Reconstruction from Multi-view Sketches: an Inverse Rendering Approach

Joan Colom[0000-0001-7577-0657]
Universitat Politècnica de València
Camino de Vera
46022, Valencia, Spain
joacoco@inf.upv.es

Hideo Saito[0000-0002-2421-9862]
Keio University
Hiyoshi Kohoku-ku
223-8522, Yokohama, Japan
hs@keio.jp

ABSTRACT

Reconstruction from real images has evolved very differently from reconstruction from sketches. Even though both present similarities, the latter aims to surpass the subjectivity that drawings present, increasing the task's uncertainty and complexity. In this work, we draw inspiration from reconstruction over real multi-view images and adapt it to work over sketches. We leverage inverse rendering as a refinement process for 3D colored meshes while proposing modifications for the domain of drawings. Compared to previous methods for sketches, our proposal recovers not only shape but color, offering an optimization system that does not require prior training. Through the results, we evaluate how different quality factors in sketches affect the reconstruction and report how our proposal adapts to them compared to directly applying existing inverse rendering systems for real images.

Keywords

Computer Vision, Graphics, Optimization, Image Processing, 3D Reconstruction, Inverse Rendering.

1. INTRODUCTION

The reconstruction from realistic multi-view images has improved remarkably in recent years, capable of recovering 3D shapes and materials [Bos20a, Goe20a, Kim16a, Li22a, Mil20a, Mun21a, Wan21a, Zha21a, Zha21b]. However, reconstruction from sketches has received less attention, having a slower development.

Multi-view reconstruction, from any source, aims to obtain a three-dimensional description of an object depicted in multiple two-dimensional representations through different views. Consequently, both tasks have a similar nature. However, reconstruction from realistic images often is approached as *inverse rendering*. This area aims to revert the rendering process, recovering unknown scene parameters from observations [Kat17a, Lai20a, Li18a, Liu19a, Mun21a, Nie04a]. A rendering pipeline is assumed and consistent, approximating the laws of light transport. Therefore, the ambiguity of the task is reduced with the number of meaningful references.

In contrast, sketches are not the result of laws of physics; instead, they are *subjective* views of reality [Xia22a]. If different people draw the same object, the results will be very different. In the same way, sketches are not interpreted equally by everyone. This higher ill-posed nature distinguishes the task over sketches from the task over realistic images. While the latter aims to invert a well-known process and recover the lost scene information, the former aims to overcome subjective interference to build the most plausible object.

However, this is challenging as any or no object may fit all the sketched views. Some previous approaches present interactive alternatives involving the user [Li16a, Li18b], leaning on the user's decisions to deal with uncertainty. Other works have tackled automatic reconstruction, either from single [Gao22a, Gry20a, Wan18a, Wan20a, Zha21c] or multi-view sketches [Lun17a, Han20a]. However, they have focused on shape, not considering color.

Color is crucial to define the visual identity of an object. When recovering color and shape, the closest references are found in reconstruction from realistic images. This paper proposes using inverse rendering to optimize textured models from sketches and flat-colored drawings by modifying the ideas of Goel et al. [Goe20a] for real images[1]. Our contributions are:

- We adapt inverse rendering techniques for real images to the ill-posed task of reconstruction over sketches and flat-colored drawings, proposing a split loss to ease the joint optimization.
- We propose a sampling method to partially recover mesh colors after a remeshing step.
- We study how different quality factors and inconsistencies in the source sketches affect the reconstruction.

[1] Code at github.com/JoanCoCo/SketchReconstruction.

The rest of this paper is structured as follows. First, Section 2 presents recent approaches related to our work. Next, Sections 3 and 4 introduce the proposed solution and the results obtained. Finally, Section 5 details the conclusions drawn from the results.

2. RECENT SOLUTIONS

Many works tackle reconstruction, either from real images or sketches. This section focuses on works close to ours: reconstruction from sketches as images and multi-view inverse rendering reconstruction.

Reconstruction from sketches

There are two works closest to ours in this area. Firstly, Han et al. [Han20a] proposed extracting attenuation maps from multi-view sketches using a CGAN. These allowed optimizing a voxel grid by a Direct Shape Optimization algorithm, obtaining a voxel representation of the target using the view poses.

Secondly, Lun et al. [Lun17a] used a similar scheme with different ideas. Through a CNN, from sketches, they generated the depth, normal, and foreground probability maps from 12 fixed views. Using them, partial point clouds were obtained and fused. Later, the global cloud was converted into a mesh and refined through contour fitting. In this case, the network had to be trained for a fixed set of input viewpoints.

Both approaches used generative models for producing intermediate spatial descriptions to optimize a 3D representation. Therefore, they required training to obtain a usable system, needing paired sketches and 3D shape information.

Even though some hand-drawn datasets exist, such as [Gry19a] and [Xia22a], they do not provide enough samples for training. Therefore, previous works relied on synthetic data [Han20a, Lun17a, Wan20a, Zha21c], overlooking the subjectivity of hand-drawn samples and resulting in low generalization. Instead of a trainable system, we propose a refinement scheme using inverse rendering.

Inverse rendering

Inverse rendering methods revert the rendering process to estimate scene parameters from images, usually thanks to differentiable rendering. Therefore, images are used as feedback, not requiring a ground truth 3D shape. However, each reconstruction requires "training" to optimize the desired scene parameters.

Kim et al. [Kim16a] already proposed a refinement approach for reconstruction based on multi-view images. With an initial Structure-from-Motion estimation [Sch16a], a differentiable rendering was used to optimize mesh, albedo, and lighting. NeRF [Mil20a] introduced an MLP characterizing the space given the position and viewing direction, which was optimized per scene through volume rendering. This allowed novel view synthesis and scene inspection, inspiring many works [Bos20a, Wan21a, Zha21a, Zha21b]. Finally, NVDiffRec [Mun21a] gathered similar ideas, proposing a mesh optimization from scratch. Given multi-view images, masks, and viewpoints, they used differentiable rendering with deferred shading to jointly optimize a deformable tetrahedral grid, materials, and environment maps, outputting a textured 3D mesh.

