Sherlock 7 Technical Resource

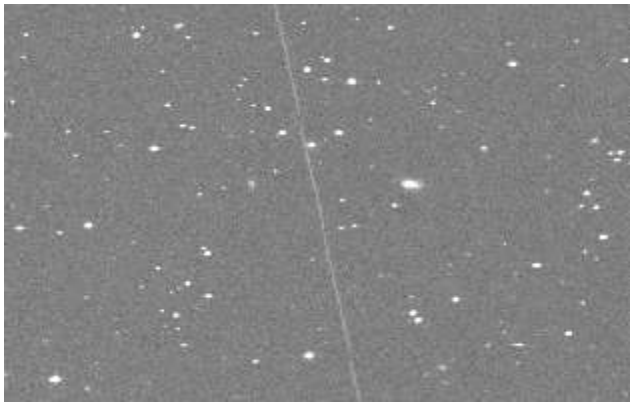
Teledyne DALSA Incorporated
Industrial Products

Ben Dawson
Document Revision: 10 August 2012

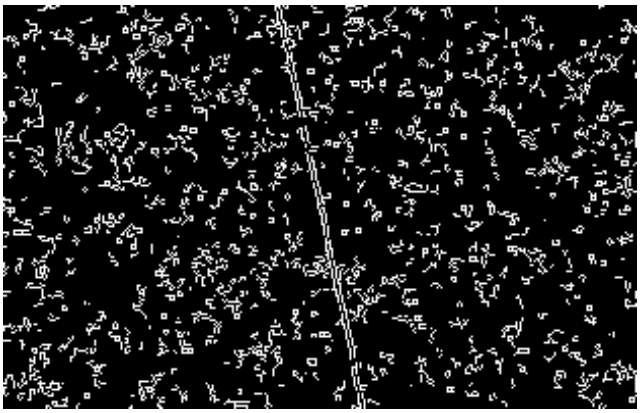
Hough Line and Segment

Introduction

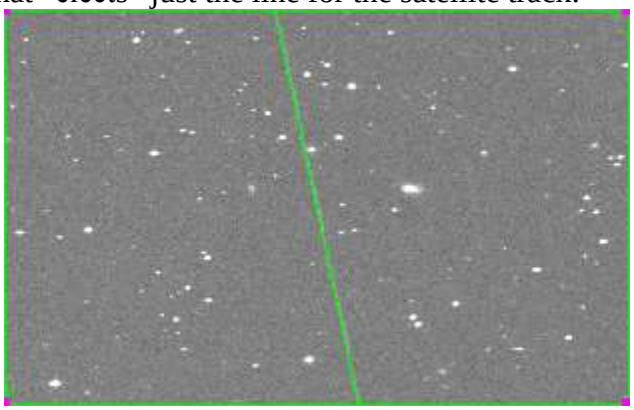
The Hough Transform[1] finds lines or other shapes in noisy images. For example, suppose we wanted to detect the linear track of a satellite against a background of stars:



A Canny edge "detector" generates lots of edge fragments but no way to link together only the fragments from the satellite track to detect it:



In the Hough Transform, all bright points in the image "vote" for all lines that could go through each point. The points on the satellite track all vote for the same line and so make a "majority" that "elects" just the line for the satellite track:



[1] P.V.C. Hough , "Methods and means for recognizing complex patterns", U.S. Patent No. 3,069,654

The Hough Transform (HT) detects shapes that can be expressed as a formula with parameters. For the line HT, the parameters could be the slope, *a*, and intercept, *b*, of lines in the formula:

