Supplementary material for
Data Driven Depth Map Recovery via Multi-scale Sparse Representation

HyeokHyen Kwon
KAIST
hyeokhyen@kaist.ac.kr

Yu-Wing Tai
KAIST
yuwing@kaist.ac.kr

Stephen Lin
Microsoft Research
stevelin@microsoft.com

1. Clean Middleburry dataset

Table 1. Upsampling of clean middleburry dataset

	Tsukuba			Venus			Teddy			Cones		
	2×	4×	8×	2×	4×	8×	2×	4×	8×	2×	4×	8×
Park *et al*. [1]	6.61	9.75	15.1	1.27	1.8	2.99	3.73	4.89	7.15	4.00	5.64	7.73
Ferstl *et al*. [2]	7.20	10.3	17.20	2.15	2.52	4.04	2.71	3.3	5.39	3.50	4.45	7.14
Kiechle *et al*. [3]	3.48	5.95	10.9	0.8	1.17	1.76	1.28	2.94	2.76	1.7	4.17	5.11
Li *et al*. [4]	8.29	11.9	15.84	2.29	3.55	5.76	2.78	4.92	7.24	3.24	6.34	8.9
Ours(w. *nAGDP*)	**2.31**	**5.56**	**5.67**	**0.53**	**1.14**	**1.68**	**0.83**	**1.80**	**2.19**	**0.92**	**2.13**	**2.37**

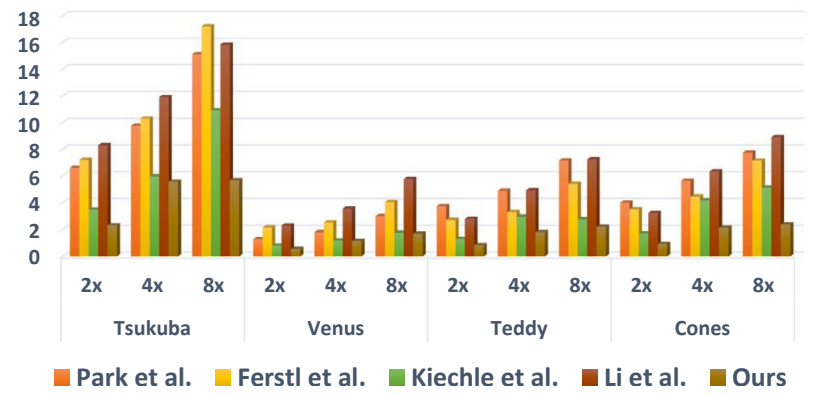


Table 1 shows comparisons of our results with those from Park *et al*. [1], Ferstl *et al*. [2] and Kiechle *et al*. [3], and Li *et al*. [4] on the Clean Middlebury dataset. We use the standard RMSE ($RMSE(\theta) = \sqrt{E((\hat{\theta} - \theta)^2)}$ where $\hat{\theta}$ is the upsampled result and θ is ground-truth) as the error metric. Our results are shown to have the lowest errors. The upsampled results and error maps of $\times 8$ upsampling for Cones, Teddy, Tsukuba, and Venus are shown from Figure 2 to Figure 5, respectively.

We note that the reported numbers from [2, 3] for Noisy Middlebury data are different from the results we obtained from their codes. Upon careful inspection, we found that their error metric was different from ours. We report their error metrics and part of their corresponding codes in Figure 1. Ferstl *et al*. [2] computed the average of absolute error[1] and Kiechle *et al*. [3] normalized the RMSE values by a scaling factor[2].

[1]Source code can be downloaded from https://rvlab.icg.tugraz.at/project_page/project_tofusion/project_tofsuperresolution.html

[2]Source code can be downloaded from http://www.gol.ei.tum.de/index.php?id=6&L=1

```
upsampling_result = uint8(upsampling_result_norm*(d_max-d_min)+d_min);

imwrite( uint8(upsampling_result), fullfile(resultpath, [num2str(trial), '.png']));

mse_ours = sum(sum(sqrt((double(disp_gt)-double(upsampling_result)).^2))) / numel(disp_gt);
```

Average of absolute errors

(a) Part of the original source codes from Ferstl *et al*. [2]

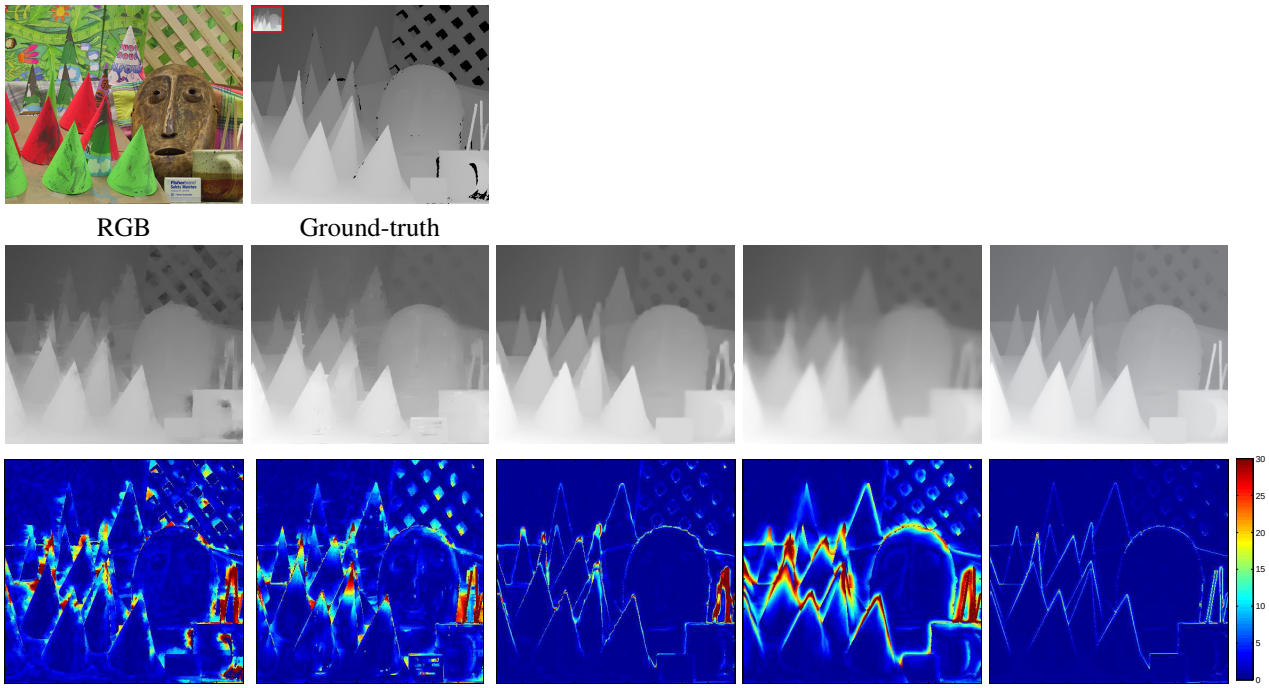
```
fact = getScaleFactor(imgs(i).name);
% results_tab = [results_tab [{imgs(i).name}; {RMSE(im2uint8(myimg),
if n==1, fprintf('%s\t', i_name); end
res(i-2) = RMSE(im2uint8(myimg), im2uint8(mygt), fact, reso);

function factor = getScaleFactor(imgname)
    [~,n,~] = fileparts(imgname);
    switch n
        case 'tsukuba'
            factor = 16;
        case 'venus'
            factor = 8;
        case 'teddy'
            factor = 4;
        case 'cones'
            factor = 4;
        otherwise
            factor = 1;
    end
end
```

```
function rmse = RMSE(img, gt, scaleFact, resolution)
    if size(img) ~= size(gt)
        disp('SCHMARRN');
        return;
    end

    mask = (gt==0);
    img(mask) = [];
    gt(mask) = [];

    switch resolution
        case 'uint8'
            error = uint8(abs(double(img(:)) - double(gt(:))));
        case 'uint16'
            error = uint16(abs(double(img(:)) - double(gt(:))));
        otherwise
            error = abs(double(img(:)) - double(gt(:)));
    end
    n = numel(error);
    rmse = sqrt(sum((error./scaleFact).^2)/n);
```

RMSE is normalized by scaling factors

(b) Part of the original source codes from Kiechle *et al*. [3]

Figure 1. Error metrics used in Ferstl *et al*. [2] and Kiechle *et al*. [3].



