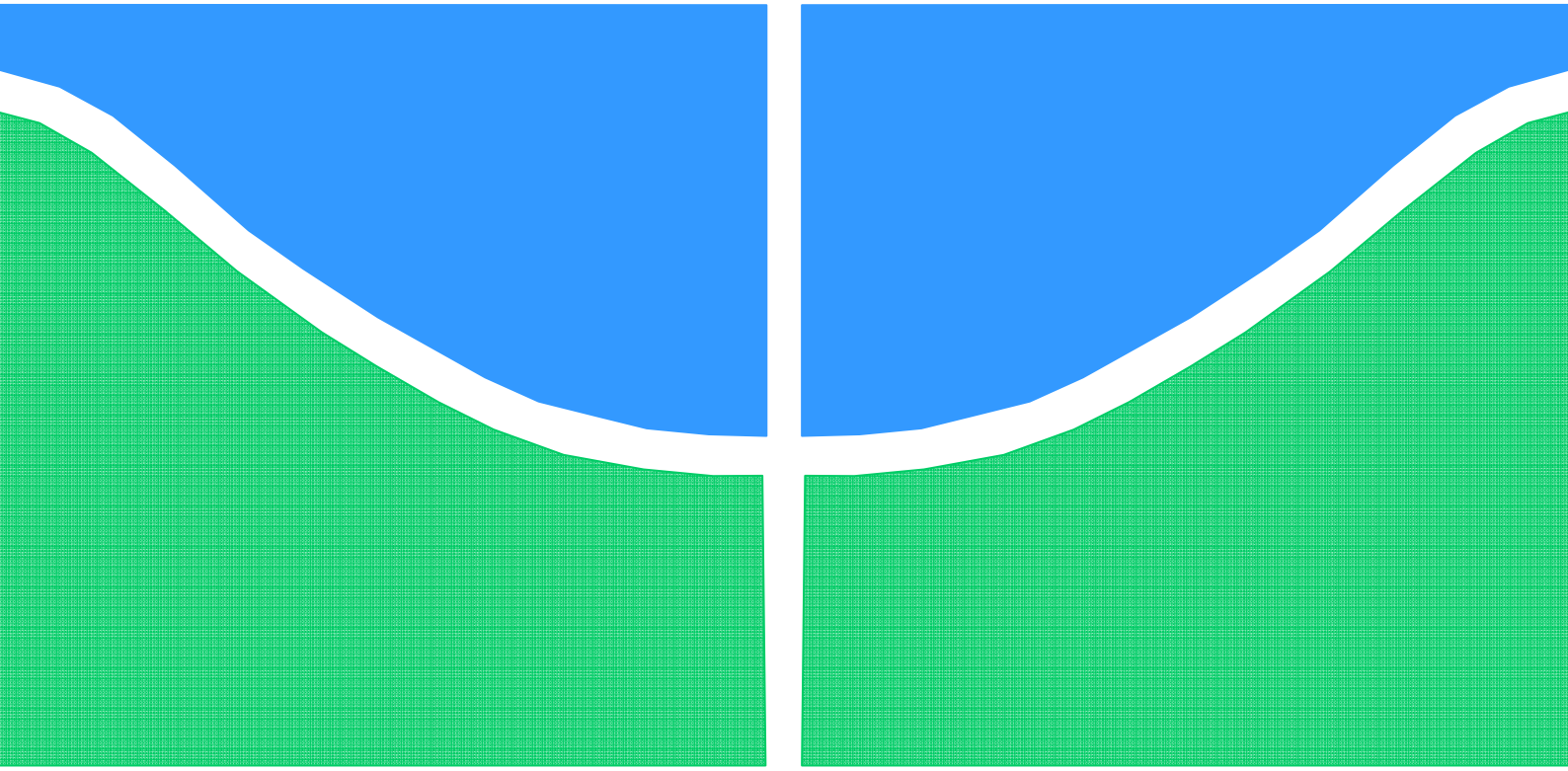

TRABALHO DE GRADUAÇÃO

ANÁLISE COMPARATIVA DE COMPRESSÃO DE IMAGENS *FISHEYE*, RETILÍNEAS E PANORÂMICAS

Augusto Cavalcante Valente

Brasília, 3 de setembro de 2010.

UNIVERSIDADE DE BRASILIA

FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASILIA
Faculdade de Tecnologia

TRABALHO DE GRADUAÇÃO

ANÁLISE COMPARATIVA DE COMPRESSÃO DE IMAGENS *FISHEYE*, RETILÍNEAS E PANORÂMICAS

Augusto Cavalcante Valente

Relatório submetido como requisito parcial para obtenção
do grau de Engenheiro Eletricista

Banca Examinadora

Prof. Ricardo Lopes de Queiroz, Ph.D.
UnB/ ENE (Orientador) _____

Prof. João Luiz Azevedo de Carvalho, Ph.D.
UnB/ ENE (Examinador interno) _____

Eng. Diogo Caetano Garcia, M.Sc.
(Examinador interno) _____

Dedicatória

À família (de sangue e de coração).

Augusto Cavalcante Valente

Agradecimentos

Agradeço ao orientador Ricardo Queiroz pelo incentivo, compreensão e puxões de orelha na hora certa e, de maneira especial, ao co-orientador Diogo Caetano, pela paciência infinita e pela perspicácia demonstrada ao propor diversas soluções durante o desenvolvimento deste trabalho.
A todos que cederam câmeras e equipamentos para os experimentos, Patrícia Mollo, Célia Cavalcante, Tito Barros e, em especial, o colega de laboratório Edson Mintsu.
Aos colegas de curso pelo companheirismo e pelos momentos de descontração que ajudaram a manter a sanidade.
Finalmente, este semestre foi um tanto quanto conturbado, e sem o apoio de familiares e amigos próximos teria sido difícil encará-lo até o final. Agradeço e dedico este trabalho a eles, portanto.

Augusto Cavalcante Valente

RESUMO

Neste trabalho é analisada a performance da compressão de imagens e vídeos nos formatos de projeção *fisheye*, retilínea e de panorama, levando em consideração a calibração de lentes *fisheye*. Esta é feita ajustando-se a distorção radial a um modelo de divisão polinomial com centro de distorção arbitrário. As análises de compressão são feitas com o padrão H.264/AVC. Na primeira análise, compara-se a eficiência entre mapear uma sequência do domínio *fisheye* para o retilíneo antes e depois de comprimi-la. Na segunda análise, compara-se, em um sistema de geração automática de panoramas a partir de um conjunto de imagens *fisheye*, a eficiência entre gerar o panorama com imagens recém-capturadas para depois comprimi-lo e comprimir as imagens recém-capturadas para depois formar o panorama. Resultados indicam que, para a primeira análise, a retificação seguida de compressão apresenta menos perdas, e para a segunda análise, a geração de panoramas seguida de compressão apresenta melhores resultados.

ABSTRACT

In this work, the compression performance of image and video in *fisheye*, rectilinear and panorama projections is analyzed, taking into account the *fisheye* lens calibration. The latter is achieved by fitting the radial distortion in a polynomial division model with arbitrary center of distortion. Image and video compression analysis are performed using the H.264/AVC standard. On a first experiment, a comparison is performed between the efficiency of mapping a sequence from its *fisheye* to a rectilinear domain before and after compressing it. On a second experiment, supposing an automated panorama generation system that uses a set of *fisheye* images, a comparison is performed between the efficiency of generating the panorama with recently captured images prior to compressing it and first compressing the images prior to generating the panorama. Results indicate that, for the first experiment, rectification before compression performs better, and for the second experiment, generating panoramas before compression also performs better.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SÍMBOLOS

Símbolos Latinos

f	Distância focal	[mm]

Símbolos Gregos

λ	Parâmetro de distorção no modelo DM	
θ	Ângulo de incidência	[º]

Subscritos

f	Projeção *fisheye*
p	Projeção perspectiva (retilínea)

Siglas

COD	Centro de distorção
DCT	Transformada de cossenos discreta
DLSR	*Digital single-lens reflex* (tipo de câmera)
DM	Modelo de divisão polinominal
DPCM	*Differential Pulse Coded Modulation*
FOV	Campo de visão
FRExt	*Fidelity Range Extensions*
H.264/AVC	Padrão de compressão de vídeo (*Advanced Video Coding*)
HVS	Sistema de visão humano
IDCT	Transformada inversa de cossenos discreta
ITU-T	Seção de padronização da União Internacional das Telecomuniações
JM	*Joint Model*
MPEG	*Motion Pictures Expert Group*
MSE	Erro médio quadrático
PSNR	*Peak Signal-to-Noise Ratio*
QP	Parâmetro de quantização
RDO	Otimização de taxa-distorção
RMSE	Raiz do erro médio quadrático
SAE	Soma dos erros absolutos
VCEG	*Video Coding Experts Group*

1 INTRODUÇÃO

Projeções do tipo *fisheye* são aquelas que representam uma porção significativa do ambiente em um plano ao abrir mão da conformidade de representação de linhas retas [1]. Como porção significativa entende-se um campo de visão que cobre mais do que 120º. A Figura 1.1 ilustra uma imagem típica na projeção *fisheye*, que, no caso, mapeia um campo de visão de 185º graus em um plano. Nota-se que os prédios retos na cena real são representados como curvos na imagem, e que os elementos perto das bordas têm um aspecto achatado.

Figura 1.1 – Imagem capturada com lente *fisheye* com campo de visão de 185º. Autor: Sunex, Inc.

Esta projeção será estudada neste trabalho e sua relação com técnicas de compressão de imagem e vídeo será analisada experimentalmente. O padrão de compressão adotado será o H.264/AVC [2], atualmente o estado da arte e um dos mais difundidos nos meios acadêmico e industrial. Este padrão utiliza técnicas de compressão que são aplicadas em blocos (quadrados ou retângulos) em cada quadro do vídeo [3]. Supõe-se que uma imagem retilínea seja melhor tratada por estas técnicas do que uma imagem *fisheye*. Projeção retilínea é aquela que mantém as retas da cena como retas na imagem, e é a mais comum em qualquer fotografia ou vídeo. Uma imagem em projeção *fisheye* possui em geral mais curvas e resolução espacial variável (o citado achatamento nas bordas).

Serão estudadas maneiras de se corrigir essa distorção na representação de linhas retas, e será analisado se a forma corrigida apresenta melhor performance em testes de compressão de vídeo. Os experimentos são realizados supondo-se um cenário prático em que se queira transmitir uma imagem adquirida em projeção *fisheye* (para se ter grande cobertura) mas que deve ser visualizada em projeção retilínea (para manter aparência mais familiar). Uma aplicação nesses moldes é descrita em [4], na qual é proposto a colocação de uma câmera *fisheye* em laterais de caminhões para eliminar seus pontos-cegos. Também vêm à mente aplicações de segurança e monitoramento como o sistema de câmeras de um prédio.

Outra técnica que será vista neste projeto é a formação de panoramas imersivos, uma aplicação que vem se tornando parte do cotidiano com a criação de serviços como Google Street View e Microsoft Streetside. Este é um campo no qual as imagens *fisheye* possuem bastante aplicação. As imagens panorâmicas são obtidas atualmente ao combinarem-se diversas fotos do mesmo local [5]. Caso seja usada a projeção *fisheye*, são necessárias menos fotos para cobrir todo o ambiente.

Será analisado se é mais vantajoso formar o panorama a partir das imagens recém-obtidas para então comprimi-lo e armazená-lo (ou transmiti-lo), ou então comprimir e armazenar (ou transmitir) as imagens separadas para formarem um panorama posteriormente.

Entre as contribuições de destaque neste projeto citam-se o desenvolvimento de um fluxo de trabalho e código para calibração manual e automática de lentes *fisheye* e os resultados práticos das comparações extensivas de processos envolvendo compressão de vídeo detalhadas há pouco.

Este trabalho está dividido da seguinte maneira. No capítulo 2, é feita uma rápida revisão de geometria óptica para, em seguida, serem abordados os detalhes relativos à projeção *fisheye* e o processo de calibração de lentes. Por fim, as etapas básicas da geração automática de panoramas são descritas. No capítulo 3, são primeiro introduzidos aspectos básicos da codificação de vídeo para depois descreverem-se as partes mais importantes do padrão H.264/AVC para este trabalho. No capítulo 4 são relatados os procedimentos experimentais adotados, sendo os seus resultados apresentados e discutidos no capítulo 5. O capítulo 6 apresenta conclusões finais e propostas de trabalhos futuros.

2 CALIBRAÇÃO DE LENTES E FORMAÇÃO DE PANORAMAS

2.1 FORMAÇÃO DA IMAGEM

Um sistema digital de aquisição de imagens fotográficas requer em geral os seguintes estágios: captura dos raios de luz, focalização da imagem, exposição a um sensor, conversão analógico-digital e pós-processamento[6]. As etapas estão organizadas em sequência na Figura 2.1, que ilustra ainda passos típicos do pós-processamento para se obter um formato comprimido.

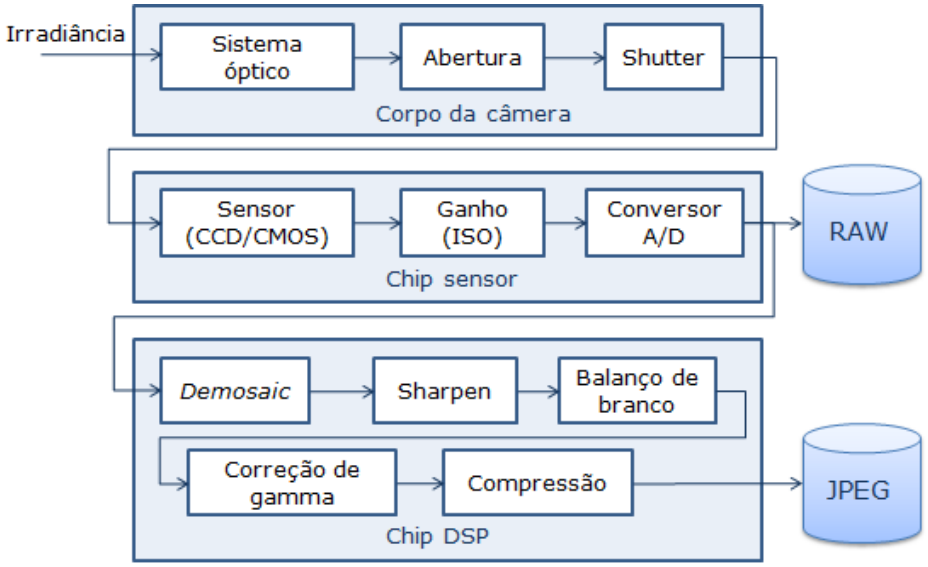

Figura 2.1 – Etapas da aquisição digital de imagens. (modificado [7])

A primeira idéia para se capturar a imagem de um objeto real é de simplesmente posicionar um material sensível à luz (filme) em sua frente. Porém, como todos os pontos do filme receberão luz de diversas direções, a imagem não será formada como esperado. A solução usualmente empregada é a câmara escura, cujo princípio de funcionamento, conhecido desde a Grécia Antiga, é ilustrado a Figura 2.2. O material fotossensível é colocado dentro de um compartimento escuro, que contém uma pequena abertura. Desta maneira, apenas raios propriamente direcionados atingem o filme.

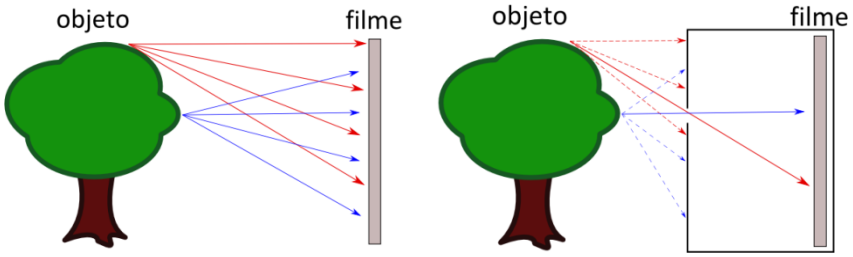

Figura 2.2 – Exposição sem barreira (esq.) e com barreira (dir.).

Para que a resolução obtida seja razoável, a abertura deve ser feita o menor possível. Em compensação, isto acarreta em menor exposição do filme à luz, além de indesejáveis padrões de difração [8]. O impasse é resolvido ao incorporar-se uma lente ao sistema, como na Figura 2.3. Uma lente é um dispositivo que produz raios de luz convergentes ou divergentes devido à refração. Aberturas maiores são permitidas, levando a uma maior iluminação disponível. Além disso, o efeito da profundidade de campo é introduzido. Apenas os raios de luz provenientes de uma determinada

profundidade são projetados como um ponto no filme, como os raios azuis na Figura 2.3. Raios originados em distâncias diferentes são projetados como manchas na forma da abertura, geralmente circular, como os raios vermelhos da mesma figura.

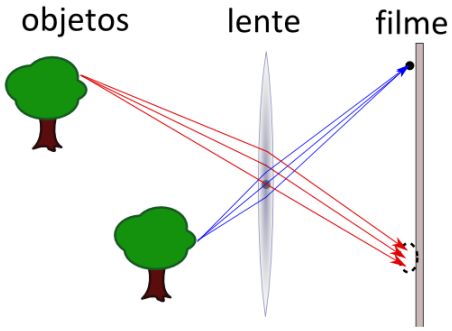

Figura 2.3 – Sistema com lente e formação de um objeto em foco (raios azuis) e fora de foco (vermelhos).

As lentes são descritas pela sua distância focal, usualmente expressa em milímetros. Essa é a distância na qual a imagem de um objeto muito distante é formada. Essa afirmação é válida para lentes cuja espessura é desprezível em relação à própria distância focal. Na Figura 2.4, a formação da imagem de um objeto de tamanho y a uma distância a por uma lente convexa é ilustrada. Neste caso, o objeto é formado com altura y' a uma distância q, a partir do ponto focal e b, a partir da lente. De semelhanças de triângulos, tem-se que:

$$m = \frac{y'}{y} = \frac{f}{p} = \frac{b}{a}, \tag{2.1}$$

onde m é o aumento da lente. Substituindo $p = a - f$, obtém-se a conhecida equação:

$$\frac{1}{f} = \frac{1}{a} + \frac{1}{b}. \tag{2.2}$$

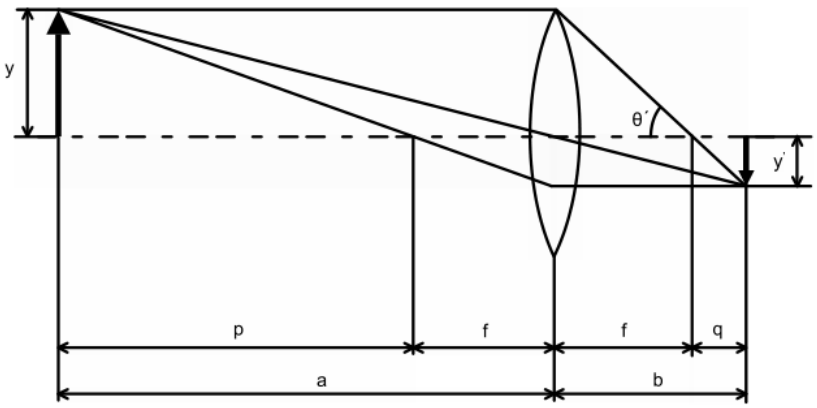

Figura 2.4 – Formação de um objeto distante [9].

Na prática as lentes fotográficas são compostas por várias camadas que atuam como lentes superpostas, a fim de minimizar efeitos indesejáveis como aberração cromática e vinheta. Neste caso, a Eq. (2.2) serve apenas como aproximação e deve ser expandida para se adaptar à geometria utilizada.

A distância focal determina o campo de visão da lente (FOV), que é uma medida da porção da cena que é capturada. À medida que a distância focal diminui, o campo de visão aumenta, pois o aumento m diminui, como visto na Eq. (2.1). A Tabela 2.1 mostra a distância focal para diferentes tipos de lentes. Nota-se que o campo de visão também depende obviamente do tamanho do filme ou sensor utilizado.

As lentes da Tabela 2.1 são classificadas como retilíneas, i.e., elas reproduzem linhas retas da cena como linhas retas na imagem. Caso seja capturado um FOV maior que 100°, é inevitável a ocorrência de distorções de perspectiva (esticamento) nos cantos da imagem. Isso pode ser explicado intuitivamente pelo fato de se estar mapeando o interior de um hemisfério em um plano, e, em seguida,

visualizando o resultado muito de perto. A imagem retilínea cobrirá, na visão do observador, um campo de visão menor do que aquele que foi capturado, causando estranheza na proporção de alguns objetos da cena.

Tabela 2.1 – Distância focal e campo de visão de alguns tipos de lentes para um filme de 35 mm.

Lente	f (mm)	FOV
Super grande angular	14 a 20	114 a 94°
Grande angular	24 a 35	84 a 63°
Normal	50	47°
Teleobjetiva	85 a 300	28 a 8°
Super teleobjetiva	400 a 1000	6 a 3°

De fato, aspectos físicos do projeto de lentes tornam difícil sua construção para ângulos de visão maiores que 120° e projeção retilínea [1]. As lentes do tipo *fisheye* são resultado de processos de construção e mapeamentos distintos para superar este limite.

2.2 LENTES *FISHEYE*

De acordo com [10], o termo "*fisheye*" (olho-de-peixe) foi cunhado por Robert Wood em 1911, ao descrever uma câmera *pinhole* preenchida com água capaz de capturar imagens com FOV de 180°. Tal sistema tinha o funcionamento baseado no fenômeno da janela de Snell, em que um observador submerso enxerga o ambiente da superfície por um cone de luz com ângulo de abertura de 97°, como na Figura 2.5. A refração na fronteira ar/água faz com que raios incidentes a 90° com a normal entrem na água com pouco menos de 50°, efetivamente comprimindo o campo de visão da superfície. Esta é a idéia básica da geometria de lentes *fisheye*.

Figura 2.5 – Visualização da janela de Snell. Autora: Suzy Walker.

O primeiro protótipo de lente *fisheye* foi desenvolvido no Escritório de Meteorologia do Reino Unido em 1924, demonstrando o cunho inicialmente puramente científico destes modelos. A técnica denominada fotografia hemisférica utiliza essas lentes e possui aplicações em meteorologia, astronomia (medição de posição de astros), biologia (acompanhamento do percurso do Sol em florestas) e outras áreas.

As lentes retilíneas citadas na seção anterior mapeiam os raios de luz como no diagrama na Figura 2.6(a). Para uma projeção perspectiva, portanto:

$$R_p = f \cdot tan(\theta)\,, \tag{2.3}$$

onde R_p é a posição radial de um ponto na imagem relativo ao centro, θ é o ângulo entre um ponto na cena e o eixo óptico e f é a distância focal. Nota-se que R_p diverge à medida que se aproxima de $\theta = 90°$.

Alternativamente, uma lente pode realizar o mapeamento de acordo com:

$$R_{eqd} = f \cdot \theta\,, \tag{2.4}$$

conforme o diagrama na Figura 2.6(b), onde R_{eqd} é a posição radial de um ponto. Esta representação é meramente didática, visto que o filme ou sensor não se curvam. Tal representação permite observar que haverá um espaço maior para se mapear os ângulos de incidência, gerando um campo de visão maior. O mapeamento da Eq. (2.4) é denominado equidistante, e favorece a medição de ângulos na imagem, pois incrementos em θ geram incrementos proporcionais em R_{eqd}.

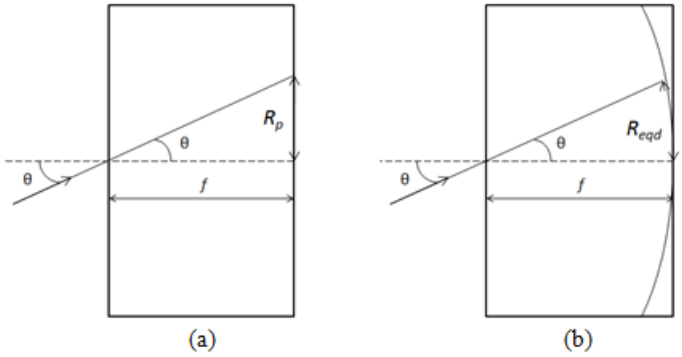

Figura 2.6 – Comparação entre projeções perspectiva e *fisheye* equidistante. [11]

Também são comuns em lentes *fisheye* as projeções equisólida e estereográfica. No primeiro caso, ângulos sólidos[1] iguais garantem áreas iguais, comprimindo mais a periferia da imagem:

$$R_{eqs} = 2 \cdot f \cdot \text{sen}\,(\theta/2)\,. \tag{2.5}$$

No caso da projeção estereográfica, garante-se um aspecto visual mais agradável ao comprimir menos as regiões longe do centro e preservar formas circulares [12]:

$$R_{est} = 2 \cdot f \cdot \tan\,(\theta/2)\,. \tag{2.6}$$

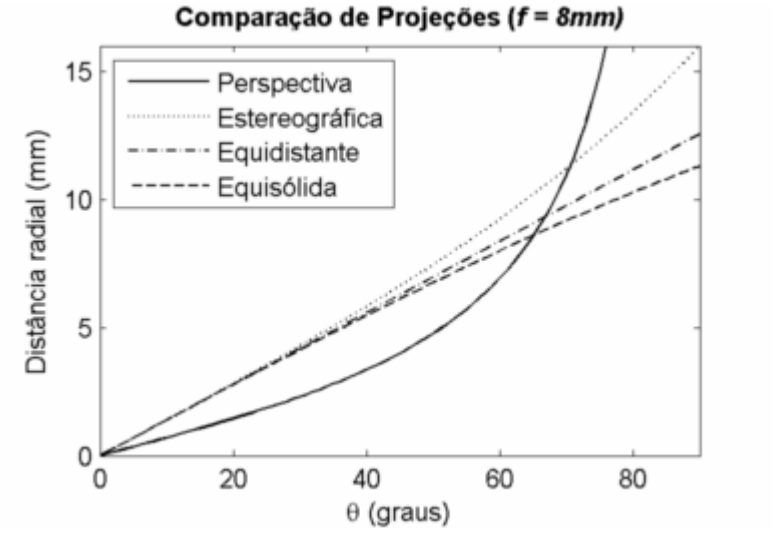

Figura 2.7 – Comparação entre projeção perspectiva (retilínea) e outras.

