# Blind Augmentation: Calibration-free Camera Distortion Model Estimation for Real-time Mixed-reality Consistency

Siddhant Prakash, David R. Walton, Rafael K. dos Anjos, Anthony Steed, and Tobias Ritschel

## Supplemental Materials

### Noise Model Details

The noise model used in this paper proceeds in two main stages: first the model is fitted, based on an input image and noise map. Second, this model is used to synthesise noise for virtual content added to the scene.

#### Fitting

The inputs to the fitting step are the input image $I$ and the estimated denoised image $I_{\text{noNoise}}$ produced by the denoising method. From these we find an estimate for the ground truth noise $N_{GT} = I - I_{\text{noNoise}}$.

We first construct a Laplacian pyramid of $N_{GT}$. The examples in this paper make use of a 4-level Laplacian pyramid, which was sufficient as noise was typically of high frequency. The final level of the pyramid containing the means was ignored, as we assume the noise to have zero mean.

We compute local standard deviations for each level of this pyramid using the method of [**?** ], to wit we use the standard formula:

$$\sigma(X) = \mathbb{E}(X^2) - \mathbb{E}(X)^2 \tag{1}$$

We find local means by applying a blur $B$ to the pyramid level, so for the $l$th level of the pyramid $N_{GT,l}$:

$$\sigma(N_{GT,l}) = B(N_{GT,l}^2) - B(N_{GT,l})^2 \tag{2}$$

In our case we choose $B$ to be a 9x9 Gaussian blur with $\sigma = 4$ at the highest resolution pyramid level. The value of $\sigma$ was halved at each level to provide the same effective pooling size at all resolutions.

At each pyramid level $l$ we then fit a linear mapping $L_l$ from the luminance at each pixel (i.e. Y channel of the image in YCbCr format) to the local standard deviations $\sigma(N_{GT,l})$. Since we have standard deviations for the red, green and blue channels, this is a mapping $L_l : \mathbb{R} \to \mathbb{R}^3$. This is fitted by least-squares linear regression, giving a 2x3 matrix encoding the model per pyramid level. In practice we sample the images once every 32 pixels in each dimension to greatly reduce the number of samples our model has to fit to, meaning the least-squares fitting has to find the pseudo-inverse of a much smaller matrix. This pixel skip is also halved as we move down each resolution level.

Our total model consists of a total of $4 \times 2 \times 3 = 24$ parameters.


- *Siddhant Prakash, Anthony Steed, and Tobias Ritschel are with University College London. E-mail: {s.prakash | a.steed | t.ritschel}@ucl.ac.uk.*
- *David R. Walton is with Birmingham City University. E-mail: david.walton@bcu.ac.uk.*
- *Rafael K. dos Anjos is with University of Leeds. E-mail: r.kuffnerdosanjos@leeds.ac.uk.*


*Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx*

#### Synthesis

To synthesise noise to add to some noise-free virtual content $I_{\text{virt}}$, we first start from an image containing random normally-distributed noise $N_{\text{init}}$ over the RGB channels. A laplacian pyramid of this noise map is constructed, and each level is normalised to have local standard deviation of 1 at all locations. This is achieved by finding the local standard deviations $\sigma$ as above, and simply dividing through:

$$N_{\text{norm},l} = \frac{N_{\text{init},l}}{\sigma(N_{\text{init},l})} \tag{3}$$

We find the luminance of the virtual content we wish to add the noise to $Y_{\text{virt}}$. For each pyramid level of the noise we input this to the corresponding linear mapping $L_l$ to obtain a map of standard deviations in red, green and blue. We multiply these by the normalised noise pyramid levels $N_{\text{norm},l}$ to achieve the correct statistics at each pyramid level. Finally, reconstructing from this Laplacian pyramid of weighted noise maps produces the synthesised noise, which can be added to the virtual content.

We note that though our current implementation is based in Pytorch, efficiency could be further improved by making use of graphics hardware accelerated MIP map generation to compute the pyramids as was done in Walton et al. [**?** ].

