José Luiz Ribeiro de Souza Filho

Alinhamento de nuvens de pontos adquiridos através de digitalizador câmera-projetor com luz estruturada

Juiz de Fora

02 de dezembro de 2010

José Luiz Ribeiro de Souza Filho

Alinhamento de nuvens de pontos adquiridos através de digitalizador câmera-projetor com luz estruturada

Orientador:
Marcelo Bernardes Vieira

UNIVERSIDADE FEDERAL DE JUIZ DE FORA
INSTITUTO DE CIÊNCIAS EXATAS
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

Juiz de Fora

02 de dezembro de 2010

Monografia submetida ao corpo docente do Instituto de Ciências Exatas da Universidade Federal de Juiz de Fora como parte integrante dos requisitos necessários para obtenção do grau de bacharel em Ciência da Computação.

Prof. Marcelo Bernardes Vieira
Orientador
Departamento de Ciência da Computação

Prof. Rodrigo Luis de Souza da Silva
Departamento de Ciência da Computação

Prof. Maicon Ribeiro Correa
Departamento de Ciência da Computação

Agradecimentos

Agradeço primeiramente à minha mãe, Elizabete, por fazer todos os esforços para que eu atingisse mais esta etapa da minha vida e por me tornar um ser capaz de alcançar grandes realizações. Ao meu pai, José Luiz, que continua a me guiar de onde quer que esteja e que assim como a minha mãe sempre será um ídolo e referência. À minha namorada, Miriam, por apoiar e incentivar todas as minhas decisões e me ouvir quando preciso. Aos meus irmãos e toda a minha família por tolerar a minha ausência durante esta caminhada.

Agradeço também aos meus colegas de classe, pois juntos soubemos enfrentar todas as dificuldades e criamos um ambiente de sincera amizade. Aos membros do Grupo de Computação Gráfica e principalmente ao Roger Correia, por colaborar no desenvolvimento deste trabalho. Agradeço a todos os professores do Departamento de Ciência da Computação e especialmente ao meu orientador, professor Marcelo Bernardes, por acreditar na minha capacidade e ajudar na construção do meu conhecimento.

Sumário

Lista de Figuras

Lista de Tabelas

Resumo

Este trabalho apresenta o problema de alinhamento entre diferentes nuvens de pontos. São apresentados os conceitos relacionados à digitalização de cenas utilizando padrões de luz estruturada em um sistema composto de câmera e projetor de alta definição. Detalha-se um algoritmo que tem como objetivo minimizar distâncias entre conjuntos de pontos arbitrários, chamado *Iterative Closest Points*. São descritas modificações propostas para este algoritmo especificamente para tratar de nuvens de pontos adquiridas por digitalizadores tridimensionais. Estas adaptações visam melhorar o desempenho e precisão do algoritmo original.

1 *Introdução*

A tecnologia 3D não é uma novidade dos dias atuais. Com os avanços tecnológicos e devido à apreciação em torno do mundo, investimentos tornam-se cada vez mais crescentes, apresentando sempre novas utilidades e dispositivos.

Imagens e cenários que apresentem muitos detalhes são de interesse para engrandecer o setor de entretenimento. Jogos para impressionar e atrair cada vez mais pessoas tentam aproximar-se ao máximo da realidade, criando além de ambientes com maior grau de movimento, efeitos e texturas realistas. A difusão de cinemas e aparelhos televisivos com maior grau de imersão do espectador nas atrações apresentadas também cria a necessidade de exibições e animações convincentes para proporcionar um melhor divertimento.

Além da importância destes recursos no setor de entretenimento, há seu papel no ramo científico. Representações precisas de objetos do mundo real são de grande valia para estudos sendo, por exemplo, utilizados em simuladores em diferentes áreas de pesquisa. Nuvens de pontos bem refinadas são muitas vezes difíceis de serem adquiridas e geralmente precisam de tratamento adequado.

1.1 Visão Geral de Digitalizadores 3D

Digitalizadores 3D são dispositivos para extração de características geométricas de objetos e cenas. São desenvolvidos digitalizadores especificamente para a extração de dados de interesse, como pontos no espaço e suas normais. Esses elementos geométricos visam atender variados objetivos como, por exemplo, reconstrução de superfícies (HOPPE et al., 1992), (CHANG; LEYMARIE; KIMIA, 2009) e análises estruturais (MARTINI, 2001). Neste trabalho utiliza-se um sistema binocular composto de uma câmera e um projetor de alta definição, mas outras abordagens são apresentadas em (MANDOW et al., 2010) e (GAMBINO et al., 2005).

Projeções provenientes de um projetor digital realizadas sobre superfícies sofrem distorções

de acordo com as formas das mesmas. A projeção de texturas que evidenciem as geometrias dos objetos possibilitam uma detecção rápida e robusta. Como o sistema câmera-projetor possui planos projetivos bem definidos, obtém-se a regularidade necessária para detectar rapidamente transições entre planos.

O maior problema desse tipo de sistema é determinar as correspondências entre os planos de projeção da câmera e do projetor e conseguir a representação de um ambiente 3D a partir de imagens 2D. Algumas outras características do ambiente também são importantes para uma melhor apresentação das imagens como, por exemplo, a não interferência de variadas fontes externas de luz que retirem a uniformidade da iluminação criada pelo projetor.

Os objetos presentes nos quadros obtidos podem apresentar peculiaridades que resultem em alguma dificuldade ou imprecisão na aquisição dos pontos tridimensionais. Características de muita reflexão ou mesmo refração, como presentes em fluidos e objetos transparentes, podem retornar impressões de falsa profundidade.

Todo o processo tem como objetivo adquirir nuvens de pontos tridimensionais representando todos os elementos arbitrários contidos na cena. Cada extração de pontos representa um modelo referente a um único ponto de vista. Para obter-se de fato um modelo 3D completo de um objeto ou da cena, pode-se integrar os conjuntos de pontos capturados sucessivamente em diferentes perspectivas dos mesmos. As nuvens de pontos referentes a cada aquisição inicialmente não apresentam nenhum vínculo, sendo assim necessário algum método para encontrar suas relações como, por exemplo, *Iterative Closest Points* (ICP) (Seção 2.2).

1.2 Definição do Problema

A partir de diferentes conjuntos de pontos tridimensionais adquiridos através de um digitalizador câmera-projetor de alta definição, o problema deste trabalho é identificar suas correspondências geométricas. Ou seja, deve-se encontrar as transformações afins de rotação e translação que alinhem as possíveis partes comuns das variadas leituras, para uma reconstrução unificada e otimizada dos modelos obtidos.

1.3 Objetivos

Este trabalho tem como principal objetivo o estudo de métodos que solucionem a questão de minimização de diferenças de distâncias relativas entre nuvens de pontos que possuem correlacionamento geométrico. Além disso, alguns objetivos secundários podem ser descritos:

- Apresentar conceitos matemáticos necessários para a compreensão dos métodos citados;
- Mostrar métodos computacionais e possíveis variações ajustadas para o problema da aquisição via digitalizador câmera-projetor.

2 *Fundamentos*

2.1 Digitalizadores câmera-projetor

2.1.1 Calibração do sistema

A calibração da câmera tem como objetivo inferir seus parâmetros intrínsecos e extrínsecos tais como, distância focal, tamanho real de *pixel*, orientação e posição do dispositivo quanto ao sistema de coordenadas do mundo. Neste processo, descrito em (TSAI, 1992), os seguintes sistemas de coordenadas são utilizados:

- *Coordenadas de mundo* (SCM): coordenadas tridimensionais que caracterizam os pontos presentes na cena. Neste sistema cada ponto é denotado como $P_w = (\mathrm{x}_w, \mathrm{y}_w, \mathrm{z}_w)$.
- *Coordenadas de câmera* (SCC): sistema também tridimensional que possui como origem o centro ótico da câmera. Cada ponto é $p = (\mathrm{x}, \mathrm{y}, \mathrm{z})$ onde o plano *xy* é paralelo ao plano da imagem e z representa o eixo ótico da câmera.
- *Coordenadas de imagem* (SCI): este representa, diferentemente dos anteriores, um sistema bidimensional identificado no plano de projeção e tem como origem a projeção ortogonal do centro ótico da câmera no plano de projeção. Os pontos são descritos por $(\mathrm{X}_u, \mathrm{Y}_u)$ quando a câmera é considerada ideal, ou seja, não há distorções causadas pelas lentes e $(\mathrm{X}_d, \mathrm{Y}_d)$ quando houver distorções.
- Coordenadas de imagem do computador ou *coordenadas de pixel* (SCP): também bidimensional, corresponde diretamente à imagem armazenada na memória do computador. Tem como origem o centro real da imagem ou, na maioria dos casos, no canto superior esquerdo e seus pontos são representados por $(\mathrm{X}_f, \mathrm{Y}_f)$.

As relações entre os diferentes sistemas de coordenadas é apresentado na Figura 2.1. A Figura 2.2 sintetiza as quatro etapas propostas por (TSAI, 1992) para a calibração da câmera.

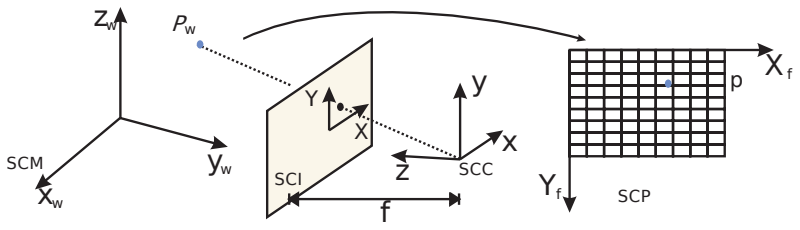


Figura 2.1: Sistemas de coordenadas (CODA et al., 2007).

