Depth Extraction from Video Using Non-parametric Sampling

Supplemental material

Kevin Karsch
University of Illinois at Urbana-Champaign

Ce Liu
Microsoft Research New England

Sing Bing Kang
Microsoft Research

In this supplementary file, we provide further details of our depth inference system as well as the application of 2D-to-3D video conversion. These details are not critical for the understanding of our paper, but enables the reader to get a more complete picture of our system. Additional results are presented at the end of this document. Our dataset, video results and code can be found online at: http://kevinkarsch.com/depthtransfer

1 Implementation details for depth inference

Here we describe the details of candidate selection and inference.

1.1 Features for candidate image matching

In order to find candidate images which match the input image/sequence semantically and in terms of depth distribution, we use a combination of GIST features [1] and features derived from optical flow. To create flow features for a video, we first compute optical flow (again using Liu's implementation [2]) for each pair of consecutive frames, which defines a warping from frame i to frame $i+1$. If the input is a single image, or the last image in a video sequence, we consider this warping to be the identity warp. Then, we segment the image into $b \times b$ uniform blocks, and compute the mean and standard deviation over the flow field in each block, for both components of the flow (horizontal and vertical warpings), and for the second moments as well (each component squared). This leads to 8 features per block, for a total of $8b^2$ features per image. We use $b = 4$ for our results.

To determine the matching score between two images, we take a linear combination of the difference in GIST and optical flow features described above. Denoting G_1, G_2 and F_1, F_2 as the gist and flow feature vectors for two images respectively, we define the matching score as

$$(1-\omega)||G_1 - G_2|| + \omega||F_1 - F_2||, \tag{1}$$

where $\omega = 0.5$ in our implementation.

1.2 Non-uniform candidate depth weights

During optimization, we ensure that the inferred depth is similar to each of the K warped candidate depths. However, some of the candidate depth values will be more reliable than others, and we model this reliability with a confidence weighting for each pixel in each candidate image (e.g. $w_i^{(j)}$ is the weight of the i^{th} pixel from the j^{th} candidate image). We compute these weights by comparing per-pixel SIFT descriptors, obtained during the SIFT flow computation, of both the input image and the candidate images:

$$w_i^{(j)} = (1 + e^{(||\mathbf{S}_i - \psi_j(\mathcal{S}_i^{(j)})|| - 0.5)/.01})^{-1}, \tag{2}$$

where $\mathbf{S}_i$ and $\mathcal{S}_i^{(j)}$ are the SIFT feature vectors at pixel i in candidate image j. Notice that the candidate image's SIFT features are computed first, and then warped using the warping function (ψ_j) calculated with SIFT flow.

1.3 Image-dependent smoothness weights

We use a spatial regularization term in our optimization, but we do not want the smoothness applied uniformly to the inferred depth, as there is some relation between image appearance and depth. Therefore, we assume that regions in the image with similar texture are likely to have similar, smooth depth transitions, and that discontinuities in the image are likely to correspond to discontinuities in depth. We enforce this with a per-pixel weighting of the spatial regularization term such that this weight is large where the image gradients are small, and vice-versa. We determine this weighting by applying a sigmoidal function to the gradients, which we found to produce more pleasing inferred depth maps than using other boundary detecting schemes such as [3,4].

Similar reasoning is applied for temporal smoothness. We can measure the confidence of the optical flow by computing the reprojection error of the flow, which we use to weight the temporal smoothness terms in the coherence term of our video objective function.

1.4 Minimizing the objective function

To minimize the objective function, we use iteratively reweighted least squares (IRLS). We choose IRLS because it is a fast alternative for solving unconstrained, nonlinear minimization problems such as ours. IRLS works by approximating the objective by a linear function of the parameters, and solving the system by minimizing the squared residual (e.g. with least squares); repeating until convergence.[1]

In the general case of videos, the size of this system can be very large (number of pixels $\times$ number of frames squared), although it will be sparse because of the limited number of pairwise interactions in the optimization. Still, given

[1] For further details and discussion of IRLS, see the appendix of Liu's thesis [2].

