To appear in ACM TOG 33(4) (Proc. of SIGGRAPH 2014).

Supplementary Material: The Natural-Constraint Representation of the Path Space for Efficient Light Transport Simulation

Anton S. Kaplanyan* Johannes Hanika* Carsten Dachsbacher*
Karlsruhe Institute of Technology

1 Implementation Details

We implemented our algorithm as a new mutation strategy in Mitsuba [Jakob 2010], to be used in the path space MLT framework. Compared to a naive implementation, the proposed implementation avoids a lot of unnecessary computational overhead. To this end, we provide a high level overview of our implementation as well as details on the required matrix operations.

Computing Ray Differentials in Half Vector Space In this step it is crucial to avoid a slow full matrix inversion. We take advantage of the sparse nature of the matrix J and use a stripped-down version of LU decomposition to solve only for the blocks D_i from Eq. 9 in the paper. Since we would need to solve two times (once for each basis vector of the 2D ray differential) we directly insert a 2×2 matrix H_i for each half vector, and then start the solve algorithm for block-tridiagonal matrices. Figure 1 shows pseudo code for this. The matrices A, B, and C in the listing are the matrices from Eq. 11 in the paper's appendix.

Solving for Vertex Offsets In Sec. 5.1 of the paper we show how to compute vertex offsets as $\Delta\mathbf{X} = J^{-1}\Delta\mathbf{H}^{\perp}$ in the prediction step of the Newtonian method. Again, there is no full matrix inversion required, and we employ the procedure outlined in Figure 2, derived from the LU decomposition to compute $\Delta\mathbf{X}$.

```
compute_ray_differentials(X):                          (Eq. 9 in the paper)
    // compute temporary A'_i, U_i, Λ^{-1}
    Λ^{-1} = B_0^{-1}
    for i = {1, ··· , k − 1}
        A'_i = A_i · Λ^{-1}
        U_i = Λ^{-1} · C_{i−1}
        Λ^{-1} = (B_i − A'_i · C_{i−1})^{-1}

    // compute all D_i
    for i = {1, ··· , k − 1}
        // input matrices H = 0, only H_i = I
        H = I
        for j = {i + 1, ··· , k − 1} :
            H = H − A'_j · H

        D = Λ^{-1} · H
        D_i^{-1} = D^{-1}

    // compute transfer matrix T_1
    T_1 = −Λ^{-1} · C_{k−1}
    for i = {k − 2, ··· , 1}
        T_1 = −U_i · T_1
```

Figure 1: *Computing ray differentials in half vector space from the constraint derivative matrix J. H is a 2×2 matrix in this listing.*

*e-mail:{anton.kaplanyan|hanika|dachsbacher}@kit.edu

```
solve_matrix_h_to_x(A_i, B_i, C_i, Δh^⊥):
    // compute temporary A'_i, Λ^{-1}
    Λ_0^{-1} = B_0^{-1}
    for i = {1, ··· , k − 1}
        A'_i = A_i · Λ_{i−1}^{-1}
        Λ_i^{-1} = (B_i − A'_i · C_{i−1})^{-1}

    H_0 = (0, 0)
    for i = {1, ··· , k − 1}
        H_i = Δh_i^⊥ − A'_i · H_{i−1}

    Δx_{k−1} = Λ_{k−1}^{-1} · H_{k−1}
    for i = {k − 2, ··· , 1}
        Δx_i = Λ_i^{-1} · (H_i − C_i · Δx_{i+1})
```

Figure 2: *Computing position offsets $\Delta\mathbf{x}_i$ from half vector offsets $\Delta\mathbf{h}_i^{\perp}$. Note that, unlike in Figure 1, H is a 2D vector here.*

Continuous Parameterization of Tangent Frames If a scene object has a continuous surface mapping (e.g., texture coordinates) almost everywhere, we always precompute the tangent bundle (tangent space for every surface point) for this object based on this provided mapping.

However, if an object comes without any on-surface parameterization, we use the tangent bases that can be constructed locally based the local parameterization of the object's primitives. E.g., for a single triangle such a basis can be computed on the fly by orthonormalizing any two edges of this triangle. The problem with this approach is that the half vector, when projected onto such a basis, should be rotated when walking from one primitive to another in order to keep the global on-surface orientation. In order to account for such arbitrary orientations of the tangent bases on different primitives, we construct a line every time we step from one primitive to another. This line from the source point to the new point, projected onto both old and new tangent basis, gives us a shared on-surface orientation, which we use to locally orient the projected half vector. This needs to be done at every iteration of the corrector pass in our predictor-corrector algorithm in order to keep the global on-surface orientation of the projected half vector on the objects without any global continuous mapping.

Anisotropic Half Vector Sampling Instead of resampling each half vector $\mathbf{h}^{\perp}$ from scratch every time, we are performing a random walk in $\mathbf{H}^{\perp}$. This means we have to sample an anisotropic offset around a current half vector $\mathbf{h}^{\perp}$ of the current path $\mathbf{X}_t$. Naively sampling a small disk centered around $\mathbf{h}^{\perp}$ can lead to early rejection of $\mathbf{X}_{t+1}$ due to a potentially asymmetric support of the mutation (i.e., the reverse walk would be impossible) or due to sampling outside the unit circle domain. To avoid both cases in a practical way, we first sample a zero-centered anisotropic Phong lobe [Ashikhmin and Shirley 2000] and rotate it according to the orthonormalized axes of the ray differentials. The orthonormalization preserves the direction of the longer component of ray differentials. We use the step size derived in Sec. 6 of the paper

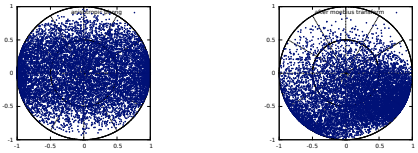

Figure 3: *Illustration of the anisotropic half vector sampling. First, an anisotropic Phong lobe is sampled (left) and then transformed around the pivot point, here at* $(0.25, -0.3)$ *(right).*

(i.e., the minimum of suggested BSDF step and the ray differential lengths $||\Delta\mathbf{h_u}||$ and $||\Delta\mathbf{h_v}||$ along the corresponding axis) and use Walter's [2007] approximation to convert Beckmann roughness to Phong exponent. Then we move this distribution to be centered around the current value of the half vector constraint $\mathbf{h}_t^\perp$ by applying a Möbius transformation [Hanika 2011, p.40]. This has the full disk as support and never samples outside it (see Figure 3). We also tried to directly sample a 2D Gaussian in plane-plane parametrization, the native space for Beckmann lobes, but found evaluation of PDFs in this space numerically instable, especially for large step sizes.

2 Numerical Validation of Properties of the Product of Jacobians

To provide a numerical validation of the properties of the product of Jacobians, we set up a simple test program to trace and visualize a path (see Figure 4, top row). Once a path is found, we can locally explore it using the Newtonian method described in the paper. For each one of the polar plots in Figure 4 we select only one half vector $\mathbf{h}_i^\perp$ to preturb and keep all other half vectors of the path fixed. We step through all pixels on the unit disk, i.e. enumerate all possible values of the half vector. Then the path is constructed in world space to keep the start and end points as well as all half vectors, except the new one $\mathbf{h}_i^\perp$, fixed. After that, the measurement contribution $f(\mathbf{X})$ and the BSDFs at the corresponding local vertex $\rho(\mathbf{x}_i)$ are computed and the normalized values are plotted in false-color into the polar plot. That is we visualize these quantities in half vector domain.

