ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Факультет Информационных технологий
Кафедра Информатики и информационных технологий

направление подготовки
09.03.02 «Информационные системы и технологии»

ЛАБОРАТОРНАЯ РАБОТА № 3

Дисциплина: «Распознавание образов в информационных и автоматизированных системах копия 1»

Тема: «Аффинные преобразования и гомография изображений»

Выполнил: студент группы 211-723

Сергеев Станислав Олегович

Дата, подпись________________ ____________
(Дата) (Подпись)

Проверил:__________________________ ____________
(Фамилия И.О., степень, звание) **(Оценка)**

Дата, подпись________________ ____________
(Дата) (Подпись)

Замечания: ____

____

____

Москва

2022

Аффинные преобразования и гомография изображений.

Цель:

Целью данной работы является изучение базовых операций над цветовыми каналами изображений и реализация некоторых фильтров на их основе.

Постановка задачи:

Необходимо разработать приложение Windows Forms, способное осуществлять:
1. загрузку и отображение двух изображений по выбору пользователя;
2. возможность применения аффинных преобразований к загруженным изображениям;
3. возможность проекции области одного изображения на другое.

Листинг программы

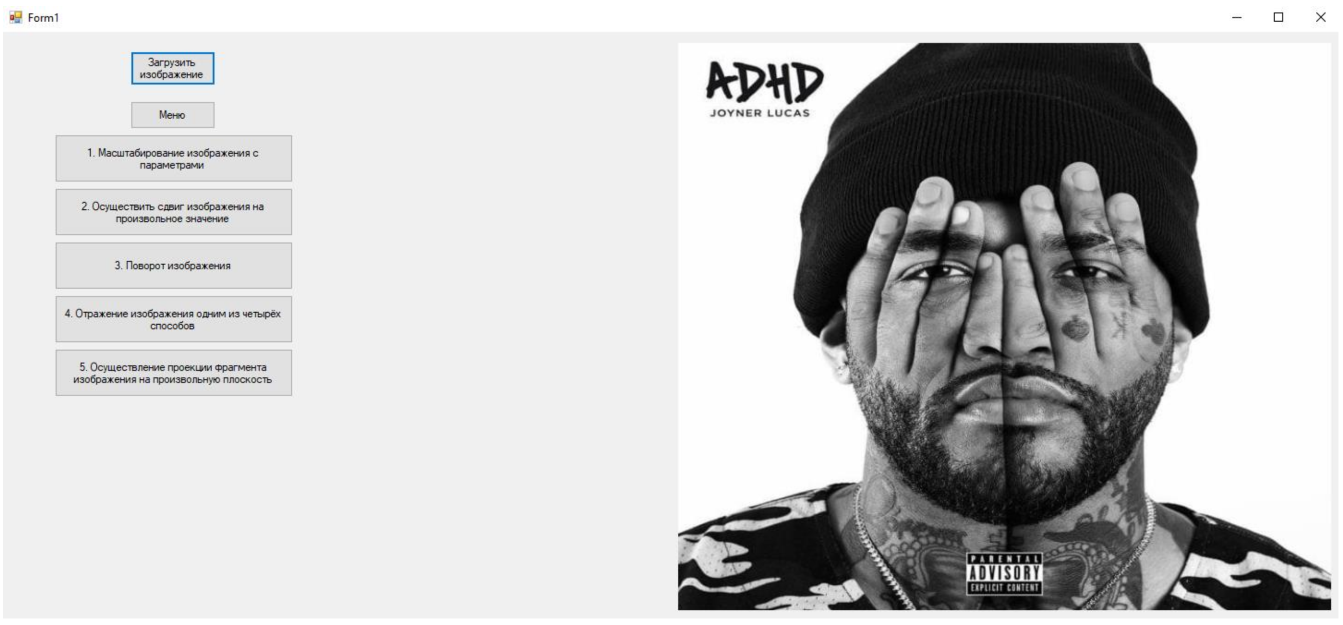
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using Emgu.CV;
using Emgu.CV.CvEnum;
using Emgu.CV.Structure;
using Emgu.CV.Util;

using Emgu.CV.ML.MlEnum;
using System.Security.Cryptography.X509Certificates;
using System.Runtime.CompilerServices;

namespace _3
{
    public partial class Form1 : Form
    {
        private Image<Bgr, byte> sourceImage = null, Redact_Image = null;
        double fkoefX, fkoefY, shearingKoef;
        List<Cordinate> src = new List<Cordinate>();
        public struct Cordinate
        {
            int X, Y;
            public Cordinate(int x, int y)
            {
                X = x;
                Y = y;
    }
    public void Set_Cord(int x, int y)
    {