RGB | Ground-truth

Park *et al*. [1] | Ferstle *et al*. [2] | Kiechle *et al*. [3] | Li *et al*. [4] | Ours

Figure 2. $\times 8$ upsampling for Cones.

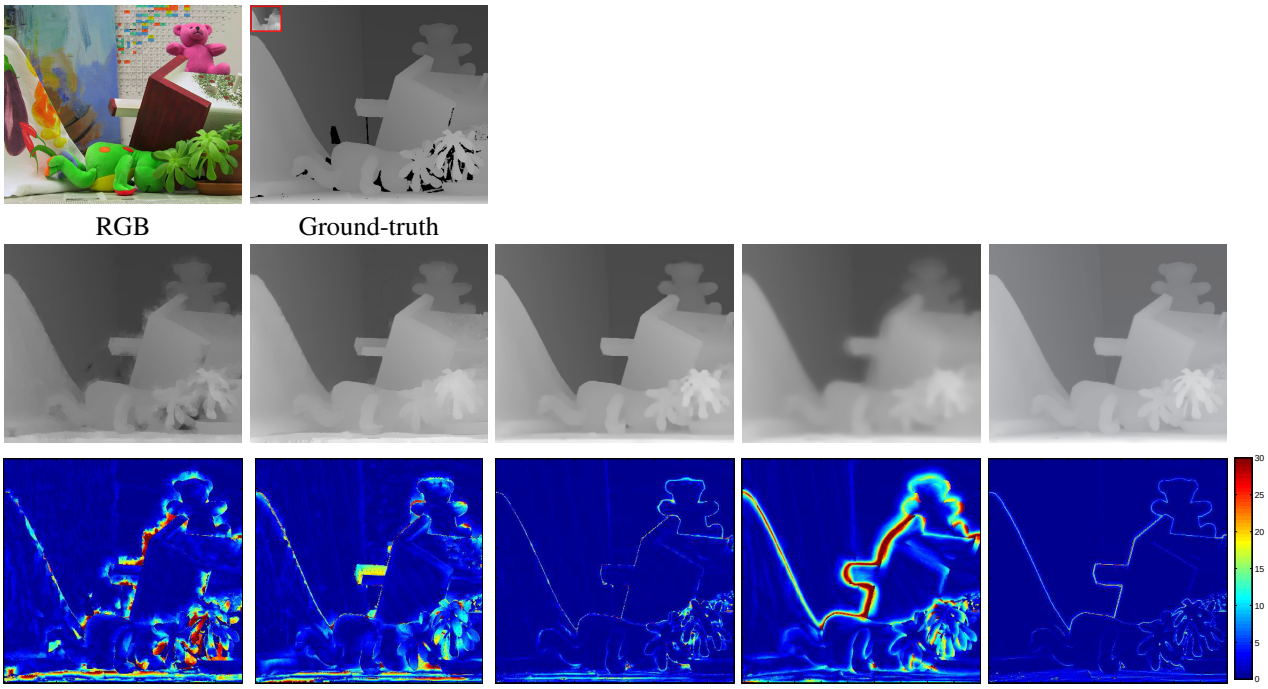


Figure 3. $\times 8$ upsampling for Teddy.

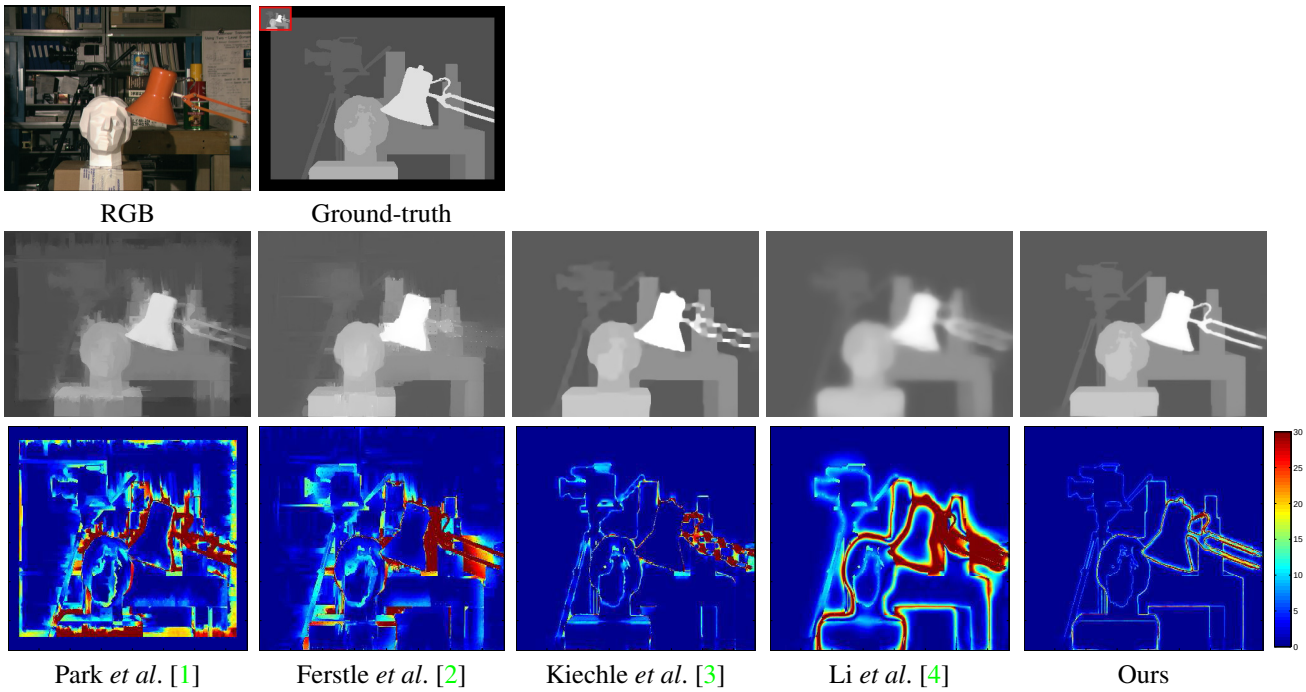


Figure 4. $\times 8$ upsampling for Tsukuba.

