code notes – Perception Module

Marc Toussaint

September 7, 2011

The perception module does a very simple thing:

1. We have a left and right image. 'EarlyVision' is computing the HSV for these images. We assume to know a specific target $hsv^* \in [0,255]^3$ values together standard deviations $\sigma_{hsv} \in [0,255]^3$. We compute the evidence $\tau_i = \exp(-(hsv_i - hsv^*)^2/\sigma_{hsv}^2)$ [sorry for sloppy notation] for each pixel i in the left and right image.

2. Given τ_i in an image, we call OpenCV's flood-fill that finds the contour of the highest τ-value region. Let's call the contour ∂C. (We do this for both images.)

3. Given the contour ∂C we compute a distance-to-contour field/image: for each pixel i we compute $d_i = \min_{j\in\partial C} |i-j|$ (using some OpenCV routine). We do this for both images. The image d_i is a good potential cost function to let 2D contour models converge to the HSV contour.

4. We have three different parametric 2D contour models: 1) for a circle (1 parameter), 2) for a polygon with 6 vertices and parallel opposing edges (a bit like a hexagon, can fit any 2D-projected 3D box), 3) a contour model that corresponds to a 2D projected cylinder.

   We fit a contour model to the HSV contour by minimizing the sum of d_i for all points on the contour model. We do this on both images. Fitting is done by gradient descent (RPROP). We get parameters of the 2D contours with sub-pixel accuracy.

5. Given the fitted 2D contours in the right and left image, we triangulate them. Giving us a 2D contour mapped into 3D space. From there it is trivial to fit a 3D ball, cylinder, or box.

Note: It is not by accident that we stay 2D for until the last step: in our experience it proved more robust to try to fit shapes/contours in 2D first with as much accuracy as possible before triangulating.