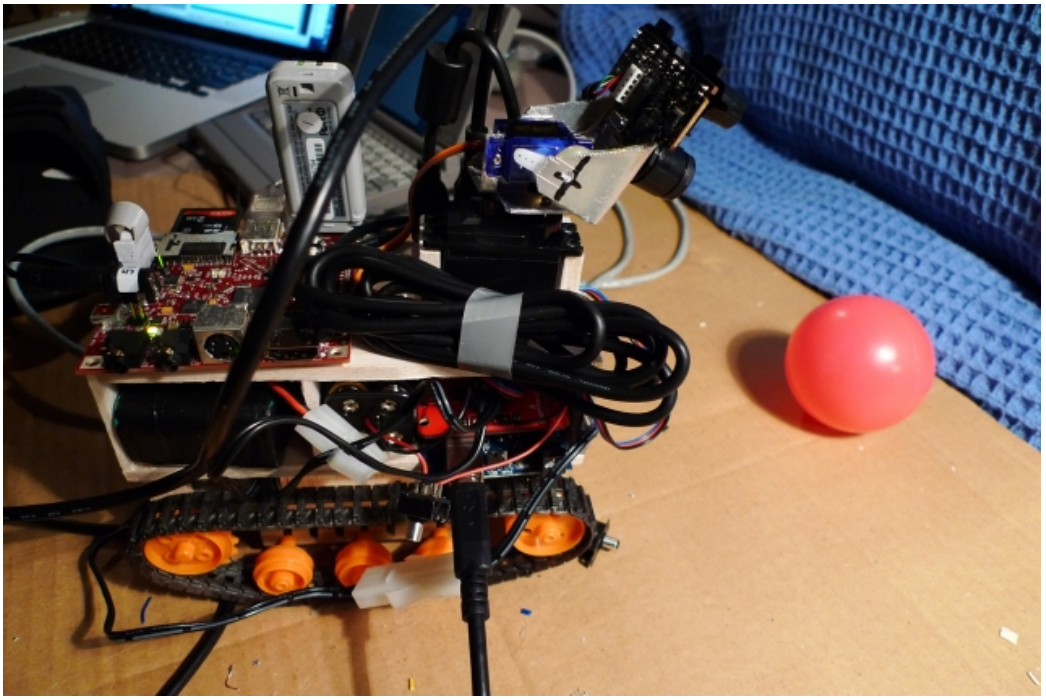
User login

Username: *

Using OpenCV on the Beagleboard to track an Aibo pink ball

Home



With OpenCV and the Playstation Eye running on the Beagleboard robot, it's time to try some image processing. Tracking a ball is an interesting task to start with, as the bright pink ball from an Aibo robot dog is fairly easy to identify. Eventually the robot will be able to find and play with the ball like an Aibo.

There are many image processing techniques that can be used, however I'm starting with a very basic approach:

- convert the color space from RGB to HSV
- threshold the appropriate hue (pink!), saturation and value (brightness)
- use a Hough circle detector to identify circles within the thresholded image

Some test C code outputs both the thresholded image, and the original image with the ball marked:

```
// Detecting a pink Aibo ball
// Copyright 2009 mechomaniac.com

#include "opencv/cvaux.h"
#include "opencv/highgui.h"
#include "opencv/cxcore.h"
#include <stdio.h>

int main(int argc, char* argv[])
{
    CvCapture* camera = cvCreateCameraCapture(-1); // Use the default camera

    IplImage* frame = 0;
```

