Санкт-Петербургский государственный университет

Кафедра системного программирования

Паршин Максим Алексеевич

Реализация алгоритма сопоставления результатов детекции торцов брёвен на изображениях штабелей круглого леса

Отчёт по учебной практике

Научный руководитель:
ст. преп. кафедры системного программирования
Смирнов М. Н.

Консультант:
менеджер проектов ООО «Системы Компьютерного Зрения»
Ткачёва Д. А.

Санкт-Петербург
2021

Оглавление

Введение **3**

1. Постановка задачи **6**

2. Обзор **7**

2.1. SIFT 7

2.2. JavaCV 8

3. Описание алгоритма **10**

3.1. Поиск торцов, лежащих рядом с границей пользовательской области 13

3.2. Поиск сопоставления между торцами 14

4. Реализация алгоритма **19**

5. Тестирование алгоритма **21**

5.1. Тестовые данные 21

5.2. Критерии оценки 22

5.3. Результаты тестирования 23

6. Заключение **26**

Список литературы **27**

Введение

Измерение таких характеристик штабелей круглого леса и лесовозов, как объем, высота, ширина, коэффициент полнодревесности[1] — неотъемлемая часть работы предприятий лесной промышленности. На многих предприятиях замеры осуществляются вручную, вследствие чего человеческий фактор может отрицательно сказываться на точности результатов. В связи с этим существует потребность в автоматизации процесса подсчета древесины. Наиболее простым для внедрения способом автоматизации является использование мобильного приложения, осуществляющего распознавание торцов бревен на фотографиях штабелей и лесовозов и рассчитывающего необходимые характеристики по полученным данным.

Рис. 1: Крупные штабели круглого леса

Штабели леса могут иметь большую длину (Рис. 1), что не позволяет пользователю поместить все торцы в один кадр, как это можно сделать с лесовозами. Одно из возможных решений данной проблемы — использование нескольких последовательных пересекающихся кадров штабеля. При данном подходе возникает задача определения торцов, которые присутствуют на обеих соседних фотографиях, и исключения

[1] Доля объёма пригодной для использования древесины

их на одном из изображений. Эта задача может быть решена полуавтоматически: пользователь отмечает область интереса на каждом из кадров, и те торцы, что не попали в область, не учитываются при подсчёте. При этом остаётся потребность в алгоритме, исключающем дублирующиеся торцы, лежащие на правой границе выделенной области на левом изображении и на левой границе на правом.

Идея простого алгоритма, решающего в том числе и задачу исключения дублирующихся торцов на границе, заключается в том, чтобы на левом изображении из пары учитывать все торцы, пересекающие область, а на правом учитывать только те торцы, которые целиком лежат в ней (Рис. 2). Недостаток данного подхода заключается в том, что он предполагает точное выделение области, которое может быть невозможным на практике по следующим причинам:

- ориентируясь на физические метки, такие как линейки, прикрепленные к торцам, или отметки краской, пользователи проводят лишь примерную границу между частями штабеля;
- торцы брёвен могут быть утопленными или выпирающими, вследствие чего возникает эффект параллакса, и торцы, лежащие у линии границы на одном изображении, на другом изображении могут быть перекрыты торцами других брёвен.

Рис. 2: Результат работы простого алгоритма исключения дублирующихся торцов. Красным выделена отмеченная пользователем область интереса, жёлтым — торцы, которые будут учтены при подсчёте объёма

Для того, чтобы улучшить результаты работы алгоритма в описанных условиях, может быть произведено сопоставление торцов, лежащих

около границы пользовательской области на соседних изображениях, и установление взаимно-однозначного соответствия между ними. С учётом найденного соответствия могут быть исключены не исключённые ранее дубликаты торцов либо добавлены торцы, исключённые ошибочно. Таким образом, возникает задача построения дескрипторов и сопоставления особых точек на изображении, где в качестве особых точек выступают торцы брёвен.

1. Постановка задачи

Цель данной работы — уменьшить число торцов брёвен, учтённых дважды или не учтённых при подсчёте объёма штабеля из нескольких частей.

Для достижения поставленной цели были сформулированы следующие задачи:

- разработать модификацию текущего алгоритма исключения дубликатов торцов, использующую автоматическое сопоставление результатов детекции между соседними изображениями;
- реализовать модифицированную версию алгоритма;
- отобрать и подготовить данные для тестирования (последовательные фотографии частей штабеля с установленными вручную сопоставлениями);
- провести тестирование и соответствующую доработку алгоритма;
- подготовить итоговую версию алгоритма к интеграции в существующее мобильное приложение для подсчёта объёма древесины.

2. Обзор

Далее будут рассмотрены технологии, использованные при реализации алгоритма.

2.1. SIFT

Для решения задачи сопоставления торцов требуется определить набор свойств, по которым торец бревна может сравниваться с другими торцами. В набор таких свойств, используемых разработанным алгоритмом, входит в том числе и структура древесины внутри торца, в связи с чем необходимо иметь дескриптор, описывающий пиксели в сегменте изображения, содержащем торец (Рис. 3). В частности, могут быть использованы такие дескрипторы особых точек как SIFT (Scale-Invariant Feature Transform) [2], BRIEF (Binary Robust Independent Elementary Features) [1] и другие. В предложенной реализации разработанного алгоритма используется дескриптор SIFT, однако реализация не предполагает использование конкретно данного дескриптора, и другие дескрипторы особых точек могут быть интегрированы в её исходный код.

