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

Факультет инноваций и высоких технологий
Кафедра анализа данных

Выпускная квалификационная работа бакалавра по направлению 010302 «Прикладные математика и информатика»

Применение нейронных сетей в задаче одновременного картирования и локализации

Студент 599 группы
Муравьев К. Ф.

Научный руководитель
Яковлев К. С.

Долгопрудный
2019

Содержание

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 структуры из движения (Structure from Motion, SfM), имеют существенные недостатки, такие, как потеря структуры при поворотах робота на месте и невозможность восстановления точного масштаба карты (все расстояния на построенной карте - относительные). Восстановление карты глубин по видеопотоку позволяет устранить данные недостатки, сведя задачу vSLAM к задаче одновременного картирования и локализации с использованием видеоданных и данных о глубине (RGBD-SLAM), для которой разработаны эффективные методы решения, например, [13].

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 видеопотока в реальном времени с помощью бортовых вычислителей робототехнических систем. Например, обработка одной пары изображений с помощью метода, предложенного в работе [25], 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 [14] 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 нейрон-

ных сетей на малогабаритных роботах. Например, встраиваемый компьютер NVIDIA Jetson TX2 [8] обладает графическим ускорителем и мощным центральным процессором, и при этом он достаточно компактен и энергоэффективен для использования в малых робототехнических устройствах. В работе [23] описывается метод восстановления карт глубин изображений, основанный на нейронной сети и работающий на NVIDIA Jetson TX2 со скоростью 30 кадров в секунду.

В данной работе предлагается метод решения задачи vSLAM, основанный на восстановлении карт глубин изображений с помощью нейронных сетей. Проводится тестирование предлагаемого метода на платформе NVIDIA Jetson TX2 в реальном времени.

Глава 2

Постановка задачи

2.1 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.

Математически задачу можно сформулировать таким образом: существует набор точек в трёхмерном пространстве $\{M_i\} = \mathbf{W}, M_i \in \mathbb{R}^3$, называемый сценой. Дана 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:

$$\mathcal{P}^t = \{\mathcal{P}^t_{c,h,w}\}_{c\in[1\ldots3],h\in[1\ldots H],w\in[1\ldots W]} \in [0,1]^{3\times H\times W}$$

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

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

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

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.

2.2 Одновременное картирование и локализация по видеоданным и данным 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. Дана 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:

$$\mathcal{P}^t = \{\mathcal{P}^t_{c,h,w}\}_{c\in[1\ldots3],h\in[1\ldots H],w\in[1\ldots W]} \in [0,1]^{3\times H\times W}$$

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

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

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

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

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

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

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

2.3 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 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. Математически задачу можно сформулировать таким образом: существует набор точек в трёхмерном пространстве $\{M_i\} = \mathbf{W}, M_i \in \mathbb{R}^3$, называемый сценой. Дана 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:

$$\mathcal{P}^t = \{\mathcal{P}^t_{c,h,w}\}_{c\in[1\ldots3],h\in[1\ldots H],w\in[1\ldots W]} \in [0,1]^{3\times H\times W}$$

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

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

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

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

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

Глава 3

Методы решения задачи одновременного картирования и локализации

3.1 EKF-SLAM

Алгоритм EKF-SLAM решает задачу одновременного картирования и локализации с помощью расширенного фильтра Калмана. EKF-SLAM принимает на вход информацию о положении ориентиров на изображениях с камеры, и по полученным данным определяет в реальном времени положение в пространстве этих ориентиров и самой камеры.

Алгоритм EKF-SLAM имеет такой вид:

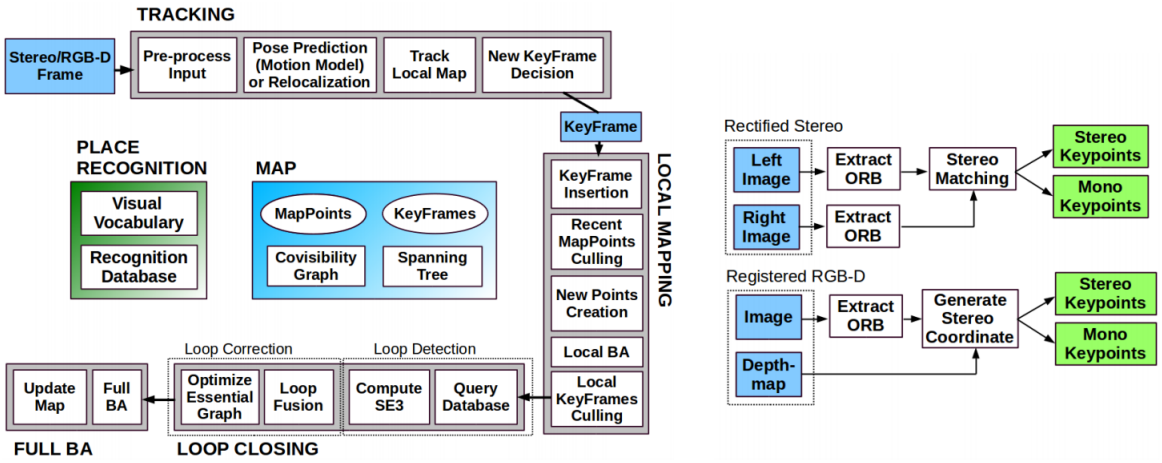
1. Инициализация карты
2. Шаг предсказания — рассчитывается оценка нового положения камеры и её точность.
3. Алгоритмами, не являющимися частью EKF-SLAM, на новом изображении с камеры находятся известные и новые ориентиры.
4. Если были обнаружены новые ориентиры, они добавляются на карту.
5. Шаг коррекции — полученная информация об ориентирах используется для уточнения карты и положения камеры.
6. Повторение шагов 2 - 6

Ориентирами называются точки в пространстве, которые видны так, что можно точно определить их координаты на нескольких изображениях. На вход алгоритму EKF-SLAM поступают не сами изображения, а только координаты ориентиров на изображениях.

Данный алгоритм имеет множество вариаций, различающихся методами выделения особых точек и сопоставления ориентиров. Преимуществами данного алгоритма являются простота реалзиации и возможность проводить картирование и локализацию по данным с единственной видеокамеры. К недостаткам EKF-SLAM можно отнести низкую точность локализации и высокую вычислительную сложность оптимизации с помощью расширенного фильтра Калмана.

3.2 ORB-SLAM

Метод ORB-SLAM [17] решает задачу одновременного картирования и локализации в реальном времени по данным с единственной видеокамеры. Данный метод основан на детекторе особых то-



(a) System Threads and Modules. (b) Input pre-processing

Рис. 1. Основные компоненты ORB SLAM

чек ORB [21]. Высокая скорость детектора ORB позволяет методу работать в реальном времени в условиях ограниченных вычислительных ресурсов. На рисунке 1 показаны основные компоненты алгоритма. Его работа разделена на три основных потока:

- Tracking — отслеживание кадров. Данный поток приблизительно определяет текущее положение камеры путем поиска похожего кадра в локальной карте и сопоставления ключевых точек с найденным кадром.
- Local Mapping — выполняет построение карты вблизи текущего положения камеры и оптимизирует карту.
- Loop Closing — алгоритм замыкания циклов, который ищет и объединяет похожие кадры.

Первый шаг алгоритма — инициализация карты, которая состоит из карты точек (MapPoints) и ключевых кадров (KeyFrames). Ключевые кадры сохраняют информацию о положении камеры и ключевых точках, присутствующих на кадре. Далее, с использованием карты точек и ключевых кадров выполняется построение неориентированного взвешенного графа пересечений (Covisibility Graph). В данном графе каждый кадр представляет собой узел. Пара узлов связывается ребрами, если у соответствующей пары кадров наблюдается более 15-ти общих точек карты. Весом ребер является число общих точек.

Поток Tracking отслеживает перемещение камеры. Он извлекает ключевые точки с помощью алгоритма ORB, а затем пытается сопоставить их с предыдущим кадром. В случае неудачи выполняется релокализация. В случае успешного сопоставления в графе пересечений ключевых кадров ищется локальная карта, после чего производится сопоставление точек текущего кадра с точками в локальной карте путем проекции. И, наконец, выполняется оптимизация локальной карты (Local Bundle Adjustment [27]), с помощью которой уточняется положение камеры.

Модуль локального построения карты (LocalMapping) обрабатывает новые кадры и добавляет их в граф пересечений и остовное дерево графа. Кроме того, он выполняет локальную оптимизацию карты с помощью метода Local Bundle Adjustment для получения более точной реконструкции облака точек вблизи положения камеры.