We can observe mostly zero-centered Gaussian-like shapes. The only other factors to observe are minor contributions from the product of Jacobians as well as from a few not microfaced-based terms (like Fresnel and sensor responsivity), which we visualize as a difference plot. The contribution of these as the surface becomes more smooth, but is relatively minor even for diffuse surfaces. The visibility affects the results in form of a binary mask.

3 Numerical Results for Difficult Visibility

Visibility is more problematic, as occlusion by the same object will appear in all half vector plots. Figure 5 shows an extreme case. The corresponding polar plots in half vector space (Figure 6) show very narrow restrictions for all half vectors, at every vertex. This means that we have to explore this space locally, i.e. by perturbing a path in the Markov chain Monte Carlo context with a relatively small step size. This is much the same as MLT would do it in vertex area measure, only that in our formulation it seems more difficult at first to keep parts of the path fixed (the part from the light source through the door for example). Nonetheless, the random walk can explore path space relatively well even in such difficult situations (see the AJAR DOOR scene in the paper).

Figure 7: *Bidirectional path tracing on the* JEWELRY *scene, using 40,000 samples per pixel. Indirect glossy caustics are still not resolved, so we compute our reference images with MLT instead.*

4 Additional Results

We provide more detailed results in this report: Figure 7 shows that bidirectional path tracing is problematic as a reference in some scenes. Figures **??**–12 provide larger insets of the KITCHEN scene and results of additional techniques and parameter settings which have been left out of the main paper. Figure 13 includes a comparison with PSSMLT in the AJAR DOOR scene. Figure 14 shows additional comparisons of the JEWELRY scene, especially sample counts of the equal-time renders of our technique alone vs. ours in addition to MLT. Figure 15 is the non-cropped version of Figure 10 in the main paper, and Figure 16 illustrates our choice of the λ parameter for manifold exploration. Figure 17 shows the importance of the spectrum-motivated sampling described in Sect. 6.3 in the paper.

References

ASHIKHMIN, M., AND SHIRLEY, P. 2000. An anisotropic Phong BRDF model. *Journal of Graphics Tools 5*, 2, 25–32.

HANIKA, J. 2011. *Spectral light transport simulation using a precision-based ray tracing architecture*. PhD thesis, Ulm University. VTS-ID/7539.

JAKOB, W., 2010. Mitsuba renderer. http://www.mitsuba-renderer.org.

WALTER, B., MARSCHNER, S., LI, H., AND TORRANCE, K. 2007. Microfacet models for refraction through rough surfaces. In *Proc. Eurographics Symposium on Rendering*, 195–206.

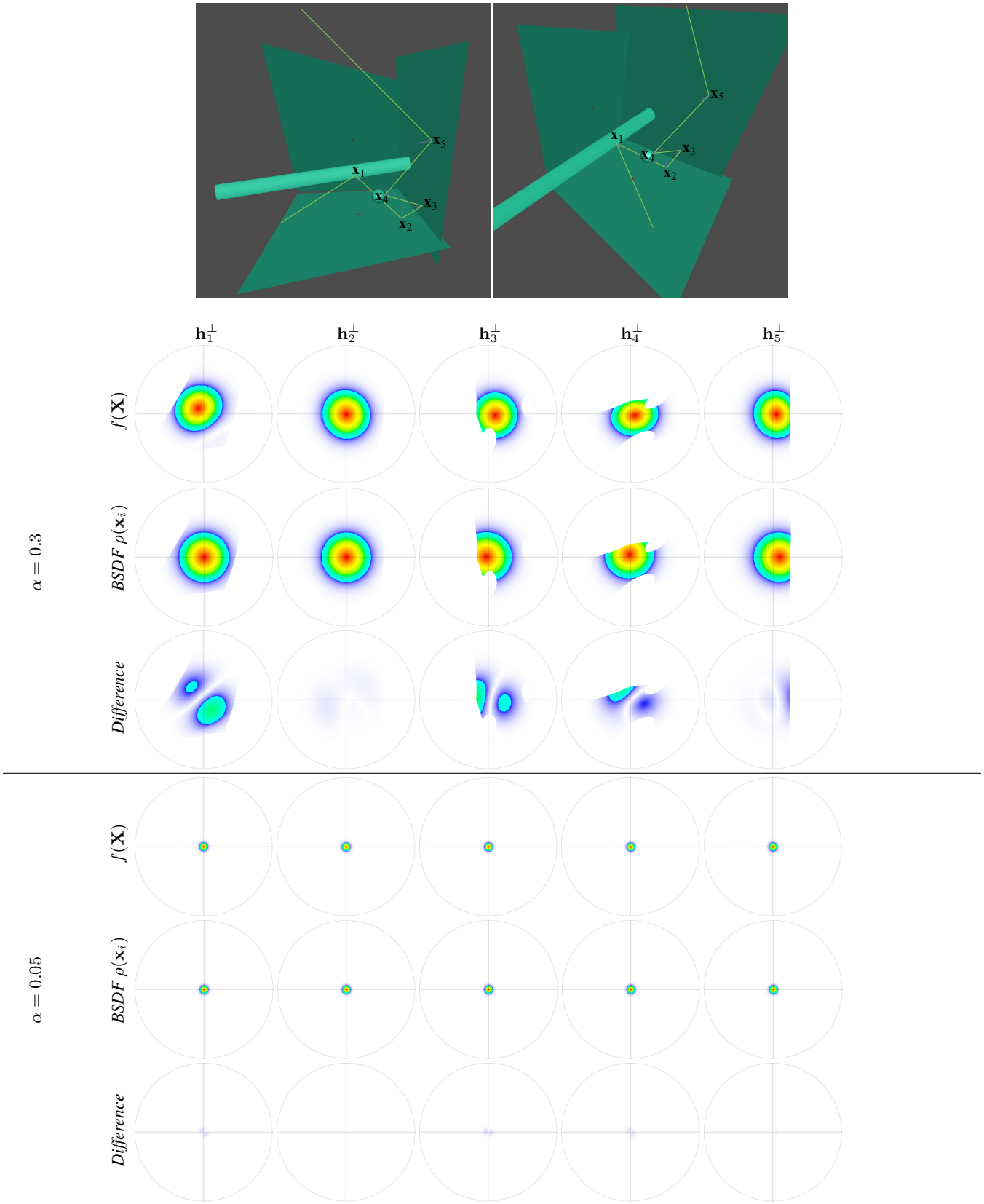

Figure 4: *Top: the images show two different views of a simple scene which we use to analyze the measurement contributions and local BSDF changes. Bottom: these plots show a numerical evaluation of the changes in the measurement contribution as opposed to the changes only in the corresponding local BSDF (with respect to every projected half vector of the path); each column corresponds to the perturbation of a single half vector. The upper three rows use a Beckmann roughness $\alpha = 0.3$, the bottom part show plots for $\alpha = 0.05$. Note the sharp boundaries of the plots are due to occlusion. Note the complete measurement changes very closely to changes in the corresponding local BSDF, which demonstrates the better decorrelation of the path integral into smaller, more independent subproblems. The insets of this experiment are shown in Fig. 6 in the paper.*

