Санкт-Петербургский государственный университет

Кафедра системного программирования

Смирнов Кирилл Вадимович

Поиск визуально схожих изображений в неорганизованных коллекциях

Дипломная работа

Научный руководитель:
доцент Пименов А. А.

Рецензент:
к.т.н. Федоренко С. И.

Санкт-Петербург
2019

SAINT-PETERSBURG STATE UNIVERSITY

Software Engineering

Kirill Smirnov

Visual similarity metrics for loop closure detection and image retrieval

Graduation Thesis

Scientific supervisor:
Associate Professor Pimenov Alexander

Reviewer:
PhD Fedorenko S. I.

Saint-Petersburg
2019

Оглавление

Введение **4**

1. Постановка задачи **7**

2. Обзор предметной области **8**

2.1. Ключевые точки и дескрипторы 8

2.1.1. Классические дескрипторы 9

2.2. BoW-словарь 11

2.3. BoW-вектор 12

3. Алгоритм построения BoW-словаря **13**

3.1. Распараллеливание кластеризации по вершинам дерева BoW-словаря 15

3.2. Многопоточная кластеризация дескрипторов ассоциированных с вершинами дерева BoW-словаря 16

3.3. GPGPU версия алгоритма построения BoW-словаря . . . 16

3.4. Оценка эффективности алгоритмов построения BoW-словаря 18

4. Алгоритм сопоставления BoW-векторов **19**

5. Сравнение различных дескрипторов **20**

Заключение **21**

Список литературы **22**

Введение

Компьютерное зрение – дисциплина, главной задачей которой является понимание содержимого изображений. Компьютерное зрение получило широкое распространение в различных областях человеческой жизни. Например, алгоритмы компьютерного зрения используются для контроля качества продукции на предприятиях [14] или в медицинских программных продуктах, помогающих врачам в анализе медицинских снимков (в частности, определения раковых опухолей [2] или создания трехмерных моделей лица для пластических операций).

Компьютерное зрение представляет огромный интерес для областей, опирающихся на работу с изображениями. Например, для такой области как робототехника, направленной на создание и изучение различных классов роботов, развитие алгоритмов компьютерного зрения для специфичных роботов является критическим при решении сопутствующих задач. Широкое распространение получила задача локализации роботов [9], то есть определение положения робота в пространстве относительно окружающих объектов. Данная задача имеет особенную актуальность для самоуправляемых автомобилей и домашних роботов (например, роботы-пылесосы и другие роботы-помощники).

Некоторые из таких роботов полагаются на набор маркеров[3] для локализации. Маркеры – это визуальные образы, местоположение которых известно заранее. Данные решения обладают высокой точностью, но требуют больших инвестиций для организацию инфраструктуры (размещение маркеров, создание и поддержание актуальной базы данных с местоположением маркеров).

Альтернативным подходом являются алгоритмы одновременной локализации и построения карт [15]. Данный подход заключается в одновременном построении и обновлении карты неизвестного окружения, а также локализации наблюдателя в данном окружении. Такие алгоритмы имеют вероятностную природу – в каждый момент дискретного времени поддерживаются несколько гипотез о местоположении робота и вероятности нахождения робота в каждой из гипотез. За теку-

щую позицию робота принимается позиция, обладающая наибольшей вероятностью. Однако, данная группа алгоритмов зачастую неспособна определить возвращение робота в ранее посещенную позицию. Другими словами, такие алгоритмы плохо справляются с задачей замыкания цикла. Последствием такого поведения является то, что алгоритм продолжает поддерживать гипотезы, существование которых невозможно из-за замыкания циклов. Данный недостаток критически сказывается на точности локализации и построения карты. Для решения этой проблемы может быть сформулирована задача устойчивого определения замыкания циклов.

Для решения задачи нахождения замыкания циклов делается предположение о визуальной схожести одной и той же локации в разное время. Поэтому задача нахождения замыкания циклов является задачей распознавания текущей визуальной сцены среди ранее посещенных. Таким образом, данную задачу можно переформулировать в задачу поиска визуально схожих изображений. Для ее решения существуют несколько подходов.