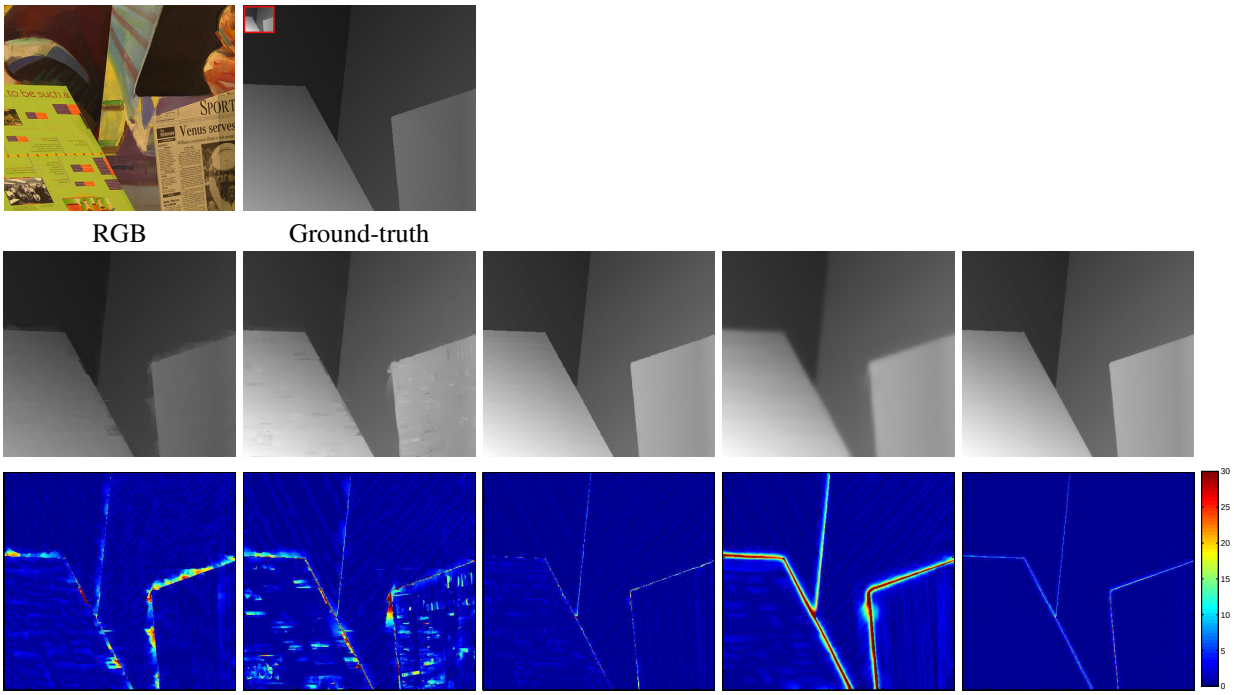
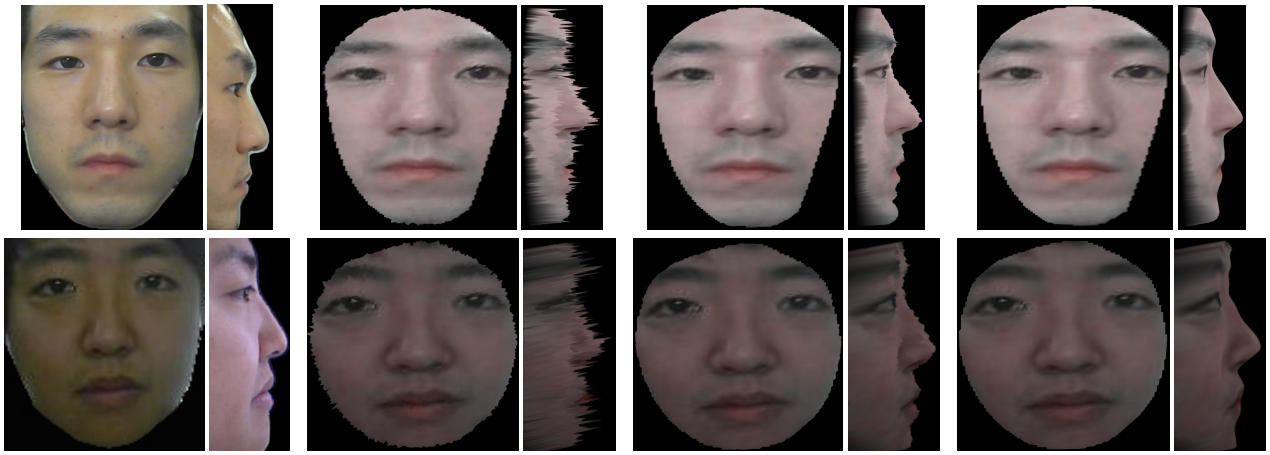


Figure 5. $\times 8$ upsampling for Venus.



(a) RGB (b) RAW (c) Intel (d) Ours

Figure 6. Intel ToF depth camera on Face data.

2. ToF face data

We tested our method on ToF face data captured by the Intel Creative Interactive Gesture Camera[3]. The high quality training data is obtained through Kinect Fusion. We measure the SNR of the ToF camera, and add noise synthetically to generate the low quality training examples. In Figure 6, we show the RAW depth maps from the ToF camera, the recovered depth maps using the embedded API provided by Intel, and our recovered depth. As can be seen in the results, the embedded method from Intel API cannot fully remove depth noise, while our method can recover more accurate depth maps for facial structure.

[3] http://click.intel.com/creative-interactive-gesture-camera-developer-kit.html

3. Kinect scene dataset

We experimented with multi-scale sparse recovery on depth completion with various public RGB-D datasets: Sturm *et al*. [5] (Figure 7 to Figure 9), Janoch *et al*. [6] (Figure 10 to Figure 13), Silberman *et al*. [7] (Figure 14), and Yang *et al*. [8] (Figure 15 and Figure 16). We also compared our results with results from the other methods: Park *et al*. [1], Ferstl *et al*. [2], Kiechle *et al*. [3], and Li *et al*. [4].

Since we used Park *et al*. [1] as initialization at the coarsest level, we compare ours with Park *et al*. [1]. The method by Kiechle *et al*. [3] is the latest RGB-D dictionary learning based approach for depth recovery. However, different from ours, they utilize only single-scale modeling, and without any mechanism to filter out irrelevant RGB features. Li *et al*. [4] utilizes joint dictionary of target depth, raw depth, and RGB, similar to ours. However, their target depth was generated from Yang *et al*. [9], which leads to bias in the solutions. Also, as in [3], they only used a single-scale dictionary without robust measurement for structural correlation between RGB and depth maps. The method of Ferstl *et al*. [2] is used as another comparison for RGB-D based depth recovery.

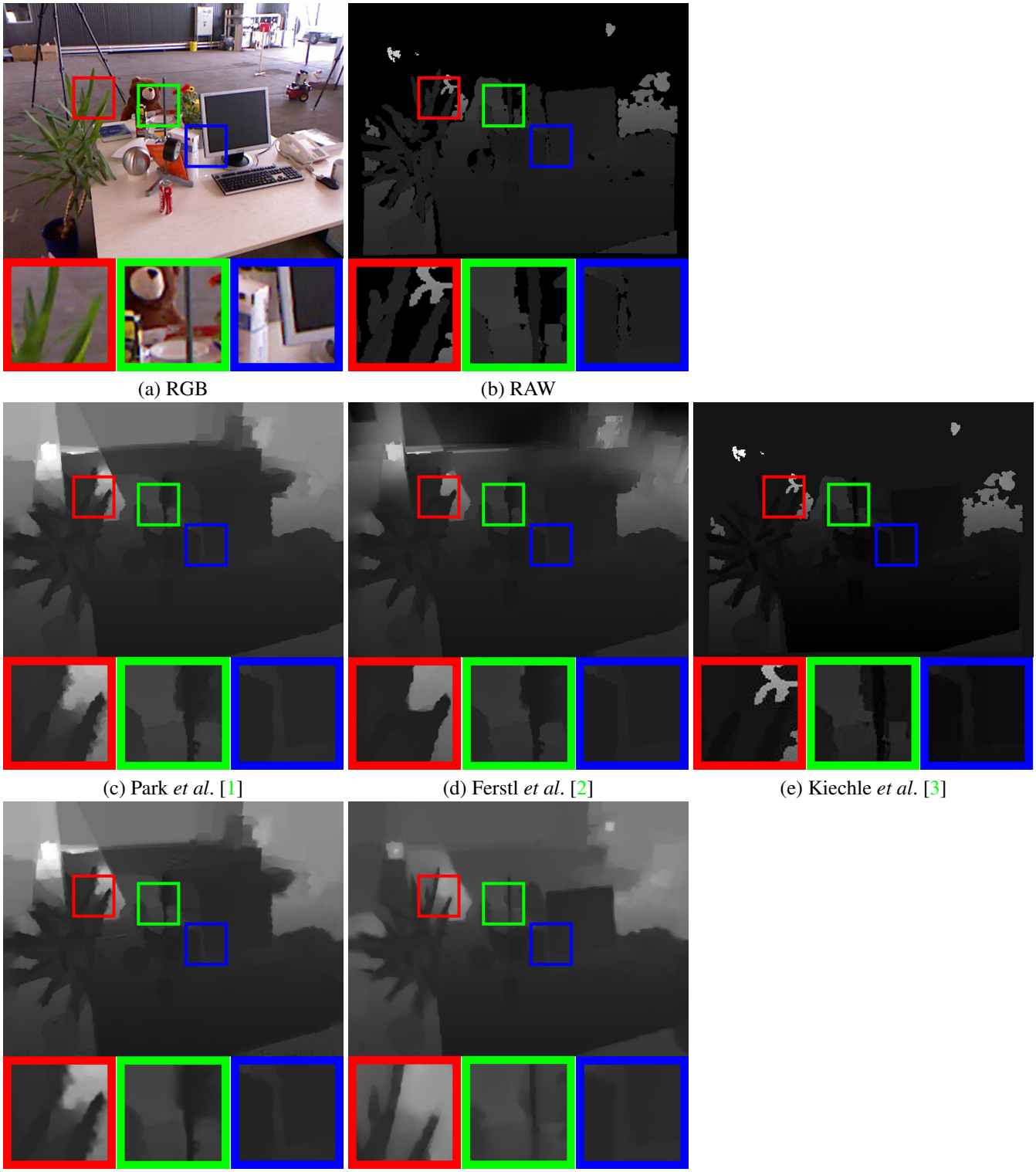
In each figure, we show RGB input, RAW depth input, Park *et al*. [1], Ferstl *et al*. [2], Kiechle *et al.*[3], Li *et al*. [4], and Ours in this order.

