Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Representação Esparsa para Preenchimento de Buracos de Expansão em Sínteses de Vistas Baseada em Profundidade

Danilo Amaral Ribeiro

Dissertação apresentada como requisito parcial para
conclusão do Mestrado em Informática

Orientador
Prof. Dr. Bruno Luiggi Macchiavello Espinoza

Coorientador
Prof. Dr. Camilo Chang Dorea

Brasília
2017

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Representação Esparsa para Preenchimento de Buracos de Expansão em Sínteses de Vistas Baseada em Profundidade

Danilo Amaral Ribeiro

Dissertação apresentada como requisito parcial para
conclusão do Mestrado em Informática

Prof. Dr. Bruno Luiggi Macchiavello Espinoza (Orientador)
CIC/UnB

Prof.a Dr.a Mylène C. Q. de Farias
UnB

Prof. Dr. Francisco Assis de O. Nascimento
UnB

Prof. Dr. Bruno Luiggi Macchiavello Espinoza
Coordenador do Programa de Pós-graduação em Informática

Brasília, 6 de julho de 2017

Dedicatória

Dedico este trabalho ao meu pai Ribeiro, sinônimo de ética e profissionalismo em nossa área, seu ensinamentos técnicos e de como se comportar diante das pessoas e problemas é uma inspiração para minha profissão e para a minha vida. Obrigado Zé!

Agradecimentos

Agradeço a Deus, o grande Mestre, sem dúvida sempre estou em seu amparo, nos momentos que estão fora da alçada dos homens ele permitiu que as coisas acontecessem para que chegasse esse dia. Me motiva sempre a doar o meu máximo em todas as atividades do dia a dia e nos dá sempre a oportunidade de superar as adversidades.

Ao Professor Bruno pelas orientações sempre pertinentes e pela disponibilidade e paciência fornecida. E também aos demais professores pelo conhecimento transmitido em suas matérias, e outros professores e funcionários que em algum momento dispôs de seu tempo para tratar algum assunto em meu nome.

Aos meus pais Mara e Ribeiro, sempre em meus pensamentos e com certeza eu nos deles, apoio e força em todos os momentos bons e ruins. A minha avó Noca que sempre reza para que tudo ocorra bem. Meu irmão Marcelo pela grande ajuda nas imagens para esse trabalho. Tia Deja, Tio Ely e Gabriella por permitirem uma casa e seu carinho durante dois anos em Brasília. Tia Marilda e Preta, Tio Edilton, Mayana, Robson e Anna Beatriz pela torcida e alegria ao meu ver.

A minha namorada Nathália, que mais uma vez se viu trocada por "outra" (a Dirce), cada passo dado em minha vida tenho ela ao meu lado, sua torcida, seu amor, minha inspiração e meu refúgio quando precisei extravasar nas dificuldades. Te amo!!

Aos amigos feitos na UnB: Daniel *Dennis* e Elton *Wheyton Protein*, Gizele e Iasmini, Dario, Jeremias, Aldo, Michel e vários outros pelos momentos de risadas, estudos, preocupações proporcionados nesses mais de dois anos. Aos grandes amigos Bruno, Kaimã e Marco e a todos os GEN e as GEN de Palmas... sempre em unidade.

A todos que estão ou passaram pela unidade GEN em Brasília e aos focolarinos pela palavras de sabedoria e experiências compartilhadas.

Enfim agradeço a todos aqueles que de algum modo permitiram que eu chegasse a esse momento. Cada contribuição, mesmo que pequena, estará guardado em minha lembrança.

Resumo

Vídeo de ponto de vista livre (FVV - do inglês *Free Viewpoint Video*) permite diferentes opções de pontos de vista de uma mesma cena tri-dimensional (3D). Esse tipo de sistema normalmente possui um custo elevado devido aos equipamentos que devem ser utilizados para captura dos vários pontos de vista. Além disso, para realmente obter um sistema de FVV é necessário sintetizar vistas virtuais baseadas em imagens de referência fornecidas pelo sistema. Essa síntese corresponde a um ponto de vista que não é fornecido, mas que pode ser criado através de relações dos pixels originais. Um exemplo prático consiste em um observador que assiste a cena 3D e as imagens são sintetizadas com base na posição da cabeça do observador, que pode ocorrer em várias direções. Artigos da área retratam principalmente movimentos horizontais em relação a cena sendo que poucos levam em consideração movimentos de aproximação e distanciamento. Independente do deslocamento realizado a vista sintetizada apresenta buracos de pixels sintetizados que não possuem referência na imagem original, isso gera os erros de desoclusão e buracos de expansão. Este trabalho é um dos poucos na literatura que visa implementar esse processo de síntese de vista, preenchendo os buracos de expansão gerados com técnicas de inpainting. Para isso foi utilizando a técnica de representação esparsa com o treinamento de dicionários com regras não-paramétricas bayesianas. Para avaliar os resultados alcançados foram utilizados métricas objetivas de comparação de imagem. Os resultados indicam um ganho de até 6.19 dB comparado com método de *inpainting* usado no software referência VSRS+.

Palavras-chave: Vista Virtual, interpolação, buracos de expansão, representação esparsa

Abstract

Free Viewpoint Video (FVV) allows various observational points of the same three-dimensional (3D) scene. A FVV system is usually very expensive due to the equipment that must be used to capture the different view-points. Moreover, in order to achieve real FVV it is required to synthesize a virtual view based on the the acquire images. This synthesis corresponds to a view-point that is not provided, but can be created by spatial relations of the original pixels. A practical example would be that of an observer watching a 3D scene, where the synthesized images are based on the position of the viewer's head, which can occur from several angles. Several articles of the area mainly portray horizontal movements in relation to the acquired signals, only a few consider movements of approximation and distance. Regardless of the displacement performed, the synthesized view has holes, which are synthesized pixels that do not have reference in the original image, this generates the disocclusion errors and expansion holes. This work is one of the few in the literature that aims to implement the synthesis process by filling the expansion holes generated with inpainting techniques. Specifically, we will be using sparse representation with the training of dictionaries with Bayesian nonparametric rules to perform the inpainting process. In order to evaluate the achieved results objective metrics, such as PSNR, were used. Our results yield a gain of up to 6.19 dB compared to the inpainting method used in the reference software VSRS+

Keywords: Virtual View, image interpolation, expansion holes, sparse representation

Sumário

1 Introdução **2**
1.1 Justificativa 6
1.2 Objetivo Geral 7
1.3 Objetivo Específicos 7
1.4 Organização 7

2 Fundamentação Teórica **8**
2.1 Aquisição de sistemas Multi-vistas 8
2.1.1 Câmera *Pinhole* 11
2.1.2 Geometria para duas câmeras 14
2.1.3 Mapas de Profundidade 14
2.2 *Depth-Image-based Rendering* (DIBR) 18
2.3 Interpolação e Inpainting 21
2.4 Representação Esparsa 23
2.5 Projeto de Dicionários 27
2.5.1 Análise de fatores 27
2.5.2 Modelagem estatística 28
2.6 Modelos Bayesianos e Não-paramétricos Bayesianos 29
2.7 Funções de distribuição de probabilidades 30
2.7.1 Função de distribuição Beta 31
2.7.2 Função de distribuição Bernoulli 31

3 Revisão da Literatura **32**
3.1 Trabalhos em síntese de vistas 33
3.2 Trabalhos em interpolação de desoclusões 35
3.3 Trabalhos em preenchimento de buracos de expansão 37
3.4 Trabalhos em Representação esparsa 38
3.4.1 K-SVD 39
3.4.2 OMP 39

4 Metodologia **40**
4.1 Gerando a vista virtual 41
4.2 Identificação de Buracos de Expansão 43
4.3 *Inpainting* por Representação esparsa 45
4.3.1 Formulação do processo Beta-Bernoulli 45
4.3.2 *Inpainting* dos buracos de expansão 47
4.4 Métrica de qualidade (PSNR) 49

5 Resultados **51**
5.1 Síntese de vistas 52
5.2 Identificando buracos de expansão 53
5.3 Resultados comparados com artigos da área 55
5.4 Resultados comparados com mais imagens e outros algoritmos 56
5.4.1 *Art* 59
5.4.2 *Laundry* 60
5.4.3 *Moebius* 61
5.4.4 *Dolls* 62
5.4.5 *Books* 63
5.4.6 *Reindeer* 64
5.4.7 *Aloe* 65
5.4.8 *Baby* 66
5.4.9 *Bowling* 67
5.4.10 *Monopoly* 68
5.4.11 *Plastic* 69
5.4.12 *Rocks* 70

6 Conclusões **71**

Referências **73**

Lista de Figuras

1.1 Problema de síntese de vista. A partir das câmeras 0 e 1, estimar a imagem na posição α. Retirado de [1] 3
1.2 Imagem convencional de textura original e seu respectivo mapa de profundidade 4
1.3 Problema de síntese de Vista. Estimar posição das imagens com câmeras disponíveis. Adaptado de [2] 4
1.4 Desoclusões (área interna da linha verde) e Buracos de Expansão (área interna da linha amarela) 5

2.1 Sequência de quadros de um vídeo digital de múltiplas vistas. Adaptado de [3] 9
2.2 Quadros iniciais de duas vistas de uma sequencia de vídeo: A da esquerda é a primeira vista e a da direita a segunda vista. Retirado de [3] 10
2.3 Projeção do ponto 3D real no plano da imagem em uma câmera *Pinhole*. Adaptado de [4] 12
2.4 Geometria para duas câmeras(vistas). Adaptado de [4]. 15
2.5 Dois tipos de equipamentos para obtenção de profundidade da cena. ToF (*Time-of-flight*) e o *Microsoft Kinect* 16
2.6 Geometria da câmera em um sistema estereotificado. 16
2.7 (a) Geração de vista virtual com DIBR e (b) Ilustração de um *framework* básico de vista virtual usando duas câmeras de entrada (referência) A e B para sintetizar a vista virtual C. Figura adaptada de [4] 20
2.8 Ilustração do um ponto 3D no mundo projetados na vista de referência e vista virtual. Adaptada de [4] 21
2.9 Ilustração do problema de *inpainting* a partir da perspectiva do *inpainting* puro 23
2.10 Modelo visual para um sinal com representação esparsa. 25
2.11 Modelo visual para modelar um dicionário 27
2.12 Problema de regressão: linear (esquerda) e não-linear (direita). Nos dois casos a função de regressão (em azul) é parâmetro do modelo. [5] 29

2.13 Curvas de densidade de distribuição Beta para diversos valores de α e β . . 31

3.1 Categorias para as técnicas de renderização e seus respectivas classificações de métodos 33

4.1 Diagrama das etapas 40
4.2 Visão lateral da câmera v_0 (vista de referência) posicionada em relação a imagem 42
4.3 Visão lateral da câmera v_0 (vista de referência) deslocada para direita (*zoom-out* para a vista virtual v_r) 42
4.4 Visão lateral da câmera v_r (vista de referência) deslocada para esquerda (*zoom-in* para a vista virtual v_0) 42
4.5 Um bloco do mapa de profundidade separado por camadas 1 e 2 de intensidades e p de pixels vazios e seu histograma 44
4.6 A camada rotulada como 1 em (i) considera todos os pixels da camada 2 e p como pixels vazios e a camada 2 em (ii) considera somente os pixels marcados como p como vazios. 45
4.7 Mapeamento do pixel p considerado vazio da vista virtual para referência para consideração de proximidade para detecção de buraco de expansão . 46
4.8 Passos para redimensionamento e preenchimento recursivo dos buracos de expansão 48
4.9 União de blocos para aprendizado de dicionário 49

5.1 Imagem *Art* - **(a)** Imagem original e **(b)** Mapa de Profundidade. Imagem *Dolls* - **(a)** Imagem original e **(b)** Mapa de profundidade 52
5.2 Imagem *Art* - **(a)** vista virtual v_r e **(b)** mapa de profundidade com *zoom out* e $\Delta z = 0.0022$. 53
5.3 Imagem *Art* - vista virtual v_0 - **(a)** Textura e **(b)** mapa de profundidade. . 54
5.4 Buracos de Expansão (*pixels* brancos) identificados 54
5.5 Imagens para *Art* - **(a)** *Ground truth*, **(b)** *zoom-out*, **(c)** *zoom-in*, **(d)** máscara dos buracos de expansão e **(e)** vista preenchida. 56
5.6 Imagens para *Laundry* - **(a)** *Ground truth*, **(b)** *zoom-out*, **(c)** *zoom-in*, **(d)** máscara dos buracos de expansão e **(e)** vista preenchida. 57
5.7 (a) vista virtual,(b) máscara com buracos de expansão e (c) pixels preenchidos com representação esparsa proposta 57
5.8 Imagens para *Art* - **(a)** *Ground truth*, **(b)** *zoom-out*, **(c)** *zoom-in*, **(d)** máscara dos buracos de expansão e **(e)** vista preenchida com algoritmo proposto. 59

5.9 Imagens preenchidas para *Art* - **(a)** VSRS+, **(b)** Interpolação Linear, **(c)** K-SVD, **(d)** OMP e **(e)** algoritmo proposto 59
5.10 Imagens para *Laundry* - **(a)** *Ground truth*, **(b)** *zoom-out*, **(c)** *zoom-in*, **(d)** máscara dos buracos de expansão e **(e)** vista preenchida com algoritmo proposto. 60
5.11 Imagens preenchidas para *Art* - **(a)** VSRS+, **(b)** Interpolação Linear, **(c)** K-SVD, **(d)** OMP e **(e)** algoritmo proposto. 60
5.12 Imagens para *Moebius* - **(a)** *Ground truth*, **(b)** *zoom-out*, **(c)** *zoom-in*, **(d)** máscara dos buracos de expansão e **(e)** vista preenchida com algoritmo proposto. 61
5.13 Imagens preenchidas para *Moebius* - **(a)** VSRS+, **(b)** Interpolação Linear, **(c)** K-SVD, **(d)** OMP e **(e)** algoritmo proposto. 61
5.14 Imagens para *Dolls* - **(a)** *Ground truth*, **(b)** *zoom-out*, **(c)** *zoom-in*, **(d)** máscara dos buracos de expansão e **(e)** vista preenchida com algoritmo proposto. 62
5.15 Imagens preenchidas para *Dolls* - **(a)** VSRS+, **(b)** Interpolação Linear, **(c)** K-SVD, **(d)** OMP e **(e)** algoritmo proposto. 62
5.16 Imagens para *Books* - **(a)** *Ground truth*, **(b)** *zoom-out*, **(c)** *zoom-in*, **(d)** máscara dos buracos de expansão e **(e)** vista preenchida com algoritmo proposto. 63
5.17 Imagens preenchidas para *Books* - **(a)** VSRS+, **(b)** Interpolação Linear, **(c)** K-SVD, **(d)** OMP e **(e)** algoritmo proposto. 63
5.18 Imagens para *Reindeer* - **(a)** *Ground truth*, **(b)** *zoom-out*, **(c)** *zoom-in*, **(d)** máscara dos buracos de expansão e **(e)** vista preenchida com algoritmo proposto. 64
5.19 Imagens preenchidas para *Reindeer* - **(a)** VSRS+, **(b)** Interpolação Linear, **(c)** K-SVD, **(d)** OMP e **(e)** algoritmo proposto. 64
5.20 Imagens para *Aloe* - **(a)** *Ground truth*, **(b)** *zoom-out*, **(c)** *zoom-in*, **(d)** máscara dos buracos de expansão e **(e)** vista preenchida com algoritmo proposto. 65
5.21 Imagens preenchidas para *Aloe* - **(a)** VSRS+, **(b)** Interpolação Linear, **(c)** K-SVD, **(d)** OMP e **(e)** algoritmo proposto. 65
5.22 Imagens para *Baby* - **(a)** *Ground truth*, **(b)** *zoom-out*, **(c)** *zoom-in*, **(d)** máscara dos buracos de expansão e **(e)** vista preenchida com algoritmo proposto. 66
5.23 Imagens preenchidas para *Baby* - **(a)** VSRS+, **(b)** Interpolação Linear, **(c)** K-SVD, **(d)** OMP e **(e)** algoritmo proposto. 66

5.24 Imagens para *Bowling* - **(a)** *Ground truth*, **(b)** *zoom-out*, **(c)** *zoom-in*, **(d)** máscara dos buracos de expansão e **(e)** vista preenchida com algoritmo proposto. 67
5.25 Imagens preenchidas para *Bowling* - **(a)** VSRS+, **(b)** Interpolação Linear, **(c)** K-SVD, **(d)** OMP e **(e)** algoritmo proposto. 67
5.26 Imagens para *Monopoly* - **(a)** *Ground truth*, **(b)** *zoom-out*, **(c)** *zoom-in*, **(d)** máscara dos buracos de expansão e **(e)** vista preenchida com algoritmo proposto. 68
5.27 Imagens preenchidas para *Monopoly* - **(a)** VSRS+, **(b)** Interpolação Linear, **(c)** K-SVD, **(d)** OMP e **(e)** algoritmo proposto. 68
5.28 Imagens para *Plastic* - **(a)** *Ground truth*, **(b)** *zoom-out*, **(c)** *zoom-in*, **(d)** máscara dos buracos de expansão e **(e)** vista preenchida com algoritmo proposto. 69
5.29 Imagens preenchidas para *Plastic* - **(a)** VSRS+, **(b)** Interpolação Linear, **(c)** K-SVD, **(d)** OMP e **(e)** algoritmo proposto. 69
5.30 Imagens para *Rocks* - **(a)** *Ground truth*, **(b)** *zoom-out*, **(c)** *zoom-in*, **(d)** máscara dos buracos de expansão e **(e)** vista preenchida com algoritmo proposto. 70
5.31 Imagens preenchidas para *Rocks* - **(a)** VSRS+, **(b)** Interpolação Linear, **(c)** K-SVD, **(d)** OMP e **(e)** algoritmo proposto. 70

Lista de Tabelas

5.1 Lista de imagens e respectivas dimensões. 51
5.2 Valores de profundidade máxima e Δz_{max} 53
5.3 Comparação em PSNR dos métodos de *inpainting* 55
5.4 PSNR para imagens 1390×1110 58
5.5 PSNR para imagens 640×552 58

Lista de Abreviaturas

Abreviatura	**Significado**
2D	Duas dimensões (bi-dimensional)
3D	Três dimensões(tri-dimensional)
3DTV	3D *Television*
AGFT	Transformada de Fourier em Grafos
BP	*Basis Pursuit*
CG	Computação Gráfica
DIBR	*Depth-image-based Rendering*
FOCUSS	*Focal Under-determined System Solver*
FVV	*Free ViewPoint Video*
GBT	Transformada Baseada em Grafos (*Graph Based Transform*)
IBR	*Image-based Rendering*
IRLS	*Fast Iterative Reweighted Least Square*
MAP	*Maximum a Posteriori*
MP	*Matching Pursuit*
MVD	*Multiview-plus-depth*
MPEG	*Motion Picture Expert Group*
MRF	Campos randômicos de Markov (*Markov Random Field*)
MVP	*Multi View Profile*
NLGBT	Transformada Baseada em Grafos Não Locais (*Non-Local Graph Based Transform*)
OMP	*Orthogonal Matching Pursuit*
PBR	*Probability-based Rendering*
PDE	Equação Diferencial Parcial (*Partial Differential Equation*)
PSNR	*Peak Signal-to-noise Ratio*
SVD	*Singular Value Decomposition*
SSMP	*Steady-state matching probability*
V+D	*Video-plus-depth*
VSRS	*View Synthesis Reference Software*

Capítulo 1

Introdução

A imagem percebida pelo cérebro humano pelo sistema de visão, resulta da combinação de duas imagens, cada uma captada por um olho. Este par de imagens recebe o nome de par estereoscópico. Esse sistema de múltiplas vistas é o utilizado por seres humanos e alguns animais [4].

Essa visão estereoscópica da cena oferece ao indivíduo uma série de informações referentes ao ambiente, que não poderiam ser obtidas por meio de uma visão monocular. Essas informações adquiridas formam a base de capacidades do tipo: identificar a relação espacial entre os objetos observados (profundidade relativa) e concentrar-se nos elementos posicionados a determinada distância [6]. Isso proporciona uma percepção 3D de objetos em uma cena.

Num sistema de aquisição de vídeo digital, um sinal multi-vistas é normalmente obtido a partir de um conjunto de câmeras sincronizadas que capturam a mesma cena a partir de diferentes pontos de vista. Esta técnica permite a criação de aplicativos tais como o Vídeo de Ponto de Vista Livre (FVV - do inglês *Free Viewpoint Video*) [7]. O FVV fornece a capacidade para que os usuários selecionem de forma interativa um ponto de vista da cena 3D do vídeo. Isto pode ser feito capturando a cena do vídeo a partir de múltiplos pontos de vista. No entanto, a vista selecionada está restrita ao conjunto de pontos de vista de câmeras físicas existentes [8], o que torna o custo de aquisição de câmeras e largura da banda necessária para transmissão bastante elevado.

Entretanto, é possível criar uma vista virtual a partir do ponto de vista escolhido. Computadores realizam essa tarefa encontrando correspondências entre os pontos que são vistos em uma imagem e os mesmos pontos vistos em outra imagem. Essas correspondências consistem em estimar para os *pixels* de uma imagem 2D para um sistema de coordenadas 3D no sistemas de coordenadas do mundo. Posteriormente, ele reprojeta cada ponto 3D no plano de imagem de destino [9]. Com isso é possível gerar uma síntese de uma imagem que fornece uma visão intermediária dos outros pontos de vistas (ima-

gens) fornecidos, este processo pode ser visto pela Figura 1.1. Essa abordagem difere da renderização em computação gráfica em que esta assume que um modelo 3D é conhecido e dado a *priori* como entrada. Devido à capacidade de alterar o ponto de vista, essa visualização interativa fornece um experiência para o usuário mais rica do que imagens 2D tomadas de pontos de vista fixos.

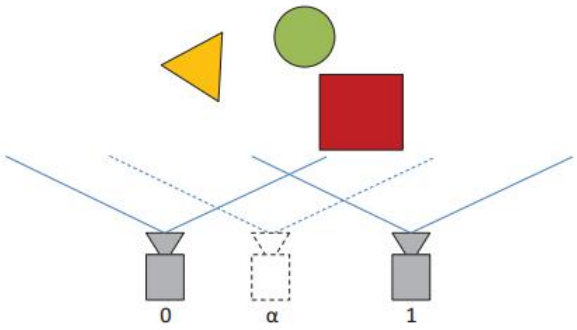


Figura 1.1: Problema de síntese de vista. A partir das câmeras 0 e 1, estimar a imagem na posição α. Retirado de [1]

Esse sistema mostrado na Figura 1.1 utiliza uma ou várias imagens de referência e os seus respectivos mapas de profundidade ou disparidade, que possuem os valores de profundidade, ou disparidade dos objetos, estimados para cada ponto de uma imagem 2D. Um mapa de profundidade indica a distância de cada pixel com relação ao ponto de captura, que pode ser visto na Figura 1.2, já um mapa de disparidade indica a diferença entre a posição espacial de um mesmo ponto em duas imagens. A utilização do formato *video-plus-depth* (V+D) e o *multi view-plus-depth* (MVD), garantem a renderização de vistas virtuais [10], onde conhecidos os parâmetros intrísecos e extrínsecos da câmera (ou várias câmeras) utilizada na aquisição da cena, é possível estimar as posições dos pixels em outro ponto de vista. Com equipamentos sensores de profundidade como *time-of-flight* and *Microsoft Kinect* [11], a utilização de mapas de profundidade (distância dos objetos capturados numa cena 3D e a câmera utilizada para captura) tem sido mais comum.

Um método fundamental de síntese de vista consiste na Renderização Baseada em Imagem de Profundidade (DIBR - do inglês *Depth-image-based Rendering*) [12] como a chamada de deformação 3D, que produz vistas virtuais usando pixel de textura ou textura de regiões e a informação de profundidade. As vistas sintetizadas são criadas baseada nas posições de origem dessas câmeras. Os movimentos de câmera podem ocorrer ao longo de um eixo x ou y (câmera movendo para a esquerda/direita ou de cima para baixo), tem sido demonstrado que esta síntese DIBR funciona razoavelmente bem [7], sendo a abordagem convencional utilizada na literatura de síntese de vistas.

Figura 1.2: Imagem convencional de textura original e seu respectivo mapa de profundidade

.

Em aplicações imersivas, tais como teleconferência [13], é possível que os participantes observem em tempo real imagens sintetizadas a partir de uma cena 3D, cuja perspectiva deve ser adaptada em resposta à posição estimada da cabeça do espectador em relação a cena. Além de movimento da cabeça no eixo x (movimentos horizontais e verticais fixos em um plano), o movimento da cabeça no eixo z (movendo a cabeça com aproximação e distanciamento em relação a câmera) também é natural para um observador sentado, o que pode induzir mudanças de perspectiva. A Figura 1.3 mostra os movimentos que podem ser efetuados para estimar as posições das câmeras virtuais.

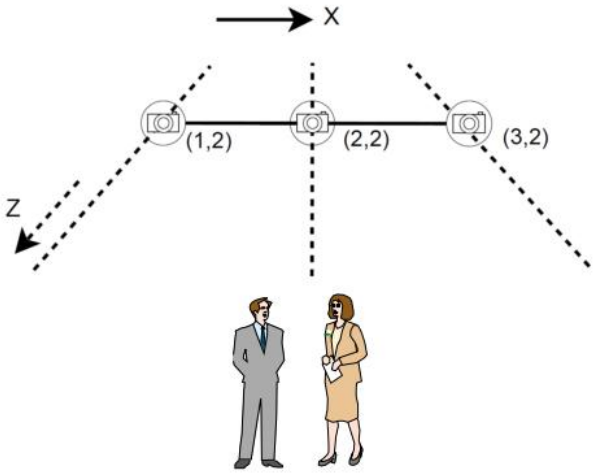


Figura 1.3: Problema de síntese de Vista. Estimar posição das imagens com câmeras disponíveis. Adaptado de [2]

Um problema inerente às sínteses de vistas virtuais no eixo x é que uma imagem intermediária sintetizada pode conte *pixels* que não possui uma referência das imagens originais. Consequentemente, a síntese de vista pode expor as oclusões da cena, pois

dependendo da posição do pixel nas imagens de referência, as áreas da cena que estão obstruídas ficarão visíveis na vista virtual, gerando os buracos na cena ou desoclusões.

No entanto, na literatura poucos trabalhos [14] [15] [16] [17] de síntese de vista baseada em DIBR tem abordado formalmente o problema de sintetizar a vista virtual correspondente ao movimento da câmera no eixo z. Trabalhos como os Cheung et. al em [14], Mao et. al em [15] e Mao et. al [16] são os pioneiros em implementar uma síntese de vista no eixo z, além de identificar e preencher os buracos de expansão com técnicas de preenchimento utilizando métodos baseados em grafos.

Quando a vista virtual está localizado mais perto da cena do que na vista de referência, os objetos mais perto da câmera vão aumentar de tamanho na visão virtual mais rápido do que objetos mais distantes. Um grande aumento no tamanho do objeto significa que uma parte de pixels amostrados a partir da superfície do objeto na vista de referência serão espalhados em uma maior área espacial, o que resulta em buracos de expansão [14]. Para complicar ainda mais, no meio desses pixels dispersos que geram os buracos de expansão, pode existir a obstrução de pixels de *background* pelo objeto mais próximo, gerando também desoclusões. A Figura 1.4 mostra cada um dos tipos de buracos.

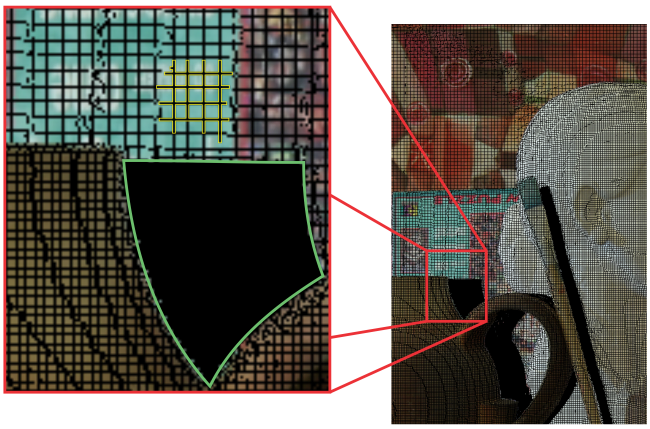


Figura 1.4: Desoclusões (área interna da linha verde) e Buracos de Expansão (área interna da linha amarela)

Assim é necessário preencher esses pixels de desoclusão e buracos de expansão resultante da síntese de vista com técnicas de interpolação ou de *inpainting*. *Inpainting* é um processo de reconstrução de partes deterioradas em imagens e vídeos e algoritmos nessa abordagem buscam completar essas áreas através da vizinhança de pixels não deteriorados dessas imperfeições. Uma boa técnica de preenchimento é aquela que mantém a coerên-

cia da área preenchida artificialmente com sua vizinhança original, causando a impressão visual de que a área preenchida é realmente parte da imagem original.

A teoria de representação esparsa é uma abordagem que pode ser aplicada em *inpainting* de imagens [18] [19]. De fato, há uma noção muito natural de esparsidade [20] nesse tipo de problema: dada um sinal de entrada, tenta-se encontrar apenas um ou vários sinais combinados entre uma grande quantidade de sinais de diferentes classes que melhor representa tal sinal de entrada. Os elementos escolhidos pertencem a um conjunto maior de elementos e apenas uma pequena quantidade (definido a priori) é escolhida para representar o sinal

Em muitas aplicações é desejável a representação de sinais/imagens usando estruturas elementares escolhidas em um conjunto chamado de dicionário, na esperança de obter uma representação onde seus coeficientes (átomos) revelem as informações relevantes. Existem vários aplicações nessa área, como no contexto de *denoising* (retirar ruído da imagem), *inpainting* [19] e codificação [21].

O dicionário pode ser obtido previamente e de tamanho fixo ou pode ser aprendido utilizando aprendizagem de máquina. Ao ser aprendido, os dados de treinamento podem ser definidos a priori ou podem ser modelados atrávez da própria imagem de teste, nesse contexto é utilizados blocos de imagens. Uma abordagem estatística é um dos elementos que permite criar um dicionário apropriado. Exemplos práticos utilizando regras não-paramétricas bayesianas têm demostrado bons resultados na literatura [22], [23].

1.1 Justificativa

Uma maneira prática de se obter um vídeo de ponto de vista livre (FVV) é através da utilização de imagens multivistas. No entanto, uma grande largura de banda se faz necessária para a transmissão de imagens multivistas, o que limita seu uso. Uma alternativa para resolver este problema é gerar vistas virtuais usando uma imagem de textura e uma imagem de profundidade correspondente [24]. Um problema inerente que surge ao gerar vistas virtuais é que as regiões abrangidas na exibição original podem causar desoclusões ou buracos de expansão nas vistas virtuais de acordo com a nova posição estabelecida.

