Avaliação de funções-custo na super-resolução de imagens para jogos eletrônicos

Felippe Durán V. G. Santos, Renato Candido e Magno T. M. Silva

***Resumo*— A reconstrução de imagens em super-resolução tem sido de interesse em diversas áreas, como imagens médicas, vigilância, jogos eletrônicos, entre outras. Neste trabalho, são avaliadas diferentes funções-custo no contexto de super-resolução de imagens para jogos eletrônicos. Para isso, utiliza-se uma versão simplificada do modelo de [1], que faz uso de múltiplos quadros, seus respectivos mapas de profundidade e vetores de movimento. Além das funções-custo tradicionais como o erro quadrático médio, o erro absoluto médio e a obtida do índice de similaridade estrutural, avaliam-se a *perceptual loss* e a *G-loss*. Resultados experimentais mostram que a escolha de diferentes camadas da *perceptual loss* tem influência positiva no desempenho e que a combinação dessas funções-custo com ela pode resultar em detalhes que melhoram a percepção qualitativa das imagens.**

***Palavras-Chave*— Super-resolução de imagens, aprendizado profundo, redes neurais convolucionais, jogos eletrônicos, funções-custo.**

***Abstract*— Super-resolution image reconstruction has been of interest in several areas, such as medical images, surveillance, electronic games, among others. In this work, different cost functions are evaluated in the context of image super-resolution for electronic games. For this, a simplified version of the model of [1] is used, which uses multiple frames, their respective depth maps, and motion vectors. In addition to traditional cost functions such as the mean squared error, the mean absolute error, and the derived from the structural similarity index, we evaluate the the perceptual loss and the G-loss. Experimental results show that the choice of different perceptual loss layers influences the performance for the better, and the combination of those cost functions with the perceptual loss can result in details that improve the qualitative perception of the images.**

***Keywords*— Image super-resolution, deep learning, convolutional neural networks, video games, loss functions.**

I. Introdução

A reconstrução de imagens em super-resolução a partir de uma única imagem de baixa resolução encontra aplicações em imagens médicas, vigilância, jogos eletrônicos etc. [2], [3]. Nessa área, técnicas tradicionais como interpolação não uniforme, reconstrução analítica no domínio de frequência e regularização [2], [4] caíram em desuso. Em contrapartida, diferentes modelos de aprendizado de máquina têm sido utilizados para se obter resultados expressivos [5], [6].

A escolha da função-custo tem sido um dos elementos cruciais para se obter bons resultados em super-resolução de imagens com aprendizado de máquina. Dentre as funções-custo propostas, algumas se destacam pela facilidade interpretação, enquanto outras por capturarem aspectos mais sutis da imagem. As funções-custo mais comuns são o erro quadrático médio (*mean squared error* – MSE) [7]–[10], o erro absoluto médio (*mean absolute error* – MAE) [11], [12] e a obtida do índice de similaridade estrutural (*structural similarity index* – SSIM) [1], [11], [13]. Outra função é a *perceptual loss* [14], que alguns dos trabalhos mencionados também usa de forma ponderada às outras para compor a função-custo final. Mais recentemente, foi também proposta a *G-loss* [15].

Em jogos eletrônicos, a super-resolução tem ganhado cada vez mais destaque [3], [17], [18]. A pressão da indústria por inserir técnicas sofisticadas de renderização 3D [19] e o aumento na densidade de pixels dos dipositivos têm levado os hardwares ao limite no processo de renderização em tempo real. Diante disso, técnicas de super-resolução têm sido utilizadas para se obter imagens de alta qualidade visual. A possibilidade de se ter a renderização e a rasterização numa resolução menor possibilita economia de memória e custo computacional [20].

Na indústria, a NVIDIA e a Intel têm utilizado técnicas de superamostragem, que fazem uso de quadros passados e vetores de movimento para gerar as imagens de alta resolução [3], [21]. Elas também inserem um ruído de posição da tela para enriquecer a variedade de informação entre quadros. Em geral, essas técnicas são proprietárias, e portanto, a pesquisa aberta dentro do campo de super-resolução de imagens em jogos eletrônicos ainda é limitada.

Na literatura, Xiao *et al.* [1] propuseram um modelo de super-resolução que faz uso de múltiplos quadros, seus respectivos mapas de profundidade e vetores de movimento para se obter melhorias visuais em relação a outros trabalhos. Já Wang *et al.* [22] aplicaram a super-resolução na compressão e *streaming* de jogos. Entretanto, um ponto que ainda não foi explorado mais a fundo é a avaliação do desempenho de diferentes funções-custo nessa aplicação.

