

Санкт-Петербургский государственный университет

КИСЕЛЁВ Михаил Николаевич

Выпускная квалификационная работа

Исследование производительности
нейросетевых компонент в
SLAM-алгоритмах на встраиваемых
устройствах

Уровень образования: бакалавриат

Направление *02.03.03 «Математическое обеспечение и администрирование
информационных систем»*

Основная образовательная программа *СВ.5162.2021 «Технологии программирования»*

Научный руководитель:
старший преподаватель кафедры системного программирования, к. т. н., Литвинов Ю. В.

Консультант:
руководитель исследовательской группы, лаборатория «ИИ для Автономных Систем»,
ААНО ВО «Сколковский Институт Науки и Технологий», Корнилова А. В.

Рецензент:
ведущий научный сотрудник, МФТИ, д.ф.-м.н. Парсегов С. Э.

Санкт-Петербург
2025

Saint Petersburg State University

Mikhail Kiselev

Bachelor's Thesis

Performance benchmark for neural network components of SLAM on embedded devices

Education level: bachelor

Speciality *02.03.03 "Software and Administration of Information Systems"*

Programme *CB.5162.2021 "Programming Technologies"*

Scientific supervisor:
C.Sc., senior lecturer Y.V. Litvinov

Consultant:
head of research group at "AI for Autonomous Systems" lab. at "Skolkovo Institute of Science and Technology", A.V. Kornilova

Reviewer:
Sc.D., senior research scientist at MIPT S.E. Parsegov

Saint Petersburg
2025

Оглавление

Введение	4
1. Постановка задачи	6
2. Обзор	7
2.1. Выделение глобальных дескрипторов, выделение и ассо- циация локальных особенностей в рамках визуального SLAM	7
2.2. Используемые метрики	9
2.3. Экспериментальные платформы	10
2.4. Техники квантования нейронных сетей	13
3. Реализация инструмента для измерения производитель- ности алгоритмов	17
3.1. Database и Queries	18
3.2. Method, FeatureDetector, FeatureMatcher, VprSystem . .	18
3.3. Backend	18
3.4. Экспорт модели в другие форматы	19
3.5. RknnExportable и TorchscriptExportable	19
3.6. Преобразование моделей	19
4. Эксперимент	21
4.1. Цель экспериментального исследования	21
4.2. Условия эксперимента	22
4.3. Результаты	23
4.4. Угрозы нарушения корректности	25
Заключение	30
Список литературы	31

Введение

Автономные системы, такие как промышленные роботы и беспилотные транспортные средства, быстро развиваются и становятся неотъемлемой частью современной жизни. Ключевым элементом работы таких устройств являются SLAM-алгоритмы (Simultaneous Localization and Mapping), которые в режиме реального времени обеспечивают возможность определения местоположения и создания карты окружающей среды. Они состоят из двух взаимосвязанных частей: фронтенда, задача которого состоит в выделении ориентиров из исходных данных с датчиков автономной системы, и бэкенда, производящего построение и обновление карты на основе этих ориентиров, а также оценку местоположения системы относительно них.

Нейронные сети находят все большее применение в качестве компонентов фронтенда SLAM-алгоритмов [10] (рис. 1) из-за более высокой точности и устойчивости по сравнению с классическими подходами. В частности, в новейших SLAM-системах приобретают популярность нейросетевые методы

- выделения глобальных дескрипторов (descriptor extraction), которые используются для быстрого поиска похожих изображений по базе данных;
- методы выделения и ассоциации локальных особенностей изображения (feature extraction), которые используются для сопоставления двух изображений.

Однако применение таких методов может быть сопряжено с высокими требованиями к вычислительным ресурсам используемой платформы. Многие автономные системы, такие как беспилотные летательные аппараты, имеют ограничения на массу, габариты, а также энергопотребление контролирующего устройства. Из-за этих требований вычислительные ресурсы, доступные для использования SLAM-алгоритмов, также ограничены.

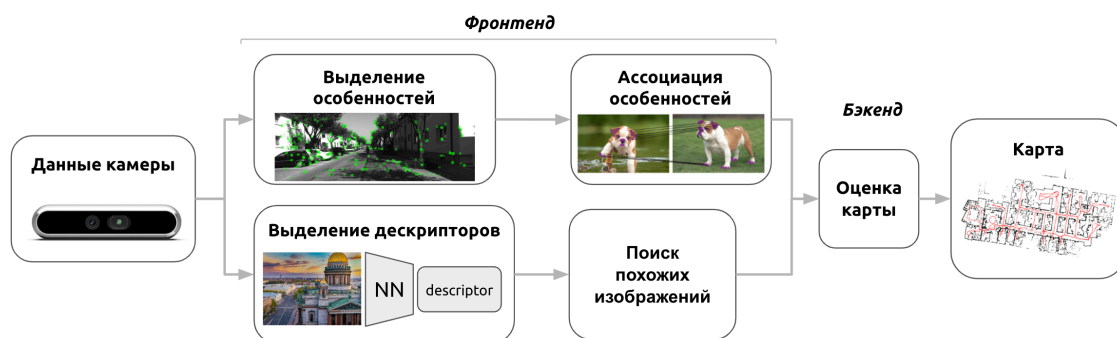


Рис. 1: Принцип работы SLAM-алгоритмов.

Ввиду большого числа подобных алгоритмов подбор подходящего по вычислительным требованиям для данной платформы и оптимального по точности сводится к ручному перебору. Кроме того, наличие способов оптимизации производительности нейронных сетей за счет уменьшения их точности еще сильнее расширяет множество потенциальных вариантов.

Одна из основных областей работы Лаборатории «Искусственного Интеллекта для Автономных Систем» ААНО ВО «Сколковский Институт Науки и Технологий» — разработка автономных решений, использующих SLAM-алгоритмы, в рамках ограничений на массу и габариты платформы. Выбор базовой платформы для каждого решения влияет на весь дальнейший процесс разработки. На это решение в числе прочего влияют алгоритмы, которые планируется использовать. Чтобы облегчить принятие этого решения, необходима разработка инструмента, позволяющего выполнять измерения производительности нейросетевых компонент SLAM-алгоритмов на целевых экспериментальных платформах, а также производство измерений на уже имеющихся устройствах для существующих алгоритмов.

