SUPPLEMENT: Inpainting of Sparse Depth Maps from Simulated Monocular Depth-from-Focus on Pixel Processor Arrays

Maciej Lewandowski
University of Manchester
Manchester, UK
maciej.lewandowski@student.manchester.ac.uk

Piotr Dudek
University of Manchester
Manchester, UK
p.dudek@manchester.ac.uk

1. Analysis of focus measure operators

1.1. Single focus measure operator analysis

Table 1 reports an evaluation of 23 focus measure operators, with 16 of them being easily implementable on the SCAMP-5 Pixel Processor array (denoted as * in table 1), while the rest are possible to implement approximately. We generated synthetic focal stacks from the NYUv2 dataset using the thin-lens layerwise simulation described in the main paper. For each operator we computed and stored the following statistics in individual NumPy files (available at the code release) :

1. Frequency: the counts of occurrences for each magnitude of the focus measure operator (values of 0 to 255)
2. Binwise RMSE: the root-mean-squared error between the predicted depth bin (an integer in $\{1, ..., 32\}$ for NYUv2 or $\{1, ..., 64\}$ for DIODE) and the ground truth bin obtained by discretising the continuous depth into 32 (NYUv2) or 64 (DIODE) levels.
3. Normalised RMSE: the RMSE between the predicted depth fraction (an integer in $\{1, ..., 32\}$ divided by 32, or $\{1, ..., 64\}$ divided by 64 for DIODE) and the true depth normalised to [0.0, 1.0].
4. Exact-bin frequency: the number of times that a particular operator with a given amplitude classified the depth bin correctly or within some distance (captured for $R \in \{0, 1, 2, 3\}$).

These per-operator metrics allow us to compare the robustness and accuracy across different focus measure operators.

1.2. Analysis of Tolerance and Join-Operator Statistics

Let d denote the ground-truth depth bin (an integer in $\{1, ..., 32\}$ for NYUv2 and $\{1, ..., 64\}$) for DIODE) and let $\hat{d}_{\mathcal{F}}$ be the predicted depth bin obtained by selecting the frame index at which the absolute value of the focus measure operator $\mathcal{F}$ is maximised. Beyond our per-operator performance in Table 1 we introduce two complementary analyses: relaxed accuracy (accuracy within some distance to ground truth) and sending two focus measure operators.

1.2.1. Depth-from-focus operator accuracy as a function of magnitude (Measurement Confidences)

In the first analysis, rather than insisting on exact bin matches $|\hat{d}_{\mathcal{F}} - d| = 0$ we measure accuracy under relaxed tolerances :

$$|\hat{d}_{\mathcal{F}} - d| \leq R$$

for $R \in \{0, 1, 2, 3\}$. For each operator $\mathcal{F}$ and each R, we compute the fraction of all pixels whose predictions fall within R bins of the true depth. This provides us with more information about the operator's performance: while the best operators achieve only $60 - 70\%$ exact matches ($R = 0$), with a one-bin tolerance ($R = 1$), the accuracy increases to more than 90%.

To understand how reliably each focus measure operator signals depth, we evaluate prediction accuracy at varying tolerance levels (R) as a function of the operator's response magnitude. Although we only show three representative operators (the best-performing three operators from Table 1), other operators follow a similar trend.

Each curve in these plots corresponds to accuracy for operator $\mathcal{F}$ given certain tolerance R (ratio $|\hat{d}_{\mathcal{F}} - d| \leq R$ to all observations) for $R \in \{0, 1, 2, 3\}$. The horizontal axis corresponds to the value of the focus measure operator.

Here we provide some key observations :

- **The metrics themselves are fairly unreliable and provide only coarse approximation of depth**. The strongest metrics have an accuracy of 62% for top 20% measurements for exact bin match. However, relaxing to one-bin tolerance shows much better performance and even for weak measurements, it increases the accuracy to 80-90%.
- **Diminishing return beyond** $R > 1$: while some metrics benefit from this, typically (and as observed on plots 2, 3, 1), accuracy does not increase much for larger tolerances. It seems that for some cases, the depth-from-focus cannot produce accurate depth estimates for simulated focal stacks. These have to be rejected or refined.

name	Acc@80.0% (frac. left)	Acc@90.0% (frac. left)	Acc@95.0% (frac. left)	Acc@99.0% (frac. left)
”modified laplacian”*	**0.621 (16.47%)**	**0.638 (8.48%)**	**0.652 (4.86%)**	0.694 (0.87%)
”diff”*	0.610 (17.73%)	0.636 (8.33%)	0.652 (4.55%)	0.692 (0.98%)
”variance 7x7”	0.602 (18.43%)	0.621 (9.52%)	0.631 (4.88%)	0.654 (0.95%)
”diagonal laplacian”*	0.601 (17.01%)	0.623 (9.84%)	0.649 (4.55%)	**0.695 (0.98%)**
”modified variance”	0.594 (18.43%)	0.622 (9.51%)	0.638 (4.62%)	0.667 (0.90%)
”variance 5x5”	0.587 (19.02%)	0.617 (9.20%)	0.631 (4.99%)	0.658 (0.98%)
”log ng”*	0.576 (19.91%)	0.614 (9.76%)	0.641 (4.88%)	0.694 (0.99%)
”log”*	0.563 (18.42%)	0.602 (9.50%)	0.627 (4.75%)	0.665 (0.98%)
”variance 3x3”*	0.554 (19.69%)	0.603 (9.63%)	0.629 (4.68%)	0.666 (0.91%)
”log2 ng”*	0.546 (18.17%)	0.585 (9.86%)	0.618 (4.71%)	0.669 (0.95%)
”log3 ng”*	0.544 (19.92%)	0.597 (8.56%)	0.620 (4.63%)	0.664 (0.98%)
”weighted modified log”	0.508 (15.57%)	0.567 (8.73%)	0.600 (4.88%)	0.644 (0.94%)
”log3”*	0.497 (14.79%)	0.545 (9.89%)	0.595 (4.75%)	0.641 (0.96%)
”log iso 5x5”*	0.491 (18.55%)	0.555 (9.32%)	0.588 (4.90%)	0.630 (0.98%)
”log 5x5”*	0.489 (17.40%)	0.550 (9.68%)	0.590 (4.87%)	0.632 (0.99%)
”tenenbaum”	0.483 (19.76%)	0.550 (9.95%)	0.595 (4.99%)	0.633 (2.24%)
”log2”*	0.483 (18.38%)	0.560 (9.12%)	0.599 (4.61%)	0.642 (0.96%)
”gra3 ng sum”*	0.398 (17.05%)	0.470 (9.31%)	0.533 (4.89%)	0.617 (0.99%)
”gra3 ng max”*	0.397 (16.99%)	0.469 (9.23%)	0.532 (4.82%)	0.617 (0.97%)
”gra3 sum”*	0.366 (17.83%)	0.445 (9.61%)	0.518 (4.65%)	0.604 (0.97%)
”gra3 max”*	0.366 (17.77%)	0.444 (9.56%)	0.518 (4.61%)	0.604 (0.96%)
”log 7x7”	0.302 (17.56%)	0.381 (9.05%)	0.486 (4.36%)	0.654 (0.92%)
”structure tensor”	0.099 (70.23%)	0.099 (70.23%)	0.099 (70.23%)	0.099 (70.23%)