[1] Medido em esferorradianos, corresponde à área coberta por um objeto na superfície de uma esfera de raio unitário. O ângulo sólido de um hemisfério é 2π, por exemplo.

A Figura 2.7 mostra uma comparação entre a distância radial atingida com as quatro projeções citadas. A distância focal foi fixada em 8 mm. Como esperado, a projeção perspectiva (retilínea) diverge antes de conseguir mapear todo o hemisfério. A diferença entre as projeções equidistante e equisólida é sutil, enquanto a estereográfica distribui mais o campo de visão.

Na Figura 2.8, um padrão xadrez é distorcido com cada projeção *fisheye*, mantendo-se a mesma distância focal ajustada para melhor visualização. Nota-se que a projeção estereográfica provoca o menor achatamento da periferia da imagem, e as projeções equidistante e equisólida são dificilmente discernidas.

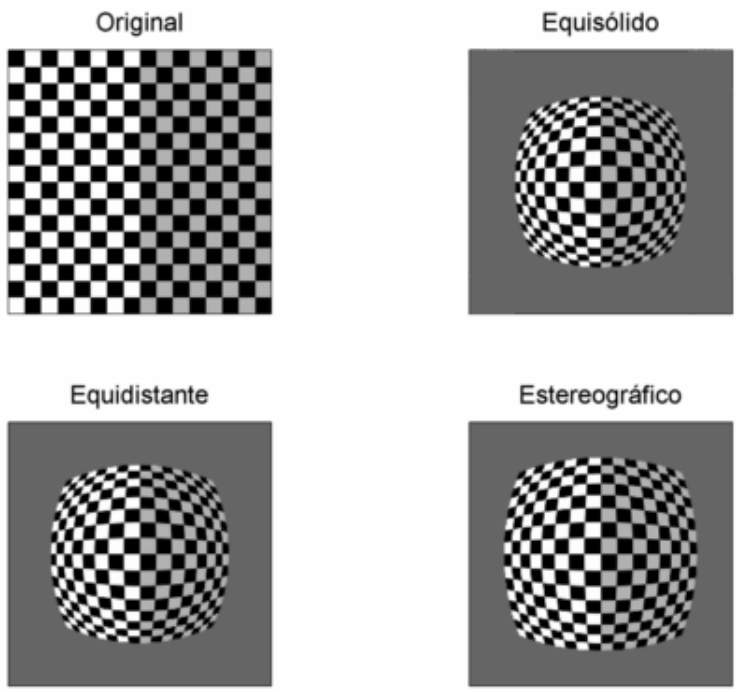

Figura 2.8 – Diferença entre projeções ao distorcer um padrão xadrez com mesma distância focal.

Como já mencionado, o campo de visão capturado depende das dimensões do filme ou sensor utilizado. Historicamente, o formato de filme mais popular é o de 35 mm (ISO 1007), batizado de 135 pela Kodak, e que possui dimensões de 36 x 24 mm e diagonal de 43,3 mm [6]. Os sensores de câmeras digitais SLR possuem diagonal em geral 20 a 40% menor, o que é denominado *crop factor*. Este fator é expresso como a razão entre a diagonal do filme 135 e a diagonal do sensor, sendo 1,3, 1,5 e 1,6 e valores comuns para este fator.

As lentes *fisheye* costumam ser projetadas para o filme 135 e são agrupadas em três classes: *full-frame*, quando o círculo da imagem circunscreve o *frame* e o campo de visão máximo se estende na diagonal; *quasi-fisheye*, também chamado *drum shot*, quando o círculo é parcialmente inscrito, deixando laterais de fora; e *circular*, quando o círculo é totalmente inscrito. A Figura 2.9 ilustra a cobertura de diferentes classes de lentes em um filme de 35 mm. A Figura 2.10 apresenta cópias de projetos de lentes patenteadas nos Estados Unidos e arquivadas na base de dados LensVIEW™ (*Optical Data Solutions*, NY). Os aspectos construtivos não serão abordados neste trabalho, mas nota-se que há um grupo de lentes frontal com grande poder de refração responsável pela forte curvatura dos raios, seguido de combinações de filtros para minimizar distorções como as já mencionadas aberração cromática e vinheta.

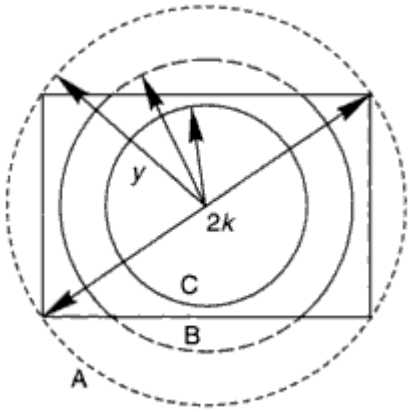

Figura 2.9 – Diferentes coberturas de classes de *fisheye* em um filme 135: *full-frame* (A), *quasi-fisheye* (B) e circular (C). [1]

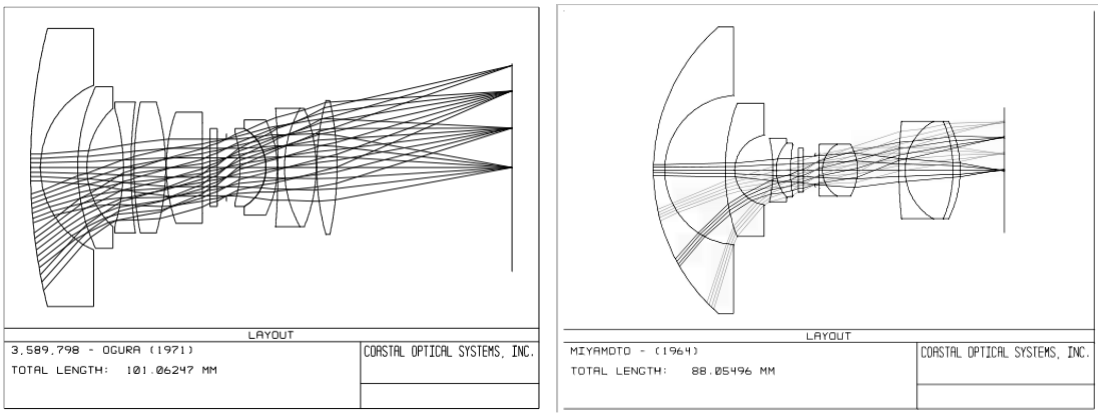

Figura 2.10 – Projeto de lentes *fisheye full-frame* de 16 mm (esq.) e circular de 8 mm (dir.). [10]

2.3 MODELAMENTO DA DISTORÇÃO

O termo "distorção" é usado nesta seção para descrever o desvio entre os mapeamentos *fisheye* já descritos e o mapeamento retilíneo. Em outro contexto, poderia ser discutido o desvio do mapeamento de uma lente *fisheye* real para o mapeamento teórico esperado.

A distorção examinada é essencialmente radial e não-linear. Por radial entende-se que pontos no plano da imagem *fisheye* estão deslocados de sua posição ideal no modelo retilíneo ao longo de um eixo radial coplanar partindo do centro de distorção (COD), que não é necessariamente o centro do círculo de imagem. O fato de ser não-linear implica que ela não pode ser expressa na forma $\hat{x} = ax + cy + m$ e $\hat{y} = dx + ey + n$, ou seja, uma transformação linear no espaço 2-D. Um dos efeitos da distorção *fisheye* é a imagem possuir maior resolução espacial na região próxima ao COD, com um decréscimo gradual não-linear à medida que se aproxima da periferia.

A fim de posteriormente cancelar essa distorção, é necessário modelar a relação entre a distância radial de um ponto na imagem retilínea e na imagem *fisheye*. O processo de cancelamento da distorção é denominado retificação. Vários modelos estão disponíveis na literatura e são classificados em polinomiais e não-polinomiais.

2.3.1 MODELOS POLINOMIAIS

O uso de polinômios para modelar a distorção radial é uma prática padrão [13]. São atrativos do ponto de vista computacional por não envolverem o uso de funções trigonométricas, embora o uso de *look-up tables* para estes casos minimiza esta vantagem. Uma desvantagem é a dificuldade de se inverter os modelos, o que é útil no processo de mapeamento, como será visto depois.

O modelo mais comum é um polinômio com graus ímpares:

$$r_f = r_p + k_1 r_p^3 + \cdots + k_n r_p^{2n+1} + \cdots \tag{2.7}$$

onde r_f é a distância radial do ponto no espaço *fisheye*, r_p a distância radial no espaço perspectivo (retilíneo) e k_n são os coeficientes do modelo. A transformação feita do espaço corrigido para o distorcido é denominada mapeamento inverso. A transformação direta para um modelo de quinta ordem pode ser aproximada por:

$$r_p = r_f - r_f\left(\frac{k_1 r_f^2 + k_2 r_f^4 + {k_1}^2 r_f^4 + {k_2}^2 r_f^8 + 2k_1 k_2 r_f^6}{1 + 4k_1 r_f^2 + 6k_2 r_f^4}\right). \tag{2.8}$$

Na Figura 2.11(a), é mostrada uma medida da qualidade do ajuste do modelo polinomial a uma curva de distorção *fisheye* equidistante. A medida adotada é a raiz do erro quadrático médio (RMSE), dada por:

$$RMSE = \sqrt{\frac{\int_0^{\max(r_p)} (r_{eqd} - r_f)^2}{\max(r_p)}} \tag{2.9}$$

onde r_{eqd} é a distância radial de referência (*fisheye* equidistante). Observa-se que é possível modelar satisfatoriamente a distorção introduzida por uma lente com FOV de até 140° com um modelo polinomial de pelo menos nona ordem. Porém, para um FOV acima disso, o erro aumenta consideravelmente. É mostrado em [14] que uma perturbação do tipo *ripple* ocorre mesmo para aproximações de 17ª ordem.

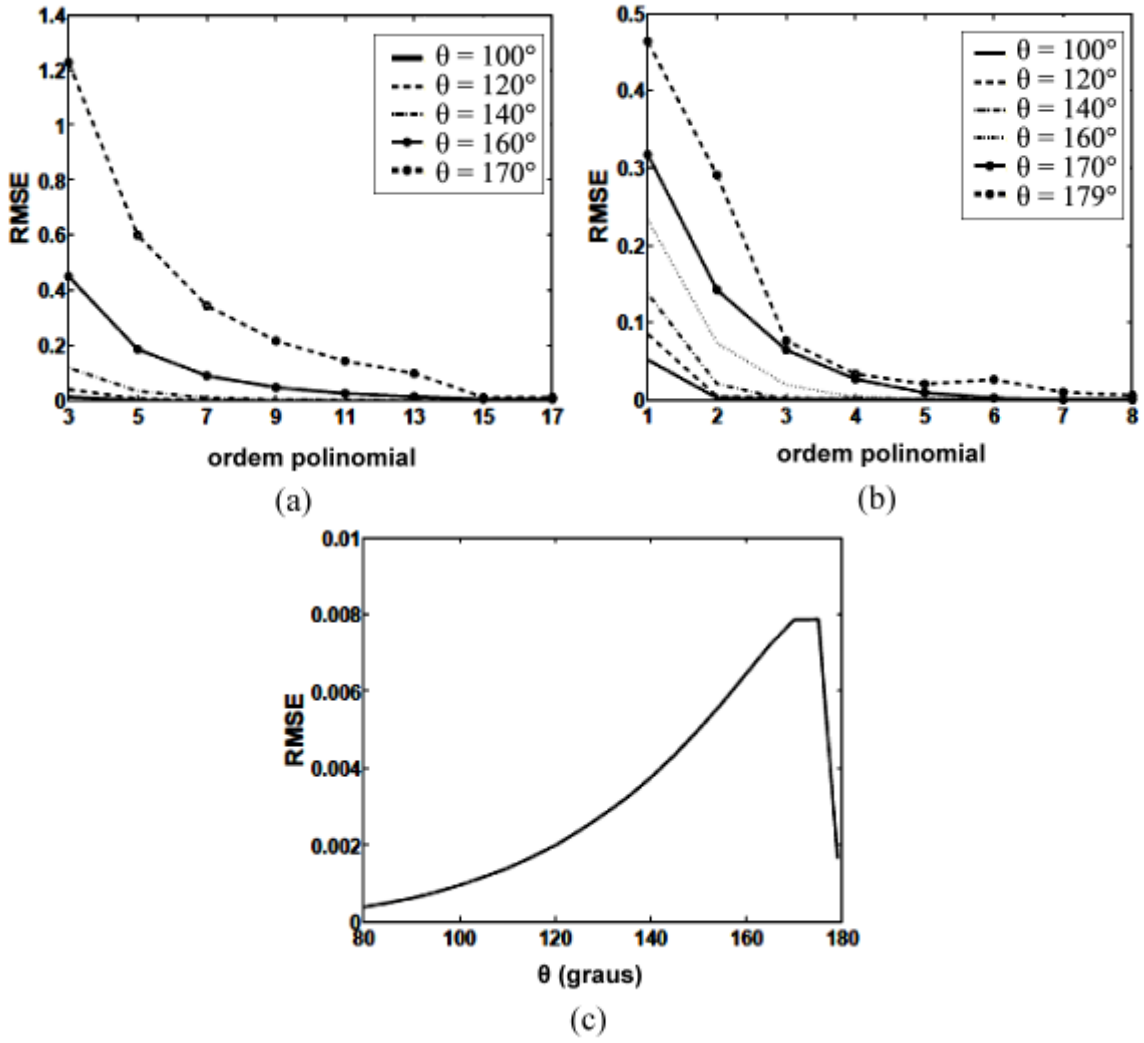

Figura 2.11 – Comparação no ajuste dos modelos a) polinomial simples, b) PFET e c) divisão. [14]

Em [15], os autores propõem a utilização de um polinômio com graus pares e ímpares, além de um termo livre, e o chamam de *Polynomial Fisheye Transform* (PFET):

$$r_f = k_0 + k_1 r_p^1 + \cdots + k_n r_p^n + \cdots. \tag{2.10}$$

Como pode ser visto na Figura 2.11(b), seu desempenho é superior ao do modelo polinomial tradicional. Com um polinômio de quinta ordem, o PFET alcança um RMSE abaixo de 0,03 para um FOV de até 179°.

Em [16], foi proposto o modelo de divisão (DM), dado por:

$$r_p = \frac{r_f}{1 + k_1 r_f^2 + \cdots + k_n r_f^{2n} + \cdots}. \tag{2.11}$$

Apesar da similaridade com a inversão do modelo polinomial simples, este é um modelo diferente que tenta aproximar a curva de distorção real. Ele é portanto um modelo naturalmente invertido, mapeando pontos do domínio distorcido de r_f para o domínio retilíneo de r_p. O modelo é usualmente aplicado com apenas um parâmetro de distorção, sendo simplificado como

$$r_p = \frac{r_f}{1 + \lambda r_f^2} \tag{2.12}$$

e o modelo direto

$$r_f = \frac{1 - \sqrt{1 - 4\lambda r_p^2}}{2\lambda r_p} \tag{2.13}$$

onde λ é o parâmetro de primeira ordem do modelo. Na Figura 2.11(c), nota-se que o modelo de divisão supera os demais, mantendo a RMSE abaixo de 0,08 para todos os FOV testados. Este modelo foi o escolhido para o sistema de calibração implementado neste trabalho por obter um excelente desempenho com apenas um parâmetro. Isso pode ser visto na Figura 2.12, que ilustra uma comparação entre o ajuste do modelo DM e o modelo PFET de grau 5 a uma curva *fisheye* equidistante para FOV = 179°. O modelo polinomial tradicional apresenta grande oscilação neste caso, e não foi incluído na comparação. O modelo PFET também apresenta oscilação, porém menos acentuada. O ajuste foi feito com a função `nlinfit` disponível na *toolbox* de estatística do ambiente MATLAB® 7.9, sendo obtido um RMSE de $4{,}8\times10^{-4}$ para o modelo DM, e de $5{,}1\times10^{-3}$ para o modelo PFET, condizente com os gráficos da Figura 2.11.

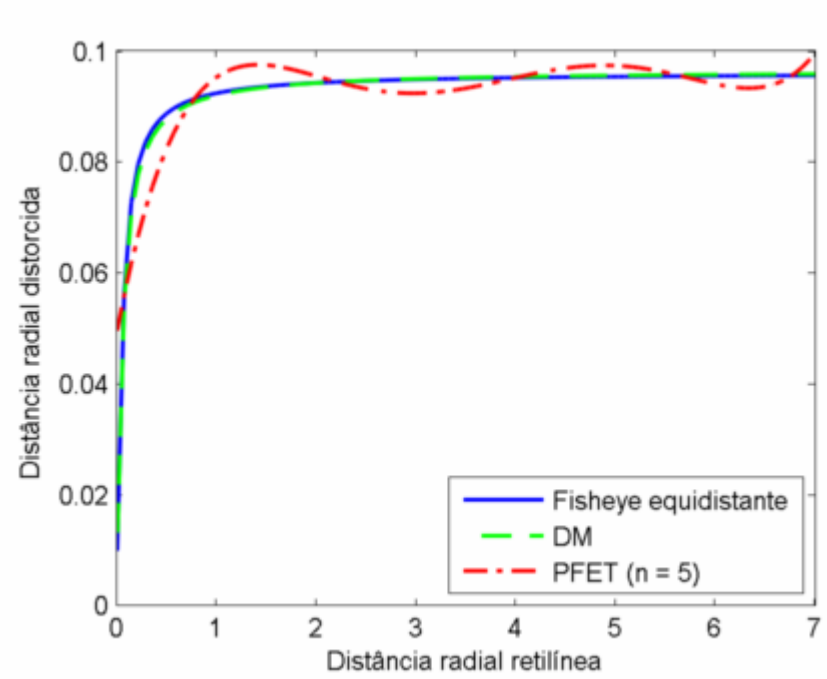

Figura 2.12 – Ajuste dos modelos DM e PFET para FOV = 179°.

2.3.2 MODELOS NÃO-POLINOMIAIS

Modelos não-polinomiais são obtidos a partir da modelagem geométrica do sistema óptico da lente. Envolvem funções transcendentais, que são menos computacionalmente eficientes, mas que podem ser utilizadas apenas para gerar *look-up tables*. Modelos inversos são geralmente possíveis de serem obtidos analiticamente, facilitando o mapeamento na correção das imagens.

A modelagem mais óbvia é a combinação da Eq. (2.3) com as Eqs. (2.4) a (2.6), para obter a relação entre a distância radial perspectiva e a distância radial em cada modelo *fisheye*. No modelo inverso, resolve-se para R_p, e, no modelo direto, resolve-se para R_{eqd}, R_{eqs} ou R_{est}, de acordo com a projeção. Para o caso equidistante, o mapeamento inverso e direto são, respectivamente:

$$\begin{aligned} R_p &= f \tan\left(\frac{R_{eqd}}{f}\right) \\ R_{eqd} &= f \tan^{-1}\left(\frac{R_p}{f}\right). \end{aligned} \tag{2.14}$$

Para a projeção equisólida, tem-se, analogamente:

$$\begin{aligned} R_p &= f \tan\left(2 \operatorname{sen}^{-1}\left(\frac{R_{eqs}}{2f}\right)\right) \\ R_{eqs} &= 2 f \operatorname{sen}\left(\frac{1}{2}\tan^{-1}\left(\frac{R_p}{f}\right)\right) \end{aligned} \tag{2.15}$$

e, finalmente, para a projeção estereográfica:

$$\begin{aligned} R_p &= f \tan\left(2 \tan^{-1}\left(\frac{R_{est}}{2f}\right)\right) \\ R_{est} &= 2 f \tan\left(\frac{1}{2}\tan^{-1}\left(\frac{R_p}{f}\right)\right). \end{aligned} \tag{2.16}$$

Devido a imperfeições na fabricação das lentes, essas projeções não serão exatamente seguidas na formação da imagem e, portanto, o seu único parâmetro, que é a distância focal, também necessita ser calibrado.

Alternativamente, em [15], os autores propõem, além do já citado PFET, o modelo não-polinomial denominado *Fish-Eye Transform* (FET). A forma direta e inversa são, respectivamente

$$\begin{aligned} R_f &= s \ln\left(1 + \lambda R_p\right) \\ R_p &= \frac{e^{R_f/s} - 1}{\lambda}. \end{aligned} \tag{2.17}$$

onde s é um fator de escala e λ controla a quantidade de distorção. O modelo é baseado na observação de que a imagem *fisheye* possui maior resolução na região próxima ao COD, sofrendo uma queda gradual ao aproximar-se a periferia. É uma tentativa de aproximar a magnificação cortical, que é a distribuição de receptores visuais na retina.

Outros modelos interessantes também estão disponíveis na literatura. O modelo proposto por Devernay e Faugeras em [17], cujas formas direta e inversa são respectivamente dadas por

$$\begin{aligned} R_f &= \frac{1}{\omega}\tan^{-1}\left(2R_p \tan\frac{\omega}{2}\right) \\ R_p &= \frac{\tan\left(R_f \omega\right)}{2\tan(\omega/2)} \end{aligned} \tag{2.18}$$

onde ω é o FOV aparente da lente, é baseado em um modelo óptico simples da lente *fisheye*, e pode ser calibrado para seus diferentes tipos. Já o modelo proposto por Pers e Kovacic em [18], com formas direta e inversa dadas respectivamente por

$$R_f = f \ln\left(\frac{R_p}{f} + \sqrt{1 + \frac{R_p^2}{f}}\right)$$
$$R_p = -\frac{f}{2}\frac{e^{-\frac{2R_f}{f}} - 1}{e^{-\frac{R_f}{f}}} \tag{2.19}$$

é baseado na abstração de que uma imagem com grande FOV é composta por várias imagens obtidas com a mesma câmera imaginária que gira a partir de um ponto fixo.

Estes modelos foram pouco testados pois logo se constatou que era necessária a calibração de seus parâmetros para que fossem utilizados com imagens geradas com lentes de projeções e campos de visão arbitrários. Sempre que se fez necessária a utilização de um modelo não-polinomial (e.g., na geração de panoramas), optou-se pelas Eqs. (2.14) a (2.16), de acordo com a projeção mencionada na especificação da lente.

2.4 CALIBRAÇÃO E CORREÇÃO

No contexto da visão computacional, a calibração de câmeras consiste em achar o mapeamento entre o espaço 3-D e o plano da imagem. Este mapeamento é separado em duas transformações distintas, os parâmetros de calibração externos e internos. A primeira trata do deslocamento entre a origem do espaço 3-D e o sistema de coordenadas da câmera, composto pelo centro da câmera e o seu direcionamento. A segunda transformação trata da forma com que pontos no espaço 3-D são mapeados em pontos no plano da imagem no sistema de coordenadas da câmera, devido à distância focal, ao formato da imagem e o ponto principal da câmera. Esta seção descreve a calibração de parâmetros internos, em particular a distorção radial. A componente radial domina a distorção provocada por uma lente [19] e sua estimação e correção são alvo de várias publicações.

Os métodos de estimação da distorção radial podem ser divididos em dois ramos: os que realizam correspondências entre mais de uma imagem e muitas vezes são auto-calibráveis e os que utilizam apenas uma imagem mas dependem da presença de alguma característica específica. Do primeiro grupo, cita-se o trabalho de Xiong e Turkowski [20], voltado para lentes *fisheye*, em que uma série de imagens com sobreposição é registrada modelando-se a distorção radial com um polinômio cúbico. Os autores afirmam, porém, que métodos auto-calibráveis são em geral bastante instáveis caso não se use nenhum parâmetro conhecido da lente.

Entre os métodos que utilizam só uma imagem, cita-se o de Hartley e Kang [21], em que um modelo sem parâmetros é estimado a partir da imagem de um padrão de calibração. Devernay e Faugeras [17] detectam segmentos de linhas 3-D em uma imagem e os ajustam em um modelo de distorção iterativamente. A detecção de linhas deve ser feita com precisão sub-pixel para que um modelo seja satisfeito. Este método explora uma propriedade fundamental: uma lente segue a projeção perspectiva se e somente se a projeção de toda linha reta no espaço 3-D é uma linha reta no plano da imagem. Consequentemente, a distorção radial pode ser estimada ao medir-se o quanto as linhas retas estão distorcidas na imagem. A desvantagem óbvia deste método é que a cena contenha linhas retas, mas isso é válido para a maioria dos ambientes construídos pelo homem. Wang et al [22] utilizam esta proposição aliada ao modelo de distorção DM para reduzir o problema ao ajuste de arcos de círculos às linhas distorcidas na imagem. Pode-se mostrar que a projeção de uma linha reta tridimensional no espaço de DM é um arco de círculo no plano da imagem, portanto a calibração se resume a ajustar círculos nas curvas da imagem que deveriam estar retas.

É também necessário ressaltar a importância da estimação do centro de distorção (COD) no processo de calibração. É comum na literatura assumir que o COD é conhecido e coincide com o centro da imagem. Para lentes com FOV reduzido, esta simplificação é válida, mas não quando se trabalha com campos de visão extensos. O COD pode estar deslocado por uma série de fatores como descentralização da lente em relação ao sensor no corpo da câmera, desalinhamento dos elementos ópticos que compõem a lente, ou o recorte da imagem visualizada pelo tamanho do sensor ou no pós-

processamento. Uma correção aplicada com o COD mal estimado pode acarretar em nova distorção radial, além de uma componente tangencial, como na Figura 2.13.

