GPU-Based Edge-Directed Image Interpolation

Martin Kraus[1], Mike Eissele[2], and Magnus Strengert[2]

[1] Computer Graphics and Visualization Group,
Informatik 15, Technische Universität München,
Boltzmannstraße 3, 85748 Garching, Germany,
e-mail: krausma@in.tum.de

[2] Visualization and Interactive Systems Group,
Institut VIS, Universität Stuttgart,
Universitätsstraße 38, 70569 Stuttgart, Germany,
e-mail: mike.eissele@informatik.uni-stuttgart.de
e-mail: magnus.strengert@informatik.uni-stuttgart.de

Abstract. The rendering of lower resolution image data on higher resolution displays has become a very common task, in particular because of the increasing popularity of webcams, camera phones, and low-bandwidth video streaming. Thus, there is a strong demand for real-time, high-quality image magnification. In this work, we suggest to exploit the high performance of programmable graphics processing units (GPUs) for an adaptive image magnification method. To this end, we propose a GPU-friendly algorithm for image up-sampling by edge-directed image interpolation, which avoids ringing artifacts, excessive blurring, and staircasing of oblique edges. At the same time it features gray-scale invariance, is applicable to color images, and allows for real-time processing of full-screen images on today's GPUs.

1 Introduction

Digital image magnification is by no means a trivial technical task—in fact, many real-life scenarios include perceptional issues which are not covered by the theory of signal processing. In particular, human subjects often perceive theoretically optimal magnifications of images as less sharp and more blurred than images magnified with algorithms that heuristically add high frequencies to the sampled signal. This is due to the fact that humans often have a more precise model of the physical signal than the sampled image data can provide. For example, we often expect regions of uniform colors with sharp edges in images although the finite sampling of a digital image cannot provide this information. Thus, improving image magnification algorithms for subjectively sharper results is—in a technical sense—an ill-posed problem. Nonetheless, the challenge exists and became more important in recent years due to the increasing popularity of higher resolution display devices such as PC screens, video beamers, and HDTVs, while video DVDs and TV signals still provide lower resolutions. There are also new popular image sources, e.g., webcams, camera phones, and internet video streams, which often provide even lower resolutions. Thus, it is common to up-sample images before rendering them.

In many of the conceivable scenarios, programmable graphics processing units (GPUs) are employed to render the image data. Therefore, we propose to use these GPUs for advanced image magnification techniques—in particular because off-line preprocessing of image data is often not possible due to the lack of communication bandwidth, available memory, or the requirement to avoid latencies; for example, in interactive applications. The image magnification algorithm presented in this work is particularly well suited for implementations on GPUs.

Additional requirements and related work are discussed in Section 2. In Section 3, a one-dimensional edge model is presented and a method for magnifying this ideal edge without blurring nor ringing artifacts is derived. The adaptation of this method for GPU-based image magnification is discussed in Section 4 while Section 5 presents experiments and results.

2 Requirements and Related Work

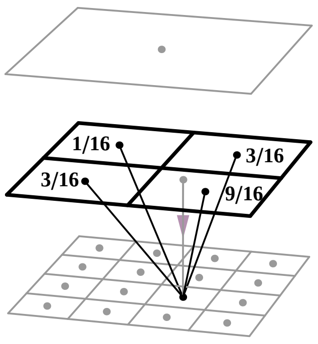
Several previously published concepts and ideas are crucial in the design of our method. In this section, we discuss the most important design requirements and related publications.

2.1 Pyramidal Magnification

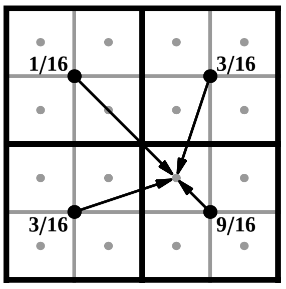
Many image magnification methods are restricted to a magnification factor of 2. Factors equal to powers of 2 are implemented by the corresponding number of magnification operations while arbitrary factors are implemented by up-sampling to the smallest power of 2 that is greater than the requested factor and a minification step that employs, for example, bilinear interpolation. This pyramidal approach provides the optimal (linear) time complexity while considerably simplifying the up-sampling algorithm and its implementation—in particular if the method is implemented in hardware. Figure 1a illustrates an image pyramid, which consists of the coarse image (1×1 pixel) at the top and two finer image levels (2×2 pixels and 4×4 pixels), which are synthesized from the original image by expanding the image data by a factor of two in each magnification step.

