Example Based 3D Reconstruction from Single 2D Images

Tal Hassner and Ronen Basri
The Weizmann Institute of Science
Rehovot, 76100 Israel
{tal.hassner, ronen.basri}@weizmann.ac.il

Abstract

*We present a novel solution to the problem of depth reconstruction from a single image. Single view 3D reconstruction is an ill-posed problem. We address this problem by using an example-based synthesis approach. Our method uses a database of objects from a single class (*e.g. *hands, human figures) containing example patches of feasible mappings from the appearance to the depth of each object. Given an image of a novel object, we combine the known depths of patches from similar objects to produce a plausible depth estimate. This is achieved by optimizing a global target function representing the likelihood of the candidate depth. We demonstrate how the variability of 3D shapes and their poses can be handled by updating the example database on-the-fly. In addition, we show how we can employ our method for the novel task of recovering an estimate for the occluded backside of the imaged objects. Finally, we present results on a variety of object classes and a range of imaging conditions.*

1. Introduction

Given a single image of an every day object, a sculptor can recreate its 3D shape (i.e., produce a statue of the object), even if the particular object has never been seen before. Presumably, it is familiarity with the shapes of similar 3D objects (i.e., objects from the same *class*) and how they appear in images, which enables the artist to estimate its shape. This might not be the exact shape of the object, but it is often a good enough estimate for many purposes. Motivated by this example, we propose a novel framework for example based reconstruction of shapes from single images.

In general, the problem of 3D reconstruction from a single 2D image is ill posed, since different shapes may give rise to the same intensity patterns. To solve this, additional constraints are required. Here, we constrain the reconstruction process by assuming that similarly looking objects from the same class (*e.g.*, faces, fish), have similar shapes. We maintain a set of 3D objects, selected as examples of a specific class. We use these objects to produce a database of images of the objects in the class (*e.g.*, by standard rendering techniques), along with their respective depth maps. These provide examples of feasible mappings from intensities to shapes and are used to estimate the shapes of objects in query images.

Our input image often contains a novel object. It is therefore unlikely that the exact same image exists in our database. We therefore devise a method which utilizes the examples in the database to produce novel shapes. To this end we extract portions of the image (i.e., image patches) and seek similar intensity patterns in the example database. Matching database intensity patterns suggest possible reconstructions for different portions of the image. We merge these suggested reconstructions together, to produce a coherent shape estimate. Thus, novel shapes are produced by composing different parts of example objects. We show how this scheme can be cast as an optimization process, producing the likeliest reconstruction in a graphical model.

A major obstacle for example based approaches is the limited size of the example set. To faithfully represent a class, many example objects might be required to account for variability in posture, texture, etc. In addition, unless the viewing conditions are known in advance, we may need to store for each object, images obtained under many conditions. This can lead to impractical storage and time requirements. Moreover, as the database becomes larger so does the risk of false matches, leading to degraded reconstructions. We therefore propose a novel example update scheme. As better estimates for the depth are available, we generate better examples for the reconstruction *on-the-fly*. We are thus able to demonstrate reconstructions under unknown views of objects from rich object classes. In addition, to reduce the number of false matches we encourage the process to use example patches from corresponding semantic parts by adding location based constraints.

Unlike existing example based reconstruction methods, which are restricted to classes of highly similar shapes (*e.g.*, faces [3]) our method produces reconstructions of objects belonging to a variety of classes (*e.g.* hands, human figures).

Proceedings of the 2006 Conference on Computer Vision and Pattern Recognition Workshop (CVPRW'06)
0-7695-2646-2/06 $20.00 © 2006 IEEE

We note that the data sets used in practice do not guarantee the presence of objects sufficiently similar to the query, for accurate reconstructions. Our goal is therefore to produce *plausible* depth estimates and not necessarily *true* depths. However, we show that the estimates we obtain are often convincing enough.

The method presented here allows for depth reconstruction under very general conditions and requires little, if any, calibration. Our chief requirement is the existence of a 3D object database, representing the object class. We believe this to be a reasonable requirement given the growing availability of such databases. We show depth from single image results for a variety of object classes, under a variety of imaging conditions. In addition, we demonstrate how our method can be extended to obtain plausible depth estimates of the *back* side of an imaged object.

