Automatic Depth-Map Colorization

T. Hassner[†] and R. Basri[†]

The Weizmann Institute of Science
{tal.hassner, ronen.basri}@weizmann.ac.il

Abstract

We present a system for automatically generating custom, structured image-maps for input depth-maps. Our system thus allows quick fitting of masses of objects with tailor-made image-maps. Given a depth-map of a novel 3D object, our method tiles it with intensities from similar, pre-collected, textured objects. These are seamlessly merged to form the new image-map. This process is performed by optimizing a well defined target likelihood function, via a hard-EM procedure. We present results for varied object classes including human figures, and fish.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Three-Dimensional Graphics and Realism]: Color, shading, shadowing, and texture

1. Introduction

The growing demand for realistic 3D renderings has prompted increasing interest in producing detailed image-maps for 3D models. Manually producing such maps can be a tedious task, made harder when rendering masses of models (e.g., crowds), where separate image-maps must be crafted for each individual object. Automatic methods have mostly focused on texture synthesis on 3D surfaces (e.g., [Tur01, WL01, YHBZ01]). Although high quality results have been demonstrated, these methods uniformly cover models with texture samples. They thus avoid the problem of producing realistic image-maps for non-textural objects, which make up much of the world around us. Take for example objects such as people. Although covering a 3D model of a human, from head to toe with texture (e.g., grass, hair), might look interesting, it will by no means be realistic.

To speed up the modeling pipeline and improve the quality of available image-maps, we propose a system for *automatic* synthesis of custom image-maps for novel shapes. We observe that if objects have similar structures, they can often share similar appearances. Our system thus uses a database of example objects and their image-maps, all from the same class (e.g., fish, busts), to synthesize appearances for novel objects of the same class. Of course, even in structured object classes it is unlikely that example image-maps will neatly fit the 3D features of similar objects. Given a novel object, we thus borrow *parts* of example maps, and seamlessly merge them, producing an image-map tailored to each object's 3D structure. When a single database object is used, this effectively performs an automatic morph of the database image-map to fit the 3D features of the input depth.

Our desired output is an image-map which both fits the 3D features of the input object, and appears similar to maps in our database. We claim that this goal can be expressed as a global target function, and optimized via a hard-EM process. To summarize, this paper makes the following contributions.

- We present a method for *automatically* producing (possibly structured) image-maps for objects.
- We show how hard-EM can be used to solve this problem. Moreover, we show how non-stationarity can be introduced to the synthesis process, and how the process can be expedited to quickly handle many examples.
- Finally, we demonstrate results on a multitude of test objects from three different classes.

Work is underway to extend this idea to colorize meshes. We believe this can be done in much the same way texture synthesis has extended from images to meshes.

2. Image-map synthesis

For a given depth $D(x,y)$, we attempt to synthesize a matching image-map $I(x,y)$. To this end we use examples of feasible mappings from depths to images of similar objects, stored in a database $S = \{M_i\}_{i=1}^{n} = \{(D_i, I_i)\}_{i=1}^{n}$, where D_i and I_i respectively are the depth and image maps of example objects. Our goal is to produce an image-map I such that $M = (D, I)$ will also be feasible. Specifically, we seek an image I which will satisfy the following criteria: (i) For every $k \times k$ patch of mappings in M, there is a similar patch in S. (ii) Database patches matched with overlapping patches in M, will agree on the colors $I(p)$, at overlapped pixels $p = (x,y)$.

Our basic approach is as follows (see also Fig. 1). At every

[†] Supported in part by the European Community grant IST-2002-506766 Aim@Shape and the Moross Foundation.

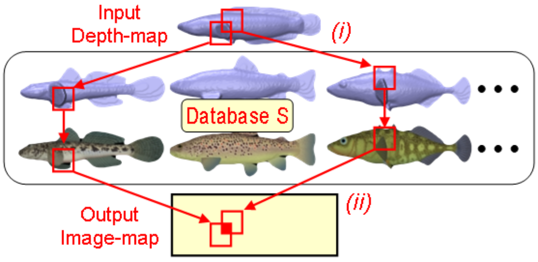

Figure 1: ***Image-map synthesis.*** *Step (i) finds for every query patch a similar database patch. Overlapping patches provide color estimates for their pixels. (ii) These estimates are averaged to determine the color at each pixel.*

p in D we consider a $k\times k$ window around p, and seek a matching window in the database, with a similar depth pattern in the least squares sense (Fig. 1.(i)). Finding such a window, we extract its corresponding $k\times k$ intensities. We do this for all p in D, matching overlapping depth patterns and obtaining k^2 intensity estimates at every pixel. The intensity value at every p is then selected as the Gaussian weighted average of these k^2 estimates (Fig. 1.(ii)).

