


# Deep Sketch-Based Modeling of Man-Made Shapes

Authors: Smirnov, D.; Bessmeltsev, M.; Solomon, J.

Presenter: Cynthia Li





# Intro & Prep

# Introduction

- Being able to generate 3D models from sketches would be very useful for designers or amateurs to create contents quickly.
- This work generates 3D models of objects from a single 2D sketch. Parametrized surface patches are used to form the model while self-intersecting in surfaces are regularized.
- The authors also solves the problem of lacking data by generating sketch-like input data from 3D shapes. The results show that their model generalizes to natural sketches.

# Previous studies

## Sketch-based 3D modeling

- Many works progressively improve the 3D shape by the stroke added in each iteration, where users are usually required to annotate their strokes.
  - This process is stroke dependence so putting extra requirements on the drawing process.
- Some other works with 2D sketches but rely on strong 3D priors of the object designed by experts.
  - E.g. for animals or natural shapes, the prior would promote smooth, round, and symmetrical shapes, and human-made shapes are usually priored with combination of geometries.
  - This work is focused on man-made objects that have sharp edges and are only piece-wise smooth. Shape priors are also learned instead of designed.

# Data preparation

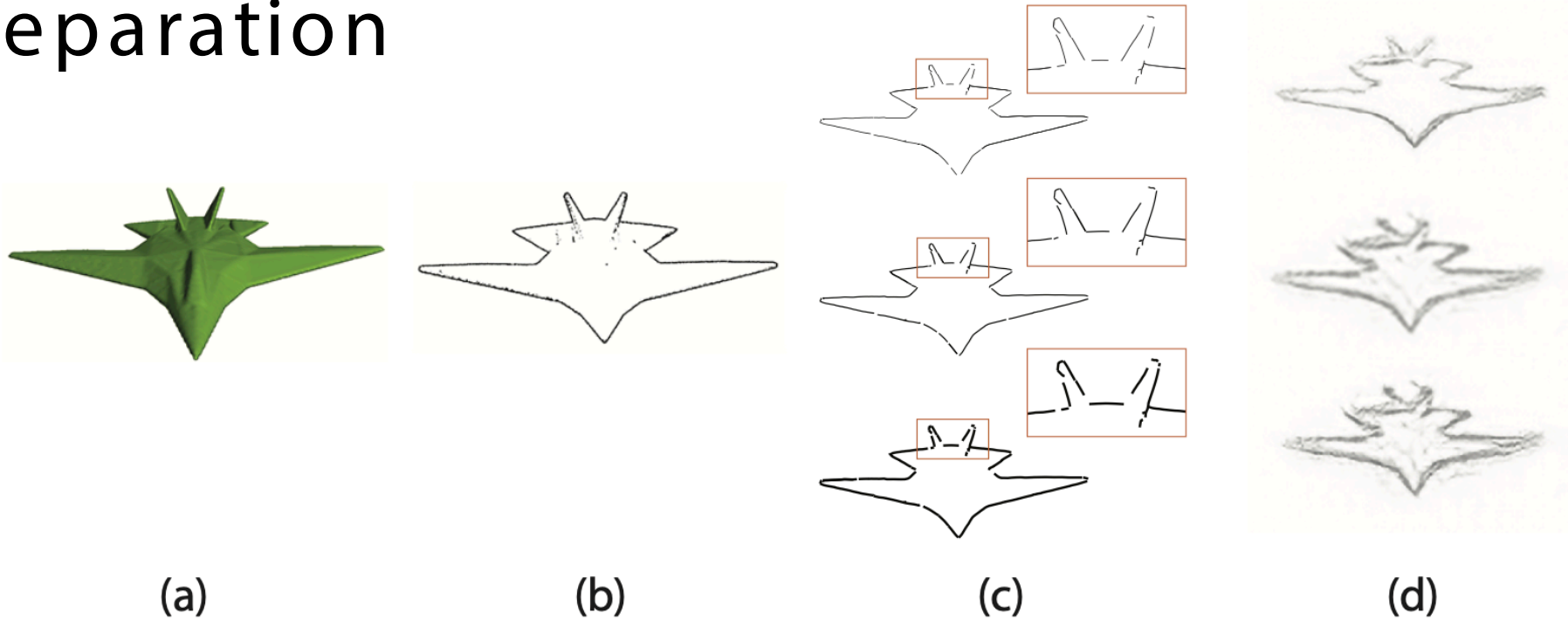


Fig. 2. Our data generation and augmentation pipeline. Starting with a 3D model (a), we use Autodesk Maya to generate its contours (b), which we vectorize using the method of Bessmeltsev and Solomon [2019] and stochastically modify (c). We then use the pencil drawing generation model of Simo-Serra et al. [2018] to generate the final image (d).



## Objects used:

- Airplane, bathtub, guitar, and knife



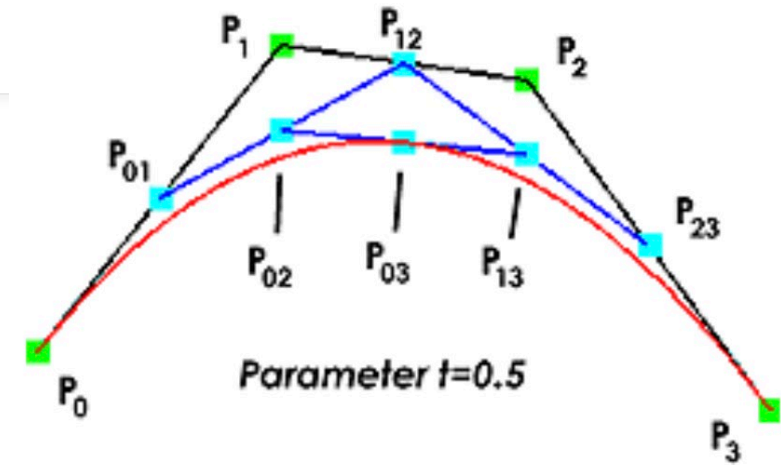
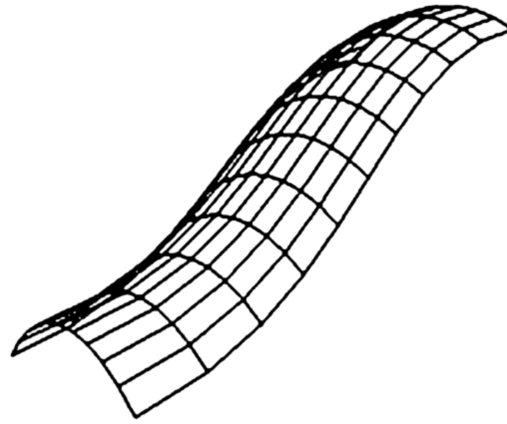
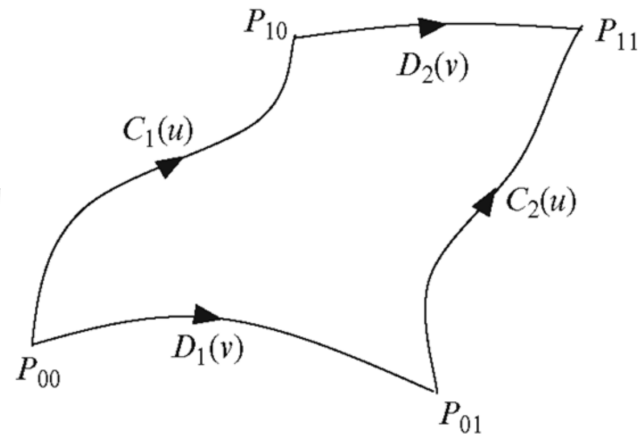
# Details



# Patch primitives

- Want to capture both smooth regions and sharp creases and corners.
- For this purpose, Coons patches are used as primitives.





