#include <stdio.h>

#include <stdlib.h>

#include <opencv/cv.h>

#include <opencv/highgui.h>

#define ALTOBLOQUE 16

#define ANCHOBLOQUE 16

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

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

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++) {

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

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

\*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++) {

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++) {

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;

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

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

}

}

}

}

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[1], CV\_LOAD\_IMAGE\_COLOR);

IplImage\* Img1 = cvLoadImage(argv[2], CV\_LOAD\_IMAGE\_COLOR);

if (!Img1) {

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

return EXIT\_FAILURE;

}

//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);

for (y0 = 0; y0 < b\_y; y0++) {

for (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);

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;

}