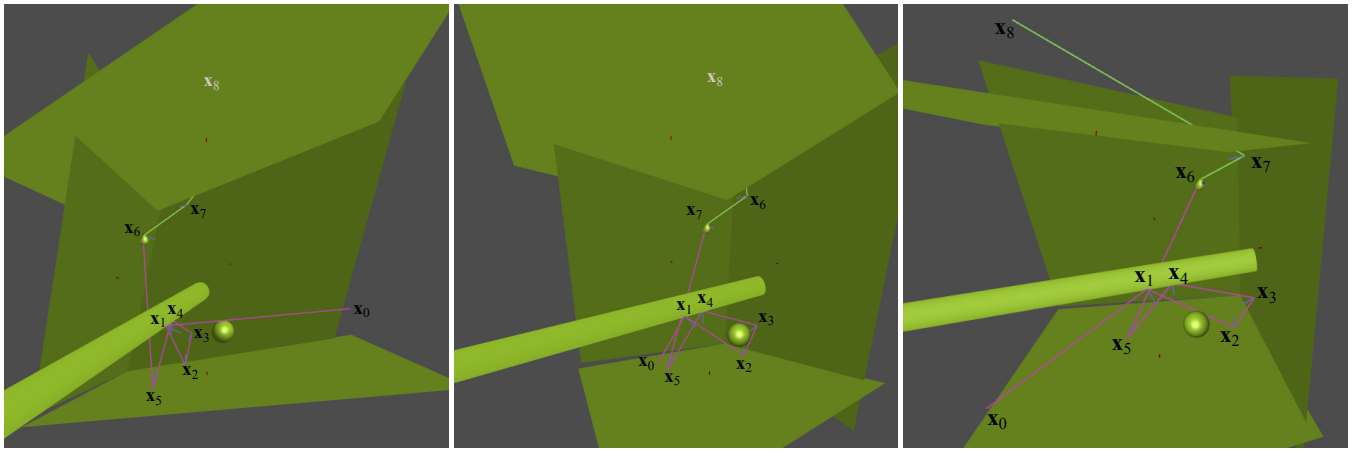

Figure 5: *These images show three different views of a simple scene which we use to analyze the measurement contributions and local BSDF changes. Compared to the previous example, this scene exhibits very difficult visibility.*

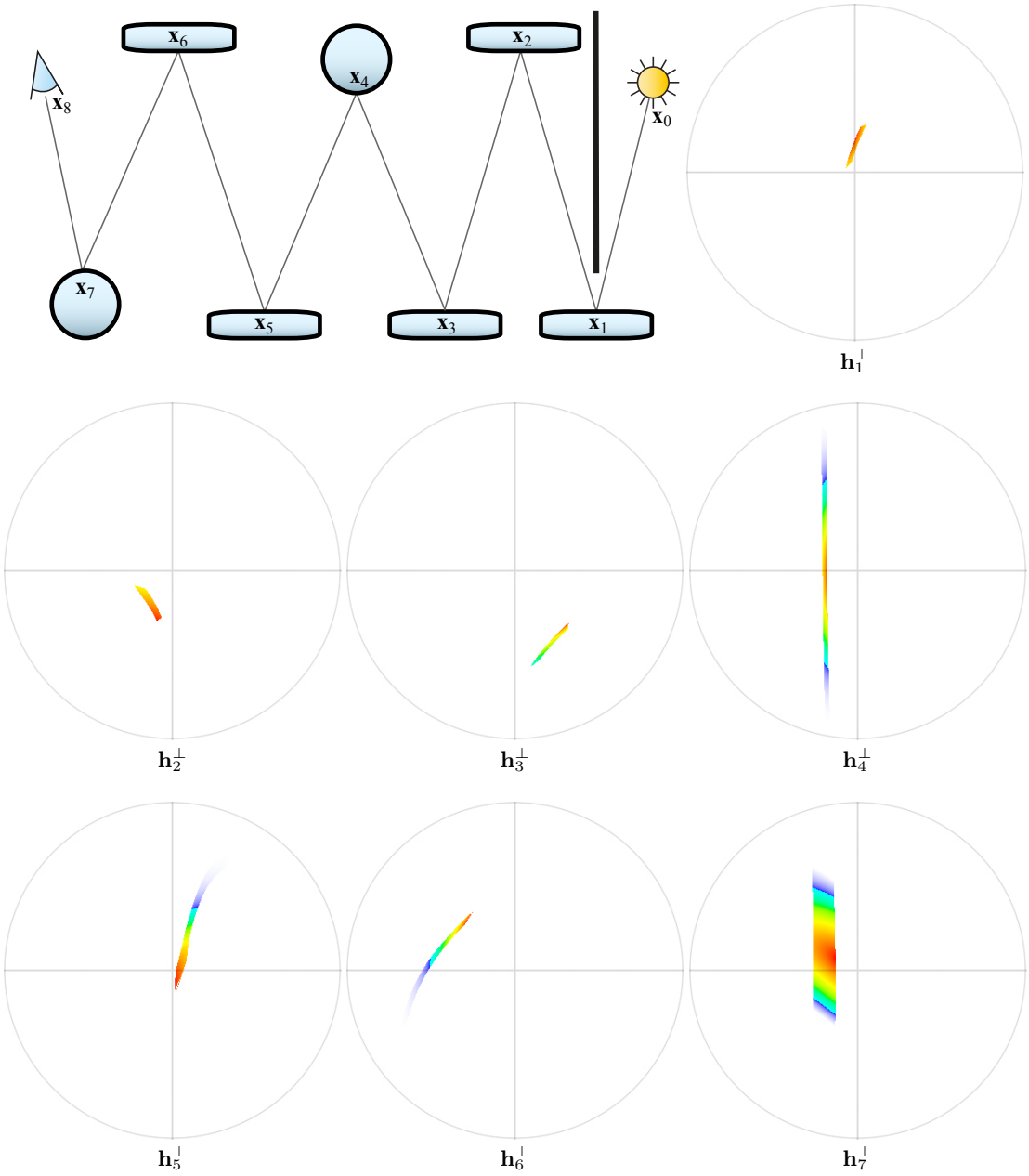

Figure 6: *The difficult visibility (Figure 5) is reflected in the visualization of the measurement contribution distribution. Note how the occlusion due to the geometry close to the light source narrows the domain in all distributions.*

