Content-Adaptive Image Downscaling

HARSHIT MAHAJAN 201501231

AYUSH RAI 201501119

AKSHAY VYAS 201530032

TA MENTOR:

JOYNEEL MISRA

Abstract

The key idea of Content Adaptive image Downscaling 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 pre- serving 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**.

Introduction

Downscaling is most commonly used image operation today. Whether you view photo in camera viewfinder, on a computer or mobile screen, or on the web all have Photos have been reduced from Megapixels to much smaller dimensions.

EARLIER METHODS

Linear filters are standard for image downscaling where 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** . **Sharpening** these images or using kernels that more closely model a sinc filter (eq Lanczos) can **cause ringing**. while simply **subsampling** the image without prefiltering typically leads to strong **aliasing artifacts** . Because all these methods are Content-Invariant (i.e., they use invariant kernels), the tradeoff between preserving detail and preventing aliasing is global. In Earlier methods, all kernels have the same shape and are arranged in a regular grid 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.

CONTENT ADAPTIVE METHOD

As in Earlier methods, the output pixels are computed as a weighted sum of the input pixels interpreted as associating an averaging kernel to every output pixel. We **adapt** the shape of these kernels in order to better align them with local image features . Following previous techniques such as bilateral filtering and mean shift 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 a small number of parameters to optimize, and sufficient flexibility to take on the shape of image features.

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. This property is of particular importance when downscaling images that contain fine lines, like cartoon art.
Our local kernel parameters are derived through an iterative max- imum 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.

PROBLEM STATEMENT :

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.

USES

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.

Related Work

Classical image downscaling techniques find their origin in sam-pling theory [Shannon 1949], and prefilter and reconstruct the sig-nal with a spatially constant lowpass filter in order to prevent alias-ing in the reconstructed signal. However, by suppressing high fre- quencies they also tend to blur the signal. Filters that are designedto 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 fil- ters (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 downsam- pled result. Recent developments [Nehab and Hoppe 2011] add a correction stage on the discrete signal before reconstruction, result- ing 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 lo- cation 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.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 ap- pearance in thumbnails by reintroducing artifacts such as blurring and noise lost in downscaling. Trentacoste et al. [2011] also rein- troduce 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 repre- sentation with a reduced color palette. Their algorithm alternates between updating SLIC (Simple Linear Iterative Clustering) super- pixels [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. Reconst. Kernels Result

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 downsam- ple image content based on application specific rules.

Method

To find the local filtering kernels, we formulate our task as a re- construction problem of the input image from a smaller set of local kernel functions w_k (probability density functions) with fixed colors 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.

- The number of kernels must be equal to the number of output pixels, hence, no kernel can vanish during optimization.
- Their position cannot vary too much from the output pixel grid and their size cannot vary too much from the output pixel size.
- To prevent aliasing artifacts, we add orientation constraints too.

Denote $X = \{x_i\}$ as the input image with pixels $x_i = (p_i, c_i)$, where p_i are the spatial coordinates and c_i the corresponding *CIELAB* color of pixel *i*. Each reconstructed image pixel color c_i' is defined as a weighted sum of the kernel colors v_k .

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

$$\gamma_k(i) = \frac{w_k(i)}{\sum_n w_n(i)}$$

is the (unknown) weight of kernel *k* at pixel *i* relative to all kernels overlapping pixel *i*. The (also unknown) kernels w_k : domain(X) $\rightarrow [0,1]$ are bilateral Gaussian kernels our case, i.e., we denote the value of kernel *k* at pixel *i* as:

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

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

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

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

$\boldsymbol{\mu}_k$ and Σ_k are the mean and covariance matrix of the spatial Gaussian, and 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 $\theta = \{\boldsymbol{\mu}_k, \Sigma_k, \mathbf{v}_k, \sigma_k\}$ of the kernels, and set of unknown variables g*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:

$$\underset{\theta}{\operatorname{argmax}} \ \Pr(X \mid \theta).$$

This problem is well known in statistics and solved using the Expectation-Maximization (EM) algorithm where the unknown variables g*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:

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

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

Where (xk,yk) is the center of output pixel k scaled to input image dimensions and rx and ry are the ratio 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:

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

In other words, g (i) quantifies how much an input pixel i con- ktributes 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 θ: parameters

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

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

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

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

We add a third **correction** step after every maximization step. In this step we enforce the different constraints specific to our down- sampling problem. 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 Above Equations are defined over the whole input image, this is not necessary in a practical implementation, since the kernels are constrained in size.