On its own, this approach has the following shortcomings. (a) Each pixel's intensity is selected independently of its neighbors, thus not guaranteeing satisfaction of our criteria. To achieve this, we describe a strong global optimization procedure in Sec. 2.1. (b) Similar depths may originate from different semantic parts with different intensities, thus spoiling result quality. We propose introducing non-stationarity to the process by constraining patch selection to use relative position as an additional cue for matching (Sec. 2.2). (c) Searching through even small databases can take impractical amounts of time. We speed this process in Sec. 2.3.

2.1. Optimization scheme

We take the image produced as described in Fig. 1 as an initial guess for the object's image-map, 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 depth as well as intensity, using I from the previous iteration for the comparison. Having found new matches, we compute a new color estimate for each pixel by taking the Gaussian weighted mean of its k^2 estimates (as in Fig. 1.(ii)). Note that a similar optimization scheme was used for depth reconstruction from single images in [HB06].

It can be shown that this process optimizes the following global target function, which in turn satisfies our two criteria. Denote by W_p a $k\times k$ window from the query M centered at p, containing both depth values and (unknown) intensities, and denote by V a similar window in some $M_i \in S$. Our target function can now be defined as

$$P(I|D,S) = \sum_{p\in M} \max_{V\in S} Sim(W_p,V),$$

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. These are provided by the user as weights (see also Sec. 3.1). We claim that the matching process can be considered a hard E-step, and computing the mean estimates to produce I is an M-step of a hard-EM process [KMN97]. Note that hard-EM is guaranteed to converge to a local maximum of the target function. Detailed proofs fall outside the scope of this paper.

2.2. Inducing non-stationarity

The scheme described in Sec. 2.1, makes an implicit stationarity assumption. Simply put, the probability for the color of any pixel, given those of its neighbors, is the same throughout the output image. This holds for textures, but it is generally untrue for structured images, where pixel colors often depend on position. For example, the probability for a pixel being lipstick red, is different at different locations of a face. Methods such as [ZWT*05] overcome this problem by requiring the modeler to explicitly form correspondences between regions of the 3D shape and different texture samples. Here, we enforce *non*-stationarity by adding additional constraints to the patch matching process. Specifically, we encourage selection of patches from similar semantic parts, by favoring patches which match not only in depth and color, 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 M, and let (x_c,y_c) be the coordinates of the centroid of the area occupied by non-background depths in 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., using the centroid of each D_i for (x_c,y_c)). These values force the matching process to find patches similar in both mapping and position.

2.3. Accelerating synthesis

Partial databases. Although over 50 mappings M_i were collected for some of our sets S, in practice, we use only small subsets of these databases. Selecting the mappings M_i used for each synthesis depends on the modeler's intentions, as follows. (a) For automatic \ mass image-map production, at the onset of synthesis, we automatically choose the m mappings M_i with the most similar depth-map to D (i.e., $(D-D_i)^2$ is minimal, D and D_i centroid aligned), where normally $m << |S|$. (b) Alternatively, the modeler can manually select specific mappings with desired image-maps.

Multi-scale processing. The optimization is performed in a multi-scale pyramid of M, using similar pyramids for each M_i. This both speeds convergence and adds global information to the process. Starting at the coarsest scale, the process iterates until intensities converge. 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. We thus upscale by multiplying selection coordinates, not interpolating intensities. This was found to better preserve fine scale high frequencies.

PCA patches. Before the first matching process of each scale commences, separate PCA transformation matrices are learned from the depth and intensity bands of the database subset used for synthesis. In practice, we keep a fifth of the basis vectors with the highest variance. The matching process thus finds the most similar PCA reduced patches in the database. This provides a speedup factor of about 5, by loosing some information, however, in all our tests this did not seem to effect the quality of our results.

Approximate nearest neighbor (ANN) search. We speed searching for matching patches by using a sub-linear ANN search [AMN*98]. This does not guarantee finding the most similar patches, but we have found the optimization robust to these approximations, and the speedup substantial.

