FRANCISCO ATAIDE MICHELOTO

nota final
8.8 (oito e oito)

PROJETO DE SISTEMA DE VISÃO ESTÉREO OMNIDIRECIONAL COM ESPELHO DUPLO DE PERFIL HIPERBÓLICO

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

São Paulo
2004

FRANCISCO ATAIDE MICHELOTO

PROJETO DE SISTEMA DE VISÃO ESTÉREO OMNIDIRECIONAL COM ESPELHO DUPLO DE PERFIL HIPERBÓLICO

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

Área de concentração:
Engenharia Mecatrônica

Orientador: Prof. Dr. Eduardo L L. Cabral

São Paulo
2004

RESUMO

O objetivo da presente dissertação é trabalhar com a etapa de reconstrução tridimensional de um sistema de visão estéreo omnidirecional. Neste projeto, o sistema de visão omnidirecional utilizado é um espelho duplo de perfil hiperbólico e uma câmera com o seu eixo óptico alinhado com o centro do espelho. Este tipo de sistema dispensa o movimento da câmera para se obter uma imagem do espaço à sua volta, além de obter tal imagem instantaneamente. No entanto, requer um processamento especial da imagem, já que ela se encontra mapeada em coordenadas polares. O perfil hiperbólico tem a propriedade de ter um centro único de projeção, o que garante que um ponto no espaço terá um único correspondente na imagem adquirida. Além disso, tem a vantagem sobre outros tipos de perfil o fato de produzir imagens livre de distorções, como acontece, por exemplo, com o perfil esférico. O fato de o espelho ser duplo nos proporciona em uma única imagem duas visões diferentes de um mesmo ambiente, uma na seção interna e outra na seção externa do espelho. Com estas imagens é possível aplicar técnicas de visão computacional estéreo para se reconstruir o ambiente a partir destas imagens. No projeto do sistema de visão estéreo omnidirecional com espelho duplo de perfil hiperbólico há várias etapas a serem realizadas, como o projeto do espelho, análise dos erros de reconstrução estéreo, etc. O escopo deste trabalho abrange apenas o processo de recuperação tridimensional, que particularmente engloba o processo de transformação da imagem omnidirecional em perspectiva; correspondência entre os pontos; processo de triangulação e cálculo da menor distância.

ABSTRACT

The aim of this dissertation is to work with the tridimensional reconstruction phase of an omnidirectional stereo vision system. In this project, the omnidirectional vision systems used is a double lobed mirror with hyperbolic profile and a camera with its optical axis aligned with the mirror's center. This kind of system dismisses camera movement in order to capture of the surrounding space and also can do it instantly. However, it requires a special processing of the image, as it is in polar coordinates. The hyperbolic profile gives a unique projection center, which garantees that a point in space will have only one correspondent in the image acquired. Besides, it has the advantage of producing images free from distortions, as we observe, for example, in spheric profiles. Being the mirror double lobed provides us in one single image two different visions of a same scene, one in the inner section and another in the outer section. With these images, it is possible to apply computer stereo vision techniques to reconstruct the environment from the images obtained. In the project of the stereo vision system with double lobed mirror there are many phases to be accomplished, as the mirror project, error analysis in stereo reconstruction, etc. This project's scope is to explore the tridimensional reconstruction, which particularly involves the transformation of the omnidirectional image in perspective; correspondence between points; triangulation process and calculate the shortest distance.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SÍMBOLOS

ε	Distância focal da lente
$\rho_{I,\max}$	Raio máximo da seção interna do espelho
$\rho_{I,\min}$	Raio mínimo da seção interna do espelho
$\rho_{E,\max}$	Raio máximo da seção externa do espelho
$\rho_{E,\min}$	Raio mínimo da seção externa do espelho
$\rho_{imagem,\max}$	Raio máximo da imagem
$\gamma_{\mathrm{Im}ax}$	Ângulo máximo de visão da seção interna
$\gamma_{\mathrm{Im}in}$	Ângulo mínimo de visão da seção interna
$\gamma_{E\max}$	Ângulo máximo de visão da seção externa
$\gamma_{E\min}$	Ângulo mínimo de visão da seção externa
r_P	Coordenada *r* do ponto P no espaço
z_P	Coordenada *z* do ponto P no espaço
ϕ_I	Ângulo de elevação interno
ϕ_E	Ângulo de elevação externo
ρ_I	Raio dos pontos na imagem interna
ρ_E	Raio dos pontos na imagem externa
λ	Ângulo que corresponde à rotação da seção externa

f	Distância focal – distância entre o plano da imagem formada e centro da lente
F_0	Foco das hipérboles
F_I	Foco da hipérbole interna
F_E	Foco da hipérbole externa
M_I	Projeção do ponto P na seção interna do espelho
M_E	Projeção do ponto P na seção interna do espelho
c_I	Parâmetro da hipérbole interna
a_I	Parâmetro da hipérbole interna
c_E	Parâmetro da hipérbole externa
a_E	Parâmetro da hipérbole externa
z_{FE}	Coordenada z do foco F_E
z_{FI}	Coordenada z do foco F_I
r_{FE}	Coordenada r do foco F_E
f'	Distância focal da seção externa rotacionada
ρ'_E	Raio dos pontos na imagem da seção externa rotacionada
ϕ'_E	Ângulo de elevação da seção externa rotacionada
$P_{\max}$	Tamanho do raio da imagem em pixels
$r_{\max}$	Raio máximo do CCD
$\nabla \vec{f}$	Vetor gradiente

Capítulo 1

INTRODUÇÃO

Sistemas de visão omnidirecional produzem imagens de 360° do ambiente, podendo ser utilizados, entre outras aplicações, em navegação, vigilância remota e em robôs. Neste projeto, o sistema de visão omnidirecional utilizado é um espelho duplo de perfil hiperbólico e uma câmera com o seu eixo óptico alinhado com o centro do espelho. Este tipo de sistema dispensa o movimento da câmera para se obter uma imagem do espaço à sua volta, além de obter tal imagem instantaneamente. O projeto de visão estéreo omnidirecional usando um espelho duplo de perfil hiperbólico tem como finalidade obter um sistema compacto e potente para, a partir de uma única imagem perspectiva, recuperá-la tridimensionalmente. A vantagem de se ter um sensor que forneça dados sobre todo o seu redor pode ser útil em muitos casos e especialmente robôs móveis, uma vez que o número de sensores utilizados para a sua localização pode ser reduzido para apenas um.

O desenvolvimento do projeto de um sensor de distâncias baseado em visão estéreo omnidirecional pode ser dividido nas seguintes etapas:

- Projeto e fabricação do espelho duplo de perfil hiperbólico;
- Análise do erro nas medidas de distância;
- Análise do erro de reconstrução estéreo;
- Configuração do sistema câmera-espelho;
- Calibração do sistema câmera-espelho;
- Desenvolvimento do processo de recuperação tridimensional;
- Implementação do sistema em hardware eletrônico dedicado;
- Testes do sistema;
- Resultados e conclusões;

Neste trabalho, apenas o processo de recuperação tridimensional será explorado. Logo, as demais etapas, como o projeto de fabricação do espelho, estão fora do escopo e não serão aprofundadas aqui, sendo que algumas serão brevemente comentadas. Com o

processo de recuperação tridimensional é possível reconstruir um ambiente a partir de duas imagens panorâmicas obtidas pelo sensor através de cálculos que utilizam como base as propriedades do espelho. O objetivo é obter um sistema de visão estéreo compacto e capaz de realizar a recuperação tridimensional com baixo custo computacional, para que possa ser usado em tempo real, e hardware eletrônico de baixo custo. O processo de recuperação em si engloba o processo transformação da imagem omnidirecional em perspectiva (retificação da imagem); correspondência entre pontos (dado um ponto em uma imagem, deve-se encontrar o ponto equivalente na outra); processo de triangulação (depois de correspondido um ponto, determinar a sua posição no espaço). Foram feitas simulações com um ambiente simulado construído no software PovRay v3.5 e em MatLab v6. Também foi feita uma rotina que calcula o ponto mais próximo da câmera.

Capítulo 2

VISÃO OMNIDIRECIONAL

Sistemas de visão omnidirecional permitem a aquisição de imagens com campo de visão de 360°. Cabral (2003) e Junior (2002) apresentam algumas formas conhecidas de se obter uma visão omnidirecional, além de citarem autores de estudos mais aprofundados. Entre esses métodos podemos citar: utilização de lente do tipo "olho de peixe" montado em uma câmera fixa, combinação em uma única imagem de várias imagens obtidas por uma câmera que gira com velocidade angular constante ao redor de um eixo vertical fixo, como mostra a Fig. 2.1, etc. No entanto, estas alternativas têm problemas de resolução ou de tempo de aquisição da imagem de 360°, como mostram os supracitados autores. O sistema apresentado na Fig. 2.1 não obtém a imagem completa em tempo real, e por isso é inadequado para situações em que o ambiente é dinâmico.

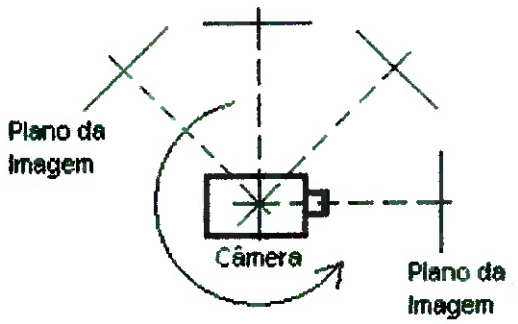

Figura 2.1 - Câmera giratória para formação de imagem omnidirecional

Um meio de se obter visão omnidirecional, porém em tempo real, é a utilização de meios catadióptricos. Um meio catadióptrico é composto por um meio dióptrico (ou seja, um meio no qual a luz atravessa um meio e continua sua propagação em outro, como, por exemplo, a lente de uma câmera) e um catóptrico (ou seja, um meio onde a luz incide e é refletida para o mesmo meio de propagação, como, por exemplo, um espelho). Usando-se uma câmera e um espelho convexo, sendo o espelho alinhado com o eixo óptico da câmera, obtém-se um sistema capaz de obter uma imagem

omnirecional. Dependendo da forma do espelho, podemos ter imagens com maior ou menor resolução e campo de visão.

Cabral (2003) e Junior (2002) mostram que diversos autores propuseram vários perfis de espelho para a construção do sistema. No entanto, o perfil hiperbólico, como todo raio incidente no espelho com esse perfil é refletido após passar pelo foco, é interessante por ter a vantagem de possuir um centro único de projeção, propriedade que livra a imagem obtida de distorções, o que facilita os cálculos de recuperação tridimensional por triangulação, agilizando o processo, além de permitir gerar imagens com resolução igual e mesmo campo de visão. Na Fig. 2.2, tem-se uma ilustração disso. A obtenção da visão estéreo neste tipo de sistema é feita através de um espelho duplo, como é apresentado no capítulo 4.

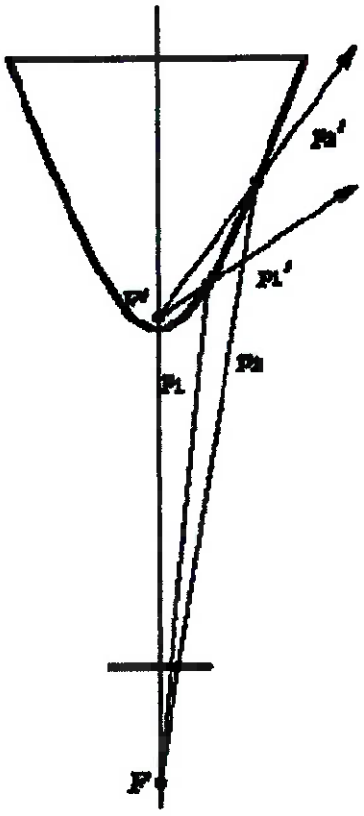

Figura 2.2 – Espelho hiperbólico

Neste projeto, foram feitas simulações utilizando a equação de um espelho duplo de perfil hiperbólico e um software de ray-tracing, Pov-Ray. Utilizando este software, construímos um ambiente tridimensional e, usando a equação do espelho, obtivemos uma imagem de 360° do ambiente construído. O código fonte do ambiente é apresentado no Anexo A. Na Fig. 2.3 é mostrada a imagem obtida.

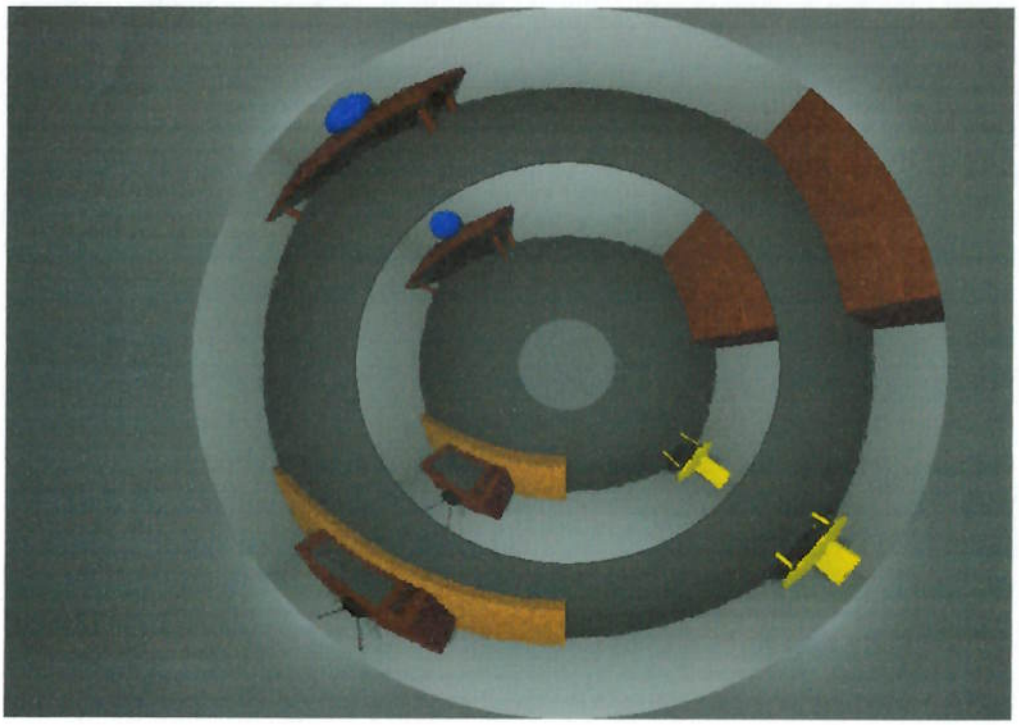

Figura 2.3 - Imagem construída no Pov-Ray

O processo de retificação da imagem, correspondência entre pontos e processo de triangulação serão discutidos posteriormente, no capítulo 4.

