Supplementary: Lensless Imaging with Focusing Sparse URA Masks in Long-Wave Infrared and its Application for Human Detection

Ilya Reshetouski, Hideki Oyaizu, Kenichiro Nakamura, Ryuta Satoh, Suguru Ushiki, Ryuichi Tadano, Atsushi Ito, and Jun Murayama

Sony Corporation, Tokyo, JAPAN
{Ilya.Reshetouski,Hideki.Oyaizu,Kenichiro.Nakamura,Ryuta.Satoh, Suguru.Ushiki,Ryuichi.Tadano,Atsushi.C.Ito,Jun.Murayama}@sony.com

1 On Reconstruction Implementation and its Complexity

We perform the reconstruction of the $M \times N$ scene by computation of the following equation:

$$\hat{x} = U^* y \tag{1}$$

where $\hat{x} \in R^{MN}$ - 1D column vector corresponding to the reconstructed scene, $y \in R^{MN}$ - 1D column vector corresponding to the sensor data, U^* - the following $(MN) \times (MN)$ matrix:

$$U^* = \alpha U^{-1} \beta + u^T v \tag{2}$$

where α, and β - $(MN) \times (MN)$ diagonal matrices and u^T, $v \in R^{MN}$ - 1D row and column vectors correspondingly, U^{-1} - inverse URA imaging matrix. Here α - pixelwise correction coefficients of the scene intensity to compensate angular falloff, β - correction coefficients of the sensor intensity to compensate average transparency differences for sub-masks visible by each sensor pixel, and $u^T v$ - rank-1 approximation of the scene signal offset caused by diffraction.

The computational complexity of the calculation of the product $U^* y$ using (2) is the same as $U^{-1} y$ (i.e. URA reconstruction), can be performed with FFT and doesn't require explicit calculation or storage of U^* or U^{-1}.

Indeed:

$$U^* y = (\alpha U^{-1} \beta + u^T v) y = \alpha U^{-1} (\beta y) + u^T v y \tag{3}$$

The computational bottleneck of (3) is, obviously, $U^{-1}(\beta y)$. But it can be efficiently calculated due to the fact that the ideal imaging with URA mask can be described as convolution of the mask with the scene. Therefore, the

computation of the $U^{-1}(\beta y)$ can be done implicitly with the help of $2D$ Discrete Fourier Transform (DFT), e.g.

$$R_{M,N}(U^{-1}(\beta y)) = \mathcal{F}^{-1}(\frac{\mathcal{F}(R_{M,N}(\beta y))}{\mathcal{F}(U_b)}) \tag{4}$$

where $R_{M,N}$ - operator rearranging $1D$ vector of length MN into $M \times N$ matrix, U_b - basic $M \times N$ $2D$ URA pattern, and division performed elementwise.

Since the equation (4) can be computed with FFT in $O((MN)\log(MN))$ operations and requires only $O(MN)$ storage space, the same is valid for the formula (3) and, correspondingly, for the equation (1).

2 Mask Design Details

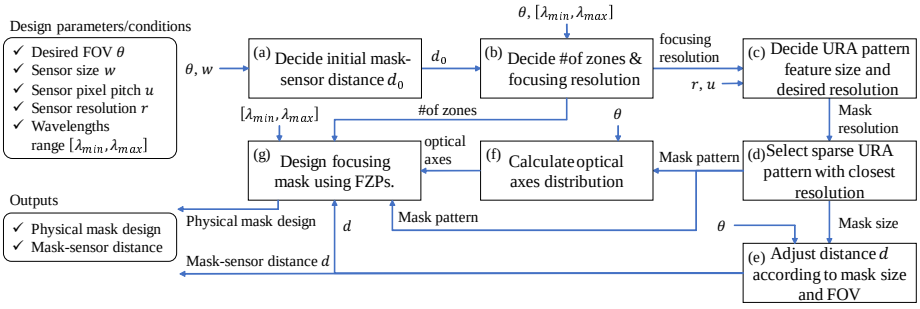

Fig. 1: Mask design pipeline

We describe the details about our mask design pipeline in this section. As shown in Fig. 1, we have several steps to deliver a specific physical mask design according to required specs such as camera FOV, image sensor size and resolution, and target wavelengths range.

As a first step (Fig. 1 (a)), initial mask-sensor distance d_0 is calculated according to the desired FOV θ and sensor size w:

$$d_0 = \frac{w}{2\tan\frac{\theta}{2}}. \tag{5}$$

Note that d_0 is a temporal parameter to roughly determine the number of zones and focusing resolution of FZP in the mask.

In the next step (Fig. 1 (b)), appropriate number of zones of the FZPs is determined. We chose this value empirically considering the sensor's spectral wavelength range (in our case it is $8-14\mu$m) and desired FOV. Consequently, focusing resolution can be calculated using the number of zones, wavelength interval $[\lambda_{min};\lambda_{max}]$, and mask-to-sensor distance d_0.

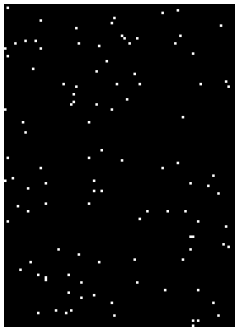

Fig. 2: Basic 127×91 sparse Singer URA mask. White dots indicate transparent features

Given the focusing resolution, the size and pixel pitch of the sensor, we decide the desired basic mask feature size and resolution (Fig. 1 (c)).

The next step, Fig. 1(d), is to select basic sparse URA pattern with resolution as close as possible to the one decided in the previous step. In our prototype we used the mask with the resolution 127×91 and the feature size 51μm (Fig. 2). Final URA pattern is constructed as 2×2 mosaic of the basic URA pattern, Fig. 3.

