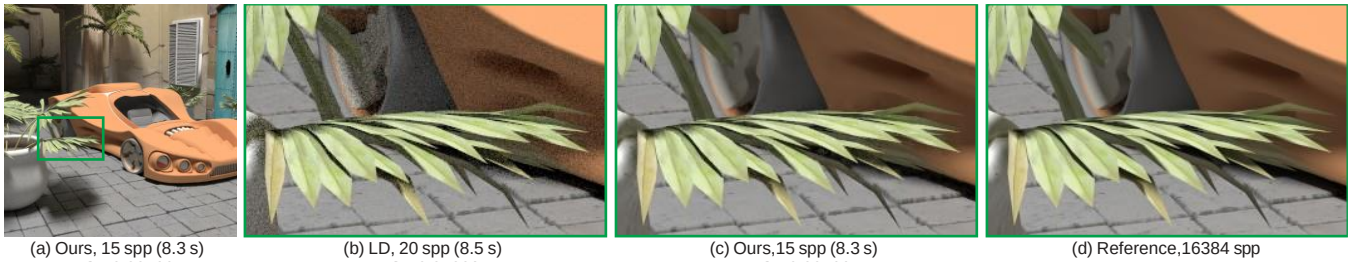
Adaptive Rendering with Linear Predictions

Bochang Moon[1], Jose A. Iglesias-Guitian[1], Sung-Eui Yoon[2], Kenny Mitchell[1]
[1]Disney Research Zu¨rich, [2]KAIST



(a) Ours, 15 spp (8.3 s) rMSE 0.00199 | (b) LD, 20 spp (8.5 s) rMSE 0.01003 | (c) Ours,15 spp (8.3 s) rMSE 0.00199 | (d) Reference,16384 spp

Figure 1: *Our adaptive rendering result in the Courtyard scene. Our method with 15 samples per pixel (spp) produces a high-quality reconstruction result and drastically reduces a relative mean squared error (rMSE) compared to a straight-forward method utilizing low discrepancy (LD) sampling patterns uniformly.*

We propose a new adaptive rendering algorithm that enhances the performance of Monte Carlo ray tracing by reducing the noise, i.e., variance, while preserving a variety of high-frequency edges in rendered images through a novel prediction based reconstruction. To achieve our goal, we iteratively build multiple, but sparse linear models. Each linear model has its prediction window, where the linear model predicts the unknown ground truth image that can be generated with an infinite number of samples. Our method recursively estimates prediction errors introduced by linear predictions performed with different prediction windows, and selects an optimal prediction window minimizing the error for each linear model. Since each linear model predicts multiple pixels within its optimal prediction interval, we can construct our linear models only at a sparse set of pixels in the image screen. Predicting multiple pixels with a single linear model poses technical challenges, related to deriving error analysis for regions rather than pixels, and has not been addressed in the field. We address these technical challenges, and our method with robust error analysis leads to a drastically reduced reconstruction time even with higher rendering quality, compared to state-of-the-art adaptive methods. We have demonstrated that our method outperforms previous methods numerically and visually with high performance ray tracing kernels such as OptiX and Embree.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Raytracing

Keywords: Adaptive rendering, image-space reconstruction, Monte Carlo ray tracing

ACM Reference Format
Moon, B., Iglesias-Guitian, J., Yoon, S., Mitchell K. 2015. Adaptive Rendering with Linear Predictions. ACM Trans. Graph. 34, 4, Article 121 (August 2015), 11 pages. DOI = 10.1145/2766992 http://doi.acm.org/10.1145/2766992 .

ACM 978-1-4503-3331-3/15/08 ... $15.00.
DOI: http://dx.doi.org/10.1145/2766992

1 Introduction

Monte Carlo (MC) ray tracing [Kajiya 1986] has received extensive attention for synthesizing realistic rendering results, but generally requires a huge number of ray samples (e.g., more than ten thousands samples per pixel) until a converged or even visually pleasing image is generated. Unfortunately, its slow convergence directly leads to prohibitive rendering time (e.g., hours), which is often proportional to the number of ray samples generated. When a relatively small number of ray samples (e.g., less than one hundred) per pixel are allocated, images are typically corrupted by MC noise, i.e., variance.