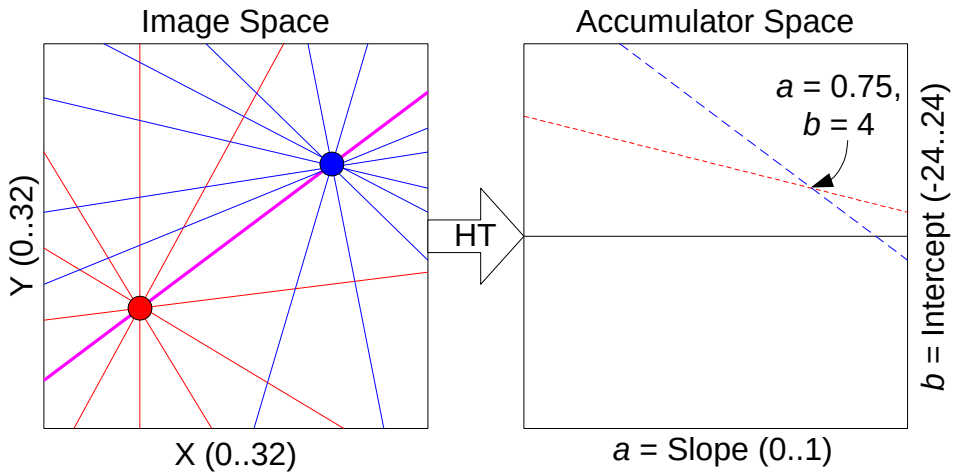
$$y = ax + b$$

The formula's parameters define a "parameter space" that is represented by a multi-dimensional digital array called the "accumulator space" or the "parameter space".

The digital accumulator space has discrete "bins". For the line HT, the dimensions of the parameter space could be {*a*,*b*} and we might divide this space into one-degree (slope) by one (intercept) bins to make the accumulator space. The bins in the accumulator space are initially set to zero.

Sherlock's Hough Transforms first transform the input image into a binary image, where values of 255 (full white) indicate points that will "vote". This binary image is scanned and each voting point increments the value in each bin of the accumulator space whose parameters could represent the line (or other formula) that could "go through" the voting point. After all points have voted, the accumulator space is scanned and bins with a count above some threshold are reported as "matching" the formula, for example a line at a particular slope and intercept. Thus voting points over the entire image can contribute to finding a line or other formula's parameters. This has similarities to the way human vision works – we use evidence over the entire image when inferring lines, etc.

This is the essence of the HT but is a bit difficult to understand. A simple example can help. Consider a Image Space (on the left) with only two voting points P1 = {8,10}, drawn in red, and P2 = {24,22}, drawn in blue. The line they are part of is drawn in magenta. You could use these two points to solve $y = ax + b$ and get $a = 0.75$, $b = 4$, but this won't work when there are many, noisy points.

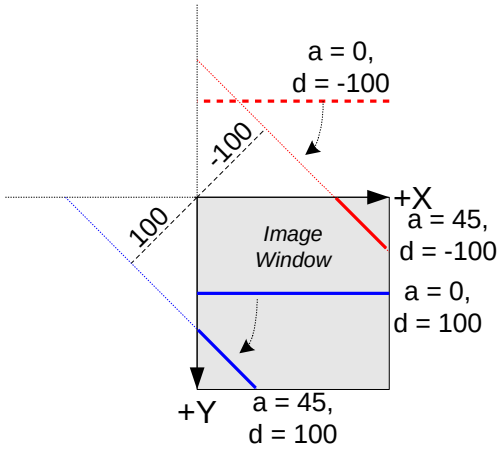


An infinite number of lines with different slopes and intercepts can go through each voting point in the image. I show a few of these lines for the red and blue points. We take each possible line through a voting point and increment the bin in accumulator space that corresponds to that slope and intercept. With the $y = ax + b$ formula, bins are incremented along lines in the accumulator space, as shown by the red and blue dotted

lines in the right drawing. The lines in accumulator space intersect at the bin representing the slope and intercept of the line the points are part of (magenta line on left), so this bin in the accumulator space gets the most votes (2, in this case). The other bins get 1 if they are a possible line through one of the points or 0 if not.

In most cases there will be many points voting and peaks in the accumulator space will be where points "agree" on the slope and intercept of the line they are part of. Other points will increment bins off of these peaks, and so are "noise" in the accumulator space. A threshold is used to separate the peaks from this background noise.

