Learning-Based, Automatic 2D-to-3D Image and Video Conversion

Janusz Konrad, *Fellow, IEEE*, Meng Wang, Prakash Ishwar, *Senior Member, IEEE*, Chen Wu, and Debargha Mukherjee

***Abstract*— Despite a significant growth in the last few years, the availability of 3D content is still dwarfed by that of its 2D counterpart. To close this gap, many 2D-to-3D image and video conversion methods have been proposed. Methods involving human operators have been most successful but also time-consuming and costly. Automatic methods, which typically make use of a *deterministic* 3D scene model, have not yet achieved the same level of quality for they rely on assumptions that are often violated in practice. In this paper, we propose a new class of methods that are based on the radically different approach of *learning* the 2D-to-3D conversion from examples. We develop two types of methods. The first is based on learning a point mapping from *local* image/video attributes, such as color, spatial position, and, in the case of video, motion at each pixel, to scene-depth at that pixel using a regression type idea. The second method is based on *globally* estimating the entire depth map of a query image directly from a repository of 3D images (image+depth pairs or stereopairs) using a nearest-neighbor regression type idea. We demonstrate both the efficacy and the computational efficiency of our methods on numerous 2D images and discuss their drawbacks and benefits. Although far from perfect, our results demonstrate that repositories of 3D content can be used for effective 2D-to-3D image conversion. An extension to video is immediate by enforcing temporal continuity of computed depth maps.**

***Index Terms*— 3D images, stereoscopic images, image conversion, nearest neighbor classification, cross-bilateral filtering.**

I. Introduction

THE availability of 3D-capable hardware today, such as TVs, Blu-Ray players, gaming consoles, and smartphones, is not yet matched by 3D content production. Although constantly growing in numbers, 3D movies are still an exception rather than a rule, and 3D broadcasting (mostly sports) is still minuscule compared to 2D broadcasting. The gap between 3D hardware and 3D content availability is likely to close in the future, but today there exists an urgent need to convert the existing 2D content to 3D.

A typical 2D-to-3D conversion process consists of two steps: depth estimation for a given 2D image and depth-based rendering of a new image in order to form a stereopair. While the rendering step is well understood and algorithms exist that produce good quality images, the challenge is in estimating depth from a single image (video). Therefore, throughout this paper the focus is on depth recovery and not on depth-based rendering, although we will briefly discuss our approach to this problem later.

There are two basic approaches to 2D-to-3D conversion: one that requires a human operator's intervention and one that does not. In the former case, the so-called semi-automatic methods have been proposed where a skilled operator assigns depth to various parts of an image or video. Based on this sparse depth assignment, a computer algorithm estimates dense depth over the entire image or video sequence. The involvement of a human operator may vary from just a few scribbles to assign depth to various locations in an image to a precise delineation of objects and subsequent depth assignment to the delineated regions.

In the case of automatic methods, no operator intervention is needed and a computer algorithm automatically estimates the depth for a single image (or video). To this effect, methods have been developed that estimate shape from shading, structure from motion or depth from defocus. Although such methods have been shown to work in some restricted scenarios they do not work well for arbitrary scenes. In an attempt to equip 3D TVs, Blu-Ray players and gaming consoles with real-time automatic 2D-to-3D conversion, consumer electronics manufacturers have developed simpler techniques that rely on various heuristic assumptions but such methods fail on more challenging scenes. Recently, machine-learning-inspired methods have been proposed to automatically estimate the depth map of a single monocular image by applying image parsing. Although restricted to architectural scenes, these methods opened a new direction for 2D-to-3D conversion. We will review both semi-automatic and automatic methods in Section II in detail.

The methods we propose in this paper, carry the "big data" philosophy of machine learning. In consequence, they apply to arbitrary scenes and require no manual annotation. Our data-driven approach to 2D-to-3D conversion has been inspired by the recent trend to use large image databases for various computer vision tasks, such as object recognition [18] and image saliency detection [19]. In particular, we propose a new class of methods that are based on the radically different approach of *learning* the 2D-to-3D conversion from examples.

Manuscript received October 18, 2012; revised March 16, 2013 and June 3, 2013; accepted June 3, 2013. Date of publication June 20, 2013; date of current version August 1, 2013. This work was supported in part by the U.S. National Science Foundation (NSF) under Award CCF-0905541. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Anthony Vetro.

J. Konrad and P. Ishwar are with the Department of Electrical and Computer Engineering, Boston University, Boston, MA 02215 USA (e-mail: jkonrad@bu.edu; pi@bu.edu).

M. Wang, C. Wu, and D. Mukherjee are with Google Inc., Mountain View, CA 94043 USA (e-mail: wangmeng1985@google.com; chenwu@google.com; debargha@google.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIP.2013.2270375

We develop two types of methods. The first one is based on learning a point mapping from *local* image/video attributes, such as color, spatial position, and motion at each pixel, to scene-depth at that pixel using a regression type idea. The second one is based on *globally* estimating the entire depth map of a query image directly from a repository of 3D images (image + depth pairs or stereopairs) using a nearest-neighbor regression type idea. Early versions of our learning-based approach to 2D-to-3D image conversion, either suffered from high computational complexity [8] or were tested on only a single dataset [9]. Here, we introduce the local method and evaluate the qualitative performance and the computational efficiency of both the local and global methods against those of the Make3D algorithm [14] and a recent method proposed by Karsch *et al.* [7]. We demonstrate the improved quality of the depth maps produced by our global method relative to state-of-the-art methods together with up to 4 orders of magnitude reduction in computational effort. We also discuss weaknesses of both proposed methods.

The paper is organized as follows. In Section II, we review the state of the art in 2D-to-3D image conversion. In Section III, we describe the conversion based on local point transformation and in Section IV we provide details of the global approach to the conversion. In Section V, we show numerous experimental results and we conclude the paper in Section VI.

II. State of the Art

There are two types of 2D-to-3D image conversion methods: semi-automatic methods, that require human operator intervention, and automatic methods, that require no such help.

A. Semi-Automatic Methods

To date, this has been the more successful approach to 2D-to-3D conversion. In fact, methods that require a significant operator intervention in the conversion process, such as delineating objects in individual frames, placing them at suitable depths, and correcting errors after final rendering, have been successfully used commercially by such companies as Imax Corp., Digital Domain Productions Inc. (formerly In-Three Inc.), etc. Many films have been converted to 3D using this approach.

In order to reduce operator involvement in the process and, therefore, lower the cost while speeding up the conversion, research effort has recently focused on the most labor-intensive steps of the manual involvement, namely spatial depth assignment. Guttmann *et al.* [6] have proposed a dense depth recovery *via* diffusion from sparse depth assigned by the operator. In the first step, the operator assigns relative depth to image patches in some frames by scribbling. In the second step, a combination of depth diffusion, that accounts for local image saliency and local motion, and depth classification is applied. In the final step, disparity is computed from the depth field and two novel views are generated by applying half of the disparity amplitude. The focus of the method proposed by Agnot *et al.* [1] is the application of cross-bilateral filtering to an initial depth map. The authors propose to use a library of initial depth maps (smooth maps consistent with the 3D perspective of outdoor scenes or rooms) from which an operator can choose one that best corresponds to the image being converted. They also suggest estimation of the initial depth map based on image blur but show only one very simple example; this initialization is unlikely to work well in more complex cases. Phan *et al.* [12] propose a simplified and more efficient version of the Guttmann *et al.* [6] method using scale-space random walks that they solve with the help of graph cuts. Liao *et al.* [10] further simplify operator involvement by first computing optical flow, then applying structure-from-motion estimation and finally extracting moving object boundaries. The role of an operator is to correct errors in the automatically computed depth of moving objects and assign depth in undefined areas.

B. Automatic Methods

The problem of depth estimation from a single 2D image, which is the main step in 2D-to-3D conversion, can be formulated in various ways, for example as a shape-from-shading problem [20]. However, this problem is severely under-constrained; quality depth estimates can be found only for special cases. Other methods, often called multi-view stereo, attempt to recover depth by estimating scene geometry from multiple images not taken simultaneously. For example, a moving camera permits structure-from-motion estimation [17] while a fixed camera with varying focal length permits depth-from-defocus estimation [16]. Both are examples of the use of multiple images of the same scene captured at different times or under different exposure conditions (e.g., all images of the Statue of Liberty). Although such methods are similar in spirit to the methods proposed here, the main difference is that while these methods use images known to depict the same scene as the query image, we use all images available in a large repository and automatically select suitable ones for depth recovery.

