Санкт-Петербургский государственный университет

Программная инженерия

Кутуев Владимир Александрович

Реализация кросс-платформенной библиотеки цифровой обработки изображений в мобильной микроскопии

Бакалаврская работа

Научный руководитель:
доц., к.т.н. Ю. В. Литвинов

Консультант:
ст. преп. Я. А. Кириленко

Рецензент:
глава мобильной разработки ООО «Мел Саенс» П. М. Катунин

Санкт-Петербург
2020

SAINT PETERSBURG STATE UNIVERSITY

Software Engineering

Vladimir Kutuev

Implementation of a cross-platform library for digital image processing in mobile microscopy

Graduation Thesis

Scientific supervisor:
C.Sc., Associate Professor Yurii Litvinov

Consultant:
Senior Lecturer Iakov Kirilenko

Reviewer:
head of mobile development at MEL Science Pavel Katunin

Saint Petersburg
2020

Оглавление

Введение

С развитием технологий оптической микроскопии стоимость простых микроскопов стала достаточно низкой для их массового использования, что делает процесс изучения микроорганизмов более доступным. Однако сама оптическая микроскопия как подход имеет ряд недостатков по сравнению с её современными аналогами, например, сканирующими электронными микроскопами. Основным недостатком оптического микроскопа является малая глубина резкости, что не позволяет рассмотреть образец целиком. Пользователю приходится механически изменять фокусное расстояние, чтобы получить полную информацию об образце. Вторым существенным недостатком, если говорить о бюджетном сегменте оптических микроскопов, является их разрешающая способность, которую можно улучшить только за счёт дорогой оптики.

Главным решением данных проблем является цифровая обработка изображений, позволяющая как повысить качество получаемого изображения, так и объединить информацию о разных частях объекта на одном изображении. Для улучшения качества изображения в микроскопии используется технология деконволюции — зная свойства оптической системы, можно понять, какие оптические эффекты она создает на изображении (свечение, размытие) и выполнить обратное преобразование, восстанавливающее оригинальное изображение. Для решения проблемы рассмотрения объемного образца используется техника фокус-стекинга — метода, который объединяет несколько изображений, сделанных с разными фокусными расстояниями. Оба подхода достаточно хорошо изучены, существует большое количество инструментов для ПК в данных областях, например, Zerene Stacker [49], Helicon Focus [27], Extended Depth of Field Plugin for ImageJ [10], Affinity Photo [46], Adobe Photoshop [1].

Однако для цифровой обработки изображений необходимо устройство, производящее захват кадров. Комплектация оптических микроскопов камерой значительно увеличивает их стоимость. Инструментом, который всегда есть под рукой, является смартфон. В статье

«Quantitative Imaging with a Mobile Phone Microscope» [43] было показано, что с использованием камер современных смартфонов можно захватывать видео без значительной потери разрешающей способности, что лишь подтверждает возможность использования смартфона для захвата изображений с микроскопа.

Несмотря на широкий выбор программ для обработки кадров с микроскопа для ПК, задача исполнения базовых алгоритмов обработки изображений непосредственно на смартфоне всё ещё остается актуальной. Это подтверждает большое количество исследований, посвященных применению мобильной микроскопии для быстрой диагностики в чрезвычайных ситуациях в «полевых» условиях и для диагностики заболеваний в развивающихся странах: диагностика малярии и туберкулеза [4], рака шейки матки и гортани [38]. Таким образом, помимо захвата кадров, видится рациональным использование вычислительных ресурсов смартфона для обработки кадров с целью повышения качества изображений для дальнейшей диагностики.

В рамках данной работы предлагается разработать кроссплатформенную библиотеку для цифровой обработки изображений в микроскопии. Для этого необходимо выделить основные шаги по улучшению качества изображения, которые выполняют в оптической микроскопии, предложить процесс их автоматизации. Также необходимо найти баланс между производительностью и качеством алгоритмов на смартфоне — не всегда дефекты обработки, видимые на дисплее ПК при сравнении, заметны на дисплее мобильного телефона. Библиотека должна быть разработана с возможностью быстрого расширения новыми алгоритмами и отладки на ПК. Данная работа выполняется совместно с курсовой студента третьего курса направления «Программная инженерия» Дмитрия Яроша, в рамках которой производится обзор, реализация и сравнение существующих алгоритмов автоматического выбора сфокусированных кадров и фокус-стекинга.

1. Постановка задачи

Целью данной работы является разработка кросс-платформенной библиотеки для цифровой обработки изображений в мобильной микроскопии. Важным аспектом работы библиотеки является соотношение «производительность-качество» для выбора наиболее оптимальной комбинации алгоритмов. Для достижения цели были поставлены следующие задачи.

1. Выполнить обзор методов цифровой обработки изображений в микроскопии, выделить методы, которые могут быть применимы в мобильной микроскопии.

2. Выполнить обзор и сравнение мобильных приложений для фотосъемки в микроскопии и макросъёмке.

3. Выполнить обзор инструментов для создания кросс-платформенных библиотек для мобильных устройств.

4. Реализовать кросс-платформенную мобильную библиотеку для цифровой обработки изображений в микроскопии с учетом выбранных инструментов.

5. Реализовать систему для визуального сравнения результатов работы алгоритмов и провести опрос для сравнения их качества.

6. Создать прототип для замера скорости работы алгоритмов и сравнить их производительность.

7. Провести апробацию библиотеки в рамках мобильного приложения захвата и обработки кадров с оптического микроскопа.

2. Обзор

2.1. Цифровая обработка изображений в микроскопии

В данном разделе будут рассмотрены основные техники цифровой обработки захваченных с микроскопа изображений, их влияние на качество получаемого изображения и эффективность работы.

2.1.1. Фокус-стекинг

Для получения резкого изображения плоского препарата достаточно подобрать расстояние фокусировки, при котором всё изображение попадает в фокус. Однако в случае с объёмными препаратами данный способ не применим, так как глубина резкости оптического микроскопа мала из-за ограничений оптической системы. Сфокусированными получаются только определённые области препарата, что не позволяет рассмотреть объект целиком. Для решения этой проблемы применяется техника фокус-стекинга — по имеющемуся набору снимков препарата с различными зонами фокусировки формируется изображение, в котором все области попадают в фокус (рис.1).

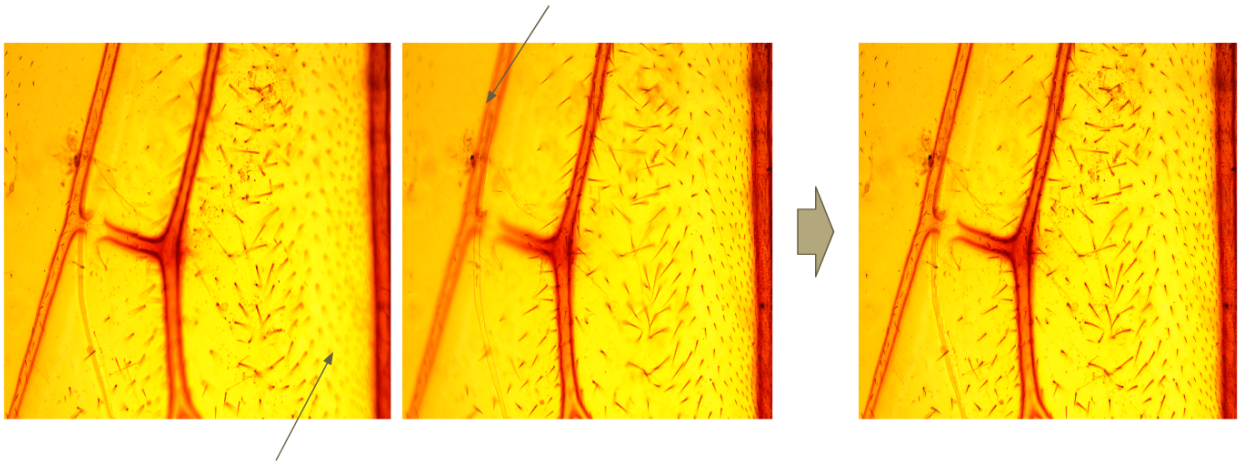


Рис. 1: Фокус-стекинг

На текущий момент алгоритмы фокус-стекинга можно разделить на три основные группы по принципу их действия [11] : pixel-based

approach, neighbour-based approach, transform-based approach. Формализуем данные подходы. Пусть $stack = \{im_i\}_{i=1}^{n}$ — набор изображений, для которых необходимо применить фокус-стекинг.

Pixel-based approach. Данный подход определяет, на каком изображении из входного набора находится самый сфокусированный пиксель с помощью фокусной меры. Выбранный пиксель переносится на изображение-результат. Пусть $P_{x,y}(im)$ — функция оценки того, насколько изображение im сфокусировано в пикселе (x, y) (фокусная мера). Тогда изображение, которое будет получено в результате, может быть записано в следующем виде.

$$result_{x,y} = stack_{x,y}[\arg \max_{im \in stack} P_{x,y}(im)]$$

Главной задачей в данном направлении является нахождение функции P. В большинстве работ в качестве фокусной меры широко используются операторы Лапласа или Собеля [11]. Данные алгоритмы достаточно эффективны, поскольку для этих операторов существуют отработанные машинно-зависимые оптимизации. Однако главным недостатком этого подхода является большая зашумленность изображений и появление разрывов на границах объектов — пиксели берутся с разных изображений и находятся вперемешку.

Neighbor-based approach. Данный подход решает проблему, указанную для pixel-based approach, однако, требует большее количество вычислительных ресурсов. В данном методе на первом шаге строится частичное изображение-результат, для которого значение фокусной меры P выше некоторого параметра $threshold$. Для оставшихся пикселей определяется, с какого изображения в окрестности рассматриваемого пикселя было взято больше всего информации. Затем в соответствии с этой информацией устанавливается значение рассматриваемого пикселя. Данный метод позволяет решить проблему зашумленности и разрывов, однако более ресурсозатратен.