What happens when the line is vertical – the slope is 90 degrees = Pi/2 – and the intercept is therefore at infinity? Unless you can constrain the slope of possible lines in the input image, the $y = ax + b$ formula won't work. The solution is to use a formula that doesn't have this singularity at 90 degrees[2]. In Sherlock, we use parameters d = the perpendicular distance from the line to the origin, and a = the angle of intercept of the line with the X axis. Because the same angle could represent lines at equal distances from the origin, we set the sign of the distance to be that of the Y-axis intercept. To add to the confusion, we have to account for the Y image coordinate increasing downward. This drawing shows two pairs of lines with equal d and a values, distinguished by the sign of d:



With this representation, d is always in the range of –(diagonal of image length) to +(diagonal of image length), and a is always in the range of 0..Pi (0 to 180 degrees).

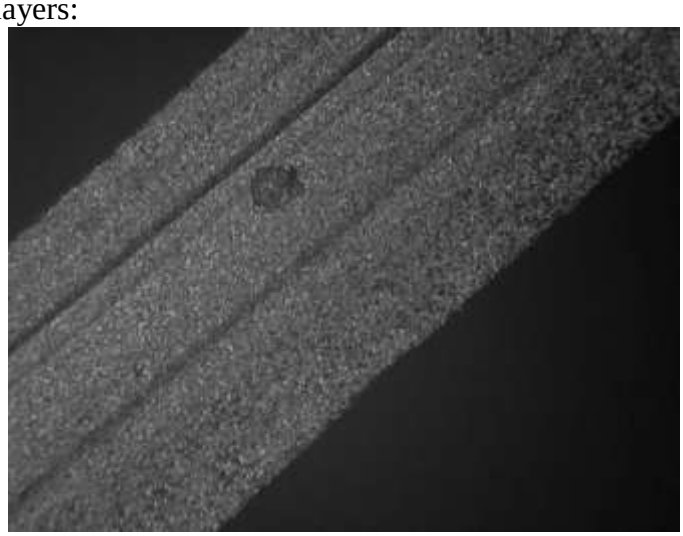
Another question: How can we accumulate votes for "An infinite number of lines with different slopes and intercepts" through each point? The discrete (digital) nature of the accumulator space (multi-dimensional array) means that we only sample the parameters (a and d, for example) for a finite number of bins. This also means that the resolution of line positions and angles is limited by the number of bins. Adding more bins gives better resolution but at the expense of additional computation time.

[2] R. Duda and P.Hart , " Use of the Hough Transform to detect lines and circles in pictures ", Communication of the ACM, vol. 15, pp. 11-15, 1972

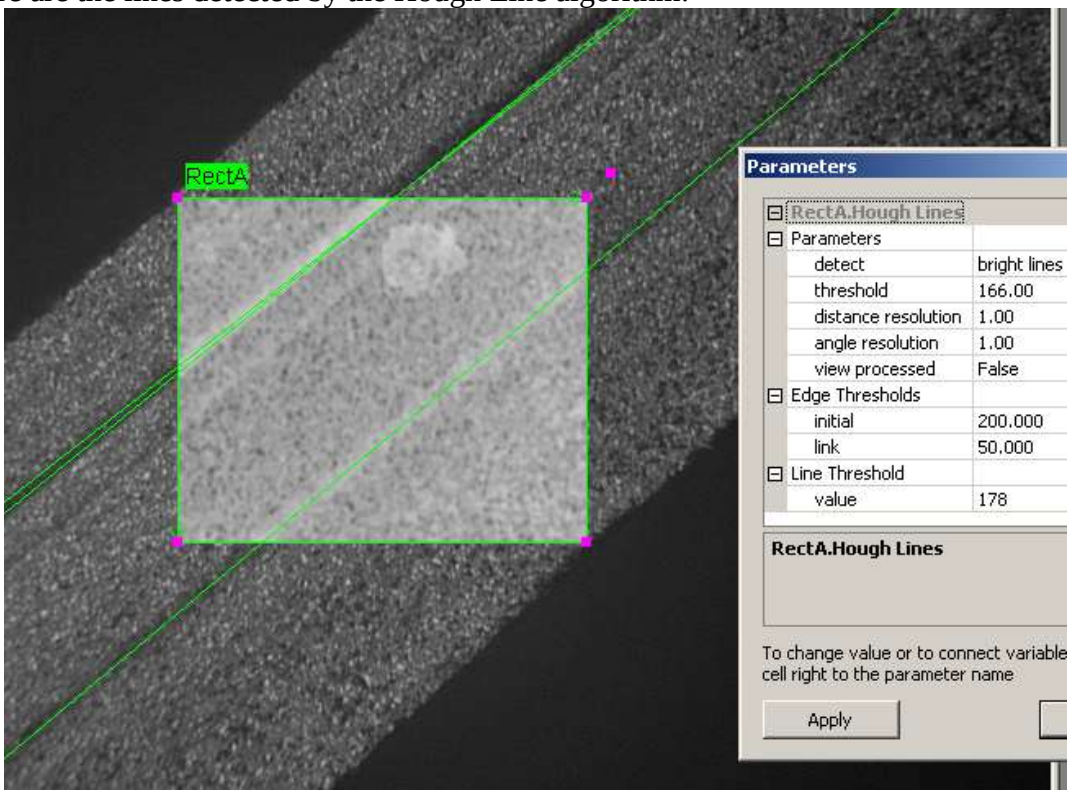
Hough Lines [Algorithm]

