Название работы: Аффинные преобразования и гомография изображений

Цель работы: Целью данной работы является изучение базовых операций над геометрией изображений и их применение к некоторым задачам обработки изображений.

Постановка задачи:

Необходимо разработать приложение Windows Forms, способное осуществлять:

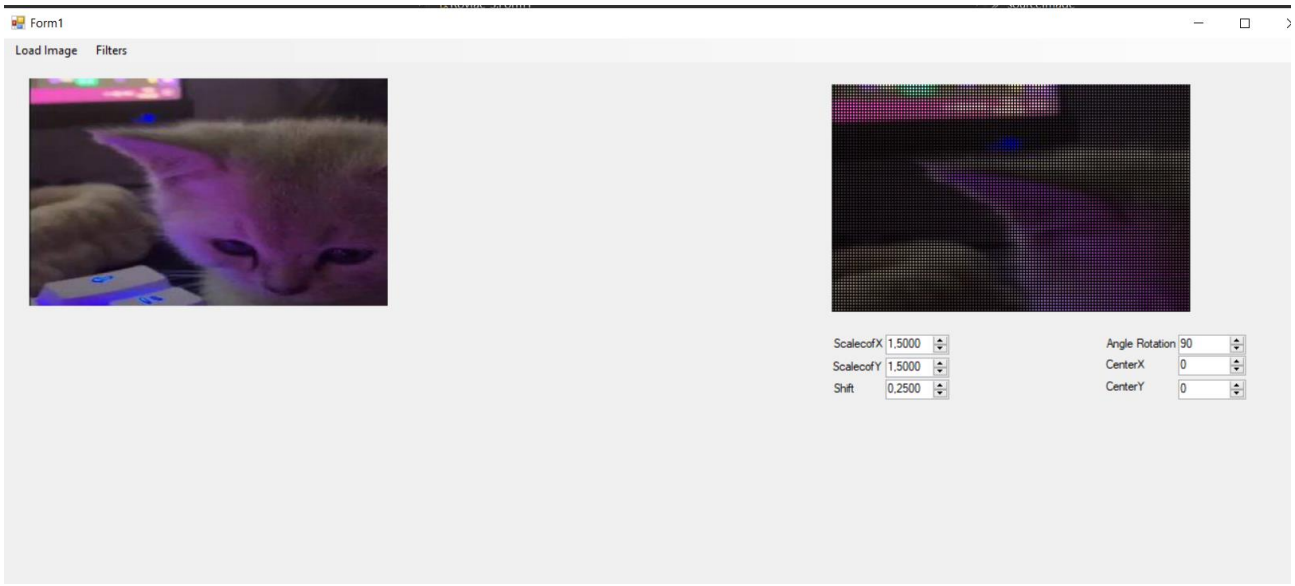
1. загрузку и отображение двух изображений по выбору пользователя;
2. возможность применения аффинных преобразований к загруженным изображениям;
3. возможность проекции области одного изображения на другое.