There are two different schemes for the positioning of the new samples, which are called primal and dual scheme (or face-split and vertex-split scheme) in the literature on subdivision surfaces. The primal scheme inserts new samples between old samples while keeping the old samples. This is the more traditional interpolation scheme in image magnification methods since it guarantees an interpolation of the original colors by preserving them, and also avoids most computations for all the old samples, i.e., one quater of the pixels of the magnified image.

The dual scheme places all new samples symmetrically between old samples and discards the old samples as illustrated in Fig. 1b. This approach has been



(a)



(b)

Fig. 1. (a) Image pyramid consisting of three levels. (b) Dual subdivision scheme.

employed by Zhao and de Haan [1] and Strengert et al. [2] as it is more appropriate for a single-pass implementation and/or a GPU-based implementation. Moreover, the uniform interpolation of samples according to the dual scheme with the weights depicted in Fig. 1 results in a C^1-continuous biquadratic B-spline filtering in the limit of infinitely many up-sampling steps [2].

2.2 Edge Preservation

In practice, image data is usually undersampled. Therefore, reconstructing images with the ideal sinc filter would result in ringing artifacts unless an additional low-pass filter is applied. This low-pass filtering, however, is often perceived as a blurring of the image. As mentioned in the introduction, human subjects estimate the optimal sharpness of edges in images not only based on the displayed data but also based on their knowledge about the depicted objects. Thus, human subjects often correctly assume that the physical signal featured sharper edges than the sampled image data. Therefore, model-based image up-sampling methods have been proposed that attempt to solve the inverse problem of determining the physical signal, which led to the given image data under the assumption of a certain observation model [3].

A somewhat more modest goal is realized by adaptive interpolation methods, which detect strong edges in images and adapt the interpolation weights accordingly to avoid an interpolation across these edges. Therefore, this approach is also called edge-directed image interpolation. In terms of a model-based approach, this corresponds to identifying those edges which have not been sampled at a sufficiently high frequency and, therefore, appear blurred in a technically correct reconstruction of the finite resolution image data. Figures 2c and 2f on page 6 illustrate the sharpening effect of edge-directed interpolation in a one-dimensional example. Since the physical signal of the edge is assumed to feature an infinitely sharp edge as depicted in Fig. 2a, the slope of the up-sampled edge signal in Fig. 2f is increased (in comparison to the original edge signal in Fig. 2c) to approximate the sampling of this edge at a higher resolution. It should be emphasized that excessive sharpening of edges has to be avoided since the lack of any blurring due to a sampling process is very noticeable. Moreover, excessive sharpening of two-dimensional images results in staircasing artifacts of oblique edges similar to aliasing artifacts.

Techniques for edge detection in adaptive edge-directed interpolation methods are usually based on pixel correlation [1, 4], local pixel classification [5–7], or local gradients [8, 9]. In this work, we adapt the boundary model proposed by Kindlmann and Durkin [10], which is related to the edge detectors by Canny [11] and by Marr and Hildreth [12].

2.3 Gray-Scale Invariance

Adaptive interpolation methods are nonlinear mappings of images because the interpolation weights depend on the image data. In general, this can result in undesirable dependencies on the overall intensity or on the local illumination. These disadvantages can be avoided if the adaptive interpolation method is required to be homogeneous; i.e., the up-sampled image should show a multiplicative scaling behaviour for scaled input data. In the context of edge-directed interpolation, this requires an edge detection method that works independently of the absolute scale of edges. Of course, this assumes a signal-to-noise ratio, which is also scale-invariant. As quantized image data already violates this assumption, gray-scale invariance cannot be achieved perfectly. Nonetheless, it is an important design requirement for adaptive interpolation methods as noted, for example, by Pietikäinen [7].

