Разработка мобильного приложения для коррекции динамических искажений на изображениях

В.А. Фурсов[1,2], Э.Ф. Фатхутдинова[1]

1 Самарский национальный исследовательский университет имени академика С.П. Королева
2 Институт систем обработки изображений РАН – филиал ФНИЦ «Кристаллография и фотоника» РАН

Аннотация. Предлагается технология коррекции изображений в мобильных устройствах, основанная на использовании параметризованного КИХ-фильтра. Фильтр строится с использованием частотного отклика, заданного в виде параметрического семейства центрально-симметричных частотных характеристик. Используется модель одномерного частотного отклика в виде отрезков трех функций: параболы, константы и экспоненты. В рамках предлагаемой модели фильтра создано мобильное приложение, реализующее две схемы настройки: идентификация параметров фильтра по эталонному изображению и настройка без эталона. Кроме того, имеется возможность варьировать параметр частотного отклика, характеризующий интервал средних частот, что обеспечивает дополнительные возможности для регулировки качества восстановления. Приводится пример обработки тестовых изображений.

Ключевые слова: обработка изображений, мобильное приложение, КИХ-фильтр, искажения

CEUR-WS.org/Vol-2260/23-468-479.pdf

The technology of correction of dynamic distortions on mobile devices

V.A. Fursov[1,2], E.F. Fatkhutdinova[1]

1 Samara National Research University named after academician S.P. Korolev
2 Image Processing Systems Institute of RAS - branch of the FSRC "Crystallography and Photonics" RAS, Samara, Russia

Abstract. We propose the technology of image correction in mobile devices based on the use of a parametric FIR filter. The filter is constructed using a frequency response specified in the form of a parametric family of centrally symmetric frequency characteristics. A model of a one-dimensional frequency response is used in the form of segments of three functions: parabola, constant, and exponential function. Within

the framework of the proposed filter model, a mobile application has been created that implements two tuning schemes: identification of the filter parameters from the reference image and adjustment without a reference. Besides, it is possible to vary the frequency response parameter characterizing the mid-range interval, which provides additional possibilities for adjusting the quality of the recovery. An example of processing test images is given.

Keywords: processing of images, mobile application, FIR filter, distortion

1. Введение

Статистические данные последних лет показывают, что объем трафика мобильных устройств в сети интернет быстро растет. Мобильное приложение — это программа, работающая на планшетных ПК и смартфонах. Число владельцев смартфонов по всему миру с каждым днём растет. Музыка, работа с фотографиями, социальные сети – самые популярные типы приложений среди владельцев смартфонов [1]. Одна из наиболее популярных функций мобильных устройств – регистрация изображений. Массовое использование этой функции связано с возможностью оперативной регистрации изображений в неожиданных и уникальных ситуациях. При этом часто приходится сталкиваться с проблемами размытия изображения вследствие расфокусировки и/или смаза (при быстром перемещении регистрируемого объекта относительно камеры). Искажения типа смаза и расфокусировки обычно называют динамическими. Задача коррекции динамических искажений на изображениях, регистрируемых мобильными устройствами, с помощью реализованных непосредственно в самом мобильном устройстве алгоритмов цифровой обработки изображений является крайне актуальной.

Известными классическими подходами к построению фильтров для обработки изображений являются инверсная и винеровская фильтрации. При построении инверсного фильтра часто приходится сталкиваться с тем, что обратный оператор не существует или соответствующая передаточная функция имеет полюса, близкие к нулю [2]. При этом на восстановленном изображении происходит подчеркивание шумов. В винеровском фильтре эта проблема преодолевается за счет учета в передаточной функции фильтра частотных характеристик шума [3]. Однако на практике эти характеристики часто отсутствуют, при этом синтез оптимального винеровского фильтра становится серьезной проблемой. К сожалению, другие известные методы построения фильтров, в той или иной форме, также сталкиваются с указанными проблемами [2], [3]. Поэтому часто фильтры для коррекции динамических искажений строятся путем параметрической идентификации в классе фильтров с конечной импульсной характеристикой (КИХ-фильтры). В работе [4] была рассмотрена такая технология, основанная на использовании параметрической модели частотного отклика в виде отрезков квадратичной и экспоненциальной функций. В указанной работе были рассмотрены два варианта технологии: идентификация

параметров импульсного отклика по прецедентам и настройка фильтра по косвенным признакам без эталонного изображения.