Capítulo 3

VISÃO COMPUTACIONAL

3.1 - Formação de imagens

Em processamento de imagens, um dos principais conceitos é o de projeção perspectiva. Esse tipo de projeção é importante, pois representa aproximadamente a maneira como uma imagem é formada observando um mundo em três dimensões.

A projeção perspectiva projeta pontos de uma imagem 3D em um plano. Isso quer dizer que o tamanho dos objetos é reduzido à medida que a distância ao plano de projeção aumenta. A Fig.3.1 mostra o processo de formação da imagem. Definimos o sistema de coordenadas de uma câmera (x, y, z) de tal forma que o plano da imagem coincida com o plano $z = 0$, e o eixo óptico (estabelecido pelo centro da lente, ou ponto de fuga) coincidente com o eixo z. Dessa forma, o ponto (0, 0, ε) representa o centro da lente e a distância ε é a distância focal da lente. No caso do sistema de visão estéreo proposto, é conveniente assumir que o sistema de coordenadas da câmera (x, y, z) coincide com o sistema de coordenada usada para representar qualquer ponto no espaço (X, Y, Z).

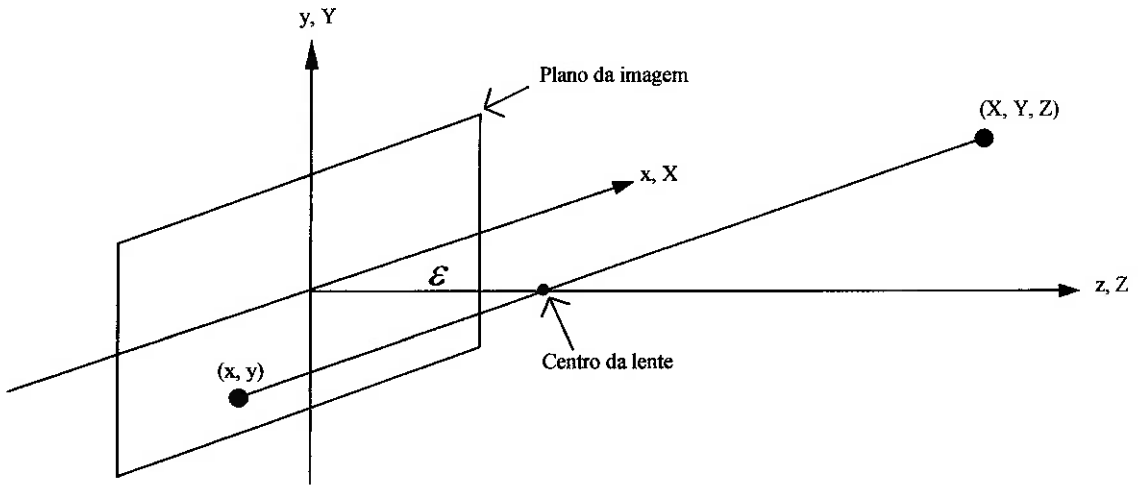

Figura 3.1 - Modelo básico de formação de imagens

Para obter um equacionamento que nos permita associar um ponto P genérico no espaço a coordenadas no plano da imagem, podemos usar simplesmente semelhança de triângulos. Desta forma:

$$\frac{x}{\varepsilon} = -\frac{X}{Z-\varepsilon} \tag{3.1}$$

$$\frac{y}{\varepsilon} = -\frac{Y}{Z-\varepsilon} \tag{3.2}$$

onde de o sinal negativo significa que a imagem é invertida no plano da imagem, como pode ser verificado na Fig.2. Isolando x e y em (3.1) e (3.2), temos:

$$x = \frac{\varepsilon X}{\varepsilon - Z} \tag{3.3}$$

$$y = \frac{\varepsilon Y}{\varepsilon - Z} \tag{3.4}$$

que nos dá, tendo um ponto em (X, Y, Z), o ponto no plano da imagem.

Podemos expressar as equações acima usando formas matriciais, mais convenientes. Neste caso, vamos primeiro definir coordenadas homogêneas. Coordenadas homogêneas de um ponto cartesiano (X, Y, Z) são definidas como (kX, kY, kZ, k), onde k é uma constante arbitrária diferente de zero. Dividindo as três primeiras coordenadas pela quarta temos as coordenadas cartesianas originais. Uma característica interessante das coordenadas homogêneas é que elas podem suportar elegantemente a representação computacional de pontos infinitamente distantes. Por exemplo, as coordenadas homogêneas (1, 0, 0, 0) representam um ponto de coordenada X infinita. Podemos expressar um ponto cartesiano como o vetor:

$$w = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{3.5}$$

e o equivalente homogêneo como:

$$w_h = \begin{bmatrix} kX \\ kY \\ kZ \\ k \end{bmatrix} \tag{3.6}$$

Definindo a matriz de transformação perspectiva:

$$P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -\frac{1}{\varepsilon} & 1 \end{bmatrix} \tag{3.7}$$

onde ε é a distância focal da lente, o produto Pw_h resulta no vetor c_h, que representa as coordenadas do ponto no plano da imagem em coordenadas homogêneas.

$$c_h = Pw_h = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -\frac{1}{\varepsilon} & 1 \end{bmatrix} \begin{bmatrix} kX \\ kY \\ kZ \\ k \end{bmatrix} = \begin{bmatrix} kX \\ kY \\ kZ \\ -\frac{kZ}{\varepsilon} + k \end{bmatrix} \tag{3.8}$$

Dividindo os três primeiros componentes de (3.8) pelo quarto, obtemos as coordenadas do ponto no plano da imagem. Assim:

$$c = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \frac{\varepsilon X}{\varepsilon - Z} \\ \frac{\varepsilon Y}{\varepsilon - Z} \\ \frac{\varepsilon Z}{\varepsilon - Z} \end{bmatrix} \tag{3.9}$$

Os dois primeiros componentes de (3.9) são as coordenadas (x, y) do ponto no plano da imagem de um ponto 3D projetado, como mostrado nas equações (3.3) e (3.4). O terceiro componente não nos interessa, do ponto de vista do modelo mostrado na Fig.3.1. No entanto, ele é usado como uma variável livre na transformação perspectiva inversa.

A inversa da transformação perspectiva nos dá, a partir da imagem, as coordenadas do ponto no espaço. Assim:

$$w_h = P^{-1} c_h \tag{3.10}$$

onde P^{-1} é a matriz inversa de P, ou seja:

$$P^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -\frac{1}{\varepsilon} & 1 \end{bmatrix} \tag{3.11}$$

Suponhamos um ponto no plano da imagem $(x_0, y_0, 0)$, onde a terceira coordenada indica que o plano é o $z = 0$. Expressando este ponto em coordenadas homogêneas, temos:

$$c_h = \begin{bmatrix} kx_0 \\ ky_0 \\ 0 \\ k \end{bmatrix} \tag{3.12}$$

Aplicando a equação (3.10), chegamos no vetor com as coordenas do ponto no espaço:

$$w_h = \begin{bmatrix} kx_0 \\ ky_0 \\ 0 \\ k \end{bmatrix} \tag{3.13}$$

ou, em coordenadas cartesianas:

$$w = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} x_0 \\ y_0 \\ 0 \end{bmatrix} \tag{3.14}$$

Observa-se, no entanto, que a coordenada Z é igual a zero, o que não é o que se espera. Na verdade, dado um ponto no plano da imagem (x_0, y_0), o ponto real pode ser qualquer um que esteja na reta que passa pelos pontos $(x_0, y_0, 0)$ e $(0,0,\lambda)$. Se arranjarmos as equações (3.1) e (3.2), chegamos em:

$$X = \frac{x_0}{\varepsilon}(\varepsilon - Z) \tag{3.15}$$

$$Y = \frac{y_0}{\varepsilon}(\varepsilon - Z) \tag{3.16}$$

Ou seja, dado um ponto na imagem, só podemos saber qual é o ponto no espaço se soubermos alguma coisa a respeito dele, a sua coordenada Z, por exemplo. Esta necessidade pode ser sanada usando-se o conceito de visão estéreo.

3.2 - Visão Estéreo

A visão estéreo é uma ferramenta importante dentro da visão computacional. A visão estéreo possibilita a reconstrução de cenários 3D usando várias imagens diferentes

da mesma cena. A Fig. 3.2 mostra um ponto projetado em dois planos de projeção (ou duas câmeras).

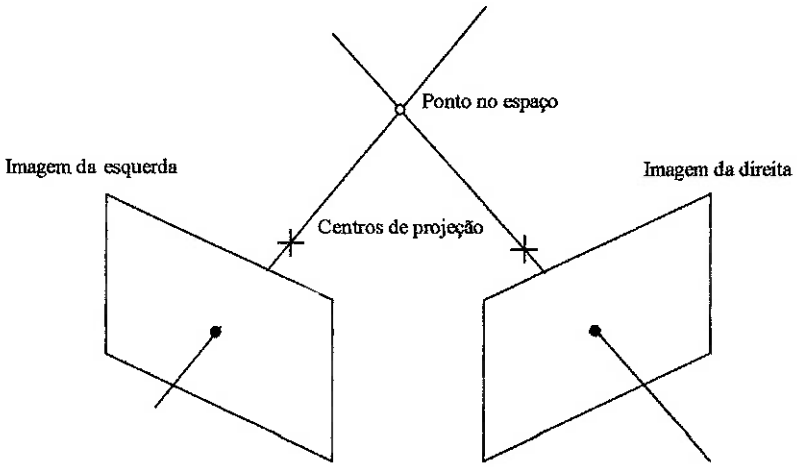

Figura 3.2 - Ponto no espaço projetado em dois planos

O princípio básico na visão estéreo é a triangulação. Como pode ser observado na Fig.3.2, se considerarmos projeções perspectivas, um ponto no espaço estará em um reta que passa pelo ponto projetado em um plano e o seu centro de projeção. Se o ponto correspondente no outro plano é conhecido, então o ponto também estará na reta que passa pelo ponto correspondente e o outro centro de projeção. Logo, o ponto no espaço é determinado pela intersecção das retas supracitadas. A determinação desta intersecção é o que chamamos de triangulação.

No entanto, empiricamente é difícil definir a correspondência entre os pontos das imagens, seja porque pode haver ambigüidade na correspondência entre vários pontos, o que leva a diferentes interpretações da mesma cena, ou porque simplesmente pode não haver o correspondente de um ponto em uma imagem na outra (o que é intuitivo, uma vez que o campo de visão de duas câmeras que vêem uma mesma cena de pontos distintos é diferente – Fig.3.3), o que é chamado de oclusão. Vários algoritmos já foram propostos para a correspondência entre pontos. Fielding e Kam (2000) mostram alguns deles.

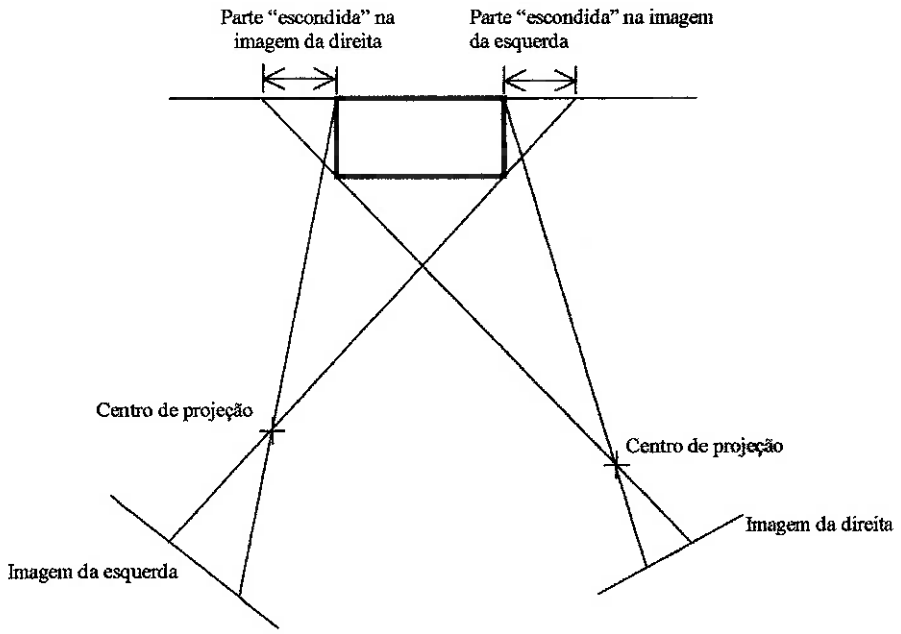

Figura 3.3 - Partes "escondidas" em cada uma das imagens (oclusão)

Para se determinar à correspondência entre elementos de duas imagens, ao invés de buscarmos, dado um ponto em uma imagem, o ponto correspondente na outra imagem inteira, usamos o que é chamada restrição epipolar. Esta restrição reduz o problema a uma busca em uma dimensão, ou seja, é necessário fazer a busca em uma única linha. Como é mostrado na Fig.3.4, dado um ponto em um plano, o ponto correspondente estará na projeção da reta que passa por este ponto e seu centro de projeção no plano em que se quer achar a correspondência.