A single Bézier curve  $c : [0, 1] \rightarrow \mathbb{R}^3$  is defined as

$$c(\gamma) = p_1(1 - \gamma)^3 + 3p_2\gamma(1 - \gamma)^2 + 3p_3\gamma^2(1 - \gamma) + p_4\gamma^3, \quad (1)$$

and a Coons patch  $P : [0, 1] \times [0, 1] \rightarrow \mathbb{R}^3$  is defined as

$$P(s, t) = (1 - t)c_1(s) + tc_3(1 - s) + sc_2(t) + (1 - s)c_4(1 - t) \\ - (c_1(0)(1 - s)(1 - t) + c_1(1)s(1 - t) + c_3(1)(1 - s)t + c_3(0)st). \quad (2)$$

Bézier curve with 4 control points

# Templates (shape priors)

- Used to specify the connectivity of a collection of Coons patches, and used as a hard topological constraints
- Templates consists of minimum number of control points to Coons patches (sharing control points on different surfaces)
- Have a distinct template for each category of objects

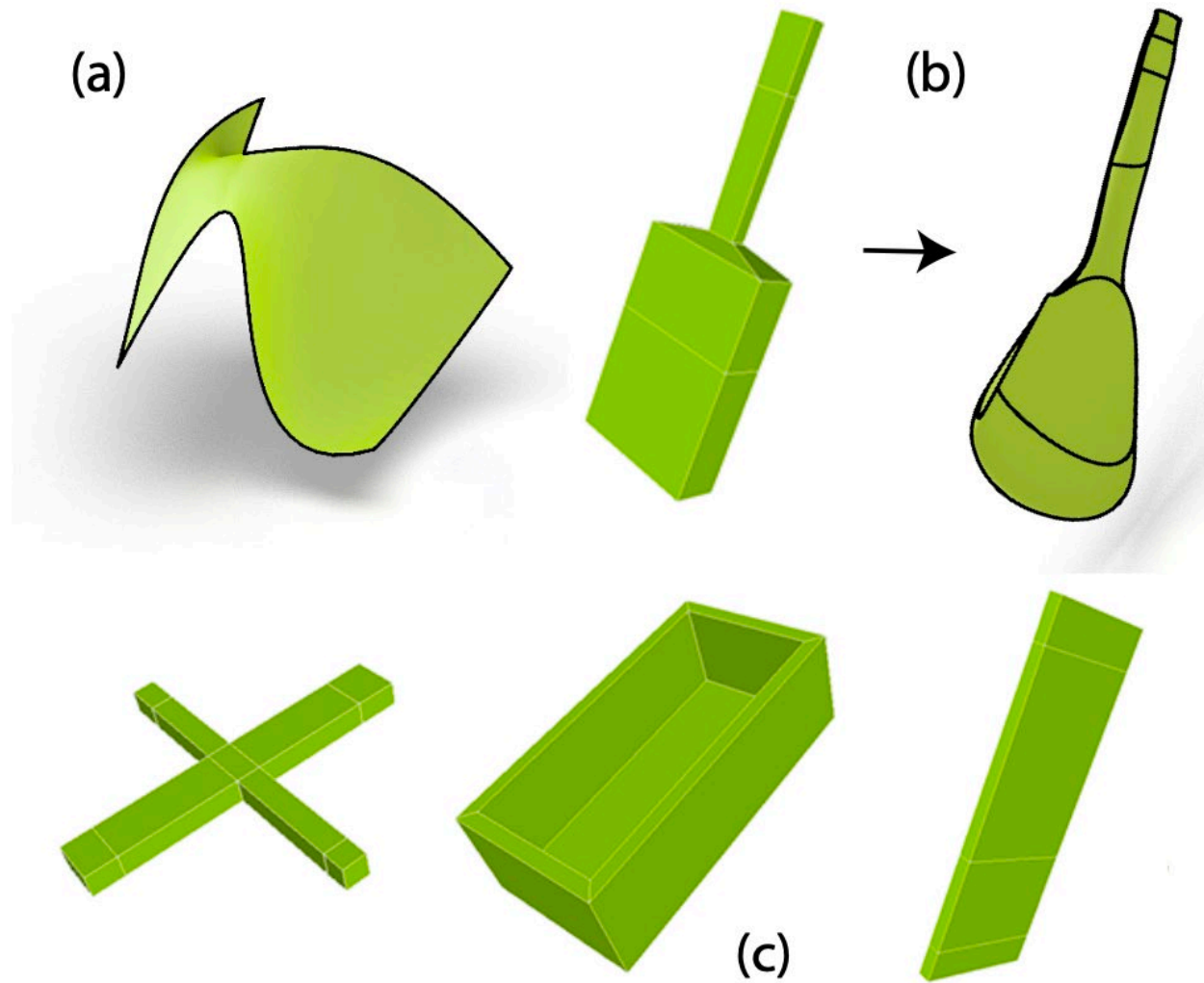


Fig. 3. Our geometry representation is composed of Coons patches (a) that are organized into a deformable template (b). We have designed a separate template per category of shapes: guitars (b), planes, bathtubs, and knives (c).