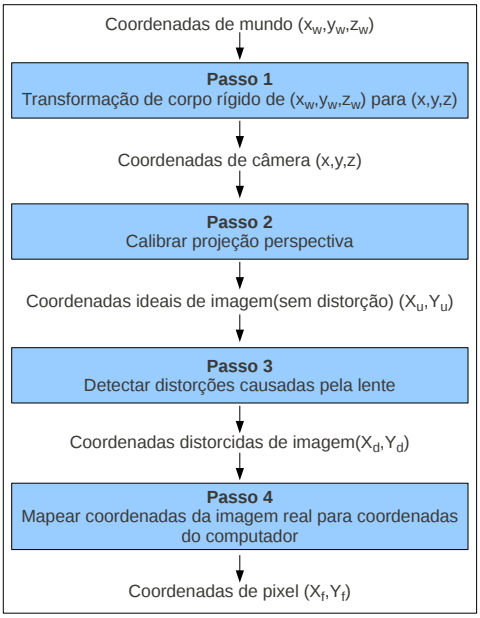
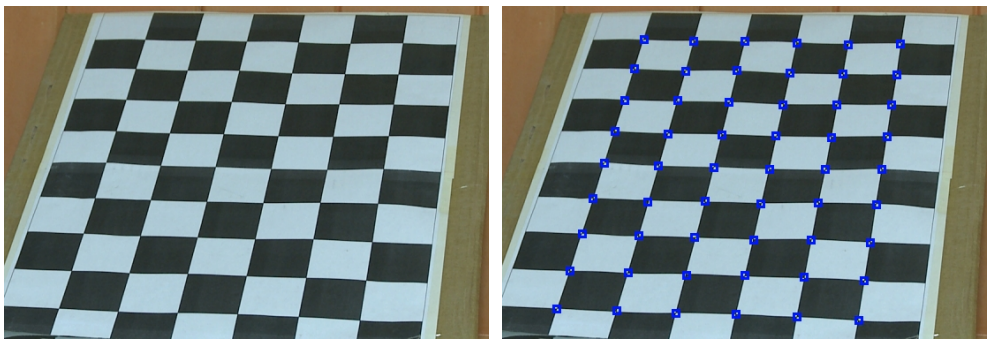


Figura 2.2: Resumo dos passos de calibração de câmera (TSAI, 1992).

Algumas informações são necessárias para que esse processo seja realizado. É importante conhecer de antemão alguns pontos da cena para uma análise inicial de correspondências e iniciar as resoluções dos sistemas de equações. Para isto, pode-se utilizar um plano com algum padrão pré-estabelecido como apresentado em (PARK; PARK, 2010) e (CODA et al., 2007). Este plano na maioria dos casos é um tabuleiro de xadrez (Fig. 2.3(a)). Por este padrão ser bem simples, pode-se marcar pontos característicos como referência, sendo suas dimensões reais previamente conhecidas (Fig. 2.3(b)).

No fim do processo, a transformação desejada para cada ponto é

$$p \simeq \mathbf{K}_{int_c}\mathbf{K}_{ext_c}P = \mathbf{K}_c P_w, \tag{2.1}$$



(a) (b)

Figura 2.3: Padrão xadrez (a) no plano e (b) com pontos característicos marcados.

com

$$\mathbf{K}_{ext_c} = \mathbf{R}_c\mathbf{T}_c = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{21} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} \quad e \quad \mathbf{K}_{int_c} = \begin{bmatrix} fs_x & f\gamma & C_x \\ 0 & fs_y & C_y \\ 0 & 0 & 1 \end{bmatrix},$$

onde $\mathbf{K}_{ext_c}$ representam os parâmetros extrínsecos de translação e rotação (obtidos no **Passo 1** da Figura 2.2) e $\mathbf{K}_{int_c}$ representam os parâmetros implícitos listados a seguir:

- f: distância focal;
- s_x e s_y: são os fatores de escala em x e y respectivamente. Descritos pelo número de *pixels* por unidade de comprimento;
- C_x e C_y: coordenadas da projeção ortogonal no centro ótico do plano de projeção;
- γ: representa a inclinação entre os dois eixos da imagem.

O projetor é visto como uma câmera inversa por projetar ao invés de capturar. Para sua calibração utiliza-se também a câmera já calibrada. A diferença é que nesta etapa projeta-se um padrão no plano (Fig. 2.4) mas da mesma forma marcam-se pontos de referência na imagem adquirida. Sua calibração segue os mesmos princípios descritos anteriormente. Esta analogia é possível porque as dimensões do padrão projetado são conhecidas em *pixels* e a câmera já é capaz de identificar as distorções presentes na imagem. Seja Q um ponto em coordenadas de mundo capturado pela câmera referente ao padrão projetado pelo projetor digital, a transformação final pretendida para cada ponto é

$$q \simeq \mathbf{K}_{int_p}\mathbf{K}_{ext_p}Q = \mathbf{K}_pQ_w. \tag{2.2}$$

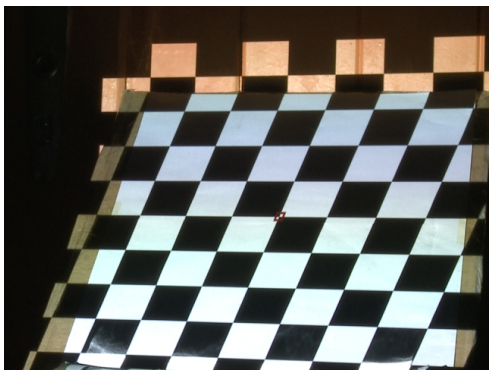


Figura 2.4: Padrão xadrez projetado.

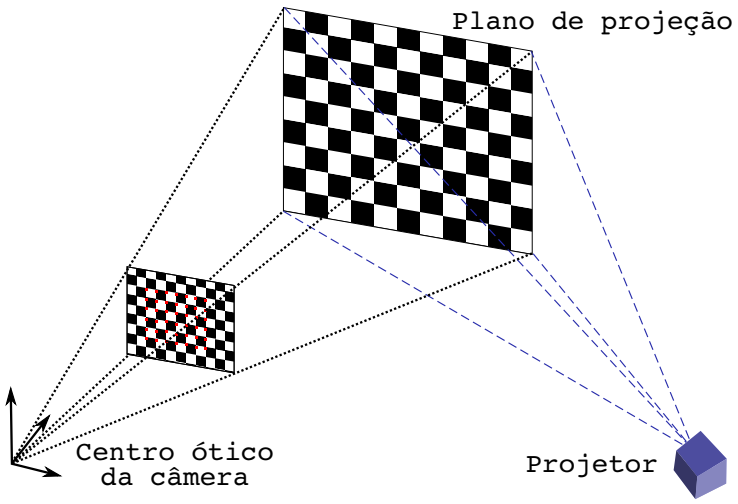


Figura 2.5: Captura da projeção.

A matriz $\mathbf{K}_p$ possui parâmetros intrínsecos e extrínsecos análogos aos da câmera porém referentes ao projetor. Com as Equações 2.1 e 2.2, pode-se calcular a transformação de corpo rígido entre a câmera e o projetor. Sendo $\mathbf{R}_c^{-1}$ a matriz inversa da rotação da câmera, a rotação relativa que alinha o sistema da câmera com o do projetor é

$$\mathbf{R}_{cp} = \mathbf{R}_p \mathbf{R}_c^{-1}.$$

E a translação relativa que leva a origem da câmera à origem do projetor é

$$\mathbf{T}_{cp} = \mathbf{T}_p - (\mathbf{R}_{cp}\mathbf{T}_c).$$

2.1.2 *Boundary Coded Structured Light* (BCSL)

A forma de processamento das imagens depende diretamente da descrição dos padrões de projeção. Há diferentes modelos e abordagens criados com fins específicos como tempo de processamento, precisão de captura e nível de detalhes.

Padrões de *Codificação de Luz Estruturada* (CSL) são projeções de *slides* contendo múltiplas faixas coloridas ou não. A organização das faixas deve ser bem estabelecida para que seja possível identificar suas posições no projetor. Desta forma, com uma câmera e um projetor previamente calibrados, pode-se fazer uma triangulação de forma a obter a profundidade de cada ponto reconhecido no processamento.

Uma forma de codificação utilizando as fronteiras entre faixas foi proposta por (HALL-HOLT; RUSINKIEWICZ, 2001). Nesta proposta são analisados os pares de faixas e não cada faixa em si. O padrão consiste na projeção de faixas pretas e brancas, amenizando o problema da variação de reflexão dos objetos contidos na cena. Mas esta abordagem possibilita transições de preto para preto e branco para branco, resultando no surgimento de fronteiras "fantasma".

A codificação utilizada no presente trabalho é denominada *Codificação de Luz Estruturada de Fronteira* ou *(b,s)-BCSL* (SA; CARVALHO; VELHO, 2002). Esta proposta utiliza a projeção de s *slides* de faixas com b cores. Foi escolhida a projeção de faixas verticais. Seu objetivo é adquirir uma resolução satisfatória sem que a complexidade do reconhecimento das transições entre faixas seja grande.

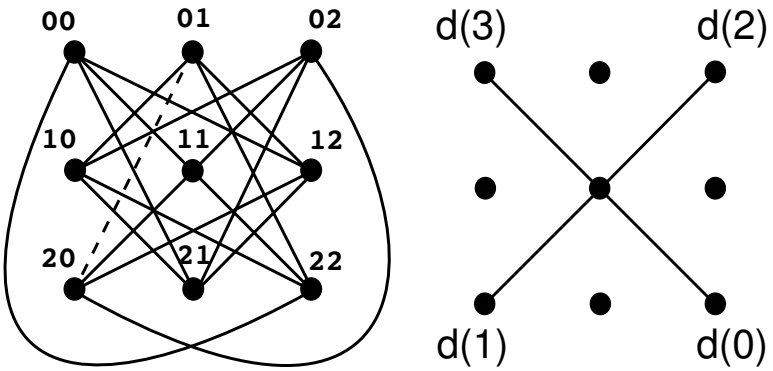
Após cada *slide* descrito no padrão, projeta-se um novo utilizando as cores complementares do anterior em suas respectivas posições. Ou seja, cada *slide* S_i possui um complementar $\overline{S}_i$. Desta forma, eliminam-se algumas restrições quanto a refletividade dos objetos em cena. A codificação *(b,s)-BCSL* possibilita $[\mathrm{b}(\mathrm{b}-1)^s]$ diferentes padrões. Ao utilizar um maior número de cores e *slides*, pode-se gerar mais faixas num mesmo padrão. Deste modo, com mais transições, torna-se possível aumentar a resolução da geometria adquirida.

As cores definidas para utilização são *vermelho* (R), *verde* (G) e *azul* (B) e suas respectivas complementares *ciano* (C), *magenta* (M) e *amarelo* (Y), totalizando seis cores. A utilização destas cores possibilita uma analogia à codificação binária pois no sistema RGB os valores de cada canal são definidos como 0 ou 1. A cor branca não é utilizada por questões de simetria e nem a preta para não ser confundida com sombras presentes na cena.

A escolha do parâmetro s está diretamente ligada a impor restrições quanto a movimento e tempo. Quando $\mathrm{s} = 1$, a codificação é apenas espacial, ou seja, não impõe restrições quanto a movimentação de objetos na cena. Se $\mathrm{s} > 1$ então a codificação torna-se espacial e temporal, pois movimentos dos objetos na cena podem dificultar ou até inviabilizar suas detecções. A forma mais simples de se utilizar o quesito temporal é definir $\mathrm{s} = 2$. Desta forma, pode-se detectar objetos que se movam lentamente de forma que seu deslocamento entre as duas capturas seja menor que a largura de cada faixa.