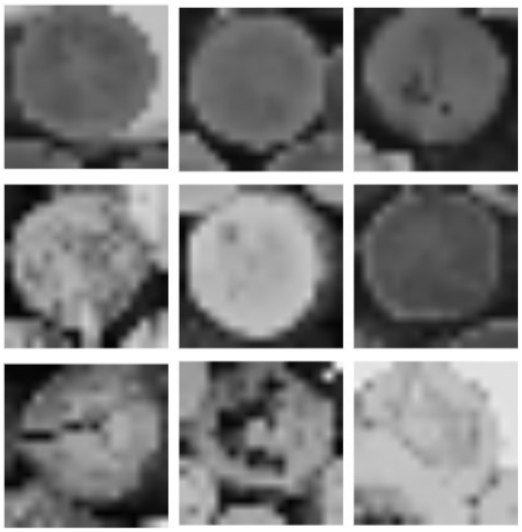

Рис. 3: Примеры сегментов изображения с торцами, в которых строится дескриптор

Дескриптор SIFT использует информацию о направлении и вели-

чине градиента функции интенсивности изображения в пикселе $I(x, y)$. Окрестность особой точки разбивается на несколько ячеек (Рис. 4), для каждой из которых строится гистограмма распределения градиента по восьми интервалам длиной $\frac{\pi}{4}$. Гистограммы ячеек записываются в виде массивов из восьми элементов, а затем они конкатенируются в один массив, который и является дескриптором.

Чтобы добиться инвариантности относительно поворота, дескриптор также должен описывать некоторое конкретное направление, по которому он будет ориентирован при сравнении. В SIFT ориентация определяется как преобладающее направление градиента функции интенсивности в окрестности особой точки.

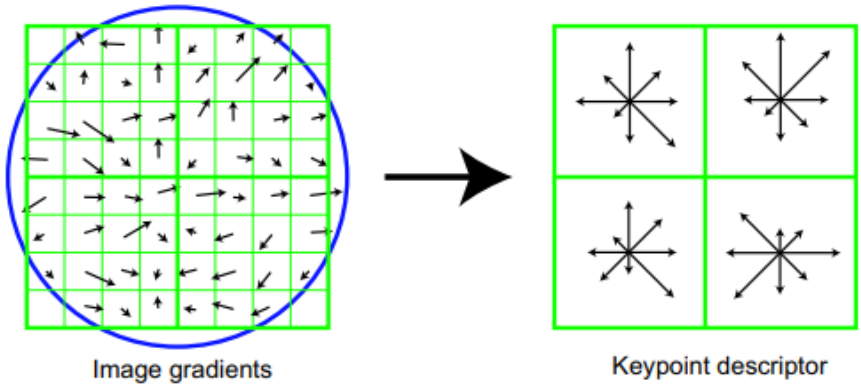

Рис. 4: Процесс построения дескриптора SIFT для ячейки четыре на четыре. Источник: [2]

Следует отметить, что, как правило, дескриптор SIFT используется для описания особых точек, найденных при помощи одноименного детектора, как дополнение к которому он и описан в [2], однако в данном случае он применяется иным образом. Вместо окрестности 16 на 16 пикселей вокруг найденной ключевой точки дескриптор строится в масштабированном до данного размера сегменте изображения, содержащем торец.

2.2. JavaCV

Поскольку исходный код Android-приложения для подсчёта объёма древесины, в которое было необходимо интегрировать разработан-

ный алгоритм, написан на языке Kotlin, требовалось выбрать Java-библиотеку, содержащую реализацию дескриптора SIFT. Так как некоторые другие алгоритмы компьютерного зрения, применяемые в приложении, используют методы библиотеки JavaCV[2], было принято решение воспользоваться предоставляемой ей реализацией SIFT.

JavaCV является Java-интерфейсом для библиотеки компьютерного зрения OpenCV[3], позволяющим использовать методы OpenCV, реализованные на C++, из кода на Java или Kotlin.

[2] `https://github.com/bytedeco/javacv` Дата обращения: 31.05.2021

[3] `https://opencv.org` Дата обращения: 31.05.2021

3. Описание алгоритма

Разработанный алгоритм принимает на вход следующие данные:

- два последовательных изображения штабеля (левое и правое), имеющих область перекрытия (торцы, лежащие в этой области, есть на обоих изображениях);
- многоугольник выделенной пользователем области интереса для каждого из изображений;
- результаты распознавания торцов брёвен на изображениях, представленные в виде боксов («bounding box»), описывающих торцы.

Результатом работы алгоритма являются разбиения множества боксов левого изображения на подмножества S_{LL} и S_{LR} и множества боксов правого изображения на подмножества S_{RL} и S_{RR} (Рис. 5). При этом требуется, чтобы:

- множества торцов, соответствующие боксам из S_{LL} и S_{RR} не пересекались;
- для каждого торца, которому соответствует бокс из S_{LR}, существовал соответствующий бокс из S_{RR};
- для каждого торца, которому соответствует бокс из S_{RL}, существовал соответствующий бокс из S_{LL}.