Neste artigo, será feita essa exploração com uma versão simplificada do modelo de Xiao *et al.* [1]. Para isso, apresentam-se na Seção a formulação do problema, na Seção III a metodologia utilizada, o banco de dados, detalhes do treinamento e resultados. O artigo termina com conclusões na Seção IV.

II. Formulação do problema

Nesta seção, são descritos os elementos-chave de renderização 3D, o modelo utilizado e as funcões-custo analisadas.

A. Mapas de profundidade e movimento

O processo de renderização 3D, ao longo do tempo, tornou-se cada vez mais complexo para incorporar diferentes propriedades físicas de luz, sombra e de câmera [20]. Para produzir tais efeitos, canais adicionais de imagem são gerados em

Felippe Durán V. G. Santos, Renato Candido e Magno T. M. Silva Departamento de Eng. de Sistemas Eletrônicos, Universidade de São Paulo, São Paulo-SP, e-mails: felippe.duran@usp.br, renatocan@lps.usp.br, magno.silva@usp.br. Este trabalho foi financiado pela CAPES (código de financiamento 001), pelo CNPq (303826/2022-3 e 404081/2023-1) e pela FAPESP (2021/02063-6).

etapas de pré-processamento, e depois produz-se a imagem final na tela com a composição deles [20].

Um desses canais extras, é o mapa de profundidade. Originalmente usado para se evitar o redesenho de pixels sobrepostos no processo de rasterização [20], ele adquiriu diversos outros usos ao longo do tempo: cálculo da posição de objetos em relação à câmera, iluminação e oclusão ambiente [20].

Outro subproduto é o mapa de movimento, que é utilizado para se calcular o deslocamento de objetos relativo à câmera a partir da diferença de posição dos pontos de cada objeto entre dois quadros consecutivos. É interessante notar que dentro do domínio de vídeo, este mapa é frequentemente estimado [10], mas no contexto de jogos, ele pode ser precisamente calculado. Entretanto, ambos possuem o problema de oclusão de objetos, em que se perde a continuidade de movimento [20].

B. O modelo

Para focar no estudo das funções-custo, decidiu-se fixar o modelo. Assim, escolheu-se a rede neural U-Net, que foi proposta originalmente para o problema de segmentação de imagens médicas em [23] e adaptada posteriormente para super-resolução [1], [24]–[26]. Baseada em *autoencoders*, essa rede conta com quatro blocos de codificação, um central, quatro de decodificação e um final. Nela, busca-se uma representação do espaço latente da imagem [27]. A U-Net também faz uso de conexões residuais entre as camadas de codificação e decodificação que permitem a preservação das informações posicionais da imagem para facilitar a reconstrução [23], [28]. Na Figura 1a, é mostrado um esquema dessa rede, no qual se indicam as camadas de convolução e a operação de *max-pooling*. Como função de ativação, utiliza-se a ReLU. O bloco final possui duas camadas de convolução: uma igual às anteriores e outra com ativação linear e filtro 1×1 para se obter a imagem na resolução desejada. Todas as outras operações de convolução possuem filtros 3×3.

Em vídeos e jogos eletrônicos, o mapa de movimento é utilizado para se obter a reprojeção de um quadro passado para o instante atual, o que permite alinhar os pixels dos dois quadros, facilitando a associação e o enriquecimento de informações [1], [10]. Seguindo essa ideia, considera-se neste trabalho uma versão simplificada do modelo de [1] que remove as etapas de extração de características e ponderação e faz o uso direto dos quadros anteriores reprojetados no processo de reconstrução. O diagrama da arquitetura considerada pode ser visto na Figura 1b. Para cada quadro, o mapa de movimento é utilizado no processo de reprojeção e apenas a imagem RGB e o mapa de profundidade são concatenados como entrada da U-Net. Dessa forma, para N_q quadros, utiliza-se uma entrada única de dimensões $H \times W \times N_q(C+1)$, em que H e W são as dimensões da imagem, e C o número de canais de cores.

C. Funções-custo

a) Erro quadrático médio: Considerada uma das funções-custo mais simples e intuitivas, o MSE é definido por

$$\text{MSE} = \mathcal{L}_2(\mathbf{Y}, \hat{\mathbf{Y}}) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2, \tag{1}$$

sendo $\mathbf{Y}$ a imagem desejada e $\hat{\mathbf{Y}}$ sua estimativa e y_i, $\hat{y}_i$, $i = 1, \cdots, N$ os valores dos pixels de $\mathbf{Y}$ e $\hat{\mathbf{Y}}$, respectivamente.