Several electronics manufacturers have developed real-time 2D-to-3D converters that rely on stronger assumptions and simpler processing than the methods discussed above, e.g., faster-moving or larger objects are assumed to be closer to the viewer, higher frequency of texture is assumed to belong to objects located further away, etc. Although such methods may work well in specific scenarios, in general it is very difficult, if not impossible, to construct heuristic assumptions that cover all possible background and foreground combinations. Such real-time methods have been implemented in Blu-Ray 3D players by LG, Samsung, Sony and others. DDD offers its TriDef 3D software for PCs, TVs and mobile devices. However, these are proprietary systems and no information is available about the assumptions used.

Recently, machine-learning-inspired techniques employing image parsing have been used to estimate the depth map of a single monocular image [11], [14]. Such methods have the potential to automatically generate depth maps, but currently work only on few types of images (mostly architectural scenes) using carefully-selected training data (precise, laser-scanned depth estimates or manually-annotated semantic depth classes). In the quest to develop data-driven approaches to

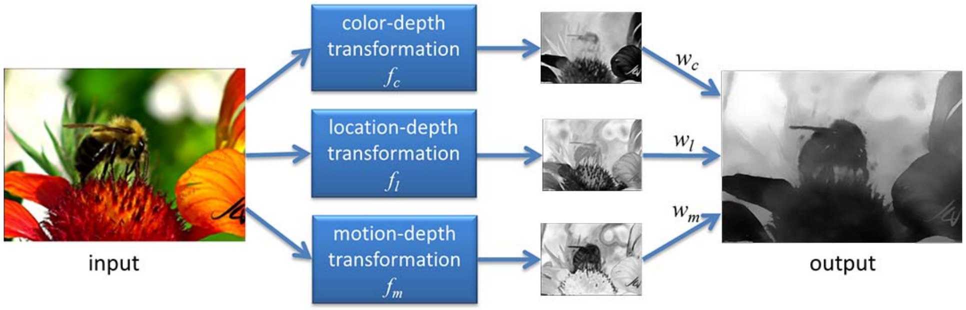

color-depth transformation f_c
location-depth transformation f_l
motion-depth transformation f_m
w_c
w_l
w_m
input
output

Fig. 1. Example of depth estimation from color, spatial location and motion. Black indicates smallest depth.

2D-to-3D conversion we have also been inspired by the recent trend to use large image databases for various computer vision tasks, such as object recognition [18] and image saliency detection [19]. In our first attempt, we developed a method that fuses SIFT-aligned depth maps selected from a large 3D database, however this approach proved to be computationally demanding [8]. Subsequently, we skipped the costly SIFT-based depth alignment and used a different metric (based on histogram of gradients) for selecting most similar depth fields from a database. We observed no significant quality degradation but a significant reduction of the computational complexity [9]. Very recently, Karsch *et al.* [7] have proposed a depth extraction method based on SIFT warping that essentially follows our initial, unnecessarily complex, approach to depth extraction [8].

III. 2D-to-3D Conversion by Learning a Local Point Transformation

The first class of conversion methods we are presenting is based on learning a point transformation that relates local low-level image or video attributes at a pixel to scene-depth at that pixel. Once the point transformation is learned, it is applied to a monocular image, i.e., depth is assigned to a pixel based on its attributes. This is in contrast to methods described in Section IV where the entire depth map of a query is estimated directly from a repository of 3D images (image+depth pairs or stereopairs) using a nearest-neighbor regression type idea.

A pivotal element in this approach is a point transformation used to compute depth from image attributes. This transformation can be estimated either by training on a ground-truth dataset, the approach we take in this paper, or defined heuristically. Let $\mathcal{I} = \{(\vec{I}^1, d^1), (\vec{I}^2, d^2), ..., (\vec{I}^K, d^K)\}$ denote a training dataset composed of K pairs $(\vec{I}^k, d^k)$, where $\vec{I}^k$ is a color image (usually in YUV format) and d^k is the corresponding depth field. We assume that all images and depth fields have the same spatial dimensions. Such a dataset can be constructed in various ways. One example is the Make3D dataset [13], [14], [21] that consists of images and depth fields captured outdoors by a laser range finder. Another example is the NYU Kinect dataset [15], [22] containing over 100 k images and depth fields captured indoors using a Kinect camera.

Examples of low-level video attributes that can be leveraged to compute relative depth of a pixel include color, spatial location, and local motion. Although the dependence of depth on pixel color, location and motion may seem counter intuitive, it is enough to observe that a bluish color is often associated with a distant sky, the bottom of a picture usually depicts ground close to the camera and a moving object stays in front of the background. These three attributes are in fact used in the YouTube 2D-to-3D conversion service recently released on-line.

Given a training set $\mathcal{I}$ consisting of K image-depth pairs, one can, in principle, learn a general regression function that maps a tuple of local features such as (color,location,motion) to a depth value, i.e.,

$$f : (\text{color}, \text{location}, \text{motion}) \longrightarrow \text{depth}.$$

However, to ensure low run-time memory and processing costs, we learn a more restricted form of transformation:

$$f[\text{color}, \mathbf{x}, \text{motion}] = w_c f_c[\text{color}] + w_l f_l[\mathbf{x}] + w_m f_m[\text{motion}].$$

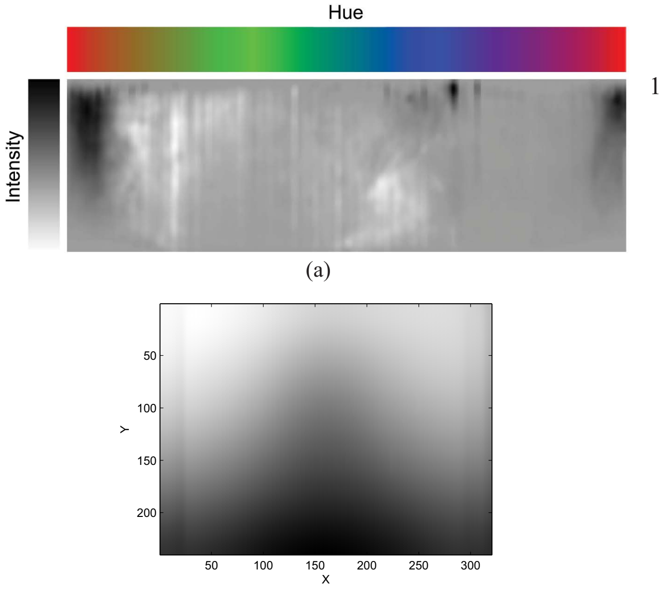
We now discuss how the individual color-depth, location-depth, and motion-depth transformations as well as the weights are learned.

Fig. 1 shows a sample video frame with depth maps estimated from color, location and motion cues separately, as well as the final combined depth map. In order to obtain a color-depth transformation f_c, we first transform the YUV space, commonly used in compressed images and videos, to the HSV color space. We found out that the saturation component (S) provides little depth discrimination capacity and therefore we limit the transformation attributes to hue (H) and value (V). Let $[H^k[\mathbf{x}], S^k[\mathbf{x}], V^k[\mathbf{x}]]^T$ be the HSV components of a pixel at spatial location $\mathbf{x}$ quantized to L levels. The depth mapping $f_c[h, v], h, v = 1, ..., L$ is computed as the average of depths at all pixels in $\mathcal{I}$ with hue h and value v:

$$f_c[h, v] = \frac{\sum_{k=1}^{K} \sum_{\mathbf{x}} 1(H^k[\mathbf{x}] = h, V^k[\mathbf{x}] = v) d^k[\mathbf{x}]}{\sum_{k=1}^{K} \sum_{\mathbf{x}} 1(H^k[\mathbf{x}] = h, V^k[\mathbf{x}] = v)} \quad (1)$$

where $1(A)$ is the indicator function which equals one if A is true and equals zero otherwise.

Fig. 2(a) shows the transformation f_c computed from a dataset $\mathcal{I}$ of, mostly, outdoor scenes. Note a large dark patch around reddish colors indicating that red elements of a scene are located closer to the camera. A large bright patch around bright-bluish colors is indicative of a far-away sky. The bright patch around yellow-orange colors is more difficult to classify but may be due to the distant sun as many videos have been captured outdoors.



(a)

(b)

Fig. 2. (a) Color-depth transformation $f_c[h, v]$ obtained from hue h and value (intensity) v, and (b) location-depth transformation $f_l[\mathbf{x}]$ obtained from pixel located at $\mathbf{x} = [x, y]^T$, both computed from dataset $\mathcal{I}$ of, mostly, outdoor scenes. Black indicates smallest depth. The hue above ranges from red on left through orange, green, cyan, blue and purple to red, again, on right. It is best to view this figure in a PDF version of the paper.