Constraints

Downscaling is a more restricted problem than general signal re- construction, and thus finding the optimal kernel shape and weight that minimizes the reconstruction error is not a sufficient condi- tion 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
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
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.

Spatial Constraints:
To retain the characteristics of the grid topology of the output pixels, we bias the spatial mean µk towards a smooth grid-like topology.
First, we limit the extent the spatial mean can move by constraining µk to lie within a box, centered around the center of the output pixel.

Second, we increase smoothness by moving µk halfway between its estimated location and the the mean of its four neighbors.

Formally, we update

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

$\overline{\mu^k} = (\Sigma_{n \in N_k^4} \mu_n) / |N_k^4|$, and N_k^4 denotes the set of cardinal (4-connected neighbors of k.

Locality and Edge Orientations:
We constrain the shape of the spatial variance Σ_k to avoid vanish- ingly small or exceedingly large kernels. We first obtain the sin- gular value decomposition $(U, S, V^\star) = \text{SVD}(\Sigma_k)$,
and modify the diagonal eigenvalue matrix S by clamping its elements to the inter- val [0.05,0.1], and finally set

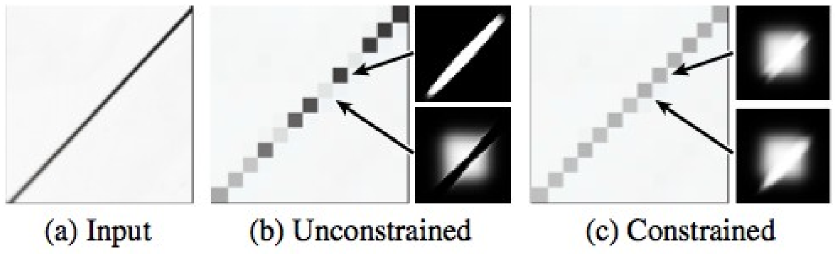
$$\Sigma_k \leftarrow US'V^\star,$$

where S' contains the clamped eigenvalues.While above Equation 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** sigma(k), which is not estimated, to directly control the amount of adaptation, and, hence, the sharpness of our results. Small sigma(k) cause kernels to be more sensitive to color variations and have sharper transitions, while larger sigma(k) lead to smoother kernels.

The reason for not freely estimating sigma(k) is that the **maximum likelihood estimation** often yields too **smooth configurations**. Rather than estimating sigma(k) from the data, we control it explicitly to locally adjust the sharpness of the result. For most kernels we let sigma(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.
(2) To prevent staircasing artifacts. Following each maximization step we search for kernels that match one of these conditions, and correct them by increasing sigma(k) by 10% (i.e., increase the smoothness of the color kernel).

Σ_k

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



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.*

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:

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

where $d \in \{(a,b)^\top \mid a,b \in [-1,1]\} \setminus \{(0,0)^\top\}$ is the offset to one of neighbors. If any directional variance s_k^d exceed a thresh- old of 0.2rx (where rx 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 (c) . The ability of our algorithm to keep linear features connected while maintaining sharpness can also be nicely observed in the pixel art results Shown in results section below.

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 shown below.

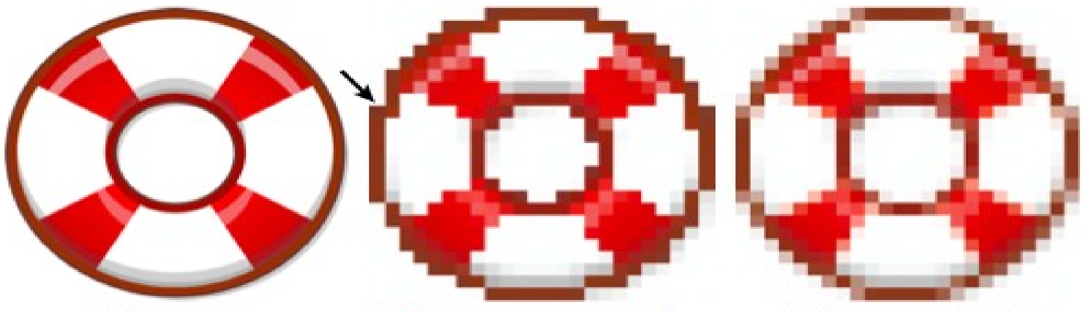

(a) Input (b) Unconstrained (c) Constrained

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.*

The corresponding image edge in the input image is almost vertical. We selectively remove such false edges by increasing sigma(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

