Rapid, Detail-Preserving Image Downscaling

Nicolas Weber[1,2] Michael Waechter[1] Sandra C. Amend[1] Stefan Guthe[1] Michael Goesele[1,2]

[1]TU Darmstadt [2]Graduate School of Computational Engineering at TU Darmstadt

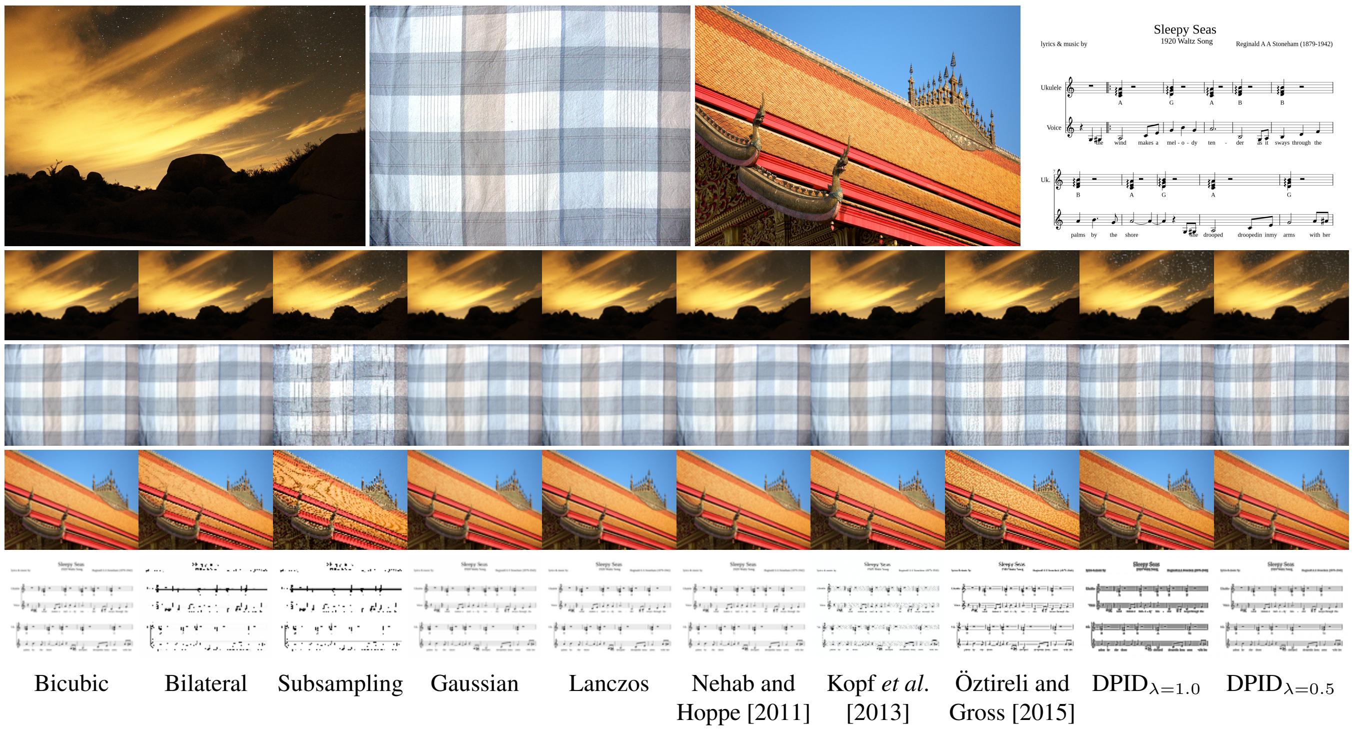


Figure 1: ***Row 1:*** *Input images with 0.5, 1.9, 2.7, and 4.6 megapixels respectively.* ***Rows 2–5:*** *Downscaled results with 128 pixels width. Our algorithm (DPID) preserves stars in Example 1, thin lines in Example 2, roof tiles in Example 3, and text, lines and notes in Example 4.*

Abstract

Image downscaling is arguably the most frequently used image processing tool. We present an algorithm based on convolutional filters where input pixels contribute more to the output image the more their color deviates from their local neighborhood, which preserves visually important details. In a user study we verify that users prefer our results over related work. Our efficient GPU implementation works in real-time when downscaling images from 24 M to 70 k pixels. Further, we demonstrate empirically that our method can be successfully applied to videos.

Keywords: image downscaling, real-time image processing

Concepts: •Computing methodologies → Image processing;

The code for this paper is open source and can be downloaded from www.gcc.tu-darmstadt.de/home/proj/dpid.

This work is licensed under a Creative Commons Attribution-ShareAlike International 4.0 License.

© 2016 Copyright held by the owner/author(s).

SA '16 Technical Papers, December 05-08, 2016, Macao
ISBN: 978-1-4503-4514-9/16/12
DOI: http://dx.doi.org/10.1145/2980179.2980239

ACM Reference Format
Weber, N., Waechter, M., Amend, S., Guthe, S., Goesele, M. 2016. Rapid, Detail-Preserving Image Downscaling. ACM Trans. Graph. 35, 6, Article 205 (November 2016), 6 pages. DOI = 10.1145/2980179.2980239 http://doi.acm.org/10.1145/2980179.2980239.