1. Постановка задачи

Целью работы является исследование производительности популярных методов детекции и ассоциации локальных особенностей и выделения глобальных дескрипторов, основанных на нейронных сетях, на встраиваемых одноплатных компьютерах с аппаратными ускорителями для нейронных сетей. Для её выполнения были поставлены следующие задачи.

1. Выполнить обзор методов выделения и ассоциации локальных особенностей и выделения глобальных дескрипторов, основанных на нейронных сетях.
2. Выполнить обзор распространенных встраиваемых платформ с ускорителями для запуска нейронных сетей.
3. Выполнить обзор техник квантования нейронных сетей.
4. Реализовать инструмент для измерения производительности алгоритмов.
5. Произвести измерения производительности и точности алгоритмов, а также их сжатых версий на имеющихся у Лаборатории платформах.

2. Обзор

В данном разделе будет проведен обзор существующих методов выделения дескрипторов, методов выделения и ассоциации особенностей, наборов данных и используемых для измерения метрик. Кроме того, будут описаны распространенные встраиваемые платформы с ускорителями запуска нейронных сетей.

2.1. Выделение глобальных дескрипторов, выделение и ассоциация локальных особенностей в рамках визуального SLAM

Одним из ключевых этапов работы SLAM-алгоритмов является распознавание уже посещенных мест (loop closure detection — распознавание замыкания цикла). Это позволяет системе проводить уточнения оценки карты, а также производить локализацию в случае потери отслеживания местоположения. Для быстрого сравнения новых данных с уже имеющимися в базе данных используются глобальные дескрипторы, которые представляют собой компактное представление всего изображения [17, 19, 11, 1, 2, 5]. Кроме того, этот подход может комбинироваться с методами выделения и ассоциации локальных особенностей: после поиска похожих изображений с помощью глобальных дескрипторов, система может использовать локальные особенности для уточнения результатов [4, 10, 6, 19, 23, 22]. Помимо этого, выделение локальных особенностей и их ассоциация используются для оценки перемещения системы в пространстве на основе двух изображений, что также является важным этапом работы SLAM-алгоритмов.

2.1.1. Методы выделения дескрипторов

Выделение дескрипторов — это преобразование изображения в векторное представление одинаковой длины, которое позволяет осуществлять быстрый поиск похожих изображений. Методы, выполняющие эту задачу, можно разделить на два типа: методы, основанные на глобо-

ком обучении и преобразовании изображения в вектор в латентном пространстве, и методы, основанные на классических подходах, таких как SIFT [15], SURF [20], ORB [18], выделяющих локальные особенности, с дальнейшим использованием мешка слов (Bag of Words) для получения глобального дескриптора изображения. Первый тип методов выгодно отличается более высокой точностью и устойчивостью к шумам, но требует больших вычислительных ресурсов.

Среди нейросетевых методов выделения дескрипторов можно выделить следующие: NetVLAD [17], CosPlace [5], EigenPlaces [11], AnyLoc [2], MixVPR [1],

NetVLAD [17] — это один из первых методов, использующих нейронные сети для задачи выделения дескрипторов. В основе этого подхода лежит использование слоя VLAD [3] для агрегации карты признаков, полученной на выходе сверточной нейронной сети.

Методы CosPlace [5] и EigenPlaces [11] представляют тренировку модели как задачу классификации в отличие от других методов, использовавших более вычислительно затратное контрастивное обучение. Кроме того, они используют модели серии ResNet [8] в качестве опорной сети (то есть сети более общего назначения, которая используется для изначального преобразования изображения с дальнейшим уточнением с помощью специализированного набора слоев — агрегатора). Эти два фактора обеспечивают масштабируемость и высокую скорость обучения.

В методе AnyLoc [2], для выделения признаков используется foundation model — готовая предобученная модель общего назначения. Это существенно снижает количество вычислительных ресурсов, необходимых для обучения. В данном случае используются модели DINO [12] и DINOv2 [9]. Эти модели, в отличие от классических подходов, основаны на визуальных трансформерах, а не на сверточных сетях (CNN). Благодаря механизмам self-attention трансформеры более эффективны в выделении сложных глобальных зависимостей в данных по сравнению с CNN, которые обладают более локальным видением. Однако трансформеры более тяжеловесные и требуют больше вычис-

лительных ресурсов.

Метод MixVPR [1] подобно CosPlace и EigenPlaces использует модели ResNet в качестве опорной сети, однако в отличие от них, использует не только конечный выход сети, но и данные, полученные на промежуточных слоях, что позволяет улучшить качество дескрипторов. Ограничение данного метода состоит в том, что он поддерживает только работу с изображениями разрешением 320×320 пикселей.

2.1.2. Методы выделения и ассоциации локальных особенностей

Наиболее широко применяемыми методами ассоциации локальных особенностей являются SuperGlue [21] и LightGlue [14].

В рамках метода SuperGlue [21] задача ассоциации локальных особенностей представляется как задача оптимального транспорта — проблема поиска наиболее эффективного способа сопоставления визуальных признаков, решение которой происходит с помощью графовой нейронной сети. Авторы статьи, посвященной методу LightGlue [14], развивают эту идею, принося возможность быстрого предсказания на интуитивно схожих парах изображений.

Оба этих метода используют сверточную нейронную сеть SuperPoint [7] для выделения локальных особенностей.

Недавней разработкой в области выделения и ассоциации локальных особенностей является метод XFeat+LighterGlue [25] (XFeat — для выделения локальных особенностей, LighterGlue — для ассоциации). Архитектурный пересмотр модели машинного обучения и оптимизация под работу в условиях ограниченных вычислительных ресурсов позволили улучшить производительность как выделения, так и ассоциации локальных особенностей.

2.2. Используемые метрики

Для оценки производительности методов необходимо измерить пропускную способность, то есть измерить количество изображений, кото-

рое модель может обрабатывать в единицу времени. Это позволит оценить их применимость в рамках систем, предназначенных для работы в режиме реального времени.

2.3. Экспериментальные платформы

Для проведения измерений производительности нейросетевых компонентов SLAM-алгоритмов используются вычислительные платформы со встроенными ускорителями для запуска нейронных сетей. Эти устройства обладают малым весом и габаритами, что делает их подходящими для использования в автономных системах с ограничениями по массе и размеру. В данном разделе представлены распространенные одноплатные компьютеры и встраиваемые модули, включающие аппаратную поддержку для запуска нейронных сетей.