The Hough Lines (HL) algorithm detects lines in noisy images by accumulating "votes" for the offset (d) and angle (a) of all possible lines through each voting point in the image. This algorithm is very useful when you have broken lines (like cracks) and / or a noisy image. Note that curved lines (like a curved crack) will not work well, as the HL attempts to "fit" a straight line to the voting points.

In this image of a magnetic "read head" the goal is to find the two major dark bands that separate material layers:



Here are the lines detected by the Hough Line algorithm:



For this application, we positioned the ROI so that it would "see" the dark bands but not the black areas outside of the part. The image in the ROI was smoothed using the Gauss 5x5 preprocessor and then inverted the intensity to get bright lines.

You can see that the Hough Line algorithm returns multiple lines for one of the bands, as the bands are thick enough that they will accumulate votes for multiple bins. You can reduce this by using more of the line (perhaps with a rotated ROI to keep it within the part) and adjusting the HL's parameters.

The Hough Line parameters are in three groups: Parameters, Edge Thresholds, and Line Thresholds.

Parameters

The ***detect*** parameter specifies what kind of lines you want to detect. Your choices are:

- "bright lines" – Threshold the input image using the Line Threshold ***value*** parameter such that pixel values above the ***value*** threshold are set to 255 and below to 0.
- "dark lines" – The same as "bright lines" except that the image intensities are inverted before the ***value*** parameter is applied.
- "edges" – A Canny edge operator is applied to the input image to select only edge points in the image. The Parameters in Edge Thresholds, ***initial*** and ***link***, set the operation of the Canny edge operator.

threshold sets the threshold in accumulator space that separates peaks (line detected) from the "noise" introduced by pixels voting off of the line. Use this parameter to select the lines you want. You might not be able to set an acceptable threshold on broken lines in very noise input images – you have to get a better image or do some preprocessing (such as the Gaussian 5x5 done in the above example to reduce the texture "noise").

distance resolution sets the resolution of the *d* parameter in the accumulator space.
angle resolution sets the resolution of the *a* parameter in the accumulator space.
NOTE: *a* is currently in degrees, but will be in your choice degree or radian units in a later release of Sherlock. So, for example, an ***angle resolution*** of 1 means that the accumulator space is divided into 180 (0..Pi) bins on the a axis. Decreasing these values increases the resolution of the Hough Line algorithm but also increases computation time. If you are detecting lines or broad lines with little concern as to their {a,d} parameters, try increasing these values to get faster processing and some improvement in detection (signal to noise in the accumulator space).

If ***viewed processed*** is False (the default), then the algorithm does not change the output image. If it is True, then the output image will show the results of the ***detect*** parameter's pre-processing for "bright lines", "dark lines", and "edges". For example, "edges" will show the result of applying the Canny edge operator. This parameter is very useful when setting up Hough Line, as you can adjust the Edge Thresholds and Line Threshold and see what best detects the lines you want.

Edge Threshold

When the ***detect*** parameter is set to "edges", the ***initial*** and ***link*** parameters specify the operation of the Canny edge detector. The ***initial*** parameter is a threshold for initially detecting edges. The ***link*** parameter is a threshold for continuing (linking) edge points into contours. The ***initial*** parameter must always be greater than the ***link*** parameter – this is enforced inside the algorithm.