Одним из самых популярных является подход, основанный на сопоставлении множества дескрипторов изображений. Дескриптором изображения называют численный вектор, описывающий отличимый участок изображения – окрестность ключевой точки. Примерами таких участков являются углы [6], края объектов [10] или блобы [13] – участки изображения, в рамках которых такие свойства, как яркость или цвет меняются незначительно. Размерности векторов дескрипторов отличаются в зависимости от используемого алгоритма для выделения и описания отличимых регионов изображения. Традиционным подходом для эффективной работы с многомерным пространством дескрипторов является использование мешка визуальных слов (BoW-словаря). Мешок визуальных слов – это множество заранее определенных дескрипторов, каждый из который принято называть визуальным словом. Для сравнения двух изображений каждое из них представляется в виде вектора визуальных слов (BoW-вектора). Для нахождения BoW-вектора изображения для каждого дескриптора этого изображения находится

ближайшее визуальное слово из BoW-словаря с помощью заранее выбранной метрики. Впоследствии BoW-вектра двух изображений сравниваются и определяется степень схожести этих изображений.

Алгоритмы построения BoW-словарей и BoW-векторов обладают высокой вычислительной сложностью. Возможное решение данной проблемы заключается в использовании шаблонов многопоточного программирования с последующим исполнением данных алгоритмов на современных CPU (центральный процессор) или на GPGPU (видеокарта). GPGPU обладают большим количеством доступных вычислительных ядер чем CPU, что позволяет обеспечить более эффективную параллелизацию алгоритмов на аппаратном уровне. В связи с этими особенностями, становится актуальна задача не только распараллеливания представленных алгоритмов, но и их последующая оптимизация для исполнения на GPGPU.

1. Постановка задачи

Целью данной работы является исследование применимости различных дескрипторов для задач поиска визуально схожих изображений. Для ее достижения были поставлены следующие задачи:

- провести исследование предметной области;
- разработать многопоточный алгоритм построения BoW-словаря для дескрипторов произвольной размерности и оптимизировать его для GPGPU в целях применения к дескрипторам большой размерности;
- разработать алгоритм построения BoW-вектора для дескрипторов изображений и поиска ближайших изображений;
- оценить производительность различных дескрипторов ключевых точек, найденных на изображении, совместно с разработанными алгоритмам.

2. Обзор предметной области

2.1. Ключевые точки и дескрипторы

Ключевой точкой называется точка, область изображения (область интереса) вокруг которой является отличительной для данного изображения. При работе с ключевыми точками основную область интереса представляют в виде численного вектора – дескриптора ключевой точки. Для оценки качества дескриптора существует ряд характеристик.

- Устойчивость к шумам. Из-за несовершенства аппаратуры и изменчивости окружающий среды любое изображение содержат некоторое количество шума – случайное изменение значения сенсоров, которое не соответствует реальным наблюдениям. Для наблюдателя шум проявляется в виде случайно появляющихся разноцветных или черно-белых писклей.

- Устойчивость к геометрическим трансформациям. При изменении точки наблюдения объекты на изображениях могут менять размеры и ориентацию в пространстве.

- Изменчивость освещенности. При попытке заснять одну и ту же сцену в разных условиях освещенности значения соответствующих писклей будет отличаться.

- Быть эффективным в контексте вычислительной сложности. Перед многими задачами компьютерного зрения стоит задача обработки определенного количества изображений за единицу времени. Например, из-за изменчивости ситуации на дорогах общего пользования автопилоту необходима актуальная информация об окружении, поэтому в задаче локализации самоуправляемых автомобилей данное требование является критическим. Как следствие, алгоритмы построения дескрипторов должны иметь определенные временные ограничения.

- Быть эффективным в контексте занимаемой памяти. При работе с большим количеством изображений и, как следствие, большим количеством дескрипторов, размер, который дескрипторы занимают в памяти, становится важной характеристикой. Особую важность данная характеристика приобретает в задачах, где есть ограничения на используемое аппаратное обеспечение. Примерами таких задач являются задачи, сопряженные с квадрокоптерами. Установка на робота производительного компьютера практически невозможна из-за конструктивных особенностей квадрокоптера. Отсюда появляется задача уменьшения размеров дескриптоов.

Ключевой набор характеристик для получаемого дескриптора может меняться в зависимости от конкретной задачи. Например, при заранее известном окружении и положении объектов можно пренебречь устойчивостью к геометрическими трансформациями и изменениям яркости. Такая ситуация распространена на предприятиях, где контроль качества продукции осуществляется с помощью алгоритмов компьютерного зрения.

