Touch2Shape: Touch-Conditioned 3D Diffusion for Shape Exploration and Reconstruction

Supplementary Material

In this supplement material, we supply additional information, experiments, and results. This includes the network architecture (Section A), the details of dataset processing and experiment settings (Section B), additional experiment results and some failure cases (Section C).

A. Network Details

As illustrated in Figure 2, our system comprises 8 key modules: the VQVAE model, touch CNN, touch embedding, contrastive touch encoder, vision embedding, diffusion model, touch shape fusion module, and policy model. The VQVAE model follows [5, 38], while the touch chart prediction model is based on method [33, 34]. Details can be found in the original papers of these methods. In the subsequent section, we will delineate the network architectures and training details for the remaining modules.

Touch Embedding. First, we use the touch CNN model [33, 34] to generate local touch charts, and then transfer the local charts to the world coordinate system based on the touch location parameters. we assume that we can obtain up to 20 tactile images, where each tactile image generates a chart as a tensor of size 25×4 (each vertex contains coordinates x, y, z, and touch status, the numnber of touch chart vertex is 25). By merging all the vertices of these charts together, we create a tensor of size $20 \times 25 \times 4$. If we have fewer than 20 tactile images, the coordinates of the vertices for the remaining charts are zeroed out. As shown in Figure 7 (a), we first apply position encoding to the centroid of each chart, then apply two CNN blocks on the touch charts to extract vertex features. After max pooling operations and adding position encoding, we finally obtain 20 tokens. The components of CNN blocks and position embedding module is illustrated in table 6.

Contrastive Touch Encoder training. The contrastive touch encoder is similar to the touch encoder shown in Figure 7 (a), with the addition of a max pooling layer at the end. The output size of contrastive touch encoder is 1×768. The latent encoder (Figure 7 (b)) takes the encoder latent vector (of size $16 \times 16 \times 16 \times 3$) as input, outputs the shape feature of size 1×768. The components of latent encoder is illustrated in table 6. We train the contrastive encoder and latent encoder using moco [17]. The queue size is 6,000, temperature parameter τ is 0.07, we train these encoders for 1 million iterations with batch size of 48.

Visual-tactile Setting. We employ ResNet18 [16] as the visual backbone. By utilizing the feature maps (of size $8 \times 8 \times 512$) from the fourth layer as input, we employ a linear layer to transform the features to a size of $8 \times 8 \times 768$. Each pixel is considered as a token, resulting in 64 visual tokens. Following the approach of SDFusion [5], we leverage dropout operations on both visual and tactile tokens to facilitate classifier-free guidance.

Diffusion Model. The denoising network is a U-Net like SDFusion [5]. During the training phase, we randomly select the number of grasp less than the maximum value in each batch. The maximum timestamps is 1,000 in the training phase. In the testing phase, the timestamp is set to 50, and the unconditional guidance scale is 5.

Touch Shape Fusion. The network is depicted in Figure 3. The architecture of the encoder layers resembles the encoder used in VQVAE [5, 38]. The layers of the voxel encoder and the VQVAE decoder are listed in Table 6. We extract the feature maps generated from the input convolution, down block 1, and the middle block. The output shapes are $64 \times 64 \times 64 \times 64$, $32 \times 32 \times 32 \times 256$, and $16 \times 16 \times 16 \times 256$, respectively. We pass the feature map of each encoder and decoder layer through a $1 \times 1 \times 1$ convolution layer. These encoded features, along with the features from the shape decoder layers are then calculated using formula 4 and processed through an additional $1 \times 1 \times 1$ convolution layer to obtain the output feature map.

Policy Model. Initially, we employ the latent encoder to encode the initial and current latent vector from the touch-conditioned diffusion model. Subsequently, we construct an action embedding module, as depicted in Figure 7 (c) and Table 6, to derive the action embedding. By combining these three vectors, we use fully connected layers to predict the value associated with each action. Each action is identified by its positional index on a sphere consisting of 50 actions, as described in [34]. The episode ends after executing maximum grasps (with a maximum of 4 tactile images for a 4-fingered hand in each grasp).