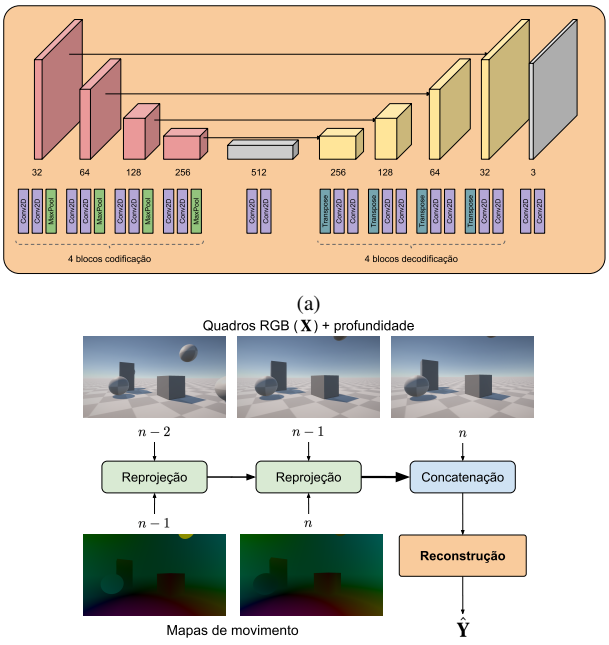

Fig. 1. **(a)** Detalhe da arquitetura U-Net utilizada. **(b)** Arquitetura do modelo completo, incluindo a rede de reconstrução U-Net da Figura 1a e o processo de reprojeção de quadros anteriores.

Apesar de ser computacionalmente barata, essa função não é capaz de capturar aspectos mais sutis da imagem [29].

b) Erro absoluto médio: O MAE é definido por

$$\text{MAE} = \mathcal{L}_1(\mathbf{Y}, \hat{\mathbf{Y}}) = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|. \tag{2}$$

O MAE tem sido utilizado como alternativa ao MSE e tem mostrado melhores resultados na preservação de detalhes e métricas como a razão sinal-ruído de pico (*peak signal-to-noise ratio* – PSNR) [11]. Entretanto, por ser similar ao MSE, essa função ainda apresenta várias das mesmas limitações [29].

c) Índice de similaridade estrutural: O SSIM entre duas imagens $\mathbf{X}$ e $\mathbf{Y}$ é definido por [13]:

$$\text{SSIM}(\mathbf{X}, \mathbf{Y}) = \frac{(2\mu_\mathbf{X}\mu_\mathbf{Y} + C_1)(2\sigma_\mathbf{XY} + C_2)}{(\mu_\mathbf{X}^2 + \mu_\mathbf{Y}^2 + C_1)(\sigma_\mathbf{X}^2 + \sigma_\mathbf{Y}^2 + C_2)} \tag{3}$$

Esse índice busca capturar a similaridade levando em consideração a luminância, o contraste e a estrutura geral a partir das médias $\mu_\mathbf{X}$ e $\mu_\mathbf{Y}$, desvios-padrão $\sigma_\mathbf{X}$, $\sigma_\mathbf{Y}$ e covariância $\sigma_\mathbf{XY}$. C_1 e C_2 são constantes derivadas do domínio de valores dos pixels e assumem os valores de 10^{-5} e 9×10^{-5}, respectivamente, para imagens com cada canal no domínio $[0, 1]$. O valor do SSIM pertence ao intervalo $[-1, 1]$, sendo que 1 indica similaridade perfeita e quanto menor o valor no intervalo $[0, 1]$ mais distintas são as imagens. [13] não faz menção específica sobre a interpretação de valores negativos.

Para se obter uma avaliação mais granular da similaridade, o SSIM pode ser calculado com janelas gaussianas 11×11 ao longo das imagens. Com isso, define-se o SSIM médio entre as imagens $\mathbf{Y}$ e $\hat{\mathbf{Y}}$ como

$$\text{MSSIM}(\mathbf{Y}, \hat{\mathbf{Y}}) = \frac{1}{M}\sum_{i=1}^{M}\text{SSIM}(\mathbf{Y}_i, \hat{\mathbf{Y}}_i), \tag{4}$$

com cada janela $\mathbf{Y}_i$ e $\hat{\mathbf{Y}}_i$ e M a quantidade total de janelas. Utilizando o MSSIM, define-se a seguinte função-custo

$$\mathcal{L}_{\text{SSIM}}(\mathbf{Y}, \hat{\mathbf{Y}}) = 1 - \text{MSSIM}(\mathbf{Y}, \hat{\mathbf{Y}}), \tag{5}$$

cujos valores pertencem ao intervalo $[0, 2]$, com 0 indicando similaridade perfeita.

