Информационные процессы, Том 25, № 1, 2025, стр. 25–34.

ТЕОРИЯ И МЕТОДЫ ОБРАБОТКИ ИНФОРМАЦИИ

Нейросетевой алгоритм регистрации трехмерных облаков точек, использующий виртуальные точки

А.Ю. Маковецкий[*], В.И. Кобер[,***], С.М. Воронин[*], А.В. Воронин[*], В.Н. Карнаухов[**], М.Г. Мозеров[**]**

[*] *Челябинский государственный университет, Челябинск, Россия,*
[**] *Институт проблем передачи информации, Российская академия наук, Москва, Россия,*
[***] *Центр научных исследований и высшего образования, Энсенада, 22860, Мексика*

Поступила в редколлегию 10.02.2025 г. Принята 23.04.2025 г.

Аннотация—Регистрация трехмерных облаков точек имеет большое значение в робототехнике и компьютерном зрении, поскольку позволяет найти геометрическое преобразование для выравнивания пары облаков точек с неизвестными соответствиями. В последние годы методы глубокого обучения стали играть важнейшую роль в области компьютерного зрения. Существенным элементом регистрации облаков точек является оценка соответствия между облаками точек. Основная идея заключается в установлении соответствий с помощью многомерных дескрипторов для каждой точки. В этой статье мы описываем нейросетевой алгоритм для регистрации неконгруэнтных облаков точек. Предложенный алгоритм использует виртуальные точки и неточное соответствие между облаками, и он частично основан на нейронной сети PointNet++. Результаты компьютерного моделирования иллюстрируют эффективность предлагаемого метода.

КЛЮЧЕВЫЕ СЛОВА: нейронная сеть, облако точек, регистрация, глубокое обучение, дескриптор.

DOI: 10.53921/18195822_2025_25_1_25

1. ВВЕДЕНИЕ

Регистрация 3D-облаков точек играет решающую роль в робототехнике и компьютерном зрении, позволяя вычислить геометрическое преобразование для выравнивания двух облаков точек с неизвестными соответствиями. Этот процесс широко применяется в различных областях, включая реконструкцию 3D-сцены, локализацию и автономное вождение [1]–[5]. Традиционный итеративный метод ближайшей точки (ICP) долгое время был стандартом трехмерной регистрации [6]–[17], включающим чередование этапов поиска соответствий точек и применение жестких преобразований. Однако ICP опирается на хорошее начальное выравнивание и часто сходится к локальным минимумам. В последние годы появление моделей глубокого обучения произвело революцию в компьютерном зрении, что привело к появлению более быстрых и надежных алгоритмов регистрации облаков точек по сравнению с классическими методами. Алгоритмы на основе глубокого обучения, такие как Deep Closest Point (DCP) [18], PRNet [19], RPMNet [20] и IDAM [21] используют нейронные сети для установления соответствий с помощью многомерных дескрипторов. Нейронная сеть DCP, использующая Dynamic Graph CNN (DGCNN), извлекает локальные особенности из облаков точек с использованием мягкого соответствия. PRNet расширяет DCP, включая модуль обнаружения ключевых точек для частичной регистрации. RPMNet использует неточное соответствие из гибридных объектов, полученных как из пространственных координат, так и из локальной геометрии. IDAM объединяет признаки и евклидову информацию в соответствующую матрицу, используя двухступенчатую обучаемую технику для исключения точек при регистрации.

Предлагаемый метод использует стратегию описания признаков с двумя ветвями, включающую как информацию о местоположении, так и локальные ортогонально-инвариантные признаки. Эта стратегия направлена на формирование многомерных дескрипторов точек из облаков, отличаясь от существующих методов, которые сталкиваются с проблемами, связанными с большими поворотами и значительными различиями в координатах облаков. В этой статье мы предлагаем архитектуру нейронной сети, которая использует мягкое сопоставление между точками исходных и целевых облаков и взвешенный функционал ICP точка-точка вместо стандартного функционала ICP точка-точка [6]–[26]. Вес пары точек выбирается по вероятности того, что пара принадлежит общим подмножествам облаков. Веса точек вычисляются на основе дескрипторов точек, сформированных предлагаемой архитектурой. Нейросетевой алгоритм использует элементы сети PointNet++ [27]. Сеть была обучена на базе данных ModelNrt40 [28]. Компьютерное моделирование иллюстрирует эффективность предлагаемого алгоритма.

