

Cognex Designer Advanced – Section 5

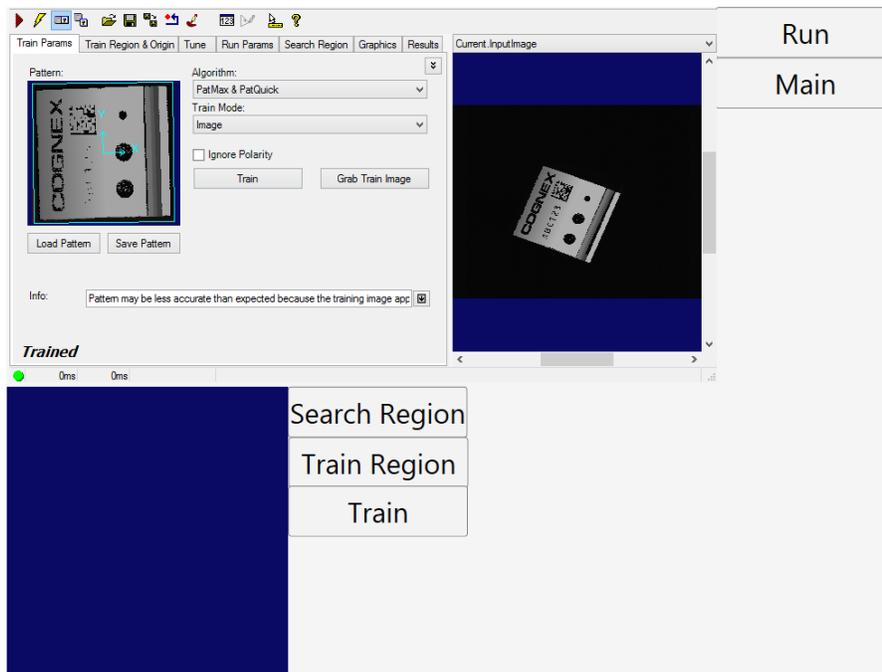
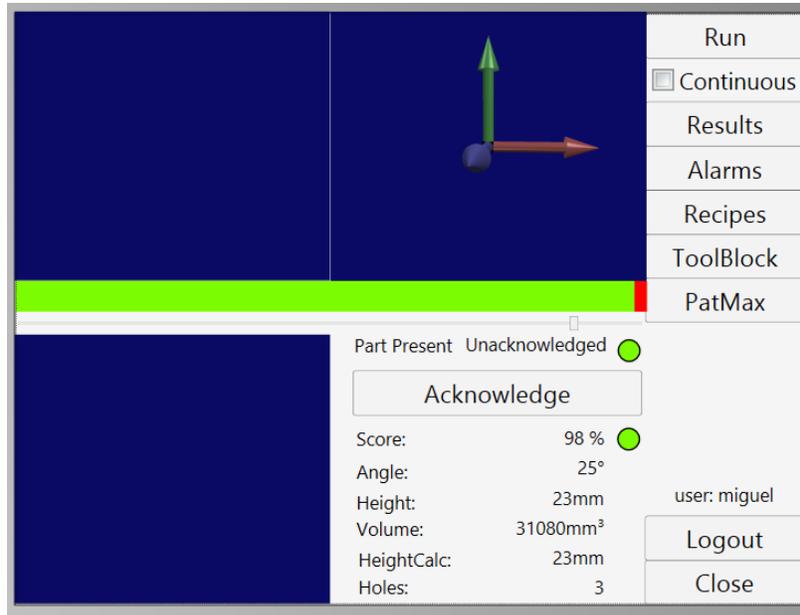
Access Tools Lab

Approximate Duration: 40 minutes

EXPECTED OUTCOMES:

- Add the ability to adjust the search region of PatMax
- Add the ability to adjust the training region of PatMax and to retrain the model
- Add the ability to access the VisionPro tool from the HMI

EXPECTED VISUAL RESULT:



OUTLINE OF LAB:

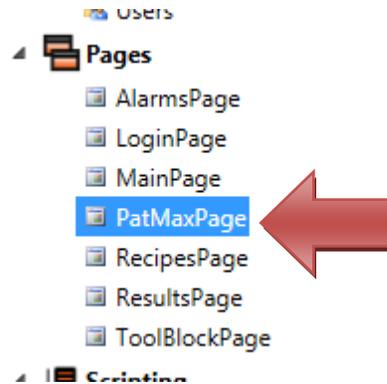
- 1. Add ToolEditor functionality to the HMI**
 - a. Add the ability to access the PatMax Tool through the HMI**
- 2. Add ability to adjust search region of PatMax**
- 3. Add ability to adjust training region of PatMax and ability to retrain the pattern**
 - a. Use Current Image to train**
 - b. Adjust the region to include origin**

Challenge: Add buttons to PatMax page to switch which tool is being used by the ToolEditor given the button pressed.