2.4 Color Interpolation

Applicability to color images is an obvious requirement for general image magnification methods. Applying the adaptive interpolation separately to all color components will usually result in unpleasant color shifts. Therefore, edge-directed interpolation methods (including our approach) are usually designed to detect edges in luminance images and adapt the weights for an interpolation of color vectors.

2.5 GPU-Based Interpolation

Some of the requirements for high-performance GPU-based interpolation methods can be identified in the design of the linear, non-adaptive image zooming method proposed by Strengert et al. [2]. Specifically, this method also applies pyramidal magnification with the dual sampling scheme. Even more important is the consistent use of bilinear image interpolation supported by OpenGL graphics hardware. In the context of adaptive interpolation, this amounts to offsetting the sampling coordinates of a (dependent) bilinear image interpolation such that the weights of the GPU-based bilinear interpolation are adapted automatically by the GPU. This indirect adaptation of interpolation weights is also applicable to color images if the offsets to the sampling coordinates are computed from luminance data while the dependent bilinear image lookup interpolates color image data.

3 Ideal Edge Characterization

Analogously to the work by Kindlmann and Durkin [10], we first choose a one-dimensional edge model and develop an exact magnification method for this continuous model. The method does not suffer from ringing artifacts but preserves the ideal edge without any blurring. Moreover, it is gray-scale invariant and can be applied to color images.

For an ideal edge the observed physical signal is assumed to feature an arbitrarily sharp, discontinuous change from a value $y_{\min}$ to $y_{\max}$ at position x_0 as depicted in Fig. 2a. Therefore, this physical signal is modeled by a parameterized step function:

$$y_{\min} + (y_{\max} - y_{\min})\,\Theta(x - x_0). \tag{1}$$

Due to the measurement, however, the sampled signal is blurred. For the sampled edge signal $f(x)$, this observation process is modeled by a convolution with a normal distribution with standard deviation σ (Fig. 2b):

$$f(x) = (y_{\min} + (y_{\max} - y_{\min})\,\Theta(x - x_0)) \otimes \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{x^2}{2\sigma^2}\right). \tag{2}$$

The resulting function $f(x)$ is a parameterized error function plotted in Fig. 2c and serves as our model of the sampled signal for an ideal edge:

$$f(x) = \frac{y_{\min} + y_{\max}}{2} + \frac{y_{\max} - y_{\min}}{2} \operatorname{erf}\left(\frac{x - x_0}{\sqrt{2}\sigma}\right). \tag{3}$$

The position x_0 of the ideal edge is characterized by the maximum of $f'(x)$ and the zero crossing of $f''(x)$ as illustrated in Fig. 2d. These criteria are exploited by the edge detectors by Canny [11] and by Marr and Hildreth [12], respectively. However, $f'(x)$ and $f''(x)$ are less appropriate for the characterization of the transition region of an ideal edge as their scale depends on $y_{\min}$ and $y_{\max}$, which are not known a priori. Therefore, we form a scale-invariant expression from $f'(x)$ and $f''(x)$:

$$d(x) \stackrel{\text{def}}{=} \frac{-\sigma^2 f''(x)}{f'(x)}. \tag{4}$$

For our model of the ideal edge, $d(x)$ is equal to $x - x_0$ as plotted in Fig. 2e; thus, a zero crossing of $d(x)$ corresponds to the position x_0 of an (ideal) edge. Moreover, the definition of $d(x)$ is independent of $y_{\min}$, $y_{\max}$, and x_0; thus, it