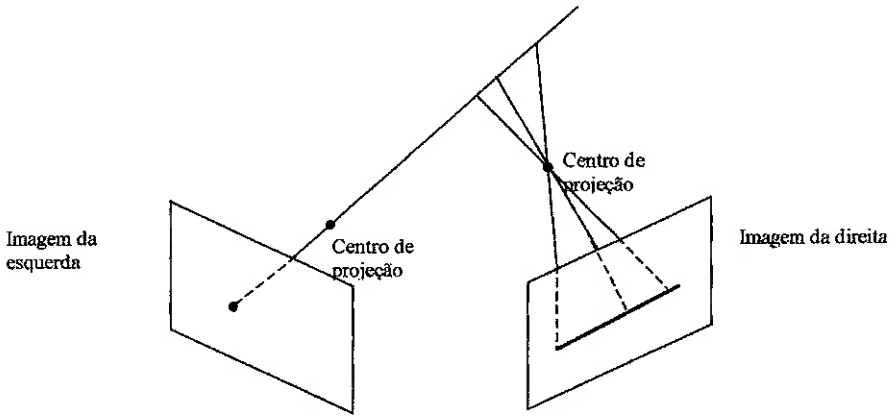

Figura 3.4 - Restrição epipolar

Podemos definir então as linhas epipolares. Considerando a Fig.3.5, a linha que conecta os dois centros de projeção é chamado de 'baseline'. Um plano qualquer que passe por essa linha é chamado de plano epipolar. Este plano, em geral, terá uma intersecção com os dois planos onde se formam as imagens. Estas intersecções, que são retas, são chamadas de linhas epipolares. Claramente qualquer ponto sobre a linha epipolar tem o seu correspondente, se este existir, na linha epipolar correspondente.

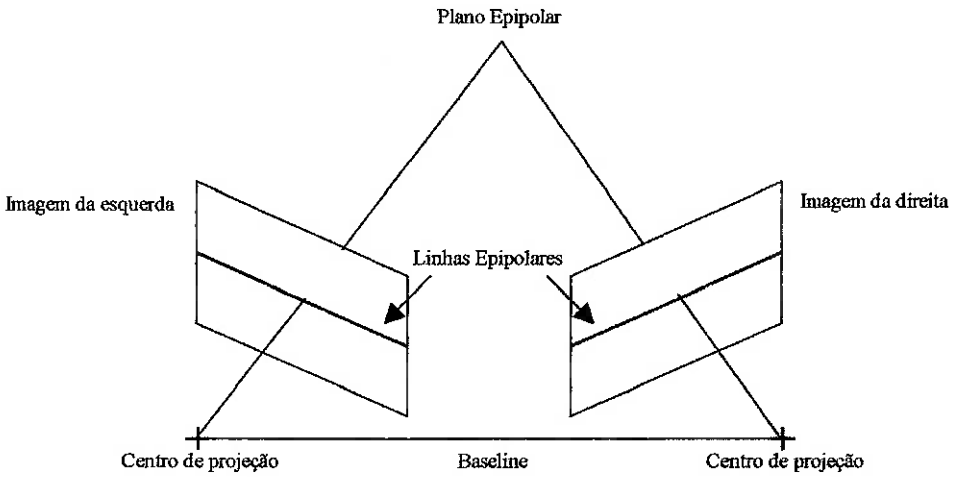

Figura 3.5 - A geometria epipolar

No caso do espelho duplo proposto, com as hipérboles alinhadas verticalmente (como será mostrado na próxima seção), obtemos as imagens dos dois espelhos em uma única e as linhas epipolares serão linhas radiais e alinhadas. Quando a imagem é projetada em um cilindro, ou seja, a imagem é retificada, as linhas epipolares tornam-se linhas paralelas e colineares. O arranjo do espelho duplo de duas seções coaxiais de perfis hiperbólicos naturalmente garante o alinhamento vertical das duas seções do espelho e, na medida em que somente uma imagem contém as duas vistas da mesma cena, o espelho duplo também garante linhas epipolares casadas, tanto na imagem omnidirecional como na imagem panorâmica. Nota-se que o alinhamento perfeito das imagens é necessário para se ter linhas epipolares verticais paralelas e colineares nas

imagens panorâmicas, de forma a permitir o processamento estéreo de forma eficiente e em tempo real, como é mostrado na Fig 3.6.

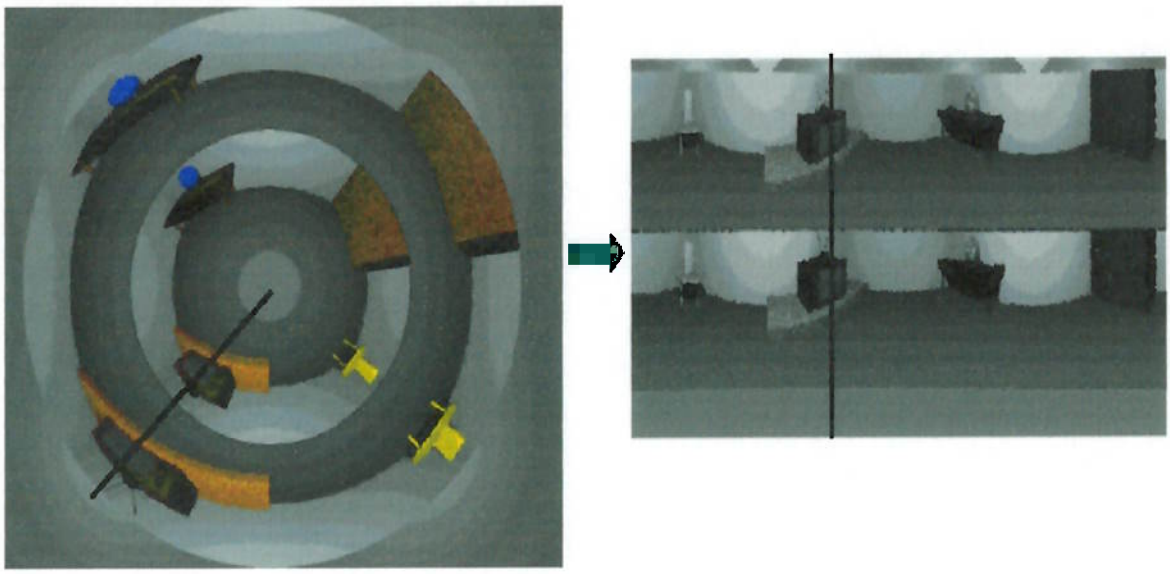

Figura 3.6 - Linhas epipolares no sistema

Capítulo 4

PROJETO DO SISTEMA

4.1-Projeto do espelho

Apesar deste trabalho não estar interessado no projeto do espelho em si, alguns pontos podem ser considerados para um melhor entendimento do processo de recuperação tridimensional. No projeto do espelho, deseja-se obter o perfil das superfícies das duas seções do espelho, interna e externa, de forma que tenham perfil hiperbólico e apresentem as seguintes propriedades:

- As duas seções do espelho são projetadas para terem o mesmo campo de visão;
- Deseja-se obter a mesma resolução nas duas seções do espelho. Para isso, as imagens das seções interna e externa devem ter o mesmo intervalo de raio, ou seja:

$$(\rho_{E,\max} - \rho_{E,\min}) = (\rho_{I,\max} - \rho_{I,\min}) \tag{4.1}$$

onde $\rho_{I,\max}, \rho_{I,\min}, \rho_{E,\max}$ e $\rho_{E,\min}$ são, respectivamente, os raio máximo e mínimo das seções interna e externa;

- Para se obter a melhor resolução possível, a projeção do espelho tem que ocupar toda a imagem, assim deve-se ter $\rho_{E,\max} = \rho_{imagem,\max}$, onde $\rho_{imagem,\max}$ é o raio máximo da imagem;

A Fig.4.1 mostra um sistema de visão estéreo omnidirecional baseado em um espelho duplo. As equações do espelho que foram usadas nas simulações foram calculadas nessa etapa do projeto.

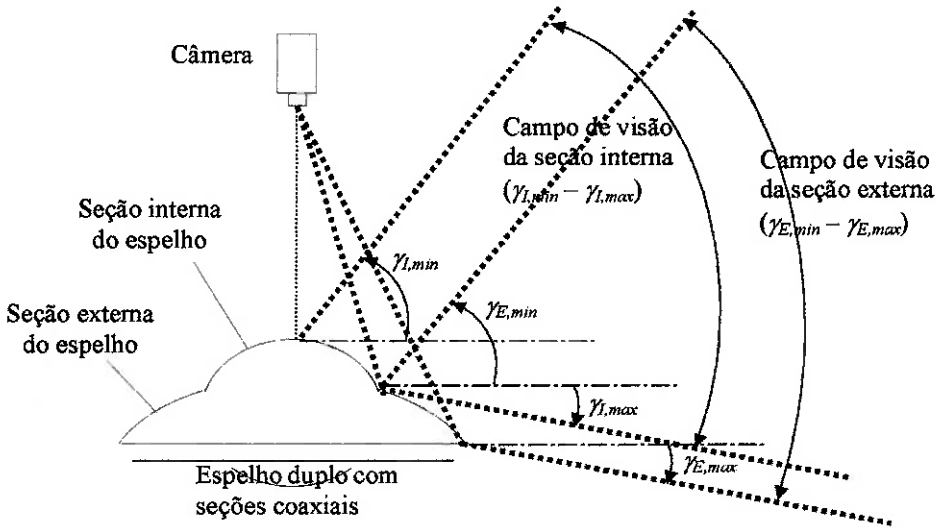

Figura 4.1 - Campo de visão das seções de um espelho duplo

4.2-Processo de recuperação tridimensional

O processo de recuperação 3D da imagem, ou seja, o cálculo da distância de cada ponto no espaço, pode ser dividido nas seguintes etapas: (1) transformação da imagem omnidirecional para perspectiva (retangular); (2) extração das características; (3) correspondência; (4) triangulação.

A etapa (1) é facilmente realizada transformando as coordenadas da imagem omnidirecional de polares para cartesianas, ou seja, projetando a imagem em um cilindro. Dessa forma, as linhas epipolares, que no caso de um espelho duplo de perfil hiperbólico, com as hipérboles alinhadas verticalmente, são curvas radiais, tornam-se linhas paralelas e colineares.

As etapas (2) e (3) talvez sejam as mais difíceis do processo de recuperação tridimensional. Alguns métodos e algoritmos para a determinação da correspondência entre as imagens são discutidos na seção 4.4.

Depois de realizadas as etapas (2) e (3), pode-se fazer a triangulação, que é o cálculo da posição no espaço propriamente dita. Observando a Fig.4.2 e a Fig.4.3, podemos equacionar as coordenadas de um ponto P no espaço da seguinte forma: um raio de luz emitido pelo ponto P que incide na seção interna é dado pela equação:

$$z_P = -r_P.\cot an\phi_I + z_{FI} \tag{4.2}$$

onde z_P, r_P são as coordenadas do ponto P. Analogamente, para a seção externa, temos:

$$z_P = -(r_P - r_{FE}).\cot an\phi_E + z_{FE} \tag{4.3}$$

Resolvendo o sistema de equações formado por (4.2) e (4.3), chegamos em:

$$r_P = \frac{\tan\phi_I\left[\tan\phi_{FE}\left(z_{FI} - z_{FE}\right) - r_{FE}\right]}{\left(\tan\phi_E - \tan\phi_I\right)} \tag{4.4}$$

$$z_P = \frac{z_{FE}.\tan\phi_E - z_{FI}.\tan\phi_I + r_{FE}}{\left(\tan\phi_E - \tan\phi_I\right)} \tag{4.5}$$

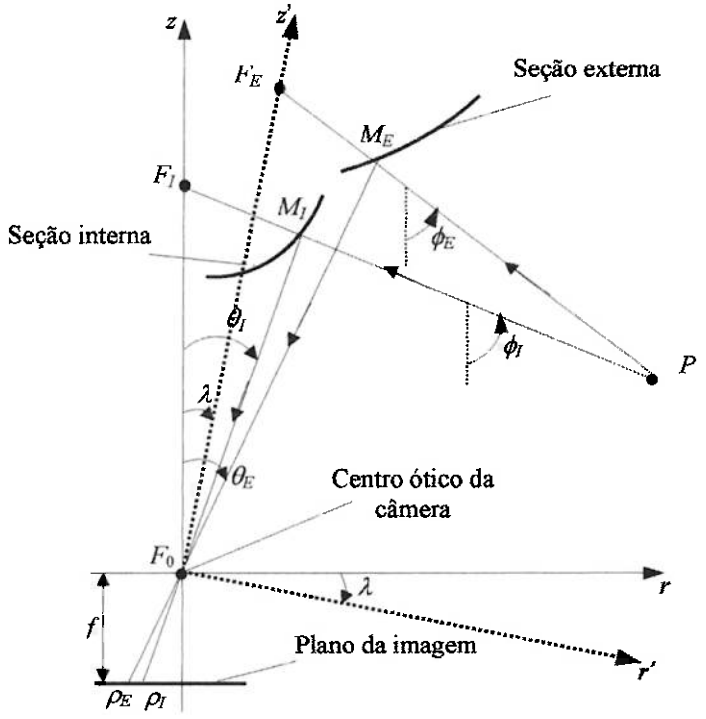

Figura 4.2 - Esquema do espelho duplo de perfil hiperbólico mostrando os raios de luz incidentes e refletidos

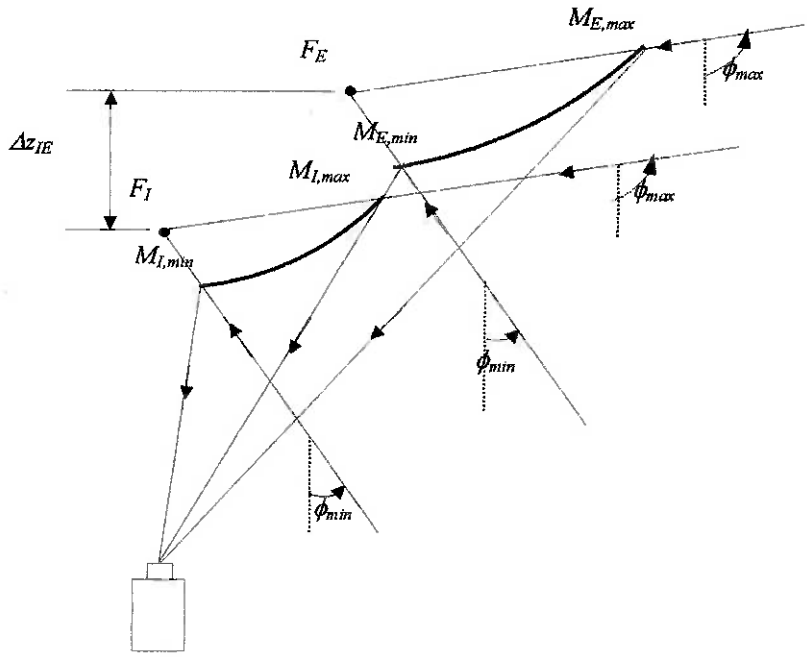

Figura 4.3 – Esquema do espelho duplo mostrando os raios de luz incidentes com ângulos de elevação máximo e mínimo

Como se pode observar nas equações (4.4) e (4.5), para calcular o ponto P no espaço, é necessário conhecer os ângulos de elevação ϕ_I e ϕ_E dos raios que partem de P. Para isso, deve-se calcular as projeções M_I e M_E do ponto P nas seções do espelho. Essas projeções podem ser calculadas a partir dos raios dos pontos nas imagens interna e externa, (ρ_I e ρ_E), obtidos no processo de correspondência de pontos das duas imagens. O cálculo dos ângulos de elevação é demonstrado detalhadamente na seção 4.5.

