Learning quadrangulated patches for 3D shape parameterization and completion

Kripasindhu Sarkar[1,2]
kripasindhu.sarkar@dfki.de

Kiran Varanasi[1]
kiran.varanasi@dfki.de

Didier Stricker[1,2]
didier.stricker@dfki.de

[1] Department of Augmented Vision, DFKI Kaiserslautern   [2] University of Kaiserslautern

Abstract

We propose a novel 3D shape parameterization by surface patches, that are oriented by 3D mesh quadrangulation of the shape. By encoding 3D surface detail on local patches, we learn a patch dictionary that identifies principal surface features of the shape. Unlike previous methods, we are able to encode surface patches of variable size as determined by the user. We propose novel methods for dictionary learning and patch reconstruction based on the query of a noisy input patch with holes. We evaluate the patch dictionary towards various applications in 3D shape inpainting, denoising and compression. Our method is able to predict missing vertices and inpaint moderately sized holes. We demonstrate a complete pipeline for reconstructing the 3D mesh from the patch encoding. We validate our shape parameterization and reconstruction methods on both synthetic shapes and real world scans. We show that our patch dictionary performs successful shape completion of complicated surface textures.

1. Introduction

With the growing data set of real world 3D scans, we need efficient 3D shape representation and encoding methods, such that the 3D data can be transmitted and reconstructed accurately. This 3D data is used not only for visualization, but also increasingly for algorithmic analysis. A wide variety of algorithms are developed for 3D object recognition and reconstruction. However, successful machine learning methods on 2D images (*e.g,* deep neural networks) cannot be extended trivially to 3D, as we still need to investigate a common parameterization on which multiple 3D scans can be aligned and compared. This continues to be a challenging problem, especially for shapes of arbitrary topologies. Existing methods typically use generic shape parameterizations such as 3D voxels or depth-maps from arbitrary view points. Unfortunately, these representations are not efficient for fine-scale surface detail. In contrast, 3D meshes can efficiently represent such detail, but do not
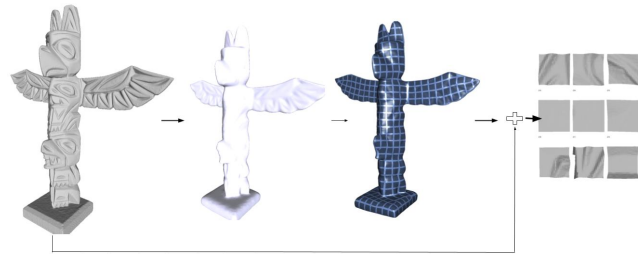

Figure 1: Our patch computation framework - Local patches are computed on reference frames from quad orientations of the quad mesh obtained from the low resolution version of the input mesh.

offer a common representation for learning across multiple shapes.

In 1978, Marr and Nishihara [24] theorized that the brain performs object recognition by matching 2D object descriptions from the visual input to 3D model representations in the memory. They also suggested that the 3D representations in the memory have to be based on object-specific frames of reference, such as the canonical axis of rotation or the medial axis of a shape. They describe three criteria for choosing a shape parameterization: accessibility (ease of computing it), uniqueness to shape, and stability to noise. Our method strikes a balance between the second and third criteria. We compute local patches of moderate length by applying automatic mesh quadrangulation algorithm [13] to the low-resolution representation of an input 3D mesh and taking the stable quad orientations for patch computation. The low-resolution mesh is obtained by applying mesh smoothing to the input 3D scan, which captures the broad outline of the shape over which local patches can be placed. We then set the average quad size and thereby choose the required scale for computing local patches. The mesh quadrangulation is a quasi-global method, which determines the local orientations of the quads based on the distribution of corners and edges on the overall shape. At the same time, the scanlines of the quads retain some robustness towards surface noise and partial scans - these orientations can be fine-tuned further by the user if needed.

Prior work in using local surface patches for 3D shape

compression [12] assumed the patch size to be sufficiently small such that a local patch on the input 3D scan could be mapped to a unit disc. Such small patch sizes would restrict learning based methods from learning any shape detail at larger scale, and for applications like surface inpainting. The contributions of our paper are as follows.

1. We propose a novel shape encoding by local patches oriented by mesh quadrangulation, and a novel method for learning a lossy 3D patch dictionary that can reproduce the original shape upto a chosen degree of accuracy.

2. We demonstrate the application of our patch dictionary to the problem of 3D surface inpainting, which consists of - hole filling and repairing damaged surface regions, using self-similarity from known regions in the mesh. We evaluate the method on real world and synthetic meshes with complex surface textures.

3. We validate the applicability of our patch-dictionary learned from multiple 3D scans thrown into a common data set, towards repairing an individual 3D scan. We show that the reconstruction quality does not deteriorate significantly with increase in the size of the data set.

We specifically address the problem of 3D surface inpainting of shapes with complex surface textures - such as 3D scans of statues, footwear etc., with the goal of inferring detail missing at holes and surface damages resulted due to bad reconstruction from 3D acquisition devices. We present our contributions as a step towards larger-scale machine learning from bigger 3D data sets.

2. Related Work

We review the literature in 3D point cloud processing in the context of 3D object databases, both synthetic [10] and captured from the real world [15, 34]. Dense 3D reconstruction of real world scenes can be achieved by SLAM from a moving camera [28, 29], or by dense multi-view triangulation [18]. We refer to the detailed survey by Berger et al. [6] for creating clean 3D models from point clouds.