Модуль замыкания циклов (Loop Closing) ищет похожие кадры для каждого нового кадра. Если такие кадры найдены — для них и текущего кадра вычисляется преобразование подобия. Затем положения найденного и текущего кадров выравниваются путем применения найденного преобразования, а одинаковые ключевые точки объединяются. Кроме локальной оптимизации, ORB SLAM выполняет глобальную оптимизацию карты (Full Bundle Adjustment), которая позволяет уменьшить накопленную ошибку с учетом найденных замыканий циклов.

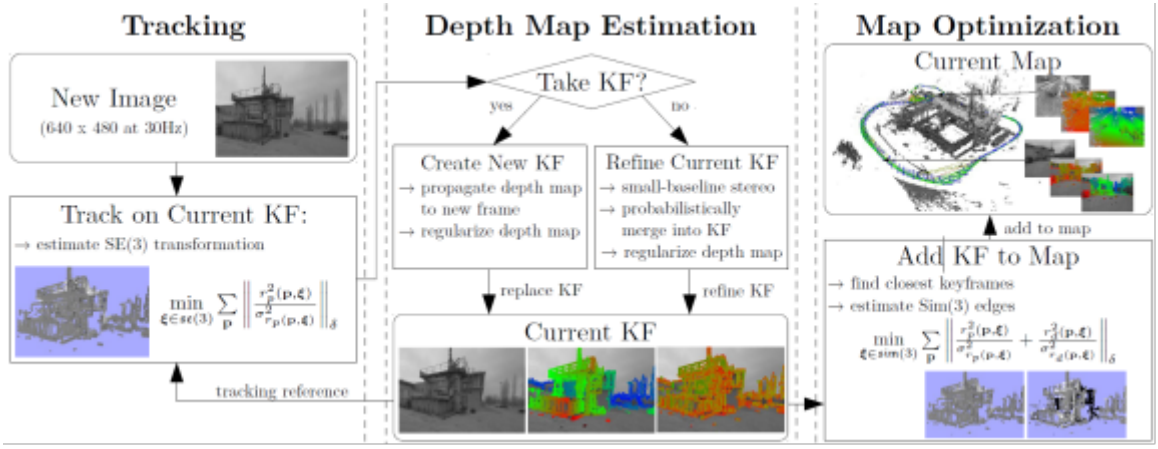


Рис. 2. Основные компоненты LSD SLAM

3.3 ORB-SLAM2

Данный метод является модификацией рассмотренного ранее метода ORB-SLAM [17]. Он может решать задачу SLAM как по данным с единственной видеокамеры, так и по данным со стереокамеры или RGB-D камеры.

При наличии стереокамеры или RGB-D камеры особые точки на изображениях подразделяются на близкие и далекие. По близким точкам с помощью триангуляции с высокой точностью вычисляются перемещение камеры и масштаб карты, а по далеким с высокой точностью вычисляется угол поворота камеры. Такое разделение позволяет повысить точность картирования и локализации по сравнению со стандартным алгоритмом ORB-SLAM.

Преимущества методов ORB-SLAM и ORB-SLAM2:

1. Высокая скорость работы
2. Возможность работы при ограниченных вычислительных ресурсах
3. Высокая точность картирования и локализации по данным со стереокамер или камер глубины

Недостатки методов ORB-SLAM и ORB-SLAM2:

1. Разреженность построенной карты
2. В случае использования данных с единственной видеокамеры: потеря структуры при поворотах на месте

3.4 LSD-SLAM

Алгоритм [7] включает в себя три основных модуля: tracking, depth map estimation и map optimization. Схема модулей показана на рисунке 2.

Модуль tracking непрерывно отслеживает новые изображения с камеры и определяет перемещение камеры. Для оценки перемещения вычисляется преобразование подобия между предыдущим и новым кадрами. В отличие от ORB SLAM, основанном на сопоставлении ключевых точек, в LSD SLAM для расчета преобразования используется минимизация фотометрической ошибки.

Модуль depthmapestimation сравнивает новый кадр с текущим, а затем уточняет или полностью заменяет текущий кадр. Для сравнения используется взвешенная сумма относительного расстояния от нового кадра до текущего и углов поворота между ними. Если вычисленная сумма больше заданного порога — текущий кадр заменяется новым.

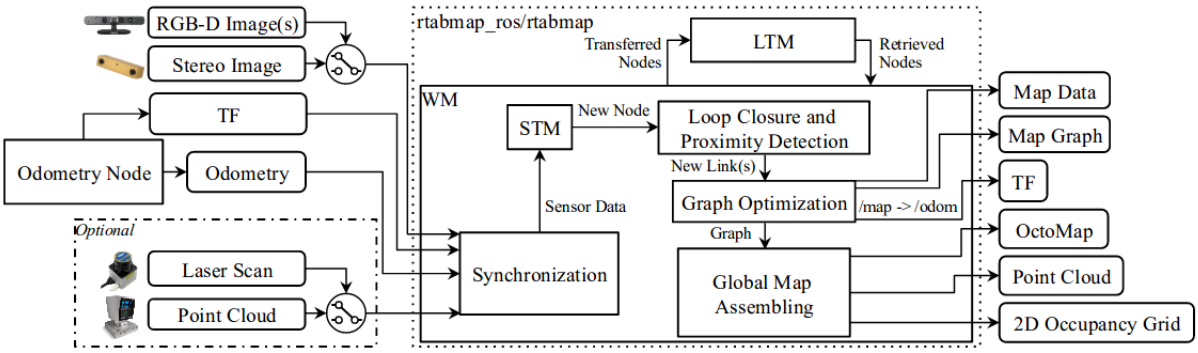


Рис. 3. Общая схема алгоритма RTAB-Map

Модуль mapoptimization выполняет оптимизацию карты. Оптимизация позволяет предотвратить накопление ошибок в отслеживании местоположения и поддерживает точность построения карты окружающей среды. Оптимизация выполняется библиотекой g2o непрерывно в отдельном потоке. Для хранения карты окружающей среды используется граф. Каждый узел графа хранит изображение, и соответствующую ему обратную карту глубины. Узлы соединяются ребрами, которые содержат найденное преобразование подобия между двумя изображениями.
Преимущества алгоритма LSD-SLAM:

1. Высокая плотность построенной карты
2. Решение задачи SLAM по данным с единственной камеры

Недостатки алгоритма LSD-SLAM:

1. Высокая ресурсоемкость
2. В среднем более низкая точность локализации, чем у алгоритма ORB-SLAM

3.5 RTABMAP

Алгоритм RTABMAP [13] 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 3.

Для одометрии по кадрам вычисляются особые точки с помощью детектора BRIEF [4]. По сопоставлению особых точек на текущем и ключевом кадрах с помощью алгоритма PnP RANSAC [3] вычисляется перемещение камеры. Полученное положение камеры корректируется с помощью алгоритма Local Bundle Adjustment [27] и предсказаний на основе предыдущих движений камеры. Новый ключевой кадр добавляется, когда у текущего кадра и ключевого будет мало сопоставлений. Схема вычисление одометрии представлена на рисунке 4.

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 5.

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 [2]. Алгоритм использует три вида памяти: WM (рабочая), в