B. Dataset and Experiment Settings

We validate our model using two datasets. The first dataset utilized is derived from [33, 34], built upon the ABC dataset [21]. This dataset is used to compare experiment results with ActiveVT [34] and VTRecon [33]. The second dataset utilized is from [7], built upon the ShapeNet dataset [3], and is used for comparing experimental results with TouchSDF [7].

dataset ABC. This dataset consists of 40,000 objects with unclear class definitions and varied shapes, posing a significant challenge for generalization. Objects that

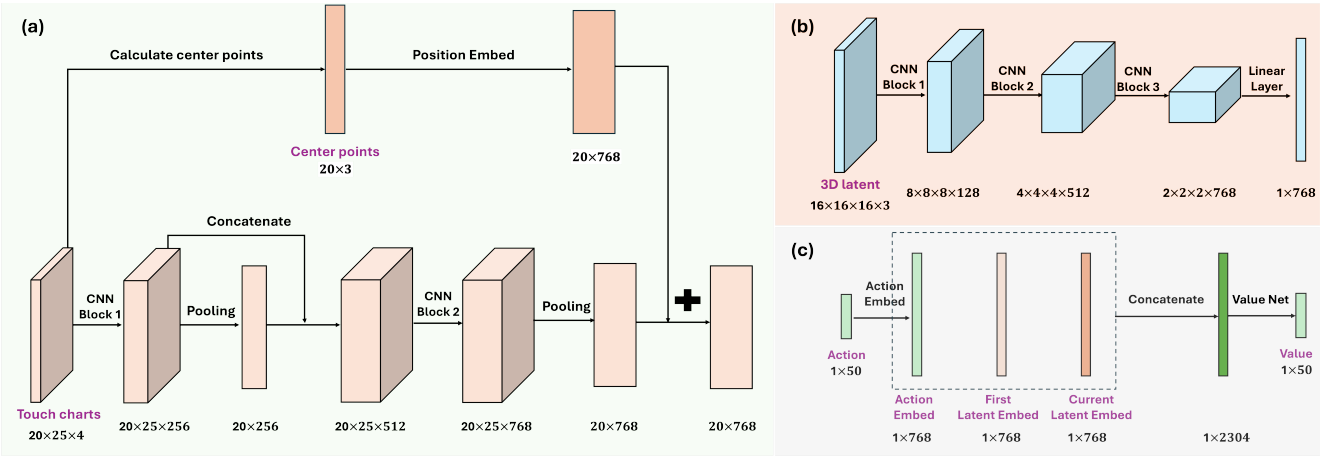

Figure 7. The network architectures of (a) the touch embedding module, (b) the latent encoder and (c) the policy model.

couldn't be reduced to a specific size due to geometric constraints or those with multiple disconnected parts were excluded, resulting in a set of 26,545 usable object models. These objects were divided into 5 sets: 3 training sets, each containing 7,700 objects, a validation set with 2,000 objects, and a test set of 1,000 objects. To acquire TSDF volume, we normalize the object mesh and compute the SDF values to construct a volume with a resolution of $64 \times 64 \times 64$, using a truncation threshold of 0.2. The simulation environment of ActiveVT [34] is employed to grasp objects and capture touch signals. The object mesh is normalized and reduced in size by a factor of 3.1 before being placed in the scene with the same position and orientation. A four-fingered hand in the simulator grasps the object, and the finger pose information during the grasp is used to position a simulated camera with the same orientation. A depth image is generated from this camera to produce a simulated touch signal. For generating vision images, the object is assigned a random color texture and positioned in an empty scene with four fixed point lights. The resulting images are of size $256 \times 256 \times 3$. Further details can be found in the appendix of ActiveVT [34]. Note that the predicted chart vertices should be scaled up by a factor of 3.1 before inputting to the touch condition module.

dataset ShapeNet. TouchSDF [7] collected a dataset consisting of 1,650 objects which are divided into three subsets: 1,100 objects for training, 200 for validation and 350 for testing. The chosen objects span six diverse categories considered previously in related work: bowls, bottles, cameras, jars, guitars and mugs. To assess the accuracy of the reconstructed shapes, they also considered 300 unseen objects and poses across these six categories. The process of creating T-SDF volumes and vision images are the same with the methodology used for the ABC dataset. For tactile images, we follow the setting of TouchSDF [7] which captures tactile image by poking the target object for fair comparison. In this setting, at most one valid tactile image can be obtained per touch action.