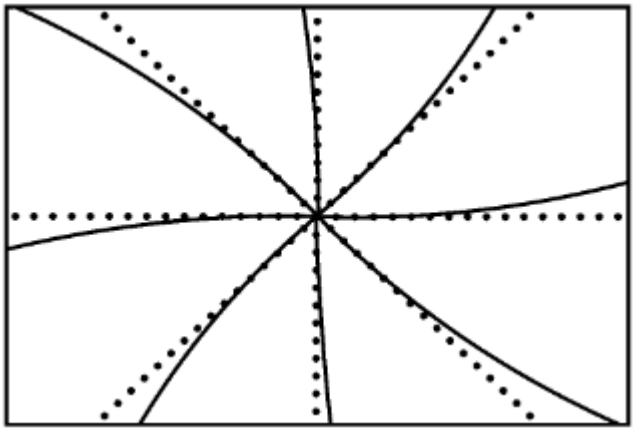

Figura 2.13 – Ilustração de distorção tangencial. [13]

A correção da distorção radial é o processo pelo qual pontos na imagem com projeção *fisheye* são transformados para pontos em uma imagem com projeção retilínea. Este é um processo feito basicamente no pós-processamento de uma imagem, uma vez que um sistema óptico capaz de realizar esta correção seria bastante caro ou provavelmente impossível de se realizar devido a limitações físicas dos materiais.

Uma maneira simples de se realizar a correção é aplicar um modelo na forma direta aos pontos da imagem distorcida, mapeando-os para pontos na imagem retilínea. Como imagens digitais são compostas por elementos (*pixels*) com posições discretas e o mapeamento é contínuo, serão geradas posições finais não inteiras, enquanto outras posições não serão mapeadas. Assim, a correção na forma direta exige a interpolação de pontos em posições inteiras a partir dos pontos já mapeados.

Outra solução é realizar o mapeamento inverso, ou seja, para cada elemento da imagem final corrigida, buscar sua posição original na imagem distorcida. Este mapeamento também é contínuo e gerará posições iniciais não inteiras, mas que agora podem ser interpoladas pelos valores conhecidos de posições vizinhas. Nota-se que é útil contar tanto com a forma direta do mapeamento, para estimar as dimensões da imagem corrigida, quanto com a forma inversa, para realizar o mapeamento da correção de fato. A Figura 2.14 ilustra o resultado do mapeamento direto e inverso na correção de uma imagem sintética. A imagem original, na parte (a), foi gerada com o modelo DM. Na parte (b), a imagem é corrigida com aplicação direta da Eq. (2.11) e simplesmente arredondando os resultados. Isso gera vários *pixels* mapeados no mesmo lugar, deixando outros vazios. Na parte (c), a Eq. (2.11) é usada apenas para estimar as dimensões da imagem corrigida, e o mapeamento é feito das posições dessa imagem para a imagem original com a Eq. (2.12) e interpolação bicúbica, gerando o resultado esperado.

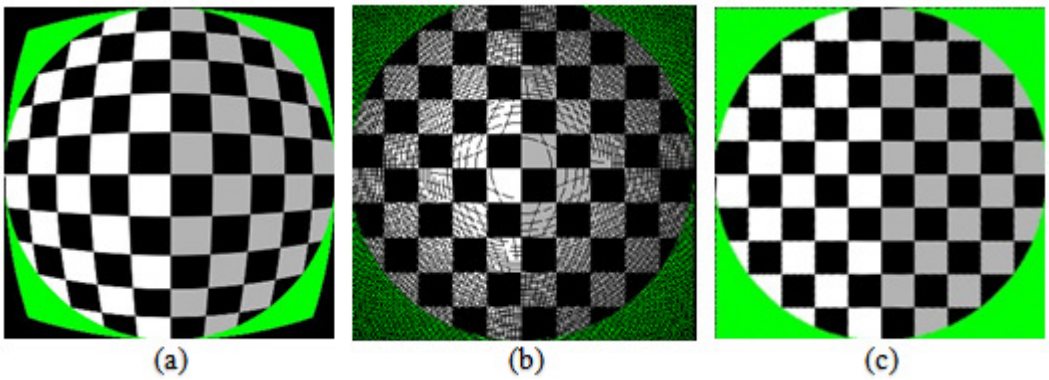

Figura 2.14 – Correção de imagem sintética: a) original, b) mapeamento direto e c) mapeamento inverso.

2.5 FORMATOS DE PANORAMAS

Apesar de não haver uma definição rígida na literatura para classificar uma imagem como panorâmica, é em geral requerido que esta represente uma cena com campo de visão horizontal ao menos igual ao da visão humana, que é em geral 160° [23]. Este trabalho tratará de panoramas que representam uma cena completa, isto é, com FOV horizontal e vertical de 360 e 180°, respectivamente, como na Figura 2.15. A técnica discutida será a de *mosaicing*, que consiste na superposição de várias fotos da mesma cena. Esta é uma técnica que desde o fim do século XIX possui aplicações em mapeamento topográfico, e teve maior desenvolvimento com a fotografia aérea no início do século XX [24]. Até então a composição das imagens tinha que ser feita manualmente, mas serão discutidas aqui as técnicas automáticas disponíveis com o processamento digital de imagens.

Figura 2.15 – Panorama com FOV horizontal e vertical de 360° e 180° respectivamente. Autor: Manfred Gruber.

Como já foi mostrado, não é possível representar um campo de visão tão elevado com uma projeção perspectiva (retilínea). Do ponto de vista de um observador imerso em uma cena, o ambiente à sua volta pode ser considerado com a superfície interna de uma esfera. O mapeamento desta cena no plano 2-D da imagem é análogo ao problema tratado na cartografia de se representar o globo em uma superfície plana. Das várias classes de projeções cartográficas disponíveis, são usadas em geral as cilíndricas para que a imagem final tenha o aspecto retangular familiar de uma foto. Neste tipo de mapeamento, a esfera é projetada em um cilindro circunscrito [25] como o diagrama à esquerda na Figura 2.16. A projeção pode ser feita de diversas maneiras a fim de se manipular a distorção envolvida para que alguma conformidade útil seja alcançada ou apenas o aspecto visual seja mais agradável. As famosas projeções de Mercator, Miller e Lambert são usadas para esse fim. Na criação de panoramas se opta em geral pela projeção equiretangular pela sua simplicidade no tratamento matemático. A coordenada horizontal é equivalente à longitude e a vertical à latitude.

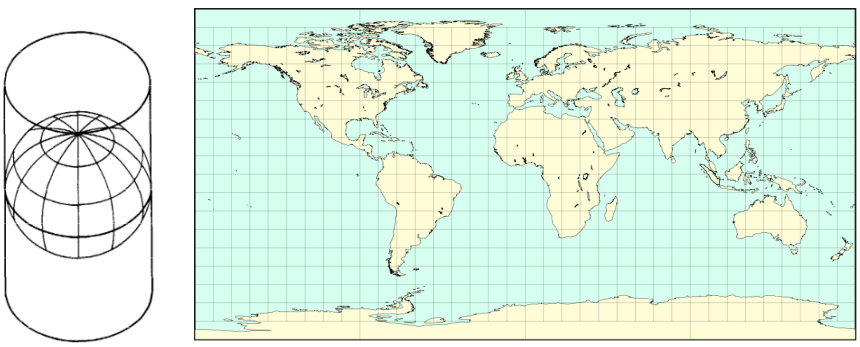

Figura 2.16 – Formação de uma projeção cilíndrica (esq.) [25] e a projeção equiretangular (dir.) [fonte: http://www.progonos.com].

No contexto da cartografia, a latitude é comumente representada por ϕ e a longitude por λ, portanto

$$x = R(\lambda - \lambda_0)$$
$$y = R\phi \tag{2.20}$$

onde R é o raio da esfera, λ_0 é a longitude central e x e y são as coordenadas no plano. A Figura 2.15 ilustra um panorama criado com este mapeamento, e nela pode-se notar que as linhas retas verticais são preservadas, enquanto as horizontais ganham um aspecto curvo.

A partir da projeção equiretangular, que mapeia toda a esfera, é possível, a partir de simples conversões de coordenadas, obter projeções retilíneas de porções da imagem. A projeção retilínea é também chamada de gnomônica, ilustrada na Figura 2.17(a). Ainda nesta figura, são ilustradas porções perspectivas extraídas da Figura 2.15. Um sistema de navegação imersiva é criado desta maneira, reprojetando um panorama equiretangular de maneira interativa.

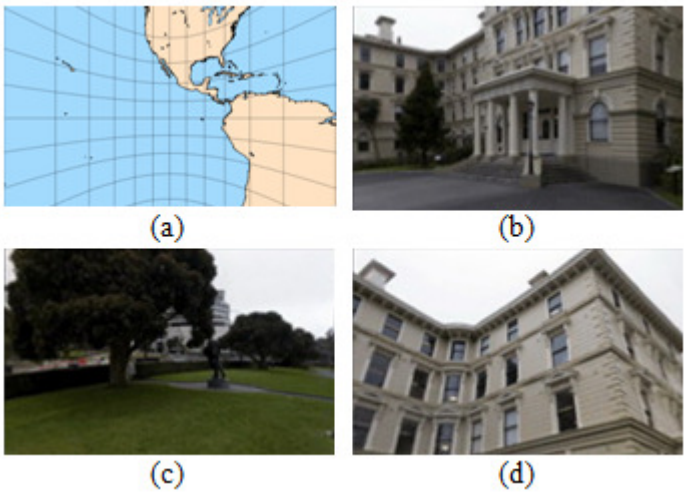

Figura 2.17 – a) Projeção gnomônica e b-d) visões retilíneas extraídas de um panorama equiretangular.

2.6 CAPTURA E REGISTRO DE IMAGENS PARA PANORAMAS

Neste trabalho são considerados os panoramas formados a partir de um conjunto de imagens *fisheye* de uma cena. O uso de lentes *fisheye* para elaboração de panoramas é uma tendência neste campo pois possibilita que um número reduzido de fotos seja suficiente para cobrir todo o ambiente, dado que o campo de visão de cada imagem é considerável. Caso sejam usadas lentes retilíneas, é necessário um conjunto de 30 fotos em média [20], tornando o processo de aquisição bastante complexo e susceptível a erros. Com imagens *fisheye* circulares cobrindo 180°, apenas três fotos são necessárias. Em teoria, apenas duas fotos bastariam para cobrir os 360° totais, mas a falta de sobreposição causaria artefatos acentuados na linha de junção ao gerar-se um panorama retangular. Algumas lentes com FOV superior a 180° garantem uma região de sobreposição suficiente, mas seu custo é bastante elevado.

A técnica com melhor custo-benefício entre quantidade de imagens e robustez no processo de *mosaicing* automático é a captura de seis fotos *full-frame* na orientação de retrato girando-se a câmera em um eixo vertical em ângulos igualmente espaçados. Mais duas fotos são necessárias para o zênite e o nadir, caso se queira cobrir os 180° verticais. Assumindo que a imagem *full-frame* possua FOV de 180° na diagonal, ela cobrirá em torno de 100° na horizontal e 140° na vertical, dependendo da projeção específica e do *crop factor*. A Figura 2.18 ilustra um exemplo de seis fotos capturadas do centro de uma sala. Também é possível a captura de apenas quatro fotos ao longo da rotação caso sejam usadas imagens *quasi fisheye*, mas a estabilidade do processo de registro será um pouco menor.

É importante ressaltar que o eixo vertical em torno do qual a câmera gira deve passar pela pupila de entrada da lente, como indicado na Figura 2.19(a). Pupila de entrada é a imagem óptica do bloqueador de abertura vista do lado do objeto. Caso se gire em torno de outro eixo, ocorrerá o efeito de paralaxe na sequência de imagens, que é o aparente deslocamento de objetos, gerando possíveis oclusões, como na Figura 2.19(b). No caso de lentes *fisheye*, o processo se complica pois a pupila de entrada varia

com o ângulo de incidência dos raios de luz. Neste caso, deve-se considerá-la na posição angular das subsequentes rotações [26].

Figura 2.18 – Imagens *full-frame* capturadas para formação de panorama. [fonte: http://www.360dof.com]

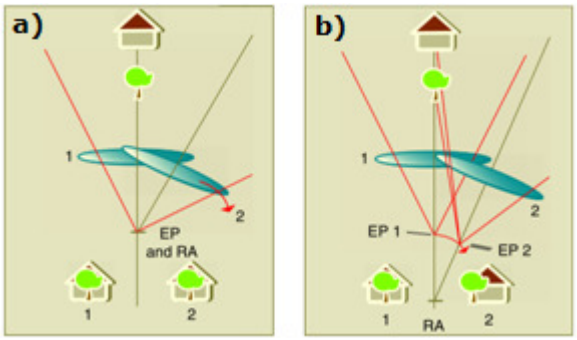

Figura 2.19 – Rotação da câmera a) correta e b) incorretamente, gerando paralaxe no segundo caso. [5]

Uma vez capturadas, as imagens devem ser registradas, i.e., correspondências entre elas devem ser estabelecidas, a fim de se determinar quais são consecutivas e o quanto estas devem ser sobrepostas. As correspondências são denominadas pontos de controle. Este é o processo central do *mosaicing* [24], que garantirá ou não o sucesso da operação. Embora possa ser feito manualmente através da seleção interativa de pontos de controle, serão focados os métodos automáticos.

Um método exaustivo que testa todos os possíveis parâmetros de deslocamento é pouco viável computacionalmente, embora um processamento hierárquico (ajuste grosseiro seguido de fino) e paralelo acelere notavelmente a performance. Já um método de ajuste iterativo não é muito robusto, pois comumente leva a um mínimo local, a não ser que uma boa estimativa inicial seja calculada com uma busca global prévia. São possíveis também métodos que exploram correlações no domínio da frequência ente as imagens, mas, embora sejam computacionalmente atrativos, necessitam de uma região de sobreposição grande entre as imagens (ao menos 50%).

A classe de métodos mais eficaz é aquela em que determinadas características são extraídas de cada imagem e, em seguida, são casadas baseado em alguma medida de similaridade. A falta das características que se procura ou imagens muito simétricas podem levar este método a falhar. Este método pode ser dividido em três passos: detecção, descrição e casamento dos pontos de interesse [7]. No primeiro, buscam-se, em cada imagem, locais que tenham grande chance de serem encontrados em outras imagens. No segundo, a região em volta desses pontos de interesse é convertida em um descritor compacto e invariante, capaz de ser casado no último passo com prováveis candidatos de outras imagens. Uma variante deste modelo que vem sendo bastante utilizada é a *Scale Invariant Feature Transform* (SIFT) [27]. Uma implementação de código aberto (*autopano-sift-C*, incluído no *software* Hugin [28]) foi utilizada na geração dos panoramas deste trabalho.

2.7 COMPOSIÇÃO E BLENDING

Após o alinhamento das imagens, é necessário mapear cada imagem de acordo com a projeção de saída escolhida. Nota-se que os pontos são mapeados de acordo com suas coordenadas de latitude e longitude nas imagens, portanto o mapeamento *fisheye* da lente utilizada deve ser conhecido. Este processo é denominado *warping*, e deve ser adaptativo, de maneira que a correspondência determinada no alinhamento seja respeitada da melhor maneira possível.

Por último, deve ser decidido como combinar de fato os *pixels* do panorama. Caso todas as imagens tenham sido perfeitamente registradas e possuam exposição idêntica, a solução é trivial, pois qualquer combinação serviria. Contudo, para imagens reais, é comum a ocorrência de junções visíveis devido a diferenças de exposição, borrados devido a um registro imperfeito e fantasmas devido à movimentação de objetos. Szeliski divide o processo em seleção de *pixels* (*de-ghosting*) e combinação de *pixels* (*blending*), embora afirme que a distinção entre os dois estágios é sutil [7].

A maneira mais simples de se combinar os *pixels* é calculando a média entre eles. Este método não possui grande vantagem além de sua simplicidade pois as imperfeições citadas continuam visíveis. Já um filtro de mediana é capaz de remover objetos com posições distintas, mas também pode gerar resultados indesejáveis ao remover características demais. Uma abordagem intermediária é a média ponderada com pesos maiores para *pixels* no centro. A distância Euclidiana de cada pixel ao pixel inválido (i.e., fora do mapeamento) mais próximo é usada como peso no cálculo da média, sendo este processo denominado *feathering*. Ele pode ainda ser melhorado ao elevar-se a métrica a uma potência alta, fazendo com que os pesos sejam dominados pelos valores altos da métrica original. A este último método dá-se o nome de *p-norm*, devido à sua similaridade a uma norma vetorial. Os métodos são comparados na Figura 2.20, onde nota-se a redução gradual de fantasmas da média simples na parte (a) para o *feathering* na parte (c) e posteriormente o *p-norm* aplicado com $p = 10$ na parte (d). A mediana, vista na parte (b), elimina diversos elementos da imagem de maneira agressiva e torna as junções mais óbvias.

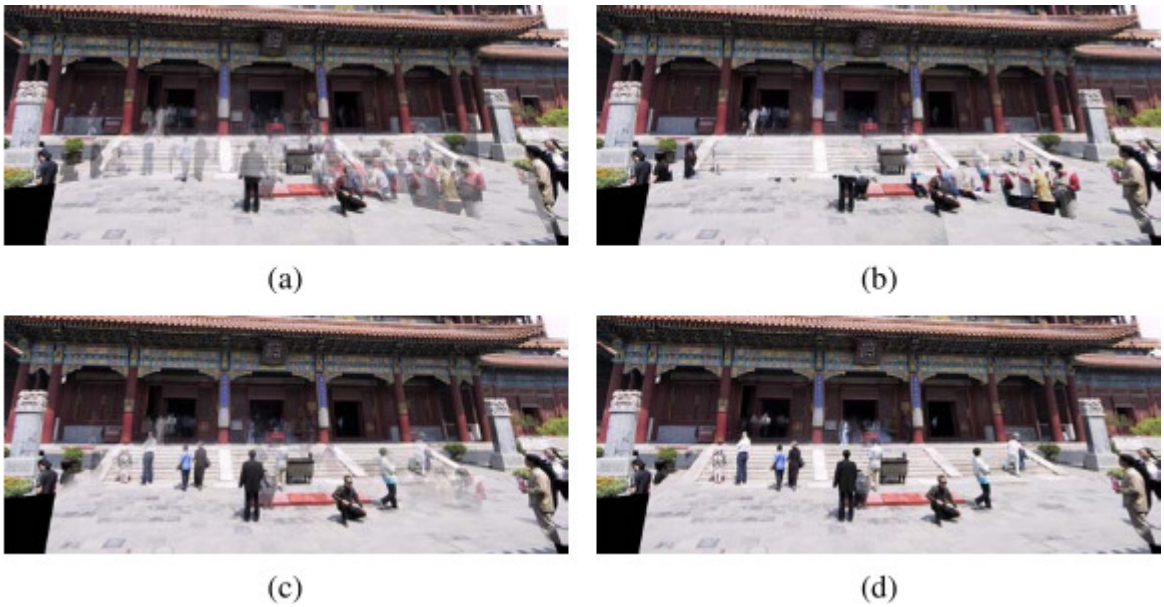

Figura 2.20 – Comparação entre métodos de seleção de *pixels*: a) média, b) mediana, c) *feathering* e d) *p-norm* (p = 10). [7]

Entre os métodos de correção de variação na exposição e demais desalinhamentos, destacam-se o uso de pirâmides Laplacianas e de operações no domínio do gradiente. A primeira técnica consiste em adaptar a largura da região de combinação para cada banda de frequência, ou seja, para cada nível da pirâmide. Uma implementação de código aberto desta técnica foi utilizada na geração dos panoramas deste trabalho (*Enblend* [29]). Alternativamente, a combinação de *pixels* pode ser feita no domínio do gradiente, sendo que a volta para o domínio da imagem é calculada em geral resolvendo-se a equação de Poisson.

3 CODIFICAÇÃO DE VÍDEO E O PADRÃO H.264

3.1 ESPAÇO DE CORES E SUBAMOSTRAGEM

Vídeo digital, no contexto deste trabalho, é a representação de uma cena natural amostrada espacialmente e temporalmente [3]. A cena é amostrada em um instante de tempo para que seja composto um quadro (*frame*), que é um composto por um conjunto de elementos (*pixels*) arranjados espacialmente em um *grid* retangular. São necessários três elementos por posição amostrada para que se tenha a informação de cor daquele ponto de maneira similar ao sistema de visão humano (HVS) [30]. O método escolhido para representar a cor e iluminação com esses três elementos é denominado espaço de cores.

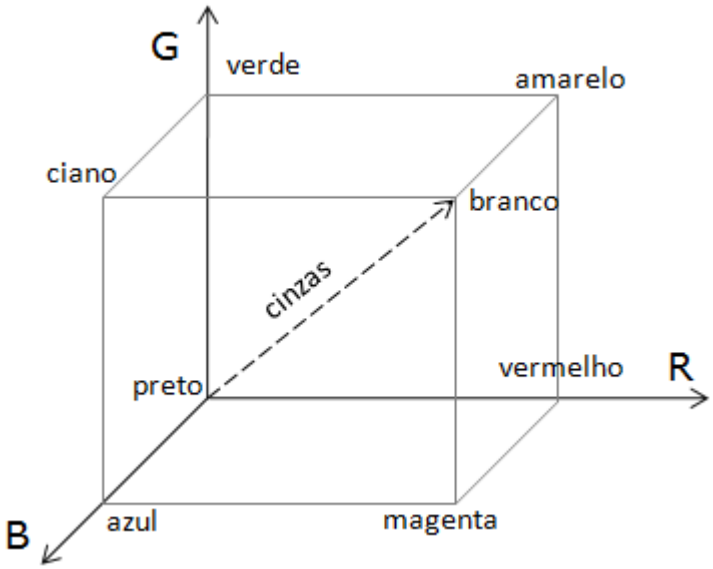

Figura 3.1 – Cubo RGB.

Um espaço de cores bastante conhecido está ilustrado na Figura 3.1, o RGB. A amostra de cor é representada com três valores que indicam as proporções relativas de vermelho, verde e azul, que são as três cores primárias aditivas. Este espaço é apropriado para a captura e exibição de imagens. A captura envolve a filtragem de cada faixa do espectro associada a uma das cores citadas para que excitem sensores separadamente. Monitores do tipo LCD (display de cristal líquido) e CRT (tubos de raios catódicos) exibem uma imagem RGB iluminando separadamente as componentes vermelha, verde e azul de cada pixel de acordo com a intensidade de cada componente. A uma distância de observação normal, os pequenos pontos se confundem e dão forma a uma imagem colorida.

Um espaço alternativo que permite a representação mais eficiente da informação é o YUV, também chamado de YCbCr no domínio digital. Valendo-se do fato de que o HVS é mais sensível à luminância (intensidade de luz ou brilho) do que às cores, o sistema separa essas componentes, permitindo representar as menos importantes com resolução menor. Em contraste, no sistema RGB, cada componente é igualmente importante e deve ser representada com a mesma resolução. A componente Y corresponde à luminância, e é calculada como uma média ponderada entre os valores R, G e B, com pesos k_i:

$$Y = k_r R + k_g G + k_b B. \tag{3.1}$$

A informação de cor pode ser representada como crominância, que é a diferença entre as componentes R, G, B e a luminância Y:

$$Cb = B - Y \quad e \quad Cr = R - Y, \tag{3.2}$$

sendo a terceira componente de crominância comumente estimada a partir das outras duas, uma vez que a soma das três é uma constante. Os pesos utilizados na Eq. (3.1) não são únicos e dependem do padrão utilizado na aplicação. A recomendação da ITU-R BT.601 define $k_r = 0{,}299$ e $k_b = 0{,}114$, sendo $k_g = 0{,}587$ obtido indiretamente, uma vez que a soma dos pesos deve ser igual à unidade. Esta é a forma de conversão adotada neste trabalho.

Como indicado anteriormente, as componentes de crominância podem ser amostradas com menor resolução espacial, sem aparente perda de qualidade subjetiva. Este processo é denominado subamostragem. A Figura 3.2 ilustra as diferentes possibilidades para a amostragem espacial. Os círculos brancos representam componentes Y e os cinzas os de crominância. Na Figura 3.2(c), as três componentes são amostradas com a mesma resolução espacial. Na Figura 3.2(b), as componentes estão presentes a cada duas amostras de Y, e, na Figura 3.2(a), a cada quatro amostras. Cada formato é descrito por uma razão de três partes na forma *J:a:b*, que descreve, para uma região com largura de J amostras e altura 2, quantos componentes de crominância há na primeira linha (a) e quantos há na segunda (b). Os formatos da Figura 3.2 são, portanto, na ordem, 4:2:0, 4:2:2 e 4:4:4. O formato 4:2:0 é bastante comum em diversas implementações de padrões de compressão de imagem e vídeo, e foi usado neste trabalho quando se fez necessário trabalhar no espaço YCbCr.

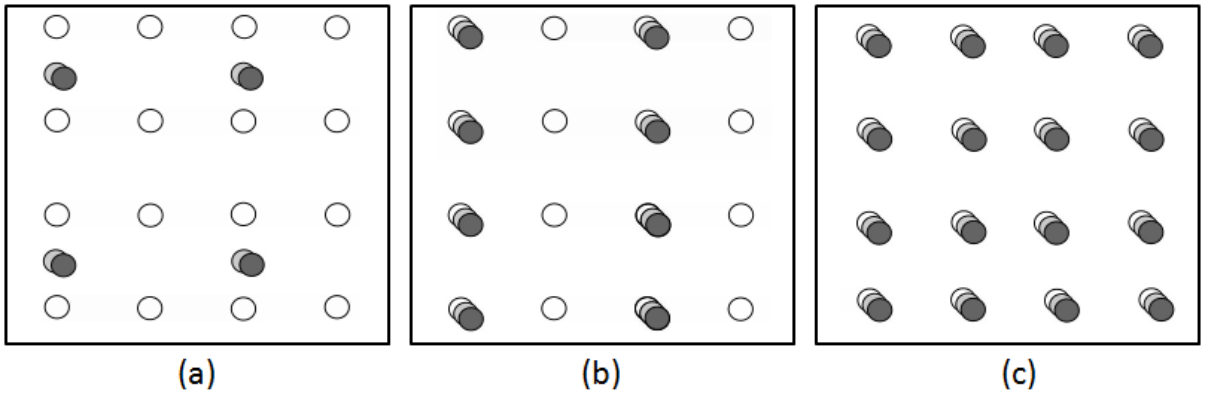

Figura 3.2 – Amostragem de componentes no YCbCr: a) 4:2:0, b) 4:2:2, c) 4:4:4. (modificado [3])

3.2 MÉTRICAS DE QUALIDADE

A codificação de vídeo estudada neste trabalho será do tipo com perdas, i.e., o vídeo descomprimido apresentará qualidade visual inferior ao original. Com isso, taxas de compressão elevadas podem ser alcançadas, ao custo de uma degradação no sinal recuperado. Esta degradação deve ser medida além da taxa de compressão para que sistemas diferentes possam ser comparados de maneira clara.