These approaches offered systems directly usable with any set of images given the required inputs, optimizing a 3D representation of the provided data. However, the rendering pipeline was usually simplified, looking for a good balance between a realistic appearance and efficiency [Bos20a, Kim16a, Mun21a, Zha21b].

Goel et al. presented a refinement approach using differentiable path tracing to avoid the limitations of local lighting [Goe20a]. Starting from an initial mesh, they applied two alternating refinement steps repeated cyclicly: a material step focused on BRDF parameters and a geometry step updating the vertex's positions. Upon convergence, subdivision increased the mesh resolution, followed by remeshing to fix artifacts.

As reported in [Col22a], these techniques are generic enough to be applied over sketches. However, they were designed for realistic images, not considering the sketches' uncertainty and subjectivity. Therefore, they present components unsuitable for drawings, showing worse performance than in their intended domain.

The following section proposes modifying the inverse rendering optimization techniques to work over sketches and flat-colored drawings. We build on the foundations of Goel et al. [Goe20a].

3. PROPOSED SOLUTION

We modify the refinement scheme by Goel et al. [Goe20a] to make it suitable for sketches and flat-colored drawings. Our changes fall in four areas:

- Sketches do not require realistic materials or complex lighting [Col22a]. Therefore, we replace the BRDF materials with a single purely diffuse material and fix the environment map to entirely white. Consequently, the input is reduced to multi-view plain image drawings, masks isolating the target, and the view poses of each image.
- Simultaneous optimization of mesh and materials is possible, as shown by [Kim16a, Mun21a]. We replace the alternating scheme in [Goe20a] with simultaneous optimization, followed by a *long-tail* refinement of the colors.
- Goel et al. reset the material after remeshing, discarding the estimated color. We propose a resampling scheme to recover the color partially.
- We guide the refinement process using split losses for shape and color, as well as regularizations.

In contrast, common elements with Goel et al.'s work are the refinement over an initial mesh, using mesh

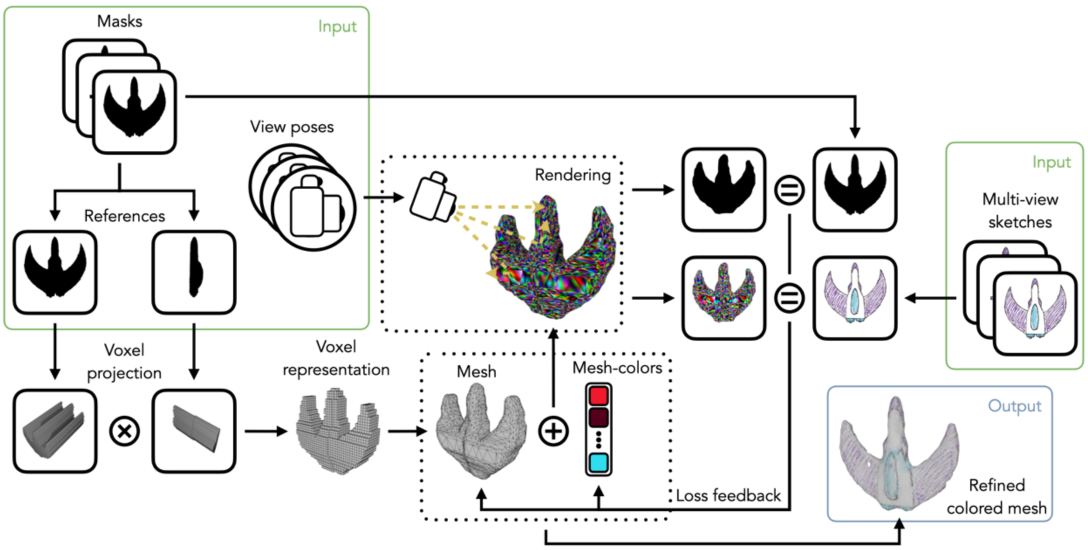

Figure 1. Summary of the basic proposal. Given multi-view sketches, masks, and view poses, an initial

colors [Yuk08a], and using remeshing for solving artifacts. The following sections present the solution in more detail, and Figure 1 summarizes it.

Optimization scheme

Given the sketches, masks, and viewpoints, an initial mesh is optimized to represent the sketched object. Goel et al. experimented with mesh initialization techniques, such as voxel carving or COLMAP [Sch16a], reporting the best performance with the latter. However, sketches and flat-colored drawings contain few key points, rendering COLMAP inadequate for our task. Instead, to obtain an initialization of the general shape, we use a simple visual hull estimation based on parallel projections from a given subset of sketches into a voxelated occupancy space. The total shape is the intersection of all the projections, and a mesh is obtained through marching cubes, remeshing, and simplification.

Mesh colors are used instead of textures to avoid the discontinuities of texture coordinates optimization [Mun21a], storing the sampled colors of the mesh triangles in a vector [Yuk08a]. Given the triangle ID t, resolution R, and barycentric coordinates (i, j) of a sample with $0 \leq i \leq R$ and $0 \leq j \leq R - i$, the index c of the sample in the color vector is computed with Equation 1. We initialize the colors randomly.

$$c = \frac{(R+1)(R+2)}{2} t + \frac{2R - i + 3}{2} i + j \qquad (1)$$

The refinement involves optimizing the mesh vertex positions and the color vector. These parameters compose a single object 3D scene that, when rendered from the given viewpoints, is converted into images. These can be compared with the references using loss functions. Finally, the differentiable rendering allows using gradient descent optimization to update the parameters jointly. Although disjoint coarse-to-fine optimization can allow finner detail [Goe20a], the detail requirement in sketches is generally low. Therefore, simultaneous optimization with fixed mesh resolution can lead to reasonable results in fewer steps.

Losses

Losses must account for the task's properties and the result's desirable features. We aim to obtain a 3D triangular mesh that resembles the sketched object. However, sketches are not consistent descriptions but interpretations of the world. Consequently, we do not aim to find a replica but a reasonable approximation.