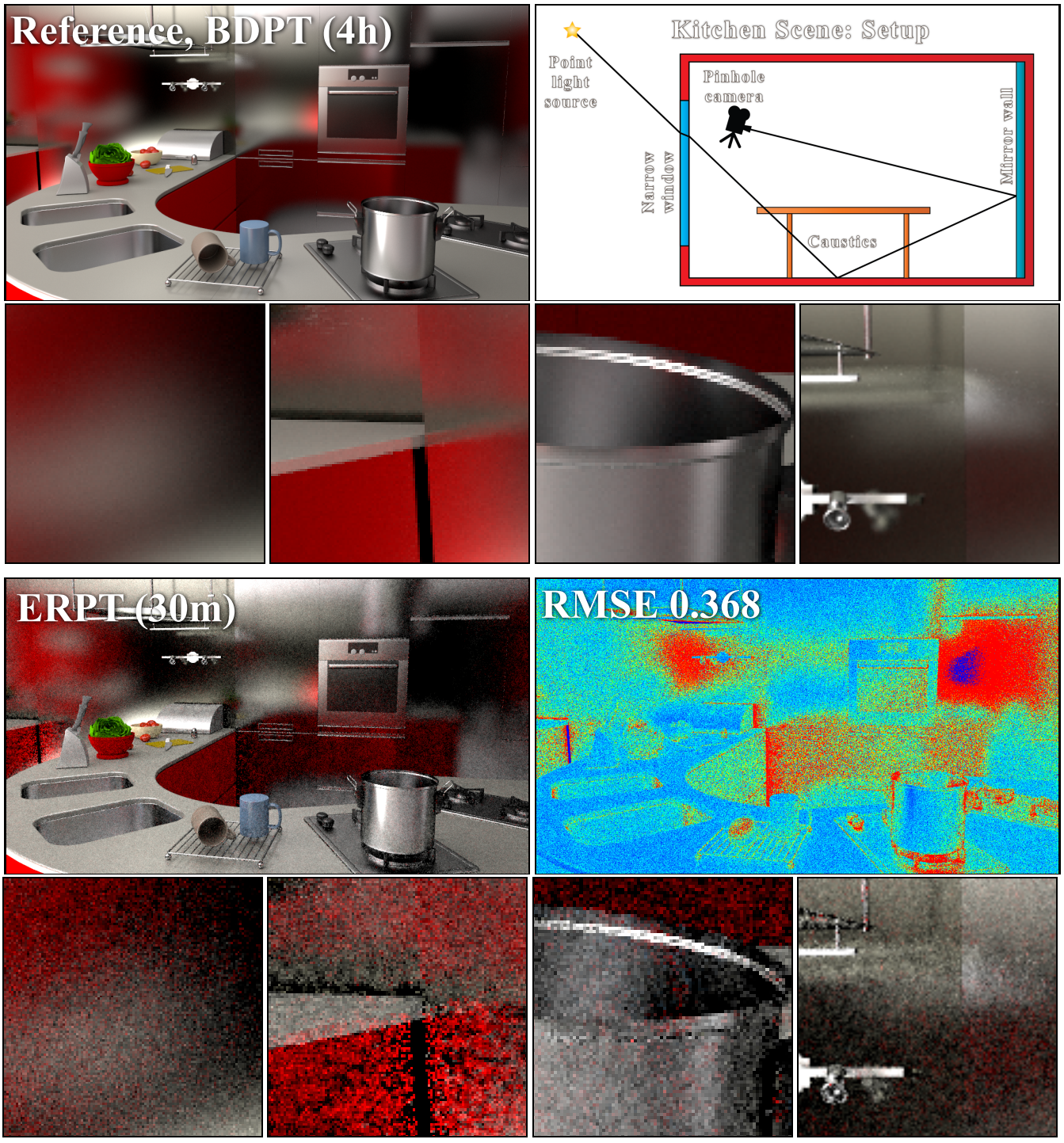

Figure 8: *Equal-time rendering of* KITCHEN *scene with difficult glossy paths.*

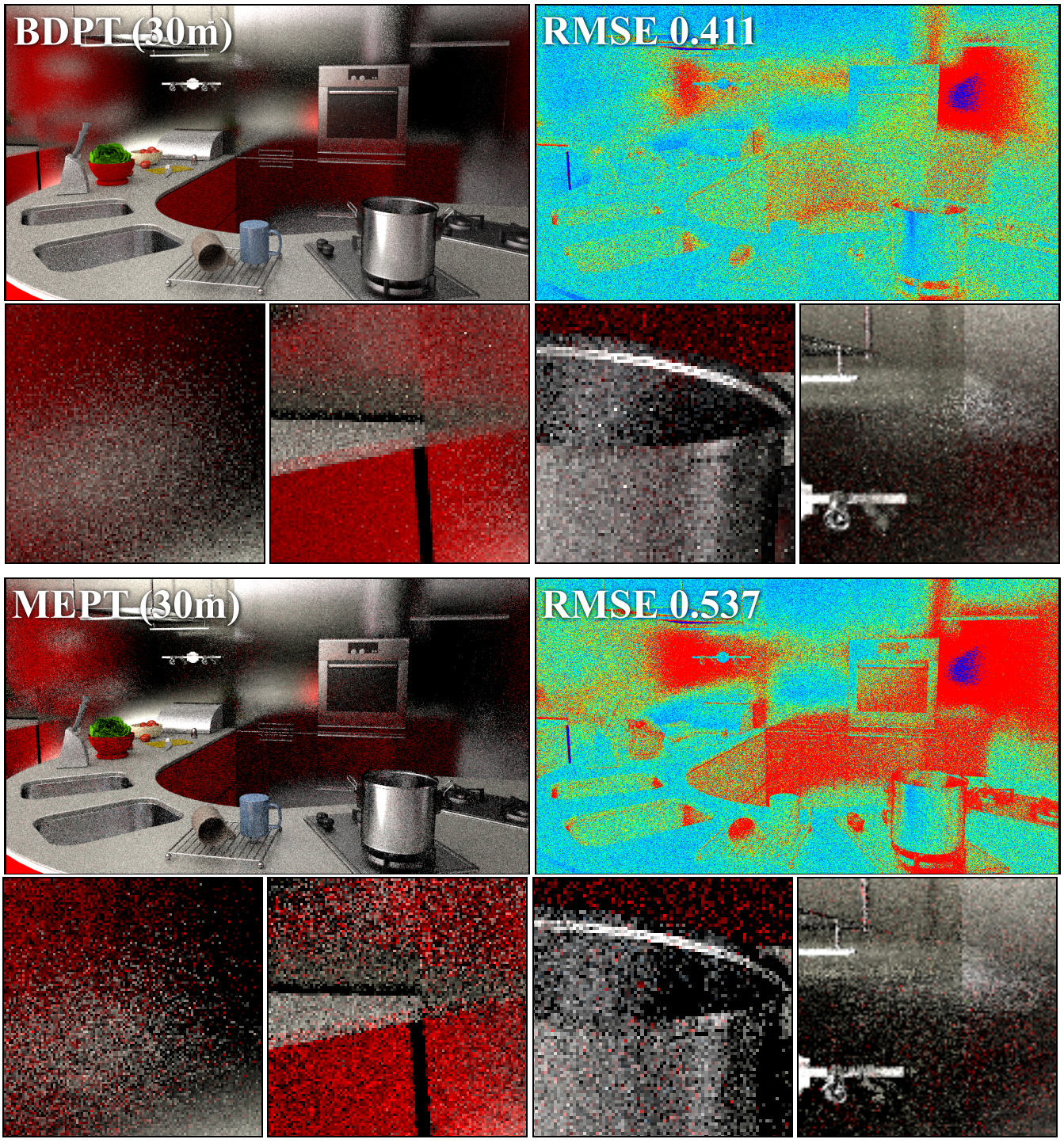

Figure 9: *Equal-time rendering of* KITCHEN *scene with difficult glossy paths.*

