Content-Adaptive Image Downscaling

Johannes Kopf
Microsoft Research

Ariel Shamir
The Interdisciplinary Center

Pieter Peers
College of William & Mary

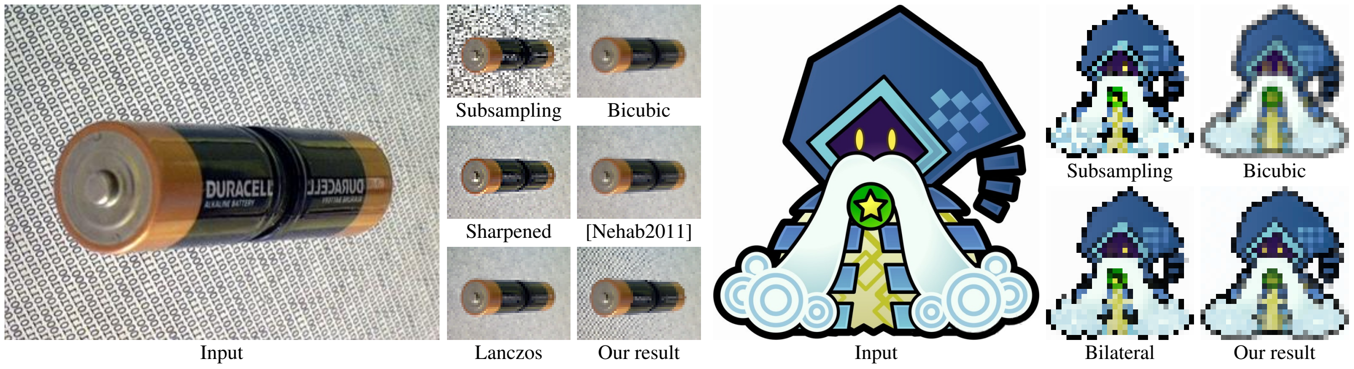


Figure 1: *Previous* content-insensitive *downscaling methods have to compromise between preserving sharpness while introducing aliasing artifacts (e.g., subsampling), or preventing aliasing at the expense of smoothing out fine details and edges (e.g., Bicubic, Lanczos, etc.). Our new* content-adaptive *algorithm provides a more balanced result, that is crisp and contains neither noise nor ringing, and mostly avoids aliasing artifacts. ("Merlon" input image © Nintendo Co., Ltd.)*

Abstract

This paper introduces a novel *content-adaptive* image downscaling method. The key idea is to optimize the shape and locations of the downsampling kernels to better align with local image features. Our content-adaptive kernels are formed as a bilateral combination of two Gaussian kernels defined over space and color, respectively. This yields a continuum ranging from smoothing to edge/detail preserving kernels driven by image content. We optimize these kernels to represent the input image well, by finding an output image from which the input can be well reconstructed. This is technically realized as an iterative maximum-likelihood optimization using a constrained variation of the Expectation-Maximization algorithm. In comparison to previous downscaling algorithms, our results remain crisper without suffering from ringing artifacts. Besides natural images, our algorithm is also effective for creating *pixel art* images from vector graphics inputs, due to its ability to keep linear features sharp *and* connected.

CR Categories: I.4.1 [Computer Graphics]: Image Processing and Computer Vision—Sampling

Keywords: Images, Downscaling

Links: DL PDF WEB CODE

1 Introduction

Downscaling is perhaps *the* most commonly used image operation today. We rarely view a photo we just took at its original resolution anymore. Instead, it is instantly reduced from its original multi-megapixel size to much smaller dimensions to be viewed on a camera viewfinder, on a computer or mobile screen, or on the web.

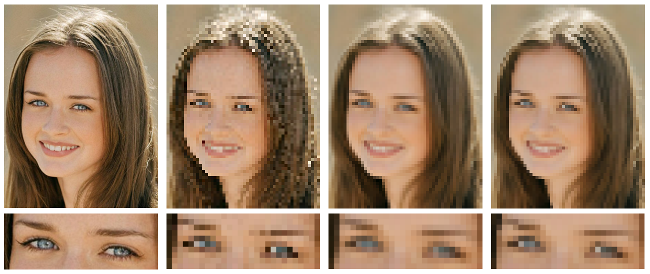


Figure 2: *Balancing sharpness and antialiasing: the* subsampled *result is sharp everywhere but suffers severely from noise and aliasing. On the other hand, the* bicubic *result, over-smoothes detail in the face.* Our algorithm *avoids both problems and produces a crisp, noise-free image that exhibits only minimal aliasing.*

The de facto standard for image downscaling are linear filters, originating from the signal processing community [Wolberg 1990]. Here, the image is first convolved with a low-pass kernel to reduce the bandwidth before it is resampled to the final resolution. The filtering pushes the image below the Nyquist frequency and prevents aliasing, but as a side effect the result might suffer from loss of fine details and blurring of sharp edges (Figure 2c). Sharpening these images or using kernels that more closely model a *sinc* filter (e.g., Lanczos) can cause ringing (Figure 3), while simply subsampling the image *without* prefiltering typically leads to strong aliasing artifacts (Figure 2b). Because all these methods are *content-invariant* (i.e., they use invariant kernels), the tradeoff between preserving detail and preventing aliasing is global.

In this work we present a new *content-adaptive* downscaling algorithm. As in classic methods, the output pixels are computed as a weighted sum of the input pixels. This can be interpreted as associating an averaging kernel to every output pixel. Our key idea is to adapt the shape of these kernels in order to better align them with local image features (Figure 2d). Following previous techniques such as bilateral filtering [Tomasi and Manduchi 1998] and mean shift [Comaniciu et al. 2002], we use kernels that are a combination of a spatial Gaussian kernel ensuring locality, and a color space Gaussian kernel for alignment with the image content. The simple parametric form of these kernels achieves a good trade-off between

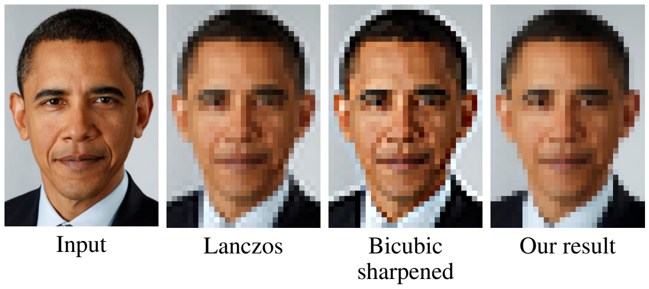


Figure 3: *The Lanczos kernel or Photoshop's "Sharpen" filter come at the expense of ringing artifacts due to negative lobes and oscillations in their kernels. Our kernels are strictly positive and mostly without oscillations, yielding results that are practically free of ringing artifacts.*

a small number of parameters to optimize, and sufficient flexibility to take on the shape of image features. We formulate the problem as a constrained reconstruction problem and optimize for a set of kernels whose combination would best reconstruct the original image. The optimized kernels are constrained to remain compact, simple, and "blob-shaped", since they correspond to simple pixels in the output image: the color of the output pixel is computed as the sum of kernel-weighted input pixel colors.