O processo de medição de qualidade visual é um campo amplo de estudo pois não é um problema com solução objetiva. A opinião de um observador acerca da qualidade de um vídeo pode variar de acordo com sua interação com o conteúdo. O observador pode estar consumindo vídeo como entretenimento em sua casa ou como parte do seu trabalho em uma videoconferência, gerando expectativas e níveis de atenção diferentes.

Metodologias de análise de qualidade visual subjetiva são em geral difíceis de serem postas em prática, portanto medidas objetivas são atrativas para pesquisadores. A métrica mais utilizada é a *Peak Signal to Noise Ratio* (PSNR), baseada no erro médio quadrático (MSE). O MSE entre duas imagens representadas pelas matrizes *I* e *J*, com dimensões $m \times n$ total de *pixels* $N = m \cdot n$, é dado por

$$MSE = \frac{1}{N}\sum_{i=1}^{m}\sum_{j=1}^{n}(I(i,j) - X(i,j))^2. \tag{3.3}$$

Assumindo que essas imagens sejam codificadas com 8 bits por amostra (valor máximo de 255), a PSNR entre elas é calculada como

$$PSNR = 10\log_{10}\frac{255^2}{MSE}. \quad (3.4)$$

Caso seja usado o espaço de cores YUV, é comum obter esta métrica para cada componente separadamente, sendo que a qualidade da luminância geralmente representa o parâmetro de maior interesse. Para um vídeo com vários *frames*, a diferença no MSE é calculada entre cada pixel correspondente (mesma posição e mesmo *frame*), e *N* passa a ser o número de *pixels* total no vídeo.

3.3 ESTRUTURA DE UM CODEC DE VÍDEO

A compressão envolve um par de sistemas complementar, um compressor (*encoder* ou codificador) e um descompressor (*decoder* ou decodificador). O codificador converte os dados originais para uma forma comprimida, ocupando menos bits, podendo ela ser transmitida ou armazenada. O decodificador recupera, a partir da informação comprimida, uma representação na forma dos dados originais, que pode ser utilizada como pretendido inicialmente. O par formado pelo codificador e decodificador é chamado de *codec*. Na Figura 3.3 é mostrado um diagrama de blocos de um sistema típico para um codec de vídeo.

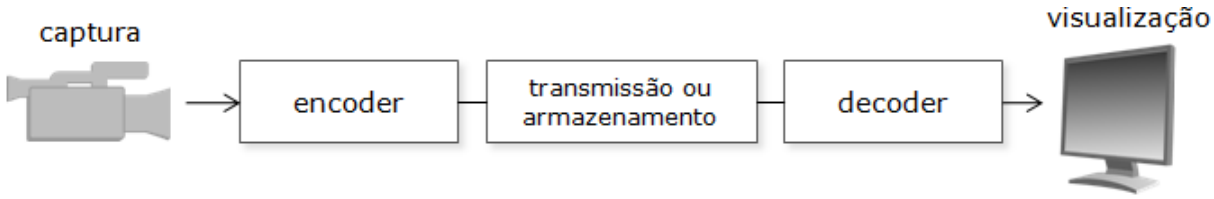

Figura 3.3 – Sistema típico de aquisição e visualização de vídeo com compressão.

Na codificação de vídeo são exploradas redundâncias espaciais e temporais a fim de se conseguir uma compressão do sinal. No domínio do tempo, há uma alta correlação (similaridade) entre *frames* que tenham sido capturados em instantes próximos, principalmente se a taxa de aquisição (*framerate*) é alta. No domínio espacial, geralmente há alta correlação entre *pixels* que estejam próximos uns dos outros, comumente chamados de vizinhos.

Um codec representa a sequência de vídeo original por um modelo, que é uma representação codificada eficiente e que pode ser utilizada para reconstruir uma aproximação do conteúdo original. De maneira ideal, o modelo deve representar utilizando o menor número de bits e com a maior fidelidade possíveis. Esses dois objetivos são quase sempre conflitantes, ou seja, quanto menor a taxa de bits (*bitrate*) de uma representação gerada pelo codificador, maior a perda em qualidade na reconstrução feita pelo decodificador.

Um codificador de vídeo possui em geral três unidades funcionais: uma etapa de predição, uma etapa de transformação e quantização, e um codificador de entropia, conforme ilustrado na Figura 3.4. A entrada da etapa de predição é a sequência original não comprimida. Neste estágio, as redundâncias temporais e espaciais são minimizadas ao construírem-se predições de quadros a partir de pixels do mesmo quadro (predição *intra*) e de outros quadros próximos, i.e., prevendo o movimento dos pixels entre os quadros (predição *inter*). As predições podem ser feitas com base em quadros passados ou futuros. A saída deste estágio é um quadro residual, que é a diferença entre o quadro estimado e o real, mais um conjunto de parâmetros descrevendo a movimentação que foi estimada.

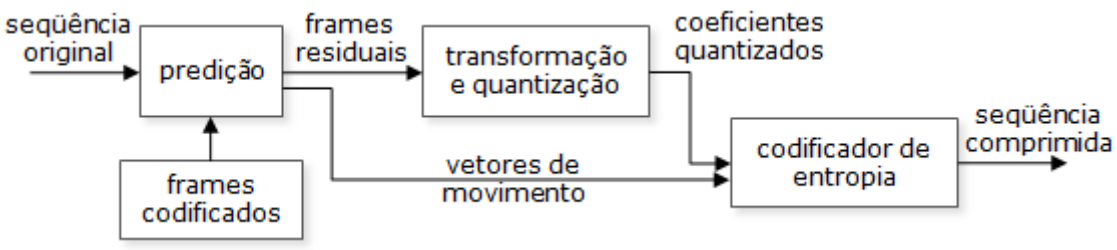

Figura 3.4 – Estrutura de um codificador de vídeo.

Os *frames* residuais formam a entrada para o próximo estágio, a etapa de transformada e quantização, no qual a redundância espacial é reduzida valendo-se das similaridades entre vizinhos. Um processo comum é a aplicação de uma transformada reversível nas amostras residuais, para representá-las como os coeficientes dessa transformação. Esses coeficientes são quantizados, semelhantemente a um processo de arredondamento, e em seguida ordenados com base em sua importância para a representação do sinal. Assim, a quantização garante que os coeficientes transformados sejam representados em valores inteiros, ao invés de ponto flutuante, e os coeficientes menos relevantes são descartados no arredondamento. Os coeficientes quantizados são a saída deste estagio.

Os parâmetros das etapas de predição, transformação e quantização são comprimidos com um codificador de entropia. Este é um processo sem perdas, pois deverá ser revertido perfeitamente no decodificador. A sequência final comprimida consiste portanto em uma série de parâmetros vetoriais (descrevendo as predições *intra* e *inter*) e coeficientes quantizados (descrevendo as amostras residuais), ambos codificados, e informações adicionais de cabeçalho.

O decodificador utiliza esta sequência para reconstruir o vídeo original repetindo os passos do codificador na ordem inversa. Um decodificador de entropia recupera os parâmetros das predições *intra* e *inter* e os coeficientes quantizados. Estes últimos são usados para reconstruir as os *frames* residuais, e os *frames* completos são obtidos a partir da informação de predição. As etapas de predição, transformação e quantização são detalhadas nas seções que seguem.

3.4 ETAPA DE PREDIÇÃO

O objetivo da etapa de predição é reduzir a redundância entre regiões do mesmo *frame* e entre *frames* adjacentes, ao apontar similaridades espaciais em um *frame* e formar predições da movimentação na cena. Subtraindo a predição do *frame* a ser codificado, obtêm-se *frames* residuais, que possuem bem menos energia que um *frame* completo original, e portanto são mais eficientemente comprimidos. A predição de movimentação da cena pode ser feita a partir de *frames* anteriores ou posteriores (*frames* de referência).

A forma mais simples de predição é utilizar o *frame* anterior como predição do *frame* atual. A Figura 3.5 ilustra dois *frames* consecutivos, sendo que o *frame* 1 atua como predição do *frame* 2. O resíduo ao subtrair-se o *frame* 1 de 2 é mostrado por último, e a ele foi somado uma constante para melhor visualização. Quanto mais claras ou escuras as regiões, mais diferença positiva e negativa, respectivamente, há entre os *frames*. Regiões cinza indicam pequena ou nenhuma diferença. Nota-se que boa parte da energia residual é fruto da movimentação de componentes da cena (o livro e a cabeça do homem à esquerda), e a energia residual pode ser reduzida ao compensar-se essa movimentação local entre os *frames*.

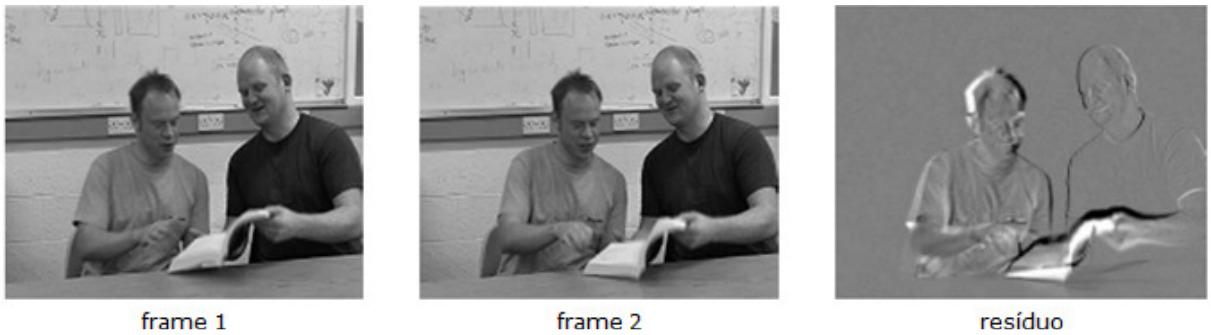

Figura 3.5 – Predição temporal simples. (modificado [3])

As mudanças entre os *frames* podem ser causadas por movimentação de objetos (rígidos ou deformáveis), movimentação da câmera, oclusão de regiões e alteração na iluminação. As duas primeiras podem ser descritas simplesmente pela movimentação de *pixels* de um *frame* para outro. Ao estimar-se a movimentação de cada pixel, é gerado o que se chama de fluxo óptico, visualizado como um campo de trajetórias. Com o conhecimento deste campo, é possível gerar uma estimativa bastante apurada ao moverem-se os *pixels* a partir do *frame* de referencia de acordo com as trajetórias

calculadas. Algoritmos de estimação de fluxo óptico são objeto de grande pesquisa na literatura. Na prática, porém, este método não se torna uma boa solução por ser computacionalmente pesado e por ser necessário armazenar vetores de movimento para cada pixel, a fim de garantir a reconstrução no decodificador, o que representa grande volume de informação e anula a vantagem de se utilizar *frames* residuais. A solução de compromisso é realizar o processamento em blocos de *pixels*.

A estimação de movimento em blocos consiste em subdividir o *frame* em blocos retangulares e compensar o movimento dessas estruturas, ao invés de realizar o processamento pixel a pixel como no cálculo de fluxo óptico. Um algoritmo comum consiste em fixar uma janela de busca em volta de um bloco a ser estimado, e procurar dentro dessa janela, nos *frames* de referência, pelo bloco de mesmo tamanho que tenha maior semelhança ao bloco original. Uma métrica comum para determinar o bloco mais semelhante é escolher aquele que minimiza a energia (soma dos erros absolutos, SAE) do bloco residual (diferença entre os blocos). O candidato escolhido se torna a predição para o bloco em questão, e o bloco residual é armazenado. O processo de busca de um bloco semelhante se denomina estimação de movimento, e a obtenção do bloco residual, compensação de movimento. Com a informação de origem do bloco de referência e a sua movimentação é gerado um vetor de movimento, que é codificado e transmitido junto ao resíduo. Este conjunto de informações é suficiente para que o decodificador seja capaz de reconstruir *frames* semelhantes aos originais ao recriar as predições com os vetores de movimento e somá-las aos blocos de referência.

Essa técnica é popular por ser de fácil compreensão e computacionalmente viável. Além disso, constitui em uma boa combinação para as transformadas em bloco aplicadas aos resíduos, como a DCT, e constituem em geral um modelo razoável para as variações temporais em uma cena. Obviamente, cenas reais não apresentam total conformidade a variações espaciais em bloco, como no caso de objetos deformáveis, rotações e movimentações complexas (líquidos, fumaça etc.). Essas variações são difíceis de serem compensadas por blocos e acabam gerando predições incompletas nas imagens reconstruídas. Apesar disso, esta é a base do modelo temporal de todos os padrões de codificação de vídeo atualmente.

O tamanho padrão de um bloco em diversos padrões de codificação é um quadrado de 16 x 16 *pixels* denominado macrobloco. Para uma sequência de vídeo no padrão YUV 4:2:0 (Figura 3.2(a)), ele consiste em 256 amostras de luminância arranjados em quatro blocos de 8 x 8 *pixels* e 128 amostras de crominância arranjados em dois blocos de 8 x 8 *pixels* (um para Cb e outro para Cr), totalizando seis blocos. Porém, os objetos que se movimentam em uma cena raramente seguem um traçado bem delimitado de macroblocos, portanto é desejável utilizar-se um tamanho variável de bloco para predição e compensação de movimento. Outra possibilidade é interpolar os *frames* de referência para uma resolução sub-pixel, a fim de melhorar a busca por partes de um objeto que não se movam um número inteiro de *pixels* entre *frames*.

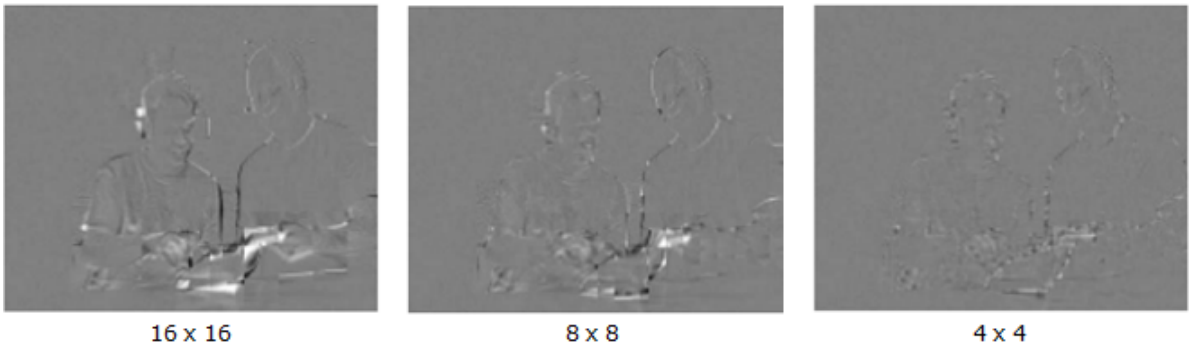

Figura 3.6 – Resíduos com compensação de movimento. (modificado [3])

A Figura 3.6 mostra como seria a predição entre *frames* da Figura 3.5 com compensação de movimento. Os resíduos são mostrados da mesma forma que o resíduo da Figura 3.5, e cada um corresponde à compensação de movimento usando um tamanho de bloco diferente, conforme a legenda. Nota-se que todos apresentam bem menos energia (regiões claras ou escuras) que o resíduo da Figura 3.5, e que, à medida que o tamanho do bloco utilizado diminui, a energia residual também diminui. Apesar de o resultado da compensação de movimento melhorar com a diminuição dos blocos, a complexidade da predição também aumenta, i.e., são necessárias mais buscas (mais processamento)

para se chegar a estes resultados e mais informação de vetores de movimento devem ser armazenadas. Como os vetores de movimento ocupam espaço na sequência codificada, deve-se avaliar com cuidado até que ponto a diminuição dos blocos é vantajosa na redução da taxa de bits.

Uma solução de compromisso interessante é adaptar o tamanho dos blocos às características de diferentes regiões da cena. Por exemplo, fundos homogêneos como um céu limpo ou uma parede lisa podem ser estimados com blocos maiores de maneira eficiente e sem muita perda em qualidade, enquanto que conteúdos com muitos detalhes e movimentos como rostos ou explosões se beneficiam da predição com blocos de tamanho reduzido.

O *frame* residual pode ser gerado também a partir de predições partindo do mesmo *frame*. Assumindo que a imagem seja codificada de maneira ordenada (e.g., *raster*) *pixels* já codificados acima e à esquerda podem ser usados para estimar o valor do pixel atual. Esta estimativa é subtraída do valor real e o resíduo resultante é codificado. O decodificador deve usar o mesmo formato de predição para reconstruir o pixel corretamente. Diferentes formatos de predição podem ser usados em áreas com determinadas características na imagem, provendo estimativas mais próximas do valor real mas aumentando o *overhead* de informação passado ao decodificador. Este processo é denominado predição *intra* ou ainda *Differential Pulse Coded Modulation* (DPCM) e é descrito em maior detalhe em 3.8.

3.5 MODELO ESPACIAL

O *frame* de um vídeo pode ser representado como uma função bidimensional, na qual as coordenadas no plano referentes à posição de um elemento são associadas a um valor que indica a intensidade do pixel. Imagens de cenas reais são em geral difíceis de comprimir com codificação de entropia por possuírem alta correlação entre elementos vizinhos. Já um *frame* residual, como os da Figura 3.6, possui baixa correlação local, sendo, portanto, mais fácil de comprimir que o *frame* original. A função da etapa de transformação e quantização é converter os dados para uma forma que aumente a eficiência de compressão.

A aplicação de uma transformada ao resíduo converte as amostras para outro domínio de representação, no qual elas devem apresentar menor correlação e maior compactação. Menor correlação significa ser representado por elementos (coeficientes) que tenham mínima dependência entre si. Maior compactação requer que a maior parte da informação (energia) esteja concentrada em um número reduzido de elementos. Além disso, a transformada deve possuir uma inversa e ser computacionalmente eficiente.

Há duas categorias básicas para os vários tipos de transformadas existentes para compressão de vídeo e imagem: transformadas de bloco e de imagem. Entre as transformadas de bloco, a mais utilizada é a Transformada de Cossenos Discreta (DCT), que será descrita em maior detalhe a seguir. Já entre as transformadas que operam na imagem inteira (ou em uma porção), destaca-se a Transformada Wavelet Discreta (DWT), utilizada no padrão JPEG2000. Apesar de algumas medições indicarem uma vantagem da DWT em relação a transformadas de bloco para imagens estáticas, ela não se ajusta bem à compensação de movimentos em bloco utilizada em vídeo, ao contrário da DCT.

A DCT bidimensional opera em um bloco de N x N amostras $\boldsymbol{X}$ e gera um bloco de coeficientes de mesmo tamanho $\boldsymbol{Y}$. A aplicação da DCT pode ser descrita por uma matriz de transformação A, assim como a aplicação da transformada inversa (IDCT). As formas direta e inversa da transformada são dadas respectivamente por

$$\begin{aligned} \boldsymbol{Y} &= \boldsymbol{A}\boldsymbol{X}\boldsymbol{A}^T \\ \boldsymbol{X} &= \boldsymbol{A}^T\boldsymbol{Y}\boldsymbol{A}. \end{aligned} \tag{3.5}$$

Os elementos de $\boldsymbol{A}$ são dados por

$$A_{ij} = C_i \cos\left(\frac{(2j+1)i\pi}{2N}\right), \tag{3.6}$$

onde $C_i = \sqrt{1/N}$, para $i = 0$, e $C_i = \sqrt{2/N}$, para $i > 0$.

Os coeficientes resultantes da aplicação da DCT podem ser interpretados como pesos de um conjunto de padrões de base no domínio da transformada. As bases são compostas de combinações de funções cosseno ao longo das linhas e colunas. Na Figura 3.7 são ilustradas as bases para a DCT 4 x 4. O bloco original de mesmo tamanho pode ser reconstruído a partir da combinação de todas essas bases, multiplicando cada uma por um peso adequado, que são os coeficientes calculados.

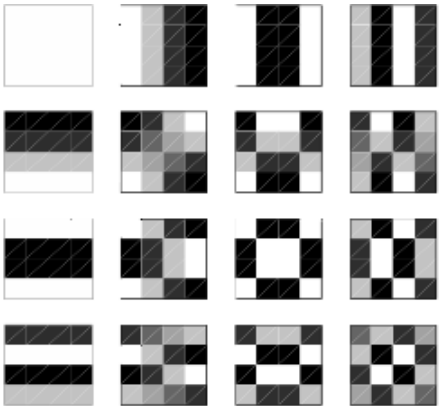

Figura 3.7 – Bases da DCT 4 x 4. (modificado [3])

A Figura 3.8 ilustra a vantagem da representação no domínio da DCT. Uma amostra ***X*** de tamanho 4 x 4 elementos e seus coeficientes ***Y*** são dadas por:

$$X = \begin{pmatrix} 126 & 159 & 178 & 181 \\ 98 & 151 & 181 & 181 \\ 80 & 137 & 176 & 156 \\ 75 & 114 & 88 & 68 \end{pmatrix}, Y = \begin{pmatrix} 537.25 & -76.0032 & -54.75 & -7.7552 \\ 106.0541 & -34.9935 & 12.7198 & 6.1412 \\ -42.75 & 46.5398 & 10.25 & -9.8065 \\ 20.2027 & -12.8588 & -3.9157 & 8.4935 \end{pmatrix}$$

Os elementos da amostra são representados por níveis de cinza entre 0 e 255. Fazendo todos os elementos de ***Y*** iguais a zero, exceto o nível DC (537,25), aplica-se a IDCT e obtém-se um bloco cujos elementos são todos iguais ao nível de cinza médio de ***X***. Ao acrescentar-se o próximo coeficiente mais significativo, a reconstrução começa a tomar a forma do bloco original, e com cinco coeficientes a reconstrução é satisfatória. Desta maneira é possível armazenar apenas cinco elementos para representar ***X***, ao invés dos 16 iniciais.

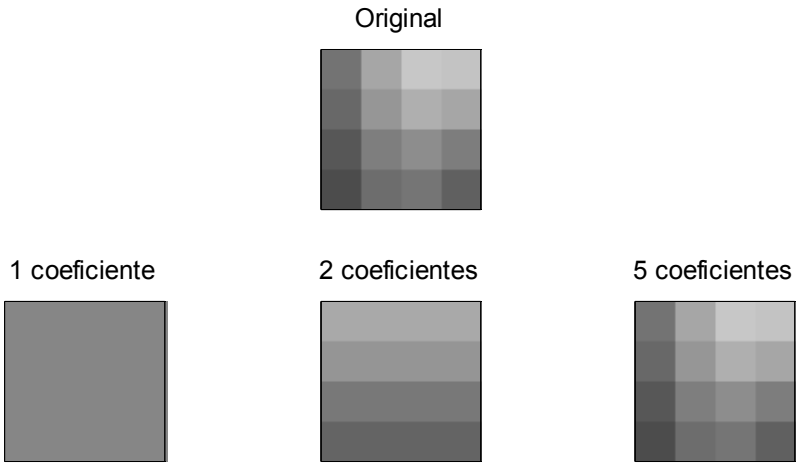

Figura 3.8 – Exemplo de reconstrução de um bloco a partir de seus coeficientes da DCT.

A etapa final do modelo espacial é a quantização dos coeficientes do bloco no domínio da DCT. Uma função quantizadora é aquela que mapeia um sinal definido em um espaço de valores permitidos para outro espaço com menos valores disponíveis. Ao se reduzir a quantidade de valores que a função pode assumir, são necessários menos bits para representá-la, portanto esta é uma forma de compressão do

sinal. Serão discutidos quantizadores escalares, que mapeiam elementos individualmente, mas também há os quantizadores vetoriais, que mapeiam grupos de elementos.

A quantização pode ser entendida como um arredondamento, e este é um processo de compressão com perdas, i.e. não-reversível, uma vez que não é guardado um registro sobre as informações descartadas. A forma geral de um quantizador é

$$Y = QP \cdot round\left(\frac{X}{QP}\right), \tag{3.7}$$

onde X é a amostra original, Y é o valor quantizado e QP é o parâmetro de quantização, também chamado de *step*, pois os valores de Y são igualmente espaçados de QP. Nota-se que quanto maior o QP, menor o alcance de valores permitidos, e o sinal pode então ser eficientemente representado de maneira comprimida. Por outro lado, a reconstrução será prejudicada e terá uma forma sucessivamente menos parecida com o sinal original. A estimação de QP é um compromisso importante na codificação do sinal, pois deve-se procurar um equilíbrio entre sua compressão eficiente e a garantia de uma reconstrução satisfatória. A saída típica da quantização de uma matriz de coeficientes da DCT é uma matriz esparsa, com vários coeficientes de baixa energia zerados. Eles são então ordenados de maneira que os coeficientes nulos estejam agrupados, facilitando o processo de compressão com um codificador de entropia. A quantização vetorial é um processo análogo, com a diferença de que o espaço para o qual se mapeia é um conjunto de padrões de coeficientes (*codebook*). O decodificador deve ter conhecimento do *codebook* utilizado para que seja possível a reconstrução.

