Vis Comput (2016) 32:625–640
DOI 10.1007/s00371-015-1078-y

ORIGINAL ARTICLE

A robust hybrid image-based modeling system

Hoang Minh Nguyen[1] · **Burkhard Wünsche**[1] · **Patrice Delmas**[1] · **Christof Lutteroth**[1] · **Eugene Zhang**[2]

Published online: 21 April 2015

Abstract This paper presents a new robust image-based modeling system for creating high-quality 3D models of complex objects from a sequence of unconstrained photographs. The images can be acquired by a video camera or hand-held digital camera without the need of camera calibration. In contrast to previous methods, we integrate correspondence-based and silhouette-based approaches, which significantly enhances the reconstruction of objects with few visual features (e.g., uni-colored objects) and improves surface smoothness. Our solution uses a mesh segmentation and charting approach in order to create a low-distortion mesh parameterization suitable for objects of arbitrary genus. A high-quality texture is produced by first parameterizing the reconstructed objects using a segmentation and charting approach, projecting suitable sections of input images onto the model, and combining them using a graph-cut technique. Holes in the texture due to surface patches without projecting input images are filled using a novel exemplar-based inpainting method which exploits appearance space attributes to improve patch search, and blends patches using Poisson-guided interpolation. We analyzed the effect of different algorithm parameters, and compared our system with a laser scanning-based reconstruction and existing commercial systems. Our results indicate that our system is robust, superior to other image-based modeling techniques, and can achieve a reconstruction quality visually not discernible from that of a laser scanner.

Keywords 3D shape recovery · Texture acquisition · Surface reconstruction · Surface parameterization · Texture inpainting

✉ Hoang Minh Nguyen
hngu039@aucklanduni.ac.nz; justin.nguyen@auckland.ac.nz

Burkhard Wünsche
burkhard@cs.auckland.ac.nz

Patrice Delmas
p.delmas@auckland.ac.nz

Christof Lutteroth
lutteroth@cs.auckland.ac.nz

Eugene Zhang
zhange@eecs.oregonstate.edu

[1] University of Auckland, Auckland, New Zealand

[2] Oregon State University, Corvallis, USA

1 Introduction

Although there has been much interest and study of 3D modeling problems over the past decades, robustly and automatically obtaining 3D models remains a difficult task. For the past 30 years, creation of 3D models using conventional graphics software such as Maya or 3D Max has continued to be the most popular approach. The reason for this is that alternative approaches either only work for a small range of objects, require special hardware, require post-processing, or special set-ups such as calibrated cameras. The introduction of specialized hardware such as laser scanners has made it possible for inexperienced users to construct 3D digital models from real physical objects. Although such specialized machinery can match classic 3D construction and cloning tools in terms of quality, they are very costly and highly selective with regards to the size and types of objects they can process (e.g., no shiny, reflective or dark surfaces).

Recovering 3D structure from photographic images is an efficient and intuitive way to create 3D digital models of real-world objects. Optical sensors (cameras) are ubiquitous (e.g., modern smartphones and tablet computers), have a high

resolution, and in contrast to other sensors (laser, structured lighting, PMD) work over a wide distance range. This paper presents a novel hybrid-based 3D construction approach that accepts a sequence of uncalibrated and unannotated images as input and automatically produces a high-quality 3D texture model. Our method does not require any a priori or supplementary information about the observed scene or the camera being used.

Reconstructing 3D scenes from a collection of photographs requires knowing where each photo was acquired. To do so, our algorithm automatically estimates the intrinsic and extrinsic parameters of the camera for each acquisition. It computes the 3D coordinates of a sparse set of points in the scene using *Structure-from-Motion* and *Bundle Adjustment* techniques. In order to handle feature-poor objects, additional 3D points are extracted and added by exploiting the silhouette information of the object. The benefit of integrating shape-from-silhouette and shape-from-correspondence approaches is that the new hybrid system handles both featureless objects and objects with concave regions. These classes of objects often pose great difficulty for algorithms using only a single approach. As the result, our solution is able to produce satisfactory reconstructions for a much larger class of objects. The obtained 3D mesh is then partitioned using a feature-based parameterization. A texture atlas is composed of the final model.

The contribution lies in the integration of various previous works. However, there are also several distinct improvements in the algorithm design as compared to related work. First, to improve the efficiency of our method, several images are added at the same time (instead of one by one) into the optimization process during the *Structure-from-Motion* stage. Second, instead of manually labeling the foreground and using marching square and Delaunay triangulation techniques to extract silhouettes, we employ the previously obtained scene geometry for this segmentation task, eliminating the need for manual segmentation processes. Third, our approach is the first that integrates an inpainting technique into 2D texture map generation. In our previous work (e.g., [1]), the inpainting technique was not integrated into the reconstruction process. The other aspect of our contribution is the comprehensive evaluation of the system. We assessed the performance through a range of different input parameters (effect of the number of input images, effect of image resolution, effect of scene illumination, effect of image distortion, and effect of the number of distinct features). This has never been explored before to that extent and provides a better insight into how these parameters influence image-based modeling methods in general. In our previous work (e.g., [2]), only two parameters (effect of the number of input images and effect of image resolution) were investigated on a much smaller scale. Additionally, the results were only visually compared.

The remainder of this chapter is organized as follows. After a description of the related work on image-based modeling, a discussion of our algorithms is presented in Sect. 3. Results are discussed in Sect. 4. Section 5 concludes and summarizes the paper and gives a brief outlook on directions for future research.

2 Literature survey

Image-based modeling algorithms can be categorized depending on the visual cues employed to perform reconstruction, i.e., silhouettes, texture, transparency, defocus, shading, or correspondence. Traditionally, the most well-known and successful visual cues have been shading, silhouettes, and correspondence [3]. Silhouettes and correspondence offer the highest degree of robustness due to their invariance to illumination changes. The shading cue requires more control over the illumination environment, but can produce excellent results. However, the requirement for strict constraints over lighting conditions renders shape-from-shading impractical for general applications.

2.1 Silhouette-based reconstruction

The shape-from-silhouette class of algorithms is very efficient and has been proven to be stable with regard to object surface properties (color, texture, and material). It is, however, very limited in the object geometries it can handle [4–6]. The earliest attempt of using silhouettes for 3D shape reconstruction was by Baumgart in 1974. In his pioneering work [7], Baumgart exploited silhouette information from four input images to compute the 3D shapes of a baby doll and a toy horse. Following Baumgart's work, many different variations of the shape-from-silhouette paradigm have been proposed.

Grauman et al. [8] presented a Bayesian approach to account and compensate for errors introduced as the result of false segmentation. The approach has been shown to produce excellent error-compensated models from erroneous silhouette information. The disadvantage of this method is that it requires prior knowledge about the objects to be reconstructed and large ground-truth training data. This renders this method impractical as such data are not often available.

Cheung et al. [9,10] proposed a method that aligns multiple silhouette images of a non-rigidly moving object over time in an attempt to improve the quality of the constructed visual hull. Their method showed a significant improvement in reconstruction quality over previous methods.

Matusik et al. [5] exploited the epipolar constraints to improve the overall performance of the visual hull reconstruction process. Instead of constructing the visual hull of a scene by applying a series of 3D constructive solid

geometry intersections of these viewing cones, it is created by projecting each viewing cone onto other image planes and computing the intersection in two-dimensional space. The intersection result is then lifted to three-dimensional space using a lifting procedure. This effectively decreases the dimensionality of the intersection computations from three to two dimensions and therefore increases the overall performance of the system. Franco et al. [11] also took advantages of the epipolar constraints to improve the overall performance of their reconstruction approach. However, instead of lifting the entire intersected polygonal faces to three-dimensional space to create faces of the visual hull, the authors only raise endpoints of these intersected polygons to form a 3D point cloud covering the object's surfaces. Local orientation and connectivity rules are then employed to create a watertight model.

2.2 Correspondence-based reconstruction

Correspondence-based reconstruction techniques extract and match information from overlapping images in order to estimate sensor parameters and depth information.

One of the most famous and successful reconstruction systems is the Façade system, which was proposed by Debevec et al. [12]. It was designed to model and render simple architectural scenes by combining a hybrid geometric and image-based approach. The system requires only a few images and some known geometric parameters. It was used to reconstruct compelling fly-throughs of the Berkeley campus and was employed for the MIT City Scanning Project, which captured thousands of calibrated images from an instrumented rig to compute a 3D model of the MIT campus. While the resultant 3D models are often impressive, the system requires considerable time and effort from the user to decompose the scene into prismatic blocks and manually select features and their correspondence in different views, followed by the estimation of the pose of these primitives. Consequently, the system is impractical for reconstructing large scenes.