В работе [5] модель частотного отклика из отрезков квадратичной и экспоненциальной функций была обобщена, в частности, расширена область средних частот за счет введения интервала постоянства частотного отклика. На модельных примерах было показано, что эта модификация повышает качество восстановления изображений по сравнению с предыдущей реализацией квадратично-экспоненциального КИХ-фильтра. Однако реализация фильтра с расширенной областью частотного отклика требует, хотя и незначительного, увеличения вычислительных ресурсов. В настоящей работе приводятся результаты работ по разработке мобильного приложения, реализующего оба указанные выше варианта.

Работа построена следующим образом. В разделе 1 приводится описание методов и алгоритмов настройки фильтра, как по эталонному изображению, так и при отсутствии эталона. В разделе 2 описываются используемые для построения мобильного приложения библиотеки, псевдокод и интерфейс пользователя. В разделе 3 приводятся примеры обработки тестовых изображений в мобильном устройстве. Сведения, изложенные во введении и в первой части раздела 2 могут представлять интерес для специалистов в области цифровой обработки изображений. Что касается разработки мобильного приложения, достаточно использовать материалы, приведенные в тексте, начиная с формулы (3), приняв эту формулу на веру.

2. Метод и алгоритмы

Строится КИХ-фильтр с радиально симметричным вещественным частотным откликом [2, 3] с опорной областью D в виде $N \times N$ - квадрата с центром в точке $\mathrm{k}_1 = 0$, $\mathrm{k}_2 = 0$. В предположении, что центральный отсчет импульсного отклика *h(0,0)* опорной области *D* находится в точке n_1, n_2, отсчеты восстановленного изображения $y(n_1, n_2)$ [2] можно представить в виде:

$$y(n_1, n_2) = \sum_{k_1=-\frac{N-1}{2}}^{\frac{N-1}{2}} \sum_{k_2=-\frac{N-1}{2}}^{\frac{N-1}{2}} h[r(k_1, k_2)] x(n_1 - k_1, n_2 - k_2), \qquad (1)$$

где $r(k_1, k_2) = \sqrt{k_1^2 + k_2^2}$, а $h[r(k_1, k_2)]$ – отсчеты одномерной импульсной характеристики, определенные на множестве окружностей с радиусами $r = r(k_1, k_2), k_1, k_2 \in D$.

Используется функция одномерного частотного отклика для всех значений $0 \le \omega < \infty$ в виде трех последовательных отрезков: параболы константы и экспоненты:

$$S(\omega) = \begin{cases} a\omega^2, \text{при } 0 \le \omega < \omega_1, \\ A = const = a\omega_1^2, \text{при } \omega_1 \le \omega \le \omega_2, \\ e^{-c\omega}, \text{при } \omega \ge \omega_2, \end{cases} \qquad (2)$$

$$S(\omega_2) = a\omega_1^2 = e^{-c\omega_2}.$$

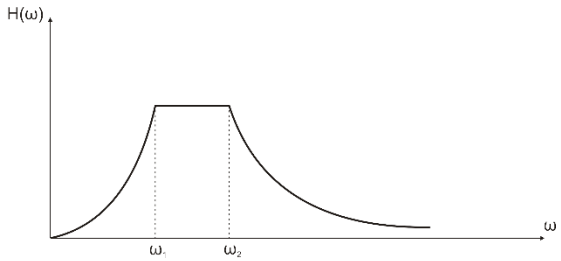

Рис. 1. Типичный график спектра GSE-фильтра

Фильтр, соответствующий описанному частотному отклику, далее будем называть обобщенным квадратично экспоненциальным фильтром (Generalized Square-Exponential) или кратко GSE-фильтром. График указанной функции приведен на рис. 1.

