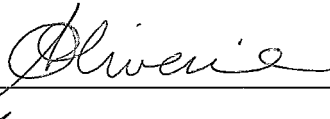


ESCANEAMENTO TRIDIMENSIONAL DE BAIXO CUSTO
USANDO LUZ ESTRUTURADA

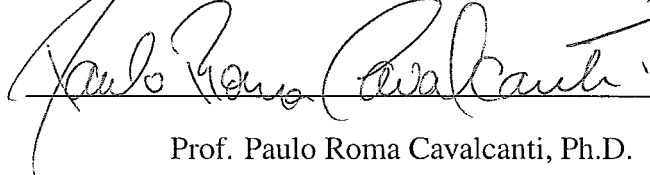
Fábio Alberto Lima Franco

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

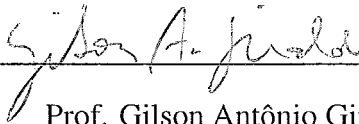
Aprovada por:



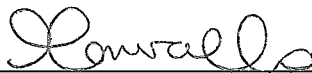
Prof. Antônio Alberto Fernandes de Oliveira, D.Sc.



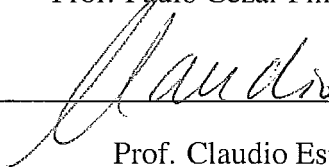
Prof. Paulo Roma Cavalcanti, Ph.D.



Prof. Gilson Antônio Giraldo, D.Sc.



Prof. Paulo César Pinto Carvalho, D.Sc.



Prof. Claudio Esperança, Ph.D.

RIO DE JANEIRO, RJ - BRASIL

MARÇO DE 2003

FRANCO, FÁBIO ALBERTO LIMA

Escaneamento Tridimensional de Baixo
Custo Usando Luz Estruturada [Rio de Janeiro]
2003

IX, 79 p., 29,7 cm, (COPPE/UFRJ, M.Sc.,
Engenharia de Sistemas e Computação, 2003)

Tese – Universidade Federal do Rio de Ja-
neiro, COPPE

1 – Escaneamento 3D

2 – Luz Estruturada

3 – Shape from Shading

I. COPPE/UFRJ II. Título (série)

Dedicatória:

Ao Professor Antônio Oliveira.

Agradecimentos:

Aos colegas Antônio Lopes e Rodrigo da Silva Moreira pelo apoio imprescindível na fase de *debug*.

Ao Professor Cláudio Esperança por ser o professor mais paciente do mundo.

Ao Professor Paulo Roma pelas infinitas caronas e altas discussões sobre o país.

E por último, mas não menos importante, ao professor Antônio Oliveira. Por tudo.

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

ESCANEAMENTO TRIDIMENSIONAL DE BAIXO CUSTO USANDO LUZ ESTRUTURADA

Fábio Alberto Lima Franco

Março/2003

Orientador: Antônio Alberto Fernandes de Oliveira

Programa: Engenharia de Sistemas e Computação

Esta tese apresenta um sistema de escaneamento tridimensional de baixo custo e as técnicas que se utilizam do conceito de luz estruturada usados na obtenção de nuvens de pontos de superfícies e objetos. Os métodos abordados incluem tres diferentes tipos de algoritmos de codificação: Gray-code, Phase-shifting e codificação por cor. Cada tipo de implementação é abordada resultando em um quadro comparativo dessas técnicas e na demonstração de seus respectivos resultados.

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

LOW COST 3D SCANNING USING STRUCTURED LIGHT

Fábio Alberto Lima Franco

Março/2003

Advisor: Antônio Alberto Fernandes de Oliveira

Department: Computing and Systems Engineering

This thesis presents a 3D scanning system using low-cost hardware and the techniques that use the structured light concept in the acquisition of surfaces and objects cloud points. The studied methods include three different types of coding algorithms: Gray-code, Phase-shifting and color code. Each implementation is analyzed resulting in a comparative survey of these techniques and in the presentation of its corresponding results.

Sumário

1	Introdução	1
2	Estado da Arte	5
2.1	Métodos de escaneamento usando luz estruturada	6
2.2	Outros métodos	9
2.3	Obtenção de modelos digitalizados	11
3	O problema da obtenção de pontos em 3D	13
3.1	Triangulação ativa	15
3.2	Condições de projeção	18
4	Sistema de escaneamento 3D usando luz estruturada	20
4.1	Calibração	21
4.1.1	Parâmetros intrínscos	22
4.1.2	Parâmetros extrínscos	25
4.2	Codificação	26
4.2.1	Gray-code	26
4.2.2	Phase-shifting	28
4.2.3	Codificação por cor	32
5	Implementação	34
5.1	Aquisição de imagens	35
5.2	Problemas inerentes ao equipamento	36
5.2.1	Questões vinculadas ao projetor	37
5.2.2	Questões vinculadas à câmera	40
5.3	Sistema implementado	42
5.3.1	Filtros	44
5.3.2	Visualização de <i>range-maps</i>	57

6	Resultados	61
6.1	<i>Gray-code e Phase Shifting</i>	64
6.2	Codificação Colorida	66
6.3	Gray-code, versão final	69
6.4	Análise comparativa com outros sistemas	72
7	Conclusões e trabalhos futuros	74

Lista de Figuras

2.1	Exemplo de escaneamento	7
2.2	<i>Gray-code</i> sobre objeto	8
2.3	Equipamento para 3D <i>Scan</i>	9
3.1	<i>Setup</i>	13
3.2	Definição dos <i>pixels</i> do modelo	14
3.3	Triangulação	15
3.4	Sistema de coordenadas	17
3.5	Oclusão	18
4.1	Tabuleiro	21
4.2	Diagrama MATLAB	25
4.3	Padrão <i>Gray-code</i>	27
4.4	Padrão <i>Phase-shifting</i>	29
4.5	Codificação com cores	32
4.6	Escaneamento com cores	33
5.1	Posição inclinada de projeção	37
5.2	<i>Broadening</i>	38
5.3	Padrão radial de luz	39
5.4	Identificação errônea	40
5.5	<i>Outliers</i>	41
5.6	Sistema implementado	43
5.7	Gráfico do código	45
5.8	Codificação sem filtragem	48
5.9	Codificação com filtragem	48
5.10	<i>Top-hat</i>	50
5.11	Detalhe de resultado em <i>Phase-Shifting</i>	51
5.12	Transição entre duas faixas	53

5.13	Objeto sob padrão colorido	54
5.14	Imagem capturada	55
5.15	Detecção de sombras e penumbras	57
5.16	Reconstrução errônea do método de cor	58
5.17	Exemplo de visualização de objeto escaneado	58
5.18	Visualizações	59
5.19	Visualização em um sistema de modelagem	60
6.1	Resultados iniciais	64
6.2	Resultados iniciais	65
6.3	Resultados finais 1	66
6.4	Resultados finais 2	67
6.5	Resultados finais 3	67
6.6	Resultados finais 4	68
6.7	Resultados finais 5	68
6.8	Resultados finais 6	68
6.9	Resultados finais 7	70
6.10	Resultados finais 8	70
6.11	Resultados finais 9	70
6.12	Resultados finais 10	71
6.13	Resultados finais 11	71
6.14	Resultados finais 12	72

Lista de Tabelas

6.1	<i>Gray-code e Phase Shifting</i> - resultados finais	65
6.2	Codificação colorida - resultados finais	66
6.3	<i>Gray-code</i> - resultados finais	69

Capítulo 1

Introdução

A aquisição de superfícies e formas por computador, também conhecida como escaneamento tridimensional (*3D scanning*), está se tornando cada vez mais importante comercialmente, com aplicações nas mais diversas áreas, incluindo não só a computação gráfica, mas também a engenharia reversa, modelagem de faces [1], o controle de qualidade de peças industriais [2], animação e cinema, escultura [3], medicina [4] e a arqueologia [5].

Na verdade, a aquisição de pontos de uma superfície é um problema clássico da Visão Computacional, já tendo sido objeto de estudo de inúmeras pesquisas com o propósito de se desenvolver algoritmos e técnicas que gerem resultados mais robustos e no menor tempo possível (tempo real).

As diversas implementações existentes utilizam técnicas passivas ou ativas para a obtenção de pontos em 3D. Essa taxonomia de sistemas óticos de obtenção de modelos digitais foi proposta por Curless [6]. As técnicas ativas envolvem os sistemas que se valem de triangulação, estereoscopia e interferometria (Moiré e holografia). Já as técnicas passivas incluem aquisição de formas a partir de sombras ou a partir de silhouetas. Algumas dessas soluções vinculam algoritmos específicos a um *hardware* adaptado para o escaneamento de objetos. O avanço no desenvolvimento desse tipo de *hardware* tende a padronizar o processo de escaneamento, registro e alinhamento dos dados obtidos, integração de mapas de profundidade

(*range maps*)¹ e otimização das malhas obtidas.

Um aspecto fundamental, quando se discute sistemas óticos de escaneamento 3D, é o de que todos estes sistemas são assim classificados por incluírem um equipamento emissor, por exemplo um projetor multimídia, que produz algum tipo de luz estruturada (*light pattern*) e um equipamento receptor, ou câmera, que captura imagens do objeto sendo escaneado sob o efeito desta iluminação especial, ou padrão de projeção, como chamaremos daqui em diante.

De um modo geral, todo sistema de escaneamento possui um conjunto de etapas bem definidas que inclui a calibração do sistema, a fase de aquisição da nuvem de pontos e a fase de construção do modelo digital final. É importante salientar que alguns autores definem reconstrução como sendo todo o processo de escaneamento acima descrito [8] [9] enquanto outros definem reconstrução como a etapa de modelagem 3D do objeto [10] [11] [12]. A razão desta diferenciação reside no fato de que muitos trabalhos na literatura abordam somente a parte de registro da nuvem de pontos, ou seja, o método pela qual diferentes *range maps* são transformados para o mesmo sistema de coordenadas. A partir deste ponto, muitos consideram as etapas como de modelagem do objeto e não mais de escaneamento. Nosso trabalho utiliza a definição de reconstrução como sendo todo o processo para a obtenção do modelo digital, desde a fase de escaneamento até a geração da malha final.

Nos últimos anos, o aumento da performance dos sistemas 3D se deve às técnicas que utilizam luz ativa (estruturada ou não-estruturada). Esses algoritmos usam um simples método geométrico de triangulação para o cálculo dos pontos escaneados e, como em todos os outros métodos de escaneamento, possui três importantes fatores a serem avaliados: robustez, precisão e velocidade de aquisição.

Define-se robustez como a capacidade de um sistema de escanear pontos de forma consistente, evitando fatores que causem problemas de aquisição como refletância do objeto, sombras ou textura. Esse fator é extremamente dependente do *hardware* utilizado [8].

¹Usaremos neste trabalho o conceito proposto por Bernardini [7], que define *range image* ou *range map* como uma matriz de valores de profundidade para pontos de um objeto adquiridos de um determinado ponto de vista.

Precisão é o valor que define o desvio, ou erro da distribuição de distâncias entre o ponto medido e o valor real do ponto escaneado. Quanto maior a precisão do sistema, mais próximos das medidas reais do objeto estarão os modelos escaneados [7]. Vale ressaltar que resolução, em escaneamento 3D, não significa necessariamente precisão. A palavra resolução é normalmente usada para se definir o tamanho do quadro de captura do equipamento de aquisição (câmera) ou de projeção (projektor multimídia). Por exemplo, alguns sistemas são projetados para escanear objetos minúsculos, com uma resolução na ordem de grandeza dos microns (*sub-pixel*). Porém, os erros proporcionais às medidas dos objetos escaneados podem ser grandes [13].

A velocidade de aquisição é a principal característica de sistemas de escaneamento de baixo custo. Os resultados de maior qualidade conseguidos pelos sistemas mais precisos são gerados, em sua maioria, em detrimento da velocidade de aquisição. Quanto maior o número de pontos escaneados por segundo (atual parâmetro de tempo usado), mais rápido é o sistema. Alguns sistemas capturam pontos rapidamente, mas são lentos na etapa de cálculo destes [14], enquanto outros obtêm os pontos mais lentamente [3], porém com maior precisão.

Outra característica importante a ser considerada quando se avalia um sistema de escaneamento tridimensional é a quantidade de imagens do objeto escaneado obtidas para a geração do modelo digital final. Cada *range map* precisa de um conjunto de imagens para o cálculo de seus pontos. Alguns sistemas, como será visto no capítulo a seguir, buscam recuperar a geometria de uma superfície a partir de apenas uma imagem [15], mas, para a reconstrução total de um modelo são necessários vários *range maps* (várias vistas) de um objeto. Assim, neste trabalho, será analisado o número de imagens necessárias para a obtenção de cada *range map*. Quanto menor o número de imagens, mais complexa e imprecisa é a reconstrução dos modelos.

Grande parte da atual literatura de escaneamento 3D está voltada para os problemas de registro e integração de *range maps* e de modelagem final do objeto escaneado. Para tanto, esses trabalhos assumem que a etapa de aquisição de pontos já gerou resultados de alta resolução e precisão [10] [12]. Tais implementações visam corrigir apenas os problemas de robustez e de otimização da malha final do modelo e assim, se valem de equipamentos de alto custo, específicos para escaneamento 3D. Nosso trabalho, porém, busca outra abordagem, também bastante

comum em Visão Computacional, que consiste na análise da etapa de aquisição da nuvem de pontos do modelo, utilizando equipamentos considerados de baixo custo, através de técnicas de iluminação ativa e luz estruturada. Os problemas decorrentes deste tipo de abordagem estão, em sua grande maioria, diretamente relacionados ao *hardware* usado e, por isso, os resultados normalmente apresentados na literatura são pouco robustos ou precisos.

No capítulo 2 será apresentado o estado da arte em escaneamento tridimensional e seus métodos mais eficientes. No capítulo 3 será mostrado o problema geométrico de obtenção de pontos em 3D. O capítulo 4 apresenta as etapas de escaneamento com luz estruturada executadas neste trabalho e os três principais padrões de projeção utilizados para esse fim. No capítulo 5 a implementação do sistema de reconstrução criado é explicada em seus detalhes e os seus resultados são analisados no capítulo 6. O capítulo 7 é reservado aos trabalhos futuros e às conclusões.

Capítulo 2

Estado da Arte

O desenvolvimento de sistemas de escaneamento 3D está diretamente relacionado com o equilíbrio entre a velocidade de aquisição [16], a robustez, a facilidade de uso e a precisão [17].

O surgimento de equipamentos comerciais de escaneamento de alta-precisão (por exemplo, *Cyberware Scan*) tem contribuído para o aumento da resolução de escaneamento e diminuição de erros, mas muitos destes sistemas ainda são caros se comparados às soluções de baixo custo também robustas [18]. Além disso, esses equipamentos são, de uma forma geral, de uso complicado e sua velocidade de aquisição é baixa [3]. Os sistemas de baixo custo, por outro lado, foram largamente difundidos por serem fáceis de usar e exigirem um *hardware* simples (câmeras de vídeo comuns e projetores multimídia)[14]. Porém, esses sistemas são altamente dependentes das condições do próprio objeto (textura e cor) e certamente não possuem a mesma precisão dos equipamentos comerciais.

Entre os sistemas considerados de baixo custo, algumas implementações se mostraram mais eficientes que outras. De um modo geral, os sistemas que utilizam luz estruturada conseguiram apresentar resultados mais robustos e ao mesmo tempo se mostraram computacionalmente mais eficientes do que outros tipos de implementações, como aquelas que utilizam sombras ou disparidade, por exemplo. Recentemente foram implementados sistemas que combinam alta precisão com um escaneamento em tempo real [16][19].

Porém, a melhoria na robustez e na velocidade dos sistemas de luz estruturada só pôde ser conseguida devido ao surgimento de equipamentos especiais para o processo de escaneamento. Esse tipo de *hardware* é especialmente projetado para fazer captura sincronizada em alta resolução. Modelos de projetor como o GMD 3000 Ganymed ou o DMD (Digital Mirror Device) usado em sincronia com a câmara de captura permitem que se escaneie objetos em resoluções superiores a 1200x800 pixels com precisão de 500nm ou até de 1000nm [20]. Ainda que especiais esses equipamentos são considerados de baixo custo se comparados a outros modelos de sistemas de escaneamento comerciais como o Cyberware *Scanner*[21], o Opton Moiré *Scanner*, o ATOS *Scanner*[22], o 3D *Digitizer*[23] ou o RS 2200 *Ranger*[24].

2.1 Métodos de escaneamento usando luz estruturada

Existem várias formas de se usar luz estruturada para o processo de escaneamento. Enquanto algumas implementações utilizam padrões de Moiré [13], outras projetam padrões binários (preto e branco) sobre o objeto com resolução crescente.

Perona [9] utiliza o conceito de "luz fracamente estruturada", projetando a sombra de um aparato fino vertical sobre o objeto. Esse método usa a idéia de coerência espaço-temporal, que consiste na determinação do instante discreto \mathbf{T} de passagem da borda da sombra deste objeto fino sobre um ponto do objeto escaneado, para extrair sua profundidade. O instante \mathbf{T} é representado por cada quadro filmado (*frames*) da sombra passando pelo objeto, ou seja, para cada quadro é determinado um plano de sombra que corta o objeto. Enquanto a sombra é projetada sobre a cena, cada pixel (i, j) da imagem têm seu valor de brilho alterado do valor máximo I_{max} (sem sombra do aparato) para o valor mínimo I_{min} (sob a sombra do aparato) e novamente para o valor máximo. A borda da sombra pode, assim, ser localizada no espaço-tempo, onde $I(i, j, t)$ passa pelo *threshold* da imagem definido como a média de I_{max} e I_{min} . Esse método, além de preservar as discontinuidades abruptas da cena, facilita a localização de pontos de borda do objeto e de áreas oclusas porque cada plano de sombra determinado em cada quadro pos-

sui um índice inteiro correspondente (de 1 para o primeiro quadro até o n-ésimo quadro), ou seja, se em nenhum instante o brilho do objeto se alterou, este estava em área oclusa ou sombreada.

Já Rusinkiewicz [19] projeta um padrão em alta resolução e captura em vídeo o deslocamento desse padrão sobre o objeto. Em seguida ele identifica a alternância de cor de uma faixa (constituída por quatro colunas pretas e brancas alternadamente) em quatro quadros consecutivos. Cada um desses quadros visualizaria um padrão projetado diferente e esses padrões são de tal forma dispostos que se torna possível indexar com precisão cada faixa sobre o objeto se esta respeitar a disposição de alternância preto-branco ao longo de 4 quadros seguidos. Assim, cada faixa indexada permite a recuperação da profundidade dos pontos da superfície do objeto.

Independente da forma de escaneamento, todos os sistemas que utilizam luz estruturada baseiam-se na codificação das faixas que compõem os padrões binários ou coloridos que são projetados sobre um objeto de forma a extrair a profundidade de seus pontos.

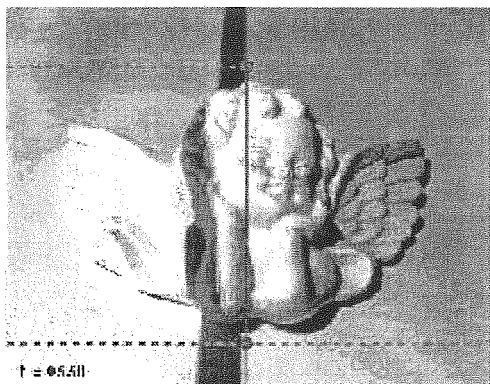


Figura 2.1: Exemplo de escaneamento com luz estruturada

A forma mais conhecida de implementação é aquela que utiliza o método de *Gray-code*. O método consiste em se projetar padrões verticais paralelos e equidistantes, alternadamente pretos e brancos, sobre um objeto, em variadas resoluções, onde a resolução posterior é sempre o dobro da anterior. Cada faixa projetada recebe uma codificação binária única que a distingue da anterior e da posterior em um *bit*. Assim, é possível discernir cada trecho do objeto a partir da deformação que este causa nas faixas. De um modo geral, o método *Gray-code* utiliza 2^n códigos

diferentes para descrever a profundidade do objeto, onde n é o número de padrões projetados. Por exemplo, se forem projetados 8 padrões temos 256 codificações possíveis para o objeto, ou seja, 256 faixas. Esse tipo de projeção é abordado no capítulo 4.2.1.



Figura 2.2: Exemplo de padrão *Gray-code* sobre um objeto

Um método complementar do *Gray-code* é o *Phase-shifting* que consiste em se substituir o último padrão projetado por 4 padrões de mesma resolução que se deslocam em $1/4$ de período em ao longo do eixo X . Esse deslocamento visa aumentar a resolução do escaneamento para o nível de sub-*pixel*, através da substituição do último *bit* de código por um número real (flutuante). Esse método é discutido no capítulo 4.2.2.

Outra forma de codificação por luz estruturada é que a que utiliza cores [25][26]. Com esse método, é possível aumentar o número de códigos indexados diminuindo o número de projeções de padrões a serem feitas. Isso ocorre porque a codificação das faixas pode ser feita de modo que em cada padrão projetado, estejam representados três níveis de resolução, cada um associado a um canal de cor diferente (RGB). Assim, podemos ter uma faixa vermelha codificada como 100, uma faixa verde como 010 e uma faixa azul por 001. Uma faixa branca é codificada como 111 e uma faixa amarela é codificada como 110, ou seja, dada uma cor P_j , a próxima cor P_{j+1} deve ser diferente em pelo menos um canal para que haja uma transição no código [26]. Dessa forma, usando 4 cores para se codificar o objeto, pode-se conseguir o mesmo número de codificações diferentes do método *Gray-code* com menos imagens: projetando apenas 4 padrões temos 4^4 faixas para 256 códigos diferentes.