A detecção das faixas não deve conter ambiguidades, ou seja, as transições presentes em cada par de *slides* nas mesmas posições não devem ser repetidas. Assim, a sequência de padrões deve ser escolhida de forma a possibilitar com eficiência e rapidez a identificação das posições das fronteiras detectadas no projetor. O problema de geração das sequências de faixas pode ser representado como descobrir um caminho Euleriano em um grafo G. Define-se então que G possui b^s vértices onde cada um corresponde a uma possível transição para cada um dos s *slides* escolhidos.

Como exemplo, pode-se definir $b = 3$ e $s = 2$ como parâmetros (SA; CARVALHO; VELHO, 2002). A Figura 2.6(a) mostra o grafo G com seus nove vértices. Todo vértice V é classificado por dois dígitos, cada um representando um número de cor em cada slide. O vértice $V(20)$, por exemplo, indica a projeção da cor 2 no primeiro *slide* e da cor 0 no segundo. As arestas representam as possíveis transições para duas posições de faixas consecutivas. Pode-se notar que não há transições para vértices onde uma das cores é igual e no mesmo *slide* (como de $V11$ para $V12$). Cada vértice possui $(b-1)^s$ vizinhos. A Figura 2.6(b) mostra um exemplo



(a) (b)

Figura 2.6: (a) Grafo do (3,2)-BCSL. (b) Vizinhança do vértice $V(11)$ (SA; CARVALHO; VELHO, 2002).

com a vizinhança do vértice $V(11)$. Vale ressaltar que as arestas são bidirecionais, ou seja, cada uma representa duas transições distintas, sendo uma para cada direção. Este grafo serve de base para a codificação dos padrões. Uma possível codificação para os parâmetros $(3,2)$ utilizando R, G e B como base é apresentada na Figura 2.7. Nela está em destaque a fronteira representada pela transição de $V(20)$ para $V(01)$.

A decodificação consiste em recuperar as posições das bordas para as transições de cada *slide* dado. Este problema pode ser definido como encontrar a posição da aresta desejada no caminho gerado durante a codificação. Uma decodificação para os parâmetros $(3,2)$ é mostrada na Tabela 2.1. Cada linha desta tabela representa um vértice $V_i \in G$ na base b e cada coluna os

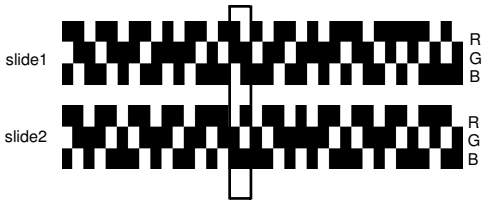


Figura 2.7: Codificação BCSL (SA; CARVALHO; VELHO, 2002).

Vértices	d(0)	d(1)	d(2)	d(3)
$V(00)$	0	3	6	9
$V(01)$	14	17	19	11
$V(02)$	28	34	22	24
$V(10)$	26	29	18	21
$V(11)$	1	31	33	**35**
$V(12)$	15	4	8	13
$V(20)$	16	23	32	12
$V(21)$	27	5	7	25
$V(22)$	2	10	20	30

Tabela 2.1: Tabela de decodificação do (3,2)-BCSL (SA; CARVALHO; VELHO, 2002).

vizinhos desse vértice.

Os valores presentes na Tabela 2.1 correspondem às posições, no caminho Euleriano, do arco que representa a transição de cada vértice V para todos os seus vizinhos. Como exemplo pode-se analisar o arco que vai do vértice $V(11)$ para $V(00)$. Logo, deve-se olhar o valor presente na linha $V(11)$ e coluna d(3), que representa o $V(00)$ como vizinho (Fig. 2.6(b)). Assim pode-se concluir que este é o 35º arco do caminho e também a 35ª fronteira do padrão projetado. Então, para achar o número da fronteira que representa a transição detectada de um vértice V_i para um V_j, procura-se a respectiva entrada na tabela de decodificação.

Por fim, obtêm-se as fronteiras detectadas na imagem capturada e estas estão identificadas no padrão de projeção. Então, com as matrizes de calibração da câmera e do projetor calculadas e os pontos identificados, é possível fazer a triangulação necessária para obter as profundidades de cada ponto adquirido.

2.2 Algoritmo Iterativo de Pontos Mais Próximos

O *ICP* (BESL; MCKAY, 1992) é um algoritmo que tem como objetivo estimar transformações que minimizem distâncias entre um modelo e um dado a ser ajustado. O conceito

utilizado consiste em encontrar, a cada iteração, uma rotação e uma translação que de forma global alinhem ou façam um registro entre o modelo e os dados. Este método pode ser utilizado para diferentes formas geométricas, tais como conjuntos de pontos, de linhas, de triângulos, curvas paramétricas e superfícies paramétricas. O desenvolvimento da aplicação descrita nesta monografia utiliza nuvens de pontos.

O ICP tem como dados de entrada duas nuvens de pontos $\mathbf{M}$ e $\mathbf{P}$, com N_m e N_p elementos respectivamente. O objetivo é estimar transformações ótimas a fim de alinhar o conjunto $\mathbf{P}$ de dados a serem ajustados ao conjunto modelo $\mathbf{M}$, ambos com pontos definidos em $\mathbb{R}^3$. Deve-se primeiramente determinar pontos correspondentes entre os dois conjuntos. Para tal, utiliza-se a distância euclideana. Para cada ponto $\vec{p} \in \mathbf{P}$, define-se como o ponto correspondente o ponto $\vec{m} \in \mathbf{M}$ de menor distância euclideana. Seja a função de distância

$$\mathrm{d}(\vec{p}, \mathbf{M}) = \min_{\vec{m} \in \mathbf{M}} ||\vec{m} - \vec{p}||,$$

denota-se por $\mathbf{Y}$ o conjunto de pontos resultantes. Denomina-se C como operador de ponto mais próximo, sendo

$$\mathbf{Y} = C(\mathbf{P}, \mathbf{M}).$$

Deve-se observar que cada ponto $\vec{p}_i \in \mathbf{P}$ possui um correspondente $\vec{r}_i \in \mathbf{Y}$. Desta forma, ambos os conjuntos têm o mesmo número N_p de elementos.

Para o próximo passo, utiliza-se o conceito de *quaternions*. Um quaternion é um vetor em $\mathbb{R}^4$ que representa um arco em $\mathbb{R}^3$. A unidade quarternion é dada pela quádrupla

$$\vec{q}_r = [q_0 \; q_1 \; q_2 \; q_3]^t, \quad q_0 \geq 0 \quad \text{e} \quad q_0^2 + q_1^2 + q_2^2 + q_3^2 = 1. \tag{2.3}$$

Com a Equação 2.3 e considerando um vetor $\vec{q}_t = [q_4 \; q_5 \; q_6]^t$ de translação, denomina-se como vetor completo de registro de estado $\vec{q} = [\vec{q}_r | \vec{q}_t]$. Para obter este vetor final, deve-se realizar a operação quaternion de mínimos quadrados (Q) descrita como

$$(\vec{q}, \mathrm{d}_{ms}) = Q(\mathbf{P}, \mathbf{Y}),$$

onde d_{ms} descreve a média do erro quadrático de correspondência dos pontos. Este valor tem a forma

$$\mathrm{d}_{ms} = \frac{1}{\mathrm{N}_p} \sum_{i=0}^{\mathrm{N}_p} ||\vec{r}_{i,k} - \vec{q}_{i,k+1}||^2. \tag{2.4}$$

A variável k representa o número da iteração. Ou seja, na resolução desta média quadrada a distância é calculada entre os pontos do conjunto **Y** e seus correspondentes já transformados no conjunto **P**.

Para encontrar o operador Q, primeiro definem-se os centroides $\vec{\mu}_p$ de **P** e $\vec{\mu}_y$ de **Y**.

$$\vec{\mu}_p = \frac{1}{\mathrm{N}_p} \sum_{i=0}^{\mathrm{N}_p} \vec{p}_i \quad \text{e} \quad \vec{\mu}_y = \frac{1}{\mathrm{N}} \sum_{i=0}^{\mathrm{N}_p} \vec{r}_i. \tag{2.5}$$

Com os centroides definidos, calcula-se uma matriz de covariância cruzada de **P** e **Y**

$$\Sigma_{py} = \frac{1}{\mathrm{N}_p} \sum_{i=0}^{\mathrm{N}_p} (\vec{p}_i \vec{r}_i^{\,t}) - (\vec{\mu}_p \vec{\mu}_y^{\,t}). \tag{2.6}$$

Utilizando a Equação 2.6, define-se uma matriz $\mathbf{A} = \Sigma_{py} - \Sigma_{py}^t$. Os componentes cíclicos desta matriz são usados para compor um vetor $\Delta = [A_{23}\, A_{31}\, A_{12}]^t$. Outra informação necessária é o valor do traço da matriz de covariância (Eq. 2.6)

$$tr(\Sigma_{py}) = \Sigma_{11} + \Sigma_{22} + \Sigma_{33}. \tag{2.7}$$

Utilizando as equações 2.5, 2.6 e 2.7 é possível montar a matriz $Q(\Sigma_{py})$ que dará origem ao quaternion.

$$Q(\Sigma_{py}) = \begin{bmatrix} tr(\Sigma_{py}) & \Delta^t \\ \Delta & \Sigma_{py} + \Sigma_{py}^t - tr(\Sigma_{py})\mathbf{I}_3 \end{bmatrix},$$

onde $\mathbf{I}_3$ é a matriz identidade 3×3. O quaternion $\vec{q}_r$ é descrito como o autovetor associado ao maior autovalor de $Q(\Sigma_{py})$, sendo também considerado como rotação ótima. Utilizando o quaternion $\vec{q}_r$, constrói-se a matriz de rotação

$$\mathbf{R}(\vec{q}_r) = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 - q_0q_3) & 2(q_1q_3 + q_0q_2) \\ 2(q_1q_2 + q_0q_3) & q_0^2 + q_2^2 - q_1^2 - q_3^2 & 2(q_2q_3 - q_0q_1) \\ 2(q_1q_3 - q_0q_2) & 2(q_2q_3 + q_0q_1) & q_0^2 + q_3^2 - q_1^2 - q_2^2 \end{bmatrix}.$$

Utilizando os centroides (Eq. 2.5) e a matriz $\mathbf{R}(\vec{q}_r)$, é possível calcular o vetor de translação

$$\vec{q}_t = \vec{\mu}_y - \mathbf{R}(\vec{q}_r)\vec{\mu}_p.$$

Por fim, pode-se montar a matriz final de transformação 4×4

$$\mathbf{T} = \begin{bmatrix} R(\vec{q}_r)_{11} & R(\vec{q}_r)_{12} & R(\vec{q}_r)_{13} & \vec{q}_{t_1} \\ R(\vec{q}_r)_{21} & R(\vec{q}_r)_{22} & R(\vec{q}_r)_{23} & \vec{q}_{t_2} \\ R(\vec{q}_r)_{31} & R(\vec{q}_r)_{32} & R(\vec{q}_r)_{33} & \vec{q}_{t_3} \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

que deverá ser aplicada a todos os pontos do conjunto **P**. Como os pontos são tridimensionais, para realizar a multiplicação pela matriz basta utilizar as coordenadas homogêneas dos mesmos. Ou seja, utiliza-se os pontos $\tilde{p}_i = (\mathrm{x}_i, \mathrm{y}_i, \mathrm{z}_i, 1)$ e normaliza-se pela quarta coordenada após transformado.