Соответствующий этой спектральной характеристике импульсный отклик, в силу свойства радиальной симметрии, получаем как функцию пространственного параметра r обратным преобразованием Фурье:

$$h(r) = \frac{1}{\pi}\mathrm{Re}\int_0^{\infty} S(\omega)\ e^{j\omega r}d\omega = \frac{1}{\pi}\mathrm{Re}\left\{\int_0^{\omega_1} \alpha\omega^2 e^{j\omega r}d\omega + \int_{\omega_1}^{\omega_2} Ae^{j\omega r}d\omega + \int_{\omega_2}^{\infty} e^{-c\omega}e^{j\omega r}d\omega\right\}$$
$$= \frac{e^{-c\alpha\omega_1}}{\pi}\left(\frac{\sin\omega_1 r}{r}\sin\omega_1 r + \frac{2\cos\omega_1 r}{r^2\omega_1} - \frac{2\sin\omega_1 r}{r^3{\omega_1}^2} + \left(\frac{\sin\alpha\omega_1 r - \sin\omega_1 r}{r}\right)\right.$$
$$\left. + \left(\frac{c\cos\alpha\omega_1 r - r\sin\alpha\omega_1 r}{c^2 + r^2}\right)\right) \qquad (3)$$

(мы исключили a и ω_2, выполнив с учетом (2) замену $a = {e^{-c\omega_2}}/{\omega_1^2}$ и положив $\omega_2 = \alpha\omega_1$).

Легко убедиться, что при $\alpha = 1$ импульсный оклик (3) совпадает с импульсным откликом квадратично-экспоненциального фильтра, который мы рассматривали в предыдущей работе [4]. В настоящей работе мы построим мобильное приложение, в котором пользователь сможет сам задавать этот параметр (или диапазон изменения этого параметра) с учетом субъективных требований к качеству восстановления и доступных вычислительных ресурсов.

Отсчеты двумерного импульсного отклика также как в [5] определяются путем дискретизации непрерывной функции (3) для всех направлений, соответствующих всем отсчетам опорной области. При этом для каждого отсчета (точки k_1, k_2) опорной области в соотношении (3) аргумент $r = r(k_1, k_2) = \sqrt{k_1^2 + k_2^2}$. Поскольку при $r = 0$ в (3) имеет место неопределенность, значение центрального отсчета вычисляется как сумма всех отсчетов за исключением центрального:

$$h(0,0) = \sum h(k_1, k_2), \forall k_1, k_2 \in D, k_1, k_2 \neq 0. \qquad (4)$$

Затем осуществляется нормировка всех отсчетов в опорной области так, чтобы выполнялось требование сохранения среднего уровня яркости обработанного изображения:

$$\sum h(k_1, k_2) = 1, \forall k_1, k_2 \in D. \quad (5)$$

Алгоритм настройки фильтра строится в виде следующей последовательности шагов:

1. Задание начальных оценок параметров $\hat{\omega}_1, \hat{c}, \hat{\alpha}$ и критерия $Q_k(\hat{\omega}_1, \hat{c}, \hat{\alpha})$ (при $k = 0$).
2. Вычисление отсчетов импульсного отклика для всех точек опорной области $D(n_1, n_2)$ с использованием соотношений (3), (4) и нормировка всех отсчетов, удовлетворяющая (5).
3. Обработка искаженного изображения и вычисление критерия качества $Q_k(\hat{\omega}_1, \hat{c}, \hat{\alpha})$.
4. Если $Q_k(\hat{\omega}_1, \hat{c}, \hat{\alpha}) > Q_{k-1}(\hat{\omega}_1, \hat{c}, \hat{\alpha})$, оценки $\hat{\omega}_1, \hat{c}, \hat{\alpha}$ сохраняются, иначе по некоторому правилу формируется новый вариант оценок и осуществляется переход к шагу 2. Если все оценки из области допустимых значений «просмотрены» – выход.

