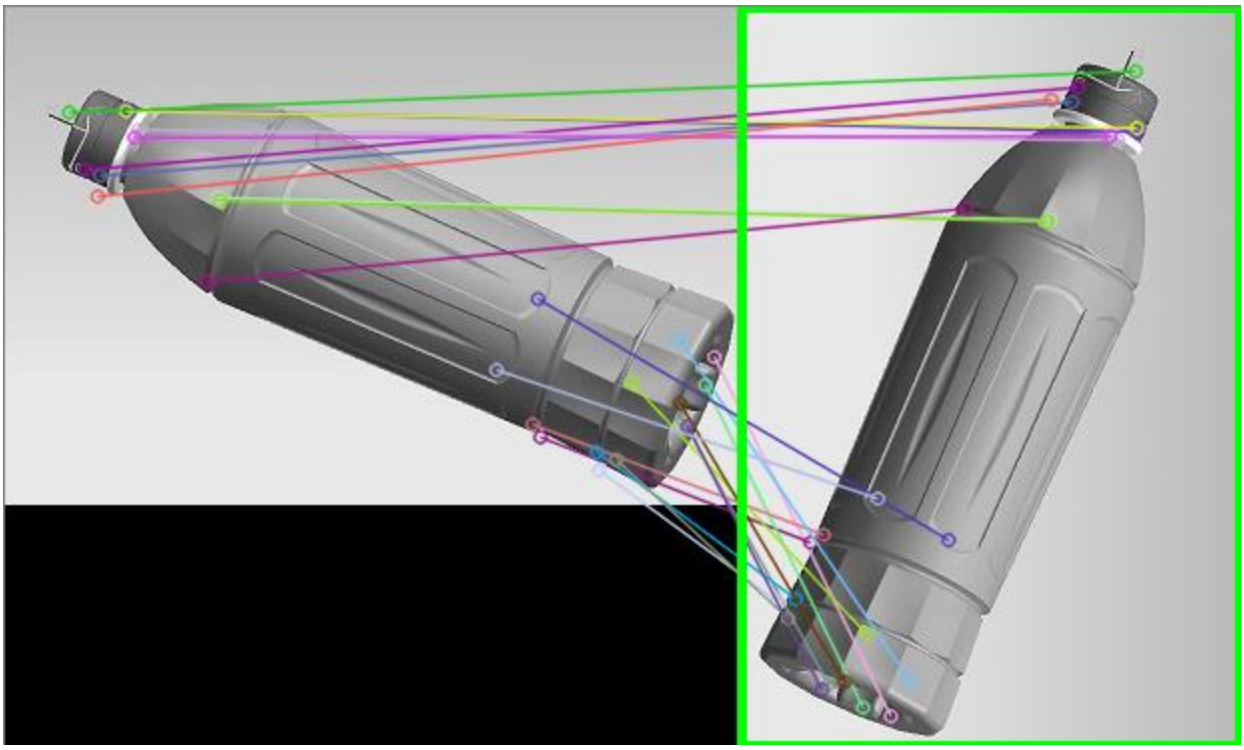
[CTT451] – [Nhập môn Thị giác Máy tính]

Tháng 5/2013

COMPUTATION HOMOGRAPHY



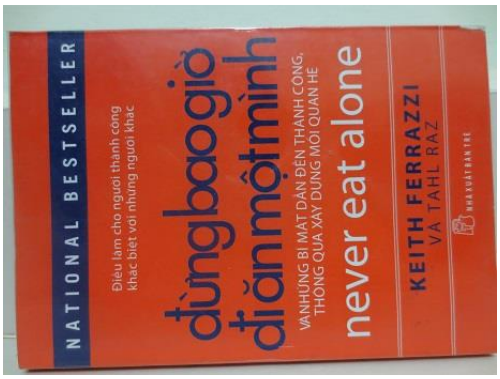
Bộ môn TGMT và KH Rô-bốt
Khoa Công nghệ thông tin
ĐH Khoa học tự nhiên TP HCM

MỤC LỤC

1 Ứng dụng Homography

Tìm một đối tượng đã biết trước

Cho đối tượng sau:



Tìm đối tượng trên trong hình sau:

Bước 1: phát hiện các keypoints sử dụng đặc trưng SURF

Bước 2: từ tập các keypoints được phát hiện trên, tính toán các vector đặc trưng (vector miêu tả).

Bước 3: đối sánh các vector miêu tả sử dụng FLANN matcher.