Apesar de algumas referências alegarem ser uma medida mais próxima da percepção humana [29], essa função apresenta diversas inconsistências na sua avaliação, especialmente quando aplicadas a imagens coloridas [30]. Diante disso, [11] propôs o uso conjunto do SSIM com o MAE para se obter melhores resultados.

d) Perceptual loss: Essa função busca capturar a similaridade entre imagens a partir da extração de características. Para isso, ela usa uma seção convolucional do modelo VGG-16 [31], pré-treinado em uma tarefa de classificação de imagens. A Figura 2 ilustra parte de sua arquitetura, com cinco blocos coloridos de camadas convolucionais. Entre cada bloco de cores, existe uma operação de *max-pooling* [31].

A *perceptual loss* utiliza o tensor de saída do segundo bloco do modelo, chamado de $\phi_{\texttt{relu2\_2}}(\mathbf{X})$ [14]. Como [14] escolheu a camada `relu2_2` empiricamente, outras saídas podem ser utilizadas dependendo da aplicação. Considerados como mapas de características da imagem, cada canal desses tensores de saída mapeia uma característica diferente [14]. A partir daí, calcula-se o erro quadrático médio entre os mapas das imagens desejada e estimada, obtendo-se

$$\mathcal{L}_{\text{Perceptual}}(\mathbf{Y}, \hat{\mathbf{Y}}) = \mathcal{L}_2(\phi(\mathbf{Y}), \phi(\hat{\mathbf{Y}})). \tag{6}$$

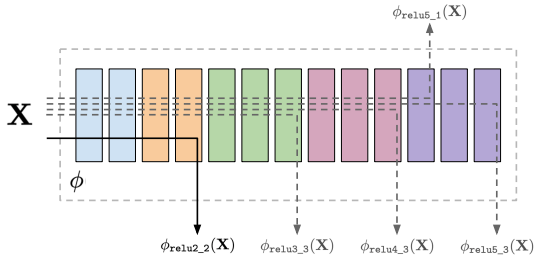

Fig. 2. Parte convolucional do modelo VGG-16 [31] e respectivas saídas para o cálculo da *perceptual loss*. A saída $\phi_{\texttt{relu2\_2}}$ é a recomendada por [14] para a aplicação de super-resolução. Cada bloco é uma camada convolucional. A saída de cada grupo de cores inclui uma operação de *max-pooling*.

e) G-loss: Essa função é formada de três termos: o MAE, o gradiente em 8 direções (cima, baixo, esquerda, direita e diagonais) e a utilização de imagens em diferentes resoluções.

O cálculo do gradiente, denotado por $\text{G}(\mathbf{X})$, é feito utilizando-se 8 filtros 3×3 na forma de

$$\mathbf{G} = \left[\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & -1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \cdots \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \right],$$

aplicados à cada canal da imagem. A saída é então a concatenação dos mapas de gradiente, resultando em um tensor de dimensões $H \times W \times 8C$, com H representando a altura da imagem, W sua largura e C o número de canais de cores.

Para considerar imagens em diferentes resoluções, utiliza-se a subamostragem definida por uma operação de *pixel-shuffle* inversa [32], chamada de *splitting* [15]. Essa operação está ilustrada na Figura 3 para uma imagem $4 \times 4 \times 1$ e fator de escala 2. Dessa forma, o tensor da imagem, de dimensões $H \times W \times C$, é decomposto em um de dimensões $H/n \times W/n \times C \times n^2$, sendo n a escala de sub-amostragem.

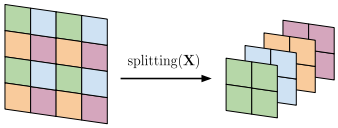

Fig. 3. Diagrama da operação de *splitting* para $n = 2$, em os que pixels da imagem são agrupados em 4 canais diferentes com metade da resolução original.