2. Related work

Methods for single image reconstruction commonly use cues such as shading, silhouette shapes, texture, and vanishing points [5, 6, 12, 16, 28]. These methods restrict the allowable reconstructions by placing constraints on the properties of reconstructed objects (*e.g.*, reflectance properties, viewing conditions, and symmetry). A few approaches explicitly use examples to guide the reconstruction process. One approach [14, 15] reconstructs outdoor scenes assuming they can be labelled as "ground," "sky," and "vertical" billboards. A second notable approach makes the assumption that all 3D objects in the class being modelled lie in a linear space spanned using a few basis objects (*e.g.*, [2, 3, 7, 22]). This approach is applicable to faces, but it is less clear how to extend it to more variable classes because it requires dense correspondences between surface points across examples. Here, we assume that the object viewed in the query image has similar looking counterparts in our example set. Semi-automatic tools are another approach to single image reconstruction [19, 29]. Our method, however, is automatic, requiring only a fixed number of numeric parameters.

We produce depth for a query image in a manner reminiscent of example-based texture synthesis methods [10, 25]. Later publications have suggested additional applications for these synthesis schemes [8, 9, 13]. We note in particular, the connection between our method, and Image Analogies [13]. Using their jargon, taking the pair A and A' to be the database image and depth, and B to be the query image, B', the synthesized result, would be the query's depth estimate. Their method, however, cannot be used to recover depth under an unknown viewing position, nor handle large data sets. The optimization method we use here is motivated by the method introduced by [26] for image and video hole-filling, and [18] for texture synthesis. In [18] this optimization method was shown to be comparable to the state of the art in texture synthesis.

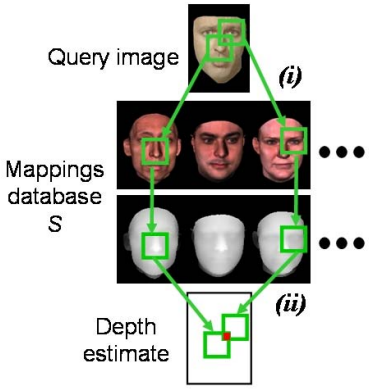

Figure 1. **Visualization of our process.** Step (i) finds for every query patch a similar patch in the database. Each patch provides depth estimates for the pixels it covers. Thus, overlapping patches provide several depth estimates for each pixel. We use these estimates in step (ii) to determine the depth for each pixel.

3. Estimating depth from example mappings

Given a query image I of some object of a certain class, our goal is to estimate a depth map D for the object. To determine depth our process uses examples of feasible mappings from intensities to depths for the class. These mappings are given in a database $S = \{M_i\}_{i=1}^n = \{(I_i, D_i)\}_{i=1}^n$, where I_i and D_i respectively are the image and the depth map of an object from the class. For simplicity we assume first that all the images in the database contain objects viewed in the same viewing position as in the query image. We relax this requirement later in Sec. 3.2.

Our process attempts to associate a depth map D to the query image I, such that every patch of mappings in $M = (I, D)$ will have a matching counterpart in S. We call such a depth map a *plausible* depth estimate. Our basic approach to obtaining such a depth is as follows (see also Fig. 1). At every location p in I we consider a $k \times k$ window around p. For each such window, we seek a matching window in the database with a similar intensity pattern in the least squares sense (Fig. 1.(i)). Once such a window is found we extract its corresponding $k \times k$ depths. We do this for all pixels in I, matching overlapping intensity patterns and obtaining k^2 best matching depth estimates for every pixel. The depth value at every p is then determined by taking an average of these k^2 estimates (Fig. 1.(ii)).

There are several reasons why this approach, on its own, is insufficient for reconstruction.

- The depth at each pixel is selected independently of its neighbors. This does not guarantee that patches in M will be consistent with those in the database. To obtain

a depth which is consistent with both input image and depth examples we therefore require a strong global optimization procedure. We describe such a procedure in Sec. 3.1.

- Capturing the variability of posture and viewing angles of even a simple class of objects, with a fixed set of example mappings may be very difficult. We thus propose an online database update scheme in Sec. 3.2.

- Similar intensity patterns may originate from different semantic parts, with different depths, resulting in poor reconstructions. We propose to constrain patch selection by using relative position as an additional cue for matching (Sec. 3.3).