Transform-based approach. Данный подход также основан на преобразовании изображений, однако в отличии от предыдущих подходов

сначала формирует промежуточное представление aux, а затем выполняет обратное преобразование:

$$aux_{x,y} = \max_{im \in stack} P_{x,y}(im)$$
$$result_{x,y} = P^{-1}_{x,y}(aux_{x,y})$$

В качестве преобразования P могут использоваться вещественные и комплексные вейвлет-преобразования [5]. Данный подход является наиболее ресурсозатратным, однако в большинстве случаев показывает стабильные результаты даже в случае с изначально зашумленными изображениями.

2.1.2. Выбор фокусной серии

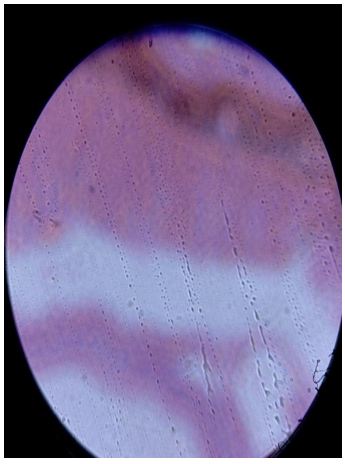
Для эффективного использования технологии фокус-стекинга, а также чтобы сократить время обработки набора кадров и уменьшить зашумлённость получаемого изображения, необходимо производить предварительный отбор кадров. Обычно в цифровой микроскопии отбор кадров производится вручную, что занимает долгое время. Поэтому автоматизация выбора фокусной серии является важной задачей в цифровой обработке изображений в микроскопии. Для уменьшения времени работы алгоритма фокус-стекинга необходимо выбрать минимальный стек изображений, в котором для каждой области исследуемого образца есть кадр, сфокусированный в этой области. Автоматизация данного процесса имеет ряд проблем — необходимо убрать одинаковые кадры (содержащие в фокусе одни и те же элементы препарата), выбрать кадры, включающие все зоны фокуса, присутствующие в исходном наборе изображений, удалить кадры с объектами, не относящимися к образцу, – пыль, конденсат и пр.

В рамках курсовой работы студента третьего курса программной инженерии Д. Яроша «Выбор фокусной серии в видеопотоке» предлагается алгоритм автоматического отбора кадров перед фокус-стекингом, обеспечивающий получение набора изображений из видео, захваченного с микроскопа с помощью камеры мобильного телефона. Данный

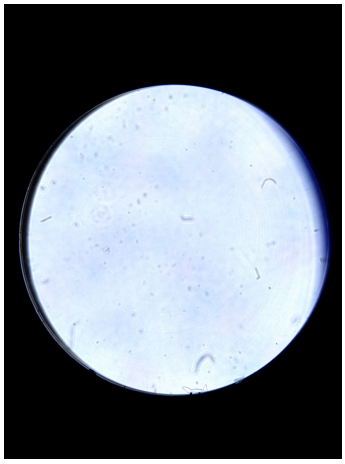
набор кадров является минимальным, сохраняет всю информацию о препарате и не содержит расфокусированных кадров.

2.1.3. Артефакты в оптической системе

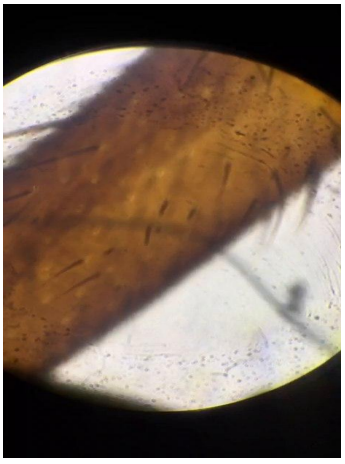
Частой проблемой мобильной микроскопии является наличие пыли в оптической системе, плохая подготовка препаратов для исследования, в результате чего на препаратах могут быть разводы и грязь. На рис. 2 приведены примеры изображений с наиболее часто встречающимися артефактами, которые могут негативно влиять на представление о рассматриваемом препарате: конденсат или грязь на препарате, кадр, в котором содержится только пыль (при автоматическом выборе кадров для фокус-стекинга такой кадр может попасть в стек и негативно повлиять на результат работы алгоритма), и пятна пыли поверх образца.



(a) Конденсат на препарате



(b) Кадр с пылью



(c) Кадр с пылью поверх образца

Рис. 2: Виды артефактов

Если первые два случая можно избежать, используя эвристики в алгоритме фильтрации кадров, как это показано в курсовой работе Дмитрия Яроша «Выбор фокусной серии в видеопотоке», то для решения третьей проблемы необходимо редактировать изображение. Главной задачей здесь является поиск маски пыли для дальнейшей «дорисовки» областей образца, скрытых за пылью.

Различные методы решения этой задачи рассмотрены в работе [28], в которой приводятся примеры фильтров, учитывающие особенности

грязи относительно образца (её форму и размер). Стоит отметить, что данные методы не устойчивы в случае, когда части препарата, например, клетки, по форме схожи с данными фильтрами.

Учитывая специфику предметной области и класса рассматриваемых микроскопов, была рассмотрена следующая идея. Выделение маски пыли возможно в тот момент, когда образец сильно расфокусирован, в таком случае изображение содержит только пыль на светлом фоне. Учитывая что пыль в оптической системе не смещается при вертикальном сканировании препарата, поиск маски пыли сводится к стандартному алгоритму адаптивной фильтрации (рис. 3). Затем возможно применение интерполяции пикселей, которые приходятся на маску, согласно значениям близлежащих пикселей вне маски (рис. 4). В рамках опроса, проведенного в рамках данной дипломной работы, было показано, что данный алгоритм дает положительную визуальную оценку, что подтверждает целесообразность его использования в разрабатываемой библиотеке.

