Идентификация препятствий для мобильного автономного робота методом семантической сегментации*

М.И. Кумсков[1], В.М. Буданов[,2], Р.Е. Ломовский[1], В.В. Маносин[1]

[1]Механико-математический факультет МГУ им. М.В. Ломоносова,
[2]НИИ Механики МГУ им. М.В. Ломоносова

В статье рассматривается семантическая сегментация видео для модуля восприятия беспилотного ровера с использованием датчиков LiDAR и камер. Основное внимание уделяется алгоритму U-Net и различным архитектурам предобученных энкодеров, позволяющим извлекать признаки из изображений. Обучение проводилось в два этапа с разным количеством классов, а также использовалось квантование в процессе обучения для уменьшения размера модели и повышения производительности. Представлены результаты экспериментов на датасете RUGD, оцененные по метрике IoU (Intersection over Union). Полученные результаты подтверждают, что выбор архитектуры энкодера влияет на качество сегментации, что критично для работы беспилотного ровера.

Ключевые слова: semantic segmentation, U-Net, quantization aware training, беспилотный ровер, модуль восприятия

1. Введение

Перед нашим коллективом стоит задача создания программного обеспечения для управления колесным роботом. Для получения информации об окружающей среде и состоянии самого робота предполагается использование большого количества датчиков. В частности, для визуального ориентирования, выявления препятствий, оценки состояния сцены предполагается использование LiDAR и камеры. По отдельности эти два датчика имеют ряд недостатков, которые можно нивелировать их совместным использованием. Видео с камер может дать нам полезную семантическую информацию, которую можно совместить с лидарным облаком точек для более подробного описания состояния окружающей среды. Автоматическому извлечению семантической информации из видео будет посвящена данная статья.

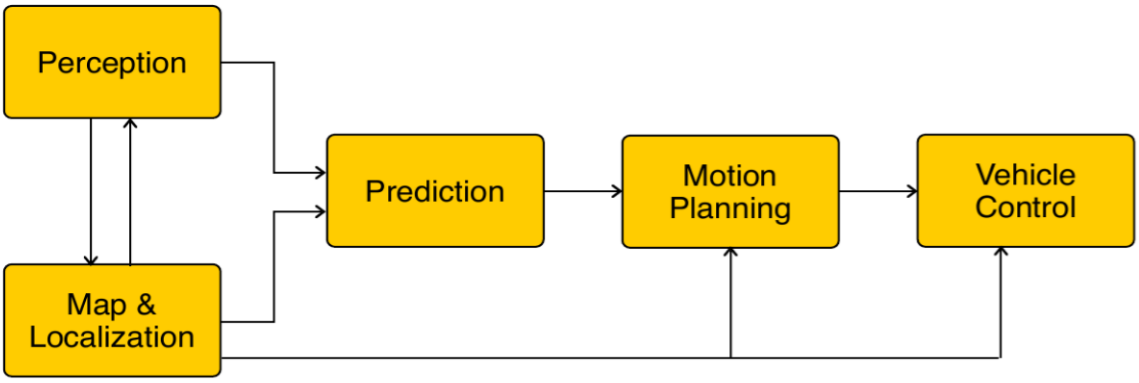
2. Постановка задачи беспилотного ровера

Подойти к задаче создания ПО для беспилотного робота довольно трудно. Предлагается рассмотреть ее как частный случай задачи **беспилотного автомобиля** (англ. self-driving-car) и использовать наработанный алгоритм, преобразовав его с учетом нашей специфики. Задача управления беспилотным автомобилем состоит из следующих блоков:

- **Localization** - корректировка положения объекта в пространстве
- **Perception** - восприятие того, что происходит вокруг робота (именно в этом блоке предполагается использовать камеры и сегментировать препятствия)
- **Prediction** - предсказания траекторий движения объектов сцены
- **Planning** - планирование траектории движения робота
- **Control** - перевод высокоуровневых инструкций в низкоуровневые (поверни колесо, ускорься, остановись и т д)
- **HD maps** - получение карты сцены с высокой точностью, (не актуально для нашей задачи, получение такой карты не предполагается)

* Работа выполнена при частичной поддержке проекта 23-Ш01-11 «Исследование интеллектуальной робототехнической системы, оснащенной ветроэнергетической установкой» НОШ МГУ «Космос»

С точки зрения обмена данными, эти блоки соотносятся следующим образом:



Статья посвящена решению задачи восприятия (Perception), остановимся на ней подробнее.

