Sparse Sampling for Real-time Ray Tracing

Timo Viitanen, Matias Koskela, Kalle Immonen,
Markku Mäkitalo, Pekka Jääskeläinen and Jarmo Takala
Tampere University of Technology, Tampere, Finland

Keywords: Ray Tracing, Sparse Sampling, Image Reconstruction.

Abstract: Ray tracing is an interesting rendering technique, but remains too slow for real-time applications. There are various algorithmic methods to speed up ray tracing through uneven screen-space sampling, e.g., foveated rendering where sampling is directed by eye tracking. Uneven sampling methods tend to require at least one sample per pixel, limiting their use in real-time rendering. We review recent work on image reconstruction from arbitrarily distributed samples, and argue that these will play major role in the future of real-time ray tracing, allowing a larger fraction of samples to be focused on regions of interest. Potential implementation approaches and challenges are discussed.

1 INTRODUCTION

Ray tracing has been long regarded in the graphics community as a potential real-time rendering technique of the future (Keller et al., 2013). There has been extensive research on real-time ray tracing, ranging from algorithms and implementations, to applications, to hardware architectures. Recently, GPU ray tracing performance has improved to the point that it is widely used to render feature film *computer graphics* (CG) (Keller et al., 2015), but still falls short of competing with conventional rasterization methods in real-time rendering. A high-end GPU ray tracer can comfortably render *primary rays*, or simple *Whitted-style ray tracing* (Whitted, 1979) in real time, usually defined as a frame rate of 30 Hz or more. However, interesting visual effects such as soft shadows, glossy reflections and indirect illumination require distribution ray tracing, where several rays are traversed based on a random distribution and averaged together. Distribution effects remain challenging to render in real time.

There is a promising set of algorithm-level techniques for reducing the computational effort of ray tracing, based on uneven sample distributions in screen space, which concentrate the rendering effort on regions of interest. These include

- adaptive sampling, where more samples are placed on regions of the screen with, e.g., high Monte Carlo variance (Zwicker et al., 2015),
- foveated rendering, where rendering effort is concentrated near the user's fovea by means of eye tracking hardware (Weier et al., 2016; Koskela et al., 2016), and
- sampling based on visual saliency, i.e., the ability of features to draw the attention of a human observer (Longhurst et al., 2006).

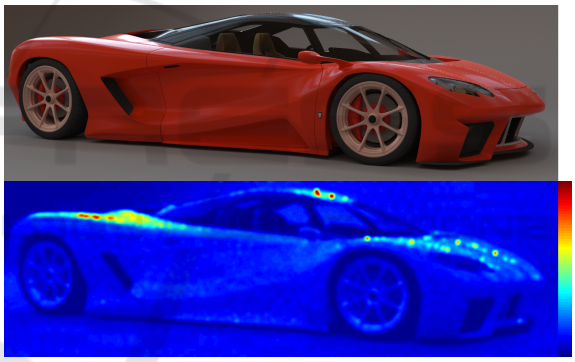


Figure 1: Path traced reference image (top) and error after 16 samples per pixel (bottom; red: high error, blue: low error).

The example in Fig. 1 illustrates that error is often concentrated on small areas of the image, therefore, performance can be significantly improved by adaptively focusing samples on these areas. Moreover, there is an extensive literature on noise-reduction methods (Zwicker et al., 2015) for distribution ray tracing, which are often combined with adaptive sampling for the best results.

Currently, a basic difficulty in speeding up real-

Viitanen, T., Koskela, M., Immonen, K., Mäkitalo, M., Jääskeläinen, P. and Takala, J.
Sparse Sampling for Real-time Ray Tracing.
DOI: 10.5220/0006655802950302
In *Proceedings of the 13th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2018) - Volume 1: GRAPP*, pages 295-302
ISBN: 978-989-758-287-5

time rendering with the above methods is that most of them depend on having at least one sample per screen pixel. This already takes up most of the compute performance of a high-end GPU with state of the art real-time ray tracing methods (Schied et al., 2017; Chaitanya et al., 2017; Mara et al., 2017). Consequently, there is little rendering performance left to distribute additional samples to regions of interest. Intuitively, rendered images often have smooth regions whose lighting could be reconstructed from sparse samples, but the literature has not yet converged on a best method to do so.

