EUROGRAPHICS 2016 / J. Jorge and M. Lin
(Guest Editors)

Volume 35 (2016), Number 2

Regularizing Image Reconstruction for Gradient-Domain Rendering with Feature Patches

M. Manzi, D. Vicini, and M. Zwicker

University of Bern, Switzerland

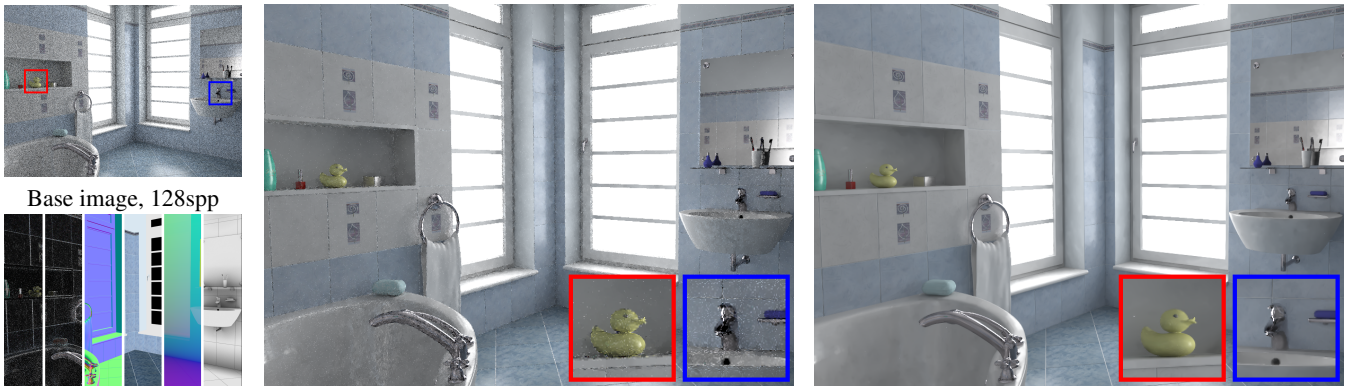

Figure 1: *Our regularized reconstruction for gradient-domain rendering obtains a high-quality image from a noisy base image, the sampled gradients, and auxiliary features (left). We (right) achieve significantly better images than standard L_1 reconstruction (middle). The depicted features are, from left to right, the vertical and horizontal gradients, normals, texture values, positions and ambient occlusion values.*

Abstract

We present a novel algorithm to reconstruct high-quality images from sampled pixels and gradients in gradient-domain rendering. Our approach extends screened Poisson reconstruction by adding additional regularization constraints. Our key idea is to exploit local patches in feature images, which contain per-pixels normals, textures, position, etc., to formulate these constraints. We describe a GPU implementation of our approach that runs on the order of seconds on megapixel images. We demonstrate a significant improvement in image quality over screened Poisson reconstruction under the L_1 norm. Because we adapt the regularization constraints to the noise level in the input, our algorithm is consistent and converges to the ground truth.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—Display Algorithms

1. Introduction

With the ongoing path tracing revolution in the movie industry [KFF*15], there has been a renewed interest in noise reduction for Monte Carlo rendering. Monte Carlo algorithms are attractive because they are conceptually simple, general, and based on a physical model of light transport. Noise artifacts, however, have remained a challenge for real world applications. Since the level of noise is inversely proportional to the square root of the number of samples per pixel, it is often impractical to eliminate visual artifacts in a brute force manner.

In this paper we build on gradient-domain rendering techniques, which sample finite difference image gradients, in addition to the usual pixel values, and then reconstruct final images by solving a screened Poisson problem using the sampled gradients and pixels. Gradient-domain rendering was originally proposed for Metropolis light transport [LKL*13], but recently Kettunen et al. [KMA*15] and Manzi et al. [MKA*15] showed that it also significantly reduces the error compared to standard (bidirectional) path tracing at equal computation time. Visual artifacts, however, often remain even at hundreds of samples per pixel as shown in Figure 1. Even

if we minimize the L_1 error of the screened Poisson reconstruction, which leads to bias but noticeably better visual quality than unbiased L_2 reconstruction, outlier pixels frequently persist.

Here we propose a technique to reduce these artifacts and further increase image quality by leveraging auxiliary image space information such as per pixel normals, albedo, or world space position. Our main contribution is to exploit these auxiliary image features to extend and regularize screened Poisson reconstruction. The basic idea of screened Poisson reconstruction is to find an output image whose pixel values and finite difference gradients are similar to the corresponding noisy, sampled data that we acquired during gradient-domain Monte Carlo rendering. Intuitively, outlier pixels in the reconstructed output appear if both the sampled pixel value and its surrounding gradients contain outliers that happen to be in rough agreement. Our regularization avoids these outliers by adding constraints based on the features. These constraints encourage each small patch in the reconstructed output to be similar to a weighted average of the corresponding feature patches. As shown in Figure 1, our approach leads to much cleaner results with significantly lower numerical error.

We present an error analysis in terms of bias and variance that allows us to study the influence of the parameters of our approach on the output error, and choose robust parameters in practice. We also describe a GPU implementation of our extended reconstruction technique that reduces the overhead of our method to a few seconds on megapixel images. Our results show a significant visual and numerical improvement over standard screened Poisson reconstruction, and we show that our technique is consistent and converges to the ground truth solution with increased sample counts. In summary, we make the following contributions:

- An extended screened Poisson reconstruction approach for gradient-domain rendering that leverages feature patches to regularize the solution.
- An error analysis that reveals the influence of the main parameters of our method on the bias and variance in the output.
- An efficient GPU implementation that runs in a matter of seconds on megapixel images.

2. Related Work

We discuss previous work in gradient-domain rendering and image space denoising for Monte Carlo rendering.

Gradient-Domain Rendering. The core idea in gradient-domain rendering is to sample finite difference image gradients in addition to the usual pixel values. A gradient sample is simply the difference between the contribution of two light paths that go through the neighboring pixels. By generating the two paths in a correlated fashion such that they are as similar as possible, the magnitude of their differences tend to become much smaller than their individual contributions. This leads to less noise in the sampled gradients compared to the conventionally sampled pixels, and screened Poisson reconstruction yields output images with a higher quality at equal computation time compared to conventional rendering. Gradient-domain rendering was proposed in pioneering work by Lehtinen et al. [LKL*13] in the context of Metropolis light transport [VG97]. Manzi et al. [MRK*14] improved the original approach by proposing more general gradient sampling techniques. They exploit feature buffers to construct adaptive gradient kernels, while we use them to regularize screened Poisson reconstruction. These Metropolis methods adapt the target distribution that is sampled by the Markov chain to include gradients, and to focus more samples in regions with high gradients. Kettunen et al. [KMA*15] demonstrated that, maybe counterintuitively, adapting the sampling distribution is not necessary to benefit from gradient sampling. They describe a gradient-domain path tracer that simply obtains four additional gradient samples for each conventional path constructed by a standard path tracer. Their approach consistently outperforms conventional path tracing in a variety of scenarios at equal render time. They also present a Fourier analysis that explains the benefits of gradient sampling and reconstruction under some simplifying assumptions. Manzi et al. [MKA*15] subsequently describe a bidirectional gradient-domain path tracer with similar benefits over the conventional approach. While screened Poisson reconstruction under the L_2 norm leads to unbiased results with all these techniques [LKL*13], they typically advocate the use of the L_1 norm, which introduces bias but improves the visual quality. Yet even L_1 reconstruction suffers from artifacts as shown in Figure 1.

Image Space Denoising for Monte Carlo Rendering. Image space filtering for Monte Carlo has a long history, but only recently these techniques attracted renewed interest in the research community and found application in movie production [Ren]. Here we focus on the most relevant and recent work in image space denoising, and we refer to the work by Zwicker et al. [ZJL*15] for a more comprehensive survey. A key idea common to the most effective techniques to date is to use auxiliary per-pixel features, such as normals, albedo, world space position, etc., to construct the denoising filter. These features are effective because they are highly correlated with the output image, but they are usually much less noisy. Bauszat et al. [BEM11] were some of the first to exploit this idea for real-time rendering, building on the guided image filter [HST10]. Dammertz et al. [DSHL10] instead perform a fast wavelet transform that considers the feature information. Shirley et al. denoise motion and defocus blur [SAC*11] by leveraging the depth buffer.