- Vẽ các đối sánh tốt nhất: là các đối sánh nhỏ hơn ngưỡng
- Xác định đối tượng trong ảnh

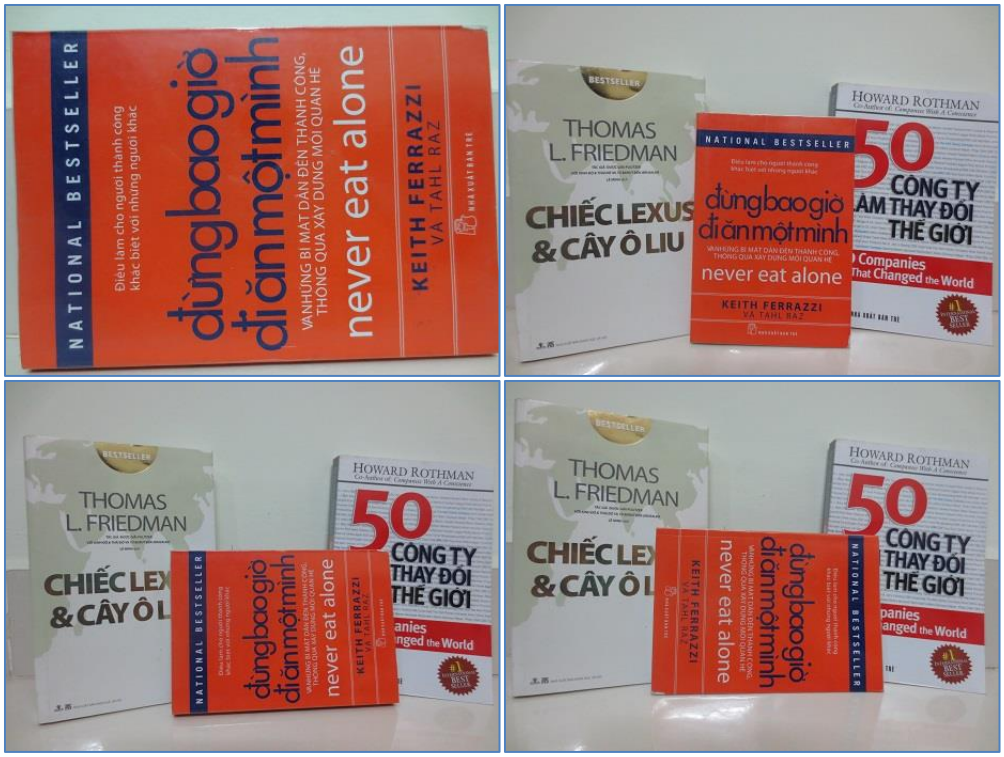
2 Chương trình

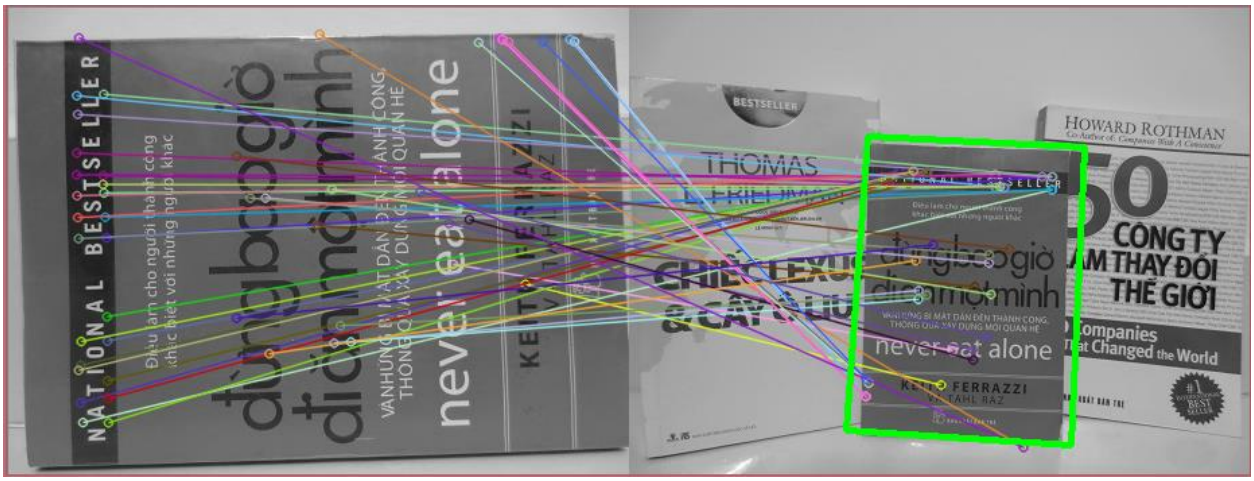
```
#include "stdio.h"
#include "iostream"

#include "opencv/cv.h"
#include "opencv/highgui.h"
```

