AutoSynth: Learning to Generate 3D Training Data for Object Point Cloud Registration

Zheng Dang[1] and Mathieu Salzmann[1,2]
[1]CVLab, EPFL, Switzerland [2]ClearSpace, Switzerland
{zheng.dang, mathieu.salzmann}@epfl.ch

Abstract

In the current deep learning paradigm, the amount and quality of training data are as critical as the network architecture and its training details. However, collecting, processing, and annotating real data at scale is difficult, expensive, and time-consuming, particularly for tasks such as 3D object registration. While synthetic datasets can be created, they require expertise to design and include a limited number of categories. In this paper, we introduce a new approach called AutoSynth, which automatically generates 3D training data for point cloud registration. Specifically, AutoSynth automatically curates an optimal dataset by exploring a search space encompassing millions of potential datasets with diverse 3D shapes at a low cost. To achieve this, we generate synthetic 3D datasets by assembling shape primitives, and develop a meta-learning strategy to search for the best training data for 3D registration on real point clouds. For this search to remain tractable, we replace the point cloud registration network with a much smaller surrogate network, leading to a 4056.43 *times speedup. We demonstrate the generality of our approach by implementing it with two different point cloud registration networks, BPNet [13] and IDAM [34]. Our results on TUD-L [26], LINEMOD [23] and Occluded-LINEMOD [7] evidence that a neural network trained on our searched dataset yields consistently better performance than the same one trained on the widely used ModelNet40 dataset [65].*

1. Introduction

3D point cloud registration, which aims to estimate the relative transformation between two given point clouds, is a traditional computer vision task. With the advent of deep learning, point cloud registration is nowadays commonly tackled with deep networks, achieving impressive results. The main research direction in this area consists of designing new network architectures to improve performance. Here, by contrast, we argue that the quantity and quality of training data have as crucial an impact on the networks' performance as its architecture and training details, and thus advocate data creation as a research goal in itself.

The traditional approach to collecting 3D registration data consists of scanning real objects. This, however, is highly time-consuming and does not scale to the quantity of data commonly expected for deep network training. Generating synthetic data, therefore, comes as a promising alternative. Nevertheless, it requires access to 3D object models, thus often limiting the number of categories, and human expertise to generate realistic data, typically leading to a domain gap w.r.t. real-world point clouds despite best efforts.

In this work, we address this by introducing an approach dubbed AutoSynth, automating the process of curating a 3D dataset. Specifically, we aim for the resulting dataset to act as effective training data for a 3D object registration network that will then be deployed on real-world point clouds. To achieve this, we develop a meta-learning strategy that searches for the optimal dataset over a space encompassing millions of potential datasets, covering a wide diversity of 3D shapes. The search is guided by a target real-world dataset, thus producing data that reduces the domain gap. Our experiments demonstrate that the resulting training dataset yields improved registration performance not only on the target data but on other real-world point clouds.

For this to be possible, we design a very large search space based on the assumption that complex shapes can be created by combining simple primitives. Diverse datasets can then be sampled from this space, and we design an evolutionary algorithm to automatically curate the best training dataset to achieve high performance on the target data. Employing a registration network in the search process, however, would be impractical as even the smallest competitive model would require $1,875$ GPU days on a single RTX8000 for only $1,000$ search steps. To make the search tractable, we observe that the true quality function, i.e., the accuracy of the registration network of interest, can be replaced with a proxy one, i.e., the reconstruction accuracy of an autoencoder. Specifically, our experiments evidence that, for the same training and testing data, registration accuracy and reconstruction quality follow the same trend, even when using

an autoencoder whose architecture is orders of magnitude smaller than that of any registration network able to produce nontrivial results. As such, our approach yields a $4056.43\times$ speedup compared to using a registration network.

We demonstrate the generality of our approach by implementing it with two different point cloud registration networks, BPNet [13] and IDAM [34]. Our results on TUD-L [26], LINEMOD [23] and Occluded-LINEMOD [7] consistently demonstrate that a neural network trained on our searched dataset achieves better performance than the same one trained on the widely used ModelNet40 dataset [65].

Our main contributions can be summarized as follows:

- We present AutoSynth, a novel meta-learning-based approach to automatically generate large amounts of 3D training data and curate an optimal dataset for point cloud registration.
- We show that the search can be made tractable by leveraging a surrogate network that is 4056.43 times more efficient than the point cloud registration one.
- We evidence that using a single scanned real-object as target dataset during the search yields a training set that leads to good generalization ability.

2. Related Work

Traditional point cloud registration methods. Point cloud registration aims to estimate the relative pose between two input point sets. Many algorithms [2, 41, 46, 42, 17, 40, 50, 27, 29, 54, 53, 70, 33, 1, 24, 22, 18, 9, 8, 21, 35] have contributed to achieving this. The best-known one probably is Iterative Closest Point (ICP) [5], which has served as basis for many variants, such as Generalized-ICP [55] and Sparse ICP [6], aiming to improve robustness to noise and mismatches. We refer the reader to [44, 52] for a review of ICP-based strategies. The main drawback of ICP-based methods is their requirement for a reasonable initialization to converge to a good solution. As a consequence, recent efforts have been made towards global optimization strategies, leading to algorithms such as Go-ICP [72], Super4PCS [41], and Fast Global Registration (FGR) [77]. While effective, these methods still suffer from the presence of noise and outliers in the point sets. This is addressed by post-processing strategies, such as that of [60], TEASER [70], and TEASER++ [71].

Learning-based object point cloud registration. Following the current trend in computer vision, much recent point cloud registration research has focused on a deep learning-based approach. A key requirement to achieve this was the design of deep networks acting on unstructured sets. Deep sets [75] and PointNet [45] constitute pioneering works in this direction. In particular, PointNetLK [3] combines the PointNet backbone with the traditional, iterative Lucas-Kanade (LK) algorithm [39] so as to form an end-to-end registration network; DCP [62] exploits DGCNN [64] backbones followed by Transformers [59] to establish 3D-3D correspondences. While effective, PointNetLK and DCP cannot tackle the partial-to-partial registration scenario. That is, they assume that both point sets are fully observed, during both training and test time. PRNet [63] and IDAM [34] address this via a deep network designed to extract keypoints from each input set and match these keypoints. By contrast, RPM-Net [73] and RGMNet [19] build on DCP and adopt a different strategy, replacing the softmax layer with an optimal transport one so as to handle outliers. DeepGMR [74] leverages mixtures of Gaussians and formulates registration as the minimization of the KL-divergence between two probability distributions to handle outliers. While the above-mentioned methods were designed to handle point clouds in full 3D, the recent BPNet [13] was shown to successfully tackle registration from 2.5D measurements, including on real scene datasets, such as TUD-L [26], LM [23] and LMO [7]. Here, we follow an orthogonal direction to these works, and address the task of learning to generate synthetic training data to generalize to real scene test-time observations. We will demonstrate our approach using both BPNet [13] and IDAM [34].