По описанной технологии можно осуществлять настройку параметров фильтра как по эталонному изображению, так и без эталона. При настройке по эталону в качестве критерия $Q_k(\hat{\omega}_1, \hat{c}, \hat{\alpha})$ близости восстановленного изображения к эталону используется показатель:

$$PSNR = 10 \log_{10} \left(\frac{MAX^2}{MSE} \right), \quad (6)$$

где *MAX* – максимальное значение, принимаемое пикселем изображения, а *MSE* – среднеквадратическая ошибка (СКО), характеризующая близость восстановленного изображения к эталонному.

В технологии настройки фильтра без эталона вычисляется СКО между восстановленным и исходным искаженным изображениями. При этом в отличие от настройки по эталону, улучшение качества восстановленного изображения сопровождается уменьшением показателя PSNR (6). Поскольку при уменьшении значения PSNR качество полученного изображения может быть как выше, так и ниже, используется дополнительное условие: увеличение дисперсии обработанного изображения. Это требование обеспечивает увеличение среднего контраста, которое обычно наблюдается при увеличении резкости изображения. Кроме того, вводится ограничение на минимально допустимое значение PSNR, поскольку при его малых значениях возможны значительные искажения, характеризующиеся высоким контрастом.

Таким образом, проверка выполнения на k-ой итерации условия $Q_k(\hat{\omega}_1, \hat{c}, \hat{\alpha}) > Q_{k-1}(\hat{\omega}_1, \hat{c}, \hat{\alpha})$ на шаге 4 описанной технологии сводится к проверке выполнения следующих условий:

$$\begin{aligned} &PSNR(\hat{\omega}_k, \hat{c}_k, \hat{\alpha}_k) < PSNR(\hat{\omega}_{k-1}, \hat{c}_{k-1}, \hat{\alpha}_{k-1}), \\ &D(\hat{\omega}_k, \hat{c}_k, \hat{\alpha}_k) > D(\hat{\omega}_{k-1}, \hat{c}_{k-1}, \hat{\alpha}_{k-1}), \\ &PSNR(\hat{\omega}_k, \hat{c}_k, \hat{\alpha}_k) > PSNR_{\text{доп}}. \end{aligned} \quad (7)$$

Здесь $PSNR(\hat{\omega}_k, \hat{c}_k, \hat{\alpha}_k)$ значение показателя (6), вычисленное на k-ой итерации для изображения, восстановленного фильтром с параметрами $\hat{\omega}_k, \hat{c}_k, \hat{\alpha}_k$ при сравнении с эталонным (при настройке по эталону) или исходным искаженным (при настройке без эталона); $D(\hat{\omega}_k, \hat{c}_k, \hat{\alpha}_k)$ – дисперсия восстановленного изображения, а $PSNR_{\text{доп}}$ – минимально допустимое значение показателя (6).

3. Реализация мобильного приложения

Разработка любого мобильного приложения по обработке изображений делится на 2 части: разработка фильтра и разработка основной части. Для первой части – реализации алгоритма обработки изображений – использовалась библиотека OpenCV, специальная версия для платформы Android. OpenCV – это кроссплатформенная библиотека с открытым исходным кодом, является свободной для использования в коммерческих и академических целях. OpenCV содержит алгоритмы интерпретации изображений, устранения оптических искажений, определения сходства, анализа перемещения объекта, определения формы объекта и многое другое, написана на C/C++ и использует параллельность и многоядерность для выполняемых задач.

Для использования библиотеки OpenCV на платформе Android существует также готовый набор Android NDK – инструментарий, позволяющий реализовывать части приложения для Android на таких компилируемых языках программирования, как C и C++ и содержит библиотеки для управления активностями и доступа к различным физическим компонентам устройства. Android NDK интегрирован с инструментами из набора компонентов для разработки программного обеспечения (Android SDK), а также с интегрированной средой разработки Android Studio.

Таким образом, существуют 2 способа использования библиотеки OpenCV на платформе Android:

1. OpenCV Java API + Android SDK: функциональность пишется на языке Java.
2. OpenCV native interface + Android NDK: функциональность пишется на языке C++.

Мы будем использовать первый способ.

Основные используемые модули библиотеки:

- модуль Core – содержит основные операции, в том числе арифметические;
- модуль Imgproc – отвечает за обработку изображений;
- модуль Utils – содержит вспомогательные методы, например, конвертацию изображения формата Bitmap в формат Mat OpenCV и назад, загрузка Mat из ресурса по идентификатору ресурса.