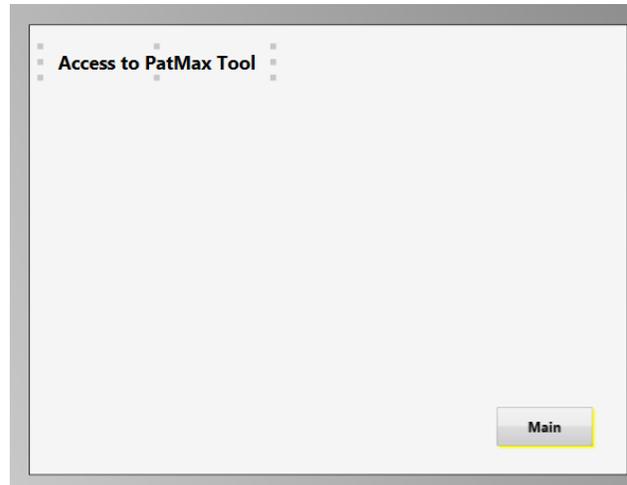
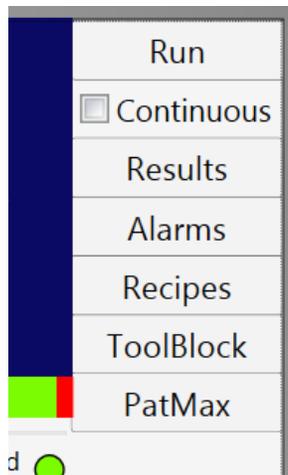
Steps for the Lab:

1. Add ToolEditor functionality to the HMI

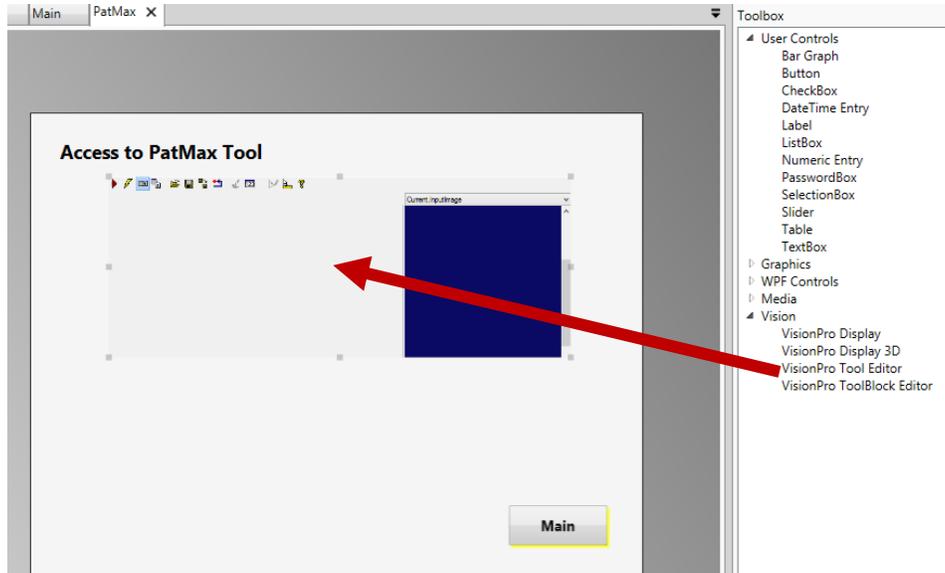
- Add a new page and name it PatMax



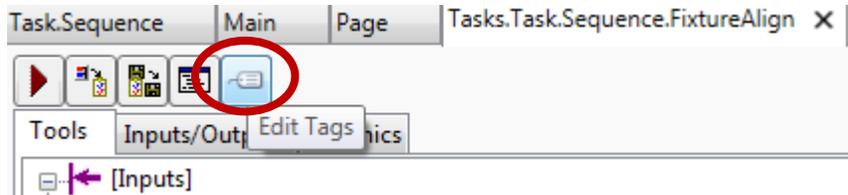
- Create labels and buttons on the PatMax Page and the Main Page to go back and forth



