Приложение для оперативной линейно-нелинейной коррекции мобильных изображений

К.С. Медведева[1]

[1]Самарский национальный исследовательский университет им. академика С.П. Королева, Московское шоссе 34А, Самара, Россия, 443086

Аннотация. В работе рассматривается технология формирования двухэтапного линейно-нелинейного фильтра для устранения различных видов искажений на изображениях, регистрируемых с помощью мобильных устройств под ОС Android. Цель разработки – обеспечить высокое качество коррекции искажений при минимальных вычислительных затратах с возможностью получения результатов обработки в реальном времени. Настройка параметров линейного фильтра проводится за счет слепой параметрической идентификации модели с радиально-симметричным квадратично-экспоненциальным частотным откликом. Параметры нелинейного фильтра настраиваются на основе визуальной оценки обработанных изображений. Приводятся алгоритм, псевдокод программы и результаты экспериментов, показывающие эффективность разработанного приложения по устранению искажений на изображениях.

1. Введение

Мобильные устройства, позволяющие регистрировать изображения, уже стали повсеместными. Однако качество получаемых изображений сильно варьируется, и многие факторы могут привести к тому, что снимки, сделанные с помощью мобильного телефона, будут низкого качества. Помимо низкого качества на отснятых изображениях могут появляться различные типы расфокусировок, смазов, артефактов, которые визуально ухудшают передачу графической информации пользователям смартфонов. Возникают перечисленные дефекты во многом из-за несовершенных условий съемок. Ведь мобильные устройства позволяют в случае необходимости запечатлеть быстро изменяющуюся сцену с движущимися объектами. Но в связи с малой глубиной резкости объектива неизбежно возникают различные искажения.

В связи с чем на сегодняшний момент существует огромное количество программ, позволяющих обрабатывать изображения на персональных компьютерах. Однако пользователи сейчас чаще используют смартфоны, чем ПК, следовательно, актуальной становится задача переноса приложений с компьютера на смартфон или разработка приложений для обработки изображений под мобильные платформы. Учитывая данные обстоятельства, мы поставили задачу разработки мобильного приложение, которое позволяло бы получать обработанные изображения с высоким качеством коррекции искажений при небольших вычислительных затратах. Исходя из поставленной перед нами цели, в настоящей работе рассматривается возможность построения технологии двухэтапного линейно-нелинейного фильтра для восстановления и коррекции изображений [1].

2. Постановка задачи и описание метода

На первом этапе будем строить SE-фильтра [1,2] с центрально-симметричным частотным откликом в виде отрезков квадратичной и экспоненциальной функций. Опорную область задаем в виде квадрата с центром в начальной точки координат. Так как фильтр необходимо применять в мобильных устройствах, то для простоты и скорости реализации применяем вариант частотного отклика, который задается в виде отрезков параболы и экспоненты.

На втором этапе построим нелинейный фильтр [1], для которого необходимо настроить два параметра: $\delta_t r$ - является показателем контрастности и понижает резкость; k - является показателем резкости обрабатываемых изображений.

Тем самым второй этап по реализации нелинейного фильтра заключается в выявлении методики по оценке двух параметров – $\delta_t r$ и k, изменения которых позволяют обеспечить высокий уровень коррекции дефектов на полученных с помощью мобильных устройств изображениях. При этом стоит отметить, что настройку приведенных параметров рекомендуется проводить «вручную». За счет чего значительно снижается вычислительная сложность по реализации мобильного приложения, увеличивается простота настройки фильтра, а также учитываются индивидуальные запросы пользователей по приемлемому количеству артефактов и уровню резкости при обработке изображений. В абсолютном большинстве случаев при съемке с мобильных устройств эталон отсутствует. Поэтому в данной работе будем рассматривать слепую идентификацию параметров фильтра $\hat{k}$ и $\hat{\delta}_{tr}$.

В данной работе предлагается следующий метод настройки данных параметров. Так как задача мобильного приложения получить быстрый результат обработки с минимальными вычислительными мощностями, то в основу оценки качества полученных изображений закладывается визуальная оценка пользователей. При данном подходе значительно повышается юзабилити разработанного приложения и добавляется возможность пользователям самостоятельно определить приемлемый уровень резкости и количество остаточных артефактов.