Password: *

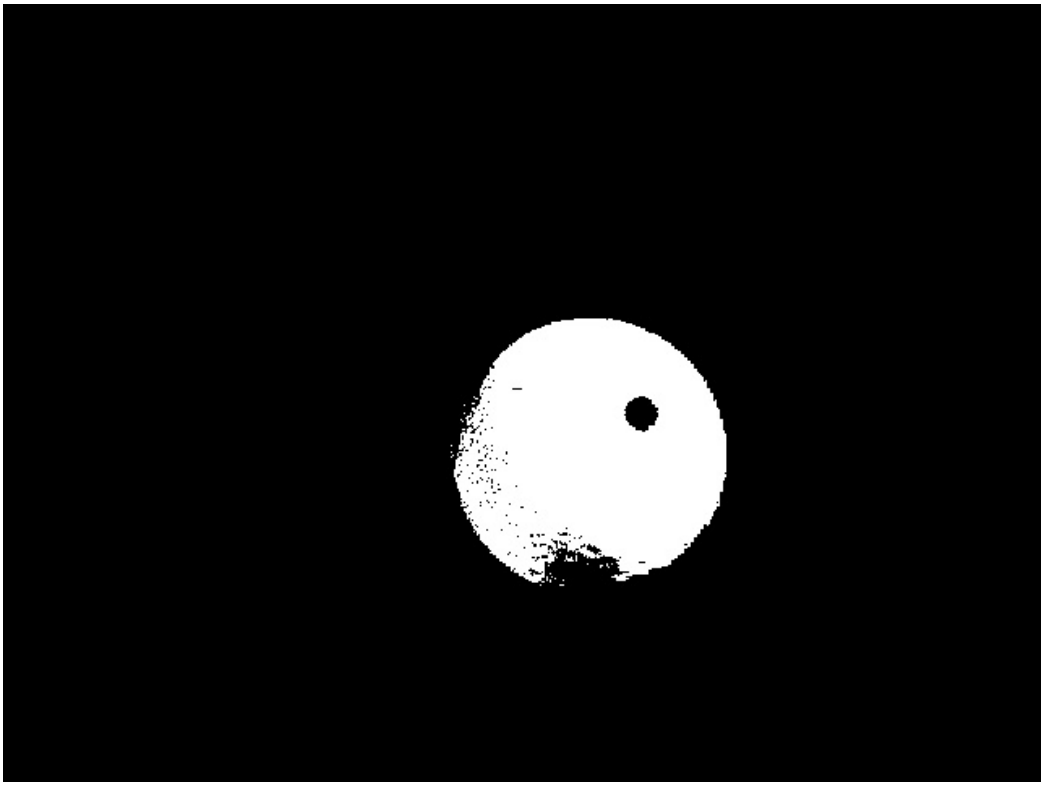
Log in

- Request new password

```
    CvMemStorage* storage = cvCreateMemStorage(0); //needed for Hough circles

    // capturing some extra frames seems to help stability
    frame = cvQueryFrame(camera);
    frame = cvQueryFrame(camera);
    frame = cvQueryFrame(camera);

    // with default driver, PSEye is 640 x 480
    CvSize size = cvSize(640,480);
    IplImage *  hsv_frame    = cvCreateImage(size, IPL_DEPTH_8U, 3);
    IplImage*  thresholded    = cvCreateImage(size, IPL_DEPTH_8U, 1);
    IplImage*  thresholded2    = cvCreateImage(size, IPL_DEPTH_8U, 1);

    CvScalar hsv_min = cvScalar(0, 50, 170, 0);
    CvScalar hsv_max = cvScalar(10, 180, 256, 0);
    CvScalar hsv_min2 = cvScalar(170, 50, 170, 0);
    CvScalar hsv_max2 = cvScalar(256, 180, 256, 0);

    //do {
        frame = cvQueryFrame(camera);
        if (frame != NULL) {
            printf("got frame\n\r");
            // color detection using HSV
            cvCvtColor(frame, hsv_frame, CV_BGR2HSV);
            // to handle color wrap-around, two halves are detected and combined
            cvInRangeS(hsv_frame, hsv_min, hsv_max, thresholded);
            cvInRangeS(hsv_frame, hsv_min2, hsv_max2, thresholded2);
            cvOr(thresholded, thresholded2, thresholded);

            cvSaveImage("thresholded.jpg",thresholded);

            // hough detector works better with some smoothing of the image
            cvSmooth( thresholded, thresholded, CV_GAUSSIAN, 9, 9 );
            CvSeq* circles = cvHoughCircles(thresholded, storage, CV_HOUGH_GRADIENT, 2, th

                    for (int i = 0; i < circles->total; i++)
                    {
                        float* p = (float*)cvGetSeqElem( circles, i );
                        printf("Ball! x=%f y=%f r=%f\n\r",p[0],p[1],p[2] );
                            cvCircle( frame, cvPoint(cvRound(p[0]),cvRound(p[1])),
                                      3, CV_RGB(0,255,0), -1, 8, 0 );
                            cvCircle( frame, cvPoint(cvRound(p[0]),cvRound(p[1])),
                                      cvRound(p[2]), CV_RGB(255,0,0), 3, 8, 0 );
                     }

            cvSaveImage("frame.jpg", frame);
        } else {
            printf("Null frame\n\r");
        }
    //} while (true);
    cvReleaseCapture(&camera);
    return 0;
}
```

