УДК 629.053
DOI: 10.17586/0021-3454-2021-64-7-559-566

АЛГОРИТМ ПОВЫШЕНИЯ ПРОСТРАНСТВЕННОЙ ПЛОТНОСТИ ЛИДАРНОГО ОБЛАКА ТОЧЕК ДЛЯ РЕШЕНИЯ ЗАДАЧ АВТОНОМНОГО ВОЖДЕНИЯ АВТОМОБИЛЯ

А. А. Старобыховская[1], О. Ю. Лашманов[2], В. В. Коротаев[1]

[1] *Университет ИТМО, 197101, Санкт-Петербург, Россия*

[2] *ООО „Эррайвал Рус“, 197229, Санкт-Петербург, Россия*
E-mail: o.lashmanov@gmail.com

Для решения задачи повышения плотности точек лидарных облаков предложены несколько алгоритмов, в том числе базирующихся на нейросетях. Лидарные облака высокой плотности могут повышать точность алгоритмов распознавания окружающей сцены и локализации. Качество алгоритмов оценивается следующими метриками: количество ложных точек, средняя погрешность, среднеквадратическая погрешность. В отличие от существующих, предложенные алгоритмы направлены на определение не только позиции дополнительных точек, но и коэффициента их отражательной способности. Приведены результаты экспериментов, показано, что наименьшей погрешностью обладает нейросетевой алгоритм без использования нормализации и сигмоидального взвешивания функции потерь. Наименьшее число ложных точек обеспечивает нейросетевой алгоритм с добавлением нормализации.

Ключевые слова: *повышение плотности облака точек, лидар, нейронные сети, нормали, Unet, беспилотные автомобили*

Введение. В современном мире многие компании по изготовлению автомобилей и автомобильных компонентов, такие, например, как Audi, BMW, Ford, стремятся к созданию собственной версии транспортного средства с функциональностью автономного или полуавтономного вождения. В большинстве комплектаций сенсоров для таких автомобилей используются лидары [1, 2]. Лидар позволяет получать информацию об окружающей сцене в виде облака точек (отображающего положение объектов в трехмерном пространстве относительно сенсора, а также величину отраженного от поверхности объектов сигнала), слабо зависящего от освещения сцены. Такие облака точек используются для решения задач распознавания объектов [3, 4] и локализации [5, 6]. Повысить точность алгоритмов возможно путем увеличения плотности облаков точек [3]. Однако использование в массовом производстве сенсоров, позволяющих получать данные с большой плотностью, затруднено по причине высокой стоимости (например, стоимость 32-лучевого лидара Outser OS-1 начинается от 3500 долл. [3], 64-лучевого — от 12000 долл. [4]).

Наибольшее число доступных наборов лидарных данных создано с использованием 32-лучевого лидара Velodyne HDL-32 [5, 6], однако подавляющее число разработанных алгоритмов и исследований предполагают работу с данными 64-лучевого лидара Velodyne 64 [7], при этом на неплотных данных алгоритмы оказываются не робастными.

В настоящей статье представлены результаты разработки и анализ алгоритмов предварительной обработки облака точек повышенной плотности для лидара беспилотного автомобиля.

Обзор решений по повышению плотности лидарных облаков точек. Для повышения плотности лидарных облаков точек существуют методы, не связанные со способом формирования облака. По результатам анализа публикаций было выделено исследование [8], основанное на использовании модели PU-Net, повышающей плотность неструктурированных облаков точек. В этой работе используется допущение, что распределение точек на входе

неравномерно, а на выходе равномерно. На первом шаге алгоритма выделяются локальные патчи (фрагменты) в облаке точек путем аппроксимации заранее заданным набором поверхностей, затем для каждого патча вычисляются проекции в векторное пространство и выделяются признаки. После этого производится увеличение числа признаков для каждого патча и осуществляется обратная реконструкция трехмерных координат. Таким образом, алгоритм работает только в масштабах одного патча и не производит глобальной оптимизации, при этом алгоритм не поддерживает свойства точек (яркость, цвет и др.). В рамках настоящего исследования были разработаны альтернативные алгоритмы повышения плотности облаков точек, в которых учитывается коэффициент отражательной способности (I) точек, а не только положение точки в пространстве.

