6 (seis)

FLÁVIO FERNANDES NACCACHE
RICARDO KENJI HACHIYA

VISÃO ESTÉREO A PARTIR DE SEQÜÊNCIAS DE IMAGENS GERADAS POR UMA CÂMERA PANORÂMICA CENTRAL COM ESPELHO HIPERBÓLICO

Trabalho de formatura apresentado à Escola Politécnica da Universidade de São Paulo para obtenção do título de graduado

São Paulo
2001

FLÁVIO FERNANDES NACCACHE
RICARDO KENJI HACHIYA

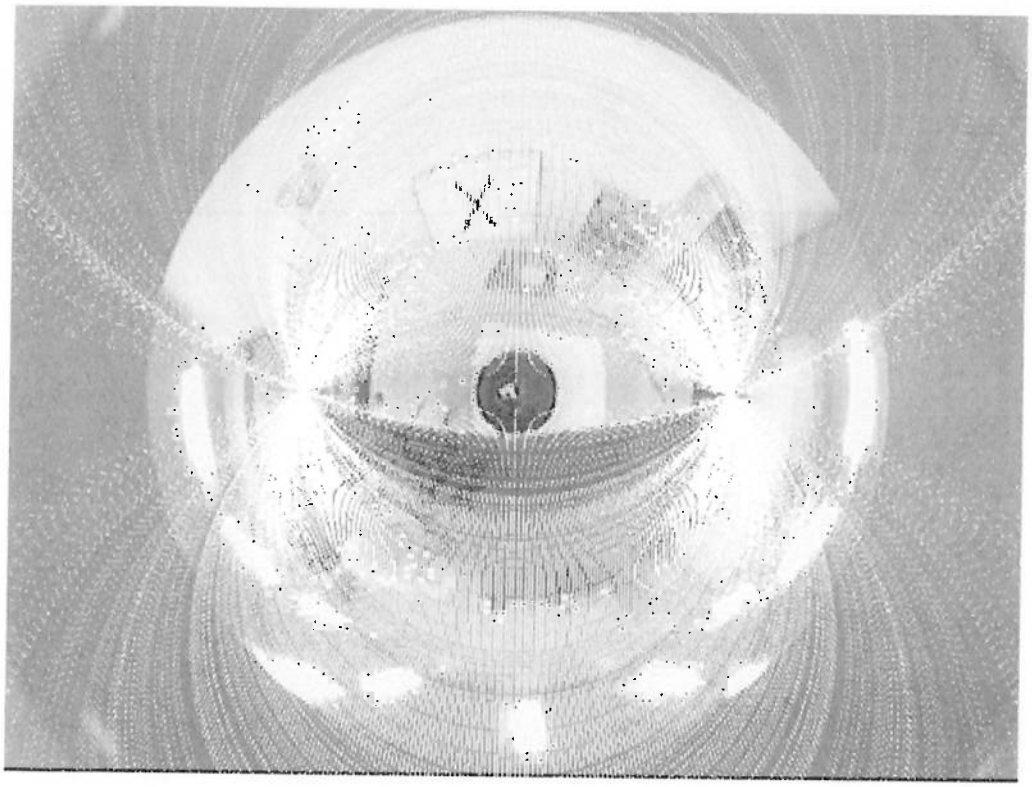
VISÃO ESTÉREO A PARTIR DE SEQÜÊNCIAS DE IMAGENS GERADAS POR UMA CÂMERA PANORÂMICA CENTRAL COM ESPELHO HIPERBÓLICO

Trabalho de formatura apresentado à Escola Politécnica da Universidade de São Paulo para obtenção do título de graduado

Área de concentração:
Departamento de Engenharia Mecatrônica e Sistema Mecânicos

Orientador:
Prof. Jun Okamoto Jr.

São Paulo
2001



Linha epipolares em uma imagem omnidirecional

SUMÁRIO

RESUMO

Este trabalho tem como objetivo a geração de mapas de distâncias a partir de seqüências de imagens utilizando-se um sistema de visão computacional com uma câmera panorâmica central e espelho hiperbólico para gerar imagens omnidirecionais, com um campo de visão de 360°. O trabalho deve ser utilizado para auxiliar a navegação de um robô autônomo com utilização de apenas um sensor (câmera panorâmica) como fonte de informações do meio. As seqüências de imagens 2D com campo de visão de 360° são adquiridas a partir de um robô do Grupo de Percepção Avançada (GPA) do Laboratório de Automação e Sistemas (LAS) do Departamento de Engenharia Mecatrônica e de Sistemas Mecânicos da EPUSP. O trabalho foi dividido em duas partes. Na primeira parte do trabalho foi implementado um programa em linguagem C para traçar Linhas Epipolares nas imagens omnidirecionais. Na segunda parte, rotinas para obtenção de pontos de correspondência entre duas imagens seqüenciais e a geração dos mapas de distância foram implementadas no programa. Os principais objetos de estudo foram: Sistemas de visão, visão omnidirecional, linhas epipolares em imagens planas, linhas epipolares em imagens omnidirecionais, cálculo de distâncias com imagens perspectivas e panorâmicas, métodos de correlação, métodos de extração de características, e métodos de tratamento de imagens. O traçado de linhas epipolares apresentou bons resultados, obtendo linhas epipolares que cruzavam a imagem em pontos correspondentes aos pontos selecionados com precisão de pixels, tanto nas imagens planas como nas cônicas. Os métodos de correspondência mostraram-se eficientes. Os melhores resultados foram obtidos utilizando-se o método de maximização da correlação exata entre janelas sobre a epipolar. A maior parte dos pontos correlacionáveis obtiveram correspondência com precisão de pixels. O resultado final foi à geração de mapas de distâncias com as informações obtidas.

1 INTRODUÇÃO

A navegação autônoma de um robô móvel envolve a sua movimentação de um determinado ponto a outro sem a intervenção de um operador externo. É necessário que o robô possa desviar de obstáculos e identificar a sua posição absoluta ou relativa dentro do ambiente. Para que isto seja possível, é necessário que o robô tenha instrumentos de percepção do ambiente como sensores, câmeras, instrumentos infravermelhos etc. e que tenha uma malha fechada de fluxo de informações.

A complexidade da navegação irá depender de uma série de fatores: o tamanho do robô, sua cinemática, o seu poder de computação e os tipos de sensores envolvidos. O ambiente pode ser conhecido ou desconhecido, estático ou dinâmico, com terreno plano ou acidentado, interno ou externo. A navegação em si pode possuir algumas restrições tais como a escolha do menor caminho, ou a execução em um tempo limitado.

Uma das técnicas utilizadas para auxiliar a navegação autônoma é a visão computacional.

De uma forma geral, na visão computacional podem ser obtidos 3 tipos de informações contidas nas imagens: movimento, textura e profundidade. Neste trabalho buscou-se, por meio de métodos de visão estéreo, a obtenção da profundidade.

Estéreo (do grego stereos) quer dizer sólido. Visão estéreo (ou estereopsia) refere-se à capacidade de se perceber variações de profundidade da imagem do sólido observado.

Os primeiros trabalhos sobre a visão computacional são da década de 60 quando ROBERTS no MIT realizou estudos sobre objetos poliédricos. Porém somente na década seguinte que surgiram os primeiros trabalhos de visão estéreo, com MARR e POGGIO (1976).

A partir de apenas uma única imagem não é possível obter informações completas sobre profundidade (AYACHE, 1991), mesmo em cenas com um contexto com uma rica variedade de informações

monoculares (que aumentam a compreensão da cena, como técnicas de pintura – sombreamento, sobreposição, perspectiva , gradiente de textura...), uma vez que estas são informações de cunho subjetivo (FRISBY, 1979).

A forma de se obter as informações da profundidade é a análise de mais de uma imagem tomada de diferentes posições relativas (por meio do movimento da câmera, do objeto, de ambos, ou da utilização de várias câmeras), e um pós-processamento das informações, sendo assim possível obter a informação da perspectiva. Desta forma, seqüências de imagens captadas por uma câmera podem fornecer, entre outras informações, as distâncias do robô aos objetos ao seu redor, dados estes importantes para a tomada de decisões.

Dentre as informações obtidas por visão é possível destacar 3: análise de movimento, análise de textura e percepção de profundidade. A visão estéreo é um método de obtenção desta informação.

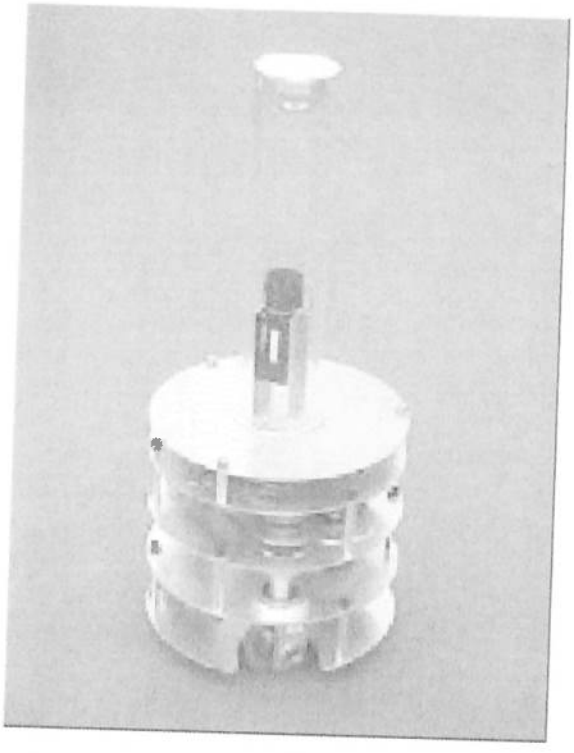


Figura 1 - Robô com sistema de visão panorâmica

O presente trabalho tem como objetivo a obtenção por visão estéreo profundidade para a navegação de um robô. A figura acima mostra o robô e o sistema de visão utilizado.

O robô faz parte de um estudo sobre navegação autônoma do Grupo de Percepção Avançada (GPA) do Laboratório de Automação e Sistemas (LAS) do Departamento de Engenharia Mecatrônica e de Sistemas Mecânicos da EPUSP. O sistema de visão implementado tem uma câmera panorâmica central com espelho hiperbólico acoplado a ele.

A partir deste sistema, foram adquiridos seqüências de imagens 2D com campo de visão de 360°, para a construção de mapas de distâncias do ambiente por meio de um programa em linguagem C em plataforma Linux.

1.1 ESTRUTURAÇÃO DO TRABALHO

O trabalho a seguir é dividido em 4 partes. A primeira é a parte teórica em que se encontram os principais conceitos abordados, assim como os trabalhos já desenvolvidos em cada área abordada pelo assunto. A seguir é apresentada a proposta implementada neste trabalho seguindo a estruturação realizada na parte teórica. O próximo tópico é análise dos resultados, onde estes são apresentados e discutidos, citando-se vantagens e desvantagens de cada método. Por fim é feita uma análise final do trabalho, comentando-se sobre os resultados obtidos e o caminho a ser seguido para uma futura continuação deste trabalho.

Em anexo são encontradas descrições sobre o software desenvolvido com instruções de uso, lista e resumo das funções implementadas, além de alguns códigos fontes comentados.

2 TEORIA

2.1 INTRODUÇÃO

A forma de se obter as informações completas da profundidade é a análise de mais de uma imagem tomada de diferentes posições relativas (através do movimento da câmera, do objeto, de ambos, ou da utilização de várias câmeras).

Este é o princípio básico de obtenção de tal característica da imagem, porém o processo a ser realizado para a obtenção final de um mapa de distâncias possui diversas fases. Analisando os estudos já realizados sobre o assunto chega-se a conclusão que as principais etapas para a obtenção dos mapas de distâncias em um sistema como este apresentado são:

- Modelagem do sistema de visão
- Pré-processamento das imagens
- Seleção de características das imagens
- Cálculo das linhas epipolares
- Correspondência de pontos (Matching)
- Cálculo de distâncias
- Processamento de informações e Geração do mapa de distâncias

A modelagem do sistema de visão fornece equações e parâmetros a serem usados nos cálculos posteriormente.

O pré-processamento é o tratamento da imagem para se eliminando ruídos, corrigir distorções, melhorar características das imagens como intensidade, brilho e contraste. O pré-processamento é um passo que na maior parte das vezes é feito em conjunto com o passo seguinte.

Seleção de características é o processo no qual procura-se nas imagens pontos notáveis, a serem utilizados nos mapas de distâncias. As características mais utilizadas são bordas de objetos e quinas de objetos.

O cálculo das linhas epipolares precede a etapa de procura por pontos correspondentes. Seu cálculo permite uma busca mais rápida e precisa.

A correspondência de pontos deve ser feita para cada ponto característico. Pontos correspondentes fornecem as informações para a etapa seguinte, cálculo de distâncias.

A geração de mapas de distâncias é feita a partir dos vários pontos correspondentes encontrados na etapa anterior. Um processamento das informações é necessário para se eliminar ruídos e correspondências falsas.

Nos próximos tópicos serão discutidas as etapas do processo e os conceitos envolvidos.

2.2 MODELAGEM DO SISTEMA DE VISÃO

2.2.1 INTRODUÇÃO

Existem diversas técnicas de visão omnidirecional para obtenção de imagens com campo de visão de 360°. Uma das técnicas utilizadas em visão é uso de várias câmeras voltadas para diferentes direções e um posterior processamento para a composição das imagens. É um método eficiente, com grande resolução, porém com um custo elevado, de difícil calibração e não compacto devido ao uso de múltiplas câmeras.

Outra forma de se conseguir uma grande resolução é a rotação de uma câmera em torno de um eixo vertical e pós-processamento das imagens para a sua composição. Esta forma de visão omnidirecional é de difícil uso dinamicamente, uma vez que para a composição final da imagem de 360° é necessária a obtenção de vários quadros em diferentes posições, que depende da velocidade de rotação da câmera. Esta velocidade não pode ser elevada devido ao tempo de focalização das imagens.

Utilizada com freqüência, a câmera com lente do tipo olho de peixe permite a obtenção em tempo real de uma imagem com campo de visão de 360°. Entretanto, este tipo de lente é cara e sua resolução é boa no centro da imagem, enquanto que nas periferias apresenta distorções consideráveis.

Uma forma efetiva de se obter uma imagem omnidirecional em tempo real, com uma boa resolução, utilizando um sistema compacto é a utilização dos chamados sistemas catadióptricos, nos quais uma câmera é direcionada para e espelhos convexos que refletem todo o ambiente.

Neste tipo de sistema, podem ser utilizados basicamente dois tipos de câmeras: de projeção perspectiva ou ortográfica. Quanto ao tipo de espelho, podem ser utilizados várias formas, bem como composições destes. As imagens geradas por este tipo de sistema dependerão destas variações.

2.2.2 SISTEMA CATADIÓPTRICOS

Diversos trabalhos já foram realizados utilizando este tipo de sistema para obtenção de imagens omnidirecionais. BAKER e NAYAR (1998) por exemplo descrevem um conjunto de sistemas catadióptricos com centro único de projeção. Desta forma é possível a geração de qualquer imagem projetada em qualquer plano definido pelo usuário. Já NAYAR (1997) e PERI (1997) desenvolveram um sistema de visão com uma câmera de projeção ortográfica e um espelho parabólico. YAGI (1994) desenvolveu um sistema de visão chamado COPIS para navegação de um robô utilizando uma câmera panorâmica e espelho cônico, obtendo boa resolução na periferia. YAMAZAWA (1993), YAGI (1993) e YACHIDA (1993,98) desenvolveram um sistema de sensoriamento utilizando espelho hiperbólico chamado HyperOminiVision, obtendo imagens que podem ser transformadas facilmente em projeções panorâmicas ou perspectiva como em (BAKER e NAYAR - 1998).

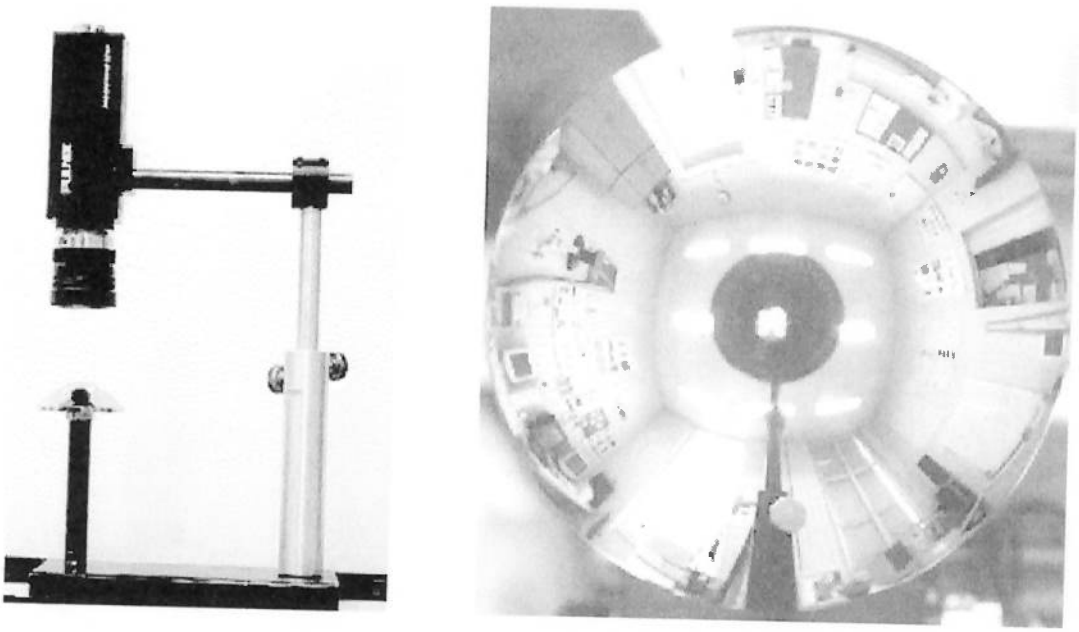


Figura 2 - Câmera panorâmica com espelho fixo, e imagem gerada por esta câmera

Sistemas com câmeras panorâmicas combinam espelhos convexos parabólicos ou hiperbólicos com uma câmera de projeção perspectiva para obter um campo de visão maior.

Na Figura 2 vemos o exemplo de uma câmera panorâmica com espelho fixo e a imagem gerada por este conjunto. A seguir serão discutidos os tipos de espelho utilizados e as principais características obtidas com cada um.

2.2.3 SISTEMAS COM ESPELHOS CONVEXOS

Os principais tipos de espelhos que são utilizados nos sistemas catadióptricos, como visto anteriormente, são os espelhos esféricos, hiperbólicos e parabólicos (cônicos).

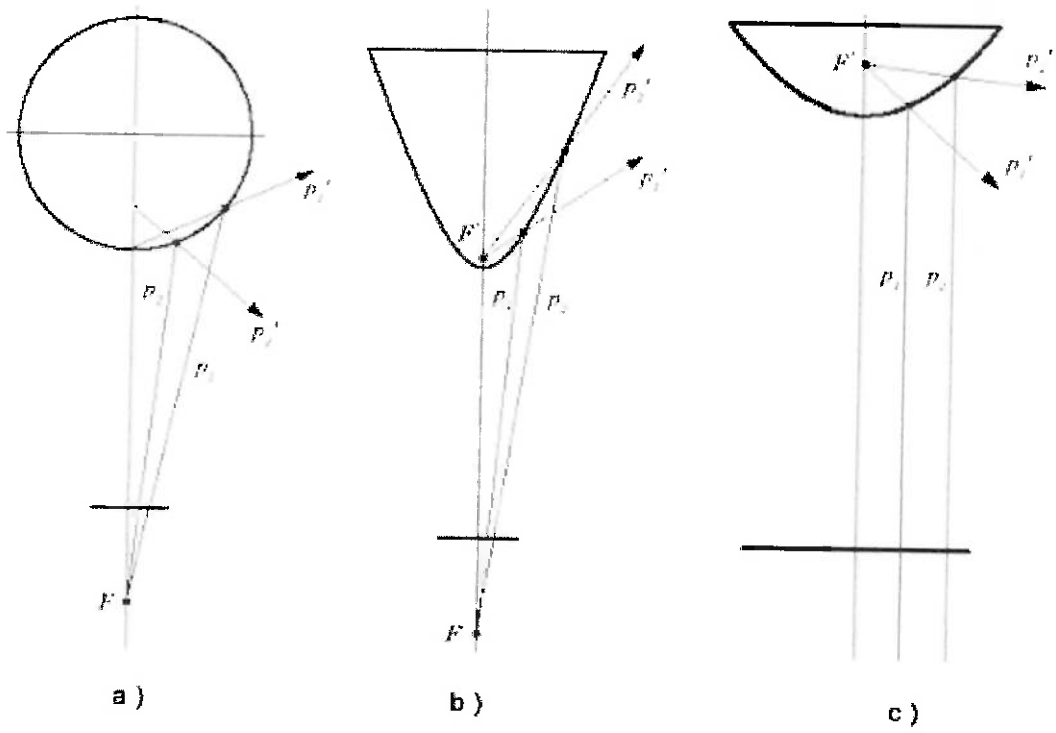


Figura 3 - Tipos de espelhos num sistema catadióptrico: a) espelho esférico; b) espelho hiperbólico; c) espelho parabólico

Em um sistema panorâmico, a câmera de visão é direcionada para um espelho de forma convexa. As imagens captadas pela câmera correspondem a 360° ao redor do eixo z do sistema.

Um raio genérico p_1 vindo (ou indo) da câmera é refletido no espelho num raio p_1' (Figura 3). Este raio p_1' pode ou não passar pelo foco do espelho, dependendo da geometria deste, porém todos os raios p devem passar pelo foco da câmera.

No caso a), utilizando-se um espelho esférico, os raios p' não se interceptam e um único ponto, o que é a chamada aberração esférica ou astigmatismo, que provoca muito desfoque na imagem gerada. Esta é uma desvantagem de se utilizar um espelho esférico, apesar de sua fácil confecção.

Em espelhos hiperbólicos, caso b), os raios p' que se interceptam no foco do espelho F', são refletidos em raios p, que se interceptam em F, dito centro único de projeção.

Ao se utilizar uma câmera de projeção perspectiva com um espelho hiperbólico, posiciona-se a câmera de forma que o foco da câmera coincida com o centro único de projeção. A distância entre os focos é de 2e (e $=\sqrt{a^2+b^2}$, com *a* e *b* parâmetros da hipérbole). Sistemas com espelhos hiperbólicos permitem obtenção de imagens panorâmicas com pouca distorção.

O sistema c) é formado com um espelho parabólico. Neste tipo de espelho, todos os raios p são refletidos em raios p' que se interceptam num único ponto F', Para que isto aconteça, o outro centro focal F deve estar no infinito, sendo necessário o uso de projeção ortográfica.

2.2.4 CÂMERA PANORÂMICA COM ESPELHO HIPERBÓLICO

A seguinte convenção de nomenclatura é utilizada (Figura 4):

q = $[q_u, q_v, 1]^T$, Ponto escrito no sistema de coordenadas da imagem e unidades em pixels.

$\mathbf{u} = [u, v, 1]^T$, Ponto escrito no sistema coordenado fixado no plano normalizado (plano com cota z igual a um).

$\mathbf{X} = [x, y, z]^T$, Ponto real no espaço escrito no sistema de coordenadas F' no centro do espelho. Unidades métricas.

$\mathbf{X_h} = [x, y, z]^T$, Ponto na superfície do espelho escrito no sistema de coordenadas F' no centro do espelho.

Um espelho hiperbólico pode ser definido, no sistema coordenado do espelho centrado no foco do espelho F´ por:

$$y = \sqrt{a^2 . \left(1 + \frac{x^2}{b^2}\right)} - \sqrt{a^2 + b^2}$$

1.

$$\alpha = \frac{\pi}{2} + a\tan\left(\frac{h - 2\sqrt{a^2 + b^2}}{r_{topo}}\right)$$

α corresponde ao ângulo máximo de visão (Figura 4).

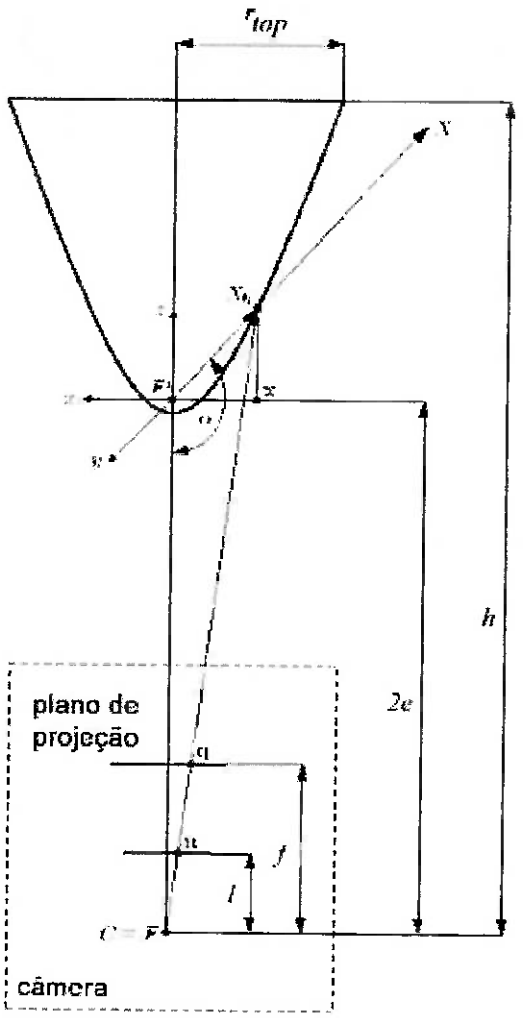


Figura 4 Geometria do espelho convexo com a câmera

Para a modelagem primeiro será feita a relação entre um sistema normalizado e o sistema em pixel. Com isso será definido um ponto genérico no espaço, e passado este ponto primeiramente ao sistema de referência do espelho, e achado o ponto em que este incide no espelho. A partir deste ponto será feita sua projeção no plano normalizado e então passado para o sistema coordenado da imagem, em pixel.

A *matriz de calibração interna da câmera K* relaciona um ponto genérico no sistema de coordenado em pixel $q=[q_u,q_v,1]^T$ a um ponto genérico $u=[u,v,1]^T$ no sistema normalizado:

$$2. \qquad u = \begin{bmatrix} \alpha_u & -\alpha_u \cot\theta & q_{u_0} \\ 0 & \frac{\alpha_v}{\operatorname{sen}\theta} & q_{v_0} \\ 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} q_u \\ q_v \\ 1 \end{bmatrix} = K^{-1} q$$

Sendo α_u e α_v os fatores de escala horizontal (u) e vertical (v), θ o ângulo de inclinação e q_{u0} e qv_0 as posições, em coordenadas em pixel, do ponto central.