In classic methods, all kernels have the same shape and are arranged in a regular grid (Figure 4b). Using just bilateral kernels without optimizing their parameters can align them with image features. However, just like the subsampling method, they might "miss" features depending on the location of their center (Figure 4c). Our kernels also align themselves with curved features in the image, but since they are optimized to represent the input image better, they tend to not miss features (Figure 4d). This property is of particular importance when downscaling images that contain fine lines, like cartoon art (Figures 1, 8, 15)

Our local kernel parameters are derived through an iterative maximum likelihood optimization using a constrained variation of the Expectation-Maximization algorithm. This optimization yields a continuum of local kernels in a single framework, ranging from smoothing in some places to edge/detail preserving filters in others, depending on local image content. By locally controlling sharpness against antialiasing, we achieve a good balance of both objectives.

Our algorithm is also useful for downscaling cartoon and vector art, and can also be combined with palette reduction to create *pixel art* imagery. Because pixels usually appear bigger in these images, the blurring of linear resampling filters shows even more severely. Subsampling can produce sharp results, but it takes only a fraction of the input pixels into account, and might miss features. This can lead to broken features and disconnected lines. Our optimized kernels also produce crisp pixel art, while minimizing broken or disconnected features (see Figure 1, 8, 15).

We tested our algorithm on a wide range of natural and vector graphic input images. We compare our results against an extensive set of alternative downscaling methods. In addition, we performed a user study to validate the perceptual quality of our results. Our method especially improves the quality of downscaled images exhibiting small details or stochastic textures.

2 Previous Work

Classical image downscaling techniques find their origin in sampling theory [Shannon 1949], and prefilter and reconstruct the signal with a spatially constant lowpass filter in order to prevent aliasing in the reconstructed signal. However, by suppressing high frequencies they also tend to blur the signal. Filters that are designed to more closely model the (theoretically ideal) sinc filter, such as Lanczos, come at the expense of negative lobes that can produce ringing artifacts near strong image edges (Figure 3). Many filters (e.g., bilinear, bicubic, etc.) have been developed [Wolberg 1990], and even mined from image data [Triggs 2001], that balance mathematical optimality with perceptual quality of the downsampled result. Recent developments [Nehab and Hoppe 2011] add a correction stage on the discrete signal before reconstruction, resulting in less ringing at a similar computational cost. However, in none of these techniques the filtering kernel is adapted to the image content—a filter that removes aliasing in one area, might produce ringing in another. In contrast, our method adapts the shape and location of every kernel to the local image content, producing sharper results. Since our kernels are strictly positive and mostly without oscillations our results are practically free of ringing artifacts.

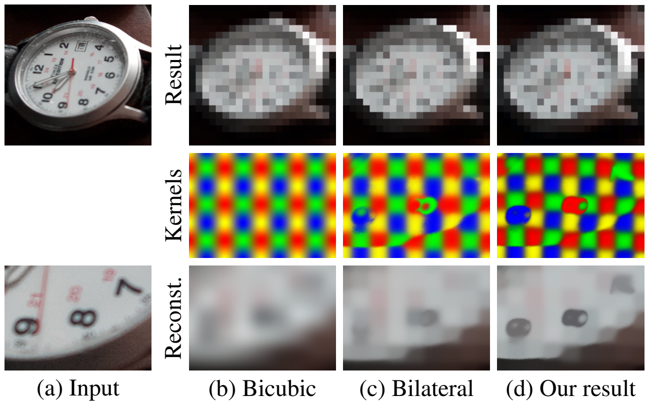


Figure 4: *Using averaging kernels for downscaling: (b) linear filters associate the same averaging kernel with every output pixel, forcing a global compromise between smoothing and aliasing. (c)* unoptimized *bilateral kernels adapt to the image, but might "miss" image features depending on the placement of their centers (e.g., the dial plate numbers). (d) Our* optimized *kernels adapt to the image even more, but do not suffer from missed or disconnected features.*

A different class of algorithms focuses on retargeting an image to different aspect ratios while preserving the salient content of the image as much as possible (e.g., [Avidan and Shamir ; Wolf et al. 2007; Rubinstein et al. 2009; Karni et al. 2009]). However, these methods are mostly geared towards altering aspect ratio, and are only suited for moderate reductions in resolution (e.g., 25% to 50%). Furthermore, retargeting methods focus on preserving the salient content, and thus can alter the global image composition. Our method, on the other hand, maintains the image composition as much as possible.

Thumbnail creation can also be viewed as a method to reduce the size of an image. Suh et al. [2003] propose to compute thumbnails by selectively cropping and scaling images guided by face detectors or saliency maps. Samadani et al. [2007] preserve the original appearance in thumbnails by reintroducing artifacts such as blurring and noise lost in downscaling. Trentacoste et al. [2011] also reintroduce blur for previewing images on a camera's view finder based on a perceptual model. Instead of preserving the blurred appearance of the input image, our method targets the opposite by selectively removing blur to preserve small image detail and texture.

Gerstner et al. [2012] abstract an image in a low resolution representation with a reduced color palette. Their algorithm alternates between updating SLIC (Simple Linear Iterative Clustering) superpixels [Achanta et al. 2012] and refining the color palette. Similar to our method, SLIC also works in the joint 5D space of location and color. However, SLIC segments the image (i.e., hard assignment), whereas our method produces a soft assignment of input to output pixels, allowing us to better adapt to the underlying high resolution image.

Manson and Schaefer [2012] present a method for generating mipmaps that adapts the filtering to the texture content and surface parameterization. Inglis and Kaplan [2012] present a method for rasterizing vector line art at low resolution, using rules established by pixel artists to avoid certain artifacts like jaggies and preserve local continuity. However, in contrast to the general downsampling strategy employed by our method, the above techniques downsample image content based on application specific rules.

3 Algorithm

To find the local filtering kernels, we formulate our task as a reconstruction problem of the input image from a smaller set of local kernel functions w_k (probability density functions) with fixed colors $\mathbf{v}_k$, and defined in the *joint 5D space* of location and color. We interpret each input image pixel as a sample drawn randomly from one of the local kernels with uniform probability and then sampling this local kernel. Theoretically, these kernels can have any shape and location. However, since they correspond to output pixels, a number of constraints on the kernels are necessary due to the application to image downscaling. First, the number of kernels must be equal to the number of output pixels, hence, no kernel can vanish during optimization. Second, their position cannot vary too much from the output pixel grid and their size cannot vary too much from the output pixel size. Third, to prevent aliasing artifacts, we add orientation constraints as described in Section 4.