Some of the methods tried so far have significant limitations. For instance, two recent state of the art real-time ray tracers—one based on denoising (Schied et al., 2017), and the other on foveated rendering (Weier et al., 2016)—rely on reprojecting samples from previous animation frames to the current frame. Reprojection makes real-time ray tracing tractable by cheaply increasing the amount of available samples, but has difficulties with animated scenes and fast camera movements, as well as reflections and glossy surfaces. Other methods are prohibitively expensive for real-time use, e.g., those based on compressive sensing (Sen and Darabi, 2011).

The problem of reconstructing a grid image from sparse samples has been studied in *digital signal processing* (DSP) literature, where it is applicable to, e.g., image warping and stitching. We argue that applying DSP methods is a promising future direction for real-time ray tracing. For best results, sparse sampling methods might be modified to take advantage of extra information that is available in a ray tracer, e.g., depth, normal, and albedo data. In this paper, we examine the problem of rendering with uneven, sparse samples and review techniques in the DSP and computer graphics literature that could be applied to the problem.

This paper is structured as follows. Section 2 reviews literature on methods to resample sparse data into grid images. Section 3 summarizes recent work on real-time ray tracing and its limitations. Section 4 discusses potential approaches to integrate sparse sampling into real-time ray tracers. Section 5 concludes the paper.

2 SPARSE SAMPLING METHODS

Image interpolation and reconstruction are well understood when the source image samples are placed in a regular grid. When the samples are at irregular positions, these tasks are less well studied. There are several applications for resampling of this kind, including, e.g., super-resolution, image compression and stereo rectification. The proposed methods are often aimed at specific applications and place special constraints on their inputs, e.g., the samples may be constrained to a specific density, or into a regular grid with small perturbations.

For adaptive sampling, we are interested in methods that permit input samples to be sparse and nonuniformly distributed, as opposed to sampling on a regular grid. This section focuses on reviewing methods evaluated with inputs with the above properties. Moreover, we omit some classes of methods which are known to be highly expensive, e.g., exemplar-based interpolation (Facciolo et al., 2009) and compressive sensing (Sen and Darabi, 2011).

A basic approach is Delaunay triangulation between samples followed by polynomial interpolation inside each triangle—this can be nearest-neighbor, bilinear or bicubic interpolation. Another simple approach is *inverse distance weighting* (IDW), where pixel colors are averages of nearby samples weighted with the inverse distance to the pixel center. In *natural neighbor interpolation* (Boissonnat and Cazals, 2000) a pixel's color is interpolated from samples that are its natural neighbors, i.e., the samples whose volumes are chopped when the pixel is inserted to a Delaunay triangulation of the samples. Parallel implementations of these algorithms typically require syncing operations or special datastructures, which might make them out of reach in typical real-time time budgets.

Another classic direction of work has been to assume that the sampled signal is band-limited, and extend Shannon's sampling theory to irregularly sampled inputs (Strohmer, 1997). In practice, the reconstruction quality in band-limited methods drops if the band-limited assumption does not hold (Seiler et al., 2015).

2.1 Model Fitting

Several methods attempt to fit models such as splines (Vazquez et al., 2005) to the observed samples. The output image can then be sampled from the model.

A recent generalization of model fitting is *generalized morphological component analysis* (GMCA) (Fadili et al., 2010), where images are represented as combinations of multiple models. For example, the countours of an image might be represented as curvelets and repeating small-scale texture in frequency form.

2.2 Variational Optimization

A set of methods take a variational approach where the parameters in model fitting are formulated as an optimization problem. The optimized cost function consists of an error term comparing the input samples to the model, and a *regularization term* to prefer solutions with properties similar to natural images. An often exploited property is the redundancy typically present in natural images, with the regularization term promoting a sparse representation, for example in a wavelet domain (Elad and Aharon, 2006).

One class of variational methods which is known to work well with sparse samples uses a model with splines placed on a grid (Arigovindan et al., 2005). Interestingly, variational reconstruction was recently extended to handle generalized, e.g., noisy or blurred, samples (Bourquard and Unser, 2013).

