South University of Science and Technology of China

Image and Video Processing

Class Project 2

# Affine Transformations and Intensity Transformations

Author:
Jie Yang

Supervisor:
Jianhong Shi

October 9, 2018

# CONTENTS

# Affine Transformations and Intensity Transformations

## Introduction

Image affine transformation is a kind of method to change image shape, which can be divided into image translation, rotation, shear operation and scaling. All of the affine transformations can be expressed as a matrix transform. The transformed image can be calculated from original image multiply by a transform matrix.**Eq.1** Change the matrix, the transformation can be done.

$$g(x,y) = I(x,y) * T \tag{1}$$

Intensity transformation is a another common used image process method. It use a filter to change the intensity of pixel locally. It slides a N×N filter through image, at each step, it choose all pixel in windows, then change their intensity base on the transformation rules. Intensity transformation can be used to smooth and sharpening image by use different filters. Another kind of intensity transform is gamma correction,which used to change image intensity globally to adjust image contrast.

In this experiment, we will determine the transformation matrix of each affine transformation and show how these transformations change image shape. We will also do intensity transformation such as smoothing, sharpening and gamma correction to comprehend how intensity transformation works.

## Method

### Translation

Translation used to translate image in horizontal and vertical image, it dose not change image shape or size, just change the pixel index in. **Eq.2** is its transformation.

$$T_{trans} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ tx & ty & 1 \end{pmatrix} \tag{2}$$

### Rotation

Rotation used to rotate image counterclockwise by $\theta$. But before rotation, original point should be set at the center of image. **Eq.3** is its transformation. And rotation will change image size because the image is storages as matrix, after rotation interpolation also be used to smooth image. In this experiment, we choose bilinear interpolation.

$$T_{rotate} = \begin{pmatrix} cos\theta & sin\theta & 0 \\ -sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{3}$$

## Shear operations

Shear operation used to distort image, it can change image in one dimension while remain another one. For example, vertical shear pull image in vertical axis but remain horizontal axis. **Eq.4** is its transformation. Horizontal shear pull image in horizontal axis but remain vertical axis. **Eq.5** is its transformation. It also need padding in another axis, for it change image size.

$$T_{sv} = \begin{pmatrix} 1 & 0 & 0 \\ s_v & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{4}$$

$$T_{sh} = \begin{pmatrix} 1 & s_h & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{5}$$

## Smoothing

There are three filters to do image smoothing: averaging, median and binarization. They has different filters.**Eq.6**. For averaging filter, it set pixel as the average intensity of whole window while median filter chooses medium. Binarization filter do binarization in each window, the threshold can be set, in this experiment we choose OTSU method to determine threshold automatically.

$$g(x,y) = \frac{\sum_{s=-a}^{a}\sum_{t=-b}^{b} w(s,t)f(x+s,y+t)}{\sum_{s=-a}^{a}\sum_{t=-b}^{b} w(s,t)} \tag{6}$$

## Sharpening

There are two methods to sharper image, they use different spatial filters. Sobel is gradient filter, it calculate gradient of image. While Laplacian operator calculate second order gradient of image.**Eq.7** Both of them has large value near edge, So add them into original image can shaper image edge.

$$T_{sobel} = \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix} s \quad T_{laplacian} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{pmatrix} \tag{7}$$

## Gamma correction

Gamma correction used to map image intensity nonlinearly by exponential function. The less $\gamma$**Eq.8** is the brighter image is, because it will increase the dark area intensity larger than brighter area. In this experiment, we set $c = 1$.

$$s = cr^{\gamma} \tag{8}$$

## Histogram enhancement

Histogram enhancement can be complete by histogram equalization.**Eq.9**. It first get the histogram of image, then for each pixel intensity, calculate how much it accounts for whole histogram. Finally sum all intensity less than it to calculate its CDF and multiply with gray scale to map it to new intensity. For global enhancement, it calculate histogram of whole image. While local one just select one part of image as ROI, calculate the CDF in this area. In this experiment, we set the local window as 20 pixels and the gray scale as 256.

$$S_k = (\sum_{j=0}^{k} \frac{n_j}{MN})(L-1) = (\sum_{j=0}^{k} p(r_k))(L-1) \tag{9}$$

