ISSN 1064-2269, Journal of Communications Technology and Electronics, 2024. © Pleiades Publishing, Inc., 2024.
Russian Text © The Author(s), 2024, published in Informatsionnye Protsessy, 2024, Vol. 24, No. 1, pp. 105–117.

THEORY AND METHODS OF INFORMATION PROCESSING

Registration of Point Clouds in 3D Space Using Soft Alignment

A. Yu. Makovetskii[a, *], V. I. Kober[b, c, **], S. M. Voronin[a], A. V. Voronin[a], V. N. Karnaukhov[b], and M. G. Mozerov[b]

[a] *Chelyabinsk State University, Chelyabinsk, 454001 Russia*
[b] *Institute for Information Transmission Problems, Russian Academy of Sciences, Moscow, 127051 Russia*
[c] *Ensenada Center for Scientific Research and Higher Education, Ensenada, 22860 Mexico*
**e-mail: artemmac@csu.ru*
***e-mail: vitaly@iitp.ru*

Received March 28, 2024; revised March 28, 2024; accepted May 13, 2024

Abstract—There was significant recent progress in the field of deep learning, which has led to compelling advances in most tasks of semantic computer vision (e.g., classification, detection, and segmentation). Point cloud registration is a problem in which two or more different point clouds are aligned by estimation of the relative geometric transformation between them. This well-known problem plays an important role in many applications such as SLAM, 3D reconstruction, mapping, positioning, and localization. The complexity of the point cloud registration increases due to the difficulty of feature extraction related to a large difference in the appearances of a single object obtained by a laser scanner from different points of view. Millions of points created every second require high-efficiency algorithms and powerful computing devices. The well-known ICP algorithm for point cloud registration and its variants have relatively high computational efficiency, but are known to be immune to local minima and, therefore, rely on the quality of the initial rough alignment. Algorithm operation with the interference caused by noisy points on dynamic objects is usually critical for obtaining a satisfactory estimate, especially when using real LiDAR data. In this study, we propose a neural network algorithm to solve the problem of point cloud registration by estimating the soft alignment of the points of the source and target point clouds. The proposed algorithm efficiently works with incongruent noisy point clouds generated by LiDAR. Results of computer simulation are presented to illustrate the efficiency of the proposed algorithm.

Keywords: neural network, point cloud, registration, surface reconstruction, soft alignment
DOI: 10.1134/S1064226924700165

INTRODUCTION

Registration of 3D point clouds plays a key role in robotics and computer vision, allowing identification of a rigid transformation for alignment of two point clouds with unknown point relations. Such a process has been widely applied in various fields, including reconstruction of 3D scenes [1–3], localization [4], and autonomous driving [5]. The conventional iterative closest point (ICP) method has long been a registration standard [6–14], involving alternating steps of solving point alignment and applying rigid transformations. However, ICP relies on good initial alignment and often converges to local minima. Recent advent of deep learning models [15–23] has revolutionized computer vision, leading to the creation of faster and more reliable point cloud registration algorithms compared to classical methods. Several algorithms based on deep learning (e.g., deep closest point (DCP) [18], PRNet [19], RPMNet [20], and IDAM [21]) employ neural networks to establish alignment using multidimensional descriptors. The DCP neural network using Dynamic Graph CNN (DGCNN) extracts local objects from point clouds for soft fitting and least squares solutions, assuming a one-to-one correspondence in two point clouds. PRNet extends DCP by including a keypoint detection module for partial registration. RPMNet uses annealing to obtain soft assignments to point alignment from hybrid objects obtained from both spatial coordinates and local geometry. IDAM combines features and Euclidean information into a corresponding matrix, using a two-stage learning method to eliminate points for registration.

On the contrary, the proposed method employs a two-branch strategy for object description that includes both location information and rotation-independent local objects. This strategy aims to achieve multi-dimensional point cloud embedding, differing from existing methods that substantially rely on similarity of object descriptors and face problems related to large rotations and significant differences in cloud coordinates. In this work, we propose a simple neural network architecture that uses soft alignment of points of source and target clouds and a weighted two-point

ICP functional instead of a conventional two-point ICP functional [6, 24]. The weight of a pair of points is chosen based on the probability that the pair belongs to common subsets of cloud points. The point weights are calculated based on the point descriptors calculated in the proposed architecture. The proposed neural network algorithm uses elements of the PointNet++ network [25]. The network was trained on the ModelNrt40 database [26]. Computer simulation illustrates the operation of the proposed algorithm.

The structure of the article is as follows. Section 1 presents the formulation of the problem and describes the algorithms for solving it. Section 2 presents the results of computer modeling. Section 3 contains the conclusions.

1. NEURAL NETWORK ARCHITECTURE

This paper describes a simple architecture of neural network using PointNet++ network elements [25]. The proposed neural network algorithm calculates descriptors of all points in point clouds, determines point alignment, and calculates transformations from the $SE(3)$ group to align two given point clouds.

Let $P = p_1,\ldots, p_{s_p}$ and $Q = q_1,\ldots,q_{s_q}$ be the source and target point clouds ($p_i, q_j \in R^3, i = 1,\ldots, s_p$, and $j = 1,\ldots, s_q$). We consider point $p_i \in P$ and its neighborhood in P with radius r. In this neighborhood, we consider K points. Additional points are ignored if the neighborhood contains more than K points, and copies of a point from the neighborhood are added if the neighborhood contains less than K points. Thus, we map a set containing K elements to point p_i and do this for all points from P and Q. The descriptors of all points in P are calculated as follows.

Let B be the batch size.

First layer. The original tensor has dimension $(B, 3, s_p)$. We use the previously calculated neighborhoods of all points s_p and obtain tensor $(B, s_p, K, 3)$; after rearranging the dimensions, the tensor dimension is $(B, 3, K, s_p)$. We use 2D convolution with parameters *in_channels* = 3, *out_channels* = 64, *kernel_size* = 1, *stride* = 1, and *padding* = 0, so that the tensor dimension after convolution is $(B, 64, K, s_p)$. The ReLu activation function and batch normalization are used. After convolution, the maximum in the tensor along the dimension with index 2 (the dimension is K) is calculated and the output tensor has parameters $(B, 64, s_p)$.