Quan et al. [13] presented a method for modeling plants. Segmentation is performed in both image space (by manually selecting areas in input images) and in 3D space. Using the segmented images and 3D data, the geometry of each leaf is recovered by fitting a deformable leaf model. Users are required to provide hints on segmentation. The main disadvantage of this method is that it requires full coverage of the observed model (360° capture), which may not always be possible for outdoor trees. Branches are interactively modeled through a simple user interface.

Brown et al. [14] presented an image-based modeling system that aims to recover camera parameters, pose estimates, and sparse 3D scene geometry from a sequence of images. Snavely et al. [15] introduced the *Photo Tourism* (*Photosynth*) system which is based on the work of Brown, with some significant modifications to improve scalability and robustness. Although these approaches address the same *Structure-from-Motion* concepts as our approach, their aim is not to reconstruct and visualize 3D scenes and models from images, but only to allow easy navigation between images in three dimensions.

3 3D reconstruction

Our 3D reconstruction process begins with distinctive features being extracted from input images and point correspondences being established in image pairs. We then isolate all matching images, selecting those that view a common subset of 3D points. Given a set of matching images, a scene geometry or point clouds and camera parameters can be estimated simultaneously by *Structure-from-Motion* (SfM) and subsequently refined by Bundle Adjustment. To enhance the density of the point clouds, additional points are generated and added by exploiting the object's silhouette information, yielding quasi-dense point clouds that cover the object surfaces. Surfaces are generated and applied onto the point clouds to produce a 3D model. The 3D mesh model is then parameterized, and a texture atlas is constructed by back-projecting the best fitting input images onto each surface segment, and smoothly fusing them together over the corresponding chart using graph-cut techniques. Missing textures are automatically generated using an inpainting technique. The outcome of this process is a complete and comprehensive 3D representation of the observed scene.

3.1 Camera parameter estimation

The camera parameter estimation process begins with salient features being extracted from input images using *SIFT* feature detector [16,17]. The principle idea behind *SIFT* is that it detects distinctive local features in a given image and describes them with vectors called SIFT feature descriptors. SIFT feature descriptors are invariant to image transformation, and partially invariant to illumination changes, noise, and camera viewpoint. Additionally, these features retain rich information content, which makes it possible to correctly match any single feature against a large database of features with high certainty.

Once all distinctive features have been extracted from the input images, they are matched across images, and changes of their relative position across multiple images are used to estimate camera parameters for each image and 3D coordinates of the matched points. The most common way to establish correspondence of features between images utilizes the feature descriptors obtained from the SIFT detection phase. Features from two images with the most similar feature

descriptors are paired as correspondences. This is achieved by considering the Euclidean distances between each feature of the candidate image and corresponding features of the other image, and choosing the pair with the smallest distance.

However, a small Euclidean distance does not necessarily mean that the points represent the same feature. For instance, if two completely different scenes are depicted in two images, a small distance between two particular features does not signify that the respective image points represent the same 3D point. It merely indicates that the two features have the highest resemblance of all the processed features. To accurately match a feature in the candidate image, we identify the closest and the second closest features in the reference image using a nearest neighbor search strategy. The correspondence is then determined using the ratio test proposed by Lowe [16]. If the ratio is below a predefined threshold (empirically set to 0.6 [18]), a feature pair is accepted as a correspondence, otherwise that match is rejected.

As our matching procedure is subject to errors and mismatches, many of our matches are spurious. It is possible to eliminate many spurious matches by enforcing geometric consistency (the *epipolar constraint*). This is predicated on the fact that, assuming a stationary scene, not all corresponding features between two images are physically resizable, regardless of what the actual shape of the scene is. Thus, for a given image pair, only matching features that agree with the epipolar constraint are admissible and all other matches are rejected.

Given a set of matching images, the next task is to recover the geometry of the scene and the motion information of the camera (camera parameters) simultaneously [6]. The motion information includes the extrinsic (position, orientation) and intrinsic parameters of the camera for the captured images. This is accomplished using *Structure-from-Motion*.

Our solution takes an incremental approach, in which a pair of images is selected to initialize the sequence. This initial pair should have a large number of feature matches, but must also have a large baseline. This is to ensure that the 3D coordinates of observed points are well-conditioned. Once the initial pair is selected, its *Essential matrix* is approximated using the *five-point algorithm*. The projection matrix can then be recovered by decomposing the obtained Essential matrix. Feature tracks visible in the two images are then triangulated, producing an initial set of 3D points.

We then use an iterative process where we add in each step the n images with the largest number of features whose 3D locations have already been estimated [1,6,19]. In other words, the new images share as many features as possible with images that have already processed, with the approximate locations of those features known. From our experiments, we selected $n = 3$ empirically, as this speeds up the optimization process significantly compared to adding images one at a time, while avoiding some inaccuracies that may occur when adding more images in a step. The camera parameters for each newly added image are initialized with the same orientation and focal length as the processed image that it matches best. This has proved to work very well even though images have a different rotation and scale. The Bundle Adjustment technique is subsequently applied to refine and improve the obtained solution, i.e., to approximate the camera parameters of every image and the 3D location of every feature.

Fig. 1 Several SfM stages of the reconstruction of our *White Rooster* data set. *Left* the initial two-frame reconstruction. *Middle* an intermediate stage after 22 images have been added. *Right* the final construction with 39 images

This last step is critical for the accuracy of the reconstruction, as concentration of pairwise homographies would accumulate errors and disregard constraints between images. The recovered geometry parameters should be consistent. That is, the reprojection error, which is defined by the distance between the projections of each feature and its observations, is minimized. The procedure is repeated until no more images can be added. The result of this stage is the camera motion information and a sparse scene geometry of the scene (Fig. 1).

3.2 Scene Geometry Enhancement

We now have successfully acquired both camera parameters and scene geometry. As the scene geometry is derived from distinctive features of the input images, the resultant point cloud is often sparse. In order to produce a more comprehensive model, the obtained scene geometry must be enhanced. This is accomplished by exploiting the silhouette information of the observed scene to generate additional 3D points. First, the foreground information is segmented in the input images by projecting the sparse scene geometry obtained from the previous stage onto each input images. The results of these projections are sets of points sparsely covering the object's surface in each image. For each set of points, a polygonal boundary is computed [20] and the background is defined as the pixels outside these regions [1].

Each set of contour points together with the camera center of that view defines a viewing cone, whose apex is located at

the camera's optical center. Each viewing cone consists of a number of cone lines. A cone line represents a 3D line formed by a silhouette contour point and the camera's optical center. The polyhedral visual hull information can be obtained by calculating the intersection of these viewing cones. However, 3D polygon intersection is non-trivial and often computationally expensive. Matusik et al. [5] proved that equivalent results can be obtained as follows: assume that we want to compute the intersection of silhouette A and B.

1. Project each cone line of the viewing cone A onto the silhouette B.
2. Calculate the intersection of the projected line and the silhouette B in 2D.
3. Lift the computed intersection points from 2D to 3D, which yields a set of 3D points, which defines a face of the polyhedral visual hull.

3.3 Mesh generation

The next step is to construct surfaces approximating the point cloud. Our goal is to find a smooth closed surface (without holes) that approximates the underlying 3D models from which the point clouds were sampled. We have evaluated several surface reconstruction techniques including the *power crust algorithm* [21], *α-shape* [22], and the *ball-pivoting algorithm* [23]. We decided to employ the *Poisson Surface Reconstruction algorithm* [24] for this task, since it produces a closed surface and works well for noisy data that exhibit high variation in term of density. In contrast to many other implicit surface fitting methods, which often segment the data into regions for local fitting and then combine these local approximations using blending functions, Poisson surface reconstruction processes all the sample points at once, without resorting to spatial segmentation or blending [24]. Figure 2 shows the computed point cloud and the resulting mesh of our Rooster data set.