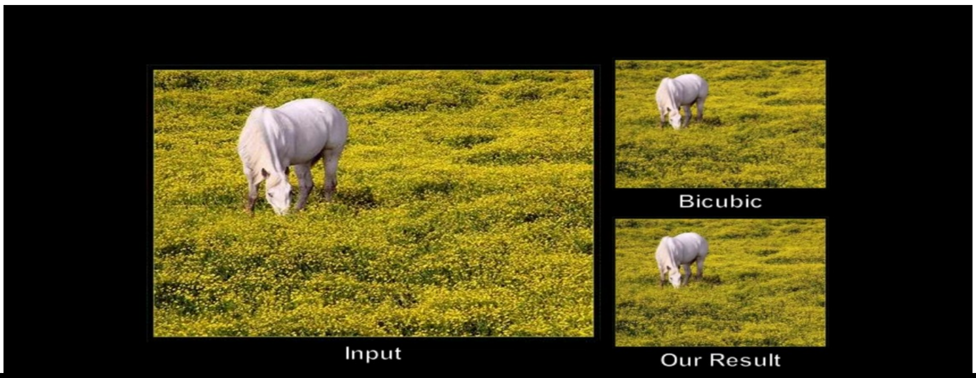
$f_{kn} = \sum_i \gamma_k(i)\gamma_n(i)$. If the transition between the two kernels is abrupt, there will be few pixels where *both* kernels take on large values, and, thus, f_{kn} will be small. We consider edges where $f_{kn} < 0.08 r_x r_y$ as strong. Next, we compute the direction of the edge between the kernels as $d_{kn} = \sum_i \nabla \left(\gamma_k(i)/(\gamma_k(i) + \gamma_n(i))\right)$. If

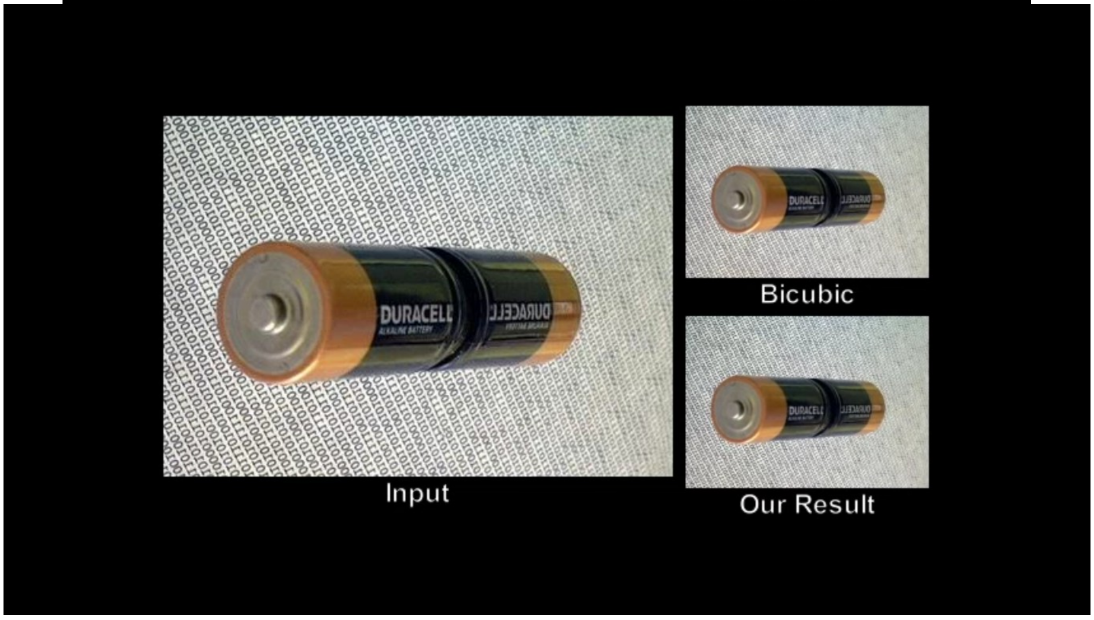
dkn deviates by more than 25 degrees from the orientation of the pixel edge, we consider this a false edge and increase the smooth- ness of both kernels involved. The effect of this correction is illus- trated in Figure 7c.