Existem também muitas abordagens para os problemas encontrados no processo de escaneamento com luz estruturada. Esses problemas são tanto resultantes da codificação errônea induzida por sombras no objeto ou por sua textura, quanto da baixa resolução do próprio equipamento de aquisição usado. Tais problemas serão abordados respectivamente nos capítulos 4 e 5.

2.2 Outros métodos

Os métodos de escaneamento que não utilizam luz estruturada são variados tanto em relação aos algoritmos usados quanto na sua precisão, robustez, necessidade de equipamento ou velocidade de aquisição.

Destes métodos, os mais difundidos são aqueles cujo sistema se baseia no *laser* ou em raios infravermelhos. Esses sistemas são largamente utilizados na indústria. Existem hoje dezenas de sistemas integrados de escaneamento tridimensional por *laser* como o *Diff hand-held scanner* [27], o *Kreon Zephyr* [28], o *Roland LPX-250* [29] ou o mais conhecido, o *Cyberware Scan* [21]. Alguns escaneadores são portáteis, pesando menos de 500g e mesmo assim possuem capacidade de capturar até 16.000 pontos por segundo (por exemplo, *AXILA G-Scan* [30]). Em muitos casos, o *laser* ou infravermelho já são projetados com coordenadas relativas (x e y) conhecidas e o receptor ou câmera apenas registra o tempo de resposta de reflexão do raio. Todos esses sistemas possuem alta precisão e resolução.

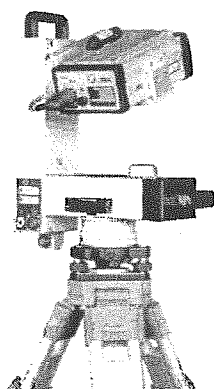


Figura 2.3: Equipamento de escaneamento automatizado

Também existem sistemas de escaneamento 3D que não utilizam nenhum tipo de projeção (laser ou de luz comum) como é o caso do Geometrix [31] ou do Cognitens [32] que usam reconstrução estereoscópica a partir das imagens de várias câmeras para adquirir grande quantidade de pontos em menos de 1 segundo. Os métodos de estereoscopia são altamente dependentes da calibração das câmeras utilizadas e se a quantidade de pontos a serem calculados for grande estes métodos são computacionalmente pouco eficientes.

Muitos algoritmos de *shape-from-shading* buscam recuperar a geometria de um objeto a partir de apenas uma imagem deste. Zhang [15] apresentou um excelente quadro comparativo destas técnicas, evidenciando as limitações claras de sistemas que implementam abordagens de minimização e propagação ou abordagens locais ou lineares.

Segundo Zhang: “A abordagem de minimização computa as soluções que reduzem uma função de energia calculada em toda a imagem. Essa função pode envolver restrições de brilho e outras restrições como a de serem suaves ou integráveis, ou ainda, de terem gradientes ou normais restritas”. O autor também comenta a abordagem de propagação que “...começa a partir de um simples ponto de referência ou um grupo de pontos da superfície onde a forma ou é conhecida ou pode ser determinada e forma única...e propaga a informação da forma através de toda a imagem” [15]. Esse processo, é claro, acumula erros, o que justifica os resultados de reconstrução pouco razoáveis encontrados na literatura, mesmo quando os objetos amostrados possuem uma geometria simples.

Atualmente existem aparelhos comerciais que capturam os pontos de um objeto com altíssima precisão (até 25 micron) em várias posições, alinham as nuvens de pontos obtidas, corrigem os erros de merging dos mapas de profundidade e imediatamente triangulam a malha que define esse objeto, gerando ao final do processo, um modelo com sombreamento e até textura (por exemplo, 3D *Digital Scan*) [33].

A existência de tal tipo de *hardware* comprova a robustez das atuais técnicas de escaneamento e, de um modo geral, eleva o estado da arte da obtenção de modelos digitais a um patamar que é altamente dependente do custo dos equipamentos usados.

2.3 Obtenção de modelos digitalizados

Nos últimos anos, o aumento da precisão dos equipamentos de escaneamento 3D transformou de forma radical as abordagens dadas ao tema.

A ênfase na aquisição de pontos encontrada ao tempo de trabalhos como [34] e [35], se apresenta bastante reduzida nos trabalhos mais recentes, em especial os que abordam a parte de implementação [19] [10] [3]. O enfoque na obtenção de modelos digitais tornou-se importante, pois devido ao aumento da precisão e robustez do processo de aquisição, a obtenção de um modelo completo de um objeto, em tempo real, se tornou viável. O alinhamento de diferentes *range maps* e a geração de malhas poligonais completas (sem perda de dados por oclusão ou sombreamento) passou a ser objeto de estudos mais intensos [36] [12] [37] [8].

Essas abordagens pressupõem um processo de aquisição que gere nuvens de pontos com poucas falhas e excelente resolução. Sistemas com essa capacidade, onde projetor e câmera, mesmo que considerados de baixo-custo na literatura, trabalham de forma integrada, são ainda muito caros (com valores atualmente entre \$7.500 a \$50.000)[38].

Ainda que os valores destes equipamentos possam diminuir com a difusão e comercialização em larga escala, os excelentes resultados apresentados pelos *hardwares* comerciais não são suficientes para determinar uma convergência única nas propostas de implementações futuras. Em especial os problemas de robustez causados pela cor, textura ou refletância dos objetos escaneados se mostram difíceis de serem solucionados, assim como as dificuldades inerentes do escaneamento de modelos de topologia complexa como a face humana.

Curless [8] definiu uma série de propriedades desejáveis para algoritmos de reconstrução tridimensional de superfícies. Entre essas propriedades, uma ainda se mostra difícil de ser respeitada: ausência de restrições topológicas. Para Curless, o algoritmo não deveria assumir que um objeto é de um determinado *genus*. Essa restrição força as implementações atuais, que têm como objetivo o escaneamento em tempo real, a integrarem totalmente os processos que compõem a reconstrução de modelos digitais totais, pois, para a visualização e análise rápida de um objeto de topologia qualquer é necessária uma outra propriedade chamada atualização incremental e ordenada, que consiste na captura, cálculo e registro sequenciais de

pontos que pertencem a *range maps* diferentes. Ou seja, enquanto um *range-map* é capturado, o anterior vai sendo acrescentado imediatamente ao modelo.

Ao contrário dos sistemas comerciais de alta precisão, os sistemas *off-the-shelf* geravam uma grande quantidade de erros. Jiang [39] por exemplo, adotou uma estratégia de minimização desses erros. Porém, atualmente, os sistemas de luz estruturada adotam, exatamente a idéia oposta [16], que consiste em não reduzir os erros cometidos na identificação dos padrões projetados, de forma que esses erros se tornem evidentes e só se calcule códigos onde se têm absoluta certeza. Assim, mesmo que a amostragem temporária de pontos seja pequena, o processo de aquisição pode ser feito de forma mais rápida (tempo real) pois elimina-se o custo computacional de se recalcular um possível erro ou de tentar reduzi-lo. Com isso, os sistemas precisam calcular apenas os pontos considerados corretos de um *range-map*. Se necessário, pode-se escanear novamente uma parte do objeto que foi pouco amostrada de forma a completar os espaços necessários (“buracos”) para a construção do modelo digital final. Desta forma os sistemas de baixo custo conseguiram acompanhar a evolução dos sistemas comerciais, permitindo a obtenção de modelos completos de forma robusta e rápida.

É importante notar, porém, que essas implementações consideradas de baixo custo também se deparam com os problemas dos equipamentos comerciais citados acima, o que dificulta ainda mais a criação de um sistema universal de escaneamento, capaz de escanear objetos de topologia, cor e textura variadas em tempo real.

Todavia, se considerarmos os excelentes resultados apresentados pelos métodos de escaneamento de baixo custo atuais, podemos entender claramente as razões que conduziram a um aumento significativo do surgimento de algoritmos e técnicas voltados para a parte de modelagem do objeto.

Apresenta-se hoje, portanto, uma clara tendência de integração da etapa de aquisição com os processos de registro de nuvens de pontos e criação da malha definitiva do modelo. As pesquisas recentes se mostram claramente voltadas para as questões de *merging* de *range-maps*, de triangulação em tempo real [19][26] e de mapeamento automático de textura dos objetos escaneados.

Capítulo 3

O problema da obtenção de pontos em 3D

A geometria ótica dos sistemas de escaneamento 3D que empregam luz estruturada segue um mesmo modelo para a obtenção de pontos chamado de princípio de triangulação ativa.

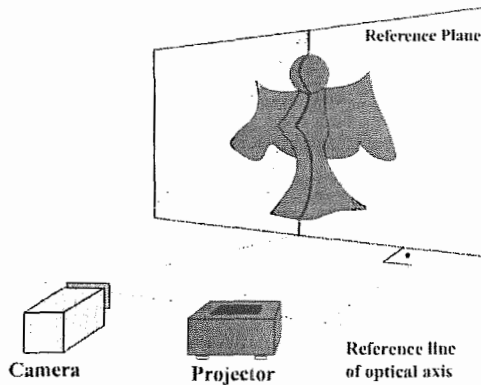


Figura 3.1: Posição dos equipamentos de escaneamento

Esse princípio baseia-se na existência de um projetor, cuja localização representaremos por E_p e de uma câmera, localizada em (E_c) . A figura acima mostra o esquema de posicionamento dos equipamentos de escaneamento. Os centros

ópticos da câmara e do projetor estão numa mesma linha horizontal, paralela ao plano de referência AR vertical. A distância dessa linha ao plano de referência é L e a existente entre a câmara e o projetor é D .

O objetivo deste posicionamento é o de facilitar o estabelecimento de uma correspondência entre os *pixels* da câmara e do projetor como mostrados na figura seguinte.

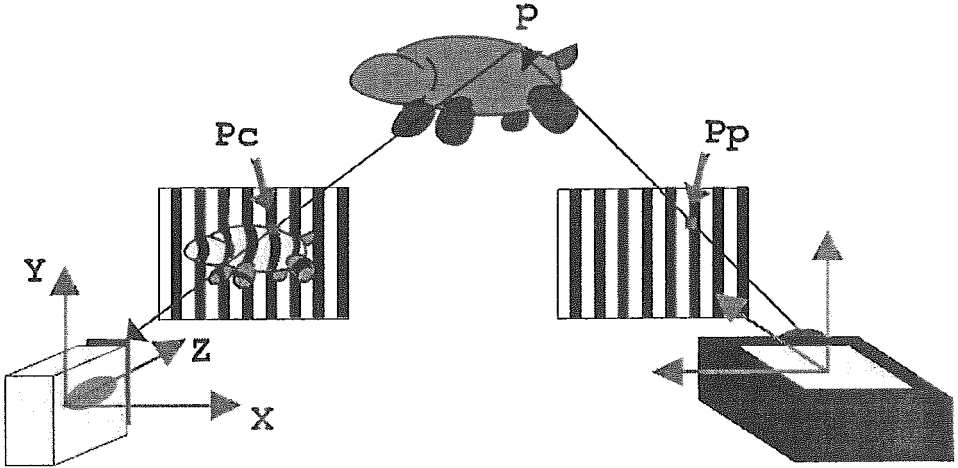


Figura 3.2: Relação entre os *pixels* do projetor e da câmara

Para tanto, costuma-se adotar um modelo para o projetor que é análogo ao da câmara. Esse modelo é chamado de *pin-hole* e estabelece um sistema de coordenadas local para o equipamento, com coordenadas x , y e z definidas de tal forma que, tanto o projetor quanto a câmara possam sofrer transformações lineares afim sem perderem a relação direta dos seus *pixels* projetados e visualizados.

Para descrever o processo pela qual a terceira dimensão do objeto é recuperada, precisamos não só das posições dos centros ópticos de ambos, projetor e câmara, definidos por O_p e O_c respectivamente, como também dos seus eixos ópticos, definidos por E_p e E_c , até o plano de referência. *Pixels* do padrão projetado sobre o plano são visualizados pela câmara com e sem a presença do objeto. Nesse último caso somente um plano vertical, o plano de referência, é visualizado. A profundidade de um ponto O sobre o objeto representado em um *pixel* p da imagem pode ser, assim, determinada pela diferença entre os *pixels* projetados associados a p nos dois casos, com e sem o objeto.

É importante salientar que o problema de obtenção dos vários pontos de um objeto através de iluminação ativa pode ser diferente se o sistema utilizado for baseado em duas ou mais câmeras ou na aquisição a partir de uma só imagem, mas a princípio a geometria do sistema ótico da câmera ou do projetor se baseia na sua posição relativa ao plano de referência onde o objeto se encontra e na forma como ambos, projetor e câmera “enxergam” esse ponto O .

3.1 Triangulação ativa

O método de triangulação ativa normalmente usado em sistemas de luz estruturada visa simplificar a determinação de um ponto sobre o objeto ao considerar que os eixos óticos tanto da câmera quanto do projetor são horizontais e que seus sistemas internos de referência sejam tais que os *pixels* gerados pelo projetor e visualizados pela câmera correspondam sempre a linhas verticais no plano de referência AR [2].

Muitos sistemas assumem que a câmera e o projetor são paralelos ao plano de referência, tornando epipolares as linhas da imagem, reduzindo assim o problema de correspondência de 2D para 1D. A figura seguinte mostra o esquema de triangulação usado para se determinar a profundidade de um ponto O sobre o objeto. Esse esquema de triangulação é o mesmo do sistema desenvolvido neste trabalho.

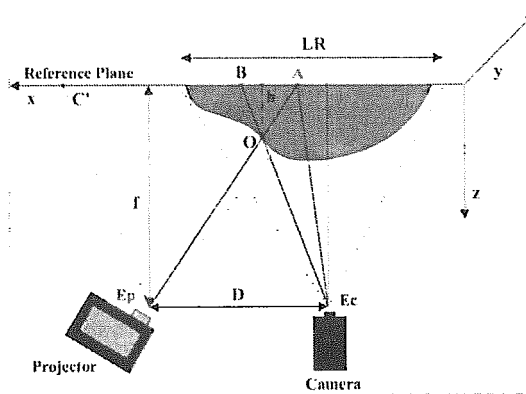


Figura 3.3: Triangulação ativa

Pela figura podemos determinar que:

a) \mathbf{O} é o elemento da superfície que representado pelo pixel (j,k) , enquanto \mathbf{A} é o ponto do plano de referência onde o raio incidente projetado em \mathbf{O} alcança este plano sem a presença do objeto.

b) Seja \mathbf{D} a distância entre $\mathbf{E_p}$ e $\mathbf{E_c}$, e \mathbf{L} a distância de ambos ao plano de referência \mathbf{AR} .

c) \mathbf{B} é o ponto no plano de referência onde a raia determinada por \mathbf{O} e $\mathbf{E_c}$ encontra o plano de referência, ou seja, a raia segundo a qual o ponto \mathbf{O} se projeta sobre o ponto do plano de projeção correspondente ao pixel (j,k) .

d) $\mathbf{C'}$ é o ponto onde a paralela da linha projetada $\mathbf{OE_c}$, passando por $\mathbf{E_p}$ atinge o plano de referência.

e) \mathbf{h} é a distância de $\mathbf{O}_{(j,k)}$ ao plano de referência.

Pela similaridade dos triângulos co-planares $[\mathbf{A}, \mathbf{E_p}, \mathbf{C'}]$ e $[\mathbf{A}, \mathbf{B}, \mathbf{O}]$, obtêm-se:

$$\frac{|\mathbf{B}, \mathbf{A}|}{|[\mathbf{C'}, \mathbf{A}]|} = \frac{h_{(j,k)}}{L} \quad (3.1)$$

Supondo que as projeções de todas as linhas no plano de referência tenham aproximadamente a mesma largura \mathbf{S} então, $|[\mathbf{B}, \mathbf{A}]| = \mathbf{S}^*s_{(j,k)}$, $|[\mathbf{C'}, \mathbf{B}]| = \mathbf{D}$ e $|[\mathbf{C'}, \mathbf{A}]| = \mathbf{S}^*s_{(j,k)} + \mathbf{D}$, e assim temos:

$$h_{(j,k)} = \frac{[\mathbf{S} \cdot s_{(j,k)} \cdot L]}{[\mathbf{S} \cdot s_{(j,k)} + D]} \quad (3.2)$$

onde $s_{(j,k)}$ é a distância das faixas na imagem.

Depois do cálculo de $\mathbf{h}_{(j,k)}$, as outras coordenadas de \mathbf{O} , $\mathbf{x}_{(j,k)}$ e $\mathbf{y}_{(j,k)}$, podem ser obtidas da seguinte maneira: sejam \mathbf{C}^\perp e \mathbf{O}^\perp as projeções ortogonais de \mathbf{C} e \mathbf{O} no plano de referência, a similaridade dos triângulos co-planares $[\mathbf{B}, \mathbf{O}^\perp, \mathbf{O}]$ e $[\mathbf{B}, \mathbf{C}^\perp, \mathbf{C}]$ mostrados na figura seguinte, determina que:

$$\frac{h_{(j,k)}}{L} = \frac{x_{(j,k)} - x_r(j,k)}{x_c - x_r(j,k)} = \frac{y_{(j,k)} - y_r(j,k)}{y_c - y_r(j,k)} \quad (3.3)$$

onde (x_c, y_c) são as coordenadas x e y de C . De forma a simplificar o cálculo, pode-se assumir que a câmera e o eixo Z são coincidentes, fazendo (x_c, y_c) .

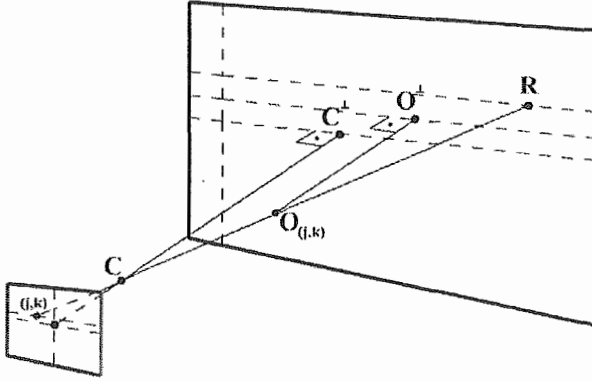


Figura 3.4: Eixo óptico da câmera

Considere agora FW e FH como as dimensões, horizontal e vertical, da porção do plano de referência vista pela câmera. A origem do sistema de coordenada é o centro desta porção, então:

$$x_r(j,k) = \frac{(K-1-k).FW}{K-1} - \frac{FW}{2} \quad (3.4)$$

$$y_r(j,k) = \frac{(J-1-j).FH}{J-1} - \frac{FH}{2} \quad (3.5)$$

onde J e K são, respectivamente, o número de linhas e colunas da imagem. Substituindo $x_{r(j,k)}$ e $y_{r(j,k)}$ pelas suas expressões, temos finalmente:

$$x_{(j,k)} = \frac{h_{(j,k)}}{L}.x_c + FW.(1 - \frac{h_{(j,k)}}{L}).(0.5 - \frac{k}{K-1}) \quad (3.6)$$

$$y_{(j,k)} = \frac{h_{(j,k)}}{L}.y_c + FH.(1 - \frac{h_{(j,k)}}{L}).(0.5 - \frac{j}{J-1}) \quad (3.7)$$

Assim, podemos determinar as coordenadas de \mathbf{O} . Deve-se observar que quando a câmera é posicionada perpendicularmente ao plano de referência, as primeiras parcelas de 3.6 e 3.7 se anulam, obtendo-se as expressões para $x_{(j,k)}$ e $y_{(j,k)}$ usualmente encontradas na literatura. [2].

A disposição do equipamento de escaneamento é, portanto, altamente restrita se forem considerados os parâmetros discutidos aqui. No entanto, essa restrição se torna ainda mais severa pelas limitações impostas pela qualidade do *hardware* disponível e, por isso, as condições necessárias para o cômputo dos pontos através do método de triangulação ativa são ainda mais difíceis de serem respeitadas, gerando problemas como será visto no capítulo 5.

3.2 Condições de projeção

Em praticamente todos os sistemas que utilizam luz estruturada, as condições de projeção influenciam diretamente o processo de aquisição de pontos de um objeto. Vários fatores têm efeito direto nessa obtenção dos pontos, como por exemplo, a auto-occlusão, o auto-sombreamento e a textura ou a cor do objeto. O problema da auto-occlusão e o auto-sombreamento do objeto é ilustrado na figura abaixo.

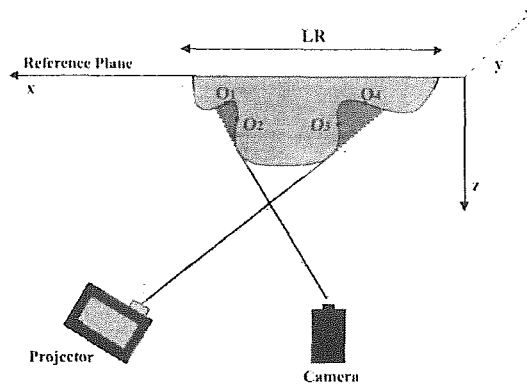


Figura 3.5: Problema de oclusão e sombreamento

Os pontos O_1 e O_2 são iluminados pelo projetor mas não podem ser vistos pela câmera e, portanto, sua profundidade não pode ser recuperada. Os pontos O_3 e O_4 são visualizados pela câmera mas se encontram em área de sombra provocada pela luz do projetor. Nesse caso, não se pode aferir nenhuma codificação com precisão. Por isso, nenhuma decodificação deve ser feita nestas regiões, o que leva ao surgimento de “buracos” no modelo digital final. Uma nova amostragem desses trechos sob sombra é necessária, obrigando assim que o objeto, ou o sistema formado pela câmera e o projetor, sofram uma rotação. Em ambos os casos, transformações lineares afim devem ser aplicadas para que os pontos obtidos estejam referenciados ao mesmo sistema de coordenadas.