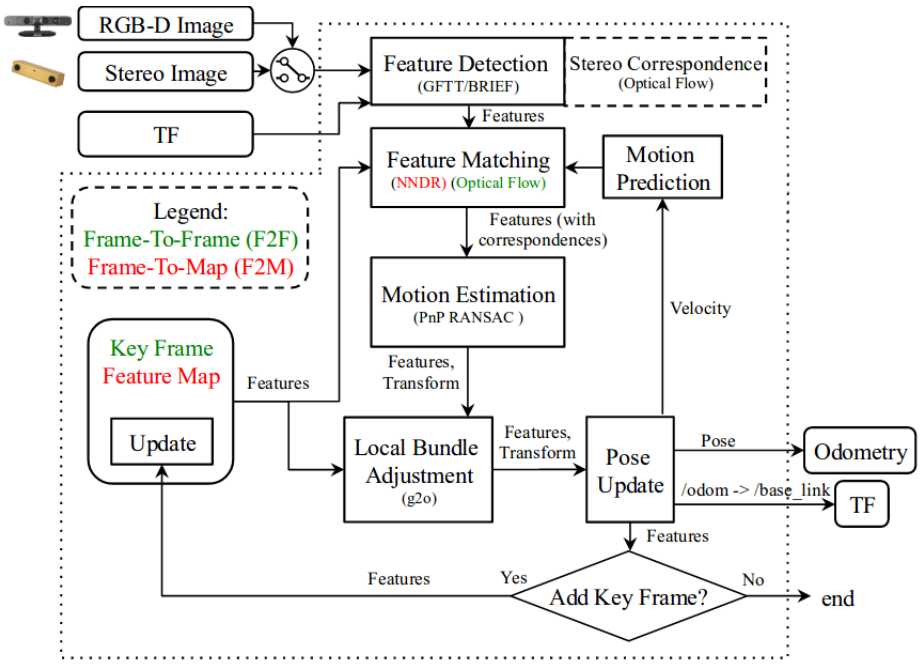


Рис. 4. Схема вычисления одометрии в методе RTAB-Map

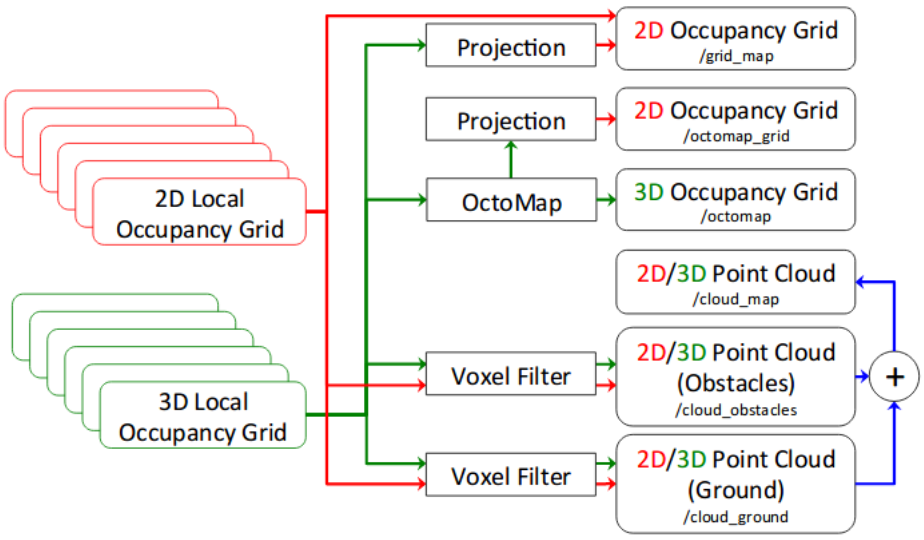


Рис. 5. Схема построения плотной глобальной карты по локальным сеткам

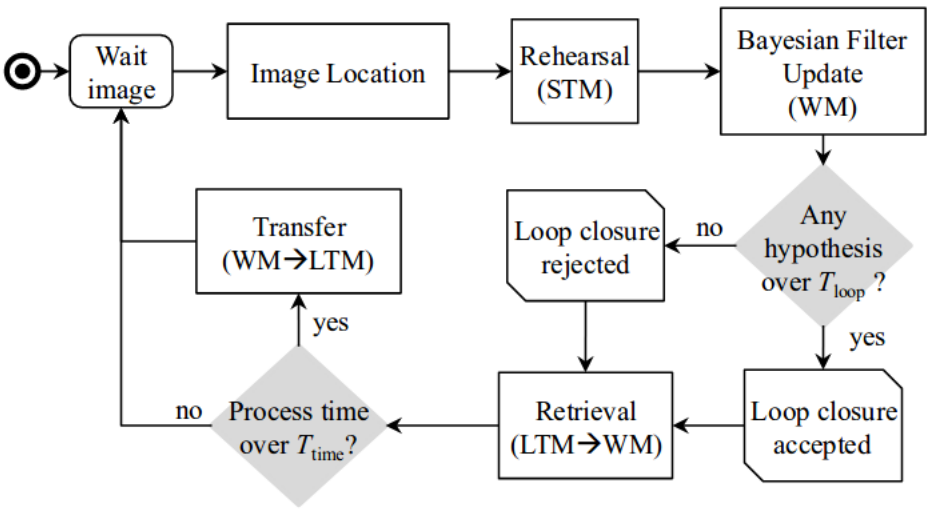


Рис. 6. Схема замыкания циклов в алгоритме RTAB-Map

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 (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 6.

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-Map обладает существенными недостатками:

1. Невозможность работы при отсутствии информации о глубине изображений
2. Потеря одометрии при отсутствии сопоставленных ориентиров
3. Высокая ресурсоемкость из-за необходимости обработки трехмерных облаков точек и построения плотной карты

3.6 Выводы

В данной главе были рассмотрены следующие методы решения задачи одновременного картирования и локализации:

1. EKF-SLAM - построение карты с использованием расширенного фильтра Калмана

2. ORB-SLAM и его модификация ORB-SLAM2 - алгоритм решения задачи SLAM с использованием детектора особых точек ORB, а также локальной и глобальной оптимизации построенной карты

3. LSD-SLAM - SLAM с использованием геометрических преобразований и представления карты в виде графа

4. RTAB-Map - SLAM с использованием стереокамеры или RGB-D камеры, с эффективным замыканием циклов

Методы EKF-SLAM, ORB-SLAM и LSD-SLAM используют для картирования и локализации данные с единственной видеокамеры; методы ORB-SLAM2 и RTAB-Map могут использовать данные о глубине изображений, полученные со стереопары камер или с датчика глубины.
Методы EKF-SLAM и ORB-SLAM строят разреженную карту окружающей местности, а методы LSD-SLAM и RTABMAP - плотную карту, которую можно использовать для планирования траектории робота.
Для использования в данной работе был выбран алгоритм RTAB-Map, так как он строит плотную карту, пригодную для планирования траектории робота, и при этом обладает высокой точностью локализации. Для применения алгоритма RTAB-Map в условиях отсутствия датчиков глубины и стереокамер применено восстановление карт глубин с помощью нейронных сетей.

Глава 4

Методы решения задачи восстановления глубины по видеопотоку

Для восстановления карт глубин по видеопотоку существует три основных семейства методов:
1) Методы, основанные на использовании движения камеры (Structure from Motion [12]). Данные методы вычисляют перемещение камеры в пространстве по изменению положения особых точек на изображениях и используют данную информацию для трехмерной реконструкции наблюдаемой сцены.
2) Методы, основанные на использовании движения камеры и априорных знаний (в виде решающих правил либо предварительно собранной коллекции данных). В данных методах построенные карты глубин уточняются с использованием информации, полученной из имеющейся коллекции данных.
3) Восстановление карт глубин по одиночным изображениям с помощью нейронных сетей, обученных на предварительно собранной коллекции данных. В данных методах карта глубины предсказывается с помощью предварительно обученной нейронной сети отдельно для каждого кадра из видеопотока.
Более подробно методы восстановления карт глубин по видеопотоку будут рассмотрены в следующих разделах.

4.1 Восстановление глубины по движению камеры

Как правило, восстановление глубины по видеопотоку с использованием движения камеры производится по следующей схеме:
1) Извлечение особых точек из текущего кадра
2) Сопоставление извлеченных особых точек с особыми точками на предыдущих кадрах
3) Фильтрация ложных соответствий
4) Вычисление перемещения камеры и глубины кадра с помощью решения системы уравнений
Особые точки извлекаются с помощью одного из многочисленых методов детекции. Самым распространенным методом извлечения особых точек является метод Scale-invariant feature transform (SIFT) [16]. Метод SIFT каждой особой точке приписывает дескриптор - вектор, кодирующий описание данной особой точки. Поиск соответствий особых точек осуществляется по данным дескрипторам - для каждой особой точки текущего кадра ищется особая точка с наиболее близким дескриптором на предыдущем кадре.
Для фильтрации ложных соответствий используются эвристические и геометрические методы. Геометрические методы фильтрации основаны на использовании эпиполярного ограничения [6]. Обыч-

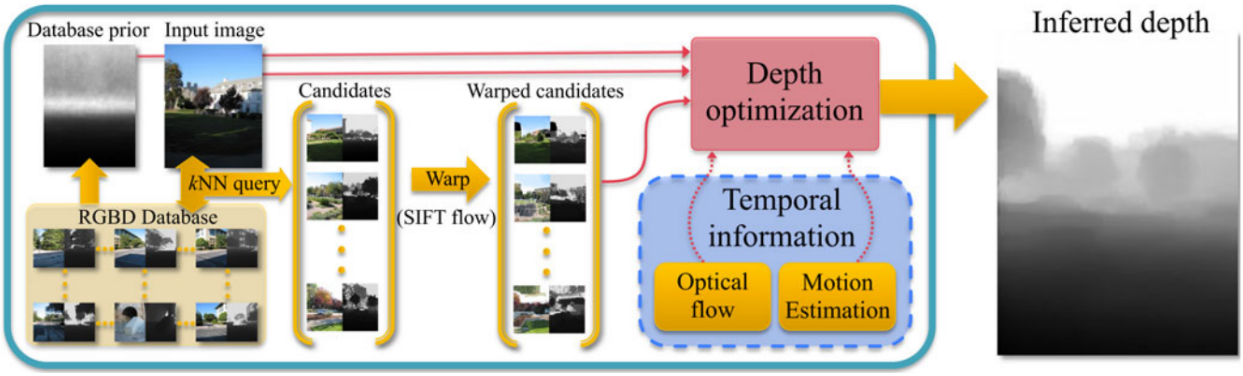


Рис. 7. Схема непараметрического метода восстановления глубины

но используется алгоритм RANSAC [3], строящий гипотезы о соответствии по нескольким случайным подвыборкам точек и выбирающий наилучшую из построенных.