4.3-Retificação das imagens

Junior (2003) discute em seu trabalho três formas de retificar a imagem panorâmica obtida através de um sistema de visão omnidirecional. Neste trabalho, optamos por usar um método de transformação direta de coordenadas polares para cartesianas. Ou seja, tendo as coordenadas polares na imagem, usamos a transformação descrita no sistema de equações (4.6). A coordenada do anglo de azimute na imagem original é mapeada no eixo de coordenada horizontal da imagem retificada, e a coordenada radial na imagem original é mapeada no eixo de coordenada vertical da

imagem retificada, conforme mostrado na Fig.4.4. O algoritmo implementado é apresentado no Anexo B.

$$\begin{cases} x = r.\cos\theta + x_0 \\ y = r.sin\theta + y_0 \end{cases} \tag{4.6}$$

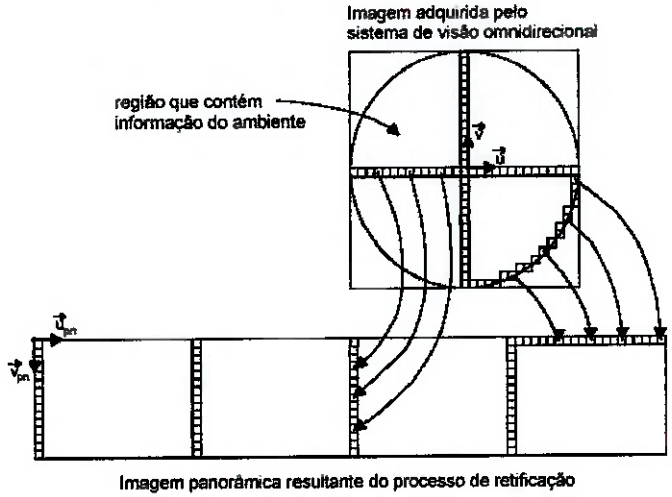

Figura 4.4 – Esquema mostrando o processo de retificação de imagem (in Junior, 2003)

Abaixo, na Fig.4.5, é mostrada a imagem da Fig.2.3 depois de aplicada a retificação. A imagem de cada seção do espelho tem 224 pixels de altura e 360 pixels de largura, ou seja, cada linha vertical das imagens correspondem a 1°.

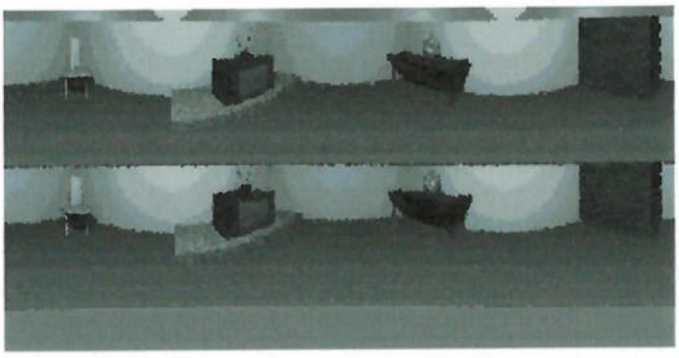

Figura 4.5 - Imagem panorâmica do ambiente mostrado na Fig.2.3

4.4-Correspondência

Um problema fundamental na visão estéreo é a determinação da correspondência entre pontos, ou seja, localizar em uma imagem o ponto que corresponde a um outro na outra imagem. Como já dito na seção 3.2, dado um ponto em uma imagem, o seu correspondente deve estar na linha epipolar da outra. Existem alguns métodos para se fazer esta procura, baseados na comparação entre as intensidades dos pixels ou na detecção de arestas. Alguns deles podem ser vistos em Shirai (1987) e Nalwa (1993). Neste projeto, utilizaremos o método descrito em Fielding e Kam (2000), chamado heurística esquerda-direita (*left-right heuristic*).

Um jeito popular de se fazer a correspondência é a minimização de uma função de similaridade. Usa-se uma função de similaridade entre um pixel de uma imagem e os pixels candidatos a correspondente da outra e para o par de pixels em que a função de similaridade assumir o menor valor está feita a correspondência. Definimos $E(y_e, x_e)$ como a imagem do espelho externo e $I(y_i, x_i)$ como a imagem do espelho interno. Além disso, ainda há um parâmetro importante a ser definido, a disparidade. A disparidade (d) é a distância (ou diferença de coordenadas) entre pixels correspondentes nas duas imagens. Como as linhas epipolares são colineares, imagens iguais em tamanho e resolução e a busca é feita em apenas uma dimensão, teremos $E(y+d,x) = I(y,x)$ para pontos correspondentes, se existirem. As funções de similaridade mais comuns são: correlação normalizada (CN), soma do quadrado das diferenças (SQD) e soma das diferenças absolutas (SAD) – equações 4.7, 4.8 e 4.9 respectivamente.

$$f_{NC}(y,x,d) = -\sum_{i,j \in W} \frac{(E(y+j,x+i)-\mu_e)}{\sigma_e} \frac{(I(y+j,x+i+d)-\mu_i}{\sigma_i} \tag{4.7}$$

$$f_{SQD}(y,x,d) = \sum_{i,j \in W} [E(y+j,x+i) - I(y+j,x+i+d)]^2 \tag{4.8}$$

$$f_{SAD}(y,x,d) = \sum_{i,j \in W} | E(y+j,x+i) - I(y+j,x+i+d) | \tag{4.9}$$

Nas equações (4.7) a (4.9), W é uma janela centrada no pixel (y,x) e é dada por $W = \{(i,j) : -p \leq i \leq p, -q \leq j \leq q\}$. Escolhemos implementar a função de similaridade SAD, pois é eficiente e simples, permitindo maior rapidez nos cálculos, requisito para aplicação em tempo real.

Como citado anteriormente, o método utilizado neste trabalho é o de heurística esquerda-direita. Este método consiste em, dado um ponto na imagem externa, por exemplo, procurar o pixel correspondente na imagem interna, usando-se para isso a função de similaridade. A busca é feita na linha epipolar e nos pixels que estejam dentro dos limites de busca, ou seja, respeitem à disparidade máxima imposta. Depois de correspondido o pixel, o inverso é feito. Usando-se o pixel correspondente na imagem interna, faz-se a busca na imagem externa. A correspondência só é aceita caso os resultados das duas buscas coincidam. O código fonte do programa que realiza este método, implementado em MatLab, é apresentado no Anexo C. Os parâmetros de entrada desta função são as duas imagens (interna e externa), o tamanho da janela, que é a área para a qual se deseja fazer a correspondência, o custo de oclusão, a máxima e mínima disparidades. O custo de oclusão representa pontos a oclusão descrita na seção 2.2. Abaixo são mostradas duas figuras resultantes do método de correspondência empregado. A Fig.4.6 mostra a plotagem das linhas epipolares nas duas imagens e a Fig.4.7 mostra dois pontos correspondidos nas imagens correspondentes ao ângulo de azimute de 50°, sendo o 0° a primeira linha vertical da Fig.4.5.

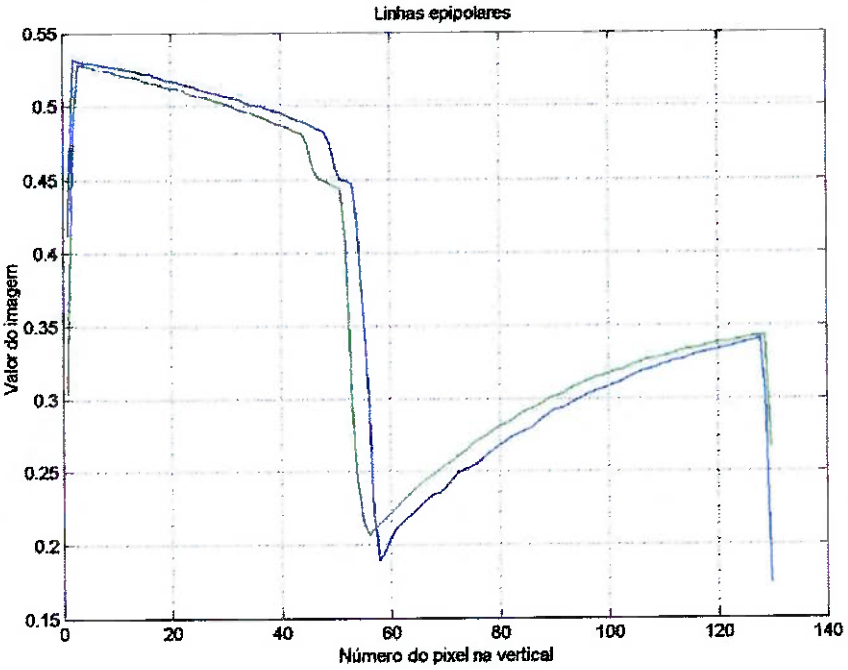

Figura 4.6 – Linhas epipolares

Figura 4.7 – Imagens com um ponto correspondido. O ponto correspondido é um pequeno ponto branco, circulado para melhor visualização.

4.5-Triangulação

Como discutido na seção 4.2, a resolução das equações (4.4) e (4.5) nos dá as coordenadas de um ponto P no espaço. Para tanto, é preciso se conhecer os ângulos de elevação ϕ_I e ϕ_E. Antes de mostrarmos como estes ângulos são conhecidos, serão mostrados algumas propriedades e elementos da hipérbole. Na Fig.4.8 vemos uma hipérbole e seus principais elementos. A hipérbole é o lugar geométrico dos pontos P de um plano tais que $| d_{PF1} - d_{PF2} | = 2a$, sendo que a é uma certa distância tal que $2a < 2c$, onde 2c é a distância entre os focos F_1 e F_2. Ainda podemos estabelecer a relação $a^2 + b^2 = c^2$ entre as medidas da hipérbole.

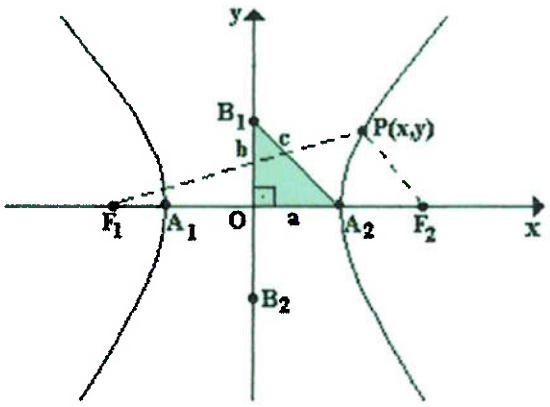

Figura 4.8 – Hipérbole

Logo, observando-se novamente a Fig.4.2, tem-se a seguinte relação:

$$\overline{F_0 M_I} - \overline{F_I M_I} = 2a_I \tag{4.10}$$

Aplicando a Lei dos senos no triângulo $F_I F_0 M_I$, tem-se:

$$\frac{\overline{F_I M_I}}{\sin\theta_I} = \frac{\overline{F_0 M_I}}{\sin\phi_I} \tag{4.11}$$

Combinando as equações (4.10) e (4.11) e re-arranjando resulta em:

$$\frac{\overline{F_I M_I}}{\sin\theta_I} = \frac{2a_I}{\sin\phi_I - \sin\theta_I}. \tag{4.12}$$

Aplicando novamente a Lei dos senos no triângulo $F_I F_0 M_I$, tem-se:

$$\frac{2c_I}{\sin(\theta_I + \phi_I)} = \frac{\overline{F_I M_I}}{\sin\theta_I}. \tag{4.13}$$

Combinado as equações (4.12) e (4.13) e re-arranjando, pode-se escrever:

$$\frac{c_I}{a_I} = \sin\theta_I \left(\frac{c_I + a_I \cos\phi_I}{a_I \sin\phi_I} \right) + \cos\theta_I. \tag{4.14}$$

Lembrando que $\cos^2\theta_I + \sin^2\theta_I = 1$ e na equação (4.14) denominado $\sin\theta_I = x$ e $\cos\theta_I = y$, tem-se duas expressões cuja solução simultânea representa o cruzamento de uma reta com um círculo de raio unitário, ou seja:

$$\text{Reta: } y = -K_1 x + K_2 \tag{4.15}$$

$$\text{Círculo: } x^2 + y^2 = 1 \tag{4.16}$$

onde $K_1 = \left(\frac{c_I + a_I \cos\phi_I}{a_I \sin\phi_I} \right)$ e $K_2 = \frac{c_I}{a_I}$. Observa-se que tanto quanto *K2* é constante para um dado ângulo ϕ_I. A Fig.4.9 ilustra o problema representado pelas equações (4.15) e (4.16). Como mostra a Fig.4.9, existem duas soluções para o ângulo θ_I (θ_1 e θ_2) que satisfazem essas equações, uma correta (θ_2) e outra incorreta (θ_1).