Um ponto real Xm = $[x_m, y_m, z_m]^T$ projeta-se na superfície do espelho por:

$$3. \qquad X_h = \Im^{-}(X_m) X_m$$

$$X_h = [x, y, z]^T$$

Sendo $\Im$ a equação não linear que relaciona um ponto à superfície do espelho, obtida a partir da equação do espelho (1) e dada por:

$$4. \qquad \Im^{\pm}(X_m) = \frac{b^2\left(\pm e z_m + a \| X_m \|\right)}{b^2 {z_m}^2 - a^2 {x_m}^2 - a^2 {y_m}^2}$$

Com '*a*', '*b*' e '*e*' parâmetros do espelho.

O ponto obtido é então projetado no sistema coordenado da câmera. Se a distância do plano de projeção e o sistema de referência do espelho é dado por um deslocamento de $t_c = [0, 0, -2e]^T$ e um deslocamento angular de R_c, obtém-se a relação entre um ponto no espelho X_h e um ponto na imagem q :

$$5. \qquad X_h = \Im^{+}\left(R_C^T K^{-1} \vec{q}\right) R_C^T K^{-1} \vec{q} + t_c$$

Com $\Im^{+}\left(R_C^T K^{-1} \vec{q}\right)$ dado pela equação (4.).

2.3 PRÉ-PROCESSAMENTO

O pré-processamento é o tratamento da imagem para se eliminar ruídos, corrigir distorções e melhorar características das imagens como intensidade, brilho e contraste. Pode também ser considerada pré-processamento a aquisição de características presentes na imagem, utilizadas para os casamentos, porém estas serão tratadas como uma outra fase do processo, sendo discutidas posteriormente.

Dependendo de cada aplicação, um tipo de pré-processamento poderá ser feito. Com essa definição, fica muito ampla a gama de aplicação do pré-processamento, sendo este entendido como uma fase de preparação da imagem. Neste conceito pode se entender que uma retificação da imagem, de forma que suas linhas epipolares sejam paralelas é considerado pré-processamento. Porém alguns tipos de tratamentos são comuns nas imagens, entre eles podem ser citados:

- Brigthness – Com este filtro é possível equalizar o brilho e a intensidade das imagens;
- Contrast – Este filtro acentua as diferenças de intensidades entre as figuras;
- Treshold – Com este filtro a imagem pode ser binarizada, ou seja, define-se um nível e se este valor for superior, o valor é definido para um nível, se não é definido para o outro;
- Sharpness – com este filtro as bordas são acentuadas, aumentando-se a nitidez da imagem;
- Smothness – é o inverso do filtro anterior, as bordas perdem a força, porém as imagens tornam-se mais suaves
- Redimensionar – As imagens são aumentadas ou diminuídas e dependendo da resolução da imagem inicial é possível obter ganhos nas precisões ou até mesmo perdas.

Os filtros normalmente estão associados a parâmetros, podendo ser aplicados com maior ou menor intensidade, dependendo de cada caso e aplicação.

2.4 EXTRAÇÃO DE CARACTERÍSTICAS DAS IMAGENS

Características são entendidas aqui como qualquer tipo de propriedade da imagem com as quais se consiga obter informações sobre a cena analisada. De uma forma mais específica, podem ainda ser consideras como propriedades existentes em cada imagem de um estereograma (par de imagens para a obtenção do estéreo) que se consigam inter-relacionar.

A escolha do tipo de característica dependerá da cena em que esta está inserida e da aplicação a ser utilizada. Desta forma neste ponto já se pode inferir que dependendo do tipo de característica escolhida o mapa final de distância será mais ou menos denso.

As características mais comuns encontradas nos trabalhos de estereopsia são: áreas ou regiões, bordas, textura e cruzamentos de zero. As regiões possuem propriedades que as tornam mais fáceis de correlacionar que pontos escolhidos ao acaso na imagem, porém são características muito influenciadas pelo tamanho. Se a área for pequena, a influência do ruído é alta, se a área for grande, pode gerar imprecisão no cálculo de distância devido a grande variação de disparidade. Além disto áreas em uma cena podem não ter correspondência precisa em uma outra imagem devido a distorções. Por exemplo pode-se citar o caso de uma janela retangular em uma imagem, e sua correspondente na outra imagem sendo uma janela distorcida devido à posição não frontal da câmera.

Ao se trabalhar com linhas epipolares, esta se transforma num registro de variações de intensidade luminosas. È razoável considerar que picos e vales em cada linha se refiram aos mesmos pontos nos objetos observados. Para a obtenção de tais regiões, costuma-se utilizar operadores matemáticos como segunda derivada (obtém-se a variação da intensidade luminosa, assim como se esta clareando ou escurecendo), ou mesmo Laplaciano de Gaussianas. Porém este tipo de característica é muito sensível a ruídos e erros de discretização. (JENKIN et al., 1991).

Bordas são características que em muitos casos conseguem representar o próprio objeto. São também estudos de diversos trabalhos, incluindo alguns em visão estéreo. Entre os inúmeros algoritmos existentes, podem ser citados Canny, Sobel e LoG (Laplaciano de Gaussiana). O que normalmente é utilizado para a obtenção de bordas são os operadores matemáticos como gradientes, Laplacianos e Gaussianas, ou seja, busca-se normalmente máximos locais de intensidade luminosas. Os métodos baseados em gradientes apresentam linhas muito grossas, tornando os dados imprecisos para uma possível correlação. Já os métodos baseados em Laplacianos são muito susceptíveis a ruídos. Um outro problema é a robustez dos algoritmos, o que demanda muito dos processadores.

Já a textura é alvo de um menor número de trabalhos, devido a quantidade de filtros necessários para sua extração assim como a dificuldade de se relacionar uma estrutura de dados eficientes a esta abordagem.

Uma outra característica que pode ser citada é o ponto em si. A busca por correlação de pontos sem nenhum critério de escolha é praticamente inviável. É necessário portanto a obtenção de pontos que apresentem certas características de interesse. Um método interessante é a utilização de quinas. De qualquer forma a utilização deste método tem suas desvantagens por se tratar de um ente com poucas propriedades, o que pode sempre levar a ambigüidades.

De uma forma geral todas as características e métodos tem seus pontos falhos e suas vantagens, devendo-se considerar o que se encaixar melhor ao caso. Porém um algoritmo eficiente de extração de características, principalmente bordas e quinas, deverá seguir alguns critérios. Primeiro, deverá ter boa detecção, ou seja, ter o mínimo possível de falsos picos (bom comportamento a ruídos). Segundo, deverá ter boa acurácia, ou seja, as posições indicadas das características deverão ser o mais próximas possíveis de seu lugar real. Terceiro, o algoritmo deverá ter apenas uma resposta a uma única borda. E por fim, para aplicações em que o tempo é fundamental , o algoritmo deverá ser rápido.

2.4.1 SUSAN

Uma forma diferente de abordagem de processamento de imagens em baixo nível é apresentada em [17], onde se pode achar todo o desenvolvimento, equacionamento e análise do algoritmo. Trata-se de um nova forma de abordar os algoritmo de detecção de bordas e quinas (corners), diferentes dos existentes atualmente, chamado SUSAN, acrônimo para "Smallest Univalue Segment Assimilating Nucleus", que significa basicamente a minimização do USAN, que é unidade que da base ao algoritmo.. Este algoritmo segue todos os critérios citados acima, além de possuir um bom desempenho computacional, comparado com outros algoritmos.

Considere a Figura 5 onde existe um retângulo escuro dentro de outro claro. Uma máscara circular (com seu centro sendo chamado de núcleo) está representada em cinco posições diferentes. Se a intensidade luminosa de cada ponto na máscara é comparada à intensidade no núcleo, é possível definir uma área que possui intensidade igual (ou similar) ao do núcleo. Esta área é a definição de USAN (Univalue Segment Assimilating Value).

O conceito de que cada ponto na imagem possui associada a ele um valor de USAN , é o princípio do algoritmo SUSAN. Desta forma fica claro que este tipo de abordagem possui uma grande diferença dos algoritmos usuais, e de cara percebe-se que não se usam derivadas e portanto não é necessário redução de ruídos.

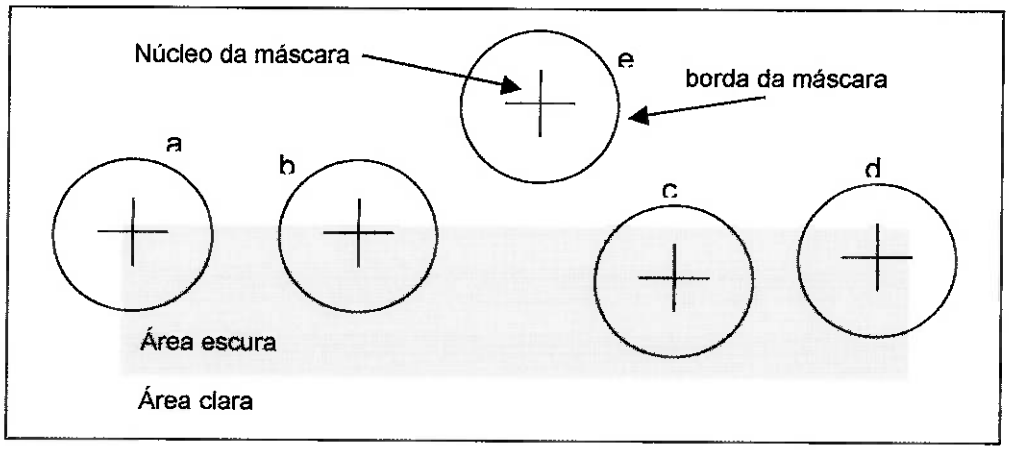


Figura 5 - cinco máscaras circulares em diferentes pontos da mesma imagem

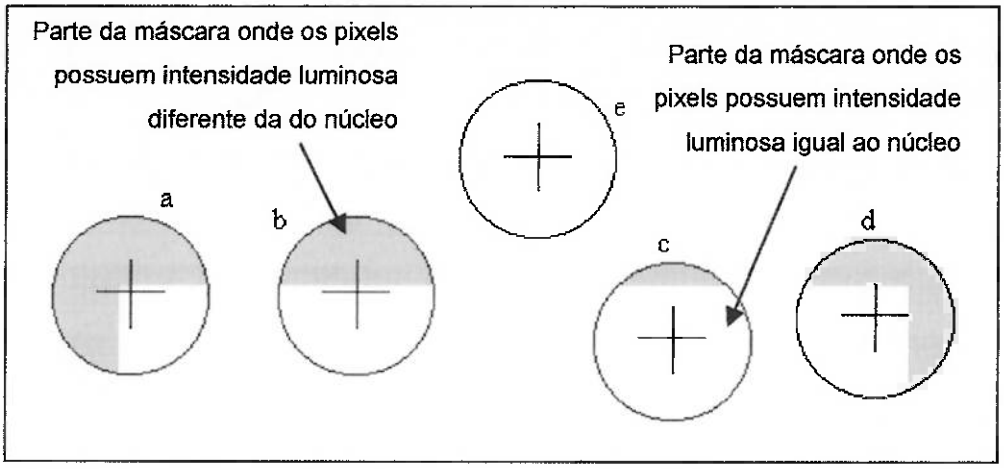


Figura 6 - Representação do USAN nas cinco máscaras circulares da imagem

Desta forma, o USAN permite uma boa estruturação da imagem. Da Figura 6 e da Figura 6 pode ser tirado ainda que:

- O USAN é máximo em regiões com mesma intensidade.
- O USAN é aproximadamente metade de seu valor máximo em regiões perto de bordas
- O USAN é mínimo em regiões perto de quina

Utilizando estas propriedades e repetindo-se o processo para todos os pontos da imagem, chega-se a ao resultado da figura a seguir.

Com esta plotagem dos valores USAN, fica claro o princípio de funcionamento para detecção de bordas e quinas do algortimo SUSAN.

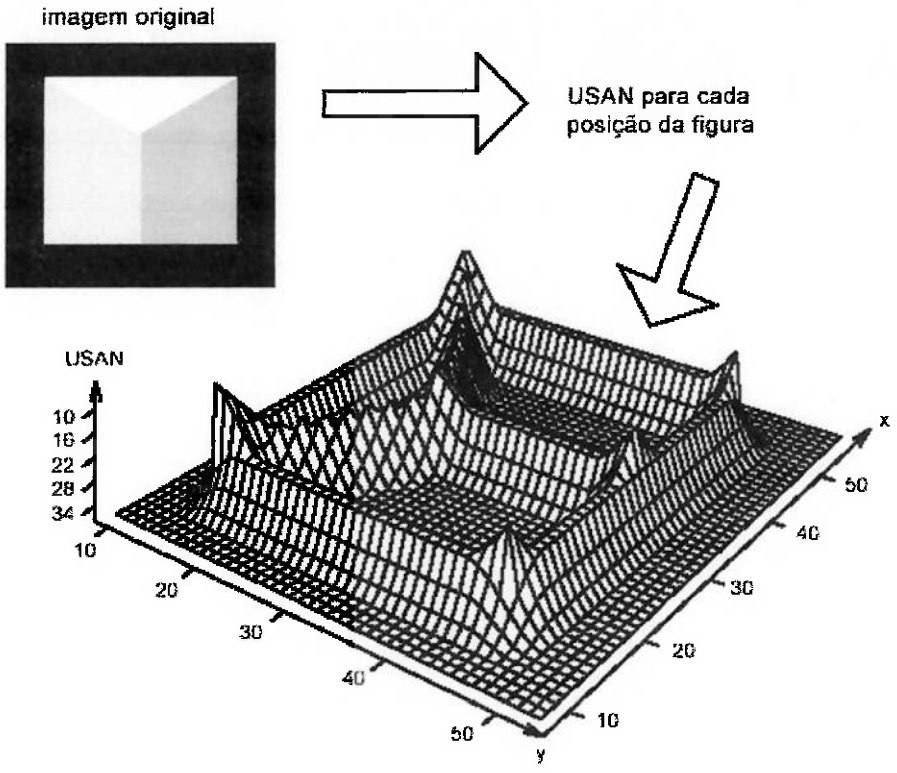


Figura 7 - Representação em 3 dimensões dos valores USAN de uma imagem

Quanto a detecção corners, o algoritmo não prevê problemas em uniões que não sejam cantos como junções em "T", "Y" , "X" ou outras, como outros detectores. Além disto o algoritmo não faz nenhuma consideração sobre a estrutura das regiões dos pontos (podendo achar sem problemas pontos em regiões inclinadas), muito menos considera a forma da junção, podendo trabalhar sem problemas com junções múltiplas. Outro ponto é que o algoritmo não procura por pontos de interesse, trabalhando a imagem como um todo.

Resumindo o algoritmo procede da seguinte forma para cada pixel da imagem. Primeiramente é criada uma máscara circular em torno do pixel em questão (núcleo). Em segundo lugar é calculado o número de pixels na máscara que possui intensidade similar ao núcleo. Para tanto utiliza-se uma equação não linear para a redução da sensibilidade a ruídos, dada por:

$$6. \qquad c(\vec{r},\vec{r}_0) = e^{-\left(\frac{I(\vec{r})-I(\vec{r}_0)}{t}\right)^6}$$

Com c o valor USAN, I a intensidade luminosa e t o valor de threshold utilizado para definir qual a variação que se poderá ter se saber se a região faz parte ou não do USAN.

Prosseguindo o algoritmo é feita a subtração do "geometric threshold" (g) do valor final do USAN ou seja, é subtraído do valor USAN o valor que se deveria ter nos corners (por exemplo 3/4 do valor total, como no exemplo das figuras 4 e 5). Tal relação é dada por:

$$7. \qquad R(\vec{n}_0) = \begin{cases} g - n(\vec{r}_0) & se \quad g > n(\vec{r}_0) \\ 0 & caso \quad contrário \end{cases}$$

Onde R é o resultado final, g o geometric threshold e n o valor parcial do USAN obtido anteriormente.

Posteriormente é feito a verificar de validade do córner, utilizando os conceitos de contigüidade e centróide. O centróide é achado pela média ponderada da distância do pixel verificado e o núcleo. Caso este esteja muito perto do núcleo, o corner é descartado por se tratar de um falso corner. Já a contigüidade é verificada analisando os pixels que estão na direção do centro de gravidade passando pelo centro. Estes pixels deverão pertencer à região do USAN, o que lhe dá uniformidade.

Por fim é feita a análise do mapa gerado pelos valores USAN, buscando-se os máximos locais, onde são as posições dos corners.

2.5 GEOMETRIA EPIPOLAR

2.5.1 GEOMETRIA EPIPOLAR PARA CÂMERAS PERSPECTIVAS

A geometria epipolar é uma ferramenta utilizada em visão estéreo para auxiliar a busca por pontos correspondentes. Sua utilização reduz o espaço da busca de dois graus de liberdade para um grau de liberdade.

A figura abaixo ilustra a geometria epipolar em imagens feitas com câmeras perspectivas.

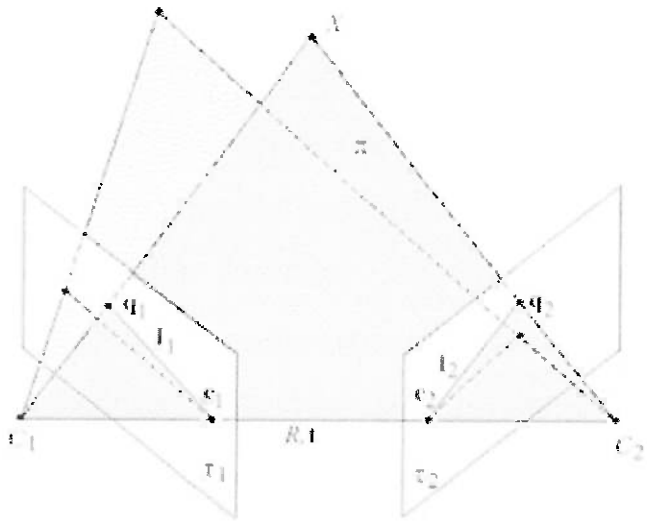


Figura 8 - Geometria epipolar de duas imagens perspectivas

Na visão estéreo, duas imagens focalizam um mesmo ponto a partir de lugares distintos. Ou seja, dois planos de projeção (imagens) para um mesmo ponto no espaço. Dado um ponto na primeira imagem (q1), deseja-se encontrar o seu correspondente na segunda imagem (q2).

O objetivo da geometria epipolar é fornecer o lugar geométrico, na segunda imagem, em que este ponto (q2) se encontra. Em imagens

perspectivas, para cada ponto q1 na primeira imagem, a geometria epipolar fornece uma linha epipolar na L2 na segunda imagem.

Os pontos C1 e C2 são os focos das imagens. Os pontos q1 e q2 são as projeções em cada plano do ponto real X. O plano epipolar (π_1) é o plano definido pelos centros de projeção das imagens (C1 e C2) e um ponto real (X). As linhas epipolares (L1 e L2) são definidas como a intersecção do plano epipolar (π_1) com os planos das imagens (t1 e t2). Os epipólos (e1 e e1) são dados pela intersecção da linha que une os centros de projeção com os planos das imagens. Todas linhas epipolares referentes a um par de imagens estéreo passam pelos epipólos, pois a localização dos pontos C1 e C2 é fixa.

Ao se projetar o ponto real X no plano da imagem (q1), perde-se a informação de sua profundidade. Dado o ponto q1, o ponto X pode estar em qualquer ponto da reta formada por C1 e q1. As informações que restam são as localizações dos pontos C1, C2 e q1, e dos planos t1 e t2. Com estes três pontos, pode-se encontrar o plano epipolar. A intersecção deste plano com o plano t2, fornece a linha epipolar L2.

Pela geometria, é possível notar que a linha epipolar é a projeção da linha formada por C1 e q1 ("candidatos" a serem o ponto real do ponto q1) no plano t2.

Com o centro de coordenadas localizado em C1, pode-se definir a linha epipolar vetorialmente como:

$$8. \qquad \vec{l}_1 = \vec{q}_1 \wedge \vec{e}_1 = B_{(e1)} \cdot \vec{q}_1$$

O vetor q1 e o vetor que liga os focos C1 e C2 definem o plano epipolar. Este plano intercepta os planos de projeção, formando as linhas epipolares (l_1 e l_2). Sabendo-se que elas são coplanares é possível definir a relação como:

$$9. \qquad \vec{l}_1 = A \cdot \vec{l}_2 \quad \textit{(A uma matriz 3x3)}$$

Associando-se estas duas equações, chega-se à relação entre os pontos da imagem:

$$10. \qquad \bar{q}_2^T . Q . \bar{q}_1 = 0$$

Sendo Q uma matriz de parâmetros do par de imagens estéreo (matriz fundamental).

2.5.2 GEOMETRIA EPIPOLAR PARA CÂMERAS PANORÂMICAS CENTRAIS COM ESPELHOS HIPERBÓLICOS

Aplicando-se a geometria epipolar em pares de imagens de câmeras panorâmicas centrais com espelhos hiperbólicos, pode-se chegar, assim como pares de imagens de câmeras perspectivas, a lugares geométricos (curvas epipolares) que limitam a busca por pontos correspondentes.

Em câmeras panorâmicas, as curvas são definidas como a projeção no plano da figura da intersecção do plano epipolar e o espelho hiperbólico. Estes lugares geométricos correspondem a diversas formas geométricas que são ilustradas nas figuras abaixo.

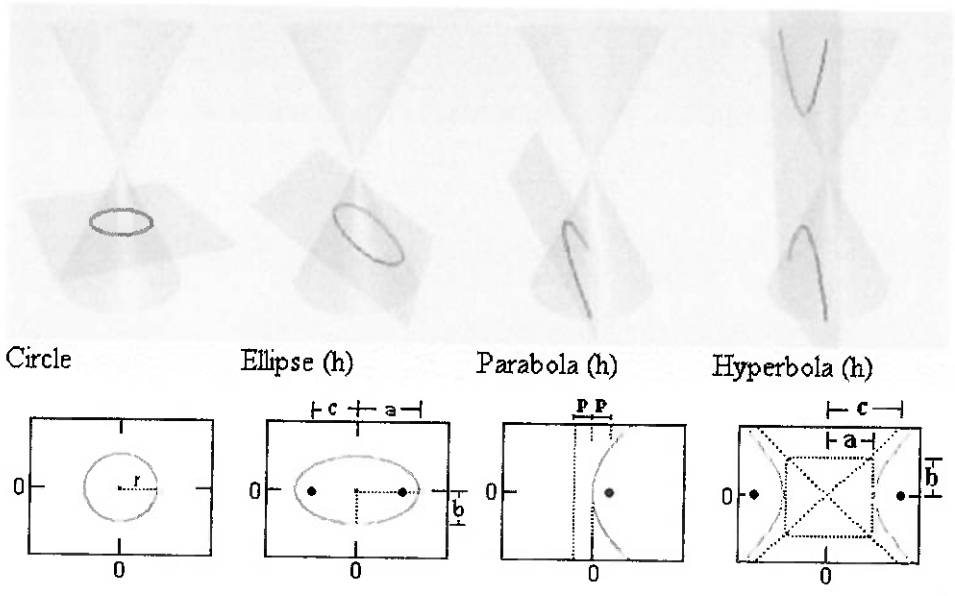


Figura 9 - Formas obtidas para as linhas epipolares

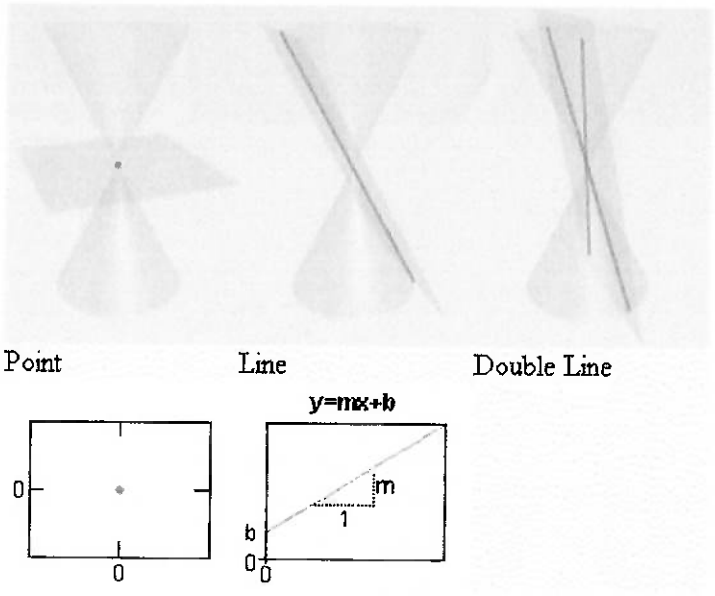


Figura 10 - Outras formas adquiridas (cônicas irregulares)

Mudando-se a inclinação do plano e o local da intersecção, as curvas geradas podem ser Cônicas regulares: círculos, elipses, parábolas ou hipérboles, ou cônicas irregulares: Ponto, linha ou duas linhas.

A fórmula matemática de uma figura cônica é dada por :

$$a.x^2 + b.x.y + c.y^2 + d.y.z + e.z^2 + f = 0 \quad (11.)$$

A sua forma matricial, pode ser expressa por:

$$p^T.M.p = 0 \quad (12.)$$

Sendo p um vetor dado por $p = [x,y,z]^T$ e M uma matriz que relaciona os parâmetros da cônica (a, b ,c, d, e).

A figura abaixo ilustra o sistema câmera-espelho hiperbólico em duas posições distintas e a geometria epipolar correspondente a um ponto no espaço.

