Intrinsic Depth:
Improving Depth Transfer with Intrinsic Images

Naejin Kong and Michael J. Black
Max Planck Institute for Intelligent Systems
Spemannstrasse 41, 72076 Tübingen, Germany
{naejin.kong,black}@tuebingen.mpg.de

1. Proxy Depth: Filling with Temporal Propagation

We compute proxy depth maps in **Section 3.2 Regularization: Prior** by propagating information over time so as to fill in those segments where no sparse points are projected. We use video segment volumes from [3] instead of image segments, thus it is possible to fill in segments without projected points. First, for each segment of a segment volume that has projected points, we take the median depth from its projected points and fill in that segment with the value. Second, for those segments of the segment volume without any projected points, we temporally interpolate the values already assigned in the other segments, only if more than a half of the segments in the segment volume already have the depth values assigned, otherwise we do not further fill in those segments.

2. Anaglyph

This section is slightly modified from the proceedings version for clarity. We show anaglyph images converted from estimated depth maps in the supplementary video. An anaglyph image encodes a pair of stereo images using red and cyan filters.

We compute the anaglyph images by modifying the technique used in [5]. The heuristic for converting a depth video to disparity maps is $normalize(\frac{1}{1+normalize(\mathcal{D})}) \times L$, where $normalize(\cdot)$ scales a video $\cdot$ so that its values stay within $[0, 1]$, $\mathcal{D}$ is an estimated depth video, $L = \lfloor \sqrt{w \cdot h}/c \rfloor$ where (w, h) indicates the width and height of the video, and $c = 10$ except an outdoor example with a statue on the lawn for which $c = 40$ is used. We do not temporally regularize the converted disparity maps. In order to compute $normalize(\mathcal{D})$, we use the minimum and maximum values in the depth video except depth from [8], in which a few pixels have abnormally higher values than the other pixels thus default normalization produces overall dark depth images with very little variation that result in flat anaglyph. Instead we normalize the depth from [8] so that the values greater than or equal to a high percentile (ex. 99^{th}) over the video is scaled to 1.

3. Original Regularization Equations

Here we repeat the original formulation in [5] for depth regularization using our notation.

3.1. Data term

The data term is defined as

$$E_{\text{data}}(D_t, C_t^{(1\ldots K)}) = \sum_{\mathbf{x}} \sum_{k=1}^{K} w_t(\mathbf{x}, k) \Big[\rho(D_t(\mathbf{x}) - \Psi_k(C_t^{(k)}(\mathbf{x}))) + \lambda \Big\{ \rho(\nabla_x D_t(\mathbf{x}) - \Psi_k(\nabla_x C_t^{(k)}(\mathbf{x}))) + \rho(\nabla_y D_t(\mathbf{x}) - \Psi_k(\nabla_y C_t^{(k)}(\mathbf{x}))) \Big\} \Big], \quad (1)$$

where $C_t^{(k)}$ is the k^{th} candidate frame for the query frame at t, $\rho(x) = \sqrt{x^2 + \epsilon^2}$ (a differentiable approximation to the L1 penalty; $\epsilon = 0.01$), Ψ_k is a warping operator that uses SIFT flow computed between the query video frame and the k^{th} candidate frame, and $\lambda = 10$. Note that Ψ_k warps gradients of the candidate depth map as well. $w_t(\mathbf{x}, k)$ is a weight function

that gives higher importance to pixel $\mathbf{x}$ of the k^{th} candidate frame if its value is more reliable. It is determined by comparing SIFT descriptors and optical flow of the query frame with those of the candidate frame. Then,

$$w_t(\mathbf{x},k) = \frac{1 - sig\left(\left|\, \mathcal{S}_t(\mathbf{x}) - \Psi_k(\mathcal{S}_t^{(k)}(\mathbf{x})) \,\right|, 10, 0.5\right)}{2} + \alpha_d \cdot \frac{1 - sig\left(\left|\, \|\mathbf{u}_t(\mathbf{x})\| - \Psi_k(\|\mathbf{u}_t^{(k)}(\mathbf{x})\|) \,\right|, 10, 0.5\right)}{2}, \quad (2)$$

where $\mathcal{S}_t(\mathbf{x})$ defines the SIFT feature vector of the query frame at t, and $\mathcal{S}_t^{(k)}(\mathbf{x})$ defines the SIFT feature vector of the k^{th} candidate frame at t. $\|\mathbf{u}_t(\mathbf{x})\|$ and $\|\mathbf{u}_t^{(k)}(\mathbf{x})\|$ are the flow magnitude of the query frame and the k^{th} candidate frame at t, receptively, and $\alpha_d = 1$. $sig(\cdot)$ is a sigmoid function in Eq. (5) of the main paper.

3.2. Spatial smoothness

The spatial term is defined as $E_{\text{spat}}(D_t) =$

$$\sum_{\mathbf{x}} s_t^x(\mathbf{x})\rho(\nabla_x D_t(\mathbf{x})) + s_t^y(\mathbf{x})\rho(\nabla_y D_t(\mathbf{x})), \quad (3)$$

where $\nabla_x D_t$ and $\nabla_y D_t$ are horizontal and vertical depth gradients, respectively. The weighting functions s_t^x and s_t^y are determined by RGB image boundaries, that is,

$$\begin{aligned} s_t^x(\mathbf{x}) &= 1 - sig(|\nabla_x I_t(\mathbf{x})|^2, 5000, 0.001) \\ s_t^y(\mathbf{x}) &= 1 - sig(|\nabla_y I_t(\mathbf{x})|^2, 5000, 0.001), \end{aligned} \quad (4)$$

where I_t is the query video frame, $sig(\cdot)$ is a sigmoid function above. $\rho(x) = \sqrt{x^2 + \epsilon^2}$ where $\epsilon = 0.01$. A median filter is applied to I_t before computing gradients.

3.3. Prior

The prior term minimizes the difference between the estimated depth map and the prior image, P, which is an average depth map over the database,

$$E_{\text{prior}} = \sum_{\mathbf{x}} \rho(D_t(\mathbf{x}) - P(\mathbf{x})), \quad (5)$$

where $\rho(x)$ is the same as above.

3.4. Temporal coherence term

This term encourages temporal coherence of the estimated depth maps by using optical flow of the query video. It is defined as

$$E_{\text{temp}}(D_t, D_{t+1}, \mathbf{u}_t) = \sum_{\mathbf{x}} s_t^{\text{temp}}(\mathbf{x}) \cdot \rho(D_{t+1}(\mathbf{x} + \mathbf{u}_t(\mathbf{x})) - D_t(\mathbf{x})), \quad (6)$$

where $\mathbf{u}_t$ is optical flow from t to $t+1$ and $s_t^{\text{temp}}(\mathbf{x})$ is a weight function, measured by the flow confidence

$$s_t^{\text{temp}}(\mathbf{x}) = 1 - sig\left(\left|\, G(I_{t+1}(\mathbf{x} + \mathbf{u}_t(\mathbf{x})) - I_t(\mathbf{x}), \sigma) \,\right|, 1000, 0.005\right) \quad (7)$$

where I_t and I_{t+1} are the query frames at t and $t+1$, respectively, and G is a Gaussian filter ($\sigma = 1$). Again, $\rho(x)$ and $sig(\cdot)$ are the same as above.

4. Optimization

In this section we show how to optimize our modified objective equation. In the original method, Eq. (4) in the main paper is solved by using iteratively reweighted least squares (IRLS). We take the similar approach by deriving an IRLS solution for each term. We alternately solve for depth D and the scale variable a.

We derive IRLS solutions for Eqs. (7) and (8) in the main paper. This can be applied similarly to other terms. Our prior term Eq. (7) in the main paper over all frames is

$$\sum_t \sum_{\mathbf{x}} s_t^{\text{pxy}}(\mathbf{x}) \cdot \rho(a \cdot D_t(\mathbf{x}) - \mathcal{P}_t(\mathbf{x})), \quad (8)$$

where $\mathcal{P}_t$ is a proxy depth map, and s_t^{pxy} is a binary weight mask, which is set to 1 if $\mathcal{P}_t$ is valid at that pixel, 0 otherwise. Our temporal term in Eq (8) of the main paper over all frames is

$$\sum_{t=1}^{T-1} \sum_{\mathbf{x}} s_t^{\text{temp}}(\mathbf{x}) \cdot \rho(S_{t+1}(\mathbf{x}+\mathbf{u}_t(\mathbf{x}), R_{t+1}, \theta) \cdot D_{t+1}(\mathbf{x}+\mathbf{u}_t(\mathbf{x})) \cdot a - S_t(\mathbf{x}, R_t, \theta) \cdot D_t(\mathbf{x}) \cdot a + o_t(c_t, c_{t+1})), \tag{9}$$

where T is the number of frames, $\mathbf{u}_t$ is optical flow from t to $t+1$, S_t is a scale function in terms of a camera rotation R_t at t and intrinsic parameters θ. An offset o_t is in terms of the 3D camera positions c_{t+1} at $t+1$ and c_t at t, a is an unknown scale variable, and the weight function $s_t^{\text{temp}}(\mathbf{x})$ is in Eq. (7).

