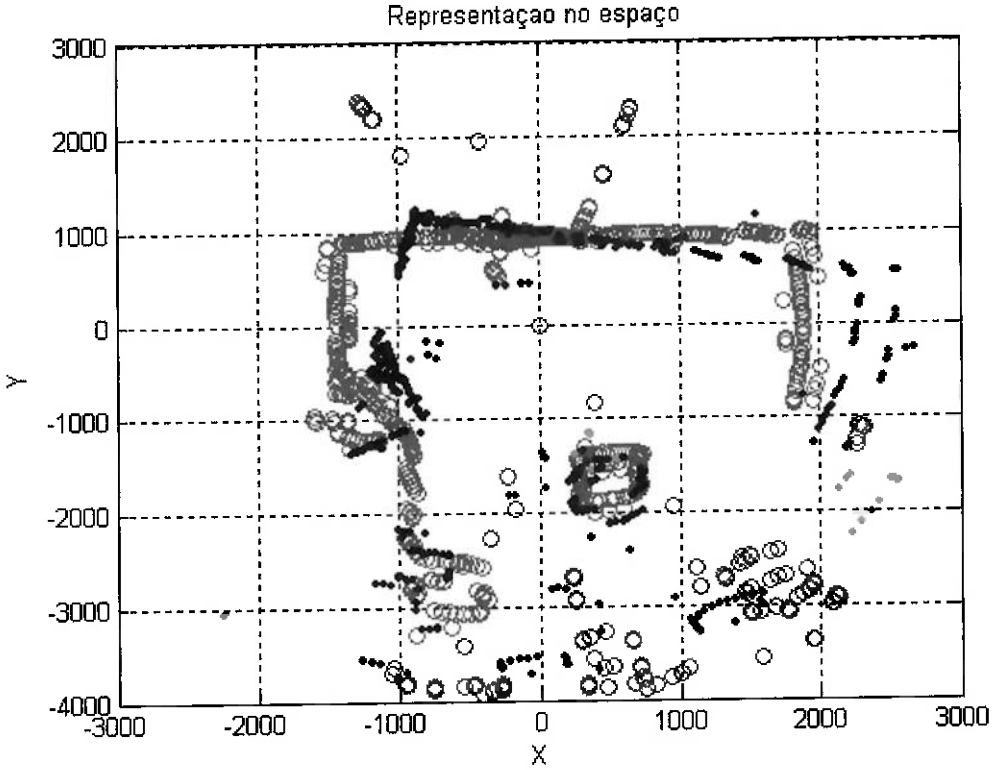

Figura 18 - Levando os pontos da figura 12 para 10.

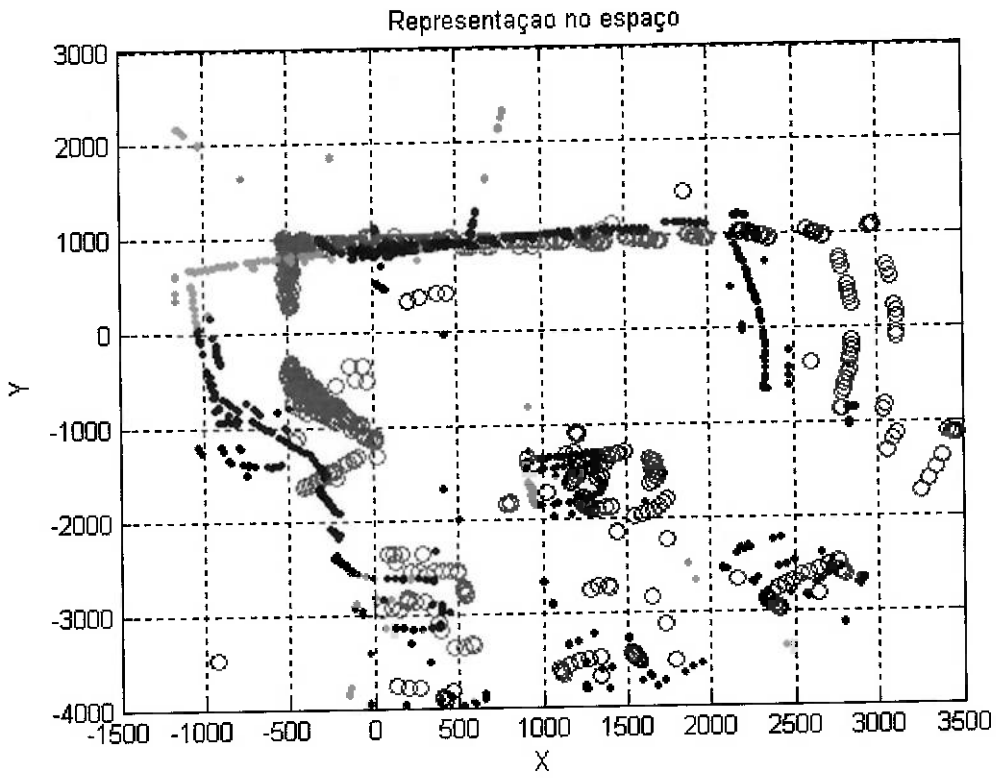

Figura 19 - Levando os pontos da figura 10 para 12.

Após estes testes volta-se para o TrICP. Duas alterações foram feitas no algoritmo original. A primeira alteração diz respeito ao número máximo de iterações do algoritmo, sendo definido pelo maior número de pontos entre o mapa e a amostra. A segunda alteração introduz um registro do mínimo global durante as iterações, ou seja, agora o algoritmo não necessariamente considera a convergência monotônica.

Foi realizado um estudo mais detalhado deste algoritmo com o conjunto das dez imagens omnidirecionais no ambiente simulado mostrado na figura 8. O objetivo desta análise foi mostrar a utilização do mínimo global e também estudar o comportamento do algoritmo quando ocorre inversão na ordem das amostras, ou seja, ao invés de tentar levar a amostra ao mapa, levar o mapa à amostra.

Na primeira linha, mapa/amostra indica a ordem das imagens para o TrICP, a iteração final mostra a iteração onde algum dos critérios de parada foi atingido, erro quadrático mostra o valor da erro quadrático na saída do algoritmo e EQM mostra o valor do erro quadrático médio na saída do algoritmo, translação x e y mostram a translação total obtida na saída do programa, real x e y mostram o valor da translação real das imagens simuladas, o erro percentual é mostrado para os casos em a translação é diferente de zero, mínimo global mostra

se o mínimo global foi usado e se sim qual a iteração em que ele foi alcançado, resultado indica se a reconstrução foi realizada com sucesso e EQM relativo mostra o valor do erro quadrático médio relativo percentual entre o inicio e o fim do algoritmo. A tabela 4 mostra os resultados para a ordem normal do TrICP.

Tabela 4 - Teste do algoritmo TrICP modificado para a ordem normal das imagens.

Mapa/amostra	1/2	2/3	3/4	4/5	5/6
Iteração final	20	56	101	36	40
Erro quadrático	4,29E+07	5,17E+07	8,00E+07	8,10E+07	4,48E+07
EQM	5,04E+04	6,65E+04	7,65E+04	6,81E+04	3,27E+04
Translação x	982,5169	1128,5	-127,608	-192,3	-479,577
Translação y	-93,611	-34,9	-684,976	-1022,1	-893,221
Real x	1000	1000	0	0	-500
Real y	0	0	-1000	-1000	-1000
Erro % x	1,74831	-12,85	-	-	4,08466
Erro % y	-	-	31,50237	-2,21	10,67793
Rotação (°)	0,9996	2,9321	-2,4539	-3,9945	-0,8929
Mínimo global?	-	-	7	14	-
Resultado	ok	ok	ok	ok	Ok
EQM relativo	80,9511	77,8573	52,8	58,2444	72,7374
Mapa/amostra	6/7	7/8	8/9	9/10	
Iteração final	50	13	60	20	
Erro quadrático	2,53E+07	1,28E+08	9,07E+06	6,40E+07	
EQM	1,58E+04	8,00E+04	6,91E+03	5,76E+04	
Translação x	-458,952	-96,3994	18,6357	-35,1203	
Translação y	-920,99	-13,2427	-943,995	-59,8758	
Real x	-500	0	0	-1000	
Real y	-1000	-1000	-1000	0	
Erro % x	8,20962	-	-	96,48797	
Erro % y	7,901	98,67573	5,60047	-	
Rotação (°)	0,09745	-3,5349	-0,2958	-5,415	
Mínimo global?	-	-	-	-	
Resultado	ok	falha	ok	falha	
EQM relativo	86,3368	5,4661	87,0789	10,5424	

A tabela 5 mostra os resultados para a ordem inversa de mapa/amostra para o TrICP.

Tabela 5 - Teste do algoritmo TrICP modificado para a ordem invertida das imagens.

Mapa/amostra	2/1	3/2	4/3	5/4	6/5
Iteração final	20	24	18	46	32
Erro quadrático	8,47E+06	2,21E+07	5,39E+07	1,20E+08	6,63E+07
EQM	1,28E+04	2,59E+04	6,94E+04	1,14E+05	5,35E+04
Translação x	907,4872	983,265	21,3093	140,4506	-258,783
Translação y	73,4825	17,0901	-579,882	437,6076	-830,391
Real x	1000	1000	0	0	-500
Real y	0	0	-1000	-1000	-1000
Erro % x	9,25128	1,6735	-	-	48,24334
Erro % y	-	-	42,0118	143,7608	16,96091
Rotação (°)	1,1375	0,2452	-0,3109	-4,6911	1,6903
Mínimo global?	-	-	-	15	-
Resultado	Ok	ok	falha	falha	ok
EQM relativo	93,2185	85,208	45,0183	16,4993	59,5491
mapa/amostra	7/6	8/7	9/8	10/9	
Iteração final	17	78	101	16	
Erro quadrático	5,47E+07	7,72E+07	4,83E+07	1,93E+07	
EQM	3,99E+04	4,84E+04	3,02E+04	1,47E+04	
Translação x	-532,552	-163,076	-7,1122	-944,864	
Translação y	-727,001	-884,475	-935,329	7,6305	
Real x	-500	0	0	-1000	
Real y	-1000	-1000	-1000	0	
Erro % x	-6,51038	-	-	5,51356	
Erro % y	27,29986	11,55255	6,46709	-	
Rotação (°)	-0,3859	3,4378	0,7865	0,5797	
Mínimo global?	-	34	17	-	
Resultado	ok	falha	ok	ok	
EQM relativo	77,782	51,1908	71,5662	92,2737	

Observa-se que o valor do erro quadrático em todas as saídas do algoritmo esta na casa de 1.0e7 e do erro quadrático médio esta na casa de 1.0e4, indicando que estes dois parâmetros não são úteis como parâmetros para determinar se a reconstrução foi ou não bem sucedida. Em todos os casos em a rotação se aproxima de 5 graus o resultado foi uma falha no processo de correspondência. Nota-se também que o mínimo global é utilizado em 5 de 18 tentativas. Com relação ao erro quadrático médio relativo, nota-se que quando o valor deste é baixo o resultado da correspondência é sempre uma falha. Nota-se, portanto, que a qualidade da correspondência pode ser medida por esta variável. No algoritmo da construção de mapa são usados os EQM relativo e o valor da rotação resultantes do TrICP como variáveis para o controle de uma correspondência bem sucedida.

De modo a reduzir o tempo computacional das iterações do TrICP, foi testada uma maneira de reduzir o número os pontos resultantes da reconstrução tridimensional. Para isso consideram-se apenas os pontos mais próximos ao centro da amostra por um incremento angular de 1°, assim resultando em no máximo 360 pontos por amostra filtrada.

Nas tabelas 6 e 7 vemos os resultados para as amostras na ordem normal e invertida respectivamente.

Tabela 6 - Teste do algoritmo TrICP modificado para a ordem normal das imagens filtradas.

Mapa/amostra	1/2	2/3	3/4	4/5	5/6
Iteração final	16	15	16	32	37
Erro quadrático	1,28E+07	2,82E+06	3,30E+07	6,76E+07	3,44E+07
EQM	4,02E+04	8,91E+03	9,72E+04	1,97E+05	1,06E+05
Translação x Translação y	971,4189 60,8647	952,7618 20,5735	-210,694 -770,681	156,2 -1159,6	-522,2 -1068,1
Real x Real y	1000 0	1000 0	0 -1000	0 -1000	-500 -1000
Erro % x Erro % y	2,85811 -	4,72382 -	- 22,93191	- -15,96	-4,44 -6,81
Rotação (°)	-1,3759	1,0742	-1,5985	-33,4072	0,9584
Mínimo global?	-	-	-	-	-
Resultado	ok	ok	falha	falha	ok
EQM relativo	89,9437	97,5069	70,3718	46,7968	66,6896

Mapa/amostra	6/7	7/8	8/9	9/10
Iteração final	35	39	35	59
Erro quadrático	2,11E+07	6,40E+07	2,29E+07	1,87E+07
EQM	6,38E+04	1,87E+05	6,70E+04	6,14E+04
Translação x Translação y	-474,2 -801,78	-42,7673 -833,938	176,5 -1047,8	-782,886 -93,2387
Real x Real y	-500 -1000	0 -1000	0 -1000	-1000 0
Erro % x Erro % y	5,16008 19,82196	- 16,60625	- -4,78	21,71143 -
Rotação (°)	-0,8266	-2,5635	-18,9598	3,1793
Mínimo global?	-	-	-	11
Resultado	ok	ok	falha	ok
EQM relativo	68,1074	35,9395	68,0671	72,1687

Tabela 7 - Teste do algoritmo TrICP modificado para a ordem invertida das imagens filtradas.

mapa/amostra	2/1	3/2	4/3	5/4	6/5
Iteração final	24	20	24	20	16
Erro quadrático	2,17E+06	3,68E+06	6,67E+06	7,39E+07	3,36E+07
EQM	1,22E+04	1,16E+04	2,11E+04	2,17E+05	9,76E+04
Translação x Translação y	972,352 40,3294	948,3271 16,1554	37,2117 -987,083	-426,479 -573,793	-363,867 -737,365
Real x Real y	1000 0	1000 0	0 -1000	0 -1000	-500 -1000
Erro % x Erro % y	2,7648 -	5,16729 -	- 1,29174	- 42,62074	27,22654 26,2635
Rotação (°)	-0,453	-2,4033	4,3158	19,5066	-2,0809
Mínimo global?	-	-	-	-	-
Resultado	ok	ok	ok	falha	falha
EQM relativo	96,0987	96,4553	89,6545	41,5266	59,6733
mapa/amostra	7/6	8/7	9/8	10/9	
Iteração final	34	96	31	101	
Erro quadrático	4,89E+07	9,98E+07	1,15E+07	9,42E+07	
EQM	1,51E+05	3,03E+05	3,35E+04	2,75E+05	
Translação x Translação y	-491,5 -1062,8	-675,514 -18,1553	45,8016 -942,151	-870,584 441,7139	
Real x Real y	-500 -1000	0 -1000	0 -1000	-1000 0	
Erro % x Erro % y	1,7 -6,28	- 98,18447	- 194,2151	12,94163 -	
Rotação (°)	-2,4818	-17,7097	-3,4429	-19,4577	
Mínimo global?	10	8	12	10	
Resultado	ok	falha	ok	falha	
EQM relativo	64,2562	31,4456	86,4813	37,0061	

Com relação ao valor do erro quadrático e ao erro quadrático médio notou-se o mesmo comportamento das amostras não filtradas. O mesmo pode ser dito da rotação e do mínimo global, inclusive o número de ocorrências 5 em 18, embora notadamente a ocorrência tenha sido maior quando as amostras estavam na ordem invertida. Embora o mesmo comportamento com relação ao EQM relativo tenha sido notado nas amostras filtradas, notou-se que o seu valor nos casos de falha são maiores que nos casos não filtrados.

