7. Supplementary Material

7.1. External Contours

Our 2D Chamfer refinement objective for matching external contours requires an estimation of these contours. We here describe the simple algorithms we use to get them for the reconstructed mesh, and for the full input sketch.

7.1.1 External Contours of Reconstructed Shapes

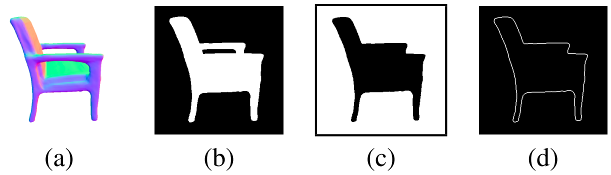
(a) (b) (c) (d)

Figure 9. **External contours of reconstructed shape: (a)** Initially reconstructed shape **(b)** Rendered foreground/background mask, **(c)** Flood-filling (b) from one image corner, and performing 1 pixel dilation of the flood-filled background **(d)** Taking the pixel-wise multiplication of (b) and (c) yields an exterior contour image, in which internal holes are ignored (the armrest for example here).

Given mesh $\mathcal{M}_\Theta$ and projection Λ, we render a $H \times W$ foreground/background mask. Then we flood fill the background, starting from one image corner, and apply morphological dilation to the result. As depicted on Fig. 9, taking the pixel-wise multiplication of this dilated flood filled background with the original foreground/background mask yields an external contour that ignores inner holes. We use this image for $\widetilde{\mathbf{F}}$ in Section 3.3.1.

7.1.2 External Contours of Input Sketches

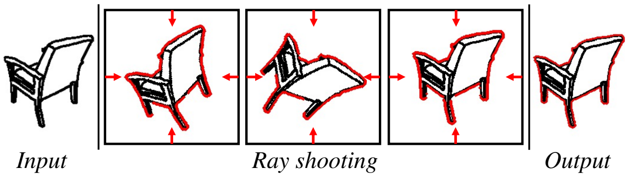

Figure 10. **External contours of input sketch:** We rotate the input sketch at various angles and shoot vertical and horizontal rays to only keep the first encountered pen stroke (red pixels). These pixels obtained at different rotation angles are then aggregated to yield the full external contour (shown in red on the last panel, superposed to the sketch).

Given an input sketch, we cannot apply the above method since line drawings might not be watertight. Instead, we apply an image-space only algorithm that extracts external contours, which can then be matched against the ones of the initial reconstruction.

As pictured in Fig. 2(c), we propose to do this by shooting rays from the image borders, at multiple angles, and only preserve the first encountered stroke for each ray. For the ease of implementation, in practice we shoot rays that are perpendicular to the image borders, but rotate the input image of $\pm\{0, 10, 20, 30, 35, 40, 45\}$ degrees and aggregate the resulting pixels at each angle. This is depicted in Fig. 10.

To achieve a relative invariance to pen size (free choice in our interface), we extract both the entry and exit pixels of the first pen stroke a ray encounters. In case the average distance over the whole image between the entry and exit pixels is greater than a threshold, we heuristically consider the line as thick and only keep the exit pixels - this corresponds to the inner shell of the external contour. Otherwise, we consider the line as thin, and keep the entry pixel.

7.2. Comparison of the two Refinement Approaches

7.2.1 Training on `SketchFD`

In the main paper, in Sec. 4.3 we present a comparison of *Sketch2Mesh/Render* and *Sketch2Mesh/Chamfer* approaches when applied on networks trained on `Suggestive` synthetic sketches, and tested on all 3 datasets. In Tab. 5 we present the same comparison, but this time for encoder/decoder pairs trained on `SketchFD`. Again, *Sketch2Mesh/Chamfer* appears to be more robust to style change and performs better than *Sketch2Mesh/Render* on datasets the latter has not been trained on.

7.2.2 Gradients and Sensitivity to Thin Components

In Fig. 11, we demonstrate how *Sketch2Mesh/Chamfer* is more sensitive to thin shape components such as chair legs. Indeed, removing a thin component only affects $\mathcal{L}_{F/B}$ of a few pixels, whereas $\mathcal{L}_{CD}$ takes into account the distance and spatial extent of the deformation.

7.2.3 Differentiable Rasterization Hyperparameters

In Figure 12, we show that the the choice of hyperparameters in the differentiable rendering process can deeply influence the refinement behavior of *Sketch2Mesh/Render*, particularly when the predicted binary masks are not accurate. Specifically, we investigate the importance of parameter `faces_per_pixel`, controlling how many triangles are used for backpropagation within each pixel: using a higher number of triangles will result in smoother gradients, as binary mask information is back-propagated to more faces. As depicted in Figure 12 this is particularly beneficial when predicted binary masks are not accurate or noisy, and results in more accurate reconstructions. In practice, we set `faces_per_pixel=25` in our experiments.

