Pontifícia Universidade Católica
do Rio de Janeiro

Felipe Augusto Weilemann Belo

Desenvolvimento de Algoritmos de Exploração e Mapeamento Visual para Robôs Móveis de Baixo Custo

Dissertação de Mestrado

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós-graduação em Engenharia Elétrica do Departamento de Engenharia Elétrica da PUC-Rio.

Orientador: Prof. Abraham Alcaim
Co-Orientador: Prof. Marco Antonio Meggiolaro

Rio de Janeiro

Abril de 2006

Pontifícia Universidade Católica
do Rio de Janeiro

Felipe Augusto Weilemann Belo

Desenvolvimento de Algoritmos de Exploração e Mapeamento Visual para Robôs Móveis de Baixo Custo

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós-Graduação em Engenharia Elétrica do Departamento de Engenharia Elétrica do Centro Técnico Científico da PUC-Rio. Aprovada pela Comissão Examinadora abaixo assinada.

Prof. Abraham Alcaim
Orientador
Centro de Estudos em Telecomunicações - PUC-Rio

Prof. Marco Antonio Meggiolaro
Co-Orientador
Departamento de Engenharia Mecânica - PUC-Rio

Prof. Armando Morado Ferreira
IME

Prof. Raul Queiroz Feitosa
Departamento de Engenharia Elétrica – PUC-Rio

Prof. Marcelo de Andrade Dreux
Departamento de Engenharia Mecânica - PUC-Rio

Prof. José Eugenio Leal
Coordenador Setorial do Centro
Técnico Científico - PUC-Rio

Rio de Janeiro, 06 de abril de 2006

Felipe Augusto Weilemann Belo

Graduou-se em Engenharia de Controle e Automação na Universidade PUC-Rio (Pontifícia Universidade Católica do Rio de Janeiro) em 2004.

Ficha catalográfica

Belo, Felipe Augusto Weilemann

Desenvolvimento de algoritmos de exportação e mapeamento visual para robôs móveis de baixo custo / Felipe Augusto Weilemann Belo ; orientadores: Abraham Alcaim, Marco Antonio Meggiolaro. – Rio de Janeiro : PUC-Rio, Departamento de Engenharia Elétrica, 2005.

276 f. ; 30 cm

Dissertação (mestrado) – Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Engenharia Elétrica

Inclui bibliografia

1. Engenharia elétrica – Teses. 2. Robôs móveis. 3. SLAM. 4. Teoria da informação. 5. Visual tracking. 6. SIFT. 7. Transformada de Fourier. 8. Transformada de Mellin. 9. Métodos de registro de imagens. 10. Visão computacional. 11. Algoritmos genéticos. I. Alcaim, Abraham. II. Meggiolaro, Marco Antonio. III. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Engenharia Elétrica. IV. Título.

CDD: 621.3

À minha mãe e meu orientador Marco Antonio Meggiolaro

Agradecimentos

Aos orientadores Marco Antonio Meggiolaro e Abraham Alcaim pela paciência, estímulo, assistência, inspiração e confiança dispensados para à realização deste trabalho.

Ao CNPq e PUC-Rio, pelos auxílios concedidos, sem os quais não teria sido possível realizar este trabalho.

À minha mãe, meu pai, minha avó, meu irmão e minha tia Geny.

À Fernanda Borges Caixeta e sua família.

Aos professores da PUC-Rio.

A meus amigos.

A todos os amigos e familiares que de alguma forma me estimularam e ajudaram.

Resumo

Belo, Felipe Augusto Weilemann; Alcaim, Abraham (Orientador). **Desenvolvimento de Algoritmos de Exploração e Mapeamento Visual para Robôs Móveis de Baixo Custo**. Rio de Janeiro, 2006. 276p. Dissertação de Mestrado - Departamento de Engenharia Elétrica, Pontifícia Universidade Católica do Rio de Janeiro.

Ao mesmo tempo em que a autonomia de robôs pessoais e domésticos aumenta, cresce a necessidade de interação dos mesmos com o ambiente. A interação mais básica de um robô com o ambiente é feita pela percepção deste e sua navegação. Para uma série de aplicações não é prático prover modelos geométricos válidos do ambiente a um robô antes de seu uso. O robô necessita, então, criar estes modelos enquanto se movimenta e percebe o meio em que está inserido através de sensores. Ao mesmo tempo é necessário minimizar a complexidade requerida quanto a hardware e sensores utilizados. No presente trabalho, um algoritmo iterativo baseado em entropia é proposto para planejar uma estratégia de exploração visual, permitindo a construção eficaz de um modelo em grafo do ambiente. O algoritmo se baseia na determinação da informação presente em sub-regiões de uma imagem panorâmica 2-D da localização atual do robô obtida com uma câmera fixa sobre o mesmo. Utilizando a métrica de entropia baseada na Teoria da Informação de Shannon, o algoritmo determina nós potenciais para os quais deve se prosseguir a exploração. Através de procedimento de *Visual Tracking*, em conjunto com a técnica SIFT (*Scale Invariant Feature Transform*), o algoritmo auxilia a navegação do robô para cada nó novo, onde o processo é repetido. Um procedimento baseado em transformações invariáveis a determinadas variações espaciais (desenvolvidas a partir de Fourier e Mellin) é utilizado para auxiliar o processo de guiar o robô para nós já conhecidos. Também é proposto um método baseado na técnica SIFT. Os processos relativos à obtenção de imagens, avaliação, criação do grafo, e prosseguimento dos passos citados continua até que o robô tenha mapeado o ambiente com nível pré-especificado de detalhes. O conjunto de nós e imagens obtidos são combinados de modo a se criar um modelo em grafo do ambiente. Seguindo os caminhos, nó a nó, um robô pode navegar pelo ambiente já explorado. O método é particularmente adequado para ambientes planos. As componentes do algoritmo proposto foram desenvolvidas e testadas no presente trabalho. Resultados experimentais mostrando a eficácia dos métodos propostos são apresentados.

Palavras-chave

Robôs Móveis, SLAM, Teoria da Informação, Visual Tracking, SIFT, Transformada de Fourier, Transformada de Mellin, Métodos de Registro de Imagens, Visão Computacional, Algoritmos Genéticos.

Abstract

Belo, Felipe Augusto Weilemann; Alcaim, Abraham (Advisor). **Exploration and Visual Mapping Algorithms Development for Low Cost Mobile Robots**. Rio de Janeiro, 2006. 276p. MSc Dissertation - Departamento de Engenharia Elétrica, Pontifícia Universidade Católica do Rio de Janeiro.

As the autonomy of personal service robotic systems increases so has their need to interact with their environment. The most basic interaction a robotic agent may have with its environment is to sense and navigate through it. For many applications it is not usually practical to provide robots in advance with valid geometric models of their environment. The robot will need to create these models by moving around and sensing the environment, while minimizing the complexity of the required sensing hardware. This work proposes an entropy-based iterative algorithm to plan the robot's visual exploration strategy, enabling it to most efficiently build a graph model of its environment. The algorithm is based on determining the information present in sub-regions of a 2-D panoramic image of the environment from the robot's current location using a single camera fixed on the mobile robot. Using a metric based on Shannon's information theory, the algorithm determines potential locations of nodes from which to further image the environment. Using a Visual Tracking process based on SIFT (Scale Invariant Feature Transform), the algorithm helps navigate the robot to each new node, where the imaging process is repeated. An invariant transform (based on Fourier and Mellin) and tracking process is used to guide the robot back to a previous node. Also, an SIFT based method is proposed to accomplish such task. This imaging, evaluation, branching and retracing its steps continues until the robot has mapped the environment to a pre-specified level of detail. The set of nodes and the images taken at each node are combined into a graph to model the environment. By tracing its path from node to node, a service robot can navigate around its environment. This method is particularly well suited for flat-floored environments. The components of the proposed algorithm were developed and tested. Experimental results show the effectiveness of the proposed methods.

Keywords

Mobile Robots, SLAM, Information Theory, Visual Tracking, SIFT, Fourier Transform, Mellin Transform, Image Registration Methods, Computational Vision, Genetics Algorithms.

Sumário

Lista de figuras

Lista de tabelas

1
Introdução

1.1.
Motivação

O crescimento e popularização da robótica têm se evidenciado nos últimos anos principalmente com a explosão da automação industrial, comercial e residencial, além do aparecimento de robôs móveis caracterizados por diversos produtos e aplicações.

O uso de robôs na automação industrial, exploração espacial, exploração submarina, em locais inóspitos e em tarefas penosas e perigosas tem experimentado ampla difusão. Robôs autônomos são capazes de substituir o homem em trabalhos de alto risco, assim como em serviços que não enobrecem o ser humano, podendo realizar atividades em áreas de difícil acesso ou em condições ambientais desfavoráveis. No Brasil, robôs são cada vez mais usados em operações como levantamentos hidrográficos, monitoração ambiental, supervisão de sistemas de produção de petróleo e gás e pesquisas oceanográficas entre outras, gerando grande demanda de pesquisa em robótica, principalmente na área submarina. Um trabalho nacional feito sobre reconhecimento de ambientes atráves de sistema robótico autônomo, que utiliza visão para alcançar objetivos pré-determinados pode ser visto em [1].

A RIA (*Robotic Industries Association*) estima que há atualmente 142 mil robôs em uso industrial nos Estados Unidos. Em 2003, a venda de robôs industriais na América do Norte observou aumento de 19% com valor estimado de 876,5 milhões de dólares. De acordo com a UNECE (*The United Nations Economic Comission for Europe*), em 2003 houve crescimento de 25% na demanda mundial de robôs, sendo a robótica atualmente uma industria de 8 bilhões de dólares globalmente, com crescimento esperado para 22,61 bilhões de dólares em 2010.

Também se podem verificar grandes mudanças no mercado de robôs móveis de uso doméstico. Segundo a UNECE o maior crescimento para os próximos 3

anos será relativo a robôs pessoais, e não industriais. Enquanto um robô industrial é projetado para realizar tarefas na manufatura, robôs pessoais são construídos para realizar tarefas para seu usuário. Hoje podem ser encontrados robôs entre eletrodomésticos, como aspiradores robóticos autônomos, cortadores de grama e equipamentos para segurança, inspeção e vigilância robóticos. A indústria de brinquedos e robôs para entretenimento, tal como o uso de ferramentas educacionais como kits de robótica, também tem tido grande crescimento. Este fenômeno não é decorrente somente do perceptível aumento do interesse relacionado a robôs móveis, mas é também relativo à queda de preços associada à produção de robôs. As aplicações possíveis para robôs móveis pessoais vão do transporte, vigilância, segurança, limpeza, entretenimento, educação a até mesmo o auxílio a deficientes físicos, entre outras.

Se a tecnologia para robôs de uso pessoal crescer como promete, com preços competitivos, e boa aceitação comercial, este pode vir a ser um grande mercado em um futuro próximo, ainda mais se for levado em conta que a indústria eletro-eletrônica é a que apresenta o segmento com o maior número de empregados mundialmente. De acordo com Dan Kara, presidente da empresa *Robotic Trends*, a industria de robôs pessoais deve crescer de 5,4 bilhões de dólares em 2005 para 17,1 bilhões em 2010.

O avanço da robótica atual se deve em grande parte a alguns fatores. Primeiramente, a redução de custos associados a hardware tem permitido a produção de robôs cada vez mais baratos e com grande poder computacional, requisito importante para desenvolvimento de sistemas autônomos. Além disto, a pesquisa em robótica tem crescido muito, permitindo grandes avanços ao integrar áreas do conhecimento tão diversas quanto Inteligência Artificial, Ciência da Computação, Matemática, Engenharia Elétrica, Engenharia de Controle e Engenharia Mecânica.

Fatores econômicos ainda requerem a substituição de arquiteturas complexas por arquiteturas mais simples, o desenvolvimento de robôs de baixo custo, além do uso de sistemas inteligentes. O grande objetivo da robótica está na criação de robôs autônomos capazes de executar tarefas sem a necessidade de intervenção humana. A autonomia de robôs é de muito interesse em uma grande quantidade de aplicações, principalmente aquelas que envolvem riscos, como operações de resgate e segurança, desarmamento de minas, uso policial,

exploração submarina e espacial, aplicações nucleares e militares. Também são importantes as aplicações que envolvem benefícios econômicos, abrangendo a indústria de serviços, agricultura, manufatura e construção, e até mesmo o uso em cirurgias médicas.

Dentro do cenário apresentado, o estudo de robôs móveis e do planejamento de movimento tem sido tema central de diversas pesquisas nos meios acadêmico e industrial. Os países que não estiverem capacitados ao desenvolvimento das tecnologias associadas à automação e robótica certamente ficarão defasados tecnologicamente no mercado global. É uma preocupação mundial atual a viabilização rápida de máquinas inteligentes, a viabilização de altos ganhos em flexibilidade e o desenvolvimento de sistemas robóticos com desempenho próximo ao humano.

A presente dissertação tem como objetivo apresentar algoritmos para navegação, exploração e mapeamento de ambientes por robôs móveis de baixo custo auxiliados por uma câmera simples e de baixa resolução.

A seção 1.2 apresenta uma visão geral sobre robôs autônomos, robôs móveis, métodos tradicionais de representação de um ambiente por um robô e técnicas tradicionais de exploração de um ambiente por um robô.

A seção 1.3 descreve os objetivos propostos nesta dissertação.

A seção 1.4 descreve os capítulos seguintes do presente trabalho.

1.2. Navegação, Localização e Mapeamento para Robôs Móveis

1.2.1. Robôs Autônomos

Robôs inteligentes são sistemas mecânicos que possuem a capacidade de funcionamento autônomo. Estes devem integrar as capacidades de perceber o mundo, raciocinar e interagir com o mesmo, podendo aprender e se adaptar às mudanças ocorridas no ambiente. Percepção, planejamento e ação constituem as três primitivas de robôs inteligentes.

A capacidade de percepção de um robô agrega os instrumentos necessários para mensurar suas propriedades internas e outras relativas ao ambiente. Com tal

propósito existe uma grande variedade de sensores com aplicação comum em robótica indo desde odômetros e sonares a câmeras digitais.

Um trabalho recente sobre a percepção de ambientes utilizando-se de câmera de vídeo acoplada a um sistema robótico autônomo pode ser visto em [1].

Aliado à percepção, robôs autônomos devem poder incorporar uma base de conhecimento em relação ao ambiente junto a qual possam ter capacidade de raciocínio. As habilidades de percepção e raciocínio é que permitirão ao robô observar o mundo e tomar ações de modo autônomo e adaptativo permitindo ao sistema se locomover de forma inteligente e decidir quais movimentos são necessários de modo a cumprir objetivos pré-determinados.

Além das habilidades de percepção e raciocínio, também é de fundamental importância para um robô, dito inteligente, que consiga aprender e evoluir quais os melhores comportamentos a serem tomados. Para muitos pesquisadores, a autonomia está diretamente relacionada à capacidade de aprendizado.

A autonomia é um fator de grande importância para robôs móveis. Uma visão geral sobre robôs móveis é apresentada a seguir.

1.2.2. Robôs Móveis

O estudo de robôs móveis é uma área relativamente recente de pesquisa que lida com o controle de veículos autônomos e sua relação com o ambiente. O estudo do planejamento do movimento tem tido papel vital no avanço da robótica. O mínimo que se pode esperar de um robô autônomo é a capacidade de planejar seus próprios movimentos. Esta capacidade é necessária dado que, para um robô realizar tarefas, este precisa se movimentar e interagir com o mundo real.

Nos Estados Unidos, agências como o DARPA e ESPIRIT têm financiado fortemente projetos de robôs de larga escala, tais como veículos convencionais e *off-road* autônomos. O tamanho destes projetos tem requerido a colaboração de centenas de pesquisadores de diversas universidades e áreas de estudo distintas.

Quanto à sua movimentação, robôs móveis autônomos devem ter as habilidades de navegação, de exploração, de mapeamento e de auto-localização. A interação mais básica de um robô com o ambiente é feita pela percepção deste e sua navegação. A navegação constitui a aptidão de se chegar a lugares desejados sabendo-se da sua localização e conhecendo-se o ambiente ao seu redor. Para

navegar, muitas vezes o robô precisa de um modelo do ambiente, sem o qual não pode tomar decisões. Em determinadas aplicações o modelo não é dado ao robô, que precisa então fazer a modelagem do ambiente movendo-se e "sentindo-o". Daí o mapeamento lida com a criação de um modelo representativo do mundo através da percepção do mesmo feita por sensores. A capacidade de explorar um ambiente está no robô descobrir áreas ainda não conhecidas do ambiente visando agregar mais informações à sua base de conhecimento da maneira mais eficaz possível. Por fim, a auto-localização é o procedimento que integra a percepção sensorial com o conhecimento previamente acumulado, objetivando-se a localização do robô em relação ao ambiente.

As quatro tarefas apresentadas se inter-relacionam e requerem a tomada de decisões de maneira autônoma e inteligente. Agentes autônomos e inteligentes devem não só poder interagir com o ambiente na busca de atingir seus objetivos, reunir informações e aprender com o mesmo, mas também se adaptarem às mudanças.

O mapeamento de um ambiente gera um representação interna ao robô do meio. A seção seguinte apresenta algumas representações tradicionais possíveis.

1.2.3. Representação do Ambiente

Ao navegar pelo espaço, um robô pode fazer uma modelagem interna deste ambiente criando um mapa do mesmo. Existem diversas possibilidades de representação do ambiente, sendo as mais usuais:

- Representações Métricas:
  - Decomposição Espacial [2-6]:
    - *Bitmaps*;
    - *BSP*;
    - *Quadtree*;
- Representações Geométricas [7-11];
- Representações Topológicas [12-14];

As representações métricas do ambiente lidam com a reprodução ótima, de acordo com as medidas obtidas, de modo quantitativo e preciso.

Na representação métrica por decomposição espacial guarda-se uma tabela ou *grid* que representa o espaço (Figura 1-1). A decomposição espacial mais simples é por *bitmap*. Nesta decomposição, existem dois componentes essenciais:

- As coordenadas do robô em uma referência global;
- Representação em *grid* do ambiente onde as propriedades do mesmo são guardadas;

A decomposição por *bitmap* usa uma representação icônica baseada em uma matriz de células, onde cada célula está centrada em coordenadas $r = (r_i, r_j)$ do mundo real e correspondem a uma área quadrada. Cada célula carrega atributos referentes ao ambiente tais como estado (vazio, ocupado, não explorado) e certeza (medida de probabilidade referente aos seus estados possíveis).

A Figura 1-1 apresenta um exemplo de uma representação métrica do ambiente. Em azul pode-se ver as paredes do ambiente. Os quadrados cinzas representam as áreas do *grid* ocupadas (paredes) e os brancos as áreas não ocupadas.

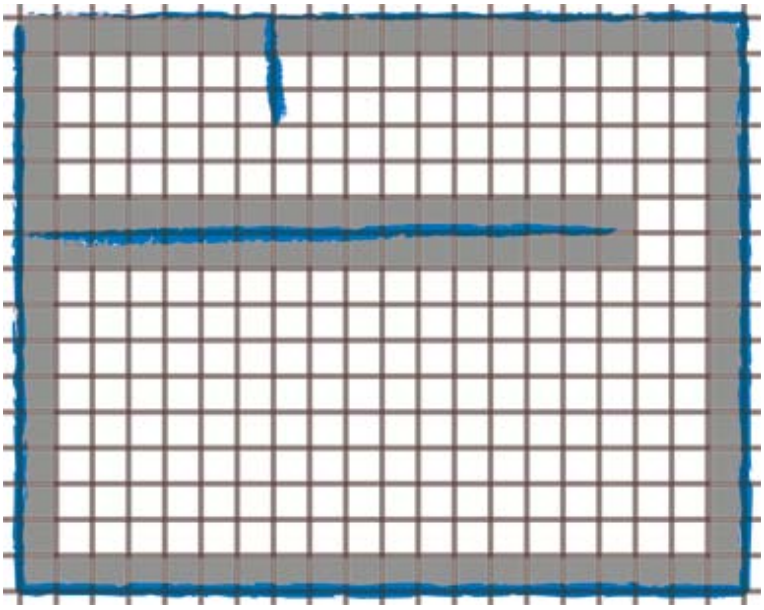

Figura 1-1: Representação métrica do ambiente

As representações topológicas do ambiente (Figura 1-2) abordam técnicas relacionais e associativas de representação. As técnicas representativas usam modelos em grafo, onde diferentes *landmarks* são tratados como nós de um grafo e seus caminhos são tratados como adjacências.

Enquanto modelos métricos são mais precisos, modelos topográficos/topológicos são mais robustos quanto a estratégias de navegação. A existência de modelos híbridos associa precisão e robustez criando uma independência de modelos que faz com que erros de percepção verificados nas representações métricas não se propaguem para as representações topológicas. O uso de um modelo híbrido pode ser visto em [14].

A Figura 1-2 apresenta o mesmo ambiente da Figura 1-1 mas utilizando uma representação topológica. Os círculos representam os nós e as linhas cinzas representam os caminhos entre os nós.

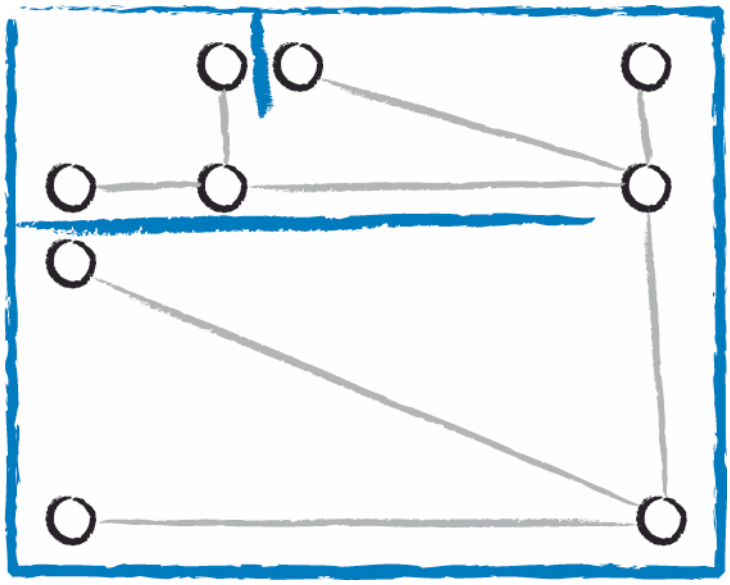

Figura 1-2: Representação Topológica do ambiente

Para poder representar um ambiente, um robô precisa fazer a exploração do mesmo. A seção seguinte apresenta técnicas tradicionais de exploração de um ambiente.

1.2.4. Técnicas de Exploração

O problema central em exploração está em cobrir uma área não explorada de maneira eficaz. Alguns métodos propostos mais tradicionais são:

- *Random Potential Field* – Busca aleatória: Percorre-se o ambiente randomicamente seguindo um campo potencial gerado de modo aleatório [13];

- *Repulsive Field*: Utiliza-se de campo de potencial onde o robô é repelido de áreas visitadas recentemente e atraído para áreas ainda não conhecidas. Um trabalho que usa campos potenciais harmônicos pode ser visto em [15].;
- Busca por centróides das áreas desconhecidas: O robô tenta seguir centróides de áreas ainda desconhecidas [13];
- *Frontier-Based Exploration* [16]: Encontrando-se as fronteiras entre as áreas conhecidas e as áreas desconhecidas, é traçada uma estratégia de exploração que pode levar em conta o centróide destas áreas, como exemplo;
- *Voronoi-Graph Methods*: Métodos baseados em grafos de *Voronoi* podem ser vistos em [2, 4, 13];

Nesta e nas últimas seções foi apresentado um panorama sobre robôs autônomos e robôs móveis, métodos de representação de um ambiente por um robô e técnicas tradicionais para a exploração de um ambiente. Dentro deste contexto, na próxima seção serão apresentados os objetivos do presente trabalho.

1.3. Objetivo

O objetivo deste projeto é desenvolver técnicas para navegação, exploração e mapeamento de ambientes por robôs móveis utilizando-se de tecnologias de áreas diversas, tendo em vista o grande número de estudos recentes com foco nestas técnicas aliado ao crescimento comercial do uso de robôs móveis.

Esta dissertação tem como fim desenvolver e aplicar as componentes de um algoritmo de auto-localização e mapeamento simultâneos (SLAM – Simultaneous Localization And Mapping) para robôs móveis que planeje uma estratégia de exploração visual (por câmeras) do ambiente, permitindo a construção eficiente de um modelo topológico do mesmo, sem necessidade de se usar sensores de alto custo.

Este objetivo é atingido através da combinação de componentes de tecnologia já existentes para a geração de um sistema robótico de baixo custo. A contribuição deste projeto está na fusão dos procedimentos da Teoria da Informação, Visão Computacional, Processamento de Imagens, Algoritmos

Genéticos e algoritmos de SLAM baseados em grafos para a criação de algoritmos de exploração e navegação de robôs móveis.

É proposto um algoritmo de SLAM para robôs móveis que planeje uma estratégia de exploração visual (por câmeras) do ambiente. Este algoritmo deve fazer a construção eficiênte de um modelo topológico (baseado em grafos) do ambiente sem necessidade de se usar sensores com alto custo. As componentes deste algoritmo são desenvolvidas e investigadas.

Todo o desenvolvimento é feito para ambientes estáticos, apesar de se verificar posteriormente robustez na presença de alguns elementos dinâmicos no ambiente.

O projeto consiste no prosseguimento de trabalhos já desenvolvidos ou em desenvolvimento, tendo como foco a aplicação prática dos métodos criados. Além disso explora técnicas clássicas na área em conjunto com técnicas mais atuais viabilizando novos avanços para as pesquisas recentes.

O algoritmo proposto se baseia na quantidade de informação presente em sub-regiões de imagens panorâmicas em 2D para determinar lugares a serem explorados pelo robô usando uma única câmera fixa sobre o mesmo. Utilizando métrica relacionada à quantidade de informação (derivada da Teoria da Informação de Shannon) [17] , o algoritmo determinará na imagem localizações potenciais de nós para se prosseguir a exploração do ambiente. O algoritmo também lidará com a navegação do robô para os novos nós, onde o processo deve ser repetido. Uma estratégia de reconhecimento de imagens invariável à translação, escala e rotação, utilizando transformadas com características invariáveis (Fourier, Mellin do tipo 1 e Mellin do tipo 2) [18-24], é utilizada para navegar o robô para nós já conhecidos. Também é apresentada técnica alternativa a esta, utilizando-se de transformações baseadas em SIFT (*Scale-Invariant Feature Transforms*) [25-35]. O algoritmo prossegue iterativamente até ter explorado o ambiente com nível de detalhamento pré-especificado.

Durante o desenvolvimento da dissertação, etapas do algoritmo foram desenvolvidas, testadas e aplicadas em experimentos práticos com um robô do tipo *differential-drive*, como será descrito na seção adiante.

1.4. Metodologia

O objetivo deste projeto foi desenvolver os procedimentos do algoritmo de exploração e mapeamento do ambiente proposto. O mapeamento do ambiente é topológico, ou seja, o mapa é um grafo onde cada nó representará uma localização do ambiente. O robô deve guardar uma imagem panorâmica de cada nó tal como as distâncias e ângulos entre os nós explorados, para que em um segundo momento, este mapa criado possa servir de base para que o robô navegue autonomamente para lugares desejados.

A princípio, o projeto pode ser sub-dividido em 5 componentes. Estas componentes são:

1. Criação de imagens panorâmicas;
2. Identificação de lugares a serem explorados;
3. Navegação para lugar não explorado;
4. Navegação para lugar já explorado;
5. Processamento das imagens obtidas e do grafo que representa o ambiente;

Na componente 1 do algoritmo, a criação das imagens panorâmicas é feita através da composição de várias imagens obtidas com o robô apontado para diferentes ângulos. Para que as imagens sejam concatenadas é necessário encontrar pontos em comum entre elas e recombiná-las. Isto foi feito utilizando técnicas baseadas em entropia e análises de correlações entre regiões das imagens, entre outras.

Na componente 2, para determinar localizações potenciais a serem exploradas, pode-se utilizar alguma métrica, como entropia por exemplo, que faça essa escolha através das imagens panorâmicas. Uma das opções escolhidas para tal foi o algoritmo de exploração se basear na quantidade de informação (entropia) presente em sub-regiões de imagens panorâmicas em 2D para determinar regiões a serem exploradas pelo robô usando uma única câmera fixa sobre o mesmo.

O algoritmo proposto também deve lidar com a navegação do robô para os novos nós através de controle com feedback servo-visual. Na componente 3, uma estratégia de reconhecimento de imagens rápida é utilizada para navegar o robô para nós já conhecidos. Isso foi inicialmente feito fazendo comparações de

transformadas invariáveis a determinadas características das imagens como escala, translação e rotação. Transformadas como de Fourier ou Mellin, por exemplo, possuem propriedades que auxiliaram nesta tarefa.

Também foi adotado outro método para a componente 3. Neste método utilizou-se de descritores SIFT, junto a um algoritmo de *Visual Tracking* proposto, para se fazer a navegação para lugares desconhecidos. Uma variação deste método foi utilizada na componente 4, onde se deseja navegar para lugares já conhecidos.

Uma segunda etapa do projeto envolve o refinamento do modelo interno criado fazendo-se refinamento do grafo relativo ao mapeamento topológico utilizando técnica baseada em algoritmos genéticos e teoria de grafos.

1.5. Organização da Tese

O restante da tese está dividido em mais 7 capítulos, conforme descrito a seguir:

- Capítulo 2 - Transformações Integrais Invariáveis: Uma série de transformadas (Fourier, Mellin do tipo 1, Mellin do ripo 2, Fourier Mellin do tipo 1 e Fourier Mellin do tipo 2) são apresentadas e é discutido o uso de propriedades características das mesmas para se fazer comparações entre imagens e auxiliar à navegação de robôs;
- Capítulo 3 - Transformação SIFT (Scale Invariant Features): Descritores SIFT são muito utilizados quando se deseja encontrar regiões em comum entre diferentes imagens. A base teórica e as aplicações da transformação SIFT são aqui vistas. A técnica SIFT será utilizada no auxílio à navegação e exploração em vários momentos do presente trabalho;
- Capítulo 4 - Registro e Correspondência de Imagens: Um breve panorama sobre técnicas de registro e correspondência de imagens pode ser conferido aqui;
- Capítulo 5 - Sistema Experimental: O sistema experimental sobre o qual o presente trabalho foi desenvolvido é apresentado;
- Capítulo 6 – Algoritmos para Exploração e Navegação Propostos: Este capítulo descreve as implementações desenvolvidas ao longo deste

trabalho. Diversos métodos são propostos com o objetivo de se explorar um ambiente navegando pelo mesmo com auxílio de uma câmera;

- Capítulo 7 - Experimentos e Resultados: Experimentos que ilustram e investigam os métodos propostos são vistos;
- Capítulo 8 – Conclusões e Trabalhos Futuros: Conclusões sobre os métodos empregados e breve descrição sobre possíveis trabalhos futuros;

2
Transformações Integrais Invariáveis

Transformações bidimensionais tais como Fourier, Mellin e Transformada Z são de grande importância para o processamento de imagens [18, 20, 21]. O uso de tais transformadas pode ser verificado extensamente em aplicações como pré-processamento de imagens, filtragem e compressão.

No presente trabalho as Transformadas de Fourier, Mellin do tipo 1 e Mellin do tipo 2 são apresentadas e estudadas porque estas apresentam propriedades que permitem que imagens sejam transformadas em versões invariáveis à escala, distorções, translação e rotação. Estas versões permitem que imagens que sofreram as variações citadas possam ser diretamente comparadas umas com as outras.

É importante ressaltar que ao longo do trabalho o uso do termo distorção será referente a dilatações horizontais e verticais, ou seja, alterações em escala horizontais e verticais isoladamente. O termo não é usado para se referir a distorções locais.

Primeiramente é apresentada a Transformada de Fourier que possui magnitude invariável à translação.

Após a Transformada de Fourier, é apresentado o conceito da Transformada de Mellin de uma dimensão, transformada esta que possui características similares à Transformada de Fourier e possui magnitude invariável à escala.

Então, são apresentados dois métodos de mapear uma imagem:

- Mapeamento *Log-Log*: Possui a propriedade de transformar variações em escala horizontal e vertical em variações em translação;
- Mapeamento Log-Polar: Possui a propriedade de transformar variações em escala e rotação em variações em translação;

Após os mapeamentos serem explicados, são apresentados métodos de combinar a Transformada de Fourier e estes mapeamentos de modo a se construir as transformadas de Mellin do tipo 1 e 2 e de Fourier Mellin do tipo 1 e 2. Neste momento são apresentados meios de se chegar às invariâncias desejadas.

Por fim, é descrito como fazer a comparação entre imagens aproveitando-se das transformadas citadas e um método de *Window Growing* é proposto, método este que será utilizado durante a etapa de navegação para lugares conhecidos.

Este capítulo apresenta as seguintes seções:

- 2.1 – Transformada de Fourier: Apresenta a Transformada de Fourier para uma dimensão, para duas dimensões, suas versões discretas e suas características. É importante verificar a propriedade de translação que será explorada para se conseguir as invariâncias desejadas. A seção possui as seguintes subdivisões:
  - 2.1.1 – Transformada de Fourier Discreta;
  - 2.1.2 – Propriedade de Translação da DFT;
  - 2.1.3 – Propriedade de Escala da DFT;
  - 2.1.4 – Propriedade de Rotação da DFT;
- 2.2 – Transformada de Mellin: Descreve o conceito da Transformada de Mellin para funções de uma dimensão e mostra as características comuns desta com a Transformada de Fourier;
- 2.3 – Mapeamento *Log-Polar*: O mapeamento *Log-Polar* transforma uma imagem que está no sistema de coordenadas cartesiano para um novo sistema de coordenadas no qual variações em escala e rotação representam variações em translação. Esta característica será aproveitada junto à Transformada de Fourier para se chegar às transformações invariáveis à escala, translações e rotação;
- 2.4 – Mapeamento *Log-Log*: Este é outro mapeamento como o anterior. Este, porém, possui a característica de transformar escalas horizontais e verticais em translações;
- 2.5 – Obtendo Invariâncias à Rotação, Translação, Dilatações e Escala: Esta seção mostra como os conceitos apresentados até a mesma são combinados de modo a se obter as transformadas de Mellin do tipo 1 e 2, e Fourier Mellin do tipo 1 e 2. A seção está dividida em:
  - 2.5.1 – Invariância à Translação;
  - 2.5.2 – Invariância à Rotação e Escala;
  - 2.5.3 – Invariância a Dilatações Horizontais e Verticais;
  - 2.5.4 – Propriedade Comutativa Entre Rotação e Dilatação;

- 2.5.5 – Invariância à Translação, Rotação e Escala Simultâneas;
- 2.5.6 – Invariância à Translação e Dilatações Verticais e Horizontais Simultâneas;

- 2.6 – Comparando Imagens: A última seção do capítulo apresenta o uso das transformadas descritas para comparação entre imagens. As sub-seções desta são:
  - 2.6.1 – Distância Euclidiana;
  - 2.6.2 – Correlação;
  - 2.6.3 – Comparando Imagens com Invariâncias;
  - 0 – *Window Growing*;

2.1. Transformada de Fourier

Em uma dimensão, a Transformada de Fourier de uma função complexa f(x) é definida como:

$$F(k) = \Im[f(x)](k) = \int_{-\infty}^{\infty} f(x) e^{-2\pi i k x} dx \quad (1)$$

A Transformada de Fourier inversa é dada por:

$$f(x) = \Im^{-1}[F(k)](x) = \int_{-\infty}^{\infty} F(k) e^{2\pi i k x} dk \quad (2)$$

A versão em duas dimensões da Transformada de Fourier é dada por:

$$F(\varpi_x, \varpi_y) = \Im[f(x,y)](\varpi_x, \varpi_y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y) e^{-2\pi i(\varpi_x x + \varpi_y y)} dx dy \quad (3)$$

Sendo sua inversa:

$$f(x,y) = \Im^{-1}[F(\varpi_x, \varpi_y)](x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F(\varpi_x, \varpi_y) e^{2\pi i(k_x x + k_y y)} d\varpi_x d\varpi_y \quad (4)$$

Se a função *f(x,y)* apresenta intensidades ao longo das coordenadas espaciais *x, y*, então w_x, w_y são freqüências espaciais que representam as variações de intensidade em relação às distâncias espaciais.

Na próxima seção é apresentada a versão discreta da Transformada de Fourier.

2.1.1.
Transformada de Fourier Discreta

Para uma função discreta de uma variável $f(x_k)$ dada para x_k inteiro e definido no intervalo $0 \le x_k < N$, sua Transformada de Fourier Discreta (DFT) $F(u)$ é dada por:

$$F(u) = \sum_{x_k=0}^{N-1} f(x_k) e^{-2\pi i u x_k / M} \tag{5}$$

Sendo sua inversa:

$$f(x_k) = \frac{1}{N} \sum_{x_k=0}^{N-1} F(u) e^{2\pi i u x_k / M} \tag{6}$$

A DFT bidimensional de uma função discreta de duas variáveis $f(x_k,y_j)$, dada para x_k inteiro e definido no intervalo $0 \le x_k < N$ e y_k inteiro e definido no intervalo $0 \le y_k < M$, é dada por:

$$F(u,v) = \sum_{y_k=0}^{M-1} \sum_{x_k=0}^{N-1} f(x_k,y_k) e^{-2\pi i (u x_k / N + v y_j / M)} \tag{7}$$

E sua inversa:

$$f(x_k,y_j) = \frac{1}{NM} \sum_{v=0}^{M-1} \sum_{u=0}^{N-1} F(u,v) e^{2\pi i (u x_k / N + v y_j / M)} \tag{8}$$

Uma imagem representada por uma função $f(x_k,y_k)$ real geralmente possui DFT complexa. Pode-se então ter uma representação em amplitude e fase da DFT:

$$A(u,v) = |F(u,v)| \tag{9}$$

$$\Phi(u,v) = \angle F(u,v) \tag{10}$$

$$F(u,v) = |F(u,v)| e^{i \angle F(u,v)}{}_W \tag{11}$$

Serão vistas a seguir algumas propriedades da Transformada de Fourier Discreta. A primeira propriedade apresentada é em relação ao comportamento da Transformada de Fourier para funções transladadas.

2.1.2.
Propriedade de translação da DFT

Translações espaciais na imagem em x_k ou y_k causam translações na fase da DFT:

$$f(x_k + a, y_k + b) \leftrightarrow F(u,v)e^{-i(au/N+bv/M)} \tag{12}$$

Perceba que tanto $F(u,v)$ quanto $f(x_k,y_j)$ são funções periódicas, portanto está implicitamente assumido que as translações são circulares.

É interessante verificar que a amplitude da DFT é invariável a translações na imagem. Isso quer dizer que duas imagens idênticas, porém transladadas, apresentariam a mesma amplitude após a Transformação de Fourier.

A próxima propriedade a ser vista é em relação a variações em escala na função a ser transformada. É interessante notar que variações em escala na função *f* são acompanhadas por variações semelhantes, porém inversas na magnitude de sua Transformada de Fourier.

2.1.3.
Propriedade de Escala da DFT

Outra propriedade importante da Transformada de Fourier é que variações em escala no domínio espacial causam variações inversas no domínio da freqüência:

$$f(\rho x_k, \alpha y_k) \leftrightarrow \frac{1}{\rho\alpha} F\left(\frac{u}{\rho}, \frac{v}{\alpha}\right) \tag{13}$$

Rotações na função *f* são acompanhadas por rotações na magnitude da Transformada de Fourier. Isto pode ser visto na seção seguinte.

2.1.4.
Propriedade de Rotação da DFT

Rotações na imagem $f(x_k,y_j)$ de um ângulo θ fazem com que a sua Transformada de Fourier possua a mesma rotação θ:

$$\begin{aligned} &f(x_k\cos\theta - y_j\sin\theta, x_k\sin\theta + y_j\cos\theta) \leftrightarrow \\ &F(u\cos\theta - v\sin\theta, u\sin\theta + v\cos\theta) \end{aligned} \tag{14}$$

Perceba que o *grid* é rotacionado, portanto o novo *grid* pode conter pontos não definidos. Os valores devem ser definidos então por interpolação.

Esta seção termina de apresentar a Transformada de Fourier. A seção seguinte apresentará a Transformada de Mellin e suas semelhanças com a Transformada de Fourier.

2.2. Transformada de Mellin

A Transformada de Mellin de uma função *f(x)* é definida em [23, 24] como:

$$M(s) = \int_0^\infty f(x) x^{s-1} dx \tag{15}$$

A partir desta definição pode-se perceber que a Transformada de Fourier de *f(exp ξ)* é a Transformada de Mellin de *M(iω)* de *f(x)* ao longo do eixo imaginário do plano complexo. Esta relação pode ser vista através das seguintes considerações:

$$x = e^{\xi} \tag{16}$$

$$dx = -e^{\xi} d\xi \tag{17}$$

Aplicando-se a eq.(16) e a eq. (17) na eq. (15) temos:

$$M(s) = \int_0^\infty f(x) x^{s-1} dx = \int_{-\infty}^\infty f(e^{-\xi}) e^{-s\xi} d\xi \tag{18}$$

Fazendo a substituição abaixo:

$$s = \theta + i^2 w \tag{19}$$

Temos (tipicamente θ = 0):

$$M(i\omega) = \int_{-\infty}^\infty f(e^{-\xi}) e^{-\sigma\xi} e^{-i2\pi\omega\xi} d\xi = \Im\left(f(e^{-\xi})\right) e^{-\sigma\xi} \tag{20}$$

Esta transformada possui a propriedade de ter amplitude invariável à escala:

$$f(ax) \leftrightarrow M(i\varpi) e^{-i(\varpi \ln(a))} \tag{21}$$

Isto pode ser verificado como a seguir. Caso se escale a função *f(x)* por um fator ρ, o seguinte resultado é obtido:

$$\int_0^\infty f(\rho x)x^{i\omega-1}dx = \int_0^\infty f(x)\left(\frac{x}{\rho}\right)^{i\omega-1}\frac{dx}{\rho} = \rho^{-i\omega}\int_0^\infty f(x)x^{i\omega-1}dx = \rho^{-i\omega}M(i\omega) \tag{22}$$

Portanto, a Transformada de Mellin de uma função escalada adquire translação em sua fase porém mantém sua amplitude. Isto pode ser visto pelo fato de:

$$z^c = e^{c\ln z} = e^{c(\ln|z|+i\arg z)} = e^{c\ln|z|}e^{ic\arg z} \tag{23}$$

Para $z = \rho$ e $c = -i\omega$, tem-se $arg(\rho) = 0$ e, portanto:

$$\rho^{-i\omega} = e^{-i\omega\ln|a|}e^0 = \cos(-\omega\ln|\rho|) + i\sin(-\omega\ln|\rho|)$$
$$\rho^{-i\omega}(x+iy) = \rho^{-i\omega}(re^{i\theta}) = e^{-i\omega\ln|\rho|}(re^{i\theta}) = re^{i(\theta-\omega\ln|\rho|)} \tag{24}$$

Isto corresponde somente a uma translação na fase dado que $|e^{i\theta}|=1$.

Outro modo de se verificar a invariância na amplitude da transformada é pensar na Transformada de Mellin como uma Transformada de Fourier de uma função $f_l(\xi)$ onde:

$$f_l(\xi) = f\left(e^{\xi}\right) = f(x) \tag{25}$$

Ou seja f_l é uma função que mapeia f da coordenada (x) para a coordenada (ξ):

$$\xi = \ln(x) \tag{26}$$

Desta maneira, uma variação em escala em (x) representa uma translação em (ξ):

$$(\rho x) \leftrightarrow (\xi + \ln\rho) \tag{27}$$

Como pode ser verificado por:

$$\xi' = \ln(\rho x) = \ln(\rho) + \ln(x) = \xi + \ln(\rho) \tag{28}$$

Portanto, como a Transformada de Fourier é invariável em amplitude à translação, e à variação em escala em $f(x)$ corresponde a uma translação em $f_l(\xi)$, pode-se concluir que a Transformada Fourier de $f_l(\xi)$ tem amplitude invariável para escalas em $f(x)$.

As Transformadas de Mellin do tipo 1 e 2 para imagens em 2 dimensões utilizam de propriedades dos mapeamentos *Log-Polar* e *Log-Log* que são apresentados nas seções 2.3 e 2.4 a seguir.

2.3. Mapeamento *Log-Polar*

Os mapeamentos de imagens para coordenadas distintas das originais são utilizados quando se deseja utilizar características específicas das novas coordenadas.

Serão apresentados aqui dois mapeamentos que possuem características interessantes quando aplicados junto à Transformada de Fourier. O primeiro a ser apresentado é o mapeamento *Log-Polar* que converte variações em escala e rotação para translações. O segundo é o mapeamentto *Log-Log* que converte dilatações horizontais e verticais em translações.

A representação *Log-Polar* de uma imagem é feita através da transformação das coordenadas $(x,y) \in \Re^2$ para as coordenadas (μ,θ) como definido em [18, 23] como a seguir:

$$x = e^{\mu}\cos\theta \tag{29}$$

$$y = e^{\mu} sen\theta \tag{30}$$

$$\mu = \ln\sqrt{x^2 + y^2} \tag{31}$$

$$\theta = \tan^{-1}\frac{y}{x} \tag{32}$$

Onde $\mu \in \Re^2$ e $0 \le \theta \le N$.

Aplicando-se a eq. (29) e a eq.(30) temos o mapeamento *Log-Polar* $f_p(\mu,\theta)$ de $f(x,y)$:

$$f_p(\mu,\theta) = f(x,y) = f\left(e^{\mu}\cos\theta, e^{\mu} sen\theta\right) \tag{33}$$

Este novo sistema de coordenadas é interessante devido ao fato de que escala e rotação são convertidas para translações.

Escala é convertida em translação na coordenada μ:

$$(\rho x, \rho y) \leftrightarrow (\mu + \ln\rho, \theta) \tag{34}$$

Como pode ser verificado abaixo:

$$x' = \rho x \tag{35}$$

$$y' = \rho y \tag{36}$$

$$\mu' = \ln\sqrt{\rho^2 x^2 + \rho^2 y^2} = \ln\rho\sqrt{x^2 + y^2} = \ln\rho + \ln\sqrt{x^2 + y^2} = \ln\rho + \mu w \tag{37}$$

$$\theta' = \tan^{-1}\frac{\rho y}{\rho x} = \theta \tag{38}$$

Rotação é convertida para translação na coordenada θ:

$$\left(x\cos(\delta) - y\,sen(\delta), x\,sen(\delta) + y\cos(\delta)\right) \leftrightarrow \left(\mu, \theta + \delta\right) \tag{39}$$

Como demonstrado a seguir:

$$x' = x\cos(\delta) - y\,sen(\delta) \tag{40}$$

$$y' = x\,sen(\delta) + y\cos(\delta) \tag{41}$$

$$\begin{aligned}
\mu' &= ln\sqrt{\left(x\cos(\delta) - y\,sen(\delta)\right)^2 + \left(x\,sen(\delta) + y\cos(\delta)\right)^2} = \\
&= ln\sqrt{\begin{matrix} x^2\cos(\delta)^2 + x^2\,sen(\delta)^2 + y^2\cos(\delta)^2 + y^2\,sen(\delta)^2 \\ +2xy\,sen(\delta)\cos(\delta) - 2xy\,sen(\delta)\cos(\delta) \end{matrix}} = \\
&= ln\sqrt{x^2 + y^2} = \mu
\end{aligned} \tag{42}$$

$$\begin{aligned}
\theta' &= tan^{-1}\frac{x\,sen\,\delta + y\cos\delta}{x\cos\delta - y\,sen\,\delta} = \\
&= tan^{-1}\frac{x\,sen\,\delta + x\,tan\,\theta\cos\delta}{x\cos\delta - x\,tan\,\theta\,sen\,\delta} = \\
&= tan^{-1}\frac{\cos\theta\,sen\,\delta + sen\,\theta\cos\delta}{\cos\theta\cos\delta - sen\,\theta\,sen\,\delta} = \\
&= tan^{-1}\frac{sen(\theta+\delta)}{\cos(\theta+\delta)} = tan^{-1}\,tan(\theta+\delta) = \theta + \delta
\end{aligned} \tag{43}$$

Estas conversões de escala e rotação para translação serão utilizadas junto à Transformada de Fourier para se obter transformadas invariáveis a estas características na seção 2.5. A seção seguinte apresenta outro mapeamento parecido com o descrito, mas que transforma variações em escala e rotação para variações em translação.

2.4. Mapeamento *Log-Log*

De modo similar ao mapeamento *Log-Polar*, pode-se fazer a transformação das coordenadas $(x,y) \in \Re^2$ para as coordenadas (ξ, η) como definido:

$$x = e^{\xi} \tag{44}$$

$$y = e^{\eta} \tag{45}$$

$$\xi = \ln x \tag{46}$$

$$\eta = \ln y \tag{47}$$

Onde $\xi \in \Re^2$ e $\eta \in \Re^2$.

Desta vez ao aplicarmos a eq. (44) e a eq. (45) encontramos o mapeamento *Log-Log* $f_l(\xi,\eta)$ de $f(x,y)$:

$$f_p(\xi,\eta) = f(x,y) = f\left(e^{\xi}, e^{\eta}\right) \tag{48}$$

Este mapeamento também possui propriedade interessante. Escalas independentes horizontais e verticais são traduzidas em respectivas translações:

$$(\rho x, \alpha y) \leftrightarrow (\xi + \ln\rho\rho, \eta + \ln\alpha) \tag{49}$$

Como pode ser demonstrado:

$$x' = \rho x \tag{50}$$

$$y' = \alpha y \tag{51}$$

$$\xi' = \ln\rho x = \ln\rho + \ln x = \ln\rho + \xi \tag{52}$$

$$\eta' = \ln\alpha y = \ln\alpha + \ln y = \ln\alpha + \eta \tag{53}$$

As propriedades apresentadas nesta seção e na seção anterior serão utilizadas na seção seguinte para se conseguir obter transformadas baseadas na Transformada de Fourier que possuam invariância à escala, rotação e dilatações alem da invariância à translação característica da própria Transformada de Fourier.

2.5. Obtendo Invariâncias à Rotação, Translação, Dilatações e Escala

Na presente seção os conceitos apresentados nas seções 2.1, 2.2, 2.3 e 2.4 serão utilizados para se definir as transformadas com invariâncias desejadas que serão aplicadas as imagens de modo a se poder fazer comparações entre imagens que representam uma mesma visão, mas com escalas, rotações, distorções e posições diferentes.

2.5.1. Invariância à Translação

Como foi apresentado na seção 2.1.4, a amplitude da Transformada de Fourier de uma imagem é invariável a translações espaciais circulares.

Veremos nas seções 2.5.2 e 2.5.3 que é possível aplicar esta invariância à translação de modo a cobrir mudanças em rotação, escala e escalas horizontais e verticais. Para isto será necessário trabalhar com os mapeamentos apresentados nas seções 2.3 e 2.4.

2.5.2. Invariância à Rotação e Escala

É possível se aplicar uma Transformada de Fourier de maneira peculiar para conseguir se obter invariância à escala e rotação.

Caso seja aplicada a Transformada de Fourier em uma imagem mapeada para o sistema de coordenadas *Log-Polar*, tem-se:

$$F(\varpi_x, \varpi_y) = \int_{-\infty}^{\infty}\int_{0}^{2\pi} f_p(\mu,\theta) e^{-2\pi i(\varpi_x \mu + \varpi_y \theta)} d\mu d\theta = \int_{-\infty}^{\infty}\int_{0}^{2\pi} f\left(e^{\mu}\cos\theta, e^{\mu} sen\theta\right) e^{-2\pi i(\varpi_x \mu + \varpi_y \theta)} d\mu d\theta \tag{54}$$

A Transformada de Fourier da representação *Log-Polar* de uma imagem é conhecida como Fourier-Mellin [23]. A Transformada de Fourier-Mellin 2D é dada por:

$$M_2(\varpi_1, \varpi_2) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y) \frac{e^{-2\pi i(\varpi_1 \ln\left(x^2+y^2\right) + \varpi_2 \tan^{-1}\frac{y}{x})}}{x^2+y^2} dxdy \tag{55}$$

Uma translação na coordenada μ representa uma mudança em tamanho no plano *x-y*, e uma translação na coordenada θ significa uma rotação no plano *x-y*. Portanto, como a amplitude da Transformada de Fourier é invariável à translação para as duas variáveis, ao aplicarmos a mesma sobre $f_p(\mu,\theta)$, conseguimos na verdade invariância quanto à escala e rotação no plano *x-y*. Doravante, escala e rotação em *f(x,y)* representam translações na fase da Transformada de Fourier Mellin:

$$f\left(\rho(x\cos\delta - y\sin\delta), \rho(x\sin\delta + y\cos\delta)\right) \leftrightarrow M(\varpi_1, \varpi_2) e^{-i(\varpi_1 \ln\rho + \varpi_2 \delta)} \tag{56}$$

É importante observar que ao se ganhar invariância à rotação e escala, se perde a invariância à translação em *x* e *y*.

A Transformada de Mellin apresentada é conhecida como Transformada de Mellin do tipo 2. A Transformada de Mellin do tipo 1 será apresentada na seção 2.5.3.

A Transformada de Mellin do tipo 2 pode ser discretizada levando-se em conta algumas considerações.

A função continua *f(x,y)* pode ser discretizado em uma matriz F de tamanho N x M cujos elementos são dados por $f_{k,j}$ para uma imagem com *pixels* de dimensões Δx e Δy:

$$f(x,y) \to f(x_k,y_j) \text{ quando } f_{k,j} \equiv f(x_k,y_j) \tag{57}$$

$$x_k = k\Delta x \tag{58}$$

$$y_j = j\Delta y \tag{59}$$

$$f_{k,j} \equiv f(k\Delta x, j\Delta y) \tag{60}$$

Com $k = 1,...,N$ e $k = 1,...,N$. A transformada continua $M_2(\omega_1, \omega_2)$ pode ser discretizada como uma matriz de tamanho U x V considerando-se intervalos de freqüência Δu/2π e Δv/2π para ω_1 e ω_2 respectivamente:

$$\omega_1 = \frac{u\Delta u}{2\pi} \tag{61}$$

$$\omega_2 = \frac{v\Delta v}{2\pi} \tag{62}$$

$$M_{(2)_{u,v}} = M_2[\{f_{k,j}\}_{k=1\ j=1}^{N\ M}](u,v) \tag{63}$$

$$M_{(2)_{u,v}} \equiv M_2(u \cdot \Delta u/2\pi, v \cdot \Delta v/2\pi) \tag{64}$$

Resultando em:

$$M_{(2)_{u,v}} = \sum_{j=1}^{M}\sum_{k=1}^{N} \frac{1}{(k\Delta x)^2 + (j\Delta y)^2} f_{k,j} e^{iu\Delta u \ln\left((k\Delta x)^2+(j\Delta y)^2\right)} e^{iv\Delta v \tan^{-1}\left(\frac{j\Delta y}{k\Delta x}\right)} \tag{65}$$

A seção seguinte apresentará a Transformada de Mellin do tipo 1 e como esta possui invariância a dilatações horizontais e verticais.

2.5.3. Invariância a Dilatações Horizontais e Verticais

De modo análogo ao apresentado na seção 2.5.2, pode-se conseguir invariância a dilatações horizontais e verticais.

Se a Transformada de Fourier for aplicada em uma imagem mapeada para o sistema de coordenadas *Log-Log*, obtém-se:

$$F(\varpi_x,\varpi_y)=\int_{-\infty}^{\infty}\int_{0}^{2\pi} f_l(\xi,\eta)e^{-2\pi i(\varpi_x\xi+\varpi_y\eta)}d\xi d\eta = \int_{-\infty}^{\infty}\int_{0}^{2\pi} f(e^{\xi},e^{\eta})e^{-2\pi i(\varpi_x\xi+\varpi_y\eta)}d\xi d\eta \tag{66}$$

Ao se considerar:

$$d\xi=-\frac{1}{x}dx \tag{67}$$

$$d\eta=-\frac{1}{y}dy \tag{68}$$

Esta transformada pode ser reescrita como a seguir, sendo conhecida como Transformada de Mellin do tipo 1:

$$M_1(\omega_1,\omega_2)=\int_0^{\infty}\int_0^{\infty} f(x,y)x^{i2\pi w_1-1}y^{i2\pi\omega_2-1}dxdy \tag{69}$$

$$M_1(\omega_1,\omega_2)=\int_0^{\infty}\int_0^{\infty}\frac{1}{xy}f(x,y)e^{i2\pi(\omega_1\ln x+\omega_2\ln y)}dxdy \tag{70}$$

Desta vez, uma dilatação no eixo *x* é traduzida para uma translação em ξ e uma dilatação no eixo y em uma translação em η. Portanto, a Fourier de $f_l(\xi,\eta)$ acaba sendo invariável em amplitude a dilatações em *x* e *y*, estas representam translações na fase da Transformada de Mellin:

$$f(\rho x,\alpha y)\leftrightarrow M_1(\varpi_1,\varpi_2)e^{-i(\varpi_1\ln\rho+\varpi_2\ln\alpha)} \tag{71}$$

Como fazer uma variação em escala na função *f(x,y)* significa igualar as dilatações ρ e η, pode-se dizer que a amplitude da Transformada de Mellin do tipo 1 é invariável à escala:

$$f(\rho x,\rho y)\leftrightarrow M_1(\varpi_1,\varpi_2)e^{-i(\varpi_1\ln\rho+\varpi_2\ln\rho)} \tag{72}$$

Para se discretizar 0 basta aplicar as considerações vistas da eq. (57) à eq. (64) apresentadas obtendo-se:

$$M_{(1)_{u,v}}=\sum_{j=1}^{M}\sum_{k=1}^{N}\frac{1}{k\Delta x}e^{iu\Delta u\ln(k\Delta x)}f_{k,j}\frac{1}{j\Delta y}e^{iv\Delta v\ln(j\Delta y)} \tag{73}$$

A Transformada de Mellin 2D do tipo 1 pode ser escrita de forma matricial como se segue:

$$Tx_{u,k} = \frac{1}{k} e^{iu\Delta u \ln(k\Delta x)} \tag{74}$$

$$Ty_{j,v} = \frac{1}{j} e^{iv\Delta v \ln(j\Delta y)} \tag{75}$$

$$M_1 = Tx \cdot F \cdot Ty \tag{76}$$

Onde:

$$T_x = \frac{1}{\Delta x}\begin{bmatrix} e^{i\Delta u \ln \Delta x} & \frac{1}{2}e^{i\Delta u \ln 2\Delta x} & \dots & \frac{1}{N}e^{i\Delta u \ln N\Delta x} \\ e^{i2\Delta u \ln \Delta x} & \frac{1}{2}e^{i2\Delta u \ln 2\Delta x} & \dots & \frac{1}{N}e^{i2\Delta u \ln N\Delta x} \\ \dots & \dots & \dots & \dots \\ e^{iU\Delta u \ln \Delta x} & \frac{1}{2}e^{iU\Delta u \ln 2\Delta x} & \dots & \frac{1}{N}e^{iU\Delta u \ln N\Delta x} \end{bmatrix} \tag{77}$$

$$T_y = \frac{1}{\Delta y}\begin{bmatrix} e^{i\Delta v \ln \Delta y} & e^{i2\Delta v \ln \Delta y} & \dots & e^{iV\Delta v \ln \Delta y} \\ \frac{1}{2}e^{i\Delta v \ln 2\Delta y} & \frac{1}{2}e^{i2\Delta v \ln 2\Delta y} & \dots & \frac{1}{2}e^{iV\Delta v \ln 2\Delta y} \\ \dots & \dots & \dots & \dots \\ \frac{1}{M}e^{i\Delta v \ln M\Delta y} & \frac{1}{M}e^{i2\Delta v \ln M\Delta y} & \dots & \frac{1}{M}e^{iV\Delta v \ln M\Delta y} \end{bmatrix} \tag{78}$$

A próxima seção explica a propriedade comutativa entre rotação e dilatação.

2.5.4. Propriedade Comutativa Entre Rotação e Dilatação

É interessante apontar que transformações em rotação dadas por $R(\theta)$ e transformações em dilatação dadas por $S(\rho,\alpha)$ comutam:

$$\begin{aligned} &R(\theta)S(\rho,\alpha)f(x,y) = R(\theta)f(\rho x,\alpha y) \\ &= f(\rho x\cos\theta - \rho y sen\theta, \alpha x sen\theta + \alpha y\cos\theta) \\ &= S(\rho,\alpha)f(x\cos\theta - y sen\theta, x sen\theta + y\cos\theta) \\ &= S(\rho,\alpha)R(\theta)f(x,y) \end{aligned} \tag{79}$$

Do mesmo modo transformações em escala dadas por $S(\rho)$ poderiam ser usadas no lugar de $S(\rho,\alpha)$ para mostrar que $S(\rho)$ e $R(\theta)$ comutam.

Similarmente, pode-se mostrar que transformações em translação $T(\kappa;\tau)$ não comutam nem com $S(\rho)$ $S(\rho,\alpha)$ nem $R(\theta)$.

Invariâncias à translação, rotação e escala por uma mesma transformada serão apresentadas a seguir utilizando a propriedade comutativa aqui mostrada.

2.5.5.
Invariância à Translação, Rotação e Escala Simultâneas

Considere os dois operadores invariáveis F e F_{M2} apresentados a seguir:

$$F \circ f(x,y) = |F[f(x,y)](u,v)| \tag{80}$$

$$F_{M2} \circ f(x,y) = |M_2[f(x,y)](u,v)| \tag{81}$$

Como pode ser visto, F, extrai o modulo da Transformada de Fourier e F_{M2} extrai o módulo da Transformada de Mellin do tipo 2.

Aplicando-se o operador híbrido $F \circ F_{M2}$ a uma imagem *f(x,y)* obtém-se:

$$I_1 = [F_{M2} \circ F] f(x,y) \tag{82}$$

Ao se aplicar o mesmo operador para uma imagem que foi transladada, rotacionada e escalada tem-se:

$$I_2 = [F_{m2} \circ F \circ R(\theta) S(\rho) \circ T(\kappa,\tau)] f(x,y) \tag{83}$$

$$= \left[F_{M2} \circ R(\theta) \frac{1}{\rho^2} S\left(\frac{1}{\rho}\right) \circ F \circ T(\kappa,\tau) \right] f(x,y) \tag{84}$$

$$= \frac{1}{\rho^2} [F_{M2} \circ F] f(x,y) \tag{85}$$

$$= \frac{1}{\rho^2} I_1 \tag{86}$$

Portanto I_2 é igual a I_1 multiplicado por um fator $1/\rho^2$ e a representação é invariável à escala, translação e rotação. Os passos 0. (83) e na eq. (84) se devem às propriedades apresentadas nas seções 2.1.3 e 2.1.4. A contração apresentada na eq. (85) se deve a propriedade de invariância de F e F_{M2}.

As invariâncias aqui descritas são suficientes para lidar com qualquer combinação ou permutação quanto à rotação, escala e translação em qualquer ordem.

De modo semelhante ao aqui apresentado, será apresentado adiante como se obter invariância à translação e dilatações verticais e horizontais.

2.5.6.
Invariância à Translação e Dilatações Verticais e Horizontais Simultâneas

Considere o operador F apresentado na seção 2.5.5 e o operador F_{M1} a seguir:

$$F_{M1} \circ f(x,y) = \left| M_1[f(x,y)](u,v) \right| \tag{87}$$

Aqui F_{M1} extrai o módulo da Transformada de Mellin do tipo 1 e F o módulo da Transformada de Fourier.

Aplicando-se o operador híbrido $F \circ F_{M1}$ a uma imagem *f(x,y)* obtém-se:

$$I_3 = [F_{M1} \circ F] f(x,y) \tag{88}$$

Ao se aplicar o mesmo operador para uma imagem que foi transladada e dilatada nos eixos x e y, tem-se:

$$I_4 = [F_{m1} \circ F \circ R(\theta) S(\rho,\alpha) \circ T(\kappa,\tau)] f(x,y) \tag{89}$$

$$= \left[F_{M1} \circ R(\theta) \frac{1}{\rho\alpha} S\left(\frac{1}{\rho\alpha} \right) \circ F \circ T(\kappa,\tau) \right] f(x,y) \tag{90}$$

$$= \frac{1}{\rho\alpha} [F_{M1} \circ F] f(x,y) \tag{91}$$

$$= \frac{1}{\rho\alpha} I_4 \tag{92}$$

Novamente encontra-se I_3 proporcional a I_4 e agora a representação é invariável às dilatações horizontal e vertical e à translação.

As transformadas invariáveis desenvolvidas serão utilizadas adiante para se comparar imagens.

2.6.
Comparando Imagens

Todos os métodos apresentados para se encontrar funções invariáveis para imagens tem como fim a identificação e classificação das imagens. Para tal deseja-se comparar tais funções para diferentes imagens de modo a se obter uma função de similaridade entre estas.

Aqui serão apresentadas duas medidas de similaridade para se fazer as comparações: distância euclidiana e correlação. Então, é explicado com usar estas funções para comparar as imagens.

Por fim, é proposto um método para se comparar imagens obtidas em diferentes distâncias para uma mesma cena chamado de *Window Growing*.

2.6.1. Distância Euclidiana

Uma medida padrão de similaridade entre duas funções *f(x,y)* e *g(x,y)* é dada pela distância Euclidiana entre estas:

Error! Objects cannot be created from editing field codes. (93)

Quanto maior a distância *d*, menor a semelhança entre as funções.

Repare que esta função tem a desvantagem de ser extremamente sensível a grandes magnitudes para valores pontuais das funções. Além disso, esta distância não é normalizada para limites dados.

Outra medida de similaridade muito comum é a correlação, esta será descrita a seguir.

2.6.2. Correlação

A Correlação entre duas funções *f(x,y)* e *g(x,y)* é definida de sua maneira mais geral pela integral contínua:

$$C(x',y') = \iint f(p,q)g(p+x',q+y') \tag{94}$$

Sua forma discreta é dada por:

$$C(x',y') = \sum\sum f(p,q)g(p+x',q+y') \tag{95}$$

Sua versão normalizada é vista como se segue:

$$C(x',y') = \frac{\sum\sum f(p,q)g(p+x',q+y')}{\sqrt{\sum\sum f(p,q)^2 \sum\sum g(p,q)^2}} \tag{96}$$

Quando *x' = 0* e *y' = 0*, esta correlação é feita para as funções sobrepostas. É, conseqüentemente, uma medida de similaridade entre as funções.

A correlação normalizada pode ser entendida como uma multiplicação vetorial para vetores normalizados. Portanto, é uma boa medida de similaridade entre duas funções por não ser tão sensível a grandes magnitudes para valores pontuais como a distância Euclidiana.

A correlação normalizada é igual a 1 para funções idênticas, igual a 0 para funções completamente diferentes e igual a –1 para funções opostas, ou seja, quando $f = -g$;

O valor de máximo da eq. (96) indica x' e y' que apresentam a melhor sobreposição entre as funções. Este valor é invariável a translações.

Para se correlacionar duas imagens com tamanhos diferentes, a imagem a ser comparada pode ser redimensionada para o tamanho da imagem base.

O uso da distância euclidiana e da correlação para se comparar imagens tendo invariância à escala, dilatações, rotações e translações é apresentado adiante.

2.6.3. Comparando Imagens com Invariâncias:

Aplicando-se a correlação ou à distância euclidiana nas diversas transformadas das imagens, é possível fazer comparações invariáveis a vários fatores. Verifique as possibilidades a seguir:

- Máximo da correlação entre duas imagens: Invariância à translação;
- Distância ou correlação entre magnitude da Transformada de Fourier de duas imagens: Invariância à translação;
- Distância ou correlação entre magnitude da Transformada de Mellin do tipo 1 de duas imagens: Invariância a distorções (dilatações);
- Distância ou correlação entre magnitude da Transformada de Mellin do tipo 2 de duas imagens: Invariância à escala e rotação;
- Distância ou correlação entre magnitude da Transformada de Mellin do tipo 1 da magnitude da Transformada de Fourier de duas imagens: Invariância a distorções e translação;
- Distância ou correlação entre magnitude da Transformada de Mellin do tipo 2 da magnitude da Transformada de Fourier de duas imagens: Invariância à translação, escala e rotação;

A seguir é explicado um método interessante para se fazer comparações que será utilizado para auxiliar a navegação de um robô para um lugar conhecido.

2.6.4
Window Growing

Em determinadas situações, deseja-se encontrar uma imagem em uma sub-janela de outra imagem. Este é o caso de um robô observando um mesmo fundo, com a mesma direção, porém para distâncias diferentes. Para tal é proposta uma técnica de "*window growing"* onde uma das imagens será sub-dividida em várias sub-imagens, para então, fazer a comparação.

Na Figura 2-1 ilustra como isto é feito. O retângulo à esquerda representa a imagem de referência que será procurada na outra imagem, representada pelo retângulo à direita. Perceba que a imagem à direita possui várias sub-divisões, ou melhor, sub-janelas. Estas sub-janelas serão comparadas com a imagem de referência.

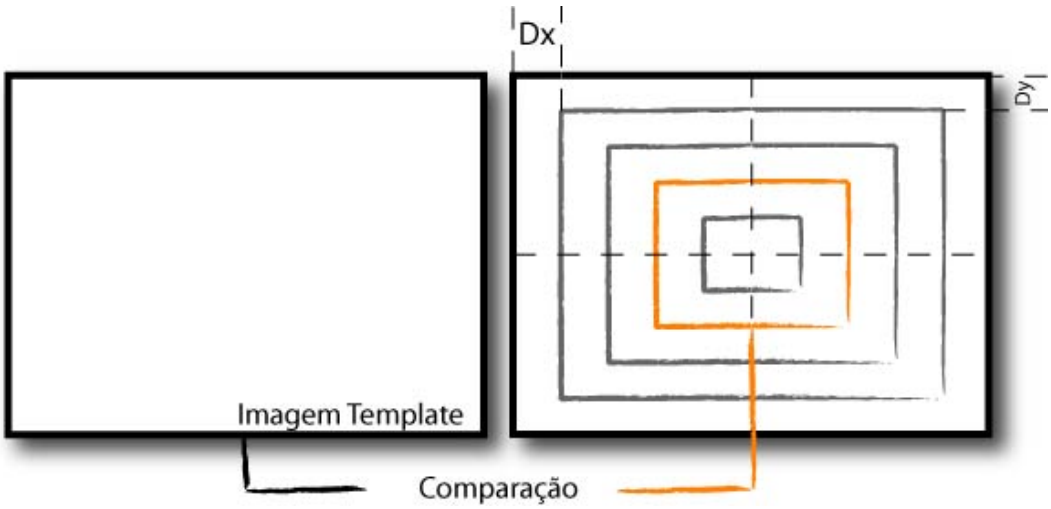

Figura 2-1: *Window Growing*

A figura na qual se deseja procurar a imagem base é subdividida em um número *n* de janelas com centro comum. As diferenças de tamanhos *Dx* e *Dy* entre uma janela e outra são dadas por:

$$Dx = sx/2n \tag{97}$$

$$Dy = sy/2n \tag{98}$$

Onde *sx* é o tamanho horizontal da imagem a ser comparada e *sy* o tamanho vertical.

Perceba que apesar das janelas a serem comparadas terem tamanhos diferentes, algumas transformadas apresentadas podem ser obtidas de modo que as

funções obtidas tenham tamanhos semelhantes. Isto pode ser feito para qualquer comparação que antes de ser feita passe por uma Transformada de Mellin do tipo 1 ou 2.

Para se comparar através de correlação ou Fourier, pode-se fazer o redimensionamento da imagem base para então fazer a comparação.

Por fim, tem-se uma série de comparações, uma para cada sub-janela e pode-se encontrar a janela com maior semelhança com a imagem base.

De modo a não ser necessário ter *n* muito grande para se encontrar uma sub-janela bem próxima da imagem base, pode-se fazer uma segunda tomada de comparações criando-se novas sub-imagens ao redor daquela que obteve melhor comparação.

Isto está exemplificado na Figura 2-2. Entre as duas sub-janelas ao redor daquela que obteve maior valor de comparação são sub-divididas novas *m* janelas e o processo é retomado.

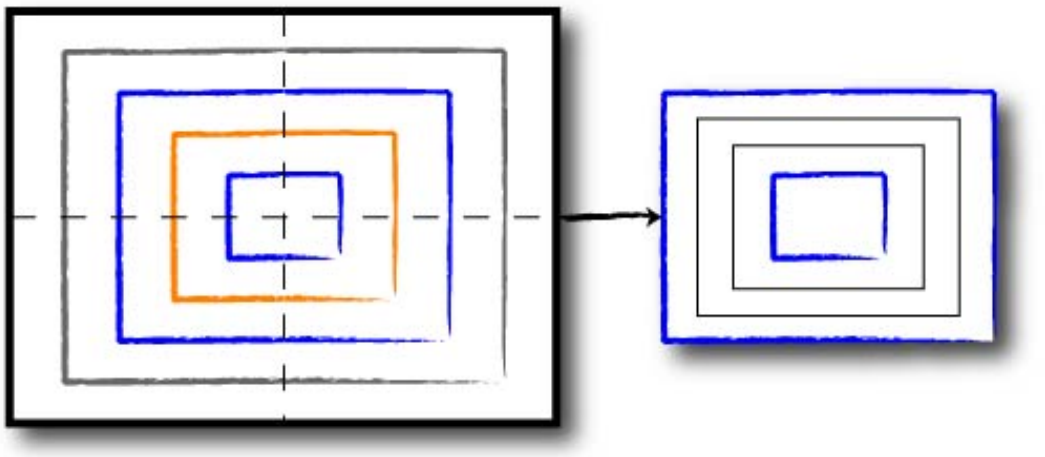

Figura 2-2: Nova tomada de comparações

Este método apresentado pode ser feito iterativamente o número de vezes necessário até que se chegue em um intervalo entre janelas desejado. O diagrama de fluxo do procedimento é dado na Figura 2-3.

O próximo capítulo apresentará uma técnica que será utilizada para se encontrar regiões locais em comum entre imagens.

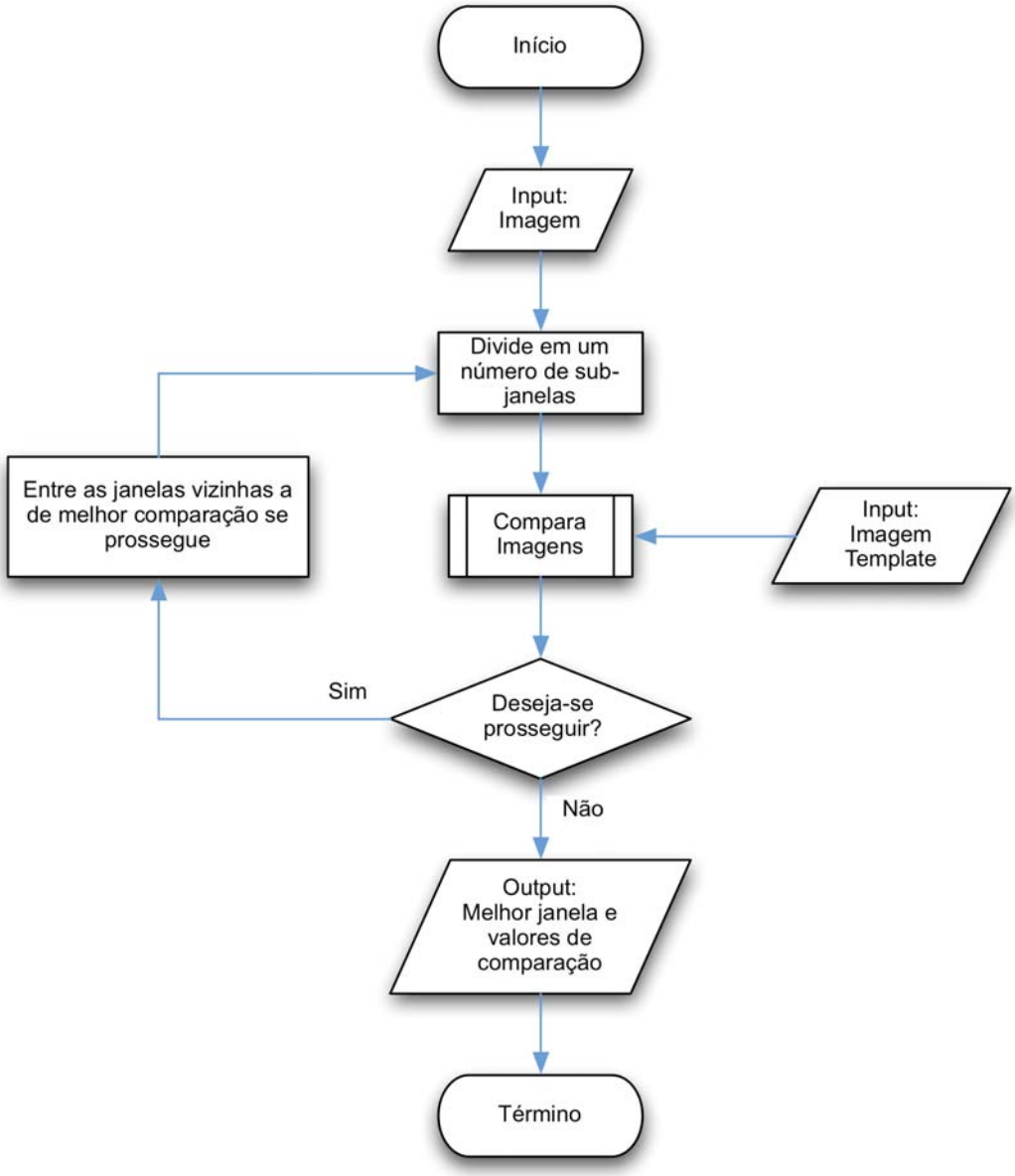

Figura 2-3: Diagrama de fluxo do procedimento de *Window Growing*

3
Transformação SIFT (Scale Invariant Feature Transform)

Este capítulo apresenta as seguintes seções:

- 3.1 – Uma Introdução Sobre Descritores Locais: A técnica SIFT ("Scale Invariant Feature Transform") é utilizada para se construir descritores. Nesta seção é apresentado o que são descritores e um panorama do uso de descritores em diversos problemas de visão computacional;
- 3.2 – Descrição do SIFT: Esta seção apresenta a técnica SIFT, apresenta trabalhos recentes que utilizam da técnica e, então, descreve sua teoria e como é aplicada. Esta seção é dividida em:
    - 3.2.1 – Introdução;
    - 3.2.2 – Detecção de Extremos;
    - 3.2.3 – Localização Exata de Pontos Chave;
    - 3.2.4 – Atribuição da Orientação dos Descritores;
    - 3.2.5 – Construção do Descritor Local;
    - 3.2.6 – Encontrando os Pontos em Comum;

3.1.
Uma Introdução Sobre Descritores Locais

Correspondência de imagens é fundamental em diversos problemas de visão computacional como reconhecimento de objetos, reconhecimento de cenas, montagem automática de mosaicos, obtenção da estrutura 3D de múltiplas imagens, correspondência estéreo e perseguição de movimentos. Uma abordagem para se trabalhar com correspondência de imagens é se usar descritores locais para se representar uma imagem. Descritores são vetores de características de uma imagem ou de determinadas regiões de uma imagem e podem ser usados para se comparar regiões em imagens diferentes. Este vetor de características é normalmente formado por descritores locais ou globais. Descritores locais computados em pontos de interesse provaram ser bem sucedidos em aplicações

como correspondência e reconhecimento de imagens [25, 31]. Descritores são distintos, robustos à oclusão e não requerem segmentação.

Existem diversas técnicas para se descrever regiões locais em uma imagem [31]. O mais simples descritor é um vetor com as intensidades dos *pixels* da imagem. A medida de correlação cruzada pode ser então usada para computar a similaridade entre duas regiões como apresentado na seção 2.6.2. Porém, a alta dimensionalidade de tal descritor aumenta a complexidade computacional da comparação. Então, esta técnica é principalmente usada para se encontrar correspondências ponto a ponto entre duas imagens. A vizinhança de um ponto também pode ser escalada de modo a reduzir sua dimensão. Outro descritor simples é a distribuição de intensidades de uma região representada por seu histograma.

Trabalhos recentes têm se concentrado em fazer descritores invariáveis a transformações nas imagens. Mikolajczyk e Schmid [30] propuseram um detector de pontos de interesse invariável a transformações afins através da combinação de um detector invariável à escala e da técnica "*second moment of Harris corners"* [32]. Ling e Jacobs [33] propuseram um sistema para a construção de descritores de intensidade locais invariáveis a deformações em geral. Em [34], Gotze, Drue e Hartmann apresentam descritores baseados na Transformada de Fourier-Mellin de regiões locais. Lowe [25-29] propôs uma maneira rápida e eficiente de computar características invariáveis a transformações em escala, que medem a distribuição do gradiente em regiões detectadas invariáveis à escala.

A seção a seguir apresentará os descritores SIFT e detalhará seu uso.

3.2. Descrição da técnica SIFT

3.2.1. Introdução

SIFT ("*Scale Invariant Feature Transform"*) é uma técnica de processamento de imagens que permite a detecção e extração de descritores locais, razoavelmente invariáveis a mudanças de iluminação, ruído de imagem, rotação, escala e pequenas mudanças de perspectiva. Estes descritores podem ser

utilizados para se fazer a correspondência de diferentes visões de um objeto ou cena.

Descritores obtidos com a técnica SIFT são altamente distintos, ou seja, um determinado ponto pode ser corretamente encontrado com alta probabilidade em um banco de dados extenso com descritores para diversas imagens.

Um aspecto importante da técnica SIFT é a geração de um número grande de descritores que conseguem cobrir densamente uma imagem quanto a escalas e localizações. A quantidade de descritores é particularmente importante para o reconhecimento de objeto, onde a capacidade de se encontrar pequenos objetos em ambientes desordenados requer ao menos 3 pontos encontrados em comum para uma identificação confiável.

A obtenção de descritores SIFT é feita através das seguintes etapas:

- Detecção de extremos: Nesta primeira etapa é feita procura para todas escalas e localizações de uma imagem. Isto é feito utilizando-se a diferença de filtros gaussianos de modo a se identificar pontos de interesse invariáveis à escala e rotação. A detecção de extremos é descrita na seção 3.2.2;
- Localização de pontos chave: Para cada localização em que foi detectado um extremo, um modelo detalhado é ajustado de modo a se determinar localização e escala. Pontos chaves, ou pontos de interesse, são então selecionados baseando-se em medidas de estabilidade. A localização dos pontos chaves é apresentada na seção 3.2.3;
- Definição de orientação: É definida a orientação de cada ponto chave através dos gradientes locais da imagem. Toda operação a partir de então será feita com relação a dados da imagem transformados em relação à orientação, escala e localização de cada ponto chave. Desta maneira se obtém invariância a estas transformações. A atribuição da orientação de cada descritor é vista na seção 3.2.4;
- Descritor dos pontos chaves: Nesta etapa é feita a construção dos descritores ao se medir Gradientes locais em uma região vizinha a cada ponto de interesse. Estas medidas são então transformadas para uma representação que permite níveis significativos de distorção e

mudança na iluminação. A construção dos descritores é apresentada em 3.2.5;

Em tarefas de comparação de imagens e reconhecimento, descritores SIFT são extraídos das imagens para então poderem ser comparados.