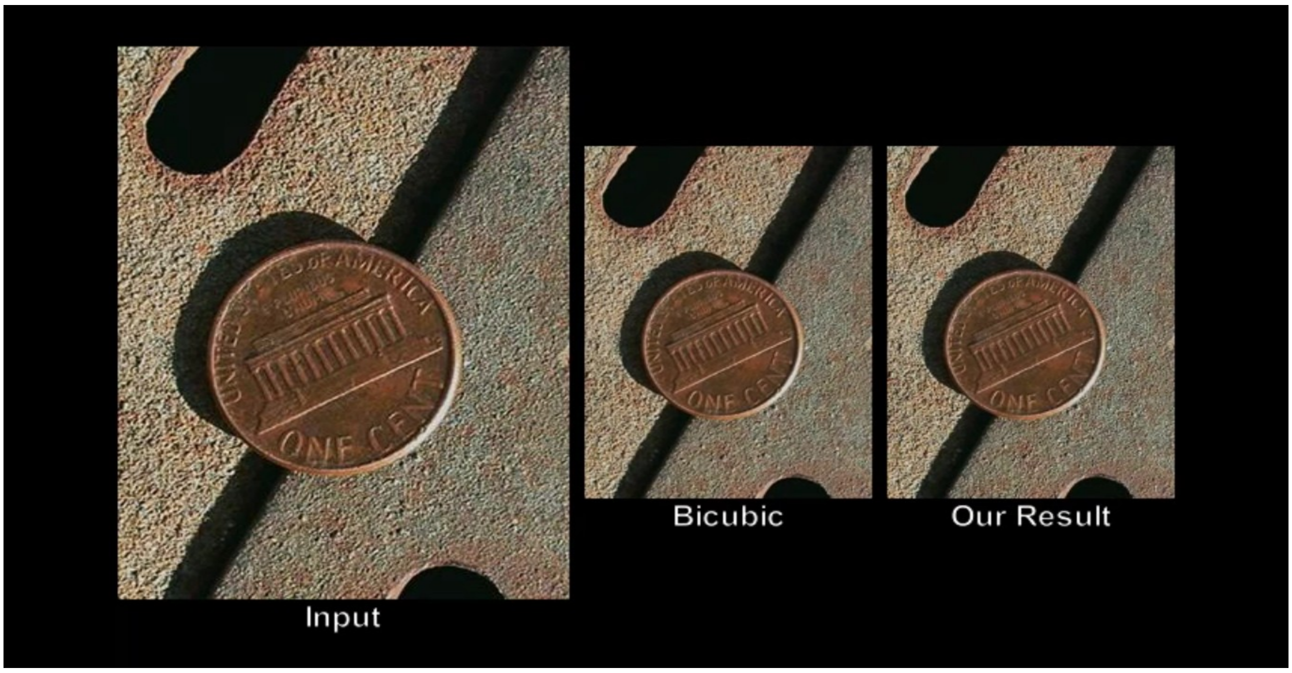
Results

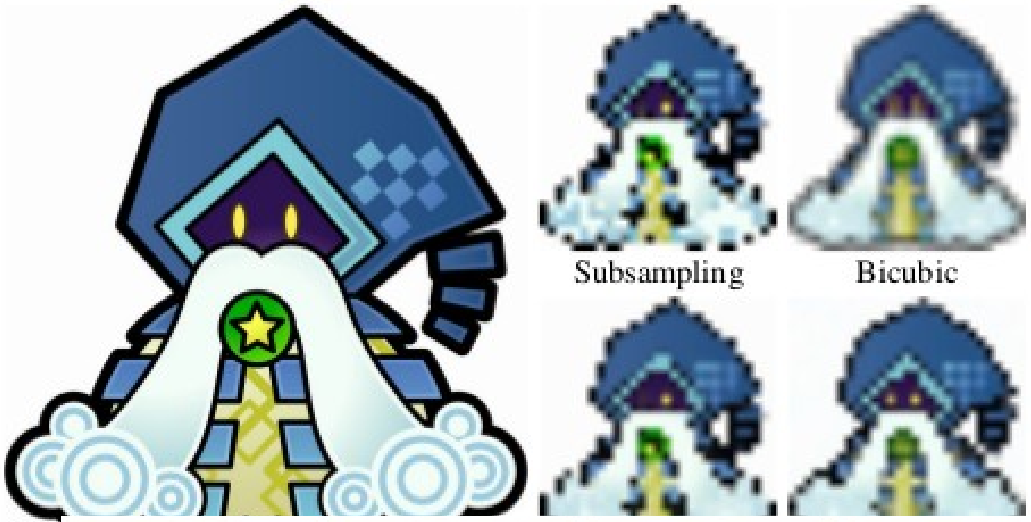
We tested our algorithm on a wide range of input images ranging from natural images to line and vector art. All results were created with the same algorithm settings. We compare our method to na¨ive subsampling and the bicubic filter (arguably the most commonly used rescaling algorithms). Our results show that our method yields sharper results (for instance on text), maintains details better, and preserves the appearance of high frequency textures.