A number of approaches targeting off-line rendering exploit the features by using them to define a cross-bilateral filter [ED04]. Sen and Darabi [SD12] propose an information theoretic approach to deal with noisy features. Li et al. [LWC12] introduced a per-pixel error estimate based on *Stein's unbiased risk estimator* (SURE) [Ste81] to select the best filter from a filter bank on a per-pixel basis. Moon et al. [MJL*13] apply a non-local means filter (NL-means) [BCM05] guided by a virtual flash image. Rousselle et al. [RMZ13] combine NL-means filter weights [RKZ12] for the noisy color image with a cross-bilateral filter on the features, and they also use SURE over several candidate filters to minimize the output error. Kalantari et al. [KBS15] employ machine learning to predict the parameters of a cross-bilateral filter at each pixel based on image features. Our approach has similarities to Moon et al.'s techniques based on local weighted regression [MCY14] and linear prediction [MIGYM15]. While Moon et al. compute local linear approximations separately, we use local linear models as regularization terms in a global energy minimization setup.

Sparse Reconstruction. Our approach is also inspired by image denoising using sparse representations [AEB06], where the idea is to express the desired output as a weighted sum of prototype signal-atoms selected from an overcomplete dictionary. Signal reconstruction is performed by finding its sparsest representation, that is, a linear combination of signal-atoms consisting of the fewest possible elements. Sparsity has been shown to effectively regularize many problems including image denoising, image superresolution, deconvolution, and many more. In our context, local patches taken from the feature images around each pixel seem to be a natural choice as the signal-atoms, because they contain most of the image structure. Then it is straightforward to extend the screened Poisson equation by expressing the image as a weighted sum of the atoms, and solve for the sparsest representation of the image that satisfies the constraints on the output pixel values and the gradients. In our experience, however, this approach did not lead to satisfactory results. Because our images are often contaminated by sparse outliers, imposing sparsity on the reconstruction from the dictionary is not effective at eliminating the rare outliers. In addition, sparse reconstruction of mega-pixel images tends to be expensive even with fast solvers [vdBF08]. Instead, we require that the reconstructed image, locally over each image patch, should be similar to a weighted sum of a small number of predetermined basis functions. These local constraints are overdetermined, since there are more pixels in each local patch than predetermined basis functions. We assemble the constraints in a global system that leads to an energy minimization problem that can be solved with the same simple methods as the original screened Poisson equation.

3. Background

The key idea in gradient-domain rendering is to sample finite difference gradients between horizontal and vertical neighbor pixels, in addition to pixel values. The final image is then reconstructed by solving a screened Poisson equation. Gradient-domain rendering is beneficial because sampled gradients typically contain less noise than sampled pixels. The noise level of high frequencies in the reconstructed image is then determined by the noise in the gradients, not the noise in the pixels [LKL*13, KMA*15]. While our results in this paper build on the gradient-domain extensions of (bidirectional) path tracing [KMA*15, MKA*15], our approach is generic and applicable to all existing gradient-domain rendering techniques [LKL*13, MRK*14].

Let us denote the conventional image sampled by a gradient-domain rendering algorithm, also called the base image, by I^g and the horizontal and vertical finite difference images by I^{dx} and I^{dy}. Screened Poisson reconstruction solves for an image I that is most consistent with both the sampled image I^g and the sampled gradients I^{dx} and I^{dy},

$$I = \arg\min_{\bar{I}} \left\|\alpha(\bar{I} - I^g)\right\| + \left\| \begin{pmatrix} H^{dx}\bar{I} \\ H^{dy}\bar{I} \end{pmatrix} - \begin{pmatrix} I^{dx} \\ I^{dy} \end{pmatrix} \right\|, \quad (1)$$

where α weights the relative influence of the pixel and gradient constraints, and H^{dx} and H^{dy} denote the horizontal and vertical finite difference operators. Solving this equation under the L_2 norm leads to unbiased reconstructions, but is susceptible to visual artifacts. In practice, the L_1 norm leads to more pleasing results, at the cost of introducing bias. Even with the L_1 norm, however, artifacts occur frequently if the input pixels and gradients are too noisy.

4. Regularized Reconstruction

The goal of our technique is to regularize screened Poisson reconstruction from Equation (1) to better suppress artifacts even at high noise levels in the input. The key idea is to leverage feature images that contain per pixel normals, albedo, position, etc. Intuitively, we add regularization constraints that push each local patch in the reconstruction to be similar to a weighted sum of corresponding patches in the feature images. Since the feature patches are largely free of noise and outliers, this leads to much cleaner outputs. We illustrate our feature patch constraints in Figure 2.

At each pixel p, let us denote its patch neighborhood by $\mathcal{N}_p$. The neighborhood consists of $s = (2r+1)^2$ pixels, where we call r the patch radius. Given the feature images, we first construct orthogonal bases for the local feature patches using truncated SVD, as described below. Assuming we have m feature bases, we unroll them and compile them into a matrix $B_p \in \mathbb{R}^{s\times m}$. In addition, the matrix $\Gamma_p \in \mathbb{R}^{s\times n}$ selects the s pixels in the local neighborhood $\mathcal{N}_p$ from the desired output image $\bar{I}$, which has n pixels. Our patch constraint says that the patch of the desired output image, $\Gamma_p\bar{I}$, should be as similar as possible to the patch itself projected onto the orthogonal basis of the feature patch (multiplication with B_p^T), followed by back-projection (multiplication with B_p). That is,

$$\left\| D_p \cdot (B_p B_p^T - \mathrm{Id}_s)\Gamma_p\bar{I} \right\| = \|P_p\bar{I}\|, \quad (2)$$

should be as small as possible. Here, Id_s is the $s\times s$ identity matrix, $D_p \in \mathrm{R}^{s\times s}$ is an additional weighting matrix, explained in detail below. We summarize the constraint using an $s\times n$ matrix P_p.

We can stack all constraint matrices $P_p, p = 1,\ldots,n$ into a matrix $M \in \mathbb{R}^{sn\times n}$, and add these constraints to the original screened Poisson problem to obtain

$$I = \arg\min_{\bar{I}} \left\|\alpha(\bar{I} - I^g)\right\| + \left\| \begin{pmatrix} H^{dx}\bar{I} \\ H^{dy}\bar{I} \end{pmatrix} - \begin{pmatrix} I^{dx} \\ I^{dy} \end{pmatrix} \right\| + \|\beta M\bar{I}\|, \quad (3)$$

where we introduced the scalar weight β to control the influence of the patch constraints. Under the L_2 norm, we obtain the minimum energy by solving the system of linear equations

$$\left(\beta^2 M^T M + H^T H + \alpha^2 \mathrm{Id}_n\right) I = \alpha I^g + H^{dx^T} I^{dx} + H^{dy^T} I^{dy} \quad (4)$$

for the desired output image I. For simplicity, we introduced the notation $H^T H = H^{dx^T} H^{dx} + H^{dy^T} H^{dy}$, which represents the discrete Laplacian opertor.

Feature Bases via Truncated SVD. Assume that we have l input feature channels, where each element of vector valued features (such as per pixel normals) is considered its own channel. At each pixel p, we can unroll the channels into a $s\times l$ matrix C_p. We use a singular value decomposition (SVD) of C_p to obtain our matrix B_p that contains the orthogonal feature basis vectors. The SVD factors the matrix $C_p \in \mathbb{R}^{s\times l}$ into a product of the form USV^T, where $U \in \mathbb{R}^{s\times s}$ and $V \in \mathbb{R}^{l\times l}$ are orthogonal matrices and $S \in \mathbb{R}^{s\times l}$ is