Aplicando a Lei dos senos no triângulo ABC da Figura 4.9, tem-se:

$$\frac{1}{\sin\varphi} = \frac{c_I / a_I}{\sin\eta} \tag{4.17}$$

Dos triângulos *ABC* e *CBD* da Fig.4.9 têm-se as seguintes relações entre ângulos:

$$\eta = 90° - \frac{\theta_1 - \theta_2}{2} \tag{4.18}$$

$$\varphi = 90° - \frac{\theta_1 + \theta_2}{2} \tag{4.19}$$

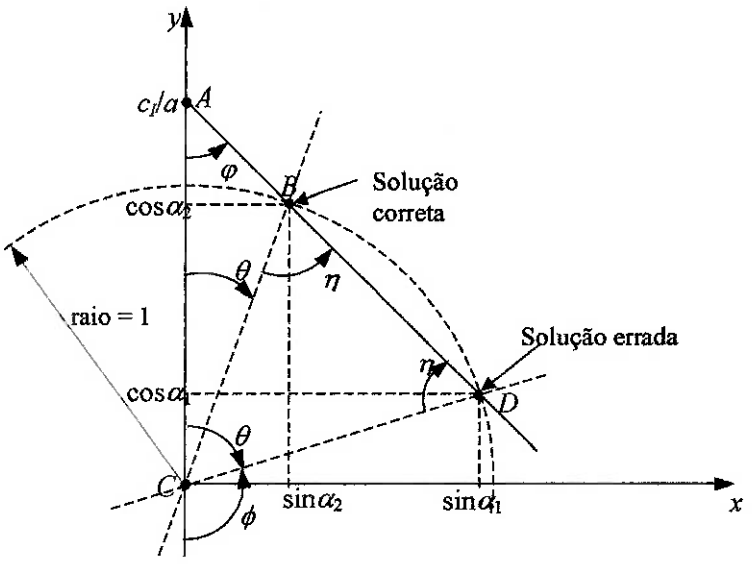

Figura 4.9 - Análise geométrica do problema da seção do espelho

Utilizando-se de relações trigonométricas a equação (4.17) pode ser reescrita da seguinte forma:

$$\frac{1}{\cos(\theta_1/2)\cos(\theta_2/2) - \sin(\theta_1/2)\sin(\theta_2/2)} = \frac{c_I / a_I}{\cos(\theta_1/2)\cos(\theta_2/2) + \sin(\theta_1/2)\sin(\theta_2/2)} \tag{4.20}$$

Multiplicando ambos os lados por $\cos(\theta_1/2)$ e re-arranjando resulta em:

$$\tan(\theta_1/2) = \cot(\theta_2/2)\left(\frac{c_I - a_I}{c_I + a_I}\right) \tag{4.21}$$

Denominando,

$$\alpha_I = \left(\frac{c_I - a_I}{c_I + a_I}\right) \tag{4.22}$$

e observando que $\theta_1 + \phi = 180^\circ$, portanto $\tan(\theta_1/2) = \tan^{-1}(\phi_I/2)$, resulta em:

$$\tan(\phi_I/2) = \frac{\tan(\theta_2/2)}{\alpha_I} \tag{4.23}$$

A equação (4.21) em conjunto com a relação trigonométrica $\tan\beta = \frac{2\tan(\beta/2)}{1-\tan^2(\beta/2)}$, resulta na equação (4.24) a seguir.

$$\tan\phi_I = \frac{2\alpha_I \tan(\theta_2/2)}{\left(\alpha_I^2 - \tan^2(\theta_2/2)\right)} \tag{4.24}$$

Na medida em que a solução correta para θ_I é θ_2, observando-se a Fig.4.2, tem-se as seguintes relações trigonométricas:

$$\sin\theta_I = \frac{\rho_I}{\sqrt{f^2 + \rho_I^2}} \tag{4.25}$$

$$\cos\theta_I = \frac{f}{\sqrt{f^2 + \rho_I^2}} \tag{4.26}$$

Fazendo uso da relação trigonométrica $\tan(\beta/2) = \sin(\beta)/\left(1+\cos(\beta)\right)$, a equação (4.24) resulta em:

$$\tan\phi_I = \frac{2\alpha_I\rho_I\left(f+\sqrt{f^2+\rho_I^2}\right)}{\alpha_I^2\left(f+\sqrt{f^2+\rho_I^2}\right)^2-\rho_I^2} \tag{4.27}$$

Similarmente para a seção externa do espelho, realizando-se a mesma análise, porém considerando a rotação do eixo da hipérbole de um ângulo λ, chega-se à seguinte expressão:

$$\tan\phi'_{E,rot} = \frac{2\alpha_E\rho'_E\left(f'+\sqrt{f'^2+\rho'^2_E}\right)}{\alpha_E^2\left(f'+\sqrt{f'^2+\rho'^2_E}\right)^2-\rho'^2_E} \tag{4.28}$$

onde $\alpha_E = \left(\frac{c_E - a_E}{c_E + a_E}\right)$, f' e ρ'_E são respectivamente a distância e o raio da imagem vistos pelo sistema de coordenadas F_0-$r'z'$, sendo dados pela inversa da equação (4.29) da seguinte forma:

$$\begin{bmatrix} r \\ z \end{bmatrix} = \begin{bmatrix} \cos\lambda & sin\lambda \\ -sin\lambda & \cos\lambda \end{bmatrix}\begin{bmatrix} r' \\ z' \end{bmatrix} \tag{4.29}$$

$$\begin{bmatrix} \rho'_E \\ f' \end{bmatrix} = \begin{bmatrix} \cos\lambda & -\sin\lambda \\ \sin\lambda & \cos\lambda \end{bmatrix}\begin{bmatrix} \rho_E \\ f \end{bmatrix} = \begin{bmatrix} \rho_E\cos\lambda - f\sin\lambda \\ \rho_E\sin\lambda + f\cos\lambda \end{bmatrix} \tag{4.30}$$

Tendo-se ϕ'_E o ângulo de elevação do raio incidente no espelho externo no sistema de coordenadas F_0-rz, ϕ_E, é obtido por:

$$\phi_E = \phi'_E - \lambda \tag{4.31}$$

Para evitar o cálculo do arco de tangente, a equação (4.28) pode ser reescrita em

função das tangentes dos ângulos resultando em:

$$\tan\phi_E = \frac{\tan\phi'_E - \tan\lambda}{1+\tan\phi'_E \tan\lambda} \tag{4.32}$$

Em resumo, tendo-se os raios dos pontos correspondentes nas imagens interna e externa (ρ_I e ρ_E), usando-se as equações (4.25), (4.26), (4.27) e (4.28) calcula-se os ângulos de elevação dos raios incidentes (θ_I e θ_E) e através das equações (4.4) e (4.5) obtém-se a posição do ponto no espaço. Nota-se que no cálculo da posição de um ponto no espaço não é necessário o cálculo dos ângulos envolvidos, somente de suas tangentes, o que torna esses cálculos eficientes e fáceis de serem implementados em um computador digital.

Capítulo 5

RESULTADOS

5.1 – Resultados obtidos

As equações apresentadas na seção 4.5 foram implementadas e testadas. O algoritmo feito é apresentado no Anexo D. Depois de rodada a função de correspondência 'corrtot', usando-se como parâmetros de entrada os valores 3 e 1 para o tamanho vertical e horizontal da janela de comparação, respectivamente, 0.20 para o critério de oclusão, 0 para mínima disparidade e 25 para a máxima, foi obtida uma matriz com os pontos correspondentes na imagem externa para cada ponto da imagem interna, sendo que os pontos não correspondidos fora atribuídos os valores 0, para os não correspondidos pelo critério flat (i.e., parte homogênea e sem textura que impossibilita a correspondência – e.g. céu, parede sem textura, etc.); -1, para os pontos não correspondidos pelo critério da heurística esquerda-direita; e -2, para os não correspondidos pelo critério de oclusão. Essa matriz é a entrada para a função que realiza a triangulação. A saída é uma matriz tridimensional de tamanho (X,Y,2), sendo que X e Y são as coordenadas do pixel e a terceira dimensão diz respeito a r_p (1) ou z_p (2). Os parâmetros do espelho são mostrados na Tab.5.1.

A função 'triangulação' primeiramente calcula os raios dos pontos correspondidos na imagem, através da função 'calcularaios', também mostrada no Anexo D. Essa função utiliza os dados da câmera P_{max} e r_{max}. O primeiro é o tamanho do raio em pixels. No caso, estamos simulando uma imagem de 1280x1024. Então P_{max} é igual a 512. Já r_{max} é o raio máximo do CCD, que é onde a imagem se forma, ou seja, é raio máximo que a imagem pode ter, em milímetros. Esse valor é $r_{max} = 3{,}5945$. Outro parâmetro da câmera é a distância focal f. Para este, temos que $f = 25mm$.

Parâmetro	Valor
c_I	135,1501 mm
a_I	123,7943 mm
c_E	210,5443 mm
a_E	193,6103 mm
λ	3,5067°
z_{FE}	420,3002 mm
z_{FI}	270,3002 mm
r_{FE}	25,7558 mm

Tabela 5.1 – Parâmetros do espelho utilizado

A função que realiza a triangulação foi testada da seguinte forma: uma parte da imagem construída no software Pov-Ray foi escolhida aleatoriamente. No caso, escolhemos os pixels compreendidos na coluna correspondente a 130°, na região da "TV". A parte testada é mostrada abaixo na Fig. 5.1. As coordenadas da tela da TV no código fonte do ambiente são mostradas na Tab.5.2.

Figura 5.1 - Parte da imagem testada

```
//tela
 box{
   <-2300,-400,1800><-1300,-900,1799>
   pigment {color DimGrey}
   finish {reflection {.1}}
 }
```

Tabela 5.2 - Código da tela da TV

Vemos então que as coordenadas de interesse em relação ao centro focal da câmera são $-400 \leq y = z_p \leq -900$ e $z = r_p = 1800$. Abaixo são mostrados os resultados obtidos para duas posições da tela, a primeira na linha 60, 130, e outra mais abaixo, linha 70.

	xyz(60,130,:)	xyz(70,130,:)
x	2213,9	2441,7
y	-440,4	-629,1

Tabela 5.3 - Alguns resultados

Antes de analisarmos os resultados obtidos, devemos lembrar que esses resultados correspondem a valores em coordenadas cilíndricas, pois são valores obtidos para uma base 'alinhada' com o ângulo de azimute 130°. A coordenada y, que corresponde à z_p, ou seja, a altura, é igual em ambas as base. Multiplicando então as coordenadas x, que correspondem à r_p, pelo seno de 130° chegamos aos novos resultados:

	xyz(60,130,:)	xyz(70,130,:)
x	1695,9	1870,45
y	-440,4	-629,1

Tabela 5.4 - Resultados após transformação de base

Para a região da tela, o mínimo valor de x obtido, para os pontos onde houve correspondência foi de 2172,0, ou, passando para a base fixa na câmera, 1663,8. O máximo foi de 2582,8, ou 1978,5 na base fixa.

5.2 – Análise de erros

O erro da reconstrução estéreo é discutido em Cabral e Souza (2004). O erro da posição é função da própria posição, ou seja, o erro depende das coordenadas r_p e z_p. Quanto mais longe o ponto do sistema de coordenadas, maior será o erro. Observa-se que o erro máximo para cada coordenada é de 15%. Usando os valores obtidos, tem-se que o erro máximo para a tela da TV é de $\frac{|1978,5-1800|}{1800} = 9,92\%$. Abaixo é mostrado um gráfico do erro para a parte da imagem correspondente à tela da TV. Os pontos onde o erro é zero são os pontos para os quais não houve correspondência – por critérios discutidos na seção 5.1.

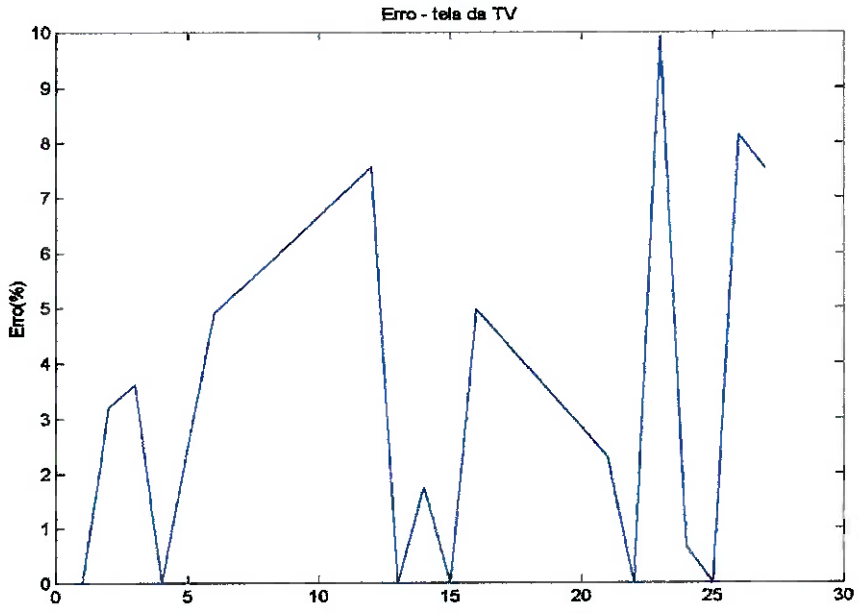

Figura 5.2 - Erro para a tela da TV

5.3 – Detecção de arestas e menor distância

De forma a reduzir o tempo computacional ao mínimo possível, ao invés de se realizar todo o procedimento de reconstrução para toda a imagem obtida, foi escolhido realizar a etapa de triangulação apenas para as bordas dos objetos e, após essa fase, calcular o ponto mais próximo do sistema. Foram utilizadas para isso algumas funções pré-existentes no próprio MatLab. As funções de detecção de arestas do MatLab procuram por bordas de objetos observando onde a intensidade da imagem muda rapidamente, usando os seguintes critérios:

- Lugares onde a primeira derivada da intensidade da imagem é maior do que a magnitude de um valor fixo pré-determinado;
- Lugares onde a segunda derivada tem um zero;

Essas funções existentes utilizam como base o gradiente da função f(x,y). A função f é a que dá a intensidade da imagem em (x,y). O vetor gradiente evidencia a direção da maior taxa de variação de f em (x,y). Esse vetor é definido como:

$$\nabla \vec{f} = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} \tag{5.1}$$

E a magnitude é:

$$\nabla f = mag(\nabla \vec{f}) = \left[G_x^2 + G_y^2\right]^{\frac{1}{2}} \tag{5.2}$$

Essa quantidade é igual ao valor máximo da taxa de incremento de f(x,y) por unidade de distância na direção de $\nabla \vec{f}$. Nas funções citadas é possível escolher o método de estimação das derivadas, sendo que os métodos testados foram o Canny e Sobel. A saber:

Canny – Encontra arestas procurando o máximo local do gradiente da imagem de entrada. O método usa duas bases para comparação, uma para arestas fortes e uma para arestas fracas e apenas mostra arestas fracas se estas estão conectadas a uma forte. Esse método é mais confiável por ter uma probabilidade maior de detectar arestas fracas e menor de ser influenciado por ruído.