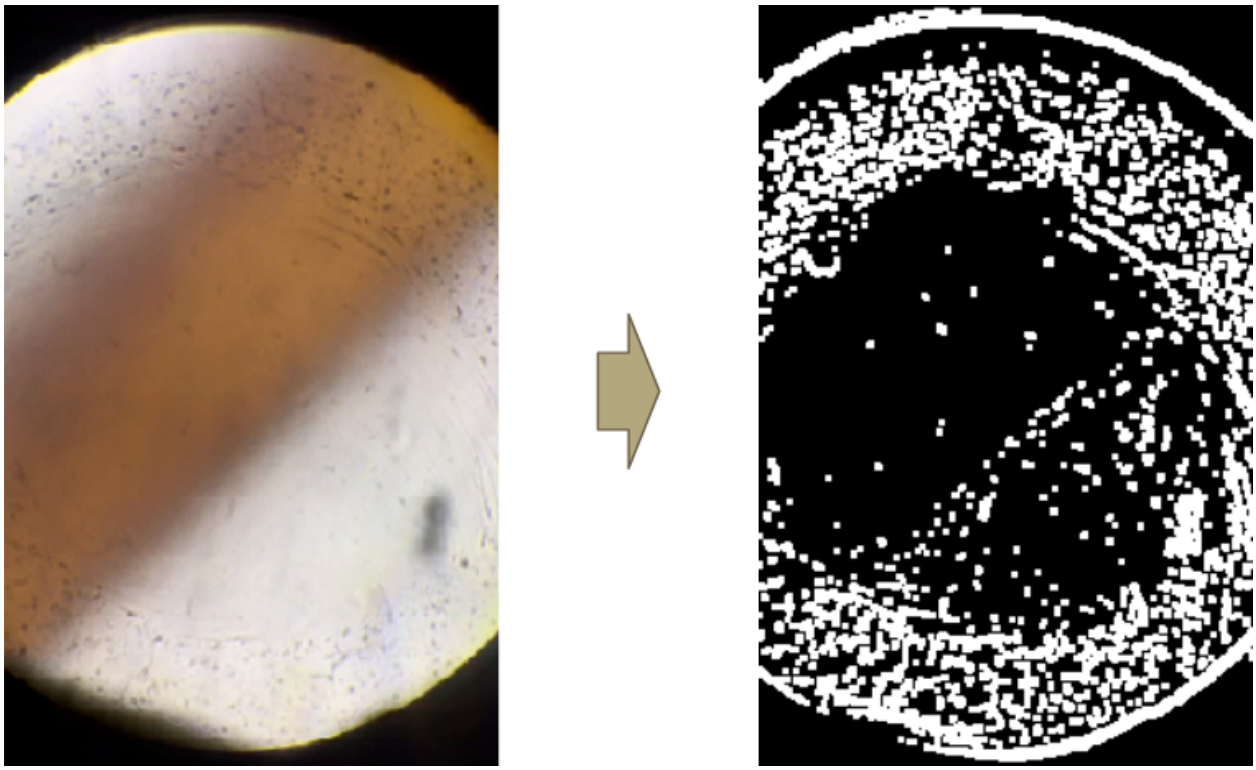


Рис. 3: Создание маски пыли по кадру

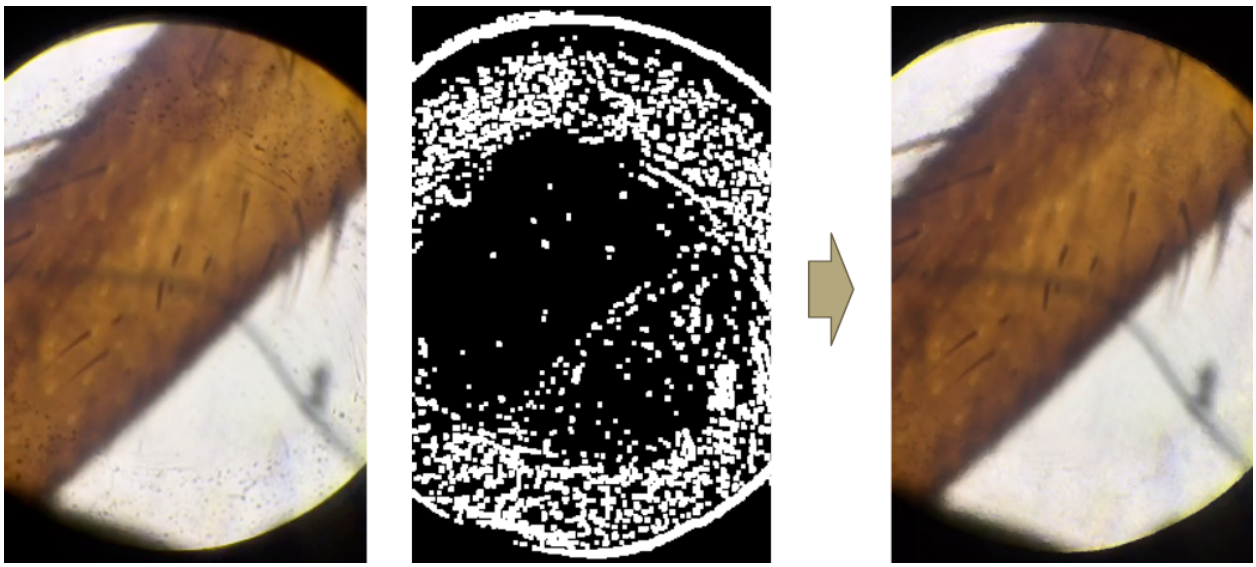


Рис. 4: Удаление пыли с изображения

2.1.4. Деконволюция

Рассеянный свет из областей, находящихся вне зоны фокуса, ниже или выше фокальной плоскости, становится причиной появления блеска, искажения и нерезкости при получении изображений. Деконволюция — способ обработки изображения, позволяющий избавиться от этих нежелательных эффектов. Данный способ, в отличие от общих способов улучшения изображения (повышение резкости, баланс белого, автокоррекция), требует знания параметров оптической системы, таких как размер пикселя, длина волны и размер относительного отверстия (numerical aperture). В основе данного подхода находится построение функции рассеивания точки (point spread function, PSF), по которой можно восстановить исходное изображение.

Одними из самых популярных open-source инструментов для построения PSF и деконволюции являются PSFGenerator [24] и DeconvolutionLab2 [23] от научно-исследовательской группы BIG университета EPFL [22]. Данные инструменты содержат большое количество известных на текущий момент алгоритмов в области деконволюции. На рис.5 представлен результат алгоритма деконволюции «Richardson–Lucy with Total Variation» [44], реализованный в рамках DeconvolutionLab2. Данный пример показывает, что деконволюция позволяет значительно

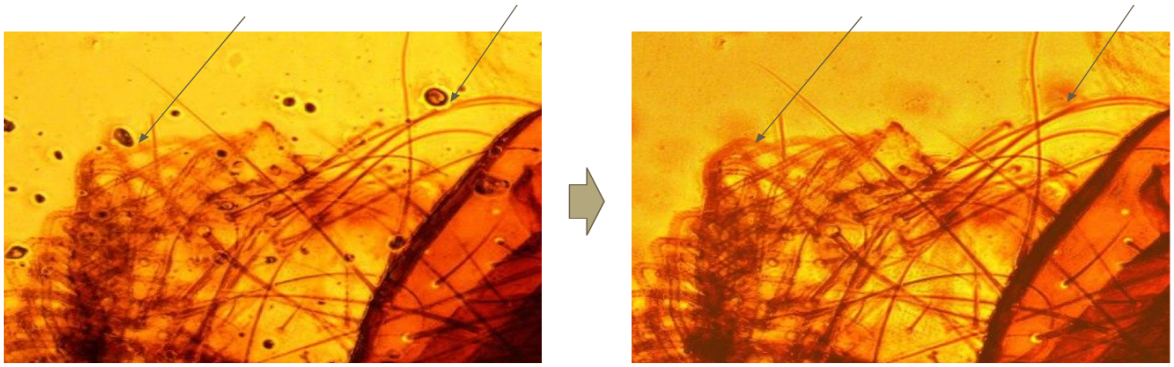


Рис. 5: Деконволюция

улучшить полученное изображение. Однако, стоит отметить, что деконволюция требует проведения калибровки оптической системы для вычисления ее свойств. Поскольку в данной работе рассматривается использование широкого спектра мобильных устройств для микроскопии, необходимо провести отдельное исследование методов калибровки для мобильных микроскопов, что выходит за рамки данной работы.

2.1.5. Полный процесс обработки изображения

Если объединить рассмотренные шаги цифровой обработки последовательности кадров, то получим следующий процесс (рис. 6).

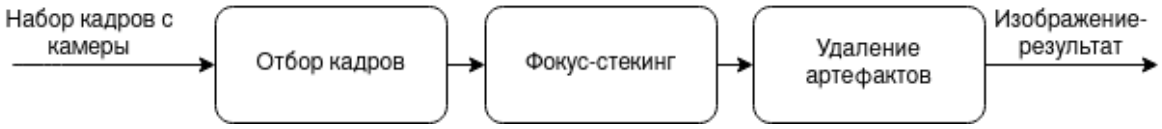


Рис. 6: Процесс обработки набора кадров

Первый и важный шаг — выбор кадров, которые несут в себе значимую информацию и могут быть полезны для исследования образца. Для этого необходимо убрать полностью нечеткие кадры, кадры, сфокусированные не на образце (например, пыль, конденсат), повторяющиеся кадры. Важной задачей на данном этапе является минимизация набора кадров. Следующий шаг — для каждого из этих кадров необходимо выполнить деконволюцию, что значительно повысит качество каждого кадра в отдельности. Процесс деконволюции выполняется для каждого

цветового канала в отдельности, после чего результаты нормализуются. За деконволюцией следует применение техники фокус-стекинга, которая объединяет все сфокусированные области на одном изображении.

Отдельным значительным шагом является применение других техник для улучшения общего качества изображения — удаление с изображения пыли и других различных артефактов, возникающих из-за возможного низкого качества оптической системы. Данные методы целесообразно применять после отбора кадров для экономии времени. Поскольку методы деконволюции и фокус-стекинга изменяют изображение, порядок, в котором необходимо применять методы удаления артефактов, зависит от влияния данного метода на результаты алгоритмов фокус-стекинга и деконволюции. Например, алгоритм удаления пыли рационально применить после фокус-стекинга к одному изображению, нежели к нескольким изображениям до фокус-стекинга, но для этого необходимо экспериментально убедиться, что алгоритм удаления пыли не влияет на качество фокус-стекинга.

Стоит отметить, что некоторые шаги из данного процесса могут быть пропущены (например, деконволюция), в зависимости от решаемых задач и требуемого качества. Предполагается, что разрабатываемое решение будет поддерживать различные вариации данного процесса.

2.2. Мобильные приложения

Для исследования предметной области было были рассмотрены мобильные приложения, связанные со съёмкой в микроскопии и макросъёмкой. Среди рассмотренных приложений было выделены три основные группы: инструменты для управления камерой или микроскопом, инструменты для макросъемки и фоторедакторы с возможностью объединять изображения.

2.2.1. Инструменты для управления камерой

AirLab [37] — приложение для Android и iOS для управления цифровыми микроскопами и камерами Leica Microsystems по Wi-Fi. Оно позволяет записывать фото и видео с камеры микроскопа, управлять увеличением камеры, яркостью получаемых изображений, зеркально отображать получаемые изображения по вертикали и горизонтали.

DinoDirect [8] и DinoConnect [7] — приложения для Android для управления портативными микроскопами Dino-Lite [6]. Эти приложения позволяют захватывать неподвижные изображения и видео, включать и выключать фонарик микроскопа, управлять экспозицией камеры микроскопа.

TinyCapture [40] — приложение для Android и iOS, позволяющее подключаться к цифровому микроскопу по Wi-Fi, управлять настройками его камеры, делать снимки и записывать видео.

Labscope Material [47] — приложение для iOS, предназначенное для управления цифровыми микроскопами. Оно позволяет подключиться к нескольким микроскопам по Wi-Fi, делать снимки и записывать видео, создавать аннотации и отчёты, делать сравнительные обзоры и производить измерения в реальном времени. Приложение также позволяет проводить обработку полученных изображений, но ни в описании приложения в магазине приложений App Store, ни на сайте разработчика не указано, в чём заключается обработка изображений. Помимо того, что данное приложение поддержано только под iOS, оно позволяет работать только с микроскопами, произведёнными компанией Carl Zeiss [48].

2.2.2. Инструменты для съёмки

HedgeCam 2 [26] — приложение камеры, позволяющее снимать фото и видео, выполнять настройку камеры. Один из режимов съёмки заключается в получении серии снимков с разным фокусным расстоянием камеры, однако снятые кадры никак не обрабатываются.

Magnifier Camera [32] — приложение камеры для Android и iOS.

Оно позволяет управлять оптическим увеличением камеры, делать снимки, делать стоп-кадр экрана, чтобы увидеть изображение стабильно. Обработки полученных изображений не происходит.

Cozy Magnifier & Microscope [25] — приложение камеры для Android. Обладает схожей с Magnifier Camera функциональностью, однако также позволяет управлять цифровым увеличением изображения.