Так как мы работаем с ограниченным объемом памяти, целесообразно загружать в память изображение с меньшим разрешением. Версия с

уменьшенным разрешением должна соответствовать размеру компонента пользовательского интерфейса, который его отображает. Изображение с высоким разрешением занимает большое количество памяти и не обеспечивает видимого преимущества при отображении. Выбор варианта может осуществляться пользователем с учетом характеристик конкретного мобильного устройства. На рисунке 2 приведен псевдокод программы, реализующий описанный в разделе 2 алгоритм, написанный с учетом особенностей программирования на мобильных устройствах.

начало
для w ∈ (w_in, w_end)/c ∈ (c_in, c_end)/α ∈ (α_in, α_end) с шагом step_w/step_c/step_α выполнить
начало ***функция1***
для i ∈ (1, N) с шагом 1 выполнить
для j ∈ (1, N) с шагом 1 выполнить

$$r = \left|\sqrt{\left(i - \frac{N+1}{2}\right)^2 + \left(j - \frac{N+1}{2}\right)^2}\right|$$

если $r > 0{,}5$ то

$$h(i,j) = \frac{\exp(-c\alpha\omega_1)}{\pi}\left(\frac{\sin\omega_1 r}{r}\sin\omega_1 r + \frac{2\cos\omega_1 r}{r^2\omega_1} - \frac{2\sin\omega_1 r}{r^3{\omega_1}^2} + \left(\frac{\sin\alpha\omega_1 r - \sin\omega_1 r}{r}\right) + \left(\frac{c\cos\alpha\omega_1 r - r\sin\alpha\omega_1 r}{c^2 + r^2}\right)\right)$$

конец
конец
конец
начало ***функция2***
для i ∈ (1, N) с шагом 1 выполнить
для j ∈ (1, N) с шагом 1 выполнить

$$XR(k,l) = XR(k,l) + X\left(k - \frac{N+1}{2} + i, l - \frac{N+1}{2} + j\right)h(i,j)$$

конец
конец
конец
ввод: минимальное значение PSNR
новое значение PSNR = psnr(XR, X, 1)
если вычисленное значение PSNR < предыдущего значения PSNR и
дисперсия обработанного изображения(XR) > дисперсия искаженного изображения (X) то
запоминаем значение параметров w/c/α
запоминаем значение PSNR
конец
вызываем ***функция1*** и ***функция2*** с новыми параметрами w, c, α
вывод: обработанное изображение.
конец

Рис. 2. Псевдокод программы

Для работы с растровыми изображениями используется тип Bitmap, который позволяет сохранить отзывчивость пользовательского интерфейса и уменьшить объем памяти. Далее для обработки изображений тип Bitmap конвертируется в тип Mat. Mat – класс для хранения изображений, который может интерпретироваться языком C++.

С помощью класса BitmapFactory можно узнать разрешение и тип графических данных до создания объекта Bitmap и до выделения памяти на этот объект. При получении информации о разрешении и размере изображения может быть принято решение о загрузке в память полноразмерной версии картинки или ее уменьшенной версии. Перечислим некоторые факторы, которые нужно учитывать.

1. Расчетное потребление памяти при загрузке полной версии.
2. Количество памяти, которое возможно потратить на изображение, учитывая общее потребление памяти приложением.
3. Разрешение компонента, который будет отображать загруженные данные.
4. Размер и плотность точек экрана на текущем устройстве.

Методы класса BitmapFactory не должны исполняться в главном потоке пользовательского интерфейса, чтобы не снижать производительность системы. Нельзя предугадать, сколько времени уйдет на загрузку данных и их обработку, это зависит от различных факторов (скорость чтения с диска, размер изображения, мощность процессора, и т.д). Если такая задача заблокирует главный поток пользовательского интерфейса, то система отметит приложение как зависшее, и пользователь может выйти из него или даже удалить.