Процесс работы алгоритма можно разделить на следующие шаги.

1. **Поиск торцов, лежащих рядом с границей пользовательской области**. Поскольку торцы, лежащие далеко от правой границы области на левом изображении и далеко от левой границы области на правом изображении, могут быть отнесены к определенному изображению без поиска соответствия между ними, алгоритм ищет сопоставления только для торцов, лежащих близко к этим границам. Суть данного шага заключается в том, чтобы

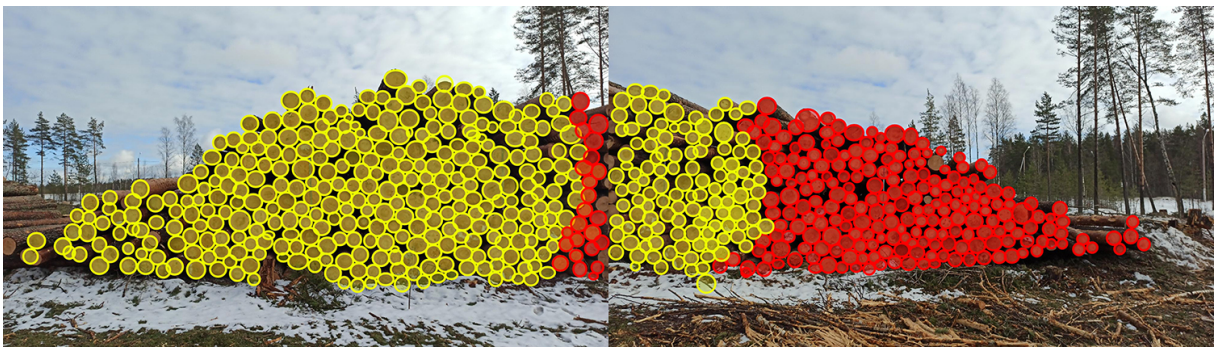

Рис. 5: Итоговое разбиение множеств торцов. На левом изображении множество S_{LL} показано жёлтым, множество S_{LR} показано красным. На правом изображении множество S_{RL} показано жёлтым, множество S_{RR} показано красным

определить правую границу области для левого изображения и левую для правого и найти торцы, лежащие на заданном расстоянии от них (Рис. 6).

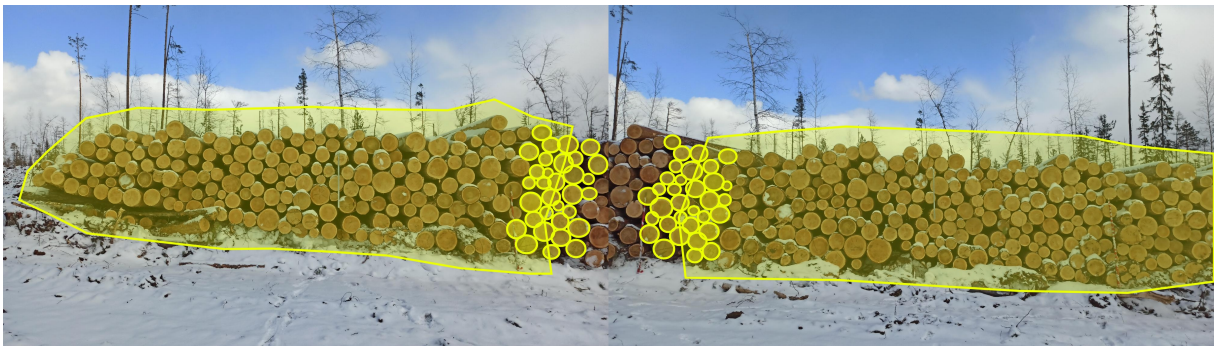

Рис. 6: Торцы брёвен, лежащие рядом с границей пользовательской области, между которыми требуется найти сопоставление

2. **Поиск сопоставления между торцами**. На данном шаге происходит поиск взаимно-однозначного соответствия между боксами, лежащими рядом с границами на левом и правом изображениях, и соответствующим фактически одному и тому же торцу (Рис. 7).

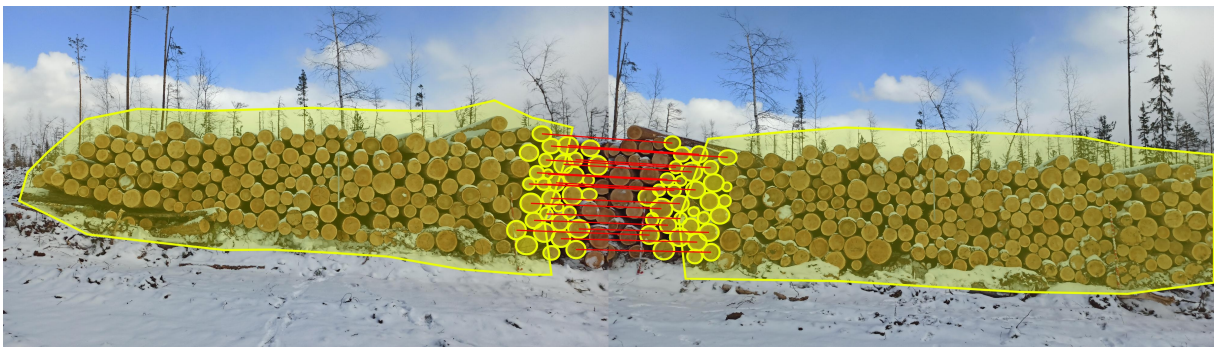

Рис. 7: Найденное сопоставление между торцами, лежащими рядом с границей пользовательской области

