/R=*roiWave* Specifies a region of interest (ROI) in the image. The ROI is defined by a wave of type unsigned byte (/b/u), which has the same number of rows and columns as the image wave. The ROI itself is defined by the entries/pixels whose value are 0. Pixels outside the ROI can take any nonzero value. The ROI does not have to be contiguous. See **ImageGenerateROIMask** for more information on creating ROI waves.

/RECT={*minRow, maxRow, minCol, maxCol*}

Limits the range of the ROI to a rectangular pixel range with /BEAM.

**Details**

The image statistics are returned via the following variables:

`V_adev` Average deviation of pixel values.

`V_avg` Average of pixel values.

`V_kurt` Kurtosis of pixel values.

`V_min` Minimum pixel value.

`V_minColLoc` Specifies the location of the column in which the minimum pixel value was found or the first eligible column if no single column was found.

`V_minRowLoc` Specifies the location of the row in which the minimum pixel value was found or the first eligible row if no single minimum was found.

`V_max` Maximum pixel value.

`V_maxColLoc` Specifies the location of the column in which the maximum pixel value was found or the first eligible column if no single column was found.

`V_maxRowLoc` Specifies the location of the row in which the maximum pixel value was found or the first eligible row if no single maximum was found.

`V_npnts` Number of points in the ROI excluding NaNs.

`V_rms` Root mean squared of pixel values.

`V_sdev` Standard deviation of pixel values.

`V_skew` Skewness of pixel values.

Most of these statistical results are similarly defined as for the WaveStats operation. WaveStats will be more convenient to use when calculating statistics for an entire wave.

If *imageWave* is 4D it is often useful to use the reversible conversion

```
Redimension/N=(rows,cols,layers*chunks) ImageWave
```

which allows you to obtain the statistics for each layer and all chunks of the wave. To convert back to 4D, execute:

```
Redimension/N=(rows,cols,layers,chunks) ImageWave
```

**See Also**

The **ImageGenerateROIMask** and **WaveStats** operations. **ImageStats Operation** on page III-371.

## ImageThreshold

**`ImageThreshold [`*`flags`*`]`** ***`imageMatrix`***

The ImageThreshold operation converts a grayscale *imageMatrix* into a binary image. The operation supports all data types. However, the source wave must be a 2D matrix. If *imageMatrix* contains NaNs, the pixels corresponding to NaN values are mapped into the value 64. The values for the On and Off pixels are 255 and 0 respectively. The resulting image is stored in the wave M_ImageThresh.

### Flags

/C — Calculates the correlation coefficient between the original image and the image generated by the threshold operation. The correlation value is printed to the history area (unless the /Q flag is specified), it is also stored in the variable V_correlation.

/I — Inverts values written to the image, i.e., sets to zero all pixels above threshold.

/M= *method* — Specifies the thresholding method. The calculated value will be printed to the history area (unless /Q is specified) and stored in the variable V_threshold.

- *method*=0: Default. In thie case you must use the /T flag to specify a manually-selected threshold.
- *method*=1: Automatically calculate a threshold value using an iterative method.
- *method*=2: Image histogram is a simple bimodal distribution.
- *method*=3: Adaptive thresholding. Evaluates threshold based on the last 8 pixels in each row, using alternating rows.

  The output wave M_ImageThresh has the same numeric type as the input wave. In particular, when the input is signed byte, the on and off pixel values are 127 and 0 respectively.

  Note that this method is not supported when used as part of the operation **ImageEdgeDetection**.
- *method*=4: Fuzzy thresholding using entropy as the measure for “fuzziness”.
- *method*=5: Fuzzy thresholding using a method that minimizes a “fuzziness” measure involving the mean gray level in the object and background.
- *method*=6: Determines an ideal threshold by histograming the data and representing the image as a set of clusters that is iteratively reduced until there are two clusters left. The threshold value is then set to the highest level of the lower cluster. This method is based on a paper by A.Z. Arifin and A. Asano (see reference below) but modified for handling images with relatively flat histograms.

  If the image histogram results in less than two clusters, it is impossible to determine a threshold using this method and the threshold value is set to NaN.

  Added in Igor Pro 7.00.
- *method*=7: Determines the ideal threshold value by maximizing the total variance between the “object” and the “background”. See http://en.wikipedia.org/wiki/Otsu%27s_method.

/N — Sets the background level to 64 (i.e., NaN).

/O — Overwrites the original image with the calculated threshold image.

If you do not specify the /O flag, the threshold image is written into the wave M_ImageThresh.

/P=*layer* — When *imageMatrix* is a 3D wave /P selects a specific layer for which to compute the threshold. *layer* is the zero-based layer index.

If *layer* is -1, which is the default value, the threshold is computed for all layers of *imageMatrix*.

The /P flag is not compatible with /O.

The /P flag was added in Igor Pro 7.00.

/Q — Suppresses printing calculated correlation coefficients (/C) and calculated thresholds (/M) to the history area.