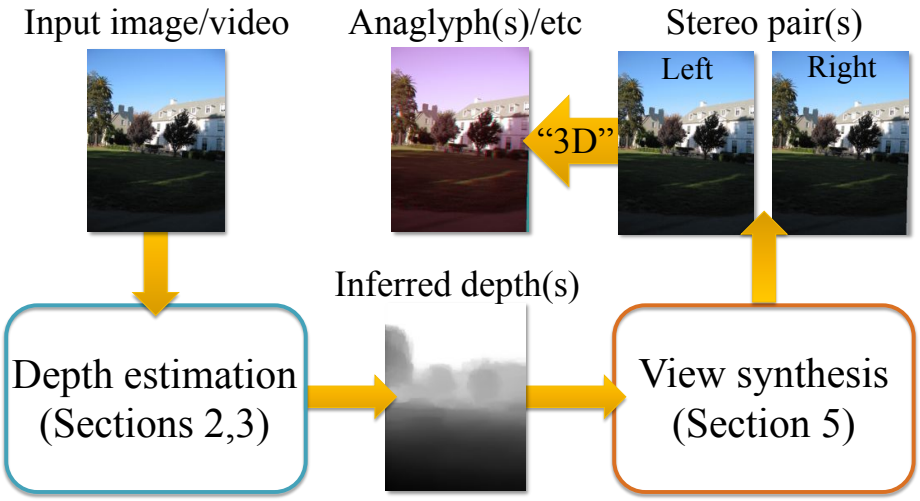

Fig. 1: Our technique for automatically converting ordinary, 2D images and video into stereo for 3D visualization. Given an input image or video, we infer depth, and use this depth to synthesize stereoscopic views for 3D viewing. Section numbers refer to the main paper.

modern hardware limitations, we cannot solve this system directly, so we also must use an iterative method to solve the least squares system at each iteration of our IRLS procedure; we use preconditioned conjugate gradient and construct a preconditioner using incomplete Cholesky factorization.

Because we use iterative optimization, starting from a good initial estimate is helpful for quick convergence with fewer iterations. We have found that initializing with some function of the warped candidate depths provide a decent starting point, and we use the median value (per-pixel) of all candidate depths in our implementation.

One issue is that this optimization can require a great deal of data to be stored concurrently in memory (several GBs for a reasonably-sized clip of a few seconds). Solving this optimization efficiently, both in terms of time and space, is beyond the scope of our paper.

2 Automatic stereoscopic view synthesis

With depth estimated for the video sequence (or single image), we can now perform depth image-based rendering (DIBR) to synthesize a new view for stereoscopic display (see Fig 1 for a complete pipeline). A typical strategy for DIBR is to simply reprojecting pixels based on depth values to a new, synthetic camera, but such methods are susceptible to large "holes" at disocclusions. Much work has been done to fill these holes (e.g. [5,6,7,8]), but visual artifacts still remain in the case of general scenes.

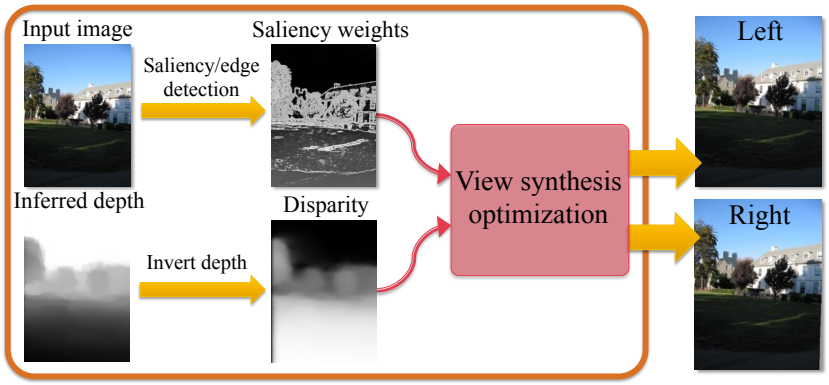

Fig. 2: Summary of the view synthesis procedure for a single image, as described by Wang et al. [9]. Given an image and corresponding depth, we compute salient regions and disparity, and compute stereoscopic images by warping the input image. We extend this method to handle videos, as in Eq 4.