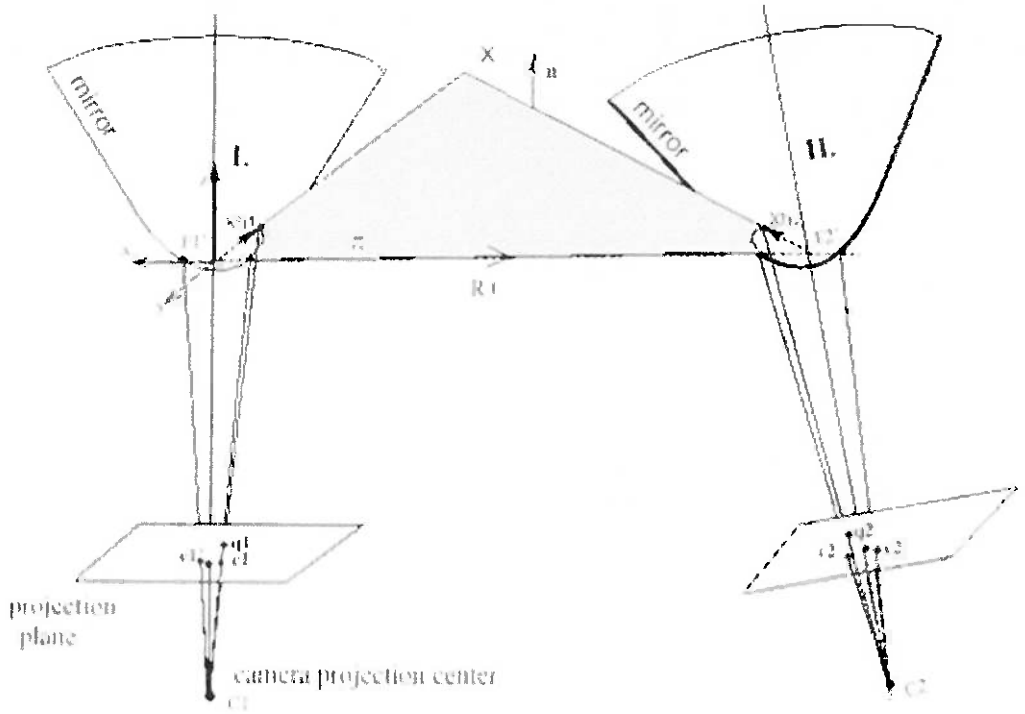


Figura 11 - Geometria epipolar com duas câmeras panorâmicas com espelhos hiperbólicos

São definidos dois sistemas de coordenadas: F1', centrado no foco do espelho 1 e F2' – centrado no foco do espelho 2. A transformação entre eles é dada pelo vetor t e pela matriz de rotação R.

Os pontos C1 e C2 correspondem aos centros de projeção das imagens.

Considerando-se um ponto genérico X no espaço, suas projeções no primeiro e no segundo espelho são dadas por Xh_1 e Xh_2 respectivamente.

Uma vez que Xh_1, Xh_2 e o vetor t são coplanares (veja figura acima), é possível definir a relação entre eles por uma rotação e uma translação:

$$Xh_2 . R.(\vec{t} \wedge Xh_1) = 0 \qquad (13.)$$

Ou, na forma matricial:

$$Xh_2^T . E . Xh_1 = 0 \qquad (14.)$$

Por meio da geometria epipolar pode-se, para um ponto genérico q1 no plano da primeira imagem, gerar a curva epipolar na segunda imagem. De forma genérica, a curva epipolar no sistema de coordenadas da segunda imagem pode ser obtida de:

$$15. \quad \vec{q}_2^{\,T} . A_2(E, \vec{q}_1) . \vec{q}_2 = 0$$

A geometria epipolar esta completamente definida pela A, que é função dos parâmetros do espelho, dos parâmetros da câmera e de sua movimentação.

No sistema de coordenadas centrada no Foco do espelho F'_2, a curva epipolar é dada por

$$16. \quad \vec{x}_2^{\,T} . A_{x2} . \vec{x}_2 = 0$$

O foco do espelho F`1, o vetor Xh_1 (liga o foco ao ponto no espelho) e o vetor de translação t (que define a distância entre os focos dos espelhos), definem o plano epipolar (Figura 11).

O plano epipolar, pode ser definido nas coordenadas do primeiro espelho, pela sua normal, que é dada pelo produto vetorial entre o vetor de translação e o vetor posição do ponto Xh_1 no espelho:

$$17. \quad \vec{n}_1 = \vec{t} \wedge Xh_1$$

A normal pode ser expressa em coordenadas do segundo espelho por:

$$18. \quad \vec{n}_1 = R . \vec{n}_2$$

Definindo o vetor $n_2 = [p,q,s]^T$, o plano epipolar pode ser expresso em coordenadas do segundo espelho por:

$$19. \quad p.x + q.y + s.z = 0$$

Escrevendo z em função de x, y na equação acima, substituindo na equação do espelho (1.), e aplicando um pouco de álgebra, é possível, obter uma equação polinomial em x e y:

20. $(p^2.b^2 - s^2.a^2)x^2 + 2.p.q.b^2.xy+(q^2.b^2 - s^2.a^2)y^2 - 2.s.p.b^2.e.x - 2.s.q.b^2.e.y - s^2.b^4 = 0$

Esta equação é uma equação de uma cônica e pode ser expressa da forma matricial por:

21. $$A_{x2} = \begin{bmatrix} p^2b^2 - s^2a^2 & pqb^2 & -pseb^2 \\ pqb^2. & q^2b^2 - s^2a^2 & -qseb^2 \\ -pseb^2 & -qseb^2 & s^2b^4 \end{bmatrix}$$

Os pontos pertencentes à curva epipolar são os pontos de intersecção do plano epipolar com o plano do espelho, desta forma, um ponto qualquer $x_2=[x,y,z]^T$ no sistema de referência do segundo espelho, pertencente à linha epipolar, é também pertencente ao plano definido pela equação (19.). Desta forma, um ponto do espelho Xh_2 pode ser expresso, em termos de x_2, da seguinte forma, fazendo z função de x e y:

22. $$Xh_2 = \begin{bmatrix} x \\ y \\ \dfrac{-px-qy}{s} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -\dfrac{p}{s} & -\dfrac{q}{s} & 0 \end{bmatrix} x_2$$

Para encontrar o ponto no plano da imagem (q) dado um ponto do espelho (Xh), sabendo-se que a distância de projeção é $t=[0,0,2e]^T$ e a rotação é R_c. pode-se aplicar a equação (5.):

$$X_h = \Im^+\left(R_C^T K^{-1}\bar{q}\right)R_C^T K^{-1}\bar{q} + \bar{t}_c$$

23. $$\Rightarrow Xh_2 + \bar{t}_c = Xh_2 + \begin{bmatrix} 0 \\ 0 \\ 2e \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -\dfrac{p}{s} & -\dfrac{q}{s} & 2e \end{bmatrix} x_2 = \Im^+\left(R_C^T K^{-1}\bar{q}\right)R_C^T K^{-1}\bar{q}$$

E desta forma pode-se obter x_2 em função de q_2 dado por:

24. $x_2 = NR_c^T K^{-1} q_2 \qquad na\,qual \quad N = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \frac{p}{2se} & \frac{q}{2se} & \frac{1}{2e} \end{bmatrix}$

Finalmente, substituindo x_2 em (16.) obtém-se a equação desejada, com a linha epipolar dada em coordenadas do plano da imagem:

25. $\bar{q}_2^T . K^{-T} R_c . N^T . A_{x2} N . R_c^T . K^{-1} . \bar{q}_2 = 0$

A(e,q_1) é dada por:

26. $K^{-T} R_c . N^T . A_{x2} N . R_c^T . K^{-1}.$

que são parâmetros da câmera, espelho e posição.

2.5.2.1 AVALIAÇÃO E PARAMETRIZAÇÃO DE CURVAS EPIPOLARES

Para se visualizar as curvas epipolares, é necessário a sua avaliação, transformação para sua posição essencial e parametrização.

Avaliação

O tipo de curva epipolar pode ser determinado pelo determinante da matriz A e pelo seu cofator ∂.

	Cônica Regular ($\Delta \neq 0$)	Cônica Singular ($\Delta = 0$)
$\partial > 0$	Elipse	Duas linhas imaginárias
$\partial < 0$	Hipérbole	Duas linhas transversais
$\partial = 0$	Parábola	Duas linhas paralelas

Transformação

O objetivo da transformação da curva é colocá-la em sua posição essencial, na qual o centro da cônica coincide com o centro do sistema de coordenadas e os eixos principais da cônica coincidem com os eixos coordenados.

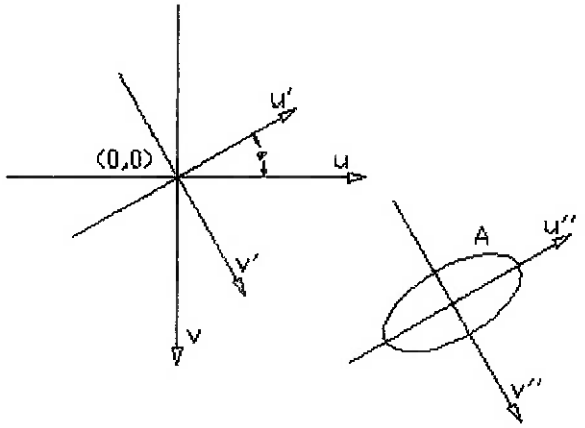


Figura 12 - Transformação de uma elipse de sua posição genérica para sua posição essencial

Para a rotação deve-se encontrar a matriz A' que satisfaça

$$u'^{T} A' u' = 0 \qquad 27.$$

Sendo φ o angulo de rotação, tem-se:

$$u' = \begin{bmatrix} \cos\varphi & -\operatorname{sen}\varphi & 0 \\ \operatorname{sen}\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix} u \rightarrow u = \begin{bmatrix} \cos\varphi & \operatorname{sen}\varphi & 0 \\ -\operatorname{sen}\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix} u' \qquad 28.$$

Substituindo 27. em 28., encontram-se as componentes da matriz A'

$$a_{11}' = a_{11}\cos^2(\varphi) + a_{22}\operatorname{sen}^2(\varphi) - 2a_{12}\cos(\varphi)\operatorname{sen}(\varphi)$$

$$a_{22}' = a_{11}\operatorname{sen}^2(\varphi) + a_{22}\cos^2(\varphi) + 2a_{12}\cos(\varphi)\operatorname{sen}(\varphi)$$

$$a_{31}' = a_{13}\cos(\varphi) - a_{23}\operatorname{sen}(\varphi) \qquad 29.$$

$$a_{32}' = a_{13}\operatorname{sen}(\varphi) + a_{23}\cos(\varphi)$$

$$a_{12}' = a_{11}\cos(\varphi)\operatorname{sen}(\varphi) - a_{22}\operatorname{sen}(\varphi)\cos(\varphi) + a_{12}(\cos^2(\varphi) - \operatorname{sen}^2(\varphi))$$

$$a_{33}' = a_{33}$$

Para a translação deve-se encontrar a matriz A'' que satisfaça

30. $$u''^T A'' u'' = 0$$

A translação de uma cônica é dada pela translação de seu centro.

31. $$u' = u'' + u_0'$$

Em XX, resulta:

$$a_{11}'' = a_{11}'$$

$$a_{22}'' = a_{22}'$$

32. $$a_{13}'' = 2a_{11}'u_0' + 2a_{13}'$$

$$a_{23}'' = 2a_{22}'v_0' + 2a_{23}'$$

$$a_{33}'' = a_{33}' + u_0'^2 a_{11}' + v_0'^2 a_{22}' + 2a_{13}'u_0' + a_{23}'v_0'$$

Para uma elipse ou hipérbole na posição essencial, deve-se ter $a_{13}'' = a_{23}'' = 0$. Em 32. resulta

33. $$u_0' = \frac{a_{13}'}{a_{11}'} \qquad v_0' = \frac{a_{23}'}{a_{22}'}$$

Para uma parábola, tem-se $a_{11}' = 0$ ou $a_{22}' = 0$. Supondo $a_{22}' = 0$, u_0' pode ser estimado pela 33. e v_0' é dado por

34. $$v_0' = \frac{-a_{11}u_0'^2 - 2a_{13}u_0' - a_{13}'}{2a_{23}}$$

<u>Parametrização:</u>

Elipse:

Uma das formas de parametrização da elipse é:

$$[u,v] = [a\cos(t), -b\,\mathrm{sen}(t)], \qquad 0 \le t \le 2\pi \tag{35.}$$

Comparando-se as equações

$$a_{11}''u''^2 + a_{22}''v''^2 = -a_{33}'' \quad \text{e} \quad \frac{u''^2}{a^2}'' + \frac{v''^2}{a^2} = 1 \tag{36.}$$

tem-se

$$a = \sqrt{-\frac{a_{33}''}{a_{11}''}} \quad \text{e} \quad a = \sqrt{-\frac{a_{33}''}{a_{22}''}} \tag{37.}$$

Demonstrações similares podem ser feitas para a hipérbole:

$$[u,v] = [t, \pm a\sqrt{1 + \frac{t^2}{b^2}}\,\mathrm{sen}(t)] \tag{38.}$$

e parábola:

$$[u,v] = [t, \frac{a_{11}''}{2a_{23}''}t^2] \tag{39.}$$

2.6 CORRESPONDÊNCIA DE PONTOS (MATCHING)

A correspondência de pontos em pares de imagens é uma etapa importante no processo de visão estéreo. Uma técnica mal implementada ou parâmetros mal equacionados podem resultar em erros e mapas de distâncias não consistentes .

Algumas características dos sistemas de visão e das imagens por eles geradas que dificultam o pareamento estéreo são:

- Regiões com pouco ou nenhuma variação de intensidade luminosa, o que impossibilita a extração de qualquer informação para casamento de intensidade luminosa.
- Regiões parcialmente ocultas. São regiões que aparecem em uma imagem, mas estão ocultas atrás de algum objeto na outra imagem.
- Regiões com textura uniforme e periódica. A dificuldade está no fato de não haver diferenças de um período a outro da textura, possibilitando casamentos com qualquer um dos períodos da textura.
- Efeitos de reflexão e refração sobre ou através dos objetos. Como as imagens são capturadas com um pequeno deslocamento, qualquer objeto que possua características de reflexão ou refração irá gerar intensidades luminosas distintas, para um mesmo ponto, em cada uma das imagens. Este problema só deixaria de existir se cada ponto de um objeto emitisse a mesma intensidade luminosa em todas as direções, o que não ocorre na realidade.
- Características intrínsecas do equipamento de captura de imagens utilizado. Isto pode causar diferenças de brilho, contraste ou mesmo diferenças não lineares entre as imagens capturadas. Outros fatores dependentes do equipamento são as distorções causadas pelas lentes de uma câmera

A correspondência ou matching (também citada como casamento) pode ser feita de diversas formas. Os métodos apresentados neste trabalho são essencialmente passivos, nos quais as informações provêm da intensidade luminosa dos pontos convertidos em valores numéricos e associada a pixels nas imagens.

Os métodos passivos podem ser agrupados em:

- métodos que procuram o casamento entre as características das imagens - correlação e correspondência.
- métodos indiretos - Predição e Verificação de hipóteses, Analise em freqüência e outros.

A seguir apresenta-se um resumo sobre os principais métodos e soluções encontrados na literatura.

2.6.1 CORRELAÇÃO

É um método baseado na correlação entre intensidades luminosas de certo trechos das imagens. Os algoritmos baseados em intensidade luminosa supõem que um ponto de um objeto possua uma intensidade luminosa igual em ambas imagens do par estéreo.

Correlação discreta é o processo no qual uma área ao redor de um ponto de interesse em uma imagem é "correlacionada" com áreas de formato similar em uma região objetivo na segunda imagem, e a área de "melhor casamento" na região objetivo é determinada. O centro da região de melhor casamento na segunda imagem é então considerada como o ponto correspondente ao ponto de interesse na primeira imagem.

Vários autores desenvolveram algoritmos baseados nesta técnica, diferindo muitas vezes no tamanho da janela. BARNEA, SILVERMAN em 1972 propuseram um método estatístico com uma janela retangular de m x n. Já KANADE e OKUTOMI em 1990, desenvolveram um algoritmo que calculava um tamanho especifico de janela para cada pixel. Outro método

baseado na correlação foi proposto por NISHIHARA em 84, que se baseava no sinal convolução da imagem ao se aplicar o Laplaciano do Gaussiano. Desta forma se obtém picos em regiões com continuidade de textura e superfícies do objeto, fazendo com que a análise seja deslocada para regiões ao invés de bordas.

MORAVEC (1977), GENERRY(1979), MATTHIES (1989), modificaram um pouco o método, adicionando algumas restrições, como a escolha de alguns pontos de interesse obtidos através de alguns operadores e validaram os casamentos através do teorema de Bayes. Outra variação relevante é a utilização da correlação de vetores ao invés de janelas, sendo que estes vetores possuem informações diversas como por exemplo filtragens espaciais sobre o ponto em questão (JONES, MALIK em 1991), informações geométricas sobre uma região (XU em 1989) ou mesmo padrões de textura (MAKIK, PERONA em 1989).

Esta abordagem falha em alguns casos, tais como: um ponto de um objeto possuir uma intensidade luminosa diferente em cada imagem do par estéreo; regiões parcialmente ocultas (regiões que aparecem em uma imagem, mas estão ocultas atrás de algum objeto na outra imagem).

O processo de correlação consiste dos seguintes passos:

1. Escolhe-se um ponto de interesse na primeira imagem.

2. A segunda imagem terá um conjunto de pontos candidatos a casamento relativos ao ponto de interesse escolhido. O objetivo é determinar o melhor ponto candidato ao casamento, e para isto utilizamos as janelas a seguir.

3. Determina-se a janela de interesse ao redor do ponto de interesse escolhido.

4. Para cada elemento do conjunto de pontos candidatos a casamento, determina-se suas janelas candidatas a casamento, cada janela

respectiva a seu ponto candidato a casamento. O formato da janela utilizada para janela de interesse e para a janela candidata a casamento deve ser o mesmo, para que se possa compará-las.

5. A comparação das janelas é feita por uma medição de casamento, que compara as intensidades luminosas da janela de interesse com as diversas janelas candidatas a casamento. A comparação é feita sobre todos pontos da janela de interesse com os respectivos pontos da janela candidatos a casamento analisados. As formas mais comuns para medição deste casamento são:

- correlação direta - o valor intensidade luminosa dos pontos respectivos em cada janela são multiplicados e somados.
- correlação por média normalizada - o valor intensidade luminosa dos pontos correspondentes em cada janela são multiplicados e somados.
- correlação por variância normalizada - semelhante ao correlação por média normalizada, mas a soma da correlação é dividida pelo produto das variâncias das intensidades em cada janela.
- soma das diferenças quadradas - a soma do quadrado das diferenças entre as intensidades dos pontos correspondentes.
- soma das diferenças absolutas - a soma dos valores absolutos das diferenças entre as intensidades dos pontos correspondentes

A medição de casamento pode ser ponderada durante a soma das comparações de cada ponto individual. A ponderação pode aumentar a contribuição para os pontos ao centro da janela e diminuir a dos pontos periféricos, tornando a comparação mais robusta.

6. Uma vez efetuadas as medições de casamento da janela de interesse com todas janelas candidatas a casamento, escolhe-se a janela candidata a casamento de melhor medição. A melhor medição depende da

forma adotada para medição. Se a forma adotada foi de correlação direta, média normalizada ou variância normalizada, a melhor medição é a de maior valor. Caso a forma adotada tenha soma das diferenças quadradas ou soma das diferenças absolutas, a melhor medição é a de menor valor.

7. Enfim, obtido o casamento da janela de interesse com uma janela candidata a casamento, casamos o ponto de interesse com o ponto candidato a casamento, cada qual respectivo à sua janela mencionada.

2.6.1.1 DESENVOLVIMENTO MATEMÁTICO

Dado um pixel de coordenadas (Uo, Vo) na primeira imagem, considera-se uma janela na segunda imagem de tamanho (2P + 1) X (2N + 1) inicialmente centrada nas mesmas coordenadas. Variando-se a posição da janela, calcula-se o fator de correlação C(t) para as várias posições.

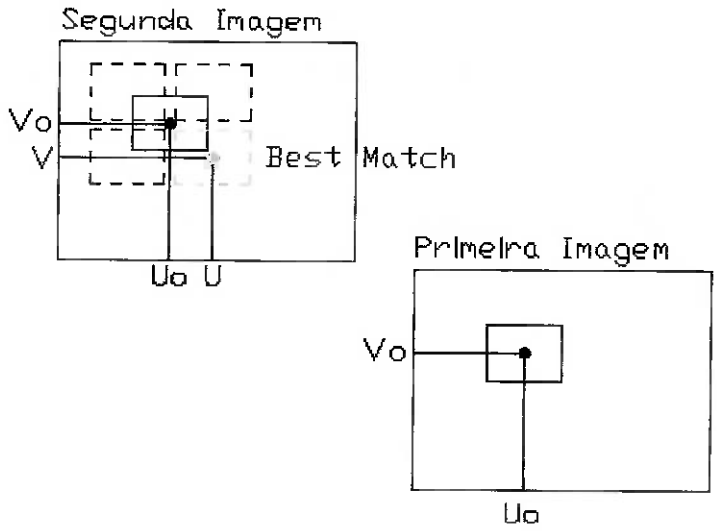


Figura 13 - Exemplificação do método de correspondência por janelas

Distribuindo-se os valores de C em um gráfico, o casamento deverá ser feito com o ponto central da janela cujo valo de C é o ponto de máximo no gráfico (ou mínimo, dependendo da função usada).

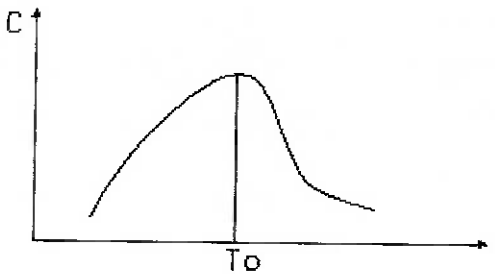


Figura 14 - Exemplo de distribuição das normal da intensidade luminosa

A função C de correlação para a janela variando na direção x é:

40.

$$C_{12}(\tau)=\frac{1}{k}\sum_{u_1=-N}^{+N}\sum_{v_1=-P}^{+P}(I_1(u_1+u_0,v_1+v_0)-\overline{I_1(u_0,v_0)}).(I_2(u_1+u_0+\tau,v_1+v_0)-\overline{I_2(u_0+\tau,v_0)})$$

na qual

41. $$K=(2N+1)(2P+1)\sigma_1(u_0,v_0)\sigma_2(u_0+\tau,v_0)$$

42. $$\overline{I_1(u_0,v_0)}=\frac{1}{(2N+1)(2P+1)}\sum_{u_1=-N}^{+N}\sum_{v_1=-P}^{+P}I_1(u_1+u_0,v_1+v_0)$$

43. $$\sigma_1{}^2(u_0,v_0)=\frac{1}{(2N+1)(2P+1)}\sum_{u_1=-N}^{+N}\sum_{v_1=-P}^{+P}(I_1(u_1+u_0,v_1+v_0)-\overline{I_1(u_0,v_0)})^2$$

Devido à normalização por σ_1 e σ_2, o valor de C_{12} será entre -1 e 1.

2.6.2 CORRESPONDÊNCIA

Ao se procurar correspondência entre as imagens, muitas vezes se cai no problema de uma mesma característica se repetir ao longo da imagem e portanto existirem inúmeras correspondências possíveis. Este problema é resolvido pelo conhecimento prévio da imagem ou, por exemplo,

restrições impostas pela geometria. Estas restrições diminuem a complexidade computacional, porém causam uma diminuição da densidade de pontos no mapa de profundidade.

Pode-se dividir estas restrições em 3 grandes grupos:

- geométricas impostas pelo sistema coordenado da imagem (curvas epipolares por exemplo).
- geométricos baseados nos objetos (gradiente de disparidade por exemplo).
- psicofísicos.

A seguir são citados alguns tipos de restrições.

2.6.2.1 CURVAS EPIPOLARES

Já mencionado anteriormente, é um dos principais recursos geométricos que podem ser utilizados. Transformam um problema de correspondência de 2D para 1D. Dada um pixel numa imagem, existe uma curva na imagem da direita na qual estará o pixel correspondente. Algoritmos utilizando este tipo de restrição são largamente citados na literatura, por exemplo HAN (2000) e SVOBODA (1997).

2.6.2.2 UNICIDADE

Cada ponto da imagem esquerda tem no máximo um único ponto correspondente na imagem direita. Pode ocorrer de não haver correspondência devido a oclusão de pontos.

2.6.2.3 CONTINUIDADE

A superfície dos objetos tem variações suaves em sua superfície, ou seja, pontos vizinhos estão normalmente a profundidades semelhantes.

2.6.2.4 ORDENAÇÃO

Os pontos em uma mesma linha epipolar tendem a preservar uma certa ordem de aparição. Porém isto nem sempre ocorre, dependendo da cena.

2.6.2.5 GRADIENTE DE DISPARIDADE

Origina-se de um estudo da continuidade das superfícies realizado por POLLARD em 1985. Este é definido como a razão entre a disparidade e a separação ciclopiana. Estes conceitos serão definidos posteriormente. O importante aqui é definir que o gradiente de disparidade é utilizado como um limite máximo de distância de busca para o casamento

2.6.3 RELAXAÇÃO/ COOPERAÇÃO

Estes tipos de processos são interativos e buscam um ponto de equilíbrio entre diversos parâmetros. São algoritmos que utilizam restrições, por exemplo, de forma. O ponto correspondente a um determinado pixel deve pertencer a um conjunto de pixels que não tenha sido excluído pelas restrições.

Os principais estudos realizados com estes tipos de algoritmos foram feitos em meados dos anos 70 ao começo dos 80. Os primeiros trabalhos foram feitos por MARR e POGGIO, tendo resultados muito bons para figuras chamadas ramdom dot stereograms. Muitos avanços realizados nesta área foram graças a GRIMSON (como o algoritmo que usava Laplaciano do Gaussiano – MPG). Porém para imagens reais esta técnica não obteve resultados muito satisfatórios. Um grande avanço foi dado com POLLARD, MAYHEW, FRISBY, que propuseram um algoritmo (PMF) para obtenção do casamento de pixels baseado principalmente no gradiente de disparidade. Mantendo as restrições de continuidade e ordenação, o sistema é executado até convergir para um resultado único. Os resultados são rápidos e muito bons, tanto com imagens sintéticas quanto imagens reais.

2.6.4 PREDIÇÃO E VERIFICAÇÃO DE HIPÓTESE

Este tipo de técnica é simples e eficiente para certos tipos de imagens. Faz-se algumas predições de possíveis casamentos, verifica-se estas predições baseando-se num processo puramente heurístico e, posteriormente, utiliza-se as restrições para a validação do resultado. AYACHE e FAVERJON introduziram este algoritmo em 1985 e foi seguido principalmente por MEDIONI e NEVATIA (1985). Esta técnica é o ponto de partida para algoritmos como o trinocular (AYACHE,LUSTMAN – 1991) e outros encontrados na literatura (YACHIDA (1990) e ROBERT (1991) entre outros)

Para exemplificar este tipo de algoritmo, é possível citar o trabalho de AYACHE e FAVERJON onde são utilizadas retas (menos susceptíveis a erros) definidas por suas características geométricas e intensidade luminosa. O algoritmo começa com a predição de algumas correspondências, feitas randomicamente e de forma que as características sejam mais ou menos parecidas (intensidade semelhante, ângulo e tamanho parecido, etc). Posteriormente ocorre a propagação, processo no qual para cada possível casamento, procura-se o vizinho (segmento mais próximo), de forma que este mantenha as restrições feitas na hora da predição com os segmentos vizinhos da outra imagem, e além disto, estejam a uma distância pré-fixada. O único casamento possível será aquele que "propagar" o maior número de vizinhos. O método é detalhado no livro "Artificial Vision for Mobile Robots" de AYACHE-1991. O problema deste método esta na baixa densidade do mapa de profundidades produzido e no fato de se considerar retas.