Adaptive rendering that adjusts sampling density non-uniformly and applies smoothing locally has been actively studied recently, as the approach significantly boosts MC ray tracing by reducing the required number of ray samples drastically [Hachisuka et al. 2008; Overbeck et al. 2009]. These methods can be classified into two categories, multi-dimensional and image space adaptive rendering in terms of the dimensionality of MC samples [Moon et al. 2014]. The multi-dimensional methods [Hachisuka et al. 2008; Lehtinen et al. 2012] allocate samples and reconstruct them in a high dimensional space, where each coordinate corresponds to a random parameter in the MC integration [Kajiya 1986]. These methods can produce a high quality image even with a small number of samples (e.g., 8 samples per pixel), but managing individual samples unfortunately requires high computational and memory overhead.

On the other hand, image space methods [Rousselle et al. 2012; Li et al. 2012; Moon et al. 2014] utilize per-pixel information (e.g., colors, variances, and G-buffers) that can be easily obtained in rendering, and thus these techniques can be easily applied into existing rendering frameworks. The state-of-the-art methods (e.g., [Rousselle et al. 2012; Li et al. 2012; Moon et al. 2014]) have been shown to improve the performance of MC ray tracing by an order of magnitude. Their main target applications, however, are often limited to offline rendering frameworks, since its computational overhead is relatively large. For example, the reconstruction times of the previuos methods [Rousselle et al. 2012; Li et al. 2012; Moon et al. 2014] are more than 3 s given the Courtyard scene (Fig. 1) due to their expensive reconstructions (e.g., non-local means and local regression). Especially, the recent local linear approximation [Moon et al. 2014] shows a superior reconstruction performance when a reference image has a strong linear correlation with given features (e.g., textures), but it has very expensive reconstruction time (e.g.,

18 s in the scene of Fig. 1), since it utilizes a complex optimization process (least-squares fitting).

To address this problem, we propose a new adaptive rendering method, which performs expensive model reconstruction and optimization only at a small number of pixels and predicts filtered values in other pixels by using estimated linear models. The key difference between our work and the previous methods is that our method estimates optimal colors in a region (i.e., multiple pixels) by performing a novel linear prediction based reconstruction. Specifically, our major technical contributions are summarized as the following:

- We construct multiple linear models iteratively by using a recursive least squares. Our method estimates coefficients of linear models recursively given prediction windows with different sizes, where we predict multiple pixels from the linear models (Sec. 4.1).
- We design a recursive error analysis to estimate the prediction error introduced by our linear prediction, and select an optimal prediction size by using the error analysis (Sec. 4.2).
- We provide an adaptive sampling approach which allocates more ray samples on high error regions based on our estimated prediction errors (Sec. 5.1).

We have demonstrated our method with high-performance ray tracing kernels such as Embree [Wald et al. 2014] and OptiX [Parker et al. 2010], and our result shows higher rendering quality compared to the state-of-the-art methods [Rousselle et al. 2012; Li et al. 2012; Moon et al. 2014] in equal-time comparisons thanks to its accurate error analysis and lower computational overhead. Our method uses a sophisticated optimization (e.g., least squares fitting), but we drastically reduce the optimization overhead (e.g., 28× lower than the previous method [Moon et al. 2014] in the Fig. 1), by running our optimization only at a sparse number of pixels thanks to our prediction, while preserving its high reconstruction quality.

2 Previous Work

Multi-dimensional adaptive rendering. As an early work, Kajiya [1986] proposed a high-level idea to allocate high-dimensional samples in a hierarchical manner and to reconstruct outputs based on the samples stored in a tree structure (e.g., kd-trees) by using a Riemann sum. In a similar research line, Hachisuka et al. [2008] refined the idea and demonstrated that this approach significantly reduces the number of samples required for synthesizing a variety of rendering effects. Frequency analysis based anisotropic reconstruction often focused on simulating specific rendering effects such as depth-of-field [Soler et al. 2009], motion blur [Egan et al. 2009], soft shadows [Egan et al. 2011b], and ambient occlusions [Egan et al. 2011a]. Lehtinen et al. [2011] presented a new reprojection method to reuse samples among multiple pixels in order to reduce noise introduced by distributed effects, and Lehtinen et al. [2012] extended the idea to support indirect illumination. These methods demonstrated that high quality reconstruction can be achieved even with a small number of samples. These techniques, unfortunately, support a limited set of rendering effects.

