МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САМАРСКИЙ ГОСУДАРСТВЕННЫЙ АЭРОКОСМИЧЕСКИЙ
УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С. П. КОРОЛЕВА
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ» (СГАУ)

УДК ______________________
№ госрегистрации ______________
Инв. № ______________________

«УТВЕРЖДАЮ»
Проректор университета
по науке и инновациям
д. т. н., профессор
А. Б. Порфирьев
________ 20____ г.

ОТЧЁТ
О НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ

по теме

ВЫДЕЛЕНИЕ ЦЕНТРАЛЬНЫХ ЛИНИЙ СОСУДОВ С ИСПОЛЬЗОВАНИЕМ ВЕЙВЛЕТ-ПРЕОБРАЗОВАНИЯ

Студент гр. 6407 Ионкин М. А.
Руководитель работы Ильясова Н. Ю.

САМАРА 2015

РЕФЕРАТ

Отчет 25 с., 6 рисунков, 10 использованных источников.

ВЕЙВЛЕТ, ДВУМЕРНОЕ ВЕЙВЛЕТ-ПРЕОБРАЗОВАНИЕ, ОБРАБОТКА ИЗОБРАЖЕНИЙ, ЦЕНТРАЛЬНЫЕ ЛИНИИ СОСУДОВ

ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

$2D\ \mathbb{C}WM$	двумерный комплексный вейвлет Морле;
$\hat{f}$	фурье-преобразование функции f;
f^*	комплексно-сопряженная к f функция;
Δf	лапласиан функции f;
$L^p(\mathbb{R}^n)$	пространство измеримых функций $\{f(x),\ x \in \mathbb{R}^n\}$, p-я ($p \geqslant 1$) степень которых абсолютно интегрируема;
$\\|f\\|$	норма функции f гильбертова пространства;
$\langle f, \phi \rangle$	скалярное произведение функций $f(x)$,$\phi(x) \in L^2(\mathbb{R}^n)$;
$\lvert x \rvert$	евклидова норма вектора x.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Выделение центральных линий сосудов на изображении помогает врачу произвести диагностику таких заболеваний, как гипертония, сахарный диабет, атеросклероз, сердечно-сосудистые заболевания и инсульт [1], а также наиболее распространенную причину слепоты — диабетическую ретинопатию. Уже созданы приборы (напр., Navitel), использующие сведения о выделенных линиях сосудов.

Автоматизация этого процесса может позволить:

– перенести рутинную работу на компьютер;

– ускорить процесс диагностики;

– уменьшить фактор субъективности;

– уменьшить вероятность ошибки;

– удешевить диагностику;

– сделать процесс диагностики доступным менее квалифицированному специалисту.

С 1980-х годов вейвлеты стали все более активно применяться в различных отраслях [2, 3]. В [1] описано применение вейвлета Морле (см. подраздел 1.3.2) в процессе сегментации кровеносных сосудов, и приводится результат выделения сосудов. Однако, в данном источнике очень сжато и нечетко описан процесс, приведший к результату. На взгляд автора, применение вейвлетов в медицине в русскоязычных источниках недостаточно освещено.

Для осуществления дискретных (в том числе двумерных) вейвлет-преобразований существуют математические пакеты от *Matlab*, *Scilab* и *Mathematica*. Также есть *open source*[1)] пакеты на таких языках, как *Fortran*, *Phyton* и, в смеси с последним, на *C*. В данной работе приводится реализация двумерных вейвлет-преобразований, — как непрерывных, так и дискретных, — на платформе .NET. Затем каждое из реализованных преобразований тестируется на некотором наборе изображений (которые предварительно фильтруются): производится настройка программного комплекса. Наконец, полученный программный комплекс проходит тестирование на контрольной выборке.

1) англ. open-source software — открытое программное обеспечение

1 ВЕЙВЛЕТ

1.1 Определение

1.1.1 Одномерный вейвлет

Стефан Маллат в [4] определяет одномерный вейвлет как функцию $\psi(t) \in L^2(\mathbb{R})$ с нулевым средним:

$$\int_{\mathbb{R}} \psi(t)\ \mathrm{d}t = 0, \tag{1.1}$$

с единичной нормой:

$$\int_{\mathbb{R}} \left|\psi(t)\right|^2\ \mathrm{d}t = 1, \tag{1.2}$$

сосредоточенную в окрестности точки $t = 0$.

Временно-частотные коэффициенты $\{\ \psi_{b,s}\ \}$ вычисляются путем масштабирования ψ по s и смещения по b:

$$\psi_{b,s}(t) = \frac{1}{\sqrt{s}}\psi(\frac{t-b}{s}), \qquad b \in \mathbb{R},\ s \in \mathbb{R}^+. \tag{1.3}$$

Дискретный случай более сложен для описания [4]:

– регулярность дискретной последовательности не определена;

– необходимо сохранение свойств ортогональности;

– для него нужно отдельно рассматривать граничные случаи.

Однако, дискретные алгоритмы требуют вычисления лишь конечного числа свёрток.

Переход от непрерывности к дискретности дает формулу для вычисления вейвлет-коэффициентов [5]:

$$\psi_{n,i}(t) = \frac{1}{\sqrt{2^i}}\psi(\frac{t}{2^i} - n), \qquad n \in \mathbb{Z},\ i \in \mathbb{N}. \tag{1.4}$$

Примечание. Вместо числа ’2’ в формуле (1.4), вообще говоря, может стоять и другое положительное число.