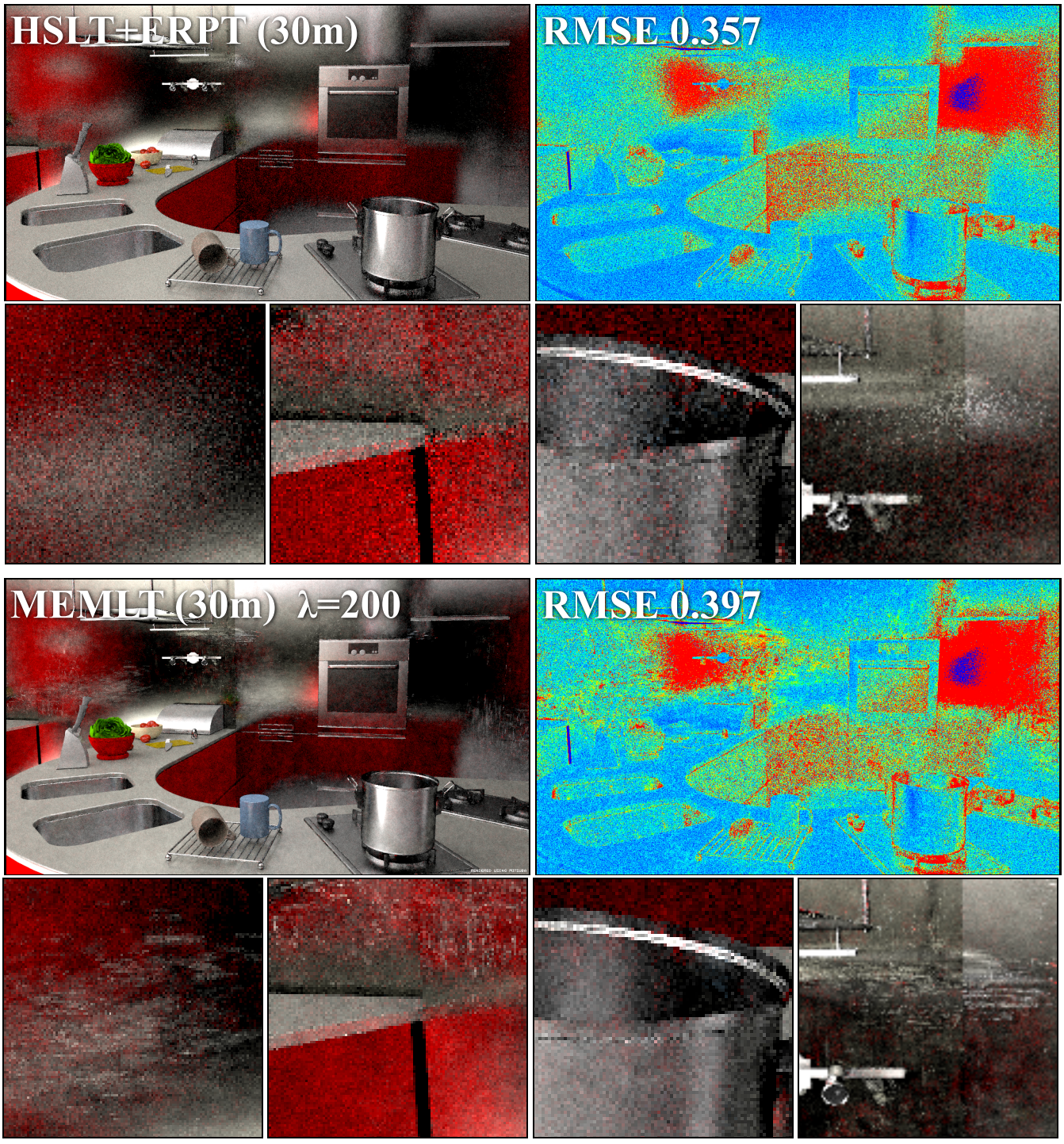

Figure 10: *Equal-time rendering of* KITCHEN *scene with difficult glossy paths.*

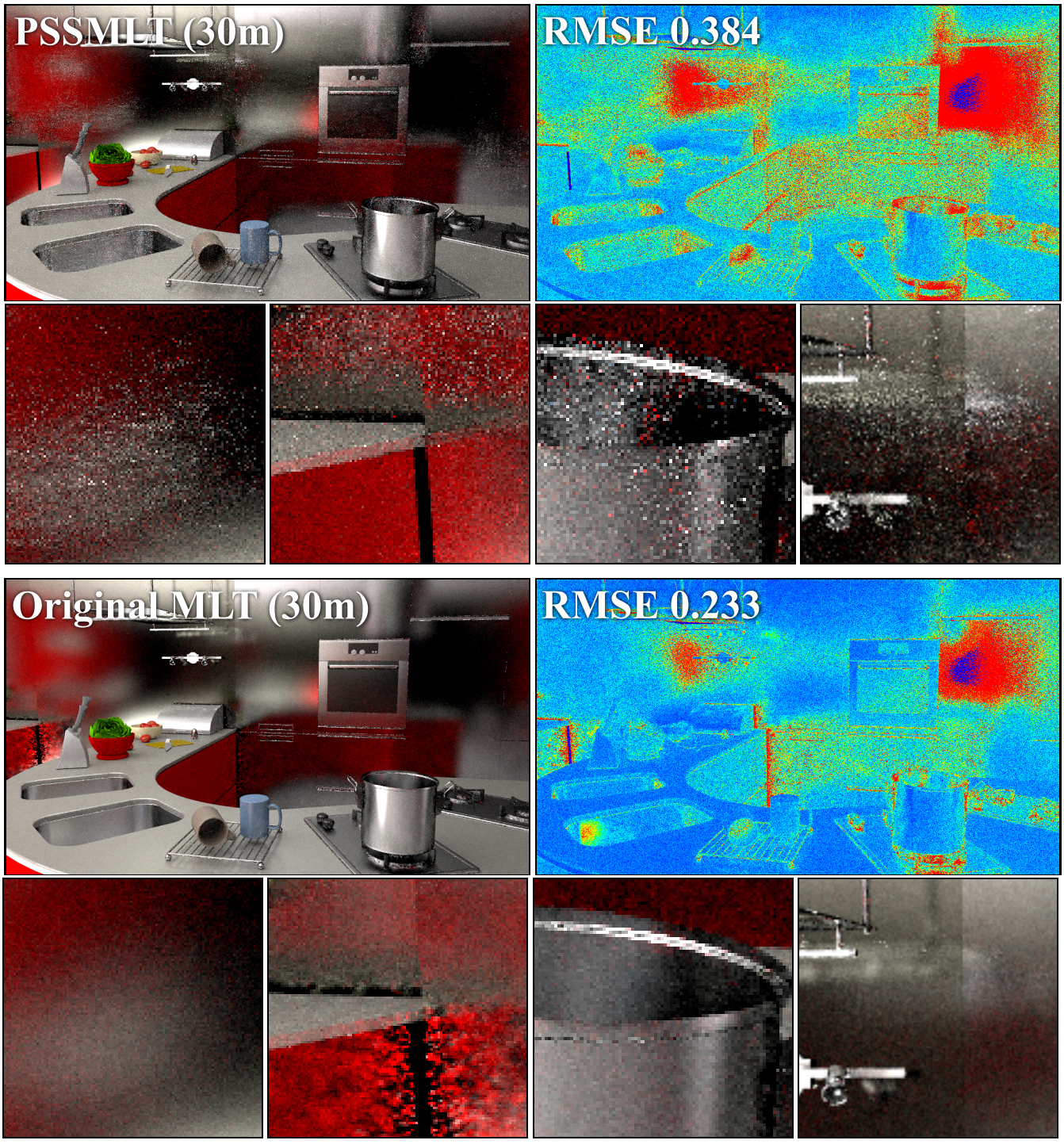

Figure 11: *Equal-time rendering of* KITCHEN *scene with difficult glossy paths.*