Third layer. The input tensor has dimension $(B, 128, s_p)$. We use the previously calculated neighborhoods of all points s_p and obtain tensor $(B, s_p, K, 128)$; after rearranging the dimensions, the tensor dimension is $(B, 128, K, s_p)$. We use 2D convolution with parameters *in_channels* = 128, *out_channels* = 256, *kernel_size* = 1, *stride* = 1, and *padding* = 0, so that the tensor dimension after convolution is $(B, 256, K, s_p)$. The ReLu activation function and batch normalization are used. After convolution, the maximum in the tensor along the dimension with index 2 (the dimension is K) is calculated and the output tensor has parameters $(B, 256, s_p)$.

Fourth layer. The input tensor has dimension $(B, 256, s_p)$. We use the previously calculated neighborhoods of all points s_p and obtain tensor $(B, s_p, K, 256)$; after rearranging the dimensions, the tensor dimension is $(B, 256, K, s_p)$. We use 2D convolution with parameters *in_channels* = 256, *out_channels* = 256, *kernel_size* = 1, *stride* = 1, and *padding* = 0, so that the tensor dimension after convolution is $(B, 256, K, s_p)$. The ReLu activation function and batch normalization are used. After convolution, the maximum in the tensor along the dimension with index 2 (the dimension is K) is calculated and the output tensor has parameters $(B, 256, s_p)$.

Fifth layer. The input tensor has dimension $(B, 256, s_p)$. We use the previously calculated neighborhoods of all points s_p and obtain tensor $(B, s_p, K, 256)$; after rearranging the dimensions, the tensor dimension is $(B, 256, K, s_p)$. We use 2D convolution with parameters *in_channels* = 256, *out_channels* = 512, *kernel_size* = 1, *stride* = 1, and *padding* = 0, so that the tensor dimension after convolution is $(B, 512, K, s_p)$. The ReLu activation function and batch normalization are used. After convolution, the maximum in the tensor along the dimension with index 2 (the dimension is K) is calculated and the output tensor has parameters $(B, 512, s_p)$.

A vector of size 512 is considered as a descriptor of point P. Similarly, we calculate the descriptors of points in the point cloud Q. When the descriptors of all points in P and Q are calculated, we generate matrix W of size $s_p \times s_q$. Elements of the matrix W_{ij} are the probabilities of alignment of points p_i and q_j ($i = 1, \ldots, s_p, j = 1, \ldots, s_q$). Matrix W is used to form a virtual point cloud $(vQ) = (vq)_1,\ldots,(vq)_{s_p}$

$$(vq)_i = \sum_{j=1}^{s_q} W_{ij} q_j, \tag{1}$$

where $i = 1, \ldots, s_p$. We assume that point p_i corresponds to point $(vq)_i$ ($i = 1, \ldots, s_p$). Thus, we set up a soft alignment of point clouds P and Q. Then, we search for the maximum value w_i in the ith row of matrix W. The value is considered as weight coefficient w_i of the pair $(p_i,(vq)_i)$ ($i = 1, \ldots, s_p$). We consider the following functional:

$$J(R,T) = \sum_{i=1}^{s_p} w_i \left\| Rp_i + T - (vq)_i \right\|^2, \tag{2}$$

where $R \in SO(3)$, $T \in R^3$ and the conditional variational problem

$$(R_*, T_*) = \arg\min_{(R,T)} J(R,T), \tag{3}$$

provided that $R \in SO(3)$. Let C_p be the center of mass of cloud P

$$C_p = \left(1 \Big/ \sum_{i=1}^{s_p} w_i\right) \sum_{j=1}^{s_p} p_j, \tag{4}$$

and P' be the centroid of cloud P

$$p_i' = p_i - C_p, \tag{5}$$

where $i = 1, ..., s_p$. Let $\mathbf{P}'$ be the centroid of matrix P

$$\mathbf{P}' = \begin{pmatrix} p_{11}' & p_{s_p1}' \\ p_{12}' & \cdots \; p_{s_p2}' \\ p_{13}' & p_{s_p3}' \end{pmatrix}. \tag{6}$$

Similarly, matrix $(\mathbf{vQ})'$ is the centroid of $(vQ)'$

$$\mathbf{P}' = \begin{pmatrix} (vq)_{11}' & (vq)_{s1}' \\ (vq)_{12}' & \cdots \; (vq)_{s2}' \\ (vq)_{13}' & (vq)_{s3}' \end{pmatrix}. \tag{7}$$

The following matrix is also introduced:

$$\mathbf{P}'' = \begin{pmatrix} w_1 p_{11}' & w_{s_p} p_{s_p1}' \\ w_1 p_{12}' & \cdots \; w_{s_p} p_{s_p2}' \\ w_1 p_{13}' & w_{s_p} p_{s_p1}' \end{pmatrix}. \tag{8}$$

Variational problem (3) can be reduced to the conditional variational problem

$$R_* = \arg\max_R \left\langle R\mathbf{P}'', (\mathbf{vQ})' \right\rangle, \tag{9}$$

provided that $R \in SO(3)$. Matrix M is given by

$$M = (\mathbf{vQ})'(\mathbf{P}'')^t. \tag{10}$$

Matrix M can be represented as a matrix product using decomposition with respect to singular values:

$$M = USV^t, \tag{11}$$

where U and V^t are orthogonal matrices and S is a diagonal matrix. Note that PyTorch supports the SVD decomposition and backpropagation of error for it. The solution to variational problem (9) is written as

$$R_* = \begin{cases} UV^t, & \text{if } \det(U)\det(V^t) = 1 \\ Udiag(1,1,-1)V^t, & \text{if } \det(U)\det(V^t) = -1 \end{cases}, \tag{12}$$

and the parallel translation vector can be calculated as

$$T_* = \left(1 \Big/ \sum_{i=1}^{s_p} w_i\right) \sum_{j=1}^{s_p} w_j((vq)_j - R_* p_j). \tag{13}$$

The number of trainable weights of the resulting network is 480344. The error function is the MSE weighted sum between the estimated and true transformation (rotation and translation) matrices.

The Adam algorithm was used with a learning rate parameter of 0.001.

The batch size is $B = 10$, and the neighborhood size is $K = 32$. The network was trained using pairs of point clouds from part of the ModelNet40 database [26].

The network was trained for 5 epochs. The loss functions were loss = 0.66, 0.0339, 0.0114, 0.0081, 0.0076, and 0.0075 prior to training and after the first, second, third, fourth, and fifth epochs, respectively.