The location-depth transformation f_l is simply the average depth computed from all depth maps in $\mathcal{I}$ at the same location:

$$f_l[\mathbf{x}] = \frac{1}{K}\sum_{k=1}^{K} d^k[\mathbf{x}]. \tag{2}$$

Fig. 2(b) shows an example of the above transformation computed from the same dataset $\mathcal{I}$. It is clear that locations in the center and towards the bottom of the image usually correspond to objects that are closer to the camera.

In addition to color and spatial attributes, video sequences may contain motion attributes relevant to depth recovery. In this case, local motion between consecutive video frames is of interest. The underlying assumption in the motion-depth transformation is that moving objects are closer to the viewer than the background. In order to estimate the motion-depth transformation f_m, the basic idea is to first compute local motion between consecutive video frames, then extract a moving object mask from this motion, and, finally, assign a distinct depth (smaller than that of the background) to this mask. This brings the moving objects closer to the viewer. The estimation of local motion may be accomplished by any optical flow method, e.g., [2], but may also require global motion compensation, e.g., [5], in order to account for camera movements. A simple thresholding of the magnitude of local motion produces a moving object's mask. However, since such masks are often noisy some form of smoothing may be needed. Cross-bilateral filtering [4] controlled by the luminance of the video frame, in which the estimated local motion is anchored, usually suffices.

In the final step, the local transformation outputs are linearly combined to produce the final depth field (Fig. 1). While the location-depth weight w_l may be kept constant, the motion-depth weight w_m can be adjusted in proportion to the number of pixels deemed moving in the image being converted. Therefore, the color-depth weight w_c equals $1 - w_l - w_m$. Assuming that the image to be converted to 3D is the left image of a fictitious stereopair, the right image is rendered from the left image and the inferred depth field. An example of such rendering is discussed in Section IV-D.

IV. 2D-to-3D Conversion Based on Global Nearest-Neighbor Depth Learning

While 2D-to-3D conversion based on learning a local point transformation has the undisputed advantage of computational efficiency – the point transformation can be learned off-line and applied basically in real time – the same transformation is applied to images with potentially different global 3D scene structure. This is because this type of conversion, although learning-based, is based on purely *local* image/video attributes, such as color, spatial position, and motion at each pixel. To address this limitation, in this section we develop a second method that estimates the *global* depth map of a query image or video frame directly from a repository of 3D images (image+depth pairs or stereopairs) using a nearest-neighbor regression type idea.

The approach we propose here is built upon a key observation and an assumption. The key observation is that among millions of 3D images available on-line, there likely exist many whose 3D content matches that of a 2D input (query) we wish to convert to 3D. We are also making an assumption that two images that are photometrically similar also have similar 3D structure (depth). This is not unreasonable since photometric properties are often correlated with 3D content (depth, disparity). For example, edges in a depth map almost always coincide with photometric edges.

Given a monocular query image Q, assumed to be the left image of a stereopair that we wish to compute, we rely on the above observation and assumption to "learn" the entire depth field from a repository of 3D images $\mathcal{I}$ and render a stereopair in the following steps:

1) **search for representative depth fields**: find k 3D images in the repository $\mathcal{I}$ that have most similar depth to the query image, for example by performing a k nearest-neighbor (kNN) search using a metric based on photometric properties,
2) **depth fusion**: combine the k representative depth fields, for example, by means of median filtering across depth fields,
3) **depth smoothing**: process the fused depth field to remove spurious variations, while preserving depth discontinuities, for example, by means of cross-bilateral filtering,
4) **stereo rendering**: generate the right image of a fictitious stereopair using the monocular query image and the smoothed depth field followed by suitable processing of occlusions and newly-exposed areas.

Specific details of these steps depend on the type of 3D images contained in the repository. The above steps apply

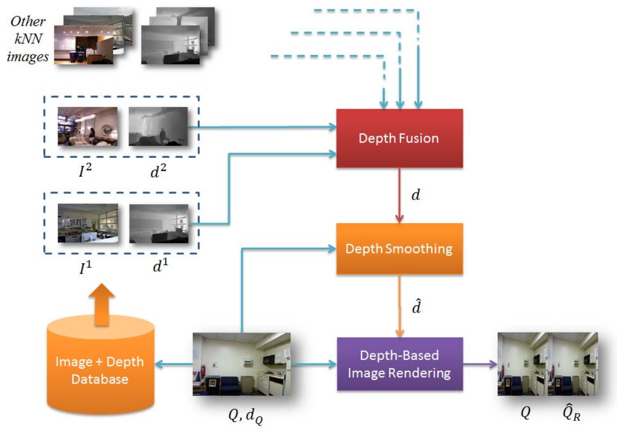


Fig. 3. Block diagram of the overall algorithm; algorithmic details for each block are provided in the sections below.

directly to 3D images represented as an image+depth pair. However, in the case of stereopairs a disparity field needs to be computed first for each left/right image pair. Then, each disparity field can be converted to a depth map, e.g., under a parallel camera geometry assumption, with fusion and smoothing taking place in the space of depths. Alternatively, the fusion and smoothing can take place in the space of disparities (without converting to depth), and the final disparity used for right-image rendering.

Fig. 3 shows the block diagram of our approach. The sections below provide a description of each step and some high-level mathematical detail. In these sections, Q_R is the right image which is being sought for each query image Q, while d_Q is the query depth (ground truth) needed to numerically evaluate the performance of a depth computation. Again, we assume that a 3D dataset $\mathcal{I}$ is available by means of laser range finding, Kinect-based capture or disparity computation. The goal is to find a depth estimate $\widehat{d}$ and then a right-image estimate $\widehat{Q}_R$ given a 2D query image Q and the 3D dataset $\mathcal{I}$.

A. kNN Search

There exist two types of images in a large 3D image repository: those that are relevant for determining depth in a 2D query image, and those that are irrelevant. Images that are not photometrically similar to the 2D query need to be rejected because they are not useful for estimating depth (as per our assumption). Note that although we might miss some depth-relevant images, we are effectively limiting the number of irrelevant images that could potentially be more harmful to the 2D-to-3D conversion process. The selection of a smaller subset of images provides the added practical benefit of computational tractability when the size of the repository is very large.

One method for selecting a useful subset of depth-relevant images from a large repository is to select only the k images that are closest to the query where closeness is measured by some distance function capturing global image properties such as color, texture, edges, etc. As this distance function, we use the Euclidean norm of the difference between histograms of oriented gradients (HOGs) [3] computed from two images. Each HOG consists of 144 real values (4×4 blocks with 9 gradient direction bins) that can be efficiently computed.

We perform a search for top matches to our monocular query Q among all images $\vec{I}^k, k = 1, \ldots, K$ in the 3D database $\mathcal{I}$. The search returns an ordered list of image+depth pairs, from the most to the least photometrically similar *vis-à-vis* the query. We discard all but the top k matches (kNNs) from this list.

Fig. 4 shows search results for two outdoor query images performed on the Make3D dataset #1. Although none of the four kNNs perfectly matches the corresponding 2D query, the general underlying depth is somewhat related to that expected in the query. In Fig. 5 we show search results for two indoor query images (office and dining room) performed on the NYU Kinect dataset. While some of the retained images share local 3D structures with the query image, e.g., a large table in the dining room, other images do not. Again, the general depth is somewhat related to that expected in the query.

The average photometric similarity between a query and its k-th nearest neighbor usually decays with the increasing k. While for large databases, larger values of k may be appropriate, since there are many good matches, for smaller databases this may not be true. Therefore, a judicious selection of k is important. We discuss the choice of k in Section V. We denote by $\mathcal{K}$ the set of indices i of image+depth pairs that are the top k photometrically-nearest neighbors of the query Q.

B. Depth Fusion

In general, none of the NN image+depth pairs $(I^i, d^i), i \in \mathcal{K}$ match the query Q accurately (Figs. 4 and 5). However, the location of some objects (e.g., furniture) and parts of the background (e.g., walls) is quite consistent with those in the respective query. If a similar object (e..g, building, table) appears at a similar location in several kNN images, it is likely that such an object also appears in the query, and the depth field being sought should reflect this. We compute this depth field by applying the median operator across the kNN depths at each spatial location $\mathbf{x}$ as follows:

$$d[\mathbf{x}] = \text{median}\{d^i[\mathbf{x}] \quad \forall i \in \mathcal{K}\}. \tag{3}$$

Examples of the fused depth fields d are shown in the central column of Fig. 7 for two NYU Kinect examples. Although these depths are overly smooth, they provide a globally-correct, although coarse, assignment of distances to various areas of the scene.