Статья организована следующим образом. В разделе 1 приведена формулировка задачи и описаны алгоритмы ее решения. Разработанный нейросетевой алгоритм описан в разделе 2. В разделе 3 представлены результаты компьютерного моделирования. Раздел 4 содержит заключение.

2. НЕЙРОСЕТЕВОЙ АЛГОРИТМ

Разработанный подход предполагает, что архитектура сети вычисляет дескрипторы всех точек в облаках точек, определяет соответствие между точками и вычисляет преобразование из группы SE(3) для выравнивания двух заданных облаков точек. Обозначим через $P = p_1, \ldots, p_{s_p}$ и $Q = q_1, \ldots, q_{s_q}$ исходное и целевое облака точек, $p_i, q_j \in \mathbb{R}^3, i = 1, \ldots, s_p, j = 1, \ldots, s_q$. Рассмотрим точку $p_i \in P$ и ее окрестность в P с радиусом r. В окрестности рассматривается K точек. Мы игнорируем дополнительные точки, если окрестность содержит более K точек, и добавляем копии точки из окрестности, если окрестность содержит менее K точек. Таким образом, мы сопоставляем множество, содержащее K элементов, точке p_i и делаем это для всех точек P и Q. Дескрипторы всех точек в P вычисляются следующим образом. Обозначим через B количество элементов в батче.

Первый слой. Исходный тензор имеет размер $(B, 3, s_p)$. Находим окрестности всех точек s_p и получаем тензор $(B, s_p, K, 3)$, после перестановки размерностей тензор имеет размер $(B, 3, K, s_p)$. Используем двумерную свертку с параметрами
$in\_channels = 3, out\_channels = 64, kernel\_size = 7, stride = 1, padding = 3$, т. е. тензор после свертки имеет размер $(B, 64, K, s_p)$. Используется функция активации ReLu и нормализация батча. После свертки находим максимум в тензоре по размерности с индексом 2 (размерность имеет размер K) и получаем тензор $(B, 64, s_p)$.

Второй слой. Исходный тензор имеет размер $(B, 64, s_p)$. Находим окрестности всех точек s_p и получаем тензор $(B, sp, K, 64)$, после перестановки размерностей тензор имеет размер $(B, 64, K, s_p)$ Используем двумерную свертку с параметрами
$in\_channels = 64, out\_channels = 128, kernel\_size = 5, stride = 1, padding = 2$, т. е. тензор после свертки имеет размер $(B, 128, K, s_p)$. Используется функция активации ReLu и нормализация батча. После свертки находим максимум в тензоре по размерности с индексом 2 (размерность имеет размер K) и получаем тензор $(B, 128, s_p)$.

Третий слой. Исходный тензор имеет размер $(B, 128, s_p)$. Находим окрестности всех точек s_p и получаем тензор $(B, s_p, K, 128)$, после перестановки размерностей тензор имеет размер $(B, 128, K, s_p)$. Используем двумерную свертку с параметрами
$in\_channels = 128, out\_channels = 256, kernel\_size = 3, stride = 1, padding = 1$, т. е. тензор после свертки имеет размер $(B, 256, K, s_p)$. Используется функция активации ReLu и нормали-

зация батча. После свертки находим максимум в тензоре по размерности с индексом 2 (размерность имеет размер K) и получаем тензор $(B, 256, s_p)$.

Четвертый слой. Исходный тензор имеет размер $(B, 256, s_p)$. Находим окрестности всех точек s_p и получаем тензор $(B, s_p, K, 256)$, после перестановки размерностей тензор имеет размер $(B, 256, K, s_p)$. Используем двумерную свертку с параметрами $in\_channels = 256, out\_channels = 256, kernel\_size = 1, stride = 1, padding = 0$, т. е. тензор после свертки имеет размер $(B, 256, K, s_p)$. Используется функция активации ReLu и нормализация батча. После свертки находим максимум в тензоре по размерности с индексом 2 (размерность имеет размер K) и получаем тензор $(B, 256, s_p)$.

Пятый слой. Исходный тензор имеет размер $(B, 256, s_p)$. Находим окрестности всех точек s_p и получаем тензор $(B, s_p, K, 256)$, после перестановки размерностей тензор имеет размер $(B, 256, K, s_p)$. Используем двумерную свертку с параметрами $in\_channels = 256, out\_channels = 512, kernel\_size = 1, stride = 1, padding = 0$, т.е. тензор после свертки имеет размер $(B, 512, K, s_p)$. Используется функция активации ReLu и нормализация батча. После свертки находим максимум в тензоре по размерности с индексом 2 (размерность имеет размер K) и получаем тензор $(B, 512, s_p)$.