Eq. (7) of the main paper. We define $N = M \cdot T$ where M as the number of pixels at each image and T is the number of frames. We then define vectorized variables over the video volume as follows: $\mathbf{D}$ is a N-vector of the depth maps $D_{1\cdots T}$, $\mathbf{P}$ is a N-vector of the proxy depth maps $\mathcal{P}_{1\cdots T}$, and $\mathbf{s}^{\text{pxy}}$ is a N-vector of the weight maps $s_{1\cdots T}^{\text{pxy}}$. We differentiate Eq. (7) in the main paper with respect to depth and set it equal to zero:

$$\sum_{p=1}^{N} \mathbf{s}_p^{\text{pxy}} \cdot \frac{d}{d\mathbf{D}_p} \rho(a\mathbf{D}_p - \mathbf{P}_p) \cdot a \cdot (a\mathbf{D}_p - \mathbf{P}_p) = 0, \tag{10}$$

where $\mathbf{v}_p$ indicates the element of vector $\mathbf{v}$ at pixel index p. Its IRLS solution is

$$\mathbf{D}^{(i+1)} = (a^2\mathbf{V}^{(i)})^+(a\mathbf{V}^{(i)}\mathbf{P}), \;\; \mathbf{V}^{(i)} = diag\left(\mathbf{s}^{\text{pxy}}\right) \cdot diag\left(\frac{d}{d\mathbf{D}} \rho(a\mathbf{D}^{(i)} - \mathbf{P})\right), \tag{11}$$

where $diag(\cdot)$ an operator that produces a diagonal matrix from a given vector, $(\cdot)^+$ is the pseudoinverse, and $\mathbf{D}^{(i)}$ is the inferred depth at iteration i.

Eq. (8) of the main paper. Again, N is the number of pixels in the full video volume. We then define that $\mathbf{D}$ is a N-vector of the depth maps $D_{1\cdots T}$, $\mathbf{O}$ is a N-vector for which column-wise combines M-vector of the camera offset o_t at t over $T-1$ frames (zero vector at T), and $\mathbf{s}^{\text{temp}}$ be a N-vector of the flow confidence maps $s_{1\cdots T-1}^{\text{temp}}$ (zero vector at T). We define $\mathbf{G}_{\text{ts}}$ as an $N \times N$ linear operator, that returns the difference between two corresponding pixels at $\mathbf{x}$ and $\mathbf{x}+\mathbf{u}_t$ from t and $t+1$, respectively, where the values of the two pixel are scaled by the scale function values $S_t(\mathbf{x})$ and $S_{t+1}(\mathbf{x}+\mathbf{u}_t)$. By applying similar derivation above, we have the IRLS solution

$$\mathbf{D}^{(i+1)} = (a^2\mathbf{G}_{\text{ts}}^T\mathbf{W}^{(i)}\mathbf{G}_{\text{ts}})^+(-a\mathbf{G}_{\text{ts}}^T\mathbf{W}^{(i)}\mathbf{O}), \;\; \mathbf{W}^{(i)} = diag\left(\mathbf{s}^{\text{temp}}\right) \cdot diag\left(\frac{d}{d\mathbf{D}} \rho(a\mathbf{G}_{\text{ts}}\mathbf{D}^{(i)} + \mathbf{O})\right), \tag{12}$$

where $diag(\cdot)$ an operator that produces a diagonal matrix from a given vector, $(\cdot)^+$ is the pseudoinverse, and $\mathbf{D}^{(i)}$ is the inferred depth at iteration i.

Alternaing between $\mathbf{D}$ and a. The unknown factor a is a single constant over the system. We first initialize it as 1, and after each iteration i we use the estimated depth $\mathbf{D}^{(i+1)}$ to solve Eq. (7) + Eq. (8) of the main paper for a. The solution is

$$a = (\mathbf{D}^{(i+1)^T}\mathbf{V}^{(i)}\mathbf{P} - \mathbf{D}^{(i+1)^T}\mathbf{G}_{\text{ts}}^T\mathbf{W}^{(i)}\mathbf{O})/(\mathbf{D}^{(i+1)^T}\mathbf{V}^{(i)}\mathbf{D}^{(i+1)} + \mathbf{D}^{(i+1)^T}\mathbf{G}_{\text{ts}}^T\mathbf{W}^{(i)}\mathbf{G}_{\text{ts}}\mathbf{D}^{(i+1)}). \tag{13}$$

We alternate this 20 times.

5. More Examples

Figures 1-6 illustrate representative examples from the NYU RGB-D test cases, and Figures 7-12 illustrate representative examples from the SUN3D test cases. These results are also shown in the video. Experiments on outdoor footage and non-rigid scenes with camera motion are shown in the video. The outdoor videos are taken from [8] (no ground truth depth data available). Our method performs reasonably well on these challenging examples, which captures the overall scene structure and strong surface discontinuities between objects consistently.

Here we report evaluation on the fully-metric method [8] ([32] in the main paper) that recovers consistent dense depth maps from a video sequence. Unlike ours or Depth Transfer this method is not example-based but formulate in terms of structure from motion and multi-vew stereo methods. It optimizes a metric energy based on photo consistency and geometric coherence. The authors only provide with a GUI based executable that only supports manual user interaction, therefore we had to apply this method to a subset of the SUN3D tests cases (7 scenes, 7 clips in total, 30 frames for each clip) and a subset of NYU RGB-D test cases (9 scenes, 12 clips in total, 30 frames for each clip). We gave the same input frames that were used for SfM reconstruction with VisualSFM. Numerical errors measured over theses examples are shown in Tables 1 and 2. We found that their method performs poorly on indoor scenes in general.

method	**rel**	**log10**	**RMS**
Fully-metric Method [8]	1.076	0.986	4.282
Baseline [5]	1.858	0.318	3.133
Ours	0.386	0.145	1.386

Table 1. Estimated depth maps for a subset of the NYU RGB-D tests cases, including all examples in this document.

method	**rel**	**log10**	**RMS**
Fully-metric Method [8]	1.413	1.049	5.397
Baseline [5]	1.160	0.243	2.939
Ours	0.232	0.078	0.953

Table 2. Estimated depth maps for a subset of the SUN3D tests cases, including all examples in this document.

6. Contour Detection

In Fig. 13 we compare contours from [2] and those from our modified detector. For the contours from [2] we use the model that is trained on RGB - boundary pairs from the single-frame NYU RGB-D dataset [6] with ground truth boundaries [4]. The new model for our modified detector is trained on RGB - albedo - shading - boundary pairs using our modified version of the Sintel dataset [1]. In Fig. 13 we visualize detected contours (the images invert the raw responses for better presentation). Our detection consistently assigns strong prediction values on most relevant surface boundaries in the scene, while the original detection does not consistently give high confidence on the surface boundaries.

Although the images in Fig. 13(a,c,e,g) from the original detection might be perceived somewhat cleaner than (b,d,f,h) from our detection, in fact it is hard to extract good surface boundary maps from the detected images in (a,c,e,g) since their values are not consistently high along the true surface boundaries. For example, if we apply simple thresholding to Fig. 13(a,c,e,g) most surface boundaries are not captured as shown in Fig. 14(a,c,e,g), while from (b,d,f,h) we consistently extract clean boundary maps that mostly obey relevant surface boundaries as in Fig. 14(b,d,f,h). Here we apply the same thresholding $1 - sig(\delta, 50, 0.3)$, where δ is the raw contour detection map and $sig(\cdot)$ is a sigmoid function in Eq. (5) of the main paper.

7. Progressive Improvement

In Figures 15-18 we present progressive improvement of depth estimation as better cues are introduced. In the figures, (a) shows the results from the baseline method [5], (b) shows the results from our method without SfM cues, (c) shows the results from our method with contours from [2] that only relies on RGB values, and (d) shows the results from our full method that fully utilizes SfM cues as well as RGB - albedo - shading values. From (a) to (d), we can observe the results are progressively enhanced both qualitatively and quantitatively. Qualitatively we can see that (a) is over-smooth while almost missing scene structure, (b) captures crude structure with sharper surface boundaries, (c) captures overall structure well but fine surface details are missing, and (d) shows the best quality with strong depth discontinuities and extra fine details.

Figure 1. Intrinsic images. (a) Images from the query video. (b) Albedo. (c) Shading. (d) Contours. (e) Point projections. (f) Proxy depth.

Figure 2. Results for the example in Fig. 1. (a) Intrinsic Depth (ours). (b) Depth Transfer [5]. (c) Fully metric method [8]. (d) Example-based single image method [7]. (e) Ground truth.