Table 1. Accuracy vs. Threshold. ”frac. left” expresses the fraction of measurements left when setting a threshold closest to the setpoint. It is impossible to find the exact threshold which corresponds to the setpoint (for example 20%), therefore we pick the threshold which gives the most measurements, less than the setpoint (given setpoint is upper bound). * denotes that an operator is easily implementable on the SCAMP5 system.

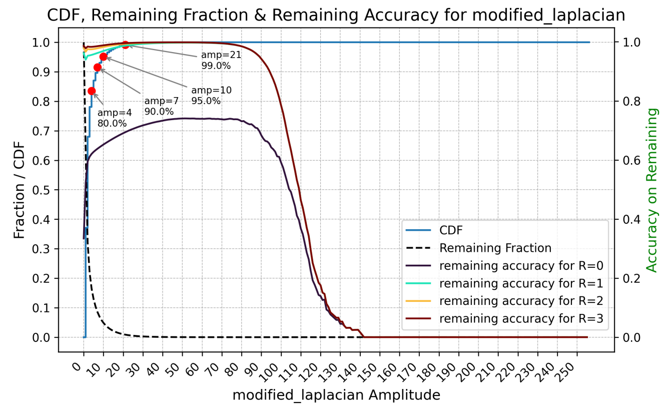

Figure 1. Depth-prediction accuracy vs. response strength for the “modified laplacian” operator. Curves show tolerances $R = 0, 1, 2, 3$.

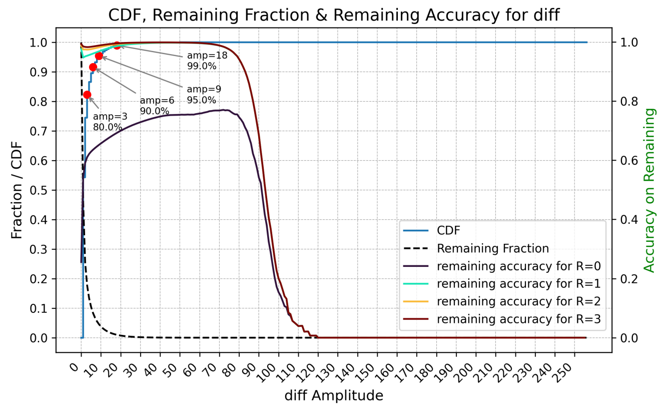

Figure 2. Depth-prediction accuracy vs. response strength for the “diff” operator. Curves show tolerances $R = 0, 1, 2, 3$.

1.2.2. Pairwise metrics

The proposed in-sensor pipeline significantly reduces data transmission. We can leverage SCAMP5 transmission of single bits and transmit 5-bit semi-dense depth images at 20 % density. In the worst case, when transmitting full 5-bit frames (no thresholding, or thresholding and still outputting zeros), this corresponds to 51.2 and 19.7 times less bandwidth for inpainting and image-guided depth completion, when compared to sending a whole stack (32 frames, each

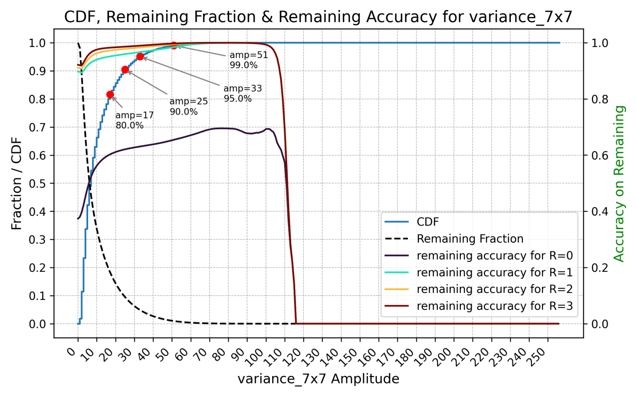

Figure 3. Depth-prediction accuracy vs. response strength for the "variance 7×7" operator. Trends mirror those in Figure 2.

containing 8-bit intensity).

Given the already large bandwidth savings, in the second analysis, we examine the benefits of sending two different focus measure operators. To understand how different operators complement one another, and whether there are benefits of sending two metrics, we capture a four-dimensional contingency tensor for every pair of operators :

$$C \in \mathbb{N}^{4\times256\times256\times4} \tag{1}$$

with axes :

1. 1st axis : Tolerance R (0, 1, 2, 3)
2. 2nd axis : Peak magnitude of operator F_1 (quantized into 256 levels)
3. 3rd axis : Peak magnitude of operator F_2 (quantized into 256 levels)
4. Contingency "matrix" :
   (a) Case 0: $|\hat{d}_{F_1} - d| \leq R \wedge |\hat{d}_{F_2} - d| \leq R$
   (b) Case 1: $|\hat{d}_{F_1} - d| \leq R \wedge |\hat{d}_{F_2} - d| > R$
   (c) Case 2: $|\hat{d}_{F_1} - d| > R \wedge |\hat{d}_{F_2} - d| \leq R$
   (d) Case 3: $|\hat{d}_{F_1} - d| > R \wedge |\hat{d}_{F_2} - d| > R$

Each entry $C[R, a, b, c]$ counts how often, at tolerance R, operator F_1 had peak magnitude a, operator F_2 had peak magnitude b, and the joint correctness fell into case c (both within, only one with, both wrong). Using the C tensor, we compute :

1. Fraction of pixels where both operators agree within R
2. pairwise complementary gain: What is the accuracy benefit of adding a second metric to the first one?

