Tracking a Colored Object

Let’s assume that Grandma wears a Coast Guard Orange coat

A frame from a movie of the orange object

Simplifying the problem of finding grandma

- In a camera image we can look for lines, then objects, and finally try to identify a chair or even a grandmother in the image. (Mataric:74)
- Action-oriented perception (also called “active sensing”): instead of trying to reconstruct the world in order to decide what to do, the robot can use the knowledge about the task to look for particular stimuli in the environment and respond accordingly.
- As an example, it is very hard for robots to recognize grandmothers in general, but it is not nearly as hard to look for a particular color pattern of a grandmother’s favorite dress, perhaps combined with a particular size and shape, and speed of movement. (Mataric:75)

A patch of color

Representations of color

RGB

HSV
The HSV color solid

Hue and Saturation Histogram for the Orange Patch

Denote a volume in HSV color space by specifying its corners

- lower_orange = np.array([7, 120, 120])
- upper_orange = np.array([12, 255, 255])

How can we find the object?

Find the pixels in the hsv image that have colors that are in the target volume of the HSV color space.

Get a frame from the movie
Convert the frame to HSV color space
Make a new binary image (call it 'mask')
For every pixel in the frame, ask “Is the color of this pixel in the specified volume of color space?”
- If it is, set the corresponding pixel in the binary mask image to 1
- Else, set the corresponding pixel in the binary mask image to 0

• mask = cv2.inRange(hsv, low, high)
HSV video frame

mask = cv2.inRange(hsv, low, high)

Mask
(where pixels are of the specified color)

Actually, the raw mask image is noisy

Erosion
Convolution of a kernel over the image

Dilation
Convolution of the same kernel, but a different rule

Erosion followed by Dilation
Also known as Opening

kernel = np.ones((5,5),np.uint8) # a 5x5 matrix of ones

omask = cv2.morphologyEx(mask, cv2.MORPH_OPEN, kernel)
Clean up the mask

- Erosion
- Dilation
- Convolutions!

Can also look through the mask at the original frame

res = cv2.bitwise_and(frame, frame, mask=mask)

How can we find the object?

- Get a frame from the movie
- Convert the frame to HSV color space
  
  \[ \text{hsv} = \text{cv2.cvtColor}(\text{frame}, \text{cv2.COLOR_BGR2HSV}) \]
- Make a new binary image (call it ‘mask’)
- For every pixel in the frame, ask “Is this pixel in the specified volume of color space?”
  
  - If it is set the corresponding pixel in the binary image to 1
  
  - Else, set the corresponding pixel in the binary image to 0
- \[ \text{mask} = \text{cv2.inRange} (\text{hsv}, \text{low}, \text{high}) \]

Finding the center of the object

Show the contour of a blob

- The contour is the line of pixels in the blob that are the edge of the blob.

Now we want to find the center of the object

- Contours and Moments!
- \( x \) : the \( x \) coordinate of a pixel
- \( y \) : the \( y \) coordinate of a pixel
- \( I(x, y) \) : the intensity of a pixel either \( \{0, 1\} \)
- \( x^p \) : the \( x \) coordinate raised to the power \( p \)
- \[ m_{p,q} = \sum I(x, y)x^py^q \]
- \( \text{center} = (\int(m_{10}/m_{00}), \int(m_{01}/m_{00})) \)

\[ X \text{ coordinate} \quad Y \text{ coordinate} \]
Movie with object center indicated

Steering command

- Once we know the x coordinate of the center of the object, we just compare it to the x coordinate of the central meridian of the frame
- $steer = \frac{fw}{2.0} - center.x$

Big Ideas

- Images are arrays of numbers
- Color spaces; a color is a point in color space
- Volumes in color space: similar colors
- inRange(image, low, high)
- Binary mask images; seeing through a mask
- Convolution: pass a kernel over an image and apply a rule at each location
- Erosion and dilation, convolutions for image cleanup.
- Contours surround blobs
- Moments of contours characterize the blobs