Os casos onde a cor ou a textura do objeto impedem a aquisição correta de pontos podem ser solucionados de várias formas. Os padrões de um sistema que usa luz estruturada colorida não deve, obviamente, possuir a mesma cor do objeto (ver capítulo 5). Além disso, *thresholds* utilizados para identificar a cor das faixas projetadas sobre o objeto, precisam ser escolhidos considerando-se a cor deste.

Porções de objetos altamente reflexivos também são muito difíceis de serem escaneados usando luz estruturada. Nessa situação, mesmo escaneadores *laser* falham na identificação de pontos [3]. Procedimentos de interpolação podem ser usados para gerar valores de profundidade para esses pontos.

Outros fatores influenciam de forma indireta a aquisição dos pontos, em especial os problemas de foco e distorção provocados pela câmera (Ver capítulo 5), ou o problema da iluminação indireta do objeto pela luz ambiente do local de escaneamento (Ver 5.3.1).

Capítulo 4

Sistema de escaneamento 3D usando luz estruturada

O escaneamento através de luz estruturada pode ser executado através de várias técnicas diferentes e que serão apresentadas nos capítulos a seguir. O presente trabalho se utilizou de três dessas técnicas para testar os seus níveis de robustez e precisão e, como será apresentado, identificar os vários fatores relacionados ao *hardware* que influenciam diretamente a qualidade de aquisição de pontos dos objetos escaneados.

Como já foi visto, todo sistema de escaneamento que utiliza luz estruturada se baseia na relação entre os *pixels* gerados por um projetor, e como estes são vistos por uma câmera, para criar uma correspondência (coerência espacial) que permita o cálculo da profundidade de um ponto sobre um objeto escaneado. Para tanto, duas etapas são fundamentais no processo de obtenção desses pontos resultantes do método de triangulação ativa: calibração do sistema e codificação dos padrões projetados.

4.1 Calibração

O sistema de calibração usado neste trabalho foi adaptado do modelo apresentado por Jean Yves Boughet [18] que é bastante similar ao modelo desenvolvido por Heikillä [40]. Esses modelos determinam parâmetros intrínsecos e extrínsecos de câmera que são utilizados em vários sistemas de escaneamento com luz estruturada. Para tanto, em nossa etapa de calibração, foi usado o software desenvolvido por Boughet [41], o *Camera Calibration Toolbox*, o que permitiu um ajuste mais preciso das medidas adquiridas manualmente (com uma régua comum).

O *Camera Calibration Toolbox* exige apenas um tabuleiro quadriculado, utilizado como referência para a extração dos parâmetros. Após a identificação dos vértices das quadrículas do tabuleiro, são computados os erros referentes às distorções tangenciais e radiais da imagem e o programa indica os valores de distância focal, as coordenadas do ponto principal da imagem e os parâmetros intrínsecos do sistema.

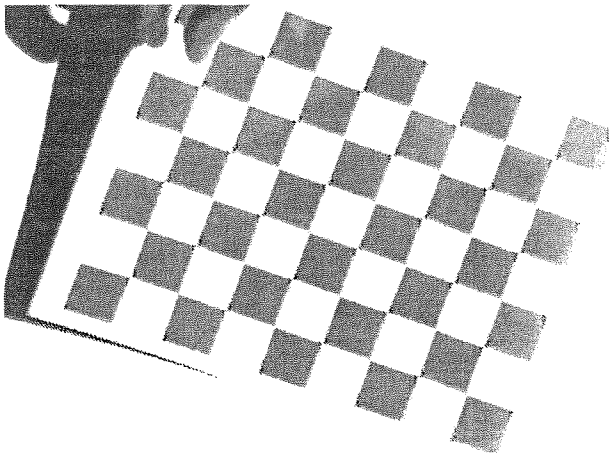


Figura 4.1: Imagem usada para a etapa de calibração

O objetivo da etapa de calibração é o de considerar as distorções sofridas pela imagem obtida de forma a se computar corretamente a posição de um ponto $O (X_o, Y_o, Z_o)$ de um objeto no plano da imagem. Os parâmetros extrínsecos são usados para se determinar as coordenadas de um ponto sobre o tabuleiro, expressas em relação ao plano da câmera. Uma vez que essas coordenadas são calculadas, elas podem ser projetadas no plano da imagem usando-se os parâmetros intrínsecos.

Na verdade, poderíamos efetuar a calibração da câmera usando imagens dos próprios padrões gerados sobre o plano de referência. O fato de termos colunas alternadamente pretas e brancas, ao invés das quadrículas do tabuleiro, não é essencial se considerarmos as linhas horizontais do padrão projetado e de suas respectivas imagens, como epipolares. Entretanto, devido à facilidade de uso do *software* desenvolvido por Boughet, resolvemos optar por sua utilização.

4.1.1 Parâmetros intrínsecos

Os parâmetros intrínsecos incluem a distância focal F (representada, pelo sistema, por um vetor $[F1 \ F2]$ devido à diferença de escala das coordenadas horizontais e verticais), o ponto principal da imagem (vetor $[Pp1 \ Pp2]$), o coeficiente de inclinação (α) e os coeficientes de distorção radial e tangencial da imagem (vetores $[\omega1 \ \omega2 \ \omega3 \ \omega4 \ \omega5]$ e d_t , respectivamente). A determinação destes parâmetros está relacionada ao seguinte modelo de formação da imagem de um ponto (X_o, Y_o) .

Seja X_n a projeção normalizada da imagem, onde:

$$X_n = \begin{vmatrix} X_o/Z_o \\ Y_o/Z_o \end{vmatrix} = \begin{vmatrix} x \\ y \end{vmatrix} \quad (4.1)$$

Fazendo $r^2 = x^2 + y^2$, depois de incluir as distorções provocadas pela lente da câmera, podemos obter o vetor X_d , de coordenadas normalizadas do ponto \mathbf{X} considerando a distorção:

$$X_d = \begin{vmatrix} X_{d1} \\ X_{d2} \end{vmatrix} = (\omega1.r^2 + \omega2.r^4 + \omega5.r^6).X_n + d_t \quad (4.2)$$

onde:

$$d_t = \left| \begin{array}{c} 2.\omega_3.x.y + \omega_4.(r^2 + 2.x^2) \\ \omega_3.(r^2 + 2.y^2 + 2.\omega_4.x.y) \end{array} \right| \quad (4.3)$$

ou

$$d_t = 2.(\omega_3.y + \omega_4.x) \cdot \left| \begin{array}{c} x \\ y \end{array} \right| + r_2 \cdot \left| \begin{array}{c} \omega_4 \\ \omega_3 \end{array} \right|$$

Observe que a primeira parcela da equação acima ainda representa uma variação na direção radial. A segunda parcela indica que a variação fora da direção radial se dá na direção de $\left| \begin{array}{c} \omega_4 \\ \omega_3 \end{array} \right|$.

Uma vez que o modelo de distorção foi aplicado, podemos determinar a coordenada final $\mathbf{x} = [x_p, y_p]$ da projeção do ponto \mathbf{O} no plano da imagem:

$$x_p = F1.(X_{d1} + \alpha.X_{d2}) + Pp1 \quad (4.4)$$

$$y_p = F2.X_{d2} + Pp2 \quad (4.5)$$

Assim, a relação entre o vetor $[x_p, y_p]$ e o vetor normalizado de distorção X_d pode ser definida pela seguinte equação linear:

$$\left| \begin{array}{c} X_p \\ Y_p \\ 1 \end{array} \right| = M_c \cdot \left| \begin{array}{c} X_{d1} \\ X_{d2} \\ 1 \end{array} \right| \quad (4.6)$$

Onde M_c é a chamada matriz da câmera, dada por:

$$M_c = \begin{vmatrix} F1 & F1.\alpha & Pp1 \\ 0 & F2 & Pp2 \\ 0 & 0 & 1 \end{vmatrix} \quad (4.7)$$

É importante salientar que a distância focal F , em milímetros, é representada pelo vetor $[F1 \ F2]$ em unidades de *pixel* verticais e horizontais. Isso permite que *pixels* que não são quadrados sejam considerados. O parâmetro α em nosso sistema foi substituído por uma correção linear no alargamento das faixas projetadas no plano de referência provocado pelo posicionamento oblíquo do projetor (Ver capítulo 5.3). Outro ponto importante a ser considerado é o fato de que os parâmetros acima podem assumir valores extremamente pequenos (menores que 1mm), especialmente nos casos onde *hardware* de alta precisão é usado.

A convenção usada para o sistema de coordenadas do programa de calibração define o ponto de origem (0, 0) o pixel esquerdo superior da imagem. De forma a ser mais conveniente com a biblioteca gráfica utilizada em nosso sistema (ver capítulo 5.3.1) a origem foi transformada para o canto inferior esquerdo da imagem.

4.1.2 Parâmetros extrínsecos

Os parâmetros extrínsecos de calibração se referem ao posicionamento e orientação do projetor e da câmera. Considerando as rotações como matrizes 3x3 e as translações como vetores 3x1, seja $X_i = (x, y, z)$ as coordenadas de um ponto \mathbf{P} , representado na imagem de referência do tabuleiro e dado que: $X_c = (x_c, y_c, z_c)$, onde X_c é o vetor de coordenadas de \mathbf{P} no plano de referência da câmera, temos:

$$X_c = R_c \cdot X_i + T_c \tag{4.8}$$

onde T_c é o vetor de coordenadas da origem do tabuleiro e a terceira coluna da matriz 3x3 definida por R_c é o vetor normal do plano contendo o tabuleiro no quadro de referência da câmera.

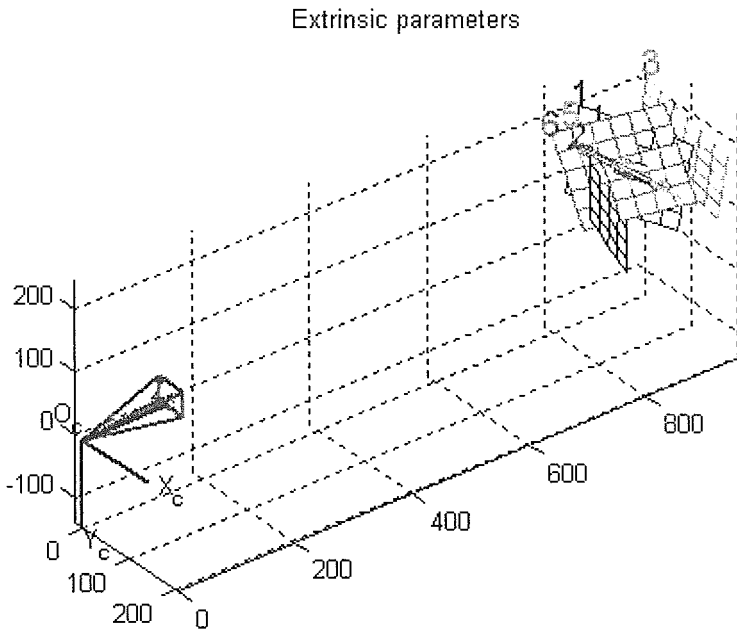


Figura 4.2: Imagem resultante dos cálculos de parâmetros extrínsecos

4.2 Codificação

Os métodos de codificação estão, é claro, diretamente relacionados com as características do padrão de projeção adotado. Como já foi dito, há tipos de padrões binários (preto e branco), em níveis de cinza e coloridos.

Se considerarmos um sistema que utiliza uma codificação que pode variar para cada *pixel*, teríamos, para reconstruir um objeto, decodificar os $n.m$ *pixels* das imagens onde ele está representado. Já um sistema que utiliza apenas colunas ou linhas para codificar a profundidade dos pontos na superfície de um objeto, permite que o problema se torne unidimensional, dado que as coordenadas verticais e horizontais dos *pixels* da câmera e do projetor são as mesmas. A complexidade de se identificar uma coluna ou linha, neste caso, varia de $O(n)$ quando temos uma única projeção de resolução máxima ou $O(\log_n)$ no caso de se adotar a sistemática do método *Gray-code*, o que permite, assim, uma aquisição mais rápida de pontos.

Nosso trabalho abrange três técnicas de luz estruturada. O primeiro deles é o método de *Gray-code*. O segundo é o método de *Phase-Shifting*, cujo objetivo é o de aumentar a precisão do padrão *Gray-code* para o nível sub-*pixel*. O terceiro método apresentado é o de codificação através de cores.

4.2.1 Gray-code

A abordagem mais simples de escaneamento com luz estruturada é o método *Gray-code*, onde k imagens de uma cena são capturadas, cada uma iluminada por um padrão P_i , $i = 1 \dots k$ que consiste de 2^i linhas verticais pretas e brancas intercaladamente. Para cada *pixel* projetado, a coluna do padrão de maior resolução a que ele pertence pode ser identificada pela expressão:

$$\sum bw(i).2^{k-i}, \quad (4.9)$$

onde $bw(i)$ é 0 ou 1 se o *pixel* for respectivamente preto ou branco na i -ésima imagem. Assumindo que a câmera e o projetor têm eixos óticos horizontais e paralelos, e seus centros óticos estão afastados apenas ao longo do eixo X (figura

3.1), as linhas epipolares do projetor e da câmara serão horizontais e de mesma altura, o que significa que os *pixels* correspondentes ao projetor e à câmara, têm as mesmas coordenadas verticais¹.

A projeção de padrões em diferentes resoluções permite a identificação de colunas sem contar as transições da esquerda para a direita, o que torna desnecessária a visualização de todas as faixas projetadas sobre o plano de referência.

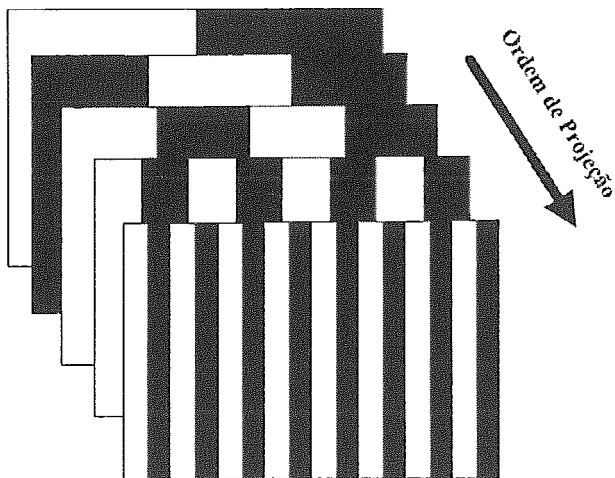


Figura 4.3: Padrão *Gray-code* utilizado

Deve-se observar que determinados tipos de padrão, diferentes do indicado acima, também permitem a identificação de uma coluna no nível de resolução maior. Entre essas formas alternativas de se fazer a projeção, adotamos, por sua maior robustez no que se refere a erros por identificação do padrão projetado nas transições de faixa, a empregada por Sansoni [42], onde a cota (preta ou branca) atribuída a um *pixel* da *j*-ésima coluna ($j = 0, \dots, N - 1$) da imagem contendo o *k*-ésimo padrão projetado ($k = 1, \dots, L$) é dada por: $P_1 = I([N/2, N - 1])$, $P_2 = I([N/4, 3N/4])$... $P_{k+1}(j) = P_k(2j \bmod N)$.

Alguns tipos de padrões *Gray-code* projetados necessitam de uma tabela de conversão de índices (*Look-up table*) para se obter o código *Gray-code* expresso em 4.9. Em nosso caso, esta tabela de conversão não é necessária, uma vez que

¹Este princípio não é válido caso a câmara ou o projetor estejam posicionados de forma oblíqua ao plano de referência. Porém, por simplicidade, podemos assumir que tal conceito é válido para inclinações de pequena ordem.

a conversão dos códigos obtidos pelo método acima é bastante simples. Seja B_k , $k = 0, \dots, L-1$ o k -ésimo *bit* do código de um *pixel* obtido pelo nosso método, para determinarmos o código *Gray-code* comum deste *pixel* usa-se o seguinte procedimento:

```
for  $k = 0$  até  $L - 2$  do
  if  $B_k = 1$  then
     $B_{k+1} = 1 - B_{k+1}$ ;
```

Conforme já foi mencionado, a principal vantagem de se usar este tipo de projeção está na robustez com respeito à classificação errônea de um *pixel* próximo à transição de uma faixa. Quando se usa um método *Gray-code* comum, um erro no k -ésimo *bit* do código de um *pixel* da imagem determina uma diferença de 2^{L-k} no índice da coluna do último nível onde esse *pixel* está.

Porém, considere um *pixel* \mathbf{p} em uma transição 0-1 do padrão P_k . Supondo que um *bit* B_k seja avaliado de forma errada, uma vez que \mathbf{p} não é um *pixel* de transição em todas as outras resoluções, ele será menos sensível à classificações incorretas nos outros níveis. Assim, pode-se assumir que os outros *bits* do código de P_k foram estimados corretamente. Como \mathbf{p} está em uma transição de P_k , podemos provar que, a parte final do código de \mathbf{p} , de B_{k+1} em diante, têm o formato 100...000, que é transformado pelo procedimento descrito acima em 000...000 se $B_k = 1$, e em 111...111 se $B_k = 0$.

Portanto, no caso de B_k ser codificado de forma errada, por exemplo, como 0 ao invés de 1, o erro é de apenas um, ou seja, para um código de 8 *bits*, o *pixel* seria codificado como 01111111 ao invés de 10000000). Essa diferença é de somente 1 no nível de resolução maior, sendo bem menos significativa do que 2^{L-k} obtido pelo método *Gray-code* tradicional.

4.2.2 Phase-shifting

A abordagem conhecida como *Phase-shifting* tem como objetivo aumentar a continuidade de uma superfície reconstruída pelo método *Gray-code*. Para tanto, a técnica se baseia na projeção de quatro padrões que substituem o padrão *Gray-code* menos significativo para codificação (último *bit*).

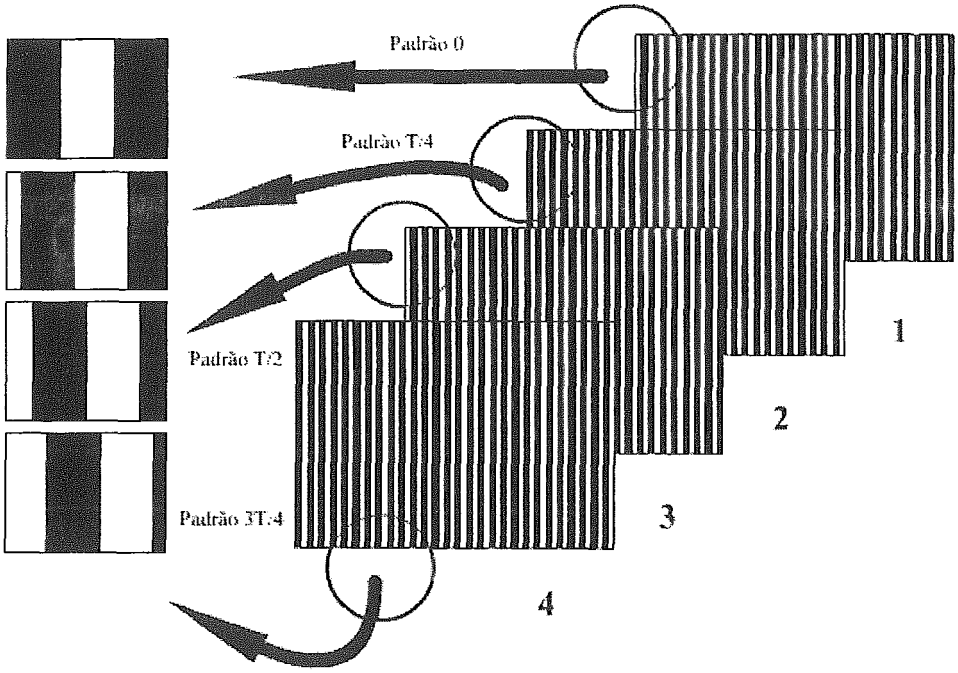


Figura 4.4: Padrão *Phase-shifting* utilizado

Como pode ser visto na figura acima, a intensidade dos quatro padrões projetados variam segundo um padrão senoidal de mesmo período T , estando, cada um, defasado do anterior em apenas $1/4$ desse período.

Considere $O_i(j, k)$ e $R_i(j, k)$ o nível de cinza no pixel (j, k) quando o padrão com *Phase-shifting* $i.T/4$, $i = 0, 1, 2, 3$ é projetado na cena com e sem o objeto respectivamente. Os valores de $O_i(j, k)$ e $R_i(j, k)$ podem ser idealmente expressos por:

$$O_i(j, k) = M_O(j, k) + \frac{1}{2} \cdot S_O(j, k) \cdot \cos(W_h \cdot (q(k) + s(j, k)) - i \cdot \frac{\pi}{2}) \quad (4.10)$$

$$R_i(j, k) = M_R(j, k) + \frac{1}{2} \cdot S_R(j, k) \cdot \cos(W_h \cdot (q(k) + i \cdot \frac{\pi}{2})) \quad (4.11)$$

onde $M_R(j, k)$ e $M_O(j, k)$ representam a média de brilho no *pixel* (j, k) nas 4 imagens obtidas com e sem o objeto respectivamente. Eles indicam a contribuição

da luz ambiente para o valor atribuído a (j,k) . S_R e S_O representam a média do contraste medida no *pixel* (j,k) , também nessas 4 imagens com e sem o objeto. $M_R(j, k)$ e $M_O(j, k)$ dependem da intensidade da projeção e das propriedades de refletividade da superfície. $W_h = 2.\pi/T$ é a frequência espacial ao longo do eixo X do padrão senoidal projetado. As variáveis $q(k)$ e $q(k) + s(j, k)$ representam as colunas do padrão projetado sobre a cena com e sem o objeto, respectivamente, vistas no *pixel* (j,k) .