Although individual focus measure operators provide useful (but noisy) depth cues, combining two or more operators can significantly improve reliability, at the cost of transmitting more data. This idea bears resemblance to ensemble methods in machine learning, where a small set of diverse "weak learners" often outperforms any one model on its own, and often achieves excellent performance. For instance, fusing a simple "diff" metric (the difference between unblurred and blurred frames) with a large-kernel-variance metric, known for its robustness. More sophisticated approaches such as discrete cosine transforms, histogram statistics, wavelets or adaptive kernels are also a possibility; however, they are not yet feasible on today's Pixel Processor Array.

We use the contingency tensor described in equation 1 to quantify how two operators F_1 and F_2 work together. From C we compute :

1. Success rate of F_1 when thresholded (SR_{F_1}) : fraction of pixels where F_1 is higher then setpoint threshold and $|\hat{D}_{F_1} - D| \leq R$.
2. Success rate of F_2 when thresholded (SR_{F_2}) : same as above but for F_2
3. Combined accuracy

$$\mathrm{OR}_{\mathrm{F_1,F_2}} = \frac{\text{\# of pixels where either } F_1 \text{ or } F_2 \text{ are within } R}{\text{pixels where } F_1 \text{ or } F_2 \text{ are confident}}$$

4. Gain: which measures additional accuracy obtained by using both operators versus the best single operator, defined as $\mathrm{Gain}_{\mathrm{F_1,F_2}} = OR_{F_1,F_2} - max(SR_{F_1}, SR_{F_2})$

To clarify how these metrics are computed for the previously described tensor C, we provide a NumPy implementation in Listing 1.

Listing 1. Computation of pairwise metrics

```python
a, b = metrics[i], metrics[j]
pair = f"{a}_vs_{b}"

# shape (256,256,4)
C     = pairwise[pair][radius]

th1   = rpct_metric_value[setpoint][a]
th2   = rpct_metric_value[setpoint][b]

slA = C[th1:, :, :]  # rows >= th1
n11, n10, n01, n00 = slA.sum(axis=(0, 1))
NA    = n11 + n10 + n01 + n00
srA   = (n11 + n10) / NA if NA else np.nan

slB = C[:, th2:, :]  # cols >= th2
n11, n10, n01, n00 = slB.sum(axis=(0, 1))
NB    = n11 + n10 + n01 + n00
srB   = (n11 + n01) / NB if NB else np.nan

# A confident
part1 = C[th1:, :, :].sum(axis=(0, 1))

# B confident, A not
part2 = C[:th1, th2:, :].sum(axis=(0, 1))
n11, n10, n01, n00 = part1 + part2
N_or  = n11 + n10 + n01 + n00
any_s = (n11 + n10 + n01) / N_or if N_or
    else np.nan
gain  = any_s - np.nanmax([srA, srB])
```

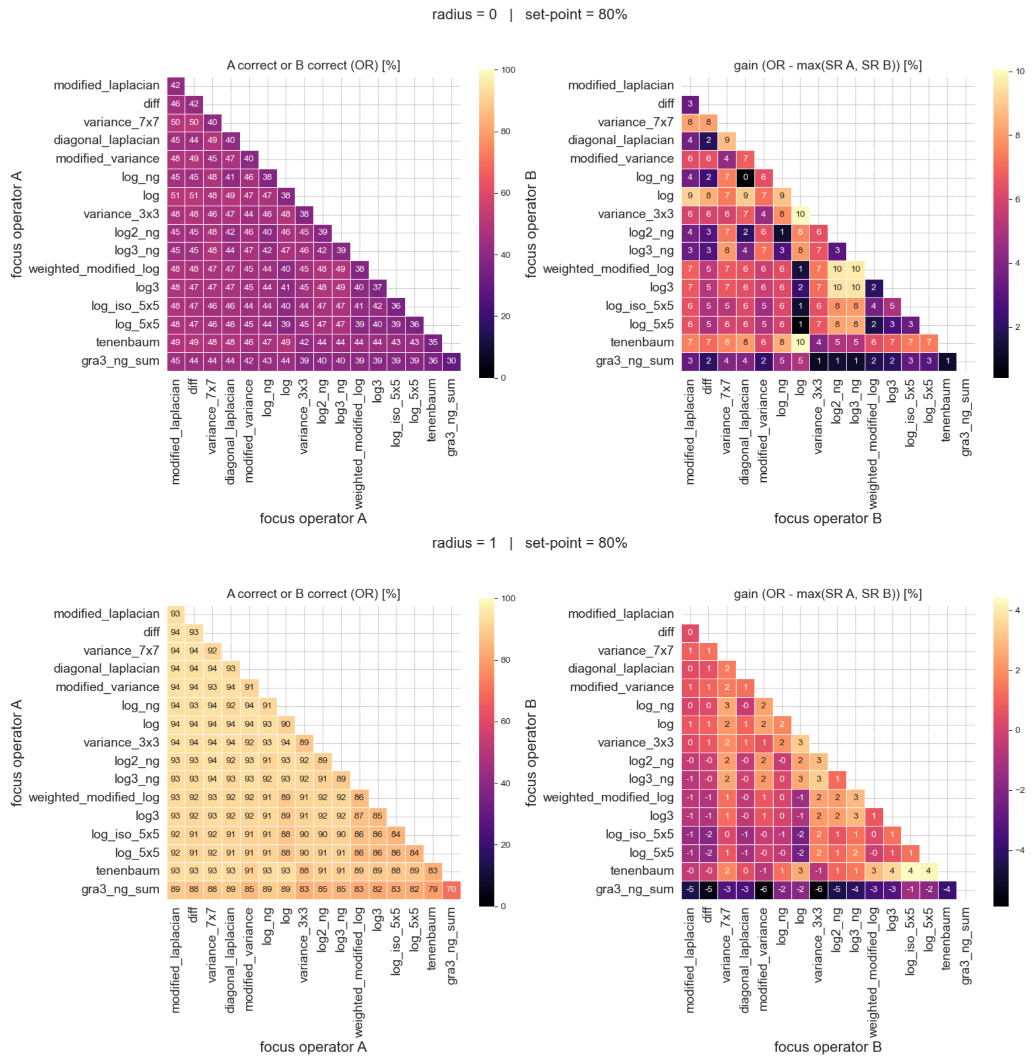

Figure 4. Pairwise success rates for the 80th percentile threshold and $R = 0$ (top) and $R = 1$ (bottom). The left panels plot the percentage of cases in which at least one of two focus-measure operators (A or B) is correct (OR), with the diagonal showing the percentage of correct predictions when the other one is ruled out (computed from pairwise statistic). Right figures show the $Gain$ which is the % improvement of using a pair of metrics versus one which was the most successful.