Evaluation Metrics. As illustrated in Section 4.2, we use CD metric to evaluate the results with ActiveVT [34] and VTRecon [33] on dataset ABC and use EMD metric to evaluate the results with TouchSDF [7] on dataset ShapeNet. It's reported in TouchSDF [7] that they achieves better EMD but lower CD. To compute the CD error, we run marching cubes to get the object meshes and extract 30,000 points uniformly from each sample. The formula for computing the CD error is as follows:

$$CD(P,G) = \lambda \cdot \frac{1}{|P|} \sum_{p \in P} \min_{g \in G} \|p - g\|^2 + \lambda \cdot \frac{1}{|G|} \sum_{g \in G} \min_{p \in P} \|p - g\|^2, \tag{7}$$

where p and g are point in poinstets P sampled on predicted mesh surfaces and point ground truth pointsets G respectively. The coefficient λ is set to 9,000. Note that the scale is reduced by 3.1 times in ActiveVT [34]. So we reduce the scale of extracted points by 3.1 times to match the ground truth scale before we computed CD error.

EMD error can be computed as follows:

$$EMD(P,G) = \min_{\phi: P \to G} \sum_{p \in P} ||p - \phi(p)||_2, \tag{8}$$

where $\phi : P \to G$ is a bijection. Note that the computation of TouchSDF [7] is conducted on normalized objects, hence when compared with TouchSDF [7] using the EMD metric, we no longer reduce the generated point cloud scale.

C. Additional Results

Additional Results on ABC. The visualization results of visual-tactile settings are shown in Figure 4. In Figure