The losses must inform the shape and colors of the target. Instead of capturing both with one loss [Goe20a], we set dedicated losses for better tailoring:

- Color loss. Color details in sketches are inconsistent. Sketch lines have two uses: conveying surface color detail or representing geometric features. Both produce color feedback, but only the former corresponds to true color information. Moreover, the second type is inconsistent between views. We use a Laplacian pyramid loss [Boj17a] for comparison at different resolution levels to deal with these issues. Coarser levels inform the general color. Meanwhile, finner levels reinforce consistent lines, while inconsistent lines are overtaken by coarse color feedback. Equation 2 presents it where I_P and I_T are the rendered and reference images; K is the number of levels; G is the Gaussian filter; $|I_P|$ and $D(I_P)$ are the number of pixels and channels of I_P; and I^l represents the image scaled down by l^{-1}.

$$L_C(I_P, I_T) = \sum_{i=1}^{|I_P|D(I_P)} \frac{\left|I_{P_i}^{K+1} - I_{T_i}^{K+1}\right|}{|I_P|D(I_P)} + \sum_{l=1}^{K} \sum_{i=1}^{|I_P|D(I_P)} \frac{\left|I_{P_i}^{l} - G\left(I_{P_i}^{l}\right) - I_{T_i}^{l} + G\left(I_{T_i}^{l}\right)\right|}{|I_P|D(I_P)} \tag{2}$$

- Silhouette loss. Due to the color inconsistencies and lack of shading, the silhouette is the primary source of shape information. We can capture it using the mean squared error of the masks. Even though the outline can present inconsistencies, this loss balances the feedback, averaging the references. Equation 3 models the loss, being M_P and M_T the rendered and ground truth masks.

$$L_M(M_P, M_T) = \frac{1}{|M_P|} \sum_{i=1}^{|M_P|} \left(M_{P_i} - M_{T_i}\right)^2 \tag{3}$$

However, to guide the reconstructions toward desirable properties, regularizations are also needed:

- Shape regularization. It favors smooth meshes and avoids degenerations. Equation 4 formulates it as an adaptation of the curvature flow smoothing [Oht01a] for uniformity on uneven distributions, where V is the set of vertices; F obtains the set of pairs of vertices that form a face with the input, and α_v is the angle at vertex v in a given triangle.

$$L_{CF}(V) = \sum_{v_i \in V} \left\| \sum_{v_j, v_k \in F(v_i)} (v_j - v_i) \cot \alpha_{v_k} + (v_k - v_i) \cot \alpha_{v_j} \right\|^2 \tag{4}$$

- Normal regularization. It favors meshes that induce automatic smooth normals by penalizing their variation inside a neighborhood. Equation 5 defines it based on Laplacian regularization [Mun21a, Oht01a], where N and n_v are the one-ring neighborhood and normal of a vertex.

$$L_N(V) = \sum_{v_i \in V} \left\| \frac{1}{|N(v_i)|} \sum_{v_j \in N(v_i)} n_{v_j} - n_{v_i} \right\|^2 \tag{5}$$

- Color smoothness regularization. It favors color uniformity inside the triangles. Equation 6 defines it, where T is the triangles set, C is a color vector, and $C_{R(s)}^{t}$ is the color of a random sample in t.

$$L_{CS}(T, C) = \sum_{t_i \in T} \left\| \frac{1}{4} \sum_{s=2}^{5} \left| C_{R(1)}^{t_i} - C_{R(s)}^{t_i} \right| \right\|^2 \tag{6}$$

- Spring regularization. Inspired by [Hop93a], it aims for a minimum solution, disfavoring overgrowing and balancing edge sizes. Equation 7 defines this function.

$$L_S(T) = \sum_{\substack{v_1, v_2, v_3 \in t_i \\ t_i \in T}} \|v_1 - v_2\| + \|v_2 - v_3\| + \|v_3 - v_1\| \tag{7}$$

Finally, decay of the normal regularization is applied to further avoid overgrowing, while progressive strengthening of the shape regularization is used to penalize significant changes towards the end. The total loss is expressed by Equations 8, 9, and 10, where MI is the maximum number of iterations and i is the current one. The weights of the losses were set empirically to make the main losses dominant while keeping the regularization magnitudes balanced.

$$L = 40L_M + 10L_C + 0.02d_{CF}(i)L_{CF} + 0.01d_N(i)L_N + 0.0002L_{CS} + 0.0025L_S \tag{8}$$

$$d_{CF}(i) = 1.0 - \max\left(0.01, \left(1.0 - 1.5\frac{i}{MI}\right)\right) \tag{9}$$

$$d_N(i) = \max\left(0.05, \min\left(0.4, \left|\log\left(\frac{i}{MI} + 10^{-4}\right)^5\right|\right)\right) \tag{10}$$

Remeshing and resampling

The measures for avoiding overgrowing still leave room for slight degeneration. Periodic screened Poisson reconstruction is applied to fix them, followed by simplification to keep the number of faces constant. However, this interferes with color estimation, as mesh colors are linked to the triangle IDs. When remeshing, a new mesh is generated, redefining vertices and faces. As a result, triangles in the same spatial position have different IDs, shuffling the surface colors. Therefore, part of the progress is lost. Instead of reinitializing the color as in [Goe20a], we propose a sampling method to recover lost progress.

By storing a copy of the mesh before remeshing, the colors of the new mesh can be updated sampling it. From now on, the input and output meshes to the remeshing are named m^i and m^o, respectively. Similarly, the color vectors of the input and resulting meshes are $\vec{c}^i$ and $\vec{c}^o$. With this, the resampling procedure consists in:

- For each triangle t_k^o in m^o, compute the world coordinates s_n^o of every sample inside it [Yuk08a].
- For each triangle t_k^i in m^i, compute its center $\bar{t}_k^i$ by averaging the positions of its vertices.
- The distance matrix from each sample to each center is computed. This allows finding the closest triangle of m^i to each sample in m^o, obtaining $\mathbb{C} = \left\{ \left(s_n^o, t_k^i\right) \middle| s_n^o \in m^o, t_k^i = \underset{t_h^i \in m^i}{\operatorname{argmin}} \left\|s_n^o - \bar{t}_h^i\right\| \right\}$.
- For every pair in $\mathbb{C}$, the barycentric coordinates of the projection of s_n^o into t_k^i are obtained [Hei05a]. With them, the index j in $\vec{c}^i$ associated to the projection is computed [Yuk08a]. By obtaining the index h of s_n^o in $\vec{c}^o$, $\vec{c}_j^i$ can be copied into $\vec{c}_h^o$.