O preenchimento das desoclusões e buracos de expansão de maneira visualmente plausível é imprescindível para minimizar a degradação da qualidade dos resultados da síntese. Portanto, o problema a ser abordado neste trabalho consiste no preenchimento dos buracos de expansão provenientes da síntese de vista através de algoritmos de *inpainting* e interpolação que utilizam imagens de profundidade como auxílio.

1.2 Objetivo Geral

O objetivo central do trabalho é desenvolver e aplicar a técnica de representação esparsa com treinamento de dicionário via regra não-paramétrica bayesiana para preenchimento somente dos buracos de expansão resultantes de síntese de vista virtuais baseada em em profundidade e textura com movimentos aparentes em relação ao eixo z.

1.3 Objetivo Específicos

- Implementar síntese de vista em relação ao eixo z;
- Testar parâmetro de distância em relação a câmera (*zoom in* e *zoom out*);
- Identificar os pixels de desoclusão e buracos de expansão decorrente da síntese de vista;
- Desenvolver técnica de treinamento de dicionário usando regras não-paramétricas bayesianas;
- Aplicar algoritmos de representação esparsa para preenchimento dos buracos de expansão.
- Comparar os resultados com outras técnicas de *inpainting* e representação esparsa.

1.4 Organização

O Capítulo 2 relata uma fundamentação teórica sobre os principais tópicos necessários para entendimento das técnicas necessárias para a síntese de uma vista virtual e de preenchimento de buracos de expansão e desoclusões. O capítulo 3 relata uma revisão da literatura referente aos principais tópicos apresentados nesse trabalho. O Capítulo 4 apresenta a metodologia da abordagem proposta, além dos algoritmos propostos e o Capítulo 5 mostra os resultados alcançados nos experimentos. O último capítulo apresenta as conclusões e os trabalhos futuros que podem ser realizados baseados neste atual trabalho.

Capítulo 2

Fundamentação Teórica

Neste capítulo serão brevemente descritas as técnicas necessárias para a síntese de vistas. Primeiramente é necessário entender o como adquirir um sistemas de múltiplas vistas, como gerar imagens com pontos de vista de posições diferentes das imagens de referência e técnicas de preenchimento dos problemas inerentes a sintetização dessas vistas.

2.1 Aquisição de sistemas Multi-vistas

A aquisição de um sistema multi-vista, relaciona-se com a captura de dados de vídeo sincronizadas representando diferentes pontos de vista de uma única cena, em contraste com os sistemas de vigilância de vídeo, que utilizam várias câmeras para cobrir visualmente um ambiente de grande escala a ser monitorado com pequena redundância [25]. Os materiais, dispositivos ou sistemas utilizados na aquisição multi-vistas abrangem várias perspectivas de um único, muitas vezes bastante restrito espaço físico e usam redundância entre as imagens. Exemplos de aplicação são:

- 3D estereoscópico ou visualização multioscópica de vídeos capturados;
- Reconstrução / virtualização de cenas:
  - Reconstrução de um ponto de vista a partir de um mapa de profundidade;
  - Reconstrução 3D de modelos digitais para objetos reais,
  - Captura de movimentos de atores virtuais para animações ou filmes;
- Diversos e complementares ajustes na produção / pós-produção de vídeos:
  - Junção de várias imagens proporcionando uma vista panorâmica ou uma imagem de alta resolução de uma cena - "*mosaicking*",

– Uma câmera virtual em movimento com tempo congelado ou muito lentamente - "*bullet time*",
– Misturar cenas virtuais/reais (realidade aumentada - AR),
– TV de ponto de vista livre - FTV(*Free View-point TV*),
– Modificação de foco após aquisição da cena (reorientação),
– Proporcionar uma maior faixa de luminosidade para melhor qualidade visual de imagens - *High Dynamic Range* - HDR,
– etc

A Figura 2.1 mostra uma cena sendo capturada por quatro câmeras posicionadas lado a lado.

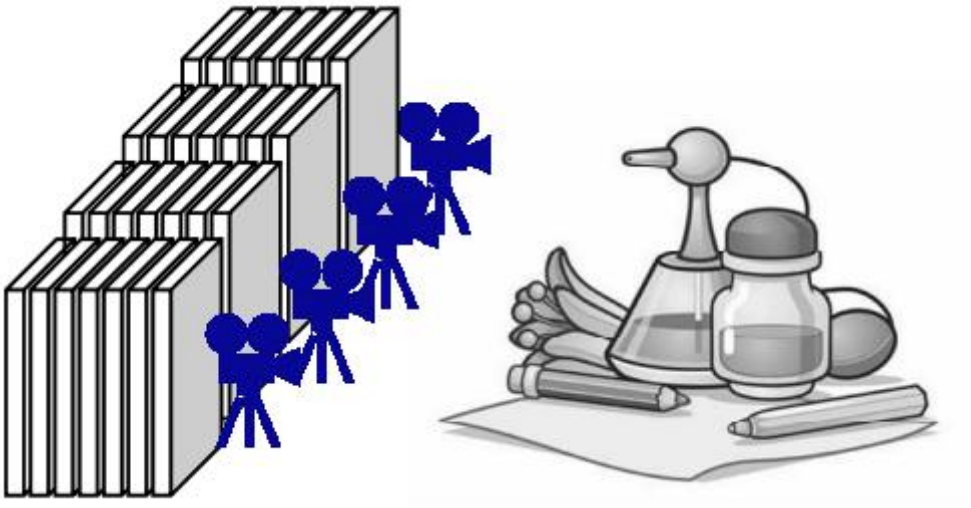


Figura 2.1: Sequência de quadros de um vídeo digital de múltiplas vistas. Adaptado de [3]

A redundância não se deve ao movimentos dos objetos, uma vez que os quadros estão sendo capturados simultaneamente. A redundância se deve ao fato de todas as câmeras observarem a mesma cena ao mesmo tempo, porém em posições distintas. A disparidade entre as vistas, ou seja o deslocamento dos objetos da cena entre os diferentes pontos observados é ilustrados na Figura 2.2.

Os quadros apresentados pela Figura 2.2 são exemplos para quantificar a disparidade entre duas vistas. A disparidade deve ser considerada para cada objeto observado na cena, dependendo da distância entre as câmeras e o objeto observado. Intuitivamente, para duas câmeras fixas separadas por uma determinada distância, quanto mais próximo o objeto se encontrar das câmeras maior será a disparidade entre as vistas. Este fato pode ser observado nos pontos $p1$ e $p2$ da Figura 2.2. Estes pontos marcam as posições do braço do

Figura 2.2: Quadros iniciais de duas vistas de uma sequencia de vídeo: A da esquerda é a primeira vista e a da direita a segunda vista. Retirado de [3]

dançarino à esquerda (no plano de fundo) e do vestido da dançarina à direita (em primeiro plano). Pela imagem observa-se um deslocamento maior do ponto $p2$ do que o ponto $p1$. Isto se deve à distância de cada objeto em relação a câmera. O dançarino está mais afastado da câmera, assim o deslocamento de *pixels* da região que o representa é menor do que o deslocamento correspondente à dançarina. Esta característica de disparidade e distância em relação a câmera deve ser levada em conta para a síntese de vistas virtuais.

Dependendo da aplicação final, o número, o *layout* e as configurações de câmeras podem variar muito. As configurações mais comuns disponíveis hoje incluem [25]:

- Sistemas Binoculares - Dois pontos de vista muito próximos, semelhantes à disposição dos olhos humanos, estes são sistemas compatíveis com a visualização em 3D estereoscópico (que exigem óculos em geral) e reconstrução de vídeo baseada em profundidade relativa dos objetos utilizada em Realidade Aumentada e *Free View-point TV*;

- Sistemas multivistas laterais ou direcionais - Fornecem múltiplos pontos de vista capturados por câmeras físicas próximas (em geral espaçadas regularmente), cada uma colocada sobre o mesmo lado de uma cena. Estes sistemas produzem uma adaptação em vídeos permitindo a visualização do 3D autoestereoscópica e dos efeitos de "bullet time". Dentro de um escopo limitado, em uma reconstrução de profundidade mais robusta está no caso particular da Realidade Aumentada e do *Free View-point TV*. O uso de diferentes perspectivas também permite a utilização diferentes configurações para cada câmera, que, com a forte redundância na captura, permite uma pós-produção de vídeo mais acentuada (Reorientação e Vídeos HDR);

- Sistemas globais multivistas ou multi-direcionais - Incluem múltiplos pontos de vista em torno de um cena. Estes sistemas são concebidos principalmente para vídeos do tipo *"bullet time"* prevendo grande movimento angular, reconstrução e captura de movimentos.

Além disso, existem sistems multi-vistas hibrídos que adicionam câmeras sensores de profundidade, que fornecem informações de profundidade relativa da cena. Um exemplo de sensor é o *Microsoft Kinect* [26].

Todos esses equipamentos necessitam de sincronização e calibração das informações captadas por câmeras diferentes que muitas vezes têm a necessidade de acompanhar os seus dados de produção de modo a capturar a cena sem perda de dados e com processamento dos dados em tempo real.

Um sistema multi-vista geralmente é usado de modo a permitir a criação de vídeos 3D, onde se tem a percepção de quão longe ou perto os objetos presentes na cena estão em relação a câmera. Uma das classes de video 3D se baseia na descrição 3D geométrica parcial da cena [27]. A geometria da cena é descrita através de uma mapa de profundidade ou imagem de profundidade, no qual especifica a distância entre um ponto no mundo real e a câmera[25]. Tipicamente, uma imagem de profundidade é estimada a partir de duas imagens por meio da identificação de pixels correspondentes nas várias vistas, isto é, ponto-correspondências, que representam o mesmo ponto da cena 3D. Usando mapas de profundidade, novos pontos de vista podem ser posteriormente sintetizados usando o algoritmo de Renderização baseada em Profundidade (DIBR - do inglês *Depth Image Based Rendering*). Isso é possível através da geometria projetiva das câmeras utilizadas para aquisição das imagens e o modelo mais simples de câmera segue o modelo *Pinhole*.

2.1.1 Câmera *Pinhole*

A geometria projetiva é uma estrutura matemática bem organizada que pode ser usada para descrever um modelo de câmera *pinhole*. Este princípio geométrico é usado para modelar a formação de imagens, a geração de vistas e a reconstrução de cenas 3D a partir de imagens múltiplas.

A câmera *pinhole* é o dispositivo de imagem mais simples como é mostrado na Figura 2.3 e seu modelo é descrito por um centro óptico v_0 (também conhecido como centro de projeção de câmera) e o plano de imagem [28]. A distância entre o centro óptico e o plano da imagem é a distância focal f. E a linha perpendicular ao plano da imagem é o eixo principal, que indica a direção de visualização da câmera e passa por V_0. A interseção do eixo principal com o plano da imagem é chamado o ponto principal.

Um sistema de coordenadas 3D é estabelecido usando V_0 como sua origem. γ_0 é a distância entre um ponto na cena 3D e V_0 (a componente projetada sobre o eixo principal). α_0 é a distância horizontal entre o ponto 3D e o eixo principal. Pela Figura 2.3 um ponto 3D Q é projetado através de V_0 até o plano da imagem no pixel P_0. A imagem projetada por uma câmera *pinhole* é virada de cabeça para baixo e da esquerda para a direita [28] e inversões são realizadas para representar as imagens como o ser humano está mais habituado.

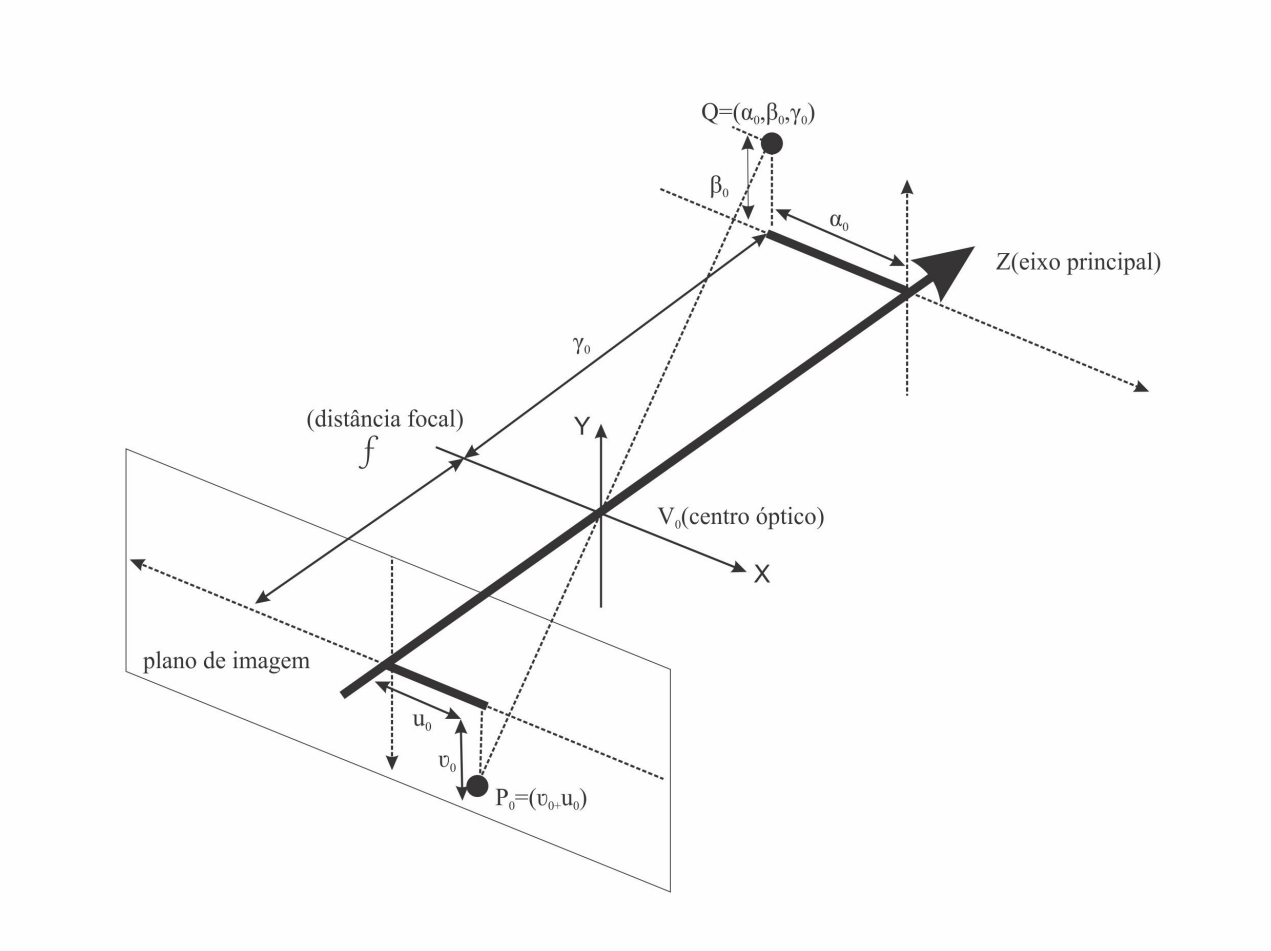


Figura 2.3: Projeção do ponto 3D real no plano da imagem em uma câmera *Pinhole*. Adaptado de [4]

A consideração é que o centro da câmera está localizado na origem do sistema de coordenadas euclidiano e que o eixo óptico é colinear com o eixo z. Supondo um ponto $Q = (\alpha_0, \beta_0, \gamma_0)^T$ 3D do mundo é projetado até o plano da imagem na posição $(u_0, v_0)^T$. O mapeamento do ponto 3D do mundo de uma cena para o plano de imagem 2D é chamado de projeção de perspectiva e usando coordenadas homogêneas pode ser reformulada como na Equação 2.1:

$$z\begin{pmatrix} u_0 \\ v_0 \\ 1 \end{pmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} \alpha_0 \\ \beta_0 \\ \gamma_0 \\ 1 \end{pmatrix}, \tag{2.1}$$

na qual $u_0 = \frac{\alpha_0 f}{z}$, $v_0 = \frac{\beta_0 f}{z}$ são posições de pixel na imagem. A matriz na equação 2.1 é chamada de matriz de projeção, que define o mapeamento linear de um ponto 3D em um plano de imagem 2D e pode ser escrito de acordo com a equação 2.2

$$z\mathbf{m} = \mathbf{PM}, \tag{2.2}$$

na qual $\mathbf{M} = (\alpha_0, \beta_0, \gamma_0, 1)^T$ e $\mathbf{m} = (u_0, v_0, 1)^T$ são coordenadas homogêneas do ponto 3D do mundo e sua coordenada de projeção no plano da imagem. P é a matriz de projeção e contém apenas informações relativas à distância focal. Normalmente, a matriz de projeção é uma matriz completa de 3×4 e pode ser decomposto em:

$$\mathbf{P} = \mathbf{K}[\mathbf{R}|\mathbf{t}], \tag{2.3}$$

na qual $\mathbf{K}$ é matriz de parâmetros intrínsecos e $[\mathbf{R}|\mathbf{t}]$ é a matriz de parâmetros extrínsecos. Os parâmetros extrínsecos são definidos como qualquer conjunto de parâmetros geométricos que identificam unicamente a transformação entre o plano da câmera e o plano 3D do mundo. Os parâmetros intrínsecos são conjunto necessários de parâmetros para caracterizar as características ópticas, geométricas e digital da câmera. O problema de estimar os valores para esses parâmetros é chamado de calibração de câmera [29].

$\mathbf{K}$ é a matriz de calibração da câmera com 5 variáveis que descreve a distância focal f, os parâmetros de distorção s, o centro da imagem o_x e o_y e quantidade de pixels da câmera s_x e s_y e é definido como:

$$\mathbf{K} = \begin{bmatrix} \frac{f}{s_x} & s & o_x \\ 0 & \frac{f}{s_y} & o_y \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.4}$$

No mundo 3D, a câmera pode estar rotacionada nos 3 eixos (x, y, z) e pode possuir um deslocamento em relação às coordenadas do objeto. As três possíveis rotações estão representadas nas matrizes abaixo:

$$R_x(\varphi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\varphi & \sin\varphi \\ 0 & -\sin\varphi & \cos\varphi \end{bmatrix} R_y(\phi) \begin{bmatrix} \cos\phi & 0 & -\sin\phi \\ 0 & 1 & 0 \\ \sin\phi & 0 & \cos\phi \end{bmatrix} R_z(\Theta) \begin{bmatrix} \cos\Theta & \sin\Theta & 0 \\ -\sin\Theta & \cos\Theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (2.5)$$

O vetor de translação t representa a orientação da câmera em relação à coordenada do ponto 3D do mundo. Além disso, a projeção não é definida para o centro de projeção de câmera.

2.1.2 Geometria para duas câmeras

Na sessão anterior, foi apresentada a geometria de uma única câmera, nesta sessão a geometria de duas vistas de câmera (câmera estéreo) é brevemente descrita. Suponha que duas câmeras capturam a mesma cena de diferentes perspectivas e o princípio é adquirir o ponto de cena M na cena 3D como mostrado na Figura 2.4. Normalmente, as informações da cena 3D estão disponíveis em ambas as vistas. Seja M um ponto 3D não ocluído que é projetado em duas imagens através de seus centros de câmera C_1 e C_2 nas posições m_1 e m_2 respectivamente. Os dois pontos são projeções do mesmo ponto 3D, de modo que são chamados de pontos correspondentes. Por outro lado, o ponto M da cena $3D$ pode ser estimado a partir dos pontos de imagem correspondentes m_1 e m_2 utilizando triangulação. Em geral, as duas câmaras C_1 e C_2 estão relacionadas pela sua posição, orientação e geometria interna. O conceito que descreve as relações entre as duas câmeras dos pontos correspondentes é chamado de geometria epipolar [30]. Usando esta geometria os pontos do objeto em uma vista da câmera podem ser extraídos do mesmo objeto na vista da outra câmera.

2.1.3 Mapas de Profundidade

Os mapas de profundidade representam a informação de profundidade das imagens. Elas são normalmente representadas por imagens em escala de cinza em que cada pixel é representado por 8 bits, havendo 256 níveis possíveis de profundidade. Eles são importantes em sistemas multi-vistas pois ajudam a determinar a relação entre os pixels das câmeras desse sistema, além de possibilitar a criação da vistas sintetizadas. A determinação da profundidade de cada pixel é realizada em distância (em metros, centímetros, etc) até um sistema de referência.

O mapa de profundidade pode ser obtido por computação gráfica, sensores que medem distância ou por análise de sistema esteriotificado de câmeras ou imagens [8]. Os dados de profundidade encontrados por computador são realizados através de modelagens em

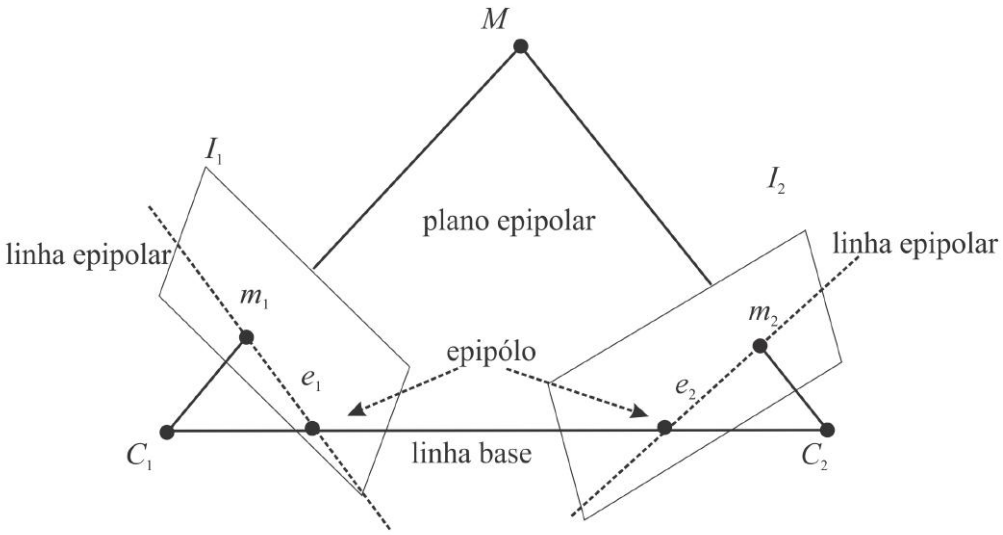


Figura 2.4: Geometria para duas câmeras(vistas). Adaptado de [4].

algoritmos especializados em construção de imagens 3D e fornecem uma precisa informação de profundidade. Os problemas comuns em profundidade a partir da correspondência estéreo, tais como diferenças de cor e oclusões são evitados, mas a desvantagem é que não são dados reais e visto que o uso em situações reais é plausível, este método torna-se limitado.

A profundidade pode ser extraída por equipamentos sensores de alcance. O princípio é realizar uma triangulação entre um ponto distinto em um objeto utilizando um *laser* e um exemplo de equipamento que utiliza essa abordagem é o *Microsoft Kinect* [31]. Outra abordagem consiste em medir a distância, de acordo com o tempo de deslocamento do sinal entre o emissor e o detector. As câmeras que dependem deste princípio são chamadas de câmeras *Time-of-Flight*(ToF) [26]. Um dos principais problemas associados aos sensores de alcance é uma discrepância de resolução entre a imagem colorida e os dados de profundidade adquiridos [32] [11]. A Figura 2.5 mostra os dois equipamentos

A forma de obtenção da profundidade é calculada a partir do cálculo da disparidade entre os quadros de vídeos representando a mesma cena. Os valores armazenados no mapa de disparidade representam a distância de cada pixel entre imagens [12].

Disparidade refere-se às diferenças espaciais entre objetos em duas imagens, esquerda e direita e é definida como sendo a distância horizontal entre dois pixels correspondentes. Esses objetos usualmente se encontram em posições diferentes nos quadros ou pode acontecer que o pixel não tenha seu homólogo. Determinar o valor da disparidade é essencial para se determinar a distância dos objetos às câmeras [33].

Figura 2.5: Dois tipos de equipamentos para obtenção de profundidade da cena. ToF (*Time-of-flight*) e o *Microsoft Kinect*
.

Dada a informação da disparidade associada a qualquer par de imagens, o cérebro gera a percepção de profundidade, fundindo essas imagens. A ideia de gerar a percepção de profundidade a partir de informações da disparidade inspirou a captura de imagens estereoscópicas de uma mesma cena, usando duas câmeras com a mesma configuração [33]. A relação entre profundidade e disparidade em uma câmera para captura das imagens, é ilustrada na Figura 2.6 onde as câmeras C_l e C_r estão separados a uma distância t_c com distância focal f, desta forma é possível estimar a profundidade Z do ponto O, utilizando a disparidade entre os pontos x e x'.

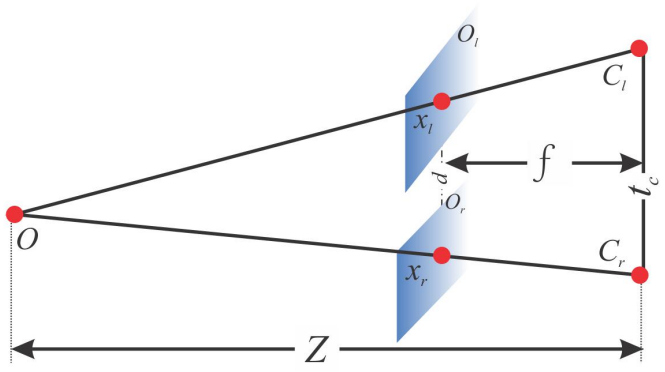


Figura 2.6: Geometria da câmera em um sistema estereotificado.

Os dados do cálculo da disparidade são armazenados na forma de mapas de disparidades, que são vetores de informação com os valores da distância do pixel horizontal para

cada coordenada de pixel da imagem. Alguns problemas como oclusão de objetos na cena podem dificultar a obtenção da disparidade [34].

O valor da disparidade entre todos os pixels da imagem direita e esquerda é obtido pela Equação 2.6. Nessa equação, são calculadas as diferenças entre dois pixels homólogos, podendo ser obtido pelo de Método de Correspondência.

$$d = |X_l - X_r| . \tag{2.6}$$

O Método de Correspondência consiste em determinar um pixel homólogo da imagem esquerda, na imagem direita. A determinação desses pixels homólogos depende da qualidade da imagem e da precisão com que a cena foi reconstruída. Os mapas de profundidade podem ser classificados como mapas esparsos ou mapas densos. Os mapas esparsos são obtidos a partir do cálculo da disparidade para alguns pontos e os mapas densos são obtidos pelo cálculo da disparidade por todas as áreas da imagem.

Uma vez determinado o valor da disparidade entre os pixels da imagem, pode-se converter esse dado em distâncias físicas [34]. Para se determinar a relação entre profundidade z com base nos valores de disparidade d, é necessário conhecer os parâmetros físicos das câmeras, t_c e f, onde t_c representa a distância entre as câmeras e f representa a distância focal de cada câmera, como demonstrado na Equação 2.7:

$$z = \frac{t_c \times f}{d}. \tag{2.7}$$

O valor de t_c e f são sempre positivos e, portanto para profundidades positiva ou infinita, a disparidade d é sempre positiva ou nula. Os valores de t_c e f devem ser descritos nas especificações de cada vídeo, pois são valores inerentes a eles levando em consideração a estrutura de filmagem [35].

Convém ressaltar que o valor da disparidade pode ser positivo, negativo ou nulo. No caso de uma disparidade nula, entende-se que foi encontrada uma distância tal que já não se consegue distinguir uma diferença de posição, pois aquele ponto encontra-se na mesma posição para ambas as câmeras. Na disparidade positiva, a imagem será formada "para dentro" do plano do display, enquanto que a disparidade negativa será formada "para fora" do plano do display.

Com base em [36], os principais problemas inerentes à estimação de mapas de profundidade são listados a seguir:

- Ruído: variações na iluminação e ruídos presentes nas imagens produzem diferentes intensidades de cores entre as vistas, fazendo com que a correspondência dos pontos estimada seja incerta.

- Regiões sem textura: em regiões com cor constante, a medida de similaridade calculada pode ser a mesma para todos os valores de disparidade estimados, resultando em valores de disparidade incorretos.
- Descontinuidades de profundidade: no mapa de profundidade, descontinuidades abruptas dos valores de profundidade estão tipicamente associadas com as bordas dos objetos. A determinação da medida de correlação de blocos próximos às descontinuidades possibilita que um bloco contenha porções de objetos com duas profundidades diferentes (*foreground* e *background*), fazendo com que a medida de similaridade seja computada de forma incorreta.
- Oclusão: não é possível determinar a correspondência de pontos de regiões obstruídas.

Uma das aplicações que tem utilizado mapas de profundidade como auxílio para geração de vistas intermediárias é o DIBR.

2.2 *Depth-Image-based Rendering* (DIBR)

Dado M câmeras de entrada, com $M \geq 1$, também chamada de câmeras de referência, uma vista virtual pode ser sintetizada. O DIBR, ou Renderização baseada em Imagem de Profundidade é umas das principais ferramentas utilizadas atualmente para a criação de vistas virtuais. Conceitualmente, o DIBR pode ser dividido em três passos principais:

1. Pré-processamento do mapa de profundidade.
2. Deformação 3D (*3D warping*).
3. Preenchimento de buracos.

O mapa de profundidade pode ser pré-processado (primeiro passo) com o objetivo de reduzir o grande número de desoclusões que são normalmente geradas pela deformação 3D (segundo passo). Além disso, o pré-processamento do mapa de profundidade também ajuda na redução do ruído geralmente presentes nos mapas de profundidade, diminuindo o erro no processo de deformação. Dependendo do algoritmo utilizado, este pré-processamento é capaz de reduzir a complexidade computacional do preenchimento dos buracos (terceiro passo). Como um exemplo, alguns trabalhos utilizam os filtros de Gauss como pré-processamento para remover completamente os buracos gerados pela deformação 3D, tornando a etapa de preenchimento de buracos desnecessária [37]. Esta abordagem, no entanto, tem algumas desvantagens, tais como a criação de distorções

geométricas não-naturais nas vistas virtuais geradas. Outras técnicas propostas incluem: filtro gaussiano assimétrico [38], filtro bilateral [39] [40] [41] e filtros adaptativos [42].

Depois do pré-processamento (ou não) no mapa de profundidade, a vista virtual pode ser criada seguindo os passos seguintes: (1) projetar os pixels da imagem de referência para o plano da imagem virtual, o que é chamado de deformação 3D ou 3D *warping*. Dada uma imagem de referência, seu correspondente mapa de profundidade, e os parâmetros da câmera, o DIBR estipula as posições dos pixels de textura da imagem de referência para o plano de imagem virtual em duas etapas. Primeiro, ele projeta os pixels em 2D no plano da imagem para pontos no sistemas de coordenadas 3D e (2) mesclar os pixels projetados para a mesma posição na vista virtual a partir de diferentes vistas de referência. E por último (3) preencher os buracos gerados (de posições sem pixels projetados) na vista virtual através de padrões de textura de seus pixels vizinhos. A Figura 2.7 ilustra as configurações das câmeras virtuais em uma abordagem básica que pode ser utilizada no DIBR.