2. COMPUTER SIMULATION

We apply the proposed neural network to the ModelNet40 database [26]. The database contains 12311 mesh CAD models belonging to 40 classes. The training part of the database consists of 9843 point clouds, and the testing part consists of 2468 point clouds. For each model, 1024 points are uniformly selected from the mesh faces. Only information from the database about the coordinates of the point is used. For each cloud $P = \{p_1, ..., p_{1024}\}$, rotation angles relative to the x, y, and z axes are randomly selected from an interval of $[-\pi/8; \pi/8]$ and coordinates of the translation vector are randomly selected from an interval of $[-0.25; 0.25]$. The matrix (in homogeneous coordinates) of this transformation is denoted as M_{true}. We apply matrix M_{true} to matrix $\mathbf{P}$ related to cloud P and obtain matrix $\mathbf{Q}$ related to cloud $Q = \{q_1, ..., q_{1024}\}$:

$$\mathbf{Q} = M_{\text{true}}\mathbf{P}. \tag{14}$$

We use the proposed neural network (CNN) as an algorithm for rough alignment of point clouds and point-to-point ICP (PtP) as a refinement algorithm. To demonstrate the efficiency of the proposed method, ten experiments were performed, and the results are presented in the following figures and tables.

Figure 1a shows the initial position of source point cloud P and target point cloud Q, Fig. 1b shows the position of the point clouds after CNN, and Fig. 1c shows the position of the point clouds after CNN + PtP. Table 1 presents the errors of the considered algorithms with respect to the L_1 and L_2 norms.

Figure 2a shows the initial position of source point cloud P and target point cloud Q, Fig. 2b shows the position of the point clouds after CNN, and Fig. 2c shows the position of the point clouds after CNN + PtP. Table 2 presents the errors of the considered algorithms with respect to the L_1 and L_2 norms.

Table 1. Accuracy of calculation of rotation and parallel translation with respect to norms L_1 and L_2

	R_{CNN}	T_{CNN}	$R_{CNN+ICP}$	$T_{CNN+ICP}$
L_1	0.949362	0.165098	0.000002	0.000000
L_2	0.376923	0.101902	0.000001	0.000000

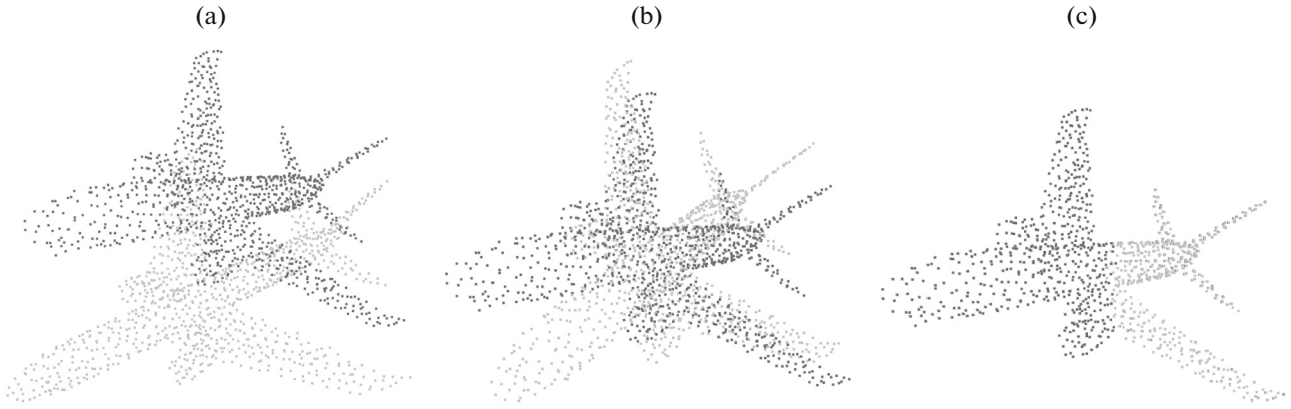

Fig. 1. (a) Point clouds: (gray dots) source and (black dots) target; (b) results of the CNN registration; and (c) result of the CNN + PtP registration.

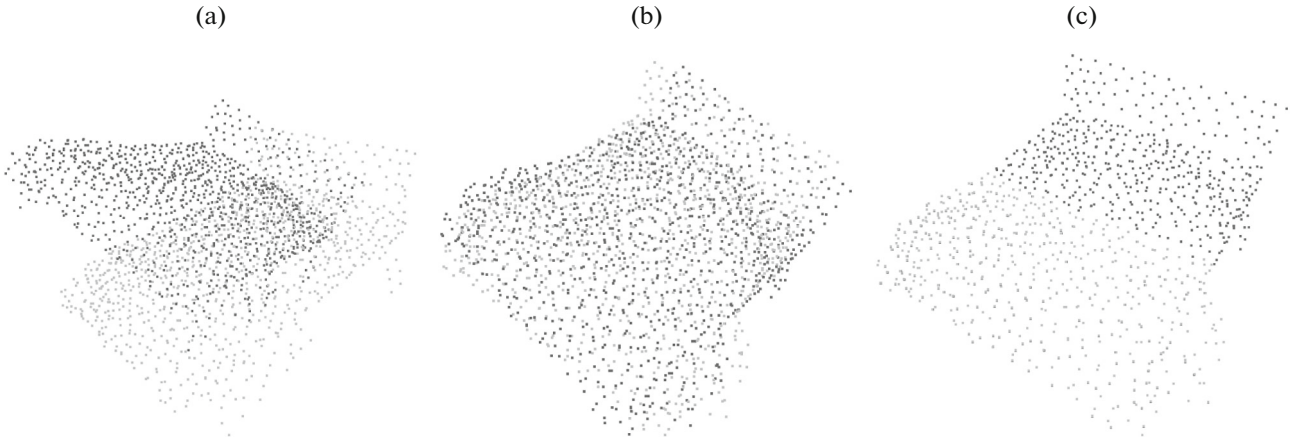

Fig. 2. (a) Point clouds: (gray dots) source and (black dots) target; (b) results of the CNN registration; and (c) result of the CNN + PtP registration.

Figure 3a shows the initial position of source point cloud P and target point cloud Q, Fig. 3b shows the position of the point clouds after CNN, and Fig. 3c shows the position of the point clouds after CNN + PtP. Table 3 presents the errors of the considered algorithms with respect to the L_1 and L_2 norms.

Figure 4a shows the initial position of source point cloud P and target point cloud Q, Fig. 4b shows the position of the point clouds after CNN, and Fig. 4c shows the position of the point clouds after CNN + PtP. Table 5 presents the errors of the considered algorithms with respect to the L_1 and L_2 norms.