2.3.1. Nvidia

Компания Nvidia¹ выпустила пять поколений линейки устройств Jetson — одноплатных компьютеров с графическими процессорами, предназначенных для работы с нейронными сетями (см. таблицу 12). Наиболее часто в автономных системах используются модели Jetson Nano, Jetson Xavier NX, Jetson Orin Nano и Jetson Orin NX, так как они обладают малыми габаритами и массой по сравнению с моделями AGX, а также более высокой производительностью по сравнению с моделями серий TX.

Устройства поддерживают запуск моделей с использованием фреймворков TensorFlow, PyTorch, Caffe и других фреймворков, позволяющих использовать ядра CUDA.

В качестве операционной системы используются ОС серии JetPack, основанные на разных версиях Ubuntu (см. таблицу 1). Эта ОС предоставляет набор инструментов для взаимодействия со встроенным ускорителем.

¹Jetson — Embedded AI Computing Platform. URL: <https://developer.nvidia.com/embedded-computing>. Дата обращения: 14 декабря 2024 г.

Версия Jetpack	Версия Ubuntu
2	14.04
3	16.04
4	18.04
5	20.04
6	22.04

Таблица 1: Соответствие версий JetPack и Ubuntu

2.3.2. Rockchip

Компания Rockchip² является производителем «систем на кристалле» (SoC) для одноплатных компьютеров. В частности, модель Rockchip RK3588 имеет встроенный ускоритель нейронных сетей (NPU) и поддерживает запуск нейронных сетей с низким энергопотреблением. Эта модель используется в одноплатных компьютерах, таких как Orange Pi 5, Radxa ROCK 5B, Khadas Edge 2. Такие компьютеры обладают массой менее 50 грамм и потребляют менее 12 Вт энергии. Заявленная производительность NPU составляет 6 TOPS (триллионов целочисленных операций в секунду).

Для взаимодействия с ускорителем используется фреймворк RKNNLite, разработанный специально для SoC от компании Rockchip. RKNNLite позволяет конвертировать модели из форматов TensorFlow, ONNX, Caffe и Pytorch для запуска на ускорителе NPU, однако список поддерживаемых операций для каждого формата ограничен. В качестве операционной системы для этих платформ производитель предоставляет Ubuntu 22.04.

2.3.3. Hailo AI

Ускорители Hailo AI³ (см. таблицу 2) предоставляются в формате одноплатных компьютеров (серия Hailo-15) и в формате интегрируемых модулей M.2, PCIe и miniPCIe (серия Hailo-8). Модули Hailo включают

²Rockchip. URL: <https://www.rock-chips.com/a/en/>. Дата обращения: 14 декабря 2024 г.

³Hailo AI Accelerators. URL: <https://hailo.ai/products/ai-accelerators/>. Дата обращения: 14 декабря 2024 г.

в себя TPU (tensor processing unit), который обеспечивает производительность от 7 до 26 TOPS в зависимости от модели.

Для преобразования нейронных сетей из других фреймворков используется инструмент Hailo Dataflow Compiler, который позволяет адаптировать модели машинного обучения под архитектуру ускорителей.

Модуль	TOPS	Устройство
Hailo-15H	20	одноплатный компьютер
Hailo-15M	11	
Hailo-15L	7	
Hailo-8L	13	модуль M.2
Hailo-8	26	модули M.2, PCIe
Hailo-8R	13	модуль miniPCIe

Таблица 2: Модули Hailo AI

2.3.4. Intel

Intel Neural Compute Stick (NCS)⁴ (NCS) — это USB-устройство на базе VPU (vision processing unit), предназначенное для ускорения работы нейронных сетей на периферийных системах. Устройство базируется на процессоре Intel Movidius Myriad 2 VPU и поддерживает работу с популярными фреймворками, такими как TensorFlow и Caffe.

NCS предназначен для выполнения задач, связанных с распознаванием изображений, видеоаналитикой и другими приложениями, требующими обработки данных на периферии. Устройство обладает производительностью 3 TOPS и потребляет 2 Вт. Для работы с устройством используется SDK OpenVINO, в котором реализованы инструменты для конвертации и запуска моделей. На данный момент поддержка этого устройства прекращена.

Помимо NCS, компания Intel предлагает решения на базе FPGA, разработанные подразделением Intel Altera. FPGA-платформы, такие

⁴Intel® Neural Compute Stick 2. URL: <https://www.intel.com/content/www/us/en/products/sku/140109/intel-neural-compute-stick-2/specifications.html>. Дата обращения: 14 декабря 2024 г.

как Intel Stratix 10 и Intel Arria 10, предназначены для ускорения вычислений нейронных сетей и других высокопроизводительных задач. Для работы с FPGA серии Altera используется набор инструментов Intel Quartus Prime, позволяющий преобразовывать модели в формат, поддерживаемый ускорителем. Такие устройства доступны в виде подключаемых PCI-e модулей и одноплатных компьютеров.

2.3.5. Google Coral

Google Coral⁵ представляет собой серию устройств с интеграцией TPU (Tensor Processing Unit), специально разработанных для ускорения выполнения нейронных сетей. Все устройства оснащены идентичным ускорителем, но имеют различные форм-факторы и интерфейсы подключения: модули M.2, miniPCIe, PCIe и USB. Устройство поддерживает запуск моделей машинного обучения на основе фреймворка TensorFlow Lite. Ускоритель обладает заявленной производительностью 4 TOPS и потребляет 2 Вт, однако эта платформа не поддерживает многие операции.

2.4. Техники квантования нейронных сетей

Квантование моделей машинного обучения основано на преобразовании значений из типа данных `float32` в тип данных с меньшей битностью (`float16`, `int8`, `int4`). Из-за этого увеличивается скорость обмена данными между CPU и устройством, производящим вычисления. Кроме того, во многих устройствах целочисленные операции оптимизированы иначе, чем операции с плавающей точкой, например, за счет отдельных ядер для целочисленных вычислений (например, в CUDA) и за счет специализированных SIMD-операций. Наиболее распространенным является квантование в тип данных `int8`, и в дальнейшем будет рассматриваться именно он.

Два параметра, которые используются при квантовании — `scale` и `zero-point`. Эти параметры зависят от диапазона значений, который

⁵Coral. URL: <https://coral.ai/>. Дата обращения: 14 декабря 2024 г.

преобразуются в `int8` и определяют квантование следующим образом:

- `quantized_value = round(real_value / scale) + zero_point`
- `real_value = (quantized_value - zero_point) * scale`

Выбор подходящих значений этих параметров позволяет оптимально использовать диапазон значений типа данных `int8` и терять меньше точности.

