#include <stdio.h>

#include <stdlib.h>

#include <opencv/cv.h>

#include <opencv/highgui.h>

#include <emmintrin.h>

#include <pthread.h>

#define ALTOBLOQUE 16

#define ANCHOBLOQUE 16

#define NTHREADS 4

IplImage\* Img1;

IplImage\* Img2;

int b\_y;

int b\_x;

void copiarBloque(int x0, int y0, IplImage\* imagenOrigen, int x1, int y1, IplImage\* imagenDestino) {

for (int fila = 0; fila < ALTOBLOQUE; fila++) {

\_\_m128i \*pImag1 = (\_\_m128i \*) (imagenOrigen->imageData + ((y0 + fila) \* imagenOrigen->widthStep)+(x0 \* imagenOrigen->nChannels));

\_\_m128i \*pDestino = (\_\_m128i \*) (imagenDestino->imageData + ((fila + y1) \* imagenDestino->widthStep)+(x1 \* imagenDestino->nChannels));

//uchar \*pImag1 = imagenOrigen->imageData + ((y0 + fila) \* imagenOrigen->widthStep)+(x0 \* 3);

//uchar \*pDestino = imagenDestino->imageData + ((fila + y1) \* imagenDestino->widthStep)+(x1 \* 3);

for (int columna = 0; columna < ANCHOBLOQUE; columna += 4) {

\*pDestino++ = \*pImag1++;

}

}

cvShowImage("Destino", imagenDestino);

}

int compararBloque(int x0, int y0, IplImage\* imagenOrigen, int x1, int y1, IplImage\* imagenDestino) {

int dif = 0;

for (int fila = 0; fila < ALTOBLOQUE; fila++) {

\_\_m128i \*pImag1 = (\_\_m128i \*) (imagenOrigen->imageData + ((y0 + fila) \* imagenOrigen->widthStep)+(x0 \* imagenOrigen->nChannels));

\_\_m128i \*pDestino = (\_\_m128i \*) (imagenDestino->imageData + ((fila + y1) \* imagenDestino->widthStep)+(x1 \* imagenDestino->nChannels));

//uchar \*pImag1 = imagenOrigen->imageData + ((y0 + fila) \* imagenOrigen->widthStep)+(x0 \* 3);

//uchar \*pDestino = imagenDestino->imageData + ((fila + y1) \* imagenDestino->widthStep)+(x1 \* 3);

\_\_m128i A, B, C, D, E;

for (int columna = 0; columna < ANCHOBLOQUE; columna += 4) {

//VER PD SEMANA 9 SAD PARA ENTEDER//

A = \*pImag1++;

B = \*pDestino++;

C = \_mm\_sad\_epu8(A, B);

D = \_mm\_srli\_si128(C, 8);

E = \_mm\_add\_epi32(C, D);

dif = dif + abs(\_mm\_cvtsi128\_si32(E));

/\*

sustituye a esto:

dif = dif + abs(\*pDestino++ - \*pImag1++);

dif = dif + abs(\*pDestino++ - \*pImag1++);

dif = dif + abs(\*pDestino++ - \*pImag1++);

\*/

}

}

return dif;

}

void buscaBloque(int x0, int y0, IplImage\* imagenOrigen, IplImage\* imagenDestino, int\* x\_parecido, int\* y\_parecido) {

int dif;

int last\_dif = 3 \* 255 \* ALTOBLOQUE\*ANCHOBLOQUE;

for (int i = 0; i < b\_x; i++) {

for (int j = 0; j < b\_y; j++) {

dif = compararBloque(x0\*ANCHOBLOQUE, y0\*ALTOBLOQUE, imagenOrigen, i\*ANCHOBLOQUE, j\*ALTOBLOQUE, imagenDestino);

if (dif < last\_dif) {

last\_dif = dif;

\*x\_parecido = i;

\*y\_parecido = j;

