DepthTransfer: Depth Extraction from Video Using Non-parametric Sampling

Kevin Karsch, *Student Member, IEEE,* Ce Liu, *Member, IEEE,* and Sing Bing Kang, *Fellow, IEEE*

Abstract—We describe a technique that automatically generates plausible depth maps from videos using non-parametric depth sampling. We demonstrate our technique in cases where past methods fail (non-translating cameras and dynamic scenes). Our technique is applicable to single images as well as videos. For videos, we use local motion cues to improve the inferred depth maps, while optical flow is used to ensure temporal depth consistency. For training and evaluation, we use a Kinect-based system to collect a large dataset containing stereoscopic videos with known depths. We show that our depth estimation technique outperforms the state-of-the-art on benchmark databases. Our technique can be used to automatically convert a monoscopic video into stereo for 3D visualization, and we demonstrate this through a variety of visually pleasing results for indoor and outdoor scenes, including results from the feature film *Charade*.

Index Terms—Depth estimation, monocular depth, motion estimation, data-driven, 2D-to-3D.

1 Introduction

Scene depth is useful for a variety of tasks, ranging from 3D modeling and visualization to robot navigation. It also facilitates spatial reasoning about objects in the scene, in the context of scene understanding. In the growing 3D movie industry, knowing the scene depth greatly simplifies the process of converting 2D movies to their stereoscopic counterparts. The problem we are tackling in this paper is: given an arbitrary 2D video, how can we automatically extract plausible depth maps at every frame? At a deeper level, we investigate how we can reasonably extract depth maps in cases where conventional structure-from-motion and motion stereo fail.

While many reconstruction techniques for extracting depth from video sequences exist, they typically assume moving cameras and static scenes. They do not work for dynamic scenes or for stationary, rotating, or variable focal length sequences. There are some exceptions, e.g., [1], which can handle some moving objects, but they still require camera motion to induce parallax and allow depth estimation.

In this paper, we present a novel solution to generate depth maps from ordinary 2D videos; our solution also applies to single images. This technique is applicable to arbitrary videos, and works in cases where conventional depth recovery methods fail (static/rotating camera; change in focal length; dynamic scenes). Our primary contribution is the use of a non-parametric "depth transfer" approach for inferring temporally consistent depth maps without imposing requirements on the video (Sec 3 and 4), including a method for improving the depth estimates of moving objects (Sec 4.1). In addition, we introduce a new, ground truth stereo RGBD (RGB+depth) video dataset[1] (Sec 5). We also describe how we synthesize stereo videos from ordinary 2D videos using the results of our technique (Sec 7).

2 Related work

In this section, we briefly survey techniques related to our work, namely 2D-to-3D conversion techniques (single image and video, automatic and manual) and non-parametric learning.

2.1 Single image depth estimation and 2D-to-3D

Early techniques for single image 2D-to-3D are semi-automatic; probably the most well-known of them is the "tour-into-picture" work of Horry et al. [2]. Here, the user interactively adds planes to the single image for virtual view manipulation. Two other representative examples of interactive 2D-to-3D conversion systems are those of Oh et al. [3] (where a painting metaphor is used to assign depths and extract layers) and Zhang et al. [4] (where the user adds surface normals, silhouettes, and creases as constraints for depth reconstruction).

One of the earliest automatic methods for single image 2D-to-3D was proposed by Hoiem et al. [5]. They created convincing-looking reconstructions of outdoor

- *Kevin Karsch is a graduate student in the Department of Computer Science, University of Illinois, Urbana, IL 61801. E-mail: karsch1@illinois.edu*
- *Ce Liu is with Microsoft Research New England, Cambridge, MA 02142. E-mail: celiu@microsoft.com*
- *Sing Bing Kang is with Microsoft Research, Redmond, WA 98052. E-mail: sbkang@microsoft.com*

[1] Our dataset and code are publicly available at http://kevinkarsch.com/depthtransfer

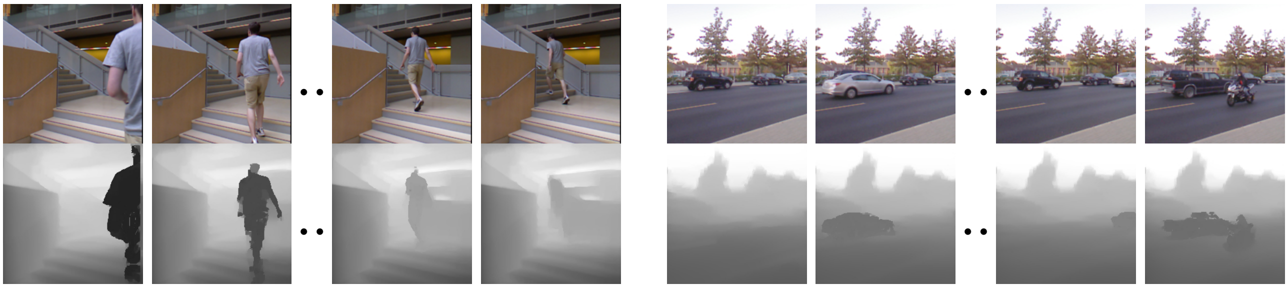


Fig. 1. Our technique takes a video sequence (*top row*) and automatically estimates per-pixel depth (*bottom row*). Our method does not require any cues from motion parallax or static scene elements; these videos were captured using a stationary camera with multiple moving objects.

images by assuming an image could be broken into a few planar surfaces; similarly, Delage et al. developed a Bayesian framework for reconstructing indoor scenes [6]. Saxena et al. devised a supervised learning strategy for predicting depth from a single image [7], which was further improved to create realistic reconstructions for general scenes [8], and efficient learning strategies have since been proposed [9]. Better depth estimates have been achieved by incorporating semantic labels [10], or more sophisticated models [11].

Apart from learning-based techniques for extracting depths, more conventional techniques have been used based on image content. For example, repetitive structures have been used for stereo reconstruction from a single image [12]. The dark channel prior has also proven effective for estimating depth from images containing haze [13]. In addition, single-image shape from shading is also possible for known (a priori) object classes [14], [15].

Compared to techniques here, we not only focus on depth from a *single image*, but also present a framework for using temporal information for enhanced and time-coherent depth when multiple frames are available.

The approach closest to ours is the contemporaneous work of Konrad et al. [16], [17], which also uses some form of non-parametric depth sampling to automatically convert monocular images into stereoscopic images. They make similar assumptions to ours (e.g. appearance and depth are correlated), but use a simpler optimization scheme, and argue that the use SIFT flow only provides marginal improvements (opposed to not warping candidates via SIFT flow). Our improvements are two-fold: their depth maps are computed using the median of the candidate disparity fields and smoothed with a cross bilateral filter, while we consider the candidate depths (and depth gradients) on a per-pixel basis. Furthermore, we propose novel solutions to incorporate temporal information from videos, whereas the method of Konrad et al. works on single images. We also show a favorable comparison in Table 2.

2.2 Video depth estimation and 2D-to-3D

A number of video 2D-to-3D techniques exist, but many of them are interactive. Examples of interactive system include those of Guttman et al. [18] (scribbles with depth properties are added to frames for video cube propagation), Ward et al. [19] ("depth templates" for primitive shapes are specified by the user, which the system propagates over the video), and Liao et al. [20] (user interaction to propagate structure-from-motion information with aid of optical flow).

There are a few commercial solutions that are automatic, e.g., Tri-Def DDD, but anecdotal tests using the demo version revealed room for improvement. There is even hardware available for real-time 2D-to-3D video conversion, such as the DA8223 chip by Dialog Semiconductor. It is not clear, however, how well the conversion works, since typically simple assumptions are made on what constitite foreground and background areas based on motion estimation. There are also a number of production houses specializing in 2D-to-3D conversions (e.g., In-Three [21] and Identity FX, Inc.), but their solutions are customized and likely to be manual-intensive. Furthermore, their tools are not publicly available.

If the video is mostly amenable to structure-from-motion and motion stereo, such technologies can be used to compute dense depth maps at every frame of the video. One system that does this is that of Zhang et al. [22], which also showed how depth-dependent effects such as synthetic fog and depth-of-focus can be achieved. While this system (and others [1], [23]) can handle some moving objects, there is significant reliance on camera motion that induces parallax.

2.3 Non-parametric learning

As the conventional notion of image correspondences was extended from different views of the same 3D scene to semantically similar, but different, 3D scenes [24], the information such as labels and motion can be transferred from a large database to parse an input image [25]. Given an unlabeled input image and a database with known per-pixel labels (e.g., sky, car, tree, window), their method works by transferring the labels from the database to the input image based on SIFT flow, which is estimated by matching pixel-wise, dense SIFT features. This simple methodology has been widely used in many computer vision applications such as image super resolution [26], image tagging [27] and object discovery [28].

We build on this work by transferring depth instead of semantic labels. Furthermore, we show that this "transfer" approach can be applied in a continuous optimization framework (Sec 3), whereas their method used a discrete MRFs.

3 Non-parametric depth estimation by continuous label transfer

We leverage recent work on non-parametric learning [29], which avoids explicitly defining a parametric model and requires fewer assumptions as in past methods (e.g., [7], [8], [10]). This approach also scales better with respect to the training data size, requiring virtually no training time. Our technique imposes no requirements on the video, such as motion parallax or sequence length, and can even be applied to a single image. We first describe our depth estimation technique as it applies to single images below, and in Sec 4 we discuss novel additions that allow for improved depth estimation in videos.

Our depth transfer approach, outlined in Fig 2, has three stages. First, given a database RGBD images, we find candidate images in the database that are "similar" to the input image in RGB space. Then, a warping procedure (SIFT Flow [24]) is applied to the candidate images and depths to align them with the input. Finally, an optimization procedure is used to interpolate and smooth the warped candidate depth values; this results in the inferred depth.

Our core idea is that scenes with similar semantics should have roughly similar depth distributions when densely aligned. In other words, images of semantically alike scenes are expected to have similar depth values in regions with similar appearance. Of course, not all of these estimates will be correct, which is why we find several candidate images and refine and interpolate these estimates using a global optimization technique that considers factors other than just absolute depth values.