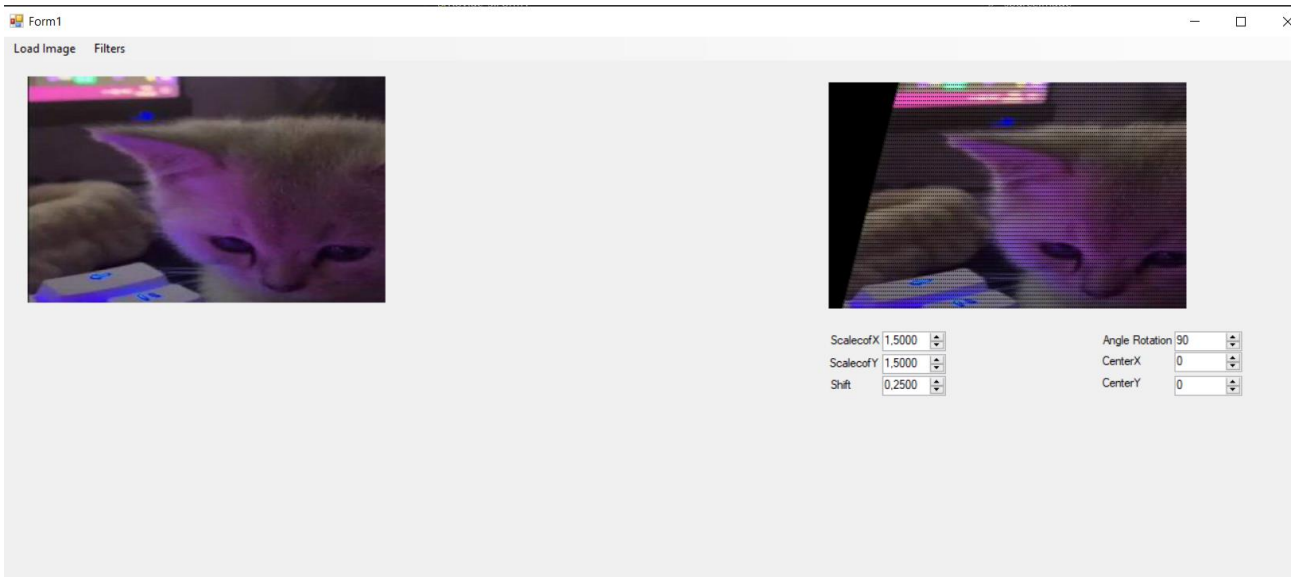
Задание:

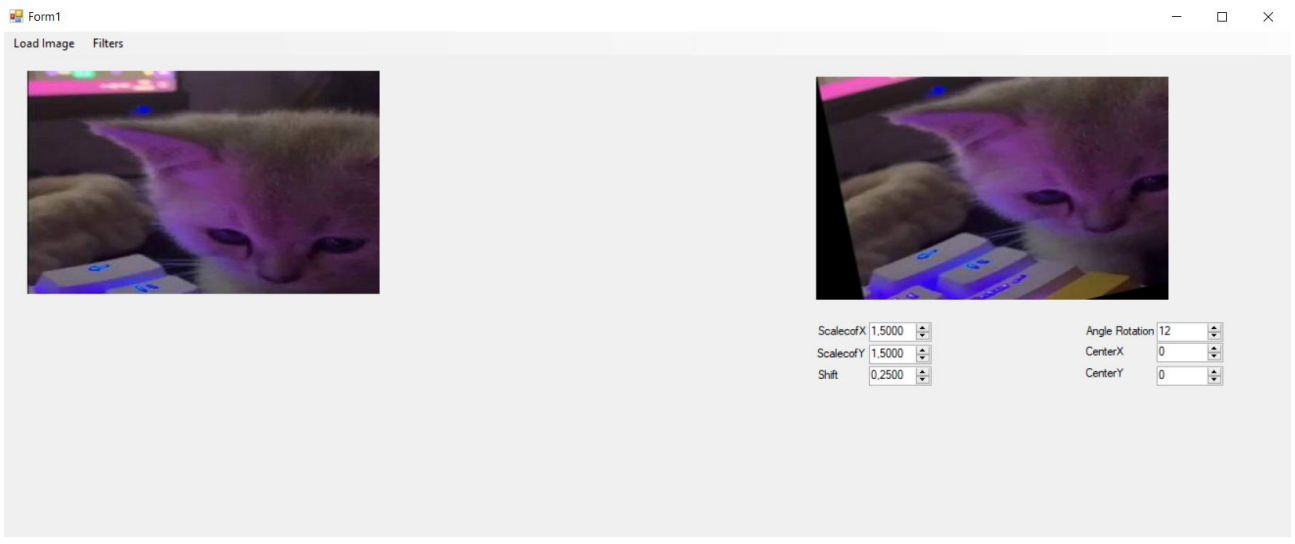
Разработать программу, позволяющую отображать в одном окне два изображения: оригинальное
изображение слева и результат обработки справа. Реализовать интерфейс, позволяющий по нажатию
на соответствующие кнопки выполнять следующие операции:

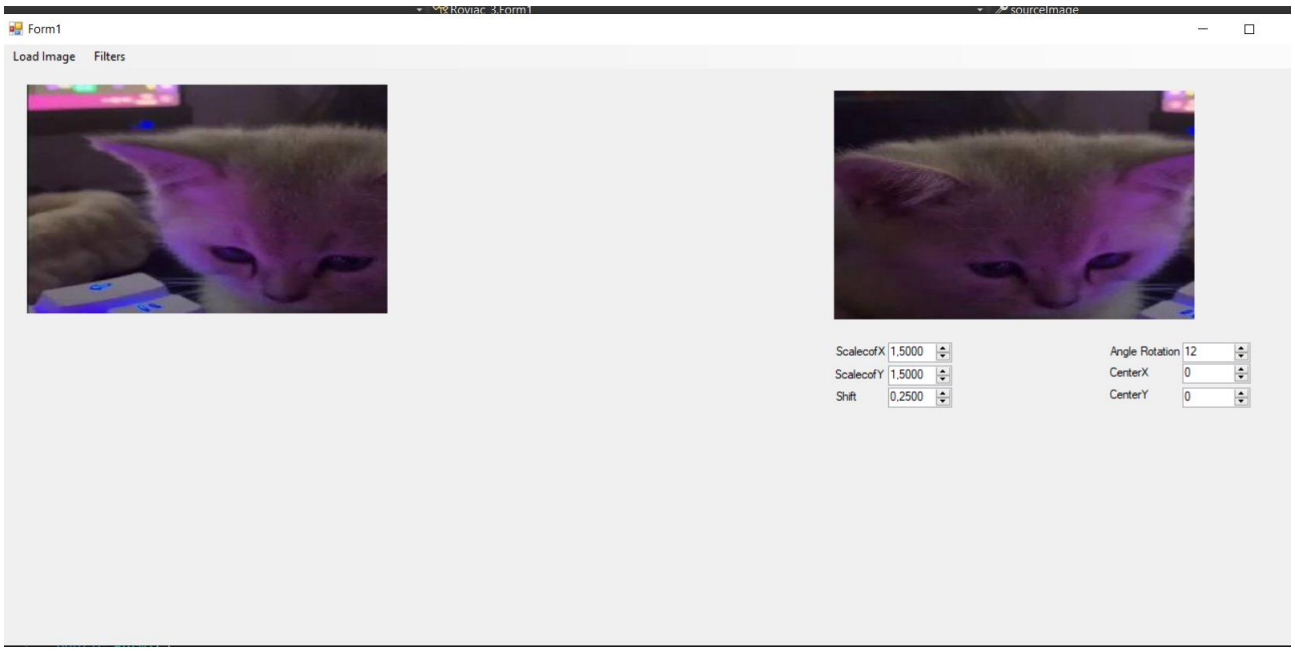
1. масштабирование изображения с параметрами, вводимыми пользователем. Размер результирующего изображения должен изменяться в соответствии с параметрами масштабирования;
2. осуществлять сдвиг изображения на произвольное значение;
3. поворот изображения относительно выбранной пользователем точки на заданный пользователем угол;
4. отражение изображения одним из четырёх способов. При отражении, размер результирующего изображения не должен изменяться;
5. применить билинейную фильтрацию при выполнении поворота и масштабирования для устранения графических дефектов;
6. осуществить проекцию фрагмента изображения на произвольную плоскость. Добавить возможность выбора фрагмента и плоскости пользователем через указание четырех точек с помощью мыши.