Pairwise analysis of these metrics, for $R = 0$ and $R = 1$ with percentile thresholds of 80% and 90%, are provided in Figures 4 and 5. In some cases, performance is improved (sometimes by 10 %), which provides an interesting alternative for the Pixel Processor Array and further motivates inpainting using neural networks, as reconstructing a full depth map from multiple, semi-dense focus measure operators computed on-sensor seems like a non-trivial problem. Although the dual-operator approach could marginally improve raw depth cues, our experimentation showed that a single, well-chosen operator, refined by a neural network, already achieves high accuracy. Consequently, the exploration of multiple focus measures is left for future exploration.

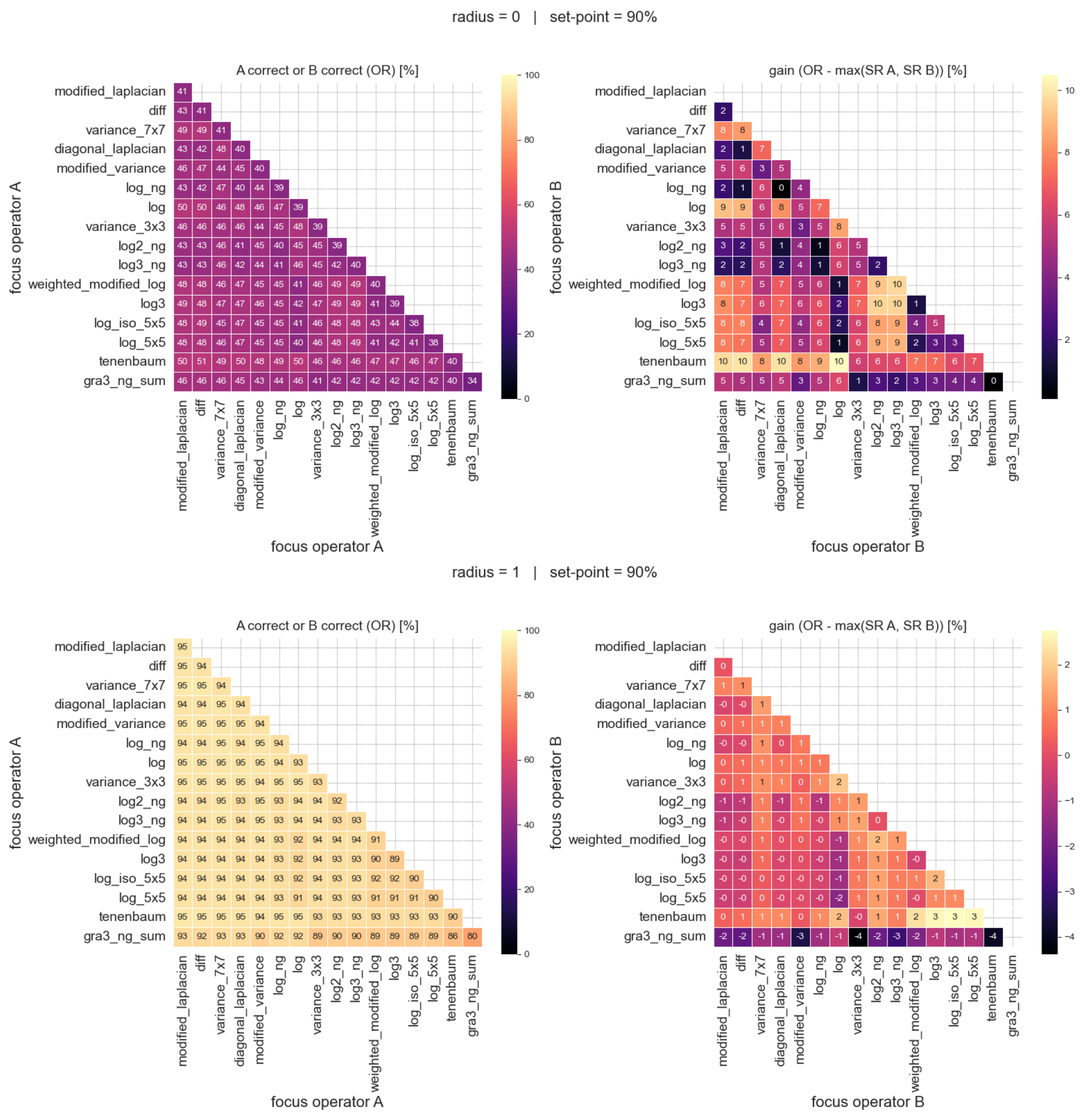

Figure 5. Pairwise success rates for the 90th percentile threshold and $R = 0$ (top) and $R = 1$ (bottom). The left panels plot the percentage of cases in which at least one of two focus-measure operators (A or B) is correct (OR), with the diagonal showing the percentage of correct predictions when the other one is ruled out (computed from pairwise statistic). Right figures show the $Gain$ which is the % improvement of using a pair of metrics versus one which was the most successful.

1.3. In-Depth Failure Analysis on a Single Frame

The bin-level statistics of the previous subsections quantify *how often* focus cues miss the true depth. In this section, we illustrate on example simulation, when are these measurements not accurate.

We selected one NYUv2 frame and rendered a high-resolution focal stack (200 simulated layers, denoted as K in the main paper, 100 focus planes, denoted as N in the main paper). Figure 6 contains six panels:

(a) **Ground truth vs. depth-from-focus prediction.** Left: metric-depth map. Right: unthresholded Laplacian-of-Gaussian (LoG) peak index (0–99).

(b) **Absolute error heat-map.** Pixels where $|\hat{D}_{\text{LoG}} - D|$ are colour-coded, then overlaid on the RGB image. This is performed for 80% setpoint (threshold for Laplacian of Gaussian is set in such a way that, on average, 20% of pixels should have measurements).

(c) **Multi-peak ambiguity.** A common case is where multiple peaks are produced. The wrong peak is selected, resulting in an error of more than 20 bins.

(d) **Neighbourhood contrast.** Four pixels away, a single dominant peak aligns perfectly with the ground truth, demonstrating the challenge for refinement that must distinguish cases (c) from (d).

(e) **Ideal case.** High-contrast corner yields a large and sharp peak, which leads to an exact match.

(f) **Textureless region.** Flat wall returns no reliable focus cue; LoG picks a random shallow bin.

Panels (c)–(f) typify the three dominant failure modes we observe across the dataset:

- 1. *Peak selection errors* (multi-modal traces).
- 2. *False positives near correct pixels* that can mislead naive smoothers.
- 3. *Missing data in low-texture areas*.

