

**Министерство образования и науки Российской Федерации
Московский физико-технический институт (государственный
университет)**

Физтех-школа прикладной математики и информатики
Кафедра анализа данных

Выпускная квалификационная работа магистра по специальности 09.03.01
«Информатика и вычислительная техника»

Одновременное картирование и локализация по
видео с построением плотной карты в реальном
времени

Студент группы
Муравьев К. Ф.

Научный руководитель

Рецензент

Долгопрудный
2021

Содержание

1. Введение	4
2. Одновременное картирование и локализация (SLAM)	6
2.1. Постановка задачи	6
2.1.1. Одновременное картирование и локализация по видеопотоку (vSLAM)	6
2.1.2. Одновременное картирование и локализация по видеоданным с картами глубины (RGBD-SLAM)	7
2.1.3. Сведение RGBD-SLAM к vSLAM: восстановление карт глубин по видеопотоку	8
2.2. Метрики качества	10
2.2.1. Метрики качества локализации	10
2.2.2. Метрики качества картирования	11
2.3. Коллекции данных	14
2.3.1. Коллекции данных из реального мира	15
2.3.2. Симуляторы и коллекции синтетических данных	17
3. Методы решения задачи vSLAM	19
3.1. Классические методы	19
3.1.1. ORB-SLAM и ORB-SLAM2	19
3.1.2. LSD-SLAM	20
3.1.3. RTAB-MAP	22
3.1.4. RGBDSLAM_v2	25
3.2. Нейросетевые методы	26
3.3. Выводы	28
4. Восстановление глубины по видеопотоку	30
4.1. Восстановление глубины по одиночным изображениям	30
4.2. Восстановление глубины с извлечением информации из видеопотока	33
4.3. Выводы	36
5. Предлагаемый метод одновременного картирования и локализации	37
5.1. Описание метода	37
5.2. Программно-аппаратное обеспечение	39
5.3. Эксперименты	41

5.3.1. Эксперименты в симуляторе	41
5.3.2. Эксперименты на реальном роботе	43
5.4. Выводы	43
6. Применение в задаче исследования неизвестной местности	45
6.1. Описание задачи исследования неизвестной местности	45
6.1.1. Постановка задачи	45
6.1.2. Метрики качества	46
6.2. Описание метода	46
6.3. Эксперименты	49
6.4. Выводы	51
7. Заключение	52

Глава 1

Введение

В последнее время мобильные роботы и беспилотные летательные аппараты все чаще используются в различных коммерческих и бытовых целях. В некоторых случаях, например, на больших расстояниях или в условиях сильных радиопомех, дистанционное управление подобными аппаратами может быть затруднено. В таких случаях возникает необходимость автономного функционирования беспилотных аппаратов. Для успешного выполнения многих задач в автономном режиме необходимо определять положение аппаратов в пространстве, а также положение окружающих объектов. Возникает задача одновременного картирования и локализации (simultaneous localization and mapping, SLAM).

Решение задачи SLAM зачастую осложняется различными ограничениями. Например, внутри помещений невозможна навигация с помощью GPS; также на малогабаритных роботах ограничения по весу и энергопотреблению не позволяют установить большое количество высокоточных датчиков. В некоторых случаях единственным доступным сенсором является видеокамера. Возникает задача одновременного картирования и локализации по видеопотоку (vision-based SLAM, vSLAM).

Классические методы vSLAM, основанные на извлечении особых точек из изображений [40][29], имеют существенные недостатки, такие, как необходимость задания масштаба пользователем, потеря локализации при повороте на месте и разреженность построенной карты. Разреженные карты, построенные подобными методами, не пригодны для планирования маршрутов, поскольку маршрут может быть проложен через не нанесенную на карту точку препятствия.

Восстановление карт глубин по видеопотоку позволяет устраниТЬ вышеописанные недостатки, сведя задачу vSLAM к задаче одновременного картирования и локализации с использованием видеоданных и данных о глубине (RGBD-SLAM). Для решения задачи RGBD-SLAM разработаны эффективные методы решения с построением плотной карты, например, [14][34].

Для восстановления глубины по видеоданным обычно используются методы, основанные на вычислении оптического потока между соседними кадрами видеоряда. Как правило, производительность таких методов недостаточна для обработки видео-

потока в реальном времени с помощью бортовых вычислителей робототехнических систем. Например, обработка одной пары изображений с помощью метода, предложенного в работе [55], занимает 110 мс на видеокарте GTX Titan. На бортовых компьютерах малогабаритных роботов обработка изображений будет выполняться еще дольше.

В настоящее время помимо классических алгоритмов восстановления глубины применяются также нейросетевые методы, обрабатывающие отдельно каждый кадр видеопоследовательности. Подобные методы позволяют достичь приемлемого качества восстановления глубины и производительности, достаточной для обработки видеопотока в реальном времени, но требуют наличия мощного графического ускорителя, что затрудняет их применение для навигации беспилотных транспортных средств малого размера. Например, в работе [35] удалось достичь скорости обработки видеопотока в 18 кадров в секунду на видеокарте GTX Titan, которая не может использоваться в мобильных роботах из-за очень высокого энергопотребления.

С развитием вычислительной техники в последнее время стало возможным применение нейронных сетей на малогабаритных роботах. Например, встраиваемый компьютер NVIDIA Jetson TX2 [16] обладает графическим ускорителем и мощным центральным процессором, и при этом он достаточно компактен и энергоэффективен для использования в малых робототехнических устройствах. В работе [57] описывается метод восстановления карт глубин изображений, основанный на нейронной сети и работающий на NVIDIA Jetson TX2 со скоростью до 175 кадров в секунду.

В данной работе предлагается метод решения задачи vSLAM, основанный на восстановлении карт глубин изображений с помощью нейронных сетей и алгоритме RTAB-MAP [34]. Проводится оценка качества предложенного метода в фотогеометрической симуляционной среде, а также тестирование на колесном роботе с бортовым компьютером NVidia Jetson TX2 в реальном времени. Для оценки качества методов vSLAM в симуляционной среде собрана коллекция данных [4], содержащая видеопоследовательности с картами глубин, а также истинные траектории движения камеры и истинные карты помещений. На видеопоследовательностях собранной коллекции вычисляются традиционные метрики качества картирования и локализации, а также оригинальная метрика качества картирования из работы [2].

С целью исследования применимости предложенного метода vSLAM для автономной навигации роботов, данный метод встраивается в систему автономного исследования неизвестной местности (ИНМ). Для определения эффективности полученной системы ИНМ проводятся эксперименты в фотогеометрической симуляционной среде.

Глава 2

Одновременное картирование и локализация (SLAM)

2.1 Постановка задачи

2.1.1 Одновременное картирование и локализация по видеопотоку (*vSLAM*)

Задача одновременного картирования и локализации по визуальным данным (visual-based simultaneous localization and mapping, vSLAM) возникает при навигации в неизвестной среде робота, не имеющего на борту никаких сенсоров, кроме единственной видеокамеры. Задача формулируется следующим образом: по изображениям с видеокамеры необходимо построить трёхмерную модель окружающего пространства, и определить траекторию перемещения камеры в этом пространстве (см. рис. 1).

Математически задачу можно сформулировать таким образом: существует набор точек в трёхмерном пространстве $\{M_i\} = \mathbf{W}, M_i \in \mathbb{R}^3$, называемый сценой. Данна последовательность кадров $\{\mathcal{P}^t\}$. Каждый кадр является проекцией точек сцены с ракурса $\mathcal{R}_t = (x_t, y_t, z_t, p_t, r_t, w_t) \in \mathbb{R}^6$. Числа x_t, y_t, z_t задают пространственное положение камеры в момент времени t , а p_t, r_t, w_t - углы направления главной оптической

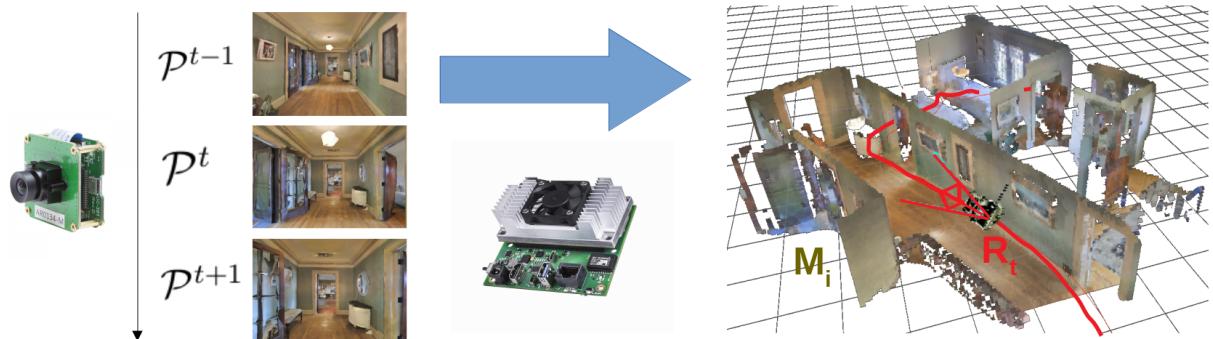


Рис. 1. Постановка задачи vSLAM

оси камеры в момент времени t .

Кадр представляется в виде трех матриц размер $H \times W$, содержащих числа от 0 до 1 - яркости соответствующих пикселей красной, синей и зеленой цветовых компонент:

$$\mathcal{P}^t = \{\mathcal{P}_{c,h,w}^t\}_{c \in [1 \dots 3], h \in [1 \dots H], w \in [1 \dots W]} \in [0, 1]^{3 \times H \times W}$$

Каждый элемент c -й матрицы кадра $\mathcal{P}_{c,h,w}^t$ представляет собой яркость c -й цветовой компоненты точки, которая в момент времени t спроектировалась на позицию (h, w) в матрице камеры:

$$P(\mathcal{R}_t, M_i) = (h, w) \Rightarrow \mathcal{P}_{c,h,w}^t = I_c(\mathcal{R}_t, M_i),$$

где $P(\mathcal{R}_t, M_i) : \mathbb{R}^6 \times \mathbb{R}^3 \rightarrow \mathbb{R}^2$ - функция проекции точки пространства на матрицу камеры, принимающая на вход ракурс и положение точки в пространстве и возвращающая координаты проекции, а $I_c(\mathcal{R}_t, M_i) : \mathbb{R}^6 \times \mathbb{R}^3 \rightarrow [0, 1]$ - функция яркости c -й цветовой компоненты точки M_i , рассматриваемой с ракурсом \mathcal{R}_t .

По имеющейся последовательности кадров $\{\mathcal{P}_t\}$ необходимо найти (x_t, y_t, z_t) для всех моментов времени t - координаты ракурсов \mathcal{R}_t , а также как можно большее количество координат точек M_i .

2.1.2 Одновременное картирование и локализация по видеоданным с картами глубины (RGBD-SLAM)

Задача одновременного картирования и локализации по видеоданным и данным глубины (RGBD-SLAM) возникает при навигации в неизвестной среде робота, имеющего на борту видеокамеру и сенсор глубины. Задача формулируется следующим образом: по изображениям с видеокамеры и картам глубины этих изображений необходимо построить трёхмерную модель окружающего пространства, и определить траекторию перемещения камеры в этом пространстве.

Математически задачу можно сформулировать таким образом: существует набор точек в трёхмерном пространстве $\{M_i\} = \mathbf{W}, M_i \in \mathbb{R}^3$, называемый сценой. Даны последовательность кадров $\{\mathcal{P}^t\}$ и карт глубины $\{\mathcal{D}^t\}$.

Кадр представляется в виде трех матриц размер $H \times W$, содержащих числа от 0 до 1 - яркости соответствующих пикселей красной, синей и зеленой цветовых компонент:

$$\mathcal{P}^t = \{\mathcal{P}_{c,h,w}^t\}_{c \in [1 \dots 3], h \in [1 \dots H], w \in [1 \dots W]} \in [0, 1]^{3 \times H \times W}$$

Каждый элемент c -й матрицы кадра $\mathcal{P}_{c,h,w}^t$ представляет собой яркость c -й цветовой компоненты точки, которая в момент времени t спроектировалась на позицию (h, w)

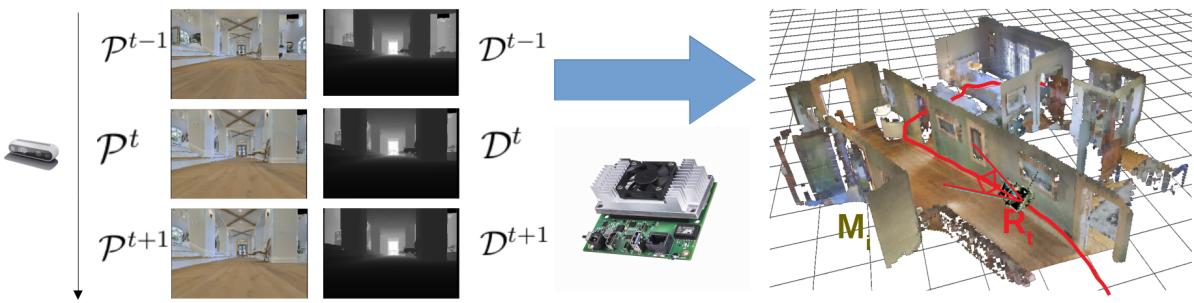


Рис. 2. Постановка задачи RGB-D SLAM

в матрице камеры:

$$P(\mathcal{R}_t, M_i) = (h, w) \Rightarrow \mathcal{P}_{c,h,w}^t = I_c(\mathcal{R}_t, M_i)$$

Карта глубины представляется в виде матрицы размера $H \times W$, содержащей положительные действительные числа - глубины соответствующих пикселей:

$$\mathcal{D}^t = \{\mathcal{D}_{h,w}^t\}_{h \in [0 \dots H], w \in [0 \dots W]}$$

Элемент матрицы карты глубины $D_{h,w}^t$ представляет собой расстояние от положения камеры в момент времени t до точки, которая в момент времени t спроектировалась на позицию (h, w) в матрице камеры:

$$P(\mathcal{R}_t, M_i) = (h, w) \Rightarrow \mathcal{D}_{h,w}^t = \rho(M_i, (x_t, y_t, z_t))$$

Здесь $P(\mathcal{R}_t, M_i) : \mathbb{R}^6 \times \mathbb{R}^3 \rightarrow \mathbb{R}^2$ - функция проекции точки пространства на матрицу камеры, принимающая на вход ракурс и положение точки в пространстве и возвращающая координаты проекции, а $I_c(\mathcal{R}_t, M_i) : \mathbb{R}^6 \times \mathbb{R}^3 \rightarrow [0, 1]$ - функция яркости c -й цветовой компоненты точки M_i , рассматриваемой с ракурса \mathcal{R}_t . Функция $\rho : \mathbb{R}^3 \times \mathbb{R}^3 \rightarrow \mathbb{R}_+$ задает евклидово расстояние между двумя точками в пространстве.

По имеющейся последовательности кадров $\{\mathcal{P}_t\}$ и карт глубин $\{\mathcal{D}_t\}$ необходимо найти (x_t, y_t, z_t) для всех моментов времени t - координаты ракурсов \mathcal{R}_t , а также как можно большее количество координат точек M_i . Визуализация задачи RGB-D SLAM представлена на рисунке 2.

2.1.3 Сведение RGBD-SLAM к vSLAM: восстановление карт глубин по видеопотоку

Задача восстановления карт глубин по видеопотоку возникает при навигации в неизвестной среде робота, не имеющего на борту никаких сенсоров, кроме видеокамеры. С помощью восстановления глубины по видеопотоку можно свести задачу

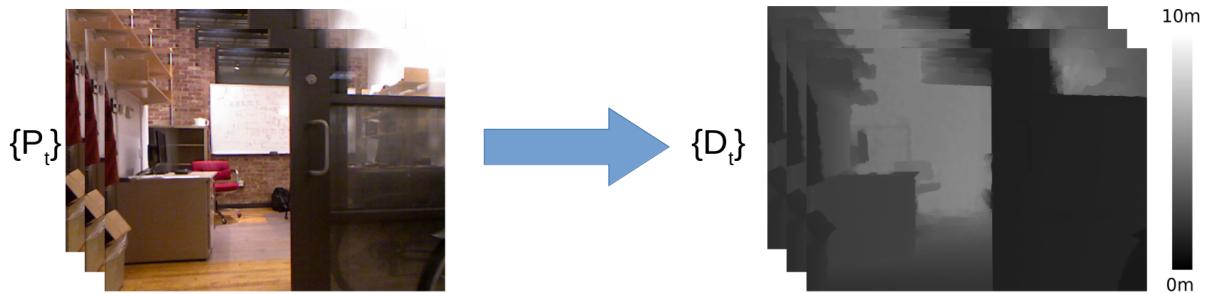


Рис. 3. Постановка задачи восстановления глубины

vSLAM к задаче RGBD-SLAM, для которой разработаны более эффективные методы решения.

Задача восстановления карт глубин по видеопотоку формулируется следующим образом: по изображениям, поступающим с единственной видеокамеры, необходимо определить расстояния до всех объектов, изображенных на этих изображениях (см. рис. 3). Математически задачу можно сформулировать таким образом: существует набор точек в трёхмерном пространстве $\{M_i\} = \mathbf{W}, M_i \in \mathbb{R}^3$, называемый сценой. Данна последовательность кадров $\{\mathcal{P}^t\}$. Каждый кадр является проекцией точек сцены с ракурса $\mathcal{R}_t = (x_t, y_t, z_t, p_t, r_t, w_t) \in \mathbb{R}^6$. Числа x_t, y_t, z_t задают пространственное положение камеры в момент времени t , а p_t, r_t, w_t - углы направления главной оптической оси камеры в момент времени t .