3. **Первичная фильтрация торцов по пользовательской области**. Данный шаг соответствует простому алгоритму исключения дубликатов торцов, описанному во Введении. В множестве на левом изображении остаются только торцы, пересекающие область, а на правом — торцы, целиком лежащие в ней (Рис. 8).

Рис. 8: Торцы, отфильтрованные по пользовательской области

4. **Уточнение границы**. На данном шаге происходит проверка того, что торцы, для которых найдено сопоставление на шаге два, лежат в множестве, найденном на шаге три, на левом изображении и исключены из множества на правом изображении. В случае, если это не так, торец добавляется в результирующее множество на левом изображении, и сопоставленный ему торец исключается из множества на правом изображении (Рис. 9).

Далее первый и второй шаги алгоритма будут рассмотрены более подробно.

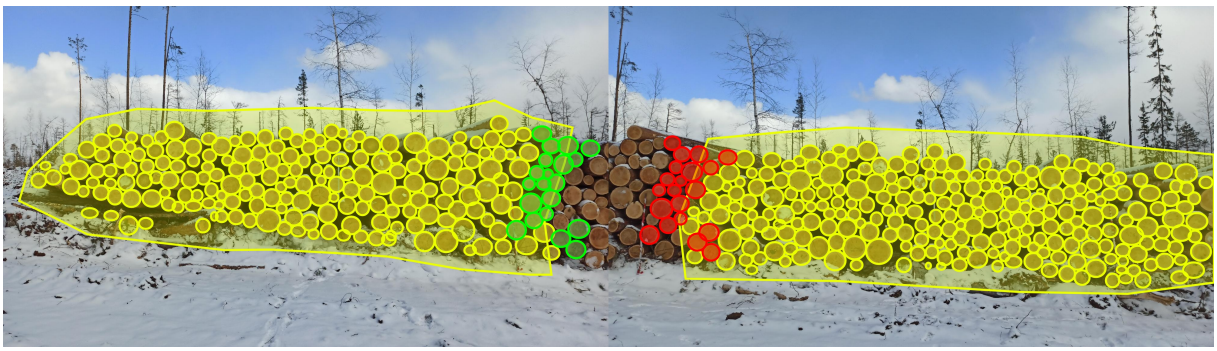

Рис. 9: Уточнение границы по найденным сопоставлениям. Торцы, для которых сопоставление было найдено, отмечены зелёным на левом изображении и красным на правом и будут включены в результирующее множество слева и исключены из него справа

3.1. Поиск торцов, лежащих рядом с границей пользовательской области

Для того, чтобы найти торцы брёвен, лежащие близко к правой границе пользовательской области на левом изображении и близко к левой границе на правом, необходимо определить стороны многоугольника, соответствующие этим границам.

1. Из каждой вершины многоугольника на левом/правом изображении проводится горизонтальная линия в направлении правого/левого края изображения.

2. Проверяется, какие из линий имеют пересечение с многоугольником (Рис. 10).

3. Стороны многоугольника, у которых из обоих концов проведены линии, не имеющие пересечение с многоугольником, считаются сторонами, составляющими границу.

После того, как найдены стороны многоугольников, составляющие границу, в результирующие множества для левого и правого изображения добавляются торцы, центры которых лежат на не большем, чем заданный порог, расстоянии от границы (Рис. 11).

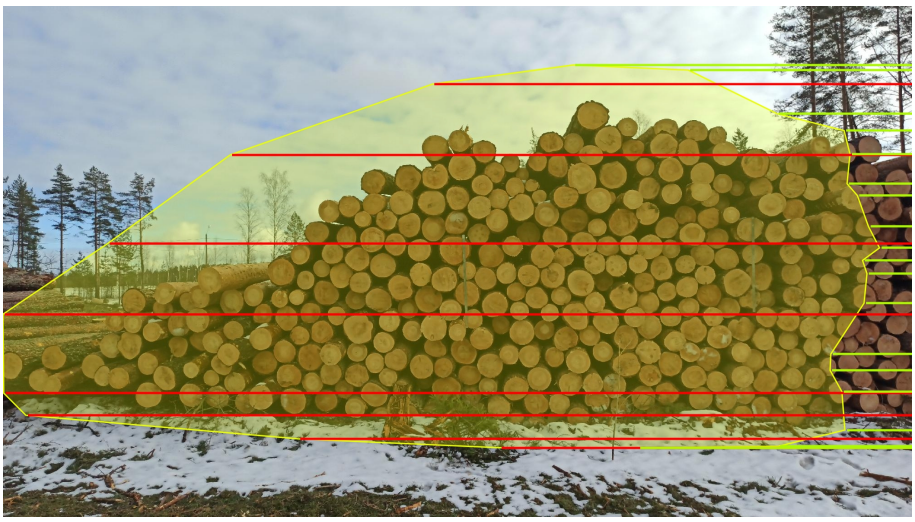

Рис. 10: Линии, проведённые из вершин многоугольника на левом изображении. Красным отмечены линии, имеющие пересечение с многоугольником, зелёным — не имеющие

Чтобы исключить из результирующих множеств торцы, которые могли ошибочно попасть туда и фактически не лежат около правой/левой границы, осуществляется дополнительная фильтрация.