Image-space adaptive rendering. Image-space approaches generally take per-pixel information (e.g., colors and variances) as an input, and then apply well-known image filters such as wavelet thresholding [Overbeck et al. 2009] and Gaussian filter [Rousselle et al. 2011] to fully utilize rendering-specific information (e.g., variances). Recently, sophisticated error estimation has been developed for supporting superior filtering methods. Kalantari et al. [2013] proposed a robust error metric based on the median absolute deviation. Rousselle et al. [2012] divided input samples into two buffers (i.e., dual buffer) and estimated the error introduced by the non-local means, and Delbracio et al. [2014] proposed a new similarity measure between two patches by using a histogram constructed with samples. Li et al. [2012] introduced a general unbiased error metric (i.e., Stein's unbiased risk estimator) to improve reconstruction quality of non-linear filtering methods (e.g., cross-bilateral filter), and the error metric was also utilized to decide how to combine the filter weights computed from an input color buffer and a feature buffer (e.g., geometries) in the non-local means filter [Rousselle et al. 2013]. Although these methods employ different error analysis and filters, their common behaviour for high-quality filtering is to apply a filter with estimated optimal parameters, e.g., bandwidths, at each pixel. Fortunately, these methods can be easily parallelized thanks to their image-space nature. Nonetheless, these approaches become computationally heavy and may take a number of seconds, since they require sophisticated error analysis as well as expensive filtering for high-quality or error-guaranteeing results. As a result, these methods have not been adopted to recent interactive rendering systems such as Embree [Wald et al. 2014] and OptiX [Parker et al. 2010], which use high-performance kernels. Computationally efficient filters such as A-trous [Dammertz et al. 2010] and guided filter [Bauszat et al. 2011] have been developed for real-time rendering. The usage of the real-time filters, however, has been limited to a preview, since its filtering quality can be sub-optimal due to the lack of robust error analysis.

Most recently, Moon et al. [2014] approximated image functions with linear models locally using features (e.g., G-buffers). In addition, they developed an error estimation of a local linear approximation by decomposing its reconstruction error into bias and variance, and estimated optimal filter bandwidths for different features to minimize the error. They demonstrated that the local regression framework can produce high-quality rendering results for a variety of rendering effects. However this method, like other high-quality adaptive techniques, suffers from a high computational overhead, since expensive optimization for estimating the optimal bandwidths is performed at every single pixel. Our method also utilizes a local linear approximation using G-buffers, but the key difference is that our approach reconstructs multiple pixels simultaneously from a single linear model so that expensive error estimation can be performed only at a sparse number of pixels, resulting in a drastically reduced computational overhead.

Table 1: *Notation used throughout this paper*

Symbol	Description
y	input image generated by Monte Carlo ray tracing
$\mathbf{x}$	feature vector for a pixel, which includes its pixel position and geometries (e.g., normal, texture and depth)
$f(\mathbf{x})$	ground truth image as a function of $\mathbf{x}$
$\nabla f(\mathbf{x})$	ground truth gradient of $f(\mathbf{x})$
Ω_c^F	filtering window with a fixed size (e.g., 19 × 19) defined at center pixel c
$\Omega_c^P(k)$	prediction window with a variable size k centered at pixel c
$\hat{\beta}_c(k)$	estimated coefficients of a linear model defined within $\Omega_c^P(k)$
$\hat{f}(\mathbf{x}_i)$	predicted value at a neighboring pixel i within a prediction window $\Omega_c^P(k)$

3 Reconstruction using Local Linear Models

The ultimate goal of image reconstruction methods is to restore the ground truth image, $f(\mathbf{x})$, from an input image, y, generated by