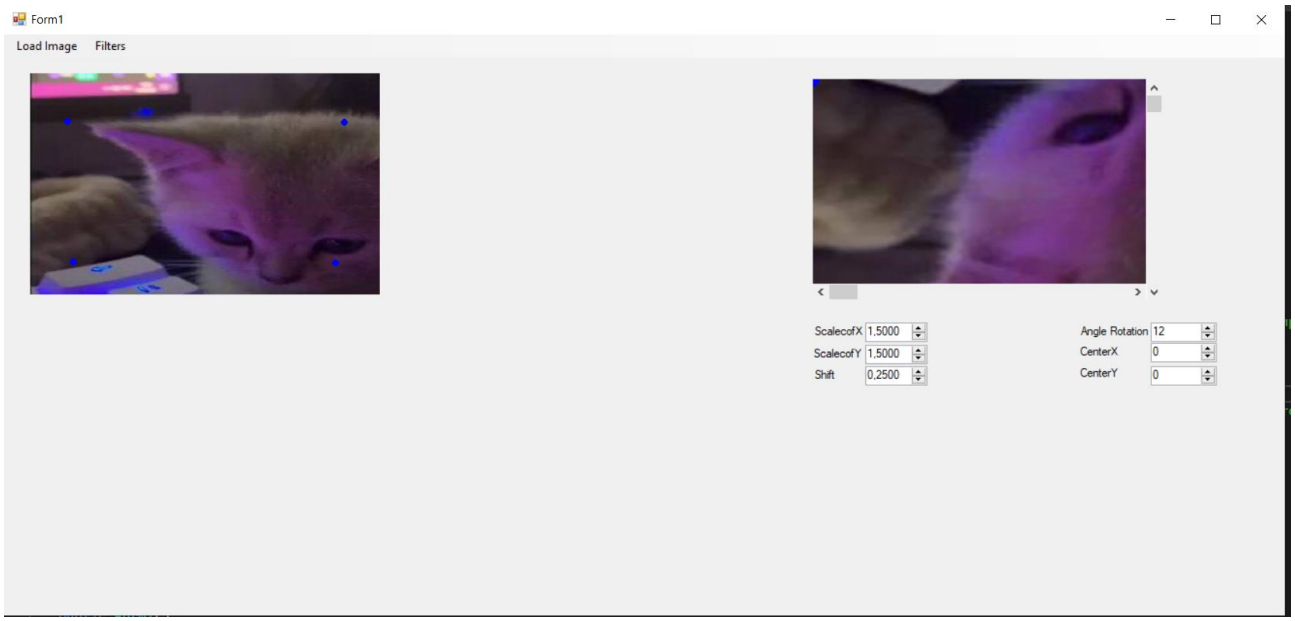
Листинг программы с комментариями:












```
using Emgu.CV.CvEnum;
using Emgu.CV.Structure;
using Emgu.CV;
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.XFeatures2D;
using System.Drawing.Drawing2D;
using System.Runtime.InteropServices.ComTypes;
```

```
//1.масштабирование изображения с параметрами, вводимыми пользователем. Размер
результирующего изображения должен изменяться в соответствии с параметрами
масштабирования; done
//2.осуществлять сдвиг изображения на произвольное значение; done
//3.поворот изображения относительно выбранной пользователем точки на заданный
пользователем угол; done
//4.отражение изображения одним из четырёх способов. При отражении, размер
результирующего изображения не должен изменяться; done
//5.применить билинейную фильтрацию при выполнении поворота и масштабирования для
устранения графических дефектов; done, автоматически делается при 2-3 шаге
//6.осуществить проекцию фрагмента изображения на произвольную плоскость. Добавить
возможность выбора фрагмента и плоскости пользователем через указание четырех точек с
помощью мыши. done

namespace Roviac_3
{
    public partial class Form1 : Form
    {
        public Image<Bgr, byte> sourceImage { get; set; }
        public Image<Bgr, byte> sourceImageBase { get; set; }
        public Form1()
        {
            InitializeComponent();
        }

        private void openToolStripMenuItem_Click(object sender, EventArgs e)
        {
            OpenFileDialog openFileDialog = new OpenFileDialog();
            var result = openFileDialog.ShowDialog();
            openFileDialog.Filter = "Picture files (*jpg, *.png)| *.jpg;*.png";
            if (result == DialogResult.OK)
            {
                string fileName = openFileDialog.FileName;
                sourceImage = new Image<Bgr, byte>(fileName);
                sourceImageBase = sourceImage;
                sourceImage = sourceImage.Resize(480, 280, Inter.Linear);
                imageBox1.Image = sourceImage;
                imageBox2.Image = sourceImage;
            }
        }

        public static Image<Bgr, byte> ScaleImage(Image<Bgr, byte> sourceImage, float
CofX, float CofY)
        {
            float sX = CofX;
            float sY = CofY;
            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++)
                {
                    int newX = (int)(x * sX);
                    int newY = (int)(y * sY);

                    newImage[newY, newX] = sourceImage[y, x];
                }
            }
            //newImage = newImage.Resize(480, 280, Inter.Linear);
            //return newImage.Resize(480, 280, Inter.Linear); ;
            return newImage;
        }

        private void scaleToolStripMenuItem_Click(object sender, EventArgs e)
        {
```

```
            imageBox2.Image = ScaleImage(sourceImage, (float)NumericCofX.Value,
(float)NumericCofY.Value);
        }

        private void ScalecofX_SelectedItemChanged(object sender, EventArgs e)
        {

        }

        public static Image<Bgr, byte> ShearingImage(Image<Bgr, byte> sourceImage,