Learning to generate training data. Data is essential for the success of learning-based methods, including point cloud registration ones. While much effort has been made to obtain real-world 3D ground truth [26, 23, 7, 25, 16, 31, 67, 48, 15, 58, 20], synthetic data generation [65, 66, 37, 61, 11] has emerged as an effective alternative source for supervision. For such synthetic datasets, e.g., ModelNet40 [65], the creation of each mesh model nonetheless requires human supervision to control its size, position, texture, etc. Hence producing a large amount of synthetic objects remains laborious. This raises the question of the feasibility to automatically generate the training data.

In this context, most existing works focus on synthesizing images. For example, the work of [51], Meta-Sim [30], Meta-Sim2 [14], and AutoSimulate [4] learn simulator hyperparameters to maximize the performance of a model on semantic segmentation or object detection. This is achieved by treating the entire data generation and network training pipeline as a black-box, and using reinforcement learning-based gradient estimation. However, these methods still require manually-designed object and scene models as input to the simulator, thus limiting the generated data to a small number of scenes. By contrast, AutoFlow [57] leverages web images to learn to generate image pairs, thus greatly increasing the data diversity. This, however, does not easily generalize to generating point cloud data. A few works have nonetheless tackled the problem of generating 3D data using shape primitives. In particular, [68] does so to build training data for a shape-from-shading network that recon-

structs object shapes from image sequences; [69] generates 3D synthetic training data to estimate the surface normals, depth, albedo, and shading maps from a single RGB image.

Importantly, these techniques rely on the main task network to evaluate the effectiveness of the training data. With the typical growth of state-of-the-art deep network for point cloud registration, this would result in an intractable computational cost. Here, we therefore propose to replace the main task network with a lightweight surrogate network in the searching phase, which we demonstrate to maintain the final performance while requiring three orders of magnitude less computation. Note that our approach does not follow the predictor-based strategy commonly used in neural architecture search [36, 43, 12]. Specifically, these methods still require training a thousand target models to then train the predictor, which remains too expensive for the computationally-intensive state-of-the-art point cloud registration networks. Here, instead, we leverage a surrogate network that completely replaces the original one.

3. Methodology

3.1. Problem Formulation

Our objective is to automatically generate a synthetic 3D dataset D_{syn} such that the main task model (MTM), i.e., a point cloud registration model Ψ in our case, achieves maximum accuracy on the test set when trained on D_{syn} until convergence. The test set is evidently not available during training, and thus we mimic it with a target dataset D_{tgt}.

Formally, we express the problem of searching for a synthetic dataset D_{syn} as that of finding a policy P, encompassing hyperparameters to generate a 3D dataset, such that $\Psi(w, D_{syn}(P))$ achieves the best performance on D_{tgt}. The set of all policies is referred to as the search space O, and we use an evolutionary algorithm to find the best policy $\hat{P}$ that minimizes the evaluation loss

$$\hat{P} = \underset{P \in O}{\operatorname{argmin}}\ \mathcal{L}_{eval}(\Psi(w, D_{syn}(P)), D_{tgt}), \qquad (1)$$

where w denotes the weights of the MTM trained on D_{syn} until convergence.

3.2. Search Space

The search space defines the set of policies that the meta-learning method can explore during training. In other words, it encompasses all possible training datasets, with each policy corresponding to the hyperparameters used to create one 3D dataset. To generate a dataset, we exploit the observation that complex shapes can be obtained by combining simple primitives [10, 68, 56], such as cuboids, cones, cylinders, etc. Following [49, 68], we define each shape primitive as an implicit surface function $\mathcal{F} : \mathbb{R}^3 \rightarrow \mathbb{R}$, such that a point $\mathbf{x} \in \mathbb{R}^3$ on the primitive's surface satisfies $\mathcal{F}(\mathbf{x}) = 0$, whereas $\mathcal{F}(\cdot) < 0$ for interior points and $\mathcal{F}(\cdot) > 0$ for exterior ones. In other words, $\mathcal{F}$ encodes a signed distance function.

Each primitive can then undergo a set of transformations. Specifically, we focus on affine transformations, such as translation, rotation, scaling, shearing, and stretching. For a 3D point $\mathbf{x}$, this can be expressed as

$$\mathcal{T}(\mathbf{x}) = \alpha T_{rot} T_{shear} T_{stretch} \mathbf{x} - \mathbf{t}, \qquad (2)$$

where α is a scaling parameter controlling the overall size of the primitive, $\mathbf{t}$ is a translation vector, T_{rot} is a rotation matrix, $T_{shear} = S_x S_y S_z$ is a matrix combining shearing operations along the different axes, and $T_{stretch} = A_x A_y A_z$ is a matrix controlling the scale of the primitive along the different axes. Given an existing shape primitive $\mathcal{F}(\mathbf{x})$, the transformed shape can be obtained as $\mathcal{F}(\mathcal{T}(\mathbf{x}))$.

To composite the individual transformed primitives into a complex shape, we utilize logic operators between shapes, as discussed below. Specifically, to create more distinct shapes from our primitives, we perform truncation with a plane. Let $\mathcal{F}(\mathbf{x})$ denote a transformed primitive, where we neglect the explicit dependency on the transformation $\mathcal{T}(\cdot)$ for ease of notation. Furthermore, let $\mathcal{F}_{plane}(\mathbf{x})$ denote a plane, defined by a point and a surface normal. The truncation operation can then be expressed as

$$\mathcal{F}_{truncation}(\mathbf{x}) = max(\mathcal{F}(\mathbf{x}),\ \mathcal{F}_{plane}(\mathbf{x}))\ . \qquad (3)$$

Given a set of m transformed and truncated primitives with implicit representations $\{\mathcal{F}_1(\mathbf{x}), \mathcal{F}_2(\mathbf{x}), \ldots, \mathcal{F}_m(\mathbf{x})\}$, we combine the shapes using the union operator, which can be expressed as

$$\mathcal{F}_{union}(\mathbf{x}) = \{\mathcal{F}_1(\mathbf{x}), \mathcal{F}_2(\mathbf{x}), \ldots, \mathcal{F}_m(\mathbf{x})\}\ . \qquad (4)$$

The final object mesh is generated by merging all vertices and faces of each transformed shape primitive. This operation is substantially faster than mesh union [28], $\mathcal{F}_{union}(\mathbf{x}) = min(\mathcal{F}_1(\mathbf{x}), \mathcal{F}_2(\mathbf{x}), \ldots, \mathcal{F}_m(\mathbf{x}))$, in practice. The mesh of the primitives can be obtained via the marching cube [38] or simply-defined vertices and faces, and shape generation can be sped up by saving and reusing them.

In this framework, a policy P consists of the 11 parameters corresponding to the above-mentioned 11 operations. Specifically, the operations we search over are rotation $\{1\}$, translation $\{1\}$, overall scale $\{1\}$, shearing on each axes $\{3\}$, and stretching on each axes $\{3\}$, with additional parameters encoding the number of primitives to consider $\{1\}$ and the truncation plane $\{1\}$. Each operation also comes with a default range of magnitude. We discretize the range of magnitude into nine values so that we can use a discrete search algorithm to find them. Ultimately, finding the optimal policy P becomes a search problem in a space that contains $9^{11} = 31,381,059,609$ possibilities. We refer to this search space as O.

