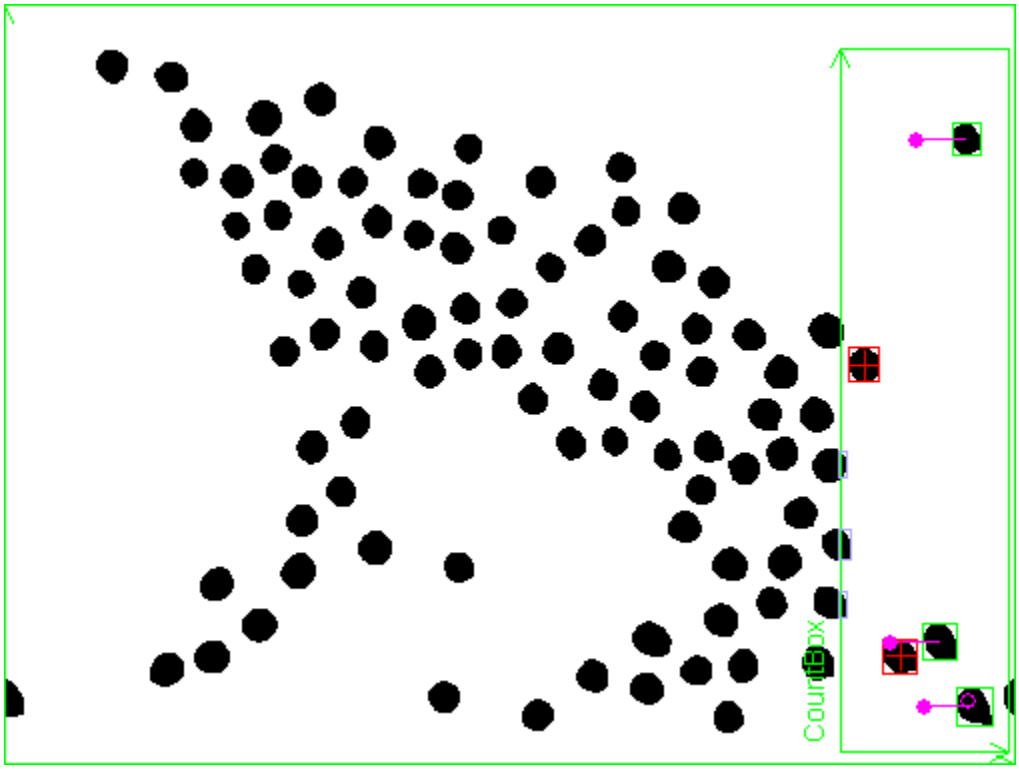
Sherlock 7 Technical Resource

Teledyne DALSA Incorporated
Industrial Products

Ben Dawson
Document Revision: 10 August 2012

Count Moving Parts Algorithm



Counting parts on a conveyer – view from above with the parts moving to the right.

Count Moving Parts [algorithm]

Count Moving Parts (CMP) counts parts (objects) moving straight through the view of an area scan camera. A common application of CMP is counting parts as they move by the camera on a conveyer belt.

Size and Speed Parameters and Measures are in Pixels

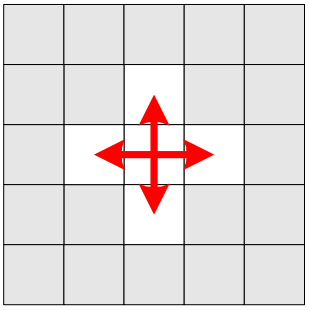
Size measures – width, height, area – are in pixels, and speed and skew are in pixels per frame. The exception is the ***current centers*** output, which is an array of points – x,y values – that Sherlock can translate into calibrated x,y locations, if you set it to do so.

CMP Algorithm Overview

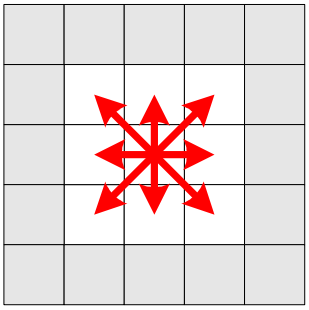
CMP takes a "binary" image ROI (Region of Interest), where the parts are either black (intensity = 0) or white (intensity = 255) and the background is the opposite intensity. If you don't provide CMP with a binary image, then CMP uses pixel values of 0 as "black" and all other pixel values as "white". This might not detect the parts you want and makes setup difficult, so threshold the ROI before using CMP.

CMP uses connectivity analysis, also known as "blob analysis", to find areas in the ROI where pixels of the same intensity (black or white) are connected or "touch" each other. These connected areas represent the "parts" that CMP counts and tracks. When the ***black parts*** parameter is TRUE, CMP finds black parts (pixel intensity = 0) and when ***black parts*** is FALSE, CMP finds white parts (pixel intensity = 255).