Conclui-se, portanto que com o uso de valores limites adequados do EQM relativo e da rotação pode-se realizar o controle sob a qualidade da correspondência entre as imagens, um fator que é essencial na montagem do mapa global. Também é importante notar que se pode usar com segurança as amostras filtradas reduzindo enormemente o tempo computacional do algoritmo de construção de mapa.

Com o TrICP desenvolvido e os parâmetros para o controle da correspondência definidos, o primeiro algoritmo para construção de mapas de ambiente foi desenvolvido. A primeira conclusão com a sua utilização é que a distância entre uma imagem e a próxima não pode ser tão grande, como por exemplo o um metro de espaçamento utilizado para a análise do TrICP. O problema é que com este espaçamento a chance de uma correspondência mal sucedida é razoável, em torno de 35%, como se pode ver nas tabelas 5 a 8 mostradas acima.

Assim sendo, a sala virtual foi fotografada novamente com um espaçamento de meio metro entre as imagens usando a mesma trajetória da figura 8, porém agora com 19 pontos de tomada de imagem nas coordenadas mostradas na tabela 8 abaixo.

Tabela 8 - Coordenadas da tomada de pontos da trajetória do robô no ambiente simulado

Ponto	Coord. X	Coord. Z	Ponto	Coord. X	Coord. Z
1	3000	1500	11	0	0
2	3000	1000	12	-500	250
3	3000	500	13	-1000	500
4	3000	0	14	-1500	500
5	3000	-500	15	-2000	500
6	2500	-500	16	-2500	500
7	2000	-500	17	-3000	500
8	1500	-500	18	-3000	1000
9	1000	-500	19	-3000	1500
10	500	-250			

O ambiente também foi fotografado com um espaçamento de vinte cinco centímetros. As coordenadas são mostradas na tabela 9, porém agora com 37 imagens.

Tabela 9 - Coordenadas da tomada de pontos da trajetória do robô no ambiente simulado

Ponto	Coord. X	Coord. Z	Ponto	Coord. X	Coord. Z	Ponto	Coord. X	Coord. Z
1	3000	1500	14	1750	-500	27	-1500	500
2	3000	1250	15	1500	-500	28	-1750	500
3	3000	1000	16	1250	-500	29	-2000	500
4	3000	750	17	1000	-500	30	-2250	500
5	3000	500	18	750	-375	31	-2500	500
6	3000	250	19	500	-250	32	-2750	500
7	3000	0	20	250	-125	33	-3000	500
8	3000	-250	21	0	0	34	-3000	750
9	3000	-500	22	-250	125	35	-3000	1000
10	2750	-500	23	-500	250	36	-3000	1250
11	2500	-500	24	-750	375	37	-3000	1500
12	2250	-500	25	-1000	500			
13	2000	-500	26	-1250	500			

A seguir, na figura 20, pode se ver a resultado da seqüência de imagens espaçadas de meio metro, na ordem direta, ou seja, do ponto 1 ao 19, passadas pelo primeiro algoritmo de construção de mapas de ambiente.

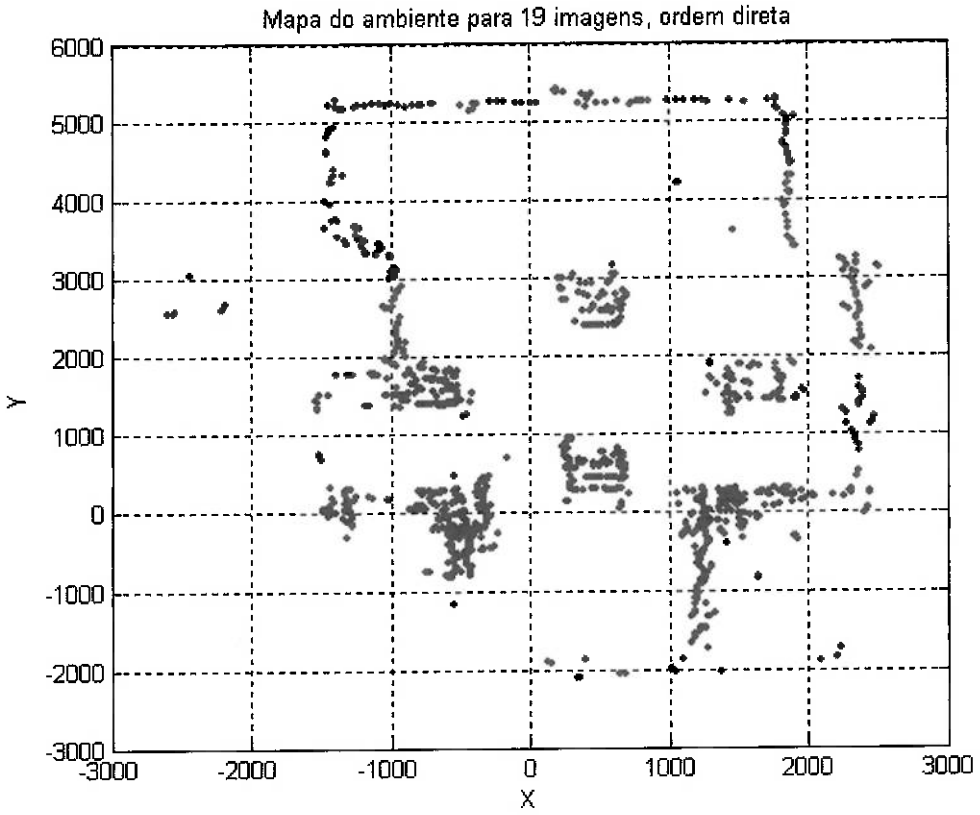

Figura 20 - Mapa do ambiente para 19 imagens na ordem direta no primeiro algoritmo.

Nota-se que a reconstrução no eixo *Y* possui um comprimento de 7,2 metros enquanto no ambiente real, esta medida possui 8 metros, ou seja, houve um erro de 10,0% nesta reconstrução. No eixo *X* houve um erro de aproximadamente 10 centímetros em 4 metros, que representa um erro de 2,5% na reconstrução do mapa. A coordenada (0,0) no mapa registra a última auto-localização conhecida, que neste mapa esta defasada de menos um metro em *X* e em *Y*. O fato é que o algoritmo não foi capaz de realizar a correspondência nas últimas quatro imagens da seqüência e conseqüentemente descartou a informação contida nas mesmas, por isso a ausência da parede na parte inferior da figura 20.

Os fatores que influenciam na capacidade do algoritmo de obter uma boa correspondência são a distância entre as imagens consecutivas, o número de pontos novos que

potencialmente poderão ser acrescentados ao mapa global e ocorrência de mudança de direção na trajetória.

No caso mostrado acima, dois destes fatores estão reunidos e comprometem a qualidade do mapa obtido, a mudança de direção da trajetória e uma grande quantidade de pontos novos das imagens.

A seguir, na figura 21, pode se ver a resultado da seqüência de imagens espaçadas de meio metro, na ordem inversa, ou seja, do ponto 19 ao 1, passadas pelo primeiro algoritmo de construção de mapas de ambiente.

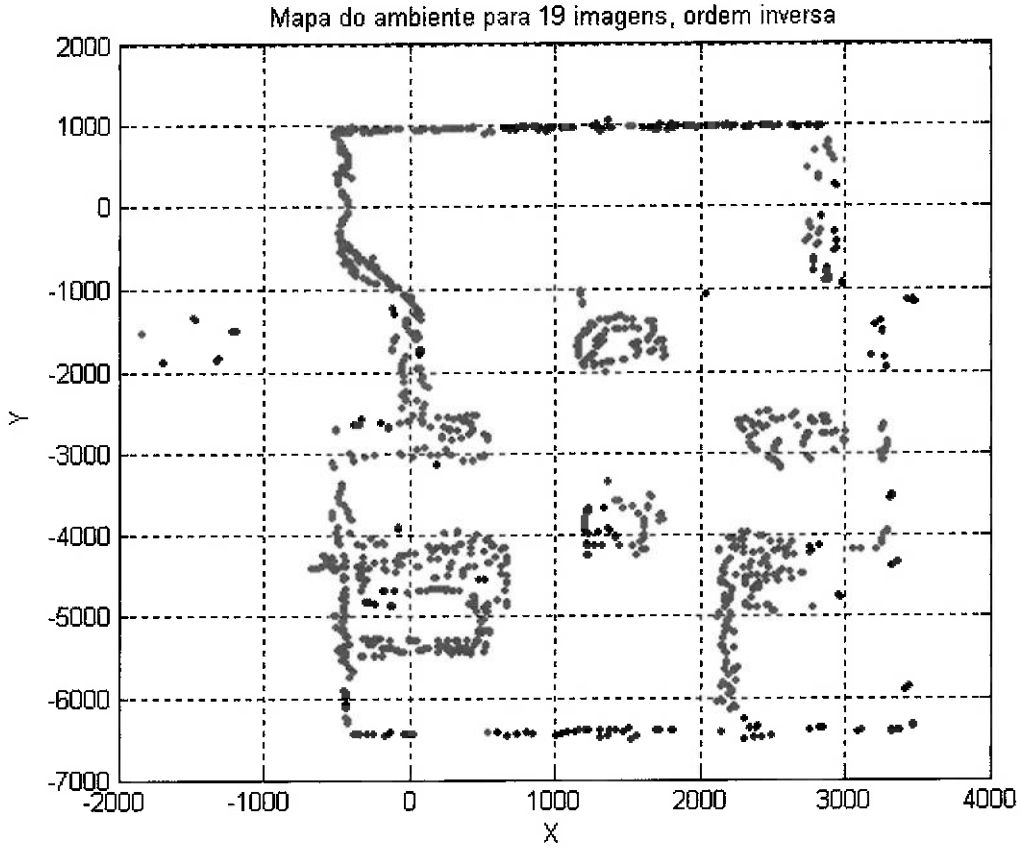

Figura 21 - Mapa do ambiente para 19 imagens na ordem inversa no primeiro algoritmo.

Nota-se que a reconstrução no eixo *Y* possui um comprimento de 7,4 metros enquanto no ambiente real, esta medida possui 8 metros, ou seja, houve um erro de 7,5% nesta reconstrução. No eixo *X* houve um erro de aproximadamente 20 centímetros em 4 metros, que representa um erro de 5,0% na reconstrução do mapa. A coordenada (0,0) no mapa registra a última auto-localização conhecida, que neste mapa esta coerente com a posição final do robô.

Ainda que o algoritmo tenha sido capaz de reconstruir o ambiente de modo correto, as paredes do lado direito da imagem estão pouco definidas.

A seguir, na figura 22, pode se ver a resultado da seqüência de imagens espaçadas de vinte cinco centímetros, na ordem direta, ou seja, do ponto 1 ao 37, passadas pelo primeiro algoritmo de construção de mapas de ambiente.

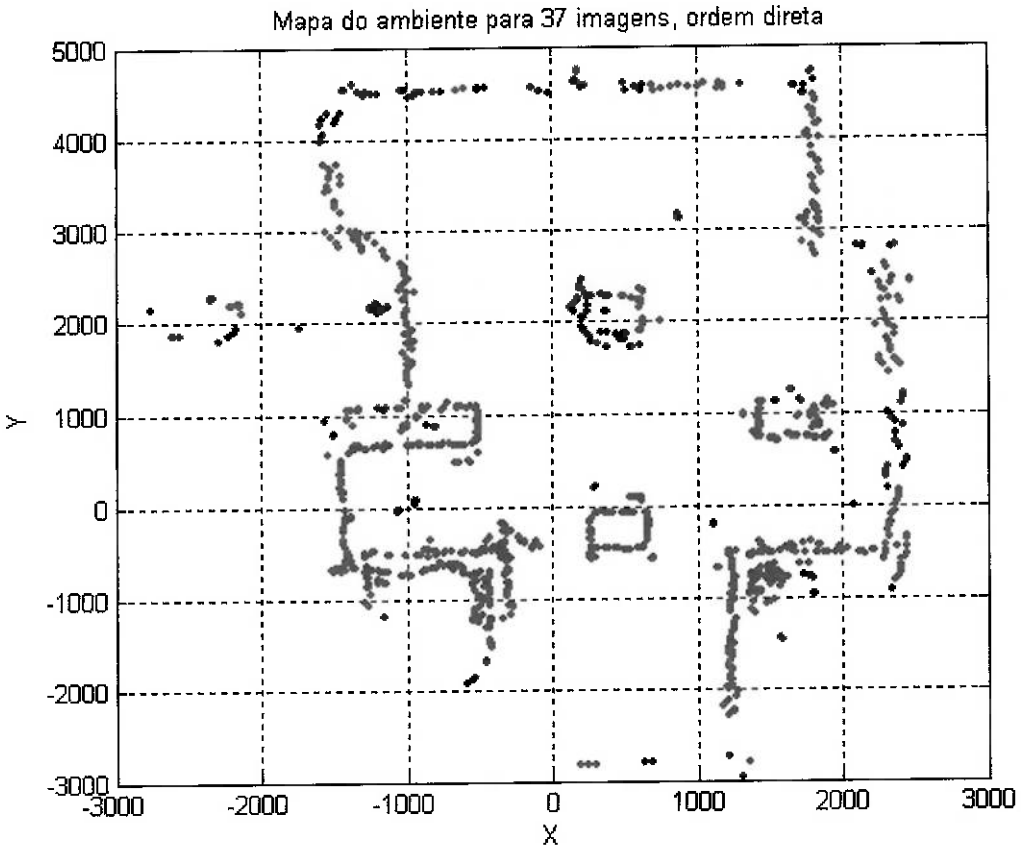

Figura 22 - Mapa do ambiente para 37 imagens na ordem direta no primeiro algoritmo.

Nota-se que a reconstrução no eixo *Y* possui um comprimento de 7,5 metros enquanto no ambiente real, esta medida possui 8 metros, ou seja, houve um erro de 6,25% nesta reconstrução. No eixo *X* houve um erro de aproximadamente 20 centímetros em 4 metros, que representa um erro de 5,0% na reconstrução do mapa. A coordenada (0,0) no mapa registra a última auto-localização conhecida, que neste mapa esta defasada de menos um metro em *X* e dois metros em *Y*. O fato é que o algoritmo não foi capaz de realizar a correspondência nas últimas doze imagens da seqüência e conseqüentemente descartou a informação contida nas mesmas, por isso a ausência da parede na parte inferior da figura 22.

A seguir, na figura 23, pode se ver a resultado da seqüência de imagens espaçadas de vinte cinco centímetros, na ordem inversa, ou seja, do ponto 37 ao 1, passadas pelo primeiro algoritmo de construção de mapas de ambiente.