3.1 RGBD database

Our system requires a database of RGBD images and/or videos. We have collected our own RGBD video dataset, as described in Sec 5; a few already exist online, though they are for single images only.[2]

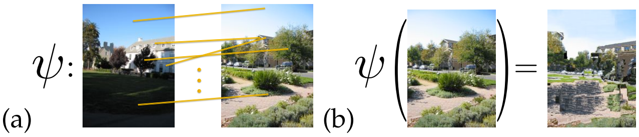


Fig. 3. SIFT flow warping. (a) SIFT features are calculated and matched in a one-to-many fashion, which defines ψ. (b) ψ is applied to achieve dense scene alignment.

3.2 Candidate matching and warping

Given a database and an input image, we compute high-level image features (we use GIST [30] and optical flow features) for each image or frame of video in the database as well as the input image. We then select the top K ($= 7$ in our work, unless explicitly stated) matching frames from the database, but ensure that each video in the database contributes no more than one matching frame. This forces matching images to be from differing viewpoints, allowing for greater variety among matches. We call these matching images *candidate images*, and their corresponding depths *candidate depths*.

Because the candidate images match the input closely in feature space, it is expected that the overall semantics of the scene are roughly similar. We also make the critical assumption that the distribution of depth is comparable among the input and candidates. However, we want pixel-to-pixel correspondences between the input and all candidates, as to limit the search space when inferring depth from the candidates.

We achieve this pixel-to-pixel correspondence through SIFT flow [24], which matches per-pixel SIFT features to estimate dense scene alignment. Using SIFT flow, we estimate warping functions $\psi_i, i \in \{1, \ldots, K\}$ for each candidate image; this process is illustrated in Fig 3. These warping functions map pixel locations from a given candidate's domain to pixel locations in the input's domain. The warping functions can be one-to-many.

3.3 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 [30] and features derived from optical flow (used only in the case of videos, as in Sec 4). To create flow features for a video, we first compute optical flow (using Liu's implementation [31]) 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

[2] Examples: Make3D range image dataset (http://make3d.cs.cornell.edu/data.html), B3DO dataset (http://kinectdata.com/), NYU depth datasets (http://cs.nyu.edu/~silberman/datasets/), RGB-D dataset (http://www.cs.washington.edu/rgbd-dataset/), and our own (http://kevinkarsch.com/depthtransfer).

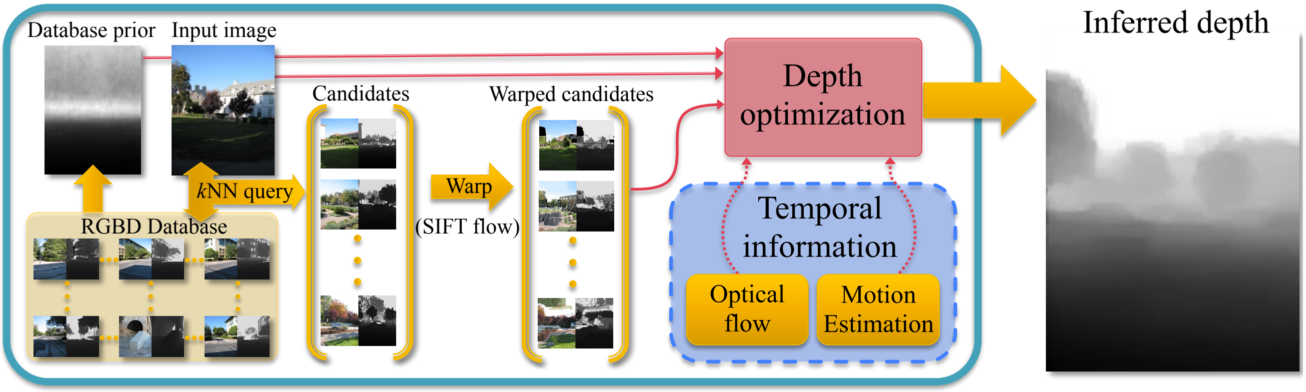


Fig. 2. Our pipeline for estimating depth. Given an input image, we find matching candidates in our database, and warp the candidates to match the structure of the input image. We then use a global optimization procedure to interpolate the warped candidates (Eq. 2), producing per-pixel depth estimates for the input image. With temporal information (e.g., extracted from a video), our algorithm can achieve more accurate, temporally coherent depth.

second moments as well (each component squared). This leads to eight 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.

3.4 Depth optimization

Each warped candidate depth is deemed to be a rough approximation of the input's depth map. Unfortunately, such candidate depths may still contain inaccuracies and are often not spatially smooth. Instead, we generate the most likely depth map by considering all of the warped candidates, optimizing with spatial regularization in mind.

Let $\mathbf{L}$ be the input image and $\mathbf{D}$ the depth map we wish to infer. We minimize

$$\begin{aligned} -\log(P(\mathbf{D}|\mathbf{L})) = E(\mathbf{D}) = \\ \sum_{i \in \text{pixels}} E_\text{t}(\mathbf{D}_i) + \alpha E_\text{s}(\mathbf{D}_i) + \beta E_\text{p}(\mathbf{D}_i) + \log(Z), \end{aligned} \tag{2}$$

where Z is the normalization constant of the probability, and α and β are parameters ($\alpha = 10, \beta = 0.5$). For a single image, our objective contains three terms: data (E_t), spatial smoothness (E_s), and database prior (E_p).

Data cost. The data term measures how close the inferred depth map $\mathbf{D}$ is to each of the warped candidate depths, $\psi_j(C^{(j)})$. This distance measure is defined by ϕ, a robust error norm (we use an approximation to the $L1$ norm, $\phi(x) = \sqrt{x^2 + \epsilon}$, with $\epsilon = 10^{-4}$). We define the data term as

$$\begin{aligned} E_\text{t}(\mathbf{D}_i) = \sum_{j=1}^{K} w_i^{(j)} \Big[\phi(\mathbf{D}_i - \psi_j(C_i^{(j)})) + \\ \gamma\big[\phi(\nabla_x \mathbf{D}_i - \psi_j(\nabla_x C_i^{(j)})) + \\ \phi(\nabla_y \mathbf{D}_i - \psi_j(\nabla_y C_i^{(j)}))\big]\Big], \end{aligned} \tag{3}$$

where $w_i^{(j)}$ is a confidence measure of the accuracy of the j^{th} candidate's warped depth at pixel i, and K ($= 7$) is the total number of candidates. We measure not only absolute differences, but also relative depth changes, i.e., depth gradients (∇_x, ∇_y are spatial derivatives). The latter terms of Eq 3 enforce similarity among candidate depth gradients and inferred depth gradients, weighted by γ ($= 10$).

Note that we warp the candidate's gradients in depth, rather than warping depth and then differentiating. Warping depth induces many new discontinuities, which would result in large gradients solely due to the warping, rather than actual depth discontinuities in the data. The downside of this is that the warped gradients are not integrable, but this does not signify as we optimize over depth anyhow (ensuring the resulting depth map is integrable).

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:

$$w_i^{(j)} = (1 + e^{(||\mathbf{S}_i - \psi_j(\mathcal{S}_i^{(j)})|| - \mu_s)/\sigma_s})^{-1}, \tag{4}$$

where $\mathbf{S}_i$ and $\mathcal{S}_i^{(j)}$ are the SIFT feature vectors at pixel i in candidate image j. We set $\mu_s = 0.5$ and $\sigma_s = 0.01$. Notice that the candidate image's SIFT features are computed first, and then warped using the warping function (ψ_j) calculated with SIFT flow.

Spatial smoothness. While we encourage spatial smoothness, we do not want the smoothness applied uniformly to the inferred depth, since there is typically some correlation 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 appearance-dependent smoothness 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 [32], [33].

The smoothness term is specified as:

$$E_{\text{s}}(\mathbf{D}_i) = s_{x,i}\phi(\nabla_x \mathbf{D}_i) + s_{y,i}\phi(\nabla_y \mathbf{D}_i). \tag{5}$$

The depth gradients along x and y ($\nabla_x \mathbf{D}, \nabla_y \mathbf{D}$) are modulated by soft thresholds (sigmoidal functions) of image gradients in the same directions ($\nabla_x \mathbf{L}, \nabla_y \mathbf{L}$), namely, $s_{x,i} = (1+e^{(||\nabla_x \mathbf{L}_i||-\mu_L)/\sigma_L})^{-1}$ and $s_{y,i} = (1+e^{(||\nabla_y \mathbf{L}_i||-\mu_L)/\sigma_L})^{-1}$. We set $\mu_L = 0.05$ and $\sigma_L = 0.01$.

Prior. We also incorporate assumptions from our database that will guide the inference when pixels have little or no influence from other terms (due to weights w and s):

$$E_{\text{p}}(\mathbf{D}_i) = \phi(\mathbf{D}_i - \mathcal{P}_i). \tag{6}$$

We compute the prior $\mathcal{P}$ by averaging all depth images in our database.

3.5 Numerical optimization details

Equation 2 requires an unconstrained, non-linear optimization, and we use iteratively reweighted least squares to minimize our objective function. 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); it is repeated until convergence.