2.1. Statistical learning of 3D shapes

For reconstructing specific classes of shapes, such as human bodies or faces, fine-scale surface detail can be learned *e.g,* [7, 8, 19], from high resolution scans registered to a common mesh template model. This presumes a common shape topology or registration to a common template model, which is not possible for arbitrary shapes. Learning can also be performed from 3D scans sampled as voxels on a discrete voxel grid [26, 38]. Unfortunately, this method cannot preserve fine-scale surface detail without incurring significant cost of sampling. Fish et al. [16] learn from a data base of synthetic 3D meshes representing a family of shapes (*e.g,* chairs, airplanes etc.), identifying shape-parts

that are common across the family. An important problem in this regard is to model the compatibility of parts when synthesizing a new shape [3, 39]. Such works considering the global context of the shape are complementary to our approach. However, in contrast to such works, we do not assume synthetic meshes or group the shapes into a common family of similar shapes. Our method is applicable to a data set of dissimilar objects. These local patches may or may not be shared across objects - this analysis is performed automatically by our algorithm.

2.2. 3D global shape parameterization

Aligning a data set of 3D meshes to a common global surface parameterization is very challenging and requires the shapes to be of the same topology. For example, *geometry images* [33] can parameterize genus-0 shapes on a unit sphere, and even higher topology shapes with some distortion. Alternatively, the shapes can be aligned on the spectral distribution spanned by the Laplace-Beltrami Eigenfunctions [9, 25]. However, even small changes to the 3D mesh structure and topology can create large variations in the global spectral parameterization - something which cannot be avoided when dealing with real world 3D scans. Another problem is with learning partial scans and shape variations, where the shape detail is preserved only locally at certain places. Sumner and Popovic [36] proposed the *deformation gradient* encoding of a deforming surface through the individual geometric transformations of the mesh facets. This encoding can be used for statistical modeling of pre-registered 3D scans [27], and describes a Riemannian manifold structure with a Lie algebra [17]. All these methods assume that the shapes are pre-registered globally to a common mesh template, which is a significant challenge for shapes with arbitrary topologies. Another alternative is to embed a shape of arbitrary topology in a set of 3D cubes in the extrinsic space, known as *PolyCube-Maps* [37]. Unfortunately, this encoding is not robust to intrinsic deformations of the shape, such as bending and articulated deformations that can typically occur with real world shapes. So we choose an intrinsic quadrangular parameterization on the shape itself [13](see also Jakob et al. [20]).

2.3. 3D patch dictionaries

A lossy encoding of local shape detail can be obtained by 3D feature descriptors [21]. However, they typically do not provide a complete local surface parameterization. Recently, Digne et al. [12] used a 3D patch dictionary for point cloud compression. They encoded local surface patches as 2D height maps from a circular disc and learned a sparse linear dictionary of patch variations [2]. They obtain seed points to compute patches by performing random sampling, and encode the local reference frames by precise 3D translation and rotation from the origin. In contrast to this work,

(1) we do not assume knowledge of the local reference frames of the seed points (2) we parameterize larger patches oriented through geometric mesh quadrangulation (3) we query the dictionary using noisy patches with missing information and (4) we perform 3D surface inpainting to produce a full reconstruction of 3D mesh. Our method is inspired by the success in 2D image inpainting by patch synthesis [4], as well as efficient algorithms for learning sparse dictionaries [23, 30].

2.4. 3D surface inpainting

Earlier methods for 3D surface inpainting regularized from the geometric neighborhood [5, 22]. More recently, Sahay et al. [32] inpaint the holes in a shape by pre-registering it to a *self-similar* proxy model in a dataset, that broadly resembles the shape. The holes are inpainted using a patch-dictionary. In this paper, we use a similar approach, but avoid the assumption of finding and pre-registering to a proxy model. The term *self-similarity* in our paper refers to finding similar patches in other areas of the shape. Our method automatically detects the suitable patches, either from within the shape, or from a diverse data set of 3D models. Zhong et al. [40] propose an alternative learning approach by applying sparsity on the Laplacian Eigenbasis of the shape. We show that our method is better than this approach on publicly available meshes.

3. 3D Patch Encoding

Given a mesh $\mathcal{M}$ depicting a 3D shape and the input parameters - patch radius r and grid resolution N, our patch encoding produces a set of fixed length local representation units, the patch set $\{P_s\}$, along with the settings $\mathcal{S} = \{(s, T_s)\}$ - the location and orientation of the patches in the global shape. Here $\{s\}$ are the seed points - the points around which patches are computed, and $\{T_s\}$ is the set of transformation matrices for the reference frames RFs (or orientations) at the seed points $\{s\}$. The parameter r affects how the set of reference frames are computed. The points in the surfaces are then represented in the coordinate of RFs and are finally sampled in a rectangular grid of size $N \times N$ to get the patch representation $\{P_s\}$.

3.1. Patch computation on provided RFs

Points are densely and uniformly sampled in $\mathcal{M}$ to get a point cloud, C. Given a seed point s on the model surface C, reference frame $\mathcal{F}_s$ at s, and an input patch-radius r, we consider all the points in the r-neighbourhood, $\mathcal{P}_s$. Each point in $\mathcal{P}_s$ is represented w.r.t. $\mathcal{F}_s$ as $P_{\mathcal{F}_s}$. That is, if the rotation between global coordinates and $\mathcal{F}_s$ is given by the rotation matrix R_s, a point $\boldsymbol{p}$ represented in the local coordinate system of $\mathcal{F}_s$ is given by $\boldsymbol{p}_s = T_s\boldsymbol{p}$, where $T_s = [R \quad -R_s\boldsymbol{O}; 0 \quad 1]$ is the transformation matrix be-
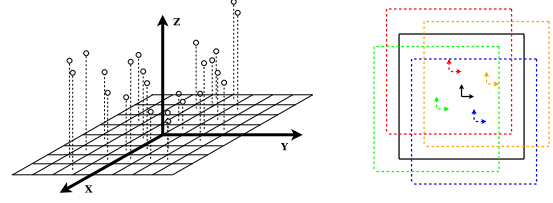

Figure 2: (Left) Patch representation - Points are sampled as a height map over the planer grid of a reference frame at the seed point. (Right) Patches computed at multiple offset from the quad centres to simulate dense sampling of patches while keeping the stable quad orientation. The black connected square represents the quad in a quad mesh and the dotted squares represents the patches that are computed at different offset.

tween the two coordinates, $\boldsymbol{O}$ the origin of $\mathcal{F}_s$ which is basically s for the patch at s.