        X = x;
        Y = y;
    }
    public int Get_X()
    {
        return X;
    }
    public int Get_Y()
    {
        return Y;
    }
};
public Form1()
{
    InitializeComponent();
    imageBox1.MouseClick += new MouseEventHandler(imageBox1_MouseClick);
}
private void ActivateDisabledButtons()
{
    convertFor1.Enabled = true;
    koefX.Enabled = true;
    koefY.Enabled = true;
    fkoefX = 1.0;
    fkoefY = 1.0;
    shearingKoef = 0.0;
    koefX.Text = fkoefX.ToString();
    koefY.Text = fkoefY.ToString();
    koefXminus.Enabled = true;
    koefXplus.Enabled = true;
    koefYminus.Enabled = true;
    koefYplus.Enabled = true;
    KforShearing.Enabled = true;
    shearingKplus.Enabled = true;
    convertFor2.Enabled = true;
    nudForXforTurn.Enabled = true;
    nudForYforTurn.Enabled = true;
    nudForAngelForTurn.Enabled = true;
    convertFor3.Enabled = true;
    nudForXforTurn.Maximum = sourceImage.Width;
    nudForYforTurn.Maximum = sourceImage.Height;
    nudForAngelForTurn.Maximum = 360;
    convertFor4gor.Enabled = true;
    convertFor4vert.Enabled = true;
    convertFor4gorvert.Enabled = true;
    convertFor6.Enabled = true;
}
private Image<Bgr, byte> loadImage()
{
    OpenFileDialog openFileDialog = new OpenFileDialog();
    var result = openFileDialog.ShowDialog(); // открытие диалога выбора файла
    if (result == DialogResult.OK) // открытие выбранного файла
    {
        string fileName = openFileDialog.FileName;
        Image<Bgr, byte> _sourceImage = new Image<Bgr, byte>(fileName);
        Redact_Image = _sourceImage.Resize(707, 614, Inter.Linear); ;
        return _sourceImage;
    }
```

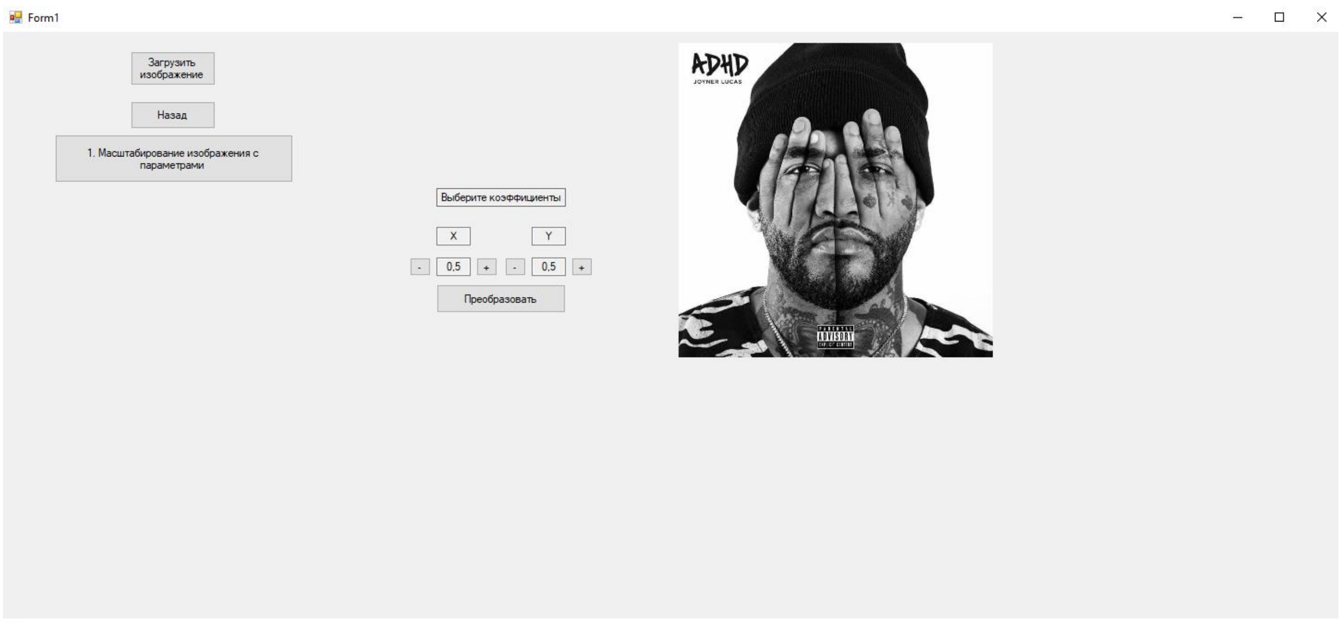
```
        return null;
    }
    private void goToMenu(Button but)
    {
        menu.Visible = false;
        menu1.Visible = false;
        menu2.Visible = false;
        menu3.Visible = false;
        menu4.Visible = false;
        menu6.Visible = false;
        back.Visible = true;
        back.Visible = true;
        if (but == menu1)
            openMenu1();
        if (but == menu2)
            openMenu2();
        if (but == menu3)
            openMenu3();
        if (but == menu4)
            openMenu4();
        if (but == menu6)
            openMenu6();
    }
    private void backToMenu()
    {
        if (menu1.Visible == true)
            closeMenu1();
        if (menu2.Visible == true)
            closeMenu2();
        if (menu3.Visible == true)
            closeMenu3();
        if (menu4.Visible == true)
            closeMenu4();
        if (menu6.Visible == true)
            closeMenu6();
        menu.Visible = true;
        menu1.Visible = true;
        menu2.Visible = true;
        menu3.Visible = true;
        menu4.Visible = true;
        menu6.Visible = true;
        back.Visible = false;
    }
    private void openMenu1()
    {
        menu1.Location = new Point(58, 111);
        menu1.Visible = true;
        back.Visible = true;
        chooseXY.Visible = true;
        koefForX.Visible = true;
        koefForY.Visible = true;
        koefX.Visible = true;
        koefY.Visible = true;
        convertFor1.Visible = true;
        koefXminus.Visible = true;
        koefXplus.Visible = true;
        koefYminus.Visible = true;
        koefYplus.Visible = true;
        if (imageBox1.Image != null)
            imageBox1.Image = sourceImage;
    }
```