As an example, consider a sub-portion of our objective, the second term in Eq 3 for candidate #1: $\sum_{i\in\text{pixels}} \phi(\nabla_x \mathbf{D}_i - \psi_1(\nabla_x C_i^{(1)})$. To minimize, we differentiate with respect to depth and set equal to zero (letting $b = [\psi_1(\nabla_x C_1^{(1)}),\ \dots,\ \psi_1(\nabla_x C_N^{(1)})]^T$, keeping in mind that $\phi(x) = \sqrt{x^2+\epsilon}$, and ∇_x is the horizontal derivative):

$$\sum_{i\in\text{pixels}} \frac{d}{d\mathbf{D}}\phi(\nabla_x \mathbf{D}_i - b)\ \nabla_x(\nabla_x \mathbf{D}_i - b) = 0. \tag{7}$$

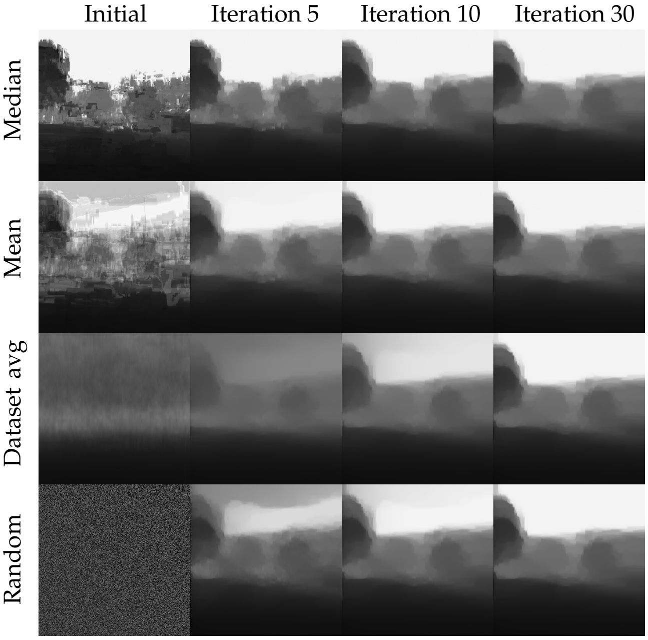


Fig. 4. Result of our optimization from different starting points (initialization methods on left). Our method typically converges to the same point given any initialization, but the median method (see text) is usually the most efficient. All depths are displayed at the same scale. The input image and depth can be seen in Fig 2.

We rewrite this using matrix notation as $G_x^T W(G_x \mathbf{D} - b) = 0$, where G_x is the $N \times N$ linear operator corresponding to a horizontal gradient (e.g., $[G_x \mathbf{D}]_i = \nabla_x \mathbf{D}_i$), and W is a diagonal matrix of "weights" computed from the non-linear portion of the derivative of ϕ. By fixing W (computed for a given $\mathbf{D}$), we arrive at the following IRLS solution:

$$\begin{aligned} W &= diag\left(\frac{d}{d\mathbf{D}}\phi(\nabla_x \mathbf{D}^{(t)} - b)\right) \\ \mathbf{D}^{(t+1)} &= (G_x^T W G_x)^{+}(G_x^T W b), \end{aligned} \tag{8}$$

where $()^{+}$ is the pseudoinverse and $D^{(t)}$ is the inferred depth at iteration t. We have found that thirty iterations typically approximates convergence. Extending IRLS to our full objective follows the same logic.[3]

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

[3] For further details and discussion of IRLS, see the appendix of Liu's thesis [31].

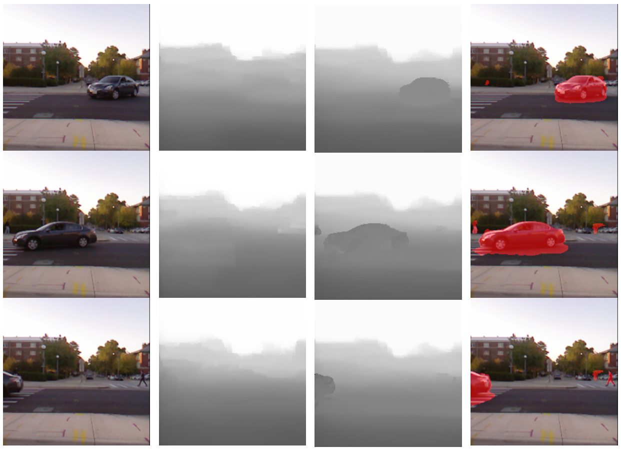


Fig. 5. Importance of temporal information. Left: input frames. Mid-left: predicted depth without temporal information. Note that the car is practically ignored here. Mid-right: predicted depth with temporal information, with the depth of the moving car recovered. Right: detected moving object.

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 (i.e., $\mathbf{D}_i^{(0)} = \text{median}\{\phi_1(C_i^{(1)}), \ \ldots, \ \phi_K(C_i^{(K)})\}$). The method of Konrad et al. [16], [17] also uses the median of retrieved depth maps, but this becomes their final depth estimate. Our approach uses the median only for initialization and allows any of the warped candidate depths to contribute and influence the final estimate (dictated by the objective function). Figure 4 shows the optimization process for different initializations.

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

4 Improved Depth Estimation for Video

Generating depth maps frame-by-frame without incorporating temporal information often leads to temporal discontinuities; past methods that ensure temporal coherence rely on a translating camera and static scene objects. Here, we present a framework that improves depth estimates and enforces temporal coherence for *arbitrary video sequences*. That is, our algorithm is suitable for videos with moving scene objects and rotating/zooming views where conventional SfM and stereo techniques fail. (Here, we assume that zooming induces little or no parallax.)

Our idea is to incorporate temporal information through additional terms in the optimization that ensure (a) depth estimates are consistent over time and (b) that moving objects have depth similar to their contact point with the ground. Each frame is processed the same as in the single image case (candidate matching and warping), except that now we employ a global optimization (described below) that infers depth for the entire sequence at once, incorporating temporal information from all frames in the video. Fig 5 illustrates the importance of these additional terms in our optimization.

We formulate the objective for handling video by adding two terms to the single-image objective function:

$$E_{\text{video}}(\mathbf{D}) = E(\mathbf{D}) + \sum_{i \in \text{pixels}} \nu E_{\text{c}}(\mathbf{D}_i) + \eta E_{\text{m}}(\mathbf{D}_i), \quad (9)$$

where E_{c} encourages temporal coherence while E_{m} uses motion cues to improve the depth of moving objects. The weights ν and η balance the relative influence of each term ($\nu = 100, \eta = 5$).

We model temporal coherence first by computing per-pixel optical flow for each pair of consecutive frames in the video (using Liu's publicly available code [31]). We define the *flow difference*, ∇_{flow}, as a linear operator which returns the change in the flow across two corresponding pixels, and model the coherence term as

$$E_{\text{c}}(\mathbf{D}_i) = s_{t,i}\phi(\nabla_{flow}\mathbf{D}_i). \quad (10)$$

We weight each term by a measure of flow confidence, $s_{t,i} = (1+e^{-(||\nabla_{flow}\mathbf{L}_i||-\mu_L)/\sigma_L})^{-1}$, which intuitively is a soft threshold on the reprojection error ($\mu_L = 0.05$, $\sigma_L = 0.01$). Minimizing the weighted flow differences has the effect of temporally smoothing inferred depth in regions where optical flow estimates are accurate.

To handle motion, we detect moving objects in the video (Sec 4.1) and constrain their depth such that these objects touch the floor. Let m be the binary motion segmentation mask and $\mathcal{M}$ the depth in which connected components in the segmentation mask contact the floor. We define the motion term as

$$E_{\text{m}}(\mathbf{D}_i) = m_i\phi(\mathbf{D}_i - \mathcal{M}_i). \quad (11)$$

4.1 Detecting moving objects

Differentiating moving and stationary objects in the scene can be a useful cue when estimating depth. Here we describe our algorithm for detecting objects in motion in non-translating movies (i.e., static, rotational, and variable focal length videos)[4]. Note that there are many existing techniques for detecting moving objects (e.g., [34], [35]); we use what we consider to be easy

[4] In all other types of videos (e.g. those with parallax or fast moving objects/pose), we do not employ this algorithm; equivalently we set the motion segmentation weight to zero ($\eta = 0$).

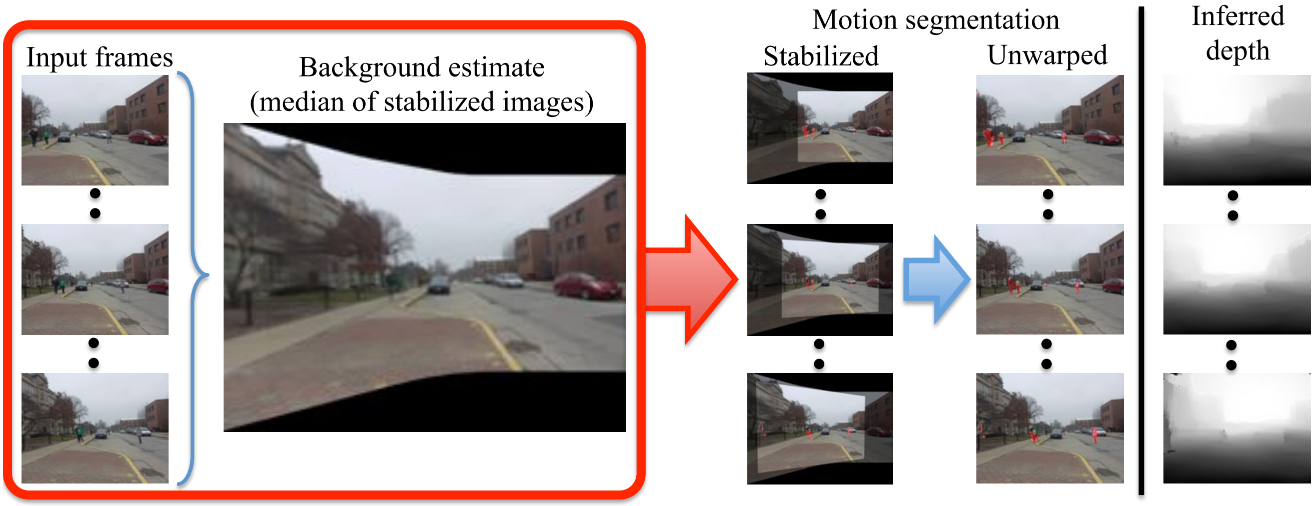


Fig. 6. Example of our motion segmentation applied to a rotating sequence. We first estimate homographies to stabilize the video frames and create a clean background image using a temporal median filter. We then evaluate our estimate of motion thresholding metric on the stabilized sequences, and unwarp 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.

to implement and effective for our purpose of depth extraction.

First, to account for dynamic exposure changes throughout the video, we find the image with the lowest overall intensity in the sequence and perform histogram equalization on all other frames in the video. We use this image as to not propagate spurious noise found in brighter images. Next, we use RANSAC on point correspondences to compute the dominant camera motion (modeled using homography) to align neighboring frames in the video. Median filtering is then used on the stabilized images to extract the background B (ideally, without all the moving objects).