Однако квантование невозможно применять для слоев с большой чувствительностью к точности операций, например, `LayerNorm` и `SoftMax`. Из-за этого при квантовании в модель могут добавляться промежуточные слои преобразования из `float32` в `int8` и наоборот.

Существует несколько распространенных способов квантования нейронных сетей: пост-тренировочное динамическое квантование (PTQ dynamic, Post-Training Dynamic Quantization), пост-тренировочное статическое квантование (PTQ static, Post-Training Static Quantization) и квантование с обучением (QAT, Quantization-Aware Training) [13, 24].

В рамках пост-тренировочного динамического квантования происходит преобразование весов модели из типа данных `float32` в тип данных `int8`. При этом активации (входные данные и промежуточные результаты слоев) преобразуются в тип данных `int8` динамически перед выполнением слоя и преобразуются обратно после выполнения. Так как каждый слой потенциально производит значения в разном диапазоне, индивидуальное преобразование после каждого слоя позволяет индивидуально выбирать подходящие значения `scale` и `zero-point`, что, в свою очередь, позволяет терять меньше точности, чем при использовании одних и тех же значений этих параметров для всех активаций. Пост-тренировочное динамическое квантование не реализовано для GPU в PyTorch и TensorRT и для NPU в RKNN-Toolkit.

Пост-тренировочное статическое квантование отличается от динамического квантования тем, что активации не преобразуются обратно в тип данных `float32`, если следующий слой позволяет использовать целочисленные значения для вычисления. Во время выполнения используется фиксированный набор значений `scale` и `zero-point`. Для опти-

мального выбора этих значений в рамках пост-тренировочного квантования проводится *калибровка* нейронной сети с помощью обучающего набора данных.

Квантование с обучением (QAT, Quantization-Aware Training) отличается от пост-тренировочного квантования тем, что не использует значения предобученной модели напрямую, а вместо этого производится обучение новой модели, где слои инициализируются и обучаются в типе данных `int8` (за исключением слоев с большой чувствительностью к точности). Это позволяет добиваться наилучшей точности среди этих методов, однако этот способ требует дополнительного обучения модели. Структура модели, квантованной с помощью QAT, идентична структуре модели, квантованной с помощью PTQ static, за исключением значений самих весов и значений параметров `scale` и `zero-point`.

Выводы

В силу доступности платформ в Лаборатории для измерения будут использоваться одноплатные компьютеры, перечисленные в таблице 3.

Устройство	Производитель	Тип ускорителя	TOPS / TFLOPS
Jetson Nano	Nvidia	GPU	0.472
Jetson Xavier AGX	Nvidia	GPU	32
Jetson Orin NX	Nvidia	GPU	100
Orange Pi 5	Orange Pi	NPU	5.6

Таблица 3: Сравнение экспериментальных платформ. TOPS — количество операций с целочисленными значениями в секунду, TFLOPS — количество операций с плавающей точкой в секунду.

В качестве тестируемых методов выделения дескрипторов были выбраны NetVLAD, CosPlace, EigenPlaces, AnyLoc, MixVPR, SALAD. Для выделения и ассоциации локальных особенностей были выбраны SuperPoint+SuperGlue, SuperPoint+LightGlue, XFeat+LighterGlue. Для определения производительности моделей машинного обучения измеряется средняя пропускная способность в изображениях в секунду.

Среди методов квантования для проведения экспериментального исследования было выбрано пост-тренировочное статическое квантование.

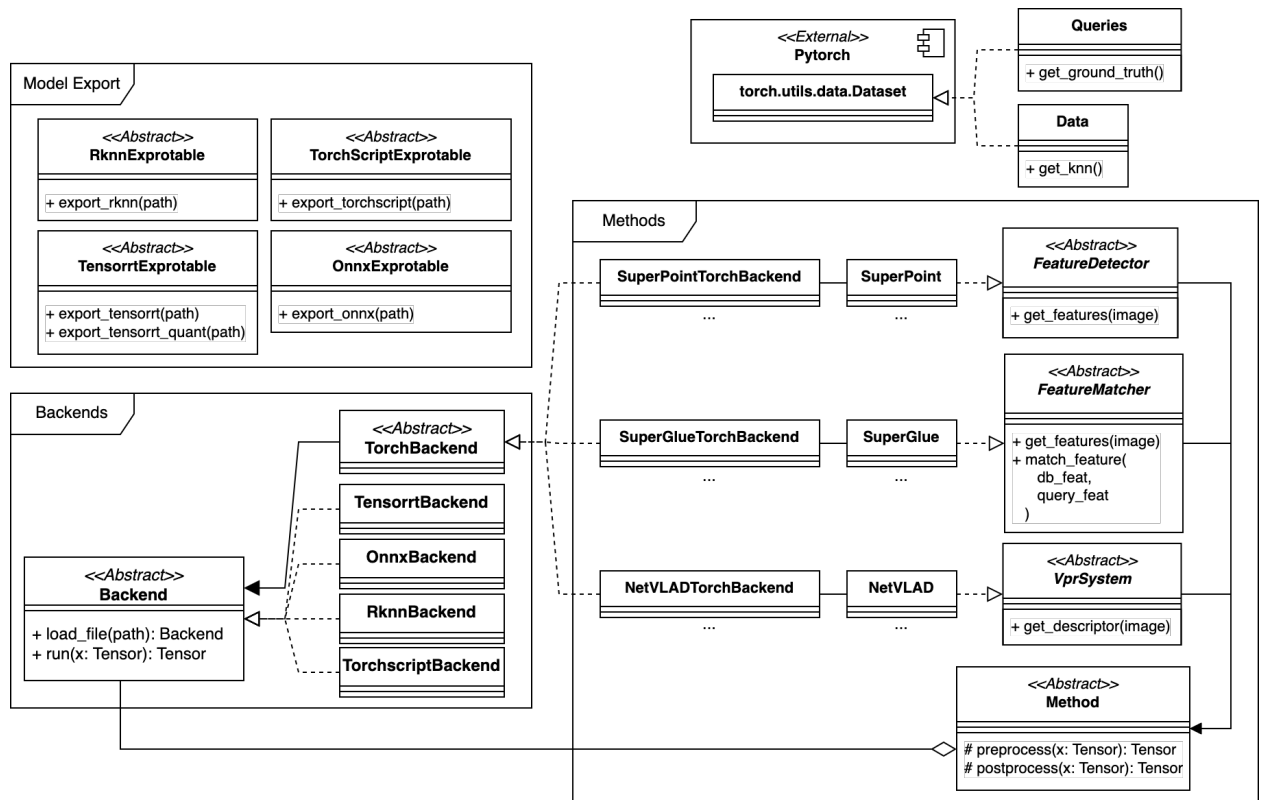


Рис. 2: Архитектура инструмента для измерения производительности алгоритмов.