2.2.3. Редакторы фото

Snapseed [20] — растровый графический редактор для обработки фотографии на Android и iOS устройствах. Он предоставляет множество инструментов для редактирования изображений, неспецифичных для микроскопии. Инструмент «двойная экспозиция» позволяет выполнять фокус-стекинг. Однако для этого необходимо иметь набор кадров в виде изображений, вручную добавлять каждый кадр, выравнивать его, рисовать маску, по которой определяется, какая часть этого кадра должна попасть на результирующее изображение.

Adobe Photoshop Mix [41] — фоторедактор, позволяющий изменять цвета, вырезать и объединять изображения. Для объединения кадров также необходимо иметь их в виде изображений и выравнивать их вручную. Объединение кадров происходит автоматически с использованием одного из доступных режимов. Однако эти режимы не используют алгоритмы фокус-стекинга.

Pixl [33] — фоторедактор для Android и iOS. Позволяет аналогично Adobe Photoshop Mix выполнять объединение нескольких изображений, для этого применяются те же режимы объединения, что и в Adobe Photoshop Mix.

2.2.4. Результаты обзора

Большинство из рассмотренных приложений предназначены только для захвата кадров с различными параметрами съемки. Основные недостатки приложений, выполняющих цифровую обработку кадров: приложения либо работают с конкретным производителем микроско-

пов, либо существует только для одной операционной системы, либо требуют ручных манипуляций от пользователя, например, ручное выравнивание кадров для алгоритмов фокус-стекинга.

2.3. Средства создания и использования кросс-платформенных библиотек и приложений

В рамках работы были сформированы два основных требования к библиотеке цифровой обработки изображений в микроскопии.

- Кросс-платформенность. В библиотеке должна быть поддержана возможность быстрой интеграции с популярными ОС для смартфонов (Android, iOS).
- Средства отладки на Desktop-платформе. Для ускорения процесса разработки и повышения качества разрабатываемого решения, должна быть поддержана возможность отладки и анализа библиотеки на ПК.

Следуя данным требованиям, в текущем разделе рассматриваются возможные средства разработки кросс-платформенных мобильных приложений и библиотек, которые позволяют быстро прототипировать мобильные приложения с использованием разрабатываемой библиотеки, хорошо документированны. Также важно, чтобы для выбранной технологии существовали инструменты разработки.

2.3.1. Xamarin

Xamarin [36] — это кросс-платформенная бесплатная платформа с открытым исходным кодом для создания приложений для Android и iOS с использованием .NET. С помощью Xamarin можно создавать собственный пользовательский интерфейс для каждой платформы и писать на языке C# общую бизнес-логику, которая будет использоваться на различных платформах.

Спецификация .NET Standard позволяет создавать библиотеки, которые могут использоваться во всех реализациях .NET, в том числе Xamarin.iOS и Xamarin.Android. Чтобы использовать такую библиотеку в нативном Android или iOS приложении, необходимо встроить в него .NET. Для этого можно воспользоваться инструментом Embeddinator-4000 [42].

2.3.2. React Native

React Native [29] — это фреймворк для разработки кросс-платформенных приложений для iOS и Android. Для разработки в нём используется язык JavaScript. React Native является аналогом JavaScript-фреймворка React.JS для нативной разработки. Компоненты, разработанные на React Native, могут быть внедрены в нативные приложения на iOS и Android [30].

Библиотеки, написанные на JavaScript могут быть использованы в различных JavaScript-фреймворках, также существует возможность исполнять их код внутри нативных Android и iOS приложений, используя LiquidCore [35].

2.3.3. Flutter

Flutter [17] — разрабатываемый Google набор средств разработки приложений для мобильных устройств, персональных компьютеров и web-сайтов используя единую кодовую базу. Flutter-приложения реализуются на языке Dart и выполняются на виртуальной машине. Разработанный на Flutter код может быть встроен в код нативного приложения на Android и iOS [13].

2.3.4. C/C++

Скомпилированную под целевую архитектуру библиотеку, написанную на C/C++, можно использовать как в рассмотренных фреймворках для создания кросс-платформенных приложений, так и в нативных

Android и iOS-приложениях. Рассмотренные фреймворки предоставляют собственные методы взаимодействия с C/C++ кодом.

Для взаимодействия кода нативного Android-приложения и C/C++ кода используется JNI [39].

Objective-C является надмножеством языка C, поэтому из кода на Objective-C можно вызывать функции, реализованные на C. Код на C++ может быть вызван только из кода, реализованного на Objective-C++. Поэтому для вызова C++ кода из Objective-C и Swift необходимо необходимо реализовать прослойку на Objective-C++, непосредственно обращающуюся к C++ коду.

Для решения данной проблему существуют инструменты, которые автоматически генерируют связующий код.

- SWIG [45] — инструмент разработки программного обеспечения, который помогает обеспечивать взаимодействие программ и библиотек, написанных на C и C++, с различными языками программирования: C#, Java, Go, JavaScript, Python и другими. Генерация кода происходит на основе файла интерфейса, который может содержать прототипы функций, объявления переменных и подключать существующие заголовочные файлы. SWIG поддерживает генерацию для Java, в результате генерации создаётся JNI-прослойка и Java-классы, которые могут использоваться в Android приложении. Однако у SWIG нет функциональности для генерации Objective-C++ прослойки, поэтому, чтобы вызывать C++ код из Objective-C и Swift, необходимо написать такую прослойку вручную.

- Djinni [9] — инструмент для генерации межъязыковых объявлений типов и привязок интерфейса. Он предназначен для соединения C++ с Java и Objective-C. Djinni генерирует код на основе описаний типов, которые содержатся в библиотеке и могут быть использованы в коде на Java и Objective-C. Файл описания типов может содержать три вида объявлений: перечисления, записи и интерфейсы. По описанию интерфейсов генерируется аб-

страктный C++ класс (класс с чисто виртуальными методами), для которого необходимо написать реализацию, Objective-C++ и JNI-обёртки вызовов библиотеки, а также классы, доступные в Java/Kotlin и Objective-C/Swift.

3. Реализация кросс-платформенной библиотеки

3.1. Выбор инструментария

Функциональность библиотеки предполагает обработку изображений с использованием численных алгоритмов, поэтому было принято решение для реализации некоторых шагов алгоритмов обработки стека изображений воспользоваться кросс-платформеннойной библиотекой OpenCV [31]. Это позволяет получить высокую производительность на большинстве смартфонов за счет присутствующих в OpenCV архитектурных оптимизаций для обработки изображений.

Разработка мобильных приложений возможна как с использованием API целевых платформ, так и используя разные фреймворки кросс-платформенной разработки мобильных приложений. В рамках данной работы было принято решение разрабатывать библиотеку на C++, потому что такая библиотека может использоваться в обоих случаях, а также OpenCV предоставляет API для C++. Для прототипирования мобильных приложений с использованием разрабатываемой библиотеки необходимо при её изменении обновлять прослойки вызова библиотеки из нативного кода. Для этого был выбран инструмент Djinni, позволяющий генерировать связующий код для использования библиотеки из Objective-C и Swift.

3.2. Процесс сборки библиотеки

Важным шагом прототипирования мобильных приложений является процесс сборки кода библиотеки под нужные платформы. Широкое разнообразие мобильных устройств требует использование разных компиляторов. Для автоматизации этого процесса было принято решение воспользоваться Polly [2] — коллекцией скриптов и CMake-файлов для различных инструментов сборки.

3.2.1. Сборка для Desktop-платформы

1. С помощью набора инструментов Polly — «gcc-cxx17-c11» компилируются файлы исходного кода OpenCV и разрабатываемой библиотеки.

2. Выполняется компоновка динамической библиотеки.

3.2.2. Сборка для операционной системы Android

1. С помощью следующих наборов инструментов Polly компилируются файлы исходного кода OpenCV, разрабатываемой библиотеки и сгенерированные с помощью Djinni файлы исходного кода, обеспечивающего связь Java и C++:

   - «android-ndk-r20-api-21-arm64-v8a-neon-clang-libcxx17»,
   - «android-ndk-r20-api-19-armeabi-v7a-neon-clang-libcxx»,
   - «android-ndk-r20-api-21-x86-64-clang-libcxx17»,
   - «android-ndk-r20-api-19-x86-clang-libcxx».

2. Выполняется компоновка динамической библиотеки.

3. Из полученной динамической библиотеки и Java-кода, сгенерированного Djinni с помощью системы автоматической сборки Gradle [21] происходит сборка Android-библиотеки (AAR [15]).