1.1.2 Двумерный вейвлет

Согласно [6], двумерный вейвлет есть комплекснозначная функция $\psi(x) \in L^2(\mathbb{R}^2)$, для которой выполняется условие

$$c_\psi \equiv (2\pi)^2 \int_{\mathbb{R}^2} \left|\hat{\psi}(x)\right|^2 \frac{\mathrm{d}^2x}{\left|x\right|^2} < \infty. \tag{1.5}$$

Если ψ — регулярная в $\mathbb{R}^2$, то из (1.5) следует выполнение условия

$$\hat{\psi}(0) = 0 \;\Leftrightarrow\; \int_{\mathbb{R}^2} \psi(x)\ \mathrm{d}^2x = 0. \tag{1.6}$$

Дискретный двумерный вейвлет можно определить через произведение ортогональных одномерных вейвлетов.

Тривиальных подход заключается в образовании нового вейвлета путем тензорного произведения [3]. В этом случае сжатие по двум осям может происходить независимо, и вейвлет-коэффициенты находятся из уравнения

$$\psi_{j_1,k_1,j_2,k_2}(x_1, x_2) = \psi_{j_1,k_1}(x_1)\ \psi_{j_2,k_2}(x_2). \tag{1.7}$$

Однако часто использую другой способ: строят вейвлет

$$2^j\psi(2^jx - k, 2^jy - l),$$

где $j,k,l \in \mathbb{Z}$ [3]. Для построения рассмотрим масштабную функцию вейвлета.

1.1.3 Масштабная функция

Скейлинг-функцией, или масштабной функцией (см. [3, 4]) называют функцию φ, удовлетворяющую уравнению

$$\varphi(x) = \sqrt{2} \sum_{k\in \mathbb{Z}} h[k]\varphi(2x - k), \tag{1.8}$$

где $\{h[k]\}$ — некоторый набор коэффициентов, определяемых вейвлет-функцией.

Для скейлинг-функции должно выполняться условие нормировки:

$$\int_{\mathbb{R}} \varphi(x)\,\mathrm{d}x = 1, \tag{1.9}$$

и, согласно [4, 7.1.4], соотношение

$$\widehat{\psi}(\omega) = \frac{1}{\sqrt{2}} e^{-i\omega}\ \widehat{h}^*(\frac{\omega + \pi}{2})\ \widehat{\varphi}(\frac{\omega}{2}). \tag{1.10}$$

1.1.4 Построение двумерных дискретный вейвлетов

Пусть имеется скейлинг-функция φ и одномерный вейвлет $\psi(x-k)$. Тогда мы можем (см. [3]) сконструировать три двумерных дискретных вейвлета:

$$\begin{cases} \Psi^{[1]}_{j,k,l}(x,y) = & 2^j\varphi(2^jx-k)\psi(2^jy-l), \\ \Psi^{[2]}_{j,k,l}(x,y) = & 2^j\psi(2^jx-k)\varphi(2^jy-l), \\ \Psi^{[3]}_{j,k,l}(x,y) = & 2^j\psi(2^jx-k)\psi(2^jy-l). \end{cases} \tag{1.11}$$

Один из этих вейвлетов хорошо обрабатывает вертикальные линии, другой — горизонтальные, ещё один — диагональные. Недостаток данных вейвлетов в том, что в такой трактовке они могут применяться лишь к квадратным (по размеру) изображениям. В зависимости от выбранного параметра j также лучше различаются мелкие или крупные детали [3].

1.2 Вейвлет-преобразование

Пусть $f(x) \in L^2(\mathbb{R}^2)$ — функция, которую можно разложить по ортогональному базису $\psi(x)$.

Двумерное вейвлет-преобразование с вейвлетом ψ и сигналом $f(x)$ может определяться [1] уравнением вида

$$W^f_\psi(b,\theta,a) = c_\psi^{-1/2}\frac{1}{a}\int_{\mathbb{R}^2} f(x)\ \psi^*(r_{-\theta}\frac{x-b}{a})\,\mathrm{d}^2x, \tag{1.12}$$

где r_θ — матрица вращения:

$$r_\theta = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix};$$

$c_\psi \in \mathbb{R}_+$ — нормализующая константа из уравнения (1.5);
$b \in \mathbb{R}^2$ — вектор смещения;
$\theta \in [0,2\pi)$ — угол поворота;
$a \in \mathbb{R}_+$ — параметр масштаба.

Такой тип вейвлет-преобразования применяется, в частности, к ориентированному[1)] двумерному вейвлету Морле.

Другое вейвлет-преобразование [7] задается уравнением

$$W^f_\psi(b,a) = \frac{1}{\sqrt{a}}\int_{\mathbb{R}^2} f(x)\ \psi(\frac{x-b}{a})\,\mathrm{d}^2x \tag{1.13}$$

и применяется к неориентированным вейвлетам.