A partir da expressão 4.11 acima, considerando $i = 0, 1, 2, 3$, podemos obter:

$$R_1(j, k) - R_3(j, k) = \frac{1}{2}.S_R(j, k).(\cos(W_h.q(k)) - \frac{\pi}{2}) - \cos(W_h.q(k) - 3.\frac{\pi}{2})) = S_R(j, k). \sin(W_h.q(k))$$

e

$$R_0(j, k) - R_2(j, k) = \frac{1}{2}.S_R(j, k).(\cos(W_h.q(k)) - \cos(W_h.q(k) - \pi)) = S_R(j, k). \cos(W_h.q(k))$$

Nas equações acima, a influência de M_R foi eliminada. Para fazermos o mesmo com $S_R(j, k)$, basta dividirmos a primeira equação pela segunda resultando em:

$$\frac{R_1(j, k) - R_3(j, k)}{R_0(j, k) - R_2(j, k)} = \tan(W_h.q(k)) \quad (4.12)$$

ou

$$(W_h.q(k)) = \Phi_R(j, k) = \tan^{-1}\left(\frac{R_1(j, k) - R_3(j, k)}{R_0(j, k) - R_2(j, k)}\right) \quad (4.13)$$

Um processo similar pode ser executado a partir da equação 4.10 para se obter os valores para a imagem com o objeto:

$$(W_h.(q(k) + s(j, k))) = \Phi_O(j, k) = \tan^{-1}\left(\frac{O_1(j, k) - O_3(j, k)}{O_0(j, k) - O_2(j, k)}\right) \quad (4.14)$$

O deslocamento lateral $S(j,k)$ pode ser obtido por:

$$S(j, k) = \left(\frac{1}{W_h}\right) \cdot (\Phi_O(j, k) - \Phi_R(j, k)) \quad (4.15)$$

O processo acima permite identificar uma dificuldade clara quando método *Phase-shifting* é usado para a codificação inteira ao invés de ser empregá-la somente para substituir a parte final da codificação *Gray-code*, como fizemos neste trabalho. Ocorre que a função \tan^{-1} só está perfeitamente definida se assumirmos que seu contradomínio é um intervalo aberto dado por $((2n - 1)\Pi, (2n + 1)\Pi)$. Assim, se $q(k)$ e $q(k) + S(k, j)$ não estiverem ambos num mesmo desses intervalos, um erro múltiplo de \mathbf{T} irá ocorrer se usarmos a expressão 4.15 para determinar $S(j,k)$. Isso impede o uso desta técnica a objetos com uma profundidade maior.

O fato de, ao invés de formas senoidais puras, termos somente aproximações discretas produzidas pelo padrão projetado, prejudica apenas a suavidade das superfícies reconstruídas, mas não impede sua reconstrução. Na verdade, ao invés de senóides, podemos usar quaisquer sinais periódicos $S_i - S(t + i.t/4)$, $i = 0, 1, 2, 3$, tais que $S_3(t) - S_1(t)/S_2(t) - S_0(t)$ seja um homeomorfismo.

Outro problema importante reside nas equações de $R_i(j, k)$ e $O_i(j, k)$, para $i = 0, 1, 2, 3$, visto que essas não são suficientemente precisas. Neste caso, os valores de $M_O(j, k)$, $S_O(j, k)$ e $\Phi_O(j, k)$ podem ser melhor aproximados através do método de mínimos quadráticos, como proposto por [20]:

$$O_i(j, k) - M_i(j, k) + \frac{1}{2} \cdot S_O(j, k) \cdot \cos(\Phi_O(j, k) - i \cdot \frac{\pi}{2})^2 \quad (4.16)$$

De uma forma geral, o método de *Phase-shifting* é melhor implementado em sistemas que se valem de equipamentos de altíssima resolução, onde o padrão senoidal projetado sofre pouca influência das distorções causadas pelo sistema de captura ou pelo sistema de projeção, que tornam as equações 4.10 e 4.11 imprecisas. Esse foi um importante obstáculo observado em nosso sistema como será visto no capítulo 5.2.2.

4.2.3 Codificação por cor

Os métodos de codificação por cor visam diminuir, como já foi comentado anteriormente, o número de padrões projetados durante o processo de escaneamento e, ainda assim, tentar manter a robustez do método *Gray-code*. A codificação usando cores permite que se possa obter *range-maps* com apenas uma imagem, como no trabalho desenvolvido por Zhang [26].

Em nosso sistema, a codificação com cores se utiliza de padrões projetados com valores RGB que podem ser convertidos em um sistema de codificação idêntico ao *Gray-code*, com códigos de sete bits, como mostra a figura seguinte:

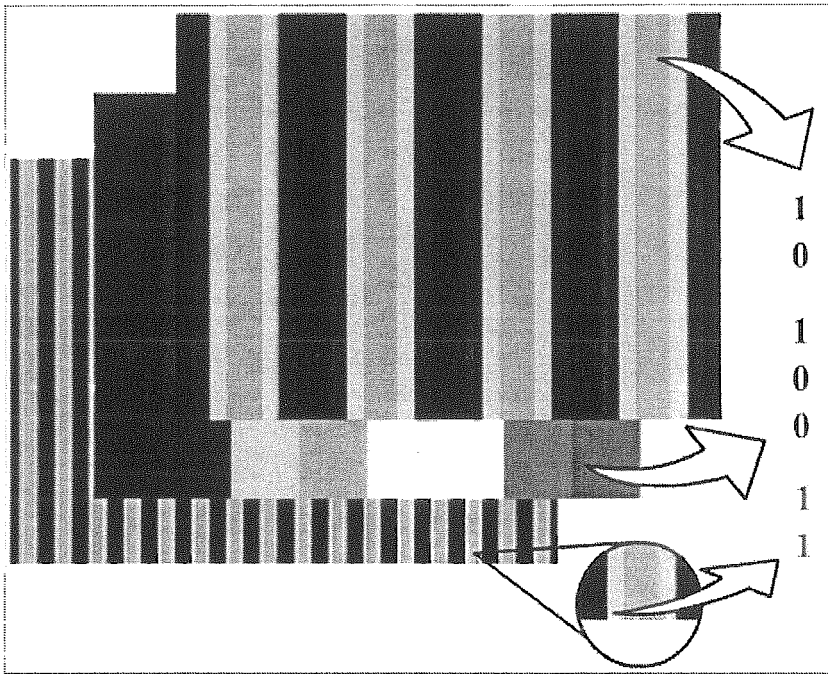


Figura 4.5: Sistema de codificação com cores.

Apesar de simplificar o processo de escaneamento, o sistema de codificação por cor é altamente dependente da textura do objeto escaneado e da qualidade do equipamento de aquisição da imagem. Em nossa implementação, por exemplo, o padrão de cores utiliza o canal vermelho o menor número de vezes possível (ver figura acima) devido à incapacidade da câmera de encontrar as bordas de faixas vermelhas muito estreitas. Objetos com cores que não são neutras, ou seja, que não refletem todos os tipos de cor, acabam criando erros que impedem a localização

das bordas das faixas. Além disso, sombras e penumbras podem alterar a cor de uma faixa, tornando-a passível de ser classificada como outra cor.

Porém, o principal problema do método de projeção colorida é o de cruzamento de cores (*cross-talking*). Isso ocorre quando um determinado canal de cor influencia parcialmente a cor das faixas vizinhas, “misturando” as cores e dificultando a sua classificação. No capítulo 5.3.1 serão apresentadas as soluções desenvolvidas para esse tipo de problema.

Os padrões coloridos podem ser adaptados para determinados tipos de cenas, como no caso do escaneamento de partes do corpo humano ou de objetos altamente reflexivos. Essa adaptação do padrão ao tipo de objeto sendo escaneado foi usada em nosso sistema de forma a identificar com maior precisão os limites entre as faixas projetadas e contornar o problema de *cross-talking* entre elas.

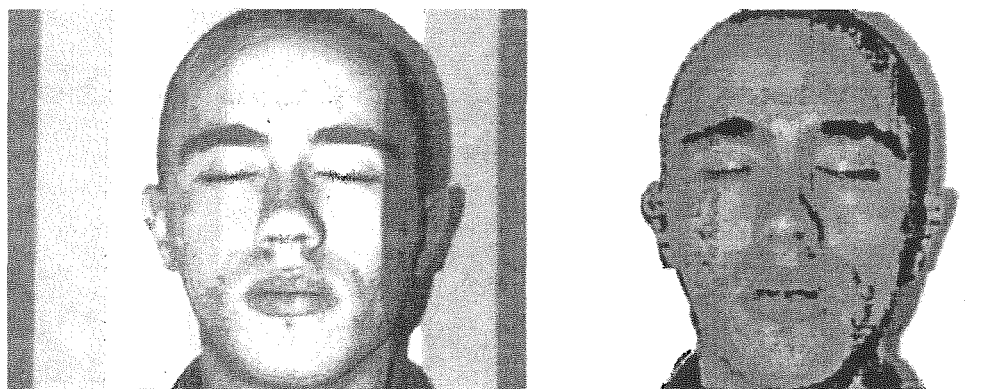


Figura 4.6: Escaneamento com padrão colorido adaptado para cor da pele.

Capítulo 5

Implementação

O principal objetivo do sistema implementado neste trabalho foi o de obter *range maps* com a menor quantidade de imagens possível usando equipamentos de aquisição e projeção de baixo custo.

O sistema de escaneamento foi implementado em linguagem C. É importante salientar que o *hardware* utilizado influenciou consideravelmente na elaboração do código fonte do programa de escaneamento. O programa final possui três versões: uma para o método *Gray-code*, outra para o método *Phase-shifting* e outra para o método de projeção com cor.

Além do procedimento de *scanning*, foram implementados programas para gerar os padrões de *Phase-shifting* e outro para gerar os padrões de projeção coloridos. Dois outros módulos foram implementados para facilitar a análise do código e dos resultados obtidos: um para mapear as cores que o sistema estava identificando e outro que apresenta o gráfico com os códigos criados pelo sistema de escaneamento ao longo de uma linha da imagem.

O equipamento utilizado consiste em um projetor multimedia Epson Power Lite 50c, uma câmera portátil ColorFinder NTSC da Sony, e uma placa de captura de vídeo com resolução máxima de 640x480 *pixels* instalada em um computador Pentium Celeron de 500 MHz. Todos os equipamentos são considerados atualmente obsoletos e nenhum deles foi especificamente projetado para o escanea-

mento de objetos. Por essa razão, vários problemas surgiram durante o processo de aquisição. Porém, o uso deste *hardware* de baixo custo permitiu uma avaliação clara da robustez das técnicas utilizadas como será mostrado a seguir.

5.1 Aquisição de imagens

O processo de aquisição de imagens começa com a etapa de posicionamento do equipamento de *scanning* e do objeto a ser escaneado. Como mostrado anteriormente, a câmera é colocada de forma perpendicular ao plano de referência do objeto e o projetor é colocado ao seu lado projetando padrões de forma oblíqua.

Os ajustes no projetor e na câmera são feitos até esta consiga visualizar um número suficiente de linhas projetadas sobre o plano. Estas linhas devem estar o mais verticais possíveis e devem apresentar-se assim em todas as diferentes resoluções dos padrões projetados.

Após o ajuste do equipamento são feitas as medições da largura e altura da região do plano de referência visualizada pela câmera (**FW** e **FH**) e das distâncias entre a câmera e o projetor e de ambos até o plano de projeção. As imagens são então capturadas, convertidas no formato **ppm** e segmentadas antes que sejam lidas pelo programa de reconstrução.

Opatamos por fazer a segmentação da imagem do objeto para restringir o procedimento de decodificação e, assim, agilizar o processo total de reconstrução. Quando foi necessário fazer uma segmentação manual, empregamos recursos de *software* para edição de imagens existentes no mercado. Lembramos que ao contrário de outras implementações que utilizam objetos necessariamente brancos contra um fundo escuro [16] [19], o objetivo de nosso trabalho era o de testar os métodos de escaneamento com luz estruturada sob diversas condições e com objetos de textura variada, o que dificulta a segmentação das imagens usando-se meramente um *threshold* para o brilho. Em função disso, nosso sistema não pode ser comparado aos equipamentos comerciais que executam identificação e calibração automaticamente [23]. É importante comentar que muitas implementações [18] [2] [42] não descrevem o processo de segmentação, o que impede uma análise comparativa mais clara de nosso sistema com estes citados.

Além da segmentação das imagens do objeto, efetuamos também a detecção de regiões de sombra e penumbra. A partir desta detecção é criada uma imagem que servirá de filtro para eliminar os pontos dessas regiões que poderiam gerar codificação errônea. Esse filtro é explicado no item 5.3.1.

5.2 Problemas inerentes ao equipamento

O principal desafio deste trabalho foi superar os problemas inerentes ao equipamento usado, em especial a câmera e o projetor, já que a limitação de resolução da placa de captura influenciou pouco nos resultados da reconstrução em comparação com as dificuldades encontradas na projeção dos padrões.

Um aspecto fundamental em relação ao equipamento de projeção usado foi a interdependência do projetor e da câmera. Enquanto a câmera foi projetada para capturar cenas num campo centrado em relação à sua lente, o projetor usado gera imagens ligeiramente para cima. Além disso, ambos possuem ajustes diferentes em relação às distâncias de captura e projeção, dificultando ainda mais a escolha do posicionamento ideal para eles. Essa relação entre os equipamentos criou ainda um problema de amostragem. Assim, como acontece com todos os tipos de sistemas de escaneamento, nosso sistema só pode escanear modelos dentro de um determinado espectro de medidas, pois as distâncias relativas entre o equipamento de projeção e de captura só podem ser mantidos até o limite de um dos dois, projetor ou câmera. Objetos muito pequenos (por exemplo, um botão [2]) ou muito grandes (por exemplo, um automóvel [14]) não conseguem ser enquadrados com nosso equipamento. A área de trabalho de nosso sistema é de, no máximo, 30x30 cm.

Outros problemas que surgiram nesta etapa como, por exemplo, a influência da luz ambiente indireta na iluminação do objeto, puderam ser contornados tanto pelo *Camera Calibration Toolbox*, como pela etapa de segmentação das imagens de entrada do nosso sistema.

5.2.1 Questões vinculadas ao projetor

Diversas questões relacionadas ao projetor multimídia usado tiveram que ser resolvidas. A principal delas foi a limitação deste em projetar padrões de resolução superior a 640x480 a pequenas distâncias. Para que possam ser identificadas as colunas numa resolução 800x600, a distância de projeção deveria ser de, pelo menos, 5 metros. Colocando-se os objetos muito próximos ao projetor e à câmera, acaba-se perdendo acuidade no processo de reconstrução, pois as variações de profundidade na superfície do objeto ficam muito menores que a distância ao plano de referência. Ao projetarmos padrões com maior resolução, também foi identificada uma suavização nas bordas destes padrões que não pôde ser eliminada. Isso dificultou enormemente a identificação das bordas das faixas projetadas.

Outro problema foi a distorção causada pelo posicionamento oblíquo do projetor em relação ao plano de referência. Este problema ocorre pois o projetor e a câmera precisam estar alinhados horizontalmente em relação ao plano de referência. O problema ocasionado ao usarmos a câmera e o projetor a diferentes distâncias do plano de referência reside no fato de que, para determinarmos a altura de um elemento da superfície do objeto representado num *pixel*, precisamos então substituir na expressão (como mostrado abaixo) a distância fixa D , entre o projetor e a câmera, por um valor que é dependente da coluna representada no *pixel*, o que certamente introduz mais imprecisão no cálculo dessa profundidade.

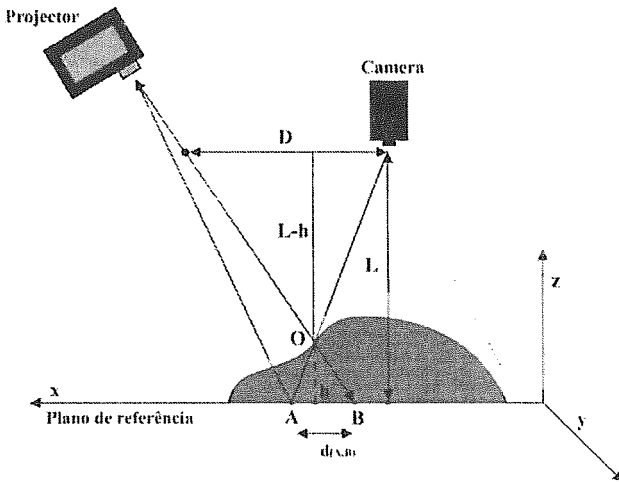


Figura 5.1: Problema relativo ao posicionamento irregular entre o projetor e a câmera

Pela figura acima podemos deduzir que:

$$\frac{L - h}{h} = \frac{D}{d_{(A,B)}} \quad (5.1)$$

ou

$$h = \frac{L \cdot d_{(A,B)}}{D + d_{(A,B)}} \quad (5.2)$$

Assim, de forma a aumentar a precisão do sistema e para que os padrões de resolução mais fina possam ser distinguíveis nas imagens obtidas pela câmera utilizada, então ambos devem estar próximos ao plano e posicionados de forma ter a mesma distância deste. Com isso, o projetor precisa ficar inclinado em relação ao plano pois, do contrário, as deformações causadas nas faixas projetadas sobre o objeto, e que servem para se determinar a profundidade dos seus pontos, são pequenas demais para serem identificadas pelo sistema de escaneamento. Este posicionamento oblíquo gera o problema do *broadening*, ou alargamento, das faixas projetadas.

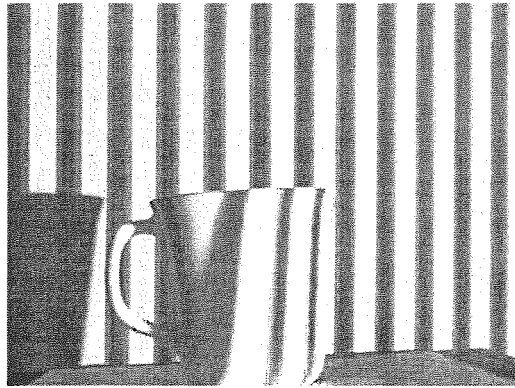


Figura 5.2: Problema de alargamento das faixas

Este caso está previsto na literatura [43] [17] e pode ser corrigido, considerando-se que as faixas aumentam a partir de uma das extremidades da projeção no plano de referência, pela fórmula:

$$faixa = (faixa + 1) \cdot ((1 - ((faixa/254) \cdot largmax) + (faixa/254) \cdot largmin))$$

onde **faixa** é o índice da faixa a ser corrigida e **largmax** e **largmin** são os tamanhos das faixas de maior e menor larguras respectivamente. Esta fórmula expressa a soma dos termos de uma projeção aritmética, e é válida para o caso em que trabalhamos, onde o último padrão de resolução possui 128 linhas.

Por último vale ressaltar que o sistema de iluminação do projetor cria padrões radiais de interferência luminosa que provocam erros na codificação das faixas como pode ser visto na figura seguinte, obtida quando o projetor é colocado bastante próximo ao plano de projeção.

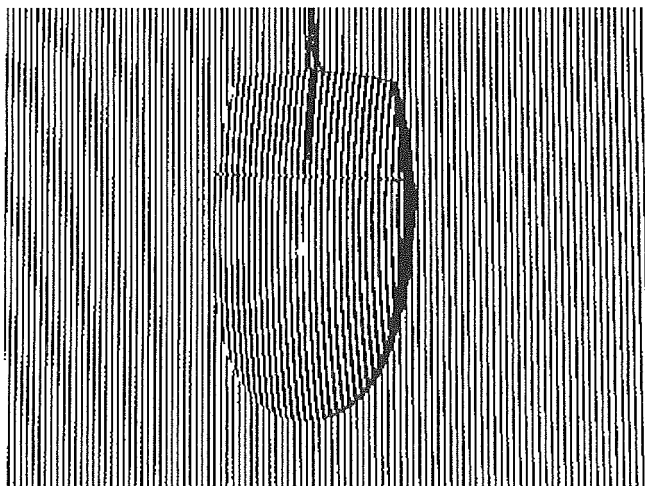


Figura 5.3: Padrão radial provocado pelo sistema de iluminação do projetor

Nesta situação, a câmera deve ser ajustada para reduzir o valor de brilho da imagem capturada. Ainda assim, o problema pode não estar totalmente resolvido e, por isso, nosso sistema se vale da entrada de duas imagens, uma com o objeto iluminado pelo projetor e outra sem a iluminação. É feita então uma média ponderada *pixel a pixel* de ambas as imagens. Essa média é usada como *threshold* pelo sistema, de forma que este possa determinar que *pixels* são brancos (*bit 1*) e quais são pretos (*bit 0*). Esse método é sugerido por Trobina [17] e se mostrou eficiente em todos os casos tratados neste trabalho.

5.2.2 Questões vinculadas à câmera

Os problemas gerados pela câmera também dificultam a correta codificação das faixas projetadas. Dentre esses problemas, o mais importante foi o *anti-aliasing* efetuado pela câmera, e que dificulta a identificação dos limites das faixas, especialmente quando os padrões são mais finos.

A figura abaixo ilustra bem este problema. A imagem mostra o resultado da identificação de faixas feita pelo programa de reconstrução. Pode-se verificar claramente que o *anti-aliasing* cria alterações nas faixas, pois estas deveriam ter a mesma largura.