Figure 5a shows the initial position of source point cloud P and target point cloud Q, Fig. 5b shows the position of the point clouds after CNN, and Fig. 5c shows the position of the point clouds after CNN + PtP. Table 5 presents the errors of the considered algorithms with respect to the L_1 and L_2 norms.

Figure 6a shows the initial position of source point cloud P and target point cloud Q, Fig. 6b shows the position of the point clouds after CNN, and Fig. 6c shows the position of the point clouds after CNN + PtP. Table 6 presents the errors of the considered algorithms with respect to the L_1 and L_2 norms.

Table 2. Accuracy of calculation of rotation and parallel translation with respect to norms L_1 and L_2

	R_{CNN}	T_{CNN}	$R_{CNN+ICP}$	$T_{CNN+ICP}$
L_1	0.505952	0.080382	0.000002	0.000000
L_2	0.227929	0.059801	0.000001	0.000000

Table 3. Accuracy of calculation of rotation and parallel translation with respect to norms L_1 and L_2

	R_{CNN}	T_{CNN}	$R_{CNN+ICP}$	$T_{CNN+ICP}$
L_1	0.059883	0.005147	0.000002	0.000000
L_2	0.028952	0.004066	0.000001	0.000000

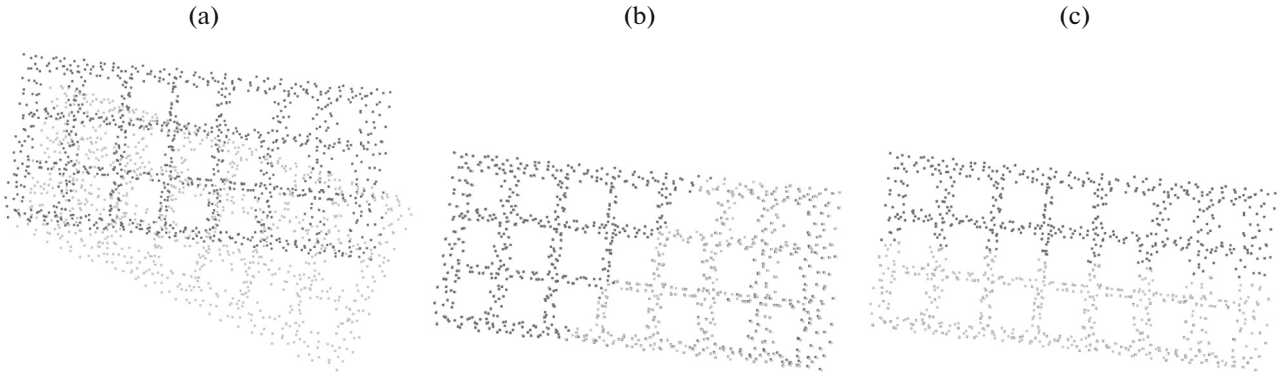

Fig. 3. (a) Point clouds: (gray dots) source and (black dots) target; (b) results of the CNN registration; and (c) result of the CNN + PtP registration.

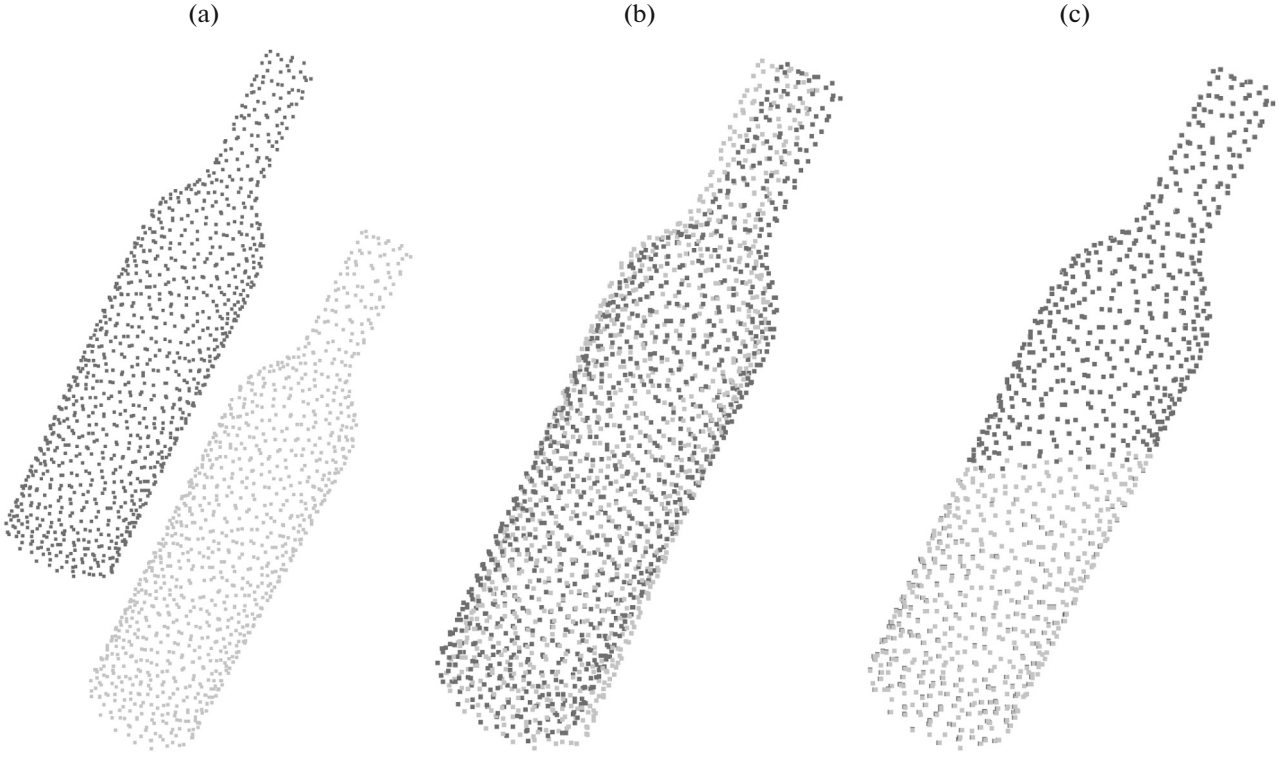

Fig. 4. (a) Point clouds: (gray dots) source and (black dots) target; (b) results of the CNN registration; and (c) result of the CNN + PtP registration.