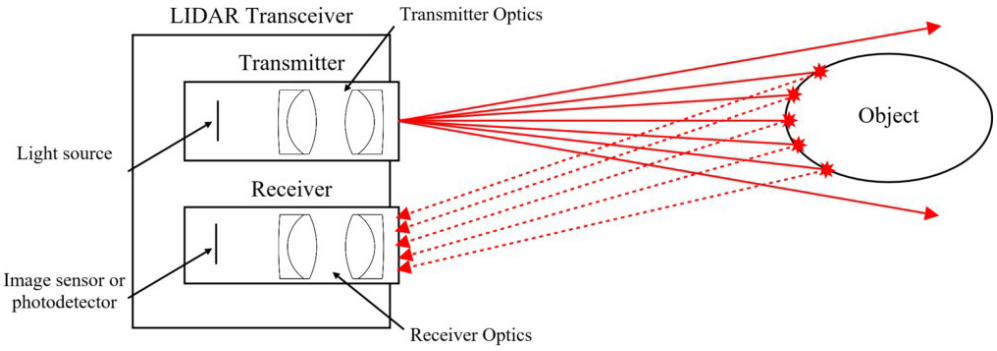
3. Модуль восприятия. LiDAR и семантическая сегментация видео

В условиях нашей задачи, предполагается использование двух средств получения визуальной информации о состоянии окружающей среды:

- Камера, установленная на передней части робота
- LiDAR

3.1. LiDAR

LiDAR (англ. Light Detection and Ranging «обнаружение и определение дальности с помощью света») — лазерный локатор, использующий технологию испускания лазером волн оптического диапазона с дальнейшей регистрацией лазерных импульсов, которые были рассеяны объектами). Примерная схема устройства LiDAR:



Не вдаваясь в схему работы данного устройства, будем считать, что с его помощью в некоторые дискретные моменты времени (порядка 10 раз в секунду) мы будем получать координаты точек в системе координат, связанной с роботом, в которых произошло отражение лидарных лучей. Такой набор точек в некоторый момент времени называется point-cloud (англ. "облако точек"). Такая информация крайне полезна в задаче оценки состояния среды, но имеет ряд недостатков, вот некоторые из них:

- Лидарные лучи не дают информацию о цвете объектов от которых они отражаются (некоторые объекты могут иметь схожую форму в виде облака точек, без информации о цвете их может быть трудно различить;

• Лидарные лучи могут отражаться от дождя, тумана или снега, что может сильно “зашумить” картину сцены.

Так или иначе, непонимание лидаром структуры препятствий - потенциально большая проблема при оценке состояния среды (например, лидар может не видеть разницы между преграждающими дорогу камнями и травой: первое препятствие придется объехать, второе можно пройти). На помощь в данном вопросе приходит камера, на основе показаний которой можно сделать вывод о семантической структуре объектов, произведя семантическую сегментацию.

3.2. Задача семантической сегментации

Семантичексая сегментация [1](англ. semantic segmentation) - присвоение каждому пикселю изображения метки определенного класса.