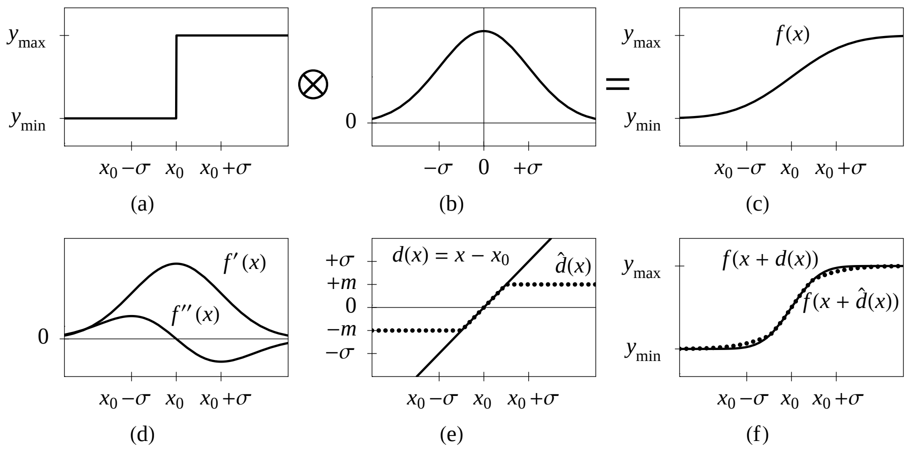


Fig. 2. Detection and sharpening of an ideal edge: (a) parameterized step function, (b) normal distribution filter modeling the measurement process, (c) sampled ideal edge signal, (d) first and second derivative of the sampled signal, (e) offset $d(x)$ for resampling (*solid line*) and clamped offset $\hat{d}(x)$ (*dotted line*), (f) resampled (and sharpened) signal $f(x + d(x))$ (*solid line*) and $f(x + \hat{d}(x))$ (*dotted line*).

can be employed for a scale-invariant characterization of transition regions in an edge magnification method as explained next.

We consider the magnification of the ideal edge signal depicted in Fig. 2c by a factor of 2. At twice the resolution, the edge signal should be twice as sharp, i.e., the convolution in (2) should be computed with a normal distribution with half the standard deviation $\sigma/2$. Since the scaling in x direction is only dependent on σ, it is also possible to compute the magnified edge signal by rescaling the distance $x - x_0$, i.e., the resulting sharper edge signal is given by $f\left(2\left(x - x_0\right) + x_0\right)$ as depicted in Fig. 2f. This expression, however, allows us to employ our scale-invariant edge characterization:

$$f\left(2\left(x - x_0\right) + x_0\right) = f\left(x + \left(x - x_0\right)\right) = f\left(x + d(x)\right). \tag{5}$$

Thus, we can achieve the sharpening of an ideal edge due to a magnification by a factor of 2 simply by resampling the signal with the positional offset $d(x) = -\sigma^2 f''(x)/f'(x)$. It should be emphasized that this resampling of the data with a positional offset maps very well to the GPU-based bilinear image interpolation discussed in Section 2.5.

While this method works for the ideal edge signal, it is obivously not very useful for arbitrary signals since the numerical computation of the offset $d(x)$ is unstable. However, this problem can be easily avoided by clamping the offset between symmetrical bounds $-m$ and $+m$:

$$\hat{d}(x) \stackrel{\text{def}}{=} \max(-m, \min(+m, d(x))). \tag{6}$$

As illustrated by the dotted curves in Figs. 2e and 2f, the clamping has only limited effect in the case of the ideal edge signal if $f(x)$ is already close to $y_{\min}$ for $x < x_0 - m$ and close to $y_{\max}$ for $x > x_0 + m$. On the other hand, clamping the offset stabilizes the resampling process even for $f'(x) \to 0$ since any offset $\hat{d}(x) \in [-m, +m]$ will lead to approximately the same resampling result $f(x + \hat{d}(x)) \approx f(x)$ for $f'(x) \approx 0$.

4 Proposed Method for GPU-Based Up-Sampling

First we present our method for a gray-scale image $f(\mathbf{x})$ with pixel data defined at integer coordinates of $\mathbf{x}$. As mentioned in Section 2 our method employs a dual up-sampling scheme, i.e., the coordinates of the two new pixel positions between the integer coordinates n and $n{+}1$ are $n{+}\frac{1}{4}$ and $n{+}\frac{3}{4}$ as illustrated in Fig. 1b. The resampling of $f(\mathbf{x})$ at these positions employs a bilinear interpolation; therefore, it corresponds to the subdivision scheme of biquadratic B-splines [2]. Analogously to the one-dimensional magnification method discussed in Section 3, we offset the resampling positions $\mathbf{x}$ by a vector $\hat{\mathbf{d}}(\mathbf{x})$ to sharpen undersampled edges by resampling at $f(\mathbf{x} + \hat{\mathbf{d}}(\mathbf{x}))$.