3. Implementation and results

3.1. Representation of Examples

The depth component of each M_i and similarly M is taken to be the depth itself and its high frequency values, as encoded in the Gaussian and Laplacian pyramids of D. We synthesize three Laplacian pyramids for each of the YCbCr bands of the image-map. The final result is produced by collapsing these pyramids. Consequently, a low frequency image-map is synthesized at the course scale of the pyramid and only refined and sharpened at finer scales.

Different patch components contribute different amounts of information in different classes, as reflected by their different variance. The modeler can thus amplify different components of each W_p by weighting them differently. We use 6 weights, one for each of the two depth components, three for the YCbCr bands, and one for relative position. Although selected manually, these weights were set once for each object class, and left unchanged in all our experiments.

3.2. Implementation

Our algorithm was implemented in MATLAB, except for the ANN code [AMN*98], which was used as a stand alone executable. Our data sets were 5 human objects courtesy of Cyberware Inc., 76 busts from [USF], and 57 fish objects from [Tou]. We assume that test depths are aligned with depths in the database. If not, depths can be aligned using, e.g., [MPD06].

3.3. Experiments

We ran leave-one-out tests on all three data sets, automatically selecting database examples for each synthesis. Parameters are reported in Table 1. Some results are presented in Fig. 2-6. To evaluate the quality of our results, we have poled 10 subjects, asking “how many image-maps are faulty or otherwise inferior to those in the database”. 28% and 24% faults were counted on average in the Fish and Face databases, respectively. Note again that no parameter optimization was carried out for individual depths. With 5 shapes, the humans set was too small for statistics.

Our average running time using un-optimized code, was approximately $7\frac{1}{2}$ minutes for 200×150 pixel depth-map, using $m = 2$ examples of similar dimensions, on a Pentium 4, 2.8GHz computer with 2GB of RAM.

DB Name	*m*	*k*	*Weights*
Humans	1	7, 9, 9	0.08, 0.06, 8, 1.1, 1.1, 10
Busts	2	7, 11, 9	0.08, 0.06, 8, 1.1, 1.1, 10
Fish	2	7, 11, 9	0.08, 0.06, 8, 1.1, 1.1, 0.1

Table 1: *DB parameters. m - Number of mappings M_i used for synthesis. k - Patch width and height, from fine to coarse scale of three pyramid levels. Weights for depth, depth high-frequencies, Y, Cb, Cr, and relative position components.*

References

[AMN*98] ARYA S., MOUNT D., NETANYAHU N., SILVERMAN R., WU A.: 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/.

[HB06] HASSNER T., BASRI R.: Example based 3D reconstruction from single 2D images. In *Beyond Patches Workshop at CVPR* (2006).

[KMN97] KEARNS M., MANSOUR Y., NG A.: An information-theoretic analysis of hard and soft assignment methods for clustering. In *UAI* (1997).

[MPD06] MAKADIA A., PATTERSON A., DANIILIDIS K.: Fully automatic registration of 3D point clouds. In *CVPR* (2006).

[Tou] Toucan virtual museum, available at: http://toucan.web.infoseek.co.jp/3DCG/3ds/FishModelsE.html.

[Tur01] TURK G.: Texture synthesis on surfaces. In *SIGGRAPH* (2001).

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

[WL01] WEI L.-Y., LEVOY M.: Texture synthesis over arbitrary manifold surfaces. In *SIGGRAPH* (2001).

[YHBZ01] YING L., HERTZMANN A., BIERMANN H., ZORIN D.: Texture and shape synthesis on surfaces. In *EGWR* (2001).

[ZWT*05] ZHOU K., WANG X., TONG Y., DESBRUN M., GUO B., SHUM H.-Y.: Texturemontage: Seamless texturing of arbitrary surfaces from multiple images. In *SIGGRAPH* (2005).

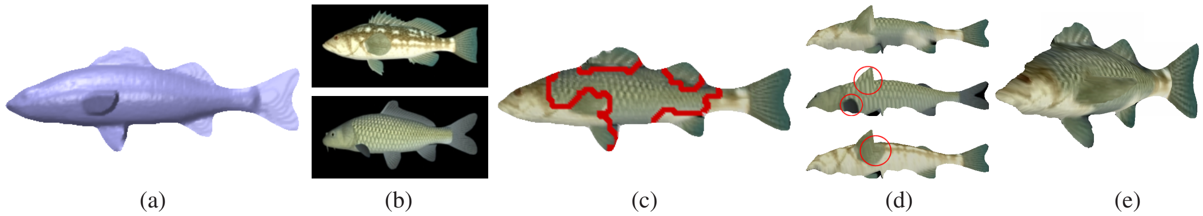

Figure 2: ***Fish image-maps.*** *(a) Input depth-map. (b) Automatically selected DB objects (image-maps displayed). (c) Output image marked with the areas taken from each DB image. (d) Input depth rendered with, from top to bottom, result image and DB image-maps. Note the mismatching features when using the DB images. (e) Textured 3D view of our output.*