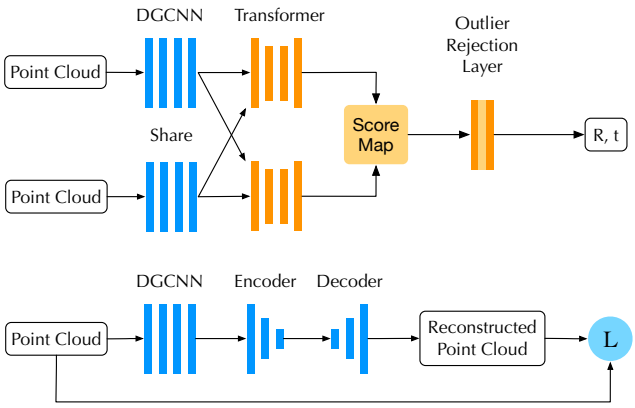

Figure 1: Comparison of the point cloud registration network (top) and the point cloud reconstruction one (bottom)

Note that the operations described above allow us to form a large search space, which we will show to be effective in practice. However, they are by no means the unique way of defining such a space, and we hope that our work will motivate others to design new search spaces.

3.3. Evolutionary Algorithm

To automatically search for the optimal policy P in the search space so as to minimize $\mathcal{L}$ in Eq. (1), we employ an evolutionary algorithm with a tournament selection strategy [47]. This algorithm acts as a meta-learner, which iteratively provides policies from which we generate the dataset $D_{syn}(P)$. The deep network is then trained on the generated dataset $D_{syn}(P)$ and evaluated on D_{tgt} to obtain feedback on its effectiveness. The meta-learner then generates a new policy based on this feedback, which causes the dataset to evolve due to policy changes. This approach allows D_{tgt} to affect the final policy, and if it consists of scanned real-objects, it can help to narrow the domain gap.

Specifically, the evolutionary algorithm starts with an initial population of k policies: $Q = \{P_1, P_2, \ldots, P_k | P_i \in O\}$. During each evolutionary step, two individuals $\{P_i, P_j\}$ are chosen from the population Q and their evaluation loss $\{\mathcal{L}(P_i), \mathcal{L}(P_j)\}$ are compared, where $\mathcal{L}(P_i) = \mathcal{L}_{eval}(\Psi(w, D_{syn}(P_i)), D_{tgt})$. After each competition, we select the best policy as parent and generate a new policy, P_{child}, through mutation. By adding the new P_{child} to the policy pool and removing the worst-performing policy, we ensure that the policy pool remains of the same size and does not shrink. Specifically, the mutation is performed by randomly choosing one out of the 11 policy hyperparameters from the duplicated best policy and changing this hyperparameter label to another discrete label. For example, for rotation, assuming a discretization in steps of $\frac{\pi}{8}$, the mutation may change the original label $\frac{\pi}{8}$ to $\frac{3\pi}{8}$. We then create a new synthetic dataset $D_{syn}(P_{child})$ from the mutated child, and train the network Ψ on it until convergence. At the next evolutionary step, the child has then the possibility of becoming a parent. This process is repeated to evolve policies until a maximum number of trials is reached. We then select the policy with the best evaluation result as the optimal policy $\hat{P}$. The details are given in Algorithm 1.

Algorithm 1: Evolutionary Policy Search

Input : Search space O, population size k, max number of trials M, target dataset D_{tgt}, deep network Ψ.
Output: Policy $\hat{P}$ to generate the data that achieves the highest validation performance

initialize_population(Q) by randomly sampling k policies from O.
current_trial_num := 0
while *current_trial_num* $< M$ **do**
  1. randomly select two individuals P_i and P_j from Q.
  2. train the network Ψ on these two datasets $D_{syn}(P_i)$ and $D_{syn}(P_j)$ until convergence.
  3. compare the evaluation loss on the target dataset D_{tgt}, and get the best_individual and the worst_individual (tournament selection)
  4. delete the worst_individual from Q;
  5. mutate the best_individual, add it to the population Q, and train the individual;
  6. current_trial_num += 1 ;
end

3.4. Surrogate Task Model

The search algorithm described in Sec. 3.3 requires training a target task model to convergence at every evolutionary trial. Unfortunately, the state-of-the-art point cloud registration networks tend to involve many parameters and expensive layers, such as transformers, as illustrated in the top portion of Fig. 1 for BPNet. As such, searching for the best training data with our search procedure would become prohibitively expensive. For example, using BPNet, one of the smallest registration models, a single trial in the search process would cost 1.875 GPU days on one Nvidia-V100. Therefore, a standard search process of $1,000$ trials would require $1,875$ GPU days.

To address this, we propose to replace the target task model with a model tackling a surrogate task. For this substitution to make sense, the surrogate model should meet the following conditions: (i) It should take as input the same type of data as the target model, i.e., point clouds; (ii)it should not require any extra annotations; (iii) it should be trainable much more quickly than the task model; (iv) its behavior, i.e., evaluation loss, should follow a similar trend to that of the target model as the training data changes. These constraints immediately discard any point registration net-

work, even much reduced versions of existing ones, as we have observed that meaningful registration results can only be obtained with architectures that would be too large for our purpose. Instead, we propose to make use of a point cloud reconstruction network. This choice was motivated by the observation that, by definition, such a network also operates on point clouds; it does not require any annotations, only the point clouds themselves; it can exploit a much more lightweight architecture, as it does not need to compare two point clouds and thus can be designed without transformer layers.

This leaves the question of evaluation loss behavior. This can be answered from the perspective of multi-task learning literature [76], which has demonstrated that different tasks performed on the same input data often follow similar behavior, i.e., improving one also improves the others. More pragmatically, we will show in our experiments that the behavior of our point cloud reconstruction network follows that of the registration one as we vary the training set.

Surrogate network architecture. The architecture of our surrogate network is shown in the bottom portion of Fig. 1. In essence, it is an autoencoder, relying on the same DGCNN block as the registration network but without any transformer layers. Instead, to prevent the network from directly copying the input point cloud to the output, we project the outputs of the DGCNN to a low-dimensional latent space, and then force the network to reconstruct the whole point cloud from this compressed representation.

Formally, the input to the network is a point set $\mathcal{X} = \{x_1, \ldots, x_v\}$, where $x_i \in \mathbb{R}^3$ represents a 3D point position. We obtain the point set $\mathcal{X}$ by uniform sampling from the mesh model. The output $\mathcal{Y}$ is the same size point set, representing the reconstructed point positions. The encoder projects each input point cloud into a latent space and the decoder reconstructs the point cloud from the latent representation. We then compute the reconstruction error for $\mathcal{Y}$ using the symmetric Chamfer distance

$$\mathcal{L}_{CD} = \frac{1}{2m}\left(\sum_{x\in\mathcal{X}} \min_{y\in\mathcal{Y}} \|x-y\|_2^2 + \sum_{y\in\mathcal{Y}} \min_{x\in\mathcal{X}} \|y-x\|_2^2\right) . \quad (5)$$

We train the surrogate network parameter θ by solving

$$\theta^* = \underset{\theta}{\operatorname{argmin}}\, \mathcal{L}_{CD}(\Psi_{surrogate}(\theta, D_{syn})). \quad (6)$$

In the searching phase, we therefore also use the symmetric Chamfer distance as fitness score, giving the loss

$$\mathcal{L}(P) = \mathcal{L}_{CD}(\Psi_{surrogate}(\theta^*, D_{syn}(P)), \hat{D}_{tgt}) . \quad (7)$$

Our surrogate task model only needs 15min to converge and only requires 1.42GB GPU memory. An experiment with $1,000$ trials only takes 0.462 GPU days on Nvidia-V100 GPU, which is 4056.43 times more efficient than using the original registration network.