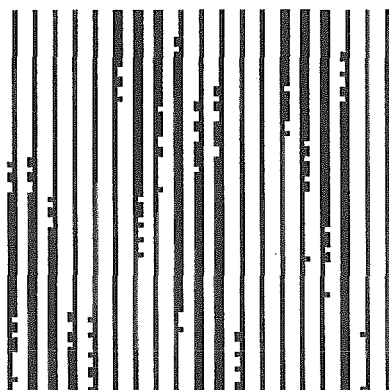
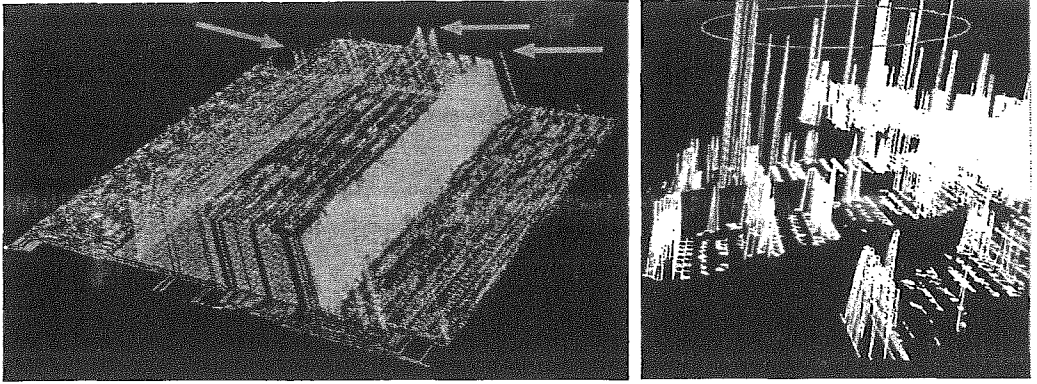


Figura 5.4: Faixas obtidas em função do efeito provocado pelo sistema de *anti-aliasing* da câmera.

A decodificação incorreta de projeção sobre um *pixel*, especialmente num nível de resolução mais baixo, cria os chamados *outliers*, ou seja, pontos cuja profundidade deveria acompanhar a da superfície escaneada ao redor, mas que acabam recebendo valores de profundidade maiores ou menores. Com isso, a reconstrução correta de uma superfície se torna dificultosa.

Este problema foi minimizado quando todos os equipamentos passaram a utilizar a mesma resolução (640x480), tanto para captura quanto para projeção, e as faixas foram projetadas e capturadas em preto e branco com alto contraste por parte do projetor. Assim, a câmera conseguiu focar mais adequadamente o objeto, reduzindo o efeito de *blurring* que prejudica a decodificação. Usando a mesma resolução, conseguimos reduzir o problema determinado pelo fato dos limites horizontais da porção do plano de referência amostrados num *pixel* não coincidirem



(a)

(b)

Figura 5.5: *Outliers* em objetos escaneados

com os limites das faixas verticais projetadas. Infelizmente, no caso dos padrões coloridos, este problema pôde ser diminuído pelo aumento do contraste somente e, por essa razão, foi utilizado um filtro de detecção de cores para resolvê-lo, como será mostrado no item 5.3.1.

O *anti-aliasing* causou dificuldades ainda maiores quando padrões *Phase-shifting* foram usados. A câmera gerou um borramento na imagem maior que o deslocamento do período das faixas, criando muitos *outliers* e tornando as superfícies irregulares.

Outro fator que influenciou o posicionamento do equipamento foi o problema de foco da câmera sob forte iluminação do projetor. Neste caso, a câmera usada se tornou incapaz de visualizar áreas com muito brilho, alterando automaticamente seu foco e exigindo que o sistema fosse novamente calibrado. Ao aproximarmos a câmera do objeto, esta se manteve estável e assim as imagens puderam ser capturadas, mesmo com alto contraste.

Os problemas de distorção radial e tangencial da câmera foram reduzidos pelo *Camera Calibration Toolbox* e não se mostraram tão relevantes se comparados aos problemas citados acima visto gerarem erros de menor grandeza do que os provocados pela codificação errônea.

5.3 Sistema implementado

O sistema implementado neste trabalho utiliza como entrada um determinado número de imagens para identificação do fundo (plano de referência) e do objeto. Três imagens são obrigatórias em todos os métodos desenvolvidos: duas imagens do objeto, com e sem iluminação, e uma imagem que será usada como filtro para que o processo de reconstrução se restrinja apenas à imagem do objeto. Esta imagem é binária e foi gerada em uma etapa de pré-processamento tendo sido criada a partir da imagem do objeto com iluminação, e contém *pixels* de valor 1 para o objeto segmentado e 0 para o fundo da cena. Ela será multiplicada às imagens contendo as projeções.

Para o sistema que usa o método *Gray-code* utilizamos, além das três imagens citadas, mais 14 imagens: 7 para o objeto e 7 para o fundo. No caso do método de *Phase-shifting* são necessárias 22 imagens: 7 de *Gray-code*, mais 4 para o *Phase-shifting* com e sem o objeto. O código para o sistema de escaneamento empregando cores utiliza apenas 6 imagens, sendo 3 com e 3 sem o objeto.

As três versões de código desenvolvidas apresentam um mesmo fluxo de operações. Porém, a implementação do sistema de escaneamento usando cores apresenta diferenças substanciais, em especial na parte de decodificação das imagens e de filtragem, devido à customização necessária para a escolha dos parâmetros (*thresholds*) que permitem a identificação das faixas devido à influência da cor do objeto escaneado. O esquema seguinte representa a arquitetura do programa de reconstrução desenvolvido.

Após a etapa de aquisição de imagens, estas são carregadas pelo programa, que aceita os formatos **ppm** e **pgm**. Depois desta etapa, determina-se o *threshold* de cada *pixel*, como foi comentado anteriormente, utilizando-se uma média do valor desses *pixels* nas imagens do objeto com e sem a iluminação do projetor. As imagens são então binarizadas de acordo com os *thresholds* estabelecidos. Vale lembrar que este processo de binarização não ocorre no método de escaneamento através de cores, sendo substituído pela etapa de identificação das cores projetadas.

Ao determinar esses valores, o programa mostra uma interface onde podem ser digitados os valores de calibração **FW**, **FH**, **L** e **D**, e os valores das faixas de menor

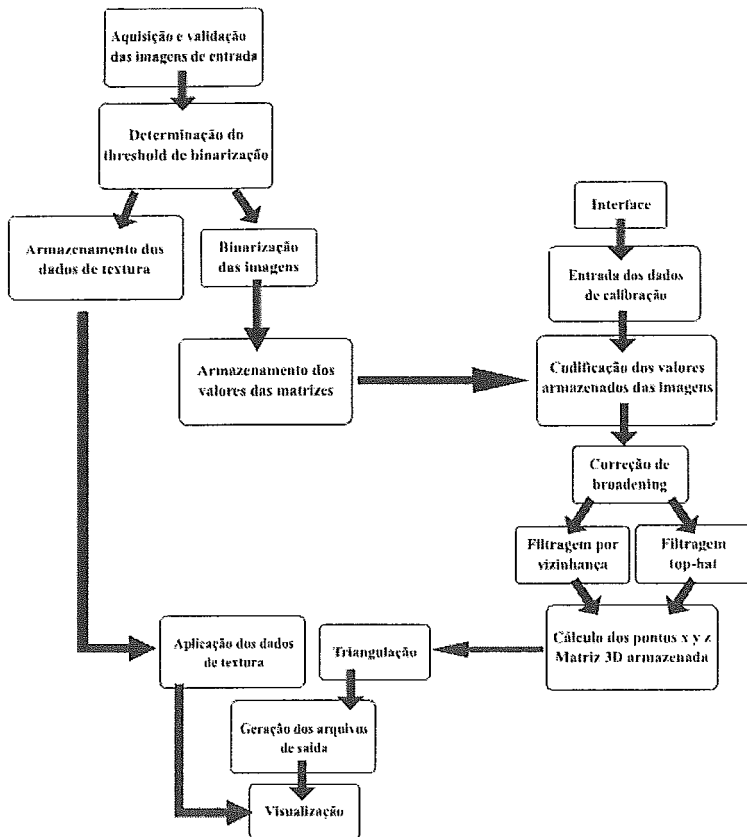


Figura 5.6: Arquitetura do programa de reconstrução

e maior largura na resolução mais fina. Clicando-se no botão de “reconstrução”, o sistema inicia o cômputo da profundidade associada a cada *pixel* usando as equações apresentadas no capítulo 3.1, e utilizando os valores de calibração dados pelo usuário. Durante esse processo o sistema corrige também o erro provocado pelo alargamento das faixas e executa processos de filtragem na codificação, destinados tanto a eliminar pontos em áreas de sombra e penumbra como corrigir os valores das codificações nos *pixels* em que ela é considerada errônea, de forma a excluir esses *pixels* da montagem do modelo geométrico final.

Ao final destes processos de filtragem, o sistema tem as coordenadas dos pontos da parte da superfície visualizada do objeto correspondente aos *pixels* considerados válidos. É feita então a triangulação dessa nuvem de pontos. Utilizamos apenas uma malha regular para triangulação, visto esta etapa não ser o objetivo principal proposto neste trabalho. Finalmente, o sistema cria os arquivos de saída

necessários para a avaliação do código (**.dat** ou **.txt**) e exporta um arquivo no formato **.plg**, com as coordenadas dos pontos e as faces dos triângulos obtidos, para que possam ser visualizados e importados por outros programas de modelagem e visualização de superfícies. A aplicação dos dados de textura sobre o objeto pode ser realizada utilizando-se algum desses programas. O *range-map* resultante do processo de reconstrução é mostrado numa janela de visualização.

5.3.1 Filtros

Os filtros utilizados durante a fase de reconstrução do objeto visam eliminar, principalmente, os pontos onde claramente a decodificação não é confiável, como aqueles em áreas de sombra ou de penumbra. Além de um filtro para identificar esses tipos de regiões, é necessária a correção dos valores obtidos nos *pixels*, devido a erros ocasionados pelos problemas descritos nas seções 5.2.1 e 5.2.2, ou determinados pela textura do objeto. Esse outro filtro busca corrigir o valor de *outliers* isolados e que podem ser corrigidos em função dos pontos mais próximos.

De uma forma geral, o método *Gray-code*, na sua versão final de implementação, se mostrou robusto o suficiente para não se beneficiar de filtros de correção de código. Os resultados deste método apresentaram uma quantidade insignificante de pontos com codificação errada e que puderam ser facilmente removidos. Já o método de codificação colorida apresentou mais erros, devido aos problemas já citados, e por isso precisou ser corrigido por todos os tipos de filtro. Por fim, o método de *Phase-shifting* apresentou problemas que foram minimizados, mas não totalmente resolvidos, pelo filtro de eliminação de *outliers*, como será visto mais tarde.

Filtro de codificação crescente

Nosso sistema se vale de um processo de filtragem que analisa e corrige erros no código provocados pelos problemas encontrados nas etapas iniciais de nossa implementação. De forma a analisar mais detalhadamente esses problemas, foi implementado um módulo para o sistema que permite a visualização do gráfico de codificação pertencente a cada linha do *range map* final.

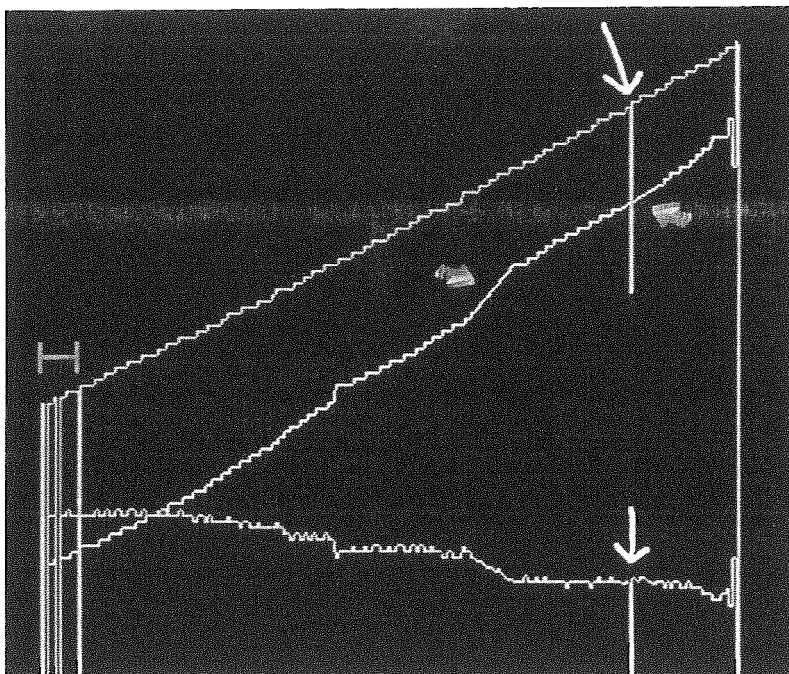


Figura 5.7: Exemplo de gráfico mostrando o código de uma das linhas da imagem

No exemplo acima, a linha verde mostra a codificação definida para as faixas projetadas sobre o plano de referência sem o objeto e a linha amarela mostra a codificação definida para a cena com o objeto, considerando-se uma dada linha da imagem. A linha azul mostra a diferença das duas codificações, a partir da qual a profundidade dos *pixels* é determinada. São os valores desta linha azul que definem as coordenadas finais dos pontos do objeto. As setas brancas apontam um *pixel* que não foi corretamente avaliado e, por isso, acarretou em um erro indicado pela linha azul. A área indicada pela linha vermelha identifica uma região da imagem onde a câmera não conseguiu identificar as cores do padrão. Pode-se notar que esse erro não faz parte do intervalo delimitado pela linha amarela, que é onde se visualiza o objeto. Os *pixels* correspondentes a área indicada pela linha vermelha foram eliminados por se encontrarem em uma região de sombra. Por último, as setas de cor laranja indicam os trechos onde houve correção no código através da interpolação de seus valores. Essa interpolação é resultado da aplicação de um filtro de codificação crescente.

Em sistemas de alta resolução encontrados na literatura, a codificação dos

pixels do *range map* final possui erros desprezíveis, mas nas condições em que realizamos o nosso trabalho é necessário recorrer a um filtro que mantenha as codificações das faixas de forma crescente. Em especial, o método de codificação colorida apresentou erros nas transições das faixas do padrão projetado, como citado no item anterior, que não puderam ser totalmente resolvidos pelo processo de detecção de cores e, por isso, necessitaram da aplicação deste filtro. O algoritmo para essa correção é detalhado a seguir:

```

for (j = 1; j <= no. de colunas; j++) do
    if ( value[j] < value[j-1] && value[j] != 0.0) then
        quebra[k] = j;
        k ++;
m = 0;
flag = FALSE;
if (k > 0) then
    for (l = 0; l < k; l ++ ) do
        if ((quebra[l] - quebra[m]) < limite) then
            flag = TRUE;
        else
            if (flag) then
                last[m] = quebra[l - 1]; flag = FALSE;
                m ++;
                quebra[m] = quebra[l];
if (m > 0) then
    for ( l = 0; l < m; l ++ ) do
        temp1 = last[l] - limite;
        r = 0;
        while (value[temp1+r]== 0) do
            r ++;
        temp1 = temp1 + r;
        temp2 = quebra[l] + limite;
        r = 0;
        while (value[temp2-r]== 0) do
            r ++;
        temp2 = temp2 - r;
        inter = temp2 - temp1;

```

```

if (value[temp1] <= value[temp2]) then
    for (q = 1; q <= inter-1; q++) do
        value[temp1 + q] = ((1 - q/inter) * value[temp1]) + ((q/inter) *
            value[temp2]);
    else
        int v = value[temp1];
        for (q = 1; q <= inter-1; q++) do
            if (value[temp2 - q] != 0.0) then
                break;
            for (r=1; r <= q; r++) do
                Vmax = Vmax(Vmax, value[temp1 + r]);
                value[temp1 + r] = Vmax;

```

Para que o procedimento acima funcione adequadamente, parte-se do pressuposto que os intervalos na codificação dos *pixels* de uma linha é monótona e crescente, têm tamanho pequeno (menor que **limite**), e encontram-se afastados entre si por uma distância maior que duas vezes o valor de **limite**. Essa suposição se justifica pela própria configuração da faixa projetada, já que os *pixels* que geram codificação decrescente são apenas aqueles que compõem a transição de uma faixa para outra e essa distância é sempre menor que a largura da própria faixa.

O vetor **quebra** contém a seqüência de pixels **S** com as seguintes propriedades:

- O valor de um *pixel* na seqüência é menor que o do imediatamente anterior, na linha sendo tratada.
- A distância entre dois *pixels* na seqüência é maior que **limite**.
- Não existe, lexicograficamente, uma seqüência com essas propriedades, anterior a **S**.

A seqüência contida no vetor **last** pode ser definida de forma análoga, com a diferença de que a distância entre dois *pixels* consecutivos (S_j e S_{j-1}) é menor que **limite**, e não existe um *pixel* à direita de S_j , com essa propriedade, que satisfaça simultaneamente a primeira condição acima citada.

Assim, os vetores **quebra** e **last** têm como objetivo tratar os intervalos onde a codificação decresce. O algoritmo trata apenas dos trechos onde existem elementos da superfície do objeto (valores maiores que zero). A figura seguinte mostra o significado das seqüências armazenadas nesses vetores.

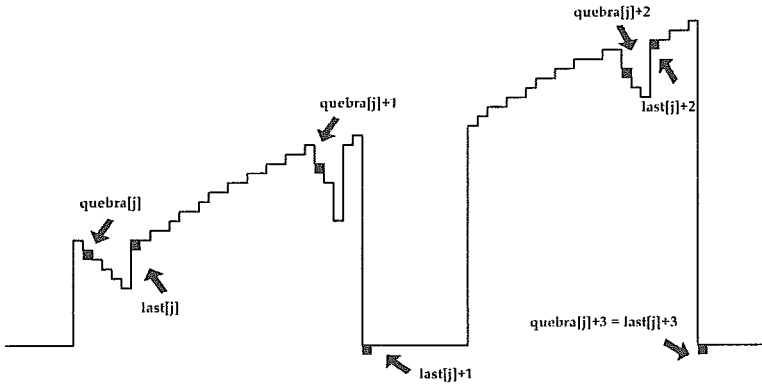


Figura 5.8: Gráfico de uma linha da imagem com erros a serem corrigidos

A figura seguinte apresenta o resultado da aplicação do algoritmo nesta linha.

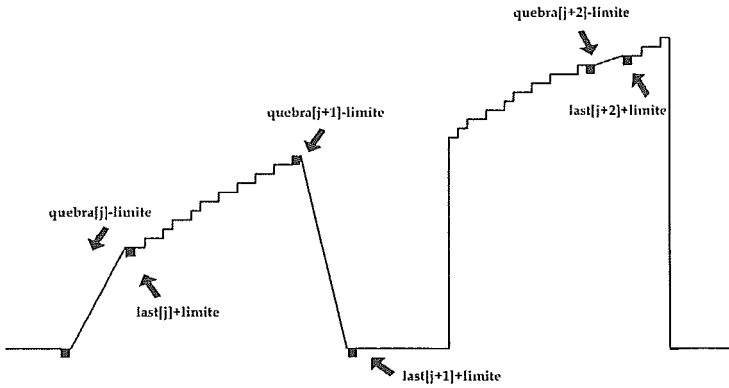


Figura 5.9: Gráfico de uma linha da imagem com erros já corrigidos

Baseando-se nas propriedades citadas anteriormente, temos que cada intervalo de forma $[m = last[j] - limite, n = quebra[j] + limite]$, contém inteiramente os valores dos *pixels* decrescentes e, além disso, não interceptam outros intervalos onde isso ocorre. Portanto, os valores em **m** e **n** podem ser considerados confiáveis.

Se m e n pertencerem ao objeto, os valores do intervalo $[m,n]$ (onde, valores de $n \geq$ valores de m) podem ser interpolados linearmente. Entretanto, pode ocorrer que m ou n pertençam ao fundo da cena. Nesse caso, optamos por fazer a interpolação linear, atribuindo aos *pixels* que representam o objeto e que estão contidos no intervalo¹, a maior codificação já obtida até ele.

Filtro *Top-hat*

Em sistemas onde todos os padrões são projetados em baixa resolução, cada faixa, mesmo no nível mais fino, aparece com a largura de vários *pixels* na imagem. Os *pixels* mais próximos das bordas das faixas estão sujeitos a erros de codificação e, assim, pode-se esperar variações normalmente pequenas, em intervalos onde os *pixels* deveriam receber o mesmo valor, pois representam a mesma faixa. Criam-se assim pequenas protuberâncias e reentrâncias no gráfico das codificações, o que torna adequada a utilização dos filtros *top-hat* [44]. Através deste tipo de filtro, *pixels* que deveriam possuir mesma codificação que os vizinhos são corrigidos, diminuindo assim as alterações mais visíveis na profundidade do objeto.

O filtro usa as duas iterações indicadas para tentar corrigir a codificação errada, onde são executados operadores de mínimo e máximo, constituindo a aplicação de um primeiro filtro *top-hat* para eliminar protuberâncias. Depois, são aplicados operadores de máximo e mínimo, compondo a passagem de um segundo filtro *top-hat*, que serve para eliminar reentrâncias. Na figura de exemplo, o resultado final ficou muito próximo do considerado ideal. Mesmo assim, alguns casos são de difícil correção, em especial as bordas do objeto ou da própria imagem.