2.1.1. Классические дескрипторы

До популяризации методов машинного обучения основным способом создания нового вида дескриптора был анализ исследователем особенностей изображений. Получаемые алгоритмы извлекали дескрипторы на основе цвета, формы или текстуры области интересов. Наиболее популярными дескрипторами, которые основываются на информации об интенсивности пикселей являются Binary Robust Independent Elementary Features (BREIF), Oriented FAST and Rotated BRIEF (ORB), KAZE [1] и AKAZE.

BRIEF – алгоритм построения дескриптора, производящий серию сравнений освещенности пикселей внутри области интересов. Результаты сравнений интерпретируются как значение битов в бинарном векторе дескриптора. Определим функцию сравнения $\tau(\mathbf{p}; \mathbf{x}, \mathbf{y})$ пикселей $\mathbf{x}$ и

$\mathbf{y}$ из множества пикселей области интересов как:

$$\tau(\mathbf{p};\mathbf{x},\mathbf{y}) := \begin{cases} 1 & if\,\mathbf{p}(\mathbf{x}) < \mathbf{p}(\mathbf{y}) \\ 0 & otherwise \end{cases},$$

где $\mathbf{p}(\mathbf{x})$ и $\mathbf{p}(\mathbf{y})$ – значения интенсивности пикселей $\mathbf{x}$ и $\mathbf{y}$ соответственно. Тогда вектор дескриптора $f_{n_d}(\mathbf{p})$ может быть вычислен по формуле:

$$f_{n_d}(\mathbf{p}) := \sum_{1 \le i \le n_d} 2^{i-1} \tau\left(\mathbf{p};\mathbf{x}_i,\mathbf{y}_i\right),$$

Несмотря на то, что дескриптор BRIEF может быть вычислен быстро и не требует большого размера занимаемой памяти, он является крайне неустойчивым к вращениям, что сильно ограничивает возможность его применения;

ORB – дескриптор, продолжающий идею BRIEF дескриптора и решающий проблему устойчивости к вращениям. В ORB определяется вектор направления дескриптора – вектор от центра области интереса до ключевой точки соответствующей данной области интереса. После чего производится поворот области интереса на угол вектора направления. Данный подход позволяет увеличить устойчивость дескриптора к повороту.

KAZE – данный подход пытается решить задачу устойчивости получаемого дескриптора особой точки изображения к изменению размера. Для ее достижения строится пирамида изображений, на каждом уровне которой изображение, полученное по результатам применения свертки к исходному. После получения пирамидальной структуры на каждом уровне выбираются особые точки и определяется их ориентация за счет градиентов пикселей. Последнем этапом является описание полученной области в виде численного вектора.

AKAZE – является попыткой улучшить временные характеристики KAZE за счет использования более эффективного алгоритма построения пирамиды изображений.

2.2. BoW-словарь

Концепция BoW (Bag of Words, мешка слов) впервые появилась в задачах обработки естественного языка. BoW – это множество, содержащее в себе слова текста. Проводя статистический анализ данного множества, возможно предоставлять гипотезы о его содержании. При работе с изображениями BoW содержит дескрипторы, полученные при обработке изображений.

Дескрипторы являются векторами в многомерном пространстве. Для сравнения набора дескрипторов с двух изображений необходимо решать задачу сравнения двух векторов. Для эффективного в смысле времени исполнения решения данной задачи мы введем понятия BoW-словаря. В отличии BoW, который содержал в себе все возможные дескрипторы, BoW-словарь является множеством, содержащим заранее выбранные дескрипторы – визуальные слова.

При решении задачи сравнения двух изображений, для каждого изображения вычисляются дескрипторы, после чего для каждого дескриптора в BoW-словаре ищется ближайшее визуальное слово по заданной метрике. В результате каждое изображение представляется как набор визуальных слов. Дальнейшее сравнение визуальных слов является менее ресурсъёмкой задачей, чем сравнение дескрипторов.

Нужно понимать, что с увлечением размера BoW-словаря повышается точность сопоставления визуальных слов дескрипторам изображения, но ухудшаются временные характеристики решения. Таким образом, выбор размера BoW-словаря решается разработчиком в зависимости от специфики задачи. Тем не менее возможно повысить временные характеристики решения при сохранении размеров BoW-словаря за счет построения иерархической структуры BoW-словаря.

Иерархическая структура BoW-словаря получается посредством кластеризации пространства визуальных слов. Процесс кластеризации повторяется многократно. В результате получается древовидная структура, листьями которой является визуальные слова исходного BoW-словаря.