In our method, the likelihood of a pixel being in motion depends on how different it is from the background, weighted by the optical flow magnitude which is computed between stabilized frames (rather than between the original frames). We use relative differencing (relative to background pixels) to reduce reliance on absolute intensity values, and then threshold to produce a mask:

$$m_{i,k} = \begin{cases} 1 & \text{if } ||flow_{i,k}||\frac{||W_{i,k}-B_i||^2}{B_i} > \tau \\ 0 & \text{otherwise,} \end{cases} \qquad (12)$$

where $\tau = 0.01$ is the threshold, and $W_{i,k}$ is the k^{th} pixel of the i^{th} stabilized frame (i.e., warped according to the homography that aligns W with B). This produces a motion estimate in the background's coordinate system, so we apply the corresponding inverse homography to each warped frame to find the motion relative to each frame of the video. This segmentation mask is used (as in Eq 11) to improve depth estimates for moving objects in our optimization. Fig 6 illustrates this technique.

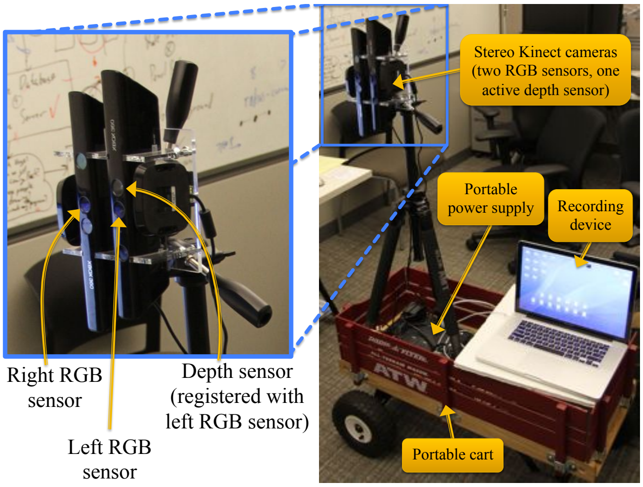

Stereo Kinect cameras (two RGB sensors, one active depth sensor)
Portable power supply
Recording device
Portable cart
Right RGB sensor
Left RGB sensor
Depth sensor (registered with left RGB sensor)

Fig. 7. Our stereo-RGBD collection rig consists of two side-by-side Microsoft Kinects. The rig is mobile through the use of an uninterruptible power supply, laptop, and rolling mount.

5 MSR-V3D DATASET

In order to train and test our technique on image/video input, we collected a dataset containing stereo video clips for a variety of scenes and viewpoints (116 indoor, 61 outdoor). The dataset primarily contains videos recorded from a static viewpoint with moving objects in the scene (people, cars, etc.). There are also 100 indoor frames (single images) in addition to the 116 indoor video clips within the database (stereo RGB, depth for the left frame). These sequences come from four different buildings in two cities and contain substantial scene variation (e.g., hallways, rooms, foyers). Each clip is filmed with camera viewpoints that are either static or slowly rotated. We have entitled our dataset the Microsoft Research Stereo Video + Depth (MSR-V3D), and it is available online at http://kevinkarsch.com/depthtransfer.

We captured the MSR-V3D dataset with two side-by-side, vertically mounted Microsoft Kinects shown in Fig 7 (positioned about 5cm apart). We collected the color images from both Kinects and only the depth map from the left Kinect. For each indoor clip, the left stereo view also contains view-aligned depth from the Kinect. Due to IR interference, depth for the right view was not captured indoors, and depth was totally disregarded outdoors due to limitations of the Kinect.

We also collected outdoor data with our stereo device. However, because the Kinect cannot produce depth maps outdoors due to IR interference from the sunlight, we could not use these sequences for training. We attempted to extract ground truth disparity between stereo pairs, but the quality/resolution of Kinect images were too low to get decent estimates. We did, however, use this data for testing and evaluation purposes.

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 to estimate depth for the input (left) sequence.

Kinect depth is also susceptible to "holes" in the data caused typically by surface properties, disoccluded/interfered IR pattern, or because objects are simply too 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.

6 Experiments

In this section, we show results of experiments involving single image and video depth extraction. We also discuss the importance of scale associated with training and the effect of the number of candidates used for depth hypothesis.

6.1 Single image results

We evaluate our technique on two single image RGBD datasets: the Make3D Range Image Dataset [8], and the NYU Depth Dataset [36].

6.1.1 Make3D Range Image Dataset

Of the 534 images in the Make3D dataset, we use 400 for testing and 134 for training (the same as was done before, e.g., [7], [8], [11], [10]). We report error for three common metrics in Table 1. Denoting $\mathbf{D}$ as estimated depth and $\mathbf{D}^*$ as ground truth depth, we compute *relative* (**rel**) *error* $\frac{|\mathbf{D}-\mathbf{D}^*|}{\mathbf{D}^*}$, $\mathbf{log_{10}}$ *error* $|\log_{10}(\mathbf{D}) - \log_{10}(\mathbf{D}^*)|$, and *root mean squared* (**RMS**) *error* $\sqrt{\sum_{i=1}^{N}(\mathbf{D}_i - \mathbf{D}_i^*)^2/N}$. Error measures are averaged over all pixels/images in the test set. Our estimated depth maps are computed at 345×460 pixels (maintaining the aspect ratio of the Make3D dataset input images).

Method	**rel**	$\mathbf{log_{10}}$	**RMS**
Depth MRF [7]	0.530	0.198	16.7
Make3D [8]	0.370	0.187	-
Feedback Cascades [11]	-	-	15.2
Semantic Labels [10]	0.375	**0.148**	-
Depth Transfer (ours)	**0.361**	**0.148**	**15.1**

TABLE 1
Comparison of depth estimation errors on the Make3D range image dataset. Using our single image technique, our method achieves state of the art results in each metric (**rel** is relative error, **RMS** is root mean squared error; details in text).

Our method is as good as or better than the state-of-the-art for each metric. Note that previously no method achieved state-of-the-art results in more than one metric. We show several examples in Fig 8. Thin structures (e.g., trees and pillars) are usually recovered well; however, fine structures are occasionally missed due to spatial regularization (such as the poles in the bottom-right image of Fig 8).

6.1.2 NYU Depth Dataset

We report additional results for the NYU Depth Dataset [36], which consists of 1449 indoor RGBD images captured with a Kinect. Holes from the Kinect are disregarded during training (candidate searching and warping), and are not included in our error analysis.

Quantitative results are shown in Table 2, and Fig 9 shows qualitative results. For comparison, we train our algorithm in two different ways and report results on each. One method trains (e.g., selects RGBD candidates) from the NYU depth dataset (holding out the particular example that is being tested), and the other method trains using all RGBD images found in MSR-V3D. We observe a significant degradation in results when training using our own dataset (MSR-V3D), likely because our dataset contains many fewer scenes than the NYU dataset, and a less diverse set of examples (NYU contains home, office, and many unique interiors, a total of 464; ours is primarily office-type scenes from four different buildings). This also suggests that generalization to interior scenes is much more difficult than outdoor, which coincides with the

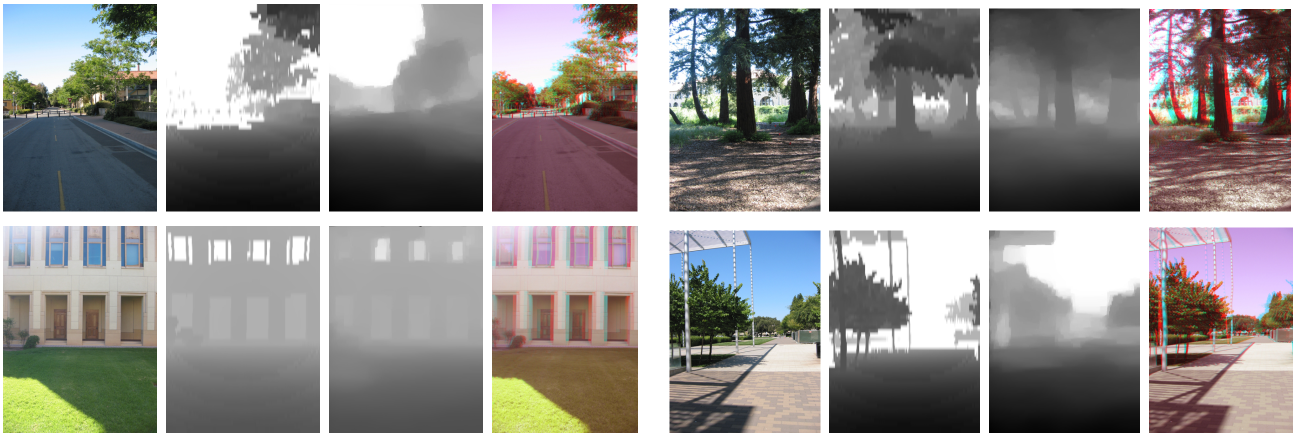


Fig. 8. Single image results obtained on test images from the Make3D dataset. Each result contains the following four images (from left to right): original photograph, ground truth depth from the dataset, our inferred depth, and our synthesized anaglyph image. The depth images are shown in log scale. 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.