Поэтому на первом этапе обработки отснятых изображений мобильным устройством используем КИХ-фильтр с радиально симметричным вещественным частотным откликом. В предыдущих работах [2, 5] экспериментальным путем было установлено, что для получения высоких результатов коррекции достаточно настроить два параметра:

- $\hat{\omega}_{1,n}$ – функцию одномерного частотного отклика.
- $\hat{c}_n$ – степень присутствия высоких частот.

Первый параметр требует достаточно «тонкой» настройки, поэтому зададим возможность его изменять от 0,5 до 1 с шагом в 0,01. Второй параметр будем изменять от 5 до 10 с шагом в 1. После того, как первый этап обработки закончен и получившийся результат в большей степени удовлетворяет пользователя – переходим к обработке нелинейным фильтром.

Нелинейный фильтр позволяет скорректировать уровень резкости обработанного изображения. При этом обеспечивается подчеркивание контуров, где наблюдаются значительные перепады яркости. Тем самым изображение, обработанное сначала линейным фильтром, а затем нелинейным, будет отличаться от исходного на небольшом числе участков. Поэтому пользователям предоставляется возможность самостоятельно настроить параметры $\hat{k}$ и $\hat{\delta}_{tr}$ в зависимости от визуальной оценки. Значение k будет изменяться от 0,5 до 1 с шагом 0,01; значение $\hat{\delta}_{tr}$ – от 0,5 до 1 с шагом в 0,1.

Дополнительно хотелось бы отметить, что пользователь получает возможность обработать отснятое изображение как всеми представленными технологиями, так и каждой по отдельности или комбинирую по своему усмотрению данные методы. Комбинирование методов предоставляет дополнительный простор для пользователей в получении графической информации с наилучшими визуальными показателями. В четвертом пункте данной работы

«Результаты экспериментов» будет предоставлена информация по вычислению PSNR между эталонным и обработанным изображениями, что наглядно продемонстрирует эффективность предложенной технологии. При этом стоит понимать, что эталонных изображений, как правило, в мобильных устройствах не имеется. Поэтому дополнительно проведем сравнения искаженного и обработанного фильтрами изображений.

3. Реализация мобильного приложения

В качестве платформы для создания мобильных приложений под ОС Android была выбрана интегрированная среда разработки «Android Studio», базовым языком программирования которой является Java. На рисунке 1 представлено создание нового проекта в выбранной среде разработки. При разработке мобильного приложения используется библиотека OpenCV.

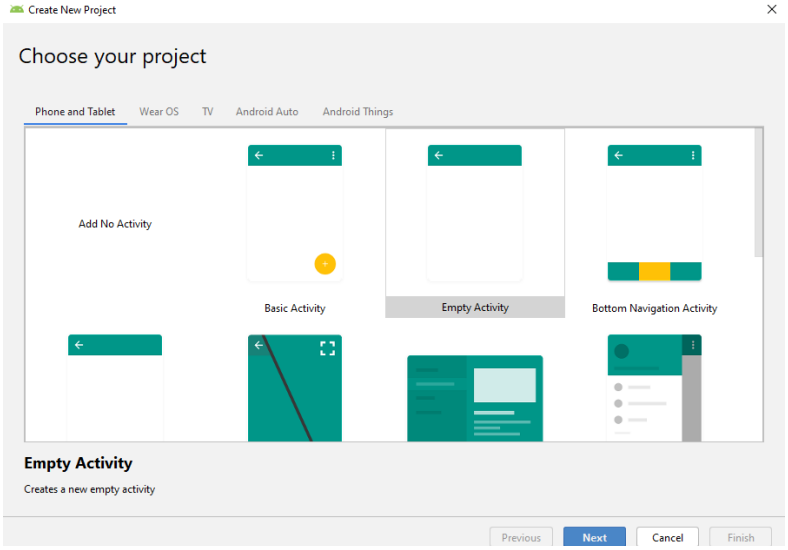


Рисунок 1. Создание нового проекта в «Android Studio».