3.1. Global optimization scheme

We produce depth estimates by applying a global optimization scheme for iterative depth refinement. We take the depth produced as described in Fig. 1 as an initial guess for the object's depth, D, and refine it by iteratively repeating the following process until convergence. At every step we seek for every patch in M, a database patch similar in both intensity as well as depth, using D from the previous iteration for the comparison. Having found new matches, we compute a new depth estimate for each pixel by taking the Gaussian weighted mean of its k^2 estimates (as in Fig. 1.(ii)). Note that this optimization scheme, is similar to the one presented for hole-filling by [26], and texture synthesis in [18].

Fig. 2 summarizes this process. The function $getSimilarPatches$ searches S for patches of mappings which match those of M, in the least squares sense. The set of all such matching patches is denoted $\mathcal{V}$. The function $updateDepths$ then updates the depth estimate D at every pixel p by taking the mean over all depth values for p in $\mathcal{V}$.

D = estimateDepth(I, S)
  $M = (I, ?)$
  repeat until no change in M
  (i) $\mathcal{V}$ = getSimilarPatches(M, S)
  (ii) D = updateDepths(M, $\mathcal{V}$)
  $M = (I, D)$

Figure 2. **Summary of the basic steps of our algorithm.**

It can be shown that this process is in fact a hard-EM optimization [17] of the following global target function. Denote by W_p a $k \times k$ window from the query M centered at p, containing both intensity values and (unknown) depth values, and denote by V a similar window in some $M_i \in S$.

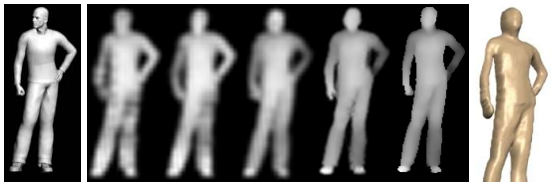

Figure 3. **Man figure reconstruction**. From left to right, input image, five intermediate depth map results from different resolutions, and a zoomed in view of our output reconstruction.

Our target function can now be defined as

$$Plaus(D|I,S) = \sum_{p \in I} \max_{V \in S} Sim(W_p, V), \qquad (1)$$

with the similarity measure $Sim(W_p, V)$ being:

$$Sim(W_p, V) = \exp\left(-\frac{1}{2}(W_p - V)^T \Sigma^{-1} (W_p - V)\right),$$

where Σ is a constant diagonal matrix, its components representing the individual variances of the intensity and depth components of patches in the class. These are provided by the user as weights (see also Sec. 5.1). To make this norm robust to illumination changes we normalize the intensities in each window to have zero mean and unit variance, similarly to the normalization often applied to patches in detection and recognition methods (*e.g.*, [11]).

We present a proof sketch for these claims in the appendix. Note that consequently, this process is guaranteed to converge to a local maximum of $Plaus(D|I,S)$.

The optimization process is further modified as follows:
Multi-scale processing. The optimization is performed in a multi-scale pyramid representation of M. This both speeds convergence and adds global information to the process. Starting at the coarsest scale, the process iterates until convergence of the depth component. Final coarse scale selections are then propagated to the next, finer scale (i.e., by multiplying the coordinates of the selected patches by 2), where intensities are then sampled from the finer scale example mappings. Fig. 3 demonstrates some intermediate depth estimates, from different scales.
Approximate nearest neighbor (ANN) search. The most time consuming step in our algorithm is seeking a matching database window for every pixel in $getSimilarPatches$. We speed this search by using a sub-linear approximate nearest neighbor search [1]. This does not guarantee finding the *most similar* patches V, however, we have found the optimization robust to these approximations, and the speedup to be substantial.

3.2. Example update scheme

Patch examples are now regularly used in many applications, ranging from recognition to texture synthesis. The

underlying assumption behind these methods is that class variability can be captured by a finite, often small, set of examples. This is often true, but when the class contains non-rigid objects, objects varying in texture, or when viewing conditions are allowed to change, this can become a problem. Adding more examples to allow for more variability (*e.g.* rotations of the input image in [8]), implies larger storage requirements, longer running times, and higher risk of false matches. In this work, we handle non-rigid objects (*e.g.* hands), objects which vary in texture (*e.g.* the fish) and can be viewed from any direction. Ideally, we would like our examples to be objects whose shape is similar to that of the object in the input image, viewed under similar conditions. This, however, implies a chicken-and-egg problem as reconstruction requires choosing similar objects for our database, but for this we first need a reconstruction.

