Assignment 5

Part 1 soure code

```
#include <opencv2/core/core.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/imgproc/imgproc.hpp>
#include <iostream>
#include <fstream>
#include <vector>

using namespace cv;
using namespace std;

//calculate the mean and modify the value of (i,j) to the mean num
void mean_filter_helper(IplImage* img, int i, int j)
{
	//calculate the sum
	int sum = 0;
	for (int row = i - 1; row < i + 2; row++)
	{
		uchar* ptr = (uchar*)img->imageData + row*img->widthStep;
		for (int col = j - 1; col < j + 2; col++)
		{
			sum += ptr[col];
		}
	}

	//modify the value
	uchar* ptr = (uchar*)img->imageData + i*img->widthStep;
	ptr[j] = sum / 9;
}

void mean_filter(IplImage* img)
{
	//don't change the pixels in the boundary
	for (int row = 1; row < img->height - 1; row++)
	{
		for (int col = 1; col < img->width - 1; col++)
		{
			mean_filter_helper(img, row, col);
		}
	}
}
```

```
//calculate the median and modify the value of (i,j) to the median num
void median_filter_helper(IplImage* img, int i, int j)
{
	//calculate the sum
	int median = 0;
	int count = 0;
	int temp[9];
	for (int row = i - 1; row < i + 2; row++)
	{
		uchar* ptr = (uchar*)img->imageData + row*img->widthStep;
		for (int col = j - 1; col < j + 2; col++)
		{
			temp[count] = ptr[col];
			count++;
		}
	}

	//sort the temp array
	int max_index = 9;
	while (max_index > 1)
	{
		//make the max number at the max_index position
		for (int temp_index = 0; temp_index < max_index - 1; temp_index++)
		{
			if (temp[temp_index]>temp[temp_index + 1])
			{
				//swap
				int swap = temp[temp_index + 1];
				temp[temp_index + 1] = temp[temp_index];
				temp[temp_index] = swap;
			}
		}
		//decrease the max_index
		max_index--;
	}

	//modify the value
	uchar* ptr = (uchar*)img->imageData + i*img->widthStep;
	ptr[j] = temp[4];
}

void median_filter(IplImage* img)
{
```

```
	//don't change the pixels in the boundary
	for (int row = 1; row < img->height - 1; row++)
	{
		for (int col = 1; col < img->width - 1; col++)
		{
			median_filter_helper(img, row, col);
		}
	}
}

void gaussian_initialize(int (&mask)[5][5])
{
	mask[0][0] = 1; mask[0][1] = 4; mask[0][2] = 7; mask[0][3] = 4; mask[0][4] = 1;
	mask[1][0] = 4; mask[1][1] = 16; mask[1][2] = 26; mask[1][3] = 16; mask[1][4] = 4;
	mask[2][0] = 7; mask[2][1] = 26; mask[2][2] = 41; mask[2][3] = 26; mask[2][4] = 7;
	mask[3][0] = 4; mask[3][1] = 16; mask[3][2] = 26; mask[3][3] = 16; mask[3][4] = 4;
	mask[4][0] = 1; mask[4][1] = 4; mask[4][2] = 7; mask[4][3] = 4; mask[4][4] = 1;
}

void gaussian_filter_helper(IplImage* img, int i, int j)
{
	const int sum_weight = 273;
	///use 5*5 gaussin matrix within 3 sigma
	int mask[5][5];
	gaussian_initialize(mask);

	//calculate the sum
	int sum = 0;
	for (int row = i - 2; row < i + 3; row++)
	{
		uchar* ptr = (uchar*)img->imageData + row*img->widthStep;
		for (int col = j - 2; col < j + 3; col++)
		{
			//get the image_data
			int image_data = ptr[col];

			//get the mask_data
			int mask_row = row - i + 2;// 4 - (row - i + 2);//since the matrix is symmetry
			int mask_col = col - j + 2;// 4 - (col - j + 2);
			int mask_data = mask[mask_row][mask_col];

			sum += image_data*mask_data;
		}
	}
```

```
	//modify the value
	uchar* ptr = (uchar*)img->imageData + i*img->widthStep;
	ptr[j] = sum / sum_weight;
}

void gaussian_filter(IplImage* img)
{
	//don't change the pixels in the boundary
	for (int row = 2; row < img->height - 2; row++)
	{
		for (int col = 2; col < img->width - 2; col++)
		{
			gaussian_filter_helper(img, row, col);
		}
	}
}

int main()
{
	//load the image
	IplImage* img = cvLoadImage("filters.png", 0);

	//choose the type of filter
	int mode = 0;
	cout << "Please choose a type of filter(1-Mean 2-Median 3-Gaussian): ";
	cin >> mode;
	cout << endl;

	//do different types of filter
	if (mode == 1)
	{
		mean_filter(img);
	}
	else if (mode == 2)
	{
		median_filter(img);
	}
	else if (mode == 3)
	{
		gaussian_filter(img);
	}
	else
	{
```

```
            cout << "Type invalid!\n";
            system("pause");
            exit(0);
        }

        //save result image
        if (mode == 1)
        {
            cvSaveImage("mean_filters.png", img);
        }
        else if (mode == 2)
        {
            cvSaveImage("median_filter.png", img);
        }
        else
        {
            cvSaveImage("gaussian_filter.png", img);
        }

        //release space
        cvReleaseImage(&img);
        system("pause");
        return 0;
}
```

Part 2 cite

(PS:you can read comment of filter.cpp to get more info)

There are three pairs of functions:

(1)mean_filter/mean_filter_helper

This just use 3*3 matrix with the same value 1 for each element.Just equal to calculate the average of 9 numbers to replace the origin data.It’s a smooth filter.

(2)median_filter/median_filter_helper

This just sort the 9 numbers around the middle point(I just use bubble sort).This cannot be replaced by a folding operation, so it’s not a smooth filter.

(3)gaussian_filter/gaussian_filter_helper

I just use 5*5 matrix as a mask within 3 sigma.I make a folding operation between the image data and the mask.

As a conclusion, the larger mask I use, the more ambigious image i get.

In my view, gaussian filter is just similar with mean filter.The only difference is that each weight of the matrix is various in gaussian filter.So the gaussian can keep more details of the origin image compared with mean_filter.

While the median filter is more suitable for salt-and-pepper noise.

Part 3 output

(1)mean_filter_image

The implementation of this filter is easy enough and can make the image smooth. However, it just decreases the influence of noises without remove it effectively. The noises are still exist which just become smaller.

(2)median_filter_image

This method isn't a liner method, however maybe median is more objective than mean, so this filter seems protect the verge of sharpen areas. Compared with the former filter, we can find that all of the noises are almost removed and the picture

seems more clear than that handled by mean filter.

(3)gaussian_filter_image

Gaussian filter is just like a development version of mean filter which uses a weighted matrix as a mask. It is also a liner filter and can decreases the effect of noises without removing it. So the results of those two filters are similar.