3.3. Архитектура библиотеки

Архитектура библиотеки разрабатывалась в соответствие с требованием возможности легко добавлять новые алгоритмы для имеющихся шагов обработки стека изображений, новые шаги обработки. Класс VideoProcessor (рис. 7) принимает в конструкторе набор опций, описывающий, какие шаги должны быть в обработке и какой алгоритм необходимо использовать на каждом шаге. Он покадрово обрабатывает видео, применяет к кадрам алгоритм удаления пыли и производит их отбор. Метод GetResult выполняет фильтрацию отобранных

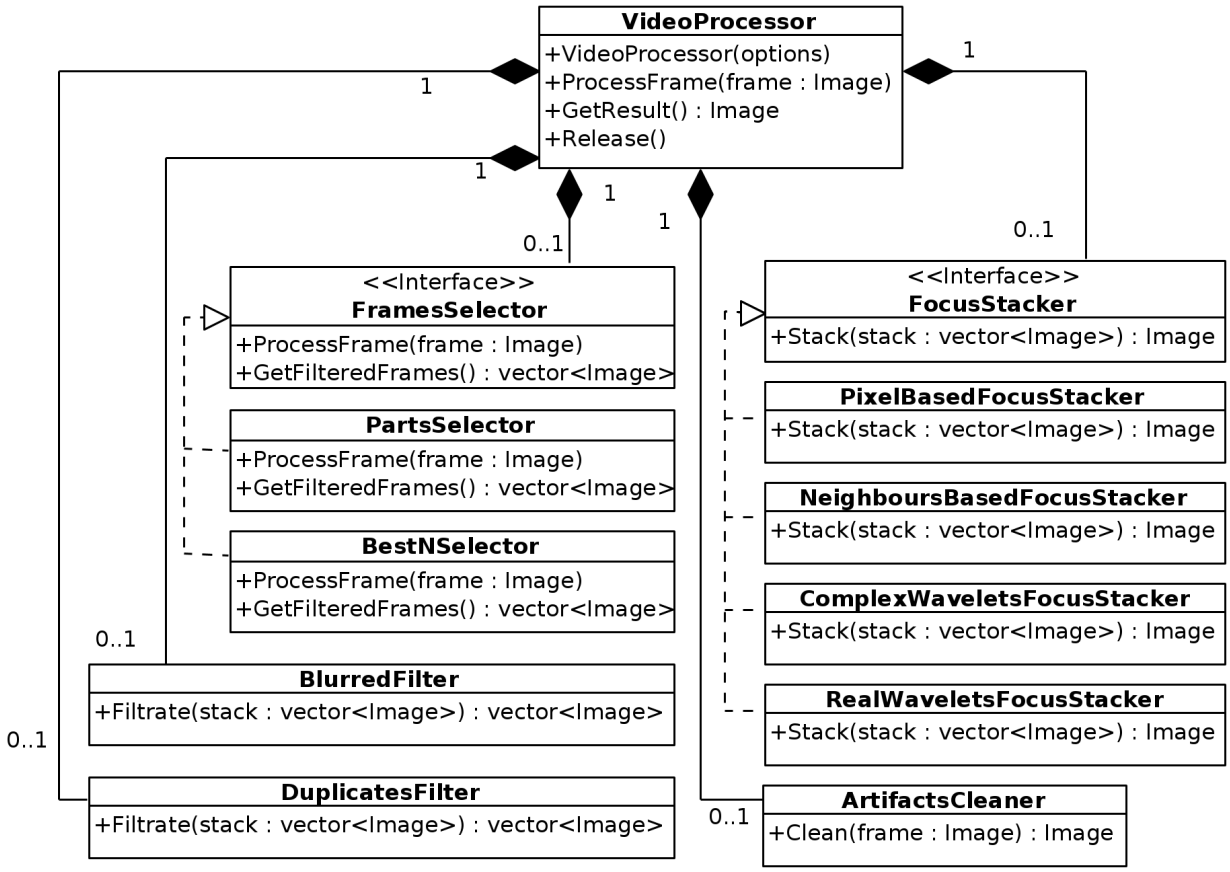


Рис. 7: Диаграмма классов

изображений, фокус-стекинг полученного набора кадров и возвращает изображение-результат. Метод Release освобождает все выделенные ресурсы.

Такая архитектура позволяет тестировать как отдельно выбранный алгоритм фокус-стекинга, так и различные комбинации с алгоритмами отбора кадров и улучшения их качества.

4. Система визуального сравнения кадров и проведение опроса

Одной из важных частей работы является визуальная оценка работы алгоритмов. Отсутствие существенных отличий в результатах работы нескольких алгоритмов позволит сфокусироваться на более быстрых. Специфика предметной области требует сравнения изображений на различных видах устройств, поэтому отдельным значимым шагом является поиск решения, которое позволит сравнить качество работы алгоритмов на мобильных устройствах. В данном разделе представлены требования к системе визуального сравнения, детали реализации собственной кросс-платформенной системы визуального сравнения и результаты проведения опроса.

4.1. Требования к системе визуального сравнения

В рамках работы были сформированы требования к системе визуального сравнения. Как уже было сказано, основным требованием является поддержка мобильных платформ. Помимо этого были сформированы следующие требования.

- Выбор опрашиваемого должен запоминаться автоматически, а также должна собираться статистика по алгоритмам для удобной обработки результатов массового сравнения.
- Переключение должно осуществляться между любыми из сравниваемых изображений, а не только между соседними.
- Необходимо поддержать сохранение увеличения и сдвига при переключении между изображениями. Это позволяет не терять из вида рассматриваемую область и сразу видеть отличия двух изображений в ней.
- Интерфейс мобильной версии должен быть разработан так, чтобы при переключении между двумя изображениями палец не пере-

крывал изображение, что помешало бы увидеть, в каких областях они отличаются.

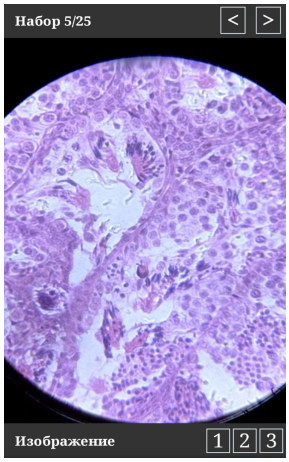
- Система должна поддерживать простую замену набора изображений для сравнения, поскольку планируется использовать её для различных технологий цифровой обработки изображений.

Рис. 8: Desktop-интерфейс

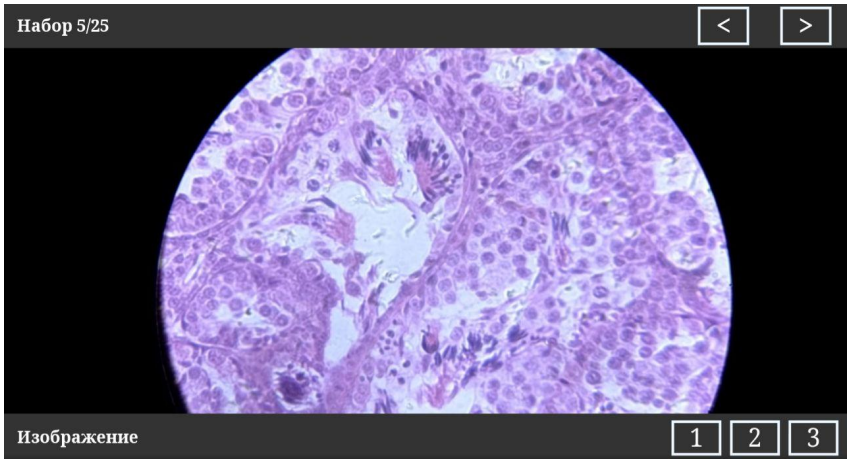
4.2. Реализация системы сравнения

Для достижения данных целей было разработано одностраничное web-приложение, что позволяет просто решить проблему поддержки мобильных устройств. Для каждого набора кадров опрашиваемому предлагается сравнить результаты работы нескольких алгоритмов и выбрать наиболее понравившийся. На рис. 8 и рис. 9 показан интерфейс для разных платформ.

Для отображения изображения используется html-тэг <canvas> и скрипт ImjTouchCanvas.js [3], который добавляет возможность увеличивать и уменьшать изображение, перетаскивать его, используя клавиатуру и мышь на компьютере, и традиционные для данных операций



(a) Портретная ориентация



(b) Альбомная ориентация

Рис. 9: Мобильный интерфейс

жесты на смартфоне. Также использование этого скрипта позволяет заменить изображение, сохраняя увеличение и сдвиг. Однако у этого скрипта был обнаружен недостаток — перерисовка изображения вызывалась с максимально возможной частотой, даже если видимая часть изображения не менялась. Поэтому в этот скрипт были внесены изменения, чтобы запрашивать перерисовку изображения только в случаях, когда пользователь сдвигал изображение или изменял его размеры. При прохождении опроса система запоминает выбранные изображения и формирует статистику.

Для сбора статистики было принято решение использовать Google Forms [19], как удобный инструмент для массового анализа результатов. Для взаимодействия с Google Forms необходимо создать в них опрос. Для отправки результатов в системе визуального сравнения генерируется URI по следующему шаблону `https://docs.google.com/forms/d/e/FORMID/formResponse?&[entry.NUMBER=VALUE]&submit=SUBMIT`, где FORMID — идентификатор формы, для каждого поля ввода формы: NUMBER — числовой идентификатор поля, VALUE — значение, передающееся в соответствующее поле. После этого система визуального сравнения выполняет POST запрос к сформированному URI. Идентификатор формы можно узнать из URI web-страницы ответов на форму, а идентификаторы полей — из исходного кода этой страницы.

4.3. Проведение опроса

С помощью разработанной системы визуального сравнения был проведен опрос. Система визуального сравнения была развернута с использованием сервиса GitHub Pages [12]. Опрос содержал в себе три категории наборов изображений.

- Результаты работы трех основных алгоритмов фокус-стекинга: pixel-based, neighbor-based, transform-based (в рамках библиотеки рассматривался подход, основанный на целочисленных вейвлетах) для сравнения качества работы алгоритмов. Для сравнения результатов работы алгоритмов были подобраны образцы с разным размером стека, что также может оказывать влияние на результаты работы алгоритмов. Всего было рассмотрено 15 наборов изображений: 1-5 — с размером стека 2, 6-10 — с размером стека 3, 10-15 — с размером стека 4.

- Кадр из стека и кадр-результат работы алгоритма фокус-стекинга для определения того, может ли алгоритм фокус-стекинга негативно влиять на изображение результат. Всего было рассмотрено 5 наборов изображений.

- Кадр с удалением пыли и без удаления пыли для определения того как данная операция влияет на результат. Всего было рассмотрено 5 наборов изображений.

В опросе приняло участие 85 респондентов. Среди них 72 человека прошло опрос с мобильного устройства, 13 – с компьютера.