For the two-dimensional generalization of the one-dimensional offset $d(x) = -\sigma^2 f''(x) \;/\; f'(x)$ from Section 3, it is necessary to compute derivatives across edges, i.e., in the direction of the gradient $\nabla f(\mathbf{x})$. Approximating the second derivative across an edge by the Laplacian $\Delta f(\mathbf{x})$ [10] and choosing the normalized gradient $\nabla f(\mathbf{x})/|\nabla f(\mathbf{x})|$ for its direction, the offset $\mathbf{d}(\mathbf{x})$ becomes:

$$\mathbf{d}(\mathbf{x}) \stackrel{\text{def}}{=} \frac{-\sigma^2 \Delta f(\mathbf{x})}{|\nabla f(\mathbf{x})|^2} \nabla f(\mathbf{x}). \tag{7}$$

Additionally, the absolute value of the offset $\mathbf{d}(\mathbf{x})$ should be clamped, say to m:

$$\hat{\mathbf{d}}(\mathbf{x}) \stackrel{\text{def}}{=} \min(m, |\mathbf{d}(\mathbf{x})|) \frac{\mathbf{d}(\mathbf{x})}{|\mathbf{d}(\mathbf{x})|}. \tag{8}$$

Alternatively, each coordinate of $\mathbf{d}(\mathbf{x})$ could be clamped between $-m$ and $+m$ separately. Actual values of m should be approximately 0.25 because of the sampling scheme depicted in Fig. 1b: For larger values of m, the translated sampling positions might no longer be well separated, which leads to a susceptibilty to noise. If m is significantly smaller, the sharpening will become ineffective.

The second free parameter of our method is the standard deviation σ of the Gaussian filter simulating the blurring due to the observation process. This parameter controls the maximum scale of edges that are sharpened; i.e., the smaller σ, the fewer (harder) edges are sharpened. For $\sigma = 0$ no edges are sharpened and our method degenerates to the biquadratic B-spline filtering proposed by Strengert et al. [2]. On the other hand, the larger σ, the more (softer) edges are sharpened. It should be noted that for $\sigma \gtrsim 1$, anti-aliased and other intentionally soft edges might be sharpened. This should be avoided because it is likely to result in aliasing artifacts such as staircasing of oblique edges.

In general, an optimal value of σ does not exist; thus, it is preferable to let users adjust σ between 0 and 1 according to their preferences. Another alternative might be to determine an appropriate σ from statistics about edges detected at various scales: If no soft edges are detected, a large value of σ is less likely to result in artifacts. If, however, no hard edges are detected, even a rather small value of σ can result in artifacts due to too strong sharpening.

The approximations of the terms $\nabla f(\mathbf{x})$ and $\Delta f(\mathbf{x})$ in our proposed computation of $\mathbf{d}(\mathbf{x})$ are based on a low-pass filtered version of the image $f(\mathbf{x})$, which is denoted by $\tilde{f}(\mathbf{x})$. Similarly to the technique presented by Strengert et al. [2], the employed 3×3 Bartlett filter can be implemented by a sequence of two convolutions with 2×2 box filters, each of which can be implemented by a single bilinear image interpolation:

$$\tilde{f} \overset{\text{def}}{=} \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \otimes f = \frac{1}{4}\begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \otimes \frac{1}{4}\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix} \otimes f. \tag{9}$$

This smoothing is necessary for a numerical robust computation of $\nabla f(\mathbf{x})$ by central differences at a new pixel position $\mathbf{x}$:

$$\nabla f(\mathbf{x}) \approx \frac{1}{2}\begin{pmatrix} \tilde{f}\left(\mathbf{x} + (1\ 0)^\top\right) - \tilde{f}\left(\mathbf{x} - (1\ 0)^\top\right) \\ \tilde{f}\left(\mathbf{x} + (0\ 1)^\top\right) - \tilde{f}\left(\mathbf{x} - (0\ 1)^\top\right) \end{pmatrix}. \tag{10}$$