We thus propose the idea of online example set update. Instead of committing to a fixed database at the onset of reconstruction, we propose updating the database *on-the-fly* during processing. We start with an initial seed database of examples. In subsequent iterations of our optimization we drop the least used examples M_i from our database, replacing them with ones deemed better for the reconstruction. These are produced by on-the-fly rendering of more suitable 3D objects with better viewing conditions. In our experiments, we applied this idea to search for better example objects and better viewing angles. Other parameters such as lighting conditions can be similarly resolved. Note that this implies a potentially infinite example database (*e.g.* infinite views), where only a small relevant subset is used at any one time. We next describe the details of our implementation.

Searching for the best views. Fig. 4 demonstrates a reconstruction using images from a single incorrect viewing angle (Fig. 4.a) and four fixed widely spaced viewing angles (Fig. 4.b). Both are inadequate. It stands to reason that mappings from viewing angles closer to the real one, will contribute more patches to the process than those further away. We thus adopt the following scheme. We start with a small number of pre-selected views, sparsely covering parts of the viewing sphere (the gray cameras in Fig. 4.c). The seed database S is produced by taking the mappings M_i of our objects, rendered from these views, and is used to obtain an initial depth estimate. In subsequent iterations, we re-estimate our views by taking the mean of the currently used angles, weighted by the relative number of patches selected from each angle. We then drop from S mappings originating from the least used angle, and replace them with ones from the new view. If the new view is sufficiently close to one of the remaining angles, we instead increase the number of objects to maintain the size of S. Fig. 4.c presents a result obtained with our angle update scheme.

Although methods exist which accurately estimate the viewing angle [20, 21], we preferred embedding this estimation in our optimization. To understand why, consider non-rigid classes such as the human body where posture cannot be captured with only a few parameters. Our approach uses information from several viewing angles simultaneously, without pre-committing to any single view.

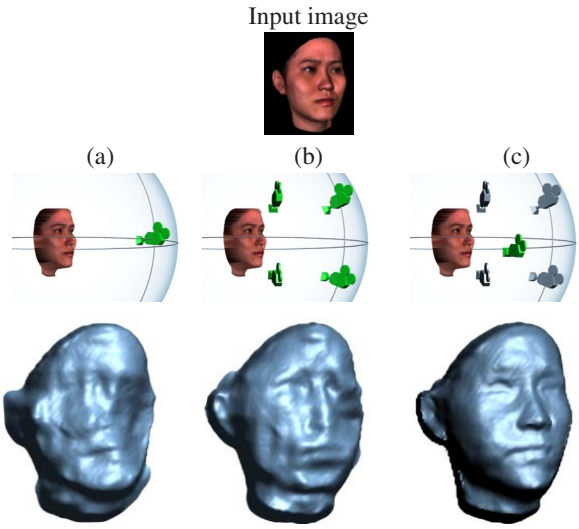

Figure 4. **Reconstruction with unknown viewing angle.** A woman's face viewed from $(\alpha, \beta) = (0^\circ, -22^\circ)$. (a) S rendered from $(0^\circ, 0^\circ)$. (b) Using the angles $(-20^\circ, 0^\circ)$, $(20^\circ, 0^\circ)$, $(-20^\circ, -40^\circ)$, and $(20, -40)$, without update. (c) Reconstruction with our on-the-fly view update scheme. Starting from the angles in (b), now updating angles until convergence to $(-6^\circ, -24^\circ)$.

Searching for the best objects. Although we have collected at least 50 objects in each database, we use no more than 12 objects at a time for the reconstruction, as it becomes increasingly difficult to handle larger sets. We select these as follows. Starting from a set of arbitrarily selected objects, at every update step we drop those leased referenced. We then scan the remainder of our objects for those who's depth, D_i, best matches the current depth estimate D (i.e., $(D - D_i)^2$ is smallest, D and D_i center aligned) adding them to the database instead of those dropped. In practice, a fourth of our objects were replaced after the first iteration of every scale of our multi-scale process.

3.3. Preserving global structure

The scheme described in Sec. 3.1, makes an implicit stationarity assumption [25]: Put simply, the probability for the depth at any pixel, given those of its neighbors, is the same throughout the output image. This is generally untrue for structured objects, where depth often depends on position. For example, the probability of a pixel's depth being tip-of-the-nose high is different at different locations of a face. To overcome this problem, we suggest enforcing *non*-stationarity by adding additional constraints to the patch

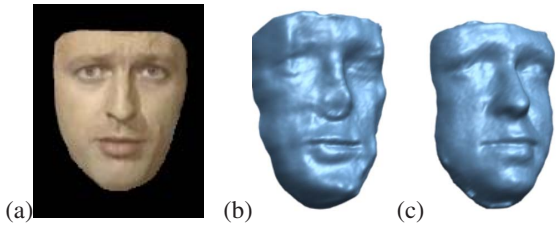

Figure 5. **Preserving relative position.** (a) Input image (b) reconstructed without position preservation constraints and (c) with them.