4. Computation time comparison

We compare the computation times for our method on Figure 8 (640$\times$ 480) with other state-of-the-art methods. For Ours, we show measured time with and without the multi-scale approach. The testing environment was Matlab 2013a, with Intel i7-4790 CPU @ 3.60 GHz and 16 GB RAM.

Method	Ferstl *et al*. [2]	Kiechle *et al*. [3]	Park *et al*. [1]	Li *et al*. [4]	Ours (Single)	Ours (Multi)
Time(sec.)	140	450	4	700	200	300

Figure 7. Comparison results with Kinect RGB-D data

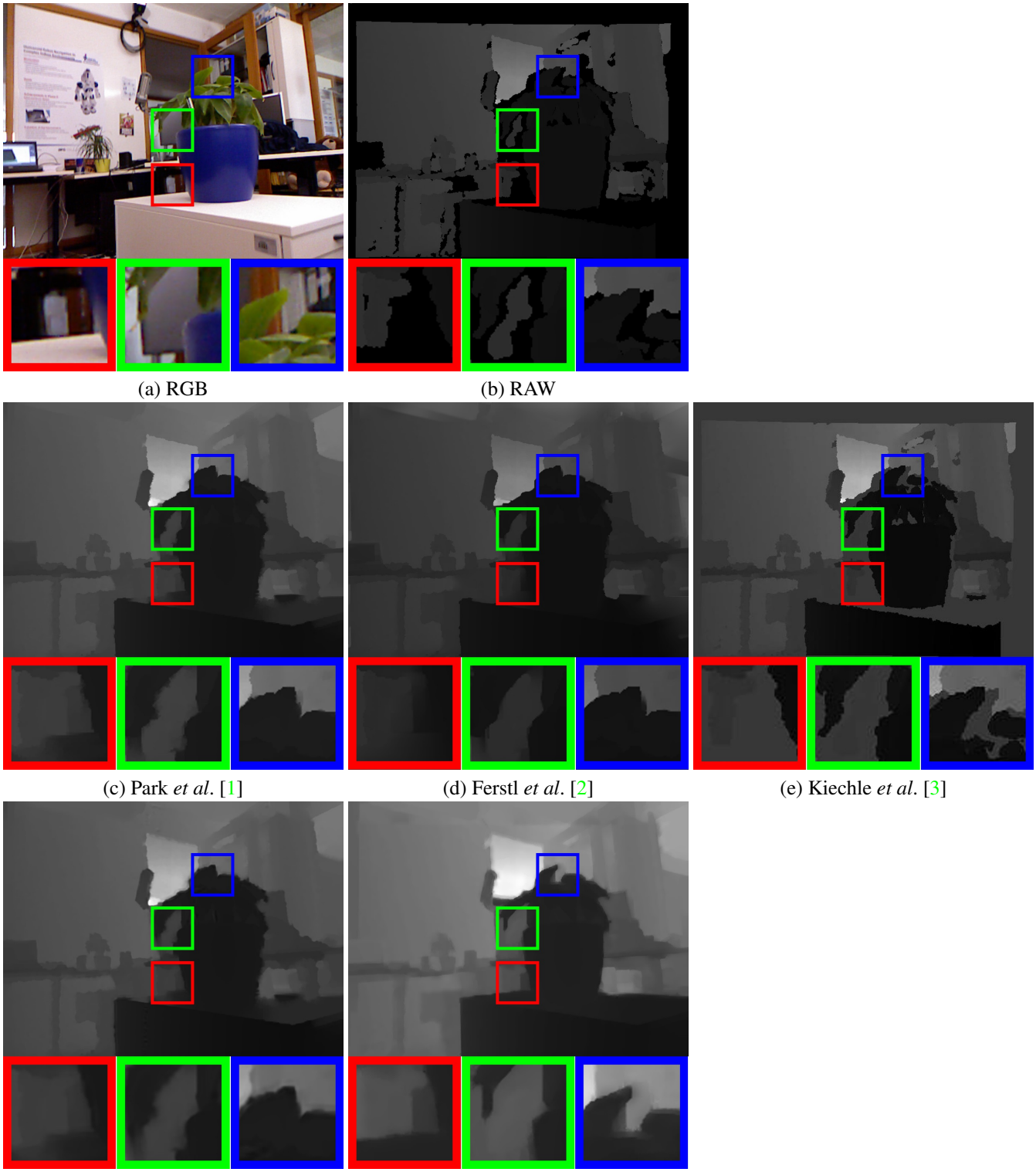


(a) RGB (b) RAW

(c) Park *et al*. [1] (d) Ferstl *et al*. [2] (e) Kiechle *et al*. [3]