Sobel – Encontra arestas usando uma aproximação de Sobel para a derivada. Retorna arestas nos pontos em que o gradiente da imagem de entrada é máximo.

Na Fig.5.3 tem-se a imagem panorâmica obtida na seção interna do espelho duplo e nas figuras 5.4 e 5.5 tem-se os resultados da aplicação dos métodos acima descritos.

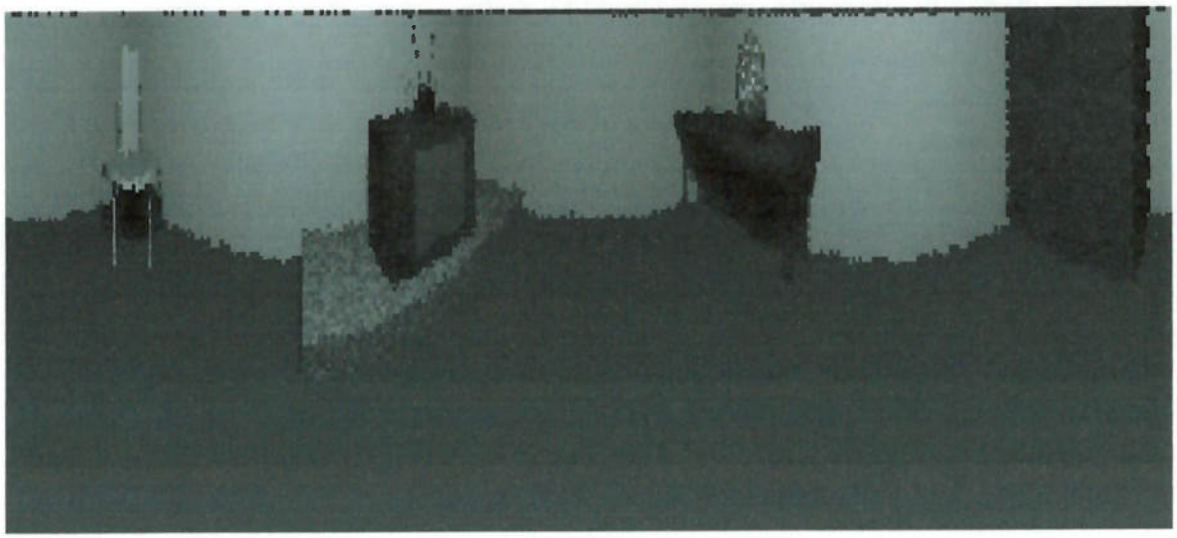

Figura 5.3 - Imagem do ambiente obtida na seção interna

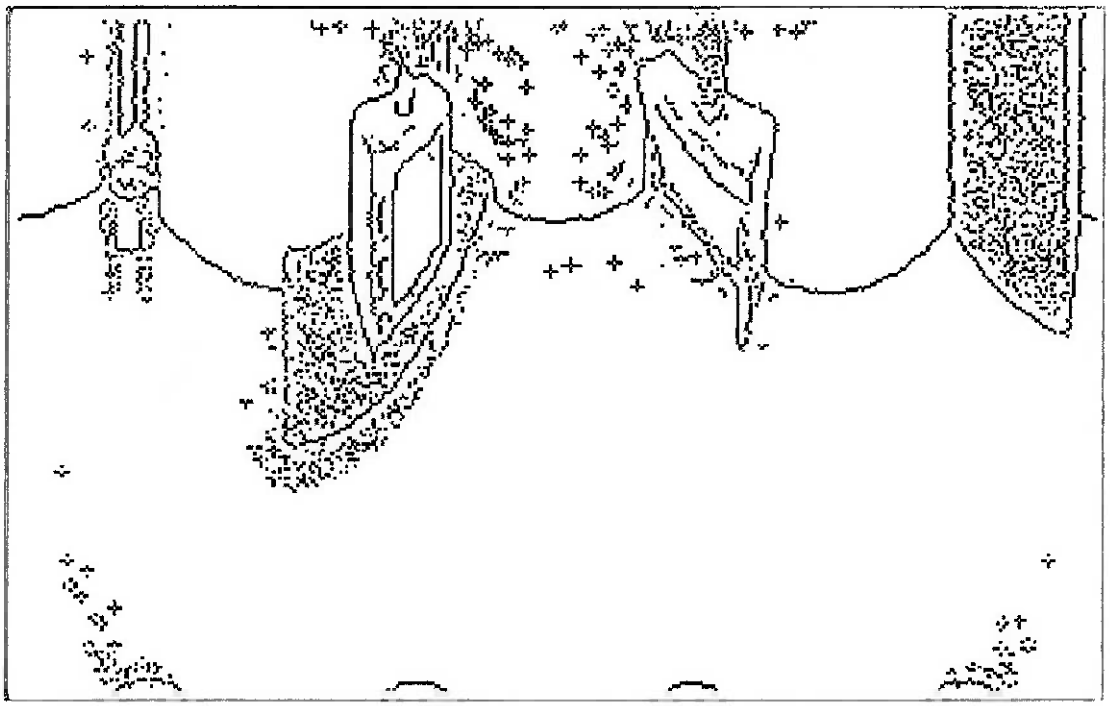

Figura 5.4 - Resultado do método Canny

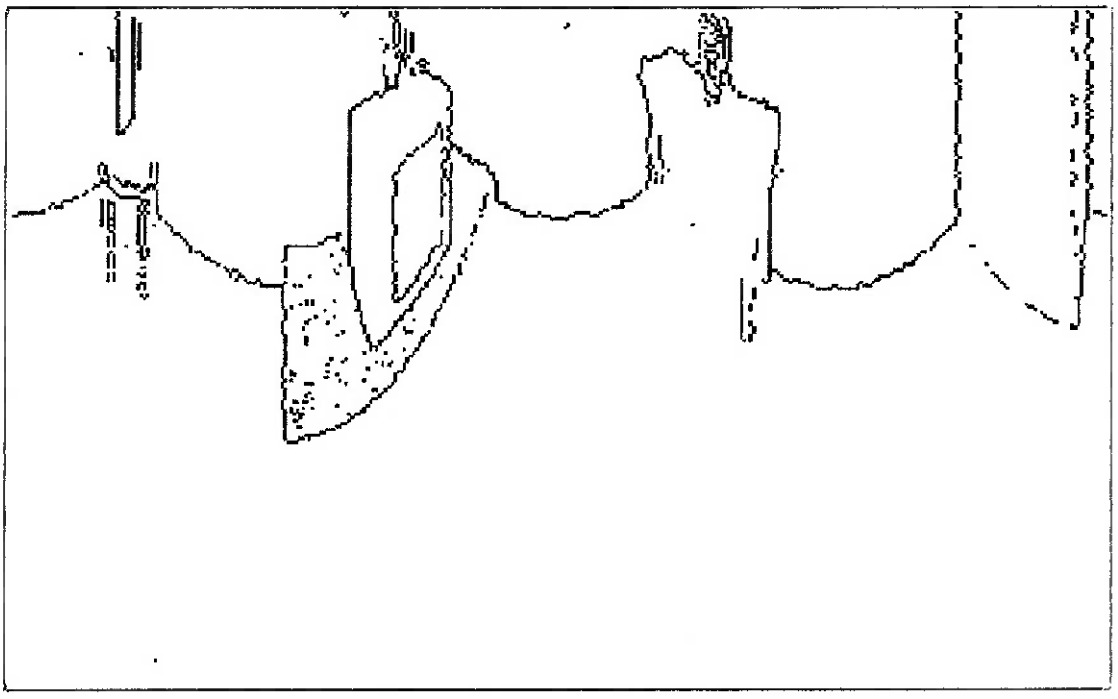

Figura 5.5 - Resultado do método Sobel

Nota-se que o método Sobel apresenta uma perda de linhas muito grande em comparação com o método Canny. Este, no entanto, apresenta alguns pontos isolados no seu resultado. Isso se deve ao fato de estarmos trabalhando com uma imagem simulada. Em uma imagem real, estes pontos nas paredes e chão que aparecem no resultado não estariam presentes.

Também foi implementado um método mais simples de detecção de arestas, simplesmente comparando-se o valor de um pixel com outro. Onde há mudanças rápidas de intensidade é onde se tem provavelmente uma borda ou aresta. Para tanto, foram escolhidas algumas colunas que contivessem algumas arestas para se ter uma idéia da magnitude da diferença de intensidade que há quando existe uma borda na imagem. A seguir, na Fig.5.6 são mostradas as seções da imagem utilizadas para determinar os valores de intensidade nas bordas e encontrar tal magnitude. As colunas escolhidas foram as que correspondem a 40°, 139° e 230° (a, b e c na Fig.5.6, respectivamente).

(a) coluna correspondente ao ângulo 40° de azimute

(b) coluna correspondente ao ângulo 139° de azimute

(c) coluna correspondente ao ângulo 230° de azimute

Figura 5.6 - Colunas escolhidas para determinar detecção de arestas

Para estas colunas, foram analisados os gráficos correspondentes à intensidade de cada pixel de cada coluna. Abaixo, são mostradas as curvas obtidas para cada uma das colunas.