# Loss functions

# Reconstruction loss

- Area-weighted Chamfer distance, penalizes on reconstruction discrepancy  $\text{Ch}_{\text{dir}}(X, Y) = \sum_{x \in X} \min_{y \in Y} d(x, y)$ ,
- Sampling uniformly from Coons patches is hard, reparametrize to sample from original unit square

$$\text{Ch}_{\text{dir}}^{\text{var}}(P, M) = \int_P \inf_{y \in M} d(x, y) dx \quad (6)$$

$$= \frac{1}{\text{Area}(P)} \int_P \inf_{y \in M} d(x, y) dS \quad (7)$$

$$= \frac{\mathbb{E}_{(u, v) \sim \mathcal{U}_{\square}} [\inf_{y \in M} d(P(u, v), y) |\det J|]}{\mathbb{E}_{(u, v) \sim \mathcal{U}_{\square}} [|\det J|]}, \quad (8)$$

# Normal alignment

- Align curvature of reconstructed 3D model
- $x$  is point closest to sampled target point  $y$  under Euclidean distance

$$d_N(x, y) = 1 - \langle n_x, n_y \rangle^2,$$

# Intersection regularization

- Trained 2 MLP models with generated samples with patches
- One to classify self-intersecting patches, the other to classify intersecting surfaces