# Results

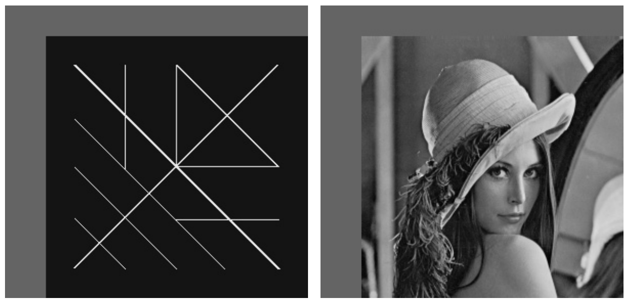
There is the original image of crosses and Lena, one is binary image while another is gray image.**Fig.1**

(a) Crosses (b) Lena

Figure 1: Original images

## Translation

We transfer image by 30 pixels at vertical axis and 40 pixels at horizontal axis. Form **Fig.2**, the transformation is well, the image be translated to southeast while remain shape. The original position be padded as gray.



(a) Crosses (b) Lena

Figure 2: Translated by 30 x 40 pixels

## Rotation

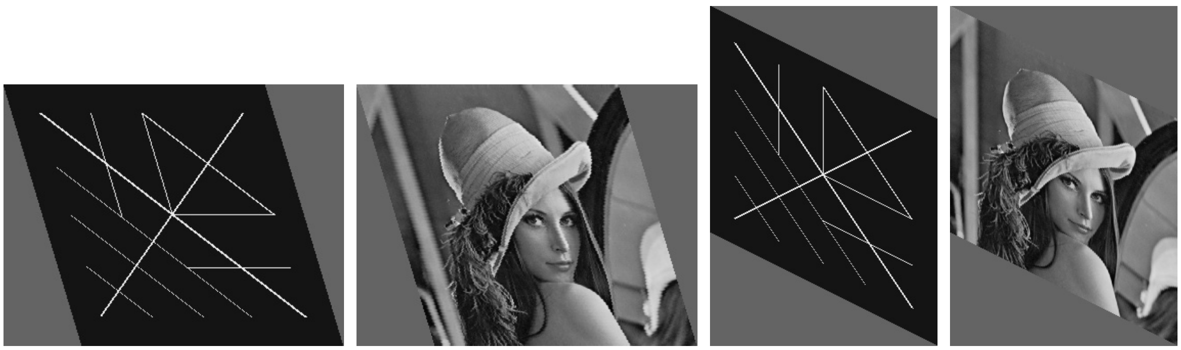
The image is rotated by $45^o$, the whole image size increased and the new area is padded by gray. Actually the meaning area still same to original image size. But during rotation, interpolation be used, so the solution of image decrease, especially in crosses image.

(a) Crosses (b) Lena

Figure 3: Rotated $45^o$

## Shear operations

Shear operation can change image shape, from **Fig.4**, it is clear to see that horizontal shear stretch image in horizontal axis while vertical shear pull image in vertical axis. Because they change image size, they also need padding and interpolation. In shear axis, the solution decreased for interpolation.



(a) 0.3 horizontal shear (b) 0.3 horizontal shear Lena (c) 0.5 vertical shear crosses (d) 0.5 vertical shear Lena

Figure 4: Image shear

## Smoothing

In this part, we use three different filters to do smoothing. First one is average filter. The filter size is $N \times N$, set all weight as $\frac{1}{N^2}$ and do dot multiply with pixels in window, then sum all values to get the average value as new pixel intensity. It is clear to see that average filter smooth the image but also blur the image.**Fig.5** The larger the filter size, the more blur the image.

For median filter, the size is same. but choose the median as new pixel value. From **Fig.6**, the image be smooth but less blur than average filter when increase filter size. Actually it may decrease image intensity a little.