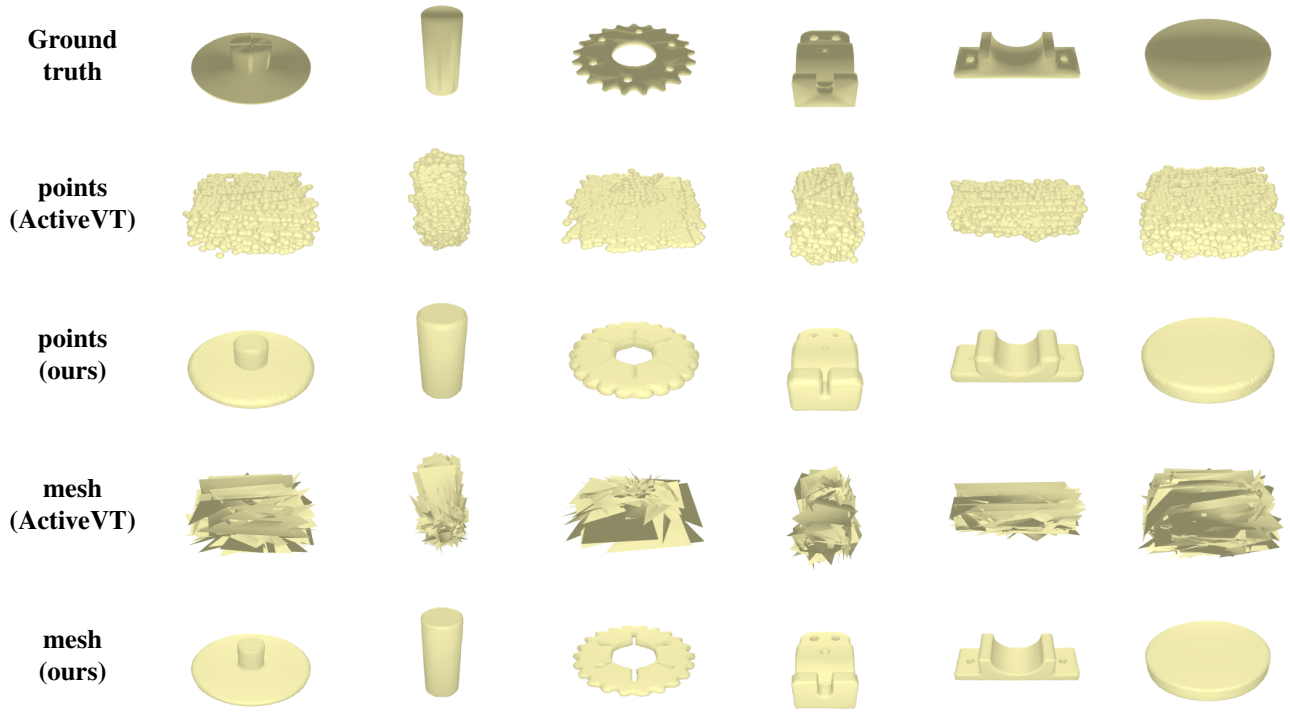

Figure 8. Qualitative results of ActiveVT [34] and ours under the tactile only setting (5 grasps). While ActiveVT struggles with visualizations and detail preservation, our method excels in maintaining global shape across diverse structures, and ensuring the local details.

8, we present the reconstruction visualization of tactile only settings (5 grasps). While ActiveVT [34] tends to generate awful visualizations on mesh surfaces and point cloud generation, it does manage to retain shapes similar to the ground truth for some structurally simple objects. However, when dealing with complex shapes or objects with cavities, it tends to lose many details. On the other hand, our approach excels in preserving overall global shape output across various shapes and delivers satisfactory results in local details as well.

Additional Results on ShapeNet. The mesh reconstruction for both seen and unseen objects, acquired from 20 randomly sampled touches, is illustrated in Figure 9. These visualizations showcase the model's capability to predict not only the overall structure but also to preserve intricate local details. Across all experimental results, it is clear that our model yields promising reconstruction on two distinct datasets, ShapeNet and ABC, underscoring its capacity for generalization.

Policies. Table 5 displays the changes in the ratio of the CD error compared to the initial CD with the increasing number of grasps under different strategies (under tactile only setting). The oracle strategy serves as the upper bound for all strategies as the true optimal policy cannot be computed in a reasonable time frame. It can be observed that the even algorithm may initially achieve better reconstruction, but as the grasps progress, our method is able to achieve greater improvements in reconstruction, validating that the learned policies is able to select more beneficial grasps. The evolution of the reconstructed shape from our model with an increasing number of grasps and relative predicted touch points can be found in Figure 5.

Failure Cases. Some failure cases under visual-tactile setting are shown in Figure 10. In these cases, although we can ensure the basic global structure, our method performs poorly on some local holes and screw threads. The main reasons may include the potential structural variations and complicated local details of these shapes, or the lack of symmetry in some hole positions, making it difficult to explore the target object.

Method	Grasp #					
	0	1	2	3	4	5
Oracle	100	12.0	6.86	5.01	4.90	4.88
Random	100	21.9	13.5	10.5	8.82	8.14
Even	100	15.3	10.2	9.13	8.23	7.44
Ours	100	16.9	9.62	7.62	6.88	6.63

Table 5. Comparison of touch exploration policy on dataset ABC under touch only setting. The evaluation metric is CD error.