Todo esse processo pode ser repetido até que uma condição seja satisfeita como, por exemplo, um determinado número fixo de iterações seja atingido ou uma medida de erro ultrapassar algum limiar pré-estabelecido. É utilizado como padrão determinar um limiar τ de precisão baseada na diferença de erros (Eq. 2.4) entre iterações. Ou seja, repete-se o algoritmo até que a seguinte condição seja satisfeita:

$$\mathrm{d}_k - \mathrm{d}_{k+1} < \tau. \tag{2.8}$$

3 *Modelo de Classe para ICP*

Neste capítulo serão descritas as implementações realizadas, assim como suas modificações e melhoramentos.

3.1 Classe *gcgICP*

A implementação do algoritmo ICP (Seção 2.2) utilizou-se dos recursos de orientação a objetos da linguagem *C++*. O diagrama de classes descrevendo a estrutura adotada pode ser visto na Figura 3.1.

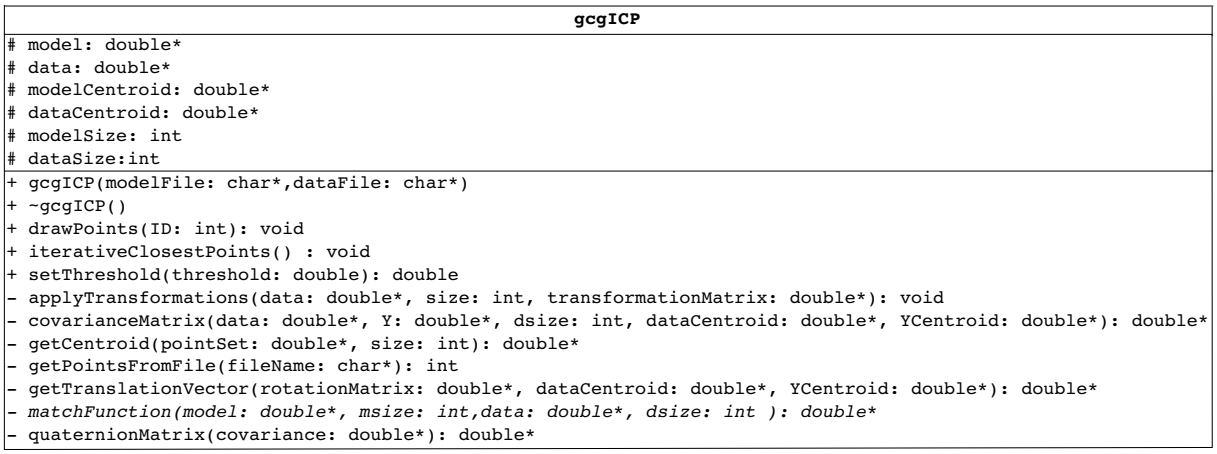


Figura 3.1: Diagrama da classe gcgICP.

Foi utilizada uma biblioteca chamada *gcgLib* que contém operações sobre vetores e matrizes, como multiplicações e somas. Esta biblioteca também possui métodos numéricos para aquisição de autovetores e autovalores de matrizes e gera matrizes de rotação a partir de quaternions. Também foi utilizada uma outra biblioteca chamada *OpenGL*, de código livre. Com ela foi possível exibir os resultados em ambiente gráfico.

Quando criada uma nova instância de um objeto da classe gcgICP, são passados como parâmetros a localização de dois arquivos. Estes arquivos contêm apenas pontos tridimensionais representando os dois conjuntos de pontos (modelo **M** e dados **P**) utilizados no algoritmo. A

função *getPointsFromFile* realiza esta leitura individualmente e retorna o respectivo número de pontos contidos no arquivo. Cada nuvem de pontos é um vetor de tamanho 3N, onde N é o total de pontos da mesma.

A função *iterativeClosestPoints* organiza e executa sequencialmente todas as funções relativas aos passos da resolução do algoritmo ICP (Seção 2.2). Ela executa todas as iterações necessárias até que o critério de parada (Eq. 2.8) seja atingido, retornando o tempo total de processamento. Deve-se assim determinar o limiar τ de erro para avaliação de parada utilizando a função *setThreshold*.Também é possível desabilitar as iterações sucessivas. Deste modo, elas podem ser executadas uma a uma para fins de análise caso seja de interesse. A seguir serão detalhados os procedimentos executados.

Como descrito na definição do ICP, primeiramente deve-se definir uma correspondência inicial entre os pontos do conjunto de dados e do modelo. A função *matchPoints* realiza esta tarefa, retornando um novo conjunto de pontos $\mathbf{Y}$ como resultado. Vale lembrar que os conjuntos $\mathbf{P}$ e $\mathbf{Y}$ possuem um mesmo número N_p de elementos e os pontos correspondentes estão na mesma posição em cada vetor.

A métrica utilizada para a identificação de correspondências pode variar de acordo com o interesse de aplicação do algoritmo. Por exemplo, no caso do algoritmo original, é definida como a distância euclideana entre pontos dos dois conjuntos. Mas, pode-se modificar esta métrica utilizando diferentes informações como, por exemplo, diferenças de texturas ou distâncias relativas a alguma referência centrada. Para permitir esta flexibilidade de implementações, a função *matchPoints* foi definida como *virtual*, sendo necessária sua implementação para cada subclasse concreta. Isto permite que seja fácil alterar os critérios de estabelecimento de correspondência de pontos de acordo com o problema sem precisar modificar a estrutura de toda a classe, criando assim variações do algoritmo ICP.

Com as correspondências definidas utiliza-se a função *getCentroid* para calcular, individualmente, os centroides de $\mathbf{P}$ e $\mathbf{Y}$. Utilizando estes dados é possível calcular a matriz de covariância cruzada Σ_{py} com a função *covarianceMatrix*.

Após obter a matriz Σ_{py}, a função *quaternionMatrix* é responsável por retornar a matriz 4×4 que dará origem ao quaternion. Como esta matriz é simétrica, utiliza-se uma função presente na *gcgLib* chamada *gcgEigenSymmetric* para extrair seus autovalores e autovetores. O retorno são dois vetores, sendo que o primeiro contém os autovalores ordenados crescentemente e o segundo com os autovetores associados em suas respectivas posições.

Como já descrito, o autovetor associado ao maior autovalor é escolhido e então utilizado

na macro *ROTATIONFROMQUATERNION* para conseguir a matriz de rotação global. Pode-se neste momento calcular o vetor de translação com a função *getTranslationVector* e então compor a transformação final. Esta será aplicada a todos os pontos do conjunto **P** através da função *applyTransformations*.

Caso o algoritmo esteja definido de forma a iterar até convergir, calcula-se o novo erro de correspondência com o método *getError*. Desta forma, avalia-se se a diferença deste erro com o anterior para determinar se há a necessidade de uma próxima iteração.

Utilizam-se comandos de *OpenGL* para exibir os pontos de um dos conjuntos. A função *drawPoints* realiza esta operação, tendo como parâmetro o conjunto a ser exibido.

3.2 Adaptação do Algoritmo para o padrão BCSL

Propõe-se, neste trabalho, uma modificação do algoritimo original do ICP. O objetivo é alinhar, com maior precisão, nuvens de pontos resultantes de sucessivas capturas utilizando o sistema câmera-projetor com o padrão BCSL. Uma abordagem semelhante foi publicada em (RUSINKIEWICZ; HALL-HOLT; LEVOY, 2002), utilizando um sistema de captura com vídeo em tempo real e padrão de faixas em preto e branco.

Sabe-se que todos os pontos identificados localizam-se em uma fronteira entre faixas. O algoritmo original do ICP, em sua função de correspondência (implementada como *matchFunction*), para cada $\vec{p} \in \mathbf{P}$ realiza uma busca em todo o conjunto **M**. É proposto que, para determinar a correspondência de $\vec{p} \in \mathbf{P}$, apenas os pontos de **M** presentes na fronteira de mesmo número que $\vec{p}$ sejam levados em consideração (Fig. 3.2). Desta forma, pretende-se obter uma maior precisão na estimação das correspondências resultando em transformações mais adequadas a serem aplicadas. Além disto, ruídos provenientes da aquisição de pontos, em sua maioria, serão descartados. Isto se deve ao fato destes ruídos representarem pontos discrepantes e muito afastados, ou seja, provavelmente não encontrarão correspondentes.

Para que esta abordagem seja possível, algumas modificações devem ser feitas na função de correspondências:

- Incluir, para cada ponto, o número da fronteira nos arquivos de entrada. Logo, o vetor que representa cada conjunto de pontos deve também ser expandido para armazenar a nova informação;
- O candidato a ponto correspondente somente deve ser avaliado caso esteja na mesma

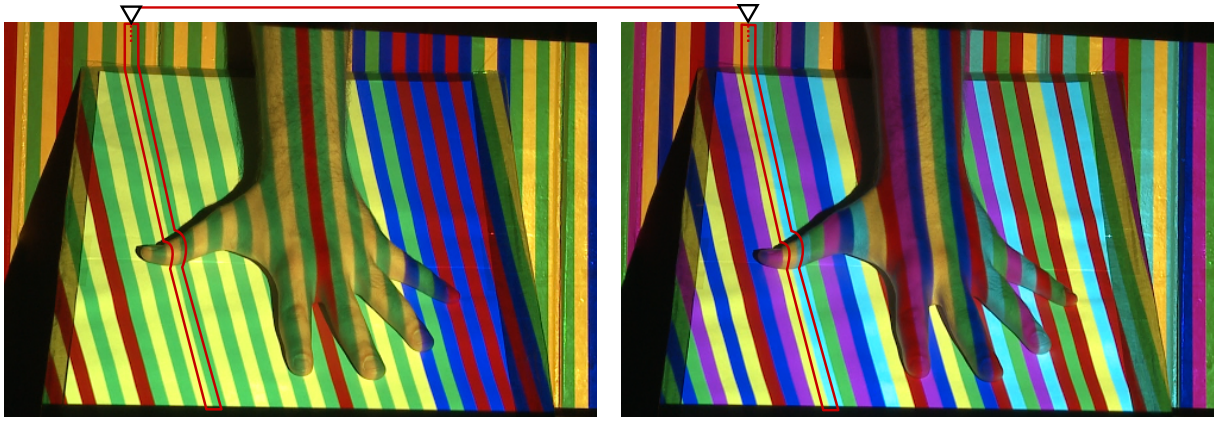


Figura 3.2: Busca de pontos apenas na mesma fronteira.

fronteira que o ponto que busca correspondência. Para isto, adiciona-se uma condicional antes do cálculo da distância euclideana;

- Caso um ponto não possua nenhum correspondente, este deve ser invalidado no cálculo das transformações. Ou seja, ele não será utilizado nos próximos passos do algoritmo e o número total N_p de pontos do conjunto **P** a ser avaliado será reduzido em 1 unidade (para cada ponto inválido). Desta forma, preserva-se a propriedade que estabelece que o número de pontos no conjunto resultante **Y** seja igual ao de pontos válidos de **P**.