1 Introduction

Most people constantly carry a device, such as a modern smart phone, that can create high-resolution digital images and videos, display them, and share them over the web. Every single day hundreds of millions of images and hundreds of thousands hours of video are uploaded to photo and video web portals. High-resolution material needs to be downscaled to make it displayable or to reduce data for mobile internet transfer since transfer rates and data plans are limited. Video portals create thumbnails to give users a quick impression of videos (even on screens with limited resolution); they create multiple downscaled versions of a video for any target resolution and connection speed. But even when images or videos are simply viewed locally on a device, downscaling is inevitable due to resolution differences: There is, *e.g.*, a factor of 58 between a Canon EOS 6D's native number of sensor pixels and screen pixels.

Throughout this paper we will focus on downscaling megapixel images to much smaller sizes. As argued above, this is a very important use case. It requires large downscaling factors, which are considerably more challenging than small factors. Both the algorithm design (with respect to runtime) as well as an evaluation with a user study must keep this scenario in mind.

Ideally, a downscaling algorithm retains the original image's impression, preserves fine details, and is memory and time efficient so that it can run on mobile devices or handle the gigantic amounts of data on photo and video web platforms. Traditionally, downscaling algorithms have been signal theory-based. However, these approaches do not take human perception into consideration and instead only concentrate on physical correctness. Kopf *et al.* [2013] and Öztireli and Gross [2015] showed, that such approaches lose visually important high-frequency details during downscaling. Kopf *et al.* proposed a filtering approach that adjusts the filter kernels based on the image content, whereas Öztireli and Gross use the structural similarity index as objective to optimize the downscaled image. Especially Kopf *et al.*'s method is prohibitively expensive for the number of images that web platforms handle.

Our method is based on convolutional filters and determines filter weights such that important details are preserved in the downscaled image. More specifically, it assigns larger weights to pixels that deviate more from their local image neighborhood. From an information theoretic point of view this is the same as saying that a piece of data deviating from its neighborhood carries valuable information, which is of course not always correct—it could also be noise or information beyond the Nyquist frequency. However, according to Beghdadi et al. [2013], the human visual system "approximates the Laplacian edge detector and adaptive low-pass filtering". Thus, a certain level of noise and aliasing may be tolerable while blur leads to loss of important details. Öztireli and Gross's experiments [2015, Figure 12] showed that users rank naïve image subsampling as runner-up behind their algorithm. It seems as if humans may have a hard time telling information and noise apart. For the time being we use this as justification for our approach and later demonstrate in a user study that on average people prefer our results over related work.

Our paper's contributions are as follows: We present an image downscaling algorithm that preserves high-frequency details—even during strong downscaling, *i.e.*, with large scaling factors—and can be applied to videos with hardly any temporal artifacts. It is based on two convolutional filters, but its algorithmic complexity is only linear in the number of input pixels independent of the filter kernel size. One major benefit of our filter-based algorithm is that it is implementable with fine-grained parallelism on SIMD hardware. We demonstrate this in a GPU version running in real-time even on 24 M pixel images.

2 Related Work

Initially, all downscaling approaches were based on filters that were designed to remove high frequencies while preserving all frequencies that will not lead to aliasing as defined by Shannon [1949]. Examples for frequency-based filters are the box, the bicubic, and the Lanczos filter [Duchon 1979]. However, since most image details are contained in the high frequencies, they get removed during strong downscaling. The same holds for the bilateral filter. Therefore, different approaches for this scenario have been investigated.

Triggs [2001] proposed an approach to design filters that allow for reconstruction of the high resolution input image with as little error as possible. This, however, leads to the same type of filters as frequency-based approaches so that small details of the original image are still lost. Nehab and Hoppe [2011] provide an efficient approach to least-squares downscaling. Since the setting is similar to Triggs' approach, fine details are lost during downscaling.

Recent works on thumbnail generation [Samadani et al. 2010; Trentacoste et al. 2011] try to maintain an input image's quality in the output by preserving the blur and noise in the image. This way, a user can judge the quality of the original image. They achieve this by artificially inserting noise and blur into the output. In contrast, the focus of our algorithm is to preserve important visual details and not necessarily noise, let alone blur.

Kopf *et al.* [2013] suggest an approach based on a joint bilateral filter. For each output pixel, they define a corresponding region in the input image. In contrast to pure segmentation, each input pixel may have a weighted contribution to a number of output pixels. They also present an optional set of constraints that avoid excessive deformation of the input image and smooth edges. Instead of optimizing a virtual reconstruction step to come as close to the original as possible, Öztireli and Gross [2015] optimize the downsampled image to be close to the original in terms of the structural similarity index. However, since this approach handles color channels individually, it may produce wrong colors (see the comic art in Figure 5d). Also, it has issues when downscaling line art (Figure 5e).

3 Algorithm

Our algorithm first computes a smooth, downscaled version of the original image as guidance image. Given our focus on strong downscaling of large images, we do this very rapidly based on a box filter. The final image is then assembled from the input image using a convolutional filter that gives more weight to pixels that differ from their local neighborhood represented by the guidance image. This can be seen as downscaling via a joint bilateral filter [Petschnigg et al. 2004; Eisemann and Durand 2004; Kopf et al. 2007] with a range kernel that—contrary to the normal bilateral filter—favors color differences instead of punishing them.