Алгоритм повышения плотности лидарных облаков точек. Главная идея разработанного подхода — представление облака точек в плотном виде и увеличение вертикального разрешения панорамного изображения сцены, где значение пиксела соответствует дистанции до точки. Из панорамного изображения с увеличенным вертикальным разрешением производится восстановление облака точек. Такой подход приемлем, так как лидарные облака обладают свойством уменьшения плотности точек с увеличением расстояния относительно сенсора, а угловое распределение точек остается равномерным. Каждой точке облака соответствует координата в трехмерном пространстве (X, Y, Z) декартовых координат и коэффициент отражательной способности.

Для построения панорамного изображения производится переход из декартовой системы координат [X, Y, Z] в сферическую [ρ, φ, θ]. Формулы для вычисления сферических координат имеют следующий вид:

$$\rho = \sqrt{x^2 + y^2 + z^2}\ ; \tag{1}$$

$$\varphi = \operatorname{arctg}\left(\frac{y}{x}\right); \tag{2}$$

$$\theta = \arccos\left(\frac{z}{\rho}\right), \tag{3}$$

где x, y, z — координаты облака точек в системе координат лидара; ось X направлена вперед, Y — влево, Z — вверх.

Представим полученные данные в виде регулярной сетки путем дискретизации пространств значений φ и θ. В проведенных экспериментах размер сетки составил 1400 по горизонтали и 64 по вертикали. Координаты точки в сетке вычисляются по следующим выражениям:

$$x_p = \frac{\varphi w}{\alpha};$$

$$y_p = \frac{\theta h}{\beta},$$

где w, h — ширина и высота панорамного изображения; α, β — горизонтальный и вертикальный углы разрешения лидара соответственно.

В данном случае панорамизацию можно рассматривать как процедуру проекции трехмерного пространства в двумерное. Каждая ячейка содержит информацию об исходных признаках точек (X, Y, Z и остальные). Такое панорамизированное облако точек, представленное в виде сетки, обозначим как $M(t)$, где t — индекс облака точек в исходном массиве данных; $M_{ij}(t)$ — ячейка, расположенная в i-м столбце и j-й строке сетки.

Одной из особенностей, в частности, лидара HDL-64 [9] является неравномерность распределения лучей по вертикальной оси. Панорамизация такого типа лидара производится способом, описанным выше, независимо для верхнего и нижнего участков облака точек с разным шагом дискретизации. Далее обработанные участки „склеиваются“ в одну сетку. Сле-

дующий шаг к получению панорамы — применение разрабатываемых алгоритмов увеличения плотности данных (см. далее). После этого осуществляется обратное преобразование в трехмерное представление для получения итогового облака точек.

Для обучения и валидации алгоритмов повышения разрешения панорамного изображения использовались прореженные данные 64-лучевого лидара из набора данных KITTI [10]. Для преобразования данных к допустимому формату учитывалась разница углов обзора 64- и 32-лучевых лидаров (32-лучевой сенсор имеет более широкое поле зрения). По этой причине входные облака „обрезались“ до 21 слоя на входе, что позволяло получать 42-слойное облако на выходе.

Модуль расширения панорамного представления облака точек состоит из двух шагов: вычисление значений признаков точек, которыми будет дополнено исходное облако, построчное объединение предсказанных значений с исходными. Далее рассмотрим три предлагаемых алгоритма по предсказанию (восстановлению) дополнительных значений признаков.

