Group Meeting - March 5, 2021

Paper review & Research progress

Truong Son Hy *

*Department of Computer Science
The University of Chicago

Ryerson Physical Lab

Content

- Literature review:
  - **ContraGAN: Contrastive Learning for Conditional Image Generation** (NeurIPS 2020), `https://arxiv.org/abs/2006.12681`
  - **PointGMM: a Neural GMM Network for Point Clouds** (CVPR 2020), `https://arxiv.org/abs/2003.13326`
- Next time:
  - **Energy-Based Processes for Exchangeable Data**, `https://arxiv.org/abs/2003.07521`

Paper 1

ContraGAN: Contrastive Learning for Conditional Image Generation

(NeurIPS 2020)

Minguk Kang, Jaesik Park

https://arxiv.org/abs/2006.12681

https://github.com/POSTECH-CVLab/PyTorch-StudioGAN

Background – (Conditional) GANs

GANs

The objective of the adversarial training between the discriminator (D) and the generator (G) is as:

$$\min_G \max_D \mathbb{E}_{\boldsymbol{x}\sim p_{\text{real}}(\boldsymbol{x})}[\log(D(\boldsymbol{x}))] + \mathbb{E}_{\boldsymbol{z}\sim p(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

where $p_{\text{real}}(\boldsymbol{x})$ is the real data distribution, and $p(\boldsymbol{z})$ is a predefined prior distribution (e.g. multivariate Gaussian). Nash equilibrium is usually hard to achieve, usually requires regularization and tricks.

Conditional GANs

Utilizing class label information to advance the performance: concatenate the latent vector with the label to manipulate the semantic characteristics of the generated image.

Background – Contrastive learning

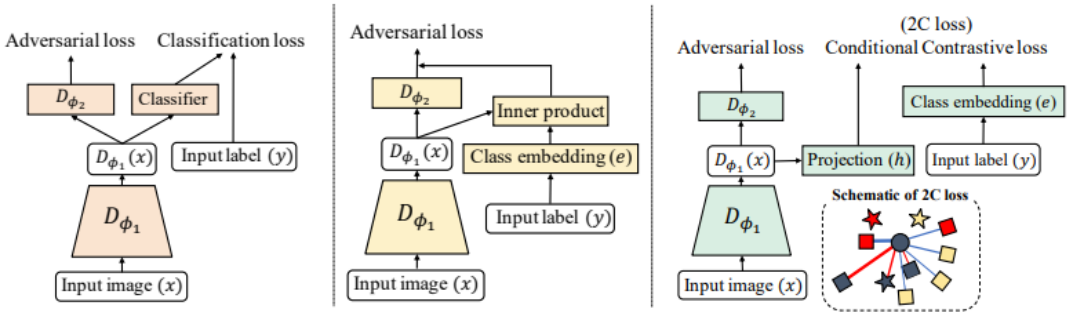
Given a minibatch of images $\boldsymbol{X} = \{\boldsymbol{x}_1, .., \boldsymbol{x}_m\}$ and labels $\boldsymbol{y} = \{y_1, .., y_m\}$. We have a neural network encoder $S(\boldsymbol{x}) \in \mathbb{R}^k$ and a projection layer $h : \mathbb{R}^k \to \mathbb{R}^d$. Let $\ell = h(S(\cdot))$. Contrastive learning considers a transformation / data-augmentation T on each individual example of $\boldsymbol{X}$:

$$\boldsymbol{A} = \{\boldsymbol{x}_1, T(\boldsymbol{x}_1), .., \boldsymbol{x}_m, T(\boldsymbol{x}_m)\} = \{\boldsymbol{a}_1, .., \boldsymbol{a}_{2m}\}$$

NT-Xent loss:

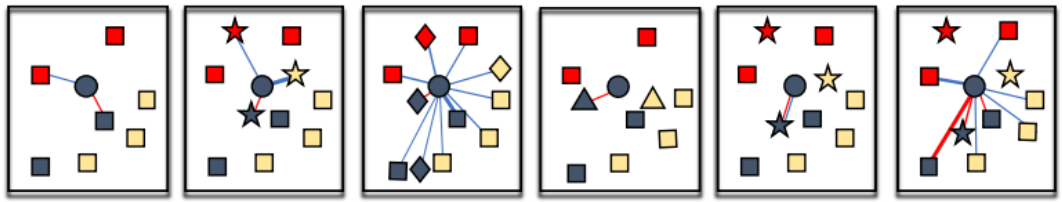
$$\ell(\boldsymbol{a}_i, \boldsymbol{a}_j; t) = \log\left(\frac{\exp(\ell(\boldsymbol{a}_i)^T \ell(\boldsymbol{a}_j)/t)}{\sum_{k=1}^{2m} 1_{k\neq i} \exp(\ell(\boldsymbol{a}_i)^T \ell(\boldsymbol{a}_k)/t)}\right)$$

Overview



(a) ACGAN [19] (b) ProjGAN [17] (c) ContraGAN (Ours)

Figure 1: Schematics of discriminators of three conditional GANs. (a) ACGAN [19] has an auxiliary classifier to guide the generator to synthesize well-classifiable images. (b) ProjGAN [17] improves ACGAN by adding the inner product of an embedded image and the corresponding class embedding. (c) Our approach extends ACGAN and ProjGAN with a conditional contrastive loss (2C loss). ContraGAN considers multiple positive and negative pairs in the same batch. ContraGAN utilizes 2C loss to update the generator as well.



(a) Triplet [32] (b) P-NCA [33] (c) NT-Xent [34] (d) ACGAN [19] (e) ProjGAN [17] (f) 2C loss (Ours)

Figure 2: Illustrative figures visualize metric learning losses (a,b,c) and conditional GANs (d,e,f). The color indicates the class label and the shape represents the role. (Square) the embedding of an image. (Diamond) the embedding of an augmented image. (Circle) a reference image embedding. Each loss is applied to the reference. (Star) the embedding of a class label. (Triangle) the one-hot encoding of a class label. The thicknesses of red and blue lines represent the strength of pull and push force, respectively. The loss function of ProjGAN lets the reference and the corresponding class embedding be close to each other when the reference is real, but it pushes far away otherwise. Compared to ACGAN and ProjGAN, 2C loss can consider both data-to-class and data-to-data relations between training examples.

Method

Propose to use the **embeddings of class labels** instead of using data augmentations. With a class embedding function $e(y) : \mathbb{R} \to \mathbb{R}^d$:

$$\ell(\boldsymbol{x}_i, y_i; t) = -\log\left(\frac{\exp(\ell(\boldsymbol{x}_i)^T e(y_i)/t)}{\exp(\ell(\boldsymbol{x}_i)^T e(y_i)/t) + \sum_{k=1}^{m} 1_{k\neq i} \exp(\ell(\boldsymbol{x}_i)^T \ell(\boldsymbol{x}_k)/t)}\right)$$

that pulls a reference sample $\boldsymbol{x}_i$ nearer to the class embedding $e(y_i)$ and pushes the others away. The final loss function:

$$\ell_{2C}(\boldsymbol{x}_i, y_i; t) = -\log\left(\frac{\exp(\ell(\boldsymbol{x}_i)^T e(y_i)/t) + \sum_{k=1}^{m} 1_{y_k=y_i} \exp(\ell(\boldsymbol{x}_i)^T \ell(\boldsymbol{x}_k)/t)}{\exp(\ell(\boldsymbol{x}_i)^T e(y_i)/t) + \sum_{k=1}^{m} 1_{k\neq i} \exp(\ell(\boldsymbol{x}_i)^T \ell(\boldsymbol{x}_k)/t)}\right)$$

that reduces the distances between the embeddings of images with the same labels.

Algorithm

Algorithm 1 : Training ContraGAN

Input: Learning rate: α_1, α_2. Adam hyperparameters [41]: β_1, β_2. Batch size: m. Temperature: t. # of discriminator iterations per single generator iteration: n_{dis}. Contrastive coefficient: λ. Parameters of the generator, the discriminator, and the projection layer: (θ, ϕ, φ).

Output: Optimized (θ, ϕ, φ).