More formally, given an input image I with a size of $w_I \times h_I$ pixels and a desired output image O with size $w_I/d \times h_I/d = w_O \times h_O$ pixels, we first create a box-filtered, downscaled image I_D of size $w_O \times h_O$. We denote the rectangular patch of pixels in I that are mapped to a pixel p in O as $\Omega_I(p)$. For simplicity we assume that d is integer. If it is not, we apply weights that specify what fraction of an input pixel belongs to $\Omega_I(p)$. The guidance image $\tilde{I}$ is then computed from I_D using a convolution as

$$\tilde{I} = I_D \otimes \frac{1}{16}\left(\begin{smallmatrix}1&2&1\\2&4&2\\1&2&1\end{smallmatrix}\right) \quad (1)$$

Downscaling I with a box filter to obtain I_D and convolving I_D with Equation 1 is equivalent to directly downscaling I with

$$\frac{1}{16d^2}\left(\begin{smallmatrix}1\cdot\mathbf{1}_{d,d}&2\cdot\mathbf{1}_{d,d}&1\cdot\mathbf{1}_{d,d}\\2\cdot\mathbf{1}_{d,d}&4\cdot\mathbf{1}_{d,d}&2\cdot\mathbf{1}_{d,d}\\1\cdot\mathbf{1}_{d,d}&2\cdot\mathbf{1}_{d,d}&1\cdot\mathbf{1}_{d,d}\end{smallmatrix}\right)$$

($\mathbf{1}_{d,d}$ being the $d \times d$ matrix of ones), a strongly discretized approximation of a $3d \times 3d$ Gaussian. As a result, $\tilde{I}$ is a smooth, downscaled version of I. This approximation is much faster than the full Gaussian (especially for large d) and gives very similar results: Figure 2 (left) shows almost no difference between a guidance image from a $3d \times 3d$ Gaussian ($\tilde{I}_{\text{Gauss}}$) and our approximation ($\tilde{I}$). In contrast, the guidance image from a $3d \times 3d$ box filter ($\tilde{I}_{\text{Box}}$) is clearly different from $\tilde{I}_{\text{Gauss}}$. A numerical analysis on all images of Section 5 confirms this: The average per-pixel difference (L^2 norm over 8-bit R, G, and B channel) is 1.6 for the Gaussian vs. our approximation and 5 for the Gaussian vs. the box filter.

When computing I_D and $\tilde{I}$, pixels along the image edges are treated in a standard way by adjusting the convolution kernel to only cover valid pixels and setting the normalization coefficient appropriately. The output image O is then computed via a joint bilateral filter:

$$O(p) = \frac{1}{k_p} \sum_{q \in \Omega_I(p)} I(q) \left(\left\| I(q) - \tilde{I}(p) \right\|_2 \Big/ V_{\max} \right)^{\lambda} \quad (2)$$

with $k_p = \sum_{q \in \Omega_I(p)} (\| I(q) - \tilde{I}(p) \|_2 \,/\, V_{\max})^{\lambda}$ being the usual normalization factor that sums up the kernel values. $V_{\max}$ is the norm

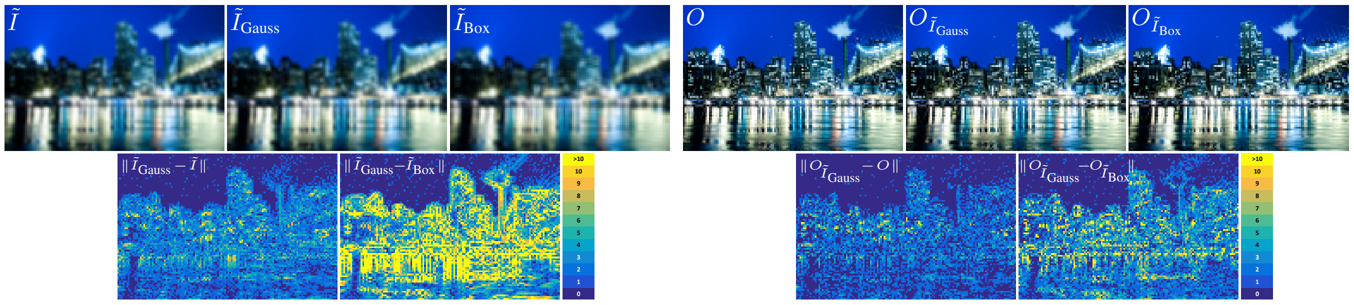

$\tilde{I}$
$\tilde{I}_{\text{Gauss}}$
$\tilde{I}_{\text{Box}}$
O
$O_{\tilde{I}_{\text{Gauss}}}$
$O_{\tilde{I}_{\text{Box}}}$
$\|\tilde{I}_{\text{Gauss}} - \tilde{I}\|$
$\|\tilde{I}_{\text{Gauss}} - \tilde{I}_{\text{Box}}\|$
$\|O_{\tilde{I}_{\text{Gauss}}} - O\|$
$\|O_{\tilde{I}_{\text{Gauss}}} - O_{\tilde{I}_{\text{Box}}}\|$
>10
10
9
8
7
6
5
4
3
2
1
0

Figure 2: *An input image (not shown) downscaled with a factor $d \approx 42$.* ***Top:*** *Guidance image obtained with our approximation ($\tilde{I}$), a $3d \times 3d$ Gaussian ($\tilde{I}_{Gauss}$), and a $3d \times 3d$ box filter ($\tilde{I}_{Box}$), and the output images (O, $O_{\tilde{I}_{Gauss}}$, $O_{\tilde{I}_{Box}}$) obtained with these three guidance images.* ***Bottom:*** *Differences (L^2 norm over 8-bit R, G, and B channel) between the guidance images respectively output images.*