(e) Li *et al*. [4] (f) Ours

Figure 8. Comparison results with Kinect RGB-D data

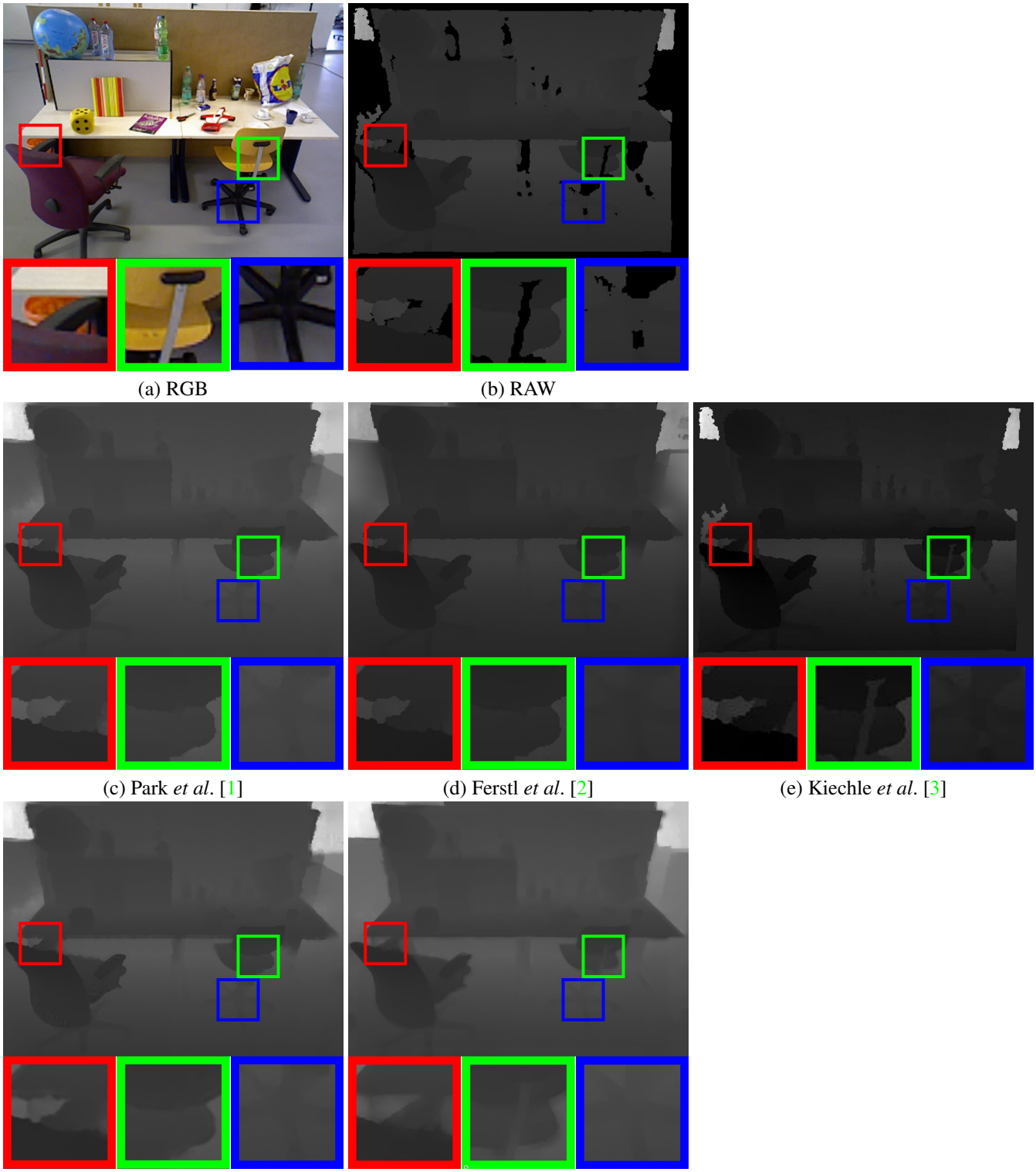


(a) RGB (b) RAW

(c) Park *et al*. [1] (d) Ferstl *et al*. [2] (e) Kiechle *et al*. [3]

(e) Li *et al*. [4] (f) Ours

Figure 9. Comparison results with Kinect RGB-D data

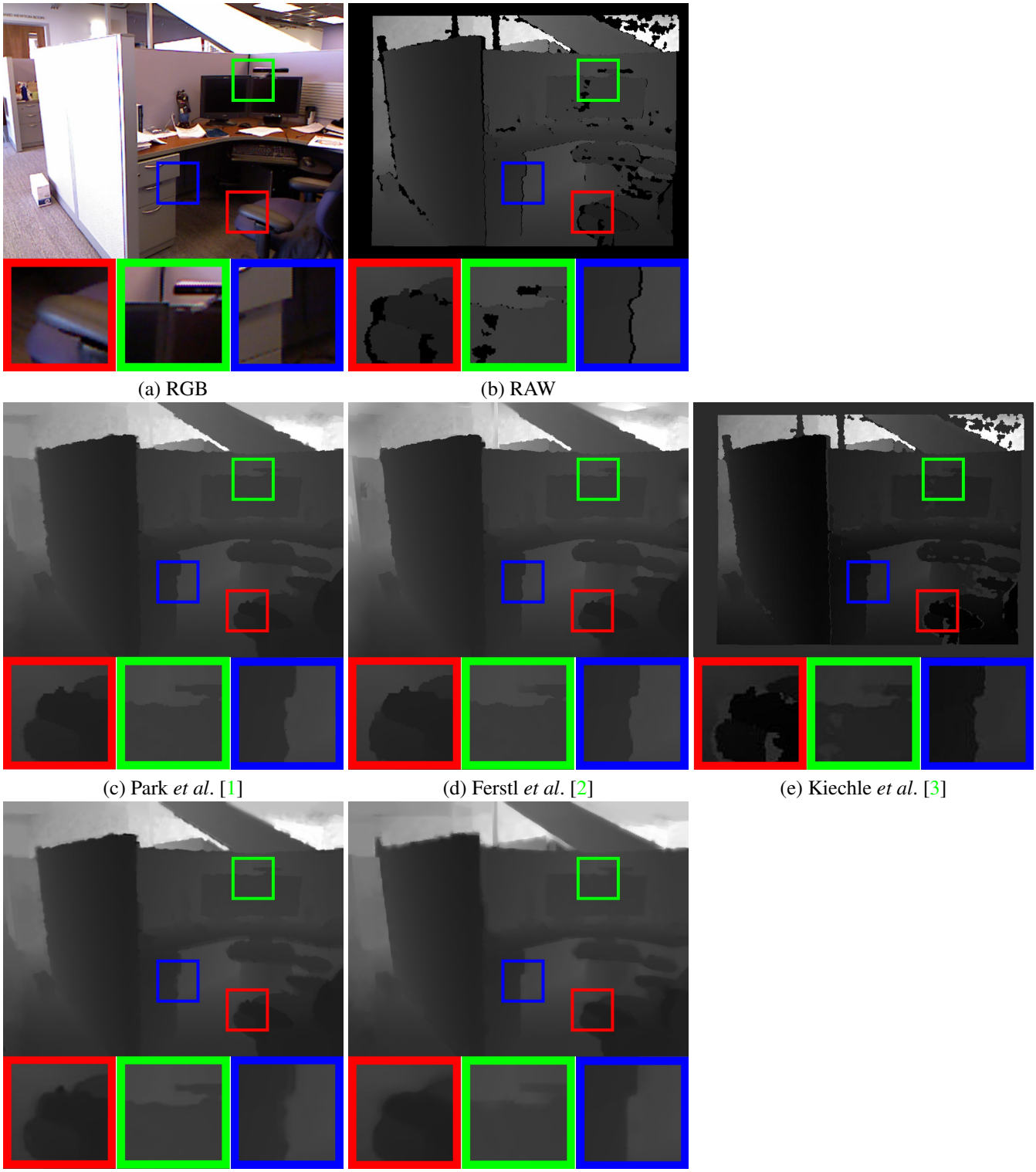


(a) RGB (b) RAW

(c) Park *et al*. [1] (d) Ferstl *et al*. [2] (e) Kiechle *et al*. [3]