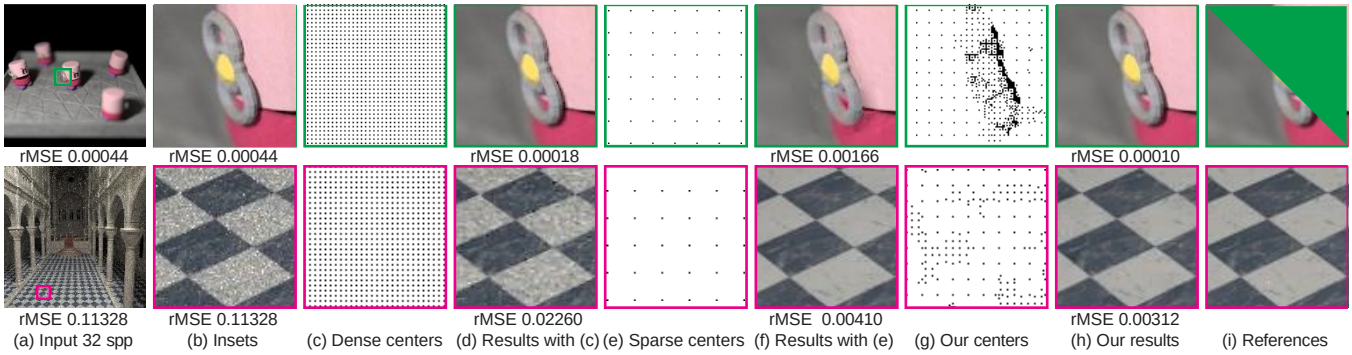


Figure 2: *Visualization of center pixels and their effects. Given uniformly sampled, i.e., 32 samples per pixel (spp), input images (a) and (b), we create our linear models at center pixels. The black dots in the image (c) indicate regularly, but densely selected center pixels with a fixed small prediction window. On the other hand, in (e), we choose regularly populated center pixels with a large fixed prediction window. When we use the small fixed prediction interval, we can preserve the detailed occlusion and features, but leave a lot of high-frequency noise (d). When using the large fixed interval, we can reduce noise well, but remove the high-frequency occlusion and features (f). On the other hand, our method adaptively selects the center pixels (g) and shows high-quality reconstruction results with lowest numerical errors.*

MC ray tracing, which is corrupted by MC noise, i.e., variance. Throughout the paper, we will use a subscript to represent the value of a function at a specific pixel. For example, $f(\mathbf{x}_i)$ and y_i denote a ground truth and input pixel value at pixel i, respectively. We summarize our notation in Table 1. To design an efficient, yet high-quality filtering technique, we propose a novel prediction based reconstruction method, which estimates the ground truth image $f(\mathbf{x})$ based on a sparse number of linear models.

Let us define a filtering window, Ω_c^F, centered at a center pixel, c. The window is considered as a set including all the pixels within the window. We also define a prediction window, $\Omega_c^P(k) \subseteq \Omega_c^F$, which has $k \equiv |\Omega_c^P(k)|$ pixels as its elements. Note that the filtering window Ω_c^F has a globally fixed size (e.g., 19×19). The prediction window $\Omega_c^P(k)$ where we predict the ground truth image $f(\mathbf{x})$ by a linear model, however, has variable size k. Within the prediction window, we define our linear model by using the first-order Taylor polynomial from the center pixel c:

$$f(\mathbf{x}_i) \approx f(\mathbf{x}_c) + \nabla f(\mathbf{x}_c)^T(\mathbf{x}_i - \mathbf{x}_c), \tag{1}$$

where $\mathbf{x}_i$ denotes a feature vector at pixel i. The ground truth value $f(\mathbf{x}_c)$ and its gradient $\nabla f(\mathbf{x}_c)$ are unknown, but can be estimated in the least squares sense, which minimizes the sum of squared residuals between the filtered image $\hat{f}(\mathbf{x})$ reconstructed by least squares and input image y. Once the estimated gradient, $\nabla \hat{f}(\mathbf{x}_c)$, is computed, we utilize it to predict the ground truth function $f(\mathbf{x}_i)$ at i pixels within the prediction window $\Omega_c^P(k)$, where i can be the center pixel c and even other pixels, i.e., $i \in \Omega_c^P(k)$, instead of fitting expensive linear models separately at other pixels within the window. Fig. 2 visualizes center pixels c where we build our linear models (black dots), and our method linearly predicts all the color values of other pixels i from a sparse number of linear models by utilizing the Taylor polynomial (Eq. 1). Even with a sparse number of linear models, we can appropriately reconstruct high-frequency details (e.g., noisy textures in the bottom row), especially when the details have a linear correlation with a rendering-specific feature (e.g., textures).

Our high-level idea of reconstructing multiple pixels within a prediction window by using a single linear model may be considered intuitive given Eq. 1. It introduces, however, novel technical challenges, since we should estimate an optimal, local prediction window, $\Omega_c^P(k_{opt})$, which minimizes our prediction error, $\xi_c(k) = \frac{1}{k} \sum_{i \in \Omega_c^P(k)} (\hat{f}(\mathbf{x}_i) - f(\mathbf{x}_i))^2$. In Fig. 2, we show the results computed by fixed prediction windows (e.g., small and large one), and these results show a noisy result or over-smoothed result. It motivates us to develop adaptive prediction sizes for high-quality reconstruction.

