CV-HW1　　施益豪

- 流程說明

1. 從LightSource，讀取光源座標，並正規化 (s_m)

```
inputFile.open( "/Users/sam/Desktop/CV_hw1/test/bunny/LightSource.txt" , ios::in );
outputFile1.open( "/Users/sam/Desktop/CV_hw1/test/bunny/bunny-surface1.ply" , ios::out );
outputFile2.open( "/Users/sam/Desktop/CV_hw1/test/bunny/bunny-surface2.ply" , ios::out );

string line;

/*read the LightSource and store in the light_info*/
int line_row=0;
int line_col=0;
while(getline(inputFile,line)){
    //cout << line.size()<<endl;
    int start=7;
    line_col=0;
    for(int i=7;i<line.size()-1;i++){
        if( line[i]==',' || line[i]==')'){
            string num=line.substr(start,i-start);
            //cout << num<<endl;
            start=i+1;
            light_info[line_row][line_col]=atoi(num.c_str());
            line_col++;
        }
    }
    line_row++;
}
```

```
/*get the vector from surface to the light and normalize*/
void getVector(){

    for(int i=0;i<6;i++){

        int temp_x=(light_info[i][0]);
        int temp_y=(light_info[i][1]);
        int temp_z=(light_info[i][2]);
        float dis = sqrt((temp_x*temp_x)+(temp_y*temp_y)+(temp_z*temp_z));
        //cout << "dis= "<<dis<<endl;
        s_vector.at<float>(i,0) = (temp_x/dis);
        s_vector.at<float>(i,1) = (temp_y/dis);
        s_vector.at<float>(i,2) = (temp_z/dis);

    }
    //cout<<"s_vector"<<s_vector<<endl;
}
```

2. 將output的ply檔，所需header部分先寫出

```
void writePlyheader(){

    outputFile1 << "ply\n";
    outputFile1 << "format ascii 1.0\n";
    outputFile1 << "comment alpha=1.0\n";
    outputFile1 << "element vertex 14400\n";
    outputFile1 << "property float x\n";
    outputFile1 << "property float y\n";
    outputFile1 << "property float z\n";
    outputFile1 << "property uchar red\n";
    outputFile1 << "property uchar green\n";
    outputFile1 << "property uchar blue z\n";
    outputFile1 << "end_header\n";

    /*outputFile2 << "ply\n";
    outputFile2 << "format ascii 1.0\n";
    outputFile2 << "comment alpha=1.0\n";
    outputFile2 << "element vertex 14400\n";
    outputFile2 << "property float x\n";
    outputFile2 << "property float y\n";
    outputFile2 << "property float z\n";
    outputFile2 << "property uchar red\n";
    outputFile2 << "property uchar green\n";
    outputFile2 << "property uchar blue z\n";
    outputFile2 << "end_header\n";*/

}
```

3. 針對六張圖同一個pixel位置的點(x, y)取其強度（$i_{x,y}$）

```
    /*Normal Estimation : loop each pixel,each loop compute the intensity of each image at pixel (x,y)*/
    for (int rowIndex = 0; rowIndex < Image1.rows; rowIndex++) {
        for (int colIndex = 0; colIndex < Image1.cols; colIndex++) {

            cout << rowIndex<<" "<<colIndex<<endl;
            //cout<<"start intentsity"<<endl;
            getIntensity(rowIndex,colIndex);
            //cout <<"start solve"<<endl;
            solve();
            surface(rowIndex,colIndex);


        }
    }
```

```
/*get the intensity of each pixel from each image*/
void getIntensity(int row, int col){

    img_intensity.at<float>(0,0) = Image1.at<uchar>(row, col);
    img_intensity.at<float>(1,0) = Image2.at<uchar>(row, col);
    img_intensity.at<float>(2,0) = Image3.at<uchar>(row, col);
    img_intensity.at<float>(3,0) = Image4.at<uchar>(row, col);
    img_intensity.at<float>(4,0) = Image5.at<uchar>(row, col);
    img_intensity.at<float>(5,0) = Image6.at<uchar>(row, col);
    //cout << img_intensity<<endl;

}
```

4. k_d及l_m假設為1做後續計算

5. 透過pseudo-inverse方法，計算出(x, y)座標的法向量

```
/*solve the matrix*/
void solve(){


    n=( ( ( ( (s_vector.t()) * (s_vector) ).inv() ) * (s_vector.t()) ) * img_intensity );
    //cout << "n \n"<< n<<endl;

}
```

6. 取得法向量，則可取得 $\frac{\partial f}{\partial x}$ 及 $\frac{\partial f}{\partial y}$

7. 用 $\frac{\partial f}{\partial x}$ 及 $\frac{\partial f}{\partial y}$ 透過累加的方式積出z值，即每一點的高度值，並輸出至ply檔

```
void surface(int row, int col){

    float temp_x,temp_y;

     if(n.at<float>(2.0)==0.0){
        temp_x=0.0;
        temp_y=0.0;
         cout << "in equal zero"<<endl;
    }else{

        temp_x =-(n.at<float>(0,0)) / (n.at<float>(2,0));
        temp_y =-( n.at<float>(1,0)) / (n.at<float>(2,0));

        if(row==0){
            if(col!=0){
                n_sum.at<float>(0,col) = n_sum.at<float>(0,col-1)+past_x;
            }
        }
        else{
            n_sum.at<float>(0,col)=n_sum.at<float>(0,col)+record_y.at<float>(0,col);
        }
        past_x=temp_x;
        record_y.at<float>(0,col)=temp_y;

    }


    //float z = sum_x+sum_y;
    float z = n_sum.at<float>(0,col);
    outputFile1 << row<<" "<<col<<" "<<z<<" 255 255 255"<<endl;
    cout << "z\n"<<z<<endl;



}
```