2.6.5 PROGRAMAÇÃO DINÂMICA

O processo de correspondência de primitivas de imagens pode ser encarado como um processo de minimização de custo. A programação dinâmica é um método eficiente de obtenção da combinação mínima de valores para um número alto de variáveis. Vários trabalhos foram realizados como por DAVID (1991); LLOYD (1987); XIAO-WEI, DUBUISSON (1990), e

entre eles BAKER, BINFORD (1981) e OHTA, KANADE (1985) que utilizam as bordas como as primitivas, porém os últimos obtiveram os melhores resultados criando um processo de busca de características nas linhas epipolares (restrição de ordenação) e adicionando mais um "custo", utilizando a restrição de continuidade entre as linhas epipolares mudando a busca para um plano 3D ("intra-scanline").

2.6.6 ANÁLISE EM FREQÜÊNCIA

Neste método não se busca explicitamente correspondência ou correlação, e sim uma análise utilizando técnicas espaço-freqüências (Gabor, Wavelet, DFFT), e assim medir a diferença de fase entre as imagens. Basicamente, o que se consegue é o mapa de distância a partir da diferença de fases entre componentes freqüências de janelas correspondentes (JENKIN-1988, SANGER-1988). Os resultados são bons, rápidos e pouco influenciados por ruídos, mas o método possui algumas limitações quanto às suas aplicações.

2.6.7 OUTROS PROCESSOS

Inúmeros outros processos são propostos, como o de minimização de custos em que se busca a correspondência ao minimizar uma função baseada em intensidade, por exemplo, e os pontos de escolha podem ser escolhidos por um operador de MORAVEC. Desta forma também podem ser criadas correspondências via rede neural (CHANG-1992), ou mesmo por algoritmos de busca em grafos (MYERS-2000).

2.7 CÁLCULO DE DISTÂNCIAS

2.7.1 CÁLCULO DE DISTÂNCIAS EM IMAGENS PERSPECTIVAS

A triangularização é o cálculo de distâncias em pares de imagens perspectivas. Dados os pontos correspondentes p_r e p_l , o vetor de translação T e a distância focal da câmera f, a distância Z é estimada pela disparidade entre os pontos correspondentes.

O x_l e x_r são as coordenadas dos pontos p_l e p_r em relação aos pontos principais c_l e c_r

Por semelhança de triângulos (Figura 11- p_l P p_r e O_l P O_r), temos:

44. $$\frac{T + x_l - x_r}{Z - f} = \frac{T}{Z}$$

Resolvendo para Z:

45. $$Z = f\frac{T}{d} \quad , \quad d = x_r - x_l$$

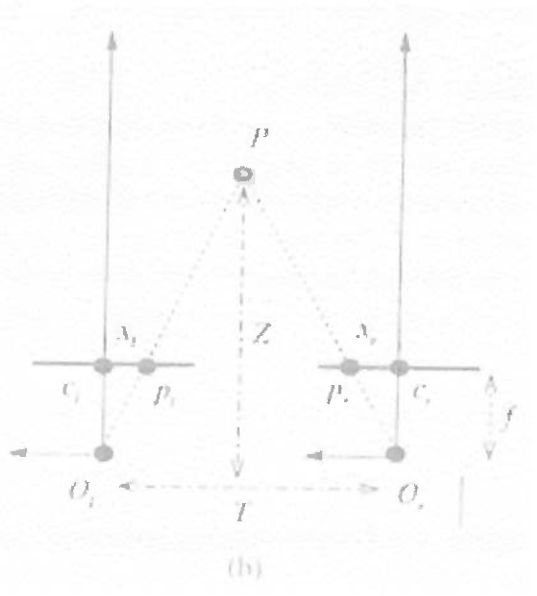


Figura 15 - Sistema Estéreo Simples

2.7.2 CÁLCULO DE DISTÂNCIAS EM IMAGENS OMNIDIRECIONAIS

A partir de duas imagens omnidirecionais, é possível se calcular distâncias para pontos reais sem retificá-las.

Dado um ponto P real, devemos determinar as distâncias D1 e D2, que se referem às distâncias do ponto P aos eixos Z dos sistemas F'1 e F'2 respectivamente. A figura abaixo mostra o problema a ser resolvido.

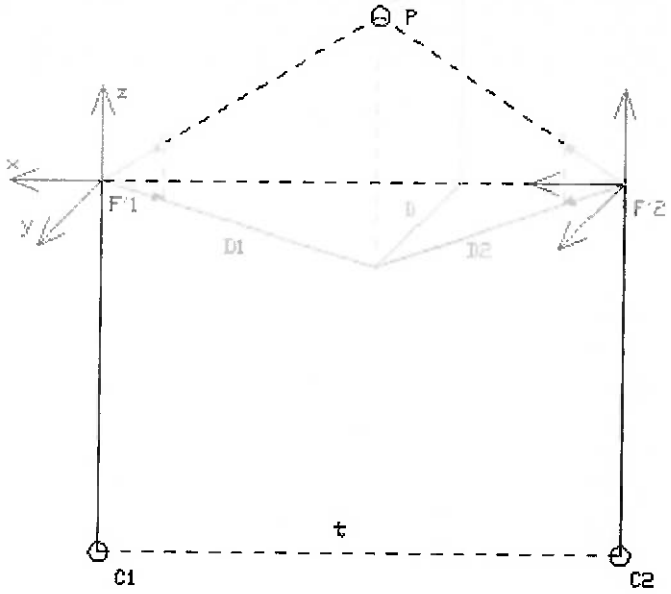


Figura 16 - Representação do problema a ser resolvido - reconstrução

Os sistemas 'F' estão centrados no foco do espelho hiperbólico. F'1 se refere à primeira imagem e F'2 se refere à segunda imagem.

Os vetores em verde (Xh) ligam o foco do espelho com a projeção do ponto P na superfície do espelho. As linhas em vermelho são a projeção dos vetores de posição do ponto P e dos vetores Xh no plano XY.

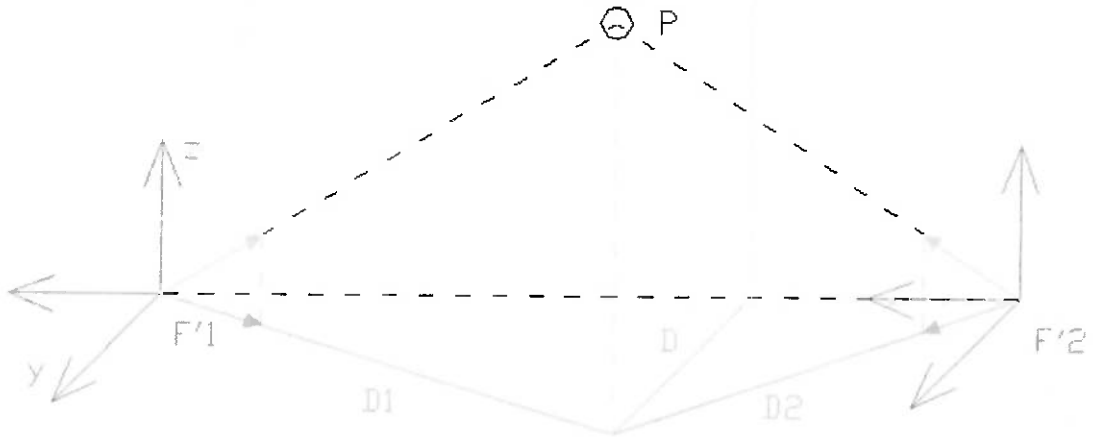


Figura 17 - Convenções de Notação: As barras indicam que a unidade é dadas em pixels. As letras sobrescritas indicam a que sistema de coordenadas o vetor se refere.

Dados dois pontos correspondentes nas imagens:

$$\bar{u}_1 = [u_1 \quad v_1 \quad 0]^t \quad \text{e} \quad \bar{u}_2 = [u_2 \quad v_2 \quad 0]^t$$

A primeira etapa é transformar estes pontos em pontos na superfície da lente da câmera.

46.
$$\begin{cases} x_1^c = \bar{u}_1.\psi + f_c \\ x_2^c = \bar{u}_2.\psi + f_c \end{cases} \Leftrightarrow \begin{cases} x_1^c = [x_1 \quad y_1 \quad f_c]^t \\ x_2^c = [x_2 \quad y_2 \quad f_c]^t \end{cases}$$

Os vetores Xh1 e Xh2 são encontrados pela fórmula:

47.
$$\begin{cases} Xh_1 = F^+(x_1^c).x_1^c + t_c \\ Xh_1 = F^+(x_1^c).x_1^c + t_c \end{cases} \Leftrightarrow \begin{cases} Xh_1^{F'_1} = [Xh_1 \quad Yh_1 \quad Zh_1]^t \\ Xh_2^{F'_2} = [Xh_2 \quad Yh_2 \quad Zh_2]^t \end{cases}$$

Projetando estes vetores no plano XY, temos:

48.
$$\begin{cases} Xh_1^{F'_1} = [Xh_1 \quad Yh_1 \quad 0]^t \\ Xh_2^{F'_2} = [Xh_2 \quad Yh_2 \quad 0]^t \end{cases}$$

Graficamente

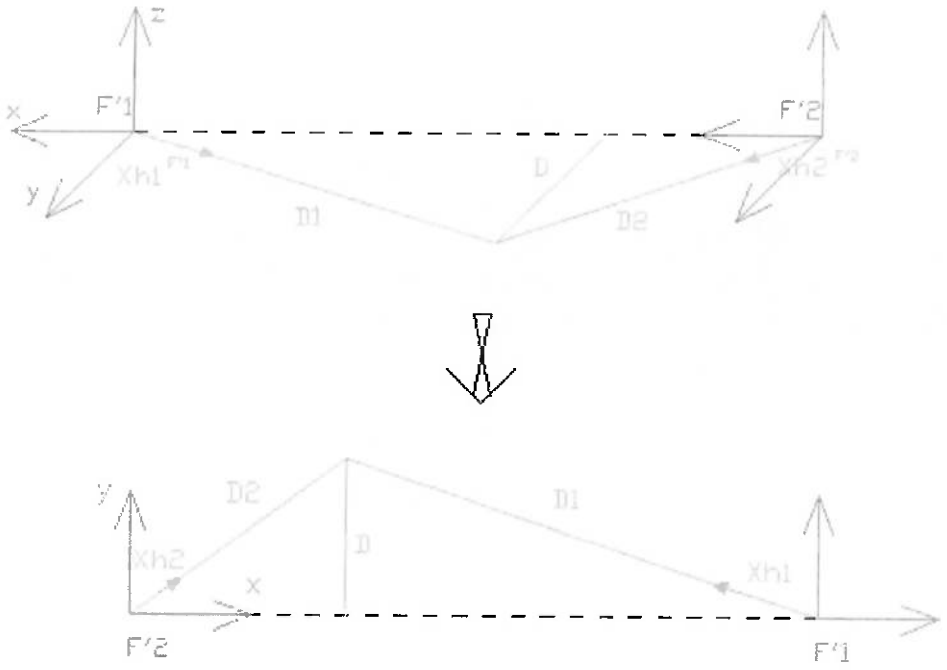


Figura 18 - Transformação dos vetores Xh para o plano XY

Decompondo os vetores:

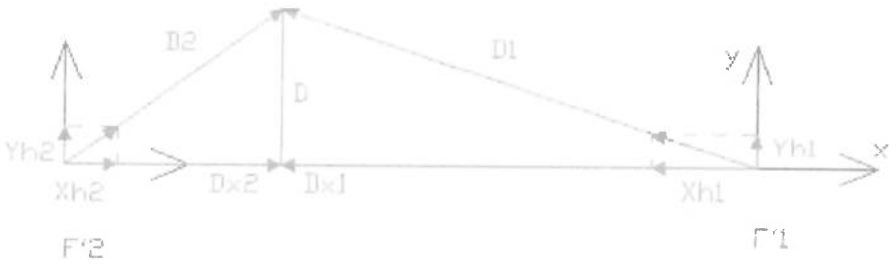


Figura 19 - Decomposição dos vetores Xh

Pela figura podemos chegar nas três equações básicas:

49. $$\frac{Yh_1}{Xh_1} = \frac{D}{Dx_1}$$

50. $$\frac{Yh_2}{Xh_2} = \frac{D}{Dx_2}$$

51. $$Dx_2 + (-Dx_1) = t$$

Manipulando-as:

$$Dx_1 = t - Dx_2 \quad ; \quad Dx_2 = \frac{Xh_1 \cdot D}{Yh_1} - t \quad ; \quad \left(\frac{Xh_1 \cdot D}{Yh_1} - t\right) \cdot Yh_2 = D \cdot Xh_2$$

Chega-se ao resultado:

52. $$D = t \cdot \frac{Yh_1 \cdot Yh_2}{Xh_1 Yh_2 - Xh_2 Yh_1}$$

As distancias para cada sistema pode ser calculada por:

$$D_1 = \sqrt{D^2 + Dx_1^{\,2}} \quad \text{e} \quad D_2 = \sqrt{D^2 + Dx_2^{\,2}}$$

53. $$D_1 = \sqrt{D^2 \cdot \left(1 + \frac{Xh_1^{\,2}}{Yh_1^{\,2}}\right)}$$

54. $$D_2 = \sqrt{D^2 \cdot \left(1 + \frac{Xh_2^{\,2}}{Yh_2^{\,2}}\right)} = \sqrt{\left(t \cdot \frac{Yh_1 \cdot Yh_2}{Xh_1 Yh_2 - Xh_2 Yh_1}\right)^2 \cdot \left(1 + \frac{Xh_2^{\,2}}{Yh_2^{\,2}}\right)}$$

A fórmula final é simples, de forma que este procedimento não representa custo computacional.

2.8 MAPAS DE DISTÂNCIA

Nas etapas de matching e cálculo de distâncias, é gerada uma lista de pontos correspondentes contendo dois tipos de informações para a geração de mapas de distância: a localização dos pares de pontos na imagem e os vetores de distância (D_1 e D_2) entre os pontos e os eixos z do sistema de visão.

Para a reconstrução do ambiente é necessário agora recriar o espaço, com as informações obtidas. A informação a ser usada a construção do mapa deve ser a distância D_2, pois se refere à última posição do robô.

Os pontos obtidos indicam a presença de um objeto ou não. Isso dependerá da quantidade de pontos achados em regiões espaciais similares a este. Isso se deve ao fato de os métodos de matching, por não serem exatos, poderem introduzir no processo informações erradas e incoerentes. Para a construção de mapas, é necessário um processamento destas informações no qual é feita a localização de prováveis objetos, ou grupos de pontos correlacionados corretamente. Após esta etapa, os pontos considerados corretos são agrupados de acordo com suas características de distância do robô e distância entre si. Pontos a uma mesma distância do robô e próximos uns dos outros representam provavelmente o mesmo objeto. Na etapa final da geração de mapas, cada grupo de pontos é interpretado de acordo com o tipo de mapa em questão. Em uma etapa adicional, os grupos podem ser ainda relacionados entre si.

3 APLICAÇÃO

Neste tópico, a teoria desenvolvida anteriormente será aplicada ao caso específico estudado neste trabalho. Para iniciar, será especificado o tipo de espelho e câmera utilizados. A seguir será introduzida a geometria epipolar ao caso, e depois os algoritmos de matching escolhidos. Por fim uma breve discussão sobre os métodos utilizados para a construção do mapa de distâncias.

3.1 CÂMERA E ESPELHO UTILIZADOS

O sistema de visão utilizado foi uma câmera panorâmica central com um espelho hiperbólico (desenvolvido em trabalhos anteriores). Os parâmetros do espelho utilizado foram escolhidos de forma que o conjunto ficasse compacto. O espelho foi feito através de um processo de usinagem de ultraprecisão no torno do Departamento de Engenharia Mecânica da USP de São Carlos. Todo o equacionamento, escolha do tipo e forma de usinagem pode ser encontrado em [10].

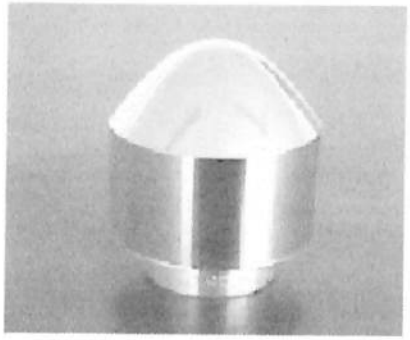


Figura 20 - Espelho hiperbólico usado no projeto

Os parâmetros do espelho são: b = 19,646, a = 39,292, e α = 121,3. A equação do espelho, em coordenadas centradas em F´ fica:

$$y = 39{,}3\sqrt{\left(1+\frac{x^2}{186}\right)} - 43{,}9 \quad (55.)$$

Os outros parâmetros mostrados na Figura 4 Geometria do espelho convexo com a câmera, que mostra o sistema do espelho, são: r_{topo} = 20mm e h = 100mm.

Quanto à câmera e lente que foram utilizadas no sistema de visão, os parâmetros utilizados são f = 12mm, e t_{pixel} = 0,01mm.

3.2 PRÉ-PROCESSAMENTO

Nesta aplicação não foi necessário nenhum tipo de pré-tratamento, isso considerando a extração de características uma fase separada. Devido ao uso de métodos de correlação e da dificuldade de se realizar a retificação da imagem, este processo também não foi realizado, eliminando assim vários algoritmos de correlação que tem como hipótese fundamental o uso da retificação.

3.3 SELEÇÃO DE CARACTERÍSTICAS

Pelas características da aplicação e do algoritmo SUSAN, este foi o escolhido para a extração das características. Apesar de se trabalhar com pontos e o mapa final ser mais sujeito a ruído e mais esparso, a escolha se justifica, além do quem o algoritmo roda bem rápido e é bem eficiente, obtendo bons pontos para a correspondência.

3.4 CURVAS EPIPOLARES

Neste trabalho, o deslocamento do sistema de visão se da paralelamente ao eixo x, obtendo o vetor de deslocamento

$$t = [x,0,0]^T$$

Com isto, podem ser eliminadas as rotações das formulas, uma vez que não haverá rotação das posições do espelho. A partir das equações finais da linha epipolar, chega-se a:

$$56. \qquad q_2^T . K^{-T} N^T . A_{x2} N . K^{-1} . q_2 = 0$$

Sendo o deslocamento unicamente no eixo x, o plano epipolar, definido por um produto vetorial, não terá parcelas em x, reduzindo a equação (21.) a:

$$57. \qquad A_{x2} = \begin{bmatrix} -s^2a^2 & 0 & 0 \\ 0. & q^2b^2 - s^2a^2 & -qseb^2 \\ 0 & -qseb^2 & s^2b^4 \end{bmatrix}$$

A substituição dos parâmetros é feita diretamente no programa.

O procedimento de cálculo para a obtenção das curvas epipolares utilizado no programa foi:

1. Dado um ponto na imagem (q) acha-se o ponto corresponde no espelho (Xh) pela equação **(5.)**.
2. Dado o deslocamento (t) e Xh, obtém-se os parâmetros p, q e s do plano epipolar pelo produto vetorial de t por Xh dado em **(17.)**.
3. Com estes parâmetros e os parâmetros do espelho (**a**,**b** e **e**), chega-se a A_{x2} e N, dados por **(21.)** e **(24.)** respectivamente.
4. Com os parâmetros da câmera, do conjunto (espelho+câmera) e os dados obtidos nos passos acima, obtém-se a equação da curva epipolar dada pela equação **(25.)**.
5. Obtida a equação da cônica em sua forma matricial, define-se seu tipo (hipérbole, parábola...)
6. Parametriza-se e desenha-se a curva.

3.5 MATCHING

Uma das características das imagens panorâmicas é que elas são distorcidas. Em uma primeira análise, pode parecer que estabelecer correspondências para pares de imagens deste tipo é muito complicado, a não ser que se use métodos especiais. No entanto, imagens panorâmicas fornecem muitas informações a respeito da cena (especialmente sobre a relação espacial entre os objetos) e permitem um tratamento global da cena, que normalmente fornece bons resultados [1].

A maior parte dos métodos de matching apresentados anteriormente foi desenvolvida para trabalhar com imagens planas, e de uma forma geral não são aplicados ao caso devido à necessidade de retificação como os algoritmos apresentados em [7,13,16,22].

Trabalhos anteriores sugerem que o melhor método para conseguir a correlação das imagens seria utilizar os métodos de correlação do tipo block matching [1]. Considerando ainda a necessidade de tempo real e um mapa de distâncias não extremamente denso, este método parece se encaixar bem ao caso.

Em [1] são mostradas pequenas modificações nos métodos que podem melhorar o desempenho em imagens panorâmicas, por exemplo o método de block matching com modificações no formato da janela de procura.

A primeira proposta é de se utilizar uma figura 1-D, na forma da linha epipolar, e a correlação é calculada deslocando-se esta figura pela linha. Num segundo passo, é proposto se utilizar uma janela retangular que seguiria a linha epipolar, com uma janela retangular. Derivando desta idéia pode-se ainda utilizar uma janela retangular distorcida, de tal maneira que sua forma média acompanhe a curvatura da linha epipolar.

Definido o método, deve-se ainda à escolha da melhor função de correlação para que satisfaça todas as características necessárias a essa aplicação. Como visto anteriormente, esta função correlaciona as

intensidades das duas imagens e pode ter diversas formas como em [7,13,18,22].

Para comparação, analise e melhora de resultados, foram implementadas diversas funções de correlação e métodos de busca. A seguir estão descritas as funções utilizadas.

3.5.1 MÉTODOS IMPLEMENTADOS

Quatro métodos foram implementados para a correspondência:

1. Correspondência entre quinas (corners).
2. Correspondência entre quinas nas curvas epipolares.
3. Correspondência por meio de janelas.
4. Correspondência por meio de janelas nas curvas epipolares.

1. Correspondência entre quinas (corners)

Para cada imagem, é gerada uma lista de quinas. Uma quina na primeira imagem deve ter uma quina correspondente na segunda imagem. Para cada quina da primeira imagem, percorre-se a lista de quinas da segunda imagem em busca de matching.

2. Correspondência entre quinas (corners) nas curvas epipolares

Para cada imagem, é gerada uma lista de quinas. Para cada quina da primeira imagem, percorre-se a lista da segunda imagem em busca de quinas próximas da curva epipolar.

3. Correspondência por meio de janelas

Para cada quina na primeira imagem, procura-se o ponto correspondente na segunda imagem computando-se o fator de correlação

entre janelas. A janela na primeira imagem permanece fixa, centrada na quina (u,v). A posição da janela na segunda imagem varia de u - Δ a u + Δ e de v - Δ a v + Δ.

4. Correspondência por meio de janelas nas curvas epipolares

Para cada quina na primeira imagem, procura-se o ponto correspondente na segunda imagem computando-se o fator de correlação entre janelas. A janela na primeira imagem permanece fixa, centrada na quina (u,v). A posição da janela na segunda imagem varia de forma que seu centro percorra a curva epipolar.

Para cada método de correspondência, as diversas funções de pré-restrição, restrição e de correlação implementadas foram utilizadas, podendo-se combinar várias funções para se ter a melhor correlação.

As funções de pré-restrição implementadas são:

1. Afastamento mínimo das Bordas - Quinas na primeira imagem a serem utilizadas no processo devem estar afastadas das Bordas da imagem. Nesta região são encontradas quinas que não correspondem a quinas reais.
2. Afastamento mínimo do eixo horizontal - As quinas na primeira imagem a serem utilizadas no processo devem estar afastadas do eixo horizontal. Nesta região se encontram os epipólos e o centro da figura, duas fontes de ruídos e falsas correspondências.

As funções de restrição implementadas são:

3. Distância máxima da curva epipolar - Os candidatos à correlação devem estar a uma distância máxima da curva epipolar.

4. Distância máxima da quina - Os candidatos à correlação devem estar a uma distância máxima da quina na primeira imagem.
5. Direção correta - Dado o vetor deslocamento do sistema de visão, sabe-se para que direção os pontos correspondentes às quinas devem esta.

As funções de correlação implementadas são:

Para janelas de tamanho P x Q, nas quais:

n = número de pixels na janela (P x N)

$I_{1,i}$ = intensidade do pixel ' i ' na primeira imagem

$I_{2,i}$ = intensidade do pixel ' i ' na segunda imagem

6. Intensidade média

58. $$f = \frac{1}{n}\sum_{i=1}^{n} I_{2,i} - \frac{1}{n}\sum_{i=1}^{n} I_{1,i}$$

7. Diferença quadrática da intensidade.

59. $$f = \frac{1}{n}\sum_{i=1}^{n} (I_{2,i} - I_{1,i})^2$$

8. Diferença quadrática normalizada da intensidade.

60. $$f = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{I_{2,i}}{\frac{1}{n}\sum_{i=1}^{n} I_{2,i}} - \frac{I_{1,i}}{\frac{1}{n}\sum_{i=1}^{n} I_{1,i}}\right)^2$$

9. Porcentagem de correspondências exatas entre pixels das janelas.

61. $$f = \frac{1}{n}\sum_{i=1}^{n} \Re\left(\frac{I_{2,i}}{\frac{1}{n}\sum_{i=1}^{n} I_{2,i}} - \frac{I_{1,i}}{\frac{1}{n}\sum_{i=1}^{n} I_{1,i}}\right) \quad com \quad \Re(X) = \begin{cases} 0 & se \quad X > e \\ 1 & se \quad X < e \end{cases}$$

e ' e ' o parâmetro de tolerância para correspondência exata:

Após testar várias combinações de métodos, restrições e funções de correlação, a combinação que apresentou melhores resultados foi: Correspondência por meio de janelas nas curvas epipolares com as funções 1 , 2 , 3 , 4 , 5 e 9.

3.6 CÁLCULO DE DISTÂNCIAS

O cálculo de distâncias implementado segue as equações apresentadas anteriormente. A função para esta tarefa recebe dois pontos correspondentes e devolve os dois vetores de distância D_1 e D_2 .

3.7 MAPAS DE DISTÂNCIAS

Os mapas de distâncias gerados pelo programa são a reconstrução da cena pela vista superior em sistema cartesiano de coordenadas.

A rotina implementada para construção dos mapas de distâncias recebe uma lista contendo os pares correlacionados e o vetor distancia D_2 dos pontos ao eixo Z do robô na segunda posição.