Tendo o mapa de profundidade e as câmeras com suas posições definidas (com configuração de uma câmera *pinhole*) e assumindo que os pontos correspondentes de pixels possuem mesmo valor de intensidade (cor) um ponto $\mathbf{P}(x, y, z)$ 3D é projetado nas vistas de referência e virtual e denota-se as posições de projeção desses pixels no plano da imagens nas duas visões como $p_r(u_r, v_r, 1)$ e $p_v(u_v, v_v, 1)$ respectivamente como mostrado na figura 2.8. Assim tem-se duas equações de projeção de perspectiva que deforma o ponto 3D nos sistemas de câmera por:

$$z_r \mathbf{p}_r = \mathbf{A}_r(\mathbf{R}_r \mathbf{P} + \mathbf{t}_r) \tag{2.8}$$

$$z_v \mathbf{p}_v = \mathbf{A}_v(\mathbf{R}_v \mathbf{P} + \mathbf{t}_v) \tag{2.9}$$

nas quais $\mathbf{R}_r$ e $\mathbf{t}_r$ são as matrizes de rotação e translação da câmera respectivamente, sendo o parâmetros extrínsecos da câmera (ver seção 2.1.1), que transformam as coordenadas do ponto 3D em coordenadas da câmera de referência. z_r é o valor de profundidade do pixel indicado pelo mapa de profundidade de entrada e $\mathbf{A}_r$ especifica os parâmetros intrínsecos da câmera de referência como na Equação 2.4. Notação similar é usada para $\mathbf{R}_v$, $\mathbf{t}_v$, z_v e $\mathbf{A}_v$.

A deformação da imagem 3D visa encontrar na vista virtual a posição correspondente de cada pixel da vista de referência, que pode ser realizado em duas etapas [4]. Primeiro, cada pixel de exibição de referência é projetado no mundo 3D, derivando sua posição no sistema de coordenadas, resolvendo a Equação 2.10,

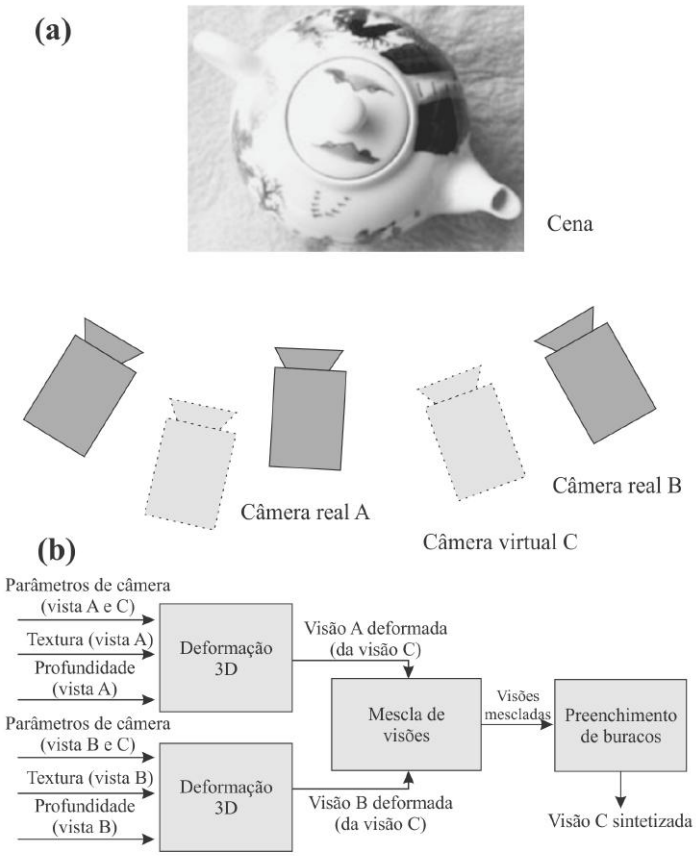


Figura 2.7: (a) Geração de vista virtual com DIBR e (b) Ilustração de um *framework* básico de vista virtual usando duas câmeras de entrada (referência) A e B para sintetizar a vista virtual C. Figura adaptada de [4]

$$\mathbf{P} = \mathbf{R}_r^{-1}(z_r \mathbf{A}_r^{-1}\mathbf{p}_r - \mathbf{t}_r). \tag{2.10}$$

Então, projeta-se o ponto 3D na vista virtual e obtém-se o pixel na posição do sistema de coordenadas da câmera virtual, o que equivale a substituir a Equação 2.10 na Equação 2.9. Consequentemente, tem-se:

$$\mathbf{p}_v = \frac{1}{z_v}\mathbf{A}_v(\mathbf{R}_v\mathbf{R}_r^{-1}(z_r \mathbf{A}_r^{-1}\mathbf{p}_r - \mathbf{t}_r) + \mathbf{t}_v). \tag{2.11}$$

No entanto, um esquema DIBR básico pode não garantir uma visão sintetizada de qualidade perfeita, especialmente quando os erros estão presentes nos mapas de profundidade. Os erros de profundidade fazem com que os pixels associados sejam deformados para posições erradas na vista virtual, produzindo distorções geométricas. As distorções

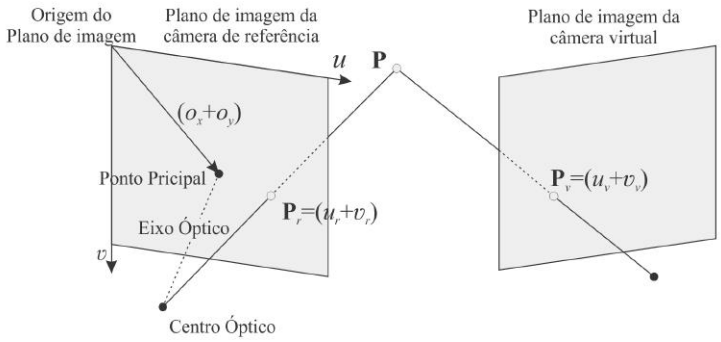


Figura 2.8: Ilustração do um ponto 3D no mundo projetados na vista de referência e vista virtual. Adaptada de [4]

geométricas mais visíveis aparecem nos limites dos objetos devido a dados de profundidade propensos a erros ao longo dessas áreas, mostrados como bordas quebradas e fundo manchado com outros valores de textura de outros objetos. Além disso, algoritmos simples de preenchimento de buracos usando interpolação com os pixels vizinhos podem falhar em recuperar as informações de textura perdidas em buracos, especialmente em regiões altamente texturizadas [4].

Após a etapa de deformação 3D, os pixels vazios na exibição de destino (buracos ou desoclusões) podem ser preenchidos por meio de informações do pixels vizinhos na etapa de preenchimento de buracos. Umas técnicas que abordam este preenchimento de buracos ou regiões são o *Inpainting* e a interpolação de pixels que é discutido na Seção 2.3.

2.3 Interpolação e Inpainting

A interpolação é um processo de predizer valores de certa váriavel (no nosso caso o *pixel*) de interesse em locais não amostrados, a partir de valores medidos nos pontos ao redor das áreas de interesse [43].

Existem várias técnicas que utilizam abordagens distintas de interpolação conservando os aspectos de detalhes da imagem. Os métodos avaliados podem ser classificados ([44]) em três tipos distintos:

- Técnicas Lineares - Utilizam filtros espaciais lineares ([45]) para a interpolação das imagens. Essa filtragem é efetuada utilizando uma vizinhança ou uma operação

pré-definida realizada sobre os pixels da imagem. Os filtros mais comuns são: a interpolação por vizinho mais próximo, bilinear, bicúbica, quadrática e gaussiana;

- Técnicas Não-lineares - Melhoram a qualidade utilizando restrições para cada característica da imagem. Trabalhos buscam preservar as bordas definindo restrições baseadas nas direções encontradas em sua orientação;

- Técnicas de Transformadas - Usam a decomposição de uma imagem em multi-resoluções e aplicam a interpolação em cada nível obtido. Um exemplo disso é o uso da transformada *wavelets* que separa a imagem em vários sub-níveis com componentes decimadas. Desta forma, são adquiridas informações relevantes no domínio das wavelets que não estão aparentes em seu domínio original.

Além dessas técnicas de interpolação dos *pixels* baseada em sua vizinhança, existem técnicas como *inpainting* cujo objetivo principal está em restaurar regiões danificadas ou remover áreas de uma imagem usando informações de áreas não danificadas. Por exemplo, a restauração de rachaduras em fotos, recuperação de oclusão, preenchimento de desoclusões, realizados de forma tal que um observador não seja capaz de perceber que um procedimento de *inpainting* foi realizado.

O problema pode ser declarado como segue. Seja I a imagem original (ou um *frame* em uma sequência de vídeo), que é composta por uma área de origem, indicada por Φ, cujos valores de pixel são conhecidos, e uma parte da imagem denominada área (ou região) de *inpainting*, denotada por Ω, representando a região danificada que deve ser reparada ou a região a ser preenchida. Como mostrado na Figura 2.9, estas não são áreas sobrepostas, ou seja, $I = \Phi \cup \Omega$, e $\delta\Omega$ representa o limite ou fronteira entre as áreas de origem e de *inpainting*.

A área de *inpainting* (Ω) é a parte da imagem que será preenchida e é, geralmente, definida de forma manual pelo usuário. Esta região representa a parte da figura que se deseja completar, e pode representar uma parte da imagem que contem algum tipo de falha de informação, que pode ser a corrupção de uma imagem digital, ou uma fotografia digitalizada que possua algum tipo de falha física, entre outras possibilidades. Também pode-se definir uma área de *inpainting* que cubra um objeto que se deseja remover da imagem original.

Outro item importante nas técnicas de *inpainting* é a fronteira da região ($\delta\Omega$). A fronteira da área de *inpainting* é o conjunto de pontos ao redor da área de *inpainting*, que contém informações importantes para que a técnica de preenchimento tenha sucesso. Em geral, as técnicas de *inpainting* analisam essa fronteira e identificam, de alguma forma, por onde deve começar e qual informação será usada no preenchimento.

Figura 2.9: Ilustração do problema de *inpainting* a partir da perspectiva do *inpainting* puro

O preenchimento da região de *inpainting* começa a ser realizado no sentido da fronteira para o interior da região. Cabe a técnica utilizada definir onde começar e como preencher esta região. Esses fatores determinam a propagação da estrutura da imagem na região de *inpainting*, que impactam na continuidade dos pontos preenchidos em relação aos vizinhos já existentes. De fato, esse ponto de discussão e diferenciação entre as abordagens existentes, é o ponto central de qualquer técnica de *inpainting*. Assim, os mescanismos de *inpainting* desenvolvidos geralmente consideram fortemente estes dois pontos: a ordem de preenchimento e a forma de preenchimento (propagação da informação) da região de *inpainting*.

Para o problema de *inpainting* é essencial entender a diferença entre a estrutura e a textura de uma imagem. A estrutura, pode ser definida com sendo as principais partes - objetos de uma imagem, cuja superfície é homogênea sem quaisquer detalhes. Por sua vez a textura pode ser definida com sendo os detalhes sobre a superfície dos objetos que tornam as imagens mais realistas [46]. Umas das áreas onde são aplicada algoritmos para *inpainting* de imagens é a representação esparsa.

2.4 Representação Esparsa

Representar esparsamente um sinal surgiu como um dos conceitos relativos em uma variedade de aplicações em processamento de sinais, como eliminação de ruído (*denoising*) , extração de característica, amostragem compressiva, restauração de imagens e suas aplicações tem utilizados em várias áreas [47].

Considere um vetor $\alpha \in \mathbb{R}^m$, $\alpha = [\alpha_1, \alpha_2, \cdots, \alpha_m]$. Este sinal é considerado esparso se a maioria de seus elementos forem iguais a zero, isto é, se o suporte $\Lambda(\alpha) = (1 \leq i \leq m | \alpha_i \neq 0)$ é de cardinalidade $L \ll m$ [47]. Portanto, um sinal é dito L-esparso quando possui no máximo L elementos não nulos.

Se um sinal não é esparso, este pode possuir uma representação esparsa em um domínio de transformada apropriado [47]. De fato, esquemas de compressão baseiam-se nesta ideia. Como exemplos, é possível citar o caso da JPEG que utiliza a DCT e a JPEG-2000 que utiliza *wavelets*. Em ambos os casos, utiliza-se destas transformadas para obter uma representação esparsa de um sinal.

Na última década, a noção de esparsidade como forma de modelar sinais gerou grande interesse [48]. De forma mais geral, é possível representar esparsamente um sinal $\mathbf{x} \in \mathbb{R}^n$ sobre um dicionario $\mathbf{D} \in \mathbb{R}^{n \times m}$. Um dicionário corresponde a um conjunto de sinais, cada elemento desse conjunto de sinais corresponde a uma de m colunas referidas como átomos. Este sinal aleatório x pode ser encontrado através de uma combinação linear de algumas colunas (átomos) de um dicionário $\mathbf{D}$, essa combinação linear corresponde a uma multiplicação de matrizes, onde essas matrizes são um dicionário D de m sinais de tamanho n e alguns desses m elementos serão selecionados através da matriz α, Matematicamente o sinal é ser representado por:

$$\mathbf{x} = \mathbf{D}\alpha \tag{2.12}$$

na qual $\|\alpha\|_0 < L$. $\|\cdot\|_0$ representa a norma de regularização l_0 que computa a quantidade de elementos não-nulos no vetor α. O conjunto de índices dessas L colunas selecionadas é chamado de suporte S. Essa modelagem assume que o sinal $\mathbf{x}$ reside em um subespaço de baixa dimensão, que é dado pelos L átomos de $\mathbf{D}$ que correspondem ao suporte de α. A figura 2.10 demonstra visualmente essa representação.

Para um sinal (ou família de sinais) de interesse como imagens, a esparsidade pode ser usado como um modelo descritivo dessas imagens. Isto é, assume-se que cada sinal tem uma representação esparsa sobre um dicionário específico $\mathbf{D}$. Pode-se nomear este modelo como *Sparseland* [49]. O modelo de *Sparseland* é definido pelos seguintes componentes:

- Família de sinais - um sinal ou conjunto de sinais com algumas propriedades (característica de sinais, como objetos e texturas em imagens) que podem ser redundantes. Essa redundância pode ser vantajosa para se encontrar sua representação em um dicionário o mais esparso possível. Por exemplo, o conjunto de imagens faciais claramente leva a uma melhor modelagem se um dicionário estiver contido de informações de um conjunto de imagens faciais, quanto mais próximo as características melhor o modelo *Sparseland* lida com seus membros;

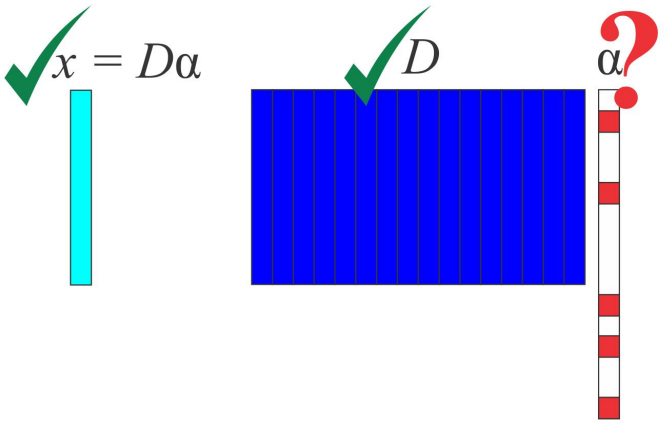


Figura 2.10: Modelo visual para um sinal com representação esparsa.

- Erro de representação - o erro de representação máxima permitido para um sinal ou cada sinal de um conjunto;
- Um dicionário **D** composto m átomos (colunas).

Assim, o sinal **x** é obtido como um problema de minimização. Nesse modelo, o sinal é recuperado computando a representação esparsa $\hat{\alpha}$ pelas equações 2.13, 2.14 ou 2.15:

$$\arg\min_{\alpha} \frac{1}{2}\|\mathbf{D}\alpha - y\|_0{}^0 \text{ tal que } \|\alpha\|_0{}^0 \leq L \tag{2.13}$$

$$\arg\min_{\alpha} \|\alpha\|_0{}^0 \text{ tal que } \frac{1}{2}\|\mathbf{D}\alpha - y\|_0{}^0 \leq \epsilon^2 \tag{2.14}$$

$$\arg\min_{\alpha} \|\alpha\|_0{}^0 + \frac{1}{2}\|\mathbf{D}\alpha - y\|_0{}^0 \tag{2.15}$$

onde y é o sinal original e ϵ um valor de erro admissível entre os sinais originais e o modelado. Esse problema é também conhecido como codificação esparsa (*sparse coding*) [48], porque é procurado uma aproximação por representação esparsa desse sinal. Uma vez obtido $\hat{\alpha}$ o sinal **x** é recuperado multiplicando a representação esparsa resultante pelo dicionário **D**, isto é, $\hat{x} = \mathbf{D}\hat{\alpha}$ e com isso acha-se um sinal $\hat{x}$ que seja o mais semelhante possível de x.

Os problemas de minimização 2.13, 2.14 e 2.15 são difíceis de resolver, devido principalmente à alta quantidade de combinações possíveis de representações e, por isso, é afirmado que esse tipo de problema é considerado NP-difícil [50], e com isso os métodos para a aproximação para esses problemas são propostos. Em termos gerais, esses métodos

podem ser classificados em técnicas baseadas em algoritmos gulosos ou métodos de relaxamento convexo. A primeira baseia sua aproximação na seleção da melhor alternativa local em cada iteração com a intenção de determinar uma solução ótima global. Nesse campo existem os algoritmos de *Orthogonal Matching Pursuit* (OMP) [51], e *Matching Pursuit* [52]. Esses métodos são muito simples de implementar, envolvendo a computação de produtos internos entre o sinal e os átomos do dicionário, e possivelmente implementando alguns solucionadores ou projeções de mínimos quadrados e o nível de esparsidade é medido utilizando a norma l_0

Em métodos de relaxamento convexo, a norma de regularização l_0 é trocada por outra método de norma de regularização l_p[20]. A norma l_p busca estabelecer pesos (penalidade) para todos os elementos de α. Essas técnicas buscam estabelecer regras menos restritivas para a esparsidade do vetor α. Encontra-se o nesse contexto os algoritmos de *Dantzig Selector method* [53] e *Basis Pursuit* (BP) [54] e *Analysis* l_1*-relaxation* [55], onde a norma l_0 e modificada para a norma l_1.

Os modelos apresentado em 2.12 podem assumir que os dicionários são conhecidos e definidos previamente. Esses dicionários são um componente importante para modelagem do sinal. Assim é importante projetar um dicionário que possa encontrar uma representação adequada para o sinal procurado. Existem dois meios comumente usados na literatura para encontrá-los. A primeira baseia-se nas propriedades analíticas dos sinais do modelo e com isso os dicionários são definidos a *priori*. Esta família de dicionários inclui a Transformada Discreta de Fourier (DFT) [56], vários tipos de *wavelets*, tais como *curvelets* [57], *countourlets* [58], e *wavelets* de Gabor [59], entre outros. Em alguns casos, o desempenho desses algoritmos podem ser limitado quando aplicado estes dicionários a *priori* [60]. O sucesso de tais dicionários nas aplicações depende da adequação para descrever os sinais em questão.

Outra abordagem é aplicar técnicas de aprendizagem de máquina e treinar um dicionário, onde o objetivo é encontrar o dicionário que produz uma representação esparsa para os dados de treinamento. Este problema é conhecido como o problema de aprendizagem do dicionário [20].

O problema para o aprendizado do dicionário é dado um conjunto de sinais (imagens) para ter uma representação esparsa sobre um dicionário desconhecido $\mathbf{D}$ e encontrar $\mathbf{D}$ na qual este possa representar um sinal ou um conjunto de sinais de maneira satisfatória.

Várias técnicas foram desenvolvido para resolver este problema e treinar dicionários a partir de dados de treinamento. Entre eles tem-se o Method of Optimal Directions (MOD) [61], e o método K-SVD [18], que generaliza o algoritmo de agrupamento de *K-means*.

2.5 Projeto de Dicionários

O treinamento de dicionários é realizado com base em um conjunto de entradas para treinamento. Dado um conjunto $\mathbf{Y} = \{\mathbf{y_i}\}_{i=1}^{N}$, e considerando o modelo de *Sparseland* descrito na seção 2.4 assumindo que existe um dicionário $\mathbf{D}$ que modela os sinais de entrada fornecidos através de combinações esparsas, ou seja, assumimos que existe $\mathbf{D}$, de modo que a solução em 2.12 para cada exemplo $\mathbf{y}_k$ com $1 \leq k \leq N$ dá uma representação esparsa $\mathbf{x}_k$. O objetivo então é encontrar um único dicionário que possa modelar todo o conjunto $\mathbf{Y}$. A figura 2.11 mostra esse processo:

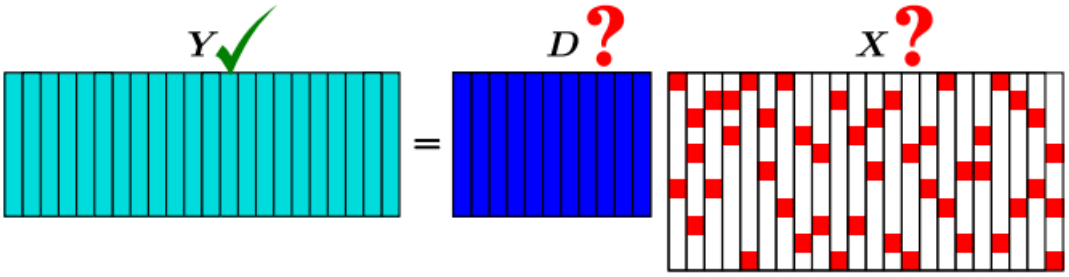


Figura 2.11: Modelo visual para modelar um dicionário

Existe uma relação entre a representação esparsa e o agrupamento ou clusterização (como exemplo, a quantificação vetorial) [62]. Na clusterização, um conjunto de vetores descritivos $\{\mathbf{d_k}\}_{k=1}^{K}$ é aprendido e cada amostra $\mathbf{y}_k$ é representada por um desses vetores (ou mais próximo dele) e não necessariamente é necessário um átomo para representar cada sinal de entrada. Assim, representações esparsas com treinamento de dicionário podem ser referidas como uma generalização do problema de agrupamento.

Quando o dicionário $\mathbf{D}$ for aprendido, determinar com precisão o tamanho M é muito importante podendo ser definido inicialmente assim como o nível de esparsidade. Para resolver problemas onde o tamanho do vetor e esparsidade de α possa ser inferido, análises de fatores com abordagem mais estatística tem sido usado recentemente em aplicações de aprendizagem de dicionário, como a formulação bayesiana não-paramétrica [23] [2].

2.5.1 Análise de fatores

O principal objetivo dessa técnica é reduzir o nível de caracterização de váriaveis de entrada dentro de um sistema encontrando fatores que justifiquem as correlações observadas entre as variáveis. A intenção é substituir o conjunto original de variáveis (em geral grande) e correlacionadas por um conjunto menor de variáveis sem correlação ou com baixa correlação.

Além disso, a análise de fatores é aplicada como redutora de dados ou método (exploratório) de detecção de estruturas. Ao analisar a estrutura das correlações entre um grande número de variáveis, definindo um conjunto menor de dimensões básicas comuns, chamadas fatores. Isso é feito através uso de classe de métodos estatísticos, cujo propósito principal é definir uma estrutura fundamental em um conjunto de dados.

2.5.2 Modelagem estatística

Num termo mais genérico modelar um sinal estatisticamente em um espaço de amostras $\mathbf{X}$, ou seja um conjunto de probabilidades em $\mathbf{X}$. Ao adotar $\mathbf{PM}(\mathbf{X})$ para o espaço de amostras em que todas as probabilidade estão em $\mathbf{X}$, um modelo $M \subset \mathbf{PM}(\mathbf{X})$ pode ser utilizado. Os elementos de M são indexados por um parâmetro Θ com valores em um espaço de parâmetro $\mathbf{T}$, isto é,

$$M = \{P_\Theta | \, \Theta \in \mathbf{T}\} \tag{2.16}$$

onde cada P_Θ é um elemento de $\mathbf{PM}(\mathbf{X})$ e consequentemente em $\mathbf{M}$[63]. Chama-se de modelo paramétrico se $\mathbf{T}$ tiver uma dimensão finita. Se $\mathbf{T}$ tem dimensão infinita, $\mathbf{M}$ é chamado de modelo não paramétrico [5].

Para formular problemas estatísticos, assume-se que n observações $x_1, \cdots, x_n$ com valores em $\mathbf{X}$ são gravadas, e pode-se modelar como variáveis aleatórias $X_1, \cdots, X_n$. Estatisticamente, é assumido que essas variáveis aleatórias são geradas independente e identicamente distribuídas (i.i.d) pelo modelo:

$$X_1, \cdots, X_n \sim_{iid} P_\theta \tag{2.17}$$

para algum θ em $\mathbf{T}$. O objetivo da inferência estatística é então tirar conclusões sobre o valor de θ (e, portanto, sobre a distribuição P_θ) das observações.

Uma intuição útil, é pensar em θ como um padrão que modela o comportamento dos dados [63]. A Figura 2.12 (esquerda) mostra um exemplo simples, um problema de regressão linear. Os pontos são os dados observados, o que mostra uma tendência linear. A linha é o padrão que usado para explicar os dados, neste caso, simplesmente uma função linear e analisando θ como esta função. O espaço de parâmetros $\mathbf{T}$ é, portanto, o conjunto de funções lineares em $\mathbb{R}$. Dado θ, a distribuição P_θ explica como os pontos se espalham pela linha. Uma vez que uma função linear em $\mathbb{R}$ pode ser especificada usando dois escalares, um deslocamento e uma inclinação, $\mathbf{T}$ pode ser expresso como $\mathbb{R}^2$. Comparando com nossas definições acima, vemos que esse modelo de regressão linear é paramétrico.

Agora suponha que a tendência nos dados seja claramente não-linear. Pode-se então usar o conjunto de todas as funções em $\mathbb{R}$ como nosso espaço de parâmetros, em vez de apenas linhas lineares. Claro, normalmente queremos que uma função de regressão seja contínua e razoavelmente suave, então podemos escolher $\mathbf{T}$ como, digamos, o conjunto de todas as funções duas vezes continuamente diferenciáveis em $\mathbb{R}$. Um exemplo de função θ com os dados gerados a partir dele pode parecer a função na Figura 2.12 (direita). O espaço $\mathbf{T}$ agora é infinito-dimensional, o que significa que o modelo é não paramétrico.

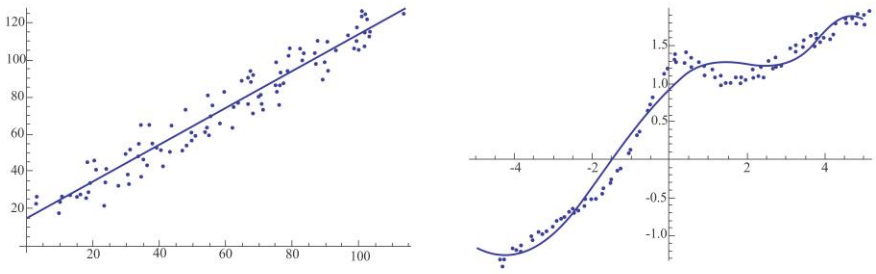


Figura 2.12: Problema de regressão: linear (esquerda) e não-linear (direita). Nos dois casos a função de regressão (em azul) é parâmetro do modelo. [5]

2.6 Modelos Bayesianos e Não-paramétricos Bayesianos

Considerando uma variável aleatória Θ com valores em $\mathbf{T}$. Faz-se uma suposição de modelagem sobre como Θ é distribuído, escolhendo uma distribuição específica Q e assumindo $Q = \mathcal{L}(\Theta)$. A distribuição Q é chamada de distribuição prévia do modelo. Um modelo bayesiano, portanto, consiste em um modelo M como em 2.16, chamado modelo de observação, e um Q anterior. Sob um modelo bayesiano, os dados são gerados em duas etapas, como:

$$\begin{aligned} \theta &\sim Q \\ X_1, X_2, \cdots, |\Theta &\sim_{iid} P_\Theta \end{aligned} \tag{2.18}$$

Isso significa que os dados estão condicionalmente independente e identicamente distribuídas (i.i.d). O objetivo é então determinar a distribuição posterior, a distribuição condicional de θ dada a informação:

$$Q[\Theta \in \bullet | X_1 = x_1, \cdots, X_n = x_n] \tag{2.19}$$

Essa é a contrapartida à estimação de parâmetros na abordagem clássica. O valor do parâmetro permanece incerto dado um número finito de observações, e as estatísticas bayesianas usam a distribuição posterior para expressar essa incerteza[5].

Um modelo bayesiano não paramétrico é um modelo bayesiano cujo espaço de parâmetros tem dimensão infinita. Para definir um modelo Bayesiano não paramétrico, tem que definir uma distribuição de probabilidade (o anterior) em um espaço de dimensões infinitas. Uma distribuição em um espaço de dimensão infinita $\mathbf{T}$ é um processo estocástico com caminhos em $\mathbf{T}$. Tais distribuições geralmente são mais difíceis de definir do que as distribuições em $\mathbb{R}^d$ [5]. Algumas distribuições utilizadas nesse trabalho é o Beta e Bernoulli.

2.7 Funções de distribuição de probabilidades

Uma função de distribuição de probabilidade também chamada função de distribuição, é a probabilidade de uma variável aleatória X assumir valores menores ou iguais a t, onde t é um número real. É representada por $F(t)$, de modo que:

$$F(t) = P(X \leq t) \tag{2.20}$$

Para uma variável aleatória discreta, a função distribuição de probabilidade $F(t)$ é:

$$F(t) = \sum_{t_i \leq t}^{i} p(t_i) = p(t_1) + p(t_2) + p(t_3) + \cdots + p(t) \tag{2.21}$$

Para uma variável aleatória contínua, a função distribuição acumulada $F(t)$ é:

$$F(t) = \int_{-\infty}^{t} f(t)dx \tag{2.22}$$

onde $f(t)$ é chamada função densidade de probabilidade. A função densidade de probabilidade associa os valores de X com a probabilidade de cada um deles ocorrer.

Algumas distribuições partem do pressuposto da existência de certas hipóteses bem definidas. Como diversas situações na vida real se aproximam destas premissas alguns experimentos aleatórios cujos resultados, refletidos em uma variável aleatória, seguem um comportamento previsível em relação às suas probabilidades de ocorrência, e portanto podem ser modelados por uma equação específica. Algumas das existentes e aplicadas nesse trabalho são a função de distribuição Beta e Bernoulli

2.7.1 Função de distribuição Beta

A distribuição Beta é frequentemente usada para modelar a proporção, ou modelagem de objetos que pertencem ao intervalo $(0, 1)$, pois essa distribuição está definida neste intervalo.