Unindo os dois últimos termos, define-se a função

$$\text{SG}_n(\mathbf{X}) = \text{G}(\text{splitting}_n(\mathbf{X})), \tag{7}$$

e com isso, a função-custo intermediária é dada por

$$\mathcal{L}_{\text{SG}_n}(\mathbf{Y}, \hat{\mathbf{Y}}) = \mathcal{L}_1(\text{SG}_n(\mathbf{Y}), \text{SG}_n(\hat{\mathbf{Y}})). \tag{8}$$

Finalmente, para se compor a *G-loss*, [15] faz uma soma ponderada de $\mathcal{L}_{\text{SG}_n}$ para diferentes escalas de subamostragem, resultando em

$$\mathcal{L}_{\text{G-loss}}(\mathbf{Y}, \hat{\mathbf{Y}}) = \mathcal{L}_1(\mathbf{Y}, \hat{\mathbf{Y}}) + \sum_{n=1}^{4} \frac{\mathcal{L}_{\text{SG}_n}(\mathbf{Y}, \hat{\mathbf{Y}})}{n^2}, \tag{9}$$

em que o fator $1/n^2$ foi definido empiricamente por [15].

III. Resultados Experimentais

Nos experimentos, foi utilizado um banco de dados gerado a partir de um único cenário de vilarejo medieval [33]. No total, 40 tomadas de câmera foram feitas em diferentes ângulos, 30 utilizadas no treinamento e 10 na validação. Cada tomada possui diferentes movimentos de câmera e duração de 1 segundo, totalizando 15 quadros. Para melhorar a qualidade das imagens em resolução 1920×1080, utilizou-se $4\times$ *multi-sampling antialiasing* (MSAA) [20]. Os quadros de mapa de movimento e profundidade utilizam 8 bits para cada canal, de modo a compor 24 bits. As imagens geradas foram então quebradas em 28 peças de 256×256 e cada tomada foi quebrada em 10 janelas aleatórias de 3 quadros consecutivos. No total, foram gerados 11200 exemplos. Para se gerar a versão de baixa resolução dos exemplos, foi executado uma subamostragem de $4\times$ sem qualquer técnica de *anti-aliasing*, seguido de uma sobreamostragem com o vizinho mais próximo para deixar a imagem já com o tamanho desejado. A ausência de *anti-aliasing* torna a imagem de entrada mais próxima do cenário real, que também possui artefatos de *aliasing* [20].

Utilizaram-se na implementação a biblioteca Tensorflow 2.15 [34], o otimizador Adam [35], taxa de aprendizado de 5×10^{-4} e mini-lotes de 8 exemplos. Os treinamentos foram executados em uma placa gráfica NVIDIA GeForce RTX 3090 Ti, e chegou-se ao número de 40 épocas experimentalmente, de forma a garantir a convergência de cada treinamento sem gerar *overfitting*. As funções-custo ponderadas foram normalizadas de maneira a tornar seus valores mais comparáveis. O método consistiu em dividir a função-custo $\mathcal{L}(n)$ do mini-lote n por $\mathcal{L}_{\text{max}}(n)$, em que $\mathcal{L}_{\text{max}}(n) = 0.9\mathcal{L}_{\text{max}}(n-1) + 0.1\mathcal{L}(n)$. Dessa forma, define-se um denominador mais robusto contra oscilações.

Cada experimento foi feito com diferentes combinações de funções-custo com a *perceptual loss*, dado que essa costuma ser a combinação mais comum [1], [11], [38]. e os resultados foram avaliados qualitativamente e quantitativamente com base nos valores de *perceptual loss*, PSNR e SSIM. Os resultados numéricos para o banco de dados de validação estão agrupados

na Tabela I, sendo que os melhores valores das métricas consideradas estão indicados em negrito, e os piores, sublinhados. Em termos da métrica *perceptual loss*, a função-custo que apresentou o menor valor foi a própria *perceptual loss*. Em contrapartida, ela apresentou as piores métricas PSNR e SSIM, fato que também foi notado em trabalhos anteriores (e.g., [36]). Em termos de PSNR, o MSE apresentou o melhor resultado. Já para o SSIM, as quatro funções-custo puras (SSIM, MSE, MAE e *G-loss*) tiveram resultados equivalentes considerando-se 3 casas decimais. As combinações das funções-custo levaram a resultados proporcionais aos pesos aplicados, com o desempenho em termos de PSNR e SSIM navegando entre os limites de cada função-custo separadas. A exceção foi para a combinação de $0.75 * \text{SSIM} + 0.25 *$ *perceptual*, onde ambos *perceptual loss* e PSNR melhoraram, indicando uma leve sinergia. Vale notar que, com exceção do teste realizado apenas com a *perceptual loss* da primeira linha da tabela, todos os outros resultados tiveram valores de PSNR e SSIM muito próximos, o que indica que a *perceptual loss* leva em conta fatores consideravelmente diferentes da PSNR ou SSIM para a avaliação das imagens.