Denote $X = \{\mathbf{x}_i\}$ as the input image with pixels $\mathbf{x}_i = (\mathbf{p}_i, \mathbf{c}_i)$, where $\mathbf{p}_i$ are the spatial coordinates and $\mathbf{c}_i$ the corresponding *CIELAB* color of pixel i. Each reconstructed image pixel color $\mathbf{c}'_i$ is defined as a weighted sum of the kernel colors $\mathbf{v}_k$:

$$\mathbf{c}'_i = \sum_k \gamma_k(i)\mathbf{v}_k \tag{1}$$

where $\gamma_k(i) = \frac{w_k(i)}{\sum_n w_n(i)}$ is the (unknown) weight of kernel k at pixel i relative to all kernels overlapping pixel i. The (also unknown) kernels $w_k : \text{domain}(X) \to [0,1]$ are bilateral Gaussian kernels in our case, i.e., we denote the value of kernel k at pixel i as:

$$w_k(i) = \frac{1}{W_k} f_k(i) g_k(i), \tag{2}$$

where the normalization factor $W_k = \sum_j f_k(j) g_k(j)$ ensures that each w_k is a probability density function (i.e., integrates to 1). The spatial component f_k and the color component g_k are given by:

$$f_k(i) = \exp\left(-\frac{1}{2}(\mathbf{p}_i - \mu_k)^\top \Sigma_k^{-1}(\mathbf{p}_i - \mu_k)\right), \text{ and} \tag{3}$$

$$g_k(i) = \exp\left(-\frac{\|\mathbf{c}_i - \mathbf{v}_k\|^2}{2\sigma_k^2}\right), \tag{4}$$

where μ_k and Σ_k are the mean and covariance matrix of the spatial Gaussian, and $\mathbf{v}_k$ and σ_k are the mean and variance of the color space Gaussian. Note that the spatial Gaussian is characterized by a full covariance matrix and can therefore take on elliptical shapes, while the color space Gaussian is characterized only by a scalar variance and remains isotropic, as we operate in the perceptually uniform CIELAB color space.

Given our input image X, we search for the most probable set of parameters $\theta = \{\mu_k, \Sigma_k, \mathbf{v}_k, \sigma_k\}$ of the kernels, and set of unknown variables $\gamma_k(i)$ that can produce (reconstruct) this image. Using Bayes rule, this converts to maximizing the (log) likelihood of the input image given the model of the set of kernels:

$$\underset{\theta}{\text{argmax}} \; \Pr(X \mid \theta). \tag{5}$$

This problem is well known in statistics and solved using the Expectation-Maximization (EM) algorithm [Hastie et al. 2005] where the unknown variables $\gamma_k(i)$ can be seen as the probability of pixel i to be drawn from kernel k.

We **initialize** kernel k corresponding to output pixel (x, y) as:

$$\mu_k \leftarrow (x_k, y_k)^\top,\ \Sigma_k \leftarrow \begin{bmatrix} \frac{r_x}{3} & 0 \\ 0 & \frac{r_y}{3} \end{bmatrix},\ \mathbf{v}_k \leftarrow \left(\tfrac{1}{2}, \tfrac{1}{2}, \tfrac{1}{2}\right)^\top,\ \sigma_k \leftarrow 10^{-4}, \tag{6}$$

where (x_k, y_k) is the center of output pixel k scaled to input image dimensions, and r_x, r_y are the ratios of input and output image width and height, respectively.

In the **expectation step** we compute soft assignment probabilities of each pixel to each kernel, assuming the current estimate of the parameters is correct:

$$\gamma_k(i) \leftarrow \Pr(k \mid \mathbf{x}_i\,;\,\theta) = \frac{w_k(i)}{\sum_n w_n(i)}. \tag{7}$$

In other words, $\gamma_k(i)$ quantifies how much an input pixel i contributes relatively to the final color of the output pixel k.

In the **maximization step** we use these soft assignments in a weighted maximum-likelihood fit to update the estimate of the parameters θ:

$$\mu_k \leftarrow \frac{\sum_i \gamma_k(i)\mathbf{p}_i}{\sum_i \gamma_k(i)}, \tag{8}$$

$$\Sigma_k \leftarrow \frac{\sum_i \gamma_k(i)(\mathbf{p}_i - \mu_k)(\mathbf{p}_i - \mu_k)^\top}{\sum_i \gamma_k(i)}, \tag{9}$$

$$\mathbf{v}_k \leftarrow \frac{\sum_i \gamma_k(i)\mathbf{c}_i}{\sum_i \gamma_k(i)}. \tag{10}$$

Note, that we control the color space Gaussians' σ_k directly to constrain the *locality* and *edge orientations* as described in the next section.

We add a third **correction step** after every maximization step. In this step we enforce the different constraints specific to our downsampling problem (Section 4).

The algorithm proceeds iteratively, alternating between performing *expectation*, *maximization*, and *correction* steps, and terminates when convergence of the model parameters is reached. While the summations in Equations 8–10 are defined over the whole input image, this is not necessary in a practical implementation, since the kernels are constrained in size (see next section). Please refer to the supplemental document for a detailed pseudo-code description.

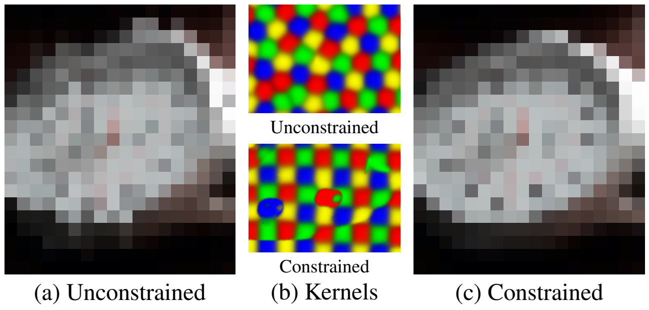
4 Constraints

Downscaling is a more restricted problem than general signal reconstruction, and thus finding the optimal kernel shape and weight that minimizes the reconstruction error is not a sufficient condition for obtaining a good downscaled result. We perform an additional **correction step** after every maximization step to enforce downscaling-specific constraints. We identified three types of such constraints:

1. **Spatial constraints:** Since the output pixel positions are arranged in a perfect lattice, it is important to constrain the locations of the corresponding kernels. Otherwise, they potentially move too far from their initial positions, which can lead to a scrambled result appearance (Figure 5).