Вектор размером 512 считается дескриптором точки в P. Мы вычисляем дескрипторы точек в облаке точек Q аналогичным образом. Когда все дескрипторы уже сделаны, мы генерируем матрицу $\mathbf{W}$ размером $s_p \times s_q$. Элемент матрицы W_{ij} является вероятностью соответствия между точками p_i и $q_j \quad i = 1, \ldots, s_p, j = 1, \ldots, s_q$. Матрица $\mathbf{W}$ используется для формирования виртуального облака точек $V = \left\{v_1, \ldots, v_{s_p}\right\}$ следующим образом:

$$v_i = \sum_{j=1}^{s_q} W_{ij} q_j, \tag{1}$$

где $i = 1, \ldots, s_p$. Предполагается, что точка p_i соответствует точке $v_i, \ i = 1, \ldots, s_p$. Таким образом, устанавливается мягкое соответствие между облаками точек P и Q. Отметим, что

$$\sum_{j=1}^{s_q} W_{ij} = 1, \ i = 1, \ldots, s\_p. \tag{2}$$

Обозначим через C_p центр масс облака P

$$C_p = \frac{1}{s_p} \sum_{i=1}^{s_p} p_i \tag{3}$$

и через P' центроид облака P

$$p_i^{'} = p_i - C_p \tag{4}$$

где $i = 1, \ldots, s_p$. Обозначим через $\mathbf{P}$’ матрицу центроида P’

$$\mathbf{P}' = \begin{pmatrix} p_{11}^{'} \ldots p_{s_p 1}^{'} \\ p_{12}^{'} \ldots p_{s_p 2}^{'} \\ p_{13}^{'} \ldots p_{s_p 3}^{'} \end{pmatrix} \tag{5}$$

где $p_i^{'} = \left(p_{\mathrm{i}1}^{'} \ p_{i2}^{'} \ p_{\mathrm{i}3}^{'}\right)^t, \ i = 1, \ldots, s_p$

Аналогично обозначим через $\mathbf{P}, \mathbf{V}$ и $\mathbf{V}^{'}$ матрицы для облаков P, V и центроида V' соответственно.

$$\mathbf{P} = \begin{pmatrix} p_{11} \dots p_{s_p 1} \\ p_{12} \dots p_{s_p 2} \\ p_{13} \dots p_{s_p 3} \end{pmatrix}, \ \mathbf{V} = \begin{pmatrix} v_{11} \dots v_{s_p 1} \\ v_{12} \dots v_{s_p 2} \\ v_{13} \dots v_{s_p 3} \end{pmatrix}, \mathbf{V}^{'} = \begin{pmatrix} v'_{11} \dots v'_{s_p 1} \\ v'_{12} \dots v'_{s_p 3} \\ v'_{13} \dots v'_{s_p 3} \end{pmatrix} \tag{6}$$

Через $\mathbf{T}$ обозначим матрицу, состоящую из s_p одинаковых столбцов:

$$\mathbf{T} = \begin{pmatrix} T_1 \dots T_1 \\ T_2 \dots T_2 \\ T_3 \dots T_3 \end{pmatrix} \tag{7}$$

где $T = (T_1 \ T_2 \ T_3)^t$ вектор параллельного переноса. Рассмотрим следующий функционал:

$$J(R,T) = ||R\ P + T\ - V||_{L_2}{}^2 \tag{8}$$

и условную вариационную задачу.

$$(R_*, T_*) = arg \min_{R,T} J(R,T) \tag{9}$$

при условии, что $R \in SO(3)$. Вариационная задача (9) сводится к решению задачи:

$$R_* = arg \min_{R} J'(R). \tag{10}$$

где $R \in SO(3)$ и функционал J' задается следующим образом

$$J'(R) = \left|\left|R\ P'\ - V'\right|\right|_{L_2}{}^2. \tag{11}$$

Вариационная задача (10) может быть сведена к условной вариационной задаче

$$R_* = arg \max_{R} \ \left\langle RP', \mathbf{V}' \right\rangle \tag{12}$$

где $R \in SO(3)$. Рассмотрим следующую матрицу $\mathbf{M}$

$$\mathbf{M} = V'\big(\ P'\big)^t. \tag{13}$$

Матрицу $\mathbf{M}$ можно разложить по сингулярным значениям в матричное произведение

$$\mathbf{M} = USV^t. \tag{14}$$

где $\mathbf{U}$ и $\mathbf{V}^t$ — ортогональные матрицы, а $\mathbf{S}$ — диагональная матрица. Отметим, что PyTorch поддерживает оператор SVD и его обратное распространение ошибки для него. Решение вариационной задачи (12) принимает вид

$$R_* = \begin{cases} UV^t, if \det(U) \det(V) = 1 \\ U diag(1,1,-1) V^t, if \det(U) \det(V) = -1 \end{cases} \tag{15}$$