Initialize (θ, ϕ, φ)
for $\{1, ..., \#\text{ of training iterations}\}$ **do**
  for $\{1, ..., n_{\text{dis}}\}$ **do**
    Sample $\{(\boldsymbol{x}_i, y_i^{\text{real}})\}_{i=1}^{m} \sim p_{\text{real}}(\mathbf{x}, \mathbf{y})$
    Sample $\{\boldsymbol{z}_i\}_{i=1}^{m} \sim p(\mathbf{z})$ and $\{y_i^{\text{fake}}\}_{i=1}^{m} \sim p(\mathbf{y})$
    $\mathcal{L}_C^{\text{real}} \longleftarrow \frac{1}{m}\sum_{i=1}^{m} \ell_{2\text{C}}(\boldsymbol{x}_i, y_i^{\text{real}}; t)$ ▷ Eq. (8) with real images.
    $\mathcal{L}_D \longleftarrow \frac{1}{m}\sum_{i=1}^{m}\{D_\phi(G_\theta(\boldsymbol{z}_i, y_i^{\text{fake}}), y_i^{\text{fake}}) - D_\phi(\boldsymbol{x}_i, y_i^{\text{real}})\} + \lambda\mathcal{L}_C^{\text{real}}$
    $\phi, \varphi \longleftarrow \text{Adam}(\mathcal{L}_D, \alpha_1, \beta_1, \beta_2)$
  end for
  Sample $\{\boldsymbol{z}_i\}_{i=1}^{m} \sim p(\mathbf{z})$ and $\{y_i^{\text{fake}}\}_{i=1}^{m} \sim p(\mathbf{y})$
  $\mathcal{L}_C^{\text{fake}} \longleftarrow \frac{1}{m}\sum_{i=1}^{m} \ell_{2\text{C}}(G_\theta(\boldsymbol{z}_i, y_i^{\text{fake}}), y_i^{\text{fake}}; t)$ ▷ Eq. (8) with fake images.
  $\mathcal{L}_G \longleftarrow -\frac{1}{m}\sum_{i=1}^{m}\{D_\phi(G_\theta(\boldsymbol{z}_i, y_i^{\text{fake}}), y_i^{\text{fake}})\} + \lambda\mathcal{L}_C^{\text{fake}}$
  $\theta \longleftarrow \text{Adam}(\mathcal{L}_G, \alpha_2, \beta_1, \beta_2)$
end for

Note: 2C loss is computed using m real images in the discriminator training step and m generated images in the generator training step.

Experiments (1)

Figure 4: Examples of generated images using the proposed ContraGAN. (left) CIFAR10 [24], FID: 10.322, (right) ImageNet [18], FID: 19.443. In the case of ImageNet experiment, we select and plot well-generated images.

Metric: FID (Frechet Inception Distance) = Wasserstein-2 distance, lower the better.

Experiments (2)

Table 1: Experiments on the effectiveness of 2C loss. Considering both data-to-data and data-to-class relations largely improves image generation results based on FID values. Mean±variance of FID is reported, and lower is better.

Dataset	Uncond. GAN [6]	with P-NCA loss [33]	with NT-Xent loss [34]	with Eq.7 loss	with 2C loss (ContraGAN)
CIFAR10 [24]	15.550±1.955	15.350±0.017	14.832±0.695	10.886±0.072	**10.597±0.273**
Tiny ImageNet [25]	56.297±1.499	47.867±1.813	54.394±9.982	33.488±1.006	**32.720±1.084**

Table 2: Experiments using CIFAR10 and Tiny ImageNet datasets. Using three backbone architectures (DCGAN, ResGAN, and BigGAN), we test three approaches using different class conditioning models (ACGAN, ProjGAN, and ContraGAN (ours)).

Dataset	Backbone	Method for class information conditioning		
		ACGAN [19]	ProjGAN [17]	ContraGAN (Ours)
CIFAR10 [24]	DCGAN [2, 4]	21.439±0.581	19.524± 0.249	**18.788±0.571**
	ResGAN [26, 16]	11.588±0.093	**11.025± 0.177**	11.334±0.126
	BigGAN [6]	10.697±0.129	10.739±0.016	**10.597±0.273**
Tiny ImageNet [25]	BigGAN [6]	88.628±5.523	37.563±4.390	**32.720±1.084**

Table 3: Comparison with state-of-the-art GAN models. We mark '*' to FID values reported in the original papers [4, 5, 7]. The other FID values are obtained from our implementation. We conduct ImageNet [18] experiments with a batch size of 256.