We propose a novel extension to a recent DIBR technique which uses image warping to overcome problems such as disocclusions and hole filling. Wang et al. [9] developed a method that takes as input a single image and per-pixel disparity values, and intelligently warps the input image based on the disparity such that highly salient regions remain unmodified. This method is illustrated in Fig 2. The idea is that people are less perceptive of errors in low saliency regions, and thus disocclusions are covered by "stretching" the input image where people tend to not notice artifacts. This method was only applied to single images, and we show how to extend this method to handle video sequences as well.

Given an input image and depth values, we first invert the depth to convert it to disparity, and scale the disparity by the maximum disparity value:

$$\mathbf{W}_0 = \frac{\mathbf{W}_{\max}}{\mathbf{D} + \epsilon}, \tag{3}$$

where $\mathbf{W}_0 = \{W_1, \ldots, W_n\}$, $\mathbf{D} = \{D_1, \ldots, D_n\}$ is the initial disparity and depth (resp) for each of the n frames of the input, and $\mathbf{W}_{\max}$ is a parameter which modulates how much objects "pop-out" from the screen when viewed with a stereoscopic device. Increasing this value enhances the "3D" effect, but can also cause eye strain or problems with fusing the stereo images if set too high. We set $\epsilon = 0.01$.

Then, to implement the saliency-preserving warp (which in turn defines two newly synthesized views), minimize the following unconstrained, quadratic objective:

$$\begin{aligned} Q(\mathbf{W}_i) &= \sum_{i \in \text{pixels}} Q_{\text{data}}(\mathbf{W}_i) + Q_{\text{smooth}}(\mathbf{W}_i), \\ Q_{\text{data}}(\mathbf{W}_i) &= l_i(\mathbf{W}_i - \mathbf{W}_{0,i})^2, \\ Q_{\text{smooth}}(\mathbf{W}_i) &= \lambda(s_{x,i}||\nabla_x \mathbf{W}_i||^2 + s_{y,i}||\nabla_y \mathbf{W}_i||^2) + \\ &\quad \mu s_{t,i}||\nabla_{flow} \mathbf{W}_i||^2, \end{aligned} \tag{4}$$

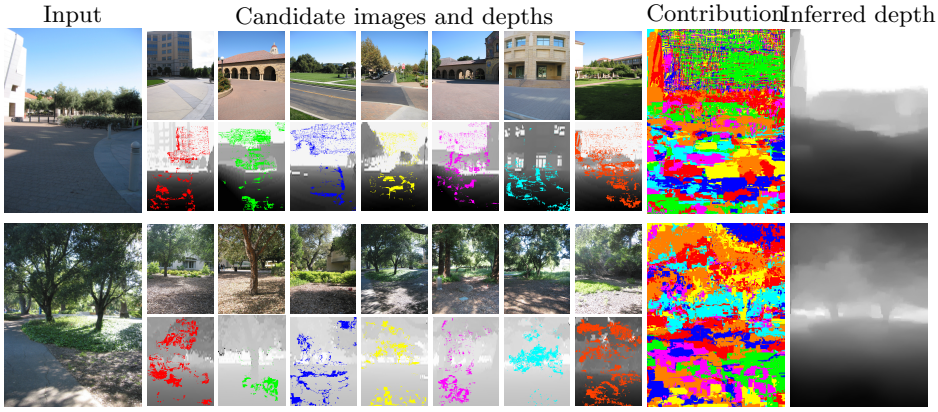

Fig. 3: Candidate contribution for depth estimation. Given an input image (*left*), we find top-matching candidate images and depths (*middle*), and infer depth for the input image (*right*) using our technique. The contribution image is color-coded to show the sources; red pixels indicate that the left-most candidate influenced the inferred depth image the most, orange indicates contribution from the right-most candidate, and so on. Notice that our algorithm finds candidate images with very similar scene layout and structure as the input image, and generally chooses reasonable candidate pixels to sample depth from. All depth maps are displayed at the same scale.