Model	Module	Input shape	Operation	Output shape
Touch Embed	**CNN Block 1**	$20 \times 25 \times 4$	Conv (1×1) + BN + ReLU	$20 \times 25 \times 128$
		$20 \times 25 \times 128$	Conv (1×1) + BN + ReLU	$20 \times 25 \times 256$
	CNN Block 2	$20 \times 25 \times 512$	Conv (1×1) + BN + ReLU	$20 \times 25 \times 1024$
		$20 \times 25 \times 1024$	Conv (1×1) + BN + ReLU	$20 \times 25 \times 768$
	Position Embed	20×3	FC + ReLU	20×128
		20×128	FC	20×768
Latent Encoder	**CNN Block 1**	$16 \times 16 \times 16 \times 3$	Conv $(3 \times 3 \times 3)$ + BN + ReLU	$16 \times 16 \times 16 \times 128$
		$16 \times 16 \times 16 \times 128$	Conv $(3 \times 3 \times 3)$ + BN + ReLU	$16 \times 16 \times 16 \times 128$
		$16 \times 16 \times 16 \times 128$	Average Pooling $(2 \times 2 \times 2)$	$8 \times 8 \times 8 \times 128$
	CNN Block 2	$8 \times 8 \times 8 \times 128$	Conv $(3 \times 3 \times 3)$ + BN + ReLU	$8 \times 8 \times 8 \times 512$
		$8 \times 8 \times 8 \times 512$	Conv $(3 \times 3 \times 3)$ + BN + ReLU	$8 \times 8 \times 8 \times 512$
		$8 \times 8 \times 8 \times 512$	Average Pooling $(2 \times 2 \times 2)$	$4 \times 4 \times 4 \times 512$
	CNN Block 3	$4 \times 4 \times 4 \times 512$	Conv $(3 \times 3 \times 3)$ + BN + ReLU	$4 \times 4 \times 4 \times 512$
		$4 \times 4 \times 4 \times 512$	Conv $(3 \times 3 \times 3)$ + BN + ReLU	$4 \times 4 \times 4 \times 512$
		$4 \times 4 \times 4 \times 512$	Average Pooling $(2 \times 2 \times 2)$	$2 \times 2 \times 2 \times 512$
	Linear Layer	$2 \times 2 \times 2 \times 512$	Reshape	1×4096
		1×4096	FC	1×768
Touch Shape Fusion	**Encoder Layers**	$64 \times 64 \times 64 \times 1$	Input Convolution	$64 \times 64 \times 64 \times 64$
		$64 \times 64 \times 64 \times 64$	Down Block 1	$32 \times 32 \times 32 \times 256$
		$32 \times 32 \times 32 \times 256$	Down Block 2,3,4 + Middle Block	$16 \times 16 \times 16 \times 256$
	Decoder Layers	$16 \times 16 \times 16 \times 3$	Convolution + Middle Block	$16 \times 16 \times 16 \times 256$
		$16 \times 16 \times 16 \times 256$	Up Block 1	$32 \times 32 \times 32 \times 256$
		$32 \times 32 \times 32 \times 256$	Up Block 2,3,4	$64 \times 64 \times 64 \times 64$
Policy Model	**Action Embed**	1×50	FC + ReLU	1×128
		1×128	FC + ReLU	1×256
		1×256	FC	1×768
	Value Net	$(1 \times 768) \times 3$	Concatenating	1×2304
		1×2304	FC + ReLU	1×512
		1×512	(FC + ReLU) $\times 3$	1×128
		1×128	FC	1×50

Table 6. Architecture of each model. Conv: convolution layers with different kernel size, BN: batch normalization layers, ReLU: Rectified Linear Unit, FC: Fully Connected layers.