C. Cross-Bilateral Filtering (CBF) of Depth

While the median-based fusion helps make depth more consistent globally, the fused depth is overly smooth and locally inconsistent with the query image due to edge misalignment between the depth fields of the kNNs and the query image. This, in turn, often results in the lack of edges in the fused depth where sharp object boundaries should occur and/or the lack of fused-depth smoothness where smooth depth is expected.

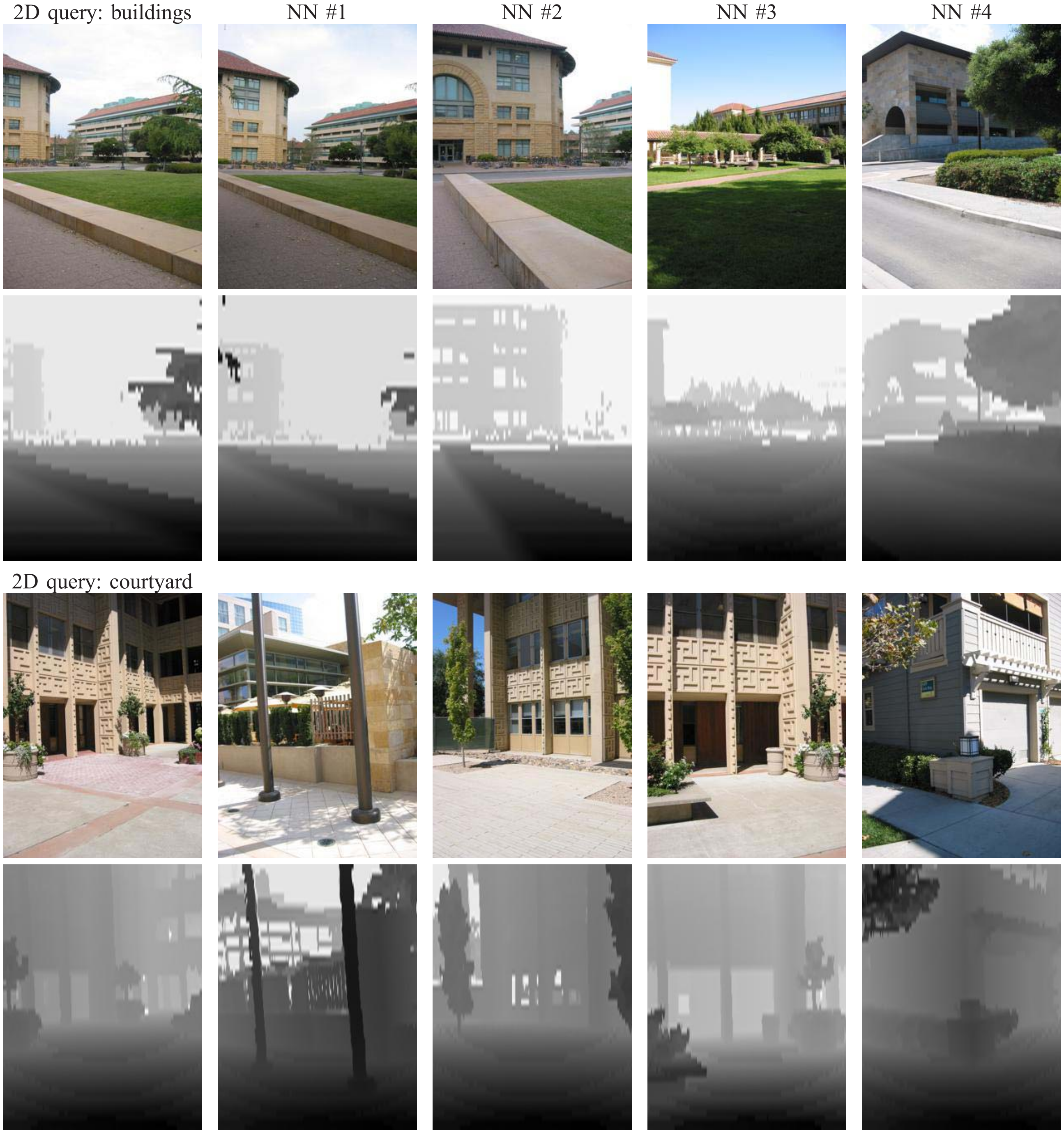


Fig. 4. RGB image and depth field of two 2D queries (left column), and their four nearest neighbors (columns 2–5) retrieved using the Euclidean norm on the difference between histograms of gradients [3]. All image+depth pairs are from the Make3D dataset #1 (see Section V).

In order to correct this, similarly to Agnot *et al.* [1], we apply cross-bilateral filtering (CBF). CBF is a variant of bilateral filtering, an edge-preserving image smoothing method that applies anisotropic diffusion controlled by the local content of the image itself [4]. In CBF, however, the diffusion is not controlled by the local content of the image under smoothing but by an external input. We apply CBF to the fused depth d using the query image Q to control diffusion. This allows us to achieve two goals simultaneously: alignment of the depth edges with those of the luminance Y in the query image Q and local noise/granularity suppression in the fused depth d. This is implemented as follows:

$$\begin{aligned} \widehat{d}[\mathbf{x}] &= \frac{1}{\gamma[\mathbf{x}]}\sum_{\mathbf{y}} d[\mathbf{y}]h_{\sigma_s}(\mathbf{x}-\mathbf{y})h_{\sigma_e}(Y[\mathbf{x}]-Y[\mathbf{y}]), \\ \gamma[\mathbf{x}] &= \sum_{\mathbf{y}} h_{\sigma_s}(\mathbf{x}-\mathbf{y})h_{\sigma_e}(Y[\mathbf{x}]-Y[\mathbf{y}]), \end{aligned} \tag{4}$$

where $\widehat{d}$ is the filtered depth field and $h_\sigma(\mathbf{x}) = \exp(-\|\mathbf{x}\|^2/2\sigma^2)/2\pi\sigma^2$ is a Gaussian weighting function. Note that the directional smoothing of d is controlled by the query image *via* the weight $h_{\sigma_e}(Y[\mathbf{x}]-Y[\mathbf{y}])$. For large