Искомая структура (пространственное положение изображенных на кадрах особых точек) задается нелинейной системой уравнений с большим количеством неизвестных. Полученная система обычно решается с помощью поиска хорошего начального приближения и уточнения найденного приближения методом градиентного спуска. Структура также может дополнительно уточняться с помощью метода Bundle Adjustment [27].

Данные методы позволяют с довольно высокой точностью восстанавливать карты глубин по видеопотоку с движущейся камеры в произвольной среде без использования коллекций данных. Однако они обладают высокой вычислительной сложностью, связанной с решением больших нелинейных систем уравнений.

4.2 Восстановление глубины по движению камеры с использованием коллекции данных

Данные методы уточняют карты глубин, полученные по движению камеры, с использованием информации, извлеченной из предварительно собранной коллекции данных, что позволяет повысить точность восстановления глубины или снизить вычислительные затраты на итеративное решение систем уравнений.

Методы подразделяются на параметрические и непараметрические. В непараметрических методах карта глубины текущего изображения обычно восстанавливается с использованием карты глубины похожего на него изображения из коллекции. В параметрических методах из коллекции данных извлекаются закономерности с помощью методов машинного обучения, и карты глубин изображений восстанавливаются с использованием найденных закономерностей. Более подробно параметрические и непараметрические методы будут рассмотрены в соответствующих подразделах.

4.2.1 Непараметрические методы

Один из вариантов непараметрического восстановления глубины по видеопотоку с использованием коллекции данных описан в [10]. Схема метода представлена на рисунке 7.

В представленном алгоритме для входного изображения извлекаются особые точки с помощью детектора GIST [18]. По дескрипторам особых точек находятся изображения из коллекции, наиболее похожие на входное. С помощью метода SIFT flow [15] ищется преобразование этих изображений во входное. Данное преобразование применяется также к глубинам изображений из коллекции.

Преобразованная глубина корректируется с помощью вероятностных методов. Скорректированная глубина дополнительно уточняется с использованием оптического потока и движения объектов на видеопоследовательности.

4.2.2 Паметрические методы

В параметрических методах восстановления глубины по видеопотоку чаще всего используются нейронные сети. Например, в [25] карты глубин вычисляются с помощью нейронной сети, принимающей на вход два изображения одного и того же объекта, снятые с разных ракурсов. Помимо карт глубин, данная нейросеть вычисляет также оптический поток между двумя изображениями и вектор перемещения камеры.
Нейросеть состоит из трех частей: Bootstrap net, Iterative net, Refinement net.
Первая часть (Bootstrap net) состоит из двух подряд идущих блоков энкодер-декодер. Первый блок предсказывает по паре изображений оптический поток с картой уверенности в нем, а также сдвиг камеры. Вторая по данному потоку, сдвигу камеры, первому изображению и второму, сдвинутому на величину оптического потока, предсказывает карту глубины и карту нормалей.
Вторая часть (Iterative net) улучшает предсказания глубины и оптического потока путем нескольких прогонов через полносверточную сеть, состоящую из двух блоков энкодер-декодер. Архитектура этих блоков такая же, как в Bootstrap net.
Третья часть (Refinement Net) преобразует полученную в низком разрешении карту глубин в разрешение исходного изображения. Архитектура сети Refinement net состоит из одного блока энкодер-декодер и принимает на вход первое изображение из входной пары и полученную на выходе Iterative Net карту глубины.
Время обработки одной пары изображений с помощью данной архитектуры составляет 110 мс на видеокарте GTX Titan. На маломощных бортовых компьютерах малых роботов время обработки одной пары изображений может достигать нескольких секунд, поэтому данная архитектура не может применяться в задаче одновременного картирования и локализации в реальном времени на малом роботе.

4.3 Восстановление глубины по одиночному изображению с использованием коллекции данных

В связи с бурным развитием вычислительной техники и нейросетевых алгоритмов обработки изображений в последнее время стали популярны методы восстановления карт глубин по единственному изображению с помощью нейронных сетей. В таких методах обычно используются полносверточные нейронные сети, состоящие из сверточной части (энкодера), преобразующей входное изображение в карту высокоуровневых признаков, и развертночной части (декодера), строящей по карте высокоуровневых признаков карту глубины.
Одна из архитектур нейросети для восстановления глубины по изображению описана в [14]. Нейросеть имеет полносверточную архитектуру, основанную на архитектуре ResNet50 [9]. Особенностью данной архитектуры является использование проекций в декодере, а также использование операции Interleaving, позволяющей ускорить вычисления в блоках развертки без потери качества. Схема архитектуры представлена на рисунке 8. Схема метода Interleaving представлена на 9.

Эксперименты на коллекциях данных NYU depth [22] и Make3D [1], содержащих видеосцены, снятые в помещениях и в городской среде, показали, что по средней ошибке предсказания данная нейросеть опережает классические методы восстановления глубины по видеопотоку.

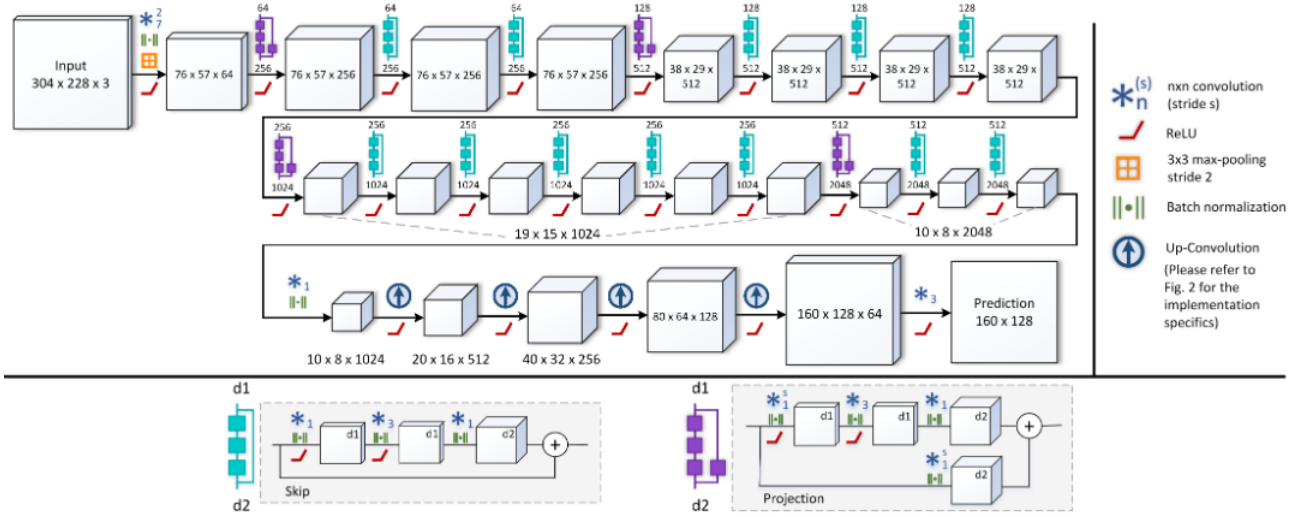


Рис. 8. Архитектура нейросети, описанной в работе [14]