На рис. 10 и 11 продемонстрированы результаты опроса по первой категории — алгоритмам фокус-стекинга. По результатам опроса на мобильной платформе алгоритм, основанный на пиксельных преобразованиях, был выбран чаще всего. Однако, стоит заметить, что на препаратах с большим количеством кадров в стеке, это лидерство неоднозначно, что должно быть принято во внимание при использовании библиотеки. Несмотря на небольшое количество респондентов, отвечающих с компьютера, опрос показал, что алгоритм, основанный на

вейвлетных преобразованиях, показывает самые стабильные результаты, что говорит о его высоком качестве, т.е. в случае использовании мобильной библиотеки для диагностических целей, когда необходимо увеличить изображение для изучения – данный алгоритм может быть самым уместным.

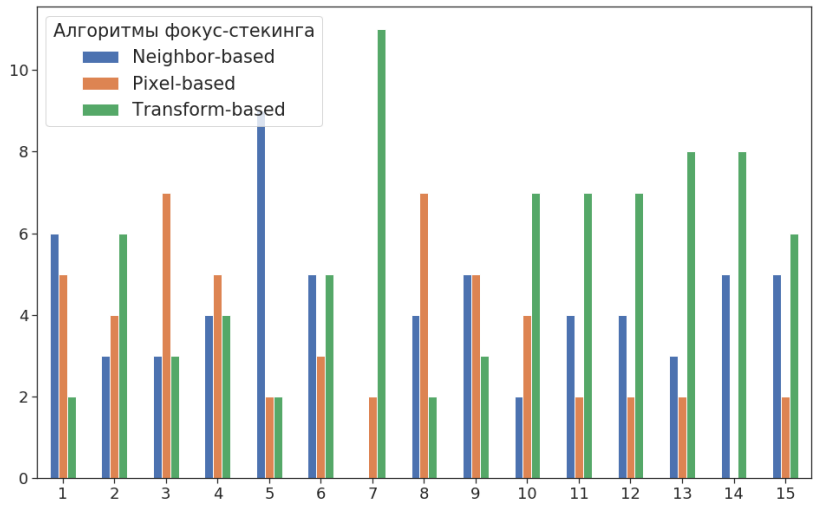


Рис. 10: Сравнение алгоритмов фокус-стекинга на Desktop-платформе

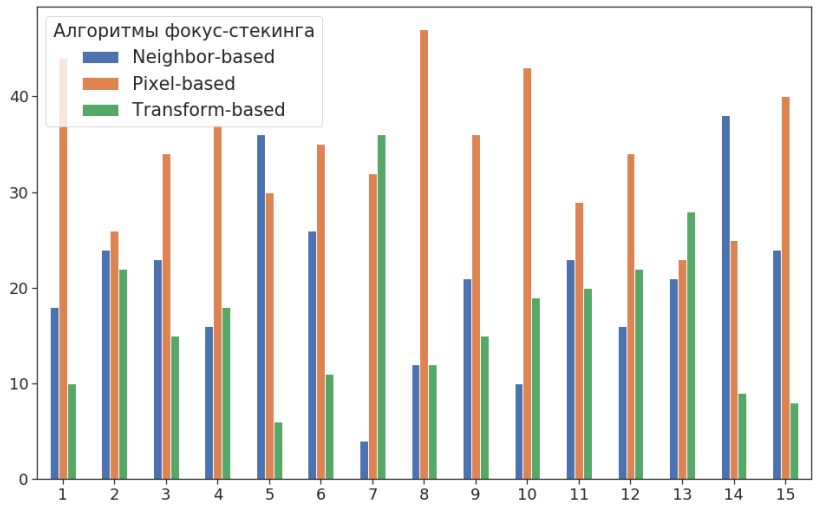
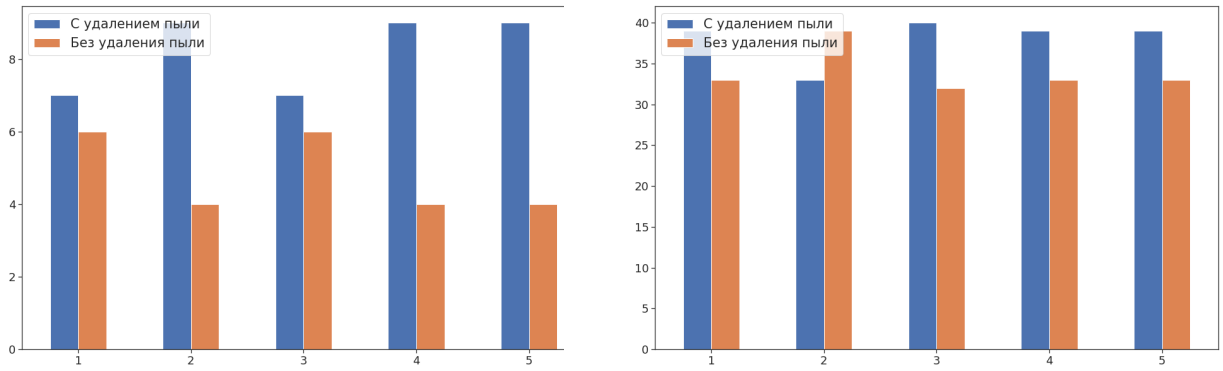


Рис. 11: Сравнение алгоритмов фокус-стекинга на мобильной платформе

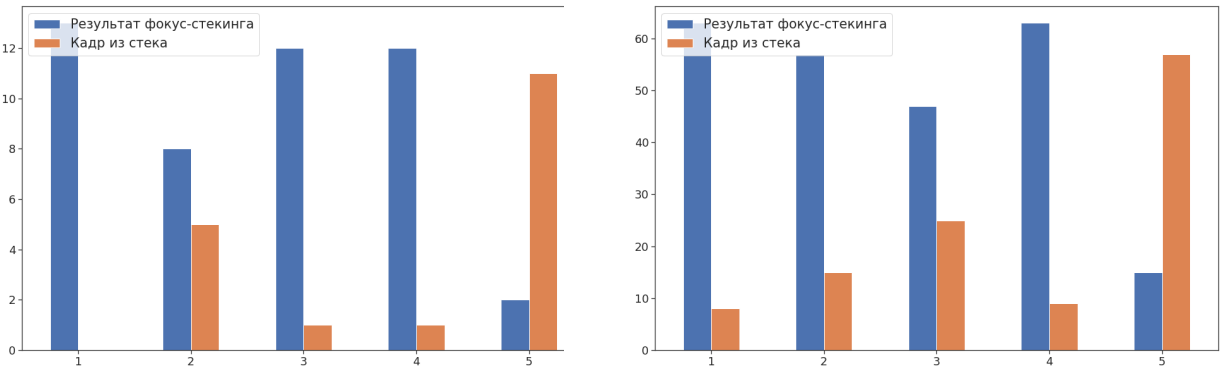
Рис. 12a и 12b демонстрирует, что, на взгляд респондентов, кадры

с убиранием пыли смотрятся лучше нежели без него. При включении данного алгоритма в общий процесс обработки изображений следует оценивать также его быстродействие, либо предоставлять пользователю выбор.



(a) Desktop-платформа (b) Мобильная платформа

Рис. 12: Сравнение результата фокус-стекинга с убиранием пыли и без



(a) Desktop-платформа (b) Мобильная платформа

Рис. 13: Сравнение кадра из стека и результата фокус-стекинга

На рис. 13a и 13b показаны результаты опроса для оценки влияния алгоритмов фокус-стекинга. На четырёх образцах из пяти применение алгоритма фокус-стекинга позволяет качественно улучшить изображение на обеих платформах – мобильные устройства и компьютеры. Стоит заметить, что на последнем наборе обычный кадр из стека показал результат лучше чем алгоритм фокус-стекинга. В данном случае

алгоритм фокус-стекинга оказался избыточным, поскольку рассматриваемый препарат оказался достаточно плоским.

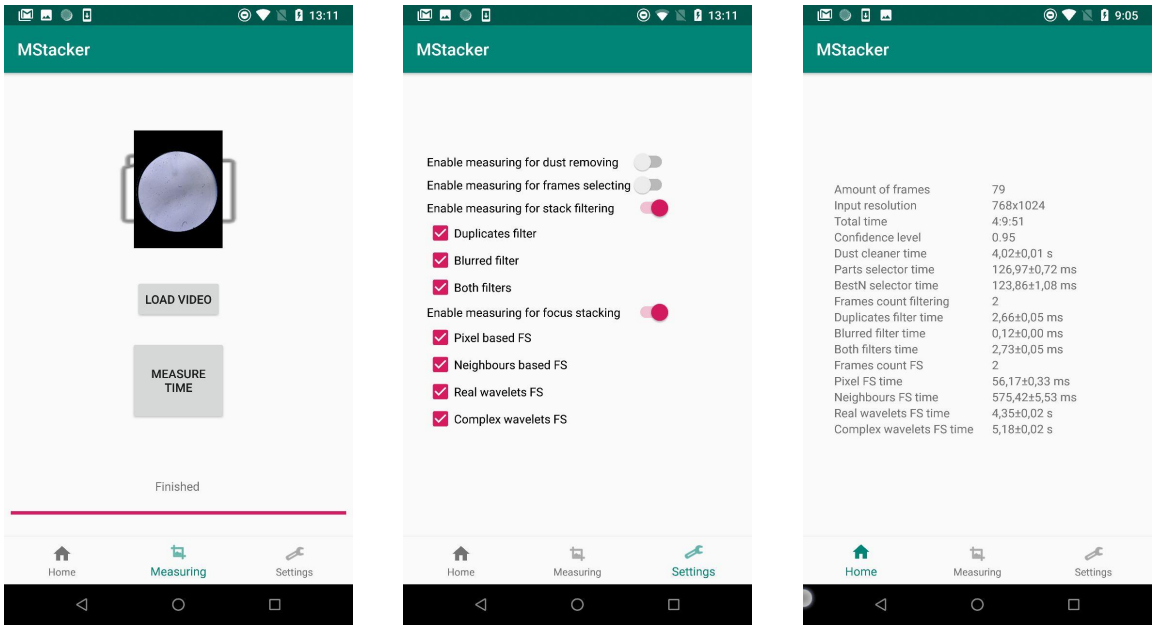
5. Прототип для замера скорости работы алгоритмов

5.1. Детали реализации

Для проведения замеров производительности реализованных в библиотеке алгоритмов было реализовано Android-приложение. Приложение разработано на языке Kotlin с использованием 19 версии Android API. Эту версия API поддерживает 96.2% устройств, работающих на Android [16].