where l_i is a weight based on image saliency and initial disparity values that constrains disparity values corresponding to highly salient regions and very close objects to remain unchanged, and is set to $l_i = \frac{W_{0,i}}{W_{\max}} + (1 + e^{-(||\nabla \mathbf{L}_i|| - 0.01)/0.002})^{-1}$. The Q_{smooth} term contains the same terms as in our spatial and temporal smoothness functions in our depth optimization's objective function, and λ and μ control the weighting of these smoothness terms in the optimization; we set $\lambda = \mu = 10$. As in the main paper, we set $s_{x,i} = (1 + e^{(||\nabla_x \mathbf{L}_i|| - 0.05)/.01})^{-1}$, $s_{y,i} = (1 + e^{(||\nabla_y \mathbf{L}_i|| - 0.05)/.01})^{-1}$, and $s_{t,i} = (1 + e^{-(||\nabla_{flow} \mathbf{L}_i|| - 0.05)/.01})^{-1}$, where $\nabla_x \mathbf{L}_i$ and $\nabla_y \mathbf{L}_i$ are image gradients in the respective dimensions (at pixel i), and $\nabla_{flow} \mathbf{L}_i$ is the flow difference across neighboring frames (gradient in the flow direction).

With this formulation, we ensure spatial and temporal coherence and most importantly that highly salient regions remain intact during view warping.

After optimization, we divide the disparities by two ($\mathbf{W} \leftarrow \frac{\mathbf{W}}{2}$), and use these halved values to render the input frame(s) into two new views (corresponding to the stereo left and right views). We choose this method, as opposed to only rendering one new frame with larger disparities, because people are less perceptive of a many small artifacts when compared with few large artifacts [9]. For rendering, we use the anisotropic pixel splatting method described by Wang et al. [9], which "splats" input pixels into the new view (based on $\mathbf{W}$) as weighted, anisotropic Gaussian blobs.

With the two synthesized views, we can convert to any 3D viewing format, such as anaglyph or interlaced stereo. For the results in this paper, we use

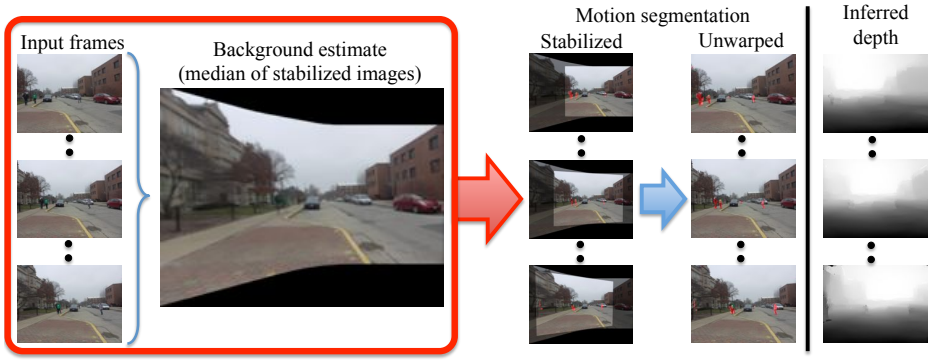

Fig. 4: Example of our motion segmentation applied to a rotating sequence. We first estimate homographies to stabilize the video frames and create a clean background image using a temporal median filter. We then evaluate our estimate of motion thresholding metric on the stabilized sequences, and unwarp the result (via the corresponding inverse homography) to segment the motion in the original sequence. We can then improve our inferred depth using this segmentation. Note that this technique is applicable to all video sequences that do not contain parallax induced from camera motion.

the anaglyph format as cyan/red anaglyph glasses are more widespread than polarized/autostereoscopic displays (used with interlaced 3D images). To reduce eye strain, we shift the left and the right images such that the nearest object has zero disparity, making the nearest object appear at the display surface, and all other objects appear *behind* the display. This is commonly known as the "window" metaphor [10].

3 Dataset collection and hardware limitations

Because the Kinect estimates depth by triangulating a pattern of projected infrared (IR) dots, multiple Kinects can interfere with each other, causing noisy and inaccurate depth. Thus, we mask out the depth sensor/IR projector from the rightmost Kinect, and only collect depth corresponding to the left views. This is suitable for our needs, as we only need estimate depth for the input (left) sequence.

The depth estimated by the Kinect is also susceptible to "holes" in the data caused typically by surface properties, disoccluded/interfered IR pattern, or because objects are simply to far away from the device. For training, we disregard all pixels of the videos which contain holes, and for visualization, we fill the holes in using a naïve horizontal dilation approach.