3.6 PADRÃO H.264/MPEG-4 AVC E SEUS PERFIS

Desenvolvido pelo *Video Coding Experts Group* (VCEG) da ITU-T e pelo *Moving Pictures Expert Group* da ISO e IEC, o padrão de codificação de vídeo denominado *Advanced Video Coding* é atualmente o mais disseminado na indústria e na academia. Publicado como parte 10 do MPEG-4 e como recomendação H.264 da IUT-T [2], é também comumente chamado *H.264/AVC*. O objetivo do desenvolvimento do padrão era prover funcionalidade similar aos padrões anteriores *H.263+* e *MPEG-4 Visual*, porém proporcionando melhores taxas de compressão à mesma qualidade e adicionando suporte à transmissão robusta em canais de comunicação. As aplicações do padrão são armazenamento e teledifusão em alta definição[2] (é o padrão adotado no Sistema Brasileiro de Televisão Digital – SBTVD – e uma das opções para discos *Blu-Ray*), videotelefonia e *streaming* em redes comutadas por pacote como a Internet.

O padrão é um documento que define uma representação codificada (sintaxe) para representar as sequências de vídeo comprimidas e um método para decodificar essa sintaxe, a fim de reconstruir a sequência corretamente. Um dos objetivos é garantir que *codecs* compatíveis sejam operáveis entre si, e ainda assim proporcionar aos desenvolvedores a liberdade para criar produtos competitivos e inovadores. Nota-se que o padrão não define um codificador (*encoder*), apenas a saída que este codificador deve produzir. Um método de decodificação (um *decoder*) é apresentado, mas os desenvolvedores são livres para programarem suas próprias versões, desde que estas produzam o mesmo resultado do método de referência.

A Figura 3.9 ilustra as unidades funcionais típicas de um codificador H.264/AVC: predição, transformação, quantização, codificação de entropia e o filtro anti-blocos, uma novidade da especificação. Podem ser observados dois sentidos de fluxo da informação: um principal, denominado direto, representado da esquerda para a direita, e um secundário para reconstrução, representado da direita para esquerda pela quantização e transformação inversas, seguido da filtragem. A Figura 3.10 representa a estrutura típica de um decodificador, com o fluxo da informação representado da direita para esquerda a fim de se realçar as semelhanças com o codificador da Figura 3.9.

Na Figura 3.9, o *frame* de entrada é processado em unidades de macrobloco, como descrito em 3.4, e para cada um deles é formada uma predição no modo *intra* ou *inter*. No modo *intra*, a referência para as predições são amostras do quadro atual que já tenham sido codificadas, decodificadas e

[2] Vídeo em alta definição se refere a qualquer sistema de vídeo com resolução superior a 480 linhas (definição padrão ou SD). Geralmente se refere aos formatos 720p e 1080p.

reconstruídas. No modo *inter*, as predições são formadas com compensação de movimento a partir de blocos de um ou dois quadros previamente codificados, reconstruídos e filtrados (*frames* de referência). Na Figura 3.9 está indicado F'_{n-1}, mas as referências também podem ser quadros posteriores, bastando que tenham sido codificados antes do atual. A predição é subtraída do bloco que está sendo processado para produzir um bloco residual, que será transformado em coeficientes por uma aproximação da DCT (T), que são em seguida quantizados ($Q(\cdot)$) e ordenados para facilitar o estágio final de codificação de entropia. Junto aos coeficientes são codificados vetores de movimento e cabeçalhos necessários (modo de predição, parâmetro de quantização etc.) para decodificar e reconstruir o bloco, formando a sequência de saída comprimida (*bitstream*). O fluxo de reconstrução, na parte inferior da figura, é responsável por decodificar cada bloco que acabou de ser codificado, para que possam ser usados nas predições seguintes. Os coeficientes da transformada são re-escalonados ($Q^{-1}(\cdot)$) e processados pela transformada inversa (T^{-1}), produzindo um bloco diferencial. Este bloco é somado ao resíduo para formar uma versão reconstruída, que é filtrada e utilizada para compor os quadros de referência.

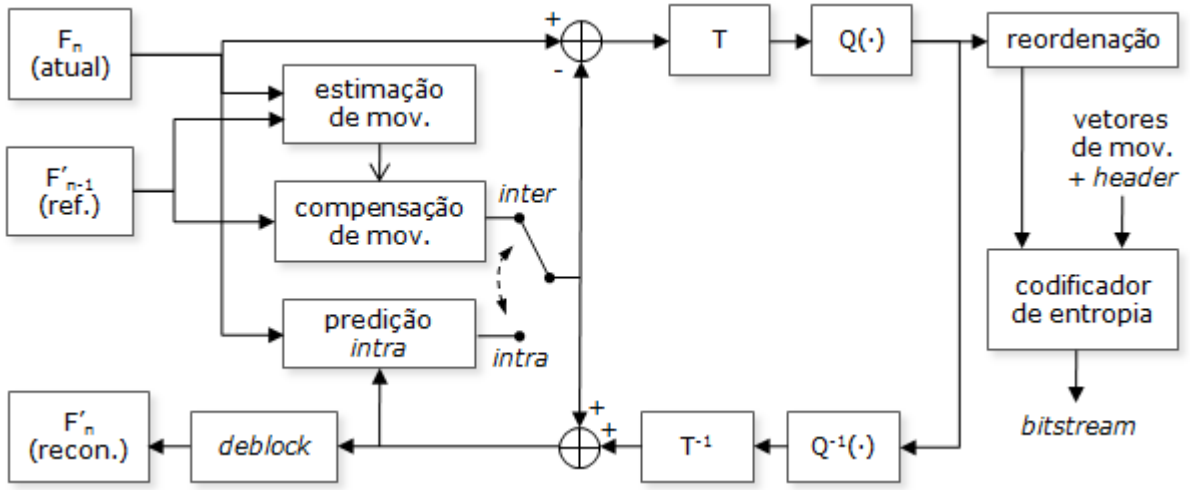

Figura 3.9 – Unidades funcionais típicas de um codificador do padrão H.264.

Na Figura 3.10, o decodificador recebe uma sequência comprimida e utiliza um decodificador de entropia para recuperar os coeficientes da transformada, os quais são reordenados, re-escalonados ($Q^{-1}(\cdot)$) e transformados em blocos diferenciais pela transformada inversa (T^{-1}). Utilizando as informações de cabeçalho e vetores de movimento, um bloco de predição é gerado e somado ao bloco diferencial, gerando um bloco reconstruído, que é filtrado e armazenado para compor a sequência a ser visualizada.

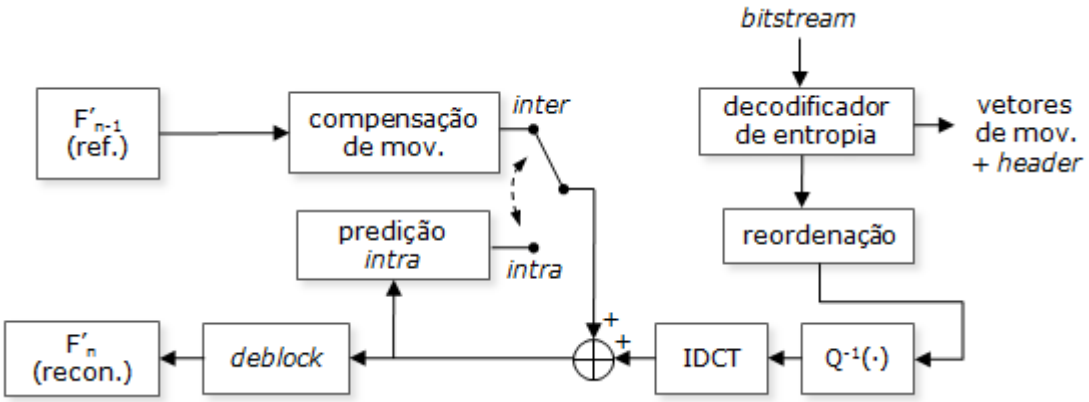

Figura 3.10 – Unidades funcionais típicas de um decodificador do padrão H.264.

O padrão H.264/AVC definiu, à época em que foi estabelecido (2003), um conjunto de três perfis de operação, que determinam um grupo de técnicas e algoritmos que podem ser usados para gerar um *bitstream*. Os perfis foram denominados *Baseline*, *Main* e *Extended*. Cada perfil é voltado para um

tipo de aplicação, e essa classificação facilita o desenvolvimento de codificadores e decodificadores específicos e que mantenham boa compatibilidade entre si. Entre as aplicações do perfil *Baseline* estão aquelas que necessitam de baixa complexidade computacional e alta resiliência como videotelefonia e comunicações sem fio em geral. O perfil *Main* já é voltado para aplicações em que a compressão eficiente é o objetivo maior como armazenamento de vídeo e teledifusão. O conjunto de técnicas do perfil *Extended* pode ser visto como um compromisso entre os dois últimos, sendo este voltado para o *streaming* de mídia. Nesta situação pode haver banda suficiente para suportar resoluções maiores de vídeo, mas as perdas na transmissão comuns em redes como a Internet tornam necessária a adoção de técnicas de resiliência.

Em 2005, foi adicionado à especificação o perfil *High* e suas variações, que, entre outras características, fazem uso de um conjunto de extensões denominado *Fidelity Range Extensions* (FRExt) para aumentarem a eficiência do codificador. Atualmente, a especificação conta com 17 perfis, destacando-se os que dão suporte a *Scalable Video Coding* (SVC, conjunto de técnicas adaptativas de compressão) e *Multiview Video Coding* (MVC, voltado para visão estereoscópica e de múltiplas vistas).

Dentro dos limites impostos pela sintaxe de um dado perfil, ainda existe a questão da variação de performance do codificador e do decodificador dependendo de valores assumidos por parâmetros da sintaxe. Para estabelecer padrões de complexidade, o H.264/AVC especifica para cada perfil o que são chamados de níveis. Os níveis especificam restrições numéricas em parâmetros ou combinações de parâmetros. Dessa maneira, é possível julgar a compatibilidade de codificadores e decodificadores com relação a uma capacidade de performance requerida.

A seguir serão analisadas as técnicas de predição em blocos, por serem o alvo de interesse das implicações deste trabalho. Para tal, é necessário introduzir o conceito de *slices* e tipos de macroblocos. *Slice* é um conjunto de macroblocos ordenados (não necessariamente contínuos) em um quadro, determinando uma região. Os macroblocos podem ser do tipo I ou P. Os do tipo I são codificados com predição *intra*, e os do tipo P com predição *inter*. Os *slices* podem ser do tipo I, contendo apenas macroblocos I, ou do tipo P, contendo macroblocos I ou P. Há ainda macroblocos do tipo B e *slices* do tipo SP e SI, mas estes não foram usados nos experimentos e não serão abordados. Serão focadas apenas técnicas disponíveis no perfil *Baseline*, utilizado nos experimentos.

3.7 PREDIÇÃO *INTER*

Na predição *inter* os blocos são estimados a partir de um ou mais *frames* previamente codificados e reconstruídos utilizando compensação de movimento. Entre as inovações do H.264/AVC nesta área estão a possibilidade de variação do tamanho dos blocos em função das características de cada região da imagem e a precisão sub-amostra dos vetores de movimento.

A variação dos tamanhos dos blocos se dá com a divisão dos macroblocos em blocos, e estes em estruturas menores. Por conta da topologia ramificada, este método é chamado de compensação de movimento em estrutura de árvore. A Figura 3.11 ilustra as possibilidades de partição para as amostras de luminância. O macrobloco (16 x 16 amostras) pode ser quebrado em duas partições de 8 x 16 (ou 16 x 8) ou em quatro de 8 x 8, como mostrado na Figura 3.11(a). No segundo caso, os blocos de 8 x 8 podem ser subdivididos analogamente, como na Figura 3.11(b), gerando o que são chamados sub-macroblocos. As partições das componentes Cb e Cr são análogas e possuem metade das dimensões.

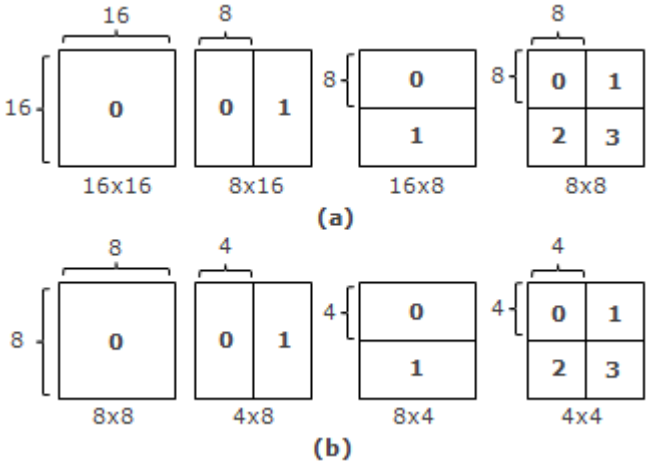

Figura 3.11 – Possibilidades de partição na luminância de a) macroblocos e b) sub-macroblocos.

A escolha da partição pode afetar diretamente a performance da compressão. Cada uma das estruturas geradas necessita de um vetor de movimento separado e de uma sinalização do método de partição, e a codificação desta informação adicional deve ser levada em conta. Caso se opte por partições de tamanho maior (e.g., 16 x 8), serão necessários poucos bits indicando os vetores de movimento e o tipo de partição escolhido, mas o resíduo da compensação de movimento poderá apresentar energia considerável se a região codificada for muito detalhada. A escolha de partições de tamanho reduzido (e.g., 4 x 4) implica na necessidade de mais bits para descrição dos vetores de movimento e do tipo de partição, mas resulta em geral em um resíduo com pouca energia. Em geral, partições de tamanho maior são apropriadas para áreas homogêneas, e as de tamanho menor para áreas com maior detalhe. A Figura 3.12, gerada com o programa *StreamEye* (Elecard, Inc.), ilustra a partição de um *frame* de tamanho 352 x 288 *pixels* da sequência de teste "foreman" codificada a 200 kbps no perfil *Baseline*. Nota-se a utilização de partições maiores nas regiões homogêneas ao fundo e menores nas regiões mais detalhadas ao centro.

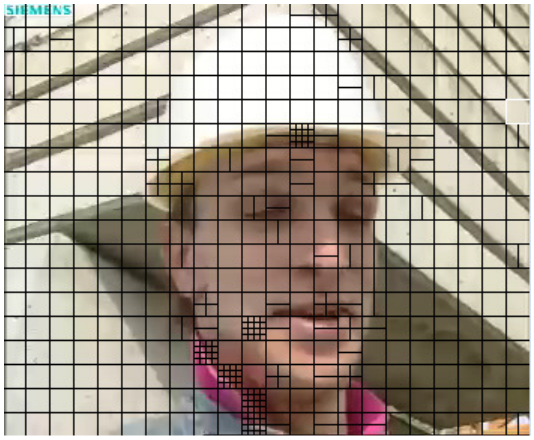

Figura 3.12 – Partições em um *frame* codificado com predição *inter*.

A diferença de posição entre o bloco processado e o bloco do qual ele foi estimado no *frame* de referência é medida com precisão de um quarto de amostra na componente Y e um oitavo nas componentes de crominância (devido à subamostragem). As amostras em posições intermediárias são geradas através da interpolação de amostras vizinhas. A grande quantidade de vetores de movimento gerada pode prejudicar a eficiência do processo de compressão, portanto sua representação deve ser otimizada. Os vetores de movimento de partições vizinhas são altamente correlacionados, portanto é possível realizar estimativas a partir de partições previamente codificadas. Essa técnica é denominada predição de vetores de movimento (MVp) e consiste em codificar a diferença entre o vetor de movimento real e uma estimativa feita a partir de vetores de partições vizinhas.

3.8 PREDIÇÃO *INTRA*

No modo *intra* as predições são feitas a partir de blocos que estejam no mesmo *slice* e que tenham sido previamente codificados e reconstruídos. Portanto, não são utilizadas informações de outros quadros. São formados blocos de tamanho 4 x 4, 8 x 8 ou 16 x 16 amostras para a componente de luminância, e de tamanho 8 x 8 para as componentes de crominância. Diferentes métodos de interpolação e extrapolação a partir das amostras vizinhas são usados para gerar um bloco. Esse bloco estimado é subtraído do bloco real, formando o resíduo que será codificado. O método usado para estimar as amostras é geralmente escolhido como aquele que minimiza a energia do resíduo.

Serão descritos os métodos de predição para um bloco de 4 x 4 por serem os mais gerais. Para os demais tamanhos os métodos são simplificações do caso 4 x 4. A Figura 3.13(a) mostra um bloco a ser estimado, denominado P, cujas amostras são identificadas por letras minúsculas, e as amostras vizinhas previamente codificadas, identificadas por letras maiúsculas. As amostras de P (*a* a *p*) são calculadas a partir de extrapolações ou interpolações envolvendo as amostras *A* a *M*, de acordo com o método escolhido. Há nove métodos disponíveis, identificados por um número de 0 a 8. O método 2 é denominado DC e consiste em uma média simples entre algumas amostras *A* a *M*, dependendo de quais estão disponíveis. Os demais só são utilizados se as amostras necessárias estiverem disponíveis, e a direção e sentido de interpolação (ou extrapolação) para cada um são mostrados na Figura 3.13(b).

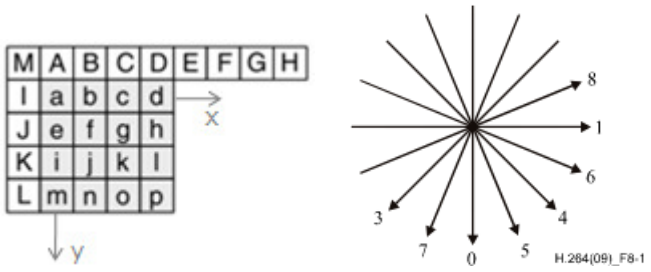

Figura 3.13 – a) Identificação das amostras a serem estimadas e b) direção e sentido da interpolação (ou extrapolação) nos métodos de predição (exceto DC) [2].

Os métodos 0 e 1 são simples e consistem em replicar respectivamente a linha A...D ou a coluna I...L. Os demais métodos são definidos como uma média ponderada de maneira um pouco mais complexa, que será exemplificada com o método 3. Adotando-se um eixo de coordenadas x e y com origem na amostra a como indicado na Figura 3.13(a), define-se a função $p(x,y)$, com $x = -1 \dots 7$ e $y = -1 \dots 3$, que mapeia cada amostra a a p e A a M de acordo com sua posição. O valor de cada amostra do bloco P é dado no método 3 por

$$pred(x,y) = \begin{cases} \dfrac{p(6,-1) + 3p(7,-1)}{4}, & x = y = 3 \\ \dfrac{p(x+y,-1) + 2p(x+y+1,-1) + p(x+y+2,-1)}{4}, & c.c. \end{cases} \quad (3.8)$$

A amostra *e*, por exemplo, é calculada neste caso como $(B + 2C + D)/4$. Nota-se que a direção indicada na Figura 3.13(b) é aproximadamente seguida.

A escolha entre esses vários métodos deve ser indicada ao decodificador, e isso implica em informação adicional a ser inserida no *bitstream*, prejudicando a taxa de compressão. Essa penalização é minimizada estimando-se o método mais provável para aquele bloco e codificando a diferença entre o método estimado e o real. Esta técnica é válida pois há alta correlação entre os métodos de codificação *intra* entre blocos 4 x 4 vizinhos.

4 EXPERIMENTOS

4.1 CALIBRAÇÃO DE LENTES *FISHEYE*

4.1.1 DESCRIÇÃO DO MODELO E ALGORITMO UTILIZADOS

A primeira parte dos experimentos realizados envolveu a calibração e correção da distorção de lentes *fisheye*, discutidos no capítulo 2. A "distorção" referida, como já mencionado, se refere ao desvio da projeção *fisheye* com relação à projeção retilínea. Essa distorção é majoritariamente radial, e a calibração é o processo de ajustar um modelo matemático a este comportamento. A correção é simplesmente o ato de se aplicar o modelo e obter uma imagem retilínea a partir de uma distorcida.

O modelo escolhido foi o da divisão polinomial (DM) por representar satisfatoriamente distorções moderadas a fortes com apenas um parâmetro e pela facilidade na sua manipulação matemática. O modelo é repetido por conveniência:

$$r_p = \frac{r_f}{1 + \lambda r_f^2}$$
$$r_f = \frac{1 - \sqrt{1 - 4\lambda r_p^2}}{2\lambda r_p} \tag{4.1}$$

onde r_p é o raio no domínio retilíneo (projeção perspectiva, sem distorção), r_f é o raio distorcido e λ é o parâmetro a ser calibrado e que determina a quantidade de distorção. A primeira forma da equação, que leva do domínio *fisheye* ao domínio retilíneo, será tratada como forma direta, e a segunda, como forma indireta. A aplicação deste modelo consiste em uma transformação espacial não-linear.

O método de calibração escolhido foi o de se mapear retas no domínio retilíneo em arcos de círculo no domínio DM, como descrito em [22]. Essa importante propriedade do modelo é agora discutida.

Supondo o centro da distorção (COD) como o centro da imagem, pode-se reescrever a forma direta de (4.1) em coordenadas cartesianas como:

$$x_p = \frac{x_f}{1 + \lambda r_f^2} \quad \text{e} \quad y_p = \frac{y_f}{1 + \lambda r_f^2} \tag{4.2}$$

onde $r_f^2 = x_f^2 + y_f^2$. Tomando-se a equação de uma reta no domínio retilíneo como $y_p = kx_p + b$, onde k é a inclinação e b o valor da ordenada para a qual a abscissa se anula, vem

$$\frac{y_f}{1 + \lambda r_f^2} = k\frac{x_f}{1 + \lambda r_f^2} + b \tag{4.3}$$

que pode ser rearranjada como

$$x_f^2 + y_f^2 + \frac{k}{b\lambda}x_f - \frac{1}{b\lambda}y_f + \frac{1}{\lambda} = 0 \tag{4.4}$$

que é a equação de um círculo que passa pela origem no domínio *fisheye*. O processo de calibração se resume, portanto, a ajustar arcos de círculo em linhas na projeção *fisheye* que correspondem a retas no domínio retilíneo, determinando-se o parâmetro λ.

Com discutido em 2.4, porém, não é correto supor que o COD coincide com o centro da imagem, portanto sua posição também deve ser estimada. Tomando-se as coordenadas do COD como (x_0, y_0) pode-se estender (4.4) para

$$(x_f - x_0)^2 + (y_f - y_0)^2 + \frac{k}{b\lambda}(x_f - x_0) - \frac{1}{b\lambda}(y_f - y_0) + \frac{1}{\lambda} = 0 \tag{4.5}$$

que pode ser reescrita como

$$x_f^2 + y_f^2 + Ax_f - By_f + C = 0 \tag{4.6}$$

fazendo

$$\begin{aligned} A &= \frac{k}{b\lambda} - 2x_0 \\ B &= -\frac{1}{b\lambda} - 2y_0 \\ C &= x_0^2 + y_0^2 - \frac{k}{b\lambda}x_0 + \frac{1}{b\lambda}y_0 + \frac{1}{\lambda} \end{aligned} \tag{4.7}$$

O conjunto de equações acima demonstra que, no domínio *fisheye*, a uma curva que deveria ser reta no domínio retilíneo pode-se ajustar um arco de círculo determinado por um grupo de parâmetros (A_i, B_i, C_i). Ao se extraírem várias curvas da imagem, o COD pode ser estimado a partir da manipulação de (4.7) como

$$\begin{cases} (A_1 - A_2)x_0 + (B_1 - B_2)y_0 + (C_1 - C_2) = 0 \\ (A_1 - A_3)x_0 + (B_1 - B_3)y_0 + (C_1 - C_3) = 0 \\ (A_2 - A_3)x_0 + (B_2 - B_3)y_0 + (C_2 - C_3) = 0 \\ \vdots \end{cases} \tag{4.8}$$

que consiste em um sistema linear montado a partir de todas as combinações possíveis entre os conjuntos de parâmetros. Os parâmetros (A_i, B_i, C_i) podem ser ajustados a (4.6) de diversas maneiras. Optou-se pelo método dos mínimos quadrados pela sua simplicidade na implementação pelo processo de aquisição das retas ser pouco automatizado, como será descrito posteriormente. Para uma curva com conjunto de pontos (x_i, y_i) no domínio *fisheye*, tem-se

$$(x_i, y_i, 1) \cdot \begin{pmatrix} A \\ B \\ C \end{pmatrix} = -\left(x_i^2 + y_i^2\right). \tag{4.9}$$

Ainda de (4.7), a estimação do parâmetro λ é dada por

$$\lambda = \frac{1}{x_0^2 + y_0^2 + Ax_0 + By_0 + C}, \tag{4.10}$$

sendo que esta equação pode ser interpretada como de maneira escalar para cada curva, gerando parâmetros λ_i independentes, ou como um sistema linear gerando uma otimização a partir de todas as curvas

$$(x_0^2 + y_0^2 + \boldsymbol{A}x_0 + \boldsymbol{B}y_0 + \boldsymbol{C}) \cdot \lambda = \begin{pmatrix} 1 \\ 1 \\ \vdots \end{pmatrix} \tag{4.11}$$

onde **A**, **B** e **C** são vetores-coluna contendo os respectivos parâmetros estimados para cada curva.

Em suma, o método adotado consiste em

1) Na imagem *fisheye*, extrair *n* ($n \geq 3$) curvas que correspondem a retas no domínio retilíneo.
2) Para cada curva, determinar (A_i, B_i, C_i), ajustando-as a arcos de círculos pelo método dos mínimos quadrados (Eq. (4.9)).
3) Estimar o COD pela Eq. (4.8).
4) Estimar o parâmetro λ pela Eq. (4.11) ou por uma otimização.

O passo 4 no algoritmo desenvolvido é ampliado para que todos os λ estimados sejam testados (incluindo o otimizado), e o que obtiver melhor resultado na correção é escolhido. O método de comparação foi a colinearidade dos pontos corrigidos. Após aplicar a forma direta de (4.1) aos pontos extraídos no passo 1, as retas resultantes são analisadas para determinar sua colinearidade. A colinearidade foi analisada calculando-se primeiro uma regressão linear dos pontos que pertenciam a uma mesma reta. As distâncias Euclidianas de cada ponto à sua reta determinada pelos parâmetros da regressão são somadas, determinando uma medida do quão eficiente foi a correção.