When the ***8-way*** parameter is TRUE (the default), then pixels are part of a connected area if they touch in any of 8 directions. When ***8-way*** is FALSE, only four directions of touching are considered when gathering pixels into a connected area:



4-Way Connectivity



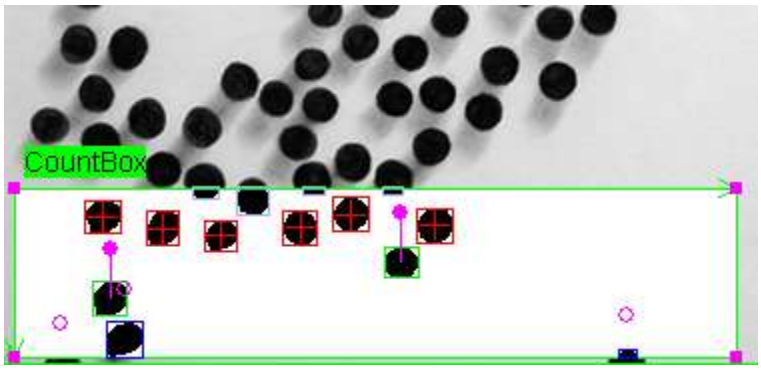
8-Way Connectivity

When parts are on a background, the usual case, ***8-Way*** is normally set to TRUE. If you want to detect the background as a "part" you should set ***8-Way*** to FALSE.

CMP considers a "part" to be a connected area of pixels that are within size and area limits. These limits are specified by the ***min width***, ***max width***, ***min height***, ***max height***, ***min area*** and ***max area*** parameters.

CMP uses a temporal sequence of images called *frames*, like the frames in a movie. Set the camera that is viewing the parts to generate frames at a regular interval or trigger the camera based on an encoder or other motion sensor. The rules for how much movement is allowed between frames are discussed below.

CMP detects parts (connected areas of black or white pixels, as selected by ***black parts***), counts them when they are fully within the ROI, and then tracks counted parts so they are not counted again.



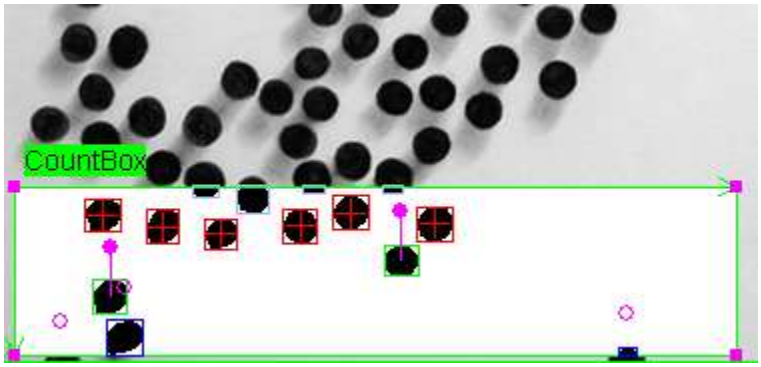
In the above image, the ROI used for counting is labeled "CountBox". I applied a threshold to make a "binary" image black (intensity = 0) parts white (intensity = 255) background. As mentioned, this makes it easier to select the parts we want to count, rather than using CMP's internal and inflexible binary conversion.

The edge of the ROI where parts enter is called the *Enter* edge and the opposite edge, where they leave, is called the *Leave* edge. In this example, the parts flow from top to bottom, so the top edge of the ROI is the Enter edge and the bottom is the Leave edge. CMP ignores parts that touch the Enter or Leave edge. If a part comes in or goes out of the side edges (left or right in this case), the count of parts could be incorrect. So make sure that parts only flow through the Enter and Leave edges and that they travel in straight lines. Some restrictions on part shapes and motion are discussed below.

The CMP algorithm ignores parts that touch the *Enter* or *Leave* edges of the ROI. With ***show bound box*** annotations enabled, parts touching the Enter edge have a light blue box around them and parts touching the Leave edge have a dark blue box around them. The other graphics are described below.

For parts that are not touching the Enter or Leave edges, the question is: Which parts are new (are now fully in the ROI) and so must be counted and which parts have just moved within the ROI and so must not be counted again?

Let's call two successive frames the *previous* and *current* frames. CMP tries to make a match or correspondence between parts counted in the previous frame, and parts that are available for counting in the current frame, that is, parts that have moved past the Enter edge to be fully within the ROI. If no correspondence is found, the part is new in the current frame and so is counted. If a correspondence is found between a part in the previous and current frame, the part is tracked but not counted.



In the above image, new parts in the current ROI are annotated with a red box and a red cross at their *centroid*, or center of gravity, locations. Previously seen parts are annotated with a green box and are not counted again. Unfilled magenta circles show previous frame locations for parts that have moved to touch the Leave edge or have entirely moved out of the ROI (see the left-most, magenta circle) in this frame. Filled magenta circles show where a part was in the previous frame and a magenta line connects that previous location to its location in the current frame. This line might not be seen if the parts are moving slowly, as it will be entirely inside the filled, magenta circle.

To establish correspondence, we need the following:

- The direction of motion, specified by the ***direction of motion*** parameter.
- Part motion, except for some possible skew and noise, is either horizontal or vertical in the camera's field of view. Parts enter and leave only by Enter and Leave edges.
- All parts move together and at the same speed, and don't touch each other.
- Parts don't slide or roll around as they are moved.
- Parts don't change shape as they move. In practice, parts can get a little larger or smaller without causing problems. This is important because optical distortion from the camera's lens can slightly change the apparent size of parts as they move through the camera's field of view.
- Unless we know the ***speed*** (a parameter) of the parts, the part motion must be less than the size of the smallest part, measured in the direction of motion.