Apesar desta modificação ser válida, ela impõe uma restrição quanto ao movimento dos objetos em cena. Isto porque, para objetos estáticos, os reais pontos correspondentes identificados nas diferentes capturas estarão presentes na mesma fronteira. Mas, caso os objetos tenham um deslocamento maior que a largura das faixas, os verdadeiros pontos correspondentes nunca serão identificados.

Para resolver esse problema o espaço de busca foi ampliado. Determina-se um fator d de deslocamento, ou seja, um intervalo de fronteiras à esquerda e à direita a serem também considerados válidos na busca de correspondência (Fig. 3.3). A Figura 3.2 demonstra um caso particular para $d = 0$. Como exemplo, pode-se determinar $d = 2$. Desta forma, para um ponto $\vec{p} \in \mathbf{P}$ presente na fronteira de número 20, os pontos de **M** nas fronteiras 18, 19, 21 e 22 também são avaliados, e não apenas os da fronteira 20. Deve-se lembrar de não tentar avaliar fronteiras não existentes, ou seja, quando percorrer o domínio de busca imposto por d, caso a fronteira em questão seja maior ou igual que o número total de fronteiras, ou menor que zero.

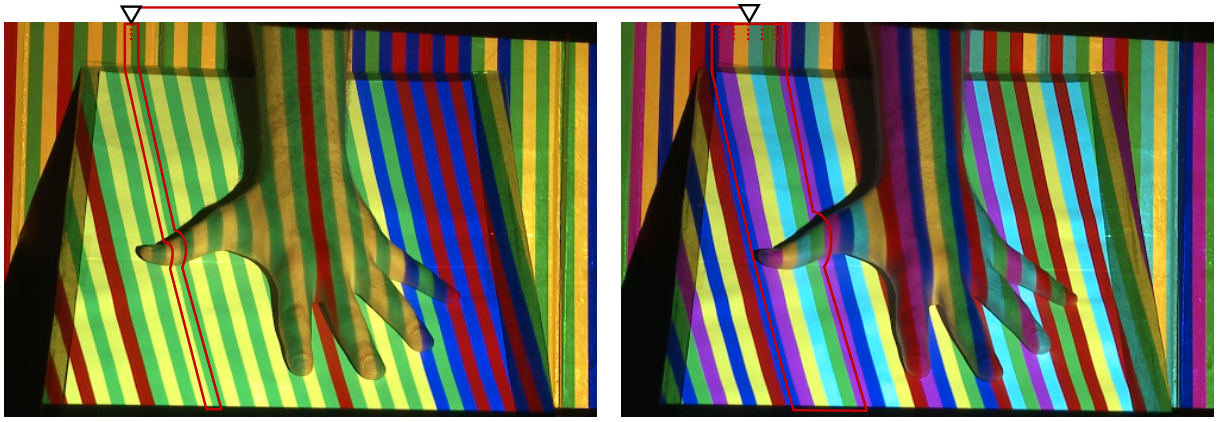


Figura 3.3: Intervalo de busca de pontos para $d = 2$.

3.3 Melhorias de Desempenho

A adaptação feita no algoritmo espera uma melhor precisão de alinhamento e consequentemente sugere uma redução do espaço de busca. Apesar de modificar o conceito de pontos válidos no cálculo de correspondências, continua-se percorrendo todo o conjunto modelo verificando se o ponto está no intervalo de fronteiras aceitável. Para reduzir o espaço de busca também no processamento, foram realizadas as seguintes modificações:

- Criar uma estrutura chamada *pointStrip* que contenha um vetor representando um ponto tridimensional, o número da fronteira do mesmo e um ponteiro para outra variável *pointStrip*;
- Não mais armazenar os conjuntos de pontos em vetores extensos. Foram criados então vetores que têm como tamanho o número total n de fronteiras. Cada posição deste vetor representa um ponteiro para *pointStrip*. Desta forma foram criadas n listas encadeadas dessa estrutura.

Assim, na leitura dos arquivos de entrada e construção dos pontos, cada ponto é adicionado na posição da tabela referente ao número de sua fronteira. Também para fins de desempenho as listas são inversas, ou seja, cada novo ponto é adicionado no início da lista. Deste modo não é necessário percorrer toda a lista a cada ponto adicionado. Com esta nova organização, pode-se diretamente realizar a busca de correspondência apenas nas fronteiras de interesse.

Uma outra modificação também foi implementada. Esta consiste em adicionar mais uma variável na estrutura *pointStrip*. Esta variável representa um outro ponteiro para *pointStrip* e armazena o ponto correspondente, caso exista, no conjunto modelo. Desta forma, para cada ponto é possível acessar diretamente seu correspondente, tornando desnecessário criar o novo

conjunto **Y** de correspondência. Deste modo é possível economizar memória e também processamento, evitando muitas operações de cópia.

4 Experimentos

Neste capítulo serão apresentados os resultados obtidos em três diferentes experimentos. Para todos os testes foi utilizado um computador com processador Intel Core 2 Quad Q8300 (2500MHz), 4GB de memória RAM DDR2 e placa de vídeo NVIDIA GeForce 9800 GTX de 512MB.

Em todas as exibições, os pontos azuis representam o conjunto modelo e os pontos verdes os dados a serem ajustados. Vale ressaltar que o fundo e demais áreas iluminadas pela projeção também foram detectadas. Mas, para fins de testes, apenas os pontos do objeto de interesse foram utilizados, sendo os demais pontos considerados ruídos. As duas nuvens de pontos estão posicionadas em relação ao centroide da nuvem modelo, onde estão desenhadas três linhas representando eixos. As linhas vermelha, verde e azul denotam os eixos x, y e z respectivamente.

Foi estabelecido como limiar de erro $\tau = 0.01$. Ou seja, quando a diferença dos erros de duas iterações consecutivas for menor que este limiar, o algoritmo é encerrado.

4.1 Experimento 1: Cachorro

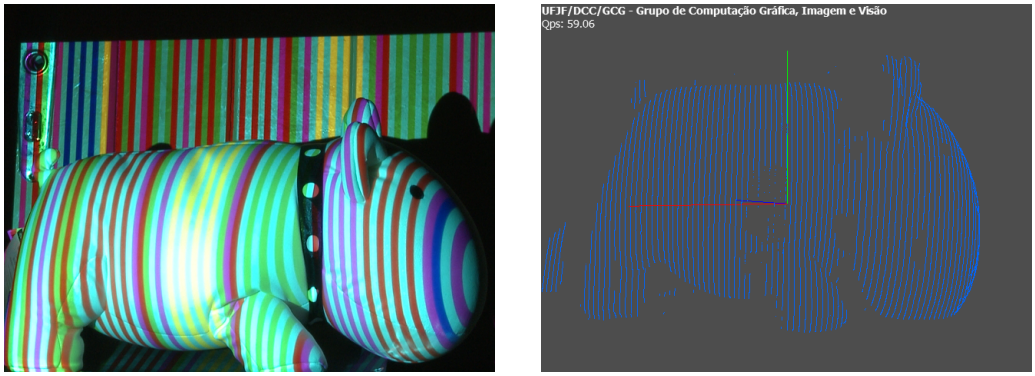
Para estes testes foi utilizada uma almofada em forma de cachorro. Sua superfície é branca e possui detalhes pretos como olhos e coleira.

O objeto foi capturado lateralmente como modelo. Para formar o conjunto de dados, o objeto foi rotacionado. As Figuras 4.1 e 4.2 demonstram, respectivamente, as digitalizações do modelo e dos dados.

O sistema antes da aplicação do algoritmo é apresentado na Figura 4.3.

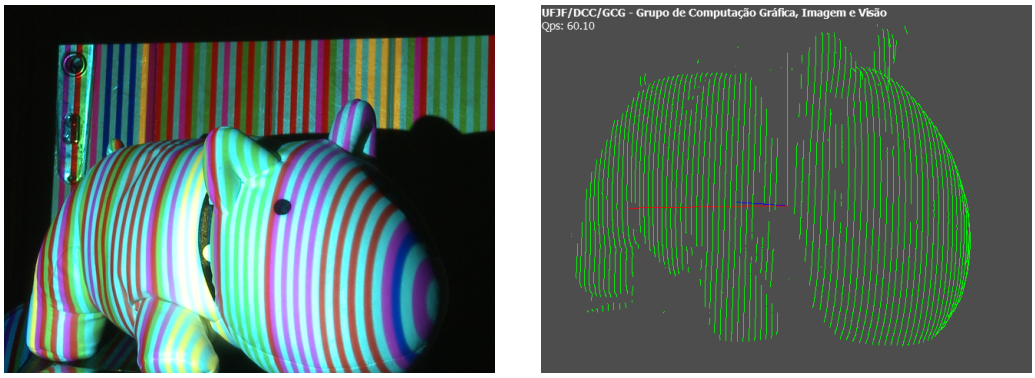
Os resultados são listados na Tabela 4.1 e podem ser visualizados na Figura 4.4.

Neste primeiro experimento pôde-se perceber que a adaptação ao padrão BCSL apresenta resultados satisfatórios. Um outro aspecto a ser analisado indica que o algoritmo adaptado, após as melhorias, apresenta um resultado muito similar ao algoritmo original, porém com menos



(a) (b)

Figura 4.1: Cachorro modelo (a) capturado e (b) digitalizado.



(a) (b)

Figura 4.2: Cachorro a ser ajustado (a) capturado e (b) digitalizado.