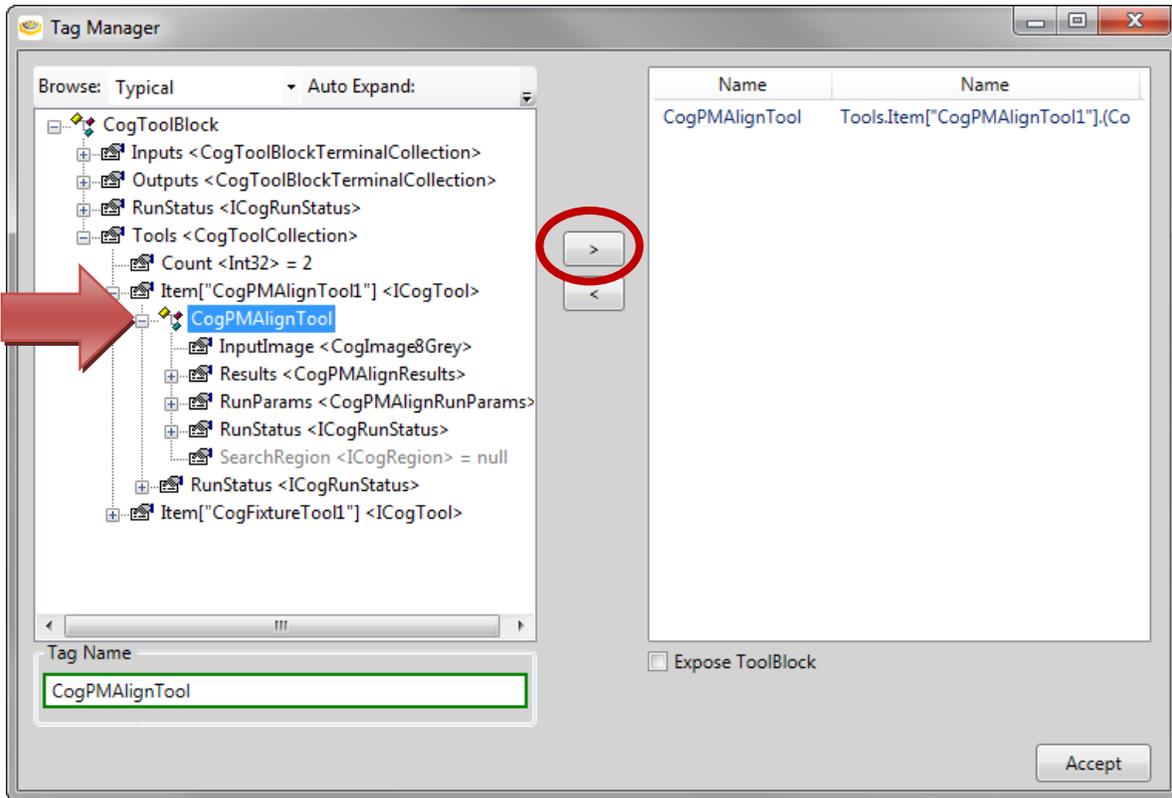
- Add a ToolEditor Object to the PatMax Page



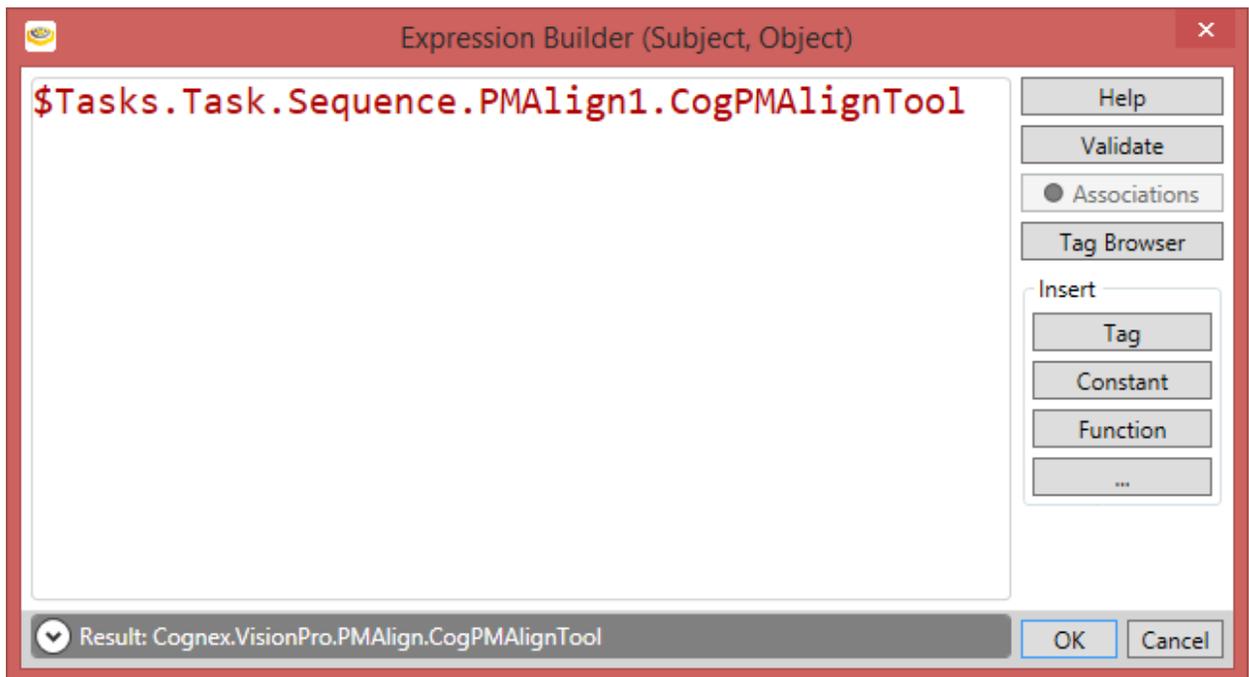
- a. Add the ability to access the PatMax Tool through the HMI
 - i. Go to the Sequence and open up the FixtureAlign ToolBlock and select "Edit Tags" button.