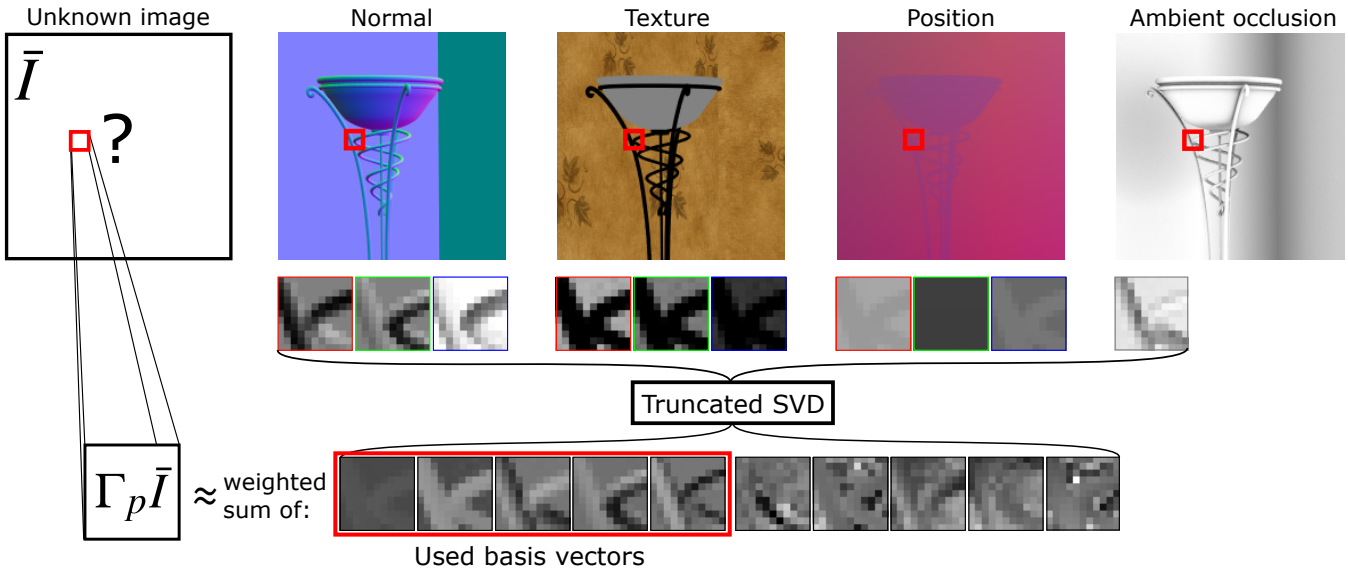

Figure 2: *Construction of the feature patch constraints. We try to reconstruct an image $\bar{I}$ where each patch $\Gamma_p \bar{I}$ is a weighted sum of the features in that patch. We orthogonalize the features by using SVD and truncate basis vectors of low singular values to avoid fitting to residual noise in the features. For better visualization we omit the weighting matrix D_p and show a larger patch size than used in practice.*

a rectangular diagonal matrix containing the singular values. The first s columns of U then form a basis of the range of C_p.

Although the features are less noisy than the sampled image and gradients, residual noise in the features will negatively impact the effectiveness of our regularization technique. Hence, we use a *truncated SVD* to remove noise from the feature subspace similar to Moon et al. [MCY14]. The vectors in U are ordered by the magnitude of the corresponding singular value. The smaller the singular value, the more noise is captured in the corresponding singular vector. This is demonstrated on an example in Figure 2. By discarding singular vectors with small singular values, one can remove noise from the subspace. We follow the approach by Moon et al. [MCY14] and discard basis vectors with a singular value that is below a pixel-wise threshold τ_p. The threshold is defined as $\tau_p = c\|E_p\|_2$, where $\|E_p\|_2$ is the spectral norm of E_p and c is a constant that we set to 0.1 in all our experiments. The entries in E_p are the square-roots of the pixel-wise variances of the features in the patch $\mathcal{N}_p$. We approximate these variances using a two-buffer approach, as described below, and approximate the spectral norm of E_p with the computationally cheaper Frobenius norm.

Per-Pixel Weights. The patch constraints can lead to inaccurate results if some pixels cannot be predicted well by the patch basis vectors. For example, a patch may contain pixels belonging to different geometric objects that may be lit differently, such that a single linear fit using the patch basis vectors cannot fit all pixels well. Each row of $(B_p B_p^T - I_s)\Gamma_p \bar{I}$ in Equation (2) measures the difference between a patch and its projection onto the feature subspace in one pixel for each color channel. The purpose of the diagonal weighting matrix D_p is to downweight the error of pixels that we consider too different to fit our model. We calculate the weights similarly to Rousselle et al. [RMZ13] by considering differences in features and color to the central pixel in the patch, and taking the minimum over all color and feature channels as our final weight.

We compute the color weights similar as in NL-means filtering [BCM05, RKZ12], that is, by averaging color differences over small neighborhoods of pixels. We derive the weights from the solution of the original Poisson problem (using the L_1 norm), because this is less noisy than the sampled pixels in I^g. We also normalize the color differences using their estimated variance, and we obtain the variance of the L_1 reconstruction using a two-buffer approach. During rendering we accumulate each half of the samples in two separate buffers, both for the base image and the gradients. We then solve the L_1 reconstruction twice, and we estimate the variance of each pixel based on the difference between the two reconstructed buffers. We blur the resulting variance with a Gaussian filter with standard deviation of two pixels, and take the maximum of the blurred two-buffer variance and the raw two-buffer variance. This reduces noise in the estimate, and it prevents the estimate from being too small when dealing with strong outliers. Concretely, the normalized color difference between two pixels p and q is

$$\Delta_i^2(p,q) = \frac{(u_i(p) - u_i(q))^2 - (\mathrm{Var}_i[p] + \mathrm{Var}_i[q])}{\varepsilon + k_c^2(\mathrm{Var}_i[p] + \mathrm{Var}_i[q])},$$

where p denotes the center pixel of the patch and q a neighboring pixel, $u_i, i \in \{1,2,3\}$ indexes a color channel, and $\varepsilon = 1e-10$ is a constant and k_c a user parameter. We next compute NL-means distances by averaging the color differences $\Delta_i^2(p,q)$ over small neighborhoods around p and q,

$$d_c^2(P(p),P(q)) = \frac{1}{3(2t_c+1)^2} \sum_{i=1}^{3} \sum_{n \in P(0)} \Delta_i^2(p+n, q+n),$$

where we sum over the color channels i of all pixels in the neigh-

borhood represented by a set $P(0)$ of pixel offsets. The size of the neighborhood is given by the radius t_c. Finally, the color weights are derived from the NL-means distances as

$$w_c(p,q) = \exp(-\max(d_c^2(P(p),P(q)),0)). \quad (5)$$

Similar as for the color differences, we measure differences between pixels p and q of a feature channel f_j as

$$\Phi_j^2(p,q) = \frac{(f_j(p)-f_j(q))^2 - (\mathrm{Var}_j[p]+\mathrm{Var}_j[q])}{\varepsilon + k_f^2(V_j[p]+V_j[q]))},$$

where we use the same two-buffer estimate for the feature variance Var_j. In the denominator, we threshold the feature variance using $V_j[p] = \max(10^{-3}, \mathrm{Var}_j[p], \|\nabla_j[p]\|^2)$, where ∇_j denotes the finite difference gradient, to avoid being too strict with feature differences in non-smooth regions. Again, we set $\varepsilon = 1e-10$, and k_f is a user parameter. We normalize all features to have values in $[0,1]$. For vector-valued features, such as the normals, we view each vector element as one individual feature. We compute NL-means distances $d_{f_j}^2(p,q)$ equivalently to $d_c^2(p,q)$, using a radius t_f instead of t_c. Finally, the feature weights are

$$w_f(p,q) = \min_{j\in\{1,\dots,l\}} \exp(-\max(d_{f_j}^2(p,q),0)), \quad (6)$$

where l is the number of feature channels. From the color and feature weights we obtain our final weights

$$w(p,q) = \min(w_c(p,q), w_f(p,q)). \quad (7)$$

Finally, the elements of our diagonal weighting matrix are $D_p(q,q) = w(p,q)$.

It is important to realize that weighting the patch constraints does not change our patch basis vectors. Even if we downweight the patch constraint error for a certain pixel to near zero, that pixel is still included in our patch basis, and it will influence the projection of all other pixels in the patch onto the basis. We address this issue by completely removing pixels with very small weights below a threshold of $1e-10$ from the patch, that is, by removing these pixels from all feature vectors. This leads to patch constraints corresponding to arbitrarily shaped patches. We orthogonalize the modified feature vectors using SVD as described above.