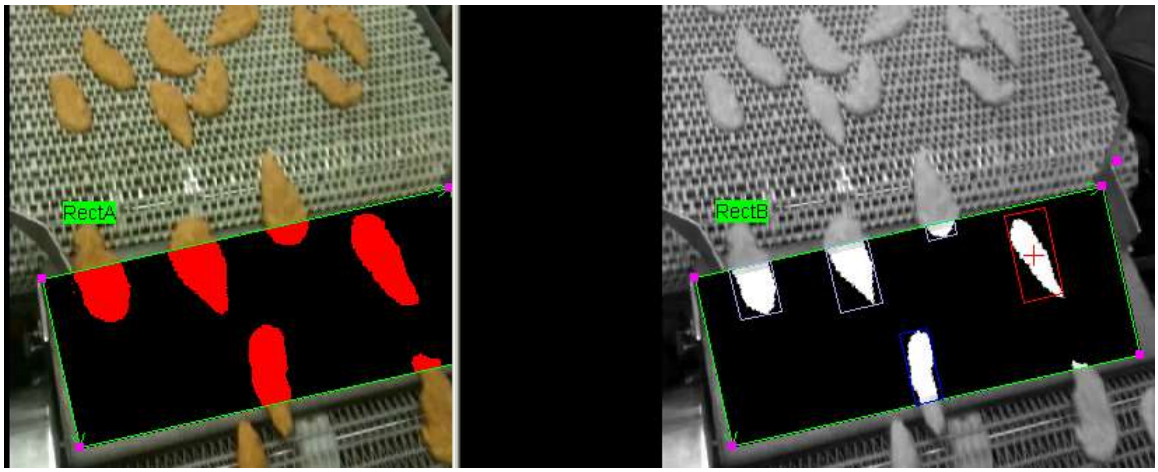
Then correspondences between parts in previous and current frames are established by finding the centroid of the closest part in the current frame that is "in front" of and on approximately the same horizontal or vertical path as a centroid of a part in the previous frame.

What Parts Work with CMP?

Parts Must be Presented to CMP as a "Binary" Image

CMP requires "parts" that are either black (intensity = 0) or white (intensity = 255) with the background of opposite intensity. You must arrange part lighting so that a threshold and other pre-processing operation separates part and background (e.g. the conveyer belt) and with low "chatter", that is pixels turning on and off around the edges of the parts. The background should be free of contamination, gross texture, product residue, etc. so that these are not seen as a part.

In the following example, "chicken fingers" cannot be distinguished from the background based on pixel intensity. We use a color camera and thresholds to separate the brown-red chicken fingers from the gray "slider" area between conveyers (see left image):



The red channel provides a binary image input into CMP (right image). This image is eroded to separate touching parts. The ROI is taken where the parts slide on a smooth, low-texture surface between the two conveyer belts (see image below), so that belt texture doesn't appear as parts or image noise.

When the ROI is rotated and the "Interpolate" box is checked in the ROI's properties, as in this example, the interpolation introduces image values other than the "binary" values

of 0 and 255. You might be able to make out some gray pixels at the Enter and Leave edges in the annotated CMP image, above right. We used erosion to separate touching "chicken fingers", and erosion extends the non-binary image values. Because CMP considers gray-valued pixels to be white (255), this can cause the apparent shape of parts to change at the edges of the image when detecting black moving parts (***black parts*** = TRUE). In this example the parts are made white (***black parts*** = FALSE), and as the interpolated values are also made white by CMP, there is no problem here. From this we conclude:

- Avoid using rotated ROIs for CMP, if you can
- If you must use a rotated ROI here are ways to avoid this interpolation problem:
    - Make sure you threshold in the ROI's processing chain (we didn't in this example), or,
    - Try to set up the processing so the parts are white, or,
    - Turn off "Interpolate".

Parts Must not Touch Each Other

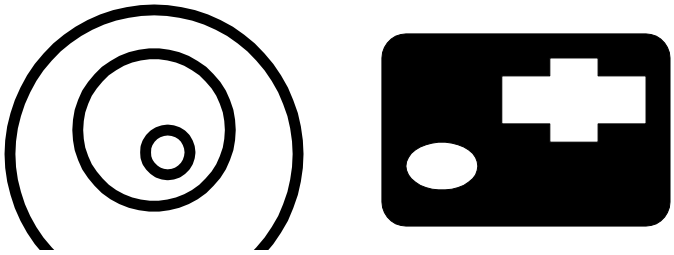
Parts are detected as separate areas of connected pixels. When parts do touch they can sometimes be "separated" by using erosion or dilation preprocessors. If the parts are black, then dilation "peels" off black pixels on the edges of the parts, thus separating them. You might follow dilations with erosions which adds black pixels to the edges of parts and so somewhat restores their size. For white parts, you would first erode to separate touching parts and then possibly dilate to somewhat restore their size.

In the "chicken fingers" example, above, sometimes processing can't separate parts because they touch over a large area. See, for example, the two "fingers" in the upper, right of the last image, above. In that case, CMP gives an incorrect count of parts. Here are some possible ways around this:

- Restructure the part presentation to separate the parts. For example, parts might be separated into channels that fan out and then be dispensed with spaces between them. This is good for parts that are quite uniform, such as pills or container closures, but not for "chicken fingers"!
- If the parts have a limited range of sizes, you might be able to detect two touching parts because the resulting "part" will be approximately double the size of a typical single part. CMP provides frame-by-frame information on part widths, heights, and areas, so this information can be use to correct the count for "doubled" parts. Again, "chicken fingers" have a wide variation in size so this method is not appropriate.
- Accept an inaccurate count. In the case of "chicken fingers", the goal was an approximate number per hour, mainly to check for product loss due to theft.