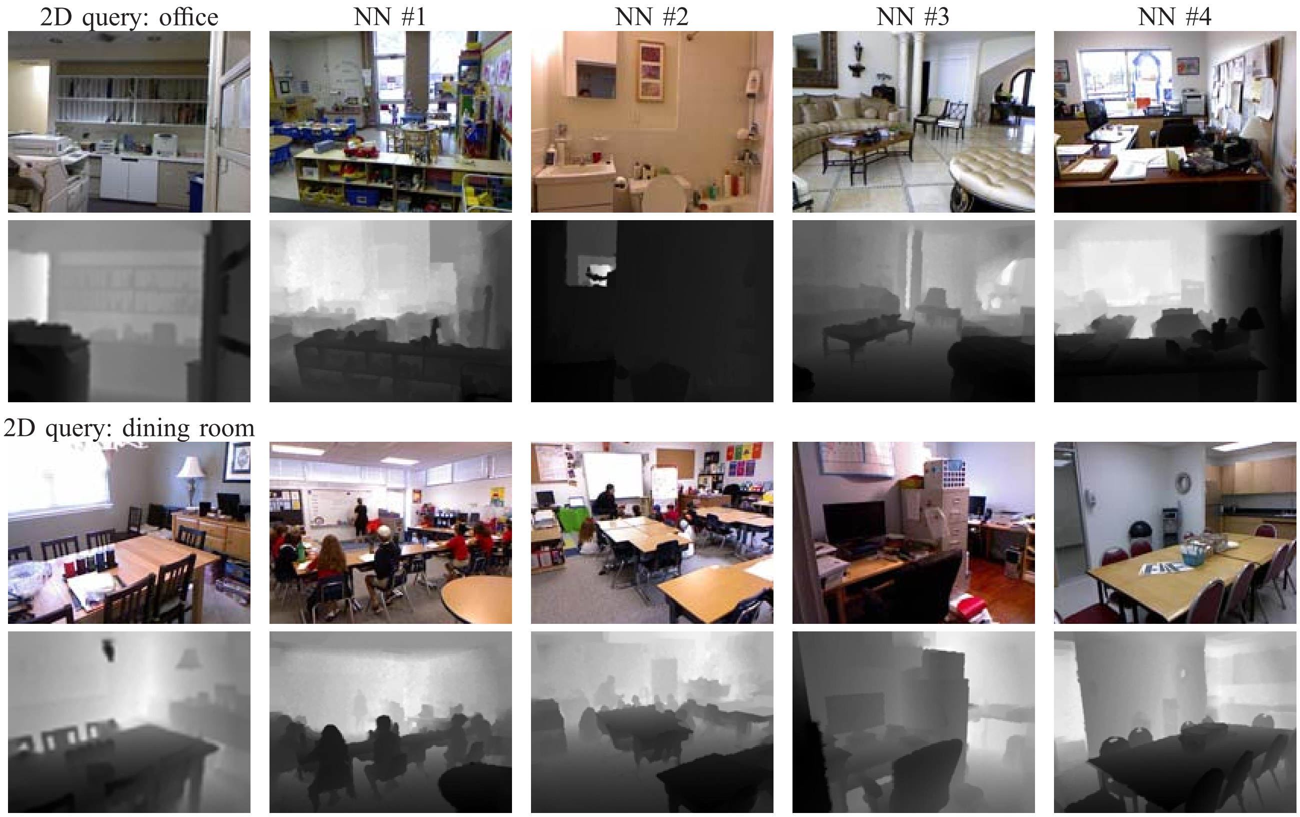


Fig. 5. RGB image and depth field of two 2D queries (left column), and their four nearest neighbors (columns 2-5) retrieved using the Euclidean norm on the difference between histograms of gradients [3]. All image + depth pairs are from the NYU Kinect dataset (see Section V).

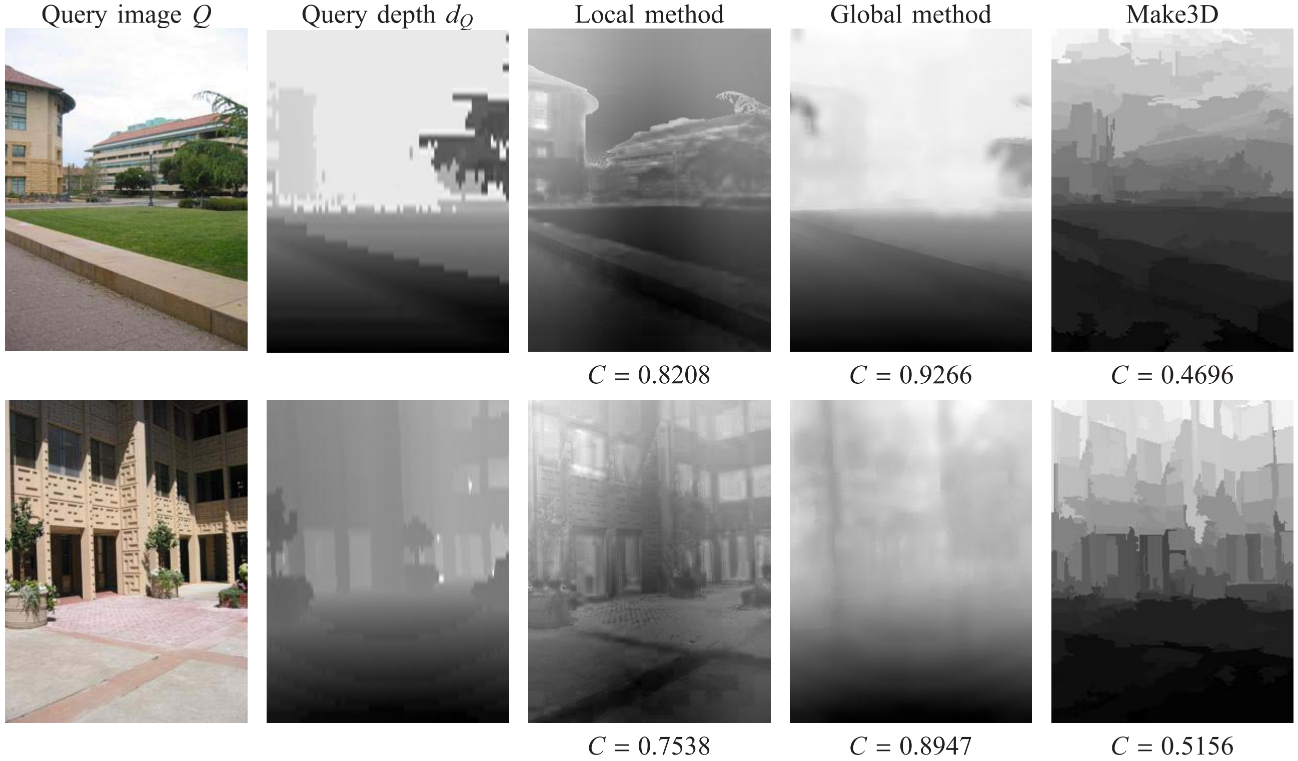


Fig. 6. Query images from Fig. 4 and depth fields: of the query, depth estimated by the local transformation method, depth estimated by the global transformation method (with CBF) and depth computed using the Make3D algorithm. Normalized depth cross-covariances (see equation (6) in Section V) are included under each estimated depth field.

luminance discontinuities, the weight $h_{\sigma_e}(Y[\mathbf{x}]-Y[\mathbf{y}])$ is small and thus the contribution of $d[\mathbf{y}]$ to the output is small. However, when $Y[\mathbf{y}]$ is similar to $Y[\mathbf{x}]$ then $h_{\sigma_e}(Y[\mathbf{x}]-Y[\mathbf{y}])$ is relatively large and the contribution of $d[\mathbf{y}]$ to the output is larger. In essence, depth filtering (smoothing) is happening along (and not across) query edges.

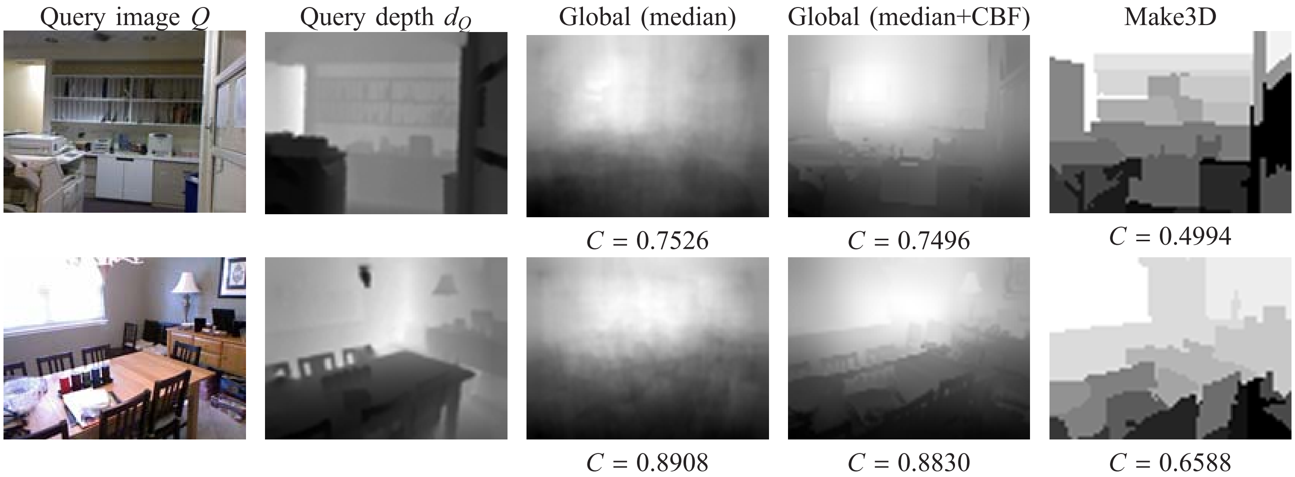


Fig. 7. Query images from Fig. 5 and depth fields: of the query, estimated depth by the global method after median-based fusion and after the same fusion and CBF, and depth computed using the Make3D algorithm. Normalized depth cross-covariances (see equation (6) in Section V) are included under each estimated depth field.

In Fig. 6, we show an example of median-fused depth field after cross-bilateral filtering. Clearly, the depth field is overall smooth (slowly varying) while depth edges, if any, are aligned with features in the query image. Fig. 7 compares the fused depth before cross-bilateral filtering and after. The filtered depth preserves the global properties captured by the unfiltered depth field d, and is smooth within objects and in the background. At the same time it keeps edges sharp and aligned with the query image structure.