Разработанное мобильное приложение устанавливается на Android устройство пользователя, которое позволяет производить фотосъемку интересующих объектов с последующей их обработкой двухэтапным линейно-нелинейным фильтром. Так как приложение позволяет сохранять обработанные изображения с различными настройками фильтра, которые могут являться как конечным результатом, так и промежуточным для визуальной оценки и сравнения, то было решено разработать облачную, SQL-ориентированную базу данных, которая располагается на сервере. Приложение сохраняет обработанные пользователем изображения и получив запрос, отображает необходимые данные.

При разработке интерфейса мобильного приложения были использованы стандартные компоненты «Android Studio» (рисунок 2). При запуске приложения пользователю необходимо в активное поле загрузить изображение, которое необходимо обработать. Первый этап обработки – это линейный фильтр. Посредствам настройки двух параметров $\hat{\omega}_{1,n}$ и $\hat{c}_n$ за счет слайдера происходит корректировка уровня искажений. Пользователь может сохранить в память устройства обработанное изображение и на этом завершить его использование. Или перейти ко второму этапу обработки – нелинейному фильтру, если необходимо скорректировать уровень яркости. Для этого также настраивается два параметра $\hat{k}$ и $\hat{\delta}_{tr}$ слайдером. Для дополнительного удобства пользователей приложение позволяет сохранить несколько промежуточных результатов обработки, чтобы затем выбрать наилучший вариант обработанных изображений.

На первом этапе тестирование приложения происходило на встроенном эмуляторе «Android Studio» на различных версиях ОС Android. После того, как тесты на эмуляторе были завершены, данное мобильное приложение было протестировано на устройстве под

управлением операционной системы Android. Оно имеет высокий уровень юзабилити и простой, интуитивно понятный интерфейс, который адаптируется под размер дисплея устройства (рисунок 2).

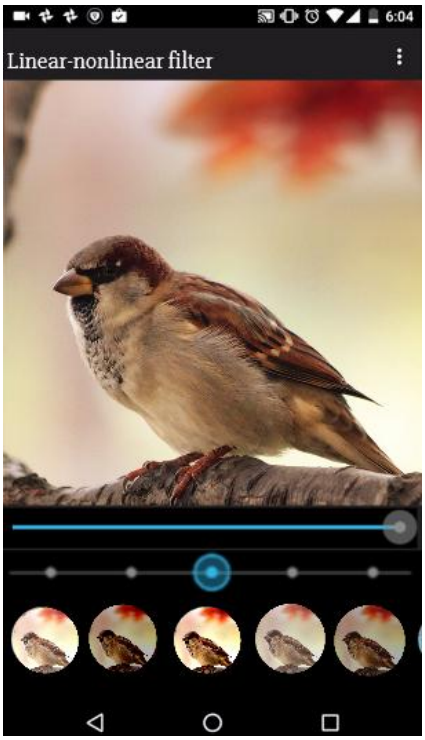


Рисунок 2. Установленное мобильное приложение на устройство Android.

На рисунке 3 представлен алгоритм работы мобильного приложения под ОС Android по обработке изображений двухэтапным линейно-нелинейным фильтром.

Рисунок 3. Алгоритм работы мобильного приложения.