Na próxima seção será descrito o primeiro passo da obtenção de descritores SIFT.

3.2.2. Detecção de Extremos

A primeira etapa da técnica SIFT é detectar extremos (máximos e mínimos) em uma pirâmide da imagem convoluída com a função Diferença de Gaussiana. Pontos chave correspondem a estes extremos para diferentes escalas. Esta etapa está descrita na presente seção.

A convolução de uma função *f(x,y)* com uma função *h(x,y)* é dada por:

$$f(x,y) * h(x,y) = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m,n) h(x-m, y-n) \tag{99}$$

Onde *x* varia de 1 a M e *y* varia de 1 a N.

Um filtro Gaussiano passa baixa é dado pela convolução de uma imagem *I* com a função *G*:

$$L(x,y,\sigma) = G(x,y,\sigma) * I(x,y) \tag{100}$$

Onde:

$$G(x,y,\sigma) = \frac{1}{2\pi\sigma^2} e^{-\left(x^2+y^2\right)/2\sigma^2} \tag{101}$$

Perceba que este filtro é variável à escala através do parâmetro σ.

A função DoG (*"Difference of Gaussian"*) é dada pela diferença de imagens filtradas em escalas próximas separadas por uma constante *k*. A função DoG é definida por:

$$\text{DoG} = G(x,y,k\sigma) - G(x,y,\sigma) \tag{102}$$

O resultado de fazer a convolução de uma imagem com o filtro DoG é dado por:

$$\begin{aligned} D(x,y,\sigma) &= \left(G(x,y,k\sigma) - G(x,y,\sigma)\right) * I(x,y) \\ &= L(x,y,k\sigma) - L(x,y,\sigma) \end{aligned} \tag{103}$$

ou seja, é a diferença entre imagens borradas por um filtro gaussiano em escalas σ e $k\sigma$. Este filtro consegue detectar variações de intensidade na imagem, tais como contornos. Perceba que variando-se σ, é possível encontrar descritores para variações em diferentes escalas espaciais.

Nas Figura 3-1 e Figura 3-2 podem-se ver alguns exemplos de aplicação dos filtros descritos.

Figura 3-1: Imagem após filtro gaussiano com σ igual a 1.6, 2.4 e 3.2

Figura 3-2: Filtro DoG para imagens apresentadas na Figura 3-1

Um modo eficiente de se construir $D(x,y,\sigma)$ é apresentado na Figura 3-3. Deseja-se construir s intervalos, onde cada intervalo representa uma imagem filtrada por DoG intervalar entre duas outras. Para se construir s intervalos serão

necessárias s+3 imagens na pilha apresentada pelas imagens superiores da Figura 3-3. A imagem inicial é convoluída progressivamente com funções gaussianas para produzir mais $s+2$ imagens separadas por um fator constante k. A imagem é inicialmente filtrada por filtro Gaussiano com escala σ. A partir de então, são geradas imagens que são progressivamente convoluídas. Cada nova imagem é filtrada com escala k vezes a escala utilizada anteriormente. Para cada duas imagens, pode-se produzir a diferença de Gaussianas D através da subtração de duas imagens consecutivas na pilha de imagens L.

Para exemplificar, imagine que se deseja gerar apenas um intervalo. Serão necessárias, então, quatro imagens na pilha superior (a pilha das imagens filtradas com Gaussianas). Estas imagens serão:

- A imagem original;
- A imagem original filtrada por Gaussiana com escala σ;
- Outras duas imagens filtradas com escalas multiplicadas por k: kσ e k2σ;

Lowe considera em [25] que é necessário fazer a convolução da imagem até 2σ para ser possível a construção de descritores invariáveis à escala. Portanto, para se gerar s intervalos é definido:

$$k = 2^{1/s} \tag{104}$$

Desta maneira, teremos s intervalos produzidas por DoG, sendo que o primeiro é dado por $D(x,y,\sigma)$ e a ultima imagem da pilha de DoG dada por $D(x,y,2\sigma)$.

Para melhor entendimento, perceba que na Figura 3-3, a primeira imagem acima à esquerda é $I(x,y)$ e a última imagem acima é $L(x,y,k2\sigma)$. A primeira imagem abaixo é $D(x,y,0)$ e a última é $D(x,y,2\sigma)$. O intervalo é dado por $D(x,y,\sigma)$.

O processo apresentado gera o que é chamado de uma oitava. Este processo é repetido para um número desejado de oitavas. Cada oitava representa um conjunto de imagens L e D para a imagem reescalada com diferentes amostragens.

Isto funciona da seguinte forma: quando uma oitava tiver sido processada, a imagem Gaussiana que possui 2σ (corresponde à penúltima imagem da pilha superior na Figura 3-3) é re-amostrada para a metade de seu tamanho. Esta será a

primeira imagem da próxima oitava. Cada oitava produz o mesmo número de intervalos.

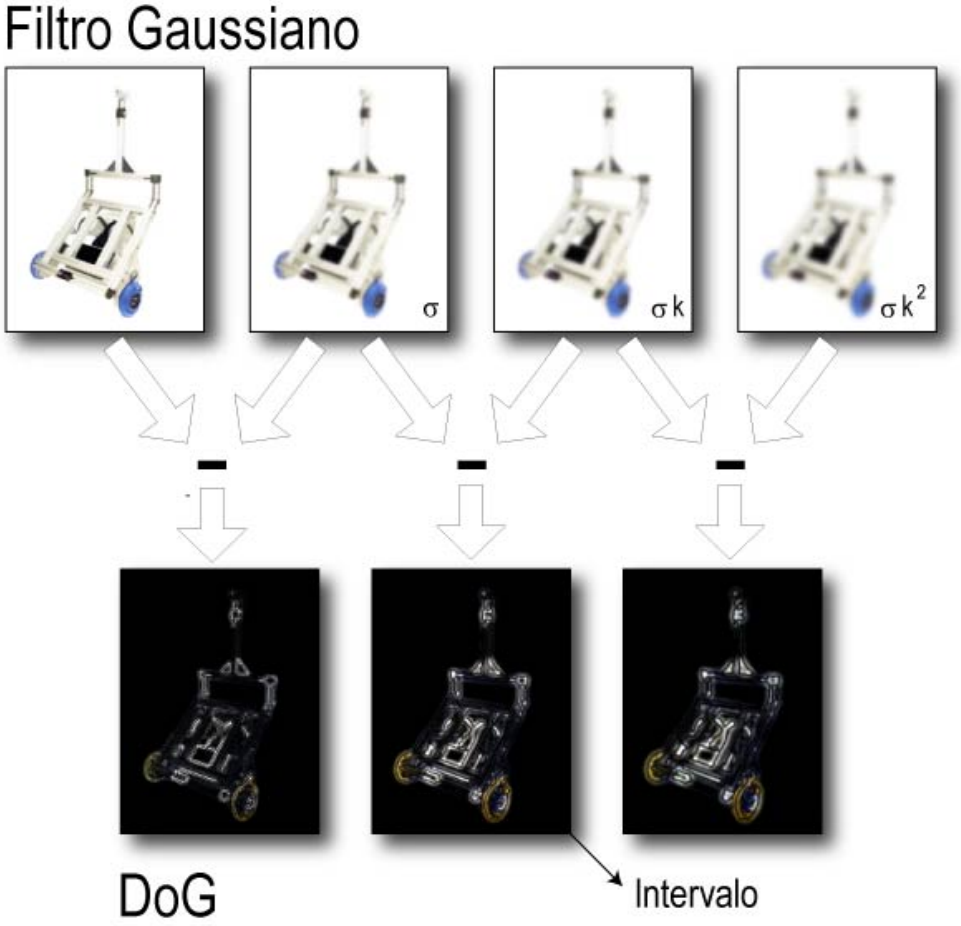

Figura 3-3: Construção das Diferenças de Gaussiana

A geração de oitavas está exemplificada na Figura 3-4.

A partir de agora será feita a detecção de extremos em cada intervalo de cada oitava. Os extremos são dados por valores locais de máximo ou mínimo para cada $D(x, y, \sigma)$ que corresponda a um intervalo. Cada ponto é comparado aos seus oito vizinhos na imagem atual, mais seus nove vizinhos na escala superior e nove vizinhos na escala inferior.

As escalas superior e inferior são correspondentes às imagens vizinhas em uma mesma oitava para a pilha de imagens DoG. Não confunda escala superior e inferior com oitavas onde a amostragem das imagens gera imagens em escalas diferentes. Quando se diz escala superior e inferior aqui, está se fazendo referência à σ.

O procedimento de detecção está exemplificado na Figura 3-5. No exemplo, o ponto marcado como *X* é comparado com seus vizinhos marcados como *O*. As 3 imagens DoG apresentadas são 3 intervalos vizinhos em uma pilha.

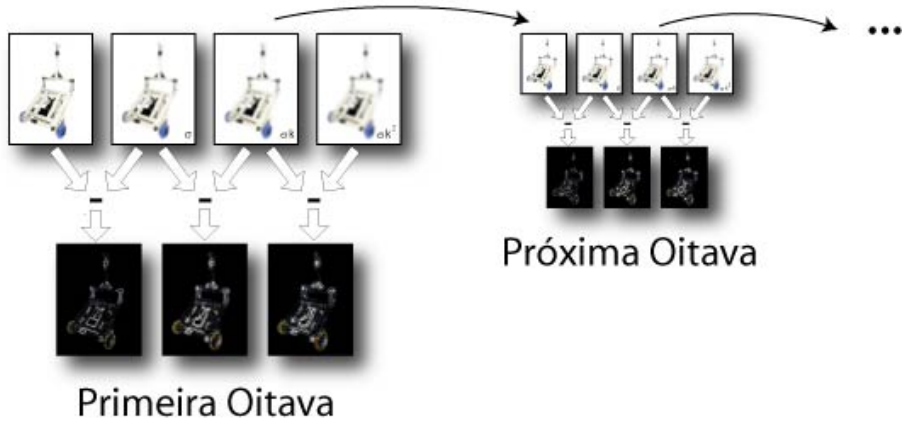

Figura 3-4: Formação das Oitavas

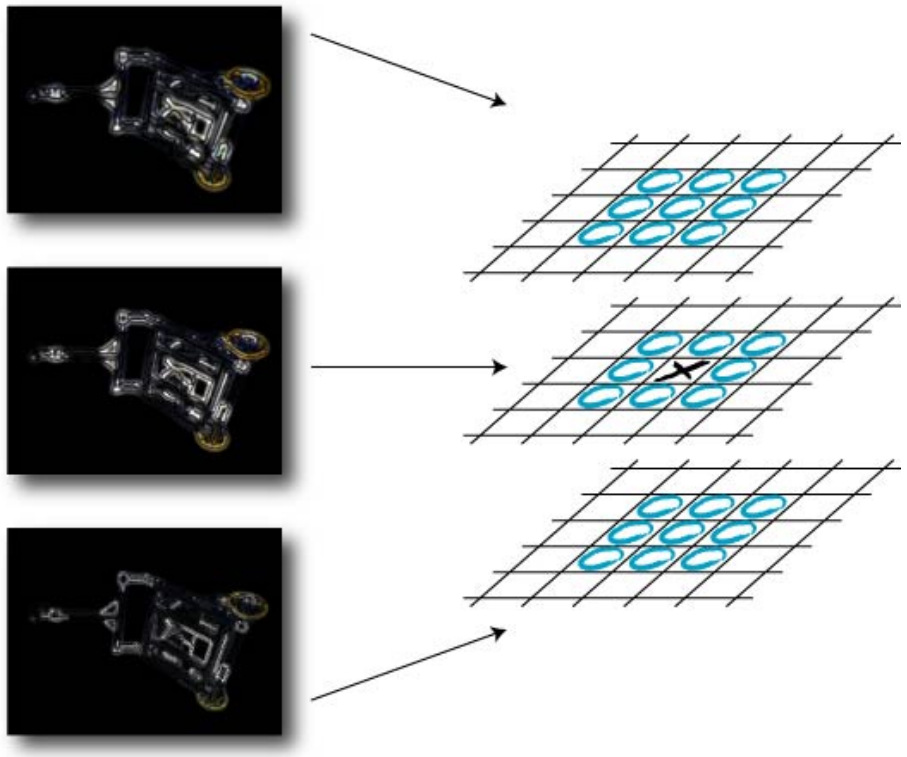

Figura 3-5: Detecção de Máximos e Mínimos

A próxima etapa é definir a localização exata dos pontos chave e fazer o descarte de pontos chave instáveis. Isto será visto na próxima seção.

3.2.3.
Localização Exata de Pontos Chave

Todos os pontos detectados como extremos são possíveis pontos chave. Deseja-se agora calcular a localização e escala Gaussiana detalhadas de cada um destes pontos. Recalcular a localização e escala interpoladas dos pontos de máximo traz melhoria para à técnica. A localização dos pontos chaves como será apresentada é especialmente importante para as últimas oitavas, porque o espaçamento amostral destas representa grandes distâncias na imagem base.

O método consiste em enquadrar uma função quadrática 3D do ponto de amostragem local de modo a determinar uma localização interpolada do máximo.

Para cada ponto analisado é utilizada uma expansão de Taylor da função $D(x,y,\sigma)$ transladada de modo que a origem desta expansão esteja localizada no ponto:

$$D(\bar{x}) = D + \frac{\partial D^T}{\partial \bar{x}}\bar{x} + \frac{1}{2}\bar{x}^T \frac{\partial^2 D}{\partial \bar{x}^2}\bar{x}\ldots \tag{105}$$

Onde:

$$D = D(x,y,\sigma) \tag{106}$$

$$D(\bar{x}) = D(x+x',y+y',\sigma+\sigma') \tag{107}$$

Esta equação deve ser entendida da seguinte maneira, D e suas derivadas são avaliados a partir do ponto analisado e $\bar{x} = (x',y',\sigma')^T$ é o *offset* em relação a este ponto. Ou seja, D é o valor da função D(x,y,σ) no ponto avaliado, $\bar{x}$ é o *offset* em relação a este ponto e $D(\bar{x})$ é a aproximação do valor de D(x,y,σ) interpolado para um ponto transladado com *offset* $\bar{x}$.

Os coeficientes quadráticos são computados aproximando-se as derivadas através das diferenças entre *pixels* das imagens já filtradas.

A localização *sub-pixel* / sub-escala do ponto de interesse é dada pelo extremo da função apresentada na eq. (105). Esta localização, $\hat{x}$, é determinada ao se fazer a derivada segunda da eq. (105) com relação a *x* e igualando o resultado a zero. Isto é feito como a seguir:

$$\frac{\partial D(\hat{x})}{\partial \bar{x}} = \frac{\partial D^T}{\partial \bar{x}} + \frac{\partial D}{\partial \bar{x}^2}\hat{x} = 0 \tag{108}$$

Perceba que esta derivada usa a expansão de Taylor até $\frac{\partial D}{\partial \bar{x}}$. Tem-se então a posição do extremo dada por:

$$\hat{x} = -\frac{\partial^2 D^{T-1}}{\partial \bar{x}^2} \frac{\partial D}{\partial \bar{x}} \tag{109}$$

O resultado é um sistema linear 3x3 que pode ser resolvido com custo mínimo. Caso $\hat{x}$ seja maior que 0.5 em alguma dimensão, isto significa que o extremo se aproxima mais de outro ponto. Neste caso, o ponto é re-alocado e a interpolação é realizada para este novo ponto. O *offset* $\hat{x}$ final é adicionado à localização do ponto analisado para se chegar à interpolação estimada da localização do extremo.

A localização estimada deverá ser usada a partir de então nos procedimentos que seguirão.

O valor da função no extremo, $D(\hat{x})$, é utilizado para se rejeitar extremos instáveis com baixo contraste. Substituindo-se a eq. (109) na eq. (105) obtemos:

$$D(\hat{x}) = D + \frac{1}{2} \frac{\delta D^T}{\delta \bar{x}} \hat{x} \tag{110}$$

É aconselhável por Lowe que se rejeite valores de $|D(\hat{x})|$ inferiores a um determinado valor. Em [25] é aconselhado trabalhar-se com o valor 0.03 (assumindo-se que os *pixels* da imagem estejam entre [0,1]).

Aqui não é refinada a posição como apresentado, porém são descartados valores de $|D(x,y,\sigma)|$ inferiores a determinado limiar.

Alem do procedimento apresentado para se descartar pontos, Lowe ainda aponta que a função DoG possui resposta forte ao longo de arestas, mesmo que a localização ao longo da borda seja mal determinada. Isto faz com que estes pontos sejam instáveis para ruído em até pequenas quantias.

Um pico mal definido em DoG terá grande curvatura principal ao longo da borda, porém pequena curvatura em sua direção perpendicular. As curvaturas principais podem ser computadas através da matriz Hessiana 2x2, *H*, computada na localização e escala do ponto:

$$H = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{xx} \end{bmatrix} \tag{111}$$

Onde:

$$D_{ab} = \delta_{ab} D = \frac{\partial\left(\frac{\partial D(\hat{x})}{\partial b}\right)}{\partial a} \tag{112}$$

Onde D_{xy} é a derivada de $D(x,y,\sigma)$ na localização e escala dados, em relação a x, e então a y; D_{xx} é a derivada segunda em relação a x; D_{yy} é a derivada segunda em relação a y;

As derivadas são estimadas através das diferenças entre pontos vizinhos à localização e escala definidos, podendo ser aproximada por:

$$D_{xx} = D(x+1,y,\sigma) - 2D(x,y,\sigma) + D(x-1,y,\sigma) \tag{113}$$

$$D_{yy} = D(x,y+1,\sigma) - 2D(x,y,\sigma) + D(x,y-1,\sigma) \tag{114}$$

$$D_{xy} = \begin{pmatrix} D(x-1,y+1,\sigma) - D(x+1,y+1,\sigma) \\ +D(x+1,y-1,\sigma) - D(x-1,y-1,\sigma) \end{pmatrix} \Big/ 4 \tag{115}$$

Os autovalores de H são proporcionais às principais curvaturas de D. Porém, não será necessário computar os autovalores pois o que se busca é a razão entre as curvaturas. Determina-se α, o autovalor com maior magnitude, e β, o de menor. Pode-se, então, calcular a soma dos autovalores pelo traço de H e o produto pelo determinante:

$$Tr(H) = D_{xx} + D_{yy} = \alpha + \beta \tag{116}$$

$$Det(H) = D_{xx} D_{yy} - (D_{xy})^2 = \alpha\beta \tag{117}$$

Para o caso em que o determinante é negativo, as curvaturas possuem sinais diferentes e então o ponto é descartado como não sendo um extremo. Sendo r a razão entre o autovalor de maior magnitude e o de menor, de modo que $\alpha = r\beta$, então:

$$\frac{Tr(H)^2}{Det(H)} = \frac{(\alpha+\beta)^2}{\alpha\beta} = \frac{(r\beta+\beta)^2}{r\beta^2} = \frac{(r+1)^2}{r} \tag{118}$$

A eq. (118) depende apenas da razão entre os autovalores, independente de seus valores individuais. O valor $(r+1)^2/r$ é mínimo quando os dois autovalores são idênticos e cresce com r. Portanto, para se conferir a razão entre as curvaturas está abaixo de determinado limiar $\tilde{r}$, basta checar:

$$\frac{Tr(H)^2}{Det(H)} < \frac{(\tilde{r}+1)^2}{\tilde{r}} \tag{119}$$

A eq. (119) é altamente eficiente de ser computada. Lowe propõe o uso de $r = 10$.

Após a definição de quais pontos serão usados como pontos chaves ainda serão feitas duas etapas na construção dos descritores. A primeira etapa consiste em estabelecer uma ou mais orientações para cada ponto chave. Então, é feita a construção dos descritores com relação às orientações definidas.

A definição das orientações de cada ponto chave é descrita na próxima seção.

3.2.4. Atribuição da Orientação dos Descritores

Ao se atribuir uma orientação para cada ponto chave, pode-se representar os descritores em relação a esta orientação, conseguindo-se assim invariância quanto à rotação. O método utilizado para se atribuir esta orientação é apresentado como se segue.

A escala Gaussiana σ é utilizada para se escolher a imagem filtrada L, com a escala mais próxima, e de oitava referente ao ponto avaliado. Dessa maneira, todas os cálculos passam a ser feitos com invariância à escala.

Para cada ponto de cada imagem $L(x,y,\sigma)$ intervalar, referente às escalas e oitavas utilizadas, são calculados os gradientes. Magnitude $m(x,y)$ e orientação $\theta(x,y)$ são calculados como se segue:

$$m(x,y) = \sqrt{\begin{pmatrix} (L(x+1,y) - L(x-1,y))^2 \\ +(L(x,y+1) - L(x,y-1))^2 \end{pmatrix}} \tag{120}$$

$$\theta(x,y) = \tan^{-1}\left(\frac{(L(x,y+1) - L(x,y-1))}{(L(x+1,y) - L(x-11,y))}\right) \tag{121}$$

Observe que σ não aparece nas equações. Isto foi feito para simplificar, pois o processamento é feito para cada imagem L. Somente as imagens correspondentes a intervalos precisam ser processadas.

Agora, monta-se um histograma das orientações para *pixels* em uma região ao redor do ponto chave. O histograma é uma função discreta h_θ um determinado número de valores discretos de θ (Lowe sugere 36) cobrindo os 360^o de orientações.

Cada ponto na vizinhança do ponto chave é adicionado ao histograma para até dois θ's discretos mais próximos de sua orientação com uma serie de pesos.

O primeiro peso é dado pela distância entre a orientação e θ discreto normalizada pelas distâncias entre θ's discretos. Este peso é dado por:

$$\alpha = \begin{cases} d/i, d < i \\ 0, d > i \end{cases} \tag{122}$$

Onde *d* é a distância absoluta em graus entre a orientação do ponto e θ discreto, e *i* é o intervalo em graus entre θ's discretos.

Por exemplo, para h_θ com θ dado por 0º,10 º,20 º...350 º, ou seja, com intervalos de 10º, um ponto com orientação de 15 º seria acrescentado em h_{10} e h_{20}. Como a distância da orientação para 10 º e 20 º é de 5 º, o peso utilizado para adicionar este ponto em para h_{10} e h_{20} é dado por 5/10, ou seja, a distância sobre o intervalo entre θ's para h_θ.

O segundo peso é dado pela magnitude *m(x,y)* de cada ponto adicionado a h_θ.

O último peso é dado por uma janela gaussiana circular com σ'com valor 1.5 vezes maior que a escala σ do ponto chave. Esta janela é definida pela função gaussiana:

$$g(\Delta x, \Delta y, \sigma') = \frac{1}{2\pi\sigma'^2} e^{-\left(\Delta x^2 + \Delta y^2\right)/2\sigma'^2} \tag{123}$$

E,

$$\sigma' = \sigma \tag{124}$$

Onde Δx e Δy são as distâncias entre cada ponto verificado e o ponto chave.

Por fim, h_θ é atualizado com estes pesos, para cada ponto na vizinhança localizado em *(x,y)*, da seguinte forma:

$$h_\theta' = h_\theta + \alpha \cdot m(x,y) \cdot g(\Delta x, \Delta y, \sigma') \tag{125}$$

Onde h_θ' é a atualização de h_θ.

Perceba que não é necessário fazer a atualização de h_θ para todos os pontos da imagem porque a função $g(\Delta x, \Delta y, \sigma')$ retorna valores muito baixos (aproximadamente zero) para a grande maioria dos pontos.

Picos na orientação do histograma correspondem a direções dominantes para os gradientes locais. O maior pico no histograma e aqueles acima de 80% do

valor do maior pico são usados para se definir a orientação de cada ponto chave. Portanto, para localizações com múltiplos picos de magnitude similar, são criados diferentes pontos chaves na mesma localização, mas com diferentes orientações.

Para se definir com maior precisão à orientação, uma parábola é interpolada entre os 3 valores do histograma próximos de cada pico, e então é interpolada a posição do pico.

Finalmente é possível construir os descritores para os pontos chaves definidos. Isto será abordado na seção a seguir.

3.2.5. Construção do Descritor Local

Até então, para cada oitava, foram escolhidos pontos chaves para localizações, escala σ e orientação definidos. A etapa atual consiste em computar o descritor que represente as regiões relativas aos pontos chaves. Não se esqueça que os procedimentos a seguir serão feitos normalizados em relação à orientação definida na seção anterior para cada ponto chave.

Também serão utilizados os gradientes das imagens *L* já calculados na etapa descrita na seção 3.2.4.

Para cada ponto chave, a construção do descritor é feita através dos seguintes passos:

- Escolhe-se a imagem filtrada L referente à escala σ e oitava relativas ao ponto chave;
- De modo a se conseguir invariância, as coordenadas dos pontos vizinhos ao descritor e das orientações dos gradientes destes pontos são giradas em relação ao ponto chave de acordo com a orientação definida na seção anterior;
- Uma função gaussiana, como apresentada na eq. (123), é utilizada como peso para se ajustar as magnitudes de cada ponto na vizinhança do ponto chave. σ' é escolhido igual a metade da largura da janela em que será calculado o descritor;
- São definidas n x n regiões, com k x k *pixels* cada, ao redor da localização do ponto chave. Geralmente n = k = 4 como exemplificado na Figura 3-6;

- Para cada região, é feito um histograma hθ. para 8 direções, como na Figura 3-7. Este histograma é feito com as magnitudes dos *pixels* pertencentes a cada região. A construção do histograma é similar à apresentada na seção 3.2.4. O peso referente à magnitude de cada *pixel* foi atenuado pela função gaussiana como já ajustado. Perceba que a função gaussiana não é aplicada de modo idêntico ao na seção anterior. Também como feito em 3.2.4, é utilizado um peso α para interpolar a direção relativa no histograma.
- O descritor é então representado pelos histogramas das regiões. A Figura 3-8 exemplifica como fica o descritor para 2 x 2 regiões (n = 2 e k = 4);
- O descritor é representado por um vetor, onde cada valor do vetor é referente a uma das direções de um dos histogramas. Para n e k iguais a 4, o vetor tem tamanho 128.
- Para que o descritor tenha invariância à iluminação, este é normalizado. Após a normalização, todos os valores acima de um determinado limiar são ajustados para este limiar. Isto é feito para que direções com magnitude muito grande não dominem a representação do descritor. Lowe sugere usar limiar 0.2. Por fim, o vetor é normalizado novamente.

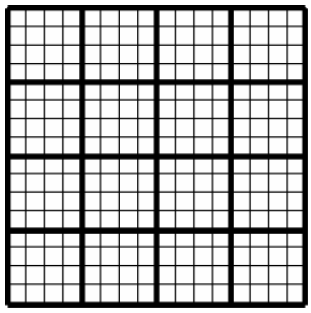

Figura 3-6: Regiões com *n* = 4 e *k* = 4;

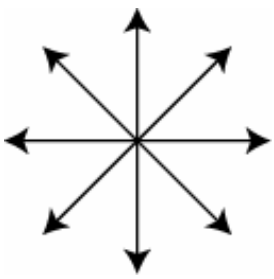

Figura 3-7: Direções do histograma

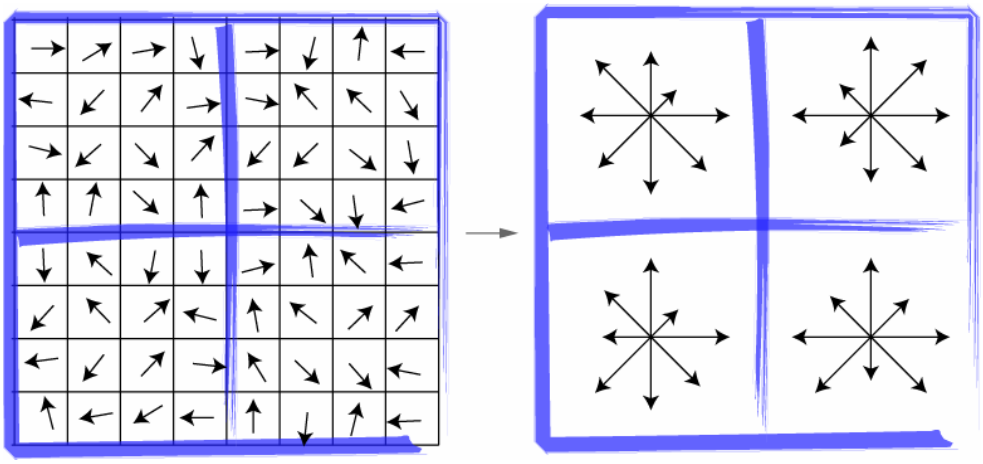

Figura 3-8: Construção do descritor

O descritor está construído. Para cada imagem, são construídos diversos descritores, cada um referente a um ponto chave. Quando se aplica a técnica SIFT em uma imagem, tem-se como resultado, portanto, um conjunto de descritores. Estes descritores podem ser, então, usados para se fazer a correspondência da imagem em outra imagem como será visto na próxima seção.

3.2.6. Encontrando os Pontos em Comum

Para se encontrar a correspondência entre duas imagens, deve-se encontrar pontos em comum entre as duas. Quando se trabalha com a técnica SIFT, pontos de interesse são detectados pelo método e representados em descritores. Tendo-se descritores de duas imagens, a tarefa de se encontrar a correspondência de uma imagem em outra é resumida por se encontrar entre os descritores de uma imagem, os melhores candidatos a serem seus equivalentes na outra imagem. Portanto, dadas duas imagens I_1 e I_2, a tarefa de se encontrar a correspondência de I_1 em I_2 pode ser definida como se segue.

Os descritores são respectivamente definidos por di_1 e dj_2, onde i e j são aos índices para cada um dos descritores de cada imagem e k é o tamanho de cada descritor:

$$di_1 = (m_1 i_1, m_1 i_2, m_1 i_3 \cdots, m_1 i_k) \tag{126}$$

$$dj_2 = (m_2 j_1, m_2 j_2, m_2 j_3 \cdots, m_2 j_k) \tag{127}$$

A magnitude de cada valor dos vetores di_1 e dj_2 é dada por $m_a i_b$, onde a representa a qual imagem se refere o descritor, i é o índice do descritor e b é o índice de cada magnitude dentro do vetor.

A correspondência é feita achando-se os descritores dj_2 que mais se assemelham aos descritores di_1, encontrando-se as falsas equivalências e eliminando-as e por fim, encontrando-se a transformação de I_1 para I_2.

A tarefa de se encontrar o melhor candidato dj_2 para determinado di_1 é feita procurando-se o vizinho mais próximo ou *"nearest neighbor"* de di_1 entre todos os possíveis candidatos, ou seja, para todo o índice j. Quando se procura classificar uma imagem em um extenso banco de dados de descritores para vários objetos, a busca exaustiva de vizinho mais próximo pode ser demorada e para tal existem diversas técnicas para se acelerar a busca. Porém, para o caso de se comparar duas imagens, a busca exaustiva não exige processamento pesado e, portanto, foi a escolhida.

O vizinho mais próximo de di_1 para i dado é definido por dj_2 que possua a menor distância euclideana em relação à di_1. Ou seja, deseja se encontrar j que minimize a função:

$$|di_1 - dj_2| = \sqrt{\begin{pmatrix}(m_1 i_1 - m_2 j_1)^2 + (m_1 i_2 - m_2 j_2)^2 + \ldots \\ +(m_1 i_k - m_2 j_k)^2\end{pmatrix}} \tag{128}$$

Isto é feito para todo i de modo a serem encontrados todos os pares de descritores correspondentes. Perceba que muitos dos pares encontrados correspondem a falsas equivalências, portanto, as correspondências serão refinadas e falsos pares descartados.

O capítulo seguinte aborda métodos de registro e correspondências entre imagens, mostrando como utilizar as correspondências entre pontos encontradas com o método SIFT.

4
Registro e Correspondência de Imagens

Duas áreas de interesse em Visão Computacional são o registro e a correspondência de imagens. Estas duas áreas possuem muito em comum por buscarem regiões em comum entre duas imagens. Neste capítulo serão apresentadas algumas teorias e técnicas relativas a estas áreas que serão usadas nos procedimentos experimentais desenvolvidos.

O registro de imagens consiste da sobreposição de imagens de uma mesma cena obtidas em momentos, pontos de vista e até mesmo por sensores diferentes [36]. A correspondência de imagens consiste em classificar imagens mesmo que estas estejam em escalas, rotações, posições e com geometrias diferentes.

A comparação de imagens utilizando-se transformações invariáveis baseadas em Fourier e Mellin está descrita no capítulo 2. Neste capítulo foram apresentados alguns procedimentos de comparação de imagens que podem ser utilizados para se fazer a classificação de duas imagens. Porém, estes procedimentos comparam a imagem de modo global. Os métodos aqui discutidos tratam da correspondência de imagens através de características locais, ou seja, buscam a correspondência de regiões da imagem. Outra grande diferença entre os métodos aqui apresentados e os do capítulo 2 é que através destes é possível se descobrir um modelo que transforma uma das imagens na outra de modo aproximado.

Durante o registro de duas imagens, uma imagem é transformada para poder ser combinada com uma imagem de referência, ou, imagem base. Esta transformação deve remapear as posições dos *pixels* da imagem transformada de modo que a área sobreposta esteja alinhada com a imagem de referência.

Primeiramente, pontos em comum entre as duas imagens devem ser encontrados, estes são chamados de pontos de controle. A partir dos pontos de controle, estimam-se os parâmetros do modelo de transformação que irá gerar a imagem transformada que será sobreposta à imagem base. Por fim é feita a transformação e a sobreposição das imagens.

Uma etapa dos métodos de registro está em se fazer a correspondência de duas imagens. Isto consiste em encontrar pontos em comum entre as duas e a transformação que leva os pontos de uma imagem para a outra de modo que se correspondam. Pode-se entender então a correspondência de imagens como uma sub-área do campo de registro de imagens.

Um outro campo de Visão Computacional ainda não comentado é o campo de Fluxo Ótico [37]. Esta área trabalha com casos que requerem a determinação de um campo vetorial que descreva deslocamentos ocorridos em quadros subsequentes de uma sequência genérica de vídeo. A determinação do Fluxo Ótico é ainda um problema sem solução (como muitos dos problemas descritos no presente trabalho). Um trabalho nacional recente sobre o assunto pode ser visto em [38].

Este capítulo está dividido nas seguintes seções:

- 4.1 - Etapas de um Método de Registro: Apresentação dos passos comuns para métodos de registro;
- 4.2 – Detecção e Casamento de Pontos de Controle: Descreve como são procurados pontos em comum entre diferentes imagens;
- 4.3 – Transformação de Coordenadas de uma Imagem: Explica diferentes maneiras de se mapear uma imagem para que esta corresponda com outra e como se obtém a função que faz o mapeamento de uma imagem em outra. A seção é sub-dividida em:
    - 4.3.1 – Mínimos Quadrados;
    - 4.3.2 – Transformação Procrustes;
    - 4.3.3 – Transformação Afim;
- 4.4 – Refinando o Modelo de Transformação T: São discutidos diferentes métodos de se melhorar a função que mapeia a imagem. São apresentados:
    - 4.4.1 – Transformada de Hough;
    - 4.4.2 – RANSAC;
    - 4.4.3 – Reajustando a Matriz de Pesos W;
    - 4.4.4 – Determinando-se Dados Inconstantes Através de Limiar;

- 4.5 – Sobreposição de imagens Utilizando Ajuste Radiométrico dos *Pixels*. Esta seção descreve como é aplicado o modelo de transformação de modo a se sobrepor duas imagens;

4.1. Etapas de um Método de Registro

De um modo geral, métodos de registro consistem das seguintes etapas [39]:

- Detecção de pontos ou áreas de interesse: Objetos ou áreas distintos (regiões fechadas, arestas, contornos, intersecção de linhas, cantos) são identificados na imagem a ser transformada. Estes objetos serão representados por suas representações pontuais, denominadas de pontos de controle;
- Casamento dos pontos de controle: Nesta etapa, é encontrada a correspondência dos pontos de controle da imagem a ser transformada na imagem de referência. A idéia do casamento de pontos de controle é ilustrada na Figura 4-1. A detecção de pontos de controle e casamento de pontos entre imagens é apresentada na seção 4.2;
- Estimação do modelo de transformação: São estimados o tipo e os parâmetros da função de transformação que mapeará a imagem para poder fazer o alinhamento com a imagem de referência. Na seção 4.3 são vistos alguns modelos de transformação possíveis tal como são calculados ;
- Refinando o modelo de transformação: O modelo de transformação visto na etapa anterior pode ser melhorado através da eliminação de pontos de controle inconsistentes como também através de novos cálculos. A seção 4.3 apresenta algumas técnicas que podem ser usadas para se refinar o modelo de transformação obtido;
- Transformação e sobreposição das imagens: A imagem é transformada pela função de transformação obtida. Os valores da imagem em coordenadas não discretas é computado por alguma técnica de interpolação. A transformação das imagens seguida da superposição das mesmas é ilustrada na Figura 4-2 e descrita na seção 4.5;

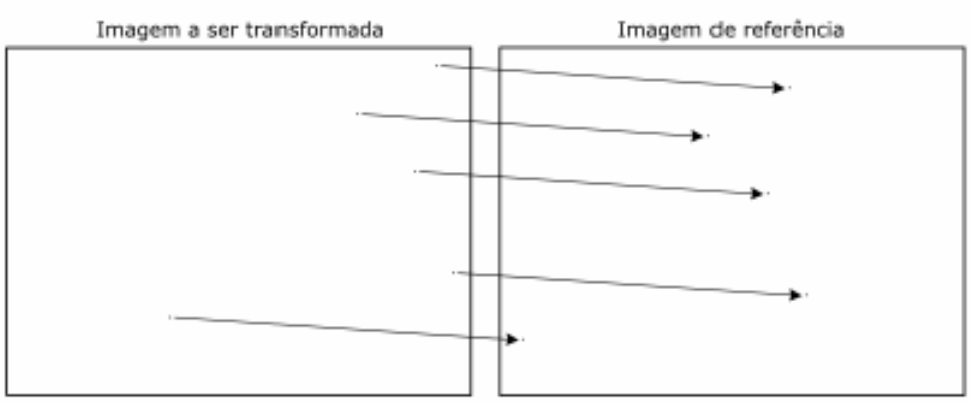

Figura 4-1: Casamento de pontos de controle

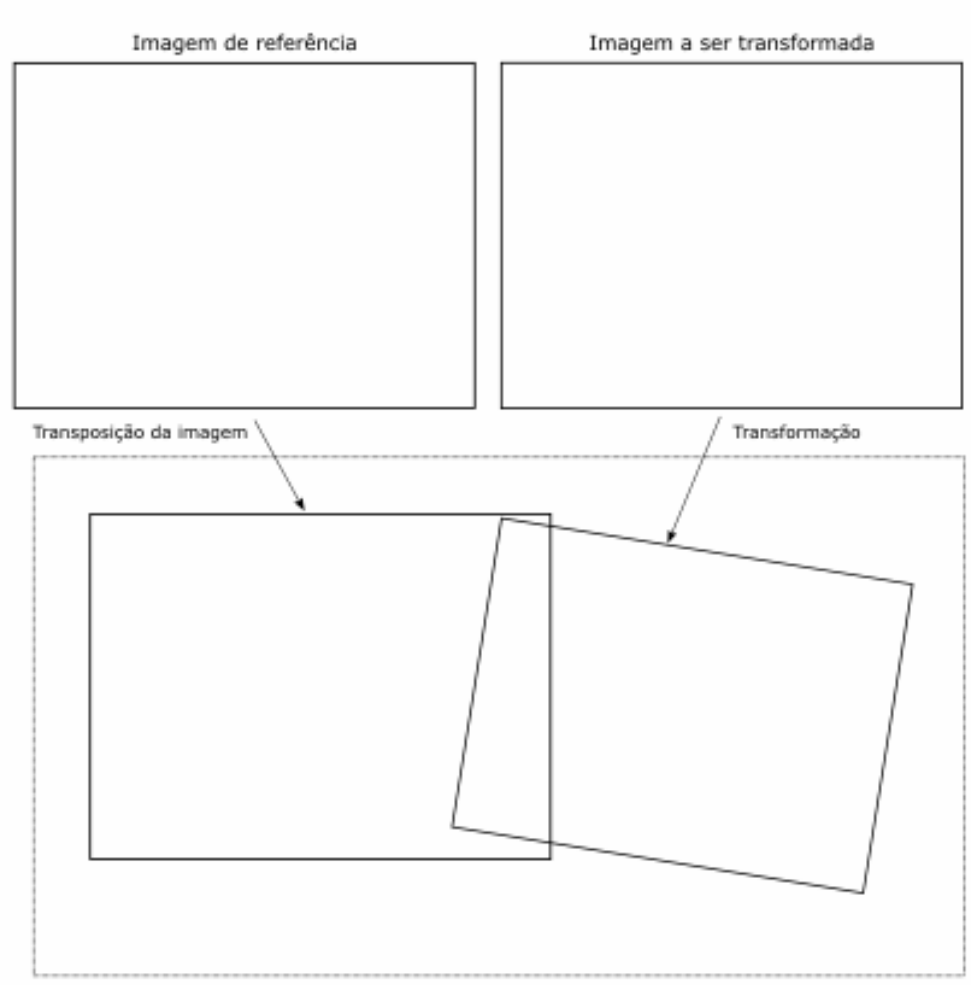

Figura 4-2: Registro de imagens

4.2.
Detecção e Casamento dos Pontos de Controle

Existem diversos métodos para identificar áreas em comum entre as duas imagens. Um método apresentado para se detectar e comparar pontos de controle é apresentado no capítulo 3 através da técnica SIFT. Neste capítulo também são citados alguns outros métodos de obtenção de descritores que podem ser utilizados para se comparar imagens.

Uma das maneiras mais simples em se casar pontos de controle consiste em usar a medida de Correlação Cruzada Normalizada. A Correlação Cruzada Normalizada entre uma imagem I e uma imagem base *w* é dada por:

$$CC(x',y') = \frac{\sum_x \sum_y \left[I(x,y) - \bar{I}(x,y)\right]\left[w(x-x',y-y') - \bar{w}\right]}{\left\{\sum_x \sum_y \left[I(x,y) - \bar{I}(x,y)\right]^2 \sum_x \sum_y \left[w(x-x',y-y') - \bar{w}\right]^2\right\}^{1/2}} \quad (129)$$

Obtêm-se a posição x,y de casamento de I na imagem base encontrando-se x' e y' que maximize CC(x',y').

Note que a correlação apresentada aqui é um pouco diferente da vista na seção 2.6.2. Aquela se mostrou mais adequada para a aplicação em questão, mas, os dois meios são válidos como medida de correlação.

Uma maneira de se encontrar pontos de controle de uma imagem em outra, é se definir uma região vizinha ao ponto de controle e procurá-la na imagem base através da correlação cruzada normalizada. Repare que este método possui a desvantagem de não ser invariável a mudanças de rotação e escala, mas somente à translação. Em contrapartida, a correlação cruzada é computada com grande eficiência computacional.

A correlação cruzada é um dos métodos mais simples e comuns de se comparar imagens.

Uma aplicação da correlação cruzada para detecção e casamento de pontos de controle na geração de imagens panorâmicas será proposta na seção 6.2.2.

Após serem encontrados pontos em comum entre duas imagens, estes pontos são usados para se encontrar uma função que mapeie as posições dos pontos de uma imagem para as posições dos pontos correspondentes na outra imagem. O mapeamento de posições é discutido na próxima seção.

4.3.
Transformação de Coordenadas de uma Imagem

Uma transformação geométrica mapeia as posições dos *pixels* de uma imagem para novas posições. Em essência, uma transformação geométrica faz o rearranjo da configuração espacial de uma imagem. A Figura 4-3 ilustra este conceito.

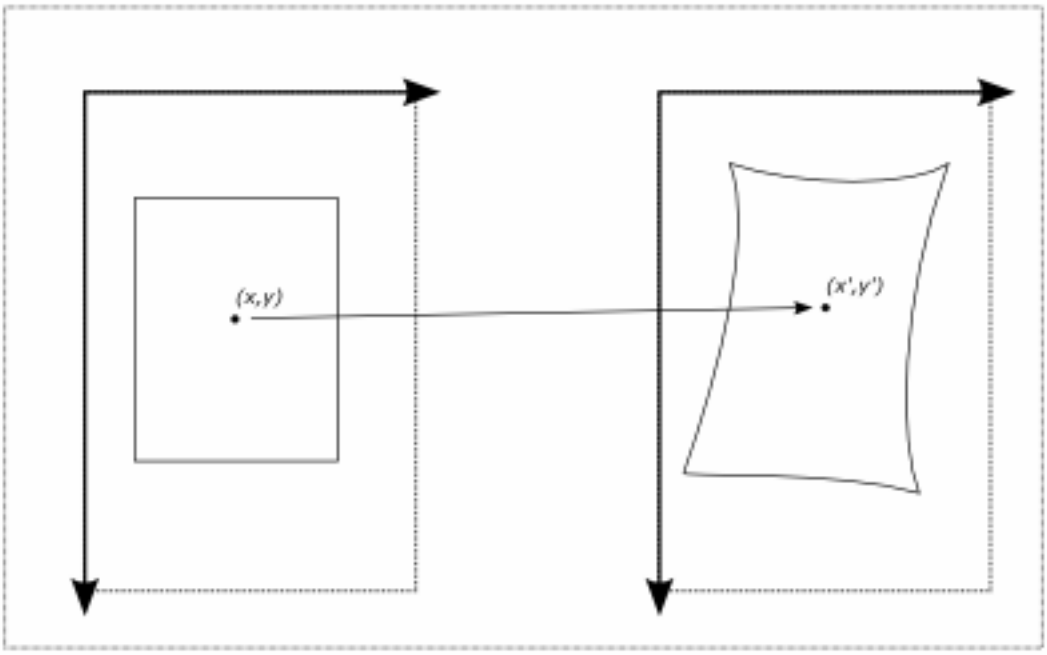

Figura 4-3: Transformação de uma imagem

Para uma transformação de coordenadas que mapeia $p = \begin{pmatrix} x \\ y \end{pmatrix}$ em $p' = \begin{pmatrix} x' \\ y' \end{pmatrix}$ dada por:

$$p' = T(p) \tag{130}$$

Ou,

$$p = T^{-1}(p') \tag{131}$$

A transformação geométrica da imagem $I(p)$ na imagem $I'(p')$ é dada por:

$$\begin{aligned} I'(p') &= I(p) \\ I'(T(p)) &= I(p) \end{aligned} \tag{132}$$

Ou:

$$I'(p') = I\left(T^{-1}(p')\right) \tag{133}$$

Perceba que há duas maneiras de se fazer a transformação. A primeira delas é dada por $I'(T(p)) = I(p)$ e é conhecida como método direto. O procedimento consiste em transferir as cores da imagem original para a imagem transformada. Cada *pixel* da imagem original resultará em um *pixel* na imagem transformada. Como o resultado de $T(p)$ pode não ser inteiro, é preciso fazer um rearranjo da posição dos *pixels* resultantes, como por exemplo trabalhar com o *pixel* vizinho mais próximo. Este procedimento tem a desvantagem de poder resultar em pontos em branco em $I'(p')$ pois nem todos os *pixels* de p' serão preenchidos a partir do resultado de $T(p)$.

Outro procedimento possível é trabalhar com a transformação pelo método indireto dado por $I'(p') = I\left(T^{-1}(p')\right)$. Através deste método cada *pixel* da imagem transformada será preenchido procurando-se suas cores na imagem original. Como o resultado de $T^{-1}(p')$ pode não ser inteiro, o valor de $I'_{x',y'}$ pode ser calculado pela interpolação bilinear dada por:

$$\begin{aligned} I'_{x',y'} = {} & ([x]+1-x)([y]+1-y)I'_{[x],[y]} \\ & +(x-[x])([y]+1-y)I'_{[x+1],[y]} \\ & +([x]+1-x)(y-[y])I'_{[x],[y+1]} \\ & +(x-[x])(y-[y])I'_{[x+1],[y+1]} \end{aligned} \tag{134}$$

Várias transformações de coordenadas são apresentadas na

Tabela 4-1 [40, 36].

As transformações Procrustes e Afim serão apresentadas com maiores detalhes nas seções 4.3.2 e 4.3.3 respectivamente. A seção seguinte mostra como calcular os parâmetros das transformações quando se conhece pontos que obedecem a correspondência dada pela transformação através de mínimos quadrados.

Tabela 4-1: Mapeando as coordenadas *p=(x,y)* de uma imagem em novas coordenadas *p'=(x',y')*

Modelo	Transformação de coordenadas	Parâmetros
Translação	$p' = p + b$	$b \in \Re^2$
Escala e translação	$p' = qp + b$	$q \in \Re, b \in \Re^2$
Rotação e translação	$p' = \begin{bmatrix} \cos\theta & sin\theta \\ -sin\theta & \cos\theta \end{bmatrix} p + b$	$b \in \Re^2$
Transformação Procrustes - Escala, rotação e translação	$p' = \begin{bmatrix} \cos\theta & sin\theta \\ -sin\theta & \cos\theta \end{bmatrix} qp + b$	$q \in \Re, b \in \Re^2$
Transformação Afim	$p' = Ap + b$	$A \in \Re^{2x2}, b \in \Re^2$
Transformação Projetiva	$p' = \frac{Ap + b}{c^T p + 1}$	$A \in \Re^{2x2}, b, c \in \Re^2$
Transformação Bilinear	$\begin{pmatrix} x' \\ y' \end{pmatrix} = A\begin{pmatrix} x \\ y \end{pmatrix} + Bxy + b$	$A \in \Re^{2x2}, B, b \in \Re^2$
Transformação Polinomial	$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{bmatrix} \sum a_{ij} x^i y^j \\ \sum b_{ij} x^i y^j \end{bmatrix}$	$a_{ij} \in \Re, b_{ij} \in \Re$

4.3.1. Mínimos Quadrados

Algumas transformações espaciais em imagens podem ser definidas por [40, 36]:

$$X' = TX \tag{135}$$

Para:

$$X = \begin{pmatrix} x & y & 1 \end{pmatrix}^T \tag{136}$$

$$X' = \begin{pmatrix} x' & y' & 1 \end{pmatrix}^T \tag{137}$$

Para problemas onde se deseja encontrar os parâmetros de uma transformação T, vista na eq. (135), em que se conhecem diversos pontos $p_i = [x_i, y_i]$ e correspondentes $p_i' = [x_i', y_i']$, tem-se:

$$P' = \begin{bmatrix} p'_1 \\ \vdots \\ p'_n \end{bmatrix} \tag{138}$$

Se parâmetros da matriz de transformação *T* são escritos em um vetor Z, e a eq. (135) é reescrita como:

$$P' = JZ \tag{139}$$

Onde *J* é a Matriz Jacobiana utilizada para descrever *P* `em termos dos parâmetros da matriz de transformação.

Procura-se a solução que aponta os melhores parâmetros de *T* que minimiza, para todo *i,* a soma dos erros entre as localizações mapeadas. Isto pode ser entendido da seguinte maneira, após *T* ter sido estimado, define-se para cada par de pontos *i*:

$$X_i' = \begin{pmatrix} x_i' & y_i' & 1 \end{pmatrix}^T \tag{140}$$

$$X_i = \begin{pmatrix} x_i & y_i & 1 \end{pmatrix}^T \tag{141}$$

Onde *X'* é referente à posição do ponto equivalente ao descritor cuja localização é dada por *X*. Com uma matriz *T* dada, a localização de *X* é mapeada para $\hat{X}$ como se segue:

$$\hat{X} = TX \tag{142}$$

Onde:

$$\hat{X}_i = \begin{pmatrix} \hat{x}_i & \hat{y}_i & 1 \end{pmatrix}^T \tag{143}$$

O erro quadrático residual entre a localização mapeada por *T* e a localização do descritor dada por *X'* é calculado como:

$$e_i = \sqrt{(\hat{x} - x')^2 + (\hat{y} - y')^2} \tag{144}$$

Ou:

$$e_i = \sqrt{(X_i' - TX_i)^T (X_i' - TX_i)} \tag{145}$$

Pretende-se estimar os parâmetros de T que minimizem a soma dos erros quadráticos dada por:

$$E = \sum e_i \tag{146}$$

A solução é:

$$Z \equiv J^{+} P' \tag{147}$$

Onde J^{+} é a matriz pseudoinversa de J:

$$J^{+} = \left(J^{T} J\right)^{-1} J^{T} \tag{148}$$

Caso sejam atribuídos pesos as correspondências p_i e p'_i, a eq. (139) pode ser reescrita como:

$$W\left(P'\right) = W\left(JZ\right) \tag{149}$$

Onde W é uma matriz diagonal com os pesos escritos em sua diagonal:

$$W = \begin{bmatrix} W_1 & & & & \\ & \ddots & & & \\ & & W_i & & \\ & & & \ddots & \\ & & & & W_k \end{bmatrix} \tag{150}$$

Observe que $1 < i < k$.

A solução aqui é dada pela eq. (147) reescrita como:

$$Z \equiv \left(\left(WJ\right)^{T} \left(WJ\right)\right)^{-1} \left(WJ\right)^{T} WP' \tag{151}$$

Quando a matriz W é a matriz identidade, a eq. (151) se iguala a eq.(147).

A próxima seção apresentará com maiores detalhes a transformação Procrustes e a aplicação da matriz pseudoinversa para cálculo de seus parâmetros.

4.3.2. Transformação Procrustes

A transformação Procrustes [40, 36] é uma transformação de 4 parâmetros que leva em consideração mudanças de escala, rotação e translação entre imagens como ilustrado na Figura 4-4.

A transformação Procrustes pode ser escrita como:

$$\begin{bmatrix} p' \\ 1 \end{bmatrix} = T \begin{bmatrix} p \\ 1 \end{bmatrix}_3 \Rightarrow \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = T \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{152}$$

Onde:

$$T = \begin{bmatrix} a\cos(\theta) & a\sin(\theta) & \Delta x \\ -a\sin(\theta) & a\cos(\theta) & \Delta y \\ 0 & 0 & 1 \end{bmatrix} \tag{153}$$

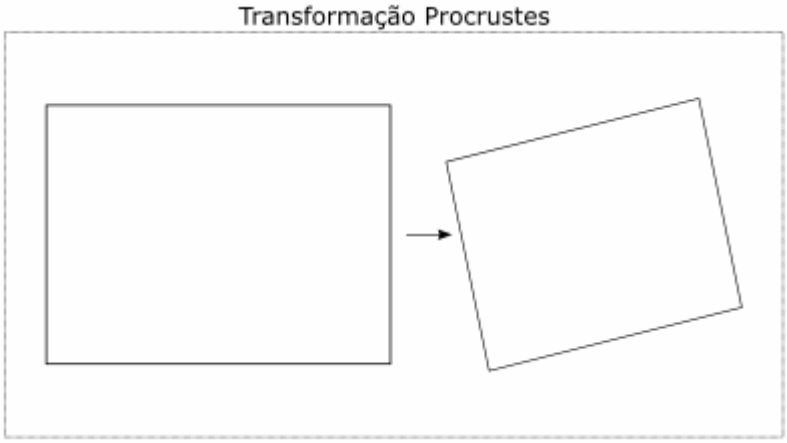

Figura 4-4: Tranformação Procrustes

O parâmetro *a* realiza mudanças em escala. Quando $a = 1$, não há variação quanto à escala. Para $a > 1$, a imagem é dilatada proporcionalmente. E para $a < 1$, a imagem é reduzida proporcionalmente.

O parâmetro Θ cuida da rotação da imagem e está em radianos.

Os parâmetro Δx e Δy realizam as translações horizontal e vertical respectivamente.

Para a transformação Procrustes podemos escrever Z como:

$$Z = \begin{bmatrix} a\sin\theta \\ a\cos\theta \\ \Delta x \\ \Delta y \end{bmatrix} \tag{154}$$

E a matriz Jacobiana J como:

$$J = \begin{bmatrix} y_1 & x_1 & 1 & 0 \\ -x_1 & y_1 & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots \\ y_n & x_n & 1 & 0 \\ -x_n & y_n & 0 & 1 \end{bmatrix} \tag{155}$$

Outra transformação de interesse é a transformação Afim que será vista na seção a seguir.

4.3.3. Transformação Afim

A transformação Afim [40, 36] é uma generalização de 6 parâmetros da transformação Procrustes que leva em consideração escala, rotação, translação e não ortogonalidade entre eixos, permitindo dilatações e reduções ao longo das linhas e colunas de uma imagem como visto na Figura 4-5.

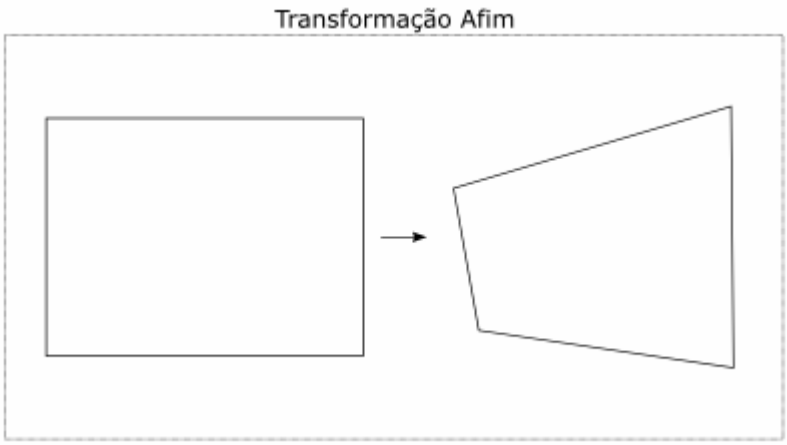

Figura 4-5: Transformação Afim

Esta é a transformação linear mais geral de todas e é definida por:

$$T = \begin{bmatrix} a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \\ 0 & 0 & 1 \end{bmatrix} \tag{156}$$

Para a transformação Procrustes podemos escrever Z como:

$$Z = \begin{bmatrix} a_{10} \\ a_{11} \\ a_{12} \\ a_{20} \\ a_{21} \\ a_{22} \end{bmatrix} \tag{157}$$

E a matriz Jacobiana J como:

$$J = \begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_n & y_n & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_n & y_n & 1 \end{bmatrix} \tag{158}$$

Os modelos de transformação *T* apresentados minimizam os erros quadráticos para os pontos utilizados. Porém, pontos com falsas correspondências induzem ao cálculo de matrizes *T* que não representam realmente a transformação entre duas imagens. De modo a se refinar o modelo calculado, a seção seguinte apresenta algumas técnicas possíveis.

4.4. Refinando o Modelo de Transformação *T*

Para procedimentos onde são encontrados pontos em comum entre imagens e se deseja computar a matriz de transformação que mapeia as posições dos pontos de uma imagem para outra imagem, é interessante refinar o modelo *T* encontrado com auxílio da matriz de pesos *W* apresentada na eq. (150) ou através de eliminação de pontos mal correspondidos. Para tal, são apresentados aqui alguns procedimentos.

As técnicas aqui apresentadas serão utilizadas para se gerar imagens panorâmicas e também junto aos procedimentos de navegação que trabalham com o método SIFT.

No caso de uso do método SIFT, as técnicas a seguir são utilizadas após o processo de encontrar pontos em comum utilizando a técnica SIFT apresentada na seção 3.2.6.

Serão apresentados métodos relacionados à:

- Transformada de Hough;
- RANSAC *(Random Sample Consensus Algorithm*);
- Reajuste da matriz de pesos W;
- Eliminação de pontos através de limiar de erro;

4.4.1. Transformada de Hough

A Transformada de Hough [41] é uma técnica de extração de características chaves em uma imagem. A técnica clássica é utilizada para a identificação de linhas e curvas em imagens, porém pode ser estendida para a identificação de posições de formas diversas ou, como no caso aqui descrito, transformações diversas.

A Transformada de Hough pode ser entendida de um modo genérico como uma tabela de parâmetros que compusesse um modelo. A tabela seria preenchida para cada dado de um conjunto de dados apresentado, encontrando todos os modelos possíveis que condissessem com cada ponto (ou sub-conjunto de pontos) e atualizando a tabela incrementando ás células referentes aos possíveis parâmetros dos modelos encontrados.

A Transformada de Hough utilizada aqui [25] pode ser entendida como um histograma em 4 dimensões que comporte os parâmetros Δx, Δy, a e θ de uma transformação Afim. Poderia ser pensada também como uma tabela com todas as possíveis combinações dos parâmetros dados, sendo estes discretizados.

Perceba que para cada par de descritores equivalentes encontrado é possível se obter uma transformação Proscrutes ou uma transformação Afim. São obtidos então os parâmetros Δx, Δy, a e θ para cada par de descritores encontrado.

A tabela utilizada para se fazer a Transformada de Hough pode ser então criada da seguinte maneira:

- Cria-se uma matriz de 4 dimensões onde cada dimensão representa um parâmetro dentre Δx, Δy, a e θ.
- Para θ, faz-se a discretização de 30^o em 30^o. Ou seja, cada intervalo na dimensão θ é relativo a 30^o. A dimensão definida por θ é dividida em 12 intervalos;
- As escalas a possíveis são dadas por α sendo α inteiro e -8 <= α <= 8. Veja que são 17 intervalos relativos a a;
- Δx, Δy tem intervalos de tamanho 0.25 vezes o tamanho da maior imagem;
- A transformada será obtida através dos seguintes passos:
- Para cada ponto em uma imagem, encontre o ponto correspondente para a imagem 2. Encontre então a Transformada Procrustes para o par de coordenadas. Desta maneira são obtidos os parâmetros Δx, Δy, a e θ da transformação deste par de coordenadas;
- Incremente a célula da tabela correspondente aos parâmetros utilizados. A célula é incrementada com a multiplicação de uma série de pesos. Cada peso é calculado pela distância entre cada parâmetro calculado e o valor discreto relativo à célula. Este peso é

normalizado de modo similar à eq. (122), com a diferença de que há um peso para cada parâmetro do modelo;

- Também são incrementadas as células vizinhas com relação a cada parâmetro. Elas são incrementadas da mesma forma que no passo anterior. Para cada parâmetro são adicionados os 4 vizinhos discretos mais próximos do valor obtido. Ao todo são atualizadas 4 X 4 X 4 X 4 células, ou seja, 16 células para cada par de pontos.

Ao terminar de preencher a tabela, a célula, ou as células com maior peso, são utilizadas para se prosseguir.

Uma alternativa possível é de se trabalhar com um limiar, e aceitar todas as células com valor acima do limiar. Todas as outras são descartadas do processo.

A partir de então, cria-se a matriz de pesos *W*, onde cada peso, referente a cada par de pontos, é relativo aos pesos encontrados para as células que corresponderam àquele par de pontos. Uma maneira de fazer isso é repetir o mesmo par de pontos, para cada célula referente a este que ficou acima do limiar, e para cada aparição do par de pontos, se usar o peso relativo ao valor da célula.

Esta alternativa utilizada aqui pode ser usada em conjunto com RANSAC, que será apresentado na seção seguinte, e depois pode-se recalcular *W* recursivamente até se obter *T* aceitável, como será visto na seção 4.4.3.

Outra alternativa é escolher os pontos relativos à célula, ou as células com maior peso e calcular a transformada afim a partir destes pontos sem usar uma matriz de peso *W*. Então, pode-se eliminar pontos com erro residual muito grande e repetir o processo por um número de vezes, até se encontrar erros abaixo de um limiar escolhido.

4.4.2. RANSAC (*Random Sample Consensus Algorithm*)

O algoritmo RANSAC é apresentado em [42, 43] como um método para se estimar os parâmetros de um modelo para um conjunto de dados conhecido, porém com presença de diversos dados errôneos.

O algoritmo não é aplicado independentemente porque sua eficiência é limitada quando há um grande número de dados, não levando a uma boa estimativa do modelo. Por isso, é sugerido que se aplique anteriormente Hough,

ou outra técnica, conseguindo-se gerar um conjunto de dados mais robusto que o inicial.

RANSAC é de um algoritmo bem simples definido como se segue.

Dado um modelo com parâmetros $\vec{x}$, deseja-se estimá-los. Para tal, é assumido:

- Os parâmetros podem ser estimados a partir de um número N de itens em um conjunto de dados conhecido;
- Existe um total de M itens no conjunto de dados;
- A probabilidade de um dado selecionado aleatoriamente fazer parte de um bom modelo é dada por p_g;
- A probabilidade de que o algoritmo termine sem que se encontre um bom modelo é dada por p_{falha};

O algoritmo é então executado através das seguintes etapas:

1. N itens são escolhidos de modo aleatório;
2. A partir dos itens escolhidos, $\vec{x}$ é estimado;
3. Encontra-se o número de itens que se enquadram ao modelo para determinada tolerância especificada. Este número é chamado de K;
4. Caso K seja grande o suficiente, para um limiar escolhido, o algoritmo termina e foi bem sucedido;
5. O algoritmo é repetido de 1 a 4 um número L de vezes;
6. Caso o algoritmo não tenha terminado após L tentativas, o algoritmo falhou;

L pode ser encontrado através das seguintes maneiras:

- *pfalha* = Probabilidade de L falhas consecutivas;
- *pfalha* = (Probabilidade de que determinado dado seja falho)*L;
- *pfalha* = (1 – Probabilidade de sucesso)*L;
- *pfalha* = (1 – (Probabilidade de que um item caiba no modelo)*N)*L;
- *pfalha* = $(1 - (pg) * N) * L$;

O algoritmo RANSAC é adaptado para se encontrar o modelo da transformação afim de T da seguinte forma:

1. São escolhidos 3 descritores di_1 e dj_2 correspondentes;
2. A partir dos descritores escolhidos, calcula-se a transformação afim T a partir da eq. (151) usando-se pesos calculados para formar W, ou

a matriz identidade caso RANSAC esteja sendo aplicado independentemente;

3. Calcula-se a mediana dos erros residuais em se usar a transformação *T* dados pela eq. (144);
4. Caso a mediana calculada tenha sido a menor até então, *T* é guardado ou então substitui a última transformação *T* guardada;
5. O algoritmo é repetido de 1 a 4 um número *L* de vezes;
6. O *T* que apresentou a menor mediana de erros residuais é utilizado;

L é calculado fazendo-se:

$$P_{sucesso} = \left(1 - \left(P_{item\_ruim}\right)^{N}\right)L \tag{159}$$

Onde:

- *Psucesso*: Probabilidade de que RANSAC seja bem sucedido. Deseja-se que seja 1;
- N: Número de itens usados para se calcular T a cada passagem de RANSAC. Como visto, N aqui é usado igual a 3;

Portanto:

$$L = \frac{P_{sucesso}}{\left(1 - P_{item\_ruim}{}^{3}\right)} \tag{160}$$

Obviamente este valor é aproximado. Aqui ele é arredondado para o primeiro inteiro superior.

Normalmente, após se aplicar RANSAC, é feito um procedimento de reajuste da matriz *T* de modo a se recalcular *T*. Este procedimento é apresentado a seguir.

4.4.3. Reajustando a Matriz de Pesos W

Este procedimento apresentado consiste em reajustar iterativamente uma matriz de pesos W' de modo a se melhorar a estimativa de *T* a partir da distribuição gaussiana dos resíduos encontrados na computação do modelo *T* por mínimos quadrados.

O procedimento aqui apresentado não é altamente eficiente para um número grande de dados inconsistentes, portanto, é interessante a aplicação de Hough e/ou RANSAC anteriormente.

A partir do método proposto em [44] é definido um algoritmo para se encontrar W' que ajude a interpolar um modelo *T* robusto de modo a se conseguir eliminar dados que levam a uma falsa estimativa do modelo.

De modo a se descrever o qual admissível é um par de pontos *i*, é estabelecida uma medida de verossimilhança que será usada para se pesar este par de pontos na obtenção de *T* por mínimos quadrados. Quando se calcula o erro residual dado pela eq. (144), espera-se que aqueles dados que possuem maior erro tenham maior chance de serem inconsistentes. Portanto, é definido uma medida de verossimilhança dada pela distribuição gaussiana do erro residual e_i:

$$p(e_i) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{{e_i}^2}{2\sigma^2}} \tag{161}$$

Onde $p(e_i)$ pode ser descrito como a probabilidade do dado *i* ser consistente.

A estimativa de escala σ é baseada na mediana dos resíduos dados pela eq. (144) como proposto em [44] :

$$\sigma = 1.482 \cdot median(e_i) \tag{162}$$

Pode-se montar então uma matriz W' dada por:

$$W' = \hat{W}W \tag{163}$$

Onde a matriz de peso *W* foi computada por procedimentos anteriores, ou então é igual à matriz identidade, e $\hat{W}$ é uma matriz de pesos onde:

$$\hat{W}_i = p(e_i) \tag{164}$$

Para dados em que $e_i/\sigma > 5$, faz-se $\hat{W}_i = 0$.

Para se definir *T* robustamente, o procedimento a seguir é realizado. $\hat{W}$ é inicialmente usado como matriz identidade.

1. Computa-se T através de mínimos quadrados com matriz de pesos *W'*;
2. Calculam-se os erros residuais ei;
3. Calcula-se σ, p(ei) e consecutivamente $\hat{W}$;
4. Para dados em que $e_i/\sigma > 5$, faz-se $\hat{W}_i = 0$;

5. Repete-se de 1 a 4 um número n de vezes;

É indicado $n = 10$.

O último procedimento a ser apresentado é o mais simples de todos e pode ser aplicado em conjunto com os métodos já apresentados, trata-se da eliminação de dados inconsistentes através de limiar relativo ao erro residual. Este procedimento é visto a seguir.

4.4.4. Determinando-se Dados inconsistentes Através de Limiar

O último método aqui apresentado de se descartar pontos inconsistentes para o cálculo da matriz *T* é através de um limiar definido para o erro residual.

A maneira mais simples de determinar que dados são inconstantes é definir um limiar de erro residual para o qual estes sejam descartados. Após ter se encontrado a matriz *T*, utilizando ou não algum dos métodos apresentados nas seções 4.4.1, 4.4.2 e 4.4.3, ou outro, calcula-se o erro residual para cada ponto utilizado. Todo ponto que apresentar erro superior ao limiar especificado é eliminado.

Após o descarte, *T* deve ser recalculada. Este procedimento costuma ser o último aplicado para se melhorar *T* calculado.

Por fim, a próxima seção descreve como utilizar do modelo *T* encontrado para se combinar imagens. Este processo é particularmente de interesse na geração automática de imagens panorâmicas que será vista no capítulo 6 , na seção 6.2.

4.5. Sobreposição das Imagens Utilizando Ajuste Radiométrico dos Pixels Superpostos

Ao concatenar duas imagens é preciso decidir como trabalhar a área em que as imagens se superpõem. A

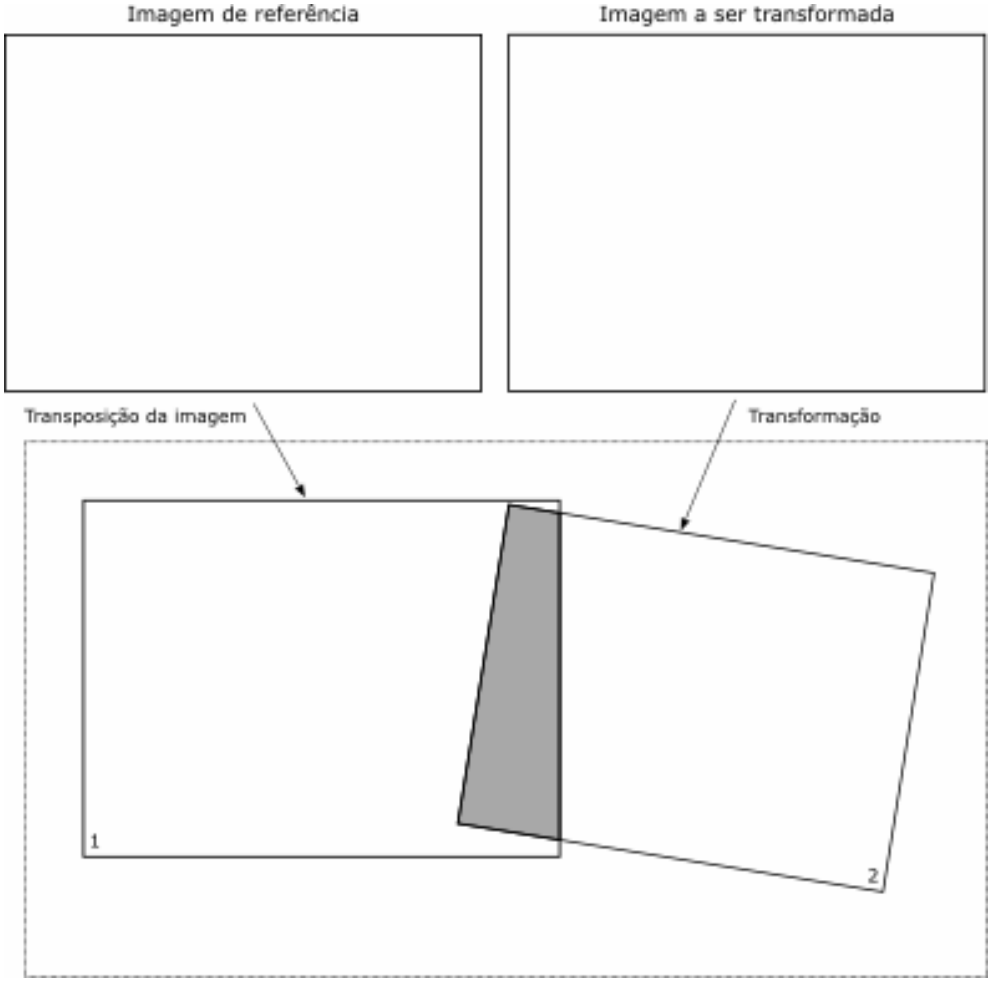

Figura 4-6 ilustra a sobreposição de imagens. Escolher trabalhar com uma imagem sobre a outra leva a possíveis descontinuidades na região de limite entre as imagens. Portanto, uma alternativa é fazer com que as cores de cada *pixel* sejam uma média ponderada entre as duas imagens.

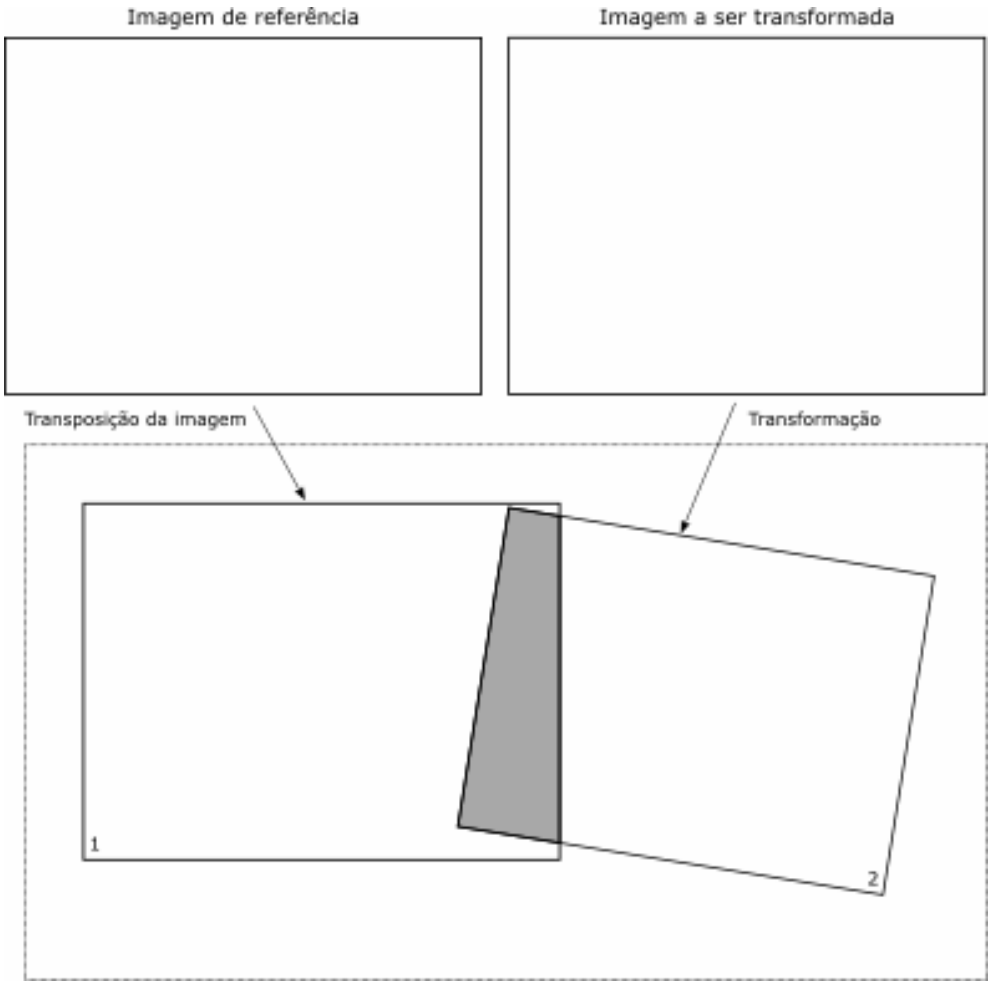

Figura 4-6: Superposição das imagens

Uma solução geral seria ponderar as imagens quanto à distância de cada *pixel* para as bordas de cada imagem. Esta solução ainda leva a algumas descontinuidades visíveis.

Outra possibilidade pode ser utilizada com a transformação Procrustes, criando-se um *dégradée* a partir do começo de uma das imagens até o fim da superposição com inclinação dada por θ. Este ajuste pode ser visto na Figura 4-7 e foi o mais utilizado durante o projeto.

O próximo capítulo apresentará o sistema experimental utilizado e o capítulo 6 passará a utilizar as técnicas discutidas até então para construir o modelo de exploração proposto.

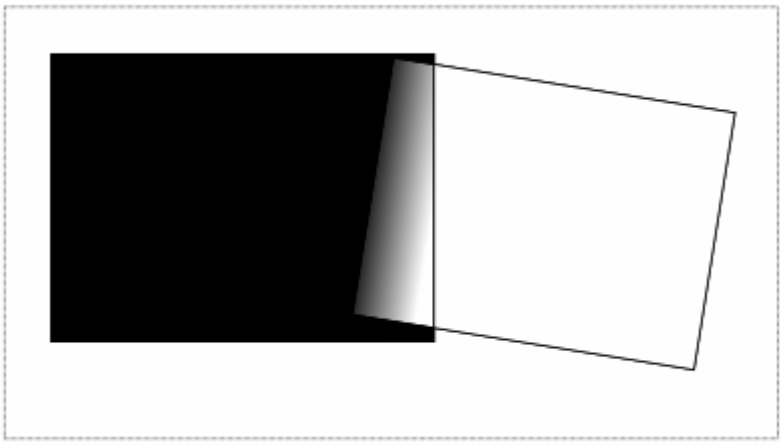

Figura 4-7: Exemplo de ajuste radiométrico

5
Sistema Experimental

Este capitulo apresenta o sistema experimental utilizado e é composto das seguintes seções:

- 5.1 – Robô ER1: Descreve o robô utilizado. É dividida nas seguintes sub-seções:
    - 5.1.1 – Hardware;
    - 5.1.2 – Software;
    - 5.1.3 – Interface com o robô;

5.1.
Robô ER1

O robô utilizado neste projeto foi construído com o kit comercial *"ER1 Personal Robot System"* da empresa *Evolution Robotics*™. O robô pode ser visto na Figura 5-1 em diferentes vistas.

Figura 5-1: ER1 – Personal Robot System

O ER1 difere da maioria das plataformas robóticas por não possuir processamento embarcado. Seu conceito é de ser uma plataforma móvel na qual pode-se colocar um *notebook* que através de programa proprietário pode controlar o robô. Tanto seus motores, quanto câmera e sensores são conectados ao computador através de interface USB. Como o processamento do robô é feito através de um *notebook*, consegue-se poder muito superior a sistemas micro-controlados, sendo o processamento limitado ao computador utilizado.

Também faz parte do conceito deste kit ser modular, pois através de acessórios é possível construir diferentes configurações do robô, além de poderem ser adicionados alguns periféricos, tais como: garra; sensores infra-vermelhos; câmeras; e outros através de entradas analógicas e digitais.

O ER1 tem sido usado em alguns centros de pesquisa e também em universidades como ferramenta de ensino. [45] apresenta este kit como uma plataforma comercialmente acessível que por ser composta de uma série de periféricos, ressalta a natureza modular do robô, e reforça a idéia de que um robô é um conjunto composto por computador, atuadores e sensores, sendo recomendado como ferramenta para experimentação e aprendizado. Além disso, também apresenta o uso do ER1 em sala de aula sendo usado para implementação de algoritmos de localização. Outros também apresentam este robô sendo utilizado em pesquisas como pode ser verificado em [45-47].

A próxima seção detalha o hardware do robô ER1.

5.1.1. Hardware

O kit do ER1 é composto de:

- Chassis: Vigas de perfil x de alumínio, conectores de plástico, porcas e placas laterais em alumínio;
- Sistema de acionamento (*Drive System*): Mecanismo pré-montado sobre placas de alumínio com dois motores de passo, rodas de 4 polegadas de diâmetro, correias e polias. O kit também contém um rodízio de 360^{o};
- Módulo de força: Possui bateria recarregável de 12V e 5,4A-h com fusível interno para proteção. Tem durabilidade de 2,5

horas ao ser utilizada com o robô. Tem duas saídas de 12V DC não reguladas e uma entrada para o carregador de bateria;

- Módulo de controle: Eletrônica que faz a interface dos motores e I/O com o computador. Contém: Conector USB *Slave*; Entrada DC para conexão com módulo de força; Sub-conector de 25 pinos D para I/O digital com 8 entradas e 8 saídas; Sub-conector de 25 pinos D para 15 entradas analógicas;
- Carregador de bateria, conector USB e conector de força;
- *Web* Câmera com resolução de 80x60 até 640x800;
- Programa *"ER1 Robot Control Center"*;

As peças das quais o kit é composto podem ser vistas em Figura 5-2, Figura 5-3, Figura 5-4 e Figura 5-5.

Figura 5-2: Vigas em x, placas laterais e saco com conectores e porcas

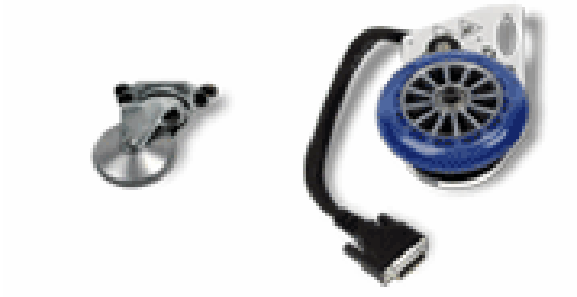

Figura 5-3: Rodízio e conjunto de motor com roda

Figura 5-4: Carregador de bateria, cabo de força e conector USB

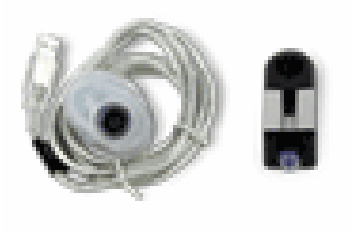

Figura 5-5: *Web* Câmera utilizada

Além do robô, também foi utilizado o acessório *"ER1 IR Sensor Pack"* que traz três sensores infra-vermelhos com conexão USB.