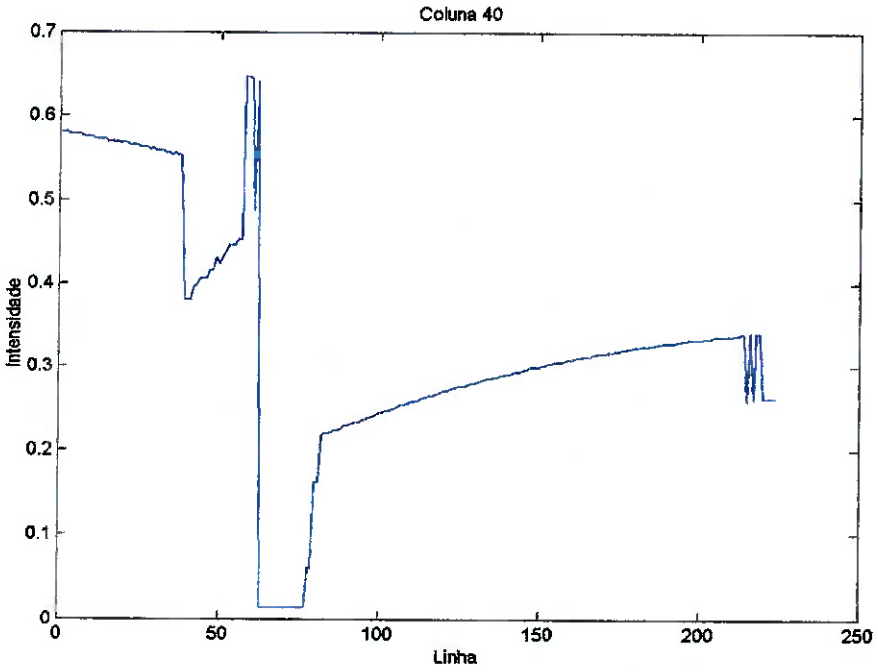

Figura 5.7 - Gráfico da intensidade para o ângulo 40°

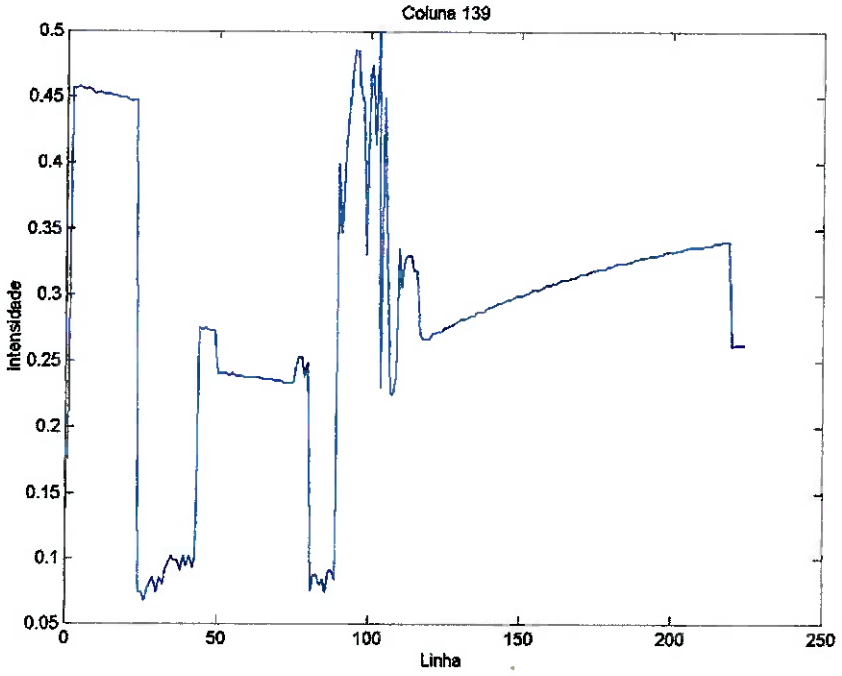

Figura 5.8 - Gráfico da intensidade para o ângulo 139°

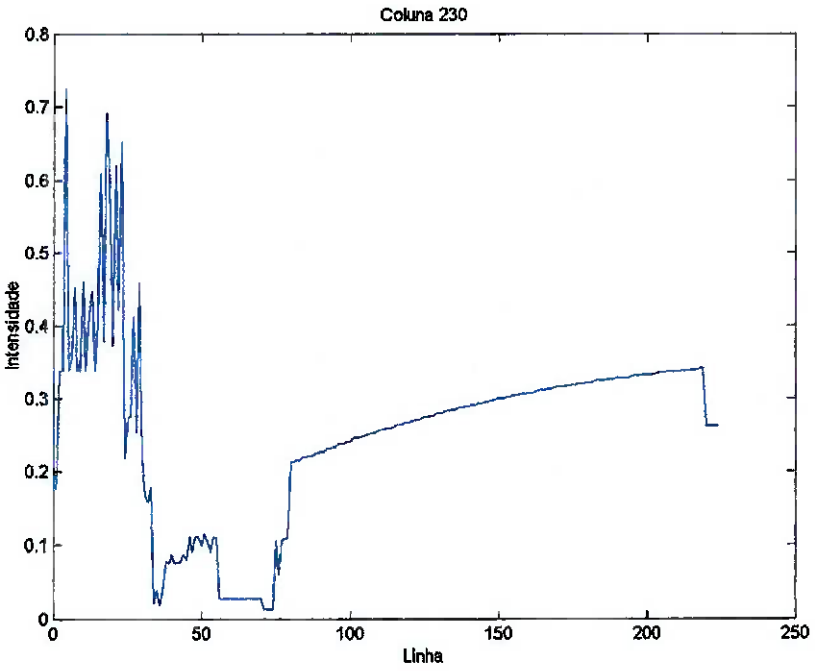

Figura 5.9 - Gráfico da intensidade para o ângulo 230°

Com estes gráficos em mão, e com a Fig.5.6, (a), (b) e (c), foram observados onde ocorria uma transição denotada pela brusca mudança de valores nas intensidades dos pixels, lembrando que esses valores variam de zero (ponto preto) a um (ponto branco) – na Fig.5.10 é mostrada uma figura com o ambiente e os valores de intensidade como exemplo. Assim, visualmente, foram determinadas as ocorrências de bordas para cada coluna. Nas tabelas 5.5, 5.6 e 5.7 são mostrados os valores de intensidade para 40°, 139° e 230° para as bordas que apresentam a menor diferença entre um pixel e outro. A menor diferença encontrada foi em uma borda no ângulo 40° de 0,0467.

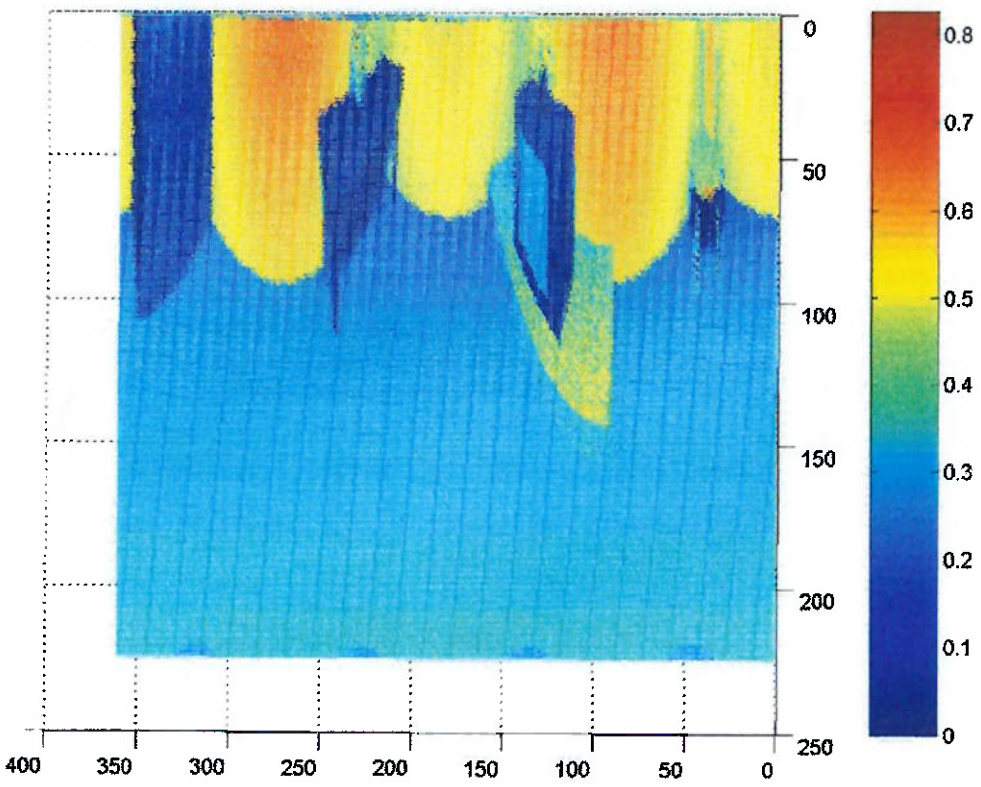

Figura 5.10 - Intensidade dos pixels na imagem do ambiente

	Intensidade	
Pixel 1	0,0137	
Pixel 2	0,0604	
Diferença		0,0467

Tabela 5.5 - Valores para 40°

	Intensidade	
Pixel 1	0,3187	
Pixel 2	0,2692	
Diferença		0,0495

Tabela 5.6 - Valores para 139°

	Intensidade	
Pixel 1	0,1000	
Pixel 2	0,0275	
Diferença		0,0725

Tabela 5.7 - Valores para 230°

Este valor encontrado para a diferença de intensidades entre dois pixels foi usado como referência para se determinar uma borda. Usando a imagem da seção interna do espelho, o que se fez foi copiar em uma variável os valores da imagem para quando temos a condição $|pixel_1 - pixel_2| > 0{,}045$. Na Tab.5.8 é mostrado um pseudocódigo usado para determinar as bordas, onde janv e janh representam uma janela pré-definida, m e n são o tamanho da imagem (colunas e linhas respectivamente), il é linha e ej é a coluna e iml é a imagem de entrada. O resultado dessa implementação é mostrado na Fig.5.11.

```
for ej = janh + 1 to m-janh
  for il = janv + 1 to n-maxdisp
    if (|(iml(il,ej)-iml(il-1,ej))| > 0.0450) then
      borda (il,ej) = 0
    else
      borda (il,ej) = 1
```

Tabela 5.8 – Pseudocódigo da detecção de arestas

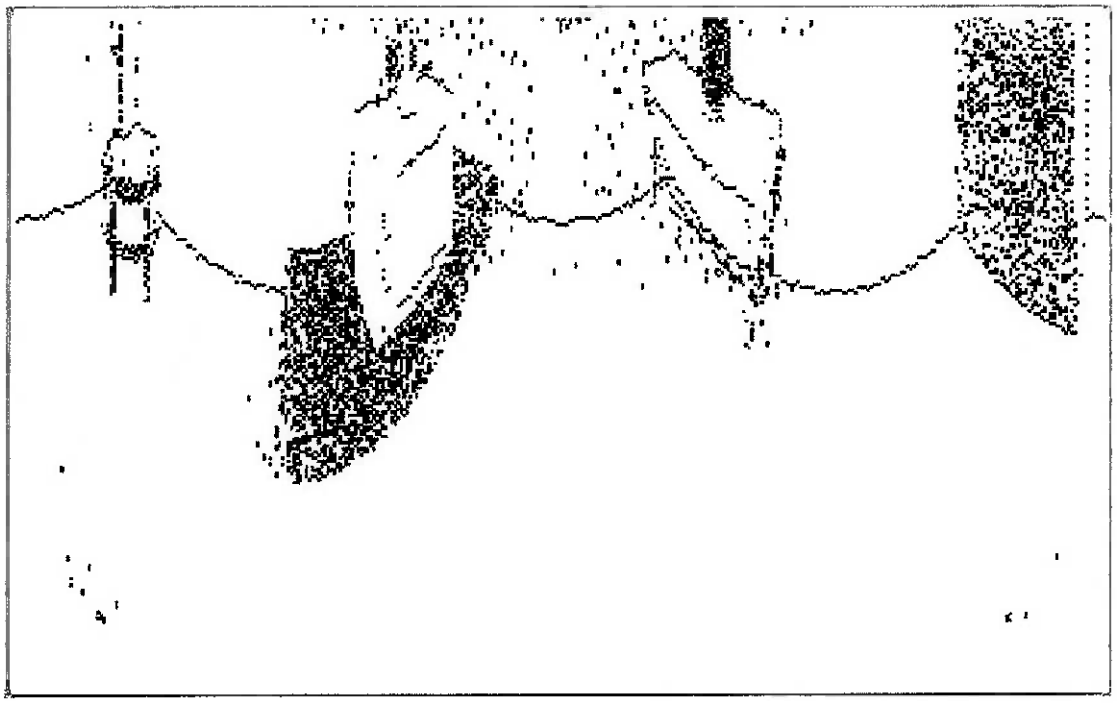

Figura 5.11 -Resultado da detecção de bordas pela comparação de intensidade entre pixels

A partir destes resultados foi calculado o ponto cuja distância ao centro focal da câmera, origem do sistema de coordenadas do sistema, é menor. Para isso utilizou-se um algoritmo simples, que é mostrado no Anexo E. Para cada uma das figuras 5.4, 5.5 e 5.11 foi rodada a rotina da que faz a triangulação. No entanto, foram calculados apenas os pontos que representam as bordas, ou seja, foi inserida uma condição a mais no código para que a triangulação fosse calculada para a matriz resultado da rotina de correspondência total, 'corrtot', apenas para as coordenadas onde nas figuras com as bordas houvesse um valor diferente de zero (valor de intensidade igual a zero corresponde a pontos pretos – as imagens citadas são apresentadas com as cores invertidas). Antes, no entanto, foi feito isso para o algoritmo de correspondência supracitado. O tempo computacional diminuiu consideravelmente para a triangulação executada apenas nas bordas. Após isso, foi executada a rotina que calcula o ponto de menor distância. O que se observou nos resultados é apresentado na Tab.5.8. Os resultados para as bordas obtidas pelo método Canny e Sobel são iguais e correspondem

a um ponto do armário, o maior objeto da cena. As coordenadas do armário no código fonte do ambiente são mostradas na Tab.5.9.

```
//Armário
 box{
    <4000,-1200,-3000><2500,1100,-500>
     pigment {
      DMFWood3
     scale 4
     }
  }
```

Tabela 5.9 - Código do armário

O resultado para as bordas pelo método de diferença entre pixels implementado apontou como o ponto mais próximo o chão, e, portanto, é ignorado, apesar de ter a magnitude menor do que os outros dois valores apresentados. Entretanto, esses valores parecem estar equivocados, uma vez que. Isso pode se dever ao fato dos métodos de detecção de bordas apresentados terem as suas performances afetadas devido ao uso de imagens simuladas ao invés de reais.

	Canny	**Sobel**	**Dif. entre pixels**
linha	59	59	181
coluna	348	348	343
distância	1511,00	1551,00	1111,6

Tabela 5.10 – Resultados da menor distância

Capítulo 6

DISCUSSÕES E CONCLUSÕES

Neste trabalho um sistema de visão estéreo omnidirecional baseado em visão estéreo computacional foi testado por meio de simulações usando-se rotinas implementadas em MatLab v.6 e um ambiente criado em PovRay v.3.5. Objetivou-se estudar, implementar e testar, dentre todas as fases do projeto apresentadas no capítulo 1, a etapa de recuperação tridimensional, ou seja, o cálculo das posições dos pontos da imagem omnidirecional, ressaltando que, devido à presença do centro único de projeção, os cálculos exigidos para realizar a triangulação estéreo, necessária para medir distâncias no espaço, são mais eficientes e rápidos do que em qualquer outra configuração de sistema de visão estéreo omnidirecional. Para tanto, usaram-se as equações do perfil do espelho previamente calculadas em uma etapa anterior do projeto.

Algumas dificuldades foram observadas por termos tratado apenas de imagens simuladas, como por exemplo, mostrado na seção 5.3, em que os algoritmos de detecção de bordas sofrem diminuição na sua performance e eficácia. As imagens simuladas também influenciam nos resultados calculados para os valores das coordenadas dos pontos nas imagens. No entanto, para os pontos testados foram obtidos valores razoáveis e com erros dentro do esperado. Apesar de um erro relativamente alto (15%), esse valor apresenta a acurácia necessária para a navegação de robôs móveis. Espera-se que os erros obtidos tenham seu valor diminuído, assim como a melhora nos métodos de detecção de bordas, quando imagens reais obtidas pelo sistema estiverem sendo tratadas. Espera-se também que em um ambiente real haja a correspondência entre mais pontos das imagens. Nos testes realizados houve muitos pontos não correspondidos pelo critério do método utilizado ou critério flat.

Pode-se concluir que os métodos utilizados para a correspondência, a heurística esquerda-direita juntamente com o SAD (soma das diferenças absolutas) para a similaridade entre pixels, apresentam um bom custo-benefício em relação ao tempo computacional, sendo que o tempo computado nas simulações deve diminuir em dez vezes quando implementado em um DSP. Os métodos de detecção de bordas também

diminuem significativamente o tempo de execução, uma vez que um número bem menor de pontos é calculado.

Por fim, vale ressaltar que o sistema de visão estéreo omnidirecional estudado e simulado, usando espelho duplo de perfil hiperbólico, apresenta algumas inovações, podendo-se incluir: espelho duplo de duas seções hiperbólicas com imagens de mesma resolução; geração de duas imagens estéreo com mesma resolução e mesmo campo de visão; espelho duplo com duas seções hiperbólicas com o eixo da hipérbole da seção externa inclinado em relação ao centro ótico da câmera e ao eixo da hipérbole da seção interna; etc..