- **額外功能**

1. Weighted Least Squares

   原公式兩側同乘一個各個圖像重要性權重之矩陣(w)，也就是 w*s, w*i，後續也透過 pseudo-inverse 方法，計算出(x, y)座標的法向量，並以法向量推側此點之高度值。

   - 權重的選取：透過亂數產生六組 0~1 之間的值作為權重，並正規化

```
void randomWeight(){

    srand((unsigned)time(NULL));
    float random[6];
    float sum_random=0.0;
    for(int i=0;i<6;i++){
        random[i]=(float)rand()/RAND_MAX;
        sum_random=sum_random+random[i];
    }

    for(int i=0;i<6;i++){
        random[i]=random[i]/sum_random;
    }

    for(int i=0;i<6;i++){
        for(int j=0;j<6;j++){
            if(i!=j){
                w.at<float>(i,j)=0.0;
            }else{
                w.at<float>(i,j)=random[i];
                cout <<"random\n"<<w.at<float>(i,j)<<endl;
            }
        }
    }
}
```

原先公式的 s，變為 w*s，原先公式的 i，變為 w*i

```
void solve1(){

    img_intensity = w* img_intensity;
    s_vector = w* s_vector;
    n=( ( ( ( (s_vector.t()) * (s_vector) ).inv() ) * (s_vector.t()) ) * img_intensity );
    //cout << "n \n"<< n<<endl;

}
```

2. Special input

在重複上述公式計算（x, y）點高度值前，透過 filter 方式將圖像平滑化，以降低雜訊。將圖像平滑化後，再透過上述方法推得出（x, y）點之高度值。

- 使用 gaussian filter，針對周遭點所形成 3*3 矩陣進行卷積，以降低雜訊干擾。上方點的強度權重為 0.125，下方點的強度權重為 0.125，左方點的強度權重為 0.125，右方點的強度權重為 0.125，中心點強度權重為 0.5。以此高斯模版，針對每一個點計算高斯模糊值。

```
float filter(Mat image,int row, int col){

    float sum=0;

    if(col-1>=0 && row-1>=0 && col+1<image.cols && row+1<image.rows){
        cout <<"each value"<<endl;
        cout <<(float)image.at<uchar>(row,col-1)<<" "<<(float)image.at<uchar>(row,col+1)<<" "<<(float)image.at<uchar>(row+1,col)<<" "<
            <(float)image.at<uchar>(row,col)<<endl;
        float temp_top=(float)image.at<uchar>(row,col-1)*0.125;
        float temp_down=(float)image.at<uchar>(row,col+1)*0.125;
        float temp_left=(float)image.at<uchar>(row-1,col)*0.125;
        float temp_right=(float)image.at<uchar>(row+1,col)*0.125;
        float temp_center=(float)image.at<uchar>(row,col)*0.5;
        cout <<"top = "<<temp_top<<" down = "<<temp_down<<" left = "<<temp_left<<" right = "<<temp_right<<endl;
        sum = temp_top+temp_down+temp_left+temp_right+temp_center;
        if(sum>255){
            sum=255;
        }
        if(sum<0){
            sum=0;
        }
    }else{
        sum = image.at<uchar>(row,col);
    }
    cout <<"the point value is = "<<sum<<endl;

    return sum;
}
```

- 使用 medianBlur，針對周遭點所形成 3*3 矩陣進行卷積，以降低雜訊干擾

```
    medianBlur(Image1, new_img1, 3);
    medianBlur(Image2, new_img2, 3);
    medianBlur(Image3, new_img3, 3);
    medianBlur(Image4, new_img4, 3);
    medianBlur(Image5, new_img5, 3);
    medianBlur(Image6, new_img6, 3);

    writePlyheader();
    /* only need to calculate once */
    getVector();

    //mediablur
```

3. box filter

   計算每一點取強度時，與周遭 9 點計算卷積和並除以 9，將圖像平滑化

```
float boxfilter(Mat image, int row, int col){

    float sum=0;

    if(col-1>=0 && row-1>=0 && col+1<image.cols && row+1<image.rows){
        for(int i=row-1;i<row+2;i++){
            for(int j=col-1;j<col+2;j++){
                sum=sum+image.at<uchar>(i,j);
            }
        }
        sum = sum/9;
        if(sum>255){
            sum=255;
        }
        if(sum<0){
            sum=0;
        }
    }else{
        sum = image.at<uchar>(row,col);
    }
    cout <<"the point value is = "<<sum<<endl;

    return sum;
}
```

• 執行方式

1. 將程式碼開啟並載入 xcode，即可執行.cpp 檔(code 內路徑為/test/bunny 的路徑為範例)
2. 透過 command 編譯並執行.cpp 檔