2.3. BoW-вектор

Готовый BoW-словарь используется для представления изображений в виде BoW-вектора. Множество дескрипторов ключевых точек, полученных с изображения, сопоставляются с визуальными словами из BoW-словаря.С этой целью, начиная с корня дерева BoW-словаря, среди дескрипторов, соответствующих сыновьям текущей вершины, производится поиск ближайшего к дескриптору ключевой точки изображения. Данный процесс продолжается до достижения листьев дерева BoW-словаря, дескрипторы которых соответствуют визуальным словам BoW-словаря. Множество визуальных слов, полученных после сопоставления дескрипторов ключевых точек изображения, называется BoW-вектор.

3. Алгоритм построения BoW-словаря

Увеличение количества дескрипторов, используемых для построения BoW-словаря, позволяет более точно аппроксимировать пространство дескрипторов и, как следствие, повышать точность алгоритмов сопоставления множеств дескрипторов. Однако, использование дескрипторов в явном виде является крайне ресурсоёмкой задачей, поэтому при построении BoW-словаря пространство дескрипторов кластеризуется. Для большей эффективности кластеризация повторяется несколько раз для получения иерархической структуры. Архитектуру данного алгоритма можно изобразить в виде схемы (Рис. 1)

Наиболее ресурсоёмким этапом является иерархическая кластеризация. Однако, данный этап обладает целым рядом возможностей для распараллеливания.