In the analysed images, most errors are at the edges. In practical applications, this might not necessarily be the problem, as nearby pixels are correct. Furthermore, depth information on edges is difficult to capture accurately too.

These modes justify our refinement stage: a CNN or belief-propagation solver must (i) suppress spurious peaks, (ii) respect sharp discontinuities, and (iii) inpaint large voids—all while relying only on semi-dense, noisy cues.

1.4. Implementation of Focus measure operators

In the following subsection, we describe how the operators were computed exactly.

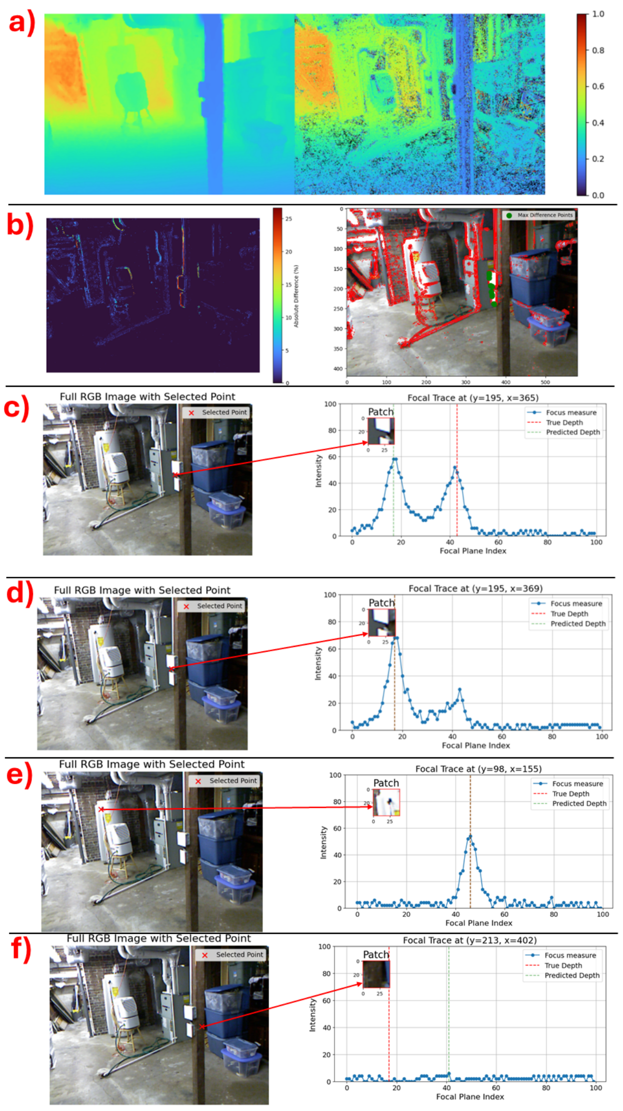

Figure 6. Analysis of focal measure operator on a reconstructed image from NYUv2.

Algorithm 1 Compute Modified Laplacian metric

```
Require: Input image $I$
Ensure: Processed data stored in $metric$
1: $f_x \leftarrow \begin{bmatrix} -1 & 2 & -1 \end{bmatrix}$
2: $f_y \leftarrow \begin{bmatrix} -1 \\ 2 \\ -1 \end{bmatrix}$
3: $I_x \leftarrow f_x * I$
4: $I_y \leftarrow f_y * I$
5: $metric \leftarrow |I_x| + |I_y|$
```

Algorithm 2 Compute ”diff” metric

```
Require: Input image $I$
Ensure: Processed data stored in $metric$
1: $blurred \leftarrow blur(I)$
2: $metric \leftarrow I - blurred$
```

Algorithm 3 Compute Diagonal Laplacian metric

```
Require: Input image $I$
Ensure: Processed data stored in $metric$
1: $S1 \leftarrow |k_1 * I|$
2: $S2 \leftarrow |k_2 * I|$
3: $metric \leftarrow max(S_1, S_2)$
```

Algorithm 4 Compute ”LoG”

```
Require: Input image $I$
Ensure: Processed data stored in $metric$
1: $k_1 \leftarrow \begin{bmatrix} 0 & -1 & 0 \\ 0 & 4 & -1 \\ - & -1 & 0 \end{bmatrix}$
2: $blurred \leftarrow blur(I)$
3: $metric \leftarrow |k_1 * blurred|$
```

Algorithm 5 Compute ”LoG ng”

```
Require: Input image $I$
Ensure: Processed data stored in $metric$
1: $k_1 \leftarrow \begin{bmatrix} 0 & -1 & 0 \\ 0 & 4 & -1 \\ - & -1 & 0 \end{bmatrix}$
2: $metric \leftarrow |k_1 * I|$
```

Algorithm 6 Compute ”modified variance”

```
Require: Input image $I$
Ensure: Processed data stored in $metric$
1: $blurred \leftarrow blur(I);$
2: $blurred_{sq} \leftarrow blur(I^2);$
3: $metric \leftarrow blurred_{sq} - blurred^2$
```

Algorithm 7 Compute ”variance 7x7”

```
Require: Input image $I$
Ensure: Processed data stored in $metric$
1: $k_1 \leftarrow ones(7, 7)/49$    ▷ Create a 7x7 matrix of ones, and then normalize
2: $I_{sq} \leftarrow I^2$
3: $mean_{sq} \leftarrow k_1 * I_{sq}$
4: $mean \leftarrow k_1 * I$
5: $metric \leftarrow mean_{sq} - mean^2$
```

Algorithm 8 Compute ”variance 5x5”

```
Require: Input image $I$
Ensure: Processed data stored in $metric$
1: $k_1 \leftarrow ones(5, 5)/25$    ▷ Create a 5x5 matrix of ones, and then normalize
2: $I_{sq} \leftarrow I^2$
3: $mean_{sq} \leftarrow k_1 * I_{sq}$
4: $mean \leftarrow k_1 * I$
5: $metric \leftarrow mean_{sq} - mean^2$
```

Algorithm 9 Compute ”variance 3x3”

```
Require: Input image $I$
Ensure: Processed data stored in $metric$
1: $k_1 \leftarrow ones(3, 3)/9$ ▷ Create a 3x3 matrix of ones, and then normalize
2: $I_{sq} \leftarrow I^2$
3: $mean_{sq} \leftarrow k_1 * I_{sq}$
4: $mean \leftarrow k_1 * I$
5: $metric \leftarrow mean_{sq} - mean^2$
```