Кадр представляется в виде трех матриц размер $H \times W$, содержащих числа от 0 до 1 - яркости соответствующих пикселей красной, синей и зеленой цветовых компонент:

$$\mathcal{P}^t = \{\mathcal{P}_{c,h,w}^t\}_{c \in [1 \dots 3], h \in [1 \dots H], w \in [1 \dots W]} \in [0, 1]^{3 \times H \times W}$$

Каждый элемент c -й матрицы кадра $\mathcal{P}_{c,h,w}^t$ представляет собой яркость c -й цветовой компоненты точки, которая в момент времени t спроектировалась на позицию (h, w) в матрице камеры:

$$P(\mathcal{R}_t, M_i) = (h, w) \Rightarrow \mathcal{P}_{c,h,w}^t = I_c(\mathcal{R}_t, M_i),$$

где $P(\mathcal{R}_t, M_i) : \mathbb{R}^6 \times \mathbb{R}^3 \rightarrow \mathbb{R}^2$ - функция проекции точки пространства на матрицу камеры, принимающая на вход ракурс и положение точки в пространстве и возвращающая координаты проекции, а $I_c(\mathcal{R}_t, M_i) : \mathbb{R}^6 \times \mathbb{R}^3 \rightarrow [0, 1]$ - функция яркости c -й цветовой компоненты точки M_i , рассматриваемой с ракурсом \mathcal{R}_t .

По имеющейся последовательности кадров $\{\mathcal{P}_t\}$ необходимо для всех h, w, t найти $\mathcal{D}_{h,w}^t$ - расстояния от положения камеры в момент t до точек сцены, изображенных на кадре:

$$P(\mathcal{R}_t, M_i) = (h, w) \Rightarrow \mathcal{D}_{h,w}^t = \rho((x_t, y_t, z_t), M_i)$$

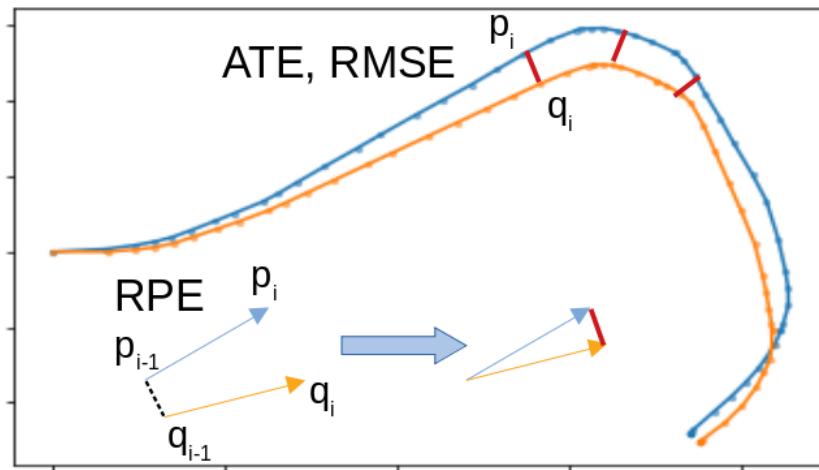


Рис. 4. Схема вычисления абсолютной и относительной ошибки траектории

2.2 Метрики качества

Для оценки качества алгоритмов одновременного картирования и локализации (vSLAM) и их сравнения между собой необходимо выбрать метрику оценки качества. Алгоритмы vSLAM, как правило, дают в качестве выходных данных карту окружающей местности и траекторию перемещения камеры. Качество построения карты и восстановления траектории, как правило, оценивается отдельно. Таким образом, метрики качества vSLAM подразделяются на две группы: метрики качества локализации и метрики качества картирования.

2.2.1 Метрики качества локализации

Метрики качества локализации, как правило, сравнивают траекторию, вычисленную алгоритмом SLAM, с истинной траекторией. Траектория представляет собой набор поз, каждая из которых включает трехмерную позицию (положение камеры в пространстве) и трехмерную ориентацию (направление главной оптической оси камеры). Оценка качества локализации сводится к вычислению ошибки между набором истинных и предсказанных поз. В научной литературе используются абсолютные и относительные ошибки, а также ошибки смещения и поворота.

Обозначим истинную траекторию как $\{(p_t, q_t)\}, t \in \{1, \dots, T\}$, где $p_t \in \mathbb{R}^3$ - трехмерная позиция камеры в момент времени t , $q_t \in \mathbb{R}^3$ - вектор направления главной оптической оси камеры в момент времени t . Предсказанную траекторию обозначим как $\{(p_t^*, q_t^*)\}, t \in \{1, \dots, T\}$. Применяя различные функции ошибки к спискам поз $\{(p_t, q_t)\}$ и $\{(p_t^*, q_t^*)\}$, можно вычислить абсолютную и относительную ошибку траектории (см. рис. 4).

Абсолютная ошибка траектории (Absolute Trajectory Error, ATE) формулируется как среднеквадратичное отклонение точек предсказанной траектории от истинной:

$$ATE = \sqrt{\frac{1}{T} \sum_{t=1}^T \|p_t - p_t^*\|_2^2} \quad (1)$$

Помимо абсолютной ошибки траектории, также широко применяется относительная ошибка позы (Relative Pose Error, RPE). Она формулируется как среднеквадратичное отклонение предсказанного смещения на каждом шаге от истинного:

$$\Delta p_t = M_{q_{t-1}}^{-1}(p_t - p_{t-1}); \Delta p_t^* = M_{q_{t-1}^*}^{-1}(p_t^* - p_{t-1}^*)$$

$$RPE = \sqrt{\frac{1}{T} \sum_{t=1}^T \|\Delta p_t - \Delta p_t^*\|_2^2}, \quad (2)$$

где $M_{q_{t-1}}$, $M_{q_{t-1}^*}$ - матрицы вращения, переводящие вектор $(1, 0, 0)$ в векторы q_{t-1} и q_{t-1}^* соответственно.

В работе [18] приводятся следующие метрики качества локализации: относительное смещение и относительная ошибка поворота. Данные метрики учитывают не только расхождение между траекториями, но и их длину:

$$E_{trans} = \frac{1}{T} \sum_{t=1}^T \frac{\|M_{q_{t-1}}(p_t - p_{t-1}) - M_{q_{t-1}^*}(p_t^* - p_{t-1}^*)\|_2}{\|p_t - p_{t-1}\|_2} \quad (3)$$

$$E_{rot} = \frac{1}{T} \frac{\angle(M_{q_{t-1}}(p_t - p_{t-1}), M_{q_{t-1}^*}(p_t^* - p_{t-1}^*))}{\|p_t - p_{t-1}\|_2} \quad (4)$$

2.2.2 Метрики качества картирования

Оценка качества картирования является более сложной задачей, чем оценка качества локализации. Если при оценке качества локализации легко установить соответствие между точками истинной и предсказанной траекторий (по времени прохождения данных точек), то явного соответствия между точками истинной и предсказанной карты не существует. Как правило, соответствия устанавливаются методом ближайшего соседа - каждая точка предсказанной карты сопоставляется с ближайшей к ней точкой истинной карты. Такой подход применяется в программном пакете CloudCompare¹ и в работах [22][56]. Применяя среднеквадратичную ошибку (RMSE) в этом подходе, получаем абсолютную ошибку картирования (Absolute Mapping Error, AME).

По заданной истинной карте, представленной в виде трехмерного облака точек:

$$M = \{m_i \in \mathbb{R}^3; i \in [1; n]; n \in \mathbb{N}\} \quad (5)$$

и карте, построенной алгоритмом vSLAM:

¹<http://cloudcompare.org/>

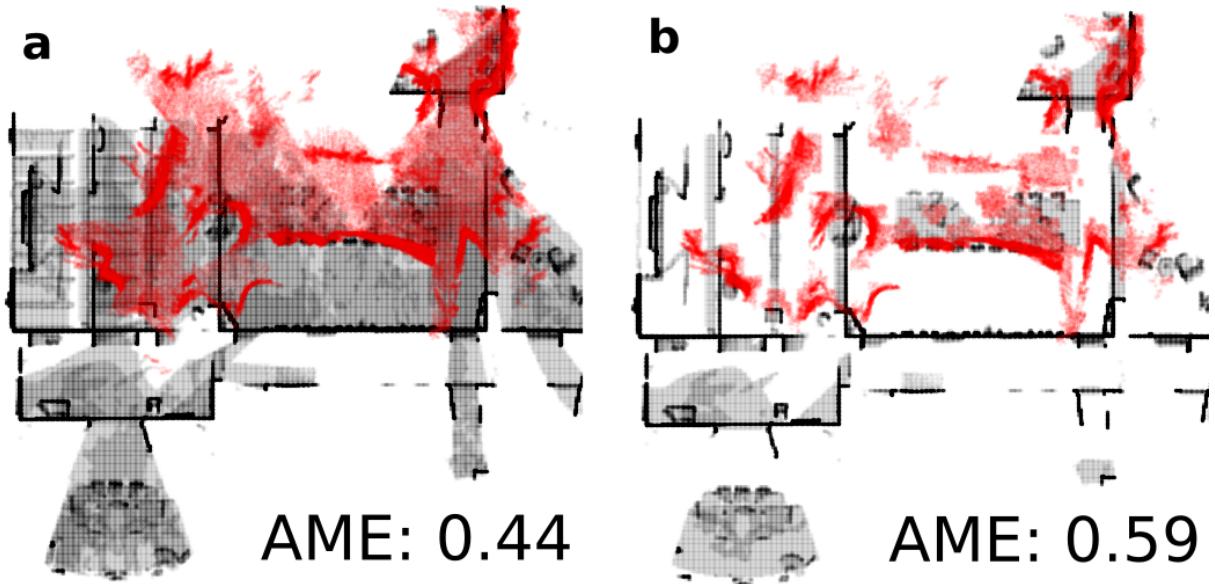


Рис. 5. Исходные карты (а) и карты с вырезанным полом (б). Чёрным отмечены точки истинной карты, красным - предсказанный методом vSLAM карты. Метрика AME на картах с полом значительно ниже, чем на тех же картах без пола.

$$M^* = \{m_j^* \in \mathbb{R}^3; j \in [1; N]; n \in \mathbb{N}\} \quad (6)$$

абсолютная ошибка картирования вычисляется следующим образом:

$$j' = \arg \min_i \|m_j^* - m_i\|_2$$

$$AME(M, M^*) = \sqrt{\frac{1}{N} \sum_{j=1}^N ||m_j^* - m_{j'}||_2^2} \quad (7)$$

Однако существуют ситуации, когда абсолютная ошибка картирования, вычисленная по методу ближайшего соседа, не является репрезентативной. Например, при картировании помещений большой площади точки стен предсказанной карты могут быть сопоставлены с точками пола истинной карты. Таким образом абсолютная ошибка картирования может быть небольшой даже при неточном картировании. Так получилось в одном из экспериментов, описанных в разделе 5.3.1. При картировании помещений алгоритмом SLAM с глубинами, предсказанными нейросетью, обнаружилось, что карты визуально получались довольно неточные, при этом значение метрики AME на этих картах оказалось низким. При удалении точек пола из истинных и предсказанных карт значение метрики AME существенно выросло (см. рис. 5). Таким образом, в контексте оценки качества алгоритмов vSLAM метрика AME с сопоставлением по методу ближайшего соседа малоприменима.

В данной работе для оценки качества картирования предлагается использовать способ сопоставления точек истинной и предсказанной карты, описанный в работе [2]. Предлагаемый способ основан на сопоставлении ракурсов, с которых видны точки

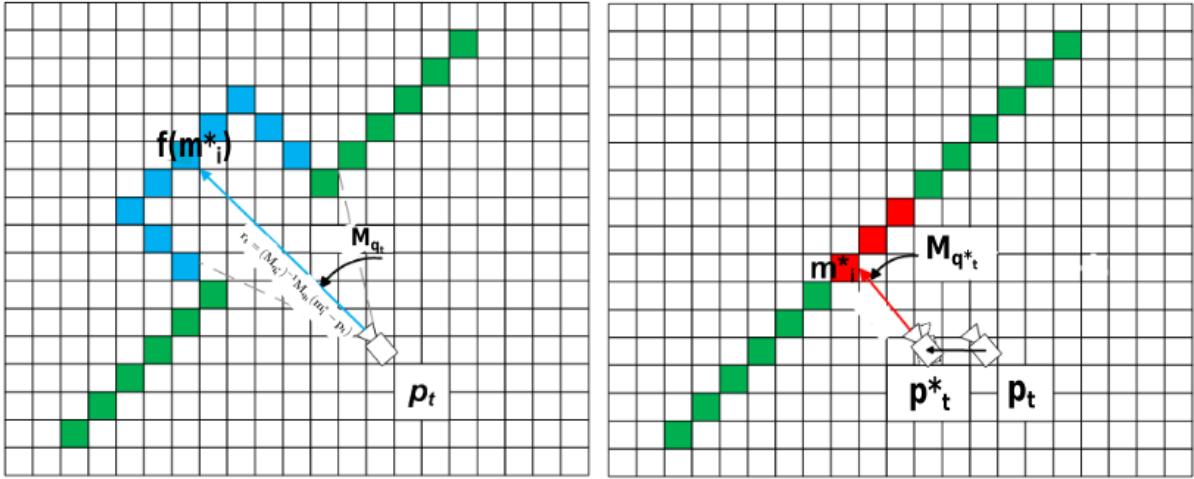


Рис. 6. Пример сопоставления точек истинной и предсказанной карты с использованием ракурса. Источник [2]

истинной и предсказанный карт. Ниже приведено его подробное описание.

Пусть M , M^* - истинная и построенная алгоритмом vSLAM карты (5, 6); m_i^* - точка карты M^* , попадающая в поле зрения камеры в момент времени t ; p_t^*, q_t^* - предсказанные методом vSLAM положение и ориентация камеры в момент времени t ; p_t, q_t - истинные положение и ориентация камеры в момент t . Нужно построить функцию $f : M^* \rightarrow M$, устанавливающую соответствие между точками построенной и истинной карты.

Обозначим матрицы вращения, заданные ориентациями q_t и q_t^* , как M_{q_t} и $M_{q_t^*}$ соответственно. Вектор $r_t = (M_{q_t^*})^{-1} M_{q_t} (m_i^* - p_t^*)$ соответствует направлению с позиции камеры на точку m_i^* в истинной карте (точка $p_t + \alpha r_t$ в истинной карте будет видна в момент t под тем же ракурсом, что точка m_i^* в предсказанный карте, см. рис. 6). Точке m_i^* будет сопоставлена ближайшая точка истинной карты, которая видна под таким ракурсом:

$$f(m_i^*) = p_t + \alpha r_t; \quad \alpha = \arg \min_{\alpha'} : p_t + \alpha' r_t \in M \quad (8)$$

Абсолютная и относительная ошибки картирования с таким методом сопоставления будет выглядеть следующим образом:

$$AME(M, M^*) = \sqrt{\frac{1}{N} \sum_{i=1}^N \|m_i^* - f(m_i^*)\|_2^2} \quad (9)$$

$$RME(M, M^*) = \sqrt{\frac{1}{N} \sum_{i=1}^N \|M_{q_t^*}^{-1}(m_i^* - p_t^*) - M_{q_t}^{-1}(f(m_i^*) - p_t)\|_2^2} \quad (10)$$

При использовании данных метрик возникает следующая проблема. Одна точка карты может быть видна с разных позиций (т.е. для одной точки m_i^* есть несколько

t , по которым можно построить разные соответствия). Поэтому нужно определиться со способом выбора t . Наиболее целесообразными выглядят следующие варианты:

- t выбирается как момент, в который точка m_i^* попала в поле зрения камеры в первый раз;
- t выбирается как момент, в который точка m_i^* попала в поле зрения камеры в последний раз;
- t выбирается как момент видимости точки m_i^* , в который точка была наиболее близка к позиции камеры;
- рассматриваются все моменты t , в который точка попадала в поле зрения камеры; при вычислении метрики берется усредненное расстояние между точкой m_i^* и точками $f(m_i^*, t)$ для всех t .

Последний вариант выбора t приводит к большим вычислительным затратам, однако метрика с ним получается наиболее стабильной. Для оценки качества алгоритмов vSLAM в данной работе был выбран именно этот вариант.

Метрики AME и RME, вычисленные на основе функции сопоставления f , являются более подходящими для оценки качества алгоритмов vSLAM, поскольку они учитывают не только расстояния между точками, но и процесс построения карты алгоритмом vSLAM. Также данные метрики являются более устойчивыми к различным изменениям структуры карты (например, удаление точек пола), что подтверждено экспериментами, описанными в разделе 5.3.1.

2.3 Коллекции данных

Проведение натурных экспериментов на реальной робототехнической системе затратно, а повторяемость таких экспериментов затруднена, поэтому для тестирования алгоритмов vSLAM и их сравнения между собой обычно применяются предварительно собранные коллекции данных. Такие коллекции должны включать в себя запись видеопотока с камеры робота (в случае RGB-D камеры - запись изображений и данных о глубине), а также данные об истинных позициях робота и истинной карте окружающей местности для вычисления метрик качества. Подобные коллекции бывают двух видов: собранные в реальном мире и синтетические.

При сборке коллекций данных в реальном мире возникают сложности с определением истинной траектории робота и точной модели окружающей местности. Для вычисления точных данных о движении робота и его окружающей среде необходимо редкое и дорогостоящее оборудование, а также значительные временные затраты. Как правило, в коллекциях данных из реального мира имеются данные лишь о траектории робота, собранные с помощью систем отслеживания движения (Motion-capture System [32]), а истинные карты окружающей среды отсутствуют. При тестировании

методов vSLAM на таких коллекциях можно вычислить метрики ATE 1, RPE 2, E_{trans} 3, E_{rot} 4, но невозможно вычислить метрики AME и RME (7, 9, 10).

Синтетические коллекции данных, а также различные робототехнические симуляторы (например, [30] [45]) позволяют легко получить истинные карты и траектории, однако в большинстве своем обладают низкой фотореалистичностью, что критически важно для алгоритмов визуального картирования и локализации.

2.3.1 Коллекции данных из реального мира

Коллекции данных, собранные с помощью прогонов системы, оснащенной камерой, по некоторой траектории в реальном мире, на данный момент очень широко распространены. Имеется множество коллекций, в которых представлены прогоны робота в помещениях, а также несколько крупных коллекций, собранных с автомобиля, оснащенного большим количеством датчиков. В большинстве таких коллекций представлены истинные траектории движения робота, но отсутствуют истинные карты окружающей местности. Подобные датасеты хорошо подходят для оценки качества локализации методов vSLAM, однако оценка качества картирования на них затруднена или вовсе невозможна. Ниже представлен обзор наиболее широко используемых наборов данных из реального мира для тестирования vSLAM.