Figure 7a shows the initial position of source point cloud P and target point cloud Q, Fig. 7b shows the position of the point clouds after CNN, and Fig. 7c shows the position of the point clouds after CNN + PtP. Table 7 presents the errors of the considered algorithms with respect to the L_1 and L_2 norms.

Figure 8a shows the initial position of source point cloud P and target point cloud Q, Fig. 8b shows the

Table 4. Accuracy of calculation of rotation and parallel translation with respect to norms L_1 and L_2

	R_{CNN}	T_{CNN}	$R_{CNN+ICP}$	$T_{CNN+ICP}$
L_1	0.376001	0.121047	0.000001	0.000000
L_2	0.167898	0.071043	0.000001	0.000000

Table 5. Accuracy of calculation of rotation and parallel translation with respect to norms L_1 and L_2

	R_{CNN}	T_{CNN}	$R_{CNN+ICP}$	$T_{CNN+ICP}$
L_1	1.948246	0.304671	0.000001	0.000000
L_2	1.948246	0.187354	0.000001	0.000000

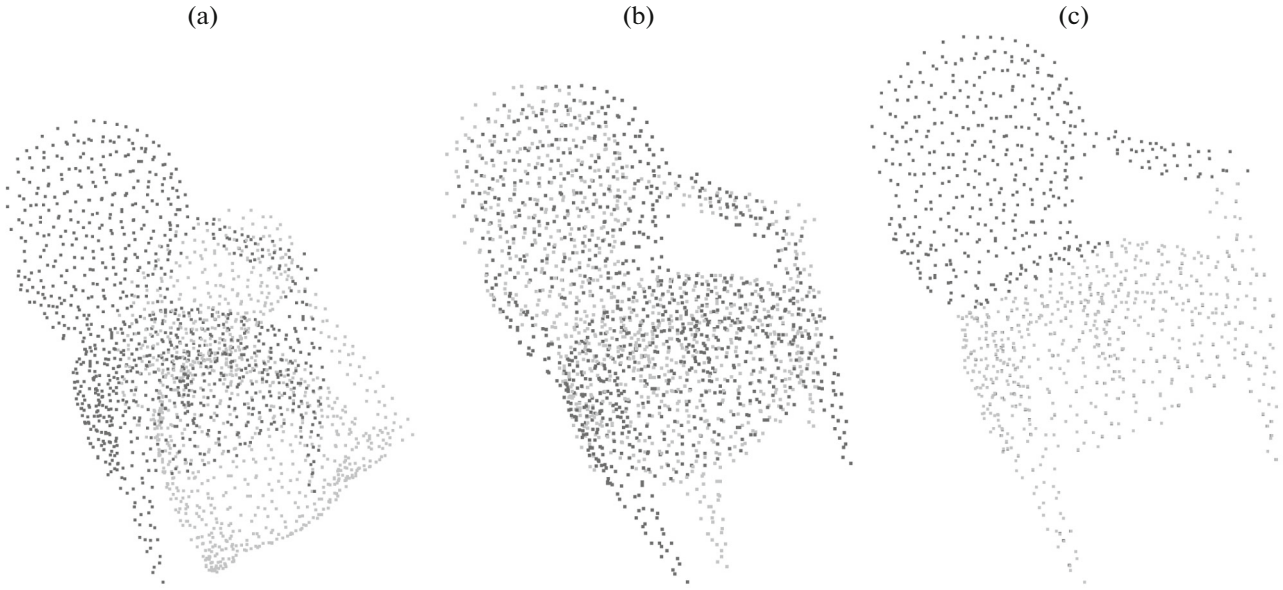

Fig. 5. (a) Point clouds: (gray dots) source and (black dots) target; (b) results of the CNN registration; and (c) result of the CNN + PtP registration.

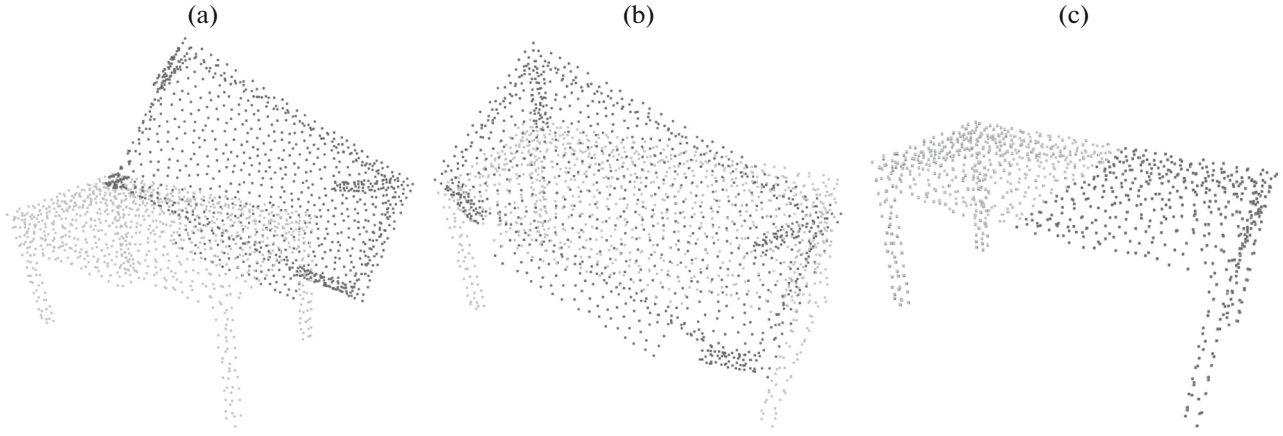

Fig. 6. (a) Point clouds: (gray dots) source and (black dots) target; (b) results of the CNN registration; and (c) result of the CNN + PtP registration.