Due to these limitations, the data we have collected is typically on the scale of rooms, and most objects are approximately 1-8 meters from the cameras. Sunlight also causes IR interference, thus we have only collected indoor scenes in our dataset.

Kinects can capture RGB and depth data at up to 30Hz; however, this puts a significant strain on the USB bus that communicates the Kinect data to the computer. Furthermore, when two Kinects transfer data simultaneously (as in our setup), the bandwidth requirement is too great too handle with one USB bus at 30Hz, which corrupts the data stream. Thus, we only collect data at 20Hz, which we found empirically to be the highest data rate that avoids this corruption issue and is still suitable for our purposes. Ideally, the data collection rig would include a computer with multiple USB busses, but this would likely require a desktop PC rather than a laptop.

4 Additional results

Additional results are provided in the remainder of this document. Fig 3 demonstrates how each candidate depth contributes to the final inferred depth map. We further illustrate our motion segmentation algorithm (applicable to any non-translating video) in Fig 4. Fig 5 gives a visual comparison of our inferred depths with the ground truth depths found in the Make3D database. Further results on indoor and outdoor sequences are shown in Figs 6-8; these sequences contain moving objects and/or stationary, rotating, or variable focal length views. Finally, 3D results on the feature film *Charade* are shown in Fig 9.

References

1. Oliva, A., Torralba, A.: Modeling the shape of the scene: A holistic representation of the spatial envelope. IJCV **42** (2001) 145–175
2. Liu, C.: Beyond pixels: Exploring new representations and applications for motion analysis. PhD thesis, MIT (2009)
3. Hoiem, D., Stein, A., Efros, A., Hebert, M.: Recovering occlusion boundaries from a single image. In: ICCV. (2007)
4. Maire, M., Arbelaez, P., Fowlkes, C., Malik, J.: Using contours to detect and localize junctions in natural images. In: CVPR. (2008)
5. Colombari, A., Fusiello, A., Murino, V.: Continuous parallax adjustment for 3D-TV. IEEE Eur. Conf. Vis. Media Prod. (2005) 194–200
6. Klein Gunnewiek, R., Berretty, R.P., Barenbrug, B., Magalhães, J.: Coherent spatial and temporal occlusion generation. In: Proc. SPIE, Stereoscopic Displays and Applications XX. Volume 7237. (2009)
7. Luo, K., Li, D., Feng, Y., M., Z.: Depth-aided inpainting for disocclusion restoration of multi-view images using depth-image-based rendering. J. Zhejiang Univ. Sci. A **10** (2009) 1738–1749
8. Zhang, L., Vazquez, C., Knorr, S.: 3D-TV content creation: Automatic 2D-to-3D video conversion. IEEE Trans. on Broadcasting **57** (2011) 372–383
9. Wang, O., Lang, M., Frei, M., Hornung, A., Smolic, A., Gross, M.: StereoBrush: Interactive 2D to 3D conversion using discontinuous warps. In: SBIM. (2011)
10. Koppal, S., Zitnick, C., Cohen, M., Kang, S., Ressler, B., Colburn, A.: A viewer-centric editor for 3D movies. IEEE Computer Graphics and Applications **31** (2011) 20–35