D. Stereo Rendering

In order to generate an estimate of the right image $\widehat{Q}_R$ from the monocular query Q, we need to compute a disparity δ from the estimated depth $\widehat{d}$. Assuming that the fictitious image pair $(Q, \widehat{Q}_R)$ was captured by parallel cameras with baseline B and focal length f, the disparity is simply $\delta[x, y] = Bf/\widehat{d}[\mathbf{x}]$, where $\mathbf{x} = [x, y]^T$. We forward-project the 2D query Q to produce the right image:

$$\widehat{Q}_R[x + \delta[x, y], y] = Q[x, y] \tag{5}$$

while rounding the location coordinates $(x + \delta[x, y], y)$ to the nearest sampling grid point. We handle occlusions by depth ordering: if $(x_i + \delta[x_i, y_i], y_i) = (x_j + \delta[x_j, y_i], y_i)$ for some i, j, we assign to the location $(x_i + \delta[x_i, y_i], y_i)$ in $\widehat{Q}_R$ an RGB value from that location (x_i, y_i) in Q whose disparity $\delta[x_i, y_i]$ is the largest. In newly-exposed areas, i.e., for x_j such that no x_i satisfies $(x_j, y_i) = (x_i + \delta[x_i, y_i], y_i)$, we apply simple inpainting using `inpaint_nans` from *MatlabCentral*. Applying a more advanced depth-based rendering method would only improve this step of the proposed 2D-to-3D conversion.

V. Experimental Results

We have tested our approach on two datasets: the Make3D dataset #1 [13], [14], [21] composed of 534 outdoor images with depth fields captured by a laser range finder and the NYU Kinect dataset [15], [22] composed of 1449 pairs of RGB images and corresponding depth fields. Note that the Make3D images are of 1704 × 2272 resolution but the corresponding depth fields are only of 55 × 305 spatial resolution and relatively coarse quantization. Therefore, for computational efficiency, we have re-sized the images to 240 × 320 resolution. On the other hand, the Kinect dataset consists of both images and depth fields at 640 × 480 resolution and the depth precision is relatively high (11 bits).

In order to evaluate the performance of the proposed algorithms quantitatively, we first applied leave-one-out cross-validation (LOOCV) as follows. We selected one image+depth pair from a database as the 2D query (Q, d_Q) treating the remaining pairs as the 3D image repository $\mathcal{I}$ based on which a depth estimate $\widehat{d}$ and a right-image estimate $\widehat{Q}_R$ are computed. As the quality metric, we used normalized cross-covariance between the estimated depth $\widehat{d}$ and the ground-truth depth d_Q defined as follows:

$$C = \frac{1}{N\sigma_{\widehat{d}}\sigma_{d_Q}} \sum_{\mathbf{x}} (\widehat{d}[\mathbf{x}] - \mu_{\widehat{d}})(d_Q[\mathbf{x}] - \mu_{d_Q}) \tag{6}$$

where N is the number of pixels in $\widehat{d}$ and d_Q, $\mu_{\widehat{d}}$ and μ_{d_Q} are the empirical means of $\widehat{d}$ and d_Q, respectively, while $\sigma_{\widehat{d}}$ and σ_{d_Q} are the corresponding empirical standard deviations. The normalized cross-covariance C takes values between -1 and +1 (for values close to +1 the depths are very similar and for values close to -1 they are complementary).

An important parameter in the global algorithm is the number of nearest neighbors k to be used. We selected k by running the LOOCV test for each image in the Kinect database for all k from 1 to 120 and averaging the resulting cross-covariance C across all tests. The average C rapidly rose for small k, achieved maximum at $k = 45$ and then gently rolled-off. Therefore, in all experiments with the global algorithm we used $k = 45$.

Since most of the fully-automatic 2D-to-3D conversion methods have been developed by 3D equipment manufacturers, the employed algorithms are proprietary. The only automatic 2D-to-3D conversion methods for which we were able to find a run-time code was Make3D developed by Saxena *et al.* [14] and a recent method by Karsch *et al.* [7]. Make3D estimates 3D scene structure from a single still image of

TABLE I

AVERAGE AND MEDIAN NORMALIZED CROSS-COVARIANCE C COMPUTED ACROSS ALL IMAGES IN THE MAKE3D DATASET #1 IN AN LOOCV TEST (SEE TEXT FOR DETAILS) FOR THE PROPOSED METHODS, MAKE3D ALGORITHM [14], AND A METHOD BY KARSCH *et al.* [7]

	Local	Global		Make3D	Karsch *et al.*
		Median	Median+CBF		
Average C	0.59	0.78	0.80	0.78	0.73
Median C	0.61	0.85	0.86	0.78	0.79

TABLE II

AVERAGE AND MEDIAN NORMALIZED CROSS-COVARIANCE C COMPUTED ACROSS ALL IMAGES IN THE NYU KINECT DATASET IN AN LOOCV TEST (SEE TEXT FOR DETAILS) FOR THE PROPOSED METHODS, MAKE3D [14], AND A METHOD BY KARSCH *et al.* [7]

	Local	Global		Make3D	Karsch *et al.*
		Median	Median+CBF		
Average C	0.57	0.68	0.71	0.45	0.62
Median C	0.61	0.71	0.75	0.48	0.67

an unstructured environment by supervised learning of 3D position and orientation of small homogeneous patches in the image. The original Make3D algorithm was trained on images from the Make3D dataset #1 and associated laser-scanned depth maps of mostly architectural structures. Admittedly, it was not optimized for indoor scenes that the Kinect depth dataset is composed of, however there is no option provided to re-train Make3D on other datasets. The method of Karsch *et al.* is essentially based on our earlier work using SIFT flow [8]. It consists of finding nearest neighbors using high-level features (they use $k = 7$), followed by SIFT-flow to warp the k depth fields to the current image, and optimization to combine the warped depths while imposing a smoothness constraint and a global depth prior. They do not, however, use median depth fusion and CBF.

Table I shows experimental results obtained from 534 LOOCV tests on the Make3D dataset #1 using various algorithms. Similarly, Table II shows results for 1449 LOOCV tests on the Kinect dataset. The performance of each algorithm has been captured by the average and median of cross-covariance C (6) across all LOOCV tests.

The local method has been trained on the Make3D and Kinect datasets, respectively, i.e., the transformations f_c and f_l (transformation f_m is not used since both datasets contain only still images), have been learned by analyzing depth-color and depth-location relationships in all image-depth pairs of either dataset. We have used weights $w_c = 0.3$ and $w_l = 0.7$ in our experiments. The global method and the method by Karsch *et al.* [7] have no training phase but learn the depth from k best examples found for each query image. As we have already mentioned, the Make3D algorithm [14] has been trained on the Make3D dataset and there is no option available to re-train it on the Kinect dataset.

Clearly, for both datasets the global method with cross-bilateral filtering of the fused depths outperforms all other algorithms, although the same algorithm without the filtering performs very similarly. The numerical gain from filtering the fused depth is rather small since its greatest impact is at depth edges (re-alignment with edges in the query image). Consequently, it affects the normalized cross-covariance at just a few pixels. The Make3D algorithm performs almost as well as our global method on the Make3D dataset. However, this result is biased towards high values of C since the test images in our LOOCV test include images from the database on which the Make3D algorithm was trained (400 training images). Not surprisingly, the same algorithm applied to the Kinect dataset fairs rather poorly but, as we already mentioned, re-training was not possible. The Karsch *et al.* method does not perform as well as the global algorithm on either dataset. Finally, the local method achieves a consistent but low performance which is not surprising given its simplicity.

Examples of computed depth fields for various algorithms are shown in Figs. 6 and 7. Note a higher consistency of depth fields and a better depth edge alignment with ground truth for the global method than other methods. We would like to point out that although C values shown in Fig. 7 are slightly lower for depth fields after cross-bilateral filtering, the depth edge alignment with the query and the high piece-wise depth smoothness are both perceptually beneficial in 3D viewing.

In Fig. 8, we show anaglyph images constructed from $(Q, \widehat{Q}_R)$ image pairs for the ground truth depth d_Q, and the estimated depths $\widehat{d}$ using the proposed global approach and Make3D on both datasets. Neither conversion is flawless. However, the skew of the building and lamp post in the first row and the edge jaggedness on the building wall in the second row for images produced by the Make3D algorithm are visually disturbing[1]. Similarly, errors on the bulletin board in the office image (third row) and under the chest of drawers in the dining room image (bottom row) produced by Make3D cause visual discomfort when viewed with anaglyph glasses.