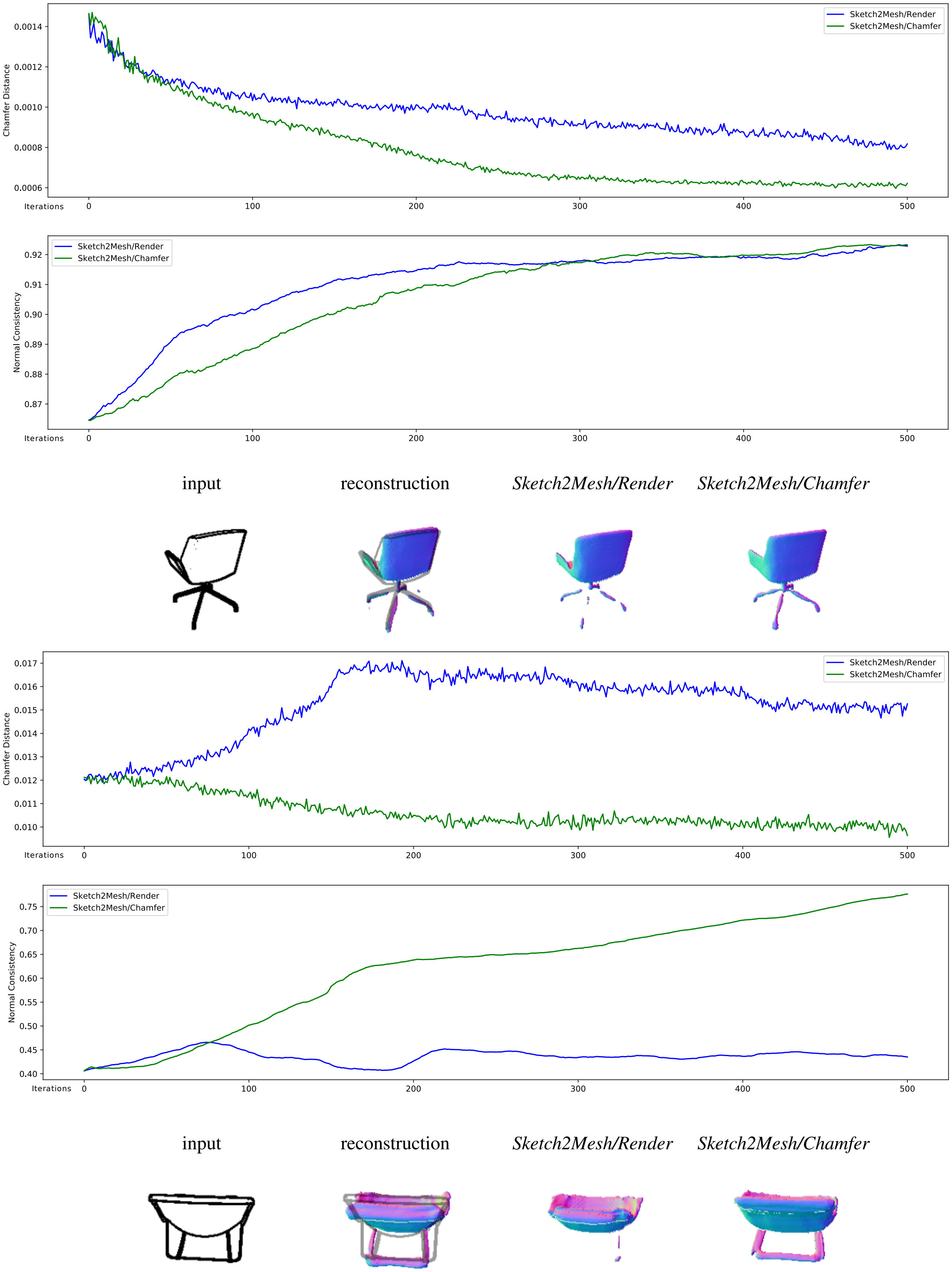

Figure 11. **Refinement.** *Sketch2Mesh/Chamfer* is more sensitive to thin shape components such as chair legs with respect to *Sketch2Mesh/Render* : this is due to the nature of the loss, penalizing chamfer distance between silhouettes, rather than per-pixel discrepancies. Best seen digitally, zoomed in.

