import org.opencv.core.Core;

import org.opencv.core.CvType;

import org.opencv.core.DMatch;

import org.opencv.core.Mat;

import org.opencv.core.MatOfDMatch;

import org.opencv.core.MatOfKeyPoint;

import org.opencv.features2d.DescriptorExtractor;

import org.opencv.features2d.DescriptorMatcher;

import org.opencv.features2d.FeatureDetector;

import org.opencv.imgcodecs.Imgcodecs;

public class Mini {

public static void main(String[] args) {

// Set image path

String filename="C:/Users/Akshay G/Desktop/0.jpg";

String filename1 = "C:/Users/Akshay G/Desktop/1.jpg";

String filename2 = "C:/Users/Akshay G/Desktop/2.jpg";

String filename3 = "C:/Users/Akshay G/Desktop/3.jpg";

String filename4 = "C:/Users/Akshay G/Desktop/4.jpg";

String filename5 = "C:/Users/Akshay G/Desktop/5.jpg";

String filename6 = "C:/Users/Akshay G/Desktop/6.jpg";

int count=0,count1=0;

int ret[]=new int[6];

ret[0] = compareFeature(filename, filename1);

ret[1] = compareFeature(filename, filename2);

ret[2] = compareFeature(filename, filename3);

ret[3] = compareFeature(filename, filename4);

ret[4] = compareFeature(filename, filename5);

ret[5] = compareFeature(filename, filename6);

for(int i=0;i<=5;i++)

{

if(i<=2)

{

if(ret[i]>0)

count++;

}

if(i>2)

{

if(ret[i]>0)

count1++;

}

}

if(count+count1>0)

System.out.println("it's going to rain");

if(count>count1)

System.out.println("it's cumulonimbus");

else if(count<count1)

System.out.println("it's nimbostartus");

else

System.out.println("as if understood you what nimbostratus is R??");

// if (ret > 0) {

// System.out.println("Two images are same.");

// } else {

// System.out.println("Two images are different.");

// }

}

/\*\*

\* Compare that two images is similar using feature mapping

\* @author minikim

\* @param filename1 - the first image

\* @param filename2 - the second image

\* @return integer - count that has the similarity within images

\*/

public static int compareFeature(String filename1, String filename2) {

int retVal = 0;

long startTime = System.currentTimeMillis();

System.loadLibrary(Core.NATIVE\_LIBRARY\_NAME);

// Load images to compare

Mat img1 = Imgcodecs.imread(filename1, Imgcodecs.CV\_LOAD\_IMAGE\_COLOR);

Mat img2 = Imgcodecs.imread(filename2, Imgcodecs.CV\_LOAD\_IMAGE\_COLOR);

// Declare key point of images

MatOfKeyPoint keypoints1 = new MatOfKeyPoint();

MatOfKeyPoint keypoints2 = new MatOfKeyPoint();

Mat descriptors1 = new Mat();

Mat descriptors2 = new Mat();

// Definition of ORB key point detector and descriptor extractors

FeatureDetector detector = FeatureDetector.create(FeatureDetector.ORB);

DescriptorExtractor extractor = DescriptorExtractor.create(DescriptorExtractor.ORB);

// Detect key points

detector.detect(img1, keypoints1);

detector.detect(img2, keypoints2);

// Extract descriptors

extractor.compute(img1, keypoints1, descriptors1);

extractor.compute(img2, keypoints2, descriptors2);

// Definition of descriptor matcher

DescriptorMatcher matcher = DescriptorMatcher.create(DescriptorMatcher.BRUTEFORCE\_HAMMING);

// Match points of two images

MatOfDMatch matches = new MatOfDMatch();

// System.out.println("Type of Image1= " + descriptors1.type() + ", Type of Image2= " + descriptors2.type());

// System.out.println("Cols of Image1= " + descriptors1.cols() + ", Cols of Image2= " + descriptors2.cols());

// Avoid to assertion failed

// Assertion failed (type == src2.type() && src1.cols == src2.cols && (type == CV\_32F || type == CV\_8U)

if (descriptors2.cols() == descriptors1.cols()) {

matcher.match(descriptors1, descriptors2 ,matches);

// Check matches of key points

DMatch[] match = matches.toArray();

double max\_dist = 0; double min\_dist = 100;

for (int i = 0; i < descriptors1.rows(); i++) {

double dist = match[i].distance;

if( dist < min\_dist ) min\_dist = dist;

if( dist > max\_dist ) max\_dist = dist;

}

//System.out.println("max\_dist=" + max\_dist + ", min\_dist=" + min\_dist);

// Extract good images (distances are under 10)

for (int i = 0; i < descriptors1.rows(); i++) {

if (match[i].distance <= 65) {

retVal++;

}

}

// System.out.println("matching count=" + retVal);

}

long estimatedTime = System.currentTimeMillis() - startTime;

//System.out.println("estimatedTime=" + estimatedTime + "ms");

return retVal;

}

}