На рисунке 4 представлен псевдокод программного модуля, реализующего обработку полученных с помощью мобильных устройств изображений на языке Java.

```
// c – параметр, характеризующий степень присутствия высоких частот, c=5
// a – параметр, характеризующий ширину области средних частот, a = 1
// w – параметр, характеризующий частотный отклик
// beta_tr – параметр порогового значения, задается пользователем <1
// k* -  угловой коэффициент линейной части нелинейной функции, задается пользователем <1
// PSNR_addit – допустимое значение PSNR, задается пользователем
// r – импульсный отклик одного (пространственного) параметра
// t – свертка
// N – размер опорной области, задается пользователем

class SE_Filter_ nonlinear_correction {
public static void main(String args[]){
int c = 5;
int a = 1;
beta_tr = 0.1;
k* = 0.5;
PSNR_addit = 30;
N=3;

File Image= new File("mob_im1.tif"); / / Загруженное изображение
      File= new File("mob_im2.tif"); / / Обработанное изображение
      BufferedImage img=ImageIO.read(Image);
      for(int i= 1;i< img.getWidth()- 1;i++)
         for( int j= 1;j< img.getHeight()- 1;j++)
         {

double r =  Math.abs(Math.sqrt(Math.pow((i-(N+1/2),2)+Math.pow((j-(N+1)/2),2)));
if (r>>0.5) {
h[i][j] = Math.exp(-c*w)*((Math.sin(w*r))/r+2*(Math.cos(w*r))/((Math.pow(r, 2))*w) - 2*(Math.sin(w*r))/(Math.pow(r, 3))*(Math.pow(w,2))+((Math.sin(a*w*r)) - (Math.sin(w*r)))/r+(c*(Math.cos(a*w*r))-r*(Math.sin(a*w*r)))/((Math.pow(c, 2))+(Math.pow(r,2))))/3.14;
            }
         }
      }

for (int k=(N+1)/2; k<imagesize-((N-1)/2)-1; k++) {
         for (int l=(N + 1)/2; l<imagesize-((N-1)/2)-1; l++) {
 mob_im2[k][l] = 0;
            for(int i=1; i<N; i++) {
               for(int j=1; j<N; j++) {
double mse = 0;
      int width = mob_im1.getWidth();
      int height = mob_im2.getHeight();
      Raster r1 = mob_im1.getRaster();
      Raster r2 = mob_im2.getRaster();
      for (int j = 0; j < height; j++)
         for (int i = 0; i < width; i++)
            mse += Math.pow(r1.getSample(i, j, 0) - r2.getSample(i, j, 0), 2);

      mse /= (double) (width * height);
      double psnr = 10.0 * logbase10(Math.pow(255, 2) / mse);
      System.err.println("PSNR = " + psnr);
      return String.format("%.4f", psnr );
   }while(PSNR >= PSNR_addit) { // условие входа в цикл
   }
}

public static String SD (BufferedImage mob_im1, BufferedImage mob_im2) {
      assert(
            mob_im1.getType() == mob_im2.getType()
                  && mob_im1.getHeight() == mob_im2.getHeight()
                  && mob_im1.getWidth() == mob_im2.getWidth());

      double sd = 0;
      int sum = 0;
      int width = mob_im1.getWidth();
      int height = mob_im1.getHeight();
      Raster r1 = mob_im1.getRaster();
      Raster r2 = mob_im2.getRaster();
      for (int i = 0; i < width; i++){
         for (int j = 0; j < height; j++){
            sd += Math.pow(r1.getSample(i, j, 0) - r2.getSample(i, j, 0), 2);
            if(r1.getSample(i, j, 0) == r2.getSample(i, j, 0)){
               sum++;
            }
         }
      }

      sd /= (double) (width * height);
      System.err.println("SD = " + mse);
      float x = (100*sum)/(width * height);
      System.out.println("Standard deviation (SD) ="+x+"%");
      return String.format("%.2f",mse);
   }

for (int x = 0; x<width; x++)
```

```
                    mob_im2[k][l] = (mob_im2[k][l] +
mob_im1[k-((N + 1)/ 2)+i][l-((N+1)/2)+j]) * hn[i][j];
                }
            }
        }
    }
    return mob_im2;

public static String  PSNR(BufferedImage mob_im1,
BufferedImage mob_im2) {
    assert(
          mob_im1.getType() ==
mob_im2.getType()
                && mob_im1.getHeight() ==
mob_im2.getHeight()
                && mob_im1.getWidth() ==
mob_im2.getWidth());
```

```
    {
     for (int y = 0; y<height; y++)
double t =  Math.h*([k][1],[k][2])*
Math.x([n][1]+[k][1],[n][2]+[k][2])
float[] kernel=[h]*
[k][1],[k][2]=h*[0]/sqrt(([k][1],2)+(([k][1],2))
  for(int [k][1]= 1; [k][1]< img.getWidth()- 1; [k][1]++)
       for( int [k][2]= 1; [k][2]< img.getHeight()- 1;
[k][2]++)
       {
BufferedImageOp op = new ConvolveOp(new Kernel);
     Object blurredImage = op.filter(mob_im3);

return mob_im3; // если при визуальной оценке
качество изображения удовлетворяет потребности
пользователя, то работа программы завершается
// если нет, то задаются новые значения переменных
beta_tr, k*, PSNR_addit и программа запускается еще
раз
 }
  }
```