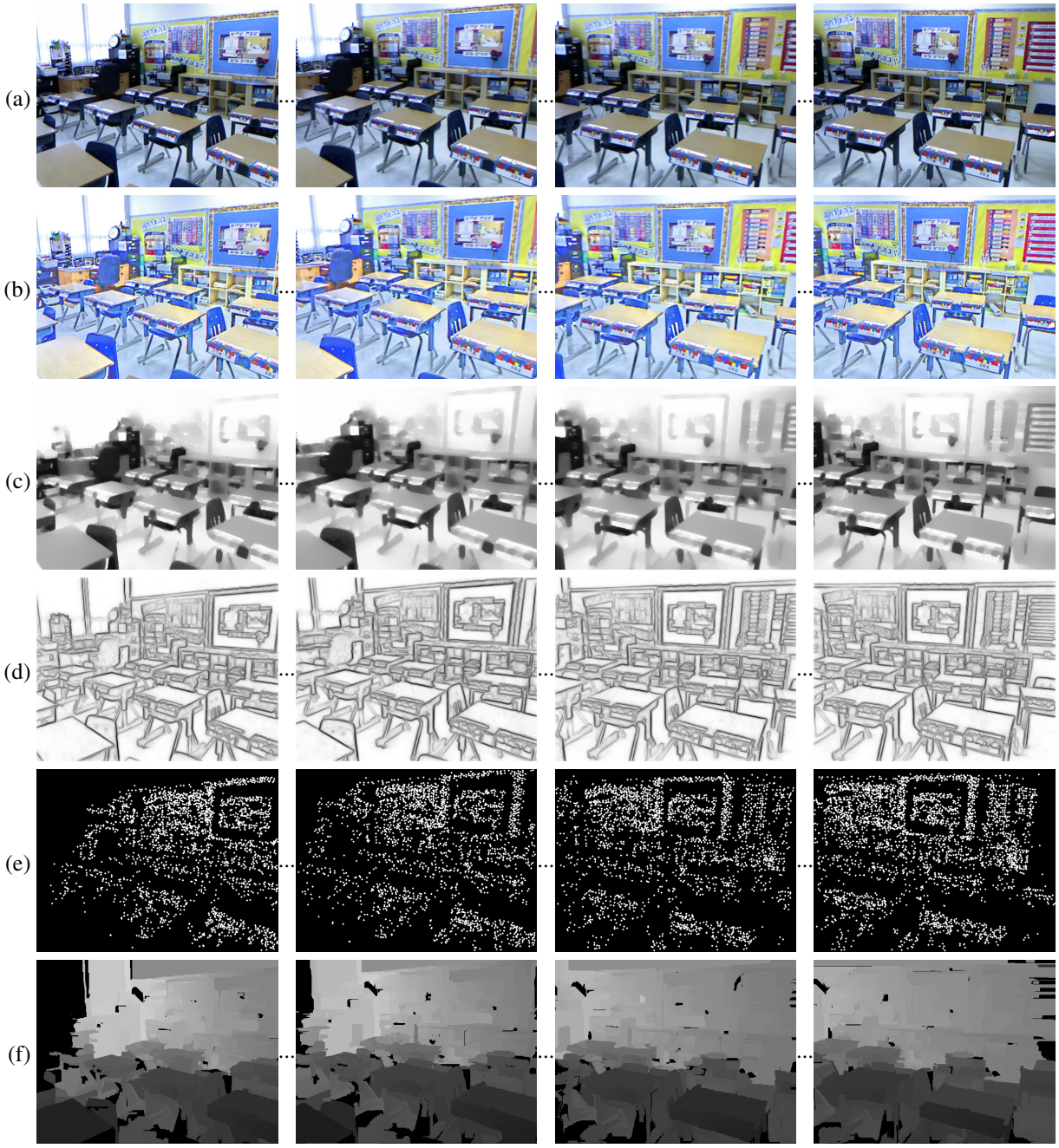


Figure 3. Intrinsic images. (a) Images from the query video. (b) Albedo. (c) Shading. (d) Contours. (e) Point projections. (f) Proxy depth.

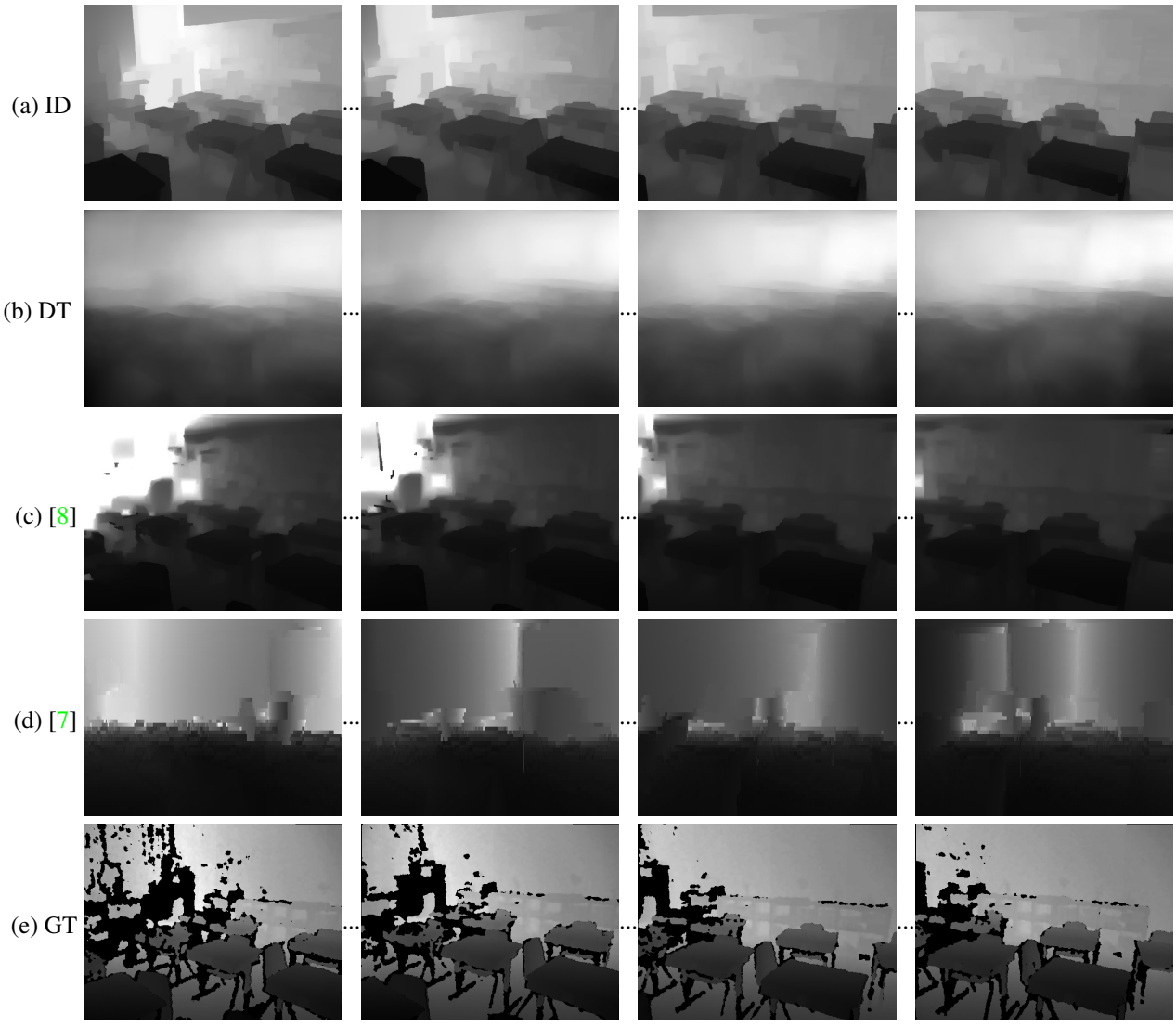


Figure 4. Results for the example in Fig. 3. (a) Intrinsic Depth (ours). (b) Depth Transfer [5]. (c) Fully metric method [8]. (d) Example-based single image method [7]. (e) Ground truth.

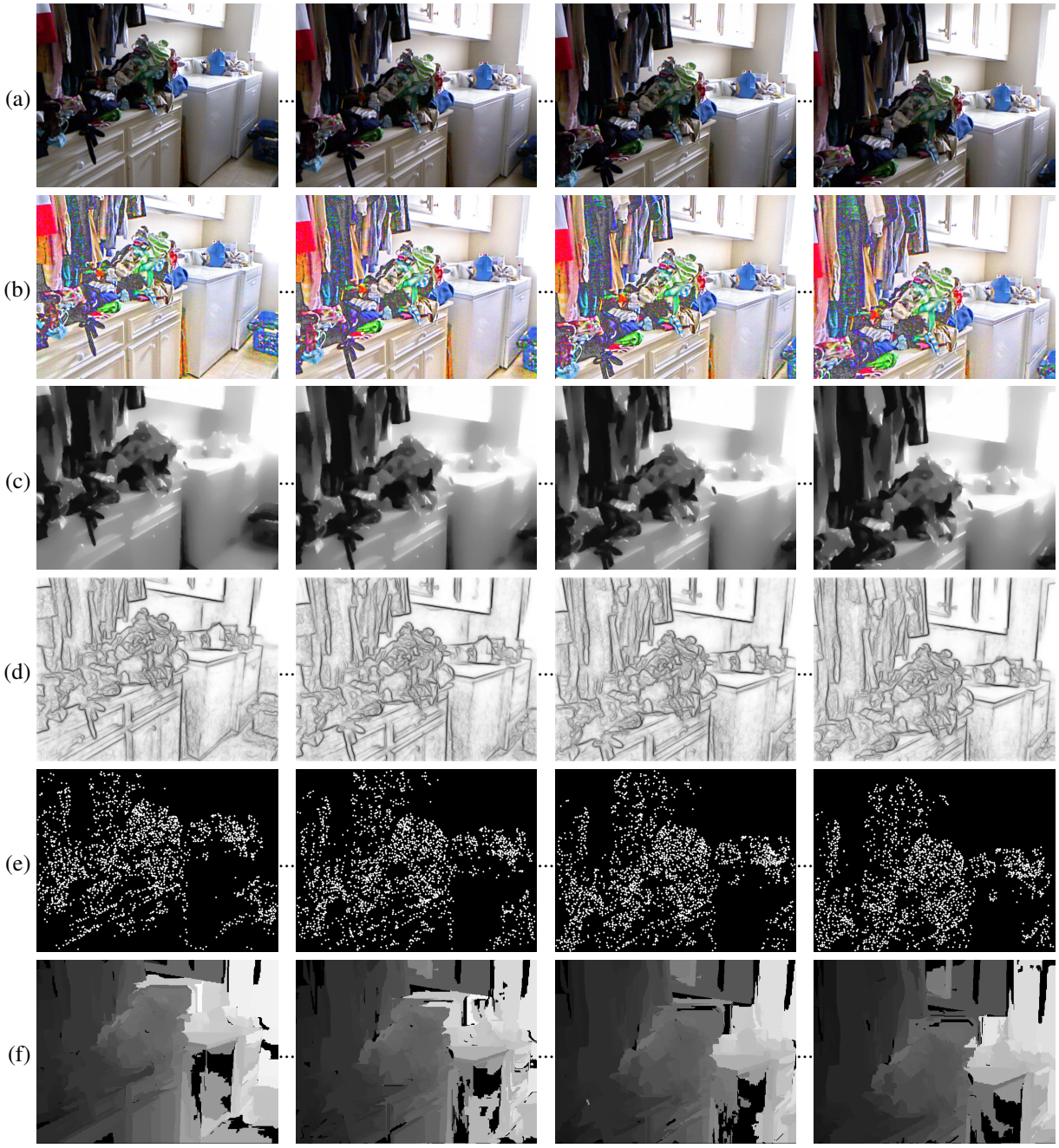


Figure 5. Intrinsic images. (a) Images from the query video. (b) Albedo. (c) Shading. (d) Contours. (e) Point projections. (f) Proxy depth.

