SCIENCE CHINA
Information Sciences

• Supplementary File •

Hybrid3D: learning 3D hybrid features with point clouds and multi-view images for point cloud registration

Bangbang YANG[1], Zhaoyang HUANG[2], Yijin LI[1], Han ZHOU[1],
Hongsheng LI[2], Guofeng ZHANG[1] & Hujun BAO[1*]

[1]*Zhejiang University, Hangzhou* 310058*, China;*
[2]*The Chinese University of Hong Kong, Hong Kong* 999077*, China*

In this supplementary material, we provide more implementation details of our Hybrid3D in Sec. Appendix A, and further quantitatively and qualitatively evaluate our proposed method in Sec. Appendix B.

Appendix A Details of Hybrid3D

Appendix A.1 Details of 2DFP

2DFP architecture. We design our 2D feature proposal module (2DFP) as shown in Table. A1. Two separated encoders are used to extract features from RGB images and projected 3D point features. Then, a 2D dense fusion layer is used to fuse the output feature maps of these encoders in a pixel-wise manner. After that, the output (#16) of 2D dense fusion layer will be fed into the descriptor decoder, score decoder and location decoder. Besides, we also use skip-connections from the middle output (#5, #13) of two encoders to the descriptor decoder.

Appendix A.2 Details of 3DSF

3DSF architecture. The architecture of soft feature fusion in 3DSP is constructed by a set abstraction (SA) layer with a balanced fusion strategy and a feature propagation/upsampling (FP) layer. The SA layer takes the hybrid point cloud through a $\texttt{MLP}(160, 256, 256, 256)$, where $\texttt{MLP}(160, 256, 256, 256)$ denotes the MLP with the FC output size of 160, 256, 256, 256. The input feature-length 160 is the summation of feature-length from multi-view 2D features (128) and point cloud features (32). Operation details of sampling and grouping will be introduced in the balanced fusion strategy. The FP layer takes the sub-sampled features from the set abstraction layer, and interpolates them to the same size as the input point cloud. After that, we obtain the final 3D hybrid features by post-processing the interpolated feature with an $\texttt{MLP}(416, 256, 64)$. The output of 3D hybrid features has the same number as the input point cloud, with a feature-length of 64.

Appendix A.3 Details of 3D scoring module

3D scoring module architecture. The 3D scoring module is constructed by an MLP with FC output sizes of 64 and 1, and then uses a sigmoid function to obtain the keypoint scores $\in (0, 1)$.

Appendix A.4 Implementation Details

The network of our Hybrid3D framework is implemented in PyTorch. We adopt the sparse convolution implementation from Monkowski [2–4], with voxel grid size of 0.02m. We first train the 2DFP, and then train the 3DSF and 3D scoring module with 2DFP weights fixed. The input point cloud is randomly sampled to 20k points to save computation. For better generalization, we augment the input point clouds with Gaussian noises ($\sigma = 0.05$) at the 3-axis, randomly rotate the point cloud with angle $\in [0^\circ, 360^\circ)$ around an arbitrary axis, and augment the input RGB images with color jitters. In 3DSF training, we set $w_{\mathrm{3d\_score}} = 1.0$, $w_{\mathrm{3d\_cons}} = 1.0$, $w_{\mathrm{3d\_peak}} = 0.05$ and $w_{\mathrm{3d\_desc}} = 1.0$.

We optimize the network using the Adam optimizer with an initial learning rate 1.0×10^{-4} for 2DFP, and 1.0×10^{-3} for 3DSF. The learning rates are exponentially decreased every epoch, with momentum set to 0.97 for 2DFP and 0.9 for 3DSF. We train 2DFP and 3DSF both for 100 epochs, with 300 and 600 randomly sampled fragment pairs for each epoch in 2DFP and 3DSF, respectively. The whole training process takes about two days in a single Nvidia RTX 2080Ti-11G, with batch sizes 8 and 1 for 2DFP and 3DSF, respectively.