```
private void closeMenu1()
{
    menu1.Location = new Point(58, 111);
    chooseXY.Visible = false;
    koefForX.Visible = false;
    koefForY.Visible = false;
    koefX.Visible = false;
    koefY.Visible = false;
    convertFor1.Visible = false;
    koefXminus.Visible = false;
    koefXplus.Visible = false;
    koefYminus.Visible = false;
    koefYplus.Visible = false;
}
private void openMenu2()
{
    menu2.Location = new Point(58, 111);
    menu2.Visible = true;
    back.Visible = true;
    chooseKforSharing.Visible = true;
    if (imageBox1.Image != null)
        imageBox1.Image = sourceImage.Resize(707, 614, Inter.Linear);
    KforShearing.Visible = true;
    shearingKminus.Visible = true;
    shearingKplus.Visible = true;
    convertFor2.Visible = true;
    shearingKminus.Enabled = false;
    KforShearing.Text = shearingKoef.ToString();
}
private void closeMenu2()
{
    menu2.Location = new Point(58, 169);
    chooseKforSharing.Visible = false;
    KforShearing.Visible = false;
    shearingKminus.Visible = false;
    shearingKplus.Visible = false;
    convertFor2.Visible = false;
}
private void openMenu3()
{
    menu3.Location = new Point(58, 111);
    menu3.Visible = true;
    back.Visible = true;
    if (sourceImage != null)
        imageBox1.Image = sourceImage.Resize(707, 614, Inter.Linear);
    chooseXYandAngle.Visible = true;
    XforTurn.Visible = true;
    YforTurn.Visible = true;
    angelForTurn.Visible = true;
    nudForXforTurn.Visible = true;
    nudForYforTurn.Visible = true;
    nudForAngelForTurn.Visible = true;
    convertFor3.Visible = true;
}
private void closeMenu3()
{
    menu3.Location = new Point(58, 227);
    chooseXYandAngle.Visible = false;
    XforTurn.Visible = false;
    YforTurn.Visible = false;
    angelForTurn.Visible = false;
    nudForXforTurn.Visible = false;
    nudForYforTurn.Visible = false;
    nudForAngelForTurn.Visible = false;
    convertFor3.Visible = false;
}
private void openMenu4()
{
    menu4.Location = new Point(58, 111);
    menu4.Visible = true;
    back.Visible = true;
    if (sourceImage != null)
        imageBox1.Image = sourceImage.Resize(707, 614, Inter.Linear);
    chooseReflection.Visible = true;
    convertFor4gor.Visible = true;
    convertFor4vert.Visible = true;
    convertFor4gorvert.Visible = true;
}
private void closeMenu4()
{
    menu4.Location = new Point(58, 285);
    chooseReflection.Visible = false;
    convertFor4gor.Visible = false;
    convertFor4vert.Visible = false;
    convertFor4gorvert.Visible = false;
}
private void openMenu6()
{
    menu6.Location = new Point(58, 111);
    menu6.Visible = true;
    back.Visible = true;
    if (sourceImage != null)
        imageBox1.Image = Redact_Image.Resize(707, 614, Inter.Linear);
    convertFor6.Visible = true;
}
private void closeMenu6()
{
    menu6.Location = new Point(58, 343);
    convertFor6.Visible = false;
}
private void zagruzitImage_Click(object sender, EventArgs e)
{
    sourceImage = loadImage();
    if (sourceImage != null)
    {
        ActivateDisabledButtons();
        imageBox1.Image = sourceImage.Resize(707, 614, Inter.Linear);
    }
}
private void menu1_Click(object sender, EventArgs e)
{
    goToMenu(menu1);
}
private void menu2_Click(object sender, EventArgs e)
{
    goToMenu(menu2);
}
private void menu3_Click(object sender, EventArgs e)
{
    goToMenu(menu3);
}
private void menu4_Click(object sender, EventArgs e)
```

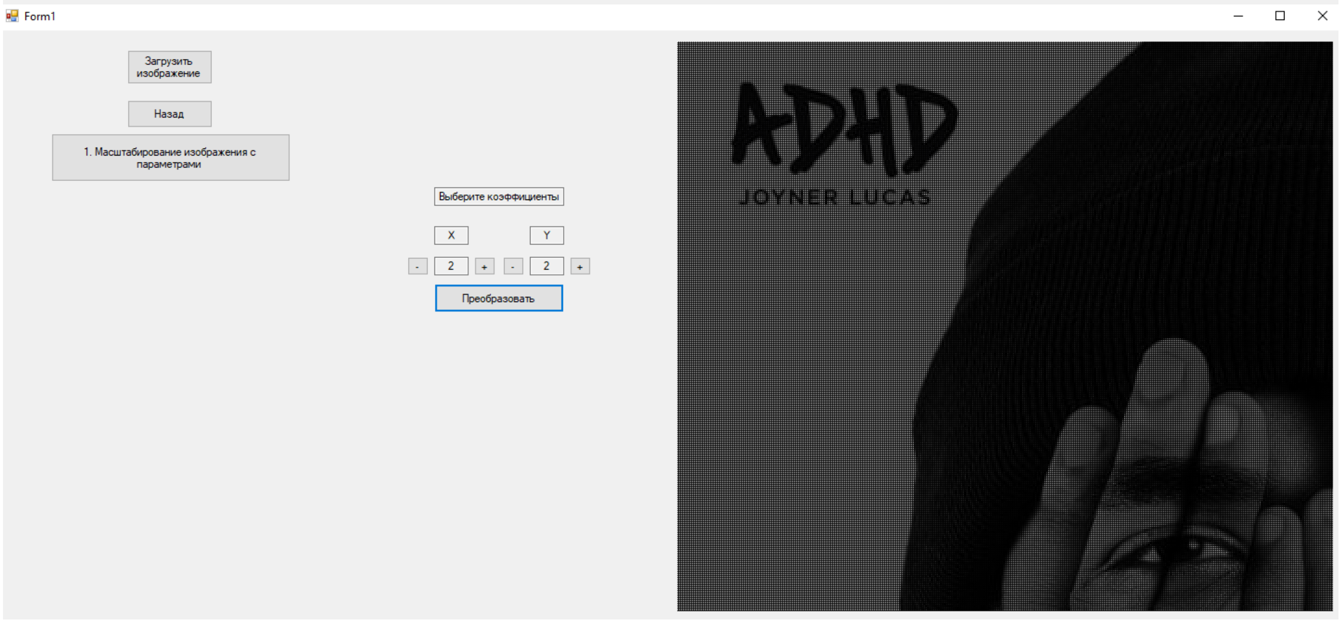
```
        {
            goToMenu(menu4);
        }
        private void menu6_Click(object sender, EventArgs e)
        {
            goToMenu(menu6);
        }
        private void back_Click(object sender, EventArgs e)
        {
            backToMenu();
        }
        private Image<Bgr, byte> ChangeXY(double sX, double sY, Image<Bgr, byte> _sourceImage)
        {
            var newImage = new Image<Bgr, byte>((int)(_sourceImage.Width * sX), (int)(_sourceImage.Height * 
sY));
            for (int x = 0; x < _sourceImage.Width; x++)
            {
                for (int y = 0; y < _sourceImage.Height; y++)
                {
                    // вычисление новых координат пикселя
                    int newX = (int)(x * sX);
                    int newY = (int)(y * sY);
                    // копирование пикселя в новое изображение
                    newImage[newY, newX] = _sourceImage[y, x];
                }
            }
            return newImage;
        }
        Image<Bgr, byte> reflection(Image<Bgr, byte> _sourceImage, int qX, int qY)
        {
            var resultImage = new Image<Bgr, byte>(_sourceImage.Width + 1, _sourceImage.Height + 1);
            int newX = 0, newY = 0;
            for (int x = 0; x < _sourceImage.Width; x++)
            {
                for (int y = 0; y < _sourceImage.Height; y++)
                {
                    if (qX == -1)
                        newX = x * qX + _sourceImage.Width;
                    else
                        newX = x * qX;
                    if (qY == -1)
                        newY = y * qY + _sourceImage.Height;
                    else
                    newY = y * qY;
                    resultImage[newY, newX] = _sourceImage[y, x];
                }
            }
            return resultImage.Resize(458, 414, Inter.Linear);
        }
        private void ProverkakoefsXY()
        {
            if (fkoefX < 0.15)
                koefXminus.Enabled = false;
            else
                koefXminus.Enabled = true;
            if (fkoefX > 4.95)
                koefXplus.Enabled = false;
            else
                koefXplus.Enabled = true;