In summary, this approach finds for each sample in m^o the closest color in m^i. Assuming topological similarity between a mesh and its remeshed version; it is expected that the colors for the new mesh will be the same as the closest points in the original mesh. This is reasonable, as Poisson remeshing tends to generate a smoother version of the original mesh, preserving the topology unless a very degenerated mesh is used.

The closest triangle is found using the distance to the center. Even though this can fail in some cases, it is generally a good approximation, allowing an efficient implementation with matrix operations. Additionally, it can be compensated by finding the N nearest triangles in m^i for each s_n^o, averaging the N projection colors. This recovers a less detailed version of the original colors. Nonetheless, the general colors are restored, and previous progress can be used.

Implementation details

Batch optimization is used with a batch size of four samples, and the initial mesh is estimated from a frontal and a side sample unless otherwise stated. Additionally, to compensate for the loss in color detail after remeshing, a long-tail training is used in which the last iterations focus on color only, fixing the shape. Remeshing is applied every N iterations and before the long tail when used, but not after.

Following [Col22a, Goe20a], we use a differentiable path tracing render as it reveals artifacts hidden by local lighting. We employ *pyredner* [Li18a], having added the mesh colors implementation of [Goe20a] to a recent version. The mesh colors resolution is set to three, and optimization renders use one bounce and four ray samples unless otherwise stated.

The estimated color vector is converted into a texture to export the result. This is done using optimization. From the reconstructed mesh, N sample images are generated from random views. Using them as references, a random texture mapped to the mesh is optimized using the color loss of Equation 2 and the texture smoothness regularization of [Mun21a]. Otherwise stated, ten reference views, 100 steps, and a texture resolution of 2048 by 2048 pixels are used. Finally, the optimizations are done using Adam, with a learning rate of 0.005 for the reconstruction and 0.05 for the texture generation.

4. EXPERIMENTAL RESULTS

Two synthetic examples were used for testing our system. Even though in Section 2 we stated why synthetic datasets do not fully represent our task, they are a baseline. Synthetic examples lack subjectivity, making their results the best case possible. Additionally, using synthetic samples allowed us to alter their quality, studying different input factors.

Through this section, we first present the data used and the baseline results. Next, we summarize how various quality factors of the sketches influence the results and show the effects of our sampling. Finally, we compare our results with those obtainable with Goel et al.'s [Goe20a] and Munkberg et al.'s [Mun21a] systems.

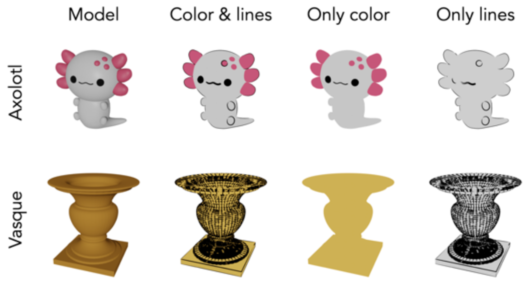

Figure 2. Models used and synthetic sketches in three styles.

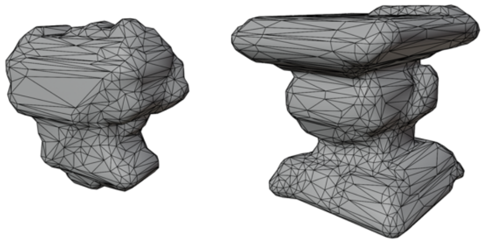

Figure 3. Initial mesh for Axolotl and Vasque.

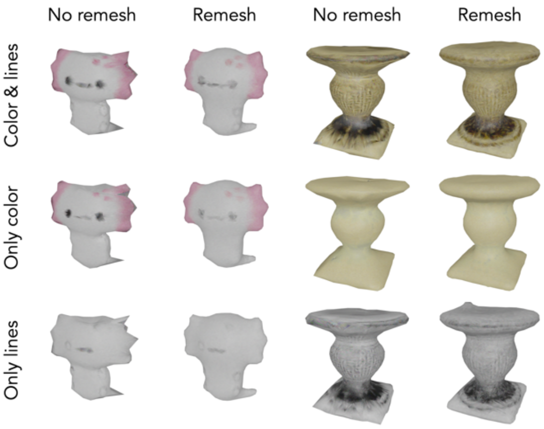

Figure 4. Baseline results for different styles.

Datasets

We gathered two 3D models to generate sketch-like references: Axolotl [fel20a] and Vasque [Fre07a], processing them to fit a cube of two units. The reasons behind this choice resided in their characteristics, representative of commonly drawn objects. While Axolotl has multiple colors, roundness, asymmetry, and a fictional appearance, Vasque represents a manufactured object with symmetry, sharp edges, and curves. Using Blender's Freestyle module as a rendering pipeline, we generated synthetic samples in three styles: lined sketches without color, flat-colored lined sketches, and flat-colored sketches without lines, shown in Figure 2. Each one of the sets contained 128 training and 128 validation random view samples of 512 by 512 pixels. The views were placed at a distance of five units around the target looking at it. From now