2.3 Kernel Regression

In the model-fitting technique *kernel regression* (KR) (Takeda et al., 2006), the model places kernel functions, e.g., splines, at each sample and finds their parameters locally via least squares optimization. In contrast, the above spline-based methods tend to place splines in a regular grid and perform global optimization. The technique can be viewed as a generalization of, e.g., bilateral filtering. Interestingly, KR can be used for both denoising and resampling.

2.4 Frequency Selective Reconstruction

Downsampling a signal in a regular pattern reflects frequencies below the Nyquist frequency into lower frequencies, corrupting the spectrum. A recent, high-quality method of *frequency selective reconstruction* (FSR) (Seiler et al., 2015) is based on an observation that downsampling in an irregular pattern has, conversely, the effect of adding noise to the spectrum. A good approximation of the high-frequency signal can be obtained by extracting the dominant frequencies one by one. FSR has been recently improved by adapting the number of iterations per block based on local texture (Jonscher et al., 2016), and extending the method to handle samples placed at non-integer positions (Koloda et al., 2017).

2.5 Super-Resolution Methods

In the task of multi-frame *super-resolution* (SR), multiple low-quality frames, taken from slightly different positions and directions, are combined into a single high-quality image with a better resolution. SR has been extensively studied, and there are recent surveys (Tian and Ma, 2011). One popular approach to SR, dubbed *interpolation-based SR* in Ref. (Tian and Ma, 2011), works by first *registering* samples from all low-resolution images to a single coordinate frame, interpolating between them to form a high-resolution image, and deblurring the result. In SR methods that adhere to this model, the interpolation component is directly applicable to the resampling problem. SR methods are of interest since they often take into account significant noise in the input images.

2.6 Denoising-based Reconstruction

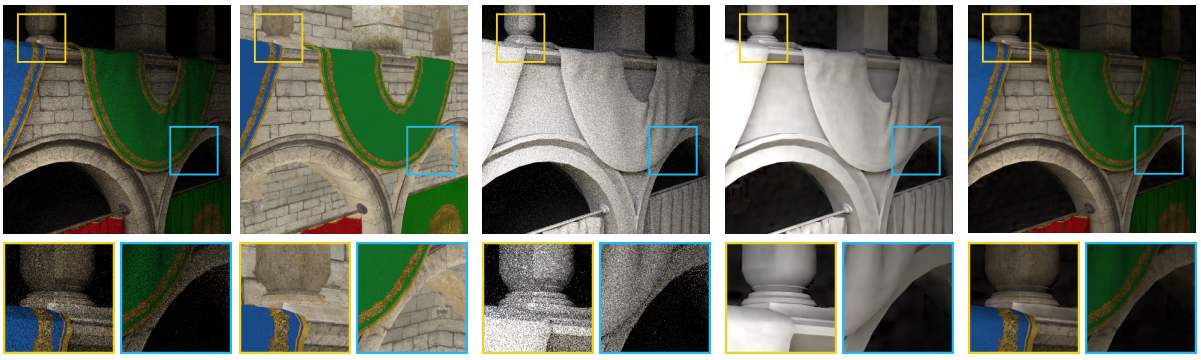
The above interpolation methods all give some error compared to an ideal image. In denoising-based reconstruction (Koloda et al., 2015), the error is treated as noise, and standard denoising algorithms are applied to remove it. A reliability metric is computed per pixel and used to adjust denoising strength. In a later work (Koloda et al., 2016), the reliability metric is improved to ca. double the image quality improvement.

2.7 Computer Graphics Methods

Sampling has been extensively studied in the computer graphics literature. Research is especially active in the efficient generation of sample patterns with desirable properties, e.g., blue-noise samples via Poisson disc sampling (Ebeida et al., 2011). Works that render images from sparse, nonuniform samples are less common. As in the DSP literature, proposed upsampling methods are often limited to regular grids (Herzog et al., 2010; Yang et al., 2008).

A very fast approach is to sample a low-resolution grid, subdivide it in regions where a higher sample rate is wanted, and fill in missing samples with linear interpolation (Steinberger et al., 2012; Kim et al., 2016; Lee et al., 2016). Steinberger et al. (2012) insert subdivisions in a diamond-square pattern based on a visual saliency model. The adaptive methods of Kim et al. (2016) and Lee et al. (2016) first sample the low-resolution grid, and then compute similarity measures between neighboring pixels to decide whether to sample or interpolate the intermediate value.