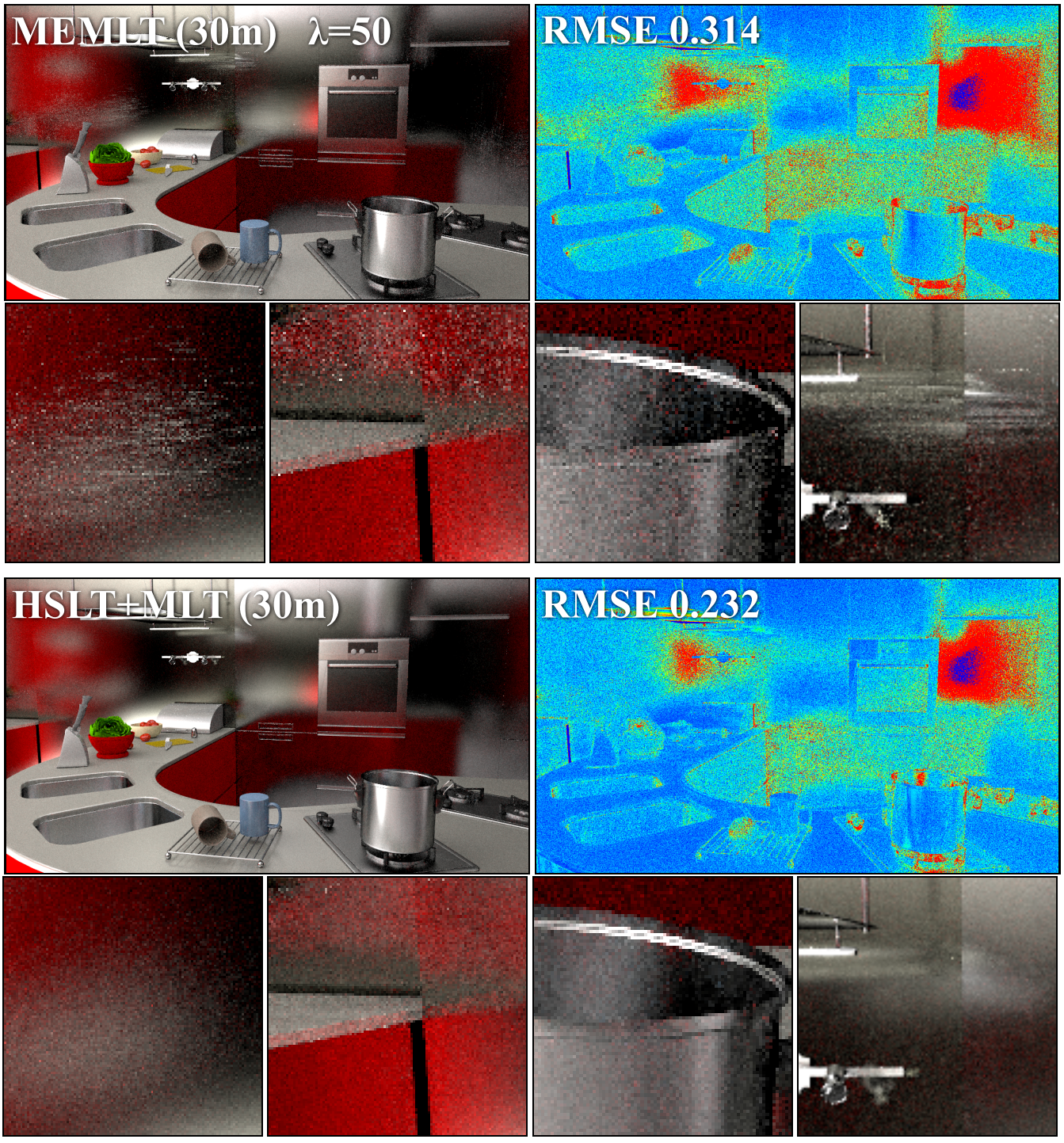

Figure 12: *Equal-time rendering of* KITCHEN *scene with difficult glossy paths. Note how efficient the new mutation can estimate the optimal sampling density of difficult regions and deal with hard features like glossy caustics.*

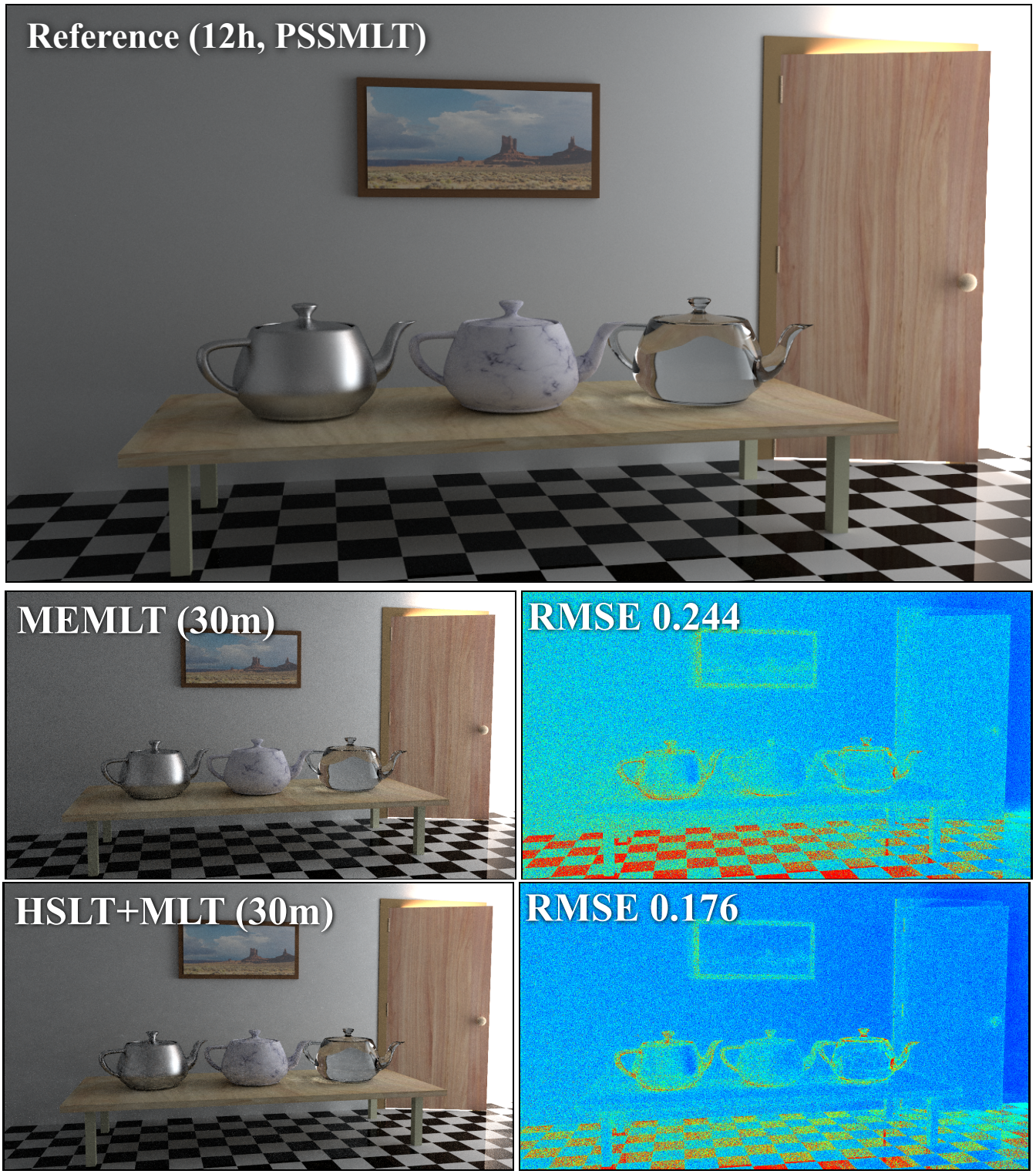

Figure 13: *The* AJAR DOOR *(reference rendered with PSSMLT for 12 hours is on top) rendered in 30 minutes using MEMLT (with the original set of mutations) (middle), and MLT with only our mutation and bidirectional mutation (bottom).*