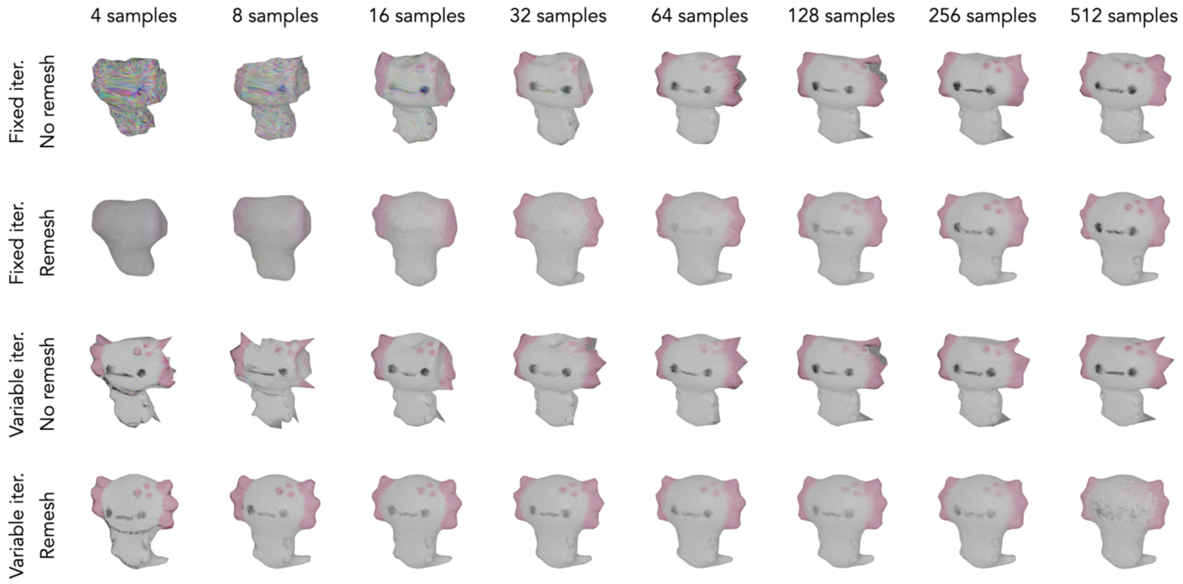

Figure 5. Reconstruction results over Axolotl with increasing numbers of samples.

on, we refer to the flat-colored lined sketches as the reference set.

Additionally, we generated modifications of the initial datasets to study the impact of quality factors. In each case, only the property under study was altered, keeping the rest of the parameters as the reference set.

Baseline results

We reconstructed each initial training set with remeshing and without it. Figure 3 shows the visual hull mesh initializations. In all cases, 30+3 iterations were used, meaning the last three only refined color. When using remeshing, it was applied every two iterations. We refer to this configuration as the reference configuration.

We evaluated the reconstructions using the validation sets and original models, measuring image metrics and average Chamfer distance. To compensate for scale mismatches, we measured both the pure distance and the distance after scaling the reconstruction to fit the largest dimension of the reference model.

Figure 4 and Table 1 show that the results present an acceptable quality. Better color estimation is observed from the style without lines, and remeshing is better for Axolotl than Vasque due to its rounded nature.

Quality factors study

Sketches contain noise and inconsistencies. Factors to consider are the number of available samples, their resolution, the precision of the masks, the consistency in the geometry between views, and the consistency between viewpoints and views. Here we summarize the results of their study while further details are provided in the complementary material.

The two most influential factors are the number of samples and the accuracy of the viewpoints provided for the drawings. Training sizes from 4 to 512 samples were used for the former, as seen in Figure 5. For each size, we used both 30+3 iterations and an adapted number of iterations and remeshing steps to keep the number of updates constant. Results with and without remeshing were obtained. The reconstructions for Axolotl are shown in Figure 5, following a similar trend to the ones for Vasque. We observe how the

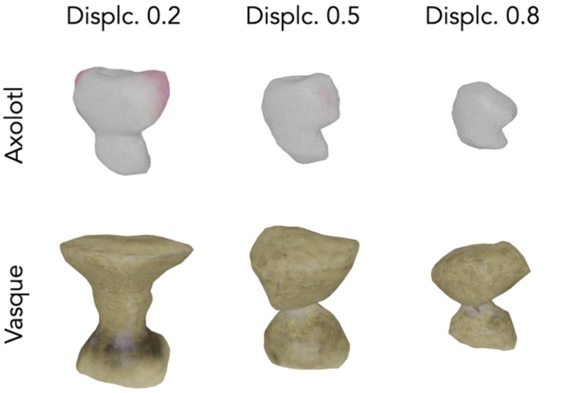

Figure 6. Results without remeshing of increasing levels of camera inaccuracy.

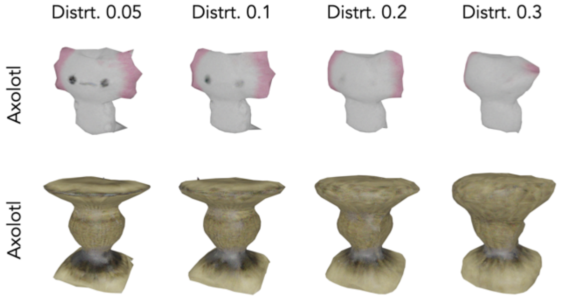

Figure 7. Results without remeshing of increasing levels of geometric inconsistency.

Model	Axolotl						Vasque					
Style	Color + lines		Only color		Only lines		Color + lines		Only color		Only lines	
Remesh.	No	Yes	No	Yes	No	Yes	No	Yes	No	Yes	No	Yes
MSE↓	0.007	0.007	0.005	**0.004**	0.007	0.006	0.026	0.027	**0.009**	0.010	0.033	0.031
PSNR↑	21.62	21.79	23.30	**23.99**	21.76	22.23	15.91	15.69	**20.57**	20.06	14.91	15.11
SSIM↑	0.923	0.926	0.934	**0.939**	0.925	0.929	0.792	0.789	**0.871**	0.870	0.790	0.789
LPIPS↓	0.109	**0.107**	0.140	0.139	0.147	0.149	**0.185**	0.197	0.191	0.190	0.246	0.247
Chamfer↓	0.057	0.083	**0.056**	0.082	0.057	0.081	0.079	0.113	0.076	0.108	**0.074**	0.119
Scaled Chamfer↓	0.014	**0.008**	0.013	**0.008**	0.013	**0.008**	0.025	0.034	0.023	0.030	**0.020**	0.037

Table 1. Results over 128 validation samples for reconstructions over 128 samples in different styles.