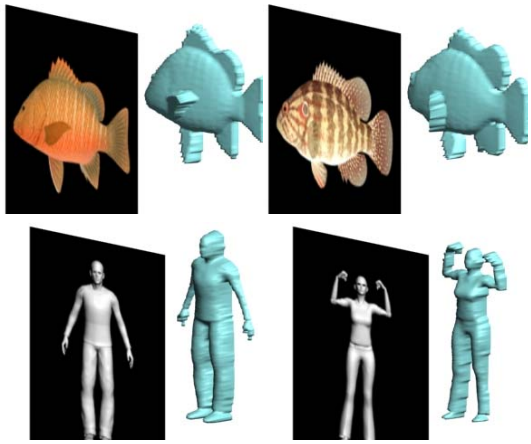

Figure 6. **Example database mappings.** In the top row, two appearance-depth images, out of the 67 in the Fish database. Bottom row, two of 50 pairs from our Human-posture database.

matching process. Specifically, we encourage selection of patches from similar semantic parts, by favoring patches which match not only in intensities and depth, but also in position relative to the centroid of the input depth. This is achieved by adding relative position values to each patch of mappings in both the database and the query image.

Let $p = (x, y)$ be the (normalized) coordinates of a pixel in I, and let (x_c, y_c) be the coordinates of the center of mass of the area occupied by non background depths in the current depth estimate D. We add the values $(\delta x, \delta y) = (x - x_c, y - y_c)$, to each patch W_p and similar values to all database patches (i.e., by using the center of each depth image D_i for (x_c, y_c)). These values now force the matching process to find patches similar in both mapping and global position. Fig. 5 demonstrates a reconstruction result with and without these constraints.

If the query object is segmented from the background, an initial estimate for the query's centroid can be obtained from the foreground pixels. Alternatively, this constraint can be applied only after an initial depth estimate has been computed (i.e., Sec. 3).

4. Backside reconstruction

We have found that our method can be easily extended to produce estimates for the shape of the occluded backside of objects. This is achieved by simply replacing our mappings database with a database containing mappings from front depth to a second depth layer, in this case the depth at the back. Having recovered the visible depth of an object (its depth map, D), we define the mapping from visible to occluded depth as $M'(p) = (D(p), D'(p))$, where D' is a second depth layer. We produce an example database of such mappings by taking the second depth layer of our 3D objects, thus getting $S' = \{M'_i\}_{i=1}^n$. Synthesizing D' can now proceed similarly to the synthesis of the visible depth layers. We note that this idea is similar in spirit to the idea behind image completion schemes.

5. Implementation and results

5.1. Representing mappings

The mapping at each pixel in M, and similarly every M_i, encodes both appearance and depth (See examples in Fig. 6). In practice, the appearance component of each pixel is its intensity and high frequency values, as encoded in the Gaussian and Laplacian pyramids of I [4]. We have found direct synthesis of depths to result in low frequency noise (*e.g.* "lumpy" surfaces). We thus estimate a Laplacian pyramid of the depth instead, producing the final depth by collapsing the depth estimates from all scales. In this fashion, low frequency depths are synthesized in the course scale of the pyramid and only sharpened at finer scales.

Different patch components, including relative positions, contribute different amounts of information in different classes, as reflected by their different variance. For example, faces are highly structured, thus, position plays an important role in their reconstruction. On the other hand, due to the variability of human postures, relative position is less reliable for that class. We therefore amplify different components of each W_p for different classes, by weighting them differently. We use four weights, one for each of the two appearance components, one for depth, and one for relative position. These weights were set once for each object class, and changed only if the query was significantly different from the images in S.