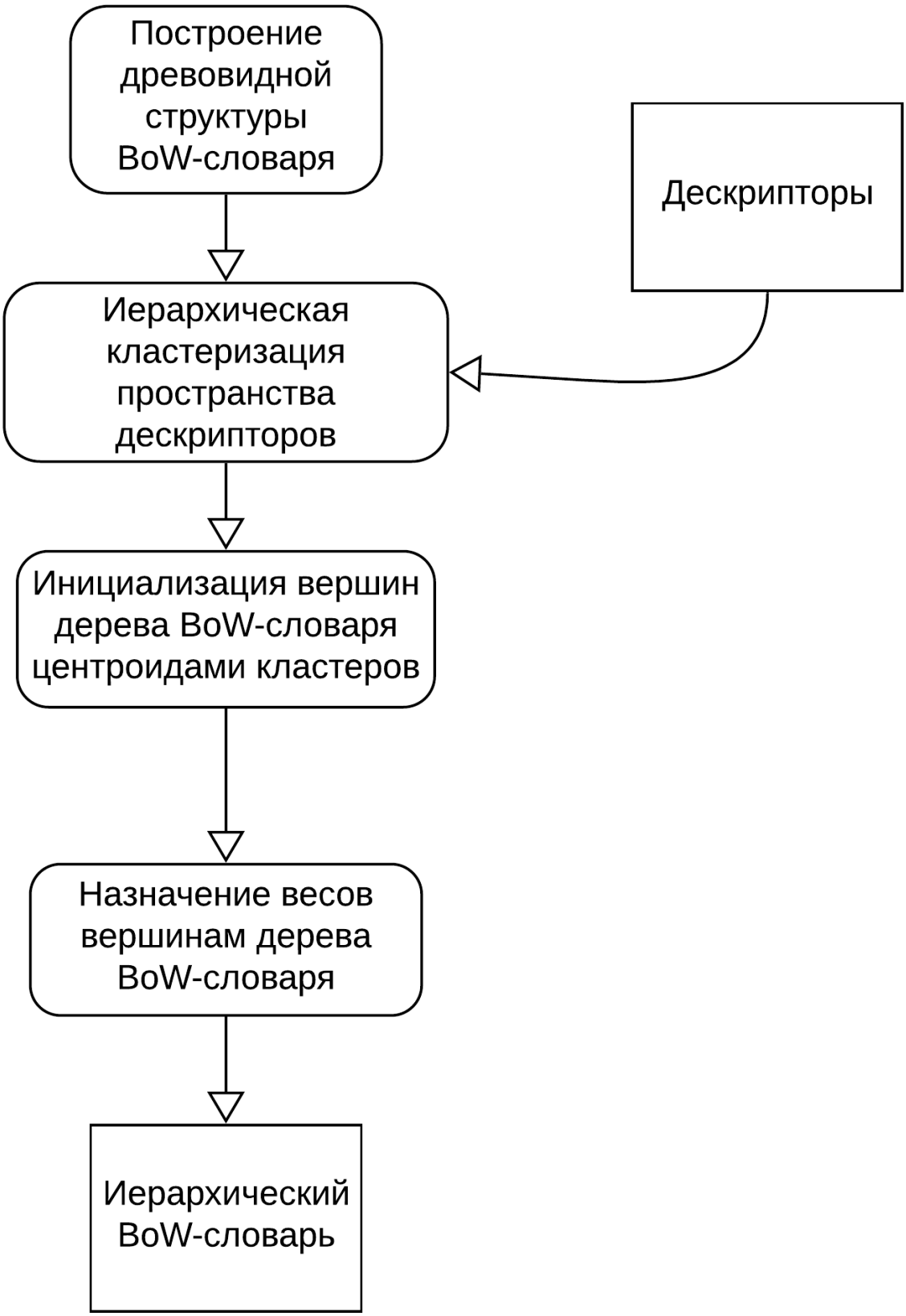

Рис. 1: Схема алгоритма построения BoW-словаря

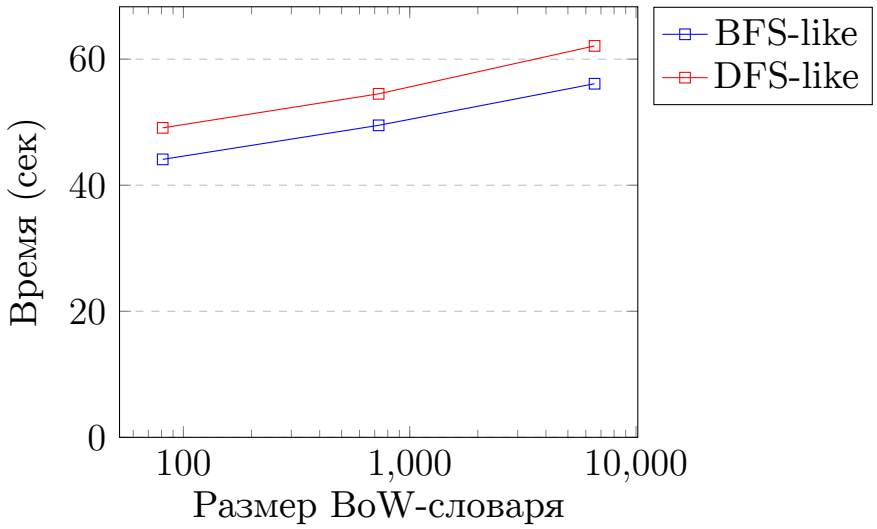

Рис. 2: Сравнение эффективности DFS-like и BFS-like подходов

3.1. Распараллеливание кластеризации по вершинам дерева BoW-словаря

За счет непересекаемости множества дескрипторов для вершин одного уровня дерева BoW-словаря возможно параллельные вычисления на уровне вершин. Для этого были разработаны два алгоритма обхода дерева BoW-словаря:

- BFS-like – подход на основе поиска в ширину.
- DFS-like – подход на основе поиска в глубину.

С целью выявления лучшего алгоритма были разработаны их прототипы. Реализации представляют код на языке C++ с использованием библиотеки Intel Threading Building Blocks [7]. Измерения эффективности производились на компьютере со следующими характеристиками: два процессора Intel Xeon E5-2670 v3 2.30GH, 252 гигабайт оперативной памяти. Для построения BoW-словаря использовалось 6 000 000 ORB дескрипторов, полученных с изображений открытого набора данных KITTI[8].

На основании поставленных экспериментов (Рис. 2) было принято решение об использование BFS-like алгоритма как основного.

3.2. Многопоточная кластеризация дескрипторов ассоциированных с вершинами дерева BoW-словаря

Для кластеризации множества дескрипторов, ассоциированных с вершиной, используется следующий алгоритм:

CLUSTERIZATION($Descriptors, Clusters$)

1 **while** $last\_association \neq current\_association$
2 $last\_association = current\_association$
3 **for** each descriptor $d \in \mathcal{D}escriptors$
4 $current\_association[d] = closest\_cluster(d, clusters)$
5 **for** each cluster $c \in \mathcal{C}lusters$
6 $c = update\_clusters(\mathcal{D}escriptors, clusters)$

Можно выделить следующие особенности данного алгоритма:

- поиск ближайшего центроида является независимым для каждого дескриптора;
- обновление значения центроида происходит независимо для каждого кластера.