[1)] с помощью него можно определить “направление” объекта на изображении

1.3 Виды вейвлетов

1.3.1 Вейвлет Марра, или мексиканская шляпа

Вейвлет Марра используется для анализа неориентированных объектов. Он описывается уравнениями вида

$$\psi_H(x) = (-\Delta)^n \ \exp\left\{-\frac{1}{2}|x|^2\right\}, \tag{1.14}$$

где $n = 1{,}2,\ldots$. Он характеризуется тем, что при увеличении n все больше его моментов равны нулю [6]. Применение вейвлета Марра в обработке изображений описано, например, в [7] (в частности, с помощью этого преобразование происходило выделение границ).

1.3.2 Вейвлет Морле

Вейвлет Морле относится к классу вейвлетов, используемых для анализа ориентированных объектов (сегментов, краев, поля направления и т.д.) [6]. Двумерный комплексный вейвлет Морле (2D $\mathbb{C}$WM) $\psi_M(x) : \ \mathbb{R}^2 \ \to \ \mathbb{C}$ задается уравнением [6]

$$\psi_M(x) = \ \exp\{ik_0 x\} \exp\left\{-\frac{1}{2}|Ax|^2\right\}, \tag{1.15}$$

где $k_0 \ \in \ \mathbb{R}^2$ — волновое число, соответствующее вейвлету ψ;
$A = diag\Big[\varepsilon^{-1/2}, \ 1\Big], \ \varepsilon \ \geq 1.$
Точность выполнения условия (1.6) при $\varepsilon = 1$ определяется величиной $\sqrt{2\pi}\exp\left\{-\frac{|k_0|^2}{2}\right\}$: для точности $4 \cdot 10^{-8}$ достаточно, чтобы $|k_0| \geq 6$. Поэтому вектор k_0 выбирается исходя из значения ε и требуемой точности решения задачи.

1.3.3 Вейвлет Добеши

Рассмотрим скейлинг-функцию вида

$$\varphi(x) = \ \sqrt{2}\sum_{k=0}^{N} d_k \ \varphi(2x - N). \tag{1.16}$$

Возьмём $N = 3$ и определим коэффициенты $\{d_k\}$ (см. [8]):

$$d_0 = \ \frac{1+\sqrt{3}}{4}, \quad d_1 = \ \frac{3+\sqrt{3}}{4}, \quad d_2 = \ \frac{3-\sqrt{3}}{4}, \quad d_3 = \ \frac{1-\sqrt{3}}{4}. \tag{1.17}$$

Согласно [3] вейвлет-коэффициенты будут однозначно определяться через коэффициенты масшабной фукнции:

$$\phi(x) = \sqrt{2} \sum_{k=0}^{N} g_k \, \varphi(2x-k), \tag{1.18}$$

где $g_k = (-1)^k d_{N-k}$ — вейвлет-коэффициенты.

С помощью вейвлета Добеши можно осуществить, например, следующее преобразование (на данный момент преобразования выполнены посредством *Matlab*).