и вектор переноса может быть вычислен как

$$T_* = \sum_{j=1}^{s_p} (v_j - R_* p_j) \tag{16}$$

Количество обучаемых весов полученной сети равно 1416320. Функция ошибки представляет собой взвешенную сумму квадратичной ошибки между оценочной и истинной матрицами преобразования (поворота и переноса). Алгоритм Адам с параметром скорости обучения равен 0,001. Размер батча $B = 10$, а количество соседей $K = 32$. Сеть обучалась по парам облаков точек из части базы данных ModelNet40 [28]. Сеть обучалась в течение 4 эпох. Значение функции потерь $loss = 0.6403$ до обучения, $loss = 0.1676$ после 1-й эпохи, $loss = 0.0779$ после 2-й эпохи, $loss = 0.0832$ после 3-й эпохи, $loss = 0.0516$ после 4-й эпохи.

3. КОМПЬЮТЕРНОЕ МОДЕЛИРОВАНИЕ

Предложенная нейронная сеть применялась к набору данных ModelNet40 [28]. Набор данных содержит 12311 моделей с триангуляцией, принадлежащих 40 классам. Обучающая часть базы данных состоит из 9843 облаков точек, а тестовая часть — из 2468 облаков точек. Для каждой модели равномерно выбирается 1024 точки из граней триангуляции.

Используется только информация о координатах точек. Для каждого облака $P = p_1, \ldots, p_{1024}$ случайно выбираются углы поворота относительно осей координат x, y, z в диапазоне $[-\frac{\pi}{4}; \frac{\pi}{4}]$ и случайно выбираются координаты вектора переноса с его координатами в диапазоне $[-0,5; 0,5]$. Матрица (в однородных координатах) заданного преобразования обозначается как M_{true}. Применяем матрицу $\mathbf{M}_{true}$ к матрице $\mathbf{P}$, которая связана с облаком P, и получаем матрицу $\mathbf{Q}$, которая связана с облаком $Q = \{q_1, \ldots, q_{1024}\}$:

$$\mathbf{Q} = M_{true} P, \tag{17}$$

Мы используем предложенную нейронную сеть (CNN) в качестве алгоритма грубого выравнивания для облаков точек и ICP точка-точка (PtP) в качестве алгоритма уточнения. Точность алгоритмов оценивается следующими выражениями:

$$E_R = ||R_{est} - R_{true}||_{L_1}\ , \quad E_T = ||T_{est} - T_{true}||_{L_1}, \tag{18}$$

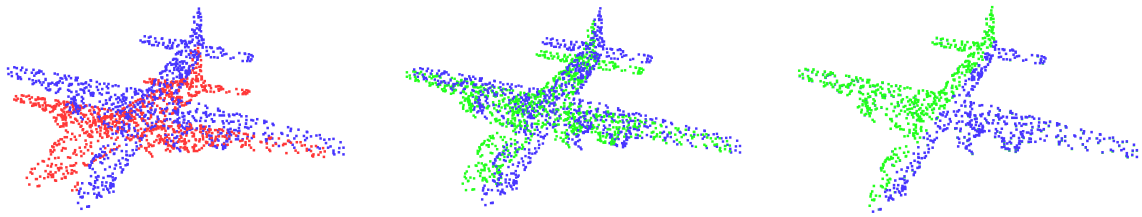
где (R_{est}, T_{est}) — поворот и перемещение, восстановленные алгоритмом, (R_{true}, T_{true}) — истинные поворот и перемещение.

В экспериментальной части статьи сравниваются предлагаемая нейронная сеть (CNN), CNN+PtP, известная нейронная сеть DCP [18] и DCP+PtP для четырех наборов данных, которые условно назовем: “самолет”, “кресло”, “бокал” и “стул”. Алгоритмы CNN+PtP и DCP+PtP содержат этап уточнения алгоритмом PtP после грубой регистрации нейронной сетью.