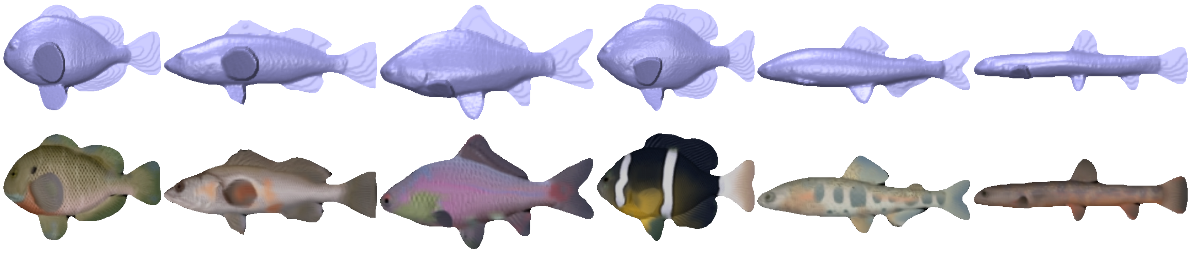

Figure 3: ***Fish image-maps.*** *Top row, input depth-maps; bottom row, our output image-maps.*

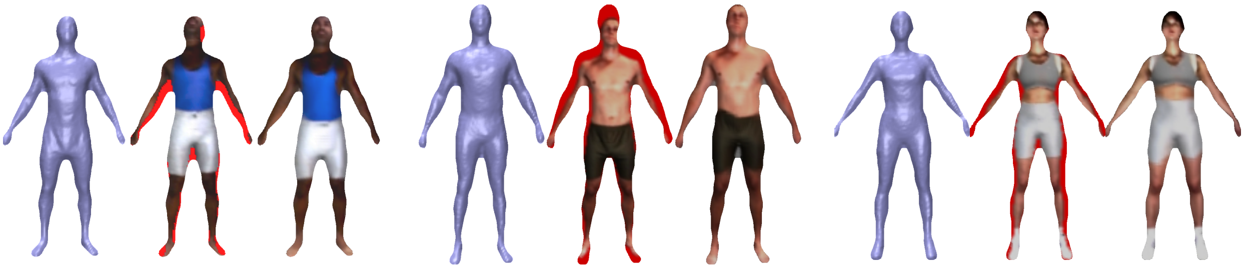

Figure 4: ***Human image-maps.*** *Three human figure results. Using a single DB object, our method effectively morphs the DB image, automatically fitting it to the input depth's 3D features. For each result, displayed from left to right, are the input depth, depth textured with automatically selected DB image-map (in red, depth areas not covered by the DB map,) and our result.*

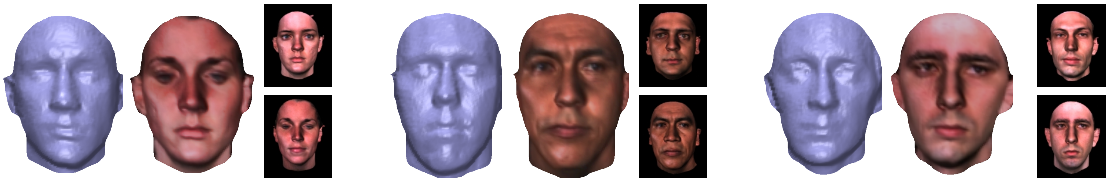

Figure 5: ***Bust image-maps.*** *Three bust results. For each result, displayed from left to right, are the input depth, our result and the two DB objects automatically selected to produce it.*

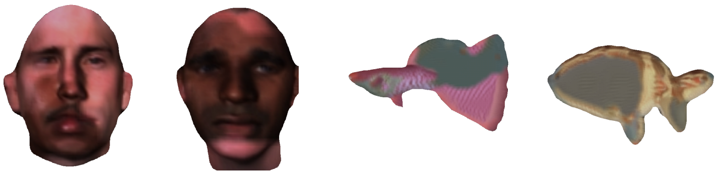

Figure 6: ***Failures.*** *Bust failures caused by differently colored DB image-maps. Fish failures are due to anomalous input depths.*