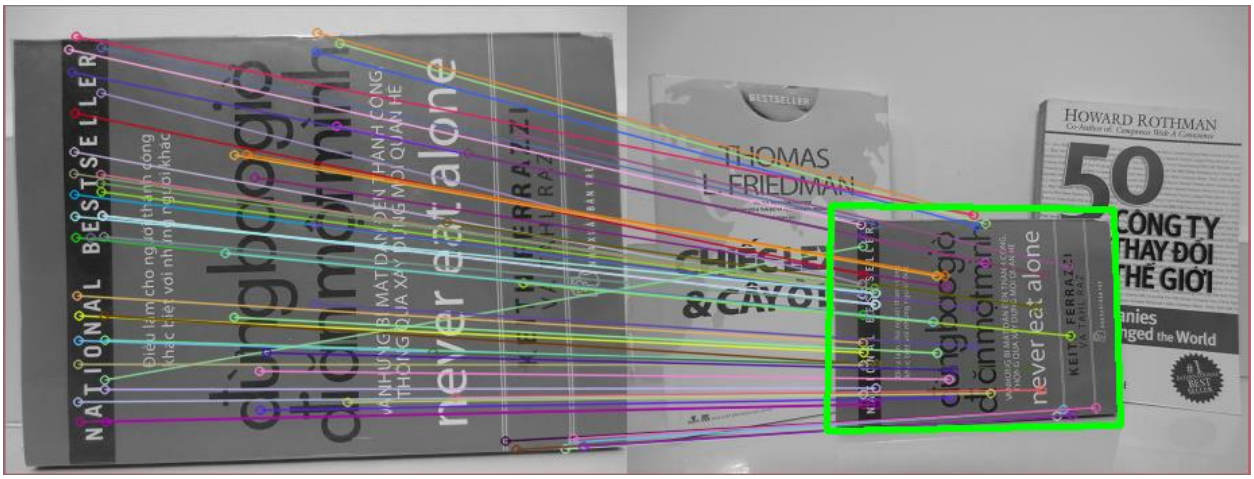
```
#include "opencv2/nonfree/features2d.hpp"
#include "opencv2/legacy/legacy.hpp"
#include "opencv2/core/core.hpp"

using namespace cv;

/** @function main */
int main( int argc, char** argv )
{
	Mat img_object = imread( "C:\\OPENCV245\\Test\\3.jpg", CV_LOAD_IMAGE_GRAYSCALE );
	Mat img_scene = imread( "C:\\OPENCV245\\Test\\5.jpg", CV_LOAD_IMAGE_GRAYSCALE );

	if( !img_object.data || !img_scene.data )
	{ std::cout<< " --(!) Error reading images " << std::endl; return -1; }

	//-- Step 1: Detect the keypoints using SURF Detector
	int minHessian = 400;
	//SURF surf (minHessian);
	SurfFeatureDetector detector(minHessian, 4, 2, true, false );

	std::vector<KeyPoint> keypoints_object, keypoints_scene;

	detector.detect( img_object, keypoints_object );
	detector.detect( img_scene, keypoints_scene );


	//-- Step 2: Calculate descriptors (feature vectors)
	SurfDescriptorExtractor extractor;

	Mat descriptors_object, descriptors_scene;

	extractor.compute( img_object, keypoints_object, descriptors_object );
	extractor.compute( img_scene, keypoints_scene, descriptors_scene );

	//-- Step 3: Matching descriptor vectors using FLANN matcher
	FlannBasedMatcher matcher;
	std::vector< DMatch > matches;
	matcher.match( descriptors_object, descriptors_scene, matches );

	double max_dist = 0; double min_dist = 100;

	//-- Quick calculation of max and min distances between keypoints
	for( int i = 0; i < descriptors_object.rows; i++ )
	{ double dist = matches[i].distance;
	if( dist < min_dist ) min_dist = dist;
	if( dist > max_dist ) max_dist = dist;
	}

	printf("-- Max dist : %f \n", max_dist );
	printf("-- Min dist : %f \n", min_dist );

	//-- Draw only "good" matches (i.e. whose distance is less than 3*min_dist )
	std::vector< DMatch > good_matches;

	for( int i = 0; i < descriptors_object.rows; i++ )
	{ if( matches[i].distance < 3*min_dist )
	{ good_matches.push_back( matches[i]); }
```