Algorithm 10 Compute ”weighted modified LoG”

```
Require: Input image $I$
Ensure: Processed data stored in $metric$
1: $k_1 \leftarrow \begin{bmatrix} -1 & -2 & -1 \\ -2 & 12 & -2 \\ -1 & -2 & -1 \end{bmatrix}$
2: $blurred \leftarrow blur(I)$
3: $metric \leftarrow |k_1 * blurred|$
```

Algorithm 11 Compute ”LoG_iso_5x5”

```
Require: Input image $I$
Ensure: Processed data stored in $metric$
1: $k_1 \leftarrow \begin{bmatrix} -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & 24 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 \end{bmatrix}$
2: $blurred \leftarrow blur(I)$
3: $metric \leftarrow |k_1 * blurred|$
```

Algorithm 12 Compute ”LoG2”

```
Require: Input image $I$
Ensure: Processed data stored in $metric$
1: $k_1 \leftarrow \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$
2: $blurred \leftarrow blur(I)$
3: $metric \leftarrow |k_1 * blurred|$
```

Algorithm 13 Compute ”LoG 5x5”

```
Require: Input image $I$
Ensure: Processed data stored in $metric$
1: $k_1 \leftarrow \begin{bmatrix} 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & -2 & -1 & 0 \\ -1 & -2 & 16 & -2 & -1 \\ 0 & -1 & -2 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 \end{bmatrix}$
2: $blurred \leftarrow blur(I)$
3: $metric \leftarrow |k_1 * blurred|$
```

Algorithm 14 Compute ”GRA3 Sum” metric

```
Require: Input image $I$
Ensure: Processed data stored in $metric$
1: $G_x \leftarrow \begin{bmatrix} -1 & 0 & -1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$
2: $G_y \leftarrow \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$
3: $I_x \leftarrow G_x * I$
4: $I_y \leftarrow G_y * I$
5: $metric \leftarrow I_x + I_y$
```

Algorithm 15 Compute ”GRA3 max” metric

```
Require: Input image $I$
Ensure: Processed data stored in $metric$
1: $G_x \leftarrow \begin{bmatrix} -1 & 0 & -1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$
2: $G_y \leftarrow \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$
3: $I_x \leftarrow G_x * I$
4: $I_y \leftarrow G_y * I$
5: $metric \leftarrow max(I_x, I_y)$
```

Algorithm 16 Compute ”GRA3 Sum” metric

```
Require: Input image $I$
Ensure: Processed data stored in $metric$
1: $G_x \leftarrow \begin{bmatrix} -1 & 0 & -1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$
2: $G_y \leftarrow \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$
3: $blurred \leftarrow blur(I)$
4: $I_x \leftarrow G_x * blurred$
5: $I_y \leftarrow G_y * blurred$
6: $metric \leftarrow I_x + I_y$
```

Algorithm 17 Compute ”GRA3 Sum” metric

```
Require: Input image $I$
Ensure: Processed data stored in $metric$
1: $G_x \leftarrow \begin{bmatrix} -1 & 0 & -1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$
2: $G_y \leftarrow \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$
3: $blurred \leftarrow blur(I)$
4: $I_x \leftarrow G_x * blurred$
5: $I_y \leftarrow G_y * blurred$
6: $metric \leftarrow max(I_x, I_y)$
```

2. Real World Experiment

To our knowledge, there are not many datasets with real-world focal stacks and high-quality ground-truth depth. One of the major reasons is the fact that capturing and storing a single stack of 32 (or more) images per single datapoint is very storage-intensive. In our case, generating synthetic 32-frame focal stacks from NYUv2 at 256×256 resolution requires roughly 160 GB of disk space for 5-bit depth measurements. This, of course, also scales exponentially with respect to depth resolution. The dataset of Herrmann et al. [1] is the most promising dataset for this work. The dataset contains 510 real focal stacks for indoor and outdoor scenes, each captured as a 48-frame stack plus an all-in-focus reference image and reconstructed depth map.We include a supplemental video of a single focal sweep to illustrate the data quality.

Unfortunately, this smartphone-based capture exhibits severe focus breathing, where internal lens movement shifts the image plane as focus changes. This results in a significant change in the field of view and pixel movement. Because pixels move between the frames, the sharpness comparisons become unreliable and introduce substantial noise (or make a completely wrong prediction) into any depth-from-focus estimate. The breathing is an artefact of the opto-mechanical setup, involving the displacement of the lens elements along the optical axis. In contrast, the liquid lens operates through the change of lens curvature, helping to keep the pixel grid more stable across the focus range. While this effect still exists, is limited, and mitigated by thresholding, as pointed out by Martel et al. [3].

We evaluated a simple Laplacian of Gaussian (”log”) focus measure operator on the Herrmann data (Figure 7). While in some regions the predicted depth is roughly correct, a large portion of the image is missclassified, and published ”ground-truth” depth maps suffer large reconstruction errors and are not reliable enough for training or evaluation. Given these two issues — focus breathing artefacts and noisy reference depths —we defer a full real-world PPA liquid lens experiment to future work and rely on our synthetic stacks here.

2.1. PPA with liquid lens

To complement our simulation study, we conducted a small experiment demonstrating on-sensor DFF with an SCAMP5 PPA with an electrically tunable liquid lens. During a near to far optical-power sweep, the PPA computes a focus measure (LoG), records the peak index (depth bin) and peak magnitude (confidence), and transmits the resulting semi-sparse map to a host PC. For qualitative reference, we also capture RGB-D using an Intel RealSense D435i. The scene RGB, the D435i depth, and the PPA DFF output are shown in Fig. 8.

Overall, high-contrast edges and textured regions (e.g.

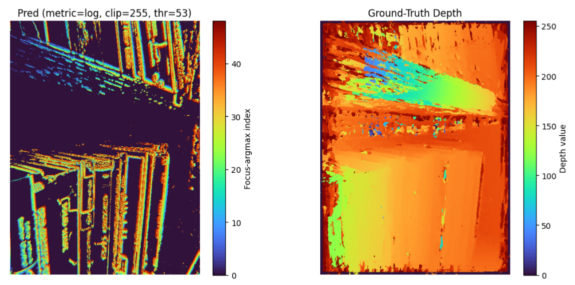

Figure 7. On the left is the result of the depth-from-focus algorithm using the Laplacian of Gaussian method on a real-world focal stack. On the right is a provided ground truth depth map.