Figure 14: *Equal-time rendering (10 min) of the* JEWELRY *scene with difficult glossy and specular paths. The reference image (top left) was computed with original MLT in 60h. BDPT cannot efficiently explore indirect caustics (see Figure 7). The enlarged insets show: original MLT shows blotchy artifacts close to glossy edges due to jumping off the constraints. ME classifies glossy vertices as specular or diffuse, thus reducing the efficiency. Our mutation (HSLT), coupled only with the bidirectional mutation, compares favorably in most difficult areas, yet is slow to compute in simple regions. Mixing original MLT mutation strategies with ours alleviates this (HSLT+MLT). The average samples per pixel were: MEMLT* 621*, MLT* 1424*, HSLT+MLT* 612*, HSLT* 577*, the image resolution is* 1024×768 *pixels.*

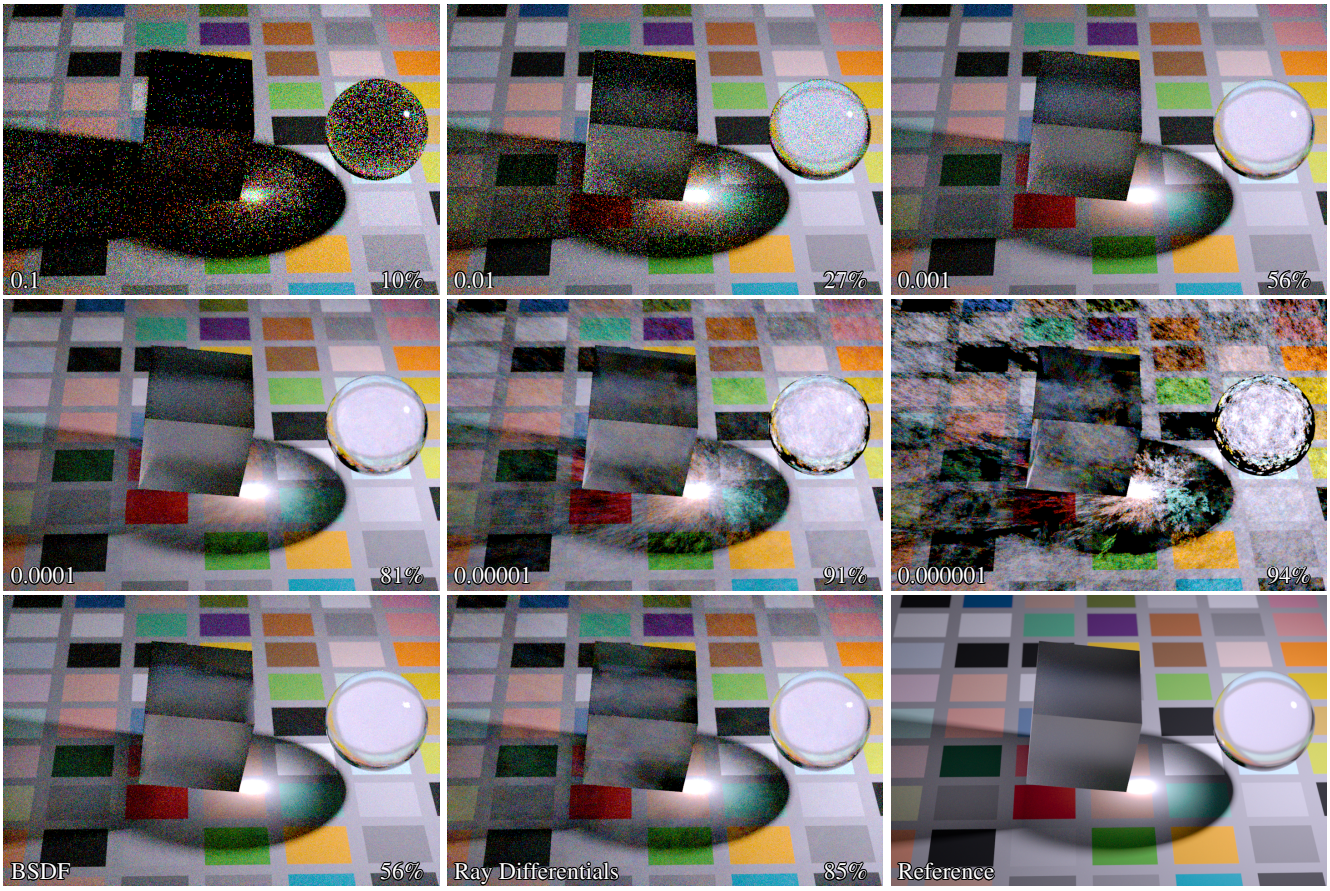

Figure 15: *A simple scene with a diffuse ground plane, a sphere, and a cube with a rough dielectric with different roughnesses (*0.01 *and* 0.1*), rendered with manually chosen step sizes as indicated. Bottom row shows our automatically selected step size for BSDF bandwidth (left) and the same combined with stratification based on ray differentials (middle), as well as a reference render. The percentage in the corner indicates the acceptance rate.*