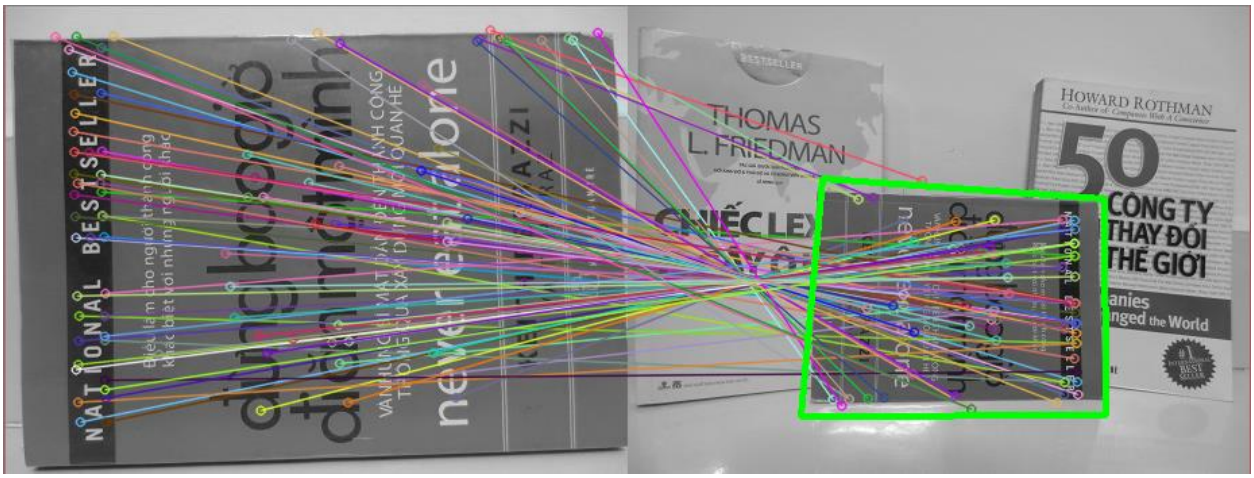
```
	}

	Mat img_matches;
	drawMatches( img_object, keypoints_object, img_scene, keypoints_scene,
		good_matches, img_matches, Scalar::all(-1), Scalar::all(-1),
		vector<char>(), DrawMatchesFlags::NOT_DRAW_SINGLE_POINTS );

	//-- Localize the object
	std::vector<Point2f> obj;
	std::vector<Point2f> scene;

	for( int i = 0; i < good_matches.size(); i++ )
	{
		//-- Get the keypoints from the good matches
		obj.push_back( keypoints_object[ good_matches[i].queryIdx ].pt );
		scene.push_back( keypoints_scene[ good_matches[i].trainIdx ].pt );
	}

	Mat H = findHomography( obj, scene, CV_RANSAC );

	//-- Get the corners from the image_1 ( the object to be "detected" )
	std::vector<Point2f> obj_corners(4);
	obj_corners[0] = cvPoint(0,0); obj_corners[1] = cvPoint( img_object.cols, 0 );
	obj_corners[2] = cvPoint( img_object.cols, img_object.rows ); obj_corners[3] =
cvPoint( 0, img_object.rows );
	std::vector<Point2f> scene_corners(4);

	perspectiveTransform( obj_corners, scene_corners, H);

	//-- Draw lines between the corners (the mapped object in the scene - image_2 )
	line( img_matches, scene_corners[0] + Point2f( img_object.cols, 0),
scene_corners[1] + Point2f( img_object.cols, 0), Scalar(0, 255, 0), 4 );
	line( img_matches, scene_corners[1] + Point2f( img_object.cols, 0),
scene_corners[2] + Point2f( img_object.cols, 0), Scalar( 0, 255, 0), 4 );
	line( img_matches, scene_corners[2] + Point2f( img_object.cols, 0),
scene_corners[3] + Point2f( img_object.cols, 0), Scalar( 0, 255, 0), 4 );
	line( img_matches, scene_corners[3] + Point2f( img_object.cols, 0),
scene_corners[0] + Point2f( img_object.cols, 0), Scalar( 0, 255, 0), 4 );

	//-- Show detected matches
	imshow( "Good Matches & Object detection", img_matches );

	waitKey(0);
	return 0;
}
```