Алгоритм восстановления по среднему значению. Алгоритм вычисления дополнительных значений по среднему основан на вычислении среднего значения признака (p_i) по вертикали между соседними (ближайшими) точками. При использовании такого примитивного подхода возможна генерация точек в воздухе. Такой эффект возникает, если ячейки с большой разницей значений ρ расположены рядом. Для удобства оценивания было введено понятие ложной точки. Ложная точка — это точка, для которой в эталонном панорамном изображении нет соответствующей.

Модифицированная версия данного алгоритма учитывает не только координаты точки в пространстве, но и вектор $M_{ij}(t)$ дополнительных признаков, который рассчитывается следующим образом:

$$M_{i,j}(t) = \frac{v_t \otimes v_l + v_b \otimes v_r}{2}; \tag{4}$$

$$v_t = M_{i-1,j}(X,Y,Z) - M_{i,j}(X,Y,Z);$$

$$v_l = M_{i,j-1}(X,Y,Z) - M_{i,j}(X,Y,Z);$$

$$v_b = M_{i+1,j}(X,Y,Z) - M_{i,j}(X,Y,Z),$$

$$v_r = M_{i,j+1}(X,Y,Z) - M_{i,j}(X,Y,Z).$$

В отличие от исходного алгоритма, в данном случае добавляется проверка угла между векторами (4) для двух соседних ячеек. Если данный угол превышает заданный порог, то такие ячейки не учитываются при расчете среднего. Для расчета вектора (4) для каждой ячейки выбираются две ближайшие ненулевые ячейки снизу и справа. Таким образом, формируются два вектора направления, произведение которых определяет вектор (4). В результате каждое значение признака будет иметь по два вектора, на базе которых вычисляется угол между ними.

Алгоритм восстановления скользящим усредняющим окном. Данный метод основан на обработке локальных данных (попадающих в скользящее окно). При сканировании скользящим окном вычисляется среднее количество (F) ячеек, тем самым определяются дополнительные значения признаков. Расчет среднего возможен двумя способами: через среднее арифметическое или через среднее взвешенное. Восстановленное значение (F) обозначается как p_i:

$$p_i = \frac{1}{n}\sum_{j=1}^{n} v_{i,j}; \qquad p_i = \sum_{j=1}^{n} v_{i,j} w_j,$$

где n — количество ячеек, попавших в окно; $v_{i,j}$ — среднее значение признаков точек в окне, w_j — вес полученного значения для j-го окна.

Для расчета весов предложен следующий алгоритм:

```
addstep_x = ws % 2
weights_x = zero_matrix(ws, ws)
weights_x[0 .. ws // 2] = [ws // 2 .. 0]
weights_x[(ws // 2 + addstep_x) .. ws] = [1 .. ws // 2]
weights_x, weights_y = meshgrid(weights_x + addstep_x, weights_x +
addstep_x)
weights = 1 / (weights_x + weights_y - 1)
return weights ^ power
```

В приведенном коде введены следующие обозначения: `ws` — размер матрицы, `power` — коэффициент значимости, `meshgrid` — функция, которая создает N-мерные массивы координат для векторизованных N-мерных оценок скалярных или векторных полей для N-мерной сетки по заданным одномерным массивам координат $x_1, x_2, \ldots, x_n$.

Нейросетевой алгоритм восстановления. Также был разработан алгоритм, основанный на сверточной нейронной сети архитектуры типа U-Net [11], так как она показывает высокие результаты в задачах повышения разрешения изображений [12]. Разработанная модель состоит из слоев кодирования и декодирования: см. рис. 1, здесь каждый блок представляет собой тензор, рядом с тензором указана его размерность; стрелками показаны преобразования, осуществляемые над данными. Каждый слой образован одним блоком Res Block [10, 13, 14] и двумя сверточными слоями. Размер входного и выходного изображений идентичен.