3.2. Local parameterisation

An $N \times N$ square grid of length $\sqrt{2}r$ is placed on the X-Y plane of $\mathcal{F}_s$, and points in $P_{\mathcal{F}_s}$ are sampled over the grid *wrt* their X-Y coordinates. Each sampled point is then represented by its 'height' from the square grid, which is its Z coordinate to finally get a height-map representation of size N^2 (Figure 2). Thus, each patch around a point s is defined by a fixed size vector P_s of size N^2 and a transformation T_s giving the patch set $\{P_s\}$ and the settings $\{(s, T_s)\}$ for the entire shape.

The patch set can be processed by algorithms and be altered to $\{P'_s\}$. For example, the patch set computed from a noisy point cloud can be denoised using some algorithm to it denoised version $\{P'_s\}$. The altered or the original patches are converted to their global positions to reconstruct the shape by the following technique.

Reconstruction of Point Cloud: For each patch P_s in $\{((s, T_s), P_s)\}$, for each bin i, the height map representation $P_s[i]$, is first converted to the XYZ coordinates in its reference frame, $\boldsymbol{p}_s$, and then to the global coordinates $\boldsymbol{p}'$, by $\boldsymbol{p}' = T_s^{-1}\boldsymbol{p}_s$.

3.3. Reference frames from quad mesh

The height map based representation accurately encodes a surface only when the patch radius is below the distance between surface points and the shape medial axis. In other words, the r-neighbourhood, $\mathcal{P}_s$ should delimit a topological disk on the underlying surface to enable parameterization over the grid defined by the reference frame. In real world shapes, either this assumption breaks, or the patch radius becomes too low to have a meaningful sampling of shape description. A good choice of seed points enables the computation of the patches in well behaved areas, such that, even with moderately sized patches in arbitrary real world shapes, the r-neighbourhood, $\mathcal{P}_s$ of a given point s delimits

a topological disk on the grid of parameterisation. It should also provide an orientation consistent with global shape.

Given a mesh $\mathcal{M}$, we obtain low-resolution representation by Laplacian smoothing [35]. The low resolution mesh captures the broad outline of the shape over which local patches can be placed. In our experiments, for all the meshes, we performed 30 Laplacian smoothing iterations (normal smoothing + vertex fitting).

Given the smooth coarse mesh, the quad mesh $\mathcal{M}^Q$ is extracted following Jakob et al. [20]. At this step, the quad length is specified in proportion to the final patch length and hence the scale of the patch computation. For each quad q in the quad mesh, its center and $4 * k$ offsets are considered as seed points, where k is the overlap level (Figure 2 (Right)). These offsets capture more patch variations for the learning algorithm. For all these seed points, the reference frames are taken from the orientation of the quad q denoted by its transformation T_s. In this reference frame, Z axis, on which the height map is computed, is taken to be in the direction normal to the quad. The other two orthogonal axes - X and Y, are computed from the two consistent sides of the quads. To keep the orientation of X and Y axes consistent, we do a breath first traversal starting from a specific quad location in the quad mesh and reorient all the axes to the initial axes.

3.4. Patches from Connected Mesh

The reconstruction of a connected mesh given a set of 3D patches and the input mesh is non trivial. For this purpose we need to keep connectivity information - information required for computing edges for a final connected mesh. This can be done by either using the connection information form the original mesh $\mathcal{M}$ or a different mesh, but with similar topology. For the later case, we use the connection information in the subdivided quad mesh M^Q at the time of patch computation. In either case we will call the mesh from which the connectivity information is deduced as $M_r = \{V_r, F_r\}$.

In addition to the computation of the patch set $\{((s, T_s), P_s)\}$, the mapping $\{(j, \{(P_s, i)\})\}$ is computed, where each vertex index j s.t. $v_j \in V_r$, is mapped to a set of patches and its location bin $\{(P_s, i)\}$ where it would get sampled during the patch computation. This is found by following the patch computation procedure using the set V_r as the input point cloud and the $\{(s, T_s)\}$ as the reference frames. The final patch description of $\mathcal{M}$ is then described by the patch set $\{((s, T_s), P_s)\}$, the vertex index mapping $\{(j, \{(P_s, i)\})\}$ and the face connectivity F_r.

Reconstruction The goal here is to reconstruct a mesh $\mathcal{M}'$ given a patch set $\{((s, T_s), P_s)\}$, the vertex index mapping $\{(j, \{(P_s, i)\})\}$ and the face connectivity F_r.

First, points in the global frame are computed from the patch set $\{((s, T_s), P_s)\}$ following the reconstruction procedure of point clouds providing a global location v_e for

each bin i for each patch P_s. Then the estimate of each vertex $v_j \in V_r$ is given by the set of vertices $\{v_e\}$. The final value of vertex v'_j is taken as the mean of $\{v_e\}$. The reconstructed mesh is then given by $\{\{v'_j\}, F_r\}$. If the estimate of a vertex v_j is empty, we take the average of the vertices in its 1-neighbour ring.