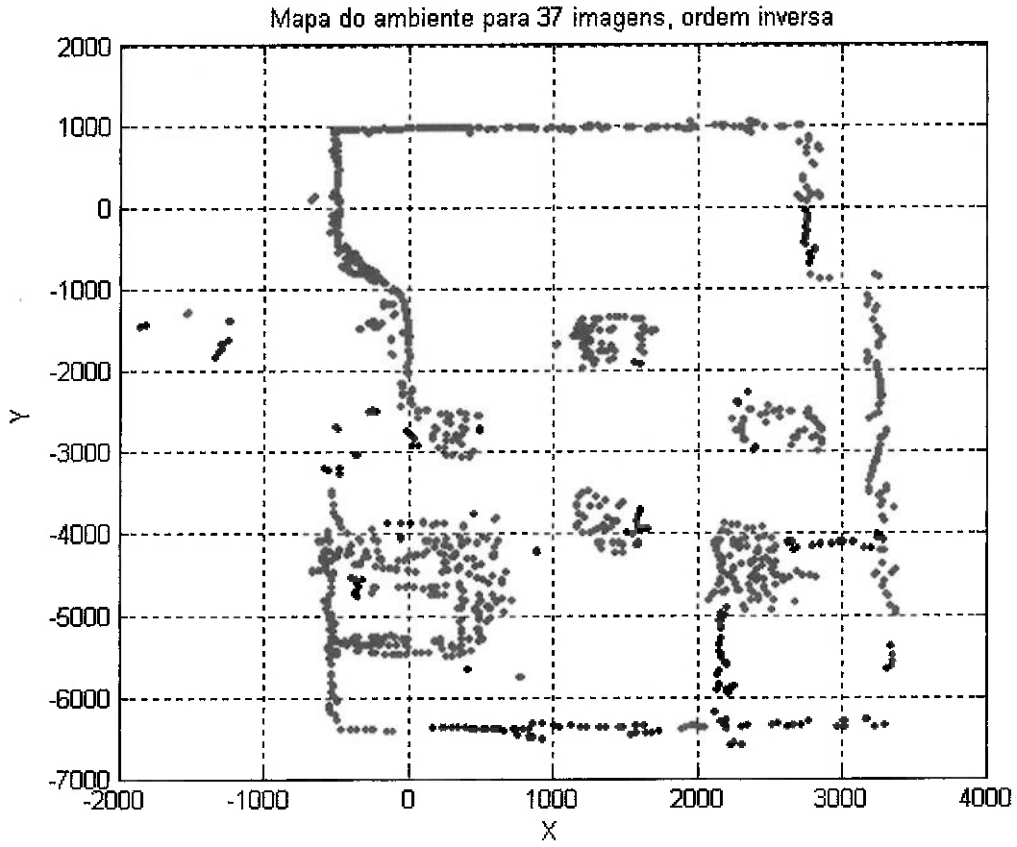

Figura 23 - Mapa do ambiente para 37 imagens na ordem inversa no primeiro algoritmo.

Nota-se que a reconstrução no eixo *Y* possui um comprimento de 7,4 metros enquanto no ambiente real, esta medida possui 8 metros, ou seja, houve um erro de 7,5% nesta reconstrução. No eixo *X* houve um erro de aproximadamente 20 centímetros em 4 metros, que representa um erro de 5,0% na reconstrução do mapa. A coordenada (0,0) no mapa registra a última auto-localização conhecida, que neste mapa esta coerente com a posição final do robô. O algoritmo foi capaz de reconstruir o ambiente de modo correto e com as paredes do ambiente bem definidas.

Após os resultados mostrados acima, uma nova proposta de algoritmo foi feita com o objetivo de aumentar o número de tentativas de correspondência para cada nova imagem

adquirida da seqüência. Com esta idéia em mente foi desenvolvido o segundo algoritmo de construção de mapa de ambiente e seus resultados são mostrados a seguir.

A seguir, na figura 24, pode se ver a resultado da seqüência de imagens espaçadas de meio metro, na ordem direta, ou seja, do ponto 1 ao 19, passadas pelo segundo algoritmo de construção de mapas de ambiente.

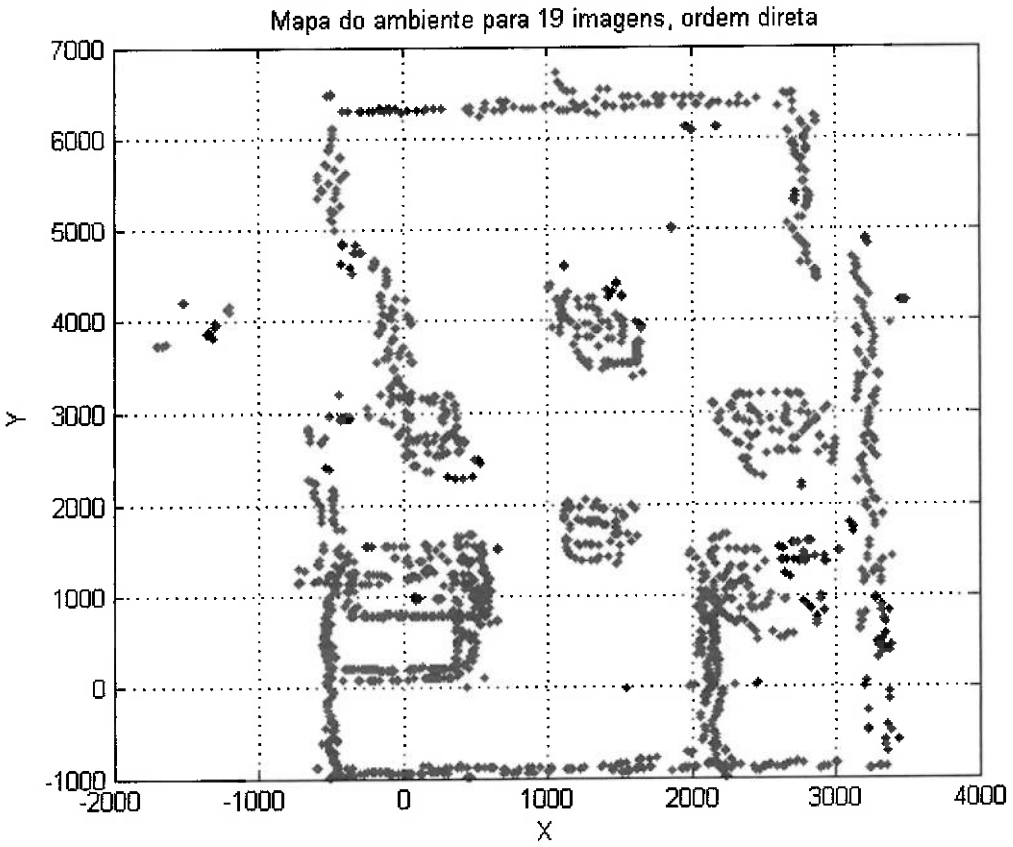

Figura 24 - Mapa do ambiente para 19 imagens na ordem direta no segundo algoritmo.

Nota-se que a reconstrução no eixo *Y* possui um comprimento de 7,3 metros enquanto no ambiente real, esta medida possui 8 metros, ou seja, houve um erro de 8,75% nesta reconstrução. No eixo *X* houve um erro de aproximadamente 20 centímetros em 4 metros, que representa um erro de 5,0% na reconstrução do mapa. A coordenada (0,0) no mapa registra a última auto-localização conhecida, que neste mapa esta coerente com a posição final do robô. O algoritmo foi capaz de reconstruir o ambiente de modo correto e com as paredes do ambiente bem definidas.

A seguir, na figura 25, pode se ver a resultado da seqüência de imagens espaçadas de meio metro, na ordem inversa, ou seja, do ponto 19 ao 1, passadas pelo segundo algoritmo de construção de mapas de ambiente.

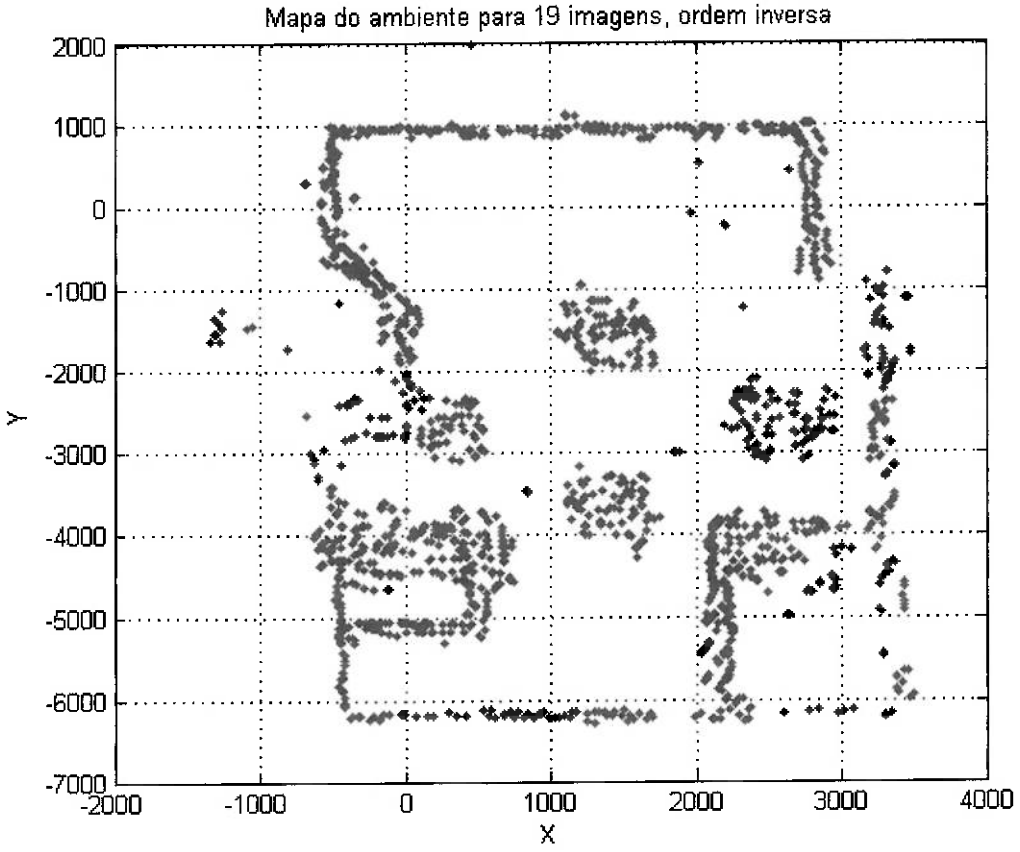

Figura 25 - Mapa do ambiente para 19 imagens na ordem inversa no segundo algoritmo.

Nota-se que a reconstrução no eixo *Y* possui um comprimento de 7,2 metros enquanto no ambiente real, esta medida possui 8 metros, ou seja, houve um erro de 10,0% nesta reconstrução. No eixo *X* houve um erro de aproximadamente 20 centímetros em 4 metros, que representa um erro de 5,0% na reconstrução do mapa. A coordenada (0,0) no mapa registra a última auto-localização conhecida, que neste mapa esta coerente com a posição final do robô. O algoritmo foi capaz de reconstruir o ambiente de modo correto e com as paredes do ambiente bem definidas.

A seguir, na figura 26, pode se ver a resultado da seqüência de imagens espaçadas de vinte cinco centímetros, na ordem direta, ou seja, do ponto 1 ao 37, passadas pelo segundo algoritmo de construção de mapas de ambiente.

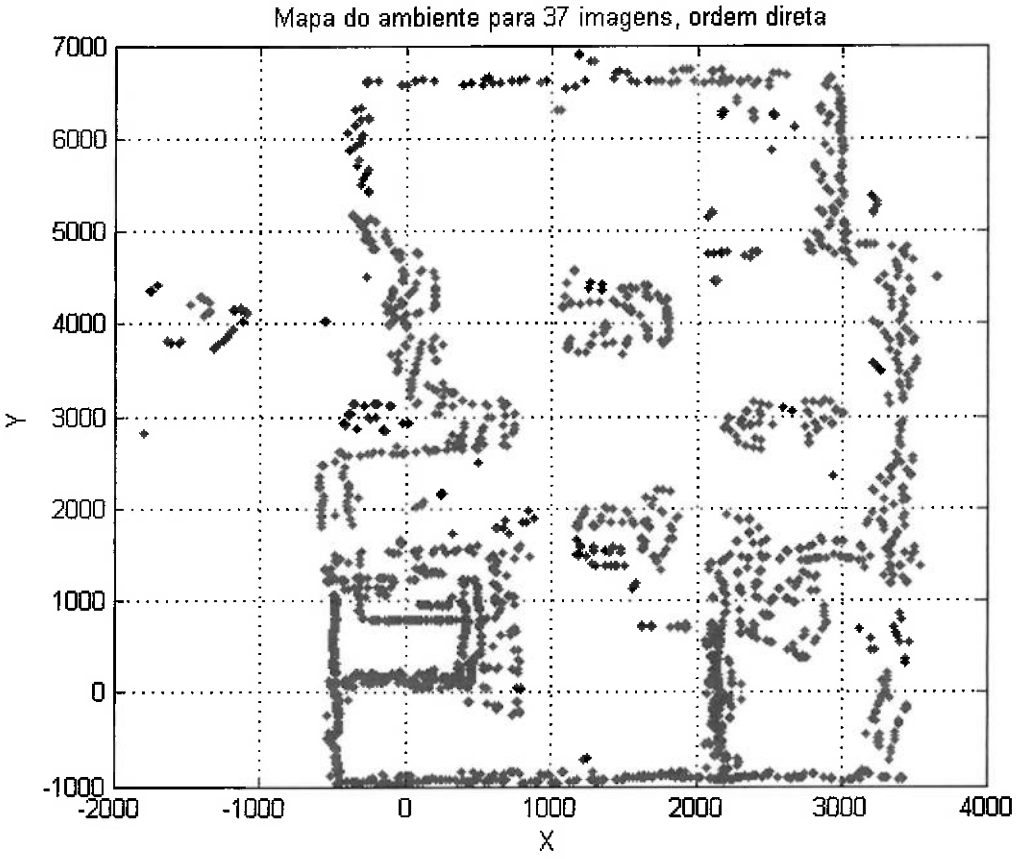

Figura 26 - Mapa do ambiente para 37 imagens na ordem direta no segundo algoritmo.

Nota-se que a reconstrução no eixo *Y* possui um comprimento de 7,6 metros enquanto no ambiente real, esta medida possui 8 metros, ou seja, houve um erro de 5,0% nesta reconstrução. No eixo *X* houve um erro de aproximadamente 5 centímetros em 4 metros, que representa um erro de 1,25% na reconstrução do mapa. A coordenada (0,0) no mapa registra a última auto-localização conhecida, que neste mapa esta coerente com a posição final do robô. O algoritmo foi capaz de reconstruir o ambiente de modo correto e com as paredes do ambiente bem definidas.

A seguir, na figura 27, pode se ver a resultado da seqüência de imagens espaçadas de vinte cinco centímetros, na ordem inversa, ou seja, do ponto 37 ao 1, passadas pelo segundo algoritmo de construção de mapas de ambiente.

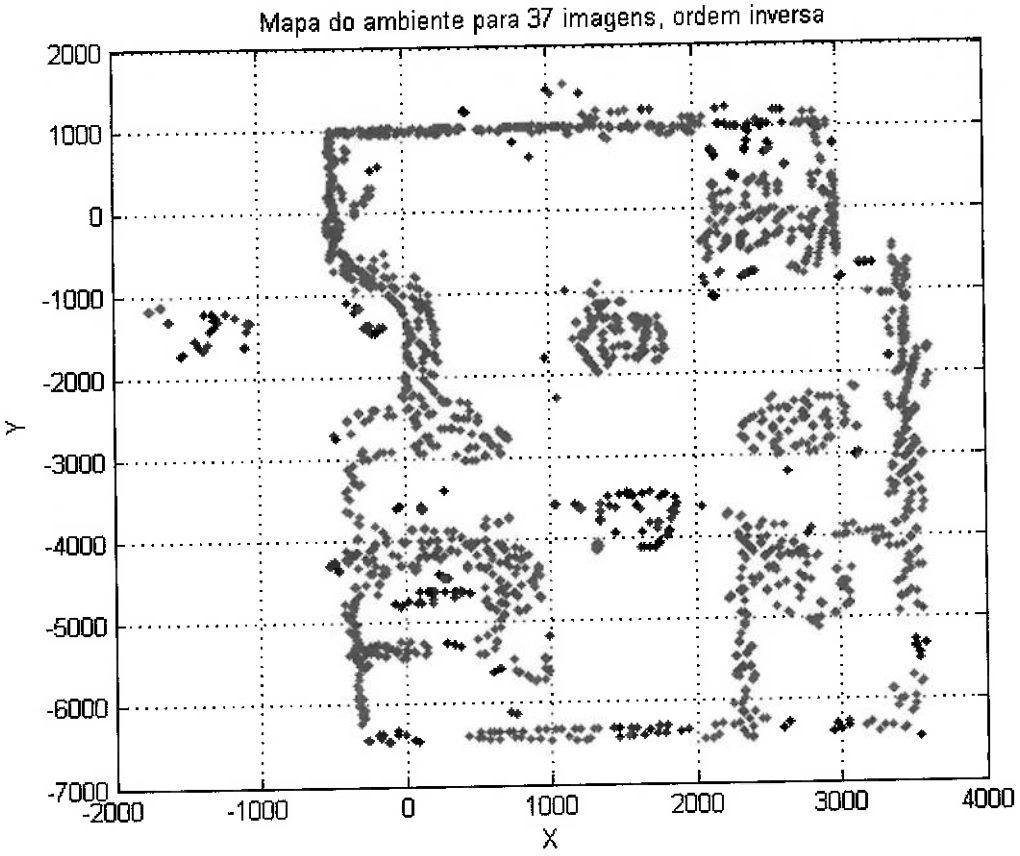

Figura 27 - Mapa do ambiente para 37 imagens na ordem inversa no segundo algoritmo.