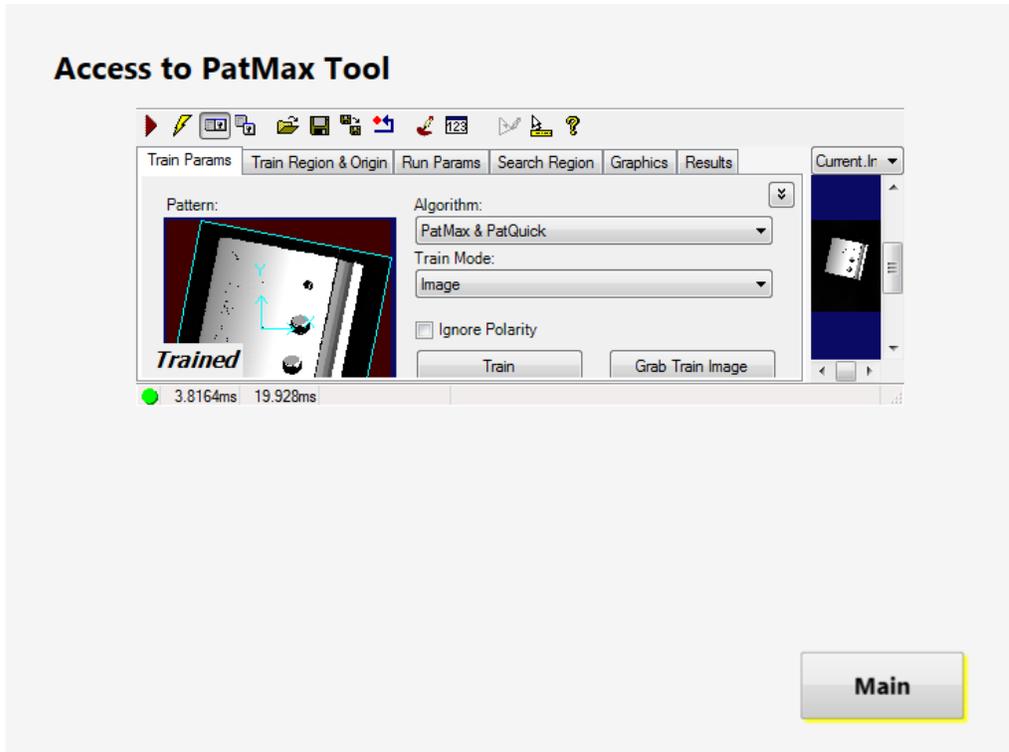
- ii. Go down through the Tools until you find the PatMax tool and add it to the list of tags and Accept.



- iii. Go back to the PatMax Page and reference the Subject Property of the ToolEditor to the PatMax tool tag just exposed

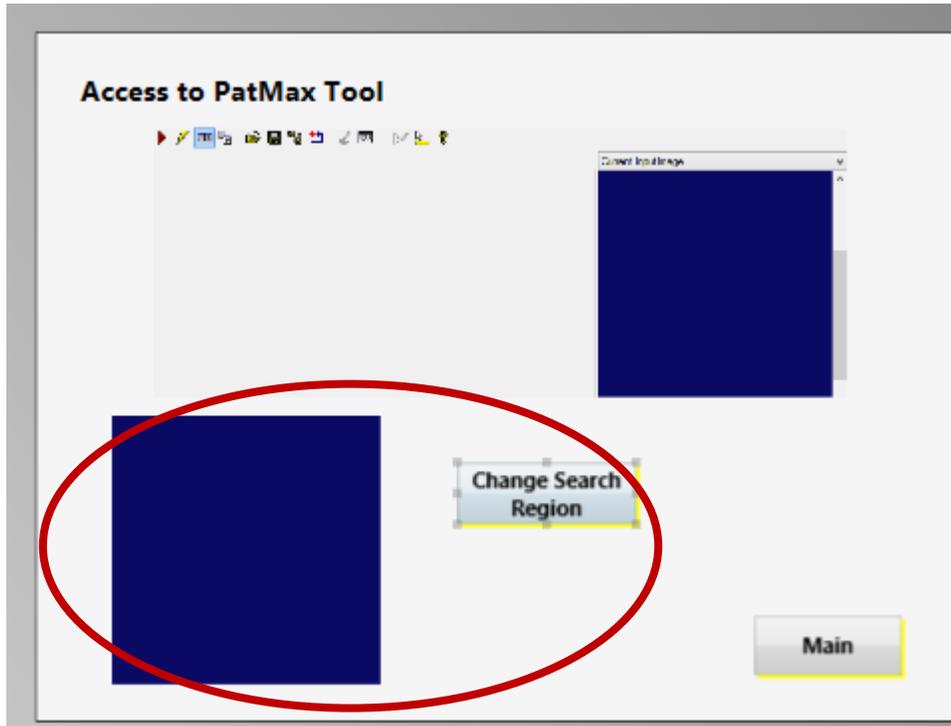


- iv. Go into Test Mode and check the connections.