Another fast method is pull-push rendering (Gortler et al., 1996), recently used as an inpainting method for foveated rendering (Stengel et al., 2016). In pull-push the input samples are splatted onto a grid image and *pulled* onto a series of low-resolution approximations, which are then *pushed* back to fill in unknown regions of the original image. Advantages of pull-push are that it is fast,



(a) Noisy input (b) Albedo (c) Irradiance (d) Filtered irradiance (e) Final image

Figure 2: Use of auxiliary data in ray tracing reconstruction. The noisy input (a) is decoupled into albedo (b) and irradiance (c). Denoised irradiance (d) is recoupled with albedo to form the final image (e).

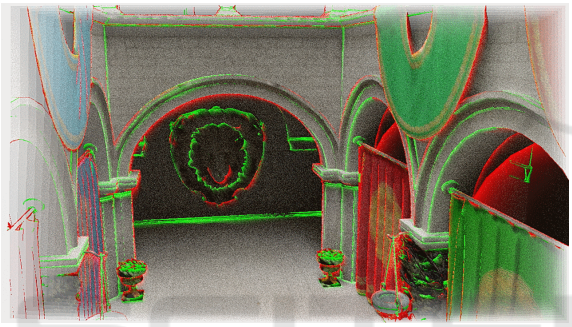


Figure 3: Temporal accumulation of samples over multiple frames in a scene flythrough animation. Accumulation uses depth data to reproject the previous frame to a new frame's camera. Some samples would blur the result and, therefore, they are rejected based on auxiliary data (green: depth, red: shading normal). In addition, some parts of the geometry were outside the image in previous frames, so accumulated samples are unavailable (white).

works with irregular sampling, and is able to fill the image from arbitrarily sparse samples.

Multi-pass *normalized convolution* has also been used to inpaint in the unknown regions of a sparsely sampled image (Galea et al., 2014). In *adaptive frameless rendering* (Dayal et al., 2005), samples are unevenly distributed in both space and time, and images are reconstructed with a spatio-temporal convolutional filter.

Some sparse sampling methods have been proposed for rasterization pipelines, in order to reduce expensive shader executions. An interesting recent technique is *coarse pixel shading* (CPS) (Vaidyanathan et al., 2014), where rasterized primitives are divided into disjoint pixel sets, and lighting is computed once per set. An advantage of CPS is that it preserves object silhouette edges while allowing uneven sample spacing.

Nonuniform samples occur naturally when rendering scenes represented as point clouds. Proposed point cloud renderers use screen-space interpolation or reconstruction techniques to fill in a smooth surface between points, such as pull-push rendering (Marroquim et al., 2008), pixel splatting (Zwicker et al., 2001) and anisotropic filtering (Pintus et al., 2011).

It should be noted that the aforementioned techniques are all evaluated with noise-free samples, though the method of Galea et al. (2014) may also have denoising properties.

3 REAL-TIME RAY TRACING

Classically, real-time ray tracing was restricted to simple visual effects, as complex distribution effects require many noisy samples to converge. Recently, there has been significant progress in adaptive sampling, and the reconstruction of high-quality images from noisy samples, both documented in a recent survey paper (Zwicker et al., 2015).

This section examines methods used in a recent batch of real-time ray tracers with complex visual effects, either based on gaze tracking (Weier et al., 2016) or denoising reconstruction (Schied et al., 2017; Chaitanya et al., 2017; Mara et al., 2017).

3.1 Noise Filtering

Mara et al. (2017) use performance-optimized bilateral filters, while Schied et al. (2017) use a hierarchical wavelet transform, similar in principle to model fitting methods for sparse resampling discussed above. Larger filter kernels are used in noisy regions.

Chaitanya et al. (2017) train a recurrent autoencoder based on a *convolutional neural network* (CNN) to reconstruct lighting based on a window of

noisy samples. The approach is distinct from other recent machine learning denoisers (Kalantari et al., 2015; Bako et al., 2017), which use conventional filters for denoising and determine the filter parameters by means of machine learning methods.