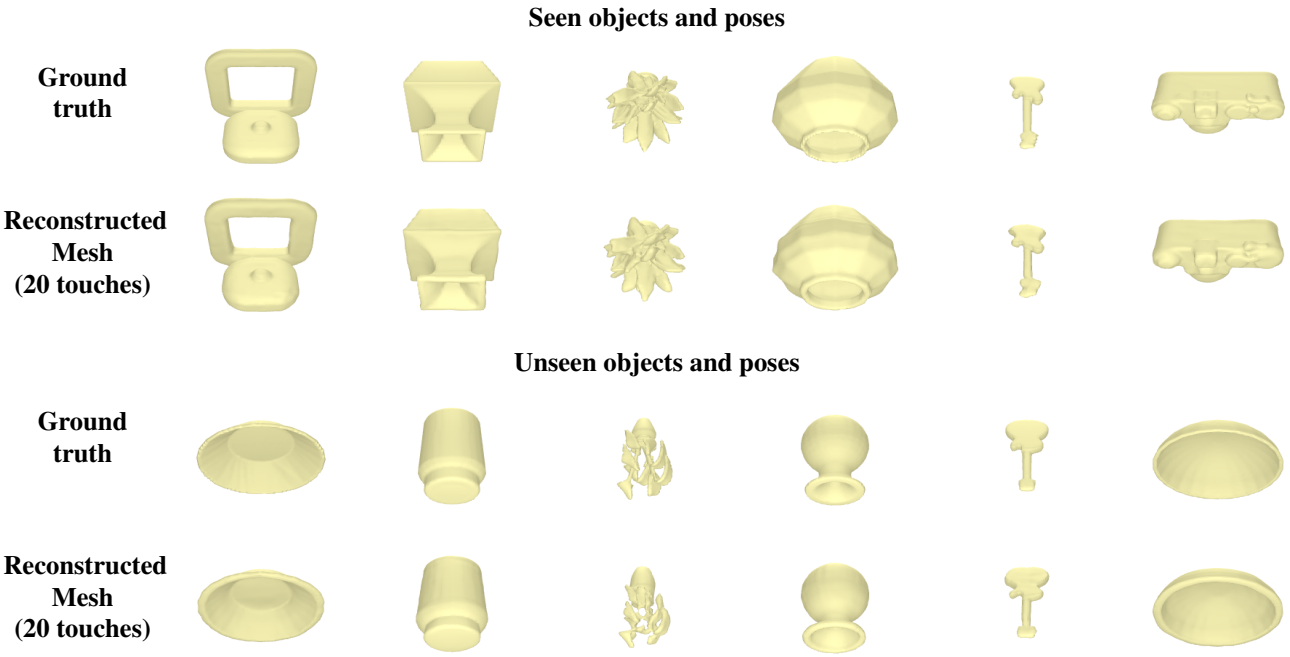

Figure 9. Reconstruction visualizations of seen and unseen objects on dataset ShapeNet.

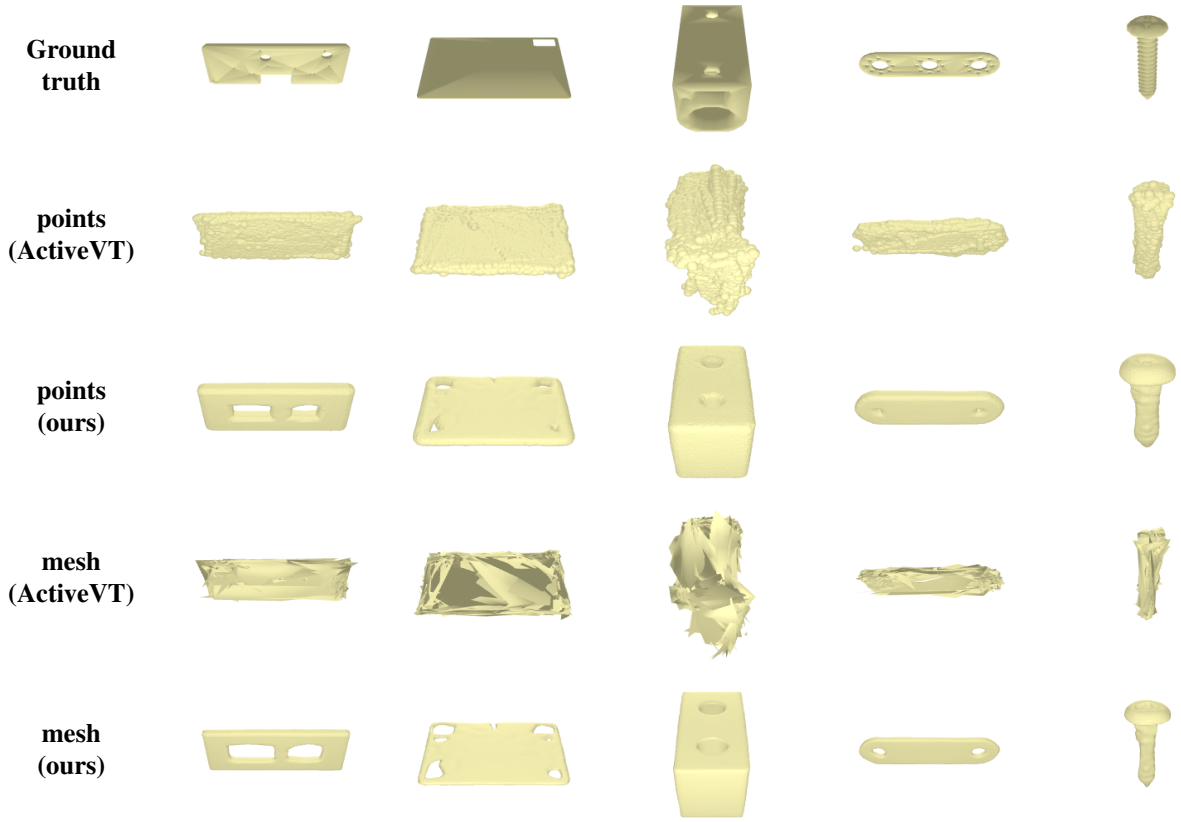

Figure 10. Some failure cases.