3. Реализация инструмента для измерения производительности алгоритмов

В качестве основы для инструмента для произведения измерений был использован проект **aero-vloc** [16]. Данный проект представляет собой программное обеспечение для измерения метрики $Recall@k$ методов выделения глобальных дескрипторов и выделения и ассоциации локальных особенностей на наборах данных, состоящих из спутниковых снимков. Однако **aero-vloc** непригоден для работы с другими типами наборов данных, не позволяет измерять производительность методов, а также поддерживает запуск только на платформах с Nvidia GPU.

Для решения поставленной задачи была пересмотрена архитектура **aero-vloc**. На рисунке 2 представлена архитектура инструмента для измерения производительности алгоритмов.

3.1. Database и Queries

В `aero-vloc` для работы со спутниковыми снимками используется класс `Map`, однако он не подходит для работы с другими типами наборов данных. Для использования выбранных датасетов были реализованы классы `Database` и `Queries`. Эти классы наследуют абстрактный класс `torch.utils.data.Dataset`, который является стандартным примитивом при работе с наборами данных в рамках фреймворка PyTorch. Класс `Database` отвечает за чтение изображений базы данных и их координат. Класс `Queries` отвечает за чтение изображений запросов и их координат. Также этот класс реализует метод `get_ground_truth`, который возвращает индексы изображений из базы данных, соответствующих изображениям запросов.

3.2. Method, FeatureDetector, FeatureMatcher, VprSystem

В абстрактном классе `Method` объявлены методы `preprocess` и `postprocess`, соответствующие предобработке и постобработке, выполняющимся до и после запуска конкретной модели. Абстрактные классы `FeatureDetector`, `FeatureMatcher` и `VprSystem` соответствуют задачам выделения особенностей, ассоциации особенностей и выделения глобальных дескрипторов. В каждом из классов объявлены методы для выполнения этих задач: `get_features(image)` в `FeatureDetector`, `match_features(db_feat, query_feat)` в `FeatureMatcher`, `get_descriptor(image)` в `VprSystem`. Эти классы также реализуют методы `preprocess` и `postprocess` в соответствии с необходимым для каждой задачи способом предобработки и постобработки. Конкретные исследуемые методы реализуют один из этих трех абстрактных классов.

3.3. Backend

Выполнение исследуемых алгоритмов производится с помощью абстрактного класса `Backend` и реализующих его классов, соответствующих

щих этим бекендам. Кроме того, каждый выбранный метод в изначальном виде реализован на основе фреймворка PyTorch. Каждому методу соответствует свой класс `...TorchBackend`, реализующий абстрактный класс `TorchBackend`. Эти классы реализуют запуск методов в оригинальном виде.

3.4. Экспорт модели в другие форматы

Абстрактные классы `OnnxExportable`, `TorchScriptExportable`, `RknnExportable`, `TensorrtExportable` объявляют методы для экспорта модели в соответствующие форматы. Если конкретная модель поддерживает экспорт в данный формат, класс этой модели реализует соответствующий абстрактный класс `...Exportable`

3.5. `RknnExportable` и `TorchscriptExportable`

Класс `RknnExportable` отвечает за возможность каждой модели быть конвертированной в формат `rknn` для запуска на платформе OrangePi, этот класс наследуют подклассы `VprSystem` и `FeatureMatcher`, которые поддерживают конвертацию в этот формат. Для преобразования в формат `rknn` необходим экспорт модели в формат `torchscript`, который используется в наборе инструментов `rknn-toolkit2`. Класс `RknnExportable` наследует абстрактный класс `TorchscriptExportable`. Подклассы `TorchscriptExportable` реализуют метод `export_torchscript`, позволяя осуществлять преобразование в форматы `torchscript` и `rknn`.

3.6. Преобразование моделей

Для измерения пропускной способности модели с использованием платформы Orange Pi 5 необходимо преобразование модели в формат RKNN. В этом формате реализованы не все операции, поддерживаемые фреймворком PyTorch, с помощью которого реализованы исследуемые методы. Из-за этого необходимо разделение некоторых моделей

на часть, выполняемую на NPU-ускорителе, и на CPU.

Кроме того, использование квантованных моделей с Nvidia GPU возможно только с использованием фреймворка TensorRT, а также такой запуск доступен только с версиями CUDA не менее 12, из-за чего запуск квантованных моделей возможен только на Nvidia Jetson Orin NX и Nvidia Jetson Xavier AGX.

4. Эксперимент

В данном разделе будут описаны условия экспериментального исследования, его цель, исследовательские вопросы, а также результаты и выводы из них.

4.1. Цель экспериментального исследования

Целью экспериментального исследования является сравнение производительности алгоритмов на различных экспериментальных платформах и оценка их применимости в условиях реального времени с учетом ограничений на вычислительные ресурсы.

Рассматриваются следующие целевые значения пропускной способности алгоритмов:

- 10 изображений в секунду (достаточное, например, при наземном передвижении)
- 20 изображений в секунду (достаточное, например, при картографировании на большой высоте)
- 30 изображений в секунду (достаточное, например, при картографировании на малой высоте)

Для достижения поставленной цели были поставлены следующие исследовательские вопросы:

RQ1 : Какие методы выделения глобальных дескрипторов и выделения и ассоциации локальных особенностей наиболее производительные на рассмотренных платформах?

RQ2 : Как квантование в тип данных `int8` влияет на производительность алгоритмов?

4.2. Условия эксперимента

В контексте задачи выделения глобальных дескрипторов и задач выделения и ассоциации локальных особенностей используются наборы данных, которые делятся на две части: базу данных (database) и набор запросов (queries). База данных используется для составления индекса поиска по дескрипторам и составления списка локальных особенностей для каждого изображения базы, а набор запросов используется для оценки качества поиска.