Одним из самых известных датасетов для тестирования алгоритмов SLAM является KITTI [18]. Он содержит 22 сцены, записанные с автомобиля в городской среде. Общая длина проезда составляет 39 км. Датасет содержит порядка 41 тысячи кадров видеопотока, записанного со стереокамеры, а также данные трехмерного лазерного сканера (лидара) и инерциальной навигационной системы (ИНС) совместно с показаниями GPS. В данной коллекции также представлены истинные позиции, вычисленные с помощью агрегации данных GPS и ИНС и качественной пост-обработки. Истинная карта окружающих объектов не представлена, однако она может быть приблизительно воссоздана по истинным позициям автомобиля и облакам точек с лидара.

Помимо исходных данных с сенсоров, в KITTI также представлено программное обеспечение, необходимое для вычисления метрик качества локализации. Также в датасете предоставлены данные для тестирования некоторых других методов компьютерного зрения: семантической сегментации, детекции объектов, вычисления оптического потока. Данный датасет широко используется для оценки качества методов визуальной одометрии [19], а также для обучения различных нейросетевых моделей [36] [63] [44]. Однако оценка качества картирования на данном датасете затруднительна, поскольку истинных карт окружающей среды в датасете не предоставлено.

Одним из наиболее широко используемых датасетов для тестирования vSLAM в помещениях является RGB-D SLAM dataset and benchmark [50], созданный учеными из Технического университета Мюнхена (TUM). Датасет содержит 39 последовательностей, записанных в различных помещениях с камеры Microsoft Kinect [61], установ-

ленной на малом колесном роботе. Последовательности были записаны в различных помещениях, длины траекторий не превосходят 40 метров. Помимо изображений и карт глубины с камеры Kinect, в датасете также имеются истинные траектории проезда робота, вычисленные с помощью высокоточной системы отслеживания движения (Motion Capture System).

Датасет от TUM также содержит набор программного обеспечения для тестирования алгоритмов SLAM и вычисления метрик ATE (1) и RPE (2). Благодаря набору ПО, а также наличию истинных траекторий, данный датасет хорошо подходит для оценки качества локализации методов vSLAM и RGB-D SLAM в помещениях. Однако оценка качества картирования на нем невозможна, так как не представлены истинные координаты точек окружающих объектов. Обучение нейронных сетей на данной коллекции затруднительно из-за ее небольшого объема, а также однообразности изображений.

Еще одним широко используемым для тестирования vSLAM датасетом является EuRoC [7]. В этом датасете представлены данные с видеокамеры, установленной на квадрокоптере, собранные с 11 прогонов по двум комнатам. Помимо видеоданных, также имеются данные ИНС и истинные траектории квадрокоптера, измеренные с помощью системы отслеживания движения (Motion Capture System). Карт глубины датасет не содержит, поэтому он не подходит для оценки качества алгоритмов RGB-D SLAM. Программного обеспечения для запуска алгоритмов vSLAM и вычисления метрик разработчики также не предоставили. Ввиду всего вышеперечисленного применимость данного датасета для оценки качества методов vSLAM весьма ограничена.

В настоящее время существуют также коллекции данных, в которых помимо траекторий предоставлены трехмерные модели окружающих объектов, что дает возможность оценить не только качество локализации, но и качество картирования. Одной из самых крупных таких коллекций является ScanNet [12]. В данной коллекции представлены 2.5 млн пар изображение-глубина, снятые RGB-D камерами в 1513 помещениях. Траектории камеры и трехмерные модели помещений были получены с помощью тщательной алгоритмической обработки данных RGB-D камеры и ИНС. Сенсоры, использованные при создании датасета, имеют достаточно большую погрешность, поэтому полученные таким образом траектории и карты сложно считать “истинными”. При проведении экспериментов с методами vSLAM возможны искажения оценок их качества.

Датасет ScanNet изначально не предназначался для тестирования алгоритмов vSLAM, и в нем не предоставлено нужного для этого программного обеспечения и удобного формата данных. Поэтому тестирование методов vSLAM на данном датасете весьма затруднительно. Однако датасет может быть полезен для решения вспомогательных задач для visual SLAM - например, обучения нейросетей восстановления глубины и визуальной одометрии.



Рис. 7. Пример симуляционной среды Gazebo (слева) и CoppeliaSim (справа). Однообразность текстур делает работу методов Visual SLAM затруднительной

2.3.2 Симуляторы и коллекции синтетических данных

Благодаря бурному развитию вычислительной техники и компьютерных технологий, в последние годы были разработаны различные робототехнические симуляторы. Появление симуляторов дало широкие возможности для тестирования робототехнических алгоритмов, в том числе методов картирования и локализации. В симуляционной среде робот может перемещаться по произвольной траектории, для которой всегда известны истинные позиции, что дает возможность проведения неограниченного числа экспериментов по оценке качества методов SLAM.

Наиболее применяемыми на данный момент симуляторами являются Gazebo [30] и CoppeliaSim (V-REP) [45]. Эти симуляторы отличаются подробным моделированием различных физических процессов, благодаря чему в них можно имитировать работу различных сенсоров, таких, как ИНС и лазерные сканеры, а также видеокамер и RGB-D камер. Однако фотoreалистичность интерьеров в подобных симуляторах довольно низкая (см. рис. 7) - из-за повторяемости текстур и неточного моделирования распространения света могут возникнуть проблемы с тестированием алгоритмов Visual SLAM.

Проблемы фотoreалистичности частично решились с появлением симулятора Habitat [47]. В данном симуляторе не моделируются такие физические процессы, как инерция движения и распространение лазерных лучей, однако изображения сцен, используемые в нем, отличаются высокой фотoreалистичностью (см. рис. 8). Как правило, в симуляторе Habitat используются сцены из коллекции Gibson [58] или Matterport3D [9]. Их фотoreалистичность обусловлена тем, что сцены построены по реальным помещениям с помощью специальной камеры² и тщательной алгоритмической постобработки. Коллекция Matterport3D содержит около 60 сцен площадью в несколько сотен квадратных метров каждая. Коллекция Gibson содержит около 500 сцен площадью несколько десятков квадратных метров каждая. В обоих коллекциях представлены здания и помещения различного типа. Симуляционная среда дает возможность перемещаться в этих помещениях по произвольной траектории.

²<https://matterport.com/>



Рис. 8. Пример изображений с камеры робота в симуляционной среде Habitat на сцене из датасета Matterport3D

В рамках данной работы на основе симулятора Habitat и коллекции сцен Matterport3D был собран новый датасет для оценки качества алгоритмов vSLAM. Датасет содержит 40 траекторий, которые разделены на 20 перекрывающихся пар, что дает возможность также тестировать алгоритмы объединения карт (Map Merging). Длины траекторий варьируются от 4 до 33 метров. В датасете представлены RGB-D данные (суммарно около 30000 пар картинка-глубина), а также истинные траектории перемещения камеры и истинные карты помещений, построенные с помощью метода обратной проекции по истинным позициям и точным картам глубин. Подробное описание собранного датасета доступно в работе [4]. Эксперименты по оценке качества методов vSLAM на нем описаны в разделе 5.3.1.

Глава 3

Методы решения задачи vSLAM

3.1 Классические методы

Как правило, для решения задачи vSLAM применяются методы, основанные на вычислении геометрических преобразований между позициями камеры по изображениям. Преобразования могут вычисляться как прямыми методами (например, минимизацией фотометрической ошибки), так и путем сопоставления извлеченных из изображений особых точек. По вычисленным преобразованиям восстанавливается траектория перемещения камеры. Также с использованием этих преобразований обычно строятся локальные карты - карты участка местности, попадающего в поле зрения камеры, которые затем соединяются в глобальную карту с помощью различных вероятностных методов.

В настоящее время существует множество алгоритмов vSLAM [51] [17]. В данной работе представлен обзор методов vSLAM, предназначенных для работы в реальном времени в условиях ограниченных вычислительных ресурсов и имеющих открытый исходный код.

3.1.1 *ORB-SLAM* и *ORB-SLAM2*

Алгоритм ORB-SLAM [40] и его модификация ORB-SLAM2 [41] являются одними из наиболее популярных методов одновременного картирования и локализации. ORB-SLAM использует в качестве входных данных видеопоток с единственной камерой, а ORB-SLAM2 является расширением алгоритма ORB-SLAM для работы по данным стереокамеры или RGB-D камеры. В основе обоих алгоритмов лежит извлечение особых точек из изображений с помощью детектора ORB [46]. Высокая скорость детектора ORB позволяет методу работать в реальном времени в условиях ограниченных вычислительных ресурсов. Однако на карту наносятся лишь координаты особых точек, поэтому построенная алгоритмом карта получается разреженной (см. рис. 9) и является непригодной для планирования траектории (планировщик может проложить маршрут между двумя особыми точками, где может быть стена или

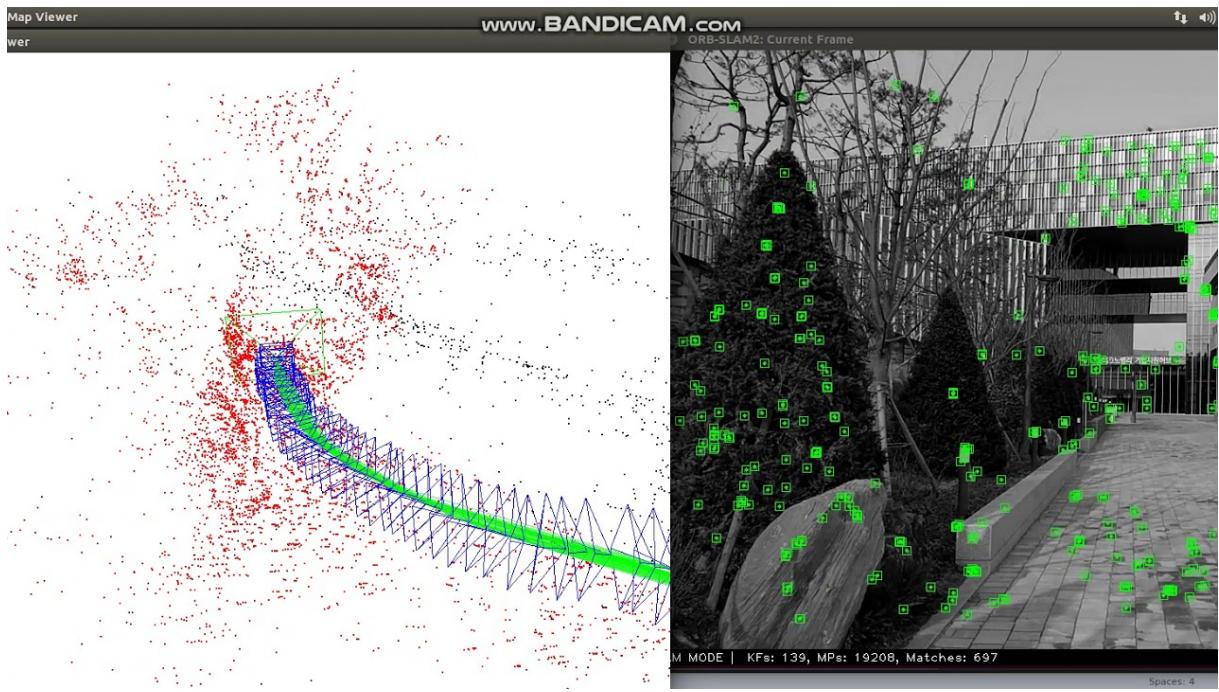


Рис. 9. Работа алгоритма ORB-SLAM: построенная карта (слева) и детекция особых точек на изображении (справа). Карта представляет собой сильно разреженный набор точек, что затрудняет навигацию в ней. Источник: https://www.youtube.com/watch?v=g_nFyS0muJw

другое препятствие).

Работа алгоритма ORB-SLAM разделена на три основных потока, выполняющихся параллельно:

- Tracking - отслеживание кадров. Данный поток приблизительно определяет текущее положение камеры с помощью поиска похожего кадра и сопоставления особых точек на нем и текущем кадре.
- Local Mapping - построение и оптимизация локальной карты (вблизи текущего положения камеры).
- Loop Closing - поиск и замыкание циклов с помощью объединения похожих кадров.

Схема основных компонентов алгоритма изображена на рисунке 10.

3.1.2 LSD-SLAM

Алгоритм LSD-SLAM [15] так же, как и ORB-SLAM, выполняет картирование и локализацию по данным с единственной камеры, однако он имеет другой принцип действия. Траектория перемещения камеры вычисляется не сопоставлением особых точек, а путем минимизации фотометрической ошибки по всему изображению. Благодаря использованию полного изображения вместо набора особых точек, LSD-SLAM

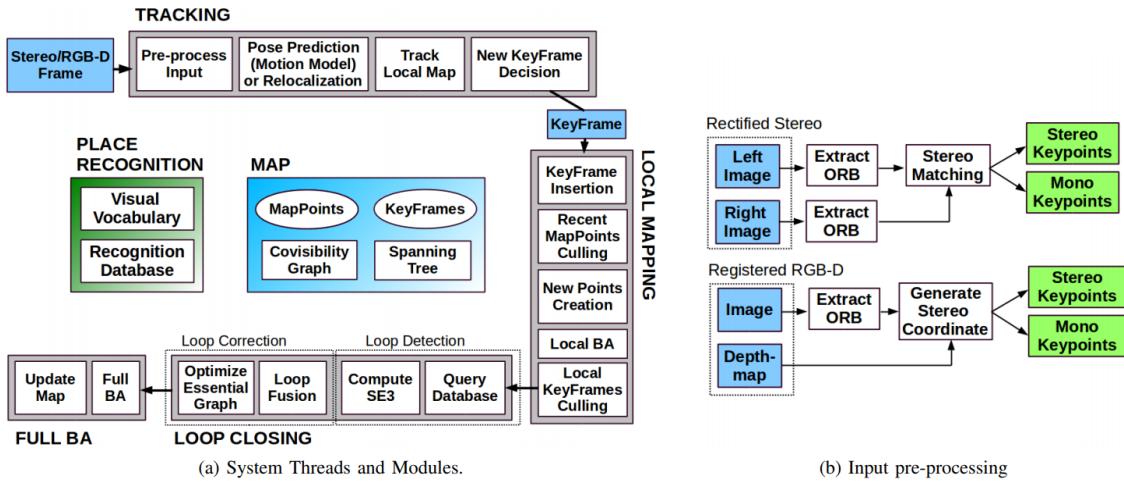


Рис. 10. Схема алгоритма ORB-SLAM. Источник [40]

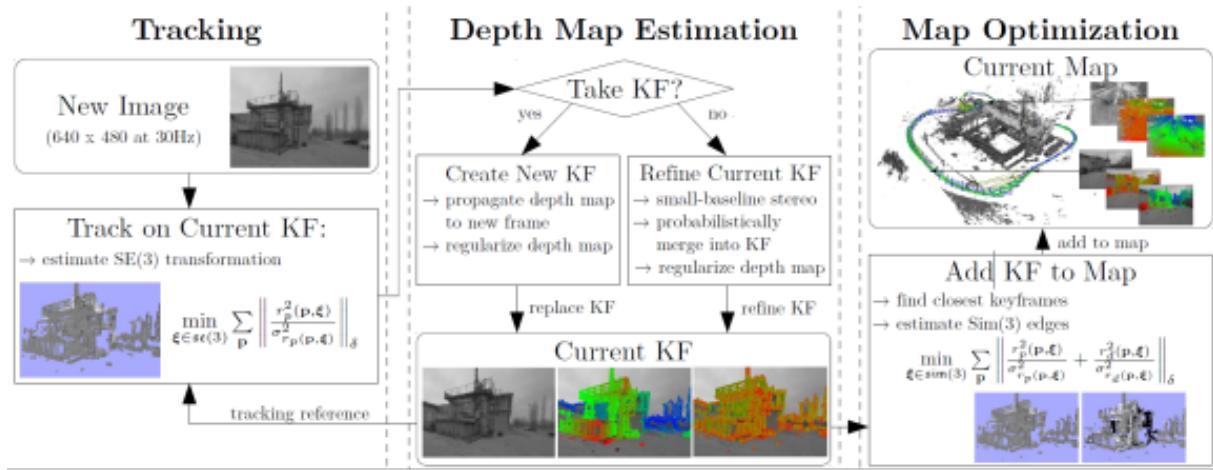


Рис. 11. Схема алгоритма LSD-SLAM. Источник [15]

строит более плотную карту, однако при этом он имеет более высокие требования к вычислительным ресурсам, чем ORB-SLAM. При работе алгоритма LSD-SLAM в реальном времени на маломощном вычислительном устройстве могут возникнуть проблемы. Недостатками данного алгоритма также являются высокие требования к калибровке камеры и неустойчивость к выбросам.

Алгоритм LSD-SLAM состоит из трех основных модулей: tracking, depth map estimation и map optimization. Схема алгоритма представлена на рисунке 11.

Модуль **tracking** определяет перемещение камеры, отслеживая входящие изображения и вычисляя преобразование подобия между ними. Для вычисления преобразования используется минимизация фотометрической ошибки между новым изображением и преобразованным текущим ключевым кадром.