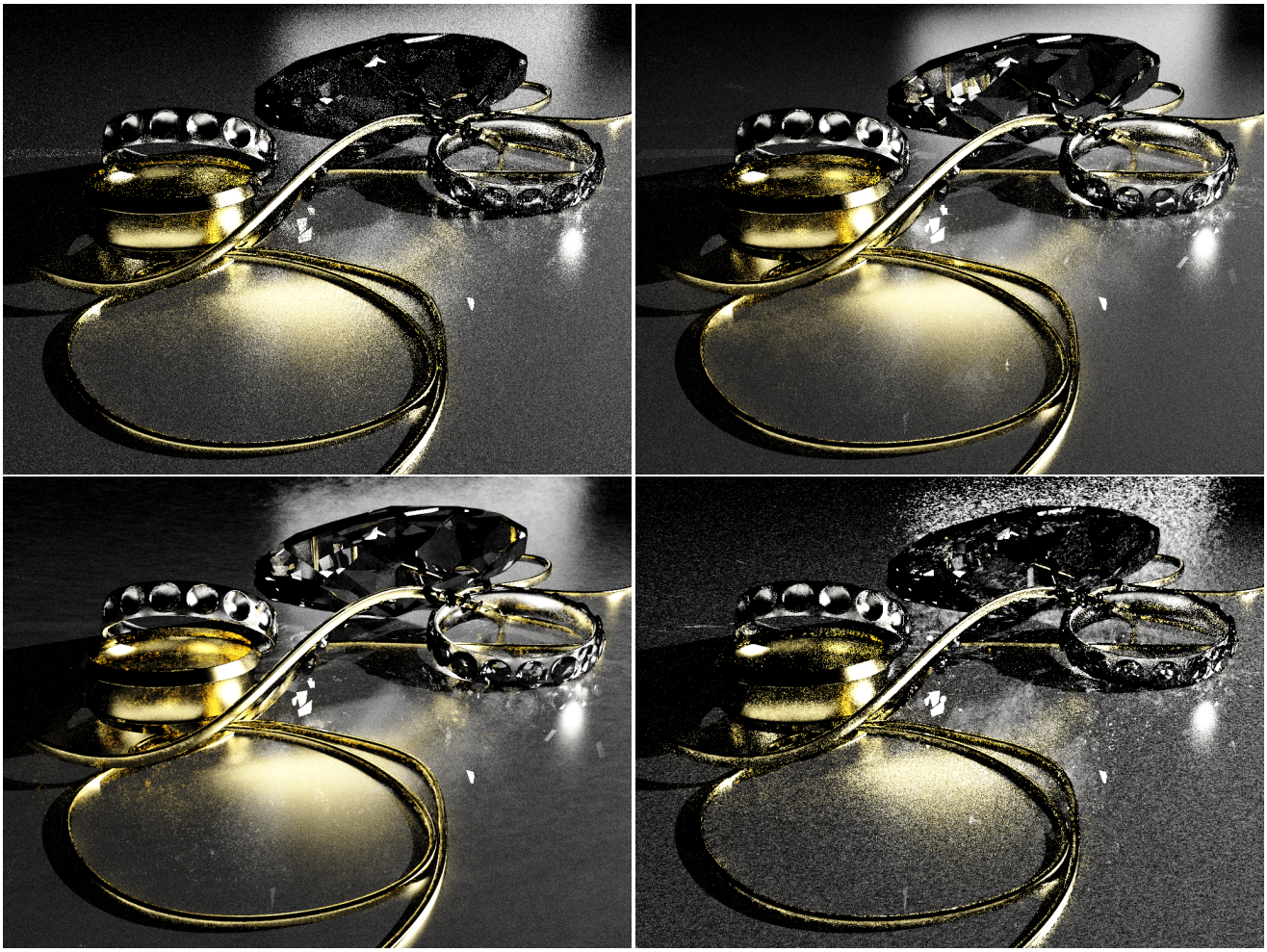

Figure 16: *The* JEWELRY *scene with MEMLT and different values for the parameter* λ*. From left to right, top to bottom:* $\lambda = \{5, 50, 500, 5000\}$ *with 1M initial samples (seeding paths) and* 256 *samples per pixel. The corresponding acceptance rates are* 10.88%*,* 35.7%*,* 65.5%*, and* 63.3%*, respectively. We used the default* $\lambda = 50$ *due to its uniform distribution of samples in the image plane, to avoid clumping in the background light. Note that the acceptance probabilities are quite different for importance and radiance transport. In our method, we dynamically pick the most convenient tracing direction (see Section 5.1,* Obtaining Path Vertices *in the main paper).*

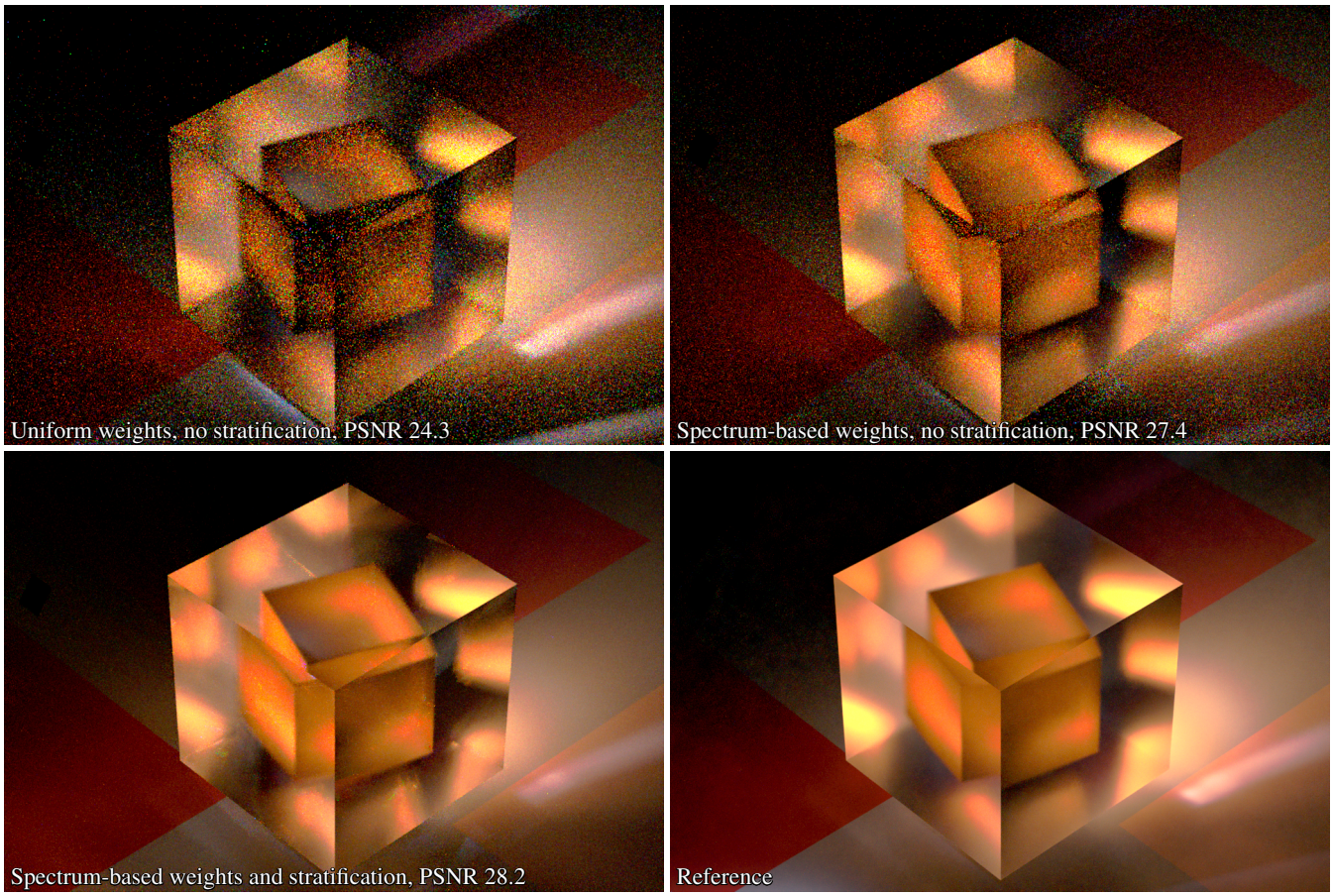

Figure 17: *A simple scene with a rough dielectric cube ($\alpha = 0.01$) enclosing another (tinted orange) rough dielectric cube ($\alpha = 0.1$) on a diffuse ground plane. The illumination comes from a small spotlight in the upper left corner shining onto the small enclosed cube. Using the sampling density motivated by estimated integrand spectral density (Sec. 6.3 in the main paper) to distribute mutation step size according to BSDFs' bandwidths ensures more even perturbations on both different roughnesses occurring on the same path. Stratification (Sec. 6.2 in the main paper) help to further optimize the sampling. All images except the reference use 1024 samples per pixel on average. The reference image uses PSSMLT with 10467 samples per pixel.*