(e) Li *et al*. [4] (f) Ours

Figure 10. Compare results with Kinect RGB-D data

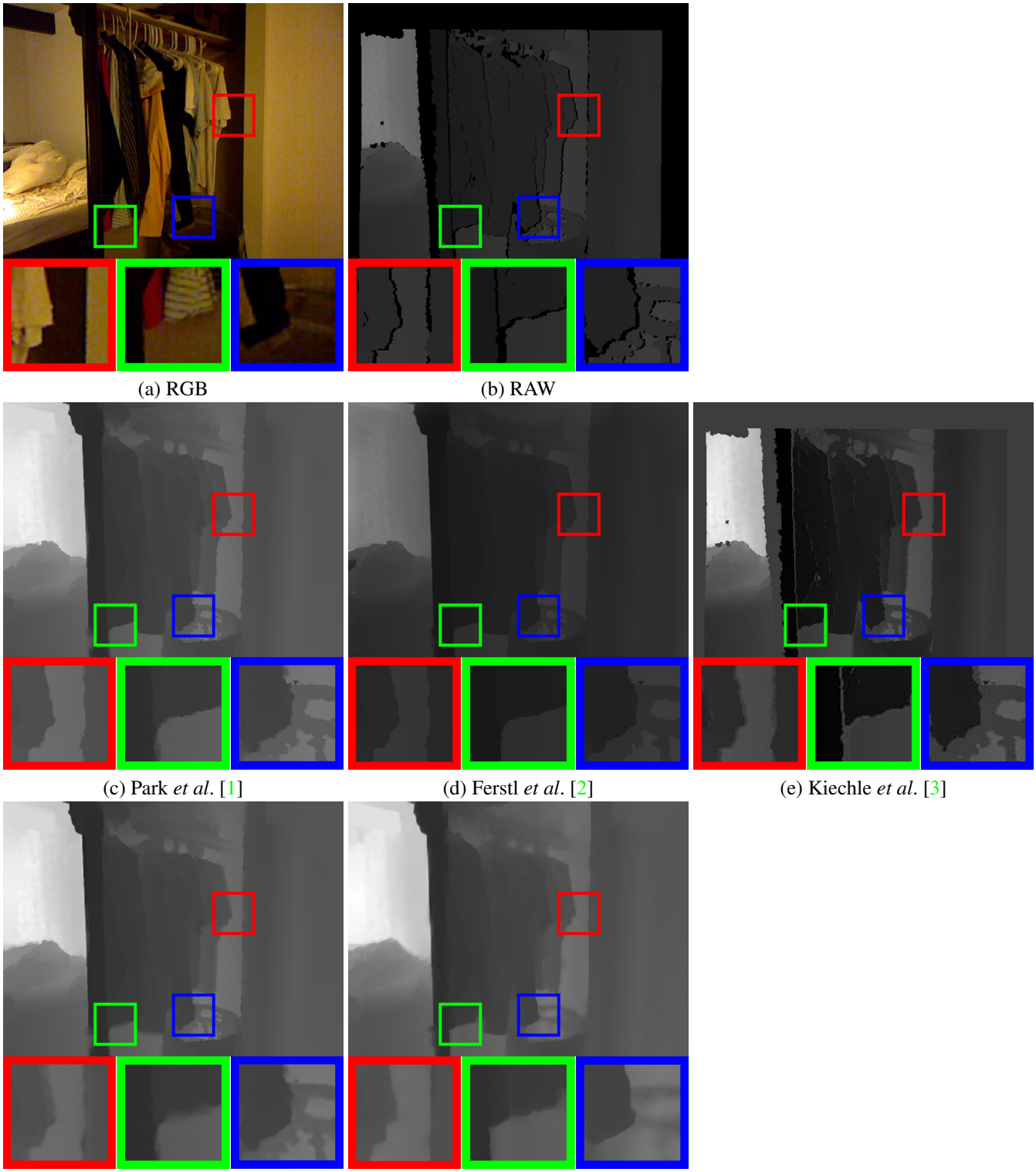


(a) RGB (b) RAW

(c) Park *et al*. [1] (d) Ferstl *et al*. [2] (e) Kiechle *et al*. [3]

(e) Li *et al*. [4] (f) Ours

Figure 11. Comparison results with Kinect RGB-D data

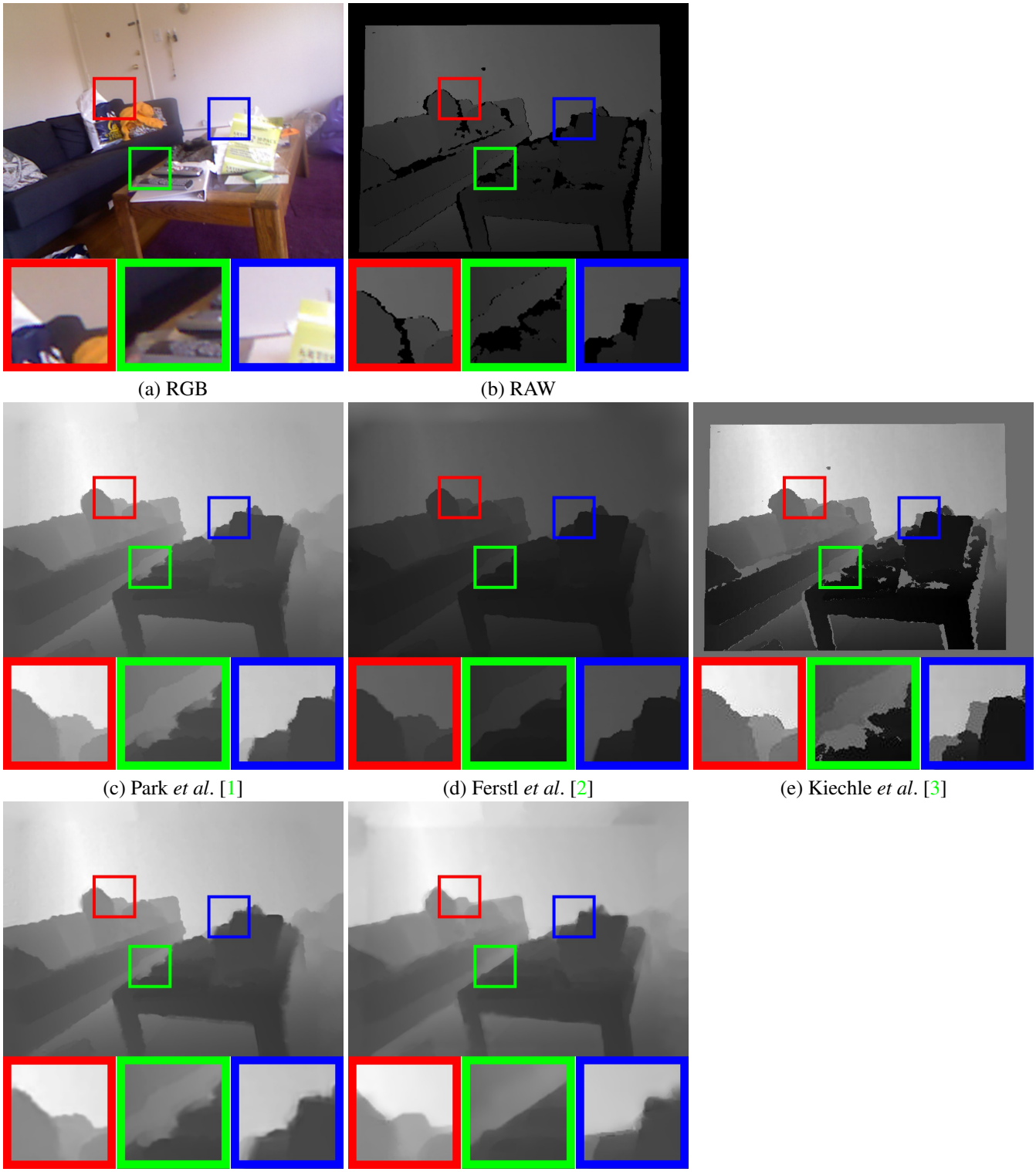


(a) RGB (b) RAW

(c) Park *et al*. [1] (d) Ferstl *et al*. [2] (e) Kiechle *et al*. [3]