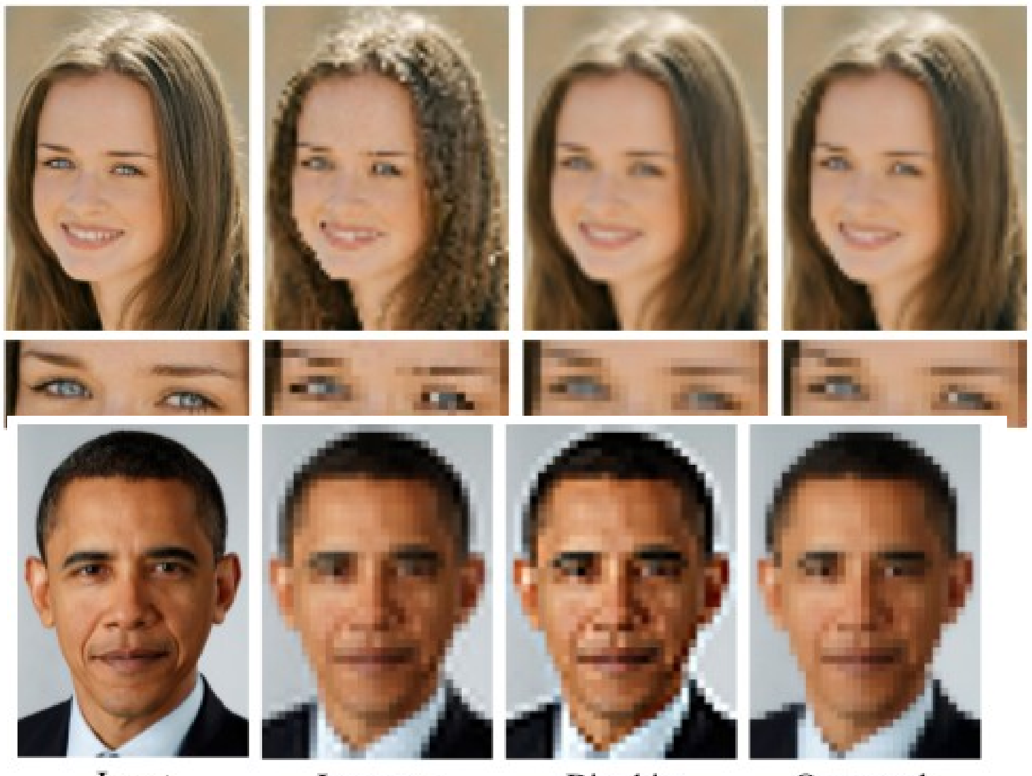
We provide an extensive comparison on a large set of images, and compare our method to a wider range of downscaling methods (including a range of linear filters, unoptimized bilateral kernels, too).