2. **Locality:** since all output pixels have the same size, the kernels should also neither become too large or vanish. Their influence should remain local (Figure 6).

3. **Edge orientations:** The boundary between two neighboring kernels should have a similar orientation as the boundary between the two pixels in the output image (Figure 7).



(a) Unconstrained (b) Kernels (c) Constrained

Figure 5: ***Spatial constraints*** *are important to maintain the grid topology of the kernels.*

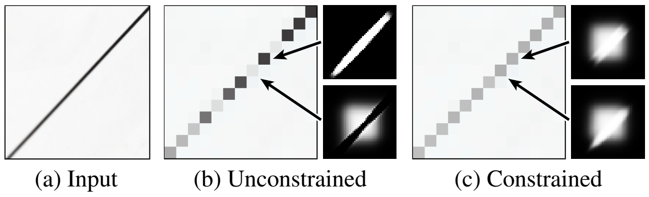

(a) Input (b) Unconstrained (c) Constrained

Figure 6: ***Enforcing local kernels.*** *The small side-images show the* normalized *kernels $\gamma_k(i)$. (b) The dark pixel's kernel grabs a long stretch of the line, while its neighbor's kernel completely avoids the line. (c) The smoothness σ_k of the kernels are increased until a more balanced configuration is reached.*

4.1 Spatial Constraints

To retain the characteristics of the grid topology of the output pixels, we bias the spatial mean μ_k towards a smooth grid-like topology.

First, we limit the extent the spatial mean can move by constraining μ_k to lie within a box, centered around the center of the output pixel. Second, we increase smoothness by moving μ_k halfway between its estimated location and the the mean of its four neighbors.

Formally, we update

$$\mu_k \leftarrow \text{clampBox}\left(\tfrac{1}{2}\mu_k + \tfrac{1}{2}\overline{\mu_k},\ \ (x_k, y_k)^\top \pm (\tfrac{r_x}{4}, \tfrac{r_y}{4})^\top\right), \tag{11}$$

where $\overline{\mu^k} = \left(\sum_{n \in N_k^4} \mu_n\right) / \left|N_k^4\right|$, and N_k^4 denotes the set of cardinal (4-connected) neighbors of k. The effect of these spatial constraints is shown in Figure 5.

4.2 Locality and Edge Orientations

We constrain the shape of the spatial variance Σ_k to avoid vanishingly small or exceedingly large kernels. We first obtain the singular value decomposition $(U, S, V^\star) = \text{SVD}(\Sigma_k)$, and modify the diagonal eigenvalue matrix S by clamping its elements to the interval $[0.05, 0.1]$, and finally set

$$\Sigma_k \leftarrow U S' V^\star, \tag{12}$$

where S' contains the clamped eigenvalues.

While Equation 12 imposes a hard constraint on the *spatial* component of our kernels to be relatively smooth, the *color* component causes them to align with features in the image. In general this is desired, however, in certain situations too much adaptation can cause visual artifacts. We use the color variance parameter σ_k, which is not estimated, to directly control the amount of adaptation, and, hence, the sharpness of our results. Small σ_k cause kernels to be more sensitive to color variations and have sharper transitions, while larger σ_k lead to smoother kernels.

The reason for not freely estimating σ_k is that the maximum likelihood estimation often yields too smooth configurations. Rather than estimating σ_k from the data, we control it explicitly to locally adjust the sharpness of the result. For most kernels we let σ_k remain at its initial "crisp" setting, and we only increase local smoothing under two specific conditions: (1) when kernels become too dominant compared to their neighbors, and (2) to prevent staircasing artifacts. Following each maximization step we search for kernels that match one of these conditions, and correct them by increasing σ_k by 10% (i.e., increase the smoothness of the color kernel).

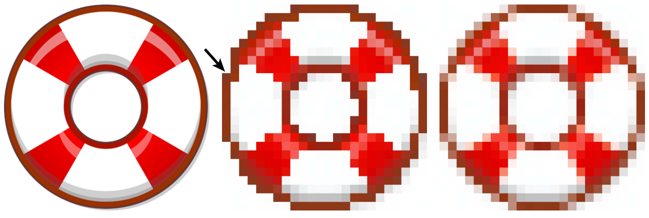

(a) Input (b) Unconstrained (c) Constrained

Figure 7: *We detect strong pixel edges whose orientation deviates from the edge between the corresponding kernels (e.g., the edge with the arrow), and selectively increase the smoothness of these kernels.*

Locality: While clamping the eigenvalues of Σ_k avoids large spatial Gaussians, the resulting bilateral kernels can still have a large spatial extent due to the normalization in Equation 7.

Figure 6b illustrates an instance of this problem. The small figures show the normalized kernel weights $\gamma_k(i)$ for two pixels. In subfigure (b) the kernels keep a small color variance, leading to a solution where one kernel grabs all dark pixels on the line, while the other kernels grab the light pixels in the surrounding. Even though the spatial weights of the kernel on the line fall off quickly, the normalization causes them to become large again, because the surrounding kernels have even lower weights due to the strong color adaption. This causes the line feature to become disconnected.

We correct this behavior by detecting kernels that are growing too strong in any direction, and then selectively increase their color variance. First, we compute the *directional* variance for each of the eight neighbors:

$$s_k^d = \sum_i \gamma_k(i) \max\left(0,\ (\mathbf{p}_i - \mu_k)^\top \cdot d\right)^2, \tag{13}$$

where $d \in \left\{(a,b)^\top \mid a, b \in [-1, 1]\right\} \setminus \left\{(0,0)^\top\right\}$ is the offset to one of the eight neighbors. If any directional variance s_k^d exceed a threshold of $0.2 r_x$ (where r_x is the ratio of the input image's width over the output image's width), we increase the smoothness of both the current kernel and the respective neighbor. The effect of this heuristic is illustrated in Figure 6c. The ability of our algorithm to keep linear features connected while maintaining sharpness can also be nicely observed in the pixel art results in Figures 1, 8, and 15.

Edge Orientations: A form of staircasing artifacts occur when the orientation of an edge between two dissimilar pixels in the output pixel grid is significantly different than the orientation of the edge between the corresponding kernels. This artifact is most visible on long lines with *almost* cardinal direction. Consider, for example, the horizontal edge marked with an arrow in Figure 7b. The corresponding image edge in the input image is almost vertical.