1. Торцы из результирующего множества сортируются по x-координате центра по убыванию для левого изображения и по возрастанию для правого.

2. Просматриваются i и $i + 1$ элементы списка до тех пор, пока расстояние между их центрами по оси x не превышает заданного порога.

3. В результирующих множествах остаются только просмотренные торцы.

3.2. Поиск сопоставления между торцами

При поиске взаимно-однозначного соответствия между торцами учитываются следующие сравнительные характеристики.

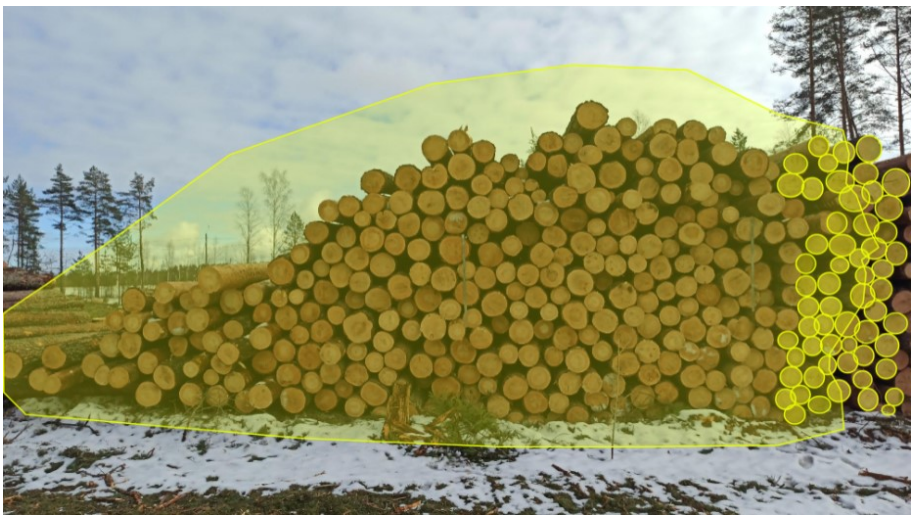

Рис. 11: Торцы, лежащие близко к найденной границе

- **Близость дескрипторов** (например, SIFT) сегментов изображения, включающих торцы. Таким образом учитывается сходство структуры древесины внутри торцов.
- **Вертикальное положение** центров торцов относительно штабеля. Торцы брёвен, лежащих вверху/внизу штабеля на левом изображении, должны соответствовать торцам брёвен, лежащих вверху/внизу штабеля на правом.
- **Число сходных торцов, лежащих в окрестности**. Сопоставленные торцы должны иметь в окрестностях некоторое число сходных торцов, лежащих под близкими углами.

Сначала происходит поиск k ближайших торцов, а затем полученное сопоставление «один-ко-многим» фильтруется и преобразуется во взаимно-однозначное.

1. Для каждого торца, лежащего около правой границы области на левом изображении, осуществляется поиск k торцов, лежащих около левой границы области на правом изображении и имеющих дескрипторы, наиболее сходные с его дескриптором (Рис. 12).

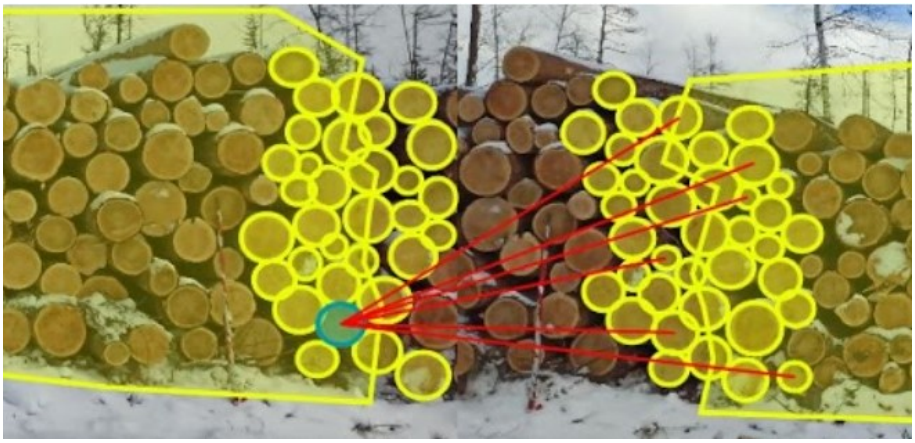

Рис. 12: Сопоставление торцу на левом изображении k торцов на правом изображении с наиболее сходными дескрипторами

2. Отбрасываются сопоставления пар торцов, у которых разница между относительными y-координатами центров больше, чем пороговое значение (Рис. 13).