TABELA I

RESULTADOS EXPERIMENTAIS PARA COMBINAÇÕES DE FUNÇÃO-CUSTO.

Loss Function	Perceptual Loss	PSNR (dB)	SSIM
Perceptual	**23671**	25.718	0.854
SSIM	33011	30.084	**0.946**
$0.75 * \text{SSIM} + 0.25 *$ Perceptual	28009	30.313	0.944
$0.50 * \text{SSIM} + 0.50 *$ Perceptual	25166	29.953	0.941
$0.25 * \text{SSIM} + 0.75 *$ Perceptual	24074	29.783	0.936
MSE	30764	**31.013**	**0.946**
$0.75 * \text{MSE} + 0.25 *$ Perceptual	27845	30.952	0.945
$0.50 * \text{MSE} + 0.50 *$ Perceptual	25416	30.821	0.943
$0.25 * \text{MSE} + 0.75 *$ Perceptual	24361	30.399	0.938
MAE	30514	30.761	**0.946**
$0.75 * \text{MAE} + 0.25 *$ Perceptual	25922	30.780	0.944
$0.50 * \text{MAE} + 0.50 *$ Perceptual	24366	30.593	0.941
$0.25 * \text{MAE} + 0.75 *$ Perceptual	24136	29.962	0.936
G-loss	30498	30.987	**0.946**
$0.75 *$ *G-loss* $+ 0.25 *$ Perceptual	25445	30.492	0.940
$0.50 *$ *G-loss* $+ 0.50 *$ Perceptual	24333	30.336	0.936
$0.25 *$ *G-loss* $+ 0.75 *$ Perceptual	24066	29.884	0.927

Diante disso, partiu-se para uma comparação qualitativa. Os resultados das funções individuais, entrada e saída estão na Figura 4. Como esperado, a função-custo baseada no SSIM puro foi a que mais errou nas tonalidades, produzindo uma acentuação de contraste [30]. Além disso, é possível notar que a *perceptual loss* busca incluir padrões de textura nos materiais, ao contrário do que ocorre com as outras funções custo. Esse comportamento, em certa medida, pode proporcionar uma percepção visual mais agradável, conforme mostrado na Figura 5. Nela, são mostrados resultados da *G-loss* combinada em diferentes proporções com a *perceptual loss*. À medida em que a proporção da *perceptual loss* é aumentada, as texturas que ela incorpora se tornam mais visíveis, fazendo com que alguns detalhes se tornem mais nítidos e proporcionem uma percepção visual mais semelhante à da imagem original. No entanto, as diferenças de percepção entre as imagens com diferentes proporções de combinação são muito sutis, com exceção do caso da imagem obtida apenas

Fig. 4. Resultados para diferentes funções-custo. Da esquerda para a direita: imagem desejada, detalhe desejado, entrada, MSE, MAE, SSIM e *perceptual*.

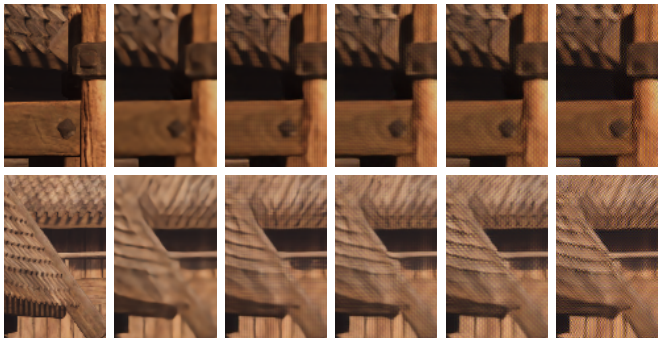

Fig. 5. Resultados para diferentes ponderações entre *G-loss* e *perceptual loss*. Da esquerda para a direita: desejada, *G-loss*, 75% *G-loss*, 50% cada, 75% e 100% *perceptual*.

com a *perceptual loss*, o que coincide com os resultados numéricos de PSNR e SSIM mostrados na Tabela I. Vale notar também que resultados equivalentes foram obtidos para as combinações com outras funções. Dessa maneira, considera-se que o resultado quantitativo reflete de alguma forma os resultados visuais, em que não se teve alterações significativas de qualidade visual.