Technically, the prediction window $\Omega_c^P(k)$ corresponds to the interval, where we approximate the unknown functions $f(\mathbf{x}_i)$ by using the first-order Taylor polynomial (Eq. 1), and thus the predicted value $\hat{f}(\mathbf{x}_i)$ varies in terms of the size of the interval (i.e., prediction size $k \equiv |\Omega_c^P(k)|$). As a result, our first technical challenge is to efficiently construct multiple linear models with different k values, i.e., different intervals (Sec. 4.1). We then need to estimate their prediction errors, i.e., $\xi_c(k)$, depending on the size of k in order to select the optimal size k_{opt} that minimizes the error. The second technical challenge is that the optimal prediction size k_{opt} is still unknown even after computing multiple linear models, since the error $\xi_c(k)$ depends on the unknown term $f(\mathbf{x}_i)$. Therefore, we should find an estimated error $\hat{\xi}_c(k)$ and its corresponding estimated optimal prediction size $\hat{k}_{opt}$ (Sec. 4.2). To realize our high-level idea while tackling these challenges, we propose a novel algorithm to estimate the optimal prediction window $\Omega_c^P(k_{opt})$ in the subsequent section.

Linear Approximation using geometries. The linear approximation based on rendering-specific features $\mathbf{x}$ (e.g., data in the G-buffer) in Eq. 1 was previously studied for filtering Monte Carlo noise [Bauszat et al. 2011; Moon et al. 2014], but the previous methods do not fully utilize $\nabla f(\mathbf{x}_c)$ for reconstructing multiple pixels of $\Omega_c^P(k)$. Our method, however, reconstructs all pixels within the prediction window Ω_c^P from a linear model simultaneously, instead of performing a filtering at every pixel.

4 Linear Model Estimation

Our optimization goal is to estimate the optimal model defined as a linear model (e.g., first-order Taylor polynomial) computed within the optimal prediction size k_{opt}, which has a minimal prediction error $\xi_c(k_{opt})$. The optimization to calculate the prediction size k_{opt} can be formulated as follows:

$$k_{opt} = \underset{k}{\operatorname{argmin}}\, \xi_c(k) = \underset{k}{\operatorname{argmin}}\, \frac{1}{k} \sum_{i \in \Omega_c^P(k)} \left(\hat{f}(\mathbf{x}_i) - f(\mathbf{x}_i) \right)^2. \tag{2}$$

Note that our goal is to minimize the averaged squared difference between predicted values $\hat{f}(\mathbf{x}_i)$ and ground truth values $f(\mathbf{x}_i)$ over the pixels defined in the prediction window $\Omega_c^P(k)$, since we plan to predict all the pixels i in $\Omega_c^P(k)$ from a single linear model. We propose an iterative estimation process for efficiently computing linear models as a function of k in Sec. 4.1 and a recursive error analysis for computing $\hat{k}_{opt}$ in Sec. 4.2.

4.1 Recursive Reconstruction of Linear Models

Our linear model construction is the process of estimating the coefficients (i.e., intercept and gradient) of a linear function, the first-order Taylor polynomial, which correspond to the ground truth $f(\mathbf{x}_c)$ and its gradient $\nabla f(\mathbf{x}_c)$ in Eq. 1, given a prediction size (i.e., an interval of the first-order Taylor polynomial). To this end, we utilize the least squares problem to compute the optimal coefficients, which minimize the sum of squared residuals between observed noisy function y and filtered image $\hat{f}(\mathbf{x})$, i.e., $\sum_{i\in\Omega_c^P(k)}(\hat{f}(\mathbf{x}_i)-y_i)^2$. We define the estimated coefficients as a vector $\hat{\beta}_c(k) \equiv (\hat{f}(\mathbf{x}_c), \nabla\hat{f}(\mathbf{x}_c))$, which is the least squares estimator for the unknown vector $\beta_c(k) \equiv (f(\mathbf{x}_c), \nabla f(\mathbf{x}_c))$.

Given this least squares problem, we propose a recursive algorithm for computing the least squares solution $\hat{\beta}_c(k)$ within a prediction size k. To compute multiple linear models, each of which is constructed within a different prediction size k, one can apply the *normal equation*, $\hat{\beta}_c(k) = (X_k^T X_k)^{-1} X_k^T Y_k$, where X_k is $k\times(d+1)$ design matrix whose i-th row is set as $(1, (\mathbf{x}_i - \mathbf{x}_c)^T)$ and d is the length of the feature vector $\mathbf{x}_i$. The design matrix is filled with the features from k pixels. Analogously, each element in the vector $Y_k = (y_1, ..., y_k)^T$ is set with the intensities of k different pixels from the Monte Carlo input image y. We consider y_i as a 1D value, since we can apply our method to each channel independently for color images.