В качестве набора данных использовался датасет St. Lucia, состоящий из городских изображений с исходным разрешением 640×480 пикселей. В рамках решения изображения были обрезаны до разрешения 480×480 и масштабированы до разрешений 800×800 , 600×600 , 400×400 , 300×300 , 200×200 . Для каждого измерения использовалось 30 запусков по 1000 изображений (или пар изображений для систем ассоциации дескрипторов). Дополнительный запуск на 30 изображениях использовался для прогрева устройства.

В качестве экспериментальных платформ были использованы устройства, представленные в таблице 4. Используемые операционные системы, версии CUDA (для платформ NVidia) и RKNNLite (для Orange Pi 5), а также версии библиотеки PyTorch представлены в таблице 5. Для измерения использовались исходные версии операционных систем без дополнительных изменений.

Устройство	Ускоритель	CPU	RAM
Jetson Nano	GPU	4x Cortex-A57	4 ГБ
Jetson Xavier AGX	GPU	8x Carmel v8.2	32 ГБ
Jetson Orin NX	GPU	8x Cortex-A78	16 ГБ
Orange Pi 5	NPU	4x Cortex-A76 + 4x Cortex-A55	16 ГБ

Таблица 4: Экспериментальные платформы.

Пропускная способность алгоритмов будет оцениваться с помощью измерения среднего количества кадров в секунду, с которым происходит обработка наборов данных.

Устройство	ОС	Версия ПО поддержки ускорителя	Версия Python	Версия PyTorch
Jetson Nano	JetPack 4.6	CUDA 10.2	3.6	1.11
Jetson Xavier AGX	JetPack 5.1	CUDA 11.8	3.8	2.0.0
Jetson Orin NX	JetPack 6.2	CUDA 12.6	3.10	2.5.0
Orange Pi 5	Ubuntu 22.04*	RKNNLite 2.3.0	3.10	2.5.0-cpu

Таблица 5: Используемое программное обеспечение.

*Использован образ, предоставляемый производителем: <http://www.orangepi.org/html/hardWare/computerAndMicrocontrollers/service-and-support/Orange-pi-5.html>.

4.3. Результаты

В данном разделе будут представлены результаты экспериментального исследования.

В таблицах 6, 7 отражена пропускная способность оригинальных и квантованных алгоритмов на платформе Jetson Orin NX 16, где **красным** цветом выделена пропускная способность более 30 кадров в секунду, **зеленым** — более 20, **синим** — более 10.

Исходя из полученных данных видно, что в большинстве случаев квантованные алгоритмы показывают бóльшую пропускную способность, чем оригинальные. Исключением является алгоритм MixVPR, который при наибольшем энергопотреблении работает более медленно в квантованном варианте (30.64 кадров в секунду), чем в оригинальном (40.61 кадров в секунду). Также алгоритм NetVLAD при наибольшем энергопотреблении для разрешений 200×200 и 300×300 показывает лучшие результаты в оригинальном виде, чем в квантованном (59.75 и 39.23 против 56.68 и 35.96 кадров в секунду соответственно).

Платформа Jetson Orin NX поддерживает пропускную способность в 30 кадров в секунду только при разрешении входных изображений до 300×300 пикселей. Для большинства алгоритмов платформа способна поддерживать пропускную способность для изображений до 600×600 пикселей.

В таблице 8 отражена пропускная способность алгоритмов на платформе Jetson Xavier AGX 32 при наибольшем энергопотреблении в 30 Вт. Результаты показывают, что только алгоритмы выделения глобальных дескрипторов CosPlace, EigenPlaces, MixVPR, NetVLAD и SALAD могут поддерживать пропускную способность в 10 кадров в секунду при разрешении не более чем 300×300 пикселей, и только CosPlace, EigenPlaces и MixVPR поддерживают пропускную способность в 20 кадров в секунду. Методы выделения и ассоциации локальных особенностей показывают пропускную способность менее 10 кадров в секунду.

В таблице 9 отражена пропускная способность алгоритмов на Jetson Nano. Платформа неспособна поддерживать пропускную способность в 10 кадров в секунду ни для одного алгоритма.

В таблицах 10, 11 отражена пропускная способность методов на платформе Orange Pi 5. Стоит отметить, что при преобразовании методов в подходящий для запуска на данной платформе формат квантование происходит в любом случае, поскольку данная платформа поддерживает только два типа данных: `float16` и `int8`. Методы CosPlace и EigenPlaces, а также MixVPR с использованием типа данных `int8` показывают наиболее высокую пропускную способность. Методы SALAD и SelaVPR для всех выбранных разрешений изображений показывают пропускную способность менее 10 изображений в секунду.

Для всех рассмотренных платформ и алгоритмов увеличение разрешения изображений приводит к уменьшению пропускной способности методов, а также увеличение энергопотребления приводит к росту пропускной способности. В большинстве случаев квантование моделей приводит к увеличению пропускной способности.

4.3.1. RQ1

Наиболее производительные методы выделения дескрипторов: NetVLAD, MixVPR, EigenPlaces, CosPlace. Наиболее производительный метод выделения локальных особенностей: XFeat. Наиболее производительные методы ассоциации локальных особенностей: LightGlue и LighterGlue, притом LightGlue более производителен при разрешениях

от 600×600 пикселей, а LighterGlue — при разрешениях до 600×600 пикселей.

4.3.2. RQ2

В большинстве случаев квантованные алгоритмы показывают более высокую пропускную способность по сравнению с оригинальными версиями. Исключением являются MixVPR и NetVLAD, при энергопотреблении 30 Вт и при разрешениях 200×200 и 300×300 пикселей, оригинальные версии этих алгоритмов работают быстрее.

4.4. Угрозы нарушения корректности

В рамках экспериментального исследования для запуска моделей на Nvidia GPU применялись библиотеки PyTorch (для оригинальных моделей) и TensorRT (для квантованных моделей). Сравнение разных библиотек, которые возможно использовать для запуска на Nvidia GPU, не производилось.