We selectively remove such false edges by increasing σ_k of the corresponding kernels. First, we detect strong edges between adjacent pixels by testing for neighboring kernels that have an abrupt transition. If k and n are horizontal or vertical neighbors, we measure the strength of the edge between the *normalized* kernels $f_{kn} = \sum_i \gamma_k(i)\gamma_n(i)$. If the transition between the two kernels is abrupt, there will be few pixels where *both* kernels take on large values, and, thus, f_{kn} will be small. We consider edges where $f_{kn} < 0.08 r_x r_y$ as strong. Next, we compute the direction of the edge between the kernels as $d_{kn} = \sum_i \nabla\left(\gamma_k(i) / (\gamma_k(i) + \gamma_n(i))\right)$. If

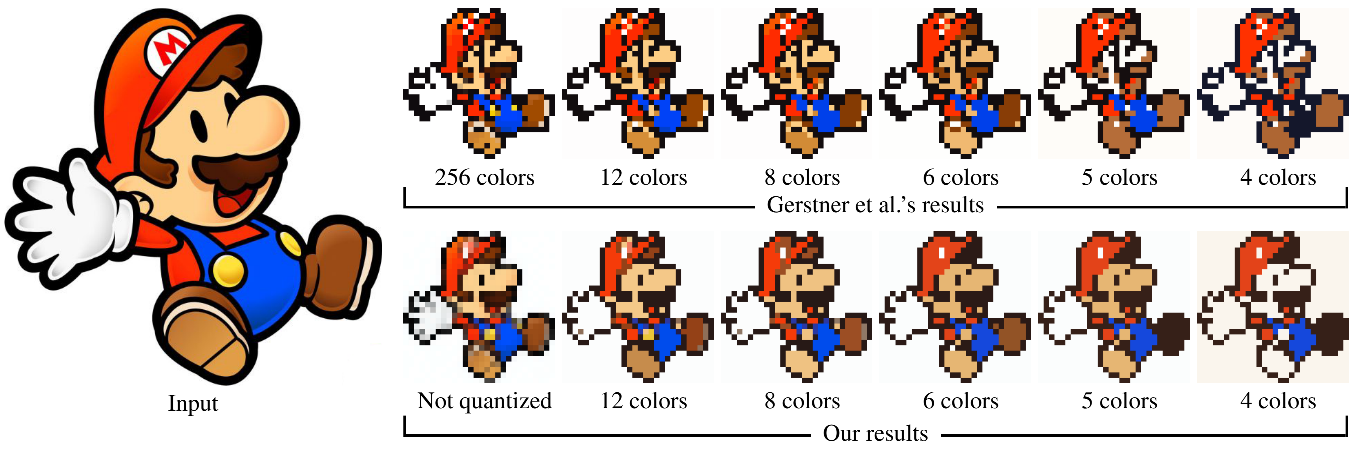


Figure 8: *Our method can also be extended to create pixel art with a limited color palette from cartoon and vector inputs. Color quantization is achieved in a post-process using mean shift segmentation. (Input image © Nintendo Co., Ltd.)*

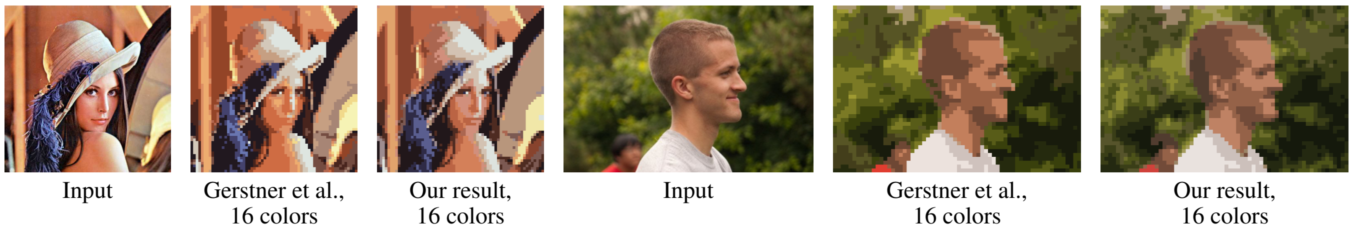


Figure 9: *Post-process color quantization using k-means clustering combined with our content-aware downsampling methods applied to natural images yields similar results to prior work.*

d_{kn} deviates by more than 25 degrees from the orientation of the pixel edge, we consider this a false edge and increase the smoothness of both kernels involved. The effect of this correction is illustrated in Figure 7c.

5 Results

We tested our algorithm on a wide range of input images ranging from natural images to line and vector art. All results were created with the same algorithm settings. Figure 13 and other figures throughout the paper show representative results of our algorithm. We compare our method to naïve subsampling and the bicubic filter (arguably the most commonly used rescaling algorithms). Our results show that our method yields sharper results (for instance on text), maintains details better (e.g., on the lunar surface or the flower in Figure 13), and preserves the appearance of high frequency textures (e.g., Figure 1-left). In the supplementary material we provide an extensive comparison on a large set of images, and compare our method to a wider range of downscaling methods (including a range of linear filters, unoptimized bilateral kernels, Generalized Sampling [Nehab and Hoppe 2011], and Pixelated Image Abstraction [Gerstner et al. 2012]).

5.1 Pixel Art Downscaling and Palette Reduction

Downscaling: Our algorithm is particularly well suited for downscaling cartoon and vector art images to create pixel art. Figures 1, 8, and 15 show representative results. When generating these results we disabled the edge orientation constraint (Section 4.2), since we are aiming for a blocky "old school" look (i.e., big pixels).

Of particular note is our algorithm's ability to keep line features connected. In comparison, many lines in the subsampled results are interrupted, while the bicubic results exhibits washed out colors due to excessive smoothing. Our algorithm strikes a balance between both extremes: it keeps outlines sharp and connected where possible, while in too detailed areas it naturally resolves to averaging out features.

Extension for Palette Reduction: One particular form of pixel art also includes a reduced color palette [Gerstner et al. 2012]. While not the focus of our method, we were interested in investigating the effectiveness of applying color palette reduction in a post-process. We use mean shift segmentation, following the description of Comaniciu and Meer's paper [2002] (using the Epanechnikov kernel, and fixed spatial bandwidth $h_s = 4$), and use the color bandwidth parameter h_r to adjust the number of colors in the output image.

We found that this works particularly well on cartoon and vector art inputs as shown in Figure 8. On these type of images, our method produces higher quality results than Gerstner et al. [2012]. However, this is mainly due to our algorithm's ability to keep line features connected.

When applied to natural images, however, we found mean shift segmentation to not work well. Instead, we use simple k-means clustering for natural images, where it produces images of similar quality as Gerstner et al. [2012] (Figure 9).

5.2 User Study

To verify our algorithm we conducted a formal user study with 51 subjects using Amazon Mechanical Turk, in which we compare our algorithm against five alternatives: (1) Generalized Sampling [Nehab and Hoppe 2011], which we consider the state-of-the-art algorithm for image scaling, (2) bicubic, since it is one of the most commonly used scaling algorithms, (3) subsampling, for its simplicity, (4) box filtering, because it yields sharper results than bicubic, and (5) unoptimized bilateral kernels, to verify the effectiveness of our optimization.