Nota-se que a reconstrução no eixo *Y* possui um comprimento de 7,4 metros enquanto no ambiente real, esta medida possui 8 metros, ou seja, houve um erro de 7,5% nesta reconstrução. No eixo *X* houve um erro de aproximadamente 5 centímetros em 4 metros, que representa um erro de 1,25% na reconstrução do mapa. A coordenada (0,0) no mapa registra a última auto-localização conhecida, que neste mapa esta coerente com a posição final do robô. O algoritmo foi capaz de reconstruir o ambiente de modo correto e com as paredes do ambiente bem definidas.

Ao final, pode-se notar que a qualidade de todos os mapas obtidos no segundo algoritmo de construção de mapas de ambiente é superior ao primeiro algoritmo. No segundo algoritmo uma melhor qualidade de mapa foi obtida com o espaçamento de meio metro entre as imagens. Isso se deve ao fato de que com um grande número de imagens a quantidade de ruído e pontos espúrios da reconstrução tridimensional aumenta consideravelmente, ainda que possibilite uma correspondência bem sucedida. Nota-se que todos os pontos finais da auto-

localização do robô no segundo algoritmo estão corretas, ou seja, nenhuma informação das imagens foi descartada, gerando um mapa de alto grau de confiabilidade.

3.3 Testes do sistema em ambiente real

Não foi possível a aquisição das imagens omnidirecionais de um ambiente real. Após a aquisição da nova câmera, Legend 544 da DVT Machine Vision foram feitas inúmeras tentativas de calibração do sistema de aquisição de imagem. O problema é que espelho não é compatível com a nova câmera, pois não foi projetado para a mesma. As dificuldades encontradas estão no ajuste do foco do conjunto espelho-câmera, focalização simultânea das imagens interna e externa do espelho duplo de perfil hiperbólico e ajuste na mudança dos parâmetros da câmera no algoritmo da reconstrução tridimensional.

A nova câmera adquirida gera imagens que possuem 1280x1024 pixels, sendo que 62 pixels são descartados da imagem panorâmica, resultando em 225 pixels de altura para cada imagem das duas seções do espelho. O sensor CCD da câmera possui 6,4mm por 4,8mm o que corresponde a 4,65μm por 4,65μm por pixel. A largura da imagem panorâmica é de 360 pixels, considerando que a imagem panorâmica cobre 360° ao redor do espelho, a resolução tangencial da imagem é 1°.

Ressalta-se que um novo espelho, especialmente projetado para esta nova câmera, será fabricado no início do primeiro semestre de 2006.

4. Conclusão

O desenvolvimento do algoritmo de construção de mapa de ambiente foi bem sucedido, tendo alcançado as expectativas iniciais do projeto. Foram obtidos mapas de boa qualidade com a informação adicional da auto-localização do robô no ambiente mapeado.

Os testes no ambiente simulado foram extremamente importantes para validação dos algoritmos desenvolvidos neste trabalho, pois não foi possível a obtenção das imagens reais. Todos os testes do algoritmo de construção de mapa de ambiente foram realizados com base em uma sala virtual construída no programa POV RAY. Com as imagens fotografadas neste

ambiente os algoritmos baseados no ICP foram testados e também o os dois algoritmos de construção de mapa de ambiente implementados neste trabalho.

4.1 Considerações finais

Com relação ao algoritmo para construção do mapa de ambiente, o programa responsável pela correspondência de duas imagens obtidas pelo sistema, foi concluído satisfatoriamente. O TrCP funciona de modo satisfatório para as pequenas translações e rotações iniciais, entretanto não foi robusto o suficiente para grandes rotações e translações iniciais. Usando-se o TrICP nas imagens obtidas no ambiente simulado a translação é reconstruída com sucesso com erro inferior a 10% na grande maioria dos casos. Nota-se que as modificações feitas no algoritmo original aumentaram consideravelmente a capacidade do algoritmo de obter correspondências bem sucedidas.

O primeiro algoritmo de construção de mapas de ambiente foi exaustivamente testado e produz mapas de qualidade média. Com ele verificou-se que são necessários pontos com menor espaçamento entre as imagens. Verificou-se também a dificuldade de obter uma correspondência quando há uma mudança de direção na trajetória, em alguns outros casos testados, detectou-se a necessidade de menor espaçamento entre as imagens adquiridas no começo da geração do mapa, de modo a certificar a coerência dos pontos do mapa global inicial.

Com base na experiência adquirida do primeiro algoritmo, um segundo algoritmo foi criado especialmente para cobrir as falhas do primeiro. O segundo algoritmo funcionou satisfatoriamente em todos os casos de teste. Com ele se atinge satisfatoriamente o objetivo proposto pelo projeto de iniciação científica, de gerar um mapa de ambiente baseado em uma seqüência de imagens omnidirecionais estéreo. Ressalta-se que nenhuma informação adicional é fornecida ao algoritmo além da seqüência de imagens, como por exemplo, odômetria. As distâncias aos objetos e a auto-localização do robô é feita diretamente das informações provenientes da reconstrução tridimensional e da correspondência da imagens fotografadas.

Nota-se, contudo, que o tempo computacional exigido pelo algoritmo impossibilita que ele seja usado em uma aplicação que necessite da geração do mapa em tempo real.

4.2 Trabalhos Futuros

Com base nos algoritmos desenvolvidos neste trabalho objetiva-se realizar um projeto de mestrado que consiste no desenvolvimento de um algoritmo para auto-localização de robôs móveis e mapeamento do ambiente simultâneo, também conhecido como problema SLAM, *Simultaneous Localization and Mapping*, ou CLM, *Concurrent Localization and Mapping*, em um ambiente estruturado, ou seja, que pode ser descrito através de primitivas geométricas, e dinâmico. O mapa será construído retirando-se imagens adquiridas por um sistema de visão omnidirecional estéreo baseado em um espelho duplo de perfil hiperbólico. A partir de uma única imagem omnidirecional obtida, utilizando-se algoritmos de visão estéreo, obtêm-se as distâncias de objetos presentes no ambiente ao sistema de visão. A partir da correspondência de várias imagens tomadas em diferentes posições cria-se o mapa do ambiente e inerentemente localiza-se o robô.

Um passo importante do desenvolvimento do algoritmo de auto-localização e mapeamento simultâneo é a sua implementação em um hardware dedicado que possa ser embarcado em um robô móvel. Será analisada a possibilidade da utilização do processador Hitachi SH4, da câmera Legend 544, com 64 Mb de memória RAM e 16 Mb de memória flash. O programa será então convertido do MatLab para a uma linguagem orientada a objetos utilizada na programação da câmera.

5. Referências bibliográficas

Arun et al. (1987). Arun, K.S.; Huang, T.S.; Blostein, S. D., "Least-squares fitting of two 3-D point sets", *Proceeding of IEEE Trans Pattern Anal Machine Intell 9,* pp.698-700, 1987.

Baker e Nayar (1998). S. Baker and S. K. Nayar, "A theory of catadioptric image formation", In *Proc. of the 6th Int. Conf. on Computer Vision* - ICCV'98, pg. 35-42, 1998.

Besl e Mckay (1992). Besl, P. J.; Mckay, N. D., "A Method for Registration of 3-D Shapes", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14(2), pp.239-256, 1992.

Brooks (1986). Brooks, R. A., "A Robust Layared Control System for mobile Robot", *IEEE Journal of Robotics and Automation*, vol. 2, n°1, pp. 14-23, 1986.

Cabral et al. (2004). Cabral, E. L. L.; Junior, J. C. S.; Hunold, M. C., "Omnidirectional Stereo Vision with a hiperbolic Double Lobed Mirror", *ICPR* 2004.

Chahl e Srinivasan (1997). J. S. Chahl and M. V. Srinivasan, "Reflective surfaces for panoramic imaging", *Applied Optics*, 30(31):8275-8282, 1997.

Chetverikov et al. (2002a). Chetverikov, D.; Svirko, D.; Stepanov, D.; Krsek, P., "The Trimmed Iterative Closest Point Algorithm", *Proceeding of ICPR'02*, Québec, 2002.

Chetverikov e Stepanov (2002b). Chetverikov, D.; Stepanov, D., "Robust Euclidean Alignment of 3D Point Sets", *First Hungarian Conference on Computer Grafics and Geometry*, Budapest, pp.70-75, 2002.

Conroy e Moore (1999). T. L. Conroy and J. B. Moore, "Resolution invariant surfaces for panoramic vision systems", In *Proc. of the 7th Int. Conf. on Computer Vision* - ICCV'99, pg. 392-397, 1999.

Deccó (2004). Deccó, C. C. G., "Construção de Mapas de Ambiente para Navegação de Robos Móveis com Visão Omnidirecional Estéreo", tese de doutorado, 2004.

Desouza e Kak (2002). Desouza, G. N.; Kak, A. C., "Vision for Móbile Robot Navigation: A Sursey", *IEEE Transactions o Pattern Analysis and Machine Intelligence*, vol. 24, pp. 237-267, Feb. 2002.

Eggert et al. (1997). Eggert, D. W.; Lorusso, A.; Fisher, R. B., "Estimating 3-D rigid body transformations: a comparison of four major algorithms", *Machine Vision and Applications 9*, pp.272-290, Março de 1997.

Fielding e Kam (2000). Fielding, G.; Kam, M., "Weighted matchings for dense stereo correspondence", *Pattern recognition 33*, pp. 1511-1524, Junho de 1999.

Gaspar e Santos-Victor (1999). Gaspar, J.; Santos- Victor, J., "Visual Path Following with a Catadioptric Panoramic Câmera", *International Symposium on Intelligent Robotic System*, SIR`99, Coimbra, Portugal, July 1999.

Gaspar et al. (2000). Gaspar, J.; Winters, N.; Santos-Victor, J., "Vision-based Navigation and Environmental Representations with an Omnidirectional Camera", *IEEE Transactions on Robotics and Automation*, vol. 16, n°6, Dec.2000.

Gaspar et al. (2002). Gaspar, J.; Decco, C. C. G.; Okamoto, J.; Santos-Victor, J., "Constant Resolution Omnidictional Cameras", *OMNIVIS'02 Workshop on Omni-directional Vision*, Copenhagen, Denmark, June 2002

Geyer e Daniilidis (2000). C. Geyer and K. Daniilidis, "A unifying theory for central panoramic systems and practical applications", In *Proc. European Conf. on Computer Vision* - ECCV'00, pg. 445-461, Dublin, Ireland, 2000.

Giachetti et al. (1998). Giachetti, A.; Campani, M.; Torre, V., "The Use of Optical Flow for Road Navigation", *IEEE Transactions on Robotics and Automation*, vol. 14, nº1, pp.34-48, 1998.

Goldberg et al. (2002). Goldberg, S. B.; Maimone, M. W.; Matthies, L., "Stereo Vision and robber navigation software for planetary exploration", *IEEE Aerospace Conference Proceedings*, vol. 5, pp. 2025-2036, March 2002.

Hicks e Bajcsy (1999a). R. A. Hicks and R. Bajcsy,. "Catadioptric sensors that approximate wide-angle perspective projections", In *Proc. of the IEEE 2nd Workshop on Omnidirectional Vision - OMNIVIS'00*, pg: 97-103, June 2000.

Horn (1987). Horn, B. K. P., "Closed-form solution of absolute orientation using unit Quaternions", *Journal of the Optical Society of America A 4,* pp.629-642, 1987.

Horn et al. (1988). Horn, B. K. P.; Hilden, H. M.; Negahdaripour, S., "Closed-form solution of absolute orientation using orthonormal matrices", *Journal of the Optical Society of America A 5*, pp.1127-1135, 1988.

Koyasu et al. (2002). Koyasu, H.; Miura, J.; Shirai, Y., "Recognizing Moving Obstacles for Robot Navigation Using Real-Time Omnidirectional Stereo Vision", *Journal of Robotics and Mechatronics*, vol. 14, nº2, pp.147-156, 2002.

Kumano et al. (2000). Kumano, M.; Ohya, A.; Yuta, S., "Obstacle Avoidance of Autonomous Mobile Robot using Stereo Vision Sensor". *Proceedings of the 2º International Symposium on Robotics and Automation*, pp.297-502, Nov. 2000.

Little e Murray (1998). Little, J. J.; Murray, D. R., "Selecting Stable Image Features for Robots Location Using Stereo", *Proceeding of the IEEE Conference on Robotics and Systems*, IROS`98, 1998.

Manessis (2003). Manessis, A., "Overview of Related work on Scene Modelling". http://www.dai.ed.ac.uk/Cvonline/LOCAL_COPIES/MANESSIS/liter.

Matsumoto et al. (1999). Matsumoto, Y.; Ikeda, K.; Inaba, M.; Inoue, H., "Visual Navigation using Omnidirectional View Sequence", *Proceeding of the IEEE International Conference on Intelligent Robots and Systems*, pp. 317-322, 1999.

Minguez et al. (2001). Minguez, J.; Montano, L.; Simeon, T.; Alami, R., "Global Nearness Diagram Navigation (GND)", *IEEE International Conference on Robotics and Automation*, ICRA, 2001.

Murray e Jennings (1997). Murray, D.; Jennings, C., "Stereo Vision Mapping and Navigation for Mobile Robots", *Proceeding of the IEEE Conference on Robotics and Automation*, May 1997.

Murray e Little (1998). Murray, D.; Little, J., "Using Real-Time Stereo Vision for Mobile Robot Navigation". *Workshop on Perception for Mobile Agents at CVPR`98*, 1998.

Nakamura e Ishiguro (2002). Nakamura, T.; Ishiguro, H. "Automatic 2D Map Construction using a Special Catadioptric Sensor", *Proceeding od the IEEE Int. Conference on Intelligent Robots and Systems*, Oct. 2002.

Nene e Nayar (1998). S. A. Nene and S. K. Nayar, "Stereo with mirrors", In *Proc. of the 6th Int. Conf. on Computer Vision - ICCV'98*, Bombay, India, January 1998.

Ollis et al. (1999). M. Ollis, H. Herman and S. Singh, "Analysis and design of panoramic stereo vision using equi-angular cameras", *The Robotics Institute, Carnegie Mellon University*, CMU-RI-TR-99-04, January 1999.

Santos-Victor et al. (1995). Santos-Victor, J., "Visual Perception for Mobile Robots: From Percepts to Behaviours", Phd Thesis, Nov. 1994. Disponível em http://vislab.isr.ist.utl.pt/.

Santos-Victor e Sandine (1997). Santos-Victor, J.; Sandine, G., "Embedded Visual Behaviors for Navigation", *Robotics and Autonomous System*, 19, pp. 299-313, 1997.

Southwell et al. (1996). M. F. D. Southwell, A. Basu, and J. Reyda, "Panoramic stereo", In *Proc. of the Int. Conf. on Pattern Recognition - ICPR*, pp. 378-382, Vienna, August 1996.