A próxima seção detalha o programa que é utilizado para controlar o robô.

5.1.2. Software

O programa que acompanha o ER1 é chamado de *"ER1 Robot Control Center"*, ou simplesmente RCC, e pode ser visto na Figura 5-6. É através deste programa que é feita a comunicação entre o computador e o robô.

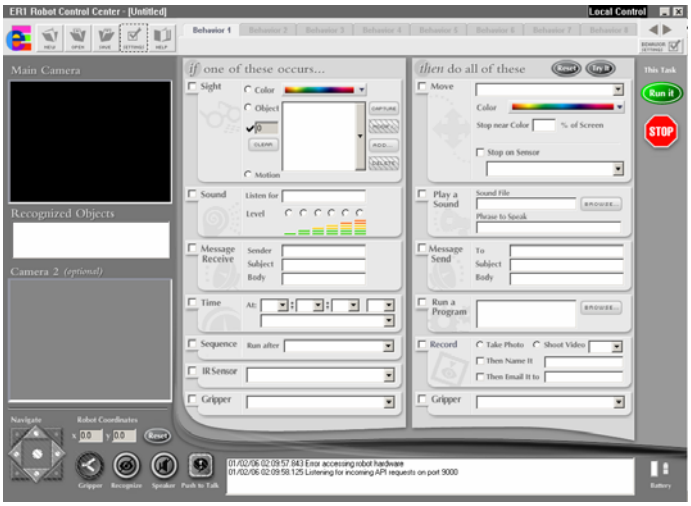

Figura 5-6: *ER1 Robot Control Center*

O programa RCC é usado para controlar e receber *feedback* do robô. Sua interface permite configurar comportamentos através de conjuntos de ordens do tipo "SE condição ENTÃO ação". Esta interface é limitada, não sendo possível utilizar o programa RCC de modo isolado para criar os algoritmos desejados e

trabalhar com o ER1. Porém, o programa pode trabalhar com três configurações distintas:

- ER1 controlado remotamente utilizando-se outro RCC em uma máquina remota conectada ao computador que controla o robô por rede;
- ER1 controlado através de comportamentos configurados no próprio programa;
- ER1 controlado remotamente através do protocolo TCP/IP;

A terceira configuração apresentada é uma solução para as limitações do programa. Podendo-se comunicar com o RRC através de TCP/IP torna-se possível controlar o robô através de um programa externo.

A seção seguinte explicará como foi feita a interface entre as aplicações desenvolvidas no presente projeto e o robô.

5.1.3. Interface com o Robô

Como visto na seção 5.1.2, é possível se comunicar com o RCC através de TCP/IP. Essa comunicação é feita através de uma API ("*Application Programming Interface"*) por linha de comando que permite a criação de programas que enviem comandos diretamente para o robô. Também se pode usar esta API para coletar informação sensorial do robô.

Os comandos e o feedback do ER1 são enviados através de *sockets* TCP. Desta maneira, é possível comandar o ER1 através de qualquer linguagem de programação ou plataforma que suporte TCP/IP tais como *Visual Basic*, C , C++, Java, Python e Matlab.

Esta API também suporta acesso às 15 portas analógicas e 16 portas digitais do ER1, possibilitando a adição de hardware adaptado para ser controlado pelo ER1.

Há, porém o inconveniente de ser obrigatório que o programa RCC esteja rodando para que a API funcione, pois na verdade a comunicação TCP/IP é feita entre o programa externo e o programa RCC e não diretamente com o robô. A

interface entre o robô e o computador é feita pelo RCC. Toda esta comunicação está exemplificada no diagrama apresentado na Figura 5-7.

Figura 5-7: Interface com ER1

Para poder controlar o ER1 via TCP/IP é necessário configurar o programa RCC. Isto é feito através da aba *"Remote Control"* dentro da caixa de diálogo *"Settings"*. A opção de controle remoto é ativada através da opção *"Allow API Control of this instance"*. Para melhor entendimento do procedimento veja a Figura 5-8.

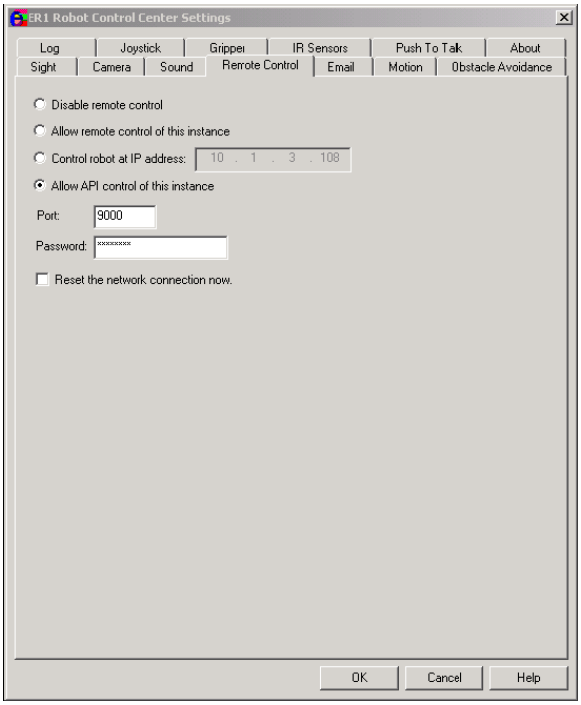

Figura 5-8: *ER1 Robot Control Center Settings*

Através da caixa *"Settings"* e aba *"Remote Control"* também é possível configurar a porta pela qual é estabelecida a comunicação com o programa. Não é possível manter controle através da API e da GUI do RCC ao mesmo tempo.

Também é necessário desligar o uso da câmera pelo RCC para permitir que esta seja acessada exteriormente. Isto é feito na caixa *"Settings"* e aba *"Camera"* desmarcando-se a opção *"enable camera usage"*. O acesso à imagem da câmera é feito através dos *drivers* do *Windows*.

A API apresenta, dentre outros, os seguintes comandos que interessam para o projeto aqui descrito e que podem ser enviados em modo texto através de TCP/IP para a porta configurada:

- "*move*": Faz com que o robô se movimente. Só é possível fazer com que o robô ande em linha reta ou rode sobre seu eixo central. A API responde a este comando após este ter sido verificado, mas antes de que um movimento tenha sido concluído. Isto quer dizer que caso outro comando "*move*" seja dado, o comando anterior será interrompido e o novo executado. Há porém como se determinar se o robô terminou um movimento através do comando "*events*";
- "*stop*": Manda o robô parar;
- "*position*": Retorna à posição *(x,y)* atual do robô relativa à posição em que o robô estava quando o programa RCC iniciou. Esta função também deveria retornar o ângulo que indica a direção do robô, porém este recurso não funciona devido a um problema no programa;
- "*sense*" e "*ir*": Estas funções permitem configurar os sensores IR e ativar o feedback dos mesmos. Porém, este feedback é feito através de outra função, a função "*events*";
- "*events*": Esta função faz com que qualquer feedback do robô seja enviado como uma mensagem de retorno na comunicação TCP/IP. O tratamento de sensores e do estado de movimentação do robô é feito com este comando. Para se tratar um feedback é necessário portanto chamar o comando "*events*" e ficar "escutando" a comunicação TCP/IP até se obter alguma resposta;
- "*exit*": Faz com que o programa RCC seja terminado. O *socket* pelo qual a conexão a API é feita também é fechado;

Como foi apresentado, todo feedback do robô implica no processo de se executar o comando *"events"* e ficar esperando alguma resposta, para então, fazer o tratamento desta resposta. Infelizmente, este processo não é simples e muitas

vezes é extremamente lento. Para se saber o estado dos sensores IR por exemplo, é necessário ficar chamando e esperando alguma resposta dos sensores, sendo que esta resposta pode não acontecer.

O controle do robô através da API dada é limitado. Algumas de suas limitações e problemas são:

- Movimento restrito a rotações e trajetórias em linha reta. Não é possível acessar os motores do robô diretamente para se controlar as velocidades individuais de cada roda. Portanto, é impossível descrever trajetórias curvilíneas.
- Não há acesso direto às contagens referentes aos *encoders* de cada roda. Só é possível saber a posição do robô através das coordenadas *(x,y)*. Deveria ser possível aquisitar também a coordenada θ relativa à direção do robô, porém descobriu-se que a API apresenta um problema que não foi nem será resolvido que impossibilita se adquirir esta coordenada (informação dada pelo suporte da empresa *Evolution Robotics*™).
- A resposta ao comando *"move"* apresenta discrepâncias em relação às distâncias percorridas ou giradas em relação às distâncias pedidas. Isto significa que é necessário estar checando a posição do robô para se corrigir possíveis erros. Porém, isto não é possível de ser feito quando o robô gira porque não é possível se obter a coordenada de giro do robô como já discutido. Estes problemas não deveriam ser esperados já que o robô trabalha com motores de passo e *encoders*, mas infelizmente acontecem.
- O acesso ao feedback do robô é lento e complicado como já descrito anteriormente;
- A resposta aos comandos não é imediata fazendo com que a comunicação com o robô possa ser lenta em vários momentos;

O capítulo seguinte passará a utilizar as teorias apresentadas nos capítulo 2, 3 e 4 para construir o algoritmo de navegação e exploração proposto, tal como suas componentes.

6
Algoritmos de Exploração e Navegação Propostos

O presente capítulo apresenta o algoritmo de exploração proposto e detalha todas suas componentes. As seções deste capítulo são:

- 6.1 - Algoritmo de Exploração;
- 6.2 - Componente 1 - Criando Imagens Panorâmicas Automaticamente;
- 6.3 - Componente 2 - Segmentação da Imagem por *Quadtree* Baseada em Entropia;
- 6.4 - *Visual Tracking*: Acompanhamento de *Features* e Navegação;
- 6.5 - Componente 3 - Navegação para Nó Desconhecido;
- 6.6 - Componente 4 - Navegação para Nó Conhecido;
- 6.7 - Componente 5 - Refinando o modelo criado através de algoritmos genéticos;

6.1.
Algoritmo de Exploração

Nesta seção é apresentado um algoritmo para a modelagem e exploração de um ambiente *indoor* estático. Este algoritmo trabalha com diferentes componentes que serão abordadas em detalhes nas seções que se seguem neste capítulo. O algoritmo de modelagem proposto se diferencia da grande maioria dos algoritmos encontrados por não fazer a modelagem através de um mapeamento usual, e sim através da definição e exploração de posições no ambiente, escolhidas através da análise visual do mesmo.

O modelo criado pelo robô é uma árvore de nós. Cada adjacência carrega a informação de distância entre dois nós e a diferença angular entre eles. Cada nó guarda a imagem panorâmica criada naquele ponto do espaço. Esta modelagem possibilitaria uma futura navegação baseada em busca de imagens, seja de objetos, lugares, ou outros.

A Figura 6-1 ilustra um possível ambiente que o robô teria explorado. Os círculos em preto representam os locais que o robô explorou, ou seja, os nós

criados. As linhas entre os círculos representam os caminhos que o robô navegou entre um nó e outro. Também são apresentados os símbolos θ e I, que representam respectivamente os ângulos entre os nós e as imagens panorâmicas. A representação deste modelo topológico é uma árvore tal como a apresentada na Figura 6-2, que corresponde à exploração da Figura 6-1.

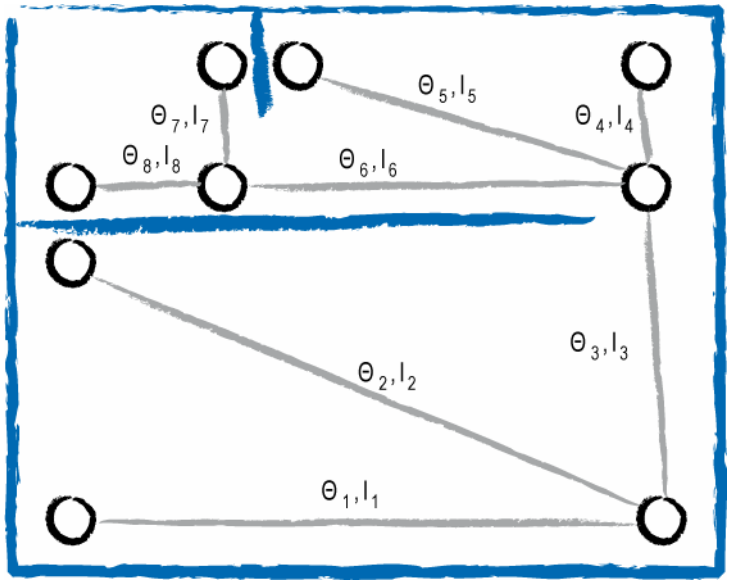

Figura 6-1: Mapa de um ambiente navegado por nós

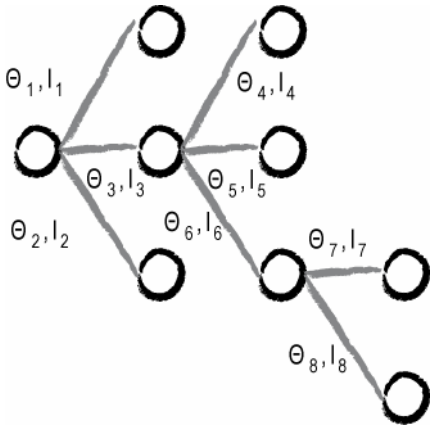

Figura 6-2: Árvore de nós

O robô deve navegar através de diferentes locais do ambiente de modo a construir um modelo topográfico que é esta coleção de nós em um grafo. Cada local explorado pelo robô é marcado como um nó no mapa criado. Para cada nó

explorado, uma série de procedimentos são realizados. Uma visão geral do algoritmo pode ser vista na Figura 6-3.

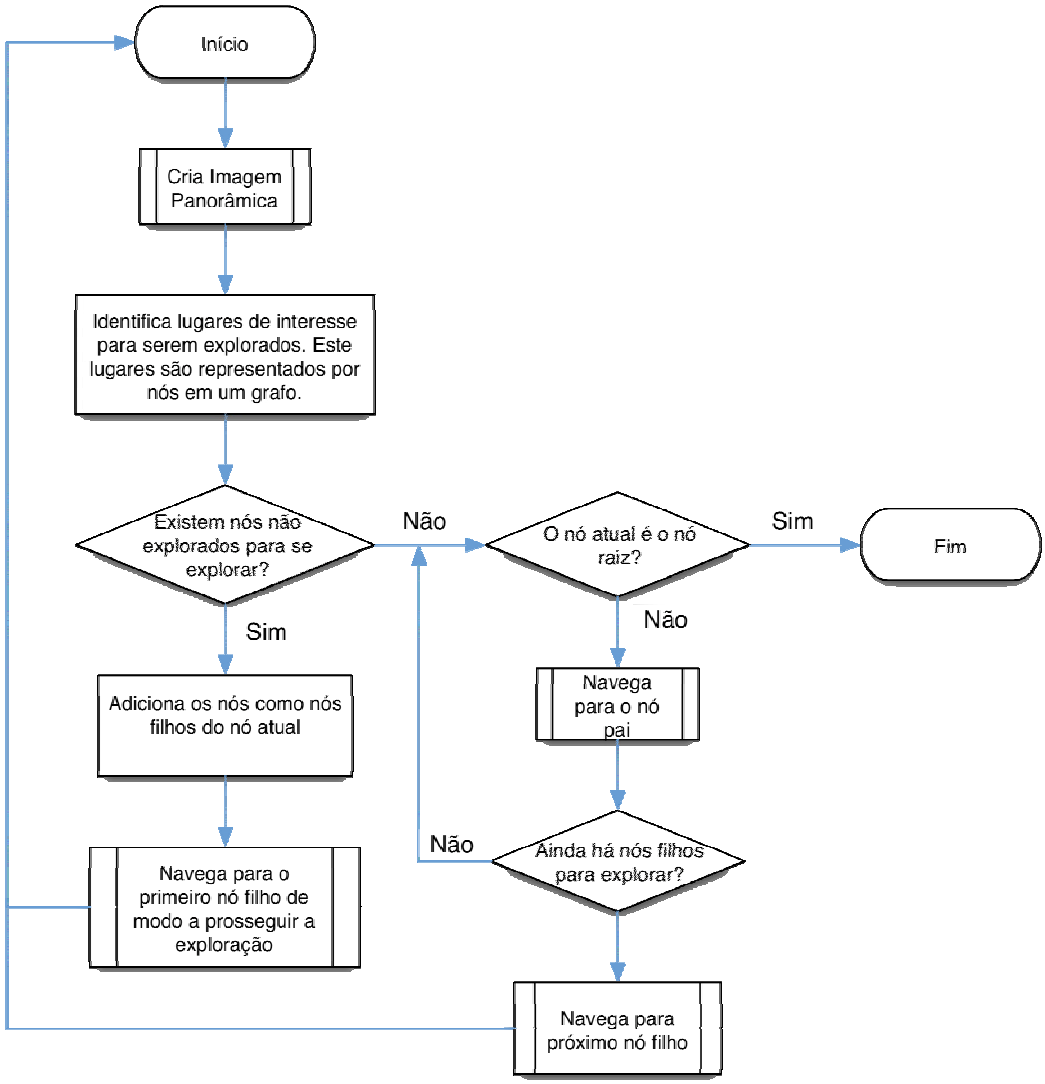

Figura 6-3: Visão Geral do Algoritmo

O algoritmo pode ser entendido como se segue:

1. A primeira posição do robô é definida como o nó raiz da árvore a ser criada;
2. Para o nó atual, registra-se uma imagem panorâmica do ambiente;
3. A partir da imagem panorâmica, novos lugares para serem explorados são escolhidos. Cada um destes lugares passa a ser representado como um nó ainda não explorado no modelo criado;

4. Verifica-se a existência de nós a serem explorados a partir do nó atual. Caso existam, navega-se para o primeiro destes nós de modo a prosseguir a exploração. Deste novo nó, a exploração segue da etapa 2;
5. Caso não existam nós para serem explorados a partir do nó atual, navega-se para o nó pai do nó atual;
6. Após chegar no nó pai, verifica-se se este nó é o nó raiz. Caso seja, o procedimento foi terminado. Caso não seja, o procedimento segue da etapa 7;
7. Uma nova verificação é feita de modo a se constatar se existem nós não explorados a partir do nó atual. Caso não existam, navega-se para o nó pai do nó atual e se prossegue a partir da etapa 5. Caso existam nós filhos para serem explorados, navega-se para o primeiro destes nós e segue a exploração da etapa 2;

A Figura 6-4 mostra um exemplo do algoritmo que ilustra a construção do modelo interno ao longo de uma exploração. No primeiro momento o robô está no nó raiz e encontra 3 novos nós para serem explorados. O robô se dirige ao primeiro destes nós e verifica que não há novos nós para serem explorados a partir deste, voltando então para o nó pai. Do nó raiz o robô vai para o segundo nó filho e neste encontra mais 3 nós para serem explorados. Se dirige ao primeiro destes nós e encontra mais 2 nós a serem explorados. O exemplo termina aqui mas em um caso real, o robô só terminaria o algoritmo após explorar todos os nós definidos.

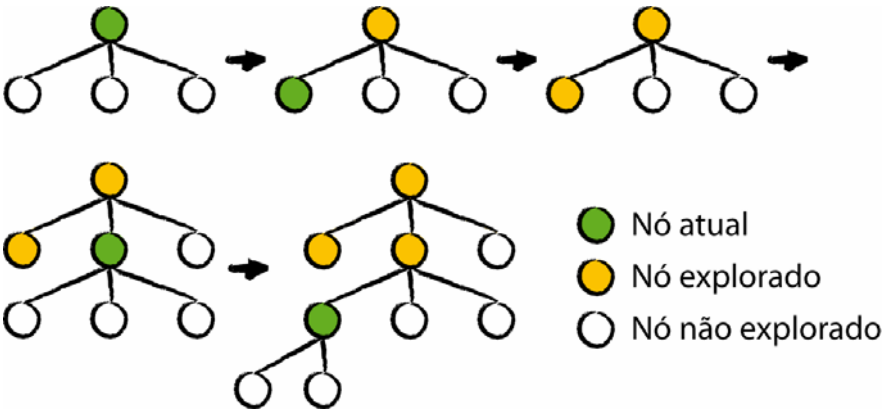

Figura 6-4: Exemplo de como é feita a exploração

O algoritmo consiste do uso de quatro componentes primários:

- Componente 1 - Geração de imagens panorâmicas: Para cada posição explorada, o robô captura uma série de imagens ao redor de si. Com estas imagens cria-se uma imagem panorâmica do ambiente para aquela posição. Esta componente é apresentada na seção 6.2;
- Componente 2 - Identificação de lugares de interesse para futura exploração: Através das imagens panorâmicas, são definidos os locais que o robô deverá explorar. A componente 2 será vista na seção 6.3;
- Componente 3 - Navegação para lugares já conhecidos: Quando o robô sai de um nó filho e busca retornar para um nó pai. Esta componente é descrita na seção 6.4;
- Componente 4 - Navegação para lugares ainda não conhecidos: A seção 6.5 explica o que é feito quando o robô navega na direção de um lugar ainda não conhecido, onde um novo nó será criado;

Para após a exploração, é proposta uma quinta componente:

- Componente 5 – Refinamento do modelo criado através de algoritmos genéticos: Esta componente tem como objetivo refinar o grafo criado ao longo da exploração e foi experimentada dentro de ambiente simulado. Esta componente será apresentada na seção 6.7;

Nas seções dadas, os detalhes de cada componente e suas possíveis implementações são apresentados, com exceção da componente 5. Na presente seção 6.1, será apresentada uma pequena descrição das abordagens utilizadas para cada uma das componentes do algoritmo.

6.1.1. Panorâmicas

Em cada nó, uma série de imagens é obtida sequencialmente com o robô girando 360^{o} em torno de uma localização dada. Consecutivamente, estas imagens devem ser combinadas automaticamente de modo a construir uma composição panorâmica relativa àquele nó.

Os métodos de combinação de imagens, comumente conhecidos como métodos de registro de imagens, consistem em encontrar pontos, ou regiões,

correlatos entre duas imagens para então calcular um modelo que transforme uma das imagens para que esta possa ser sobreposta à outra (como discutido no capítulo 4). Por fim a sobreposição é feita através de ajuste radiométrico dos *pixels* de cada uma das imagens.

Para se encontrar as regiões coincidentes entre as imagens subseqüentes optou-se por trabalhar com Correlação Cruzada. Esta escolha foi feita devido ao custo computacional da técnica em relação a técnicas mais robustas como SIFT. Posteriormente são aplicados procedimentos diversos de modo a se eliminar regiões mal correlacionadas.

O capítulo 4 descreve algumas teorias e técnicas de registro de imagens utilizados no presente trabalho. A seção 6.2 apresenta como foram desenvolvidos os experimentos para a geração automática das imagens panorâmicas.

A partir da imagem panorâmica deve se fazer a identificação dos lugares para os quais o robô irá navegar como será visto na próxima seção.

6.1.2. Identificação de Lugares de Interesse

A segunda componente do algoritmo é a identificação de potenciais nós filhos para uma localização dada. Para cada nó, potenciais nós filhos são escolhidos através de uma análise da imagem panorâmica. Esta análise tem como objetivo a identificação de áreas de interesse. Estas áreas devem ser selecionadas de acordo com características que as diferenciem do resto da imagem.

A abordagem escolhida para se definir as regiões de interesse da imagem panorâmica foi baseada no cálculo de entropia para diferentes seções da imagem. A entropia de uma imagem é uma métrica da quantidade de informação presente no histograma de tons de cinza da mesma. Ao se buscar regiões com grande entropia, é possível se encontrar áreas distintas com grandes variações na imagem tais como quinas, portas, objetos, e outros.

Se determinada região de interesse contiver quantidade suficiente de informação, então esta área é identificada como um nó potencial e deve ser explorada.

A seção 6.3 do presente capitulo apresenta algumas possibilidades para se fazer a segmentação de uma imagem de modo a se obter regiões de interesse

baseadas em informação. Os métodos descritos se baseiam em dividir uma imagem através de um *quadtree* que trabalha com o valor de entropia das regiões para determinar contínuas sub-divisões.

Identificados os locais de interesse, o robô deverá fazer a navegação para estes como será descrito na seção seguinte.

6.1.3. Navegação para Nós Desconhecidos

A navegação para nós desconhecidos consiste em alinhar o robô continuamente com a região de interesse escolhida na imagem panorâmica que represente o nó a ser explorado. Tendo o robô se direcionado corretamente, este segue em frente até encontrar algum obstáculo ou então começar a perder de vista a região de interesse procurada.

Escolheu-se trabalhar com descritores SIFT de modo a encontrar a região de interesse nas imagens obtidas para a partir de então se fazer o acompanhamento destas regiões durante a navegação do robô.

O procedimento pode ser entendido através de duas etapas:

- Deve-se encontrar na imagem vista a região de interesse que represente o nó a ser explorado. Esta correspondência é feita através de descritores SIFT;
- Tendo-se encontrado pontos em comum entre a região de interesse e a imagem vista pelo robô, passa a se fazer um acompanhamento visual destes pontos através de *Visual Tracking* por correlação, permitindo o alinhamento adequado constante do robô com o nó pai. Este procedimento é feito até se encontrar algum obstáculo ou até que a região de interesse comece a sair da tela por cima ou por baixo;

A seção 6.4 descreve as possibilidades de se trabalhar com *Visual Tracking* de pontos na tela a partir de correlação.

A seção 6.5 explica em maiores detalhes a navegação para nós desconhecidos utilizando-se de descritores SIFT e de *Visual Tracking*.

Para voltar para os nós já conhecidos outros procedimentos são abordados como visto na próxima seção.

6.1.4.
Navegação para Nós Conhecidos

A navegação para nós já conhecidos usa como base de comparação a imagem que se espera ver quando se chegar ao nó. Esta imagem é extraída da panorâmica do nó pai, para o qual se navegará, tendo como base a direção em que aponta o nó filho para o nó pai.

Tendo esta imagem como referência, dois procedimentos diferentes foram experimentados de modo a alinhar o robô com o nó pai e navegar para este. As duas abordagens diferentes tomadas foram: transformações invariáveis e uso da transformação SIFT junto com *Visual Tracking*.

Na primeira abordagem, a imagem de referência é constantemente comparada com a visão atual do robô utilizando-se uma transformação invariável. Quando a comparação começa a retornar valores inferiores ao esperado, é feita uma busca pelo melhor alinhamento do robô para que a navegação possa continuar. Algumas das transformações apresentadas no capítulo 2 foram testadas.

A segunda abordagem consiste de duas etapas:

- Busca da área da imagem atual correspondente à imagem de referência através de descritores SIFT;
- Realização de um acompanhamento dos pontos encontrados por correlação em comum entre imagens na etapa anterior. Este acompanhamento é em tempo real (5 a 10 Hz nos experimentos realizados) permitindo um constante alinhamento do robô com o nó pai;

A seção 6.4 do presente capítulo descreve como é feito o *Visual Tracking* de modo a se perseguir os pontos encontrados através do uso da transformação SIFT.

A seção 6.6 descreve em detalhes tanto a navegação para um nó conhecido usando a transformação SIFT quanto a primeira abordagem feita do problema utilizando-se transformações invariáveis.

6.2.
Componente 1 - Criando Imagens Panorâmicas Automaticamente

A presente seção descreve a componente 1 do algoritmo de exploração proposto.

É desejado criar uma imagem panorâmica através de uma seqüência de imagens obtidas por um robô girando sobre seu próprio centro. O robô vai girando de um número pequeno de graus por vez e obtendo as imagens a serem combinadas. Após um giro completo de 360^{o}, estas imagens devem ser utilizadas para compor uma imagem panorâmica da posição em que o robô efetuou o giro. Para tal é necessário combinar cada par de imagens vizinhas através de um processo de registro automático.

A combinação destas imagens a partir da informação de ângulo (que poderia vir a ser extraída do sistema utilizado) implicaria em erros de sobreposição das imagens derivados dos erros das medidas de ângulos. Para o sistema experimental utilizado, o problema é ainda mais crítico, dado que a informação de orientação do robô não pode ser obtida diretamente.

Este mapeamento panorâmico pode ser usado num segundo momento para que se conheça a imagem que o robô estaria vendo em um ângulo arbitrado, mesmo que o robô não tenha obtido uma imagem para aquele ângulo específico.

Outro uso futuro está em se utilizar as imagens panorâmicas obtidas para se escolher áreas interessantes do ambiente para serem exploradas. A partir da comparação da panorâmica com a visão do robô, este pode se dirigir para estas áreas de interesse e explorá-las.

O capítulo 4 apresentou uma visão geral de como são usados métodos de registro. Para se compor a panorâmica algumas das técnicas apresentadas foram utilizadas.

A técnica proposta encontra os pontos de controle automaticamente através da correlação de blocos da imagem a ser transformada com a imagem de referência. Os parâmetros da função de transformação são obtidos utilizando-se a matriz pseudoinversa da jacobiana que relaciona a posição na imagem base dos pontos de controle com os parâmetros. Por fim, a sobreposição das imagens é feita utilizando-se ajuste radiométrico das intensidades dos *pixels* sobrepostos por *dégradée*.

A imagem panorâmica resultante pode ser utilizada para extração da imagem de um ângulo arbitrado. Isto pode ser feito porque as posições no panorama referentes ao centro de cada imagem obtida são guardadas tal como os ângulos referentes a estas imagens. Daí, esta informação pode ser utilizada para se

interpolar onde estaria no panorama o centro de uma imagem em um ângulo arbitrado.

A seção seguinte descreve como foi feita a comparação das imagens de um modo geral.

6.2.1. Comparação Entre Imagens e Construção da Matriz de Transformação *T*

Optou-se por trabalhar com um método rápido e eficiente, que atendesse ao sistema experimental usado, trabalhando-se com correlação. A busca dos pontos em comum por correlação cruzada não é tão poderosa quanto o uso de técnicas mais complexas, como SIFT, mas tem como vantagem menor custo computacional. Apesar de não ser extremamente robusto, o método demonstrou ser confiável e eficiente para a geração de panorâmicas usando-se o ER1.

Os experimentos feitos para se comparar cada par de imagem seguiram de um modo geral as seguintes etapas:

- Detecção de pontos de interesse: Como dito, optou-se trabalhar com correlação cruzada. As imagens são sub-dividas em blocos para então serem casadas em outras imagens. Inicialmente blocos com baixa entropia podem ser descartados. Este tópico está detalhado na seção 6.2.2;
- Descarte de pontos inconsistentes: Algumas das possibilidades testadas serão apresentadas na seção 6.2.3, outras foram vistas no capítulo 4;
- Cálculo final da matriz de transformação T como visto no capítulo 4;

A detecção de pontos de interesse está detalhada na seção a seguir.

6.2.2. Detecção e Casamento dos Pontos de Controle por Correlação Cruzada

Sempre se busca encontrar pontos de uma imagem em outra, sendo a primeira a imagem a ser transformada e a segunda a imagem de referência. Estes pontos comuns entre as imagens serão chamados de pontos de controle.

Inicialmente, deve-se dividir a imagem a ser transformada em blocos de tamanho N x M. Estes blocos serão procurados na imagem de referência. A idéia está exemplificada na Figura 6-5 que mostra à esquerda a imagem a ser transformada dividida em blocos, e à direita a imagem de referência onde alguns blocos são encontrados.

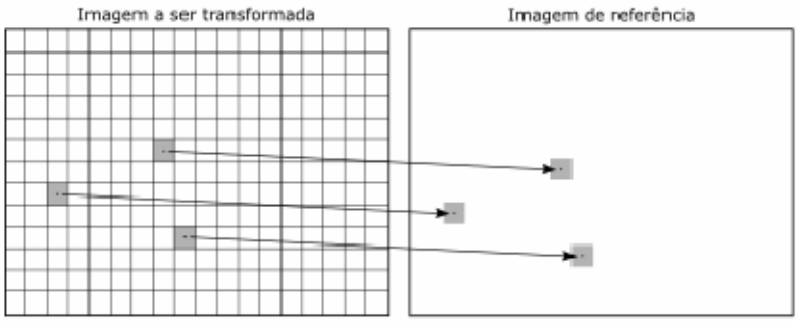

Figura 6-5: Detecção e casamento de pontos de controle

É interessante que o tamanho definido seja adequado: nem muito pequeno, dificultando a procura por falta de informação suficiente, nem muito grande, o que torna a procura mais lenta e ineficaz.

De um modo geral, ao dividir-se a imagem em blocos, pode-se criar um número de blocos muito grande. Portanto, caso todos os blocos sejam procurados, pode-se ter um tempo de procura muito grande e até desnecessário. É interessante buscar maneiras de se escolher um conjunto de blocos do todo que sejam mais adequados para a procura de modo a tornar a detecção dos pontos de controle mais rápida.

Percebeu-se que blocos com maior entropia têm maior chance de serem encontrados corretamente na imagem de referência. Isto acontece porque quanto mais distinto for um bloco, maior será a chance de se encontrar um bloco em outra imagem que case com ele de modo único. Blocos com baixa entropia são mais uniformes e portanto mais fáceis de serem erroneamente identificados.

Portanto, é uma boa idéia de pré-seleção de blocos, se eliminar aqueles com baixa entropia. Isto pode ser feito tanto definindo-se um limiar quanto definindo-se um número *k* de blocos com maior entropia para serem procurados.

A seção 6.3.2 deste capítulo descreve o uso da entropia de imagens.

A entropia de cada um destes blocos é calculada pela fórmula de entropia [17]:

$$H(q_1,q_2,...,q_n) = -\sum_{k=1}^{n} q_k \log_2 q_k \tag{165}$$

$$q_i = f_i / N \quad \text{para} \quad \text{i} = 1 \ldots \text{N}_{\text{gray}} \tag{166}$$

Onde:

- q: representa o histograma de tons de cinza da imagem;
- f_i: Número de *pixels* na imagem com o tom de cinza i;
- N: Número de *pixels* da imagem;
- Ngray: Número de níveis de cinza utilizados;

Então, selecionam-se k blocos com maior entropia. Como dito, isto é feito de modo a descartar aqueles blocos que por terem baixa quantidade de informação, poderiam levar a casamentos incorretos de blocos.

Os blocos restantes são procurados na imagem de referência através da maximização da correlação entre os blocos e a imagem como apresentado em 4.2.

Obtém-se a posição x, y de casamento de cada bloco na imagem base encontrando-se x' e y' que maximize a correlação cruzada CC(x',y'). Os blocos têm seus centros utilizados como pontos de controle.

O casamento dos blocos por correlação possui a desvantagem de não funcionar muito bem para blocos que tenham sofrido grande distorção ou variação de intensidade. Observe que o procedimento apresentado é bem específico para o casamento de imagens com mudanças de escala e rotação bem pequenas, se adaptando bem para o caso da construção de panorâmicas e sendo bastante eficiente em termos de processamento em relação a procedimentos mais complexos para a definição dos parâmetros do modelo de transformação entre as imagens como os vistos no capítulo 4.

É proposto que após encontrar blocos correspondentes, como apresentado, seja feita uma seleção dos pontos de controle consistentes utilizando os métodos aqui propostos e outros apresentados no capítulo 4.

Após encontrar blocos em comum, é interessante fazer o descarte de pontos de controle inconsistentes como visto na próxima seção.

6.2.3. Descarte de Pontos de Controle Inconsistentes

Após ser feita a busca dos blocos por correlação, muitos pontos não serão casados corretamente. Algumas idéias foram experimentadas de modo a descartar pontos inconsistentes e serão aqui descritas.

6.2.3.1. Baixa Correlação

É feito um processo de seleção de pontos eliminando blocos que receberam baixo valor de correlação. Esse valor é relativo à correlação máxima entre bloco e imagem de referência. Tal como para a seleção de blocos por entropia, é possível fazer a eliminação destes pontos utilizando-se um limiar ou escolhendo um número *k* de pontos com maior correlação.

O uso de um limiar de correlação é bem interessante e eficaz. Porém, tem como desvantagem não ser possível se determinar o número de pontos restantes, ou mesmo se haverá pontos restantes.

O uso de um número determinado de pontos com melhores valores de correlação implica em ou não se eliminar pontos mal correlacionados, ou se eliminar pontos com boa correlação.

6.2.3.2. Má Correlação

Uma outra maneira de se verificar se um ponto está sendo bem ou mal correlacionado consiste em conferir se este obteve alta correlação para um grande número de pontos na imagem de referência. Quando isto ocorre pode-se entender que este ponto não está sendo identificado de modo único porque existem vários locais para os quais o bloco possui grande semelhança. Será portanto qualificado como um ponto de má correlação.

A idéia proposta está em definir C_{max} como o maior valor de correlação obtido e a partir de então contar para quantos pontos se obteve uma correlação próxima desta. O valor utilizado para a comparação é definido como:

$$Caux = \rho C\max \tag{167}$$

Com $0 < \rho < 1$.

A partir de então, é contado o número de pontos com correlação superior a C_{max}. Caso o percentual de pontos computado (em relação ao número total de pontos correlacionados) seja superior a β dado, então este ponto pode ser descartado e qualificado como um ponto com má correlação.

Esta técnica também será usada mais adiante na descrição dos procedimentos de "*Visual Tracking"*.

6.2.3.3. Eliminação por Análise Amostral

É proposta uma análise dos vetores que levam o bloco casado da imagem de referência para o bloco original da imagem a ser transformada. São descartados aqueles blocos cujo vetor de movimento fuja do padrão geral. Isto é feito de modo a tentar garantir que os blocos resultantes tenham sido encontrados na imagem de referência corretamente.

A representação pontual dos blocos é dada pelo seu centro. Dados os centros dos blocos que apresentaram correspondências, é possível definir um vetor que apresenta o movimento relativo de um bloco ao outro. Sabendo-se da posição x_1,y_1 do bloco w na imagem a ser transformada e da posição x_2,y_2 na imagem base, temos um vetor de movimento de cada bloco dado por:

$$V_w = \begin{bmatrix} x_2 - x_1 \\ y_2 - y_1 \end{bmatrix} \tag{168}$$

Este vetor possui norma:

$$[V_w] = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{169}$$

E ângulo:

$$\theta_w = \arctan\left(\frac{y_2 - y_1}{x_2 - x_1}\right) \tag{170}$$

Nas imagens panorâmicas a serem compostas, a relação de coordenadas entre imagens subseqüentes pode ser razoavelmente aproximada por uma

translação em *(x,y)*. Portanto, o vetor de movimento calculado, caso corresponda a pontos equivalentes entre as imagens, constitui de uma boa aproximação da transformação entre coordenadas.

Consecutivamente, são calculados as médias e o desvio padrão para as normas e ângulos dos blocos. São descartados todos blocos que ficam fora do desvio padrão para norma ou ângulo.

Após o descarte de blocos, o procedimento pode ser repetido até que nenhum bloco seja descartado.

Observe que este procedimento não trará bons resultados para comparação de imagens que sofreram transformações geométricas ou de rotação, mas somente para casos em que a transformação entre imagens pode ser aproximada por uma translação. Como este é o caso de imagens panorâmicas, esta proposta traz bons resultados.

Uma desvantagem da técnica está em que o número de pontos descartados pode ser grande, podendo-se chegar a somente dois pontos restantes. Se este for o caso, é necessário aproximar a transformação T por uma transformação Procrustes.

6.2.3.4. RANSAC e Ajuste de Matriz de Pesos

As técnicas RANSAC e ajuste da matriz de pesos apresentadas no capítulo 4 trazem bons resultados no momento de se eliminar pontos mal casados. O uso destas técnicas também é interessante para se ajustar o modelo de transformação T.

Outra técnica que poderia ser aplicada aqui é a Transformada de Hough como descrita na seção 4.4.1, porém esta não foi utilizada nos experimentos realizados.

Finalmente, é visto na seção seguinte como é criada a imagem panorâmica utilizando as técnicas apresentadas.

6.2.4.
Criando a panorâmica – Registro de Múltiplas Imagens

Para se fazer o registro de uma seqüência de imagens são encontrados os pontos de controle para cada par de imagens vizinhas e estimados os parâmetros de todas as transformações entre imagens vizinhas, para só então fazer a união de todas.

É dada uma seqüência de *n* imagens $I_1, I_2, \cdots, I_n$, ordenadas em série de modo que I_x esteja intercalada entre I_{x-1} e I_{x+1} para todo *x*. Encontram-se os pontos de controle entre cada par de imagens vizinhas e são obtidas as transformadas T_{i+1} que levariam as imagens I_{i+1} para I_i (observe que $i+1$ se refere à imagem a ser transformada e *i* à imagem base).

Desta maneira são calculadas as transformadas entre todas as imagens vizinhas e fica estabelecido que:

$$T_1 = \begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{vmatrix} \tag{171}$$

A partir de então para se calcular a transformação T^i que levaria a imagem I_i para a panorâmica, basta fazer:

$$T^i = T_1 \cdot \ldots \cdot T_i \tag{172}$$

Por fim cada imagem é por sua vez levada para a panorâmica fazendo-se o devido ajuste radiométrico [39].

Veja na Figura 6-6 o método ilustrado. A figura apresenta uma seqüência de imagens para as quais são calculadas as matrizes T_i para cada par, e por fim são calculadas as matrizes T^i que são usadas para construir a panorâmica.

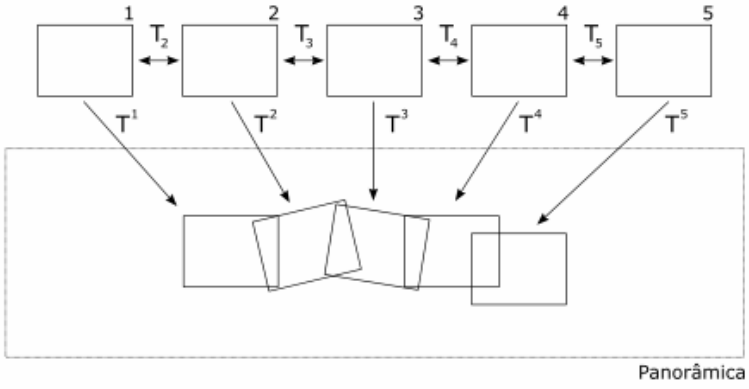

Figura 6-6: Registro de múltiplas imagens

A panorâmica criada pode ser usada para se extrair uma imagem vista em ângulo especificado. Isto é descrito na seção seguinte.

6.2.5. Extraindo Imagens da Panorâmica

Cada imagem I_i é obtida em um ângulo Θ_i específico. Portanto, para calcular onde estaria na imagem panorâmica o centro relativo a cada ângulo Θ_i, basta aplicar a transformação T^i usada para a posição do centro de uma imagem de tamanho *sx* x *sy* dado pela aproximação de:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} [sx/2] \\ [sy/2] \end{bmatrix} \tag{173}$$

Desta maneira pode-se criar uma tabela do tipo:

Tabela 6-1: Tabela que acompanha a imagem panorâmica

Ângulo Θ_i	Posição em *x*	Posição em *y*

Portanto ao se arbitrar um ângulo qualquer, basta calcular o centro da imagem a ser extraída através de interpolação utilizando-se a tabela criada.

Caso o modelo de transformação seja o Procrustes, pode-se armazenar na tabela o valor do parâmetro Θ das transformadas e trabalhar com sua interpolação para saber o ângulo da imagem a ser extraída da panorâmica. Outro modo seria trabalhar com imagens horizontais somente.

O tamanho das imagens deve respeitar o tamanho das imagens da panorâmica.

A partir das imagens panorâmicas é definido para onde o robô irá navegar. Esta é a componente 2 do algoritmo de exploração e é vista na seção seguinte.

6.3.
Componente 2 - Segmentação da Imagem por *Quadtree* Baseada em Entropia

6.3.1.
Visão Geral

Sujan propõe em [48] o uso de um algoritmo *quadtree* baseado em entropia para encontrar regiões de interesse em uma imagem. Aqui este algoritmo é explorado e adaptado de modo a segmentar uma imagem em regiões de interesse. O objetivo final é definir áreas a serem exploradas pelo robô de acordo com regiões que apresentem maior quantidade de informação para serem exploradas.

A idéia do algoritmo proposto é de dividir a imagem em regiões utilizando-se a técnica de *quadtree*. Para cada região da imagem, esta será subdividida em novas regiões caso possua quantidade de informação desejada. Por fim, o particionamento *quadtree* é utilizado para fazer a segmentação da imagem em regiões e pontos de interesse.

Na Figura 6-7, é apresentada uma visão geral do algoritmo.

O procedimento pode ser entendido através de três componentes:

- Pré-processamento: Aplicação de filtros na imagem de modo a eliminar efeitos de iluminação e possíveis ruídos;
- Divisão da imagem: A imagem é recursivamente sub-dividida através da técnica "*quadtree*" de regiões com alta quantidade de informação. No caso, alta entropia;
- Pós-processamento: A partir da árvore "*quadtree*" gerada, é feita a segmentação da imagem em áreas e pontos de interesse (*feature points*);

As próximas seções detalham o procedimento de segmentação proposto. A seção a seguir apresenta o cálculo de entropia de uma imagem e descreve suas características.

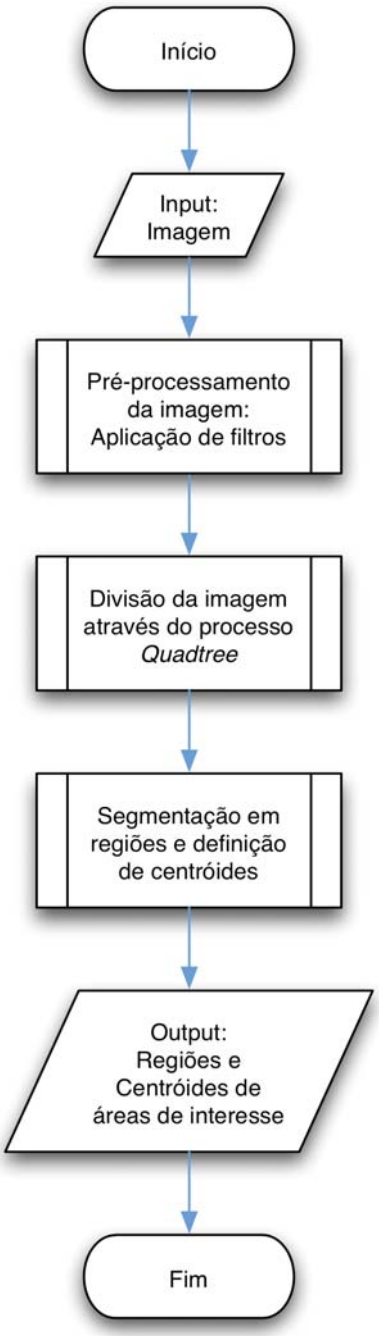

Figura 6-7: Visão geral do algoritmo de segmentação

6.3.2. Cálculo de Entropia

A informação ganha ao se observar um evento específico dentre uma amostra de possibilidades é descrita pela função de entropia dada pela eq. (165) vista na seção 6.2.2.

Esta definição de informação [17] também pode ser interpretada como o menor número de estados (bits) necessários para descrever completamente uma amostra de dados.

A teoria da informação dá ênfase na descrição de sinais 1-D. Para sinais em 2-D como imagens, o histograma de tons de cinzas, visto na eq. (166) da seção 6.2.2, pode ser usado para definir a distribuição de probabilidades.

Com esta definição, a entropia máxima de uma imagem será dada por uma distribuição dos níveis de cinzas uniforme, ou seja, máximo contraste. Quanto menos uniforme é o histograma, menor será a entropia da imagem.

Para uma imagem, existem diversas interpretações para a entropia, incluindo:

- A incerteza média relativa ao histograma;
- A quantidade teórica mínima de bits necessários para se codificar o histograma;
- A medida de aleatoriedade na distribuição de tons de cinza da imagem;

Um aumento na entropia corresponde à maior incerteza e maior informação contida na imagem. Aqui o interesse está na terceira interpretação de onde se conclui que a entropia de uma imagem pode ser usada para se apontar regiões de interesse na mesma.

A próxima seção descreve como é feito o procedimento *quadtree* utilizando a métrica de entropia apresentada.

6.3.3. Processo *Quadtree*

No particionamento *Quadtree*, a imagem é sucessivamente dividida em 4 quadrantes. Cada quadrante é avaliado de modo a se determinar se novas divisões necessitam ser realizadas. Esta avaliação é baseada em alguma métrica. O processo deve continuar recursivamente até que nenhuma nova divisão seja necessária. A Figura 6-8 ilustra o procedimento para uma imagem que inicialmente é divida em quatro regiões, das quais somente a região 2 é novamente dividida.

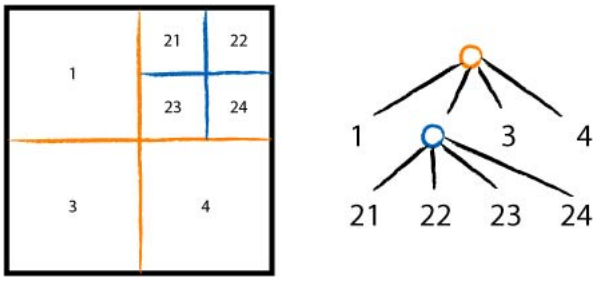

Figura 6-8: *Quadtree*

A métrica utilizada pode ser escolhida de acordo com a necessidade da aplicação. O particionamento *Quadtree* proposto usa como condição para a divisão a quantidade de informação, entropia, da região a ser subdividida. O particionamento *Quadtree* é feito dividindo-se as regiões que estiverem acima de determinado limite escolhido. Repare que alguma outra medida poderia ser usada para tal fim.

O objetivo de se determinar a quantidade de informação na imagem é de se identificar regiões onde existam mudanças significativas. Estas mudanças poderiam indicar a presença de cantos, extremidades, quinas, objetos e outras áreas interessantes para serem exploradas.

Áreas com grande quantidade de ruído também poderiam ser identificadas com alta entropia, por isso indica-se o uso de filtros para se fazer um pré-processamento da imagem antes que esta seja subdividida.

Na Figura 6-9 o procedimento está apresentado em diagrama de fluxo do método usando-se entropia como métrica de decisão. Inicialmente a imagem é sub-dividida em quatro regiões. Cada região pode ser filtrada. Caso a entropia de uma região seja superior a um limite, entra-se em um processo recursivo onde esta será subdividida. O processo termina quando não houver mais regiões a serem subdivididas.

A Figura 6-10 apresenta um exemplo do aplicação da técnica utilizando-se entropia. A figura apresenta uma imagem após ter sido particionada através do *quadtree*. As linhas brancas mostram as divisões encontradas.

A próxima seção explicará como usar a divisão *quadtree* apresentada para se segmentar uma imagem em áreas de interesse.

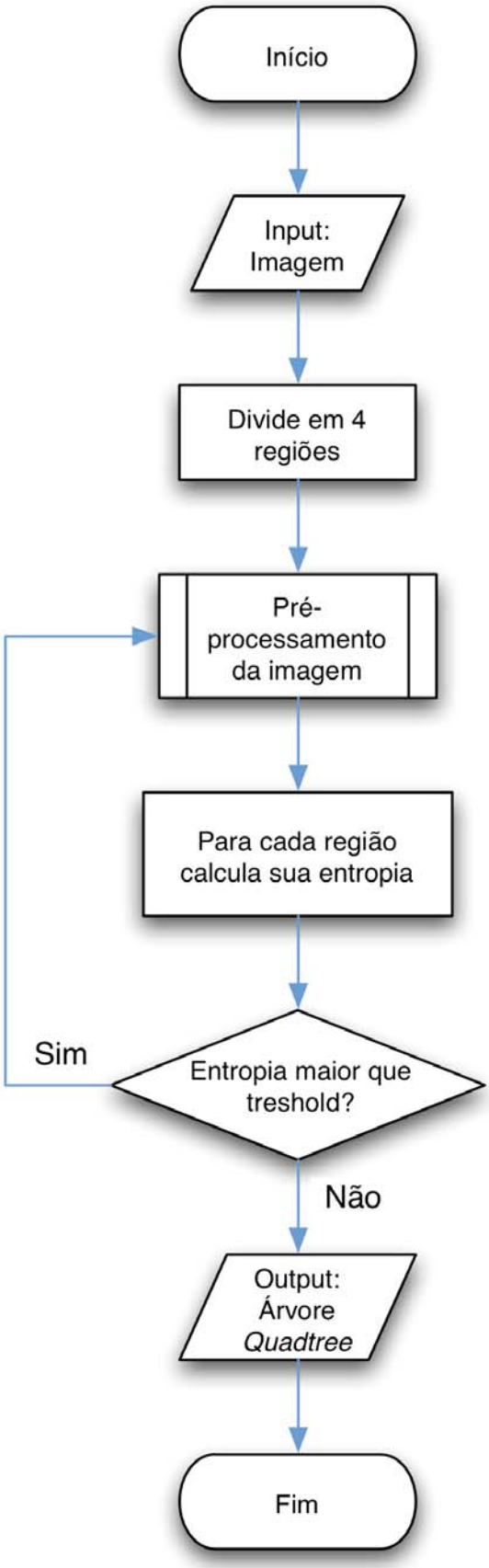

Figura 6-9: Divisão por processo *Quadtree*

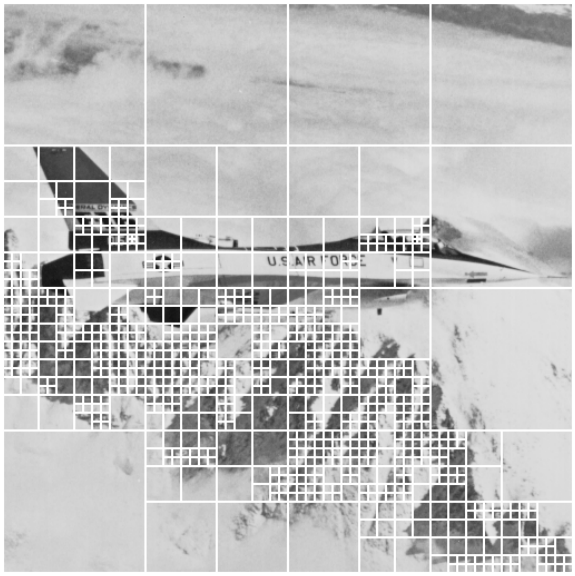

Figura 6-10: Exemplo da divisão baseada em entropia

6.3.4. Segmentação em Áreas de Interesse e Definição de *Feature Points*

Após o particionamento *quadtree* da imagem, deseja-se fazer uma segmentação da mesma em regiões de interesse que representem áreas com grande quantidade de informação, como já foi explicado. Para se fazer esta segmentação são propostos três métodos diferentes:

- Método padrão;
- Método Open-Close;
- Método Close-Open;

Estes métodos tem como linha geral agrupar os menores quadrantes encontrados e aplicar operações morfológicas nestes de modo a se conseguir criar conglomerados que representem as regiões de interesse da imagem. Estes métodos serão descritos nas seções seguintes.

A primeira coisa a ser feita é a criação de uma imagem modelo, do mesmo tamanho da imagem analisada, com todos os seus *pixels* zerados. Esta imagem será usada para se criar o mapa das regiões segmentadas, ou seja, indicará se determinado *pixel* faz parte ou não de uma região.

O procedimento, independente do método utilizado, pode ser resumido da seguinte maneira:

1. Criação de uma imagem *template* com tamanho igual à imagem original com todos os *pixels* desligados. Como dito anteriormente;

2. Todos os *pixels* que estejam nas bordas das regiões do *quadtree* são ativados. Isto facilitará o futuro agrupamento das regiões ;
3. Liga todos os *pixels* que estejam nas menores regiões do *quadtree*. Isto pode ser feito para as regiões até um determinado tamanho. A escolha deste tamanho pode ser pré-determinada ou feita a partir de um número de menores tamanhos;
4. Aplica-se uma série de operações morfológicas no *template* de acordo com o método escolhido. Estas operações serão descritas a partir da próxima seção, quando são detalhados os diferentes métodos propostos;
5. Faz-se um pós-processamento no *template* de modo a se descartar regiões muito pequenas;

O resultado deste processo será uma série de regiões. Por fim, são calculados os centróides das regiões segmentadas. Estes centróides são definidos como *feature points*, ou seja, pontos de interesse para se prosseguir a exploração do robô.

As três seções seguintes descrevem os métodos de segmentação propostos.

6.3.5. Método Padrão

O primeiro método apresentado foi denominado de método padrão por ter sido originalmente proposto em [48] , os outros métodos apresentados foram desenvolvidos no presente trabalho.

O método padrão consiste em se aplicar a operação morfológica de abertura na imagem *template* com um disco de raio *r* dado. O diagrama de fluxo do método padrão é apresentado na Figura 6-11. Este método não agrupa muito bem as regiões como pode ser visto na Figura 6-12 que mostra um exemplo do método aplicado. O que pode ser visto na Figura 6-12 são as diversas regiões resultantes do método empregado.

A próxima seção apresenta outro método de segmentação baseado nas operações morfológicas de abertura e fechamento.

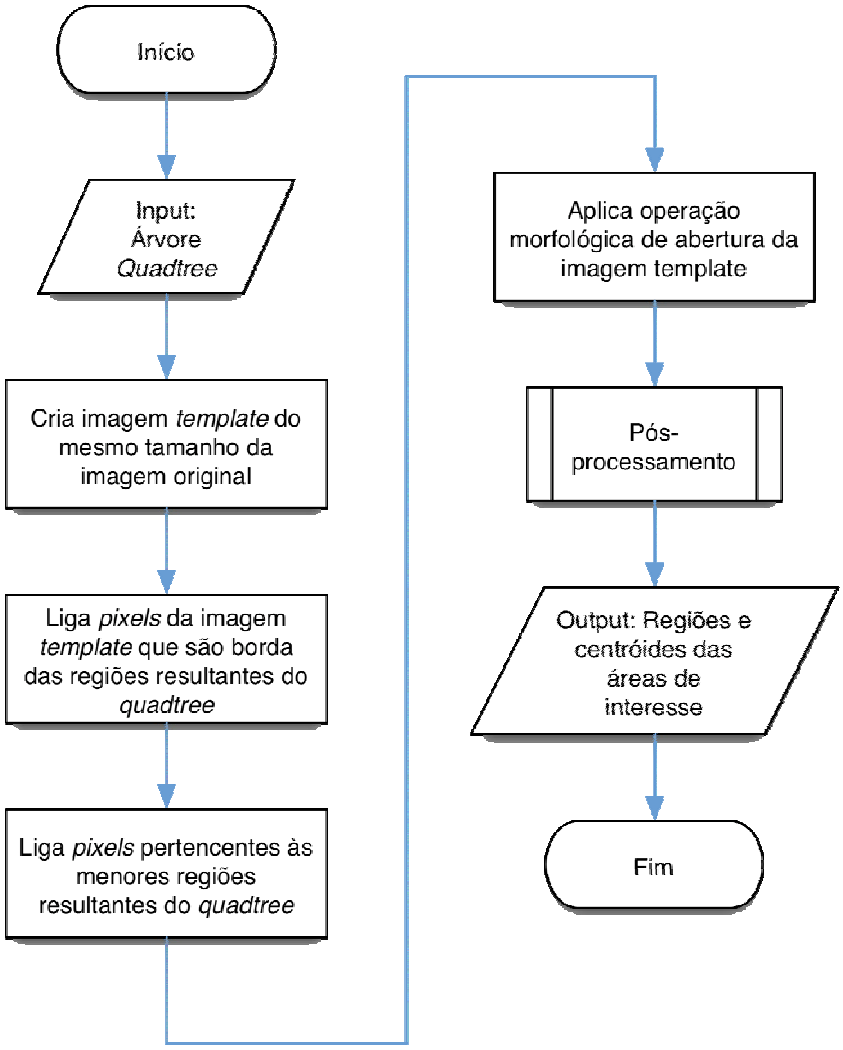

Figura 6-11: Método padrão

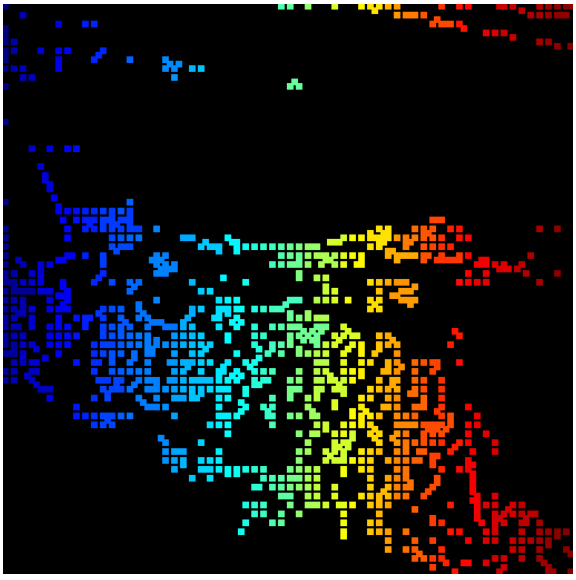

Figura 6-12: Exemplo do método padrão

6.3.6.
Método Open Close

O método Open Close se diferencia pela operação morfológica de abertura seguida da operação de fechamento na imagem *template*, ambas com um disco de raio *r* dado. Este método consegue agrupar bem as regiões sendo sensível a *r* escolhido. O diagrama de fluxo do método Open Close é apresentado na Figura 6-13. A Figura 6-14 apresenta um exemplo do método mostrando as regiões encontradas.

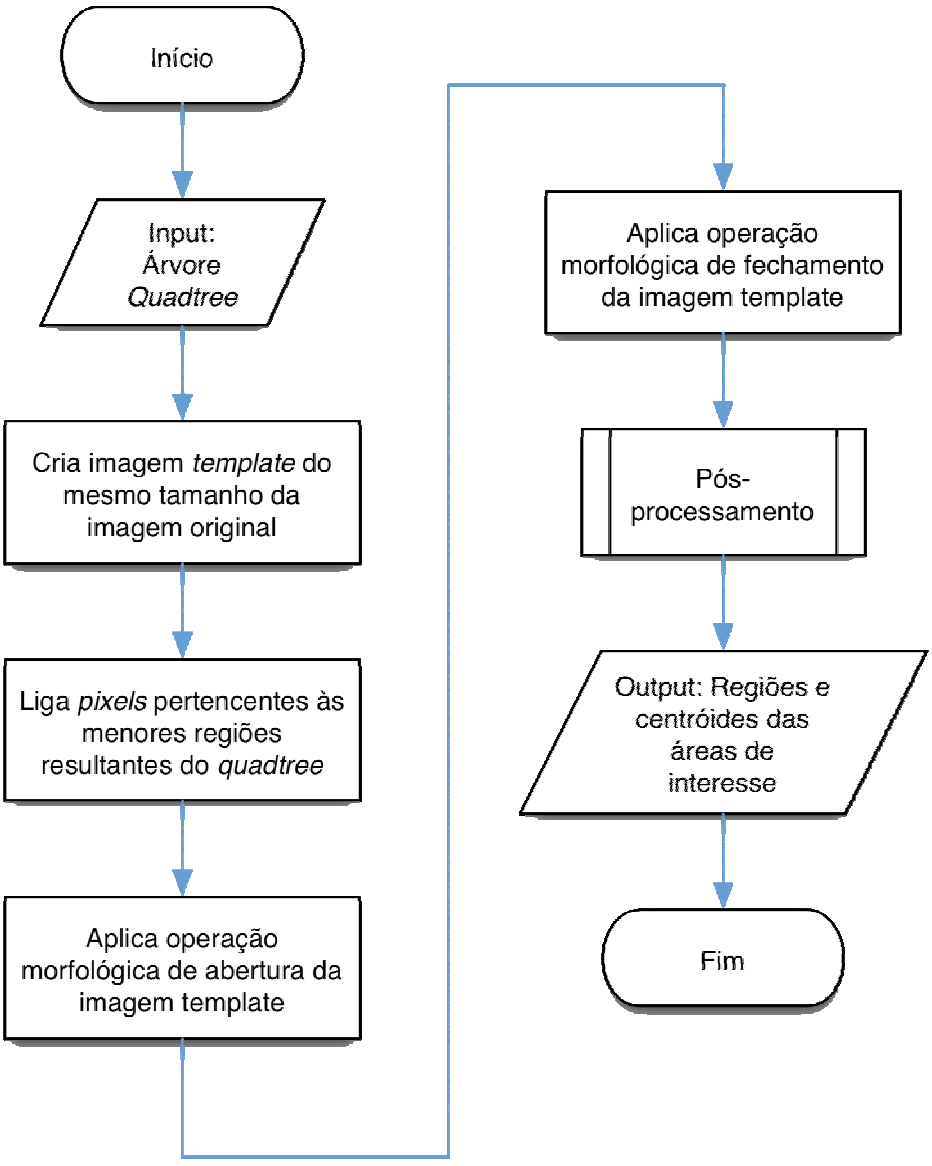

Figura 6-13: Método Open Close

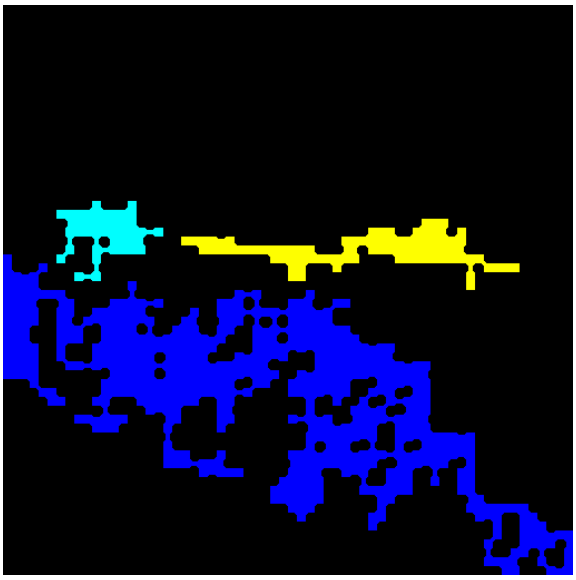

Figura 6-14: Exemplo do método Open Close

Outro método similar a este é proposto na seção seguinte.

6.3.7. Método Close Open

O método Close Open primeiro aplica a operação morfológica de fechamento e depois a de abertura. Como nos outros métodos, isso é feito com um disco de raio *r* dado. O diagrama de fluxo do método Open Close é apresentado na Figura 6-15. Como resultado são obtidas regiões que se assemelham a focos centrados em áreas de interesse como pode ser percebido pela Figura 6-16.

Este é o último método proposto para a segmentação usando *quadtree*. Após a escolha das regiões da panorâmica para as quais o robô deverá prosseguir, será feita a navegação para estes nós.

A comparação entre os três métodos propostos será apresentada com experimentos na seção 7.6.

A próxima seção descreve um algoritmo de *Visual Tracking* proposto para auxiliar as etapas de navegação para nós não explorados ainda e também para os nós já conhecidos.

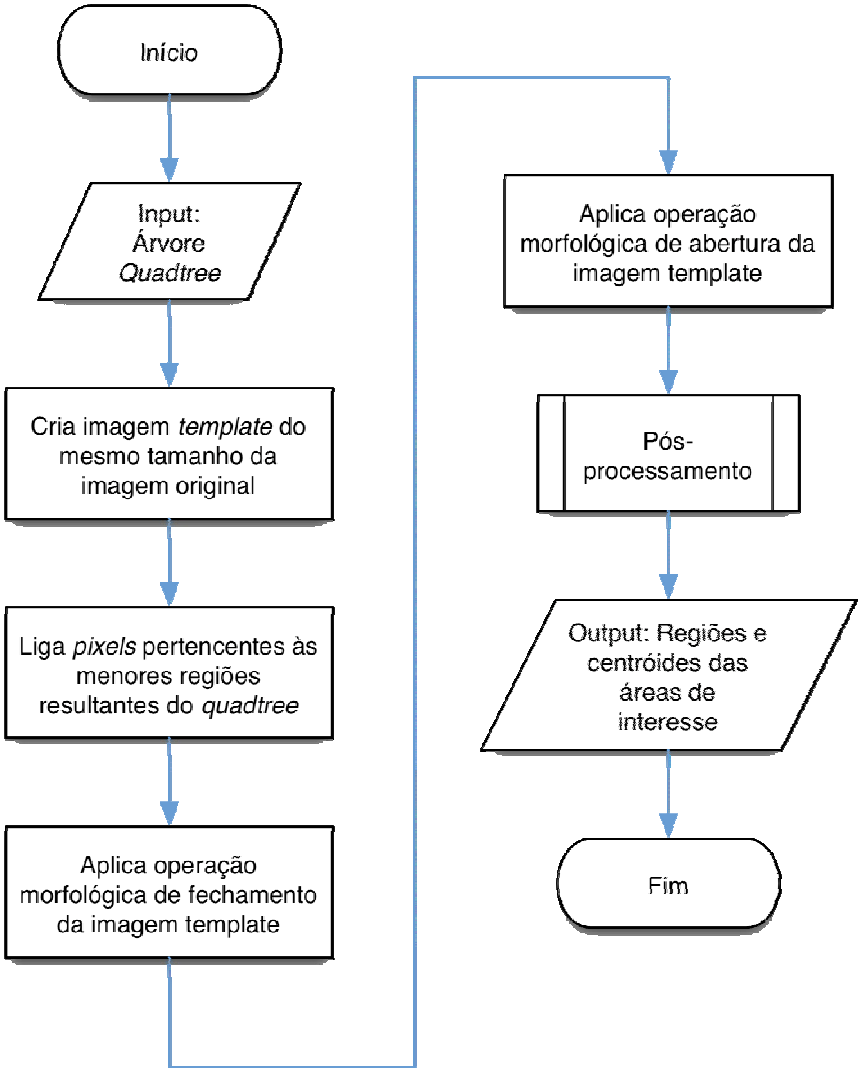

Figura 6-15: Método Close Open

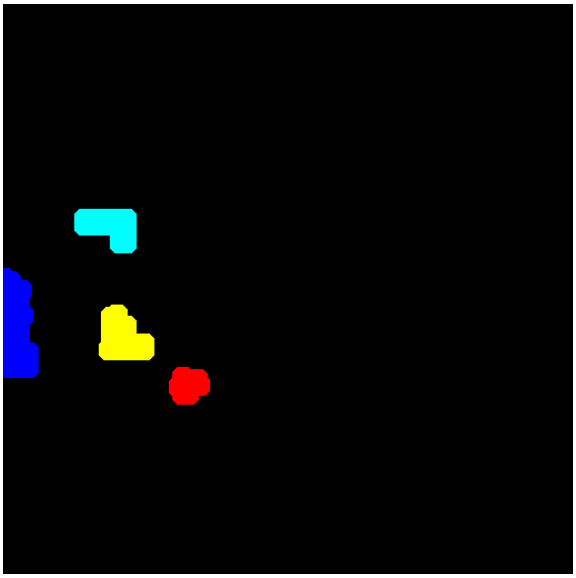

Figura 6-16: Exemplo do método close open

6.4. *Visual Tracking*: Acompanhamento de *Features* e Navegação

6.4.1. Introdução

Em diversos momentos, é interessante se fazer o acompanhamento de pontos na tela enquanto o robô está em movimento. A principal função deste acompanhamento está em permitir que o robô possa ajustar sua trajetória a partir das imagens observadas em tempo real (5 a 10 Hz nos experimentos realizados).

O objetivo é se acompanhar pontos que não se modifiquem muito ao longo da trajetória, possivelmente relativos ao fundo do ambiente observado. Perceba que pontos ao fundo são menos variáveis ao longo do tempo permitindo um melhor acompanhamento e a correção da direção do robô.

Algumas dificuldades em se fazer o acompanhamento de pontos na tela são:

- Podem ocorrer oclusões;
- A aparência de objetos e da cena sofre mudanças ao longo do tempo devido à iluminação ou mesmo mudanças geométricas relativas à câmera;
- Mudanças temporárias ou permanentes relativas a variações em 3D dos objetos ou cenas observados;

De modo a se lidar com mudanças relativas aos pontos observados, o procedimento deve se adaptar a pequenas mudanças. Um problema é que esta adaptação pode induzir a erros.

O procedimento também deve ser robusto a ponto de saber descartar pontos que não estão sendo acompanhados corretamente, ou que sofreram oclusão durante muito tempo, ou que não são mais encontrados com a certeza desejada.

Caso o número de pontos acompanhados caia muito, alguma estratégia mais robusta, como utilizando a técnica SIFT (descrita mais adiante), pode ser utilizada para se reencontrar as regiões que estão sendo acompanhadas. Isto implicaria em o robô parar de se movimentar para se reencontrar e depois prosseguir seu movimento.

De modo a se atender às necessidades apresentadas, optou-se por trabalhar com correlações de pequenas regiões na imagem para imagens subseqüentes, em

conjunto com algumas técnicas que serão apresentadas, de modo a se fazer o acompanhamento visual de pontos na tela quando necessário. Esta estratégia é similar à utilizada para a correspondência de pontos entre as imagens de uma panorâmica.

Como será visto, o algoritmo SIFT foi escolhido para se reencontrar as regiões procuradas caso fosse preciso.

O problema a ser tratado será definido da seguinte maneira:

- Deseja-se fazer o acompanhamento de coordenadas na tela chamadas de pontos chave. Estas coordenadas são acompanhadas indiretamente a partir de pontos de referência;
- Pontos de referência são pontos com características de interesse (particulares a cada aplicação) que serão procurados em *frames* subseqüentes e podem ser descartados ao longo do tempo. Da mesma forma, novos pontos de referência podem ser escolhidos;

A escolha dos pontos chave e dos pontos de referência é particular a cada problema tratado. A estratégia descrita a seguir é geral para pontos chave e pontos de referência dados.

Os tópicos teóricos utilizados no método a ser descrito foram apresentados nos capítulos 3 e 4.

A seção seguinte descreve como foi feita a correspondência dos pontos a cada momento.

6.4.2. Correspondência de Múltiplos Pontos para Imagens em Movimento

Definem-se os pontos chave por PC_i, onde i é o índice de cada um destes pontos. Cada ponto chave guarda as seguintes informações:

- Coordenadas *(x,y)* no momento inicial do *tracking*;
- Coordenadas *(x,y)* no momento atual;

Definem-se os pontos de referência por PR_i, onde i é o índice de cada um destes pontos. Cada ponto de referência guarda as seguintes informações:

- Coordenadas *(x,y)* no momento inicial do tracking;
- Coordenadas *(x,y)* no momento atual;

- Modelo, ou *template*, que descreve cada ponto e será usado para fazer sua correspondência em cada imagem. Este modelo será adaptado ao longo do tempo. O modelo de cada ponto é dado por w_i;
- Modelo inicial;

Como previamente citado, o acompanhamento dos pontos de referência será feito através de correlação. Portanto, o modelo inicial de cada ponto é dado por uma janela de tamanho *(Sx1, Sy1)* ao redor de cada ponto obtida no primeiro momento do *Visual Tracking*.

O diagrama de fluxo do processo é apresentado na Figura 6-17.

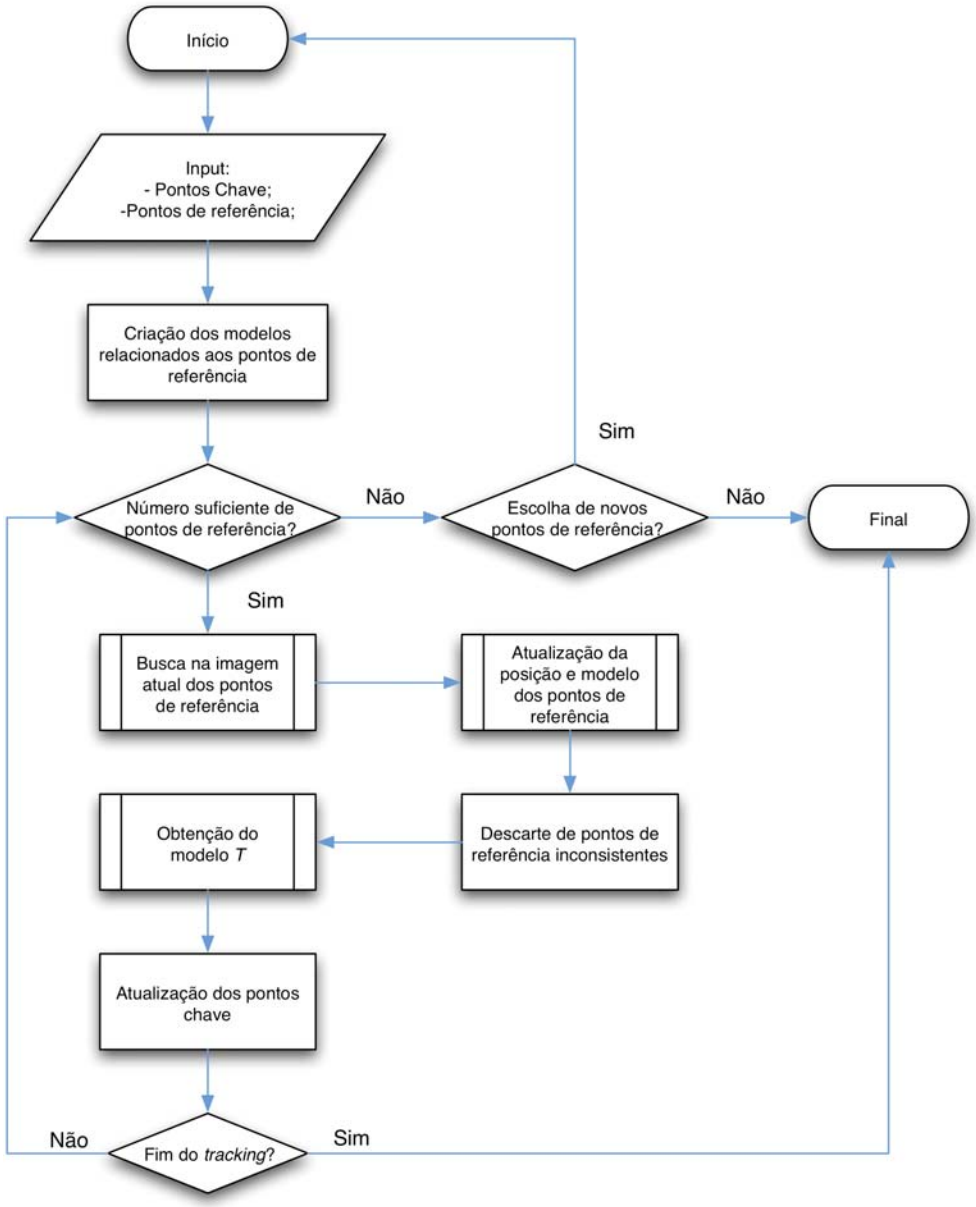

Figura 6-17: Diagrama do processo de *Visual Tracking*

A cada momento, as seguintes etapas são realizadas:

1. Busca dos pontos de referência: Procura dos pontos de referência na imagem atual através de correlação. Será descrito na seção 6.4.3;
2. Atualização dos pontos de referência: A posição e o modelo dos pontos pode ou não ser atualizada. Pontos ruins também são descartados. Esta etapa também será descrita na seção 6.4.3;
3. Obtenção do modelo: Escolha dos pontos que serão utilizados para se computar os pontos chave e cálculo da transformada afim que mapeia as posições iniciais dos pontos chaves para as posições atuais. Este passo está descrito na seção 6.4.4;
4. Caso não existam pontos de referência suficientes, o *tracking* é interrompido e novos pontos de referência devem ser escolhidos;

A seção seguinte descreve como é feita a busca dos pontos de referência a cada momento e como são atualizados estes pontos.

6.4.3. Busca e Atualização dos Pontos de Referência

A busca dos pontos de referência é feita para cada ponto buscando a posição de correlação máxima dentro de uma vizinhança de tamanho *(Sx2, Sy2)* ao redor da última posição de cada ponto. Veja a Figura 6-18. Caso a janela de busca extrapole o tamanho da imagem, o ponto de referência é descartado.

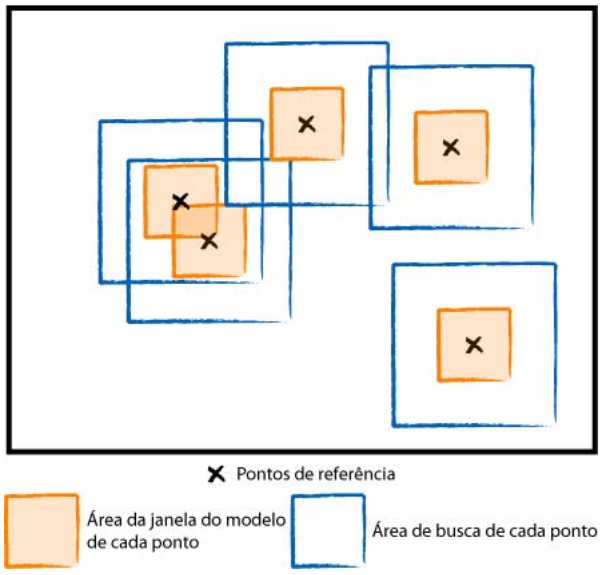

Figura 6-18: Busca dos pontos de referência

O valor de correlação máxima de um ponto PR_i será definido como $Cmax_i$.

A atualização de cada ponto deve seguir as seguintes regras:

- Pontos com $Cmax_i$ inferior a um limiar α escolhido não são atualizados;
- Define-se $Caux_i = \rho Cmax_i$ para $0 < \rho < 1$. Computa-se o percentual de pontos dentro da janela de busca que tiveram correlação superior a $Caux_i$. Caso este percentual seja superior a β dado, o ponto não é atualizado. Ou seja, $Caux_i$ é um limiar, de preferência próximo ao valor de correlação máxima, utilizado para se eliminar pontos que obtiveram alta correlação para uma alta percentagem de pontos dentro da janela de busca. Caso isto tenha acontecido, pode-se imaginar que este ponto não foi encontrado de modo bem definido e portanto é melhor não atualizá-lo;
- Caso a posição encontrada esteja fora da imagem, o ponto não é atualizado;

Pontos não atualizados não são imediatamente descartados, porém não são utilizados para se computar o modelo de transformação entre o momento inicial e o modelo atual.

Pontos que não forem atualizados um número k seguido de vezes são descartados da busca. Perceba que isto permite que oclusões temporárias não afetem o procedimento, ao mesmo tempo em que se consegue descartar pontos que não estão bem definidos ou que não são encontrados durante um tempo significativo.

Os pontos que são atualizados têm sua posição (x,y) recalculada a partir da posição de correlação máxima. Estes pontos também têm seu modelo atualizado da seguinte maneira:

- Utiliza-se uma janela $\hat{w}_i$ ao redor da localização atual do ponto de tamanho $(Sx1, Sy1)$;
- Utiliza-se esta janela para atualizar o modelo fazendo $w_i' = \mu w_i + (1-\mu)\hat{w}_i$, onde μ deve obedecer $0 < \mu < 1$, w' é o modelo atualizado e w é o modelo anterior. O valor de μ deve ser próximo de 1 para que o modelo não seja deteriorado rapidamente levando a falsas

correspondências do ponto chave. O objetivo de se atualizar o modelo é conseguir se adaptar a pequenas variações de luminosidade e pequenas variações de perspectiva;

Com os pontos de referência calculados, deve-se então obter o modelo de transformação *T* que leva estes pontos da imagem obtida em um momento anterior para um momento seguinte. Isto é visto na seção seguinte.