In terms of the computational complexity, a comparison of these algorithms is rather difficult. An extended variant of the proposed local algorithm, that includes the motion transformation f_m, is currently in use by YouTube for automatic conversion of monoscopic videos to stereo. This implementation works on YouTube's distributed computing infrastructure and it would be unfair to compare its speed to our experiments with the other algorithms on a single PC. The global algorithm including CBF requires about 2−3 s for the complete LOOCV test on the Make3D dataset and 5–6s on the Kinect dataset,

[1] It is interesting to note that the anaglyph image based on the ground-truth depth in the second row of Fig. 8 exhibits significant distortions on the vertical edges around the doors and the windows in the mid-right. This ground-truth depth was captured at low-resolution (significantly different horizontally and vertically) by a laser-range finder (Make3D dataset #1). Such distortions are largely absent from the anaglyph image derived from the global method's depth (center column). The underlying depth seems locally more accurate, than the laser-scanned depth, likely due to resolution.

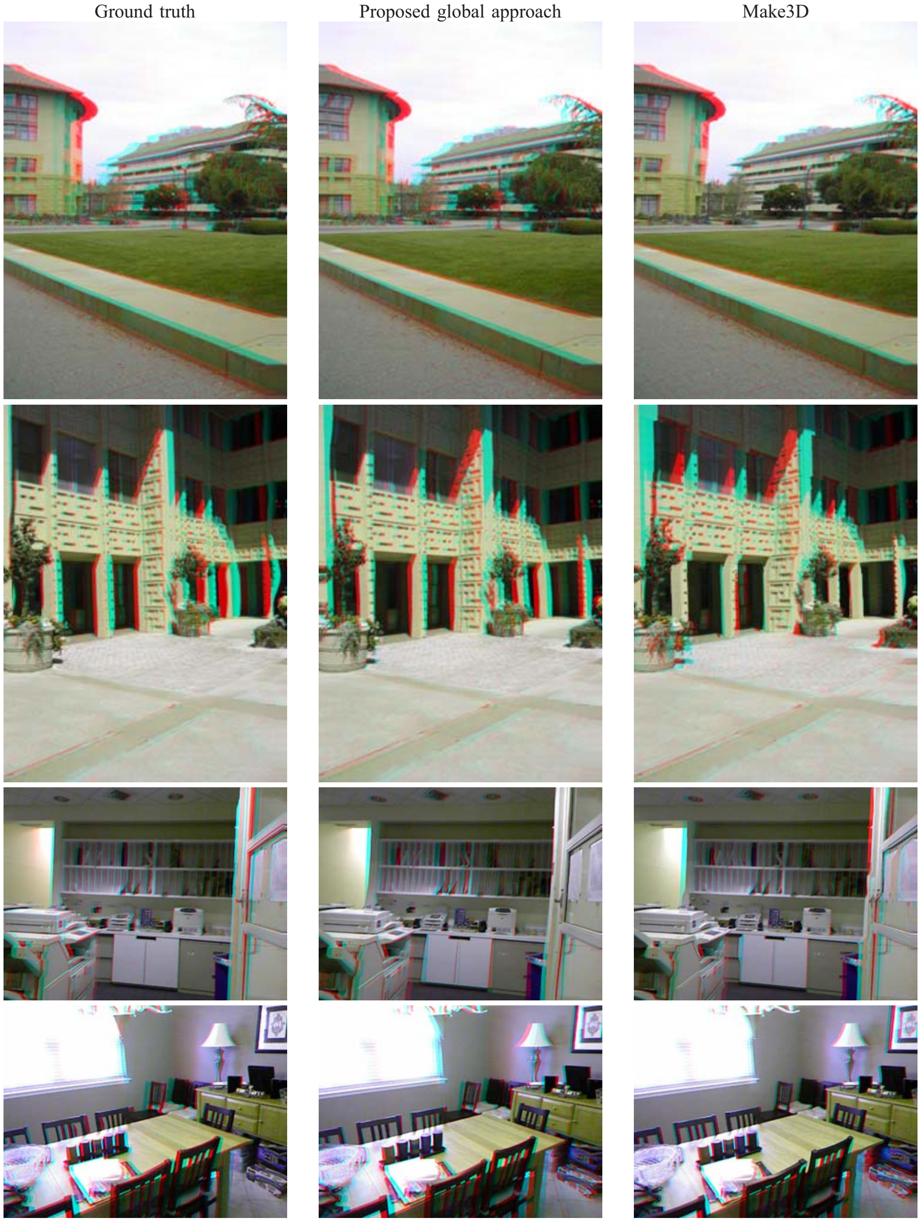


Fig. 8. Anaglyph images generated using the ground-truth depth and depths estimated by the proposed global and Make3D algorithms. In order to appreciate depth, these images should be viewed in color (e.g., in a PDF version of the paper) through red-blue or red-cyan anaglyph glasses.

implemented in C and performed on a 32-core 3.0 GHz Xeon E5 CPU running 12 threads. As for the other algorithms, we simply ran the provided code on the same computer. The Make3D algorithm required about 4h and 12h in the respective tests, whereas the Karsch *et al.* algorithm needed almost 9h and over 26h, respectively. This is not surprising since the Make3D algorithm uses an involved depth learning step[2], whereas the Karsch *et al.* algorithm uses SIFT warping, which is computationally very intensive, followed by depth optimization.

In fact, in our very first paper where we proposed a learning-based method for 2D-to-3D conversion [8] we did indeed use SIFT depth alignment prior to depth fusion. We quickly realized, however, that gains offered by SIFT warping are small for the computational effort required. For example, the additional step of estimating SIFT flow from a query and each of the kNN images, and the subsequent SIFT-based depth warping increased the run time from about 5–6s to 16h in a complete LOOCV test on the Kinect dataset while offering no increase in the average C value and a 0.01 increase in the median C value to $C = 0.76$ [9]. Clearly, warping the depths associated with the k best-matched images is counterproductive yet costly for individual depths are often quite different from the query.

In addition to LOOCV tests on the Make3D dataset #1, where the test image may belong to the set of original 400 training images, we also applied the test used by

[2] We must note at this point that, due to Make3D's complexity, the depth learning step was performed on reduced-resolution Kinect images and depth fields (80 × 60) as opposed to full-resolution (640 × 480). Had we used full-resolution data, we would have to wait over 4 weeks for Make3D's output. We believe that depth learning at low resolution is acceptable if depth edges are aligned with photometric boundaries, because depth varies smoothly within objects and background. The estimated depth fields $\widehat{d}$ were interpolated to full resolution prior to the right-image rendering.

TABLE III

AVERAGE AND MEDIAN NORMALIZED CROSS-COVARIANCE C COMPUTED ACROSS 134 TEST IMAGES IN THE MAKE3D DATASET #1 USING THE PROPOSED METHODS, MAKE3D ALGORITHM [14], AND A METHOD BY KARSCH *et al.* [7]. ALL METHODS WERE TRAINED ON 400 IMAGES OF THE SAME DATASET AND EXCLUDED THE TEST IMAGES

	Local	Global Median+CBF	Make3D	Karsch *et al.*
Average C	0.58	0.73	0.68	0.76
Median C	0.60	0.76	0.72	0.79

Saxena *et al.* [14]. Namely, the 534 images of this dataset were divided into 134 test images and 400 training images (on which the Make3D algorithm was trained). We selected our test image from the test set and used the training set to find the k nearest neighbors. Table III shows the numerical results obtained. Comparing with the results of Table I we can observe, roughly, a global performance drop across all methods. This is to be expected since LOOCV uses more training images. The performance of the local method appears to be only marginally affected. This can be attributed to the use of a fixed point mapping. Both Make3D and our global method with CBF experience a significant performance drop but Make3D continues to trail behind our method. The method of Karsch *et al.* appears to be more robust, even improving slightly in terms of the average C value, but it takes about 2 hours to execute while processing 12 images in parallel. In contrast, Make3D takes about 30mins and our global method with CBF takes about 1 second to process.

VI. CONCLUSION

We have proposed a new class of methods aimed at 2D-to-3D image conversion that are based on the radically different approach of learning from examples. One method that we proposed is based on learning a point mapping from local image attributes to scene-depth. The other method is based on globally estimating the entire depth field of a query directly from a repository of image + depth pairs using nearest-neighbor-based regression. We have objectively validated our algorithms' performance against state-of-the-art algorithms. While the local method was outperformed by other algorithms, it is extremely fast as it is, basically, based on table look-up. However, our global method performed better than the state-of-the-art algorithms in terms of cumulative performance across two datasets and two testing methods, and has done so at a fraction of CPU time. Anaglyph images produced by our algorithms result in a comfortable 3D experience but are not completely void of distortions. Clearly, there is room for improvement in the future. With the continuously increasing amount of 3D data on-line and with the rapidly growing computing power in the cloud, the proposed framework seems a promising alternative to operator-assisted 2D-to-3D image and video conversion.