Unfortunately, the normal equation used in least squares based methods [Moon et al. 2014] commonly requires a matrix inversion, i.e., $(X_k^T X_k)^{-1}$, for each prediction size k. Furthermore, when we consider $|\Omega_c^F|$ candidates, which are computed by adding pixels one-by-one from the prediction window to the least-squared based reconstruction, we need to solve the normal equations $|\Omega_c^F|$ times. This is impractical, since this approach would require a prohibitive computational cost. Our main idea to avoid the expensive matrix inversion is to use the recursive least squares [Ljung and So¨derstro¨m 1987], which updates the inverse covariance matrix, $P_c(k) \equiv (X_k^T X_k)^{-1}$, incrementally without performing the matrix inversion.

Specifically, we update the inverse covariance matrix $P_c(k)$ and the corresponding linear model $\hat{\beta}_c(k)$ by using both $\mathbf{x}_k$ and y_k at the k-th pixel from the ones computed using prior $k-1$ pixels as follows:

$$
\begin{aligned}
G_c(k) &= \frac{P_c(k-1)\mathbf{z}_k}{1+\mathbf{z}_k^T P_c(k-1)\mathbf{z}_k},\\
P_c(k) &= P_c(k-1) - G_c(k)\mathbf{z}_k^T P_c(k-1),\\
\hat{\beta}_c(k) &= \hat{\beta}_c(k-1) + G_c(k)\left(y_k - \hat{\beta}_c^T(k-1)\mathbf{z}_k\right), \qquad (3)
\end{aligned}
$$

where $\mathbf{z}_k^T = (1, (\mathbf{x}_k - \mathbf{x}_c)^T)$ corresponds to the k-th row in the design matrix of the normal equation. The vector $G_c(k)$ can be considered as a weight allocated to a new sample pair, $\mathbf{x}_k$ and y_k, and the linear model $\hat{\beta}_c(k)$ is updated by considering a weighted a priori error $(y_k - \hat{\beta}_c^T(k-1)\mathbf{z}_k)$. Analogously, the inverse covariance matrix $P_c(k)$ is also updated by using the weight [1]. Technically, the computed linear model $\hat{\beta}_c(k)$ up to k pixels is the least squares solution, which minimizes the least squares cost function $\sum_{i\in\Omega_c^P(k)}(\hat{f}(\mathbf{x}_i) - y_i)^2$.

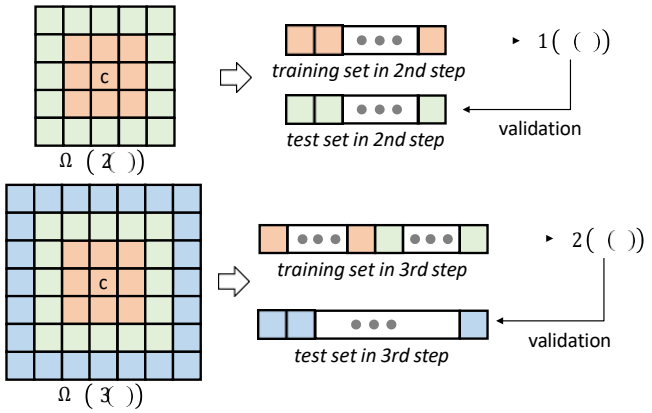


Figure 3: *We visualize how our method estimates prediction errors of linear models $\hat{\beta}_c(k(r))$, given r, (the floor integer of) the half of the width of the prediction window; $r = 2$ in the top and $r = 3$ in the bottom row. When increasing r of a square window from a prior step, we split pixels into two disjoint sets (i.e., training and test sets). We then estimate the error of a new r value and its new prediction window, by testing the prior model $\hat{\beta}_c(k(r-1))$ computed from its training set against newly included samples (i.e., the test set). This process is recursively performed for estimating the optimal prediction size k_{opt} that minimizes the prediction error.*

Note that the equation above is not an approximation of the batch solution, i.e., the normal equation, but is a recursive formula that is exactly derived based on the matrix inversion lemma (i.e., Woodbury matrix identity) from the batch version [Ljung and So¨derstro¨m 1987]. Its computational complexity for updating the matrix $P_c(k)$ and $\hat{\beta}_c(k)$ with k-th pixel is $O(d^2)$, where d is the length of the feature vector $\mathbf{x}_i$ containing a pixel position, normal, texture, and depth. Therefore, the complexity for computing all the linear models, each of which is constructed within a different prediction size k, is $O(|\Omega_c^F|d^2)$. We explain how to find an estimated optimal model $\hat{\beta}_c(k_{opt})$ in the next section.