Метод	Энергопотребление, Вт	Разрешение изображений				
		200	300	400	600	800
CosPlace	15	16.71	12.75	9.16	5.41	3.38
	25	18.62	13.85	9.56	5.27	3.67
	30	28.34	25.93	22.58	11.97	7.68
EigenPlaces	15	18.97	14.77	10.23	5.89	3.64
	25	19.35	16.53	10.77	5.72	3.67
	30	25.23	25.82	22.60	11.95	7.60
MixVPR	15		22.00			
	25		24.82			
	30		40.61			
NetVLAD	15	26.26	17.03	9.29	5.47	2.67
	25	28.60	18.61	10.45	5.35	3.05
	30	59.75	39.23	22.66	11.35	6.50
SALAD	15	4.44	3.44	1.19	0.52	0.22
	25	7.71	4.16	2.16	0.68	0.29
	30	15.06	8.69	4.57	1.44	0.62
SelaVPR	15	1.47				
	25	1.83				
	30	4.08				
SuperPoint	15	15.22	14.15	9.48	5.75	3.36
	25	20.12	14.60	10.62	6.52	4.04
	30	20.61	17.83	13.51	9.70	4.22
XFeat	15	19.67	18.84	17.97	13.99	9.34
	25	20.78	20.00	18.95	14.39	12.25
	30	22.04	21.33	21.88	17.11	13.91
SuperGlue	15	9.55	9.46	9.57	4.28	1.20
	25	10.12	10.01	9.73	4.52	1.52
	30	13.16	13.11	12.90	9.58	8.76
LightGlue	15	14.55	12.43	10.13	9.21	6.91
	25	15.28	13.16	11.69	10.83	6.96
	30	26.21	18.42	16.94	15.04	12.76
LighterGlue	15	19.68	15.57	10.17	4.49	4.01
	25	20.26	16.94	11.60	4.85	4.34
	30	26.21	24.55	20.79	9.76	8.76

Таблица 6: Пропускная способность алгоритмов на платформе Jetson Orin NX 16. **Красным** цветом выделена пропускная способность более 30 кадров в секунду, **зеленым** — более 20, **синим** — более 10. Стандартное отклонение составило менее 5% от среднего для всех приведенных значений.

Метод	Энергопотребление, Вт	Разрешение изображений				
		200	300	400	600	800
CosPlace	15	29.36	23.30	20.83	11.94	7.10
	25	37.79	25.27	22.50	13.06	8.81
	30	39.83	31.80	21.99	17.89	11.60
EigenPlaces	15	29.07	22.87	20.95	12.45	7.14
	25	37.24	24.85	22.24	13.22	8.49
	30	38.89	30.81	22.33	17.74	12.85
MixVPR	15		23.85			
	25		29.74			
	30		30.64			
NetVLAD	15	40.29	26.96	20.98		
	25	52.25	35.29	26.00		
	30	56.68	35.96	24.96		
SALAD	15	23.21	13.34	6.65		
	25	23.50	15.08	7.91		
	30	27.83	21.54	14.35		
SelaVPR	15	9.63				
	25	10.73				
	30	19.18				

Таблица 7: Пропускная способность квантованных версий алгоритмов на платформе Jetson Orin NX 16. **Красным** цветом выделена пропускная способность более 30 кадров в секунду, **зеленым** — более 20, **синим** — более 10. Стандартное отклонение составило менее 5% от среднего для всех приведенных значений.

Метод	Разрешение изображений				
	200	300	400	600	800
CosPlace	20.34	11.96	8.29	4.22	2.51
EigenPlaces	20.41	11.97	8.33	4.22	2.51
MixVPR		21.27			
NetVLAD	19.96	11.65	7.11	3.75	2.10
SALAD	10.81	4.01	1.68	0.39	0.20
SelaVPR	2.10				
SuperGlue	4.34	4.23	3.78	2.15	1.92
XFeat	4.45	4.25	3.80	3.01	1.88
SuperPoint	3.53	1.94	1.06	0.40	0.15
LightGlue	6.57	6.01	5.93	5.10	4.53
LighterGlue	8.46	7.92	7.38	3.31	3.11

Таблица 8: Пропускная способность алгоритмов на платформе Jetson Xavier AGX при максимальном допустимом энергопотреблении 30 Вт. Зеленым цветом выделена пропускная способность более 20 кадров в секунду, синим — более 10. Стандартное отклонение составило менее 5% от среднего для всех приведенных значений.

Метод	Разрешение изображений				
	200	300	400	600	800
CosPlace	5.46	3.03	2.06	1.14	0.69
EigenPlaces	5.56	3.03	2.06	1.14	0.69
MixVPR		6.19			
NetVLAD	5.35	3.09	1.91	0.95	0.55
SALAD	1.91	1.51	0.83	0.28	
SuperPoint	5.80	3.26	1.94	0.86	0.49
XFeat	7.41	7.07	6.75	4.56	2.73
SuperGlue	3.64	3.43	2.34	0.56	
LightGlue	7.72	5.97	4.32	1.30	
LighterGlue	6.99	3.17	1.44	0.35	0.31

Таблица 9: Пропускная способность алгоритмов на платформе Jetson Nano. Стандартное отклонение составило менее 5% от среднего для всех приведенных значений.

Метод	Разрешение изображений				
	200	300	400	600	800
CosPlace	33.81	22.16	14.12	7.41	
EigenPlaces	34.63	21.86	13.31	7.43	
MixVPR		29.33			
NetVLAD	13.29	9.22	5.72		
SALAD	7.48	3.37	1.64		
SelaVPR	3.19				

Таблица 10: Пропускная способность алгоритмов в типе данных `float16` на платформе Orange Pi 5. **Красным** цветом выделена пропускная способность более 30 кадров в секунду, **зеленым** — более 20, **синим** — более 10. Стандартное отклонение составило менее 5% от среднего для всех приведенных значений.

Метод	Разрешение изображений				
	200	300	400	600	800
CosPlace	50.89	30.70	21.23	11.77	
EigenPlaces	46.81	32.22	21.43	11.58	
MixVPR		44.73			
NetVLAD	25.15	17.22	9.10		
SALAD	9.49	4.20	1.76		
SelaVPR	3.26				

Таблица 11: Пропускная способность квантованных алгоритмов в типе данных `int8` на платформе Orange Pi 5. **Красным** цветом выделена пропускная способность более 30 кадров в секунду, **зеленым** — более 20, **синим** — более 10. Стандартное отклонение составило менее 5% от среднего для всех приведенных значений.

Заключение

В рамках данной работы были выполнены следующие задачи.

1. Выполнен обзор методов ассоциации особенностей и выделения дескрипторов, основанных на нейронных сетях.
2. Выполнен обзор распространенных встраиваемых платформ с ускорителями для запуска нейронных сетей.
3. Выполнен обзор техник квантования нейронных сетей.
4. Реализован инструмент для измерения производительности алгоритмов.
5. Произведены измерения производительности алгоритмов, а также их сжатых версий на выбранных платформах.