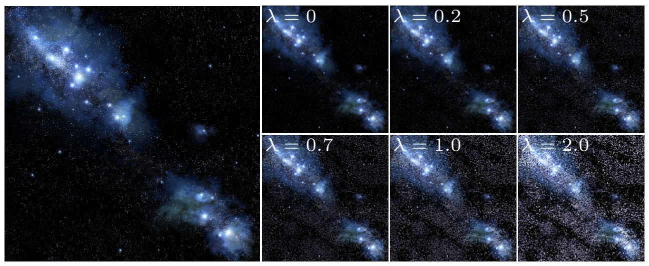


Figure 3: ***Left:*** *Input image (1200×1200 px).* ***Right:*** *Results (240×240 px) with different λ.*

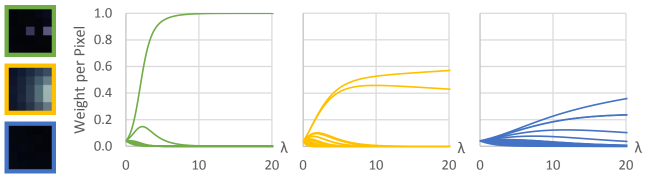


Figure 4: ***Left:*** *Three input image patches.* ***Right:*** *The weights for each of these patches' pixels with respect to λ.*

of the color space's maximum value (for unnormalized 8-bit RGB images: $V_{\max} = \sqrt{3 \cdot 255^2}$) and normalizes all values into the $[0, 1]$ range. In the following we refer to a pixel's $\|I(q) - \tilde{I}(p)\|_2$ as its distinctness (since it measures the pixel's difference from the local neighborhood), to $(\|I(q) - \tilde{I}(p)\|_2 \,/\, V_{\max})^\lambda$ as its range kernel (from the joint bilateral filter), and to $\frac{1}{k_p}(\|I(q) - \tilde{I}(p)\|_2 \,/\, V_{\max})^\lambda$ as its weight. Equation 2 has two big differences to the regular joint bilateral filter: First, instead of *decreasing*, the range kernel *increases* with increasing distinctness, *i.e.*, it *favors* differences to the local pixel neighborhood represented by $\tilde{I}$. Second, the (implicit) spatial kernel is not a Gaussian but a rectangular function that is 1 within $\Omega_I(p)$ and 0 elsewhere.

We now explore the influence of λ. Since the $V_{\max}$-normalized distinctness is in $[0, 1]$, exponentiating it with λ increases it for $\lambda < 1$ and decreases it for $\lambda > 1$. But how this influences a pixel's *weight* depends not only on its own distinctness but also on that of all other pixels in $\Omega_I(p)$, because they are all connected through k_p. Figure 3 shows an input image with strong high-frequency content. Figure 4 shows three patches from this input image and one graph per patch. These graphs contain one curve per pixel within the corresponding patch, that shows how the weight for this pixel behaves under varying λ. For $\lambda = 0$ all pixels are uniformly (independent of their distinctness) assigned a range kernel of 1 and thus a weight of $\frac{1}{k_p} = \frac{1}{d^2}$. In this case our filter is a box filter.

When λ increases, the set of curves splits into three subsets (most easily seen in the green graph): The first subset corresponds to pixels whose distinctness is maximal within their patch. For $\lambda \to \infty$, such a pixel eventually dominates its patch, its patch's k_p converges against its range kernel, and its weight thus converges to 1 (or $\frac{1}{m}$ if there are m maximally distinct pixels within a patch). The second subset are pixels whose distinctness is smaller than the average within their patch. When λ increases, their weight monotonically converges to 0. The third subset are pixels whose distinctness is greater than the average but not the maximum within their patch. Increasing λ first increases their weight up to a turning point. After this, the patch's pixel with the maximal distinctness (first subset) starts dominating and the weight of the non-maximal pixel monotonically converges to 0.

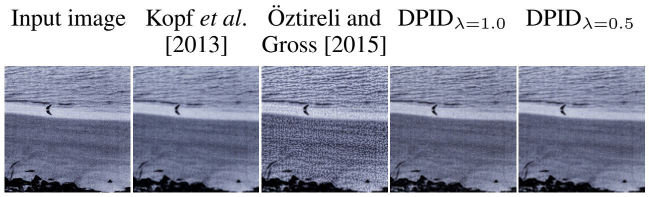
With λ we can tune the amplification of the weights of pixels that represent detail—from a box filter over an emphasis of distinct pixels towards a selection of only the most distinct pixels. However, a very large λ is in general undesirable because the results are too extreme and unsightly. Figure 3 (right) shows outputs for various λ. For $\lambda = 2$, details are overemphasized. In our experience, values of $\lambda > 1$ typically produce undesirable results. Since we can only examine a finite set of λs in our user study, we choose $\lambda = 1.0$ and 0.5 and refer to the resulting algorithms as "DPID$_{\lambda=1.0}$" and "DPID$_{\lambda=0.5}$".

To evaluate how our Gaussian approximation (Equation 1) influences the final output, Figure 2 (right) shows three output images: Ours (O), one from a Gaussian guidance image ($O_{\tilde{I}_{\text{Gauss}}}$), and one from a box-filtered guidance image ($O_{\tilde{I}_{\text{Box}}}$). For $\lambda = 1.0$ the per-pixel difference (averaged over all images from Section 5) is 1.0 for $O_{\tilde{I}_{\text{Gauss}}}$ vs. O and 2.1 for $O_{\tilde{I}_{\text{Gauss}}}$ vs. $O_{\tilde{I}_{\text{Box}}}$. Our approximation's error seems tolerable, even though it is a strong discretization since the average d is 25.