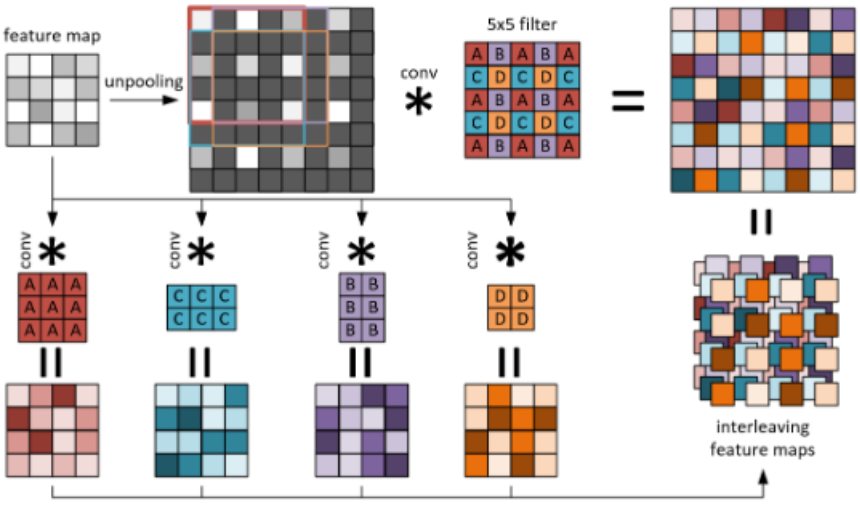


Рис. 9. Операция Interleaving и ее эквивалентность операциям Unpooling+convolution

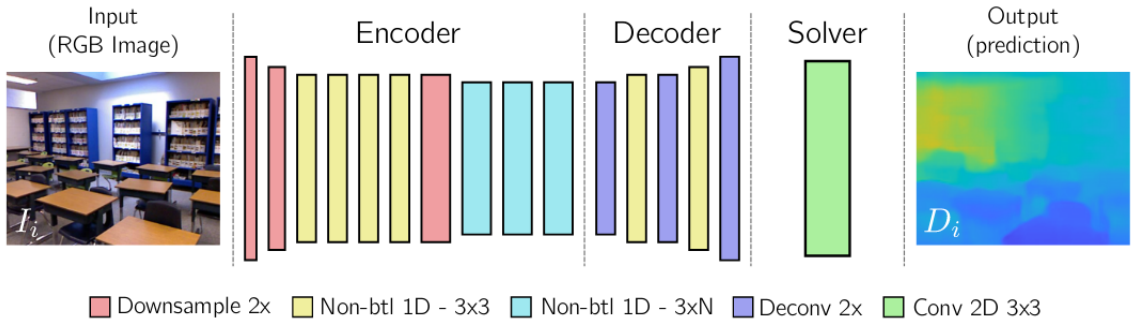


Рис. 10. Архитектура нейросети, описанной в работе [23]

По заверениям авторов, время обработки одного изображения с помощью данной нейросети составило 55 мс на видеокарте GTX Titan. Этого достаточно для обработки видео со скоростью 18 кадров в секунду, но на роботах, не имеющих такую мощную видеокарту на борту, изображения будут обрабатываться намного медленнее, что затрудняет использование данной архитектуры в задаче SLAM в реальном времени.
В статье [23] описана более быстрая нейросетевая архитектура для восстановления карты глубины по изображению. Схема архитектуры представлена на рисунке **??**.

Для достижения приемлемого качества восстановления глубины был использован подход Knowledge transfer - обучение с использованием выходов другой, более мощной нейросети. Эксперименты показали, что относительная ошибка на датасете NYU Depth v2 составляет 19%, а скорость обработки видеопотока в разрешении 320x240 - до 30 кадров в секунду на встраиваемом компьютере NVIDIA Jetson TX2. Такая скорость позволяет обрабатывать видеопоток в реальном времени с помощью бортового компьютера небольшого робота, но из-за низкой точности восстановления глубины при применении данной архитектуры в задаче SLAM могут возникнуть проблемы.

4.4 Выводы

Для восстановления карт глубин изображений по видеопотоку могут применяться как классические алгоритмы, основанные на извлечении структуры из движения с помощью геометрических преобразований, так и нейросетевые методы, основанные на извлечении закономерностей из предварительно собранной коллекции данных. Эксперименты на коллекциях видеоданных показали, что нейросетевые методы обладают в целом более высокой точностью восстановления глубины, чем классические.
С помощью нейросетей можно восстанавливать карты глубин как по паре последовательных кадров, так и по одному кадру из видеопотока. Использование последовательности кадров повышает точность, но сильно снижает скорость восстановления глубины. Некоторые архитектуры восстановления глубины по одному изображению обладают достаточной скоростью для обработки видеопотока в реальном времени даже на небольших маломощных компьютерах.
В данной работе для применения в задаче SLAM был выбран нейросетевой метод, восстанавливающий карту глубины по каждому кадру изображения отдельно, так как он обладает одновременно большой скоростью работы и высокой точностью предсказания глубины.

Глава 5

Восстановление карты глубины по видеопотоку с помощью полносверточных нейронных сетей

5.1 Архитектуры

Используемые в работе нейросети принимают на вход цветное трехканальное изображение и выдают предсказанную карту глубины. Карта глубины представляет собой двумерный массив той же ширины и высоты, что и входное изображение, в каждой ячейке которого находится глубина соответствующего пикселя входного изображения. Глубина задается положительным числом с плавающей точкой.

Нейросети состоят из двух частей: сверточной (энкодер) и разверточной (декодер). Также в некоторых из рассмотренных архитектур присутствуют проекции из энкодера в декодер (shortcuts). Сверточная часть содержит серию слоев свертки (convolution) и субдискретизации (pooling) и последовательно уменьшает размерность изображения, преобразуя его в набор высокоуровневых признаков. Разверточная часть преобразует набор высокоуровневых признаков, предсказанных энкодером, в карту глубины. Схемы архитектур рассмотренных в работе нейросетей изображены на рисунке 11.

В данной работе в качестве энкодера используется сверточная часть архитектуры ResNet-50 [9], предобученная на коллекции изображений ImageNet [5]. Декодер, рассмотренный в данной работе, начинается со свертки с ядром размера 1x1 и 1024 фильтрами. Затем следуют 5 блоков развертки, последовательно повышающие размерность карты признаков до размерности исходного изображения. После блоков развертки следует сверточный слой с одним фильтром, выводящий предсказанную карту глубины. Схема декодера изображена на рисунке 11.

В качестве блоков развертки были рассмотрены следующие структуры: **Deconv**, **Upsampling + nonbt**, **Up-convolution** и его модификация **Interleaving**. Схемы рассмотренных структур изображены на рисунке 12, а их подробное описание представлено ниже.

Блок **Deconv** состоит из слоев нормализации (BatchNormalization), транспонированной свертки (Deconvolution [26]) с шагом 2 и ядрами размера 5x5, активации (ReLU).

Кажджый блок **Upsampling + nonbt** содержит слой апсемплинга (Upsampling), увеличивающий ширину и высоту карты признаков в 2 раза, и блок Non-bottleneck [20]. Блок Non-bottleneck состоит из двух пар сверток с ядрами 3x1 и 1x3, разделенных слоем нормализации (Batch Normalization).

Каждый блок **Up-convolution** состоит из слоев нормализации (BatchNormalization), повышения размерности (Upsampling или Unpooling), свертки с ядром 5x5 и активации (ReLU).

Блок **Interleaving** содержит слои нормализации (BatchNormalization), Interleaving [14] и актива-

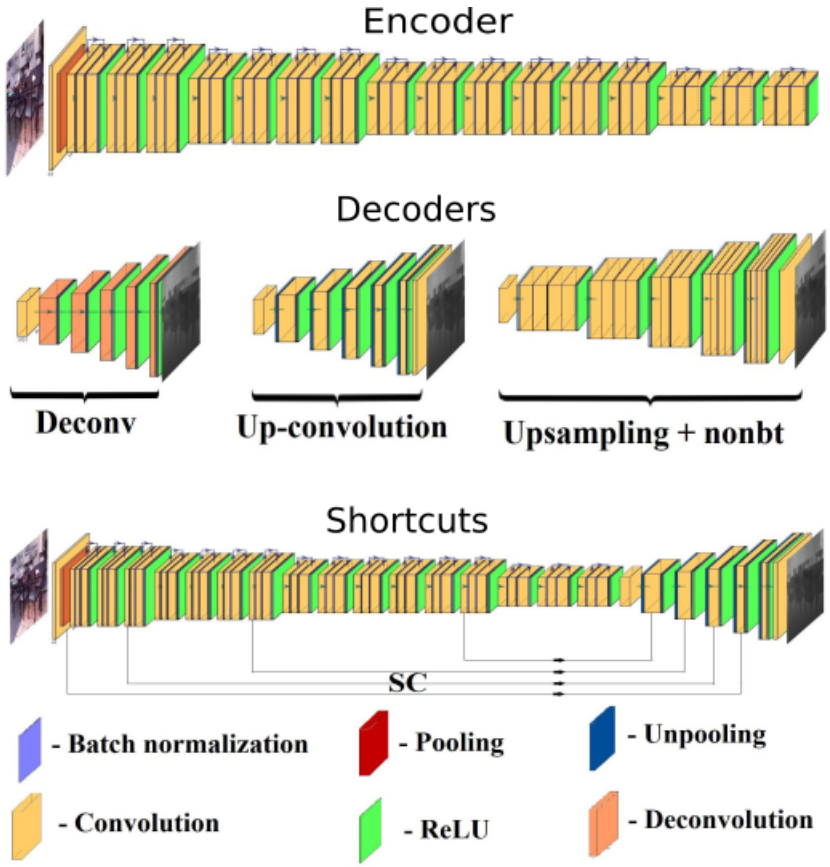


Рис. 11. Схемы архитектур используемых в работе нейросетей

ции (ReLU). Операция Interleaving является ускоренной версией операции Up-Convolution (Unpooling + 5x5 Convolution). Она состоит из четырех сверток с ядрами размеров 3x3, 2x3, 3x2 и 2x2 и соединения четырех карт признаков, полученных после применения этих сверток, в одну. Схема операции Interleaving и ее эквивалентность операции Up-Convolution показаны на рисунке 9.