Parts Must not be Nested but May Have Holes

Parts must not be nested as there is no way to tell that there are multiple parts based on their centroids. It is generally OK to have holes in the parts, as long as there are enough pixels to reliably compute the centroid of the part. The following drawing shows nested black parts (NO!) and a part with holes in it (OK!).



NO! OK!

Part Dimensions

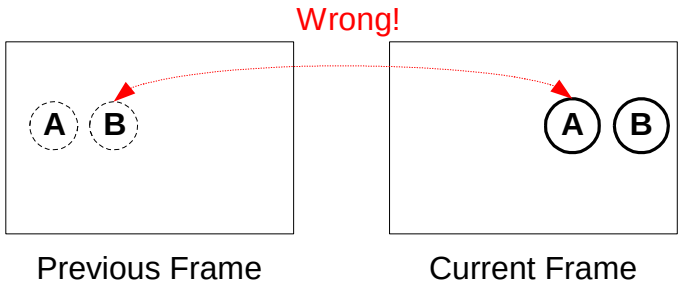
Part size should be at least 10 x 10 pixels (100 pixels area) for reliable counting. As a part gets smaller, noise from the camera, lighting, optics and vibration corrupts part centroid locations and causes correspondence and counting errors.

Each part must be entirely within the ROI for at least one frame, so the maximum part size (in the direction of motion) must be smaller than the ROI dimension in the direction of motion (called the *ROI Length*) minus 4. That allows one frame where the part is entering the ROI, one when it is entirely within the ROI, one where it is leaving the ROI and one to allow for some jitter in the motion. We do not recommend going to this limit. First, a slight increase in the part size will stop it from being counted, and second, the part motion must be very slow – one pixel per frame in this case. As discussed below, we recommend a maximum part size of 1/3 of the ROI Length.

How Fast can the Parts Move?

The maximum allowed speed of movement of the parts, in pixels per frame along the direction of part motion, is a function of the part size and the ROI size. We start with general principles and then get into details.

If parts move too fast, then part correspondences between frames can't be established and the count will be wrong. In this diagram, parts labeled **A** and **B** appear in a previous frame at the left (dotted circles) and in the current frame at the right (thick circles).

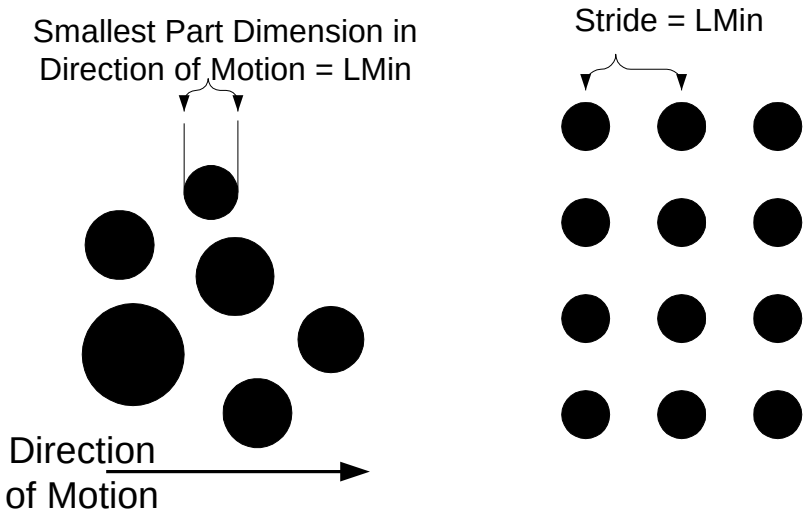


CMP knows parts are moving horizontally to the right but not how far they have moved. **A** in the current frame is closest to **B** in the previous frame, and is "in front" of and on the same horizontal line. Therefore **A** in the current frame will be incorrectly corresponded with **B** in the previous frame.

To prevent this mistake, the part movement (in pixels per frame) must be limited to the minimum dimension (in the direction of motion) of the smallest part, so that the correct centroid for matching a current part position is always the closest centroid in the previous part position (frame). In the figure below, this is marked as LMin. If, for example, LMin is 55 pixels, then parts must not move more than 55 pixels between frames.

If you know the ***speed*** (a parameter) of the parts, you can tell CMP where **A** and **B** are expected to be in the current frame, so the parts can move faster without causing this mistake.

If the parts are in columns, then the fastest the parts can move (if you don't use ***speed***) is the *Stride*, the smallest distance between columns. See the right side of this figure.



Detailed Calculation of Maximum Part Speed with ROI Size

Assume we are not using the ***speed*** input to tell the CMP algorithm how fast (in pixels per frame) the parts are moving.