4. Experiments

In this section, we evaluate the effectiveness of our AutoSynth training set search strategy. Below, we first provide implementation details. We then present results on real scenes, and finally analyze different aspects of our approach via ablation studies.

Implementation details. Our complete pipeline consists of two steps: Searching for the best policy using AutoSynth, and training the registration network on the training dataset generated using the best policy.

To generate complex 3D datasets, we utilize a set of shape primitives that includes sphere, cuboid, cone, cylinder, torus, tetrahedron, octahedron, icosahedron, and dodecahedron, as these have shown promising results in our analysis. This set of primitives, however, is not exhaustive and we hope that our results will encourage other researchers to further expand it and propose better alternatives.

In the search process, we build our target dataset D_{tgt} using one scanned real object, i.e., Stanford bunny. We augment it with random rotations to generate 100 samples, which constitute D_{tgt}. We set the population size to be 32 and the maximum number of trials to be $1,000$, which we observed to be sufficient to obtain a good policy. For the reconstruction network, we set the batch size to be 8 and use the Adam [32] with a learning rate of 0.001. For each trial in the search phase, we train the reconstruction network for $20,000$ iterations, after which the network has typically converged. Once the best policy is found, we use it for all the experiments, i.e., we only searched for the policy once.

For BPNet [13] and IDAM [34], we use the modified versions of [13] with Match Normalization. We only replace the training data but keep the same parameter settings as in [13] to train them to convergence. For the real-scene datasets, we use the provided training sequence. For ModelNet40 [65], we use the official training split, which consists of $9,843$ mesh models across 40 categories. To obtain a source point cloud, we sample points uniformly from a mesh model. For the target point cloud, we generate a depth map from the mesh and with a random camera pose, and sample points from it. For our AutoSynth search process, we only need the source point cloud as input, which also acts as ground truth for the reconstruction network.

Following [13], we report the rotation and translation mAP, the ADD, and the BOP benchmark metrics.

4.1. Results on Real-scene Datasets

Here, we compare our AutoSynth searched dataset to ModelNet40 by evaluating the performance of the registration models, i.e., BPNet and IDAM, trained on them. To this end, we evaluate the trained models on three different real-scene datasets, i.e, TUD-L, LM, and LMO. Note that this corresponds to an unseen-object setting, as the training mesh models do not overlap with the test ones.

Method	Rotation mAP			Translation mAP			ADD	BOP Benchmark			
	5°	10°	20°	$1cm$	$2cm$	$5cm$	$0.1d$	VSD	MSSD	MSPD	AR
IDAM-Real	0.56	0.58	0.61	0.55	0.66	0.81	0.58	0.580	0.604	0.618	0.601
BPNet-Real	**0.91**	**0.92**	**0.93**	**0.86**	**0.95**	**0.99**	**0.93**	**0.859**	**0.914**	**0.935**	**0.903**
ICP	0.02	0.02	0.02	0.01	0.14	0.57	0.02	0.117	0.023	0.027	0.056
FGR(FPFH)	0.00	0.01	0.01	0.04	0.25	0.63	0.01	0.071	0.007	0.008	0.029
TEASER++(FPFH)	0.13	0.17	0.19	0.03	0.22	0.56	0.17	0.175	0.196	0.193	0.188
Super4PCS	0.30	0.50	0.56	0.05	0.40	0.92	0.54	0.265	0.500	0.488	0.418
⋆Vidal-Sensors18	-	-	-	-	-	-	-	**0.811**	**0.910**	**0.907**	**0.876**
⋆Drost	-	-	-	-	-	-	-	0.809	0.875	0.872	0.852
IDAM-MN40	0.30	0.32	0.36	0.31	0.41	0.73	0.34	0.373	0.362	0.364	0.366
IDAM-AutoSynth	0.40	0.43	0.46	0.41	0.54	0.83	0.45	0.496	0.454	0.471	0.474
BPNet-MN40	0.71	0.74	0.77	0.70	0.80	0.94	0.76	0.724	0.772	0.796	0.763
BPNet-AutoSynth	**0.78**	**0.81**	**0.85**	**0.77**	**0.86**	**0.95**	**0.84**	0.777	0.845	0.867	0.829

Table 1: Quantitative comparison of registration models trained on AutoSynth and ModelNet40 on the **TUD-L** real scene dataset. Note that BPNet-Real and IDAM-Real were trained with the TUD-L real scene training sequence, i.e., not in the unseen-object setting. BPNet-MN40 was trained on ModelNet40-full. BPNet-AutoSynth was trained on our AutoSynth generated dataset with the Stanford bunny as target dataset. The results for Vidal-Sensor18 [60] and Drost (Drost-CVPR10-3D-Edges) [17] were directly taken from the BOP leaderboard.

Method	Rotation mAP			Translation mAP			ADD	BOP Benchmark			
	5°	10°	20°	$1cm$	$2cm$	$5cm$	$0.1d$	VSD	MSSD	MSPD	AR
IDAM-Real	0.15	0.23	0.27	0.25	0.54	0.91	0.23	0.352	0.311	0.345	0.336
BPNet-Real	**0.43**	**0.59**	**0.67**	**0.49**	**0.83**	**0.97**	**0.60**	0.616	0.680	0.737	0.678
ICP	0.00	0.01	0.01	0.04	0.27	0.82	0.01	0.092	0.014	0.027	0.044
FGR(FPFH)	0.00	0.00	0.00	0.05	0.31	0.89	0.00	0.068	0.000	0.010	0.026
TEASER++(FPFH)	0.01	0.03	0.05	0.03	0.21	0.73	0.03	0.108	0.076	0.098	0.094
Super4PCS	0.02	0.09	0.15	0.04	0.31	0.89	0.10	0.117	0.178	0.201	0.165
⋆PPF_3D_ICP	-	-	-	-	-	-	-	**0.719**	**0.856**	**0.866**	**0.814**
⋆Drost	-	-	-	-	-	-	-	0.678	0.786	0.789	0.751
IDAM-MN40	0.08	0.11	0.14	0.15	0.44	0.89	0.12	0.258	0.178	0.206	0.214
IDAM-AutoSynth	0.21	0.29	0.33	0.28	0.60	0.91	0.29	0.420	0.359	0.398	0.392
BPNet-MN40	0.31	0.42	0.50	0.37	0.69	0.95	0.43	0.491	0.518	0.571	0.527
BPNet-AutoSynth	**0.36**	**0.49**	**0.58**	**0.41**	**0.74**	**0.94**	**0.50**	0.538	0.579	0.641	0.586

Table 2: Quantitative comparison of registration models trained on AutoSynth and ModelNet40 on the **LINEMOD** real scene dataset. PPF_3D_ICP [17] and Drost (Drost-CVPR10-3D-Only) [17] are traditional methods and represent the best depth-only performers from the BOP leaderboard.