6.4.4. Obtenção do modelo *T*

Deseja-se encontrar o modelo *T* que mapeie as coordenadas dos pontos de referência de suas posições iniciais para as posições atuais. A matriz *T* e as técnicas utilizadas para tal foram apresentadas no capítulo 4.

Apesar de objetos no ambiente sofrerem variações geométricas independentes do fundo, considera-se que as variações geométricas do fundo do ambiente observado possam ser aproximadas por uma transformação afim, que leva em consideração mudanças de perspectiva. Desta maneira, é esperado que o método proposto consiga eliminar pontos de referência que não estejam associados ao fundo e que possam estar levando à construção de um modelo errôneo de *T*.

Propõe-se que o modelo de transformação *T* seja inicialmente construído a partir da minimização dos erros médios quadráticos de modo a se mapear as posições inicias dos pontos de referência para as posições atuais. A partir de então *T* pode ser refinado com a combinação de uma ou mais das seguintes técnicas já apresentadas no capítulo 4:

- Transformada de Hough;
- RANSAC;
- Reajuste da matriz de pesos W;

A escolha de quais técnicas são usadas deve ser feita de modo a contrabalançar a robustez do modelo encontrado com a eficiência de processamento. Não se pode esquecer que o procedimento está sendo realizado em tempo real (5 a 10 Hz nos experimentos realizados) e que o sistema experimental usado já implica em baixa eficiência computacional.

Os pontos de referência que forem considerados inconsistentes um número *k* seguido de vezes devem ser descartados da busca, conforme já mencionado. Estes pontos ou estão errados ou não condizem com objetos de fundo da cena, mas sim com algum objeto que estiver tendo sua posição relativa ao robô alterada de modo diferente dos objetos do fundo da cena.

O modelo *T* encontrado é utilizado para computar as coordenadas atuais dos pontos chaves.

O *Visual Tracking* é assim feito de modo a auxiliar a navegação. A navegação utilizando *Visual Tracking* será detalhada na seção a seguir.

6.4.5. Navegação usando *Visual Tracking*

O principal uso do acompanhamento de pontos chave e pontos de referência nesta dissertação é o de fazer a navegação do robô auxiliada por visão.

A seguir é apresentado como utilizar o *Visual Tracking* de modo a auxiliar a navegação do robô. Serão apresentados dois tópicos semelhantes: como se fazer a correção da direção do robô auxiliada por visão e como navegar para um ponto chave dado auxiliado por visão. Apesar de serem tarefas parecidas, é interessante fazer a explicação das mesmas separadamente.

6.4.5.1. Correção de Direção do Robô

O problema é definido da seguinte maneira:

- Deseja-se fazer com que o robô corrija sua direção auxiliado por pontos de referência e ponto chave dados;
- O ponto chave indica a direção que se deseja obter;
- Os pontos de referência são utilizados para se calcular a localização do ponto chave durante a correção de direção;

A tarefa consiste em se fazer o acompanhamento de pontos de referência na tela de modo a se determinar a cada momento a posição do ponto chave e conseqüente direção desejada do robô. Caso a direção desejada não corresponda à direção real do robô, então o robô deve se movimentar de modo a corrigir sua orientação.

Perceba que quanto maior o número de pontos de referência, maior a redundância associada à tarefa e, conseqüentemente, maior a robustez do método. Ao mesmo tempo, quanto maior o número de pontos tratados, maior o custo computacional.

Em linhas gerais, o procedimento é apresentado no diagrama de fluxo da Figura 6-19.

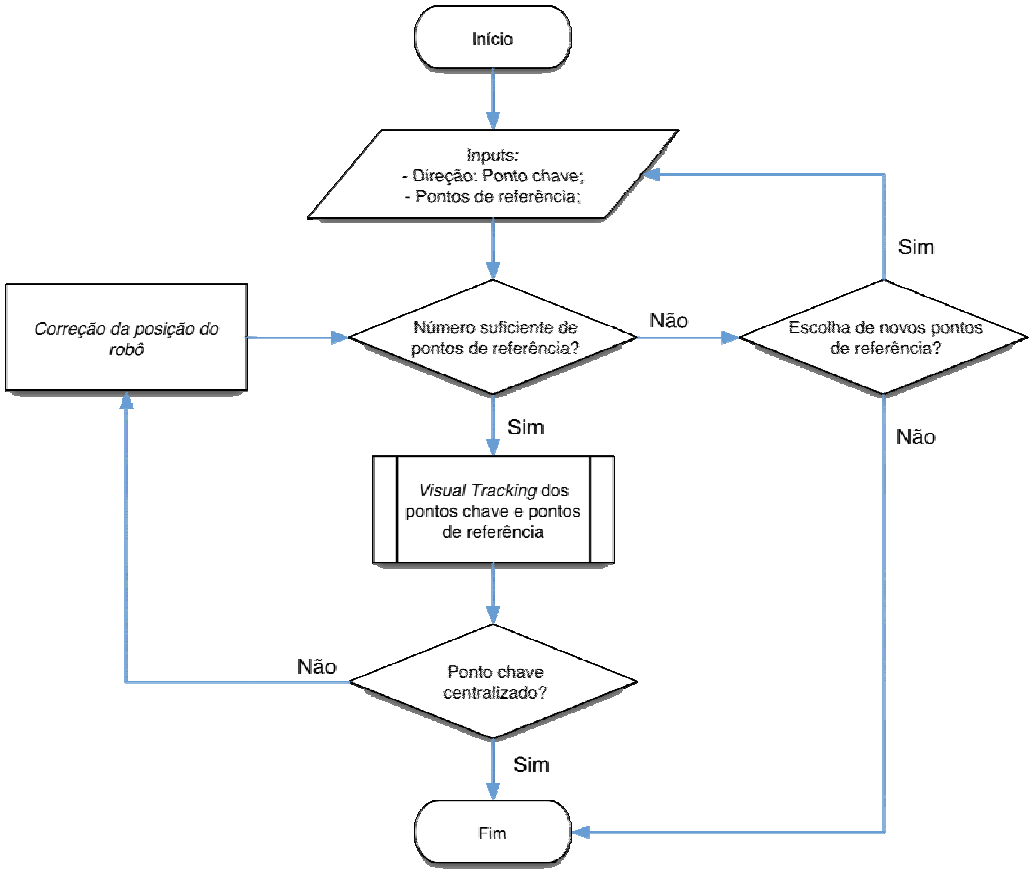

Figura 6-19: Correção da direção do robô

O procedimento de correção é realizado seguindo-se as seguintes etapas:

1. É realizado o *Visual Tracking* dos pontos de referência dados de modo a: encontrá-los na imagem atual; fazer a atualização dos mesmos; e computar a posição do ponto chave que indica a direção da imagem;
2. Caso a direção do robô já esteja alinhada com a direção desejada, o procedimento termina. Caso a direção do robô não corresponda à direção desejada, o robô deve se movimentar para esquerda ou

direita de modo a buscar centralizar o ponto chave. A posição do ponto de referência indica a direção desejada para a qual o robô deveria estar apontando. A correção de direção é feita quando o ponto chave se distancia um número η de *pixels* do centro da imagem (sugerido o valor de 5% da resolução horizontal para o sistema experimental utilizado). Esta distância representa o erro da direção do robô em relação à imagem observada;

3. Caso o número de pontos de referência caia abaixo de *m* dado, novos pontos de referência devem ser obtidos;
4. O procedimento recomeça a partir de 1;

Para o caso do sistema robótico utilizado, a correção da direção do robô é feita através de pequenas rotações (aproximadamente 0.5^o). Caso uma nova correção seja computada antes de que um movimento instruído ao robô tenha sido terminado, este movimento é interrompido e nova instrução é dada.

A rotação de aproximadamente 0.5^o é a menor possível para o robô. Esta rotação pode ser grande o suficiente para se ter alguma instabilidade na correção da direção caso o limiar η de erro estipulado seja muito pequeno.

Pontos de referência devem ser constantemente selecionados durante a tarefa. Caso o número de pontos de referência caia abaixo de um número *m* dado, novos pontos de referência podem ser escolhidos de acordo com a necessidade da aplicação.

A seção seguinte descreve o uso de *Visual Tracking* para a tarefa de navegar seguindo uma direção e não somente corrigir a direção.

6.4.5.2. Navegação para Ponto Chave Auxiliado por Visão

Outra tarefa comum de navegação pode ser definida da seguinte maneira:

- Deseja-se fazer com que o robô navegue para um lugar desejado auxiliado por pontos de referência e ponto chave dados;
- A direção ao longo da trajetória pode ser corrigida usando o ponto chave;
- Para auxiliar a tarefa de se obter a localização do ponto chave a cada momento, pontos de referência são utilizados;

- São dadas condições para a parada do robô;

O diagrama de fluxo da tarefa pode ser visto na Figura 6-20.

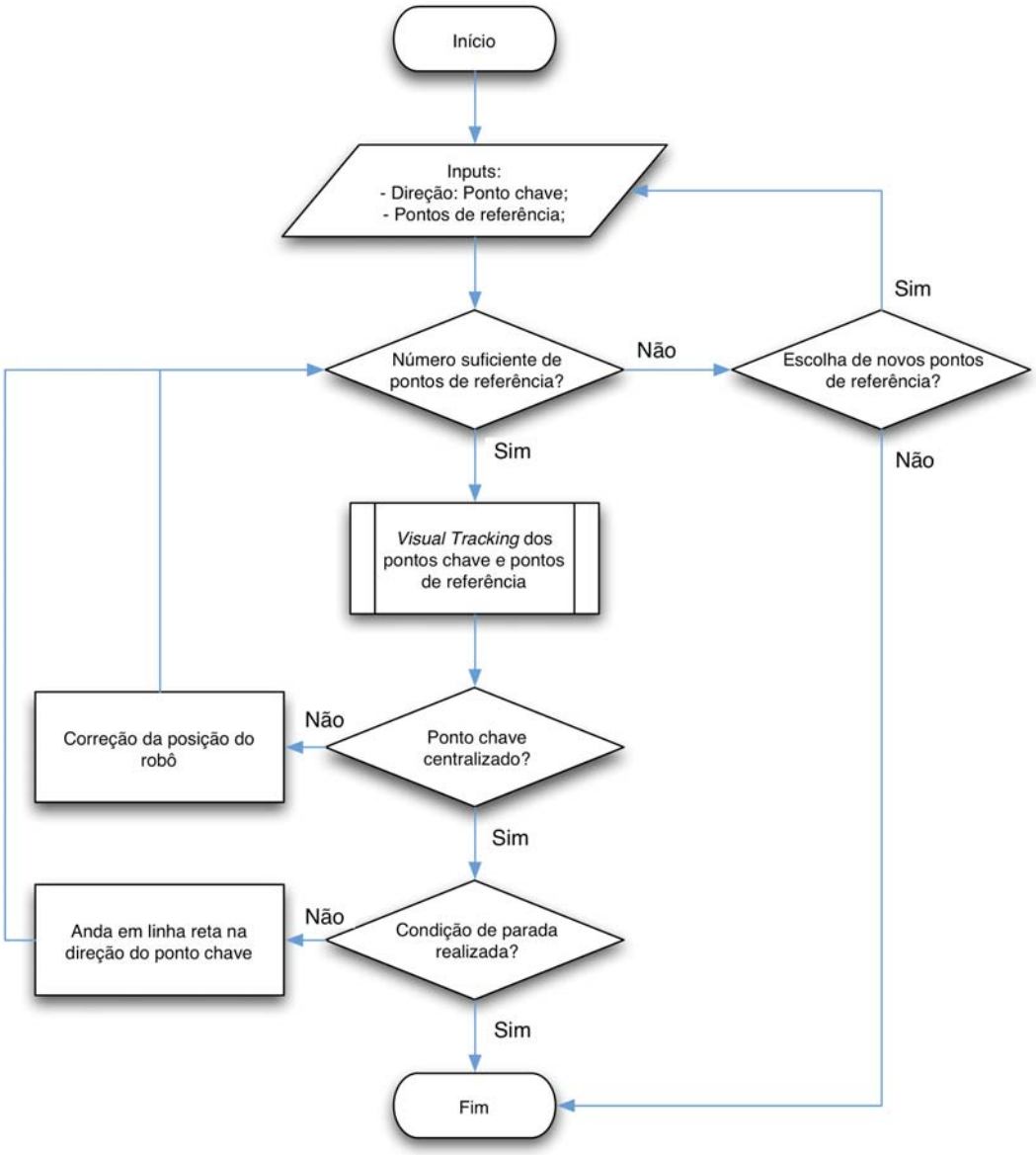

Figura 6-20: Navegação para ponto chave auxiliado por visão

Repare que a tarefa é bem semelhante à correção de direção apresentada, com a diferença de que neste caso, além do robô corrigir sua direção quando incorreta, ele também deve se movimentar de modo a se aproximar do ponto chave.

A cada momento se verifica se a direção está correta e caso não esteja é feita correção. Caso a direção esteja correta, o robô se move para frente até que uma ou várias condições de parada sejam encontradas.

Como na tarefa de correção de direção, caso o número de pontos de referência caia abaixo de um número *m* dado, novos pontos de referência podem ser escolhidos para que o procedimento continue.

A seção seguinte apresenta a componente 3 do algoritmo de exploração. Esta componente utiliza a navegação auxiliada por *Visual Tracking* aqui apresentada para fazer com que o robô seja guiado até um nó novo a ser explorado.

6.5. Componente 3 - Navegação para Nó Desconhecido

A navegação para um nó desconhecido consiste em, após ter sido feita a escolha das direções para o qual o robô irá navegar a partir de um nó atual, fazer com que ele siga seu rumo sem perder a direção escolhida e saber o momento de parar.

Os nós desconhecidos são encontrados a partir da segmentação (apresentada na seção 6.3) da imagem panorâmica em um nó conhecido. Esta segmentação gera regiões que foram denominadas de regiões de interesse. O centróide de cada região de interesse define a direção dos nós a serem explorados, denominados de nós desconhecidos.

A direção que o robô deve manter é dada portanto pelo centróide da região de interesse relativa ao nó em questão.

A navegação para um novo nó definido pelo centróide é feita através das etapas apresentadas na Figura 6-21.

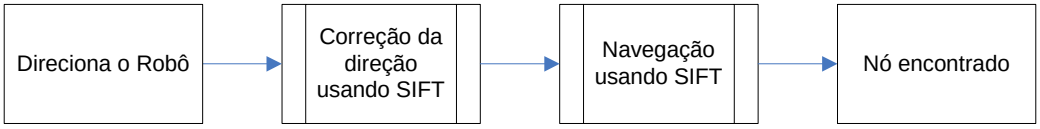

Figura 6-21: Navegação para nó desconhecido

Primeiramente deve-se direcionar o robô para o nó a ser explorado. O robô é, portanto, girado levando-se em conta os parâmetros dados pelos *encoders*. Como a resposta do sistema experimental adotado para os comandos dados possui baixa precisão, é interessante se refinar essa direção a partir das imagens obtidas pela câmera.

A correção da direção do robô é feita através da comparação dos descritores SIFT obtidos para a imagem observada e para imagem extraída da panorâmica referente à direção desejada. Este procedimento será descrito com maiores detalhes a seguir.

Com a direção do robô corrigida, deseja-se fazer com que ele navegue até o nó a ser explorado. Ao longo da sua trajetória, o robô deve acompanhar o centróide da região de interesse e fazer um controle visual de modo a corrigir possíveis desvios de direção. Este procedimento também será descrito com maiores detalhes a seguir.

Fica estabelecido que o robô chegou ao nó desconhecido caso o centróide suma da imagem, por cima ou por baixo, ou caso algum obstáculo seja detectado pelos sensores infra-vermelhos (*infra-red*, IR).

A próxima seção descreverá em maiores detalhes como é feita a correção de direção usando de descritores SIFT para se apontar o robô para o local a ser explorado.

6.5.1. Correção de Direção Usando a Transformação SIFT

Como a resposta do robô ER1 possui baixa precisão para comandos de movimento, é altamente recomendado que se corrija a direção do ER1 em relação à direção desejada antes de se navegar para um nó desconhecido.

Esta correção será feita com auxílio de *Visual Tracking* como apresentado na seção 0. Para tal, é preciso se estabelecer o ponto chave e os pontos de referência e optou-se por utilizar a técnica SIFT.

O diagrama de fluxo do processo está na Figura 6-22.

Como se deseja que o robô se dirija para uma região escolhida na imagem panorâmica do nó em que ele está, é extraída a imagem da panorâmica relativa ao ângulo θ em que está centralizado o centróide da região. Então, aplica-se a transformação SIFT na imagem extraída e na imagem atual da câmera do robô.

A partir dos descritores SIFT encontrados, é possível calcular o modelo T (verificar capítulos 3 e 4) que mapeia as coordenadas da imagem extraída da panorâmica para a imagem observada. Com T computado, encontra-se a posição

do centróide da região na imagem atual da câmera mapeando-se sua posição na imagem extraída para a imagem atual usando *T*.

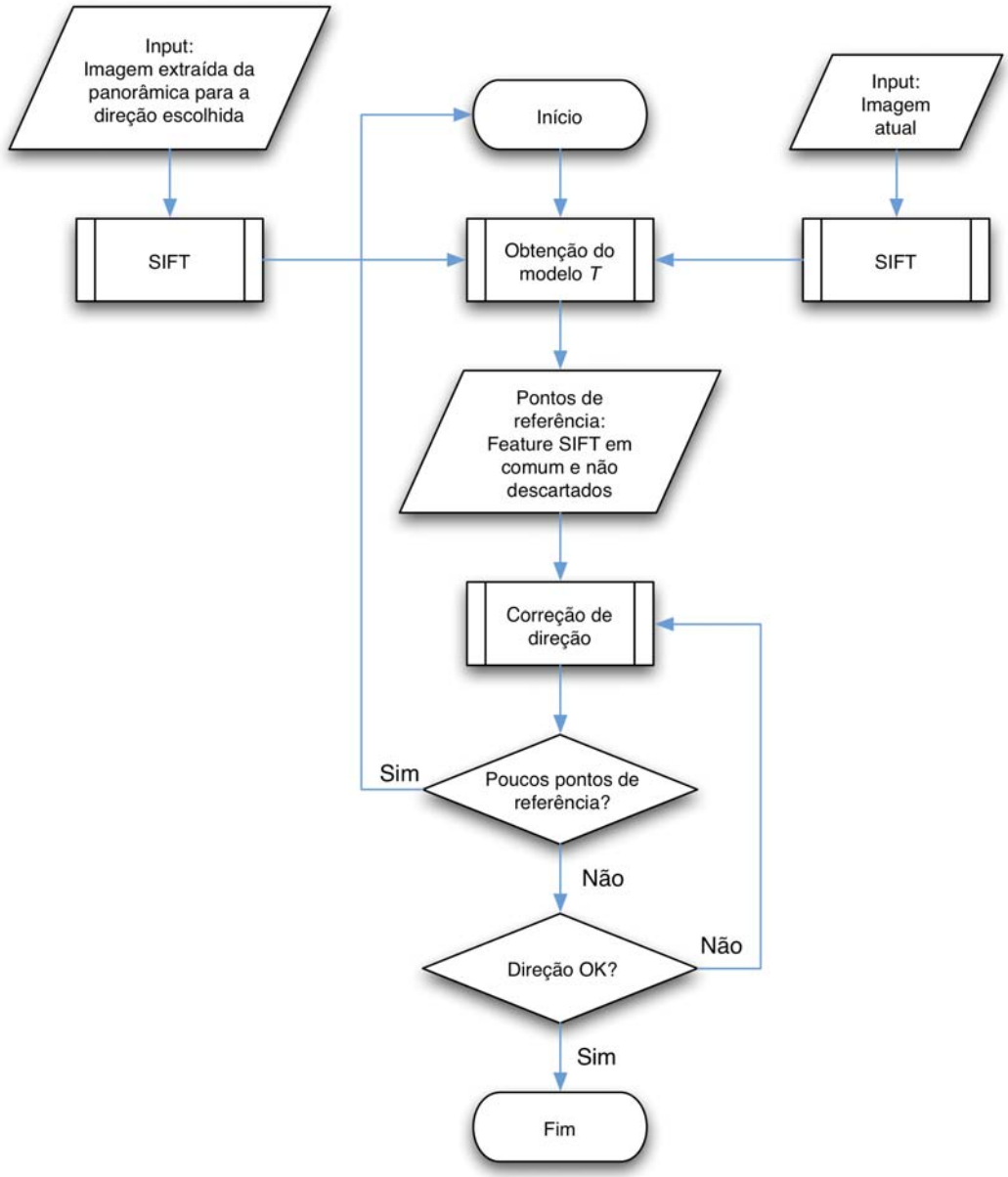

Figura 6-22: Correção da direção usando a transformação SIFT

O ponto da imagem extraída da panorâmica referente à posição do centróide será usado como ponto chave.

Aqueles pontos da imagem atual que foram correlacionados através do método SIFT na imagem extraída da panorâmica serão usados como pontos de referência.

A partir de então pode-se fazer a correção da direção com o procedimento apresentado na seção 0.

Caso ao longo da correção o número de pontos de referência caia abaixo de um número *m* dado, o algoritmo é recomeçado, encontrando-se novamente ponto chave e pontos de referência usando a transformação SIFT.

Após a correção de direção do robô, este deve seguir para o nó como visto na seção a seguir.

6.5.2. Seguindo para o Nó Desconhecido

A Figura 6-23 apresenta o fluxograma do procedimento feito para se navegar para um nó desconhecido.

Após o robô já ter sido direcionado corretamente, deseja-se seguir a trajetória até o nó desconhecido. Isto será feito com procedimento semelhante ao aplicado na correção da direção:

- Aplica-se a transformação SIFT na imagem extraída da panorâmica para o ângulo θ dado pela direção escolhida. A imagem é extraída da panorâmica como proposto na seção 6.2.5;
- Aplica-se a transformação SIFT na imagem observada;
- Encontram-se os pontos correlacionados e computa-se o modelo que leva da imagem panorâmica para a imagem observada;
- Para o centróide da região escolhida na imagem panorâmica, calcula-se sua posição na imagem observada usando *T*. Este será o ponto chave;
- Os descritores SIFT obtidos na imagem observada encontrados em comum com a imagem extraída da panorâmica são usados como pontos de referência para se fazer o *Visual Tracking* junto à navegação;

A partir de então o robô procede navegando para o ponto chave com o procedimento apresentado na seção 0. Enquanto não é obedecida nenhuma condição de parada, o robô prossegue. As condições de parada do robô são:

- É encontrado algum obstáculo, ou seja, os sensores IR detectaram algum obstáculo;

- O ponto chave está acima de um limiar superior ou abaixo de um limiar inferior da imagem;
- O número de pontos de referência está abaixo de *m* dado;

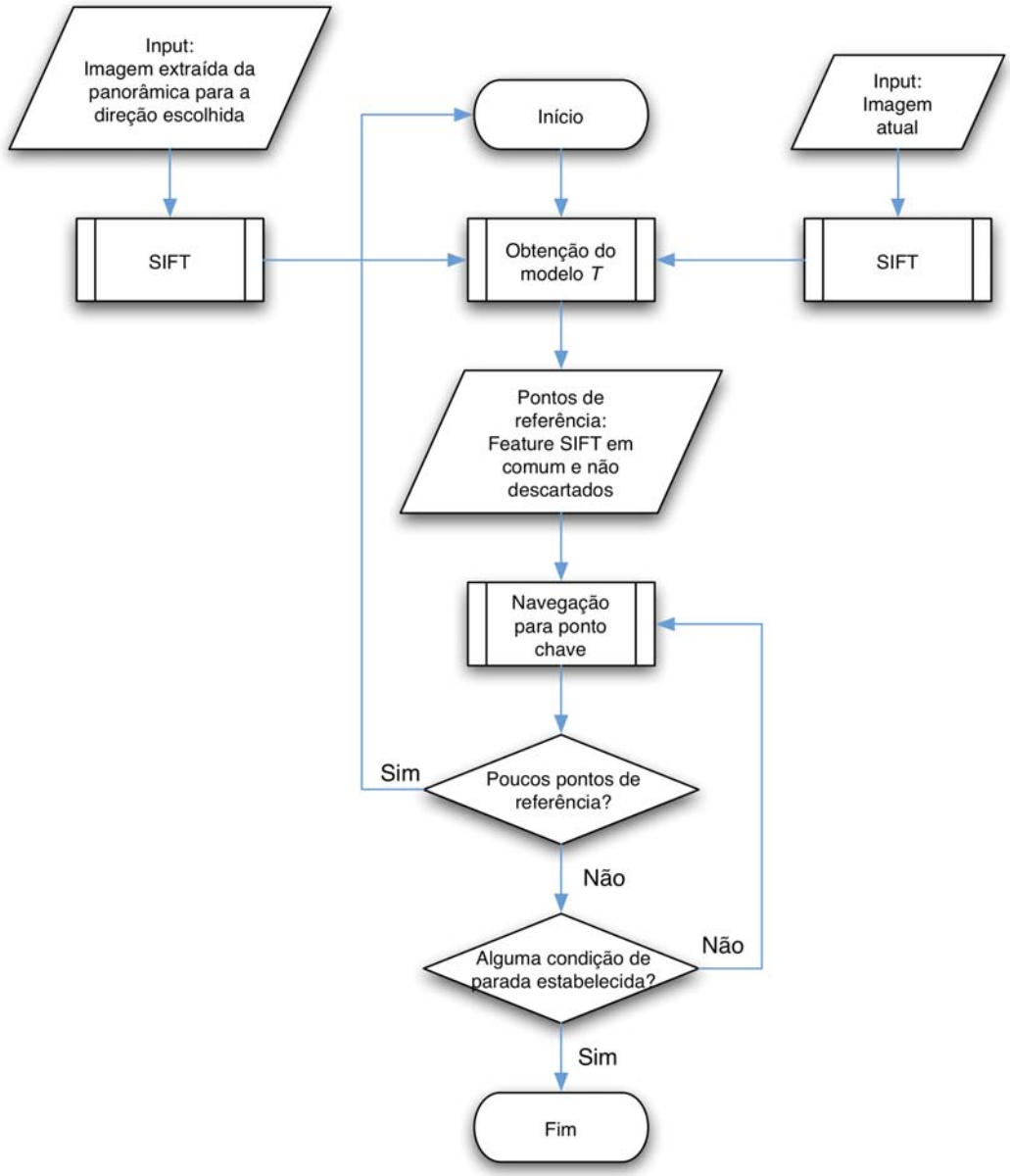

Figura 6-23: Navegando para nó desconhecido usando a transformação SIFT

Para o caso de a condição de parada ser relativa ao baixo número de pontos de referência, pode-se aplicar a transformação SIFT na imagem atual e estabelecer novos pontos de referência comparando-se estes pontos com a transformação SIFT da imagem inicial. O ponto chave pode ser recalculado a partir destes pontos

de referência e do modelo *T* calculado. Então, o robô pode continuar seguindo até chegar ao nó desconhecido.

De modo a retornar para os nós já conhecidos, os procedimentos propostos são um tanto diferentes dos apresentados para se navegar para locais desconhecidos. A próxima seção descreve as duas alternativas propostas e estudadas.

6.6. Componente 4 - Navegação para Nó Conhecido

Para se navegar para um nó conhecido, pode-se utilizar a imagem panorâmica do nó conhecido como referência de navegação. A partir da imagem panorâmica, consegue-se extrair uma imagem que representa a visão do robô quando este chegar ao nó conhecido a partir de um nó atual.

A partir de então, duas abordagens diferentes foram dadas ao problema.

A primeira abordagem implementada lidava com a comparação da visão do robô com a imagem de referência utilizando-se transformações invariáveis. Estas comparações eram feitas constantemente e utilizadas para se corrigir a trajetória do robô. A seção 6.6.3 descreve esse procedimento.

A segunda abordagem tomada é parecida com a navegação para um nó não conhecido. A imagem de referência é procurada na visão do robô utilizando-se do método *SIFT* e a partir de então faz-se o acompanhamento dos pontos gerados pela comparação *SIFT* através de procedimento de *Visual Tracking*. A seção 6.6.2 descreve esse procedimento.

A seção seguinte descreve como é obtida a imagem de referência que é utilizada nas duas abordagens implementadas do problema.

6.6.1. Obtendo a Imagem de Referência

É possível se obter da panorâmica do nó pai uma imagem que corresponda à visão que o robô terá ao completar sua navegação para o nó pai. Veja a Figura 6-24. Esta figura apresenta a posição do nó filho, a posição do nó pai, o caminho que deve ser feito de um ao outro e uma representação da imagem panorâmica feita no nó pai. Repare que sabendo-se a direção do nó filho para o nó pai, pode-se

extrair uma janela da panorâmica relativa à visão do nó pai na direção dada. A extração desta imagem pode ser feita como descrito na seção 6.2.5.

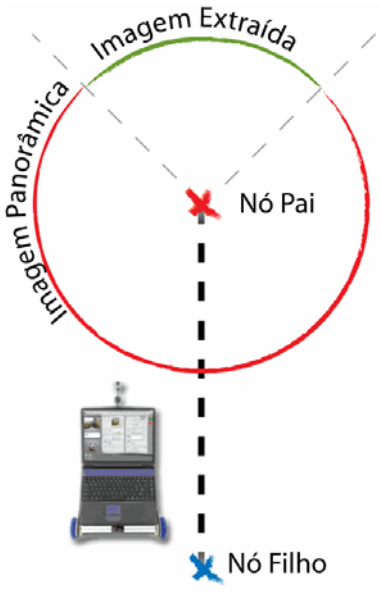

Figura 6-24: Obtendo a imagem de referência

Com auxílio da imagem de referência são propostas duas alternativas de navegação. A primeira estratégia é apresentada na seção seguinte e utiliza *Visual Tracking.*

6.6.2. Navegação Utilizando-se de a técnica SIFT e *Visual Tracking*

A navegação para um nó conhecido utilizando-se o método SIFT é semelhante àquela realizada para nós desconhecidos. A principal diferença está na imagem utilizada para se fazer a busca com a técnica SIFT. Outra diferença está em não ser necessário fazer uma correção de ângulo inicial, pois esta será feita automaticamente ao longo do processo. Além disso, as condições de parada não são as mesmas.

Para se fazer a navegação para um nó desconhecido, uma imagem referente à área de interesse da exploração era extraída da panorâmica do robô no nó atual. Por outro lado, aqui se usa a imagem de referência obtida no nó pai pelo processo descrito na seção anterior.

A partir de então o processo segue de modo parecido. O procedimento como um todo pode ser entendido através das seguinte etapas:

- Na imagem extraída da panorâmica é aplicada a transformação SIFT;
- Na imagem observada é aplicada a transformação SIFT;
- Os pontos em comum entre as duas imagens são encontrados e, então, é calculado o modelo que leva da imagem de referência para a imagem observada;
- Usando *T*, é calculada a posição na imagem observada do centro da imagem de referência. O ponto da imagem com esta posição será o ponto chave;
- Os pontos de referência do *Visual Tracking* são dados pelos descritores SIFT obtidos na imagem observada encontrados em comum com os descritores da imagem de referência;

Caso o número de pontos de referência caia abaixo de *m* dado, novos pontos de referência e ponto chave devem ser recalculados e o procedimento de *Visual Tracking* deve ser retomado.

Outra grande diferença está nas condições de parada do robô que não são as mesmas da navegação para um nó desconhecido. No presente caso, espera-se parar o robô no momento em que a imagem observada corresponder à imagem vista. Algumas estratégias foram propostas:

- Determinar a condição de parada a partir do acompanhamento do valor de correlação da imagem vista com a imagem de referência;
- Determinar a condição de parada a partir do acompanhamento do valor de comparação de alguma transformação invariável da imagem vista com a imagem de referência;
- Fazer o acompanhamento do enquadramento da imagem de referência na imagem vista. Quando se encontram pontos em comum entre as imagens, passa a ser possível descobrir qual o enquadramento da imagem de referência na visão do robô. Pode-se então acompanhar este enquadramento ao longo da navegação;

Para as opções em que se faz a comparação da imagem vista com a de referência, se espera que os valores de comparação cresçam até o momento em que o robô alcança na posição desejada, a partir de então estes valores

começariam a cair. Neste momento o robô pararia e corrigiria sua posição para que volte a de maior valor de comparação.

Na verdade existem algumas possibilidades para se parar o robô utilizando comparações. Uma delas é que quando os valores de comparação chegarem a um limiar dado, o robô pode parar o movimento. Outra delas é observar o momento em que o valor de comparação começa a subir e proceder como já dito.

Para o caso de se acompanhar o enquadramento, isto é feito da seguinte maneira:

- A posição dos pontos relativos às extremidades da imagem de referência pode ser calculada na visão do robô através do modelo *T* que é computado a cada momento;
- Ao longo da navegação, estes pontos tendem a se aproximar das extremidades da visão observada pelo robô.
- Após o robô passar da posição desejada, estes pontos começam a se afastar das extremidades.

A cada momento se calcula a distância euclidiana média entre estes pontos acompanhados e as extremidades da tela. O valor computado é um valor de erro que deve ser minimizado. Como no caso das comparações diretas entre imagens, pode-se ou estabelecer um limiar para este erro, ou acompanhar este erro e verificar o momento em que este começa a crescer novamente.

Outra possibilidade de navegação experimentada é apresentada na seção seguinte.

6.6.3. Navegação Utilizando-se de Comparação de Transformações invariáveis

A navegação para um nó conhecido usando transformações invariáveis tem como base a comparação da imagem de referência com a imagem vista pelo robô utilizando-se da técnica de *Window Growing* apresentada no capítulo 2.

A idéia aqui apresentada admite que a imagem de referência pode ser encontrada como uma sub-janela da imagem atual, ou seja, que a imagem de referência seria uma versão em escala de uma janela centralizada na imagem atual. Portanto, ao se aplicar *Window Growing* pode-se encontrar a sub-janela da

imagem atual correspondente à imagem de referência. Acompanhando o valor de comparação desta sub-janela com a imagem de referência, passa a ser possível determinar se é necessária uma correção na direção do robô e encontrar a direção correta.

A Figura 6-25 apresenta o robô em uma posição inicial e o mesmo em uma posição desejada. Esta figura busca exemplificar a idéia de que a visão do robô em um nó pai pode ser vista em um nó filho como uma sub-imagem do robô no mesmo.

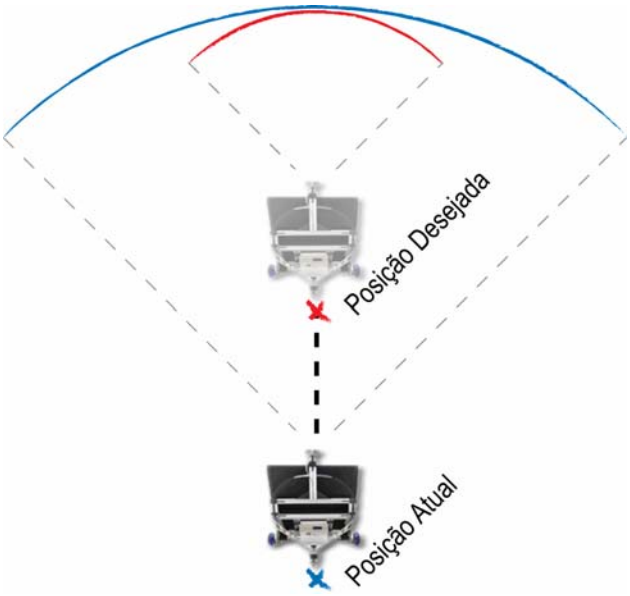

Figura 6-25: Navegação para um nó conhecido

A Figura 6-26 traz outro exemplo em que visões do robô em diferentes distâncias podem ser entendidas de modo bruto como versões em escala uma da outra. No próprio exemplo apresentado, pode-se perceber que esta concepção não é robusta mas sim uma aproximação da realidade.

A partir da idéia apresentada a navegação pode ser compreendida como apresentado no diagrama de fluxo da Figura 6-27.

O procedimento é feito através das seguintes etapas:

- Faz-se um procedimento de correção de ângulos que será apresentado adiante;
- A cada momento se compara a imagem atual com a imagem de referência usando *Window Growing*;

- Através da análise do valor das comparações ao longo do tempo, decide-se a necessidade de fazer nova correção de ângulo;
- Quando determinadas condições de paradas forem estabelecidas, a navegação é dada por terminada;
- Ao terminar a navegação, o robô corrige novamente o ângulo e anda para trás até a posição de melhor valor de comparação;

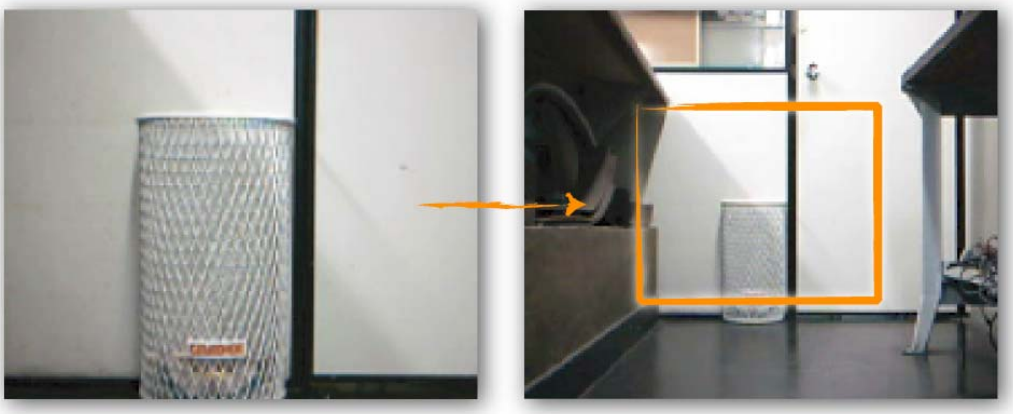

Figura 6-26: Encontrando imagens para diferentes distâncias

A decisão de se fazer uma correção de ângulo é feita verificando-se a cada momento se houve uma queda no valor de comparação. Espera-se que ao longo do tempo, o valor de comparação cresça. Porém, se este começar a decrescer, pode-se admitir que o robô esteja perdendo a direção. Sempre que o valor de comparação for inferior ao valor de um momento anterior dado um percentual escolhido, pode-se fazer uma correção de ângulo.

As condições de parada são as seguintes:

- A janela de comparação encontrada no *Window Growing* deve ser do mesmo tamanho que a imagem inteira;
- O valor de comparação está abaixo do melhor valor de comparação encontrado com relação a um percentual escolhido;

O valor de comparação pode representar o erro de comparação. Neste caso deseja-se encontrar o menor valor e portanto monitora-se a subida do erro;

A correção de direção é feita da seguinte maneira:

- Faz-se com que o robô se movimente um número n de θ graus em uma direção;

- Faz-se com que o robô se movimente um número $2n$ de θ graus na direção contrária. Para cada passo dado, uma imagem é obtida;
- O conjunto de $2n$ imagens obtidas é comparado com a imagem de referência utilizando-se *Window Growing*. Aquela imagem que possuir melhor valor de comparação deve representar a melhor direção;
- Por fim a direção é corrigida voltando à posição do robô para aquela onde se obteve a imagem de melhor comparação;

A Figura 6-28 ilustra como se fazer a correção de ângulo utilizando-se comparações de transformações invariáveis.

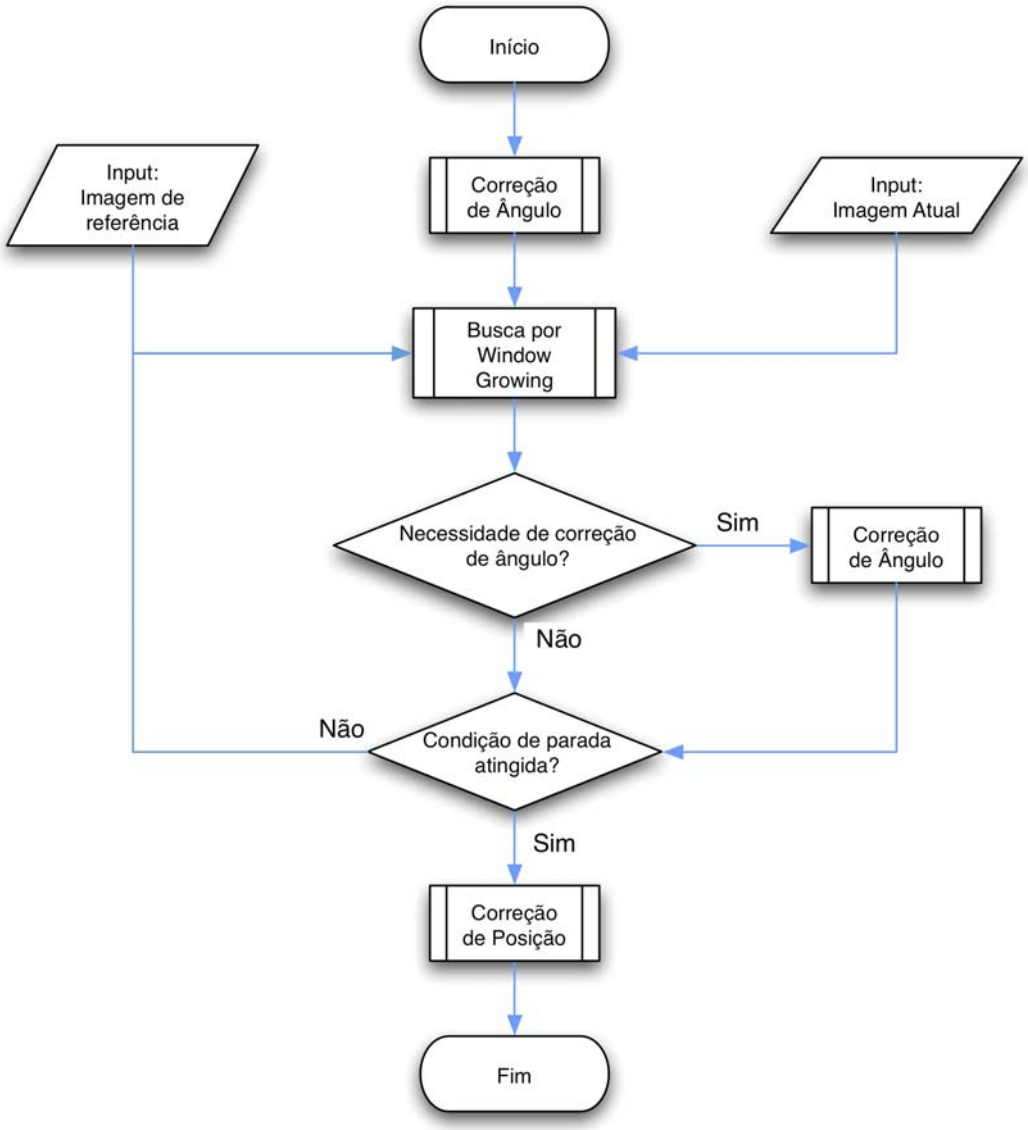

Figura 6-27: Navegação através de transformadas invariáveis

Figura 6-28: Correção de ângulo

Esta seção termina de apresentar as quatro principais componentes do algoritmo de exploração proposto. A próxima seção mostra uma quinta componente que pode ser utilizada para refinar o grafo que o algoritmo de exploração gera em uma segunda etapa de exploração. Esta componente só foi testada em simulações.

6.7. Componente 5 - Refinando o modelo criado através de algoritmos genéticos

O modelo criado pelo robô é uma árvore de nós. Cada adjacência carrega a informação de distância entre dois nós e o ângulo entre eles. Cada nó guarda a imagem panorâmica criada naquele ponto do espaço. Esta modelagem possibilitará uma futura navegação baseada em busca de imagens, seja de objetos, lugares, ou outros.

Em uma segunda etapa da modelagem, podemos melhorar o modelo criado. Para tal, buscam-se novas adjacências entre os nós encontrados. Desta maneira, quando houver navegação, o robô não necessitará percorrer muitos caminhos desnecessários para chegar à algum lugar escolhido.

Para criar novas adjacências, o robô precisará tentar percorrer o caminho entre os dois nós que a compõem. Em alguns casos, este caminho pode conter obstáculos. Como esta tarefa requer tempo, seria impensável tentar criar todas as adjacências possíveis, para todos os nós. Portanto, seria interessante escolher as adjacências a serem percorridas. Esta escolha seria feita de maneira a otimizar a futura navegação.

Este problema é abordado utilizando o método de Algoritmos Genéticos que não será detalhado aqui, podendo ser verificado em [48-50].

Aqui entra o problema de adicionar novas adjacências em um grafo que será descrito na próxima seção.

6.7.1. Descrição do problema de adicionar novas adjacências em um grafo

Dado o grafo $G = (X, A)$, composto do conjunto de nós X e o conjunto de adjacências A, deseja-se definir um conjunto A' de n' novas adjacências para acrescentá-lo a G, criando-se um novo grafo $G'(X,A+A')$. A escolha do conjunto A' deve ser feita de maneira a minimizar a soma das distâncias dos menores caminhos entre todos os nós:

$$F(x_1,x_2,...,x_n)=\frac{\sum_{i=1}^{n}\sum_{j=1}^{n}g(x_i,x_j)}{2} \tag{174}$$

Onde x_i representa o nó i de X, e $g(x_i,x_j)$ é uma função que retorna a distância do menor caminho entre os nós x_i e x_j.

A função $g(x,y)$ é calculada utilizando o algoritmo A^* [4, 51] que encontra o menor caminho entre dois nós de um grafo.

O conjunto de nós X foi definido neste projeto como pontos em um plano cartesiano com coordenadas (x,y), pois os grafos trabalhados representam um modelo de um ambiente em 2-D. A distância associada a uma adjacência é dada pela distância entre os dois nós que esta adjacência liga:

$$d(i,j)=\sqrt{(x_i-x_j)^2+(y_i-y_j)^2} \tag{175}$$

6.7.2. O Algoritmo A*

Para um grafo *G* qualquer, quando as adjacências em *G* possuem um custo e o custo de um caminho é definido como a soma dos custos relativos às adjacências que compõem o caminho, pode ser interessante definir o caminho de menor custo entre um nó inicial N_{init} e um nó objetivo N_{goal}. Um algoritmo clássico para gerar este caminho é o *A** [4, 51].

*O A** explora *G* iterativamente, percorrendo caminhos originados em N_{init}. No início de cada iteração, há nós que o algoritmo já visitou e outros ainda não visitados. Para cada nó *N* visitado, as iterações anteriores produziram um ou mais caminhos conectando N_{init} a *N*, porém o algoritmo só guarda a representação do caminho de menor custo. A qualquer momento, o conjunto de tais caminhos forma uma árvore *T* com o grupo de nós explorados. T é representada associando a cada nó visitado *N* (exceto N_{init}) um ponteiro ao nó parente.

*O A** associa uma função custo *f(N)* para cada nó *N* em *T* atual. Esta função é uma estimativa de custo do custo mínimo para o caminho entre N_{init} e N_{goal} através de *N*:

$$f(N) = g(N) + h(N) \tag{176}$$

Onde:

- *g(N)* é o custo do caminho entre N_{init} em *N* em *T* atual;

- *h(N)* é uma estimativa heurística do custo *h*(N)* do custo do menor caminho entre *N* e N_{goal};

A função *h* é dita admissível se e somente se satisfizer:

$$\forall N \in G : 0 \le h(N) \le h^*(N) \tag{177}$$

Para o problema em questão poderia se fazer *h(N)* igual à distância cartesiana entre *N* e N_{goal} , o que provavelmente faria o algoritmo rodar mais rápido, porém foi adotado *h(N) = 0*. Fica proposto o uso de *h(N)* igual à distância cartesiana.

O algoritmo *A** é feito como a seguir.

As entradas consistem em *G*, N_{init} , N_{goal} , *k*, e *h*, onde $k : X \times X \to R^+$ é a função que especifica a distância de cada adjacência em *G*. Todos os nós são inicialmente marcados como não visitados. O algoritmo faz uso de uma lista denominada *OPEN* que suporta as seguintes operações:

- *FIRST(OPEN)*: Remove o nó de *OPEN* com o menor valor de *f* e retorna o mesmo;

- *INSERT(N,OPEN)*: Insere o nó *N* em *OPEN*;

- *DELETE(N,OPEN)*: Remove o nó *N* de *OPEN*;

- *MEMBER(N,OPEN)*: Retorna *verdadeiro* se *N* estiver em *OPEN* e *falso* caso contrário;

-*EMPTY(OPEN)*: Retorna *verdadeiro* se *OPEN* estiver vazio e *falso* caso contrário;

6.7.2.1. Procedimento A*(G, Ninit , Ngoal , k, h)

inicio
    insira *Ninit* em *T*;
    INSERT(N_{init} , OPEN);
    marcar N_{init} como visitado;
    while ! EMPTY(OPEN) do
        N = FIRST(OPEN)
        if N = N_{goal} then sair do *loop while*;
        for todo nó *N'* adjacente a *N* em *G do*
            if N' não visitado *then*
                adicionar *N'* em *T* com ponteiro para *N*;
                f(N) = f(N) + k(N,N');
                INSERT(N',OPEN);
                marcar *N'* como visitado;
                fim;
            else if f(N') > g(N) + k(N,N') then
                f(N) = f(N) + k(N,N');
                modificar *T* redirecionando o ponteiro de *N'* para *N*
                fim;
            fim
        fim
    if N = N_{goal} then
        retorna o caminho construido traçando os ponteiros em *T* de N_{goal} a N_{inti};
    else retorna *falha*;
fim

6.7.3. Aplicando o Algoritmo Genético

Para encontrar *n* novas adjacências a serem adicionadas em um grafo, será empregada um Algoritmo Genético [48-50] com o objetivo de minimizar a soma das distâncias mínimas entre todos os nós após a adição das novas adjacências.

6.7.4. Representação do cromossomo

O cromossomo utilizado deve representar as novas adjacências a serem criadas. Algumas possíveis representações são:

- Máscara de bits: Cada gene do cromossomo está associado a uma adjacência do grafo e representa sua adição ao grafo. Se o bit estiver ligado, aquela adjacência é adicionada, se desligado, não. O tamanho do cromossomo deve ser de $k^2/2$, onde k é o número de nós do grafo. O espaço de busca é dado por $2^{\frac{k^2}{2}}$. Tenhamos como exemplo um cromossomo definido como (1-0-0-1). Neste exemplo há 4 possíveis adjacências, sendo que a primeira e a última serão adicionadas ao grafo;
- Representação por inteiros: Os genes são números inteiros representando os nós do grafo. Cada adjacência a ser adicionada ao grafo é representada por um par de genes. O tamanho do cromossomo deve ser de 2n, onde n é o número de nós a serem adicionados ao cromossomo. O espaço de busca é dado por $k^{2\pi}$. Tenhamos o cromossomo definido por ((1 - 2) , (5 - 8)). Neste exemplo, n = 2 e as adjacências entre os nós 1 e 2 e entre os nós 5 e 8 são adicionadas.

A segunda representação foi a escolhida para este projeto. Perceba que diferentes cromossomos podem representar a mesma solução, atrapalhando o desempenho do algoritmo.

6.7.5. Operadores utilizados

Somente dois operadores foram utilizados, *crossover* e mutação.

O operador de crossover utilizado foi o *crossover* de um ponto. Dado um ponto aleatório entre dois genes, os cromossomos pais são divididos e recombinados para gerar os cromossomos filhos. Outros operadores de *crossover* possíveis são o *crossover* de dois pontos e o *crossover* uniforme.

O operador de mutação utilizado, sorteava a troca do valor de um gene por um valor aleatório. É proposto, apesar de não utilizado, fazer uma mutação por adjacência. Esta mutação se daria da seguinte maneira. Sorteado um gene para ocorrer mutação, seu novo valor seria um nó sorteado dentre os nós vizinhos ao nó representado por tal gene.

6.7.6. Método de seleção

O método de seleção utilizado foi o método de seleção por roleta, com uso de normalização e *Steady-State* com duplicados.

6.7.7. Avaliação do Algoritmo Genético

Para avaliar os cromossomos gerados, as adjacências são adicionadas ao grafo *G* trabalhado, e então, este grafo é avaliado segundo a seguinte métrica denominada *Eficiência do Grafo*:

$$f(G) = \frac{D(x_1, x_2, ..., x_n)}{F(x_1, x_2, ..., x_n)} \tag{178}$$

Sendo a função *D*, somatório das distâncias cartesianas entre todos os nós. dada por:

$$D(x_1, x_2, ..., x_n) = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n} d(x_i, x_j)}{2} \tag{179}$$

A função *f* será máxima e igual a 1 quando ou todos os nós estiverem ligados, ou o caminho mínimo para todos os nós seja uma reta.

Caso algum dos nós não tenha adjacência com nenhum dos outros nós, a função de avaliação retorna 0. Isto permite que o Algoritmo Genético seja utilizado para gerar adjacências em grafos sem adjacências. Desta maneira o A.G. pode solucionar problemas como o de encontrar o melhor conjunto de rotas para um conjunto de localizações. Porém, apesar de possibilitar tal uso, esta avaliação não é a melhor para este tipo de problema, pois o Algoritmo Genético pode demorar muito tempo para encontrar soluções validas (que não retornem $f = 0$) sem conseguir encontrar bons schematas, dado que todos os cromossomos que representam soluções inválidas retornam o mesmo valor, sem diferenciação.

6.7.8.
O Programa desenvolvido

Para testar a solução proposta foi desenvolvido um programa em C++ utilizando o programa *Microsoft Visual C++ 6.0*. Este programa permite gerar os grafos testados e visualizá-los, verificar as métricas utilizadas, rodar o algoritmo *A** e rodar o Algoritmo Genético. Além disto, o programa gera curvas de desempenho do Algoritmo Genético e possibilita salvar os resultados para serem visualizados em outros programas, como o *Microsoft Excel* por exemplo. Imagens do programa podem ser vistas em Figura 6-29 e Figura 6-30.

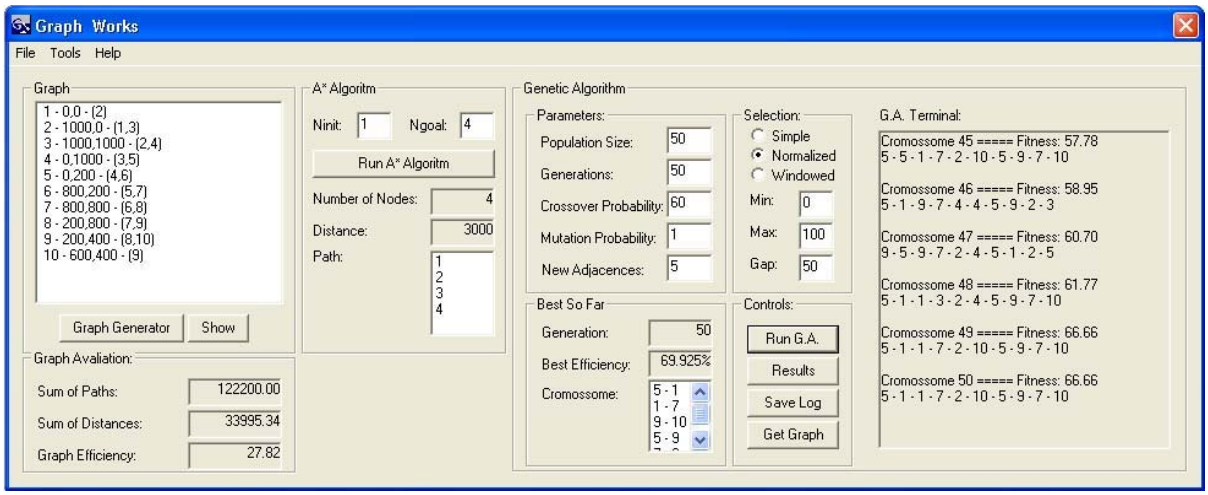

Figura 6-29: Interface do programa desenvolvido

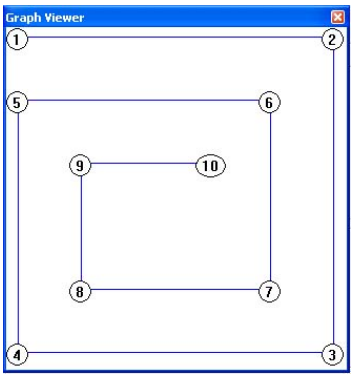

Figura 6-30: Interface de uma das ferramentas do programa desenvolvido

Este programa foi feito com as seguintes limitações:

- Grafos com o máximo de 100 nós;
- Espaço cartesiano limitado em $0 \le x \le 1000, 0 \le y \le 1000$;

7
Experimentos e Resultados

Neste capítulo serão apresentados alguns experimentos que ilustram as implementações desenvolvidas e propostas no capítulo anterior. São mostrados experimentos que investigam as componentes do algoritmo de exploração proposto. Entretanto, não foi feita a integração destes de modo a compor o algoritmo apresentado na seção 6.1.

Este capítulo está dividido nas seguintes seções:

- 7.1 - Transformações Invariáveis;
- 7.2 - Comparações;
- 7.3 - Navegação utilizando transformações invariáveis – Análise de *Window Growing*;
- 7.4 - SIFT;
- 7.5 - Encontrando pontos de controle para montar imagens panorâmicas;
- 7.6 – Segmentação da imagem por *Quadtree* baseada em entropia;
- 7.7 – Navegação para Nós Conhecidos e Desconhecidos utilizando-se de *Visual Tracking* e Transformação SIFT;
- 7.8 – Refinando o modelo utilizando Algoritmos Genéticos.

7.1.
Transformações Invariáveis

As transformações invariáveis tiveram sua comprovação matemática na seção 2. Entretanto, a aplicação destas transformações sobre funções no domínio discreto implicam em pequenas variações no resultado. De modo a se estudar a implementação discreta das Transformadas apresentadas e encontrar as particularidades desta implementação, diversos experimentos foram feitos.

As 3 imagens vistas na Figura 7-1 foram utilizadas para se experimentar o uso de transformações invariáveis.

Figura 7-1: Círculo, quadrado e triângulo

As 3 figuras usadas tinham resolução de 100 x 100 *pixels*.

As seguintes operações foram aplicadas nas imagens utilizadas:

- Escala: As imagens resultantes tem dimensão de 120 por 120 pixels;
- Rotação: Rotações de 20^{o};
- Translação: As imagens foram transladadas de 20 pixels no sentido vertical e 20 pixels no sentido horizontal;
- Distorção: As imagens resultantes tem dimensão de 120 por 100 pixels;
- Rotação e escala;
- Translação e escala;
- Translação e distorção;
- Rotação e distorção;
- Rotação e translação;
- Rotação, escala e translação;
- Rotação, distorção e translação;

As operações foram feitas em relação a posição superior e esquerda da figura porque todas as transformações utilizadas, com exceção da Transformada de Mellin do tipo 2, trabalham com este ponto como coordenada zero. Devido a este fato, é importante lembrar que as operações de rotação com o círculo afetam sua posição e não sua forma. Por isso os resultados relativos as rotações do círculo são semelhantes aos resultados relativos a translações.

As imagens após operações podem ser vistas nas seguintes figuras: Figura 7-2, Figura 7-3 e Figura 7-4. As interpolações feitas foram do tipo bicúbica.

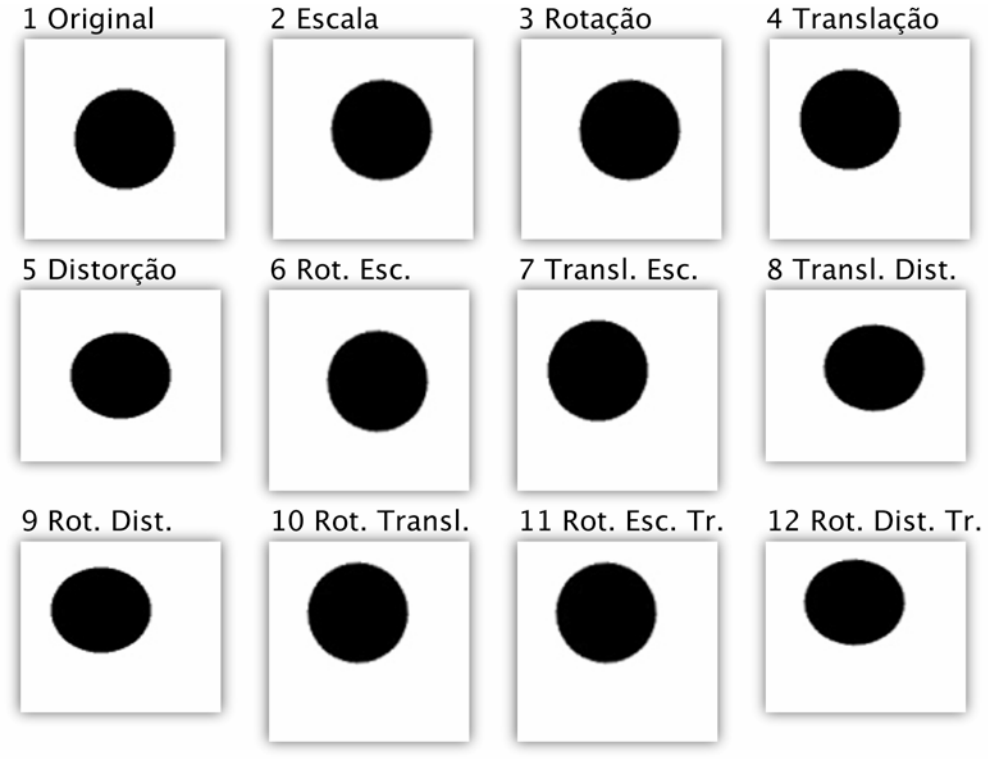

Figura 7-2: Círculos

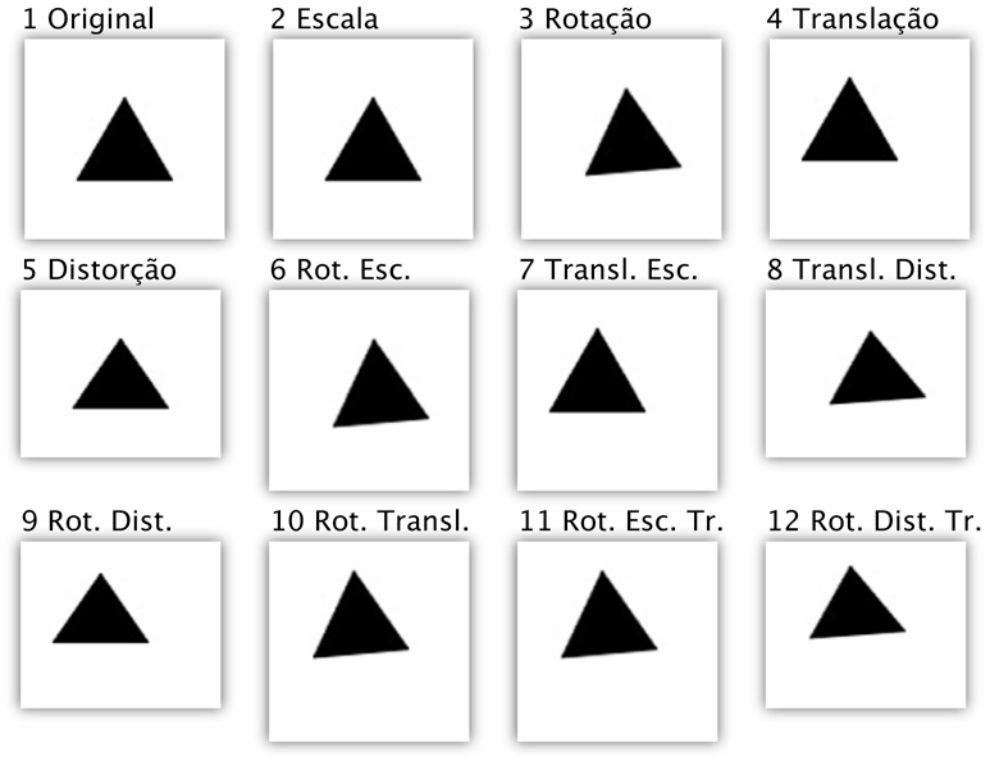

Figura 7-3: Triângulos

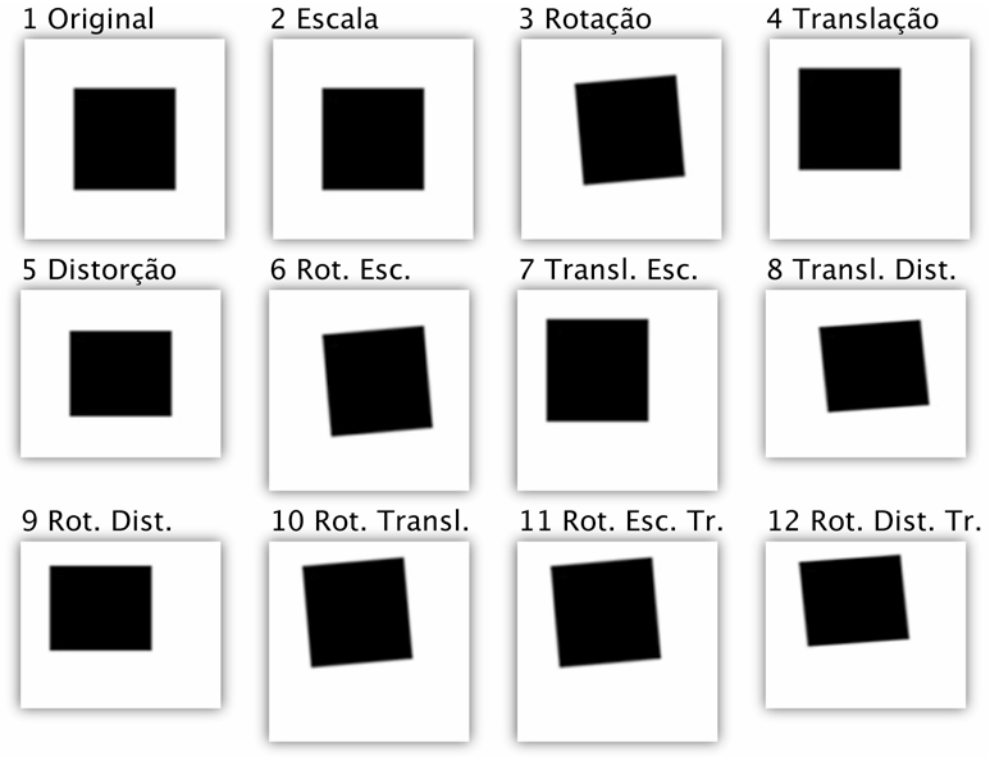

Figura 7-4: Quadrados

A Transformada de Mellin do tipo 2 utiliza o centro da imagem como eixo de coordenadas para realizar a operação *log-polar*. Portanto, todas as operações que apresentavam rotação foram refeitas com eixo centralizado para serem testadas com a Transformada de Mellin do tipo 2. Não foi preciso fazer nada em relação às operações de escala porque estas simplesmente alteravam a resolução da imagem, não sendo relativas nem ao canto superior esquerdo nem ao centro. Em Figura 7-5, Figura 7-6, Figura 7-7 são apresentadas as imagens com rotação centralizadas.

A Transformada de Fourier Mellin do tipo 2 utilizou as figuras já apresentadas por ser invariável à translação.

De modo a se verificar as invariâncias esperadas, as seguintes transformações foram experimentadas:

- Fourier;
- Mellin do tipo 1;
- Mellin do tipo 2;
- Fourier Mellin do tipo 1;
- Fourier Mellin do tipo 2;

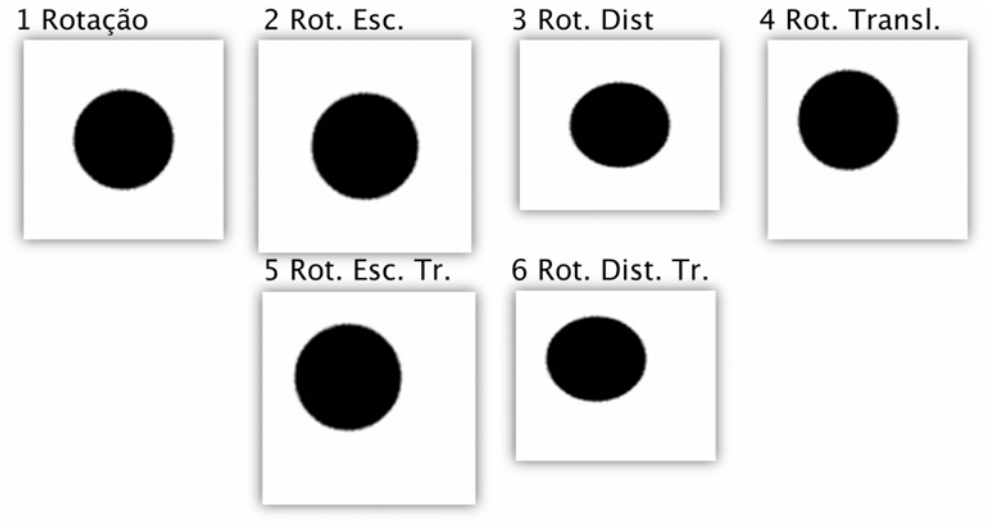

Figura 7-5: Círculos com rotações centralizadas

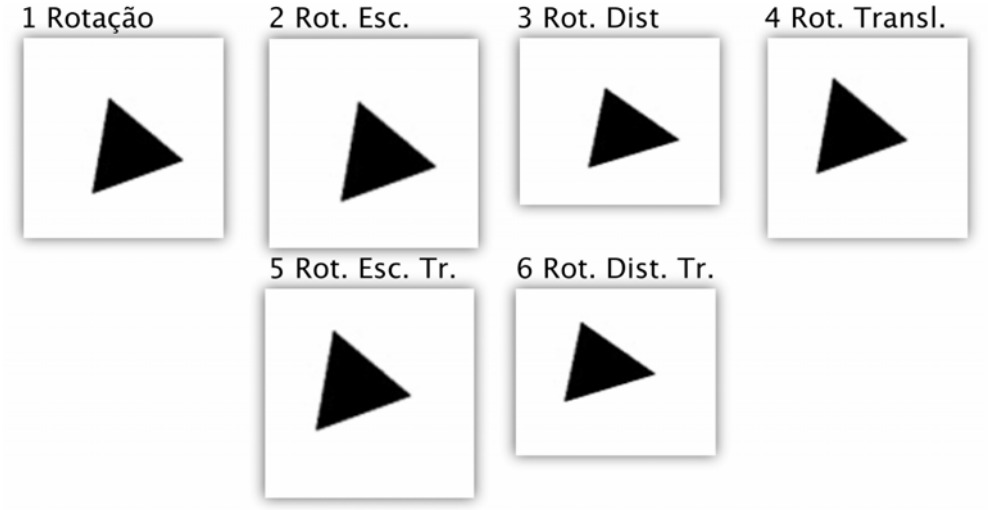

Figura 7-6: Triângulos com rotações centralizadas

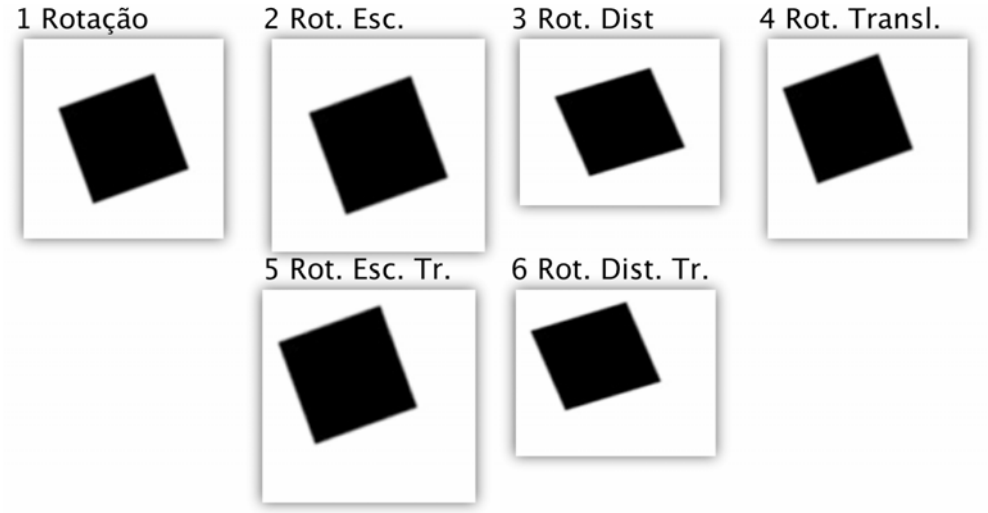

Figura 7-7: Quadrados com rotações centralizadas

As magnitudes das transformações são observadas nos experimentos a seguir. As transformações foram feitas com o programa *Matlab* e foram aplicados mapas de cor de modo a facilitar as visualizações.

7.1.1. Fourier

Da Figura 7-8 à Figura 7-10 são apresentados a magnitude da Transformada de Fourier para as imagens vistas da Figura 7-2 à Figura 7-4.

Pode-se verificar que as imagens referentes a magnitude da Transformada de Fourier são similares para os casos de translação observados. Também pode-se ver que a transformada acompanha as operações de escala, rotação e distorção:

- Imagens em diferentes resoluções implicam em Transformadas de Fourier de diferentes resoluções;
- Imagens rotacionadas implicam em Transformadas de Fourier rotacionadas;

Estas propriedades, já verificadas matematicamente para Transformadas contínuas (seção 2), puderam ser observadas através da implementação sobre as imagens discretas.

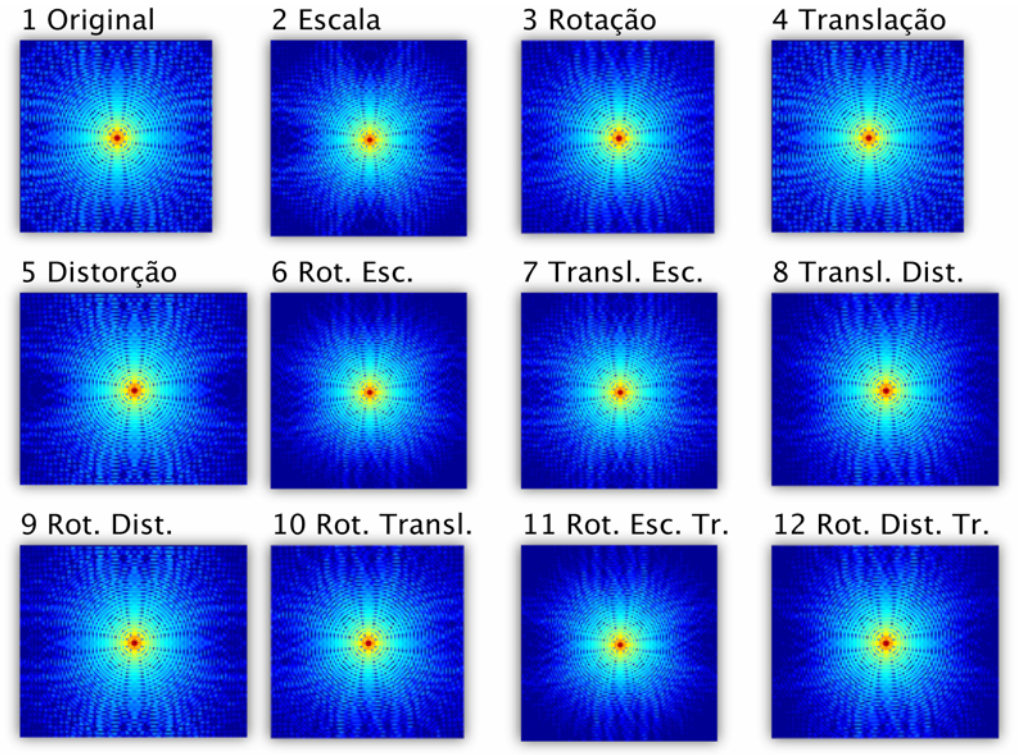

Figura 7-8: Transformadas de Fourier para círculos

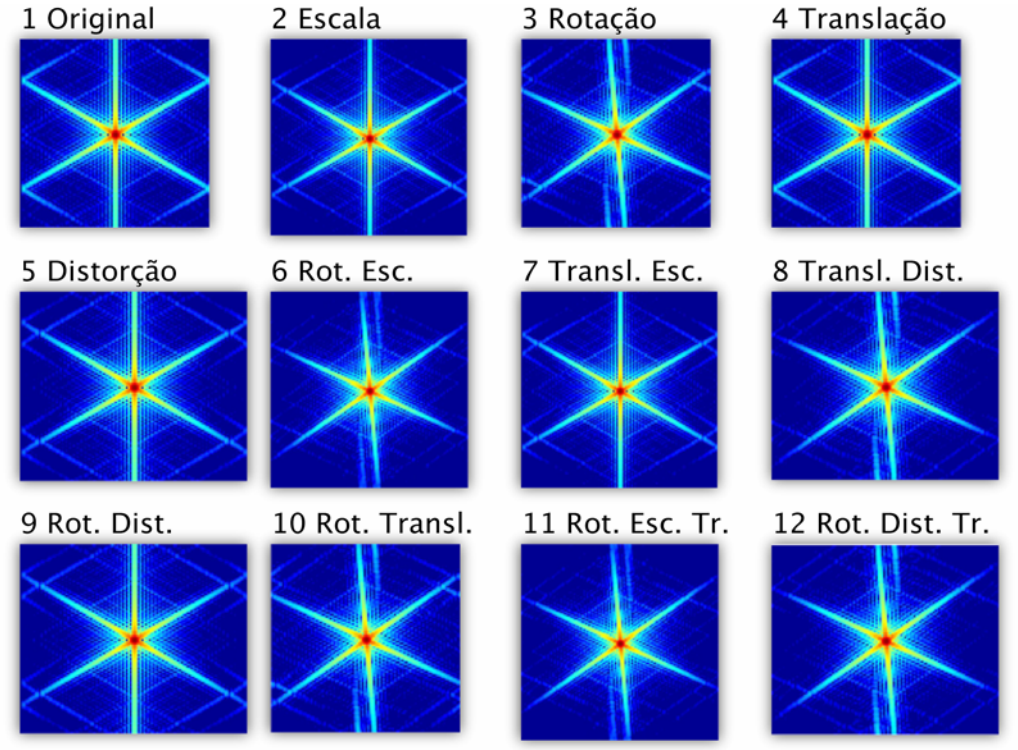

Figura 7-9: Transformadas de Fourier para triângulos

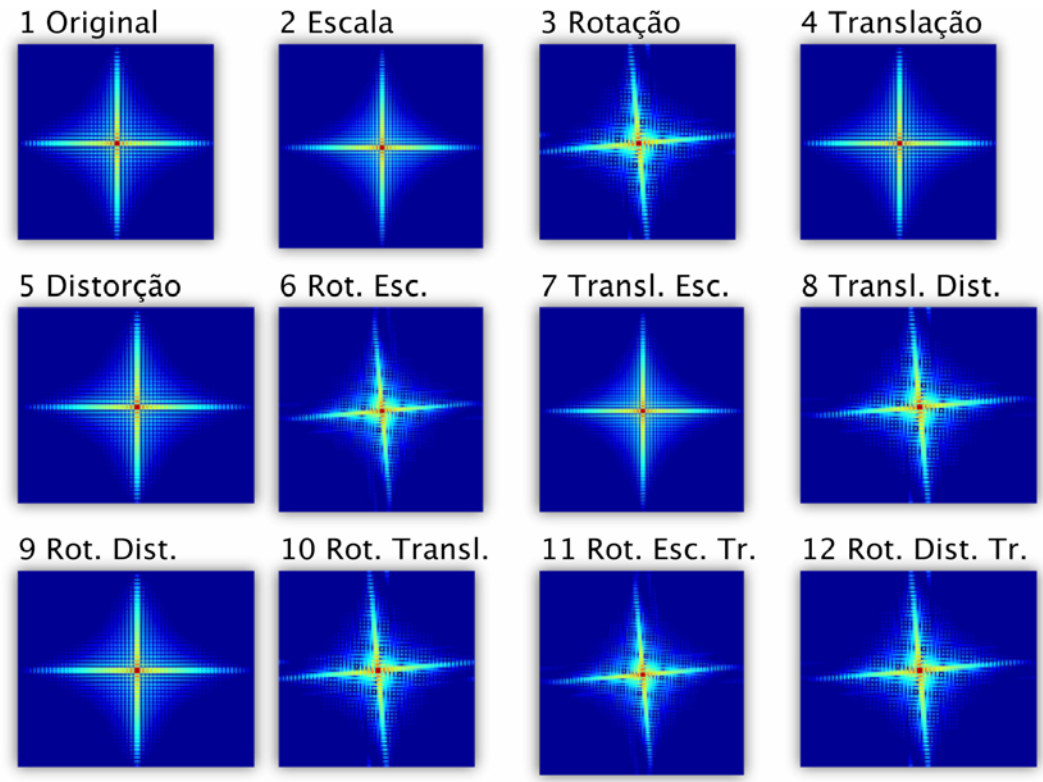

Figura 7-10: Transformadas de Fourier para Quadrados

A quantização porém, apresentou mostrar pequenas variações no resultado para as transformações do triângulo. É interessante verificar que as imagens escaladas ou distorcidas obtiveram magnitude da Transformada de Fourier com pequenas diferenças em relação à imagem original. Isto acontece devido ao fato das imagens terem tido a resolução alterada (nas operações de escala e distorção) através de interpolação que é uma função de aproximação. A imagem resultante não é portanto uma versão em escala ideal, mas sim uma versão aproximada. O mesmo acontece para todas as imagens rotacionadas.

Apesar dos erros gerados pela interpolação, é clara a diferença de resultados para o quadrado, o triângulo e o círculo. Isto é de interesse quando estas imagens serão utilizadas para comparações.

Através destes experimentos pôde se verificar visualmente que a implementação quando desenvolvida dentro das especificações apresentadas gerou resultados para o domínio discreto que confirmam as expectatívas relacionadas as comprovações matemáticas feitas para o domínio contínuo.

7.1.2. Mellin do tipo 1

As Transformadas de Mellin do tipo 1 aplicadas foram feitas seguindo a discretização desenvolvida na seção 2.5.3, Os parâmetros utilizados foram:

- $\Delta u = \Delta v = 0.2$;
- Variáveis u e v variando de 1 a 100: Resultado com resolução de 100 por 100;

Da Figura 7-11 à Figura 7-13 são apresentadas a magnitude da Transformada de Mellin do tipo 1 para as imagens vistas da Figura 7-2 à Figura 7-4.