4. Patch Dictionary and surface reconstruction

4.1. Dictionary Learning and Sparse Models

Given a matrix D in $\mathbb{R}^{m \times p}$ with p column vectors, sparse models in signal processing aims at representing a signal $\boldsymbol{x}$ in $\mathbb{R}^m$ as a sparse linear combination of the column vectors of D. The matrix D is called *dictionary* and its columns *atoms*. In terms of optimization, approximating $\boldsymbol{x}$ by a sparse linear combination of atoms can be formulated as finding a sparse vector $\boldsymbol{y}$ in $\mathbb{R}^p$, with k non-zero coefficients, that minimizes

$$\min_{\boldsymbol{y}} \frac{1}{2} \|\boldsymbol{x} - D\boldsymbol{y}\|_2^2 \qquad \text{s.t. } \|\boldsymbol{y}\|_0 \leq k \qquad (1)$$

The dictionary D can be learned or evaluated from the signal dataset itself which gives better performance over the off-the-shelf dictionaries in natural images. In this work we learn the dictionary from the 3D patches for the purpose of mesh processing. Given a dataset of n training signals $\boldsymbol{X} = [\boldsymbol{x}_1, ..., \boldsymbol{x}_n]$, dictionary learning can be formulated as the following minimization problem

$$\min_{D, \boldsymbol{Y}} \sum_{i=1}^{n} \frac{1}{2} \|\boldsymbol{x}_i - D\boldsymbol{y}_i\|_2^2 + \lambda \psi(\boldsymbol{y}_i), \qquad (2)$$

where $\boldsymbol{Y} = [\boldsymbol{y}_1, ..., \boldsymbol{y}_n] \in \mathbb{R}^{p \times n}$ is the set of sparse decomposition coefficients of the input signals $\boldsymbol{X}$, ψ is sparsity inducing regularization function, which is often the l_1 or l_0 norm.

Both optimization problems described by equations 1 and 2 are solved by approximate or greedy algorithms; for example, Orthogonal Matching Pursuit (OMP) [30], Least Angle Regression (LARS) [14] for sparse encoding (optimization of Equation 1) and KSVD [2] for dictionary learning (optimization of Equation 2)

Missing Data: Missing information in the original signal can be well handled by the sparse encoding. To deal with unobserved information, the sparse encoding formulation of Equation 1 can be modified by introducing a binary mask M for each signal $\boldsymbol{x}$. Formally, M is defined as a diagonal matrix in $\mathbb{R}^{m \times m}$ whose value on the j-th entry of the diagonal is 1 if the pixel $\boldsymbol{x}$ is observed and 0 otherwise. Then the sparse encoding formulation becomes

$$\min_{\boldsymbol{y}} \frac{1}{2} \|M(\boldsymbol{x} - D\boldsymbol{y})\|_2^2 \qquad \text{s.t. } \|\boldsymbol{y}\|_0 \leq k \qquad (3)$$

Here $M\boldsymbol{x}$ represents the observed data of the signal $\boldsymbol{x}$ and $D\boldsymbol{y}$ is the estimate of the full signal. The binary mask does not drastically change the optimization procedure and one can still use the classical optimization techniques for sparse encoding.

4.2. 3D Reconstruction from Patch Dictionary

In this work we learn patch dictionary D with the generated patch set $\{P_s\}$ as training signals. This patch set may come from a single mesh (providing *local dictionary*), or be accumulated globally using patches coming from different shapes (providing a *global dictionary* of the dataset). Also in the case of the application of hole-filling, a dictionary can be learnt on the patches from clean part of the mesh, which we call *self-similar* dictionary which are powerful in meshes with repetitive structures. For example a tiled floor, or the side of a shoe has many repetitive elements that can be learned automatically.

Reconstruction of the 3D shape Using a given patch dictionary D, we can reconstruct the original shape whose accuracy depends on the number of atoms chosen for the dictionary. For each 3D patch P_i from the generated patches and the learnt dictionary D of a shape, its sparse representation, y is found following the optimization in Equation 1 using the algorithm of Orthogonal Matching Pursuit (OMP). It's approximate representation, the locally reconstructed patch P'_i is found as $P'_i \approx Dy$. The final reconstruction is performed using the altered patch set $\{P'_i\}$ and $\mathcal{S}$ following the procedure in Section 3.4.

4.3. Inpainting and hole filling of 3D surfaces

4.3.1 Surface reconstruction with missing data

In case of 3D mesh with missing data, for each 3D patch x_i computed from the noisy data having missing values, we find the sparse encoding y_i following Equation 3. The estimate of the full reconstructed patch is then $x' = Dy$.

When the underlying mesh connection is known in the noisy data, we then perform the global reconstruction from the valid patches together with the estimated patches $\{x'\}$ and the connectivity of the subdivided quad mesh to get the final inpainted shape (Section 3.4). This assumption of known mesh connection is true for applications like as repairing damaged surface regions which is quite common in underexposed areas of an object while performing a 3D reconstruction pipeline. Also it is assumed that the size of the masked or damaged region is less than the patch size or the average quads length. The selection of stable reference frames in our patch computation framework allows us to compute moderate length patches and inpaint continuous regions of moderately size.

4.3.2 Filling holes in a noisy shape

For the application of hole filling, the inpainting regions are completely empty and has no edge connectivity/missing vertices information. Thus, to establish the interior mesh connectivity there has to be a way of inserting vertices and performing triangulation before the application of the in-
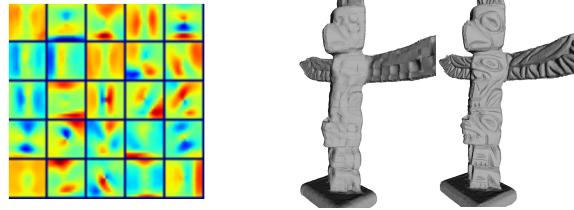

Figure 3: (Left) Visualization of dictionary atoms learnt from the shape *Totem* ($m = 16 \times 16$). (Right) Reconstruction of the shape *Totem* using local dictionary of size 5 atoms and 100 atoms

painting method stated in the previous section. We use an existing popular method, namely [22], for this purpose of hole triangulation to get a connected hole filled mesh based on local geometry. This hole-triangulated mesh is used for inferring missing vertex connectivity as well as quad mesh computation on the mesh with holes at the time of testing.

Using the above reference frames from the computed quad mesh, we compute the patches from the shape with holes and missing vertices. Patches which overlap with the holes will contain missing information and we reconstruct them as explained in the above section (Section 4.3.1).