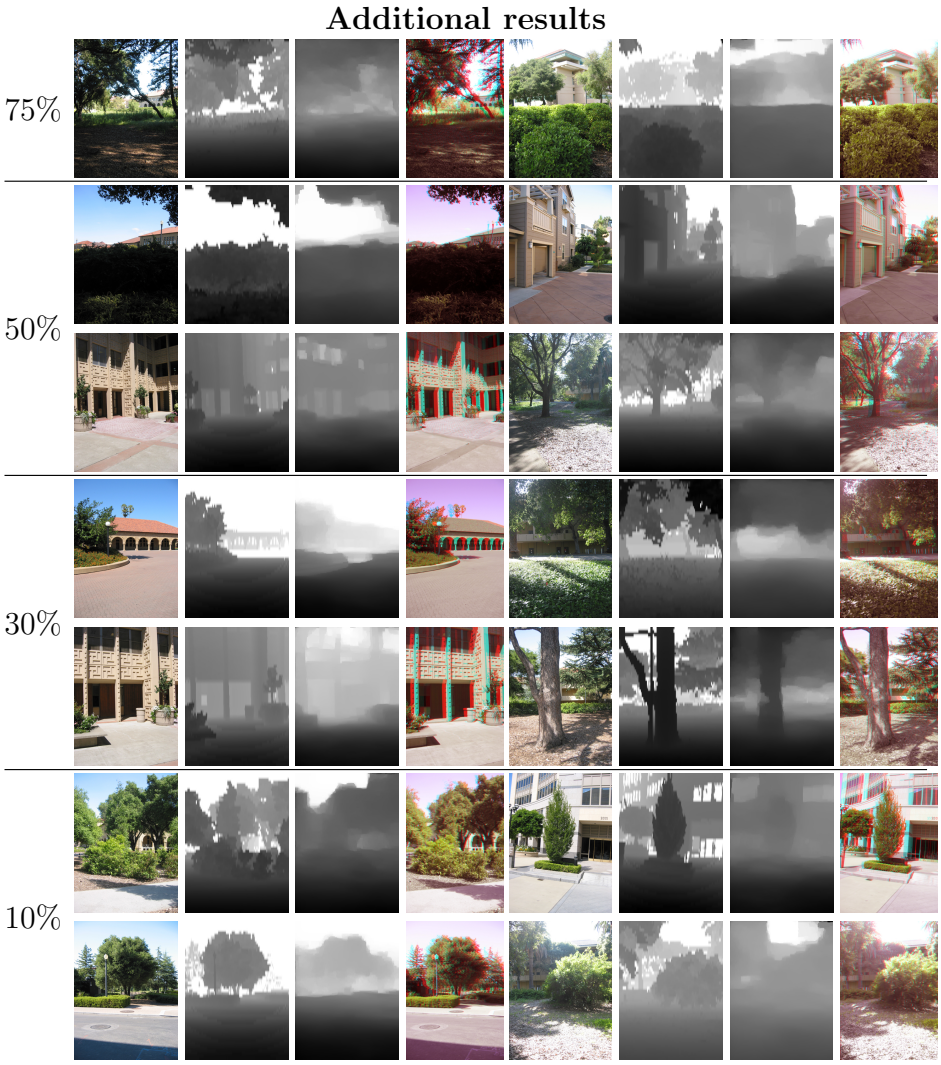

Fig. 5: Single image results obtained on test images from the Make3D dataset. Each result contains the following four images (from left to right): original photograph, ground truth depth from the dataset, our inferred depth, and our synthesized 3D anaglyph image. Results are partitioned by percentile; determined by average depth error ($\log_{10}$, relative, RMS) rankings (e.g. 50% indicates these results have lower error than at least 50% of the other test images). The depth images are shown in log scale, and darker pixels indicate nearby objects (black is roughly 1m away) and lighter pixels indicate objects farther away (white is roughly 80m away). Each pair of ground truth and inferred depths are displayed at the same scale.

Fig. 6: Video results obtained on test images from our stereo RGBD dataset. Each result contains the following four images (from left to right): original photograph, ground truth depth, our inferred depth, and our synthesized 3D anaglyph image. Each pair of inferred and ground truth depths are displayed in log space at the same scale.

Fig. 7: Video results obtained on images captured with our stereo Kinect rig using the Make3D range image dataset for training. Each result contains the following four images (from left to right): original photograph, our inferred depth, ground truth anaglyph image, and our synthesized 3D anaglyph image. Since we could not measure depth for these images (see Sec 3), we display the ground truth anaglyph result from the our side-by-side Kinect rig. Notice that because the ground truth 3D images are recorded with a fixed interocular distance (roughly 5cm), we cannot control the amount of "pop-out," and the 3D effect is subtle. However, this is a parameter we can set using our automatic approach to achieve a desired effect, which allows for an enhanced 3D experience.