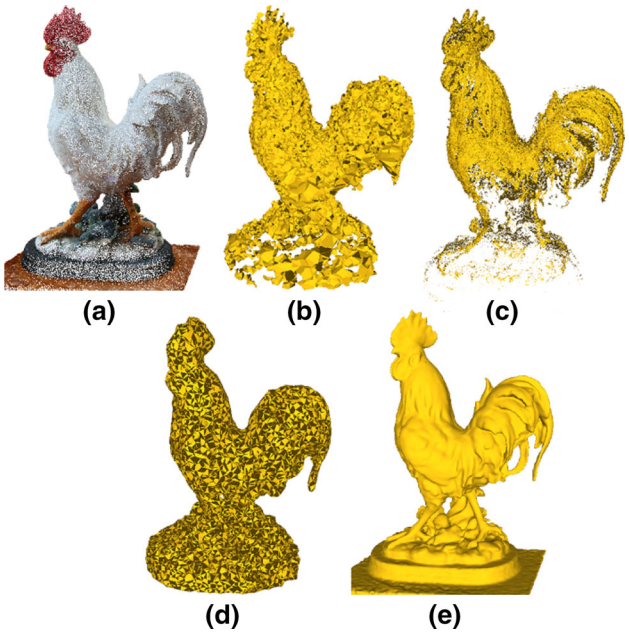

Fig. 2 Comparison of surface reconstruction techniques: an input point cloud (**a**) and the resulting 3D meshes obtained using the *Power Crush* algorithm (with a sampling density constant parameter of 0.6 and a default alpha angle of 0.4) (**b**), the *Ball Pivot* algorithm (with a ball radius of 0.05) (**c**), the *Alpha Shape* algorithm (with an alpha value of 0.98) (**d**), and *Poisson Surface Reconstruction* (with an octree depth of 9) (**e**)

3.4 Texture map construction

The construction of the high-quality texture map consists of three steps: first, the 3D mesh model is parameterized yielding a one-to-one mapping from 3D surface to a 2D parameter space. Next, input images are projected onto the surface and suitable texture regions are identified, cut, and fused together to form a 2D texture atlas. Finally, missing textures are generated using a poisson-exemplar-based inpainting method.

3.4.1 Surface parameterization

The objective is to segment the surface of the reconstructed 3D model into patches and unwrap them onto a 2D planar surface. We evaluated different surface parameterization techniques, but found that existing libraries, such as Blender, either create a very disjoint map of triangle patches, or create a single-parameter patch with large distortions. We hence use a Feature-based Surface Parameterization, which consists of three stages [25]: genus reduction, feature identification, and patch creation.

1. *Genus reduction* The genus of a surface is defined as the largest number of nonintersecting simple closed curves that can be drawn on the surface without separating it. In order to identify non-zero genus surfaces, a surface-based *Reeb* graph [26] induced by the *average geodesic distance* [27] is constructed. The leaf nodes of this Reeb graph reveal the tips of the protrusions of the meshes, while loops in the graph signify the existence of handles.

For an n-genus surface, loops that do not separate the surface into two disjoint connected components are *nonseperating cycles*. Conceptually, one can visualize these loops by imagining a hollow handle connected to the surface. One of the loop cuts across the handle, while the other follows it. The principle behind genus reduction is to identify appropriate *nonseperating cycles* for each handle and cut the surface open along the cycles, which essentially reduces the genus of the surface by one. This process is repeated until there is no more handles.

2. *Feature identification* From the *Reeb* graph, we can identify the tips of the protrusions. Once the tip of a protrusion

has been found, we separate the feature from the rest of the surface by constructing a closed curve γ. This closed curve is constructed by first separating region R that corresponds to the tip of the protrusion.

Separating region R for a tip point $\mathbf{p}$ is determined by first computing the function $f_{\mathbf{p}}(\mathbf{q}) = g(\mathbf{p}, \mathbf{q})$, where $g(\mathbf{p}, \mathbf{q})$ is the *geodesic distance* between the points $\mathbf{p}$ and $\mathbf{q}$ [27]. The value of $f_{\mathbf{p}}$ is normalized to fit in the interval [0, 1].

Regions which are bounded by a given iso-value are examined. Specifically, the interval [0, 1] is partitioned into k equal sections. The surface is then divided into level-set bands by performing region growing from the tip of the protrusion $\mathbf{p}$ based on the values of $f_{\mathbf{p}}$ in these intervals [25]:

$$\mathbf{M}_i = \left\{ \mathbf{q} \in S \mid \frac{i-1}{k} \leq f_{\mathbf{p}}(\mathbf{q}) \leq \frac{i}{k} \right\}, \tag{1}$$

where S is the surface (mesh) and $1 \leq i \leq k$. As variation in the *area* of this sequence of bands tends to be small along a protrusion slope, and large where the feature connects to the remaining of the surface, the separating region R can be extracted by examining $\mathbf{A}_i = Area(\mathbf{M}_i$, which is considered as a continuous function $\mathbf{A}(x)$. To remove any small undulations, $\mathbf{A}(x)$ is passed through a *Gaussian filter* function for N number of times. All three values ($iso\text{-}value$, k, and N) influence the effectiveness and efficiency of the region separation process. The larger the $iso\text{-}value$ is, the farther the region-growing process grows. This leads to fewer number of surface patches be generated. Higher k values result in more samples to be used to discretize $\mathbf{A}(x)$, increasing the probability of small noise being considered as potential candidate places for the separating region. In practice, the value $k = 100$ seems to produce best results. Too large N values tend to cause the location of the separating region to shift or in some cases lost, while too small values often result in false separations.

Once R has been properly identified, γ is then constructed from R as follows: a collection of edges in the surface separating the feature from the rest of the surface (the skeleton) of R is found. During this process, dangling edges are rejected. A separating cycle ρ from this skeleton is then extracted. Finally, a shorter and smoother separating cycle γ is constructed based on ρ.

3. *Patch creation* Patches are created as follows: first, patches are unwrapped using a *discrete conformal mappings* technique [28]. This works by first positioning the texture coordinates of the boundary vertices, and then proceeding to solve the texture coordinates of the interior vertices through a closed form system. The main problem with this mapping technique is that regions can be stretched or compressed during the process leading to areas of the meshes being not preserved. This in turn results in uneven sampling rates across the surface.

Fig. 3 Our white rooster model segmented into patches (*left*) and the corresponding regions in the texture atlas (*right*)

To overcome this, a post-processing step is required. During this optimization stage, the interior vertices' texture coordinates are optimized to reduce the geometric distortion. This is achieved by first computing an initial harmonic parameterization [29]. A *square virtual boundary* enclosing the patch is constructed. The exact coordinates of the boundary are not important as long as they do not coincide with those of the patch's boundary. We then triangulate the regions between the virtual boundary and the original boundary using Scaffold triangles. A patch optimization technique proposed by Sandle et al. [30] is then applied on the enlarged patch.

Figure 3 depicts the resulting parameterization of our white rooster model. Each patch in the 2D texture map corresponds to a surface segment of the 3D model.

3.4.2 Texture atlas generation

At this stage, we have successfully obtained a parameterization of the 3D model. The next task is to construct a 2D texture atlas using the computed parameterization. This computation is achieved in three stages:

1. *Image region identification* For each 3D patch of the surface parameterization, we need to identify the image regions mapping onto it. We project all triangles of a patch onto all input images where it is visible, i.e., (1) the face normal forms an angle of less than 90° with the vector to the estimated camera position; (2) the face is not occluded by other surface regions. The resulting 2D image regions and the one-to-one correspondence between projected 2D triangles and original 3D triangles of the patch are saved for the next stage of the algorithm.

2. *Texture atlas computation* At this stage for each patch, we have a set of texture regions. The goal is to process these texture regions to produce a new texture that will cover the patch. To extract a texture region from an image and paste it over a patch, we need to find a transformation that transforms the arbitrary-shaped texture region to the desired shape of the patch. For each triangle on the object's surface, we determine the affine transformation Φ_2 mapping the corresponding region of the best fitting image to it, and we know

the surface parameterization Φ_1 mapping a section of the parameter space to this triangle. The mapping form the best fitting input image to the texture atlas is hence $\Phi_2 \circ \Phi_1^{-1}$.

The procedure is repeated for each texture region yielding a set of overlapping textures covering the face of the processing patch. These overlapping patches will be fused together using a *graph-cut technique* [31] to minimize seams between overlapping regions.

3. *Seam minimization* Seams between overlapping textures are minimized using a graph-cut technique [31]. Consider two overlapping image regions A and B, the objective is to find a cut within the overlap region, which creates the best transition between these images. The overlap region is represented as directed graph, where each node represents a pixel position p in the overlap region, which is denoted $A(p)$ and $B(p)$ for the two images A and B, respectively. Nodes are connected by edges representing 4-connectivity between pixels. Each edge is given a cost encoding the pixel differences between the two source images at that position.