Рисунок 4. Псевдокоды программных модулей на языке Java.

4. Результаты экспериментов

Цель настоящей работы показать возможности слепой настройки параметров фильтра (deblurring) на изображениях зарегистрированных с использованием мобильных устройств без использования эталона. Для этого будем использовать изображение, полученное со смартфона BQ-6000L, диафрагма f/2, выдержка 1/3559 с., скорость ISO – ISO-115, фокусное расстояние 4 мм, которое представлено на рисунке 5 «Water channel». Искажения появились при съемке фотографии за счет движения руки снимающего. Данная ситуация может возникнуть практически у любого пользователя смартфона, после чего изображение считается безвозвратно испорчено, если не применить последующую дополнительную обработку. Искаженное изображение будем обрабатывать линейно-нелинейным фильтром на мобильном устройстве.

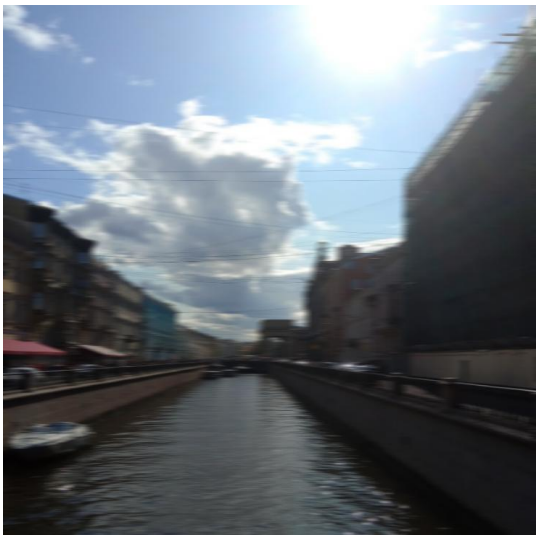


Рисунок 5. Исходное изображение – «Water channel».

Так как в данной работе рассматривается вариант слепой идентификации параметров фильтра, то возможность рассмотреть улучшения изображений после обработки фильтром предоставляется на основании визуальной оценки.

На рисунке 3 представлен интерфейс мобильного приложения. Пользователь загружает необходимое для обработки изображение из галереи. Первый этап обработки – это применение линейного фильтра для устранения искажений и расфокусировок. Под формой с обрабатываемым изображением расположены два слайдера. Первый позволяет настраивать параметр $\hat{\omega}_{1,n}$, изменяя его значение от 0,5 до 1 с шагом в 0,01. Второй вносит изменения параметра $\hat{c}_n$ от 5 до 10 с шагом в 1. Ниже в овальных областях выводятся промежуточные результаты обработки фильтром, чтобы пользователи могли выбрать наиболее удовлетворительный результат. На рисунке 6 приведен результат обработки исходного изображения линейным фильтром без использования эталонного изображения.

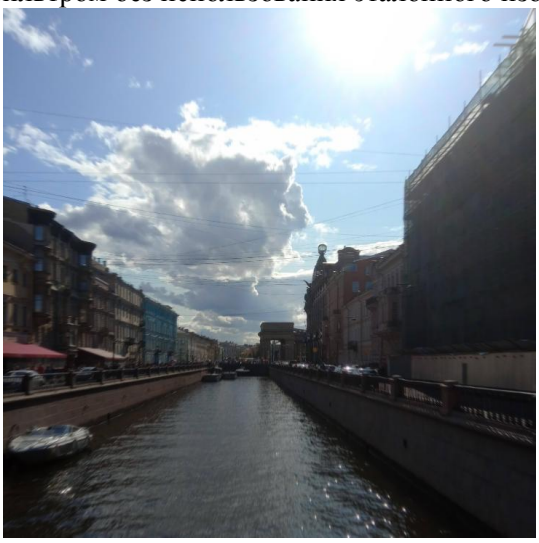


Рисунок 6. Изображение, обработанное линейным фильтром.