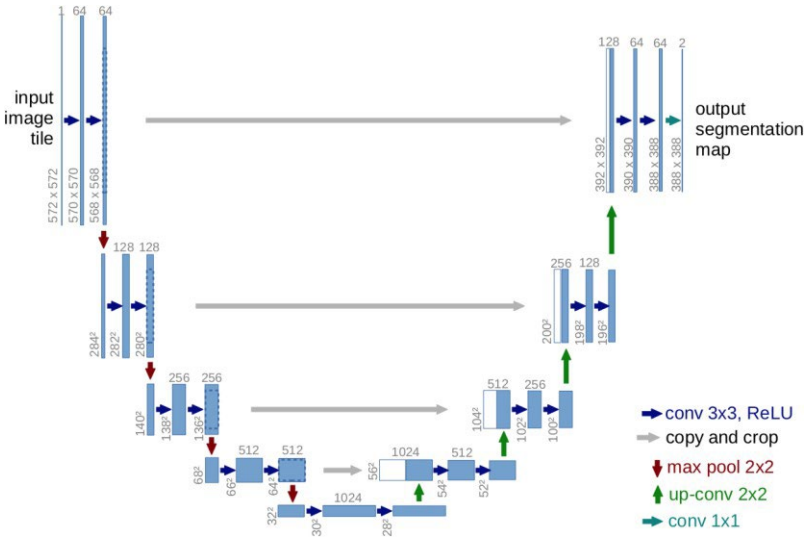
Самым современным и распространенным подходом решения задачи семантической сегментации является подход глубоких нейронных сетей различных архитектур, наиболее распространенными из которых являются:

• **Unet** [2] - архитектура, изначально созданная для сегментации медицинских изображений, нашла широкое применение во многих областях;
• **DRN-A-50** [3] - группа архитектур, описанная в статье Dilated Residual Networks;
• **HRNet** [4];

Далее будем работать с архитектурой U-Net[2], т.к. она оказалась компромиссной с точки зрения "размера"и эффективности.

3.4. U-Net

Данная архитектура представляет собой 4 последовательных уменьшения разрешения изображения в 2 раза с помощью операции **max-pooling 2 × 2**, после чего следуют 4 увеличения разрешения также в 2 раза с помощью операции **up-conv 2 × 2**, что позволяет извлекать из изображения признаки на разных уровнях разрешающей способности.



3.4. Предобученные энкодеры

Работа с изображениями в пиксельном пространстве может быть затруднительной по многим причинам, перед подачей на сегментатор стоит извлечь из него карту признаков (англ.

feature map) - тензор, в каждой ячейке которого будет закодирован какой-то предобученный признак. Для извлечения этих признаков используются предобученные нейронные сети различных архитектур, такая сеть называется **encoder** [5]. В целях улучшения качества итоговой модели или улучшения ее производительности следует перебрать несколько архитектур энкодеров, об этом подробнее в следующем разделе.

4. Способы оптимизации работы модели

Тот факт, что сегментатор должен работать автономно, накладывает некоторые ограничения на размер модели и тип данных, в котором хранятся веса. О первой проблеме следует позаботиться на этапе обучения. Если с количеством обучающих параметров U-Net[2] ничего не поделаешь, то итоговый вес модели можно уменьшить с помощью подбора архитектуры энкодера: произвести обучение для нескольких архитектур. В случае, если они покажут сопоставимое качество - выбрать ту, которая "меньше"(об этом подробнее в разделе “вычислительные эксперименты”).

Вторая проблема решается за счет применения **квантования** [6] - преобразования параметров модели из представления с плавающей точкой в представление с более низкой точностью, например, с использованием 8-битных целых чисел.

5. Вычислительные эксперименты

5.1. Обучение сегментатора с маленьким числом классов

Эксперименты проводились с использованием фреймворка **segmentation_models.pytorch** [7] на языке программирования **Python**. Для обучения использовался открытый датасет **RUGD** [8] сегментированных фотографий, снятых с ровера, размеченных на классы: dirt, tree, sand, water, rock , log (классы используемые в обучении).