Model	Axolotl								
Set	B		NC		C6			CM	
System	Ours	Mun21a	Ours	Mun21a	Ours	Mun21a	Goe20a	Ours	Mun21a
MSE↓	0.006	**0.004**	**0.006**	0.007	0.008	**0.007**	0.033	0.020	**0.017**
PSNR↑	22.41	**23.70**	**22.61**	21.58	21.08	**21.67**	14.82	17.00	**17.83**
SSIM↑	0.924	**0.947**	0.925	**0.926**	0.917	**0.929**	0.907	0.912	**0.896**
LPIPS↓	0.104	**0.086**	0.142	**0.141**	0.107	**0.099**	0.128	0.133	**0.125**
Chamfer↓	0.068	**0.049**	0.065	**0.032**	0.058	**0.047**	0.048	0.033	**0.025**
Scaled Chamfer↓	0.005	**0.003**	**0.005**	0.016	**0.006**	0.010	0.064	**0.021**	0.022

Model	Vasque								
Set	B		NC		C6			CM	
System	Ours	Mun21a	Ours	Mun21a	Ours	Mun21a	Goe20a	Ours	Mun21a
MSE↓	**0.026**	0.033	0.030	**0.026**	**0.037**	0.043	0.095	0.067	**0.041**
PSNR↑	**15.96**	14.85	15.35	**15.88**	**14.41**	13.81	10.28	11.77	**13.87**
SSIM↑	0.791	**0.817**	0.793	**0.821**	0.779	**0.783**	0.746	0.757	**0.773**
LPIPS↓	**0.185**	0.195	0.239	**0.214**	**0.202**	0.214	0.251	0.260	**0.244**
Chamfer↓	0.073	**0.063**	0.077	**0.065**	0.056	0.086	**0.044**	0.136	**0.072**
Scaled Chamfer↓	0.024	**0.017**	0.024	**0.018**	**0.024**	0.031	0.086	0.110	**0.023**

Table 2. Validation results over 128 samples of the reconstructions with different systems (ours,

(NC), a set with six canonical views (C6), and the set with a camera displacement of 0.5 (CM).

quality increases with the number of samples and iterations. Good results are obtained with 16 and 32 samples and enough iterations, not improving significantly for more than 256 samples.

The camera inaccuracy was simulated by disturbing camera positions randomly so that cameras do not look at the mesh. Meanwhile, our system expects all views to look at it, effectively causing a discrepancy. We experimented with displacements of 0.2, 0.5, and 0.8. The results can be seen in Figure 6, showing how they quickly degrade. This is caused by the inconsistency of color and silhouette positions in the ground truth relative to the camera used for rendering. Consequently, results are averaged through the view space, reducing the mesh to what is consistently seen.

The rest of the factors were evaluated following similar procedures. A random scaling vector was applied to each mesh before rendering each sample, simulating geometric inconsistency. Magnitudes of scaling of 0.05, 0.1, 0.2, and 0.3 were evaluated. The results, seen in Figure 7, show that the inconsistency reduces the quality, averaging all the seen shapes.

The resolution study showed that lower resolutions decrease the reconstruction quality, as seen in Figure 8. However, good results are obtained from 128 by 128 pixels onwards. High resolutions increase the quality mildly, mainly improving color.

Finally, the mask precision was studied by generating eroded and dilated masks. The results, seen in Figure 9, are only slightly worse, being the effects averaged while reducing their negative impact.

Sampling method

Figure 10 shows the results of different approaches for dealing with color shuffling after remeshing under the reference set and configuration. Comparing grey restart as in [Goe20a] with our proposal, the former washes out colors, being the grey tone still noticeable. Meanwhile, leaving the system to refine the colors automatically gives a close result to our approach. However, mismatched color patches and higher bleeding still appear due to the shuffling, while our system significantly reduces these effects.

Comparison with inverse-rendering-based reconstruction techniques

We compared our performance with Goel et al.'s system (SFT) [Goe20a] and NVDiffRec [Mun21a] to study how our proposal adapts to sketches compared with standard inverse rendering. Synthetic datasets were used to see how the systems react to adversities common in hand-drawn sketches. We evaluated the reconstructions for the already introduced reference set (B), lined sketches without color (NC), and set with a camera displacement of 0.5 (CM). Additionally, we used a set with the same properties as the reference set but with only six canonical views (C6).

The configurations were the following. Our system used 30+10 iterations, applying remeshing for Axolotl every two iterations and no remeshing for Vasque. The normal, shape, spring, and smooth regularizations were removed when using remeshing; further details are found in the ablation study in the complementary material. For NVDiffRec, we used 5000 iterations, a fixed white environment, a grid of 128, and the remaining default parameters. For SFT, we used our initial meshes, a limit of 2048 triangles, and 12 cycles. For the first 10, the iterations for material –diffuse color and roughness– and geometry were limited to 5 and 150, respectively. From that point, the limits were set to 75 and 300. Learning rates of 0.01 and 0.0005 were used for material and geometry.

The reconstructions were evaluated with the reference validation set, considering only diffuse colors. Table 2 displays the results. Due to time constraints, SFT was only evaluated for C6, as its execution time with the other sets was significantly higher. Nonetheless, the available results show how our system adapts better to sketches than SFT. This reflects that our split loss captures better the shape and color while our regularizations help guide the optimization. When comparing with NVDiffRec, we observe mixed results. It is important to note that we work with 807 triangles on average against 35037 for NVDiffRec. In the baseline case, we can see that NVDiffRec presents more accurate results thanks to a sharper shape and color estimation. However, in the case of Vasque, we also observe that image metrics are penalized. This is because NVDiffRec estimates specular properties,

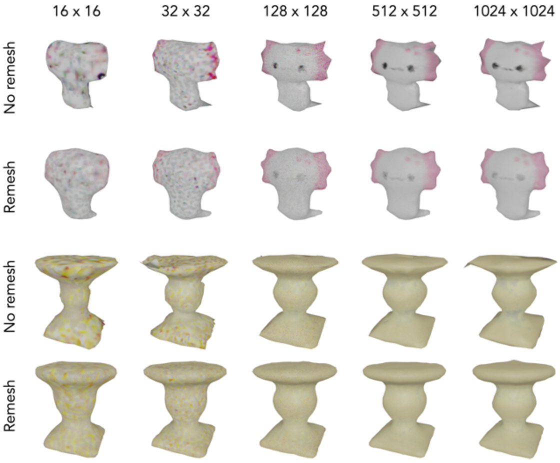

Figure 8. Results from different resolutions.