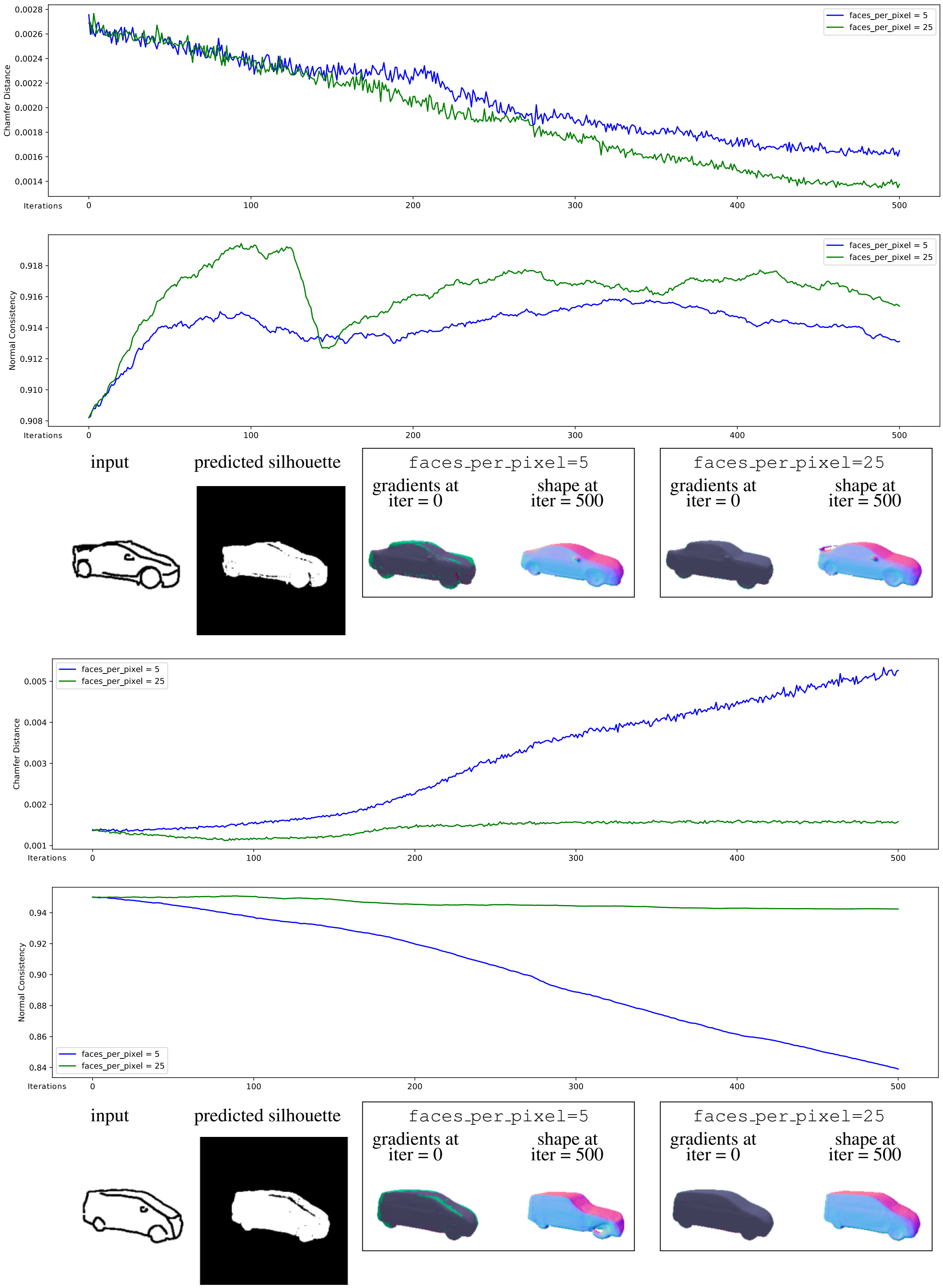

Figure 12. **Silhouette Alignment.** We compare different settings of pytorch3D [35] on two human-drawn car samples. Surface gradients are shown in color (green = intrusion along the surface normal, purple = extrusion).When considering a lower number of triangles for backpropagation of the rasterization process, gradients are more influenced by erroneous silhouette predictions, making refinement less effective (top) or even detrimental (bottom). Best seen digitally, zoomed in.

Metric	Method	Test Drawing Style		
		Suggestive	SketchFD	Hand-drawn
CD-$l_2 \cdot 10^3 \downarrow$	*Initial*	3.231	1.815	2.534
	Sketch2Mesh/Render	2.538	**1.515**	2.054
	Sketch2Mesh/Chamfer	**2.419**	1.516	**2.047**
Normal Consistency $\uparrow$	*Initial*	89.67	90.94	89.06
	Sketch2Mesh/Render	90.92	**92.34**	91.02
	Sketch2Mesh/Chamfer	**91.23**	92.09	**91.03**

Cars

Metric	Method	Test Drawing Style		
		Suggestive	SketchFD	Hand-drawn
CD-$l_2 \cdot 10^3 \downarrow$	*Initial*	12.290	7.770	17.395
	Sketch2Mesh/Render	10.761	**6.517**	16.091
	Sketch2Mesh/Chamfer	**9.524**	6.737	**12.585**
Normal Consistency $\uparrow$	*Initial*	76.76	80.49	63.11
	Sketch2Mesh/Render	80.39	**84.43**	68.67
	Sketch2Mesh/Chamfer	**81.00**	83.10	**70.49**

Chairs

Table 5. **Cars and Chairs.** Reconstruction metrics when using the encoding/decoding network trained on `SketchFD` synthetic sketches of cars and of chairs, and tested on all 3 datasets. We show *initial* results before refinement and then using our two refinement methods. Note that *Sketch2Mesh/Chamfer* does better than *Sketch2Mesh/Render* on the styles it has *not* been trained for, indicating a greater robustness to style changes.