Рисунок 1 – Исходное изображение №1

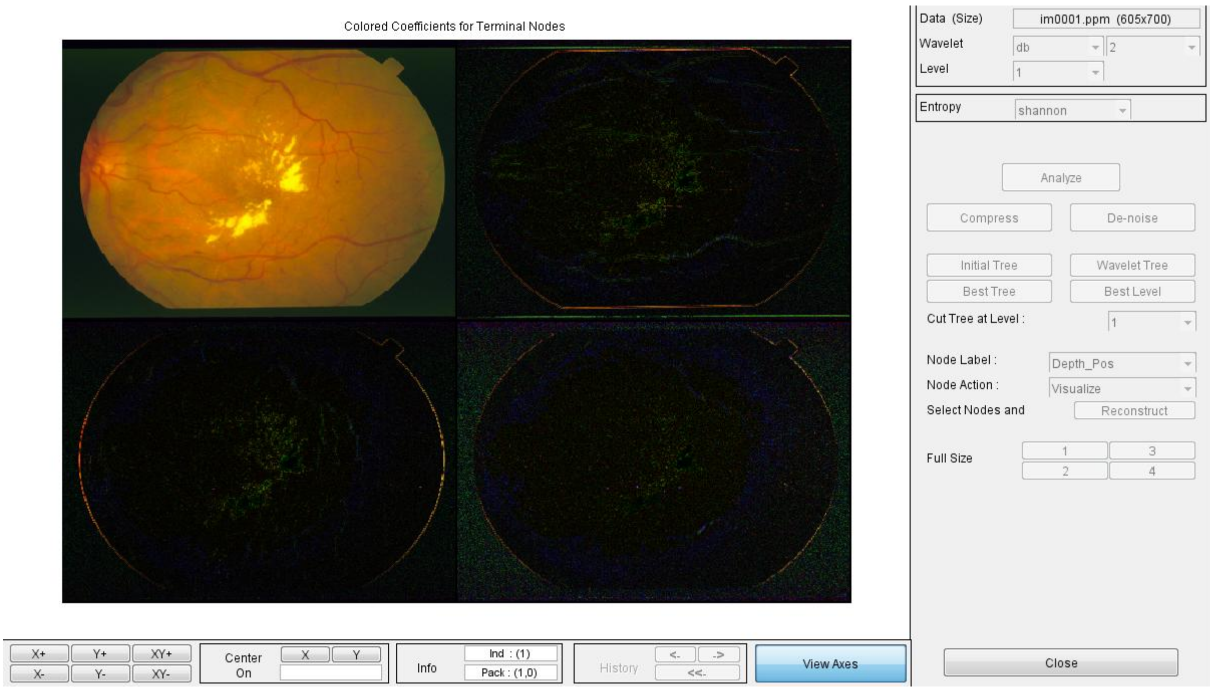

Рисунок 2 – Результат преобразования вейвлетом Добеши изображения № 1

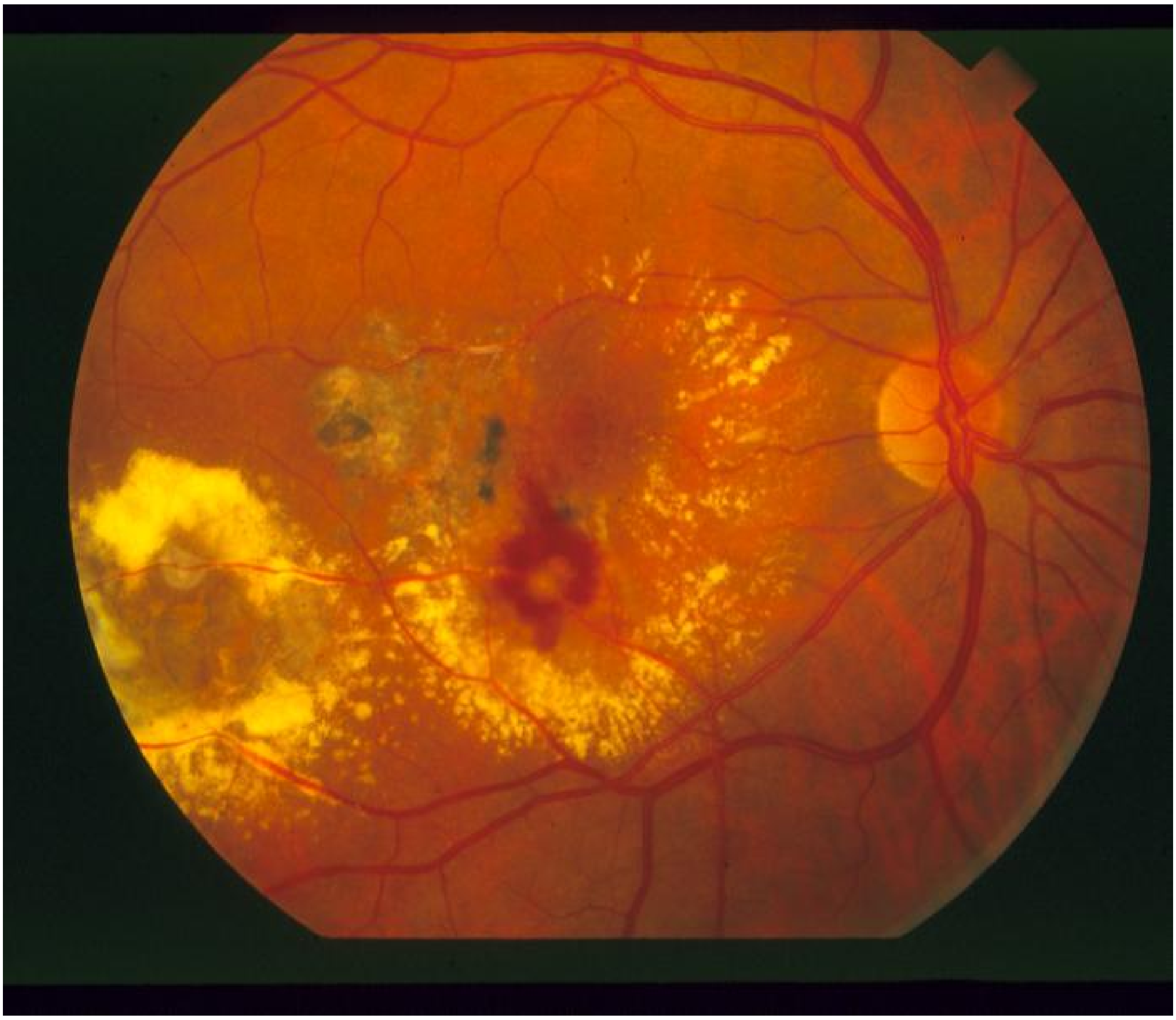

Рисунок 3 – Исходное изображение № 2

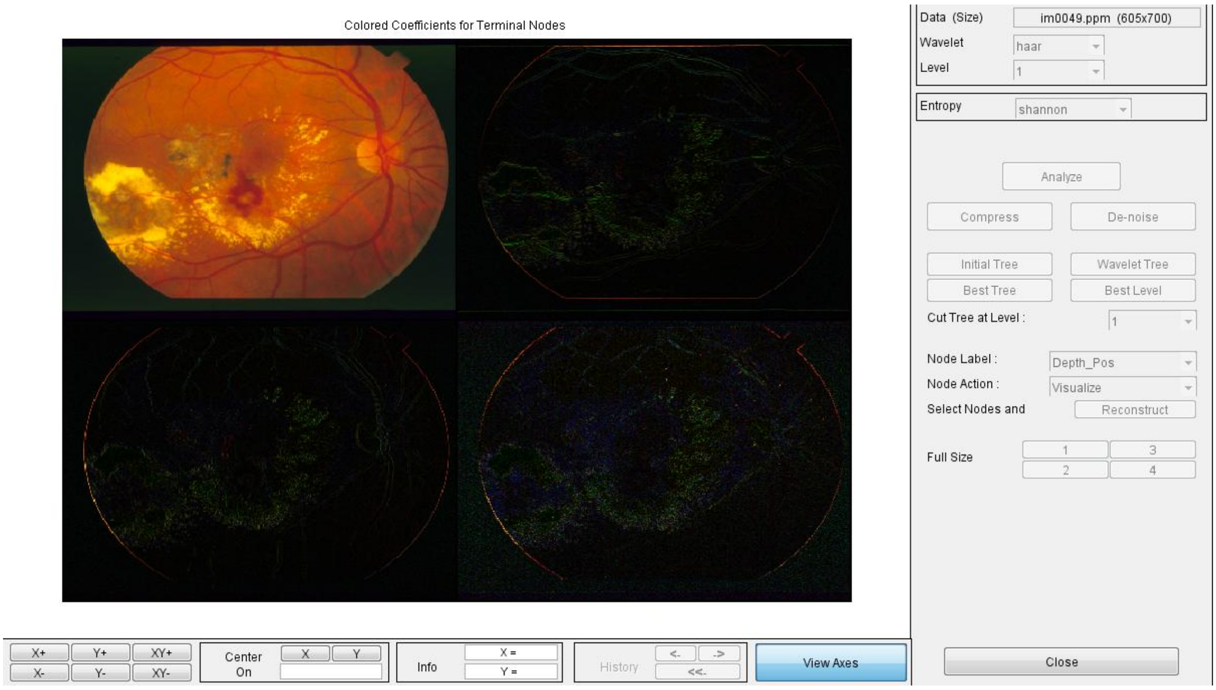

Рисунок 4 – Результат преобраобования вейвлетом Добеши изображения № 2

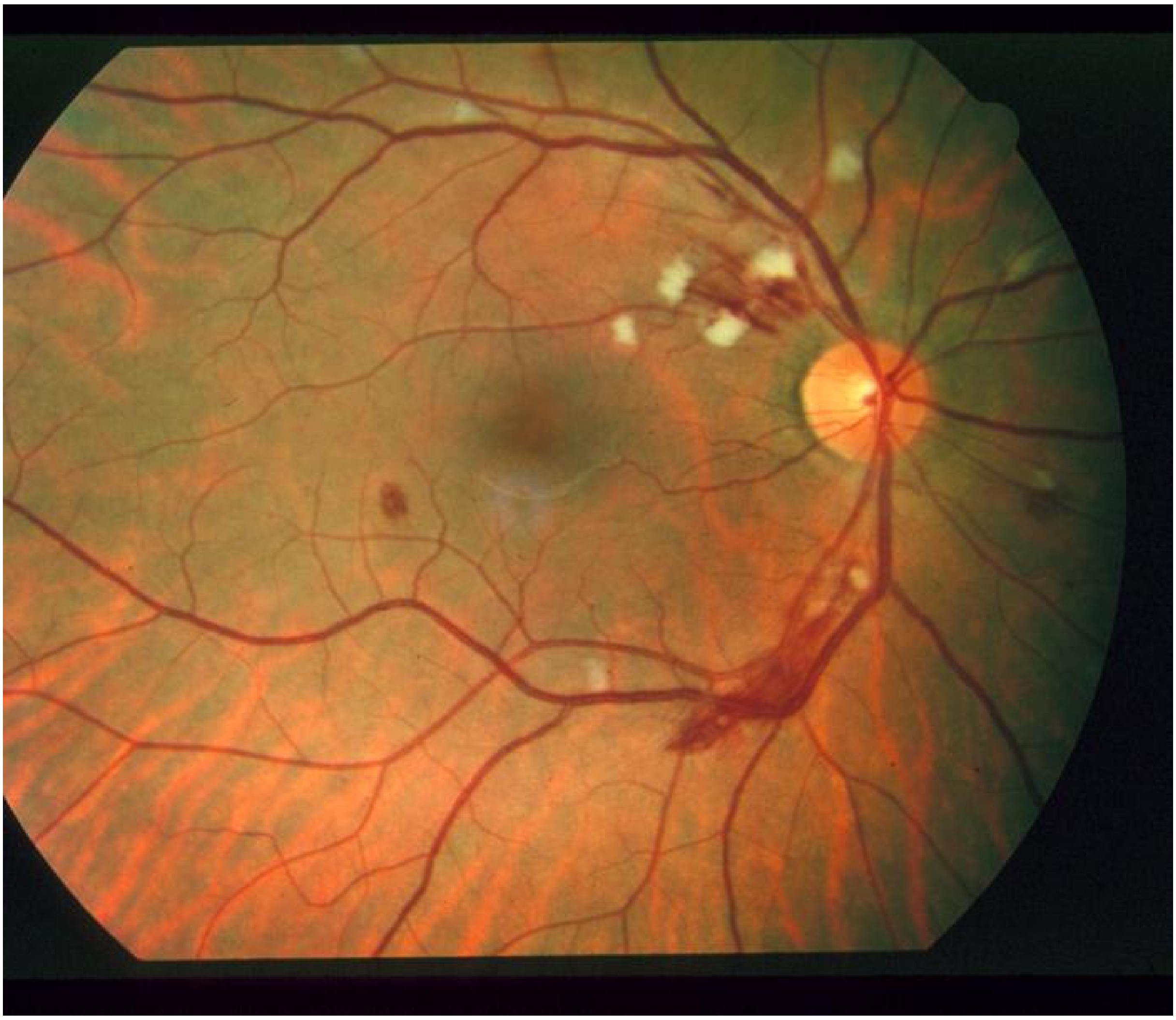

Рисунок 5 – Исходное изображение № 3

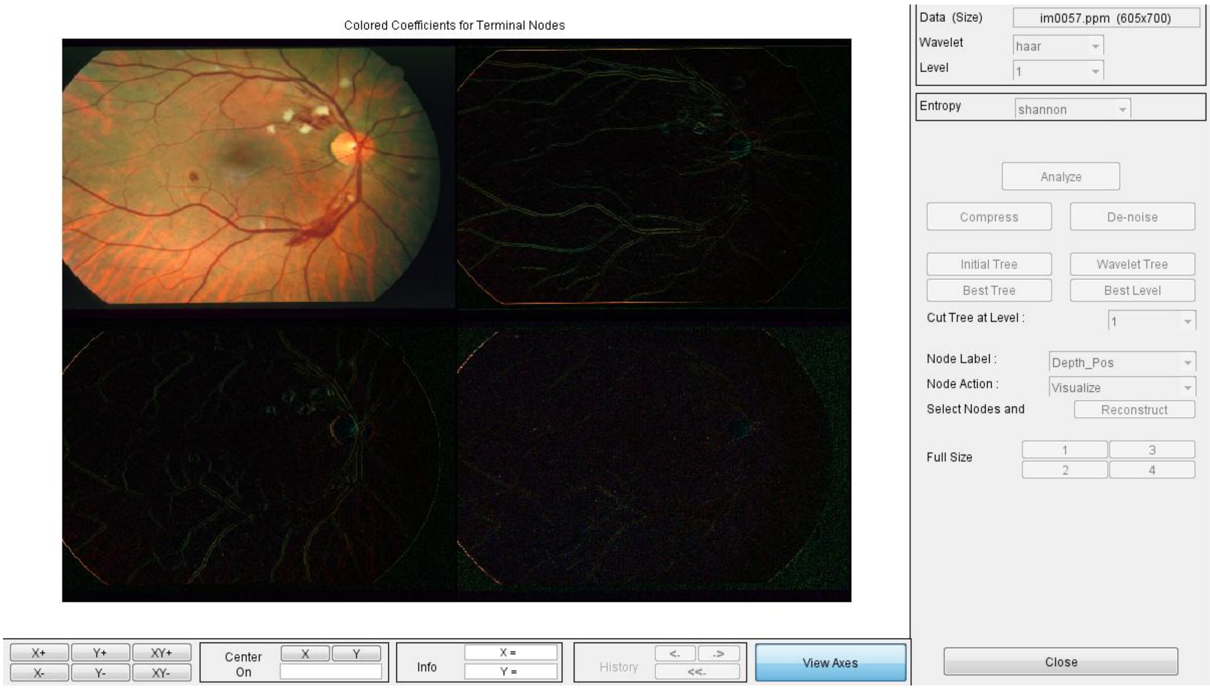

Рисунок 6 – Результат преобразования вейвлетом Добеши изображения № 3

2 СОЗДАНИЕ ПРОГРАММЫ

2.1 Дискретное преобразование

Был использован файл *dwt2d.c*, распространяющийся под лицензией *GNU General Public License* и опубликованный как *Free Software Foundation* на сайте Scilab [9].