ANEXOS

ANEXO A - Código PovRay

```
#include "colors.inc"
#include "textures.inc"
#include "stars.inc"

// focal blur camera
camera {
 location <0, 0, 0>

 right <8.960,0,0>    //obtido de 1280*7um
 up <0,0,-7.168>      //obtido de 1024*7um
 direction <0,25,0>   //distância focal de 25mm
 look_at  <0, 1, 0>
}

// Equação do perfil do espelho resolvida para vários pontos-Espelho 9.
//Não são mostrados todos os pontos do espelho, pois são muitos.
lathe {
 linear_spline
 2423,
        <4.5391, 259.3773>
        <4.5913, 259.3873>
        <4.6430, 259.3973>
        <4.6940, 259.4073>
        <4.7445, 259.4173>
        <4.7945, 259.4273>
          ...                 //demais pontos não mostrados
          ...
          ...
 texture { Aluminum}
 translate <0 0 0>
```

```
}

// Lâmpadas
light_source {<-200, 1300, -200> color White}
light_source {<200, 1300, -200> color White}
light_source {<-200, 1300, 200> color White}
light_source {<200, 1300, 200> color White}

 // Chão
plane { y, -1200
 //pigment { checker color Black, color White scale 500}
 pigment {Gray30}
}

// Teto
plane { y, 1300
 pigment { White}
}

// Paredes Laterais
plane { x, 4000
 pigment { Gray50}
}

plane { x, -4000
 pigment { Gray50}
}

plane { z, 3000
 pigment { Gray60}
```

```
}

plane { z, -3000
 pigment { Gray60}
}

//mesa
box{
<-4000,-400,-3000><-1000,-500,-2000>
pigment {
     DMFWood5      // pre-defined in textures.inc
     scale 4       // scale by the same amount in all directions
    }
    finish { Shiny } // pre-defined in finish.inc

}

cylinder{
<-3800,-1200,-2800><-3800,-500,-2800>50
pigment {
     DMFWood5      // pre-defined in textures.inc
     scale 4       // scale by the same amount in all directions
    }
    finish { Shiny } // pre-defined in finish.inc
}

cylinder{
<-3800,-1200,-2200><-3800,-500,-2200>50
pigment {
     DMFWood5      // pre-defined in textures.inc
```

```
      scale 4       // scale by the same amount in all directions
    }
    finish { Shiny } // pre-defined in finish.inc
}

cylinder{
<-1200,-1200,-2800><-1200,-500,-2800>50
pigment {
      DMFWood5      // pre-defined in textures.inc
      scale 4       // scale by the same amount in all directions
    }
    finish { Shiny } // pre-defined in finish.inc
}

cylinder{
<-1200,-1200,-2200><-1200,-500,-2200>50
pigment {
      DMFWood5      // pre-defined in textures.inc
      scale 4       // scale by the same amount in all directions
    }
    finish { Shiny } // pre-defined in finish.inc
}

//objeto
sphere {
 <-2000, -150, -2400>, 250
 texture { Blue_Sky3 }
 finish {reflection {.1}}
}
```

```
//Rack
box{
 <-4000,-1000 ,3000 ><0,-1200 ,1500 >
 pigment {
     DMFWood4      // pre-defined in textures.inc
     scale 4       // scale by the same amount in all directions
    }
    finish { Shiny } // pre-defined in finish.inc
}

//TV
union{

 box{
  <-2500,-1000,2500 ><-1000,-300,1800>
  pigment {
      DMFWood5      // pre-defined in textures.inc
      scale 4       // scale by the same amount in all directions
     }
     finish { Shiny } // pre-defined in finish.inc
 }

//tela
box{
<-2300,-400,1800><-1300,-900,1799>
 pigment{color DimGrey}
 finish {reflection {.1}}
}

//botões
```

```
cylinder{
 <-1150,-900,1770><-1150,-900,1800>50
}

cylinder{
 <-1150,-700,1770><-1150,-700,1800>50
}

cylinder{
 <-1150,-500,1770><-1150,-500,1800>50
}

//antenas
sphere{
 <-1750,-300,2150>150
 }

cylinder{
 <-1750,-300,2150><-2000,200,2200>10
 pigment {color Gray30}
}

cylinder{
 <-1750,-300,2150><-1500,200,2250>10
 pigment {color Gray30}
 }
}

//Armário
box{
```

```
 <4000,-1200,-3000><2500,1100,-500>
 pigment {
     DMFWood3       // pre-defined in textures.inc
     scale 4        // scale by the same amount in all directions
    }
}

//Banquinho
union{

 cylinder{
  <2500,-600,2000><2500,-650,2000>500
  pigment {color Yellow}
 }

 cylinder{
  <2500,-650,2400><2500,-1200,2400>30
   pigment {color Yellow}
 }

cylinder{
 <2500,-650,1600><2500,-1200,1600>30
  pigment {color Yellow}
}

cylinder{
 <2900,-650,2000><2900,-1200,2000>30
  pigment {color Yellow}
}
 cylinder{
```

```
  <2100,-650,2000><2100,-1200,2000>30
   pigment {color Yellow}
 }

 box{
 <2900,-610,2450><2850,50,2000>
  pigment {color Yellow}
   }
}
```

ANEXO B - Algoritmo de retificação da imagem omnidirecional

```
% im = imagem omnidirecional
% h = altura em pixel da imagem carteziana
% L = largura em pixel da imagem carteziana
% y0 = posiçao vertical do centro da imagem omnidirecional
% x0 = posiçao horizontal do centro da imagem omnidirecional

function [imp]=unwrap(im,h,L,y0,x0);
[n,m]=size(im)
imp=0*ones(h,L);
for i=1:L
  i;
  ii=i-1;
  teta=2*pi*ii/(L-1);
  j=0:h-1;
  r = j;
  x=round(r*cos(teta)+x0);
  y=round(r*sin(teta)+y0);
  ind=(x'-1)*n+y';
  imp(1:h,i)=im(ind);
end
iu = imp;
iu=flipud(iu);
imshow(iu);
imp=iu;
```

ANEXO C - Algoritmo de correspondência entre pontos

```
% im2 = imagem externa;
% janv = meio altura vertical da janela
% janh = meia largura horizontal da janela
% c0 = custo de oclusao
% mindisp = disparidade minima
% maxdisp = disparidade maxima

function mate=corrtot(im1,im2,janv,janh,c0,mindisp,maxdisp);
clf
h=3;

% suaviza imagens originais
h = fspecial('gaussian',3,0.75);

im1 = filter2(h,im1,'same');
im2 = filter2(h,im2,'same');

[n,m] = size(im1);
mate = 0*ones(n,m);
nl = 1:n;

for ej=janh+1:m-janh
   for i1=janv+1:n-maxdisp
     % obtém janela da imagem interna
     jan1 = janela(im1,janv,janh,i1,ej);

     % verfica se janela é flat
     janmax = max(max(jan1));
     janmin = min(min(jan1));
     difmaxmin = janmax - janmin;
```

```
% se janela não for flat procura correspondetes
if difmaxmin>0.005
  % inicia busca com imagem interna como referência
  % determina limites de  busca na imagem externa
  ini = i1 - mindisp;
  if ini<janv+1
    ini = janv+1;
  end
  ifim = i1 + maxdisp;
  if ifim>n-janv
    ifim = n-janv;
  end

  % calculo da SDA
  crint = 10*ones(n,1);
  for i2=ini:ifim
    jan2 = janela(im2,janv,janh,i2,ej);
    crint(i2) = sum(sum(abs(jan1 - jan2)));
  end

  % Obtém o valor e posição do mínimo SDA
  [minimoint,iminint] = min(crint);

  % se o mínimo for menor do que o limite de oclusão aceita o correspondente
  % e realiza checagem pela busca com imagem interna como referência
  if minimoint<c0
    mate(i1,ej) = iminint;

    % checa o correspondente pela busca com imagem externa como referência
    jan2 = janela(im2,janv,janh,iminint,ej);
```

```
% determina limites de busca na imagem interna
ini = iminint - maxdisp;
if ini<janv+1
  ini = janv+1;
end
ifim = iminint + mindisp;
if ifim>n-janv
  ifim = n-janv;
end

% calculo da SDA
crext = 10*ones(n,1);
for in1=ini:ifim
  jan1 = janela(im1,janv,janh,in1,ej);
  crext(in1) = sum(sum(abs(jan1 - jan2)));
end

% Obtém o valor e posição do mínimo SDA
[minimoext,iminext] = min(crext);

% se o mínimo for igual da busca pela imagem interna aceita-se o
correpondente
% senão o pixel fica sem correspondente
if iminext~=i1
  mate(i1,ej) = -1;
  [ej,i1,difmaxmin,minimoint,iminint]
  [minimoext,iminext,mate(i1,ej)]
else
  [ej,i1,difmaxmin,minimoint,iminint]
```

```
            [minimoext,iminext,mate(i1,ej)]
          end
        else
          mate(i1,ej) = -2;
          [ej,i1,difmaxmin,minimoint,iminint]
        end

      else
        mate(i1,ej) = 0;
        [ej,i1,difmaxmin,mate(i1,ej)]
      end

    end
end
```

ANEXO D - Algoritmo de triangulação

```
function XYZ=triangulacao(mate);

x=1;
y=2;

%parâmetros do espelho
ci=135.1501;
ai=123.7943;
ce=210.5443;
ae=193.6103;
f=25;
lamb=3.5067*2*pi/360;
zfe=420.3002;
zfi=270.3002;
rfe=25.7558;

%calcula os raios ro internos e externos
raios=calcularaios(mate);

[n,m,o]=size(raios);

%calculo do angulo de elevaçao interno
alfai=((ci-ai)/(ci+ai));
alfae=((ce-ae)/(ce+ae));

for j=1:m
   for i=1:n
      if (raios(i,j,1)~=0 & raios(i,j,1)~=-1 & raios(i,j,1)~=-2)
         %variaveis auxiliares
         a=f+sqrt((f^2)+((raios(i,j,1))^2));
```

```
        b=2*alfai*(raios(i,j,1));
        c=((alfai^2)*(a^2))-((raios(i,j,1))^2);

        %angulo de elevaçao (tangente)
        fii =((a*b)/c);

        %calculo do angulo de elevaçao externo
        rext1=(raios(i,j,2))*cos(lamb)-(f)*sin(lamb);
        f1=(f)*cos(lamb)+(raios(i,j,2))*sin(lamb);

        %angulo de elevaçao sem levar em conta o angulo lambida(tangente)
        %variaveis auxiliares
        a=f1+sqrt((f1^2)+(rext1^2));
        b=2*alfae*rext1;
        c=((alfae^2)*(a^2))-(rext1^2);

        fierot=((a*b)/c);
        fie=(fierot-tan(lamb))/(1+(fierot*tan(lamb)));

        %calculo da posicao
        rp=(fii*((fie*(zfi-zfe))-rfe))/(fie-fii);
        zp=((zfe*fie)-(zfi*fii)+rfe)/(fie-fii);
        XYZ(i,j,x)=rp;
        XYZ(i,j,y)=zp;
    end
  end
end
```

```
function raio = calcularaios(mate);

[n,m]=size(mate);
rmax=3.5945;
Pmax=512;

%raio(i,j,1)-->interno
%raio(i,j,2)-->externo

for j=1:m
   for i=1:n
      if (mate(i,j)~=0 & mate(i,j)~=-1 & mate(i,j)~=-2)
         raio(i,j,1)=(287-i+1)*(rmax/Pmax);
         raio(i,j,2)=(512-mate(i,j)+1)*(rmax/Pmax);
      else
         raio(i,j,1)=mate(i,j);
         raio(i,j,2)=mate(i,j);
      end
   end
end
```

ANEXO E - Algoritmo que calcula a menor distância

```
%Calcula menor distancia
function x=dist(XYZ);

[n,m,o]=size(XYZ);
d=10000000000;  %inicializaç~ao da variavel
r = 0*ones(n,m);
for j=1:m
   for i=1:n
    if(XYZ(i,j,1)~=0 & XYZ(i,j,1)~=-1 & XYZ(i,j,1)~=-2)
      r(i,j)=sqrt(XYZ(i,j,1)^2+XYZ(i,j,2)^2);
    end
   end
end

for j=1:m
  for i=1:n
    if(r(i,j)~=0 & r(i,j)~=-1 & r(i,j)~=-2)
       if(d>r(i,j))
          d=r(i,j);
          lin=i;
          col=j;
       end
    end
  end
end

x(1)=d;
x(2)=lin;
x(3)=col;
```

REFERÊNCIAS BIBLIOGRÁFICAS

CABRAL, E.L.L. Omnidirecional Stereo Vision With a Hiperbolic Double Lobed Mirror. São Paulo, não publicado, 2003.

CABRAL, E.L.L. Sistema de Visão Estéreo Omnidirecional Com Espelho Duplo de Perfil Hiperbólico. Relatório enviado a FAPESP, São Paulo, 2003.

CABRAL, E. L. L.; SOUZA, J. C. Three Dimensional Reconstruction With An Omnidirectional Stereo Vision System. São Paulo, não publicado, 2004.

CHAL, J. S.; SRINIVASAN, M. V. Reflective Surfaces for Panoramic Imaging. **Applied Optics**, v.36, n.31, p.8275-8285, 1997.

FIELDING, G.; KAM, M. Weighted Matchings for Dense Stereo Correspondence. **Pattern Recognitio**, 33, pp1511-1524, 2000.

FU, K. S.; GONZALEZ R.C.; LEE C.S.G. **Robotics**: Control, Sensing, Vision, and Intelligence. New York : McGraw-Hill, 1987. Cap 7. p. 313-323: Low-Level Vision.

JUNIOR, V. G. **Sistema de visão omnidirecional aplicado no controle de robôs móveis**. 2003, 104 p., Dissertação (Mestrado), Escola Politécnica, Universidade de São Paulo. São Paulo, 2003.

NALWA, V.S. **A guided tour of computer vision.** Addison-Wesley Pub Co, 1993. Cap 7. p. 217-240: Stereo.

POLI, M. **Visão Computacional Aplicada ao Futebol de Robôs**. 2001, 72 p., Dissertação (Graduação), Escola Politécnica, Universidade de São Paulo. São Paulo, 2001

SHIRAI, Y. **Three-Dimensional computer vision.** Springer Verlag, 1987. Cap 7. p. 122-140: Stereo Vision.