Если требуется совершать операции, занимающие некоторое время, их обязательно нужно выполнять в отдельных потоках, называемых «фоновыми» или «рабочими» потоками. Для реализации этой задачи используется класс AsyncTask, который предлагает простой и удобный механизм для перемещения трудоёмких операций в фоновый поток.

Методы класса AsyncTask позволяют выполнять асинхронную работу в пользовательском интерфейсе. Они могут выполнять операции загрузки небольших изображений, файловые операции, операции с базой данных и т.д. в рабочем потоке, а затем публикуют результаты в главном потоке пользовательского интерфейса без необходимости самостоятельно обрабатывать потоки и/или обработчики.

На рисунке 3 представлен макет приложения, сконструированный с помощью онлайн-инструмента для прототипирования приложений marvelapp.com.

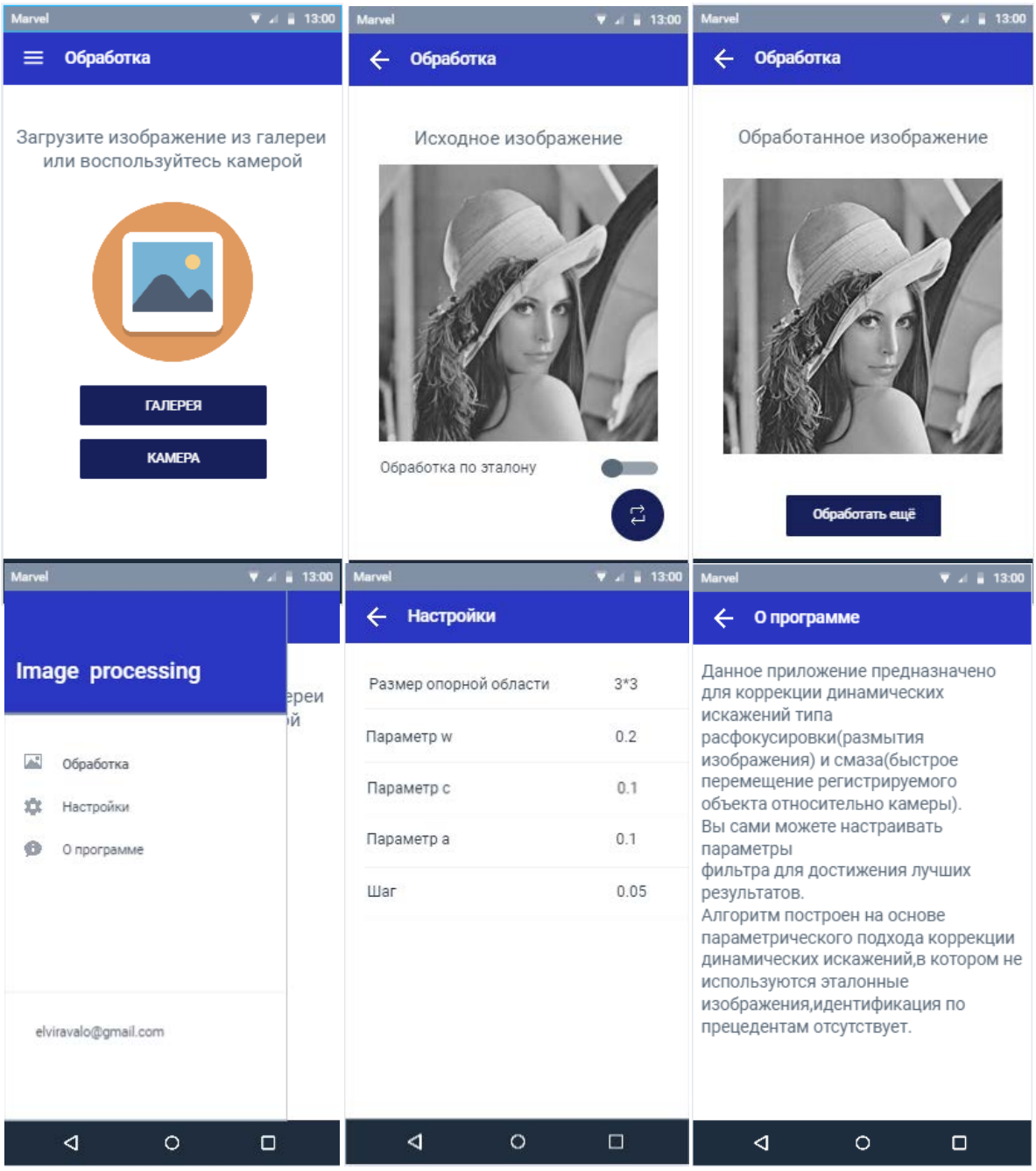

Рис. 3. Макет приложения