Figure 6. Results for the example in Fig. 5. (a) Intrinsic Depth (ours). (b) Depth Transfer [5]. (c) Fully metric method [8]. (d) Example-based single image method [7]. (e) Ground truth.

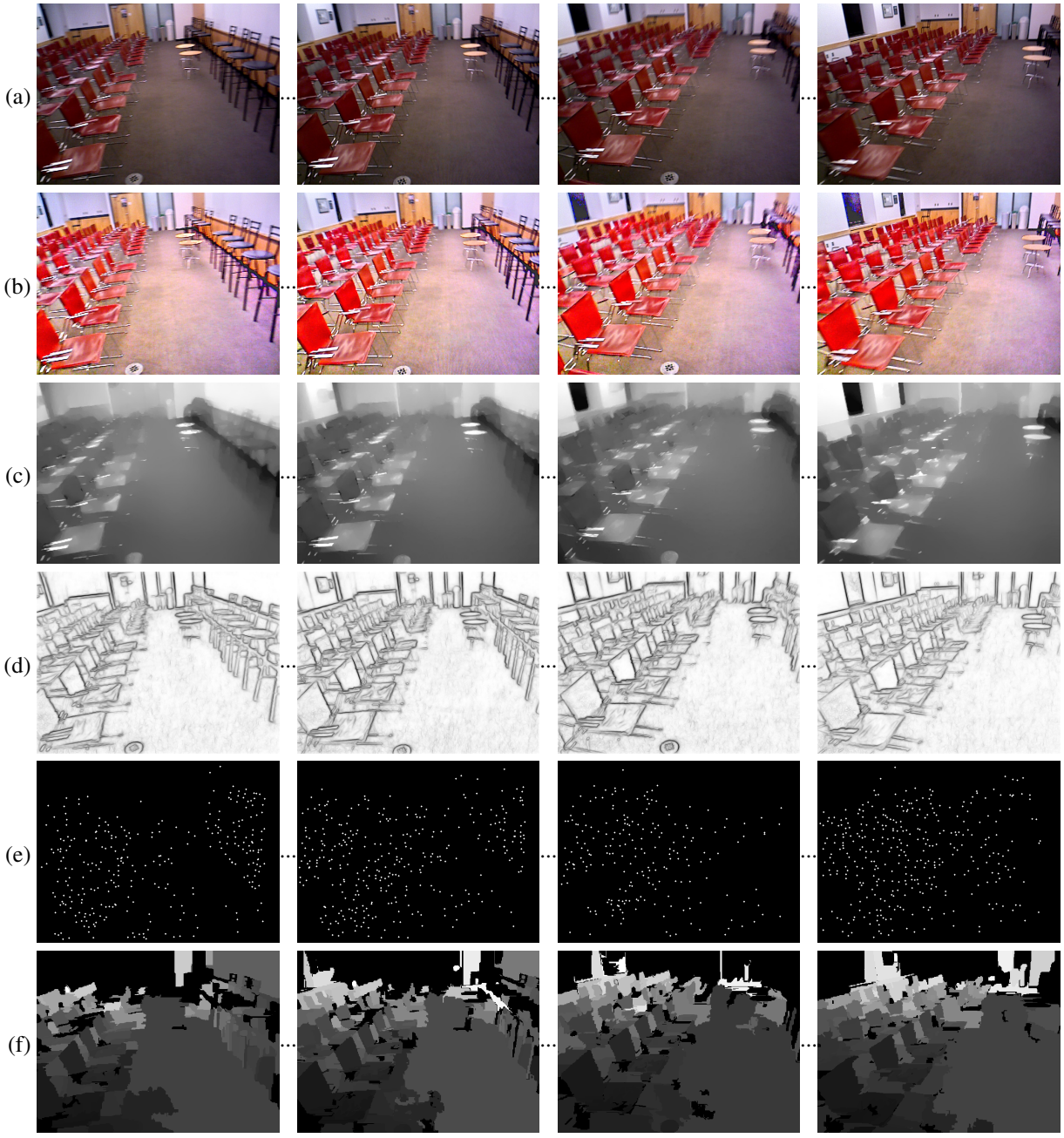


Figure 7. Intrinsic images. (a) Images from the query video. (b) Albedo. (c) Shading. (d) Contours. (e) Point projections. (f) Proxy depth.

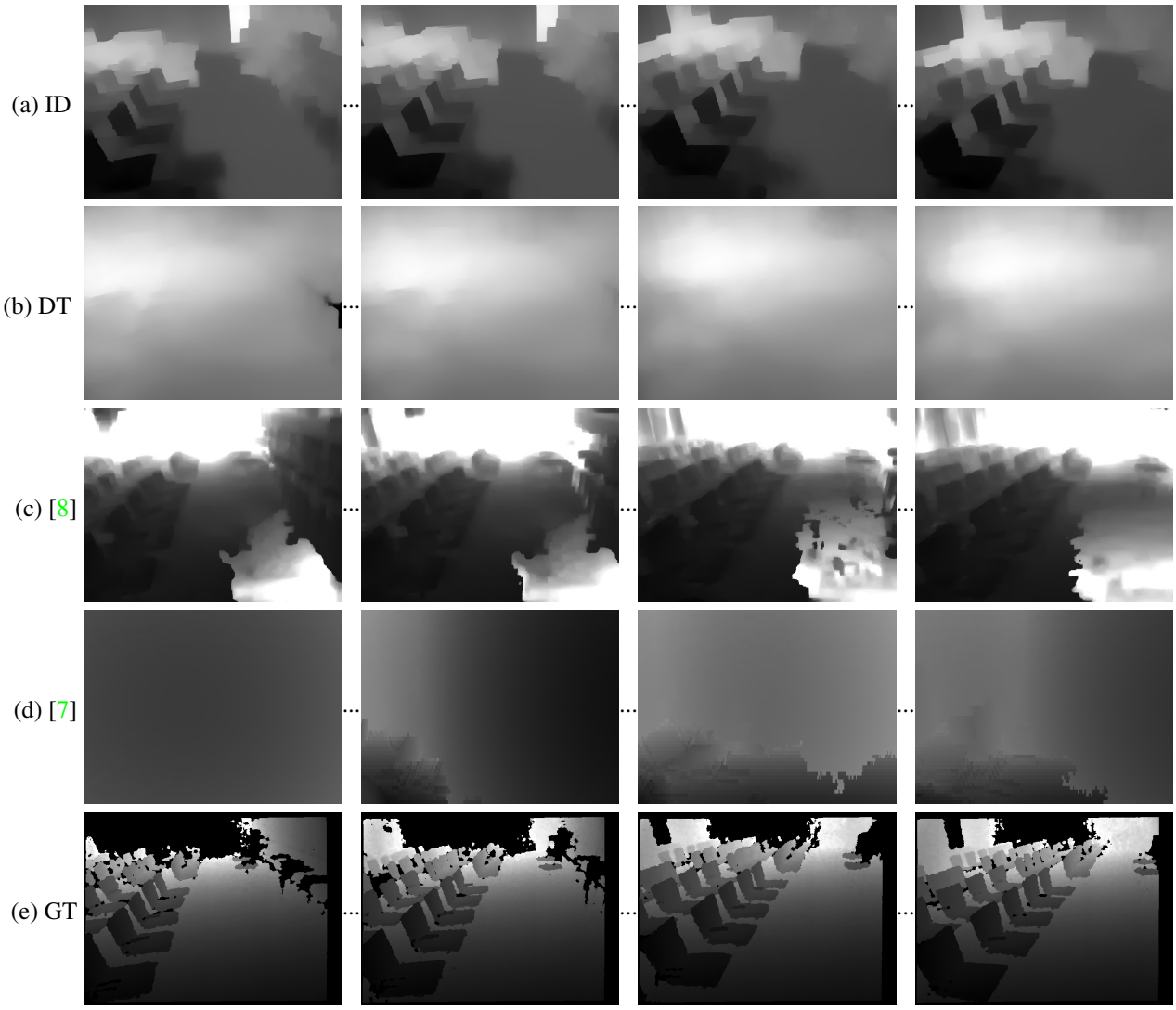


Figure 8. Results for the example in Fig. 1. (a) Intrinsic Depth (ours). (b) Depth Transfer [5]. (c) Fully metric method [8]. (d) Example-based single image method [7]. (e) Ground truth.