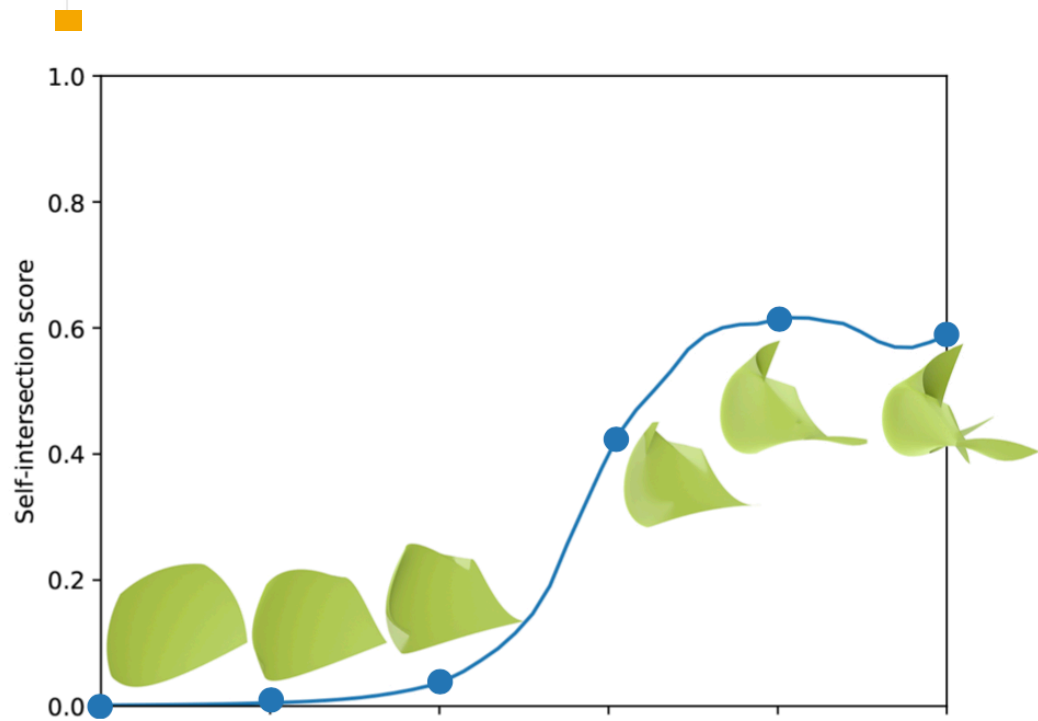


Fig. 4. Intersection scores predicted by the self-intersection MLP evaluated on 50 patches linearly interpolated between a flat patch and a patch with several self-intersections. The score increases as the patch becomes “more self-intersecting.” Six out of the 50 patches, including the initial and final patches, are displayed.



Fig. 5. Example patches misclassified by the self-intersection MLP. Two false positives (incorrectly predicted to be self-intersecting) are shown in green, and two false negatives are shown in orange.



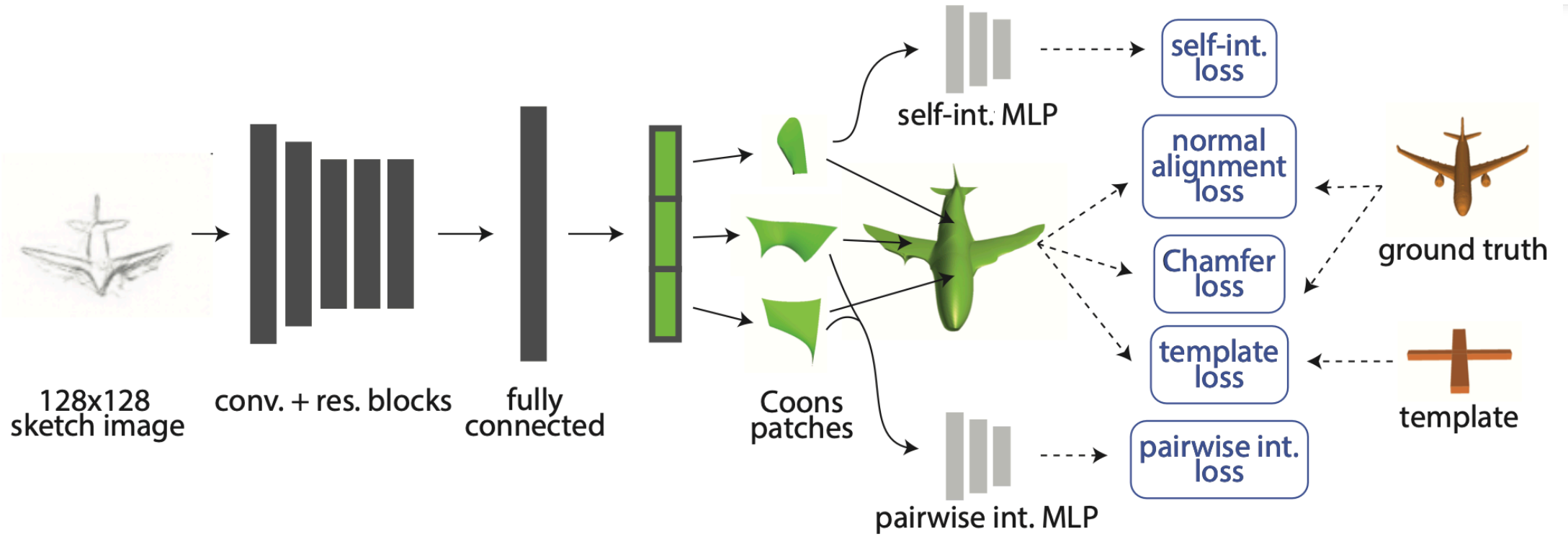
# Template initialization loss

- Fit the generated model to the template as a initialization, and reduces its effect later

$$\mathcal{L}_{\text{template}}(\{P_i\}) = \sum_i \gamma^{(t/s)} \|P_i - T_i\|_2^2,$$



# Pipeline





# Results



Fig. 7. Results on synthetic sketches taken from our test datasets for bathtubs, guitars, and knives.