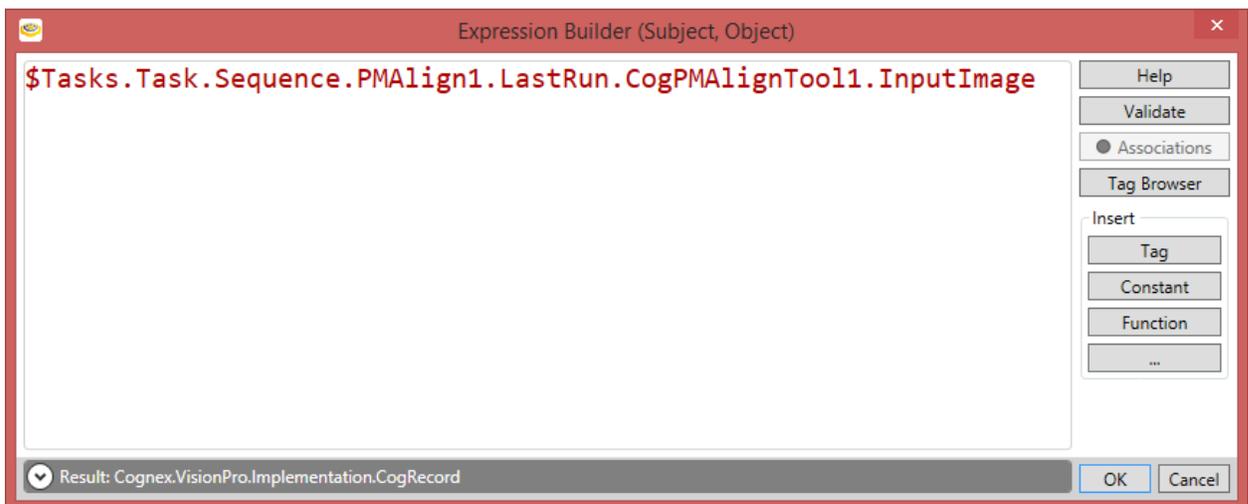


2. Add ability to adjust search region of PatMax

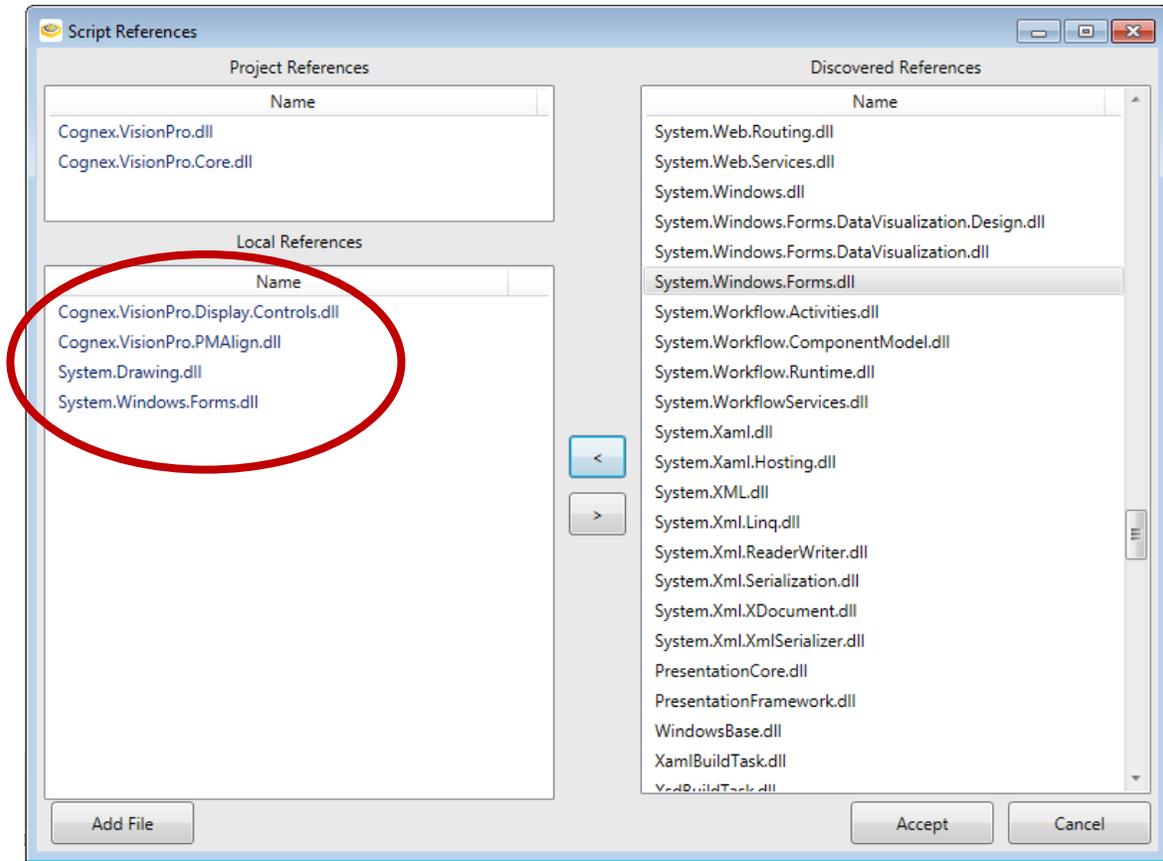
- Add a display to the PatMax page as well as a button that says “Change Search Region”.