Приложение позволяет:

- загружать видео с устройства или из облачного хранилища Google Drive [18];
- выбирать, время работы каких алгоритмов должны быть измерено;
- замерять время работы выбранных алгоритмов.



(a) Выбор видео (b) Выбор алгоритмов (c) Статистика замеров

Рис. 14: Прототип мобильного приложения для замеров производительности алгоритмов

На рис. 14 продемонстрированы основные окна приложения: окно выбора видео, на котором проводятся замеры, выбор алгоритмов, участвующих в замерах, и окно с результатами статистической оценки.

5.2. Проведение замеров производительности

Замеры времени работы алгоритмов проводились на смартфоне Nexus 6P, оснащённом центральным процессором Qualcomm Snapdragon 810 MSM8994, 3 ГиБ оперативной памяти, с установленной операционной системой Android версии 8.1.0.

Стоит отметить, что алгоритмы удаления пыли и отбора кадров подразумевают покадровую обработку, а алгоритмы фокус-стекинга и фильтрации отобранных кадров применяются к стеку изображений. Поэтому для алгоритмов удаления пыли и отбора кадров замерялось время обработки одного кадра, а для алгоритмов фильтрации стека и фокус стекинга — время обработки стека размером 2, 3, 4 и 5 изображений.

Основные рассмотренные алгоритмы для сравнения.

- Алгоритм удаления пыли.
- Алгоритмы отбора сфокусированных кадров: Best3 — алгоритм отбора 3 лучших кадров по фокусной мере и Parts — алгоритм отбора, применяющий фокусную меру к частям изображения, который был предложен в рамках курсовой работы Дмитрия Яроша.
- Алгоритм удаления дубликатов и размытых кадров.
- Алгоритмы фокус-стекинга: pixel-based, neighbor-based, transform-based с различным размером стека — 2, 3, 4 и 5 кадров.

В таблицах 1, 2, 3, 4, 5 представлены доверительные интервалы времени работы алгоритмов с доверительной вероятностью $p = 0,95$. Для их построения каждый алгоритм на каждом видео был запущен по 1000 раз. Исключение составляют алгоритм удаления пыли и transform-based алгоритм фокус-стекинга на видео с разрешением 1080×1920. В

силу долгой работы алгоритмов было принято решение снизить число запусков до 100, так как для понимания соотношения времени работы алгоритмов полученной точности измерений достаточно.

	Разрешение видео	Удаление пыли, с.	Отбор кадров	
			Parts, мс.	Best3, мс.
1	464×848	$1,32 \pm 0,01$	$56,25 \pm 0,5$	$36,73 \pm 0,1$
2	768×1024	$4,02 \pm 0,01$	$156,41 \pm 1,5$	$126,97 \pm 1,1$
3	1080×1920	$67,16 \pm 0,21$	$615,90 \pm 7,6$	$472,87 \pm 3,4$

Таблица 1: Результаты замеров времени удаления пыли и отбора кадров

	Размер стека — 2	Размер стека — 3	Размер стека — 4	Размер стека — 5
1	$1,2 \pm 0,1$	$2,0 \pm 0,1$	$2,9 \pm 0,1$	$3,9 \pm 0,1$
2	$2,7 \pm 0,1$	$3,5 \pm 0,1$	$5,2 \pm 0,1$	$6,6 \pm 0,1$
3	$7,0 \pm 0,1$	$11,4 \pm 0,1$	$17,0 \pm 0,1$	$22,3 \pm 0,2$

Таблица 2: Результаты замеров времени удаления дубликатов и размытых кадров на различных стеках (мс.)

	Размер стека — 2	Размер стека — 3	Размер стека — 4	Размер стека — 5
1	$30,0 \pm 0,2$	$42,81 \pm 0,4$	$49,9 \pm 1,0$	$63,1 \pm 1,4$
2	$56,2 \pm 0,3$	$78,8 \pm 0,5$	$112,7 \pm 1,1$	$128,2 \pm 2,9$
3	$150,6 \pm 2,2$	$222,7 \pm 3,6$	$286,8 \pm 3,7$	$415,5 \pm 8,6$

Таблица 3: Результаты замеров времени pixel-based алгоритма фокус-стекинга на различных стеках (мс.)

	Размер стека — 2	Размер стека — 3	Размер стека — 4	Размер стека — 5
1	$355,9 \pm 3,9$	$340,1 \pm 9,3$	$363,4 \pm 4,6$	$457,9 \pm 11,8$
2	$575,4 \pm 5,5$	$773,0 \pm 4,7$	$853,2 \pm 8,6$	$870,7 \pm 27,6$
3	$1877,9 \pm 20,1$	$2544,1 \pm 27,1$	$2952,8 \pm 32,2$	$3487,0 \pm 44,8$

Таблица 4: Результаты замеров времени neighbor-based алгоритма фокус-стекинга на различных стеках (мс.)

	Размер стека — 2	Размер стека — 3	Размер стека — 4	Размер стека — 5
1	$2,5 \pm 0,1$	$3,2 \pm 0,1$	$3,8 \pm 0,1$	$4,7 \pm 0,1$
2	$4,4 \pm 0,1$	$6,1 \pm 0,1$	$9,0 \pm 0,1$	$9,6 \pm 0,2$
3	$24,2 \pm 0,2$	$24,3 \pm 0,3$	$30,5 \pm 0,1$	$37,4 \pm 0,2$

Таблица 5: Результаты замеров времени transform-based алгоритма фокус-стекинга на различных стеках (с.)

На основании проведённого опроса о качестве работы алгоритмов и замерах времени их работы можно сделать следующие выводы.

- Применение техники фокус-стекинга позволяет улучшить качество изображения, однако для пользователей, использующих мобильные устройства, разница между результатами работы разных алгоритмов малозаметна, поэтому осмысленно использовать алгоритм фокус-стекинга, основанный на пиксельных преобразованиях, как самый быстрый. В случае когда необходимо получать более качественные изображения, например, для диагностики, рекомендуется использовать алгоритм, основанный на вейвлетных преобразованиях.

- Среди алгоритмов фильтрации кадров наиболее быстрым оказался Best3, однако алгоритм Parts использует более сложные эвристики для выбора кадров и позволяет добиться стека изображений, который дает более качественные результаты в алгоритмах фокус-стекинга.

- Фильтрация отобранных кадров — удаление дубликатов и размытых кадров — очень быстрые операции, которые при этом значительно уменьшают стек и, тем самым, гарантируют более качественный результат фокус-стекинга.

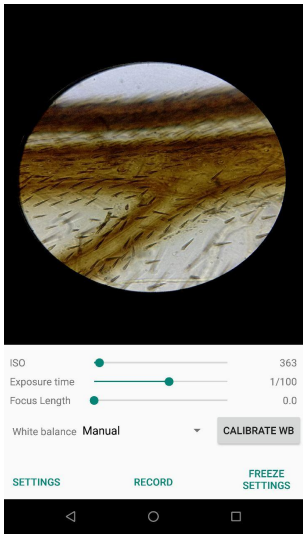
- Удаление пыли — долгая операция, поэтому применять её к каждому кадру не имеет смысла, так как время обработки видео увеличивается очень сильно. Однако результаты опроса показывают,что данная операция позволяет повысить качество получаемых изображений, поэтому разумным видится применения этой операции к результату фокус-стекинга.

6. Апробация библиотеки на ОС Android

Для апробации реализованной библиотеки было реализовано Android-приложение, позволяющее управлять цифровым микроскопом, записывать видео и выполнять его обработку. Приложение позволяет выполнять настройку экспозиции и светочувствительности камеры, для этого используется Camera2 API [14]. Интерфейс приложения представлен на рис. 15.

Для апробации библиотеки компанией MEL Science [34] был предоставлен цифровой микроскоп, который поддерживает управление предметным столиком по Bluetooth с использованием внешних двигателей. В рамках приложения были поддержаны 4 режима движения предметного столика с разными скоростями.

Записанное видео и результат обработки загружаются в облачное хранилище Google Drive. Для реализованных в библиотеке алгоритмов фокус-стекинга была проведена верификация результата. Получившиеся в результате работы изображения визуально-эквивалентны результатам прототипов алгоритмов. С помощью данного приложения были записаны видео со 150 микропрепаратами, которые были использованы для создания опроса и проведения замеров времени работы реализованных в библиотеке алгоритмов.



(a) Запись видео

(b) Настройки приложения

Рис. 15: Android-приложение для записи и обработки видео

Заключение

В ходе работы над проектом были достигнуты следующие результаты.

1. Выполнен обзор методов цифровой обработки изображений в микроскопии, были выделены алгоритмы, которые нужно реализовать в кросс-платформенной библиотеке.

2. Выполнен обзор и сравнение мобильных приложений для фотосъемки в микроскопии и макросъемке.

3. Выполнен обзор инструментов для создания кросс-платформенных библиотек для мобильных устройств.

4. Реализована кросс-платформенная мобильная библиотека для цифровой обработки изображений в микроскопии с использованием выбранных инструментов.