Chạy chương trình

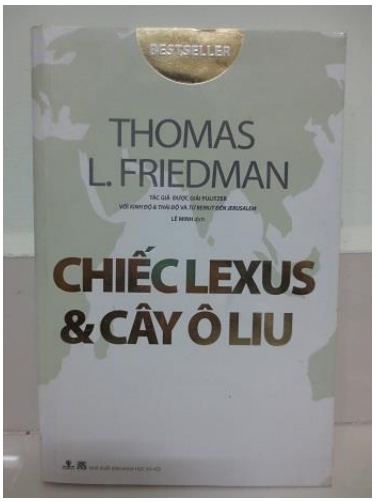

NATIONAL BESTSELLER
dừng bao giờ dẫn một mình
never eat alone
KEITH FERRAZZI
VÀ TAHL RAZ
THOMAS L. FRIEDMAN
CHIẾC LEXUS & CÂY Ô LIU
HOWARD ROTHMAN
50 CÔNG TY THAY ĐỔI THẾ GIỚI

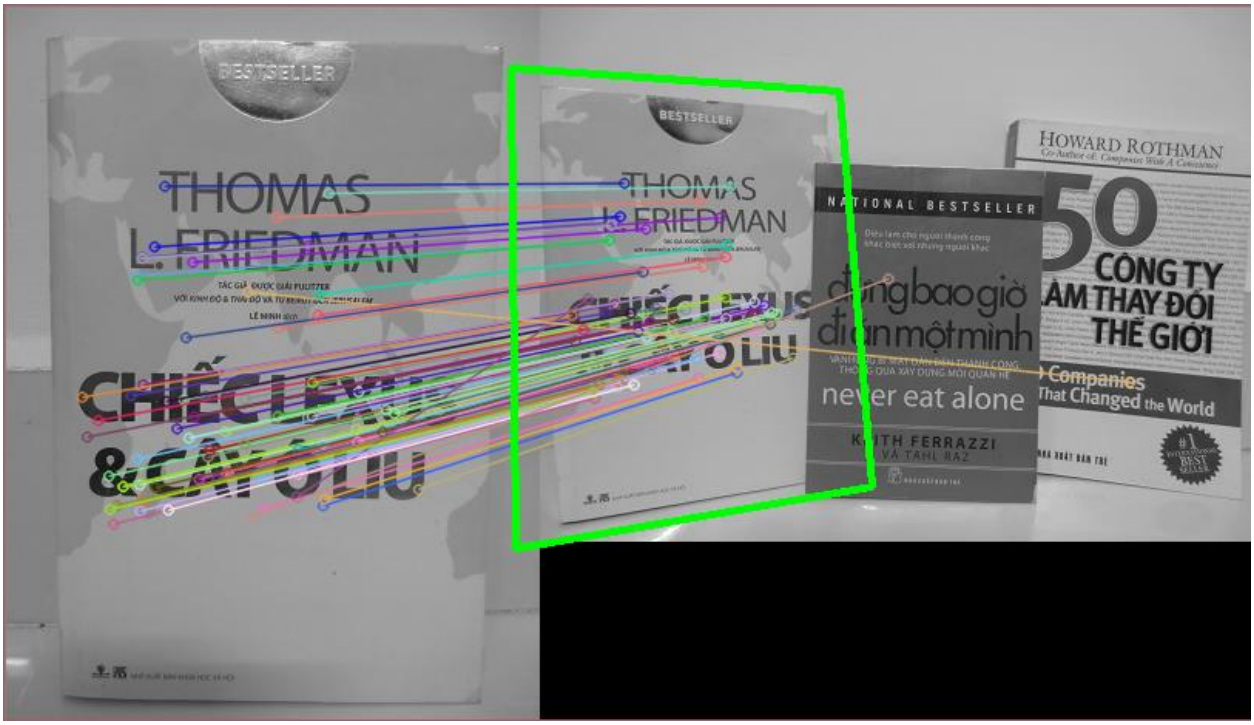

NATIONAL BESTSELLER
never eat alone
THOMAS
HOWARD ROTHMAN
50
CÔNG TY
THAY ĐỔI
THẾ GIỚI





Đối tượng thứ 2:




BESTSELLER
THOMAS
L. FRIEDMAN
BESTSELLER
THOMAS
L. FRIEDMAN
NATIONAL BESTSELLER
never eat alone
HOWARD ROTHMAN
50
CÔNG TY
ĐỔI THẾ GIỚI
Companies
That Changed the World