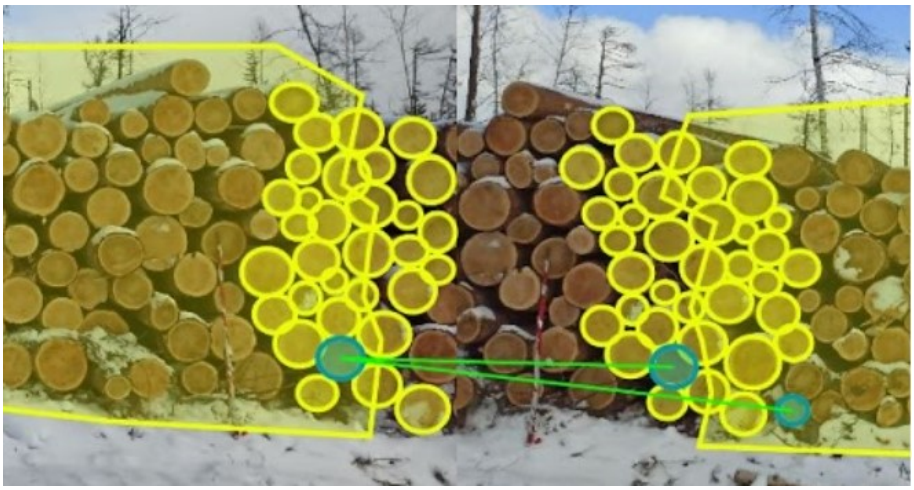

Рис. 13: Оставшиеся сопоставления с торцами, лежащими на сходной высоте

3. Для каждого сопоставленного торца слева и справа рассматриваются окрестности: n торцов с наименьшими расстояниями до его центра (Рис. 14).

4. Для каждого торца в окрестности на левом изображении рассматриваются торцы на правом изображении, сопоставленные ему на шаге два (Рис. 15).

Рис. 14: Торцы, лежащие в окрестностях сопоставленных торцов (отмечены синим)

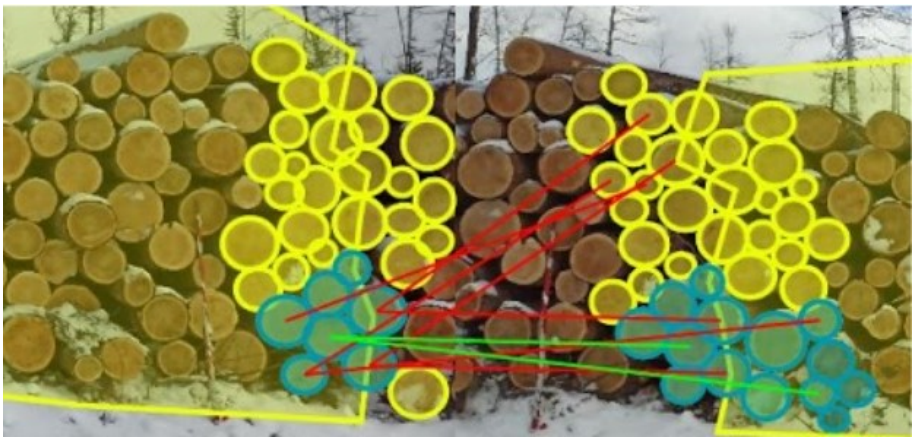

Рис. 15: Сопоставления торцам из окрестности на левом изображении торцов на правом изображении с наиболее сходными дескрипторами

5. Отбрасываются сопоставления, для которых малое число соседей слева было сопоставлено соседям справа. Предполагается, что в случае правильно найденного сопоставления у сопоставленных торцов должно быть сопоставлено и большинство соседей.

6. Если после предыдущих шагов для какого-либо из торцов осталось больше одного сопоставления, выбирается сопоставление с торцом, имеющим наиболее близкий дескриптор (Рис. 16). В результате каждому торцу, лежащем рядом с границей на левом изображении, сопоставлено не более одного торца, лежащего рядом с границей на правом изображении.

7. Поскольку всё ещё возможна ситуация, когда один и тот же торец

Рис. 16: Торцу на левом изображении сопоставлен только один торец на правом

на правом изображении был сопоставлен более чем одному торцу на левом изображении, строится обратное сопоставление и аналогично предыдущему шагу выбирается единственное сопоставление с торцом, имеющим наиболее близкий дескриптор. В результате каждому торцу слева сопоставлено не более одного торца справа и каждый торец справа сопоставлен не более чем одному торцу слева.

8. Происходит добавление дополнительных сопоставлений по найденным. Если в окрестности левого торца есть еще не сопоставленный торец и в окрестности правого торца есть еще не сопоставленный торец, лежащий под сходным углом, добавляется сопоставление между ними (Рис. 17).

Рис. 17: Дополнительно добавленные сопоставления (отмечены красным)

4. Реализация алгоритма

Для подготовки алгоритма к интеграции в мобильное приложение логика его работы была декомпозирована на шаги, и данные шаги были реализованы в соответствующих классах (Рис. 18). Реализация алгоритма содержится в модуле, отличном от модуля Android-приложения, где реализованы и прочие алгоритмы, уже используемые в нём. Данный модуль не имеет зависимостей от Android SDK, что облегчает тестирование.