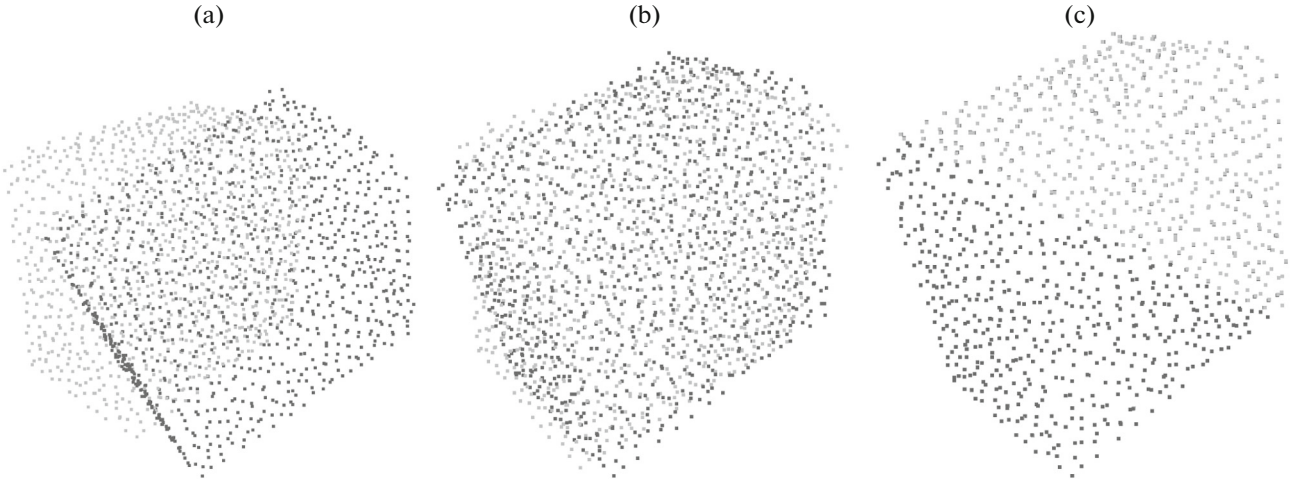

Fig. 7. (a) Point clouds: (gray dots) source and (black dots) target; (b) results of the CNN registration; and (c) result of the CNN + PtP registration.

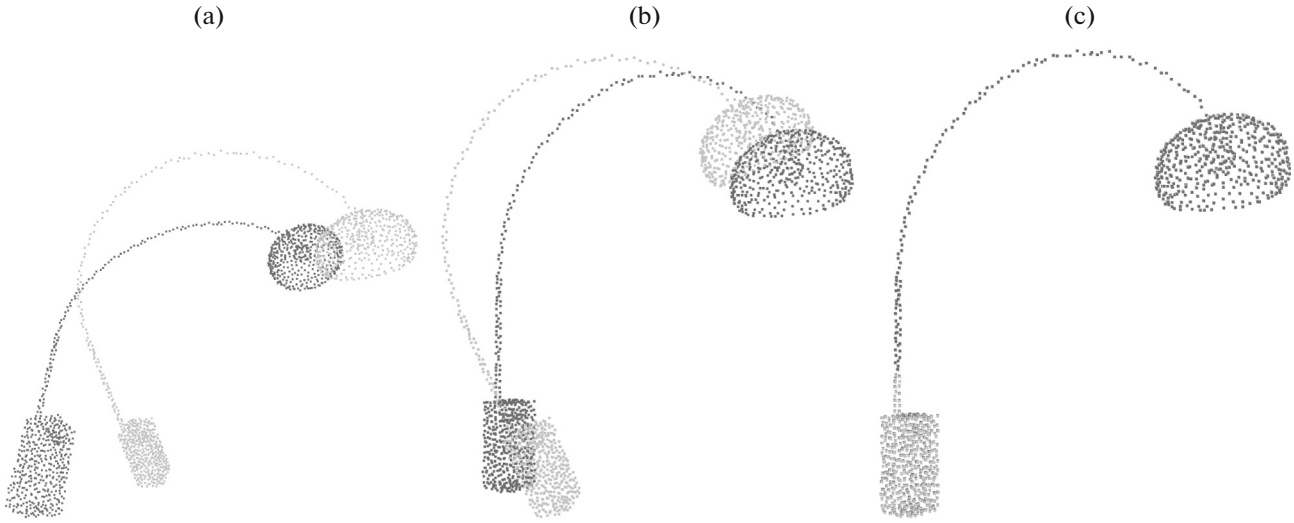

Fig. 8. (a) Point clouds: (gray dots) source and (black dots) target; (b) results of the CNN registration; and (c) result of the CNN + PtP registration.

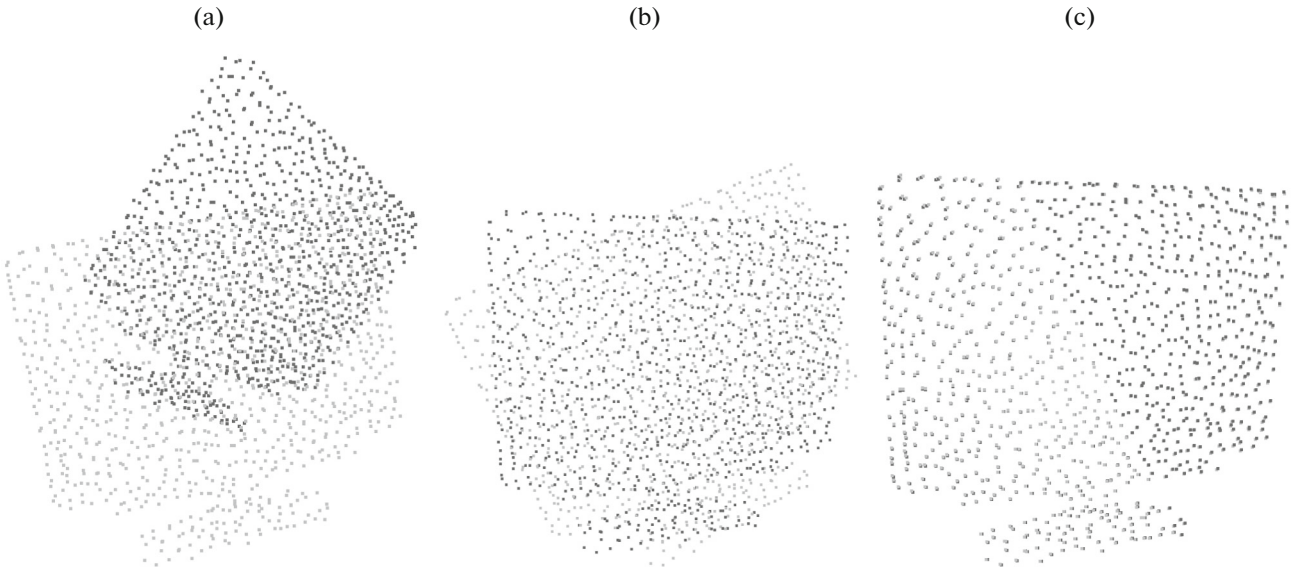

Fig. 9. (a) Point clouds: (gray dots) source and (black dots) target; (b) results of the CNN registration; and (c) result of the CNN + PtP registration.

position of the point clouds after CNN, and Fig. 8c shows the position of the point clouds after CNN + PtP. Table 8 presents the errors of the considered algorithms with respect to the L_1 and L_2 norms.