The Laplacian operator for the computation of $\Delta f(\mathbf{x})$ is approximated by a particular filter applied to $\tilde{f}(\mathbf{x})$, which can be evaluated by means of a second convolution with a Bartlett filter:

$$\Delta f(\mathbf{x}) \approx \frac{1}{4}\begin{bmatrix} 1 & 2 & 1 \\ 2 & -12 & 2 \\ 1 & 2 & 1 \end{bmatrix} \otimes \tilde{f}(\mathbf{x}) = 4\left(\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \otimes \tilde{f}(\mathbf{x}) - \tilde{f}(\mathbf{x})\right). \tag{11}$$

In summary, the computation of $\mathbf{d}(\mathbf{x})$ requires 7 (non-dependent) bilinear image interpolations per new, resampled pixel (2 for $\tilde{f}$ per pixel of the original image, i.e., 0.5 per pixel of the magnified image; 4 for $\nabla f(\mathbf{x})$; 0.5 for the Bartlett-filtered $\tilde{f}$; and 2 for $\Delta f(\mathbf{x})$). The clamped offset vector $\hat{\mathbf{d}}(\mathbf{x})$ is then employed for a dependent bilinear image interpolation $f(\mathbf{x} + \hat{\mathbf{d}}(\mathbf{x}))$, which determines the pixel data of the magnified image. It is important that this last dependent image interpolation accesses the unfiltered data f (instead of $\tilde{f}$); thus, the low-pass filtering, which is necessary to compute smooth derivatives, does not lead to any blurring of the magnified image.

As discussed in Section 2.4, the extension of this method to color images is straightforward if luminance edges are detected and sharpened. For each pixel of a color image f_c, the luminance is computed and stored in a gray-scale image f. This image is used to compute $\hat{\mathbf{d}}(\mathbf{x})$ as described above. However, the dependent image interpolation accesses the original color image f_c in order to adaptively up-sample this image. Results of a prototypical implementation of the proposed magnification method are presented in the next section.

5 Experiments and Results

Figure 3 presents some results of our adaptive image up-sampling algorithm. Figure 3c shows a magnification by factor 4 and Figure 3f by factor 8, both for the parameter settings $\sigma = 0.7$ and $m = 0.25$. For comparison, Figs. 3a and 3d depict the original pixels with constant pixel colors, while Figs. 3b and 3e show the magnification with biquadratic B-spline filtering [2], which corresponds to our method for $\sigma = 0$.

We have implemented our method for GPUs that support the OpenGL extensions `GL_ARB_fragment_program` and `GL_EXT_framebuffer_object` using four 16 bit floating-point RGBA buffers. For zoom factors of 2, 4, 8, and 64 with a target image size of one megapixel, our implementation for static color images achieves frame rates of 256, 186, 172, and 165 frames per second (i.e., 3.9, 5.4, 5.8, and 6.1 milliseconds per frame) on an NVIDIA GeForce 6800 GT, which was released in 2004. For the same problem, the more recent NVIDIA GeForce 8800 GTX performed at 1437, 1055, 961, and 876 frames per second (0.70, 0.95, 1.04, and 1.14 milliseconds).