In each test we showed the participant the "high resolution" (400 pixels on the long side) input image as well as two downscaled re-

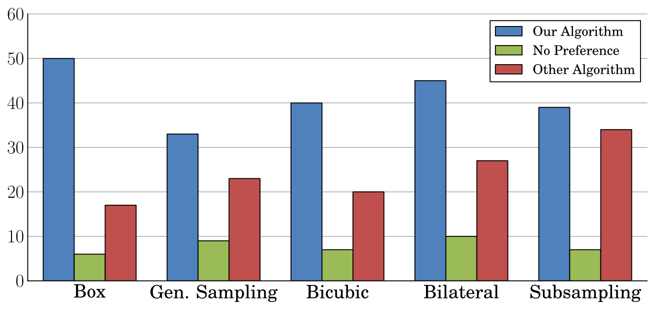


Figure 10: *Results of the user study comparing our algorithm against several existing algorithms.*

sults (128 pixels), one produced by our algorithm, and the other produced by one of the competing algorithms. Participants were asked which result "represents a better downscaled version of the input image", and had to choose either one of the results or express "no preference". No time limit was imposed. All images were shown at native display resolution and participants were not provided with any means to zoom into the images. We did not, however, control the user distance from the screen to better simulate realistic application conditions, i.e., subjects could move closer to the screen to examine details.

Each participant was presented 13 tests in total, each testing our algorithm against a random competing algorithm on a different input image, i.e., no participant saw the same input image more than once. We repeated every question throughout the test to filter unreliable participants by removing all answers from participants who were consistent on less than 80% of the tests. For the study we selected a variety of natural images from the MSRA Salient Object Database [Liu et al. 2007] that span different categories, including people, stochastic and regular textures, text, and smooth areas. The images used for the study are provided in the supplementary material. Results are shown in Figure 10. A χ^2-analysis between each condition indicates that our algorithm was significantly preferred over each of the competing techniques.

5.3 Performance

Our method employs an iterative optimization strategy to downscale images, consequently, it is computationally more demanding than classical linear rescaling filters. In the following we analyze the performance of our C++ implementation, running on a Intel Xeon E5640 CPU at 2.66 GHz. We *partially* use multiple cores in our implementation, but we have not fully parallelized or optimized the implementation.

The convergence proofs of the original EM-Algorithm do not carry through onto our algorithm due to our modifications. However, we did not encounter convergence issues on several thousand images tested—if this would happen one could simply terminate the algorithm after a fixed number of iterations.

A single iteration of our algorithm is linear both in the input and output image sizes. Due to the content dependent nature of our algorithm, the number of iterations varies for different in-/output images of the same size. Figure 11 reports the runtime (blue) and number of iterations (red) averaged over processing 100 randomly selected natural images. The shaded region indicates the standard deviation and the dashed lines indicate the min–max ranges. In Figure 11-left, the output size is kept fixed at 80×60 pixels, while the *input size* varies from 160×120 to 640×480. In Figure 11-right, the *output size* varies from 40×30 to 160×120 while the input size remains fixed at 640×480 pixels.

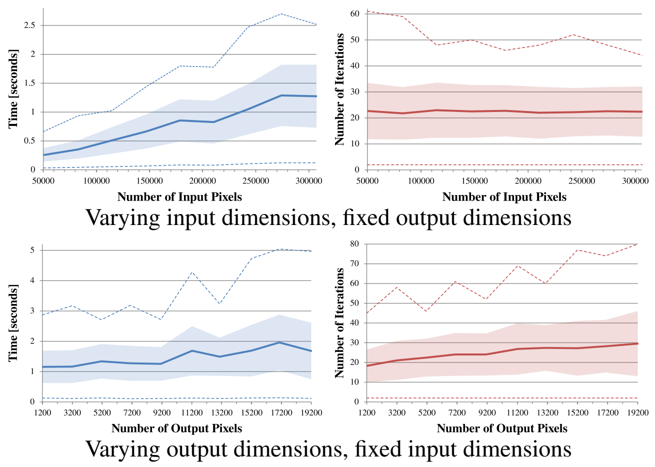


Figure 11: *Average runtime (blue) and number of iterations (red) from processing 100 random natural images. The shaded region indicates the standard deviation, and the dashed lines indicate the min-max range. The left/right figures show the result of varying the input/output image dimensions while keeping the other fixed.*

5.4 Limitations

Our algorithm relies on a number of heuristic constraints to prevent certain downscaling artifacts (Section 4). It would be desirable to incorporate these constraints directly into the EM optimization, however, most of these constraints are of a fundamentally different nature. The EM steps process each kernel independently. The constraints, on the other hand, rely on the relation between neighboring kernels, and hence, cannot be directly solved in the E or M step. Therefore, these constraints are handled in an additional third step.

Due to the content-adaptive nature our algorithm behaves temporally less coherent than linear filters when applied to smooth animations, e.g., a slow zoom into a picture. Our results are flickering slightly, while each individual image appears crisper and exhibits more detail. Please refer to the supplementary material for videos illustrating this issue. A similar problem can occur for symmetric features in input images. For example, our algorithm fails to preserve the symmetry of the yellow buttons in Figure 12.

Our algorithm does not prevent aliasing under all circumstances. Consequently, our method does not perform well on most standard aliasing tests, e.g., the zone plate pattern in Figure 12. Furthermore, our results cannot reach the quality that well-trained experts achieve when manually hinting fonts and manually creating pixel art (Figure 12, bottom).

While our method significantly improves the quality of downscaled images exhibiting small details such as eyes or stochastic textures, it does not always produce better results on images with blurred features, or images that contain structured textures. In the latter case, despite our efforts (Section 4), staircasing can still occur. This artifact shows up in particular on long, *almost* cardinal lines, e.g., the right edge of the sign in the top row of Figure 13. A systematic investigation of this artifact can be found in the supplementary material and accompanying web site.

Lastly, since Equation (5) may have multiple local minima, we may reach *slightly* different solutions depending on our initialization. In the supplementary material and accompanying web site we show that various sensible initialization choices yield similar solutions. We settled on using a "middle gray" initialization, which worked well in our tests.