5. Error Analysis

We perform an error analysis to better understand the behavior of our extended Poisson reconstruction approach. We express mean squared error (MSE) as the sum of squared bias and variance, and investigate the source of bias and variance separately. Let us write our noisy sampled image as $I^g = I_{\mathrm{ref}} + \Sigma^g$, where I_{ref} is the ground truth image, and Σ^g is a vector of i.i.d, zero-mean normally distributed random variables with variance σ^2. Similarly, we write the sampled gradients as $I^{dx} = H^{dx} I_{\mathrm{ref}} + \gamma\Sigma^{dx}$ and $I^{dy} = H^{dy} I_{\mathrm{ref}} + \gamma\Sigma^{dy}$. We assume both horizontal and vertical gradients have the noise variance $\gamma^2\sigma^2$, which is related to the variance of the pixels by a factor of γ^2. We also assume that the constraint matrix M is not influenced by noise in the input, and that we minimize the error under the L_2 norm. We then express our reconstruction I as the sum of the ground truth image I_{ref} and a per-pixel reconstruction error ε, $I = I_{\mathrm{ref}} + \varepsilon$. Substituting this into Equation (4) yields

$$\left(\beta^2 M^T M + H^T H + \alpha^2 \mathrm{Id}_n\right)(I_{\mathrm{ref}} + \varepsilon)$$
$$= \alpha(I_{\mathrm{ref}} + \Sigma^g) + H^{dx^T}(H^{dx} I_{\mathrm{ref}} + \gamma\Sigma^{dx}) + H^{dy^T}(H^{dy} I_{\mathrm{ref}} + \gamma\Sigma^{dy}).$$

Taking into account that the ground truth image I_{ref} satisfies the constraints of the conventional screened Poisson equations, that is,

$$\left(H^T H + \alpha^2 \mathrm{Id}_n\right) I_{\mathrm{ref}} = \alpha I_{\mathrm{ref}} + H^{dx^T} H^{dx} I_{\mathrm{ref}} + H^{dy^T} H^{dy} I_{\mathrm{ref}}, \quad (8)$$

and after some algebraic reformulation, we obtain an expression for the error

$$\varepsilon = A^{-1}\left(\alpha\Sigma^g + H^{dx^T}\gamma\Sigma^{dx} + H^{dy^T}\gamma\Sigma^{dy} - \beta^2 M^T M I_{\mathrm{ref}}\right), \quad (9)$$

where, for simplicity, we introduced the shorthand notation

$$A = \left(\beta^2 M^T M + H^T H + \alpha^2 \mathrm{Id}_n\right). \quad (10)$$

Bias. We now obtain the expected error, that is the bias, of our reconstruction as

$$E[\varepsilon] = -A^{-1}\beta^2 M^T M I_{\mathrm{ref}}, \quad (11)$$

where we exploited that the noise Σ^g, Σ^{dx}, and Σ^{dy} is zero-mean. Clearly, bias vanishes if $\beta = 0$, or the ground truth image fully satisfies the patch constraints, that is $M^T M I_{\mathrm{ref}} = 0$. Note that $E[\varepsilon]$ expresses the per-pixel bias, and we can obtain the squared bias of an image (or image region) as $E[\varepsilon]^T E[\varepsilon]$.

Variance. To compute variance we start by formulating $\varepsilon - E[\varepsilon]$ by combining Equations (9) and (11),

$$\varepsilon - E[\varepsilon] = A^{-1}\left(\alpha\Sigma^g + H^{dx^T}\gamma\Sigma^{dx} + H^{dy^T}\gamma\Sigma^{dy}\right). \quad (12)$$

The variance over an image (or image region) is $E[(\varepsilon - E[\varepsilon])^T(\varepsilon - E[\varepsilon])]$, and we find that

$$E[(\varepsilon - E[\varepsilon])^T(\varepsilon - E[\varepsilon])] = \sigma^2\left(\alpha^2 \mathrm{tr}(A^{-1^T}A^{-1})\right.$$
$$\left. + \gamma^2\left(\mathrm{tr}(H^{dx^T}A^{-1^T}A^{-1}H^{dx}) + \mathrm{tr}(H^{dy^T}A^{-1^T}A^{-1}H^{dy})\right)\right), \quad (13)$$

where tr is the matrix trace. We observe that variance goes to zero for large values of β, because A^{-1} tends to be dominated by a factor $1/\beta^2$ (see Equation (10)). Using a spectral analysis, one can also show that for $\beta = 0$ variance is minimized at $\alpha^2 = \gamma^2$, which corresponds to the result from previous work [KMA*15].

Discussion. In Figure 3, we visualize squared bias, variance, and their sum (that is, MSE) for a small image region over a range of the α, β, and k_c parameters of our method. On the left we show the image region (top), and the bias over this region depending on our parameters. Parameter α varies along the vertical, and β over the horizontal axis. In addition, for each α we plot a range of k_c values along the vertical axis. We show variance and MSE (middle and right) for two noise levels, assuming Gaussian noise for pixels and gradients, where the variance of the gradients is $\gamma^2 = 0.2$ times the variance of the pixels. In the top row, the MSE is dominated by the bias, and in the bottom row by the variance. Our main observation is that the lowest variance for any β, and the lowest MSE, are both

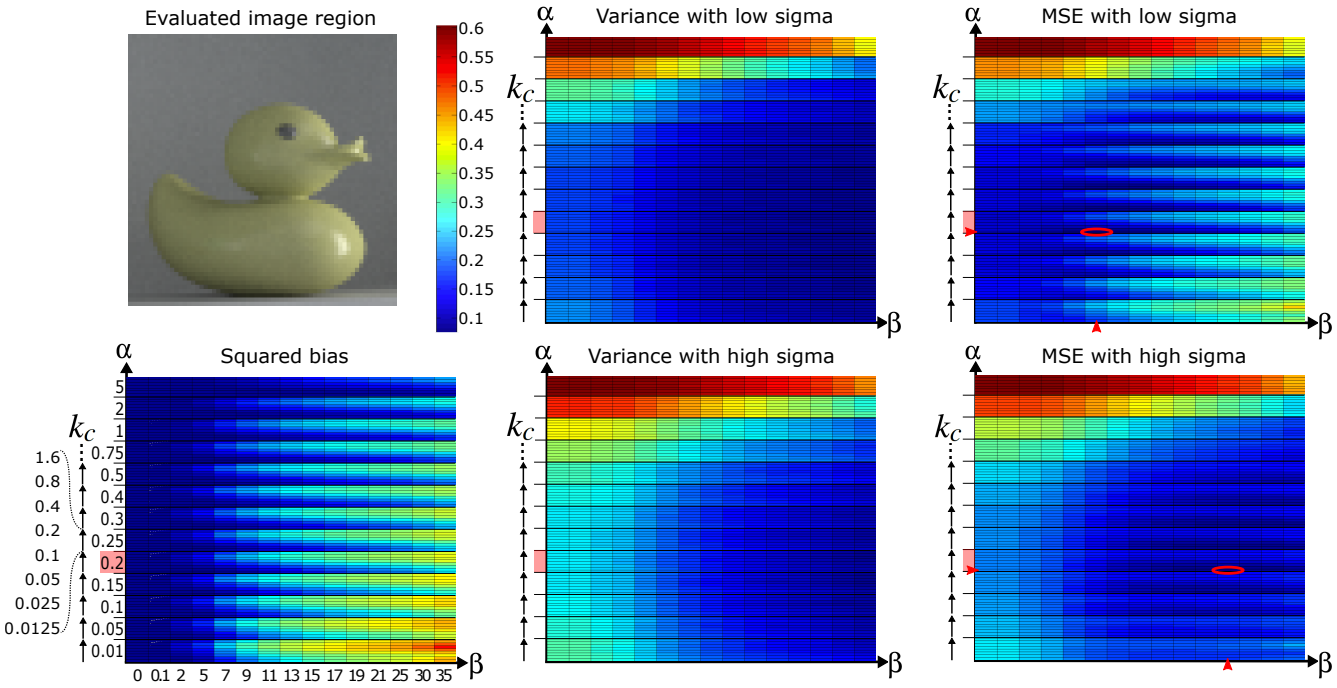

Figure 3: *We plot the logarithm of the bias, variance and mean squared error of a* 64×64 *pixels region of Bathroom according to Equation 11 and 13 for different* α, β *and* k_c. *We show* β *on the horizontal axis , and the unrolled parameters* α *and* k_c *on the vertical axis. We plot variance and MSE for two variance levels* $\sigma = 0.01$ *(low) and* $\sigma = 0.1$ *(high). We encircled the parameters with the minimal MSE in red. The optimal* α *and* k_c *are not affected by the noise level, whereas the optimal* β *gets shifted to the right with increasing noise.*

achieved at approximately $\alpha^2 \approx \gamma^2$. Parameter β provides a bias-variance tradeoff, where the value of β to minimize MSE depends on the noise level and on k_c.

6. Conjugate Gradient Solver

In practice, including the L_1 norm to solve Equation (3) provides superior results over pure L_2 minimization. We use an iteratively reweighted least squares (IRLS) approach, with the important extension to provide an estimate of the residual variance of the solution. This enables further post-processing based on the residual variance to improve the visual quality of the output, as described below, and to perform error estimation and reconstruction scale selection (Section 7). Solving Equation (3) is equivalent to obtaining

$$I = \arg\min_{\bar{I}} \left\| \underbrace{\begin{pmatrix} \alpha I_n \\ H^{dx} \\ H^{dy} \\ \beta M \end{pmatrix}}_{A} \bar{I} - \underbrace{\begin{pmatrix} \alpha I^b \\ I^{dx} \\ I^{dy} \\ 0 \end{pmatrix}}_{b} \right\|. \tag{14}$$

We estimate the variance of our solution by expressing the right hand side b as the sum of two image buffers $\hat{b}$ and $\check{b}$, where each contains one half of the samples that we rendered. We solve separately for both right hand sides, obtaining two solutions $\hat{I}$ and $\check{I}$. Because our solutions are linear in the two buffers $\hat{b}$ and $\check{b}$, we have $I = (\hat{I} + \check{I})/2$, and we can estimate the residual variance of I as $(\hat{I} - \check{I})^2/4$.