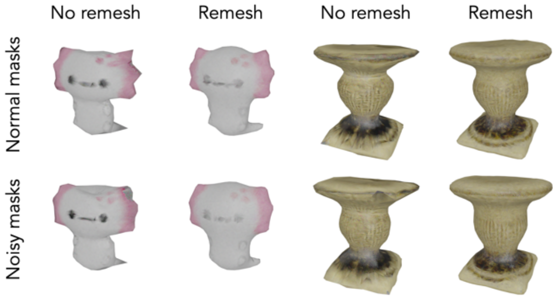

Figure 9. Results from altered masks.

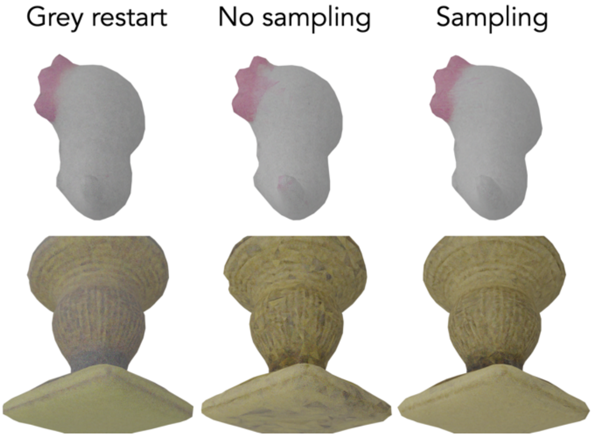

Figure 10. Comparison of the results obtained with different methods to fix color shuffling.

which can interfere with the diffuse color. This was also reported in [Col22a] and further backs using only diffuse materials to avoid material ambiguity.

We observe better performance for our approach under a lack of color, as NVDiffRec generates distorted surfaces with holes like in [Col22a]. This is mainly seen in Axolotl due to the lower line density. In C6, NVDiffRec generates squared reconstructions while our approach preserves roundness and closer shape estimation. With CM, both fail, allowing our system a higher surface uniformity than NVDiffRec. Finally, thanks to the joint estimation, our proposal presented a temporal cost similar to NVDiffRec, taking two to three hours. Meanwhile, our experiments with SFT on the same hardware have shown times ranging from 10 hours to several days.

5. CONCLUSION

This paper has presented how inverse rendering techniques can be used for 3D reconstruction from multi-view sketches. Considering the inherent uncertainty of drawings, we proposed dedicated and tailored losses for shape and color, easing the use of joint optimization. Additionally, we presented a resampling method to partially restore colors after remeshing. Finally, we reported how the quality factors of sketches affect the system's performance.

The results have shown that our system obtains acceptable reconstructions with enough samples and consistency, being more robust and flexible than previous inverse rendering techniques to the lack of color and the use of canonical views. By designing the system to find approximations rather than replicas, we obtained a broader genericity at the cost of lower baseline performance.

We have seen how remeshing can solve degenerations but tends to round the shape, which is not always beneficial. Therefore, its use should be considered case by case, being most helpful with inconsistencies or few samples.

Nonetheless, there are still issues to cover. The large number of samples required to obtain good results is a limiting factor when considering real use cases. The same applies to the high dependency on the view poses and their accuracy. This dependency is inherent to a method based on inverse rendering, as cameras are a crucial component. Future work is required to increase the robustness and reduce the need for provided viewpoints and large amounts of samples.

In conclusion, we proposed the use of inverse rendering to recover shape and color from multi-view sketches, with the caveat of requiring viewpoint specification and enough informative samples for good results. Therefore, overcoming these caveats will be critical for practically applying the system in real applications and use cases with hand-drawn sketches.

6. REFERENCES

[Wan20a] J. Wang, J. Lin, Q. Yu, R. Liu, Y. Chen, and S. X. Yu. 3D Shape Reconstruction from Free-Hand Sketches. *CoRR*, vol. abs/2006.09694, 2020.

[Col22a] J. Colom, and H. Saito. 3D Shape Reconstruction from Non-realistic Multiple-view Depictions Using NVDiffRec. Asia-Pacific Workshop on Mixed and Augmented Reality 2022, Yokohama, Japan, 2022.

[Lun17a] Z. Lun, M. Gadelha, E. Kalogerakis, S. Maji, and R. Wang. 3D Shape Reconstruction from Sketches via Multi-view Convolutional Networks. *CoRR*, vol. abs/1707.06375, 2017.

[Hei05a] W. Heidrich. Computing the Barycentric Coordinates of a Projected Point. *J. Graphics Tools*, vol. 10, pp. 9–12, Jan. 2005, doi: 10.1080/2151237X.2005.10129200.

[fel20a] felixyadomi. Cute Axolotl. 2020. URL: https://sketchfab.com/3d-models/cute-axolotl-e4625a288edf41afab1054a0fa529b3a

[Li18a] T.-M. Li, M. Aittala, F. Durand, and J. Lehtinen. Differentiable Monte Carlo Ray Tracing through Edge Sampling. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*, vol. 37, no. 6, p. 222:1-222:11, 2018.

[Xia22a] C. Xiao, W. Su, J. Liao, Z. Lian, Y.-Z. Song, and H. Fu. DifferSketching: How Differently Do People Sketch 3D Objects? arXiv, 2022, doi: 10.48550/ARXIV.2209.08791.

[Mun21a] J. Munkberg, J. Hasselgren, T. Shen, J. Gao, W. Chen, A. Evans, T. Müller, and S. Fidler. Extracting Triangular 3D Models, Materials, and Lighting From Images. arXiv, 2021, doi: 10.48550/ARXIV.2111.12503.

[Nie04a] M. B. Nielsen, and A. Brodersen. Inverse rendering of polished materials under constant complex uncontrolled illumination. *WSCG*, vol. 12, no. 1-3, pp. 309-316, 2004.

[Gry20a] Y. Gryaditskaya, F. Hähnlein, C. Liu, A. Sheffer, and A. Bousseau. Lifting Freehand Concept Sketches into 3D. *ACM Trans. Graph.*, vol. 39, no. 6, Nov. 2020, doi: 10.1145/3414685.3417851.

[Yuk08a] C. Yuksel, J. Keyser, and D. H. House. Mesh Colors. Department of Computer Science, Texas A&M University, 2008.

