Playercam, Blobcolors and YUV color space

LAB 2 : ROBOT VISION

Lab Outline

- New config file
- colors.txt
- YUV color space
- playercam/playercam_yuv
- Color Calibration
- blobs

New Config File

- The first step towards letting your robot see the world
- Two devices need to be specified:
  - camerauvc: enables camera proxy that provides direct access to the camera image
  - cmvision: enables “blobfinder” proxy that processes images provided by “camera” proxy and groups like-colored pixels of a particular color into regions/blobs

```
driver
(
  name: “camerauvc”
 provides [“camera:0”]
)
driver
(
 name: “cmvision”
 provides [“blobfinder:0”]
 requires [“camera:0”]
 colorfile “colors.txt”
)
```

  - cameracompress (optional device): uses compressed video stream

```
driver
(
  name “cameracompress”
  provides [“camera:1”]
  requires [“camera:0”]
  image_quality 0.3
)
```

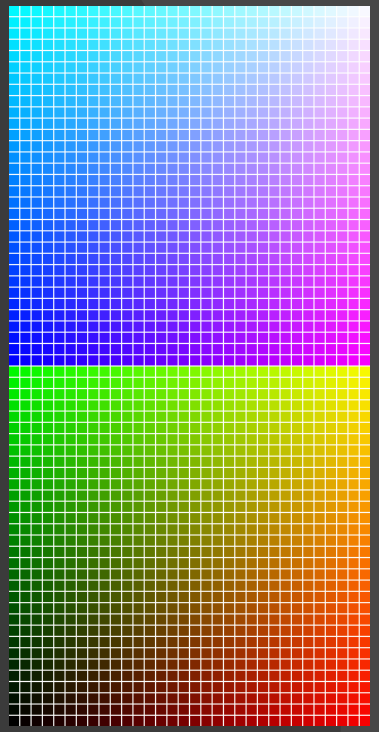
Color calibration file

- “blobfinder” provides bounding box around each image region containing a specified colors of interest.
- “cmvision” device knows about the colors of interest through color calibration file, or “colorfile”.
- The colorfile contains two sections, listing each blob color in sequential order:
  - “[Colors]” section specifying identifiers for each color
  - “[Thresholds]” section defines YUV color ranges associated with each blob color

Color calibration file

- Sample colors.txt file:
  [Colors]
  (255, 0, 0) 0.000000 10 Red

  [Thresholds]
  (25:164, 80:120, 150:240)



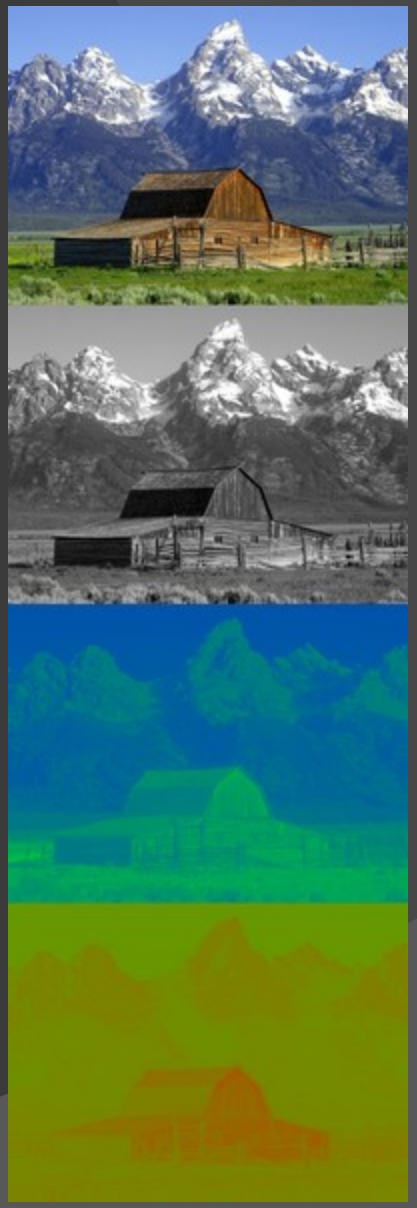
- Using this colorfile, any pixel with YUV values within this range will be labelled as the blob color “Red”.
- To calibrate the blobfinder, you need to add appropriate lines in your colorfile containing identifiers and YUV thresholds.

YUV color space

- YUV and RGB color coordinates are vastly different representations.
- Colors in YUV specify hue, saturation and intensity.
- Conversion between YUV and RGB:

$$\begin{bmatrix} Y' \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.14713 & -0.28886 & 0.436 \\ 0.615 & -0.51499 & -0.10001 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1.13983 \\ 1 & -0.39465 & -0.58060 \\ 1 & 2.03211 & 0 \end{bmatrix} \begin{bmatrix} Y' \\ U \\ V \end{bmatrix}$$



http://en.wikipedia.org/wiki/YUV

playercam/playercam_yuv

- Use playercam to color calibrate:

  playercam -h 10.100.0.x -p 6665 -i 1

  playercam_yuv -h 10.100.0.x -p 6665 -i 1
- Only use *-i 1* argument if the “cameracompress” device is in the config file (-i is the identifier of which camera to use, in this case the compressed video stream).
- playercam_yuv is a wrapper for playercam that also outputs YUV values associated with a pixel.
- Clicking on a pixel in playercam will output to the terminal the associate XY location, RGC color values (and YUV color values if using playercam_yuv):

  [51, 145] = RGB [0 140 0] : : YUV [82 81 69]

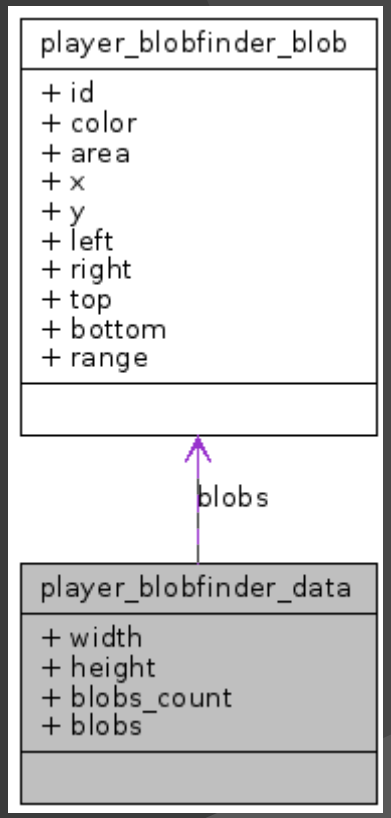
Color calibration

- You should calibrate for the color of specific objects by sampling their pixel colors in playercam.
- First, make sure that all the lights are on in 404 and the door is closed.
- Start playercam and put objects of interest in the robot's view. Click on a set of pixels for a single object.
- Obtain color values for the same object at different distances and locations to get a broad enough range, but watch out for shadows and specular artifacts.
- playercam will produce list of YUV values, find appropriate mix and max values for each range. Use best judgment to eliminate noisy outliers.
- Once you have a colorfile, restart player server and playercam to check effectiveness of the new thresholds.
- Playercam automatically overlays extracted blobs from blobfinder using the colorfile
- Start early! This is a time-consuming process.

Blobs

- The blobfinder returns a player_blobfinder_data_t which contains several player_blobfinder_blob_t.
- player_blobfinder_blob_t has several attributes such as color, size and position.

- In your client:
  BlobfinderProxy* blobProxy= new BlobfinderProxy(client, 0);
  playerc_blobfinder_blob_t blob = blobProxy->GetBlob(i);



http://playerstage.sourceforge.net/doc/Player-2.1.0/player/structplayer__blobfinder__blob.html