package org.bytedeco.javacv.android.recognize.technowings;

import org.bytedeco.javacpp.BytePointer;

import org.bytedeco.javacpp.FloatPointer;

import org.bytedeco.javacpp.IntPointer;

import org.bytedeco.javacpp.Loader;

import org.bytedeco.javacv.\*;

import static org.bytedeco.javacpp.opencv\_core.\*;

import static org.bytedeco.javacpp.opencv\_imgproc.\*;

import static org.bytedeco.javacpp.opencv\_video.\*;

import static org.bytedeco.javacpp.opencv\_highgui.\*;

import static org.bytedeco.javacpp.opencv\_imgcodecs.\*;

public class OpticalFlowTracker {

private static final int MAX\_CORNERS = 500;

public static void main(String[] args) {

// Load two images and allocate other structures

IplImage imgA = cvLoadImage(

"D:\\work\\project\\OMRProject\\PersonCounting-Eclipse\\Test\\database\\filename009.bmp",

CV\_LOAD\_IMAGE\_GRAYSCALE);

IplImage imgB = cvLoadImage(

"D:\\work\\project\\OMRProject\\PersonCounting-Eclipse\\Test\\database\\filename010.bmp",

CV\_LOAD\_IMAGE\_GRAYSCALE);

CvSize img\_sz = cvGetSize(imgA);

int win\_size = 15;

// IplImage imgC = cvLoadImage("OpticalFlow1.png",

// CV\_LOAD\_IMAGE\_UNCHANGED);

IplImage imgC = cvLoadImage(

"image0.png",

CV\_LOAD\_IMAGE\_UNCHANGED);

// Get the features for tracking

IplImage eig\_image = cvCreateImage(img\_sz, IPL\_DEPTH\_32F, 1);

IplImage tmp\_image = cvCreateImage(img\_sz, IPL\_DEPTH\_32F, 1);

IntPointer corner\_count = new IntPointer(1).put(MAX\_CORNERS);

CvPoint2D32f cornersA = new CvPoint2D32f(MAX\_CORNERS);

CvArr mask = null;

cvGoodFeaturesToTrack(imgA, eig\_image, tmp\_image, cornersA,

corner\_count, 0.05, 5.0, mask, 3, 0, 0.04);

cvFindCornerSubPix(imgA, cornersA, corner\_count.get(),

cvSize(win\_size, win\_size), cvSize(-1, -1),

cvTermCriteria(CV\_TERMCRIT\_ITER | CV\_TERMCRIT\_EPS, 20, 0.03));

// Call Lucas Kanade algorithm

BytePointer features\_found = new BytePointer(MAX\_CORNERS);

FloatPointer feature\_errors = new FloatPointer(MAX\_CORNERS);

CvSize pyr\_sz = cvSize(imgA.width() + 8, imgB.height() / 3);

IplImage pyrA = cvCreateImage(pyr\_sz, IPL\_DEPTH\_32F, 1);

IplImage pyrB = cvCreateImage(pyr\_sz, IPL\_DEPTH\_32F, 1);

CvPoint2D32f cornersB = new CvPoint2D32f(MAX\_CORNERS);

cvCalcOpticalFlowPyrLK(imgA, imgB, pyrA, pyrB, cornersA, cornersB,

corner\_count.get(), cvSize(win\_size, win\_size), 5,

features\_found, feature\_errors,

cvTermCriteria(CV\_TERMCRIT\_ITER | CV\_TERMCRIT\_EPS, 20, 0.3), 0);

// Make an image of the results

for (int i = 0; i < corner\_count.get(); i++) {

if (features\_found.get(i) == 0 || feature\_errors.get(i) > 550) {

System.out.println("Error is " + feature\_errors.get(i) + "/n");

continue;

}

System.out.println("Got it/n");

cornersA.position(i);

cornersB.position(i);

CvPoint p0 = cvPoint(Math.round(cornersA.x()),

Math.round(cornersA.y()));

CvPoint p1 = cvPoint(Math.round(cornersB.x()),

Math.round(cornersB.y()));

cvLine(imgC, p0, p1, CV\_RGB(255, 0, 0),

2, 8, 0);

}

cvSaveImage(

"image0-1.png",

imgC);

cvNamedWindow( "LKpyr\_OpticalFlow", 0 );

cvShowImage( "LKpyr\_OpticalFlow", imgC );

cvWaitKey(0);

}

}