При входе в приложение пользователь попадает на главный экран, где может выбрать способ загрузки изображения: выбрать готовое из «галереи» или сделать фотографию самостоятельно с помощью камеры. Причем для новых версий Android, начиная с 6, необходимо разрешить доступ приложению к камере. На следующем шаге пользователь может увидеть загруженное им изображение в компоненте пользовательского интерфейса, принять решение, обрабатывать ли это изображение по эталону, и нажать на кнопку, которая запустит процесс. После обработки пользователь переходит на экран, где может

увидеть результат. Обработанное изображение сохраняется в специальную папку приложения, доступную через встроенное приложение «галерея».

Чтобы поменять параметры фильтра для достижения лучших результатов при обработке, пользователь может перейти в раздел «настройки» главного меню приложения.

Чтобы получить информацию о приложении, пользователю нужно перейти в раздел «о программе».

Диаграмма основных классов приложения, реализующих описанную функциональность, представлена на рис. 4.

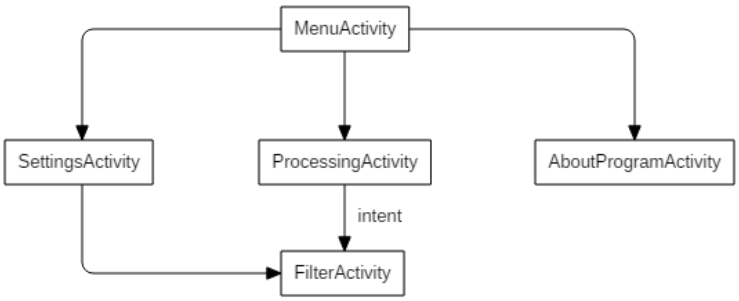

Рис. 4. Диаграмма классов

На рис. 5 представлена подробная схема работы приложения.