Souza Jr. (2004). Souza Jr, J. C., "Visão Estéreo Omnidirecional utilizando espelho duplo de perfil hiperbólico", *tese de doutorado*, 2004.

Svoboda et al. (1998). Svoboda, T.; Pajdla, T.; Hlavác, V., "Epipolar Geometry for Panoramic Cameras", *Computer Vision ECCV`98*, Freiburg, Germany, July 1998.

Walker et al. (1991). Walker, M. W.; Shao, L.; Volz, R. A., "Estimating 3-D location parameters using dual number quaternions", *CVGIP: Image Understanding 54*, pp.358-367, 1991.

Yagi (1999). Y. Yagi, "Omnidirectional sensing and its applications", *IEICE Trans. Inf. & Syst.*, Vol. E82-D, n.3, March 1999.

Yagi et al. (1995). Yagi, Y.; Nishizawwa, Y.; Yachida, M., "Map-Based Navigation for Mobile Robot with Omnidirectional Image Sensor COPIS", *IEEE Transactions on Robotics and Automation*, vol.11, n.5,Oct. 1995.

Yamazawa et al. (1999). K. Yamazawa, Y. Yagi, and M. Yachida, "Omnidirectional imaging with hyperboloidal projection", In *Proc. of the IEEE/RSJ Int. Conf. Intell. Robots and Systems*, July 1999.

ANEXO A

PROGRAMAS COMPUTACIONAIS

Esse anexo apresenta os programas que implementam os algoritmos desenvolvidos nesse trabalho usando MatLab versão 6.5.0.180913a r13.

A.1 Algoritmo do gerador de pontos implementado no MatLab

Este programa implementado no MatLab, gera randomicamente um conjunto de pontos no espaço tridimensional. A partir deste espaço gerado, o programa gera um subespaço, com uma dada rotação em torno do eixo z, translação nos eixos x e y e um certo grau de encobrimento (λ).

```
%Geradordepontos.m

clc;
close all;
fprintf('Entre com a rotaçao \n' );
rotacao=input('(default 0 graus): ' );
if(isempty(rotacao))
   rotacao=0;
end
fprintf('Entre com a translaçao em X \n' );
transX=input('(default 0): ' );
if(isempty(transX))
   transX=0;
end
fprintf('Entre com a translaçao em Y \n' );
transY=input('(default 0): ' );
if(isempty(transY))
   transY=0;
end
fprintf('Entre com o overlap \n' );
overlap=input('(default 1) max=1, min=0.5: ' );
if(isempty(overlap))
   overlap=1;
end
fprintf('Entre com o numero de pontos do modelo \n' );
```

```
tamanho=input('(default randomico): ' );
if(isempty(tamanho))
  tamanho=round(rand*50) %ate 50 pontos
end
while tamanho==0
  tamanho=round(rand*50)
end

M=zeros(tamanho,3); %matriz modelo
Am=rand(tamanho,1)*2*pi; %ângulo
Dm=rand(tamanho,1)*5000; %distâncias
Hm=rand(tamanho,1)*1000; %altura
M=[Am Dm Hm]; %modelo
[Xm,Ym,Zm] = pol2cart(Am,Dm,Hm); %coordenadas polares para cartesianas

%plota o mapa randomizado
figure(1)
plot3(Xm,Ym,Zm,'o')
grid on;
title('Representaçao no espaço');
xlabel('X');
ylabel('Y');
zlabel('Altura');
hold on;

exclui=round(tamanho*(1-overlap));

tamanho-exclui

if(exclui~=0)
%randomiza os pontos a serem excluidos
  vetor=zeros(exclui);
  for i=1:exclui
    flag=1;
    while flag==1
      flag=0;
      indice=round(rand*tamanho);
      for j=1:exclui
        if(vetor(j)==indice)
          flag=1;
        end
      end
    end
    vetor(i)=indice;
  end
  j=1;
  b=1;
```

```
  Ap=zeros(tamanho-exclui,1);
  Dp=zeros(tamanho-exclui,1);
  Hp=zeros(tamanho-exclui,1);

  Ae=zeros(exclui,1);
  De=zeros(exclui,1);
  He=zeros(exclui,1);

  %exclui os pontos escolhidos
  for i=1:tamanho
    flag=0;
    for a=1:exclui
      if(vetor(a)==i)
        flag=1;
      end
    end
    if flag==0
      Ap(j,1)=Am(i,1)+rotacao*2*pi/360;
      Dp(j,1)=Dm(i,1);
      Hp(j,1)=Hm(i,1);
      j=j+1;
    else
      Ae(b,1)=Am(i,1)+rotacao*2*pi/360;
      De(b,1)=Dm(i,1);
      He(b,1)=Hm(i,1);
      b=b+1;
    end
  end
else
  Ap=Am+rotacao*2*pi/360;
  Dp=Dm;
  Hp=Hm;
end

[Xp,Yp,Zp] = pol2cart(Ap,Dp,Hp);
Xp=Xp+transX;
Yp=Yp+transY;
[Ap,Dp,Hp]=cart2pol(Xp,Yp,Zp);
P=[Ap Dp Hp];
plot3(Xp,Yp,Zp,'k.')

%plota a amostra randomizada, transladada e rotacionada
if exclui > 0
  [Xe,Ye,Ze] = pol2cart(Ae,De,He);
  Xe=Xe+transX;
  Ye=Ye+transY;
  [Ae,De,He]=cart2pol(Xe,Ye,Ze);
  E=[Ae De He];
```

```
  plot3(Xe,Ye,Ze,'r.')
end

%salva os dados para serem utilizados no TrICP
save ('pontosgerados.mat' ,'M' ,'P', 'overlap')
```