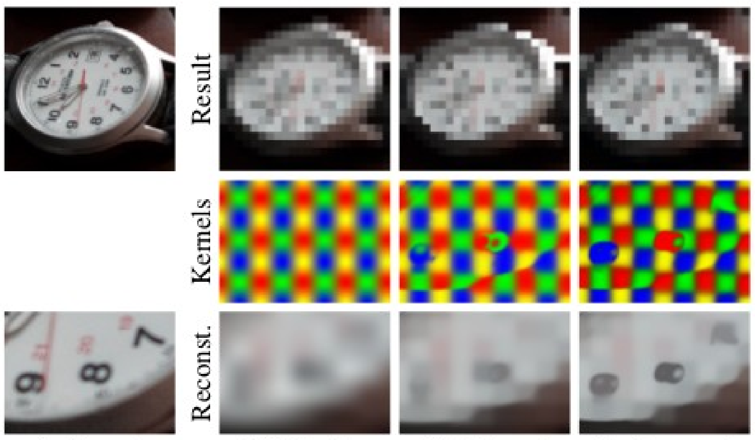




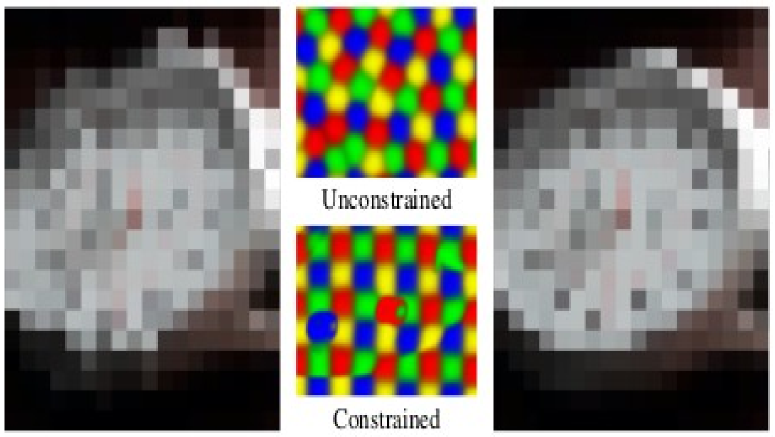

Bicubic
Our Result
Input


Subsampling
Bicubic

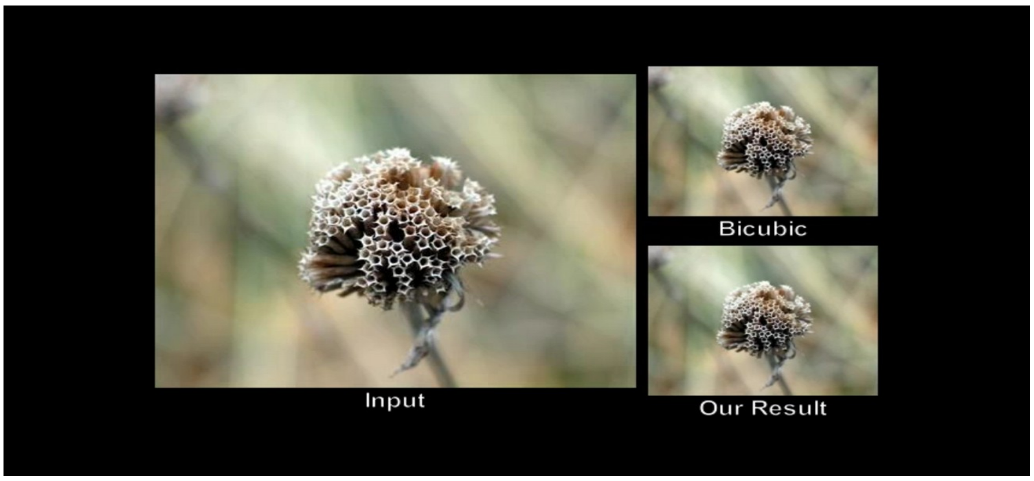

Input
Lanczos
Bicubic sharpened
Our result



(a) Input (b) Bicubic (c) Bilateral (d) Our result



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



Limitations

Our algorithm relies on a number of heuristic constraints to prevent certain downscaling artifacts. 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. A similar problem can occur for symmetric features in input images. For example, our algorithm fails to preserve the symmetry in some images.

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.

Further- more, our results **cannot reach the quality** that well-trained experts achieve when manually hinting fonts and manually creating pixel art .

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 fea- tures, or images that contain structured textures. In the latter case, despite our efforts , 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 shown at last.Lastly, since eqaution may have multiple local minima, we may reach slightly different solutions depending on our initialization .We show that various sensible initialization choices yield similar solutions. We settled on using a “middle gray” initialization, which worked well in our tests.