A distribuição Beta é uma distribuição de probabilidade contínua, com dois parâmetros α e β cuja função de densidade para valores $0 < x < 1$ é

$$f(x) = \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(b)} x^{\alpha-1}(1-x)^{\beta-1} \tag{2.23}$$

onde $x \in (0, 1)$ e $\alpha, \beta > 0$. No modelo, os parâmetros α e β definem a forma da distribuição como visto pela imagem 2.13. Se $\alpha = \beta$, a distribuição é simétrica, se $\alpha > \beta$, a assimetria é negativa e, no caso de $\alpha < \beta$, sua assimetria é positiva.

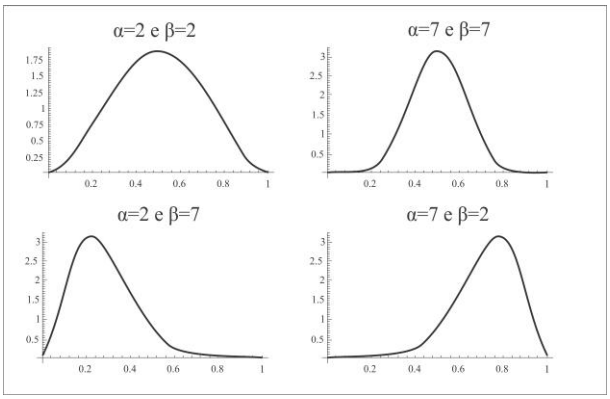


Figura 2.13: Curvas de densidade de distribuição Beta para diversos valores de α e β

2.7.2 Função de distribuição Bernoulli

Esse tipo de distribuição corresponde a qualquer experimento em que se produza dois resultados. Esses resultados podem ser descritos como "0" ou "fracasso " e "1" ou "sucesso". A probabilidade de ocorrência de "sucesso" é representada por p e a de insucesso por $q = 1 - p$. Assim a função de distribuição de probabilidade é dada por:

$$f(x) = \begin{cases} 1 - p = q & \text{se } x = 0 \\ p & \text{se } x = 1 \end{cases} \tag{2.24}$$

Capítulo 3

Revisão da Literatura

A imagem 3D foi iniciada em 1838, quando Charles Wheatstone criou o estereoscópio que é dispositivo baseado numa combinação de prismas e espelhos que permitia ver imagens em 3D a partir de imagens 2D, mostrando que duas imagens visualmente combinadas podem criar a ilusão de profundidade e três dimensões [64]. Em seguida surgiu uma nova maneira de separar o par de imagens estereográficas: o anaglifo. O par de imagens era desenhado usando duas cores (vermelho e azul) e para vê-las em 3D, era usado um par de óculos com filtros coloridos (também azul e vermelho). Os primeiros filmes 3D utilizava esse sistema anaglífico com dois projetores para criar a ilusão de profundidade [65]. Na década de 50, diversos títulos em 3D foram lançados, e a moda chegou também às revistas. Nas décadas a seguir, o cansaço visual e o custo alto das produções causaram uma queda nas produções tridimensionais.

No início dos anos 90, os esforços de pesquisa em 3DTV foram aumentados devido à transição gradual do serviço analógico para o digital. Inspirado pelo interesse em serviços de transmissão 3D o *Motion Picture Expert Group* (MPEG) começou a trabalhar em uma tecnologia de compressão para a seqüência de vídeo estereoscópica que resultou em um *Multi View Profile*(MVP) como parte do padrão MPEG-2 [66]. O cinema 3D também obteve uma boa reputação por uma compreensão mais profunda da percepção de profundidade e análise de imagem. As melhorias nas tecnologias de vídeo 3D aumentaram o interesse em 3DTV e FVV. Enquanto 3DTV fornece a impressão de profundidade sem usar óculos e FVV permite ao usuário escolher o ponto de vista de uma cena em qualquer direção. A experiência de visualização 3D foi gradualmente melhorada pelo suporte de câmeras digitais avançadas e mais compreensão da percepção de profundidade. Para minimizar os custos de obtenção e transmissão de cada ponto de vista é possível sintetizar a vista escolhida pelo usuário.

3.1 Trabalhos em síntese de vistas

Os métodos de renderização baseados em imagens estão tendo um impacto substancial no campo da computação gráfica e também desempenham um papel importante em áreas relacionadas a sistemas multimídia, com aplicações em teleconferência, instrução remota e cirurgia, realidade virtual e entretenimento.

Lippman [67] foi um dos primeiros a propor o uso de uma representação baseada em imagens para visualização de cena 3D computadorizada. Sua abordagem *Movie-Map* permitiu a exploração virtual interativa de um ambiente, acessando seletivamente vistas previamente capturadas e armazenadas em disco. Uma contribuição adicional foi o uso de imagens panorâmicas para capturar um amplo campo de visão. Estas imagens foram corrigidas opticamente no tempo de reprodução utilizando um aparelho de visualização que emprega um espelho cônico. Movendo a cabeça em relação a este dispositivo, o usuário poderia efetuar uma rotação da câmera virtual sobre seu centro óptico. Essa abordagem utiliza uma vista escolhida já existente.

Trabalho como de Kang et. al [68] classificam as várias técnicas de renderização em três categorias chamadas de renderização sem geometria, renderização com geometrias implícita e renderização com geometria explícita. O termo geometria refere-se as informações de localização e parâmetros de câmeras necessárias para a síntese. A Figura 3.1 mostra essa classificação com as técnicas utilizadas para renderizar novas sínteses.

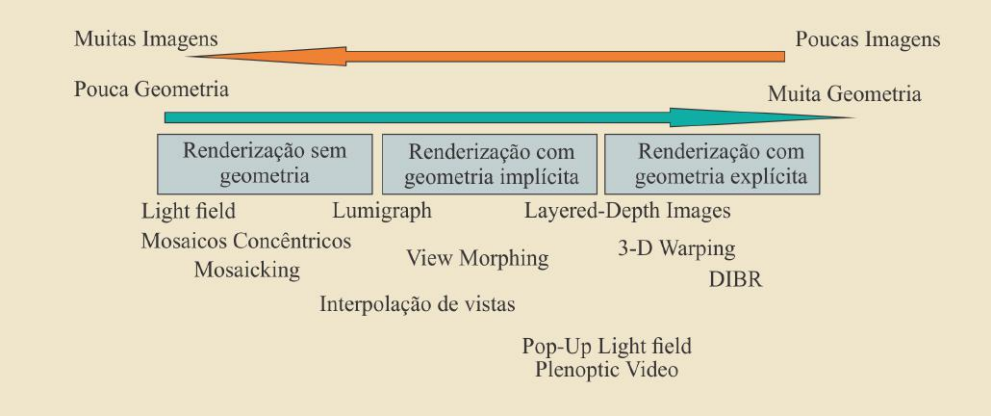


Figura 3.1: Categorias para as técnicas de renderização e seus respectivas classificações de métodos

No primeiro extremo, renderização de *light field* usa muitas imagens, mas não requer nenhuma informação geométrica ou correspondência entre as imagens. Esse tipo de renderização proposta por Levoy e Hanrahan [69] produz uma nova imagem de uma cena filtrando apropriadamente e interpolando um conjunto pré-adquirido de amostras. O *Lumigraph* proposto por Gortler et. al [70] é semelhante à renderização *light field* , mas usa

geometria aproximada para compensar a amostragem não uniforme, a fim de melhorar o desempenho de renderização. Ao contrário do *light field* e *Lumigraph* onde as câmeras são colocadas em uma grade bidimensional a representação de Mosaicos Concêntricos de Shum e He [71] reduz a quantidade de dados capturando uma sequencia de imagens ao longo de um caminho circular. Além disso, ele usa uma forma muito primitiva de um impostor geométrico, cuja distância radial é uma função do ângulo de *panning*. (Um impostor geométrico é basicamente uma forma 3D usada em técnicas de IBR para melhorar a predição de aparência por correção de profundidade. É também conhecido como proxy geométrico).

Como a renderização *light field* não depende de nenhum impostor geométrico, ela tende a se basear em superamostragem para contrariar efeitos de *aliasing* indesejáveis na exibição de saída. A superamostragem significa aquisição de dados mais intensiva, mais armazenamento e maior redundância.

Alguns sistemas de renderização baseados em imagens não exigem modelos geométricos explícitos. Em vez disso, eles requerem correspondência entre imagens. Por exemplo, as técnicas de interpolação de vista de Chen e Willians [72] geram vistas novas interpolando (juntando) o fluxo óptico entre pontos correspondentes. Por outro lado, a técnica de *view morphing* de Seitz e Dier [73] resulta em matrizes de câmeras intermediárias ao longo da linha de dois centros de câmera originais, com base em correspondências de pontos. Técnicas de visão por computador são normalmente utilizadas para gerar tais correspondências.

No lado direito da figura 3.1, o mapeamento de pixels da imagem utiliza modelos geométricos muito exatos das imagens de referência do sistema. Em um sistema de renderização de imagem com base em mapas de profundidade (como 3D *warping* de Mark et. al [74], imagens em camadas de profundidade (*Layered Depth Images* - LDI) de Shade et. al[75], e imagens em camadas de profundidade com árvores *Layered Depth Tree* de Chang et. al [76], o modelo consiste em um conjunto de imagens de uma cena e seus mapas de profundidade associados. O *Pop-Up Light Field* de Wood et.al [77] é outra representação de IBR baseada em geometria que usa imagens e dados de intervalo de varredura *Cyberware*.

A técnica de DIBR (*image-based rendering*), é baseado no trabalho de McMillan [78], sendo um processo de sintetizar uma ou várias vistas virtuais de uma cena a partir de câmeras (imagens) posicionadas imovéis ou em movimento associadas a imagem. Trabalho pioneiros como de Scharstein [79], com pares de imagens com mapas de correspondência podem ser computados com técnicas de visão estéreo (câmeras posicionadas na mesma linha de visão, separadas uma da outra por uma pequena distância), esses mapas de correspondências fornecem informações diretas sobre a profundidade relativa dos pontos

de cena visíveis. Assim, cada mapa constitui uma representação baseada em imagens com a geometria da cena, suas informações podem ser usadas para estimar novas imagens correspondentes a novos pontos de vista. Um problema gerado por essa técnica são as oclusões que existiam na imagem de referência que aparecem na vista virtual.

Conceitualmente, essa nova vista gerada pode ser entendida como um seguinte processo que envolve dois passos: Em primeiro lugar, os pontos de imagem originais são reprojetados para o mundo 3D, utilizando os respectivos dados de profundidade. Posteriormente, esses pontos de espaço 3D são projetados no plano de imagem de uma câmera "virtual" que está localizada na posição de visualização desejada [27]. A concatenação de reprojeção (2D para 3D) e a projeção subseqüente (3D para 2D) é geralmente chamada de deformação de imagem 3D na Computação Gráfica (CG) ou *3D warping*. Em sistemas FTV's proposto por Tanimoto et. al [7], o uso de DIBR pode ser utilizado para gerar vistas sintéticas como no trabalho de Yang et. al [9].

Recentemente, em trabalhos como de Pujades e Devernay [80], o uso do formalismo bayesiano foi introduzido nas técnicas de IBR, baseado no trabalho proposto por Wanner e Goldluecke [81]. Eles fornecem a primeira estrutura bayesiana para a síntese de novas vistas, descrevendo o processo de formação de imagens com um modelo generativo baseado na física e derivando sua estimativa com *Maximum a Posteriori* (MAP).

Método de renderização baseado em probabilidade (PBR) é descrito por Ham et. al [82] para reconstruir uma visão intermediária virtual com uma função de densidade de probabilidade de correspondência de estado estacionário (*Steady-state matching probability* - SSMP). O problema das desoclusões é tratado formulando o processo de renderização como uma fusão de imagem em que as texturas de todos os pontos de correspondência prováveis são adaptadas misturadas com o SSMP que representa a probabilidade que os pontos entre as imagens de referência de entrada são combinados.

Em relação a síntese de vista a maior parte mostra a vista sintetizada considerando um movimento no eixo x e esse tipo de síntese geralmente apresentam buracos que são chamados de desoclusões, elas se caracterizam por serem pixels que não estão presentes nas imagens de referência e ao gerar a vista sintetizada elas aparecem. Para uma melhor visualização é necessário o preenchimento desse tipo de buraco.

3.2 Trabalhos em interpolação de desoclusões

As desoclusões são um dos problemas gerados nas vistas sintetizadas quando reconstruídas a partir do DIBR. Esses ocorrem nas bordas dos objetos causados pelas propriedades de deformação na vista virtual. O problema de desoclusão pode ser considerado como falta de informação na textura e é recuperado usando técnicas de *inpainting* [83]. *Inpainting* é

um processo de preencher as informações ausentes usando suas informações de vizinhança de *pixel* disponíveis.

A noção de Inpainting digital foi introduzido pela primeira vez por Bertalmio et al [84]. Utilizando equação diferencial parcial (*partial differential equations* - PDE) de ordem superior (como Laplaciano da imagem) para fins de restauração. O algoritmo propõe que a direção do gradiente dê a direção da região para preencher.

Inpainting por textura busca preencher as regiões vazias, replicando os padrões repetitivos em uma imagem. Para preencher grandes regiões desaparecidas Sung et al [85] propôs uma abordagem com a ajuda de informações globais a partir de imagens múltiplas. Anat Levin et.al [86] usou imagens de treinamento para construir a distribuição exponencial da família sobre as imagens que serão usadas para pintar as regiões que faltam. Outra abordagem sugerida por Ruzic et. al [87] e Efros et. al [88] usam pequenas quantidades de blocos de texturas para modelar campos randômicos de Markov (*Markov Random Field* - MRF) e gerar as novas texturas por amostragem e copiando texturas de regiões vizinhas. *Inpainting* por textura consegue melhor resultados para imagens de textura homogêneas, mas encontram dificuldades para preencher cenas do mundo real. As imagens reais consistem em texturas de objetos com mistura de diferentes texturas e estruturas lineares.

Para resolver esses problemas, os métodos de inpainting por estrutura propagam as estruturas lineares (como valores de cor) em regiões ausentes através de um processo de difusão como nos trabalhos de Bertalmio et.al em [84] e Criminisi e Toyama em [89] que tem servido como base para desenvolvimento de várias pesquisas na área. A desvantagem associada a estes métodos é que não é capaz de recuperar grandes regiões texturizadas devido ao processo que pode perder informaçãoes sobre as bordas de objetos na imagem cujo resultado é um desfoque na imagem pintada.

Algoritmos de *inpainting* atuais buscam integrar essa dois métodos para preenchimento de regiões em imagens.

O algoritmo proposto por Fehn em [27] propõe que tendo a imagem de profundidade, este passa a dar maior prioridade aos pixels do fundo da imagem de referência(*background*). Já Gautier et. al em [90] realiza cálculo da prioridade dos pixels através da exploração da informação de profundidade, em primeiro lugar, definindo um tensor 3D, e depois restringindo o lado por onde começar a interpolação. O tensor 3D permite a difusão da estrutura, não só ao longo da cor, mas também da informação de profundidade. Por sua vez, a restrição do lado de ínicio da interpolação pode ser definida observando o movimento horizontal da câmera, verificando se o mesmo foi para a direita ou esquerda.

No trabalho de Jawas e Suciati em [91] apresenta uma técnica de interpolação baseada em operações morfológicas de dilatação e erosão, que através da convolução de uma

mascára da imagem com um elemento estruturante realiza o preenchimento das regiões de interpolação. A técnica proposta produziu bons resultados, mas identificou uma falha do algoritmo em danos horizontais.

Mais recentemente, Macchiavello et. al em [92] propuseram um algoritmo de preenchimento de desoclusões que inclui o valor da saliência no processo de busca do melhor bloco para síntese da textura. O algoritmo baseado em Bertalmio et. al em [84] seleciona o bloco que possui ambos a menor distância e a menor saliência. Tendo em vista que regiões do fundo possuem menor saliência, os pixels do *background* foram priorizados e o processo de *inpainting* obteve bom resultados. Silva et.al [93] utilizam o mapa de profundidade no cálculo de prioridade e casamento de blocos, com a combinação linear de blocos, os buracos são preenchidos de tal forma que a estrutura da imagem como um todo é preservada, trazendo consigo um maior conforto visual. O algoritmo apresenta duas novidades com relação aos algoritmos presentes na literatura, o uso do termo de relevância e o uso da combinação linear de blocos.

3.3 Trabalhos em preenchimento de buracos de expansão

Os artigos anteriores citados tratam do problema de preenchimento de desoclusões provenientes de síntese de vista virtual com deslocamento de câmera na horizontal (eixo x). Poucos artigos têm tratado do uso de interpolação para os buracos de expansão. Os artigos de Cheung et. al em [14], Mao et. al em [15] e Mao et. al [16] são pioneiros pois implementam, a síntese no eixo z e ainda propõe uma metodologia para identificar e diferenciar as desoclusões e buracos de expansão baseada no histograma dos blocos que compõe o mapa de profundidade da imagem virtual.

Em [14] propõe dois algoritmos para preenchimento de buracos de expansão. A primeira consiste em interpolação linear. Para cada pixel vazio identificado como buraco de expansão, busca-se os três pixels sintetizados mais próximos i, j, k e constrói um plano linear que conecta seus valores de textura, $t(x_i, y_i)$, $t(x_j, y_j)$, $t(x_k, y_k)$ dadas suas coordenadas (x_i, y_i), (x_j, y_j), (x_k, y_k). O pixel vazio é interpolado usando o plano construído e sua própria coordenada de pixel. A vantagem deste método é que é simples e tem baixa complexidade computacional. A segunda mais complexa utiliza Transformada baseada em grafos (GBT - *Graph Based Transform*) para interpolação, dividindo a imagem em blocos e para cada bloco montar um grafo G onde cada aresta com pixels i e j conectados tem um peso $e_{i,j}$ e matrizes auxiliares, o objetivo dessa abordagem é encontrar o valor do pixel identificado minimizando o peso entre o bloco o qual pertence o buraco e a imagem original.

Em [15], a interpolação é baseada no trabalho explicado anteriormente que utiliza somente a informação do bloco vigent e ao incorporar informações não-locais similares aos blocos para construção dos grafos, obteve resultados melhores.

Em [16] propõe selecionar transformadas de Fourier por grafos apropriadas (*Graph Fourier transforms*) adaptativas as estruturas de sinal únicas dos blocos de pixels vigentes identificados como buracos de expansão. O algoritmo consiste de dois passos. Primeiro, usando o tensor estrutural, calcular um kernel adaptativo centrado em um pixel vazio alvo para identificar pixels adjacentes adequados para a construção de um grafo esparso. Em segundo lugar, dado o grafo construído com pesos de aresta ajustados, para preencher o pixel alvo formula-se um problema de programação quadrática iterativo (com uma solução de forma fechada em cada iteração) usando uma suavidade anterior no domínio GFT.

Por último, o trabalho recente de Mao et. al [17]. Para preencher os buracos de expansão é formulado um problema *maximum a posteriori* (MAP) baseados em blocos da imagem. Ao utilizar mais um vez processamento de sinais com grafos foi projetado um algoritmo *fast iterative reweighted least square (IRLS)* para resolver o problema.

3.4 Trabalhos em Representação esparsa

Como dito na seção 2.4, a metodologia para modelar sinais com representação esparsa concentra-se em dois problemas principais: (i)algoritmos de busca, aqui entram o *Pursuit Algorithms* e ainda métodos que utilizem estratégia gulosa e (ii) métodos de composição de dicionário. As abordagens explanadas a seguir aborda alguns algoritmos utilizados para *inpainting* em imagens.

A abordagem de busca *Basis Pursuit* (BP), desenvolvida por Chen et. al [54] sugerem um método onde um critério global é utilizado de forma a forçar a esparsidade dos coeficientes da representação do sinal em um dicionário, substituindo a norma l^0 para a norma l^1. O *Focal Under-determined System Solver* (FOCUSS) é muito semelhante, essa proposta de Gorodnitsky e Rao usam normas l^p com $p \leq 1$, como substituição a norma l^0 [94].

Os algoritmos com estratégia gulosa mais robustos são *Matching Pursuit* (MP), desenvolvido por Pati et. al [51] e *Orthogonal Matching Pursuit* (OMP) de Malalt e Zhang [52]. Estes são algoritmos gulosos que selecionam os átomos do dicionário sequencialmente. Esses métodos são muito simples, envolvendo a computação de produtos internos entre o sinal e os átomos do dicionário. Uma explicação mais detalhada desse algoritmo, que também é usado nesse trabalho para comparação de resultados é explicado na sessão 3.4.2.

Algoritmos com estrátegia de dicionário se baseiam em basicamente dois estágios: Codificação esparsa seguida de atualização dos átomos de dicionário escolhido. Nessa categoria o algoritmo mais usado é o K-SVD.

3.4.1 K-SVD

Este algoritmo cria um dicionário $\mathbf{D}$ que leva a representações esparsas para o dado conjunto de sinais de treinamento. O K-SVD é uma generalização direta do algoritmo *K-Means*, usado para quantização vetorial [18]

Mencionando brevemente o processo *K-Means* aplica dois passos por cada iteração: (i) cada $\{\mathbf{d_k}\}_{k=1}^{K} \in \mathbf{D}$, será atribuído a um ou vários exemplos de treinamento pelo método de vizinho mais próximo; E (ii) dada essa conjunto selecionado, atualizar $\{\mathbf{d_k}\}_{k=1}^{K}$ para melhor se encaixar nos exemplos.

As abordagens para o prejeto do dicionário que foram testadas até agora estão muito em linha com o processo de duas etapas descrito acima. O primeiro passo encontra os coeficientes dados o dicionário - um passo que devemos referir como "codificação esparsa". Então, o dicionário é atualizado assumindo coeficientes conhecidos e fixos. As diferenças entre os vários algoritmos que foram propostos estão no método utilizado para o cálculo de coeficientes e no procedimento usado para modificar o dicionário.

3.4.2 OMP

Orthogonal matching pursuit (OMP)[51] é um algoritmo de estratégia gulosa realizada por etapas. Em cada etapa, este método seleciona o elemento de dicionário com menor erro residual. Após cada seleção, os coeficientes de representação são escolhidos encontrados através de função de mínimos quadrados. Formalmente, dado um sinal $y \in R$, e um dicionário $\mathbf{D}$ com K colunas, qualquer $\{\boldsymbol{d}_k\}_{k=1}^{K}$, é escolhido. O processo inicializa com $r_0 = y$ como o erro residual, $k = 1$ e realiza os seguintes passos:

- Selecione o índice do próximo elemento do dicionário $i_k = argmax_w \left|\langle \mathbf{r}_{k-1}, \mathbf{d}_w \rangle\right|$
- Atualizar a aproximação $y_k = argmax_{y_k} \left\|\mathbf{y} - \mathbf{y_k}\right\|_2^2$ tal que $\mathbf{y}_k \in \{\mathbf{d}_{i1}, \mathbf{d}_{i2}, \cdots, \mathbf{d}_{ik}\}$ e ;
- Atualizar o erro residual $\mathbf{r}_k = \mathbf{y} - \mathbf{y}_k$

O algoritmo pode ser interrompido após um número predeterminado de etapas, portanto, depois de ter selecionado um número fixo de átomos. Alternativamente, a regra de parada pode ser baseada na norma do residual, ou no produto interno máximo calculado na próxima etapa de seleção de átomos. Esse algoritmo pode ser programado para fornecer uma representação com um número fixo a priori de entradas não-zero. [49]

Capítulo 4

Metodologia

O trabalho tem como objetivo a construção de um sistema de síntese de vistas como um deslocamento proporcional no eixo z, ou seja um movimento de aproximação ou distanciamento em relação a câmera. Além disso preencher os buracos de expansão resultantes com técnicas de interpolação e *inpainting* por representação esparsa previamente explicados.

A abordagem é inspirada na técnica de síntese baseada no DIBR. Uma maneira simples de visualizar a estrutura é considerar quatro grandes tarefas: a aquisição de dados, geração de vista virtual, posteriormente identificar os pixels que são desoclusões dos buracos de expansão, aplicar as técnicas de *inpainting* somente nos pixels considerados como buracos de expansão e por fim avaliar a qualidade de interpolação calculando somente o PSNR dos *pixels* identificados como provenientes da síntese no eixo z. O diagrama da Figura 4.1 mostra os passos propostos para a metodologia. As próximas seções irão descrever os mecanismos envolvidos em cada uma dessas tarefas.

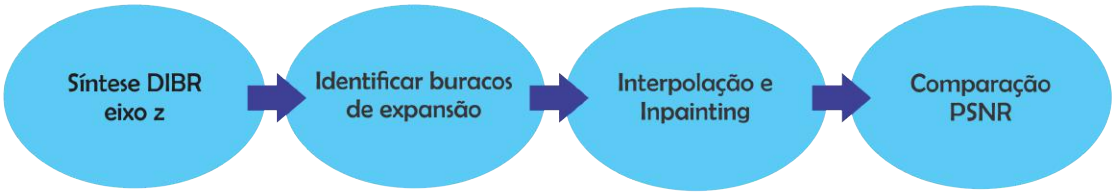


Figura 4.1: Diagrama das etapas

4.1 Gerando a vista virtual

O DIBR pode gerar uma vista sintética através de duas imagens, na qual a vista intermediária é obtida deslocando cada umas das imagens de referência de acordo com um parâmetro de deslocamento gerando uma imagem que seja um "*matching*" das imagens deslocadas. Essa abordagem é usada para deslocamentos horizontais ou verticais da câmera, sem a necessidade de aproximação ou distanciamento em relação ao dispositivo de captura.

A síntese no eixo z é realizada com somente uma imagem de referência, essa síntese simula um processo de *zoom in* e *zoom out* comuns as câmeras de hoje em dia.

Quando o ponto de vista virtual está localizado mais perto da cena do que a vista de referência (*zoom in*), os objetos mais próximos da câmera vão aumentar de tamanho na vista virtual mais rápido do que objetos mais distantes. Um grande aumento no tamanho do objeto significa que uma parte dos *pixels* amostrados a partir de uma superfície do objeto na vista de referência serão espalhados a uma maior área espacial, o que resulta em buracos de expansão. Para complicar ainda mais, no meio desses *pixels* dispersos e seu consequente aumento de área podem obstruir outros objetos, também gerando as desoclusões.

Usamos a mesma metodologia de um trabalho anterior [14], usando como *ground truth*, a vista de referência fornecida chamada de v_0. Essa representação inicial desse posicionamento da câmera em relação a imagem pode ser visto pela Figura 4.2. Em seguida, usa-se o DIBR para gerar a visualização v_r que é mais distante da câmera do que v_0, conforme a Figura 4.3. Nesta vista não haverá buracos de expansão. Usando mapas de textura e profundidade de v_r, aplica-se DIBR novamente para gerar a vista virtual na mesma posição que v_0, este processo é semelhante a um processo de *zoom-out / zoom-in* em câmeras digitais e seu posicionamento final pode ser visto na Figura 4.4.

Nos experimentos foi considerado uma redução da distância entre a câmera e o objeto mais próximo pela metade, o que significa que a resolução espacial do objeto mais próximo pode ser aumentada em 2x. Utiliza-se essa razão para podermos comparar os resultados deste trabalho com os métodos propostos anteriormente [14] [15] [16].

Usando a imagem de textura e seu respectivo mapa de profundidade é possível gerar essas imagens intermediárias. O parâmetro que determina a distância entre a câmera e a cena e necessário para a síntese é denominado Δz, esse valor varia de acordo com o mapa de profundidade de cada cena. O valor de referência de Δz para cada cena é calculado como na Equação 4.1 :

$$\Delta z_{max} = 1/max(di()) \tag{4.1}$$

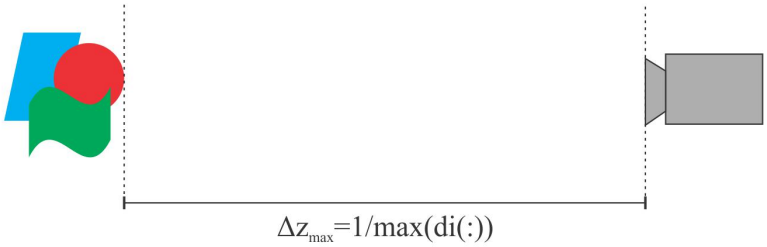


Figura 4.2: Visão lateral da câmera v_0 (vista de referência) posicionada em relação a imagem

Figura 4.3: Visão lateral da câmera v_0 (vista de referência) deslocada para direita (*zoom-out* para a vista virtual v_r)

Figura 4.4: Visão lateral da câmera v_r (vista de referência) deslocada para esquerda (*zoom-in* para a vista virtual v_0)

na qual $max(di())$ é a maior profundidade da imagem, ou seja a distância do objeto mais próximo da câmera. Esse valor é utilizado pois o máximo de deslocamento que uma câmera pode realizar para gerar vistas representativas é justamente no limite do objeto mais próximo.

Logo, pode ser escolhido um valor Δz, no intervalo $[-\Delta z_{max}, \Delta z_{max}]$. Se o objeto mais próximo da cena tiver uma valor de profundidade 200, então sua distância será de 0.005, assim os parâmetros de Δz para a síntese vão variar entre -0.005 e 0.005. Valores negativos representar um *zoom in* ou aproximação da câmera a cena e valores positivos representam um *zoom out* ou distanciamento da câmera a cena.

Assim, dado um pixel p_r da imagem de referência com posição (x, y), ele será mapeado para um novo pixel p_v na posição (x', y') na vista virtual seguindo a seguinte equação:

$$c(p_v) = round([x, y] - [w/2, h/2] * Z_{p_r}/Z_{p_v} + [w/2, h/2]) \tag{4.2}$$

na qual $c(p_v)$ são as coordenadas do pixel (x, y) original na imagem sintetizada na posição (x', y') e w e h correspondem a largura e altura da imagem respectivamente, Z_{p_r} e Z_{p_v} são obtidos pelas seguintes equações:

$$Z_{p_r} = 1/depth(p_r) \tag{4.3}$$

$$Z_{p_v} = Z_{p_r} + \Delta z \tag{4.4}$$

Assim a nova posição é uma proporção entre a distância do pixel na posição de câmera original com a distância do pixel com o deslocamento de câmera Δz. É necessário verificar se a novas coordenadas estão dentro dos limites da resolução da imagem original.

Depois, é necessário identificar os *pixels* da síntese como desoclusão ou buraco de expansão. Para isso, baseado em [14], foi implementada uma técnica de identificação que será explicada na próxima seção.

4.2 Identificação de Buracos de Expansão

Denota-se (x, y) a coordenada do pixel na imagem de referência e $t(x, y)$ e $d(x, y)$ como o valor de textura e de profundidade da coordenada, respectivamente. Quando é renderizado a vista da imagem de referência para a virtual a função de renderização $\mathcal{F}(x, y) = (x', y')$ mapeia (x, y) na imagem de referência para (x', y') na vista virtual. A função inversa de mapeamento $\mathcal{F}'(x', y') = (x, y)$ mapeia (x', y') na vista virtual para (x, y) na imagem de referência. Esses mapeamentos são realizados para a imagem de textura e mapa de profundidade de referência.