A.2 Primeiro Algoritmo do Mapa de ambiente implementado no MatLab

Este algoritmo é composto de seis programas. Um destes programas, o de reconstrução tridimensional, foi desenvolvido por Souza Jr. (2004) em sua tese de doutorado e não será reproduzido aqui. O algoritmo implementado no MatLab segue a baixo.

```
%mapa.m
%Programa principal que faz a montagem do mapa de ambiente para uma serie de
%imagens obtidas pelo espelho duplo hiberbolico
clear;
clc;
imag = 0; %contador do numero de imagens
autolocalizacaoant = [0 0]; %autolocalizaçao na iteraçao anterior
direcaoant = 0; %direçao da iteraçao anterior
mapatemp2 = [0 0 0]; %mapa com filtragem dupla

%inicialmente deve-se colocar a primeira imagem obtida pelo robo, este
%serah o primeiro mapa.
fprintf('Entre o nome do arquivo de imagem (bmp) sem extensão, entre
"singlequotes"');
fprintf('para a primeira imagem do mapa sem o primeiro numero\n' );
imageminicial = input('(default Imagem): ' );
if (isempty(imageminicial))
   imageminicial = 'Imagem';
end

imag = imag+1;
temp = num2str(imag);
imagem = strcat(imageminicial,temp);
M = analisetotalmapa(imagem); % faz a reconstruçao tridimensional da imagem
M(:,1) = M(:,1)*pi/180;
[X,Y] = pol2cart(M(:,1),M(:,2));
Mapa = [X Y ones(length(M),1)];

flag = 0;
%inicia o looping principal do programa
while flag == 0
   imag = imag+1;
   temp = num2str(imag);
   imagem = strcat(imageminicial,temp);
   P = analisetotalmapa(imagem); % faz a reconstruçao tridimensional da imagem
   %montamapa eh uma funçao que realiza a montagem do mapa global
   [Mapa,mapatemp2,autolocalizacaoant,direcaoant,errorflag] =
   montamapa(Mapa,mapatemp2,P(:,1:2),autolocalizacaoant,direcaoant,imag);
   %controle das tentativas do TrICP
```

```
   if errorflag > 0
      fprintf('*************************************************************\n');
      fprintf('Na primeira tentativa a diferença relativa entre o erro quadratico \nmedio
            inicial e final foi superior a 50%% para a %s\n',imagem);
      fprintf('*************************************************************\n');
      if errorflag > 1
         fprintf('*************************************************************\n');
         fprintf('Na segunda tentativa a diferença relativa entre o erro quadratico
\nmedio
               inicial e final foi superior a 50%% para a %s\n',imagem);
         fprintf('*************************************************************\n');
         if errorflag > 2
            fprintf('*************************************************************\n');
            fprintf('Na terceira tentativa a diferença relativa entre o erro quadratico
                  \nmedio inicial e final foi superior a 50%% para a %s\n',imagem);
            fprintf('*************************************************************\n');
            if errorflag > 3
               fprintf('*************************************************************\n');
               fprintf('Na quarta tentativa a diferença relativa entre o erro quadratico
                     \nmedio inicial e final foi superior a 50%% para a %s\n',imagem);
               fprintf('*************************************************************\n');
               if errorflag > 4
                  fprintf('*************************************************************\n');
                  fprintf('Na quinta tentativa a diferença relativa entre o erro quadratico
                        \nmedio inicial e final foi superior a 50%% para a
%s\n',imagem);
                  fprintf('*************************************************************\n');
                  if errorflag > 5

fprintf('*************************************************************\n');
                     fprintf('Na sexta tentativa a diferença relativa entre o erro quadratico
                           \nmedio inicial e final foi superior a 50%% para a %s e a
                           mesma foi descartada\n',imagem);

fprintf('*************************************************************\n');
                  end
               end
            end
         end
      end
   end

   fprintf('deseja incluir mais uma amostra sim = 0, nao = 1?\n' );
   fprintf('default (sim = 0)' );
   flagt = input(': ' );
   if (isempty(flagt))
      flagt = 0;
   end
```

```
    flag = flagt;

end
%fim do looping principal
close all
%fim de mapa.m

%montamapa.m
function [Mapa,mapatemp2,autolocalizacaoant,direcaoant,errorflag] =
montamapa2(Mapa,mapatemp2,P,autolocalizacaoant,direcaoant,imag)

M=Mapa;
Np=length(P);
Nm=length(M);

Nmapa=length(Mapa);
Mapatemp=Mapa;
errorflag = 0; %variavel que controla o numero de tentativas do TrICP
testef = 0

Nmapa=length(Mapatemp);
[A,D]=cart2pol(Mapatemp(:,1),Mapatemp(:,2));
M=[A D Mapatemp(:,3)];
M(:,1)=round(M(:,1)*180/pi);
for i=1:Nmapa
    if M(i,1)<0
        M(i,1)=360+M(i,1);
    end
end
%ordena os conjuntos de pontos segundo o incremento angular
M=sortrows(M);
P=sortrows(P);
%funcao que seleciona os pontos mais proximos ao centro da amostra
[M,Pf]=analisa2(M,P);

%iniciam-se as tentativas de correspondencia
fprintf('TrICP\n');

[translacaopura,rotacaototal,translacao,teta,iteracao,Ptrans,Srel]=TrICPotmapa(M,Pf)
;

P(:,1)=P(:,1)*pi/180;
[X,Y]=pol2cart(P(:,1),P(:,2));
P=[X Y];
Pfirst=P;
Pfirst(:,1)=Pfirst(:,1)+translacaopura(1);
Pfirst(:,2)=Pfirst(:,2)+translacaopura(2);
Ptrans=Pfirst;
```

```
if Srel < 50 | abs(rotacaototal*180/pi) > 5

[translacaopura,rotacaototal,translacao,teta,iteracao,Ptrans,Srel]=TrICPotmapa(Pf,M)
;
   translacao=-translacao;
   teta=-teta;
   translacaopura=-translacaopura;
   rotacaototal=-rotacaototal;
   errorflag=1;
   Pfirst=P;
   Pfirst(:,1)=Pfirst(:,1)+translacaopura(1);
   Pfirst(:,2)=Pfirst(:,2)+translacaopura(2);
   Ptrans=Pfirst;
end

if Srel < 50 | abs(rotacaototal*180/pi) > 5
   errorflag=2;


[translacaopura,rotacaototal,translacao,teta,iteracao,Ptrans,Srel]=TrICPotmapa(M,P);
   Pfirst=P;
   Pfirst(:,1)=Pfirst(:,1)+translacaopura(1);
   Pfirst(:,2)=Pfirst(:,2)+translacaopura(2);
   Ptrans=Pfirst;
end

if Srel < 50 | abs(rotacaototal*180/pi) > 5
   errorflag=3;


[translacaopura,rotacaototal,translacao,teta,iteracao,Ptrans,Srel]=TrICPotmapa(P,M);
   translacao=-translacao;
   teta=-teta;
   translacaopura=-translacaopura;
   rotacaototal=-rotacaototal;

   Pfirst=P;
   Pfirst(:,1)=Pfirst(:,1)+translacaopura(1);
   Pfirst(:,2)=Pfirst(:,2)+translacaopura(2);
   Ptrans=Pfirst;
end

if Srel < 50 | abs(rotacaototal*180/pi) > 5
   errorflag=4;
   [translacaopura,rotacaototal,translacao,teta,iteracao,Ptrans,Srel] =
   TrICPotmapa(Mapa,P);
   Pfirst=P;
```

```
    Pfirst(:,1)=Pfirst(:,1)+translacaopura(1);
    Pfirst(:,2)=Pfirst(:,2)+translacaopura(2);
    Ptrans=Pfirst;

end

if Srel < 50 | abs(rotacaototal*180/pi) > 5
    errorflag=5;

    [translacaopura,rotacaototal,translacao,teta,iteracao,Ptrans,Srel] =
    TrICPotmapa(P,Mapa);
    translacao=-translacao;
    teta=-teta;
    translacaopura=-translacaopura;
    rotacaototal=-rotacaototal;
    Pfirst=P;
    Pfirst(:,1)=Pfirst(:,1)+translacaopura(1);
    Pfirst(:,2)=Pfirst(:,2)+translacaopura(2);
    Ptrans=Pfirst;
end

if Srel < 50 | abs(rotacaototal*180/pi) > 5
    errorflag=6;
end

if errorflag ~= 6

    autolocalizacao=autolocalizacaoant+translacaopura;
    direcao=direcaoant+rotacaototal;

    %inicia o processo de fusao dos pontos
    fprintf('fussao do pontos \n');
    if testef==0
        Nptrans=length(Ptrans);
        it=0;
        for ip=1:Nptrans
            dq=Ptrans(ip,1)^2+Ptrans(ip,2)^2;
            if dq <=1e6
                erro=50;
            end
            if dq>1e6 & dq<=4e6
                erro=75;
            end
            if dq>4e6 & dq<=9e6
                erro=100;
            end
            if dq>9e6 & dq<=12e6
                erro=125;
```

```
        end
        if dq>12e6
           erro=150;
        end

        ii=0;
        buscaXmax=Ptrans(ip,1)+erro;
        buscaXmin=Ptrans(ip,1)-erro;
        buscaYmax=Ptrans(ip,2)+erro;
        buscaYmin=Ptrans(ip,2)-erro;
        for ipm=1:Nmapa
           if ((Mapa(ipm,1) < buscaXmax) & (Mapa(ipm,1) > buscaXmin) &
(Mapa(ipm,2) < buscaYmax) & (Mapa(ipm,2) > buscaYmin))
              ii=ii+1;
              it=it+1;
              vetorindiceb(ii)=ipm; %vetor das linhas a serem excluidas a cada
iteraçao
              vetorindicebt(it)=ipm; %vetor das linhas totais a serem excluidas
           end
        end
        media=[Ptrans(ip,1) Ptrans(ip,2) 1];
        if ii > 0
           for iexc=1:ii
              media=media+[Mapa(vetorindiceb(iexc),1) Mapa(vetorindiceb(iexc),2)
                           Mapa(vetorindiceb(iexc),3)];
           end
           media=[media(1)/(ii+1) media(2)/(ii+1) media(3)];
           Mapa(length(Mapa)+1,:)=media;
        else
           Mapa(length(Mapa)+1,:)=media;
        end
     end

     %reconstruçao do mapa

     fprintf('Reconstruçao dos mapas \n');
     for iinfinito=1:it
        Mapa(vetorindicebt(iinfinito),1)=inf;
     end
     Nmapa=length(Mapa);
     cont=0;
     for iexc=1:Nmapa
        if Mapa(iexc,1)~=inf
           cont=cont+1;
           novoMapa(cont,:)=Mapa(iexc,:);%alterado
        end
     end
  end
```

```
    %fim de testef==0

    %nesta representaçao o mapa atual esta nas coordenadas do mapa anterior
    clf;
    figure(2)
    grid on;
    title('Representaçao no espaço');
    xlabel('X');
    ylabel('Y');
    hold on;
    plot(Mapa(:,1),Mapa(:,2),'b.')


    if testef==0
       novoMapa(:,1)=novoMapa(:,1)-translacaopura(1);
       novoMapa(:,2)=novoMapa(:,2)-translacaopura(2);
        Mapa=novoMapa;
     end

     %agora o mapa atual esta nas coordenadas atuais

%filtragem
Nmapa=length(Mapa);
eclud=75;
for pf=1:Nmapa
   ii=0;
   buscaXmax=Mapa(pf,1)+eclud;
   buscaXmin=Mapa(pf,1)-eclud;
   buscaYmax=Mapa(pf,2)+eclud;
   buscaYmin=Mapa(pf,2)-eclud;
   for ipm=1:Nmapa
      if ((Mapa(ipm,1) < buscaXmax) & (Mapa(ipm,1) > buscaXmin) & (Mapa(ipm,2) <
buscaYmax) & (Mapa(ipm,2) > buscaYmin))
         ii=ii+1;
         vetor(ii)=ipm;
      end
   end
   if ii <= 1
      for i=1:ii
         Mapa(vetor(i),1)=inf;
      end
   end
end

cont=0;
for iexc=1:Nmapa
   if Mapa(iexc,1)~=inf
      cont=cont+1;
```

```
      novoM(cont,:)=Mapa(iexc,:);
   end
end
Mapa=novoM;

 limite=1/imag*100;
   %funçao q mostra o mapa global de diferentes formas e niveis de filtro

[mapatemp,mapatemp1,mapatemp2,porcentagem]=mostramapa(Mapa,imag,limite);

   Mapa=[mapatemp(:,1) mapatemp(:,2) mapatemp(:,4)];
   autolocalizacaoant=autolocalizacao;
   direcaoant=direcao;
end
return;
%fim de montamapa.m

%analisa2.m
%funçao que seleciona os pontos mais proximos ao centro do conjunto com
incremento %angular de 1 grau
function [M,P]=analisa2(M,P);
close all
Nm=length(M);
Np=length(P);

maxM=10000;
maxP=10000;

i=0;
j=1;
for t=0:359
   i=i+1;
   if i>Nm
      break;
   end
   if (M(i,1)==t)
      if  M(i,2)<maxM
         Mf(j,:)=M(i,:);
         j=j+1;
      end
      i=i+1;
      if i>Nm
         break;
      end
      while M(i,1)==t
         i=i+1;
         if i>Nm
            break;
```

```
        end
      end
    end
    if i>Nm
      break;
    end
    i=i-1;
end

i=0;
j=1;
for t=0:359
    i=i+1;
    if i>Np
      break;
    end
    if (P(i,1)==t)
      if  P(i,2)<maxP
        Pf(j,:)=P(i,:);
        j=j+1;
      end
      i=i+1;
      if i>Np
        break;
      end
      while P(i,1)==t
        i=i+1;
        if i>Np
          break;
        end
      end
    end
    if i>Np
      break;
    end
    i=i-1;
end

Mf(:,1)=Mf(:,1)*pi/180;
Pf(:,1)=Pf(:,1)*pi/180;

figure(1)
[X,Y]=pol2cart(Mf(:,1),Mf(:,2));
M=[X Y Mf(:,3)];
plot(X,Y,'o')
grid
hold
[X,Y]=pol2cart(Pf(:,1),Pf(:,2));
```

```
P=[X Y];
plot(X,Y,'k.')
pause
return;
%fim de analisa2.m

%TrICPotmapa
%funçao que faz a correspondencia entre as imagens
function [translacaopura,rotacaototal,translacao,teta,iteracao,Ptrans,Srel] =
TrICPotmapa(M,P)

close all;
janela=500;

%carrega o modelo
Nm=length(M);
%modelo em X Y

%carrega a amostra
Np=length(P);
Pfirst=P;
%amostra em X Y

N=max(Np,Nm);

if N < 500
   itlim=50;
end
if N >= 500 & Np < 1000
   itlim=40;
end
if N >= 1000
   itlim=30;
end
S=1E10;
parada(1,:)=[S S/Np];
Santerior=1E10;
flag=0;
iteracao=0;
translacaototal=[0 0];
rotacaototal=0;
ming=[0 1E10 1E10];

while flag==0

   iteracao=iteracao+1;
   janela=510-.5*iteracao;
   for p=1:Np %passo 1
```

```
        im=0; %indice do modelo
        while im == 0
            maxvertical=P(p,2)+janela;
            minvertical=P(p,2)-janela;
            maxhorizontal=P(p,1)+janela;
            minhorizontal=P(p,1)-janela;
            for m=1:Nm  %monta um vetor com os indices do modelo q se encontram
dentro do quadrande de busca
                if (M(m,1) < maxhorizontal) & (M(m,1) > minhorizontal) &
                  (M(m,2) < maxvertical) & (M(m,2) > minvertical)
                   im=im+1;
                   vetorindice(im)=m;
                end
            end
            if im > 0
                distquad=0;
                for idist=1:im %calcula as distancias a todos os pontos dentro do quadrado
                   distquad(idist)=(P(p,1)-M(vetorindice(idist),1))^2 +
                                   (P(p,2)-M(vetorindice(idist),2))^2;
                end

                [temp,itemp]=min(distquad); %descobre o ponto mais proximo do modelo
                dquadrado(p,1)=vetorindice(itemp); %indice da menor distancia do modelo
ao ponto
                dquadrado(p,2)=temp; %valor da distancia ao quadrado
                dquadrado(p,3)=p; %indice da amostra
            else
                janela=janela + 10;
            end
        end
    end

    %passo 2
    crescentemp=sortrows(dquadrado,2);  %ordena das distancias em ordem
crescente
    Stemp=sum(crescentemp(:,2)); %calcula a soma das distancias ao quadrado

    %passo 3, condiçao de parada
    parada(iteracao+1,:)=[Stemp Stemp/Np];
    if (iteracao > itlim) | (parada(iteracao+1,2) < 1E-10) | (abs(parada(iteracao+1,2)-
parada(iteracao,2)) < 10)
        flag=1;
        break;
    else
        if          (parada(iteracao+1,2)-parada(iteracao,2)          <          0)          &
(ming(3)>parada(iteracao+1,2))
            ming=[iteracao Stemp parada(iteracao+1,2)];
        end
```

```
   S=Stemp;
end

%passo 4

%media dos pontos casados
Ntemp=length(crescentemp);
xsoma=0;
ysoma=0;
for media=1:Ntemp
   xsoma=xsoma+M(crescentemp(media,1),1);
   ysoma=ysoma+M(crescentemp(media,1),2);
end
mediaMc(1)=xsoma/Ntemp;
mediaMc(2)=ysoma/Ntemp;

xsoma=0;
ysoma=0;
for media=1:Ntemp
   xsoma=xsoma+P(crescentemp(media,3),1);
   ysoma=ysoma+P(crescentemp(media,3),2);
end
mediaPc(1)=xsoma/Ntemp;
mediaPc(2)=ysoma/Ntemp;

%os dois mapas relativos as suas centroides
Mmc(:,1)=M(:,1)-mediaMc(1);
Mmc(:,2)=M(:,2)-mediaMc(2);

Pmc(:,1)=P(:,1)-mediaPc(1);
Pmc(:,2)=P(:,2)-mediaPc(2);

%rotaçao, conciderando pontos coplanares
Se=0;
C=0;
for icres=1:Np
   Se=Se+(Mmc(crescentemp(icres,1),1)*Pmc(crescentemp(icres,3),2)-
            Mmc(crescentemp(icres,1),2)*Pmc(crescentemp(icres,3),1));
   C=C+(Mmc(crescentemp(icres,1),1)*Pmc(crescentemp(icres,3),1)+
         Mmc(crescentemp(icres,1),2)*Pmc(crescentemp(icres,3),2));
end
sinteta=Se/(sqrt(Se^2+C^2));
costeta=C/(sqrt(Se^2+C^2));
teta(iteracao,1)=acos(sqrt((1+costeta)/2))*2;
teta(iteracao,2)=asin(sinteta/sqrt((1+costeta)*2))*2;

%escala, conciderando o erro maior na amostra q no modelo
```

```
    %rotacionando a amostra media
    [A,D]=cart2pol(Pmc(:,1),Pmc(:,2));
    A=A-teta(iteracao,2);
    [X Y]=pol2cart(A,D);
    Pmr=[X Y];

    den=0;
    num=0;
    for i=1:Np
       num=num+(Mmc(crescentemp(i,1),1)*Pmr(crescentemp(i,3),1)+
                Mmc(crescentemp(i,1),2)*Pmr(crescentemp(i,3),2));
       den=den+(Pmc(crescentemp(i,3),1)*Pmc(crescentemp(i,3),1)+
                Pmc(crescentemp(i,3),2)*Pmc(crescentemp(i,3),2));
    end
    scale=num/den;

    %translaçao

    %rotacionando a media da amostra
    [A,D]=cart2pol(mediaPc(1),mediaPc(2));
    A=A-teta(iteracao,2);
    [X,Y]=pol2cart(A,D);
    mediaPcr=[X Y];

    translacao(iteracao,1)=mediaMc(1)-scale*mediaPcr(1);
    translacao(iteracao,2)=mediaMc(2)-scale*mediaPcr(2);

    %passo 5
    %aplicando a rotaçao e a translaçao a amostra

    P(:,1)=P(:,1)+translacao(iteracao,1);
    P(:,2)=P(:,2)+translacao(iteracao,2);

    [A,D]=cart2pol(P(:,1),P(:,2));
    A=A-teta(iteracao,2);
    [X,Y]=pol2cart(A,D);
    P=[X Y];

    translacaototal=translacaototal + [translacao(iteracao,1) translacao(iteracao,2)];
    rotacaototal=rotacaototal + teta(iteracao,2);
    Santerior=S;

end

iteracao-1
Plast=Pfirst(1,:);
figure(1)
plot(M(:,1),M(:,2),'o')
```

```
grid on;
title('Representaçao no espaço');
xlabel('X');
ylabel('Y');
hold;

if (parada(iteracao,2)>ming(3)*1.1)
   iteracao=ming(1)
   ming(2)
   ming(3)
   fprintf('mudança relativa percentual: ')
   Srel=-(ming(2)-parada(2,1))/parada(2,1)*100
   translacaototal=[0 0];
   rotacaototal=0;

   for it=1:ming(1)
      Pfirst(:,1)=Pfirst(:,1)+translacao(it,1);
      Pfirst(:,2)=Pfirst(:,2)+translacao(it,2);

      [A,D]=cart2pol(Pfirst(:,1),Pfirst(:,2));
      A=A-teta(it,2);
      [X,Y]=pol2cart(A,D);
      Pfirst=[X Y];
      translacaototal=translacaototal + [translacao(it,1) translacao(it,2)];
      rotacaototal=rotacaototal + teta(it,2);
   end
   plot(Pfirst(:,1),Pfirst(:,2),'k.')
   Ptrans=Pfirst;
else
   parada(iteracao,1)
   parada(iteracao,2)
   fprintf('mudança relativa percentual: ')
   Srel=-(parada(iteracao,1)-parada(2,1))/parada(2,1)*100

   for it=1:(iteracao-1)
      Pfirst(1,1)=Pfirst(1,1)+translacao(it,1);
      Pfirst(1,2)=Pfirst(1,2)+translacao(it,2);
      [A,D]=cart2pol(Pfirst(1,1),Pfirst(1,2));
      A=A-teta(it,2);
      [X,Y]=pol2cart(A,D);
      Pfirst=[X Y];
   end
   plot(P(:,1),P(:,2),'k.')
   Ptrans=P;
   iteracao=iteracao-1;
end

[A,D]=cart2pol(Plast(1),Plast(2));
```

```
A=A-rotacaototal;
[X,Y]=pol2cart(A,D);
Plast=[X Y];

translacaopura=Pfirst(1,:)-Plast
translacaototal
rotacaototal*180/pi
pause
return;
%fim de TrICPotmapa

%mostramapa.m
%funçao q mostra o mapa global com diferentes niveis de filtragem
function [mapatemp,mapatemp1,mapatemp2,porcentagem] =
mostramapa(Mapa,imag,limite)

figure(3)
scatter(Mapa(:,1),Mapa(:,2),15,Mapa(:,3),'filled')
colorbar;
grid on;
title('Mapa com numero de pontos vizinhos e colapsados');
xlabel('X');
ylabel('Y');

Nm=length(Mapa);
porcentagem=zeros(Nm,1);
max=Mapa(1,3);
for i=2:Nm
   if Mapa(i,3)>max
      max=Mapa(i,3);
   end
end
for i=1:Nm
   porcentagem(i,1)=(1-(max-Mapa(i,3))/(max*imag))*100;
end
figure(4)
scatter(Mapa(:,1),Mapa(:,2),15,porcentagem,'filled')
colorbar;
grid on;
title('Mapa com numero de pontos vizinhos e colapsados em porcentagem');
xlabel('X');
ylabel('Y');

figure(5)
hold on;
ipor=0;
for i=1:Nm
   if porcentagem(i,1)>=limite
```

```
        ipor=ipor+1;
        mapatemp(ipor,:)=[Mapa(i,1) Mapa(i,2) porcentagem(i) Mapa(i,3)];
    end
end
scatter(mapatemp(:,1),mapatemp(:,2),15,mapatemp(:,3),'filled')
colorbar;
grid on;
temp=num2str(limite);
temp=strcat('Mapa com porcentagem minima =',temp,' porcento');
title(temp);
xlabel('X');
ylabel('Y');

figure(6)
hold on;
ipor=0;
Nmt=length(mapatemp);
for i=1:Nmt
    if mapatemp(i,4) > 1
        ipor=ipor+1;
        mapatemp1(ipor,:)=[mapatemp(i,1) mapatemp(i,2) mapatemp(i,3)];
    end
end
scatter(mapatemp1(:,1),mapatemp1(:,2),15,mapatemp1(:,3),'filled')
colorbar;
grid on;
temp=num2str(limite);
temp =
strcat('Mapa com porcentagem minima =',temp,' porcento e pelo menos 2 pontos');
title(temp);
xlabel('X');
ylabel('Y');

figure(7)
hold on;
ipor=0;
Nmt=length(Mapa);
for i=1:Nmt
    if Mapa(i,3) > 1
        ipor=ipor+1;
        Mapatemp(ipor,:)=[Mapa(i,1) Mapa(i,2) Mapa(i,3)];
    end
end
mapatemp2=Mapatemp;

scatter(mapatemp2(:,1),mapatemp2(:,2),15,mapatemp2(:,3),'filled')
colorbar;
grid on;
```

```
temp=strcat('Mapa com pelo menos 2 pontos');
title(temp);
xlabel('X');
ylabel('Y');
return;
%fim de mostramapa.m
```