Модуль **depth map estimation** сравнивает кадр с текущим ключевым кадром, а затем уточняет или полностью заменяет его. Для сравнения используется взвешенная сумма расстояний и углов поворота между кадрами. Если она больше некоторого порога, ключевой кадр заменяется новым. По ключевому кадру вычисляется инвертированная карта глубины путем сопоставления точек с предыдущим ключевым кад-

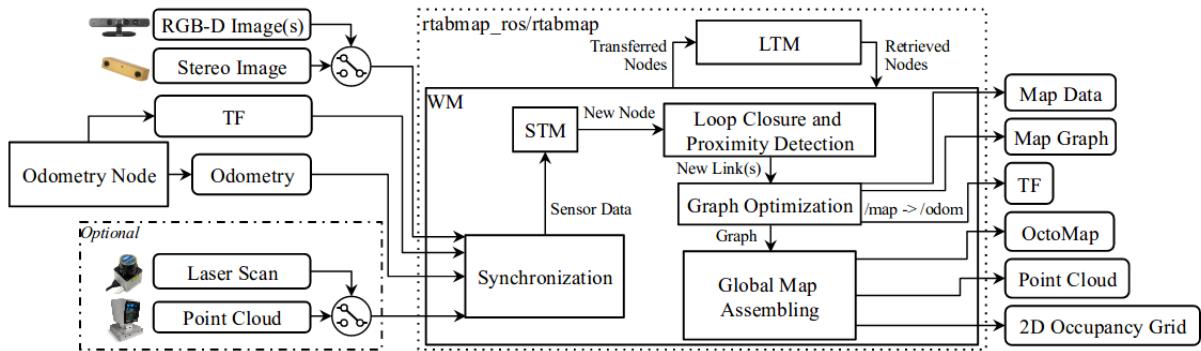


Рис. 12. Общая схема алгоритма RTAB-Мар. Источник [34]

ром с использованием преобразования подобия, вычисленного модулем tracking. По вычисленному местоположению камеры и инвертированным картам глубины строится карта окружающей местности.

Модуль **map optimization** выполняет оптимизацию карты с использованием библиотеки g2o [21]. Оптимизация позволяет предотвратить накопление ошибок вычисления траектории и поддерживает точность построения карты.

Для хранения карты окружающей среды используется граф. Каждый узел этого графа хранит соответствующий ключевой кадр и инвертированную карту его глубины. Узлы соединяются ребрами, содержащими преобразование подобия между кадрами.

3.1.3 RTAB-MAP

Алгоритм RTAB-MAP [34] предназначен для решения задачи SLAM с использованием информации о глубине изображений (по данным видеокамеры и лидара, или RGB-D камеры, или стереопары камер). Алгоритм использует три независимых процесса: вычисление движения камеры (одометрии), картирование и замыкание циклов. Схема алгоритма представлена на рисунке 12.

Для одометрии по кадрам вычисляются особые точки с помощью детектора BRIEF [8]. По сопоставлению особых точек на текущем и ключевом кадрах с помощью алгоритма PnP RANSAC [6] вычисляется перемещение камеры. Полученное положение камеры корректируется с помощью алгоритма Local Bundle Adjustment [62] и предсказаний на основе предыдущих движений камеры. Новый ключевой кадр добавляется, когда у текущего кадра и ключевого будет мало сопоставлений. Схема вычисления одометрии представлена на рисунке 13.

Картирование выполняется по локальным сеткам заполненности (occupancy grid), полученных из карт глубины. Локальные карты с помощью воксельного фильтра сшиваются в глобальную карту. При замыкании цикла карта перестраивается. Схема процесса построения карты по локальным облакам точек представлена на рисунке 14.

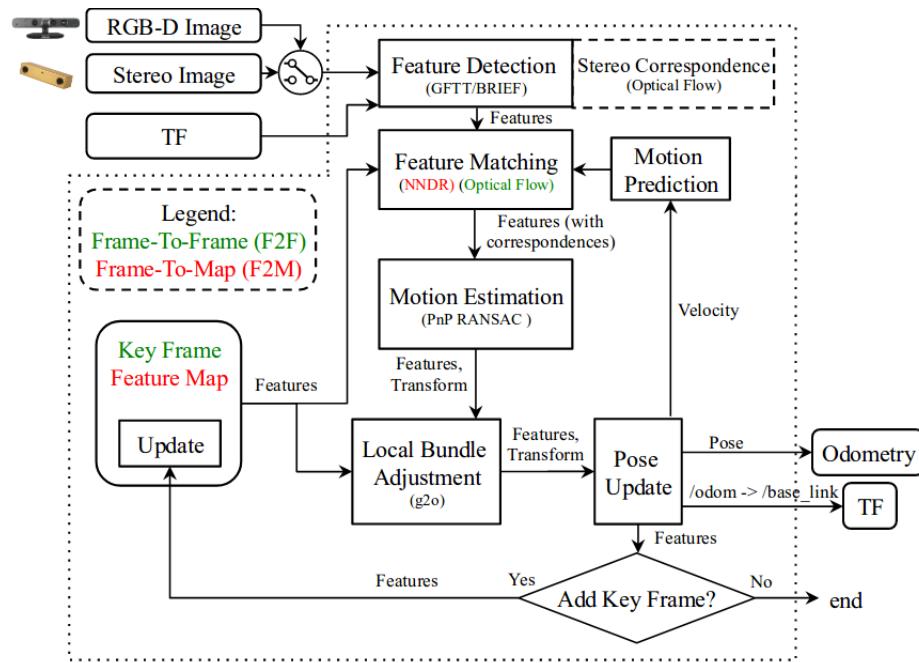


Рис. 13. Схема вычисления одометрии в методе RTAB-Мар. Источник [34]

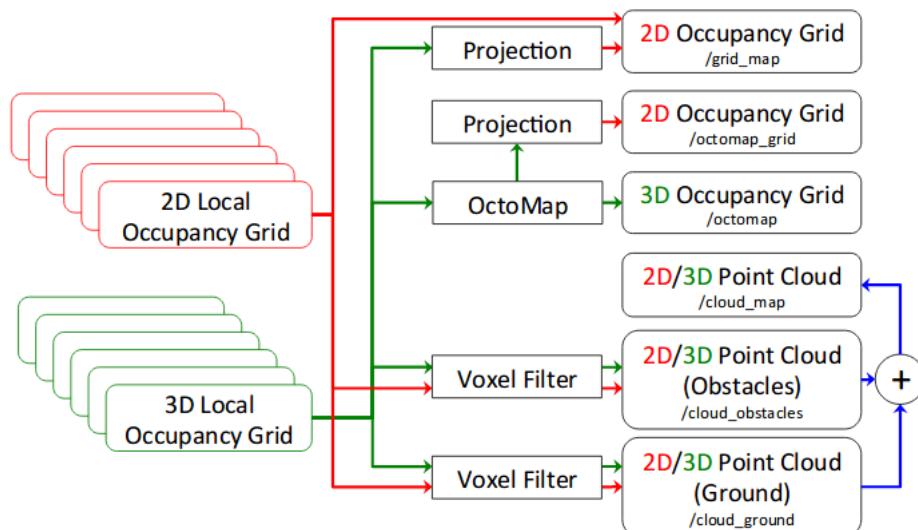


Рис. 14. Схема построения плотной глобальной карты по локальным сеткам. Источник [34]

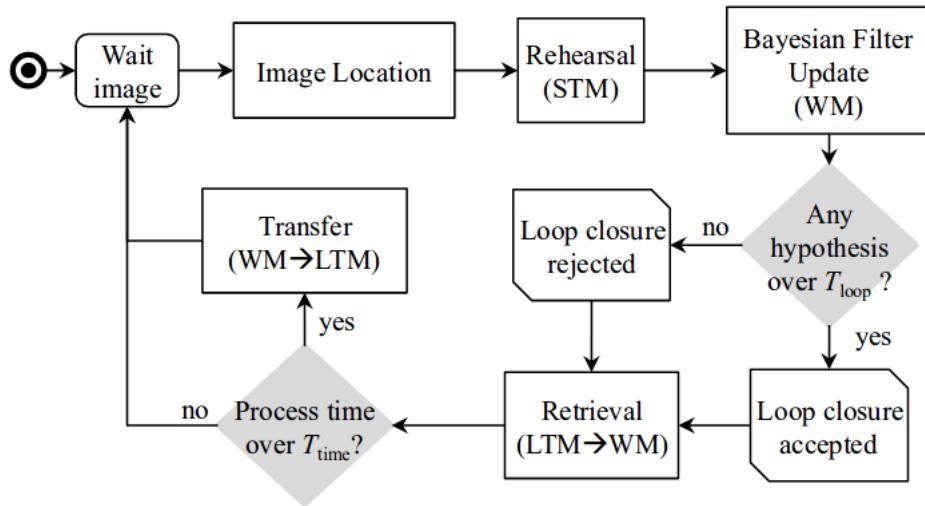


Рис. 15. Схема замыкания циклов в алгоритме RTAB-Мар. Источник [33]

Замыкание циклов основывается на сопоставлении особых точек на кадрах с видеопотока. Ключевая особенность данного метода - эффективное хранение изображений в памяти. Кадры хранятся в памяти как набор дескрипторов особых точек, организованный в kd-деревья. Дескрипторы извлекаются с помощью алгоритма SURF [1]. Алгоритм использует три вида памяти: WM (рабочая), в которой хранятся самые “полезные” кадры, STM (кратковременная), в которой хранятся последние кадры, и LTM (долгосрочная), в которой хранятся все кадры. Из STM в WM перемещаются те кадры, у которых больше всего похожих особых точек (похожесть мерится по дескрипторам). Для замыкания циклов используется кадр из рабочей памяти, который наиболее вероятно похож на текущий. Вероятности высчитываются байесовским фильтром. Схема процесса замыкания циклов представлена на рисунке 15.

Данный алгоритм имеет следующие преимущества в сравнении с другими методами SLAM:

1. Эффективная обработка данных с видеокамер и датчиков глубины в реальном времени
2. Эффективное замыкание циклов
3. Возможность работы в больших картах благодаря хранению долгосрочной памяти на жестком диске
4. Высокая плотность построенной карты и возможность построения карты препятствий в формате Octomap
5. Легкость использования в различных приложениях, а также большое количество настраиваемых параметров

Помимо преимуществ, алгоритм RTAB-Мар обладает существенными недостатками:

1. Невозможность работы в монокулярном режиме

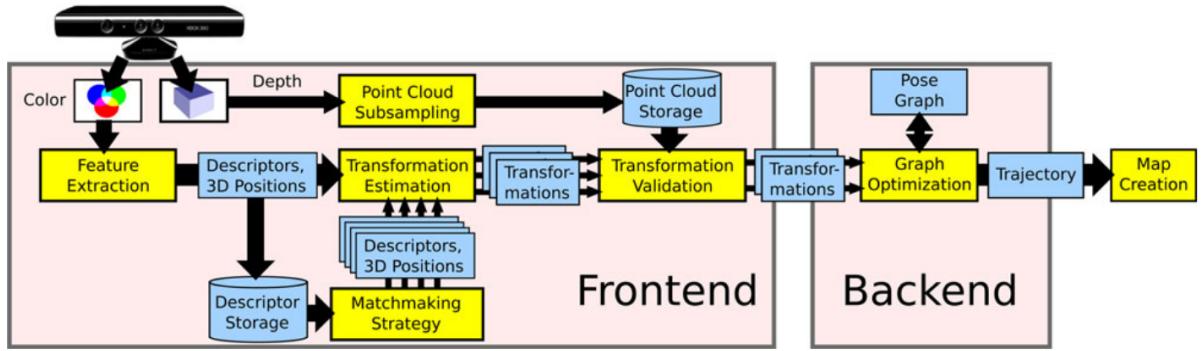


Рис. 16. Общая схема алгоритма RGBDSLAM_v2. Источник [14]

2. Потеря одометрии при отсутствии сопоставленных ориентиров
3. Высокая ресурсоемкость из-за необходимости обработки трехмерных облаков точек и построения плотной карты

3.1.4 RGBDSLAM_v2

Алгоритм RGBDSLAM_v2 [14] также является популярным решением задачи одновременного картирования и локализации по данным с RGB-D камеры. Он был разработан примерно в то же время, что и описанный выше алгоритм RTAB-MAP, и основан на схожих принципах. Однако в деталях методов имеются существенные различия. Схема алгоритма RGBDSLAM_v2 изображена на рисунке 16.

Вычисление перемещения камеры в алгоритме RGBDSLAM_v2 осуществляется по тем же принципам, что и в алгоритме RTAB-MAP - путем сопоставления особых точек на ключевых кадрах. Изображение с камеры добавляется в множество ключевых кадров, когда у него не будет совпадающих особых точек с предыдущим ключевым кадром. Для извлечения особых точек используются детекторы SIFT [38], SURF [1] или ORB [46]. Сопоставления уточняются и фильтруются с помощью методов RANSAC [6] и ICP [11]. Преобразования, вычисленные по сопоставлениям особых точек, валидируются по картам глубин вероятностными методами.

По ключевым кадрам и найденным преобразованиям строится граф поз. Вершинами в этом графе являются ключевые кадры, ребрами - вычисленные алгоритмом геометрические преобразования между кадрами. По данному графу проводится глобальная оптимизация с помощью библиотеки g2o [21].

Метод замыкания циклов в алгоритме RGBDSLAM_v2, так же, как и в RTAB-MAP, основан на вычислении преобразования по особым точкам между текущим кадром и похожими на него старыми ключевыми кадрами. Однако в алгоритме RGBDSLAM_v2 используется более простой отбор кандидатов на “похожесть”. По графу поз строится минимальное остовное дерево. В качестве кандидатов для рассмотрения выбираются n предков текущего кадра в этом дереве, а также k случайно выбранных ключевых кадров в части дерева, оставшейся после удаления этих n

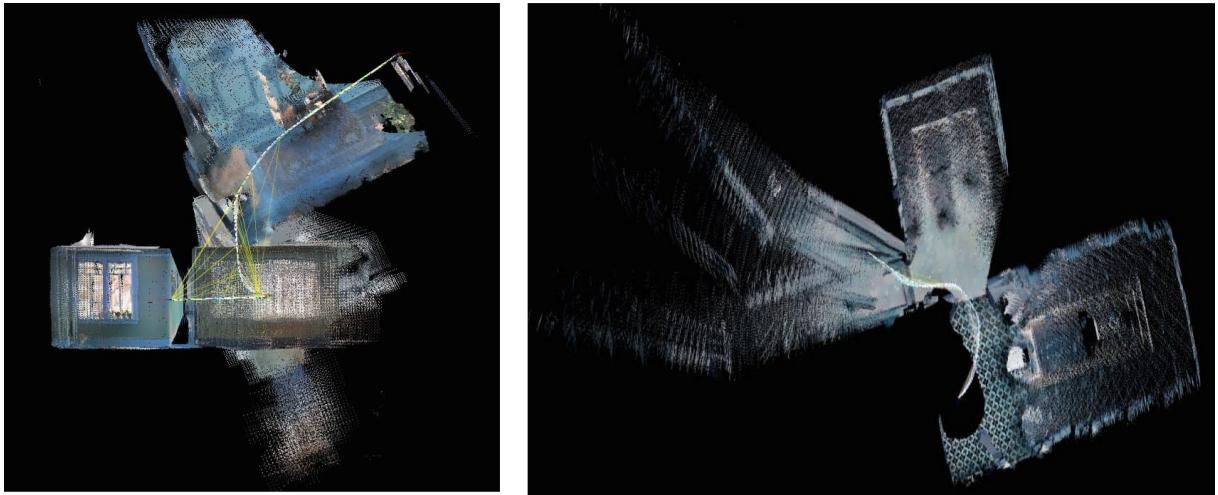


Рис. 17. Примеры некорректной работы алгоритма RGBDSLAM_v2: ложное замыкание цикла (слева) и раздвоение коридора (справа)

предков, и еще l ключевых кадров, случайно выбранных по всему графу. Для эффективного замыкания циклов на разных длинах траекторий необходимо использовать разные n, k, l .

Алгоритм RGBDSLAM_v2 обладает высокой вычислительной эффективностью, однако эксперименты, проведенные в работе [3], показали, что он обладает более низкой точностью по сравнению с алгоритмом RTAB-MAP. В частности, при резких поворотах робота RGBDSLAM_v2 может найти ложные замыкания или построить карту с раздвоением коридора (см. рис. 17).

3.2 Нейросетевые методы

В связи с бурным развитием вычислительной техники и нейронных сетей, в последние годы для решения задачи vSLAM также стали применяться методы, основанные на глубоком обучении. В подобных методах, как правило, вычисляются преобразования между позициями камеры с помощью сверточных нейронных сетей, также с помощью сверточных нейросетей вычисляются карты глубин по изображениям.

Одним из наиболее известных нейросетевых методов картирования и локализации является CNN-SLAM [53]. Как и в алгоритме LSD-SLAM [15], в CNN-SLAM из множества всех входящих кадров отбираются ключевые кадры. По ключевым кадрам строится граф позиций, который оптимизируется с помощью библиотеки g2o [21]. По каждому кадру вычисляется преобразование между этим кадром и текущим ключевым кадром путем минимизации фотометрической ошибки.

Метод CNN-SLAM принимает на вход видеопоток с камеры и по каждому изображению видеопотока вычисляет карту глубины с помощью полносверточной нейронной сети [35]. Также вместе с картой глубины вычисляется карта неопределенности, которая показывает, насколько глубина объектов, изображенных на текущем кадре,

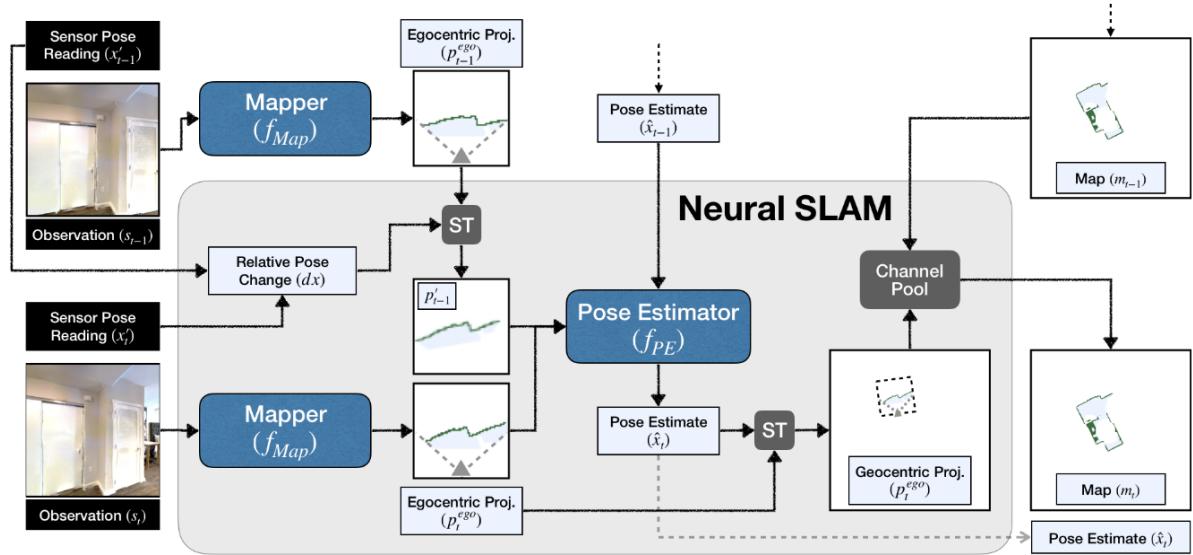


Рис. 18. Схема метода Active Neural SLAM. Источник [10]

соотносится с глубинами этих объектов на других кадрах. Глубина каждого кадра оптимизируется исходя из карты неопределенности и глубины текущего ключевого кадра.