            if (fkoefY < 0.15)
        koefYminus.Enabled = false;
    else
        koefYminus.Enabled = true;
    if (fkoefY > 4.95)
        koefYplus.Enabled = false;
    else
        koefYplus.Enabled = true;
}
private void koefXminus_Click(object sender, EventArgs e)
{
    fkoefX -= 0.1;
    koefX.Text = fkoefX.ToString();
    ProverkakoefsXY();
}
private void koefXplus_Click(object sender, EventArgs e)
{
    fkoefX += 0.1;
    koefX.Text = fkoefX.ToString();
    ProverkakoefsXY();
}
private void koefYminus_Click(object sender, EventArgs e)
{
    fkoefY -= 0.1;
    koefY.Text = fkoefY.ToString();
    ProverkakoefsXY();
}
private void koefYplus_Click(object sender, EventArgs e)
{
    fkoefY += 0.1;
    koefY.Text = fkoefY.ToString();
    ProverkakoefsXY();
}
private void convertFor2_Click(object sender, EventArgs e)
{
    imageBox1.Image = Shearing(shearingKoef, sourceImage).Resize(707, 614, Inter.Linear);
}
private void ProverkakoefShearing()
{
    if (shearingKoef < 0.15)
    {
        shearingKminus.Enabled = false;
        shearingKoef = 0.0;
    }
    else
        shearingKminus.Enabled = true;
    if (shearingKoef > 2.9)
    {
        shearingKplus.Enabled = false;
        shearingKoef = 3.0;
    }
    else
        shearingKplus.Enabled = true;
}
private void shearingKminus_Click(object sender, EventArgs e)
{
    shearingKoef -= 0.2;
    KforShearing.Text = shearingKoef.ToString();
    ProverkakoefShearing();
}
private void shearingKplus_Click(object sender, EventArgs e)
{
```

```
            shearingKoef += 0.2;
            KforShearing.Text = shearingKoef.ToString();
            ProverkakoefShearing();
        }
        private void convertFor1_Click(object sender, EventArgs e)
        {

            imageBox1.Image = ChangeXY(fkoefX, fkoefY, sourceImage);
        }
        private Image<Bgr, byte> Shearing(double shift, Image<Bgr, byte> _sourceImage)
        {
            _sourceImage = _sourceImage.Resize(707, 614, Inter.Linear);
            var newImage = new Image<Bgr, byte>(_sourceImage.Size);

            for (int x = 0; x < _sourceImage.Width; x++)
            {
                for (int y = 0; y < _sourceImage.Height; y++)
                {
                    int newX;
                    if ((int)(x + shift * (_sourceImage.Height - y)) >= _sourceImage.Width)
                    {
                        newX = _sourceImage.Width - 1;
                    }
                    else
                    {
                        newX = (int)(x + shift * (_sourceImage.Height - y));
                    }

                    int newY = (int)y;
                    // копирование пикселя в новое изображение
                    newImage[newY, newX] = _sourceImage[y, x];
                }
            }
            return newImage;
        }
        private void convertFor4gor_Click(object sender, EventArgs e)
        {
            imageBox1.Image = reflection(sourceImage, -1, 1).Resize(707, 614, Inter.Linear);
        }
        private void convertFor4vert_Click(object sender, EventArgs e)
        {
            imageBox1.Image = reflection(sourceImage, 1, -1).Resize(707, 614, Inter.Linear);
        }
        private void convertFor4gorvert_Click(object sender, EventArgs e)
        {
            imageBox1.Image = reflection(sourceImage, -1, -1).Resize(707, 614, Inter.Linear);
        }
        private void convertFor3_Click(object sender, EventArgs e)
        {
            imageBox1.Image = turn(sourceImage, Decimal.ToDouble(nudForAngelForTurn.Value),
Decimal.ToInt32(nudForXforTurn.Value), Decimal.ToInt32(nudForYforTurn.Value)).Resize(707, 614,
Inter.Linear);
        }
        byte blineChannel(Image<Bgr, byte> Origin_Image, double X_old, double Y_old, int channel)
        {
            double FloorX, FloorY, ratioX, ratioY, inversX, inversY, CeilX, CeilY;
            FloorX = Math.Floor(X_old);
            FloorY = Math.Floor(Y_old);
            CeilX = Math.Min(Math.Ceiling(X_old), Origin_Image.Width - 1);
            CeilY = Math.Min(Math.Ceiling(Y_old), Origin_Image.Height - 1);
            ratioX = X_old - FloorX; ratioY = Y_old - FloorY;
        inversX = CeilX - X_old; inversY = CeilY - Y_old;
        byte v1, v2, v3, v4;
        double q1, q2, q;
        if ((CeilX == FloorX) && (CeilY == FloorY))
        {
            q = Origin_Image.Data[(int)(Y_old), (int)(X_old), channel];
        }
        else if (CeilX == FloorX)
        {
            q1 = Origin_Image.Data[(int)(FloorY), (int)(X_old), channel];

            q2 = Origin_Image.Data[(int)(CeilY), (int)X_old, channel];

            q = q1 * inversY + q2 * ratioY;
        }
        else if (CeilY == FloorY)
        {
            q1 = Origin_Image.Data[(int)Y_old, (int)FloorX, channel];

            q2 = Origin_Image.Data[(int)Y_old, (int)CeilX, channel];