Considerar diferentes camadas da *perceptual loss* foi o experimento que trouxe maior variação nos resultados. Os resultados numéricos estão na Tabela II e as imagens reconstruídas na Figura 6. Nota-se uma troca entre geração de detalhes de textura e nitidez de bordas. Quanto mais profunda a camada, mais detalhes em detrimento da nitidez. Outro ponto a se observar é a presença de padrões de alta frequência, que tendem a ser diluídos para as camadas mais profundas. De forma geral, a opinião é de que a camada `relu5_1` produziu o melhor balanço de características, especialmente quando se observa a imagem como um todo. Do ponto de vista numérico, as camadas `relu2_2` e `relu5_3` foram as que obtiveram os piores resultados de SSIM e PSNR. Mais resultados estão disponíveis no repositório `http://github.com/felippeduran/superresolution-sbrt2024`.

TABELA II

RESULTADOS PARA *perceptual loss* COM DIFERENTES CAMADAS.

Layer Name	Perceptual Loss	PSNR (dB)	SSIM
`relu2_2`	**23686**	25.260	0.827
`relu3_3`	26478	**26.900**	**0.893**
`relu4_3`	29732	25.975	0.884
`relu5_1`	30808	26.659	0.885
`relu5_3`	33821	25.277	0.857

Fig. 6. Resultados para diferentes camadas da *perceptual loss*. Da esquerda para a direita: desejada e detalhe, entrada, `relu2_2`, `relu3_3`, `relu5_1`, `relu5_3`.

IV. Conclusões e Trabalho Futuro

Diante dos resultados obtidos, pode-se chegar em três conclusões principais. A primeira é que a adição da *perceptual* às outras funções ajuda no sentido de trazer, ainda que de forma sutil, algumas texturas que tornam a percepção visual mais agradável. No entanto, 100% *perceptual* leva a um resultado estranho aos olhos, que se reflete nos valores de PSNR e SSIM. A segunda conclusão é que, contrariamente à [14], a camada `relu2_2` não foi a mais adequada para a aplicação de super-resolução, e a `relu5_1` apresentou melhores resultados, quantitativamente e qualitativamente. Finalmente, notou-se uma importância na definição do critério de avaliação. Porque, apesar da *perceptual loss* apresentar artefatos periódicos na textura gerada em alguns casos, isso é o que justamente implica na melhora da qualidade visual da imagem como um todo, e levou a resultados consistentemente melhores. Como trabalho futuro, vale a exploração da combinação das funcões-custo com as outras camadas da *perceptual loss* que mostram resultados melhores e a expansão do banco de dados para uma melhor generalização dos resultados. Além disso, motivado por [12], [37], cabe também buscar outros modelos pré-treinados para se obter uma versão alternativa e mais atualizada da *perceptual loss*.

Referências

[1] L. Xiao *et al.*, "Neural supersampling for real-time rendering," *ACM Trans. Graph.*, vol. 39, no. 4, 2020.

[2] X. Li *et al.*, "Superresolution image reconstruction: Selective milestones and open problems," *IEEE Sig. Process. Mag.*, vol. 40, pp. 54–66, 2023.

[3] NVIDIA, "NVIDIA DLSS 2.0: A big leap in AI rendering," 2020. [Online]. Available: https://www.nvidia.com/en-us/geforce/news/nvidia-dlss-2-0-a-big-leap-in-ai-rendering/

[4] S. C. Park, M. K. Park, and M. G. Kang, "Super-resolution image reconstruction: a technical overview," *IEEE Signal Process. Mag.*, vol. 20, no. 3, pp. 21–36, 2003.

[5] C. Saharia *et al.*, "Image super-resolution via iterative refinement," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, pp. 4713–4726, 2023.

[6] J. Liang *et al.*, "VRT: A video restoration transformer," *IEEE Trans. Image Process.*, vol. 33, pp. 2171–2182, 2024.

[7] C. Dong *et al.*, "Learning a deep convolutional network for image super-resolution," in *Computer Vision – ECCV 2014*. Springer International Publishing, 2014, pp. 184–199.

[8] T. Tong *et al.*, "Image super-resolution using dense skip connections," in *IEEE Int. Conf. on Computer Vision (ICCV)*, 2017, pp. 4809–4817.

