#include "get\_date\_roi.h"

#include "loc\_detect.h"

#include "ocr.h"

#include <chrono>

int main() {

string path = "C://Users//pc//Desktop//img//3.bmp";

Mat src = imread(path, 1);

std::chrono::high\_resolution\_clock::time\_point beginTime = std::chrono::high\_resolution\_clock::now();

Mat roi\_img;

get\_date\_roi get\_date;

get\_date.do\_get\_date\_roi(src, roi\_img);

//imshow("roi\_img", roi\_img);

std::chrono::high\_resolution\_clock::time\_point endTime = std::chrono::high\_resolution\_clock::now();

std::chrono::milliseconds timeInterval = std::chrono::duration\_cast<std::chrono::milliseconds>(endTime - beginTime);

std::cout << "do\_get\_date\_roi:" << timeInterval.count() << endl;

Mat date\_img;

cvtColor(roi\_img, roi\_img, COLOR\_BGR2GRAY);

loc\_detect loc;

loc.image\_cut\_process(roi\_img, date\_img);

//imshow("date\_img", date\_img);

//waitKey(0);

endTime = std::chrono::high\_resolution\_clock::now();

timeInterval = std::chrono::duration\_cast<std::chrono::milliseconds>(endTime - beginTime);

std::cout << "image\_cut\_process:" << timeInterval.count() << endl;

FuncOnnx\_ocr ocr;

ocr.loadFuncOnnx\_ocr("C://Users//pc//Desktop//img//torch\_model\_train\_Acc100.000\_test\_Acc99.567\_2\_3.onnx");

string predict\_date;

ocr.ocr\_predict(date\_img, predict\_date);

cout << predict\_date << endl;

endTime = std::chrono::high\_resolution\_clock::now();

timeInterval = std::chrono::duration\_cast<std::chrono::milliseconds>(endTime - beginTime);

std::cout << "ocr\_predict:" << timeInterval.count() << endl;

system("pause");

return 0;

}