Данные наблюдения позволяют реализовать многопоточную версию алгоритма. В первой версии многопоточного алгоритма под CPU был использован паттерн parallel for для параллельного поиска ближайшего центроида и обновления центроидов кластеров. Однако, с целью дальнейшего улучшения временны характеристик алгоритма был реализован подход с использованием паттерна parallel reduce.

По результатам экспериментов (Рис. 3) было принято решение об использовании подхода с parallel reduce.

3.3. GPGPU версия алгоритма построения BoW-словаря

С расчетом на дальнейшее улучшение временных характеристик данного подхода было принято решение реализации алгоритма построения BoW-словаря на GPGPU с использованием Compute Unified Device

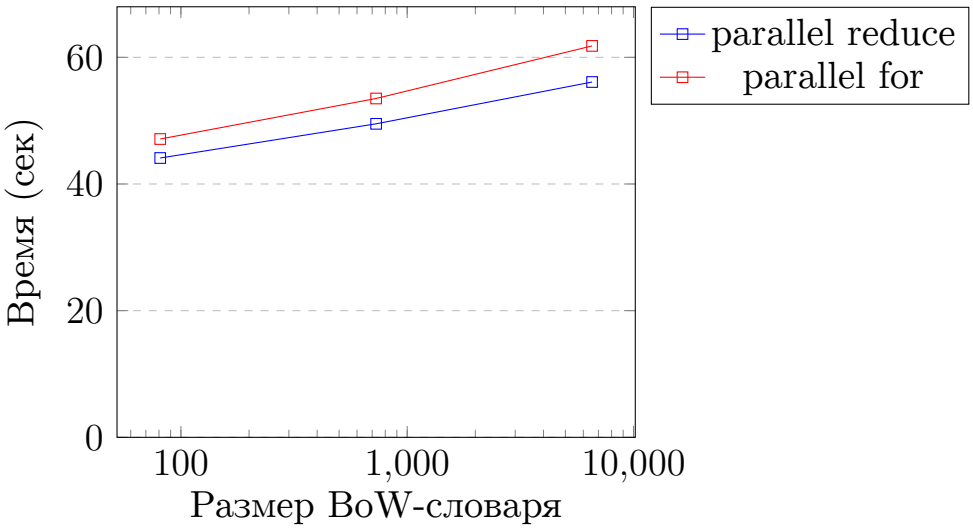

Рис. 3: Сравнение эффективности parallel reduce и parallel for

Architecture (CUDA) [4]. Реализация подхода с обходом в ширину и паттерном parallel reduce показала крайне неудовлетворительные результаты. Данное решение по временным показателям многократно уступало существующему решению на CPU. Анализ показал, что значительную часть времени исполнения занимают синхронизации при реализации паттерна parallel reduce.

Из-за неудовлетворительных характеристик первой версии алгоритма на GPGPU было принято решение о реализации альтернативной версии. Главной особенностью нового решения было избавления от паттерна parallel reduce. Для этого было реализовано два GPGPU-kernel'а. Функциональность первого заключается в поиске ближайшего центроида для каждого дескриптора, функциональность второго заключается в обновлении центроидов кластеров. Как было сказано ранее, этап поиска ближайшего центроида является независимым для каждого дескриптора и этап обновления центроидов является независимым для каждого кластера. Таким образом реализованные GPGPU-kernel'ы не требуют внутренней синхронизации. Синхронизация между GPGPU-kernel'ами реализована по средствам синхронизации через host.

Несмотря на значительно лучшие характеристики второй версии GPGPU алгоритма в сравнении с первой версий, она все равно уступает версии на CPU (Рис. 4).

