Supplemental Material for Paper: Efficient Multi-Frequency Phase Unwrapping using Kernel Density Estimation

Felix Järemo Lawin, Per-Erik Forssén, and Hannes Ovrén

Computer Vision Laboratory, Linköping University, Sweden
{felix.jaremo-lawin, per-erik.forssen, hannes.ovren}@liu.se

1 Introduction

This document is a supplement to the ECCV submission "Efficient Multi-Frequency Phase Unwrapping using Kernel Density Estimation". Here we describe the used parameter tuning and dataset generation in more detail. We also provide more examples of how the method works in difficult cases such as large-depth scenes and outdoor scenes. We also give more examples of meshes produced by KinFu when fed with output from the proposed KDE method and from *libfreenect2*.

2 Dataset generation

2.1 Ground Truth for Unwrapping

We have generated three ground truth datasets, that are used to quantitatively evaluate the correctness of the phase unwrappings. The accuracy of the ground truth must be good enough to tell a correct unwrapping from an incorrect one. As described section 3.2 in the paper [1] we have 30 unwrapping candidates in an 18.75m range. Thus, the distance between the candidates is on average 60cm. To ensure that no incorrect unwrappings are accidentally counted as inliers, we require an accuracy of at least half the candidate distance, i.e. better than 30cm.

The required accuracy can easily be met using the Kinect sensor itself. By fusing many frames from the same camera pose, we can reduce the amount of unwrapping errors, and also increase the accuracy in correctly unwrapped measurements to a minimum. By also fusing data from multiple poses we can detect and suppress multipath responses, which vary with camera position.

In practise we implement this as follows. For a given scene we place the camera in a 3×3 positions, with different elevation, and sideways position. During capture, the Kinect is mounted on a tripod, and 100 frames in each pose are acquired, and fused to a single frame using per-pixel expectation maximization (EM) of 4 Gaussian mixture models (GMM) [2]. In the fusion, each sample is weighted with its unwrapping likelihood, see equation 15 in the paper. The GMM is initialized by a channel encoded temporal average of the samples using 32 channels and a $\cos^2$-kernel [3]. The average standard deviation after the GMM step is between 4cm and 10cm.

Fig. 1. Ground truth generation for **kitchen** dataset: Left: Single depth frame without outlier rejection. Large depth values on table are incorrect, and due to multi-path effects. Center: Single view depth fusion of 100 frames. Right: Final ground truth image from a multiview depth fusion of 9 single view depth fusion images, which are registered and splatted into the chosen reference view. Inconsistent pixels are suppressed, here highlighted in green.

The acquired depth values are then used to produce a point cloud for each camera pose. These point clouds are then aligned using GMM based RGB-D point-cloud alignment [4], and finally projected and splatted [2] into the depth image of a chosen reference view. Each pixel now contains a set of depth values, each with a mixture weight from the EM procedure, and from the splatting, a spatial Gaussian weight on the distance to the pixel. The output depth is selected as the most probable depth point cluster mean from a mean-shift filtering procedure [2] using a Gaussian kernel with $\sigma = 7.5$cm. In 98% of the pixels we had 4 or more camera poses within the 7.5cm radius, and thus the accuracy of the estimate is $7.5/\sqrt{4} = 3.75$ or better, using the reasonable assumption that data transferred from different camera poses are independent. Points where the mean-shift KDE value is low are suppressed. An example of the ground truth generation for the **kitchen** dataset is shown in figure 2.1. As can be seen the noisy pixels in the corners of the image as well as the multipath pixels on the table is either corrected or suppressed.

3 Parameter tuning

The parameters of the likelihood weight in our method are tuned by evaluating many values along each parameter axis on the **library** dataset. The following parameters were found to maximize the number of inliers at a 1% outlier rate:

- the kernel scale h in the KDE kernel was set to $h = 0.1547$.
- the spatial support r was set to $r = 5$. (the Gaussian has a spatial support of $(2r + 1) \times (2r + 1)$ and $\sigma = r/2$.)
- the number of hypotheses $|\mathcal{I}|$ was set to 2.
- the scalings s_1 and s_2 in the unwrapping likelihood, and the phase likelihood were set to $s_1 = \sqrt{2}$, and $s_2 = \sqrt{3}$. (in practise, the parameters $s_1^2 = 2$ and $s_2^2 = 3$ are used on the GPU, to save floating point operations).

– the coefficients $(\gamma_0, \gamma_1, \gamma_2)$ in the amplitude mapping when the bilateral filter is used were fitted using least squares as discussed in the main paper [1].