Filtro para eliminação de *outliers*

A filtragem dos *outliers* existentes na nuvem de pontos do objeto escaneado foi especialmente necessária no método com padrão colorido. Porém, além da remoção dos *outliers*, existe também a necessidade de se suavizar algumas superfícies que apresentam um aspecto *fuzzy*, ou seja, com muitos pontos de valores próximos

¹ Na verdade, assumimos que esses *pixels* são todos aqueles entre m e o último *pixel* com valor não nulo no intervalo, como se pode constatar pelo último **ELSE** do algoritmo

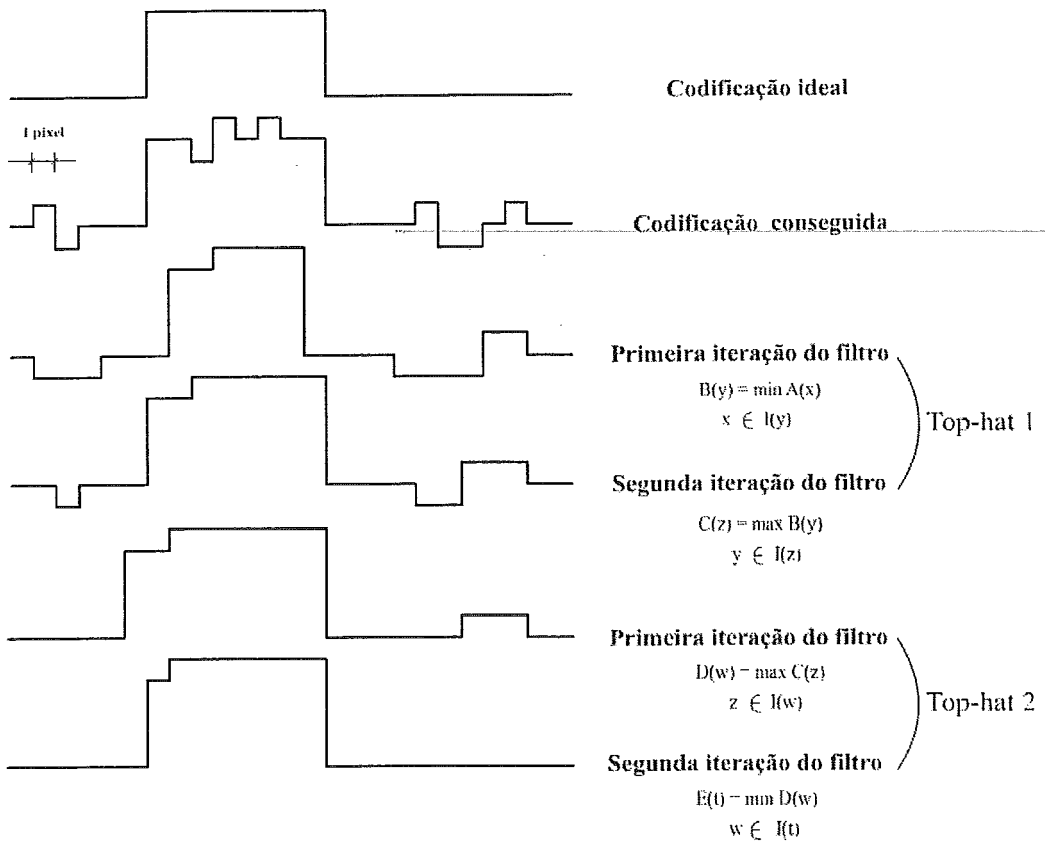


Figura 5.10: Atuação do filtro *top-hat* sobre a codificação

acima e abaixo da linha que define sua codificação no método *Gray-code* simples. Esse problema ocorre bastante no método *Phase-shifting*, principalmente porque este gera valores reais entre 0 e 1 que substituem o último *bit* da codificação dada pelo *Gray-code*. A figura abaixo ilustra o tipo de resultado que pode ser gerado, onde pode-se notar a necessidade de suavização da superfície. As setas indicam os planos associados às codificações de valor inteiro.

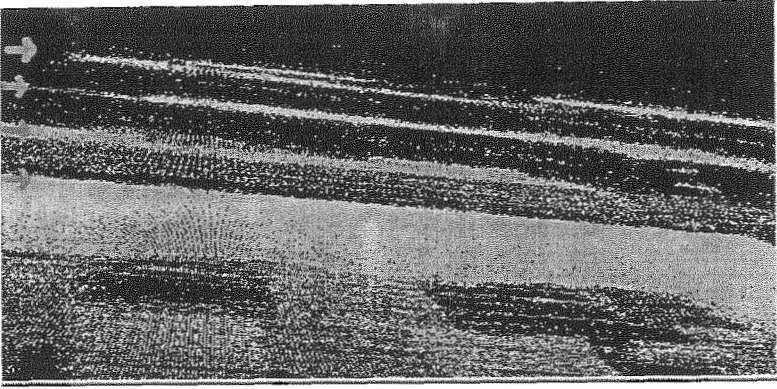


Figura 5.11: Camadas que identificam as codificações no método *Phase-shifting*

O método de escaneamento com cores também se valeu do filtro acima para suavizar os resultados obtidos. O seguinte algoritmo descreve o processo de interpolação destes pontos:

```

for ( i = 1; i < no. de colunas; i++ ) do
    H = 0;
    m = 1;
    for ( j = 1; j > no. de linhas; j++ ) do
        Aplicar filtro de suavização

```

O Filtro de suavização é demonstrado no seguinte algoritmo:

```

if ( i > m+1 ) then
    CoordZ[i - (m + 1)][j] = temp[(int) fmod(i, (m + 1))][j];
    temp[(int) fmod(i, (m + 1))][j] = CoordZ[i][j];
    for (int k = -m; k <= m; k++) do
        if (temp[(int) fmod(i, (m + 1))][j] == H) then
            return;
        for (intl = -m; l <= m; l++) do
            if (abs(CoordZ[i + k][j + l] - CoordZ[i][j]) >= Precisão) then
                temp[(int) fmod(i, (m + 1))][j] = H;
            return;
    H = temp[(int) fmod(i, (m + 1))][j];

```

No trecho acima, **H** representa o último valor determinado que foi considerado correto e **m** o raio da vizinhança que será utilizado pelo processo de suavização que, em nossos experimentos, foi igual a 5 *pixels*, ou seja, um *pixel* central, dois anteriores e dois posteriores. **Precisão** é um valor inteiro arbitrário, escolhido para definir a distância considerada como erro de codificação pelo algoritmo. Na maior parte dos experimentos usamos o valor 2. O parâmetro **temp** armazena valores dos *pixels* já alterados, sem que esses valores interfiram no cálculo do valor a ser atribuído a outros *pixels*. Vale ressaltar que, apesar de a filtragem por vizinhança ter interpolado pontos de forma a tornar as superfícies escaneadas mais uniformes, isto não implica necessariamente em uma diminuição nos erros apresentados. É claro que se os valores obtidos forem errados em uma seqüência de *pixels* maior que a vizinhança estabelecida para a aplicação do filtro de suavização, alguns desses valores permanecerão. Entretanto, o uso de vizinhanças maiores pode suavizar indevidamente os contornos do objeto.

Detecção de cores

No processo de codificação das imagens pelo método de projeção com padrões coloridos encontramos mais problemas nas faixas projetadas do que no método *Gray-code*. Como já foi citado, se um objeto possui a mesma cor de alguma faixa do padrão colorido projetado, a identificação desta faixa pode-se tornar impossível e, por isso, a escolha das cores projetadas deve ser adequada ao objeto, o que torna o processo de codificação dos pontos altamente dependente de um critério específico para a identificação de cores.

Em nosso sistema, foi necessária uma customização do procedimento de identificação de faixas, obtida de forma experimental, para que este pudesse contornar os problemas encontrados. Ainda que o objeto não possuísse nenhuma cor como aquelas projetadas, a percepção de suas cores pela câmera não era a esperada, em especial em relação ao canal vermelho. Outro problema em relação à identificação de cores, que foi agravado no caso colorido, foi o *blurring*, determinado pela câmera nas transições de cor entre as faixas. Esse *blurring* acarretou em uma falsa identificação da cor dos *pixels* de faixas vizinhas em algumas colunas próximas à borda, como pode ser visto na figura 5.12.

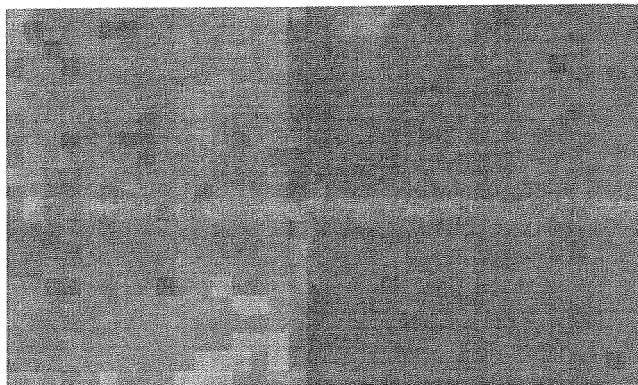


Figura 5.12: Problema provocado na borda de faixas coloridas pelo sistema de *anti-aliasing* da câmera.

A transição entre as faixas, que deveria ser uma única linha vertical, oscila numa faixa de três colunas na região próxima à transição. Além disso, os valores de RGB desta região de transição podem ser significativamente diferentes daqueles encontrados no meio das duas faixas, o que torna complicada a identificação das bordas das faixas, determinando assim a necessidade do uso de filtros.

Na figura seguinte pode-se identificar o outro problema de alteração das cores. Neste caso, as cores foram modificadas pela penumbra provocada pela auto-occlusão do objeto, como será explicado posteriormente. Nas regiões que tiveram sua cor alterada foram gerados valores errados de profundidade.

No que se refere à influência da cor do objeto nos resultados obtidos, foi interessante observar que quando esta cor era homogênea, foi possível identificar as cores do padrão projetado mais acertadamente. Para isso, usamos o critério em que cores mais confiáveis são identificadas primeiro. Essas cores, de um modo geral, eram aquelas cujos valores de RGB eram mais consistentes, tanto sobre o objeto, quanto para o fundo. Eventualmente, em situações de dúvida, os valores dos vizinhos podem ser utilizados na escolha da cor adequada de um determinado *pixel*.

O trecho de código a seguir exemplifica o tipo de customização necessária para a essa identificação de cores. Pode-se observar que as cores branca, preta e amarela se mostraram mais confiáveis, enquanto o magenta e o vermelho foram deixadas por último, sendo detectadas depois da codificação das outras.

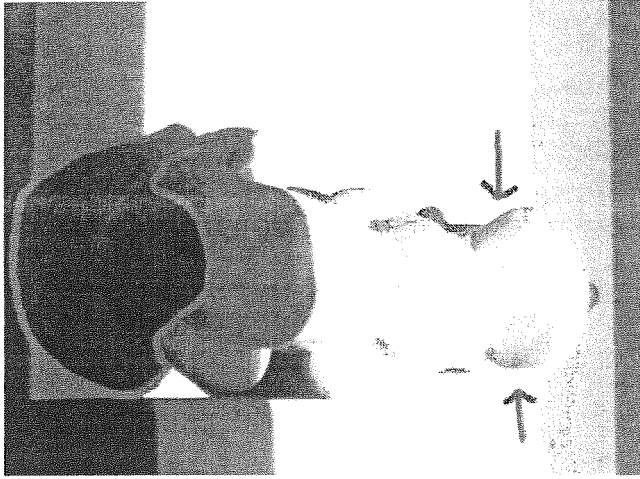


Figura 5.13: Regiões sujeitas a identificação errada de cores

```

if (R <= T & G <= T & B <= T)
  R = 0; G = 0; B = 0;
else if (R > T & G > T & B > T)
  R = 1; G = 1; B = 1;
else if (R > T & G > T)
  R = 1; G = 1; B = 0;
else if (G > R & R > B)
  R = 0; G = 1; B = 0;
else if (B > G & G > R)
  R = 0; G = 0; B = 1;
else if (G > R)
  R = 0; G = 1; B = 1;
else if (B >= R & R > G)
  R = 1; G = 0; B = 1;
else
  R = 1; G = 0; B = 0;

```

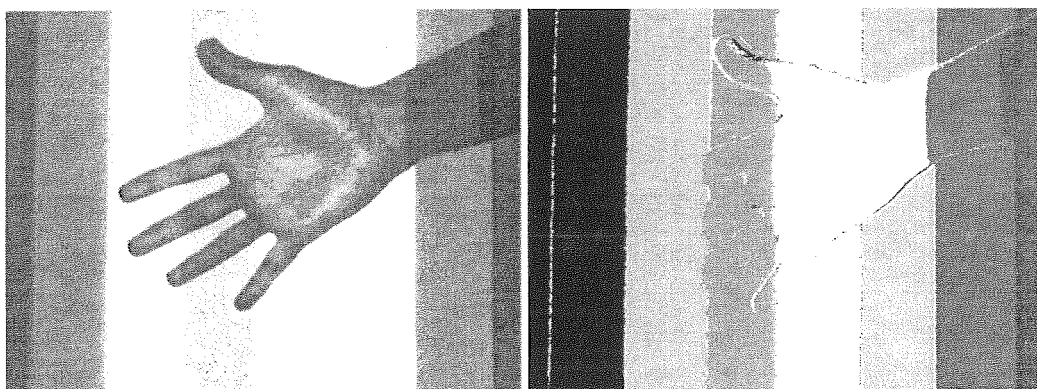
Esse procedimento foi usado para identificação das faixas sobre o objeto e sobre o plano de referência. O valor de T foi definido experimentalmente². Um procedi-

² Por exemplo, no caso da cor branca, esses valores foram R = 230, G = 240 e B = 240.

mento similar foi adotado para os outros dois padrões coloridos, que não possuíam a cor vermelha, onde se armazenava, obviamente, somente os valores de verde e de azul.

Um dos casos onde a identificação de cores necessitou deste tipo de customização é o do escaneamento da mão representada na figura 5.14. A pele clara da pessoa sendo escaneada modifica as cores projetadas, criando novos valores de cor para cada faixa, quando vistas pela câmera.

As figuras a seguir mostram a imagem vista pela câmera e a identificação feita pelo nosso sistema. Nota-se que este obteve sucesso na identificação das faixas, em especial a faixa de cor amarela, que se mescla à cor da pele. Além disso o sistema conseguiu identificar alguns trechos de borda que estão sob área de sombra ou penumbra.



(a)

(b)

Figura 5.14: Resultado da identificação de cores em nosso sistema

Nosso sistema também avalia a codificação das faixas do plano de referência de forma customizada. Os valores de T usados nesta identificação de cores são, obviamente, diferentes daqueles usados para a identificação de faixas sobre o objeto, o que justifica, no exemplo anterior, a presença de uma faixa azul no fundo da imagem enquanto a cor, desta mesma faixa, sobre o contorno da mão escaneada, é branca.

Detecção de sombras e penumbras

As partes consideradas de penumbra são aquelas afetadas pela iluminação ambiente ou pelo sombreamento parcial provocado pelo problema de auto-occlusão do objeto. Essas regiões não ocasionaram muitos erros nos exemplos onde usamos o método *Gray-code*, mas se mostraram problemáticas no método de projeção com cores.

A detecção de sombras e penumbras foi incorporada ao processo de segmentação do objeto, que passou a gerar, então, uma imagem binária que identifica essas regiões e as retira da etapa de codificação.

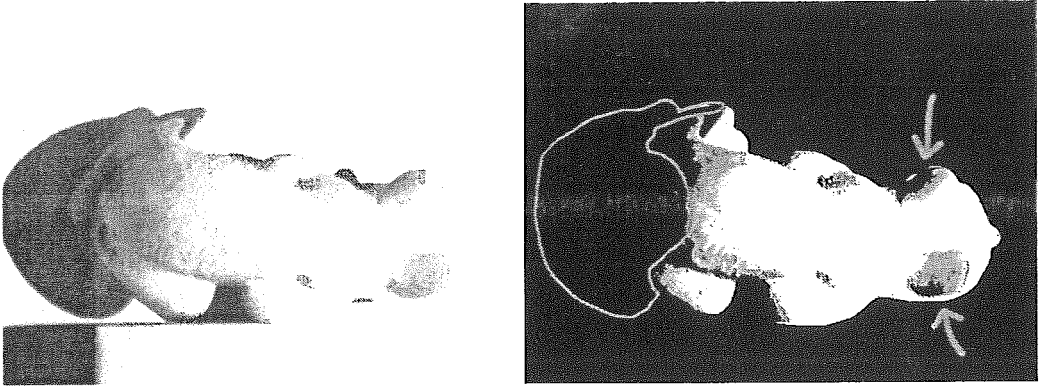
Para caracterizar as regiões de sombra e de penumbra, utilizamos vários critérios. Aquele que apresentou melhores resultados foi o de eliminar os *pixels* cujos valores $I_c(p)$ e $I_s(p)$, das imagens com e sem a iluminação do projetor, não atendessem, para algum canal, a seguinte condição:

$$\max(I_c(p), I_s(p)) \leq \frac{1}{N} \cdot \sum_{p \in I} \frac{I_c(p) + I_s(p)}{2} \quad (5.3)$$

onde N é o número de *pixels* e o lado direito representa a média das médias de brilho em I_c e I_s .

As figuras a seguir mostram uma imagem do objeto sob iluminação do projetor e o filtro para segmentação de sombras e penumbras resultante.

A operação de filtragem descrita permite diferenciar o que é sombra total (regiões onde todos os canais de cor são anulados) do que é penumbra (regiões onde um ou dois canais são anulados). As áreas de cor verde na figura 5.15 identificam as partes do objeto que possuem sombreamento parcial. As regiões circundadas de vermelho mostram trechos de sombra total e as setas indicam as penumbras. Os *pixels* correspondentes a essas áreas são desconsiderados quando da etapa de cálculo das coordenadas dos pontos. *Pixels* em áreas de sombreamento total são, é claro, de fácil detecção. Já os *pixels* parcialmente sombreados podem ter valores muito próximos de regiões bem iluminadas mas não o suficiente para que sejam codificados corretamente.



(a)

(b)

Figura 5.15: Imagem do objeto sob a luz do projetor e resultado da detecção

Como discutido na seção anterior, as regiões de penumbra alteram a cor dos *pixels* ao invés de simplesmente torná-los pretos, o que gera problemas ainda mais graves pois, se estes *pixels* fossem considerados na etapa de filtragem, seus valores afetariam diretamente os valores finais dos *pixels* vizinhos, propagando erros, como pode ser visto na imagem a seguir.

A partir do exemplo da figura 5.15, fica claro que se considerássemos as faixas de cor amarela como verdes, como apresentado na figura 5.13, nosso sistema nivelaria, com uma parte anterior do objeto, um trecho que deveria ter, como resultado da codificação, valores menores de profundidade. A retirada desses pontos é necessária visto ser difícil corrigí-los pelos filtros implementados em nosso sistema.

5.3.2 Visualização de *range-maps*

Em quase todos os sistemas de escaneamento existe uma etapa final de digitalização do objeto escaneado, onde o modelo resultante é estruturado na forma de pontos, *voxels* [45], malha de triângulos [10], curvas de nível das superfícies [46][1] ou *grid* [2].

Para a visualização dos modelos, o sistema desenvolvido neste trabalho apresenta

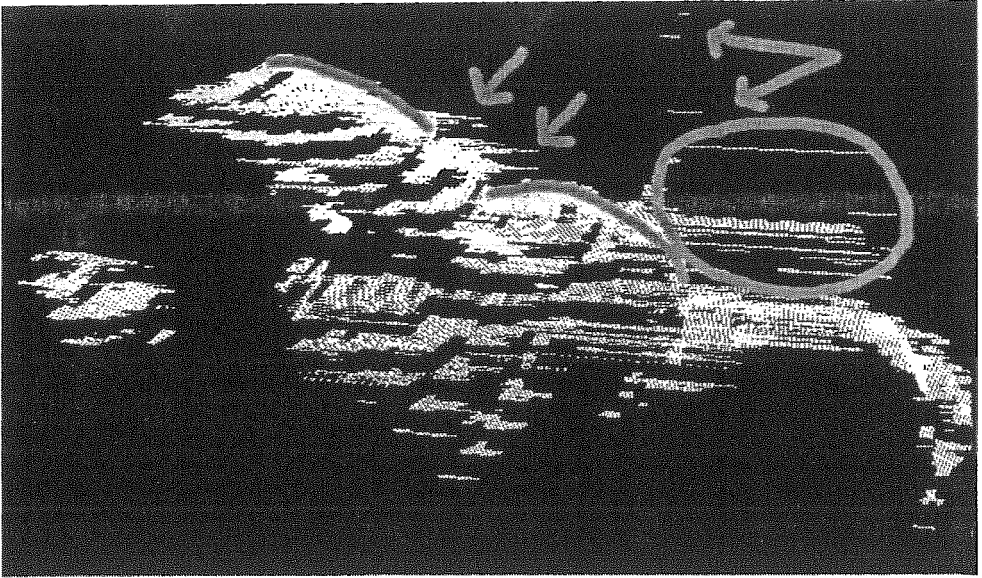
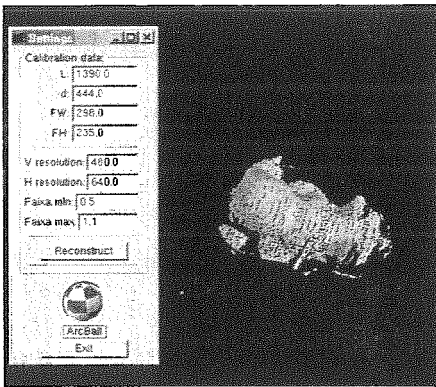
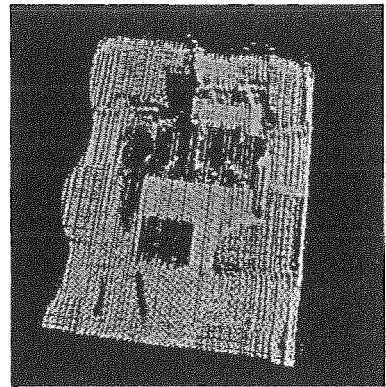


Figura 5.16: Contorno ideal do objeto e erro provocado por penumbra

a seguinte saída, apresentado nas figuras a seguir.



(a)



(b)

Figura 5.17: Interface do sistema e outro tipo de visualização

Outros sistemas existentes apresentam uma nuvem de pontos completa do objeto, obtida pela integração de vários *range maps*, ou o mapa de profundidade correspondente às imagens adquiridas em uma determinada posição. Nosso sistema

permite somente a visualização de um *range map* por vez, mas as coordenadas referentes a essa vista do objeto são armazenadas, e podem ser usadas posteriormente para uma etapa de registro e alinhamento da nuvem de pontos total do modelo.