Para a Transformada de Mellin do tipo 1 também foi possível verificar visualmente que a implementação sobre o domínio discreto apresentou os seguintes resultados esperados em relação as comprovações matemáticas para o domínio contínuo:

- As imagens correspondentes à magnitude da Transformada de Mellin do tipo 1 são próximas da imagem original para as operações de escala e distorção;

- As outras operações geram imagens que apresentam diferenças em relação as imagens originais;

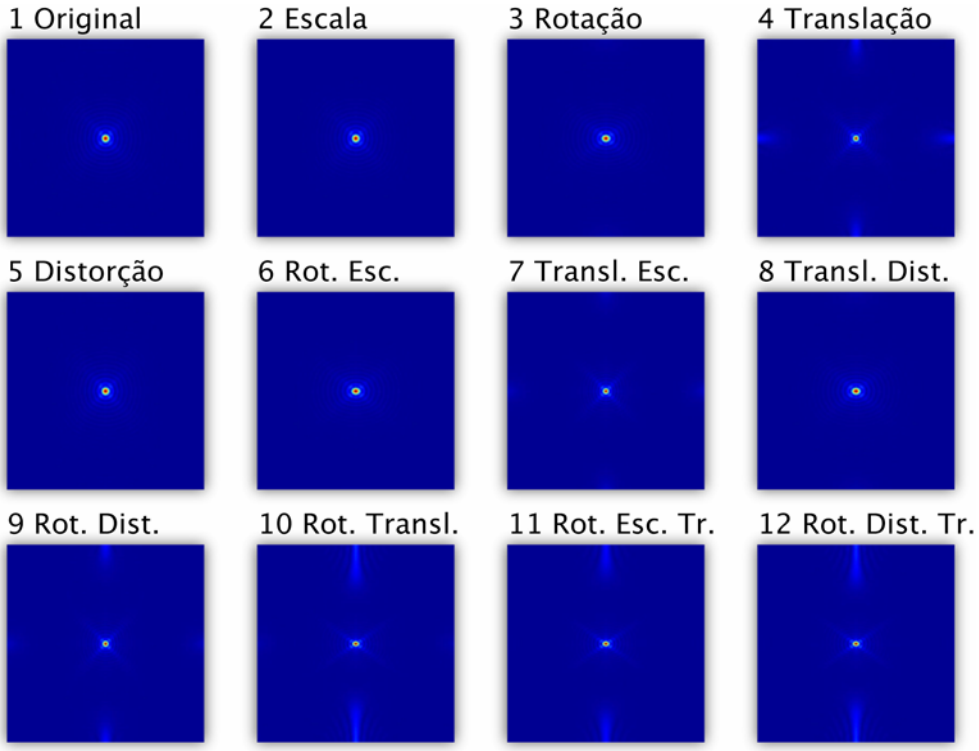

Figura 7-11: Transformadas de Mellin para círculos

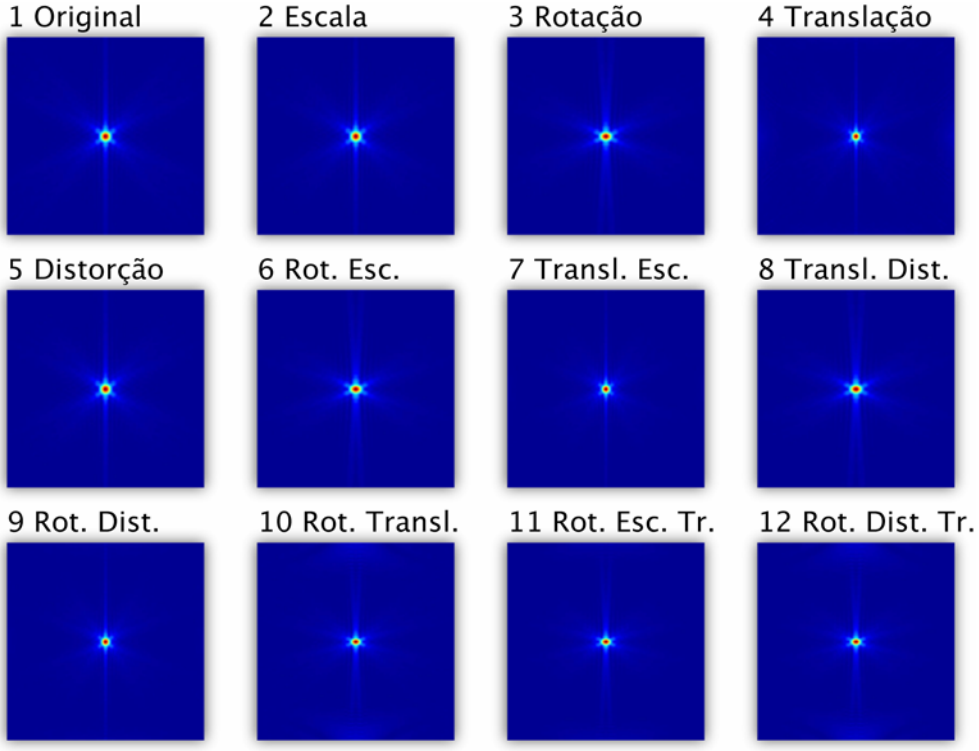

Figura 7-12: Transformadas de Mellin para triângulos

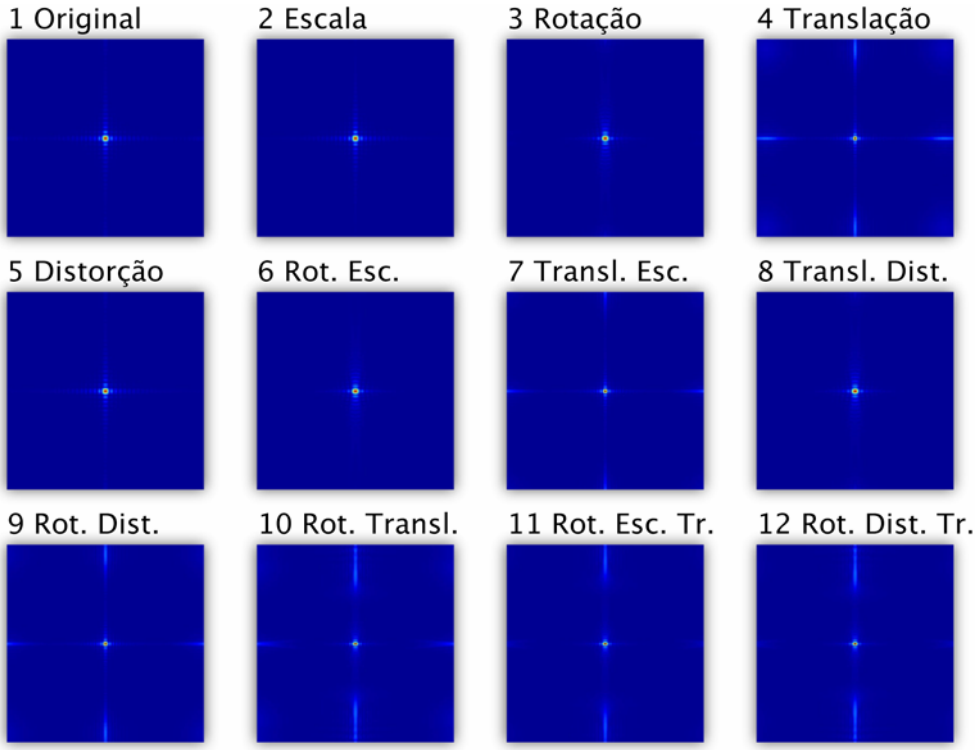

Figura 7-13: Transformadas de Mellin para quadrados

7.1.3. Mellin do tipo 2

Da Figura 7-20 à Figura 7-22, são apresentadas a magnitude da Transformada de Mellin do tipo 2 para as imagens vistas em Figura 7-2, Figura 7-3 e Figura 7-4. No caso da variações que apresentam rotação, são utilizadas as imagens vistas da Figura 7-5 à Figura 7-7.

Para a Transformada de Mellin do tipo 2, as imagens relativas a aos caso de rotação e escala são aquelas mais próximas das imagens originais. As outras imagens já apresentam variações mais distintas.

Pode-se perceber que apesar das imagens relativas a escala e rotação serem muito próximas das imagens originais, estas não são idênticas. Isto ocorre devido ao fato de as imagens não se tratarem de funções contínuas e sim funções discretas.

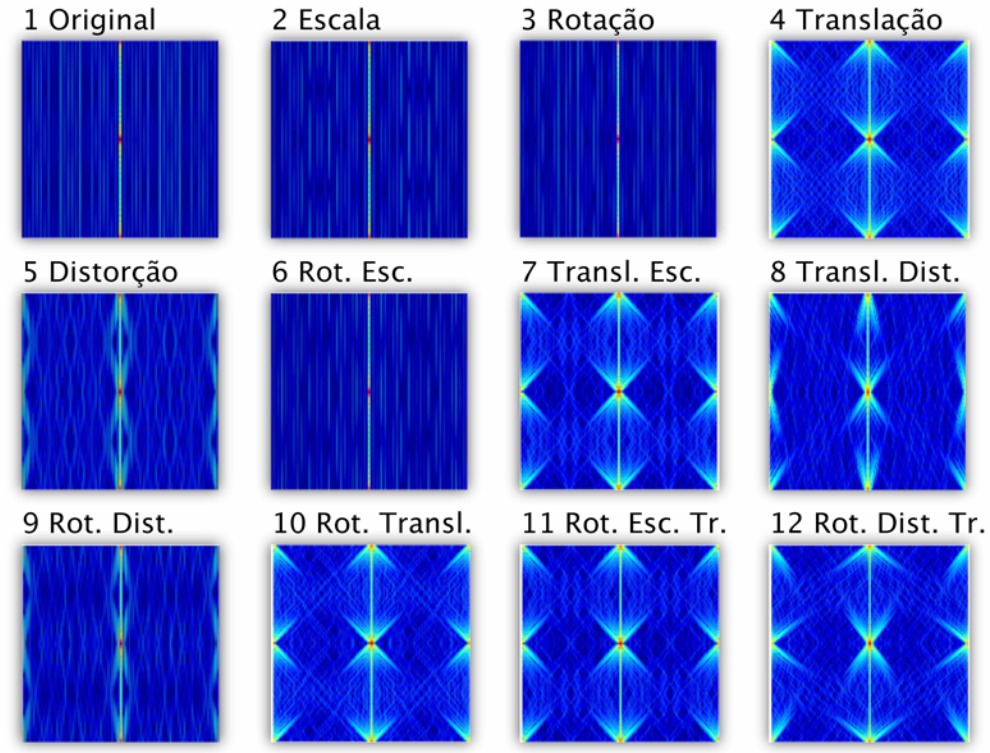

Figura 7-14: Transformadas de Mellin tipo 2 para círculos

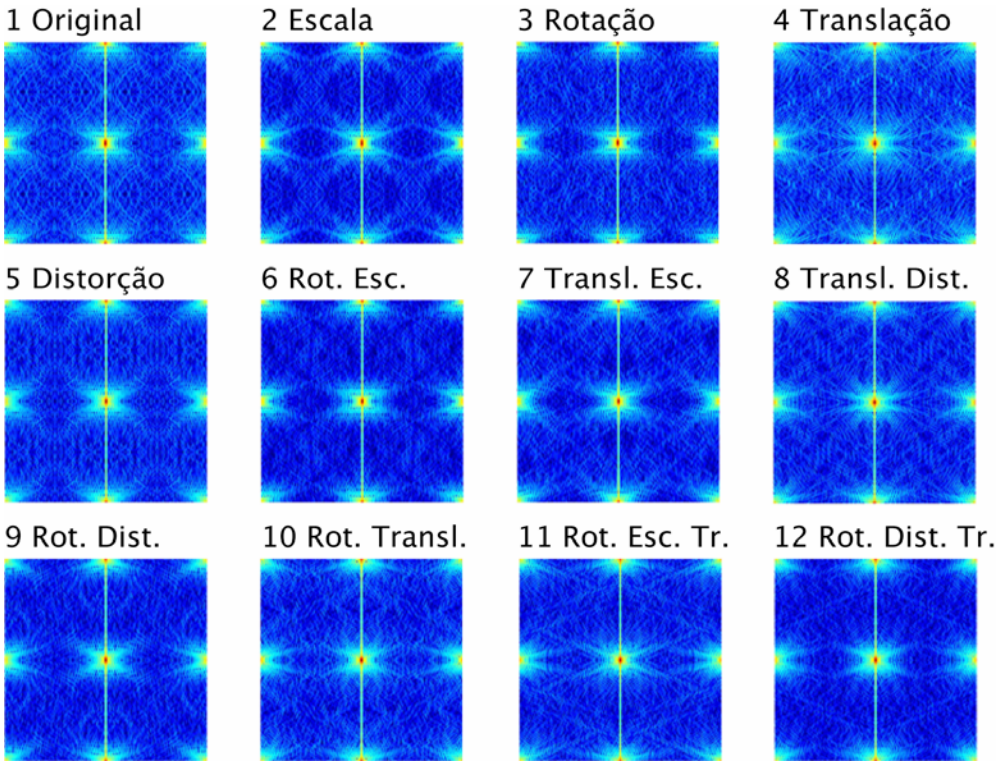

Figura 7-15: Transformadas de Mellin tipo 2 para triângulos

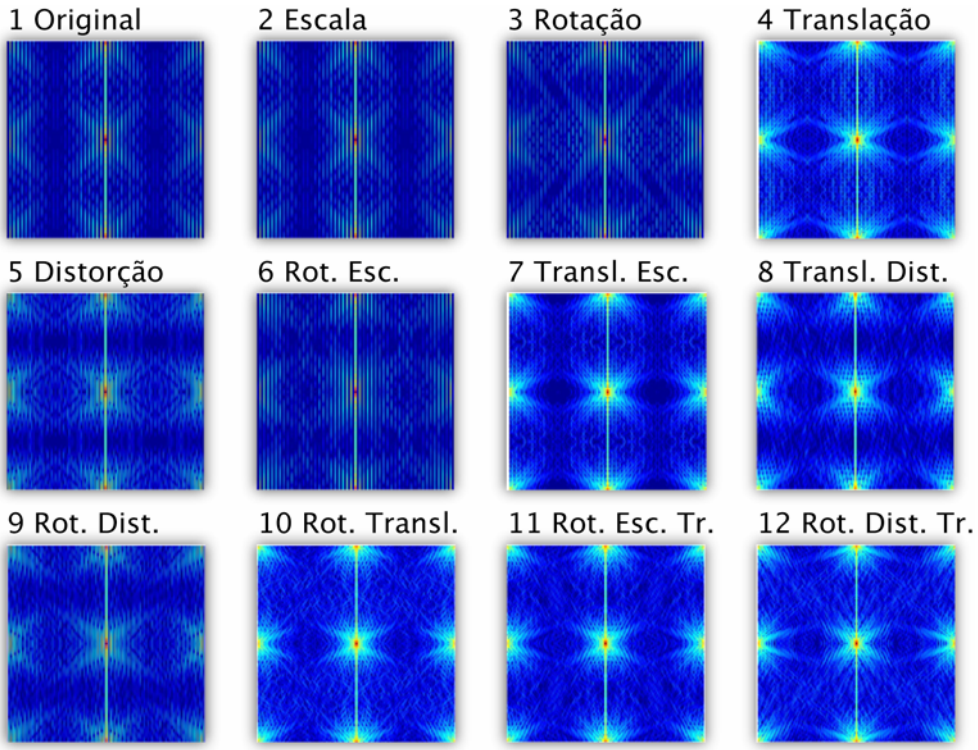

Figura 7-16: Transformadas de Mellin tipo 2 para quadrados

7.1.4. Fourier Mellin do tipo 1

Da Figura 7-17 à Figura 7-19 são apresentadas a magnitude da Transformada de Fourier Mellin do tipo 1 para as imagens vistas da Figura 7-2 à Figura 7-4.

Apesar de não ser possível visualizar as invariâncias esperadas através das figuras relativas a uma mesma forma, é possível observar que os resultados são bastante diferentes para as 3 formas utilizadas. É interessante notar que as transformações relacionadas ao quadrado e ao círculo são mais parecidas, porém bem diferentes das relacionadas ao triângulo. Isto é de se esperar dada a maior semelhança entre o círculo e o quadrado utilizados.

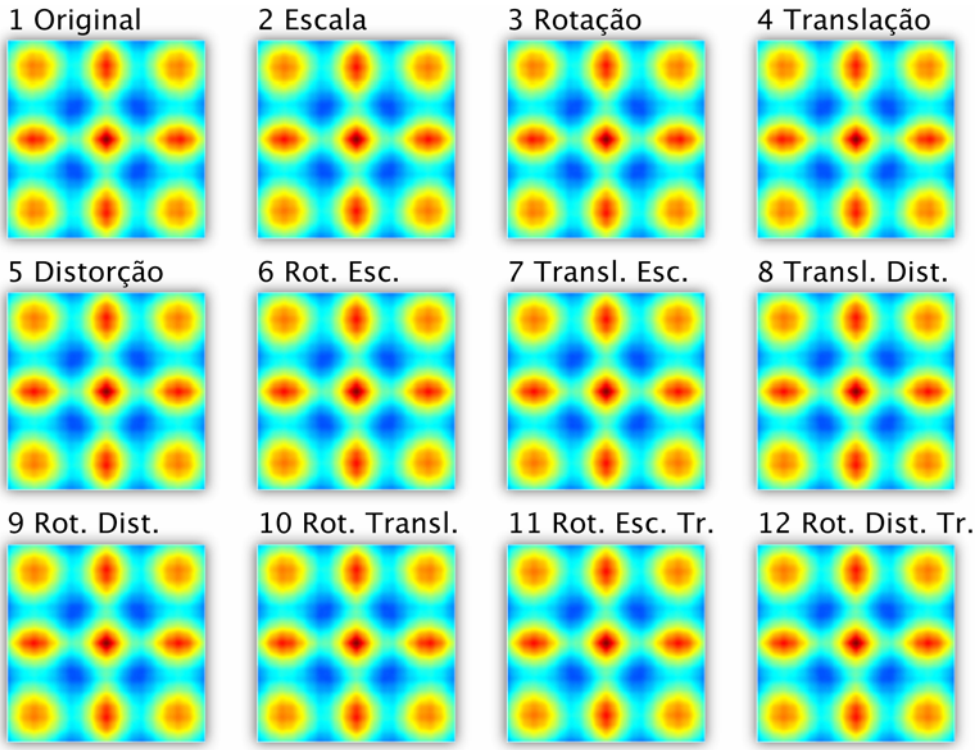

Figura 7-17: Transformadas de Fourier Mellin tipo 1 para círculos

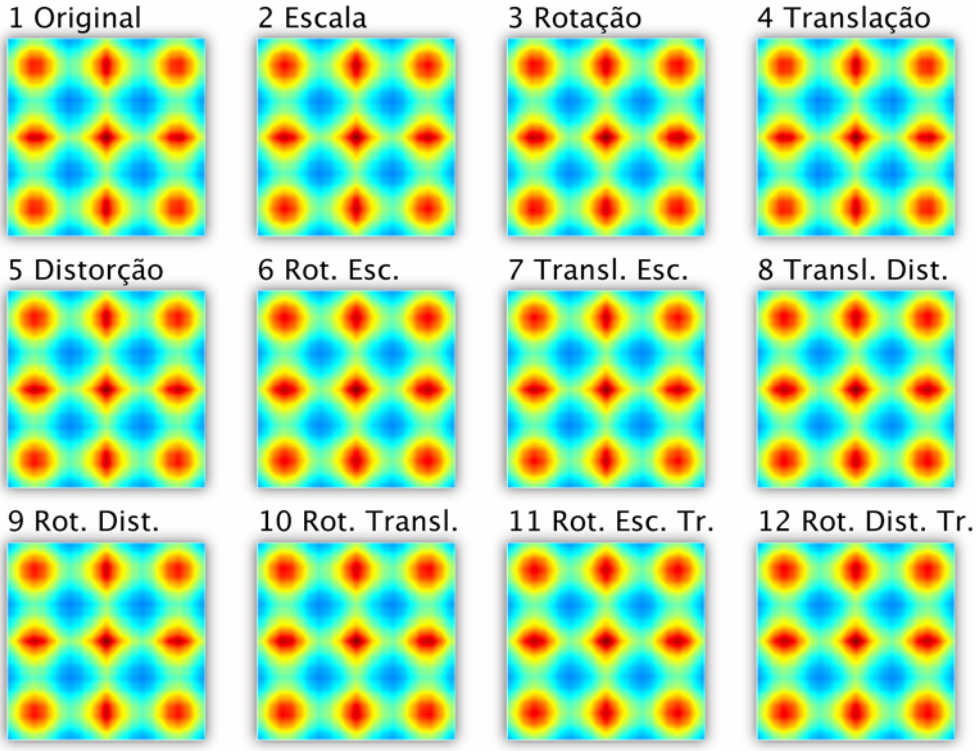

Figura 7-18: Transformadas de Fourier Mellin tipo 1 para quadrados

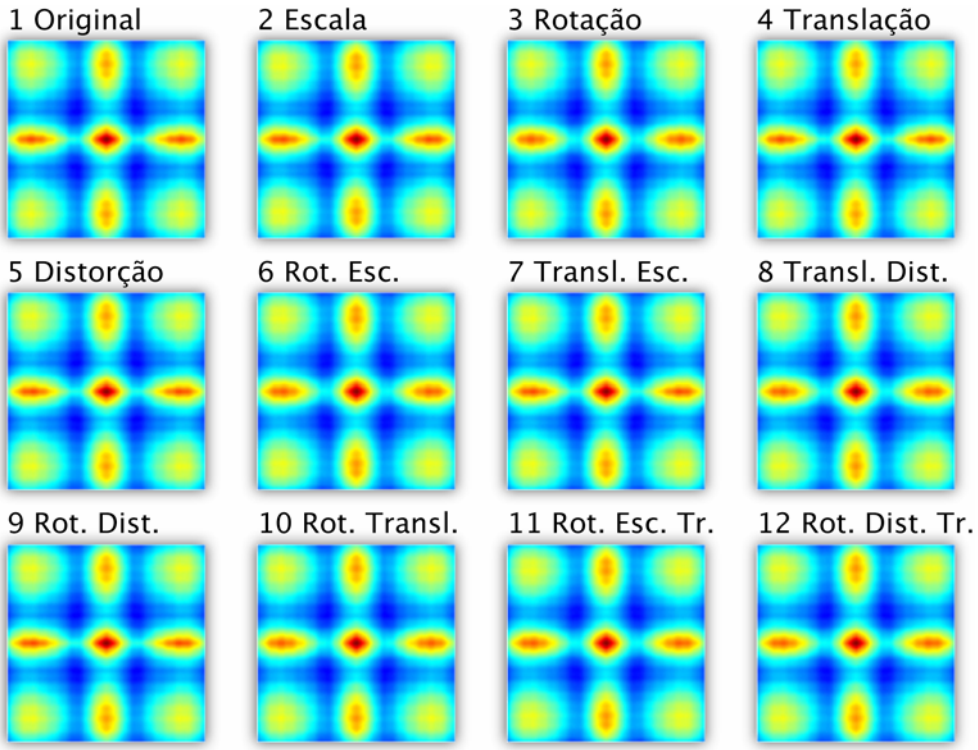

Figura 7-19: Transformadas de Fourier Mellin tipo 1 para triângulos

7.1.5. Fourier Mellin do tipo 2

Da Figura 7-20 à Figura 7-22 são apresentadas a magnitude da Transformada de Fourier Mellin do tipo 2 para as imagens vistas da Figura 7-2 à Figura 7-4.

Tal como para a Transformada de Fourier Mellin do tipo 1, não é possível se fazer uma análise visual apurada a partir das figuras. Também é perceptível que os resultados são próximos para o círculo e para o quadrado.

Após observar-se as Transformadas de Fourier Mellin do tipo 1 e do tipo 2 pode-se perceber que quando as Transformadas de Fourier e Mellin são aplicadas individualmente, o resultado é mais sensível a variações nas imagens, tal como as propriedades destas transformadas são mais perceptíveis em termos visuais.

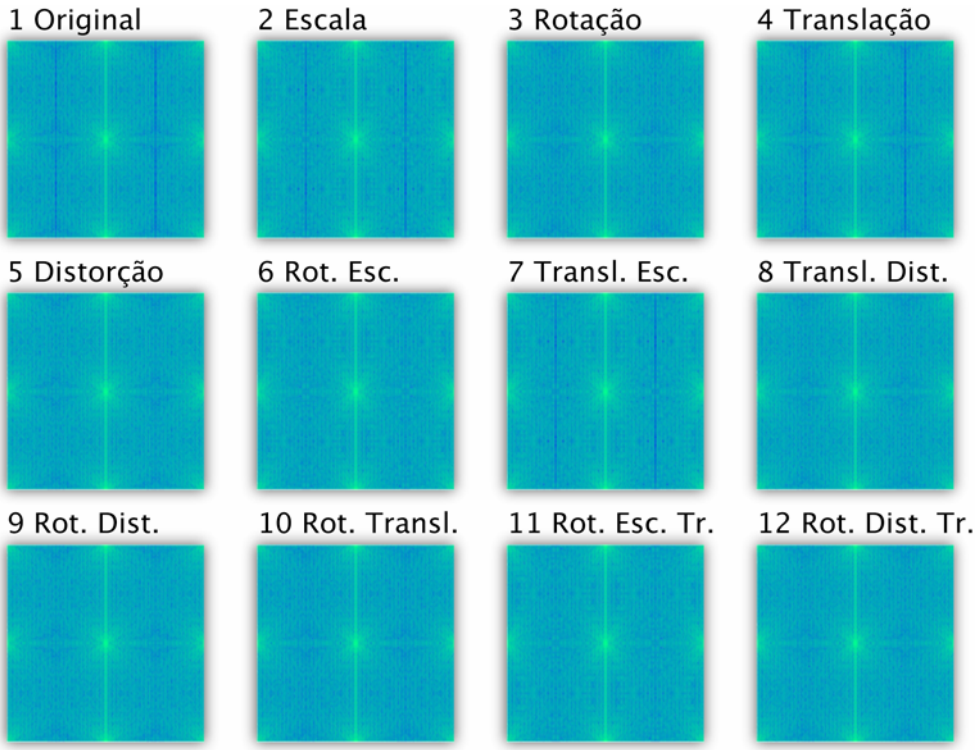

Figura 7-20: Transformadas de Fourier Mellin tipo 2 para círculos

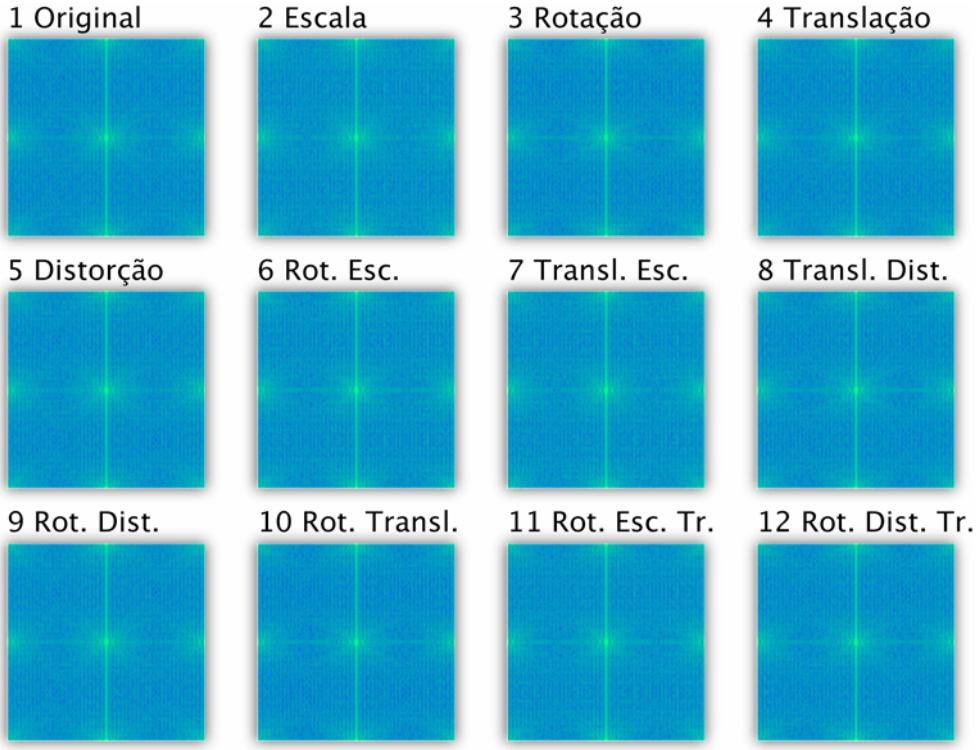

Figura 7-21: Transformadas de Fourier Mellin tipo 2 para triângulos

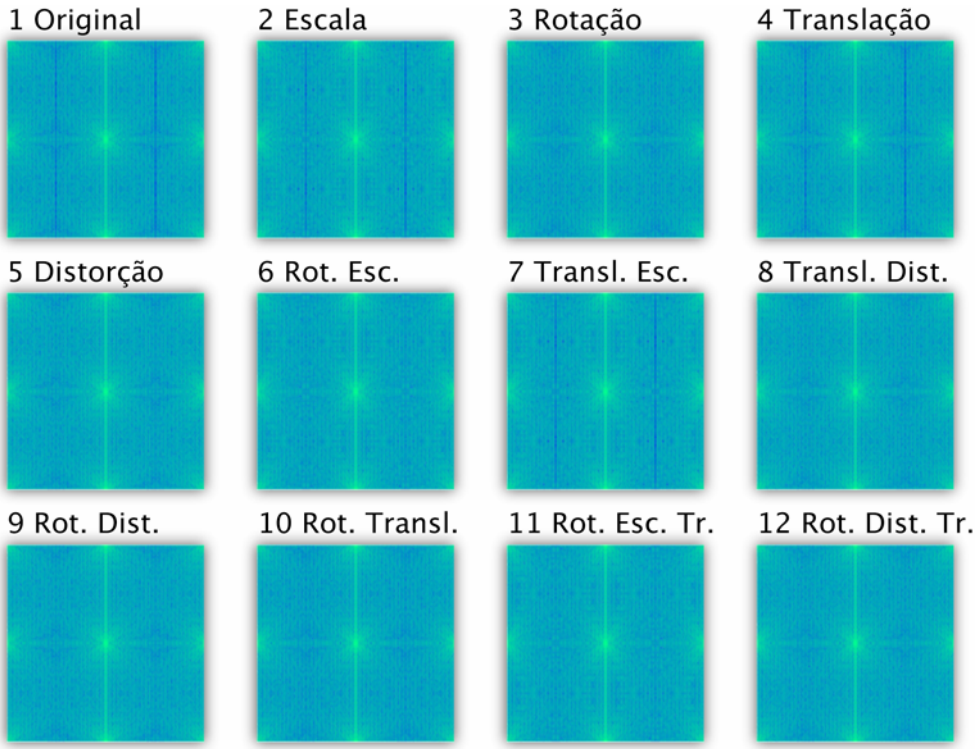

Figura 7-22: Transformadas de Fourier Mellin tipo 2 para quadrados

7.2. Comparações entre Transformadas

Foram feitas diversas comparações entre as imagens utilizando as transformações apresentadas. Estas comparações tiveram como objetivo a verificação quantitativa das invariâncias relativas a cada transformação.

Os resultados aqui obtidos também são de interesse de modo a se investigar as características de cada um dos métodos apresentados para poder se escolher aqueles mais adequados as necessidades da aplicação.

Apesar das invariâncias terem sido comprovadas matematicamente na seção 2 para funções contínuas, a aplicação das comparações com os métodos utilizados em funções discretas (imagens) apresenta particularidades que serão observadas.

As comparações foram feitas da seguinte forma:

- Testou-se o uso de correlação e distância euclidiana;
- Comparou-se as imagens originais do triângulo, quadrado e círculo com suas variações em relação à escala, rotação, translação e distorção;
- Foram feitas comparações entre as imagens originais do triângulo, quadrado e círculo;

Os valores que são apresentados foram calculados de modo a que as comparações representem o erro de comparação entre as imagens. Portanto, para o caso de distâncias euclidianas, os valores apresentados são a própria distância entre as imagens transformadas. Para o caso de correlações, os valores apresentados são dados por $1-R$, onde R é o valor da correlação entre as imagens transformadas. Perceba que como o maior valor de correlação entre duas funções é 1, então $1-R$ pode ser entendido como o erro de correlação entre duas funções. Isto foi feito para que a apresentação dos resultados seja mais facilmente compreendida. Assim, menores valores sempre significam uma melhor comparação entre as imagens.

Da

Tabela 7-1 à Tabela 7-3 são mostrados os valores obtidos para as comparações realizadas com as imagens originais.

Após a apresentação das tabelas, estes valores são visualizados em gráficos de modo a facilitar a investigação dos experimentos. Estes gráficos serão vistos a da Figura 7-23 à Figura 7-49, podendo sendo seguidos por comentários sobre os mesmos.

Nos gráficos apresentados, as barras coloridas em vermelho apresentam aquelas imagens para as quais a comparação deveria apresentar invariância.

Tabelas Gerais

Tabela 7-1: Comparações para o círculo

	Correlação Max.	Mellin 1 Cor.	Mellin 1 Dist.	Mellin 2 Cor.	Mellin 2 Dist.	Fourier Mellin 1 Cor.	Fourier Mellin 1 Dist.	Fourier Mellin 2 Cor.	Fourier Mellin 2 Dist.
Original	3,46E-14	6,66E-16	0,00E+00	1,44E-15	0,00E+00	0,00E+00	0,00E+00	0,00E+00	0,00E+00
Escala	1,15E-02	7,13E-03	5,72E-02	8,40E-04	1,39E+05	2,15E-05	5,51E+07	3,13E-04	1,94E+10
Rotação	9,22E-03	2,54E-02	1,98E-01	8,03E-05	1,33E+04	7,65E-06	1,93E+07	1,21E-04	7,36E+09
Translação	1,66E-02	1,66E-01	2,44E+00	4,78E-02	7,72E+06	1,99E-05	4,65E+07	2,81E-04	1,68E+10
Distorção	5,06E-03	3,44E-03	2,73E-02	5,27E-04	8,71E+04	8,69E-06	1,36E+07	9,64E-05	5,58E+09
Rot. Esc.	1,30E-02	2,96E-02	2,30E-01	7,66E-04	1,27E+05	2,09E-05	5,14E+07	2,95E-04	1,84E+10
Transl. Esc.	1,19E-02	1,75E-01	2,61E+00	4,92E-02	7,94E+06	2,08E-05	5,41E+07	3,07E-04	1,90E+10
Rot. Dist.	1,05E-02	2,70E-02	2,09E-01	5,50E-04	9,10E+04	1,35E-05	3,48E+07	2,01E-04	1,25E+10
Transl. Dist.	1,42E-02	1,71E-01	2,52E+00	4,85E-02	7,83E+06	2,02E-05	5,00E+07	2,91E-04	1,78E+10
Rot. Transl.	1,14E-02	2,39E-01	3,96E+00	4,73E-02	7,64E+06	1,79E-05	4,23E+07	2,49E-04	1,52E+10
Rot. Transl. Esc.	1,33E-02	2,41E-01	4,03E+00	4,91E-02	7,91E+06	2,06E-05	5,09E+07	2,91E-04	1,82E+10
Rot. Transl. Dist.	1,24E-02	2,35E-01	3,88E+00	4,82E-02	7,78E+06	2,55E-05	5,86E+07	3,44E-04	2,10E+10
Quadrado	1,30E-01	9,54E-02	8,31E-01	8,92E-03	1,52E+06	1,27E-03	3,35E+09	1,53E-02	9,92E+11
Triângulo	3,28E-01	1,06E-01	9,09E-01	4,86E-02	7,87E+06	3,97E-03	1,24E+10	3,99E-02	3,33E+12

Tabela 7-2: Comparações para o triângulo

	Correlação Max.	Mellin 1 Cor.	Mellin 1 Dist.	Mellin 2 Cor.	Mellin 2 Dist.	Fourier Mellin 1 Cor.	Fourier Mellin 1 Dist.	Fourier Mellin 2 Cor.	Fourier Mellin 2 Dist.
Original	9,89E-14	7,77E-16	0,00E+00	1,67E-15	0,00E+00	2,22E-16	0,00E+00	3,33E-16	0,00E+00
Escala	3,93E-03	1,68E-03	6,82E-03	3,71E-04	5,27E+04	7,82E-06	3,18E+06	2,01E-05	9,30E+08
Rotação	5,97E-02	2,98E-02	1,20E-01	6,48E-04	9,20E+04	1,00E-04	3,36E+06	4,44E-04	8,56E+09
Translação	3,23E-06	3,05E-02	2,88E-01	1,27E-02	1,80E+06	1,87E-09	4,31E+02	1,32E-08	2,61E+05
Distorção	2,03E-03	1,20E-03	4,90E-03	2,45E-04	3,48E+04	6,91E-07	2,32E+04	6,40E-06	1,25E+08
Rot. Esc.	5,43E-02	2,86E-02	1,15E-01	8,54E-04	1,21E+05	8,14E-05	2,79E+06	4,46E-04	8,63E+09
Transl. Esc.	3,94E-03	3,58E-02	3,13E-01	1,28E-02	1,81E+06	7,87E-06	3,22E+06	2,05E-05	9,36E+08
Rot. Dist.	5,88E-02	2,97E-02	1,20E-01	7,66E-04	1,09E+05	9,82E-05	3,29E+06	4,50E-04	8,68E+09
Transl. Dist.	2,03E-03	3,33E-02	2,94E-01	1,31E-02	1,85E+06	6,88E-07	2,31E+04	6,45E-06	1,27E+08
Rot. Transl.	5,98E-02	6,48E-02	4,79E-01	7,93E-03	1,12E+06	1,00E-04	3,36E+06	4,44E-04	8,56E+09
Rot. Transl. Esc.	5,43E-02	6,23E-02	4,44E-01	8,19E-03	1,16E+06	8,14E-05	2,79E+06	4,45E-04	8,62E+09
Rot. Transl. Dist.	5,87E-02	6,13E-02	4,49E-01	7,99E-03	1,13E+06	9,81E-05	3,29E+06	4,50E-04	8,68E+09
Quadrado	4,11E-01	2,03E-01	1,74E+00	5,38E-02	9,02E+06	8,55E-03	2,83E+10	9,01E-02	6,88E+12
Triângulo	3,27E-01	1,01E-01	8,33E-01	4,87E-02	7,85E+06	3,55E-03	1,29E+10	3,58E-02	3,49E+12

Tabela 7-3: Comparações para o quadrado

	Correlação Max.	Mellin 1 Cor.	Mellin 1 Dist.	Mellin 2 Cor.	Mellin 2 Dist.	Fourier Mellin 1 Cor.	Fourier Mellin 1 Dist.	Fourier Mellin 2 Cor.	Fourier Mellin 2 Dist.
Original	4,03E-14	0,00E+00	0,00E+00	0,00E+00	0,00E+00	0,00E+00	0,00E+00	0,00E+00	0,00E+00
Escala	7,62E-04	7,99E-04	7,32E-03	4,08E-05	6,96E+03	4,92E-06	8,18E+06	3,61E-05	2,37E+09
Rotação	3,26E-02	6,52E-02	6,08E-01	2,43E-04	4,14E+04	5,58E-05	9,77E+06	7,59E-04	3,96E+10
Translação	7,02E-07	3,65E-01	9,45E+00	3,77E-02	6,33E+06	1,15E-09	8,35E+02	6,09E-09	3,22E+05
Distorção	3,85E-04	3,98E-04	3,64E-03	2,88E-05	4,90E+03	2,47E-06	2,28E+06	1,31E-05	8,08E+08
Rot. Esc.	3,69E-02	6,47E-02	6,07E-01	3,86E-04	6,58E+04	6,34E-05	1,12E+07	9,59E-04	5,01E+10
Transl. Esc.	7,63E-04	3,69E-01	9,47E+00	3,70E-02	6,22E+06	4,91E-06	8,18E+06	3,61E-05	2,37E+09
Rot. Dist.	3,53E-02	6,37E-02	5,94E-01	3,23E-04	5,50E+04	6,31E-05	1,11E+07	8,81E-04	4,60E+10
Transl. Dist.	3,84E-04	3,67E-01	9,45E+00	3,74E-02	6,27E+06	2,47E-06	2,28E+06	1,31E-05	8,06E+08
Rot. Transl.	3,26E-02	3,62E-01	1,02E+01	4,21E-02	7,05E+06	5,58E-05	9,77E+06	7,59E-04	3,96E+10
Rot. Transl. Esc.	3,69E-02	3,64E-01	1,05E+01	4,26E-02	7,14E+06	6,34E-05	1,12E+07	9,58E-04	5,00E+10
Rot. Transl. Dist.	3,53E-02	3,63E-01	1,03E+01	4,22E-02	7,08E+06	6,31E-05	1,11E+07	8,80E-04	4,60E+10
Quadrado	1,52E-01	1,06E-01	9,18E-01	9,97E-03	1,70E+06	1,58E-03	3,10E+09	1,96E-02	1,13E+12
Triângulo	4,11E-01	2,03E-01	1,74E+00	5,38E-02	9,02E+06	8,55E-03	2,83E+10	9,01E-02	6,88E+12

Como já explicado, os valores apresentados nas tabelas são referentes ao erro de comparação entre as diferentes imagens e a imagem original.

A investigação sobre estes valores será feita a seguir, onde para cada método e cada figura geométrica (círculo, quadrado e triângulo) são apresentados gráficos, comparando os erros de comparação encontrados, seguindo de comentários sobre os mesmos.

7.2.1. Correlações

Os gráficos relativos às correlações podem ser vistos da Figura 7-23 à Figura 7-25.

É importante citar que a correlação entre imagens de diferentes resoluções foi feita após igualar a escala entre as imagens como proposto em 2.6.2.

Os resultados mostram que o erro de correlação entre as diferentes formas geométricas (círculo, quadrado e triângulo) foi maior do que entre as variações (escala, rotação e outras) das figuras quando comparadas com suas imagens originais. A diferença entre erros porém não é muito grande. Isto é interessante ao mostrar que o método pode ser empregado para se distinguir imagens.

A medida de correlação proposta apresentou os melhores resultados (menores erros) ao se comparar imagens que sofreram translação, escala e distorção.

Apesar de ser uma medida simples e comum, o uso da correlação como feito neste trabalho mostrou conseguir distinguir imagens distintas e correlacionar imagens semelhantes em um número de casos próximo ao quando se usou a transformação de Fourier Mellin do tipo 1.

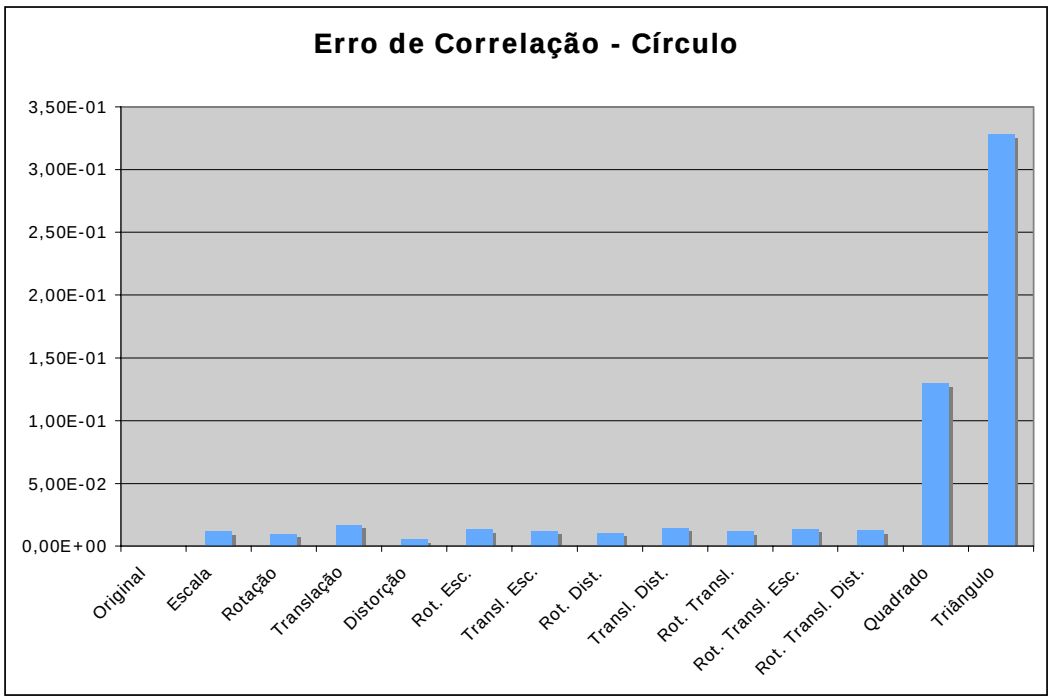

Figura 7-23: Erro de Correlação – Círculo

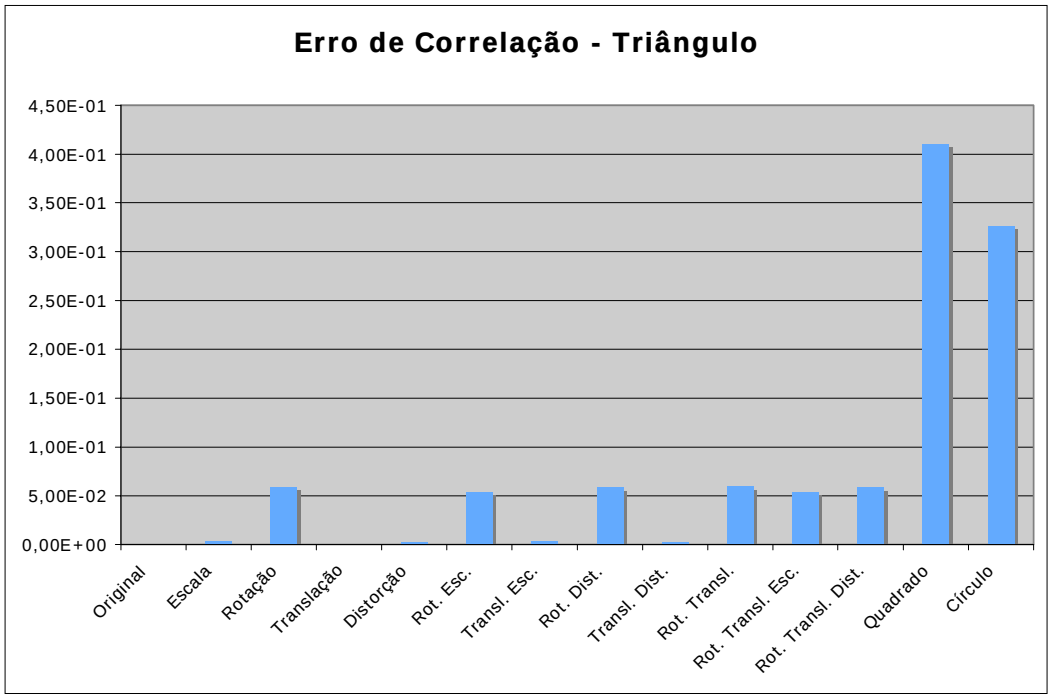

Figura 7-24: Erro de Correlação – Triângulo

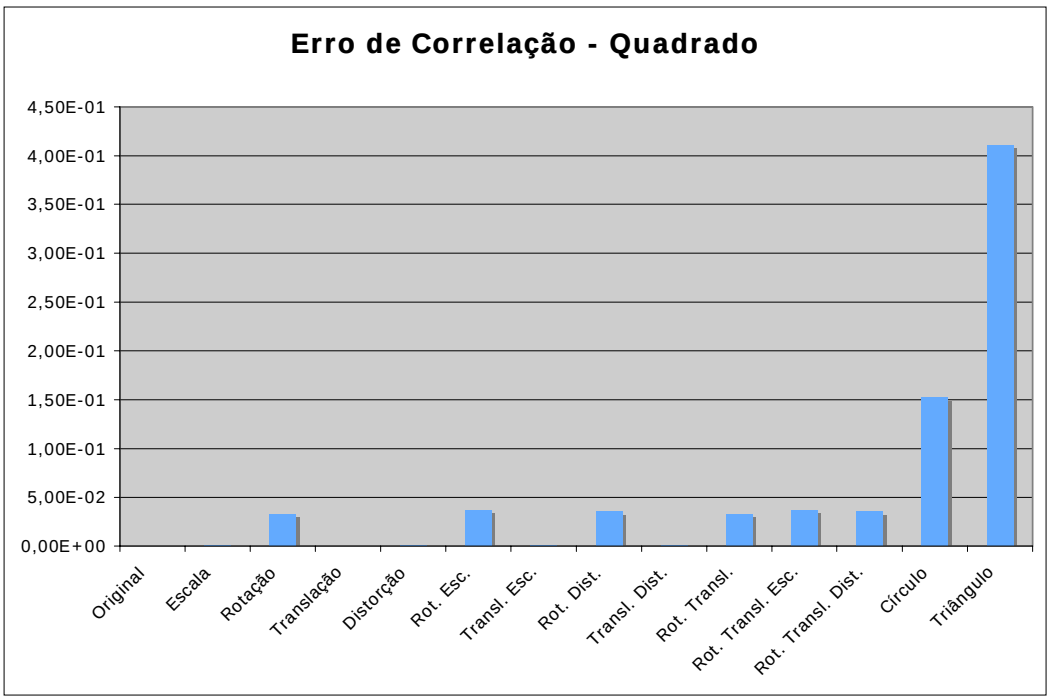

Figura 7-25: Erro de Correlação – Quadrado

7.2.2. Mellin do tipo 1

Os gráficos relativos à Transformada de Mellin do tipo 1 podem ser vistos da Figura 7-26 à Figura 7-31.

Havia sido comprovado matematicamente que a Transformada de Mellin do tipo 1 é invariável à escala horizontal e vertical para funções contínuas.

Na aplicação feita pode-se observar que as comparações utilizadas realmente apresentam menor erro quando são comparadas imagens que sofreram tais tipos de variações no domínio discreto.

Em alguns dos casos observados as comparações entre imagens que haviam sofrido rotação ou translação e as imagens originais apresentaram erro inferior as comparações entre formas geométricas diferentes. Entretanto, a Transformada de Mellin do tipo 1 não é uma boa medida de comparação para imagens que sofreram estas transformações, como já havia sido verificado na análise matemática da Transformada.

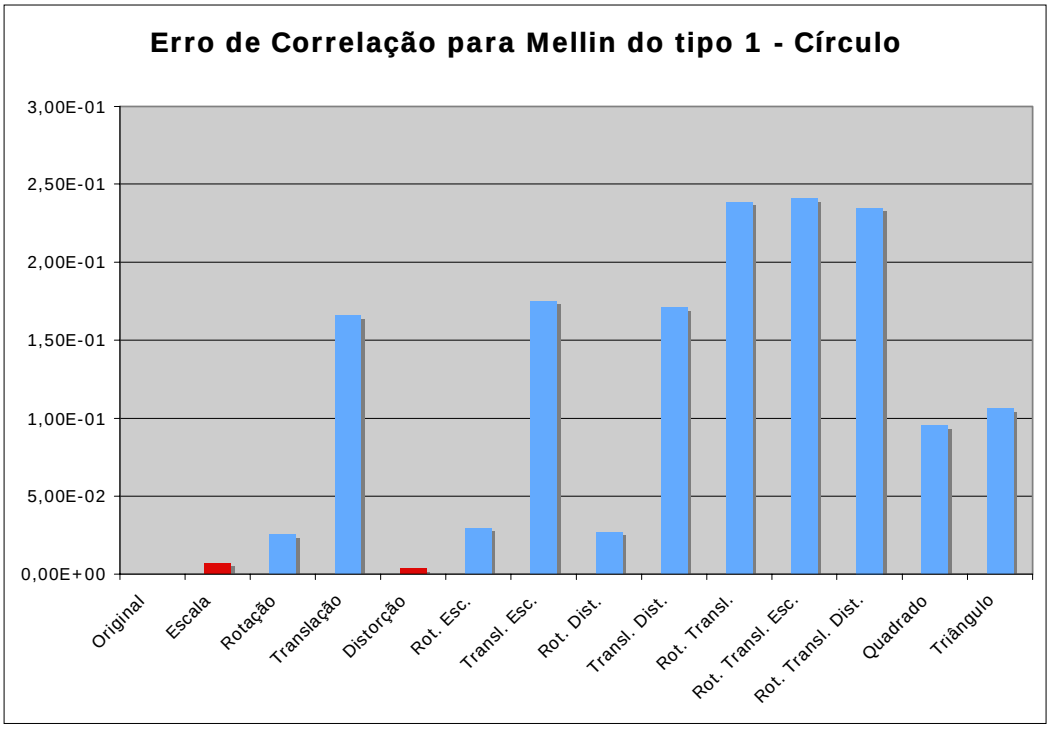

Figura 7-26: Erro de Correlação para Mellin do tipo 1 – Círculo

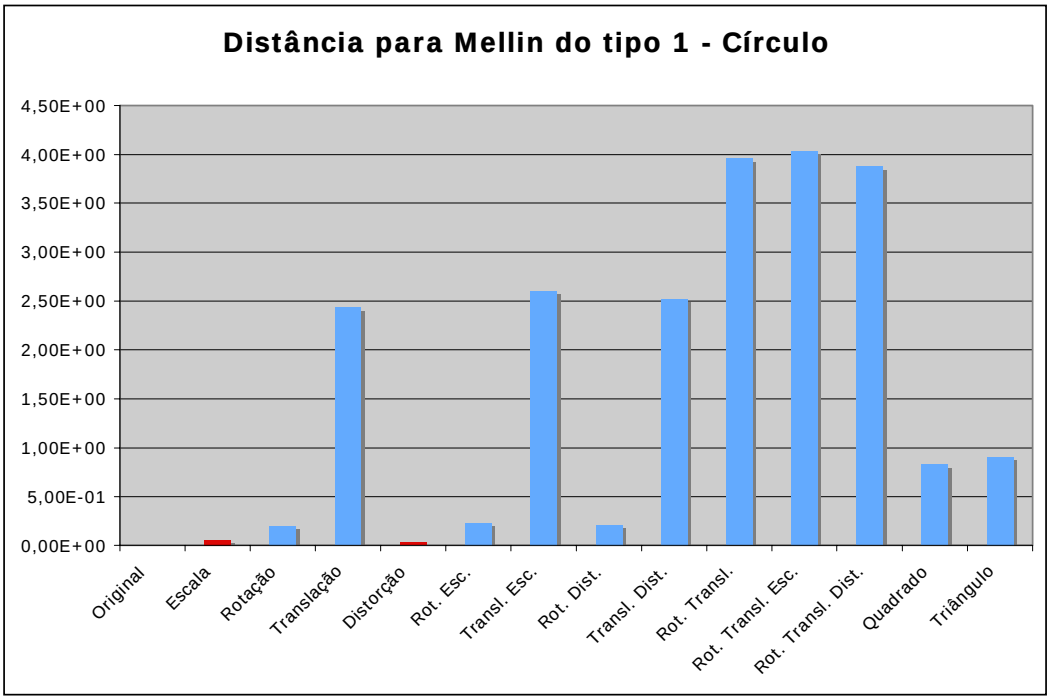

Figura 7-27: Distância para Mellin do tipo 1 – Círculo

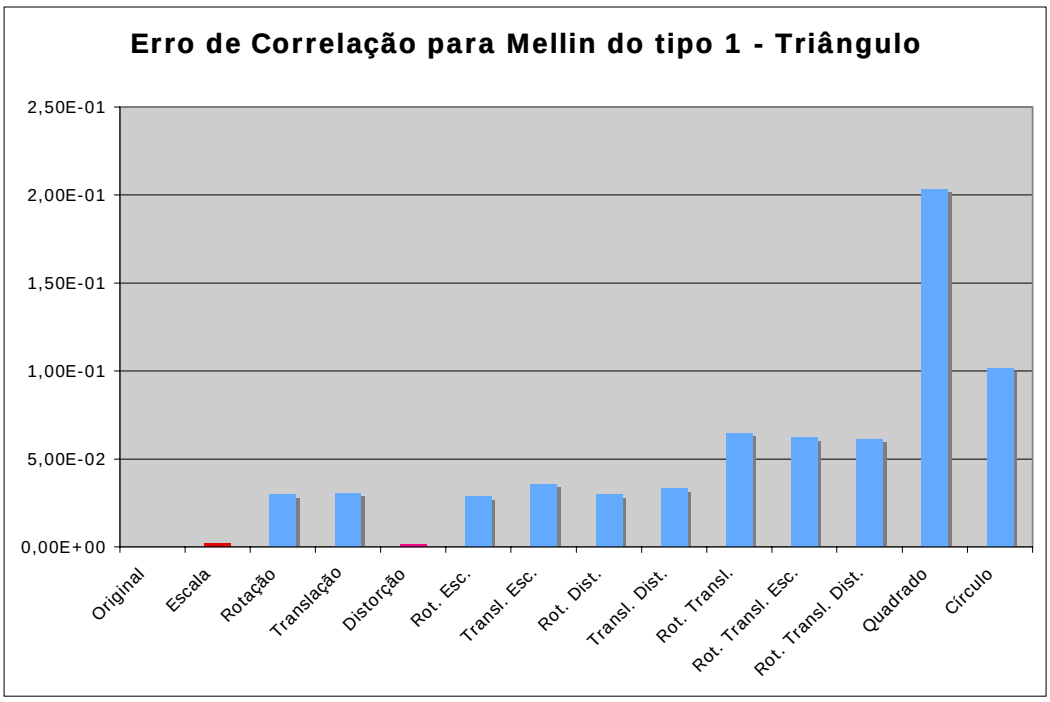

Figura 7-28: Erro de Correlação para Mellin do tipo 1 – Triângulo

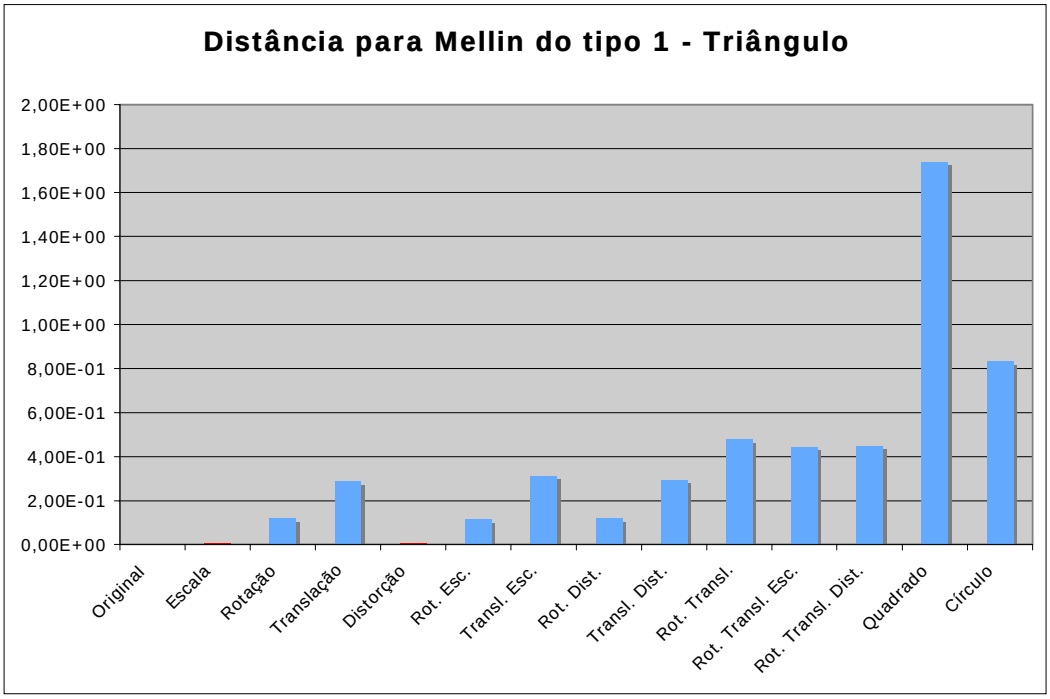

Figura 7-29: Distância para Mellin do tipo 1 – Triângulo

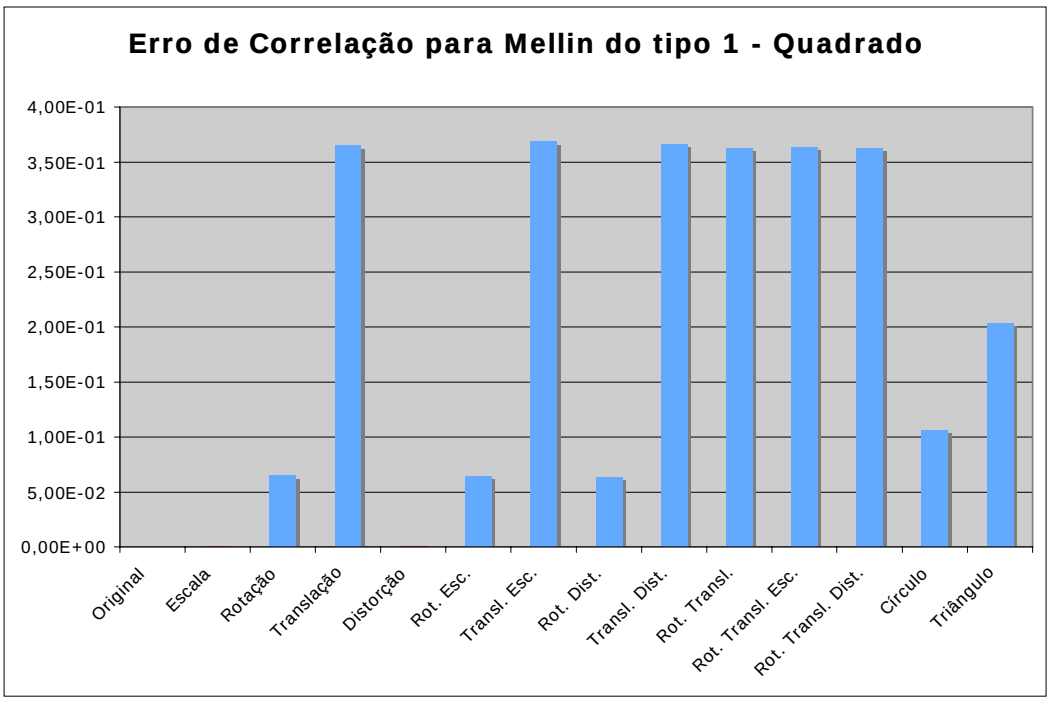

Figura 7-30: Erro de Correlação para Mellin do tipo 1 – Quadrado

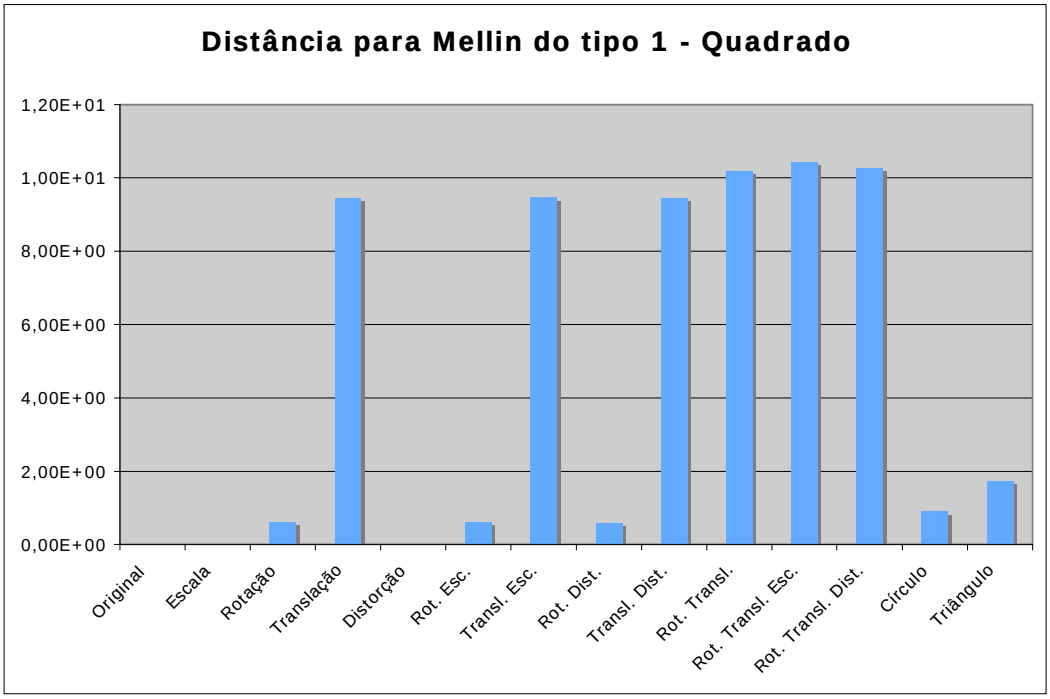

Figura 7-31: Distância para Mellin do tipo 1 – Quadrado

7.2.3.
Mellin do tipo 2

Os gráficos relativos à Transformada de Mellin do tipo 2 podem ser vistos da Figura 7-32 à Figura 7-37.

A invariância à escala e rotação pode ser observada dado que os erros de comparação para as imagens que sofreram estas modificações foram inferiores àqueles relativos as que não sofreram.

Menores erros também foram verificados para o caso de distorções. Porém, para distorções muito grandes isto não seria observado.

Novamente as comparações como aplicadas refletiram o que já havia sido comprovado para o caso de funções contínuas.

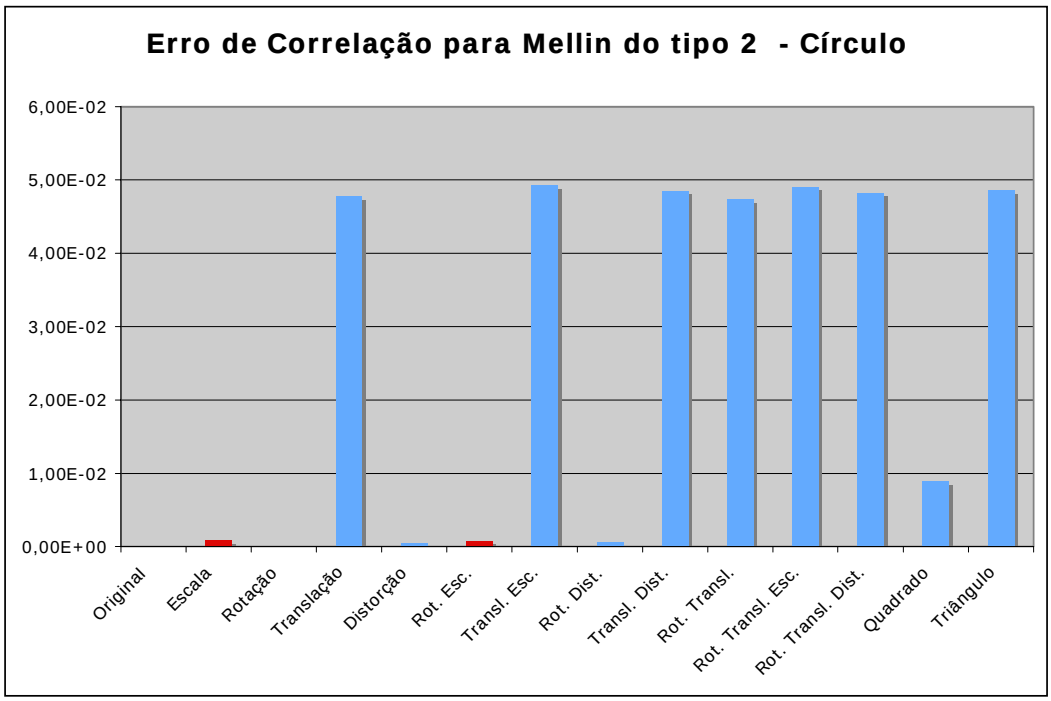

Figura 7-32: Erro de Correlação para Mellin do tipo 2 – Círculo

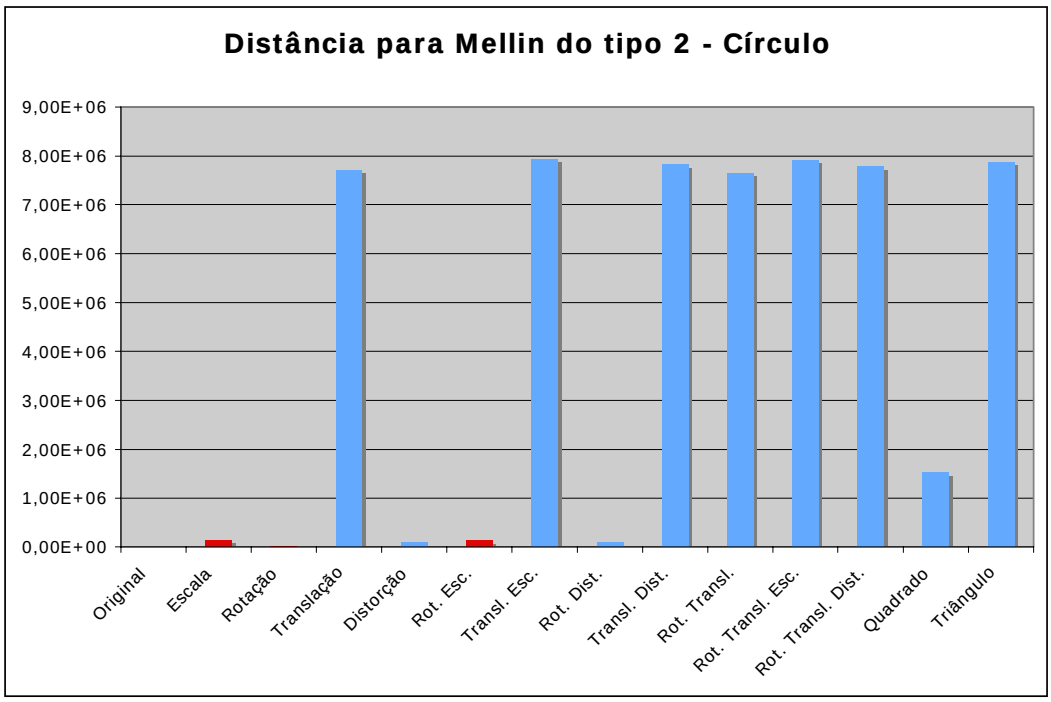

Figura 7-33: Distância para Mellin do tipo 2 – Círculo

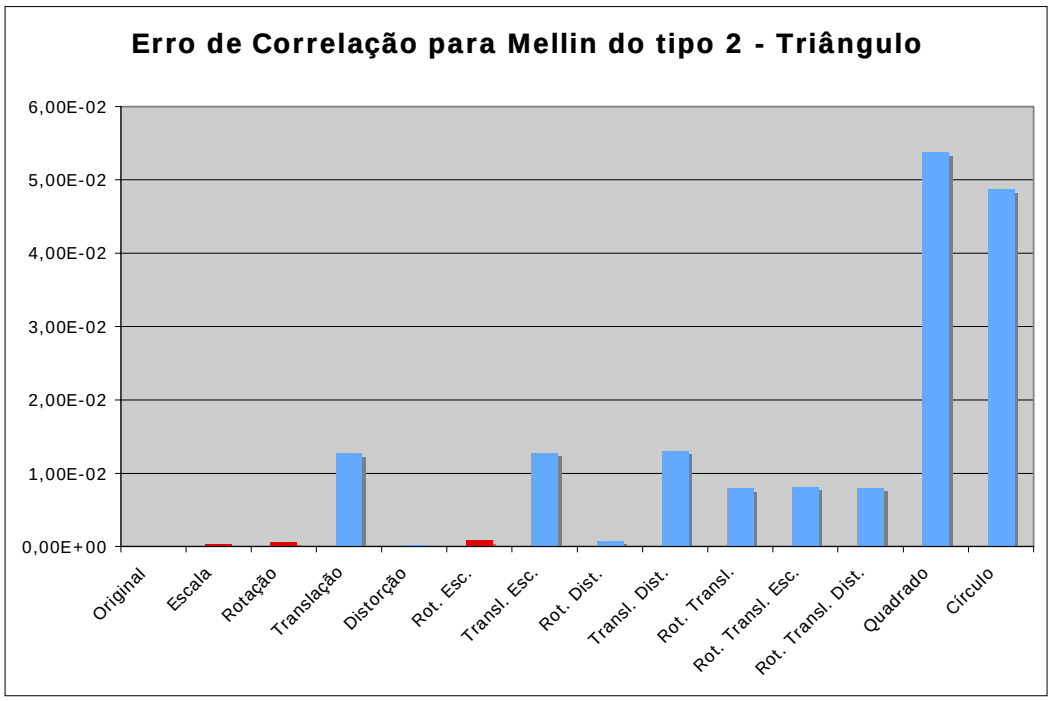

Figura 7-34: Erro de Correlação para Mellin do tipo 2 – Triângulo

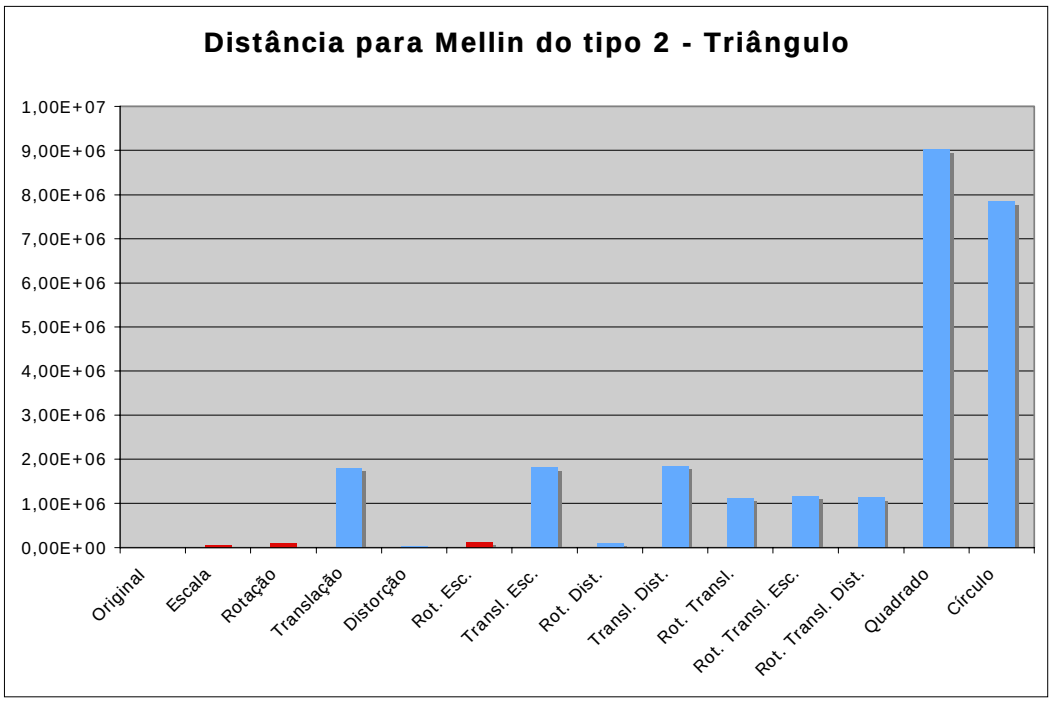

Figura 7-35: Distância para Mellin do tipo 2 – Triângulo

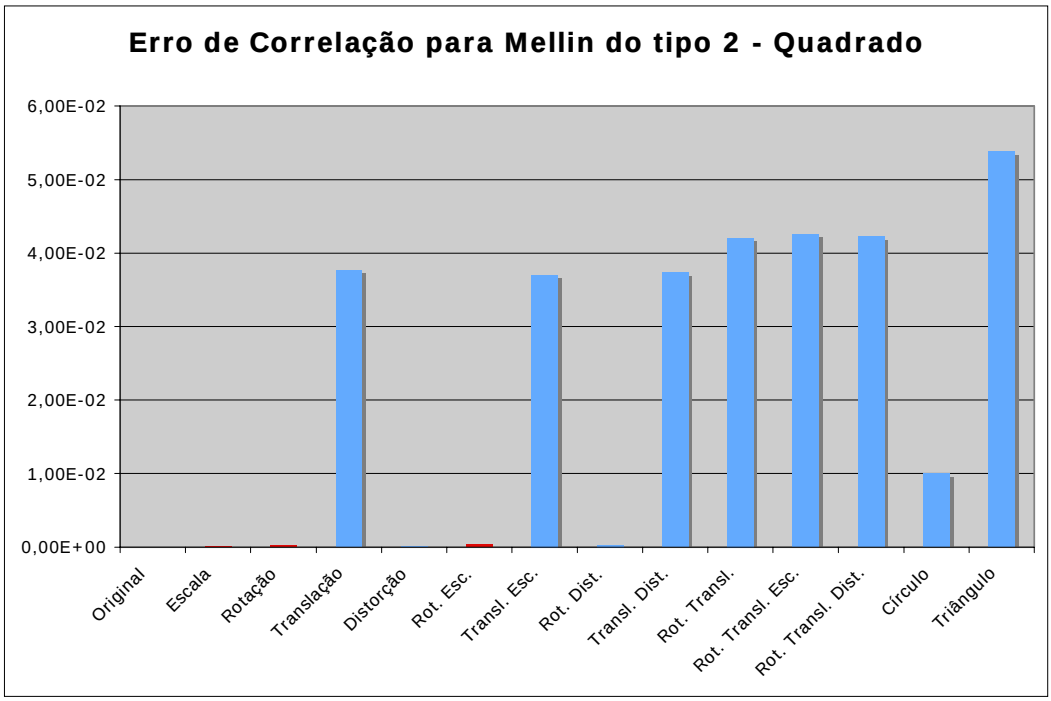

Figura 7-36: Erro de Correlação para Mellin do tipo 2 – Quadrado

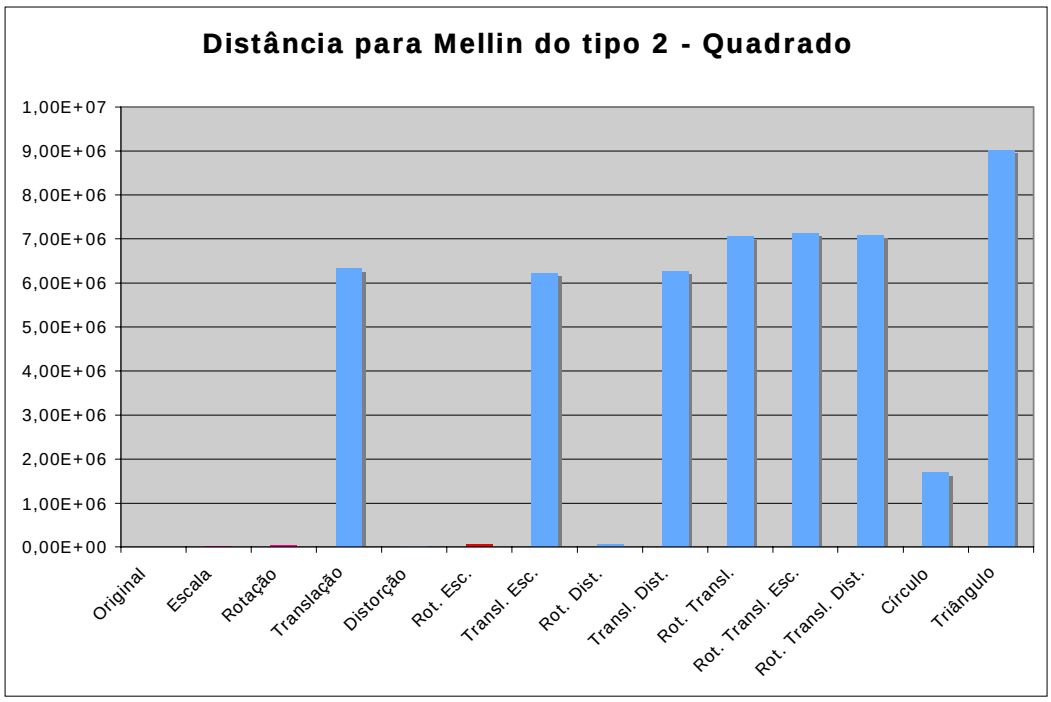

Figura 7-37: Distância para Mellin do tipo 2 – Quadrado

7.2.4. Fourier Mellin do tipo 1

Os gráficos relativos à Transformada de Fourier Mellin do tipo 1 podem ser vistos da Figura 7-38 à Figura 7-43.

Os erros de correlação para a aplicação da Transformada de Fourier Mellin do tipo 1 demonstraram que esta realmente só não apresentou invariância à rotação, como já era esperado. Entretanto, isto não pode ser verificado ao se observar os erros de comparação utilizando-se distância euclidiana.

Os gráficos relativos a distância euclidiana demonstram que esta não é uma boa medida a ser adotada. Isto foi verificado não só nos experimentos descritos aqui mas ao longo do projeto em diversas situações.

Portanto, esta análise apresenta casos em que, apesar de se esperar que as comparações refletissem as invariâncias comprovadas matematicamente para o domínio contínuo, o modo em que estas comparações são implementadas no domínio discreto pode levar a não ocorrência do esperado.