Помимо карты глубины, по каждому кадру входящего видеопотока с помощью нейросети выполняется семантическая сегментация. По картам глубин и картам сегментации с помощью глобальной сегментационной модели (GSM) [52] строится итоговая карта окружающей местности.

Эксперименты на датасетах TUM [50] и ICL-NUIM [22] показали, что по точности определения местоположения алгоритм CNN-SLAM более чем в два раза превосходит классический метод LSD-SLAM [15]. Однако качество метода CNN-SLAM, как и любого метода, использующего предобученные нейросети, сильно зависит от качества обучающей выборки.

Одним из наиболее известных недавних нейросетевых методов vSLAM является Active Neural SLAM [10]. Данный метод решает задачу vSLAM в контексте более глобальной задачи - исследование неизвестной местности (Exploration). Помимо картирования и локализации, алгоритм выполняет постановку цели для исследования и планирование траектории.

Метод Active Neural SLAM принимает на вход видеопоток с камеры и данные одометрии с сенсоров робота. На выходе получается двумерная карта местности и траектория перемещения робота. В отличие от метода CNN-SLAM, где карта строится по предсказанным нейросетью глубинам с помощью различных методов оптимизации, в методе Active Neural SLAM локальные карты предсказываются непосредственно нейросетью по каждому изображению видеопотока. Перемещение робота также предсказывается нейросетью по локальным картам и входящим данным одометрии (как правило, сильно зашумленным). Схема алгоритма Active Neural SLAM изображена на рисунке 18.

Эксперименты, проведенные на датасете Gibson [58], показали, что метод Active Neural SLAM способен исследовать в среднем 95% помещения, решая задачу vSLAM совместно с задачей планирования маршрутов и исследования неизвестной местности. Однако оценка качества построения карты и вычисления траектории в работе [10] не производилась.

3.3 Выводы

В данной главе были рассмотрены классические и нейросетевые методы решения задачи vSLAM. Среди классических методов были рассмотрены следующие:

1. ORB-SLAM - метод, основанный на сопоставлении особых точек на изображениях
2. LSD-SLAM - метод, основанный на вычислении геометрических преобразований путем минимизации фотометрической ошибки
3. RTAB-MAP - метод, принимающий на вход данные со стереокамеры или RGB-D камеры, обладающий эффективным замыканием циклов
4. RGBDSLAM - метод, принимающий на вход данные с RGB-D камеры, с более простым замыканием циклов по сравнению с RTAB-MAP

Алгоритм ORB-SLAM обладает высокой вычислительной эффективностью, однако строит разреженную карту местности, не пригодную для планирования маршрутов. Метод LSD-SLAM строит довольно плотную карту, однако он более требователен к вычислительным ресурсам и имеет в среднем более низкую точность. Методы RTAB-MAP и RGBDSLAM строят плотную карту и имеют довольно высокую точность, однако они требуют на вход данные со стереокамеры или RGBD-камеры. Для их работы по данным с единственной камеры может применяться восстановление глубин изображений с помощью нейронных сетей.

Помимо классических методов, были также рассмотрены нейросетевые методы картирования и локализации:

1. CNN-SLAM - построение карты по предсказанным нейросетью глубинам и картам сегментации с помощью глобальных моделей
2. Active Neural SLAM - построение карты и вычисление траектории с помощью нейросетей непосредственно

На некоторых коллекциях данных методы CNN-SLAM и Active Neural SLAM имеют более высокое качество по сравнению с классическими методами. Однако их применение на реальных роботах затруднено, поскольку данные методы требуют наличие графического ускорителя на борту, и качество их работы напрямую зависит от выборки, на которой производилось обучение нейросетей.

В данной работе был выбран алгоритм RTAB-MAP, поскольку он строит плотную трехмерную карту местности и не требователен к вычислительным ресурсам. Для применения алгоритма на данных с единственной видеокамеры используется восстановление карт глубин с помощью полносверточных нейронных сетей с легкой архитектурой, способных работать с высокой скоростью в условиях ограниченных вычислительных ресурсов.

Глава 4

Восстановление глубины по видеопотоку

Классические методы решения задачи vSLAM по данным с монокулярной камеры [40] [15], описанные в предыдущей главе, выдают разреженную карту, непригодную для планирования маршрутов. Построение плотной карты возможно при наличии информации о глубине изображений, например, при помощи методов [33], [14]. Таким образом, для построения плотной карты окружающей местности по данным с единственной видеокамеры необходимо решить задачу восстановления глубины изображений. Схема метода vSLAM с использованием восстановления глубины изображений показана на рисунке 19.

В настоящее время задача восстановления глубины изображений решается, как правило, с помощью глубоких нейронных сетей. Нейросети, предсказывающие глубину изображений по видеопотоку, делятся на два типа. Первые принимают на вход одиночное изображение, не используя информацию о контексте из видео. Вторые предсказывают глубину изображений с использованием информации об изменении положений объектов на кадрах видеопотока. Ниже рассмотрены обе группы нейросетевых методов восстановления глубины.

4.1 Восстановление глубины по одиночным изображениям

В настоящее время существует большое количество нейросетевых архитектур для восстановления глубины по одиночным изображениям. Как правило, такие нейросети состоят из сверточного энкодера и декодера, состоящего из нескольких блоков, в которых используются слои свертки и повышения дискретизации (Upsampling).

Одной из наиболее известный нейросетей восстановления глубины является FCRN [35]. Архитектура данной сети состоит из энкодера и декодера. В качестве энкодера выбрана широко известная сеть ResNet [24], содержащая 50 сверточных слоев. В качестве декодера используется 5 блоков Up-Convolution или Up-Projection (см. рис. 20).

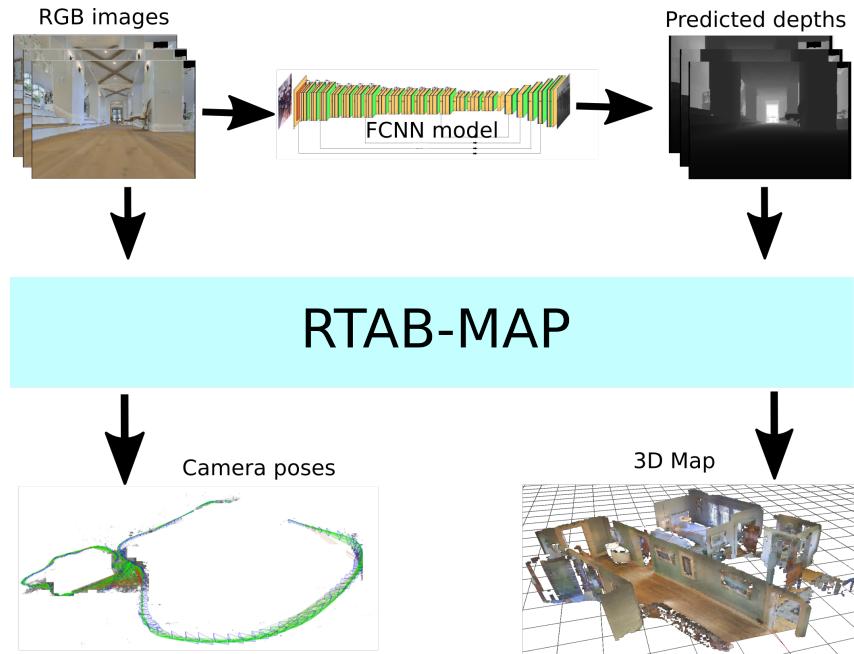


Рис. 19. Схема представленного в данной работе метода vSLAM. По изображениям с камеры при помощи нейронной сети вычисляются карты глубин, которые вместе с изображениями подаются на вход алгоритму RTAB-MАР. Алгоритм RTAB-МАР выдает плотную трехмерную карту окружающей местности и траекторию перемещения камеры

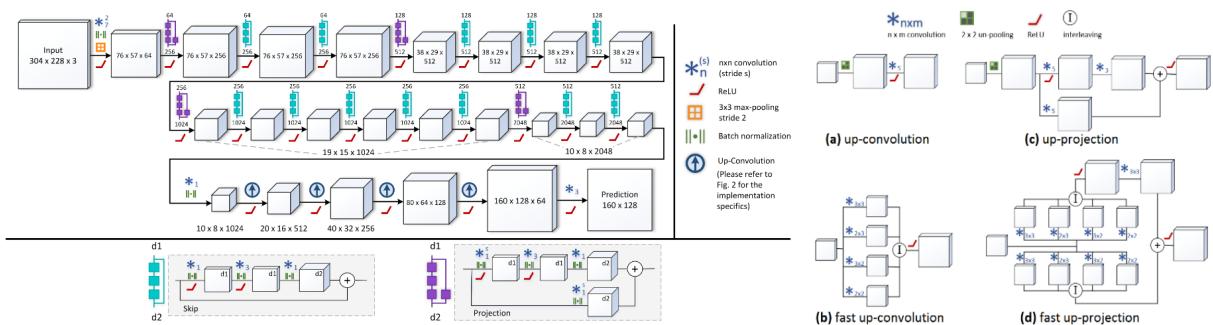


Рис. 20. Схема архитектуры FCRN (слева) и различных блоков декодера - Up-Projection и Up-Convolution (справа). Источник [35]

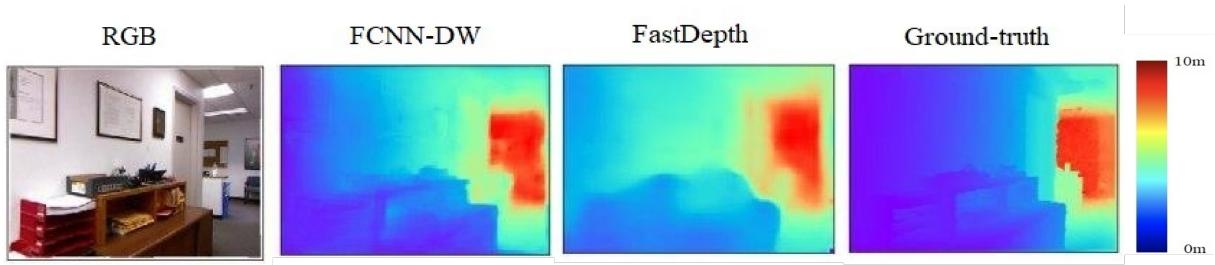


Рис. 21. Примеры глубин, предсказанных нейросетью FastDepth, в сравнении с глубинами, предсказанными нейросетью из данной работы (FCNN-DW)

Обучение сети FCRN производилось на датасете NYU Depth v2 [49]. Исследовались разные функции потерь - стандартная функция MSE, а также функции Huber и BerHu [43] [64], являющиеся комбинацией квадратичной и линейной ошибки. Наилучший результат показала архитектура с декодером Up-Projection, обученная с функцией потерь BerHu. Относительная ошибка этой архитектуры на валидационной выборке NYU Depth v2 составила 0.127. Была также измерена скорость работы данной архитектуры на видеокарте GTX Titan. Она составила 78 мс на одно изображение, или 13 кадров в секунду, что недостаточно для работы в реальном времени (для обработки всех кадров стандартного видеопотока необходима скорость не менее 30 кадров в секунду).

В 2020 году была предложена нейросетевая архитектура восстановления глубины с более высокой скоростью и качеством работы [26]. В этой архитектуре предсказание глубины уточняется с помощью механизма внимания, установленного между энкодером и декодером. Механизм внимания предсказывает, насколько глубина одного участка изображения может быть полезна для предсказания глубины другого участка, формируя карты внимания глубины (depth-attention maps). Для построения истинных карт внимания, необходимых для обучения сети, по изображеному на кадре участку сцены вычисляется несколько плоскостей. Значением внимания глубины одного пикселя для другого является максимальная сумма расстояний точек, спроецированных в эти пиксели, до плоскостей. Такой метод обосновывается тем, что у точек, расположенных в одной плоскости, легко вычислить глубину с помощью геометрических преобразований, если известна точная глубина хотя бы одной из этих точек.

С помощью вышеописанной архитектуры и механизма внимания была достигнута относительная ошибка 0.108 на датасете NYU Depth v2. Скорость работы данной архитектуры на видеокарте GTX 1080 составила 218 кадров в секунду. Однако на бортовых вычислителях робототехнических систем, не обладающих мощными видеокартами, скорость работы нейросети будет значительно меньше. Точно измерить скорость работы описанной нейросети на бортовых вычислителях не удалось, поскольку исходный код описанной архитектуры на момент написания данной работы предоставлен не был.

В работе [57] описана архитектура FastDepth, пригодная для работы в реальном

времени даже на маломощных вычислительных устройствах. С помощью различных технических и архитектурных оптимизаций авторам работы удалось добиться скорости восстановления глубины 175 кадров в секунду на встраиваемом компьютере NVidia Jetson TX2, который широко используется в различных робототехнических системах. Ошибка восстановления глубины на NYU Depth v2 при этом составила 0.158 - почти в полтора раза выше, чем у архитектуры из работы [26]. Контуры объектов на предсказанных картах глубины также получились размытыми (см. рис. 21), что может негативно сказаться на качестве карты при применении предсказанных глубин в задаче vSLAM.

В данной работе с учетом преимуществ и недостатков всех вышеописанных методов была разработана новая нейросетевая архитектура для достижения баланса между скоростью и качеством восстановления глубины. В основе разработанной архитектуры лежит классическая схема энкодер-декодер. В качестве энкодера выбрана сеть ResNet с 50 слоями, как и в архитектуре FCRN. В качестве декодера была выбрана серия поканальных сверток (depthwise convolution), как и в архитектуре FastDepth. Также для повышения качества работы и четкости контуров были добавлены проекции из энкодера в декодер (shortcuts). С помощью предложенной архитектуры удалось достичь приемлемого качества восстановления глубины (относительная ошибка 0.170 на NYU Depth v2) и вместе с тем приемлемой скорости работы (24 кадра в секунду на NVidia Jetson TX2). Примеры глубин, предсказанных разработанной нейросетью и архитектурой FastDepth, показаны на рисунке 21. Глубина, предсказанная нейросетью из данной работы, имеет значительно более четкие контуры, чем глубина, предсказанная нейросетью FastDepth.

4.2 Восстановление глубины с извлечением информации из видеопотока

При наличии видеопотока можно получить дополнительную информацию о глубине изображений из перемещения объектов на кадрах. Например, в методе LSD-SLAM [15] карты глубин изображений вычисляются с использованием преобразований подобия, задающих перемещение камеры от кадра к кадру и вычисляемых путем минимизации фотометрической ошибки. С помощью глубоких нейронных сетей также можно извлекать временную информацию из видеопотока, тем самым уточняя карты глубин каждого изображения.

В работе [60] описана архитектура нейронной сети ST-CLSTM, в которой временная информация извлекается из видеопотока с помощью сверточно-рекуррентных блоков ConvLSTM. Модуль ConvLSTM представляет собой комбинацию слоев, используемую в традиционном модуле LSTM [20], в которой полно связные слои заменены на сверточные.

Архитектура ST-CLSTM состоит из двух частей. Её схема изображена на рисун-

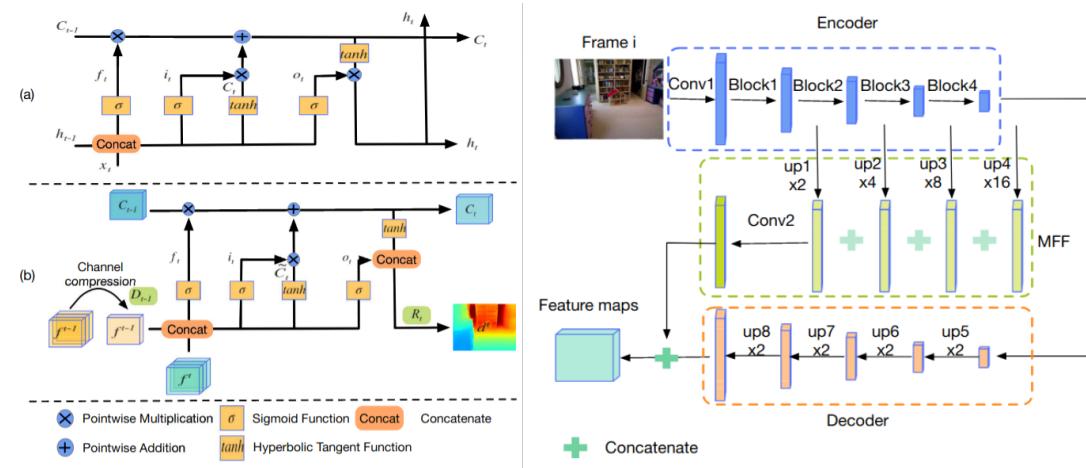


Рис. 22. Схема архитектуры ST-CLSTM: часть, кодирующая пространственные признаки (справа), и модуль Conv-LSTM (слева). Источник [60]

ке 22. Первая часть имеет схему энкодер-декодер и по входящему изображению вычисляет набор пространственных признаков. В качестве энкодера используется сеть ResNet-18 [24]. Декодер представляет собой четыре блока, состоящих из последовательного повышения дискретизации и двух параллельных сверток. Для повышения информативности вычисляемых признаков в архитектуру первой части встроен также модуль слияния признаков с несколькими масштабами (Multi-scale feature fusion, MFF). Модуль MFF представляет собой конкатенацию выходов четырех блоков энкодера, приведенных к размеру исходного изображения с помощью слоев повышения дискретизации (Upsampling). Данная конкатенация преобразуется с помощью сверточного слоя и стыкуется с выходом декодера, в результате чего получается карта пространственных признаков.

