EduFill Tutorial
Colour Calibration
for Cube Detection Block

July 25, 2013

0.1 Introduction

This document describes color calibration procedure that may be necessary to perform in such cases when standard calibration shipped with EduFill distribution shows poor detection performance. Color calibration tool is included in the EduFill distribution, so that its users could resolve issues of varying illumination conditions and particular color tones of the cubes used in particular classes where the cube detection might possibly be used.

Essentially, colour calibration saves colour range values for particular cubes in particular environment into file named **cur_min_max_hsv.yaml**. That file is later chosen by cube detector node if custom calibration is requested. If custom calibration settings are not specified for the node, the default calibration is used. See Section 0.2.5 for details.

Calibration procedure consists of the scene arrangement, connecting Kinect sensor to a PC with installed EduFill software and running colour calibration tool.

0.2 Calibration Procedure

This section describes the calibration requirements and procedure.

0.2.1 Required equipment and objects

Calibration procedure requires the following items:

1. Kinect **camera** (or a robot with mounted Kinect camera)
2. Appropriate **workspace**
3. Uniformly coloured **cubes** (cubes of size $2x2x2$ cm. are assumed by default cube detection settings)
4. **PC** with installed EduFill software.

0.2.2 Calibration Modes

It is possible to perform calibration either from the camera or from RGB image file with the scene. If you already have the image file of the scene with the cubes, skip the next subsection and proceed with Section 0.2.4.

0.2.3 Calibration Steps with Camera

The following steps must be performed for scene preparation and camera start-up.

1. Open several windows (or tabs) with terminal emulator, **gnome-terminal** on the PC. It is more convenient to run each continuous shell command in a different terminal. Here different terminal windows will be referred to as capital T and number (e.g. $T1$ or $T2$) to distinguish between them

2. Connect the camera to the PC via USB port and connect the camera to the power. Make sure the camera's LED is blinking green

3. In terminal $T1$ run the following command to start ROS nodes responsible for capturing data from the camera:

   ```
   $ roslaunch edufill_object_detection openni.launch
   ```

   Wait till the nodes are finished launching. The following message will signify the nodes have launched successfully:

   ```
   [ WARN] [1374137239.778621498]: Using default parameters for IR camera calibration.
   ```

4. **Scene preparation**
   The **workspace** for calibration should be the same as the one which will be used for cube detection during Blockly program execution. For YouBot it is natural to have a horizontal pallet of height about $10cm$ measured from the ground.

5. **Camera placement and orientation**
   Place and orient the camera such that it observes the working space. Viewing image feedback from the camera simplifies this task. Run the following command in terminal $T2$ to get image data:

   ```
   rosrun image_view image_view image:=/camera/rgb/image\_rect\_colour
   ```

6. **Cube placement**
   Place the cubes on the workspace in such a way, that they are fully observable from the camera point of view, e.g. every cube is not occluded by other objects or other cubes. See Figure 1 for an example of a workspace with cubes. When the camera and objects poses adjustment is finished, press $Ctrl + C$ in terminal $T2$.

0.2.4 Running Calibration Tool

Calibration tool is a ROS node and part of **edufill_object_detection** package, as such it must be executed through the corresponding launch file. Perform the following steps in order to accomplish colour calibration.

- In a free terminal ($T1$ if camera was not used or $T2$ otherwise) run the following command to launch the calibration tool:

  ```
  roslaunch edufill_object_detection calibrate_colour.launch
  ```

  At the end of the node initialization the following request appears in the terminal:

  ```
  Calibrate from /tower_cam3d/rgb/image_colour [y/<another_topic>/f]?
  ```

  Here the user has to choose the source of colour image data, which can be: **ROS topics** or **image file**:

  - to use **ROS topics**, make sure **openni.launch** is executed (see Subsection 0.2.3). If the camera is connected to a PC with standard ROS distribution, type */camera/rgb/image_rect_colour*. If *y* is selected instead of */camera/rgb/image_rect_colour*, then */tower_cam3d/rgb/image_colour* - default image topic name for YouBot is used;
  - to use **image file**, enter *f*, then confirm pressing *Enter*, then enter full name of the file with calibration image and press
  - press *Enter* to confirm the selected image source

- The user is then asked whether he wants to make a back-up of the calibration results:

  ```
  params/cur_min_max_hsv.yaml already exist. Make a backup [y/n]?
  ```

  Current and backed-up calibration files are stored in *params* subdirectory of *edufill_object_detection* directory. Backed-up files have postfix with a time-stamp, e.g. *cur_min_max_hsv.yaml_*1371586640.57. Input *y* or *n* if the most recently created custom calibration is needed or not needed respectively. Confirm the input by pressing *Enter*

- Then a window with an image of the scene with the cubes appear. The user has to calibrate each cube - which will be used for further detection - by its colour. Calibration consists of **zooming** the area of a particular cube and **selecting** the contour of the cube:
  - **zooming**: Perform rectangular selection of the cube sub-image as in a raster graphics editor (left-click at a point which is close to left-upper part of the cube, drug down-right, release the button) necessary for these two coincide with any corners or edge points of the cube, the points should form a diagonal of a rectangular containing the cube)

