Tips

*Use the **Array Operator** to handle **indexed measurements***

Many functions, like Detect Object, produce a series of results with indexed measurements. It is sometimes in interest to perform additional actions on those results, such as computing the mean or average area of those hundreds of particles found in the image. Loop all of them right through by inserting a **For**-loop, selecting the cycle index according to Detect Object - # Objects, for instance, then inserting an Array Operator inside the loop, choosing the interesting indexed measurement, like Detect Object - Object [i]. Area, and finally for the Index assign the "For - Cycle Index". Now, when the for-loop runs, the Array Operator step picks a value from the results according to the loop cycle index and this value can be fed into additional processing.

Note that **Min** and **Max** values can be obtained without a loop by changing the **Sort by** options of the function, like Detect Object, and then inserting an Array Operator with index 1 for instance.

*Utilise **Coordinate Systems** to ensure the **ROI**'s are always in the right places*

If items under inspection are not always guaranteed to be in the exact same place in the image, it is a good idea to use some easy-to-find "landmark" as a reference. Use locating algorithms like **Match Pattern**, **Geometric Matching** or the **Edge finders**, or even **Gauge** under the Measure Features, to first locate that landmark and its orientation, create a **Coordinate System** based on that result and link other **ROI**'s to that as a **Reference System**.

*Don't forget the multitude of ways to process point information with the **Geometry** function*

The **Geometry** step under **Measure Features** is a versatile tool for processing the points already located/defined in the image. It can be used to measure distances between dynamic points, measure angles between lines, bisection, form perpendicular or parallel projections etc.

*Make use of the **Image Buffer** to detect changes*

In certain types of inspection applications it is of use to detect changes. This could, for instance, mean that there is a reference image, a "good" one, and the image under inspection would be compared to that. In Modular-X, such scenarios can be easily implemented using the **Image Buffer** function to store certain images into memory and later performing a comparison using for example the **Image Operator** function.

*If in possession of **LabVIEW**, don't be afraid to try out adding **more functionality** into the applications*

Want to perform more complex math operations, for instance, and feel that Modular-X math function is not enough? Create a **VI** with **LabVIEW** with the appropriate inputs and outputs to take care of the task.