3.2 Auxiliary Data

Denoising a rendered image is different from the general image processing problem since various noise-free auxiliary scene data can be cheaply provided to the renderer, e.g., depth, normal, and albedo buffers. The recent real-time ray tracers take advantage of such data in two main ways. The first is to separate the image into irradiance and albedo components, and focus on reconstructing the irradiance, as shown in Fig. 2. This significantly simplifies the denoising problem by, e.g., preventing any blurring of texture details. Mara et al. (2017) split irradiance into matte and glossy components, which are filtered separately, while Schied et al. (2017) similarly separate direct and indirect lighting. Secondly, noise filtering is directed to avoid blurring across edges in the auxiliary buffers.

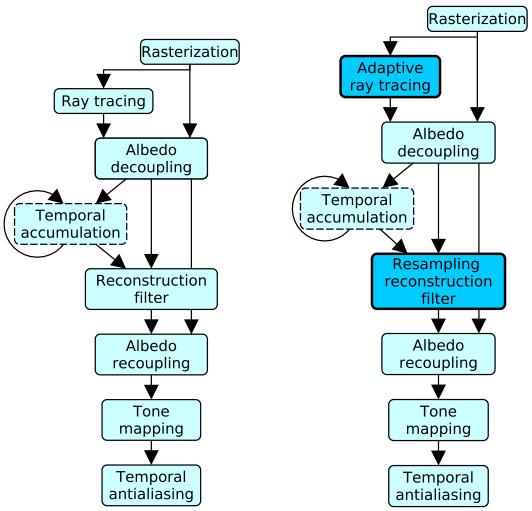
3.3 Temporal Accumulation

Path traced samples are sufficiently noisy that reconstruction from one sample per pixel is challenging. Most of the recent real-time implementations (Weier et al., 2016; Schied et al., 2017; Mara et al., 2017) rely on temporal accumulation from multiple frames to improve the effective sample count. In this approach, samples from previous frames are reprojected to the current frame, as shown in Fig. 3. Extra samples may be taken to fill image regions that were, e.g., occluded in the previous frame. As a drawback, dynamic scenes are difficult to handle.

The recurrent autoencoder proposed by Chaitanya et al. (2017) takes a different approach of including recurrent connections in the CNN model, which pass hidden data between animation frames. Moreover, two methods (Chaitanya et al., 2017; Schied et al., 2017) also employ screen-space temporal antialiasing (Karis, 2014) which is used in computer game engines.

3.4 Nonuniform Sampling

Out of the reviewed works, only Weier et al. (2016) use nonuniform sampling based on eye tracking. This is done by ensuring via temporal accumulation that there is at least one sample per pixel, and placing additional samples in the foveal region.



(a) Conventional real-time ray tracing system (b) Adaptive ray tracing

Figure 4: Modifications to adapt a conventional real-time ray tracer (Schied et al., 2017; Mara et al., 2017; Chaitanya et al., 2017) to use adaptive sampling.

In contrast, some previous works have rendered images with sparse nonuniform sampling at interactive or real-time rates (Shevtsov et al., 2010; Galea et al., 2014; Lee et al., 2016), but, to our best knowledge, only in the easier case of noise-free samples obtained via Whitted-style ray tracing or rasterization.

4 DISCUSSION

The above sections reviewed the rich recent literature on image resampling from scattered samples to a grid, as well as state of the art real-time ray tracers. A potential low-hanging fruit in real-time ray tracing is to incorporate adaptive sampling techniques so that, e.g., large flat surfaces receive less than one sample per pixel. The main technical challenge lies in integrating denoising with sparse sampling. There are multiple potential approaches toward this goal:

1. Extension of ray tracing denoising methods to handle scattered input samples.
2. Extension of sparse resampling methods to perform denoising.
3. Sparse resampling followed by denoising.

In each case, sparse resampling would be integrated to the reconstruction component of a ray tracer as shown in Fig. 4.

Out of the recent real-time ray tracers, two methods (Schied et al., 2017; Mara et al., 2017) perform

denoising with bilateral filters, the former using the filters as a low-level implementation technique for hierarchical wavelet reconstruction. One method (Chaitanya et al., 2017) uses a recurrent autoencoder. To our best knowledge, there is so far no analysis in the literature on adapting these methods to sparse inputs. Bilateral filtering appears conceptually difficult to adapt since it incorporates radiometric distance between the target sample and nearby source samples, which is unavailable if the target pixel was not sampled—though this is not an issue if only auxiliary data is used for edge stopping. Kernel regression (Takeda et al., 2007) can be viewed as a generalization of bilateral filtering which handles sparse inputs.