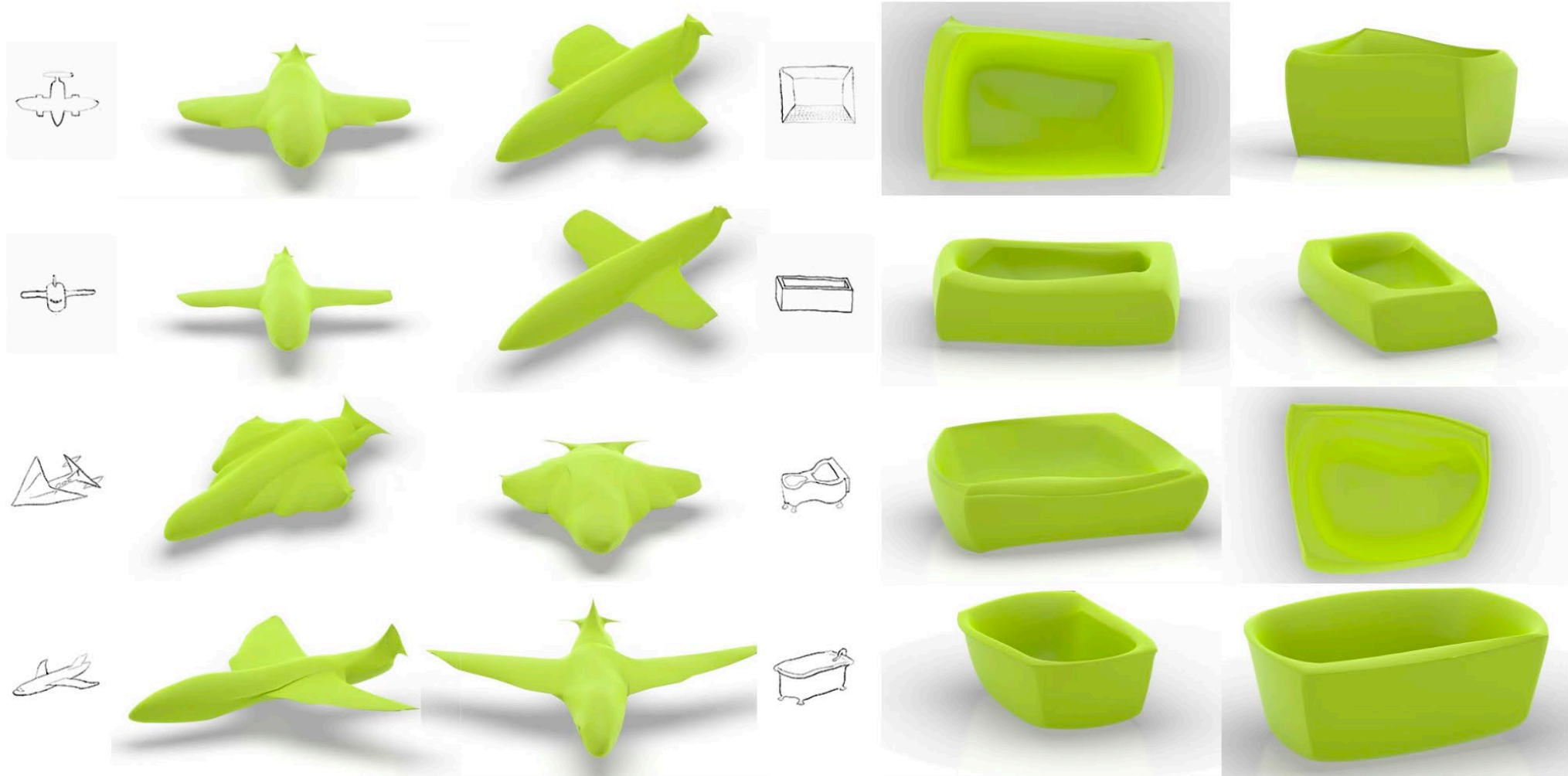


Fig. 8. Results on real human-drawn sketches. The top two rows are sketches are drawn on pencil and paper and scanned while the bottom two rows are drawn on iPad. Each artist was shown the sample sample 3D models rendered from several viewpoints but was not provided with sample sketches or given instructions on how to draw the sketches.