(e) Li *et al*. [4] (f) Ours

Figure 12. Comparison results with Kinect RGB-D data

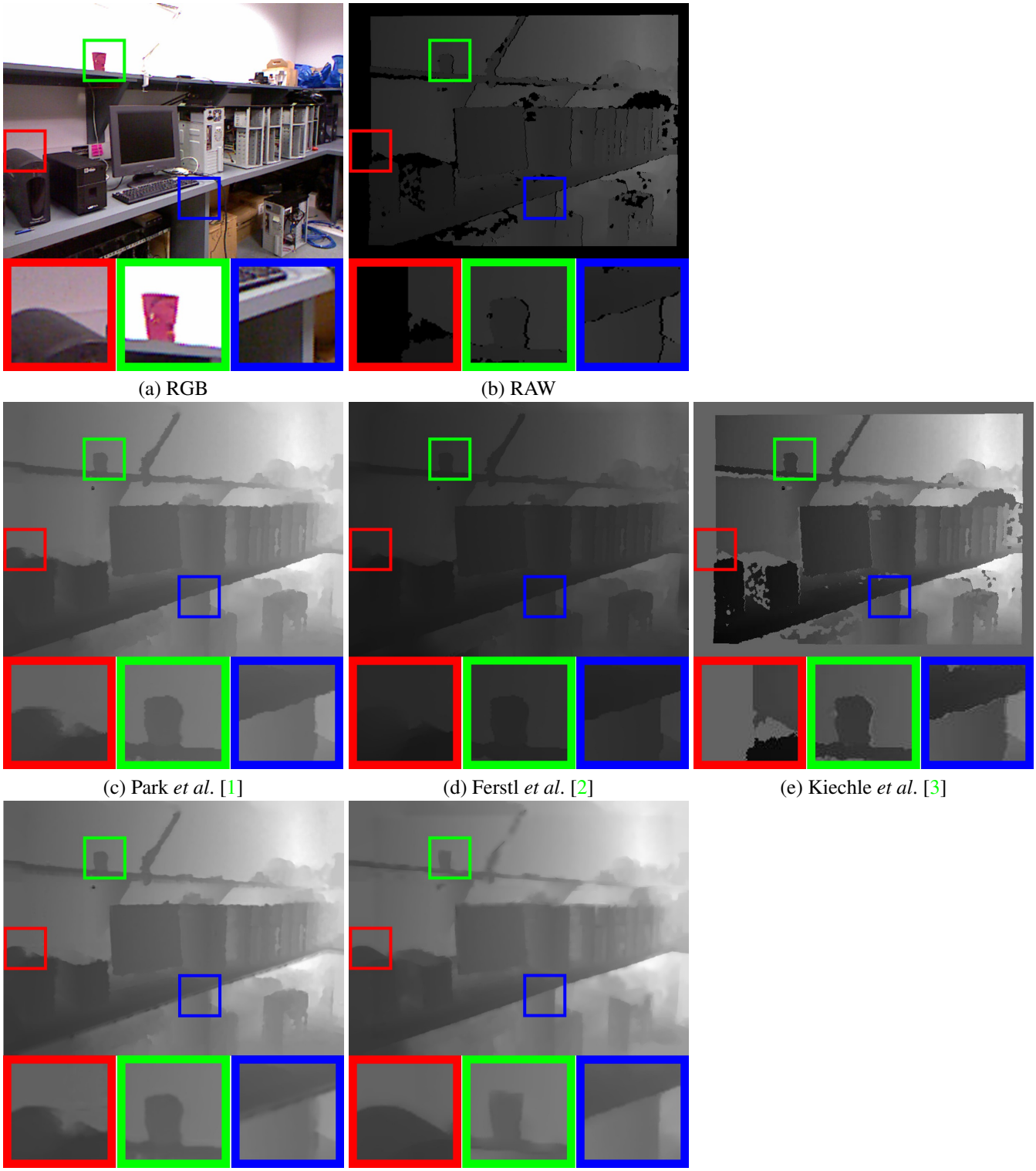


(a) RGB (b) RAW

(c) Park *et al*. [1] (d) Ferstl *et al*. [2] (e) Kiechle *et al*. [3]

(e) Li *et al*. [4] (f) Ours

Figure 13. Comparison results with Kinect RGB-D data

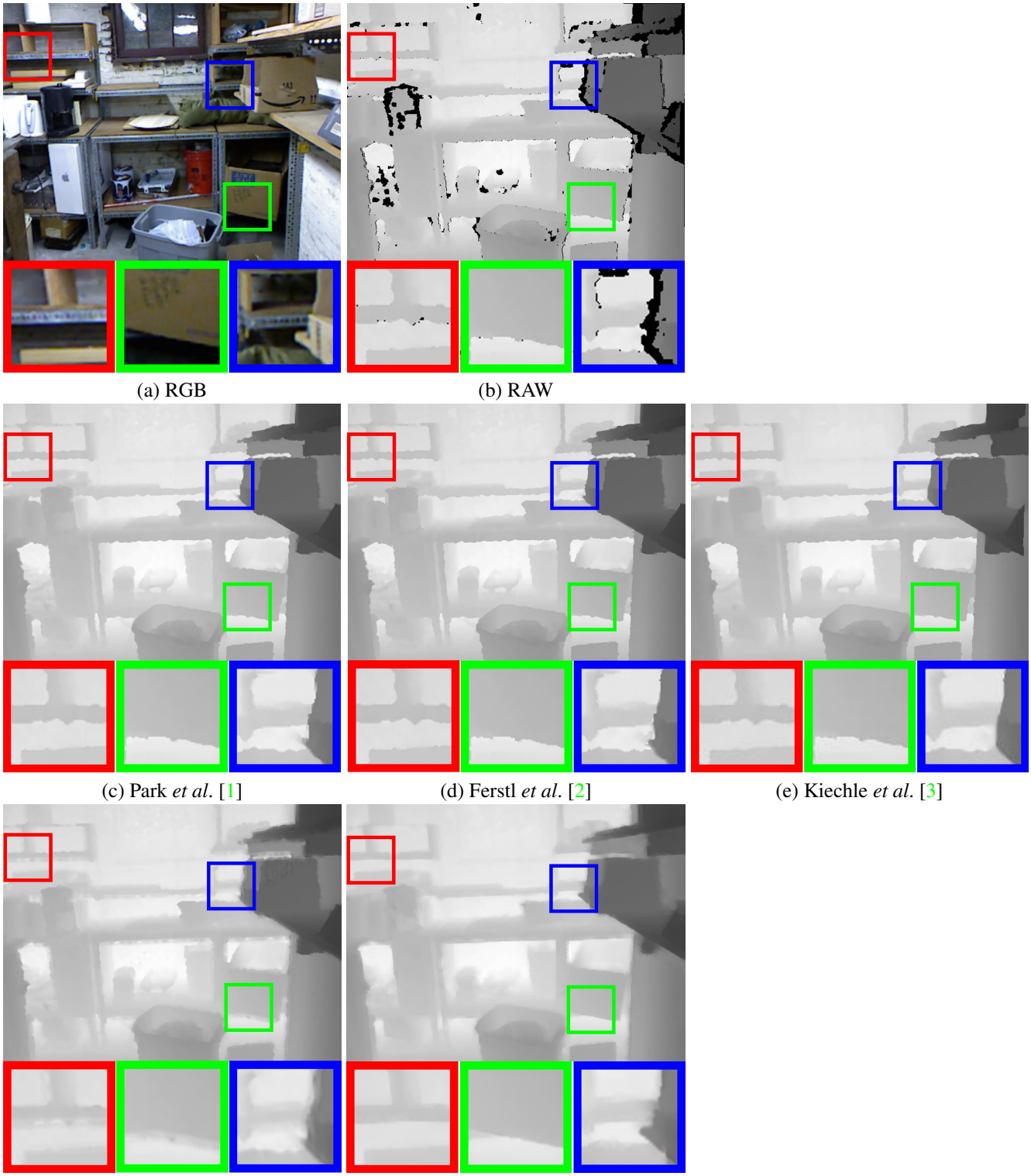


(a) RGB (b) RAW

(c) Park *et al*. [1] (d) Ferstl *et al*. [2] (e) Kiechle *et al*. [3]