[Hop93a] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Mesh Optimization. *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*, New York, NY, USA, 1993, pp. 19–26, doi: 10.1145/166117.166119.

[Oht01a] Y. Ohtake, A. Belyaev, and I. Bogaevski. Mesh regularization and adaptive smoothing.

Computer-Aided Design, vol. 33, no. 11, pp. 789–800, 2001, doi: https://doi.org/10.1016/ S0010-4485(01)00095-1.

[Lai20a] S. Laine, J. Hellsten, T. Karras, Y. Seol, J. Lehtinen, and T. Aila. Modular Primitives for High-Performance Differentiable Rendering. arXiv, 2020, doi: 10.48550/ARXIV.2011.03277.

[Kim16a] K. Kim, A. Torii, and M. Okutomi. Multi-view Inverse Rendering Under Arbitrary Illumination and Albedo. in *Computer Vision – ECCV 2016*, Cham, 2016, pp. 750–767.

[Bos20a] M. Boss, R. Braun, V. Jampani, J. T. Barron, C. Liu, and H. P. A. Lensch. NeRD: Neural Reflectance Decomposition from Image Collections. arXiv, 2020, doi: 10.48550/ ARXIV.2012.03918.

[Mil20a] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. arXiv, 2020, doi: 10.48550/ ARXIV.2003.08934.

[Zha21a] X. Zhang, P. P. Srinivasan, B. Deng, P. Debevec, W. T. Freeman, and J. T. Barron. NeRFactor: Neural Factorization of Shape and Reflectance Under an Unknown Illumination. *ACM Transactions on Graphics*, vol. 40, no. 6, pp. 1–18, Dec. 2021, doi: 10.1145/3478513.3480496.

[Kat17a] H. Kato, Y. Ushiku, and T. Harada. Neural 3D Mesh Renderer. arXiv, 2017, doi: 10.48550/ ARXIV.1711.07566.

[Wan21a] P. Wang, L. Liu, Y. Liu, C. Theobalt, T. Komura, and W. Wang. NeuS: Learning Neural Implicit Surfaces by Volume Rendering for Multi-view Reconstruction. arXiv, 2021, doi: 10.48550/ARXIV.2106.10689.

[Gry19a] Y. Gryaditskaya, M. Sypesteyn, J. W. Hoftijzer, S. Pont, F. Durand, and A. Bousseau. OpenSketch: A Richly-Annotated Dataset of Product Design Sketches. *ACM Trans. Graph.*, vol. 38, no. 6, Nov. 2019, doi: 10.1145/3355089.3356533.

[Boj17a] P. Bojanowski, A. Joulin, D. Lopez-Paz, and A. Szlam. Optimizing the Latent Space of Generative Networks. arXiv, 2017, doi: 10.48550/ARXIV.1707.05776.

[Li22a] Z. Li, L. Wang, X. Huang, C. Pan, and J. Yang. PhyIR: Physics-based Inverse Rendering for Panoramic Indoor Images. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 12703–12713, doi: 10.1109/CVPR52688.2022.01238.

[Zha21b] K. Zhang, F. Luan, Q. Wang, K. Bala, and N. Snavely. PhySG: Inverse Rendering with Spherical Gaussians for Physics-based Material Editing and Relighting. arXiv, 2021, doi: 10.48550/ARXIV.2104.00674.

[Han20a] Z. Han, B. Ma, Y.-S. Liu, and M. Zwicker. Reconstructing 3D Shapes From Multiple Sketches Using Direct Shape Optimization. *IEEE Transactions on Image Processing*, vol. 29, pp. 8721–8734, 2020, doi: 10.1109/TIP.2020.3018865.

[Li18b] C. Li, H. Pan, Y. Liu, A. Sheffer, and W. Wang. Robust Flow-Guided Neural Prediction for Sketch-Based Freeform Surface Modeling. *ACM Trans. Graph. (SIGGRAPH ASIA)*, vol. 37, no. 6, p. 238:1-238:12, 2018, doi: 10.1145/3272127.3275051.

[Goe20a] P. Goel, L. Cohen, J. Guesman, V. Thamizharasan, J. Tompkin, and D. Ritchie. Shape From Tracing: Towards Reconstructing 3D Object Geometry and SVBRDF Material from Images via Differentiable Path Tracing. arXiv, 2020, doi: 10.48550/ARXIV.2012.03939.

[Li16a] C. Li, H. Lee, D. Zhang, and H. Jiang. Sketch-based 3D modeling by aligning outlines of an image. *Journal of Computational Design and Engineering*, vol. 3, no. 3, pp. 286–294, 2016, doi: 10.1016/j.jcde.2016.04.003.

[Zha21c] S.-H. Zhang, Y.-C. Guo, and Q.-W. Gu. Sketch2Model: View-Aware 3D Modeling from Single Free-Hand Sketches. arXiv, 2021, doi: 10.48550/ARXIV.2105.06663.

[Gao22a] C. Gao, Q. Yu, L. Sheng, Y.-Z. Song, and D. Xu. SketchSampler: Sketch-based 3D Reconstruction via View-dependent Depth Sampling. arXiv, 2022, doi: 10.48550/ ARXIV.2208.06880.

[Liu19a] S. Liu, T. Li, W. Chen, and H. Li. Soft Rasterizer: A Differentiable Renderer for Image-based 3D Reasoning. arXiv, 2019, doi: 10.48550/ARXIV.1904.01786.

[Sch16a] J. L. Schönberger and J.-M. Frahm-Structure-from-Motion Revisited. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 4104–4113, doi: 10.1109/CVPR.2016.445.

[Wan18a] L. Wang, C. Qian, J. Wang, and Y. Fang. Unsupervised Learning of 3D Model Reconstruction from Hand-Drawn Sketches. *Proceedings of the 26th ACM International Conference on Multimedia*, New York, NY, USA, 2018, pp. 1820–1828, doi: 10.1145/3240508.3240699.

[Fre07a] Fredo6. Vasque. 2007. URL: https://3dwarehouse.sketchup.com/model/de673ddf9df03b8278cf1a714198918/Vasque-in-Sketchu