5. Experimental Results

5.1. Dataset

We considered dataset having 3D shapes of two different types. The first type (**Type 1**) consists of meshes that are in general, simple in nature without much surface texture. In descending order of complexity 5 such objects considered are - *Totem, Bunny, Milk-bottle, Fandisk* and *Baseball*. *Totem* is of very high complexity containing a high amount of fine level details whereas *Bunny* and *Fandisk* are some standard graphics models with moderate complexity. In addition, we considered 5 models with high surface texture and details (**Type 2**) consisting of shoe soles and a human brain specifically to evaluate our hole-filling algorithm - *Supernova, Terrex, Wander, LeatherShoe* and *Brain*. Therefore, we consider in total 10 different meshes for our evaluation (all meshes are shown in the supplementary material).

Other than the models *Baseball, Fandisk* and *Brain*, all models considered for the experimentation are reconstructed using a vision based reconstruction system - 3Digify [1]. Since this system uses structured light, the output models are quite accurate, but do have inherent noise coming from structured light reconstruction and alignments. Nonetheless, because of its high accuracy, we consider these meshes to be 'clean' for computing global patch database. These models were also reconstructed with varying accuracy by changing the reconstruction environment before considering for the experiments of inpainting.

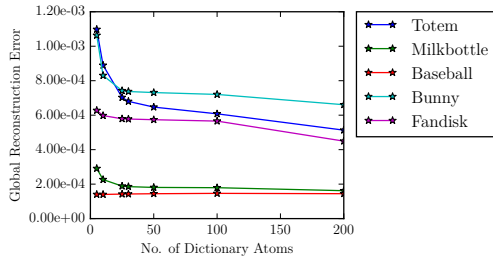**Dataset normalization and scale selection** For normalization, we put each mesh into a unit cube followed by up-

Figure 4: Reconstruction error of different shapes with Dictionaries with increasing number of atoms.

Meshes	Mesh entities	#patches	Patch entities	Compr factor	PSNR
Totem	450006	658	12484	36.0	56.6
Milkbottle	441591	758	14420	30.6	72.3
Baseball	415446	787	14974	27.7	75.6
Bunny	501144	844	16030	31.3	60.6
Fandisk	65049	874	16642	3.9	62.1

Table 1: Results for compression in terms of number of entities with a representation with global dictionary of 100 atoms. Mesh entities consists of the number of entities for representing the mesh which is: $3 \times$ #Faces and #Vertices. Patch entities consists of the total number of sparce dictionary coefficients (20 per patch) used to represent the mesh plus the entities in the quad mesh. Compr factor is the compression factor between the two representation. PSNR is Peek Signal to Noise Ratio where the bounding box diameter of the mesh is considered as the peek signal following [31].
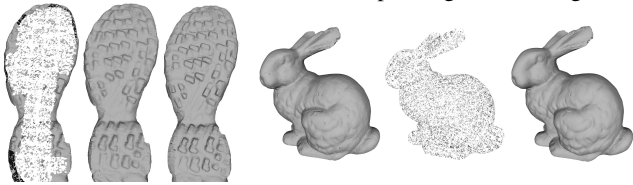

Figure 5: Inpainting of the models with 50% missing vertices (Left - noisy mesh, Middle - inpainted mesh, Right - ground truth) of *Terrex* and *Bunny*, using the local dictionary. Here we use the quad mesh provided at the testing time.

Missing Ratio	0.2		0.5	
	ours	[40]	ours	[40]
bunny	**1.11e-3**	1.90e-2	**1.62e-3**	2.20e-2
fandisk	**1.32e-3**	8.30e-3	**1.34e-3**	1.20e-2

Table 2: RMS Inpainting error of missing vertices from our method using local dictionary and its comparison to [40].

Potential for Compression The reconstruction error is low after a certain number of atoms in the learned dictionary, even when global dictionary is used for reconstructing all the shapes (more on shape independence in Section 5.4). Thus, only the sparse coefficients and the connectivity information needs to be stored for a representation of a mesh using a common global dictionary, and can be used as a means of mesh compression. Table 1 shows the results of information compression on Type 1 dataset.

5.3. Surface Inpainting

Recovering missing geometry To evaluate our algorithm for geometry recovery, we randomly label certain percentage of vertices in the mesh as missing. The reconstructed vertices are then compared with the original ones. The visualization of our result is in Figure 5. Zoomed view highlighting the details captured as well as the results from other objects are provided in the supplementary material. We compare our results with [40] which performs similar task of estimating missing vertices, with the publically available
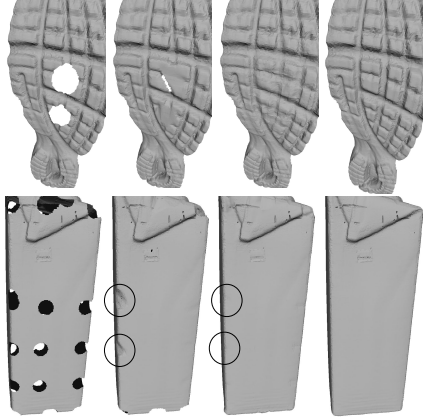
sampling (by subdivision) or downsampling (by edge collapse) to bring it to a common resolution. After normalization, we obtained the low resolution mesh by applying Laplacian smoothing with 30 iterations. We then performed the automatic quadiangulation procedure of [13] on the low resolution mesh, with the targeted number of faces such that, it results an average quad length to be 0.03 for Type 1 dataset and 0.06 for Type 2 dataset (for larger holes); which in turns become the average patch length of our dataset. The procedure of smoothing and generating quad mesh can be supervised manually in order to get better quad mesh for reference frame computation. But, in our varied dataset, the automatic procedure gave us the desired results.