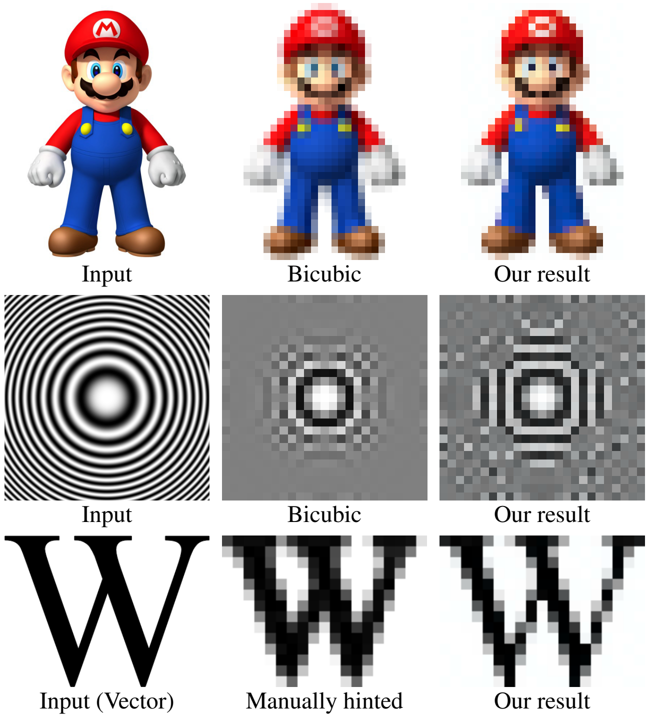


Figure 12: ***Limitations of our method.*** Top: *our algorithm fails to preserve the symmetric arrangement of the yellow buttons on Mario's overall.* Middle: *Our method was not specifically designed to prevent aliasing under all circumstances.* Bottom: *Our method cannot compete with manually downscaled images. ("Mario" input image © Nintendo Co., Ltd.)*

6 Conclusions

We have presented a novel content-adaptive image downscaling method that adapts the shape of its downsampling kernel, yielding sharper and more detailed downscaled results. Contrary to common wisdom that dictates that frequencies above the Nyquist frequency introduce artifacts in the downsampled image (in the form of aliasing), we show that by careful sampling, certain high frequencies features can still be preserved in the downscaled image without artifacts.

Given the growing "resolution gap" between cameras and display devices and the advent of gigapixel panoramic imaging, we believe that this work opens up an exciting area of research. There are plentiful avenues for future research. Our work has shown that it is possible to sometimes *drastically* improve quality over existing downscaling methods. For future work we would like to further improve the robustness of the method, e.g., through smarter heuristics, so that our method *always* outperforms simpler filters. It would also be interesting to look at other signals than images as inputs. A natural immediate step would be to analyze and constrain the temporal behavior of our algorithm, e.g., when applying it to videos.

Acknowledgements

This work was in part supported by NSF IIS-1217765 and by the Israeli Science Foundation (grant no. 324/11).

References

ACHANTA, R., SHAJI, A., SMITH, K., LUCCHI, A., FUA, P., AND SÜSSTRUNK, S. 2012. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE Trans. Pattern Anal. Mach. Intell. 34*, 11, 2274–2282.

AVIDAN, S., AND SHAMIR, A. Seam carving for content-aware image resizing. *ACM Transactions on Graphics, (Proc. SIGGRAPH 2007) 26*, 3, article no. 10.

COMANICIU, D., MEER, P., AND MEMBER, S. 2002. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence 24*, 603–619.

GERSTNER, T., DECARLO, D., ALEXA, M., FINKELSTEIN, A., GINGOLD, Y., AND NEALEN, A. 2012. Pixelated image abstraction. *Proceedings of the International Symposium on Non-Photorealistic Animation and Rendering (NPAR)*, 29–36.

HASTIE, T., TIBSHIRANI, R., FRIEDMAN, J., AND FRANKLIN, J. 2005. The elements of statistical learning: data mining, inference and prediction. *The Mathematical Intelligencer 27*, 2, 83–85.

INGLIS, T. C., AND KAPLAN, C. S. 2012. Pixelating vector line art. *Proceedings of the Symposium on Non-Photorealistic Animation and Rendering*, 21–28.

KARNI, Z., FREEDMAN, D., AND GOTSMAN, C. 2009. Energy-based image deformation. *Proceedings of the Symposium on Geometry Processing (SGP 2009)*, 1257–1268.

LIU, T., SUN, J., ZHENG, N.-N., TANG, X., AND SHUM, H.-Y. 2007. Learning to detect a salient object. *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2007)*, 1–8.

MANSON, J., AND SCHAEFER, S. 2012. Parameterization-aware mip-mapping. *Computer Graphics Forum (Proc. Eurographics Symposium on Rendering) 31*, 4, 1455–1463.

NEHAB, D., AND HOPPE, H. 2011. Generalized sampling for computer graphics. Tech. rep., feb.

RUBINSTEIN, M., SHAMIR, A., AND AVIDAN, S. 2009. Multi-operator media retargeting. *ACM Transactions on Graphics (Proceedings SIGGRAPH 2009) 28*, 3, 1–11.

SAMADANI, R., LIM, S. H., AND TRETTER, D. 2007. Representative image thumbnails for good browsing. *Proceedings of the International Conference on Image Processing (ICIP 2007)*, 193–196.

SHANNON, C. E. 1949. Communication in the presence of noise. *Proceedings of the Institute of Radio Engineers 37*, 1, 10–21.

SUH, B., LING, H., BEDERSON, B. B., AND JACOBS, D. W. 2003. Automatic thumbnail cropping and its effectiveness. *Proceedings of the 16th annual ACM symposium on User interface software and technology*, 95–104.

TOMASI, C., AND MANDUCHI, R. 1998. Bilateral filtering for gray and color images. *Proceedings of IEEE International Conference on Computer Vision (ICCV '98)*, 836–846.

TRENTACOSTE, M., MANTIUK, R., AND HEIDRICH, W. 2011. Blur-aware image downsizing. *Computer Graphics Forum (Proc. Eurographics 2011) 30*, 2, 573–582.

TRIGGS, B. 2001. Empirical filter estimation for subpixel interpolation and matching. *Proceedings of IEEE International Conference on Computer Vision (ICCV 2001) 2*, 550–557.

WOLBERG, G. 1990. *Digital Image Warping*. IEEE Computer Society Press, Los Alamitos, CA, USA.

WOLF, L., GUTTMANN, M., AND COHEN-OR, D. 2007. Non-homogeneous content-driven video-retargeting. *Proceedings of IEEE International Conference on Computer Vision (ICCV 2007)*, 1–6.