O passo 1 do algoritmo foi feito tanto de maneira manual, para imagens obtidas de terceiros, como de maneira automatizada, para as imagens obtidas no laboratório. O processo será descrito a seguir.

4.1.2 FLUXO DE TRABALHO

Para os testes de calibração, foram utilizadas imagens adquiridas no laboratório e outras disponíveis em sites especializados na Internet. Para a captura das imagens no laboratório, foi utilizado como equipamento uma câmera fotográfica digital do tipo *single-les reflex* (SLR) semi-profissional da marca Sony, modelo DLSR-A200, e uma lente adaptadora *fisheye* com multiplicador focal 0.2X da marca Opteka. Uma lente adaptadora apenas multiplica a distância focal de outra lente na qual esteja montada, fazendo com que esta passe a operar de maneira distinta. O equipamento está ilustrado na Figura 4.1. A distância focal utilizada na lente que acompanha a A200 foi de 35 mm, e a altura de seu sensor é de 15,8 mm. Assumindo uma projeção equisólida no adaptador (não é informado em sua documentação), resolvendo para o ângulo de incidência θ na Eq. (2.5) e multiplicando por 2, o campo de visão do sistema é de aproximadamente 140°.

Figura 4.1 – Equipamento utilizado para captura de imagens de teste: a) câmera digital Sony com adaptador montado e b) adaptador *fisheye* Opteka.

Com as imagens capturadas, aplicou-se o algoritmo de calibração descrito na seção anterior. Os códigos para este trabalho foram desenvolvidos no ambiente MATLAB® 7.9 (Mathworks, Inc.). A escolha do ambiente se deu pela facilidade das operações de entrada e saída com imagens e a notação matricial de sua linguagem, que facilita as operações de transformação.

O primeiro passo do algoritmo utilizado é a extração de curvas na imagem que correspondem a retas no domínio retilíneo. Em [17] é descrito um processo automatizado para isso, mas sua implementação envolve a detecção de bordas com precisão sub-pixel, conexão de bordas (*edge linking*) e um ajuste iterativo para determinar as curvas de interesse. Ainda assim são detectados vários *outliers* e é necessário um algoritmo mais robusto como RANSAC ou Levenberg-Marquardt para o ajuste dos pontos em arcos de círculos. Dada a complexidade apresentada, optou-se pela seleção manual dos pontos, visto que é um processo que só deveria ser requerido uma vez em aplicações práticas, i.e., uma vez determinados os parâmetros para uma combinação câmera-lente pode-se somente reutilizá-los toda vez que se for aplicar a correção.

Para auxiliar o processo, foi desenvolvida uma aplicação simples com auxílio de ferramentas disponíveis na *toolbox* de processamento de imagens do ambiente MATLAB®. A aplicação é ilustrada na Figura 4.2. Caso a imagem seja do tipo *fisheye* circular, é necessário indicar a posição do círculo efetivo da imagem, como em outros softwares de calibração de câmeras (e formação de panoramas) como PTGui (New House Internet Services BV) e Hugin (*open source*). Em seguida, é extraída uma reta por vez, selecionando-se os pontos e confirmando a operação. As retas anteriores são indicadas com marcadores coloridos.

Este método funcionou de maneira adequada para imagens obtidas na Internet com lentes profissionais e de maior qualidade, mas para as imagens do laboratório foi notada uma imprecisão na calibração

feita a partir da extração manual de pontos, como será discutido no capítulo seguinte. A solução adotada foi o uso de padrões de calibração que fossem facilmente detectados de maneira automática. Um exemplo de padrão adotado é ilustrado na Figura 4.3(a). Sua detecção consistiu em uma demarcação manual da área de interesse (Figura 4.3(b)), para em seguida binarizar (representar com apenas dois níveis) a imagem a partir de um limiar global, invertê-la e, por fim, detectar todas as formas binárias existentes, que correspondem às marcações pretas. Foram utilizadas as coordenadas dos centróides das áreas detectadas para estabelecer um *grid* deformado (Figura 4.3(c)). Os pontos desse *grid* foram conectados em retas com um algoritmo recursivo.

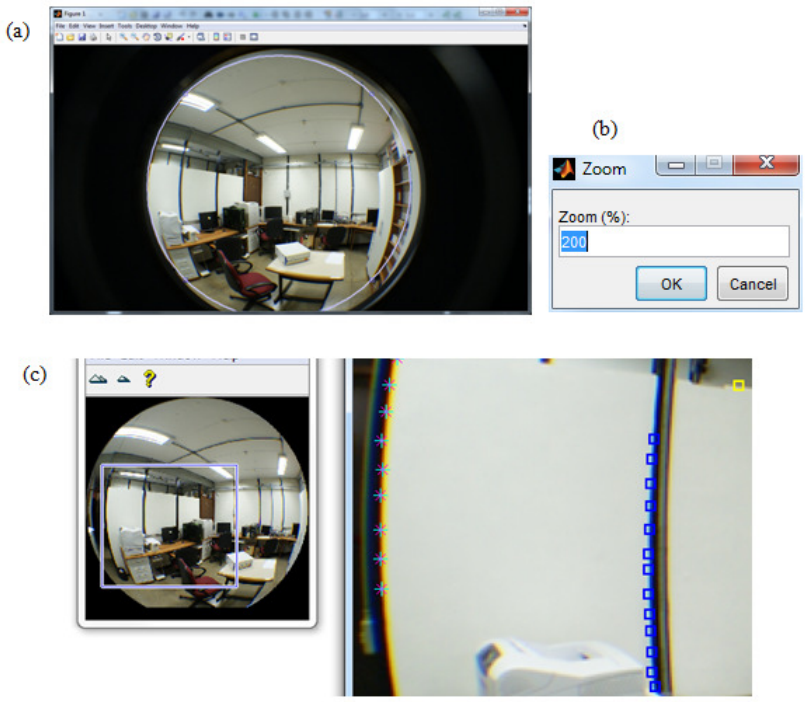

Figura 4.2 – Rotina de extração de linhas manualmente: a) seleção do círculo de imagem efetivo, b) entrada do zoom para o passo posterior e c) seleção de pontos na figura ampliada (esq.) e janela de navegação (dir.).

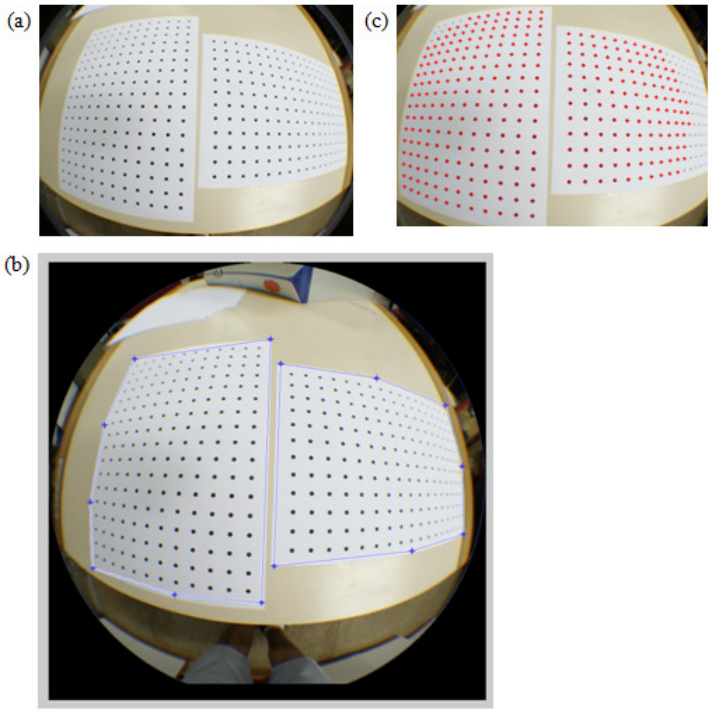

Figura 4.3 – Rotina de extração de linhas automaticamente a partir de padrões de calibração.

O algoritmo recursivo parte de cada ponto do *grid* e seleciona um segundo ponto entre os vizinhos deste. A cada iteração, tenta-se selecionar outro ponto dentre os vizinhos do anterior de maneira que a variação da inclinação seja inferior a um limite imposto a partir de testes. Caso o ponto inicial possua dois vizinhos com mesma inclinação, ambos são usados para se obter um conjunto maior. São ainda descartadas combinações repetidas de pontos. Ao final, têm-se vários conjuntos de pontos que seriam colineares em um domínio retilíneo, portanto são usados como curvas de interesse para o algoritmo de calibração.

Os parâmetros estimados pela calibração (λ e COD) são utilizados para aplicar a transformação espacial na imagem dada pela Eq. (4.1). O mapeamento é feito como descrito na seção 2.4, ou seja, utiliza-se a forma direta para estimar as dimensões da saída, a forma inversa para fazer o mapeamento, e um método de interpolação para se determinar os valores. Os resultados obtidos são detalhados no capítulo 5.

4.2 COMPARATIVO DE COMPRESSÃO ENTRE IMAGENS *FISHEYE* E RETILÍNEAS

Como aplicação do método de calibração desenvolvido, propôs-se uma comparação da performance de compressão com o padrão H.264/AVC entre sequências de imagens com projeção *fisheye* e com projeção perspectiva (retilínea). Dado que as técnicas de compressão são aplicadas em blocos retangulares e quadrados da imagem, espera-se que as imagens retilíneas obtenham melhores resultados.

As sequências de imagens consistiram em fotos da mesma cena com bastante sobreposição, simulando os *frames* de um vídeo. Não foram encontradas câmeras de vídeo capazes de gravação com baixa compressão e que fossem compatíveis com o adaptador *fisheye* utilizado. Foram utilizadas somente imagens do tipo *fisheye* circular, pois nelas a retificação guarda a proporção das dimensões da imagem (quadrada, circunscrevendo o círculo), podendo-se gerar imagens retificadas com o mesmo tamanho da original. Em imagens *full-frame*, a retificação altera a forma da imagem (ver seção 5.1) e a área útil possui razão de aspecto diferente da imagem original. As imagens foram convertidas do espaço de cores RGB para o espaço de cores YUV, padrão em aplicações de compressão.

A codificação dos vídeos foi feita com o *software* de referência JM [31] para o padrão H.264/AVC. Não foi feita nenhuma modificação no código, sendo alterados apenas parâmetros de codificação em arquivos de configuração.

Os processos que seriam analisados a princípio estão ilustrados na Figura 4.4. As sequências em projeção *fisheye* são denotadas pela letra F, e as retificadas pela letra R. No processo FISH1, uma sequência de imagens *fisheye* é codificada e decodificada. São medidas a distorção (PSNR) e taxa de compressão (total de bits por total de *pixels*) da sequência de saída. No processo FISH2, a sequência é retificada antes da codificação utilizando a calibração descrita na seção anterior. Após ser decodificada, ela é retornada à projeção *fisheye* original. Esta comparação não se mostrou justa ao analisar-se que somente o processo de mapeamento e re-mapeamento das imagens introduz perdas consideráveis. A PSNR média entre uma imagem que passe por esse processo e sua original é 30 dB, que demonstra uma reconstrução pobre. Isso se deve ao fato de que regiões no centro do círculo *fisheye*, com maior resolução espacial, passam a ocupar menor espaço ao serem retificadas. Ao se desfazer o mapeamento, as regiões agora de baixa resolução devem voltar a ocupar mais espaço, o que leva a grande perda de detalhes. Um processo análogo é reduzir o tamanho de uma imagem em 50% (um quarto da área total) para em seguida retornar ao tamanho original. O processo de interpolação realizado no aumento não é capaz de recuperar os detalhes originais, e a PSNR entre as imagens redimensionada e original é em torno de 30 dB também.

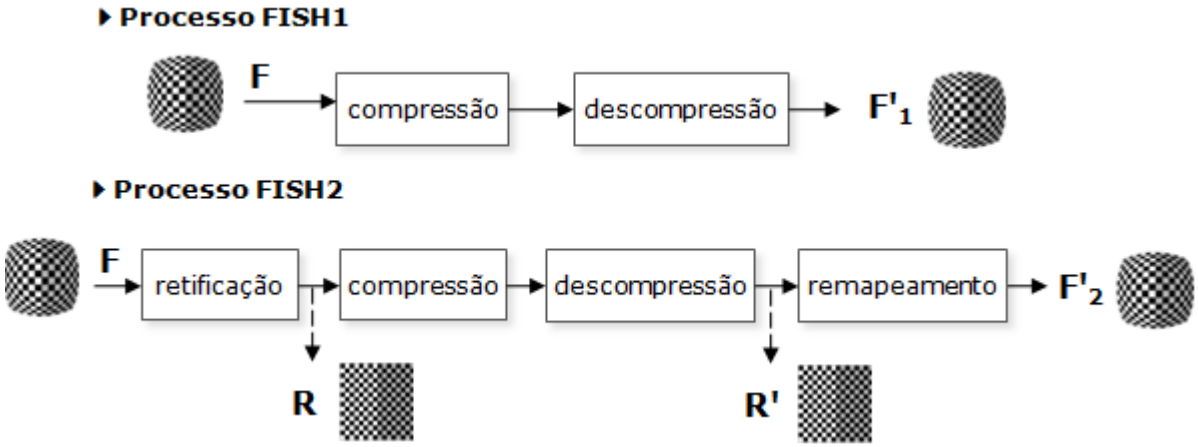

Figura 4.4 – Cadeias de processos originalmente propostas para análise de compressão de imagens *fisheye*.

Passou-se então a considerar processos que tivessem como saída a imagem retificada, evitando as perdas do mapeamento inverso. A Figura 4.5 ilustra os processos que serão comparados de fato nesta parte do trabalho, com a mesma notação da Figura 4.4. No processo RET1, a sequência é retificada após ser codificada e decodificada. No processo RET2, ela é retificada antes das etapas de compressão. Nota-se que em ambos os casos a distorção da sequência de saída é calculada com relação à imagem retificada marcada como R. A taxa de compressão é obtida do *bitstream* gerado em cada processo.

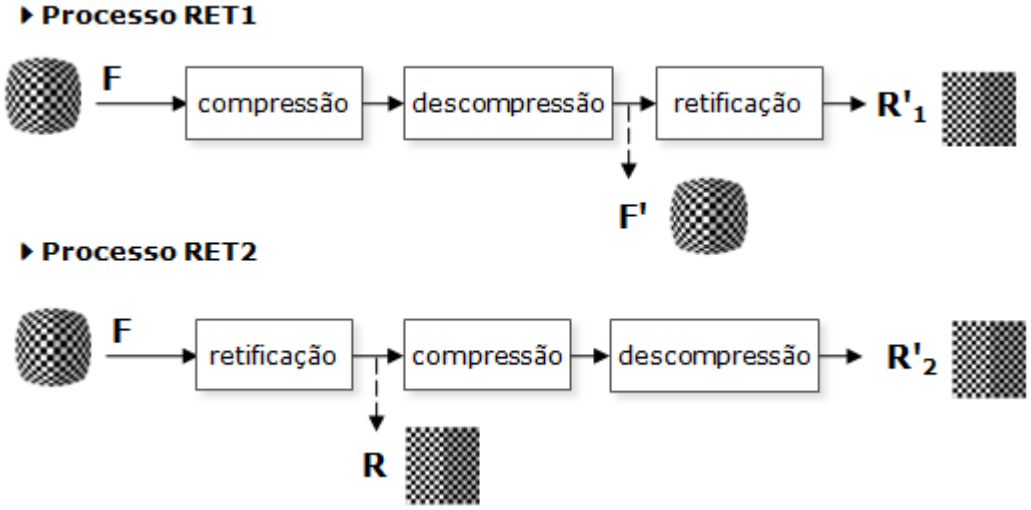

Figura 4.5 – Cadeias de processos aplicados na análise comparativa de compressão de imagens *fisheye*.

Foram feitos testes com duas sequências de imagens, embora do mesmo cenário (uma sala no laboratório). Para a primeira sequência, a retificação no processo RET1 é feita no espaço de cores YUV, para evitar as perdas inerentes da conversão YUV-RGB. Todos os testes utilizaram o perfil *Baseline* no nível 3. As sequências foram codificadas de quatro maneiras: com predição *intra* em todos *frames* ou somente no primeiro, e com otimização de taxa-distorção (RDO) habilitado ou desabilitado.

4.3 COMPARATIVO DE COMPRESSÃO ENTRE IMAGENS *FISHEYE* E PANORÂMICAS

O segundo comparativo proposto visa analisar a melhor alternativa na transmissão de imagens a fim de se ter um panorama no receptor. Tem-se em mente como aplicação sistemas que necessitam gerar e em seguida armazenar ou transmitir imagens panorâmicas em sequência. Estes sistemas necessitam gerar o panorama a partir de um número limitado de câmeras, portanto o uso de lentes *fisheye* é

atrativo dado seu campo de visão ampliado. Após a captura das imagens, pode-se optar entre as duas cadeias de processamento descritas na Figura 4.6. No processo PANO1, as imagens são codificadas antes do armazenamento ou transmissão para serem posteriormente utilizadas na geração de um panorama. No processo PANO2, gera-se o panorama para então codificá-lo e armazená-lo ou transmiti-lo.

Foram feitos testes para dois grupos de 6 imagens *fisheye*, disponíveis em páginas especializadas em panoramas na Internet. As imagens estavam no formato *fisheye full-frame* e possuíam áreas de sobreposição de 20% entre si. Foi utilizada a ferramenta *open source* Hugin [28] para automatização da formação de panoramas. Esta ferramenta consiste em um conjunto de aplicações que realizam etapas distintas da criação de panoramas, e que podem ser controladas de maneira independente pela linha de comando, possibilitando a criação de rotinas.

Um panorama foi criado a partir das imagens de entrada sem nenhuma interferência manual, e este serviu de base para as comparações de distorção. As coordenadas dos pontos de correspondência detectados entre as imagens e demais opções como formato do arquivo são guardados em um arquivo em formato texto para ser utilizado posteriormente, garantindo uniformidade no processo de criação dos panoramas. Este é um cenário realista pois o cálculo das correspondências é bastante custoso computacionalmente, sendo feito somente para um conjunto de imagens inicial. O arquivo texto é comprimido de maneira automática no formato ZIP, e seu tamanho é levado em conta no cálculo da taxa de compressão.

Para serem codificadas, as imagens foram agrupadas como *frames* de uma sequência e convertidas para o espaço de cores YUV. O software de referência JM foi novamente utilizado como implementação do padrão H.264/AVC. Foi utilizado o perfil *Baseline* no nível 5, dado que as imagens eram de resolução elevada, e foi habilitado o RDO. A compressão foi testada com duas variações: predição *intra* em todos os *frames* e só no primeiro.

A imagem panorâmica também foi convertida para o espaço de cores YUV e comprimida com os mesmos parâmetros, porém apenas com predição *intra*. Procurou-se manter, tanto nas imagens *fisheye* quanto no panorama, dimensões múltiplas de 16, garantindo melhor performance na taxa de compressão.

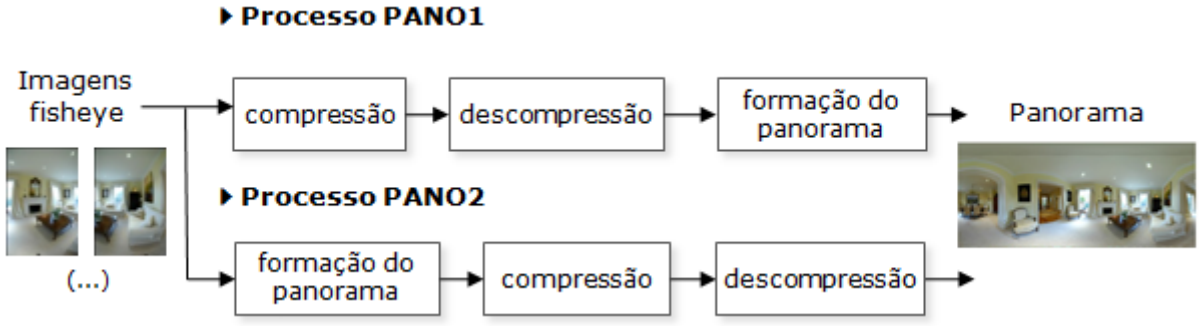

Figura 4.6 – Cadeia de processos para análise comparativa de compressão de panoramas.

5 RESULTADOS E ANÁLISES

5.1 CALIBRAÇÃO DE LENTES *FISHEYE*

Os algoritmos de calibração desenvolvidos foram testados com as imagens do tipo *fisheye* circular capturadas no laboratório e com imagens do tipo *full-frame* disponíveis no sítio da empresa australiana *360 Degrees Of Freedom* [32].

As imagens deste sítio possuíam FOV diagonal de 180º, foram capturadas com uma câmera DSLR profissional Nikon D100 e lente Nikkor, de distância focal 10,5 mm e estavam comprimidas em JPEG com qualidade 91. As curvas foram extraídas destas imagens manualmente com auxílio das rotinas desenvolvidas para tal. A Figura 5.1 ilustra a imagem de teste com as curvas selecionadas destacadas.

Figura 5.1 – Imagem *full-frame* para teste de calibração com as curvas selecionadas destacadas.

As coordenadas dos pontos das curvas foram usadas como entrada no algoritmo descrito na seção 4.1.1, resultando em $\lambda = -2{,}16 \cdot 10^{-6}$ e COD 13,2 px à direita e 33,8 px acima do centro da imagem, cujas dimensões eram 660 x 993 px. Nos gráficos da Figura 5.2 é ilustrada a correção das curvas com e sem estimação do COD, que é indicado por um marcador (asterisco). Os pontos considerados estão indicados por círculos, sobrepostos às retas ajustadas a eles por regressão linear.

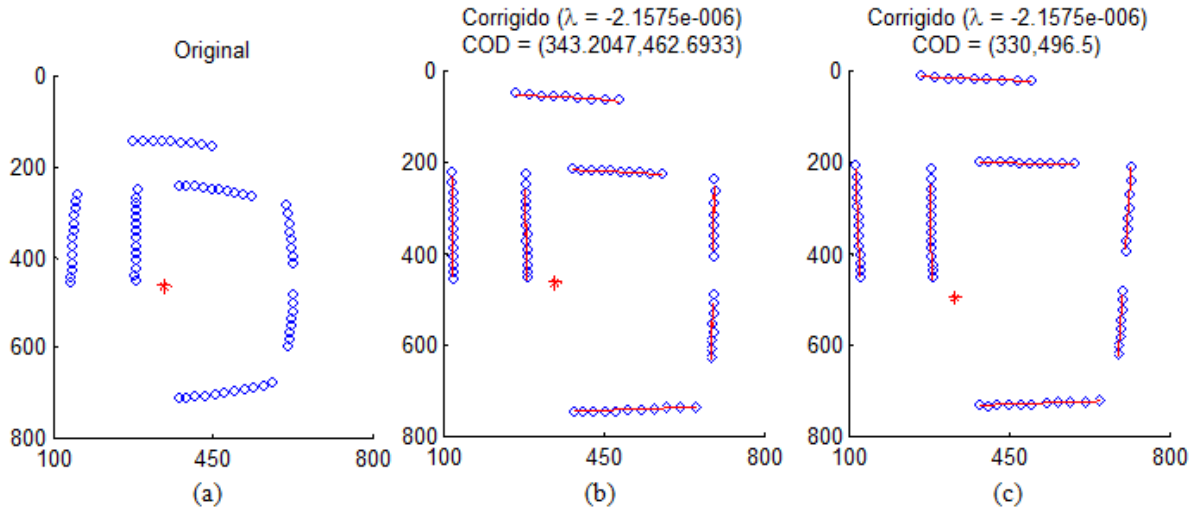

Figura 5.2 – Retificação das curvas extraídas manualmente (eixos em *pixels*): a) pontos extraídos, b) pontos corrigidos com estimação do COD e c) pontos corrigidos sem estimação do COD.

A regressão linear foi calculada com a função `polyfit` disponível no ambiente MATLAB®. A métrica de qualidade estabelecida para a correção foi a colinearidade dos pontos corrigidos, estimada como a soma das distâncias de cada ponto à reta ajustada correspondente. Com a estimação do COD, as distâncias apresentaram média 3,79 px e desvio padrão 1,22 px. Desprezando-se a estimação, estes valores foram 3,93 e 1,76 px. A descentralização vertical do COD estimado correspondeu a cerca de 3,4% da altura da imagem, o que pode ser considerado como significativo para uma lente profissional como a que foi utilizada. Há outras características que levam à descentralização do COD, mas pode-se atribuir essa diferença à sensibilidade do método com relação à distribuição das curvas extraídas, como será visto adiante.

As imagens corrigidas são apresentadas na Figura 5.3. Na Figura 5.3(b) é mostrada a imagem corrigida por inteiro. Como a correção é puramente radial, os cantos da imagem, como estão mais distantes do centro, recebem uma correção mais acentuada, deixando a imagem corrigida com o aspecto mostrado. Nas Figura 5.3(c) e (d) é mostrada a área útil da correção, que corresponde ao maior retângulo inscrito na região de *pixels* válidos da Figura 5.3(b). Nas Figura 5.3(b) e (c) é usado a estimação do COD, e, na Figura 5.3(d), assume-se que o COD coincide com o centro da imagem.

