ImageAnalyzeParticles

`ImageAnalyzeParticles [`*`flags`*`]` ***`keyword imageMatrix`***

The ImageAnalyzeParticles operation performs one of two particle analysis operations on a 2D or 3D source wave *imageMatrix*. The source image wave must be binary, i.e., an unsigned char format where the particles are designated by 0 and the background by 255 (the operation will produce erroneous results if your data uses the opposite designation). Note that all nonzero values in the source image will be considered part of the background. Grayscale images must be thresholded before invoking this operation (you may need to use the /I flag with the **ImageThreshold** operation).

Note: ImageAnalyzeParticles does not take into account wave scaling. All image metrics are in pixels and all pixels are assumed to be square.

Parameters

keyword is one of the following names:

mark — Creates a masking image for a single particle, which is specified by an internal (seed) pixel using the /L flag. The masking image is stored in the wave M_ParticleMarker, which is an unsigned char wave. All points in M_ParticleMarker are set to 64 (image operations on binary waves use the value 64 to designate the equivalent of NaN) except points in the particle which are set to the 0. This wave is designed to be used as an overlay on the original image (using the explicit=1 mode of ModifyImage). This keyword is superseded by the **ImageSeedFill** operation.

stats — Measures the particles in the image. See **ImageAnalyzeParticles Stats** on page V-365 for details.

Flags

/A=*minArea* — Specifies a minimum area as a threshold that must be exceeded for a particle to be counted (e.g., use *minArea*=0 to find single pixel particles). The minimum area is measured in pixels; its default value is *minArea*=5.

When the source wave is 3D, *minArea* specifies the minimum number of voxels that constitute a particle.

/A has no effect when used with the *mark* method.

/B — Erases a 1 pixel wide frame inset from the boundary. This insures that no particles will have boundary pixels (see /EBPC below) and all boundary waves will describe close contours.

/CIRC={*minCircularity*,*maxCircularity*}

Use this flag to filter the output so that only particles in the range of the specified circularity are counted.

/D=*dataWave* — Specify a wave from which the minimum, maximum, and total particle intensity are sampled when used with the stats keyword. dataWave must be of the same dimensions as the input binary image imageMatrix. It can be of any real numeric type. Results are returned in the waves W_IntMax, W_IntMin, and W_IntAvg.

/E — Calculates an ellipse that best fits each particle. The equivalent ellipse is calculated by first finding the moments of the particle (i.e., average x-value, average y-value, average x^2, average y^2, and average x*y), and then requiring that the area of the ellipse be equal to that of the particle. The resulting ellipses are saved in the wave M_Moments. When *imageMatrix* is a 2D wave, the results returned in M_Moments are the columns: the X-center of the ellipse, the Y-center of the ellipse, the major axis, the minor axis, and the angle (radians) that the major axis makes with the X-direction. When *imageMatrix* is a 3D wave, the results in M_Moments include the sum of the X, Y, and Z components as well as all second order permutations of their products. They are arranged in the order: sumX, sumY, sumZ, sumXX, sumYY, sumZZ, sumXY, sumXZ, and sumYZ.

/EBPC

Use this flag to exclude from counting any particle that has one or more pixels on any boundary of the image.

/F

Fills 2D particles having internal holes and adjusts their area measure for the removal of holes. Internal boundaries around the holes are also eliminated. When the boundary of the particle consists of thin elements that cannot be traversed as a single closed path which passes each boundary pixel only once, the particle will not be filled. Note that filling particles may increase execution time considerably and on some images it may require large amount of memory. It is likely that a more efficient approach would be to preprocess the binary image and remove holes using morphology operations. This flag is not supported when *imageMatrix* is a 3D wave.

/FILL

Use /FILL to fill holes inside particles. The reported values of area and perimeter are computed as if there are no holes. The filling algorithm could fail if, for example, there is a closed contour of zeros around the particles.

If you specify both /F and /FILL the operation used /FILL only.

Added in Igor Pro 7.00.

/L= (*row*,*col*)

Specifies a 2D particle location in connection with the mark method. (*row*, *col*) is a seed value corresponding to any pixel inside the particle. If the seed belongs to the particle boundary, the particle will not be filled. This flag is not supported when *imageMatrix* is a 3D wave.

/M=*markerVal*

Use this flag with the stats mode for 2D images. See **stats** keyword for a full description of the following waves:

markerVal=0:	No marker waves.
markerVal=1:	M_ParticlePerimeter.
markerVal=2:	M_ParticleArea.
markerVal=3:	M_Particle.

This flag does not apply to 3D waves.