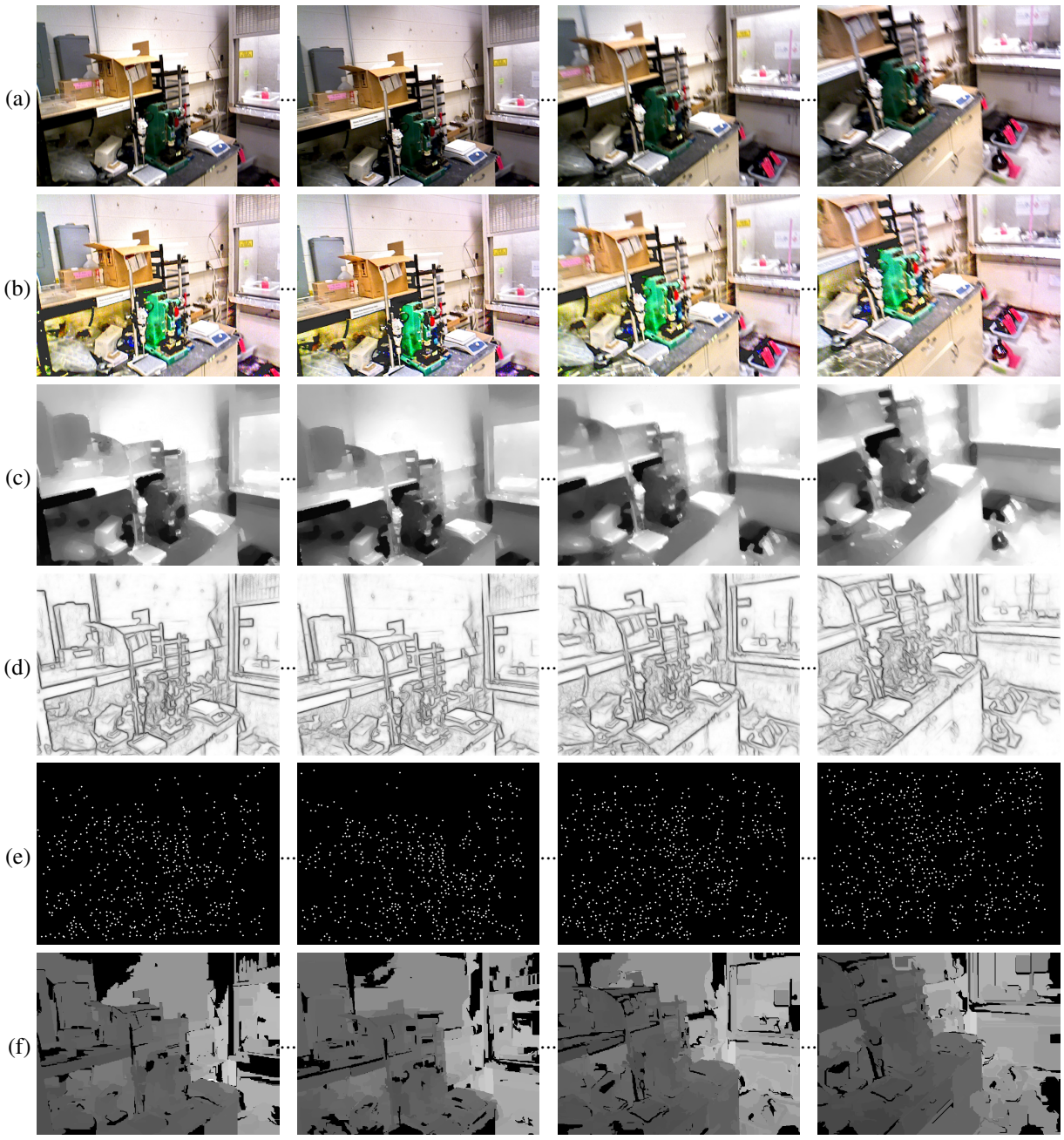


Figure 9. Intrinsic images. (a) Images from the query video. (b) Albedo. (c) Shading. (d) Contours. (e) Point projections. (f) Proxy depth.

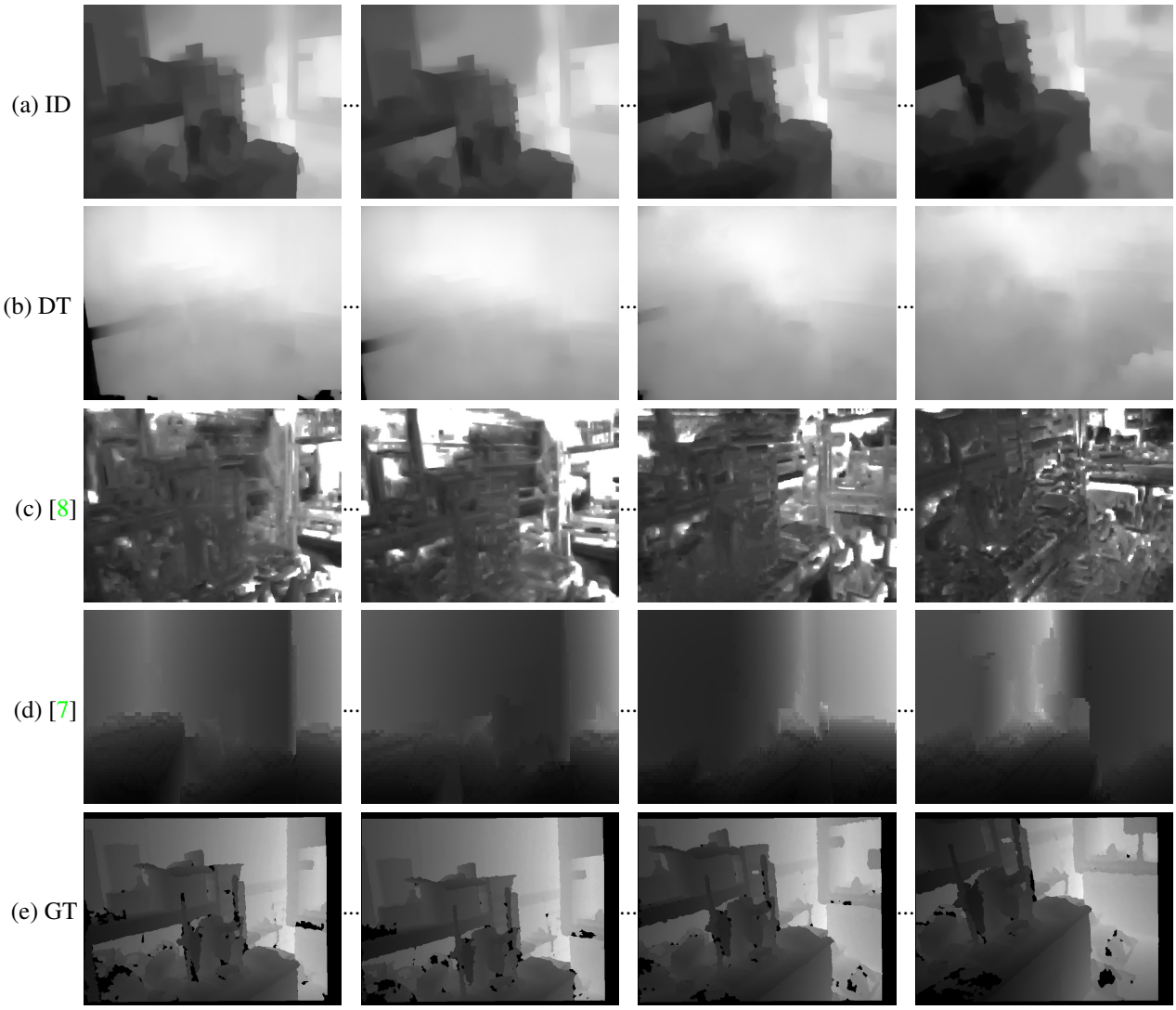


Figure 10. Results for the example in Fig. 1. (a) Intrinsic Depth (ours). (b) Depth Transfer [5]. (c) Fully metric method [8]. (d) Example-based single image method [7]. (e) Ground truth.