Below we discuss the behaviour of these parameters. In figure 2 we also provide plots for a selection of values that deviate from the chosen ones along one axis at a time.

the kernel scale h: This parameter should be adapted to the distance between hypotheses, which is roughly the maximal range divided by 30 for the Kinect v2. From this base the parameter could be varied and the performance could be measured on the tuning dataset, see figure 2. As can be seen the difference is small between 3 of the curves, however the proposed parameter value of $h = 0.1547$ is marginally better.

spatial support r: As discussed in the main paper, larger the size of the spatial support the slower the calculations. The setting with the smallest spatial support is inferior to the others in performance. There seems to be an upper limit to the improvement of performance as the spatial support gets larger. The proposed spatial support of 11×11, i.e. $r = 5$ and $\sigma = 2.5$ (always set to $r/2$), is a good trade off, which also results in a confidence that is better at ordering the pixels for thresholding. For better speed however, a smaller spatial support may also be interesting.

the likelihood scales s_1 **and** s_2: These parameters should correspond to the standard deviations of the unwrapping error and the phase measurements respectively. By using the statistics of these on the tuning dataset and the plots in figure 2, reasonable values were found.

4 Examples of output depth maps

In the main paper we gave quantitative measures of how the proposed KDE method improves over *libfreenect2* on large depth scenes, when using the full depth range of 18.75m. Here we complement this by providing a few qualitative examples of performance for the two methods. In figure 3 we compare the output from the two methods on two large-depth indoor scenes, and on two outdoor scenes. In the third row we can notice that the bike leaning on the wall is almost removed in the *libfreenect2* output while detailed structures are visible in the output from the KDE method. See also the supplied video for more comparative examples of performance.

As can bee seen in the second row of figure 3 there seems to be a fattening artifact around edges of foreground objects. This only seems to occur when the background is noisy and partly suppressed such as on the the plant and the grid railing in this scene. However, this is merely a defect caused by multi-path interference and leakage in the sensor itself. In figure 4 it can be seen that the depth output from the *libfreenect2* without outlier rejection has the same fattening artifacts, which verifies that this is not caused by our method. For the full *libfreenect2* method many of these pixels are suppressed, but not all of them as can be seen in figure 3. One way to reduce the impact of fattening in

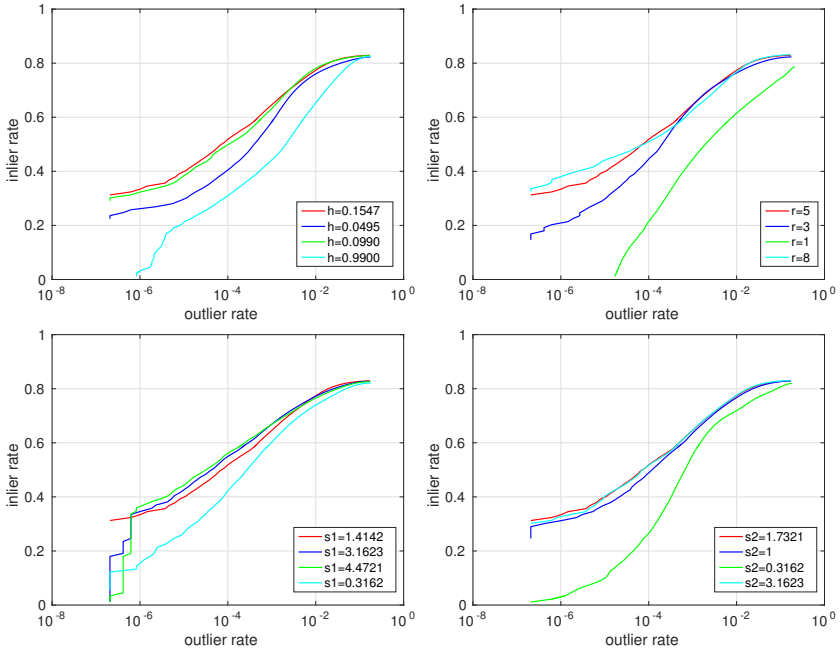

Fig. 2. Performance plots on the **library** dataset for varied parameter settings. In each plot, one parameter is varied and the others are set to the chosen values. Top left: varied h. Top right: varied r. Bottom left: varied s_1. Bottom right: varied s_2. The red curve in each plot corresponds to the chosen parameter settings. Values were chosen to maximise the inlier rate at 1% outlier rate.

our method would be to post-process the output using a similar approach as in *libfreenect2*, see [5].

5 Examples of KinFu meshes

We have implemented a data-logger that saves all output from the Kinect v2 to a file for later playback. This allows us to feed the Kinect Fusion implementation KinFu in the *Point Cloud Library* [6] with Kinect v2 output unwrapped with both *libfreenect2* and the proposed KDE method. Figure 7 in the main paper [1] gives one such example in the **lecture** scene, here extended with a second view in figure 5. In figure 6 we give an example from a KinFu run in the outdoor scene shown in figure 3, third row. Both examples show that our method produces meshes with more coverage and less noise than *libfreenect2*.