It may be easier to adapt resampling algorithms, which often act as denoising filters, to perform the reconstruction step. To this end, the resampling algorithm should be adapted to use auxiliary data for kernel size adjustment and edge stopping. In order to be useful, a resampling method should be able to run in real time, handle nonuniformly distributed inputs, and preferably be able to cope with noise in the inputs. Empirical evaluation is needed to determine which methods are suitable. Based on a literature survey, some interesting candidates in recent work are frequency selective reconstruction (Seiler et al., 2015), kernel regression (Takeda et al., 2006) and variational interpolation (Bourquard and Unser, 2013).

The most straightforward approach is to perform resampling and denoising as separate steps. Careful analysis of noise behavior is then needed to ensure that resampling does not harm denoising performance. Similar requirements apply to the resampling method as above.

A separate question is how to distribute the adaptive samples. Offline adaptive ray tracers most often sample based on Monte Carlo variance from several previous samples (Zwicker et al., 2015). Variance might be computed based on temporal accumulation as is done in recent work (Schied et al., 2017) to determine denoising filter kernel sizes. Typically, more samples are needed near edges and fine features in the image (Overbeck et al., 2009), so a rasterized depth image could help guide sampling as shown in Fig. 4. Other auxiliary data might also be used in the spirit of Lee et al. (2016).

5 CONCLUSION

This position paper argued that a way forward for real-time ray tracing is to incorporate heavily adaptive sampling, placing less than one sample per pixel in areas that are easy to render, and performing image reconstruction from sparse data. There is a rich literature on methods to resample from sparse samples into a grid image, which might be integrated into ray tracing systems. Three possible implementation approaches were discussed: extending denoising methods to take sparse inputs, extending resampling methods to function as ray tracing oriented denoising methods, and performing resampling and denoising in separate steps.

ACKNOWLEDGEMENTS

We would like to thank Frank Meinl for the Sponza model and Yasutoshi Mori for the Sports Car model (released under CC BY 4.0 license: https://creativecommons.org/licenses/by/4.0/)

This work is supported by Tekes — the Finnish Funding Agency for Innovation under the funding decision 40142/14 (StreamPro) in the FiDiPro program.

REFERENCES

Arigovindan, M., Suhling, M., Hunziker, P., and Unser, M. (2005). Variational image reconstruction from arbitrarily spaced samples: A fast multiresolution spline solution. *IEEE Transactions on Image Processing*, 14(4):450–460.

Bako, S., Vogels, T., Mcwilliams, B., Meyer, M., NováK, J., Harvill, A., Sen, P., Derose, T., and Rousselle, F. (2017). Kernel-predicting convolutional networks for denoising monte carlo renderings. *ACM Transactions on Graphics*, 36(4):97.

Boissonnat, J.-D. and Cazals, F. (2000). Smooth surface reconstruction via natural neighbour interpolation of distance functions. In *Proceedings of the Annual Symposium on Computational Geometry*, pages 223–232.

Bourquard, A. and Unser, M. (2013). Anisotropic interpolation of sparse generalized image samples. *IEEE Transactions on Image Processing*, 22(2):459–472.

Chaitanya, C. R. A., Kaplanyan, A. S., Schied, C., Salvi, M., Lefohn, A., Nowrouzezahrai, D., and Aila, T. (2017). Interactive reconstruction of Monte Carlo image sequences using a recurrent denoising autoencoder. *ACM Transactions on Graphics*, 36(4):98.

Dayal, A., Woolley, C., Watson, B., and Luebke, D. (2005). Adaptive frameless rendering. In *Proceedings of the Eurographics Conference on Rendering Techniques*, pages 265–275.

Ebeida, M. S., Davidson, A. A., Patney, A., Knupp, P. M., Mitchell, S. A., and Owens, J. D. (2011). Efficient maximal poisson-disk sampling. *ACM Transactions on Graphics*, 30(4):49.