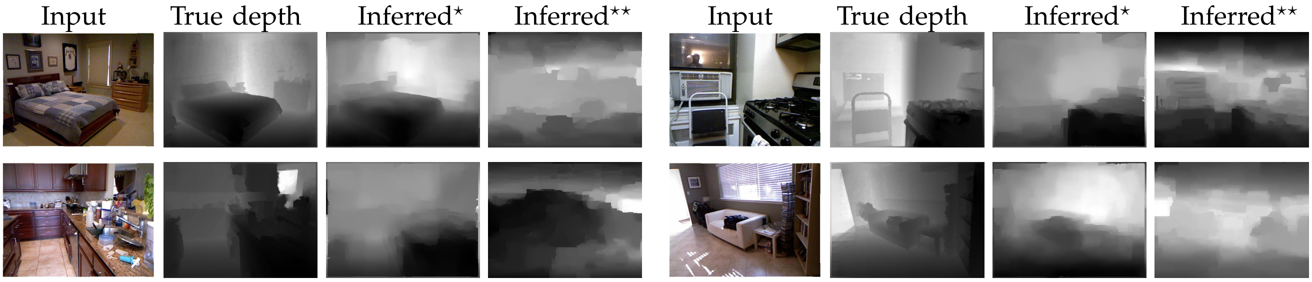


Fig. 9. Single image results obtained on the NYU Depth Dataset. Each result contains the following four images (from left to right): original photograph, ground truth depth from the dataset, our inferred depth trained on the NYU depth dataset (*), holding out the particular image), and our inferred depth trained on our MSR-V3D dataset (**). The top left result is in the fifth percentile (in terms of $\log_{10}$ error), the top right is in the bottom fifth percentile, and both bottom results are near the median. Notice how the NYU-trained results are much better; this is likely due to the high variation of indoor images (MSR-V3D images appear very dissimilar to the NYU ones), and is also reflected in the quantitive results (Tab 1). Holes in the true depth maps have been filled using the algorithm in [36], and each pair of ground truth and inferred depths are displayed at the same scale.

Method	rel	$\log_{10}$	RMS
Depth Transfer $^{(NYU)}$	**0.350**	**0.131**	**1.2**
Depth Transfer $^{(MSR-V3D)}$	0.806	0.231	3.7
Depth Fusion [16]	0.368	0.135	1.3
Depth Fusion (no warp) [17]	0.371	0.137	1.3
NYU average depth†	0.491	0.327	4.3
NYU per-pixel average††	0.561	0.164	20.1

TABLE 2
Quantitative evaluation of our method and the methods of Konrad et al. on the NYU Depth Dataset. All methods are trained on the NYU dataset using a hold-one-out scheme, except for Depth Transfer $^{(MSR-V3D)}$ which is trained using our own dataset (containing significantly different indoor scenes). Each method uses seven candidate images ($K = 7$). It is interesting that the per-pixel average (††) performs worse than a single depth value (†), evidencing that these metrics are likely not very perceptual.

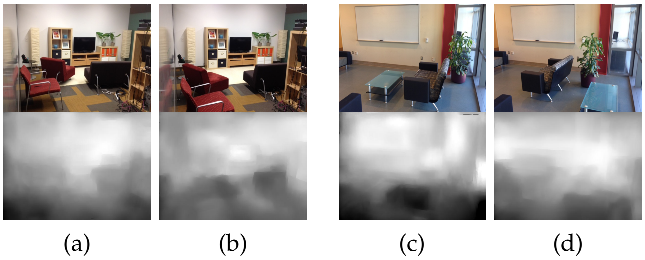


Fig. 10. Demonstration of our method on scenes where objects have been repositioned. The chairs and couch in (a) are moved closer in (b), and furniture in (c) are repositioned in (d). Although the depth maps contain noticeable errors, it is evident that our estimates are influenced by factors other than appearance. For example, although the chairs and couch in (a-b) have the same appearance, the estimated depth of (b) is noticeably closer in the chair/couch regions, as it should be.

intuition that indoor scenes, on average, contain more variety than outdoor scenes.

Additionally, we compare our method to the single-image approaches of Konrad et al. [16], [17]. Both of these methods choose candidate RGBD images using global image features (as in our work), but dissimilar from our optimization, they compute depth as the median value of the candidate depth images (per-pixel), and post-process their depths with a cross-bilateral filter. [16] applies SIFT flow to warp/align depth (similar to our method), while [17] opts not to for efficiency (their main focus is 2D-to-3D conversion rather than depth estimation). We show improvements over both of these methods, but more importantly, our depth and 2D-to-3D conversion methods apply to videos rather than only single images.

Finally, we compare all of these results to two baselines: the average depth value over the entire NYU dataset (†), and the per-pixel average of the NYU depth (††). We observe that our method significantly outperforms these baselines in three metrics (relative, $\log_{10}$, and RMS error) when trained on the NYU dataset (hold-one-out). Our error rates increase by training on our dataset (MSR-V3D), but this training method still outperforms the baselines in several metrics. Note that these baselines do use some knowledge from the NYU dataset, unlike our method trained on MSR-V3D.

One concern of our method may be its reliance on appearance. In Fig 10, we demonstrate that although appearance partially drives our depth estimates, other factors such as spatial relationships, scale, and orientation necessarily contribute. For example, photographing the same scene with objects in different configurations will lead to different depth estimates, as would be expected.

6.2 Video results

Our technique works well for videos of many types scenes and video types (Figs 1, 5, 6, 11, 12, 13). We use the dataset we collected in Sec 5 to validate our method for videos (we know of no other existing methods/datasets to compare to). This dataset contains ground truth depth and stereo image sequences for four different buildings (referred to as Buildings 1, 2, 3, and 4), and to gauge our algorithm's ability to generalize, *we only use data from Building 1 for training*. We still generate results for Building 1 by holding each particular example out of the training set during inference.

We show quantitative results in Table 3 and qualitative results in Fig 12. We calculate error using the same metrics as in our single image experiments, and to make these results comparable with Table 1, we globally rescale the ground truth and inferred depths to match the range of the Make3D database (roughly 1-81m). As expected, the results from Building 1 are the best, but our method still achieves reasonable errors for the other buildings as well.

Dataset	rel	$\log_{10}$	RMS	PSNR
Building 1$^\diamond$	0.196	0.082	8.271	15.6
Building 2	0.394	0.135	11.7	15.6
Building 3	0.325	0.159	15.0	15.0
Building 4	0.251	0.136	15.3	16.4
Outdoors$^{\diamond\diamond}$	-	-	-	15.2
All	0.291	0.128	12.6	15.6

TABLE 3
Error averaged over our stereo-RGBD dataset. $^\diamond$Building used for training (results for Building 1 trained using a hold-one-out scheme). $^{\diamond\diamond}$No ground truth depth available.

Fig 12 shows a result from each building in our dataset (top left is Building 1). As the quantitative results suggest, our algorithm performs very well for this building. In the remaining examples, we show results of videos captured in the other three buildings, all which contain vastly different colors, surfaces and structures from the Building 1. Notice that even for these images our algorithm works well, as evidenced by the estimated depth and 3D images.

We demonstrate our method on videos containing parallax in Fig 13. Such videos can be processed with structure from motion (SfM) and multi-view stereo algorithms; e.g. the dense depth estimation method of Zhang et al. [23]. We visually compare our method to the method of Zhang et al., as well as a version of our method where the depth prior comes from a sparse SfM point cloud. Specifically, we solve for camera pose, track and triangulate features using publicly available code from Zhang et al. Then, we project the triangulated features into each view and compute their depth; this sparse depth map is used as the prior term in Eq 6, and for videos with parallax, we increase the prior weight ($\beta = 10^3$) and turn off motion segmentation ($\eta = 0$). We note that in most cases of parallax, a multi-view stereo algorithm is preferable to our solution, yet in some cases our method seems to perform better qualitatively.

As further evaluation, a qualitative comparison between our technique and the publicly available version of Make3D is shown in Fig 14. Unlike Make3D, our technique is able to extract the depth of the runner throughout the sequence, in part because Make3D does not incorporate temporal information.

Our algorithm also does not require video training data to produce video results. We can make use of static RGBD images (e.g., Make3D dataset) to train our algorithm for video input, and we show several outdoor video results in Figs 1, 5, 6, and 13. Even with static data from another location, our algorithm is usually able to infer accurate depth and stereo views.

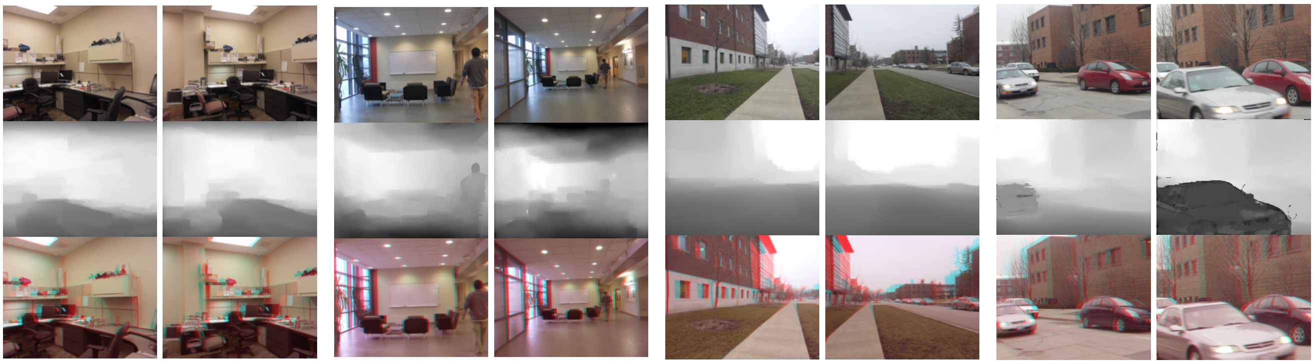


Fig. 11. Results obtained on four different sequences captured with a rotating camera and/or variable focal length. We show the input frames (*top*), inferred depth (*middle*) and inferred 3D anaglyph (*bottom*). Notice that the sequences are time-coherent and that moving objects are not ignored.