Набор данных “самолет”:
На рисунке Рис.1(а) показано начальное положение исходного облака точек P (красный/серый цвет) и целевого облака точек Q (синий/черный цвет), на рисунке Рис.1(б) показано положение облаков точек после предлагаемой CNN, на рисунке Рис.1(в) показано положение облаков точек после CNN+PtP.

В таблице 1 показаны ошибки поворота и трансляции для алгоритмов CNN, CNN+PtP, DCP и DCP+PtP для набора данных “самолет”.

Набор данных “кресло”:
На рисунке Рис.2(а) показано начальное положение исходного облака точек P (красный/серый цвет) и целевого облака точек Q (синий/черный цвет), на рисунке Рис.2(б) показано положение

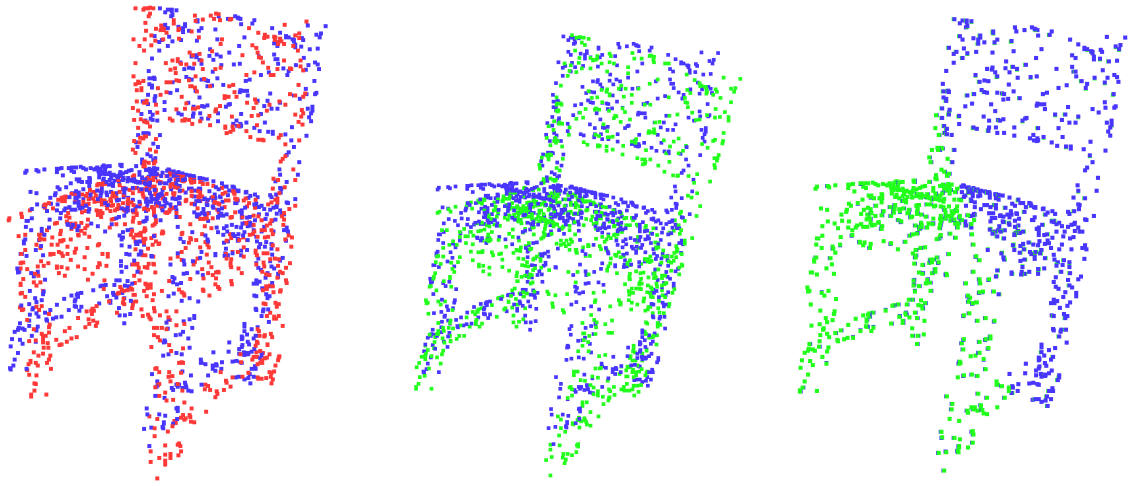


(а) (б) (в)

Рис. 1. (а) — Исходное облако точек (красные/светло-серые точки) и целевое облако точек (синие/черные точки); (б) —результат регистрации CNN (зелёные/темно-серые точки); (в) — результат регистрации CNN+PtP.

Таблица 1. Точность алгоритмов CNN, CNN+PtP, DCP и DCP+PtP для набора данных “самолет”.

	CNN	**CNN+PtP**	**DCP**	**DCP+PtP**
$\mathbf{E_R}$	0.0357	$< 10^{-4}$	0.0075	$< 10^{-4}$
$\mathbf{E_T}$	0.0033	$< 10^{-4}$	$< 10^{-4}$	$< 10^{-4}$



(а) (б) (в)

Рис. 2. (а) — Исходное облако точек (красные/светло-серые точки) и целевое облако точек (синие/черные точки); (б) — результат регистрации CNN (зелёные/темно-серые точки); (в) — результат регистрации CNN+PtP.

облаков точек после предлагаемой CNN, на рисунке Рис.2(в) показано положение облаков точек после CNN+PtP.

В таблице 2 показаны ошибки поворота и трансляции для алгоритмов CNN, CNN+PtP, DCP и DCP+PtP для набора данных “кресло”.