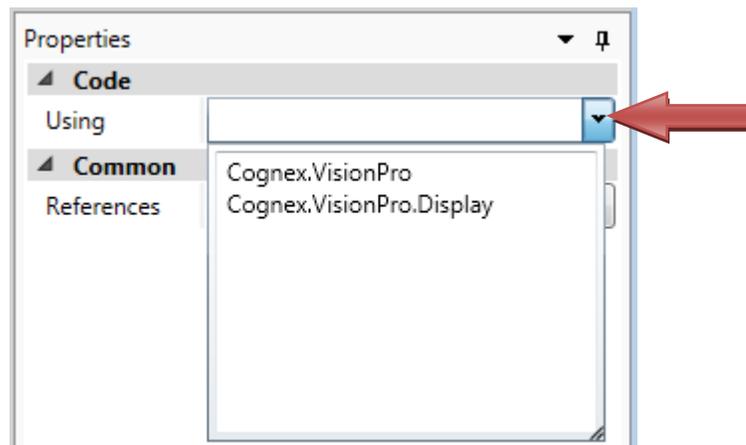
- Have the Display Subject Property reference the InputImage from the PatMax tool



- Go to the OnMouseClicked script of the Change Search Region button and reference the following .DLLs



- Add these Using References:
 - Cognex.VisionPro
 - Cognex.VisionPro.Display
- Don't be alarmed. The using reference is not visible after you leave the "Using" property unless you click on it again.



- Add the following code to the script:

```
// Gain control of the VisionPro display as a CogDisplay
CogDisplay myDisplay
= (CogDisplay)$System.GetInternalObject("Pages.PatMaxPage.VisionProDisplay");

// Clear any previous interactive graphics
myDisplay.InteractiveGraphics.Clear();

// Create a CogRectangleAffine and assign it the PMAAlign tool's search region property
CogRectangleAffine myRectangle
= (CogRectangleAffine)$Tasks.Task.Sequence.PMAAlign1.CogPMAAlignTool.SearchRegion;

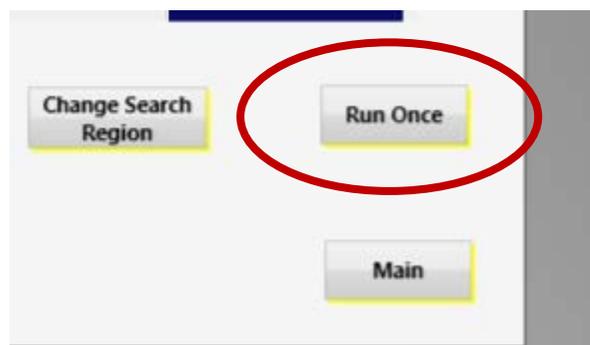
// Set the display's DrawingEnabled property to false before adding any graphics
myDisplay.DrawingEnabled = false;

// Add the CogRectangleAffine to the display
myDisplay.InteractiveGraphics.Add(myRectangle, "SearchRegion", false);

// Set the display's DrawinEnabled property to true so graphics are drawn just once
// instead of each time a graphic is added
myDisplay.DrawingEnabled = true;
```

```
Task.Sequence Button1.OnClick X PatMaxPage MainPage LoginPage
public void OnClick(Object sender, EventArgs e)
1 // Gain control of the VisionPro display as a CogDisplay
2 CogDisplay myDisplay = (CogDisplay)$System.GetInternalObject("Pages.PatMaxPage.VisionProDisplay");
3
4 // Clear any previous interactive graphics
5 myDisplay.InteractiveGraphics.Clear();
6
7 // Create a CogRectangleAffine and assign it the PMAAlign tool's search region property
8 CogRectangleAffine myRectangle = (CogRectangleAffine)$Tasks.Task.Sequence.PMAAlign1.CogPMAAlignTool.SearchRegion;
9
10 // Set the display's DrawingEnabled property to false before adding any graphics
11 myDisplay.DrawingEnabled = false;
12
13 // Add the CogRectangleAffine to the display
14 myDisplay.InteractiveGraphics.Add(myRectangle, "SearchRegion", false);
15
16 // Set the display's DrawinEnabled property to true so graphics are drawn just once
17 // instead of each time a graphic is added
18 myDisplay.DrawingEnabled = true;
19
```