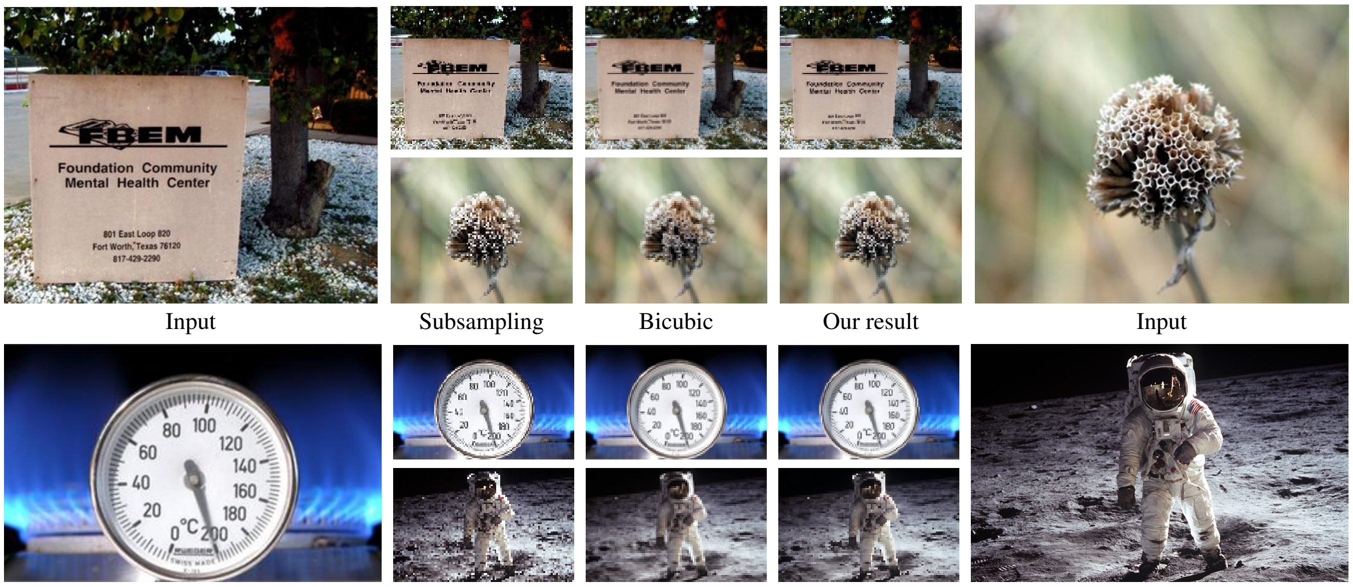


Figure 13: *A comparison of natural images downscaled using subsampling, bicubic, and our algorithm. To fully appreciate the quality difference of the downsampled results, we recommend viewing the results in native resolution when viewed electronically. An extended set of results can be found in the supplementary material.*

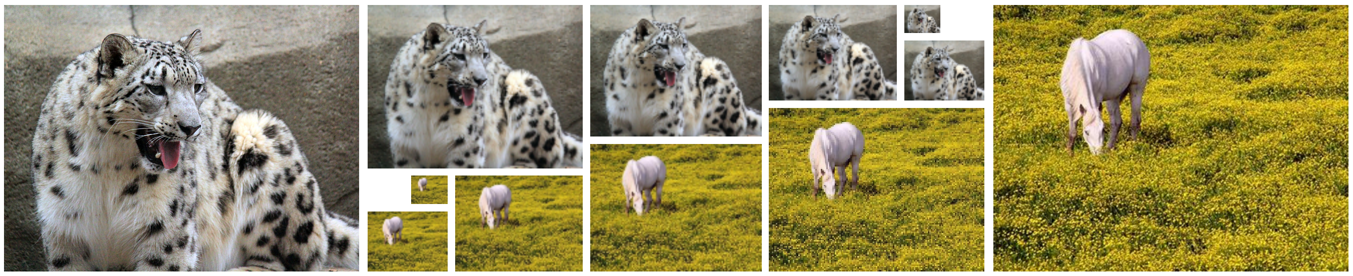


Figure 14: *A selection of downscaled results at various output resolutions ranging from* 32×24 *to* 192×144 *(original resolution:* 708×531 *and* 384×288 *for the "Snow Leopard" and "Flowers" respectively). Our method produces good results at any scale. In the supplementary material we compare against the bicubic filter on slowly zooming videos.*

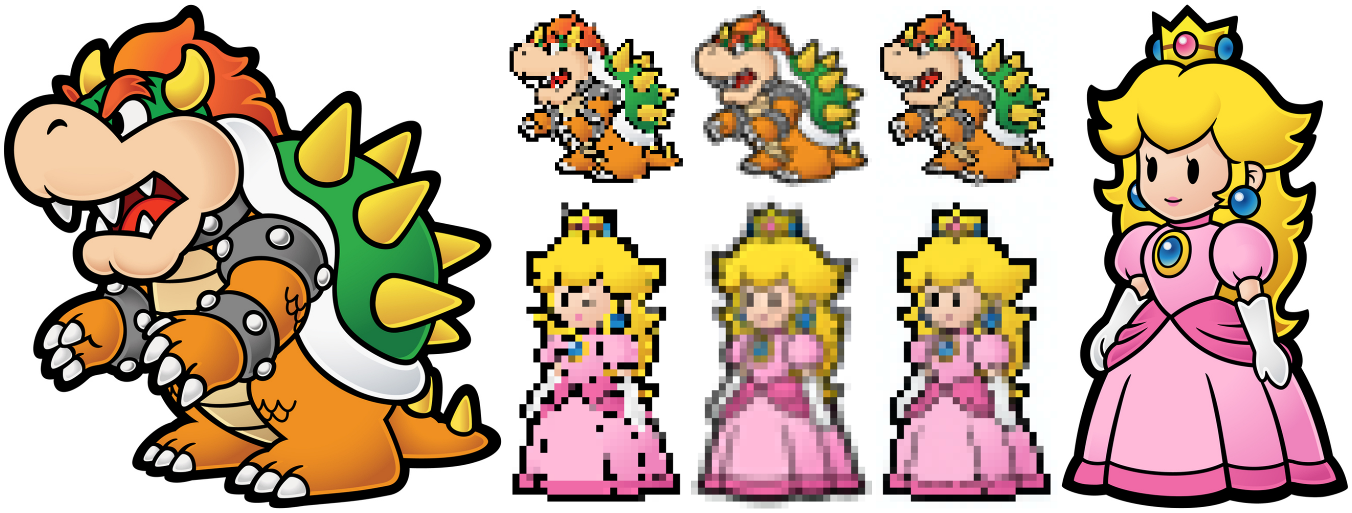


Figure 15: *Our method can be used to create pixel art from drawn inputs. Note the ability of our algorithm to keep features connected (e.g., the outline). Furthermore, the "staircase" correction step was disabled to achieve the typical quantized look of pixel art. (Input images © Nintendo Co., Ltd.)*