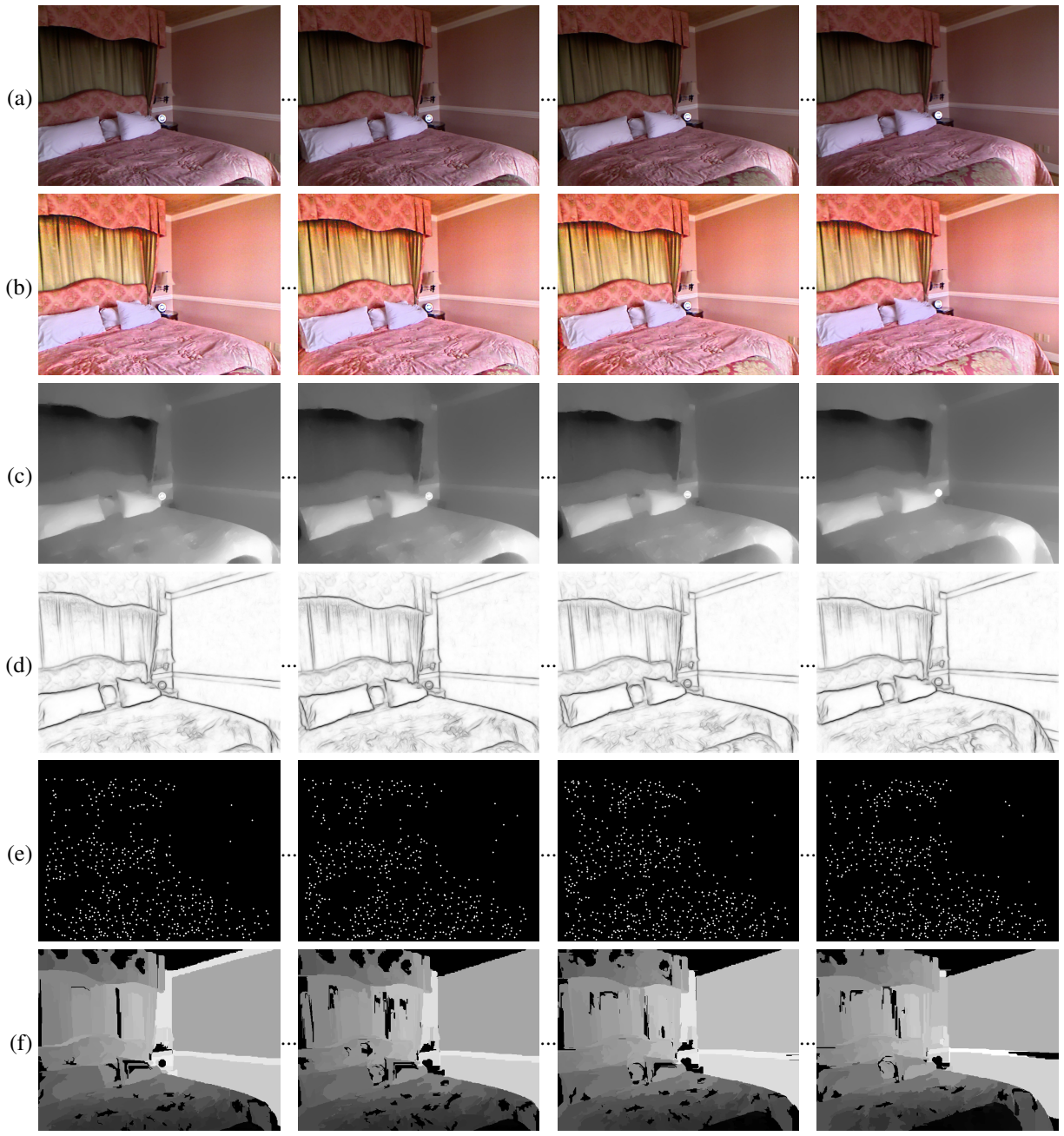


Figure 11. Intrinsic images. (a) Images from the query video. (b) Albedo. (c) Shading. (d) Contours. (e) Point projections. (f) Proxy depth.

Figure 12. Results for the example in Fig. 1. (a) Intrinsic Depth (ours). (b) Depth Transfer [5]. (c) Fully metric method [8]. (d) Example-based single image method [7]. (e) Ground truth.

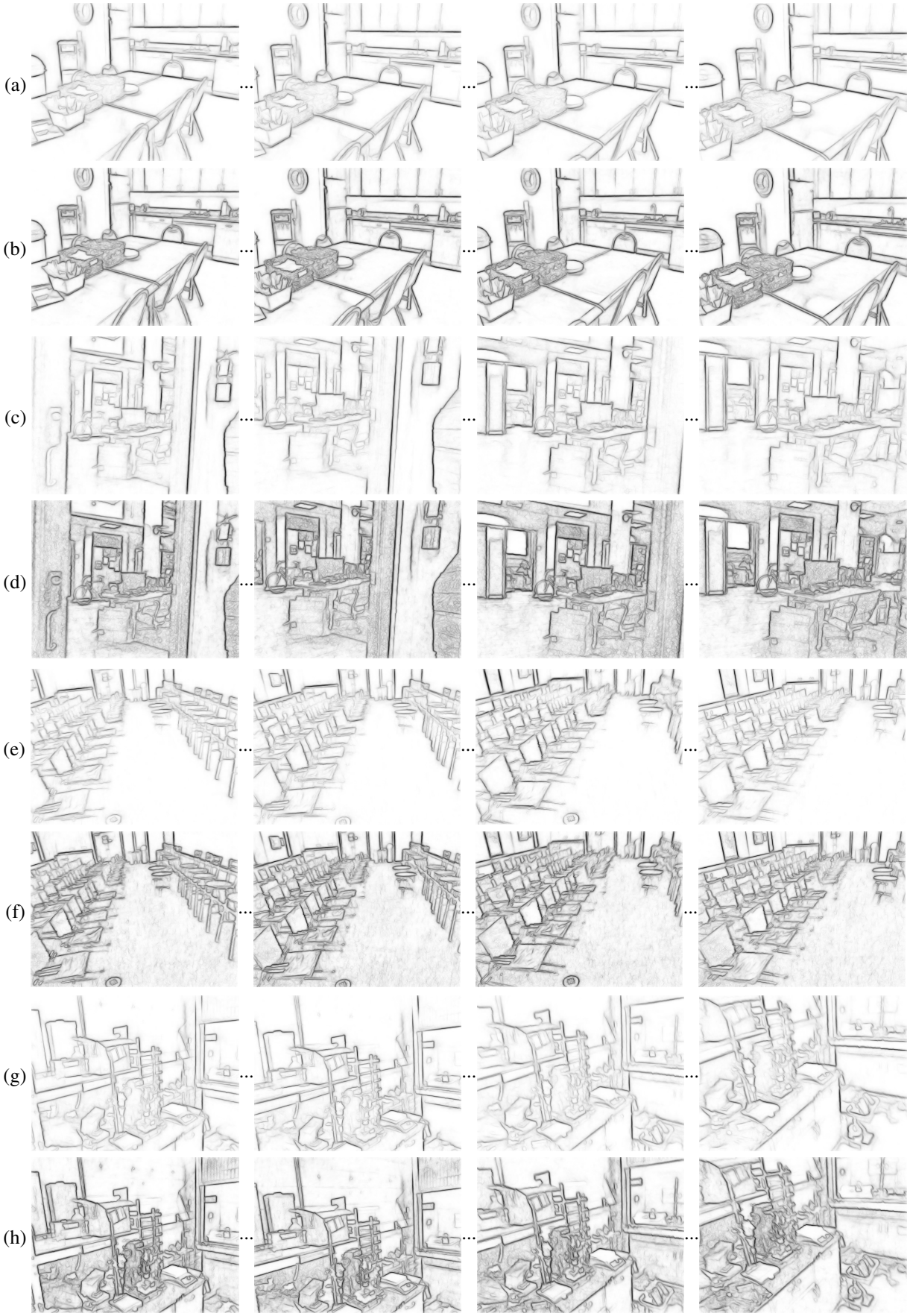


Figure 13. Contour detection. (a),(c),(e),(g) Contours from [2] trained on RGB - boundary pairs for the examples in Fig. 1 of the main paper, Figures. 1, 7 and 9, respectively. (b),(d),(f),(h) Contours from our modified detector trained on RGB - albedo - shading - boundary pairs. Note that images are inverted from the raw detection values for better presentation.