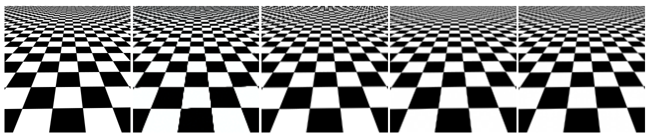
Our full algorithm's complexity is $\mathcal{O}(w_I h_I)$: We iterate once over I to compute I_D, iterate over I_D to compute $\tilde{I}$, and iterate over I again to compute the output image O.

4 Results

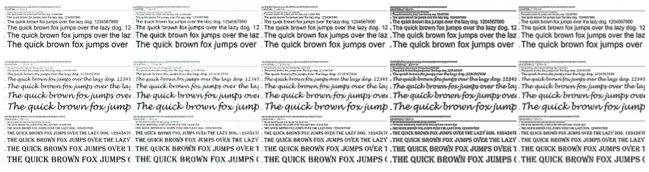
We applied our algorithm to an extensive set of images taken from the Yahoo 100 M image dataset [Thomee et al. 2016] and the NASA Image Gallery [2016], providing a large variety of images. In addition, we use 44 images taken from the user studies or the supplemental material of Kopf *et al.* [2013] and Öztireli and Gross [2015] to be comparable with previous work. Our results are available in



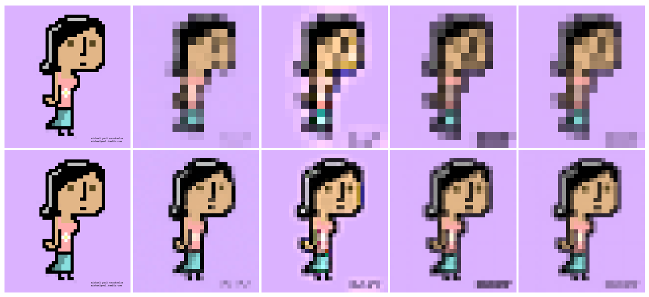
(a) *Noisy image, 2048×2048 px → 128×128 px.*



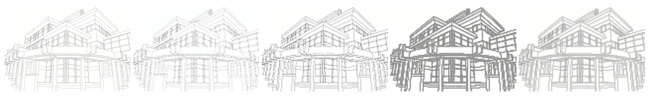
(b) *Checker board, 4670×4670 px → 128×128 px.*



(c) *Text in three different fonts, 1408×580 px → 156×64 px.*



(d) *Comic art, 1350×1500 px → 16×18 px resp. 32×36 px.*



(e) *Line art, 3302×2192 px → 128×85 px.*

Figure 5: *Downscaling challenging input.*

Figures 1, 3, 5, 8, 11, and in the supplemental material. For an objective evaluation of the perceptual quality of the results, we performed a user study using a subset of the images (see Section 5).

We now discuss some challenging cases shown in Figure 5: Kopf *et al.* [2013] perform well on noise (Figure 5a) and the checkerboard (5b). On comic art (5d) their contours are the sharpest, but they struggle with line art (5e). Öztireli and Gross [2015] amplify noise (5a), produce aliasing on the checkerboard (5b), produce ringing and wrong colors (colors not present in the input) in comic art (5d), and struggle with line art (5e). DPID$_{\lambda=1.0}$ and DPID$_{\lambda=0.5}$ both handle noise (5a) well if the distribution of the noise is not skewed. In this case contributions above and below the mean cancel each other out. A pixel only influences its output if it differs from the mean and has no counterpart with a difference of same absolute value and opposite sign (such as the stars in Figure 3). Further, DPID$_{\lambda=1.0}$ and DPID$_{\lambda=0.5}$ both perform well on the checkerboard (5b) and on comic art (5d). DPID$_{\lambda=1.0}$ fattens text (5c) and line art (5e) too much, whereas DPID$_{\lambda=0.5}$ does a good job on both.

To demonstrate that our algorithm performs well on videos, we applied it to the 4K video "Fuerteventura 4K – A Timelapse Adventure" [Schall and Schmid 2015]. We downscaled it from 4096×2160 px to 320×180 px. In our supplemental material we compare DPID$_{\lambda=1.0}$ and DPID$_{\lambda=0.5}$ against the Lanczos filter and Öztireli and Gross [2015]. Further, we provide a short sequence using Kopf *et al.*'s [2013] algorithm. To downscale this short, 15 second long video, Kopf *et al.*'s algorithm required more than 16 hours. Although Kopf *et al.* stated that their algorithm is not suited for videos, it shows very few temporal artifacts.

5 User Study

We validated our algorithm in a user study as follows: Analogous to Kopf *et al.* [2013] and Öztireli and Gross [2015], we showed users combinations of an input image and two downscaled versions for which they had to decide without any time limit "which of the two represents a better downscaled version of the input image" or indicate no preference. Input images were displayed unscaled on a 4K monitor. Users could not zoom but only pan input images larger than 4K. The downscaled images were displayed unscaled on a 1080 p monitor (using a second 4K monitor instead would have made most downscaled images < 3 cm high). Both monitors were calibrated to the CIE D65 white point and $160\,\mathrm{cd/m^2}$ to ensure similar luminance and color display. All participants underwent an eye test to ensure they can see fine details and are not color blind.