Вторая часть архитектуры ST-CLSTM состоит из блока ConvLSTM и двух сверточных слоев. Слои блока ConvLSTM используют информацию, извлеченную из карт пространственных признаков предыдущих изображений. Сверточные слои преобразуют данную информацию в карту глубины.

Для обучения сети ST-CLSTM использовалась комбинация пространственной и временной функции потерь. Пространственная функция потерь представляла собой среднюю логарифмическую ошибку между предсказанной и истинной глубиной. Для эффективного обучения временной составляющей сети ST-CLSTM применялся генеративно-состязательный подход. В качестве генератора выступала непосредственно сеть ST-CLSTM, а в качестве дискриминатора - трехмерная сверточная нейронная сеть, принимающая на вход последовательность карт глубин и предсказывающая происхождение этих глубин (истинные или предсказанные нейросетью-генератором).

Эксперименты, проведенные на датасете NYU Depth v2 [49], показали, что с помощью архитектуры ST-CLSTM удалось добиться довольно высокого качества восстановления глубины. Относительная ошибка составила 0.132. В ходе экспериментов

была также измерена скорость обработки одного изображения на видеокарте GTX 1080Ti. Скорость работы оказалась равной 33 кадрам в секунду, что достаточно для обработки видеопотока в реальном времени. Однако на менее мощной видеокарте, в частности, на бортовых вычислителях большинства робототехнических систем, данная архитектура будет работать значительно медленнее, что затруднит ее использование в реальном времени. Например, скорость работы на встраиваемом компьютере NVidia Jetson TX2 составила менее 10 кадров в секунду.

В работе [39] для предсказания глубины изображений используется традиционная полносверточная нейросеть, а предсказанные нейросетью глубины дополнительно уточняются с помощью оптического потока и метода COLMAP [48]. Оптический поток вычисляется нейронной сетью FlowNet2 [27].

С помощью вычисленного оптического потока нейросеть восстановления глубины может дообучаться на конкретном видео без данных об истинных глубинах изображений. По предсказанным нейросетью глубинам и перемещениям пикселей изображений (оптического потока) вычисляется временная консистентность, которая служит в данном случае функцией потерь при обучении.

Коррекция предсказанных глубин методом COLMAP позволила существенно повысить качество восстановления глубины в тех случаях, когда тестовая выборка значительно отличается от обучающей. Так, на датасете ScanNet [12] относительная ошибка предсказания глубины составила 0.073 (против 0.208 у полносверточной нейронной сети без коррекции). Однако данный подход малопригоден для применения в реальном времени на борту робота - нейросеть FlowNet, вычисляющая оптический поток, обрабатывает одну пару кадров более 100 мс на видеокарте компьютера NVidia Jetson TX2; а также метод COLMAP потребляет значительную часть ресурсов процессора, что затрудняет работу алгоритмов SLAM.

В работах [63], [59], [54] совместно с задачей восстановления глубины решается также задача визуальной одометрии - вычисления перемещения камеры по видеоданным с нее. Для вычисления карт глубин и перемещения камеры используются отдельные нейронные сети, которые обучаются совместно с использованием общей функции потерь, основанной на пространственно-временной консистентности. Таким образом, возможно обучение нейросетей при отсутствии данных об истинной глубине изображений, на неразмеченном видеопотоке.

Совместное использование нейросетей глубины и позиции позволило достичь очень низкой ошибки восстановления глубины. Так, в работе [54] относительная ошибка на датасете NYU Depth v2 [49] составила 0.061 (по сравнению с ошибкой в 0.106 у классических полносверточных нейросетей). Однако параллельная работа двух нейронных сетей в реальном времени на бортовом вычислителе затруднительна. Так, в работе [54] время обработки одного кадра составляет 690 мс (при том, что в стандартном видеопотоке кадры приходят каждые 33 мс).

4.3 Выводы

Восстановление глубины изображений дает возможность строить плотную карту местности по данным с единственной видеокамеры с помощью методов vSLAM. В данной главе приведен обзор основных методов восстановления глубины по изображениям. Методы, использующие в качестве входных данных одиночное изображение ([35], [57]) обладают достаточно высокой скоростью работы для обработки стандартного видеопотока в реальном времени, однако имеют довольно низкое качество (относительная ошибка порядка 13-17%). Методы, принимающие во внимание информацию о перемещении между кадрами ([60] [39] [54]), имеют значительно более высокое качество (относительная ошибка восстановления глубины порядка 6-10%), однако обладают низкой скоростью работы, что затрудняет их применение на борту робототехнической системы в реальном времени.

В данной работе для решения задачи VSLAM была выбрана полносверточная нейронная сеть, принимающая на вход одиночные изображения. Архитектура нейронной сети была оптимизирована для работы в реальном времени на встраиваемом компьютере NVidia Jetson TX2, который может применяться в качестве бортового вычислителя на малых робототехнических системах. Подробное описание данной архитектуры приведено в главе 5.

Глава 5

Предлагаемый метод одновременного картирования и локализации

5.1 Описание метода

В данной работе представлен алгоритм одновременного картирования и локализации по видеопотоку с единственной камеры, основанный на восстановлении карт глубин изображений. По изображениям, поступающим с видеокамеры, вычисляются карты глубин с помощью полносверточной нейронной сети. По изображениям и предсказанным картам глубин осуществляется картирование и локализация с помощью метода RTAB-Мар, описанного в разделе 3.1.3. Схема алгоритма представлена на рисунке 19.

Нейросеть, используемая для восстановления глубины, имеет полносверточную архитектуру, состоящую из энкодера и декодера. Сеть принимает на вход трехканальное цветное изображение размера 320x240 и выдает карту глубины такого же размера. В качестве энкодера используется сеть ResNet-50 [24] без полносвязных слоев, предобученная на датасете MS Coco [37]. На выходе энкодера получается карта высокоровневых признаков размерности 10x8x2048. Декодер состоит из пяти блоков:

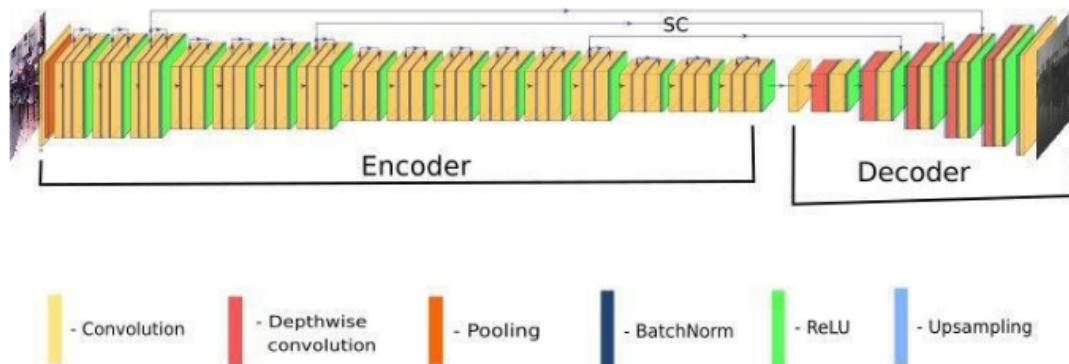


Рис. 23. Схема нейронной сети восстановления глубины

Таблица 1. Оценка качества восстановления глубины и сравнительный анализ различных нейросетей

Архитектура	NYU Dataset v2		RealSense			FPS
	RMSE	REL	RMSE	REL	δ^1	
FastDepth [57]	0.594	0.170	1.315	0.292	0.467	55
FCNN-DW (ours)	0.696	0.158	1.179	0.257	0.441	24

ков, каждый из которых повышает пространственную размерность в два раза. Каждый блок состоит из слоя повышения дискретизации (Upsampling), свертки с ядром 1x1 (pointwise convolution) и поканальной свертки (depthwise convolution) с ядром 3x3. Схема архитектуры нейросети представлена на рисунке 23.

Обучение нейросети восстановления глубины проводилось на датасете NYU Depth v2 [49]. Датасет содержит 464 видеопоследовательности, снятые в разных помещениях с размеченными картами глубин. Суммарно во всех последовательностях содержится более 400000 пар изображение-глубина. Для обучения была отобрана выборка из 10% (порядка 40000) изображений датасета. Из них около 30000 изображений составляла обучающая выборка и около 10000 - валидационная. В обучающей и валидационной выборке использовались изображения из разных сцен.

Оптимизация параметров проводилась методом Adam [28] с параметрами $\beta_1 = 0.9$, $\beta_2 = 0.999$ в течение 30 эпох. В качестве функции потерь использовалась комбинация среднеквадратичной и относительной квадратичной ошибки между истинной и предсказанной глубиной:

$$L(D, \hat{D}) = \frac{1}{H \cdot W} \left(\sum_{i,j} (D_{i,j} - \hat{D}_{i,j})^2 + \alpha \sum_{i,j} \left(\frac{D_{i,j} - \hat{D}_{i,j}}{D_{i,j}} \right)^2 \right), \quad (11)$$

где $D_{i,j}, i = 1, \dots, H; j = 1, \dots, W$ - истинные значения глубин, а $\hat{D}_{i,j}, i = 1, \dots, H; j = 1, \dots, W$ - предсказанные значения глубин.

После обучения была проведена оценка качества нейросети на официальной тестовой выборке датасета NYU Depth v2, содержащей 1449 изображений с высокоточной глубиной с качественной пост-обработкой. Также для проверки устойчивости нейросети к изменениям выборки была проведена оценка качества на наборе данных, собранных с RGB-D камеры Intel Realsense. Набор содержал приблизительно 9500 пар изображение-глубина.

Для оценки качества вычислялись три основные метрики: среднеквадратичная ошибка (RMSE), относительная ошибка (REL), и также метрика δ_1 - доля пикселей, на которых относительная ошибка глубины не превысила 0.25. Помимо метрик качества, также была измерена скорость обработки изображений на бортовом компьютере NVidia Jetson TX2. Результаты сравнения представлены в таблице 1.

Эксперименты показали, что представленная в данной работе нейросеть имеет приемлемое качество восстановления глубины, сравнимое с другими современными архитектурами, и при этом обладает достаточной эффективностью для работы в реальном времени на встраиваемом компьютере (при скорости в 24 кадра в секунду сеть способна обрабатывать почти каждый кадр входящего видеопотока). Данная сеть показывает схожие значения метрик качества с архитектурой FastDepth [57], однако предсказанные глубины у нее получаются значительно более четкими (см. рис. 21). Также эксперименты показали, что данная нейросеть обладает более высокой обобщающей способностью, чем FastDepth (качество восстановления глубины на данных с RealSense оказалось лучше по всем метрикам).

Помимо оценки качества восстановления глубины, была также проведена оценка качества работы метода vSLAM с разными нейросетевыми архитектурами восстановления глубины. Результаты оценки качества vSLAM описаны ниже.

5.2 Программно-аппаратное обеспечение

Архитектура нейросети была построена с использованием библиотеки глубинного обучения TensorFlow и ее расширения Keras. Программный код построения архитектуры был реализован на языке Python. Обучение нейросетей проводилось на гибридном высокопроизводительном вычислительном кластере ФИЦ ИУ РАН¹.

Обученные на кластере ФИЦ ИУ РАН архитектуры тестировались в реальном времени на встраиваемом компьютере NVIDIA Jetson TX2 [16]. Данный компьютер имеет размеры 50x87 мм, а его энергопотребление составляет 10-13 Вт на максимальной тактовой частоте, что позволяет встраивать его в малогабаритные робототехнические системы, в том числе в малые беспилотные летательные аппараты. Высокая скорость работы нейронных сетей на NVIDIA Jetson достигается за счет поддержки библиотек CuDNN и TensorRT², а также аппаратной поддержки вычислений с половинной точностью (fp16).

Для эффективной работы нейросети на встраиваемом компьютере она была переведена в формат TensorRT engine с поддержкой вычислений с половинной точностью. Код, осуществляющий обработку полученных с камеры изображений и визуализацию предсказанной карты глубины в реальном времени, был реализован на языке C++ с использованием технологии CUDA. Изображение с камеры захватывалось с помощью библиотеки Gstreamer и записывалось в видеопамять с помощью CUDA. Затем изображение переводилось в формат RGBA и нормализовалось для подачи на вход нейросети. Конвертация в RGBA и нормализация были распараллелены с помощью CUDA. Далее нормализованное изображение подавалось на вход нейросети для восстановления карты глубины. Восстановленная нейросетью карта глубины и

¹Федеральный исследовательский центр Информатика и управление РАН [Электронный ресурс]: сайт. – Москва: ФИЦ ИУ РАН. – URL: <http://hpc.fccsc.ru> (дата обращения: 28.02.2020)

²<https://developer.nvidia.com/tensorrt>

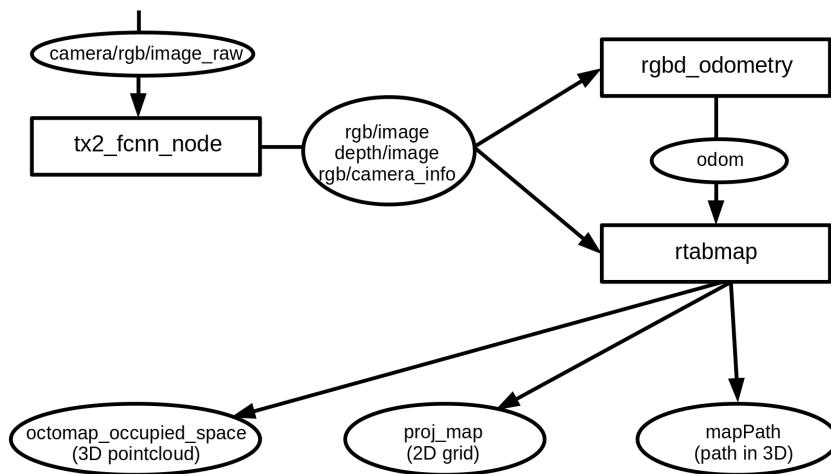


Рис. 24. Схема ROS-узлов представленного алгоритма vSLAM. Узел `tx2_fcnn_node` принимает на вход изображения с камеры и дает на выходе изображения, синхронизированные с предсказанными нейросетью картами глубин. Узел `rgbd_odometry` принимает на вход изображения и глубины и выдает одометрию. Узел `rtabmap` принимает изображения, глубины и одометрию, и выдает карту в форматах 2D и 3D, а также трехмерную траекторию



Рис. 25. Роботехническая платформа МПРМ с установленным на ней компьютером NVidia Jetson TX2

исходное изображение отрисовывались на экране с помощью библиотеки OpenGL.

Для взаимодействия нейросети восстановления глубины с алгоритмом RTAB-MAP, а также для облегчения запуска на робототехнической системе была использована библиотека Robot Operating System (ROS)[31]. Исходный код модуля восстановления глубины в формате ROS-узла, а также инструкции по установке и запуску доступны в репозитории³. Схема ROS-узлов разработанного алгоритма vSLAM представлена на рисунке 24.

Для оценки качества, эффективности и стабильности разработанного алгоритма vSLAM было проведено два вида экспериментов - в симуляционной среде и на реальном роботе. Для симуляционных экспериментов использовался компьютер со следующими характеристиками:

- Процессор: Intel Core i5-9600K, 6 ядер
- Видеокарта: GeForce RTX 2060, 1920 CUDA-ядер

³https://github.com/cnndepth/tx2_fcnn_node

- Оперативная память: 32 Гб
- Видеопамять: 6 Гб
- OC: Ubuntu 18.04

В качестве реального робота использовалась колесная платформа МПРМ (см. рис. 25) с установленным на борту компьютером NVidia Jetson TX2 со следующими характеристиками:

- Процессор: ARM Cortex, 4 ядра + NVidia Denver, 2 ядра
- Видеокарта: NVidia Pascal GPU, 256 CUDA-ядер
- Оперативная память: 8 Гб
- Видеопамять: общая с оперативной памятью
- OC: Ubuntu 18.04

Подробное описание экспериментов в симуляции и на реальном роботе предоставлено ниже.

5.3 Эксперименты

5.3.1 Эксперименты в симуляторе

Для оценки качества предложенного алгоритма vSLAM было проведено масштабное экспериментальное исследование в симуляторе Habitat [47] на датасете из работы [4]. На каждой из 40 видеопоследовательностей датасета был запущен vSLAM в режиме реального времени. Построенные vSLAM карты и траектории затем сравнивались с истинными для оценки качества. Оценка качества производилась по метрике ATE 1, а также по метрике AME 7 с сопоставлением точек по методу ближайшего соседа (AME) и по ракурсам, как описано в разделе 2.2.2 (AME ours). Значения метрик приведены в таблице 2.