5.2 Данные для обучения

Обучение нейросетей производилось на коллекции NYU Depth Dataset v2 [22], содержащей около 1100 видеосцен и более 400 тысяч кадров, снятых в помещениях различного типа, с размеченными с помощью сенсора Microsoft Kinect картами глубины. Глубина изображений ограничивалась 10 метрами. Коллекция была разделена на обучающую и валидационную выборки. Валидационная выборка содержала кадры из 15 сцен, а обучающая - кадры из всех остальных сцен коллекции NYU Depth.

Так как разметка с помощью сенсора Microsoft Kinect содержала пробелы в картах глубин, возникла необходимость в предобработке данных. Карты глубин всех изображений коллекции были предобработаны с помощью билатерального фильтра [24]. Также во избежание проблемы переобучения нейросетей ко всем изображениям из обучающей выборки были применены преобразования, такие, как повороты на небольшие углы и зеркальные отражения.

5.3 Функции потерь

Для обучения нейросетей применялись следующие функции потерь:

- **MSE**: $L(y, \widehat{y}) = (y - \widehat{y})^2$

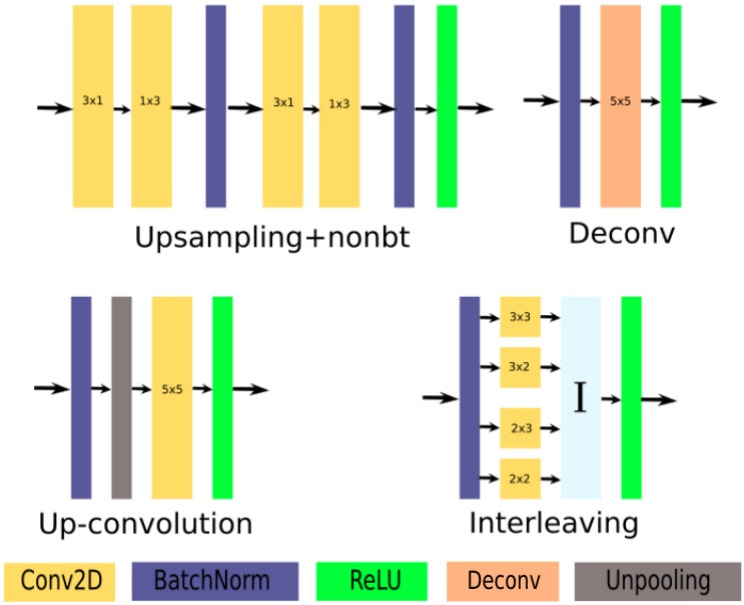


Рис. 12. Схемы используемых блоков декодера

- **Relative Squared Error**: $L(y, \widehat{y}) = (\frac{y-\widehat{y}}{y})^2$
- **MSE + Rel**: $L(y, \widehat{y}) = (y - \widehat{y})^2(1 + \frac{2}{y^2})$
- **BerHu**: $L(y, \widehat{y}) = \begin{cases} |y - \widehat{y}|, & |y - \widehat{y}| < \delta \\ \frac{(y-\widehat{y})^2+\delta^2}{2\delta}, & |y - \widehat{y}| \geq \delta \end{cases}$

Здесь y - истинная глубина пикселя на изображении, $\widehat{y}$ - предсказанная. Во время обучения функция потерь вычислялась попиксельно и усреднялась по всем пикселям всех изображений батча.

С помощью экспериментов было выяснено, что наилучшие результаты получаются при использовании функции потерь MSE + Rel, а наихудшие - при использовании Relative Squared Error.

5.4 Метрики качества

Качество восстановления глубины оценивалось по следующим метрикам:

- **MSE**: $L(y, \widehat{y}) = (y - \widehat{y})^2$
- **Relative error**: $L(y, \widehat{y}) = \frac{|y-\widehat{y}|}{y}$
- $\boldsymbol{\delta_1, \delta_2, \delta_3}$: $\delta_k(y, \widehat{y}) = I\{\max(\frac{y}{\widehat{y}}, \frac{\widehat{y}}{y}) < 1.25^k\}$

Здесь y - истинная глубина пикселя на изображении, $\widehat{y}$ - предсказанная. Во время обучения функция потерь вычислялась попиксельно и усреднялась по всем пикселям всех изображений выборки.

5.5 Параметры обучения

Обучение нейросетей производилось на подвыборке, содержащей 10% коллекции NYU Depth. Полная коллекция изображений для обучения не использовалась, так как в ней содержится в среднем по 1000 изображений из одной видеосцены, и из-за большого количества похожих изображений нейросеть быстро переобучалась. Для оптимизации параметров нейросети использовался метод Adam [11] с начальным шагом 10^{-4} и гиперболическим затуханием (inverse-time decay). Обучение проводилось в 30 эпох. На каждой эпохе обучающая выборка разбивалась на батчи размера 16,

которые поочередно подавались на вход нейросети. После окончания эпохи вычислялось значение метрик качества на валидационной выборке, и веса нейросети сохранялись на диск. Для экспериментов в реальном времени использовались веса, сохраненные после той эпохи, на которой качество на валидационной выборке было наилучшим.

5.6 Программно-аппаратное обеспечение

5.6.1 Обучение

Архитектура нейросети была построена с использованием библиотеки глубинного обучения TensorFlow и ее расширения Keras. Программный код построения архитектуры был реализован на языке Python. Для борьбы с переобучением после каждого блока развертки был включен слой дропаута (Dropout) с коэффициентом 0.5. Обучение нейросетей проводилось на гибридном высокопроизводительном вычислительном кластере ФИЦ ИУ РАН [1].

5.6.2 Тестирование в реальном времени

Обученные на кластере ФИЦ ИУ РАН архитектуры тестировались в реальном времени на встраиваемом компьюютере NVIDIA Jetson TX2 [8]. Данный компьютер оснащен 256-ядерной видеокартой с архитектурой PASCAL и 6-ядерным центральным процессором CPU Complex ARMv8 и имеет 8 ГБ оперативной памяти, разделяемой между GPU и CPU. При этом он 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 TensorRT, 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 записывались 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 исходное изображение отрисовывались на экране с помощью библиотеки OpenGL.

5.7 Результаты

Все архитектуры, описанные в разделе 5.1, были обучены и протестированы на коллекции NYU Depth v2 [22]. Детали обучения описаны в предыдущих разделах. Примеры восстановления глубины на изображениях из валидационной выборки представлены на рисунке 13.