/MAXA=*maxArea*

Specifies an upper limit of the area of an acceptable particle when used with the stats keyword. The area is measured in pixels and the default value of *maxArea* is the number of pixels in the image. In 3D the maximum value applies to the number of voxels.

/NSW

Creates the marker wave (see /M flag) but not the particle statistics waves when used with the stats keyword. This should reduce execution time in images containing many particles.

/P=*plane*

Specifies the plane when operating on a single layer of a 3D wave.

/PADB

Use this flag with the stats keyword to pad the image with a 1 pixel wide background. This has the effect that particles touching the image boundary are now interior particles with closed perimeter (that extend one pixel beyond the original image frame). In addition, entries in the wave W_ObjPerimeter will be longer for all boundary particles which will also affect other derived parameters such as circularity.

/PADB is different from /B in that it takes into account all pixels belonging to the particle that lie on the boundary of the image. The two flags are mutually exclusive.

/PADB was added in Igor Pro 7.00.

/PDLG

Displays a progress dialog.

/PDLG is useful when you are processing very large 3D images. The progress dialog provides feedback and allows the user to abort the operation.

/PDLG was added in Igor Pro 9.00.

/Q

Quiet flag, does not report the number of particles to the history area.

/R=*roiWave* Specifies a region of interest (ROI). The ROI is defined by a wave of type unsigned byte (/b/u) that has the same number of rows and columns as *imageMatrix*. The ROI itself is defined by the entries or pixels in the *roiWave* with value of 0. Pixels outside the ROI may have any nonzero value. The ROI does not have to be contiguous. When *imageMatrix* is a 3D wave, *roiWave* can be either a 2D wave (matching the number of rows and columns in *imageMatrix*) or it can be a 3D wave that must have the same number of rows, columns and layers as *imageMatrix*. When using a 2D *roiWave* with a 3D *imageMatrix* the ROI is understood to be defined by *roiWave* for each layer in the 3D wave.

See **ImageGenerateROIMask** for more information on creating 2D ROI waves.

/U Saves the wave M_ParticleMarker as an 8-bit unsigned instead of the default 16-bit when used with the mark keyword.

/W Creates boundary waves W_BoundaryX, W_BoundaryY, and W_BoundaryIndex for a 2D *imageMatrix* wave. W_BoundaryX and W_BoundaryY contain the pixels along the particle boundaries. The boundary of each particle ends with a NaN entry in both waves. Each entry in W_BoundaryIndex is the index to the start of a new particle in W_BoundaryX and W_BoundaryY, so that you can quickly locate the boundary of each particle.

When there are holes in particles, the entries in W_BoundaryX and W_BoundaryY start with the external boundary followed by all the internal boundaries for that particle. There are no index entries for internal boundaries.

This flag is not supported when *imageMatrix* is a 3D wave.

Details

Particle analysis is accomplished by first converting the data from its original format into a binary representation where the particle is designated by zero and the background by any nonzero value. The algorithm searches for the first pixel or voxel that belongs to a particle and then grows the particle from that seed while keeping count of the area, perimeter and count of pixels or voxels in the particle. If you use additional flags, the algorithm must compute additional quantities for each pixel or voxel belonging to the particle.

If your goal is to mask only the particle, a more efficient approach is to use the **ImageSeedFill** operation, which similarly follows the particle but does not spend processing time on computing unrelated particle properties. ImageSeedFill also has the additional advantage of not requiring that the input wave be binary, which will save time on performing the initial threshold and, in fact, may produce much better results with the adaptive/fuzzy features that are not available in ImageAnalyzeParticles.

ImageAnalyzeParticles Stats

The ImageAnalyzeParticles stats keyword measures the particles in the image. Results of the measurements are reported for all particles whose area exceeds the *minArea* specified by the /A flag. The results of the measurements are:

V_NumParticles Number of particles that exceed the *minArea* limit.

W_ImageObjArea Area (in pixels) for each particle.

W_ImageObjPerimeter Perimeter (in pixels) of each particle. The perimeter calculation involves estimates for 45-degree pixel edges resulting in noninteger values.

W_circularity Ratio of the square of the perimeter to (4*π*objectArea). This value approaches 1 for a perfect circle.

W_rectangularity Ratio of the area of the particle to the area of the inscribing (nonrotated) rectangle. This ratio is π/4 for a perfectly circular object and unity for a nonrotated rectangle.

W_SpotX and W_SpotY Contain a single x, y point from each object. There is one entry per particle and the entries follow the same order as all other waves created by this operation. Each (x,y) point from these waves can used to define the position of a tag or annotation for a particle. Points can also be used as seed pixels for the associated *mark* method or for the ImageSeedFill operation.