Define:

LMin	= The minimum part size in the direction of motion if parts are randomly placed or Stride if parts are in columns
LMax	= The maximum part size in the direction of motion if parts are randomly placed or Stride if parts are in columns.
ROI Length	= The size of the frame's ROI in the direction of motion
Min(…)	= Returns the minimum of its arguments (…)

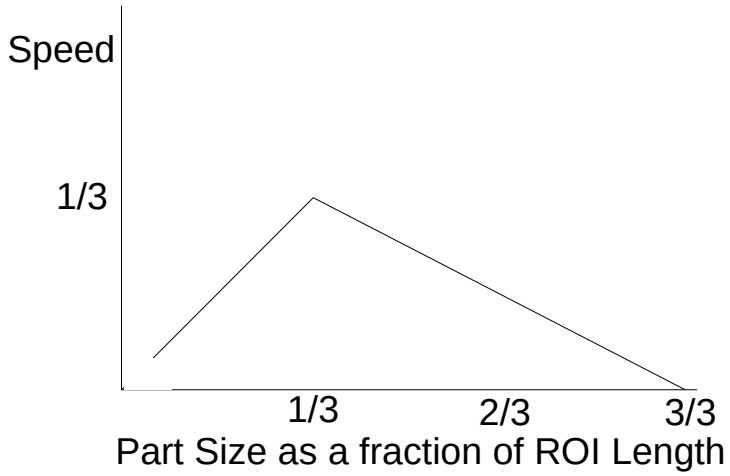
Then the maximum speed in pixels per frame is:

Maximum speed = *Min*(((ROI Length – 4 – LMax)/2) , LMin)

This equation says that when the parts are small, the most we can move is LMin pixels per frame, that is, the smallest part dimension or the distance between columns of parts if

the parts are in columns. As the part size gets larger or the spacing between columns gets larger, we can increase the speed (pixels per frame) but at some point we have to reduce the speed to insure that there is at least one frame when a part is entirely within the ROI.

This plots the maximum speed, in fractions of the ROI Length, as a function of the part size, also as a fraction of the ROI Length:



So the maximum speed is when the part size in the direction of motion is approximately 1/3 the ROI size. I've also shown the part size as not starting at 0, suggesting that you don't want a part size so small that it is influenced by noise, and shown that the part size can't go all the way to the ROI Length (3/3). Note that the for part sizes below 1/3 of the ROI length the speed is linear with a slope of 1 and above that, linear with a slope of -1/2.

Mechanical and optical considerations generally set the part size. You can then set the ROI to be 3 times the part size in the direction of motion for the maximum speed, assuming that the ROI still fits within the Image Window! That speed can be computed from the above equation.

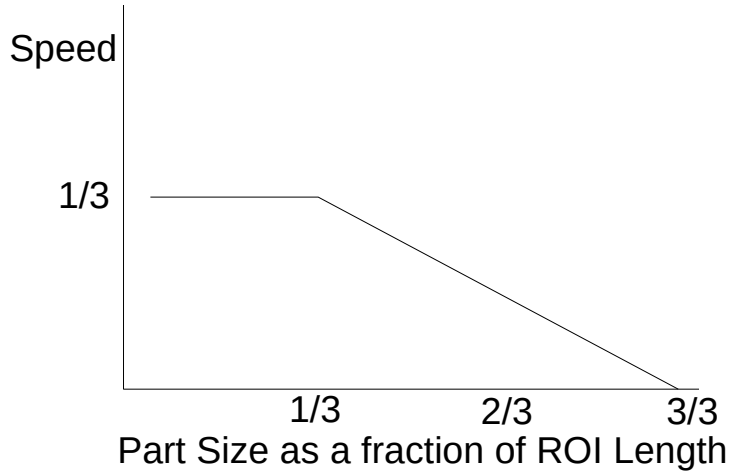
To estimate the speed in "real world" values, compute the speed in pixels per frame, and multiply by pixel size on the part (WPS) and by frames per second (FPS):

World Speed = Speed in pixels per frame * WPS * FPS

If you have a ***speed*** parameter, then smaller size parts can move faster and still be tracked. The equations become:

Maximum speed = Min(((ROI Length – 4 – LMax)/2) , (ROI Length)/3)

That is, below a part size that is 1/3 of the ROI Length, the maximum speed is 1/3 of the ROI Length, but above that size, the speed decreases to insure that at least one frame is entirely within the ROI. The curve now looks like:



The World Speed is then computed as above.

So the advantage to having a ***speed*** parameter is that the maximum speed is attained over a wide range of part sizes. The CMP algorithm can track parts over a larger area per frame and so can deal with faster-moving parts. If you don't have a ***speed*** input, you are limited to movements that are no larger than the smallest effective size (LMin), regardless of the ROI Length.

Here is a "worked" example. Suppose that:

No ***speed*** input.

LMin	= 127 pixels	Minimum part size in motion direction
LMax	= 150 pixels	Maximum part size in motion direction
ROI Length	= 480 pixels	Length of ROI in motion direction
FPS	= 30	Frames per second
WPS	= 0.01"	World pixel size = 1/100 of an inch

Because both the minimum and maximum part length are less than 1/3 of the ROI Length, we only need to consider the slope = 1 relationship between part size and speed. The maximum speed is thus 127 pixels per frame.

Multiplying this by a WPS of 0.01" we get 1.27 inches per frame, and multiplying this by 30 frames per second we get 38.1 inches per second as the maximum World Speed of the parts. In practice, use a maximum below this, say 36 inches per second, to allow for uncertainty in the part measurements and calculation.