Описанные в разделе 5.1 архитектуры также были протестированы на платформе NVIDIA Jetson TX2 [8] с использованием библиотеки TensorRT. При тестировании осуществлялось восстановление карт глубин изображений, поступающих в реальном времени с установленной на платформе видеокамеры. Примеры восстановления глубины в реальном времени на NVIDIA Jetson представлены на

[1] Федеральный исследовательский центр Информатика и управление РАН [Электронный ресурс]: сайт. – Москва: ФИЦ ИУ РАН. – URL: http://hhpcc.frccsc.ru (дата обращения: 12.09.2018)

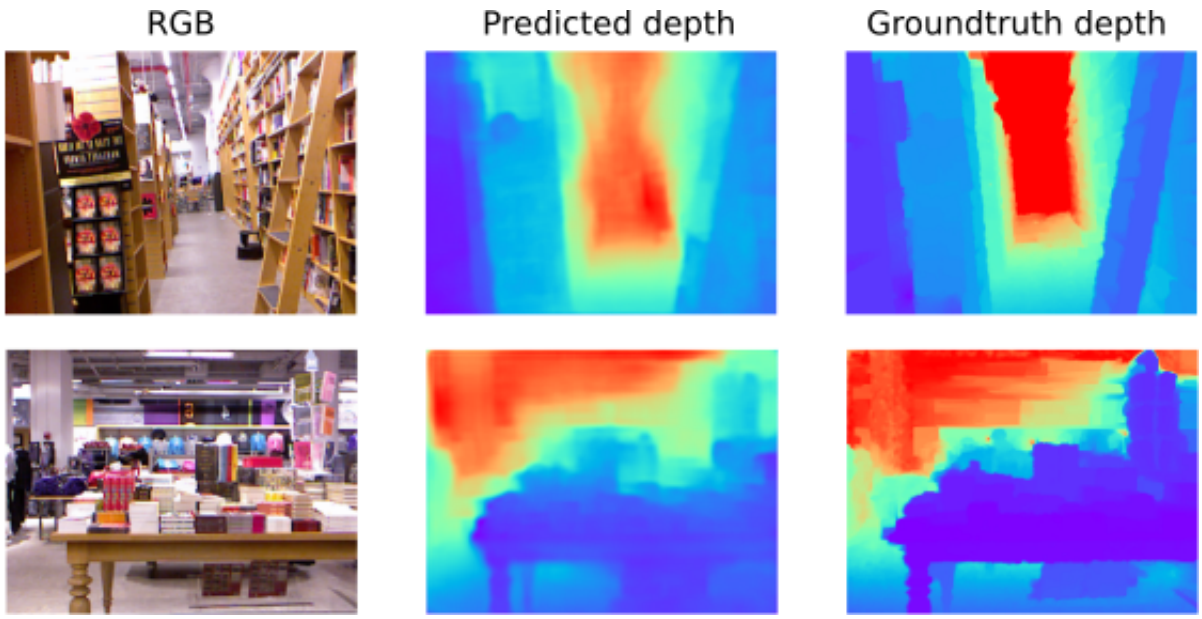


Рис. 13. Визуализация предсказаний нейросети на изображениях из коллекции NYU Depth v2

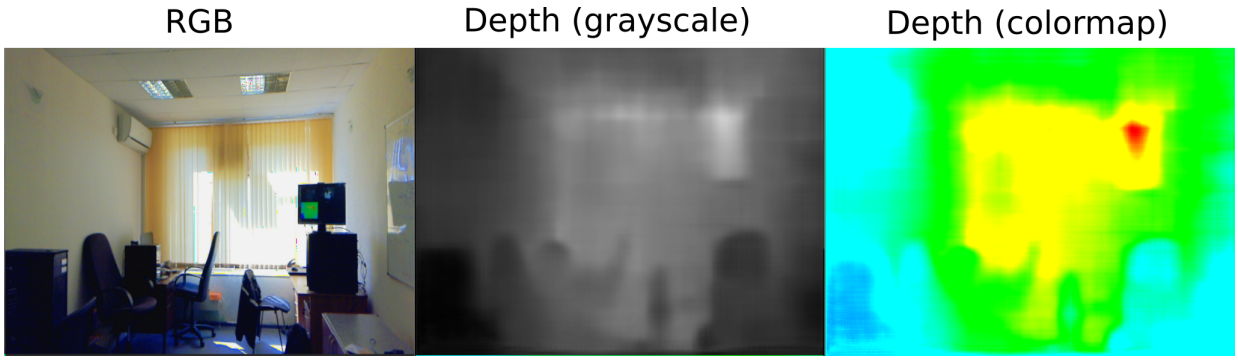


Рис. 14. Пример восстановления глубины по изображниям с видеокамеры в реальном времени

рисунке 14. Значения метрик качества на NYU Depth v2 и скорости работы нейросетей на NVIDIA Jetson TX2 представлены в таблице 1.

Таблица 1. Сравнение архитектур по скорости и качеству

Architecture	MSE	Rel	δ^1	δ^2	δ^3	Jetson time (ms)
Deconv	0.467	0.186	0.718	0.929	0.980	78
Nonbt+shortcuts	0.419	0.173	0.748	0.944	0.987	61
Interleaving+shortcuts	0.445	0.178	0.714	0.939	0.987	62
Laina et al. [14]	0.328	0.127	0.811	0.953	0.988	82
Spek et al. [23]	0.472	0.190	0.704	0.917	0.977	33

В таблице приведено сравнение результатов, полученных в данной работе, с результатами других работ по восстановлению карт глубин изображений. Выяснилось, что исследованные архитектуры позволяют достичь более высокой скорости восстановления глубины, чем в [14], и более высокого качества, чем в [23].

5.8 Выводы

В данной работе были представлены архитектуры нейронных сетей для восстановления карт глубин изображений, полученных с единственной видеокамеры в реальном времени. Представленные нейросети были обучены и протестированы на коллекции изображений NYU Depth v2. Также было проведено тестирование нейросетей в реальном времени на платформе NVIDIA Jetson TX2. Сравнение показало, что представленные в данной работе архитектуры не уступают по скорости и качеству восстановления глубины другим современным архитектурам. Относительная ошибка предсказания глубины на коллекции NYU Depth v2 составила 18%, а скорость обработки изображений на NVIDIA Jetson - 16 кадров в секунду. Такая скорость работы в совокупности с приемлемым качеством восстановления глубины делает возможным применение данных нейросетей в методах одновременного картирования и локализации, в том числе для навигации малых беспилотных летательных аппаратов.

Глава 6

Одновременное картирование и локализация с использованием восстановления карт глубины

6.1 Описание алгоритма

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, описанной в разделе 5.1. По изображениям и предсказанным картам глубин осуществляется картирование и локализация с помощью метода RTAB-Map, описанного в разделе 3.5. Схема алгоритма представлена на рисунке 15.

Алгоритм реализован на языке программирования C++ с использованием библиотеки Robotic Operation Systems (ROS) [19]. Исходный код алгоритма в формате ROS Node, а также параметры запуска доступны в репозитории [1].

Для восстановления карт глубин использовалась нейросеть с декодером Upsampling + nonbt (см. 5.1). Для повышения скорости работы алгоритма RTAB-Map использовался детектор особых точек ORB [21] вместо установленного по умолчанию GFTT. Из каждого изображения извлекалось 2000 особых точек. Минимальное количество особых точек для сопоставления изображений было установлено равным 10 вместо установленных по умолчанию 20, так как совпадения происходили реже, чем при проведении SLAM по датчикам глубин, из-за приближенного восстановления глубины.

6.2 Результаты

Описанный в разделе 6.1 алгоритм был опробован в реальном времени на платформе NVIDIA Jetson TX2, а также на ноутбуке Lenovo Z50. Эксперименты проводились в различных помещениях и коридорах зданий. Видеозапись одного из экспериментов на платформе NVIDIA Jetson TX2 доступна по ссылке [2]. Пример построенной алгоритмом карты (коридоры здания ИСА РАН) представлен на рисунке 16.

Эксперименты показали, что представленный алгоритм работает с приемлемой скоростью (карта перестраивается 5-10 раз в секунду) и позволяет получить правдоподобные карты помещений. Также данный алгоритм оказался способен не терять одометрию при поворотах на месте, в отличие