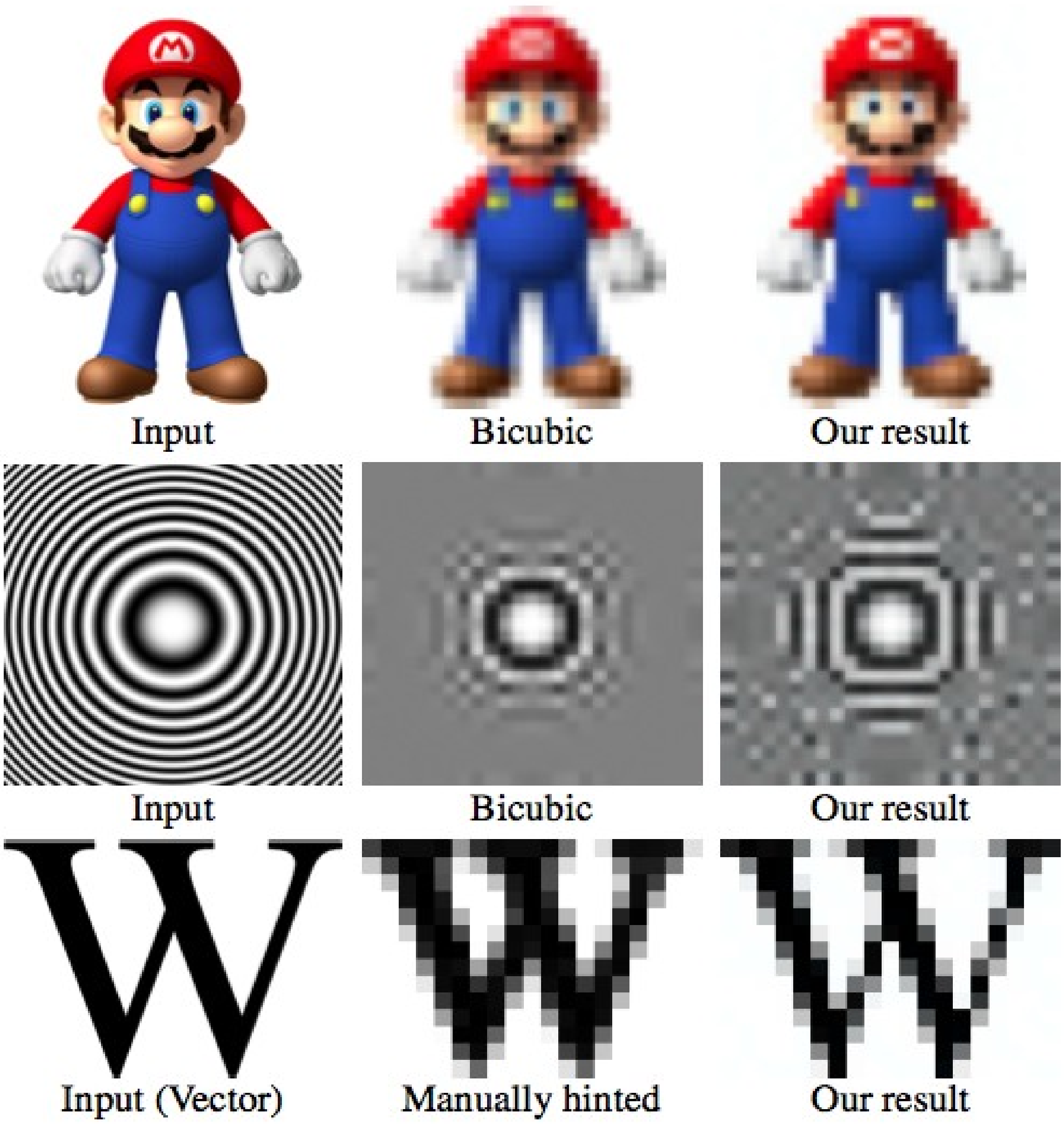


Figure 12: ***Limitations of our method.*** Top: *our algorithm fails to preserve the symmetric arrangement of the yellow buttons on Mario's overall.* Middle: *Our method was not specifically designed to prevent aliasing under all circumstances.* Bottom: *Our method cannot compete with manually downscaled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