Figure 9a shows the initial position of source point cloud P and target point cloud Q, Fig. 9b shows the position of the point clouds after CNN, and Fig. 9c shows the position of the point clouds after CNN +

Table 6. Accuracy of calculation of rotation and parallel translation with respect to norms L_1 and L_2

	R_{CNN}	T_{CNN}	$R_{CNN+ICP}$	$T_{CNN+ICP}$
L_1	2.275989	0.249078	0.000003	0.000001
L_2	0.933218	0.160645	0.000001	0.000000

Table 7. Accuracy of calculation of rotation and parallel translation with respect to norms L_1 and L_2

	R_{CNN}	T_{CNN}	$R_{CNN+ICP}$	$T_{CNN+ICP}$
L_1	0.942749	0.160958	0.000002	0.000001
L_2	0.367263	0.109005	0.000001	0.000000

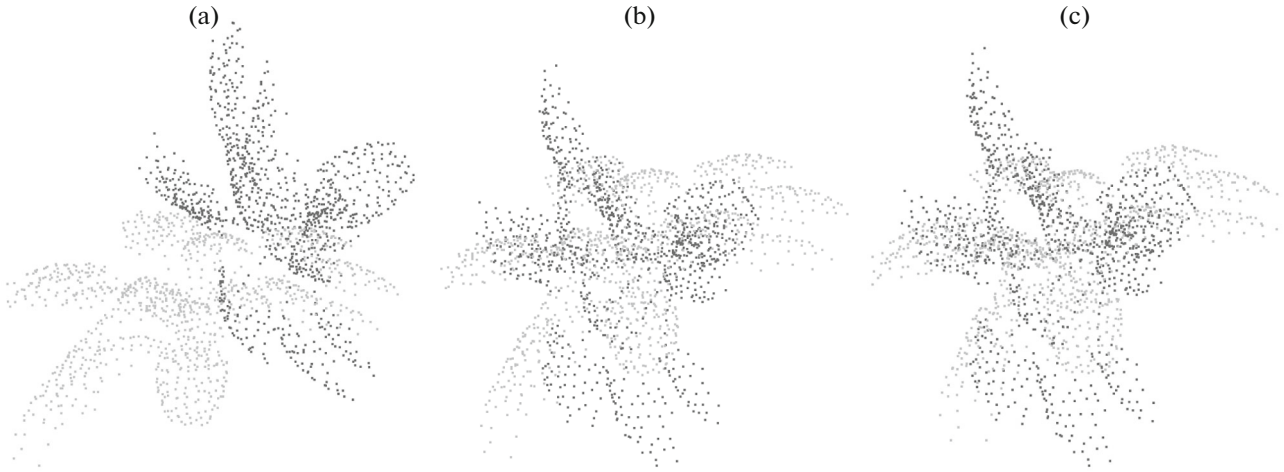

Fig. 10. (a) Point clouds: (gray dots) source and (black dots) target; (b) results of the CNN registration; and (c) result of the CNN + PtP registration.

PtP. Table 9 presents the errors of the considered algorithms with respect to the L_1 and L_2 norms.

Figure 10a shows the initial position of source point cloud P and target point cloud Q, Fig. 10b shows the position of the point clouds after CNN, and Fig. 10c shows the position of the point clouds after CNN + PtP. Table 10 presents the errors of the considered algorithms with respect to the L_1 and L_2 norms.

We use large rotation angles in experiment 10, and the algorithm yields a false transformation, since the proposed CNN is trained on relatively small rotation angles.

Table 8. Accuracy of calculation of rotation and parallel translation with respect to norms L_1 and L_2

	R_{CNN}	T_{CNN}	$R_{CNN+ICP}$	$T_{CNN+ICP}$
L_1	1.808907	0.174372	0.000001	0.000000
L_2	0.689345	0.111230	0.000000	0.000000

Table 9. Accuracy of calculation of rotation and parallel translation with respect to norms L_1 and L_2

	R_{CNN}	T_{CNN}	$R_{CNN+\text{ICP}}$	$T_{CNN+ICP}$
L_1	1.198202	0.270670	0.000001	0.000000
L_2	0.541937	0.167172	0.000001	0.000000

Table 10. Accuracy of calculation of rotation and parallel translation with respect to norms L_1 and L_2

	R_{CNN}	T_{CNN}	$R_{CNN+\text{ICP}}$	$T_{CNN+ICP}$
L_1	7.443286	1.626208	7.383439	1.719010
L_2	2.778001	1.059065	2.812945	1.072411

CONCLUSIONS

We proposed a simple neural network algorithm for registering point clouds in 3D space. The proposed neural network is based on soft alignment of source and target point clouds and employs a weighted point-to-point ICP function to estimate an orthogonal transformation that matches the point clouds. Soft alignment of point clouds using a probability matrix describes the semantic similarity of points. The proposed network together with the two-point ICP shows good results in the reconstruction of a 3D model. The efficiency of the proposed algorithm is evaluated in the ModelNet40 database.

FUNDING

This work was supported by the Russian Science Foundation, project no. 22-19-20071.

CONFLICT OF INTEREST

The authors of this work declare that they have no conflicts of interest.

