# Robonaut Panel Analysis - athyssen

## Basic Strategy

The basic strategy was the following:

* Determine the environment, camera settings, and panel positions of the items.
* Determine the 4 corners of the panel in image space
* Solve the 3D pose of the panel in camera space
* Solve the 3D positions of the items in camera space
* Convert the item positions back into image space
* Determine state of Buttons and LEDs by looking for illumination
* Infer state of toggle switches using Button and LED states.

## Intensity, Illumination, Dark Channels

Once the image is initially received, it is processed into three channels which are the primary information used by the algorithm. The intensity channel for each pixel is simply the average of R,G,B converted into a (0->1) range.

The illumination channel represents pixels that are either button illumination or LED illumination. This channel is computed in CalcIlluminationMask and is dependent on which environment we are in. The logic for each environment is hand tuned but involves simple thresholded comparisons of the max of RGB vs the average or min of the remaining components. The Illumination channel is a mask of values either 0 or 1.

The dark channel is used only in the LAB3 algorithm and is set to 1 if the average of a pixels RGB values are less than half the average illumination in the scene.

## Solve Camera Settings for each Environment

Camera settings were tediously hand solved for each environment. The values provided by NASA were used as a starting point. The values used by SIM varied a great deal. For instance, the sensor size was 10.5 microns rather than 4-5micron for the Labs. I have no way of knowing if these settings were accurate but they worked well in practice. The values can be found in the EnvironmentInfo table. The camera settings were vital to further parts of the algorithm. If this style of contest continues I highly suggest NASA provides accurate camera values.

## Solve Item Positions within the Panel

Item positions within the panel were extracted by hand using the Lab3 HTOG image. I used photoshop to cut out the panel, rotate it and then note the positions for the items into the ItemInfo tables. NASA published a clean reference photo late in the contest but I had already tweaked the positions by that time.

The dimensions of the panel were collected by finding the tskbd\_panel\_assembly\_slot1\_panel.dae file in the simulator data set, extracting all the vertex positions and finding the min and max bounds of the vertices. That exercise led me to conclude the panel width was 22cm and height was 32cm.

## Determining the Environment

The particular environment was needed to determine which camera settings to use, any specific thresholds, and differences in placement of items on the panel. For instance, the position of the power switch cover was lower in the simulated images than in real panels.

Four statistics were used to determine which environment we were in. WhiteSum and BlackSum are the percentage of intensities above 245 and below 5 in the scene. TopAvg was the average intensity across the top of the scene. AvgBright was the average brightness across the entire scene. And BlackBarRatio was the percentage of the scene with a fully black bar. This was used to pickup black panel mounting edges.

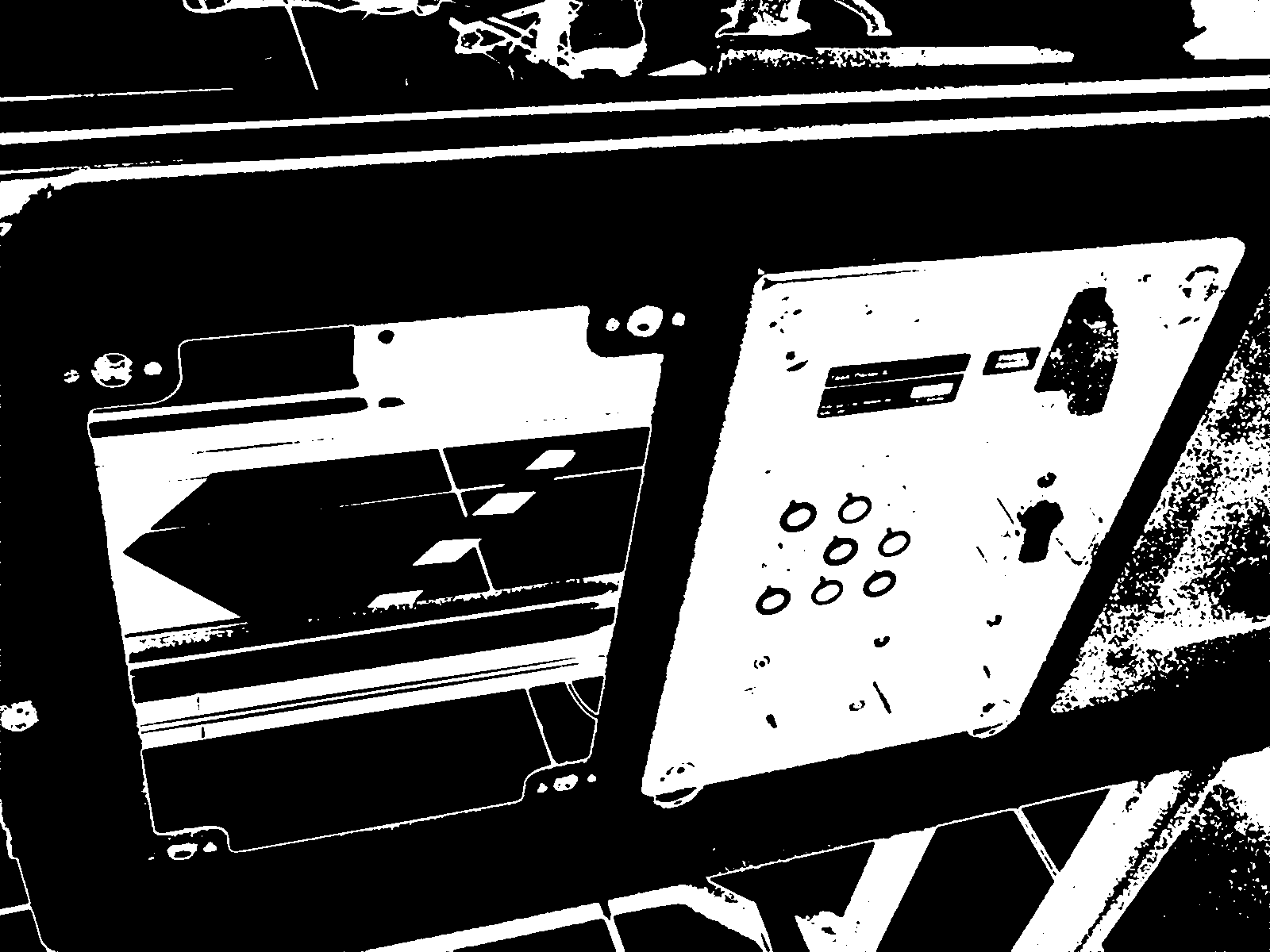
If the scene had a width > 1600 it was either ISS or LAB. If the AvgBright < 0.2 then it was an extremely dark ISS scene, otherwise if the WhiteRatio > 0.05 it was an ISS scene with the Robonaut hand in the view. Otherwise it was a general LAB scene.