Elad, M. and Aharon, M. (2006). Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Transactions on Image Processing*, 15(12):3736–3745.

Facciolo, G., Arias, P., Caselles, V., and Sapiro, G. (2009). Exemplar-based interpolation of sparsely sampled images. In *Proceedings of the International Conference on Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 331–344.

Fadili, M. J., Starck, J.-L., Bobin, J., and Moudden, Y. (2010). Image decomposition and separation using sparse representations: an overview. *Proceedings of the IEEE*, 98(6):983–994.

Galea, S., Debattista, K., and Spina, S. (2014). GPU-based selective sparse sampling for interactive high-fidelity rendering. In *Proceedings of the International Conference on Games and Virtual Worlds for Serious Applications*, pages 1–8.

Gortler, S. J., Grzeszczuk, R., Szeliski, R., and Cohen, M. F. (1996). The lumigraph. In *Proceedings of the Annual Conference on Computer Graphics and Interactive Techniques*, pages 43–54.

Herzog, R., Eisemann, E., Myszkowski, K., and Seidel, H.-P. (2010). Spatio-temporal upsampling on the GPU. In *Proceedings of the ACM SIGGRAPH symposium on Interactive 3D Graphics and Games*, pages 91–98.

Jonscher, M., Seiler, J., and Kaup, A. (2016). Texture-dependent frequency selective reconstruction of non-regularly sampled images. In *Proceedings of the Picture Coding Symposium*, pages 1–5.

Kalantari, N. K., Bako, S., and Sen, P. (2015). A machine learning approach for filtering Monte Carlo noise. *ACM Transactions on Graphics*, 34(4):122–1.

Karis, B. (2014). Advances in real-time rendering in games. *SIGGRAPH Courses*, 1.

Keller, A., Fascione, L., Fajardo, M., Georgiev, I., Christensen, P. H., Hanika, J., Eisenacher, C., and Nichols, G. (2015). The path tracing revolution in the movie industry. In *SIGGRAPH Courses*, pages 24–1.

Keller, A., Karras, T., Wald, I., Aila, T., Laine, S., Bikker, J., Gribble, C., Lee, W.-J., and McCombe, J. (2013). Ray tracing is the future and ever will be... In *SIGGRAPH Courses*.

Kim, Y., Seo, W., Kim, Y., Lim, Y., Nah, J.-H., and Ihm, I. (2016). Adaptive undersampling for efficient mobile ray tracing. *The Visual Computer*, 32(6-8):801–811.

Koloda, J., Seiler, J., and Kaup, A. (2015). Denoising-based image reconstruction from pixels located at non-integer positions. In *Proceedings of the IEEE International Conference on Image Processing*, pages 4565–4569.

Koloda, J., Seiler, J., and Kaup, A. (2016). Reliability-based mesh-to-grid image reconstruction. In *Proceedings of the IEEE International Workshop on Multimedia Signal Processing*, pages 1–5.

Koloda, J., Seiler, J., and Kaup, A. (2017). Frequency selective mesh-to-grid resampling for image communication. *IEEE Transactions on Multimedia*, 19:1689–1701.

Koskela, M., Viitanen, T., Jääskeläinen, P., and Takala, J. (2016). Foveated path tracing. In *Proceedings of the International Symposium on Visual Computing*, pages 723–732.

Lee, W.-J., Hwang, S. J., Shin, Y., Ryu, S., and Ihm, I. (2016). Adaptive multi-rate ray sampling on mobile ray tracing GPU. In *SIGGRAPH ASIA 2016 Mobile Graphics and Interactive Applications*, page 3.

Longhurst, P., Debattista, K., and Chalmers, A. (2006). A GPU based saliency map for high-fidelity selective rendering. In *Proceedings of the International Conference on Computer Graphics, Virtual Reality, Visualisation and Interaction in Africa*, pages 21–29.

Mara, M., McGuire, M., Bitterli, B., and Jarosz, W. (2017). An efficient denoising algorithm for global illumination. In *Proceedings of High Performance Graphics*, page 3.

Marroquim, R., Kraus, M., and Cavalcanti, P. R. (2008). Efficient image reconstruction for point-based and line-based rendering. *Computers & Graphics*, 32(2):189–203.