The matrix $A^T A$ is symmetric and positive-definite, hence we apply the conjugate gradient (CG) method. We summarize our modified CG algorithm in Figure 4. The main idea is to solve for $\hat{b}$ and $\check{b}$ simultaneously, while computing the IRLS weights based on the desired final solution for $b = (\hat{b} + \check{b})/2$. We use a fixed number of five reweighting and 500 CG steps. The IRLS weights are stored in the diagonal W matrix. We use a standard reweighting scheme (Line 14). We normalize the weights such that they average to one by multiplying W with the scalar $A.rows/\mathrm{tr}(W)$, where $A.rows$ is the number of rows in A and $\mathrm{tr}(W)$ the trace of W (Line 16). It is important, however, to note that we take the L_2 norm for the term $\|\alpha(\bar{I} - I^g)\|^2$ from Equation (3), that is, we do not reweight the corresponding elements of our system in Line 15. We found that using the L_1 norm for this term was not necessary thanks to our patch constraints, and the advantage of the L_2 norm is that we avoid the loss of image brightness due to outlier removal under the L_1 norm. Although we use a combination of L_1 and L_2 norm in practice, as opposed to the L_2 norm in our analysis in Section 5, we empirically observe similar behavior of bias and variance as summarized in Figure 3. In particular, we retain the ability to suppress a lot of noise by introducing only a little bias as shown in Figure 3.

GPU Implementation. We implemented our solver in CUDA. The regular structures of our sparse matrices allow us to calculate indices of non-zero values directly, instead of reading them from index arrays. As the algorithm is limited by memory bandwidth, this directly translates into performance gains. We also use shared

POISSONREGULARIZED$(A,\hat{b},\check{b},x_0)$

1 **for** $k = 1$ **to** 5
2 $\quad\hat{r}_0 = A^T W^2 \hat{b} - A^T W^2 A x_0$
3 $\quad\check{r}_0 = A^T W^2 \check{b} - A^T W^2 A x_0$
4 $\quad\hat{p}_0 = \hat{r}_0$; $\check{p}_0 = \check{r}_0$
5 $\quad$**for** $i = 1$ **to** 500
6 $\quad\quad\hat{q} = A^T W^2 A \hat{p}_i$; $\check{q} = A^T W^2 A \check{p}_i$
7 $\quad\quad\hat{\alpha} = (\hat{r}_i^T \hat{r}_i)/(\hat{p}_i^T \hat{q})$; $\check{\alpha} = (\check{r}_i^T \check{r}_i)/(\check{p}_i^T \check{q})$
8 $\quad\quad\hat{x}_{i+1} = \hat{x}_i + \hat{\alpha}\hat{p}_i$; $\check{x}_{i+1} = \check{x}_i + \check{\alpha}\check{p}_i$
9 $\quad\quad\hat{r}_{i+1} = \hat{r}_i - \hat{\alpha}\hat{q}$; $\check{r}_{i+1} = \check{r}_i - \check{\alpha}\check{q}$
10 $\quad\quad\hat{\beta} = (\hat{r}_{i+1}^T \hat{r}_{i+1})/(\hat{r}_i^T \hat{r}_i)$; $\check{\beta} = (\check{r}_{i+1}^T \check{r}_{i+1})/(\check{r}_i^T \check{r}_i)$
11 $\quad\quad\hat{p}_{i+1} = \hat{r}_{i+1} + \hat{\beta}\hat{p}_i$; $\check{p}_{i+1} = \check{r}_{i+1} + \check{\beta}\check{p}_i$
12 $\quad e = A(\hat{x}_{i+1} + \check{x}_{i+1}) - (\hat{b} + \check{b})/2$
13 $\quad$**for** $j = 1$ **to** $A.rows$
14 $\quad\quad W_{jj} = 1/(||e_j||_2 + 0.05 \times 0.5^{k-1})$
15 $\quad W = \text{setRowsToOne}(W, 1, x_0.rows)$
16 $\quad W = W \cdot A.rows/\text{tr}(W)$
17 **return** $[\hat{I} = \hat{x}_{i+1}, \check{I} = \check{x}_{i+1}]$

Figure 4: *Pseudocode of our modified IRLS CG solver. We compute the solution for two image buffers containing half the samples separately, which allows us to estimate the variance of the solution. Crucially, we compute the IRLS weights based on the residual of the average (Line 12), such that the average of our solution is equivalent to the solution of the average of the two buffers.*

memory and coalesced memory access patterns to further improve performance.

The most expensive step in the algorithm is the computation of the sparse matrix-vector product in line 6, where performance is limited by memory access to load non-zero matrix elements. We found that for our constraint matrix A it is beneficial to precompute the matrix product $A^T W^2 A$. For a feature patch radius of $r = 2$, the product has about seven times fewer non-zero elements than the matrix A itself. Precomputing $A^T W^2 A$ reduces the time required to read matrix elements for the computation of the matrix-vector product, and the corresponding performance increase for the matrix-vector product outweighs the cost of the precomputation.

We use a fixed number of conjugate gradient iterations, but it would also be possible to return earlier from the algorithm if it already has converged sufficiently. The convergence speed of the conjugate gradient algorithm depends on the root of the condition number of the matrix $A^T W^2 A$. Using our patch constraints, the condition number can get significantly larger than in the original Poisson problem and the algorithm can suffer from slow convergence. We thus use a default of 500 conjugate gradient iterations, while the original Poisson solver only uses 50 iterations. We could resort to the preconditioned conjugate gradient method, but finding a good preconditioner can be challenging and is left for furture work.

Post-Processing. The CG solver returns two images $\hat{I}$ and $\check{I}$, and we estimate the variance in the reconstructed image $I = (\hat{I} + \check{I})/2$ as $(\hat{I} - \check{I})^2/4$. We leverage this estimate in a post-processing step that is targeted at further reducing variance and artifacts. We achieve this by filtering I with a cross NL-means filter using weights as described in Section 4, Equation (7). We provide more details in Section 7. Figure 5 illustrates the post-processing step.

Before post-processing (relative MSE: 0.0176) | After post-processing (relative MSE: 0.0083)

Figure 5: *This figure shows the effect post-processing step on Bookshelf using G-BDPT with 32 samples per pixel and the third reconstruction scale (Section 7).*

7. Multiscale Reconstruction and Scale Selection

We have shown (Figure 3) that by adjusting the parameters of our approach we can minimize the reconstruction error depending on the relative amount of noise in the rendered pixels and gradients (α parameter), and depending on the levels of bias and variance (β, k_c parameters). Since these properties vary within each image, we expect to be able to reduce error by adjusting parameters locally. We employ a multiscale reconstruction and scale selection approach similar as in previous work to achieve this: we obtain three scales by running our reconstruction with different parameters, estimate the error of each scale locally, and combine them into a final image by selecting for each pixel the scale that minimizes the error.

Reconstruction Parameters. Our goal was to determine the α, β, and k_c parameters for three reconstruction scales by finding parameters that minimize the error over a set of training images at different sample counts. We found that the other reconstruction parameters are insensitive to local bias and noise levels, and we set them to constants reported in Section 8. Let us denote the set of reconstruction scales in our search by P, where scale i has parameters $P_i = [\alpha_i, \beta_i, k_{c,i}]$. We determined the combination of three scales i_1, i_2 and i_3 that yielded the smallest error of all combinations.

For simplicity we ran a brute force search where we evaluated scales with $\alpha \in \{0.05, 0.1, \ldots 0.25\}$, $\beta \in \{3, 5, \ldots, 19\}$, and $k_c \in \{0.025, 0.05, 0.1, \ldots, 0.35\}$ for a total of $|P| = 360$ scales, and $360^3 \approx 47$m combinations. For each combination we computed the reconstruction error over the training images. Since we want to select scales locally, we computed the reconstruction error for small image patches. We chose 10×10 pixels in practice. Let $\text{relMSE}(i, p)$ be the reconstruction error of scale i measured in terms of relative MSE over a patch around pixel p with respect to a reference image. The error of a combination of three scales for the patch around p is then $\text{relMSE}(i_1, i_2, i_3, p) = \min(\text{relMSE}(i_1, p), \text{relMSE}(i_2, p), \text{relMSE}(i_3, p))$, and the total error of each combination i_1, i_2, i_3 is the sum over all local patches in all training images.