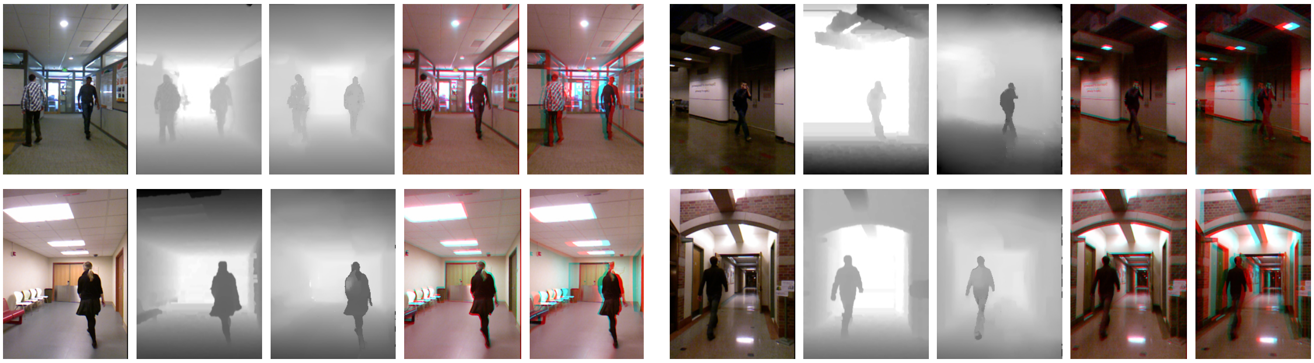


Fig. 12. Video results obtained on test images for each building in our stereo-RGBD dataset (buildings 1-4, from left to right and top to bottom). For each result (from left to right): original photograph, ground truth depth, our inferred depth, ground truth anaglyph image, and our synthesized anaglyph image. Because the ground truth 3D images were 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. Note also that our algorithm can handle multiple moving objects (*top*).

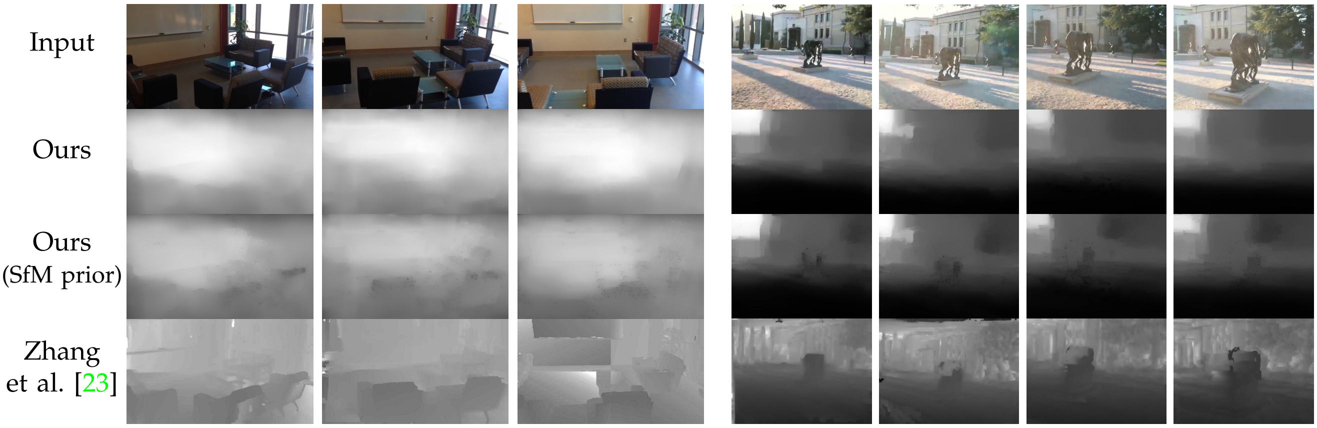


Fig. 13. Comparison of our method on videos containing parallax. For each input, we show the depth estimated using our method with no modification, and the depth estimated when our method is bootstrapped with SfM (e.g. a sparse depth map, calculated using SfM, is used as the prior). We also compare these results to the dense depth estimates of Zhang et al. [23], whose method works only for videos with parallax. When SfM estimates are poor (left sequence), multi-view stereo methods may perform worse than our method, but can be quite good given decent SfM estimates and mostly Lambertian scenes (right sequence). Overall, SfM seems to help estimate relative depth near boundaries, whereas our method seems to better estimate global structure.

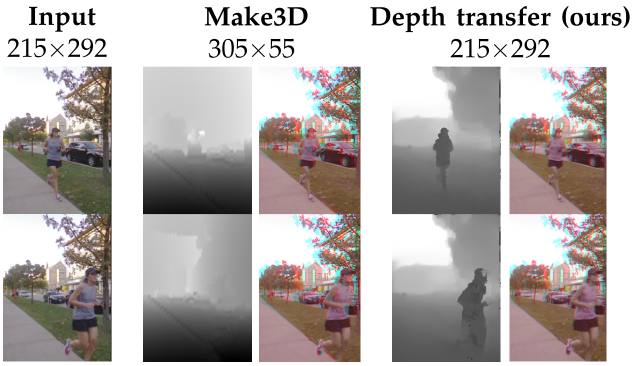


Fig. 14. Comparison between our technique and the publicly available version of Make3D (http://make3d.cs.cornell.edu/code.html). Make3D depth inference is trained to produce depths of resolution 55×305 (bilinearly resampled for visualization), and we show results of our algorithm at the input native resolution. The anaglyph images are produced using the technique in Sec 7. Depths displayed at same scale.

Training set $(N = 64)$	rel	$\log_{10}$	RMS
GIST candidates $(K = 7)$	0.431	0.164	15.9
Full dataset $(K = 64)$	0.475	0.207	20.3

TABLE 4
Importance of training set size. We select a subset of 64 images from the Make3D dataset, and compare our method using only 7 candidate images (selected using GIST features) versus using the entire dataset (64 candidates). The results suggest that selectively re-training based on the target image's content can significantly improve results.

Since we collected ground truth stereo images in our dataset, we also compare our synthesized right view (from our 2D-to-3D algorithm, see Sec. 7) with the actual right view. We use peak signal-to-noise ratio (PSNR) to measure the quality of the reconstructed views, as shown in Table 3. We could not acquire depth outdoors, and we use this metric to compare our outdoor and indoor results.

6.3 Importance of training scale

One implicit hypothesis our algorithm makes is that *training with only a few "similar" images is better than training with a large set of arbitrary images*. This is encoded by our nearest neighbor candidate search: we choose k similar images (in our work, based on GIST features), and use *only* these images to sample/transfer depth from. Conversely, Saxena et al. [8] train a parametric model for predicting depth using their entire dataset (Make3D); Liu et al. [10] found that training a different models for each semantic classes (tree, sky, etc.) improves results. The results in Table 4 are good evidence that training on only similar *scenes* improves results (rather than only similar semantic classes as in [10], and on the entire dataset [8]).

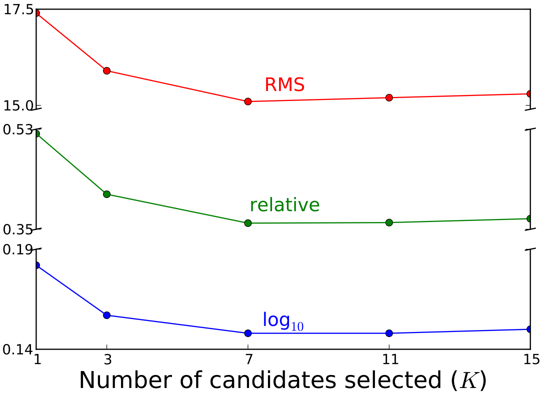


Fig. 15. Effect of the number of chosen candidates (K). Errors are reported on the Make3D dataset with varying values of K. For this dataset, $K = 7$ is optimal, but increasing K beyond 7 does not significantly degrade results.

We verify this further with another experiment: we created a sub-dataset by randomly choosing 64 images from the Make3D dataset, then we compared the results of our method using only similar images for training (7 nearest neighbors) and the results of our method trained using the entire dataset (64 nearest neighbors) during inference. As the results in Table 4 suggest, training using only similar scenes greatly improves quantitative results. In addition, since our "training" is simply a nearest neighbor query (which can be done quickly in seconds), it can be orders of magnitude faster than retraining parametric models.

6.4 Effect of K (number of candidates)

We also evaluate how our technique behaves given different values of K, i.e., how many candidate images are selected prior to inference. On the Make3D dataset, we evaluate three error metrics (same as above: relative, $\log_{10}$, and RMS) averaged over the entire dataset using different values of K. Fig 15 shows the results, and we see that $K = 7$ is optimal for this dataset, but we still achieve comparable results with $K \geq 7$.

Empirically, we find that K acts as a smoothing parameter. This fits with the intuition that more candidate images will likely lead to diversity in the candidate set, and since the inferred depth is in some sense sampled from all candidates, the result will be smoother as K increases.

7 Application: 2D-to-3D

In recent years, 3D[5] videos have become increasingly popular. Many feature films are now available in 3D,

[5] The presentation of stereoscopic (left+right) video to convey the sense of depth.

and increasingly more personal photography devices are now equipped with stereo capabilities (from point-and-shoots to attachments for video cameras and SLRs). Distributing user-generated content is also becoming easier. Youtube has recently incorporated 3D viewing and uploading features, and many software packages have utilities for handling and viewing 3D file formats, e.g., Fujifilm's FinePixViewer.

As 3D movies and 3D viewing technology become more widespread, it is desirable to have techniques that can convert legacy 2D movies to 3D in an efficient and inexpensive way. Currently, the movie industry uses expensive solutions that tend to be manual-intensive. For example, it was reported that the cost of converting (at most) 20 minutes of footage for the movie "Superman Returns" was $10 million[6].

Our technique can be used to automatically generate the depth maps necessary to produce the stereoscopic video (by warping each input frame using its corresponding depth map). To avoid generating holes at disocclusions in the view synthesis step, we adapt and extend Wang et al.'s technique [37]. They 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. Their method was applied only to single images; we extend this method to handle video sequences as well.

7.1 Automatic stereoscopic view synthesis

After estimating depth for a video sequence (or a single image), we perform depth image-based rendering (DIBR) to synthesize a new view for stereoscopic display. A typical strategy for DIBR is 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., [38], [39], [40], [41]), but visual artifacts still remain in the case of general scenes.

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. [37] 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 16. 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 in the following text.