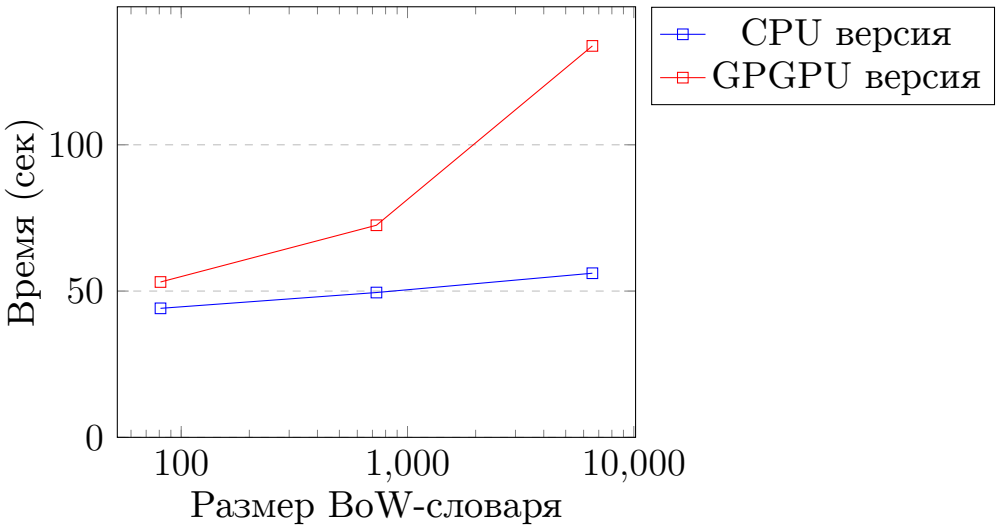

Рис. 4: Сравнение эффективности GPGPU и CPU версий

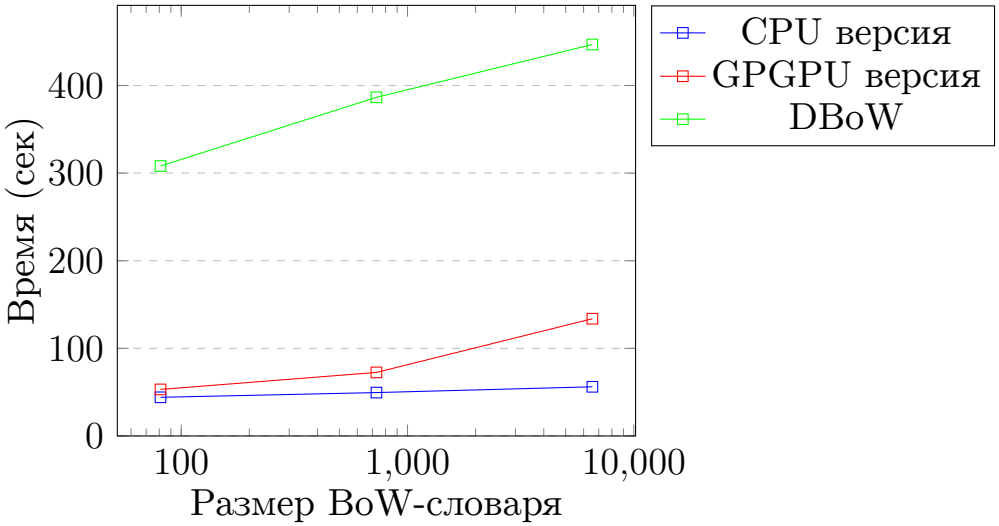

Рис. 5: Сравнение эффективности DBoW и реализованных версий алгоритма

3.4. Оценка эффективности алгоритмов построения BoW-словаря

Из-за высокой ресурсоёмкости задачи построения BoW-словаря была разработана многопоточная версия данного алгоритма. С целью оценки итогового результата была поставлена серия экспериментов (Рис. 5). В качестве опорного решения был выбран алгоритм DBoW [5]. Причинами выбора этого алгоритма являются распространенность в научном сообществе [11].

Версия алгоритма	Время (сек)
DBoW	446
реализованный алгоритм	56

Таблица 1: Сравнение алгоритмов сопоставления BoW-векторов

4. Алгоритм сопоставления BoW-векторов

В задаче детектирования замыкания цикла новое изображение сопоставляется с ранее полученными. Для этого производится сравнение BoW-векторов сопоставляемых изображений. Процесс сопоставления является независимым для любой пары изображений. Данный факт позволяет производить этап сопоставления параллельно для всех пар изображений. Была разработана параллельная версия алгоритма под CPU. Поставлены эксперименты для оценки эффективности полученного решения (Таб. 1). Измерения производились на BoW-словаре содержащем 59049 визуальных слов, было сделано 5000 запросов на сопоставление к базе данных, содержащей 5000 изображений.