A.3 Segundo Algoritmo do Mapa de ambiente implementado no MatLab

Este algoritmo é composto de cinco programas. Um destes programas, o de reconstrução tridimensional, foi desenvolvido por Souza Jr. (2004) em sua tese de doutorado e não será reproduzido aqui. O algoritmo implementado no MatLab segue a baixo.

```
%mapa.m
%faz a montagem do mapa de ambiente para uma serie de imagens obtidas
%pelo espelho duplo hiberbolico.
clear;
clc;
imag=0;
autolocalizacaoant=[0 0];
direcaoant=0;
verrorflag(1)=0;
Srelim=50;

%coordenads cartesianas
ultP=[0 0];
ultPf=[0 0];

penP=[0 0];
penPf=[0 0];

antP=[0 0];
antPf=[0 0];

Mapa=[0 0 0];

%inicialmente deve-se colocar a primeira imagem obtida pelo robo, este
%serah o primeiro mapa.
fprintf('Entre o nome do arquivo de imagem (bmp) sem extensão, entre
"singlequotes"');
fprintf('para a primeira imagem do mapa sem o primeiro numero\n' );
imageminicial=input('(default Imagem): ' );
if(isempty(imageminicial))
   imageminicial='Imagem';
end

imag=imag+1;
temp=num2str(imag);
imagem=strcat(imageminicial,temp);
M=analisetotalmapa(imagem);
M=sortrows(M(:,1:2));
```

```
ultPf=analisa(M);
M(:,1)=M(:,1)*pi/180;
[X,Y]=pol2cart(M(:,1),M(:,2));
ultP=[X Y];
Mapa=[X Y 2*ones(length(M),1)];
flag=0;
while flag==0
   imag=imag+1;
   temp=num2str(imag);
   imagem=strcat(imageminicial,temp);
   Pt=analisetotalmapa(imagem);
   P=Pt(:,1:2);
   P=sortrows(P);

   %desloca as amostras de modo a descartar a antP
   antP=penP;
   antPf=penPf;

   penP=ultP;
   penPf=ultPf;

   ultP=P;
   ultPf=analisa(ultP);

   P(:,1)=P(:,1)*pi/180;
   [X,Y]=pol2cart(P(:,1),P(:,2));
   ultP=[X Y];

   %inicio de montamapa
   Nultp=length(ultP);
   Npenp=length(penP);
   Nantp=length(antP);
   Nmapa=length(Mapa);

   errorflag=0;

   %poem o mapa em coordenadas polares em graus
   [A,D]=cart2pol(Mapa(:,1),Mapa(:,2));
   M=[A D Mapa(:,3)];
   M(:,1)=round(M(:,1)*180/pi);
   for i=1:Nmapa
      if M(i,1)<0
         M(i,1)=360+M(i,1);
      end
   end
   M=sortrows(M);
   M=analisa(M);
```

```
%TrICP%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% M ultPf
figure(1)
clf
plot(M(:,1),M(:,2),'o')
hold
plot(ultPf(:,1),ultPf(:,2),'k.')
grid
pause

verrorflag(imag)=0;

[translacaopura,rotacaototal,translacao,teta,iteracao,Ptrans,Srel]=TrICPotmapa(M,ult
Pf);

Pfirst=ultP;
Pfirst(:,1)=Pfirst(:,1)+translacaopura(1);
Pfirst(:,2)=Pfirst(:,2)+translacaopura(2);
Ptrans=Pfirst;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if Srel < (Srelim) | abs(rotacaototal*180/pi) > 5
   verrorflag(imag)=1;

   figure(1)
   clf
   plot(ultPf(:,1),ultPf(:,2),'o')
   hold
   plot(M(:,1),M(:,2),'k.')
   grid
   pause

[translacaopura,rotacaototal,translacao,teta,iteracao,Ptrans,Srel]=TrICPotmapa(ultPf,
M);
   translacao=-translacao;
   teta=-teta;
   translacaopura=-translacaopura;
   rotacaototal=-rotacaototal;

   Pfirst=ultP;
   Pfirst(:,1)=Pfirst(:,1)+translacaopura(1);
   Pfirst(:,2)=Pfirst(:,2)+translacaopura(2);
   Ptrans=Pfirst;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Mapa ultP
if (Srel < (Srelim) | abs(rotacaototal*180/pi) > 5) & verrorflag(imag)==1
```

```
      verrorflag(imag)=2;

      figure(1)
      clf
      plot(Mapa(:,1),Mapa(:,2),'o')
      hold
      plot(ultP(:,1),ultP(:,2),'k.')
      grid
      pause

[translacaopura,rotacaototal,translacao,teta,iteracao,Ptrans,Srel]=TrICPotmapa(Mapa
,ultP);

      Pfirst=ultP;
      Pfirst(:,1)=Pfirst(:,1)+translacaopura(1);
      Pfirst(:,2)=Pfirst(:,2)+translacaopura(2);
      Ptrans=Pfirst;
   end
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   if (Srel < Srelim | abs(rotacaototal*180/pi) > 5) & verrorflag(imag)==2
      verrorflag(imag)=3;

      figure(1)
      clf
      plot(ultP(:,1),ultP(:,2),'o')
      hold
      plot(Mapa(:,1),Mapa(:,2),'k.')
      grid
      pause

[translacaopura,rotacaototal,translacao,teta,iteracao,Ptrans,Srel]=TrICPotmapa(ultP,
Mapa);
      translacao=-translacao;
      teta=-teta;
      translacaopura=-translacaopura;
      rotacaototal=-rotacaototal;

      Pfirst=ultP;
      Pfirst(:,1)=Pfirst(:,1)+translacaopura(1);
      Pfirst(:,2)=Pfirst(:,2)+translacaopura(2);
      Ptrans=Pfirst;
   end
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% penPf ultPf
   if (Srel < (Srelim) | abs(rotacaototal*180/pi) > 5) & verrorflag(imag)==3
```

```
        verrorflag(imag)=4;

        figure(1)
        clf
        plot(penPf(:,1),penPf(:,2),'o')
        hold
        plot(ultPf(:,1),ultPf(:,2),'k.')
        grid
        pause

[translacaopura,rotacaototal,translacao,teta,iteracao,Ptrans,Srel]=TrlCPotmapa(penP
f,ultPf);

        Pfirst=ultP;
        Pfirst(:,1)=Pfirst(:,1)+translacaopura(1);
        Pfirst(:,2)=Pfirst(:,2)+translacaopura(2);
        Ptrans=Pfirst;
    end
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    if (Srel < (Srelim) | abs(rotacaototal*180/pi) > 5) & verrorflag(imag)==4
        verrorflag(imag)=5;

        figure(1)
        clf
        plot(ultPf(:,1),ultPf(:,2),'o')
        hold
        plot(penPf(:,1),penPf(:,2),'k.')
        grid
        pause

[translacaopura,rotacaototal,translacao,teta,iteracao,Ptrans,Srel]=TrlCPotmapa(ultPf,
penPf);
        translacao=-translacao;
        teta=-teta;
        translacaopura=-translacaopura;
        rotacaototal=-rotacaototal;

        Pfirst=ultP;
        Pfirst(:,1)=Pfirst(:,1)+translacaopura(1);
        Pfirst(:,2)=Pfirst(:,2)+translacaopura(2);
        Ptrans=Pfirst;
    end
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% penP ultP
    if (Srel < (Srelim) | abs(rotacaototal*180/pi) > 5) & verrorflag(imag)==5
```

```
      verrorflag(imag)=6;

      figure(1)
      clf
      plot(penP(:,1),penP(:,2),'o')
      hold
      plot(ultP(:,1),ultP(:,2),'k.')
      grid
      pause

[translacaopura,rotacaototal,translacao,teta,iteracao,Ptrans,Srel]=TrICPotmapa(penP
,ultP);

      Pfirst=ultP;
      Pfirst(:,1)=Pfirst(:,1)+translacaopura(1);
      Pfirst(:,2)=Pfirst(:,2)+translacaopura(2);
      Ptrans=Pfirst;
   end
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   if (Srel < Srelim | abs(rotacaototal*180/pi) > 5) & verrorflag(imag)==6
      verrorflag(imag)=7;

      figure(1)
      clf
      plot(ultP(:,1),ultP(:,2),'o')
      hold
      plot(penP(:,1),penP(:,2),'k.')
      grid
      pause

[translacaopura,rotacaototal,translacao,teta,iteracao,Ptrans,Srel]=TrICPotmapa(ultP,p
enP);

      translacao=-translacao;
      teta=-teta;
      translacaopura=-translacaopura;
      rotacaototal=-rotacaototal;
      Pfirst=ultP;

      Pfirst(:,1)=Pfirst(:,1)+translacaopura(1);
      Pfirst(:,2)=Pfirst(:,2)+translacaopura(2);
      Ptrans=Pfirst;
   end
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% antP ultP
if ((Srel < Srelim | abs(rotacaototal*180/pi) > 5) & (imag > 2)) & verrorflag(imag)==7
   verrorflag(imag)=8;

   figure(1)
   clf
   plot(antP(:,1),antP(:,2),'o')
   hold
   plot(ultP(:,1),ultP(:,2),'k.')
   grid
   pause

[translacaopura,rotacaototal,translacao,teta,iteracao,Ptrans,Srel]=TrICPotmapa(antP,
ultP);

   Pfirst=ultP;
   translacaopura=translacaopura-autolocalizacaoant(imag-1,:);
   Pfirst(:,1)=Pfirst(:,1)+translacaopura(1);
   Pfirst(:,2)=Pfirst(:,2)+translacaopura(2);
   Ptrans=Pfirst;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if ((Srel < Srelim | abs(rotacaototal*180/pi) > 5) & (imag > 2)) & verrorflag(imag)==8
   verrorflag(imag)=9;

   figure(1)
   clf
   plot(ultP(:,1),ultP(:,2),'o')
   hold
   plot(antP(:,1),antP(:,2),'k.')
   grid
   pause


[translacaopura,rotacaototal,translacao,teta,iteracao,Ptrans,Srel]=TrICPotmapa(ultP,a
ntP);
   translacao=-translacao;
   teta=-teta;
   translacaopura=-translacaopura;
   rotacaototal=-rotacaototal;

   Pfirst=ultP;
   translacaopura=translacaopura-autolocalizacaoant(imag-1,:);
   Pfirst(:,1)=Pfirst(:,1)+translacaopura(1);
   Pfirst(:,2)=Pfirst(:,2)+translacaopura(2);
   Ptrans=Pfirst;
```

```
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if (Srel < Srelim | abs(rotacaototal*180/pi) > 5) & verrorflag(imag)==9
  verrorflag(imag)=10;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if errorflag ~= 10

  autolocalizacao=sum(autolocalizacaoant)+translacaopura;
  direcao=sum(direcaoant)+rotacaototal;

  %fussao do pontos

  Nptrans=length(Ptrans);
  it=0;
  for ip=1:Nptrans
    erro=50;
    ii=0;
    buscaXmax=Ptrans(ip,1)+erro;
    buscaXmin=Ptrans(ip,1)-erro;
    buscaYmax=Ptrans(ip,2)+erro;
    buscaYmin=Ptrans(ip,2)-erro;
    for ipm=1:Nmapa
      if((Mapa(ipm,1) < buscaXmax) & (Mapa(ipm,1) > buscaXmin) &
(Mapa(ipm,2) < buscaYmax) & (Mapa(ipm,2) > buscaYmin))
        ii=ii+1;
        it=it+1;
        vetorindiceb(ii)=ipm; %vetor das linhas a serem excluidas a cada
iteraçao
        vetorindicebt(it)=ipm; %vetor das linhas totais a serem excluidas
      end
    end
    media=[Ptrans(ip,1) Ptrans(ip,2) 1];
    if ii > 0
      for iexc=1:ii
        media=media+[Mapa(vetorindiceb(iexc),1) Mapa(vetorindiceb(iexc),2)
Mapa(vetorindiceb(iexc),3)];
      end
      media=[media(1)/(ii+1) media(2)/(ii+1) media(3)];
      Mapa(length(Mapa)+1,:)=media;
    else
      Mapa(length(Mapa)+1,:)=media;
    end
  end
```

```
%reconstruçao do mapa

fprintf('Reconstruçao dos mapas \n');
for iinfinito=1:it
   Mapa(vetorindicebt(iinfinito),1)=inf;
end
Nmapa=length(Mapa);
cont=0;
for iexc=1:Nmapa
   if Mapa(iexc,1)~=inf
      cont=cont+1;
      novoMapa(cont,:)=Mapa(iexc,:);
   end
end

%nesta representaçao o mapa atual esta nas coordenadas do mapa anterior

clf;
figure(1)
grid on;
title('Representaçao no espaço');
xlabel('X');
ylabel('Y');
hold on;
plot(Mapa(:,1),Mapa(:,2),'b.')

novoMapa(:,1)=novoMapa(:,1)-translacaopura(1);
novoMapa(:,2)=novoMapa(:,2)-translacaopura(2);
Mapa=novoMapa;

%agora o mapa atual esta nas coordenadas atuais

Nmapa=length(Mapa);
eclud=75;
for pf=1:Nmapa
   ii=0;
   buscaXmax=Mapa(pf,1)+eclud;
   buscaXmin=Mapa(pf,1)-eclud;
   buscaYmax=Mapa(pf,2)+eclud;
   buscaYmin=Mapa(pf,2)-eclud;
   for ipm=1:Nmapa
      if((Mapa(ipm,1) < buscaXmax) & (Mapa(ipm,1) > buscaXmin) &
(Mapa(ipm,2) < buscaYmax) & (Mapa(ipm,2) > buscaYmin))
         ii=ii+1;
         vetor(ii)=ipm;
      end
   end
   if ii <= 1
```

```
            for i=1:ii
               Mapa(vetor(i),1)=inf;
            end
         end
      end

      cont=0;
      for iexc=1:Nmapa
         if Mapa(iexc,1)~=inf
            cont=cont+1;
            novoM(cont,:)=Mapa(iexc,:);
         end
      end
      Mapa=novoM;

      [mapatemp,Mapa]=mostramapa(Mapa,imag);

      autolocalizacaoant(imag,:)=translacaopura;
      direcaoant(imag,:)=rotacaototal;

   else  %fim do if errorflag
      autolocalizacaoant(imag,:)=[0 0];
      direcaoant(imag,:)=0;
   end

   if errorflag > 0
      fprintf('*****************************************************************\n');
      fprintf('Na primeira tentativa a diferença relativa entre o erro quadratico \nmedio
inicial e final foi superior a 50%% para a %s\n',imagem);
      fprintf('*****************************************************************\n');
      if errorflag > 1
         fprintf('*****************************************************************\n');
         fprintf('Na segunda tentativa a diferença relativa entre o erro quadratico
\nmedio inicial e final foi superior a 50%% para a %s\n',imagem);
         fprintf('*****************************************************************\n');
         if errorflag > 2
            fprintf('*****************************************************************\n');
            fprintf('Na terceira tentativa a diferença relativa entre o erro quadratico
\nmedio inicial e final foi superior a 50%% para a %s\n',imagem);
            fprintf('*****************************************************************\n');
            if errorflag > 3
               fprintf('*****************************************************************\n');
               fprintf('Na quarta tentativa a diferença relativa entre o erro quadratico
\nmedio inicial e final foi superior a 50%% para a %s\n',imagem);
               fprintf('*****************************************************************\n');
               if errorflag > 4
                  fprintf('*****************************************************************\n');
```

```
                fprintf('Na quinta tentativa a diferença relativa entre o erro quadratico
\nmedio inicial e final foi superior a 50%% para a %s\n',imagem);
                fprintf('*****************************************************************\n');
                if errorflag > 5

fprintf('*****************************************************************\n');
                  fprintf('Na sexta tentativa a diferença relativa entre o erro quadratico
\nmedio inicial e final foi superior a 50%% para a %s\n',imagem);

fprintf('*****************************************************************\n');
                  if errorflag > 6

fprintf('*****************************************************************\n');
                    fprintf('Na setima tentativa a diferença relativa entre o erro
quadratico \nmedio inicial e final foi superior a 50%% para a %s\n',imagem);

fprintf('*****************************************************************\n');
                    if errorflag > 7

fprintf('*****************************************************************\n');
                      fprintf('Na oitava tentativa a diferença relativa entre o erro
quadratico \nmedio inicial e final foi superior a 50%% para a %s\n',imagem);

fprintf('*****************************************************************\n');
                      if errorflag > 8

fprintf('*****************************************************************\n');
                        fprintf('Na nona tentativa a diferença relativa entre o erro
quadratico \nmedio inicial e final foi superior a 50%% para a %s\n',imagem);

fprintf('*****************************************************************\n');
                        if errorflag > 9

fprintf('*****************************************************************\n');
                          fprintf('Na decima tentativa a diferença relativa entre o
erro quadratico \nmedio inicial e final foi superior a 50%% para a %s e a mesma foi
descartada\n',imagem);

fprintf('*****************************************************************\n');
                        end
                      end
                    end
                  end
                end
              end
            end
          end
        end
      end
```

```
    end

    fprintf('deseja incluir mais uma amostra sim = 0, nao = 1?\n' );
     fprintf('default (sim = 0)' );
     flagt=input(': ' );
     if(isempty(flagt))
        flagt=0;
     end
     flag=flagt;

end
close all
%fim de mapa.m


%TrICPotmapa.m
%tendo em vista a utilizaçao do mattotal, q possui tres colunas com as
%informaçoes respectivamente de angulo, distancia radial e altura para cara
%ponto da correspondencias das imagens interna e externa.

function
[translacaopura,rotacaototal,translacao,teta,iteracao,Ptrans,Srel]=TrICPotmapa(M,P)

close all;
janela=500;

%carrega o modelo
Nm=length(M);
%modelo em X Y

%carrega a amostra
Np=length(P);
Pfirst=P;
%amostra em X Y

N=max(Np,Nm);

if N < 500
   itlim=50;
end
if N >= 500 & Np < 1000
   itlim=40;
end
if N >= 1000
   itlim=30;
end
S=1E10;
```

```
parada(1,:)=[S S/Np];
Santerior=1E10;
flag=0;
iteracao=0;
translacaototal=[0 0];
rotacaototal=0;
ming=[0 1E10 1E10];

while flag==0

   iteracao=iteracao+1;
   janela=510-.5*iteracao;
   for p=1:Np %passo 1
      im=0; %indice do modelo
      while im == 0
         maxvertical=P(p,2)+janela;
         minvertical=P(p,2)-janela;
         maxhorizontal=P(p,1)+janela;
         minhorizontal=P(p,1)-janela;
         for m=1:Nm  %monta um vetor com os indices do modelo q se encontram
dentro do quadrande de busca
            if (M(m,1) < maxhorizontal) & (M(m,1) > minhorizontal) & (M(m,2) <
maxvertical) & (M(m,2) > minvertical)
               im=im+1;
               vetorindice(im)=m;
            end
         end
         if im > 0
            distquad=0;
            for idist=1:im %calcula as distancias a todos os pontos dentro do quadrado
               distquad(idist)=(P(p,1)-M(vetorindice(idist),1))^2 + (P(p,2)-
M(vetorindice(idist),2))^2;
            end

            [temp,itemp]=min(distquad); %descobre o ponto mais proximo do modelo
            dquadrado(p,1)=vetorindice(itemp); %indice da menor distancia do modelo
ao ponto
            dquadrado(p,2)=temp; %valor da distancia ao quadrado
            dquadrado(p,3)=p;%indice da amostra
         else
            janela=janela + 10;
         end
      end
   end

   %passo 2
   crescentemp=sortrows(dquadrado,2); %ordena das distancias em ordem
crescente
```

```
    Stemp=sum(crescentemp(:,2)); %calcula a soma das distancias ao quadrado

    %passo 3, condiçao de parada
    parada(iteracao+1,:)=[Stemp Stemp/Np];
    if (iteracao > itlim) | (parada(iteracao+1,2) < 1E-10) | (abs(parada(iteracao+1,2)-
parada(iteracao,2)) < 10)
        flag=1;
        break;
    else
        if (parada(iteracao+1,2)-parada(iteracao,2) < 0) &
(ming(3)>parada(iteracao+1,2))
            ming=[iteracao Stemp parada(iteracao+1,2)];
        end
        S=Stemp;
    end

    %passo 4

    %media dos pontos casados
    Ntemp=length(crescentemp);
    xsoma=0;
    ysoma=0;
    for media=1:Ntemp
        xsoma=xsoma+M(crescentemp(media,1),1);
        ysoma=ysoma+M(crescentemp(media,1),2);
    end
    mediaMc(1)=xsoma/Ntemp;
    mediaMc(2)=ysoma/Ntemp;

    xsoma=0;
    ysoma=0;
    for media=1:Ntemp
        xsoma=xsoma+P(crescentemp(media,3),1);
        ysoma=ysoma+P(crescentemp(media,3),2);
    end
    mediaPc(1)=xsoma/Ntemp;
    mediaPc(2)=ysoma/Ntemp;

    %os dois mapas relativos as suas centroides
    Mmc(:,1)=M(:,1)-mediaMc(1);
    Mmc(:,2)=M(:,2)-mediaMc(2);

    Pmc(:,1)=P(:,1)-mediaPc(1);
    Pmc(:,2)=P(:,2)-mediaPc(2);

    %rotaçao, conciderando pontos coplanares
    Se=0;
    C=0;
```

```
   for icres=1:Np
      Se=Se+(Mmc(crescentemp(icres,1),1)*Pmc(crescentemp(icres,3),2)-
Mmc(crescentemp(icres,1),2)*Pmc(crescentemp(icres,3),1));

C=C+(Mmc(crescentemp(icres,1),1)*Pmc(crescentemp(icres,3),1)+Mmc(crescentemp
(icres,1),2)*Pmc(crescentemp(icres,3),2));
   end
   sinteta=Se/(sqrt(Se^2+C^2));
   costeta=C/(sqrt(Se^2+C^2));
   teta(iteracao,1)=acos(sqrt((1+costeta)/2))*2;
   teta(iteracao,2)=asin(sinteta/sqrt((1+costeta)*2))*2;

   %escala, conciderando o erro maior na amostra q no modelo

   %rotacionando a amostra media
   [A,D]=cart2pol(Pmc(:,1),Pmc(:,2));
   A=A-teta(iteracao,2);
   [X Y]=pol2cart(A,D);
   Pmr=[X Y];

   den=0;
   num=0;
   for i=1:Np

num=num+(Mmc(crescentemp(i,1),1)*Pmr(crescentemp(i,3),1)+Mmc(crescentemp(i,1
),2)*Pmr(crescentemp(i,3),2));

den=den+(Pmc(crescentemp(i,3),1)*Pmc(crescentemp(i,3),1)+Pmc(crescentemp(i,3),
2)*Pmc(crescentemp(i,3),2));
   end
   scale=num/den;

   %translaçao

   %rotacionando a media da amostra
   [A,D]=cart2pol(mediaPc(1),mediaPc(2));
   A=A-teta(iteracao,2);
   [X,Y]=pol2cart(A,D);
   mediaPcr=[X Y];

   translacao(iteracao,1)=mediaMc(1)-scale*mediaPcr(1);
   translacao(iteracao,2)=mediaMc(2)-scale*mediaPcr(2);

   %passo 5
   %aplicando a rotaçao e a translaçao a amostra

   P(:,1)=P(:,1)+translacao(iteracao,1);
   P(:,2)=P(:,2)+translacao(iteracao,2);
```

```
    [A,D]=cart2pol(P(:,1),P(:,2));
    A=A-teta(iteracao,2);
    [X,Y]=pol2cart(A,D);
    P=[X Y];

    translacaototal=translacaototal + [translacao(iteracao,1) translacao(iteracao,2)];
    rotacaototal=rotacaototal + teta(iteracao,2);
    Santerior=S;

end

iteracao-1
Plast=Pfirst(1,:);
figure(1)
plot(M(:,1),M(:,2),'o')
grid on;
title('Representaçao no espaço');
xlabel('X');
ylabel('Y');
hold;

if(parada(iteracao,2)>ming(3)*1.1)
    iteracao=ming(1)
    ming(2)
    ming(3)
    fprintf('mudança relativa percentual: ')
    Srel=-(ming(2)-parada(2,1))/parada(2,1)*100
    translacaototal=[0 0];
    rotacaototal=0;

    for it=1:ming(1)
        Pfirst(:,1)=Pfirst(:,1)+translacao(it,1);
        Pfirst(:,2)=Pfirst(:,2)+translacao(it,2);

        [A,D]=cart2pol(Pfirst(:,1),Pfirst(:,2));
        A=A-teta(it,2);
        [X,Y]=pol2cart(A,D);
        Pfirst=[X Y];
        translacaototal=translacaototal + [translacao(it,1) translacao(it,2)];
        rotacaototal=rotacaototal + teta(it,2);
    end
    plot(Pfirst(:,1),Pfirst(:,2),'k.')
    Ptrans=Pfirst;
else
    parada(iteracao,1)
    parada(iteracao,2)
    fprintf('mudança relativa percentual: ')
```

```
    Srel=-(parada(iteracao,1)-parada(2,1))/parada(2,1)*100

    for it=1:(iteracao-1)
        Pfirst(1,1)=Pfirst(1,1)+translacao(it,1);
        Pfirst(1,2)=Pfirst(1,2)+translacao(it,2);
        [A,D]=cart2pol(Pfirst(1,1),Pfirst(1,2));
        A=A-teta(it,2);
        [X,Y]=pol2cart(A,D);
        Pfirst=[X Y];
    end
    plot(P(:,1),P(:,2),'k.')
    Ptrans=P;
    iteracao=iteracao-1;
end

[A,D]=cart2pol(Plast(1),Plast(2));
A=A-rotacaototal;
[X,Y]=pol2cart(A,D);
Plast=[X Y];

translacaopura=Pfirst(1,:)-Plast
translacaototal
rotacaototal*180/pi
pause
%fim de TrICPotmapa.m

%mostramapa.m
function [mapatemp,Mapa]=mostramapa(Mapa,imag)

figure(2)
scatter(Mapa(:,1),Mapa(:,2),15,Mapa(:,3),'filled')
colorbar;
grid on;
title('Mapa com numero de pontos vizinhos e colapsados');
xlabel('X');
ylabel('Y');

n=4;

figure(3)
hold on;
ipor=0;
Nmt=length(Mapa);
for i=1:Nmt
    if Mapa(i,3) > n
        ipor=ipor+1;
        mapatemp(ipor,:)=[Mapa(i,1) Mapa(i,2) Mapa(i,3)];
    end
```

```
end

scatter(mapatemp(:,1),mapatemp(:,2),15,mapatemp(:,3),'filled')
colorbar;
grid on;
temp=strcat('Mapa com pelo menos 3 pontos');
title(temp);
xlabel('X');
ylabel('Y');

if imag-floor(imag/5)*5 == 0
    Mapa=mapatemp;
end
%fim de mostramapa.m


% analisa.m
function [P]=analisa(P);

close all

Np=length(P);

maxP=10000;

i=0;
j=1;
for t=0:359
    i=i+1;
    if i>Np
        break;
    end
    if (P(i,1)==t)
        if  P(i,2)<maxP
            Pf(j,:)=P(i,:);
            j=j+1;
        end
        i=i+1;
        if i>Np
            break;
        end
        while P(i,1)==t
            i=i+1;
            if i>Np
                break;
            end
        end
    end
```

```
    if i>Np
        break;
    end
    i=i-1;
end

Pf(:,1)=Pf(:,1)*pi/180;

[X,Y]=pol2cart(Pf(:,1),Pf(:,2));
P=[X Y];
%fim de analisa.m
```