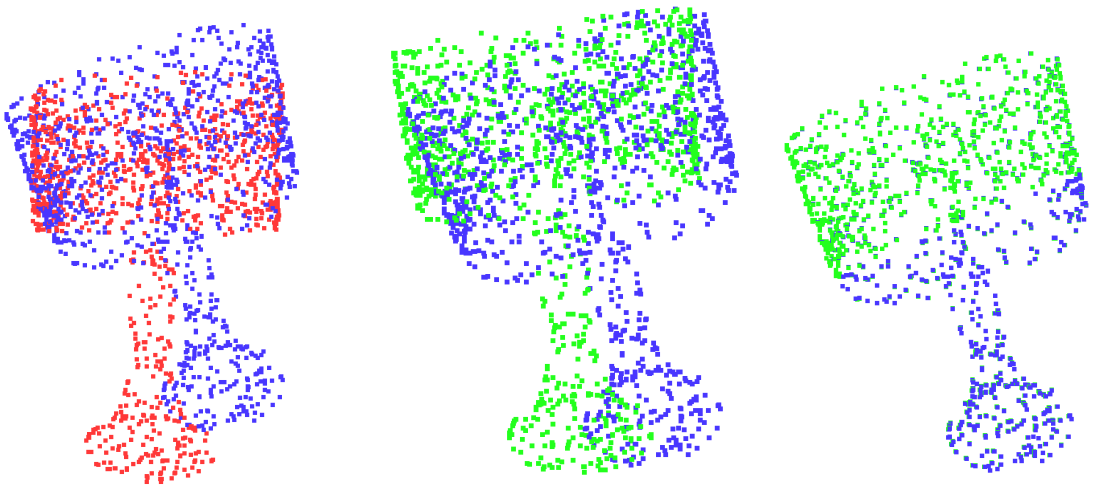
Набор данных “бокал”:

На рисунке Рис.3(а) показано начальное положение исходного облака точек P (красный/серый цвет) и целевого облака точек Q (синий/черный цвет), на рисунке Рис.3(б) показано положение облаков точек после предлагаемой CNN, на рисунке Рис.3(в) показано положение облаков точек после CNN+PtP.

В таблице 3 показаны ошибки поворота и трансляции для алгоритмов CNN, CNN+PtP, DCP и DCP+PtP для набора данных “бокал”.

Таблица 2. Точность алгоритмов CNN, CNN+PtP, DCP и DCP+PtP для набора данных “кресло”.

	CNN	**CNN+PtP**	**DCP**	**DCP+PtP**
$\mathbf{E_R}$	0.0622	$< 10^{-4}$	0.0157	$< 10^{-4}$
$\mathbf{E_T}$	0.0149	$< 10^{-4}$	0.0002	$< 10^{-4}$



(а) (б) (в)

Рис. 3. (а) — Исходное облако точек (красные/светло-серые точки) и целевое облако точек (синие/черные точки); (б) — результат регистрации CNN (зелёные/темно-серые точки); (в) — результат регистрации CNN+PtP.

Набор данных “стул”:

На рисунке Рис.4(а) показано начальное положение исходного облака точек P (красный/серый цвет) и целевого облака точек Q (синий/черный цвет), на рисунке Рис.4(б) показано положение облаков точек после предлагаемой CNN, на рисунке Рис.4(в) показано положение облаков точек после CNN+PtP.

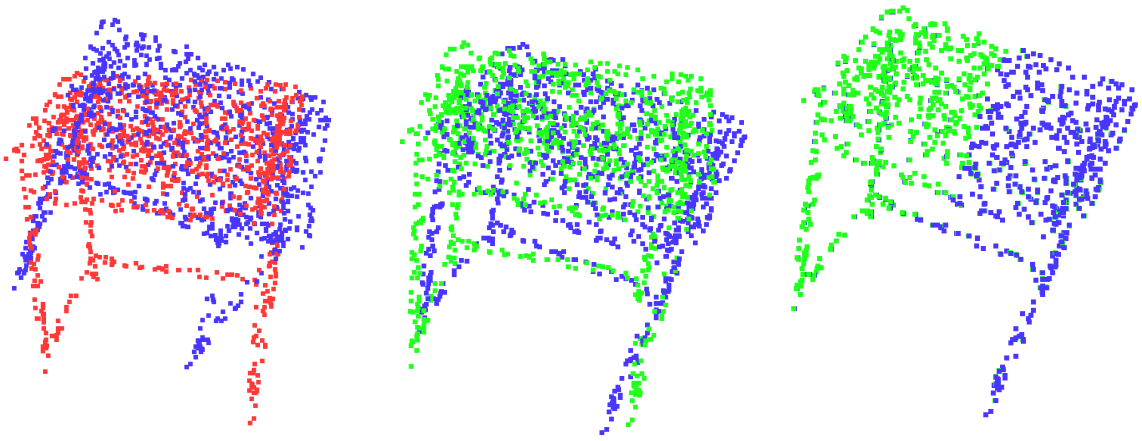
В таблице 4 показаны ошибки поворота и трансляции для алгоритмов CNN, CNN+PtP, DCP и DCP+PtP для набора данных “стул”.