We focus on extreme downscaling of high resolution input images, e.g., from native digital camera resolution of several megapixels to thumbnail size. In contrast, Öztireli and Gross [2015] used images with an average of 0.12 Mpx in their study and downscaled all of them with a factor of 3.1. Since this is far away from our use case, we selected a different set of images: We picked (randomly, to have input with diverse properties) 19 images from the Yahoo 100 M image dataset [Thomee et al. 2016] and one image from the NASA Image Gallery [2016]. Our input images have an average resolution of 9.4 Mpx and we downscaled each of them to a width of 128 px, which is important because downscaling results are likely depending on the downscaling factor: *E.g.*, in Figures 1 and 6 the results for subsampling are disastrous whereas subsampling performed remarkably well in Öztireli and Gross's study. We will come back to this issue in our final discussion in Section 7.

For all 20 input images we showed every participant all pairs from $\{$Bicubic, Bilateral, Subsampling, Gaussian, Lanczos, Nehab, Kopf, Öztireli$\} \times \{$DPID$_{\lambda=1.0}$, DPID$_{\lambda=0.5}\} \cup \{($DPID$_{\lambda=1.0}$, DPID$_{\lambda=0.5})\}$, resulting in $20 \cdot (8 \cdot 2 + 1)$ pairwise decisions per user. In addition, we showed 20 downscaled pairs a second time to check whether users decide randomly or deliberately. In this check the average user achieved 81 % self-agreement and the worst achieved 63 %. Thus, we did not discard any user's results. All image pairs were shown in random temporal order and both images of a pair were shown in random spatial on-screen order.

We had 26 study participants, from which 8 ranked their image processing knowledge as 1 on a five-level Likert scale, 6 as level 2, 2 as level 3, 6 as level 4, and 4 as level 5. We analyzed the test answers using Welch's t-test as well as the likelihood-ratio test and found that except for experience level 3 answers did not depend on experience. For level 3 there were too few participants to make definitive statements about this group.

The user study's cumulative results can be found in Figure 6. On average, both DPID$_{\lambda=1.0}$ as well as DPID$_{\lambda=0.5}$ outperform all other methods. The exact performance depends on the image under consideration: Figures 8a and 8c show an image and the corresponding study results, where our algorithm achieves 85 % to 100 % preference over all other algorithms. On the contrary, Figures 8b and 8d show a failure case, where both subsampling as well as Öztireli and Gross's algorithm outperform ours. One striking property of this image is a strong dominance of high-frequency details.

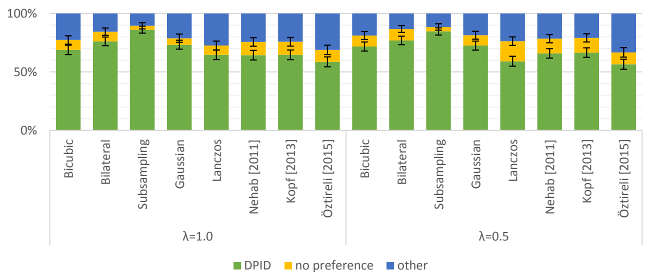


Figure 6: *User preference for our algorithm vs. related work. Each data point is an average over 26 study participants and 20 input images. The error bars indicate the 95 % confidence interval for both our approach and others.*

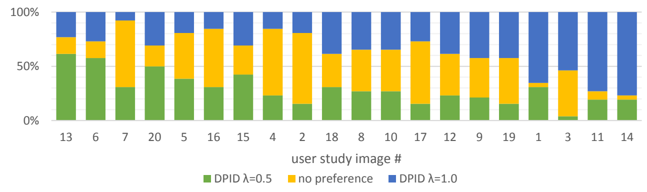


Figure 7: *User preference for DPID$_{\lambda=1.0}$ vs. DPID$_{\lambda=0.5}$ on the different user study input images, sorted by decreasing "preference for DPID$_{\lambda=0.5}$" $+0.5 \cdot$ "no preference".*

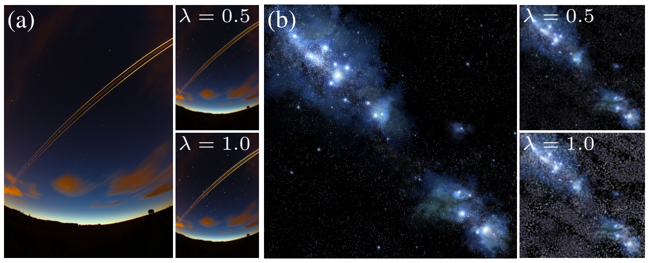
While in general both our algorithm variants outperform related work, the best choice of λ depends on the input image: Figure 7 shows the results of a comparison of DPID$_{\lambda=1.0}$ vs. DPID$_{\lambda=0.5}$. The average over all images as well as the results of some individual images, such as Images 4 or 20, are more or less evenly split between both λs. However, for some images users had a strong preference for one or the other λ: *E.g.* for Images 11 and 14 users preferred $\lambda = 1.0$, while for Images 13, 6 and 20 $\lambda = 0.5$ is preferred. For others, such as Images 2 or 7, 'no preference' was the most frequent choice. Figure 11 shows examples for each of these three cases. Due to this variability, it may be preferable in an image editing software to let experienced users choose λ.

