**kefset**

* Asks for the desired image
* Uses Aidan’s **getMatrixOutliers** code to attempt to remove noise
* Finds the canny edges of the image
  + removes lines with 10 pixels or less and displays the canny edges
* asks for input as to whether the breasts are small are large, and based on that finds the ellipses using the canny edges, using **ellipseDetection** and **ellipse** to draw them on.
  + converts the ellipses to pixels to display on the image and easier manipulate
* morphologically closes the image and bwmorph cleans them
* removes pixels from the top fourth of the image
  + if small breast, removes pixels from the left and right thirds of the image.
* the top 5% of pixels is found, using Nada’s hotpixelfinder code.

|  |  |
| --- | --- |
| If large breasts: | If small breasts: |
| Moves to script ‘**keflarge1**’ | Moves to script ‘**kefsmall1**’ |
| * Uses a point system.   + Uses canny edges to add 1 point, places with hot pixels adds 1 point, and ellipse pixels add 2 points. Places that had more than 2 points were included in the final figure.   + Using the point system, the top pixels and bottom pixels are found. A Total binary image was made using the inner fifth of the upper pixels, and the outside two/fifths on either side using the bottom pixels. * Using bwmorph and strels/imclose the image was cleaned. * Using bwconncomp, the connected components are connected to each other. * Code created by Zainab to connect the insides and the outsides as found with log edges. * Further connects points and finds the largest connected component to display (in an attempt to find the main boundary underneath the curvature). * Finds the top pixel on the right side of the image, and the top pixel on the left side of the image, and connects them to display the final line * Example: | * Uses hough transform imfindcircles to find circles and transform to pixels. Takes only bottom half by finding the lowest pixel in each column. * \*All above is done once it is determined there are only two circles found/reduced to be just two circles\*   Moves to script ‘**kefsmall2**’   * Uses a point system   + Canny edges add 2 points, lower half of ellipses add 3, hot pixels add 1, and the ellipses add 1. Places that had more than 1 point were included. * Makes the image thicker using bwdist command * Cleans lines using bwareaopen and bwmoprh operations * Again runs the **ellipseDetection** script on what has been collected so far, finds bottom two thirds of these pixels for use * Does a second point system   + Uses lower half of circles to add 3 points, places with hot pixels adds 1 point, and the new ellipses add 2 points. Places that had more than 1 point were included. * The central part of this is found, by creating a new binary image that just includes the inside third of the image. * Uses Nada’s code to connect the two sides to each other * Code created by Zainab to connect the insides and the outsides as found with log edges. * Further connects points and finds the largest connected component to display (in an attempt to find the main boundary underneath the curvature). * Finds the top pixel on the right side of the image, and the top pixel on the left side of the image, and connects them to display the final line   Example: |

More work has been continued using curvature and checking intensity levels to get rid of unneeded ‘slices in’ and straight lines:

* P10 and P12 were the images used for testing in particular

**tryingtodeterminecurv**

* The points from logfin are put into **points2contour** code from the file exchange, and the output x and y values are put into **LineCurvature2D** to determine the curvature along all of the points. Where the curvature is 0, the pixels are removed.
* The image is split in two, to better see if using the **points2contour** code will work when plotting it over the image. This has not proven to be very effective, as it’s difficult to find the correct starting point, and the lines sometimes still do not choose the nearest points. Finally, this system is not quite acceptable as the image doesn’t present the lines as pixels.

**intensitybased**

* Finds the top 10 rows of most intensity and deletes them. The idea behind this is to get any straight horizontal lines that may come from background/clothing deleted.
* The Hough transform codes are then used, hough, houghlines, and houghpeaks to find straight lines in the image and remove those. The thought process is that the breasts should not have straight lines.
* Bridging and thickening are done with morphological operations, and the log edges are combined on the outside with the found lines on the inside for what is at this point the final.