Para a construção dos mapas, filtros são aplicados à lista para se encontrar pontos correlacionados que sejam reais e não ruídos. A primeira etapa filtra todos os pontos que não possuam outros pontos a uma mesma distância . A segunda etapa, agrupa pontos a uma mesma distância (z) que se localizem próximos no plano (x, y). A terceira etapa elimina grupos que não possuam um número mínimo de pontos. Todos os parâmetros dos filtros podem ser variados.

Para cada grupo de pontos, é traçada uma linha média no mapa cartesiano.

Uma outra representação possível foi a definição de regiões de concentrações de pontos, ou seja, o mapa foi discretizado em regiões de 10cm por exemplo (parâmetro modificável), de forma que se existam um número mínimo de pontos nesta região, este quadrado "acende", ou seja, há um objeto em sua região. Para a eliminação de ruídos, foi implementado um filtro definindo como requisito mínimo a existência de algum ponto numa região pré-definida em sua volta. Caso não exista este ponto é descartado.

4 RESULTADOS

Foram realizados os testes de validação. O programa e as funções implementadas puderam ser testados e avaliados.

Foram feitas fotos com uma câmera digital perspectiva para os testes com imagens planas e fotos com a câmera omnidirecional instalada no robô para os testes finais.

4.1 IMAGENS PLANAS

4.1.1 LINHAS EPIPOLARES

As imagens abaixo mostram as linhas epipolares em imagens planas. Os deslocamentos entre as fotos foram paralelos ao plano da imagem, por tanto os epipólos não se encontram nas imagens e as linhas são paralelas ao eixo 'x'. Dois pontos foram selecionados (imagem da esquerda) e as correspondentes linhas epipolares foram traçadas pelo programa na imagem da direita.

Figura 21 - imagem (1) plana

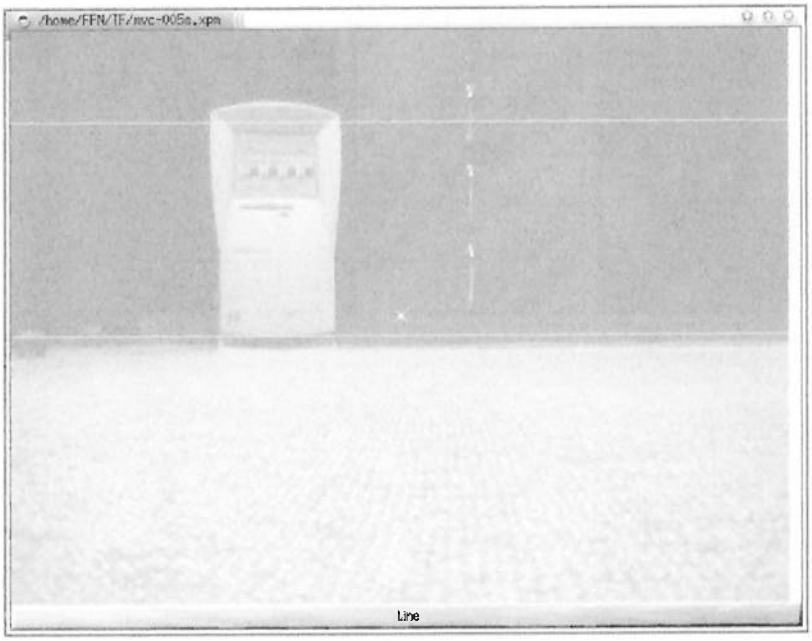


Figura 22 - imagem (2) plana com linhas epiploares

4.1.2 TRIANGULARIZAÇÃO

Para os teste das rotinas de triangularização, foram feitos quatro pares de imagens com deslocamentos distintos entre as imagens e distâncias a pontos reais das fotos conhecidas. A distância focal foi tirada do manual e uma etapa de calibração da câmera foi necessária para achar as outras constantes de calibração. Os resultados estão na tabela abaixo.

Par de Imagens Número	Deslocamento	Distância Real	Resultado	Erro
1	45	180	174	3,3%
1	45	236	236	0
2	30	150	145	3,3%
3	20	180	170	5,6%
4	18	90	92	2,2%

Pode-se ver que os valores obtidos como resultado se aproximam bastante dos valores reais. Os erros permanecem em uma faixa aceitável para o tipo de aplicação proposto e podem ser melhorados com o aumento da precisão das imagens e dos deslocamentos.

4.2 IMAGENS OMNIDIRECIONAIS

Para os testes das rotinas de linhas epipolares foram feitas seqüências de imagens com distâncias a objetos e deslocamentos entre elas conhecidos. A distância focal e as constantes de calibração foram tiradas de [10]. A seguir são mostrados resultados das curvas epipolares, seleção de características, correlação de pontos, cálculo de distâncias e mapas de distâncias.

4.2.1 CURVAS EPIPOLARES

A imagem abaixo mostra os epipólos encontrados em um par de imagens com deslocamentos sobre o eixo 'x'. Os pontos se situam no mesmo eixo, de acordo com o esperado. Todas as linhas epipolares traçadas sobre esta imagem devem passar por estes pontos obrigatoriamente.

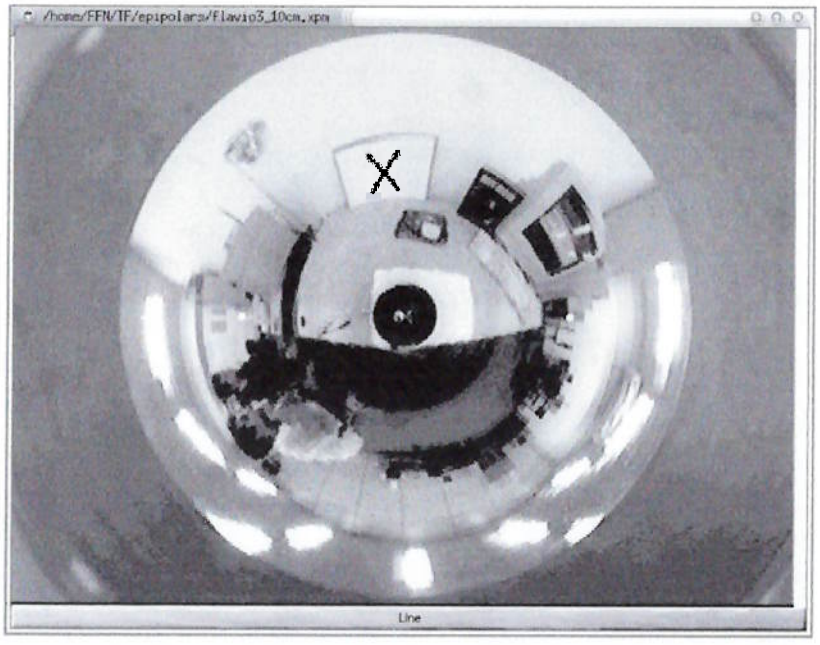


Figura 23 - Imagem omnidirecional com os epipólos obtidos

As imagens a seguir mostram as linhas epipolares em imagens omnidirecionais. Pontos foram selecionados na primeira imagem e as correspondentes linhas epipolares foram traçadas pelo programa na segunda imagem. O primeiro par possui um deslocamento de 100 mm entre as fotos. O segundo possui um deslocamento de 200 mm.

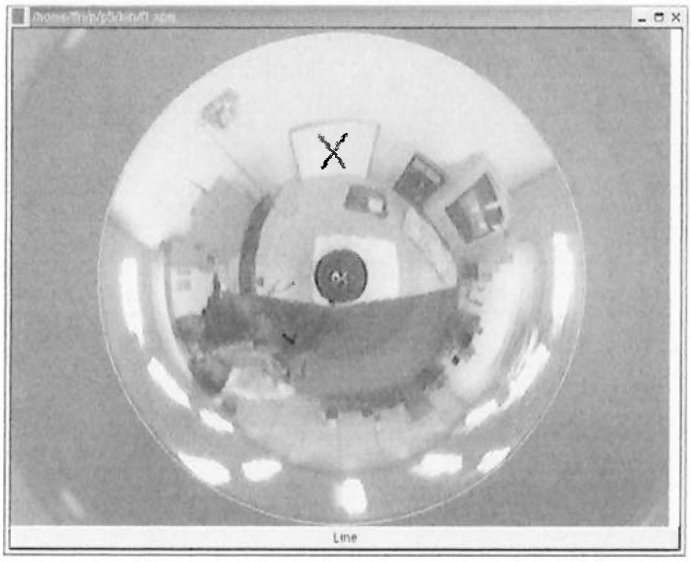


Figura 24 - Imagem 1 - 1° Par -> t = [100 0 0] mm

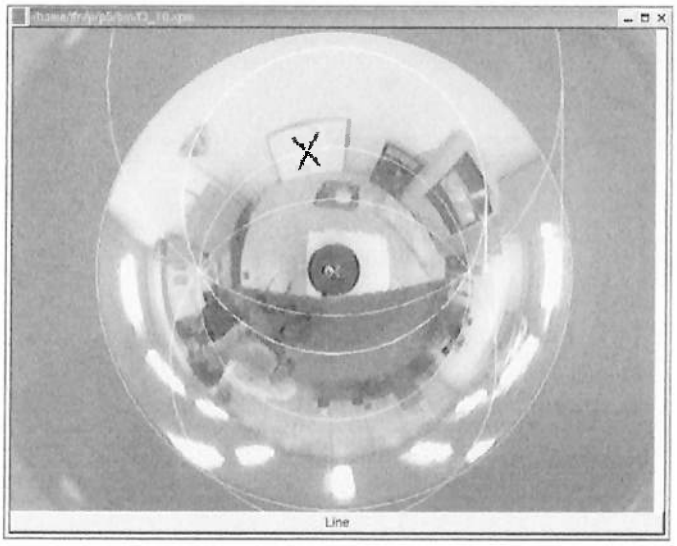


Figura 25 - Imagem 2 - 1° Par -> t = [100 0 0] mm

Na figura abaixo se pode ver o cruzamento das curvas epipolares em um dos epipólos.

Figura 26 - Detalhe das Curvas Epipolares se cruzando em um dos epipolos

O para de imagens abaixo foi retirado das fotos apresentadas acima. No detalhe ampliado pode-se ver a curva epipolar passando pelo centro do ' x ', marcado na imagem da direita.

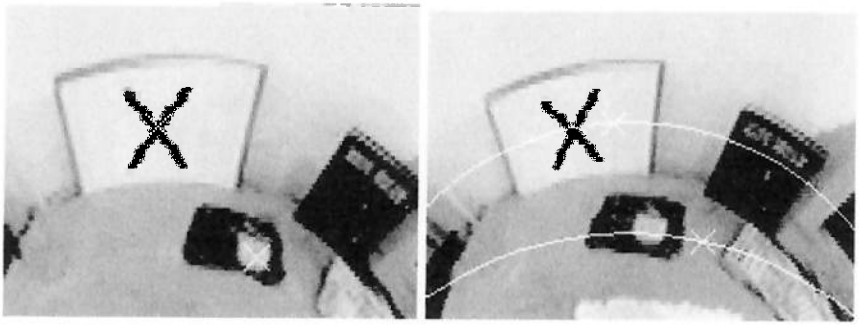


Figura 27 - a) Pontos selecionados na imagem 1 b) Curvas epipolares na imagem 2

Na figura a seguir, foram geradas na segunda imagem as curvas epipolares de todas as quinas na primeira imagem. Pode-se notar que as curvas podem ser geradas para quinas em toda a figura.

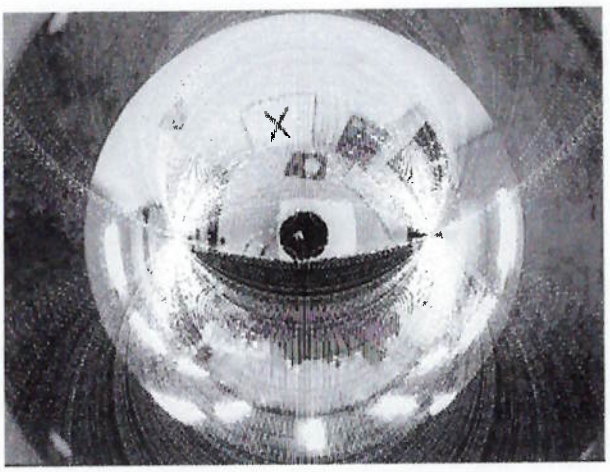


Figura 28 - Linhas Epipolares

No par de imagens abaixo pode-se ver que a precisão das curvas epipolares não foi alterada pelo aumento de deslocamento entre as fotos. As

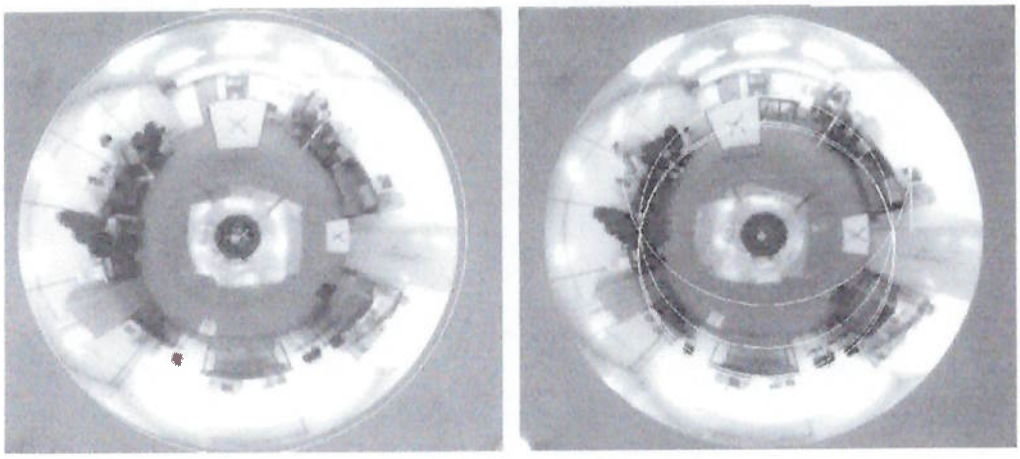


Figura 29 - 2° Par -> t = [200 0 0] mm

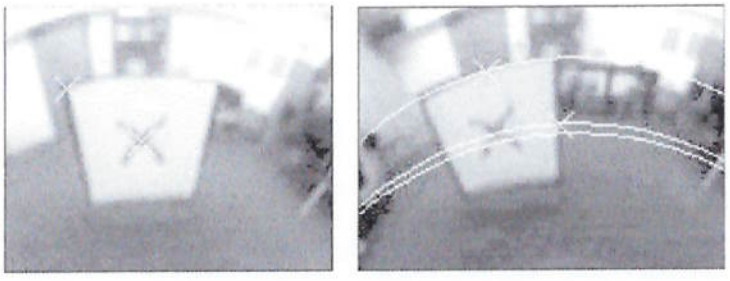


Figura 30 - a) Pontos selecionados na imagem 1 b) Curvas epipolares na imagem 2

Os resultados obtidos com as linhas epipolares foram muito bons. A precisão obtida está de acordo com as expectativas. Pode-se ver nos dois casos apresentados que as linhas epipolares traçadas passam pelos epipólos e passam pelos pontos especificados.

4.2.2 SELEÇÃO DE QUINAS

A imagem abaixo mostra o resultado do algoritmo de seleção de quinas. Pode-se ver que foi possível localizar pontos tanto no primeiro plano (figura do 31) quanto no segundo plano (ambiente)

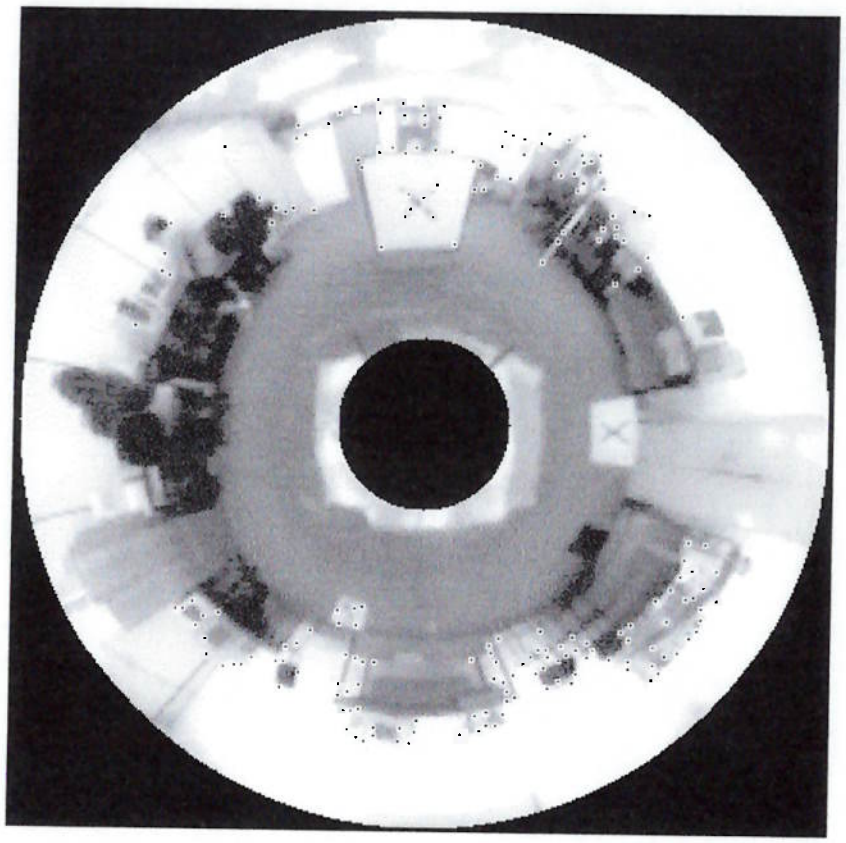


Figura 31 - Seleção de Pontos para correlação

Pode-se ver em detalhes abaixo pontos selecionados nas bordas do X, nas bordas da caixa, na porta ao fundo e nos objetos ao fundo.

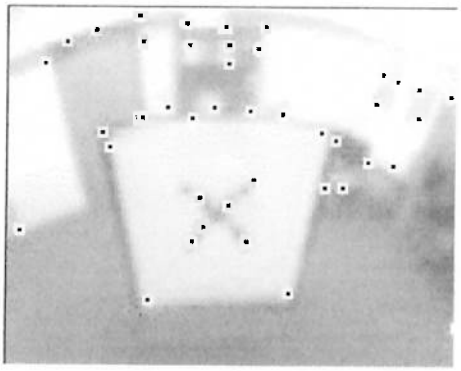


Figura 32 - Seleção de Pontos - Detalhe

Os resultados da seleção de quinas foram bons. Pontos característicos foram selecionados para serem correlacionados na outra imagem. Na figura abaixo está outro exemplo com os pontos obtidos

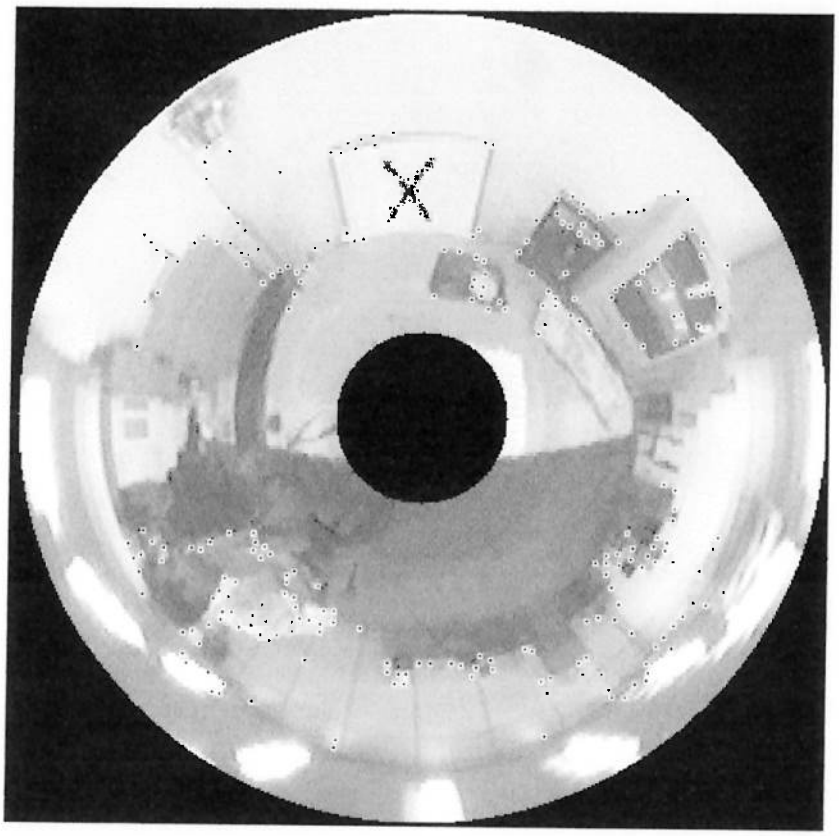


Figura 33 - Seleção de Pontos para correlação

4.2.3 CORRESPONDÊNCIA DE PONTOS (MATCHING)

A partir dos pontos obtidos no passo anterior, os algoritmos de correspondência são utilizados. Na figura abaixo está representado um exemplo da representação obtida para as correspondências com uma movimentação de 10cm. A figura é a imagem na segunda posição, as linhas representam o deslocamento da posição inicial (na primeira imagem) e a final (na segunda imagem), e os quadrados são as posições inicial e final.

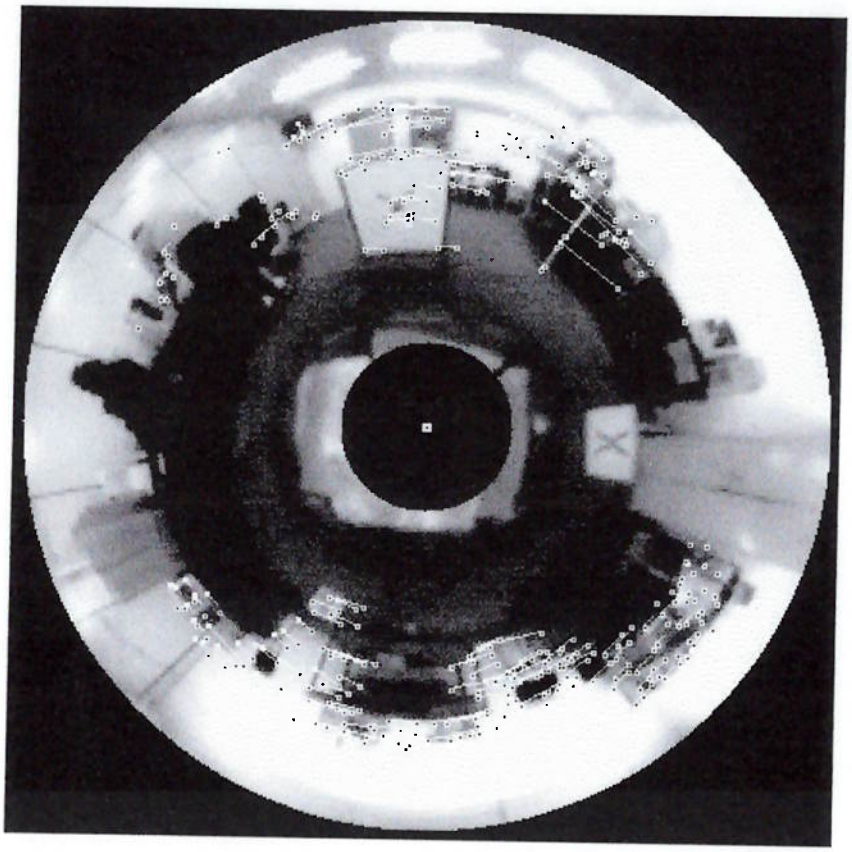


Figura 34 - Matching de Pontos - Deslocamento de 10 cm entre as fotos

Na figura a seguir é mostrado um detalhe das imagens, em que se pode ver o alvo. É interessante notar que a maior parte dos pontos encontra a correspondência exata.

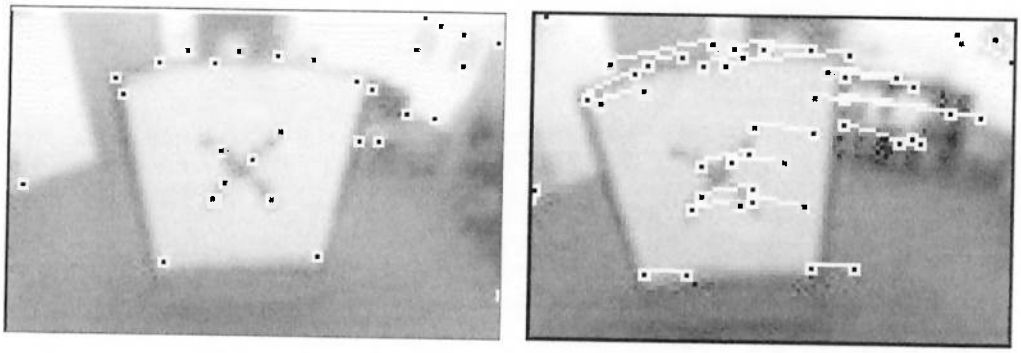


Figura 35 - a) Pontos Selecionados na primeira imagem b) Pontos Correlacionados - Pontos iniciais e finais

No detalhe abaixo, pode-se ver a localização de quinas em um objeto distante do robô e quinas de objetos ao fundo da sala como a mesa e telas de computadores.

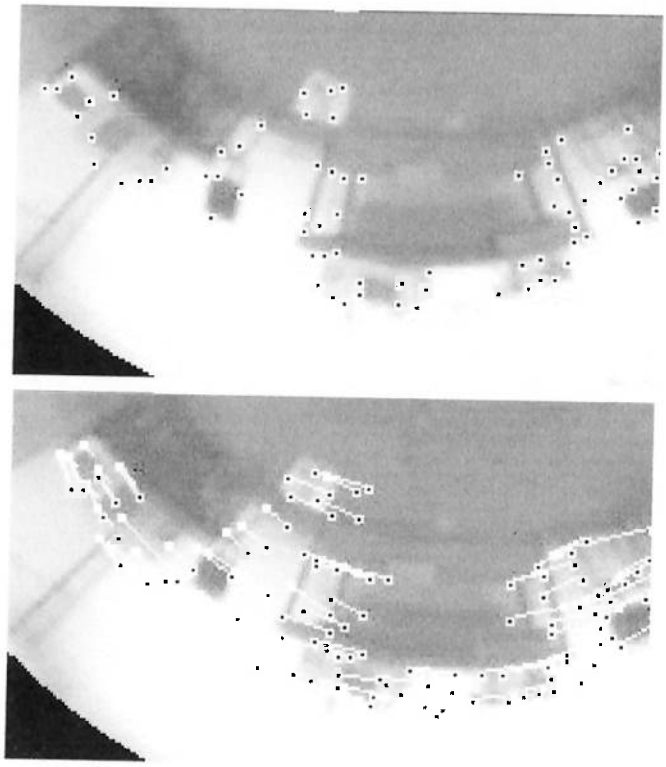


Figura 36 - a) Pontos Selecionados na primeira imagem b) Pontos Correlacionados - Pontos iniciais e finais

A seguir mais um exemplo de correspondência entre os pontos. Note que alguns pontos realmente são falsos matches, mas a maioria corresponde exatamente aos pontos escolhidos

Figura 37 - Matching de Pontos - Deslocamento de 5 cm

Outro ponto interessante de ser notado é o fato de a região perto do eixo horizontal não possui pontos de interesse nem correspondência. Isso se deve ao fato de esta ser a região de deslocamento, e conseqüentemente onde se encontram os epipólos. Nesta região como já fora mencionado anteriormente, os resultados são incoerentes e sujeitos a muito ruído e portanto são eliminados antes do processo de correlação.