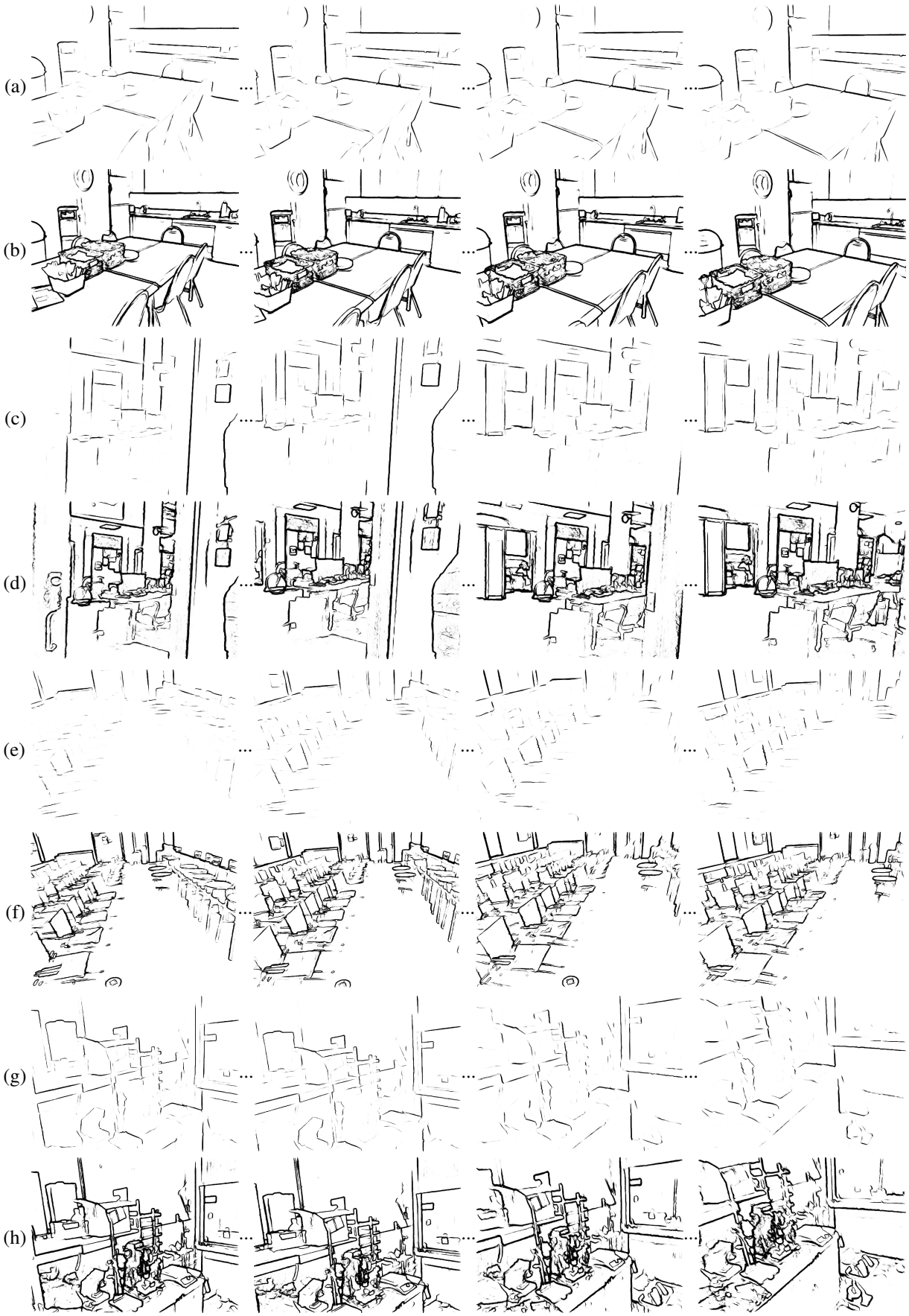
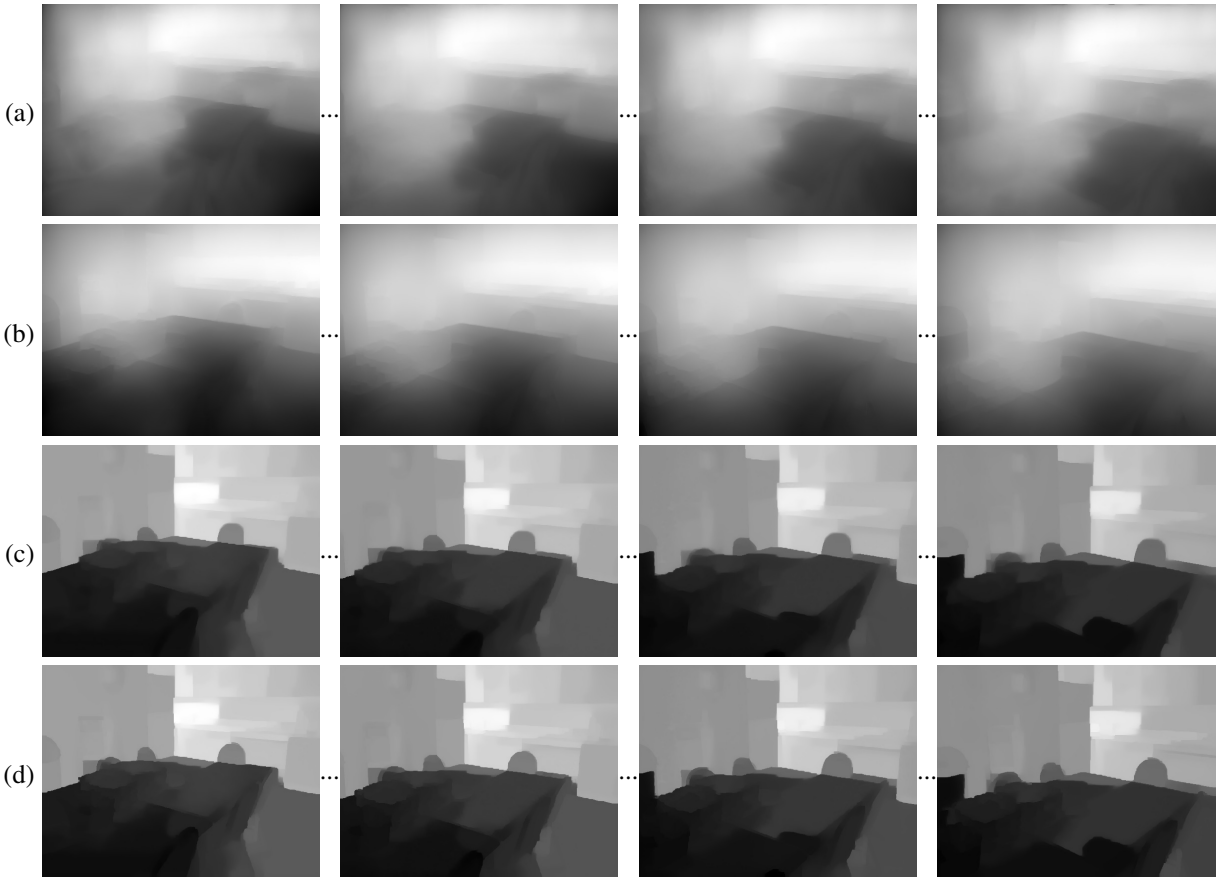
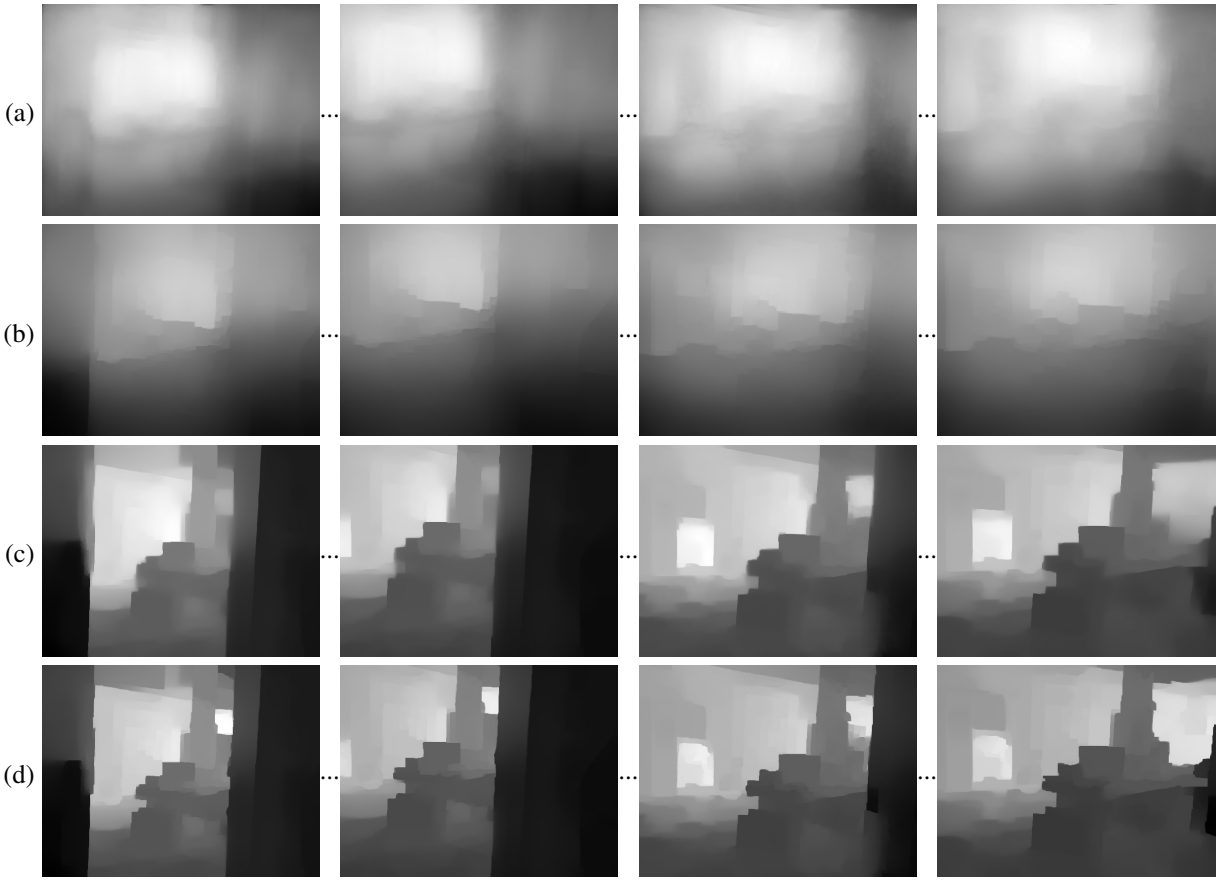


Figure 14. Thresholding on the detection images in Fig. 13. The output shows that our contours in (b,d,f,h) better capture most relevant surface boundaries than the contours from [2] in (a,c,e,g).