- **MatchingInteractorImpl** Класс, реализующий интерфейс MatchingInteractor, который предоставляет модулю Android-приложения методы для работы с алгоритмом.
- **EntireLogSiftKDetectionMatchingAlgorithm** Класс, содержащий логику построения дескрипторов SIFT для торцов брёвен и поиска k ближайших соседей для данных дескрипторов.
- **OneToOneDetectionMatchingAlgorithmImpl** Класс, содержащий логику поиска взаимно-однозначного сооветствия между торцами.
- **BoundaryDetectionsSearchAlgorithmImpl** Класс, содержащий логику поиска торцов, лежащих рядом с границей пользо-

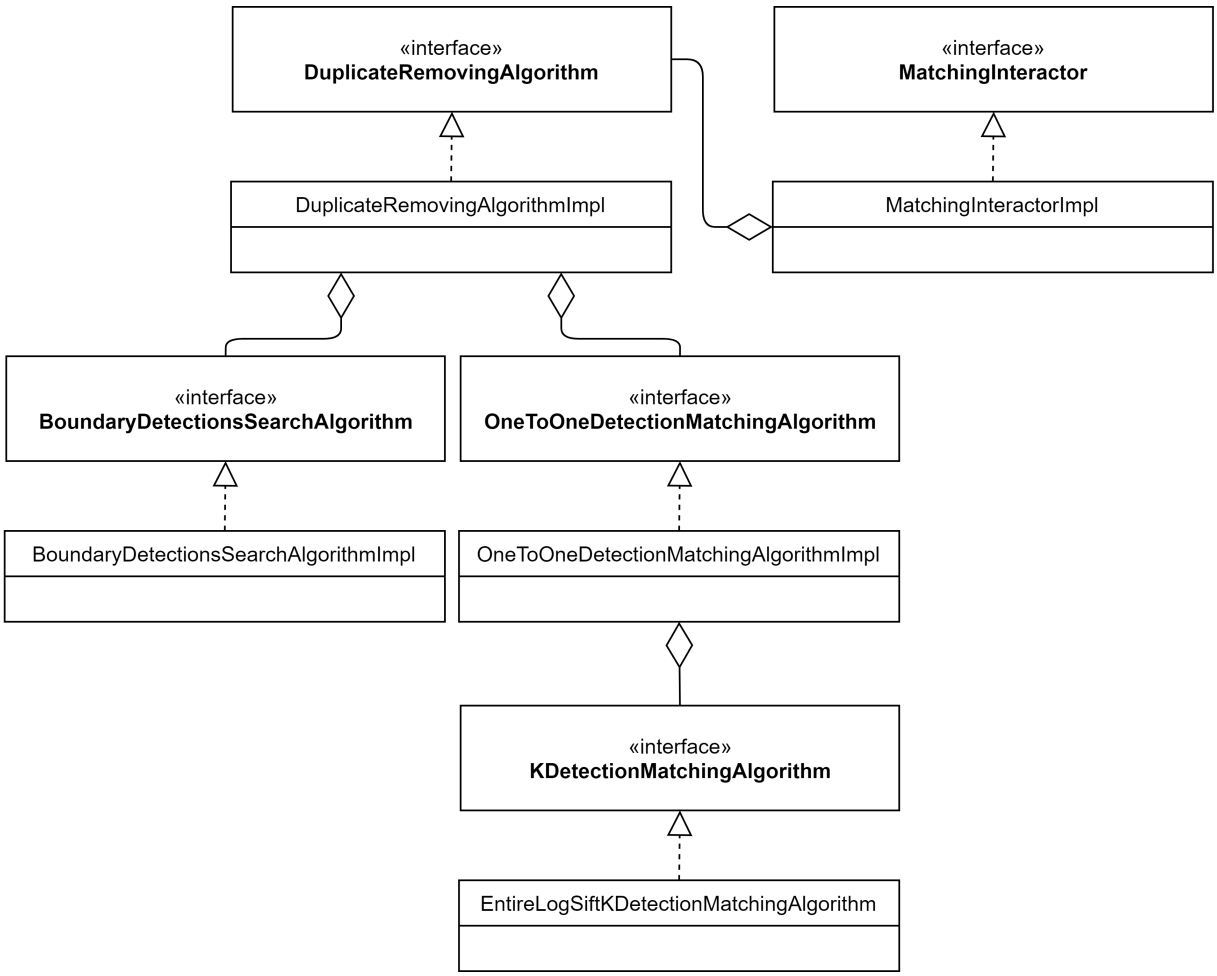

Рис. 18: Диаграмма классов, реализующих алгоритм

вательской области.

- **DuplicateRemovingAlgorithmImpl** Класс, содержащий логику поиска итогового разбиения множеств торцов.

5. Тестирование алгоритма

Было проведено тестирование улучшенного алгоритма и его сравнение с простым алгоритмом, описанным во Введении.

5.1. Тестовые данные

Для тестирования алгоритма были отобраны четыре пары последовательных фотографий штабелей (Рис. 19). Между торцами, присутствующими на обоих изображениях, было вручную установлено взаимно-однозначное соответствие. Для каждой пары изображений также было подготовлено несколько вариантов возможной разметки пользовательской области различной точности.