2.2 Вейвлет Морле

С использованием файла [10], опубликованного на сайте Matlab под лицензией *BSD*, в соответствии с подразделом 1.3.2 автором на платформе *.NET* был написан модуль, реализующий двумерное преобразование Морле.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Blood vessels segmentation in nonmydriatic images using wavelets and statistical classifiers / Jorge J. G. Leandro, Joao V. B. Soares, Roberto M. Cesar Jr., Herbert F. Jelinek / Computer Graphics and Image Processing, 2003. SIBGRAPI 2003. XVI Brazilian Symposium on. — IEEE, 2003. — October. — P. 262 – 269. — URL: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1241018.

2 Вейвлет-анализ в примерах: Учебное пособие / О.В. Нагорнов, В.Г. Никитаев, В.М. Простокишин et al. — М. : НИЯУ МИФИ, 2010. — URL: http://library.mephi.ru/Data-IRBIS/book-mephi/Nagornov_Vejvlet-analiz_v_primerah_2010.pdf.

3 Дремин И.М., Иванов О.В., Нечитайло В.А. Вейвлеты и их использование // Успехи физических наук. — 2001. — Май. — Vol. 171, no. 5. — P. 465–502. — URL: https://ufn.ru/ufn01/ufn01_5/Russian/r015a.pdf.

4 Mallat S. A Wavelet Tour of Signal Processing. — 3 edition. — Academic Press, 2008.

5 Shensa M. J. The discrete wavelet transform: Wedding the à trous and Mallat algorithms, IEEE Trans. Signal Process. — 1992.