Table 1: List of cameras and lenses used.

| Camera + Lens | Figure |
|---|---|
| Flir Chameleon 3 + TV 16mm 1:1.4 | Fig. 1, Fig. 2 Fig. 3 Row 1, 2, Fig. 10 (a, c, d), Fig. 1 (a, d) (supp) |
| Sony Alpha 7III + Tamron 28-75 F/2.8 | Fig. 3 Row 4, 5, Fig. 10 (b), Fig. 11, Fig. 2 (supp) |
| Nikon D3100 + Nikkor 55-200mm F/1.4-5.6 | Fig. 3 Row 3 Fig. 10 (e, f) |
| Meta Quest 3 + on-device lenses | Fig. 9 |

### Additional Results and Comparison

A list of cameras and lenses used for results in each figure is provided in Tab. 1.

We provide additional results similar to Fig. 10 and Fig. 11 of the main paper in Fig. 1 and Fig. 2, respectively.

We also show the parameters recovered as a result of our optimization on the images in Fig. 3 of the main paper in Fig. 3, Fig. 4, and Fig. 5.

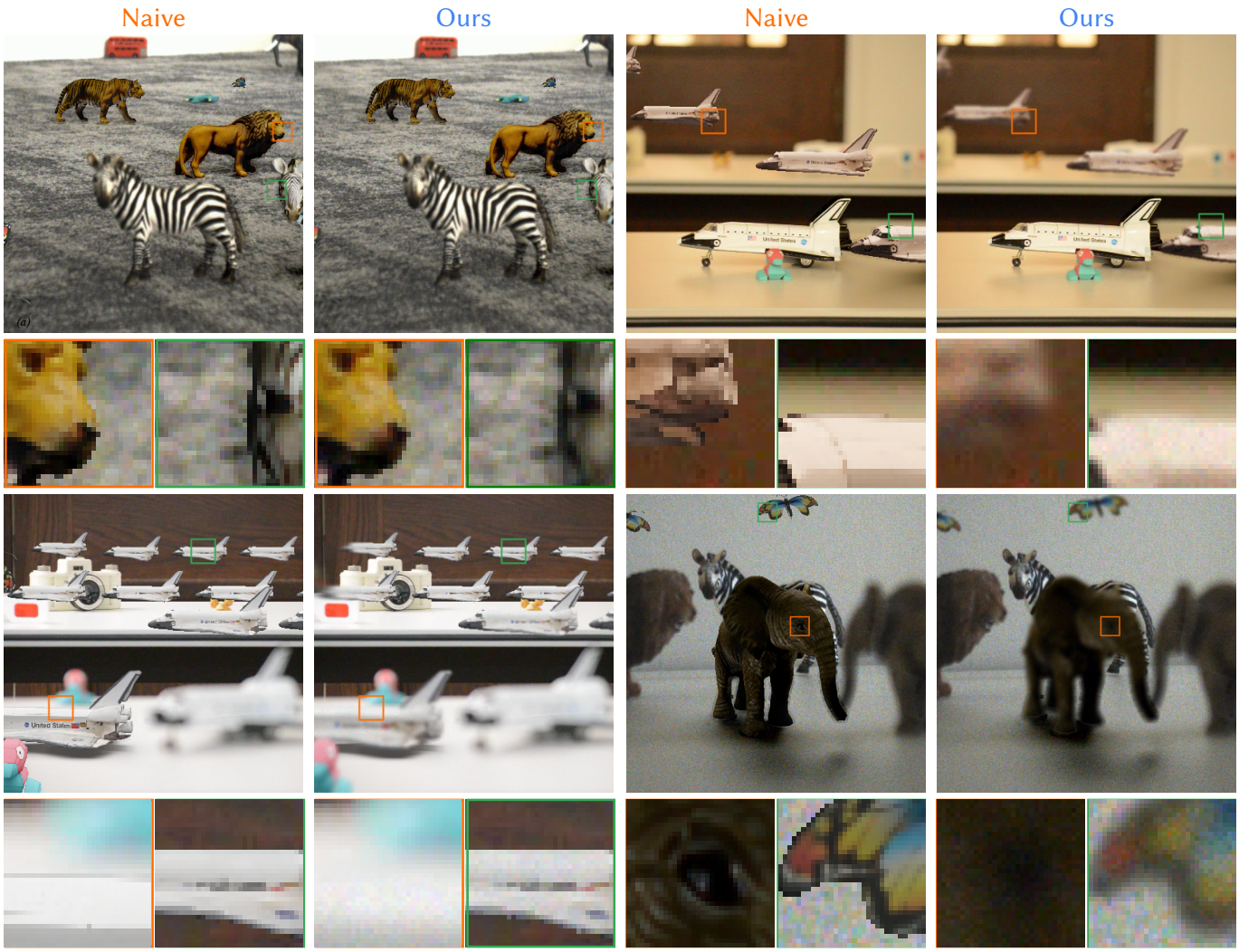


Fig. 1: Additional typical results produced by our approach. We always show a pair of `Naive`(naïve) compositing (columns 1,3) and `Ours` (columns 2,4). Below each pair, we show insets from both methods.