The results show that the ball has been identified:

The thresholded image shows the result of the HSV thresholding to detect just the pink colored ball:



With some basic vision code running, it's now possible for the robot to attempt to track the ball. To do this, the Beagleboard needs to send commands to the Arduino to move servos so that the camera will attempt to move as the ball moves, keeping it in frame.

For faster development and easier serial port programming, I've switched to Python. OpenCV provides an excellent Python interface and since all of the CPU intensive work is being done by OpenCV, there's not much performance difference.

I've only got a single servo connected at the moment, to pan the camera. The control algorithm is also very primitive, attempting to keep the ball roughly in the middle of the frame:

```
// Tracking a pink Aibo ball
// Copyright 2009 mechomaniac.com

from opencv.cv import *
from opencv.highgui import *
import serial

ser = serial.Serial('/dev/ttyUSB0', 9600, timeout=1)
servoPos = 90

def servo(id, position):
    ser.write("#S" + str(id) + str(position) + "#")

size = cvSize(640, 480)
hsv_frame = cvCreateImage(size, IPL_DEPTH_8U, 3)
thresholded = cvCreateImage(size, IPL_DEPTH_8U, 1)
thresholded2 = cvCreateImage(size, IPL_DEPTH_8U, 1)

hsv_min = cvScalar(0, 50, 170, 0)
hsv_max = cvScalar(10, 180, 256, 0)
hsv_min2 = cvScalar(170, 50, 170, 0)
hsv_max2 = cvScalar(256, 180, 256, 0)

storage = cvCreateMemStorage(0)

# start capturing form webcam
capture = cvCreateCameraCapture(0)
#cvSetCaptureProperty(capture, CV_CAP_PROP_FRAME_WIDTH, 320)
#cvSetCaptureProperty(capture, CV_CAP_PROP_FRAME_HEIGHT, 240)
#cvSetCaptureProperty(capture, CV_CAP_PROP_FPS, 15)


if not capture:
    print "Could not open webcam"
    sys.exit(1)

while 1:
    # get a frame from the webcam
    frame = cvQueryFrame(capture)

    if frame is not None:
    #cvSaveImage("test.jpg", frame)

        # convert to HSV for color matching
        # as hue wraps around, we need to match it in 2 parts and OR together
        cvCvtColor(frame, hsv_frame, CV_BGR2HSV)
        cvInRangeS(hsv_frame, hsv_min, hsv_max, thresholded)
        cvInRangeS(hsv_frame, hsv_min2, hsv_max2, thresholded2)
```

```
        cvOr(thresholded, thresholded2, thresholded)

        # pre-smoothing improves Hough detector
        cvSmooth(thresholded, thresholded, CV_GAUSSIAN, 9, 9)
        circles = cvHoughCircles(thresholded, storage, CV_HOUGH_GRADIENT, 2, thresholded.h

        # find largest circle
        maxRadius = 0
        x = 0
        y = 0
        found = False
        for i in range(circles.total):
            circle = circles[i]
            if circle[2] > maxRadius:
                found = True
                maxRadius = circle[2]
                x = circle[0]
                y = circle[1]

        if found:
            print "ball detected at position:",x, ",", y, " with radius:", maxRadius

            if x > 420:
                # need to pan right
                servoPos += 5
                servoPos = min(140, servoPos)
                servo(2, servoPos)
            elif x < 220:
                servoPos -= 5
                servoPos = max(40, servoPos)
                servo(2, servoPos)
            print "servo position:", servoPos
        else:
            print "no ball"

ser.close()
```

Design by niGraphic