TUD-L dataset. The results of all methods on TUD-L are summarized in Tab. 1. In Tab. 1, the '-Real' model was trained and tested on TUD-L's real scene data, corresponds to the 'seen' object setting. On the other hand, the '-AutoSynth' model, trained on synthetic data and tested on TUD-L's real scene data, represents an 'unseen' object setting. This discrepancy in settings accounts for the observed performance difference. The same principle applies to Tabs. 2 and 3, which were tested using the LM and LMO datasets. Furthermore, we also report the results of the top-performing traditional, learning-free, registration methods.

BPNet and IDAM trained on our AutoSynth searched dataset yield significantly better performance than their counterparts trained on ModelNet40. This evidences the superiority of our searched dataset, containing more diverse and complex objects.

Note that the traditional methods based on FPFH features yield poor results. However, 'VidalSensors18' and 'CVPR10-3D-Edges', two traditional methods corresponding to the top depth-only performers in the BOP leaderboard, remain more effective than any learning-based method, including ours, in the unseen-object setting. Nevertheless, we push the limits of what synthetic data can achieve for deep learning-based methods, thus opening the

Method	Rotation mAP			Translation mAP			ADD	BOP Benchmark			
	$5°$	$10°$	$20°$	$1cm$	$2cm$	$5cm$	$0.1d$	VSD	MSSD	MSPD	AR
IDAM-Real	0.15	0.22	0.32	0.23	0.58	0.88	0.25	0.349	0.320	0.374	0.348
BPNet-Real	**0.31**	**0.46**	**0.56**	**0.37**	**0.70**	**0.91**	**0.47**	0.478	0.542	0.612	0.544
ICP	0.01	0.01	0.01	0.07	0.36	0.85	0.01	0.085	0.014	0.032	0.044
FGR(FPFH)	0.00	0.00	0.00	0.08	0.43	0.85	0.00	0.055	0.000	0.009	0.021
TEASER++(FPFH)	0.01	0.02	0.05	0.04	0.26	0.77	0.02	0.096	0.060	0.093	0.083
Super4PCS	0.01	0.03	0.06	0.06	0.31	0.83	0.03	0.054	0.072	0.113	0.080
⋆Vidal-Sensors18	-	-	-	-	-	-	-	0.473	0.625	0.647	0.582
⋆PPF_3D_ICP	-	-	-	-	-	-	-	**0.523**	**0.669**	**0.716**	**0.636**
IDAM-MN40	0.04	0.08	0.11	0.12	0.47	0.88	0.07	0.205	0.112	0.153	0.157
IDAM-AutoSynth	0.14	0.21	0.26	0.23	0.57	0.88	0.20	0.316	0.272	0.322	0.303
BPNet-MN40	0.22	0.32	0.41	0.30	0.63	0.92	0.34	0.395	0.404	0.472	0.423
BPNet-AutoSynth	**0.25**	**0.35**	**0.41**	**0.34**	**0.65**	**0.92**	**0.37**	0.410	0.429	0.501	0.447

Table 3: Quantitative comparison of registration models trained on AutoSynth and ModelNet40 on the **Occluded-LINEMOD** real scene dataset. Vidal-Sensors18 [60] and PPF_3D_ICP [17] are traditional methods and represent the best depth-only performers from the BOP leaderboard.

door to future research on learning to generate training data.

The reason why BPNet-Real and IDAM-Real achieve better performance than these models trained on synthetic data is twofold. First, they work in the easier setting where the test object has been observed during training. Second, there remains a domain gap between real-scene depth maps and synthetic ones. While our results show that our AutoSynth approach bridges part of this gap, further reducing it remains a topic for future research.

LINEMOD dataset. The LINEMOD dataset is more challenging than TUD-L because of the presence of symmetric objects and minor occlusions at the object boundaries. As shown in Tab. 2, even Super4PCS fails to yield meaningful results on this dataset. Our BPNet-AutoSynth and IDAM-AutoSynth again achieve better performance than BPNet-MN40 and IDAM-MN40. This shows that the dataset searched by our AutoSynth algorithm on the Stanford bunny generalizes well to different real-scene datasets. Note that, the IDAM-AutoSynth achieves even better performance than IDAM-Real. This is because the LINEMOD dataset does not provide real depth maps for training data, and we thus used the synthetic ones provided by LINEMOD, which also suffer from a domain gap w.r.t. the real test data. This shows that training on data with more diverse shapes can improve evaluation performance when the domain gap is large.

Occluded-LINEMOD dataset. The Occluded-LINEMOD dataset depicts an even more challenging scenario than LINEMOD by including severe occlusions. As such, as shown in Tab. 3, the results of all the methods deteriorate. Nevertheless, BPNet-AutoSynth and IDAM-AutoSynth still significantly outperform BPNet-MN40 and IDAM-MN40, respectively. This further demonstrates that our searched dataset delivers a consistent performance improvement across different real evaluation datasets and different point cloud registration frameworks.

4.2. Analysis

Here, we conduct ablation studies to analyze (i) the behavior similarity of the main and surrogate task networks; (ii) the impact of the target dataset; (iii) the effectiveness of the guidance from the surrogate network; and (iii) the impact of pre-training on the searched data.

Behavior of the main and surrogate task networks. We conduct experiments to compare the performance of models trained on datasets with different numbers of shapes by adjusting the number of ModelNet40 mesh models used for training. Specifically, we randomly sample $M \in \{1, 5, 10, 50\}$ models per ModelNet40 category. For example, MN40(01_per_cate) was built by taking a single mesh model from each category, and thus contains 40 mesh models. For this set of experiments, we use BPNet as our main task registration network.

We summarize the results in Tab. 4, where we also report the reconstruction errors of the surrogate reconstruction network trained on the same data. These results evidence that both tasks, i.e., registration and reconstruction, follow the same trend as the number of training meshes changes. In short, increasing the number of training models improves pose estimation accuracy and lowers reconstruction error. Importantly, the results obtained with our AutoSynth searched dataset are the best, confirming the effectiveness of our surrogate task network.

Impact of the target dataset D_{tgt}. To assess the influence of D_{tgt} on the search, we compare the use of a scanned real-object with an MN40(01_per_cate) dataset, using BPNet as backbone. Our framework leverages a feedback mechanism to learn from D_{tgt}, which helps to narrow

Method		BPNet							AutoEncoder
		Rotation mAP			Translation mAP			ADD	Recontruction
Dataset	Setting	5°	10°	20°	$1cm$	$2cm$	$5cm$	$0.1d$	Chamfer Dist
TUD-L	MN40(01_per_cate)	0.59	0.62	0.68	0.59	0.71	0.92	0.65	22.27
	MN40(05_per_cate)	0.61	0.66	0.69	0.62	0.74	0.90	0.68	17.81
	MN40(10_per_cate)	0.66	0.70	0.73	0.66	0.82	0.92	0.72	9.97
	MN40(50_per_cate)	0.69	0.72	0.75	0.69	0.81	0.91	0.75	6.82
	MN40-full	0.71	0.74	0.77	0.70	0.80	0.94	0.76	6.65
	AutoSynth	0.78	0.81	0.85	0.77	0.86	0.95	0.84	4.16

Table 4: Comparison of BPNet trained with different datasets and evaluated on the **TUD-L** real scene dataset. For ModelNet40, we pick 1, 5, 10 and 50 mesh models from each category, corresponding to 40, 200, 400, 2000 mesh models. The Chamfer distance is multiplied by 10^3. Note that, for both registration and reconstruction, increasing the number of training mesh models improves the performance.