Line Threshold

When the ***detect*** parameter is set to either "bright lines" or "dark lines", the ***value*** parameter sets the threshold for converting the input image to a binary (0 and 255) image, where pixel values of 255 get to vote in the Hough Line transform and those with 0 do not vote. "dark lines" inverts image intensities, so the ***value*** parameter is still "255 when above".

The Hough Line outputs:

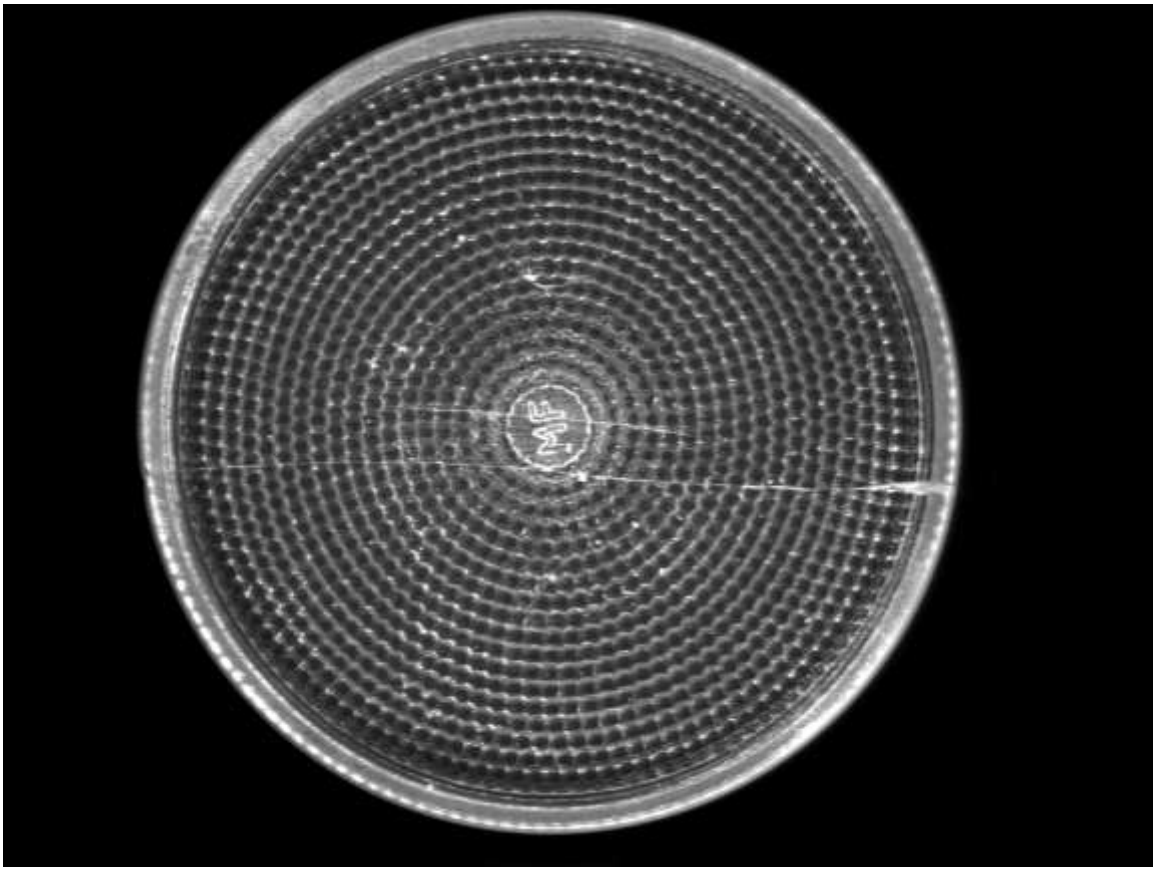
- "count" – The number of lines found. 0 if no lines were found.
- "lines" – An array of lines, represented in {a,d} format

Hough Segments [Algorithm]

