```
#include <iostream>
#include <stdio.h>
#include <opencv2/opencv.hpp>
#include <math.h>

using namespace cv;
using namespace std;

int main(void)
{
    Mat img=imread("sat2.jpg",CV_LOAD_IMAGE_GRAYSCALE);
    int s[]={img.rows, img.cols, img.channels()};
    float hist[256];
    for(int i=0;i<256;i++)
    {
        hist[i]=0.0;
    }
    for(int i=0;i<s[0];i++)
    {
        for(int j=0;j<s[1];j++)
        {
            hist[img.at<uint8_t>(i,j)]+=1.0/(s[0]*s[1]);
        }
    }
    for(int i=1;i<256;i++)
    {
        hist[i]+=hist[i-1];
    }
    Mat img1(img.rows,img.cols,CV_8UC1);
    for(int i=0;i<img.rows;i++)
    {
        for(int j=0;j<img.cols;j++)
        {
            img1.at<uint8_t>(i,j)=hist[img.at<uint8_t>(i,j)]*255.0;
        }
    }

    int b_size=3;
    int n_cluster=4;
    int power=2;
    float epsilon=0.001;
    double cluster[n_cluster];
    for(int i=0;i<n_cluster;i++)
    {
        cluster[i]=i*255.0/n_cluster;
        cout<<cluster[i]<<" ";
    }
    cout<<endl;
    float mean=0.0;
    // distance
    float d[n_cluster];
    float w[n_cluster];
    float w_old[n_cluster];
    float sum[n_cluster];
    float num[n_cluster];
    for(int i=0;i<n_cluster;i++)
    {
        d[i]=0.0;
        w[i]=0.0;
        w_old[i]=0.0;
        sum[i]=0.0;
        num[i]=0.0;
    }
    float max_new=0.0;
    float max_old=0.0;
    int c=0;
    for(int p=0;p<5;p++)
    {
        max_new=0.0;
        for(int i=b_size;i<s[0]-b_size;i++)
        {
            for(int j=b_size;j<s[1]-b_size;j++)
```

```
            {
                mean=0.0;
                // mean of the block
//              cout<<i<<" "<<j<<endl;
                c=0;
                for(int x=i-b_size;x<=i+b_size;x++)
                {
                    for(int y=j-b_size;y<=j+b_size;y++)
                    {
                        mean+=float(img.at<uint8_t>(x,y));
                        c++;
                    }
                }
                mean=mean/c;
//              cout<<mean<<endl;
//              int temp;
//              cin>>temp;
                // finding distance
                for(int x=0;x<n_cluster;x++)
                {
                    d[x]=abs(mean-cluster[x]);
                    if(d[x]<0.001)
                    {
                        d[x]=0.001;
                    }
                }
                // initialising weight
                for(int x=0;x<n_cluster;x++)
                {
                    w[x]=0.0;
                }

                for(int x=0;x<n_cluster;x++)
                {
                    for(int y=0;y<n_cluster;y++)
                    {
                        w[x]+=pow((float(d[x])/float(d[y])),power);
                    }
                    w[x]=1.0/w[x];
                    if(abs(w[x]-w_old[x])>max_new)
                        max_new=abs(w[x]-w_old[x]);
                    sum[x]+=pow(w[x],power)*mean;
                    num[x]+=pow(w[x],power);
                }
            }
        }
        cout<<"New cluster is"<<endl;
        for(int i=0;i<n_cluster;i++)
        {
            cluster[i]=sum[i]/num[i];
            cout<<cluster[i]<<" ";
            sum[i]=0.0;
            num[i]=0.0;
        }
        cout<<endl;
        cout<<max_new<<endl;
//      int temp;
//      cin>>temp;
//      if(max_new<epsilon)
//      {
//          break;
//      }
//      else
//      {
//
//      }
    }
    Mat im_c(img.rows,img.cols,CV_8UC1);
    float min=10000000000;
    int index=0;
    for(int i=b_size;i<s[0]-b_size;i++)
    {
```

```
        for(int j=b_size;j<s[1]-b_size;j++)
        {
            min=10000000000;
            mean=0.0;
            index=0;
            c=0;
            for(int x=i-b_size;x<=i+b_size;x++)
            {
                for(int y=j-b_size;y<=j+b_size;y++)
                {
                    mean+=float(img.at<uint8_t>(x,y));
                    c++;
                }
            }
            mean/=c;
//          cout<<mean<<endl;
            for(int x=0;x<n_cluster;x++)
            {
                if(abs(mean-cluster[x])<min)
                {
                    min=abs(mean-cluster[x]);
                    index=x;
                }
            }
//          cout<<index<<endl;
            im_c.at<uint8_t>(i,j)=index*255/n_cluster;
        }
    }
    imshow("Original",img);
    imshow("Cluster",im_c);
    waitKey(0);
    return 1;
}
```