A quick way to get LMin and LMax is to run "Connectivity – Binary" on an image of the parts and find the minimum and maximum dimensions from this algorithm's Outputs. Already in pixels!

Other Motion Limitations

Besides the above limitations on part speed, the CMP algorithm has the following limitations on the motion of the parts.

- Parts must only move in the direction of motion, specified by the ***direction of motion*** parameter, but perhaps with some skew. That means parts can't roll around, slide, pile up on each other, etc.
- Parts must move slowly enough so that each part is entirely within one or more frame's ROI Length.
- Parts must enter and leave the ROI only by Enter and Leave edges. It is OK if parts touch the other two side edges of the ROI and perhaps go a little outside of the ROI. However, as parts go out of the ROI on the side edges, the chance of a miscount increases. Use rails on either side of the conveyer to limit part lateral motion and then set your ROI to go nearly to those rails.
- Parts must move together and at the same speed.
- Parts can't change shape as they move but a little change, usually due to optical distortion, is OK.
- Unless we know the ***speed*** (a parameter) of the parts, the part motion must be less than the size (measured in the direction of motion) of the smallest part.

Set Up

Camera Direction: Typically the camera is set so that the "long" image dimension is perpendicular to (across) the conveyer belt or part motion, but the CMP algorithm can use the "long" or "short" camera axis either direction of motion. For example, with a 640 by 480 camera, you might set the camera width (640) to be perpendicular to the direction of motion – across the conveyer belt. Then the ***direction of motion*** is either "up" or "down" in the camera's images.

Optics and Object Size

The camera should view the entire width of the belt and be straight above and centered over the moving objects. The plane of the camera's sensor should be parallel to the surface of the belt. Some optical and perspective distortions are tolerated by the CMP algorithm. However, these distortions will affect the accuracy of the part width, height, area, etc. measures. Both types of distortions can often be reduced by using a long focal length lens and longer working distance, but at the expense of less light into the camera.

The minimum object size should be more than 10 x 10 pixels in size. The largest object size should be somewhat less than 1/3 the Length of the ROI – Length is the dimension of the ROI in the direction of motion.

Skew

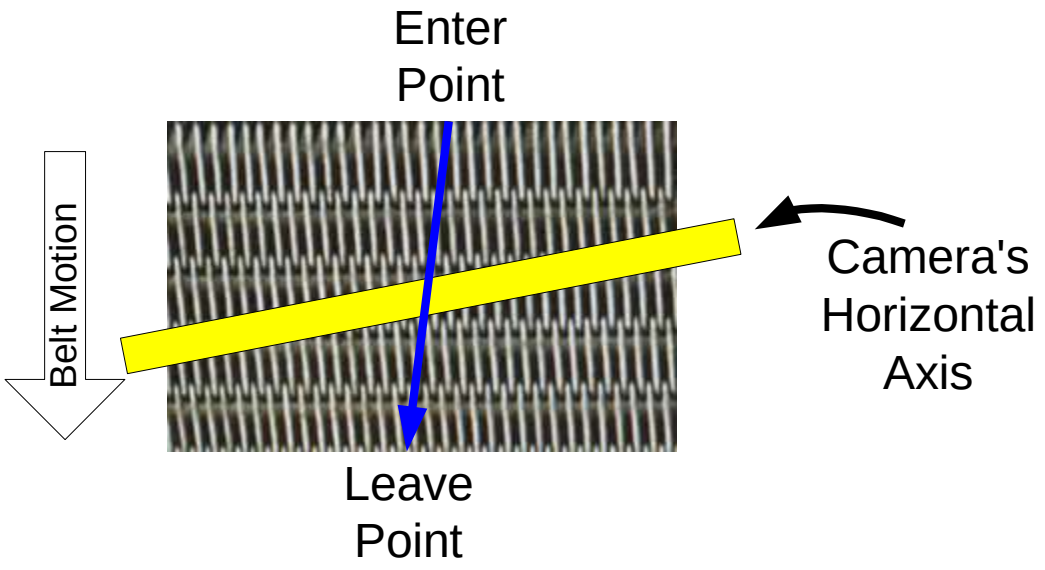
Skew refers to apparent lateral motion of the parts within the ROI – parts appear to "drift" sideways as they move. Skew occurs when the camera's vertical (or horizontal) axis is not exactly aligned with the direction of motion of the conveyer belt (or other motion machinery). CMP allows some frame-to-frame skew using the ***move box size*** parameter. The ***move box size*** parameter is meant to specify the noise tolerance in tracking the centroid position from frame to frame, but skew is a kind of "noise" in the centroid position. Increasing ***move box size*** to greater than about ½ the "height" of the

smallest part to be counted, can cause a miscount. Here, the "height" is the dimension of the part perpendicular to the direction of motion.

To test for skew, move a small spot on the conveyer from the Enter edge to the Leave edge of the ROI and see if there are more than ***move box size*** pixels of sideways movement. If there is, the best solution is to rotate the camera around its optical axis – the axis through the center of the lens – to reduce the skew to less than ***move box size*** pixels.

If you are unable to eliminate skew by rotating the camera (or motion system) or to tolerate it by enlarging ***move box size***, use the ***skew*** parameter to compensate for skew.