4. ЗАКЛЮЧЕНИЕ

В этой статье предложен нейросетевой алгоритм для регистрации облаков точек в трехмерном пространстве. Предлагаемая нейронная сеть основана на использовании виртуальных точек и мягкого сопоставлении между исходными и целевыми облаками точек. Сеть использует взвешенную функцию ICP точка-точка для оценки ортогонального преобразования, которое выравнивает облака точек. Мягкое сопоставление облаков точек с использованием матрицы вероятности описывает семантическое сходство точек. Предлагаемая сеть вместе с ICP алгоритмом показывает хорошие результаты в задаче реконструкции 3D-модели. Эффективность предлагаемого алгоритма сравнивается с известной нейронной сетью для регистрации облаков точек DCP и оценивается на наборе данных ModelNet40.

Таблица 3. Точность алгоритмов CNN, CNN+PtP, DCP и DCP+PtP для набора данных “бокал”.

	CNN	**CNN+PtP**	**DCP**	**DCP+PtP**
$\mathbf{E_R}$	0.1568	$< 10^{-4}$	0.0428	$< 10^{-4}$
$\mathbf{E_T}$	0.0187	$< 10^{-4}$	0.0011	$< 10^{-4}$



(а) (б) (в)

Рис. 4. (а) — Исходное облако точек (красные/светло-серые точки) и целевое облако точек (синие/черные точки); (б) — результат регистрации CNN (зелёные/темно-серые точки); (в) — результат регистрации CNN+PtP.

Таблица 4. Точность алгоритмов CNN, CNN+PtP, DCP и DCP+PtP для набора данных “стул”.

	CNN	**CNN+PtP**	**DCP**	**DCP+PtP**
$\mathbf{E_R}$	0.1462	$< 10^{-4}$	0.0377	$< 10^{-4}$
$\mathbf{E_T}$	0.0233	$< 10^{-4}$	0.0002	$< 10^{-4}$

СПИСОК ЛИТЕРАТУРЫ

1. Izadi, S., Kim, D., Hilliges, O., Molyneaux, D., Newcombe, R., Kohli, P., Shotton, J., Hodges, S., Freeman, D., Davison, A., et al., “Kinectfusion: Real-time 3d reconstruction and interaction using a moving depth camera,” Proc. of the ACM Symposium on Utilizer Interface Software and Technology, 559–568 (2011).

2. Eldefrawy, M., King, S. and Starek, M., “Partial Scene Reconstruction for Close Range Photogrammetry Using Deep Learning Pipeline for Region Masking,” Remote Sens., 14, 3199 (2022).

3. Zhang, Z., Dai, Y. and Sun, J., “Deep learning based point cloud registration: An overview,” Virtual Real. Intell. Hardw., 2, 222–246, (2020).

4. Chen, K., Lopez, B., Agha-Mohammadi, A. and Mehta, A., “Direct lidar odometry: Fast localization with dense point clouds,” IEEE Robot. Autom. Lett., 7, 2000–2007 (2022).

5. Zheng, Y., Li, Y., Yang, S. and Lu, H., “Global-PBNet: A novel point cloud registration for autonomous driving,” IEEE Trans. Intell. Transp. Syst., 23, 22312–22319 (2022).

6. Besl, P. and McKay, N., “A method for registration of 3-D shapes,” IEEE Transactions of Pattern Analysis and Machine Intelligence, 2, 239-256 (1992).

7. Chen, Y. and Medioni, G., “Object modeling by registration of multiple range images,” Image and Vision Computing, 10, 145-155, (1992).

8. Horn, B., Hilden, H. and Negahdaripour, S., "Closed-form Solution of Absolute Orientation Using Orthonormal Matrices,"Journal of the Optical Society of America, Series A, 5, 7, 1127-1135 (1988).

9. Umeyama, S. “Least-squares estimation of transformation parameters between two point patterns,” IEEE-TPAMI, 13(4), 376-380 (1991).

10. Makovetskii, A., Voronin, S., Kober, V. and Voronin, A., “An efficient algorithm for non-rigid object registration,” Computer Optics, 44, 67-73 (2020).

11. Makovetskii, A., Voronin, S., Kober, V. and Voronin, A., “A non-iterative method for approximation of the exact solution to the point-to-plane variational problem for orthogonal transformations,” Mathematical Methods in the Applied Sciences, 41(18), 9218-9230 (2018).