4.2 Recursive Estimation of Prediction Error

In this section, we explain our process of choosing the optimal prediction size $\hat{k}_{opt}$ and its corresponding optimal linear model $\hat{\beta}_c(\hat{k}_{opt})$ among possible candidates. Given our recursive reconstruction (Sec. 4.1), we have iteratively computed multiple linear models $\hat{\beta}_c(k)$ as we grow its prediction size k. To select the optimal prediction size, we should estimate the prediction error $\xi_c(k)$ introduced when we predict k pixels by the linear model $\hat{\beta}_c(k)$. To choose the optimal prediction size in an efficient and robust manner, we propose a novel, iterative technique of estimating the prediction error $\xi_c(k)$ as a function of k.

A few techniques exist for estimating reconstruction errors such as Stein's unbiased risk estimator [Li et al. 2012] and estimated mean squared error based on the asymptotic expressions of weighted local regression [Moon et al. 2014]. Unfortunately, these prior techniques utilize the general error estimation tools developed in

[1] The recursive equations have a similar structure to the Kalman filter.

statistics for reducing the point error only at the center pixel, i.e., $(\hat{f}(\mathbf{x}_c)-f(\mathbf{x}_c))^2$. As a result, for our optimization goal (Eq. 2), we take a more aggressive approach and attempt to estimate the prediction error $\xi_c(k)$ defined in multiple pixels, since we plan to predict values of the k pixels in the prediction window $\Omega_c^P(k)$ based on a single linear model $\hat{\beta}_c(k)$.

Our high-level idea for addressing both accuracy and efficiency of evaluating the prediction error is to fully utilize the recursive least squares (Sec. 4.1), where we can naturally predict subsequent samples from previous samples. For example, given a linear model $\hat{\beta}_c^T(k-t)$ computed with $k-t$ pixels, we can estimate its prediction error in the next t steps as $(\hat{\beta}_c^T(k-t)\mathbf{z}_i-y_i)^2$ with newly added t samples before updating the linear model with those t samples. In this context, let us call the t pixels *a test set* and $k-t$ pixels *a training set*. Based on this idea, we propose a new iterative validation process, which estimates prediction errors by splitting pixels into the two disjoint sets. Furthermore, we design it to have a recursive form for high efficiency when considering k different linear models.

Given a $(2R+1)\times(2R+1)$ square filtering window Ω_c^F, we estimate the prediction error, when $k \in \{1^2, 3^2, 5^2, \ldots, (2R+1)^2\}$; such k values are chosen by increasing half of the width (or height), r, of our prediction window, and thus are computed based on a function of r. For computational efficiency, we consider only a subset of possible prediction size k, which is defined using the half of the width (or height), r, instead of taking all positive integers. As a result, we parameterize k by $k(r)$ and formulate our iterative validation approach into the following recursion:

$$\hat{\xi}_c(k(r)) = \frac{\hat{\xi}_c^{acc}(k(r))}{(2r+1)^2} = \frac{\hat{\xi}_c^{acc}(k(r-1)) + \Delta\hat{\xi}_c^{acc}(k(r))}{(2r+1)^2}, \quad (4)$$

where $\hat{\xi}_c^{acc}(k(r))$ is the accumulated prediction error, which needs to be normalized by its pixel count, $k(r) \equiv (2r+1)^2$. We decompose the error into two terms, accumulated error from $k(0)$ and to $k(r-1)$, $\hat{\xi}_c^{acc}(k(r-1))$, and the newly added error at the current r-th step, $\Delta\hat{\xi}_c^{acc}(k(r))$.