Dataset	SNResGAN [4]	SAGAN [5]	BigGAN [6]	ContraGAN (Ours)	Improvement
CIFAR10 [24]	*17.5	17.127±0.220	*14.73/10.739±0.016	**10.597±0.273**	*+28.1%/**+1.3%**
Tiny ImageNet [25]	47.055±3.234	46.221±3.655	31.771±3.968	**29.492±1.296**	**+7.2%**
ImageNet [18]	-	-	21.072	**19.443**	**+7.7%**

Paper 2

PointGMM: a Neural GMM Network for Point Clouds (CVPR 2020)

Amir Hertz, Rana Hanocka, Raja Giryes, Daniel Cohen-Or

https://arxiv.org/abs/2003.13326

Note

I have not checked all the technical details of this paper yet, but the idea seems to be applicable to our case.

Overview

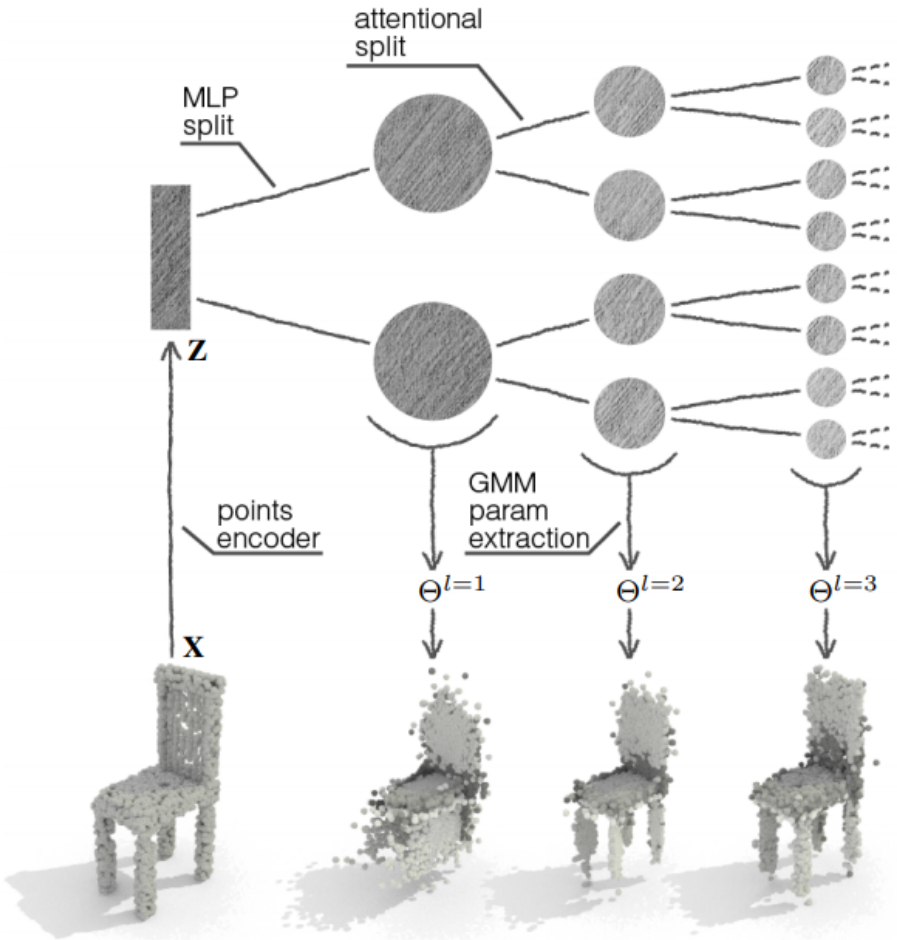


Figure 2: *Method Overview*. PointGMM learns a hierarchical GMM representation of the input X. Each depth d of the tree corresponds to a group of GMMs (with parameters Θ^d) representing the input distribution at different resolutions.