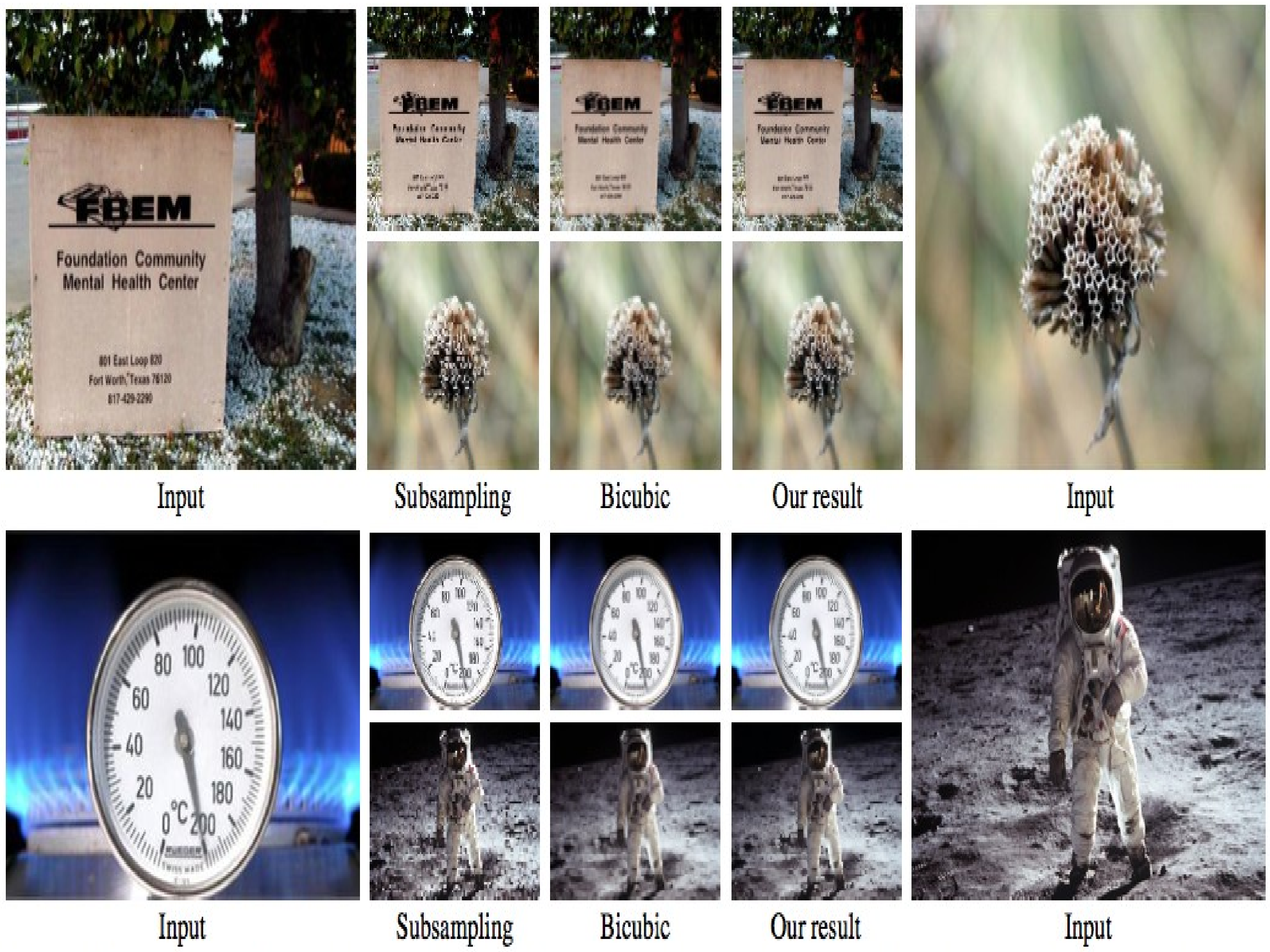


Figure 13: *A comparison of natural images downscaled using subsampling, bicubic, and our algorithm. To fully appreciate the quality difference of the downsampled results, we recommend viewing the results in native resolution when viewed electronically. An extended set of results can be found in the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

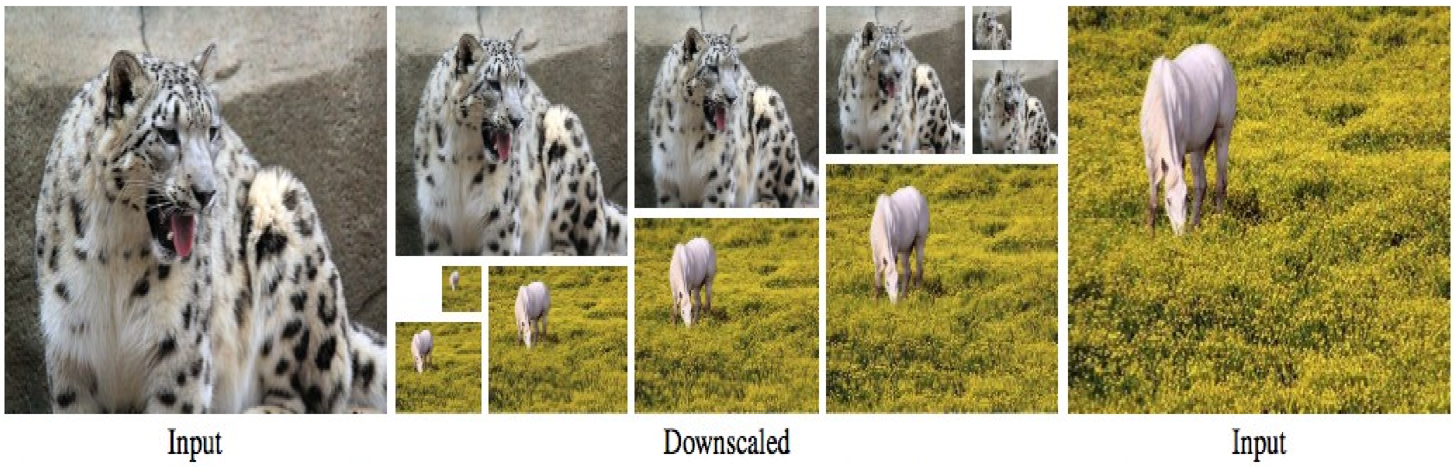


Figure 14: *A selection of downscaled results at various output resolutions ranging from* 32×24 *to* 192×144 *(original resolution:* 708×531 *and* 384×288 *for the "Snow Leopard" and "Flowers" respectively). Our method produces good results at any scale. In the supplementary material we compare against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