ACKNOWLEDGMENT

The authors would like to acknowledge Geoffrey Brown for the implementation of the very first, SIFT-based, variant of the 2D-to-3D image conversion method reported here.

REFERENCES

[1] L. Angot, W.-J. Huang, and K.-C. Liu, "A 2D to 3D video and image conversion technique based on a bilateral filter," *Proc. SPIE*, vol. 7526, p. 75260D, Feb. 2010.

[2] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert, "High accuracy optical flow estimation based on a theory for warping," in *Proc. Eur. Conf. Comput. Vis.*, 2004, pp. 25–36.

[3] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2005, pp. 886–893.

[4] F. Durand and J. Dorsey, "Fast bilateral filtering for the display of high-dynamic-range images," *ACM Trans. Graph.*, vol. 21, pp. 257–266, Jul. 2002.

[5] M. Grundmann, V. Kwatra, and I. Essa, "Auto-directed video stabilization with robust L1 optimal camera paths," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2011, pp. 225–232.

[6] M. Guttmann, L. Wolf, and D. Cohen-Or, "Semi-automatic stereo extraction from video footage," in *Proc. IEEE Int. Conf. Comput. Vis.*, Oct. 2009, pp. 136–142.

[7] K. Karsch, C. Liu, and S. B. Kang, "Depth extraction from video using non-parametric sampling," in *Proc. Eur. Conf. Comput. Vis.*, 2012, pp. 775–788.

[8] 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," *Proc. SPIE*, vol. 8288, p. 82880F, Jan. 2012.

[9] J. Konrad, M. Wang, and P. Ishwar, "2D-to-3D image conversion by learning depth from examples," in *Proc. IEEE Comput. Soc. CVPRW*, Jun. 2012, pp. 16–22.

[10] M. Liao, J. Gao, R. Yang, and M. Gong, "Video stereolization: Combining motion analysis with user interaction," *IEEE Trans. Visualizat. Comput. Graph.*, vol. 18, no. 7, pp. 1079–1088, Jul. 2012.

[11] B. Liu, S. Gould, and D. Koller, "Single image depth estimation from predicted semantic labels," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2010, pp. 1253–1260.

[12] R. Phan, R. Rzeszutek, and D. Androutsos, "Semi-automatic 2D to 3D image conversion using scale-space random walks and a graph cuts based depth prior," in *Proc. 18th IEEE Int. Conf. Image Process.*, Sep. 2011, pp. 865–868.

[13] A. Saxena, S. H. Chung, and A. Y. Ng, "Learning depth from single monocular images," in *Advances in Neural Information Processing Systems*. Cambridge, MA, USA: MIT Press, 2005.

[14] A. Saxena, M. Sun, and A. Ng, "Make3D: Learning 3D scene structure from a single still image," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 5, pp. 824–840, May 2009.

[15] N. Silberman and R. Fergus, "Indoor scene segmentation using a structured light sensor," in *Proc. Int. Conf. Comput. Vis. Workshops*, Nov. 2011, pp. 601–608.

[16] M. Subbarao and G. Surya, "Depth from defocus: A spatial domain approach," *Int. J. Comput. Vis.*, vol. 13, no. 3, pp. 271–294, 1994.

[17] R. Szeliski and P. H. S. Torr, "Geometrically constrained structure from motion: Points on planes," in *Proc. Eur. Workshop 3D Struct. Multiple Images Large-Scale Environ.*, 1998, pp. 171–186.

[18] A. Torralba, R. Fergus, and W. T. Freeman, "80 million tiny images: A large data set for nonparametric object and scene recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 11, pp. 1958–1970, Nov. 2008.

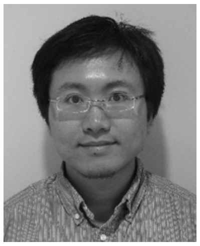
[19] M. Wang, J. Konrad, P. Ishwar, K. Jing, and H. Rowley, "Image saliency: From intrinsic to extrinsic context," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2011, pp. 417–424.

[20] R. Zhang, P. S. Tsai, J. Cryer, and M. Shah, "Shape-from-shading: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 21, no. 8, pp. 690–706, Aug. 1999.

[21] (2012). *Make3D* [Online]. Available: http://make3d.cs.cornell.edu/data.html

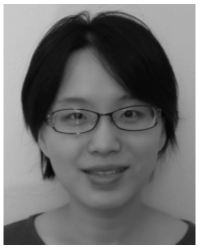
[22] (2012). *NYU Depth V1* [Online]. Available: http://cs.nyu.edu/~silberman/datasets/nyu_depth_v1.html

Janusz Konrad (M'93–SM'98–F'08) received the M.Eng. degree from the Technical University of Szczecin, Szczecin, Poland, in 1980, and the Ph.D. degree from McGill University, Montréal, QC, Canada, in 1989. From 1989 to 2000, he was with the INRST élécommunications, Montréal. Since 2000, he has been with Boston University, Boston, MA, USA. He is an Area Editor for the *EURASIP Signal Processing: Image Communications Journal*. He was an Associate Editor for the IEEE Transactions on Image Processing, *Communications Magazine*, and Signal Processing Letters, as well as the *EURASIP International Journal on Image and Video Processing*. He was a member of the IMDSP Technical Committee of the IEEE Signal Processing Society, the Technical Program Co-Chair of ICIP in 2000, Tutorials Co-Chair of ICASSP in 2004, and Technical Program Co-Chair of AVSS in 2010. He is currently the General Chair of AVSS, Kraków, Poland, in 2013. He is a co-recipient of the 2001 Signal Processing Magazine Award for a paper co-authored with Dr. C. Stiller and the EURASIP Image Communications Best Paper Award from 2004 to 2005 for a paper co-authored with Dr. N. Božinović. His current research interests include image and video processing, stereoscopic and 3-D imaging and displays, visual sensor networks, and human-computer interfaces.



Meng Wang is currently a Software Engineer with Google, Inc., Mountain View, CA, USA. He received the B.A.Sc. degree in electrical engineering from Shanghai Jiaotong University, Shanghai, China, in 2008, and the M.Sc. degree from Boston University, Boston, MA, USA, in 2012. His current research interests include large scale image/video retrieval and its applications to 2D-to-3D image conversion and scene understanding. He has been exploring how to convert standard 2-D videos into 3-D videos, and how to automatically measure the video quality.

Prakash Ishwar (SM'07) received the B.Tech. degree in electrical engineering from the Indian Institute of Technology, Bombay, India, in 1996, and the M.S. and Ph.D. degrees in electrical and computer engineering from the University of Illinois, Urbana-Champaign, Urbana, IL, USA, in 1998 and 2002, respectively. After two years as a Post-Doctoral Researcher in the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA, USA, he joined Boston University, Boston, MA, USA, where he is currently an Associate Professor in the Department of Electrical and Computer Engineering. His current research interests include statistical signal processing, machine learning, visual information analysis and processing, information theory, and security. He is a recipient of the 2005 United States National Science Foundation CAREER Award, a co-recipient of the Best Paper Award with the 2010 IEEE International Conference on Advanced Video and Signal-based Surveillance, and a co-winner of the 2010 Aerial View Activity Classification Challenge in the International Conference on Pattern Recognition. He is an Associate Editor of the IEEE Transactions on Signal Processing, an Elected Member of the IEEE Signal Processing Theory and Methods Technical Committee, and a Former Member of the IEEE Image, Video, and Multidimensional Signal Processing Technical Committee.



Chen Wu received the B.S. degree from the Department of Automation, Tsinghua University, Beijing, China, in 2005, and the M.S. and Ph.D. degrees in electrical engineering from Stanford University, Stanford, CA, USA, in 2007 and 2011, respectively. She is currently a Senior Software Engineer with Google, Inc., Mountain View, CA, USA.

Debargha Mukherjee received the M.S. and Ph.D. degrees in ECE from the University of California, Santa Barbara, CA, USA, in 1995 and 1999, respectively. From 1999 to 2010, he was with Hewlett Packard Laboratories conducting research on video and image coding and processing. Since 2010, he has been with Google, Inc., Mountain View, CA, USA, where he is involved with next generation open-source video codec development. He was responsible for 2-D to 3-D conversion on YouTube. His current research interests include signal processing, image and video compression, and information theory. He has authored or co-authored more than 70 papers and holds more than 20 U.S. patents, with several more pending. He was a recipient of the Student Paper Award from IEEE ICIP in 1998, and was a Keynote Speaker with the SPIE Stereoscopic Displays and Applications conference, and the CVPR 3-D Cinematography workshop, in 2012, to talk about YouTube's 2-D-to-3-D conversion service.