Дескриптор	Точность	Время (сек)
ORB	0.43	31
AKAZE	0.64	210
KAZE	0.63	339

Таблица 2: Сравнение дескрипторов

5. Сравнение различных дескрипторов

Были отобраны следующие дескрипторы: ORB, KAZE и AKAZE. Данные дескрипторы популярны в научном сообществе и имеют реализации в открытых библиотеках компьютерного зрения [12]. Для оценки точности результатов был выбран открытый набор данных KITTI [8]. На основе предоставленных данных о местоположении наблюдателя был выбран участок маршрута, который был посещен дважды. В качестве изображений, которые хранятся в базе данных, было взято 10 последовательных снимков выбранного участка маршрута. В качестве изображений для запросов к базе данных были взяты 10 последовательных изображений повторного проезда по выбранного участка маршрута. В качестве меры близости дескрипторов использовалось расстояние Хемминга, оценка близости BoW-векторов производилась по средствам метрики L1. Итоговые значения точности при использовании различных дескприторов были получены по средствам усреднения значений результатов запросов к базе данных.

Кроме точности получаемых результатов важной характеристикой является ресурсоёмкость вычисления дескриптора особой точки изображения. Временные характеристики получены для 1000 изображений с использованием библиотеки OpenCV (Таб. 2).

Заключение

В результате выполнения работы были получены следующие результаты:

- проведено исследование предметной области;
- разработаны две версии многопоточного алгоритма построения BoW-словаря для дескрипторов произвольной размерности на CPU и GPGPU;
- разработан многопоточный алгоритм построения BoW-вектора для дескрипторов изображения и поиска ближайших изображений;
- проведена оценка классических дескрипторов совместно с разработанными алгоритмами.

Список литературы

[1] Alcantarilla Pablo Fernández, Bartoli Adrien, Davison Andrew J. KAZE features // European Conference on Computer Vision / Springer. — 2012. — P. 214–227.

[2] Automatic Brain Tumor Segmentation Using Cascaded Anisotropic Convolutional Neural Networks / Guotai Wang, Wenqi Li, Sébastien Ourselin et al. // Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries. — Cham : Springer International Publishing, 2018. — P. 178–190.

[3] Betke Margrit, Gurvits Leonid. Mobile robot localization using landmarks // IEEE transactions on robotics and automation. — 1997. — Vol. 13, no. 2. — P. 251–263.

[4] Compute Unified Device Architecture Main Page. — `https://developer.nvidia.com/cuda-toolkit`. — Accessed: 2019-05-05.

[5] DBoW2 Main Page. — `https://github.com/dorian3d/DBoW2`. — Accessed: 2019-05-05.

[6] Gu Yu-Hua, Tjahjadi Tardi. Corner-based feature extraction for object retrieval // Proceedings 1999 International Conference on Image Processing (Cat. 99CH36348) / IEEE. — Vol. 1. — 1999. — P. 119–123.

[7] Intel Threading Building Blocks Main Page. — `https://www.threadingbuildingblocks.org/`. — Accessed: 2019-05-05.

[8] KITTI Main Page. — `http://www.cvlibs.net/datasets/kitti/eval_odometry.php`. — Accessed: 2019-05-05.

[9] Leonard John J, Durrant-Whyte Hugh F. Mobile robot localization by tracking geometric beacons // IEEE Transactions on robotics and Automation. — 1991. — Vol. 7, no. 3. — P. 376–382.

[10] Mikolajczyk Krystian, Zisserman Andrew, Schmid Cordelia. Shape recognition with edge-based features // British Machine Vision

Conference (BMVC'03) / The British Machine Vision Association. — Vol. 2. — 2003. — P. 779–788.

[11] Mur-Artal Raul, Montiel Jose Maria Martinez, Tardos Juan D. ORB-SLAM: a versatile and accurate monocular SLAM system // IEEE transactions on robotics. — 2015. — Vol. 31, no. 5. — P. 1147–1163.

[12] OpenCV Main Page. — `https://opencv.org/`. — Accessed: 2019-05-05.

[13] Wong-Riley Margaret. Changes in the visual system of monocularly sutured or enucleated cats demonstrable with cytochrome oxidase histochemistry // Brain research. — 1979. — Vol. 171, no. 1. — P. 11–28.

[14] A computer vision approach for textile quality control / Christos Anagnostopoulos, D Vergados, Eleftherios Kayafas et al. // The Journal of Visualization and Computer Animation. — 2001. — Vol. 12, no. 1. — P. 31–44.

[15] The slam problem: a survey. / Josep Aulinas, Yvan R Petillot, Joaquim Salvi, Xavier Lladó // CCIA. — 2008. — Vol. 184, no. 1. — P. 363–371.