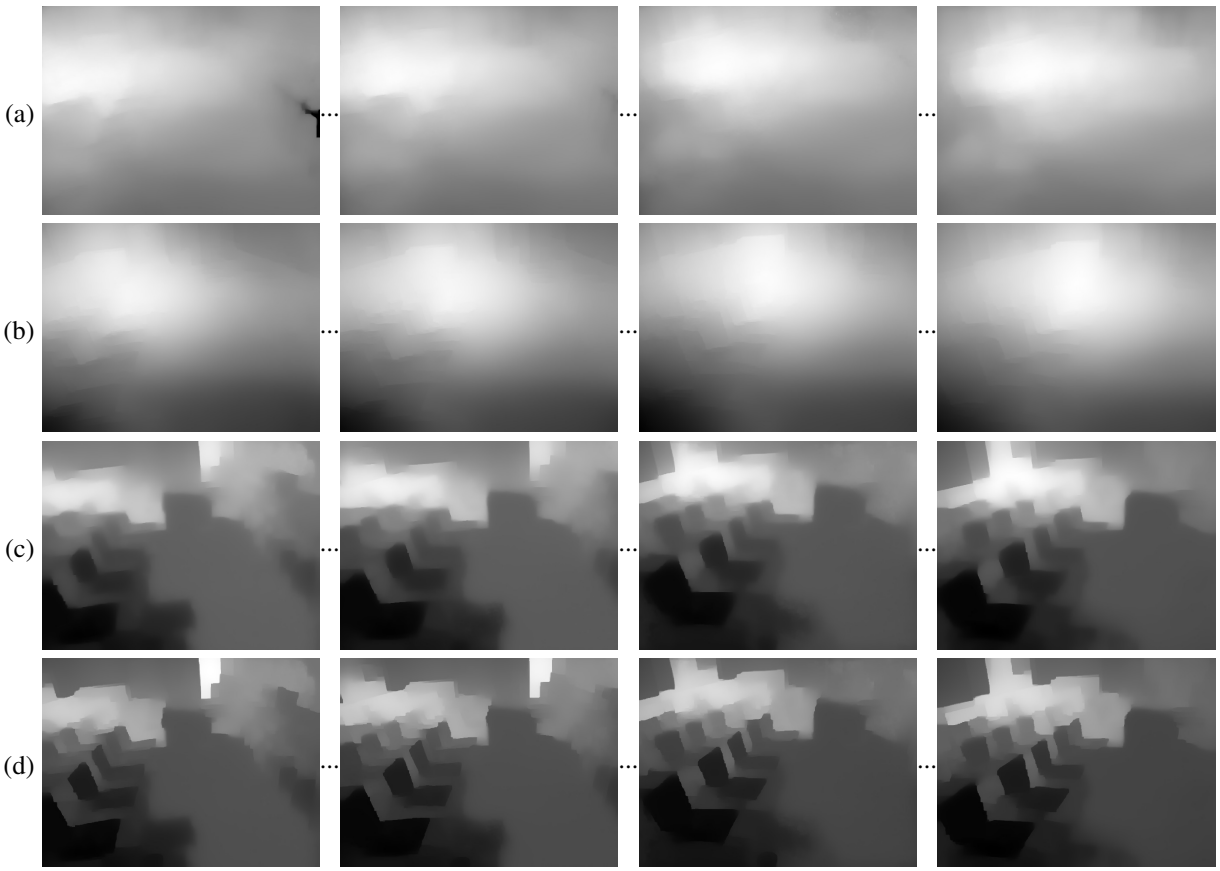
method	rel	log10	RMS
(a)	1.187	0.277	2.489
(b)	0.777	0.218	2.055
(c)	0.141	0.064	0.765
(d)	0.138	0.062	0.762

Figure 15. Progressive improvement for the example in Fig. 1 of the main paper. (a) Baseline method [5]. (b) Our method without SfM cues. (c) Our method with contours from [2] that only relies on pixel RGB values. (d) Our full method. The results are progressively improved qualitatively and quantitatively from (a) to (d).



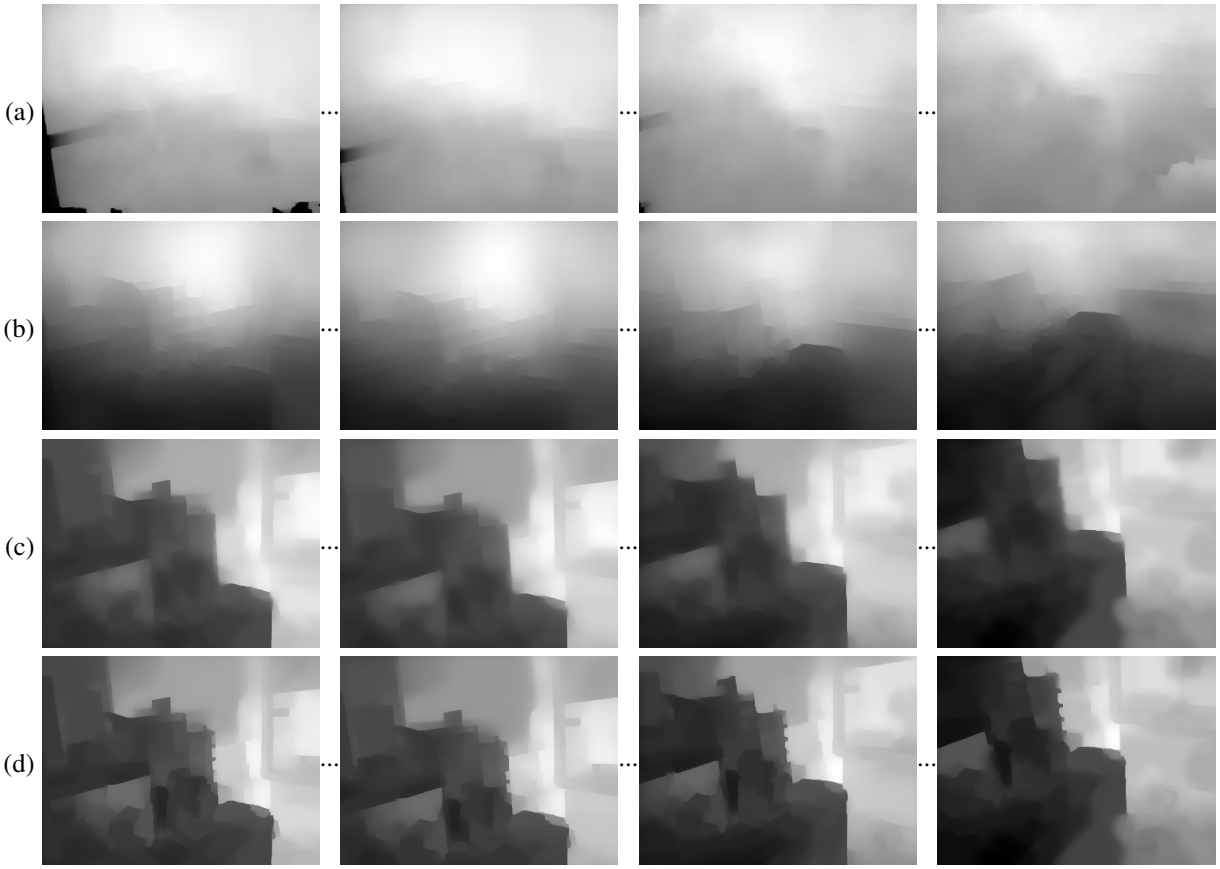
method	rel	log10	RMS
(a)	2.868	0.414	3.252
(b)	2.306	0.313	2.392
(c)	0.930	0.188	1.503
(d)	0.680	0.148	1.126

Figure 16. Progressive improvement for the example in Fig. 1. (a) Baseline method [5]. (b) Our method without SfM cues. (c) Our method with contours from [2] that only relies on pixel RGB values. (d) Our full method. The results are progressively improved qualitatively and quantitatively from (a) to (d).



method	rel	log10	RMS
(a)	2.758	0.486	4.158
(b)	1.780	0.396	3.338
(c)	0.676	0.202	1.659
(d)	0.487	0.164	1.439

Figure 17. Progressive improvement for the example in Fig. 7. (a) Baseline method [5]. (b) Our method without SfM cues. (c) Our method with contours from [2] that only relies on pixel RGB values. (d) Our full method. The results are progressively improved qualitatively and quantitatively from (a) to (d).



method	rel	log10	RMS
(a)	1.391	0.334	4.095
(b)	0.765	0.242	2.872
(c)	0.192	0.073	1.072
(d)	0.158	0.063	0.881

Figure 18. Progressive improvement for the example in Fig. 9. (a) Baseline method [5]. (b) Our method without SfM cues. (c) Our method with contours from [2] that only relies on pixel RGB values. (d) Our full method. The results are progressively improved qualitatively and quantitatively from (a) to (d).

References

[1] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In *Proc. European Conference on Computer Vision (ECCV)*, volume 7577 of *Part IV. LNCS*, pages 611–625, 2012. 4

[2] P. Dollár and C. L. Zitnick. Fast edge detection using structured forests. *IEEE Trans. Pattern Anal. Mach. Intell.*, 37(8):1558–1570, 2015. 4, 17, 18, 19, 20, 21, 22

[3] M. Grundmann, V. Kwatra, M. Han, and I. Essa. Efficient hierarchical graph based video segmentation. *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2141–2148, 2010. 1

[4] S. Gupta, P. Arbelaez, and J. Malik. Perceptual organization and recognition of indoor scenes from RGB-D images. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 564–571, 2013. 4

[5] K. Karsch, C. Liu, and S. B. Kang. Depthtransfer: Depth extraction from video using non-parametric sampling. *IEEE Trans. Pattern Anal. Mach. Intell.*, 36(11):2144–2158, 2014. 1, 4, 6, 8, 10, 12, 14, 16, 19, 20, 21, 22

[6] P. K. Nathan Silberman, Derek Hoiem and R. Fergus. Indoor segmentation and support inference from rgbd images. In *Proc. European Conference on Computer Vision (ECCV)*, volume 7576 of *Part V. LNCS*, pages 746–760, 2012. 4

[7] A. Saxena, M. Sun, and A. Y. Ng. Make3D: Learning 3D scene structure from a single still image. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(5):824–840, May 2009. 6, 8, 10, 12, 14, 16

[8] G. F. Zhang, J. Y. Jia, T. T. Wong, and H. J. Bao. Consistent depth maps recovery from a video sequence. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(6):974–988, June 2009. 1, 3, 4, 6, 8, 10, 12, 14, 16