Outro ponto interessante de ser ressaltado é o fato de que as correspondências não dependem muito da variação de deslocamento do sistema óptico, ou seja, pelos resultados obtidos, este sistema se comporta bem tanto para deslocamentos de 5cm como para deslocamentos de 30cm.

Acima deste valor os resultados não são muito bons devido ao formato do espelho que faz com que os objetos de uma imagem se distorçam muito na outra janela, além poderem até entrar na região dos epipólos.

Um teste interessante que foi realizado foi a utilização de 2 imagens idênticas, obtendo como resultado a obtenção dos mesmos pontos na maioria absoluta dos casos, confirmando mais ainda a independência relativa do processo dos deslocamentos do robô.

Na figura em seqüência estão em detalhes a região onde se obteve os melhores resultados. Note que no alvo os matches são praticamente todos verdadeiros.

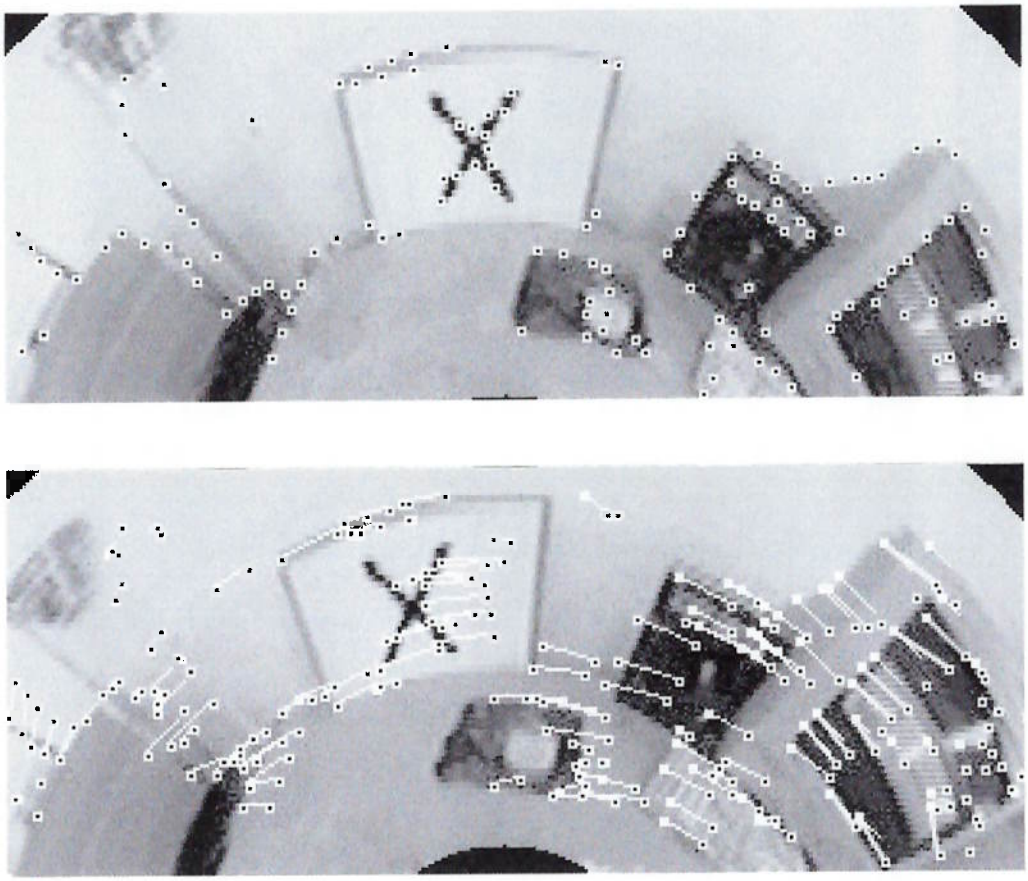


Figura 38 - a) Pontos Selecionados na primeira imagem b) Pontos Correlacionados - Pontos iniciais e finais

Os resultados apresentados até agora utilizam o método escolhido, e que apresentou os melhores resultados. Trata-se do método de correlação por maximização do coeficiente de correspondências exatas entre janelas nas linhas epipolares.

A seguir serão mostrados os resultados dos outros métodos utilizados para confirmar a eficiência do método escolhido em relação aos demais.

Abaixo está o método de correlação sem o uso de epipolares, utilizando apenas uma janela de busca em torno do ponto inicial, e processando o valor de correlação com uma função SSD (soma das diferenças ao quadrado)

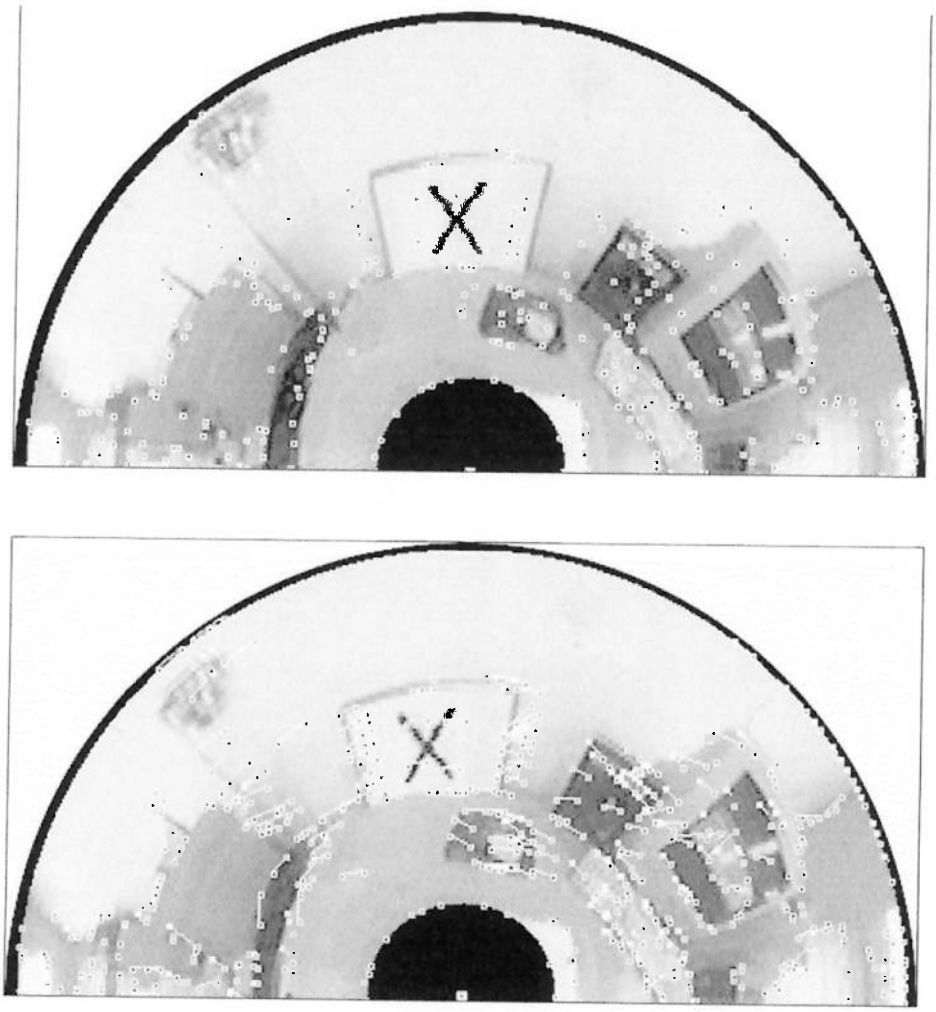


Figura 39 - a) Pontos Selecionados na primeira imagem b) Pontos Correlacionados - Deslocamento de 5cm e método SSD sem epipolar

A primeira vista pode ser um bom resultado com vários matches, porém o deslocamento foi pequeno. Veja a seguir o mesmo método porém agora com um deslocamento um pouco maior.

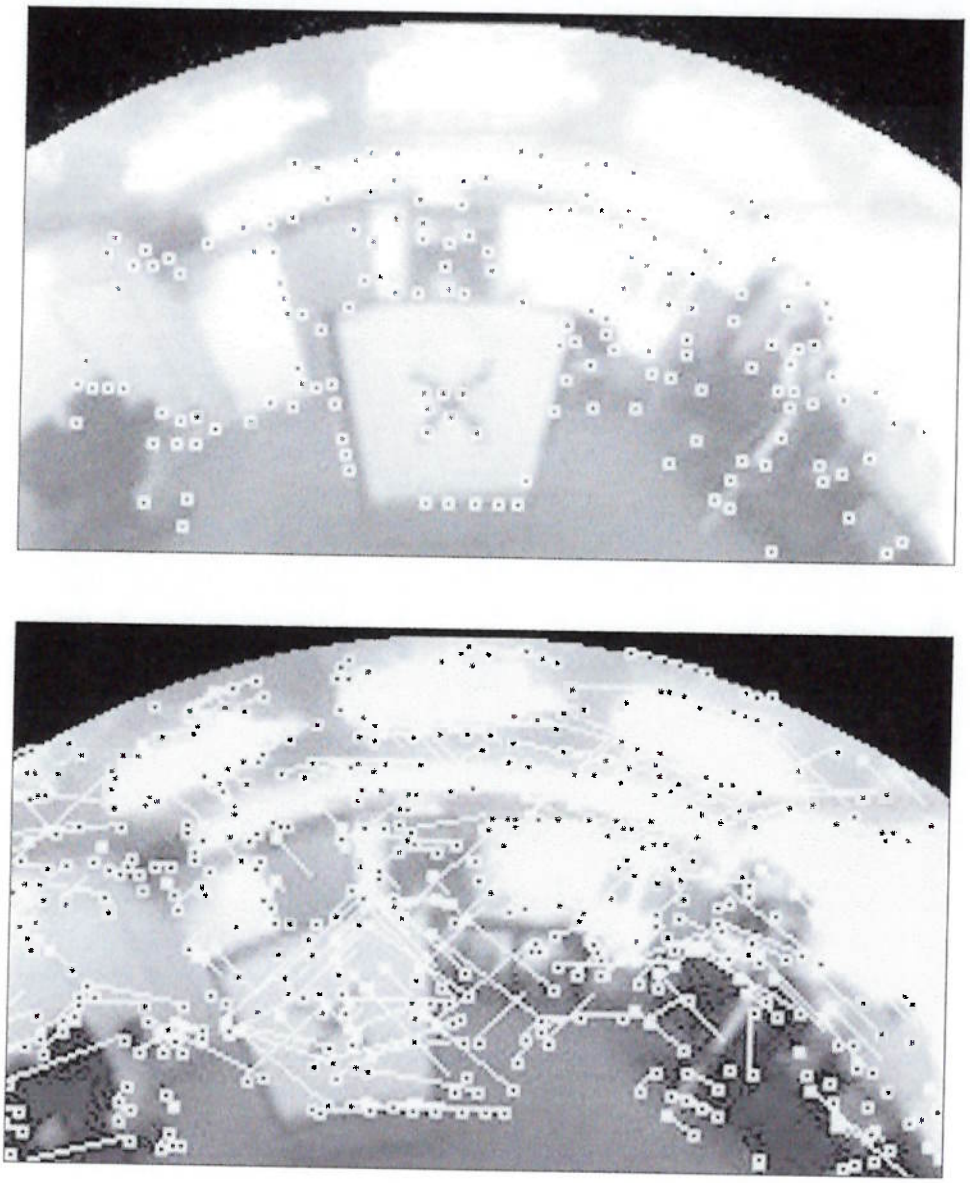


Figura 40 - a) Pontos Selecionados na primeira imagem b) Pontos Correlacionados - Deslocamento de 10cm e método SSD sem epipolar

Como pode ser notado, alguns pontos são achados porém a maioria é perdida. É portanto justificado o uso das epipolares, provando esse ser um

método importantíssimo para guiar as buscas, principalmente no caso de ser necessário o uso de deslocamentos maiores.

Considerando agora o método de busca de janelas em torno da epipolar, é possível ainda verificar diferenças nos resultados considerando métodos diferentes de correlação. A seguir será mostrado os resultados obtidos com o método de MMQ normalizado.

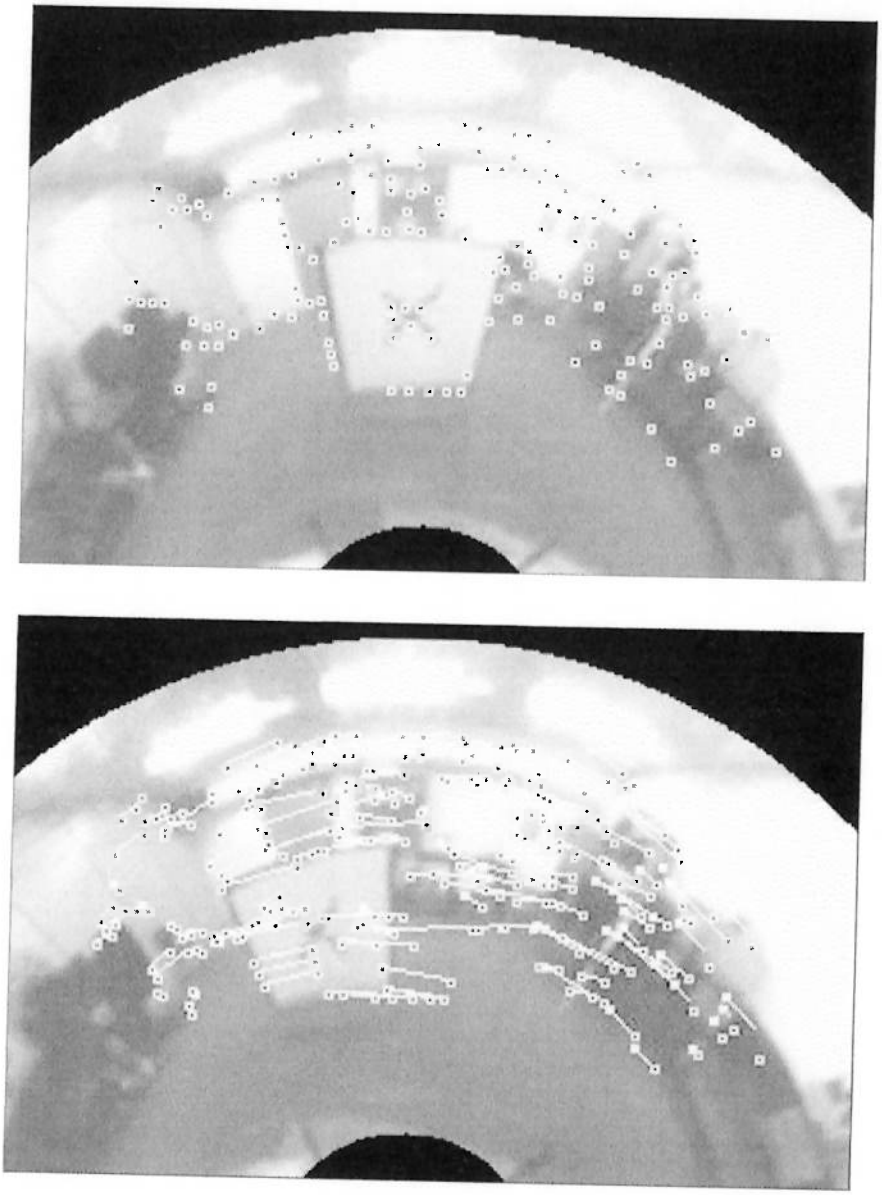


Figura 41 - a) Pontos Selecionados na primeira imagem b) Pontos Correlacionados - Deslocamento de 10cm e método MMQ com epipolar

Como pode ser notado, os resultados são muito superiores aos métodos que não utilizam a epipolar, mas ainda piores que o método escolhido.

Foram testados ainda outros métodos, mas os resultados foram ruins, e não vale a pena mostra os resultados obtidos, apenas comentar um pouco sobre cada um.

O método de busca de corners sobre a epipolar foi um pouco melhor que o método de busca sem a epipolar, porém os resultados ficaram muito aquém do desejado.

Quanto aos métodos de busca em janelas sobre a epipolar, este apresentaram também resultados diversos. O uso de filtros provou ser realmente necessário, obtendo melhoras significativas nos resultados . Desta forma, a comparação final foi feita utilizando em todos os métodos os filtros de restrições e pré-restrições citados anteriormente. O filtro que obteve maior destaque foi o de busca no lado certo da epipolar, dado a direção e sentido do movimento.

Entre os métodos implementados, cabe ainda uma classificação de eficiência entre eles. O método de porcentagem de correlação, como já fora citado, apresentou os melhores resultados. O método de MMQ normalizado apresentou resultados pouco piores que o método anterior. Já o método de MMQ puro, apresentou resultados ruins, principalmente pelo fato das imagens terem certa diferença de intensidade luminosa. Os outros métodos por considerarem a média da intensidade luminosa no cálculo, são menos susceptíveis a estas variações.

4.2.4 CÁLCULO DE DISTÂNCIAS E MAPAS DE DISTÂNCIAS

Como visto nos resultados anteriores a distância de movimentação do robô é um fator importante. Nos mapas de distâncias os resultados levam a conclusão que este fator é mais importante ainda.

Para a análise dos resultados foram gerados alguns mapas com diferentes movimentações e com diferentes representações, além de diferentes quantidades de pontos iniciais, o que gera mapas mais ou menos esparsos. Os métodos implementados foram a discretização do espaço. e o agrupamento de pontos com uso de retas médias (MMQ), médias em X e Y ou ligações de pontos para a representação.

Para a análise consideramos a seguinte cena, com as seguintes correspondências. Note que nesta correlação utilizou-se um maior número de pontos iniciais.

Figura 42 - Matching de Pontos - Deslocamento de 20 cm

Note que as correspondências estão boas e também que foi utilizado um deslocamento de 20cm, um pouco maior que os anteriores.

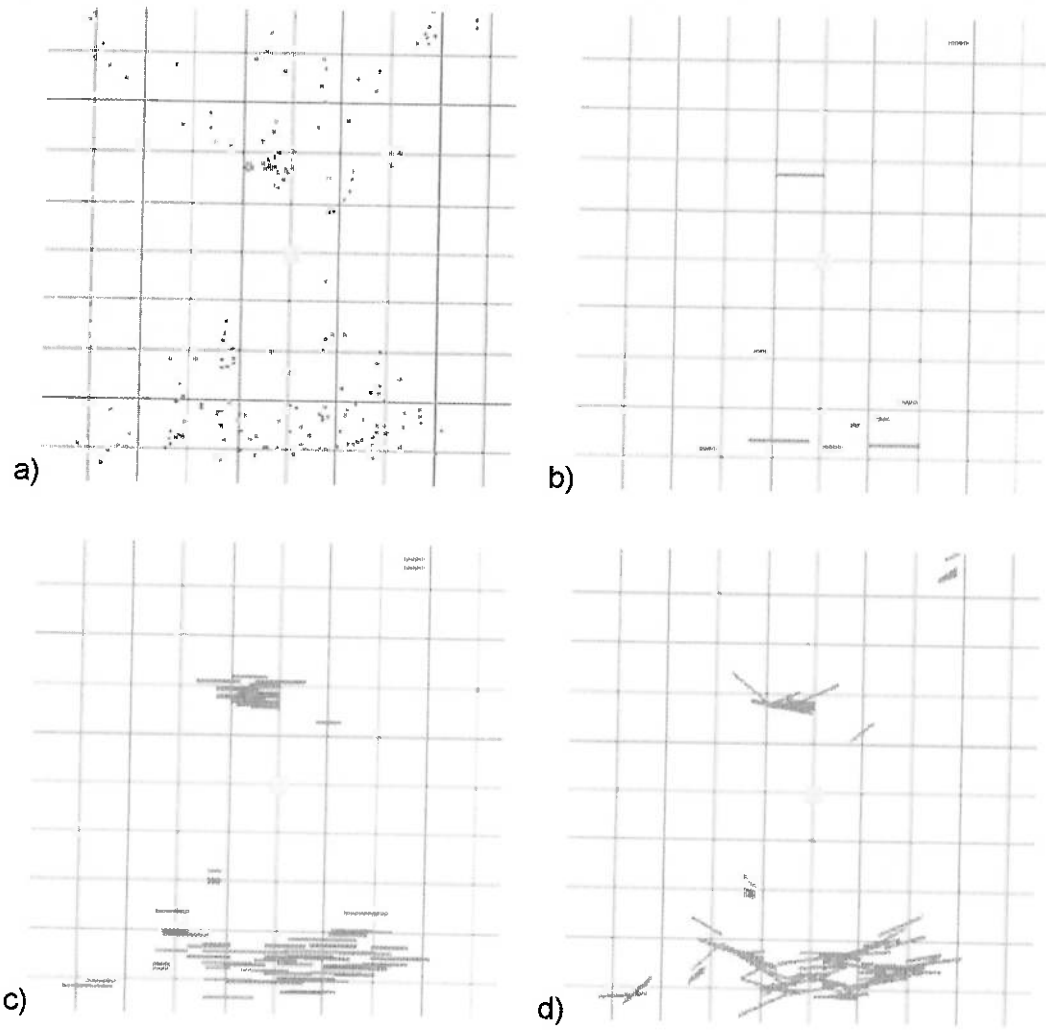


Figura 43 – Mapas de distâncias gerados com deslocamento de 20cm. a) Dispersão de pontos , b) linhas médias em x dos grupos de pontos, c) linha média em x para grupos múltiplos , d) MMQ para grupos múltiplos

Na figura acima vemos os diversos métodos implementados para visualizar a dispersão dos objetos em relação ao robô. As divisões correspondem a 50cm e o centro da figura, marcado com um quadrado é a posição do robô. Todas estas distâncias estão em relação à segunda posição do robô, que corresponderia à posição atual dele, dada na figura anterior a esta. Em todos os métodos é possível distinguir com uma certa precisão os objetos existentes no ambiente.

Uma outra forma de representação foi a citada anteriormente, na qual o mapa é discretizado em quadrado com tamanhos definidos, os quais só serão "acessos" ou melhor conterão alguma informação no caso de haver mais de um número pré-fixado de pontos que estão nesta região. Acertando os parâmetros é possível obter uma boa representação do ambiente. Na figura a seguir vemos esta representação.

Figura 44 - Imagem da cena e matches gerados para deslocamento de 20cm

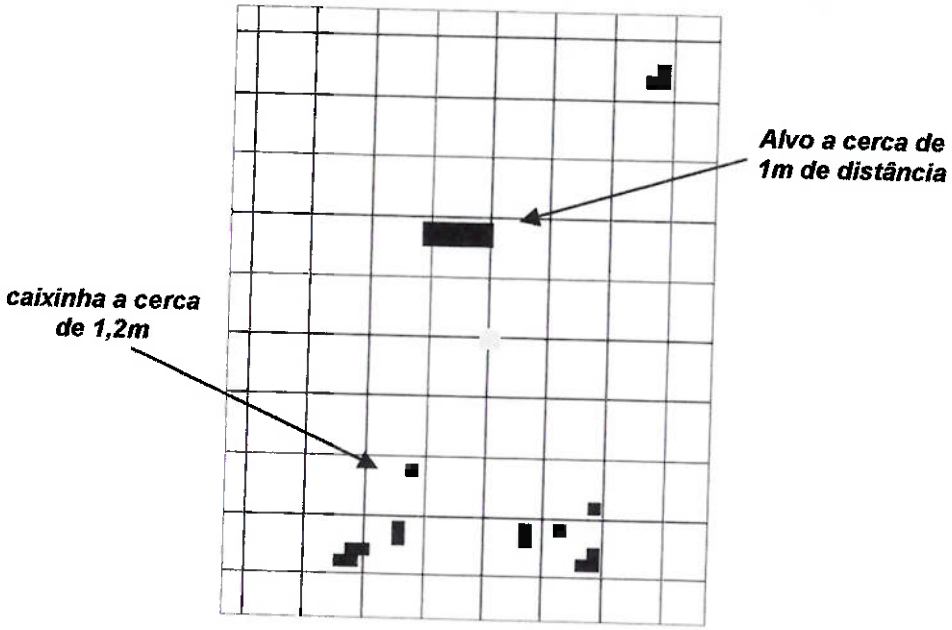


Figura 45 - Mapa gerado pelo método de discretização com área de 4x4 cm

A representação gerada é bem interessante e demonstra uma certa precisão e parece melhor que as demais representações.

Na figura a seguir é feita uma análise do resultado quanto às distâncias obtidas. Os resultados obtidos são para uma distância de 20cm.