If the scene width was 1600 and the black ratio was extremely small it was considered LAB3. If the TopAvg < 0.25 then it was the grey background of SIM. If TopAvg > 0.30 then it was the natural background of LAB2. Otherwise the WhiteRatio was slightly higher for SIM than LAB2.

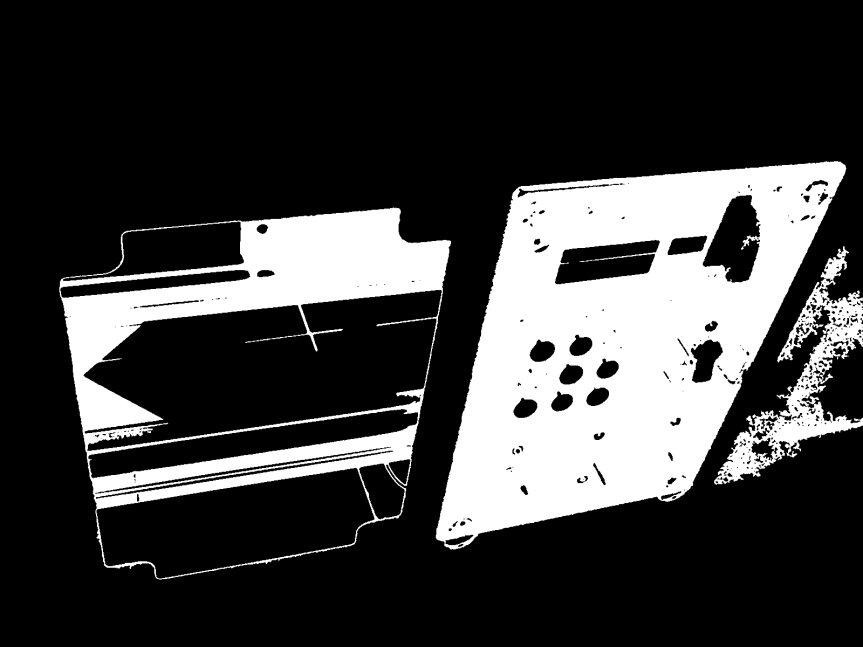
All thresholds were determined by hand. Once the environment was determined a lookup table was referenced that contained the camera settings and items positions.

## Find Panel 4 Corners in Image Space in LAB, LAB2, ISS, SIM

Otsu thresholding was used with an expectation of 2/3 black and 1/3 white.



A floodfill algorithm was then applied and only objects with Width >= (SceneWidth/8) and Height >= (SceneHeight/4) were kept.