(e) Li *et al*. [4] (f) Ours

Figure 14. Comparison results with Kinect RGB-D data

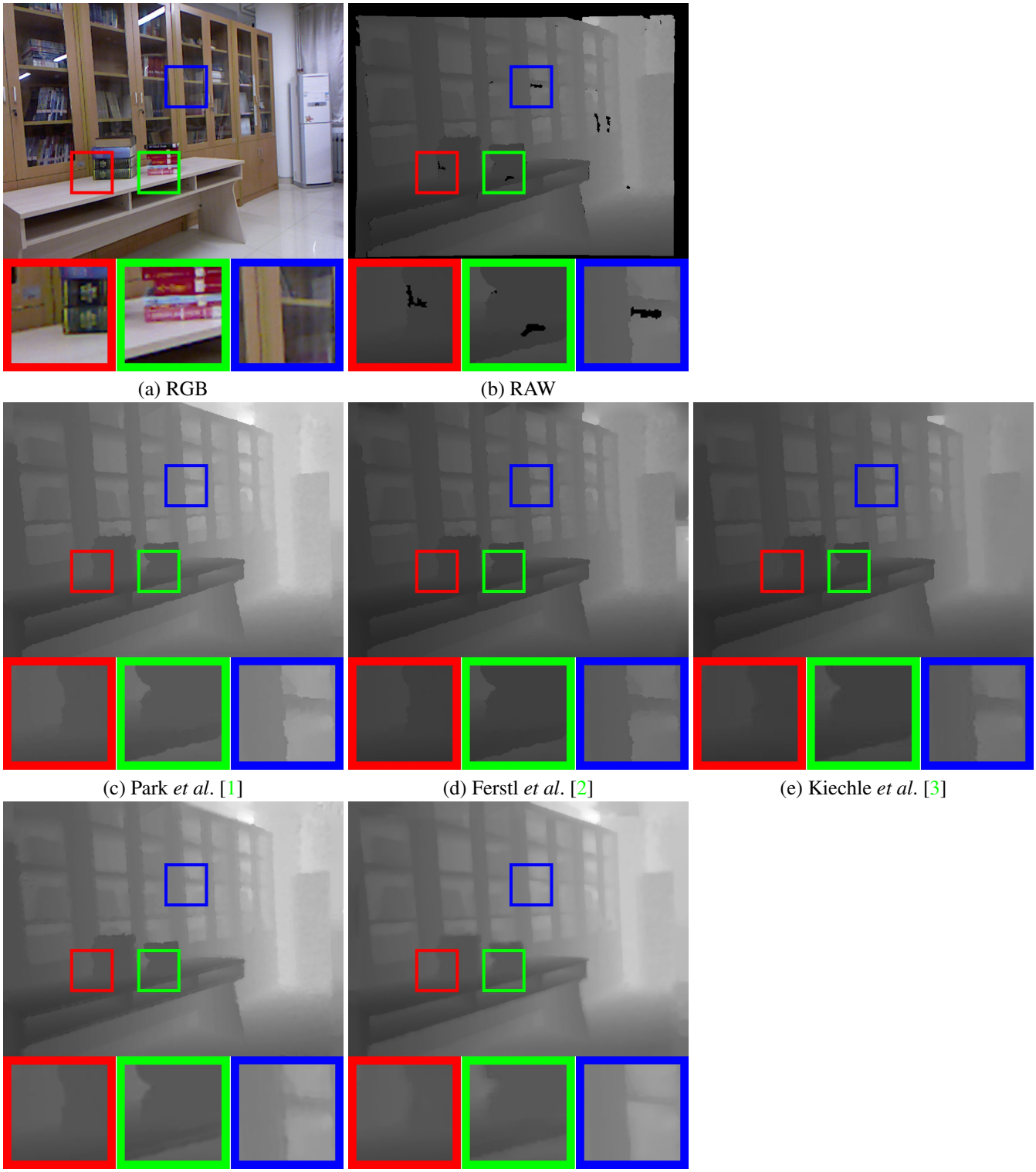


(a) RGB (b) RAW

(c) Park *et al*. [1] (d) Ferstl *et al*. [2] (e) Kiechle *et al*. [3]

(e) Li *et al*. [4] (f) Ours

Figure 15. Comparison results with Kinect RGB-D data

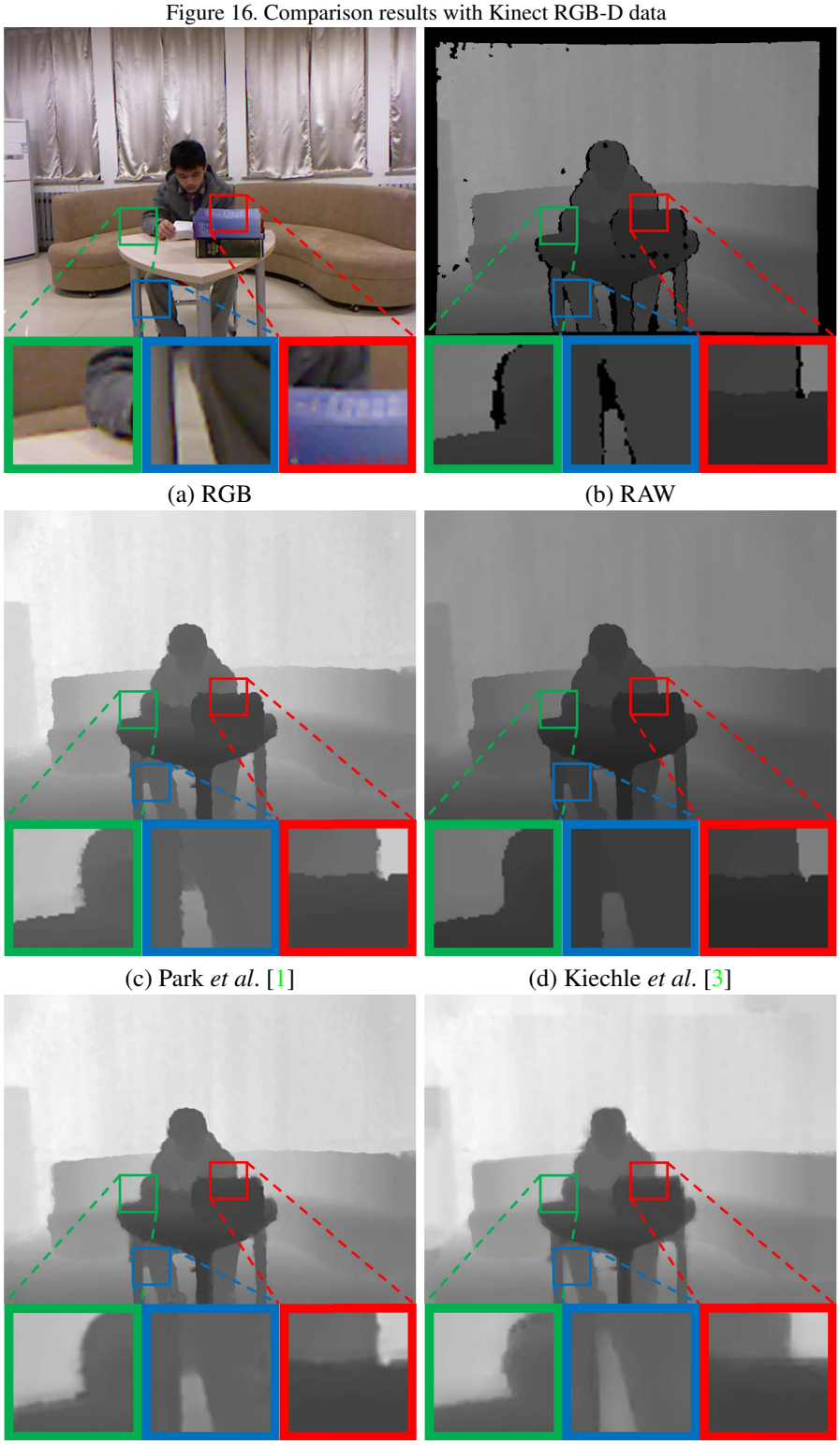


(a) RGB (b) RAW

(c) Park *et al*. [1] (d) Ferstl *et al*. [2] (e) Kiechle *et al*. [3]

(e) Li *et al*. [4] (f) Ours

Figure 16. Comparison results with Kinect RGB-D data



(a) RGB (b) RAW

(c) Park *et al*. [1] (d) Kiechle *et al*. [3]

(e) Li *et al*. [4] (f) Ours

5. Videos

We provide three videos for additional qualitative comparisons. One is a Kinect face depth recovery comparison with Kiechle *et al*. [3]. The other two (one for face depth map and one for scene depth map) are recovery results using the Intel ToF depth map (Intel Creative Gesture Camera). We compared our results with the embedded API provided by Intel. Although our results are not perfectly clean, they indeed surpass previous state-of-the-art methods. For the real-world ToF examples (ToF 3D scenes), there exists substantial non-uniform noise that is not fully removed after downsampling. This is a reason why the refined results are not entirely planar. Including an additional level in the multiscale processing might have helped to lessen this noise. Also, adding a prior that favors planar surfaces would help to reduce such distortions.

References

[1] J. Park, H. Kim, Y.-W. Tai, M.S. Brown, and I. Kweon. High quality depth map upsampling for 3d-tof cameras. *ICCV*, pages 1623–1630, 2011. 1, 2, 3, 4, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16

[2] D. Ferstl, C. Reinbacher, R. Ranftl, M. Rther, and H. Bischof. Image guided depth upsampling using anisotropic total generalized variation. *ICCV*, 2013. 1, 2, 3, 4, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15

[3] M. Kiechle, S. Hawe, and M. Kleinsteuber. A joint intensity and depth co-sparse analysis model for depth map super-resolution. *ICCV*, 2013. 1, 2, 3, 4, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17

[4] Y. Li, T. Xue, L. Sun, and J. Liu. Joint example-based depth map super-resolution. *ICME*, pages 152–157, 2012. 1, 2, 3, 4, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16

[5] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of rgb-d slam systems. *In Intelligent Robots and Systems, IEEE/RSJ International Conference on*, pages 573–580, 2012. 6

[6] A. Janoch, S. Karayev, Y. Jia, J. T. Barron, M. Fritz, K. Saenko, and T. Darrell. A category-level 3-d object dataset: Putting the kinect to work. In *ICCV*, 2011. 6

[7] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012. 6

[8] J. Yang, X. Ye, K. Li, and C. Hou. Depth recovery using an adaptive color-guided auto-regressive model. *ECCV*, pages 158–171, 2012. 6

[9] Q. Yang, R. Yang, J. Davis, and D. Nistér. Spatial-depth super resolution for range images. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007. 6