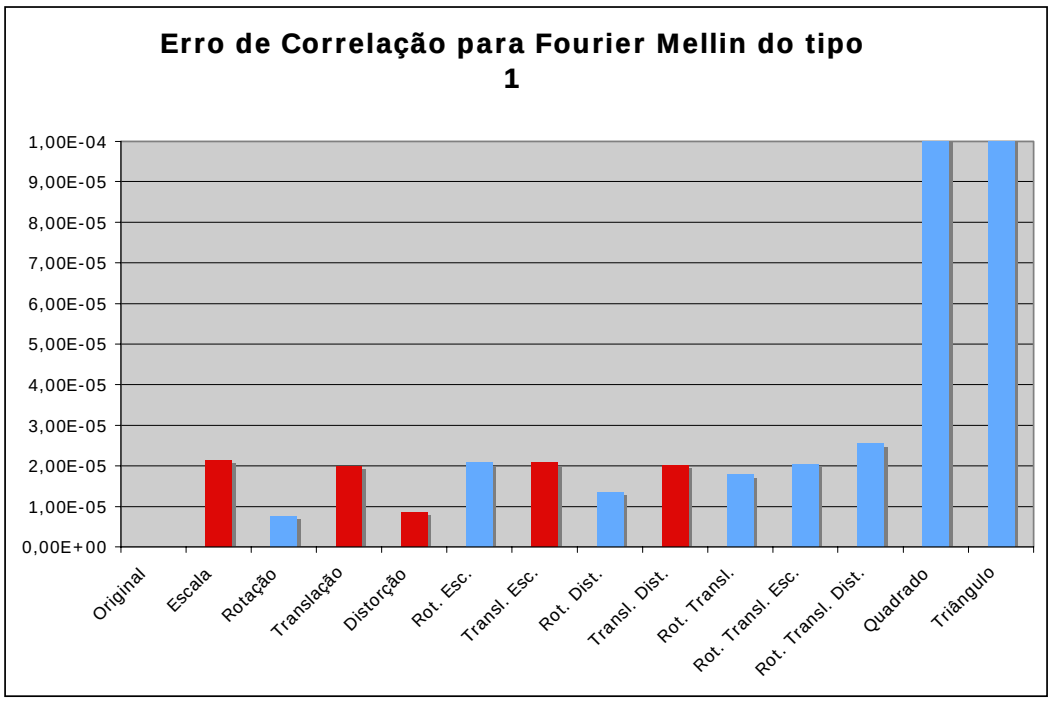

Figura 7-38: Erro de Correlação para Fourier Mellin do tipo 1 – Círculo

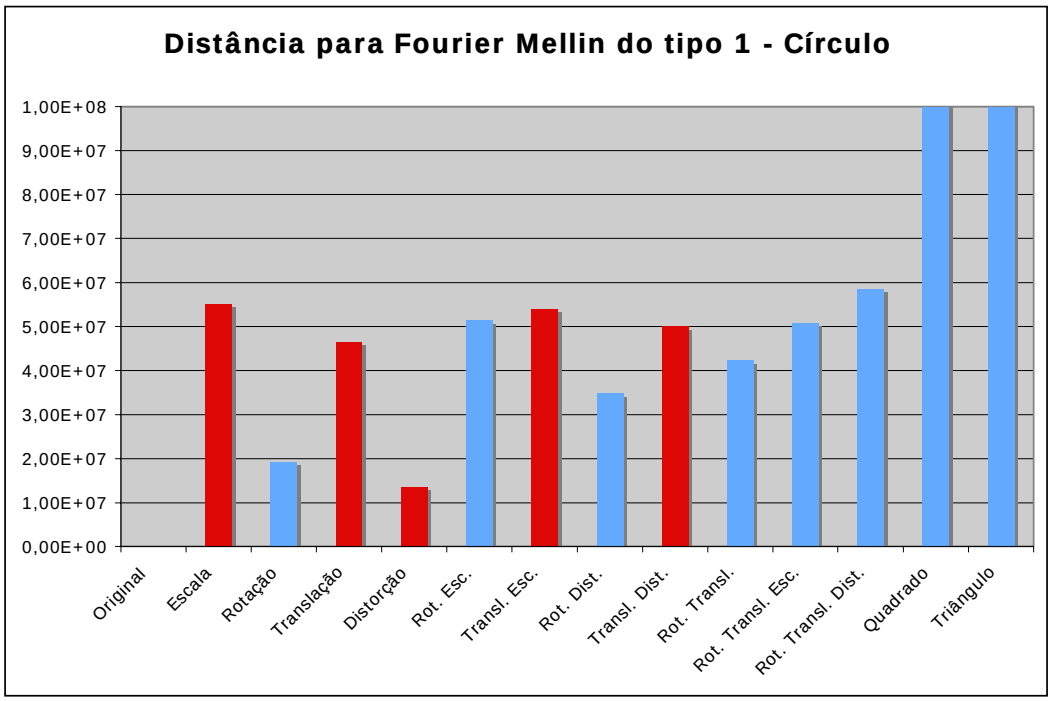

Figura 7-39: Distância para Fourier Mellin do tipo 1 – Círculo

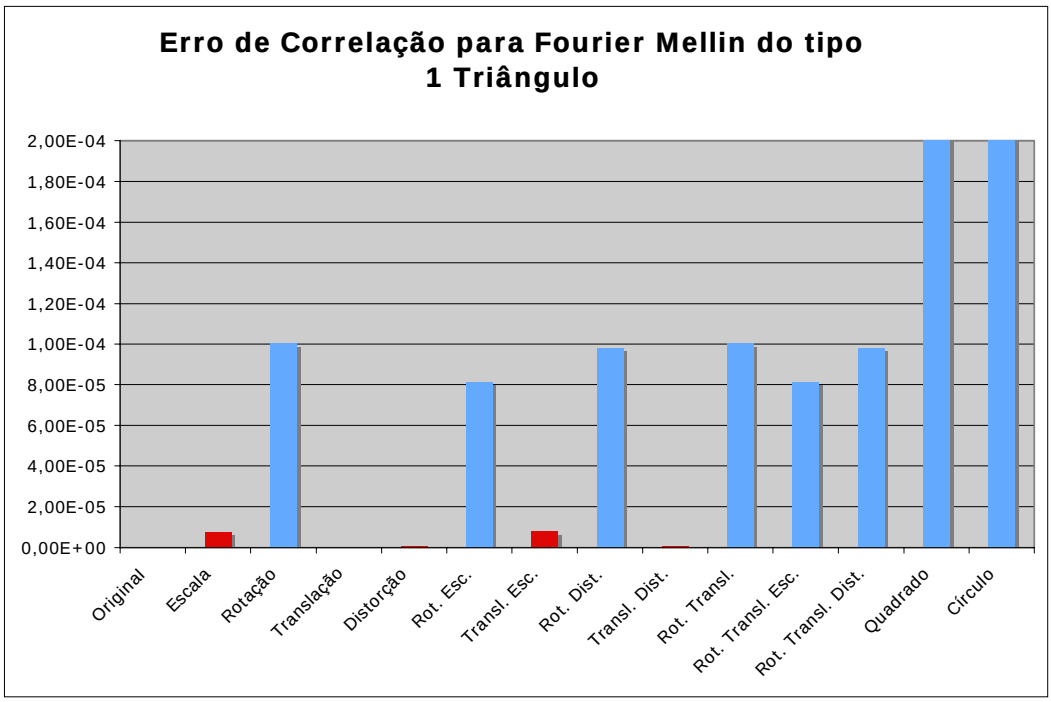

Figura 7-40: Erro de Correlação para Fourier Mellin do tipo 1 – Triângulo

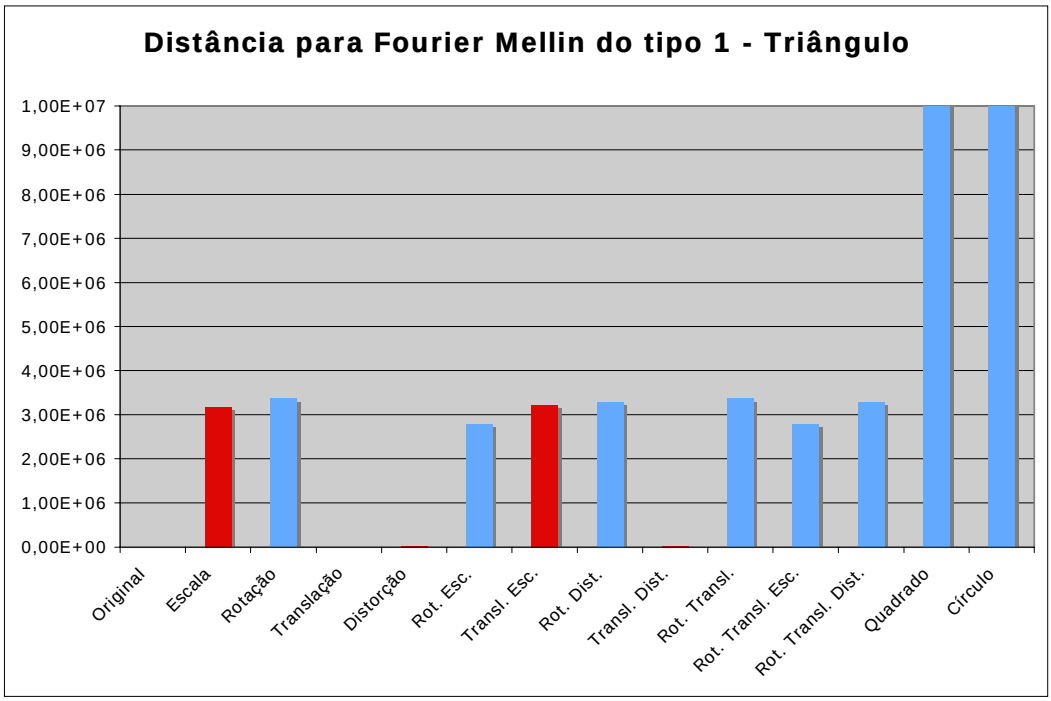

Figura 7-41: Distância para Fourier Mellin do tipo 1 – Triângulo

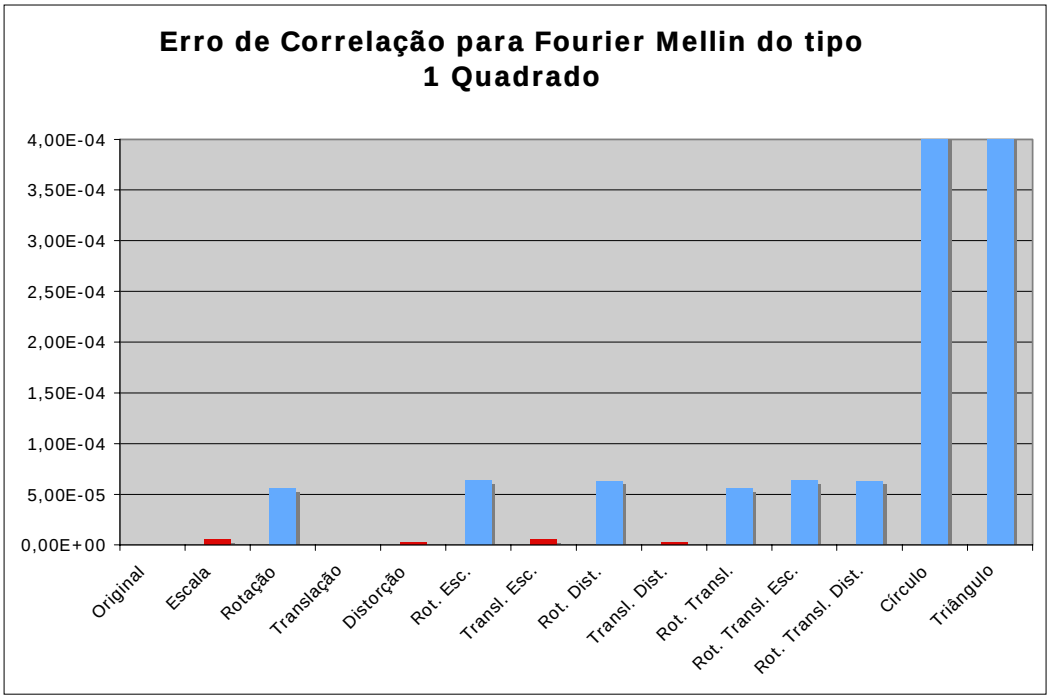

Figura 7-42: Erro de Correlação para Fourier Mellin do tipo 1 – Quadrado

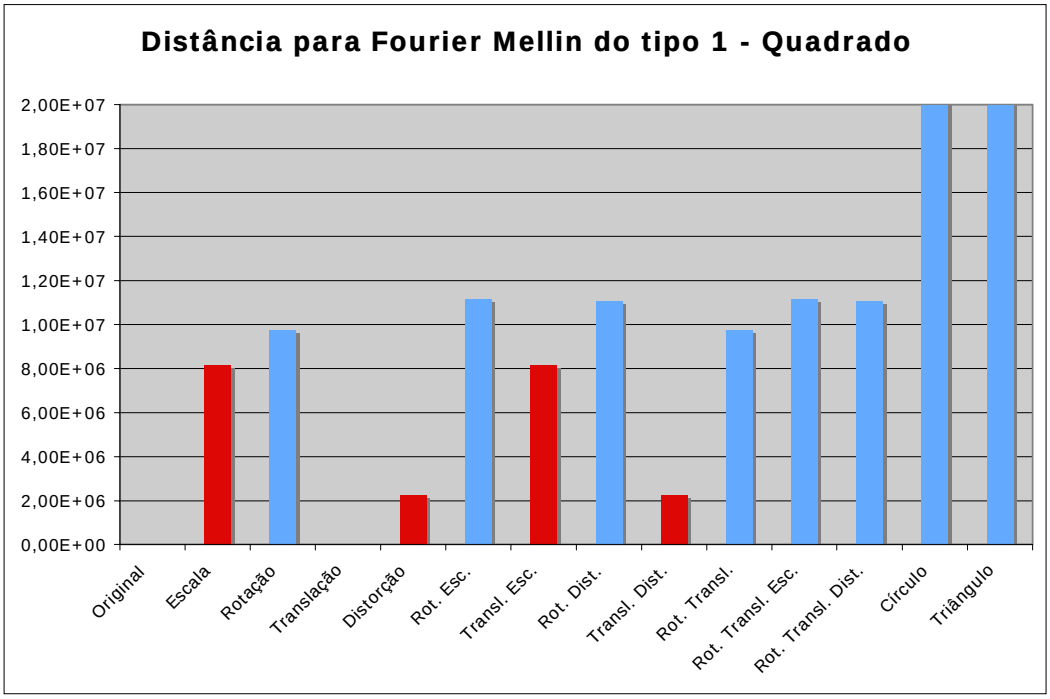

Figura 7-43: Distância para Fourier Mellin do tipo 1 – Quadrado

7.2.5.
Fourier Mellin do tipo 2

Os gráficos relativos à Transformada de Fourier Mellin do tipo 2 podem ser vistos da Figura 7-44 à Figura 7-49.

Para todas as variações aplicadas nas imagens, o erro de comparação ao se usar a Transformada de Mellin do tipo 2 foi bastante inferior ao erro em se comparar as diferentes formas geométricas.

Pode-se perceber nos gráficos que o erro para as variações que possuiam rotação em alguns dos experimentos é superior as outras. Este fato não pode ser entendido como a Transformada em questão não possuindo invariância à rotação pois ele não é comum a todos experimentos e, além disso, os erros relativos a rotação nos casos comentados ainda é extremamente inferior ao erro em se comparar as diferentes formas geométricas.

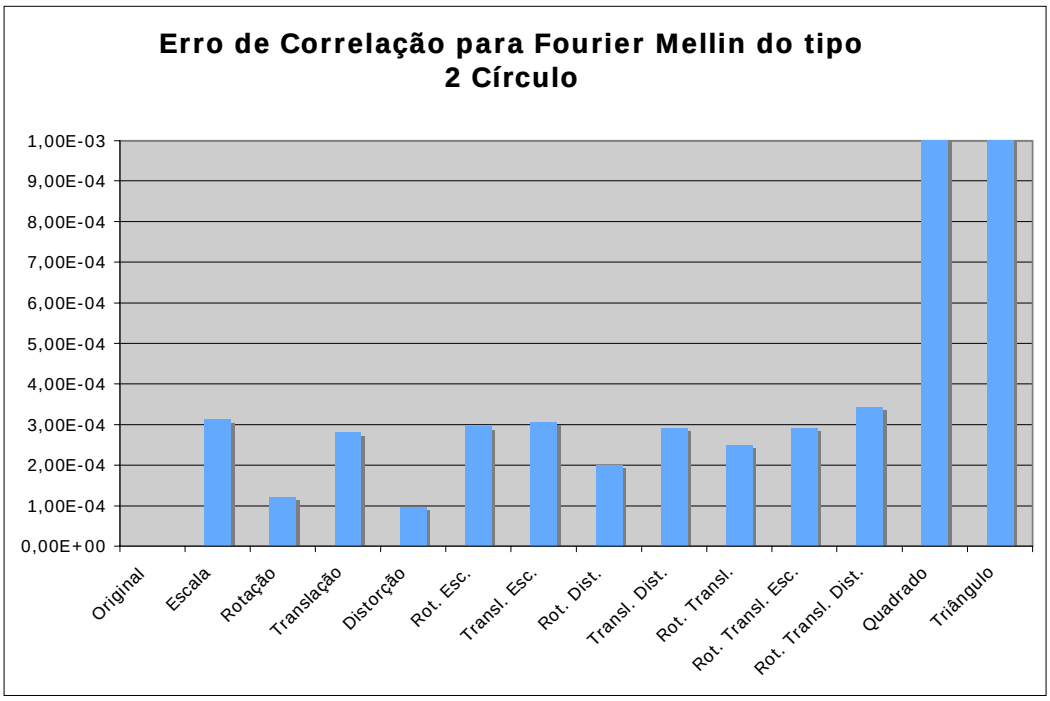

Figura 7-44: Erro de Correlação para Fourier Mellin do tipo 2 – Círculo

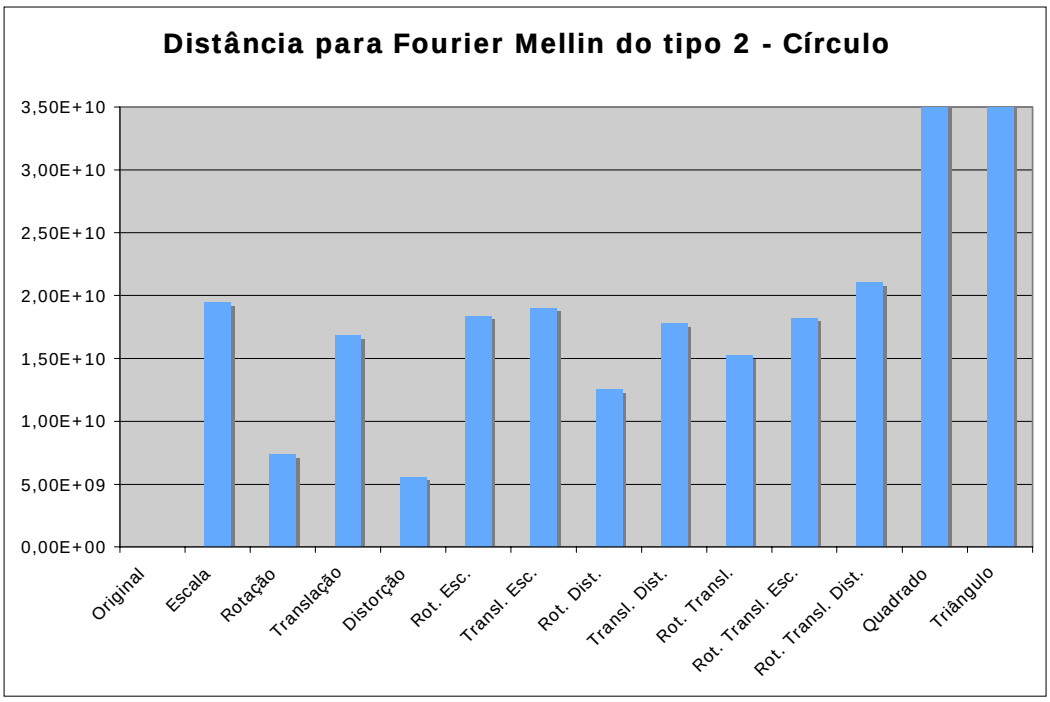

Figura 7-45: Distância para Fourier Mellin do tipo 2 – Círculo

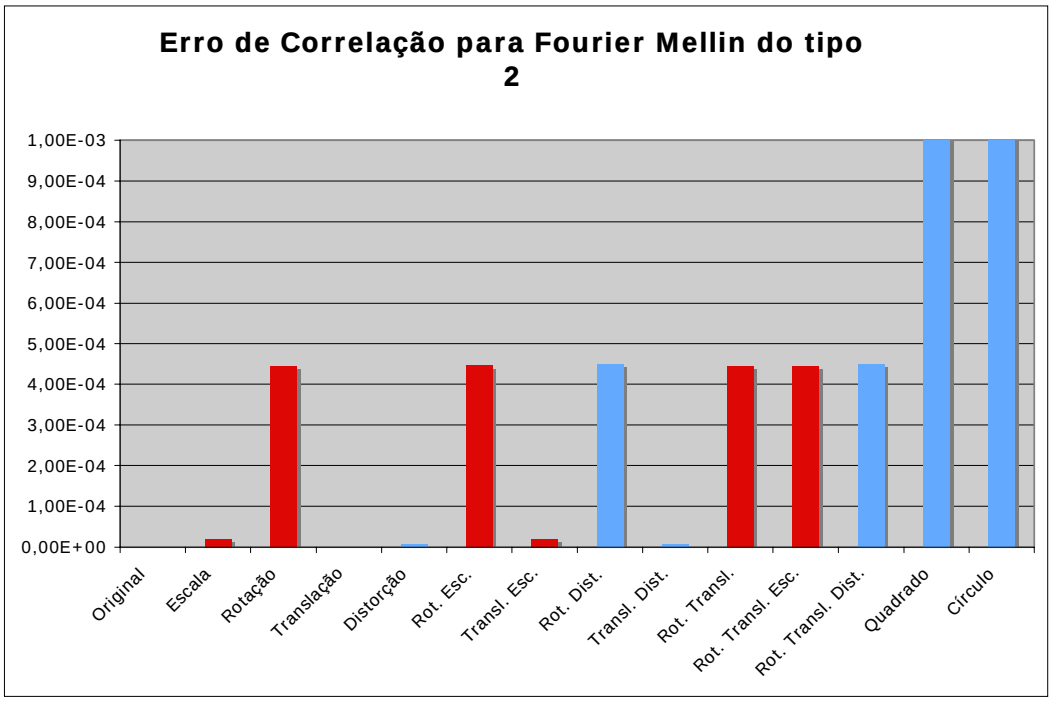

Figura 7-46: Erro de Correlação para Fourier Mellin do tipo 2 – Triângulo

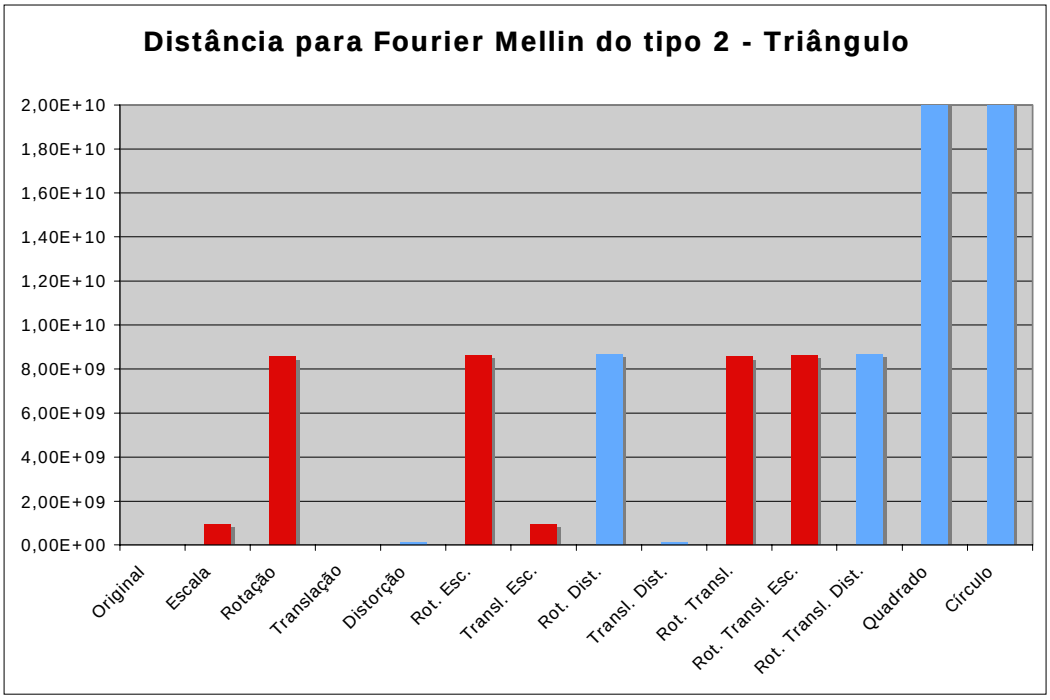

Figura 7-47: Distância para Fourier Mellin do tipo 2 – Triângulo

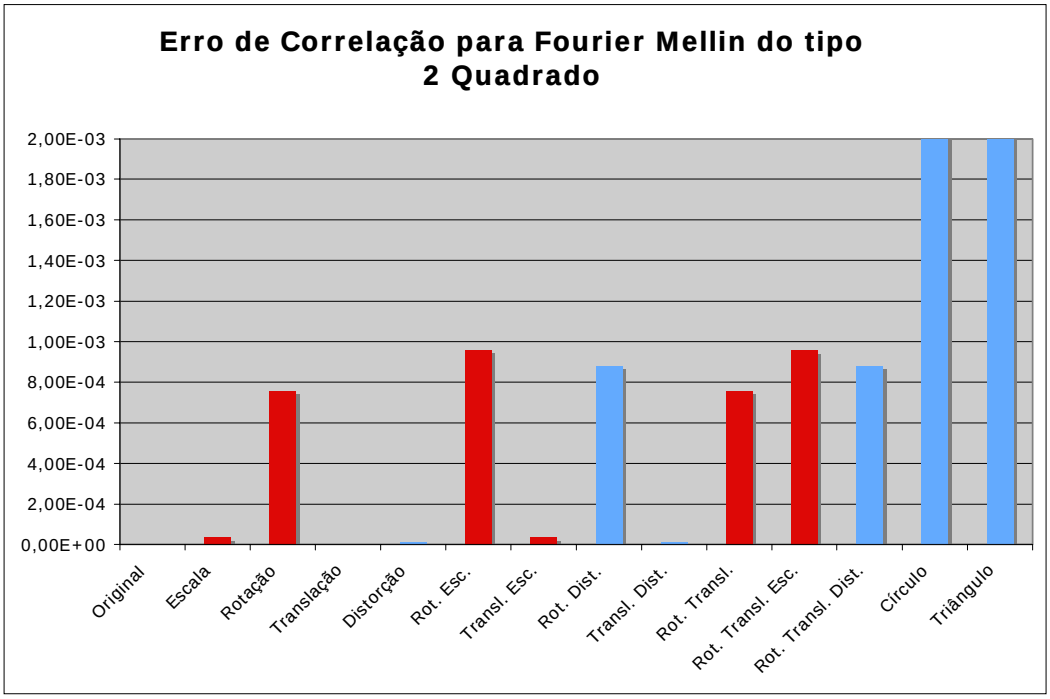

Figura 7-48: Erro de Correlação para Fourier Mellin do tipo 1 – Quadrado

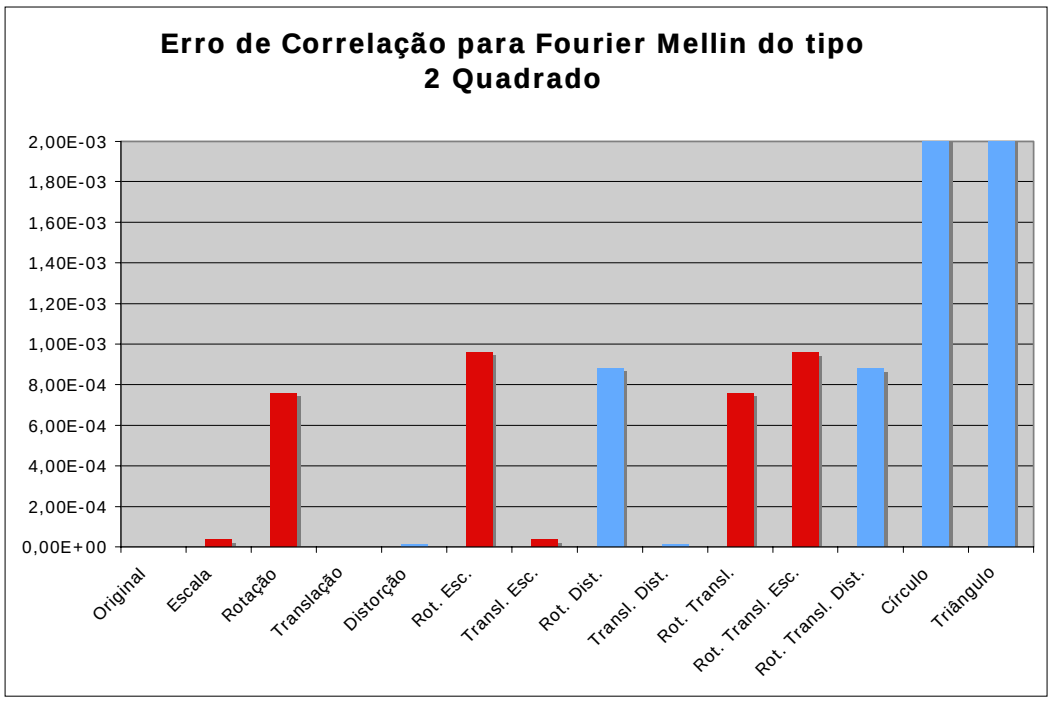

Figura 7-49: Distância para Fourier Mellin do tipo 2 – Quadrado

7.2.6. Conclusões gerais sobre as comparações entre Transformadas

Pode-se verificar visualmente nos experimentos mostrados na seção 7.1 que ao se trabalhar com funções não contínuas, como imagens, as Transformadas estudadas apresentam resultados compatíveis com as invariâncias esperadas e verificadas matematicamente para o domínio contínuo.

Porém, pequenas variações em resultados que deveriam ser idênticos no domínio contínuo foram visualisadas ao se trabalhar no domínio discreto. Estas variações são consequentes não somente do fato das imagens serem quantizadas mas também pelo fato de que algumas transformações (tal como escala) são feitas utilizando técnicas de aproximação como interpolação. Estas pequenas variações observadas mostram que algumas Transformadas possuem grande sensibilidade a determinadas modificações na imagem.

Verificou-se ser adequado utilizar as transformações propostas e a implementação desenvolvida para se comparar imagens que parassaram pelas variações estudadas, sendo que cada uma das transformadas é mais adequada para algumas destas variações.

Por fim, cabe ressaltar que o uso de correlação é superior ao uso de distâncias euclidianas para se fazer a avaliação das comparações, pois o segundo método demostrou não ter conseguido fazer comparações de modo correto para alguns experimentos.

7.3. Navegação utilizando transformações invariáveis – Análise de *Window Growing*

Foram realizados diversos experimentos para se avaliar a aplicação de *Window Growing* de modo a se encontrar uma imagem distante como uma sub-janela de uma imagem atual. Serão apresentados aqui alguns exemplos que caracterizam os resultados verificados. Estes exemplos têm como objetivo mostrar a validade da técnica tanto quanto suas deficiências.

Os experimentos apresentados seguiram as seguintes etapas:

- O robô era posicionado em um local definido e uma imagem era obtida;
- O robô se movimentava para trás em linha reta por uma distância determinada;
- O robô obtinha outra imagem;
- O robô girava 20 vezes aproximadamente $0{,}5^{o}$ e obtinha uma imagem a cada direção;

Para o conjunto de imagens obtidas se aplicou a comparação com a primeira imagem através de *Window Growing* de modo a se analisar se a técnica conseguia determinar a direção correta em relação à imagem de referência e se conseguia encontrar corretamente a sub-janela relacionada à imagem de referência.

A análise da sub-janela encontrada é apresentada com relação às imagens de longe.

Um experimento realizado foi referente às imagens vistas na Figura 7-50. Estas imagens apresentam diferentes visões do robô de uma mesma cena: de perto; de longe; de longe e em ângulo de aproximadamente $10^{o.}$

Pode-se perceber que não há uma centralização correta da imagem à distância em relação à imagem próxima. Isto implica em dificuldades na aplicação do *Window Growing* dado que a técnica assume que a câmera do robô mantém nivelamento ao longo da navegação. O que foi feito neste experimento foi deixar a

câmera um tanto deslocada para cima. Apesar disso, os resultados demonstraram a capacidade do robô em determinar a direção correta para muitas das variações da técnica.

a)  b) 

c) 

Figura 7-50: Diferentes visões do robô: a) Próximo a parede; b) Afastado da parede; c) Afastado e em ângulo;

7.3.1. Imagens de longe – Encontrando a sub-janela

Primeiro são apresentados os resultados relativos à janela de longe da Figura 7-51 à Figura 7-54.

Nas figuras apresentadas tem-se a mesma vista observada sendo processada pelo algorítmo de *Window Growing* utilizando-se diferentes Transformadas e comparações (por correlação e distância euclidiana).

O enquadramento dado pelo quadrado vermelho representa a janela obtida como aquela mais próxima da Figura 7-50 – a.

Na Figura 7-51 e na Figura 7-52 é observado que os enquadramentos encontrados correspodem, apesar de não perfeitamente, a aproximações razoáveis da vista procurada. Já na Figura 7-53 e na Figura 7-54, os enquadramentos estão visivelmente incorretos.

É interessante de se perceber que os enquadramentos errados são relativos a comparações feitas utilizando-se distância euclidiana que já havia sido apontada nos experimentos da seção 7.2 como inferior à correlação, confirmando a observação.

Figura 7-51: Janela obtida por: Correlação, correlação da Mellin do tipo 2, correlação da Mellin do tipo 2 e correlação da Fourier Mellin do tipo 1 e 2

Figura 7-52: Janela obtida por: Correlação Mellin do tipo 1

Figura 7-53: Janela obtida por: Distância da Fourier Mellin do tipo 2 e distância da Mellin do tipo 1

Figura 7-54: Janela obtida por: Distância da Fourier Mellin do tipo 1 e distância da Mellin do tipo 2

Na próxima seção será verificado quantitativamente os resultados do algorítmo *Window Growing* para as várias imagens obtidas.

7.3.2. Encontrando a direção correta

Serão vistos resultados utilizando-se correlação e distância euclidiana como funções de similaridade para as comparações;

Tal como foi feito para os experimentos da seção 7.2, os valores relativos à correlação são do erro de correlação, desta forma o entendimento dos gráficos relativos à distância e correlação seguem o mesmo padrão, quanto menor o valor, melhor a comparação.

Espera-se que os gráficos mostrem menores valores para os ângulos próximos de 0 e maiores para ângulos superiores.

Uma série de testes foram feitos ao longo do projeto. Através destes a das experiências a serem apresentadas pode-se verificar que os melhores resultados foram obtidos com a Transformada de Fourier Mellin do tipo 1 e da do tipo 2.

A comparação por correlação vista na seção 7.3.2.1 também mostrou ser bastante adequada.

Porém, as comparações globais utilizadas têm algumas limitações e são extremamente sensíveis. As limitações encontradas são:

- O método não funciona para imagens que possuam objetos a diferentes distâncias do robô e que impliquem em nenhuma imagem centralizada ser parecida com a imagem de referência;
- Para o caso da câmera não estar bem nivelada, a imagem de referência pode não aparecer centralizada quando vista à distância. O método é sensível ao nivelamento da câmera;
- O método não funciona caso ocorram oclusões de áreas na imagem;
- O método não funciona caso apareçam informações visuais diferentes das verificadas na imagem de referência dentro da área centralizada que seria referente à imagem de referência quando o robô está à distância;

Da Figura 7-55 a Figura 7-63 são apresentados os valores obtidos com *Window Growing* em relação aos ângulos em que as imagens foram obtidas. Isto é feito para os diversos métodos propostos no capítulo 2. Alguns comentários aqui descritos podem ser verificados nestes experimentos.

7.3.2.1. Correlação

O gráfico da Figura 7-55 aponta menores erros de correlação para ângulos próximos a 0, com o erro subindo ao se afastar do mesmo. Isto vai de acordo com

o esperado e mostra que para este caso a correção de direção seria feita corretamente.

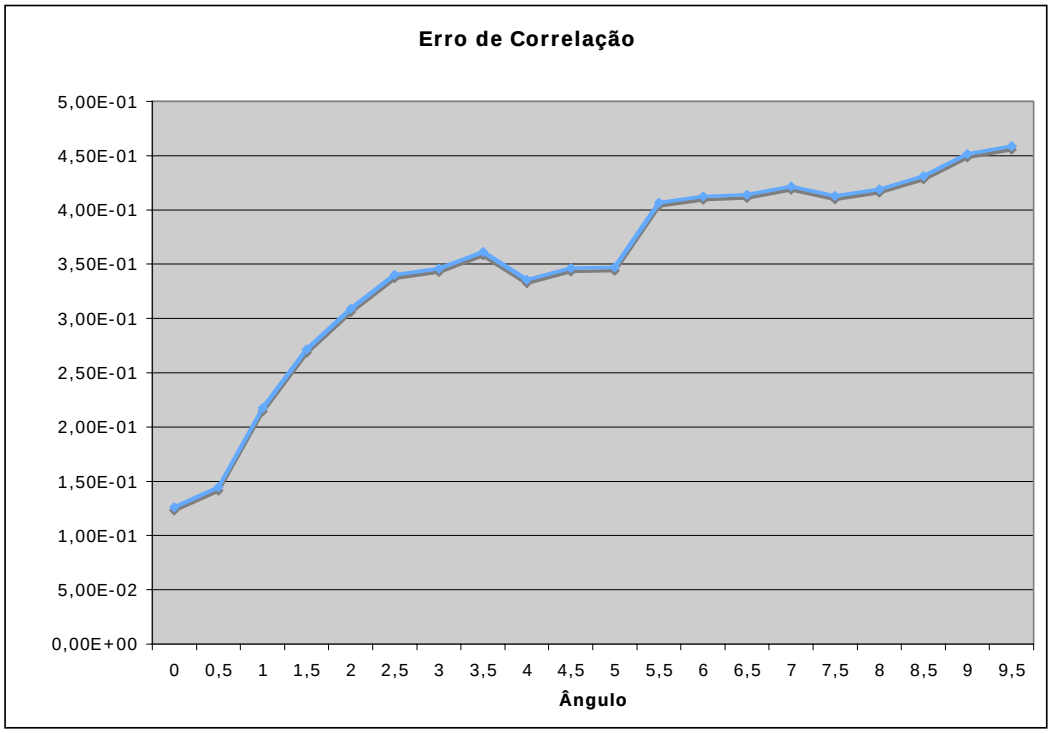

Figura 7-55: Erro de Correlação

É interessante verificar que o valor de erro sobe de modo quase que constante ao se afastar da direção correta. Esta subida deixa de ser constante por volta do ângulo 3.5^{o} e, quanto maior o afastamento, menor a discrepância do erro de correlação medido entre um ângulo e outro.

7.3.2.2. Mellin do tipo 1

Ao verificar o gráfico relativo à correlação (Figura 7-56), percebe-se que realmente o erro medido é inferior para ângulos próximos de 0, apesar de o menor valor não estar exatamente em 0. Porém, quando o robô se afasta do ângulo correto, os resultados podem não mais seguir uma inclinação constante, dado que as janelas encontradas através do algorítmo *Window Growing* podem estar completamente erradas. O importante é verificar que mesmo assim os melhores resultados são próximos do ângulo correto.

Entretanto, o robô não encontra a direção corretamente com comparações usando distância euclidiana (Figura 7-57). Ao se utilizar a distância euclidiana para se fazer as comparações, o erro medido não é inferior para ângulos próximos de 0 nem cresce ao se afastar da posição correta.

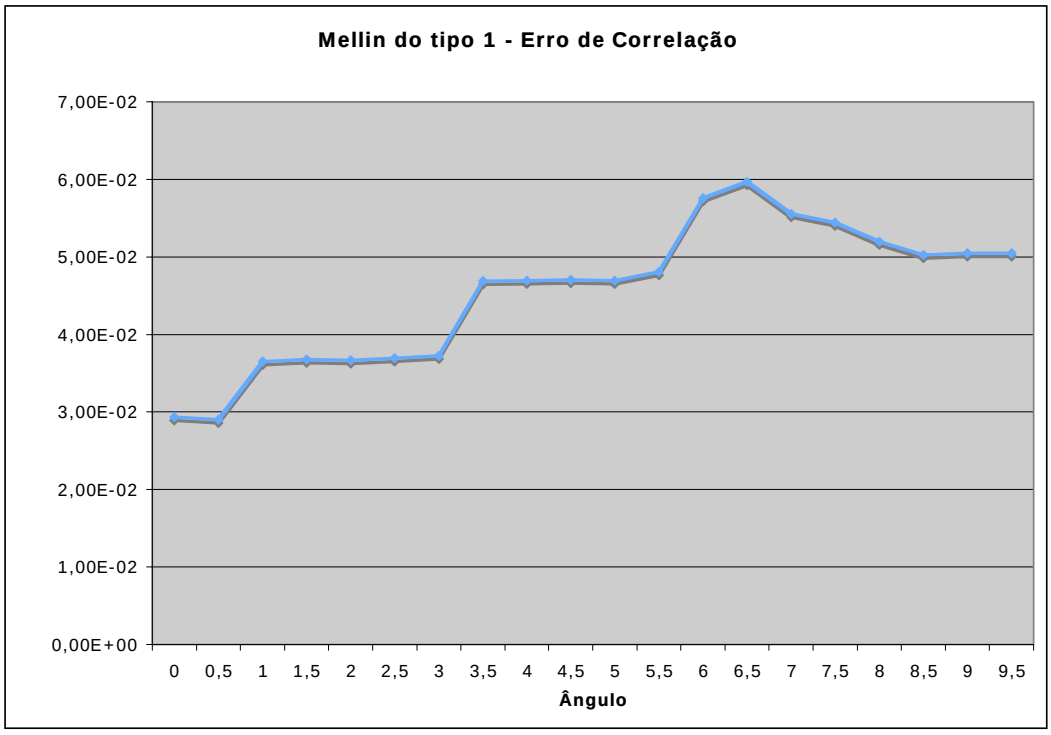

Figura 7-56: Erro de Correlação para Mellin do tipo 1

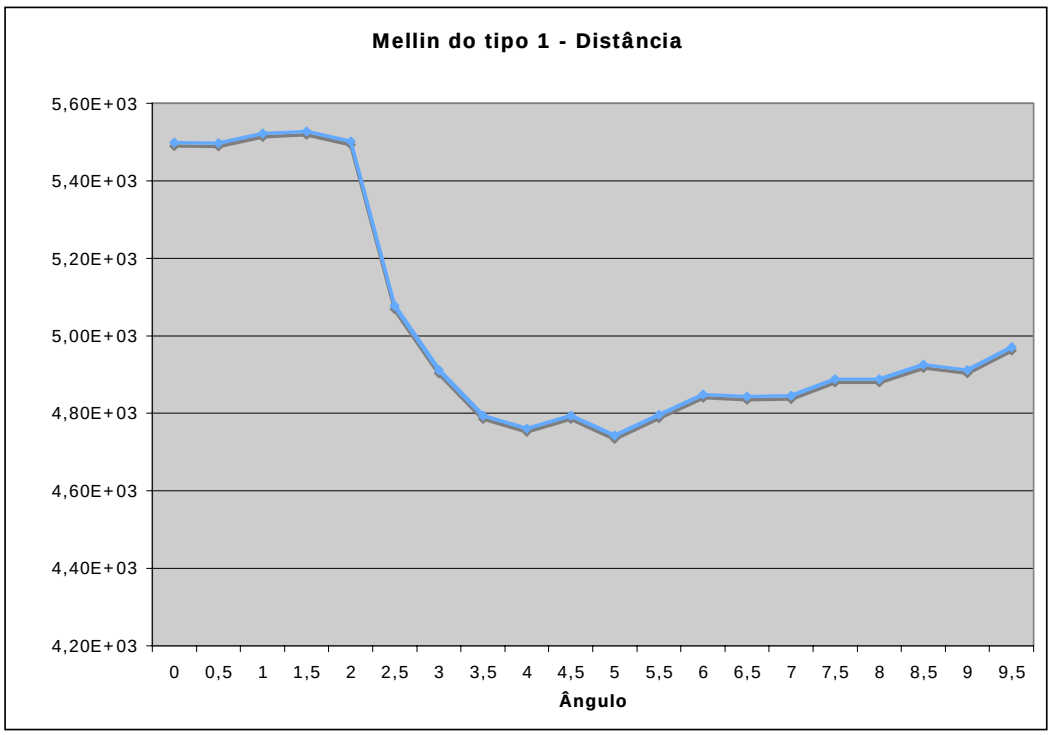

Figura 7-57: Distância para Mellin do tipo 1

7.3.2.3. Mellin do tipo 2

Os gráficos relativos à transformada de Mellin do tipo 2 podem ser vistos na Figura 7-58 e na Figura 7-59.

O uso da Transformada de Mellin do tipo 2 apresentou resultado similar a Transformada de Mellin do tipo 1, pois o erro de correlação é inferior para quando próximo da direção correta e o método usando comparação usando distância euclidiana falha em apontar a direção correta do robô. Porém, o crescimento do erro ao se afastar da posição correta é mais perceptívo para a Transformada de Mellin do tipo 1.

Novamente pode-se verificar que a partir de determinada direção, o crescimento deixa de ser constante. Isto acontece pelo motivo já citado de que a partir de determinado momento, o algorítmo *Window Growing* não deve mais encontrar a janela correta.

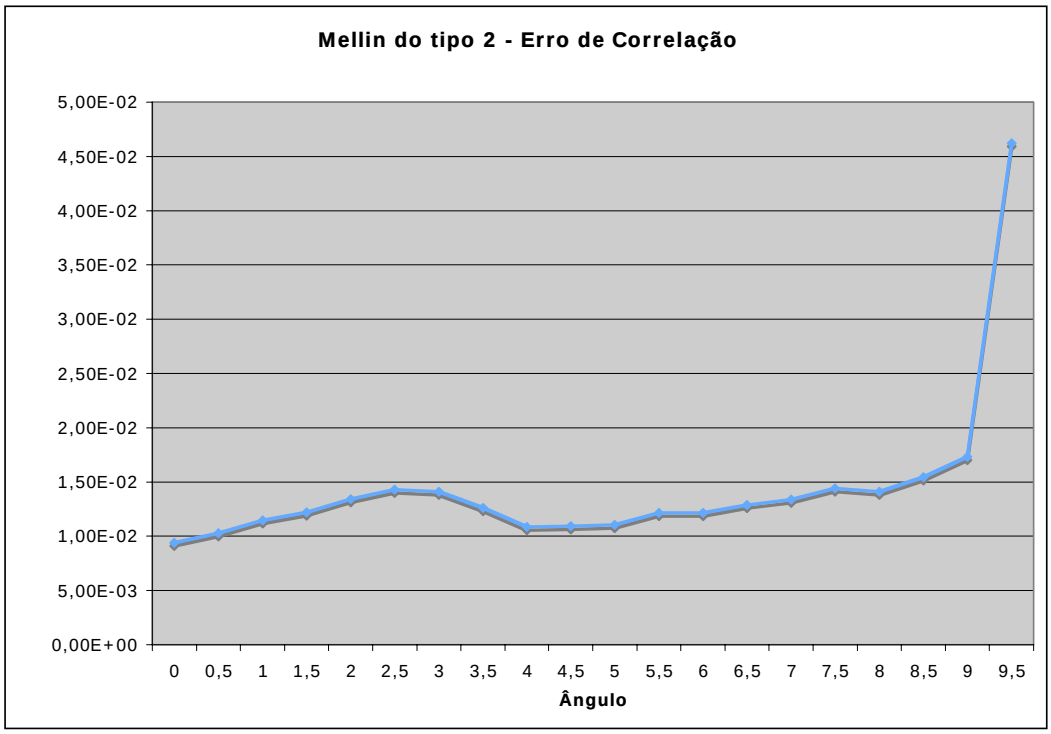

Figura 7-58: Erro de Correlação para Mellin do tipo 2

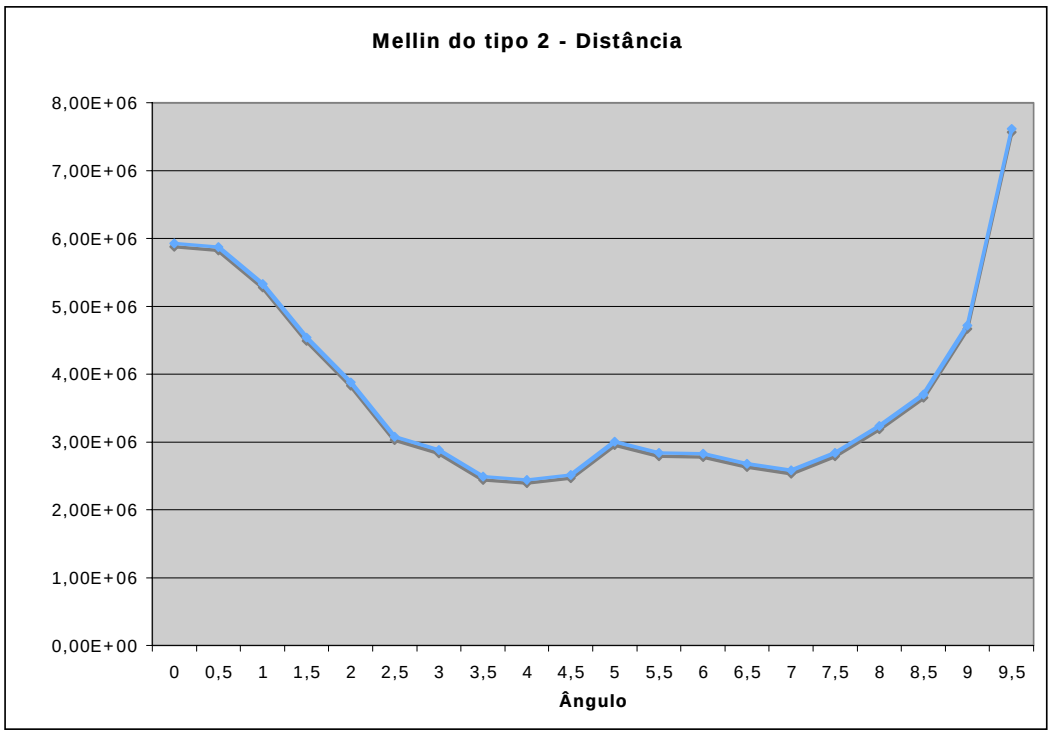

Figura 7-59: Distância para Mellin do tipo 2

7.3.2.4. Fourier Mellin do tipo 1

Para o caso específico deste experimento realizado com a Transformada de Fourier Mellin do tipo 1, a curva associada a comparação feita com distância euclidiana (Figura 7-61) se mostra bastante adequada ao comportamento esperado e já discutido (erro menor para quando próximo da direção correta e crescendo quando se distancia desta). Entretanto, este caso é uma excessão dado que o uso de distância euclidiana não costuma apresentar sucesso normalmente.

Para o uso de comparação por correlação (Figura 7-60), novamente foram obtido menores erros quando próximo a direção correta, com erro crescendo ao se afastar da mesma, até o momento em que a janela obtida por *Window Growing* deixa de estar correta.

Pode-se verificar que as janelas obtidas por *Window Growing* costumam estar corretas para uma gama maior de direções quando a Transformada de Fourier Mellin do tipo 1 é utilizada.

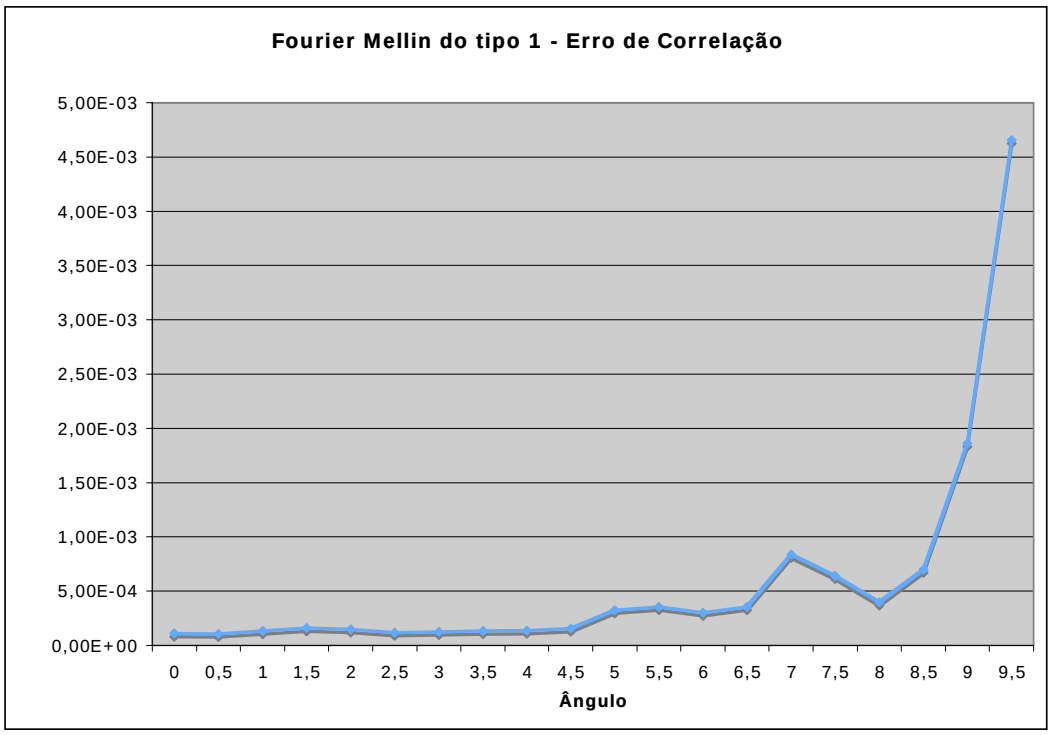

Figura 7-60: Erro de Correlação para Fourier Mellin do tipo 1

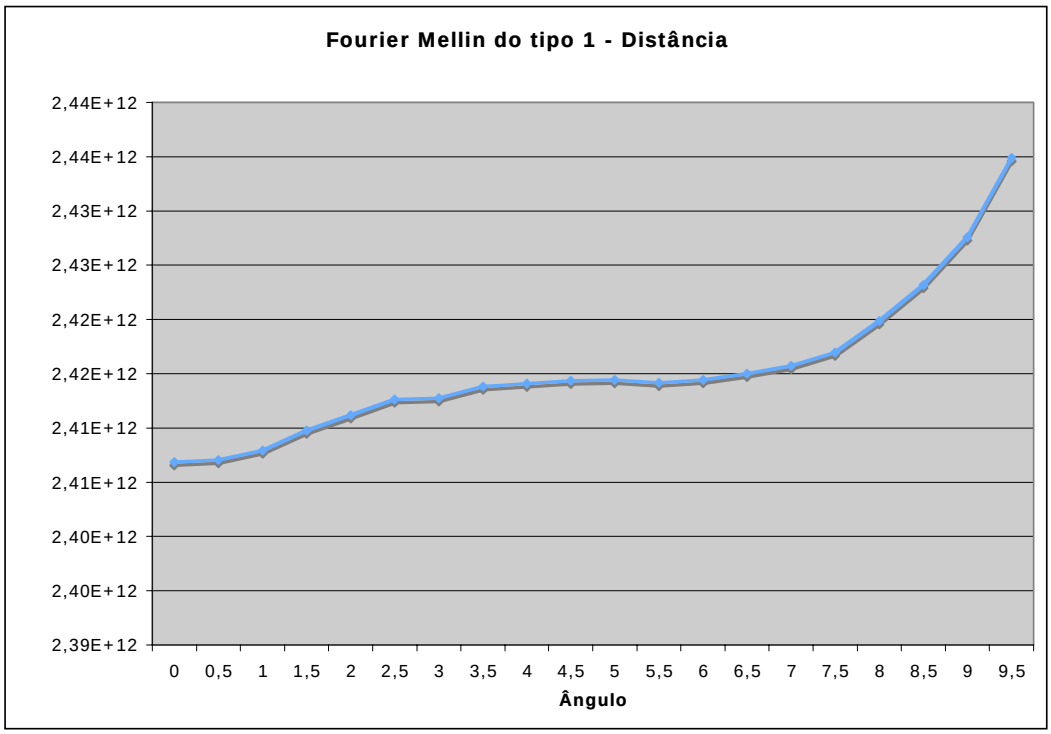

Figura 7-61: Distância para Fourier Mellin do tipo 1

7.3.2.5. Fourier Mellin do tipo 2

Este caso novamente apresenta uma excessão dado que a curva associada ao uso de distância euclidiana para as comparações (Figura 7-63) demonstra comportamento esperado (erro menor para quando próximo da direção correta e crescendo quando se distancia desta).

A curva associada à comparações por correlação (Figura 7-62) também demonstra o comportamento esperado dado que o erro cresce quando se afasta da posição correta.

Tal como com a Transformada de Fourier Mellin do tipo 1, as janelas obtidas por *Window Growing* costumam estar corretas para uma gama maior de direções.

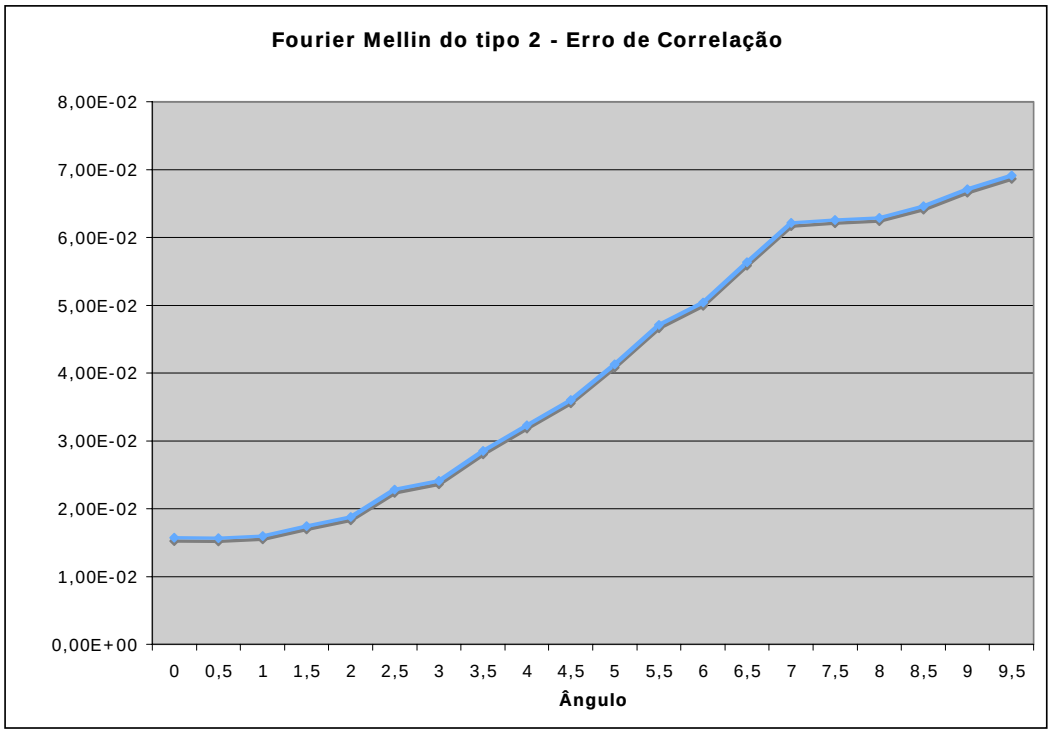

Figura 7-62: Erro de Correlação para Fourier Mellin do tipo 2

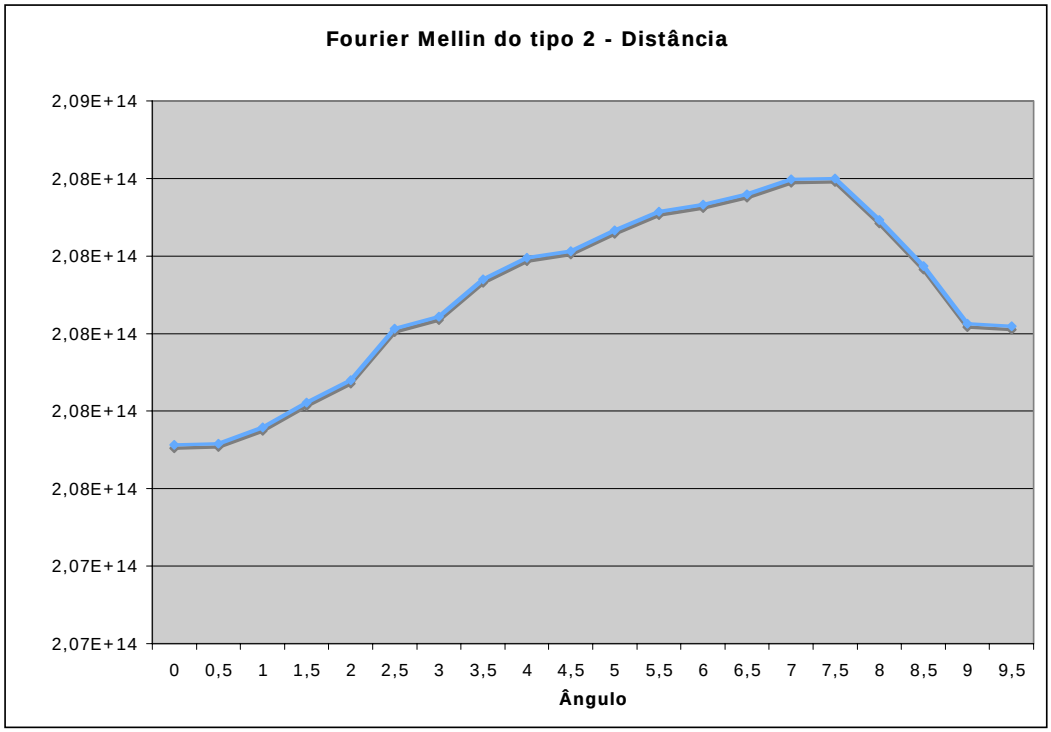

Figura 7-63: Distância para Fourier Mellin do tipo 2

Ao longo dos experimentos feitos, constatou-se que os melhores resultados são verificados para os seguintes métodos experimentados:

- Correlação;
- Fourier Mellin do tipo 1;
- Fourier Mellin do tipo 2;

Os métodos foram apresentados em ordem crescente em relação ao número de sucessos observados ao longo do desenvolvimento deste projeto. De um modo geral estes 3 métodos apresentaram maior estabilidade para vários casos testados.

7.4. SIFT

Descritores SIFT são usados para auxiliar a navegação e corrigir a direção do robô em diversos momentos. Têm como finalidade encontrar pontos característicos entre diferentes visões do robô, para perto e longe de um nó, e para diferentes ângulos para um nó.

A busca dos pontos foi feita seguindo:

- Extração dos descritores SIFT para 4 intervalos e 8 oitavas;

- Aplicação da transformação de Hough;
- Aplicação de RANSAC e matriz de pesos W;

Os parâmetros utilizadados seguiram os padrões indicados ao longo do presente trabalho.

De modo a testar a eficiência da aplicação dos descritores SIFT foram feitos diversos experimentos em que o robô obtinha 3 imagens para diferentes visões de uma mesma cena e, então, estas imagens eram comparadas utilizando-se da transformação SIFT. As visões testadas foram:

- Em uma posição escolhida;
- Afastado desta posição: O robô movia-se para trás por uma distância determinada;
- Na mesma posição afastada, o robô girava de aproximadamente 20^{o} e obtinha a última imagem;

Serão apresentadas aqui alguns dos experimentos feitos. Os experimentos apresentados tanto mostram em que situações a técnica se comporta melhor, aquelas em que não foi possível encontrar pontos em comum entre imagens e aquelas em que foi possível encontrar pontos em comum mas que o enquadramento de uma imagem na outra aparenta estar incorreto apesar de não estar.

As imagens são apresentadas em pares da Figura 7-64 à Figura 7-87. O primeiro par de cada experimento compara as imagens para perto e longe. O segundo par compara as visões de perto e de longe em ângulo de 20^{o}. Em todas as imagens são mostrados os pontos em comum entre elas e como fica o enquadramento da imagem de perto quando transformado com a matriz *T* encontrada.

As imagens utilizaram resolução de 166 por 144 *pixels*.

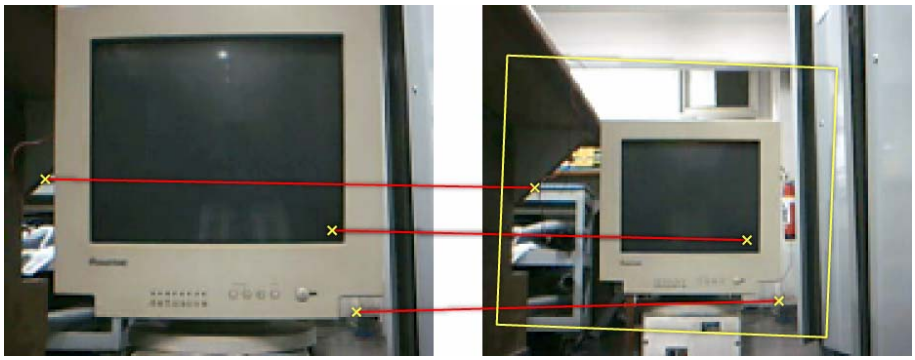

Figura 7-64: Visão de perto e visão de longe

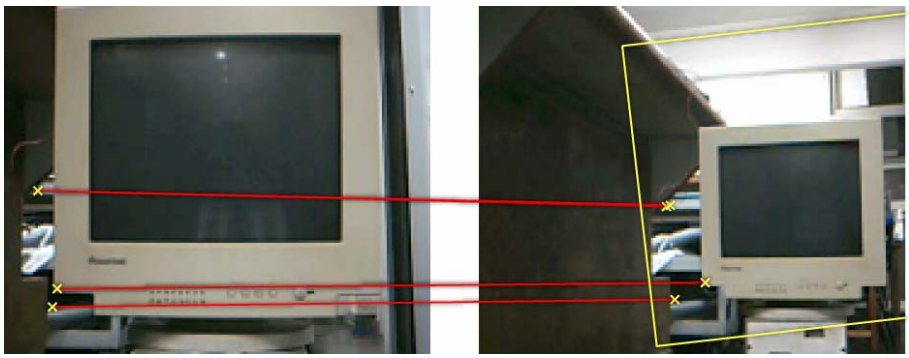

Figura 7-65: Visão de perto e visão de longe em ângulo

Neste primeiro experimento pode-se verificar que os pontos encontrados foram corretos.

Na Figura 7-64, o enquadramento seguiu pontos relacionados a objetos no fundo da imagem e não à superfície da tela do computador porque a maioria dos pontos encontrados são relativos a objetos do fundo. Perceba que o enquadramento está correto.

Figura 7-65 o enquadramento parece estranho porque alguns dos pontos encontrados são relativos à tela do computador que estava mais próxima do robô e outros relativos ao fundo que estava mais longe. Perceba que objetos a diferentes distâncias do robô representam diferentes matrizes T por conta das diferentes perspectivas associadas aos mesmos.

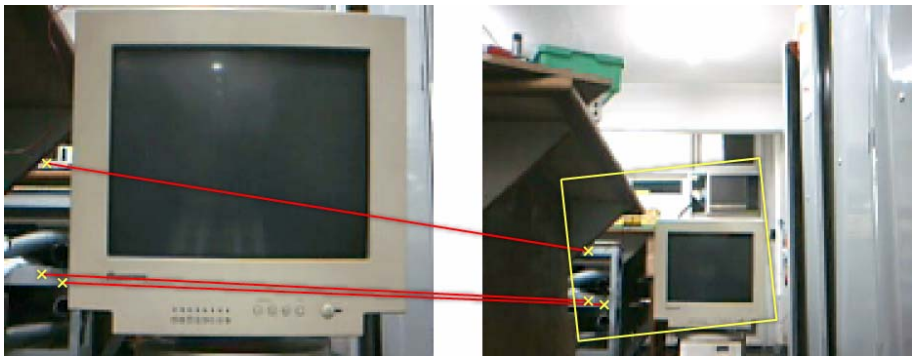

Figura 7-66: Visão de perto e visão de longe

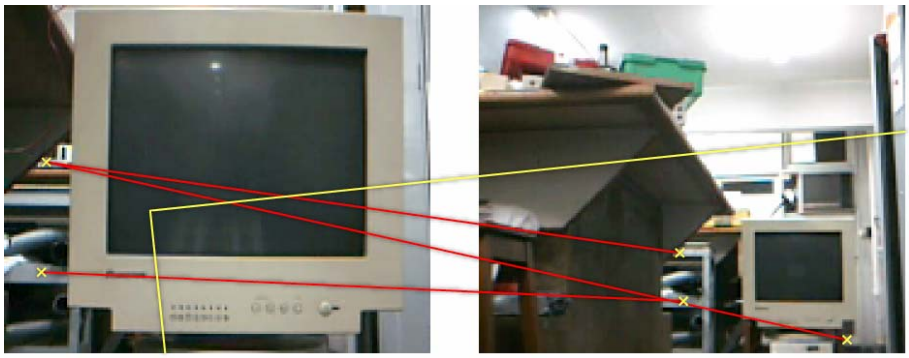

Figura 7-67: Visão de perto e visão de longe em ângulo

A Figura 7-66 obteve os pontos corretamente, com pequeno erro. O enquadramento segue o fundo e só não foi melhor porque o número de pontos encontrados foi pequeno e, portanto, qualquer erro pequeno implica em uma matriz de transformação *T* não muito exata.

A Figura 7-67 apresentou alguns pontos corretos e outros não. O enquadramento não ficou correto.

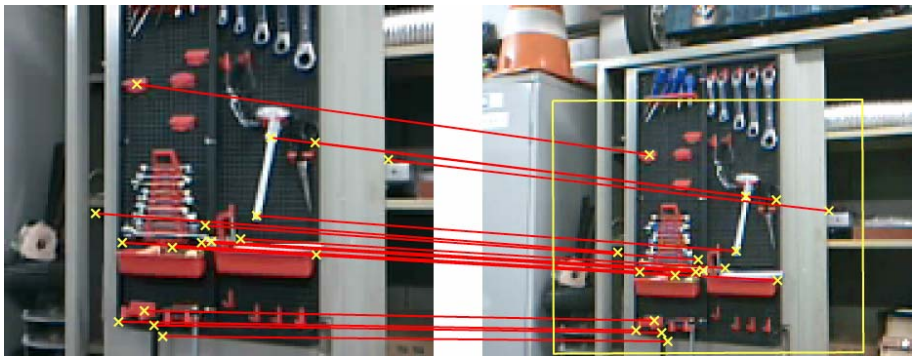

Figura 7-68: Visão de perto e visão de longe

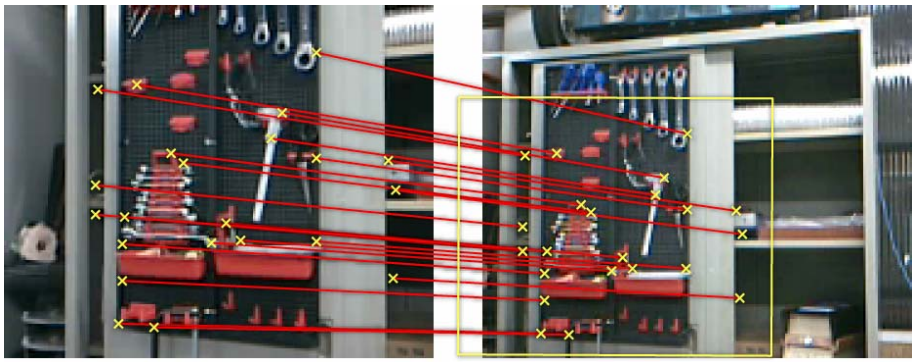

Figura 7-69: Visão de perto e visão de longe em ângulo

As duas figuras apresentadas aqui obtiveram bons resultados por dois motivos. Um deles é que as imagens possuíam grande quantidade de informação e um grande número de descritores SIFT. O outro é que todos os pontos eram referentes a objetos a mesma distância do robô.

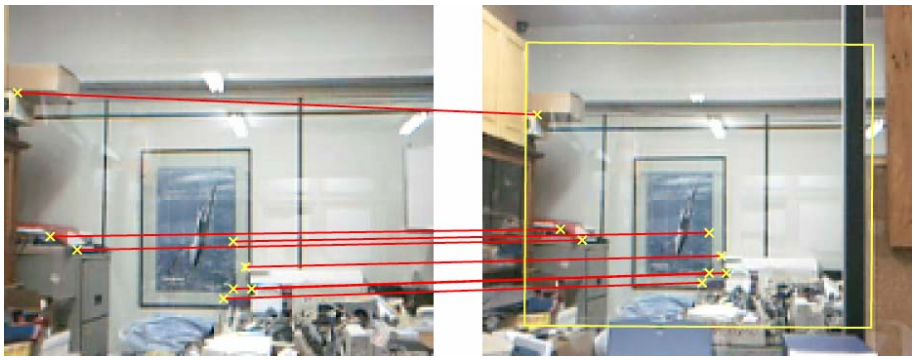

Figura 7-70: Visão de perto e visão de longe

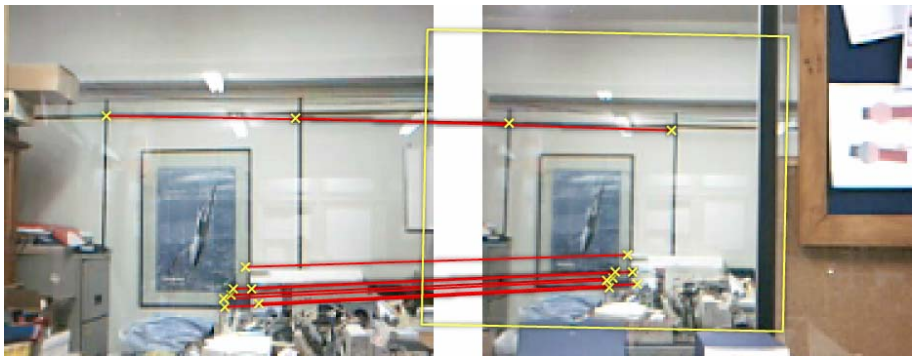

Figura 7-71: Visão de perto e visão de longe em ângulo

Este experimento é interessante pois mostra que apesar de haver oclusão nas duas figuras apresentadas, como os descritores são locais, o resultado é correto. Todos os pontos encontrados foram de acordo, tal como o enquadramento.

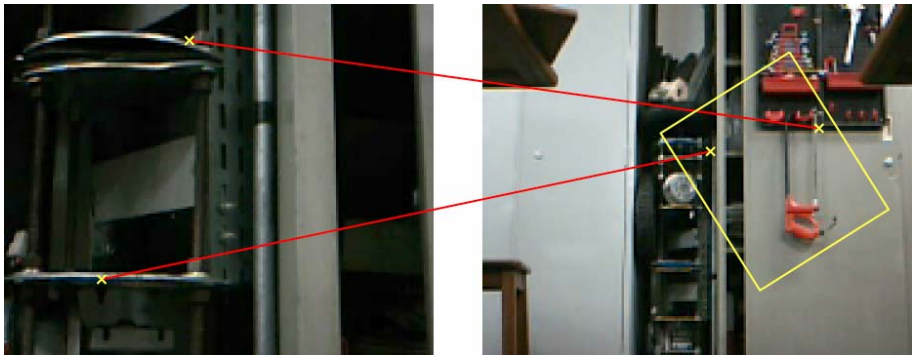

Figura 7-72: Visão de perto e visão de longe

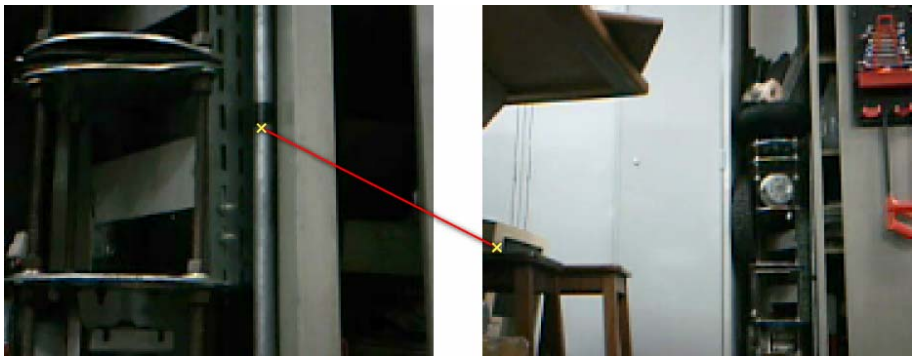

Figura 7-73: Visão de perto e visão de longe em ângulo

Este exemplo mostra visões muito diferentes porque quando de longe, a imagem obtida é muito distante da imagem de perto. Existem poucos elementos que facilitassem encontrar pontos em comum. Além de tudo, a resolução das imagens é muito baixa, provavelmente o método funcionaria para resoluções maiores.

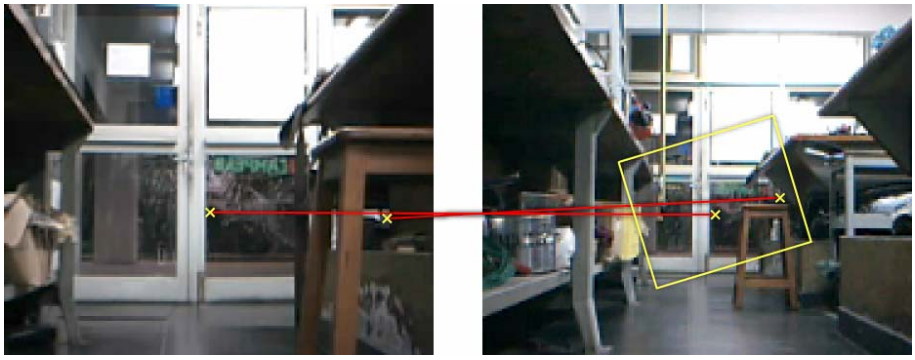

Figura 7-74: Visão de perto e visão de longe

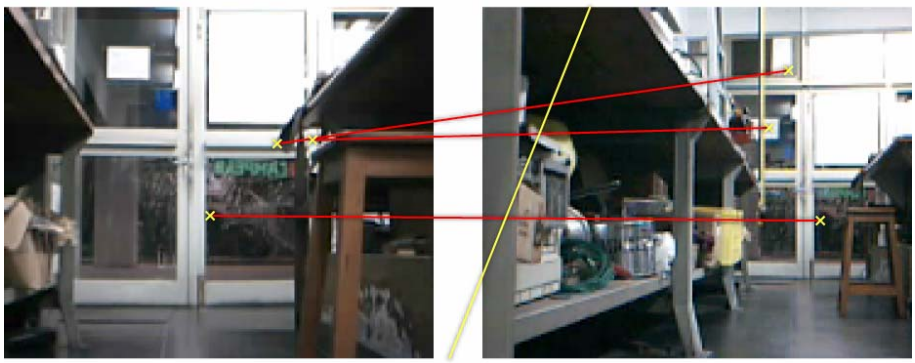

Figura 7-75: Visão de perto e visão de longe em ângulo

As imagens foram obtidas a grandes distâncias se for levado em consideração a resolução utilizada nas imagens. Deste modo, alguns pontos são encontrados corretamente, mas a maioria não e a técnica falha.

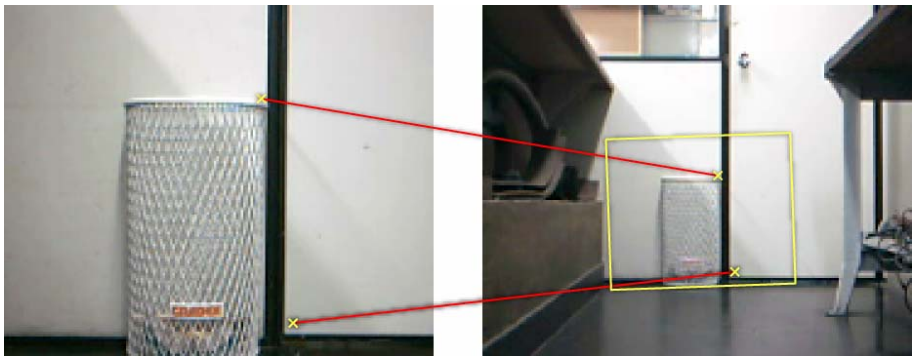

Figura 7-76: Visão de perto e visão de longe

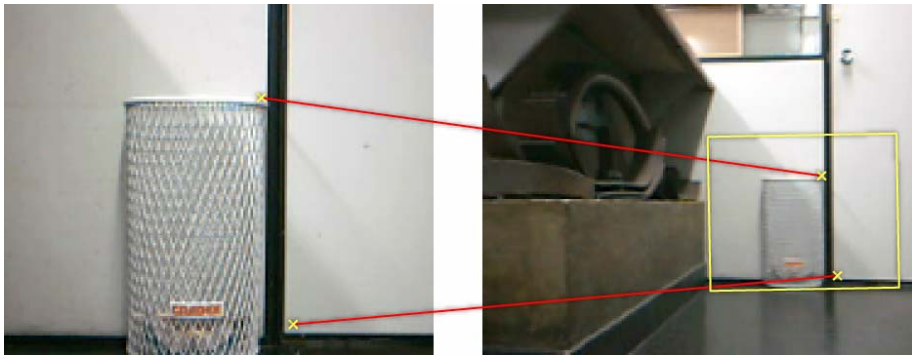

Figura 7-77: Visão de perto e visão de longe em ângulo

Este também é um bom exemplo, e um exemplo fácil para se aplicar a técnica, porque todos os elementos observados estão a uma mesma distância do robô.

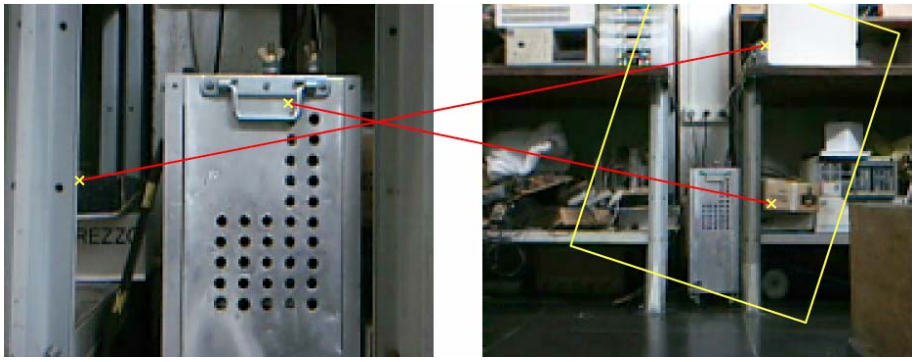

Figura 7-78: Visão de perto e visão de longe

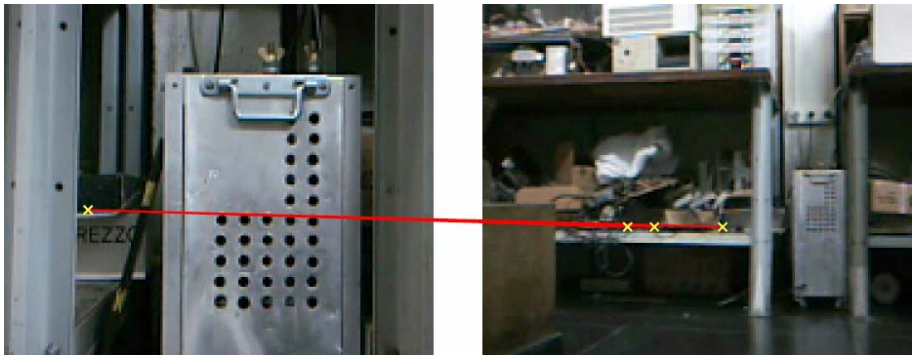

Figura 7-79: Visão de perto e visão de longe em ângulo

Este é mais um exemplo em que a distância dos objetos observados variou muito e a técnica não encontrou os pontos corretamente. Imagens com maior resolução resolveriam este problema.