We have investigated the effect of different parameters for image fusion applications [32] and tested them with various 3D models. Based on this, we use the following parameters: Image pixels are represented in the RGB color space. Color distances are computed using the L_2 norm. The cost function w corresponds to the gradient weighted color difference between the images A and B at the neighboring pixels p and q, i.e.,

$$w = w(p, q, A, B) = ||A(p) - B(p)|| + ||A(q) - B(q)||. \tag{2}$$

This cost function has been originally devised by [31] based on the observation that seams are more noticeable in low-frequency regions, and a visually more pleasing cut is computed by increasing the cost of an edge with a decreasing image gradient.

Figure 4 shows an example in which two texture patches of our Rooster model are fused together to form a larger and more complete texture patch. The newly merged texture patch is then fused together with the next available texture patch in the list. The process terminates when all texture patches have been successfully merged.

3.4.3 Inpainting

The final task is to generate missing textures for surface regions that are not visible in the input images. The problem of modifying an image to revert deterioration in an unobstructive way has long been an intensive research field in computer graphics. There are a number of well-known image completion methods available in the literature. These methods can be generally classified into two classes: exemplar-based and non-exemplar-based class.

Fig. 4 Seam minimization. Source texture patches are on the *left*. The *middle column* shows a merge by overlaying one patch on top of the other. The *right most column* displays the result obtained using a graph-cut technique

The dominant approaches of non-exemplar-based class are typically pixel-based technique. These methods [33–35] attempt to generate missing pixels by applying various mathematical models. This class of methods often produces good results for small and narrow gaps, but tends to yield blurred inpainted textures for larger gaps such as those in our case, rendering them unsuitable for our goals. Additionally, since these methods compute pixels' intensity values using mathematical models, they do not aim to preserve the semantic information of the texture.

Among exemplar-based methods, two main groups can be further distinguished: pixel-based and patch-based techniques. Pixel-based techniques attempt to find and copy best-fit pixels from the source image and transfer them over to the target region, whereas patch-based techniques (e.g., [36,37]) search and find patches or group of pixels and copy them over to the target region. These methods are designed to handle larger missing regions. However, problems of accurately finding best-fit patches and smoothly merging different patches are the key difficulties of this class of methods. These drawbacks render them unsuitable for our purposes.

Our aim is to fill moderately large holes of arbitrary shapes in an image using plausible texture information. We hence decided to employ a patch-based exemplar-based texture synthesis. The algorithm fills a missing texture region by starting from the boundary of the missing regions and finding best-fit texture patches in the valid texture regions and coping them over to the missing regions (Fig. 5). Seams between partially overlapping texture patches (Fig. 6) are eliminated by blending them using a Poisson-guided interpolation.

Filling order is critical for inpainting techniques and in particular non-parametric texture synthesis. Traditionally, the most well-known method has been the "onion peel," where the inpainted region is synthesized in concentric layers inwardly [36]. Therefore, in our method, we iteratively shrink the gap of the inpainted region by continuously assigning colors to boundary pixels. The entire procedure is as follows:

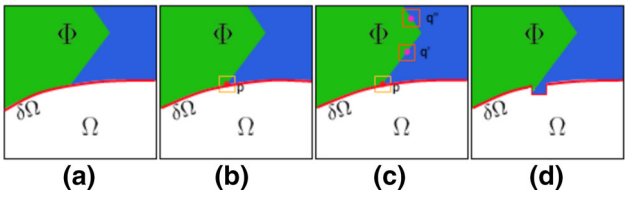

Fig. 5 **a** The original input image with the source region Φ, the target region Ω, and the boundary $\delta\Omega$. **b** Attempting to reconstruct an area around pixel p. **c** Several likely candidate matches are found in the source region. **d** The content of the best patch is copied over, resulting in a partial filling of Ω (adapted from [36])

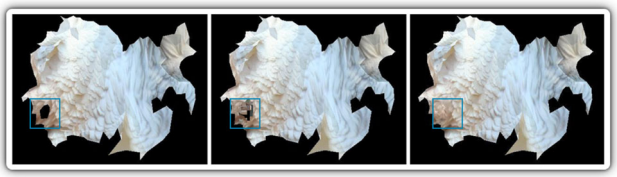

Fig. 6 *Left* A portion of the original texture atlas. *Middle* inpainting result using the exemplar-based inpainting method by Criminisi et al. *Right* our result

1. *Patch Priorities* Given a set of boundary pixels, the objective is to determine the order or the priority of the pixels to be processed. For each boundary pixel p, let Ψ_{pA} be a patch centered around p. The priority of p is defined as follows [36]:

$$Priority(p) = Confidence(p) \times Data(p). \tag{3}$$

The *confidence term*, which quantifies the amount of reliable information in the pixel's neighborhood, is defined as

$$Confidence(p) = \frac{\sum_{q \in \Psi_{pA} \cap \Omega} \sum_{i \in I} n_f \cdot v_{if}}{|\Psi_{pA}|} \tag{4}$$

with $|\Psi_{pA}|$ the area of the patch Ψ_{pA} and Ω denotes the target region to be inpainted. The function $\sum_{i \in I} n_f \cdot v_{if}$ measures the reliability of a pixel.

Here, "$\cdot$" denotes the dot product operation and I represents a set of the input images mapping to the processing pixel q. n_f denotes the normal of the face f that contains the 3D vertex corresponding to the pixel q, and v_{if} denotes a vector from the center of the face f to the camera center of the image i. The confidence term aims to boost the priorities of patches that have more already-filled and reliable pixels, allowing them to be synthesized first.

The *data term*, which defines the strength of the isophotes arriving the boundary, is defined as

$$Data(p) = \frac{|\bigtriangledown I_p^{\perp} \cdot n_p|}{\alpha}, \tag{5}$$

where $\bigtriangledown I_p^{\perp}$ represents a vector that is orthogonal to the gradient vector at p, n_p is the normal at p, and α denotes a normalization factor ($\alpha = 255$ for typical RGB color images). The intention here is to give higher priorities to patches that have an isophote "flown" into. This essentially encourages linear structure to be processed first.

2. *Patch Search* The algorithm's performance is significantly affected by the ability to identify the patch in the image that retains the highest resemblance to the processed patch. This is achieved by iteratively traversing through each pixel of the image outside the missing region and computing the similarity of the patch centered around that pixel and the original patch. Instead of using the standard *Sum of Squared Differences* (*SSD*) to measure the similarity of two given patches, we employ *appearance space attributes* [38,39], which provide much more information and thus improve the search result. For each pixel, the appearance space attribute contains not only the RGB color value, but also its signed feature distance and gradient in both directions. This provides far more accurate information about each pixel and its neighborhood, and hence makes it possible to find better matching image regions.

When searching for a matching patch, an 11×11 window centered around the processing pixel is considered. For each pixel of this neighborhood, we take into account the *RGB* colors, the *gradient vector* as well as the signed Euclidean distance to the closest dominant feature to the original texture. We showed previously [40] that these attributes seem most effective for exemplar-based texture synthesis. The entire information is encapsulated into a $11 \times 11 \times (3 + 2 + 1) = 726$-dimensional vector.

Determining the similarity of two given patches by comparing two 726-dimensional vectors is not efficient. In order to make the appearance space more practicable, the 726-dimensional vectors are projected into low-dimensional vectors using *principal component analysis* (*PCA*) [39]. In our method, the dimensionality is reduced to 12, which from our experiments on different types of images provided the best trade-off between image quality and computation time [40].

The clear advantage of attribute space over the conventional *SSD* is that the attribute space approach permits any meaningful information about the pixels and their surrounding to be embedded for matching purposes. By reducing the dimensionality, the computation time can be kept manageable.

3. *Patch Fusion* The final step is to replicate the content of the candidate patch and smoothly blend it with the target region. We employ *Poisson-guided interpolation* proposed by Pérez at al. [35] for this task. The principle behind this is fairly straightforward.