После устранения искажений, пользователи могут перейти ко второму этапу обработки изображений нелинейным фильтром. Здесь также по аналогии с линейным фильтром первый слайдер отвечает за настройку параметра $\hat{k}$ и изменяется от 0,5 до 1 с шагом в 0,01, а второй – параметр $\hat{\delta}_{tr}$ с изменениями значений от 0,5 до 1 с шагом в 0,1. На рисунке 7 представлено обработанное изображение с различными настройками резкости. Визуально можно оценить уменьшение размытия (это происходит за счет линейного фильтра), но настройка резкости выбирается индивидуально каждым пользователем самостоятельно. Поэтому в работе приводится несколько вариантов обработанных изображений с различными настройками резкости, чтобы каждый мог подобрать наилучший вариант лично для себя.

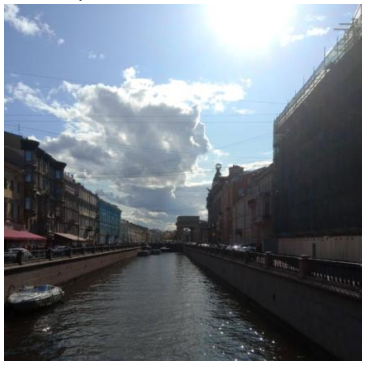
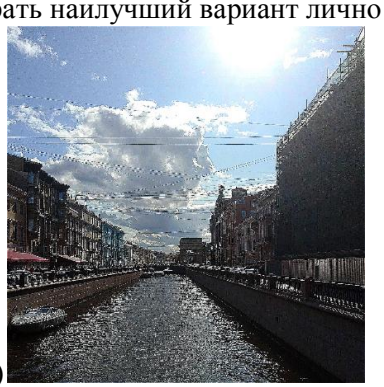
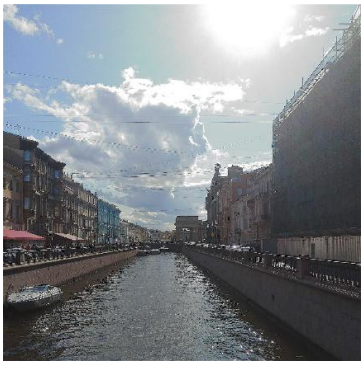
а)  б)  в) 

Рисунок 7. Обработанные изображения нелинейным фильтром с различными настройками резкости.

Настройки параметров двухэтапного линейно-нелинейного фильтра подбирались таким образом, чтобы наилучшим образом убрать искажения и продемонстрировать результаты при различных настройках резкости нелинейного фильтра. Наиболее часто для измерения уровня искажений используют показатель PSNR. Стоит заметить, что пользователям мобильного приложения не всегда будет важен показатель PSNR, гораздо важнее визуальная оценка, которая бы полностью удовлетворяла их потребностям. Следовательно, настройка параметров фильтра для каждого отдельного пользователя может быть своя и не давать наибольший показатель PSNR, но визуально изображение будет смотреться лучшим образом. Приложение разработано так чтобы была возможность настроить как все четыре перечисленных параметра фильтра, так и только некоторые из них, а остальные оставить исходными. Все это открывает дополнительный простор для обработки мобильных приложений и получения удовлетворительных результатов работы двухэтапного линейно-нелинейного фильтра.

5. Заключение

В представленной работе описывается реализация мобильного приложения под ОС Android для восстановления изображений двухэтапным линейно-нелинейным фильтром. Достоинством предложенного метода является простота реализации и возможность обработки мобильных изображений с достижением высокого качества без использования эталонного изображения. Метод слепой коррекции искажений позволяет устранить смазы и расфокусировки в режиме реального времени и с задействованием небольших вычислительных мощностей мобильных устройств.