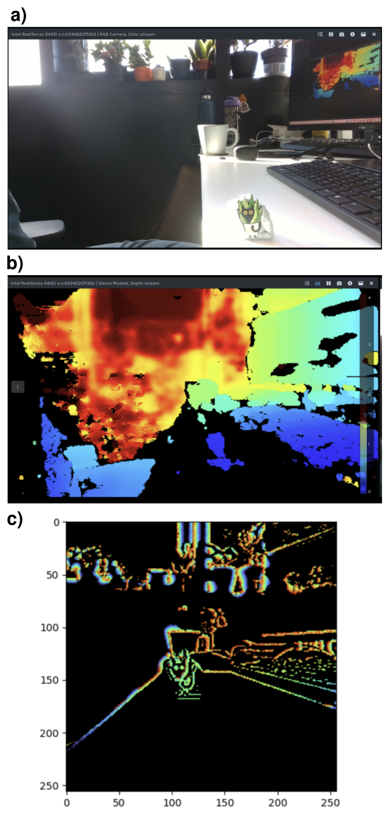

Figure 8. a) Reference RGB (intel RealSense D435i) b) Reference depth (used for qualitative comparison). c) Semi-sparse depth from DFF SCAMP5 algorithm from near to far sweep.

desk edge, keyboard, mug outline, cat pin) produce a reasonable depth measurement which is consistent with the D435i reference. The depth map computed on the PPA however exhibits errors manifesting themselves as a banded, rainbow-like pattern in regions of high contrast, this is due to combination of focus breathing, optics/sensor mismatch and imperfect alignment and challenging lighting. The investigated threshold $\theta = 32$, did not remove all of the ”rainbow-band” artifacts. These issues however, have been mitigated to good extent by Martel et al. [3] by more careful thresholding.

3. Comparison to classical filling

In-sensor depth-from-focus on a PPA naturally produces semi-dense depth maps that must be completed externally to yield fully dense reconstructions. While neural networks or belief propagation has the potential to be mapped onto the PPA grid (a direction for future hardware work), most traditional inpainting algorithms remain CPU- or GPU-bound, particularly variational methods.

A notable lightweight candidate for on-sensor deployment is the approach, developed by Ku et al. [2], which relies on morphological operations and convolutions. Using simple operations, which can be performed on PPA, this method achieved state-of-the-art results on the KITTI LiDAR completion benchmark, running in real time on a CPU [2]. Although relatively large kernels may challenge current PPAs, the algorithm's low computational effort makes it an interesting direction for future in-sensor depth completion.

However, when applied to our depth-from-focus data, this classical approach has several limitations :

- **Unfiltered noise propagation**: LiDAR returns are precise and roughly, uniformly spaced, enabling reliable morphological filling. In contrast, focus-based measurements are only a rough approximation (often by a single bin and there is a large portion of measurements that vary by more) and cluster in textured regions. Without any measurement rejection mechanisms or refinement based on confidence, the algorithm propagates these noisy cues across the scene.
- **Irregular sample patterns.** LiDAR grids cover the frame much more evenly, but the focus samples concentrate along edges and high-contrast textures, leaving large gaps. Morphological or any spatial propagation over such non-uniform data either oversmooths or leaves large holes, reducing completeness.
- **Loss of fine detail**. It was observed that there is a significant loss of finer features. This motivates the neural network refinement and BP, which preserves high-frequencies much better.

Figure 9 demonstrates these effects. The top row shows the all-in-focus RGB image of scene (left) and the semi-dense depth map from a Laplacian-of-Gaussian focus measure operator (right). The bottom row compares (left) the output of Ku et al. inpainting and (right) the ground-truth depth. The classical method attempts to complete the map, but it fails to reject noisy measures from depth-from-focus and smooths away fine scene details. This further underscores the need for learning-based or belief-propagation completion on PPA-derived data.

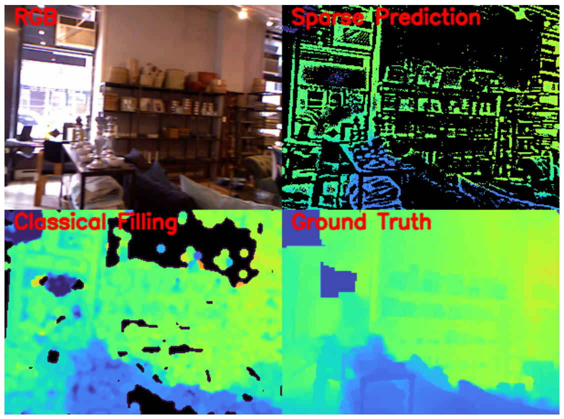

Figure 9. On the top left corner, we have the original RGB image. In the top right corner, the output of the simulated depth-from-focus algorithm is displayed, obtained using the Laplacian of Gaussian focus measure operator. on bottom left, reconstructed depth using the algorithm developed by [2]. on Bottom right, a ground truth depth map is displayed.

4. Factor graph for depth inpainting

Here, we describe how the belief propagation framework is applied to the depth filling problem, leveraging the sparse outputs generated by the PPA-based depth-from-focus algorithm. We employ the factor graph structured as shown in the main paper's Figure 3. Each variable node $V_{x,y}$ denotes a belief about a depth at pixel (x, y), and is represented by an N-dimensional probability distribution ($N = 32$). There are two types of factors: unary factors $F^{(m)}_{x,y}$ representing depth measurements, and pairwise smoothing factors $F^{(s)}$, which try to make the variables take similar values. Unary factors are not created for pixels that do not have a depth measurement. The factors encode the coupling strength between variables (with higher values indicating stronger agreement on a particular assignment). The smoothing function is computed as :

$$f^{(s)}(v_1, v_2) = \begin{cases} e^{-c_1\cdot(v_1-v_2)^2} & \text{if } |v_1 - v_2| \leq \delta, \\ e^{-c_1\cdot\delta\cdot(2\cdot(v_1-v_2)-\delta)} & \text{otherwise} \end{cases} \tag{2}$$

where $c_1 = 0.2$ and $\delta = 1$ and v_1, v_2 are the values of the two variables connected to the factor.

Discrete depth measurements from depth-from-focus, $m(x, y)$, are mapped to unary factors. In the proposed approach, we also use the confidence $\alpha(x, y)$ for each depth measurement. Consequently, the unary function will be defined as :

$$f^{(m)}_{x,y}(v) = e^{-c_2\cdot\sqrt{\frac{\alpha(x,y)}{128}}\cdot(v-m(x,y))^2} \tag{3}$$

where $c_2 = 0.5$ and v are the values of the variable connected to the factor. We run this with fixed damping of 0.5, and Temperature set to 0.