Suppose Ψ_B is the candidate patch to be copied and fused over the target patch Ψ_A, and let ∂_A and ∂_B be the boundaries of the target and candidate patches, respectively. The goal is to adjust the color information of Ψ_B, while preserving the *relative information* (image gradient) as much as possible, so

that the transition between the newly modified patch Ψ_C and the rest of the image is gracefully blended. This is accomplished as follows:

First, the values of the boundary pixels of Ψ_C are initialized to be equal to the corresponding values of the boundary pixels of Ψ_A. This is to ensure that the isophotes arriving at the boundary is properly maintained.

$$\Psi_{C_{(x,y)}} = \Psi_{A_{(x,y)}} \quad \forall (x, y) \in \partial_B \tag{6}$$

Next, each color channel's value of the renaming interior pixels within Ψ_C is independently adjusted to be consistent with the boundary pixels while constraining the image gradient to be the same to those of Ψ_B.

$$\bigtriangledown C(x, y) = \bigtriangledown B(x, y) \quad \forall (x, y) \in \Psi_C \backslash \partial_C, \tag{7}$$

where $\bigtriangledown C(x, y)$ and $\bigtriangledown B(x, y)$ are defined as

$$\bigtriangledown C(x, y) = |N| \;\; C(x, y) \;\; - \kappa \;\; - \lambda \tag{8}$$

with N the number of valid pixels, and κ and λ are defined as follows:

$$\kappa = \sum_{(x+\delta x, y+\delta y) \in \Psi_A} C(x + \delta x, y + \delta y) \tag{9}$$

$$\lambda = \sum_{(x+\delta x, y+\delta y) \in \partial_A} A(x + \delta x, y + \delta y). \tag{10}$$

A pixel is considered valid if it is inside the processing patch.

$$\bigtriangledown B(x, y) = \sum_{(x+\delta x, y+\delta y) \in \Psi_A \cup \partial_A} \widehat{B} \tag{11}$$

δx and δy designate a set of 4-connected neighbors around x and y and

$$\widehat{B} = B(x, y) - B(x + \delta x, y + \delta y). \tag{12}$$

Eq. 7 can then be expressed in the form of a system of linear equations with $\boldsymbol{i}$ variables ($\boldsymbol{i}$ is the number of pixels in $\Psi_{C_{(x,y)}}$), and can be easily solved using an iterative matrix solving technique such as the *Jacobi Method*.

4 Results

We have evaluated our image-based modeling system using simulated image data (to determine sensitivity with respect to different input parameters), a 3D model obtained with a laser range scanner (the most precise scanning technology), and by comparing it with other popular image-based modeling systems.

4.1 Sensitivity to input parameters

In order to determine the effect of different input parameters on reconstruction quality, we used a 3D model of the Stanford bunny [41]. A set of N input images was rendered and used as input to our image-based modeling system. The virtual camera locations for rendering the input images were defined using a spiral point data set [42]. Unless specified otherwise, the images had a resolution of 3648 × 2056 pixels. We tested the effect of the number N of input images, the resolution of input images, scene illumination (brightness), lens effects (barrel distortion), and the feature richness of the model's texture. The reconstructed 3D models were aligned with the original bunny model using an Iterative Closest Point algorithm [43], and the geometric error between them was computed using the mean and root-mean-square error of the two-sided Hausdorff distance, which is defined as

$$D(A, B) = max\,\{h(A, B),\ h(B, A)\} \tag{13}$$

$$h(A, B) = \max_{a \in A} \left\{ \min_{b \in B} \{d(a, b)\} \right\}, \tag{14}$$

where $d(a, b)$ denotes the distance between two faces: a and b. All results were obtained using a Windows PC with an Intel i7 Quad-Core GPU. Parts of the algorithm were parallelized, but no GPU acceleration has been used for the 3D reconstruction.

4.1.1 Effect of the number of input images

In order to determine the effect of the number of input images on the reconstruction quality, nine data sets were created with between 10 and 50 images of the bunny model. Table 1 displays the number of images and the time complexity of the reconstruction of each data set.

Each of the nine data sets above was used as input for our image-based modeling system. The computation time varied

Table 1 Number of images for the data sets N1 to N9

Data set	Number of images	Time complexity
N1	50	2 h and 52 min
N2	45	2 h and 21 min
N3	40	2 h and 05 min
N4	35	1 h and 38 min
N5	30	1 h and 16 min
N6	25	59 min
N7	20	45 min
N8	15	33 min
N9	10	26 min

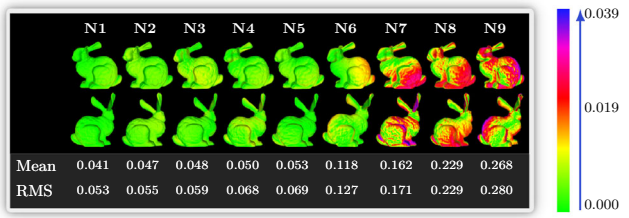

Fig. 7 Geometric error of 3D models reconstructed from the data sets N1 to N9 compared to the original bunny model. The color mapping depicts the two-sided Hausdorff distance between the reconstructed and original models. The values mean and RMS refer to the mean and root-mean-square error of the two-sided Hausdorff distance between the reconstructed and original models

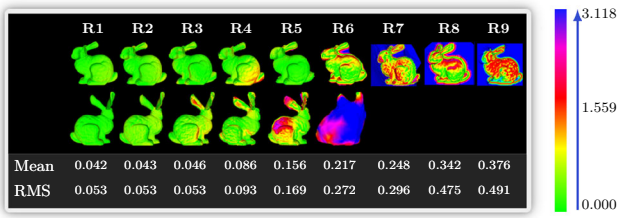

Fig. 8 Geometric error of 3D models reconstructed from the data sets R1 to R9 compared to the original bunny model. The color mapping depicts the two-sided Hausdorff distance between the reconstructed and original models. The values mean and RMS refer to the mean and root-mean-square error of the two-sided Hausdorff distance between the reconstructed and original models

between 26 min for data set N9 and 2 h and 52 min for data set N1.

Figure 7 demonstrates that the reconstruction quality decreases with a decreasing number of input images. The diagonal of the bounding box of the bunny model is roughly 14.3 units long. The max error of 0.039 indicates a reconstruction error of 0.27 % of the objects diagonal. The effect is initially quite small, but the error increases significantly when 25 or less images are used. This is due to the fact that the overlap between input images becomes too small for reliable feature matching and hence estimating camera parameters. We performed tests with more than 40 models and found that on average 25–35 images were sufficient to get good quality results. For complex objects (e.g., high genus, many concave regions), up to 60 input images were required.

4.1.2 Effect of image resolution

In order to determine the effect of the image resolution on the reconstruction quality, nine data sets were created. Each data set contained 50 input images obtained using the same camera parameters, except that the initial image resolution of 3648 × 2056 pixels was reduced by 10 % points at each step (see Table 2).

Table 2 Image resolution for the data sets R1 to R9

Data set	Image resolution	Time complexity
R1	3648 × 2056	2 h and 52 min
R2	3283 × 1850	2 h and 48 min
R3	2918 × 1644	2 h and 31 min
R4	2553 × 1439	2 h and 09 min
R5	2189 × 1234	1 h and 48 min
R6	1824 × 1028	1 h and 15 min
R7	1459 × 822	58 min
R8	1094 × 617	39 min
R9	730 × 411	17 min

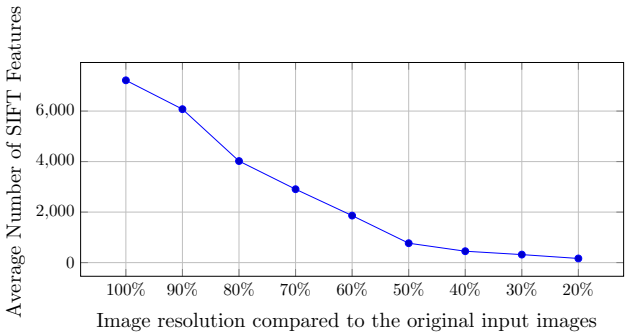

Fig. 9 Distinctive SIFT features versus image resolution

Each of the nine data sets above was used as input for our image-based modeling system. The computation time varied between 17 min for data set R9 and 2 h and 52 min for data set R1.

Figure 8 illustrates how the reconstruction quality decreases with reduced image resolution. The diagonal of the bounding box of the bunny model is approximately 14.3 units long. The max error of 3.118 indicates a reconstruction error of 21.8 % of the objects diagonal. The geometric error is initially barely noticeable, but increases rapidly with decreasing image resolution once the image resolution falls below 2.5 MPixels. Figure 9 provides an explanation. The number of SIFT features reduces rapidly with decreasing image resolution, but initially the number of features is still high enough to allow reliable matching of features across images. Once the image size halves, the number of SIFT features reduces by roughly 90 % compared to the original images. We found that at this stage only 29 images out of 50 are successfully registered. This results in a very sparse point cloud and hence large errors in the Poisson surface reconstruction. For the data sets R7 to R9, the number of registered images drops even further and the algorithm is not able to determine the relationship between points on the front and back of the model, and as a result, only the front section of the model is reconstructed.