5.2. Implementation

Our algorithm was implemented in MATLAB, except for the ANN code [1], which was used as a stand alone executable. We experimented with the following data sets. Hand and body objects, produced by exporting built-in models from the Poser 6 software, the USF head database [24], and a fish database [23]. Our objects are stored as textured 3D models. We can thus render them

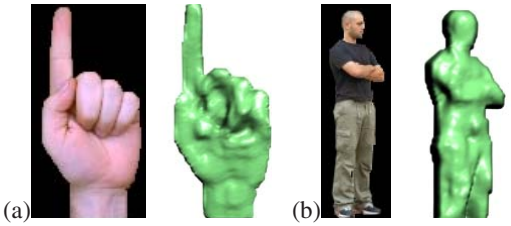

Figure 7. **Failures.** (a) Hand reconstructions are particularly challenging, as they are largely untextured, and can vary greatly in posture. (b) The uniform black shirt differed greatly from the ones worn by our database objects (see Fig. 6). No reliable matches could thus be found, resulting in a lumpy surface. Resulting surface presented from a zoomed-in view.

to produce example mappings using any standard rendering engine. Example mappings from the fish and human posture data-sets are displayed in Fig. 6. We used 3D Studio Max for rendering. We preferred pre-rendering the images and depth maps instead of rendering different objects from different angles on-the-fly. Thus, we trade rendering times with disk access times and large storage. Note that this is an implementation decision; at any one time we load only a small number of images to memory. The angle update step (Sec. 3.2) therefore selects the existing pre-rendered angle closest to the mean angle.

5.3. Experiments

We found the algorithm to perform well on structured rigid objects, such as faces. However, hands, having little texture and varying greatly in shape, proved more difficult, requiring more parameter manipulations. In general the success of a reconstruction relies on the database used, the input image, and how the two match (some failed results are presented in Fig. 7). Results are shown in Figures 5, 4, 8– 11. In addition, Fig. 8 and 9 present results for the backside of imaged objects (i.e., Sec. 4). The query objects were manually segmented from their background and then aligned with a single preselected database image to solve for scale and image-plane rotation differences.

Our running time was approximately 40 minutes for a 200×150 pixel image using 12 example images at any one time, on a Pentium 4, 2.8GHz computer with 2GB of RAM. For all our results we used three pyramid levels, with patch sizes taken to be 5×5 at the coarsest scale, 7×7 at the second level, and 9×9 for the finest scale.

6. Conclusions and future work

This paper presents a method for reconstructing depth from a single image by using example mappings from appearance to depth. We show that not only is the method applicable to many diverse object classes, and images acquired under a range of conditions, it can additionally be used for the novel task of estimating the depth of the back of the object in the image. We further address the problems of global structure preservation, handling large databases, and viewing angles estimation.

We believe our method can be extended in many ways. First, the process itself can be improved to produce better results, consuming less time and memory. Second, we believe it will be interesting to examine if this approach can be extended to handle other vision related problems, in particular to combine segmentation with the reconstruction (*e.g.* by adding binary segmentations to the example mappings.)

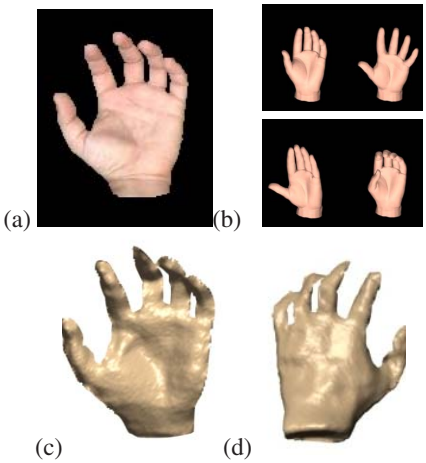

Figure 8. **Hand reconstruction.** (a) Input image. (b) Four most referenced database images in the last iteration of the algorithm. (c) Our output. (d) Output estimate for the **back** of the hand.

7. Acknowledgements

This research has been supported in part by the European Community grant IST-2002-506766 Aim@Shape. The vision group at the Weizmann Institute is supported in part by the Moross Foundation. The authors also thank Eli Shechtman, Lihi Zelnik-Manor, and Michal Irani for helpful comments and discussions, and additionally Shachar Weis for his help implementing the system.

References

[1] S. Arya, D. Mount, N. Netanyahu, R. Silverman, and A. Wu. An optimal algorithm for approximate nearest neighbor searching in fixed dimensions. *Journal of the ACM*, 45(6), 1998. Source code available from www.cs.umd.edu/∼mount/ANN/. 3, 5

[2] J. Atick, P. Griffin, and A. Redlich. Statistical approach to shape from shading: Reconstruction of three-dimensional face surfaces from single two-dimensional images. *Neural Computation*, 8(6):1321–1340, 1996. 2

Figure 9. **Full body reconstructions.** Left to right: Input image, the output depth without and with texture, input image of a man, output depth, textured view of the output, output estimate of the depth at the **back**. Man results shown zoomed-in.