[1] https://github.com/cnndepth
[2] https://www.youtube.com/watch?v=umuBjIwA2bU

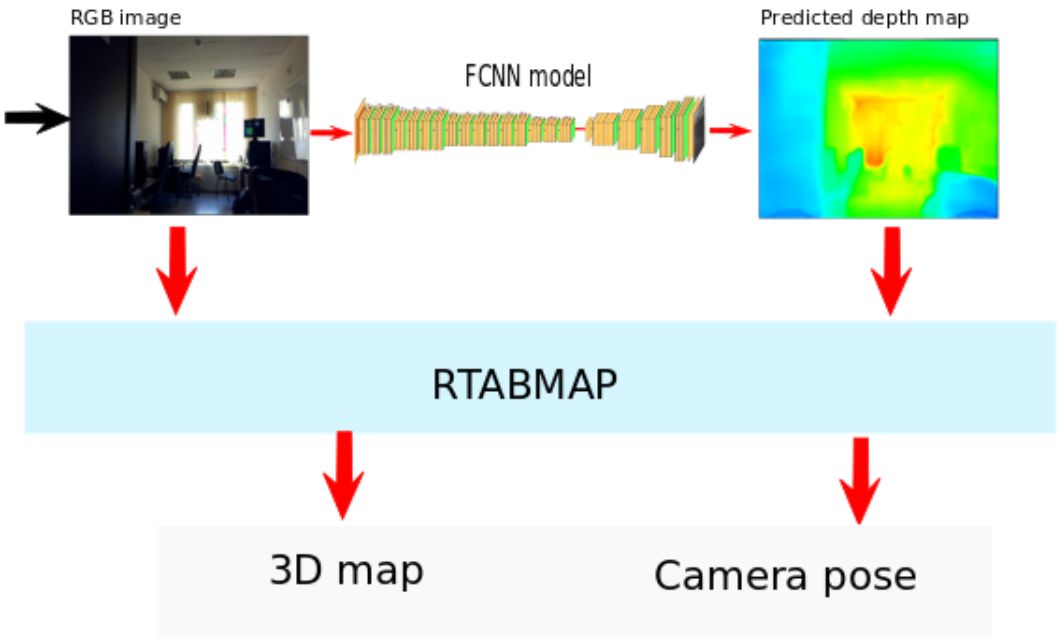


Рис. 15. SLAM на основе восстановления глубины с помощью нейросетей

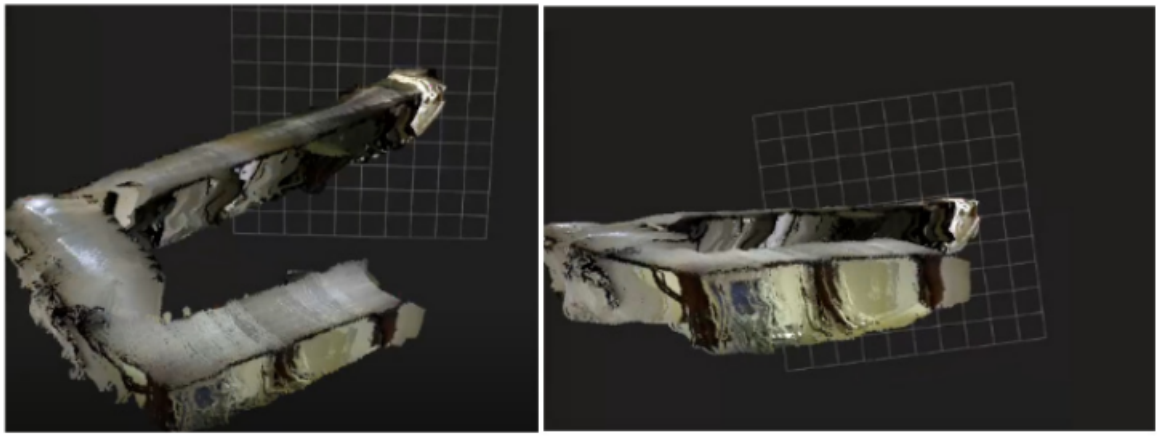


Рис. 16. Карта коридора, построенная алгоритмом SLAM. Вид сверху и сбоку

от алгоритма ORB-SLAM. Исключение составили повороты, при которых в поле зрения камеры попадала только однотонная стена без ориентиров.

6.3 Выводы

В данной работе представлен алгоритм решения задачи vSLAM по данным с единственной видеокамеры в реальном времени, основанный на восстановлении карт глубин изображений с помощью нейронной сети. Представленный алгоритм протестирован на встраиваемом компьютере NVIDIA Jetson TX2 в реальном времени. Эксперименты в различных помещениях показали, что данный алгоритм обладает рядом значительных преимуществ по сравнению с популярными методами решения задачи vSLAM - он строит плотную правдоподобную карту окружающей местности с известным масштабом и не теряет одометрию при поворотах на месте. Скорость картирования и локализации и плотность построенной карты делают возможным применение разработанного алгоритма для навигации и планирования траектории автономного робота.

Глава 7

Заключение

В данной работе представлен алгоритм решения задачи одновременного картирования и локализации по данным с единственной видеокамеры. Алгоритм основан на восстановлении карт глубин изображений с помощью нейронной сети и методе одновременного картирования и локализации RTAB-Map. Для подтверждения работоспособности алгоритма были проведены эксперименты в реальном времени на встраиваемом компьютере. Эксперименты показали, что данный алгоритм имеет значительные преимущества перед традиционными методами решения задачи vSLAM. Плотная правдоподобная карта окружающей местности, построенная алгоритмом, может быть использована для навигации и планирования траектории мобильного робота.

Программная реализация алгоритма интегрирована с библиотекой Robot Operation System (ROS) и выложена в открытый доступ. Исходный код алгоритма, а также параметры запуска доступны в репозитории [1].

[1] https://github.com/cnndepth

Список литературы

[1] Saxena Ashutosh, Sun Min, Andrew Y Ng. “Make3d: Learning 3d scene structure from a single still image”. *IEEE Transactions on Pattern Analysis & Machine Intelligence* **31** 5 (2009), с. 824—840.

[2] Herbert Bay, Tinne Tuytelaars, Luc Van Gool. “Surf: Speeded up robust features”. *European conference on computer vision*. Springer. 2006, с. 404—417.

[3] Eric Brachmann и др. “DSAC-differentiable RANSAC for camera localization”. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, с. 6684—6692.

[4] Michael Calonder и др. “Brief: Binary robust independent elementary features”. *European conference on computer vision*. Springer. 2010, с. 778—792.

[5] Jia Deng и др. “Imagenet: A large-scale hierarchical image database”. *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, с. 248—255.

[6] David D Diel, Paul DeBitetto, Seth Teller. “Epipolar constraints for vision-aided inertial navigation”. *2005 Seventh IEEE Workshops on Applications of Computer Vision (WACV/MOTION'05)-Volume 1*. Т. 2. IEEE. 2005, с. 221—228.

[7] Jakob Engel, Thomas Schöps, Daniel Cremers. “LSD-SLAM: Large-scale direct monocular SLAM”. *European conference on computer vision*. Springer. 2014, с. 834—849.

[8] D Franklin. “NVIDIA Jetson TX2 Delivers Twice the Intelligence to the Edge”. *NVIDIA Accelerated Computing| Parallel Forall* (2017).

[9] 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.

[10] Kevin Karsch, Ce Liu, Sing Bing Kang. “Depth transfer: Depth extraction from video using non-parametric sampling”. *IEEE transactions on pattern analysis and machine intelligence* **36** 11 (2014), с. 2144—2158.

[11] Diederik P Kingma, Jimmy Ba. “Adam: A method for stochastic optimization”. *arXiv preprint arXiv:1412.6980* (2014).

[12] Jan J Koenderink, Andrea J Van Doorn. “Affine structure from motion”. *JOSA A* **8** 2 (1991), с. 377—385.

[13] 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.

[14] 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.

[15] Ce Liu, Jenny Yuen, Antonio Torralba. “Sift flow: Dense correspondence across scenes and its applications”. *IEEE transactions on pattern analysis and machine intelligence* **33** 5 (2010), с. 978—994.

[16] David G Lowe и др. “Object recognition from local scale-invariant features.” *iccv*. Т. 99. 2. 1999, с. 1150—1157.

[17] 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.

[18] Aude Oliva, Antonio Torralba. “Modeling the shape of the scene: A holistic representation of the spatial envelope”. *International journal of computer vision* **42** 3 (2001), с. 145—175.

[19] Morgan Quigley и др. “ROS: an open-source Robot Operating System”. *ICRA workshop on open source software*. Т. 3. 3.2. Kobe, Japan. 2009, с. 5.

[20] Eduardo Romera и др. “Erfnet: Efficient residual factorized convnet for real-time semantic segmentation”. *IEEE Transactions on Intelligent Transportation Systems* **19** 1 (2017), с. 263—272.

[21] Ethan Rublee и др. “ORB: An efficient alternative to SIFT or SURF.” *ICCV*. Т. 11. 1. Citeseer. 2011, с. 2.

[22] Nathan Silberman, Rob Fergus. “Indoor scene segmentation using a structured light sensor”. *2011 IEEE international conference on computer vision workshops (ICCV workshops)*. IEEE. 2011, с. 601—608.

[23] Andrew Spek, Thanuja Dharmasiri, Tom Drummond. “CReaM: Condensed Real-time Models for Depth Prediction using Convolutional Neural Networks”. *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, с. 540—547.

[24] Carlo Tomasi, Roberto Manduchi. “Bilateral filtering for gray and color images.” *Iccv*. Т. 98. 1. 1998, с. 2.

[25] 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.

[26] Matthew D Zeiler и др. “Deconvolutional networks.” *Cvpr*. Т. 10. 2010, с. 7.

[27] Zhengyou Zhang, Ying Shan. *Incremental motion estimation through local bundle adjustment*. US Patent 6,996,254. Февр. 2006.