Обучалась связка U-Net + encoder, с энкодерами следующих архитектур:

- **imm-mobilenetv3_small_minimal_100** [9] - 0.43 млн. обучаемых параметров
- **efficientnet-b0** [10] - 4 млн. обучаемых параметров
- **vgg11** [11] - 9 млн. обучаемых параметров
- **resnet18** [12] - 11 млн. обучаемых параметров
- **resnext50_32x4d** [13] - 22 млн. обучаемых параметров

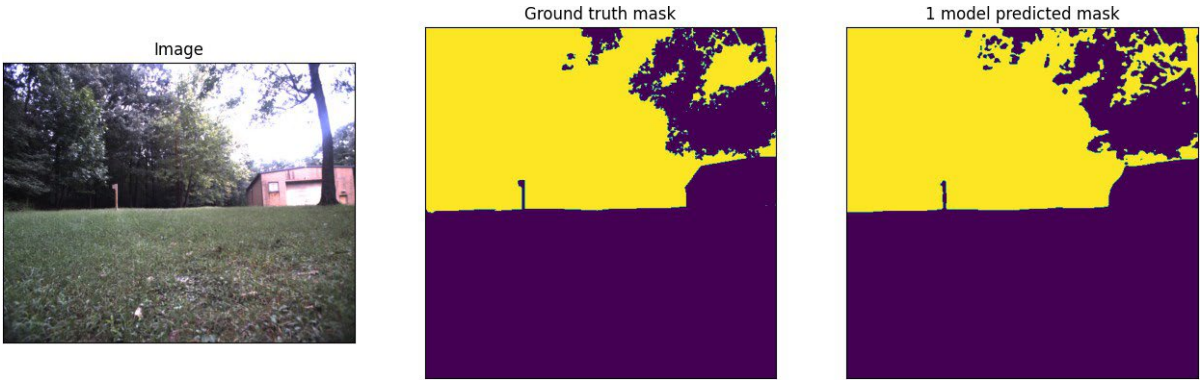
Качество модели измерялось по метрике IoU [14] (Intersection over Union), считая попиксельные пересечения и объединения ненулевых классов (0 - класс void, недетектируемый) в предсказанной и ground truth масках.

По результатам проведенных вычислительных экспериментов были получены следующие результаты:

encoder	**кол-во обучаемых параметров**	**dice_loss**	**IoU - score**
timm-mobilenetv3_sall_minimal_100	0.43 млн.	0.07892	0.8679
efficientnet-b0	4 млн.	0.06945	0.8817
vgg11	9 млн.	0.0752	0.873

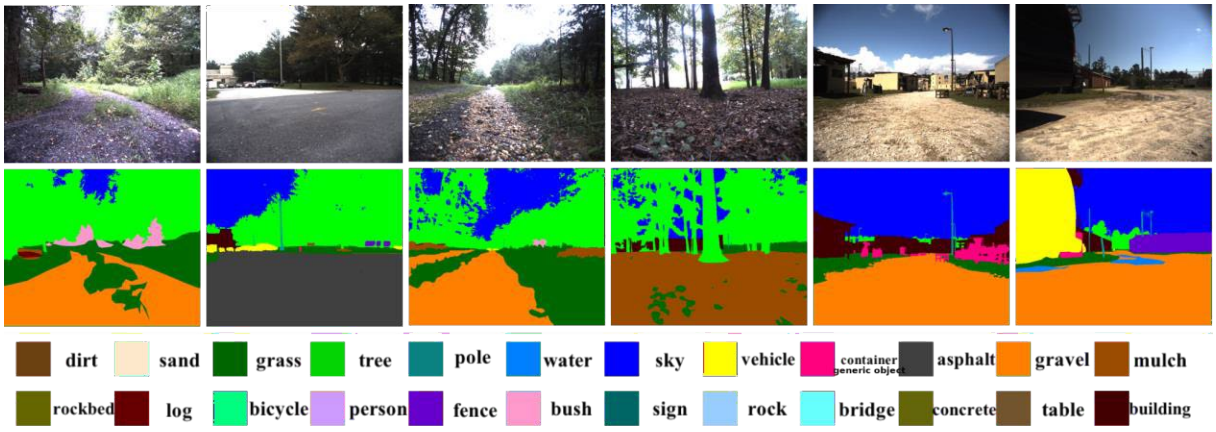
resnet18	11 млн.	0.07515	0.8734
se_resnext50_32x4d	22 млн.	0.06947	0.8818