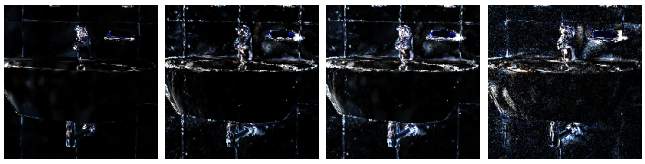

Figure 6: *We compare our error estimate, with variance and bias, to the ground truth on Bathroom (1024 samples per pixel).*

Error Estimation. Given the three scales, which we determined using the procedure described above, we estimate the MSE of each scale to enable local scale selection. We express MSE as the sum of squared bias and variance and estimate each separately.

The bias of a reconstructed scale I_i is defined as the difference of the expected value of the reconstructed image I_i and the ground truth image I_{ref}, that is, $Bias(I_i) = E[I_i] - I_{\text{ref}}$. Since the expected value of the base image I^g is the ground truth we can reformulate this to $Bias(I_i) = E[I_i - I^g]$. While $I_i - I^g$ is trivial to compute, I^g is typically very noisy, which will lead to a high variance in the bias estimate. Even worse, squaring it to get the squared bias will systematically overestimate the true bias. Hence, we reduce the noise of both I^g and I_i by filtering both images with an identical edge preserving filter. We use the color based NL-means distances $w_c(p,q)$ from Equation (5) with the parameters $k_c = 0.6$ and $f = 25$ as the filtering weights. Intuitively, this aggregates data of similar regions to get a more reliable bias estimate. For the variance estimation we use the squared difference between our two reconstructed buffers as explained in Section 6. Finally, after adding our variance and squared bias estimates, we filter this MSE estimate with a small 3×3 box-filter to further remove noise.

Scale Selection. We obtain our final image I by interpolating the values of the three scales in each pixel with weights that are inversely proportional to the estimated errors,

$$I(p) = \sum_{i=1}^{3} \frac{1/(\text{estMSE}(I_i(p)) + \varepsilon)}{\sum_j 1/(\text{estMSE}(I_j(p)) + \varepsilon)} I_i(p), \tag{15}$$

where $\text{estMSE}(I_i(p))$ is our error estimate of scale i at pixel p, $I_i(p)$ is the reconstructed value of scale i at pixel p, and $\varepsilon = 10e-10$ prevents divisions by zero. Figure 7 compares a selection based on our error estimate and a selection based on the true error. Our error estimation and scale selection effectively reduces the error of the final image below the error of any of the individual scales.

Final Algorithm. Figure 8 shows the complete reconstruction algorithm. The inputs are the rendering outputs distributed into two separate buffers, including the base image $\hat{I}^g, \check{I}^g$, the gradients $\hat{I}^{dx}, \check{I}^{dx}, \hat{I}^{dy}, \check{I}^{dy}$, and features $\hat{F}, \check{F}$, as well as the reconstruction scale parameters α_i, β_i and $k_{c,i}$, $i \in \{1,2,3\}$. The features are interpreted as images with one channel per feature. Many reconstruction steps must be applied on both buffers ($\hat{\cdot}$ and $\check{\cdot}$) separately. We represent steps that are applied on both buffers separately with the star subscript ($\cdot_\star$). We denote the average of both buffers as $\text{avg}(\cdot)$, and $\text{var}(\cdot)$ computes the two-buffer variance as described in Section 6.

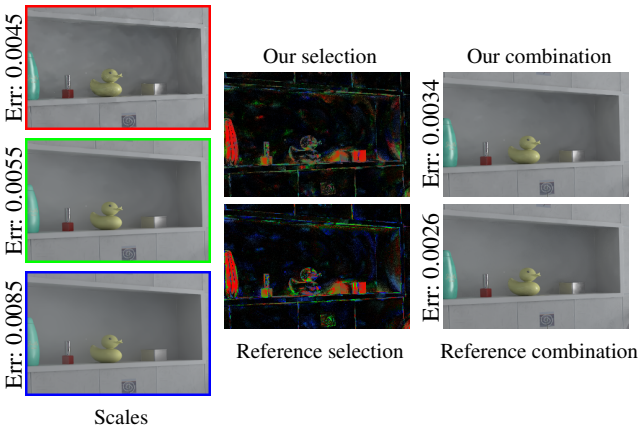

Figure 7: *We compare our scale selection to a reference selection based on ground truth data. The color of the selection maps in the middle are set according to the border colors of the scales on the left. To highlight regions where correct scale selection is crucial we modulated the brightness of the selection map per pixel by the highest error of any scale in that pixel. We successfully select the correct scales in regions where the scales differ dramatically (e.g. the highlights) and obtain an overall numerical benefit.*

Line 1 unrolls the input into a 1-d vector, used as the right hand side in Equation 14. Line 2 removes excess noise from the features with a NL-means filter, processing each channel of F separately. This is similar to the feature pre-processing described in Rousselle et al. [RMZ13], and we found it to be beneficial in addition to SVD truncation. We compute filter weights using Equations (5), (6), and (7). The function $\text{nlmFilter}(\cdot,\cdot,\cdot,\cdot,\cdot)$ takes as arguments, in this order, the image to be filtered, a guide image and its variance to compute color weights (Equation (5)), a multi-channel feature image (one channel per feature) and their variances to compute feature weights (Equation (6)). We set the parameters to $k_f = 1$, the NL-means radius to $t_f = 3$, and we ignore the color guide image. We use a filter window size of 11×11 pixels. Line 3 applies standard L_1 Poisson reconstruction with $\alpha = 0.2$ on both inputs, and the resulting images $G_\star$ are used in Line 5 to compute per-pixel weights for the patch constraints (Section 4).

We apply the reconstruction steps from Line 4 to 10 for each scale separately. Line 5 and 6 builds the constraint-matrix A of our equation system. Line 7 is the core of our algorithm where we perform the regularized reconstruction as described in Section 6. The found optimal parameters for the three scales are $\alpha_{1,2,3} = [0.25, 0.25, 0.1]$, $\beta_{1,2,3} = [5, 17, 19]$ and $k_{c,1,2,3} = [0.1, 0.35, 0.15]$. For the regularized Poisson step we further set $r = 2$, $t_c = 1$ and $t_f = 0$ and $k_f = \infty$ for all scales. Line 8 applies the post-process filter on the reconstruction to get rid of residual noise (see also Section 6). The parameters for the NL-means filter are $k_c = 0.45$, $k_f = 1$, $t_c = 1$, $t_f = 0$, and we filter over windows of 21×21 pixels. Finally, we get the reconstruction scale I_i by averaging the two reconstructed buffers (Line 9). Line 10 estimates the reconstruction error I_i as described earlier in this section. We obtain our final output by combining the three scales using Equation 15 (Line 11).

RECONSTRUCT($\hat{I}^g, \check{I}^g, \hat{I}^{dx}, \check{I}^{dx}, \hat{I}^{dy}, \check{I}^{dy}, \hat{F}, \check{F}, \alpha, \beta, k_c$)

1 $b_\star = \text{unroll}(I^g_\star, I^{dx}_\star, I^{dy}_\star)$
2 $F_\star = \text{nlmFilter}(F_\star, \text{null}, \text{null}, \text{avg}(F), \text{var}(F))$
3 $G_\star = \text{poisson}_{\text{L1}}(I^g_\star, I^{dx}_\star, I^{dy}_\star, 0.2)$
4 **for** $i = 1$ to 3
5 $\quad M = \text{buildM}(\text{avg}(G), \text{var}(G), \text{avg}(F), \text{var}(F), k_{c,i})$
6 $\quad A = \text{buildA}(M, \alpha_i, \beta_i)$
7 $\quad [\hat{x}, \check{x}] = \text{poissonRegularized}(A, \hat{b}, \check{b}, \text{avg}(I^g))$
8 $\quad x_\star = \text{nlmFilter}(x_\star, avg(x), var(x), avg(F), var(F))$
9 $\quad I_i = \text{avg}(x)$
10 $\quad \text{estMSE}_i = \text{estimateError}(I_i, \text{avg}(I^g), \hat{x}, \check{x})$
11 $I = \text{combineScales}(I_1, I_2, I_3, \text{estMSE}_1, \text{estMSE}_2, \text{estMSE}_3)$
12 **return** I

Figure 8: *Pseudocode of our complete reconstruction pipeline.*

8. Results and Discussion

Implementation and Performance. We implemented our approach on top of the gradient-domain path tracing (G-PT) [KMA*15] and gradient-domain bidirectional path tracing (G-BDPT) [MKA*15] implementations in Mitsuba [Jak10] by the original authors. We modified G-PT and G-BDPT to render the sampling data into two separate buffers and to store feature information (position, normal and texture). We also use an ambient occlusion feature that helps to preserve some shading effects. We use existing Mitsuba functionality to efficiently generate the ambient occlusion maps. On an NVidia Titan X our approach takes about one minute to reconstruct a one mega-pixel image. The pre-filtering of all our features (Line 2) takes 0.5 seconds, solving the L_1 reconstruction for both input buffers (Line 3) 0.3 seconds, building the patch constraint matrix M (Line 5) 5 seconds per scale, solving the regularized reconstruction (Line 7) 14 seconds per scale, and post processing both outputs (Line 8) takes 1.1 second per scale. The time taken for error estimation and final scale selection is negligible. Memory consumption of our implementation is rather high since we store $A^T W^2 A$ on the GPU. For a one mega-pixel image we require approximately 7.5GB of GPU memory. This could be reduced by performing the reconstruction on overlapping tiles of the image separately, at the cost of some performance.