Method	Rotation mAP			Translation mAP			ADD
	5°	10°	20°	$1cm$	$2cm$	$5cm$	$0.1d$
MN40(01)	0.59	0.62	0.68	0.59	0.71	0.92	0.65
MN40	0.71	0.74	0.77	0.70	0.80	0.94	0.76
AS(MN40(01))	0.76	0.80	0.84	0.77	0.86	0.93	0.82
AS(Real)	**0.78**	**0.81**	**0.85**	**0.77**	**0.86**	**0.95**	**0.84**

Table 5: Results of employing different datasets as the target dataset with BPNet as backbone. MN40(01) stands for MN40(01_per_cate); AS stands for AutoSynth.

Method	Rotation mAP			Translation mAP			ADD
	5°	10°	20°	$1cm$	$2cm$	$5cm$	$0.1d$
full-range	0.66	0.68	0.70	0.64	0.75	0.90	0.69
no-feedback	0.61	0.65	0.69	0.62	0.74	0.89	0.68
Surrogate net	**0.78**	**0.81**	**0.85**	**0.77**	**0.86**	**0.95**	**0.84**

Table 6: Effectiveness of the feedback mechanism. Using our surrogate reconstruction network to guide the search clearly outperforms both selecting a random policy and using the full range policy.

the reality gap when using scanned real-objects. The results presented in Tab. 5 show that AutoSynth(Real) outperforms AutoSynth(MN40(01_per_cate)), which confirms our claim. Note that BPNet trained using MN40(01_per_cate) as D_{syn} yields better results than the one trained on it directly. This is due to the fact that the 3D dataset evolved from MN40(01_per_cate) contains more distinct shapes than it, resulting in better performance.

Effectiveness of the guidance from the surrogate network. Here, we evaluate the effectiveness of the surrogate network $\Psi_{surrogate}$ at guiding the search towards the best policy by comparing it with two alternatives that offer no guidance: (i) A no-feedback strategy corresponding to randomly picking a policy from the search space; (ii) a full-range policy consisting of randomly sampling using the largest possible range of transformations during training. The comparison in Tab. 6 on the TUD-L dataset testing sequence and with BPNet as registration network clearly shows the benefits of the surrogate network for the search.

Method	Rotation mAP			Translation mAP			ADD
	5°	10°	20°	$1cm$	$2cm$	$5cm$	$0.1d$
Real	0.91	0.92	0.93	0.86	0.95	0.99	0.93
AutoSynth	0.78	0.81	0.85	0.77	0.86	0.95	0.84
Pretrain	**0.94**	**0.95**	**0.96**	**0.90**	**0.97**	**1.00**	**0.96**

Table 7: BPNet trained on TUD-L vs AutoSynth vs AutoSynth pre-trained followed by TUD-L fine-tuning.

Impact of pre-training on the searched data. To evaluate the use of our approach as a pre-training strategy, we pre-train the network on the AutoSynth-searched data and fine-tune it on the TUD-L training set. As shown in Tab. 7, this lets us reach a new SOTA performance (**0.94** in R5° mAP), showing the effectiveness of our AutoSynth dataset.

5. Conclusion

We have introduced a novel algorithm to automatically generate large amounts of 3D training dataset and curate the optimal one from the millions of options. To this end, we have proposed to use a surrogate reconstruction network while searching for a data generation policy, thus accelerating the search by 4056.43 times. We have evidenced the generality of our approach by evaluating it with two different point cloud registration methods, BPNet and IDAM. Our experiments on real-scene datasets have evidenced that a network trained on our searched dataset consistently outperforms the same model trained on the widely used ModelNet40 dataset. As shown by our results, however, there remains a gap between our searched dataset and real scans. In the future, we will study how to further bridge this gap by improving the realism of the synthesized data.

6. Acknowledgements

Zheng Dang would like to thank H. Chen for the highly-valuable discussions and for her encouragement. This work was funded in part by the Swiss Innovation Agency (Innosuisse).

References

[1] Gabriel Agamennoni, Simone Fontana, Roland Y Siegwart, and Domenico G Sorrenti. Point clouds registration with probabilistic data association. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4092–4098. IEEE, 2016. 2

[2] Dror Aiger, Niloy J Mitra, and Daniel Cohen-Or. 4-points congruent sets for robust pairwise surface registration. In *ACM SIGGRAPH 2008 papers*, pages 1–10, 2008. 2

[3] Yasuhiro Aoki, Hunter Goforth, Rangaprasad Arun Srivatsan, and Simon Lucey. Pointnetlk: Robust & efficient point cloud registration using pointnet. In *Conference on Computer Vision and Pattern Recognition*, pages 7163–7172, Long Beach, California, 2019. 2

[4] Harkirat Singh Behl, Atilim Güneş Baydin, Ran Gal, Philip HS Torr, and Vibhav Vineet. Autosimulate:(quickly) learning synthetic data generation. In *European Conference on Computer Vision*, pages 255–271. Springer, 2020. 2

[5] P. Besl and N. Mckay. A method for registration of 3d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, February 1992. 2

[6] Sofien Bouaziz, Andrea Tagliasacchi, and Mark Pauly. Sparse iterative closest point. In *Computer graphics forum*, volume 32, pages 113–123, Hoboken, New Jersey, 2013. Wiley Online Library. 2

[7] Eric Brachmann, Alexander Krull, Frank Michel, Stefan Gumhold, Jamie Shotton, and Carsten Rother. Learning 6d object pose estimation using 3d object coordinates. In *European conference on computer vision*, pages 536–551, Zürich, Switzerland, 2014. Springer. 1, 2

[8] Alexander M Bronstein and Michael M Bronstein. Regularized partial matching of rigid shapes. In *European Conference on Computer Vision*, pages 143–154. Springer, 2008. 2

[9] Alexander M Bronstein, Michael M Bronstein, Alfred M Bruckstein, and Ron Kimmel. Partial similarity of objects, or how to compare a centaur to a horse. *International Journal of Computer Vision*, 84(2):163–183, 2009. 2

[10] Jeff Clune and Hod Lipson. Evolving 3d objects with a generative encoding inspired by developmental biology. *ACM SIGEVOlution*, 5(4):2–12, 2011. 3

[11] Jasmine Collins, Shubham Goel, Kenan Deng, Achleshwar Luthra, Leon Xu, Erhan Gundogdu, Xi Zhang, Tomas F Yago Vicente, Thomas Dideriksen, Himanshu Arora, et al. Abo: Dataset and benchmarks for real-world 3d object understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21126–21136, 2022. 2