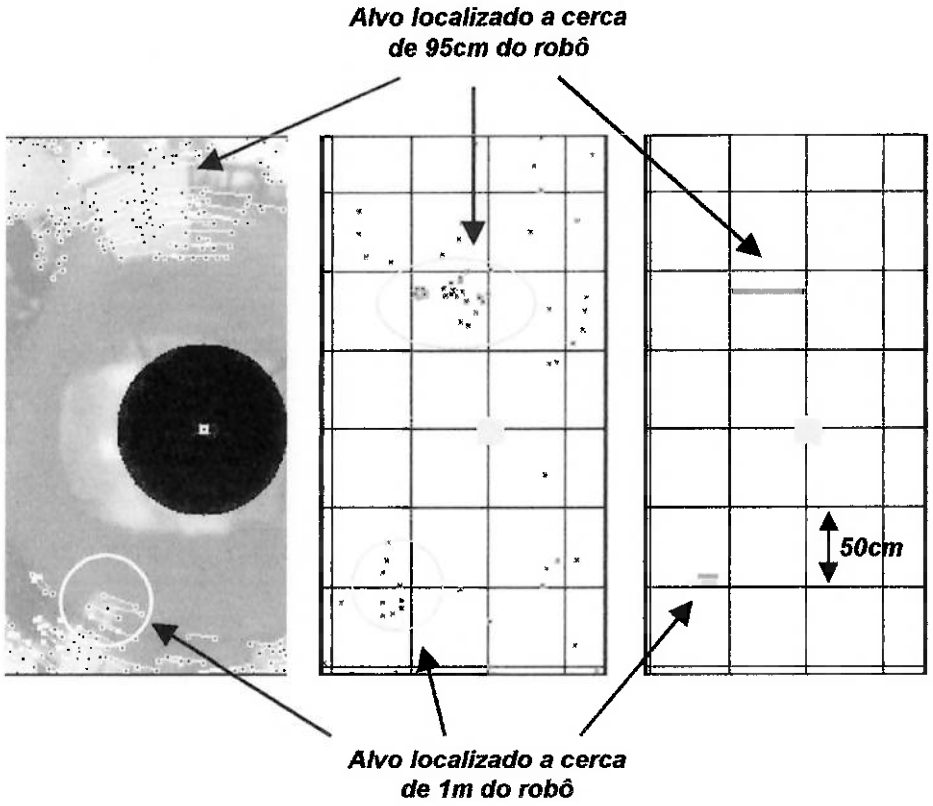


Figura 46 - análise do resultado obtido, a) imagem real, b) dispersão dos pontos correlacionáveis, c) mapa com linhas médias

Nesta seqüência de imagens é possível verificar que dentro de um certo padrão de precisão o sistema consegue encontrar objetos. Para distâncias maiores, os erros aumentam tornando o resultado um pouco vago. Por exemplo na Figura 43 b é possível identificar alguns traços na parte inferior, que segundo a Figura 42 seria a representação da mesa. Pelo diagrama eles estariam à cerca de 2m, porém estavam mais distantes, aproximadamente a 2,5m.

Um outro exemplo também com 20cm de deslocamento pode ser visto na figura seguir, onde alguns objetos podem ser identificados e possuem até um certo grau de acerto.

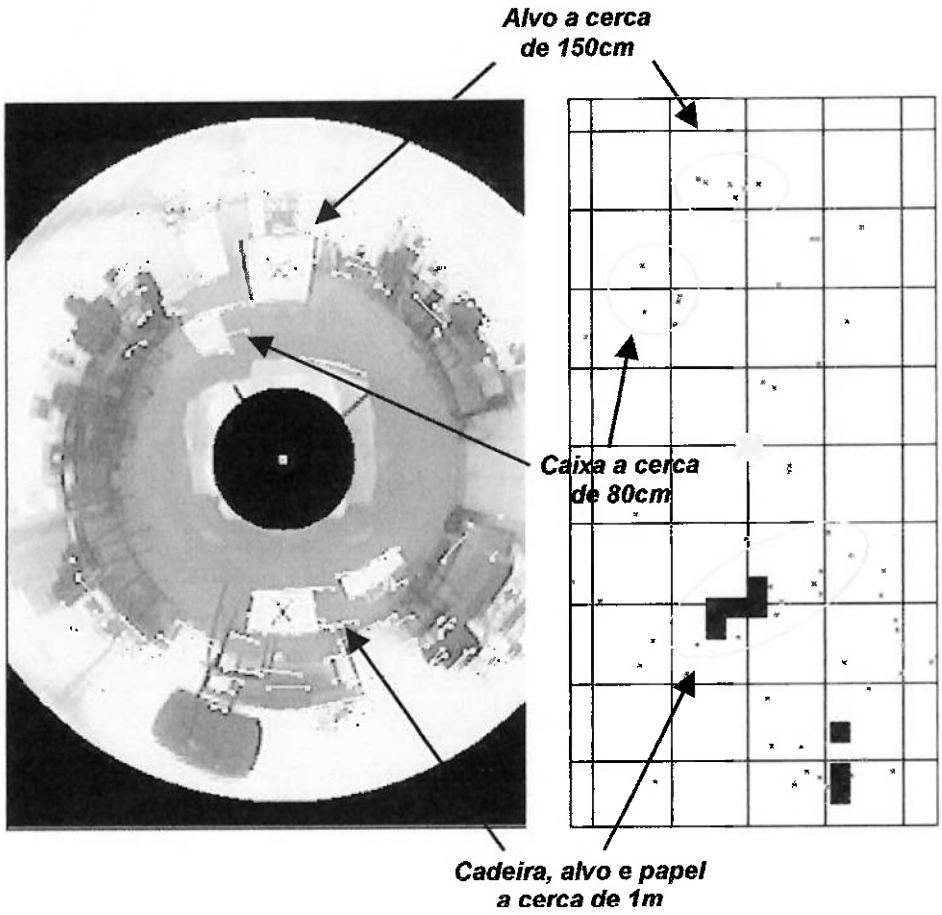


Figura 47 - Mapa de distâncias gerados para um deslocamento de 20cm - a) imagem na segunda posição com as correspondências, b) mapa de distâcia gerado

Diminuindo-se a distância de deslocamento do robô pode-se observar o novo mapa gerado na próxima figura.

Pode ser notado que com distâncias menores os pontos do fundo tendem a se aproximar do centro, além de alguns pontos causarem ruídos. O alvo (que estava a 1m aproximadamente) nos dois mapas gerados continua a aparecer com uma certa precisa, porém o fundo da sala (dispersão de pontos na parte inferior dos mapas) se aproximou muito do robô.

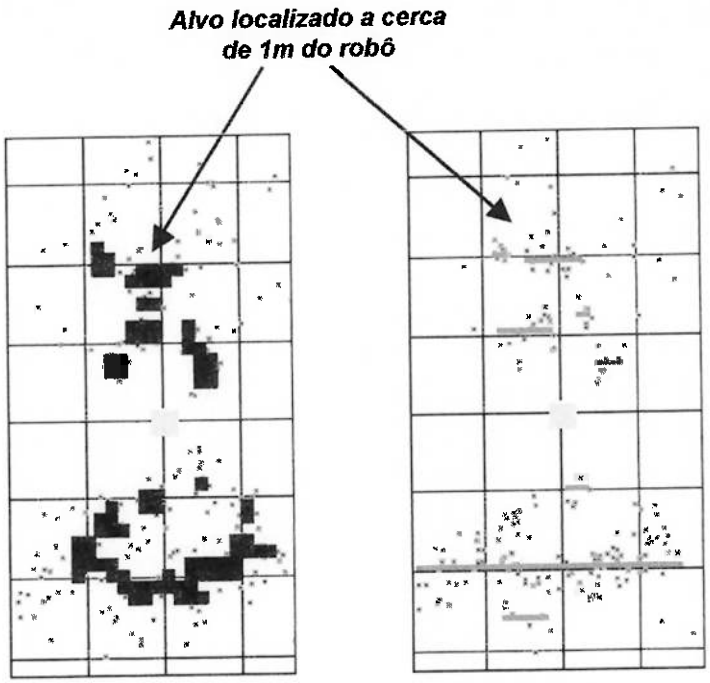


Figura 48 - Mapas gerados com distâncias de 10cm entre as fotos – a) mapa gerado com sipesão depontos e também mapa discretizado (quadrados pretos), b) mapa gerado com dispersão de pontos e linha mádia para agrupamentos únicos.

Como pode ser percebido e pelo testes e inúmeros resultados obtidos, o parâmetro que influencia muito o resultado é a distância percorrida pelo robô. Analisando os resultados, a distância mínima para gerar mapas não muito discrepantes, seria de 20cm, ou mais. Porém aumentando-se a distância, os matches, começariam a se tornar mais discrepantes. Desta forma, nota-se que a distância ideal seria por volta de 20cm. Porém mesmo assim, objetos localizados muito distantes estariam com um pouco de erro na distância, problema este devido em grande parte aos erros de precisão e arredondamentos nos algoritmos. Devido a grande deformação gerada pelo espelho, os objetos a grande distância são deformados, e diferenças de alguns poucos pixels na disparidade significariam vários centimetros no mapa final.

5 CONCLUSÕES E DISCUSSÃO FINAL

Os resultados obtidos na primeira parte do trabalho obtiveram resultados muito bons. O traçado de linhas epipolares foi feito de forma eficiente, obtendo linhas epipolares que cruzavam a imagem em pontos correspondentes aos pontos selecionados com precisão de pixels. Desta forma a busca de correspondência obteve um ferramental eficiente para sua realização.

Muitos métodos foram estudados, sendo que a uma primeira vista vários poderiam ser aplicados ao caso. O grande problema neste tipo de imagens é a sua deformação e a sua forma polar, o que faz com que muitos dos algoritmos usuais, não funcionem adequadamente nesta situação. Muitos dos algoritmos necessitam da retificação da imagem, o que pode até ser feito porém seria mais uma etapa, que geraria mais erros, e consumiria mais tempo de processamento.

Buscou-se desta forma um processo que fosse menos susceptível a estas características da imagem. O método escolhido , a correlação, é um dos métodos mais comuns encontrados na literatura porém pelos resultados obtidos neste trabalho e na literatura, mostrou-se muito eficiente e estável.

Ao serem utilizados pré-restrições e filtros para a eliminação de falsos matches, os resultados se mostraram mais precisos ainda. O processo final de correspondência de uma forma geral apresentou resultados bons, além do esperado.

Quanto à geração de mapas de distâncias o maior problema encontrado foi a dispersão dos pontos devido à escolha por um mapa mais esparso. Ao se escolher pontos como as características a serem extraídas e assim correlacionadas, ficou-se sujeito a este tipo de problema. Na literatura foram encontrados algoritmos que trabalhavam com retas para a reconstrução final, porém devido a sua forma, a imagem restringia este tipo de forma. Trabalhando-se com pontos, a susceptibilidade a ruídos foi aumentada. Pontos perto e com matches perto, devido a erros de precisão e discretização poderiam estar até um pouco distantes, gerando um certo

problema para uma representação final. Os algoritmos implementados para a representação final tentaram inibir este tipo de problema, chegando a resultados satisfatórios. Por fim podemos citar que ruídos, imprecisões, e arredondamentos para a discretização, causaram uma certa discrepância no resultado final, principalmente para distâncias maiores que 1m.

Um parâmetro que influenciou muito na precisão do mapa final foi o deslocamento do robô. O deslocamento mínimo necessário para obtenção de mapas de distância, considerado como ideal, foi de cerca de 20cm. Deslocamentos maiores tendem a gerar problemas com as correlações. Deslocamentos menores causam problemas com os mapas de distâncias.

Quanto à velocidade do algoritmo pode-se citar que uma aplicação em tempo-real é totalmente viável. Para ter-se idéia da ordem de grandeza do tempo de execução, o algoritmo atual conclui todas as etapas em torno de 3s em um Pentium III 550Mhz. O algoritmo de extração de características utilizado é muito veloz, e é o que apresenta maior desenvolvimento computacional, uma vez que já vem sendo desenvolvido há anos. Os outros algoritmos implementados podem ser melhorados podendo apresentar uma melhora significativa no tempo de execução.

Por fim, pode-se concluir que o resultado final do trabalho foi satisfatório, e com algumas modificações aponta para um sistema eficiente de navegação. Para a continuidade deste trabalho, poderia ser criado um algoritmo que obtivesse algumas seqüências de imagens e através delas gerasse um mapa mais consistente do ambiente. Outro ponto necessário seria a implementação de um algoritmo para sanar o problema da falta de informações consistentes na região dos epipólos, ou seja na região do movimento. Adicionando estes pontos ao trabalho atual seria possível a implementação de um algoritmo útil para a navegação de robôs com apenas um sensor.

6 BIBLIOGRAFIA

[1] Shou,Kang Wei *Stereo Matching of Catadioptric Panoramic Images*. Center for Machine Perception – Czech Technical University, 2000.

[2] T. Svoboda, T Pajdla, V. Hlavác. *Central Panoramic Cameras: Geometry and Design*. Research report, No. K335/97/147, Dezembro, 1997.

[3] Y. Yagi. *Omnidirectional sensing and its applications*. IEICE Trans. Inf. & Syst., Vol. E82-D, NO. 3, Março 1999

[4] M. Ollis, H. Herman, S. Singh. *Analysis and Design of Panoramic Stereo Vision Using Equi-Angular Pixel Cameras*. Technical Report CMU-RI-TR-99-04, January, 1999

[5] K. Moriya, K. Ohba. *Image Projection Criteria with the Epipolar Line for a Precise Correlation with Omni Images*. 1999

[6] T. Svoboda, T Pajdla, V. Hlavác. *Epipolar geometry for Panoramic Cameras*. In fifth European Conference on Computer Vision, Freiburg, Alemanha, pg 218-232, LNCS 1406, Springer ,Junho, 1998.

[7] E. Trucco, A. Verri. *Introductory Thechniques for 3-D computer vision*, Prentice Hall, 1998.

[8] T. Svoboda. *Evaluation, transformation, and parametrization of epipolar conics*. Research Report, No. CTU-CMP-2000-11, Julho, 2000

[9] T. Svoboda, T Pajdla, V. Hlavác. *Motion Estimation Using Central Panoramic Cameras*. In IEEE Conference on Intelligent Vehicles, Stuttgart, Alemanha, pg 335-340, Outubro, 1998.

[10] V. Grassi Jr, C. C. G. Deccó, J. Okamoto Jr., A. J. V. Porto. *Desenvolvimento de Um Sistema de Visão Omnidirecional*. Technical Report TRB0073. Abril, 2001

[11] N. Ayache. *Artificial Vision for Mobile Robots – Stereo Vision and Multisensory Perception*. IMT Press, 1991.

[12] R. Owens, *Computer Vision: Camera calibration* http://www.dai.ed.ac.uk/Cvonline/LOCAL_COPIES/OWENS/LECT9/ , 1997

[13] M. Tacoshi Jr. *Paralelização de um algoritmo para visão estáereo computacional por análise de diferença de fase entre imagens.* Dissertação de Mestrado da Escola Politécnica da USP, Departamento de Engenharia de Computação e Sistemas Digitais, São Paulo, 1994

[14] R. C. Gonzalez. *Digital Image Processing.* Addison-Wesley Publishing Company, 1992

[15] J. E. W. Mayhew, J. P. Frisby. *3D Model recognition from stereoscopic cues.* MIT Press, 1991

[16] W. E. Grimson. *Object recognition by computer.* MIT Press, 1990

[17] S. M. Smith, J. M. Brady. *SUSAN – A New Approach to Low Level Image Processing.* Technical Report, TR95SMS1c, 1995

[18] O. Faugeras, B. Hotz, H. Mathieu, T. Viéville, Z Zhang, P. Fuá, E. Théron, L. Moll, G. Berry, J. Vuillemin, P. Bertin, C. Proy. *Real time correlation-based stereo: algorithm, implementations and applications.* Research report, n° 2013, INRIA, 1993

[19] S. Birchfield, C. Tomasi, *A Pixel Dissimilarity Measure That Is Insensitive to Image Sampling*, IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 20, NO. 4, pg 401-406, Abril, 1998.

[20] M. Sonka, V. Hlavac, R. Boyle, *Digital Image Processing - Image Analysis and Understanding - Chapter 9, 3D Vision (Part I): Geometry for 3D vision,* University of Iowa, http://www.icaen.uiowa.edu/~dip/LECTURE/ , 1997

[21] J. Blandy, *An Ellipse Drawing Algorithm*, http://www.cs.oberlin.edu/students/public/unixhelp-local/ellipse_drawing.html, 1994

[22] O. Faugeras, *Three-Dimensional Computer Vision*, MIT Press, Novembro, 1993.

[23] T. Svoboda, T. Padja, *Panoramic cameras for 3D computation,* In Procedingngs of the Czech Pattern Recognation WorkShop, pg 63-70, Fevereiro, 2000.

[24] J. Gluckman, S. K. Nayar, K. J. Thoresz, *Real-Time Omnidirecional and Panoramic Stereo,* Department of Computer Science, Columbia University, Nova York.

[25] Z. Zhang, *A Flexible New Technique for Camera Calibration,* Technical Report, No.MSR-TR-98-71, Dezembro, 1998

[26] J. Hee Han, J. S. Park, *Countour Matching Using Epipolar Geometry,* In IEEE Transaction on Patern analysis and machine intelligence, vol 22, no 4, pg 358-369, Abril, 2000.

[27] R. Deriche, O Faugeras, Z. Zhang, Q. T. Luong, *A Robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry,* Research Report, No 2273, Maio, 1994.

[28] G. Medioni, R. Nevatia, *Matching Images Using Linear Features,* In IEEE Transaction on Patern analysis and machine intelligence, vol PAMI-6, no 6, pg 675-685, Novembro, 1984.

[29] C. Geyer, K. Daniilidis, *Catadioptric Camera Cailbration,* In Proceedings International Conference on Computer Vision, pg 398-404, Novembro, 1999.

[30] H. M. Hang, Y. M. Chou, *Motion Estimation,*

ANEXOS A

O PROGRAMA

A implementação do programa para o traçado de linhas epipolares e geraçÃo de mapas de distância foi feita em linguagem C e plataforma Linux. As ferramentas GDK e GTK foram utilizadas para dar suporte às rotinas gráficas do programa. Para desvincular as rotinas específicas do trabalho das ferramentas comerciais, o código relacionado à visão estéreo foi separado em arquivos '.h' e carregados como 'include' no programa principal.

O programa está dividido basicamente em duas estruturas, que são 'c_Image' e 'c_Pair', e funções relacionadas a cada uma delas. A estrutura 'c_image', possui todos os atributos da imagem, tanto os atributos para sua parte visual como os atributos para os algoritmos de tratamento de imagens. Já a estrutura 'c_Pair' possui os atributos das imagens do par e das características do par em si como a translação. Para maiores informações basta consultar o ANEXO B na biblioteca "Classes.h", onde estas estruturas estão comentadas.

Estruturalmente ele também está dividido em duas partes. A primeira parte está relacionada com as linhas epipolares. A segunda parte está relacionada com a geração de mapas de distância. As funções relacionadas com as epipolares trabalham com fotos .XPM, enquanto que as funções relacionadas com mapa trabalham com fotos .PGM. Outro ponto importante é que as rotinas de visualização de epipolares rodam em janelas, enquanto que as de mapas de distâncias, gravam arquivos.

USO DO PROGRAMA

O procedimento para o uso do programa encontra-se abaixo.

EPIPOLARES

- Carregam-se duas imagens .XPM no programa (botões Load Image 1&2)
- Entra-se com os parâmetros extrínsecos – vetor translação e vetor rotação entre as duas posições da câmera – e parâmetros intrínsecos – distância focal e constantes de proporção pixel/milímetros. Uma janela de diálogo para a entrada destes valores é acionada pelo botão 'Get Parameters'. Devido a calibração da câmera, os parâmetros instrinsicos já estão definidos diretamente no programa.
- Com o botão 'Show Images', as imagens previamente carregadas são mostradas na tela.
- O cálculo de distâncias em imagens planas é feito selecionando com o botão do mouse dois pontos correspondentes nas imagens e pressionando-se o botão 'Triangularization'.
- As linhas epipolares são geradas selecionando-se um ponto na primeira imagem e apertando-se o botão 'Epipolar' para imagens planas e 'Epipolar Conics' para imagens omnidirecionais.
- Os epipólos em imagens omnidirecionais podem ser vistos pressionando-se o botão 'Epipoles'.

MAPAS DE DISTÂNCIAS

- Carregam-se duas imagens .PGM no programa (botões Load Image 1&2)
- Entra-se com os parâmetros de translação
- Com o botão 'SUSAN', o processo de geração de mapas de distâncias é iniciado

- Os resultados estarão no diretório do programa nos arquivos Corner1.pgm (arquivo com os pontos selecionados), Corner2 (arquivo com os pontos correlacionados e as linhas de trajetória dos pontos) e Corner_Map.PGM (arquivo com a vista de topo da cena)

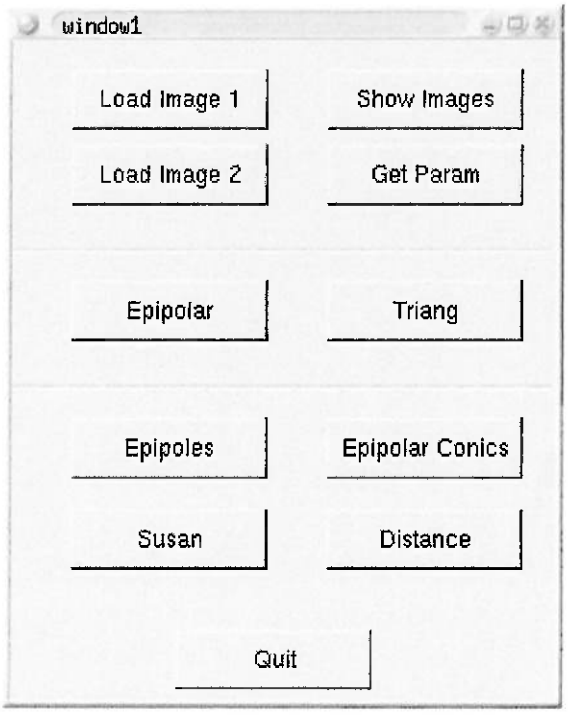


Figura 49 - Tela de controle do programa implementado

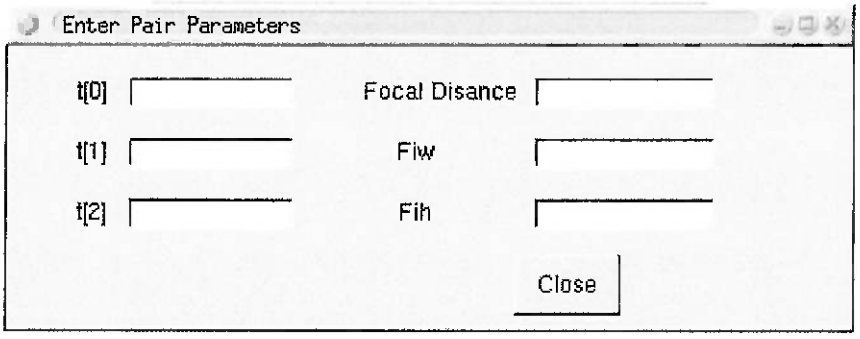


Figura 50 - Tela para obter os parâmetros

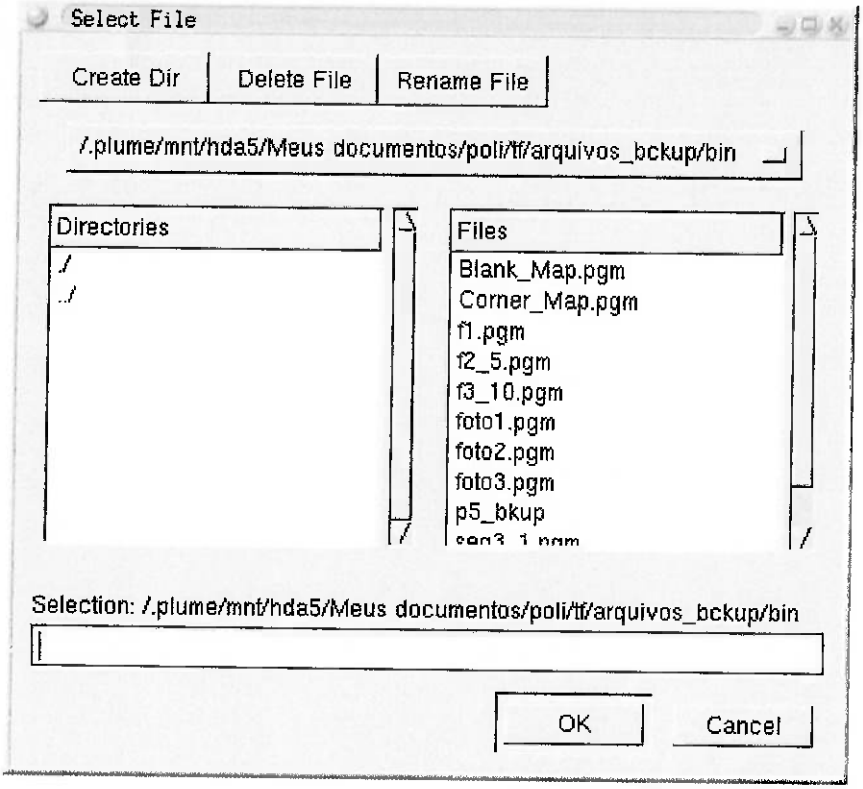


Figura 51 - Tela de seleção de arquivos pa carregamento de imagens

VISÃO GERAL DO PROGRAMA

Como dito anteriormente o programa está dividido em bibliotecas especificas de cada parte do projeto. As bibliotecas estão no ANEXO B devidamente comentadas, e para uma visão geral delas são citadas abaixo:

- **s.h** : biblioteca com as funções para extração de características e de interface entre arquivo e imagem no programa. Possui os algoritmos SUSAN citados na parte teórica.
- **Classes.h**: biblioteca com as definições de estruturas usadas no programa, assim como variáveis globais e funções relacionadas com estas estruturas como as funções de cálculo de distância e traçado de epipolares
- **Callbacks**: bibliotecas com as funções que fazem a interface dos botões com suas ações. Estão nesta biblioteca as funções primárias de execução do programa.

- **Correlação.h**: biblioteca com as funções de correlação entre as imagens, assim como algumas funções de tratamento da imagem (retirar a borda e o centro da imagem), e interface da imagem carregada à estrutura do programa.
- **Maps.h:** biblioteca com as funções de geração dos mapas de distâncias e das funções de desenho e marcação na imagem.
- **Libmatrix.h:** biblioteca com as definições das estruturas matriciais e vetoriais necessárias para o programa, assim como algumas funções relacionadas, como multiplicação, cofator, etc.
- **Libffn.h:** biblioteca com algumas funções auxiliares
- **Functions.h:** biblioteca com funções adicionais da parte gráfica como as funções para obter parâmetros.
- **Support.h**: funções do glade, ou seja, funções que geram as janelas.
- **Interfaces.h:** Outra biblioteca criada pelo glade para definir a parte gráfica.