For binary filter, increase filter size may make edge thicker, so that sharp image but decrease

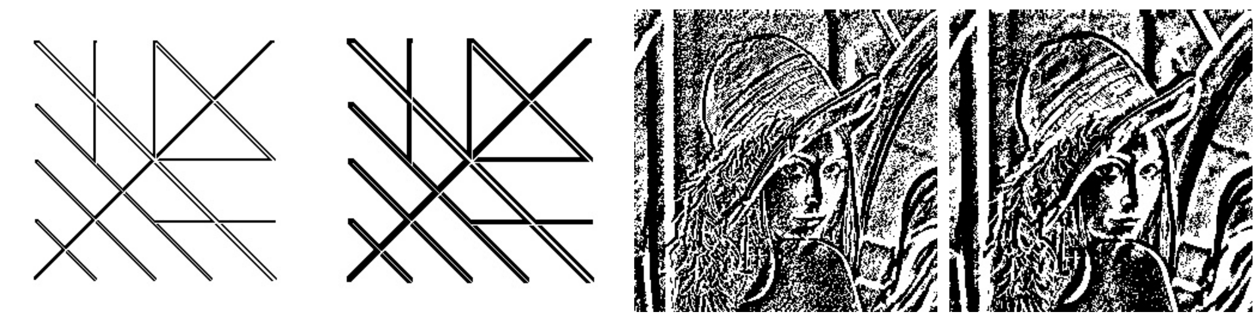
(a) 3x3 average crosses (b) 5x5 average crosses (c) 3x3 average Lena (d) 5x5 average Lena

Figure 5: Image smoothing by average filter

details.**Fig.7**. The smooth performance is the worst in three filters.

(a) 3x3 median crosses (b) 5x5 median crosses (c) 3x3 median Lena (d) 5x5 median Lena

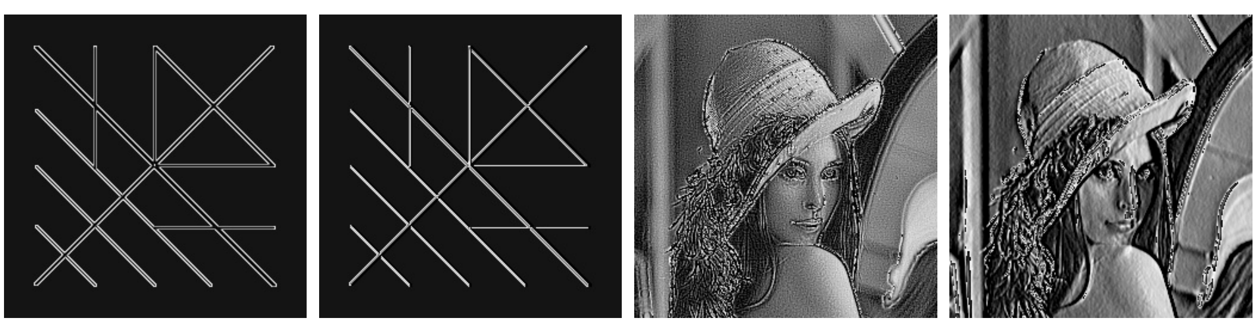
Figure 6: Image smoothing by median filter



(a) 3x3 binarization crosses (b) 5x5 binarization crosses (c) 3x3 binarization Lena (d) 5x5 binarization Lena

Figure 7: Image smoothing by binarization filter

## Sharpening



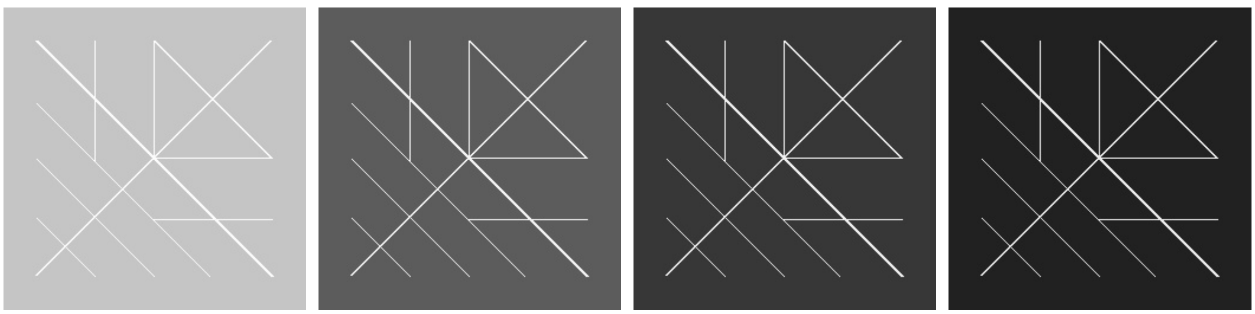
(a) Laplacian sharpened (b) Sobel sharpened crosses (c) Laplacian sharpened (d) Sobel sharpened Lena