During testing, we additionally apply a 3D non-maximum suppression with radius $r_{nms} = 0.03\mathrm{m}$, which is similar to the hard selection strategy in D3Feat [1]. Practically, we multiply the scores of the suppressed points with $\epsilon = 1.0 \times 10^{-4}$, so that we can avoid selecting keypoints too close to each other when only a small keypoint number is allowed, while keeping the score order of the suppressed points.

The design of Hybrid3D has considered the computation resources as we only select informative 2D features for 3D fusion, rather than densely combining multi-modal features. We compare the runtime (including data preparing) on a PC (Intel i7-9700K, 32G RAM, Nvidia RTX 2070) in Table A2. It is noteworthy that the OverlapPredator [6] requires pairwise point cloud input for cross-attention, which is much slower on the total processing time on the 3DMatch, as each point cloud has multiple matching candidates that need standalone network forwarding.

Method	3DSmoothNet	D3Feat	FCGF	BYOC	Predator	Our
Total Runtime per Fragment (s)	18.353	0.169	0.195	0.196	0.526	0.887

Table A2 Total runtime per fragment on 3DMatch test set.

Input	Layer Description	Output	Resolution
	Projected 3D Point Features	#1	$\times 1$
	Encoder_1		
#1	`DownSample` (Nearest)	#2	$\times 1/2$
#2	`DoubleConv` (3 × 3, 1, 32)	#3	$\times 1/2$
#3	`MaxPool` ($\times 1/2$)	#4	$\times 1/4$
#4	`DoubleConv` (3 × 3, 32, 64)	#5	$\times 1/4$
#5	`MaxPool` ($\times 1/2$)	#6	$\times 1/8$
#6	`DoubleConv` (3 × 3, 64, 128)	#7	$\times 1/8$
	Input Images	#8	$\times 1$
	Encoder_2		
#8	`DoubleConv` (3 × 3, 1, 32)	#9	$\times 1$
#9	`MaxPool` ($\times 1/2$)	#10	$\times 1/2$
#10	`DoubleConv` (3 × 3, 32, 64)	#11	$\times 1/2$
#11	`MaxPool` ($\times 1/2$)	#12	$\times 1/4$
#12	`DoubleConv` (3 × 3, 64, 128)	#13	$\times 1/4$
#13	`MaxPool` ($\times 1/2$)	#14	$\times 1/8$
#14	`DoubleConv` (3 × 3, 128, 256)	#15	$\times 1/8$
	2D Dense Fusion Layer		
#7 $\oplus$ #15	`DoubleConv` (3 × 3, 384, 256)	#16	$\times 1/8$
	Descriptor Decoder		
#16	`DoubleConv` (3 × 3, 256, 512)	#17	$\times 1/8$
#17	`PixelShuffle` ($\times 2$)	#18	$\times 1/4$
#5 $\oplus$ #13 $\oplus$ #18	`Conv2d` (3 × 3, 320, 256) + `BN` + `LReLU`	#19	$\times 1/4$
#19	`Conv2d` (3 × 3, 256, 128)		$\times 1/4$
	Score Decoder		
#16	`Conv2d` (3 × 3, 256, 256) + `BN` + `LReLU`	#20	$\times 1/8$
#20	`Conv2d` (3 × 3, 256, 1) + Sigmoid		$\times 1/8$
	Location Decoder		
#16	`Conv2d` (3 × 3, 256, 256) + `BN` + `LReLU`	#21	$\times 1/8$
#21	`Conv2d` (3 × 3, 256, 256) + Tanh		$\times 1/8$

Table A1 2DFP diagram. `DoubleConv`$(3 \times 3, C_{in}, C_{out})$ denotes the double convolution [8] with kernel size 3, input C_{in} channel and output C_{out} channels. $\oplus$ is the operation of feature map concatenation.