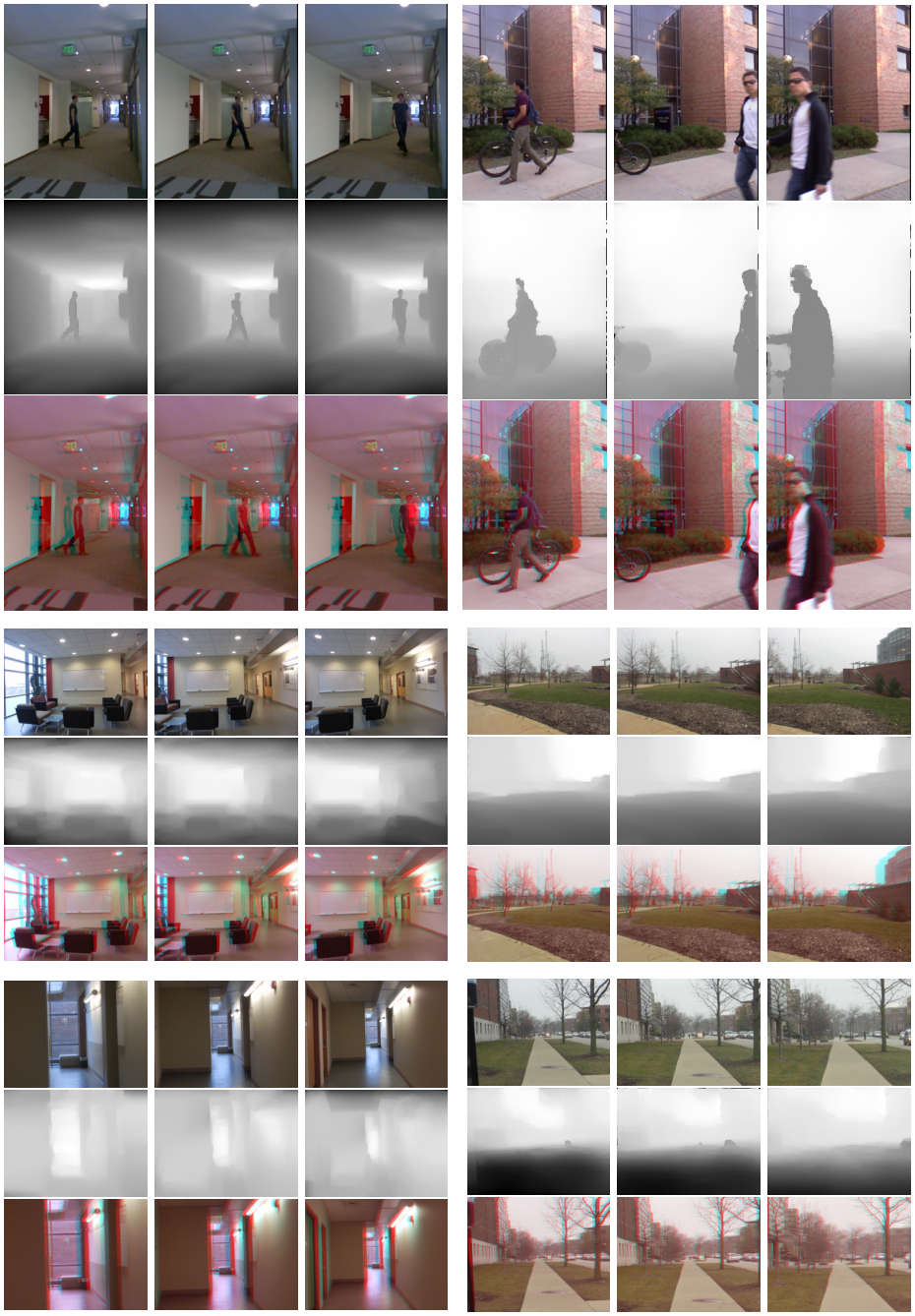

Fig. 8: Additional depth and anaglyphs extracted from videos using our approach. Our algorithm is applicable both indoors and outdoors, and can handle dynamic scenes, as well as stationary, rotating, and zooming views. In each 3×3 block of images, we show the input frames (*top*), inferred depth (*middle*) and inferred 3D anaglyph (*bottom*). Notice that the sequences are time-coherent and that moving objects are not ignored.

Fig. 9: Our method is also applicable to feature films. Here, we show three clips from the movie *Charade*. The first, third and fifth rows show several frames of input from one clip, and below each row of input is the 3D anaglyph image automatically generated by our algorithm (using only the same indoor training dataset as in the main paper).