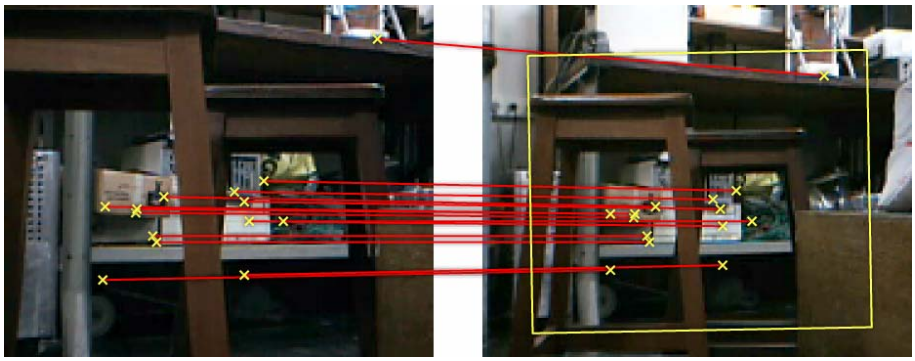

Figura 7-80: Visão de perto e visão de longe

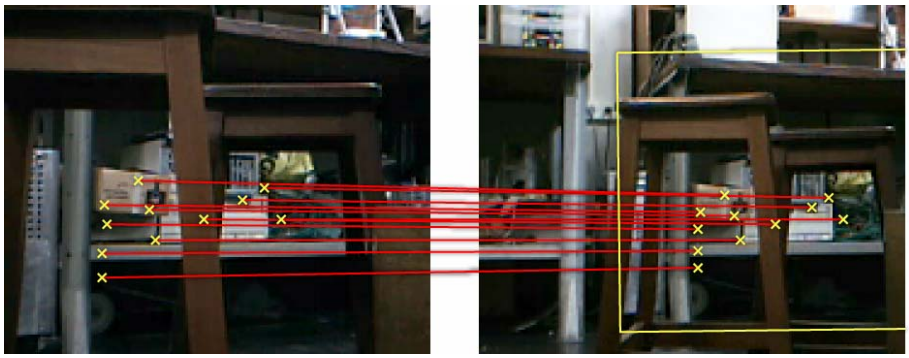

Figura 7-81: Visão de perto e visão de longe em ângulo

Este é um exemplo interessante porque mostra objetos a diferentes distâncias do robô sendo observados. Mesmo assim, todos os pontos encontrados em comum foram relativos a regiões há uma mesma distância do robô. É interessante notar que quando este tipo de coisa acontece, o procedimento de transformação *Hough* faz com que pontos que estão a uma mesma distância sejam escolhidos. Normalmente, aquela distância que apresenta o maior número de pontos é privilegiada.

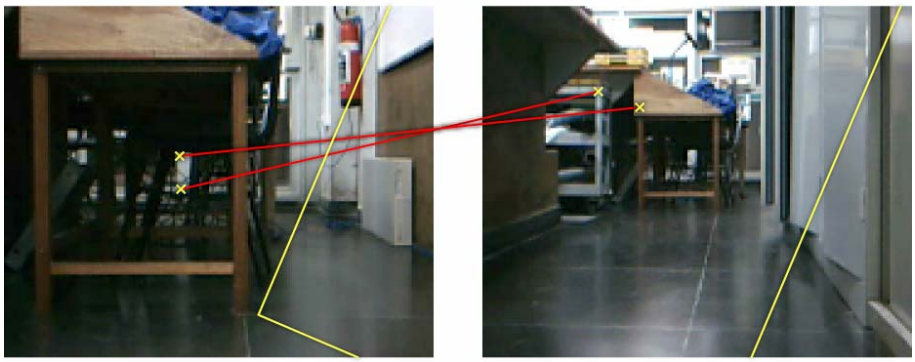

Figura 7-82: Visão de perto e visão de longe

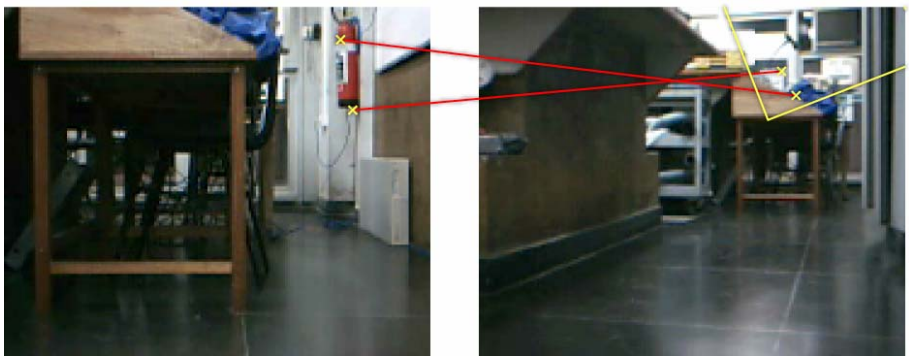

Figura 7-83: Visão de perto e visão de longe em ângulo

Apesar da técnica não ter sido eficaz para as imagens acima, quando se trabalhou com distâncias um pouco menores nas imagens abaixo o procedimento funcionou muito bem (exceto para a Figura 7-85).

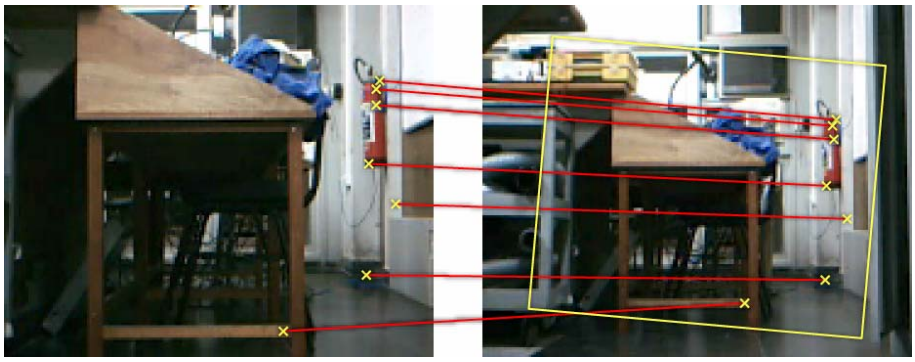

Figura 7-84: Visão de perto e visão de longe

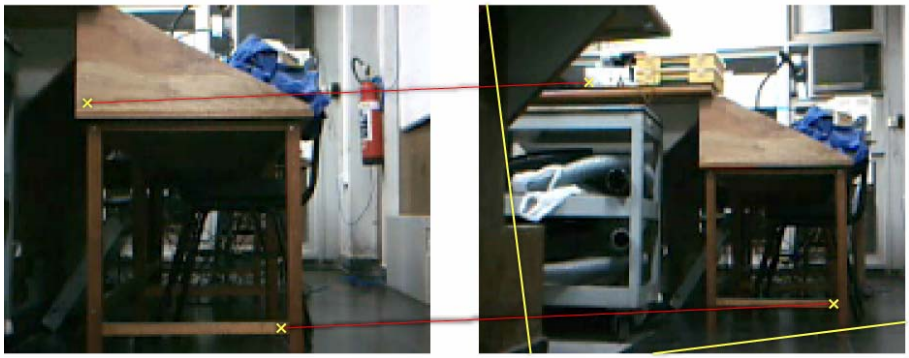

Figura 7-85: Visão de perto e visão de longe em ângulo

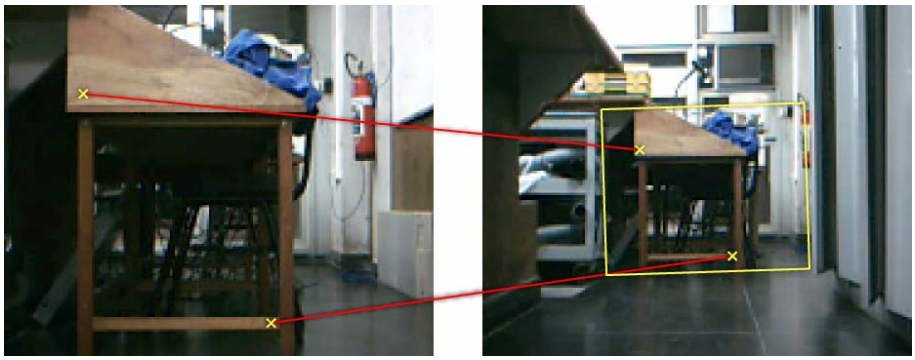

Figura 7-86: Visão de perto e visão de longe

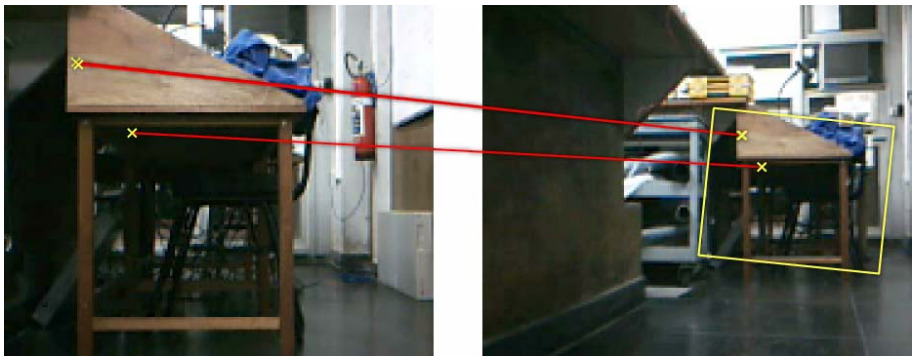

Figura 7-87: Visão de perto e visão de longe em ângulo

Como já observado, o procedimento funcionou muito bem em 3 destas últimas 4 figuras (a exceção é a Figura 7-85).

Pode-se concluir dos resultados observados:

- A técnica SIFT consegue encontrar dentro de algumas limitações vistas pontos em comum entre imagens, mesmo quando as seguintes condições são apresentadas: mudanças em escala, rotação, posição e pequenas mudanças na perspectiva e iluminação;
- Imagens obtidas a distâncias superiores a aproximadamente 3 metros, para as quais os objetos sofriam grandes variações de escala, podiam não ter seus pontos corretamente correlacionados. Este problema poderia ser resolvido trabalhando-se com resoluções superiores a utilizada;
- Os enquadramentos calculados são aproximações melhores dos que as encontradas utilizando-se a técnica de *Window Growing* e podem ser utilizados em operações de navegação, como será visto em experimento na seção 7.7;

7.5. Encontrando pontos de controle para montar imagens panorâmicas

Ao longo do projeto também foram feitos diversos outros experimentos de modo a se chegar a uma boa implementação da geração de panorâmicas. Os testes aqui apresentados tem como finalidade mostrar os resultados da aplicação das técnicas apresentadas no presente documento e para tal cobre somente uma parcela dos experimentos realizados.

7.5.1.
Buscando pontos correlatos entre duas imagens

Para se avaliar a detecção e casamento dos pontos de controle por correlação cruzada e as técnicas propostas no presente trabalho, aplicou-se a busca por pontos em comum em uma seqüência de 20 imagens obtidas pelo robô cobrindo uma área de 360^{o} do laboratório em que os experimentos foram feitos. Foram estudados:

- Limiar de correlação utilizado para se eliminar pontos incorretos;
- Tamanho de blocos utilizados para se fazer a procura;
- Eliminação de pontos por amostragem;
- Aplicação de RANSAC e ajuste de matriz de pesos W;
- Descarte de pontos inconsistentes por má correlação;

Foram feitos dois tipos de experimentos. No primeiro deles avaliou-se o número de pontos corretos encontrados entre as duas primeiras imagens da seqüência obtida pelo robô. O segundo tipo de experimento foi a avaliação do erro médio para os pontos encontrados para as 20 imagens da seqüência. O erro médio para cada par de imagens foi calculado como apresentado na seção 4.4.

Os resultados serão apresentados de acordo com o seguinte padrão:

- Tabela com resultados: Esta tabela apresenta os seguintes campos:
  - Número total de pontos: Número de pontos encontrados em comum entre a primeira e a segunda imagem da seqüência de 20 imagens;
  - Pontos corretos: Número de pontos encontrados corretamente para essas duas primeiras imagens;
  - Percentual de pontos corretos: Calculado a partir dos valores relativos aos dois tópicos anteriores;
  - Erro médio: Calculado para os pontos encontrados entre os pares consecutivos da seqüência de 20 imagens utilizadas;
  - Número de pontos em média: A média do número de pontos encontrados para os pares consecutivos da seqüência de 20 imagens;
- Gráfico do percentual de pontos corretos encontrados:;

- Gráfico do erro médio encontrado;
- Figuras apresentando o casamento de pontos entre duas imagens: Feito para as duas primeiras imagens da seqüência;
- Um experimento de geração de panorâmica (geração de mosaico automática) dentro dos parâmetros e técnicas que estão sendo avaliadas;

Cabe ressaltar que as imagens panorâmicas geradas não são corretas para quase todos os experimentos devido ao fato de que as técnicas utilizadas para eliminação de pontos incorretos que estão sendo avaliadas devem ser utilizadas em conjunto para que tudo funcione perfeitamente.

Os resultados são apresentados da Figura 7-88 à Figura 7-112 e da Tabela 7-4 à Tabela 7-8.

7.5.1.1. Eliminação por limiar de correlação – blocos 16 x 16

Características do experimento:

- O tamanho dos blocos utilizados foi de 16 por 16;
- Transformação Procrustes;

Tabela 7-4: Eliminação por limiar de correlação – blocos 16 x 16

Limiar de correlação	Número total de pontos	Pontos Corretos	Percentual Ptos. Corretos	Erro médio	Número de pontos em média
0	93	26	27,96%	5,23E+01	92,89
0,6	72	26	36,11%	5,01E+01	84,32
0,65	72	26	36,11%	5,00E+01	83,63
0,7	72	26	36,11%	4,98E+01	82,00
0,75	71	25	35,21%	4,91E+01	78,53
0,8	67	24	35,82%	4,78E+01	72,05
0,85	62	24	38,71%	4,55E+01	64,42
0,9	55	24	43,64%	4,05E+01	51,53
0,95	43	20	46,51%	3,29E+01	34,37

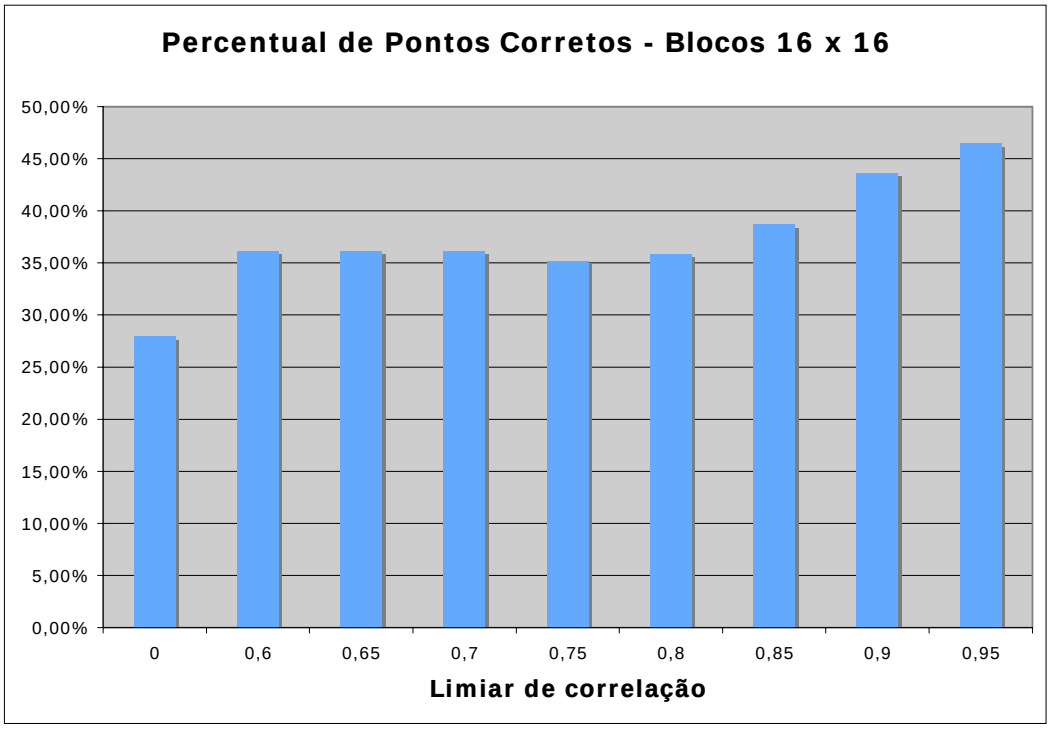

Figura 7-88: Percentual de pontos corretos para blocos 16 por 16

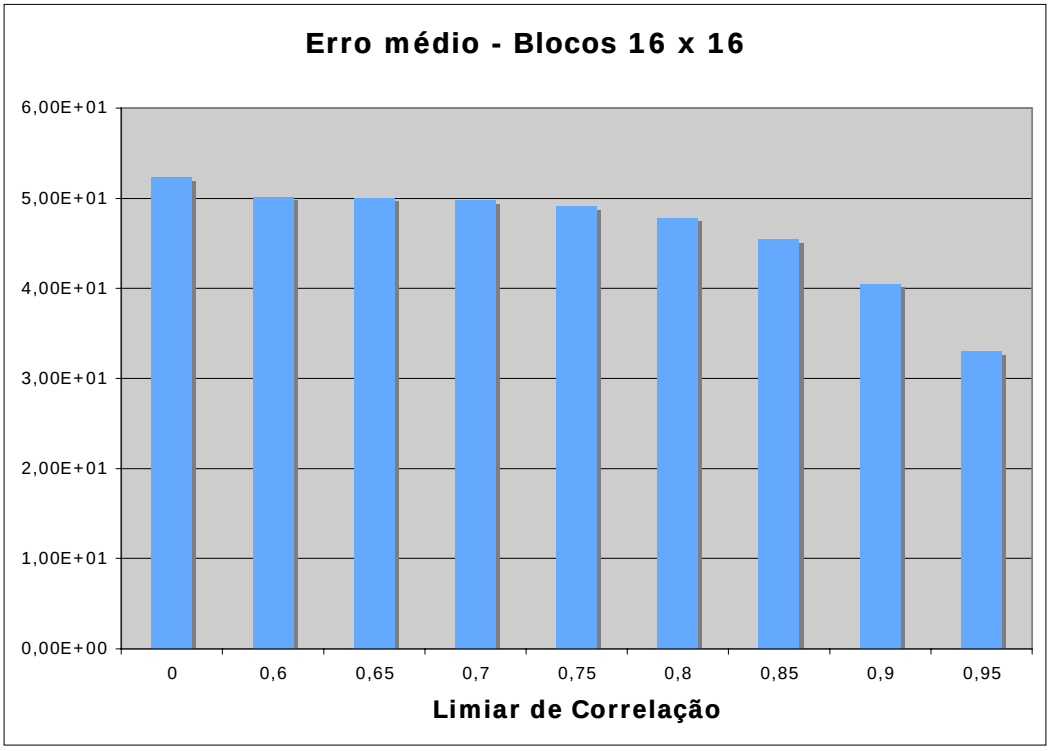

Figura 7-89: Erro médio para blocos 16 por 16

Neste e na maioria dos experimentos apresentados é verificado o comportamento da detecção de pontos em comum com relação ao limiar de correlação utilizado para se eliminar pontos inconsistentes.

É interessante perceber que o percentual de pontos corretos e o erro médio melhoram quando se cresce este limiar. Porém, para o caso do limiar ser muito alto, isto pode acabar atrapalhando pois o número de pontos em comum entre as imagens pode cair demais.

No exemplo presente, como nenhuma outra técnica foi utilizada para se eliminar pontos mal casados, o número médio de pontos encontrado ainda foi aceitável.

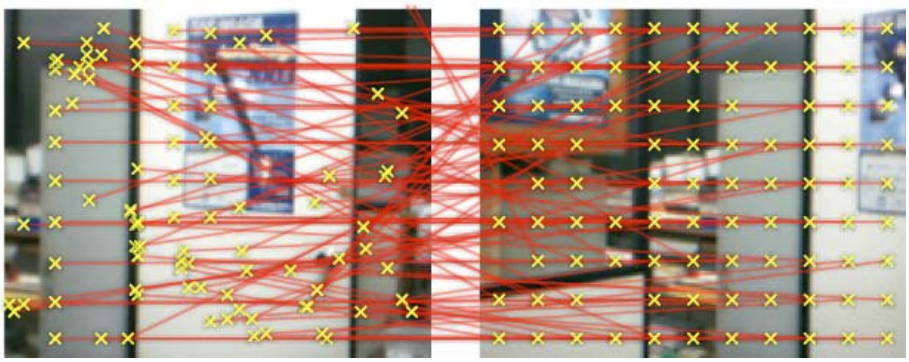

Figura 7-90: Pontos encontrados com limiar de correlação igual a 0

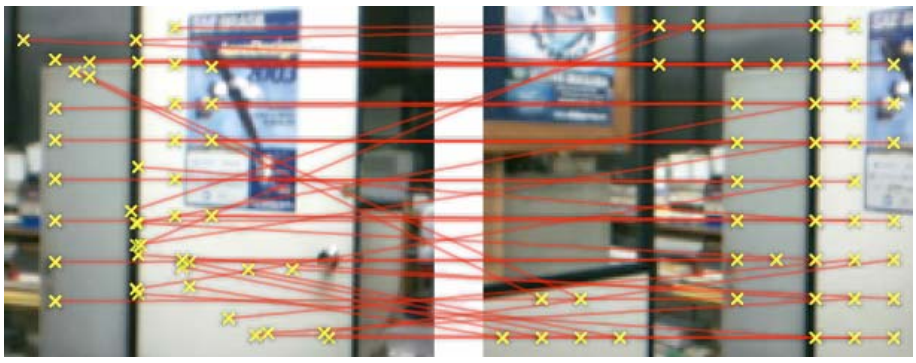

Figura 7-91: Pontos encontrados com limiar de correlação igual a 0,95

A partir das imagens que apresentam os pontos encontrados (Figura 7-90 e Figura 7-91), é visível que o número de pontos corretos para um limiar de correlação superior é maior. Também pode-se ver que esta técnica encontra um número grande de pontos, sendo que muitos incorretos.

Na geração da imagem panorâmica automática (Figura 7-93), nenhuma das imagens foram alinhadas corretamente.

Figura 7-92: Geração de mosaico automática para limiar de correlação igual a 0,90

7.5.1.2. Eliminação por limiar de correlação – blocos 32 x 32

Características do experimento:

- O tamanho dos blocos utilizados foi de 32 por 32;
- Transformação Procrustes;

Tabela 7-5: Eliminação por limiar de correlação para blocos 32 por 32

Limiar de correlação	Número total de pontos	Pontos Corretos	Percentual Ptos. Corretos	Erro médio	Número de pontos em média
0	14	8	57,14%	4,36E+01	16,32
0,6	14	8	57,14%	3,95E+01	14,32
0,65	13	8	61,54%	3,81E+01	13,74
0,7	13	8	61,54%	3,52E+01	12,32
0,75	13	8	61,54%	3,12E+01	11,58
0,8	12	8	66,67%	2,88E+01	10,42
0,85	10	8	80,00%	2,15E+01	8,95
0,9	7	7	100,00%	1,59E+01	6,89
0,95	6	6	100,00%	--------	4,11

Não foi possível calcular o valor do erro médio para o limiar de correlação igual a 0.95 porque houve imagens que não apresentaram pontos em comum. Devido a este fato deve ser observado que o gráfico apresentado na Figura 7-93 não possui a última coluna.

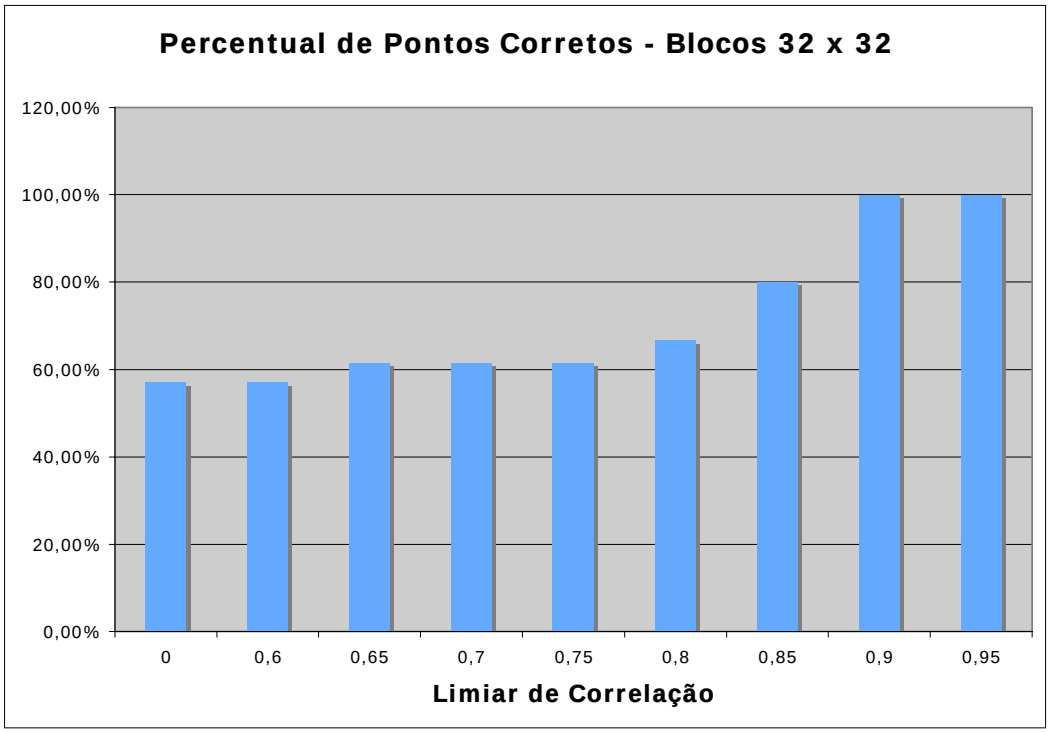

Figura 7-93: Percentual de pontos corretos para blocos 32 por 32

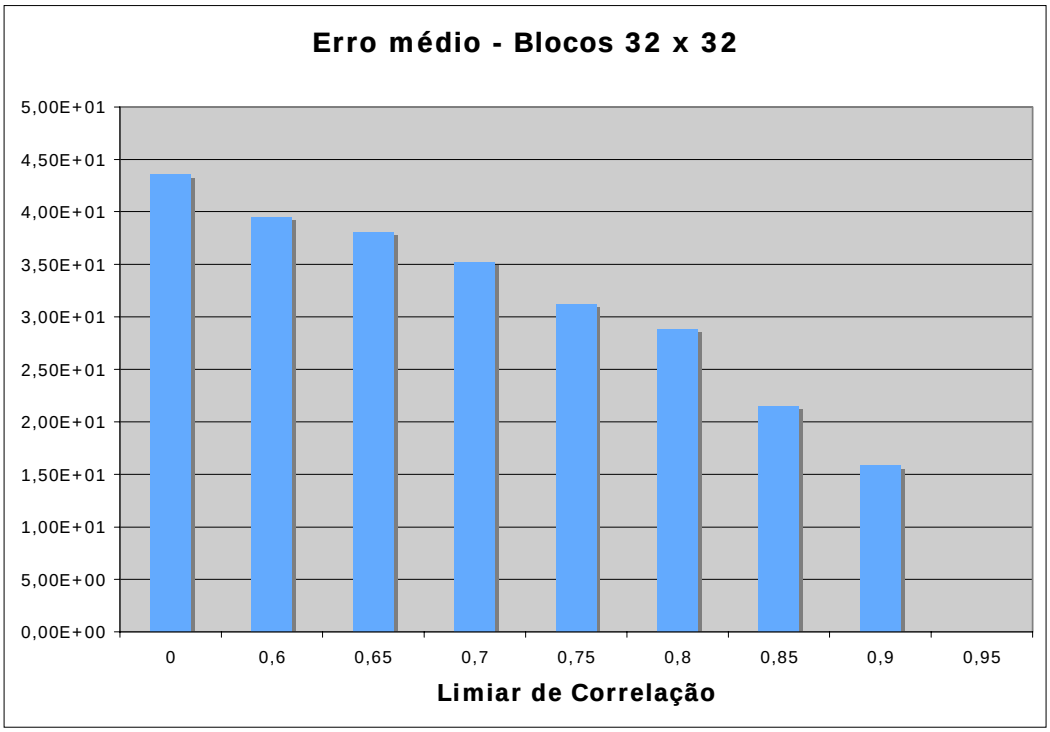

Figura 7-94: Erro médio para blocos 32 por 32

Blocos maiores facilitam o casamento correto de pontos através de correlação. As desvantagens em se trabalhar com blocos maiores está em que o custo computacional é maior e que o número de pontos avaliados é menor.

No caso apresentado, quando com limiar igual a 0.95, chegou a acontecer de nenhum ponto ser encontrado em comum entre as imagens. Neste caso não é possível se montar a imagem panorâmica.

Número pequeno de pontos resultantes leva a algumas desvantagens que serão comentadas mais adiante.

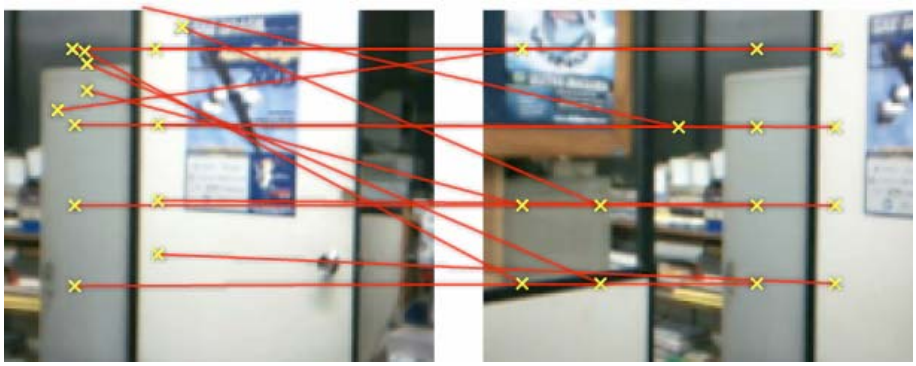

Figura 7-95: Pontos encontrados com limiar de correlação igual a 0

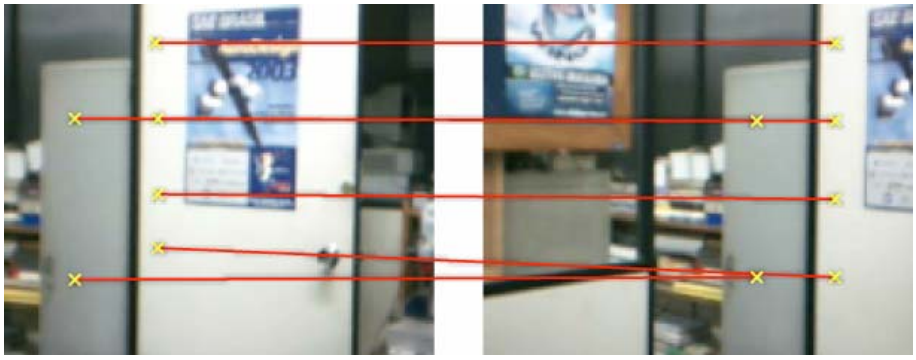

Figura 7-96: Pontos encontrados com limiar de correlação igual a 0,95

A partir das imagens que apresentam os pontos encontrados (Figura 7-95 e Figura 7-96), pode-se ver que esta técnica encontra um número menor de pontos como já observado nos gráficos comentados.

Na panorâmica gerada (Figura 7-97) pode-se perceber que as duas primeiras imagens foram alinhadas corretamente.

Figura 7-97: Geração de mosaico automática para limiar de correlação igual a 0,90

7.5.1.3. Eliminação por amostragem

Tabela 7-6 - Eliminação por amostragem

Limiar de correlação	Número total de pontos	Pontos Corretos	Percentual de Pontos Corretos	Erro médio	Número de Pontos em Média
0	6	6	100,00%	9,07E-02	5,95
0,6	3	3	100,00%	3,21E-14	5,63
0,65	3	3	100,00%	3,21E-14	5,63
0,7	3	3	100,00%	3,21E-14	5,63
0,75	3	3	100,00%	3,02E-14	5,42
0,8	3	3	100,00%	1,95E-02	5,58
0,85	3	3	100,00%	3,42E-14	5,21
0,9	2	2	100,00%	2,58E-02	5,37
0,95	2	2	100,00%	5,87E-02	4,79

Não há porque apresentar o gráfico do percentual de pontos corretos devido ao fato de que este foi de 100% para todos os limiares de correlação. É interessante perceber que isto ocorre devido ao fato de que a eliminação por amostragem descarta a maioria dos pontos, sobrando muito poucos como pode ser

visto nos campos de pontos corretos e número de pontos em média da tabela acima.

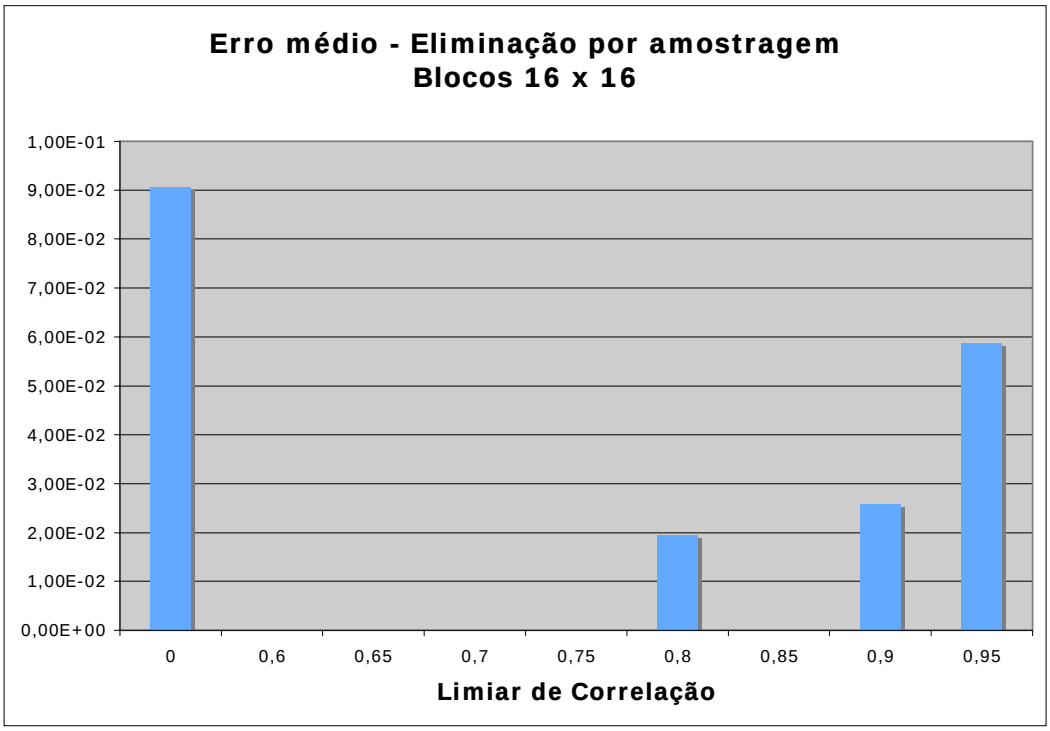

Figura 7-98: Erro médio para eliminação por amostragem

A técnica de eliminação por amostragem deve ter seus resultados avaliados de modo particular. A primeira vista esta técnica aparenta ser muito boa dado que os erros médios encontrados foram desprezíveis e o percentual de pontos corretos foi de 100%.

Primeiramente é bom se lembrar que o erro médio não é um espelho de se os pontos encontrados estão corretos ou não e sim uma medida de o quão de acordo com a matriz de transformação calculada estão os pontos encontrados.

Acontece que é propriedade da técnica eliminar pontos que fujam da transformação *T* predominante e ter como resultado um número muito pequeno de pontos (como pode ser verificado) para os quais o erro de se aplicar a transformação calculada é mínimo. Como o erro médio só é calculado para os pontos não eliminados, este será muito baixo independente de se os pontos foram realmente casados corretamente.

No experimento apresentado, todos os pontos encontrados, para todas as imagens, foram corretos. Porém, é possível que a técnica não funcione em

algumas situações. Neste caso, a técnica tem como desvantagem que se algum ponto for encontrado incorretamente, provavelmente todos os pontos serão.

A eliminação por amostragem foi a técnica mais testada ao longo do projeto e em raras situações se mostrou deficiente.

Para os casos em que a eliminação por amostragem não resulta em bons resultados, há a possibilidade de se fazer a combinação dos outros métodos apresentados.

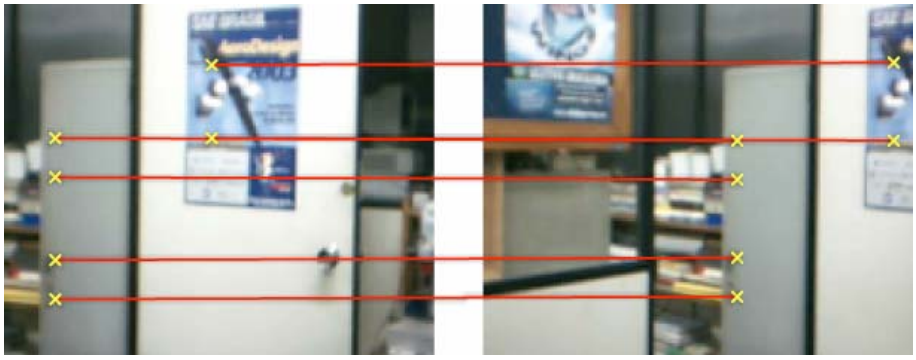

Figura 7-99: Pontos encontrados com limiar de correlação igual a 0

Figura 7-100: Pontos encontrados com limiar de correlação igual a 0,95

Pode-se ver nas imagens que apresentam os pontos encontrados (Figura 7-99 e Figura 7-100) que a técnica encontrou todos os pontos corretamente. Também é visto que o número de pontos encontrados é pequeno, podendo chegar a somente 2 pontos para o limiar de correlação mais alto como visto na Figura 7-100

A imagem panorâmica gerada (Figura 7-101) neste experimento apresenta todas as vistas alinhadas corretamente.

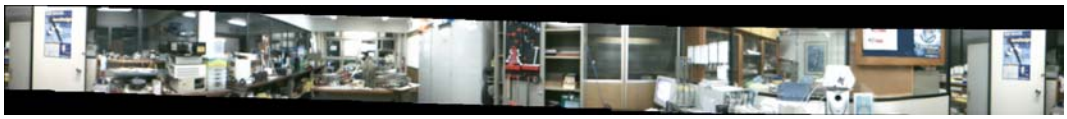

Figura 7-101: Geração de mosaico automática para limiar de correlação igual a 0,90

7.5.1.4. Eliminação por RANSAC e matriz de pesos W

Tabela 7-7: Eliminação por RANSAC e matriz de pesos W

Limiar de correlação	Número total de pontos	Pontos Corretos	Percentual de Pontos Corretos - RANSAC - Blocos 16 x 16	Erro médio - RANSAC - Blocos 16 x 16	Número de pontos em média
0	50	26	52,00%	3,50E+01	76,47
0,6	50	26	52,00%	2,40E+01	61,95
0,65	50	26	52,00%	2,37E+01	61,42
0,7	50	26	52,00%	2,11E+01	58,42
0,75	49	25	51,02%	1,73E+01	52,95
0,8	48	24	50,00%	9,87E+00	45,16
0,85	46	24	52,17%	7,51E+00	41,05
0,9	41	24	58,54%	5,54E+00	34,47
0,95	29	20	68,97%	4,30E+00	23,63

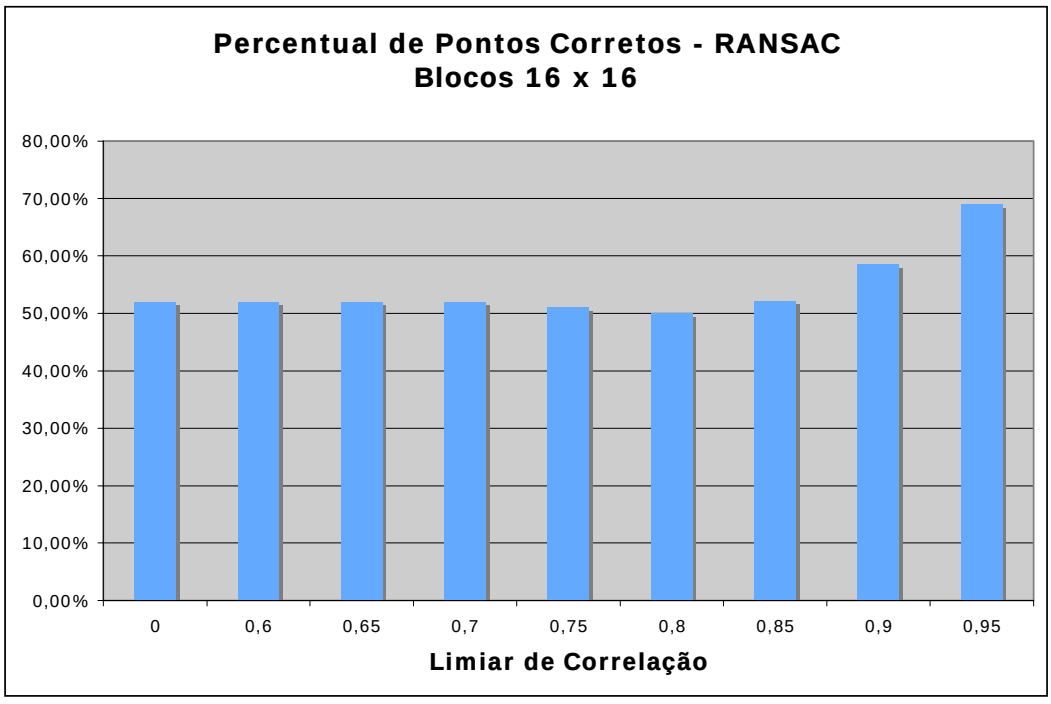

Figura 7-102: Percentual de pontos corretos para RANSAC e matriz de pesos W

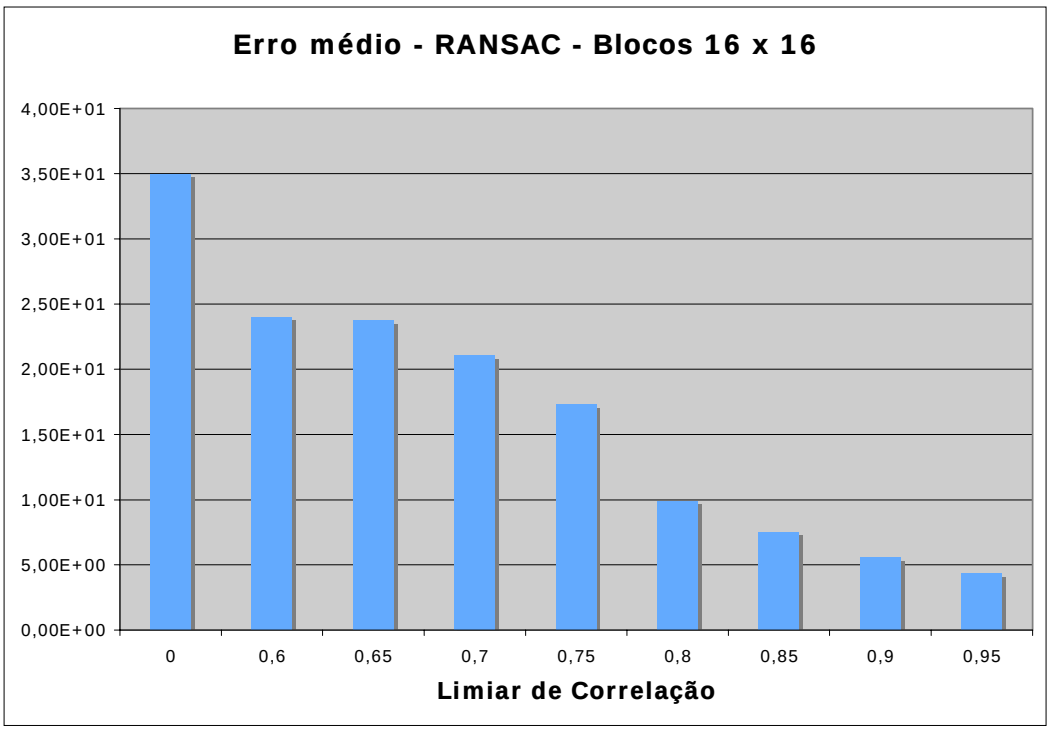

Figura 7-103: Erro médio para RANSAC e matriz de pesos W

O uso de RANSAC e do ajuste da matriz de pesos *W* demonstrou melhorar substancialmente os resultados sem ter como contrapartida a queda elevada do número de pontos em comum.

O uso desta técnicas deve ser combinado com outras de modo que seja mais eficiente do que visto nos resultados aqui presentes. Sabe-se que RANSAC não é extremamente eficaz quando o número de pontos errados antes do seu uso é muito maior que o número de pontos corretos, o que é o caso em questão.

Várias panorâmicas foram experimentadas combinando-se RANSAC com as outras técnicas avaliadas ao longo do projeto levando a resultados similares aos apresentados para a eliminação por amostragem.

A vantagem em se combinar RANSAC com outras técnicas em relação a eliminação por amostragem está em se conseguir um maior número de pontos em comum entre as imagens. Portanto, enquanto o uso da técnica eliminação por amostragem pode resultar em uma panorâmica mau formada devido a poucos pontos incorretos, o mesmo não acontece para o uso de RANSAC em combinação com outras técnicas.

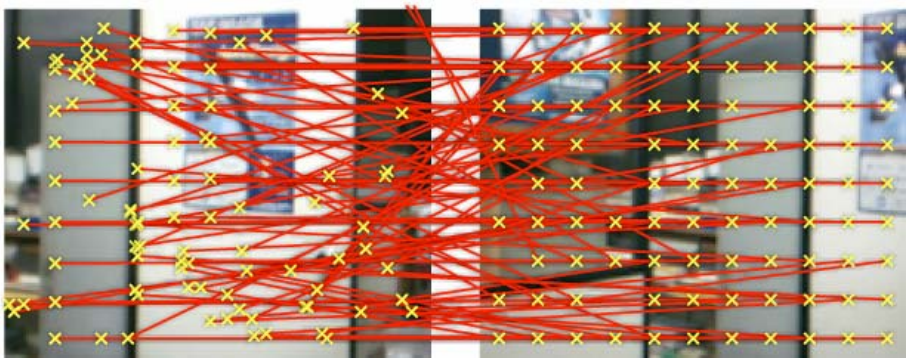

Figura 7-104: Pontos encontrados com limiar de correlação igual a 0

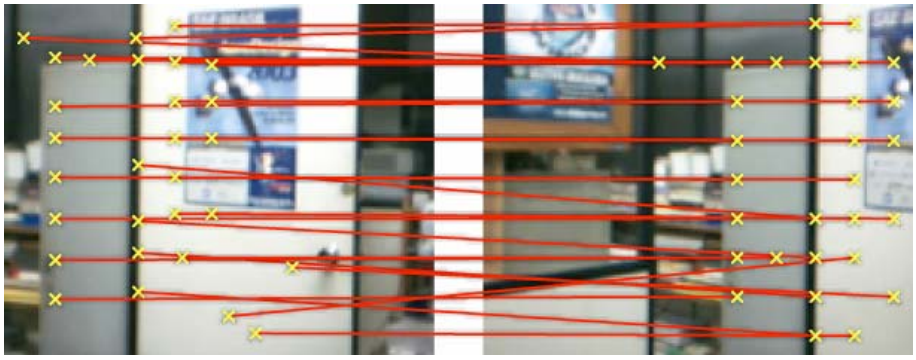

Figura 7-105: Pontos encontrados com limiar de correlação igual a 0,90

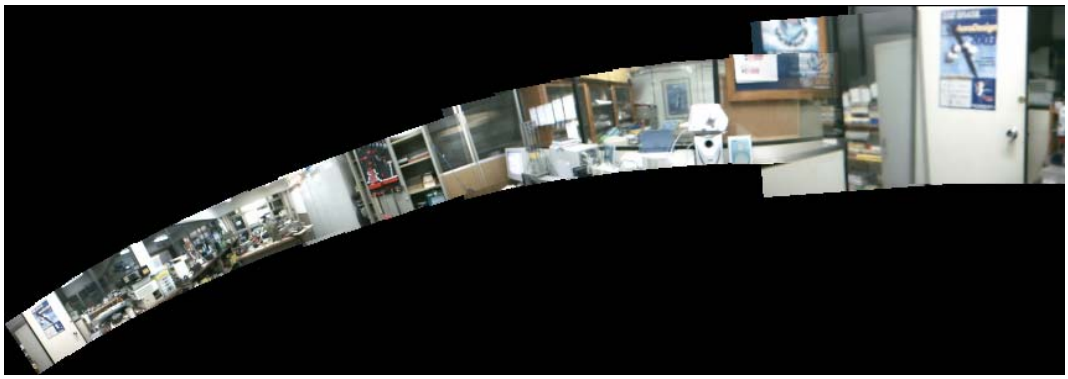

Figura 7-106: Geração de mosaico automática para limiar de correlação igual a 0,90

Perceba que utilizando RANSAC e ajuste da matriz *W*, mesmo quando a combinação de imagens não é perfeita, a técnica não leva a combinações completamente equivocadas. Isto é de grande interesse e traz maior estabilidade para a geração de panorâmicas.

Quando a eliminação por amostragem não funciona, ela gera panorâmicas que não podem ser utilizadas por serem completamente erradas.

7.5.1.5.
Eliminação por má correlação

Tabela 7-8: Eliminação por má correlação

ρ	β	Número total de pontos	Pontos Corretos	Percentual de Pontos Corretos	Erro médio	Número de pontos em média
0,95	10,00%	28	19	67,86%	4,61E+01	52,47
0,95	20,00%	32	19	59,38%	4,79E+01	60,95
0,95	30,00%	36	19	52,78%	4,81E+01	65,37
0,95	40,00%	40	22	55,00%	4,84E+01	68,21
0,9	10,00%	17	12	70,59%	3,97E+01	31,26
0,9	20,00%	18	13	72,22%	4,49E+01	42,00
0,9	30,00%	22	13	59,09%	4,60E+01	47,11
0,9	40,00%	23	14	60,87%	4,65E+01	50,79
0,8	10,00%	6	6	100,00%	1,28E+01	10,74
0,8	20,00%	9	9	100,00%	2,37E+01	15,79
0,8	30,00%	9	9	100,00%	3,15E+01	20,16
0,8	40,00%	12	9	75,00%	3,70E+01	24,16
0,7	10,00%	2	2	100,00%	-------- ---	3,21
0,7	20,00%	4	4	100,00%	1,00E+01	6,00
0,7	30,00%	4	4	100,00%	1,05E+01	6,95
0,7	40,00%	5	5	100,00%	1,38E+01	8,79

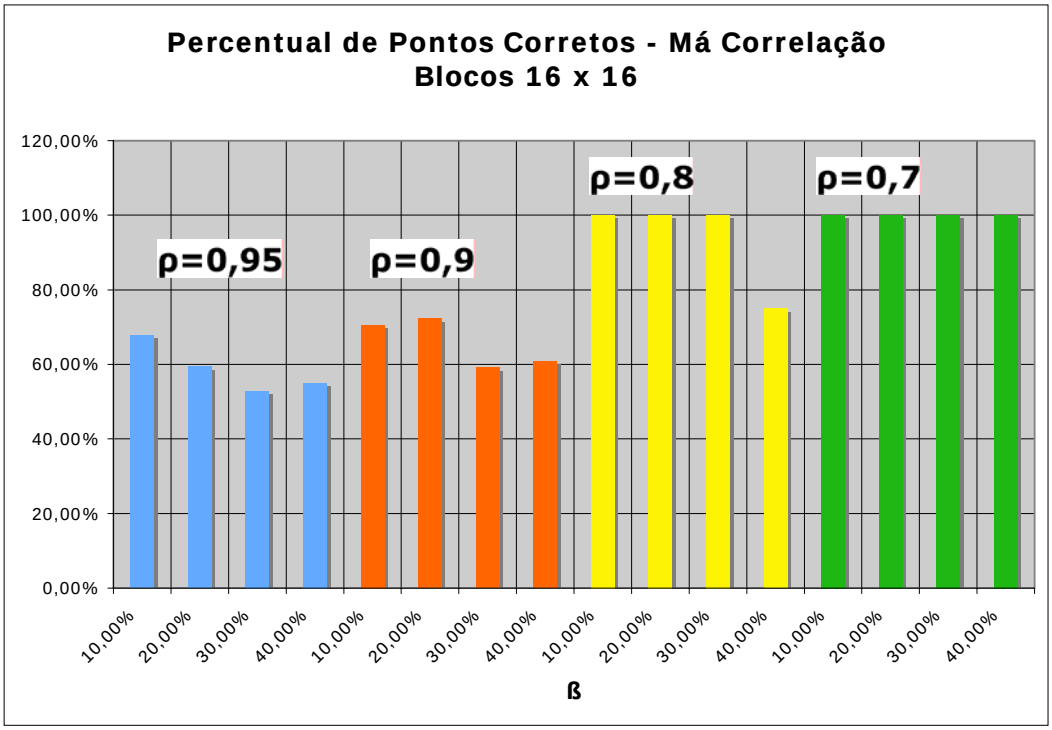

Figura 7-107: Percentual de pontos corretos para a técnica de eliminação por má correlação

Legenda de cores para Figura 7-107:

- Azul: $\rho = 0,95$;
- Laranja: $\rho = 0,9$;
- Amarelo: $\rho = 0,8$;
- Azul: $\rho = 0,7$;

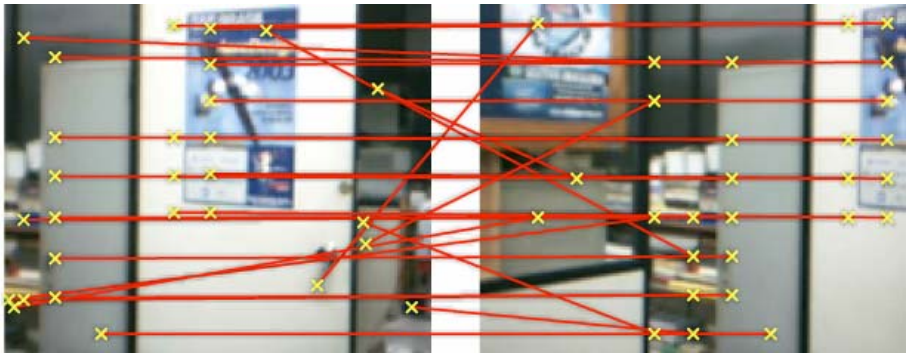

Figura 7-108: Pontos encontrados com $\rho = 0,95$ e $\beta = 10\%$

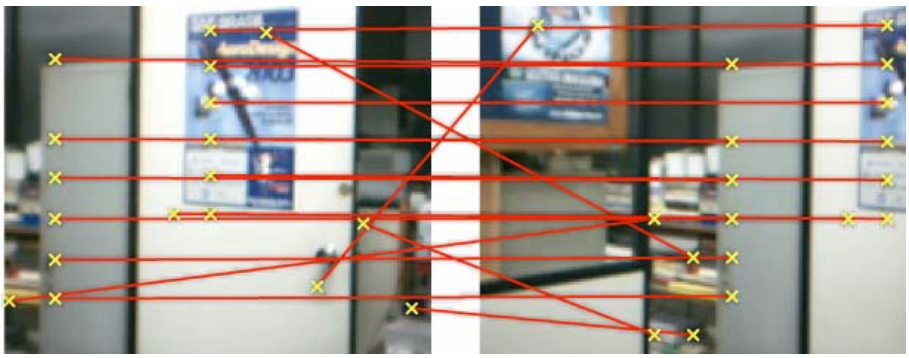

Figura 7-109: Pontos encontrados com $\rho = 0,9$ e $\beta = 10\%$

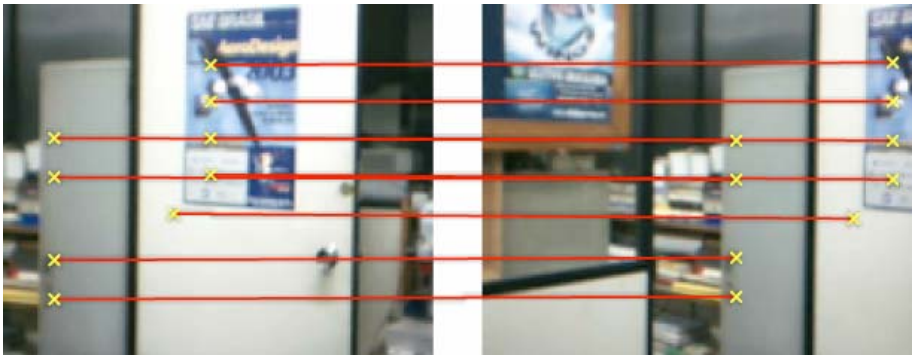

Figura 7-110: Pontos encontrados com $\rho = 0,8$ e $\beta = 10\%$

Figura 7-111: Pontos encontrados com $\rho = 0,7$ e $\beta = 10\%$

Figura 7-112: Geração de mosaico automática para limiar de correlação igual a $\rho = 0,9$ e $\beta = 30\%$

Este procedimento proposto demonstrou ser uma boa técnica para eliminar pontos mal correlacionados. Quanto menor o valor de ρ e menor o percentual de β, mais pontos serão eliminados e melhor será o resultado. Porém, como em todas as técnicas apresentadas deve se tomar o cuidado de não se eliminar pontos em excesso.

Percebe-se que mesmo para o maior valor de ρ e o maior percentual de β utilizados, os resultados já são muito superiores do que quando a técnica não é utilizada. Com ρ e β adequados, consegue-se eliminar grande parte dos pontos incorretos sem eliminar muitos pontos corretos.

O uso desta técnica com RANSAC e ajuste da matriz de pesos é bastante adequado para a geração de panorâmicas para o presente projeto.

7.6. Segmentação da imagem por *Quadtree* baseada em entropia

A imagem da Figura 7-113 foi utilizada para se fazer um estudo das possibilidades encontradas ao se aplicar a segmentação por *Quadtree* baseada em entropia. Esta imagem possui resolução de 512 por 512 *pixels*.

Figura 7-113: Figura *Airplane*

Foram feitos 240 testes de modo a se estudar:

- Métodos padrão, Open-Close e Close-Open;
- Limiares do valor de entropia para se fazer a divisão *quadtree*;
- Tamanho de raios r utilizados para operações de abertura e fechamento;
- Tamanho mínimo de regiões aceitas;
- Pré-aplicação de filtros do tipo mediana;
- Pré-aplicação de filtro de média de formato radial de raio r' sobre a imagem;

Dos 240 testes foram escolhidos aqueles que apresentam resultados mais relevantes e que melhor caracterizam as observações verificadas.

7.6.1 Diferentes métodos

As imagens apresentadas na Figura 7-114 foram feitas com os seguintes acertos:

- Figura 7-114 a: Método padrão; Pré-processada com filtro de média radial de raio r' igual a 4; Limiar do valor de entropia para a divisão *quadtree* igual a 4;
- Figura 7-114 b: Método Open-Close; Pré-processada com filtro de média radial de raio r' igual a 4; Limiar do valor de entropia para a divisão *quadtree* igual a 4; Raio r utilizado para operações de abertura e fechamento igual a 5;
- Figura 7-114 c: Método Close-Open; Pré-processada com filtro de média radial de raio r' igual a 4; Limiar do valor de entropia para a divisão *quadtree* igual a 5; Raio r utilizado para operações de abertura e fechamento igual a 5;

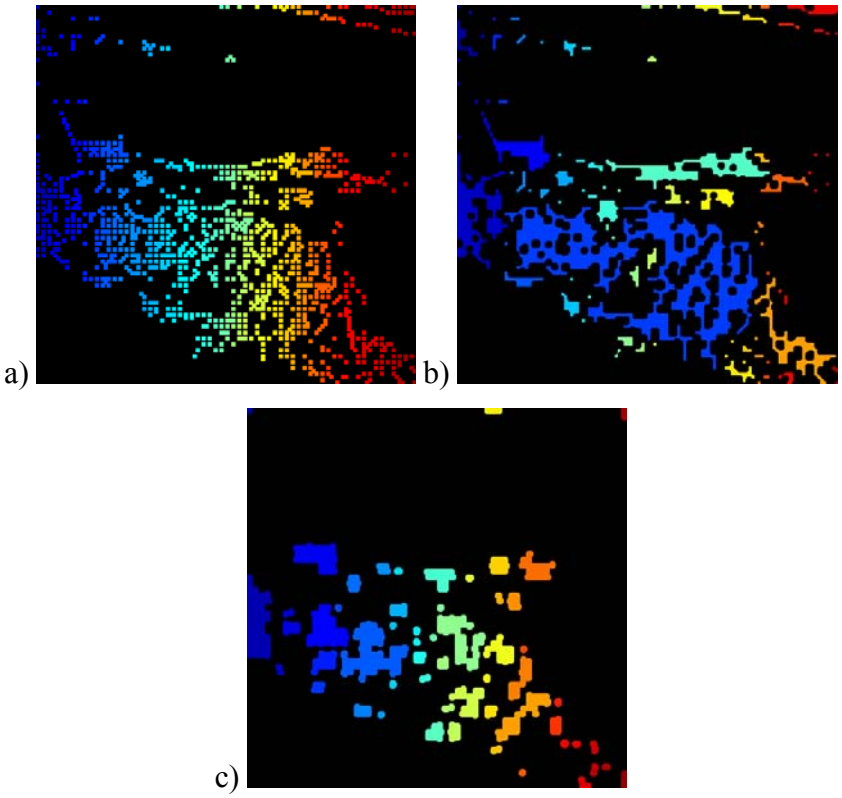

Figura 7-114: Diferentes métodos utilizados. a) Método padrão; b) Método *Open-Close*; c) Método *Close-Open*;

Ao longo dos experimentos a serem apresentados, mais resultados relativos aos diferentes métodos serão apresentados e comparados. As figuras apresentadas nesta seção têm como objetivo ilustrar de um modo geral como se comportam os diferentes métodos.

Pode-se dizer sobre os três métodos em linhas gerais:

- O método padrão faz o agrupamento de regiões quadradas diretamente vizinhas. Não é muito eficaz em agrupar regiões próximas podendo resultar em uma série de regiões de tamanho pequeno;
- O método Open-Close é eficaz em agrupar regiões próximas tal como em descartar regiões pequenas e esparsas. Como resultado costuma-se ter regiões bem definidas representativas de áreas detalhadas na imagem. O método Open-Close é o que se mostrou mais adequado dentro do presente trabalho;
- O método Close-Open gera conglomerados em torno de centróides. Não costuma gerar regiões bem definidas em relação a detalhes na imagem mas sim blocos que representam o centro de regiões de interesse. Apesar de eficaz em encontrar centróides, não segmenta bem áreas com grande quantidade de detalhes;

7.6.1.1. Variando o limiar do valor de entropia para divisão *quadtree*

A Figura 7-115 apresenta como se comporta a divisão *quadtree* quando se varia o limiar do valor de entropia mínimo utilizado para se uma célula será ou não sub-dividida. Os valores utilizados foram de 6.5, 6, 5 e 4. É interessante verificar que a imagem usada possui entropia igual a aproximadamente 6.7.

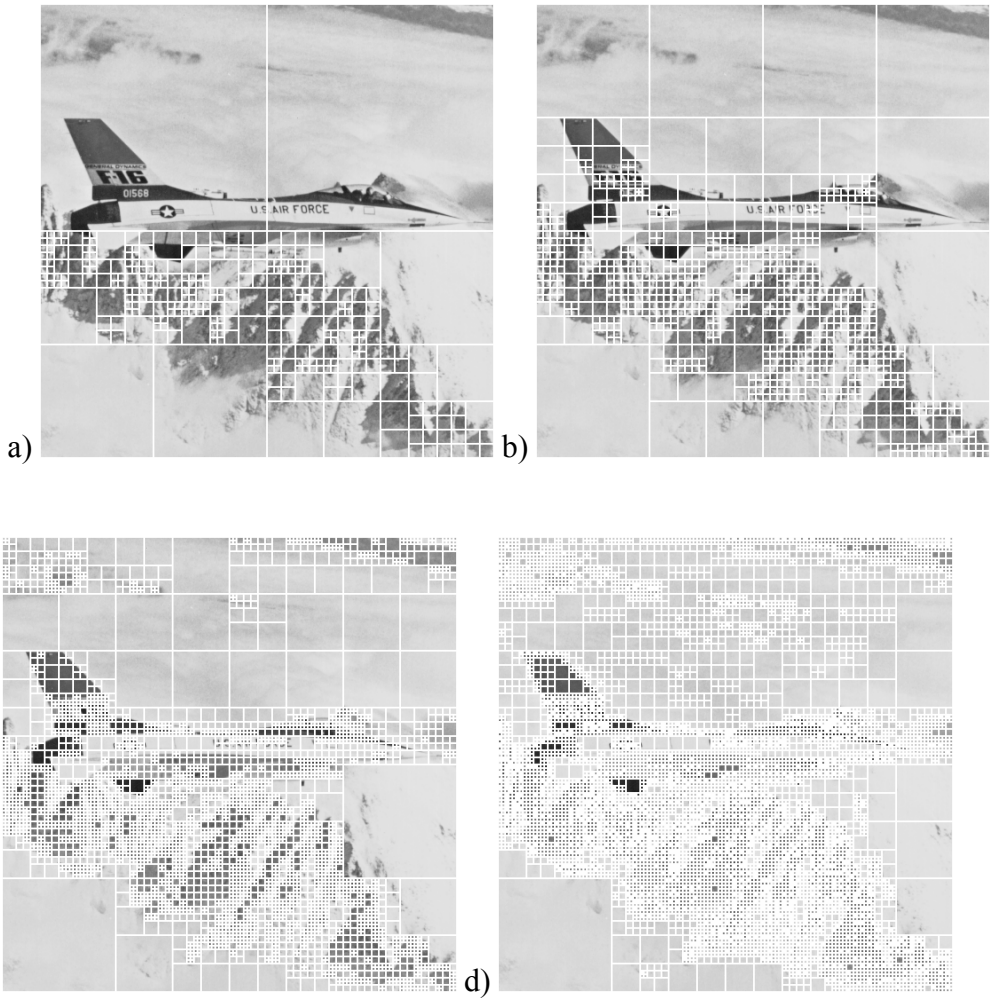

Figura 7-115: Diferentes limiares do valor de entropia para divisão *quadtree*. a) Limiar igual a 6.5; b) Limiar igual a 6; c) Limiar igual a 5; d) Limiar igual a 4;

Pode-se verificar que quanto menor o limiar do valor de entropia, mais a imagem será sub-dividida e, portanto, mais detalhes são encontrados. A escolha do limiar utilizado é portanto muito importante pois define a quantidade de informação desejada que se procura para detalhes da imagem.

O reflexo do limiar do valor de entropia para a divisão *quadtree* para a segmentação em diferentes regiões através dos diferentes métodos é apresentado a seguir.

7.6.1.2. Método Padrão

As imagens apresentadas nestes testes (Figura 7-116) foram pré-processadas com filtro de média radial de raio *r'* igual a 2.

Pode-se verificar que quando o resultado do *quadtree* são regiões muito pequenas e estas não são diretamente vizinhas, pode acontecer como no caso *d* em que o método não consegue agrupar corretamente as regiões e acaba gerando um número muito grande de regiões extremamente pequenas.

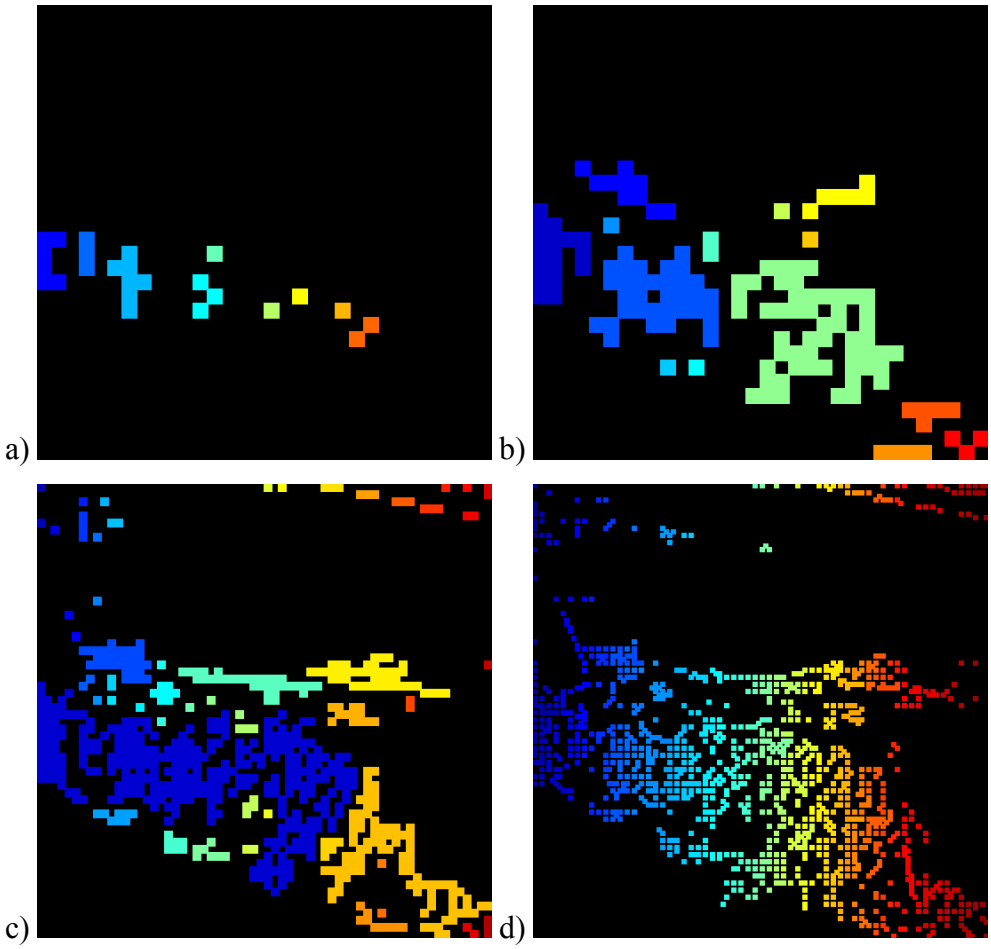

Figura 7-116: Diferentes limiares do valor de entropia para método padrão. a) Limiar igual a 6.5; b) Limiar igual a 6; c) Limiar igual a 5; d) Limiar igual a 4;

Por outro lado, quando se trabalha com limiar do valor de entropia muito alto, próximo ao da entropia da imagem, o número de regiões geradas pode ser muito pequeno e o resultado final pode não representar bem regiões de interesse.

7.6.1.3. Método Open-Close

As imagens da Figura 7-117 foram pré-processadas com filtro de média radial de raio *r'* igual a 4 e tiveram raio *r* utilizado para operações de abertura e fechamento igual a 5.

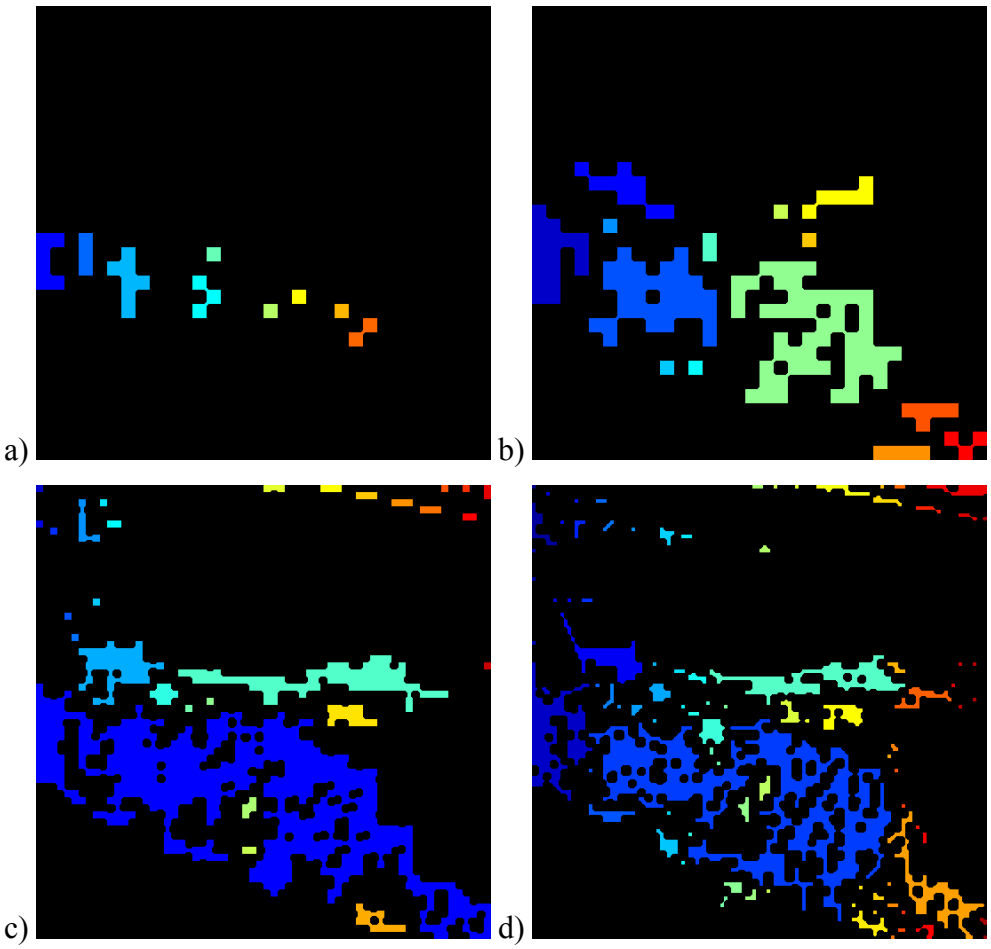

Figura 7-117: Diferentes limiares do valor de entropia para método *Open-Close*. a) Limiar igual a 6.5; b) Limiar igual a 6; c) Limiar igual a 5; d) Limiar igual a 4;

Como com o método padrão, limiares do valor de entropia muito altos fazem com que a imagem não seja muito dividida e o nível de detalhes encontrado acaba sendo baixo.

Pode-se perceber nas figuras *a* e *b* que as regiões resultantes não representam muito bem as áreas de interesse. Já nas figuras *c* e *d* o nível de detalhes é maior.

De um modo geral, o que acontece é que quanto menor o limiar, mais a imagem é dividida e regiões menores são encontradas para então o método *open-close* agrupá-las como visto.

7.6.1.4. Método Close-Open

As imagens da Figura 7-118 foram pré-processadas com filtro de média radial de raio *r'* igual a 4 e tiveram raio *r* utilizado para operações de abertura e fechamento igual a 5.

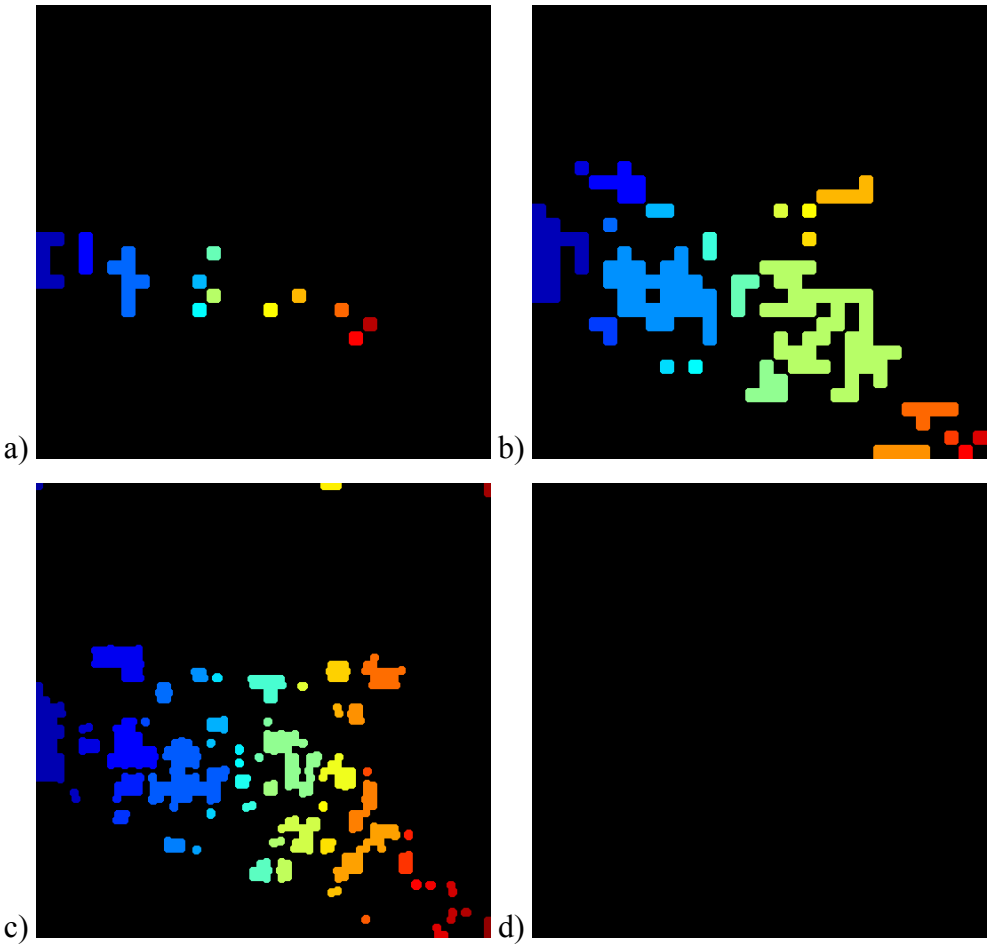

Figura 7-118: Diferentes limiares do valor de entropia para método *Close-Open*. a) Limiar igual a 6.5; b) Limiar igual a 6; c) Limiar igual a 5; d) Limiar igual a 4;

Regiões muito pequenas costumam desaparecer quando se usa o método *Close-Open*. Isto pode ser verificado nas imagens apresentadas. A imagem *d*

caracteriza bem este problema dado que não sobrou nenhuma região após os procedimentos morfológicos de fechamento e abertura.

Este método gerou diversas imagens sem nenhuma região ao longo das experiências feitas, portanto, deve se trabalhar com o mesmo com o cuidado de não se eliminar todas as regiões após sua aplicação.

7.6.2. Tamanho de raios *r* utilizados para operações de abertura e fechamento

Método Open-Close

As imagens da Figura 7-119 foram pré-processadas com filtro de média radial de raio *r'* igual a 4 e o limiar do valor de entropia para divisão de *quadtree* utilizado foi de entropia igual a 4.

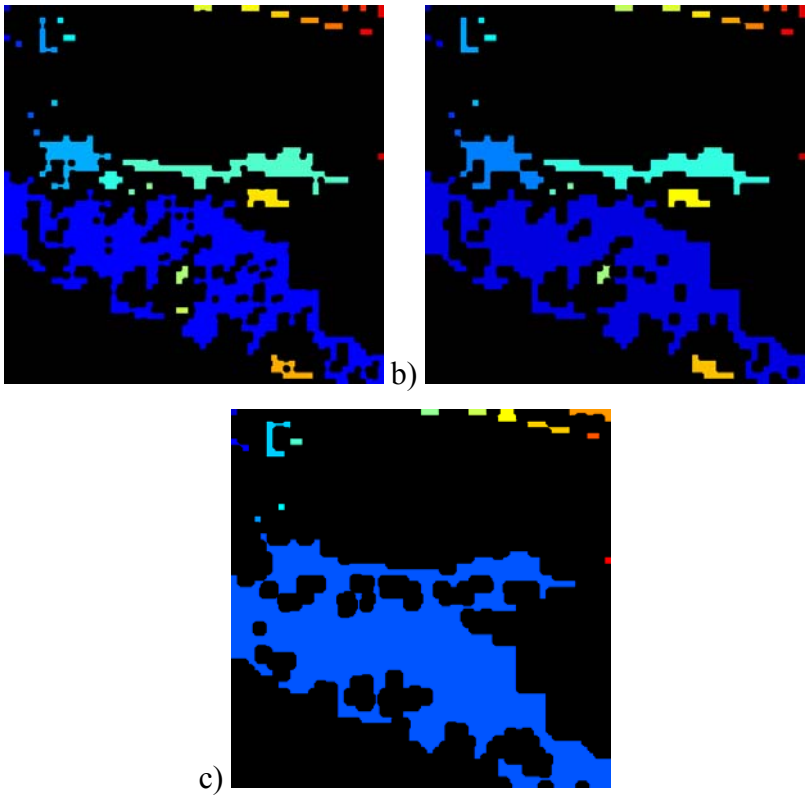

Figura 7-119: Diferentes raios nas operações de abertura e fechamento do método *Open- Close*. a) Raio igual a 5; b) Raio igual a 7; c) Raio igual a 10

Pode-se verificar que regiões próximas tendem a se agrupar quando se aumenta o raio *r'* utilizado. Isto é de interesse em muitos casos de modo a se

evitar que o resultado final obtido apresente um número excessivo de regiões próximas.

Ao mesmo tempo, quando se cresce o tamanho de *r'*, regiões isoladas tendem a ser descartadas.

Valores muito grandes de *r'* têm a desvantagem de unir áreas que não possuam qualquer continuidade real.

Método Close-Open

As imagens da Figura 7-120 foram pré-processadas com filtro de média radial de raio *r'* igual a 4 e o limiar do valor de entropia para divisão de *quadtree* utilizado foi de entropia igual a 6.

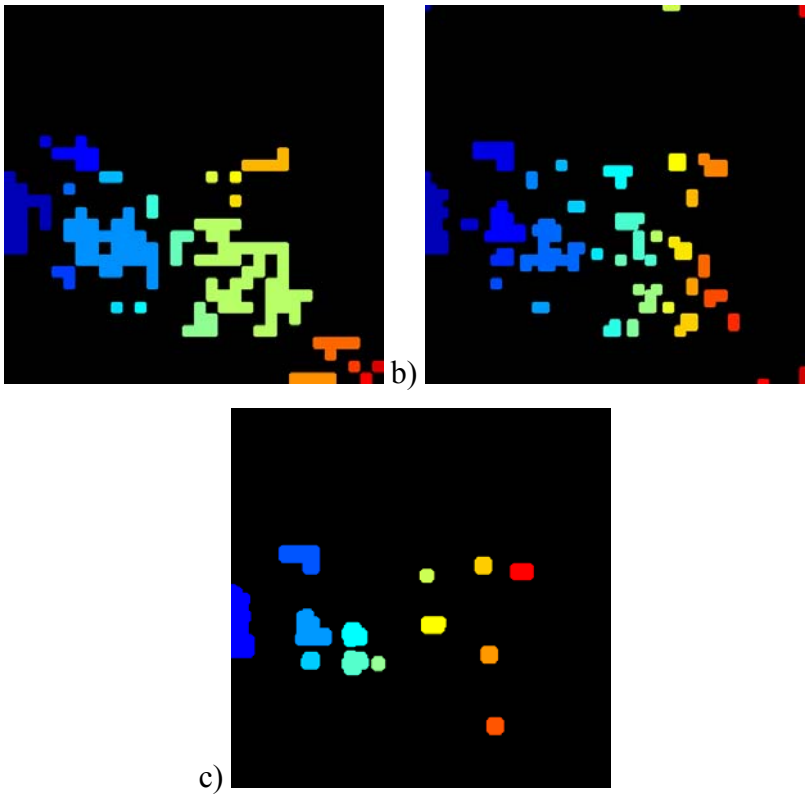

Figura 7-120: Diferentes raios nas operações de abertura e fechamento do método *Close-Open*. a) Raio igual a 5; b) Raio igual a 7; c) Raio igual a 10;

Para valores altos de raio r, pode acontecer do número de regiões cair até zero. Isto ocorreu em muitos experimentos, principalmente aqueles com limiar de entropia mais baixo ao contrário do método padrão. Baixos limiares de entropia significam regiões pequenas para serem processadas pelas operações de

fechamento e posterior abertura, podendo ser descartadas logo na operação de fechamento.