Appendix B Extended Experiments

Our 3DSF vs. various view-pooling methods. To our best knowledge, most previous works fuse features in multi-view images by view-pooling. One of the limitations of view-pooling is that they require a hard one-to-many association to select candidates. In contrast, our 3DSF softly fuses features from different views in the canonical space, which achieves better tolerance for erroneous associations. To demonstrate this, we compare our 3DSF with some representative view-pooling methods, including max-view pooling [5], Fuception [9] and soft-view pooling [7]. Specifically, for max-view pooling methods, we gather the neighboring 2D features in radius r for each 3D point and go through a max-pooling operation. Then, we concatenate these pooled features to the corresponding 3D features, and use an MLP to obtain the 2D-3D fused features. For Fuception, we mimic the author's design and query three nearest 2D features within radius r for each 3D point, and implement Fuception module to output the fused 2D features. The 2D-3D fused feature is also obtained by concatenation and MLP linear projection. For soft-view pooling, we query neighboring features (up to 32) within radius r for each 3D point, and use an MLP followed by a SoftMax function to obtain the attentive weights for each neighboring feature. The fused 2D features are the weighted sum of the neighboring features. Similarly, we concatenate these 2D features to the 3D features and utilize an MLP to obtain the fused features. As shown in Table B1, our Hybrid3D achieves higher registration recall than the above view-pooling methods, which shows the effectiveness of 3DFP when dealing with 2D-3D feature fusion in multi-view cases.

Our scoring strategy vs. D3Feat scoring strategy. Besides our distinctiveness scores, D3Feat [1] also provides a scoring strategy. In contrast with D3Feat which uses raw 3D points, our Hybrid3D exploits texture cues from images and inspects the distinctiveness in multiple views. We compare our scores with D3Feat to present the effectiveness. First, we replace our 3D score distinctiveness loss with score loss from D3Feat, which is denoted as *D3Feat Score Loss*. Then, we use the whole D3Feat scoring strategy, including keypoint selection strategy and the score loss, and denote it as *D3Feat Score Strategy*. Our scores achieve similar registration performance with D3Feat when using 500,0 keypoints because the large number of keypoints suppresses the impact of scores for fragment registration. As shown in Table B1 and Fig. B1, our scores significantly surpass D3Feat score loss and D3Feat score strategy with fewer keypoints, which indicates the superiority of our method.

Robustness against pose noise. To show the robustness of our method against pose noise, we evaluate by manually adding noise to the input camera poses before the soft fusion process. As shown in Table B2, our method still achieves high feature matching recall and registration recall even when adding 10° rotation errors and 10cm position errors, with only the inlier ratio slightly affected. Meanwhile, we do observe pose error/inconsistency on the 3DMatch dataset (even cm level for some sequences), which may introduce errors both in the point clouds and 2D-3D association. Despite that, we still outperform other methods on the evaluated metrics.

* Corresponding author (email: bao@cad.zju.edu.cn)

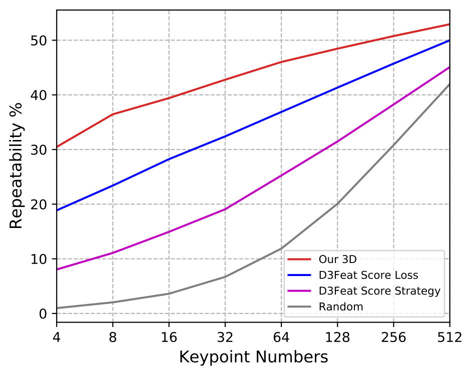

Figure B1 Relative repeatability comparison between our scoring strategy and D3Feat scoring strategy on 3DMatch.