- In the zoomed window draw a rough **contour** of the object by dragging the mouse cursor over the cube with left mouse button pressed. Refer to Figure 2 for an example of a zoomed cube image with a rough contour around it. Notice that the contour is very rough and even discontinuous, but that is not an issue for the calibration tool. But try not to capture pixels outside the cube. When the contour is roughly closed, release the mouse button and press *Enter*

- Switch the active window to the terminal, where the calibration launch command (*roslaunchedufill_object_detectioncalibrate_colour.launch*) command was given. Last line of the terminal is request for colour name:

  ```
  colour [r, g, b, c, m, y]>
  ```

  Here *r* stands for *red*, *g* for *green*, *b* for *blue*, *c* for cyan, *m* for *magenta* and *y* for *yellow*. Enter the letter corresponding to the current cube selection and acknowledge pressing *Enter*

- The scene image will appear again. Proceed with the next cube

- After all cubes' colours are calibrated using the above steps, press *Esc* in the window with the scene image.

0.2.5 Choosing between Default and Custom Calibration settings

Default calibration settings are stored in *params* sub-directory of *edufill_object_detection* package under name *def_min_max_hsv.yaml*. This default calibration was created using scene image stored in *data/def_calibration_image.jpg*. By default, i.e. without parameters to the launch file for the detection node, this calibration settings are used:

```
$ roslaunch edufill_object_detection cube_color_detector.launch
```

Custom calibration is selected by appending calibration switch parameter to the launch file:

```
$ roslaunch edufill_object_detection cube_color_detector.launch  default_hsv_file:=false
```

0.2.6 Testing calibration

After the cube detection node is launched (see Subsection 0.2.5) it is possible to test if calibration results in proper detections without running any Blockly code by calling the corresponding service. In a free terminal run one of the following command:
If the camera is used:

```
$ rosservice call /edufill_objdetector/detect_cubes "{color: 'green',
min_size: 10, max_size: 50}"
```

If the scene image file is used:

```
$ rosservice call /edufill_objdetector/detect_cubes "{color: 'green',
min_size: 10, max_size: 50, image_fname: 'IMAGE_FILE'}"
```

Where $IMAGE\_FILE$ must be substituted with the full path name to the scene image file.
min_size and *max_size* parameters designate minimum and maximum size of the cube in the image plane in pixels and those values should be changed depending on the size of cubes and distance from the camera to the objects. The given values were selected for typical grasp scenario with YouBot and cubes of the size $2x2x2$ cm. with somewhat relaxed margins for size/distance variations. These values can be assessed by first measuring side of a cube on the scene image in pixels. The following heuristic rule then can be used to calculate *min_size* and *max_size*:
if the measured size is S pixels, and $D = k \cdot S$, where $0 < k < 1$, then $min\_size = S - D$ and $max\_size = S + D$. For typical YouBot scenarios $k = 0.5$ is sufficient.

The output the *detect_cubes* service call with the default scene image file should look like the following:

```
poses:
  -
    header:
      seq: 0
```

```
      stamp:
        secs: 0
        nsecs: 0
      frame_id: ''
    pose:
      position:
        x: 236.0
        y: 426.0
        z: 0.0
      orientation:
        x: 0.0
        y: 0.0
        z: 0.0
        w: 0.0
sizes: [19]
```

This output means there is one green cube instance detected, and its coordinates in the image plane in pixels are $(x : 236, y : 426)$. These values can be then approximately compared to the image coordinates of the cube center on the given image as shown by any raster graphics editor with coordinates display capability, e.g. **kolourpaint**. Note that for detection with camera, position of the cube is given with respect to the camera frame of reference and measured in meters, thus testing calibration with the scene image is more convenient.
If any of x or y are equal to -1, it means that no cubes of the given colour are detected.

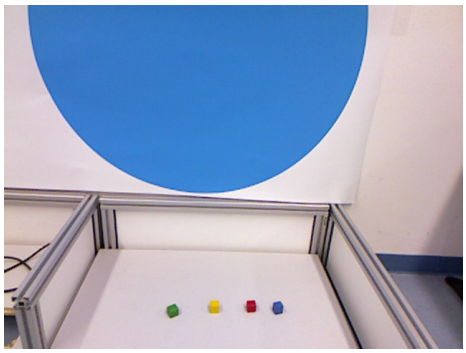

Figure 1: Example of a workspace scene

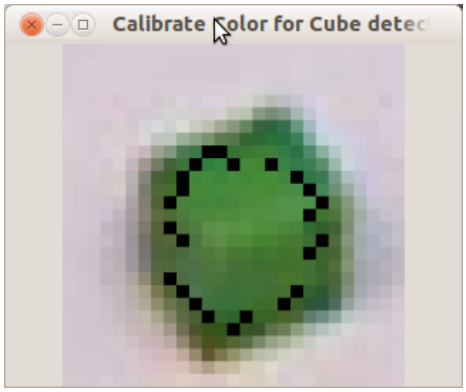

Figure 2: Contour around the cube's pixels