Дополнительным достоинством реализации предложенной технологии является возможность различных настроек параметров фильтра. Их можно комбинировать между собой и вносить изменения на основе индивидуальной визуальной оценки, которая для каждого пользователя будет разной. Тем самым пользователям мобильного приложения доступна технология, позволяющая восстанавливать изображения и настраивать резкость или контрастность на основе личных предпочтений и потребностей. При этом возможно сохранение промежуточных результатов обработки, что повышает наглядность при выборе наилучшего результата.

Приведен пример обработки мобильным приложением изображения с реальными искажениями, полученными в ходе съемки. Изображение обрабатывалось линейным фильтром и нелинейным. Для нелинейного приведены обработанные изображения с различными настройками резкости. Полученные результаты могут заинтересовать пользователей мобильных устройств.

6. Благодарности

Работа выполнена при поддержке Российского Фонда Фундаментальных Исследований (проект №17-29-03112) и Министерства науки и образования Российской Федерации (госзадание).

7. Литература

[1] Фурсов, В.А. Технология повышения детализации изображений с нелинейной коррекцией высокоградиентных фрагментов / В.А. Фурсов, Е.В. Гошин, К.С. Медведева // Компьютерная оптика. – 2019. – Т. 43, № 3. – С. 484-491. DOI: 10.18287/281 2412-6179-2019-43-3-484-491.

[2] Фурсов, В.А. Технология формирования адаптивных восстанавливающих фильтров в мобильных устройствах / В.А. Фурсов, К.С. Медведева, Э.Ф. Фатхутдинова // Сборник трудов V междунар. конф. и молодеж. шк. «Информ. технологии и нанотехнологии» (ИТНТ) – Самара: Новая техника, 2019. – Т. 4. – 2019. – С. 812-820.

[3] Винокуров, А.С. Разработка мобильного приложения для музыкального магазина в среде Android Studio / А.С. Винокуров, Р.И. Баженов // Постулат. – 2016. – № 9.

[4] Ким, В.Ю. Особенности разработки дизайна пользовательского интерфейса для мобильного приложения // Новые информационные технологии в автоматизированных системах, 2015.

[5] Рахимов, Б.К. Мобильные приложения // Технические науки: теория и практика: материалы III Междунар. науч. конф. – Чита: Издательство Молодой ученый, 2016. – С. 15-16.
[6] Сорока, В.Г. Современные подходы к разработке мобильных приложений для платформы Android / В.Г. Сорока, В.И. Сабурова, Д.С. Фатеев // Молодой ученый. – 2017. – № 25. – С. 47-49.
[7] Зверев, Г.И. Особенности эмуляции мобильной операционной системы Android // Молодой ученый. – 2017. – № 3. – С. 93-96.
[8] Гриффтс, Д. Head First. Программирование для Android – Санкт-Петербург: Питер, 2018.
[9] Дейтел, П. Android для разработчиков / П. Дейтел, Х. Дейтел, А. Уолд – Санкт-Петербург: Питер, 2016.
[10] Библиотека обработки изображений OpenCV [Электронный ресурс]. – Режим доступа: http://opencv.org.

Application for operational linear-nonlinear correction of mobile images

K.S. Medvedeva[1]

[1]Samara National Research University, Moskovskoe Shosse 34A, Samara, Russia, 443086

Abstract. The paper considers the technology of forming a two-stage linear-nonlinear filter to eliminate various types of distortion in images recorded using mobile devices running Android. The purpose of the development is to provide high quality distortion correction with minimal computational costs with the possibility of obtaining processing results in real time. The linear filter parameters are adjusted due to blind parametric identification of the model with a radially symmetric quadratic-exponential frequency response. Non-linear filter parameters are adjusted based on a visual assessment of the processed images. An algorithm, pseudo-code of the program, and experimental results are presented that show the effectiveness of the developed application for eliminating distortions in images.