5. Реализована система для визуального сравнения изображений, проведён опрос для сравнения качества работы алгоритмов, реализованных в библиотеке.

6. Реализован прототип для замера скорости работы алгоритмов и проведено сравнение их производительности.

7. Проведена апробация библиотеки в рамках Android-приложения захвата и обработки кадров с оптического микроскопа.

Благодарности

Автор выражает отдельную признательность коллегам, помогавшим ему при создании данной работы:

- Корниловой Анастасии Валерьевне — за уточнение постановки задач и регулярный контроль их неукоснительного выполнения, за огромную помощь в структурировании текста, организации опроса и обработке его результатов;
- Ярошу Дмитрию Сергеевичу — за предложенный алгоритм отбора кадров и другую работу, выполненную в рамках его курсовой;
- Кириленко Якову Александровичу — за знания по C++, полученные при рецензировании кода.

А также благодарит компанию MEL Science за предоставленное оборудование и лично Павла Михайловича Катунина, согласившегося выступить рецензентом.

Список литературы

[1] Adobe. Adobe Photoshop // официальный сайт. — URL: https://www.adobe.com/ru/products/photoshop.html (online; accessed: 20.11.2019).

[2] Baratov Ruslan. Polly // официальный репозиторий. — URL: https://github.com/ruslo/polly (online; accessed: 14.12.2019).

[3] Beaudon Romain. ImgTouchCanvas // репозиторий. — URL: https://github.com/rombdn/img-touch-canvas (online; accessed: 22.05.2020).

[4] Breslauer DN Maamari RN Switz NA Lam WA Fletcher DA. Mobile phone based clinical microscopy for global health applications // PLoS One. — 2009. — Vol. 4.

[5] Complex wavelets for extended depth-of-field: A new method for the fusion of multichannel microscopy images / Brigitte Forster-Heinlein, Dimitri Van De Ville, Jesse Berent et al. // Microscopy research and technique. — 2004. — 09. — Vol. 65. — P. 33–42.

[6] Corporation AnMo Electronics. Dino-Lite // официальный сайт. — URL: https://www.dino-lite.com/index.php (online; accessed: 28.04.2020).

[7] Corporation AnMo Electronics. DinoConnect. — URL: https://play.google.com/store/apps/details?id=com.anmo.dinoconnect (online; accessed: 28.04.2020).

[8] Corporation AnMo Electronics. DinoDirect. — URL: https://play.google.com/store/apps/details?id=tw.com.anmo.dinodirect&hl=en-EN (online; accessed: 27.04.2020).

[9] Dropbox. Djinni // Github repository. — URL: https://github.com/dropbox/djinni (online; accessed: 20.11.2019).

[10] Extended depth-of-field via focus stacking and graph cuts / C. Zhang, J. Bastian, C. Shen et al. // 2013 IEEE International Conference on Image Processing. — 2013. — Sep. — P. 1272–1276.

[11] FocusALL: Focal Stacking of Microscopic Images Using Modified Harris Corner Response Measure / M. S. Sigdel, M. Sigdel, S. Dinç et al. // IEEE/ACM Transactions on Computational Biology and Bioinformatics. — 2016. — March. — Vol. 13, no. 2. — P. 326–340.

[12] GitHub Inc. GitHub Pages // официальный сайт. — URL: `https://pages.github.com/` (online; accessed: 22.05.2020).

[13] Google. Add Flutter to existing app // документация. — URL: `https://flutter.dev/docs/development/add-to-app` (online; accessed: 14.12.2019).

[14] Google. Camera2 API // документация. — URL: `https://developer.android.com/reference/android/hardware/camera2/package-summary` (online; accessed: 25.05.2020).

[15] Google. Create an Android library. Anatomy of an AAR file // документация. — URL: `https://developer.android.com/studio/projects/android-library#aar-contents` (online; accessed: 14.12.2019).

[16] Google. Distribution dashboard // документация. — URL: `https://developer.android.com/about/dashboards` (online; accessed: 14.12.2019).

[17] Google. Flutter // официальный сайт. — URL: `https://flutter.dev/` (online; accessed: 14.12.2019).

[18] Google. Google Drive // официальный сайт. — URL: `https://www.google.com/drive/` (online; accessed: 22.05.2020).

[19] Google. Google Forms // официальный сайт. — URL: `https://www.google.com/intl/en/forms/about/` (online; accessed: 22.05.2020).

[20] Google. Snapseed. — URL: https://play.google.com/store/apps/details?id=com.niksoftware.snapseed&hl=en-EN (online; accessed: 27.04.2020).

[21] Gradle // официальный сайт. — URL: https://gradle.org/ (online; accessed: 17.12.2019).

[22] Group Biomedical Imaging. Biomedical Imaging Group // Official webpage. — URL: http://bigwww.epfl.ch (online; accessed: 20.11.2019).

[23] Group Biomedical Imaging. DeconvolutionLab2 // Github repository. — URL: https://github.com/Biomedical-Imaging-Group/DeconvolutionLab2 (online; accessed: 20.11.2019).

[24] Group Biomedical Imaging. PSF Generator // Official webpage. — URL: http://bigwww.epfl.ch/algorithms/psfgenerator (online; accessed: 20.11.2019).

[25] HANTOR. Cozy Magnifier Microscope. — URL: https://play.google.com/store/apps/details?id=com.hantor.CozyMag&hl=ru (online; accessed: 27.04.2020).

[26] Harman Mark. HedgeCam 2. — URL: https://play.google.com/store/apps/details?id=com.caddish_hedgehog.hedgecam2 (online; accessed: 27.04.2020).

[27] Helicon Soft Ltd. Helicon Focus // официальный сайт. — URL: https://www.heliconsoft.com/heliconsoft-products/helicon-focus (online; accessed: 20.11.2019).

[28] Hilsenstein V. Robust Autofocusing for Automated Microscopy Imaging of Fluorescently Labelled Bacteria. — 2005. — Dec. — P. 15–15.

[29] Inc. Facebook. React Native // официальный сайт. — URL: https://facebook.github.io/react-native/ (online; accessed: 14.12.2019).

[30] Inc. Facebook. React Native. Integration with Existing Apps // документация. — URL: https://facebook.github.io/

react-native/docs/integration-with-existing-apps.html (online; accessed: 14.12.2019).

[31] Itseez Inc. OpenCV // официальный сайт. — URL: https://opencv.org/ (online; accessed: 20.11.2019).

[32] JK.Fantasy. Magnifier Camera. — URL: https://play.google.com/store/apps/details?id=com.jkfantasy.camera.jkpmagnifiercamera&hl=ru (online; accessed: 27.04.2020).

[33] Limited 123RF. Pixlr. — URL: https://play.google.com/store/apps/details?id=com.pixlr.express (online; accessed: 28.04.2020).

[34] MEL Science LLC. MEL Science // официальный сайт. — URL: https://melscience.com/ (online; accessed: 28.05.2020).

[35] MicroBuilderz. LiquidCore // официальный репозиторий. — URL: https://github.com/LiquidPlayer/LiquidCore (online; accessed: 14.12.2019).

[36] Microsoft. Xamarin // официальный сайт. — URL: https://visualstudio.microsoft.com/xamarin/ (online; accessed: 14.12.2019).

[37] Microsystems Leica. AirLab. — URL: https://play.google.com/store/apps/details?id=com.leicams.leicaair (online; accessed: 27.04.2020).

[38] Naqvi A Manglik N Dudrey E Perry C Mulla ZD Cervantes JL. Evaluating the performance of a low-cost mobile phone attachable microscope in cervical cytology // BMC Womens Health. — 2020.

[39] Oracle. Java Native Interface // документация. — URL: https://docs.oracle.com/javase/8/docs/technotes/guides/jni/ (online; accessed: 14.12.2019).

[40] P2PCam. TinyCapture. — URL: https://play.google.com/store/apps/details?id=com.g_zhang.TinyCapture (online; accessed: 28.04.2020).

[41] Photoshop. Adobe Photoshop Mix. — URL: https://play.google.com/store/apps/details?id=com.adobe.photoshopmix&hl=ru (online; accessed: 27.04.2020).

[42] Project Mono. Embeddinator-4000 // официальный репозиторий. — URL: https://github.com/mono/Embeddinator-4000 (online; accessed: 14.12.2019).

[43] Quantitative Imaging with a Mobile Phone Microscope / Arunan Skandarajah, Clay D. Reber, Neil A. Switz, Daniel A. Fletcher // PLOS ONE. — 2014. — 05. — Vol. 9, no. 5. — P. 1–12. — URL: https://doi.org/10.1371/journal.pone.0096906 (online; accessed: 20.11.2019).

[44] Richardson–Lucy algorithm with total variation regularization for 3D confocal microscope deconvolution / Nicolas Dey, Laure Blanc-Feraud, Christophe Zimmer et al. // Microscopy Research and Technique. — 2006. — Vol. 69, no. 4. — P. 260–266. — URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/jemt.20294 (online; accessed: 20.11.2019).

[45] SWIG // официальный сайт. — URL: http://www.swig.org/ (online; accessed: 14.12.2019).

[46] Serif (Europe) Ltd. Affinity Photo // официальный сайт. — URL: https://affinity.serif.com/ru/photo (online; accessed: 20.11.2019).

[47] Zeiss Carl. Labscope Material. — URL: https://apps.apple.com/app/labscope/id688689220 (online; accessed: 28.04.2020).

[48] Zeiss Carl. Labscope Material // официальный сайт. — URL: https://www.zeiss.ru/consumer-products/home.html (online; accessed: 27.05.2020).

[49] Zerene Systems LLC. Zerene Stacker // официальный сайт. — URL: https://zerenesystems.com/cms/stacker (online; accessed: 20.11.2019).