Figure 10. **Two face reconstructions**. Left to right: Input image, four most referenced database images in the last iteration, our output without and with texture, input image, four most referenced database images in the last iteration, our output without and with texture.

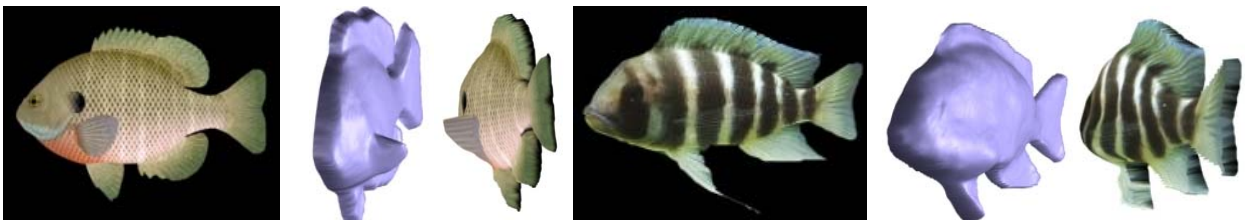

Figure 11. **Two fish reconstructions.** Left to right: Input image (removed from the example database); recovered depth and a textured view of the output; Input image; recovered depth and a textured view of the output.

[3] V. Blanz and T. Vetter. A morphable model for the synthesis of 3D faces. In *SIGGRAPH*, 1999. 1, 2

[4] P. Burt and E. Adelson. The laplacian pyramid as a compact image code. In *IEEE Trans, on Communication*, 1983. 5

[5] R. Cipolla, G. Fletcher, and P. Giblin. Surface geometry from cusps of apparent contours. In *ICCV*, 1995. 2

[6] A. Criminisi, I. Reid, and A. Zisserman. Single view metrology. *IJCV*, 40(2), Nov. 2000. 2

[7] R. Dovgard and R. Basri. Statistical symmetric shape from shading for 3D structure recovery of faces. In *ECCV*, 2004. 2

[8] I. Drori, D. Cohen-Or, and H. Yeshurun. Fragment-based image completion. In *SIGGRAPH*, 2003. 2, 4

[9] A. Efros and W. Freeman. Image quilting for texture synthesis and transfer. In *SIGGRAPH*, 2001. 2

[10] A. Efros and T. Leung. Texture synthesis by non-parametric sampling. In *ICCV*, 1999. 2

[11] R. Fergus, P. Perona, and A. Zisserman. A sparse object category model for efficient learning and exhaustive recognition. In *CVPR*, 2005. 3

[12] F. Han and S.-C. Zhu. Bayesian reconstruction of 3D shapes and scenes from a single image. In *Workshop on Higher-Level Knowledge in 3D Modeling and Motion Analysis*, 2003. 2

[13] A. Hertzmann, C. Jacobs, N. Oliver, B. Curless, and D. Salesin. Image analogies. In *SIGGRAPH*, 2001. 2

[14] D. Hoiem, A. Efros, and M. Hebert. Automatic photo pop-up. In *SIGGRAPH*, 2005. 2

[15] D. Hoiem, A. Efros, and M. Hebert. Geometric context from a single image. In *ICCV*, 2005. 2

[16] B. Horn. *Obtaining Shape from Shading Information*. McGraw-Hill, 1975. 2

[17] M. Kearns, Y. Mansour, and A. Ng. An information-theoretic analysis of hard and soft assignment methods for clustering. In *UAI*, 1997. 3, 8

[18] V. Kwatra, I. Essa, A. Bobick, and N. Kwatra. Texture optimization for example-based synthesis. In *SIGGRAPH*, 2005. 2, 3

[19] B. Oh, M. Chen, J. Dorsey, and F. Durand. Image-based modeling and photo editing. In *SIGGRAPH*, 2001. 2

[20] R. Osadchy, M. Miller, and Y. LeCun. Synergistic face detection and pose estimation with energy-based model. In *NIPS*, 2004. 4

[21] S. Romdhani, V. Blanz, and T. Vetter. Face identification by fitting a 3D morphable model using linear shape and texture error functions. In *ECCV*, 2002. 4