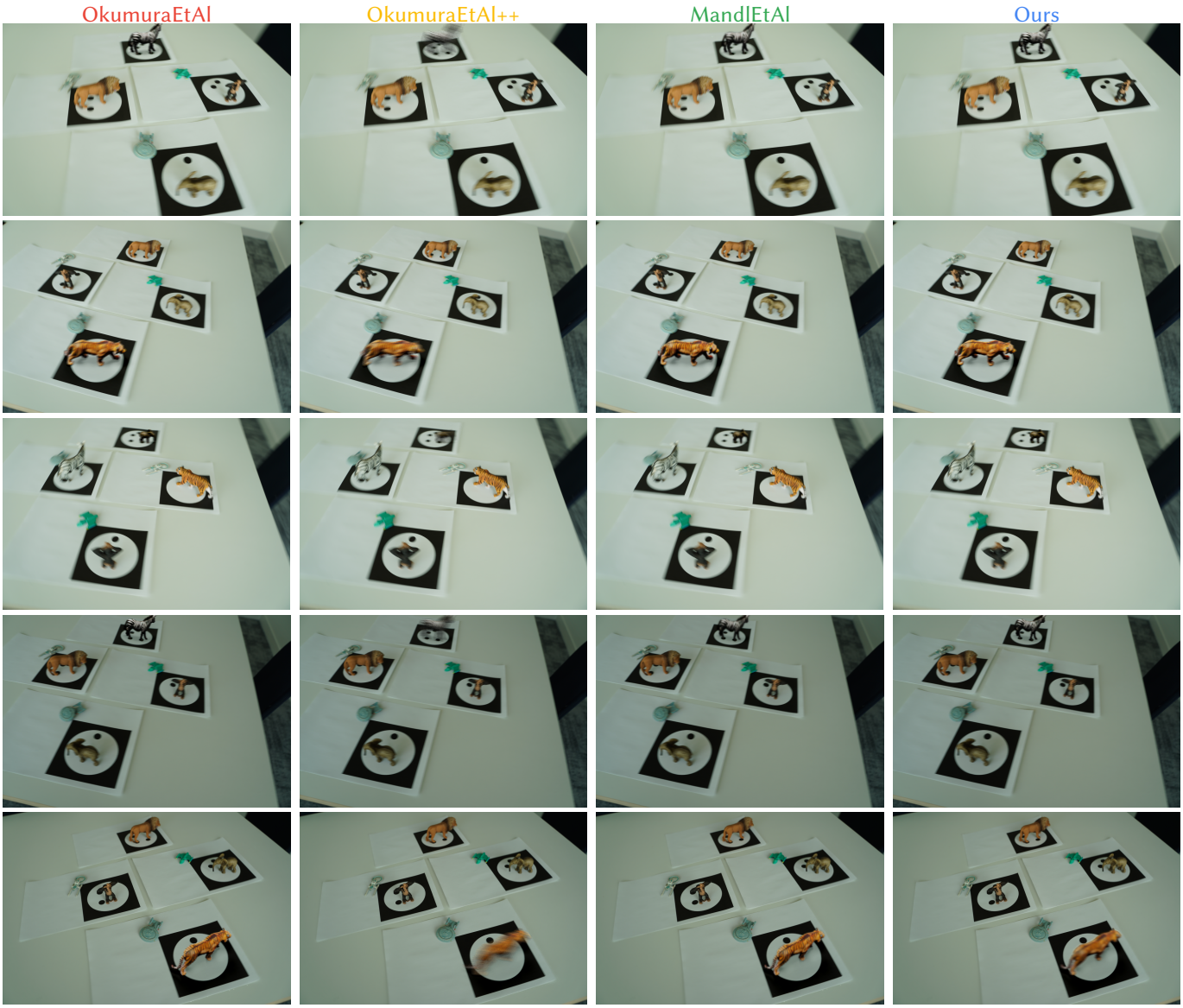


Fig. 2: Additional comparison between different methods **(columns)** on different scenes **(rows)**. Please note that all methods except `Ours` either have seen the marker and need it to be present (`OkumuraEtAl` and `OkumuraEtAl++`) or need previous calibration on a known object from the scene `MandlEtAl`. Our methods does not use that marker, neither did it see the scene before. We add a virtual Zebra figurine (rows 1, 4), virtual Tiger figurine (rows 2, 5) and a virtual Elephant figurine (row 3) to the scene. All other objects are real.