# Ablation studies

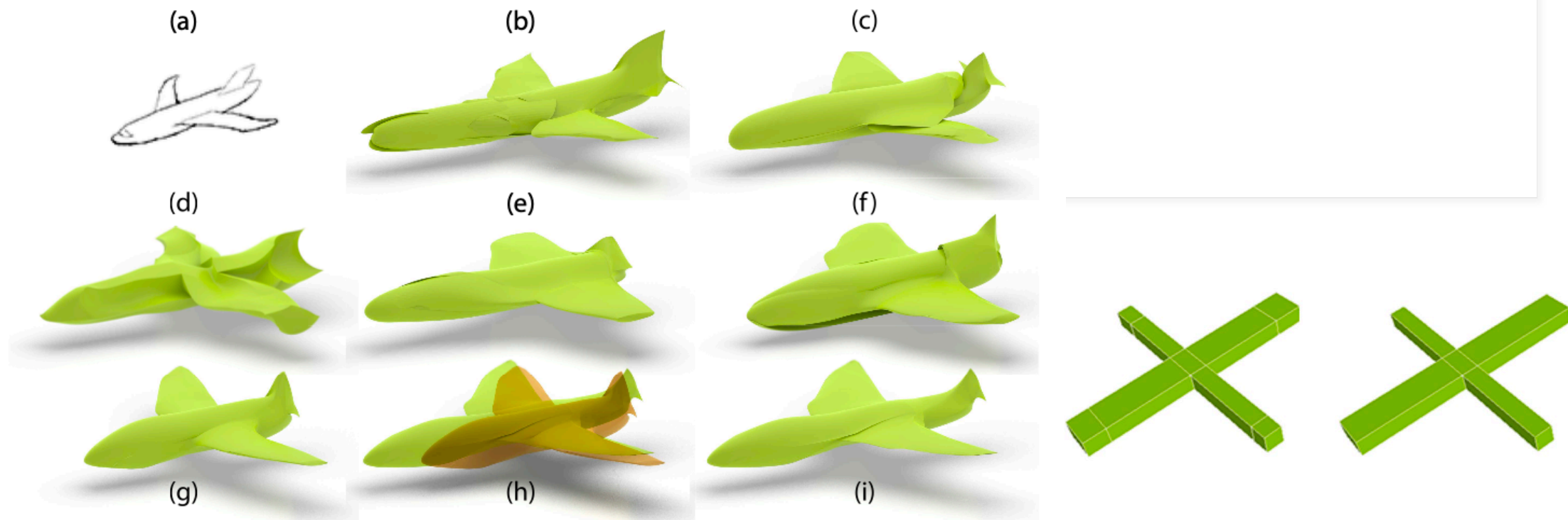


Fig. 9. For a human-drawn sketch (a), we perform an ablation study of our algorithm, training the network (b) without the self-intersection loss, (c) without pairwise intersection loss, (d) without normal loss, or using a simple template (e). We also study the effects of various data augmentation stages (§3) by training the network: (f) only on contour renders without any augmentation, (g) with the sketch filter, but no vector augmentation. In (h), we overlay (g, shown in brown) with the final result (i).





Thank you!



# Shape representation for this problem

- Voxels: most popular choice, but usually have low resolution due to memory limitations, which makes it hard to present surfaces
- Point-based: no memory issue, but do not capture connectivity
- Mesh-based: hard to make the diffeomorphic operations differentiable, but possible if use a parametrization on the meshes
- Other representations

# Data preparation

- Because of the lack of data, the authors generated sketches from complete 3D shapes from ShapeNet.
- Rendered the object from fixed number of distinct camera views, and get the contour of the object.
- Then the set of contours is augmented to mimic the ambiguity when sketching:
  - Made broken lines by adding gaps at random position
  - Truncate some curves randomly
  - Remove curves that are too short
  - Using pencil drawing generation model to generate sketching texture