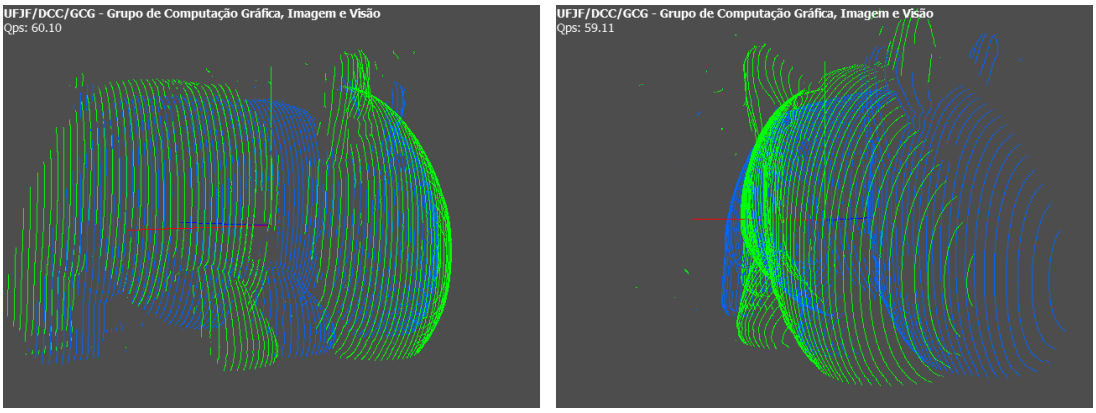
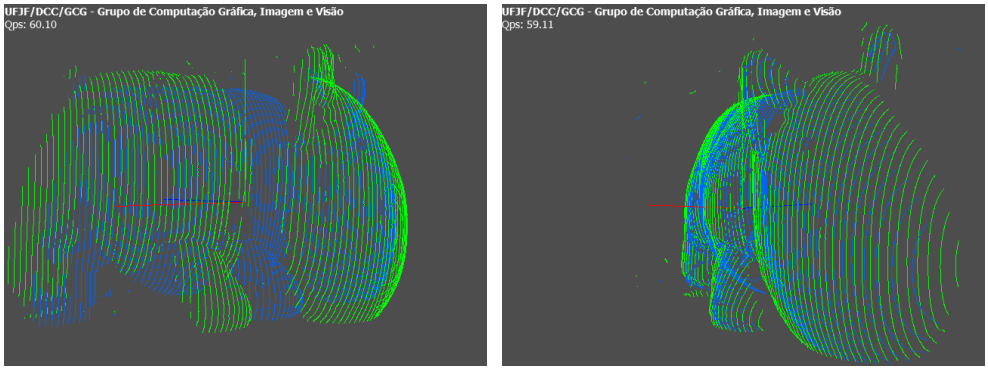
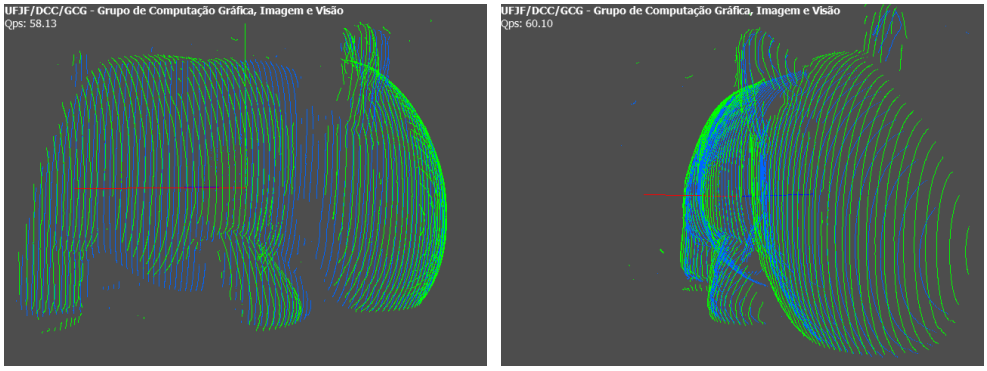


Figura 4.3: Experimento 1 antes das transformações.

iterações e quase dez vezes mais rápido.



(a)



(b)

Figura 4.4: Resultados do Experimento 1 utilizando (a) ICP original e (b) adaptação BCSL.

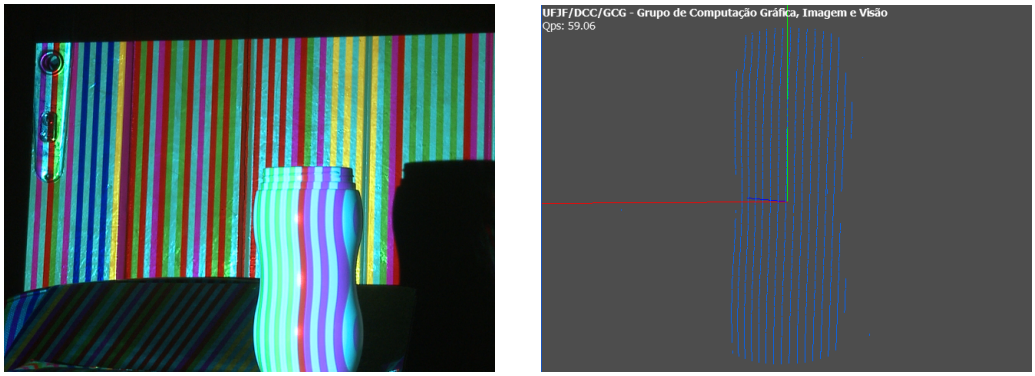
Cachorro	**Total de Pontos**		
Modelo	27767		
Dados	26541		
	Iterações	**Tempo (s)**	**Diferença de Erros**
ICP original	16	221,719	0,009880800086
ICP original com melhorias	16	189,944	0,009880800086
Adaptação BCSL	9	91,948	0,007256108463
Adaptação BCSL com melhorias	9	22,625	0,007256108463

Tabela 4.1: Resultados do Experimento 1.

4.2 Experimento 2: Pote

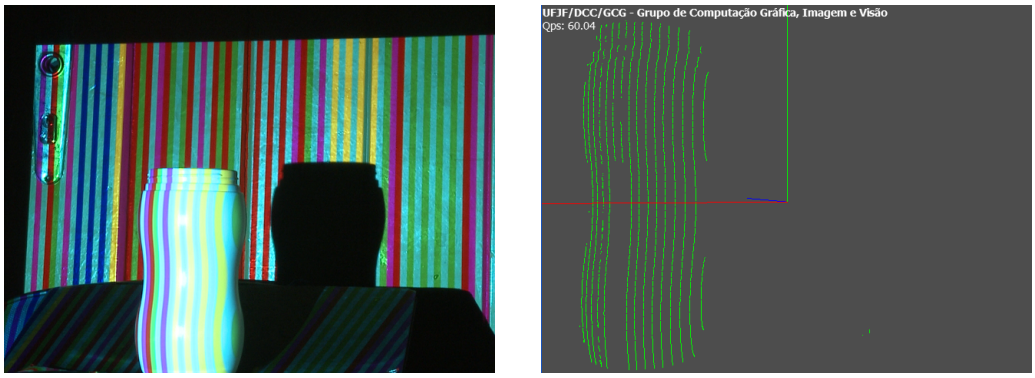
O segundo experimento foi realizado utilizando um pote branco. Foi realizada apenas uma translação do modelo para aplicar o algoritmo. A disposição inicial dos objetos é apresentada na Figura 4.7 e as digitalizações são exibidas nas Figuras 4.5 e 4.6.

Percebe-se que a intensidade da translação não deixou nenhuma área comum entre as duas



(a) (b)

Figura 4.5: Pote modelo (a) capturado e (b) digitalizado.



(a) (b)

Figura 4.6: Pote a ser ajustado (a) capturado e (b) digitalizado.

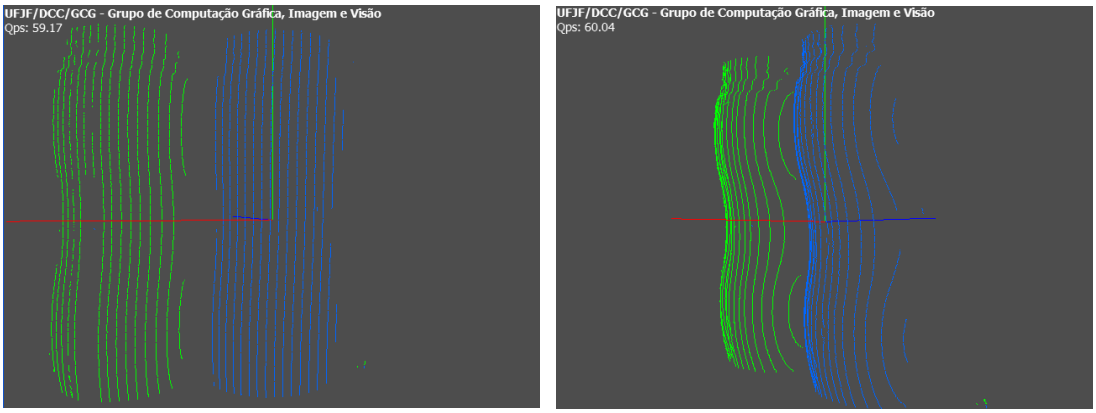


Figura 4.7: Experimento 2 antes das transformações.

digitalizações. Desta forma, sem interseções, pode-se deduzir que o algoritmo adaptado não funcionaria, visto que nenhum ponto encontraria correspondência. Para este caso em particular, foi inserido um deslocamento inicial no algoritmo, relacionando a primeira faixa incidente sobre

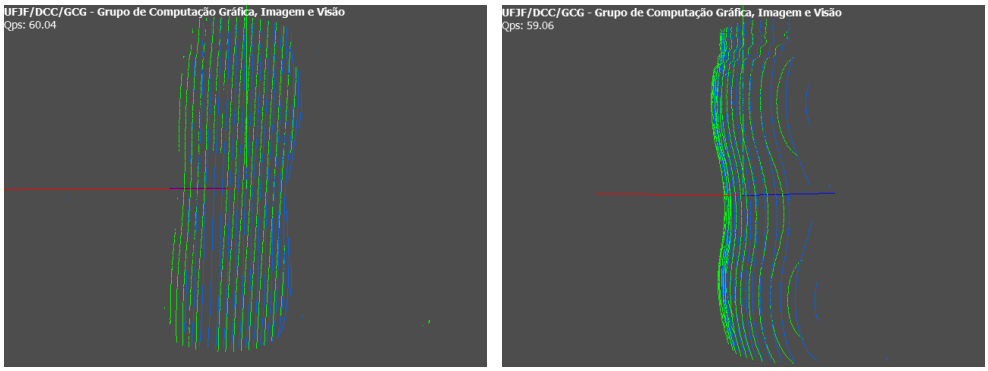
o modelo com a primeira incidente nos dados.