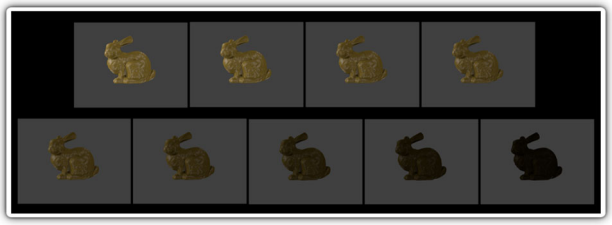

Fig. 10 Brightness of input images for the test data sets testing the relationship between image brightness and reconstruction quality

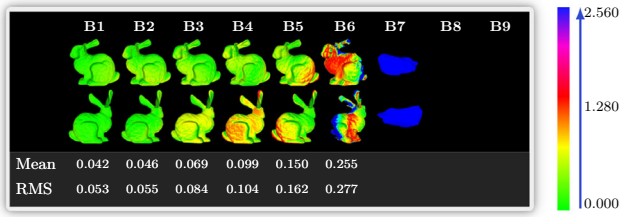

Fig. 11 Geometric error of 3D models reconstructed from the data sets B1 to B9 compared to the original bunny model. The color mapping depicts the two-sided Hausdorff distance between the reconstructed and original models. The values mean and RMS refer to the mean and root-mean-square error of the two-sided Hausdorff distance between the reconstructed and original models

4.1.3 Effect of scene illumination

In order to determine the effect of scene illumination, nine data sets B1 to B9 were created. Each data set contained 50 input images obtained using the same camera parameters. The intensity of the scene's light sources was reduced by 10 % points in each step, such that for data set B9, the light sources' intensity was only 20 % of those used for data set B1. Figure 10 demonstrates the effect of the reduced scene illumination on brightness of the rendered input images.

Each of the nine data sets above was used as input for our image-based modeling system. The computation time varied between 27 min for data set B6 and 2 h and 52 min for data set B1. No 3D models could be reconstructed for the data sets B7 to B9.

Figure 11 demonstrates that the reconstruction quality decreases with a decreasing scene illumination. The diagonal of the bounding box of the bunny model is approximately 14.3 units long. The max error of 2.56 (B7) indicates a reconstruction error of 17.9 % of the objects diagonal. The error is initially barely visible, but increases rapidly once the light source's brightness reduces by more than 40 %. The reconstruction fails once the light source intensity is reduced by 60 %. Figure 12 shows that in this case the number of SIFT features is reduced to about 2000 per image. While this was enough to reconstruct a 3D model from the image data set R5, a closer investigation reveals that for low scene illumination the quality of SIFT features reduces and the number

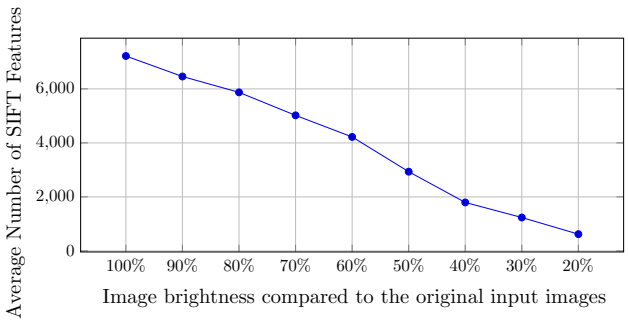

Fig. 12 Distinctive SIFT features versus image brightness

Fig. 13 One input image of the data sets D1 to D9, demonstrating the effect of adding an increasing amount of barrel distortion

of wrong matches increases dramatically. The reason for this is that the SIFT descriptor can compensate for global linear brightness changes using a normalization step, but it cannot compensate for local brightness changes [44].

4.1.4 Effect of image distortion

In order to determine the effect of lens distortion, nine data sets D1 to D9 were created. Each data set contained 50 input images, which were obtained using the same camera parameters, except that an increasing amount of barrel distortion was added using Eqs. 12 and 13.

$$x_u = x(1 + \kappa_1 r^2 + \kappa_2 r^4) \tag{15}$$

$$y_u = y(1 + \kappa_1 r^2 + \kappa_2 r^4), \tag{16}$$

where initially $\kappa_1 = 0.0021746$ and $\kappa_2 = 1/3\kappa_1$ and the values were increased by 10 % for each subsequent data set. Figure 13 demonstrates this effect using one input image for each of the nine data sets.

Each of the nine data sets above was used as an input for our image-based modeling system. The computation time was approximately 2 h and 50 min for each of the nine data sets. Figure 14 shows that the reconstruction error increases with increasing distortion, but initially it appears barely noticeable. For larger distortion factors (data sets D7 to D9), the reconstruction errors become more pronounced. The errors seem to be largest in areas where there are few silhouette points (i.e., on the body, rather than the ears and

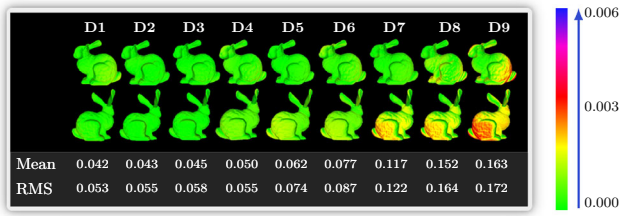

Fig. 14 Geometric error of 3D models reconstructed from the data sets D1 to D9 compared to the original bunny model. The color mapping depicts the two-sided Hausdorff distance between the reconstructed and original models. The values mean and RMS refer to the mean and root-mean-square error of the two-sided Hausdorff distance between the reconstructed and original models

Fig. 15 3D models with patterns showing increasing coarseness used to create the data sets F1 to F9

limbs). This indicates that the feature matching step is more sensitive to barrel distortion than the depth estimate from silhouette points. The diagonal of the bunny model is approximately 14.3 units long. The max error of 0.006 indicates a reconstruction error of 0.042 % of the objects diagonal.

4.1.5 Effect of the number of distinct features

In order to determine the effect of texture richness on the reconstruction process, we created nine data sets F1 to F9. Each data set contained 50 input images with constant resolution and were obtained using the same camera parameters. For each data set, the bunny model was given a pattern (*Voronoi crackle*) with increasing coarseness and hence a decreasing number of SIFT features. Figure 15 depicts the nine textured models used for this test suite, and Table 3 summarizes the average of the number of SIFT features in each data set.

Each of the nine data sets above was used as input for our image-based modeling system. The computation time varied between 29 min for data set F9 and 2 h and 52 min for data set F1. The reconstructed 3D models were aligned with the original bunny model, and the geometric error between them was computed using the two-sided Hausdorff distance.

The resulting reconstruction errors are illustrated in Fig. 16. The diagonal of the bounding box of the bunny model is approximately 14.3 units long. The max error of

Table 3 Average number of SIFT features per image for the data sets F1 to F9

Data set	Number of features
F1	6954
F2	5980
F3	3999
F4	2689
F5	1759
F6	1175
F7	892
F8	551
F9	303

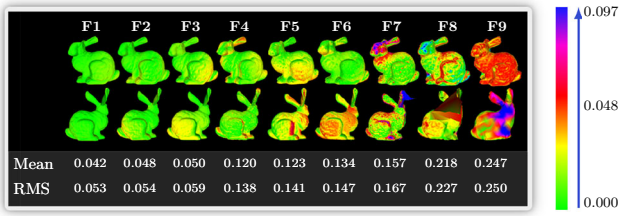

Fig. 16 Geometric error of 3D models reconstructed from the data sets F1 to F9 compared to the original bunny model. The *color mapping* depicts the two-sided Hausdorff distance between the reconstructed and original models. The values mean and RMS refer to the mean and root-mean-square error of the two-sided Hausdorff distance between the reconstructed and original models

0.097 indicates a reconstruction error of 0.68 % of the objects diagonal. The reconstruction quality is initially very good, but decreases markedly starting from data set F5, where each image has roughly 1800 SIFT features. For the two data sets using the coarsest texture, the number of SIFT features in the input images drops to roughly 550 and 300, respectively, which results in many unmatched input images, and hence poor estimation of camera parameters and depth estimation of features.

4.2 Comparison with laser scanning-based reconstruction

The objective here is to evaluate the accuracy of our technique and that of other available 3D reconstruction methods using a laser scanning data as a ground truth. A 3D representation of an owl sculpture was acquired using a Minolta VIVID 910 3D Laser Scanner. The object was scanned using two different orientations and a turntable. The resulting meshes were merged using MeshLab. The resulting mesh had 657,721 faces and contained only a few small holes in regions where the object's surface was very dark (e.g., the eyes). The holes were filled using the *Minimum Weight* filling technique in *MeshLab*.