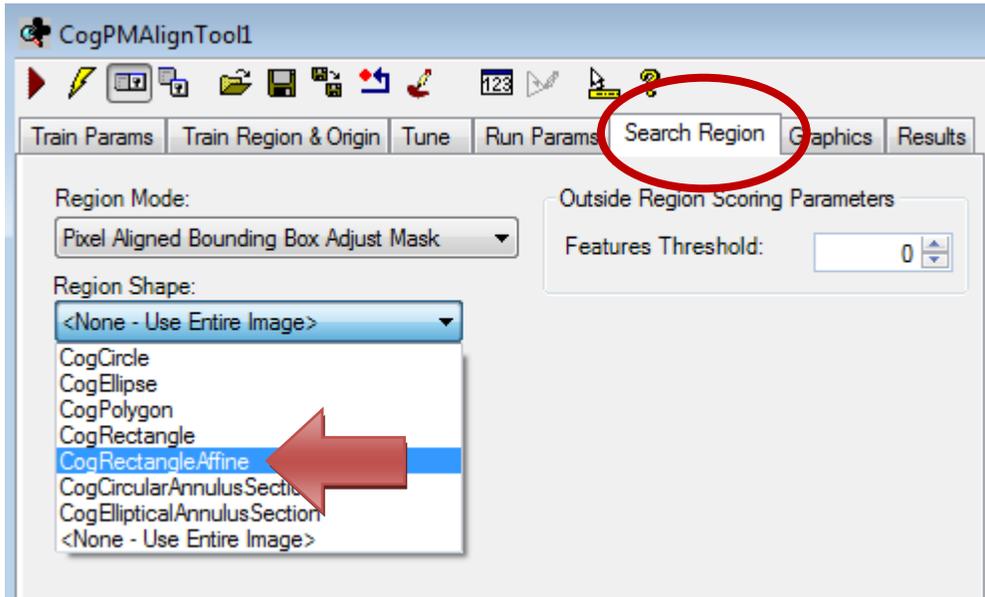
- To make things easier, add a button to the PatMax page cause the sequence to run



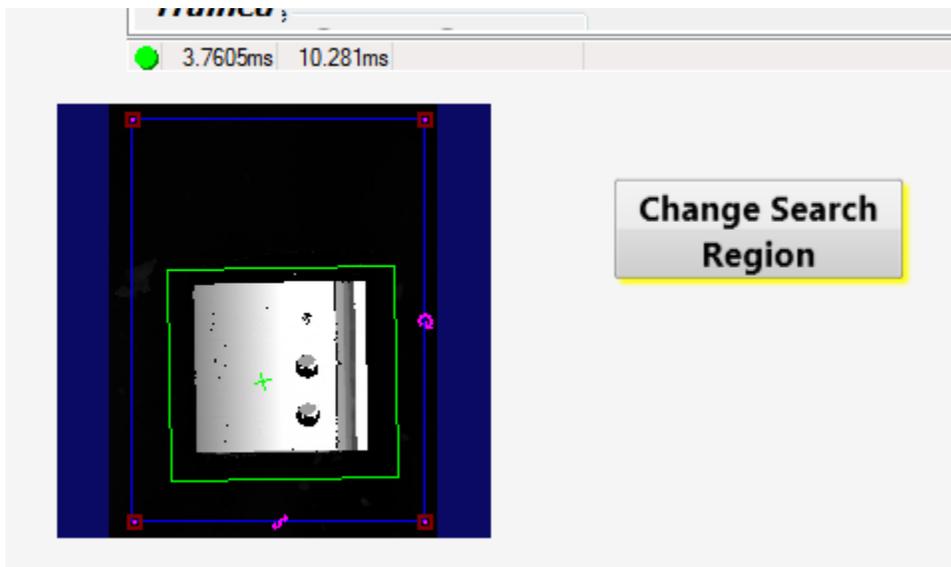
- Test out the code in Test Mode. You might note that the Search region does not change.

```
Output | Errors
Warn: Exception during Script (Pages.PatMax.Button1.OnClick) - Value does not fall within the expected range. - Script began execution at 2014-09-12 03:13:16,447
```

- This is because the Search Region of PatMax is still set to the whole region. Go to the ToolEditor, choose the SearchRegion tab and select CogRectangleAffine from the Region Shape list.

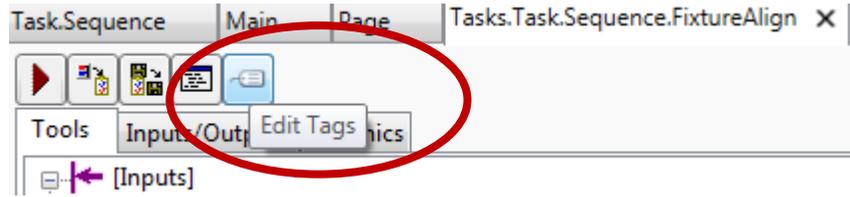


- Now test again. You should have access to the Search Region. Make sure your part stays within the Search Region you set.