We then generated 3D patches from each of the clean meshes using the procedure provided in Section 3. We chose the number of bins N, to be 16 for Type 1 dataset and 24 for Type 2 dataset; to match the resolution the input mesh. To perform experiment in a common space (global dictionary), we also generated patches with patch dimension of 16 in Type 2 dataset with the loss of some output resolution.

5.2. Dictionary Learning and Mesh Reconstruction

Dictionary Learning We learn the local dictionary for each shape with varying numbers of dictionary atoms with the aim to reconstruct the shape with varying details. Atoms of one such learned dictionary is shown in Figure 3 (Left). Observe the 'stripe like' structures the dictionary of *Totem* in accordance to the fact that the *Totem* has more line like geometric textures.

Reconstruction of shapes We then perform reconstruction of the original shape using the local dictionaries with different number of atoms (Section 4.2). Figure 3 (Right) shows qualitatively the difference in output shape when reconstructed with dictionary with 5 and 100 atoms. Figure 4 shows the plot between the *Global Reconstruction Error* - the mean Point to Mesh distance of the vertices of the reconstructed mesh and the reference mesh - and the number of atoms in the learned dictionary for our Type 1 dataset. We note that the reconstruction error saturates after a certain number of atoms (50 for all).

Figure 6: Qualitative analysis of the inpainting algorithm of *Super-nova* and *Milk-bottle*. From left to right - mesh with holes, hole filling with [22], our results from global dictionary and ground truth mesh. Detailed visualization of the results of other meshes are presented in the provided supplementary material.

meshes *Bunny* and *Fandisk*, and provide the recovery error measured as the Root Mean Square Error (RMSE) of the missing coordinates in Table 2. As seen in the table, we improve over them by a large margin.

This experiment also covers the case when the coarse mesh of the noisy data is provided to us which we can directly use for computing quad mesh and infer the final mesh connectivity (Section 3.4). This is true for the application of recovering damaged part. If the coarse mesh is not provided, we can easily perform poisson surface reconstruction using the non-missing vertices followed by Laplacian smoothing to get our low resolution mesh for quadriangulation. Since the low resolution mesh is needed just for the shape outline without any details, poisson surface reconstruction does sufficiently well even when 70% of the vertices are missing in our meshes.

Hole filling We systematically punched holes of different size (limiting to the patch length) uniform distance apart in the models of our dataset to create noisy test dataset. We follow the procedure in Section 4.3.2 in this noisy dataset and report our inpainting results in Table 3. Here we use mean of the Cloud-to-Mesh error of the inpainted vertices as our error metrics. Please note that the noisy patches are independently generated on its own quad mesh. No information about the reference frames from the training data is used for patch computation of the noisy data. Also, note that this logically covers the inpainting of the missing geometry of a scan due to occlusions. We use both local and global dictionaries for filling in the missing information and found our results to be quite similar to each other.

For baseline comparison we computed the error from the popular filling algorithm of [22] available in MeshLab [11]. Here the comparison is to point out the improvement achieved using a data driven approach over geometry. We
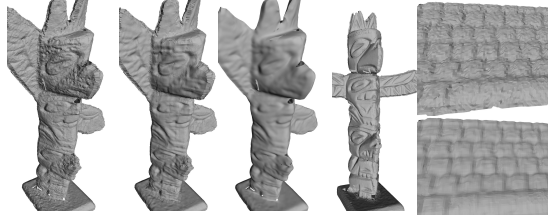

Figure 7: Denoising meshes using a clean patch dictionary of a similar object. (Left) Results on *Totem* (from left to right) - noisy reconstruction from SFM, our denoising using patch dictionary from a clean reconstruction, denoising by Laplacian smoothing [35], the high quality clean mesh with different global configuration. (Right) Result for the mesh *Keyboard* with the same experiment. Zoomed versions of similar results are provided in the supplementary material.

could not compare our results with [40] because of the lack of systematic evaluation of hole-filling in their paper. As it is seen, our method is clearly better compared to the [22] quantitatively and qualitatively (Figure 6). The focus of our evaluation here is on the Type 2 dataset - which captures complex textures. In this particular dataset we also performed the hole filling procedure using self-similarity, where we learn a dictionary from the patches computed on the noisy mesh having holes, and use it to reconstruct the missing data. The results obtained is very similar to the use of local or global dictionary (Table 4).

In fact with smaller holes, the method of [22] performs as good as our algorithm, as shape information is not present in such a small scale. The performance of our algorithms becomes noticeably better as the hole size increases as described by Figure 8a. This shows the advantage of our method for moderately sized holes.

Improving quality of noisy reconstruction Our algorithm for inpainting can be easily extended for the purpose of denoising. We can use the dictionary learned on the patches from a clean or high quality reconstruction of an object to improve the quality of its low quality reconstruction. Here we approximate the noisy patch with its closest linear combination in the Dictionary following Equation 1. Because of the fact that our patches are local, the low quality reconstruction need not be globally similar to the clean shape. This is depicted by the Figure 7 (Left) where a different configuration of the model *Totem* (with the wings turned compared to the horizontal position in its clean counterpart) reconstructed with structure-from-motion with noisy bumps has been denoised by using the patch dictionary learnt on its clean version reconstructed by Structured Light. A similar result on Keyboard is shown in Figure 7 (Right).