O mapa de profundidade da vista sintetizada é dividido em blocos que não sobrepostos com dimensão $B \times B$. Para um dado bloco, divide-se em camadas de acordo com a intensidade de profundidade, sendo os pixels com intensidade 0 são considerados da camada p, como se segue:

1. Construir um histograma do valores de *pixels* sintetizados de cada bloco;
2. Separar os pixels de intensidade em k camadas representado cada intensidade do histograma, onde a primeira camada são os pixels que estão mais próximos da câmera;
3. Processar cada camada de acordo com seu valor de profundidade.

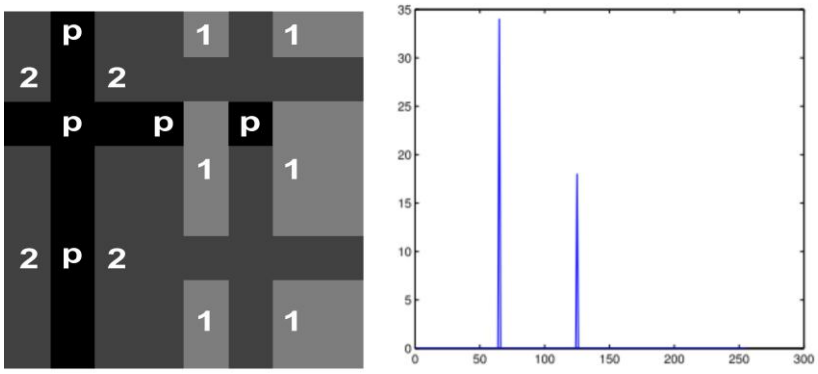


Figura 4.5: Um bloco do mapa de profundidade separado por camadas 1 e 2 de intensidades e p de pixels vazios e seu histograma

No processamento de uma camada l_k todos os *pixels* de uma camada l_{k+1} serão tratados como pixels vazios (além dos buracos de expansão). A Figura 4.6 mostra essa configuração. Isso oferece uma oportunidade para determinar se um pixel sintetizado de *background* também faz parte do conjunto de buracos de expansão.

Cada *pixel* vazio p do bloco é tratado da seguinte maneira. Como mostrado na Figura 4.7, divide-se a vizinhança de cada pixel vazio (marcado como p na Figura 4.7) em quatro quadrantes. Em cada quadrante, o pixel sintetizado s mais próximo (não-vazio) é encontrado. A distância entre os *pixels*, usada para encontrar o pixel mais próximo, é medida pela seguinte equação:

$$H((x_p, y_p), (x_s, y_s)) = |x_p - x_s| + |y_p - y_s| \tag{4.5}$$

O par do pixel vazio e seu pixel mais próximo em cada quadrante são referidos como pares vizinhos. É possível que não exista *pixels* sintetizados não-vazios em um determinado quadrante.

A função de mapeamento inversa $\mathcal{F}'$ é usada, e se $H(\mathcal{F}'(x'_p, y'_p), \mathcal{F}'(x'_s, y'_s)) < n$, onde n é um limiar de distância predefinido, o par de *pixels* é considerado próximo ao pixel na vista de referência. Se dois ou mais pares vizinhos estiverem próximos de *pixels* na

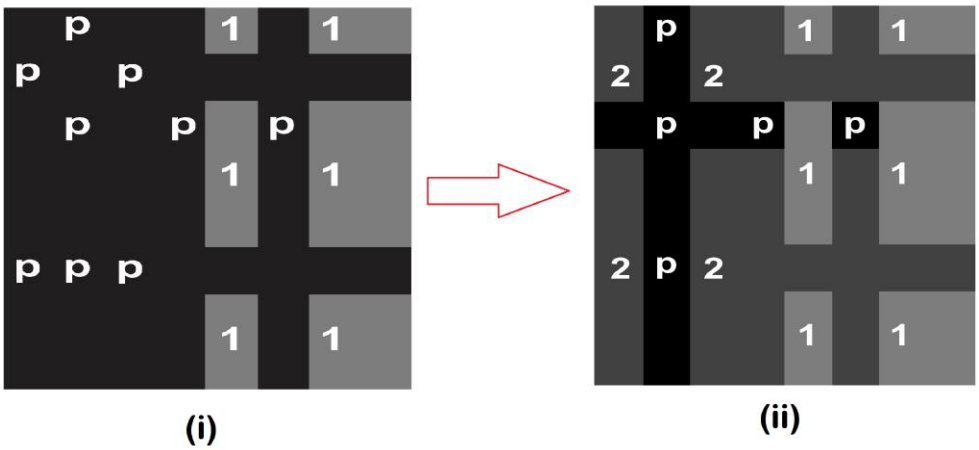


Figura 4.6: A camada rotulada como 1 em (i) considera todos os pixels da camada 2 e p como pixels vazios e a camada 2 em (ii) considera somente os pixels marcados como p como vazios.

vista de referência, é declarado que este pixel vazio faz parte do conjuntos de buracos de expansão. Isso é feito recursivamente para cada camada.

O intuito por trás desse método é verificar se *pixels* próximos na vista virtual são pixels de proximidade na vista de referência. O passo posterior então é preencher somente os buracos de expansão identificados com técnicas de interpolação e inpainting.

4.3 *Inpainting* por Representação esparsa

4.3.1 Formulação do processo Beta-Bernoulli

Nós queremos modelar $\hat{\mathbf{x}} = \mathbf{D}b$ e queremos aprender $\mathbf{D}$ e impor que b seja esparso. Para isso, é considerado um dicionário $\mathbf{D} \in \mathbb{R}^{n \times K}$, com $K \to \infty$ e um vetor binário $b \in \{0,1\}^K$ para indicar qual das colunas K de $\mathbf{D}$ devem ser usadas para a representação $\hat{\mathbf{x}}$ de $\mathbf{x}$. Se um componente específico de b for igual a um, então a coluna correspondente de $\mathbf{D}$ é usada na representação de $\mathbf{x}$. Também queremos que b seja esparso, onde uma pequena fração das colunas de $\mathbf{D}$ são usadas para a representação de um dado $\mathbf{x}$.

Especificamente, suponha que temos um conjunto de treinamento, como um dicionário inicial, $\mathbf{D} = [d_1, d_2, \cdots, d_K]$ para representar $\mathbf{x}_i = \mathbf{D}b_i$, e suponha também que cada coluna deste dicionário está associado a vetores binários $\{b_i\}$ e o processo Beta-Bernoulli fornece uma forma conveniente de encontrar um dicionário apropriado usando esses vetores. Este processo foi desenvolvido em [95], e pode ser representado pelas seguintes equações:

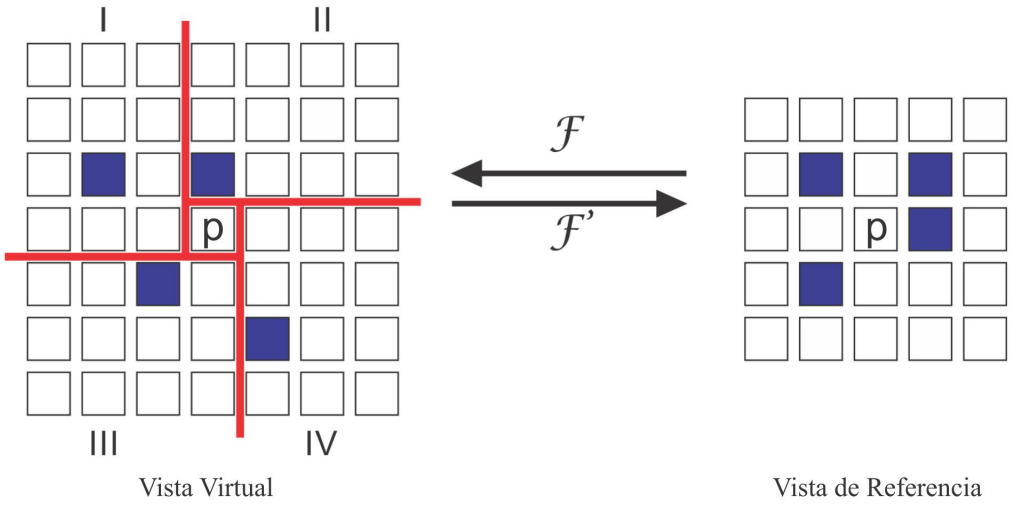


Figura 4.7: Mapeamento do pixel p considerado vazio da vista virtual para referência para consideração de proximidade para detecção de buraco de expansão

$$
\begin{aligned}
H(d) &= \sum_{k=1}^{K} \pi_k \delta_{d_k}(d) \\
\pi_k &\sim B(a/K, b(K-1)/K) \\
d_k &\sim H_0
\end{aligned}
\tag{4.6}
$$

na qual$H(d)$ representa um vetor de probabilidades π_k, com cada componente associado a um respectivo átomo d_k, no limite $K \to \infty$, $H(d)$ corresponde a um vetor infinito de probabilidades. A expressão $\delta_{d_k}(d)$ é igual a 1 se $d = d_k$ e é zero se o valor for diferente de $d \neq d_k$. $B()$ é a função distribuição de probabilidades Beta que associa a probabilidade a um evento de escolha de átomo do dicionário. Os parâmetros $a > 0$ e $b > 0$ determinam a forma da distribuição. H_0 é uma base inicial.

Usando $H(d)$, podemos agora determinar N vetores binários b_i e o componente k-ésimo de b_i é dado por $b_{ik} \sim Be(\pi_k)$. Onde $Be()$ é a distribuição de probabilidade de Bernoulli. Estes N vetores binários são usados para constituir uma matriz $\mathbf{Z} \in \{0,1\}^{K \times N}$, com i-ésima coluna correspondente a b_i e a linha k-ésima associada ao átomo d_k. Neste problema os átomos corresponderão aos membros candidatos para constituir o dicionário $\mathbf{D}$ e o vetor binário b_i define quais membros do dicionário são usados para representar a amostra $\mathbf{x}_i$.

O problema que existe é a imposição de que b_i deve ser binário [23], porém isso é muito restritivo na hora de realizar a combinação dos átomos. Pode ser desejável que os

átomos selecionados sejam ponderados por um determinado peso. Assim, o vetor esparso será agora $\alpha_i = b_i \circ w_i$, onde $\circ$ representa a multiplicação elemento por elemento de dois vetores e assim o sinal é $\mathbf{x}_i = \mathbf{D}\alpha_i$. Onde w_i são pesos de 0 a 1.

A base H_0 é usada para projetar os elementos do dicionário, definidos pelos átomos d_k. Os elementos atuais e posteriores do modelo podem ser gerados via análise de amostragem de Gibbs. Depois de realizar essa inferência, mantemos aquelas colunas d que são usadas na representação dos dados de treinamento, criando um dicionário $\mathbf{D}$ com tamanho $M < K$. Para garantir que α é esparso, ajuste nos parâmetros a e b pode ser feita.

4.3.2 *Inpainting* dos buracos de expansão

Suponha que seja dada uma imagem $\mathbf{I} \in \mathbb{R}^{N_x \times N_y}$ com pixels ausentes (*buracos de expansão*). Como é feito em [21], partimos a imagem em $N_B = (N_x - B + 1) \times (N_y - B + 1)$ blocos $\{\mathbf{x}_i\}_{i=1,N_B}$ para cada um dos quais $\mathbf{x}_i \in \mathbb{R}^{B^2}$ $(B = 8)$. Para aplicar a técnica de interpolação, em vez de amostrar $\mathbf{x}_i$ para montar o dicionário, observa-se um subconjunto dos pixels que não pertecem a nenhum buraco. Observe que $\mathbf{D}$ e $\{\alpha_i\}_{i=1,N_B}$ são usados para preencher os buracos de expansão são inferidos diretamente da vista sintetizada.

O nível de esparsidade é influenciado pelos parâmetros a e b descritos na Equação 4.6. Examinando o posterior $p(\pi_k|-) \sim Beta(a/K + \sum_{i=1}^{N} z_{ik}, b(K-1)/K + N - \sum_{i=1}^{N} z_{ik})$, condicionada a todos os outros parâmetros, foi encontrado que a maioria dos ajustes de a e b tendem a não ser informativos. Portanto, o nível médio de esparsidade da representação é inferido pelos próprios dados e cada amostra $\mathbf{x}_i$ tem sua própria representação esparsa, o que torna muito mais flexível do que aplicar o mesmo nível de esparsidade para cada amostra [23].

Este algoritmo de representação esparsa já foi utilizado para preenchimento de imagens, durante processo de restauração de imagens [23]. Neste trabalho, modificados esta técnica para aplicar em buracos de expansão. Logo, não somente a aplicação é nova, mas também como a técnica é aplicada. O algoritmo descrito é executado recursivamente em diferentes resoluções. Isso não só reduzirá o tempo de execução, mas principalmente melhorará o processo de preenchimento. A imagem com a resolução original é reduzida pela metade recursivamente, realizando um mapeamento das posições de pixels. No primeiro nível L_1 (com a resolução mais baixa), o aprendizado de dicionário bayesiano não paramétrico anteriormente descrito é usado para interpolar os buracos de expansão presentes nesta imagem. Em seguida, um mapeamento é realizado para replicar os pixels de preenchimento na imagem do nível acima L_2. Os buracos que ainda existem neste nível são então preenchidos usando o mesmo algoritmo. Isso é executado até que o último nível L_k (resolução original) esteja com todos os buracos de expansão preenchidos. Desta forma,

a área dos buracos a serem preenchida é sempre pequena, reduzindo o erro entre $\hat{\mathbf{x}}_i$ e $\mathbf{x}_i$. A Figura 4.8 retrata todo esse processo.

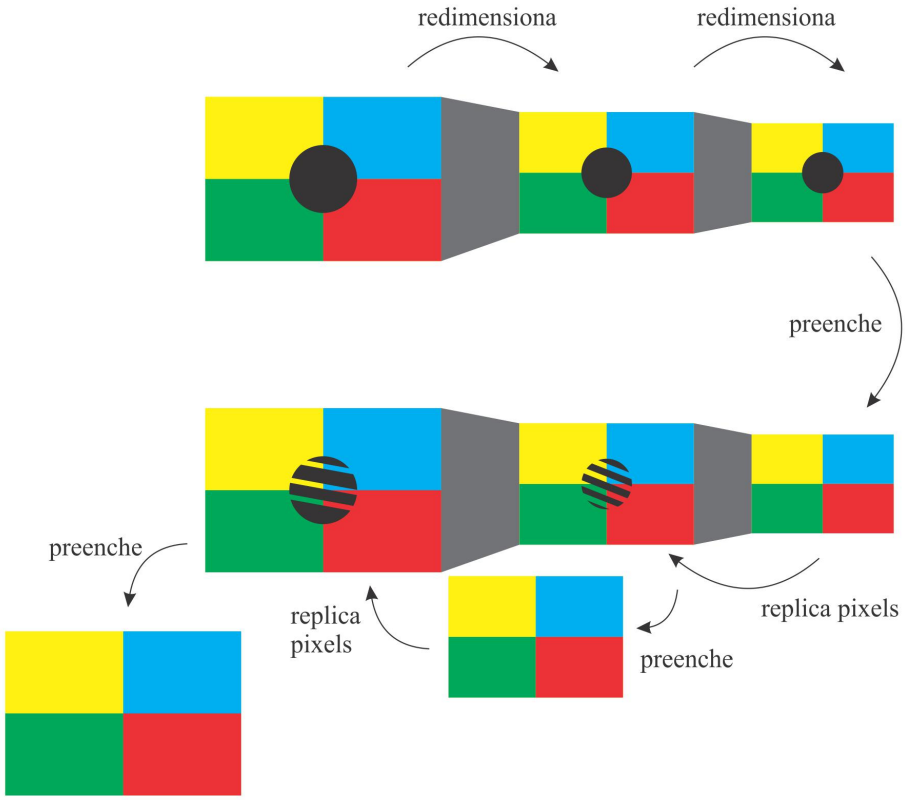


Figura 4.8: Passos para redimensionamento e preenchimento recursivo dos buracos de expansão

Com os buracos de expansão identificados, utiliza-se o seguinte procedimento que é mostrado na Figura 4.9. Considere qualquer pixel (p, j) onde $p, j \in [1, B]$, e utilize esse pixel constituir o pixel esquerdo-superior em um novo bloco $B \times B$. Então considera-se todos os blocos $B \times B$ com pixels esquerdo-superior nas posições $(p, j + B) \cup (p + B, j)$ respeitando as dimensões da imagem. Este conjunto de blocos é chamado conjunto de dados D_{pj}, considerando $1 \leqslant p \leqslant B$ e $1 \leqslant j \leqslant B$. Na primeira iteração de aprendizado de $\mathbf{D}$, é empregado os blocos em D_{11}, e para esta primeira rodada inicializa-se $\mathbf{D}$ e α_i com base em uma decomposição de valor singular (*Singular value decomposition - SVD*) dos blocos em D_{11}.

Fazemos várias iterações de Gibbs com D_{11} retendo a última amostra de $\mathbf{D}$ e α_i da etapa anterior. Esses $\mathbf{D}$ e α_i são então usados a amostragem na segunda rodada, agora aplicado aos blocos $B \times B$ em $D_{11} \cup D_{21}$. A amostragem de Gibbs agora é executado

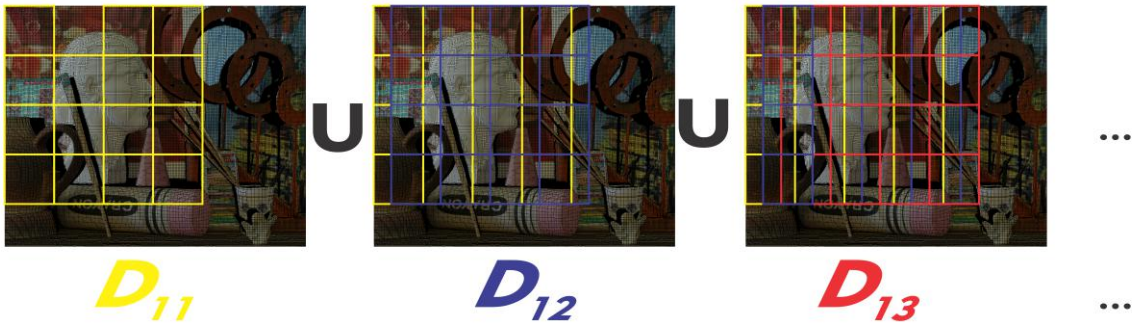


Figura 4.9: União de blocos para aprendizado de dicionário

nestes dados expandidos. Após várias iterações, a última amostra é retida, e o conjunto de dados é aumentado novamente. Isto é feito $B^2 = 64$ vezes até que todos os blocos são processados. Assim cada bloco da imagem tem sua representação atráves dos blocos(átomos) selecionados do dicionário e o buraco de expansão é preenchido com o valor do pixel respectivo em todos os blocos dividido pelo seu peso (probabilidade).

4.4 Métrica de qualidade (PSNR)

A avaliação da qualidade de imagens é importante para dimensionar distorções que podem reduzir a qualidade no momento de exibição. Existem duas formas de se obter essa avaliação: através de métodos subjetivos, isto é, por meio de notas produzidas por observadores humanos, ou através de métodos objetivos, onde algoritmos simulam o comportamento do sistema visual humano.

O grande objetivo de um modelo para quantificar a qualidade de imagens de forma objetiva é prever o resultado que um observador humano acharia de forma automática e precisa. Para isso é necessário comparar pixel a pixel a imagem resultante(vista virtual) com uma imagem *ground truth*. Métricas amplamentes utilizada pela comunidade científica são o MSE (*mean-squared error*) e o PSNR (*peak signal-to-noise ratio*) que definem a relação entre a máxima energia possível de um sinal e o ruído que afeta a representação do sinal entre as imagens comparadas. Seja a imagem de referência de tamanho $M \times N$ denotada por X e a imagem processada denotada por $\overline{X}$. O MSE é dado por [96]

$$\text{MSE} = \sqrt{\frac{1}{MN}\sum_{j=1}^{M}\sum_{i=1}^{N}\left[X(i,j) - \overline{X(i,j)}\right]^2}, \tag{4.7}$$

e representa o desvio padrão do erro. O PSNR entre as imagens é definido classicamente como [96]

$$\text{PSNR} = 10 \log_{10} \left(\frac{255}{MSE} \right) \tag{4.8}$$

para uma imagem em nível de cinzas e oito bits por nível de cinza.

Já que foi proposto somente a interpolação dos *pixels* identificados como buracos de expansão, o PSNR será calculado somente com a intensidades desses *pixels*. O método PSNR também foi escolhido para base de comparação com os trabalhos citados anteriormente em preenchimento de buracos de expansão.

Capítulo 5

Resultados

Os resultados que serão apresentados a seguir retratam os experimentos que foram realizados de acordo com a metodologia proposta. Antes de apresentar os resultados obtidos é importante ressaltar a configuração inicial utilizada nesse trabalho. As imagens utilizadas nesse trabalho foram obtidas do *Middlebury Dataset*[1]. As imagens usadas estão na tabela 5.1, assim como suas respectivas dimensões e para cada arquivo citado possuem duas imagens de textura e seu respectivo mapa de profundidade. Os mapas de profundidade foram obtidos usando técnicas descritos por Mao et. al [31].

Tabela 5.1: Lista de imagens e respectivas dimensões.

1390 x 1110	**641 x 555**
Art	*Aloe*
Laundry	*Baby*
Moebius	*Bowling*
Dolls	*Monopoly*
Books	*Plastic*
Reindeer	*Rocks*

Cada um desses arquivos contém 7 imagens estereotificadas (imagens geradas com câmeras calibradas, alinhadas com eixos ópticos paralelos, assim como sua escala e distância focal iguais), rotuladas com *Views* 0 até 6, e mapas de disparidade para as *Views* 1 e 5. Utilizou-se a *View* 1 e seu respectivo mapa de profundidade para gerar a vista virtual, sendo essa vista a imagem *ground truth* para as vistas geradas e preenchidas. A figura 5.1 mostra dois exemplos de imagens utilizadas e seu respectivo mapa de profundidade nesse trabalho.

Os parâmetros a serem definidos são basicamente a distância Δz para uma vista virtual mais próxima ou distante em relação a cena, e a definição de tamanho de bloco para identificação dos buracos de expansão e no treinamento de dicionário para a representação

[1] http://vision.middlebury.edu/stereo/data/scenes2005/

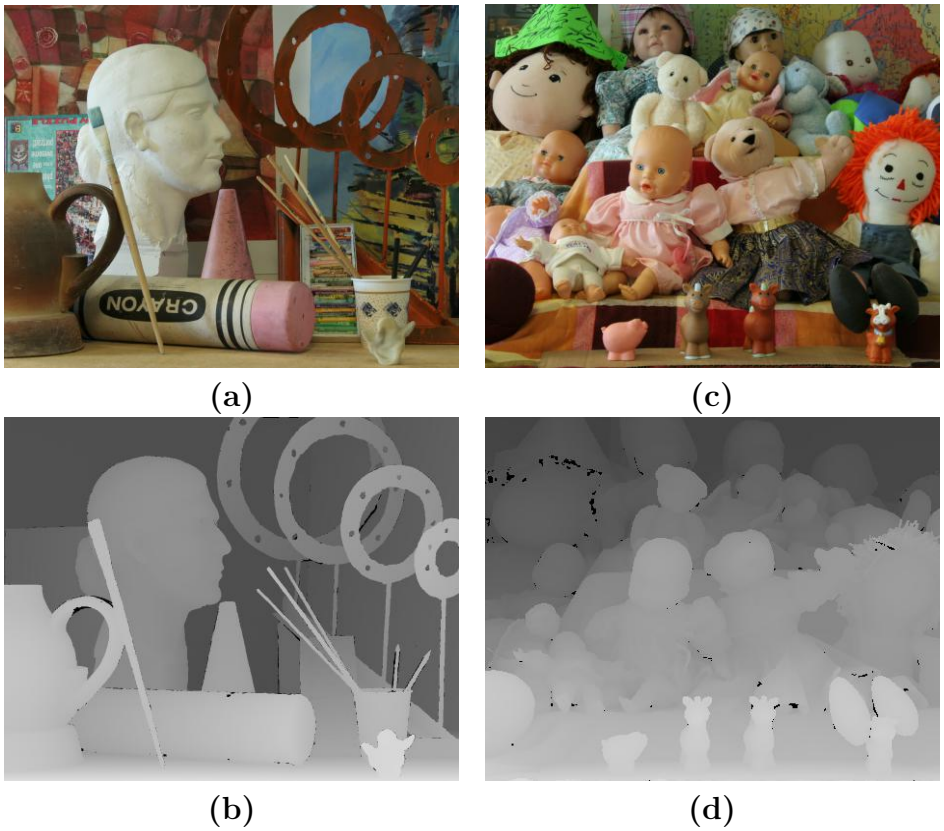


Figura 5.1: Imagem *Art* - **(a)** Imagem original e **(b)** Mapa de Profundidade. Imagem *Dolls* - **(a)** Imagem original e **(b)** Mapa de profundidade
.

esparsa. Para os testes aplicou-se o valor de Δz relativo a metade da profundidade máxima de cada imagem e tamanho de bloco igual a 8 para identificação dos buracos de expansão e para o treinamento de dicionário para *inpainting* por representação esparsa.

5.1 Síntese de vistas

Primeiramente é necessário redimensionar a imagem para uma dimensão múltipla do tamanho de bloco $B = 8$, assim a imagem de referência possui uma nova resolução de 1104×1384 e 640×552. Usamos a mesma metodologia de um trabalho anterior [14], considerando como *ground truth*, a vista de referência fornecida chamada de v_0. Em seguida, usa-se o DIBR para gerar a visualização v_r que é mais distante da câmera do que v_0. Nesta vista não haverá buracos de expansão. A Figura 5.2 retrata esse procedimento de *zoom out*.

Usando a imagem de textura e o mapa de profundidade de v_r, usa-se novamente o DIBR para gerar a vista virtual v_0 usando o parâmetro $-\Delta z$, assim tem-se a imagem

(a) **(b)**

Figura 5.2: Imagem *Art* - **(a)** vista virtual v_r e **(b)** mapa de profundidade com *zoom out* e $\Delta z = 0.0022$.

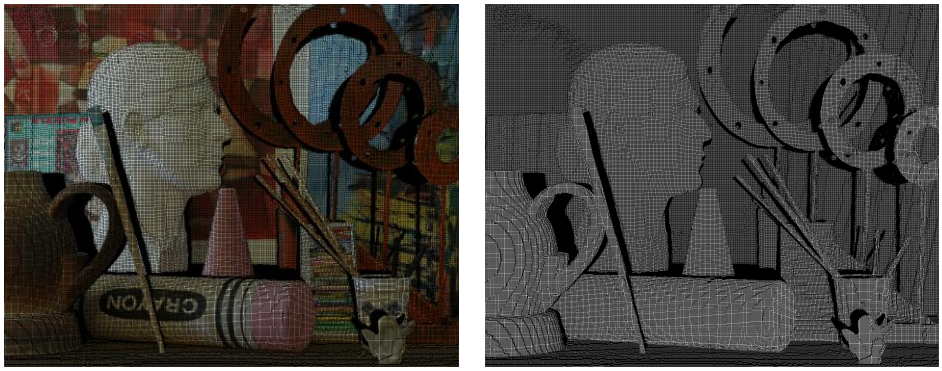
v_0 na mesma configuração da distância em relação a cena que a imagem original. Os parâmetros Δz_{max} e a profundidade máxima de cada imagem são mostrados na Tabela 5.2. Nessa imagem v_0 haverá os buracos de expansão que devem ser preenchidos assim como as desoclusões geradas. A Figura 5.3 mostra a vista virtual v_0 com os buracos de expansão. O passo a seguir é identificar e diferenciar os pixels sintetizados erroneamente classificando-os como desoclusão ou buracos de expansão.

Tabela 5.2: Valores de profundidade máxima e Δz_{max}

Imagem	**Profundidade Máxima**	Δz_{max}
Art	225	0.002222
Laundry	232	0.002155
Moebius	218	0.002293
Dolls	221	0.002262
Books	222	0.002252
Reindeer	201	0.002487
Aloe	211	0.002369
Baby	137	0.003649
Bowling	230	0.002173
Monopoly	160	0.003125
Plastic	196	0.002551
Rocks	170	0.002941

5.2 Identificando buracos de expansão

O procedimento para identificação dos buracos de expansão foi realizado com uma tamanho de bloco compatível com o tamanho da imagem redimensionada, é importante salientar que esses tipos de buracos não necessariamente tem valor nulo (como a desoclusão),



(a) **(b)**

Figura 5.3: Imagem *Art* - vista virtual v_0 - **(a)** Textura e **(b)** mapa de profundidade.

pois os *pixels* de profundidade menor (mais longe em relação a câmera) podem também ser configurados como buracos de expansão. Então seguindo a metodologia proposta é gerado uma máscara onde as posições dos *pixels* da imagem do mapa de profundidade que foram considerados como buracos de expansão possuem valor 1 ou seja não-nulo. A Figura 5.4 mostra os pixels identificados.

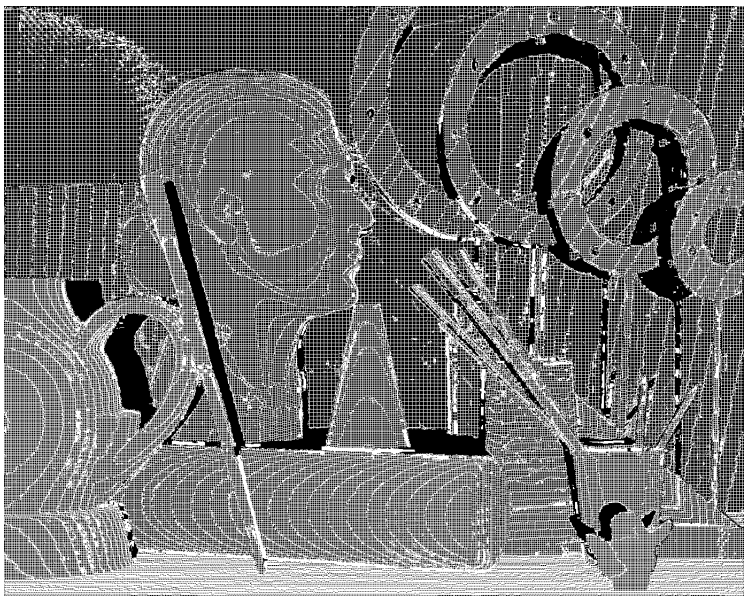


Figura 5.4: Buracos de Expansão (*pixels* brancos) identificados

É notável a presença de buracos de expansão em toda a imagem o que a torna uma grande problema. A característica desses buracos é muito diferente das desoclusões que possuem áreas com região contígua muito maior. Por isso a necessidade de se preencher somentes os buracos de expansão se torna relevante.

Os resultados mostrados a seguir vão se dividir em duas partes. No primeiro momento é demonstrado os resultados obtidos com os já estabelecidos na literatura. Posteriormente

apresentaremos os experimentos e seus resultados com uma quantidade maior de imagens e com mais algoritmos propostos explicados.