[12] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 113–123, 2019. 3

[13] Zheng Dang, Wang Lizhou, Guo Yu, and Mathieu Salzmann. Learning-based point cloud registration for 6d object pose estimation in the real world. In *European Conference on Computer Vision*, 2022. 1, 2, 5

[14] Jeevan Devaranjan, Amlan Kar, and Sanja Fidler. Metasim2: Unsupervised learning of scene structure for synthetic data generation. In *European Conference on Computer Vision*, pages 715–733. Springer, 2020. 2

[15] Andreas Doumanoglou, Rigas Kouskouridas, Sotiris Malassiotis, and Tae-Kyun Kim. Recovering 6d object pose and predicting next-best-view in the crowd. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3583–3592, Las Vegas, Nevada, 2016. 2

[16] Bertram Drost, Markus Ulrich, Paul Bergmann, Philipp Hartinger, and Carsten Steger. Introducing mvtec itodd-a dataset for 3d object recognition in industry. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 2200–2208, Venice, Italy, 2017. 2

[17] Bertram Drost, Markus Ulrich, Nassir Navab, and Slobodan Ilic. Model globally, match locally: Efficient and robust 3d object recognition. In *2010 IEEE computer society conference on computer vision and pattern recognition*, pages 998–1005, 2010. 2, 6, 7

[18] Andrew W Fitzgibbon. Robust registration of 2d and 3d point sets. *Image and vision computing*, 21(13-14):1145–1153, 2003. 2

[19] Kexue Fu, Shaolei Liu, Xiaoyuan Luo, and Manning Wang. Robust point cloud registration framework based on deep graph matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8893–8902, 2021. 2

[20] Ruohan Gao, Zilin Si, Yen-Yu Chang, Samuel Clarke, Jeannette Bohg, Li Fei-Fei, Wenzhen Yuan, and Jiajun Wu. Objectfolder 2.0: A multisensory object dataset for sim2real transfer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10598–10608, 2022. 2

[21] Natasha Gelfand, Niloy J Mitra, Leonidas J Guibas, and Helmut Pottmann. Robust global registration. In *Symposium on geometry processing*, page 5. Vienna, Austria, 2005. 2

[22] Dirk Hähnel and Wolfram Burgard. Probabilistic matching for 3d scan registration. In *Proc. of the VDI-Conference Robotik*, volume 2002. Citeseer, 2002. 2

[23] Stefan Hinterstoisser, Vincent Lepetit, Slobodan Ilic, Stefan Holzer, Gary Bradski, Kurt Konolige, and Nassir Navab. Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes. In *Asian Conference on Computer Vision*, pages 548–562, Daejeon, 2012. 1, 2

[24] Timo Hinzmann, Thomas Stastny, Gianpaolo Conte, Patrick Doherty, Piotr Rudol, Marius Wzorek, Enric Galceran, Roland Siegwart, and Igor Gilitschenski. Collaborative 3d reconstruction using heterogeneous uavs: System and experiments. In *International Symposium on Experimental Robotics*, pages 43–56. Springer, 2016. 2

[25] Tomáš Hodan, Pavel Haluza, Štepán Obdržálek, Jiri Matas, Manolis Lourakis, and Xenophon Zabulis. T-less: An rgb-d dataset for 6d pose estimation of texture-less objects. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 880–888, Santa Rosa, CA, USA, 2017. IEEE. 2

[26] Tomas Hodan, Frank Michel, Eric Brachmann, Wadim Kehl, Anders GlentBuch, Dirk Kraft, Bertram Drost, Joel Vidal, Stephan Ihrke, Xenophon Zabulis, et al. Bop: Benchmark for 6D Object Pose Estimation. In *European Conference on Computer Vision*, pages 19–34, Munich, Germany, 2018. 1, 2

[27] Gregory Izatt, Hongkai Dai, and Russ Tedrake. Globally optimal object pose estimation in point clouds with mixed-integer programming. In *Robotics Research*, pages 695–710, Ventura, CA, 2020. Springer. 2

[28] Xiaotong Jiang, Qingjin Peng, Xiaosheng Cheng, Ning Dai, Cheng Cheng, and Dawei Li. Efficient booleans algorithms for triangulated meshes of geometric modeling. *Computer-Aided Design and Applications*, 13(4):419–430, 2016. 3

[29] Andrew E. Johnson and Martial Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. *TPAMI*, 21(5):433–449, 1999. 2

[30] Amlan Kar, Aayush Prakash, Ming-Yu Liu, Eric Cameracci, Justin Yuan, Matt Rusiniak, David Acuna, Antonio Torralba, and Sanja Fidler. Meta-sim: Learning to generate synthetic datasets. In *Conference on Computer Vision and Pattern Recognition*, pages 4551–4560, 2019. 2

[31] Roman Kaskman, Sergey Zakharov, Ivan Shugurov, and Slobodan Ilic. Homebreweddb: Rgb-d dataset for 6d pose estimation of 3d objects. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 0–0, Seoul, Korea, 2019. 2

[32] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, San Diego, CA, USA, 2015. 5

[33] Huu M Le, Thanh-Toan Do, Tuan Hoang, and Ngai-Man Cheung. Sdrsac: Semidefinite-based randomized approach for robust point cloud registration without correspondences. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 124–133, 2019. 2

[34] Jiahao Li, Changhao Zhang, Ziyao Xu, Hangning Zhou, and Chi Zhang. Iterative distance-aware similarity matrix convolution with mutual-supervised point elimination for efficient point cloud registration. In *ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIV 16*, pages 378–394. Springer, 2020. 1, 2, 5

[35] Or Litany, Alexander M Bronstein, and Michael M Bronstein. Putting the pieces together: Regularized multi-part shape matching. In *European Conference on Computer Vision*, pages 1–11. Springer, 2012. 2

[36] Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search. In *European Conference on Computer Vision*, pages 19–34, 2018. 3

[37] Liu Liu, Wenqiang Xu, Haoyuan Fu, Sucheng Qian, Qiaojun Yu, Yang Han, and Cewu Lu. Akb-48: A real-world articulated object knowledge base. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14809–14818, 2022. 2

[38] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. *ACM siggraph computer graphics*, 21(4):163–169, 1987. 3

[39] Bruce D Lucas, Takeo Kanade, et al. An iterative image registration technique with an application to stereo vision. In *International Joint Conference on Artificial Intelligence*, Vancouver, British Columb, 1981. 2

[40] Haggai Maron, Nadav Dym, Itay Kezurer, Shahar Kovalsky, and Yaron Lipman. Point registration via efficient convex relaxation. *ACM Transactions on Graphics (TOG)*, 35(4):1–12, 2016. 2

[41] Nicolas Mellado, Dror Aiger, and Niloy J Mitra. Super 4pcs fast global pointcloud registration via smart indexing. In *Computer graphics forum*, volume 33, pages 205–215. Wiley Online Library, 2014. 2

[42] Mustafa Mohamad, Mirza Tahir Ahmed, David Rappaport, and Michael Greenspan. Super generalized 4pcs for 3d registration. In *2015 International Conference on 3D Vision*, pages 598–606. IEEE, 2015. 2