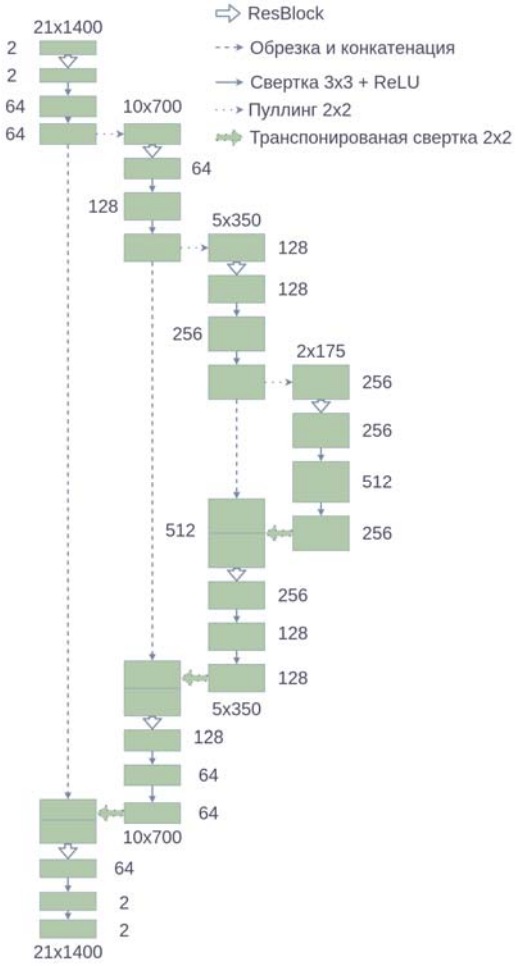

Рис. 1

Важный этап работы с моделями — их обучение. На основании идеи модификации U-net [15—17] из стандартной архитектуры были удалены слои пакетной нормализации. Оптимизируемая величина состоит из суммы значений дисперсий (ρ и I) и количества ложных

точек. Дисперсия рассчитывается только для ячеек, имеющих валидное значение p_i в обучающем наборе данных. Итоговая формула функции потерь (оптимизируемого параметра) имеет следующий вид:

$$\mathrm{L} = C_{dd} D_\rho + C_{di} D_I + C_{fpd} FP_\rho + C_{fpi} FP_I ,$$

где D — дисперсия, C — весовые значения, FP — количество ложных точек.

Также была добавлена нормализация с помощью сигмоидальной функции вместо константных весовых множителей.

Эксперименты и полученные результаты. В рамках данного исследования работы обучение модели и тестирование алгоритмов проводились на наборе данных KITTI. Качество работы алгоритма оценивалось по следующим критериям: количество ложных точек, максимальная погрешность определения ρ, средняя погрешность, среднеквадратическое отклонение (СКО). В результате было проведено 7 экспериментов на базе существующих алгоритмов и разработанных алгоритмов с различными модификациями. Полученные результаты приведены в таблице.

Алгоритм	Средняя погрешность (δ_ρ), м	СКО ($\Delta\delta_\rho$), м^2	Количество ложных точек, шт.
Нейросетевая модель	0,398	1,498	5834
Нейросетевая модель с нормализацией значений признаков	0,546	1,692	5531
Нейросетевая модель с использованием сигмоиды при расчете ошибки	0,548	1,729	5548
Нейросетевая модель с нормализацией значений признаков и использованием сигмоиды при расчете ошибки	0,686	1,836	5588
Восстановление по среднему значению	0,681	2,234	9364
Восстановление скользящим усредняющим окном	0,680	2,054	10368

Согласно проведенным экспериментам наименьшее число ложных точек показала нейросетевая модель с добавлением блока нормализации значений признаков, при этом наименьшие значения погрешности для признака демонстрирует разработанная нейросетевая модель без нормализирующего блока.

На рис. 2 представлены результаты повышения плотности облаков точек (вид сверху; начало координат находится в точке установки лидара, ось X соответствует направлению движения автомобиля, ось Y направлена влево, ось Z — вверх). Цветом закодирована погрешность δ определения ρ: синий — $\delta_\rho < 0{,}05$ м, зеленый — $\delta_\rho < 0{,}10$ м, желтый — $\delta_\rho < 0{,}15$ м, красный — $\delta_\rho > 0{,}15$ м.