Dataset	Sparsity	UNET					BiSeNetV2					LETNet				
Metric		$\Delta_{rel}MAE$	$\Delta_{rel}RMSE$	$\Delta\delta_{1.25}$	$\Delta\delta_{1.25^2}$	ΔPSNR	$\Delta_{rel}MAE$	$\Delta_{rel}RMSE$	$\Delta\delta_{1.25}$	$\Delta\delta_{1.25^2}$	ΔPSNR	$\Delta_{rel}MAE$	$\Delta_{rel}RMSE$	$\Delta\delta_{1.25}$	$\Delta\delta_{1.25^2}$	ΔPSNR
NYUv2	80%	-77.20%	-72.30%	0.243	0.098	11.19 dB	-77.80%	-72%	0.308	0.12	9.17 dB	-75.81%	-69.58%	0.374	0.1816	10.287 dB
	90%	-71.60%	-66.80%	0.229	0.088	9.76 dB	-68.20%	-62.79%	0.278	0.111	8.07 dB	-64.25%	-59.67%	0.41	0.205	7.4 dB
	95%	-63%	-58.90%	0.204	0.078	7.78 dB	-73.50%	-66.90%	0.293	0.114	6.5 dB	-71.34%	-65.62%	0.36	0.18	9.43 dB
DIODE INDOORS	80%	-81.10%	-73.00%	0.494	0.306	10.61 dB	-72.94%	-66.30%	0.428	0.273	8.48 dB	-77.54%	-71.18%	0.5052	0.382	8.73 dB
	90%	-78.90%	-70.60%	0.47	0.303	10.13 dB	-72.35%	-65.28%	0.423	0.265	8.73dB	-59.00%	-50.66%	0.438	0.3678	5.52 dB
	95%	-72.80%	-64.10%	0.423	0.29	8.42 dB	-64.11%	-57.51%	0.373	0.244	5.72 dB	-63.19%	-55.72%	0.362	0.317	5.6 dB
DIODE OUTDOORS	80%	-55.09%	-41.17%	0.305	0.25	4.92 dB	-55.10%	-41.62%	0.274	0.261	5.7 dB	-57.53%	-41%	0.32	0.2	5.8 dB
	90%	-55.18%	-37.88%	0.3069	0.246	4.75 dB	-77.67%	-67.67%	0.236	0.239	11.83 dB	-57.89%	-35.72%	0.231	0.121	4.3 dB
	95%	-81.81%	-65.53%	0.27	0.222	11.33 dB	-77.50%	-67.51%	0.215	0.231	11.53 dB	-51.00%	-35.72%	0.232	0.126	4.8 dB

Table 2. Recorded improvement for 3 network: UNET [4], BiSeNetV2 [7] and LETNET [6]. The table presents the differences in performance between the network using RGB and noisy semi-sparse depth, compared to the same network utilising only RGB data. This was recorded for 3 degrees of sparsity: $\approx 80\%$, $\approx 90\%$ and $\approx 95\%$, and 5 measures: Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), $\delta_{1.25}$, $\delta_{1.25^2}$ and peak signal to noise ration (PSNR)

In total, there are 65536 variables (one per pixel), 130560 pairwise smoothing factors and a variable number of unary factors (depending on the sparsity of the depth-from-focus data). We run the sum-product BP for a fixed number of iterations, allowing the solution to mostly settle, obtaining the final probability distributions of the variables. To obtain a depth value for a pixel from the computed probability distribution of the corresponding variable we use point estimates such as the most probable value, median value , or the expected value of the distribution. It is also possible to assign an uncertainty value to the depth estimation, for instance by computing entropy [5].

In the first step, each variable node sends a message to its neighboring factor nodes. The message from variable V_i to factor f_s is given by equation 4 :

$$\mu_{V_i \to f_s}(V_i) = \prod_{h \in n(V_i)/\{f_s\}} \mu_{h \to V_i}(V_i) \tag{4}$$

where $n(\cdot)$ denotes the neighborhood function, which takes a node and returns all adjacent nodes in the graph.

In the next step, the algorithm computes factors for variables messages. This step involves transmitting messages from variables to factors, which can be interpreted as a 'prior belief' about the variable that the factors will refine. The following equation will compute this:

$$\begin{aligned}\mu_{f_s \to V_i}(V_i) = \sum_{v_1, v_2, \ldots, v_{C_s}} \Bigg(& f_s(V_i, v_1, v_2, \ldots, v_{C_s}) \\ & \times \prod_{y \in n(f_s) \setminus \{V_i\}} \mu_{y \to f_s}(y) \Bigg)\end{aligned} \tag{5}$$

Finally, the belief (or marginal probability distribution) at each variable node is computed by combining all incoming messages as in equation 6:

$$g_i(V_i) = \prod_{h \in n(V_i)} \mu_{h \to V_i}(V_i) \tag{6}$$

5. Improvement for Neural Networks

Table 2 summarises the Experiment 1 deltas obtained for every dataset, network and evaluation metric. Each value represents the benefit of concatenating RGB data with depth-from-focus cues, demonstrating consistent improvement across all sparsity levels and metrics.

References

[1] Charles Herrmann, Richard Strong Bowen, Neal Wadhwa, Rahul Garg, Qiurui He, Jonathan T. Barron, and Ramin Zabih. Learning to autofocus. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2227–2236. IEEE. 9

[2] Jason Ku, Ali Harakeh, and Steven L. Waslander. In defense of classical image processing: Fast depth completion on the CPU. 10

[3] Julien NP Martel, Lorenz K Müller, Stephen J Carey, Jonathan Müller, Yulia Sandamirskaya, and Piotr Dudek. Real-time depth from focus on a programmable focal plane processor. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 65(3):925–934, 2017. Publisher: IEEE. 9, 10

[4] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pages 234–241. Springer, 2015. 11

[5] Jian Sun, Nan-Ning Zheng, and Heung-Yeung Shum. Stereo matching using belief propagation. *IEEE Transactions on pattern analysis and machine intelligence*, 25(7):787–800, 2003. Publisher: IEEE. 11

[6] Guoan Xu, Juncheng Li, Guangwei Gao, Huimin Lu, Jian Yang, and Dong Yue. Lightweight real-time semantic segmentation network with efficient transformer and CNN. *IEEE Transactions on Intelligent Transportation Systems*, 24(12): 15897–15906, 2023. Publisher: IEEE. 11

[7] Changqian Yu, Changxin Gao, Jingbo Wang, Gang Yu, Chunhua Shen, and Nong Sang. Bisenet v2: Bilateral network with guided aggregation for real-time semantic segmentation. *International journal of computer vision*, 129:3051–3068, 2021. Publisher: Springer. 11