#include "opencv2/core/core.hpp"

#include "opencv2/highgui/highgui.hpp"

#include "opencv2/imgproc/imgproc.hpp"

#include "filter.cpp"

#include <stdio.h>

#include <iostream>

#include <omp.h>

#include <ctime>

#include <string>

#include <vector>

#include <iostream>

#include <omp.h>

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//Namespaces

using cv::Mat;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//Global Variables

const unsigned int SLIDER\_MAX = 25;

const std::string WINDOW\_NAME = "KMeans program";

const std::string TRACKBAR\_NAME = "Clusters";

int MAX\_ITERATIONS = 50;

int sliderValue;

Mat image;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

\*/

void trackBarMovement(int, void\*);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int main () {

std::string imageName;

std::cout << "Enter the name of the image" << std::endl;

std::cin >> imageName;

image = imread(imageName, cv::IMREAD\_COLOR);

image = cv::imread("c:\\ban\_scorch.png");

if (!image.data){

std::cout << "Invalid name" << std::endl;

cv::destroyAllWindows();

return EXIT\_FAILURE;

}

cv::namedWindow(WINDOW\_NAME, cv::WINDOW\_AUTOSIZE);

cv::createTrackbar(TRACKBAR\_NAME,WINDOW\_NAME, &sliderValue, SLIDER\_MAX, trackBarMovement );

cv::MedianBlur( image, outputImage, cv::Size(3,3), 6.75);

cv::waitKey(0);

cv::destroyAllWindows();

return EXIT\_SUCCESS;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Mat applyKMeans(const Mat& source){

if (sliderValue==0){

sliderValue=1;

}

int start=clock();

const unsigned int singleLineSize = source.rows \* source.cols;

Mat data = source.reshape(1, singleLineSize);

data.convertTo(data, CV\_32F);

std::vector<int> labels;

cv::Mat1f colors;

#pragma omp parallel sections

{

cv::kmeans(data, sliderValue,labels,cv::TermCriteria(cv::TermCriteria::EPS+cv::TermCriteria::COUNT, 10, 1.),MAX\_ITERATIONS,cv::KMEANS\_PP\_CENTERS,colors);

#pragma omp parallel for

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

data.at<float>(i, 0) = colors(labels[i], 0);

data.at<float>(i, 1) = colors(labels[i], 1);

data.at<float>(i, 2) = colors(labels[i], 2);

}

}

Mat outputImage = data.reshape(3, source.rows);

outputImage.convertTo(outputImage, CV\_8U);

int end = clock();

int elapsedTime = (end-start)/double(CLOCKS\_PER\_SEC)\*1000;

std::cout<< "Execution time for processing with " << sliderValue <<" clusters " << elapsedTime << "ms" << std::endl;

return outputImage;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void trackBarMovement(int, void\*){

cv::imshow(WINDOW\_NAME, applyKMeans(image));

}