Приведем примеры работы сегментатора с энкодером timm-mobilenetv3_sall_minimal_100 (класс ’tree’):



5.2. Обучение сегментатора с расширенным количеством классов

В дальнейших экспериментах мы пошли по направлению увеличения количества полезных классов с целью лучшей интерпретации сцены: в обучающий датасет были добавлены классы grass, pole, sky, vehicle, container/generic-object, asphalt, gravel, building, mulch, rock-bed, bicycle, person, fence, bush, sign, bridge, concrete, picnic, table.



После обучения представленных архитектур, качество модели по метрике IoU ожидаемо снизилось, но осталось приемлемым:

encoder(на 24 класса)	кол-во обучаемых параметров	IoU - score
timm-mobilenetv3_small_minimal_100	0.43 млн.	0.7815
efficientnet-b0	4 млн.	0.7771
vgg11	9 млн.	0.7911

Литература

1. Jonathan Long, Evan Shelhamer, Trevor Darrell. Fully Convolutional Networks for Semantic Segmentation. arXiv, **2015**. URL: https://arxiv.org/pdf/1411.4038
2. Olaf Ronneberger, Philipp Fischer, Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. arXiv, **2015.** URL: https://arxiv.org/abs/1505.04597
3. YSaeideh Yousefzadeh, Hamidreza Pourreza, Hamidreza Mahyar.A Triplet-loss Dilated Residual Network for High-Resolution Representation Learning in Image Retrieval. arXiv, **2023**. URL: https://arxiv.org/pdf/2303.08398
4. Shun Takagi, Li Xiong, Fumiyuki Kato, Yang Cao, Masatoshi Yoshikawa. HRNet: Differentially Private Hierarchical and Multi-Resolution Network for Human Mobility Data Synthesization. arXiv, **2024**. URL: https://arxiv.org/abs/2405.08043
5. Yann LeCun, Yoshua Bengio & Geoffrey Hinton. Deep learning. Nature. **2015**
6. Jiwei Yang, Xu Shen, Jun Xing, Xinmei Tian, Houqiang Li, Bing Deng, Jianqiang Huang, Xian-sheng Hua. Quantization Networks. **2019**
7. segmentation_models.pytorch.URL:https://github.com/qubvel-org/segmentation_models.pytorch
8. RUGD Team. RUGD Dataset for Semantic Segmentation, 2019. URL: http://rugd.vision/, дата публикации: 1 сентября **2019**.
9. Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, Hartwig Adam. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. arXiv, **2017**. URL:https://arxiv.org/abs/1704.04861
10. Mingxing Tan, Quoc V. Le. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks.arXiv, **2019**. URL: https://arxiv.org/abs/1905.11946
11. Karen Simonyan, Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv, **2014**. URL: https://arxiv.org/abs/1409.1556
12. Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. Deep Residual Learning for Image Recognition.arXiv, **2015**. URL:https://arxiv.org/abs/1512.03385
13. Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, Kaiming He. Aggregated Residual Transformations for Deep Neural Networks. arXiv, **2016**. URL: https://arxiv.org/abs/1611.05431
14. Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, Silvio Savarese. Generalized Intersection over Union: A Metric and A Loss for Bounding Box Regression. arXiv, **2015**. URL: https://arxiv.org/abs/1902.09630
15. Neural Network Compression Framework. URL: https://github.com/openvinotoolkit/nncf
16. Eunhyeok Park, Sungjoo Yoo, Peter Vajda. Quantization Networks. 2019. URL: https://openaccess.thecvf.com/content_CVPR_**2019**/html/Yang_Quantization_Networks_CVPR_2019_paper.html