Figure 8: Sharpen crosses and Lena with Laplacian and Sobel operator

The sharp operators has high intensity near edge, so that they enhance the edge. Compare to Sobel and Laplacian operator, it is clear to see that Laplacian operator has finer edge while Sobel operator create thicker edge. It the image of Sobel operator seems more blur that Laplacian.**Fig.8** From crosses image, it is clear to see that, Laplacian operator will cause cavity in line but enhance edge while Sobel did not. This also shows that Sobel has larger action domain that Laplacian.

## Gamma correction

Form **Fig.9**, it is clear to see the lower the $\gamma$, the brighter the image. The gamma correction can increase dark background.



(a) $\gamma = 0.1$ (b) $\gamma = 0.4$ (c) $\gamma = 0.6$ (d) $\gamma = 0.8$

Figure 9: Gamma correction of crosses using gamma value 0.1, 0.4, 0.6, 0.8

For Lena image, when $\gamma < 0.4$, the image be overexposure.**Fig.10**. The gamma correction also enhance background.

(a) $\gamma = 0.1$ (b) $\gamma = 0.4$ (c) $\gamma = 0.6$ (d) $\gamma = 0.8$

Figure 10: Gamma correction of Lena using gamma value 0.1, 0.4, 0.6, 0.8

## Histogram enhancement

It is clear to see that for global histogram, the crosses image has low contrast. The reason is that the image only has two intensities: 0 or 1. So it map 0 to brighter intensity. However, for Lena image, it has good performance. The reason is that the intensity distribution in Lena image is broader, so the enhancement just stretch intensity in range of 0 to 255.

(a) Global crosses (b) Local crosses (c) Global Lena (d) Local Lena

Figure 11: Histogram enhancement of crosses and Lena

For local histogram enhancement. Lena has poor performance, because in selected window, the intensity distribution may be too concentrate, while crosses image has better performance than global one. Because when windows at edge, there is has balanced intensity distribution which induces shadow near edge.

# Discussion

In this experiment, we can find that average filter can smooth image but also induce blur, in most situation , median filter has better performance in smoothing. For sharpening, Laplacian has finer edge than Sobel but Sobel can emphasize edge base on thicker border.Global histogram enhancement should be used when the image intensity distribute balance while local histogram enhancement can be considered in unbalanced intensity distribution.

# Reference

https://blog.csdn.net/zhangfuliang123/article/details/76659467

# Supplementary

This is the code used in this project.

```cpp
#include <opencv2/opencv.hpp>
#include <iostream>
#include <string>
#define _USE_MATH_DEFINES
#include<math.h>
#include "PUABLIC.h"


using namespace cv;
using namespace std;


enum Sheartype{ V,H};
enum Smoothtype{ave,med,bin};
enum Sharpentype{ Lap, Sob };
double Bilinearinterpolation(double xhat, double yhat, Mat img){
  double val;
  int x_left = floor(xhat);
  int x_right = ceil(xhat);
  int y_left = floor(yhat);
  int y_right = ceil(yhat);
  uint ll = *(img.data + img.step[0] * x_left + img.step[1] * y_left);
  uint lr = *(img.data + img.step[0] * x_left + img.step[1] * y_right);
  uint rl = *(img.data + img.step[0] * x_right + img.step[1] * y_left);
  uint rr = *(img.data + img.step[0] * x_right + img.step[1] * y_right);
  double p = xhat - x_left;
  double q = yhat - y_left;

  val = (1 - q)*(p*rl + (1 - p)*ll) + q*(p*lr + (1 - p)*rr);
  return val;

}

struct Result{
  double x;
  double y;
};

uint getval(int x, int y, Mat img){

  int xsize = img.rows;
  int ysize = img.cols;
  double val;
  if (0 <= x&&x < xsize && 0 <= y&&y < ysize)
    val = Bilinearinterpolation(x, y, img);

  else
    val = 100;

  return uint(val);

