#include <stdio.h>

#include <stdlib.h>

#include<time.h>

#include <opencv/cv.h>

#include <opencv/highgui.h>

#include<pthread.h> /\*POSIX Threads\*/

#define NTHREADS 8

#define ALTOBLOQUE 16

#define ANCHOBLOQUE 16

int b\_x, b\_y;

IplImage\* Img1;

IplImage\* Img2;

void mosaico\_thread(void\*ptr) {

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

int \*fila = (int\*) ptr;

int x\_parecido, y\_parecido;

for (int y0 = \*fila; y0 < \*fila + (b\_y / NTHREADS); y0++) {

for (int x0 = 0; x0 < b\_x; x0++) {

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

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

}

}

}

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

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

\_\_m128i \*pImg1 = (\_\_m128i\*) (imagenOrigen->imageData + (fila + y0) \* imagenOrigen->widthStep + x0 \* imagenOrigen->nChannels);

\_\_m128i \*pImg2 = (\_\_m128i\*) (imagenDestino->imageData + (fila + y1) \* imagenDestino->widthStep + x1 \* imagenDestino->nChannels);

for (int col = 0; col < ANCHOBLOQUE \* imagenOrigen->nChannels; col += 16) {

\*pImg2++ = \*pImg1++;

}

}

}

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

int dif = 0;

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

\_\_m128i \*pImg1 = (\_\_m128i\*) (imagenOrigen->imageData + ((fila + y0) \* imagenOrigen->widthStep) + (x0 \* imagenOrigen->nChannels));

\_\_m128i \*pImg2 = (\_\_m128i\*) (imagenDestino->imageData + ((fila + y1) \* imagenDestino->widthStep) + (x1 \* imagenDestino->nChannels));

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

for (int col = 0; col < ANCHOBLOQUE \* imagenOrigen->nChannels; col += 16) {

A = \*pImg1++;

B = \*pImg2++;

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));

}

}

return dif;

}

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

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

int dif;

int b\_x = imagenOrigen->width / ANCHOBLOQUE;

int b\_y = imagenOrigen->height / ALTOBLOQUE;

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

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

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

if (dif < minor\_dif) {

\*x\_parecido = i;

\*y\_parecido = j;

minor\_dif = dif;

}

}

}

}

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

if (argc != 3) {

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

return EXIT\_FAILURE;

}

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

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

// Always check if the program can find a file

if (!Img1) {

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

return EXIT\_FAILURE;

}

if (!Img2) {

printf("Error: fichero %s no leido\n", argv[2]);

return EXIT\_FAILURE;

}

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

//cvNamedWindow(argv[1], CV\_WINDOW\_AUTOSIZE);

//cvNamedWindow(argv[2], CV\_WINDOW\_AUTOSIZE);

// img is shown in 'image' window

b\_x = Img1->width / ANCHOBLOQUE;

b\_y = Img1->height / ALTOBLOQUE;

pthread\_t threads[NTHREADS];

int filas[NTHREADS];

int i;

struct timespec start, finish;

float elapsed;

clock\_gettime(CLOCK\_MONOTONIC, &start);

/\*CODE to be measured\*/

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

filas[i] = i \* (b\_y / NTHREADS);

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

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

}

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

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

}

clock\_gettime(CLOCK\_MONOTONIC, &finish);

elapsed = (finish.tv\_sec - start.tv\_sec);

elapsed += (finish.tv\_nsec - start.tv\_nsec) / 1000000000.0;

printf("\nTiempo transcurrido: %f", elapsed);

cvShowImage(argv[1], Img1);

cvShowImage(argv[2], Img2);

//cvWaitKey(0);

// memory release for img before exiting the application

cvReleaseImage(&Img1);

cvReleaseImage(&Img2);

// Self-explanatory

cvDestroyWindow(argv[1]);

cvDestroyWindow(argv[2]);

return EXIT\_SUCCESS;

}