O programa segue basicamente o seguinte fluxo:

Carregamento de imagens (.XPM ou .PGM) - Inserção dos nomes à estrutura 'c_Pair' e 'c_Image'
Entrada dos parâmetros - Definições dos parâmetros à estrutura 'c_Pair'
Execução de alguma rotina

Traçado de Epipolares	*SUSAN*
- execução de rotinas para visualização das imagens - obtenção de dados externos (coord. X,Y) na imagem 1 - traçado das epipolares na imagem 2	- obtenção dos dados dos arquivos externos e inserção nas estruturas 'in' de cada 'c_Image' - execução da função de obtenção de corners 'fr_find_corners' e inserção dos dados obtidos na estrutura 'corner_list' de 'c_image' - execução de alguma função de correlação. A função com melhores resultados é a 'fr_window_on_ep' - execução de funções para apresentação dos resultados como 'fr_upper_view' - gravação dos strings para arquivos

ANEXOS B

CLASSES.H

```
/*************************************************************

Definicoes de classes e estruturas e funcoes relacionadas

*************************************************************/


#include <gnome.h>

#ifndef _Classes
#define _Classes

#include "libffn.h"
#include "libmatrix.h"
#include "s.h"

//------------classe c_Picture-------------------//

struct c_Picture
{

      gchar             *name;            // Armazena o nome da
imagem carregada no programa
      GtkWidget   *pixmap;    // ponteiro usado para carregar
imagens xpm
      GtkWidget   *widget;    // ponteiro da janela de vizualizacao
de imagens

      gint        x;          // Parametros para armazenar o tamanho
das imagens
      gint        y;          //
      gint        width;            //
      gint        height;           //

      guint       cx;         //Ponto para armazenar o centro da
imagem
      guint       cy;
      guint       radius;           // Raio da imagem ominidirecional

      guint       epipole1;   //Localizacao dos epipolos na imagem
      guint       epipole2;

      guint       triangx;    //Ponto para armazenar coordenadas
dadas pelo usuario
      guint       triangy;    // com o mouse.
```

```
//-------Susan--------//

      uchar *in,        // ponteiro com a posicao inicial da figura
            *bp,        // ponteiro com a LUT definida pela funcao
para achar corner
            *map;       // pontiero para definicao da figura
      int   *r,
            bt,
            drawing_mode,
            max_no_corners,   // Numero maximo de corners permitido
            x_size, y_size; //

      CORNER_LIST       corner_list; //Estrutura para armazenar
pontos correnacionados (coordenadas e distancias)

};
typedef struct c_Picture c_Picture;

// Imprime as informacoes da estrutura c_Picture na tela
void
fr_print_picture_data(c_Picture*);

//Carrega um arquivo xpm em uma tabela da janela
void
fr_load_pixmap_on_fixed(GtkWidget *picture_window, c_Picture
*picture,GtkWidget *fixed1,
                              guint x, guint y, guint dx, guint dy);

//Carrega um arquivo xpm em um box da janela
void
fr_load_pixmap_on_box(GtkWidget *picture_window, c_Picture
*picture,GtkWidget *vbox);

// Inicializa os parametros de uma imagem (cx, cy...)
void
fr_set_picture_param(c_Picture*,gint,gint,gint,gint);

//Carrega um arquivo xpm no ponteiro
GtkWidget*
fr_load_pixmap(GtkWidget *picture_window,c_Picture *picture);

//Cria e mostra a janela contendo a imagem
GtkWidget*
fr_create_picture_window(c_Picture *picture);

// Inicializa os parametros da janela
void
on_picture_window_show (GtkWidget*, c_Picture*);

// Responde a cliques do mouse na janela
gboolean
on_window_button_press_event          (GtkWidget       *widget,
                                       GdkEventButton  *event,
                              c_Picture         *picture);

// Redesenha a imagem na janela, apos alguma alteracao
void
fr_redraw_picture(GtkWidget*,c_Picture*);
```

```
// Define parametros da janela
void
fr_set_utils(c_Picture* , guint, guint, guint);


//---------------------classe c_Pair-----------------------//

struct c_Pair
{
      c_Picture   Picture1;   // Ponteiro para a primeira imagem
      c_Picture   Picture2;   // Ponteiro para a sugunda imagem
      gfloat            focaldistance;  // Distancia focal do
sistema de visao
      gfloat            t[3];       // Vetor deslocamento entre as
imagens
      gfloat            fiw;        // Parametro de calibracao do
sistema de visao
      gfloat            fih;        // Parametro de calibracao do
sistema de visao
      gfloat            e;          // Distancia entre o foco do
espelho e o foco da camera

      };
typedef struct c_Pair c_Pair;

// Imprime as informacoes da estrutura c_Pair na tela
void
fr_print_pair_data(c_Pair *pair);

// Calcula distancias em pares de imagens planas
gfloat
fr_triangularization(gfloat, gfloat);

//Carregar um par de imagens no programa
void
fr_load_pair(c_Pair*);

// Traca linhas epipolares em imagens planas paralelas
void
fr_trace_epipolar(c_Pair*);

// Calcula a funcao `F+`
gfloat
fr_F(vet3x1*);

// Calcula a funcao `F-`
gfloat
fr_F_Menos(vet3x1 *x);

// Desenha na tela os epipolos das imagens ominidirecinais
void
fr_trace_epipoles(c_Pair*);

// Calcula e desenha na tela curvas epipolares
mat3x3
fr_epipolar_conic_xpm(c_Pair*, gfloat x, gfloat y);

// Calcula curvas epipolares
mat3x3
```

```
fr_epipolar_conic(c_Pair*, gfloat x, gfloat y);


//Desenha na tela curvas epipolares
void
fr_trace_epipolar_conic(c_Pair*, mat3x3*, guint);

// Seleciona o tipo de figura epipolar
int
fr_select_shape(mat3x3*);

// Calcula distancias em pares de imagens ominidirecionais
vet3x1
fr_get_distance(c_Pair *pair, int u1, int v1, int u2, int v2);


// -------Variaveis definidas globalmente---------//

//--------GLOBALS - BEGIN --------//

      c_Pair            Pair;
      gfloat            prec;


//--------GLOBALS - END ---------//

#endif
```

CORRELACAO.H

```
/*************************************************************

funcoes de correlacoes

*************************************************************/


#ifndef _Corr
#define _Corr

#include "Classes.h"
#include "libmatrix.h"
#include "libffn.h"
#include "s.h"
#include "Maps.h"

// função de aplicação dos filtros e metodos de correlacao para cada
// par de pontos de interesse
float
fr_method_applicator(int function, c_Picture *picture1, c_Picture
*picture2, int x1, int y1, int x2, int y2);

// funcao principal de correlacao de busca em janelas sobre as
// epipolares
void
fr_correlation_window_on_ep(c_Pair *pair, int variation);

// funcao de pre-tratamento que retira o centro, que nao possui
// informacoes de interesse, inserindo valores 0 nos pontos
void
tira_centro(c_Picture *picture ,int size, int color);

// funcao de pre-tratamento que retira as bordas que nao possui
// informacoes de interesse
void
tira_borda(c_Picture *picture ,int size);

// faz a interface da estrutura do programa com a funcao que carrega
// o arquivo externo no string 'in' e obtem as demais informacoes
void
fr_load_pmg(c_Picture *picture);

// faz o contrario da outra funcao
void
fr_record_pmg(c_Picture *picture, gchar *name);

// funcao principal de busca de corners, faz a inteface da estrutura
// do programa com as funcoes de baixo nivel que buscam os corners
void
fr_find_corners(c_Picture *picture, int draw);

// devolve a intensidade media da janela de tamanho especificado
// centrada no ponto especificado
```

```
float
fr_Average_IntensityXY(c_Picture *picture, int x, int y, int xsize,
int ysize);
// devolva a intensidade, mas passaando-se a posicao de memoria ao
// inves das coordenadas
float
fr_Average_IntensityP(c_Picture *picture, uchar *p, int xsize, int
ysize);

// Calcula o fator de diferencas quadradas entre as janelas
// especificadas
float
fr_MMQ( c_Picture *picture1, int x1, int y1,
      c_Picture *picture2, int x2, int y2,
      int x_window, int y_window);

// Calcula o fator de diferencas quadradas normalizadas pela
// intensidade media das janelas especificadas
float
fr_MMQ_Normal( c_Picture *picture1, int x1, int y1,
            c_Picture *picture2, int x2, int y2,
            int x_window, int y_window, int parametro);

// devolve a porcentagem de exatidao dos pixels nas janelas
// especificads pelos pixels em cada imagem
float
fr_exact_match(c_Picture *picture1, int x1, int y1,
            c_Picture *picture2, int x2, int y2,
            int x_window, int y_window, float param);

// retorna 1 ou 0 indicando se esta no lado correto da epipolar
int
fr_rigth_sideX(int x1, int y1, int x2, int y2);

// retorna 1 ou 0 indicando se esta na região perto do eixo y
int
fr_out_Y(int x1,int paran);

// retorna 1 ou 0 indicando se esta na região perto do eixo x
int
fr_in_X(int x1,int paran);

// funcao de correlacao entre duas listas de corners das imagens
void
fr_correlation_corners(c_Picture *picture1, c_Picture *picture2);

// funcao de correlacao entre duas listas de corners das imagens,
// buscando na região da epipolar
void
fr_correlation_corners_on_ep(c_Pair *pair);

// retorna a distancia que estes pontos estao
float
fr_distance(int x1, int y1, int x2, int y2);

// funcao de correlacao de janelas por soma de diferencas ao
// quadrado sem epipolar
void
```

```
fr_correlation_SSD(c_Picture *picture1, c_Picture *picture2, int
tresh, int janela_busca, int janela_cor);


// funcao de correlacao por janelas usando intensidade media sem
// epipolar
void
fr_correlation_Linear(c_Picture *picture1, c_Picture *picture2);

// funcao de correlacao por janelas usando diferenca de intensidade
// sem epipolar
void
fr_correlation_Linear2(c_Picture *picture1, c_Picture *picture2);

// funcao de tracado de epipolar em fotos pgm
void
fr_trace_epipolar_conic_pgm(c_Pair *pair, mat3x3 *A, guint step);

// funcao de correlacao por janelas usando diferenças ao quadrado
void
correl_vetor(c_Pair *pair, int thresh, int janela_busca,int
      janela_cor);

// funcao que cria lista de pontos de uma janela
void
cria_vetor_janela(c_Picture *picture,int x, int y, CORNER_LIST
janela_list,int sizex,int sizey);

// funcao de correlacao por janelas usando diferenças ao quadrado
void
correl_vetor_window(c_Pair *pair, int x, int y, int thresh, int
janela_busca, int janela_cor);

#endif
```

CALLBACKS.C

```
/*************************************************************

funcoes que conectam os botoes aos comandos

*************************************************************/


#ifdef HAVE_CONFIG_H
#  include <config.h>
#endif

#include <gnome.h>

#include "callbacks.h"
#include "interface.h"
#include "support.h"


#include "s.h"
#include "Correlacao.h"

#include "Classes.h"
#include "Paran.h"

void
on_window1_show                        (GtkWidget       *widget,
                                        gpointer         user_data)
{

/* definicoes necessaries para o funcionamento da correlacoa e mapa
de distancia */

      Pair.Picture1.name = "0";
      Pair.Picture2.name = "0";

      Pair.Picture1.bt = 20;
      Pair.Picture1.drawing_mode=0;
      Pair.Picture1.max_no_corners=1850;

      Pair.Picture2.bt = 20;
      Pair.Picture2.drawing_mode=0;
      Pair.Picture2.max_no_corners=1850;

}


void
on_bQuit_clicked                       (GtkButton       *button,
                                        gpointer         user_data)
{
      exit(0);
}
```

```
void
// gera janela para obter o nome dos arquivos
on_bLoadImage1_clicked                  (GtkButton       *button,
                                         gpointer         user_data)
{
      GtkWidget *window;

      window =  create_file_selection ();
      gtk_widget_show (window);

}


void
on_bLoadImage2_clicked                  (GtkButton       *button,
                                         gpointer         user_data)
{
      GtkWidget *window;

      window =  create_file_selection ();
      gtk_widget_show (window);
}

// carega as imagens XPM para serem vistas
void
on_bShowImage_clicked                   (GtkButton       *button,
                                         gpointer         user_data)
{
      fr_load_pair(&Pair);
}

// gera janela para obter os parametros
void
on_bGetParam_clicked                    (GtkButton       *button,
                                         gpointer         user_data)
{
      GtkWidget *window;

      window =  fr_create_wEnterParam ();
      gtk_widget_show (window);
}

// trabcado da epipolar plana
void
on_bPEpipolar_clicked                   (GtkButton       *button,
                                         gpointer         user_data)
{
  fr_trace_epipolar(&Pair);
}

// executa a obtencao de distancia para imagens planas e XPM
void
on_bPTriang_clicked                     (GtkButton       *button,
                                         gpointer         user_data)
{
  gfloat  distance;
  gfloat  xl, xr;

  xl = (gfloat)Pair.Picture1.cx - (gfloat)Pair.Picture1.triangx;
```

```
   xr = (gfloat)Pair.Picture2.cx - (gfloat)Pair.Picture2.triangx;


   distance = fr_triangularization(xl, xr);
   g_print("Distance -> %f \n", distance);
}

// tracado dos epipolos
void
on_bEpipoles_clicked                    (GtkButton       *button,
                                         gpointer         user_data)
{
      fr_trace_epipoles(&Pair);
}


// tracado das linhas epipolares conicas
void
on_bEpLine_clicked                      (GtkButton       *button,
                                         gpointer         user_data)
{
      gfloat x, y;

      x = (gfloat)Pair.Picture1.triangx - (gfloat)Pair.Picture1.cx;
      y = (gfloat)Pair.Picture1.triangy - (gfloat)Pair.Picture1.cy;
      fr_epipolar_conic_xpm(&Pair, x, y);
}


void
on_bDistance_clicked                    (GtkButton       *button,
                                         gpointer         user_data)
{
      //fr_get_distance(&Pair, Pair.Picture1.triangx,
Pair.Picture1.triangy,
                  //                Pair.Picture2.triangx,
Pair.Picture2.triangy);

      fr_get_distance(&Pair, 334, 90, 356, 94);
}

// insercao do nome da imagem pega na janela à estrutura
void
on_ok_bFileSelec_clicked                (GtkButton       *button,
                                         GtkFileSelection
*file_selection)
{
  gchar *filename1;
  gchar *filename2;

if (Pair.Picture1.name == "0")
  {
  g_print ("on_ok_Picture1\n");
  filename1 = gtk_file_selection_get_filename (GTK_FILE_SELECTION
(file_selection));
  Pair.Picture1.name = filename1;

  gtk_widget_hide(file_selection);
  }
```

```
else if (Pair.Picture2.name == "0")
  {
  g_print ("on_ok_Picture2\n");
  filename2 = gtk_file_selection_get_filename (GTK_FILE_SELECTION
(file_selection));
  Pair.Picture2.name = filename2;
  gtk_widget_hide(file_selection);
  }
  g_print (Pair.Picture1.name,"\n");
  g_print (Pair.Picture2.name,"\n");

}


void
on_cancel_bFileSelec_clicked           (GtkButton       *button,
                                        GtkFileSelection
*file_selection)
{
        gtk_widget_hide(file_selection);
}



// funcao principal de geracao de mapas de distancia e correlacao
void
on_bSusan_clicked                              (GtkButton
*button,
                                       gpointer         user_data)
{

// definicoes da figura
      Pair.Picture1.cx = 320;
      Pair.Picture1.cy = 240;
      Pair.Picture1.radius = 235;
      Pair.Picture2.cx = 320;
      Pair.Picture2.cy = 240;
      Pair.Picture2.radius = 235;

//funcao para carregar as imagens em strings
      fr_load_pmg(&Pair.Picture1);
      //Pair.Picture2.name = "/home/ffn/p/p5/bin/f4_15.pgm";  /*
como fixar um nome de arquivo */
      fr_load_pmg(&Pair.Picture2);


      //pre tatamento da figura
      tira_centro(&Pair.Picture1,45, 0);
      tira_borda(&Pair.Picture1,215);

      tira_centro(&Pair.Picture2,45, 0);
      tira_borda(&Pair.Picture2,215);

//funções de correlações implementadas
      //fr_correlation_corners_on_ep(&Pair);
      fr_correlation_window_on_ep(&Pair, 3);  //funcao usada
      //fr_correlation_SSD(&Pair.Picture1, &Pair.Picture2, 5000, 15,
5);
```

```
      //fr_correlation_Linear(&Pair.Picture1, &Pair.Picture2);
      //fr_correlation_Linear2(&Pair.Picture1, &Pair.Picture2);
      //correl_vetor(&Pair, 8000, 15, 5);

//gravar a imagem em arquivo
      fr_record_pmg(&Pair.Picture1, "Corner1.pgm");
      fr_record_pmg(&Pair.Picture2, "Corner2.pgm");


}
```

FUNCTIONS.H

```
/*************************************************************

funcoes adicionais da parte gráfica

*************************************************************/

#include <gnome.h>
#include "Classes.h"

#ifndef _Functions
#define _Functions


// Desenha pontos em imagens xpm
void
fr_draw_point(c_Picture *picture, guint x0, guint y0);

// Desenha linhas em imagens xpm
void
fr_draw_line(c_Picture *picture, guint x0, guint y0, guint x1, guint
y1);

// Desenha marcas em imagens xpm
void
fr_draw_mark(c_Picture* ,guint, guint);

// Desenha circulos em imagens xpm
void
fr_draw_circle(GtkWidget *widget, c_Picture *picture, guint x0,
guint y0, guint radius, guint full);

// Cria a janela de entrada de parametros
GtkWidget*
fr_create_wEnterParam (void);

// Funcoes de entrada dos parametros no programa
void
fr_entry_t0(GtkWidget *widget, GtkWidget *entry);
void
fr_entry_t1(GtkWidget *widget, GtkWidget *entry);
void
fr_entry_t2(GtkWidget *widget, GtkWidget *entry);
void
fr_entry_f(GtkWidget *widget, GtkWidget *entry);
void
fr_entry_fiw(GtkWidget *widget, GtkWidget *entry);
void
fr_entry_fih(GtkWidget *widget, GtkWidget *entry);

//Fecha a janela de entrada de parametros
void
on_bEnter_clicked (GtkButton *button, GtkWidget *window);

#endif
```

MAPS.H

```
/*************************************************************

funcoes de geracao de mapas de distancias e representacoes
graficas na imagerm (pontos, quadrados, linhas)

*************************************************************/

#ifndef _Maps
#define _Maps

#include "Classes.h"
#include "libmatrix.h"
#include "libffn.h"
#include "s.h"
#include "Correlacao.h"

// Desenha pontos
void
fr_mark_pixel(c_Picture *picture, int x, int y, int color);

// Desenha marcas
void
fr_mark_point(c_Picture *picture, int x, int y, int color);

// Desenha centro da imagem
void
fr_mark_central_point(c_Picture *picture, int x, int y, int color);

// Desenha linhas dado 2 pontos
void
fr_draw_line_pgm(uchar *in, int x_size, int y_size, int x1, int
y1,int x2, int y2, int);

// desenha linha dos pontos maximos aos minimos no mapa de ditancia,
// dado uma lista de pontos
void
fr_points_max_min(uchar *in, int x_size, int y_size, CORNER_LIST
corner_list, int color);

// Desenha a linha media dos pontos em CORNER_LIST (função de
//  preparação)
void
fr_draw_line_MMQ(uchar *in, int x_size, int y_size, CORNER_LIST, int
color);

// usando mmq desnha reta maedia dado uma lista de pontos e o
// numero da sequencia
void
fr_points_line_MMQ(uchar *in, int x_size, int y_size, CORNER_LIST
corner_list, int color, int n_seq);
```

```
// Desenha quadrados
void
fr_draw_square(c_Picture *picture, int x, int y, int sizey, int
sizex, int cor);

// Marca todos os matches na imagem
void
fr_mark_map(c_Pair *pair, CORNER_LIST match_list);

// marca um ponto dado sua posicao de memoria
void
fr_mark_point_in(uchar *in, int x_size, int x, int y, int color);

// Desenha a lista de pontos correlacionados em imagens xpm
void
fr_mark_map_xpm(c_Pair *pair, CORNER_LIST match_list);

// Constrói os mapas de Distancia - função principal
void
fr_upper_view(c_Pair *pair, CORNER_LIST dist_list);

// dedenha quadrados nos pontos da lista
void
fr_square_map(c_Pair *pair, CORNER_LIST match_list);

// desenha linhas no amapa de diatancia indicando a trajetoria dos
// matches
void
fr_line_map(c_Pair *pair, CORNER_LIST match_list);

// zera lista
void
fr_clean_list(CORNER_LIST *list);

// agrupa os corners dadios uma distancia minima entre eles e
// chama fincoes para desenhar linhas
void
fr_group_corners(c_Pair *pair, CORNER_LIST match_list, int paran);

// conecta todos os pontos da lista
void
fr_draw_conect_line(uchar *in, int x_size, int y_size, CORNER_LIST
corner_list, int color);

// desenha alinha media entre todos os pontos da lista
void
fr_draw_average_line(uchar *in, int x_size, int y_size, CORNER_LIST
corner_list, int color, int mode);

//desenha linha entre dois pontos
void
desenha_linha(c_Picture *picture, int x1, int y1, int x2, int y2,
int cor);

// desenha linhas retas em x ou y
void
desenha_linha_reta(uchar *in, int x_size, int y_size, int x1, int
x2, int y, int dir, int color);
```

```
/* funcao principal de agrupar pontos dados dois parâmetros de
busca. O primeiro tira todos os pontos que não possuem pontos nesta
distancia, a outra é a distancia mínima que alimenta a função que
gera a lista de pontos pertos recursivamente */
void
gera_lista(CORNER_LIST corner_list, int thresh1, int thresh2);

// funcao recursive que procura pontos e cria sequencias, inserindo
// no valor info o numero da sequencia
void
pontos_perto(CORNER_LIST corner_list, int x_ref, int y_ref, int
n_seq, int thresh);

// marca com valor 50 no info os pontos que possuem pontos perto a
// distancia passada como parametro
void
kill_alone(CORNER_LIST corner_list, int thresh);

// conecta os pontos da lista
void
fr_connect_point(uchar *in, int x_size, int y_size, CORNER_LIST
corner_list, int color);

#endif
```

LIBMATRIX.H

```
/*************************************************************
Definições de estruturas matriciais e vetores necessários ao
programa e funções relacionadas
*************************************************************/


#ifdef HAVE_CONFIG_H
#  include <config.h>
#endif

#ifndef LIB_MAT
#define LIB_MAT

#include <gdk/gdk.h>
#include <math.h>

// vetor para as distancia e deslocamentos
struct vet3x1
{
      gdouble     a1;
      gdouble     a2;
      gdouble     a3;

};
typedef struct vet3x1 vet3x1;

// imprime na tela os valores do vetor
void
fr_print_vet3x1(vet3x1*);

//copia vetor
void
fr_copy_vet3x1(vet3x1*, vet3x1* );

// modulo do vetor
gfloat
fr_mod_vet3x1(vet3x1*);

// multiplica o vetor por uma constante
void
fr_vet3x1_k(vet3x1*, gfloat k);

// somas vetores
void
fr_sum_vet3x1(vet3x1*,vet3x1*);


void
fr_simplify_vet3x1(vet3x1*);

// matriz 3x3 utilizadas para as epipolares
struct mat3x3
{
      gdouble     a11;
```

```
        gdouble     a12;
        gdouble     a13;

        gdouble     a21;
        gdouble     a22;
        gdouble     a23;

        gdouble     a31;
        gdouble     a32;
        gdouble     a33;
};
typedef struct mat3x3 mat3x3;

// imprime na teal a matriz
void
fr_print_mat3x3(mat3x3*);

// zera a matriz
void
fr_zera_mat3x3(mat3x3*);

// copia a matriz
void
fr_copy_mat3x3(mat3x3 *, mat3x3* ); //source - > dest.

// acha o determinante da matriz
gfloat
fr_det_mat3x3(mat3x3 *mat);

void
fr_tr_mat3x3(mat3x3 *mat);

// multiplica por constante
void
fr_mat3x3_k(mat3x3*, gfloat k);

//multiplica por vetor
void
fr_mat3x3_vet3x1(mat3x3 *mat, vet3x1 *vet, vet3x1* vet1);

//multiplica por matriz
void
fr_mat3x3_mat3x3(mat3x3 *mat1, mat3x3 *mat2, mat3x3 *mat3);

void
fr_cofat_mat3x3(mat3x3 *mat);

// acha ainversa
void
fr_inv_mat3x3(mat3x3 *mat);


#endif
```

LIBFFN.H

```
#ifdef HAVE_CONFIG_H
#  include <config.h>
#endif

#ifndef LIB_FFN
#define LIB_FFN


#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>
#include <string.h>

#include <gnome.h>
#include <math.h>

struct c_Point
{
      guint       x;
      guint       y;
};
typedef struct c_Point c_Point;

GtkWidget*
create_dialog (char *text);

void
fr_aviso(char *texto);

// verifica se esta na regiao
int
inrange(float a, float b, float e);

//obtema distancia entre dois pontos
float
distance(c_Point a, c_Point b);

#endif
```

s.h

```
/*************************************************************

funcoes de extracao de caracteristicas e interface entre
arquivo externos de imagem e strings para tratamento

*************************************************************/

#ifndef LIB_S
#define LIB_S

/* ********** Optional settings */

#ifndef PPC
typedef int        TOTAL_TYPE; /* this is faster for "int" but
should be "float" for large d masks */
#else
typedef float      TOTAL_TYPE; /* for my PowerPC accelerator only */
#endif

#define SEVEN_SUPP           /* size for non-max corner suppression;
SEVEN_SUPP or FIVE_SUPP */
#define MAX_CORNERS   15000  /* max corners per frame */

#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <sys/file.h>    /* may want to remove this line */
#include <malloc.h>      /* may want to remove this line */
#include <sys/resource.h>

#define  exit_error(IFB,IFC) { fprintf(stderr,IFB,IFC); exit(0); }
#define  FTOI(a) ( (a) < 0 ? ((int)(a-0.5)) : ((int)(a+0.5)) )


typedef  unsigned char uchar;
typedef  struct {int x,y,info, dx, dy, I;} CORNER_LIST[MAX_CORNERS];
// definicao da estrutura das listas usads no programs

// acha inteiro no arquivo
int
getint(FILE *fd);

// grava na string a imagem a partir do arquivo
void
get_image(  char filename[200], unsigned char  **in,  int *x_size,
            int *y_size);

//grava no arquivo
void
put_image(  char filename[100], char *in, int x_size, int y_size);

/* gera um LUT(look up table) com a formula para achar a diferenca
de intensidades entre os pontos da imgem e assim definir o valor
USAN */
```

```
void
setup_brightness_lut(   uchar **bp, int thresh, int form);

// desenha os corner dado uma lista
void
corner_draw(uchar *in,  CORNER_LIST corner_list,
            int x_size, int drawing_mode);

// funcao que obtem os corner e grava na corner_list
void
susan_corners( uchar *in, int *r, uchar *bp,
                  int max_no, CORNER_LIST corner_list,
                  int x_size, int y_size);


#endif
```