Fig. 17 From *left* to *right* model obtained with laser scanner before and after filling holes, model obtained using our image-based modeling system before and after texture reconstruction, and a photograph of the original model

In the next step, 66 images of the owl sculpture were acquired using a standard consumer-level hand-held camera. The images had a resolution of 4000 × 3000 pixels. A 3D model was created using these images as input to our image-based modeling system. The computation took 6 h and 52 min on an Intel i7 Quad-Core PC. Figure 17 shows the resulting 3D models.

Two other reconstructions are obtained from the same input images using the two most well-known commercial 3D reconstruction systems [2]: *Agisoft* and *123D Catch*. The reconstruction using Agisoft took over 19 h on the same machine. 123D Catch required approximately 41 min to produce the final 3D model. However, we do not have information about what resources 123D Catch employs (e.g., GPU implementation or a multi-cloud platform for High-Performance Computing).

In order to compare the models obtained from the laser scanner and the reconstructed models, we aligned the models using the *iterative closest point* algorithm [43] and computed a mesh difference using the two-sided Hausdorff distance. The diagonal of the bounding box of the owl model is roughly 275 units long. The mean error of our image-based modeling system is roughly 0.1 (0.04 % of the objects diagonal). Most of the errors are in concave regions (e.g., the folds along the body), regions with poor illumination and shadowing (under the wing and ear), and regions covered by few photos (underside of the body and the ear). The mean error of 123D Catch's model is approximately 0.72 (0.26 % of the objects diagonal). The mean error of Agisoft's reconstruction is 0.69 (0.25 % of the objects diagonal). Figure 18 shows a visualization of the mesh differences between the models obtained from the laser scanner and other systems.

4.3 Comparison with other image-based modeling systems

In previous work, we evaluated several commercial image-based modeling systems and found that *Agisoft* and *123D Catch* performed best [2]. We hence compare the performance of our system with these systems.

Figure 19 depicts the reconstruction results for the *rooster* and the *statue of liberty* data sets using the above systems.

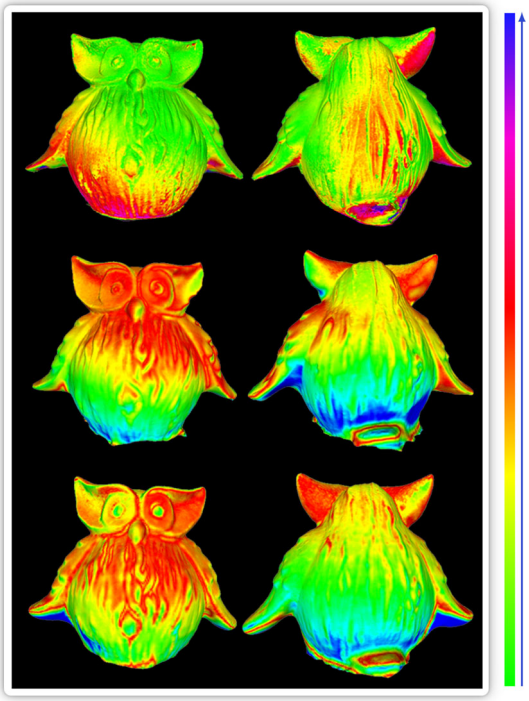

Fig. 18 Mesh error between the models obtained from the laser scanner, our image-based modeling, and other systems measured using the two-sided Hausdorff distance. From *top* to *bottom* error-encoded visualization of the 3D model reconstructed using our system, 123D catch, and Agisoft, respectively

Fig. 19 3D reconstructions of the *rooster* and *statue of liberty* data sets using (*left*) *Agisoft*, (*middle*) *123D Catch*, and (*right*) our system

Fig. 20 Reconstructions obtained using our system

There are 35 input images for the white rooster data set and 27 images for the a statue of liberty data set. 123D Catch produces reasonably good models in both test cases, although there is a slight disruption in the geometry in the head and tail regions of the rooster model or in the head and regions around the book of the statue of liberty model. The overall texture of the model is also not very well-reconstructed. Detailed texture information is replaced by rough and unrealistic texture. Agisoft produces rough models with disconnected regions and missing geometry. The reconstructed textures have little similarity to the original statues. The models produced with our system exhibit the highest accuracy both in terms of geometry and texture. The computation was done on an Intel Quad-Core i7 with 6GB RAM and took approximately 4 h and 32 min (for the white rooster model) and 2 h and 18 min (for the state of liberty model) for our system. Agisoft software was run on the same PC and took over a day for the white rooster model and roughly around 9 h and 27 min for the state of liberty model. The reconstruction using 123D Catch was done by submitting images through a web interface and took around 36 min (for the white rooster model) and 31 min (for the state of liberty model). Similar to the previous case, we do not have information about what resources 123D Catch employs.

4.4 Examples of reconstructed objects

Figure 20 illustrates some additional examples of objects reconstructed using the image-based modeling system described in this paper. The aim is to demonstrate the wide range of objects our system can handle. Note that many of the objects are poorly illuminated, have few visually distinctive features, and have a high genus (e.g., the shoe has gaps between the laces and the tongue). Several of the objects have very intricate geometry structures. Objects such as *the Holy Family* (top right), the tiger, and the pink cat exhibit highly reflective surfaces with very few distinctive features, which would pose great challenges for laser scanners.

5 Conclusion and future works

Our research was motivated by the observation that there is an increasing demand for virtual 3D models. Toward that end, we have described a image-based modeling system capable of creating high-quality 3D models from a set of unannotated images. Our method does not require any a priori or supplementary information about the scene. In contrast to previously presented methods, we integrate shape-from-silhouette and correspondence-based methods, which gives us very reliable camera parameter estimates and excellent geometry reconstruction. This equips our method to deal with both feature-poor objects and objects containing concave regions and holes.

Textures are combined using a greedy algorithm and a graph-cut technique minimizing gradient weighted color differences. The texture reconstruction uses an advanced surface parameterization method which takes into account the genus and geometric features of an object. Missing textures are generated by combining an exemplar-based inpainting method, appearance space attributes, and Poisson-guided interpolation. We have demonstrated the quality of the reconstruction process using objects with different geometries, genus, colors, and surface properties. In all cases, we achieved an excellent reconstruction and realistic texture. In contrast to laser scanners, our system also works for shiny and dark objects, and is easily scalable.

References

1. Nguyen, M.H., Wunsche, B., Delmas, P., Lutteroth, C.: A hybrid image-based modelling algorithm. In: Proceedings of 36th Australasian Computer Science Conference (ACSC 2013) (2013)
2. Nguyen, H.M., Wunsche, B., Delmas, P., Lutteroth, C.: 3D models from the black box: investigating the current state of image-based modeling. In: WSCG 2012 Communication Proceedings, pp. 249–258 (2012)
3. Hernandez, C., Vogiatzis, G., Cipolla, R.: Multi-view photometric stereo. IEEE Trans. Pattern Recognit. Mach. Intell. **30**, 548–554 (2008)
4. Franco, J.-S., Lapierre, M., Boyer, E.: Visual shapes of silhouette sets. In: 3D Data Processing, Visualization and Transmission, pp. 397–404 (2006)
5. Matusik, W., Buehler, C., Raskar, R., Gortler, S., McMillan, L.: Image-based visual hulls. In: Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques, pp. 369–374 (2000)
6. Nguyen, M.H., Wunsche, B., Delmas, P., Lutteroth, C.: Realistic 3D scene reconstruction from unconstrained and uncalibrated images. In: Proceedings of GRAPP 2011, Algarve, Portugal, vol. 31, pp. 67–75 (2011)
7. Baumgart, B.G.: Geometric modeling for computer vision. Doctoral Dissertation, Stanford University (1974)
8. Grauman, K., Shakhnarovich, G., Darrell, T.: A Bayesian approach to image-based visual hull reconstruction. In: IEEE International Conference on Computer Vision and Pattern Recognition, vol. 1, pp. 187–194 (2003)