Overbeck, R. S., Donner, C., and Ramamoorthi, R. (2009). Adaptive wavelet rendering. *ACM Transactions on Graphics*, 28(5):140–1.

Pintus, R., Gobbetti, E., and Agus, M. (2011). Real-time rendering of massive unstructured raw point clouds using screen-space operators. In *Proceedings of the International Symposium on Virtual Reality, Archaeology and Intelligent Cultural Heritage*.

Schied, C., Kaplanyan, A., Wyman, C., Patney, A., Chaitanya, C. R. A., Burgess, J., Liu, S., Dachsbacher, C., Lefohn, A., and Salvi, M. (2017). Spatiotemporal variance-guided filtering: real-time reconstruction for path-traced global illumination. In *Proceedings of High Performance Graphics*, page 2.

Seiler, J., Jonscher, M., Schöberl, M., and Kaup, A. (2015). Resampling images to a regular grid from a non-regular subset of pixel positions using frequency selective reconstruction. *IEEE Transactions on Image Processing*, 24(11):4540–4555.

Sen, P. and Darabi, S. (2011). Compressive rendering: A rendering application of compressed sensing. *IEEE Transactions on Visualization and Computer Graphics*, 17(4):487–499.

Shevtsov, M., Letavin, M., and Rukhlinskiy, A. (2010). Low cost adaptive anti-aliasing for real-time ray-tracing. In *Proceedings of the International Conference on Computer Graphics and Vision*, volume 10, pages 45–49.

Steinberger, M., Kainz, B., Hauswiesner, S., Khlebnikov, R., Kalkofen, D., and Schmalstieg, D. (2012). Ray prioritization using stylization and visual saliency. *Computers & Graphics*, 36(6):673–684.

Stengel, M., Grogorick, S., Eisemann, M., and Magnor, M. (2016). Adaptive image-space sampling for gaze-contingent real-time rendering. In *Computer Graphics Forum*, volume 35, pages 129–139.

Strohmer, T. (1997). Computationally attractive reconstruction of bandlimited images from irregular samples. *IEEE Transactions on Image Processing*, 6(4):540–548.

Takeda, H., Farsiu, S., and Milanfar, P. (2006). Robust kernel regression for restoration and reconstruction of images from sparse noisy data. In *IEEE International Conference on Image Processing*, pages 1257–1260.

Takeda, H., Farsiu, S., and Milanfar, P. (2007). Kernel regression for image processing and reconstruction. *IEEE Transactions on Image Processing*, 16(2):349–366.

Tian, J. and Ma, K.-K. (2011). A survey on super-resolution imaging. *Signal, Image and Video Processing*, 5(3):329–342.

Vaidyanathan, K., Salvi, M., Toth, R., Foley, T., Akenine-Möller, T., Nilsson, J., Munkberg, J., Hasselgren, J., Sugihara, M., Clarberg, P., et al. (2014). Coarse pixel shading. In *Proceedings of High Performance Graphics*, pages 9–18.

Vazquez, C., Dubois, E., and Konrad, J. (2005). Reconstruction of nonuniformly sampled images in spline spaces. *IEEE Transactions on Image Processing*, 14(6):713–725.

Weier, M., Roth, T., Kruijff, E., Hinkenjann, A., Pérard-Gayot, A., Slusallek, P., and Li, Y. (2016). Foveated real-time ray tracing for head-mounted displays. *Computer Graphics Forum*, 35(7):289–298.

Whitted, T. (1979). An improved illumination model for shaded display. In *ACM SIGGRAPH Computer Graphics*, volume 13, page 14.

Yang, L., Sander, P. V., and Lawrence, J. (2008). Geometry-aware framebuffer level of detail. *Computer Graphics Forum*, 27(4):1183–1188.

Zwicker, M., Jarosz, W., Lehtinen, J., Moon, B., Ramamoorthi, R., Rousselle, F., Sen, P., Soler, C., and Yoon, S.-E. (2015). Recent advances in adaptive sampling and reconstruction for Monte Carlo rendering. *Computer Graphics Forum*, 34(2):667–681.

Zwicker, M., Pfister, H., Van Baar, J., and Gross, M. (2001). Surface splatting. In *Proceedings of the Annual Conference on Computer Graphics and Interactive Techniques*, pages 371–378.