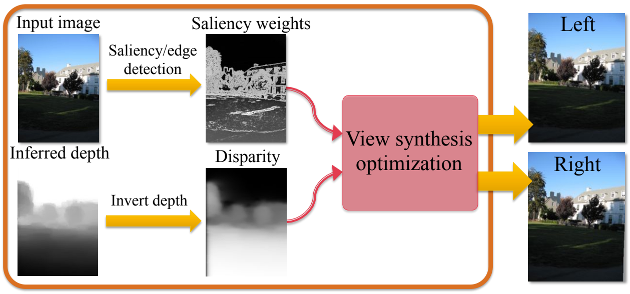


Fig. 16. Summary of the view synthesis procedure for a single image, as described by Wang et al. [37]. Given an image and corresponding depth, we compute salient regions and disparity, and compute stereoscopic images by warping the input image. We extend this method to handle videos, as in equation 14.

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:

$$\mathbf{W}_0 = \frac{\mathbf{W}_{\text{max}}}{\mathbf{D} + \epsilon}, \tag{13}$$

where $\mathbf{W}_0 = \{W_1, \dots, W_n\}$, $\mathbf{D} = \{D_1, \dots, D_n\}$ is the initial disparity and depth (resp) for each of the n frames of the input, and $\mathbf{W}_{\text{max}}$ 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:

$$\begin{aligned} Q(\mathbf{W}_i) &= \sum_{i \in \text{pixels}} Q_{\text{data}}(\mathbf{W}_i) + Q_{\text{smooth}}(\mathbf{W}_i), \\ Q_{\text{data}}(\mathbf{W}_i) &= l_i(\mathbf{W}_i - \mathbf{W}_{0,i})^2, \\ Q_{\text{smooth}}(\mathbf{W}_i) &= \lambda(s_{x,i}||\nabla_x \mathbf{W}_i||^2 + s_{y,i}||\nabla_y \mathbf{W}_i||^2) + \\ &\quad \mu s_{t,i}||\nabla_{flow} \mathbf{W}_i||^2, \end{aligned} \tag{14}$$

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 $l_i = \frac{W_{0,i}}{W_{\text{max}}} + (1 + e^{-(||\nabla \mathbf{L}_i|| - 0.01)/0.002})^{-1}$. 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 Eq 4, we set $s_{x,i} = (1 + e^{(||\nabla_x \mathbf{L}_i|| - 0.05)/.01})^{-1}$, $s_{y,i} = (1 + e^{(||\nabla_y \mathbf{L}_i|| - \mu_L)/\sigma_L})^{-1}$, and $s_{t,i} = (1 + e^{-(||\nabla_{flow} \mathbf{L}_i|| - \mu_L)/\sigma_L})^{-1}$, 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

[6] See http://en.wikipedia.org/wiki/Superman_Returns.

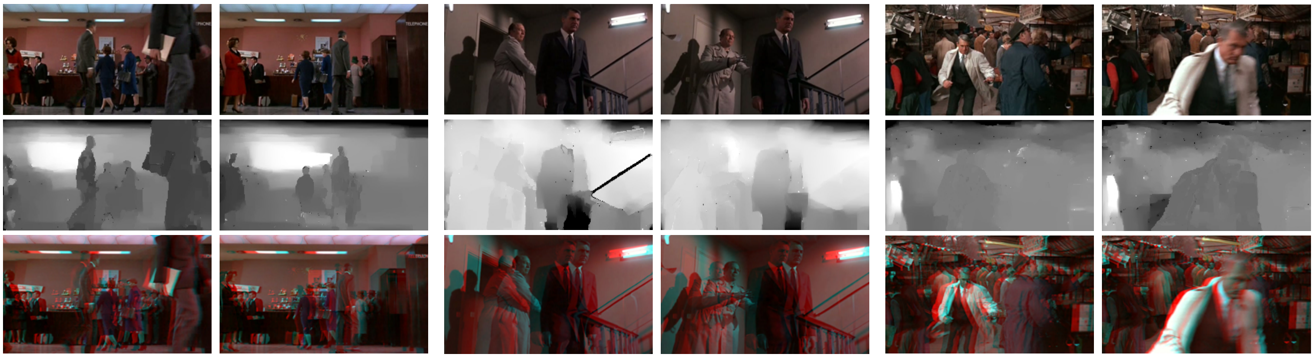


Fig. 17. Several clips from the feature film *Charade*. Each result contains (from top to bottom): the original frames, estimated depth, and estimated anaglyph automatically generated by our algorithm. Some imperfections in depth are conveniently masked in the 3D image due to textureless or less salient regions.

flow direction). We set $\mu_L = 0.05$ and $\sigma_L = 0.01$. 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 [37]. For rendering, we use the anisotropic pixel splatting method described by Wang et al. [37], 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 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 [42].

7.2 2D-to-3D Results

Our estimated depth is good enough to generate compelling 3D images, and representative results are shown in (Figs 11, 12). We also demonstrate that our algorithm may be suitable for feature films in Fig 17. More diverse quantities of training are required to achieve commercial-quality conversion; however, even with a small amount of data, we can generate plausible depth maps and create convincing 3D sequences automatically.

Recently, Youtube has released an automatic 2D-to-3D conversion tool, and we compared our method to theirs on several test sequences. Empirically, we noticed that the Youtube results have a much more subtle 3D effect. Both results are available online at http://kevinkarsch.com/depthtransfer.

Our algorithm takes roughly one minute per 640×480 frame (on average) using a parallel implementation on a quad-core 3.2GHz processor.

8 Discussion

Our results show that our depth transfer algorithm works for a large variety of indoor and outdoor sequences using a practical amount of training data. Note that our algorithm works for arbitrary videos, not just those with no parallax. However, videos with arbitrary camera motion and static scenes are best handled with techniques such as [1]. In Fig 18, we show that our algorithm requires some similar data in order to produce decent results (i.e., training with outdoor images for an indoor query is likely to fail). However, our algorithm can robustly handle large amounts of depth data with little degradation of output quality. The only issue is that more data requires more comparisons in the candidate search.

This robustness is likely due to the features we use when determining candidate images as well as the design of our objective function. In Fig 19, we show an example query image, the candidates retrieved by our algorithm, and their contribution to the inferred depth. By matching GIST features, we detect candidates that contain features consistent with the query image, such as building facades, sky, shrubbery, and similar horizon location. Notice that the depth of the building facade in the input comes mostly from another similarly oriented building facade (teal), and the ground plane and shrubbery depth come almost solely from other candidates' ground and tree depths.

In some cases, our motion segmentation misses or falsely identifies moving pixels. This can result in inaccurate depth and 3D estimation, although our spatio-temporal regularization (Eqs. 5, 10) helps to

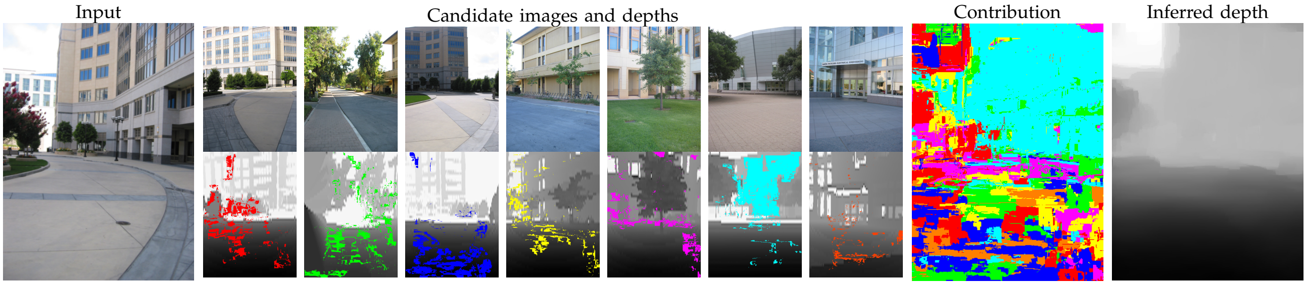


Fig. 19. Candidate contribution for depth estimation. For an input image (*left*), we find top-matching candidate RGBD images (*middle*), and infer depth (*right*) for the input using our technique. The contribution image is color-coded to show the sources; red pixels indicate that the left-most candidate influenced the inferred depth image the most, orange indicates contribution from the right-most candidate, etc.

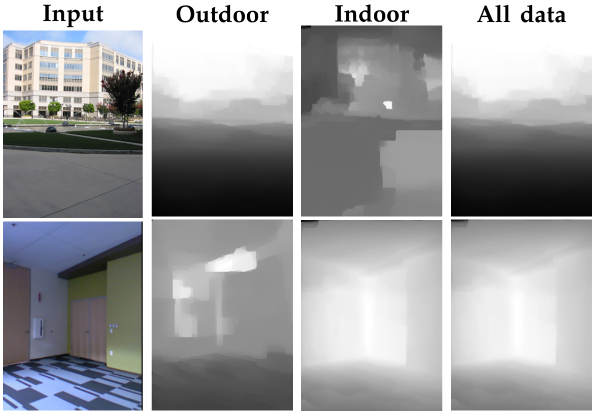


Fig. 18. Effect of using different training data for indoor and outdoor images. While the results are best if the proper dataset is used, we also get good results even if we combine all of the datasets.

overcome this. Our algorithm also assumes that moving objects contact the ground, and thus may fail for airborne objects (see Fig 20).

Due to the serial nature of our method (depth estimation followed by view synthesis), our method is prone to propagating errors through the stages. For example, if an error is made during depth estimation, the result may be visually implausible. It would be ideal to use knowledge of how the synthesized views should look in order to correct issues in depth.

9 Concluding Remarks

We have demonstrated a fully automatic technique to estimate depths for videos. Our method is applicable in cases where other methods fail, such as those based on motion parallax and structure from motion, and works even for single images and dynamics scenes. Our depth estimation technique is novel in that we use a non-parametric approach, which gives qualitatively good results, and our single-image algorithm quantitatively outperforms existing methods. Using our technique, we also show how we can generate stereoscopic videos for 3D viewing from conventional 2D videos. Specifically, we show how to generate time-coherent, visually pleasing stereo sequences using our inferred depth maps. Our method is suitable as a good starting point for converting legacy 2D feature films into 3D.