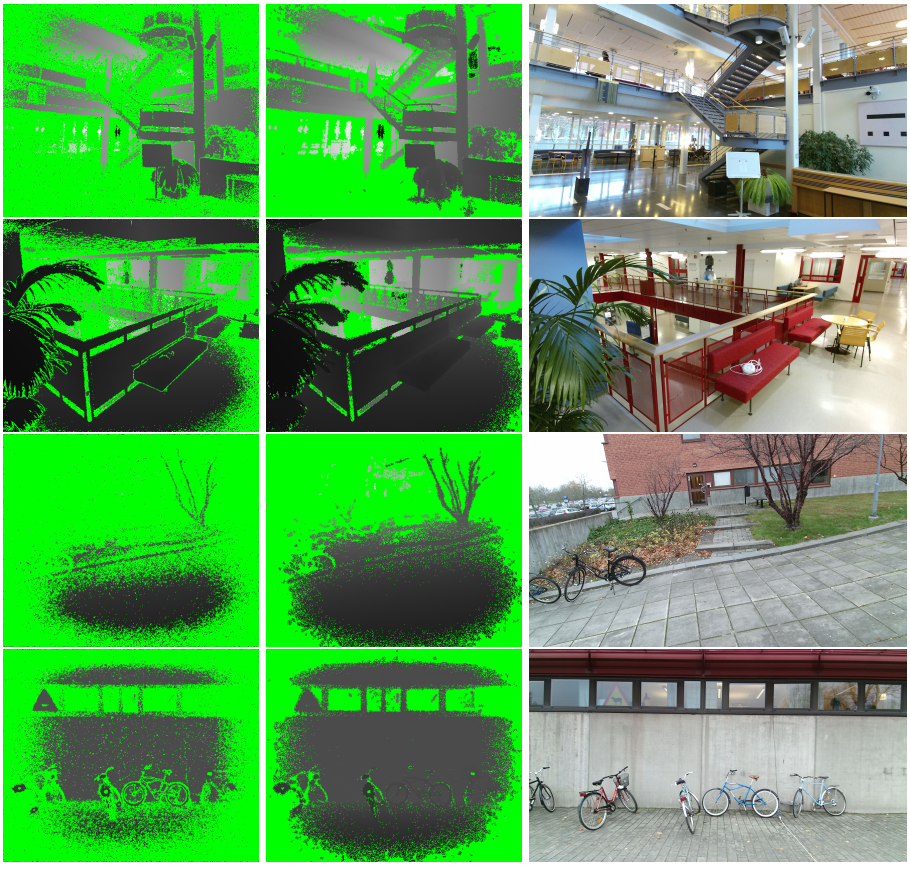

Fig. 3. Single frame output comparisons. First two rows show output on scenes with greater than 18.75m depth range, and the last two rows show outdoor scene outputs. Left column: *libfreenect2*. Center column: proposed KDE method. Right column: corresponding RGB images. Pixels suppressed by outlier rejection are shown in green. The proposed KDE method has more valid depth points than *libfreenect2* resulting in a denser and more well defined depth scene. While the suppressed areas are clean from outliers for the proposed KDE method, the *libfreenect2* images are covered in salt and pepper noise.

References

1. Anonymous: Efficient multi-frequency phase unwrapping using kernel density estimation. the paper (2016)
2. Szeliski, R.: Computer Vision: Algorithms and Applications. Springer-Verlag New York, Inc. (2010)
3. Forssén, P.E.: Low and Medium Level Vision using Channel Representations. PhD thesis, Linköping University, Sweden, SE-581 83 Linköping, Sweden (March 2004) Dissertation No. 858, ISBN 91-7373-876-X.

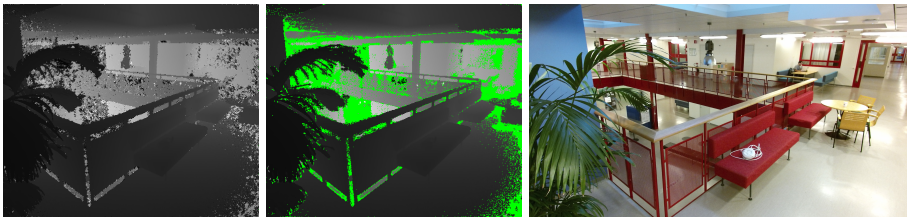

Fig. 4. Fattening effects in *libfreenect2* and the proposed KDE method. Left: *libfreenect2* without outlier rejection. Center: proposed KDE method. Right: corresponding RGB image. Notice that the same fattening artifacts on foreground objects is present in both methods.