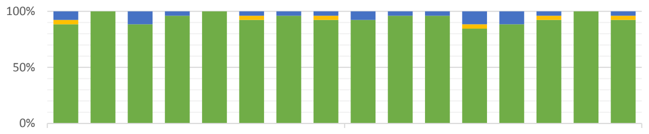
6 Runtime

We designed our algorithm for high performance on large images. The runtime when testing our CUDA implementation on a four year old consumer GPU (GTX 680) is shown in Figure 9. Runtime scales linearly with input and output image size and reaches real-time performance (40 ms) even for downscaling 24 M to 2 M pixel images. Our algorithm requires 1.5 s (including file I/O) for all 126 benchmark images. In contrast, even with perfect parallelization on a 6-core Intel i7-3930K, Kopf *et al.*'s [2013] algorithm requires more than 1 h. Öztireli and Gross [2015] require 5.2 min. Their code is, however, written in Matlab and not optimized for speed.

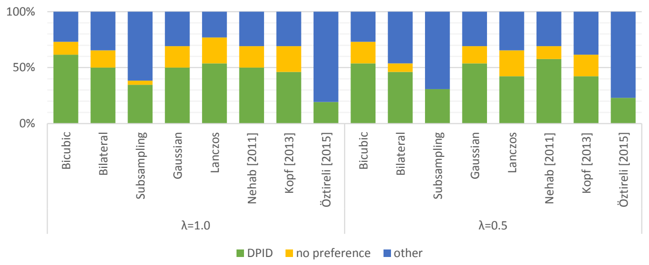
For a fairer comparison we do a theoretical analysis: Assuming that our and Öztireli's image operations are all purely memory bandwidth limited, the runtime of both is dominated by the box filter on I (all subsequent filters operate on much smaller images). For N input pixels and a downscaling factor d, the required bandwidth is $2N + 4\frac{N}{d} + 56\frac{N}{d^2}$ for Öztireli's and $2N + \frac{14}{3}\frac{N}{d} + 6\frac{N}{d^2}$ for our algorithm (see Figure 10). Both converge to $2N$. Our algorithm requires less bandwidth for $d < 75$ and after that Öztireli is marginally better but has a fixed overhead since it requires more filter operations.

Depending on the application, our code's performance could be further increased, *e.g.*, by only supporting integer scaling factors, or by tailoring it for a specific factor which would allow us to remove conditionals from the code.





(c) *Users' algorithm preference for image (a).*



(d) *Users' algorithm preference for image (b).*

Figure 8: *Images where our algorithm has been chosen (a) most and (b) least frequently. On image (a) our algorithm keeps the stars and light rays intact and it is superior to the other algorithms (see preferences in (c)). On (b) our algorithm overemphasizes the stars and it was chosen about 50 % of the time (see preferences in (d)), except when compared with subsampling or Öztireli [2015].*

7 Discussion and Conclusion

In this paper we presented an algorithm that preserves visually important details in downscaled images and is especially suited for large downscaling factors. In a user study we showed that in a very heterogeneous group with varying image processing experience our algorithm was usually preferred over algorithms from related work. Further, as shown in our supplemental material, the algorithm can also be applied to videos with hardly any temporal artifacts.

Our algorithm handles noise well if its distribution around the mean is not skewed. Our algorithm's most striking limitations are the fattening of thin edges/dots and aliasing. This is a consequence of our definition of detail: The range kernel in Equation 2 is not an ideal low-pass filter and does not remove frequency information above the Nyquist limit, which may introduce aliasing. While this may contravene established filtering theory, our study showed that users actually prefer this—at least on still images: DPID$_{\lambda=1.0}$, DPID$_{\lambda=0.5}$, and Öztireli and Gross's algorithm all produce aliased results but performed well in our study. Further, in Öztireli and Gross's study users even preferred the results of naïve subsampling. We could not reproduce this finding and attribute this to our much larger input image sizes and downscaling factors, for which subsampling produces very disturbing artifacts. In our algorithm we can usually alleviate aliasing and edge fattening by decreasing λ.

Future Work. In the future we plan to automatically determine optimal values for λ, depending on the image content (*c.f.* Figure 11). For this we want to use machine learning to predict the

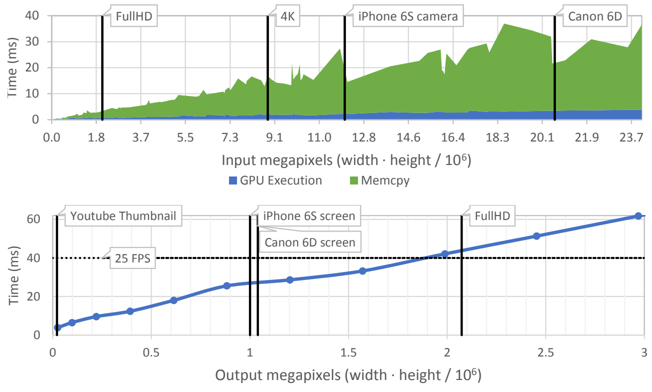


Figure 9: *Runtimes for various input and output image sizes. Each data point is an average of 100 runs on a GTX 680.* ***Top:*** *126 input images with 0.16 M–24 Mpx, all downscaled to 128 px width. The high time for memcopy is due to PCIe 2.0's bandwidth of* 8 GB/s *compared to* 192 GB/s *to access the GPU memory.* ***Bottom:*** *One 24 Mpx input image downscaled to 128, . . . , 3 Mpx.*