NumericUpDown shift1)
        {
            var ShearingImage = new Image<Bgr, byte>(sourceImage.Width,
sourceImage.Height);
            float shift = (float) shift1.Value;
            for (int x = 0; x < (int) ShearingImage.Width * (1-shift);x++)
                {
                for (int y = 0; y < ShearingImage.Height; y++)
                {
                    int pixelNewX = Convert.ToInt32(x + shift * (sourceImage.Height -
y));

                    ShearingImage[y, pixelNewX] = sourceImage[y, x];
                }
            }
            return ShearingImage.Resize(480, 280, Inter.Linear);
        }

        private void shearingToolStripMenuItem_Click(object sender, EventArgs e)
        {
            imageBox2.Image = ShearingImage(sourceImage, ShiftValue);
        }

        public static Image<Bgr, byte> RotateImage(Image<Bgr, byte> sourceImage,
NumericUpDown angle1, int CenterX, int CenterY)
        {
            var angle = (float) angle1.Value;
            var RotatedImage = sourceImage;
            PointF rCenter = new PointF(CenterX, CenterY); // Означаем центральные
координаты x,y
            RotationMatrix2D rotationMatrix = new RotationMatrix2D(rCenter, angle, 1);
// rCenter - относительно чего крутим, angle - градус, 1 - разрешение, и создаем
матрицу
            Bgr background = new Bgr(0, 0, 0); // Цвет заднего фона
            RotatedImage = sourceImage.WarpAffine(rotationMatrix,
Emgu.CV.CvEnum.Inter.Cubic, Emgu.CV.CvEnum.Warp.FillOutliers,
Emgu.CV.CvEnum.BorderType.Constant, background);

            return RotatedImage.Resize(480, 280, Inter.Linear);
        }
        private void rotatePictureToolStripMenuItem_Click(object sender, EventArgs e)
        {
            imageBox2.Image = RotateImage(sourceImage, AngleRotationValue,
(int)CenterXvalue.Value, (int)CenterYvalue.Value);
        }

        private void label6_Click(object sender, EventArgs e)
        {

        }

        public static Image<Bgr, byte> ReflectionImage(Image<Bgr, byte> sourceImage)
        {
            var ReflectionImage = new Image<Bgr, byte>(sourceImage.Width,
sourceImage.Height);
            var width = ReflectionImage.Width-1;
```

```
            var height = ReflectionImage.Height-1;
            var qX = -1;
            var qY = 1;

            for (int x = 0; x < sourceImage.Width; x++)
            {
                for (int y = 0; y < sourceImage.Height; y++)
                {
                    var PixelNewX = x * qX + width;
                    var PixelNewY = y * qY;
                    ReflectionImage[PixelNewY, PixelNewX] = sourceImage[y, x];
                }
            }
            return ReflectionImage.Resize(480, 280, Inter.Linear);
        }
        private void reflectionToolStripMenuItem_Click(object sender, EventArgs e)
        {
            imageBox2.Image = ReflectionImage(sourceImage);
        }

        public static Image<Bgr, byte> FiltrationImage(Image<Bgr, byte> sourceImage)
        {
            var FiltrationImage = ScaleImage(sourceImage, 1.5f, 1.5f);

            return FiltrationImage.Resize(480,280,Inter.Linear);