6 Antoine J-P. The Continuous Wavelet Transform in Image Processing, Institut de Physique Theorique Universite Catholique de Louvain. — Louvain-la-Neuve, Belgium, 2007.

7 Sarvaiya Jignesh N, Patnaik Dr. Suprava. Automatic Image Registration Using Mexican Hat Wavelet, Invariant Moment, and Radon Transform // International Journal of Advanced Computer Science and Applications(IJACSA), Special Issue on Image Processing and Analysis. — 2011. — URL: http://dx.doi.org/10.14569/SpecialIssue.2011.010111.

8 Демьянович Ю.К., Ходаковский В.А. Введение в теорию вейвлетов, Петербургский государственный университет путей сообщения. — Санкт-Петербург, 2007. — URL: http://www.math.spbu.ru/parallel/pdf/dh_theory.pdf.

9 Liu Roger, Nahrstaedt Holger. 2-D signal decomposition and reconstruction (version: 0.2.0). — URL: http://forge.scilab.org/index.php/p/swt/source/tree/master/src/c/dwt2d.c.

10 Cicconet Marcelo. Morlet Wavelet Kernel (version: 1.3). — 2012. — URL: http://www.mathworks.com/matlabcentral/fileexchange/37839-morlet-wavelet-kernel.

ПРИЛОЖЕНИЕ А.

ИСХОДНЫЙ КОД ПРОГРАММЫ

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Drawing;
using System.Drawing.Imaging;

namespace ScienceWork
{
    public static class MyMath
    {
        public static double two(double x)
        {
            return x + x;
        }
        public static double fo(double x)
        {
            return two(x + x);
        }
        public static double sqr(double x)
        {
            return x * x;
        }

    }
    public static class Matrix
    {
        public static double[] Multi(double[][] A, double[]
           b)
        {
            int n = A.Length;
            double[] res = new double[n];
            for (int i = 0; i < n; i++)
                res[i] = Vector.Scalar(A[i], b);
            return res;
        }
    }
    public static class Vector
    {
        public static double Scalar(double[] a, double[] b)
        {
            double res = 0.0;
```

```
            for (int i = 0; i < a.Length; i++) res += a[i] *
               b[i];
            return res;
        }
        public static double Sum(double[] a)
        {
            double res = 0.0;
            for (int i = 0; i < a.Length; i++) res += a[i];
            return res;
        }
        /// <summary>
        /// a - b
        /// </summary>
        /// <param name="a"></param>
        /// <param name="b"></param>
        /// <returns></returns>
        public static double[] Dif(double[] a, double[] b)
        {
            double[] res = new double[a.Length];
            for (int i = 0; i < a.Length; i++)
                res[i] = a[i] - b[i];
            return res;
        }
        public static double[] Del(double[] a, double b)
        {
            double oneDelB = 1.0 / b;
            double[] res = new double[a.Length];
            for (int i = 0; i < a.Length; i++)
                res[i] = a[i]*oneDelB;
            return res;
        }
    }
    public static class Integral
    {
        public static double Simpson(double[] f)
        {
            double res = 0.0;
            int n = f.Length;
            res += (f[0] + f[n]);

            double sum1 = 0.0;
            for (int i = 1; i < n - 1; i += 2) sum1 += f[i];

            double sum2 = 0.0;
            for (int i = 2; i < n - 1; i += 2) sum2 += f[i];

            res += MyMath.fo(sum1) + MyMath.two(sum2);
```

```
        res *= 0.333333;
        return res;
    }
}
public interface Wavelet
{
}
public class ValImage
{
    public double[][] f { public get; private set; }
    public int Height { public get; private set; }
    public int Widht { public get; private set; }
    void init(Bitmap bitmap)
    {
        Height = bitmap.Height;
        Widht  = bitmap.Width;
        for (int i = 0; i < Height; i++)
            for (int j = 0; j < Widht; j++)
                f[i][j] = bitmap.GetPixel(i, j).G;
    }
    public ValImage(String fileName)
    {
        init(new System.Drawing.Bitmap(fileName));
    }
}
public class Morlet
{
    class MyExp
    {
        double argRe;
        double argIm;
        double modul;
        public MyExp(double argRe, double argIm)
        {
            this.argIm = argIm; this.argRe = argRe;
            modul = Math.Exp(argRe);
        }
        public double Modul()
        {
            return modul;
        }
        public double Arg()
        {
            return argIm;
        }
        public double Re()
        {
```