5.3 Resultados comparados com artigos da área

As imagens interpoladas resultantes são comparadas com trabalhos anteriores de [14, 15, 16]. Essas técnicas implementam respectivamente uma interpolação baseada em grafos com restrição de esparsidade (GBT), interpolação baseada em grafos com Médias Não-Locais (NLGBT) e Transformada de Fourier em grafos (AGFT).

Todas esses trabalhos relataram um valor de PSNR (somente considerando os buracos de expansão) para um método que modificou o software VSRS versão 3.5 (chamada de VSRS+) e usam uma técnica de *inpainting* padrão para preencher todos os buracos na visualização virtual. Os valores em PSNR foram de 19.11 dB em imagem *Art* e 19.17 dB na imagem *Laundry*. Estes métodos propuseram a abordagem da camada de profundidade para identificar os buracos de expansão. Em todos os casos, o PSNR é calculado considerando somente as áreas de buracos de expansão.

Todas as imagens geradas e usadas são mostradas na Figura 5.5 e na Figura 5.6 para imagem *Art* e *Laundry* respectivamente. Em primeiro lugar é mostrada uma imagem de *ground-truth* (e vista de referência), as duas imagens seguintes são o processo de *zoom-out / zoom-in*, note no caso de *zoom-out* não há buracos de expansão. A última imagem mostra a imagem preenchida com a abordagem proposta neste trabalho.

Na Figura 5.7 vemos uma imagem recortada de uma visão sintetizada da imagem *Art*, (a) é uma vista virtual com buracos de expansão e de desoclusão, (b) mostra a máscara onde os pixels brancos são identificados como buraco de expansão e (c) os pixels preenchidos com a representação esparsa explicada. O algoritmo recursivo foi definido para funcionar com 4 níveis.

Na Tabela 5.3 são mostrados os valores de PSNR para as imagens testadas. Nosso método superou os trabalhos anteriores para essas imagens. Para a imagem *Art* a diferença com VSRS + é muito significativa, em torno de 6.19 dB. E nosso método superou o GBT em 1.94 dB, NLGBT em 1.72 dB e AGBT em 1.61 dB

Tabela 5.3: Comparação em PSNR dos métodos de *inpainting*

	GBT	NLGBT	AGBT	Proposto
Art	23.36	23.58	23.69	**25.30**
Laundry	22.53	-	23.04	**23.70**

A imagem *Laundry* tem resultados semelhantes. Nossa abordagem tem um melhor de-

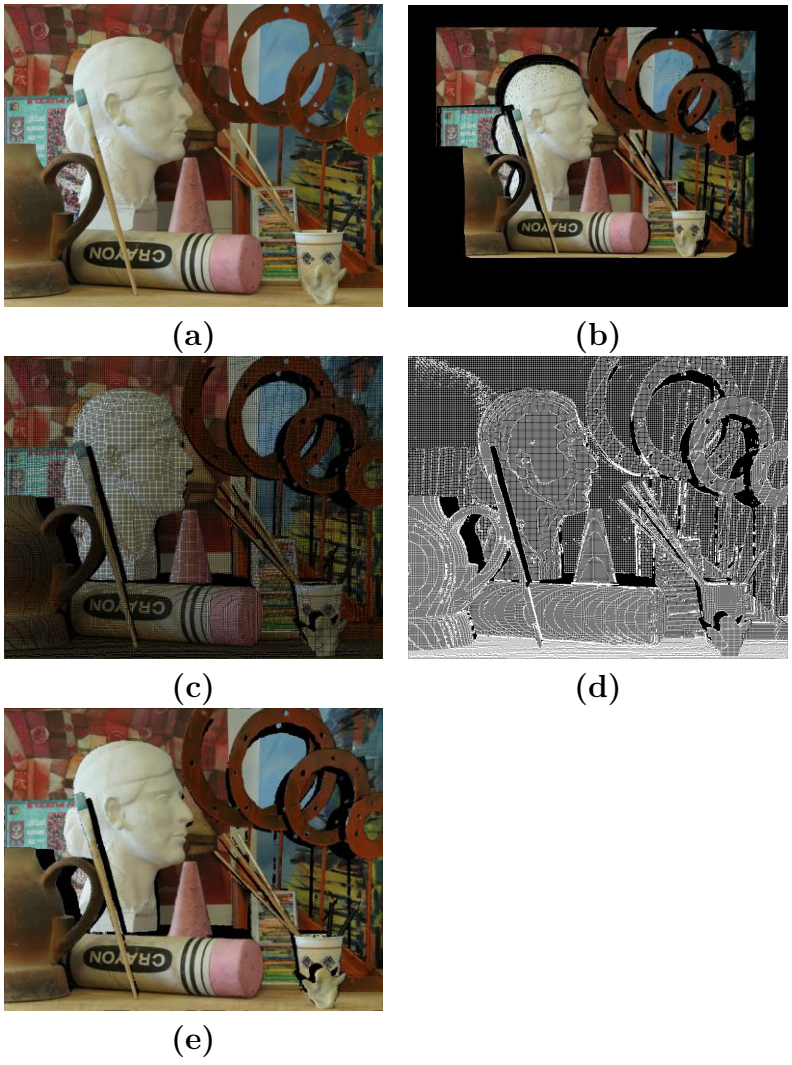


Figura 5.5: Imagens para *Art* - **(a)** *Ground truth*, **(b)** *zoom-out*, **(c)** *zoom-in*, **(d)** máscara dos buracos de expansão e **(e)** vista preenchida.

sempenho e superou o VSRS + em 4.53 dB, GBT em 1.17 dB e AGBT em 0.66 dB. O algoritmo NLGBT não forneceu um valor PSNR para a imagem *Laundry*

5.4 Resultados comparados com mais imagens e outros algoritmos

Os resultados mostrados nessa seção são um acréscimo dos resultados mostrados na seção anterior. Esse acréscimo corresponde a mais imagens e outros algoritmos que foram implementados para comparação com a abordagem proposta nesse trabalho, as algoritmos utilizados são;

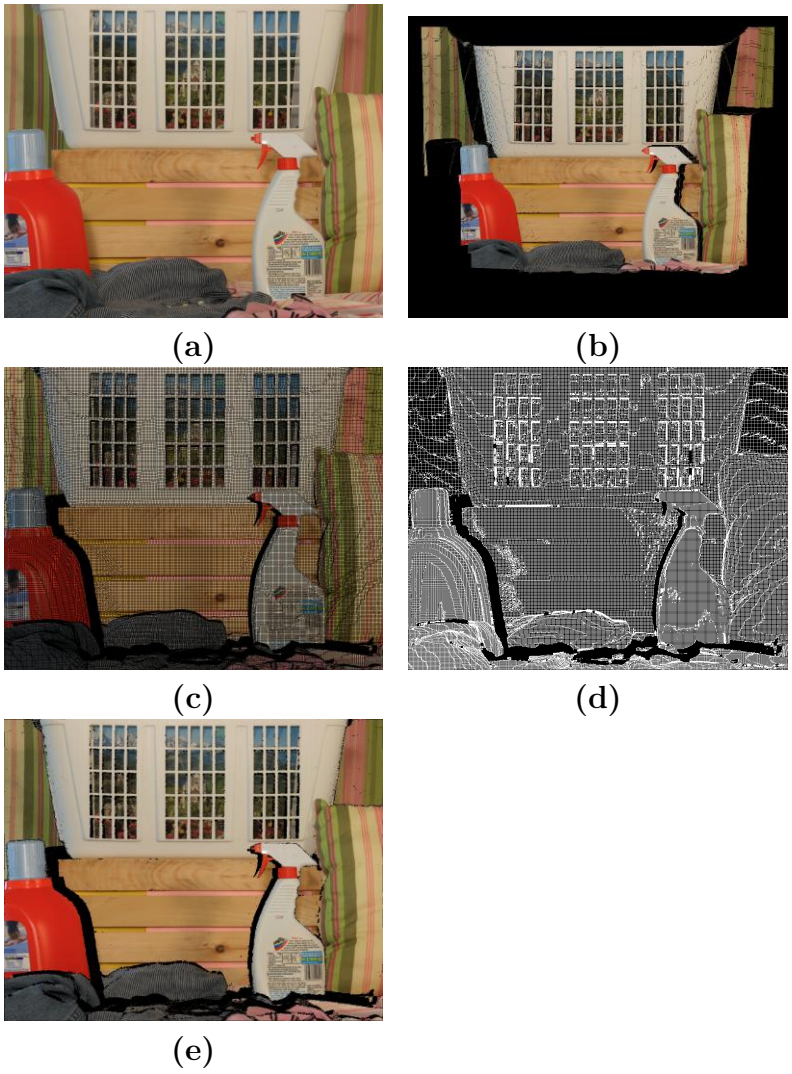


Figura 5.6: Imagens para *Laundry* - **(a)** *Ground truth*, **(b)** *zoom-out*, **(c)** *zoom-in*, **(d)** máscara dos buracos de expansão e **(e)** vista preenchida.

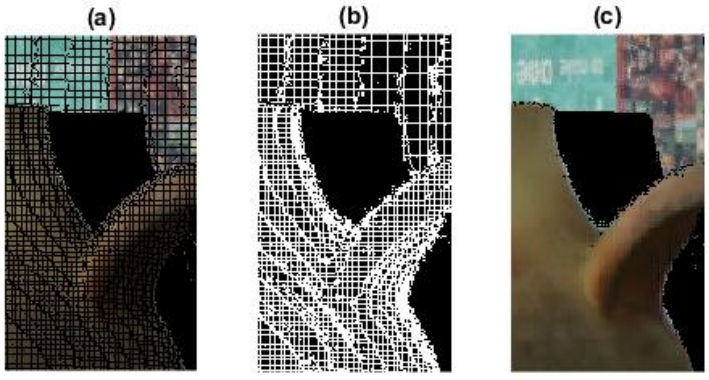


Figura 5.7: (a) vista virtual,(b) máscara com buracos de expansão e (c) pixels preenchidos com representação esparsa proposta

- **VSRS+** - Esse algoritmo foi criado pelo grupo MPEG, o VSRS (View Synthesis Reference Software) é um programa que realiza o procedimento de síntese de vista, sua abordagem aplica o método padrão de *inpainting* do OpenCV desenvolvido por [97], para adequar ao contexto de preenchimento de buracos de expansão, renomeamos para VSRS+;
- **Linear** - Esse algoritmo foi implementado baseado em [14]. Para cada pixel vazio identificado como buraco de expansão, busca-se os três pixels sintetizados mais próximos i, j, k e constrói um plano linear que conecta seus valores de textura, $t(x_i, y_i)$, $t(x_j, y_j)$, $t(x_k, y_k)$ dadas suas coordenadas (x_i, y_i), (x_j, y_j), (x_k, y_k). O pixel vazio é interpolado usando o plano construído e sua própria coordenada de pixel.
- **K-SVD** - Algoritmo desenvolvido por [18], seu funcionamento pode ser verificado na Seção 3.4.1
- **OMP - Orthogonal Matching Pursuit** - Algoritmo desenvolvido por [51], seu funcionamento pode ser verificado na Seção 3.4.2

Os resultados em PNSR para as imagens de resolução diferentes e com os algoritmos indicados previamente são mostrados nas tabelas 5.4 e 5.5, lembrando que esses valores foram gerados considerando somente os buracos de expansão.

Tabela 5.4: PSNR para imagens 1390×1110

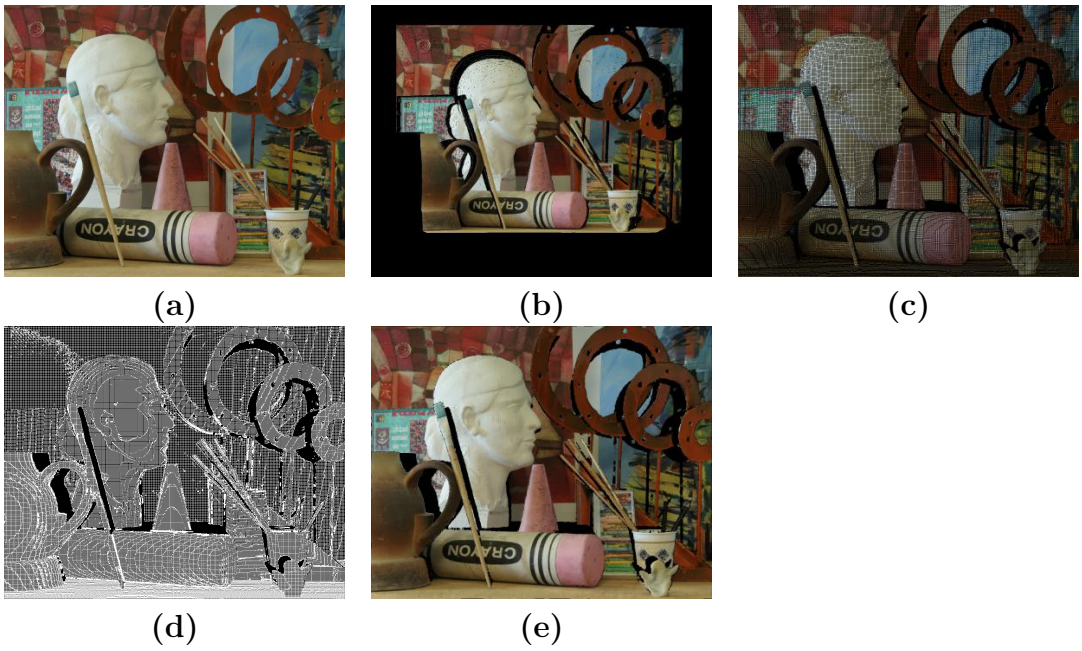
	VSRS+	Linear	K-SVD	OMP	Proposto
Art	19.11	20.38	22.72	21.72	**25.30**
Laundry	19.17	19.23	22.23	21.56	**23.70**
Moebius	16.80	17.74	26.82	25.20	**27.92**
Dolls	17.83	18.06	25.20	24.19	**33.67**
Books	19.27	19.92	26.28	24.81	**34.58**
Reindeer	24.27	25.25	30.34	28.22	**39.22**

Tabela 5.5: PSNR para imagens 640×552

	VSRS+	Linear	K-SVD	OMP	Proposto
Aloe	19.22	15.64	21.53	19.75	**35.03**
Baby	23.67	21.12	27.51	26.03	**39.45**
Bowling	21.60	15.52	30.74	27.68	**40.81**
Monopoly	18.37	14.99	23.69	21.64	**35.00**
Plastic	28.02	25.38	34.10	33.19	**34.75**
Rocks	25.06	21.16	28.12	27.53	**37.66**

As próximas seções mostram as imagens geradas tanto do processo de síntese de cada imagem e um comparação subjetiva de todos os algoritmos implementados.

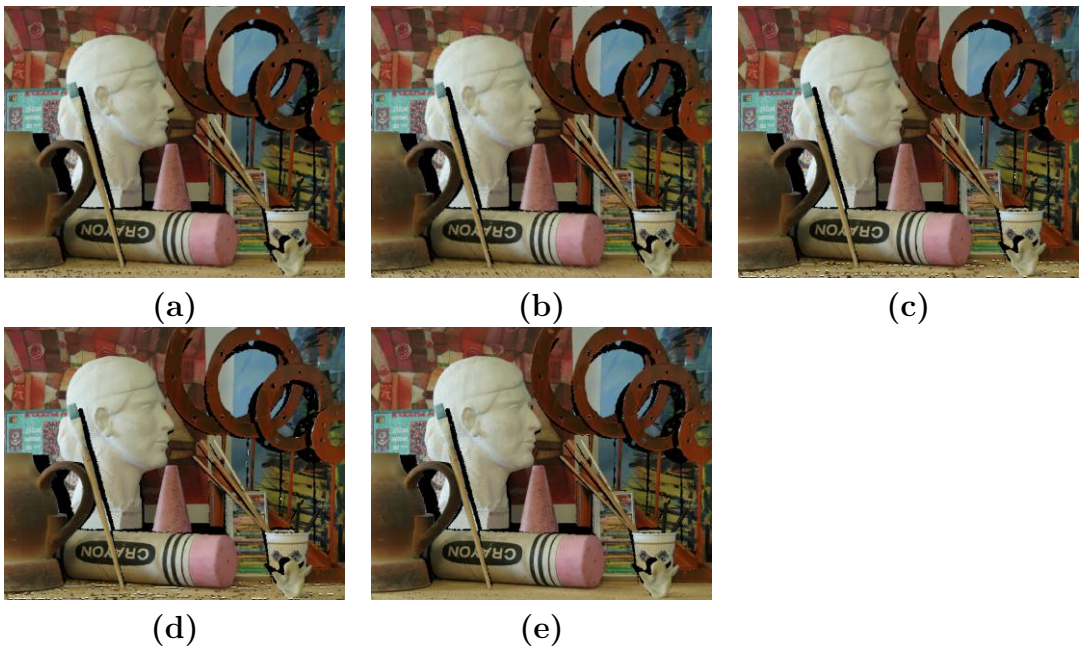
5.4.1 *Art*



(a) **(b)** **(c)**

(d) **(e)**

Figura 5.8: Imagens para *Art* - **(a)** *Ground truth*, **(b)** *zoom-out*, **(c)** *zoom-in*, **(d)** máscara dos buracos de expansão e **(e)** vista preenchida com algoritmo proposto.

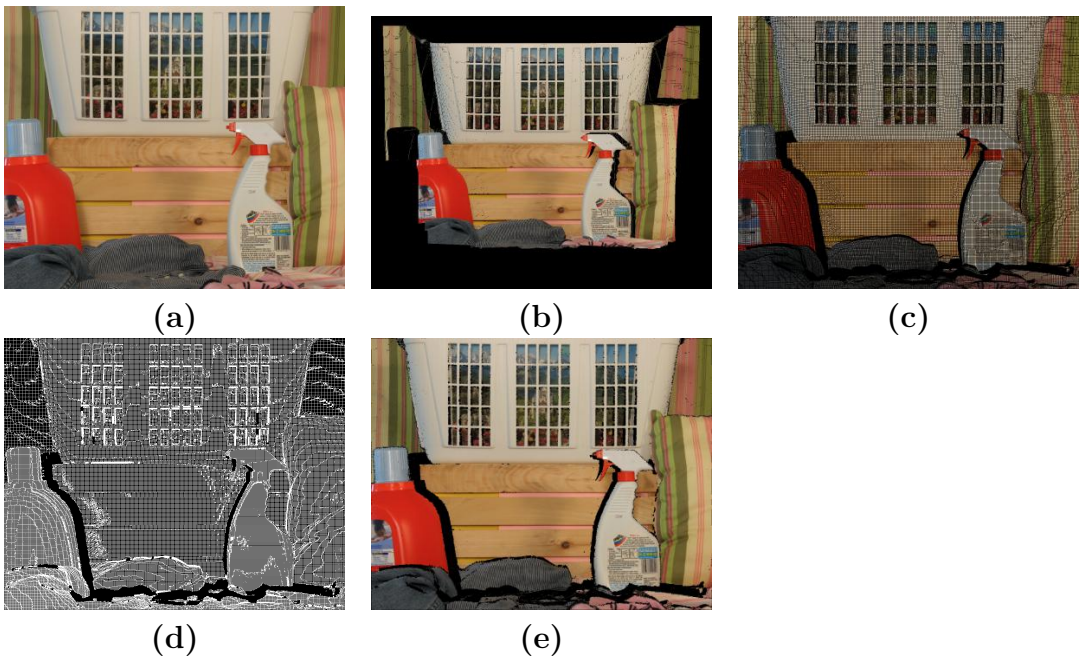


(a) **(b)** **(c)**

(d) **(e)**

Figura 5.9: Imagens preenchidas para *Art* - **(a)** VSRS+, **(b)** Interpolação Linear, **(c)** K-SVD, **(d)** OMP e **(e)** algoritmo proposto

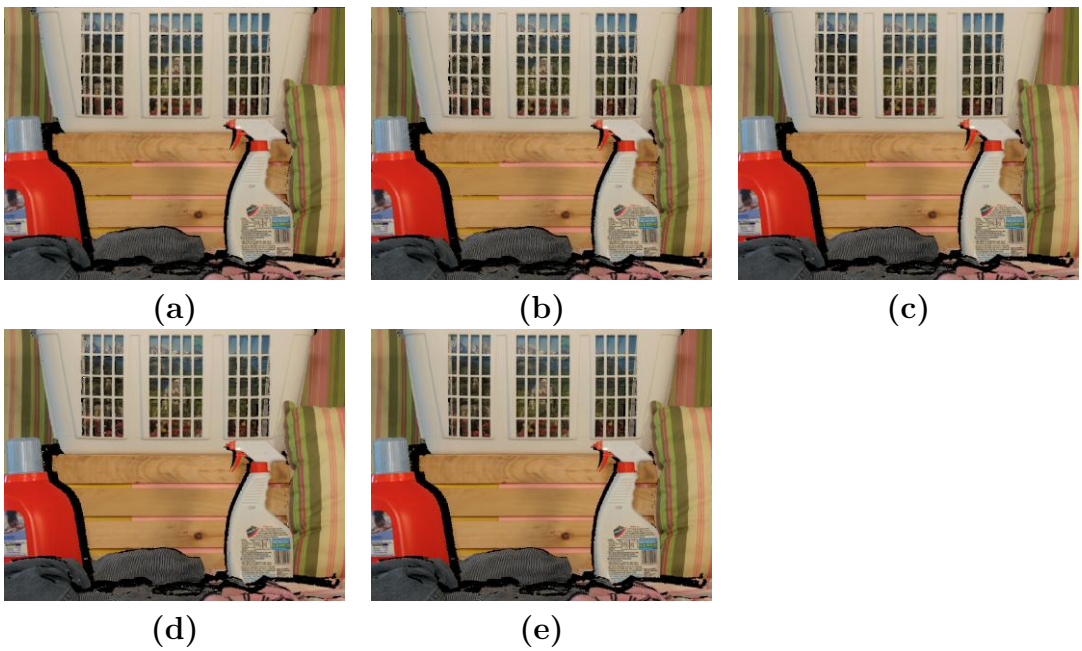
5.4.2 *Laundry*



(a) **(b)** **(c)**

(d) **(e)**

Figura 5.10: Imagens para *Laundry* - **(a)** *Ground truth*, **(b)** *zoom-out*, **(c)** *zoom-in*, **(d)** máscara dos buracos de expansão e **(e)** vista preenchida com algoritmo proposto.

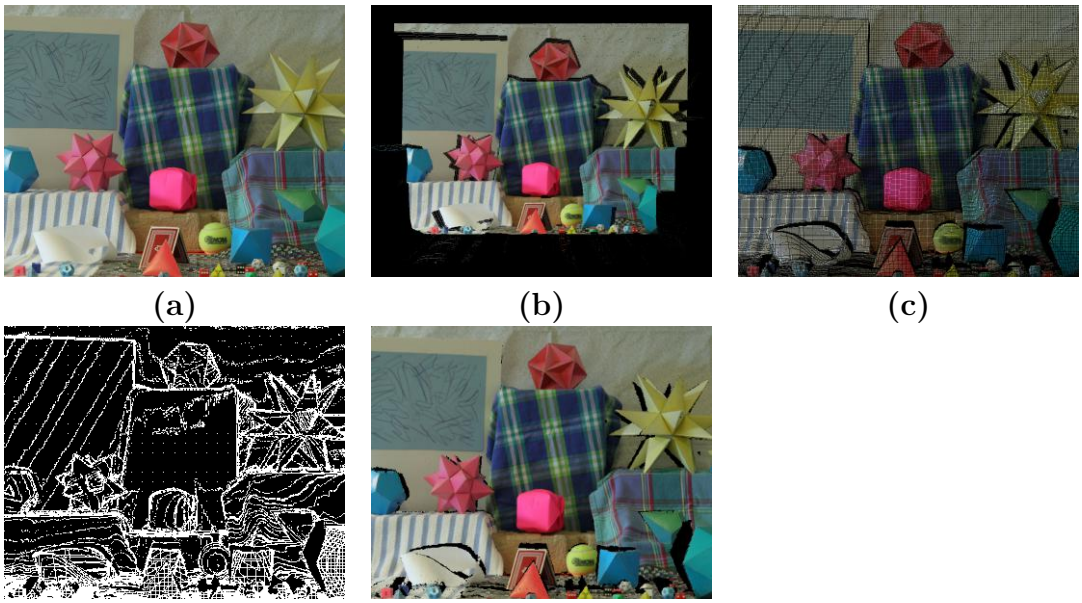


(a) **(b)** **(c)**

(d) **(e)**

Figura 5.11: Imagens preenchidas para *Art* - **(a)** VSRS+, **(b)** Interpolação Linear, **(c)** K-SVD, **(d)** OMP e **(e)** algoritmo proposto.

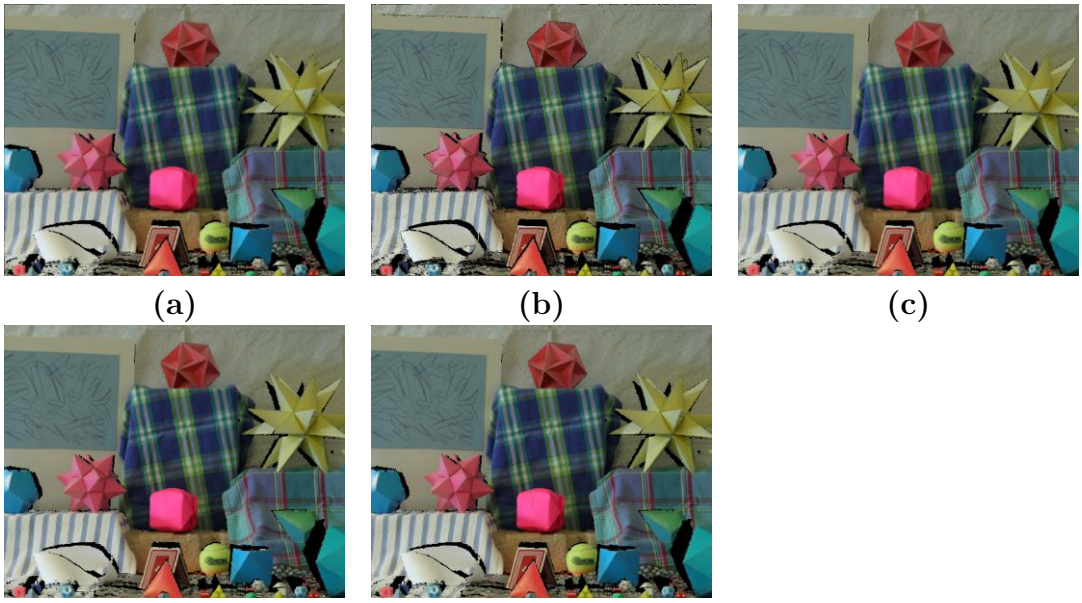
5.4.3 *Moebius*



(a) **(b)** **(c)**

(d) **(e)**

Figura 5.12: Imagens para *Moebius* - **(a)** *Ground truth*, **(b)** *zoom-out*, **(c)** *zoom-in*, **(d)** máscara dos buracos de expansão e **(e)** vista preenchida com algoritmo proposto.



(a) **(b)** **(c)**

(d) **(e)**

Figura 5.13: Imagens preenchidas para *Moebius* - **(a)** VSRS+, **(b)** Interpolação Linear, **(c)** K-SVD, **(d)** OMP e **(e)** algoritmo proposto.

5.4.4 *Dolls*

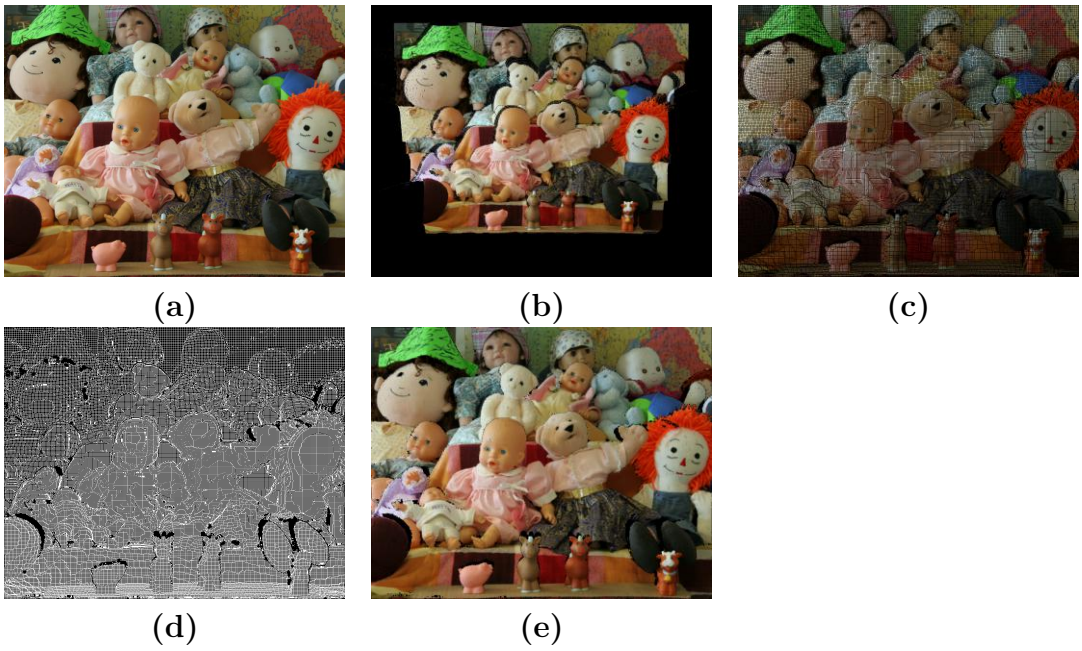


Figura 5.14: Imagens para *Dolls* - **(a)** *Ground truth*, **(b)** *zoom-out*, **(c)** *zoom-in*, **(d)** máscara dos buracos de expansão e **(e)** vista preenchida com algoritmo proposto.