Comparison to Previous Work. We compare the denoising performance of our algorithm in Figure 9 and 10 to gradient-domain rendering using conventional L_1 reconstruction (L1) and to *Robust Denoising with Feature and Color Information* (RDFC) [RMZ13]. We measure all errors as relative mean squared error (rel. MSE) $E = \text{mean}((I - I_{ref})^2/(I^2_{ref} + 10^{-3}))$. Figure 10 shows that our approach outperforms L_1 reconstruction by a large margin. While L_1 reconstruction suffers from isolated spikes and low frequency noise, our reconstruction yields a clean image even at low sampling counts. In our tested scenes we report a significant improvement in relative MSE of a factor of 1.5 to 5.

RDFC is representative for recent image space denoising techniques for Monte Carlo rendering, achieving state of the art performance for moderate and higher sampling rates (64 samples per pixel and higher) [KBS15]. It leverages features similar to our approach, and for a fair comparison we used the exact same features for both algorithms, including RGB texture, normal, world space position, and a scalar ambient occlusion term. We use RDFC with uniform sampling, since adaptive sampling for gradient-domain rendering has not been described in the literature yet. We compare our reconstruction on top of G-BDPT to RDFC on top of BDPT, and our reconstruction on top of G-PT to RDFC on top of PT. The authors of G-PT and G-BDPT report an overhead of their methods of roughly 2.5x for G-PT and 4x for G-BDPT compared to the base algorithms. Hence, for equal time comparisons RDFC can use 2.5x more base samples when applied on top of PT and 4x more base samples when applied on top of BDPT. Despite this, we still achieve slightly better results with our method in our tested scenes. With PT as base algorithm we show in Figure 10 that our method improves upon RDFC in Bathroom and Bookshelf by 12% and in Sponza by 50%. With BDPT as base algorithm we improve upon RDFC in Bottle by 16%. Finally, Figure 9 demonstrates the convergence of our method for the Bathroom scene using a log-log plot. We observe that our method converges to the correct solution, with errors by a factor of three to five lower than L_1 reconstruction. Independent of rendering time, our method also consistently outperforms RDFC time in this scene.

Discussion. The denoising results of our method and RDFC are quite similar despite the rather different reconstruction approaches. In regions that are well captured by features, both algorithms manage to remove variance nearly completely without introducing artifacts. Remaining error concentrates mostly at edges or around scene details that are not well captured by the features. Our method seems to benefit from the gradient information in these areas, which often manages to capture such details with less noise. One such example is the shading on the rubber duck in the Bathroom scene (Figure 10). Gradients do not guarantee to reduce noise, however, and there are also image regions where our approach has higher error than RDFC. An interesting observation is that RDFC uses very large filtering kernels of size 21×21 or even 41×41 pixels, while our method achieves comparable results using patch constraints of size 5×5 ($r = 2$). This is because our patch constraints are linked in a global system, and as we showed, the parameter β allows us to reduce variance independently of patch size.

To ensure a fair comparison to RDFC, we also tried to enhance RDFC with the gradient information directly. First, we tried to post-process the conventional screened Poisson reconstruction with RDFC, and second, we tried to prefilter both gradients and pixels with RDFC, followed by conventional screened Poisson reconstruction. In equal time comparisons, both these variants perform significantly worse than RDFC without gradients and our proposed approach.

Our approach is also orthogonal to specifics of the underlying gradient-domain rendering algorithm, as long as the renderer returns auxiliary feature data. Since variance computation relies on a two-buffer approach [RKZ12], our technique can also be used with Metropolis gradient-domain rendering [LKL*13,MRK*14], for example. This would simply require to run two Markov chains with different seeds. In summary, our approach closes the gap between gradient-domain rendering and image space denoising. A key advantage over existing image space denoising techniques is that it

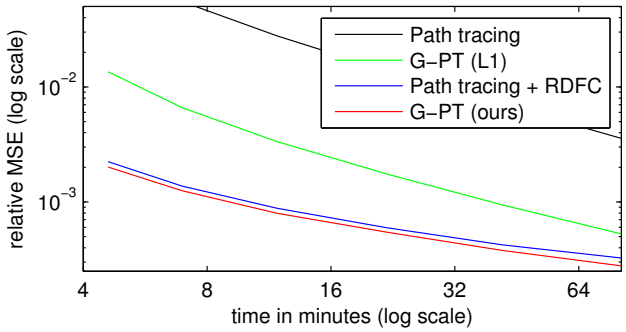

Figure 9: *Convergence for the Bathroom scene in a log-log plot. We show path-tracing without de-noising (black), path-tracing with RDFC (blue), gradient-domain path-tracing using L_1 reconstruction (green), and our reconstruction (red).*

will benefit from future improvements in gradient-domain rendering, such as improvements in gradient sampling using better shift mappings, higher order finite differences, or adaptive sampling.

9. Conclusions

We presented an improved reconstruction technique for gradient-domain rendering that leverages auxiliary image features. A GPU implementation processes one mega-pixel images in about one minute. Our approach outperforms previous solutions of the screened Poisson equation under the L_1 norm by a large margin. We also compare our approach to a state of the art image space denoising technique for Monte Carlo rendering and demonstrate, for the first time, that gradient-domain rendering can outperform conventional denoising. The margin, however, is modest and our observations raise the challenge to further improve gradient sampling. Since our approach is orthogonal to the underlying sampling distribution, there are various avenues for future work. For example, adaptive sampling based on the estimated error of the reconstructed image should be straightforward to add on top of our improved reconstruction technique. Finally, enforcing temporal consistency would also be an interesting direction for future work.

References

[AEB06] AHARON M., ELAD M., BRUCKSTEIN A.: K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Trans. Signal Process. 54*, 11 (Nov 2006), 4311–4322. 3

[BCM05] BUADES A., COLL B., MOREL J.-M.: A non-local algorithm for image denoising. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 2005.* (June 2005), vol. 2, pp. 60–65 vol. 2. 2, 4

[BEM11] BAUSZAT P., EISEMANN M., MAGNOR M.: Guided image filtering for interactive high-quality global illumination. *Computer Graphics Forum 30*, 4 (June 2011), 1361–1368. 2

[DSHL10] DAMMERTZ H., SEWTZ D., HANIKA J., LENSCH H. P. A.: Edge-avoiding à-trous wavelet transform for fast global illumination filtering. In *Proceedings of the Conference on High Performance Graphics* (2010), Eurographics Association, pp. 67–75. 2

[ED04] EISEMANN E., DURAND F.: Flash photography enhancement via intrinsic relighting. *ACM Trans. Graph. 23*, 3 (Aug. 2004), 673–678. 2

[HST10] HE K., SUN J., TANG X.: Guided image filtering. In *Proceedings of the 11th European conference on Computer vision: Part I* (Berlin, Heidelberg, 2010), ECCV'10, Springer-Verlag, pp. 1–14. 2

[Jak10] JAKOB W.: Mitsuba renderer, 2010. http://www.mitsuba-renderer.org. 9

[KBS15] KALANTARI N. K., BAKO S., SEN P.: A machine learning approach for filtering monte carlo noise. *ACM Trans. Graph. 34*, 4 (July 2015), 122:1–122:12. 2, 9