В результате экспериментов выяснилось, что представленный алгоритм vSLAM способен строить правдоподобную карту помещений. Средняя ошибка траектории составила 1.68м, а средняя ошибка картирования - 0.39м по метрике AME standard и 2.23м по метрике AME ours. Такие большие значения ошибки получились из-за некорректного определения масштаба, что связано с различием в параметрах камеры, с которой записана обучающая выборка, и камеры в симуляторе. После коррекции масштаба ошибка значительно уменьшилась по всем метрикам. Примеры карт и траекторий, построенных методом vSLAM, показаны на рисунке 26.

sample	ATE	AME	AME ours	sATE	sAME	sOAME
1_first	3.193	0.449	3.517	0.568	0.277	1.310
1_second	2.482	0.396	3.119	0.122	0.298	1.299
2_first	2.310	0.414	3.065	0.679	0.259	1.765
2_second	2.855	0.470	3.567	0.221	0.179	1.181
3_first	1.294	0.364	1.885	0.233	0.148	0.871
3_second	1.042	0.415	1.589	0.224	0.252	1.045
4_first	2.068	0.688	3.074	0.356	0.226	1.471
4_second	2.441	0.474	2.997	0.691	0.246	1.474
5_first	2.940	0.329	3.281	0.338	0.186	0.867
5_second	1.598	0.371	1.938	0.146	0.261	0.765
6_first	1.907	0.359	2.564	0.285	0.264	1.410
6_second	2.766	0.362	3.438	0.256	0.197	1.017
7_first	0.461	0.381	1.194	0.089	0.374	1.262
7_second	0.976	0.373	1.358	0.290	0.219	0.736
8_first	1.462	0.400	2.200	0.160	0.192	1.022
8_second	0.877	0.473	1.607	0.162	0.228	0.659
9_first	1.459	0.321	1.627	0.296	0.432	1.099
9_second	0.162	0.174	0.438	0.162	0.174	0.438
10_first	1.072	0.281	1.342	0.182	0.188	0.671
10_second	0.908	0.307	1.395	0.461	0.426	1.195
11_first	1.331	0.292	2.060	1.027	0.402	1.703
11_second	2.740	0.314	3.212	0.367	0.260	1.409
12_first	3.602	0.444	4.315	0.315	0.205	1.145
12_second	2.479	0.633	3.457	0.237	0.274	1.398
13_first	3.114	0.702	3.568	0.421	0.178	0.838
13_second	2.633	0.305	3.030	0.504	0.242	1.291
14_first	1.843	0.304	2.643	0.182	0.208	0.700
14_second	1.418	0.543	2.093	0.318	0.155	0.818
15_first	1.369	0.473	2.010	0.221	0.211	0.858
15_second	0.688	0.314	0.952	0.356	0.274	0.865
16_first	2.128	0.403	2.701	0.260	0.219	1.073
16_second	2.265	0.447	2.719	0.442	0.243	1.358
17_first	1.054	0.336	1.503	0.203	0.157	0.663
17_second	0.525	0.273	1.061	0.061	0.190	0.666
18_first	0.768	0.281	1.215	0.214	0.172	0.698
18_second	0.991	0.316	1.395	0.375	0.356	0.987
19_first	0.612	0.356	1.362	0.218	0.239	1.072
19_second	0.711	0.326	1.095	0.219	0.564	0.736
20_first	1.672	0.297	2.021	0.230	0.208	0.692
20_second	1.094	0.365	1.637	0.287	0.247	1.007
Average	1.683	0.388	2.231	0.309	0.251	1.037

Таблица 2. Результаты экспериментов с алгоритмом vSLAM на симуляционных данных. ATE, AME, AME ours - значения соответствующих метрик на исходных данных. sATE, sAME, sOAME - значения метрик ATE, AME standard, AME ours после коррекции масштаба

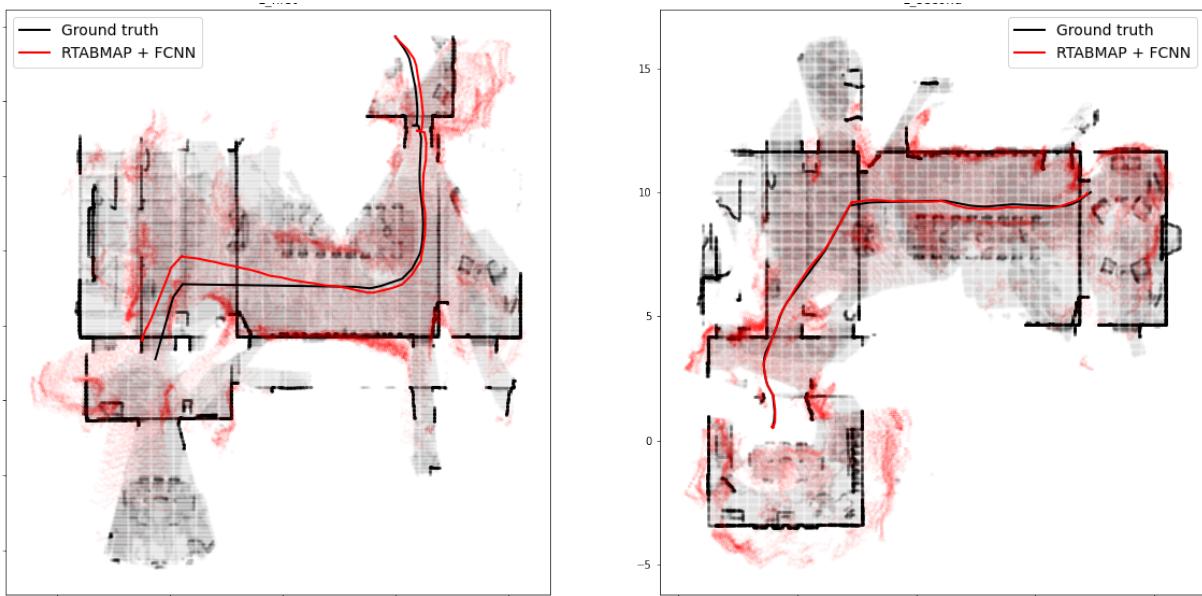


Рис. 26. Примеры карт, построенных методом vSLAM с нейросетевым восстановлением глубины. Черным показаны истинная карта и траектория, красным - восстановленная методом vSLAM

5.3.2 Эксперименты на реальном роботе

Помимо экспериментов в симуляционной среде, было также проведено тестирование разработанного алгоритма vSLAM в реальном времени на колесной робототехнической платформе МПРМ с бортовым компьютером NVidia Jetson TX2. Восстановление глубины входящих изображений с помощью нейросети производилось на графическом процессоре бортового компьютера, а алгоритм RTAB-MAP выполнялся на центральном процессоре. Таким образом, модули восстановления глубины и картирования работали параллельно, не отнимая ресурсы друг друга.

Эксперимент проводился на полигоне робототехнического центра ФИЦ ИУ РАН. Робот двигался по замкнутой траектории длиной около 40м. Управление движением робота осуществлялось дистанционно через контроллер Raspberry Pi.

По итогам эксперимента, с помощью предложенного алгоритма vSLAM удалось построить правдоподобную карту той части полигона, по которой двигался робот. В начале траектории произошла ошибка определения угла поворота, однако в конце траектории, при успешном замыкании цикла в методе RTAB-MAP, данная ошибка была устранена. Видео с демонстрацией эксперимента доступно по ссылке⁴.

5.4 Выводы

В данной работе был предложен метод одновременного картирования и локализации по данным с единственной видеокамеры, основанный на восстановлении глубин изображений с помощью нейросети и алгоритме RTAB-MAP. Для оценки качества и

⁴<https://drive.google.com/file/d/1eTrIA7D91SNjfQrG0LbKaEpmi511zcAh/view>

эффективности предложенного метода были проведены эксперименты в симуляционной среде и на реальном роботе. Эксперименты показали, что данный метод способен работать в реальном времени в условиях ограниченных вычислительных ресурсов и строить плотную трехмерную правдоподобную карту при движении внутри помещения. Высокие значения ошибки на данных из симулятора обусловлены главным образом некорректным определением масштаба, что может быть скорректировано путем изменения обучающей выборки для нейронной сети.

Глава 6

Применение в задаче исследования неизвестной местности

6.1 Описание задачи исследования неизвестной местности

6.1.1 Постановка задачи

Рассматриваемая проблема исследования неизвестной местности (ИНМ) описывается следующим образом. Робот, оснащенный только визуальными датчиками (момокулярная, стерео или RGB-D камера), находится в неизвестной среде ограниченной площади (обычно в закрытом помещении). Его задача - построить двумерную карту окружающей среды, перемещаясь по ней.

На каждом шаге t робот получает наблюдение I_t - изображение с его камеры. Используя это наблюдение, алгоритм ИНМ отслеживает его местоположение, картирует информацию, полученную в результате наблюдения, и решает, куда двигаться, чтобы исследовать и нанести на карту новое пространство. Результатом работы алгоритма A на шаге t является M_t - карта исследованной части окружения и действие a_t - намерение двигаться в неком направлении (см. рис. 27):

$$A(I_t, M_{t-1}) = (M_t, a_t)$$

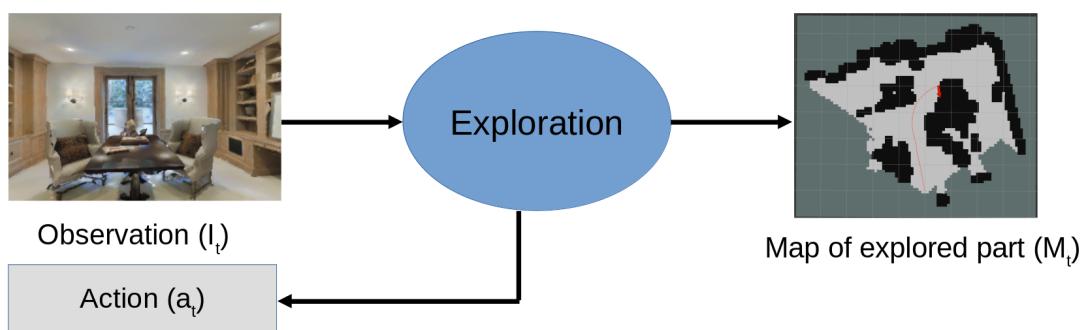


Рис. 27. Постановка задачи исследования неизвестной местности

Карта представлена в виде 2D-матрицы и состоит из свободных, занятых и неисследованных ячеек. Каждая ячейка этой матрицы представляет собой небольшой квадрат фиксированного размера (например, 5x5 см). В карте также указывается положение верхнего левого угла матрицы в глобальной системе координат. На начальном этапе карта представляет собой пустую матрицу.

$$M_t = (P_t \in \{0, 1, -1\}^{H \times W}; (x_t, y_t) \in \mathbb{R}^2); \quad M_0 = (\emptyset; (0, 0))$$

Здесь 0 кодирует свободную ячейку карты, 1 - занятую, -1 - неисследованную.

Действие представлено в виде изменения позиции робота:

$$a_t = (dx, dy, \delta)$$

Это команда для робота «двигаться на расстояние (dx, dy) (относительно его текущего положения) и вращаться на угол δ ». Чтобы упростить нашу модель, мы рассматриваем только четыре возможных действия: двигаться вперед ($a_t = (dx, 0, 0)$), повернуть налево ($a_t = (0, 0, \delta)$), повернуть направо ($a_t = (0, 0, -\delta)$) и оставаться на месте ($a_t = (0, 0, 0)$).

6.1.2 Метрики качества

Для измерения эффективности ИНМ обычно используются показатели абсолютного и относительного покрытия. Значение абсолютной метрики - это площадь построенной алгоритмом карты на определенных промежутках времени t . Значение относительной метрики - это доля площади окружающей среды, которая была исследована (нанесена на карту) на определенных промежутках времени t :

$$C_{abs} = \{|(i, j) : M_t^{i,j} \geq 0|\}, t \in T \quad (12)$$

$$C_{rel} = \left\{ \frac{|(i, j) : M_t^{i,j} \geq 0|}{|(i, j) : M^{i,j} \geq 0|} \right\}, t \in T \quad (13)$$

где M - истинная карта окружающей среды.

6.2 Описание метода

В работе предложен алгоритм для полностью автономного исследования неизвестной местности с помощью vSLAM. Алгоритм состоит из следующих шагов:

- Модуль одновременного картирования и локализации (SLAM) получает данные с камеры робота и оценивает его вычисляет траекторию и 2D-карту окружающей среды в реальном времени

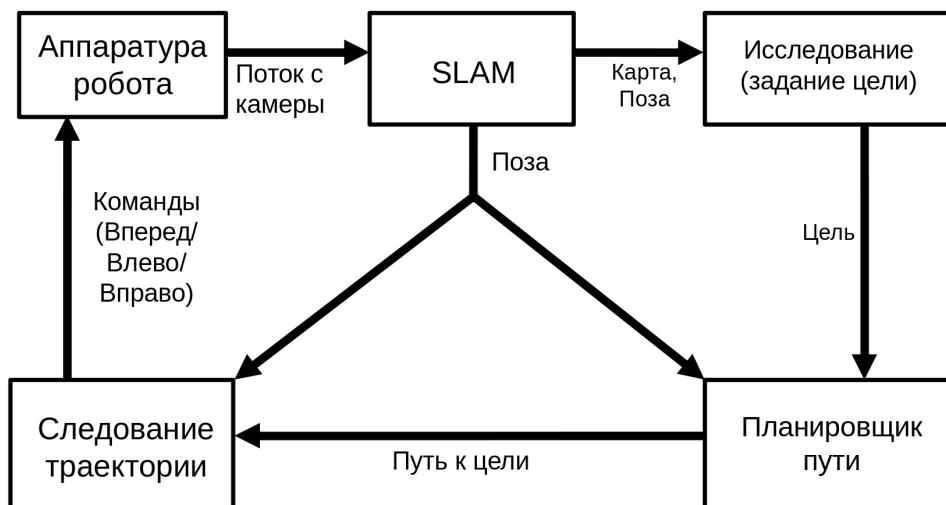


Рис. 28. Схема организации работы алгоритма автономной навигации

- Модуль **исследования неизвестной местности (ИНМ)** берет текущее положение робота и построенную модулем SLAM карту, и выбирает целевую точку, к которой должен идти робот
- Модуль **планирования пути** строит путь от робота до позиции цели на созданной SLAM карте
- Модуль **следования по траектории** принимает текущее положение робота и путь к цели и задает низкоуровневые команды контроллеру робота - куда двигаться: вперед, влево или вправо

С целью упрощения реализации на роботе и взаимодействия между модулями предложенный алгоритм был разработан с использованием программной библиотеки Robot Operating System (ROS)¹. Полная схема взаимодействия модулей представлена на рисунке 28.

Подробное описание модуля SLAM приведено в предыдущей главе. Для постановки целей используется алгоритм, основанный на ROS-пакете `explore_lite`², описанный в работе [25]. Алгоритм постановки целей ищет границы между свободным и неизвестным пространством на двумерной карте, построенной модулем SLAM. Чтобы найти эти границы, используется поиск в ширину (BFS) в графе. После выполнения поиска вычисляется стоимость всех границ исходя из расстояния до них и их размера. Центроид наиболее “ценной” границы помечается как цель для робота. Пример границ и цели показан на рисунке 29.

В данной работе в базовый алгоритм постановки целей из пакета `explore_lite` были внесены некоторые улучшения для повышения его эффективности и стабильности. В частности, была изменена функция стоимости границ - расстояния до границ

¹<http://www.ros.org>

²http://wiki.ros.org/explore_lite

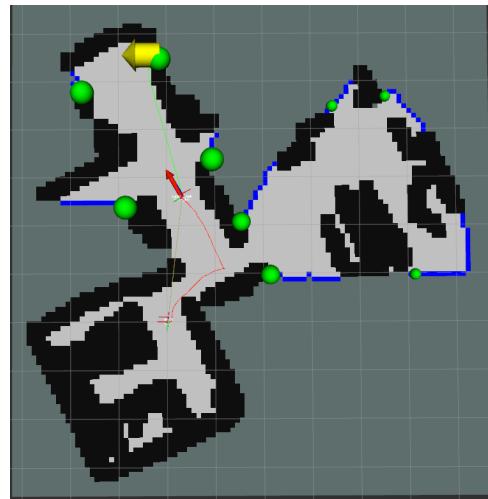


Рис. 29. Пример работы ИНМ, основанного на поиске границ. Черная область - это занятые ячейки карты, белая область - свободные ячейки карты, синие линии - границы, красная стрелка - позиция робота, а большая желтая стрелка - цель, выбранная алгоритмом. Зеленые круги отображают стоимость границ - чем больше круг, тем ценнее граница.

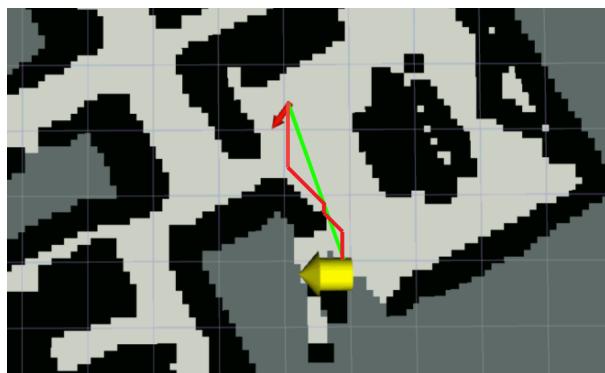


Рис. 30. Пример пути, построенного алгоритмами A^* (красный) и Θ^* (зеленый)

вычисляются как длины пути в построенной модулем SLAM карте, также учитывается ориентация робота и направление от его позиции до границы. Помимо этого, было добавлено расширение препятствий в построенной модулем SLAM карте, что позволило ИНМ находить меньше некорректных путей и работать более стабильно. Также, во избежание застревания перед невидимым препятствием, был добавлен “детектор столкновений”, который срабатывает, когда модуль следования траектории дает команду “вперед”, а модуль SLAM фиксирует отсутствие движения робота.

Для планирования пути от робота к цели был использован алгоритм Θ^* [42]. Этот алгоритм имеет высокую вычислительную эффективность и строит пути с произвольным углом. Таким образом, пути, построенные Θ^* в двумерной карте короче и намного плавнее, чем пути традиционных алгоритмов, таких как алгоритм Дейкстры [13] или A^* [23] (см. Рис. 30). Плавность траектории критически важна для систем визуального картирования и локализации, поскольку резкие движения могут сделать vSLAM нестабильным.



Рис. 31. Изображения и карты некоторых сцен набора данных Gibson

Для перемещения робота по предложенному пути используется простой и понятный алгоритм. Ориентация робота сравнивается с направлением от позиции робота до следующей точки пути. Если угол между ориентацией робота и направлением на точку пути ниже некоторого порогового значения (например, менее 5 градусов по модулю), робот перемещается вперед. Если он выше порога и отрицателен, поворачиваем робота налево. Если он выше порога и положительный, поворачиваем робота вправо.

6.3 Эксперименты

Экспериментальное исследование алгоритма ИНМ проводилось на датасете Gibson [58] в симуляторе Habitat [47]. Для экспериментов из 493 сцен датасета была отобрана 31 сцена без лестниц и дефектов текстуры. Площадь выбранных сцен варьировалась от 28 до 251 м². Пример изображений и карт этих сцен показан на рис. 31.

Для измерения эффективности исследования неизвестной местности была использована метрика покрытия - площадь пространства карты, исследованного в определенный момент времени. Была измерена как абсолютная, так и относительная площадь. Значения площадей измерялись для разного времени, отсчитанного от старта - от 15 до 240 с.