REFERENCES

1. S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, and A. Fitzgibbon, "KinectFusion: Real-time 3d reconstruction and interaction using a moving depth camera," in *Proc. 24th Annu. ACM Symp. on User Interface Software and Technology, Santa Barbara, Calif., 2011* (Association for Computing Machinery, New York, 2011), pp. 559–568. https://doi.org/10.1145/2047196.2047270
2. M. Eldefrawy, S. A. King, and M. Starek, "Partial scene reconstruction for close range photogrammetry using deep learning pipeline for region masking," Remote Sens. **14**, 3199 (2022). https://doi.org/10.3390/rs14133199

3. Zh. Zhang, Yu. Dai, and J. Sun, "Deep learning based point cloud registration: An overview," Virtual Reality Intell. Hardware **2**, 222–246 (2020). https://doi.org/10.1016/j.vrih.2020.05.002
4. K. Chen, B. T. Lopez, A.-A. Agha-Mohammadi, and A. Mehta, "Direct LiDAR Odometry: Fast localization with dense point clouds," IEEE Rob. Autom. Lett. **7**, 2000–2007 (2022). https://doi.org/10.1109/lra.2022.3142739
5. Yu. Zheng, Yu. Li, Sh. Yang, and H. Lu, "Global-PBNet: A novel point cloud registration for autonomous driving," IEEE Trans. Intell. Transp. Syst. **23**, 22312–22319 (2022). https://doi.org/10.1109/tits.2022.3153133
6. P. J. Besl and N. D. Mckay, "A method for registration of 3-D shapes," IEEE Trans. Pattern Anal. Mach. Intell. **14**, 239–256 (1992). https://doi.org/10.1109/34.121791
7. Ya. Chen and G. Medioni, "Object modelling by registration of multiple range images," Image Vision Comput. **10**, 145–155 (1992). https://doi.org/10.1016/0262-8856(92)90066-c
8. B. K. P. Horn, H. M. Hilden, and Sh. Negahdaripour, "Closed-form solution of absolute orientation using orthonormal matrices," J. Opt. Soc. Am. **5**, 1127–1135 (1988). https://doi.org/10.1364/josaa.5.001127
9. S. Umeyama, "Least-squares estimation of transformation parameters between two point patterns," IEEE Trans. Pattern Anal. Mach. Intell. **13**, 376–380 (1991). https://doi.org/10.1109/34.88573
10. A. Makovetskii, S. Voronin, V. Kober, and A. Voronin, "An efficient algorithm for non-rigid object registration," Komp'yuternaya Opt. **44**, 67–73 (2020). https://doi.org/10.18287/2412-6179-co-586
11. A. Makovetskii, S. Voronin, V. Kober, and A. Voronin, "A non-iterative method for approximation of the exact solution to the point-to-plane variational problem for orthogonal transformations," Math. Methods Appl. Sci. **41**, 9218–9230 (2018). https://doi.org/10.1002/mma.5173
12. A. Makovetskii, S. Voronin, V. Kober, and A. Voronin, "A regularized point cloud registration approach for orthogonal transformations," J. Global Optim. **83**, 497–519 (2020). https://doi.org/10.1007/s10898-020-00934-8
13. A. Makovetskii, S. Voronin, V. Kober, and A. Voronin, "Point cloud registration based on multiparameter functional," Mathematics **9**, 2589 (2021). https://doi.org/10.3390/math9202589
14. A. Makovetskii, S. Voronin, V. Kober, and A. Voronin, "Coarse point cloud registration based on variational functionals," Mathematics **11**, 35 (2023). https://doi.org/10.3390/math11010035
15. Ya. Aoki, H. Goforth, R. A. Srivatsan, and S. Lucey, "PointNetLK: Robust & efficient point cloud registration using PointNet," in *2019 IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR), Long Beach, Calif., 2019* (IEEE, 2019), pp. 7163–7172. https://doi.org/10.1109/cvpr.2019.00733
16. V. Sarode, X. Li, H. Goforth, Y. Aoki, R. Srivatsan, S. Lucey, and H. Choset, "PCRNet: Point cloud registration network using pointnet encoding," arXiv Preprint (2019). https://doi.org/10.48550/arXiv.1908.07906
17. H. Xu, Sh. Liu, G. Wang, G. Liu, and B. Zeng, "OMNet: Learning overlapping mask for partial-to-partial point cloud registration," in *2021 IEEE/CVF Int. Conf. on Computer Vision (ICCV), Montreal, 2021* (IEEE, 2021), pp. 3132–3141. https://doi.org/10.1109/iccv48922.2021.00312
18. Yu. Wang and J. Solomon, "Deep closest point: Learning representations for point cloud registration," in *2019 IEEE/CVF Int. Conf. on Computer Vision (ICCV), Seoul, 2019* (IEEE, 2019), pp. 3522–3531. https://doi.org/10.1109/iccv.2019.00362
19. Y. Wang and J. Solomon, "PRNet: Self-supervised learning for partial-to-partial registration," in *Proc. 33rd Int. Conf. on Neural Information Processing Systems, Vancouver, 2019*, Ed. by H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, and E. B. Box (Curran Associates, Red Hook, N.Y., 2019), p. 791.
20. Z. J. Yew and G. H. Lee, "RPM-Net: Robust point matching using learned features," in *2020 IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR), Seattle, Wash., 2020* (IEEE, 2020), pp. 11824–11833. https://doi.org/10.1109/cvpr42600.2020.01184
21. J. Li, Ch. Zhang, Z. Xu, H. Zhou, and Ch. Zhang, "Iterative distance-aware similarity matrix convolution with mutual-supervised point elimination for efficient point cloud registration," in *Computer Vision – ECCV 2020*, Ed. by A. Vedaldi, H. Bischof, T. Brox, and J. M. Frahm, Lecture Notes in Computer Science, Vol. 12369 (Springer, Cham, 2020), pp. 378–394. https://doi.org/10.1007/978-3-030-58586-0_23
22. W. Lu, Ya. Zhou, G. Wan, Sh. Hou, and Sh. S. Song, "L3-Net: Towards learning based LiDAR localization for autonomous driving," in *2019 IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR), Long Beach, Calif., 2019* (IEEE, 2019), pp. 6382–6391. https://doi.org/10.1109/cvpr.2019.00655
23. S. Voronin, A. Vasilyev, V. Kober, A. Makovetskii, A. Voronin, and D. Zhernov, "Neural network for 3D point clouds alignment," Proc. SPIE **12226**, 122261H (2022). https://doi.org/10.1117/12.2633565
24. S. Voronin, A. Vasilyev, V. Kober, A. Makovetskii, A. Voronin, and D. Zhernov, "Deep neural network for incongruent point clouds registration," Proc. SPIE **12674**, 409–419 (2023). https://doi.org/10.1117/12.2677782
25. C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep hierarchical feature learning on point sets in a metric space," Proc. 31st Int. Conf. Neural Inf. Process. Syst., 5105–5114 (2017).
26. Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3D ShapeNets: A deep representation for volumetric shapes," in *2015 IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), Boston, 2015* (IEEE, 2015), pp. 1912–1920. https://doi.org/10.1109/CVPR.2015.7298801

Translated by A. Chikishev

Publisher's Note. Pleiades Publishing remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.