9. Cheung, K., Baker, S., Kanade, T.: Shape-from-silhouette across time part 1: theory and algorithms. Int. J. Comput. Vis. **62**(1), 221–247 (2005)
10. Cheung, K., Baker, S., Kanade, T.: Shape-from-silhouette across time part 2: applications to human modeling and markerless motion tracking. Int. J. Comput. Vis. **63**(1), 225–245 (2005)
11. Franco, J.S., Boyer, E.: Exact polyhedral visual hulls. In: British Machine Vision Conference, pp. 329–338 (2003)
12. Debevec, P.E., Taylor, C.J., Malik, J.: Modeling and rendering architecture from photographs: a hybrid geometry and image-based approach. ACM Trans Graph. pp. 11–20 (1996)
13. Quan, L., Tan, P., Zeng, G., Yuan, L., Wang, J., Kang, S.B.: Image-based plant modeling. ACM Trans. Graph. **25**(3), 599–604 (2006)
14. Brown, M., Lowe, D.G.: Unsupervised 3D object recognition and reconstruction in unordered datasets. In: Fifth International Conference on 3D Digital Imaging and Modeling, pp. 56–63 (2005)
15. Snavely, N., Seitz, S., Szeliski, R.: Photo tourism: exploring photo collections in 3D. ACM Trans. Graph. **25**(3), 835–846 (2006)
16. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vis. **60**(2), 91–110 (2004)
17. Lowe, D.G.: Object recognition from local scale-invariant features. In: The Proceedings of the Seventh IEEE International Conference on Computer Vision, vol. 2, pp. 1150–1157 (1999)
18. Nguyen, H.M.: Accelerated 3D Content Creation using Stereo from Motion. Master Thesis. The University of Auckland, New Zealand (2012)
19. Remondino, F., El-Hakim, S.: Image-based 3D modelling: a review. Photogramm. Rec. **21**, 269–291 (2006)
20. Garai, G., Chaudhuri, B.B.: A split and merge procedure for polygonal border detection of dot pattern. Image Vis. Comput. **17**, 75–82 (1999)
21. Amenta, N., Choi, S., Kolluri, R.K.: The power crust. Int. J. Comput. Geom. Theory Appl. **19**, 127–153 (2000)
22. Edelsbrunner, H., Mucke, E.P.: Three-dimensional alpha shapes. ACM Trans. Graph. **13**, 43–72 (1994)
23. Bernardini, F., Mittleman, J., Rushmeier, H., Silva, C., Taubin, G.: The ball-pivoting algorithm for surface reconstruction. IEEE Trans. Vis. Comput. Graph. **5**(4), 349–359 (1999)
24. Kazhdan, M., Bolitho, M., Hoppe, H.: Poisson surface reconstruction. In: Proceedings of the Fourth Eurographics Symposium on Geometry Processing, pp. 61–70 (2006)
25. Zhang, E., Mischaikow, K., Turk, G.: Feature-based surface parameterization and texture mapping. ACM Trans. Graph. **24**(1), 1–27 (2005)
26. Reeb, Georges: Sur les points singuliers dune forme de pfaff completement integrable ou diune fonction numerique [on the (singular points of a completely integrable pfaff form or of a numerical function)]. Comptes Randus Acad. Sci. Paris **222**, 847–849 (1946)
27. Hilaga, M., Shinagawa, Y., Komura, T., Kunii, T.L.: Topology matching for fully automatic similarity estimation of 3D shapes. In: Computer Graphics Proceedings, Annual Conference Series (SIGGRAPH 2001), pp. 203–212 (2001)
28. Eck, M., DeRose, T., Duchamp, T., Hoppey, H., Lounsberyz, M., Stuetzle, W.: Multi-resolution analysis of arbitrary meshes. In: Computer Graphics Proceedings, Computer Graphics Proceedings, Annual Conference Series (SIGGRAPH 1995), pp. 173–182 (1995)
29. Floater, S.M.: Parametrization and smooth approximation of surface triangulations. Comput. Aided Geom. Des. **14**(3), 231–250 (1997)
30. Sander, P.V., Gortler, S.J., Snyder, J., Hoppe, H.: Signal-specialized parameterization, In: Proceedings of the 13th Eurographics Workshop on Rendering, pp. 87–100 (2002)
31. Kwata, V., Schodl, A., Essaa, I., Turk, G., Bobick, A.: Graphcut textures: image and video synthesis using graph cuts. ACM Trans. Graph. **22**(3), 277–286 (2003)
32. Clark, X.B., Finlay, J., Wilson, A., Milburn, K., Nguyen, M.H., Lutteroth, C., Wunsche, B.C.: An investigation into graphcut parameter optimisation for image-fusion applications. In: Proceedings of Image and Vision Computing New Zealand (IVCNZ 2012), pp. 480–485 (2012)
33. Bertalmio, M., Sapiro, G., Caselles, V., Ballester, C.: Image inpainting. In: Proceeding SIGGRAPH '00 Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques, pp. 417–424 (2000)
34. Telea, A.: An image inpainting technique based on the fast marching method. J. Graph. Tools **9**(1), 23–34 (2004)
35. Perez, P., Gangnet, M., Blake, A.: Poisson image editing. J. ACM Trans. Graph. **22**(3), 313–318 (2003)
36. Criminisi, A., Perez, P., Toyama, K.: Object removal by exemplar-based inpainting. IEEE Trans. Image Process. **19**(9), 1200–1212 (2004)
37. Harrison, P.: A non-hierarchical procedure for re-synthesis of complex texture. In: Proceedings of International Conference on Graphics, Visualisation and Computer Vision, pp. 190–197 (2001)
38. Manke, F., Wunsche, B.: Fast spatially controllable 2D/3D texture synthesis and morphing for multiple input textures. In: Proceedings of the 4th International Conference on Computer Graphics Theory and Applications (GRAPP 2009), pp. 5–12 (2009)
39. Lefebvre, S., Hoppe, H.: Appearance-space texture synthesis. In: ACM SIGGRAPH, pp. 541–548 (2006)
40. Nguyen, H.M., Wunsche, B., Delmas, P., Lutteroth, C.: Parameter optimisation for texture reconstruction. In: Proceedings of Image and Vision Computing New Zealand (IVCNZ 2013), pp. 226–230 (2013)
41. Stanford Bunny. https://graphics.stanford.edu/data/3Dscanrep/. Accessed 1st May 2014
42. Rakhmanov, E.A., Saff, E.B., Zhou, Y.M.: Minimal discrete energy on the sphere. Math. Res. Lett. **1**(6), 647–662 (1994)
43. Rusinkiewicz, S., Levoy, M.: Efficient Variants of the ICP Algorithm. In: Proceeding of Third International Conference on 3D Digital Imaging and Modeling (3DIM), pp. 145–152 (2001)
44. Zambanini, S., Kampel, M.: A local image descriptor robust to illumination changes. In: Proceedings of the 18th Scandinavian Conference on Image Analysis, pp. 11–21 (2013)

Hoang Minh Nguyen received his Master degree (with first class) at the University of Auckland and has been a Ph.D. candidate at the same university for the past 3 years. He is set to complete his study in 2015. His research centers on accelerated 3D contents creation and multi-view reconstructions. His interests are in the areas of computer graphics and vision. Along these lines, he has done work on segmentation, pose estimation, shape-from-X, volumetric reconstruction, texture inpainting, and sketching-based modeling.

Burkhard Wünsche received his Vordiplom at the University of Kaiserlautern, Germany, in 1994, and he received his M.Sc. and Ph.D. at the University of Auckland, New Zealand, in 1996 and 2004, respectively. He is currently a Senior Lecture at the Department of Computer Science at the University of Auckland, and he is the director of the Graphics Group and the Division of Biomedical Imaging and Visualization. Burkhard Wuensche has published more than 120 articles. His research interests include image-based and sketch-based modeling, scientific and biomedical visualization, game technology, AR and Mixed Reality, telehealthcare, and exergames.

Patrice Delmas is with the Department of Computer Science (The University of Auckland) since 2001, and currently he is an associate professor. Patrice is the founder and director of the IVS (Intelligent Vision Systems) Research group and IVSLab. Patrice is active in (1) the design of low-cost N-D computer vision systems for real-world applications; and (2) 2D/3D image processing applied to environmental sciences with a view towards trans-disciplinary expert systems.

Christof Lutteroth studied Computer Science at the Freie Universität Berlin and completed his Ph.D. at the University of Auckland in 2008, where he is now a Senior Lecturer. He is working in the fields of Human–Computer Interaction, Computer Graphics and Software Engineering. He is particularly interested in technologies that make it easier for end-users to create, change, and control interactive software systems.

Eugene Zhang received the Ph.D. degree in computer science from the Georgia Institute of Technology in 2004. He is currently an associate professor at Oregon State University, where he is a member of the School of Electrical Engineering and Computer Science. During part of 2011 and 2012, he was a guest professor at the Free University of Berlin and the Max-Planck-Institute in Informatics. His research interests include computer graphics, scientific visualization, geometric modeling, and computational topology. He received the National Science Foundation CAREER award in 2006. He is a senior member of the IEEE and the ACM.