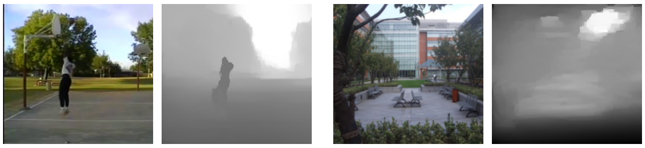


Fig. 20. Example failure cases. Left: thin or floating objects (pole and basketball) are ignored. Right: input image is too different from training set.

Acknowledgement

We would like to thank Tom Blank for his critical help in creating our dual-Kinect data collection system.

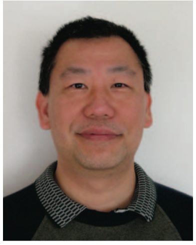
References

[1] G. Zhang, J. Jia, W. Hua, and H. Bao, "Robust bilayer segmentation and motion/depth estimation with a handheld camera," *IEEE TPAMI*, vol. 33, no. 3, pp. 603–617, 2011.
[2] Y. Horry, K. Anjyo, and K. Arai, "Tour into the picture: Using a spidery mesh interface to make animation from a single image," *SIGGRAPH*, 1997.
[3] B. Oh, M. Chen, J. Dorsey, and F. Durand, "Image-based modeling and photo editing," *SIGGRAPH*, 2001.
[4] L. Zhang, G. Dugas-Phocion, J.-S. Samson, and S. Seitz, "Single view modeling of free-form scenes," *Journal of Visualization and Computer Animation*, vol. 13, no. 4, pp. 225–235, 2002.
[5] D. Hoiem, A. Efros, and M. Hebert, "Automatic photo pop-up," in *ACM SIGGRAPH*, 2005.
[6] E. Delage, H. Lee, and A. Ng, "A dynamic Bayesian network model for autonomous 3D reconstruction from a single indoor image," in *CVPR*, 2006.
[7] A. Saxena, S. Chung, and A. Ng, "Learning depth from single monocular images," *NIPS*, 2005.
[8] A. Saxena, M. Sun, and A. Ng, "Make3D: Learning 3D scene structure from a single still image," *IEEE TPAMI*, vol. 31, no. 5, pp. 824–840, 2009.
[9] D. Batra and A. Saxena, "Learning the right model: Efficient max-margin learning in laplacian crfs," in *CVPR*, 2012.
[10] B. Liu, S. Gould, and D. Koller, "Single image depth estimation from predicted semantic labels," in *CVPR*, 2010.
[11] C. Li, A. Kowdle, A. Saxena, and T. Chen, "Towards holistic scene understanding: Feedback enabled cascaded classification models," *NIPS*, 2010.
[12] C. Wu, J.-M. Frahm, and M. Pollefeys, "Repetition-based dense single-view reconstruction," *CVPR*, 2011.

[13] K. He, J. Sun, and X. Tang, "Single image haze removal using dark channel prior," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 33, no. 12, pp. 2341–2353, Dec 2011.
[14] F. Han and S.-C. Zhu, "Bayesian reconstruction of 3D shapes and scenes from a single image," in *IEEE HLK*, 2003.
[15] T. Hassner and R. Basri, "Example based 3D reconstruction from single 2D images," *CVPR Workshop on Beyond Patches*, 2006.
[16] J. Konrad, G. Brown, M. Wang, P. Ishwar, C. Wu, and D. Mukherjee, "Automatic 2d-to-3d image conversion using 3d examples from the internet," in *SPIE Stereoscopic Displays and Applications*, vol. 8288, January 2012.
[17] J. Konrad, M. Wang, and P. Ishwar, "2d-to-3d image conversion by learning depth from examples," in *3DCINE*, 2012.
[18] M. Guttmann, L. Wolf, and D. Cohen-Or, "Semi-automatic stereo extraction from video footage," in *ICCV*, 2009.
[19] B. Ward, S. B. Kang, and E. P. Bennett, "Depth Director: A system for adding depth to movies," *IEEE Comput. Graph. Appl.*, vol. 31, no. 1, pp. 36–48, 2011.
[20] M. Liao, J. Gao, R. Yang, and M. Gong, "Video stereolization: Combining motion analysis with user interaction," *IEEE TVCG*, vol. 18, no. 7, pp. 1079–1088, Jul. 2012.
[21] A. Van Pernis and M. DeJohn, "Dimensionalization: Converting 2D films to 3D," in *SPIE Stereoscopic Displays and Applications XIX*, vol. 6803, 2008.
[22] G. Zhang, Z. Dong, J. Jia, L. Wan, T.-T. Wong, and H. Bao, "Refilming with depth-inferred videos," *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, vol. 15, no. 5, p. 828?40, 2009.
[23] G. Zhang, J. Jia, T.-T. Wong, and H. Bao, "Consistent depth maps recovery from a video sequence," *IEEE TPAMI*, vol. 31, pp. 974–988, 2009.
[24] C. Liu, J. Yuen, and A. Torralba, "SIFT Flow: Dense correspondence across scenes and its applications," *IEEE TPAMI*, vol. 33, no. 5, pp. 978–994, May 2011.
[25] ——, " Nonparametric Scene Parsing via Label Transfer," *IEEE TPAMI*, vol. 33, no. 12, pp. 2368–2382, December 2011.
[26] M. Tappen and C. Liu, "A bayesian approach to alignment-based image hallucination," in *ECCV*, 2012.
[27] M. Rubinstein, C. Liu, and W. Freeman, "Annotation propagation: Automatic annotation of large image databases via dense image correspondence," in *ECCV*, 2012.
[28] M. Rubinstein, A. Joulin, J. Kopf, and C. Liu, "Unsupervised joint object discovery and segmentation in internet images," in *CVPR*, 2013.
[29] C. Liu, J. Yuen, and A. Torralba, "Nonparametric scene parsing: Label transfer via dense scene alignment," *CVPR*, 2009.
[30] A. Oliva and A. Torralba, "Modeling the shape of the scene: A holistic representation of the spatial envelope," *IJCV*, vol. 42, pp. 145–175, 2001.
[31] C. Liu, "Beyond pixels: Exploring new representations and applications for motion analysis," Ph.D. dissertation, MIT, 2009.
[32] D. Hoiem, A. Stein, A. Efros, and M. Hebert, "Recovering occlusion boundaries from a single image," in *ICCV*, 2007.
[33] M. Maire, P. Arbelaez, C. Fowlkes, and J. Malik, "Using contours to detect and localize junctions in natural images," in *CVPR*, 2008.
[34] M. Heikkila and M. Pietikainen, "A texture-based method for modeling the background and detecting moving objects," *IEEE TPAMI*, vol. 28, no. 4, pp. 657–662, April 2006.
[35] Y. Sheikh, O. Javed, and T. Kanade, "Background subtraction for freely moving cameras," in *ICCV*, 2009.
[36] P. K. Nathan Silberman, Derek Hoiem and R. Fergus, "Indoor segmentation and support inference from rgbd images," in *ECCV*, 2012.
[37] O. Wang, M. Lang, M. Frei, A. Hornung, A. Smolic, and M. Gross, "StereoBrush: Interactive 2D to 3D conversion using discontinuous warps," in *SBIM*, 2011.
[38] A. Colombari, A. Fusiello, and V. Murino, "Continuous parallax adjustment for 3D-TV," *IEEE Eur. Conf. Vis. Media Prod.*, pp. 194–200, Nov. 2005.
[39] R. Klein Gunnewiek, R.-P. Berretty, B. Barenbrug, and J. Magalhães, "Coherent spatial and temporal occlusion generation," in *Proc. SPIE, Stereoscopic Displays and Applications XX*, vol. 7237, 2009.
[40] K. Luo, D. Li, Y. Feng, and Z. M., "Depth-aided inpainting for disocclusion restoration of multi-view images using depth-image-based rendering," *J. Zhejiang Univ. Sci. A*, vol. 10, no. 12, pp. 1738–1749, Dec. 2009.
[41] L. Zhang, C. Vazquez, and S. Knorr, "3D-TV content creation: Automatic 2D-to-3D video conversion," *IEEE Trans. on Broadcasting*, vol. 57, no. 2, pp. 372–383, June 2011.
[42] S. Koppal, C. Zitnick, M. Cohen, S. Kang, B. Ressler, and A. Colburn, "A viewer-centric editor for 3D movies," *IEEE Computer Graphics and Applications*, vol. 31, pp. 20–35, 2011.

Kevin Karsch received a BS degree in computer science and mathematics from the University of Missouri in 2009, and is currently working towards his PhD in computer science at the University of Illinois. He holds the NSFGRF and NDSEG fellowship, and was recently awarded the Lemelson-MIT student prize for innovation. His research interests include computer vision, graphics, and computational photography.

Ce Liu received the BS degree in automation and the ME degree in pattern recognition from the Department of Automation, Tsinghua University in 1999 and 2002, respectively. After receiving the PhD degree from the Massachusetts Institute of Technology in 2009, he now holds a researcher position at Microsoft Research New England. From 2002 to 2003, he worked at Microsoft Research Asia as an assistant researcher. His research interests include computer vision, computer graphics, and machine learning. He has published more than 20 papers in the top conferences and journals in these fields. He received a Microsoft Fellowship in 2005, the Outstanding Student Paper award at the Advances in Neural Information Processing Systems (NIPS) in 2006, a Xerox Fellowship in 2007, and the Best Student Paper award at the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2009. He is a member of the IEEE.



Sing Bing Kang is a principal researcher at Microsoft Research. He received his Ph.D. in robotics from Carnegie Mellon University, Pittsburgh in 1994. His areas of interest are computer vision and computer graphics, more specifically image-based modeling as well as image and video enhancement. Sing Bing has co-edited two books ("Panoramic Vision" and "Emerging Topics in Computer Vision") and co-authored two books ("Image-Based Rendering" and "Image-Based Modeling of Plants and Trees"). He has served as area chair and member of technical committee for the major computer vision conferences (ICCV, CVPR, ECCV). In addition, he has served as papers committee member for SIGGRAPH and SIGGRAPH Asia. Sing Bing was program chair for ACCV 2007 and CVPR 2009, and is currently Associate Editor-in-Chief for IEEE Transactions on Pattern Recognition and Machine Intelligence. He is an IEEE Fellow.