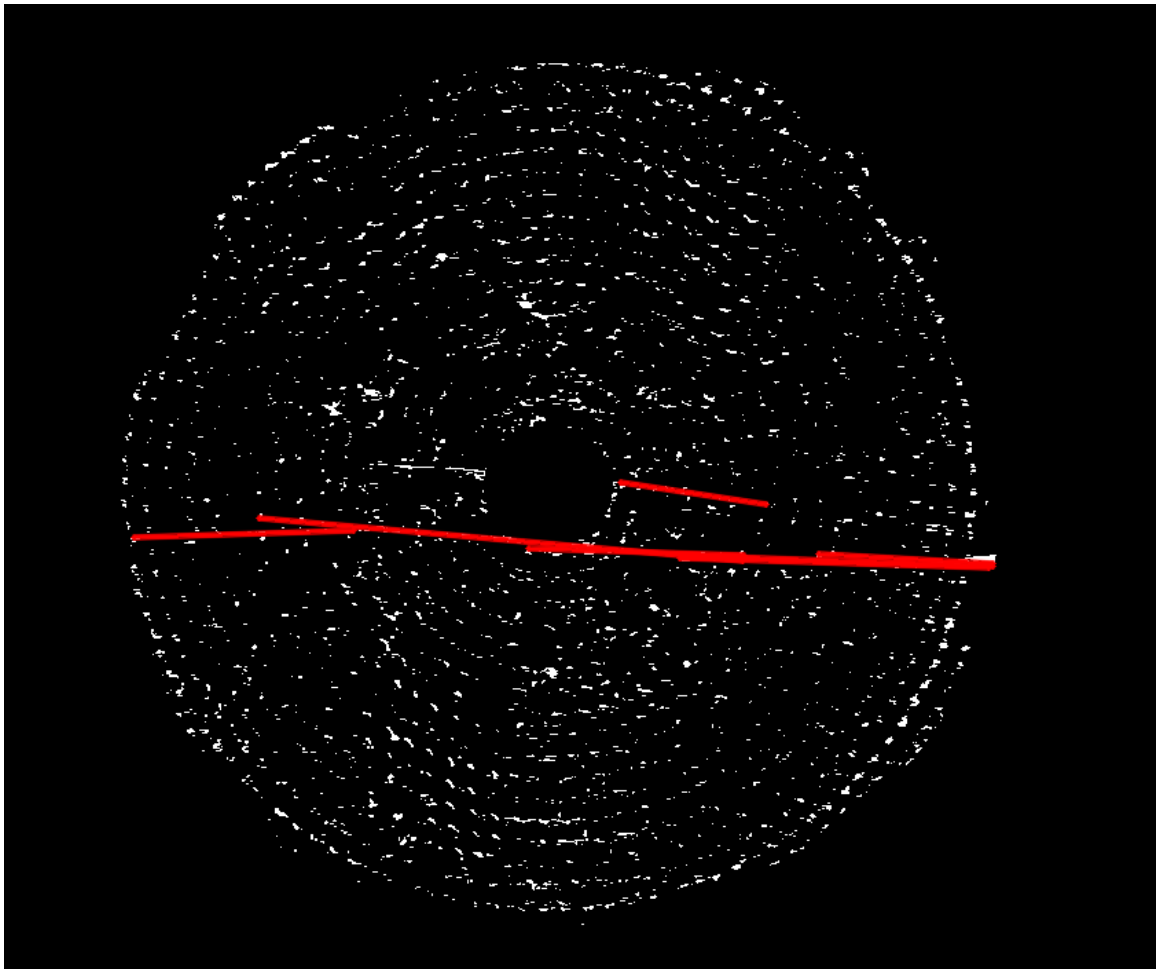
The Hough Segments (HS) algorithm detects lines in noisy images by accumulating "votes" for the offset (d) and angle (a) of all possible lines through each voting point in the image. This algorithm is very useful when you have broken lines (like cracks) and / or a noisy image.

The Hough Segments algorithm uses a sampling method that tests if randomly chosen pairs of points could be part of the same line. This has the advantage that you can get the end points of the line segments and that slightly curved lines can be better fit by a set of short line segments. Neither Hough Segments nor Hough Lines will do well on curved lines – that is not what they are designed for. The sampling method is also faster than Hough Lines but, in theory, can miss lines. Enough samples are taken that this is a low (but not 0) probability.