            q = (q1 * inversX) + (q2 * ratioX);
        }
        else
        {
            v1 = Origin_Image.Data[(int)(FloorY), (int)(FloorX), channel];
            v2 = Origin_Image.Data[(int)(FloorY), (int)(CeilX), channel];
            v3 = Origin_Image.Data[(int)(CeilY), (int)(FloorX), channel];
            v4 = Origin_Image.Data[(int)(CeilY), (int)(CeilX), channel];
            q1 = v1 * inversX + v2 * ratioX;
            q2 = v3 * inversX + v4 * ratioX;
            q = q1 * inversY + q2 * ratioY;
        }
        return Convert.ToByte(q);
    }
    private void convertFor6_Click(object sender, EventArgs e)
    {
        fdf();
    }
    Image<Bgr, byte> turn(Image<Bgr, byte> _sourceImage, double angle, int Xc, int Yc)
    {
        var resultImage = new Image<Bgr, byte>(_sourceImage.Width, _sourceImage.Height);
        double radians = (angle * Math.PI) / 180;
        double oldX, oldY;
        for (int newX = 0; newX < resultImage.Width; newX++)
        {
            for (int newY = 0; newY < resultImage.Height; newY++)
            {
                oldY = (newY - Yc) * Math.Cos(radians) - (newX - Xc) * Math.Sin(radians) + Xc;
                oldX = (newY - Yc) * Math.Sin(radians) + (newX - Xc) * Math.Cos(radians) + Yc;
                if (oldX >= 0 && oldX < _sourceImage.Width && oldY >= 0 && oldY < _sourceImage.Height)
                {
                    resultImage.Data[newY, newX, 0] = blineChannel(_sourceImage, oldX, oldY, 0);
                    resultImage.Data[newY, newX, 1] = blineChannel(_sourceImage, oldX, oldY, 1);
                    resultImage.Data[newY, newX, 2] = blineChannel(_sourceImage, oldX, oldY, 2);
                }
                else
                {
                    continue;
                }
            }
            }
            return resultImage;
        }
        private void imageBox1_MouseClick(object sender, MouseEventArgs e)
        {
            if (src.Count < 4)
            {
                int x = (int)(e.Location.X / imageBox1.ZoomScale);
                int y = (int)(e.Location.Y / imageBox1.ZoomScale);
                Cordinate crd = new Cordinate();
                crd.Set_Cord(x, y);
                src.Add(crd);
                Point center = new Point(x, y);
                int radius = 2;
                int thickness = 2;
                var color = new Bgr(Color.Blue).MCvScalar;
                CvInvoke.Circle(Redact_Image, center, radius, color, thickness);
                imageBox1.Image = Redact_Image.Resize(707, 614, Inter.Linear);
            }
        }
        private void fdf()
        {
            var srcPoints = new PointF[]
                    {

                        new PointF(src[0].Get_X(), src[0].Get_Y()),
                        new PointF(src[1].Get_X(), src[1].Get_Y()),
                        new PointF(src[2].Get_X(), src[2].Get_Y()),
                        new PointF(src[3].Get_X(), src[3].Get_Y()),

                    };
            var destPoints = new PointF[]
            {
                        new PointF(0, 0),
                        new PointF(0, sourceImage.Height - 1),
                        new PointF(sourceImage.Width - 1, sourceImage.Height - 1),
                        new PointF(sourceImage.Width - 1, 0),