A interface de nosso sistema foi implementada com o auxílio das bibliotecas GLU e GLUT, e toda a visualização do modelo foi feita com o auxílio da biblioteca gráfica OpenGL. Assim como apresentado por Hall-Holt [16] e [47], os resultados finais do processo de reconstrução são mostrados de forma a permitir a visão das camadas que compõem o objeto escaneado.

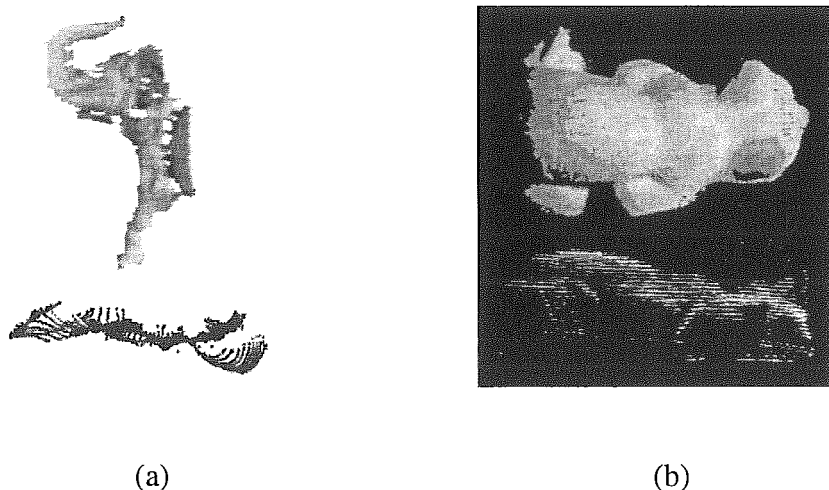


Figura 5.18: Imagem de objeto escaneado por Holt [16] e saída de nosso sistema

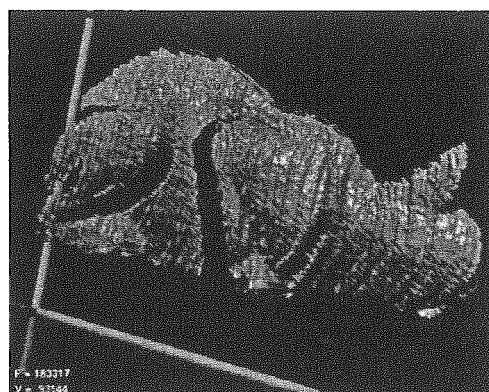


Figura 5.19: Saída gerada por um sistema de modelagem

Como já foi comentado, nosso programa também gera arquivos de saída em formato **.plg**, contendo as coordenadas dos pontos e os índices das faces, permitindo a visualização do modelo em diferentes sistemas de modelagem, como pode ser visto na figura 5.19.

Capítulo 6

Resultados

Durante as fases iniciais de implementação com o método *Gray-code*, tivemos dificuldades para encontrar uma disposição de *hardware* e fazer seu ajuste fino de forma a gerar resultados robustos. Em particular, a posição do projetor em relação ao objeto não permitia a visualização, de forma correta, das deformações que este provocava nas faixas. Em uma tentativa de aumentar a resolução do sistema, e obter reconstruções mais suaves, foi implementado o método de *Phase-shifting*. Infelizmente os resultados nos experimentos mostraram a dificuldade de se trabalhar com esse método com um equipamento de baixa precisão. A título de ilustração, os resultados conseguidos nesta fase também serão apresentados.

Em uma etapa posterior dos experimentos, o método *Gray-code* foi novamente usado mas, desta vez, o posicionamento do equipamento de aquisição foi ajustado de forma a se evitar os problemas de amostragem apresentados nos métodos anteriores. Apesar da baixa resolução de aquisição, o novo ajuste na posição do projetor se mostrou imprescindível para a melhoria dos resultados finais. Como já foi citado anteriormente, essa versão final da implementação não exigiu o uso de várias etapas de filtragem, apresentando poucos *outliers*, mesmo quando foram escaneados objetos reflexivos.

Procurando reduzir a quantidade de imagens necessárias para o processo de reconstrução, foi implementado, por último, o método de codificação por cor. Esse método, apesar dos inúmeros problemas já citados, conseguiu aproximar razoavel-

mente os objetos reconstruídos dos modelos originais mas, ainda assim, os resultados foram insatisfatórios do ponto de vista de robustez, mesmo empregando-se critérios de customização e de filtragem a cada experimento.

Os exemplos mostrados na seção 6.3, se encontram nos limites do equipamento usado, uma vez que resoluções mais finas, que gerariam reconstruções mais precisas e suaves, não podem ser conseguidas, pois a câmera empregada, como já foi explicado, não conseguia diferenciar devidamente faixas em resoluções acima de 640x480. Finalmente parece claro que quanto mais longe do plano de referência estiver o sistema, menos acuidade teremos na estimativa de profundidade.

Para a apresentação dos resultados conseguidos neste trabalho, serão usadas estatísticas, incluindo: número de pontos do modelo obtido, número de triângulos e erro no eixo de coordenadas **X**, **Y** e **Z**. Em todos os casos mostrados a seguir, o tempo de escaneamento de cada *range map* foi inferior a 30 segundos. Infelizmente os resultados apresentados pelo método *Phase-shifting* e pela primeira implementação do método *Gray-code* não se mostraram suficientemente robustos para que fossem triangulados e, por isso, esse dado não consta de suas tabelas.

Os erros nos eixos **X**, **Y** e **Z** serão indicados como a diferença, em **cm**, entre as medidas do objeto escaneado e os valores obtidos para o modelo adquirido. Como foi visto no capítulo 3, a coordenada **Z**, de um ponto **P** sobre o objeto, é calculada em função de **X**, e este é obtido a partir dos dados relativos à distância entre a câmera e o projetor e destes ao plano de referência (**D** e **L** respectivamente). Assim, a precisão¹ em **Z**, pode ser estimada da seguinte maneira:

$$Precisão_Z \leq 2 \cdot \left(\frac{(x+r) \cdot L}{(x+r) + D} - \frac{x \cdot L}{x + D} \right) \quad (6.1)$$

$$Precisão_Z \leq \frac{2r \cdot D \cdot L}{(x+r+D) \cdot (x+D)} \quad (6.2)$$

¹ O fator 2 na equação 6.1 reflete o fato da expressão indicada entre parênteses considerar tanto a cena com o objeto quanto sem ele.

Onde r é a largura entre as faixas nos limites da área visualizada do plano de referência. O valor máximo desta expressão (ponto de menor precisão) ocorre para:

$$x_{min} = \frac{Z_{min} \cdot D}{L - Z_{min}} \quad (6.3)$$

onde Z_{min} representa a distância do ponto visualizado na superfície do objeto mais próximo ao plano de referência. Substituindo x na equação anterior, temos:

$$PrecisãoZ \leq \frac{2r \cdot D \cdot L}{\left(\frac{Z_{min} \cdot D}{L - Z_{min}} + r + D\right) \cdot \left(\frac{Z_{min} \cdot D}{L - Z_{min}} + D\right)} \quad (6.4)$$

$$PrecisãoZ \leq \frac{2r \cdot (L - Z_{min})^2}{r \cdot (L - Z_{min}) + D \cdot L} \quad (6.5)$$

Então, se $k = Z_{min}/L$ e $r \ll D$, teremos:

$$PrecisãoZ \leq 2(1 - k)^2 \cdot \left(\frac{L}{D}\right) \cdot r \quad (6.6)$$

que para $k = 1/2$ se torna:

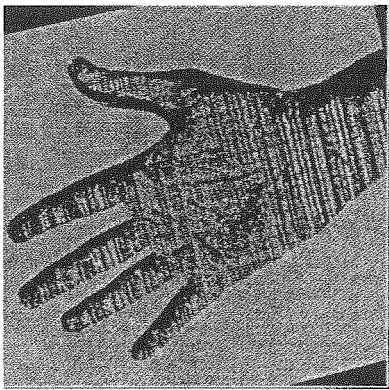
$$PrecisãoZ \leq \frac{r \cdot L}{2 \cdot D} \quad (6.7)$$

A dificuldade de se montar um *setup* adequado do equipamento pode ser analisada pela expressão 6.7. Primeiramente ela nos mostra os problemas criados pela inclinação do projetor. Quanto menos perpendicular em relação ao plano de referência este se encontra, maior o efeito de *broadening* nas faixas. Isso carreta num aumento do valor de r , diminuindo a precisão das coordenadas Z calculadas. O pior caso ocorre na borda da imagem mais distante do projetor, onde r é máximo. A precisão de Z apresentada nas tabelas dos itens a seguir considera a média dos valores de r por toda a imagem.

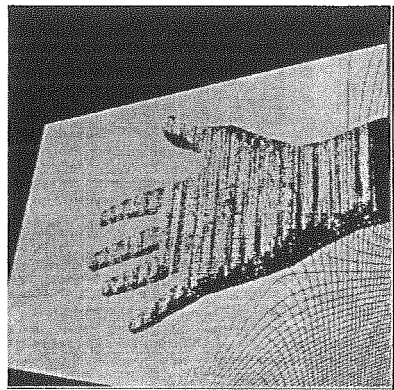
Além disso, temos o problema da distância ótima entre a câmera e o projetor, determinada experimentalmente. A expressão 6.7, apesar de ser decrescente em D , tem valores mínimos somente até onde a câmera consegue visualizar as faixas projetadas, ou seja, afastando-se a câmera do projetor acarreta em uma melhora da precisão de Z até o ponto onde as faixas visualizadas se tornam “borradas” e suas transições não podem mais ser identificadas.

6.1 *Gray-code e Phase Shifting*

Os resultados iniciais obtidos neste trabalho mostram que o sistema de escaneamento conseguiu recuperar, com relativa acuidade, os contornos dos objetos escaneados. O posicionamento quase perpendicular do projetor em relação ao plano de referência permitiu que os erros no eixo Y de coordenadas fossem pequenos se comparados aos erros dos outros métodos, como veremos a seguir. Porém, a pouca precisão conseguida para os valores de profundidade (Z) tornou as superfícies escaneadas bastante fragmentadas, impedindo uma reconstrução melhor do objeto. Os experimentos apresentados aqui usaram os seguintes parâmetros de calibração: $L = 810.00$, $D = 100.00$, $FH = 335.00$ e $FW = 236.00$.



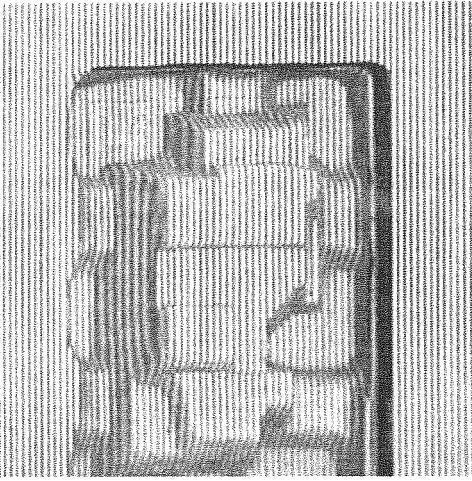
(a)



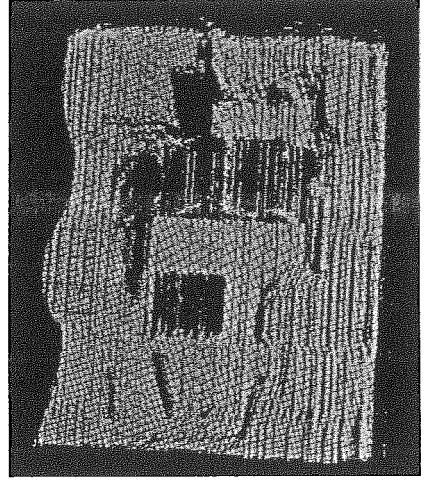
(b)

Figura 6.1: Duas vistas da mão escaneada

Outro fator importante nesta fase foi a ausência de correção dos valores de X , visto que a inclinação do projetor em relação ao plano era mínima. Ainda assim,



(a)



(b)

Figura 6.2: Isopor escaneado e resultado

Tabela 6.1: *Gray-code e Phase Shifting* - resultados finais

Modelo	Erro X	Erro Y	Erro Z	Precisão em Z	No. pontos
Mão	1.1cm	0.7cm	3.4cm	2.2cm	79.668
Isopor	2.0cm	0.1cm	3.0cm	2.2cm	84.088

os erros neste eixo de coordenadas são grandes em relação aos erros ocorridos nos outros métodos.

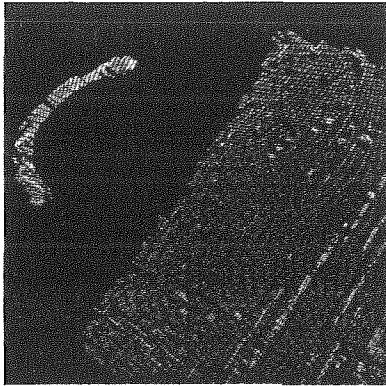
Esta fase inicial de implementação serviu, porém, para esclarecer dúvidas importantes em relação ao posicionamento do sistema de *scanning* e aos algoritmos de decodificação das faixas.

6.2 Codificação Colorida

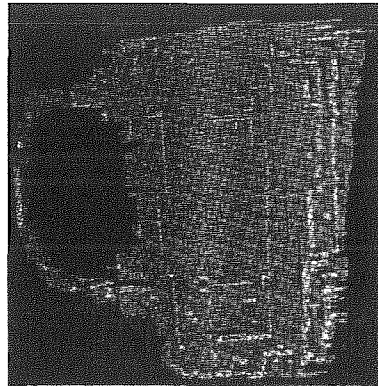
As figuras seguintes mostram os resultados dos mapas de profundidade obtidos pelo método de escaneamento colorido. Foram usados, como modelos, um gato de gesso (escaneado em duas posições) e uma caneca, ligeiramente reflexiva, ambos de cor branca. Neste experimento os valores de calibração foram $L = 855.00$, $D = 565.00$, $FH = 245.00$ e $FW = 325.00$.

Tabela 6.2: Codificação colorida - resultados finais

Modelo	Erro X	Erro Y	Erro Z	Precisão em Z	No. pontos	Triângulos
Gato - v. superior	2.9cm	0.22cm	0.9cm	0.71cm	46.595	90.150
Gato - v. lateral	1.53cm	1.8cm	1.3cm	0.71cm	61.227	118.869
Caneca	0.36cm	0.95cm	1.2cm	0.71cm	20.595	40.060

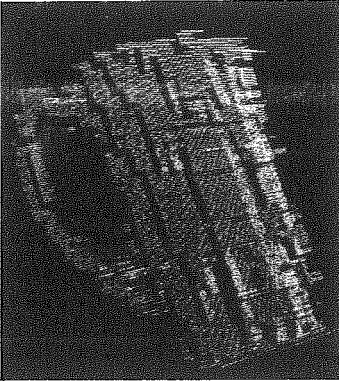


(a)

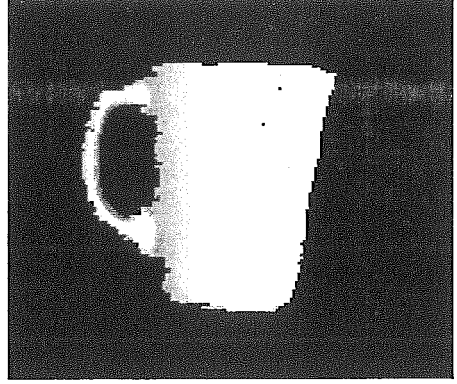


(b)

Figura 6.3: Imagens da caneca reconstruída



(a)



(b)

Figura 6.4: Níveis de codificação e *range map* com textura

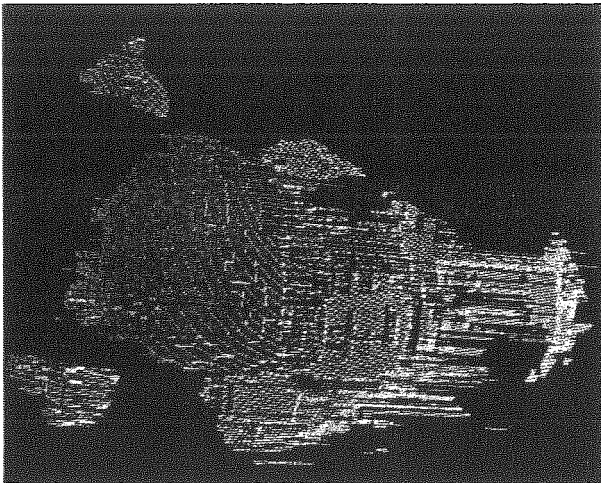
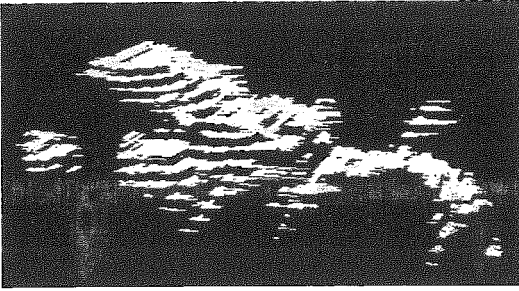
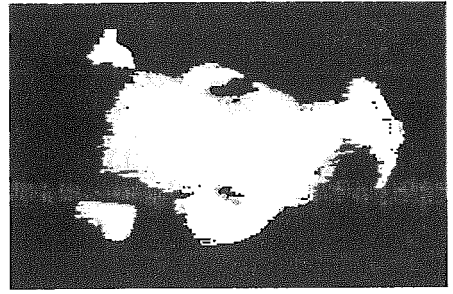


Figura 6.5: Vista aproximada do gato

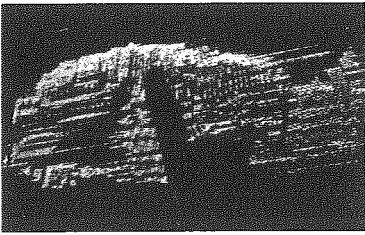


(a)

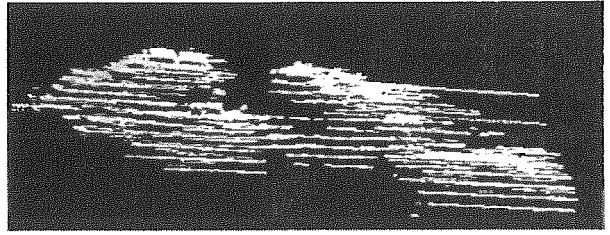


(b)

Figura 6.6: *Range map* com textura e camadas que compõem o gato

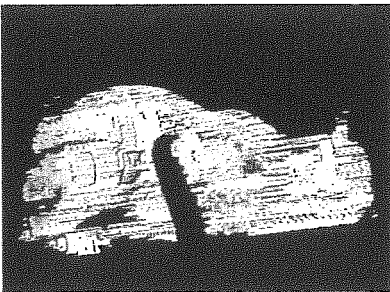


(a)



(b)

Figura 6.7: Camadas da vista lateral e *range map* com textura



(a)



(b)

Figura 6.8: Vistas laterais do gato

Apesar da precisão de **Z** ter sido maior no caso do método de codificação por cores, os resultados dos experimentos com esse tipo de padrão se mostraram menos satisfatórios, do ponto de vista de suavidade das superfícies, do que os apresentados pelo método *Gray-code* anterior. A principal justificativa para isso é o fato do sistema de decodificação dos padrões coloridos ser mais impreciso do que no caso preto e branco, em razão de todos os problemas gerados pelo equipamento, como discutido no capítulo 5.

6.3 Gray-code, versão final

Os últimos resultados apresentados serão os do método *Gray-code* em sua versão final. Para uma comparação mais estreita entre os métodos utilizados, foram escaneados os mesmos objetos do método de codificação com cores, com os valores 890.00, 444.00, 235.00 e 298.00, para **L**, **D**, **FH** e **FW**, respectivamente.

Esta fase posterior de implementação gerou resultados muito mais aceitáveis do ponto de vista de robustez, se comparada à fase inicial apresentada em 6.1, como pode ser visto pelas figuras a seguir. Deve-se observar, com especial atenção, a maior suavidade apresentada pelas superfícies. Os erros em **X** apresentados aqui são insignificantes se comparados aos outros métodos, devido à correção do *broadening* das faixas projetadas. Infelizmente, os erros em **Y** aumentaram por causa da acentuada inclinação do projetor.