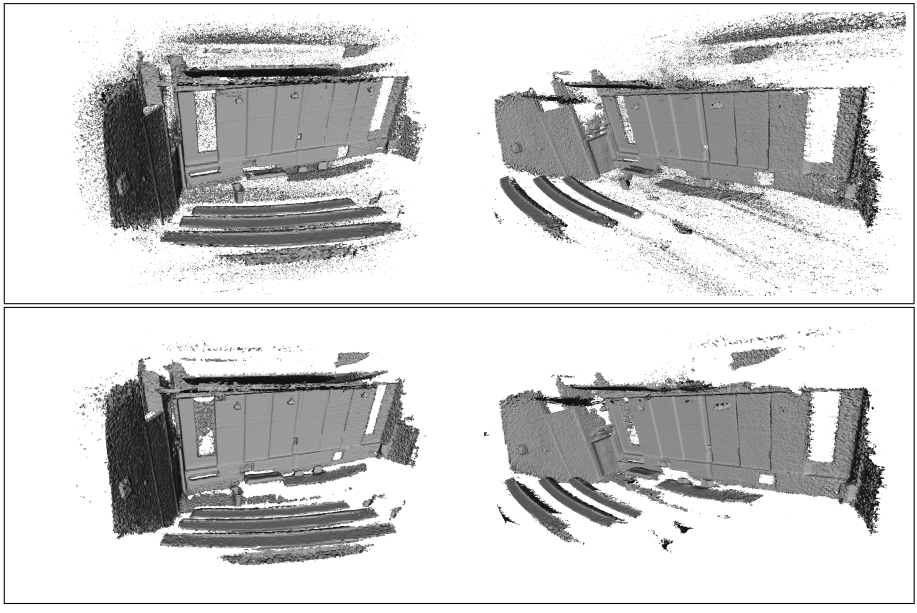

Fig. 5. KinFu run from **lecture**, see figure 7 in the main paper [1], scene with 200 frames. Top: unwrapped with *libfreenect2*. Bottom: unwrapped with proposed KDE method. Left and right columns show different views of the same mesh. Our method has more coverage of the scene and less noise, seen as scattered vertices around the model.

4. Danelljan, M., Meneghetti, G., Khan, F., Felsberg, M.: A probabilistic framework for color-based point set registration. In: IEEE International Conference on Computer Vision and Pattern Recognition (CVPR'16). (2016)
5. Open Source: libfreenect2, library for Kinect v2. https://github.com/OpenKinect/libfreenect2 (2015)
6. Rusu, R.B., Cousins, S.: 3D is here: Point Cloud Library (PCL). In: IEEE ICRA. (May 9-13 2011)

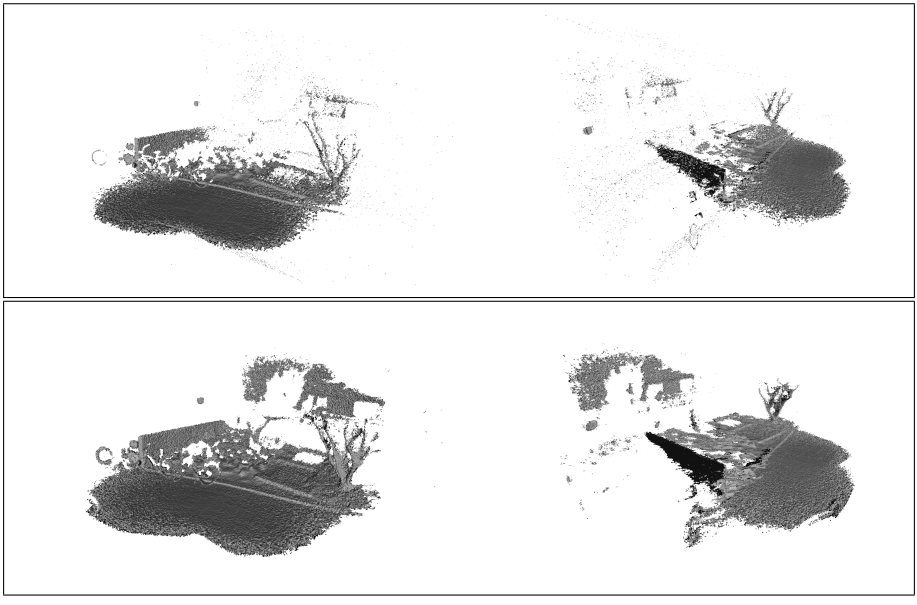

Fig. 6. Outdoors KinFu run with 100 depth frames, see figure 3 third row, for examples input and corresponding RGB frames. Top: unwrapped with *libfreenect2*, Bottom: unwrapped with proposed KDE method. Left and right columns show different views of the same mesh. Our method has more coverage and less noise. See for example the tree and the ground. Notice also the wall far in the background, on which our method produces significantly more valid measurements than *libfreenect2*.