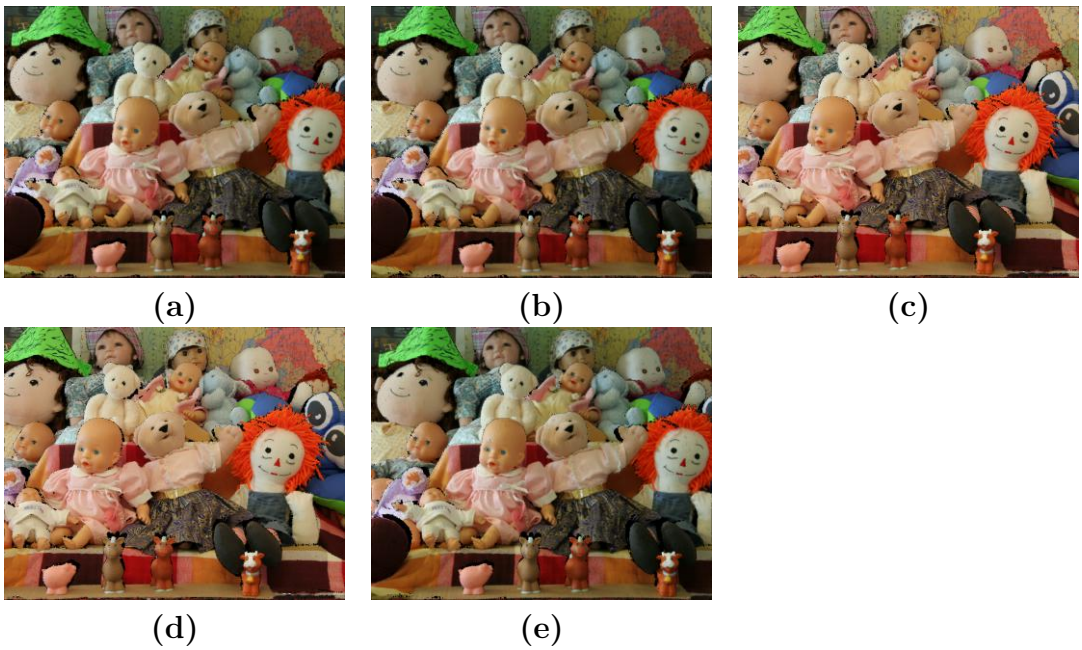


Figura 5.15: Imagens preenchidas para *Dolls* - **(a)** VSRS+, **(b)** Interpolação Linear, **(c)** K-SVD, **(d)** OMP e **(e)** algoritmo proposto.

5.4.5 *Books*

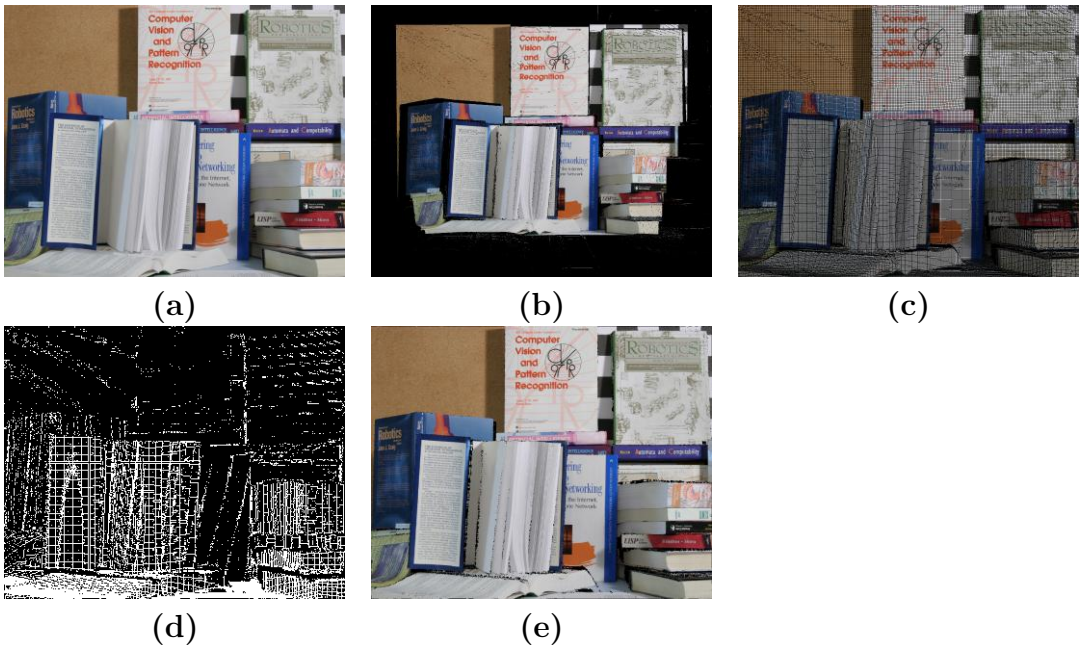


Figura 5.16: Imagens para *Books* - **(a)** *Ground truth*, **(b)** *zoom-out*, **(c)** *zoom-in*, **(d)** máscara dos buracos de expansão e **(e)** vista preenchida com algoritmo proposto.

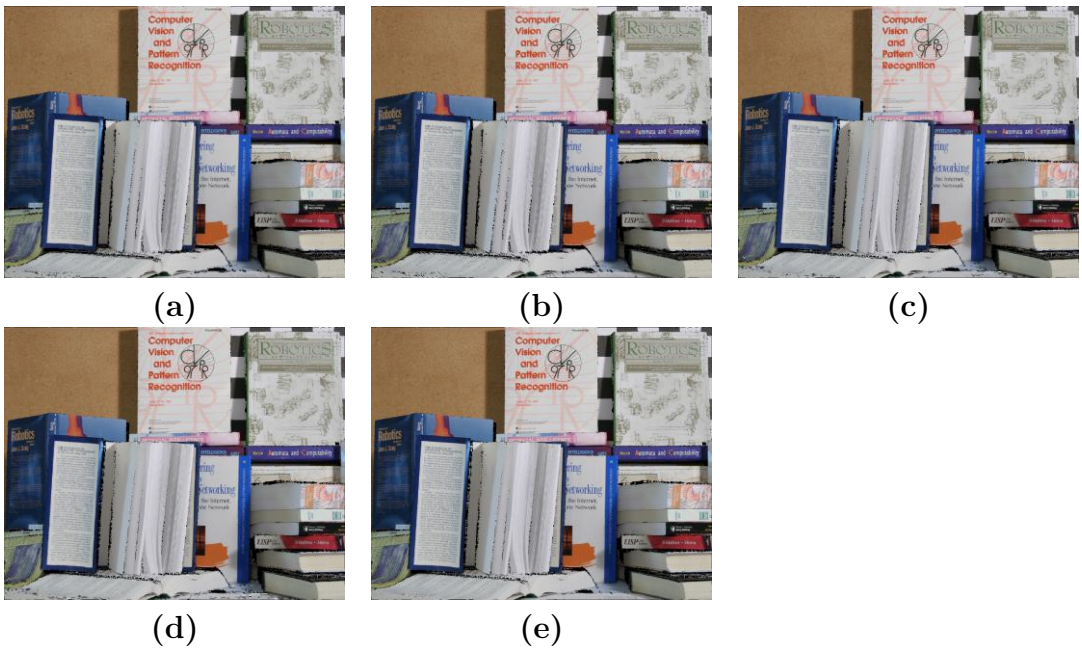


Figura 5.17: Imagens preenchidas para *Books* - **(a)** VSRS+, **(b)** Interpolação Linear, **(c)** K-SVD, **(d)** OMP e **(e)** algoritmo proposto.

5.4.6 *Reindeer*

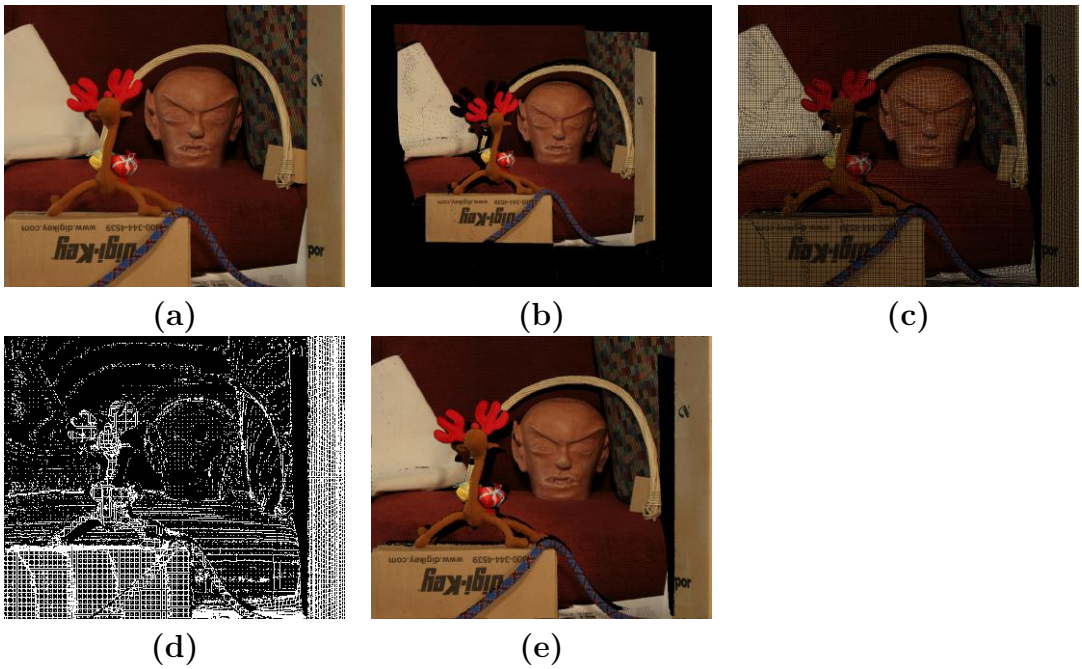


Figura 5.18: Imagens para *Reindeer* - **(a)** *Ground truth*, **(b)** *zoom-out*, **(c)** *zoom-in*, **(d)** máscara dos buracos de expansão e **(e)** vista preenchida com algoritmo proposto.

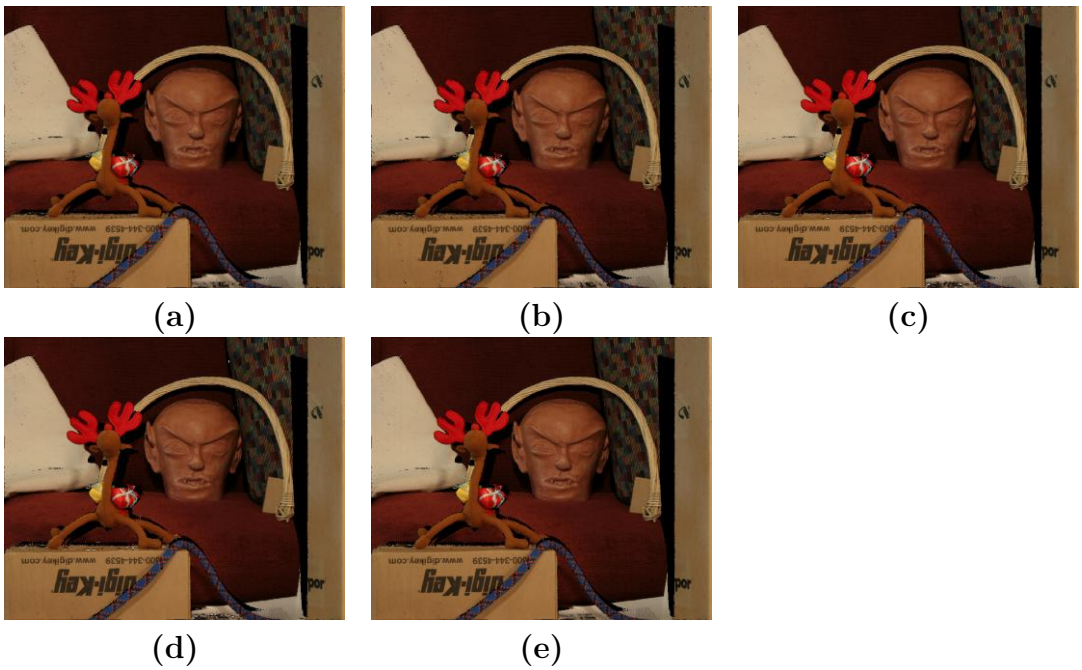


Figura 5.19: Imagens preenchidas para *Reindeer* - **(a)** VSRS+, **(b)** Interpolação Linear, **(c)** K-SVD, **(d)** OMP e **(e)** algoritmo proposto.

5.4.7 *Aloe*

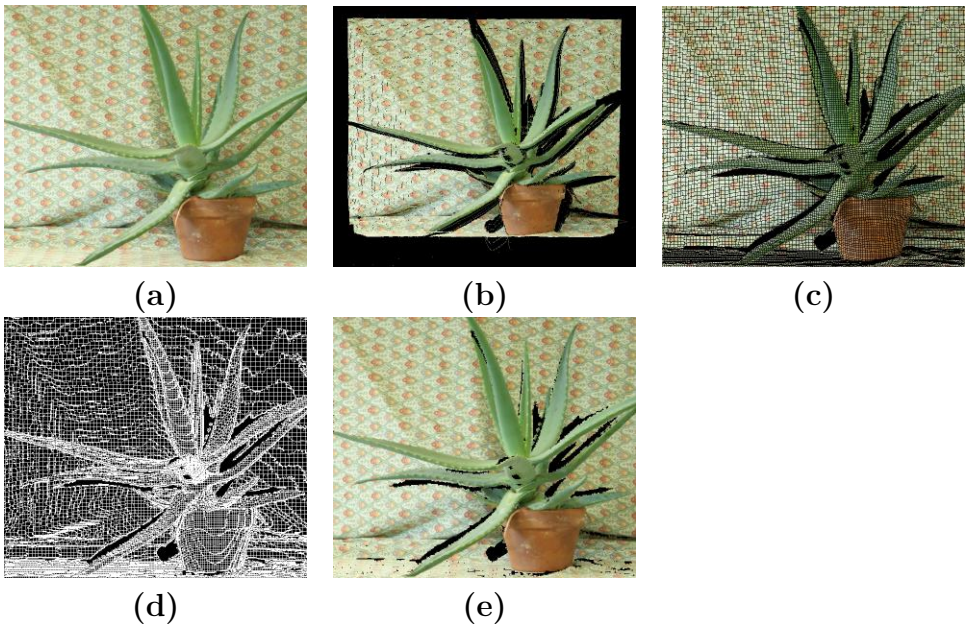


Figura 5.20: Imagens para *Aloe* - **(a)** *Ground truth*, **(b)** *zoom-out*, **(c)** *zoom-in*, **(d)** máscara dos buracos de expansão e **(e)** vista preenchida com algoritmo proposto.

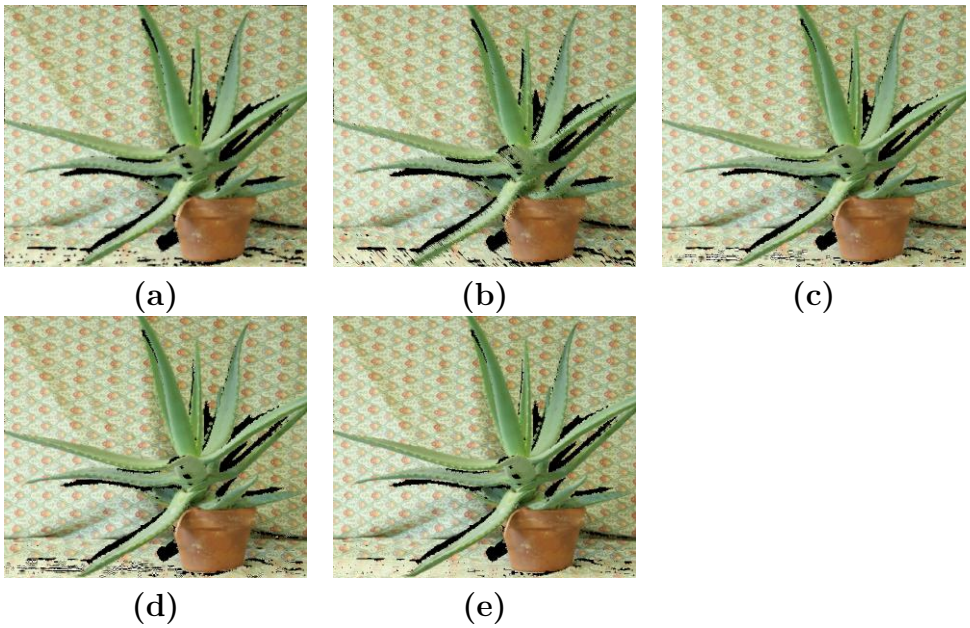
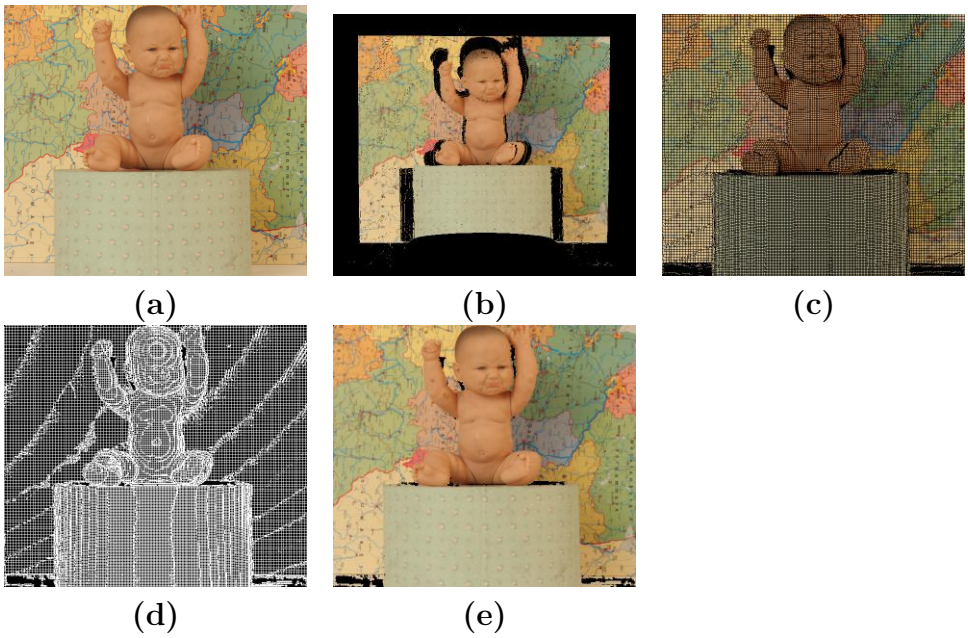


Figura 5.21: Imagens preenchidas para *Aloe* - **(a)** VSRS+, **(b)** Interpolação Linear, **(c)** K-SVD, **(d)** OMP e **(e)** algoritmo proposto.

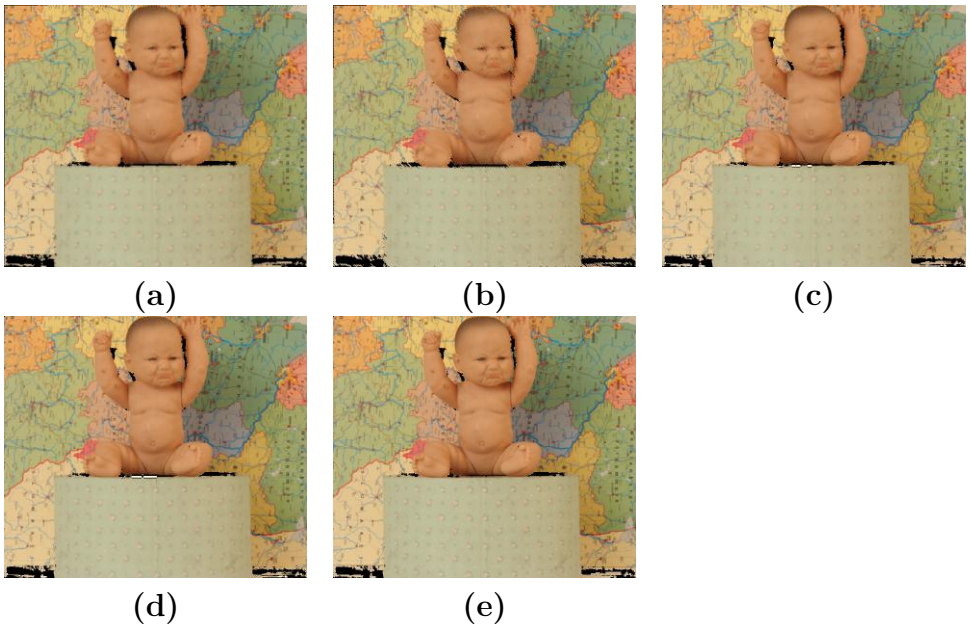
5.4.8 *Baby*



(a) **(b)** **(c)**

(d) **(e)**

Figura 5.22: Imagens para *Baby* - **(a)** *Ground truth*, **(b)** *zoom-out*, **(c)** *zoom-in*, **(d)** máscara dos buracos de expansão e **(e)** vista preenchida com algoritmo proposto.

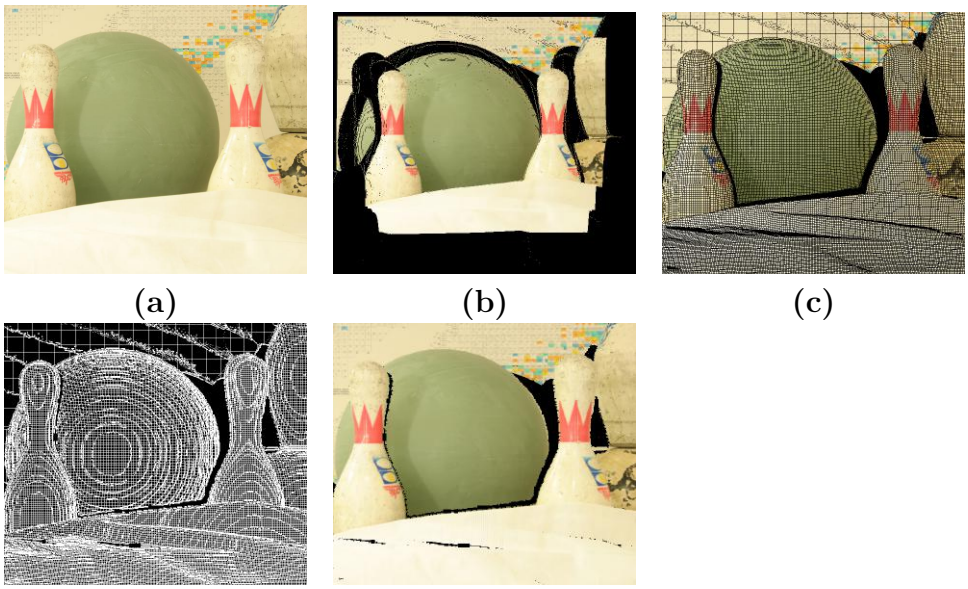


(a) **(b)** **(c)**

(d) **(e)**

Figura 5.23: Imagens preenchidas para *Baby* - **(a)** VSRS+, **(b)** Interpolação Linear, **(c)** K-SVD, **(d)** OMP e **(e)** algoritmo proposto.

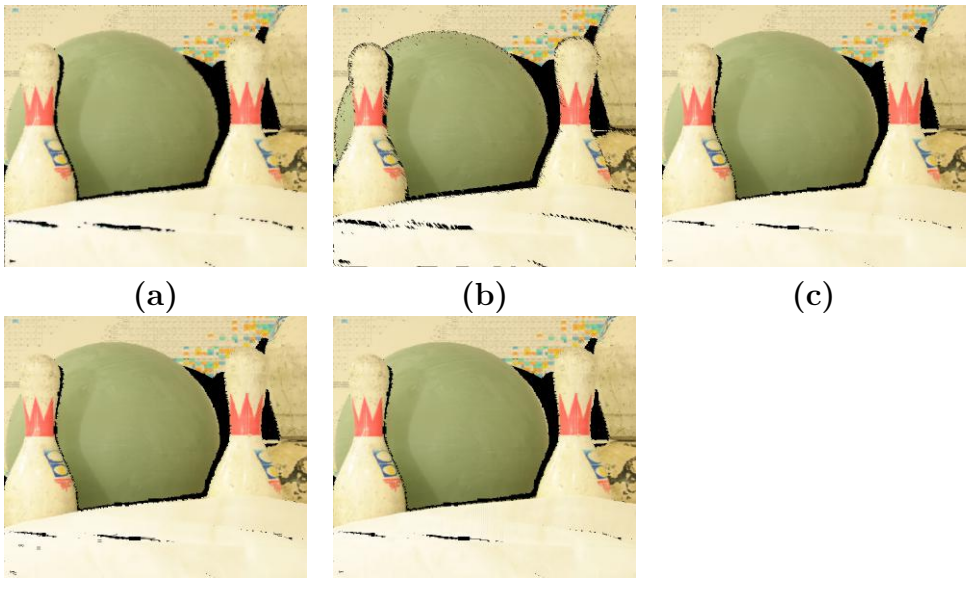
5.4.9 *Bowling*



(a) **(b)** **(c)**

(d) **(e)**

Figura 5.24: Imagens para *Bowling* - **(a)** *Ground truth*, **(b)** *zoom-out*, **(c)** *zoom-in*, **(d)** máscara dos buracos de expansão e **(e)** vista preenchida com algoritmo proposto.



(a) **(b)** **(c)**

(d) **(e)**

Figura 5.25: Imagens preenchidas para *Bowling* - **(a)** VSRS+, **(b)** Interpolação Linear, **(c)** K-SVD, **(d)** OMP e **(e)** algoritmo proposto.

5.4.10 *Monopoly*

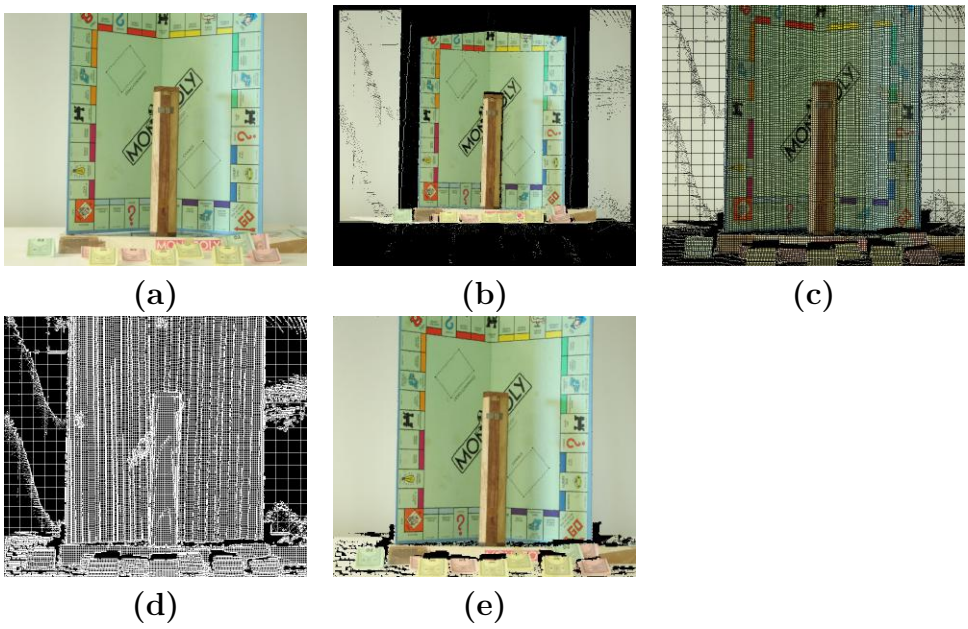


Figura 5.26: Imagens para *Monopoly* - **(a)** *Ground truth*, **(b)** *zoom-out*, **(c)** *zoom-in*, **(d)** máscara dos buracos de expansão e **(e)** vista preenchida com algoritmo proposto.

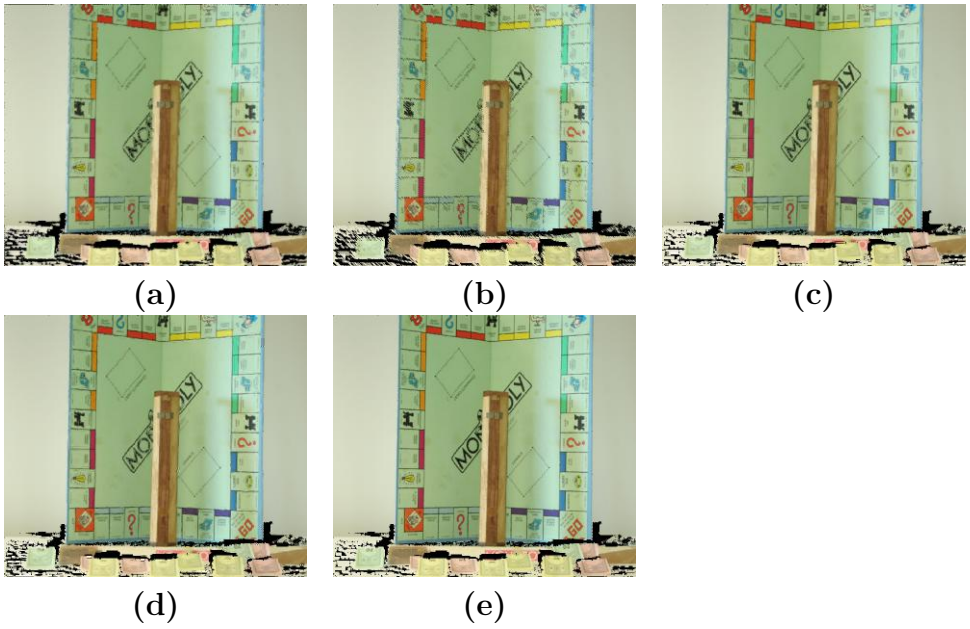
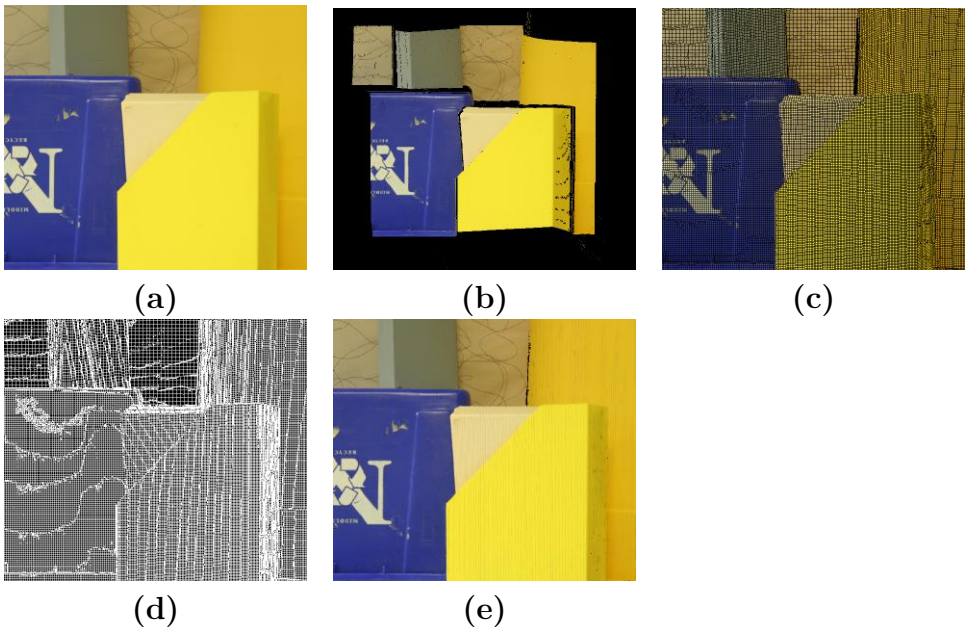


Figura 5.27: Imagens preenchidas para *Monopoly* - **(a)** VSRS+, **(b)** Interpolação Linear, **(c)** K-SVD, **(d)** OMP e **(e)** algoritmo proposto.

