Robotik: Ball Tracking with NAO's Head

Dr.-Ing. John Nassour

November 16, 2015

1 The Goal

The main goal of this task is to establish a bridge between visual target detection and motion module in order to make NAO robot track a ball with specific color by moving his head Pitch and Yaw motors.

2 Task Description

2.1 Step.1

- Familiarize yourself with getting pictures remotely by Nao's cameras. Try get pictures with different resolutions, different color space, and different fps. For more information about the supported colorspaces and resolution please refer to the following link: `http://doc.aldebaran.com/2-1/family/robots/video_robot.html`
- Example [Retrieving images]: The aim of this example is to show how to retrieve images from NAO's cameras and visualize an image using PIL.
  `http://doc.aldebaran.com/2-1/dev/python/examples/vision/get_image.html#get-an-image`
- Familiarize yourself with different vision examples. `http://doc.aldebaran.com/2-1/dev/python/examples/vision/index.html`
- OpenCV (Open Computer Vision) is a C++ library containing various state-of-the-art vision algorithms, from object recognition to video analysis, image processing etc..
  Helpful links:
  `http://docs.opencv.org/doc/tutorials/tutorials.html`
  `http://doc.aldebaran.com/2-1/dev/cpp/examples/vision/opencv.html?highlight=vision%20opencv`

2.2 Step.2

Your task is to track a ball with specific color by controlling the two head's joint of NAO robot in order to keep the center of the tracked ball in the middle of the camera view.**Note: In this task it is not allowed to use "ALRedBallTracker" for your algorithm.**

- *Different colors*: Your project must consider tracking balls from different colors (red, yellow, blue). It is suggested to use HSV colorspace for color detection.
- *Different resolutions*: Perform tracking of one selected ball (one selected color) with different camera resolutions (at least three different resolutions). Measure the performance of the tracking with each resolution.
- *Performance*: Calculate the position of the center of the ball in pixel coordinates. Record "HeadPich" and "HeadYaw" angels measured by the sensors (e.g.: getangle()). Figure out the ball positions and the angles values for the head after terminating the experiment.

3 Start with This:

The following code is available for you to be used for ball detection. You can use it and build your task basing on it. It is recomended to use SCITE editor for better visualization of your python code.

```
import cv2
import imutils
from imutils.object_detection import non_max_suppression
from imutils import paths
import numpy as np
from naoqi import ALProxy
from vision_definitions import kQVGA,kBGRColorSpace

NAO="nao.local"

if __name__=="__main__":    # this is to check if we are importing

    camera_index=0 # top camera

    # http://colorizer.org/
    # define the lower and upper boundaries of the "yellow"
    # ball in the HSV color space, then initialize the
    yellowLower = (25, 86, 6)
    yellowUpper = (35, 255, 255)

    colorLower = yellowLower
    colorUpper = yellowUpper

    # Create a proxy for ALVideoDevice
    name="nao_opencv"
    video=ALProxy("ALVideoDevice",NAO,9559)
```

```
# subscribe to video device on a specific camera # BGR for opencv
name=video.subscribeCamera(name,camera_index,kQVGA,kBGRColorSpace,30)
print "subscribed name",name

try:
    frame=None
    # keep looping
    while True:
        key=cv2.waitKey(33)&0xFF
        if  key == ord('q') or key==27:
            break

        # obtain image
        alimg=video.getImageRemote(name)

        # extract fields
        width=alimg[0]
        height=alimg[1]
        nchannels=alimg[2]
        imgbuffer=alimg[6]

        # build opencv image (allocate on first pass)
        if frame is None:
            print 'Grabbed image: ',width,'x',height,' nchannels=',nchannels
            frame=np.asarray(bytearray(imgbuffer), dtype=np.uint8)
            frame=frame.reshape((height,width,3))
        else:
            frame.data=bytearray(imgbuffer)

        # Smoothing Images
        # http://docs.opencv.org/master/d4/d13/tutorial_py_filtering.html#gsc.tab=0
        blurred = cv2.GaussianBlur(frame, (11, 11), 0)
        # Converts an image from one color space to another
        #http://docs.opencv.org/master/df/d9d/tutorial_py_colorspaces.html#gsc.tab=0
        hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)

        # construct a mask for the color, then perform
        #  a series of dilations and erosions to remove any small
        # blobs left in the mask
        mask = cv2.inRange(hsv, colorLower, colorUpper)
        mask = cv2.erode(mask, None, iterations=2)
        mask = cv2.dilate(mask, None, iterations=2)

        # find contours in the mask and initialize the current
        # (x, y) center of the ball
        cnts = cv2.findContours(mask.copy(), cv2.RETR_EXTERNAL,
            cv2.CHAIN_APPROX_SIMPLE)[-2]
        center = None

        # only proceed if at least one contour was found
        if len(cnts) > 0:
            # find the largest contour in the mask, then use
            # it to compute the minimum enclosing circle and
            # centroid
            c = max(cnts, key=cv2.contourArea)
            ((x, y), radius) = cv2.minEnclosingCircle(c)
            M = cv2.moments(c)
            center = (int(M["m10"] / M["m00"]), int(M["m01"] / M["m00"]))

            # only proceed if the radius meets a minimum size
            if radius > 10:
```

```
                # draw the circle and centroid on the frame,
                # then update the list of tracked points
                cv2.circle(frame, (int(x), int(y)), int(radius),
                    (0, 255, 255), 2)
                cv2.circle(frame, center, 5, (0, 0, 255), -1)
            # show the frame to our screen
            # Do not run this code if your run your python in the robot
            # NAO has no screen to show
            cv2.imshow("Frame", frame)

    finally: # if anything goes wrong we'll make sure to unsubscribe
        print "unsubscribing",name
        video.unsubscribe(name)
```

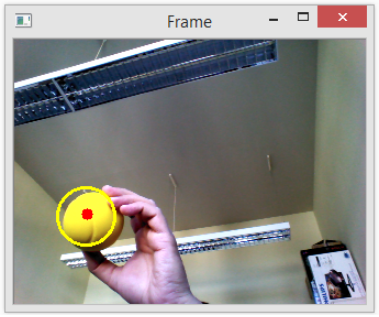

Figure 1: Output of color detection python example.

4 Results to Submit

- A PDF report containing description, comments, and screenshot of each subtask. Please use the template.
- Source code and the project directory. Not the binaries, etc.
- Note: Comment your code. Also, write your names at the beginning of the source code file.
- Make a demo video showing the results.
- Compress all the required results into a .zip or .tar.gz file.
- Naming convention: LastName1_LastName2_GroupNo_TaskNo.
  Example: If two students from group "Y" Franz Müller and Peter Maier work together on task 2, then they submit the file Müller_Maier_GY_T2.zip
  or
  Müller_Maier_GY_T2.tar.gz.
  Don't send the video file if it is too big. You can send a link on it [max size 20 MB].

  Submit the file containing your results before the deadline to: john.nassour@informatik.tu-chemnitz.de

5 Deadline

Dec. 5th 2014