To set ***skew***, place a spot on, say the conveyer belt, right at the Enter edge of the Image Window – not the ROI. Measure the position, in pixels, of this spot. Now move the conveyer belt so that the spot moves to the Leave edge of the Image Window (again, not the ROI) and measure the position there. These measurements are easy to do: Take a picture of the spot using Sherlock and use the mouse and the x,y position values in the lower, right status bar to read out the spot's position. The absolute value of the sideways movement spot is the ***skew*** value. In this example, the spot skews horizontally to the left so ***skew direction*** would be *skew left*.



Skew is measured on the full Image Window so that you have the longest distance ("lever arm") to measure the lateral shift. It also allows you to change the ROI size without "recalibrating" the skew.

The sign of the skew depends on the orientation of the camera, the direction of motion and the direction of the skew. If the "long" axis (the horizontal axis) of the camera is across (perpendicular to) the direction of motion then:

There are factors other than a rotated camera that can cause apparent skew. These include changes in lighting over the frame's ROI, optical distortion, and parts sliding due

to the conveyer belt being tilted. These distortions cannot be reliably "removed" by the ***skew*** parameter. If they are influencing the count, you have to fix them.

Starting and Stopping

Starting the CMP algorithm can be tricky. If you open the algorithm's parameters dialog, when you change ***operation*** from *idle* or *reset* to *count* this causes a single execution of the algorithm. If there are any countable parts in the image, they will be counted. If these parts were left from a previous "run" and so have been counted, they now have been counted twice!

Here are two good ways to start CMP:

1. Use a variable to select the ***operation*** so you can *reset* at start-up and then set it to *count* when new parts are available.
2. Make sure the field of view is empty when you start up CMP.

When you stop the motion, say by stopping the conveyer belt, the parts remaining in the camera's field of view may appear to drift backwards and forwards. This causes them to be counted multiple times. This apparent drift is caused by "chatter" noise, that is noise in the optical and imaging system that causes the binary outline of the part to change and hence the position of the centroid that is used for tracking the part.

The optional parameter***, max back drift***, allows some apparent backward drift of the parts due to this "chatter" noise. ***max back drift*** should be kept small, perhaps 1 or 2, to prevent loss of correspondence and hence miscounts.

When you stop the motion, the ideal would be to stop CMP just before the part motion stops. When you restart, you ideally want the part motion to start just before CMP starts. If the starting and stopping can be done with no more than LMin (the minimum size of the part in the horizontal direction) motion between the last frame before stopping and the first frame after starting, then you don't have to worry about ***max back drift***.

Output Types

Two types of outputs are provided. Those with "current" in their name are measures of parts in the current frame's ROI, while outputs without "current" in their name are summary statistics of the entire "run" of parts. When the ***operation*** parameter is set to *reset*, the summary statistics are cleared.

Note that the bounding annotations drawn if the ***show bound box*** parameter is TRUE, might not align with or completely enclose parts that have holes in them

Set Up Parameters

These parameters should only be changed while the CMP algorithm is in *idle* or *reset* mode.

operation [Enum]

idle (0) Pause the counting operation (*default)*
reset (1) Reset the counting operation. All summary statistics are zeroed.
count (2) Count moving parts.

direction of motion [Enum]
moving left (0) The part motion is from right to left in the image
moving right (1) The part motion is from left to right in the image (*default)*
moving up (2) The part motion is from bottom to top in the image
moving down (3) The part motion is from top to bottom in the image

Note: Changing the ***direction of motion*** forces a *reset* ***operation***, so all summary statistics (such as the average height of the parts) are set to zero. This is done because a direction change invalidates the counting of the parts currently in the frame's ROI.

black parts [Bool]
TRUE Parts are black on a white background (*default)*
FALSE Parts are white on a black background

8 way [Bool]
TRUE 8-way connectivity is used to define the binary part images (*default)*
FALSE 4-way connectivity is used to define the binary part images

Note: 8-way connectivity means that each pixel that is in a part must be "touching" another pixel in that part in one or more of 8 directions – up, down, left, right, and the 4 diagonal directions from the pixel. 4-way connectivity means that each pixel that is in a part must be touching another pixel in that part in one or more of 4 directions – up, down, left, and right. 8-way connectivity is typically used on "foreground" parts, 4-way on background parts. Touching parts can sometimes be separated by using 4-way connectivity.

Move box size [Double]
The allowed variation in part position from frame to frame for a correspondence match. This is the size of a square area surrounding the predicted location of a part seen in the previous frame. If the centroid of a part seen in the current frame falls into this square area, that part is in correspondence with a part seen in the previous frame. Corresponding parts are not counted, as they were counted in a previous frame.
Default: 4
Minimum: 2
Maximum: 1000

<u>Part Dimension Limits</u>

These specify the minimum and maximum dimensions and area of parts. Use them to filter out apparent "parts" that are due to noise, dirt, contamination. We recommend that you not try to count parts that are smaller than 10 x 10 pixels (area of 100).

min width [Integer]
The minimum width (in the image's X direction) needed for a part to be counted.