```

```
}
uint getvalat(int x, int y, Mat img){

  int xsize = img.rows;
  int ysize = img.cols;
  double val;
  if (0 ≤ x&&x < xsize && 0 ≤ y&&y < ysize + 1)
    val = img.at<uchar>(y,x);//这样取值只有大小1/4

  else
    val = 100;

  return val;

}
int getmed(Mat roi){
  Mat mat_dst; //: 需要排序的mat_srcMat类型的矩阵变量
  Mat smat;
  roi.copyTo(mat_dst);
  //cout << roi << endl;
  mat_dst = mat_dst.reshape(0, 1);
  //cout << mat_dst << endl;
  cv::sort(mat_dst, smat, CV_SORT_EVERY_ROW + CV_SORT_ASCENDING);
  //cout << smat << endl;
  int c = ceil(smat.cols / 2.0) - 1;
  int medval = *(mat_dst.data + mat_dst.step[1] * c);
  return medval;
}


struct Result gettrans(double x, double y, Mat img,int flag){
  struct Result trans;
  int xsize = img.rows;
  int ysize = img.cols;
  if (flag){
    trans.x = x - xsize / 2.0 +0.5;
    trans.y = ysize/2.0 - y  -0.5;
  }
  else{
    trans.x = x + xsize / 2.0 -0.5;
    trans.y = ysize / 2.0  -0.5 - y;
  }
  return trans;

}


Mat Transform(int Xlen, int Ylen, Mat img, Mat T ){
  Mat I = Mat::zeros(Xlen, Ylen, CV_8UC1);
  for (int x = 0; x < Xlen; ++x){
    uchar * p = I.ptr<uchar>(x);
```

```
    for (int y = 0; y < Ylen; ++y){
      struct Result tans = gettrans(x, y, I, 1);
      //cout << "(x,y) (" << x << "," << y << ")\t";
      //cout << "(tx,ty) (" << tans.x << "," << tans.y << ")\t";

      Mat XY(3, 1, CV_64FC1);
      XY.at<double>(0, 0) = tans.x;
      XY.at<double>(1, 0) = tans.y;
      XY.at<double>(2, 0) = 1;
      //cout << "T " << T << "\t";
      Mat UXY = T*XY;
      double ux = UXY.at<double>(0, 0);
      double uy = UXY.at<double>(1, 0);
      tans = gettrans(ux, uy, img, 0);
      //cout << "(ux,uy) (" << ux << "," << uy << ")\t";
      //cout << "(ox,oy) (" << tans.x << "," << tans.y << ")\t"<<endl;
      uint b = getval(ceil(tans.x), ceil(tans.y), img);
      p[y] = b;

    }

  }
  return I;

}

Mat Rotation(Mat img, double theta){


  int xsize = img.rows;
  int ysize = img.cols;
  double r = sqrt(pow(xsize, 2) + pow(ysize, 2)) / 2.0;
  double beta = acos(xsize / (2*r));
  theta = theta*M_PI / 180;
  Mat T = Mat::eye(3, 3, CV_64FC1);
  T.at<double>(0, 0) = cos(theta);
  T.at<double>(0, 1) = sin(theta);
  T.at<double>(1, 0) = -sin(theta);
  T.at<double>(1, 1) = cos(theta);


  double Xlen = ceil(r*max(abs(cos(beta + theta)), abs(cos(beta - theta))) * 2);
  double Ylen = ceil(r*max(abs(sin(beta + theta)), abs(sin(beta - theta))) * 2);
  Mat I = Transform(Xlen, Ylen, img, T);

  return I;
}



Mat Translation(Mat img, double tx,double  ty){
```

```


  int xsize = img.rows;
  int ysize = img.cols;
  Mat T = Mat::eye(3, 3, CV_64FC1);
  T.at<double>(0, 2) = -tx /2.0;
  T.at<double>(1, 2) = ty / 2.0;


  int Xlen = xsize + ceil(tx);
  int Ylen = ysize + ceil(ty);
  Mat I = Transform(Xlen, Ylen, img, T);
  return I;
}



Mat Shear(Mat img, double s, Sheartype ST  ){

  int xsize = img.rows;
  int ysize = img.cols;
