Home
Announcements and Updates
Tutorial: Object Recognition With OpenCV and Android
Old Notes and Research
Links
Recent Updates
Contact
Sitemap

Recent site activity
Project Remake Fail
created by Aaron Forster
It's The Amps
created by Aaron Forster
OK, Where Is The Camera?
created by Aaron Forster
Home
edited by Aaron Forster
OMGHOLYCRAPWOW Maker Faire
created by Aaron Forster
View All

1067
days since
Project Launch

Tutorial 1: Object Recognition With OpenCV and Android - Overview of Object Recognition

posted Mar 7, 2012, 12:50 PM by Aaron Forster **[updated Mar 7, 2012, 2:01 PM]**

Part 1 of several. Overview here.

First lets get specific. We're talking about Object Recognition not Object Detection though the terms are mistakenly used synonymously. I myself had to update the title of this tutorial once I started writing the overview. Object detection is often used in things like robotics for object avoidance or for measurements. iRobot's Roomba does a great job of object detection but it can't tell you the difference between a hat and a handbag or even between dark and light.

Object Recognition is some times called Computer Vision though Computer Vision really refers to a larger problem space which includes Object Detection, Object Tracking, Object Recognition and probably a few things I'm leaving out. In this article I am specifically talking about two dimensional object recognition. 3D object recognition is a less developed technology though recent advances have taken it leaps and bounds ahead. The folks at Willow Garage have some great work on that subject and one of their child projects OpenCV has some capabilities there as well as the 2d work we will be using it for but they will not be discussed further.

Computer vision and object recognition is an extremely difficult problem. It is a problem space that I have seen described in multiple locations as "Impossible to solve." The human brain without concious input does a great deal of what amounts to some very complex mathimatical equations. Catching a ball, for example, is a calculation which includes trajectory, velocity and mass. We do it in real time. Most computers these days can do it but most can't accomplish it in real time the way that the human brain can.

As a whole this is a non-trivial project. Fortunately people smarter than me have been working on it for a long time. We will stand on their shoulders and make it look easy but let us take a moment to acknowledge the amazing work that has been done not only by the team that comprises Willow Garage but by the other researchers that they have built upon. From the research on computer vision we can step back through the many researchers who have developed different image recognition algorithms and technologies. To the components of those who have developed edge detection, color intensity determination and signal processing. Finally at it's base we get to the pure mathmatical research that has produced Fast Furrier Transforms and Haar Wavelets and many other algorithms and functions without which none of this would be possible. The research that has taken us this far encompasses thousands of human minds over centuries of work.

All so that I can keep my cat out of the house using a device that fits in my pocket.

Since this problem is not new researchers have and continue to develop new methodologies to tackle the challenge. The links here, largely to wikipedia, are informational and not required for this project.

Some popular techniques in use today:

Edge detection: Determining the boundries between objects is accomplished via a number of techniques including
    Canny Edge Detection
    Hugh Transforms which actually extends into
Blob detection: Taking edge detection to the next step blob detection is the art of identifying discrete sections of an image such as a hand, a teacup or a car. These blobs can then be used later to make decisions about the content of the image.
Corner detection: Is another technique used to break the problem up into multiple pieces which can be simplified. It is obviously very useful in determining basic shapes such as squares, triangles and circles (because they have zero) but most things have corners on them in some form or another. Which is probably why these two combine and overlap with.
Interest Point Detection: Which roughly involves determining regions of the image which can be used to identify an object (potentially a blob) such as bright spots, dark spots, corners, intersecting lines or just concrete variations in something (such as intensity) that the computer can latch on to and use to match images.

Many of these are used to comprise more specific image to image matching techniques (Here is a picture of a cup. Here is an image with a cup in it somewhere. Now identify the cup.) such as:
Speeded Up Robust Feature (SURF)
Local Binary Patterns
and
Haar Cascades

OpenCV has functions that include all of the above. Ultimately the sample code here is based on Haar Cascades because it was the one that I found the most samples for and the only one that I understood well enough to implement. Code that includes it is reasonably simple though the training process involved (covered in detail later) can be a bit more complex. While I found a great deal of code that covered edge detection and blob detection most sample code merely draws boxes or circles around these objects and does nothing to describe a way to match that against some previously identified sample edge or blob.

Sample code should work with only minor adjustments since we are using OpenCV's CascadeClassifier class which supports both.

Continue to part two.

Comments