# Keypoints	**50**	**100**	**250**	**500**	**1000**	**2500**	**5000**
			Registration Recall (%)				
Our 3D	**60.9**	**74.3**	**85.0**	**88.3**	**87.3**	**87.5**	**87.4**
Max-View Pooling (0.05)	35.0	50.9	71.8	81.6	86.2	85.1	87.1
Max-View Pooling (0.075)	35.2	52.0	72.4	82.9	85.8	85.1	86.1
Max-View Pooling (0.1)	33.4	50.8	72.0	82.0	85.4	85.3	87.0
Soft-View Pooling (0.005)	37.4	53.0	72.5	80.7	84.8	84.1	86.5
Soft-View Pooling (0.075)	33.0	50.8	70.3	81.6	86.6	83.8	85.8
Soft-View Pooling (0.1)	33.2	50.3	70.4	81.8	85.7	84.7	85.4
Fuception (0.005)	38.0	52.6	72.9	80.5	85.4	84.4	86.2
Fuception (0.075)	34.4	52.0	72.8	82.3	85.3	85.0	86.7
Fuception (0.1)	32.0	51.7	72.8	83.7	86.3	84.1	86.4
D3Feat Score Loss	28.0	48.0	68.0	78.5	83.5	84.8	84.5
D3Feat Score Strategy	21.7	41.3	68.6	79.4	84.6	86.9	**87.4**

Table B1 Point cloud registration evaluation on 3DMatch. Our Hybrid3D with 3DSF is denoted as *Our 3D*, and the number in the bracket is the radius of r ball query in neighbors search. For example, *Max-View-Pooling (0.075)* means the max-view pooling with ball query radius $r = 0.075\text{m}$ in neighbors search.

	Origin	R(5°)	R(10°)	P(5cm)	P(10cm)	R(5°)+P(5cm)	R(10°)+P(10cm)
FMR (%)	97.7	97.1	96.8	97.7	97.6	97.1	96.6
Reg Recall (%)	87.4	87.8	87.2	87.1	87.7	87.5	87.4
Inlier Ratio (%)	56.1	53.4	52.0	55.1	54.5	53.2	51.9

Table B2 Analysis of pose error on 3DMatch dataset, where R and P denote the error added on rotation and position.

References

1 Xuyang Bai, Zixin Luo, Lei Zhou, Hongbo Fu, Long Quan, and Chiew-Lan Tai. D3feat: Joint learning of dense detection and description of 3d local features. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 6358–6366. IEEE, 2020.

2 Thomas Chaton, Chaulet Nicolas, Sofiane Horache, and Loic Landrieu. Torch-points3d: A modular multi-task frameworkfor reproducible deep learning on 3d point clouds. In *2020 International Conference on 3D Vision (3DV)*. IEEE, 2020.

3 Christopher Choy, Jaesik Park, and Vladlen Koltun. Fully convolutional geometric features. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 8957–8965. IEEE, 2019.

4 Christopher B. Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 3075–3084. Computer Vision Foundation / IEEE, 2019.

5 Haibin Huang, Evangelos Kalogerakis, Siddhartha Chaudhuri, Duygu Ceylan, Vladimir G Kim, and Ersin Yumer. Learning local shape descriptors from part correspondences with multiview convolutional networks. *ACM Transactions on Graphics (TOG)*, 37(1):1–14, 2017.

6 Shengyu Huang, Zan Gojcic, Mikhail Usvyatsov, Andreas Wieser, and Konrad Schindler. Predator: Registration of 3d point clouds with low overlap. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4267–4276, 2021.

7 Lei Li, Siyu Zhu, Hongbo Fu, Ping Tan, and Chiew-Lan Tai. End-to-end learning local multi-view descriptors for 3d point clouds. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 1916–1925. IEEE, 2020.

8 Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells III, and Alejandro F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention - MICCAI 2015 - 18th International Conference Munich, Germany, October 5 - 9, 2015, Proceedings, Part III*, volume 9351 of *Lecture Notes in Computer Science*, pages 234–241. Springer, 2015.

9 Lei Zhou, Siyu Zhu, Zixin Luo, Tianwei Shen, Runze Zhang, Mingmin Zhen, Tian Fang, and Long Quan. Learning and matching multi-view descriptors for registration of point clouds. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XV*, volume 11219 of *Lecture Notes in Computer Science*, pages 527–544. Springer, 2018.