  Mat T = Mat::eye(3, 3, CV_64FC1);
  int Xlen = xsize;
  int Ylen = ysize;
  switch (ST){
  case H:{T.at<double>(1, 0) = s;
      Ylen += ceil(Xlen*s);
      }
    break;
  case V:{T.at<double>(0, 1) = s;
      Xlen += ceil(Ylen*s);
      }
    break;
  default: cout << "Shear type error !" << endl; break;
  }
  Mat I = Transform(Xlen, Ylen, img, T);

  return I;
}


Mat Smooth(Mat img, int size, Smoothtype SMT){
  int r = floor(size / 2);

  int xsize = img.rows;
  int ysize = img.cols;
  int Xlen = xsize - size + 1;
  int Ylen = ysize - size + 1;
  Mat I = Mat::zeros(Xlen, Ylen, CV_8UC1);
  Mat broi;
  double b;
```

```
  for (int x = 0; x < Xlen; ++x){
    uchar * p = I.ptr<uchar>(x);
    for (int y = 0; y < Ylen; ++y){
      Mat roi(img, Rect(y, x, size,size));
      //cout<<size << "roi\n" << roi << endl;
      switch (SMT){
      case ave:{Scalar c = mean(roi);
        b = c[0];
      }
        break;
      case med:{ b = getmed(roi);


      }
        break;
      case bin:{ threshold(roi, broi, 0, 255, CV_THRESH_OTSU);

        b = broi.at<uchar>(0, 0);
        //cout << "roi" <<roi <<"\nbroi" << broi << "\n ui" << b<<endl;
      } break;

      default: cout << "Smooth type error !" << endl; break;
      }
      p[y] = uint(b);


      //Mat XY(3, 1, CV_64FC1);
      //uint b = getval(ceil(tans.x), ceil(tans.y), img);
      //;
    }

  }
  return I;
}

Mat Sharpen(Mat img,  Sharpentype SPT){
  Mat T = Mat::ones(3, 3, CV_64FC1);
  Mat I = Mat::zeros(img.rows - 2, img.cols - 2, CV_8UC1);
  Mat Fr;

  switch (SPT){
  case Lap: {T.at<double>(1, 1) = -8 ; }break;
  case Sob:{ T = (Mat_<double>(3, 3) <<-1,0,1,-2,0,2,-1,0,1 ); }break;
  default: cout << "Sharpen type error !" << endl; break;

  }
  for (int x = 0; x < img.rows-2; ++x){
    uchar * p = I.ptr<uchar>(x);
    for (int y = 0; y < img.cols-2; ++y){
      Mat roi(img, Rect(y, x, 3,3));
      roi.convertTo(Fr, CV_64FC1);
```

```
        //cout << "roi\n" << roi << endl;
        double AB = T.dot(Fr);
        uint a = *(img.data + img.step[0] * (x + 1) + img.step[1] * (y + 1));
        double c = ((AB + a) + abs(AB + a)) / 2;

        p[y] = uint(c);


      }
    }

    return I;

}


Mat GammaCorect(Mat img, double gama,int c){

    Mat I = Mat::zeros(img.rows , img.cols, CV_8UC1);


    for (int x = 0; x < img.rows; ++x){
      uchar * p = I.ptr<uchar>(x);
      for (int y = 0; y < img.cols; ++y){

        uint r = *(img.data + img.step[0] * x  + img.step[1] * y );
        double s = 255.0*pow(r/255.0, gama);
        p[y] = uint(s);


      }
    }

    return I;

}

Mat HisEnh(Mat img){

    Mat I = Mat::zeros(img.rows, img.cols, CV_8UC1);

    double minv = 0.0, maxv = 0.0;
    double* minp = &minv;
    double* maxp = &maxv;
    minMaxIdx(img, minp, maxp);
    //cout << *minp << endl;
    //cout << *maxp << endl;
    int L = *maxp - *minp + 1;
    int histSize[] = { L }; /
    float midRanges[] = { *minp, *maxp + 1 };