Чтобы исследовать разработанную систему в монокулярном режиме, было проведено несколько экспериментов на выбранных сценах. Для предсказания глубины использовалась полносверточная нейронная сеть, описанная в главе 5. Чтобы адаптировать эту нейросеть к симуляционной среде Habitat и параметрам используемой в Habitat камеры, она было дообучена на выборке из порядка 38000 пар изображение-

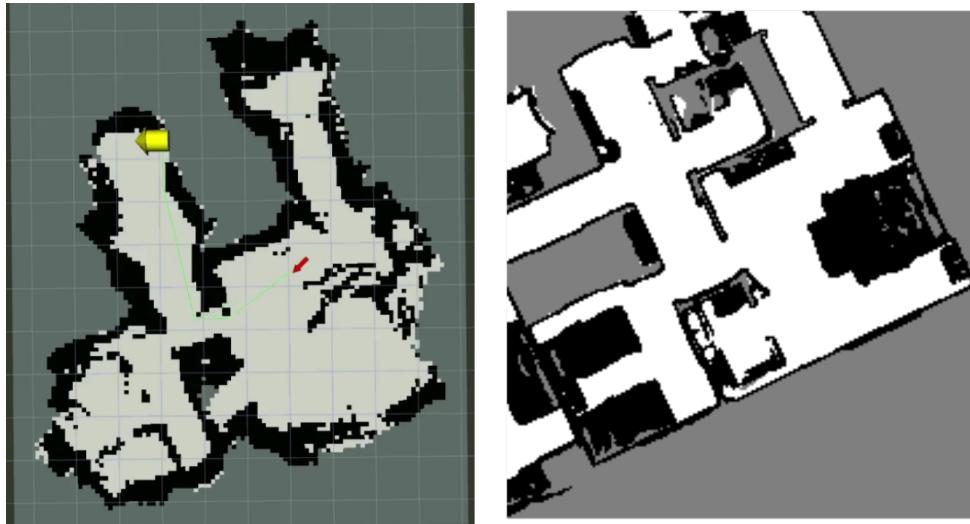


Рис. 32. Пример карты, построенной при исследовании с глубинами FCNN (слева) по сравнению с истинной картой (справа)



Рис. 33. Пример неточного картирования с предсказанными FCNN глубинами. В поле зрения камеры (левая часть изображения) наблюдается небольшой дверной проем. Но на карте (правая часть рисунка) дверной проем отмечен как стена (см. красный эллипс).

глубина, собранной по 25 сценам выбранной коллекции. Остальные 6 выбранных сцен были использованы для тестирования алгоритма ИНМ.

Эксперименты показали, что полученная система может работать автономно и строить правдоподобную карту (см. Рис. 32) в монокулярном режиме. Однако ошибки в оценке глубины нейросетью вызвали некоторые отклонения в построенной карте. Например, узкие дверные проемы иногда отображались как сплошная стена (см. Рис.33). Из-за такого неточного отображения планировщик не мог найти пути к далеким целям, и алгоритм исследовал только часть сцены. Значения показателей покрытия показаны на рисунке 34. Средняя исследованная часть сцены достигла 44 %.

В целом, тесты показали, что предлагаемая система способна работать в монокулярном режиме с нейросетевым восстановлением глубины, но неточное предсказание глубины может привести к ошибкам при картировании и неполному покрытию сцены. Возникающие ошибки могут быть устранены путем более тщательного обучения нейронной сети и точной настройки параметров SLAM. Видеозапись эксперимента с

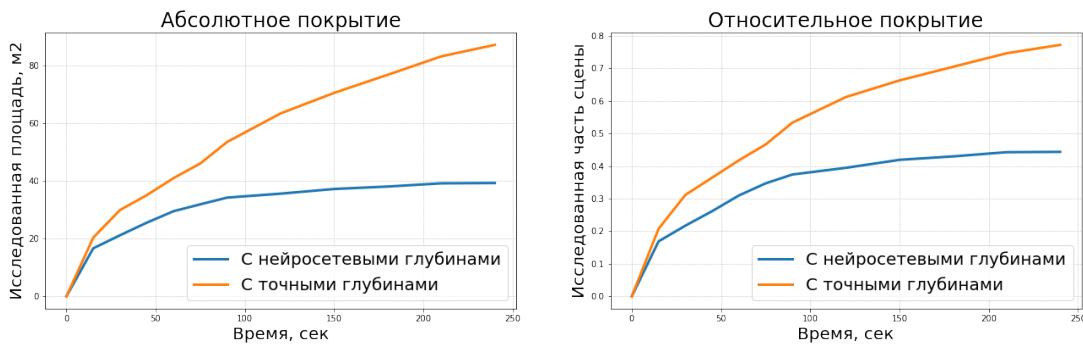


Рис. 34. Сравнение работы ИНМ с предсказанными FCNN глубинами и с точными глубинами: абсолютное покрытие (слева) и относительное покрытие сцены (справа)

предлагаемой системой ИНМ доступна по ссылке³.

6.4 Выводы

В данной работе представленный алгоритм vSLAM был интегрирован в систему автономного исследования неизвестной местности. Для оценки качества работы системы было проведено экспериментальное исследование на датасете Gibson в фотoreалистичном симуляторе Habitat. В результате экспериментов среднее значение покрытия площади сцены составило 44% за 240 секунд работы алгоритма. Причиной неполного покрытия сцены являются ошибки картирования, возникающие из-за неточного определения глубины нейросетью. Данные ошибки могут быть устранены путем более тщательного обучения нейросети.

³<https://drive.google.com/file/d/1QJWmjR9Y2VWbycZVwz3Y6Dl9Rzkp-zjB/view?usp=sharing>

Глава 7

Заключение

В данной работе был рассмотрен подход к решению задачи одновременного картирования и локализации по видеоданным (vSLAM) в реальном времени в контексте автономной навигации малого мобильного робота в неизвестной местности. Дан-ный подход основан на восстановлении карт глубин изображений с помощью полно-сверточных нейронных сетей и методе одновременного картирования и локализации RTAB-МАР. Для эффективной работы нейросети в реальном времени в условиях ограниченных вычислительных ресурсов были применены различные архитектур-ные и технические оптимизации.

Для оценки качества предложенного метода vSLAM была собрана коллекция дан-ных в фотореалистичном симуляторе Habitat, также был использован оригинальный под-ход к оценке качества построенной алгоритмом vSLAM карты, учитывающий кон-текст задачи vSLAM. С целью оценки качества предложенного алгоритма vSLAM были про-веденны эксперименты на собранной коллекции данных, а также на колес-ной робототехнической платформе с компьютером NVidia Jetson TX2. Эксперименты показали, что предложенный алгоритм способен строить плотную правдоподобную карту местности в реальном времени по данным с единственной видеокамеры.

С целью оценки применимости предложенного метода vSLAM для автономной навигации роботов, данный метод был интегрирован в систему исследования неиз-вестной местности (ИНМ). Было проведено экспериментальное исследование полу-ченной системы в фотореалистичном симуляторе Habitat. Помимо задачи vSLAM, в ходе экспериментов с помощью системы ИНМ решались задачи постановки це-лей, а также планирования и следования траектории. Эксперименты показали, что с использованием предложенного метода vSLAM система ИНМ способна функционировать полностью автономно в реальном времени, строить правдоподобные карты помещений и планировать в них маршруты.

Программная реализация предложенного метода vSLAM интегрирована с библиотекой Robotic Operating System (ROS) и выложена в открытый доступ. Исходный код алгоритма, а также параметры запуска, инструкции по установке и утилиты для

работы с нейронными сетями доступны в репозитории¹.

¹https://github.com/cnndepth/tx2_fcnn_node

Список литературы

- [1] Herbert Bay, Tinne Tuytelaars, Luc Van Gool. “Surf: Speeded up robust features”. *European conference on computer vision*. Springer. 2006, c. 404—417.
- [2] Andrey Bokovoy, Kirill Muraviev. “Assessment of Map Construction in vSLAM”. *2021 International Siberian Conference on Control and Communications (SIBCON)*. IEEE. 2021, c. 1—6.
- [3] Andrey Bokovoy, Kirill Muraviev, Konstantin Yakovlev. “Map-merging algorithms for visual slam: Feasibility study and empirical evaluation”. *Russian Conference on Artificial Intelligence*. Springer. 2020, c. 46—60.
- [4] Andrey Bokovoy, Kirill Muravyev, Konstantin Yakovlev. *MAOMaps: A Photo-Realistic Benchmark For vSLAM and Map Merging Quality Assessment*. 2021. arXiv: 2105.14994 [cs.CV].
- [5] Andrey Bokovoy, Kirill Muravyev, Konstantin Yakovlev. “Real-time vision-based depth reconstruction with Nvidia Jetson”. *2019 European Conference on Mobile Robots (ECMR)*. IEEE. 2019, c. 1—6.
- [6] Eric Brachmann и др. “DSAC-differentiable RANSAC for camera localization”. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, c. 6684—6692.
- [7] Michael Burri и др. “The EuRoC micro aerial vehicle datasets”. *The International Journal of Robotics Research* **35** 10 (2016), c. 1157—1163.
- [8] Michael Calonder и др. “Brief: Binary robust independent elementary features”. *European conference on computer vision*. Springer. 2010, c. 778—792.
- [9] Angel Chang и др. “Matterport3d: Learning from rgb-d data in indoor environments”. *arXiv preprint arXiv:1709.06158* (2017).
- [10] Devendra Singh Chaplot и др. “Learning to explore using active neural slam”. *arXiv preprint arXiv:2004.05155* (2020).
- [11] Dmitry Chetverikov, Dmitry Stepanov, Pavel Krsek. “Robust Euclidean alignment of 3D point sets: the trimmed iterative closest point algorithm”. *Image and vision computing* **23** 3 (2005), c. 299—309.

- [12] Angela Dai и др. “Scannet: Richly-annotated 3d reconstructions of indoor scenes”. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, с. 5828—5839.
- [13] Edsger W Dijkstra и др. “A note on two problems in connexion with graphs”. *Numerische mathematik* **1** 1 (1959), с. 269—271.
- [14] Felix Endres и др. “3-D mapping with an RGB-D camera”. *IEEE transactions on robotics* **30** 1 (2013), с. 177—187.
- [15] Jakob Engel, Thomas Schöps, Daniel Cremers. “LSD-SLAM: Large-scale direct monocular SLAM”. *European conference on computer vision*. Springer. 2014, с. 834—849.
- [16] Dustin Franklin. “Nvidia jetson tx2 delivers twice the intelligence to the edge”. *NVIDIA Accelerated Computing—Parallel Forall* **6** (2017).
- [17] Jorge Fuentes-Pacheco, José Ruiz-Ascencio, Juan Manuel Rendón-Mancha. “Visual simultaneous localization and mapping: a survey”. *Artificial intelligence review* **43** 1 (2015), с. 55—81.
- [18] Andreas Geiger, Philip Lenz, Raquel Urtasun. “Are we ready for autonomous driving? the kitti vision benchmark suite”. *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2012, с. 3354—3361.
- [19] Andreas Geiger и др. “The KITTI vision benchmark suite”. URL <http://www. cvlbs. net/datasets/kitti> **2** (2015).
- [20] Klaus Greff и др. “LSTM: A search space odyssey”. *IEEE transactions on neural networks and learning systems* **28** 10 (2016), с. 2222—2232.
- [21] Giorgio Grisetti и др. “g2o: A general framework for (hyper) graph optimization”. *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Shanghai, China*. 2011, с. 9—13.
- [22] Ankur Handa и др. “A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM”. *2014 IEEE international conference on Robotics and automation (ICRA)*. IEEE. 2014, с. 1524—1531.
- [23] Peter E Hart, Nils J Nilsson, Bertram Raphael. “A formal basis for the heuristic determination of minimum cost paths”. *IEEE transactions on Systems Science and Cybernetics* **4** 2 (1968), с. 100—107.
- [24] Kaiming He и др. “Deep residual learning for image recognition”. *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, с. 770—778.
- [25] Jiří Hörner. *Map-merging for multi-robot system*. Bachelor’s thesis. Prague, 2016. URL: <https://is.cuni.cz/webapps/zzp/detail/174125/>.
- [26] Lam Huynh и др. “Guiding monocular depth estimation using depth-attention volume”. *European Conference on Computer Vision*. Springer. 2020, с. 581—597.

- [27] Eddy Ilg и др. “Flownet 2.0: Evolution of optical flow estimation with deep networks”. *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, с. 2462—2470.
- [28] Diederik P Kingma, Jimmy Ba. “Adam: A method for stochastic optimization”. *arXiv preprint arXiv:1412.6980* (2014).
- [29] Georg Klein, David Murray. “Parallel tracking and mapping for small AR workspaces”. *2007 6th IEEE and ACM international symposium on mixed and augmented reality*. IEEE. 2007, с. 225—234.
- [30] Nathan Koenig, Andrew Howard. “Design and use paradigms for gazebo, an open-source multi-robot simulator”. *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*. т. 3. IEEE. 2004, с. 2149—2154.
- [31] Anis Koubâa и др. *Robot Operating System (ROS)*. т. 1. Springer, 2017.
- [32] Kazutaka Kurihara и др. “Optical motion capture system with pan-tilt camera tracking and real time data processing”. *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*. т. 2. IEEE. 2002, с. 1241—1248.
- [33] Mathieu Labbe, Francois Michaud. “Memory management for real-time appearance-based loop closure detection”. *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2011, с. 1271—1276.
- [34] Mathieu Labb , Fran ois Michaud. “RTAB-Map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation”. *Journal of Field Robotics* **36** 2 (2019), с. 416—446.
- [35] Iro Laina и др. “Deeper depth prediction with fully convolutional residual networks”. *2016 Fourth international conference on 3D vision (3DV)*. IEEE. 2016, с. 239—248.
- [36] Zhengqi Li, Noah Snavely. “Megadepth: Learning single-view depth prediction from internet photos”. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, с. 2041—2050.
- [37] Tsung-Yi Lin и др. “Microsoft coco: Common objects in context”. *European conference on computer vision*. Springer. 2014, с. 740—755.
- [38] David G Lowe. “Distinctive image features from scale-invariant keypoints”. *International journal of computer vision* **60** 2 (2004), с. 91—110.
- [39] Xuan Luo и др. “Consistent video depth estimation”. *ACM Transactions on Graphics (TOG)* **39** 4 (2020), с. 71—1.
- [40] Raul Mur-Artal, Jose Maria Martinez Montiel, Juan D Tardos. “ORB-SLAM: a versatile and accurate monocular SLAM system”. *IEEE transactions on robotics* **31** 5 (2015), с. 1147—1163.

- [41] Raul Mur-Artal, Juan D Tardós. “Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras”. *IEEE Transactions on Robotics* **33** 5 (2017), c. 1255—1262.
- [42] Alex Nash и др. “Theta^{*}: Any-angle path planning on grids”. *AAAI*. т. 7. 2007, c. 1177—1183.
- [43] Art B Owen. “A robust hybrid of lasso and ridge regression”. *Contemporary Mathematics* **443** 7 (2007), c. 59—72.
- [44] Vaishakh Patil и др. “Don’t forget the past: Recurrent depth estimation from monocular video”. *IEEE Robotics and Automation Letters* **5** 4 (2020), c. 6813—6820.
- [45] S Rooban и др. “CoppeliaSim: Adaptable modular robot and its different locomotions simulation framework”. *Materials Today: Proceedings* (2021).
- [46] Ethan Rublee и др. “ORB: An efficient alternative to SIFT or SURF”. *2011 International conference on computer vision*. Ieee. 2011, c. 2564—2571.
- [47] Manolis Savva и др. “Habitat: A platform for embodied ai research”. *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, c. 9339—9347.
- [48] Johannes L Schonberger, Jan-Michael Frahm. “Structure-from-motion revisited”. *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, c. 4104—4113.
- [49] Nathan Silberman и др. “Indoor segmentation and support inference from rgbd images”. *European conference on computer vision*. Springer. 2012, c. 746—760.
- [50] Jürgen Sturm и др. “A benchmark for the evaluation of RGB-D SLAM systems”. *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2012, c. 573—580.
- [51] Takafumi Taketomi, Hideaki Uchiyama, Sei Ikeda. “Visual SLAM algorithms: a survey from 2010 to 2016”. *IPSJ Transactions on Computer Vision and Applications* **9** 1 (2017), c. 1—11.
- [52] Keisuke Tateno, Federico Tombari, Nassir Navab. “Real-time and scalable incremental segmentation on dense slam”. *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2015, c. 4465—4472.
- [53] Keisuke Tateno и др. “Cnn-slam: Real-time dense monocular slam with learned depth prediction”. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, c. 6243—6252.
- [54] Zachary Teed, Jia Deng. “Deepv2d: Video to depth with differentiable structure from motion”. *arXiv preprint arXiv:1812.04605* (2018).

-
- [55] Benjamin Ummenhofer и др. “Demon: Depth and motion network for learning monocular stereo”. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, с. 5038—5047.
 - [56] Oliver Wasenmüller, Marcel Meyer, Didier Stricker. “CoRBS: Comprehensive RGB-D benchmark for SLAM using Kinect v2”. *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE. 2016, с. 1—7.
 - [57] Diana Wofk и др. “Fastdepth: Fast monocular depth estimation on embedded systems”. *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, с. 6101—6108.
 - [58] Fei Xia и др. “Gibson env: Real-world perception for embodied agents”. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, с. 9068—9079.
 - [59] Nan Yang и др. “D3vo: Deep depth, deep pose and deep uncertainty for monocular visual odometry”. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, с. 1281—1292.
 - [60] Haokui Zhang и др. “Exploiting temporal consistency for real-time video depth estimation”. *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, с. 1725—1734.
 - [61] Zhengyou Zhang. “Microsoft kinect sensor and its effect”. *IEEE multimedia* **19** 2 (2012), с. 4—10.
 - [62] Zhengyou Zhang, Ying Shan. *Incremental motion estimation through local bundle adjustment*. US Patent 6,996,254. февр. 2006.
 - [63] Tinghui Zhou и др. “Unsupervised learning of depth and ego-motion from video”. *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, с. 1851—1858.
 - [64] Laurent Zwald, Sophie Lambert-Lacroix. “The berhu penalty and the grouped effect”. *arXiv preprint arXiv:1207.6868* (2012).