Default: 10
Minimum: 1
Maximum: 1000000

max width [Integer]
The maximum width (in the image's X direction) allowed for a part to be counted.
Default: 200
Minimum: 1
Maximum: 1000000

min height [Integer]
The minimum height (in the image's Y direction) needed for a part to be counted.
Default: 10
Minimum: 1
Maximum: 1000000

max height [Integer]
The maximum width (in the image's Y direction) allowed for a part to be counted.
Default: 200
Minimum: 1
Maximum: 1000000

min area [Integer]
The minimum area needed for a part to be counted.
Default: 100
Minimum: 1
Maximum: 1000000000

max area [Integer]
The minimum area allowed for a part to be counted.
Default: 1000000
Minimum: 1
Maximum: 2000000000

Optional Parameters

These parameters make the tracking more accurate and (for ***speed***) possibly faster.

Skew [Double]
The number of pixels a part moves sideways (perpendicular to the direction of part motion) over the entire image – the Image Window, not the ROI. This is most easily measured by putting a spot on the motion system and measuring its position as it enters and leaves the Image Window. Skew is the absolute difference in sideways position.

Skew occurs when the camera is not exactly perpendicular to the motion of the parts. It is best if you rotate the camera to remove the skew but, if this is not possible, the ***skew*** parameter allows you to correct for some rotation. Apparent skew can also arise from

variations in lighting, optical distortion, noise and background. The ***skew*** parameter cannot correct for these.

Default: 0 0 means disables skew
Minimum: 0
Maximum: 1000

speed [Double]
The speed of movement of parts, in pixels per frame. Set to -1 to disable using this parameter. The speed of movement is usually measured by an encoder on the motion system (say, a conveyer belt) and has to be translated into pixels per frame, that is, how far a part moves between frames (image acquisitions), measured in pixels. Using the ***speed*** parameter better accuracy and, if the parts are less than 1/3 the ROI size, the speed of counting is increased.

Default: -1 -1 means Disable ***speed*** parameter
Minimum: -1
Maximum: 10000

max back drift [Double]
When there is no motion the parts might appear to move due to "chatter" noise, that is changes in the apparent outline of the parts due to optical and image system noise. This noise changes the apparent position of the part centroids that are used for tracking. If the part drifts backward and then forward, it will be counted again. To prevent this, set ***max back drift*** to a small value, much less than the minimum size of the part in the direction of motion. Large values of ***max back drift*** can prevent parts from being correctly tracked.

Default: 0
Minimum: 0
Maximum: 100

Annotation Parameters

These parameters enable or disable the display of annotations for the parts. They are useful for setup and debugging, but turn them off if you need to increase CMP's speed.

NOTE: Red crosses (+) are drawn on parts that are counted in the current frame. The cross is positioned at the part's centroid (center of gravity). These annotations are drawn by default for the ***current centers*** output. You can turn them off by double-clicking that output and setting "Display in image window" to [None].

show bound box [Bool]
TRUE A colored bounding box is shown around each part. *(default)*
FALSE No bounding box is shown
The colors for the bounding box are:
Red - This part is being counted in the current frame
Green - This part is not counted as it was counted in a previous frame
Light Blue - This part is entering the ROI and is ignored in this frame

Dark Blue - This part is leaving the ROI and is ignored

show vectors [Bool]

TRUE Show a motion vector from the previous to the current part position (*default)*

FALSE Do not show motion vectors

The motion vectors are shown as a magenta filled circle and a line (a "lollypop" graphic). The filled circle is the position of the part in the previous frame and the line connects this position to the position of the part in the current frame. If the part is moving slowly, the line part of the vector graphic (the lollypop "stick") might be entirely within the filled magenta circle and so will not be seen. If the part has moved partially or totally out of the frame, only an unfilled, magenta circle is drawn.

Outputs

Outputs with "current" in their names apply to the current frame. All the others are cumulative over all frames seen while ***operation*** = *count*. A *reset* clears all cumulative values and is done by:

- Setting ***operation*** = *reset*
- Changing ***direction of motion***
- Putting the algorithm into an ROI

total count [Double]
The total number of parts counts since the last *reset*.

current count [Double]
The number of parts counted in the current frame.

current widths [Double array]
Width of counted parts in the current frame.

current heights [Double array]
Height of counted parts in the current frame.

current areas [Double array]
Areas of counted parts in the current frame.

current centers [Point array]
Centroid ("center of gravity") of counted parts in the current frame.

min width [Double]
The smallest part width seen in parts counted since the last *reset*.

max width [Double]
The largest part width seen in parts counted since the last *reset.*

avg width [Double]
The average of counted part widths seen since the last *reset.*

std width [Double]

The standard deviation of counted part widths seen since the last *reset.*

min height [Double]
The smallest part height seen in parts counted since the last *reset.*

max height [Double]
The largest part height seen in parts counted since the last *reset*.

avg width [Double]
The average of counted part heights seen since the last *reset.*

std width [Double]
The standard deviation of counted part heights seen since the last *reset.*

min area [Double]
The smallest part area seen in parts counted since the last reset ***operation***.

max area [Double]
The largest part area seen in parts counted since the last reset ***operation***.

avg area [Double]
The average of counted part areas seen since the last reset ***operation***.

std area [Double]
The standard deviation of counted part areas seen since the last reset ***operation***.

estimated speed [Double]
The speed of the parts, in pixels per frame, estimated from the part tracking algorithm. If the speed of the parts changes slowly, it might be possible to feed this output into the ***speed*** parameter and so improve tracking accuracy and speed.