[22] S. Romdhani and T. Vetter. Efficient, robust and accurate fitting of a 3D morphable model. In *ICCV*, 2003. 2

[23] Toucan virtual museum, free fish models, available at: http://toucan.web.infoseek.co.jp/3DCG/3ds/FishModelsE.html. 5

[24] USF. DARPA Human-ID 3D Face Database: Courtesy of Prof. Sudeep Sarkar. University of South Florida, Tampa, FL. http://marthon.csee.usf.edu/HumanID/. 5

[25] L.-Y. Wei and M. Levoy. Fast texture synthesis using tree-structured vector quantization. In *SIGGRAPH*, 2000. 2, 4

[26] Y. Wexler, E. Shechtman, and M. Irani. Space-time video completion. In *CVPR*, 2004. 2, 3

[27] Y. Wexler, E. Shechtman, and M. Irani. Space-time video completion. Submitted as a journal version. See www.wisdom.weizmann.ac.il/∼vision/VideoCompletion.html. 8

[28] A. Witkin. Recovering surface shape and orientation from texture. *AI*, 17(1–3):17–45, 1981. 2

[29] L. Zhang, G. Dugas-Phocion, J.-S. Samson, and S. M. Seitz. Single view modeling of free-form scenes. In *CVPR*, 2001. 2

A. Plausibility as a likelihood function

We now sketch how $Plaus(D|I,S)$ of Eq. 1 can be derived as a likelihood function from a graphical model representation of our problem (Fig. 12). In addition, we show that our optimization (Fig. 2) is a hard-EM variant, producing the local maximum of this likelihood. This derivation is similar to the one in [27], presented here in the context of our scheme.

In Fig. 12 we represent the intensities of the query image I as observables and the matching database patches $\mathcal{V}$ and the sought depth values D as hidden variables. The joint probability of the observed and hidden variables can be formulated through the edge potentials by

$$f(I,\mathcal{V};D)=\prod_{p\in I}\prod_{q\in W_p}\phi_I(V_p(q),I(q))\cdot\phi_D(V_p(q),D(q))$$

where V_p is the database patch matched with W_p by the global assignment $\mathcal{V}$. Taking ϕ_I and ϕ_D to be Gaussians with different covariances over the appearance and depth respectively, implies

$$f(I,\mathcal{V};D)=\prod_{p\in I}Sim(W_p,V_p).$$

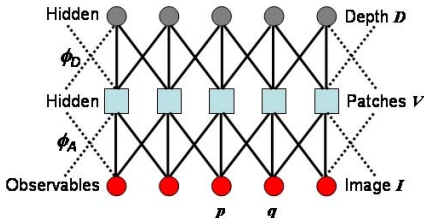

Figure 12. **Graphical model representation.** Please see text for more details.

Integrating over all possible assignments of $\mathcal{V}$ we obtain the likelihood function

$$L=f(I;D)=\sum_{\mathcal{V}}f(I,\mathcal{V};D)=\sum_{\mathcal{V}}\prod_{p\in I}Sim(W_p,V_p).$$

We approximate the sum with a maximum operator. Note that this is common practice for EM algorithms, often called hard-EM (*e.g.*, [17]). Since similarities can be computed independently, we can interchange the product and maximum operators, obtaining the following maximum log likelihood:

$$\max\log L\approx\sum_{p\in I}\max_{V\in S}Sim(W_p,V)=Plaus(D|I,S),$$

which is our cost function (1).

The function $estimateDepth$ (Fig. 2) maximizes this measure by implementing a hard-EM optimization. The function $getSimilarPatches$ performs a hard E-step by selecting the set of assignments $\mathcal{V}^{t+1}$ for time $t+1$ which maximizes the posterior:

$$f(\mathcal{V}^{t+1}|I;D^t)\propto\prod_{p\in I}Sim(W_p,V_p)$$

Here, D^t is the depth estimate at time t. Due to the independence of patch similarities, this can be maximized by finding for each patch in M the most similar patch in the database, in the least squares sense.

The function $updateDepths$ approximates the M-step by finding the most likely depth assignment at each pixel:

$$D^{t+1}(p)=\arg\max_{D(p)}(-\sum_{q\in W_p}(D(p)-depth(V_q^{t+1}(p))^2)).$$

This is maximized by taking the mean depth value over all k^2 estimates $depth(V_q^{t+1}(p))$, for all neighboring pixels q.