For each remaining object edge detection FindRANSACQuad was called and 4 corner calculation was done. ‘rays’ were fired from the four sides of the images (T,L,B,R) and first pixel intersections with the object were considered edge points. For each side a line solving RANSAC algorithm was run to generate a robust edge estimation. Within the RANSAC algorithm, 500 trials were run. For each trial, two points out of the edge point set were compared. If their distance > 64 pixels then the two points were considered as candidates for the true edge. All other points on the edge were compared to this line and an inlierCount was incremented if a point was closer than 4 pixels to the candidate edge. The edge candidate with the most inliers was kept as the true edge. Once all four edges had RANSAC lines solved, the 2D intersection of the lines was used to calculate the 2D corners of the panel. Once the corners were available, calls to IsQuadShapedLikePanel were used. These were sanity checks to be sure the proportions of the candidate panels were reasonable. If the left and right side lengths ratio was more than 1.8 it failed. If the top and bottom ratio was more than 2.8 it failed. These values were found by hand tweaking until all true panels were included.



For each candidate panel that passed the 2D quad finding stage, a call to Solve3DQuad was made. This algorithm used the Solve4PointCoplanarPose which is described further in this document. Sove3DQuad takes the 2D corner positions, the known panel dimensions, and the camera settings, and returns a best estimate of the 3D camera positions of the four corners.

Once 3D panel corners are available, I then calculate all 2D item positions using CalcPerspectiveAwareBilinearPos. This takes an item position’s bilinear coordinates within the panel, calculates the 3D camera position by interpolating the four 3D corners of the panel, then uses the camera settings to project the 3D position into 2D. XY(0,0) maps to the panel’s top left corner and XY(1,1) maps to the panels bottom right corner. The item 2D positions are stored for later use.

The final determination for which of the candidate panels is the true panel is by looking for the red power switch cover. Using the 2D predicted position of the cover, a color match score is generated using CalcColorMatchScore and the panel with the highest score is chosen to be the true panel. A window of radius 64 is scanned in the image around the expected cover position. Each pixels red ratio of R / avg(G,B) is calculated and if the ratio is higher than a hand tuned PowerCapRedRatio then the pixel is counted as a vote for the cap being present. The final score is the number of votes / total pixels examined.

## Find Panel 4 Corners in Image Space in LAB3

Because LAB3 did not have the panel mounted, and due to large occlusion and background clutter a different algorithm was developed to find the four corners of the panel. Further work in this area would probably lead to a better strategy overall.

The strategy for LAB3 was to find the 3 dark rings of the toggle switch LEDs. From there find illuminated references, either a button or power LED. Choose a set of four points that maximizes the quads area. Knowing the panel positions of those items, solve the 4 point camera pose. In camera space, use barycentric coordinates to then calculate the panel’s 4 corners in camera space.

Finding reference points was done using the illumination and dark channels and running floodfill similar to the panel discovery of the other environments. This provided a set of possible reference points. All combinations of dark reference points were used to find the toggle switch LED positions. There are known to be collinear and equally spaced. Once these were located, they were used as rough guidelines to denote regions where the buttons and power led might be found.

The best set of four points was then set to Solve4PointCoplanarPose to solve the 3D positions.

The four panel corners were converted into barycentric coordinates and then used to interpolate the 3D positions of the camera space corners.

## Solve4PointCoplanarPose

Inputs were the camera parameters, ray directions from the camera position through the 2D positions of each of the 4 points and the panel positions of each of the 4 points.

It is known that the 4 points must lie along the ray projecting from the camera origin, through the image at the focal distance. A correct pose will have all 4 points being along the rays, and the inter-point distances matching the known inter-point distances found in the panel.

I step the first point in 1mm steps along its ray, starting at a distance of 100mm and ending at 1500mm. For each candidate position of p0, I compute the intersection between rays1-3 and spheres1-3 which are centered at p0 with radii matching the known panel distances between the points. This solves the constraint that for p1-3, a proper pose position will be along its ray but also a known distance from p0.

Each ray/sphere intersection can produce two candidate positions. That provides 2x2x2 = 8 possible combinations of pose positions for p1-3. All 8 combinations are considered and scored using Score4PointPose. The combination with the minimum error score is kept for this candidate pose p0.

The global best p0 has the lowest error produces by Score4PointPose.

Score4PointPose calculates the sum of squares error between the inter-point distances in camera space and the expected inter-point distances in panel space.

## Determine Button and LED On/Off Using Image

Once both eyes have been processed and 2D item positions have been calculated, Button and LED regions are examined to see if illumination is present. This is done as a simple region scan around the 2D position, accumulating the illumination channel and checking of the sum is above zero. If so, the item is marked as ON in that view.

## Determine all other States using Button and LEDs

If an LED or Button was determined ON in either view, it is considered ON.

All toggles are set based on their corresponding LEDS.

A variation of the original red power cover check is used to determine if the power cover is up or down. Two reference points in panel space are used that signify where the cover would be in an up and down position. Their 2D positions are calculated and the number of red votes is computed for each point. The Power Cover UP/DOWN state is determined by comparing which position has the majority of votes.