```
            return modul * Math.Math.Cos(argIm);
        }
        public double Im()
        {
            return modul * Math.Math.sin(argIm);
        }
    }
    double[] k;
    double oneToSqrtEps;
    // no correct!!!
    double c_psi_norm = 1.0;
    double[][] fConstMatr;
    int n;
    int m;
    double[][] R;
    void SetR(double angle)
    {
        double cosAngle = Math.Cos(angle);
        double sinAngle = Math.Sin(angle);
        R = new double[2][];
        R[0] = new double[] { cosAngle, -sinAngle };
        R[1] = new double[] { sinAngle,  cosAngle };
    }
    public MyExp Psi(double[] x0)
    {
        double[] x = new double[2];
        x[0] = x0[0] * oneToSqrtEps; x[1] = x0[1];
        double argRe = -0.5 * Vector.Scalar(x, x);
        double argIm = Vector.Scalar(k, x);
        return new MyExp(argRe, argIm);
    }
    public double[] W(double[] b, double angle, double a)
    {
        double[] funToIntRe = new
           double[fConstMatr[0].Length];
        double[] funToIntIm = new
           double[fConstMatr[0].Length];
        for (int i = 0; i < fConstMatr.Length; i++)
        {
            for(int j = 0; j<fConstMatr[0].Length;j++)
            {
                double[] x = new double[]{i,j};
                double[] pr2 =
                   Vector.Del(Vector.Dif(x,b),a);
                SetR(angle);
                MyExp exps = Psi(Matrix.Multi(R, pr2));
                double[] psi = new double[] {exps.Re(),
```

```
                    exps.Im()};
                funToIntRe[j] = fConstMatr[i][j] *
                    psi[0];
                funToIntIm[j] = fConstMatr[i][j] *
                    psi[1];
            }

        }
        double[] ints = new double[2];
        //ints[0] = Integral.Simpson();
        //ints[1] = Integral.Simpson();
        return ints;
    }
    void setK(double accuracy)
    {
        k = new double[] { 6.0, 6.0 };
    }
    public double[] kernelCWT2D(double[] a, double[] sc,
        double theta)
    {
        int n = a.Length;
        double[] res = new double[n];

        return res;
    }
    public Morlet(String fileName, double eps, double
       accuracy)
    {
        ValImage valImage = new ValImage(fileName);
        fConstMatr = valImage.f;
        n = fConstMatr.Length;
        m = fConstMatr[0].Length;

        oneToSqrtEps = 1.0 / Math.Sqrt(eps);
        setK(accuracy);
    }
}
public class Gabor
{
    // original author:
    // Marcelo Cicconet, New York University
    // marceloc.net
    public double[,] kernelCWT2D(double scale, double
       orientation, int npeaks){
        double theta =
           -(orientation-90.0)/360.0*2.0*Math.PI;
        // controls width of gaussian window (default:
```

```
   scale)
double sigma = scale;
// controls elongation in direction
   perpendicular to wave
double gamma = 0.5;

// width and height of kernel
double support = 2.5*sigma/gamma;

// wavelength (default: 4*sigma)
double lambda = 1/npeaks*4*sigma;

// phase offset (in radians)
double psi = 0;

double xmin = -support;
double xmax = -xmin;
double ymin = xmin;
double ymax = xmax;

double[] xs = new double[(int)(xmax-xmin)+1];
double[] ys = new double[(int)(ymax-ymin)+1];

int n = xs.Length;
int m = ys.Length;

double[,] xprime = new double[m,n];
double[,] yprime = new double[m,n];
for (int i = 0; i<m; i++) for (int j = 0; j < n;
   j++){
    xprime[i,j] =  Math.Cos(theta)*xs[j] +
       Math.Sin(theta)*ys[i];
    yprime[i,j] = -Math.Sin(theta)*xs[j] +
       Math.Cos(theta)*ys[i];
}
double[,] expf = new double[m,n];
for (int i = 0; i<m; i++) for (int j = 0; j < n;
   j++){
    expf[i,j] = Math.Exp(-0.5/MyMath.sqr(sigma)*
       (MyMath.sqr(xprime[i,j])
       +MyMath.sqr(gamma*yprime[i,j])));
}
// matrixs
double[,] mr = new double[m,n];
double[,] mi = new double[m,n];
for(int i = 0; i<m; i++) for (int j =0; j<n;
   j++){
```

```
            mr[i,j] = expf[i,j] * Math.Cos( 2*Math.PI/
               lambda*xprime[i,j] +psi);
            mi[i,j] = expf[i,j] * Math.Sin( 2*Math.PI/
               lambda*xprime[i,j] +psi);
        }

        double scal1 = scalar(mr,mr)/numel(mr);
        double scal2 = scalar(mi,mi)/numel(mi);
        // mean = 0
        for (int k = 0; k < m; k++) for (int l = 0; l <
           n; l++) {
            mr[k,l] -= scal1;
            mi[k,l] -= scal2;
        }

        // norm = 1
        for (int k = 0; k < m; k++) for (int l = 0; l <
           n; l++) {
            mr[k,l] = mr[k,l] / Math.Sqrt(scalar(mr,mr));
            mi[k, l] = mi[k, l] / Math.Sqrt(scalar(mi,
               mi));
        }
        return new double[]{mr,mi};
    }
    public double scalar(double[] a, double[] b)
    {
        double sum = 0.0;
        for (int k = 0; k < a.Length; k++)
            sum += a[k] * b[k];
        return sum;
    }
    double numel(double[,] A)
    {
        return (double)(A.GetLength(2));
    }
    double scalar(double[,] A, double[,] B)
    {
        double sum = 0.0;
        for (int k = 0; k < A.Length; k++)
        {
            double sum2 = 0.0;
            for (int k = 0; k < a.Length; k++)
                sum += a[k] * b[k];
            sum += sum2;
        }
        return sum;
    }
```

```
    }

    class Program
    {
        static void Main(string[] args)
        {
        }
    }
}
```

Листинг 1 – a test for a C$^\sharp$ code