Given this recursion, we estimate the newly added error at r-th step $\Delta\hat{\xi}_c^{acc}(k(r))$ introduced when we increase the prediction size from $k(r-1)$ to $k(r)$, by using the following equation:

$$\Delta\hat{\xi}_c^{acc}(k(r)) = \sum_{i=1}^{\times} \left(\hat{\beta}_c^T(k(r-1))\,\mathbf{z}_i - y_i\right)^2, \quad (5)$$

where $\hat{\beta}_c(k(r-1))$ is the estimated linear model from $k(r-1)$ samples and these samples are defined as the training set of the r-th step in order to test newly included $8r \equiv k(r)-k(r-1)$ samples, i.e., the test set of the step. Fig. 3 illustrates how we iteratively split samples into training and test sets. We substitute our estimated prediction error $\hat{\xi}_c(k(r))$ (Eq. 4) for the unknown error $\xi_c(k(r))$ (Eq. 2), and then select the optimal prediction size $\hat{k}_{opt}$ and its corresponding linear model $\hat{\beta}_c(\hat{k}_{opt})$.

In Fig. 4, we compare our estimated error $\hat{\xi}_c(k(r))$ with its reference error $\xi_c(k(r))$. Also, we visualize our estimated optimal prediction size $\hat{k}_{opt}$ by using our estimated error $\hat{\xi}_c(k(r))$, with a reference prediction size k_{opt} computed from the reference error $\xi_c(k(r))$. For our visualization purpose, we compare our estimations with references for all pixels although we run our reconstruction on a sparse set of image pixels. We use a reference image generated by 8K ray samples per pixel, and then plug the reference values into Eq. 2. As a result, the reference optimal size is computed by minimizing the actual L^2 error between predicted images and its reference, which cannot be achieved in practice. Our prediction size shows a similar pattern with its reference and a very high correlation (e.g., 0.87). Around the key-spindles of the toasters in Fig. 4, our method with 19×19 prediction windows has relatively high prediction errors. In these regions, we need to use smaller prediction windows and allocate more ray samples to achieve high-quality results. This is addressed in the next section.

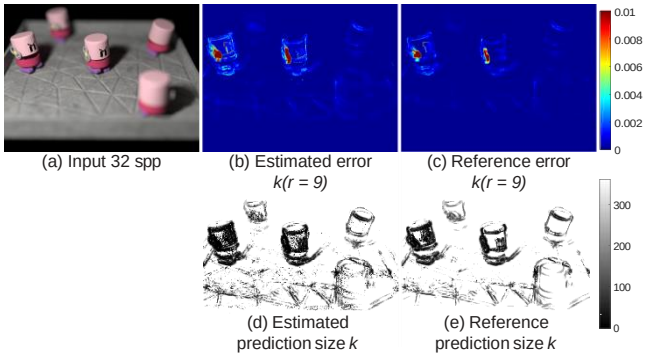


Figure 4: *We compare our estimated error and estimated optimal prediction size with a reference error and reference prediction size, respectively. Given a uniformly sampled input image (a), our estimated error (b) with a prediction size $k(r=9)$ shows a similar pattern compared to its reference (c). Also, our estimated opti- mal prediction size (d) has a strong correlation (e.g., 0.87) with the ground truth (e).*

5 Linear Model Construction and Adaptive Sampling

In this section, we introduce an algorithm to determine positions, i.e., center pixels c, of linear models (Sec. 5.1) and adaptive sampling process to guide additional ray samples on high error regions of our filtered image (Sec. 5.2).

5.1 Iterative Construction of Linear Models

We present a simple iterative algorithm to find center pixels c, where our local linear models (Sec. 3) are created by our recursive estimation process (Sec. 4). The computational complexity of a linear model estimation is $O(|\Omega_c^F|d^2)$, and thus our overall complexity for reconstructing the values of all pixels is $O(L|\Omega_c^F|d^2)$, where L is the number of linear models, i.e., the number of center pixels c. Ideally, L needs to be much smaller than the total pixel count of an input image y, while maintaining a high quality reconstruction.

On the first pass, we regularly select center pixels c by using a granularity factor g, which is initialized to a large one (e.g., width of filtering window Ω_c^F) along the X and Y directions in the screen space; for example, we choose a pixel as the center pixel c, whose x and y positions are multiples of the factor g. After we decide the center pixels, we estimate an optimal linear model within its optimal prediction size k_{opt} per each center pixel c, and then predict k_{opt} pixels from each model.

In the second pass, we reduce the global granularity factor (e.g., $g/2$), and test the pixels whose positions are multiples of $g/2$ to see whether or not each newly tested pixel is predicted by existing linear models constructed in the prior pass. If it is not reconstructed by prior prediction, mainly because of small $\hat{k}_{opt}$ values caused by drastic illumination changes, we create a new center pixel c on