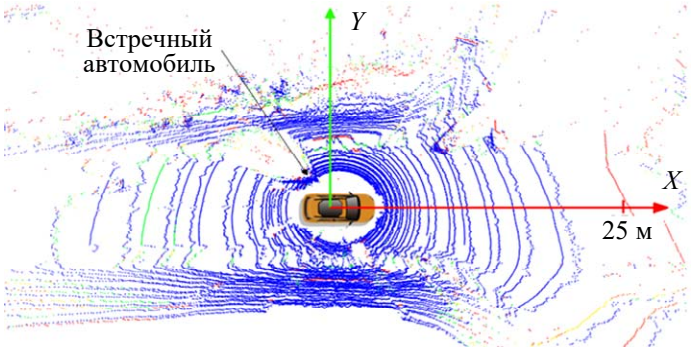

Рис. 2

Заключение. Предложены алгоритмы повышения плотности облаков точек для лидарных данных. Согласно результатам экспериментов, из предложенных алгоритмов лучшим по совокупности оценок является нейросетевой алгоритм на основе U-Net архитектуры. Данный алгоритм позволяет увеличить плотность лидарного облака точек в 2 раза по вертикальной оси. Разработанное решение потенциально позволит расширить объем данных для обучения, валидации и тестирования алгоритмов детектирования объектов и локализации в сфере автономного вождения автомобиля.

СПИСОК ЛИТЕРАТУРЫ

1. *Caesar H., Bankiti V., Lang A. H., Vora S., Liong V. E., Xu Q., Krishnan A., Pan Y., Baldan G., Beijbom O.* nuScenes: A multimodal dataset for autonomous driving // Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR). 2020. Vol. 1. P. 11621—11631.
2. *Sun P., Kretzschmar H., Dotiwalla X., Chouard A., Patnaik V., Tsui P., Guo J., Zhou Y., Chai Y., Caine B., Vasudevan V.* Scalability in perception for autonomous driving: Waymo open dataset // IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR). 2020. Vol. 1. pp. 2446—2454.
3. *Yang B., Luo W., Urtasun R.* PIXOR: Real-time 3D object detection from point clouds // IEEE Computer Society. 2018. Vol. 1. P. 7652—7660.
4. *Liang M.* et al. Deep continuous fusion for multi-sensor 3d object detection // Proc. of the European Conf. on Computer Vision (ECCV). 2018. Vol. 1. P. 641—656.
5. *Rozenberszki D., Majdik A. L.* LOL: Lidar-only odometry and localization in 3D point cloud maps // IEEE Intern. Conf. on Robotics and Automation (ICRA). 2020. P. 4379—4385.
6. *Chen S., Liu B., Feng C., Vallespi-Gonzalez C., Wellington C.* 3D point cloud processing and learning for autonomous driving: Impacting map creation, localization, and perception // IEEE Signal Processing Magazine. 2021. Vol. 38, N 1. P. 68—86.
7. *Bai X., Luo Z., Zhou L., Fu H., Quan L., Tai C. L.* D3Feat: Joint learning of dense detection and description of 3D local features // IEEE Computer Society. 2020. Vol. 1. P. 6358—6366.
8. Ouster, Inc. Mid-Range High-Resolution Imaging Lidar [Электронный ресурс]: <https://ouster.com/products/os1-lidar-sensor/>, 18.7.2019.
9. *Alsadik B.* Ideal angular orientation of selected 64-channel // Remote Sensing. 2020. Vol. 12, N 510.
10. *Geiger A., Lenz P., Stiller C., Urtasun R.* Vision meets robotics: the KITTI dataset // Intern. Journal of Robotics Research. 2013. Vol. 32. P. 1231—1237.
11. *Carballo A., Lambert J., Cano A. M., Wong D., Narksri P., Kitsukawa Y., Takeuchi E., Kato S., Takeda K.* LIBRE: The multiple 3D LiDAR Dataset // IEEE Intelligent Vehicles Symp. 2020. Vol. 4. P. 1094—1101.
12. *Casas S., Gulino C., Liao R., Urtasun R.* SPAGNN: Spatially-aware graph neural networks // IEEE Intern. Conf. on Robotics and Automation, Paris, France. 2020. P. 9491—9497.
13. *He K., Zhang X., Ren S., Sun J.* Deep residual learning for image recognition // IEEE Computer Society. 2016. Vol. 1. P. 770—778.
14. *Wang X., Yu K., Wu S., Gu J., Liu Y., Dong C., Qiao Y., Change Loy C.* Esrgan: Enhanced super-resolution generative adversarial networks // Proc. of the European Conf. on Computer Vision (ECCV) Workshops. 2018. P. 63—79.
15. *Lequan Yu, Xianzhi Li, Chi-Wing Fu, Cohen-Or D., Pheng-Ann Heng.* PU-Net: Point Cloud Upsampling Network // IEEE Conf. on Computer Vision and Pattern Recognition (CVPR). 2018. Vol. 1. P. 2790—2799.
16. *Ronneberger O., Fischer P., Brox T.* U-net: Convolutional Networks for Biomedical Image Segmentation. Cham: Springer, 2015. Vol. 9351. P. 234—241.
17. *Hu X., Naiel M. A., Wong A., Lamm M., Fieguth P.* RUNet: A Robust UNet architecture for image super-resolution // Computer Society. 2019. Vol. 1. P. 505—507.