This image shows a fine crack across the lens of a spotlight:



We first process this image to reduce the texture caused by the lens pattern and then mask out the edges of the spotlight's lens to reduce the "noise" in the accumulator space. Applying Hough Segments to the processed image gives:



Here Hough Segments detects the two major cracks in the lens. This is not an easy task.

The parameters for Hough Segments are similar to Hough Lines, but with the addition of the ***min segment***, ***max gap***, and ***draw segments*** parameters. The Hough Segments parameters are in three groups: Parameters, Edge Thresholds, and Line Thresholds.

Parameters

The ***detect*** parameter specifies what kind of lines you want to detect. Your choices are:

- "bright lines" – Threshold the input image using the Line Threshold ***value*** parameter such that pixel values above the ***value*** threshold are set to 255 and below to 0.
- "dark lines" – The same as "bright lines" except that the image intensities are inverted before the ***value*** parameter is applied.
- "edges" – A Canny edge operator is applied to the input image to select only edge points in the image. The Parameters in Edge Thresholds, ***initial*** and ***link***, set the operation of the Canny edge operator.

threshold sets the threshold in accumulator space that separates peaks (line detected) from the "noise" introduced by pixels voting off of the line. Use this parameter to select the lines you want. You might not be able to set an acceptable threshold on broken lines in very noise input images – you have to get a better image or do some preprocessing (such as the Gaussian 5x5 done in the above example to reduce the texture "noise").

distance resolution sets the resolution of the *d* parameter in the accumulator space.
angle resolution sets the resolution of the *a* parameter in the accumulator space.
NOTE: *a* is currently in degrees, but will be in your choice degree or radian units in a later release of Sherlock. So, for example, an ***angle resolution*** of 1 means that the accumulator space is divided into 180 (0..Pi) bins on the a axis. Decreasing these values increases the resolution of the Hough Segments algorithm but also increases computation time. If you are detecting lines or broad line segments with little concern as to their {a,d} parameters, try increasing these values to get faster processing and some improvement in detection (signal to noise in the accumulator space).

min segment sets the minimum length for detecting a segment. Sampling will not draw pixels closer than this value and so you will have no output line segments shorter than this length.

max gap sets the maximum gap (in pixels) between segments before they are merged. This allows you to combine "dotted" or broken lines in to one, stronger line.

If ***viewed processed*** is False (the default), then the algorithm does not change the output image. If it is True, then the output image will show the results of the ***detect*** parameter's pre-processing for "bright lines", "dark lines", and "edges". For example, "edges" will show the result of applying the Canny edge operator. This parameter is very useful when setting up Hough Segments, as you can adjust the Edge Thresholds and Line Threshold and see what best detects the lines you want.

If ***draw segments*** is True (the default), the line segments are drawn as annotations on the output image. If this is False, the annotations are not drawn. Because these are line segments rather than Sherlock lines, there are no automatic line annotations for Hough Segments as there is for Hough Lines. The end points of the segments are automatically marked with the cross (+) annotation, which you can turn off by double-left-clicking the corresponding output and setting the display to [None].

<u>Edge Threshold</u>

When the ***detect*** parameter is set to "edges", the ***initial*** and ***link*** parameters specify the operation of the Canny edge detector. The ***initial*** parameter is a threshold for initially detecting edges. The ***link*** parameter is a threshold for continuing (linking) edge points into contours. The ***initial*** parameter must always be greater than the ***link*** parameter – this is enforced inside the algorithm.

Line Threshold

When the ***detect*** parameter is set to either "bright lines" or "dark lines", the ***value*** parameter sets the threshold for converting the input image to a binary (0 and 255) image, where pixel values of 255 get to vote in the Hough Segments transform and those with 0 do not vote. "dark lines" inverts image intensities, so the ***value*** parameter is still "255 when above".

The Hough Segments does not output Sherlock lines. Instead it outputs a count of segments and the start and end points of the segments:

- "count" – The number of line segments found. 0 if no line segments were found.
- "start" – An array of points, giving the {x,y} location of the start of each segment
- "end" – An array of points, giving the {x,y} location of the end of each segment.

The start and end points for a segment have the same index in these arrays.