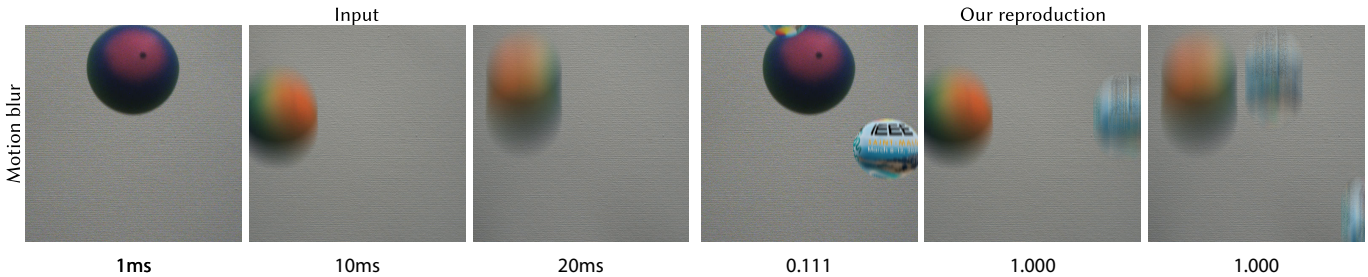


Fig. 3: Recovered parameters for motion blur (MB). On the left three columns we show images captured at varying exposure levels. This can be observed from varying level of motion blur in the real falling ball. On the right we have our reproduction of motion blur on virtual falling balls which is blurred with the parameters recovered from our optimization. The exposure parameters recovered are shown below each frame.