[KFF*15] KELLER A., FASCIONE L., FAJARDO M., GEORGIEV I., CHRISTENSEN P., HANIKA J., EISENACHER C., NICHOLS G.: The path tracing revolution in the movie industry. In *ACM SIGGRAPH 2015 Courses* (2015), pp. 24:1–24:7. 1

[KMA*15] KETTUNEN M., MANZI M., AITTALA M., LEHTINEN J., DURAND F., ZWICKER M.: Gradient-domain path tracing. *ACM Trans. Graph. 34*, 4 (Aug. 2015). 1, 2, 3, 5, 9

[LKL*13] LEHTINEN J., KARRAS T., LAINE S., AITTALA M., DURAND F., AILA T.: Gradient-Domain Metropolis Light Transport. *ACM Trans. Graph. 32*, 4 (2013). 1, 2, 3, 9

[LWC12] LI T.-M., WU Y.-T., CHUANG Y.-Y.: Sure-based optimization for adaptive sampling and reconstruction. *ACM Trans. Graph. 31*, 6 (Nov. 2012), 194:1–194:9. 2

[MCY14] MOON B., CARR N., YOON S.-E.: Adaptive rendering based on weighted local regression. *ACM Trans. Graph. 33*, 5 (Sept. 2014), 170:1–170:14. 2, 4

[MIGYM15] MOON B., IGLESIAS-GUITIAN J. A., YOON S.-E., MITCHELL K.: Adaptive rendering with linear predictions. *ACM Trans. Graph. 34*, 4 (July 2015), 121:1–121:11. 2

[MJL*13] MOON B., JUN J. Y., LEE J., KIM K., HACHISUKA T., YOON S.-E.: Robust image denoising using a virtual flash image for monte carlo ray tracing. *Computer Graphics Forum 32*, 1 (2013), 139–151. 2

[MKA*15] MANZI M., KETTUNEN M., AITTALA M., LEHTINEN J., DURAND F., ZWICKER M.: Gradient-domain bidirectional path tracing. In *Proc. Eurographics Symposium on Rendering* (2015). 1, 2, 3, 9

[MRK*14] MANZI M., ROUSSELLE F., KETTUNEN M., LEHTINEN J., ZWICKER M.: Improved sampling for gradient-domain metropolis light transport. *ACM Trans. Graph. 33*, 6 (Nov. 2014), 178:1–178:12. 2, 3, 9

[Ren] Renderman denoiser. `http://renderman.pixar.com/view/denoiser`. Accessed: 2015-08-30. 2

[RKZ12] ROUSSELLE F., KNAUS C., ZWICKER M.: Adaptive rendering with non-local means filtering. *ACM Trans. Graph. 31*, 6 (Nov. 2012), 195:1–195:11. 2, 4, 9

[RMZ13] ROUSSELLE F., MANZI M., ZWICKER M.: Robust denoising using feature and color information. *Computer Graphics Forum 32*, 7 (2013), 121–130. 2, 4, 8, 9

[SAC*11] SHIRLEY P., AILA T., COHEN J., ENDERTON E., LAINE S., LUEBKE D., MCGUIRE M.: A local image reconstruction algorithm for stochastic rendering. In *Prof. ACM Symposium on Interactive 3D Graphics and Games* (2011), pp. 9–14. 2

[SD12] SEN P., DARABI S.: On filtering the noise from the random parameters in monte carlo rendering. *ACM Trans. Graph. 31*, 3 (June 2012), 18:1–18:15. 2

[Ste81] STEIN C. M.: Estimation of the mean of a multivariate normal distribution. *The Annals of Statistics 9*, 6 (1981), pp. 1135–1151. 2

[vdBF08] VAN DEN BERG E., FRIEDLANDER M. P.: Probing the pareto frontier for basis pursuit solutions. *SIAM Journal on Scientific Computing 31*, 2 (2008), 890–912. 3

[VG97] VEACH E., GUIBAS L. J.: Metropolis light transport. In *Proc. ACM SIGGRAPH 97* (1997), pp. 65–76. 2

[ZJL*15] ZWICKER M., JAROSZ W., LEHTINEN J., MOON B., RAMAMOORTHI R., ROUSSELLE F., SEN P., SOLER C., YOON S.-E.: Recent advances in adaptive sampling and reconstruction for monte carlo rendering. *Computer Graphics Forum 34*, 2 (2015), 667–681. 2

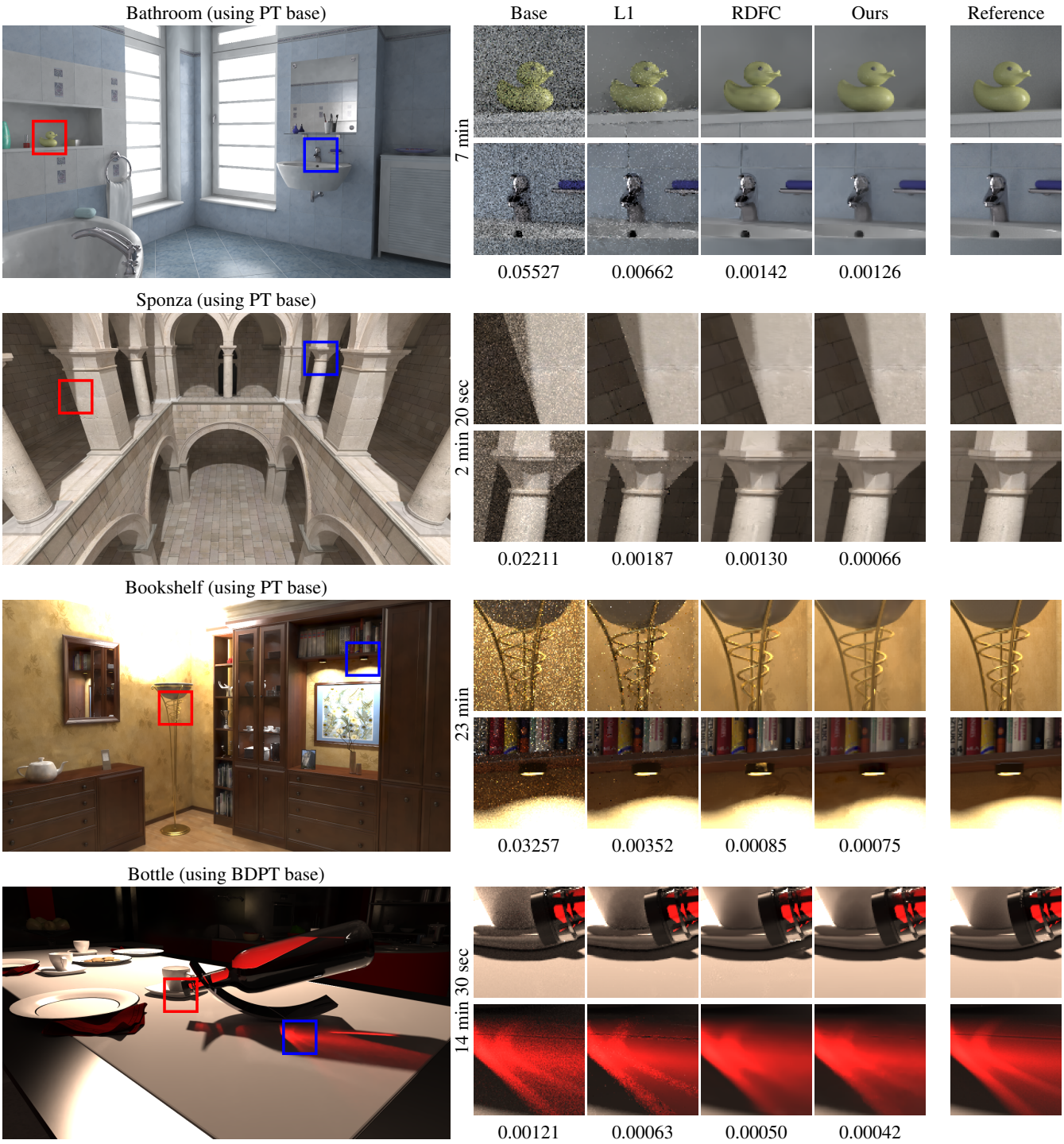

Figure 10: *We compare our new algorithm (Ours) on several scenes to gradient-domain rendering using the L_1 reconstruction (L1), to ordinary rendering using Robust Denoising (RDFC) and to the used base algorithm (Base). The base algorithm for Bathroom, Sponza and Bookshelf is unidirectional path tracing, while for Bottle it is bidirectional path tracing. The rendering times for all methods are approximatively the same (except for the reference image) and are shown on the left of the insets. Below the insets we show the relative mean squared error of each method for the entire image. The full resolution images on the left-most column show the result using our method.*