5.4. Global dictionary and shape independence

We perform reconstruction of *Totem* using both the local dictionary and global dictionary having different number of
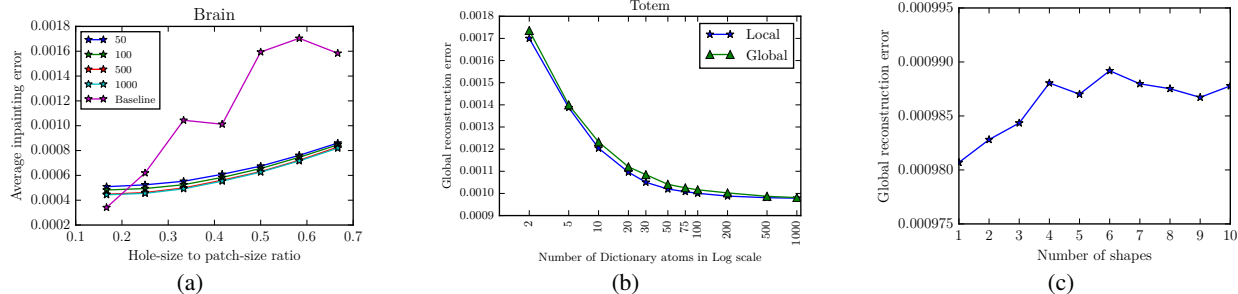
Figure 8: (a) Inpainting error vs Hole-size to patch-size ratio for *Brain* inpainted using the global dictionary. The patchsize here is 0.062 (patch radius ≈ 0.044). Plots of other shapes are in provided in the supplementary material (b) Comparison of the reconstruction error of *Totem* using local and global dictionaries with different number of atoms. For better visualization the X axis provided in logarithmic scale. (c) Reconstruction error of *Totem* with global dictionaries (with 500 atoms) having patches from different number of shapes.

	[22]	Our - Local	Our - Global
Supernova	0.001646	**0.000499**	0.000524
Terrex	0.001258	0.000595	**0.000575**
Wander	0.002214	0.000948	**0.000901**
LeatherShoe	0.000854	0.000569	**0.000532**
Brain	0.002273	0.000646	**0.000587**
Milk-bottle	0.000327	0.000126	**0.000123**
Baseball	0.000158	**0.000138**	0.000168
Totem	0.001065	0.001065	**0.001052**
Bunny	**0.000551**	0.000576	0.000569
Fandisk	0.001667	0.000654	**0.000634**

Table 3: Mean inpainting error for our dataset of hole size 0.015, 0.025 and 0.035 for the dataset Type 2 (top block of the table) and 0.01 and 0.02 for Dataset Type 1 (bottom block of the table). *Local* uses the local dictionary learned from the clean mesh of the corresponding shape and *Global* uses a global dictionary learned from the entire dataset.

	[22]	Self-Similar
Supernova	0.001162	**0.000401**
Terrex	0.000900	**0.000585**
Wander	0.001373	**0.000959**
LeatherShoe	0.000596	**0.000544**
Brain	0.001704	**0.000614**

Table 4: Mean inpainting error comparison with self similar dictionary with 100 atoms. Hole size considered is 0.035

atoms to know if the reconstruction error, or the shape information encoded by the dictionary, is dependent on where the patches come from at the time of training. We observed that when the number of dictionary atoms is sufficiently large (200 - 500), the global dictionary performs as good as the local dictionary (Figure 8b). This is also supported by our superior performance of global dictionary in therms of hole filling.

Keeping the number of atoms fixed at which the performances between Local and Global dictionary becomes indistinguishable (500 in our combined dataset), we learned

global dictionary using the patches from different shapes, with one shape at a time. The reconstruction error of *Totem* using these global dictionary varied very little. But we notice a steady increase in the reconstruction error with increase in the number of object used for learning; which becomes steady after a certain number of object. After that point (6 objects), adding more shapes for learning does not create any difference in the final reconstruction error (Figure 8c). This verifies our hypothesis that the reconstruction quality does not deteriorate significantly with increase in the size of the dataset for learning.

6. Conclusion

In this paper, we proposed a new method for encoding 3D surface of arbitrary shapes using rectangular local patches, such that shape variations can be learned into a dictionary. We performed quantitative analysis of our method on real world 3D scans towards faithful 3D reconstruction and inpainting of holes. Our experimental results indicate that learning can be performed on a data set of diverse real world objects assembled into a common pool. We tested the paradigm of sparse linear models, such as patch dictionaries, to encode the variation in the data set. A principal contribution of our paper is a pipeline for preprocessing the 3D shapes: scaling, mesh quadrangulation, and aligning the patch scan-lines. This pre-processing can be followed for assembling a large data set of real world 3D shapes, similar to the scaling and centring of 2D images in public data sets. In this paper, we showed that this learning can be done without imposing a common mesh template, topology, or class of shapes. However, our method is still not capable of handling fine-scaled topological detail, such as with wires or hair. We leave this problem towards future work.

7. Acknowledgements

References

[1] 3Digify. 3digify, http://3digify.com/, 2015.

[2] M. Aharon, M. Elad, and A. Bruckstein. K -svd: An algorithm for designing overcomplete dictionaries for sparse representation. *Signal Processing, IEEE Transactions on*, 54(11):4311–4322, Nov 2006.

[3] M. Averikou, V. Kim, Y. Zheng, and Mitra. Shapesynth: Parameterizing model collections for coupled shape exploration and synthesis. *Computer Graphics Forum (Pr*, 2014.

[4] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman. PatchMatch: A randomized correspondence algorithm for structural image editing. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 28(3), Aug. 2009.

[5] G. H. Bendels, M. Guthe, and R. Klein. Free-form modelling for surface inpainting. In *Proceedings of the 4th International Conference on Computer Graphics, Virtual Reality, Visualisation and Interaction in Africa*, AFRIGRAPH '06, pages 49–58, New York, NY, USA, 2006. ACM.

[6] M. Berger, A. Tagliasacchi, L. Seversky, P. Alliez, J. Levine, A. Sharf, and C. Silva. State of the art in surface reconstruction from point clouds. *Eurographics: State of the Art Reports*, pages 161–185, 2014.

[7] A. H. Bermano, D. Bradley, T. Beeler, F. Zund, D. Nowrouzezahrai, I. Baran, H. Sorkine-Hornung, O.and Pfister, R. W. Sumner, B. Bickel, and Gross. Facial performance enhancement using dynamic shape space analysis. *ACM Transactions on Graphics*, 33(2):13:1–13:12, 2014.