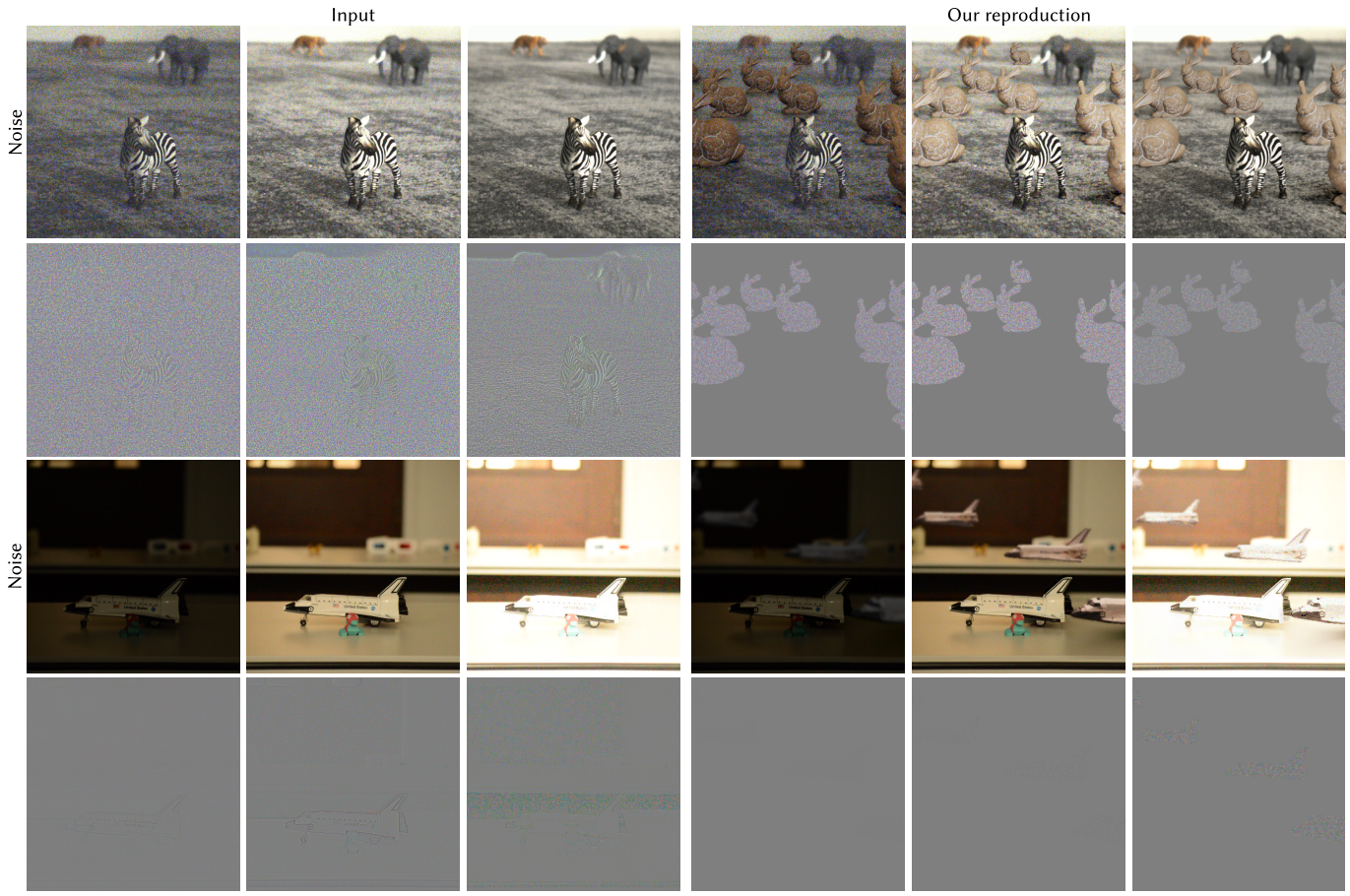


Fig. 4: Recovered parameters for noise. We show the residual noise from input images (left three columns, second and fourth row) and our generated noise (right three columns, second and fourth row) as a result of our optimization for the images (left three columns, first and third row). The generated noise are synthesized over the virtual composites (right three columns, first and third row) on the input images to make them consistent with overall noise in the image.

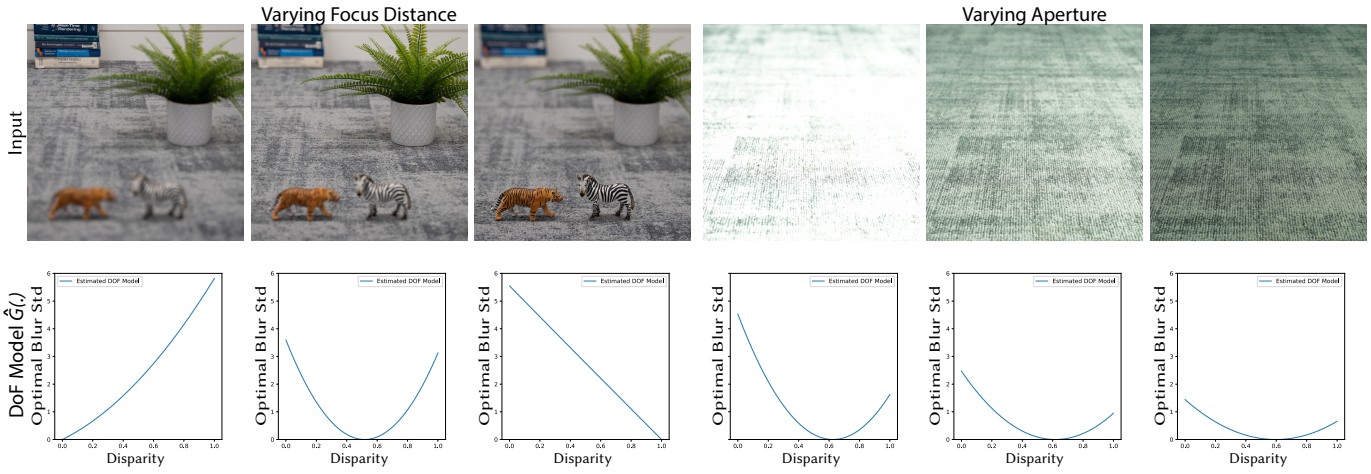


Fig. 5: Recovered parameters for depth-of-field (DoF). We show the DoF model $G(.)$ (bottom row) recovered for different input images (top row) with varying parameters. In the left three columns we show input images with varying focus distance, focused at far, middle and near plane respectively. In the right three columns we show input images with varying aperture setting of f/2.8, f/5.6 and f/11.0 respectively. We notice our recovered model accurately predicts the focal plane as well as the shallowness of DoF with decreasing (narrowing) aperture.