Сведения об авторах

Анастасия Александровна Старобыховская — бакалавр; Университет ИТМО, Институт дизайна и урбанистики; E-mail: nana10.06@yandex.ru

Олег Юрьевич Лашманов — канд. техн. наук; ООО „Эррайвал Рус“; ст. алгоритмист; E-mail: o.lashmanov@gmail.com

Валерий Викторович Коротаев — д-р техн. наук, профессор; Университет ИТМО, факультет прикладной оптики; E-mail: korotaev_v_v@mail.ru

Поступила в редакцию
22.03. 2021 г.

Ссылка для цитирования: *Старобыховская А. А, Лашманов О. Ю., Коротаев В. В.* Алгоритм повышения пространственной плотности лидарного облака точек для решения задач автономного вождения автомобиля // Изв. вузов. Приборостроение. 2021. Т. 64, № 7. С. 559—566.

ALGORITHM FOR INCREASING THE SPATIAL DENSITY OF A LIDAR POINT CLOUD FOR SOLVING PROBLEMS OF AUTONOMOUS DRIVING

A. A. Starobyhovskaya[1], O. Yu. Lashmanov[2], V. V. Korotaev[1]

[1] *ITMO University, 197101, St. Petersburg, Russia*

[2] *Arrival Rus Ltd, 197229, St. Petersburg, Russia*

E-mail: o.lashmanov@gmail.com

Several algorithms are proposed to solve the problem of increasing the density of points of lidar clouds, including the algorithms based on neural networks. High density lidar clouds can improve the accuracy of scene recognition and localization algorithms. The algorithms quality is assessed by the following metrics: the number of false points, the average error, the root-mean-square error. In contrast to existing ones, the proposed algorithms are aimed at determining not only additional points positions, but also their reflectivity coefficients. Presented results of experiments demonstrate that the neural network algorithm without the use of normalization and sigmoidal weighting of the loss function has the smallest error. The least number of false points is provided by the neural network algorithm with an added normalization.