A Tabela 4.2 apresenta os resultados e estes podem ser visualizados na Figura 4.8.

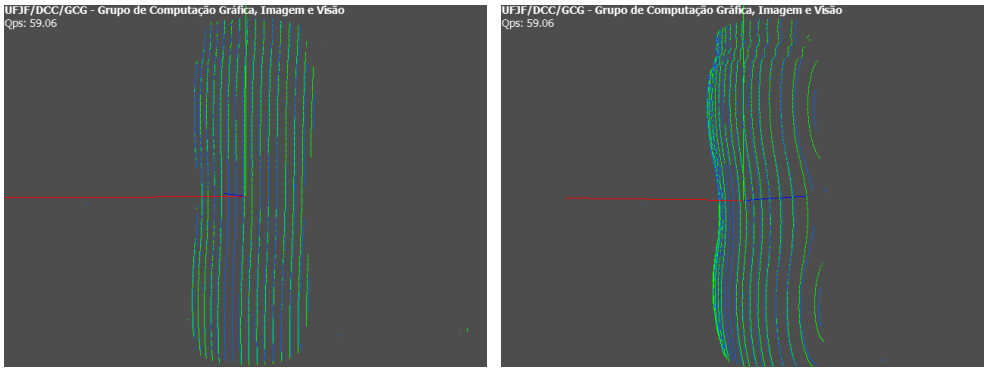
Pote	**Total de Pontos**		
Modelo	6009		
Dados	5357		
	Iterações	**Tempo (s)**	**Diferença de Erros**
ICP original	12	7,437	0,006493794667
ICP original com melhorias	12	5,802	0,006493794667
Adaptação BCSL	3	2,484	0,000470649482
Adaptação BCSL com melhorias	3	0,25	0,000470649482

Tabela 4.2: Resultados do Experimento 2.

Neste experimento em particular, apenas o ICP original foi utilizado sem ajustes. Mas, pode-se perceber que ao inserir um deslocamento inicial, o algoritmo adaptado apresenta uma boa precisão, resultando em um melhor alinhamento dos pontos reais em comum.



(a)

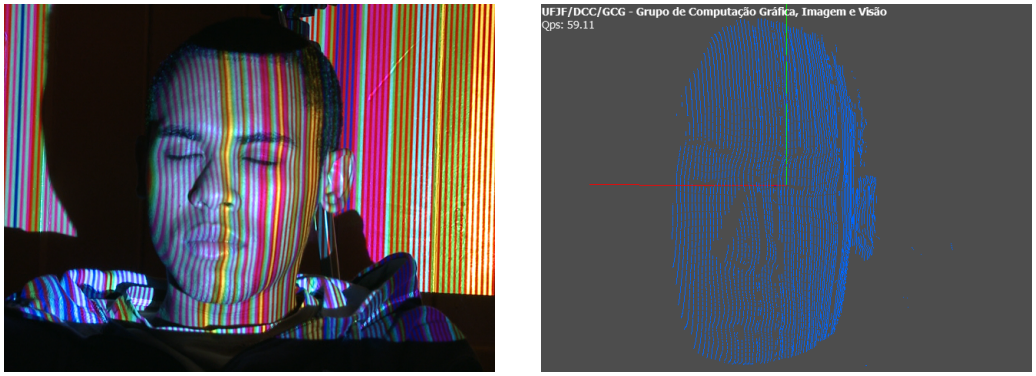


(b)

Figura 4.8: Resultados do Experimento 2 utilizando (a) ICP original e (b) adaptação BCSL.

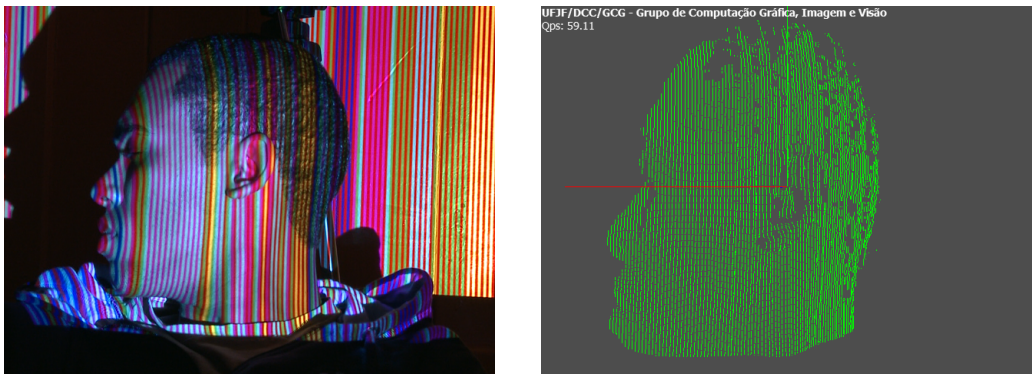
4.3 Experimento 3: Rosto

Para o terceiro experimento foram capturadas imagens das projeções sobre o rosto de uma pessoa. Para o modelo, foi feita uma captura frontal e para os dados uma lateral. As digitalizações são exibidas nas Figuras 4.9 e 4.10. A relação inicial entre o modelo e os dados a serem ajustados pode ser visto na Figura 4.11.



(a) (b)

Figura 4.9: Rosto modelo (a) capturado e (b) digitalizado.



(a) (b)

Figura 4.10: Rosto a ser ajustado (a) capturado e (b) digitalizado.

Assim como no Experimento 2, foi realizada uma transformação brusca. Desta vez foi imposta uma rotação de aproximadamente 90^{o}. A Figura 4.12 apresenta os resultados dos diferentes algoritmos. Os dados estatísticos estão presentes na Tabela 4.3.

O resultado obtido com a aplicação do algoritmo original não apresenta transformações quanto à rotação original do objeto de estudo. Os pontos a serem ajustados foram apenas levados à posições mais próximas do modelo. Nos algoritmos adaptados, como restringe-se o espaço de

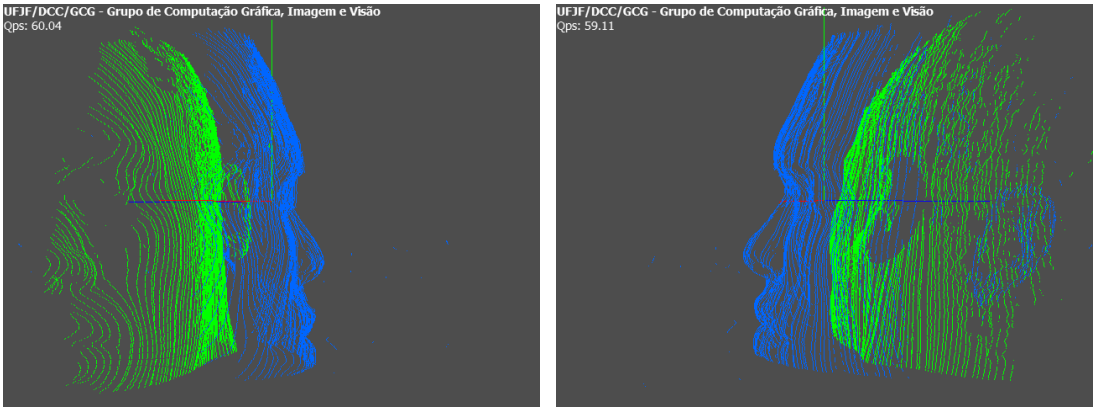
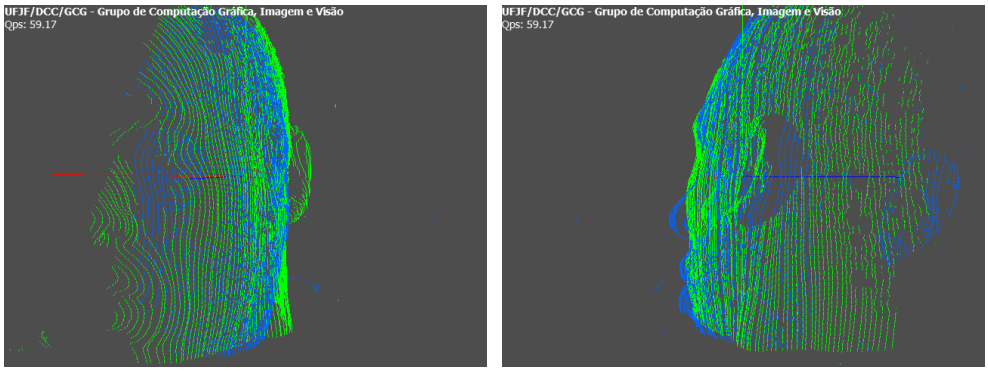
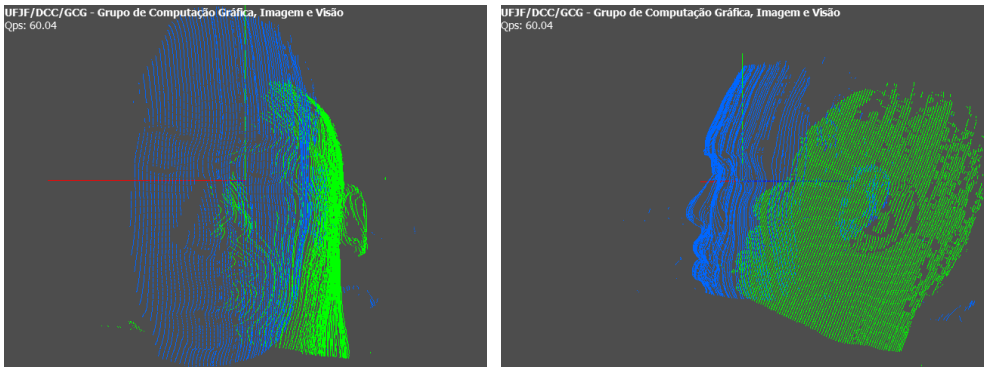


Figura 4.11: Experimento 3 antes das transformações.



