//#include "opencv2/imgproc/imgproc.hpp"

//#include <opencv2/highgui.hpp>

#include<opencv>

#include <iostream>

#include <string>

#include <stdio.h>

#include "parallel.cu"

#include "Serial.cpp"

using namespace std;

using namespace cv;

int MAX\_KERNEL\_LENGTH = 9;

struct timeval t1, t2;

void sharpeningFilter\_CPU(const Mat& input, Mat& output);

int main( int argc, char\*\* argv ) {

string image\_name = "sample";

string input\_file = image\_name+".jpeg";

string output\_file\_cpu = image\_name+"\_cpu.jpeg";

string output\_file\_gpu = image\_name+"\_gpu.jpeg";

// Read input image

Mat srcImage = imread(input\_file ,CV\_LOAD\_IMAGE\_UNCHANGED);

if(srcImage.empty())

{

cout<<"Image Not Found: "<< input\_file << endl;

return -1;

}

cout <<"\ninput image size: "<<srcImage.cols<<" "<<srcImage.rows<<" "<<srcImage.channels()<<"\n";

// Declare the output image

Mat dstImage (srcImage.size(), srcImage.type());

// run median filter on GPU

sharpeningFilter\_CPU(srcImage, dstImage);

imwrite(output\_file\_cpu, dstImage);

return 0;

}

// The wrapper is used to call sharpening filter

void sharpeningFilter\_CPU (const Mat& input, Mat& output)

{

Point anchor = Point( -1, -1 );

double delta = 0;

int ddepth = -1;

int kernel\_size;

int64 t0 = getTickCount();

/// Update kernel size for a normalized box filter

kernel\_size = 3;

Mat kernel = (Mat\_<double>(kernel\_size,kernel\_size) << -1, -1, -1, -1, 9, -1, -1, -1, -1);

// Apply 2D filter to image

filter2D(input, output, ddepth, kernel, anchor, delta, BORDER\_DEFAULT );

int64 t1 = getTickCount();

double secs = (t1-t0)/getTickFrequency();

cout<< "\nProcessing time on CPU (ms): " << secs\*1000 << "\n";

}