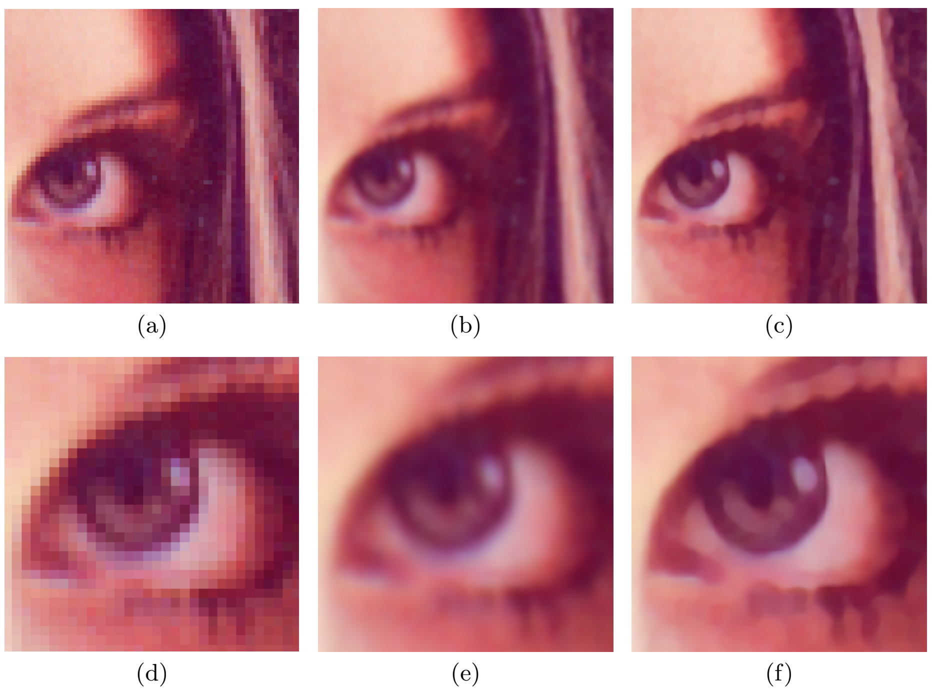


Fig. 3. Comparison of three image magnification methods for the 512×512 Lena image with magnification factor 4 in the top row and factor 8 in the bottom row: (a) and (d) sample-and-hold, (b) and (e) biquadratic B-spline filtering (a special case of our method for $\sigma = 0$), (c) and (f) our method for $\sigma = 0.7$ and $m = 0.25$.

6 Conclusion

We have identified several important requirements for a GPU-based, adaptive image magnification algorithm in order to design and implement an appropriate method on programmable GPUs. In particular, our algorithm features gray-scale invariance, is applicable to color images, and adapts interpolation weights for an edge-directed image interpolation to avoid the blurring of edges. The method is designed to exploit GPU-supported dependent image interpolation for the adaptation of bilinear interpolation weights; therefore, our implementation makes good use of the rasterization performance offered by modern GPUs and provides full-screen image zooming in real time for almost all application scenarios that include a GPU.

Apart from improving the proposed method and its parameters, long-term future work should include research on GPU-based implementations of alternative image magnification algorithms, e.g., adaptive interpolation based on pixel correlation [1, 4] and model-based image magnification methods [3].

References

1. Zhao, M., de Haan, G.: Content adaptive video up-scaling. In: Proceedings ASCI 2003. (2003) 151–156
2. Strengert, M., Kraus, M., Ertl, T.: Pyramid methods in gpu-based image processing. In: Proceedings Vision, Modeling, and Visualization 2006. (2006) 169–176
3. Aly, H.A., Dubois, E.: Image up-sampling using total-variation regularization with a new observation model. IEEE Transactions on Image Processing **14**(10) (2005) 1647–1659
4. Li, X., Orchard, M.T.: New edge-directed interpolation. IEEE Transactions on Image Processing **10**(10) (2001) 1521–1527
5. Atkins, C., Bouman, C., Allebach, J.: Optimal image scaling using pixel classification. In: 2001 International Conference on Image Processing. Volume 3. (2001) 864–867
6. Kondo, T., Node, Y., Fujiwara, T., Okumura, Y.: Picture conversion apparatus, picture conversion method, learning apparatus and learning method. US-patent 6,323,905 (2001)
7. Pietikäinen, M.: Image analysis with local binary patterns. In: Image Analysis, Springer Berlin/Heidelberg (2005) 115–118
8. Hwang, J., Lee, H.: Adaptive image interpolation based on local gradient features. **11**(3) (2004) 359–362
9. Wang, Q., Ward, R.K.: A new edge-directed image expansion scheme. In: ICIP (3). (2001) 899–902
10. Kindlmann, G., Durkin, J.W.: Semi-automatic generation of transfer functions for direct volume rendering. In: Proceedings 1998 IEEE Symposium on Volume Visualization. (1998) 79–86
11. Canny, J.F.: A computational approach to edge detection. (1987) 184–203
12. Marr, D., Hildreth, E.: Theory of edge detection. In: Proceedings of the Royal Society of London. Volume 207. (1980) 187–217