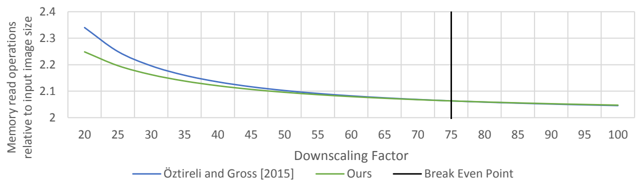


Figure 10: *Theoretical runtime comparison of our approach versus Öztireli and Gross [2015], assuming that all operations are purely memory bandwidth limited.*

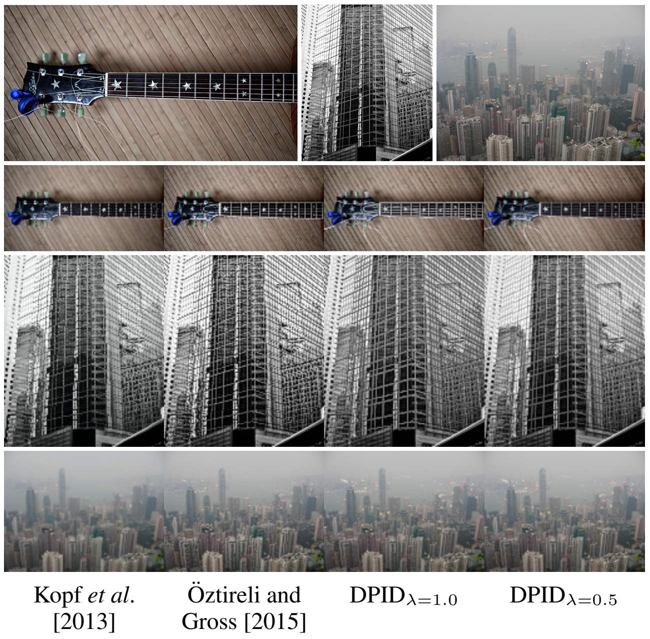


Figure 11: *Input (1st row) and downscaling results (2nd–4th row) for Images 11, 20, and 2 of our user study. For Image 11 users preferred $\lambda = 1.0$ over $\lambda = 0.5$, for Image 6 they preferred the opposite, and for Image 2 they had no preference for either.*

best λ for a given image. This requires a user study with thousands of input images and various λs to obtain label annotations.

Acknowledgements

We used images from the Yahoo 100 M image project [Thomee et al. 2016], NASA Image Gallery [2016], and Chi King (Flickr). We sincerely thank Kopf and Öztireli for their help in comparing with their works, and Schall and Schmid [2015] for granting us usage of their video. The work of N. Weber is supported by the 'Excellence Initiative' of the German Federal and State Governments and the Graduate School of Computational Engineering at Technische Universität Darmstadt. S. Guthe and M. Waechter are supported through the 7th Framework Programme (project CRPlay) resp. the Intel Visual Computing Institute (project RealityScan).

References

BEGHDADI, A., LARABI, M.-C., BOUZERDOUM, A., AND IFTEKHARUDDIN, K. 2013. A survey of perceptual image processing methods. *Signal Processing: Image Communication 28*, 8.

DUCHON, C. E. 1979. Lanczos filtering in one and two dimensions. *Journal of Applied Meteorology 18*, 8.

EISEMANN, E., AND DURAND, F. 2004. Flash photography enhancement via intrinsic relighting. In *SIGGRAPH*.

KOPF, J., COHEN, M., LISCHINSKI, D., AND UYTTENDAELE, M. 2007. Joint bilateral upsampling. In *SIGGRAPH*.

KOPF, J., SHAMIR, A., AND PEERS, P. 2013. Content-adaptive image downscaling. In *SIGGRAPH Asia*.

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION, 2016. NASA image gallery. nasa.gov/multimedia/imagegallery.

NEHAB, D., AND HOPPE, H. 2011. Generalized sampling in computer graphics. Tech. Rep. MSR-TR-2011-16.

ÖZTIRELI, A. C., AND GROSS, M. 2015. Perceptually based downscaling of images. In *SIGGRAPH*.

PETSCHNIGG, G., AGRAWALA, M., HOPPE, H., SZELISKI, R., COHEN, M., AND TOYAMA, K. 2004. Digital photography with flash and no-flash image pairs. In *SIGGRAPH*.

SAMADANI, R., MAUER, T. A., BERFANGER, D. M., AND CLARK, J. H. 2010. Image thumbnails that represent blur and noise. *Transactions on Image Processing 19*, 2.

SCHALL, S., AND SCHMID, L., 2015. Fuerteventura 4K – A Timelapse Adventure. youtu.be/40s_HSZkt3U.

SHANNON, C. E. 1949. Communication in the presence of noise. *Proc. IRE 37*.

THOMEE, B., SHAMMA, D. A., FRIEDLAND, G., ELIZALDE, B., NI, K., POLAND, D., BORTH, D., AND LI, L.-J. 2016. YFCC100M: The new data in multimedia research. *Communications of the ACM 59*, 2.

TRENTACOSTE, M., MANTIUK, R., AND HEIDRICH, W. 2011. Blur-aware image downsampling. *Comput. Graph. Forum 30*, 2.

TRIGGS, B. 2001. Empirical filter estimation for subpixel interpolation and matching. In *ICCV*.