            };
            var homographyMatrix = CvInvoke.GetPerspectiveTransform(srcPoints, destPoints);
            var destImage = new Image<Bgr, byte>(sourceImage.Size);
            CvInvoke.WarpPerspective(sourceImage, destImage, homographyMatrix, destImage.Size);
            imageBox1.Image = destImage;
            src.Clear();
            Redact_Image = new Image<Bgr, byte>(sourceImage.Size);
            for (int y = 0; y < sourceImage.Height; y++)
                for (int x = 0; x < sourceImage.Width; x++)
                    Redact_Image[y, x] = sourceImage[y, x];
        }
    }
}
```

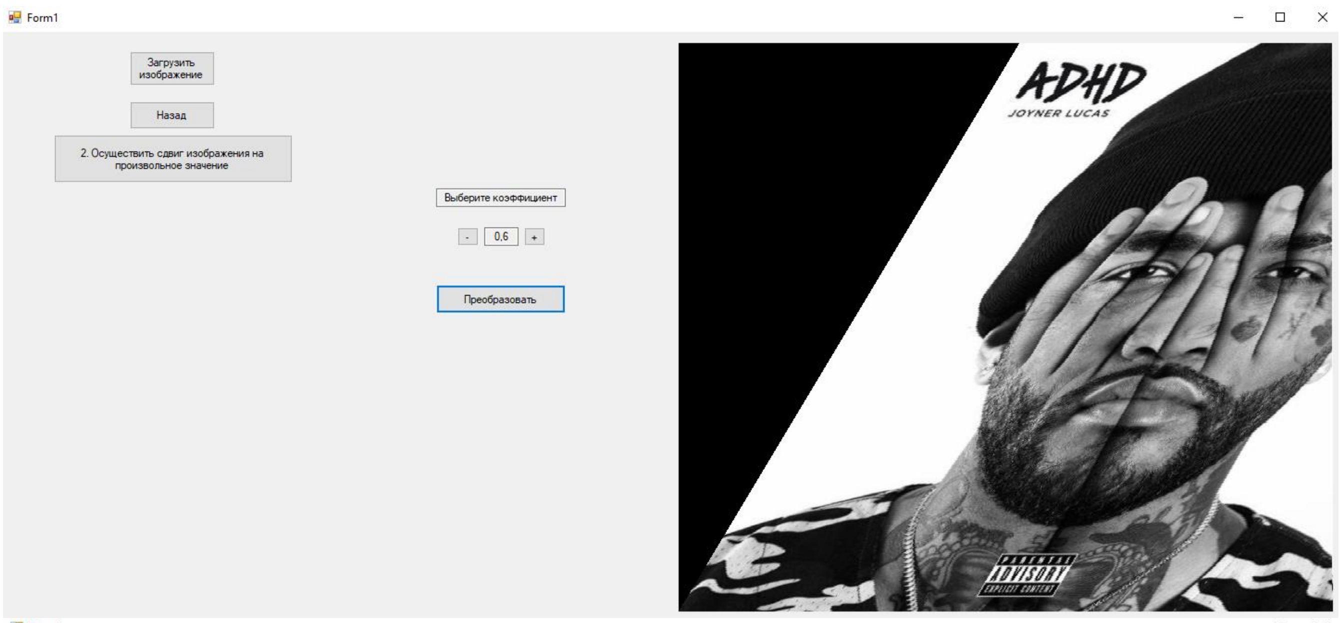

Form1
Загрузить изображение
Меню
1. Масштабирование изображения с параметрами
2. Осуществить сдвиг изображения на произвольное значение
3. Поворот изображения
4. Отражение изображения одним из четырёх способов