[43] Juan-Manuel Perez-Rua, Moez Baccouche, and Stephane Pateux. Efficient progressive neural architecture search. *arXiv preprint arXiv:1808.00391*, 2018. 3

[44] François Pomerleau, Francis Colas, Roland Siegwart, et al. A review of point cloud registration algorithms for mobile robotics. *Foundations and Trends® in Robotics*, 4(1):1–104, 2015. 2

[45] C.R. Qi, H. Su, K. Mo, and L.J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Conference on Computer Vision and Pattern Recognition*, Honolulu, Hawaii, 2017. 2

[46] Carolina Raposo and Joao P Barreto. Using 2 point+ normal sets for fast registration of point clouds with small overlap. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5652–5658. IEEE, 2017. 2

[47] Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Jie Tan, Quoc V Le, and Alexey Kurakin. Large-scale evolution of image classifiers. In *International Conference on Machine Learning*, pages 2902–2911. PMLR, 2017. 4

[48] Colin Rennie, Rahul Shome, Kostas E Bekris, and Alberto F De Souza. A dataset for improved rgbd-based object detection and pose estimation for warehouse pick-and-place. *IEEE Robotics and Automation Letters*, 1(2):1179–1185, 2016. 2

[49] Antonio Ricci. A constructive geometry for computer graphics. *The Computer Journal*, 16(2):157–160, 1973. 3

[50] David M Rosen, Luca Carlone, Afonso S Bandeira, and John J Leonard. Se-sync: A certifiably correct algorithm for synchronization over the special euclidean group. *The International Journal of Robotics Research*, 38(2-3):95–125, 2019. 2

[51] Nataniel Ruiz, Samuel Schulter, and Manmohan Chandraker. Learning to simulate. In *International Conference on Learning Representations*, 2019. 2

[52] Szymon Rusinkiewicz and Marc Levoy. Efficient variants of the icp algorithm. In *Proceedings Third International Conference on 3-D Digital Imaging and Modeling*, pages 145–152, Quebec City, Canada, 2001. IEEE. 2

[53] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (fpfh) for 3d registration. In *International Conference on Robotics and Automation*, pages 3212–3217, Kobe, Japan, 2009. IEEE. 2

[54] Radu Bogdan Rusu, Nico Blodow, Zoltan Csaba Marton, and Michael Beetz. Aligning point cloud views using persistent feature histograms. In *International Conference on Intelligent Robots and Systems*, pages 3384–3391, Nice, France, 2008. IEEE. 2

[55] Aleksandr Segal, Dirk Haehnel, and Sebastian Thrun. Generalized-icp. In *In Robotics: Science and Systems*, Cambridge, 2009. 2

[56] Kenneth O Stanley. Compositional pattern producing networks: A novel abstraction of development. *Genetic programming and evolvable machines*, 8(2):131–162, 2007. 3

[57] Deqing Sun, Daniel Vlasic, Charles Herrmann, Varun Jampani, Michael Krainin, Huiwen Chang, Ramin Zabih, William T Freeman, and Ce Liu. Autoflow: Learning a better training set for optical flow. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10093–10102, 2021. 2

[58] Alykhan Tejani, Danhang Tang, Rigas Kouskouridas, and Tae-Kyun Kim. Latent-class hough forests for 3d object detection and pose estimation. In *European Conference on Computer Vision*, pages 462–477, Zürich, Switzerland, 2014. Springer. 2

[59] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, Long Beach, California, United States, 2017. 2

[60] Joel Vidal, Chyi-Yeu Lin, Xavier Lladó, and Robert Martí. A method for 6d pose estimation of free-form rigid objects using point pair features on range data. *Sensors*, 18(8):2678, 2018. 2, 6, 7

[61] Pengyuan Wang, HyunJun Jung, Yitong Li, Siyuan Shen, Rahul Parthasarathy Srikanth, Lorenzo Garattoni, Sven Meier, Nassir Navab, and Benjamin Busam. Phocal: A multi-modal dataset for category-level object pose estimation with photometrically challenging objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21222–21231, 2022. 2

[62] Yue Wang and Justin M Solomon. Deep closest point: Learning representations for point cloud registration. In *International Conference on Computer Vision*, pages 3523–3532, Seoul, Korea, 2019. 2

[63] Yue Wang and Justin M Solomon. Prnet: Self-supervised learning for partial-to-partial registration. In *Advances in Neural Information Processing Systems*, pages 8812–8824, Vancouver, British Columbia, Canada, 2019. 2

[64] Y. Wang, Y. Sun, Z. Liu, S. Sarma, M. Bronstein, and J.M. Solomon. Dynamic graph cnn for learning on point clouds. In *ACM Transactions on Graphics (TOG)*, TOG, 2019. 2

[65] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Conference on Computer Vision and Pattern Recognition*, pages 1912–1920, Boston, MA, USA, 2015. 1, 2, 5

[66] Yu Xiang, Wonhui Kim, Wei Chen, Jingwei Ji, Christopher Choy, Hao Su, Roozbeh Mottaghi, Leonidas Guibas, and Silvio Savarese. Objectnet3d: A large scale database for 3d object recognition. In *European conference on computer vision*, pages 160–176. Springer, 2016. 2

[67] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. In *Robotics: Science and Systems Conference*, Pittsburgh, PA, USA, 2018. 2

[68] Dawei Yang and Jia Deng. Shape from shading through shape evolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3781–3790, 2018. 2, 3

[69] Dawei Yang and Jia Deng. Learning to generate 3d training data through hybrid gradient. In *Conference on Computer Vision and Pattern Recognition*, pages 779–789, 2020. 3

[70] H. Yang and L. Carlone. A polynomial-time solution for robust registration with extreme outlier rates. In *Robotics: Science and Systems Conference*, Freiburg im Breisgau, Germany, 2019. 2

[71] H. Yang, J. Shi, and L. Carlone. Teaser: Fast and certifiable point cloud registration. In *arXiv Preprint*, 2020. 2

[72] Jiaolong Yang, Hongdong Li, Dylan Campbell, and Yunde Jia. Go-icp: A globally optimal solution to 3d icp point-set registration. *TPAMI*, 38(11):2241–2254, 2015. 2

[73] Zi Jian Yew and Gim Hee Lee. Rpm-net: Robust point matching using learned features. In *Conference on Computer Vision and Pattern Recognition*, Online, 2020. 2

[74] Wentao Yuan, Benjamin Eckart, Kihwan Kim, Varun Jampani, Dieter Fox, and Jan Kautz. Deepgmr: Learning latent gaussian mixture models for registration. In *European Conference on Computer Vision*, pages 733–750. Springer, 2020. 2

[75] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. In *Advances in Neural Information Processing Systems*, pages 3391–3401, Long Beach, California, United States, 2017. 2

[76] Amir R Zamir, Alexander Sax, William Shen, Leonidas J Guibas, Jitendra Malik, and Silvio Savarese. Taskonomy: Disentangling task transfer learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3712–3722, 2018. 5

[77] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Fast global registration. In *European Conference on Computer Vision*, pages 766–782, Amsterdam, the Netherlands, 2016. Springer. 2