Рис. 19: Тестовые изображения

5.2. Критерии оценки

Работа алгоритма оценивалась по следующим критериям:

- качество найденного в процессе работы алгоритма взаимно-однозначного соответствия между торцами;

- качество работы алгоритма в целом, точность разбиения множества торцов в сравнении с простым алгоритмом, описанным во Введении.

Для оценки качества найденного взаимно-однозначного соответствия между торцами использовались метрики precision и recall [3]:

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

В данных формулах TP — число верно найденных сопоставлений, FP — число неверно найденных сопоставлений, FN — число ненайденных сопоставлений.

Для оценки качества работы алгоритма в целом использовался вариант метрики Simple Matching Coefficient [4], показывающей схожесть полученных для левого и правого изображения разбиений множеств торцов:

$$SMC = \frac{S_{IE} + S_{EI}}{2S}$$

В данной формуле S_{IE} — число торцов, включённых в множество S_{LL} и не включённых в S_{RR}, S_{EI} — число торцов, не включённых в множество S_{LL} и включённых в S_{RR}, S — число торцов, присутствующих и на левом, и на правом изображении.

Метрика Simple Matching Coefficient была также рассчитана для простого алгоритма фильтрации торцов по пользовательской области, описанного во Введении, и был оценен прирост этой метрики для улучшенного алгоритма.

5.3. Результаты тестирования

В таблице 1 приведены значения описанных метрик для различных наборов входных данных (различные пары изображений и различные варианты разметки пользовательской области).

набор входных данных	precision	recall	SMC, простой алгоритм	SMC, улучшенный алгоритм	прирост SMC
1	1,000	0,828	1,000	1,000	0,000
2	1,000	0,839	1,000	1,000	0,000
3	0,942	0,733	0,942	0,962	0,019
4	0,962	0,774	0,962	1,000	0,038
5	0,827	0,786	0,827	0,981	0,154
6	1,000	0,706	1,000	1,000	0,000
7	0,981	0,933	0,981	1,000	0,019
8	0,975	0,633	0,975	0,983	0,008
9	0,983	0,787	0,983	0,992	0,008
10	0,916	0,653	0,916	0,966	0,050
11	0,958	0,776	0,958	0,983	0,025
12	0,916	0,620	0,916	0,958	0,042
13	0,916	0,650	0,916	0,983	0,067
14	0,933	0,773	0,933	0,966	0,034
15	0,986	0,795	0,986	0,986	0,000
16	0,986	0,676	0,986	0,986	0,000
17	0,945	0,677	0,945	0,973	0,027
18	0,945	0,868	0,945	1,000	0,055
19	0,986	0,744	0,986	0,986	0,000
20	0,836	0,939	0,836	0,986	0,151
21	0,978	0,585	0,978	0,978	0,000
22	0,972	0,615	0,972	0,978	0,006
23	0,905	0,586	0,905	0,961	0,056
24	0,927	0,662	0,927	0,939	0,011
25	0,939	0,594	0,939	0,972	0,034
26	0,950	0,688	0,950	0,961	0,011
27	0,950	0,658	0,950	0,966	0,017

Таблица 1: Значения метрик на различных наборах входных данных

Анализируя полученные результаты, можно сделать следующие выводы.

- Ни на одном из наборов входных данных использование улучшенного алгоритма не привело к ухудшению результата (уменьшению SMC).
- В большинстве случаев получен прирост SMC, причём в некоторых случаях максимально возможный: к примеру, на наборе входных данных номер семь было получено значение SMC, равное единице, что означает отсутствие учтённых дважды или не учтённых торцов.
- Прирост качества работы достигается, несмотря на то что репрезентативность множества взаимно-однозначно сопоставленных пар (recall) в большинстве случаев невелика, в отличие от точности (precision).

6. Заключение

Все поставленные цели были достигнуты:

- разработана модификация текущего алгоритма исключения дубликатов торцов, использующая автоматическое сопоставление результатов детекции между соседними изображениями;
- реализована модифицированная версия алгоритма;
- отобраны и подготовлены данные для тестирования (последовательные фотографии частей штабеля с установленными вручную сопоставлениями);
- проведено тестирование и соответствующая доработка алгоритма;
- итоговая версия алгоритма подготовлена к интеграции в существующее мобильное приложение для подсчёта древесины.

Исходный код реализации алгоритма загружен в закрытый репозиторий проекта.

Список литературы

[1] Calonder M., Lepetit V., Strecha C., Fua P. BRIEF: Binary robust independent elementary features // European conference on computer vision / Springer. — 2010. — P. 778–792.

[2] Lowe D. G. Distinctive image features from scale-invariant keypoints // International journal of computer vision. — 2004. — Vol. 60, no. 2. — P. 91–110.

[3] Powers, David. Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness and Correlation // Mach. Learn. Technol. — 2008. — 01. — Vol. 2.

[4] Rand W. Objective Criteria for the Evaluation of Clustering Methods // Journal of the American Statistical Association. — 1971. — Vol. 66. — P. 846–850.