Реализация инструмента доступна в GitHub репозитории: <https://github.com/true-real-michael/nn-slam-benchmark>.

Список литературы

- [1] Ali-Bey Amar, Chaib-Draa Brahim, Giguere Philippe. Mixvpr: Feature mixing for visual place recognition // Proceedings of the IEEE/CVF winter conference on applications of computer vision. — 2023. — P. 2998–3007.
- [2] Anyloc: Towards universal visual place recognition / Nikhil Keetha, Avneesh Mishra, Jay Karhade et al. // IEEE Robotics and Automation Letters. — 2023.
- [3] Arandjelovic Relja, Zisserman Andrew. All about VLAD // Proceedings of the IEEE conference on Computer Vision and Pattern Recognition. — 2013. — P. 1578–1585.
- [4] Barbarani Giovanni, Mostafa Mohamad, Bayramov Hajali et al. Are Local Features All You Need for Cross-Domain Visual Place Recognition? — 2023. — 2304.05887.
- [5] Berton Gabriele, Masone Carlo, Caputo Barbara. Rethinking visual geo-localization for large-scale applications. — 2022.
- [6] Cao Bingyi, Araujo Andre, Sim Jack. Unifying deep local and global features for image search // Computer Vision—ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XX 16 / Springer. — 2020. — P. 726–743.
- [7] DeTone Daniel, Malisiewicz Tomasz, Rabinovich Andrew. SuperPoint: Self-Supervised Interest Point Detection and Description // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops. — 2018. — June.
- [8] Deep Residual Learning for Image Recognition / Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun // CoRR. — 2015. — Vol. abs/1512.03385. — arXiv : 1512.03385.

- [9] Dinov2: Learning robust visual features without supervision / Maxime Oquab, Timothée Darcet, Théo Moutakanni et al. // arXiv preprint arXiv:2304.07193. — 2023.
- [10] Tzachor Issar, Lerner Boaz, Levy Matan et al. EfficVPR: Effective Foundation Model Utilization for Visual Place Recognition. — 2024. — 2405.18065.
- [11] Eigenplaces: Training viewpoint robust models for visual place recognition / Gabriele Berton, Gabriele Trivigno, Barbara Caputo, Carlo Masone // Proceedings of the IEEE/CVF International Conference on Computer Vision. — 2023. — P. 11080–11090.
- [12] Emerging properties in self-supervised vision transformers / Mathilde Caron, Hugo Touvron, Ishan Misra et al. // Proceedings of the IEEE/CVF international conference on computer vision. — 2021. — P. 9650–9660.
- [13] Guo Yunhui. A Survey on Methods and Theories of Quantized Neural Networks. — 2018. — 1808.04752.
- [14] Lindenberger Philipp, Sarlin Paul-Edouard, Pollefeys Marc. Lightglue: Local feature matching at light speed // Proceedings of the IEEE/CVF International Conference on Computer Vision. — 2023. — P. 17627–17638.
- [15] Lowe D.G. Object recognition from local scale-invariant features. — 1999.
- [16] Moskalenko Ivan, Kornilova Anastasiia, Ferrer Gonzalo. Visual place recognition for aerial imagery: A survey // Robotics and Autonomous Systems. — 2025. — Vol. 183. — P. 104837.
- [17] NetVLAD: CNN architecture for weakly supervised place recognition / Relja Arandjelovic, Petr Gronat, Akihiko Torii et al. // Proceedings of the IEEE conference on computer vision and pattern recognition. — 2016. — P. 5297–5307.

- [18] ORB: An efficient alternative to SIFT or SURF / Ethan Rublee, Vincent Rabaud, Kurt Konolige, Gary Bradski // 2011 International conference on computer vision / Ieee. — 2011. — P. 2564–2571.
- [19] Patch-netvlad: Multi-scale fusion of locally-global descriptors for place recognition / Stephen Hausler, Sourav Garg, Ming Xu et al. // Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. — 2021. — P. 14141–14152.
- [20] Speeded-Up Robust Features (SURF) / Herbert Bay, Andreas Ess, Tinne Tuytelaars, Luc Van Gool // [Computer Vision and Image Understanding](#). — 2008. — Vol. 110, no. 3. — P. 346–359. — Similarity Matching in Computer Vision and Multimedia. URL: <https://www.sciencedirect.com/science/article/pii/S1077314207001555> (дата обращения: 20.05.2025).
- [21] Superglue: Learning feature matching with graph neural networks / Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, Andrew Rabinovich // Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. — 2020. — P. 4938–4947.
- [22] Lu Feng, Zhang Lijun, Lan Xiangyuan et al. Towards Seamless Adaptation of Pre-trained Models for Visual Place Recognition. — 2024. — 2402.14505.
- [23] Transvpr: Transformer-based place recognition with multi-level attention aggregation / Ruotong Wang, Yanqing Shen, Weiliang Zuo et al. // Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. — 2022. — P. 13648–13657.
- [24] Nagel Markus, Fournarakis Marios, Amjad Rana Ali et al. A White Paper on Neural Network Quantization. — 2021. — 2106.08295.
- [25] XFeat: Accelerated Features for Lightweight Image Matching / Guilherme Potje, Felipe Cadar, André Araujo et al. // Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. — 2024. — P. 2682–2691.

Поколение	Устройство	Архите- ктура GPU	Ядер CUDA	TOPS	TFLOPS	RAM	Габариты, мм Масса, г	Энерго- потре- бление	
TX1 (2015)	TX1	Maxwell	256		1.3	4	170 x 170 x 15		
TX2 (2017)	TX2	Pascal	256		1.33	4	400	15	
	TX2 NX					8	100 x 80 x 29 140		
Xavier (2018)	AGX 32 GB	Volta	512	32		32	105 x 105 x 65	30	
	AGX 64 GB					64	670		
	NX 8 GB		384	21		8	20		
	NX 16 GB					16			
Nano (2019)	Nano	Maxwell	128		0.47	4			
Orin (2022)	Nano 4 GB	Ampere	512	20		4	100 x 80 x 29	10	
	Nano 8 GB			40		8	200	15	
	NX 8 GB		1024	70		8		20	
	NX 16 GB					16		25	
	AGX 32 GB			1792	200		32	110 x 110 x 71.7	40
	AGX 64 GB			2048	275		64	870	60

Таблица 12: Заявленные характеристики встраиваемых платформ Nvidia Jetson.