Shape generation

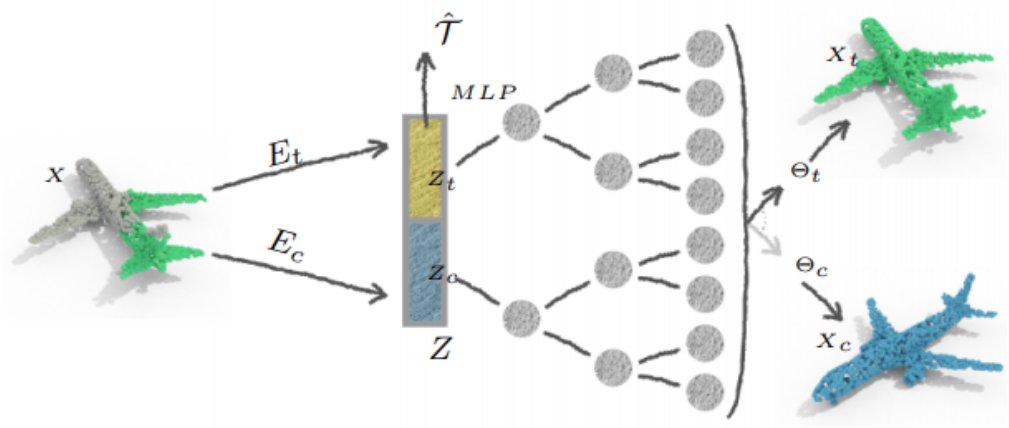


Figure 3: *Registration overview.* The input point cloud X (green) is disentangled into two different embeddings: transformation (Z_t) and the shape (Z_c); via two parallel encoders E_t and E_c.

Question

Can we apply into molecules? For example, given a part of the molecule, we have a generative model to fill out the rest such that the whole molecule is a valid one.

Experiments (1)

shape	Baseline	Point Decoder	PointGMM
chair	0.0612	0.1185	**0.0382**
car	0.1514	0.2073	**0.0447**
airplane	0.1688	0.1817	**0.0447**

Table 1: Decoder ablation. The registration results improve when adding a PointGMM decoder (compared to baseline), and excel compared to a vanilla point decoder.

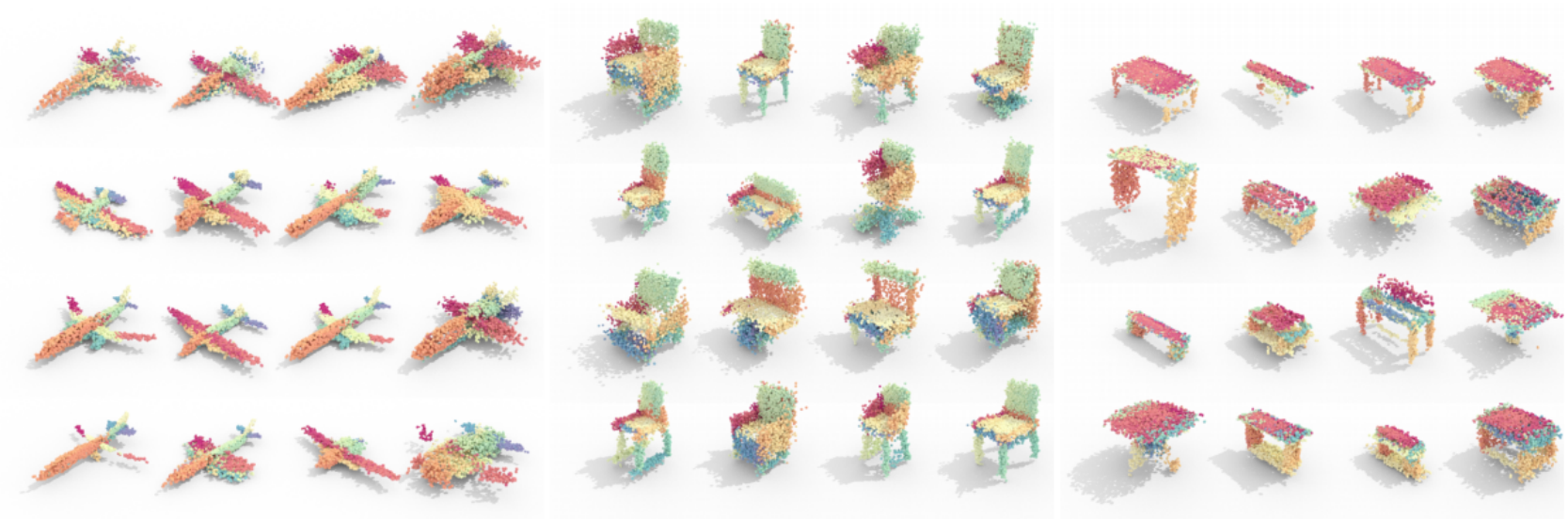


Figure 5: Randomly sampled shapes using a PointGMM generative model.