        }


        //public static Bitmap ResizeBeb(Bitmap bmp, float scale, InterpolationMode
mode = InterpolationMode.Bilinear)
        //{
        //    var w = (int)(bmp.Width * scale);
        //    var h = (int)(bmp.Height * scale);

        //    var res = new Bitmap(w, h);
        //    using (var gr = Graphics.FromImage(res))
        //    {
        //        gr.InterpolationMode = mode;
        //        gr.DrawImage(bmp, 0, 0, w, h);
        //    }

        //    return res;
        //}

        private void filtrationToolStripMenuItem_Click(object sender, EventArgs e)
        {
            //Bitmap sharpenImage = new Bitmap(sourceImage.Width, sourceImage.Width);
            imageBox2.Image = FiltrationImage(sourceImageBase);
        }

        private void projectionToolStripMenuItem_Click(object sender, EventArgs e)
        {
            var destPoints = new PointF[]
            {
            // плоскость, на которую осуществляется проекция,
            // задаётся четыремя точками
            new PointF(0, 0),
            new PointF(0, sourceImage.Height - 1),
            new PointF(sourceImage.Width - 1, sourceImage.Height - 1),
            new PointF(sourceImage.Width - 1, 0)
            };
            //MessageBox.Show(srcPoints[0, 0].ToString(), srcPoints[0, 1].ToString());
            //MessageBox.Show(srcPoints[1, 0].ToString(), srcPoints[1, 1].ToString());
```

```
            var homographyMatrix = CvInvoke.GetPerspectiveTransform(srcPoints,
destPoints);
            var destImage = new Image<Bgr, byte>(sourceImage.Size);
            CvInvoke.WarpPerspective(sourceImage, destImage, homographyMatrix,
destImage.Size);

            imageBox2.Image = destImage.Resize(480,280,Inter.Linear);
        }

        int count_click = 0;

        PointF[] srcPoints = new PointF[4];
        PointF[] buffer = new PointF[1];
        PointF nuller = new PointF(0, 0);
        private void imageBox1_MouseClick(object sender, MouseEventArgs e)
        {
            int x = (int)(e.Location.X / imageBox1.ZoomScale);
            int y = (int)(e.Location.Y / imageBox1.ZoomScale);
            if (count_click < 4)
            {
                var buf = new Point(x, y);
                MessageBox.Show(buf.ToString());
                if (buf != nuller && buf != buffer[0])
                {
                    buffer[0] = buf;
                    srcPoints[count_click] = buf;
                    count_click++;
                    Point center = new Point(x, y);
                    int radius = 2;
                    int thickness = 2;
                    var color = new Bgr(Color.Blue).MCvScalar;
                    // функция, рисующая на изображении круг с заданными параметрами
                    CvInvoke.Circle(sourceImage, center, radius, color, thickness);
                    imageBox1.Image = sourceImage;
                }

            }
        }

        private void imageBox1_Click(object sender, EventArgs e)
        {
            imageBox1.MouseClick += new MouseEventHandler(imageBox1_MouseClick);
        }
    }
}
```