[9] Y. Tai *et al.*, "Memnet: A persistent memory network for image restoration," in *IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 4549–4557.

[10] J. Caballero *et al.*, "Real-time video super-resolution with spatio-temporal networks and motion compensation," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2848–2857.

[11] H. Zhao *et al.*, "Loss functions for image restoration with neural networks," *IEEE Transactions on Computational Imaging*, vol. 3, no. 1, pp. 47–57, 2017.

[12] M. M. Thomas *et al.*, "Temporally stable real-time joint neural denoising and supersampling," *Proc. ACM Comput. Graph. Interact. Tech.*, vol. 5, no. 3, 2022.

[13] Z. Wang *et al.*, "Image quality assessment: from error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, 2004.

[14] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *Computer Vision – ECCV 2016*. Springer International Publishing, 2016, pp. 694–711.

[15] L. Ge and L. Dou, "G-loss: A loss function with gradient information for super-resolution," *Optik*, vol. 280, p. 170750, 2023.

[16] T. An *et al.*, "Patch loss: A generic multi-scale perceptual loss for single image super-resolution," *Pattern Recognition*, vol. 139, p. 109510, 2023.

[17] A. Watson, "Deep learning techniques for super-resolution in video games," 2020. Available https://arxiv.org/abs/2012.09810

[18] AMD, "AMD FidelityFX super resolution 2.0," 2022. [Online]. Available: https://gpuopen.com/gdc-presentations/2022/GDC_FidelityFX_Super_Resolution_2_0.pdf

[19] J. Fisher, "GeForce RTX propels PC gaming golden age with real-time ray tracing," 2018. [Online]. Available: https://blogs.nvidia.com/blog/geforce-rtx-real-time-ray-tracing/

[20] T. Akenine-Möller, E. Haines, and N. Hoffman, *Real-Time Rendering*, 4th ed. CRC Press, 2018.

[21] "Intel Xe super sampling (XeSS)." [Online]. Available: https://www.intel.com/content/www/us/en/products/docs/discrete-gpus/arc/technology/xess.html

[22] Y. Wang *et al.*, "Efficient super-resolution for compression of gaming videos," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2023, pp. 1–5.

[23] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*. Springer International Publishing, 2015, pp. 234–241.

[24] P. Liu *et al.*, "Multi-level wavelet-cnn for image restoration," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2018, pp. 886–88 609.

[25] K. H. Jin *et al.*, "Deep convolutional neural network for inverse problems in imaging," *IEEE Trans. Image Process.*, vol. 26, pp. 4509–4522, 2017.

[26] X.-J. Mao, C. Shen, e Y.-B. Yang, "Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections," in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, ser. NIPS'16. Red Hook, NY, USA: Curran Associates Inc., 2016, p. 2810–2818.

[27] M. A. Kramer, "Nonlinear principal component analysis using autoassociative neural networks," *AIChE Journal*, vol. 37, no. 2, pp. 233–243, 1991.

[28] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[29] Z. Wang and A. C. Bovik, "Mean squared error: Love it or leave it? a new look at signal fidelity measures," *IEEE Signal Process. Mag.*, vol. 26, no. 1, pp. 98–117, 2009.

[30] J. Nilsson and T. Akenine-Möller, "Understanding SSIM," 2020. Available: https://arxiv.org/abs/2006.13846v2

[31] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015. Available: https://arxiv.org/abs/1409.1556v6

[32] W. Shi *et al.*, "Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[33] Unity Technologies, "Viking Village." [Online]. Available: https://assetstore.unity.com/packages/essentials/tutorial-projects/viking-village-urp-29140

[34] M. Abadi *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. [Online]. Available: https://www.tensorflow.org/

[35] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd Int. Conference for Learning Representations ICLR*, 2015.

[36] Y. Blau *et al.*, "The 2018 PIRM challenge on perceptual image super-resolution," in *Computer Vision – ECCV 2018 Workshops: Munich, Germany, September 8-14, 2018, Proceedings, Part V*. Berlin, Heidelberg: Springer-Verlag, 2019, p. 334–355.

[37] R. Zhang *et al.*, "The unreasonable effectiveness of deep features as a perceptual metric," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Los Alamitos, CA, USA: IEEE Computer Society, jun 2018, pp. 586–595.

[38] X. Wang, *et al.*, "ESRGAN: Enhanced super-resolution generative adversarial networks," in *Computer Vision – ECCV 2018 Workshops*. Springer International Publishing, 2019, pp. 63–79.