7.6.3. Aplicando filtro de média radial

A aplicação de filtro foi experimentada para diferentes raios *r'* para o método padrão.

As imagens da Figura 7-121 utilizaram limiar do valor de entropia para divisão de *quadtree* de 5.

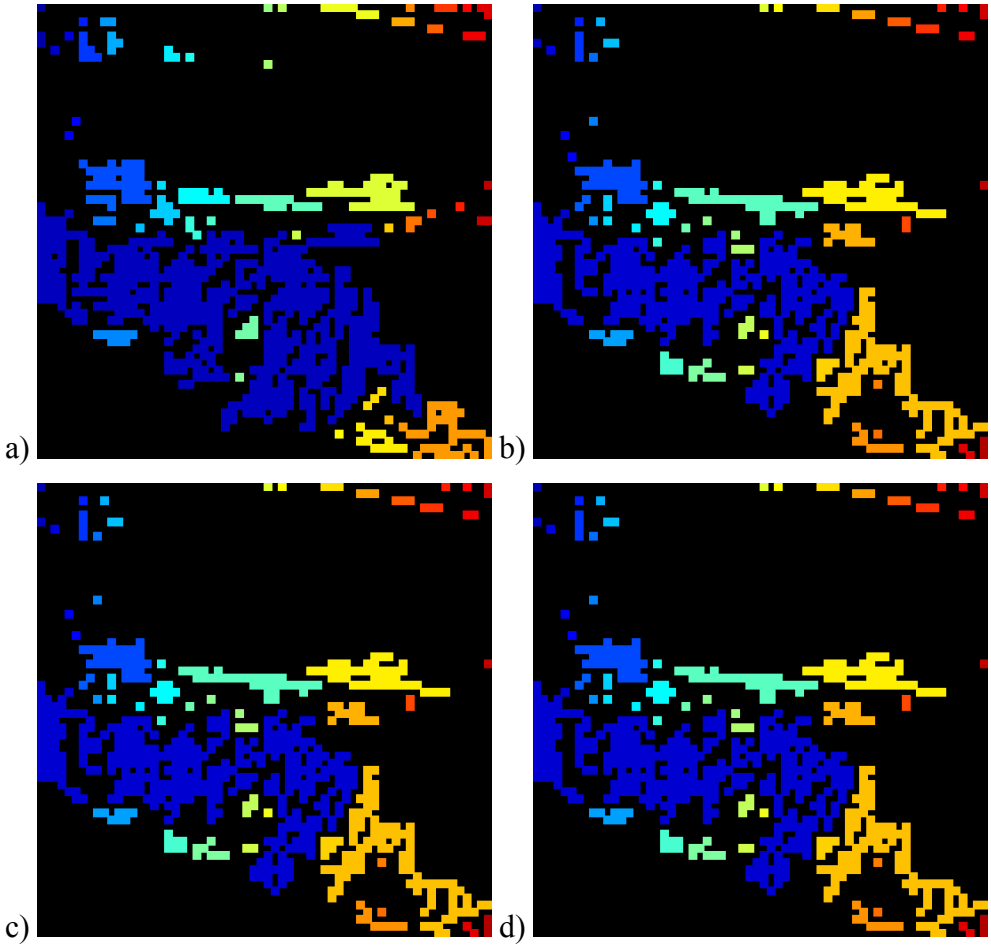

Figura 7-121: Diferentes raios para filtro de média. a) Sem filtro; b) Raio igual a 2; c) Raio igual a 4; d) Raio igual a 8;

Apesar das diferenças apresentadas nas figuras terem sido pequenas, percebeu-se ao longo do desenvolvimento do projeto que era interessante se

trabalhar com alguma filtragem das imagens de modo a se eliminar ruídos que poderiam resultar em sub-divisões indesejadas.

7.6.4. Tamanho mínimo de regiões aceitas

Um último processamento é feito sobre as regiões de modo a descartar regiões de tamanho desprezível

As imagens da Figura 7-122 foram pré-processadas com filtro de média radial de raio *r'* igual a 4, tiveram raio *r* utilizado para operações de abertura e fechamento igual a 5 e usaram limiar do valor de entropia para divisão *quadtree* de entropia igual a 4.

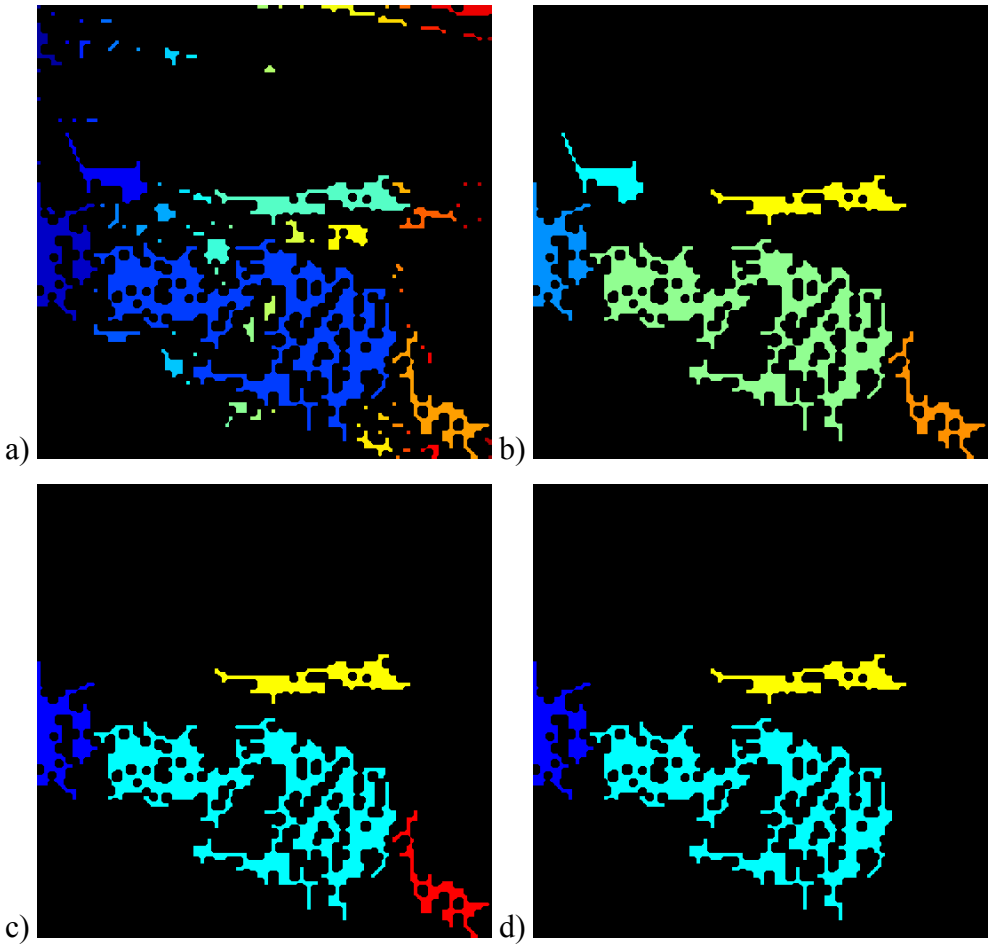

Figura 7-122: Diferentes tamanhos de regiões mínimos. a) Sem restrição; b) Mínimo de 1000 *pixels*; c) Mínimo de 2000 *pixels*; d) Mínimo de 3000 *pixels*

A eliminação de regiões muito pequenas é de grande interesse de modo a se descartar detalhes isolados e pequenos.

Outra possibilidade trabalhada, é de se escolher um número *n* das maiores regiões obtidas para se seguir a exploração.

7.6.5. Aplicando *Quadtree* em panorâmicas

Para imagens panorâmicas, a primeira divisão feita da imagem é pela metade verticalmente e em *n* janelas horizontais de tamanho próximo ao tamanho da janela horizontal. Portanto, a primeira divisão não seria um *quadtree* mas sim um *ntree*. A partir de então as divisões são em quatro janelas. Isto foi feito para se obter resultados mais coerentes. Se a imagem panorâmica fosse dividida inicialmente em quatro, isto resultaria em janelas retangulares extremamente largas. As janelas terão tamanho aproximadamente quadrado com o procedimento adotado.

As imagens apresentadas da Figura 7-123 à Figura 7-125 são relativos à aplicação da divisão *quadtree* em uma imagem panorâmica. A Figura 7-123 apresenta a panorâmica inteira com as regiões coloridas sobrepostas. Em Figura 7-124 e Figura 7-125 pode-se ver a mesma panorâmica porém dividida em duas imagens para melhor visualização. A configuração utilizada foi:

- Método *quadtree*;
- Filtro de média de raio 4;
- Raio para operações morfológicas igual a 10;
- Limiar do valor de entropia na divisão igual a 90% da entropia total da imagem;
- Somente as 5 maiores regiões foram escolhidas;

Figura 7-123: Imagem panorâmica inteira com regiões sobrepostas

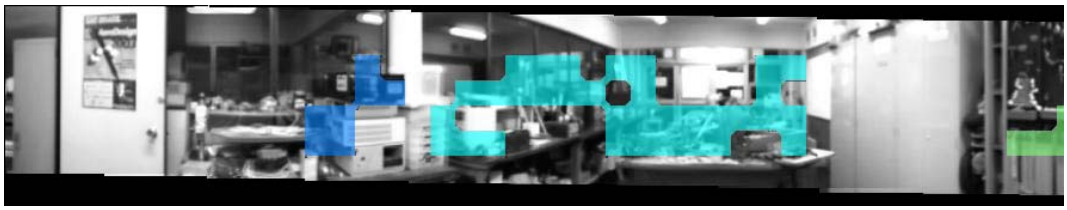

Figura 7-124: Metade esquerda da panorâmica

Figura 7-125: Metade direita da panorâmica

A mesma configuração com raio para operações morfológicas igual a 20, foi aplicada na panorâmica gerando o resultado verificado na Figura 7-126. Como feito anteriormente, são apresentadas imagens relativas à seção esquerda e direita da panorâmica, na Figura 7-127 e na Figura 7-128Figura 7-34, para facilitar o entendimento do resultado.

Figura 7-126: Imagem panorâmica inteira com regiões sobrepostas

Figura 7-127: Metade esquerda da panorâmica

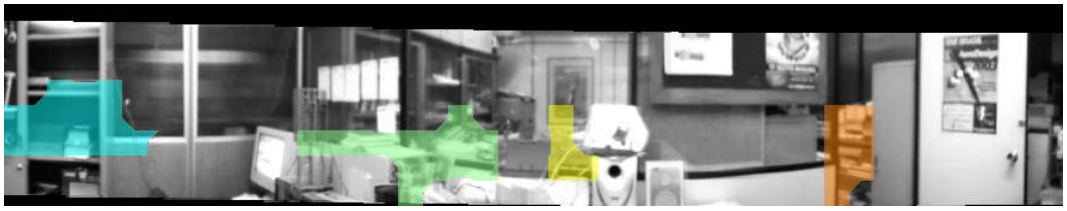

Figura 7-128: Metade direita da panorâmica

Por último é apresentado o mesmo experimento com raio igual a 3 da Figura 7-129 à Figura 7-131.

Figura 7-129: Imagem panorâmica inteira com regiões sobrepostas

Figura 7-130: Metade esquerda da panorâmica

Figura 7-131: Metade direita da panorâmica

Pode-se perceber que para o raio maior, diferentes regiões formaram um mesmo aglomerado. É importante ressaltar que os mesmos parâmetros levam a resultados diferentes em imagens com diferentes características. O valor do raio utilizado nas operações morfológicas é, portanto, muito importante pois pode levar a um número grande de regiões muito pequenas, tal como a uma região muito grande que ocupe toda a panorâmica. O ajuste automático deste valor é portanto de interesse para futuros projetos.

Para os 3 experimentos pode-se verificar que aquelas regiões com mais detalhes na imagem foram selecionadas através da segmentação por *quadtree* demonstrando a validade da técnica.

7.7. Navegação para Nós Conhecidos e Desconhecidos utilizando-se de *Visual Tracking* e Transformação SIFT

Ao longo do projeto, a navegação por *Visual Tracking* e com auxílio de descritores SIFT foi amplamente testada através de diversas configurações e diferentes implementações. Este experimento aqui apresentado tem como objetivo

ilustrar de um modo genérico a implementação aqui proposta e mostrar sua eficiência mostrando de um modo geral os métodos de navegação propostos com auxílio de descritores SIFT.

O experimento aqui descrito é de uma navegação feita com o robô para um lugar já visto. Neste experimento, o robô foi colocado sobre uma mesa no laboratório e uma imagem foi obtida. Então, o robô foi movido para trás e girado. A partir desta nova posição o robô navegou em direção à posição inicial como descrito na seção 6.6.2. Este experimento é interessante por abordar diferentes etapas discutidas ao longo desta dissertação:

- SIFT: O primeiro passo do experimento é comparar a visão do robô com a visão de referência para que o mesmo possa encontrar pontos em comum entre as duas;
- *Visual Tracking*: A estratégia de *Visual Tracking* proposta na seção 6.4 é investigada aqui quando utilizada para guiar a navegação do robô;
- Navegação para nós desconhecidos: A navegação para nós desconhecidos é feita como neste experimento. A única diferença está em que a imagem de referência utilizada aqui não foi extraída de uma vista panorâmica. Neste experimento também não foi estabelecida uma condição de parada para que se pudesse fazer uma avaliação ao longo da navegação e após o robô ter passado pela posição de referência;
- Navegação para nós novos: A navegação para nós novos não será abordada em um experimento isolado dado que no presente experimento as etapas que compõem a navegação para nós novos já são abordadas;

Como já foi comentado, no experimento o robô navegou de um ponto dado para uma posição onde havia obtido uma imagem. A Figura 7-132 mostra à esquerda a imagem que foi utilizada como referência e à direita a imagem na posição em que o robô começou a navegação.

Figura 7-132: Diferentes visões do robô

A navegação do experimento constituiu de 31 iterações e de um percurso de aproximadamente 70 cm ao todo. Para cada iteração o robô realizava um pequeno movimento de correção de sua posição. A posição do robô mais próxima da posição de referência foi encontrada na iteração 18. A partir de então o robô continua corrigindo sua direção, porém, se afasta da posição de destino.

Lembre que o sistema experimental ER1 impõe limitações à movimentação do robô. Ao mesmo tempo, o processamento do algoritmo pelo programa *Matlab* é lento fazendo com que o tempo entre iterações tenha sido aproximadamente o tempo da conclusão dos menores movimentos permitidos pelo ER1.

O experimento será apresentado em 3 seções:

- 7.7.1 - SIFT: Mostra o resultado da busca por pontos em comum entre as duas imagens utilizando-se de descritores SIFT;
- 7.7.2 – *Visual Tracking*: Apresenta algumas da imagens vistas ao longo da navegação com pontos de referência e ponto chave;
- 7.7.3 – Condição de parada: Discussão sobre a condição de parada e uma breve validação do uso de Fourier Mellin para se comparar imagens;

7.7.1. SIFT

A configuração utilizada para encontrar os pontos em comum através do método SIFT foi:

- 4 Oitavas;
- 1 Intervalo;

- Limiar de contraste igual a 0,03;
- Limiar de curvatura igual a 10,0;
- Aplicação da transformação de Hough, RANSAC e matriz de pesos W com parâmetros seguindo os padrões indicados ao longo do presente trabalho;

A Figura 7-133 apresenta os pontos em comum entre as duas vistas do robô. À esquerda está a imagem que foi utilizada como referência e à direita a imagem na posição em que o robô começou a navegação.

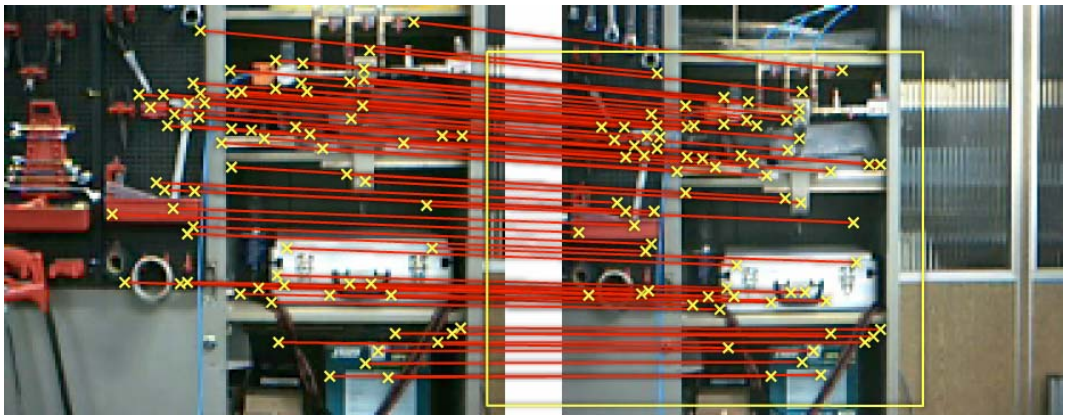

Figura 7-133: Pontos encontrados em comum

É válido ressaltar que o número de pontos encontrados em comum entre as vistas é de grande importância para o bom funcionamento do *Visual Tracking*. Dos pontos encontrados é que serão escolhidos os pontos de referência. Quanto maior o número de pontos de referência, melhor a eficiência do *Visual Tracking*.

Caso o número de pontos encontrados em comum seja muito pequeno, a navegação pode-se dar de modo deficiente. Portanto, propõe-se que não se trabalhe com nós a distâncias muito grandes ou, se for o caso, que sejam utilizadas imagens com maior resolução. Foi visto no experimento da seção 7.4 que o método SIFT empregado pode não encontrar muitos pontos em comum entre duas imagens.

7.7.2. *Visual Tracking*

O *Visual Tracking* foi feito utilizando-se a seguinte configuração:

- Tamanho da janela do modelo: $Sx1 = Sy1 = 16$;
- Tamanho da janela de busca: $Sx2 = Sy2 = 42$;
- Limiar de correlação α igual a 0,8;
- ρ igual a 0,95;
- β igual a 0,1;
- Não trabalhou-se com Hough, RANSAC e matriz de pesos W;

Da Figura 7-134 à Figura 7-141 são observadas as imagens obtidas pelo robô ao longo das iterações 1, 5, 10, 15, 18, 20, 25 e 30 do algoritmo de navegação. As cruzes vermelhas representam os pontos de referência, o círculo azul representa o ponto chave que é usado para corrigir a direção do robô e o quadrado azul representa o enquadramento da imagem de referência na imagem atual. Este enquadramento terá suas extremidades usadas para se computar a distância entre estas e as extremidades da imagem como será visto na seção seguinte.

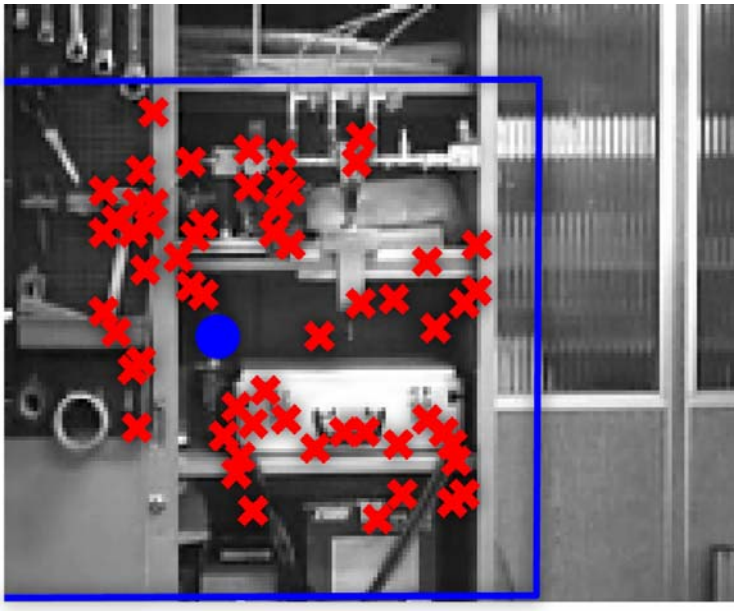

Figura 7-134: Iteração 1

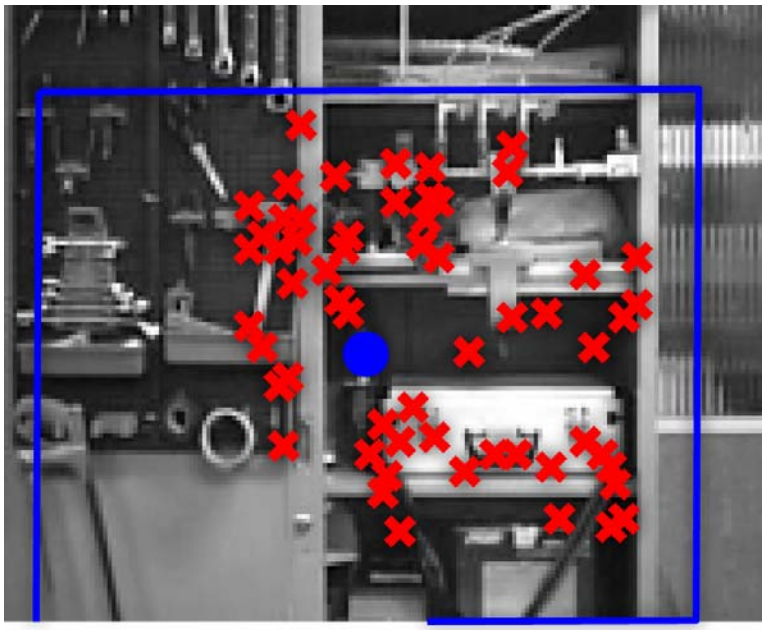

Figura 7-135: Iteração 5

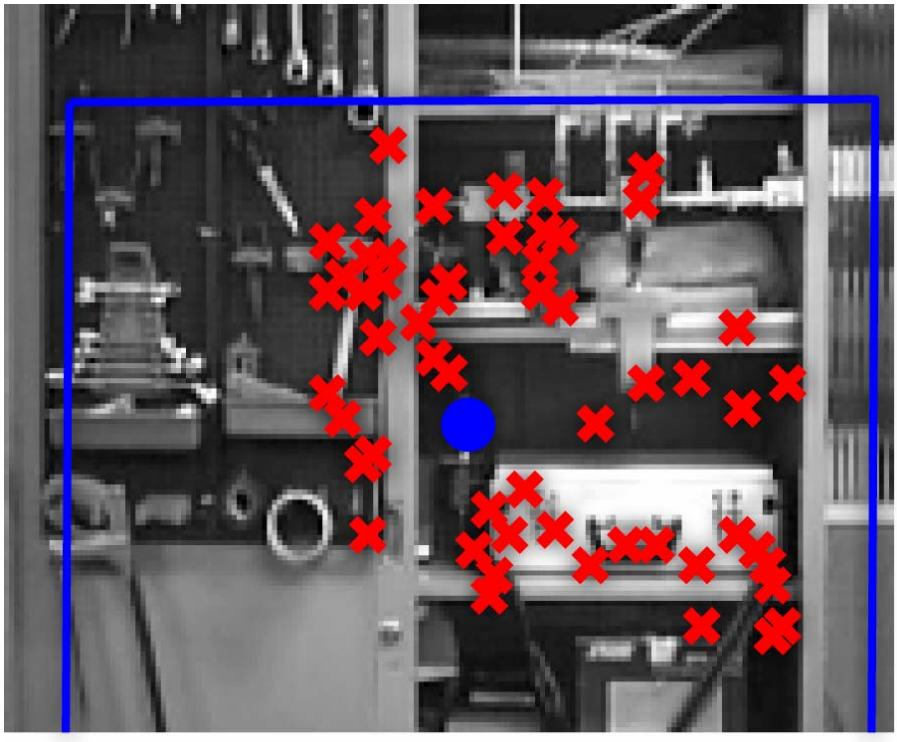

Figura 7-136: Iteração 10

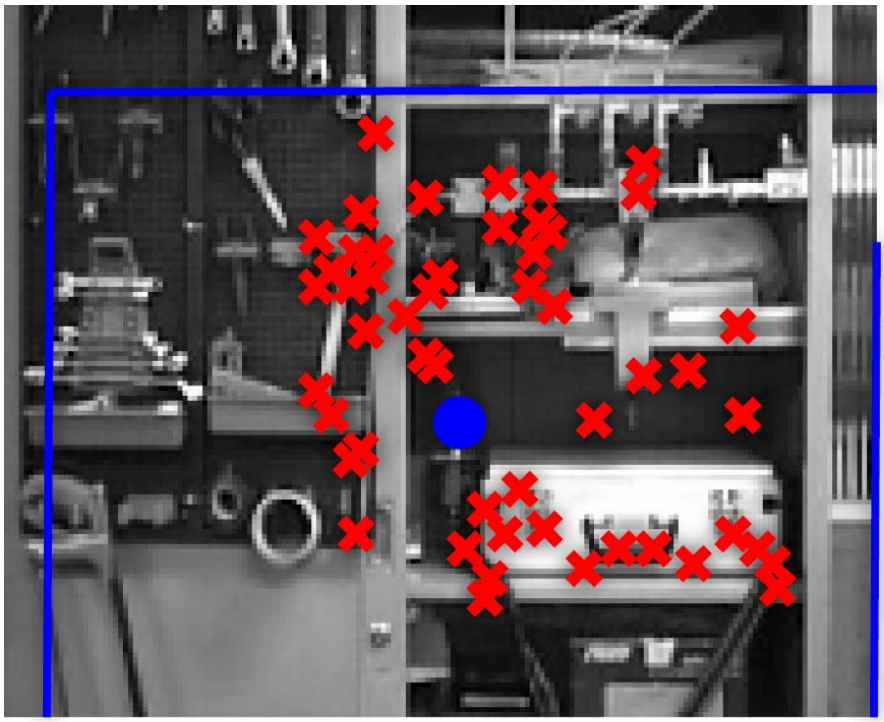

Figura 7-137: Iteração 15

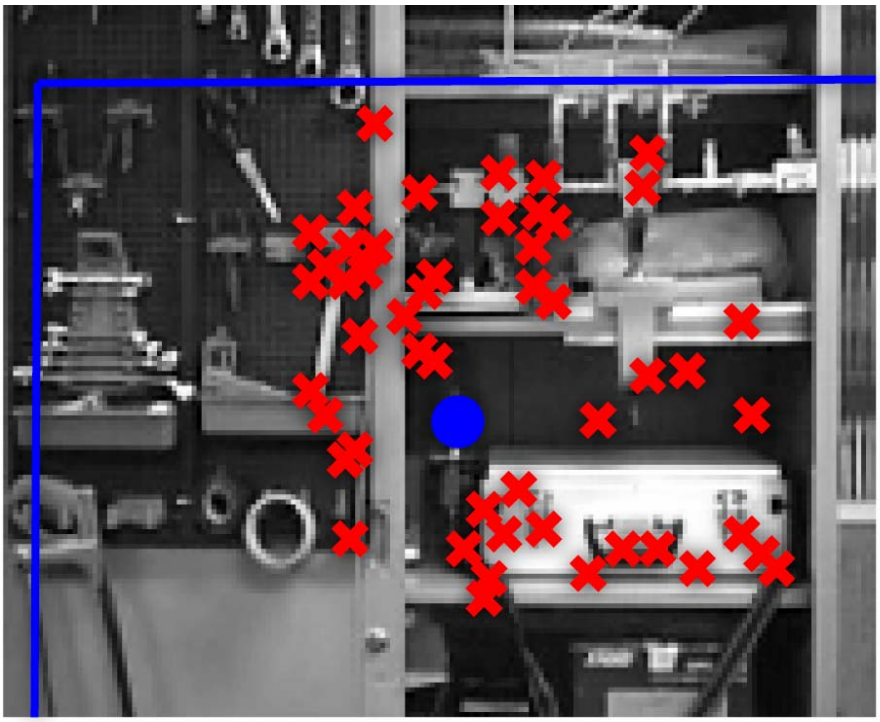

Figura 7-138: Iteração 18

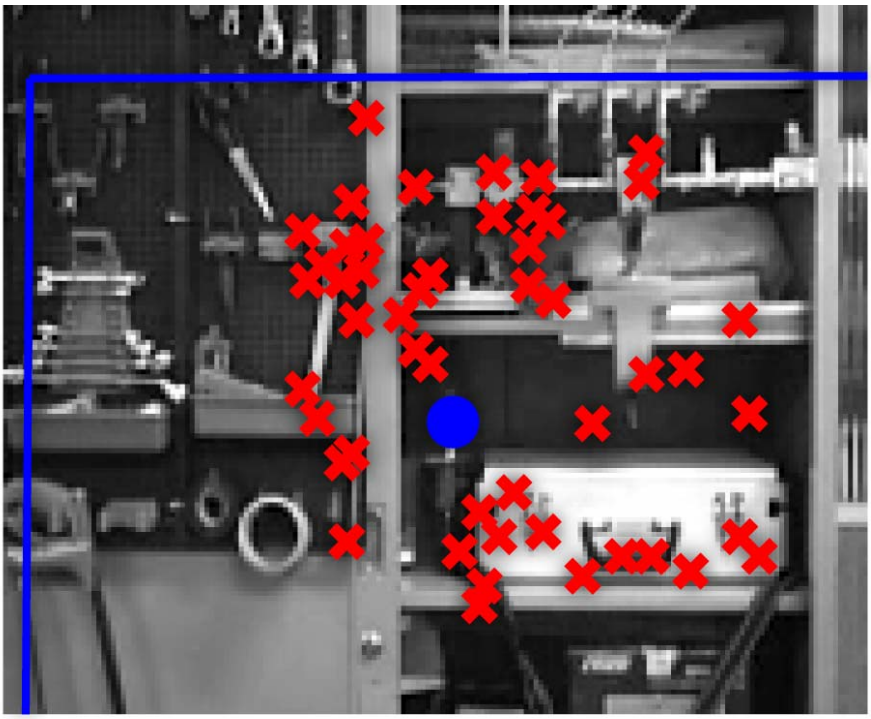

Figura 7-139: Iteração 20

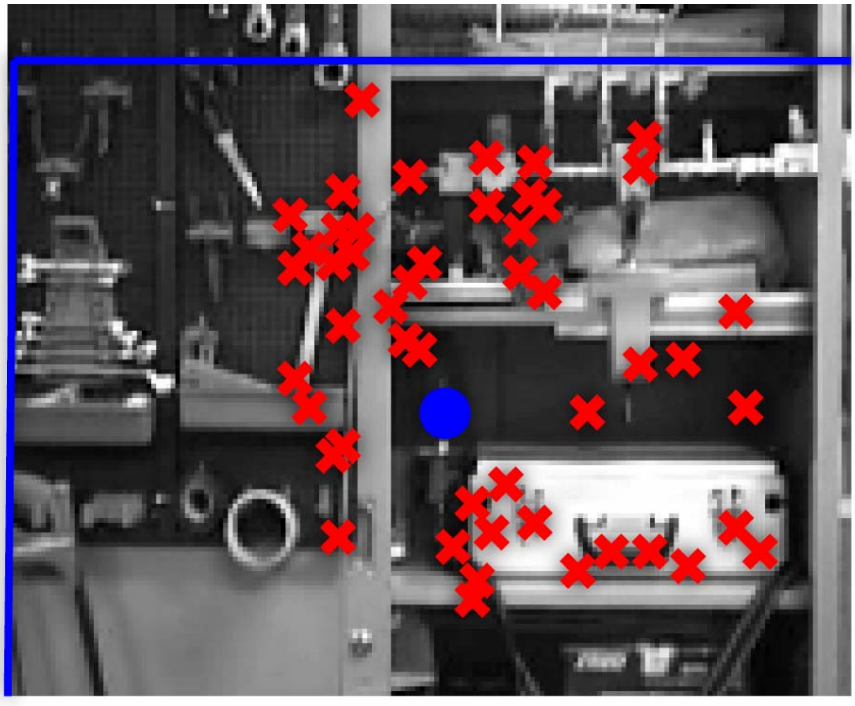

Figura 7-140: Iteração 25

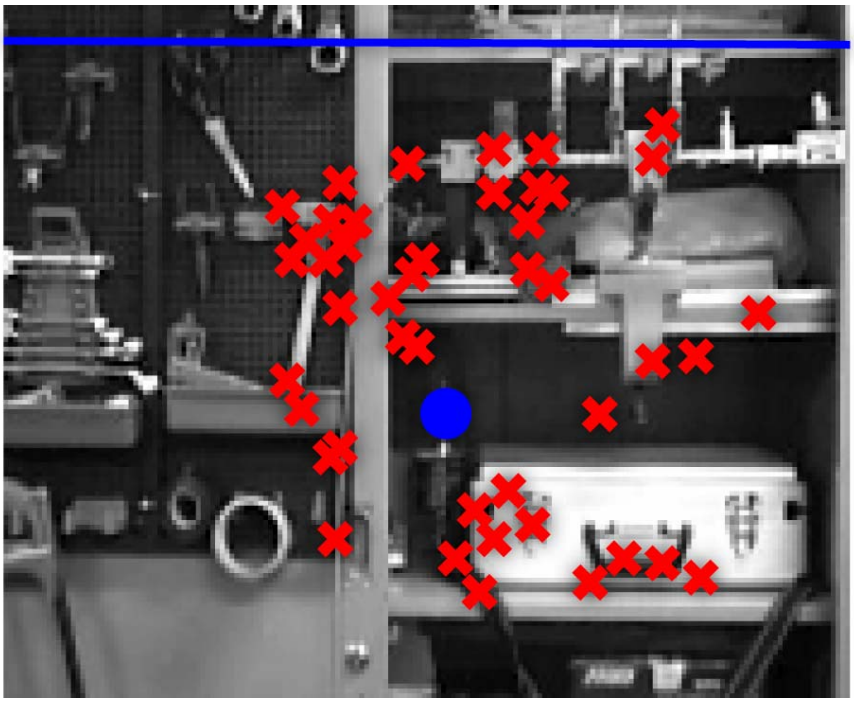

Figura 7-141: Iteração 30

Através das imagens pode-se ver que o robô conseguia corrigir os erros de direção e encontrar de maneira consistente o ponto chave através das várias imagens. A idéia de se usar diversos pontos de referência para se calcular o ponto chave está em conseguir manter esta estabilidade para o ponto chave dado que pequenos erros em se encontrar os pontos de referência não afetam muito a posição encontrada do ponto chave.

Ao longo do projeto verificou-se que quanto maior o número de pontos de referência, maior a estabilidade em encontrar o ponto chave em uma imagem observada pelo robô. Também é interessante se ter muitos pontos de referência pois estes podem ser descartados ao longo do tempo caso não estejam sendo mais encontrados de forma consistente.

Neste experimento não se utilizou Hough, RANSAC e matriz de pesos W. O uso destas técnicas melhora muito a eficiência do método quando pontos de referência são encontrados erroneamente. Entretanto, o uso destas técnicas implica em maior custo computacional.

7.7.3. Condição de parada

Na seção 6.6.2 descreve-se o uso de diferentes possibilidades de condição de parada para o robô quando este navega para um lugar conhecido. Foram apresentadas 3 possibilidades principais para condições de parada:

- Observando a correlação entre a imagem observada e a imagem de referência;
- Observando a comparação através de transformações invariáveis entre a imagem observada e a de referência;
- Verificando a distância média entre as extremidades da imagem e as extremidades do enquadramento da imagem de referência na imagem atual;

Tanto a distância média quanto à comparação por transformações invariáveis são investigadas aqui.

A Figura 7-142 apresenta a distância média entre as extremidades do enquadramento da imagem de referência e as extremidades da imagem atual para todas as iterações feitas. Esta distância média é chamada de erro das extremidades. Na figura, a cruz vermelha indica o melhor valor obtido, o que se deu na iteração 18.

Não só neste experimento como ao longo do projeto, percebeu-se que esta distância média para as extremidades representa um bom indicador do momento de parada do robô. No experimento em questão pode-se verificar que o melhor valor encontrado coincidiu com a melhor posição do robô.

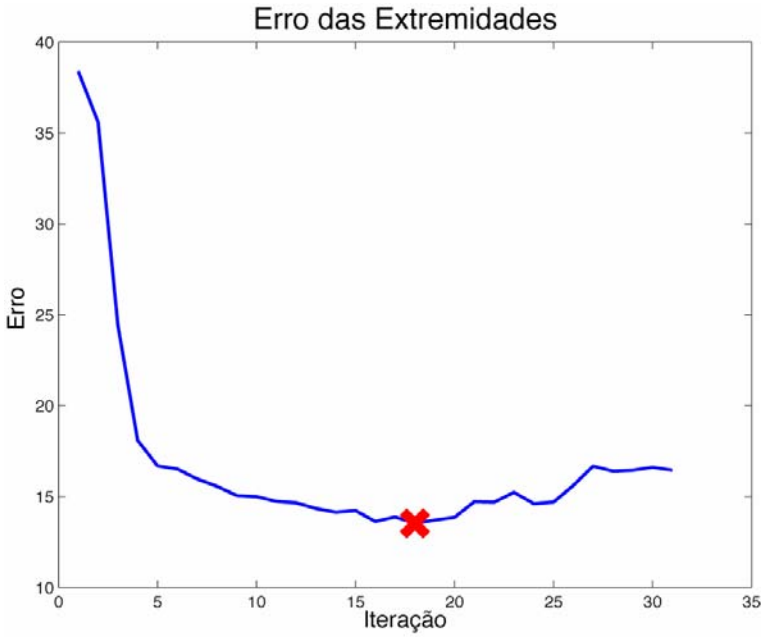

Figura 7-142: Erro médio das extremidades

Por fim, para cada imagem obtida pelo robô, foi feita uma comparação da mesma com a imagem de referência através de Fourier Mellin do tipo 1 de modo a validar novamente o uso desta transformação em comparações.

A Transformada de Mellin foi feita com a seguinte configuração:

- $\Delta u = \Delta v = 0.1$;
- Variáveis u e v variando de 1 a 100: Resultado com resolução de 100 por 100;

A Figura 7-143 apresenta o erro da comparação para cada iteração. Pode-se verificar que os melhores valores são encontrados próximos da iteração 18, que é a iteração na qual o robô obteve a melhor posição. Entretanto, o melhor valor de comparação é o da iteração 16 (marcado pela cruz vermelha). Isto mostra que a comparação Fourier Mellin apresentou um bom indicativo do momento de parada do robô, tal como se mostrou uma boa comparação entre as imagens, apesar de não ser perfeita.

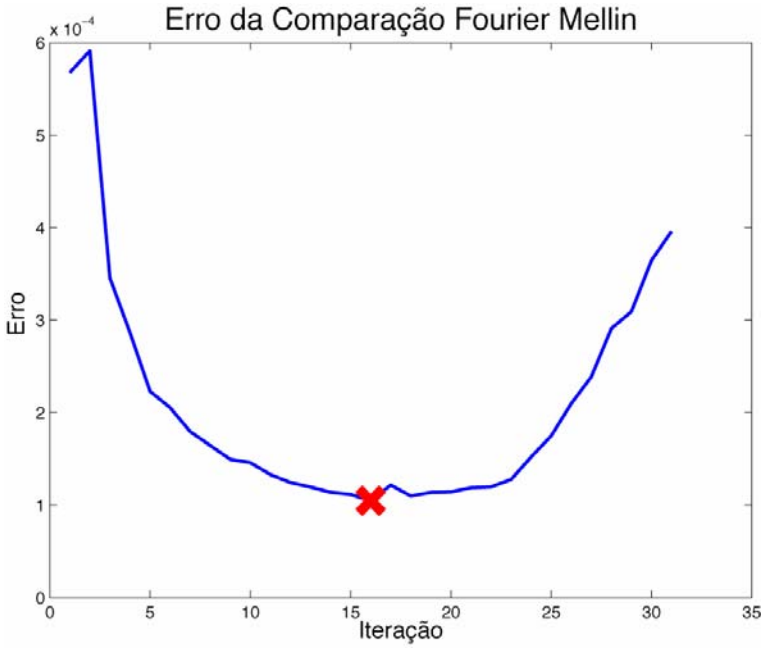

Figura 7-143: Erro de comparação do tipo Fourier Mellin

Ao longo do projeto percebeu-se em experimentos do gênero que as comparações baseadas nas transformações de Fourier Mellin do tipo 1 e 2 podem ser utilizadas para auxiliar a parada do robô e são comparações superiores à correlação simples entre duas imagens.

7.8. Refinando o modelo utilizando Algoritmos Genéticos

Para testar o desempenho do Algoritmo Genético, este foi empregado na adição de adjacências para dois grafos diferentes, um de 10 nós, outro de 25 nós, denominados respectivamente de grafo *Espiral* e grafo *Robô* (ver Figura 7-144 e Figura 7-146). Para alguns dos testes, o desempenho do A.G. é comparado ao desempenho de uma busca aleatória. Foram realizados alguns testes para encontrar-se a melhor combinação de parâmetros para o A.G. e são apresentados aqui testes realizados com a seguinte configuração:

- População de 100 indivíduos;
- i de 90 indivíduos;
- *Crossover* em 80% ;
- Mutação em 5% ;
- Normalização entre 0 e 100;

Também foram feitos 2 testes com os mesmos grafos porém sem adjacências (ver Figura 7-148), procurando-se adicionar o número mínimo de adjacências de maneira a cobrir todos os nós e encontrar o melhor resultado.

Devido ao fato do processamento do A.G. ser lento, principalmente para um número de nós elevado, os resultados apresentados são relativos a somente um teste para cada experimento, e não a uma média relativa a vários experimentos. Os testes apresentados foram escolhidos dentre vários testes realizados por sua maior relevância quanto à discussão dos resultados obtidos.

7.8.1. Grafo “Espiral”

O primeiro grafo a ser apresentado possui 10 nós e tem o formato de uma espiral. A Figura 7-144 e a

Figura 7-145 apresentam respectivamente os melhores resultados obtidos para algumas adições de adjacências e gráficos comparativos da evolução do A.G. e busca aleatória. Perceba que o número de gerações varia de acordo com as experiências devido ao fato de algumas levarem mais tempo para convergirem.

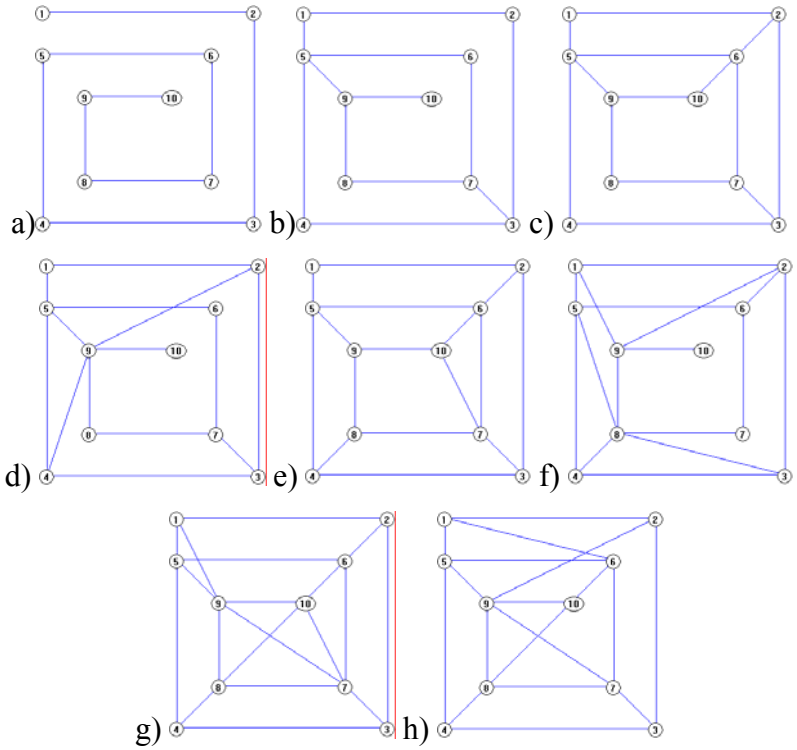

Figura 7-144: (a) Grafo *Espiral* sem adição de adjacências, (b) Adição de 3 adjacências pelo A.G e busca aleatória, (c) Adição de 5 adjacências pelo A.G, (d) Adição de 5 adjacências pela busca aleatória, (e) Adição de 7 adjacências pelo A.G, (f) Adição de 7 adjacências pela busca aleatória, (g) Adição de 10 adjacências pelo A.G, h) Adição de 10 adjacências pela busca aleatória

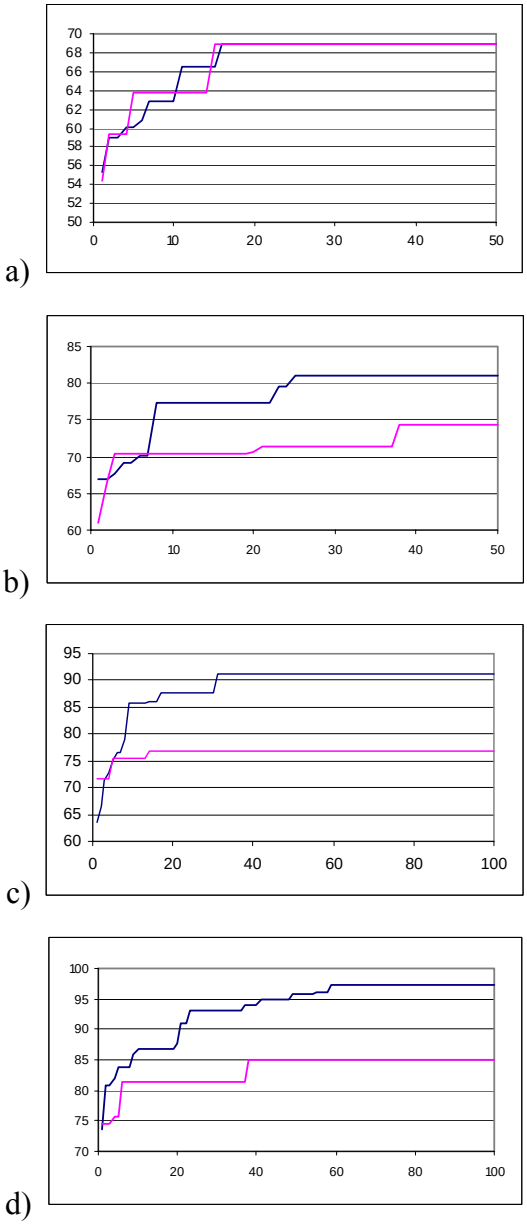

Figura 7-145: Aptidão dos melhores indivíduos x Gerações para grafo Espiral - (a) Adição de 3 adjacências, (b) Adição de 5 adjacências, (c) Adição de 7 adjacências, (d) Adição de 10 adjacências. OBS: As curvas em rosas são referentes à busca aleatória e as curvas azuis são referentes ao A.G.

Pode-se verificar que o A.G. apresenta bons resultados para grafos com pequeno número de nós em poucas gerações. A superioridade do A.G. em relação à busca aleatória se mostra mais claramente com um número maior de adjacências acrescentadas ao grafo. Isto se deve ao fato de que quando o espaço de busca é maior, a busca aleatória é lenta e pouco eficaz. Para a adição de poucas adjacências, não é verificada grande diferença entre o A.G. e a busca aleatória.

Para os testes efetuados com adições de até 7 adjacências, o A.G. demonstrou encontrar soluções ótimas. Para a adição de 10 adjacências, o resultado encontrado é sub-ótimo. Isto acontece, entre outros possíveis motivos, pois foi percebido ao acompanhar os cromossomos criados, que os melhores indivíduos tendiam a se repetir ao longo das gerações, levando a uma convergência precipitada do algoritmo para configurações sub-ótimas. Para resolver tal problema ficam propostas algumas técnicas:

- Trabalhar com representações do problema onde diferentes cromossomos não representam o mesmo resultado;
- Trabalhar com *Steady State* sem duplicados. Na implementação feita, o uso de *Steady State* leva a um grande número de indivíduos replicados, fazendo com que não exista grande variação nas populações criadas, o que leva à convergência do algoritmo;
- Interpolar os parâmetros do A.G. A interpolação dos parâmetros permite que quando o algoritmo estiver convergindo para um resultado sub-ótimo, se diversifique a população, seja aumentando a probabilidade de mutação do algoritmo, seja diminuindo a pressão seletiva sobre os melhores indivíduos;
- Incluir novos operadores ao A.G. que permitam seu melhor desempenho e possibilitem uma diversificação da população de forma mais eficaz. Um possível operador é o de mutação por adjacências já apresentado em 3.2 Operadores Utilizados;.
- Rodar o algoritmo algumas vezes, semeando as novas populações com os melhores indivíduos dos experimentos anteriores;

7.8.2. Grafo “Robô”

O segundo grafo apresentado possui 25 nós e tem o formato de uma árvore. Este grafo é uma possibilidade de modelo retornado por um robô após a exploração do ambiente e por isso foi denominado *Robô*. Resultados na Figura 7-146 e na Figura 7-147.

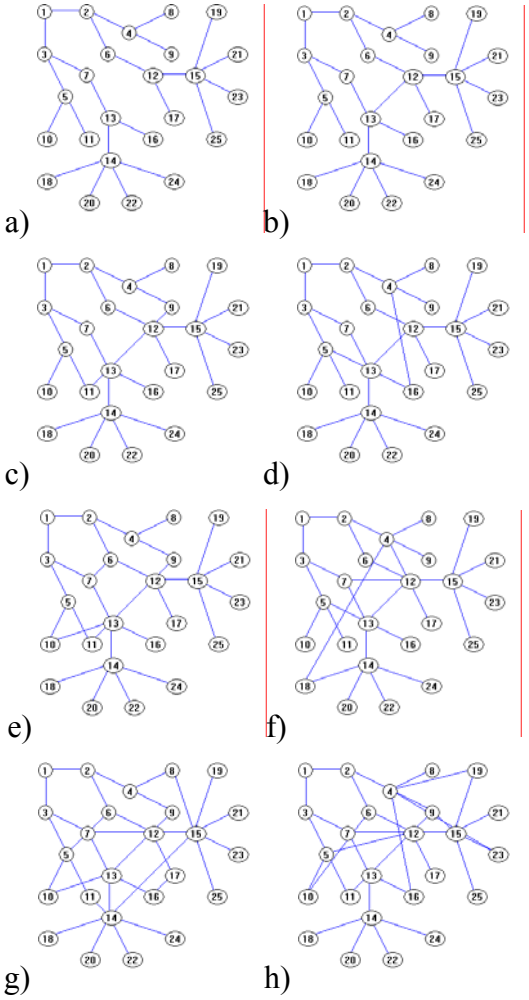

Figura 7-146: (a) Grafo *Robô* sem adição de adjacências, (b) Adição de 1 adjacência pelo A.G e busca aleatória, (c) Adição de 3 adjacências pelo A.G, (d) Adição de 3 adjacências pela busca aleatória, (e) Adição de 5 adjacências pelo A.G, (f) Adição de 5 adjacências pela busca aleatória, (g) Adição de 10 adjacências pelo A.G, h) Adição de 10 adjacências pela busca aleatória.

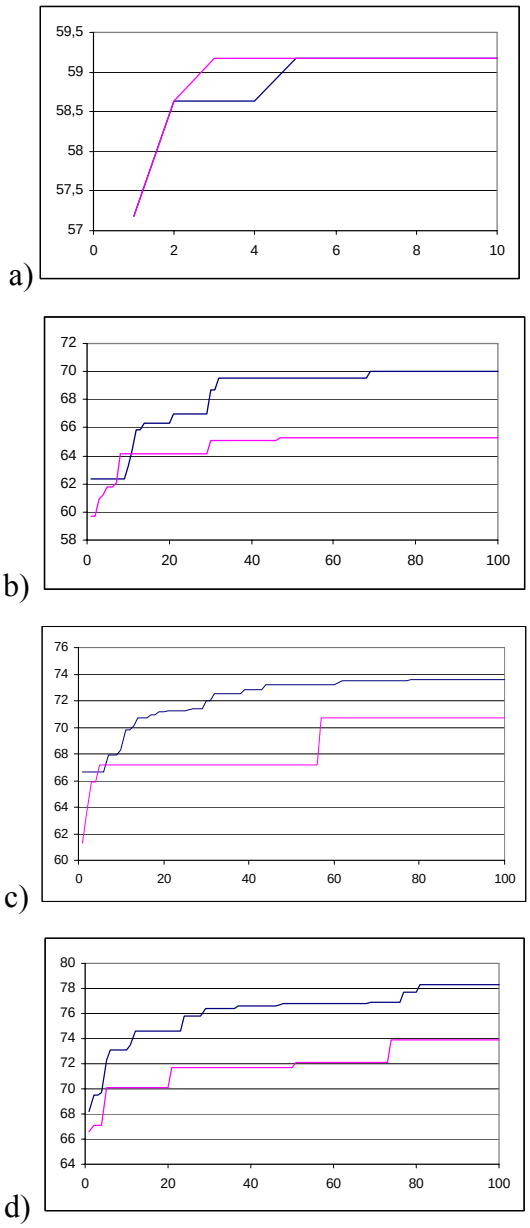

Figura 7-147: Aptidão dos melhores indivíduos x Gerações para grafo *Robô* - (a) Adição de 1 adjacência, (b) Adição de 3 adjacências, (c) Adição de 5 adjacências, (d) Adição de 10 adjacências. OBS: As curvas em rosa são referentes à busca aleatória e as curvas azuis são referentes ao A.G.

Novamente pode-se perceber que o Algoritmo Genético retornou bons resultados. Como para o grafo *Espiral*, com o aumento do número de adjacências acrescentadas, e conseqüentemente, o aumento do cromossomo, os resultados obtidos não são mais ótimos. Porém, para conseguir melhores resultados, podem ser aplicadas às idéias propostas anteriormente.

Os comentários relativos ao grafo *Espiral* se aplicam ao grafo R*obô*.

7.8.3.
Grafos sem arcos

Os resultado para o grafo sem arcos podem ser conferidos na Figura 7-148 e na Figura 7-149.

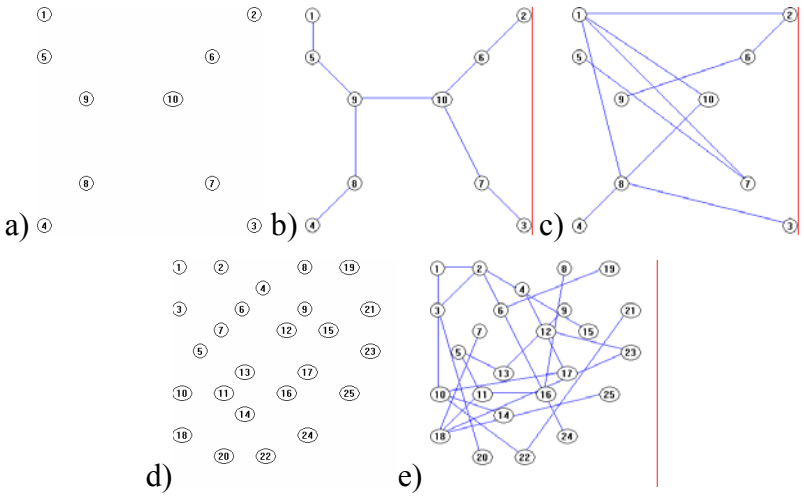

Figura 7-148: (a) Grafo *Espiral Sem Arcos* sem adição de adjacências, (b) Grafo *Espiral Sem Arcos* com adição de 1 adjacência pelo A.G, (c) Grafo *Espiral Sem Arcos* com adição de 1 adjacência pela busca aleatória, (d) Grafo *Robô Sem Arcos* sem adição de adjacências, (e) Grafo *Robô Sem Arcos* com adição de 1 adjacência pelo A.G.

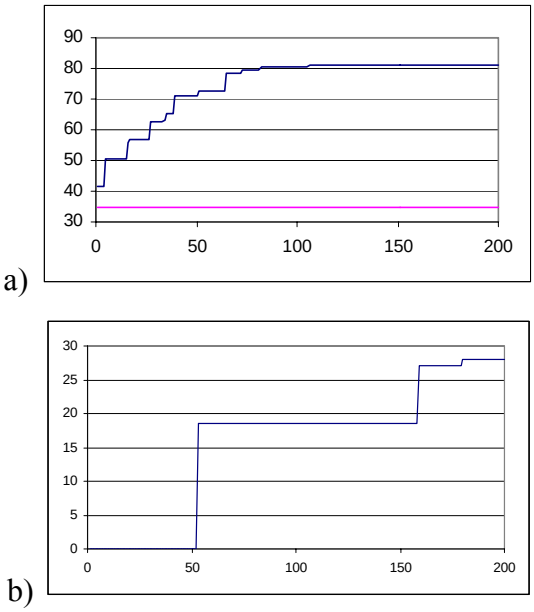

Figura 7-149: Aptidão dos melhores indivíduos x Gerações - (a) Adição de 9 adjacências no grafo *Espiral sem Arcos*, (b) Adição de 24 adjacências no grafo *Robô sem arcos*. OBS: As curvas rosas são referentes à busca aleatória e as curvas azuis são referentes ao A.G.

Ao se utilizar o Algoritmo Genético como proposto, buscando resolver o problema de escolher a melhor combinação para um número mínimo de adjacências, dado um conjunto de nós, foi encontrada boa solução para o caso de 10 nós. Porém, o algoritmo não obteve boa evolução para o conjunto de 25 nós testado. O mau resultado se deve ao fato da função de avaliação retornar 0 para cromossomos que não representam soluções possíveis, fazendo com que o algoritmo demore em encontrar cromossomos viáveis e não possa desenvolver melhores cromossomos até então. Para resolver este problema, fica proposto gerar uma população inicial de indivíduos que já contenham cromossomos que representem soluções plausíveis, ou, seja alterada a função de avaliação de modo a conseguir avaliar estes cromossomos.

8
Conclusões e Trabalhos Futuros

A presente dissertação apresentou uma proposta de algoritmo para exploração e mapeamento completa para robôs móveis na seção 6.1. Ao longo da dissertação os componentes deste algoritmos foram descritos e investigados. O capítulo 7 mostrou alguns experimentos que ilustraram estes componentes, suas capacidades e suas deficiências.

Ao longo do projeto, todos os métodos propostos foram implementados e estudados para diversas configurações possíveis. Entretanto, não foi apresentado aqui a integração dos componentes propostos de modo a se fazer a exploração do ambiente como proposto na seção 6.1. Esta integração fica como proposta para um trabalho futuro.

Serão apresentadas aqui conclusões relativas aos experimentos vistos tal como conclusões gerais observadas ao longo do projeto. Este capítulo está dividido em seções devido ao fato da presente dissertação apresentar uma série de técnicas diferentes. Cada seção apresenta as conclusões relativas ao tópico específico tal como propostas para trabalhos futuros. Por fim são apresentadas conclusões e propostas gerais sobre o presente trabalho.

8.1.
Transformações Invariáveis

Nos experimentos apresentados na seção 7.1 foi possível verificar visualmente que a implementação realizada gerou resultados dentro do domínio discreto de imagens bitmap compatíveis com as invariâncias esperadas e verificadas matematicamente para o domínio contínuo. Entretanto, algumas pequenas variações foram visualisadas. Estas variações são consequentes da aplicação ser feita no domínio contínuo e do fato de que algumas transformações (tal como escala) serem feitas sobre as imagens utilizando técnicas de aproximação como interpolação.

As pequenas variações encontradas mostraram que algumas Transformadas possuem grande sensibilidade a determinadas variações na imagem.

Os experimentos apresentados mostraram que o uso das transformações propostas foi adequado para se fazer comparação entre imagens que sofreram variações em escala, rotação e translação, sendo que cada uma das transformadas é mais adequada para algumas destas variações.

Pode-se observar também que o uso de correlação é superior ao uso de distâncias euclidianas para se fazer a avaliação das comparações.

8.2. Window Growing

Não só pelas experiências apresentadas, mas também pela série de testes que foram feitos ao longo do projeto, pode-se perceber que os melhores resultados foram verificados com o uso da Transformada de Fourier Mellin do tipo 1 e da do tipo 2. O uso da comparação por correlação após escala das imagens também demonstrou resultados próximos aos obtidos com as transformações invariáveis.

Porém, as comparações globais utilizadas têm algumas limitações e são extremamente sensíveis. As limitações encontradas são:

- O método não funciona para imagens que possuam objetos a diferentes distâncias do robô e que impliquem em nenhuma imagem centralizada ser parecida com a imagem de referência;
- Para o caso da câmera não estar bem nivelada, a imagem de referência pode não aparecer centralizada quando vista à distância. O método é sensível ao nivelamento da câmera;
- O método não funciona caso ocorram oclusões de áreas na imagem;
- O método não funciona caso apareçam informações visuais diferentes das verificadas na imagem de referência dentro da área centralizada que seria referente à imagem de referência quando o robô está à distância;

Portanto, devido as limitações apresentadas, o uso de comparações locais utilizando o método SIFT mostrou ser uma opção mais adequada dentro do contexto do presente trabalho.

A aplicação das comparações globais apresentadas para auxílio na navegação foi desenvolvida e testada em um primeiro momento. Ao serem observadas suas deficiências optou-se por trabalhar com a técnica SIFT que demonstrou ser uma boa escolha.

8.3. SIFT

Não só dentro dos exemplos apresentados mas durante uma série de testes feitos, a aplicação de descritores SIFT para se encontrar pontos em comum entre imagens se mostrou extremamente eficaz e poderosa. Mesmo em casos onde ocorre oclusão, ou em casos em que se tem objetos a diferentes distâncias dentro da visão do robô, a técnica mostrou não ter encontrado grandes problemas, com poucas exceções.

As dificuldades encontradas foram em se trabalhar com imagens a grandes distâncias onde os objetos vistos variavam muito em escala. Este problema pode ser facilmente resolvido trabalhando-se com resoluções superiores ou então limitando os percursos entre nós durante o processo de exploração.

Apesar da transformação SIFT ser adequada a uma série de aplicações, exige alto custo computacional. Isto implica em ser necessário utilizá-la somente em aplicações mais críticas e trabalhar com técnicas mais simples durante tarefas que necessitam de processamento em tempo real (5 a 10 Hz nos experimentos realizados).

8.4. Panorâmicas

Apesar de o algoritmo não garantir que as imagens sejam alinhadas corretamente, isto só deve vir a ocorrer em imagens com baixa quantidade de informação, sem elementos visuais que se distinguam do resto da imagem para que possam ser utilizados ao se fazer o alinhamento. Outro possível caso de problemas de alinhamento é quando a área em comum entre as duas imagens é muito pequena.

Um modo de garantir maior perfeição de alinhamento seria seguir alguma outra estratégia de estimação de parâmetros que não pela pseudoinversa. A

estratégia obtida garante que a transformação levará ao menor erro possível para os pontos de controle obtidos e tem como vantagem baixíssimo tempo de processamento. Existem estratégias diversas de estimação de parâmetros que buscam minimizar o erro entre a imagem transformada e a imagem de referência.

Quando as duas imagens possuem muitos objetos com diferentes profundidades, podem surgir imperfeições de alinhamento devido à paralaxe. Um modo de corrigir estas imperfeições seria trabalhando com transformações locais.

8.5 *Quadtree*

Pode-se verificar que com o uso de parâmetros adequados, a segmentação realmente aponta áreas com maior detalhamento na imagem, ou, com maior quantidade de informação. A técnica demonstrou conseguir detectar regiões com maior quantidade de informação nas imagens sendo viável de ser usada dentro da proposta do presente projeto. É interessante ressaltar que a escolha de regiões com maior quantidade de informação facilita a navegação do robô pois os métodos utilizados ao longo da navegação dependem de áreas com bastante informação para serem efetivos. Isto acontece tanto para o uso de *features* SIFT quanto para a comparação de transformações invariáveis.

Muitas vezes o número de regiões correspondentes a menor divisão *quadtree* é muito pequeno. Quando isto ocorre, independente do método escolhido, os resultados não são muito bons e é de interesse utilizar também outros tamanhos de divisões *quadtree* que não somente o tamanho menor encontrado.

Se forem usadas regiões de segundo menor tamanho ou até mais, pode-se conseguir resultados ainda melhores. Portanto é aconselhável que se faça uma verificação do número de regiões mínimas após o *quadtree* de modo a se determinar quantos níveis de tamanho para as regiões é interessante de se trabalhar.

8.6
Visual Tracking

O método apresentado, apesar de simples, consegue apresentar os seguintes resultados dentro do sistema experimental utilizado:

- O robô consegue fazer o acompanhamento de pontos na tela desde que sua posição não seja alterada por fatores externos;
- Pequenas variações luminosas são compensadas pela técnica;
- O acompanhamento dos pontos observados é correto ao longo da movimentação executada pelo próprio robô;
- A técnica consegue lidar com oclusões temporárias (alguns segundos), mesmo que para toda a imagem;

Apesar de métodos mais complexos poderem ser usados, estes implicariam em perda de eficiência (quanto a velocidade de processamento) e poderiam não ser viáveis para o sistema usado.

8.7
Navegação auxiliada por *Visual Tracking*

Ao longo do projeto a navegação auxiliada por *Visual Tracking* demonstrou ser melhor que a navegação auxiliada por comparações baseadas em transformações invariáveis nos devidos aspectos:

- Maior robustez quanto a variações na iluminação, no posicionamento da câmera (inclinações por exemplo), na composição das cenas observadas, entre outros;
- Maior robustez quanto a oclusões temporárias na imagem. Isto permitiria o uso dos algoritmos propostos em ambientes não estáticos;
- Melhor funcionamento de um modo geral. A navegação auxiliada por comparações baseadas em transformações invariáveis possui sérias restrições que impossibilitam seu uso em determinadas situações;

O uso de descritores SIFT mostrou ser extremamente adequado, permitindo reencontrar os pontos de referência caso o robô por algum motivo não tenha pontos suficientes para se localizar e corrigir sua direção. A restrição em trabalhar

com descritores SIFT está em limitar a distância entre os nós para permitir que pontos sejam encontrados entre as diferentes visões do robô. Esta restrição pode ser balanceada escolhendo uma resolução para as imagens adequada à distância máxima entre os nós desejada.

Outra restrição do uso de SIFT está em seu custo computacional. Este problema foi contornado quando não se trabalhou com SIFT ininterruptamente, ou apenas somente em determinados momentos dos procedimentos propostos.

Quanto à estratégia de *Visual Tracking* proposta, pode-se dizer que esta se mostrou adequada para as tarefas em questão. A configuração apresentada no experimento visto mostrou-se bastante adequada. Mesmo assim ainda são propostas técnicas como Hough e RANSAC que possibilitariam melhora dos resultados com a desvantagem de maior custo computacional. As ferramentas utilizadas possibilitam configurações diversas de modo a adequar o *Visual Tracking* às necessidades encontradas.

Por fim, cabe citar que um problema observado é devido ao fato de que os movimentos mínimos do robô ER1 são grandes e de que o processamento no programa Matlab é relativamente lento, e portanto pode-se ter alguma instabilidade na correção da direção.

Fica a sugestão de se trabalhar com outros métodos de navegação auxiliados por *Visual Tracking*, porém sem deixar de se utilizar descritores SIFT dado que estes demonstraram ser uma boa adição aos métodos propostos.

8.8
Refinando o Modelo Criado Utilizando A.G.

O problema de adicionar novas adjacências a um grafo é tratado através de um Algoritmo Genético gerando bons resultados. Percebe-se que quanto maior o número de nós, maior a dificuldade associada ao problema, e melhores os resultados obtidos pelo Algoritmo Genético em relação à busca aleatória. A função de avaliação utilizada permitiu ainda o uso do A.G. para tentar resolver o problema de escolher a melhor combinação para um número mínimo de adjacências. Porém, os resultados obtidos só foram bons para um número pequeno de nós. Para resolver o problema de um grande número de nós, seria necessário

fazer modificações na função de avaliação ou semear a população inicial com indivíduos válidos.

São propostas diversas sugestões visando resultados ainda melhores, entre elas:

- Uso de *Steady State* sem duplicados;

- Trabalhar com diversos experimentos, semeando novas populações com os melhores indivíduos de populações anteriores;

- Interpolar os parâmetros do A.G.;

- Trabalhar com operadores diferentes dos utilizados;

Verificou-se a possibilidade da aplicação do método implementado não somente na aplicação proposta em robótica mas também no planejamento de rodovias, rotas aéreas ou redes de computadores, como alguns exemplos.

8.9
Conclusões e Propostas Gerais para Trabalhos Futuros

O presente trabalho apresentou uma variedade de técnicas e possibilidades para se fazer a exploração e mapeamento de um ambiente como proposto na seção 6.1. Como já observado, não foi feita a integração das componentes estudadas e desenvolvidas. Esta é portanto a principal proposta para trabalhos futuros de um modo geral.

As componentes desenvolvidas apresentaram bons resultados individualmente demonstrando estarem preparadas para a integração em um trabalho futuro.

Outra proposta que acrescentará muito às implementações feitas é se trabalhar com um sistema experimental mais adequado e menos limitado que o utilizado.

A primeira carência do sistema experimental está no robô utilizado. Como apontado diversas vezes este possui uma série de limitações, dentre as quais:

- Movimento restrito a rotações e trajetórias em linha reta. Não é possível acessar os motores do robô diretamente para se controlar as velocidades individuais de cada roda. Portanto, é impossível descrever trajetórias curvilíneas.

- Não há acesso direto às contagens referentes aos *encoders* de cada roda. Só é possível saber a posição do robô através das coordenadas *(x,y)*. Deveria ser possível aquisitar também a coordenada θ relativa à direção do robô, porém descobriu-se que a API apresenta um problema que não foi nem será resolvido que impossibilita se adquirir esta coordenada (informação dada pelo suporte da empresa *Evolution Robotics*™).
- A resposta ao comando *"move"* apresenta discrepâncias em relação às distâncias percorridas ou giradas em relação às distâncias pedidas. Isto significa que é necessário estar checando a posição do robô para se corrigir possíveis erros. Porém, isto não é possível de ser feito quando o robô gira porque não é possível se obter a coordenada de giro do robô como já discutido. Estes problemas não deveriam ser esperados já que o robô trabalha com motores de passo e *encoders*, mas infelizmente acontecem.
- O acesso ao feedback do robô é lento e complicado como já descrito anteriormente;
- A resposta aos comandos não é imediata fazendo com que a comunicação com o robô possa ser lenta em vários momentos;

Outra carência do sistema experimental está na implementação feita toda em *Matlab*. Apesar deste ser um programa bastante adequado para experimentos e prototipagem, trata-se de uma linguagem de programação interpretada, e portanto é pouco eficiente em termos computacionais.

Novas etapas para o presente projeto que podem ser desenvolvidas em trabalhos futuros são:

- Uma segunda etapa no projeto envolvendo o refinamento do modelo interno criado podendo trabalhar com técnicas de pré-processamento de imagem e *feature extraction* sobre as imagens obtidas durante a exploração;
- Uma terceira etapa abordaria a navegação do robô após o reconhecimento do ambiente ter sido feito. Isto seria feito apresentando ao robô imagens de objetos ou lugares para o qual este deverá navegar. Neste momento, podem ser aplicadas técnicas de

Redes Neurais e *Feature Extraction* entre outras já citadas para poder se fazer o reconhecimento das imagens apresentadas em seu banco de imagens panorâmicas já criado. Após a identificação do nó que possui a imagem apresentada, o robô deve se auto-localizar e percorrer os caminhos já aprendidos até o nó desejado;

- Adaptar o algoritmo proposto para ambientes não estáticos;
- Explorar características métricas do ambiente durante a exploração;
- Adequar o presente projeto para ambientes não planos;
- Estudar a interação da dinâmica do robô dentro do algoritmo de exploração e navegação proposto;

9
Referências Bibliográficas

1 SAUERBRONN, L. E. A. **Robótica Aplicada a Reconhecimento de Ambientes.** Dissertação (Mestrado), Pontifícia Universidade Católica do Rio de Janeiro, 1998.

2 DUDEK, G.; JENKIN, M.. **Computational Principles of Mobile Robots.** Cambridge University Press, 2000.

3 KONOLIGE, K. **Improved occupancy grids for map building.** Autonomous Robots 4(4), p. 351–367, 1997.

4 LATOMBE, J.C. **Robot Motion Planning**, Kluwer Academic Publishers, 1991.

5 MATTHIES, L.; ELFES, A. **Integration of sonar and stereo range data using a grid-based representation.** Proceedings of the IEEE International Conference of Robotics and Automation, p. 727–733, 1988.

6 PAGAC, D.; NEBOT, E. M.; DURRANT-WHYTE, H. F. **An evidential approach to map-building for autonomous vehicles**. IEEE Transactions on Robotics and Automation 14(4), p. 623–629, 1998.

7 CROWLEY, J. L. **World modeling and position estimation for a mobile robot using ultrasonic ranging.** Proceedings of the IEEE International Conference Robotics and Automation, p. 674–680, 1989.

8 DURRANT-WHYTE, H. F.; LEONARD, J. J. **Direct Sonar Sensing for Mobile Robot Navigation.** Kluwer Academic, Dordrecht. (IROS'99), v. 3, p. 1687–1692, 1992.

9 FEDER, H. J. S.; LEONARD, J. J.; SMITH, C. M. **Adaptive mobile robot navigation and mapping.** International Journal of Robotics Research 18(7), p.650–668, 1999.

10 HÉBERT, P.; BETGÉ-BREZETZ, S.; CHATILA, R.; **Decoupling odometry and exteroceptive perception in building a global world map of a mobile robot: the use of local maps.** Proceedings of the IEEE International Conference of Robotics and Automation, p. 757–764, 1996.

11 MOUTARLIER, P.; CHATILA, R. G. **Stochastic multisensory data fusion for mobile robot location and environment modeling.** Proceedings of the International Symposium on Robotics Research, Tokyo, Japan, p. 85–94, 1989.

12 HÉCTOR, H.; GONZÁLEZ-BAÑOS; LATOMBE, J.C; **Navigation Strategies for Exploring Indoor Environments**. The International Journal of Robotics Research, V. 21, No. 10–11, p. 829-848, 2002.

13 MURPHY R. R. **Introduction to AI Robotics.** The Mit Press, 2000.

14 VICTORINO, A.C.; RIVES, P.; BORRELY, J.J. **Safe Navigation for Indoor Mobile Robots. Part II**: Exploration, Self-Localization and Map Building. The International Journal of Robotics Research, v. 22, No. 12, p. 1019-1039, 2003.

15 PRESTES, E.; IDIART, M. A. P.; TREVISAN, M.; ENGEL, P.M. **Autonomous Learning Architecture for Environmental Mapping**, Journal of Intelligent and Robotic Systems 39, p. 243–263, 2004;

16 YAUMACHI, B.; ADAMS, W.; SHAFER, G. **Integrating Exploration, Localization, Navigation and Planning with a Common Representation.** Autonomous Robots, v. 6, No. 3, p.293-308, 1999.

17 REZA, F.M. **An Introduction to Information Theory.** Dover, New York, 1994.

18 RUANAIDH J. O.; PUN T. **Rotation, scale and translation invariant digital image watermarking.** Proceedings IEEE International Conference on Image Processing 1997 (ICIP 97), Santa Barbara, CA, USA, v. 1, p. 536--539, Outubro, 1997.

19 VIVEK A. S.; MICHAEL P. M. **Fingerprint identification using space invariant transforms.** Pattern Recognition Letters, v. 23 , No. 5, p. 609–619, ISSN:0167-8655, 2002.

20 WEISSTEIN E. W. **Discrete Fourier Transform**. MathWorld - A Wolfram Web Resource, http://mathworld.wolfram.com/DiscreteFourierTransform.html, arquivo consultado em 01 de abril de 2006.

21 WEISSTEIN. E. W. **Fourier Transform**. MathWorld - A Wolfram Web Resource, http://mathworld.wolfram.com/FourierTransform.html, arquivo consultado em 01 de abril de 2006.

22 JAIN, A. K. **Fundamentals of Digital Image Processing.** Prentice-Hall International Editions, 1989.

23 SAZBON D.; ZALEVSKY Z.; RIVLIN E.; MENDLOVIC D. **Using Fourier/Mellin-based correlators and their fractional versions in navigational tasks.** Pattern Recognition 35, p. 2993-2999, 2002.

24 PARIS, R. B. **Asymptotics and Mellin-Barnes Integrals.** Cambridge University Press, 2001.

25 LOWE, D. **Distinctive image features from scale-invariant keypoints.** International Journal of Computer Vision (submitted), 2003.

26 SE, S.; LOWE, D.; LITTLE, J. **Global localization using distinctive visual features**. Proc. Int. Conf on Intelligent Robots and Systems (IROS), p. 226--231, Lausanne, Switzerland, 2002.

27 LOWE, D. **Local feature view clustering for 3D object recognition**. Proc. CVPR, p. 682—688, Springer, 2001.

28 SE, S. , LOWE, D.; LITTLE, J. **Mobile robot localization and mapping with uncertainty using scale-invariant visual landmarks.** The International Journal of Robotics Research, v. 21, No. 8, p. 735-758, 2002.

29 LOWE, D. **Object Recognition from Local ScaleInvariant Features.** ICCV, Kerkyra, 1999.

30 MIKOLAJCZYK, K.; SCHMID, C. **Scale and affine invariant interest point detectors**, Int. J. Computer Vision, v. 60, No. 1, p. 62--86, 2004.

31 MIKOLAJCZYK, K.; SCHMID, C. **A performance evaluation of local descriptors**. Proc. IEEE Conf. Comp. Vision Patt. Recog., v. 2, p. 257--263, 2003.

32 HARRIS, C.; STEPHENS, M. "A combined corner and edge detector", Proc. Alvey Vision Conf., Univ. Manchester, p. 147-151, 1988.

33 LING, H.; JACOBS, D.W. **Deformation Invariant Image Matching**. ICCV, p. 1466-1473, 2005.

34 GOTZE, N.; DRUE, S.; HARTMANN, G. **Invariant Object Recognition with Disciminant Features based on Local Fast-Fourier Mellin Transform**. Proceedings of 15th International Conference on Pattern Recognition, v. 1, Barcelona. Los Alamitos (IEEE Computer Society), p. 948 – 951, 2000.

35 BROWN, M.; LOWE, D. G. **Invariant features from interest point groups.** British Machine Vision Conference, BMVC, Cardiff , p. 656-665, 2002.

36 GLASBEY, C.A.; MARDIA, K.V. **A Review of Image Warping Methods.** Journal of Applied Statistics, 25, p. 155-171, 1998;

37 BEAUCHEMIN, S.S.; BARRON, J.L. **The Computation of Optical Flow.** ACM Computing Surveys (ACMCS1995) 27(3), p. 433-467, 1996.

38 SAUERBRONN, L. E. A. **Um Método para o Cálculo de Fluxo Ótico com Estimativa de Confiabilidade**, Tese (Doutorado), Pontifícia Universidade Católica do Rio de Janeiro, 2002.

39 ZITOVA, B.; FLUSSER, J. **Image registration methods:** a survey. Image and Vision Computing 21, p. 977–1000, 2003.

40 TRAKA, M.; TZIRITAS, G. **Panoramic View Construction.** Signal Processing: Image Communication 18, p. 465-481, 2003.

41 HOUGH, P.V.C. **Method and means of recognizing complex patterns.** U.S. Patent 306965418, 1962.

42 FISCHLER, M. A. ; BOLLES, R. C. **Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography**. Comm. of the ACM, v. 24, p. 381-395, 1981.

43 LACEY, A. J.; PINITKARN, N.; THACKER, N.A. **An Evaluation of the Performance of RANSAC Algorithms for Stereo Camera Calibrarion.** British Machine Vision Conference, BMVC, 2000.

44 LI, L. **On-line Visual Tracking with Feature-based Adaptive Models.** Master of Science Thesis, The University of British Columbia, 2004.

45 UCHIYAMA, H. et al. **A Semi-Autonomous Wheelchair with HelpStar.** Proceedings of the 18th International Conference on Industrial & Engineering Applications of Artificial Intelligence and Expert Systems, IEA/AIE'05, p. 809-818, Bari, Italy, 2005.

46 CISTERNINO, A. et al. **Robotics4.NET:** Software body for controlling robots. IEE Proceedings Software, v. 152, No. 5, p. 215-222, 2005.

47 ONO, Y.; UCHIYAMA, H.; POTTER, W. **A Mobile Robot For Corridor Navigation:** A Multi-Agent Approach. Proceedings of the 42nd Annual ACM Southeast Conference, p. 379-384, Huntsville, Alabama, 2004.

48 SUJAN, V.A. et al. **A Multi Agent Distributed Sensing Architecture with Application to Planetary Cliff Exploration.** Proceedings of the International Symposium on Robotics Research (ISRR), Siena, Italy, 2003.

49 PACHECO, M. A. C. **Algoritmos Genéticos**: Princípios e Aplicações. INTERCON99: V Congreso Internacional de Ingeniería Electrónica, 1999, Lima, Peru. Proceedings of the INTERCON99: V Congreso Internacional de Ingeniería Electrónica. Lima, Peru, 1999. p. 11-16.

50 PACHECO, M.A.C. **Notas de Aula em Computação Evolucionária.** http://www.ica.ele.puc-rio.br, Arquivo consultado em 01 de abril de 2006.

51 HART, P.E.; NILSSON, N.J.; RAPHAEL, B. **A Formal Basis for the Heuristic Determination of Minimal Cost Paths**. IEEE Transactions on Systems, Science and Cybernetics SSC-4(2), p. 100-107, 1968.

52 BRÄUNL, T. **Embedded Robotics**, Springer-Verlag, 2003.

53 MITCHELL, M. **An Introduction to Genetic Algorithms**. Mit Press, 1999.

54 LOADER, C. **Local Regression and Likelihood.** Springer-Verlag New York, Incorporated, 1999.

55 JONES, J. J.; FLYNN, A.M.. **Mobile Robots, Inspiration to Implementation**. A K Peters, 1998.

56 MARTÍNEZ, R.G.; BORRAJO, D. **An Integrated Approach of Learning, Planning, and Execution.** Journal of Intelligent and Robotic Systems 29, p. 47–78, 2000.

57 DODDS, Z. et al. **Teaching robot localization with the evolution ER1. In Accessible Hands-on Artificial Intelligence and Robotics Education.** AAAI Spring Symposium. Tech. Rep. SS-04-01, p. 18-23, 2004.

58 http://www.evolution.com. Arquivo consultado em 01 de abril de 2006.