Tabela 6.3: *Gray-code*- resultados finais

Modelo	Erro X	Erro Y	Erro Z	Precisão em Z	No. pontos	Triângulos
Gato - v. superior	0.01cm	0.7cm	0.8cm	0.8cm	86.318	169.034
Gato - v. lateral	0.01cm	1.25cm	1.2cm	0.8cm	93.150	183.337
Caneca	0.02cm	1.0cm	0.8cm	0.8cm	30.619	60.306

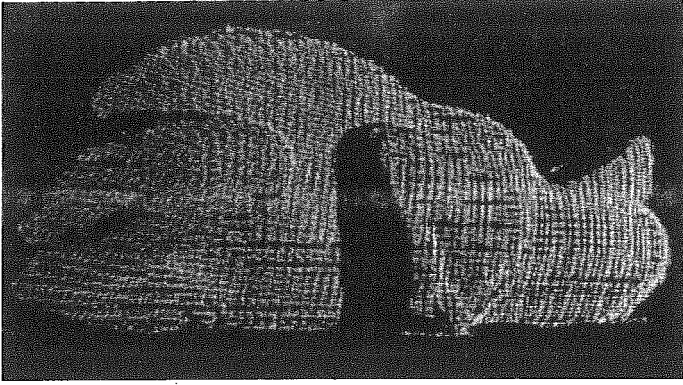


Figura 6.9: Vista lateral

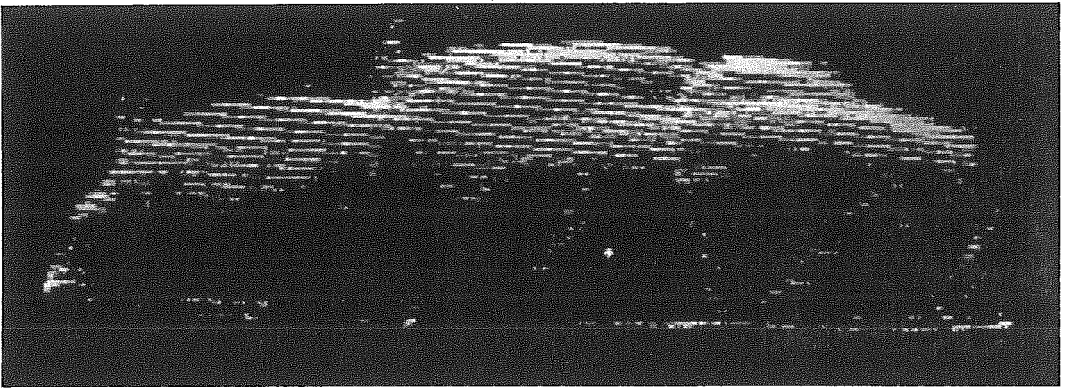
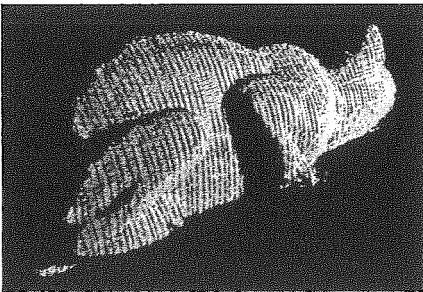


Figura 6.10: Camadas que compõem a vista lateral

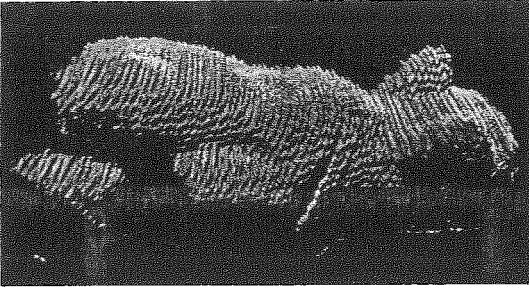


(a)

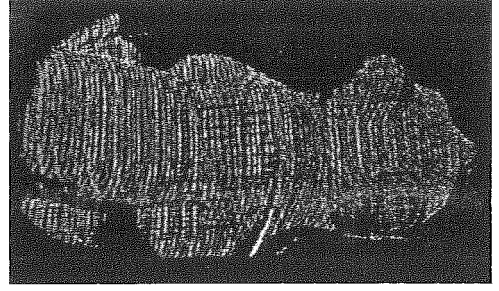


(b)

Figura 6.11: Outra vista lateral e *range map* com textura

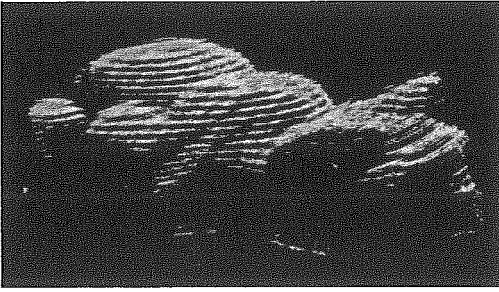


(a)

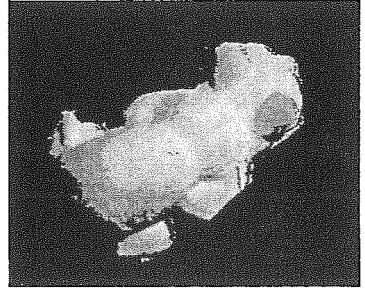


(b)

Figura 6.12: Vista superior do gato

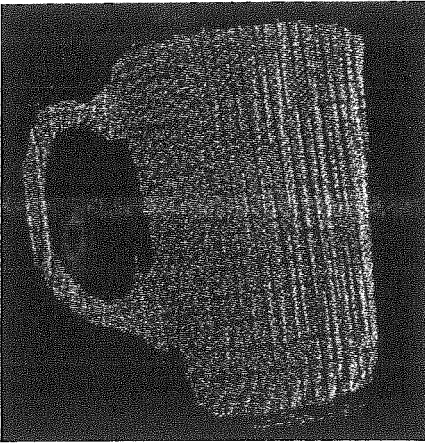


(a)

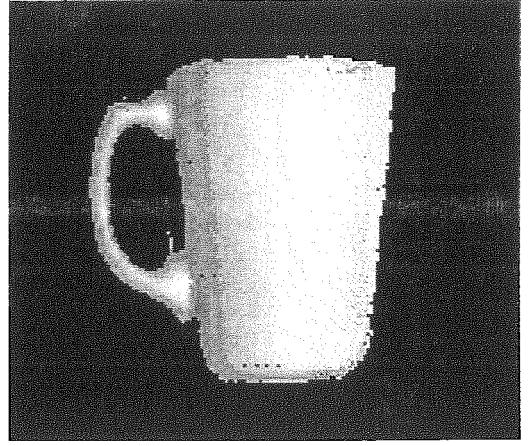


(b)

Figura 6.13: Camadas que compõem o gato e *range map* com textura



(a)



(b)

Figura 6.14: Vista aproximada e caneca com textura

6.4 Análise comparativa com outros sistemas

Os resultados apresentados neste capítulo revelaram os vários problemas encontrados durante a etapa de aquisição das imagens. Da implementação inicial do método *Gray-code* até sua versão final, os experimentos sempre apontavam para correções no posicionamento ou no ajuste fino do *hardware* utilizado.

Dentre os objetivos alcançados, o principal foi o de conseguirmos reconstruir mapas de profundidade de objetos, com forma e textura variadas, utilizando poucas imagens. A reconstrução dos modelos pelo método colorido merece, neste sentido, especial atenção, visto o número de imagens ser muito pequeno (apenas três) e o número de pontos adquiridos ser grande, quando comparamos com outros sistemas considerados de baixo custo [19]. Trabalhos recentes, que utilizam algoritmos específicos na detecção de cores das faixas projetadas, têm conseguido excelentes resultados com apenas uma imagem do objeto [26]. Porém, a comparação com esse tipo de sistema é impossível visto não utilizarmos equipamentos de alta resolução ou especialmente projetados para o processo de *scanning*.

Ao levantarmos a questão sobre número de imagens necessárias para a

reconstrução de um mapa de profundidade, é pertinente avaliarmos também o número de pontos codificados de forma válida em cada uma dessas imagens. Quanto maior o número de pontos corretamente identificados e calculados, menor o número de imagens a serem obtidas para a reconstrução total ou mesmo parcial do objeto. No caso do *range map* lateral do gato de gesso escaneado pelo método *Gray-code* final, por exemplo, nosso sistema conseguiu identificar aproximadamente 93.000 pontos com apenas 7 imagens. Boughet [14] utilizou 670 imagens, de um objeto de volume próximo ao gato escaneado, para obter em torno de 15.000 pontos para um *range map*, enquanto Holt [16] precisou de 644 *range maps*, e 16.850 imagens, para reconstruir totalmente um objeto com aproximadamente metade do volume do gato, tendo uma amostragem média de apenas 3.680 pontos para cada mapa de profundidade.

Outra comparação relevante é a diferença entre a precisão das profundidades obtidas para os pontos em nosso sistema, e a obtida por outros sistemas considerados não comerciais. Sob este aspecto, os resultados conseguidos neste trabalho podem ser comparados somente àqueles apresentados por Boughet [18], visto este não se utilizar de nenhum equipamento específico para o escaneamento tridimensional. Neste último trabalho, o autor indica vários erros com valores entre 0.065 e 1.0 **cm** para **Z**. Nossa implementação, como foi visto, gerou erros entre 0.9 e 1.3 **cm** para o método colorido e entre 0.8 e 1.2 **cm** para o método *Gray-code*. Vale ressaltar que os modelos gerados por Boughet são extremamente robustos e aproximam bem as superfícies escaneadas, enquanto em nosso trabalho apenas o método *Gray-code* gerou resultados aceitáveis.

Um aspecto fundamental no desenvolvimento do trabalho foi o de compreender o funcionamento dos equipamentos de *scan* e suas interdependências. O ajuste do posicionamento da câmera e do projetor foi um fator primordial para o aumento da precisão dos modelos finais. Perona [9], por exemplo, cita que só conseguiu obter melhores resultados após retirar a proteção externa da lâmpada do refletor que utilizou para iluminar os objetos escaneados. Este tipo de problema é um exemplo claro das dificuldades encontradas quando se trabalha com equipamentos de baixo custo.

Capítulo 7

Conclusões e trabalhos futuros

Neste trabalho foram apresentadas três técnicas de escaneamento tridimensional usando luz estruturada e os algoritmos necessários para a geração de *range maps* utilizando *hardware* de baixo custo.

Em uma primeira etapa foi implementado um sistema que une os métodos *Gray-code* e *Phase-shifting* de codificação para o escaneamento dos objetos. Posteriormente foi implementado um sistema de codificação por cores que permite uma diminuição no número de padrões de projeção necessários para se obter os valores de profundidade dos pontos sobre a superfície do objeto. Em uma fase final, foi implementado um sistema que utiliza somente o método *Gray-code* para obtenção de pontos.

A implementação inicial permitiu a identificação de parte dos problemas relacionados com o equipamento utilizado e, além disso, forneceu subsídios para que fossem implementados algoritmos para análise e correção do sistema de decodificação dos padrões projetados. O método de codificação colorida, devido aos erros no processo de aquisição das imagens, exigiu o uso de vários algoritmos de filtragem e identificação dos padrões de projeção, o que permitiu a avaliação, de forma mais precisa, do nível de influência do equipamento usado sobre os resultados gerados. Os resultados nesta fase foram pouco satisfatórios em relação à robustez do sistema. A última etapa do trabalho, com o método *Gray-code*, gerou *range maps* mais robustos e que aproximam melhor os objetos escaneados.

Trabalhos recentes [16] [19] [26] apresentam excelentes resultados, porém se utilizam de equipamentos que, ainda que possam ser considerados de baixo custo, são especialmente projetados para o processo de escaneamento. Boughet [14] demonstrou ser possível gerar resultados robustos usando equipamento não especializado (*off-the-shelf*). Sua implementação, como foi discutido, necessita de uma amostragem elevada (aproximadamente 500 imagens) para a obtenção de um mapa de profundidade do objeto escaneado. Neste sentido, nosso trabalho pode servir como base para a análise do uso de técnicas de escaneamento com poucas projeções.

Os algoritmos de identificação de cores e de filtragem de códigos apresentados neste trabalho servem de contribuição para vários tipos de implementação com luz estruturada, podendo ser adaptados para outros sistemas que utilizem *hardware* de baixo custo. Em relação aos sistemas considerados não-comerciais, o atual estado da arte em escaneamento 3D aponta para algumas direções em termos de trabalhos futuros a serem abordados, tais como:

- Automatização do processo de escaneamento permitindo a aquisição de todos os pontos que compõem o objeto, e não somente os pontos de uma determinada vista.
- Registro em tempo real da nuvem de pontos adquirida, facilitando a integração das malhas parciais obtidas.
- *Merging de range maps* que considerem técnicas de preenchimento de “buracos” (*hole filling*) na malha final do modelo digital.
- Aquisição dos pontos em alta resolução de forma a se obter precisão *sub-pixel* para seus valores de profundidade.

Além destas indicações, nosso trabalho pode, mais especificamente, visar os seguintes melhoramentos:

- Filtragem mais eficiente dos *outliers* nos *range maps* obtidos.
- Aumento da precisão e correção de erros nos casos onde o posicionamento do equipamento influenciou os valores de reconstrução.

- Decodificação em tempo real dos padrões sobre objeto sendo escaneado.

Nesse último caso, podemos sugerir o uso de uma *turn-table* para a obtenção de todos os pontos que compõem o objeto, como sugerido por Sansoni[43]. Para esse tipo de aquisição é necessário tratar as seguintes questões:

a) Entre a projeção de dois padrões usados para compor o códigos dos raios, o objeto se move. Entretanto, na grande maioria dos experimentos reportados na literatura esse movimento é considerado desprezível devido a fatos como a velocidade do processo de projeção/aquisição ser muito maior que a de rotação do objeto. Assumindo que trabalhamos com um *hardware* menos eficiente, podemos comparar essa alternativa com os resultados de uma outra mais elaborada, onde os códigos são obtidos como se não houvesse movimento. Tendo as profundidades obtidas a partir desses códigos, identificamos *pixels* em imagens obtidas sob padrões sucessivos que corresponderiam ao mesmo elemento de superfície, caso essas profundidades estivessem corretas. Feita essa identificação novos códigos são obtidos e o processo é repetido quantas vezes se queira (em geral apenas mais uma ou duas vezes) ou até que se satisfaça algum critério de convergência. Dado que isso só depende do *set-up* (câmera/projetor/mesa giratória) podemos, numa etapa de pré-processamento associar a cada *pixel* um conjunto de candidatos, todos sobre uma mesma reta, a serem o correspondente a ele em cada uma de duas ou três imagens imediatamente anteriores. Tendo essa informação, podemos agilizar a forma iterativa de computar os códigos indicada acima. Além disso, para estabelecermos corretamente pares de *pixels* correspondentes em imagens distintas precisamos ter controle sobre as situações de auto-occlusão, o que implica, em particular que vamos precisar acompanhar a evolução das descontinuidades internas do objeto e não só da sua silhueta.

b) A parte da imagem, onde está a informação nova em relação à tomada anterior, está contida numa vizinhança da borda do objeto nessa última tomada. Restringir o tratamento a uma faixa em torno dessa borda agilizaria consideravelmente o processo de reconstrução. Entretanto, pode acontecer que o conjunto de cores que se projeta sobre essa faixa fosse muito reduzido tornando a reconstrução menos robusta em relação a obtida por um processo em que a profundidade de um elemento da superfície do objeto é computada sucessivas vezes, a partir de imagens onde esse elemento é iluminado por cores diferentes, devido à rotação. Diversos experimentos tomando amostras nas bordas e espaçadamente no interior da ima-

gem do objeto devem ser realizados para tentar resolver de forma satisfatória esse *trade-off* entre rapidez e confiabilidade.

Referências Bibliográficas

- [1] N. C. KWANG, Philip L. WORTHINGTON, and Edwin R. HANCOCK. Facial pose using shape-from-shading. *10th British Machine Vision Conference*, September 1999.
- [2] G. SANSONI et al. 3d imager for dimensional gauging of industrial workpieces: State of the art of the development of a robust and versatile system. *IEEE Transactions on Instrum. Meas.*, 1997.
- [3] M. LEVOY et al. The digital michelangelo project: 3D scanning of large statues. In Kurt Akeley, editor, *Siggraph 2000, Computer Graphics Proceedings*, pages 131–144. ACM Press / ACM SIGGRAPH / Addison Wesley Longman, 2000.
- [4] Gaël NEUEZ. Disponível na internet, December 2002. <http://www.s2.chalmers.se/research/image/Research/facescan/details.htm>.
- [5] C. ROCCHINI, Paulo CIGNONI, C. MONTANI, P. PINGI, and Roberto SCOPIGNO. A low cost 3d scanner based on structured light. In A. Chalmers and T.-M. Rhyne, editors, *EG 2001 Proceedings*, volume 20(3), pages 299–308. Blackwell Publishing, 2001.
- [6] Brian CURLESS and Steven SEITZ. 3d photography. ACM Siggraph '00 Course Notes No. 19, August 2000.
- [7] Fausto BERNARDINI and Holly RUSHMEIER. The 3d model acquisition pipeline. *Eurographics - Computer Graphics Forum*, 21(2):149–172, 2000.
- [8] Brian CURLESS and Marc LEVOY. A volumetric method for building complex models from range images. *Computer Graphics*, 30(Annual Conference Series):303–312, 1996.

- [9] Markus WEBER Jean-Yves BOUGHET and Pietro PERONA. What do planar shadows tell us about scene geometry?, 1998.
- [10] G. HAUSLER and S. KARBACHER. Reconstruction of smoothed polyhedral surfaces from multiple range images, 1997.
- [11] Hugues HOPPE, Tony DeROSE, Tom DUCHAMP, John McDONALD, and Werner STUETZLE. Surface reconstruction from unorganized points. *Computer Graphics*, 26(2):71–78, 1992.
- [12] F. BERNARDINI, J. MITTLEMAN, H. RUSHMEIER, C. SILVA, and G. TAUBIN. The ball-pivoting algorithm for surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, 5(4):349–359, 1999.
- [13] G. CONG and B. PARVIN. Shape from interference patterns, 1998.
- [14] Jean-Yves BOUGHET and Pietro PERONA. 3d photography using shadows in dual-space geometry, 1999.
- [15] R. ZHANG, P.S. TSAI, J.E. CRYER, and M. SHAH. Shape from shading: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(8):690–705, August 1999.
- [16] Olaf HALL-HOLT and Szymon RUSINKIEWICZ. Stripe boundary codes for real-time structured-light range scanning of moving objects. pages 359–366, 2000.
- [17] M. TROBINA. Error model of a coded-light range sensor. Technical report, 1995.
- [18] Jean-Yves BOUGHET and Pietro PERONA. 3d photography on your desk. In *ICCV*, pages 43–52, 1998.
- [19] Szymon RUSINKIEWICZ, Olaf HALL-HOLT, and Marc LEVOY. Real-time 3d model acquisition, 2002.
- [20] Georg WIORA. High resolution measurement of phase-shift amplitude and numeric object phase calculation. In *SPIE Proceedings Vision Geometry IX*, pages 289–299, 2000.
- [21] CYBERWARE. Disponível na internet, December 2002. <http://www.cyberware.com>.

- [22] Capture 3D. Disponível na internet, December 2002. <http://www.capture3d.com/products.htm>.
- [23] MINOLTA. Disponível na internet, December 2002. <http://www.3dscanner.ch>.
- [24] RSI GmbH 3D Systems Software. Disponível na internet, December 2002. http://www.rsi.gmbh.de/rsihome_e.htm.
- [25] K.L. BOYER and A.C. KAK. Color-encoded structured light for rapid active ranging. *T-PAMI*, 9:14–28, 1987.
- [26] Li ZHANG, Brian CURLESS, and Steven SEITZ. Rapid shape acquisition using color structured light and multi-pass dynamic programming. *1st International Symposium on 3D Data Processing Visualization and Transmission (3DPVT'02)*, July 2002.
- [27] Diff Scan. Disponível na internet, December 2002. <http://home.earthlink.net/scan3d/>.
- [28] KREON Technologies. Disponível na internet, December 2002. <http://www.kreon3d.com>.
- [29] Roland DGA. Disponível na internet, December 2002. <http://www.rolanddga.com>.
- [30] AXILA Inc. Disponível na internet, December 2002. <http://www.axila.com/UKgscan.htm>.
- [31] GEOMETRIX Inc. Disponível na internet, December 2002. <http://www.geometrix.com/facevision/index.html>.
- [32] COGNITENS 3D Vision Systems. Disponível na internet, December 2002. <http://www.cognitens.com>.
- [33] 3D Digital Corp. Disponível na internet, December 2002. <http://www.3ddigitalcorp.com/industrial.htm>.
- [34] M. J. BROOKS and B. K. P. HORN. Shape and source from shading. In B. K. P. Horn and M. J. Brooks, editors, *Shape from Shading*, pages 53–68. MIT Press, Cambridge, MA, 1989.

- [35] A.G. JONES and C.J. TAYLOR. Robust shape from shading. *IVC*, 12(7):411–421, September 1994.
- [36] K. PULLI, T. DUCHAMP, H. HOPPE, J. McDONALD, L. SHAPIRO, and W. STUETZLE. Robust meshes from multiple range maps, 1997.
- [37] K. PULLI. Multiview registration for large data sets, 1999.
- [38] Tower Graphics. Disponível na internet, December 2002. http://www.towergraphics.it/products_easy3D.php.
- [39] X. JIANG and H. BUNKE. Range data acquisition by coded structured light: Error characteristic of binary and gray projection code. *Optical 3-D Measurement Techniques II*, pages 386–393, 1993.
- [40] J. HEIKILLÄ. Geometric camera calibration using circular control points. *EEE Transactions on Pattern Analysis and Machine Intelligenc*, 22:1066–1077, October 2000.
- [41] Jean Yves BOUGHET. Disponível na internet, December 2002. http://www.vision.caltech.edu/bouguetj/calib_doc/#system.
- [42] G. SANSONI and R. RODELLA. Fast digitization of heritage objects by means of a 3d vision system based on the projection of structured light. *Canada-Italy Workshop: Heritage Applications of 3D Digital Imaging*, October 1999.
- [43] G. SANSONI and R. RODELLA. 3d shape recovery and registration based on the projection of non-coherent structured light. 2000.
- [44] Michel COSTER and Jean-Louis CHERMANT. *Précis D'Analyse D'Image*. Centre National de la Recherche Scientifique, first edition, October 1985.
- [45] K. KUTULAKOS and S. SEITZ. What do n photographs tell us about 3d shape? *Technical Report TR680, Computer Science Dept., U. Rochester, 1998*, 1998.
- [46] Philip L. WORTHINGTON, Benoit HUET, and Edwin R. HANCOCK. Appearance-based object recognition using shape-from-shading. *The British Machine Vision Conference - BMVA98*, 1998.

[47] Brian Lee CURLESS. New methods for surface reconstruction from range images. Technical Report CSL-TR-97-733, 1997.