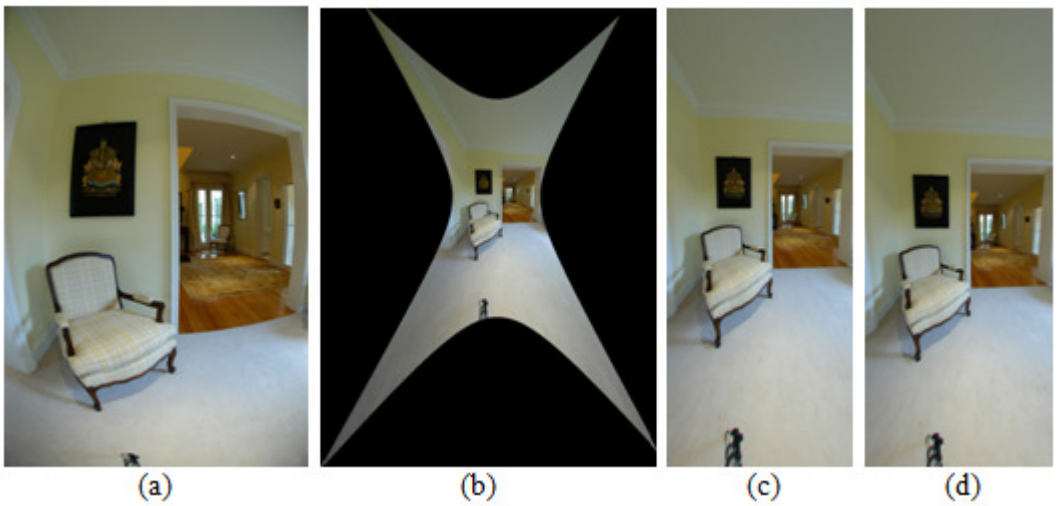

Figura 5.3 – Correção da imagem *fisheye full-frame*: a) original, b) área total, c) área útil e d) área útil desprezando estimação do COD.

Para as imagens do laboratório, tentou-se primeiramente a extração de curvas manual. Na Figura 5.4(a) é mostrada a imagem utilizada com as curvas selecionadas em destaque. Os parâmetros estimados pelo algoritmo de calibração foram $\lambda = -4{,}81 \cdot 10^{-6}$ e COD 6,15 px à direita e 27,51 px acima do centro do círculo da imagem, cujas dimensões eram 732 x 732 px. Na Figura 5.4(b), é mostrado o resultado para a retificação com a estimação do COD, e, na Figura 5.4(c), desprezando-se tal estimação. Em nenhuma imagem capturada no laboratório foi possível obter uma correção satisfatória como em outras que foram capturadas utilizando-se lentes *fisheye* de qualidade profissional. Neste caso, as imagens corrigidas apresentam uma combinação de distorção radial e tangencial, dada a oscilação apresentada por algumas curvas corrigidas.

A fim de investigar este fato e possivelmente melhorar os resultados, foi aplicada então a extração automática de curvas com a ajuda de padrões de calibração. A Figura 5.5(a) ilustra o padrão utilizado, que é o mesmo representado na Figura 4.3. Os pontos marcados em vermelho naquela figura são unidos em 111 curvas representadas na Figura 5.5(b). O algoritmo de calibração estimou como parâmetros $\lambda = -3{,}65 \cdot 10^{-6}$ e COD 9 px à esquerda e 1,22 px acima do centro do círculo da imagem, cujas dimensões eram 732 x 732 px (como no caso anterior). O COD mais próximo ao centro leva a crer que a sua estimação pode estar vinculada à distribuição das curvas extraídas na imagem, que nos casos anteriores estavam dispersas e neste estão quase simétricas em relação ao centro. A Figura 5.6 ilustra a correção radial aplicada aos pontos detectados. A colinearidade das curvas retificadas foi medida e apresentou média 4,46 px e desvio padrão 3,5 px. Desprezando a correção radial, obtém-se um resultado bastante parecido com o da Figura 5.6, e as medidas de colinearidade passam a ter média e desvio padrão 4,8 e 4,32 px, respectivamente.

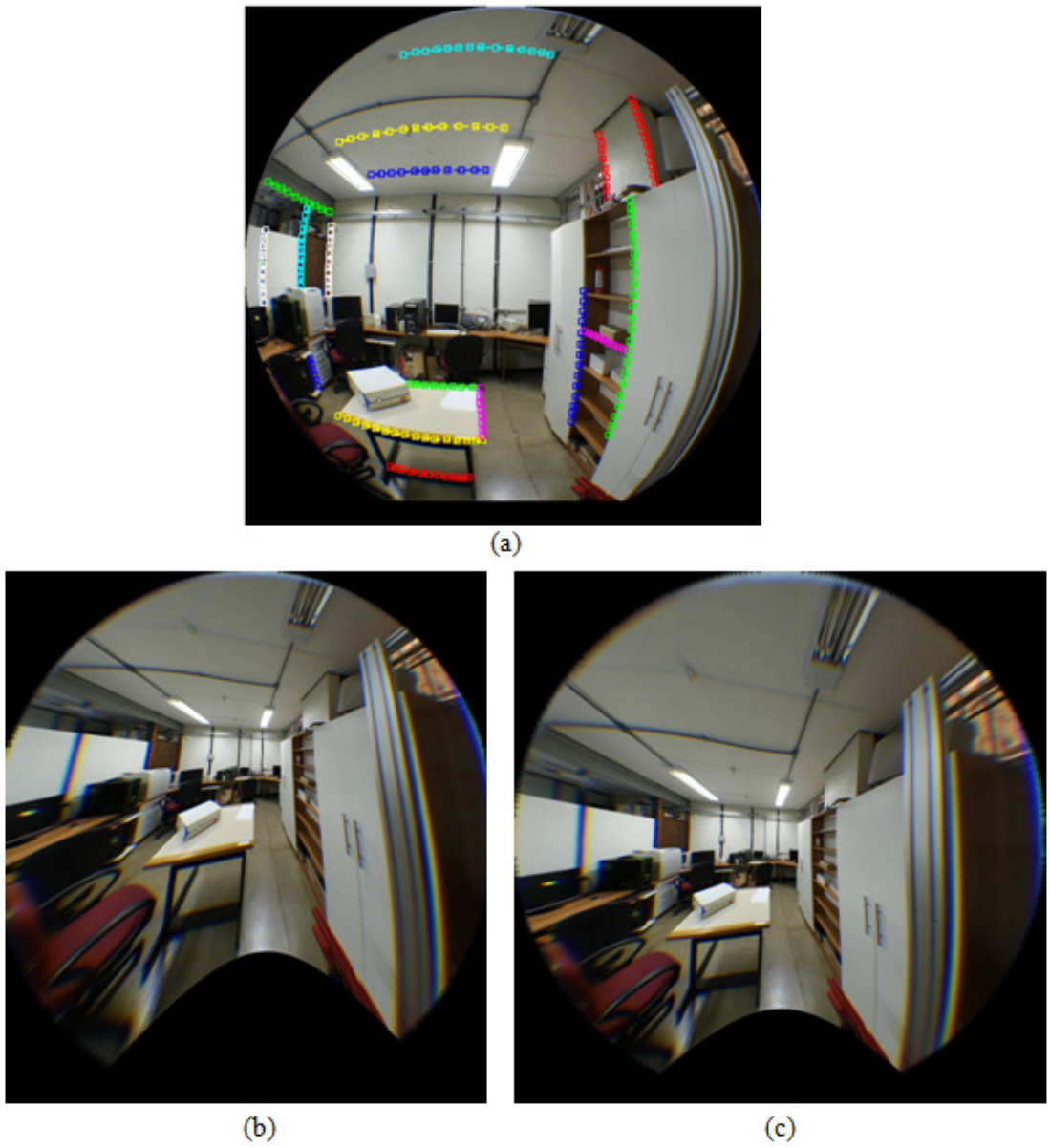

Figura 5.4 – Extração manual de curvas: a) imagem original com curvas destacadas e retificação b) levando em conta o COD estimado e c) desprezando-o.

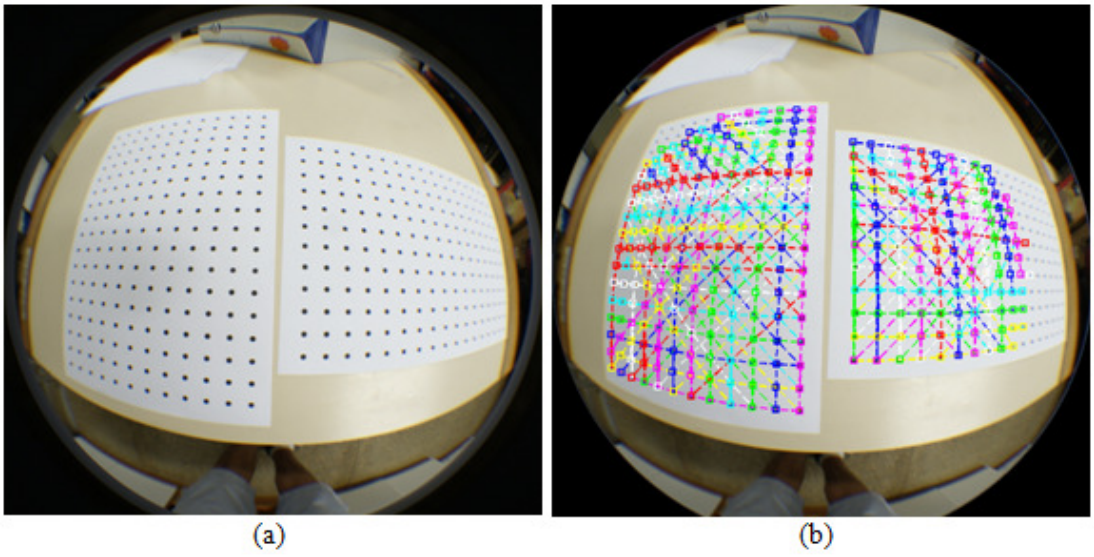

Figura 5.5 – Extração automática de curvas: a) padrão de calibração e b) curvas detectadas.

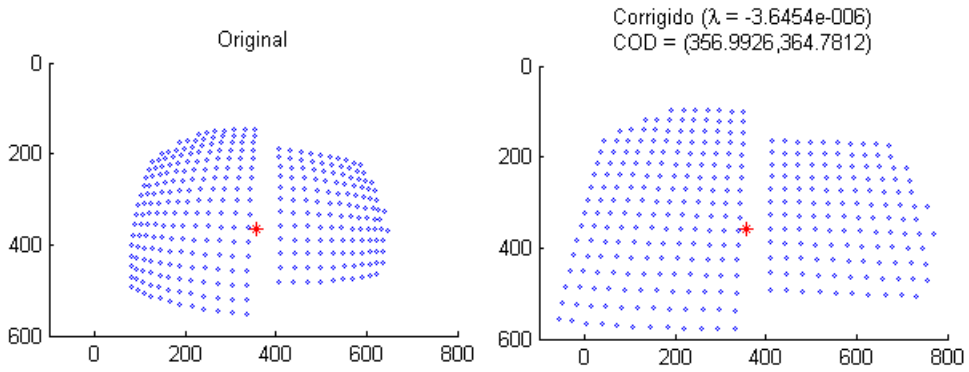

Figura 5.6 – Retificação das curvas extraídas automaticamente (eixos equivalem às coordenadas em *pixels*).

Contudo, o resultado da retificação para este caso, mostrado na Figura 5.7, evidencia que a concentração de curvas extraídas no centro da imagem fez com que apenas a distorção naquela região fosse levada em conta, visto que o aspecto retilíneo diminui à medida que se afasta do centro. Isso leva a crer que a distorção introduzida por esta lente adaptadora não é radialmente uniforme, e não seria possível corrigir ao mesmo tempo regiões mais próximas ao centro e às bordas do círculo de imagem com o modelo utilizado, justificando os resultados insatisfatórios comentados anteriormente. Por apresentar melhor performance de compromisso, os resultados da calibração automática foram utilizados na seção seguinte. A Tabela 5.1 resume os resultados obtidos nesta seção.

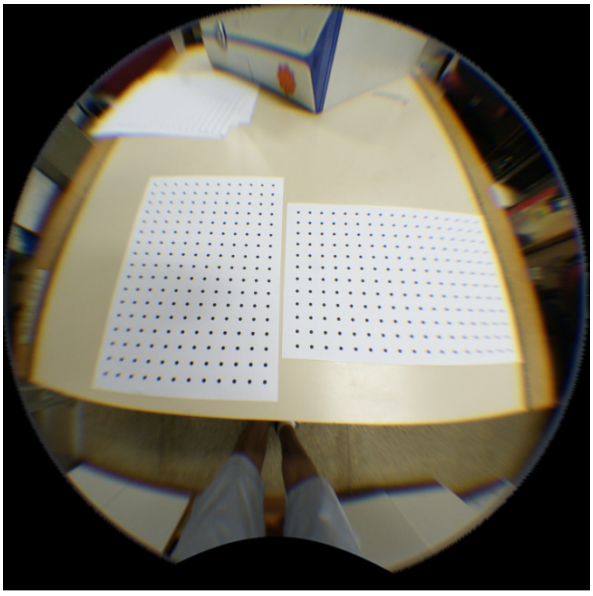

Figura 5.7 – Retificação após calibração automática.

Tabela 5.1 - Resultados obtidos nos experimentos de calibração.

Imagem	Dimensões (altura x largura) (px)	Extração de retas	$\lambda \cdot 10^6$	COD (x_0,y_0) (px)	Distâncias de pontos corrigidos às suas respectivas retas (px)			
					Utilizando COD		Desprezando COD	
					Média	Desvio	Média	Desvio
Figura 5.1	660 x 993	manual	-2,16	(343,2, 462,7)	3,79	1,22	3,93	1,76
Figura 5.4(a)	732 x 732	manual	-4,8	(372,65, 338,99)	11,45	8,10	13,97	10,40
Figura 5.5(a)	732 x 732	automático	-3,65	(356,9, 364,8)	4,46	3,5	4,8	4,32

5.2 COMPRESSÃO DE IMAGENS *FISHEYE* E RETILÍNEAS

Serão comparados os processos RET1 e RET2 da Figura 4.5. No processo RET1, as sequências são retificadas após serem codificadas e decodificadas. No processo RET2, a retificação ocorre antes da codificação e decodificação. A retificação é feita com os parâmetros obtidos na calibração automática descrita na seção anterior. A distorção ao final de ambos os processos é medida com relação à sequência retificada não comprimida, e a taxa de compressão é calculada com base no *bitstream* gerado no processo. As curvas foram geradas com o parâmetro de quantização de *slices* I assumindo os valores 7, 12, 17, 22, 27, 32, 37, 42 e 47. O parâmetro de quantização de *slices* P assumiu os mesmos valores somados a 1. A sequência de imagens *fisheye* é mostrada na Figura 5.8(a), e a sequência retificada na Figura 5.8(b). O conjunto de imagens da Figura 5.8 será identificada como Sequência 1. Foram testadas configurações com predição *intra* somente no primeiro *frame* (*inter* nos outros) e em todos os *frames*, e ainda com otimização de taxa-distorção (RDO) habilitada e desabilitada, totalizando quatro combinações. As comparações entre curvas de PSNR versus taxa dos processos RET1 e RET2 para cada caso são apresentadas na Figura 5.9, identificadas pelo título.

(a)

(b)

Figura 5.8 – Sequência 1 utilizada na comparação: a) *fisheye* e b) retilínea.

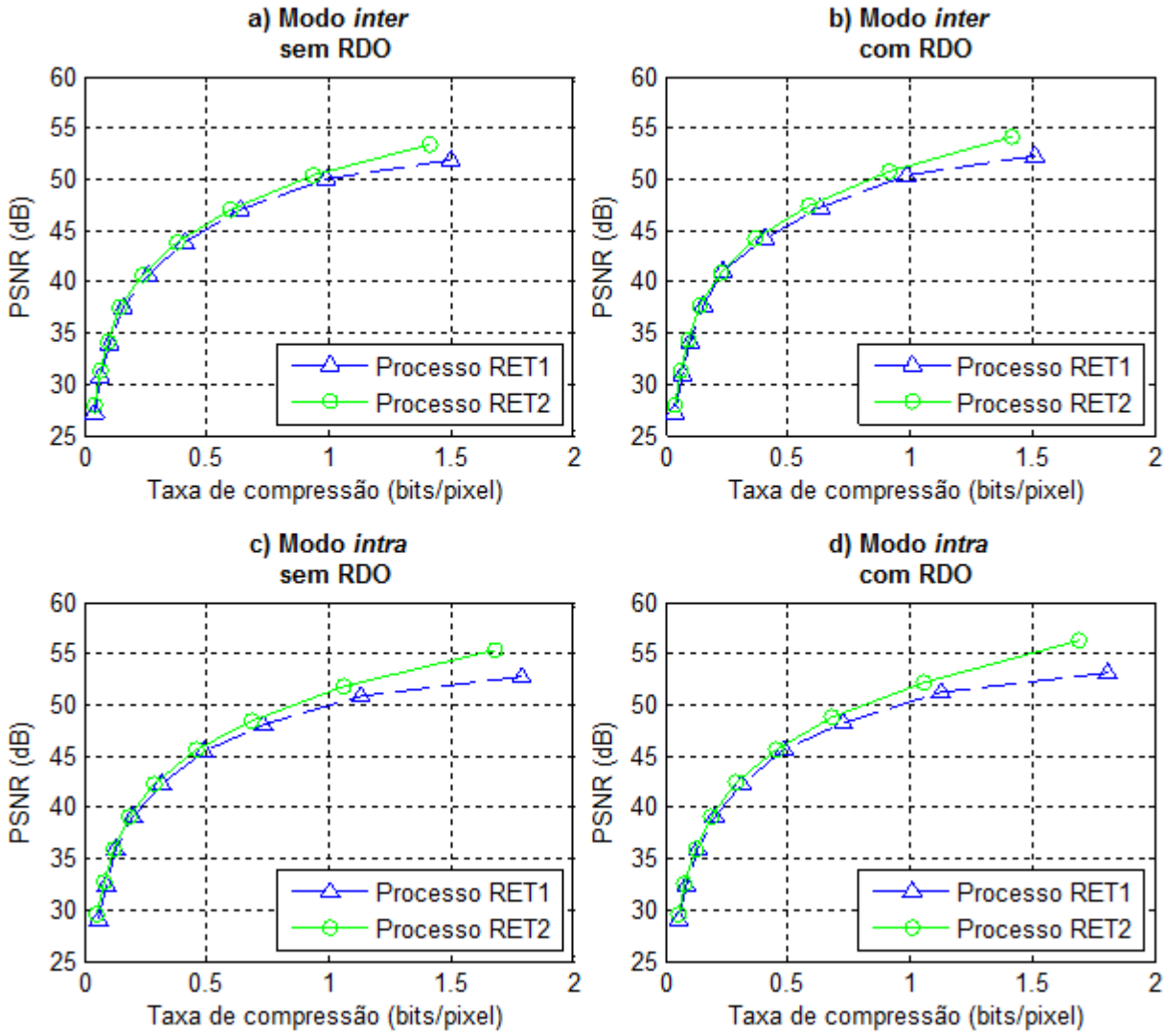

Figura 5.9 – Curvas de PSNR para Sequência 1: com predição *inter* e RDO a) desabilitado e b) habilitado, e somente com predição *intra* e RDO c) desabilitado e d) habilitado.

Nota-se que as curvas favorecem bastante o caso em que a retificação é feita antes da compressão (processo RET2). Isso pode ser justificado pelo fato de as operações de predição no padrão H.264/AVC serem feitas em blocos retangulares, que são prejudicadas pele fato de uma imagem possuir distorção espacial acentuada, como é o caso na projeção *fisheye*. Pode-se observar também que as imagens retificadas possuem mais áreas homogêneas ao afastar-se do centro, devido à distorção perspectiva introduzida pela correção. Tais áreas com menos detalhes apresentam maior redundância espacial e são mais facilmente comprimidas. Por fim, o processo de correção radial com mapeamento inverso baseia-se em um método de interpolação para gerar a imagem de saída. Caso a imagem de entrada já possua artefatos de compressão, estes são amplificados pela correção, degradando o sinal.

As curvas para o processo RET2 são sobrepostas na Figura 5.10, onde pode-se notar que há um compromisso entre melhora na taxa de compressão e piora na distorção ao utilizar-se a predição *inter*, além de os resultados com RDO apresentarem melhor performance geral.

Foi analisada uma segunda sequência obtida no mesmo local a fim de testar a repetibilidade do experimento. Ela será identificada como Sequência 2 e é mostrada na Figura 5.11. A retificação foi feita com os mesmos parâmetros do caso anterior. A comparação se limitou a habilitar e desabilitar a RDO, mantendo a predição *inter* para os demais *frames* que não o primeiro. O parâmetro de quantização de *slices* I assumiu os valores 22, 27, 32 e 37, sendo que para *slices* P os valores eram os mesmos somando-se 1. As curvas comparativas entre os processos RET1 e RET2 são apresentadas na Figura 5.12. Na Figura 5.13 os resultados são sobrepostos aos da Sequência 1 para os mesmos parâmetros de quantização e modo de predição.

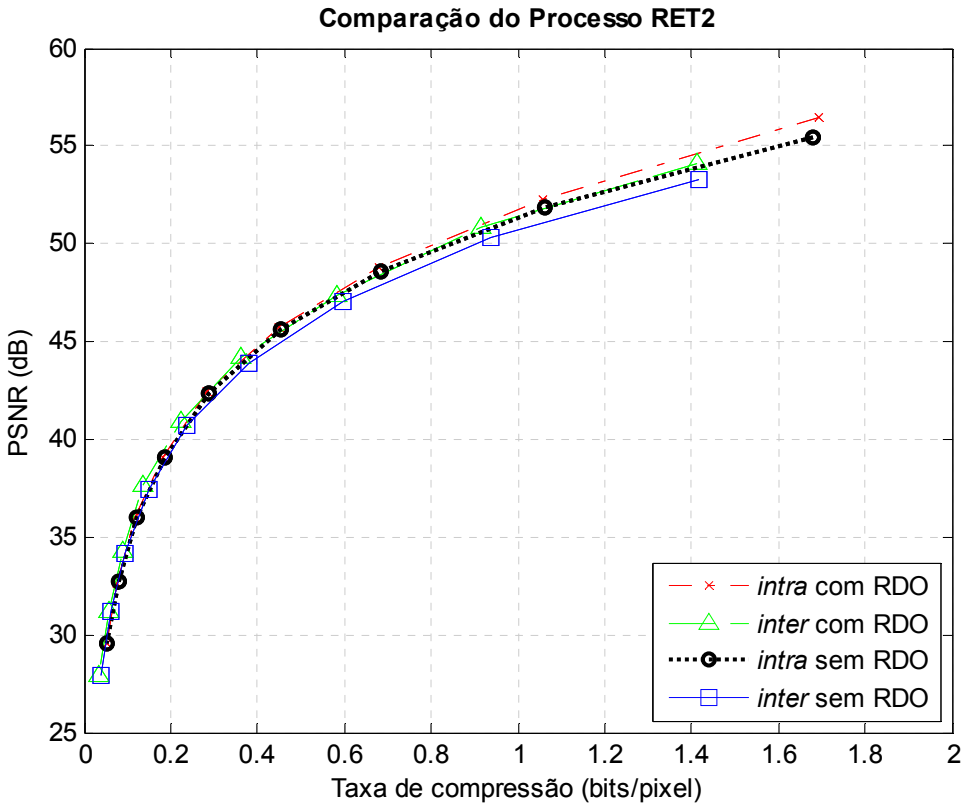

Figura 5.10 – Curvas do processo RET2 para Sequência 1 sobrepostas.

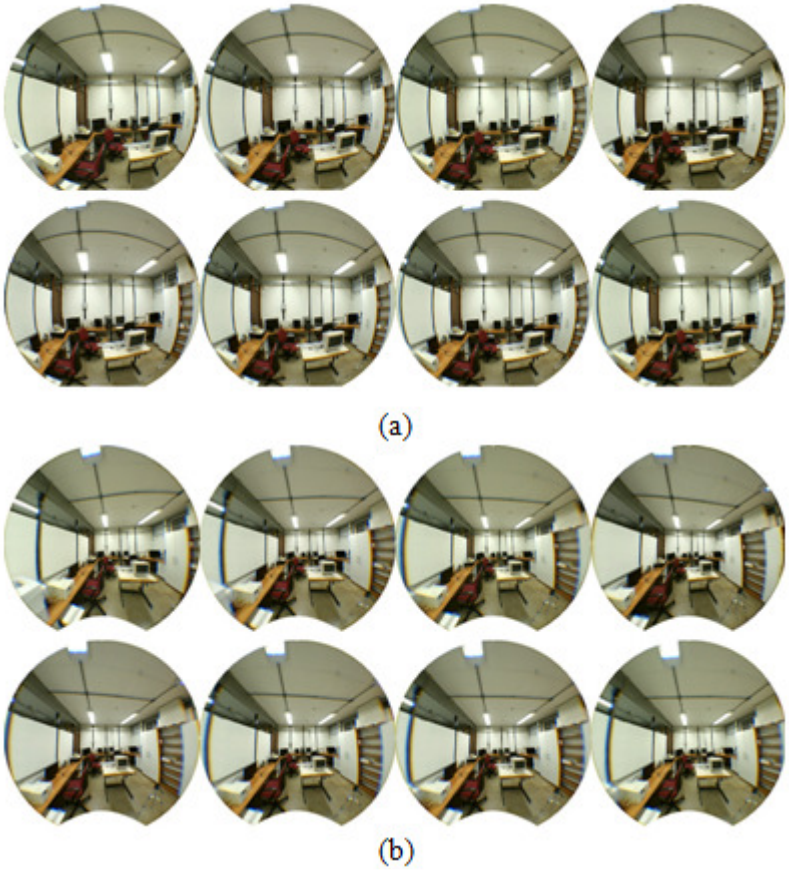

Figura 5.11 – Sequência 2 utilizada na repetição da comparação: a) *fisheye* e b) retilínea.