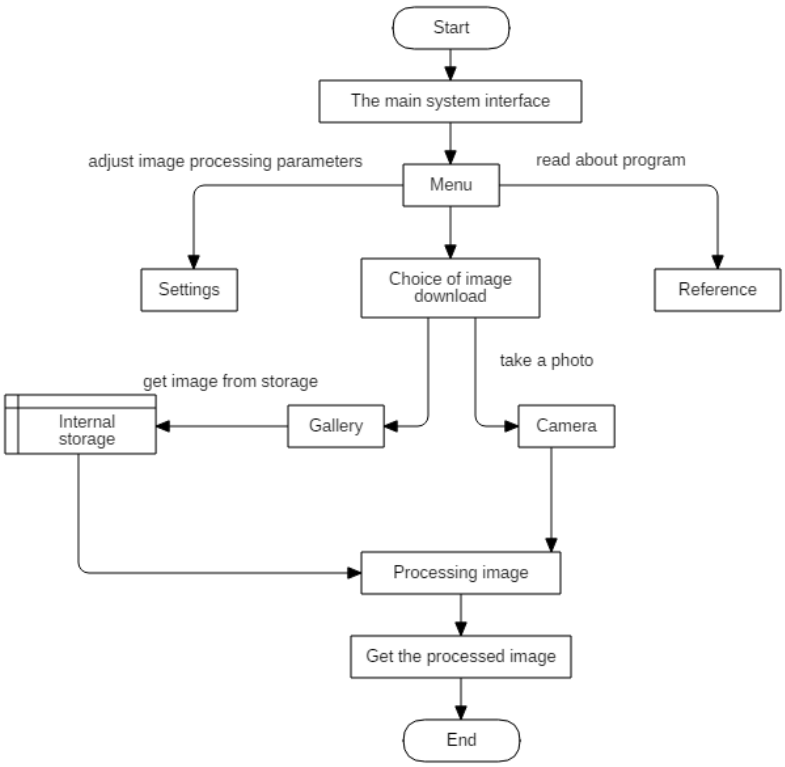

Рис. 5. Схема работы приложения

4. Результаты экспериментов

На рис. 4 представлен результат работы фильтра с настроенными параметрами ω, c и a. На рис. 6(а) приведен пример искаженного полутонового изображения с помощью моделирования фильтра Гаусса нижних частот со степенью размытия: $\sigma = 3$. На рис. 6(б) приведено восстановленное изображение с помощью фильтра, настроенного без эталонного изображения, при размерности опорной области, равной 7×7. Результат, достигнутый при восстановлении: PSNR = 25,13.

Для количественной оценки достижимого качества восстановления мы использовали тестовые изображения (неискаженное и искаженное). Подчеркнем, что для настройки фильтра и восстановления исходное неискаженное изображение не использовалось. Мы осуществляли «слепую» настройку.

(а) (б)

Рис. 6. Изображения «Лена»: (а) – искаженное с размытием $\sigma = 3$, (б) – обработанное

Данный алгоритм восстановления изображения без использования эталонного также может применяться и для восстановления цветных изображений. При этом обработка осуществляется по тем же алгоритмам для каждого компонента.

5. Заключение

Реализован метод коррекции динамических искажений, основанный на «слепой» идентификации параметров восстанавливающего фильтра, не требующей задания эталонного изображения. Разработана программа на платформе Android, предназначенная для обработки изображений в мобильных устройствах. Приведен пример обработки тестового изображения, полученного

путем моделирования фильтра Гаусса нижних частот. Результат, достигнутый при восстановлении изображения со степенью размытия: σ = 3 по критерию PSNR = 25,13. Полученные результаты могут заинтересовать пользователей планшетов и смартфонов. Дальнейшие планы авторов – отработка разработанного приложения с целью сделать его доступным широкому кругу пользователей.

Работа выполнена при поддержке Министерства образования и науки РФ и Российского фонда фундаментальных исследований, проекты № 17-29-03112, № 16-07-00729-а.

Литература

1. Mail.Ru Group. Мобильный Интернет в России. – URL: https://corp.mail.ru/media/files/40314-researchmobilemail.pdf.
2. Soifer V.A. et al. Computer Image Processing, Part II: Methods and algorithms. – VDM Verlag, 2010. – 584 p.
3. Прэтт У. Цифровая обработка изображений. Кн.2. / Пер. с англ. М.: Мир, – 1982. – 480 с.
4 Фурсов В.А., Якимов П.Ю. Интернет-технология коррекции динамических искажений на изображениях в мобильных устройствах // Научный сервис в сети Интернет: труды XIX Всероссийской научной конференции (18-23 сентября 2017 г., г. Новороссийск). – М.: ИПМ им. М.В.Келдыша, 2017. – С. 436-445. – doi:10.20948/abrau-2017-09 .
5. Фурсов, В.А. Построение квадратично-экспоненциальных КИХ-фильтров с расширенной средней областью частотного отклика / В.А. Фурсов // Компьютерная оптика. – 2018. – Т. 42, № 2. – С. 297-305. – doi: 10.18287/2412-6179-2018-42-2-297-305
6. Класс AsyncTask. – URL: http://developer.alexanderklimov.ru/android/theory/asynctask.php
7. Android developers. – URL: https://developer.android.com/index.html
8. Парамонов И.В. Разработка мобильных приложений для платформы Android: учебное пособие. – Ярославль: ЯрГУ, 2013. – 88 с.
9. Харди Б., Филлипс Б., Стюарт К., Марсикано К. Android. Программирование для профессионалов. 2-е изд. – СПб.: Питер, 2016. – 640 с.
10. Howse J. Android Application Programming with OpenCV. – Packt Publishing Ltd, 2013. – P. 1–131.