}

}

}

}

void mosaico\_thread(void\*ptr) {

//ptr apunta a un entero que indica el mutiplo de la fila a sustituir

int \*fila\_inicial = (int\*) ptr; // TO DO

int x\_parecido;

int y\_parecido;

for (int fila=\*fila\_inicial; fila < \*fila\_inicial+(b\_y/NTHREADS); fila++) {

for(int col = 0; col < b\_x; col++){

buscaBloque(fila, col, Img2, Img1, &x\_parecido, &y\_parecido);

copiarBloque(x\_parecido\*ANCHOBLOQUE, y\_parecido\*ALTOBLOQUE, Img1, fila\*ANCHOBLOQUE, col\*ALTOBLOQUE, Img2);

}

}

}

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

if (argc != 3) {

printf("Usage: %s image\_file\_name\n", argv[0]);

return EXIT\_FAILURE;

}

//CV\_LOAD\_IMAGE\_COLOR = 1 forces the resultant IplImage to be colour.

//CV\_LOAD\_IMAGE\_GRAYSCALE = 0 forces a greyscale IplImage.

//CV\_LOAD\_IMAGE\_UNCHANGED = -1

IplImage\* Img2 = cvLoadImage(argv[2], CV\_LOAD\_IMAGE\_COLOR);

IplImage\* Img1 = cvLoadImage(argv[1], CV\_LOAD\_IMAGE\_COLOR);

if (!Img1) {

printf("Error: fichero %s no leido\n", argv[1]);

return EXIT\_FAILURE;

}

IplImage\* Img1\_4ch = cvCreateImage(cvSize(Img1->width, Img1->height), 8, 4);

cvCvtColor(Img1, Img1\_4ch, CV\_BGR2BGRA);

IplImage\* Img2\_4ch = cvCreateImage(cvSize(Img2->width, Img2->height), 8, 4);

cvCvtColor(Img2, Img2\_4ch, CV\_BGR2BGRA);

//IplImage\* Destino = cvCreateImage(cvSize(Img1->width, Img1->height), Img1->depth, Img1->nChannels);

srand((unsigned) time(0));

int x0, y0, x\_parecido, y\_parecido;

int b\_y = (Img1->height / ALTOBLOQUE);

int b\_x = (Img1->width / ANCHOBLOQUE);

int array[b\_x][b\_y];

memset(array, 0, sizeof (array));

cvNamedWindow("Img1", CV\_WINDOW\_AUTOSIZE);

cvNamedWindow("Destino", CV\_WINDOW\_AUTOSIZE);

int filas[NTHREADS];

pthread\_t threads[NTHREADS];

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

filas[i] = i;

printf("\nEl thread %d sustituye las filas %d", i, filas[i]);

pthread\_create(&threads[i], NULL, (void\*) &mosaico\_thread, (void\*) &filas[i]);

}

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

pthread\_join(threads[i], NULL);

}

/\*

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

mosaico\_thread(threads[i], Img1, Img2, Img1\_4ch, Img2\_4ch);

}

\*/

/\*

int cuenta = b\_x\*b\_y;

while (cuenta > 0) {

x0 = (random() % b\_x);

y0 = (random() % b\_y);

if (!array[x0][y0]) {

buscaBloque(x0, y0, Img2, Img1, &x\_parecido, &y\_parecido);

copiarBloque(x\_parecido\*ANCHOBLOQUE, y\_parecido\*ALTOBLOQUE, Img1\_4ch, x0\*ANCHOBLOQUE, y0\*ALTOBLOQUE, Img2\_4ch);

array[x0][y0] = 1;

cuenta--;

cvWaitKey(1);

cvShowImage("Img1", Img1);

}

}

\*/

cvWaitKey(0);

// a visualization window is created with title 'image'

// memory release for img before exiting the application

cvReleaseImage(&Img1);

cvReleaseImage(&Img2);

// cvReleaseImage(&Destino);

// Self-explanatory

cvDestroyWindow(argv[1]);

cvDestroyWindow(argv[2]);

return EXIT\_SUCCESS;

}