Since the size of the basic mask can be different from the size of the sensor, in order to maintain the desired FOV, we need to adjust the mask-sensor distance d_0. The final mask-sensor distance d is decided in the block (e) of the diagram (Fig. 1) using same method as in the block (a).

Once final URA pattern is known, we can produce the sub-lens optical axes map by calculating average meaningful incoming light direction for each transparent feature of the pattern, Fig. 4.

Resulting focusing mask is constructed in Fig. 1 (g) using: the final URA pattern, mask-sensor distance d, target wavelengths range $[\lambda_{min}; \lambda_{max}]$, the map of optical axes, and the number of zones for each FZP decided in Fig. 1 (b).

3 Reconstruction Algorithm Benchmark

We implemented theoretical and experimental comparisons of the performance of our reconstruction method and the classical inverse imaging matrix multiplication method (the inversion was performed with Tikhonov regularization), see Table 1. The experiments were performed on CPU only (Intel Core i9-9900K @ 3.60 GHz). From the comparison results we conclude, that our method is speed and memory efficient and significantly outperforms matrix multiplication approach.

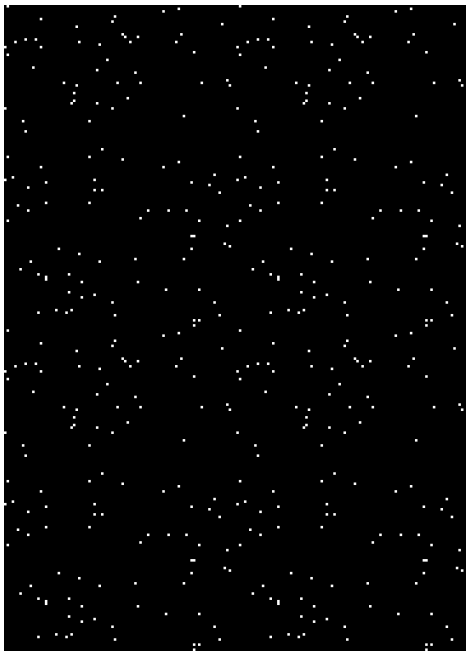

Fig. 3: Final 253×181 URA mask used as focusing template for our prototype. It constructed as 2×2 mosaic of the basic URA mask. Note that it has only 432 (0.94%) transparent features which makes it suitable for local focusing

Method	Computational Complexity	Measured Speed (FPS)	Space Complexity	Measured Memory (Mb)
Tikhonov	$\mathcal{O}((MN)^2)$	28.8	$\mathcal{O}((MN)^2)$	1019
Ours	$\mathcal{O}((MN)\log(MN))$	3405	$\mathcal{O}(MN)$	0.44

Table 1: Performance comparison between our method and Tikhonov regularized inverse matrix multiplication

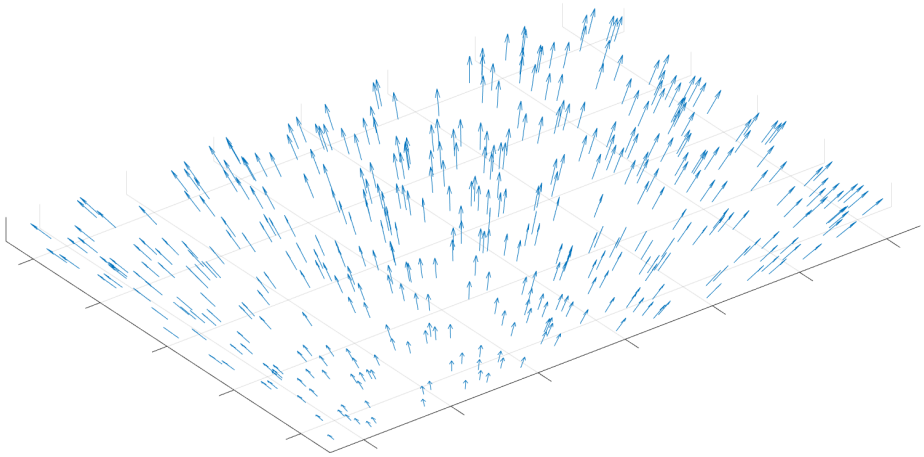

Fig. 4: Distribution of the focusing axes for transparent features of the sparse binary URA mask, used in our prototype, according to the average incoming light directions. Each blue arrow indicates the position and the optical axis direction of the corresponding sub-lens (implemented with FZP). Since the mask is approximately twice larger than the sensor, the average incoming light directions are more inclined on the edges of the mask

4 Reconstruction Pipeline Flowchart

Here we summarize the reconstruction pipeline process into the simple flowchart, see Fig. 5.

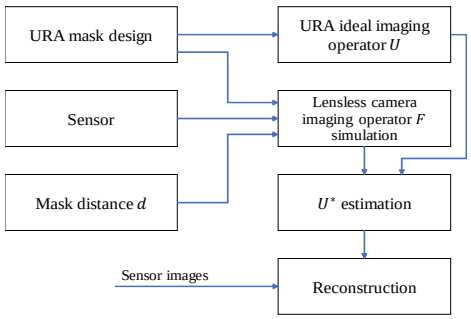

Fig. 5: Here is the flowchart of the reconstruction process. The blocks on the left are used for URA ideal imaging operator U evaluation and for simulation of the system's imaging operator F. After that, final imaging model parameters are estimated and used in the reconstruction block. Note, that all blocks except the Reconstruction block are executed only once