ANEXO B

Imagens fotografadas do ambiente simulado

De modo a testar o algoritmo TrICP uma trajetória foi criada na sala, com pontos de tomada de imagens pelo sistema de visão omnidirecional. A figura 28 mostra esta trajetória, onde cada ponto vermelho marca a tomada das imagens. A tabela 10 mostra as coordenadas dos pontos no sistema de coordenadas do ambiente.

Tabela 10 - Coordenadas da tomada de pontos da trajetória do robôs no ambiente simulado

Ponto	Coord. X	Coord. Z	Ponto	Coord. X	Coord. Z
1	3000	1500	6	0	0
2	3000	500	7	-1000	500
3	3000	-500	8	-2000	500
4	2000	-500	9	-3000	500
5	1000	-500	10	-3000	1500

altura 3m, [-700,2700]

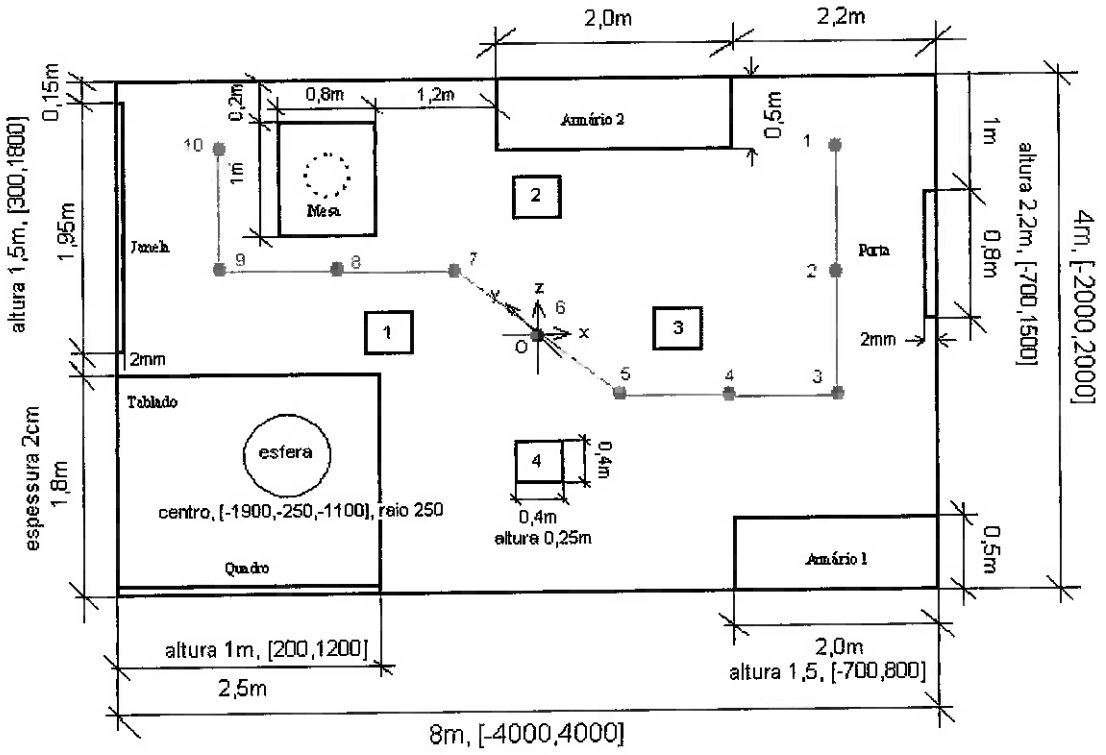

Figura 28 - Trajetória do robô pelo ambiente simulado com as posições das tomadas de imagem.

Abaixo seguem as dez imagens adquiridas.

Figura 29 - Primeiro ponto da trajetória.

Figura 30 - Segundo ponto da trajetória.

Figura 31 - Terceiro ponto da trajetória.

Figura 32 - Quarto ponto da trajetória.

Figura 33 - Quinto ponto da trajetória.

Figura 34 - Sexto ponto da trajetória.

Figura 35 - Sétimo ponto da trajetória.

Figura 36 - Oitavo ponto da trajetória.

Figura 37 - Nono ponto da trajetória.

Figura 38 - Décimo ponto da trajetória.