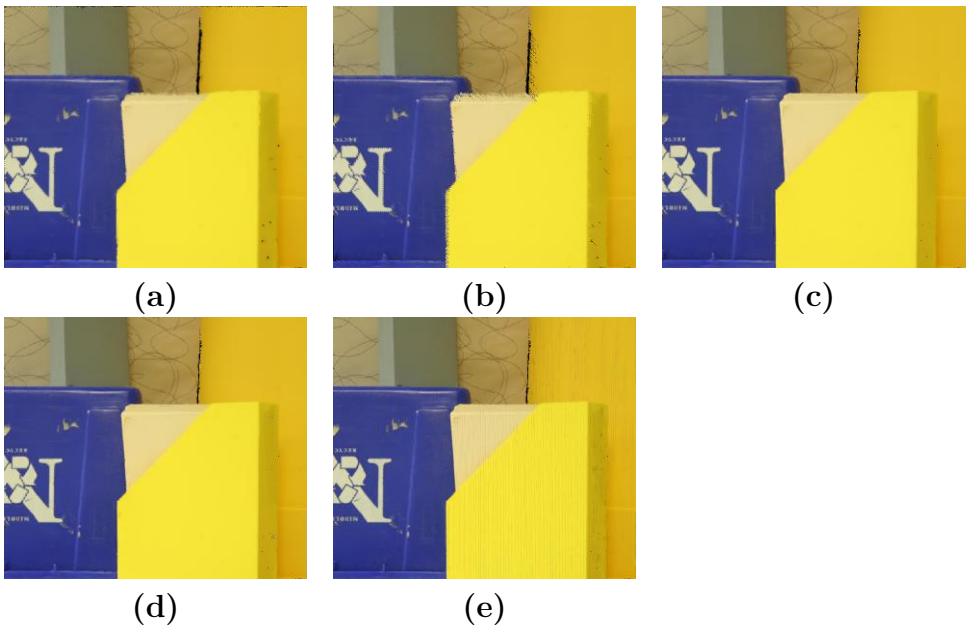
5.4.11 *Plastic*



(a) **(b)** **(c)**

(d) **(e)**

Figura 5.28: Imagens para *Plastic* - **(a)** *Ground truth*, **(b)** *zoom-out*, **(c)** *zoom-in*, **(d)** máscara dos buracos de expansão e **(e)** vista preenchida com algoritmo proposto.



(a) **(b)** **(c)**

(d) **(e)**

Figura 5.29: Imagens preenchidas para *Plastic* - **(a)** VSRS+, **(b)** Interpolação Linear, **(c)** K-SVD, **(d)** OMP e **(e)** algoritmo proposto.

5.4.12 *Rocks*

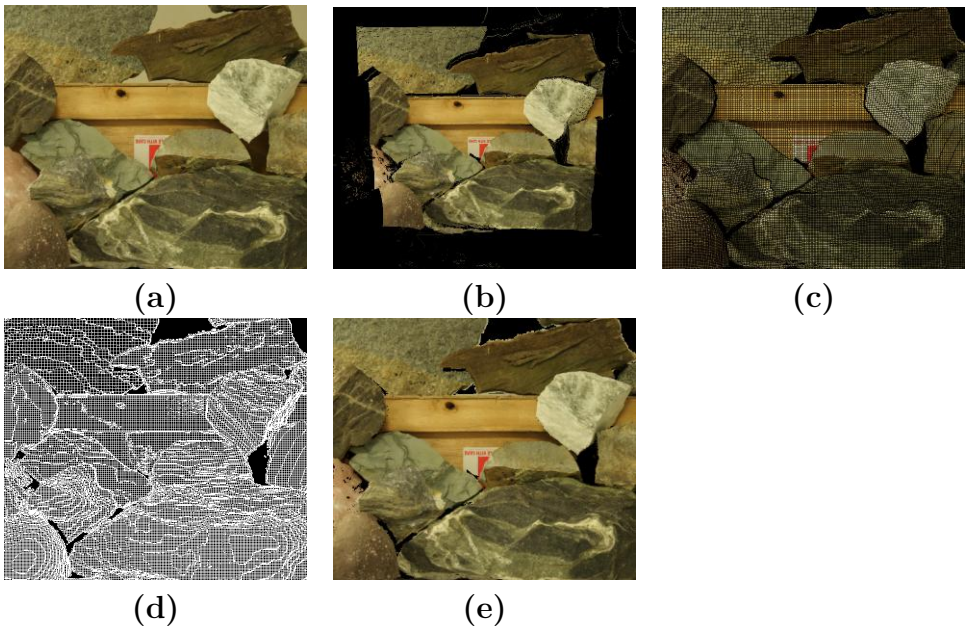


Figura 5.30: Imagens para *Rocks* - **(a)** *Ground truth*, **(b)** *zoom-out*, **(c)** *zoom-in*, **(d)** máscara dos buracos de expansão e **(e)** vista preenchida com algoritmo proposto.

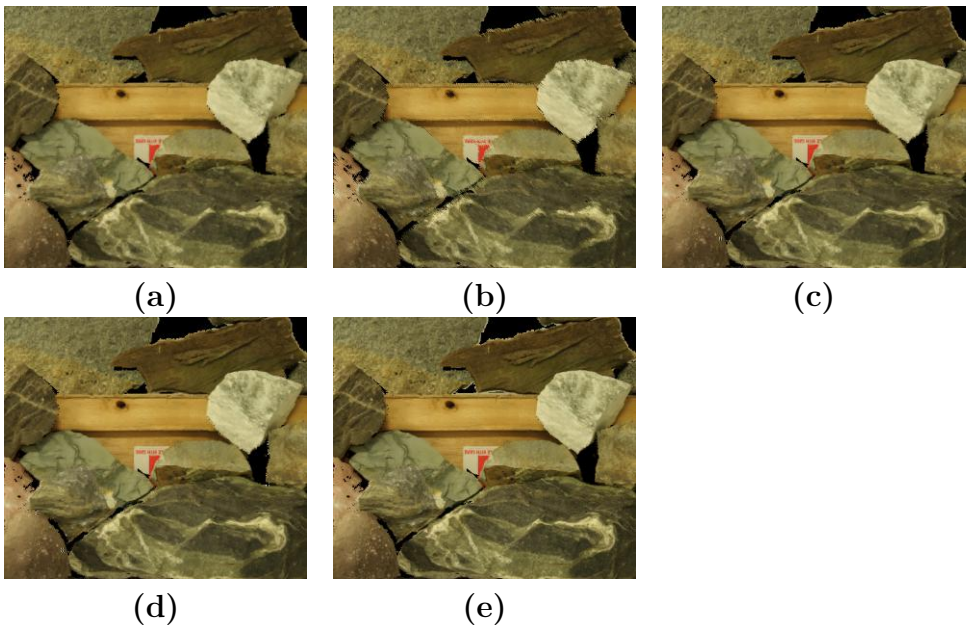


Figura 5.31: Imagens preenchidas para *Rocks* - **(a)** VSRS+, **(b)** Interpolação Linear, **(c)** K-SVD, **(d)** OMP e **(e)** algoritmo proposto.

Capítulo 6

Conclusões

Neste trabalho, propusemos um método de *inpaiting* para buracos de expansão em síntese de vista DIBR com movimentos no eixo z. O método utiliza uma identificação consistente deste tipo de buracos que foi previamente proposto. A representação esparsa é uma modificação de também um trabalho anterior de forma a adaptar a técnica para preenchimento de buracos de expansão. Na literatura atual não há nenhum trabalho anterior que usa a representação esparsa para o preenchimento de buracos de expansão em DIBR. Além disso, a técnica de *inpainting* é realizada de forma recursiva, a fim de melhor ajustar o modelo de representação esparsa e reduzir o tempo de execução. Os resultados superaram as abordagens citadas e implementadas do estado da arte vigente na área de representação esparsa e de algoritmos de preenchimento.

Os resultados obtidos superaram todas as estratégias elaboradas para preenchimento de buracos de expansão tanto da literatura atual, quanto das modificações em implementações já vigentes. O valores de PSNR variam de acordo com os tipos de imagens e dois fatores parecem ser relevantes para um melhores resultados. Primeiramente, imagens muitos texturizadas e com vários objetos apresentaram resultados objetivos e subjetivos piores, exemplos incluem a imagem *Art* e *Laundry*. Além disso objetos muito próximos tendem a ter mais buracos de expansão e o tamanho de bloco usado na etapa de identificação talvez não achasse corretamente esses tipos de buracos, visto não ter uma vizinhança que contivesse *pixels* não nulos.

Num primeiro momento o algoritmo de preenchimento por representação esparsa foi executado sobre a imagem (vista virtual) final, com resolução igual ao *ground-truth*, essa abordagem exigia uma alta carga computacional e com a implementação recursiva de preenchimento dessas vistas nas imagens dizimadas, o tempo de execução de algoritmo diminuiu bastante.

Quando ocorre uma significativa mudança no eixo z, objetos mais próximos da câmera crescem em tamanho de área mais que objetos mais longes. O número insuficiente de *pixels*

para representar esse objeto cria os buracos de expansão. Nessa condição, a vista virtual gerada tem muitos buracos de expansão nessas áreas mais próximas, e nesse contexto imagens com objetos mais distantes tiveram uma avaliação subjetiva e objetiva melhor.

Com isso é possível sugerir algumas melhorias para trabalhos futuros

- Implementação de dicionário de representação esparsa utilizando blocos ou imagens já treinadas;
- Melhor ajuste de tamanhos de bloco - uma abordagem mais dinâmica de escolha de tamanho de blocos podem apresentar melhores resultados;
- Aplicar essa técnica também para o preenchimento de desoclusões;
- Realizar um preenchimento híbrido para preencher os buracos com técnicas que melhor se adaptem a determinado tipo de buraco.

Referências

[1] Jain, Ankit K., Lam C. Tran, Ramsin Kloshabeh e Truong Q. Nguyen: *Efficient stereo-to-multiview sysnthesis*. Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference, páginas 889–892, 2011. ix, 3

[2] Zhou, M., H. Chen, J. Paisley, L. Ren, L. Li, Z. Xing, D. Dunson, G. Sapiro e L. Carin: *Nonparametric bayesian dictionary learning for analysis of noisy and incomplete images*. IEEE Transactions on Image Processing, 21(1):130–144, Jan 2012, ISSN 1057-7149. ix, 4, 27

[3] Vizzotto, Bruno Boessio: *Algoritmos para o módulo de controle de taxa de codificação de vídeo multivistas do padrão h.264/mvc*. Tese de Mestrado, Universidade Federal do Rio Grande do Sul, 2012. ix, 9, 10

[4] Zhu, Ce, Yin Zhao, Lu Yu e Masayuki Tanimoto: *3D-TV System with Depth-Image-Based Rendering: Architectures, Techniques and Challenges*. Springer Publishing Company, Incorporated, 2012, ISBN 1441999639, 9781441999634. ix, 2, 12, 15, 19, 20, 21

[5] Orbanz, Peter e Yee Whye Teh: *Bayesian Nonparametric Models*, páginas 81–89. Springer US, Boston, MA, 2010, ISBN 978-0-387-30164-8. `http://dx.doi.org/10.1007/978-0-387-30164-8_66`. ix, 28, 29, 30

[6] Tomoyose, Alexandre N., Silvio R. R. Sanches e Romero Tori: *Integração da estereoscopia à mecânica dos jogos*. VIII Brazilian Symposium on Games and Digital Entertainment, 2009. 2

[7] Tanimoto, M., M. P. Tehrani, T. Fujii e Yendo T.: *Free-viewpoint tv*. IEEE Signal Processing Magazine, 28(1), Janeiro 2011. 2, 3, 35

[8] Morvan, Yannick: *Acquisition, Compression and Rendering of Depth and Texture for Multi-View Video*. Eindhoven University of Technology, 2009. 2, 14

[9] Yang, Xiaohui, Ju Liu, Jiande Sun, Xinchao Li, Wei Liu e Yuling Gao: *Dibr based view synthesis for free-viewpoint television*. Em *3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON), 2011*, páginas 1–4, May 2011. 2, 35

[10] Suryanarayana, M. Muddala, Roger Olsson e Marten Sjostrom: *Disocclusion handling using depth-based inpainting*. The Fifth International Conferences on Advances in Multimedia, 2013. 3

[11] Gokturk, S. Burak, Hakan Yalcin e Cyrus Bamji: *A time-of-flight depth sensor – system description, issues and solutions*. Conference on Computer Vision and Pattern Recognition Workshop, 2004. 3, 15

[12] Tian, Dong, Po Lin Lai, Patrick Lopez e Cristina Gomila: *View synthesis techniques for 3d video*. Proc. SPIE, 7443:74430T–74430T–11, 2009. 3, 15

[13] Zhang, C., Z. Yin e D. Florencio: *Improving depth perception with motion parallax and its apllication in teleconference*. IEEE International Workshop on Multimedia Signal Processing, Outubro 2009. 4

[14] Mao, Yu, Gene Cheung, Antonio Ortega e Yusheng Ji: *Expansion hole filling in depth-image-based rendering using graph-based interpolation.* ICASSP, páginas 1859–1863, 2013. 5, 37, 41, 43, 52, 55, 58

[15] Mao, Yu, Gene Cheung e Yusheng Ji: *Graph-based interpolation for dibr-synthesized images with nonlocal means*. Em *Global Conference on Signal and Information Processing (GlobalSIP), 2013 IEEE*, páginas 451–454, Dec 2013. 5, 37, 38, 41, 55

[16] Mao, Y., G. Cheung e Y. Ji: *Image interpolation for dibr viewsynthesis using graph fourier transform*. Em *2014 3DTV-Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON)*, páginas 1–4, July 2014. 5, 37, 38, 41, 55

[17] Mao, Y., G. Cheung e Y. Ji: *On constructing z-dimensional dibr-synthesized images*. IEEE Transactions on Multimedia, 18(8):1453–1468, Aug 2016, ISSN 1520-9210. 5, 38

[18] Aharon, M., M. Elad e A. Bruckstein: *K-svd: An algorithm for designing overcomplete dictionaries for sparse representation*. IEEE Transactions on Signal Processing, 54(11):4311–4322, Nov 2006, ISSN 1053-587X. 6, 26, 39, 58

[19] Mairal, J., M. Elad e G. Sapiro: *Sparse representation for color image restoration*. Trans. Img. Proc., 17(1):53–69, janeiro 2008, ISSN 1057-7149. `http://dx.doi.org/10.1109/TIP.2007.911828`. 6

[20] Elad, Michael: *Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing*. Springer Publishing Company, Incorporated, 1st edição, 2010, ISBN 144197010X, 9781441970107. 6, 26

[21] Mairal, Julien, Francis Bach, Jean Ponce e Guillermo Sapiro: *Online dictionary learning for sparse coding*. Em *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, páginas 689–696, New York, NY, USA, 2009. ACM, ISBN 978-1-60558-516-1. `http://doi.acm.org/10.1145/1553374.1553463`. 6, 47

[22] Ribeiro, Danilo, Bruno Espinoza e Camilo Dorea: *Representação esparsa para preenchimento de buracos de expansão em sínteses de vistas virtuais baseada em profundidade*. Em *XXXV Simpósio Brasileiro de Telecomunicações e Processamento de Sinais (SBrT 2017) (SBrT 2017)*, São Pedro, Brazil, setembro 2017. 6

[23] Zhou, M., H. Chen, J. Paisley, L. Ren, G. Sapiro e L. Carin: *Non-parametric Bayesian dictionary learning for sparse image representations.* Em *NIPS*, páginas 2295–2303, 2009. 6, 27, 46, 47

[24] Ahn, I. e C. Kim: *Depth-based disocclusion filling for virtual view synthesis.* Internaciol Conference on Multimedia and Expo, páginas 109–114, 2012. 6

[25] Lucas, Laurent, Céline Loscos e Yannick Remion: *Multiview acquisition systems*, páginas 43–69. John Wiley & Sons, Inc., 2013, ISBN 9781118761915. `http://dx.doi.org/10.1002/9781118761915.ch3`. 8, 10, 11

[26] Leyvand, Tommer, Casey Meekhof, Yi Chen Wei, Jian Sun e Baining Guo: *Kinect identity: Technology and experience.* IEEE Computer, 2011. 11, 15

[27] Fehn., C.: *Depth-image-based rendering (dibr), compressio, and trasmission for a new approach on 3d-tv.* Proc. SPIE Stereoscopic Displays and Virtual Reality Systems XI, 2004. 11, 35, 36

[28] Schreer, O., P. Kauff e T. Sikor: *3D Video communication Algorithms, concepts and real-time systems in human centered communication.* John Wiley and Sons, 2nd edition, 2005. 11, 12

[29] Trucco, Emanuele e Alessandro Verri: *Introdutory Techniques for 3-D Computer Vision.* Prentice-Hall, Inc., 1998. 13

[30] Hartley, R. I. e A. Zisserman: *Multiple View Geometry in Computer Vision.* Cambridge Univeristy Press, 2nd edition, 2004. 14

[31] Scharstein, D. e R. Szeliski: *High-accuracy stereo depth maps using structured light.* IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2003), 1:195–202, Junho 2003. 15, 51

[32] Lange, R. e P. Seitz: *Solid-state time-of-flight range camera.* IEEE Journal of Quantum Electronics, 37(3):390–397, Mar 2001, ISSN 0018-9197. 15

[33] Muddala, Suryanaryana M.: *View rendering for 3DTV.* Tese de Doutoramento, Mid Sweden University, 2013. 15, 16

[34] Chan, S.C., H.Y. Shum e K. Ng: *Image-based rendering and synthesis.* IEEE Signal Processing Magazine, 2007. 17

[35] Zhang, L. e W.J. Tam: *Stereoscopic image generation based on depth images for 3d tv.* IEEE Transactions on Broadcasting, 2005. 17

[36] Sun, Jian, Nan Ning Zheng e Heung Yeung Shum: *Stereo matching using belief propagation.* IEEE Trans. Pattern Anal. Mach. Intell., 25(7):787–800, julho 2003, ISSN 0162-8828. `http://dx.doi.org/10.1109/TPAMI.2003.1206509`. 17

[37] Horng, Y. R., Y. C. Tseng e T. S. Chang: *Stereoscopic images generation with directional gaussian filter.* Em *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, páginas 2650–2653, May 2010. 18

[38] Köppel, M., M. B. Makhlouf, M. Müller e P. Ndjiki-Nya: *Temporally consistent adaptive depth map preprocessing for view synthesis.* Em *2013 Visual Communications and Image Processing (VCIP)*, páginas 1–6, Nov 2013. 19

[39] Du, Haiyang e Zhenjiang Miao: *Kinect depth maps preprocessing based on rgb-d data clustering and bilateral filtering.* Em *2015 Chinese Automation Congress (CAC)*, páginas 732–736, Nov 2015. 19

[40] Feng, C. e S. L. Dai: *Adaptive depth map enhancement based on joint bilateral filter.* Em *Proceedings of 2014 IEEE Chinese Guidance, Navigation and Control Conference*, páginas 2568–2571, Aug 2014. 19

[41] Shen, Y., J. Li e C. Lü: *Depth map enhancement method based on joint bilateral filter.* Em *2014 7th International Congress on Image and Signal Processing*, páginas 153–158, Oct 2014. 19

[42] Köppel, M., M. Ben Makhlouf e P. Ndjiki-Nya: *Optimized adaptive depth map filtering.* Em *2013 IEEE International Conference on Image Processing*, páginas 1356–1360, Sept 2013. 19

[43] Burrough, Peter A. e Rachael A. McDonnell: *Principles of Geographical Information Systems.* Oxford University Press, 1998. 21

[44] Kumar, M: *An adaptive zooming algorithm for images.* Tese de Mestrado, Thapar University, 2009. 21

[45] Gonzalez, Rafael C. e Richard E. Woods: *Digital Image Processing (3rd Edition).* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2006, ISBN 013168728X. 21

[46] Sangeetha, K.: *Combined structure and texture image inpainting algorithm for natural scene image completion.* Em *Journal of Information Engineering and Applications*, páginas 7–12, 2011. 23

[47] Starck, Jean Luc, Fionn Murtagh e Jalal Fadili: *Sparse Image and Signal Processing: Wavelets, Curvelets, Morphological Diversity.* Cambridge University Press, New York, NY, USA, 2010, ISBN 0521119138, 9780521119139. 23, 24

[48] Turek, J.S.: *Topics in Sparse Representation Modeling and Applications.* Technion - Israel Institute of Technology, Faculty of Computer Science, 2015. `https://books.google.com.br/books?id=SBRWAQAACAAJ`. 24, 25

[49] Aharon, Michal: *Overcomplete Dictionaries for Sparse Representation of Signals.* Tese de Doutoramento, Israel Institute of Technology, November 2006. 24, 39

[50] Nam, Sangnam, Mike E. Davies, Michael Elad e Rémi Gribonval: *The cosparse analysis model and algorithms.* CoRR, abs/1106.4987, 2011. 25

[51] Y.C. Pati, R. Rezaiifar e P.S. Krishnaprasad: *Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition.* Conference Record of The 27th Asilomar Conference on Signals, Systems and Computers, 1:40–44, 1993. 26, 38, 39, 58

[52] Mallat, S.G. e Zhifeng Zhang: *Matching pursuits with time-frequency dictionaries.* Trans. Sig. Proc., 41(12):3397–3415, dezembro 1993, ISSN 1053-587X. http://dx.doi.org/10.1109/78.258082. 26, 38

[53] Candès, E. e T. Tao: *The dantzig selector: Statistical estimation when p is much larger than n.* The Annals of Statistics, 35(6):2313–2351, 2007. 26

[54] Chen, Scott Shaobing, David L. Donoho e Michael A. Saunders: *Atomic decomposition by basis pursuit.* SIAM Rev., 43(1):129–159, janeiro 2001, ISSN 0036-1445. http://dx.doi.org/10.1137/S003614450037906X. 26, 38

[55] M. Elad, P. Milanfar e R. Rubinstein: *Analysis versus synthesis in signal priors.* Inverse Problems, 23(3):947–968, 2007. 26

[56] Rudelson, M. e R. Vershynin: *Sparse reconstruction by convex relaxation: Fourier and gaussian measurements.* Proceedings of the 40th Annual Conference on Information Sciences and Systems, páginas 207–212, Março 2006. 26

[57] Candès, E.J. e D.L. Donoho: *Recovering edges in ill-posed inverse problems: optimality of curvelet frames.* he Annals of Statistics, 30(3):784–842, Junho 2002. 26

[58] Do, M. N. e M. Vetterli: *Contourlets: a new directional multiresolution image representation.* Em *Conference Record of the Thirty-Sixth Asilomar Conference on Signals, Systems and Computers, 2002.*, volume 1, páginas 497–501 vol.1, Nov 2002. 26

[59] Fischer, S., G. Cristobal e R. Redondo: *Sparse overcomplete gabor wavelet representation based on local competitions.* IEEE Transactions on Image Processing, 15(2):265–272, Feb 2006, ISSN 1057-7149. 26

[60] Rubinstein, R., A. M. Bruckstein e M. Elad: *Dictionaries for sparse representation modeling.* Proceedings of the IEEE, 98(6):1045–1057, June 2010, ISSN 0018-9219. 26

[61] Engan, K., S. O. Aase e J. Hakon Husoy: *Method of optimal directions for frame design.* Em *1999 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings. ICASSP99 (Cat. No.99CH36258)*, volume 5, páginas 2443–2446 vol.5, 1999. 26

[62] Kreutz-Delgado, Kenneth, Joseph F. Murray, Bhaskar D. Rao, Kjersti Engan, Te Won Lee e Terrence J. Sejnowski: *Dictionary learning algorithms for sparse representation.* Neural Comput., 15(2):349–396, fevereiro 2003, ISSN 0899-7667. http://dx.doi.org/10.1162/089976603762552951. 27

[63] Leon-Garcia, Alberto: *Probability, Statistics, and Random Processes for Electrical Engineering.* Pearson/Prentice Hall, Upper Saddle River, NJ, third edição, 2008, ISBN 9780131471221 0131471228. 28

[64] Wheatstone, Charles: *On some remarkable, and hitherto unobserved, phenomena of binocular vision.* Philosophical Transactions of the Royal Society of London, 128:371–394, 1838. 32

[65] Wheelwright, G. W.: *Possibilities of stereoscopic motion pictures.* Journal of the Society of Motion Picture Engineers, 29(6):603–613, Dec 1937, ISSN 0097-5834. 32

[66] Javidi, Bahram e Fumio Okano: *Three-Dimensional Television, Video, and Display Technologies.* Springer Press, 2002. 32

[67] Lippman, Andrew: *Movie-maps: An application of the optical videodisc to computer graphics.* SIGGRAPH Comput. Graph., 14(3):32–42, julho 1980, ISSN 0097-8930. http://doi.acm.org/10.1145/965105.807465. 33

[68] Kang, Sing Bing, Yin Li, Xin Tong e Heung Yeung Shum: *Image-based rendering.* Found. Trends. Comput. Graph. Vis., 2(3):173–258, janeiro 2006, ISSN 1572-2740. http://dx.doi.org/10.1561/0600000012. 33

[69] Levoy, M. e P. Hanrahan: *Light field rendering.* Computer Graphics (SIGGRAPH), página 31–42, agosto 1996. 33

[70] Gortler, S. J., R. Grzeszczuk, R. Szeliski e M. F. Cohen: *The lumigraph.* Computer Graphics (SIGGRAPH), página 43–54, agosto 1996. 33

[71] Shum, H. Y. e L. W. He: *Rendering with concentric mosaics.* Computer Graphics (SIGGRAPH), página 299–306, agosto 1999. 34

[72] Chen, S. e L. Williams: *View interpolation for image synthesis.* Computer Graphics (SIGGRAPH), página 279–288, agosto 1993. 34

[73] Seitz, S. M. e C. M. Dyer: *View morphing.* Computer Graphics (SIGGRAPH), página 21–30, agosto 1996. 34

[74] Mark, W., L. McMillan e G. Bishop: *Post-rendering 3d warping.* Symposium on I3D Graphics, página 7–16, abril 1997. 34

[75] Shade, J., S. Gortler, L. W.He e R. Szeliski: *Layered depth images.* Computer Graphics (SIGGRAPH), página 231–242, julho 1998. 34

[76] Chang, C., G. Bishop e A. Lastra: *Ldi tree: A hierarchical representation for image-based rendering.* Computer Graphics (SIGGRAPH), página 291–298, agosto 1999. 34

[77] Wood, D. N., D. I. Azuma, K. Aldinger, B. Curless, T. Duchamp, D. H. Salesin e W. Stuetzle: *Surface light fields for 3D photography.* Computer Graphics (SIGGRAPH), página 287–296, julho 2000. 34

[78] McMillan Jr., Leonard: *An Image-based Approach to Three-dimensional Computer Graphics.* Tese de Doutoramento, Chapel Hill, NC, USA, 1997. UMI Order No. GAX97-30561. 34

[79] Scharstein, Daniel: *View Synthesis Using Stereo Vision.* Springer-Verlag, Berlin, Heidelberg, 1999, ISBN 3-540-66159-X. 34

[80] Pujades, S., F. Devernay e B. Goldluecke: *Bayesian view synthesis and image-based rendering principles*. Em *2014 IEEE Conference on Computer Vision and Pattern Recognition*, páginas 3906–3913, June 2014. 35

[81] Wanner, S. e B. Goldluecke: *Spatial and angular variational super-resolution of 4D light fields*. Em *ECCV*, página 608–621. Springer, 2012. 35

[82] Ham, B., D. Min, C. Oh, M. N. Do e K. Sohn: *Probability-based rendering for view synthesis*. IEEE Transactions on Image Processing, 23(2):870–884, Feb 2014, ISSN 1057-7149. 35

[83] Tauber, Z., Z. N. Li e M. S. Drew: *Review and preview: Disocclusion by inpainting for image-based rendering*. IEEE Transactions on Systems, Man and Cybernetics, Part C: Applications and Reviews, 37(4):527–540, 2007. 35

[84] Bertalmio, Marcelo, Guillermo Sapiro, Vicent Caselles e Coloma Ballester: *Image inpainting*. ACM Computer Graphics Proceedings (SIG- GRAPH), 2000. 36, 37

[85] Kang, Sung Ha, T. F. Chan e S. Soatto: *Inpainting from multiple views*. Em *3D Data Processing Visualization and Transmission, 2002. Proceedings. First International Symposium on*, páginas 622–625, 2002. 36

[86] Levin, A., A. Zomet e Y. Weiss: *Learning how to inpaint from global image statistics*. Em *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, páginas 305–312 vol.1, Oct 2003. 36

[87] Ružić, T. e A. Pižurica: *Context-aware patch-based image inpainting using markov random field modeling*. IEEE Transactions on Image Processing, 24(1):444–456, Jan 2015, ISSN 1057-7149. 36

[88] Efros, A. A. e T. K. Leung: *Texture synthesis by non-parametric sampling*. Em *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 2, páginas 1033–1038 vol.2, 1999. 36

[89] Criminisi, P. Perez e K. Toyama: *Region filling and object removal by exemplar-based image inpainting*. Em *IEEE TRANSACTIONS ON IMAGE PROCESSING*. IEEE, 2004. 36

[90] Gautier, Josselin, Olivier Le Meur e Christine Guillemot: *Depth-based image completion for view synthesis*. The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON),, 2011. 36

[91] Jawas, N. e N. Suciati.: *Image inpainting using erosion and dilation operation.* In International Journal of Advanced Science and Technology, 2013. 36

[92] Macchiavello, Bruno, Camilo Dorea, Edson M. Hung, Gene Cheung e Ivan Bajic: *Low-saliency prior for disocclusion hole filling in dibr-synthesized images.* IEEE Transactions on Multimedia, 2014. 37

[93] Silva, Ennio Willian Lima, Bruno Macchiavello e Camilo Dorea: *Preenchimento de buracos em síntese de vista baseado em mapa de profundidade*. XXXIV SIMPÓSIO BRASILEIRO DE TELECOMUNICAÇÕES - SBrT 2016, 2016. 37

[94] Gorodnitsky, I. F. e B. D. Rao: *Sparse signal reconstruction from limited data using focuss: a re-weighted minimum norm algorithm*. IEEE Transactions on Signal Processing, 45(3):600–616, Mar 1997, ISSN 1053-587X. 38

[95] Paisley, John e Lawrence Carin: *Nonparametric factor analysis with beta process priors*. Em *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, páginas 777–784, New York, NY, USA, 2009. ACM, ISBN 978-1-60558-516-1. http://doi.acm.org/10.1145/1553374.1553474. 45

[96] Sammon, J. W.: *A nonlinear mapping for data structure analysis*. IEEE Trans. Comput., 18(5):401–409, maio 1969, ISSN 0018-9340. http://dx.doi.org/10.1109/T-C.1969.222678. 49

[97] Telea, Alexandru: *An image inpainting technique based on the fast marching method*. Journal of Graphics, GPU, and Game Tools, páginas 23–34, 2004. 58