Experiments (2)

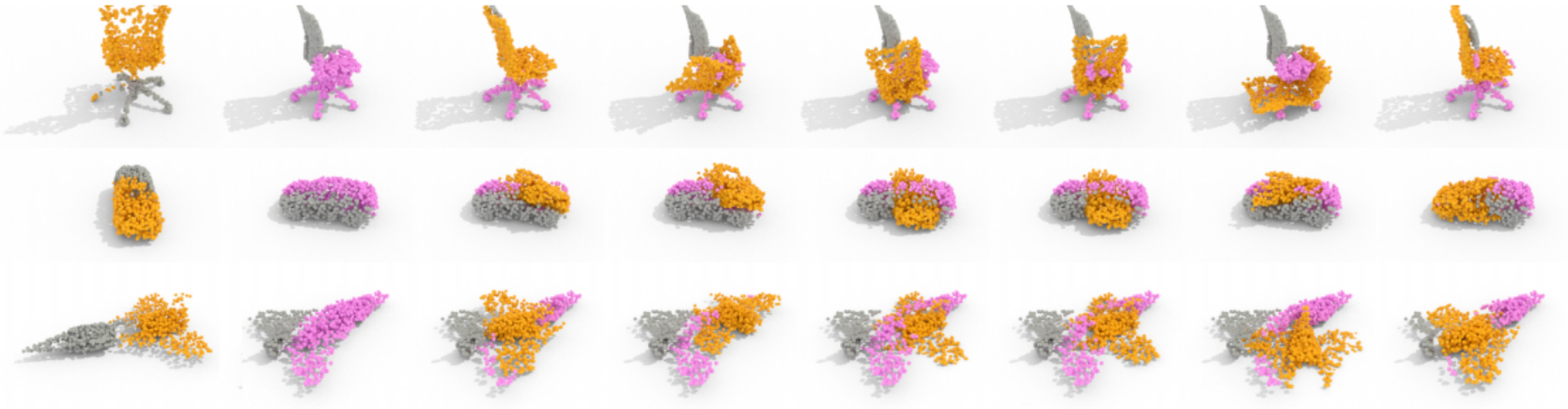


Figure 6: Qualitative results from the rigid registration comparison.

Experiments (3)

shape	max rotation (°)	sampling coverage %	RANSAC FPFH [37]	S4PCS [30]	GMMREG [22]	ICP [6]	PointNetLK [3]	Ours PointGMM
chair	30	50 - 80	0.2113	0.3343	0.0434	0.0430	0.1665	**0.0226**
chair	30	30 - 50	0.2804	0.3500	0.0842	0.0824	0.2617	**0.0496**
chair	180	50 - 80	0.2481	0.3479	0.2586	0.2578	0.2768	**0.0232**
chair	180	30 - 50	0.3132	0.3732	0.2829	0.2817	0.3386	**0.0574**
car	30	50 - 80	0.1352	0.2344	0.0399	0.04003	0.0566	**0.0246**
car	30	30 - 50	0.2134	0.2573	0.0884	0.08774	0.1647	**0.0552**
car	180	50 - 80	0.1754	0.2411	0.2134	0.2134	0.2288	**0.0290**
car	180	30 - 50	0.2357	0.2593	0.2354	0.2350	0.2548	**0.0702**
airplane	30	50 - 80	0.0765	0.1254	0.0632	0.0661	0.0798	**0.0312**
airplane	30	30 - 50	0.1501	0.1637	0.1052	0.1070	0.1301	**0.0490**
airplane	180	50 - 80	0.1485	0.1768	0.1983	0.1979	0.2023	**0.0350**
airplane	180	30 - 50	0.1961	0.2084	0.2293	0.2302	0.2308	**0.0489**

Table 2: Quantitative comparisons for rigid registration on partial shapes.