(a)



(b)

Figura 4.12: Resultados do Experimento 3 utilizando (a) ICP original e (b) adaptação BCSL.

busca e já se pré-estabelece uma correspondência inicial, as transformações obtidas foram mais satisfatórias. O conjunto de dados sofreu uma grande transformação quanto ao eixo de rotação original ao que o rosto foi submetido. Mas, não realizou uma transformação final que alinhasse de fato os pontos dos dois conjuntos. Deve-se ressaltar que a rotação imposta no objeto de

Rosto	**Total de Pontos**		
Modelo	24234		
Dados	29782		
	Iterações	**Tempo (s)**	**Diferença de Erros**
ICP original	10	78,917	0.007668735325
ICP original com melhorias	10	61,668	0.007668735325
Adaptação BCSL	3	21,368	0.000514942483
Adaptação BCSL com melhorias	3	1.047	0.000514942483

Tabela 4.3: Resultados do Experimento 3.

interesse foi brusca, deixando poucos pontos realmente comuns entre as duas capturas.

4.4 Análise Geral

Observando os resultados dos três experimentos, foi possível constatar que a diminuição do espaço de busca resultou em convergências com menor número de iterações. Mas, a adaptação do algoritmo original impôs uma maior restrição quanto à disposição dos objetos de estudo. Ou seja, estabeleceu que estes devem possuir alguma área comum em um intervalo de faixas projetadas. Pois, caso contrário, não encontrarão pontos correspondentes.

O conjunto de pontos final do Experimento 2 utilizando o algoritmo adaptado com melhorias pode ser observado na Figura 4.13. O resultado obtido necessitou da inserção de um deslocamento inicial, visto que o o sistema, devido à uma translação brusca, não apresentou interseções entre os conjuntos modelo e dados.

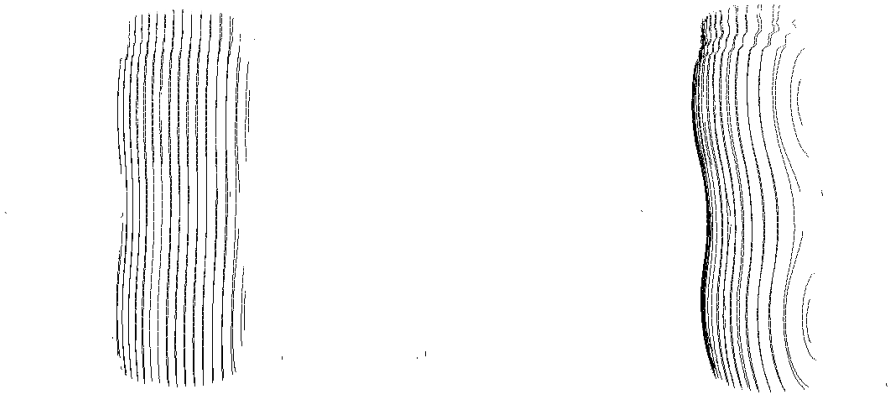


Figura 4.13: Nuvem de pontos resultante do Experimento 2.

Nos experimentos 1 e 3, onde essa restrição não esteve presente, o algoritmo adaptado apresentou resultados satisfatórios em um tempo muito menor. No caso do Experimento 1 onde os conjuntos modelo e dados possuíam um grande número de ponto comuns reais, os resultados

foram muito semelhantes, diferindo no número de iterações necessárias. A Figura 4.14 mostra a nuvem de pontos resultante do processamento. Pode-se observar que os conjuntos foram bem alinhados. Isto foi possível porque, devido à existência de muitos pontos reais comuns ao conjunto modelo e ao conjunto dos pontos a serem ajustados, correspondências mais precisas foram encontradas.

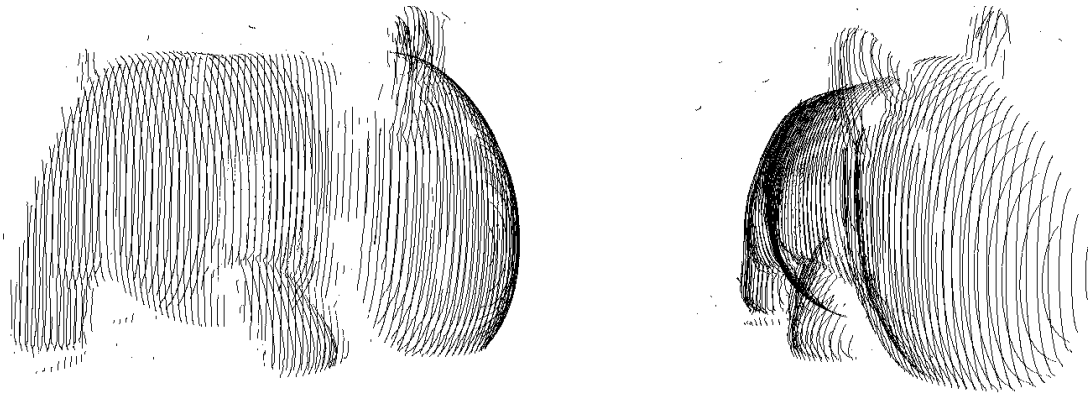


Figura 4.14: Nuvem de pontos resultante do Experimento 1.

Quanto ao Experimento 3 (Fig. 4.15), as disposições dos objetos de estudo nas duas cenas capturadas apresentam poucos pontos em comum. Portanto, o algoritmo adaptado, por estabelecer uma relação inicial entre os pontos dos dois conjuntos e restringir o espaço de busca, apresentou resultados satisfatórios em tempo muito menor que o algoritmo original. Isto pôde ser concluído pois o ICP original procura relações com todos os pontos presentes nas duas nuvens de pontos, sendo que a maioria não são correspondentes reais. Por isto, o algoritmo original não apresentou rotações que se assemelhassem com a imposta sobre o objeto real.

Figura 4.15: Nuvem de pontos resultante do Experimento 3.

5 *Conclusão*

Neste trabalho foram apresentados os fundamentos necessários para a compreensão do funcionamento de um digitalizador tridimensional que utiliza o conceito de luz estruturada. Além da calibração do sistema, foi descrita uma abordagem específica de análise centrada na transição entre as faixas projetadas e não nas faixas em si. Foram demonstrados os processos de codificação e decodificação de um padrão de projeção em particular chamado *(b,s)-BCSL* (Seção 2.1.2).

Uma das formas de minimização de distâncias entre diferentes nuvens de pontos foi introduzida. O algoritmo ICP (Seção 2.2) foi utilizado para aproximar conjuntos de pontos adquiridos através de sucessivas capturas de um digitalizador. O objetivo desta utilização era integrar capturas de diferentes perspectivas de um objeto, possibilitando a criação de um modelo 3D completo. Foi descrita também a estruturação do algoritmo ICP em forma de uma classe.

Com os estudos sobre o padrão BCSL, propôs-se utilizar as informações de fronteiras de faixas para dar maior precisão e reduzir o espaço de busca do algoritmo ICP. A modificação consistia em estabelecer correspondências diretas entre faixas equivalentes de distintas aquisições de pontos. Juntamente com a adaptação do algoritmo, foram apresentadas algumas melhorias implementadas na estrutura de dados de forma a reduzir o processamento necessário pelo algoritmo, e consequentemente seu tempo total de execução.

Observou-se que as alterações, na maioria dos testes, apresentavam resultados equivalentes em um tempo muito menor que o ICP original. Em um caso especial apresentado no Experimento 3 (Seção 4.3), as transformações geradas pelo algoritmo modificado assemelharam-se mais às sofridas pelo objeto real. Mas, notou-se que realizar tais adaptações resultou numa maior restrição quanto ao movimento dos objetos em cena. Isto porque as disposições dos objetos de interesse nos diferentes conjuntos devem apresentar alguma área comum para que sejam encontradas correspondências.

Para trabalhos futuros, pretende-se utilizar também parâmetros de textura no algoritmo ICP. Ou seja, considerando no estabelecimento de correspondências, além da distância espacial en-

tre os pontos, diferenças de cores. Desta forma tentando obter uma maior precisão no registro dos pontos reais. Deseja-se também conseguir reduzir o processamento de forma a possibilitar alinhamentos em tempo real, utilizando digitalizadores em capturas de vídeo e não apenas imagens.

Referências Bibliográficas

BESL, P. J.; MCKAY, N. D. A method for registration of 3-d shapes. *IEEE Trans. Pattern Anal. Mach. Intell.*, IEEE Computer Society, Washington, DC, USA, v. 14, p. 239–256, February 1992. ISSN 0162-8828.

CHANG, M.-C.; LEYMARIE, F. F.; KIMIA, B. B. Surface reconstruction from point clouds by transforming the medial scaffold. *Comput. Vis. Image Underst.*, Elsevier Science Inc., New York, NY, USA, v. 113, p. 1130–1146, November 2009. ISSN 1077-3142.

CODA, C. et al. Um sistema generico de calibracao de camera. In: VI WORKSHOP DE TESES DE DISSERTAÇÕES. *Proceedings of SIBGRAPI*. [S.l.], 2007.

GAMBINO, M. et al. A 3d scanning device for architectural relieves based on time-of-flight technology. In: DICKMANN, K.; FOTAKIS, C.; ASMUS, J. (Ed.). *Lasers in the Conservation of Artworks*. [S.l.]: Springer Berlin Heidelberg, 2005, (Springer Proceedings Physics, v. 100). p. 469–476.

HALL-HOLT, O.; RUSINKIEWICZ, S. Stripe boundary codes for real-time structured-light range scanning of moving objects. In: *Eighth International Conference on Computer Vision (ICCV)*. [S.l.: s.n.], 2001.

HOPPE, H. et al. Surface reconstruction from unorganized points. *COMPUTER GRAPHICS (SIGGRAPH 92 PROCEEDINGS)*, p. 71–78, 1992.

MANDOW, A. et al. Fast range-independent spherical subsampling of 3d laser scanner points and data reduction performance evaluation for scene registration. *Pattern Recogn. Lett.*, Elsevier Science Inc., New York, NY, USA, v. 31, p. 1239–1250, August 2010. ISSN 0167-8655.

MARTINI, K. Non-linear structural analysis as real-time animation. *Proceedings of the Computer-Aided Architectural Design Futures 2001 Conference*, p. 643–656, 2001.

PARK, S.-Y.; PARK, G. G. Active calibration of camera-projector systems based on planar homography. In: *ICPR*. [S.l.: s.n.], 2010. p. 320–323.

RUSINKIEWICZ, S.; HALL-HOLT, O.; LEVOY, M. *Real-Time 3D Model Acquisition*. 2002.

SA, A.; CARVALHO, P. C.; VELHO, L. (b, s)-bcsl : Structured light color boundary coding for 3d photography. In: *Proceedings of 7th International Fall Workshop on VISION, MODELING, AND VISUALIZATION*. [S.l.: s.n.], 2002.

TSAI, R. Y. Radiometry. In: WOLFF, L. B.; SHAFER, S. A.; HEALEY, G. (Ed.). , USA: Jones and Bartlett Publishers, Inc., 1992. cap. A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses, p. 221–244.