5. Осуществление проекции фрагмента изображения на произвольную плоскость
ADHD
JOYNER LUCAS
PARENTAL ADVISORY EXPLICIT CONTENT

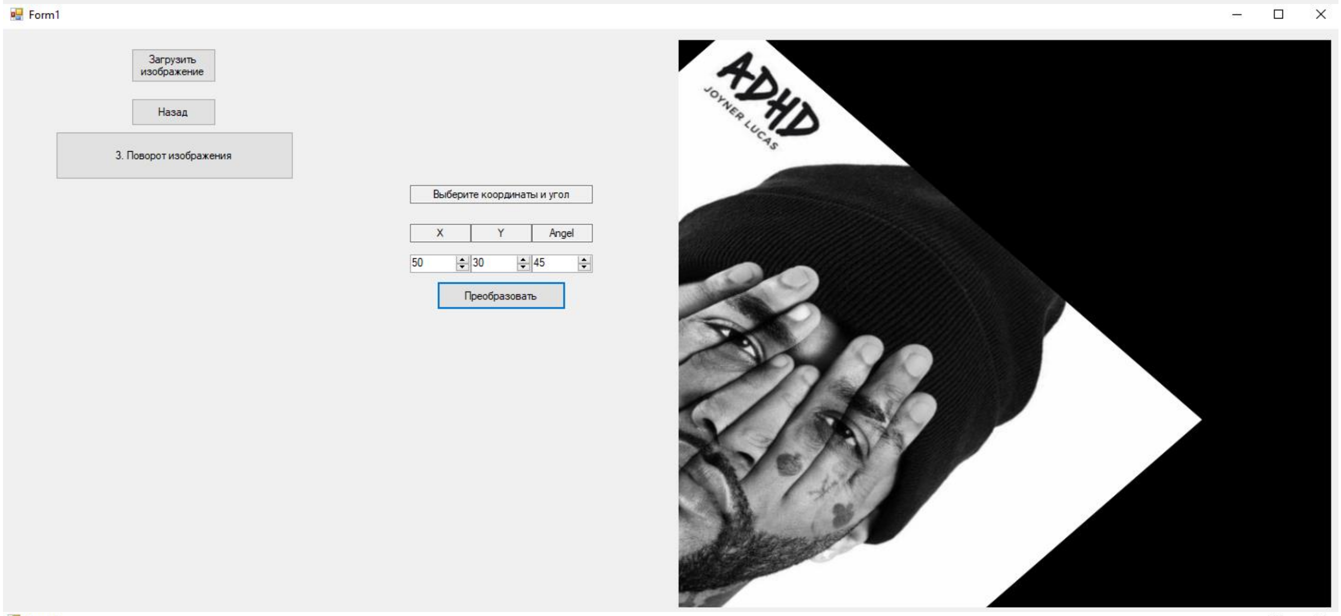

Form1
Загрузить изображение
Назад
1. Масштабирование изображения с параметрами
Выберите коэффициенты
-
+
-
+
Преобразовать
ADHD
JOYNER LUCAS

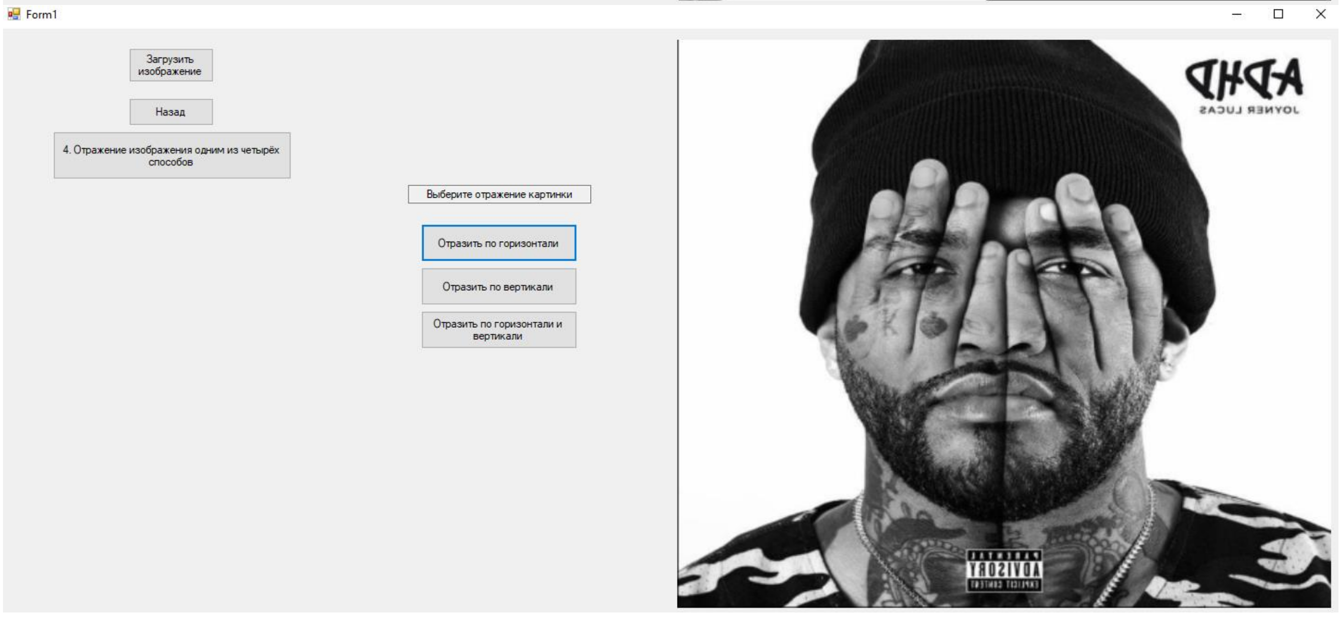
X 0,5
Y 0,5

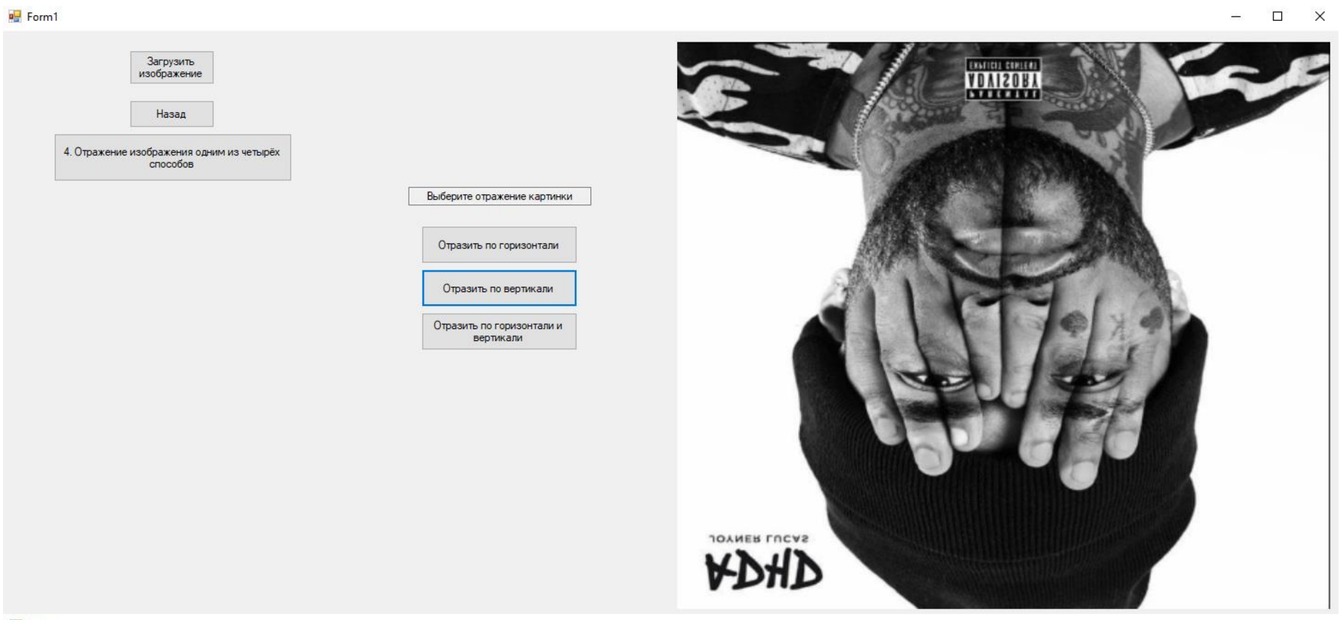

Form1
Загрузить изображение
Назад
1. Масштабирование изображения с параметрами
Выберите коэффициенты
-
+
-
+
Преобразовать
ADHD
JOYNER LUCAS

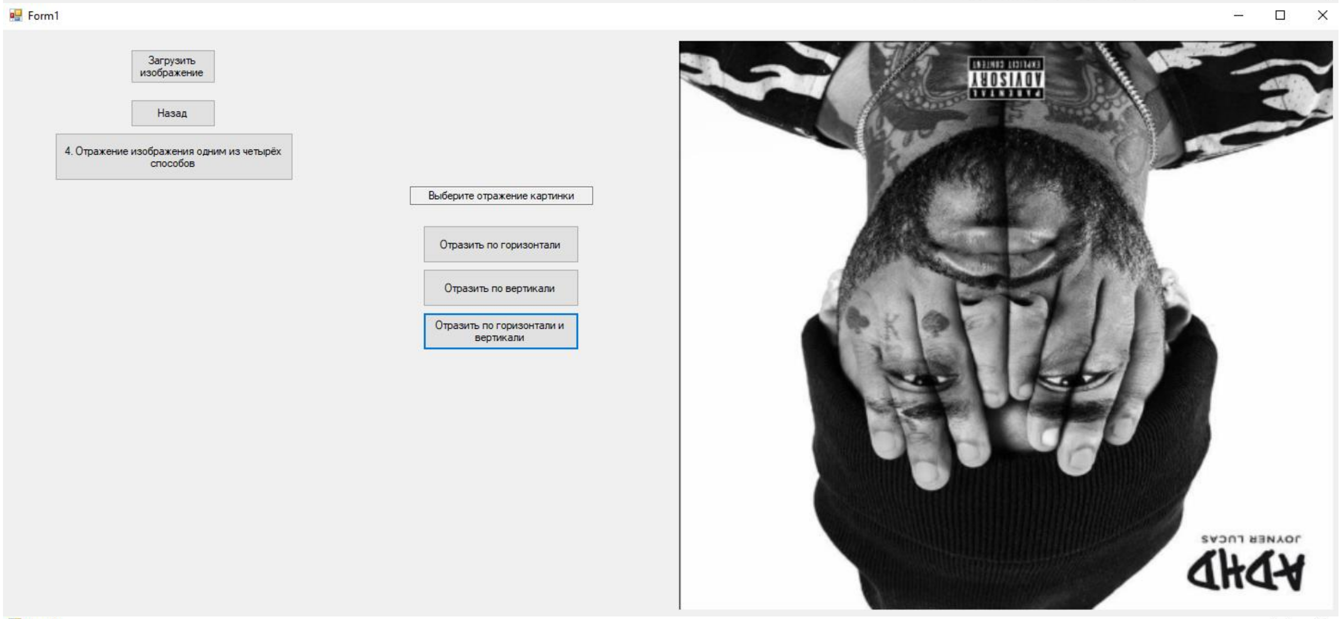
X 2
Y 2

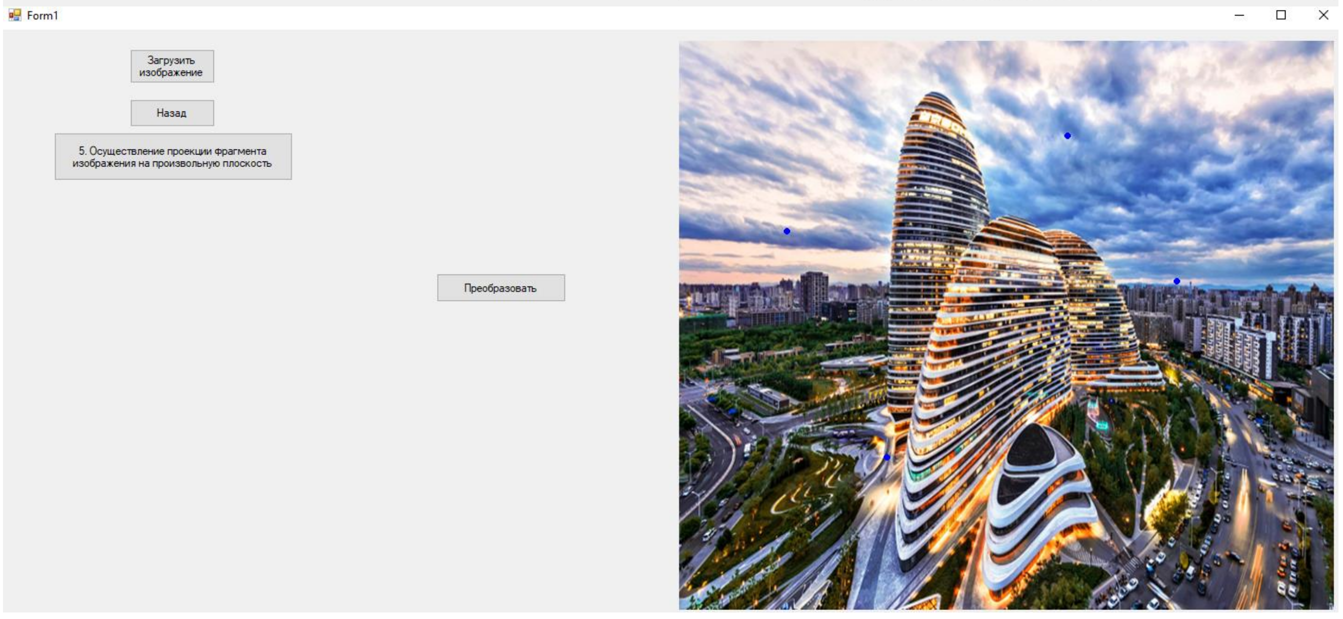

Form1
Загрузить изображение
Назад
2. Осуществить сдвиг изображения на произвольное значение
Выберите коэффициент
-
0,6
+
Преобразовать
ADHD
JOYNER LUCAS

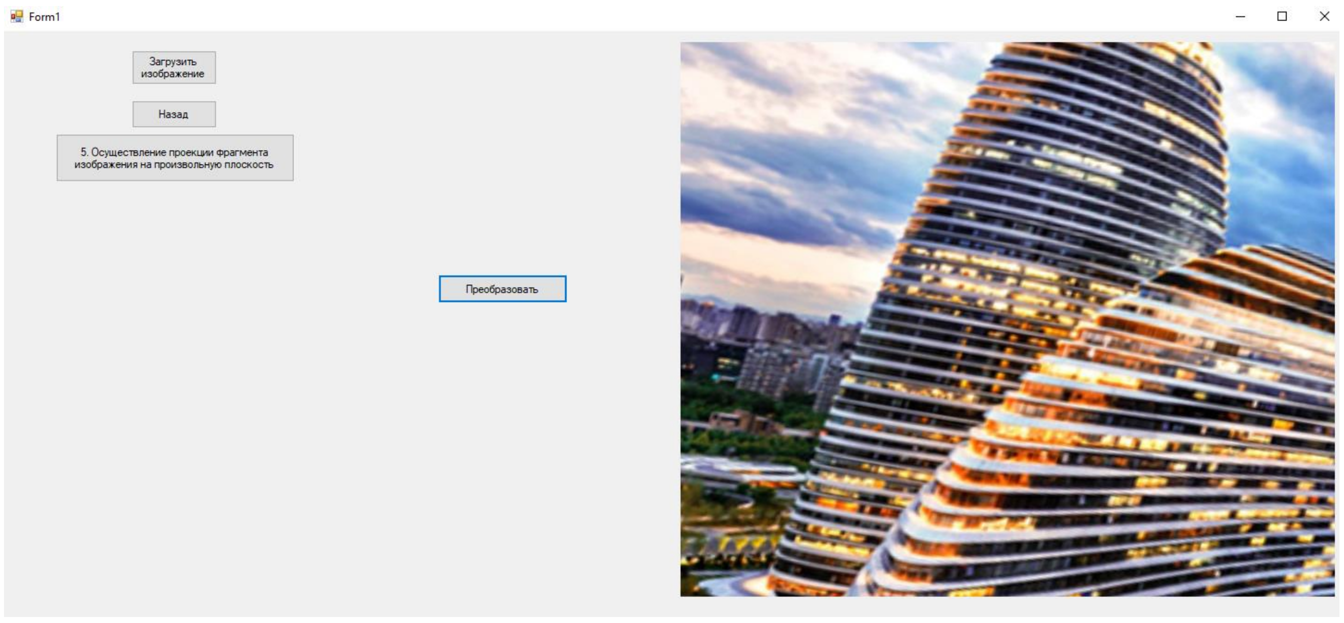

Form1
Загрузить изображение
Назад
3. Поворот изображения
Выберите координаты и угол
X
Y
Angel
50
30
45
Преобразовать
ADHD
JOYNER LUCAS


Form1
Загрузить изображение
Назад
4. Отражение изображения одним из четырёх способов
Выберите отражение картинки
Отразить по горизонтали
Отразить по вертикали
Отразить по горизонтали и вертикали


Form1
Загрузить изображение
Назад
4. Отражение изображения одним из четырёх способов
Выберите отражение картинки
Отразить по горизонтали
Отразить по вертикали
Отразить по горизонтали и вертикали


Form1
Загрузить изображение
Назад
4. Отражение изображения одним из четырёх способов
Выберите отражение картинки
Отразить по горизонтали
Отразить по вертикали
Отразить по горизонтали и вертикали


Form1
Загрузить изображение
Назад
5. Осуществление проекции фрагмента изображения на произвольную плоскость
Преобразовать


Form1
Загрузить изображение
Назад
5. Осуществление проекции фрагмента изображения на произвольную плоскость
Преобразовать