```

```
  int channels = 0;
  MatND dstHist; //hist


  const float *ranges[] = { midRanges };

  calcHist(&img, 1, &channels, Mat(), dstHist, 1, histSize, ranges, true, false);
  //cout << dstHist << endl;
  for (int x = 0; x < img.rows; ++x){
    uchar * p = I.ptr<uchar>(x);
    for (int y = 0; y < img.cols; ++y){
      uint r = *(img.data + img.step[0] * x + img.step[1] * y);
      int s = r - *minp+1;
      Mat roi(dstHist, Rect(0, 0, 1,s));
      //cout << r << "\n" << roi << endl;
      Scalar c = sum(roi);
      Scalar d = sum(dstHist);
      double pk = c[0]/d[0];//get percent
      p[y] = 255 * pk;
    }
  }

  return I;


Mat Hislocal(Mat img,int size){
  int r = floor(size / 2);

  int xsize = img.rows;
  int ysize = img.cols;
  int Xlen = xsize - size + 1;
  int Ylen = ysize - size + 1;
  Mat I = Mat::zeros(Xlen, Ylen, CV_8UC1);

  double b;
  for (int x = 0; x < Xlen; ++x){
    uchar * p = I.ptr<uchar>(x);
    for (int y = 0; y < Ylen; ++y){
      Mat local(img, Rect(y, x, size, size));
      Mat hisenlocal = HisEnh(local);
        b = hisenlocal.at<uchar>(0, 0);
        p[y] = uint(b);
    }

  }
  return I;

}


void lab2_main()
```

```
{
  ImgProp imgprop = {
    "D://graduated//Image_process//lab//PGM_images//",
    ".pgm" };
  ImgProp imgsave = { "D://graduated//Image_process//lab//lab_report//lab3//...
    imagesave//",
    ".jpg" };
  for (int i = 1; i < 3; i++)
  {
    if (i == 1)imgprop.img_name = "lena";
    else imgprop.img_name = "crosses";
    imgsave.img_name = imgprop.img_name;
    string img_path = imgprop.img_fold + imgprop.img_name + imgprop.type;
    Mat img = imread(img_path, 0);

    imshow("original     " + imgprop.img_name, img);
    imgsave.img = img;
    imgsave.mark = "original";
    img_save(imgsave);

    double tx = 30;
    double ty = 40;
    imgsave.img = Translation(img, tx, ty);
    imgsave.mark = "Trans" + to_string(int(tx)) + "_" + to_string(int(ty));
    img_save(imgsave);

    double theta = 45;
    imgsave.img = Rotation(img, theta);
    imgsave.mark = "rota"+to_string(int(theta));
    img_save(imgsave);




    double  s = 0.5;
    imgsave.img = Shear(img, s, V);
    imgsave.mark = "Shear" + to_string(int(s)) + "_V";
    img_save(imgsave);

    s = 0.3;
    imgsave.img = Shear(img, s, H);
    imgsave.mark = "Shear" + to_string(int(s)) + "_H";
    img_save(imgsave);

    int size=3;
    imgsave.img = Smooth(img, size, ave);
    imgsave.mark = "ave" + to_string(int(size)) ;
    img_save(imgsave);
    imgsave.img = Smooth(img, size, med);
    imgsave.mark = "med" + to_string(int(size));
    img_save(imgsave);
```

```
    imgsave.img = Smooth(img, size, bin);
    imgsave.mark = "bin" + to_string(int(size));
    img_save(imgsave);

    size = 5;
    imgsave.img = Smooth(img, size, ave);
    imgsave.mark = "ave" + to_string(int(size));
    img_save(imgsave);
    imgsave.img = Smooth(img, size, med);
    imgsave.mark = "med" + to_string(int(size));
    img_save(imgsave);
    imgsave.img = Smooth(img, size, bin);
    imgsave.mark = "bin" + to_string(int(size));
    img_save(imgsave);

    imgsave.img = Sharpen(img, Lap);
    imgsave.mark = "Lap" ;
    img_save(imgsave);
    imgsave.img = Sharpen(img, Sob);
    imgsave.mark = "Sob";
    img_save(imgsave);

    int c = 1;
    double gamaset[] = { 0.1,0.4,0.6,0.8 };
    for (int i = 0; i < 4; i++){
      double gama = gamaset[i];
    imgsave.img = GammaCorect(img, gama, c);
    imgsave.mark = "Gamma_" + to_string(int(gama)) + "_" + to_string(int(gama * ...
    10));
    img_save(imgsave);
    }

    imgsave.img = HisEnh(img);
    imgsave.mark = "His_global";
    img_save(imgsave);
    imgsave.img = Hislocal(img,20);
    imgsave.mark = "His_local";
    img_save(imgsave);
  }


}
```

C++ code for image processing