[8] F. Bogo, M. J. Black, M. Loper, and J. Romero. Detailed full-body rreconstruction of moving people from monocular rgb-d sequences. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 2300–2308, 2015.

[9] D. Boscaini, J. Masci, E. Rodola, and M. M. Bronstein. Learning shape correspondence with anisotropic convolutional neural networks. In *Proceedings of the Neural Information Processing Systems (NIPS)*, number arXiv:1605.06437, 2016.

[10] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu. ShapeNet: An Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015.

[11] P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, and G. Ranzuglia. Meshlab: an open-source mesh processing tool. In *Eurographics Italian Chapter Conference*, volume 2008, pages 129–136, 2008.

[12] J. Digne, R. Chaine, and S. Valette. Self-similarity for accurate compression of point sampled surfaces. *Computer Graphics Forum*, 33(2):155–164, 2014.

[13] H.-C. Ebke, D. Bommes, M. Campen, and L. Kobbelt. QEx: Robust quad mesh extraction. *ACM Transactions on Graphics*, 32(6):168:1–168:10, 2013.

[14] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *Ann. Statist.*, 32(2):407–499, 04 2004.

[15] M. Firman. List of rgb-d data sets. http://www0.cs.ucl.ac.uk/staff/M.Firman/RGBDdatasets/.

[16] N. Fish, M. Averikou, O. Van Kaick, O. Sorkine-Hornung, C.-O. D., and N. Mitra. Meta-representation of shape families. *ACM Transactions on Graphics (P*, 2014.

[17] O. Freifeld and M. J. Black. Lie bodies: A manifold representation of 3d human shape. In *Proceedings of the 12th European Conference on Computer Vision - Volume Part I*, ECCV'12, pages 1–14, Berlin, Heidelberg, 2012. Springer-Verlag.

[18] Y. Furukawa and J. Ponce. Accurate, dense, and robust multiview stereopsis. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(8):1362–1376, Aug 2010.

[19] P. Garrido, M. Zollhoefer, D. Casas, L. Valgaerts, K. Varanasi, P. Perez, and C. Theobalt. Reconstruction of personalized 3d face rigs from monocular video. *ACM Transactions on Graphics*, 35(3):28:1–28:15, 2016.

[20] W. Jakob, M. Tarini, D. Panozzo, and O. Sorkine-Hornung. Instant field-aligned meshes. *ACM Trans. Graph.*, 34(6):189:1–189:15, Oct. 2015.

[21] H. Kim and A. Hilton. Evaluation of 3d feature descriptors for multi-modal data registration. In *3D Vision - 3DV 2013, 2013 International Conference on*, pages 119–126, June 2013.

[22] P. Liepa. Filling holes in meshes. In *Proceedings of the 2003 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, SGP '03, pages 200–205, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[23] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online dictionary learning for sparse coding. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 689–696, New York, NY, USA, 2009. ACM.

[24] D. Marr and H. K. Nishihara. Representation and recognition of the spatial organization of three-dimensional shapes. *Proceedings of the Royal Society of London, Series B, Biological Sciences*, 200(1140):269–294, 1978.

[25] J. Masci, D. Boscaini, M. M. Bronstein, and P. Vandergheynst. Shapenet: Convolutional neural networks on non-euclidean manifolds. (arXiv:1501.06297), 2015.

[26] D. Maturana and S. Scherer. VoxNet: A 3D Convolutional Neural Network for Real-Time Object Recognition. In *IROS*, 2015.

[27] T. Neumann, K. Varanasi, N. Hasler, M. Wacker, M. Magnor, and C. Theobalt. Capture and statistical modeling of arm-muscle deformations. In *Computer Graphics Forum*, volume 32, pages 285–294. Blackwell Publishing Ltd, 2013.

[28] R. A. Newcombe, S. Izadi, O. Hiliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *Proceedings of the 10th IEEE International Symposium on Mixed and Augmented Reality (IS-MAR)*, 2011.

[29] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison. Dtam: Dense tracking and mapping in real-time. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 2320–2327, 2011.

[30] Y. C. Pati, R. Rezaiifar, and P. S. Krishnaprasad. Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition. In *Signals, Systems and*

Computers, 1993. 1993 Conference Record of The Twenty-Seventh Asilomar Conference on, pages 40–44 vol.1, Nov 1993.

[31] E. Praun and H. Hoppe. Spherical parametrization and remeshing. *ACM Trans. Graph.*, 22(3):340–349, July 2003.

[32] P. Sahay and A. Rajagopalan. Geometric inpainting of 3d structures. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–7, 2015.

[33] A. Sinha, J. Bai, and K. Ramani. Deep learning 3d shape surfaces using geometry images. In *Proceedings of 15th European Conference on Computer Vision (ECCV)*, pages 223–240, 2016.

[34] Song.S, S. P. Lichtenberg, and J. Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 567–576, 2015.

[35] O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rössl, and H.-P. Seidel. Laplacian surface editing. In *Proceedings of the EUROGRAPHICS/ACM SIGGRAPH Symposium on Geometry Processing*, pages 179–188. ACM Press, 2004.

[36] R. W. Sumner and J. Popović. Deformation transfer for triangle meshes. *ACM Trans. Graph.*, 23(3):399–405, Aug. 2004.

[37] M. Tarini, K. Hormann, P. Cigoni, and C. Montani. Polycube-maps. *ACM Trans. Graph.*, 2004.

[38] Z. Wu, S. Song, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shape modeling. In *Proceedings of 28th IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[39] Y. Zheng, C.-O. D., and N. J. Mitra. Smart variations: Functional substructures for part compatibility. *Computer Graphics Forum (Proc*, 2013.

[40] M. Zhong and H. Qin. Surface inpainting with sparsity constraints. *Computer Aided Geometric Design*, 41:23 – 35, 2016.