Keywords: increasing point cloud density, lidar, neural networks, normal, Unet, unmanned vehicle

REFERENCES

1. Caesar H., Bankiti V., Lang A.H., Vora S., Liong V.E., Xu Q., Krishnan A., Pan Y., Baldan G., Beijbom O. *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020, vol. 1, pp. 11621–11631.
2. Sun P., Kretzschmar H., Dotiwalla X., Chouard A., Patnaik V., Tsui P., Guo J., Zhou Y., Chai Y., Caine B., Vasudevan V. *IEEE/CVF Conf. on Computer Vision and Pattern Recognition* (CVPR), 2020, vol. 1, pp. 2446–2454.
3. Yang B., Luo W., Urtasun R. *IEEE Computer Society*, 2018, vol. 1, Jun, pp. 7652–7660.
4. Liang M. et al. *Proceedings of the European Conference on Computer Vision* (ECCV), 2018, vol. 1, pp. 641–656.
5. Rozenberszki D., Majdik A.L. *IEEE Intern. Conference on Robotics and Automation* (ICRA), 2020, pp. 4379–4385.
6. Chen S., Liu B., Feng C., Vallespi-Gonzalez C., Wellington C. *IEEE Signal Processing Magazine*, 2021, no. 1(38), pp. 68–86.
7. Bai X., Luo Z., Zhou L., Fu H., Quan L., Tai Cl. *IEEE Computer Society*, 2020, vol. 1, Jun, pp. 6358–6366.
8. *Ouster Inc.* Mid-Range High-Resolution Imaging Lidar, 2020, https://ouster.com/products/os1-lidar-sensor/.
9. Alsadik B. *Remote Sensing*, 2020, no. 510(12), Feb.
10. Geiger A.L., Stiller P., Urtasun C. *Intern. Journal of Robotics Research*, 2013, vol. 32, Sep., pp. 1231–1237.
11. Carballo A., Lambert J., Cano A.M., Wong D., Narksri P., Kitsukawa Y., Takeuchi E., Kato S., Takeda K. *IEEE Intelligent Vehicles Symposium*, 2020, vol. 4, pp. 1094–1101.
12. Casas S., Gulino C., Liao R., Urtasun R.. 2020 IEEE Intern. Conference on Robotics and Automation, Paris, France, 2020, pp. 9491–9497.
13. He K., Zhang X., Ren S. and Sun J. *IEEE Computer Society*, 2016, vol. 1, Jun, pp. 770–778.
14. Wang X., Yu K., Wu S., Gu J., Liu Y., Dong C., Qiao Y., Change Loy C. *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, 2018, pp. 63–79.
15. Yu L., Li X., Fu C.-W., Cohen-Or D., Heng Ph.-A. *IEEE Conf. on Computer Vision and Pattern*

Recognition (CVPR), 2018, vol. 1, Mar., pp. 2790–2799.
16. Ronneberger O., Fischer P., Brox T. *U-net: Convolutional networks for biomedical image segmentation*, Cham, Springer, 2015, vol. 9351, Oct., pp. 234–241.
17. Hu X., Naiel M.A., Wong A., Lamm M. and Fieguth P. *Computer Society*, 2019, vol. 1, Jun., pp. 505–507.

Data on authors

Anastasiya A. Starobyhovskaya — Bachelor; ITMO University, Institute of Design & Urban Studies; E-mail: nana10.06@yandex.ru

Oleg Yu. Lashmanov — PhD; Arrival Rus Ltd; Senior Algorithm Developer; E-mail: o.lashmanov@gmail.com

Valery V. Korotaev — Dr. Sci., Professor; ITMO University, Faculty of Applied Optics; E-mail: korotaev_v_v@mail.ru

For citation: Starobyhovskaya A. A., Lashmanov O. Yu., Korotaev V. V. Algorithm for increasing the spatial density of a lidar point cloud for solving problems of autonomous driving. *Journal of Instrument Engineering*. 2021. Vol. 64, N 7. P. 559—566 (in Russian).

DOI: 10.17586/0021-3454-2021-64-7-559-566