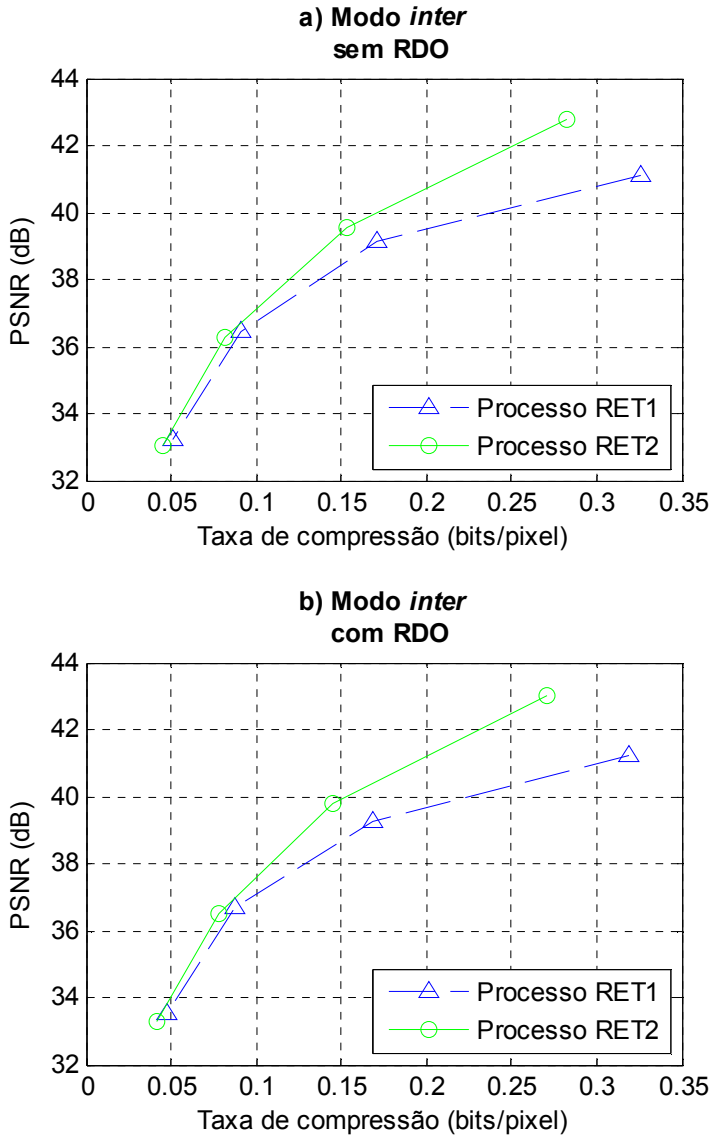

Figura 5.12 – Curvas de PSNR para Sequência 2: com predição *inter* e RDO a) desabilitado e b) habilitado.

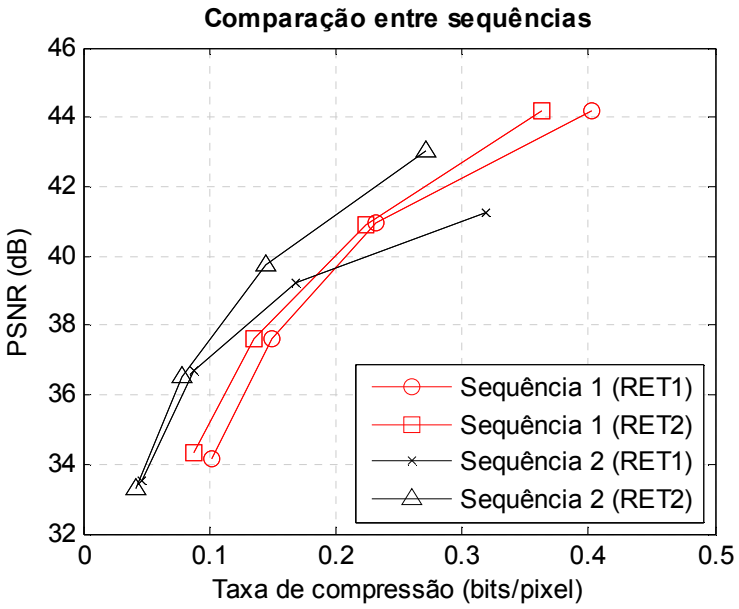

Figura 5.13 – Comparação entre curvas de PSNR geradas para Sequências 1 e 2.

Nota-se que as curvas apresentam a mesma forma que as da primeira sequência, indicando que o experimento foi reproduzido como esperado.

Os processos analisados nesta seção poderiam ter aplicações práticas caso se necessitasse, por exemplo, transmitir as imagens por algum meio a fim de se visualizar a projeção retilínea ao final. Neste caso, seria mais adequado, de acordo com os resultados obtidos nesta seção, que a retificação do vídeo fosse feita em um passo anterior à compressão e transmissão do sinal. Caso a correção radial fosse aplicada na imagem já comprimida e transmitida, a perda em qualidade visual seria maior.

5.3 COMPRESSÃO DE IMAGENS *FISHEYE* E PANORAMAS

Nesta etapa são comparados os processos PANO1 e PANO2 da Figura 4.6. No processo PANO1, são codificadas e decodificadas as imagens separadas para em seguida formar-se um panorama. No processo PANO2, é formado um panorama com as imagens originais, que é em seguida codificado e decodificado. Foram utilizadas duas sequências de imagens capturadas com equipamento profissional. A primeira foi obtida na mesma fonte descrita na seção 5.1, e é mostrada na Figura 5.14(a). A segunda, mostrada na Figura 5.14(b), é de autoria de Karl Harrison [33]. Esta sequência foi capturada com uma câmera DSLR Canon 5D e lente Canon de 15 mm, e estava comprimida com JPEG em qualidade 98. As imagens de ambas as sequências possuem FOV de 180º na diagonal, proporcionando a formação de um panorama que cobre 360º na horizontal e cerca de 150º na vertical. O panorama equiretangular formado a partir dessas imagens possui faixas na parte inferior e superior, representando a região não coberta na vertical. Os panoramas obtidos a partir das sequências de maneira automática com o software Hugin são mostrados na Figura 5.15. As imagens da Sequência 1 possuíam dimensões 672 x 1008 px, e as da Sequência 2, 608 x 912 px. Os dois panoramas gerados possuíam dimensões 3008 x 1504 px.

Figura 5.14 – Sequências utilizadas na codificação de panoramas: a) Sequência 1 e b) Sequência 2.

(a)

(b)

Figura 5.15 – Panoramas obtidos a partir da a) sequência 1 e b) sequência 2 (as bordas foram inseridas para visualização do tamanho total).

Os processos foram comparados com a sequência de imagens sendo codificada de duas maneiras: com predição *intra* em todos os *frames* e com predição *intra* só no primeiro *frame* (predição *inter* nas demais). Foi mantida a otimização de taxa-distorção (RDO) habilitada, e o parâmetro de quantização de *slices* I foi variado entre os valores 7, 12, 17, 22, 27, 32, 37, 42 e 47. Os mesmos valores somados a 1 foram usados no parâmetro de quantização de *slices* P. As curvas de PSNR obtidas na comparação dos processos para a Sequência 1 são mostradas na Figura 5.16. Para a Sequência 2, as curvas são mostradas na Figura 5.17. Na Figura 5.18 é apresentada uma comparação entre a performance das sequências em cada processo.

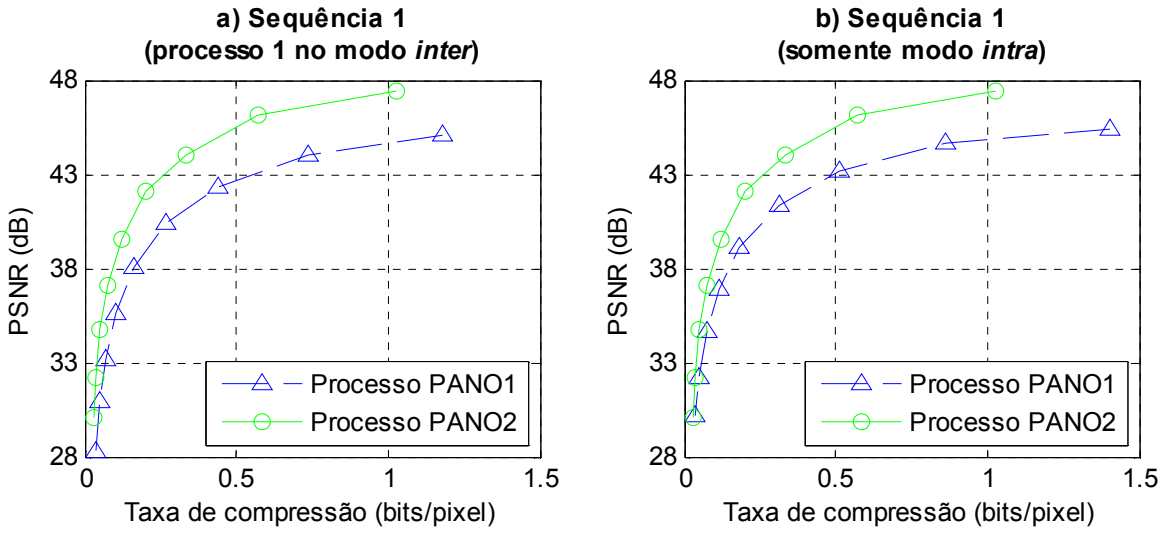

Figura 5.16 – Curvas de PSNR para Sequência 1: a) com predição *inter* e b) somente com predição *intra*.

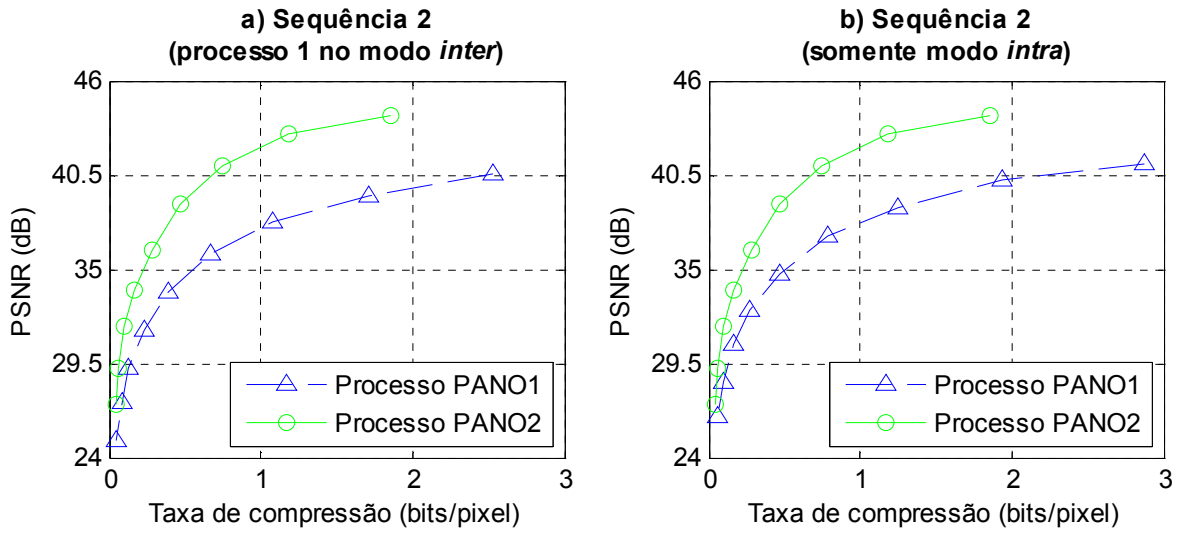

Figura 5.17 – Curvas de PSNR para Sequência 2: a) com predição *inter* e b) somente com predição *intra*.

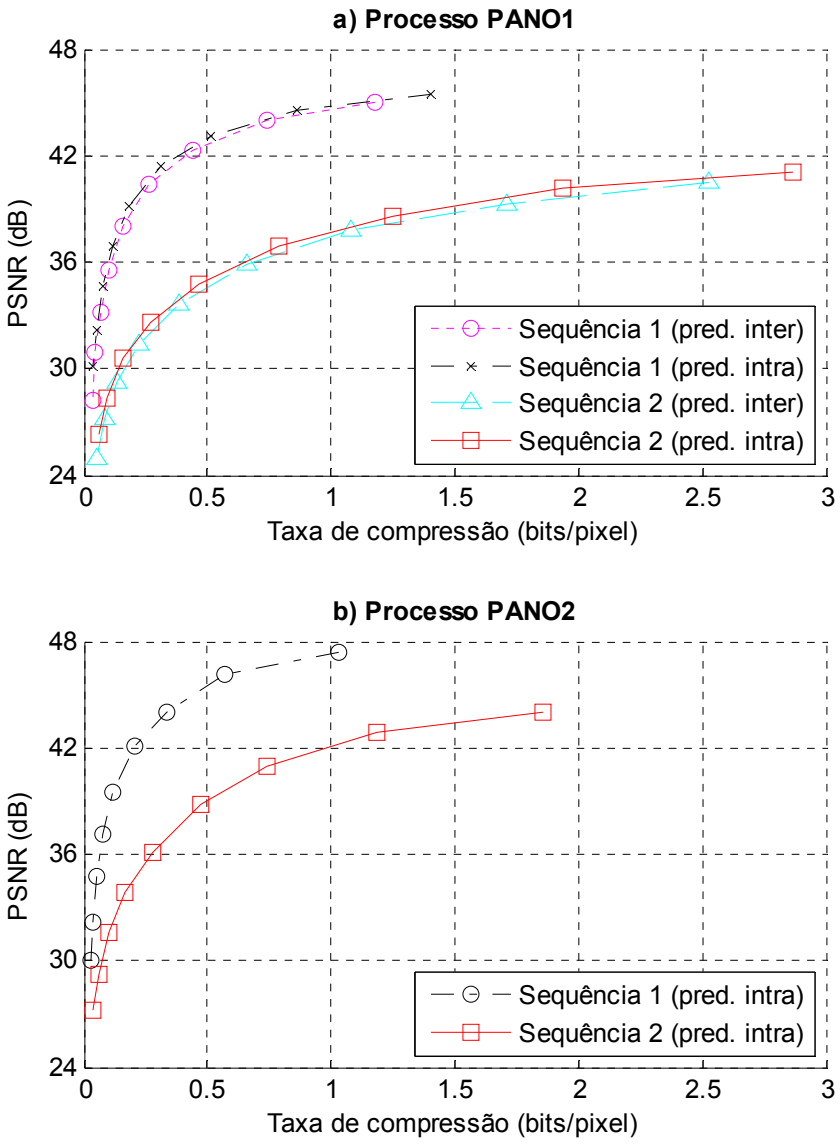

Figura 5.18 – Comparação de curvas de PSNR para sequências no processo a) PANO1 e b) PANO2.

É observado a partir das curvas que o processo PANO2 é bastante mais eficiente do ponto de vista das taxas de distorção e de compressão. É necessário ressaltar, como já descrito em 4.3, que as taxas de compressão do processo PANO1 levam em conta um arquivo texto de 4000 bytes, comprimido no formato ZIP, com as informações para formação do panorama.

Apesar deste fato, a maior causa da baixa performance do processo PANO1 nas métricas utilizadas é a aparição de artefatos em áreas de junções entre imagens devido a, durante o processo de formação do panorama, ocorrer uma otimização ruim na etapa de sobreposição das imagens. Isso ocorre pois, mesmo com os pontos de correspondência entre as imagens fixados no arquivo texto, o software deve adequar o mapeamento das imagens na projeção equiretangular de maneira que uma métrica de erro na sobreposição das imagens seja minimizada. Este processo iterativo depende portanto do conteúdo das imagens, e não serão obtidos os mesmos resultados para todas as sequências comprimidas. A Figura 5.19 ilustra esse aparecimento de imperfeições na sequência 2 em uma área onde duas imagens foram sobrepostas. A Figura 5.19(a) é o panorama de referência, e as Figura 5.19(b) a (d) são geradas a partir de imagens sucessivamente mais comprimidas. Nota-se que a perda de qualidade em si das imagens é dificilmente notada, mas a presença dos erros de sobreposição prejudicam bastante a medida de distorção utilizada (PSNR).

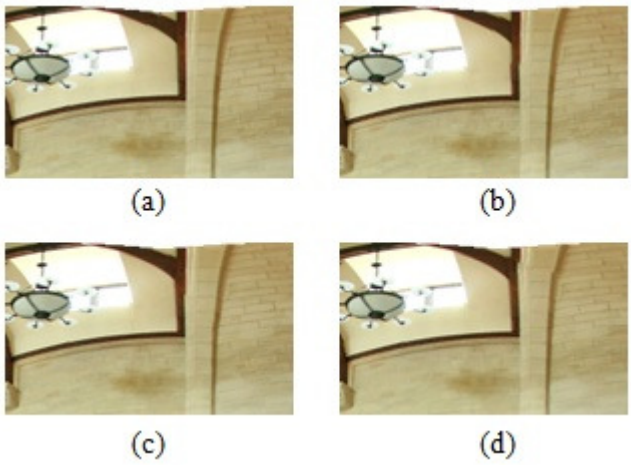

Figura 5.19 – Erro na sobreposição de imagens no panorama formado pelo processo PANO1 na sequência 2: a) referência, b) QPI = 7, c) QPI = 12 e d) QPI = 17.

Analisando-se a taxa de compressão nas curvas obtidas, nota-se que o processo PANO2 sempre apresenta vantagem. A utilização de dimensões das imagens múltiplas de 16 e de imagens com tamanho menor na sequência 2 foram tentativas de melhorar a peformance em compressão do processo PANO1, mas ambas não surtiram efeito. A utilização da predição *inter* ao invés da *intra* apresentou uma melhora na razão taxa-distorção, mas insuficiente para alcançar a performance do processo PANO2. As pequenas falhas observadas na Figura 5.19 poderiam ser relevadas caso a performance em compressão do processo PANO1 fosse mais vantajosa, o que não ocorre. A melhor alternativa é, portanto, o processo PANO2, a não ser que seja possível repetir exatamente o estágio de sobreposição de imagens na formação automática do panorama no processo PANO1.

6 CONCLUSÃO

Neste trabalho foi estudada a representação de imagens em projeção *fisheye* e sua relação com as técnicas de compressão do padrão de codificação de vídeo H.264/AVC. Foram desenvolvidos algoritmos de calibração e correção de imagens obtidas nesta projeção que apresentaram bons resultados para lentes de qualidade profissional e resultados de compromisso para lentes semi-profissionais. Uma série de testes de codificação de sequências de imagens em projeção *fisheye* e retilínea foi feita, gerando resultados favoráveis à projeção retilínea, esperados dentro das condições impostas. Não se pôde concluir diretamente sobre a implicação do modelo de projeção nas técnicas de compressão, mas o sistema como um todo pôde ser analisado e recomendações podem ser feitas para aplicações práticas baseado nos resultados aqui obtidos. Entre os sistemas de aquisição de imagens que podem se beneficiar ao mesmo tempo do campo de visão ampliado das lentes *fisheye* e da familiaridade da projeção retilínea, citam-se o monitoramento de câmeras de segurança e o auxílio à navegação de veículos de grande porte. Em ambos os casos, pode-se requerer a transmissão das imagens por algum meio, sendo necessária a compressão do sinal para obter-se maior eficiência. Estes cenários são equivalentes aos procedimentos avaliados neste trabalho, e pode-se recomendar que a correção da imagem seja feita anteriormente à sua compressão e posterior transmissão. Caso a correção da imagem seja realizada ao final da cadeia de processamento, antes da visualização, haverá maiores perdas em eficiência de compressão e qualidade visual.

Foram estudadas também a formação de panoramas imersivos. Gerados a partir de um conjunto de imagens que cobrem toda a cena, também se beneficiam de imagens obtidas com lentes *fisheye* pelo seu maior campo de visão. Tendo em vista uma unidade móvel que obtém imagens panorâmicas sequencialmente, foi analisada a vantagem em se formar o panorama antes ou depois de se comprimirem as imagens. No caso em que o panorama é formado a partir das imagens já comprimidas, foi notado o fato de se necessitar de uma base de correspondências fixa entre as imagens. Com isso, poupa-se esforço computacional e garante-se que panoramas formados em instantes distintos tenham uma composição suficientemente parecida. Isso, contudo, não foi o bastante para que se evitassem erros na sobreposição das imagens, gerando imperfeições na imagem panorâmica. A alternativa melhor avaliada, portanto, consistiu em formar-se o panorama logo após a sua aquisição das imagens, para então comprimir a imagem resultante e transmiti-la ou armazená-la.

Como perspectiva de trabalhos futuros tem-se o aperfeiçoamento da geração de panoramas automaticamente a partir de um número reduzido de imagens utilizando-se lentes *fisheye*. Este ainda é um tópico aberto para pesquisas pois a distorção radial da projeção *fisheye* dificulta o estabelecimento de correspondências entre as imagens e, ainda, as lentes com campo de visão hemisférico possuem um custo elevado, sendo necessárias soluções criativas. Pode-se também analisar mais a fundo o comportamento das técnicas de compressão, aplicadas geralmente em blocos retangulares, em imagens que não possuam projeção retilínea, a fim de proporem-se métodos alternativos para esses casos.

7 REFERÊNCIAS BIBLIOGRÁFICAS

[1] Sidney Ray, *Applied Photographic Optics*, 3rd ed. Oxford, UK: Focal Press, 2002.

[2] ITU-T Rec. H.264 and ISO/IEC 14496-10 (MPEG-4 Part 10), "Advanced Video Coding for Generic Audiovisual Services," Edição 5.0, Março 2010.

[3] Iain Richardson, *H.264 and MPEG-4 Video Compression*. West Sussex, UK: John Wiley & Sons, 2003.

[4] Hughes C., Glavin M., Jones E., and Denny P., "Wide-angle camera technology for automotive applications: a review," *IET Intelligent Transport Systems*, vol. 3, no. 1, pp. 19-31, Março 2009.

[5] Arnaud Frich, *Panoramic photography*. Oxford, UK: Focal Press, 2007.

[6] Robert Hirsch, *Light and Lens: Photography in the Digital Age*. Oxford, UK: Focal Press, 2008.

[7] Richard Szeliski, *Computer Vision: Algorithms and Applications*. New York, US: Springer, 2010, [Online]. Disponível: http://szeliski.org/Book. [Acessado: Agosto, 2010].

[8] Ravi Ramamoorthi, Computational Imaging and Photography, 2009, [Online]. Disponível: UC Berkeley EECS Instructional and Electronics Support, http://www-inst.eecs.berkeley.edu/~cs294-13/fa09/. [Acessado: Agosto, 2010].

[9] Takeshi Koyama, "Optics in Digital Still Cameras," in *Image Sensors and Signal Processing for Digital Still Cameras*, Junichi Nakamura, Ed. Florida, Estados Unidos: CRC Press, 2006, ch. 2.

[10] James Kumler and Martin Bauer, "Fisheye Lens Designs and their Relative Performance," in *SPIE Proceedings - Current Developments in Lens Design and Optical Systems Engineering*, 2000, pp. 360-369.

[11] Altera Corporation. (2008) A Flexible Architecture for Fisheye Correction in Automotive Rear-View Cameras. [Online]. http://www.altera.com/literature/wp/wp-01073-flexible-architecture-fisheye-correction-automotive-rear-view-cameras.pdf

[12] Margaret Fleck, "Perspective Projection: The Wrong Imaging Model," Dep. of Computer Science, University of Iowa, Techical 1995.

[13] C. Hughes, M. Glavin, E. Jones, and P. Denny, "Review of Geometric Distortion Compensation in Fish-Eye Cameras," in *16th IET Irish Signals and Systems Conference, ISSC*, Galway, 2008.

[14] C. Hughes, M. Glavin, E. Jones, and P. Denny, "Validation of Polynomial-based Equidistance Fish-Eye Models," in *20th IET Irish Signals and Systems Conference, ISSC*, Dublin, 2009.

[15] A. Basu and S. Licardie, "Alternative models for fish-eye lenses," *Elsevier Pattern Recognition Letters*, vol. 16, no. 4, pp. 433–441, 1995.

[16] A. Fitzgibbon, "Simultaneous linear estimation of multiple view geometry and lens distortion," in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2001, pp. 125–132.

[17] F. Devernay and O. Faugeras, "Straight lines have to be straight: automatic calibration and removal of distortion from scenes of structured enviroments," *Springer-Verlag Journal of Machine Vision and Applications*, vol. 13, no. 1, pp. 14–24, 2001.

[18] Janez Pers and Stanislav Kovacic, "Nonparametric, Model-Based Radial Lens Distortion Correction Using Tilted Camera Assumption," in *Computer Vision Winter Workshop 2002*, 2002, pp. 286-295.

[19] Z. Zhang, "Flexible camera calibration by viewing a plane from unknown orientations," in *IEEE Int. Conference on Computer Vision*, 1999, pp. 666–673.

[20] Y. Xiong and K. Turkowski, "Creating image-based VR using a self-calibrating fisheye lens," in *In IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'97)*, San Juan, 1997, pp. 237–243.

[21] R Hartley and S Kang, "Parameter-free radial distortion correction with center of distortion estimation," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2007, pp. 1309–1321.

[22] A. Wang, T. Qiu, and L. Shao, "A Simple Method of Radial Distortion Correction with Centre of Distortion Estimation," *Journal of Mathematical Imaging and Vision*, vol. 35, no. 3, pp. 165-172, Novembro 2009.

[23] M. Deering, "The Limits of Human Vision," in *2nd International Immersive Projection Technology Workshop*, 1998.

[24] Sevket Gumustekin. (1999, Julho) Image Processing Place. [Online]. http://www.imageprocessingplace.com/downloads_V3/root_downloads/tutorials/An_Introduction_to_Image_Mosaicing.htm

[25] John P Snyder, *Map Projections: A Working Manual*. Washington, DC, Estados Unidos: United States Government Printing Office, 1987.

[26] John Houghton. (2009, Dezembro) Finding the No-Parallax Point. [Online]. http://www.johnhpanos.com/epcalib.htm

[27] G Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[28] Pablo d'Angelo. (2003) hugin. [Online]. http://hugin.sourceforge.net

[29] Andrew Mihal. (2004) Enblend. [Online]. http://enblend.sourceforge.net

[30] Alan Bovik, "Introduction to Digital Image and Video Processing," in *Handbook of Image and Video Processing*, Alan Bovik, Ed. Londres, UK: Academic Press, 2000, ch. 1.

[31] Joint Model. (2009) JM. [Online]. http://iphome.hhi.de/suehring/tml

[32] 360 Degree of Freedom. 360dof. [Online]. http://www.360dof.com

[33] Karl Harrison. Panorama Tutorials. [Online]. http://www.chem.ox.ac.uk/oxfordtour/tutorial/