12. Makovetskii, A., Voronin, S., Kober, V. and Voronin, A., “A regularized point cloud registration approach for orthogonal transformations,” Journal of Global Optimization (2020).

13. Makovetskii, A., Voronin, S., Kober, V. and Voronin, A., “Point Cloud Registration Based on Multiparameter Functional,” Mathematics, 9, 2589 (2021).

14. Makovetskii, A., Voronin, S., Kober, V. and Voronin, A., “Coarse Point Cloud Registration Based on Variational Functional,” Mathematics, 11, 35 (2023).

15. Aoki, Y., Goforth, H., Srivatsan, R. and Lucey, S., “Pointnetlk: Robust & efficient point cloud registration using pointnet,” Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 7163–7172 (2019).

16. Sarode, V., Li, X., Goforth, H., Aoki, Y., Srivatsan, R., Lucey, S. and Choset, H., “Pcrnet: Point cloud registration network using pointnet encoding,” arXiv:1908.07906 (2019).

17. Xu, H., Liu, S., Wang, G., Liu, G. and Zeng, B., “Omnet: Learning overlapping mask for partial-to-partial point cloud registration,” Proc. of the IEEE/CVF International Conference on Computer Vision, 3132–3141 (2021).

18. Wang, Y. and Solomon, J., “Deep closest point: learning representations for point cloud registration,” International Conference on Computer Vision (ICCV), IEEE, 3522–3531 (2019).

19. Wang, Y. and Solomon, J., “Prnet: Self-supervised learning for partial-to-partial registration,” arXiv:1910.12240, (2019).

20. Yew, Z. and Lee, G., “Rpm-net: Robust point matching using learned features,” Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 11824–11833 (2020).

21. Li, J., Zhang, C., Xu, Z., Zhou, H. and Zhang, C., “Iterative distance-aware similarity matrix convolution with mutual-supervised point elimination for efficient point cloud registration,” Proc. of the 16th European Conference on Computer Vision, 378–394 (2020).

22. Lu, W., Zhou, Y., Wan, G., Hou, S. and Song. S., “L3-Net: Towards learning based LiDAR localization for autonomous driving,” Proc. of the IEEE Conference on Computer Vision and Pattern Recognition, (2019).

23. Voronin, S., Vasilyev, A., Kober, V., Makovetskii, A., Voronin, A. and Zhernov, D., "Neural network for 3D point clouds alignment,"Proc. SPIE 12226, Applications of Digital Image Processing XLV, 122261H, (2022).

24. Voronin, S., Vasilyev, A., Kober, V., Makovetskii, A., Voronin, A. and Zhernov, D., “Deep neural network for incongruent point clouds registration,” Proc. SPIE 12674, Applications of Digital Image Processing XLVI, 409-419, (2023). [

25. Makovetskii, A., Kober, V., Voronin, S., Voronin, A., Karnaukhov, V., Mozerov, M. Registration of point clouds in 3d space using soft alignment // Journal of Communications Technology and Electronics., V. 69 (2024).

26. A. Makovetskii, V. Kober, S. Voronin, A. Voronin, V. Karnaukhov, M. Mozerov, “Global refinement algorithm for 3d scene reconstruction from a sequence of point clouds,” J. Commun. Technol. Electron. V. 68 (2023).

27. Qi, C. R., Yi, L., Su, H. and Guibas, L. J., “Pointnet++: Deep hierarchical feature learning on point sets in a metric space,” Advances in neural information processing systems, 30, (2017).

28. Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X. and Xiao, J., “3D Shapenets: A Deep Representation for Volumetric Shapes, “Proc. CVPR, 1912-1920, (2015).

Neural network algorithm to 3D point cloud registration using virtual points

A. Makovetskii, V. Kober, S. Voronin, A. Voronin, V. Karnaukhov, M. Mozerov.

Three-dimensional point cloud registration is of great importance in robotics and computer vision because it allows finding a geometric transformation to align a pair of point clouds with unknown correspondences. In recent years, deep learning methods have become a major player in the field of computer vision. An essential element of point cloud registration is the correspondence estimation between point clouds. The basic idea is to establish correspondence using multidimensional descriptors for each point. In this paper, we describe a neural network algorithm for registering incongruent point clouds. The proposed algorithm uses virtual points and fuzzy correspondence between clouds, and it is partially based on the PointNet++ neural network. Computer simulation results illustrate the effectiveness of the proposed method.

KEYWORDS: neural network, point cloud, registration, deep learning, descriptor.