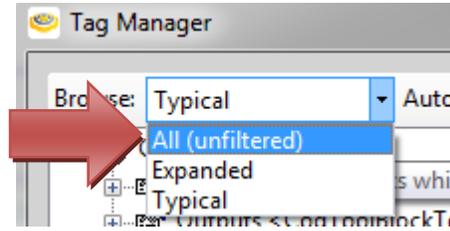


3. Add ability to adjust training region of PatMax and ability to retrain the pattern

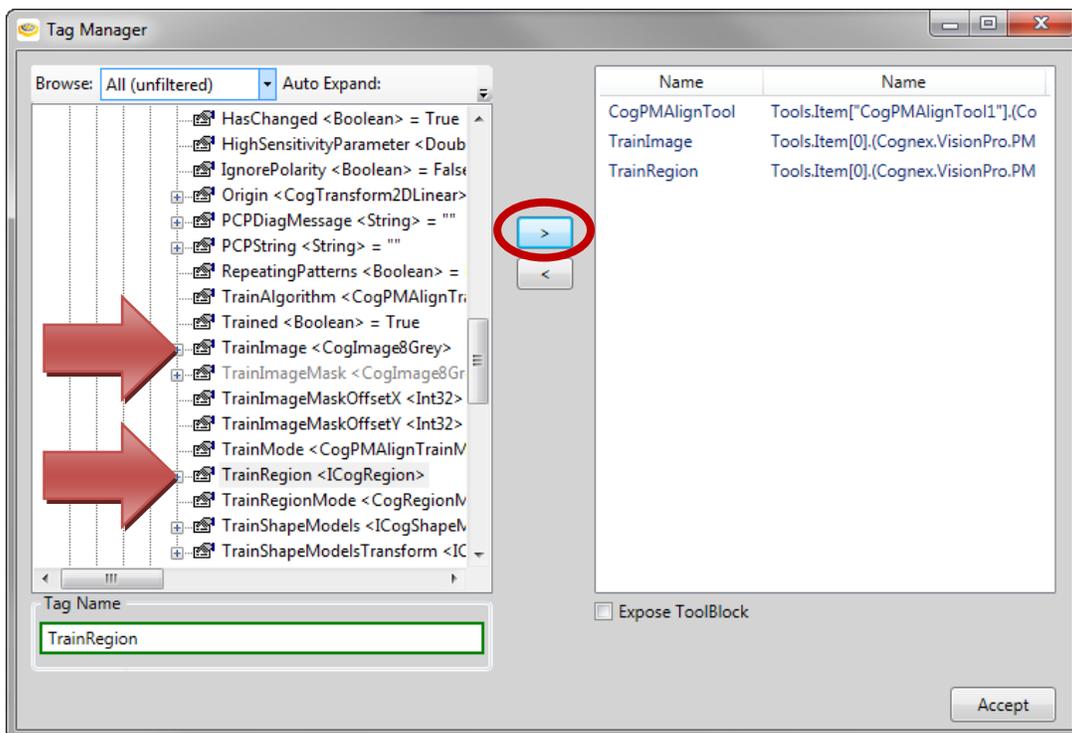
- Go into the FixtureAlign toolblock and go to the Tag Editor



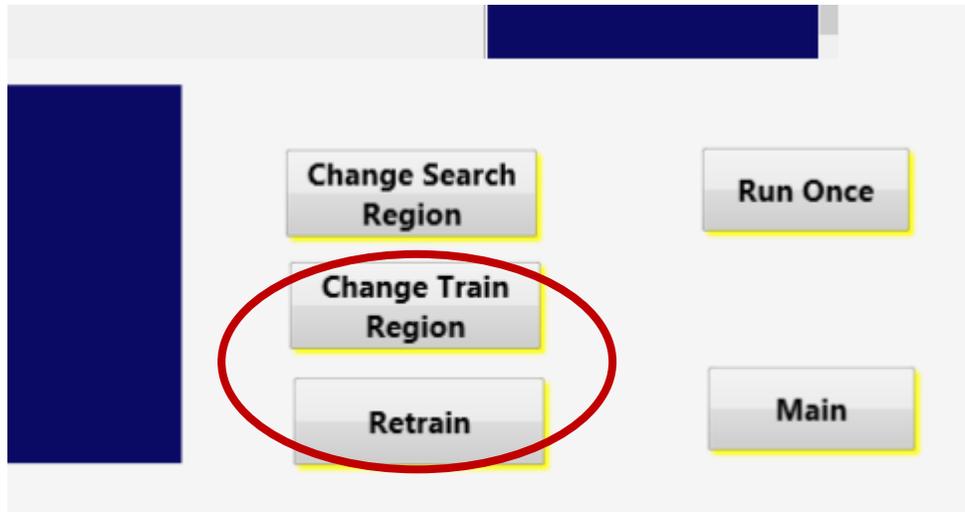
- Under the Browse selection, choose All (unfiltered)



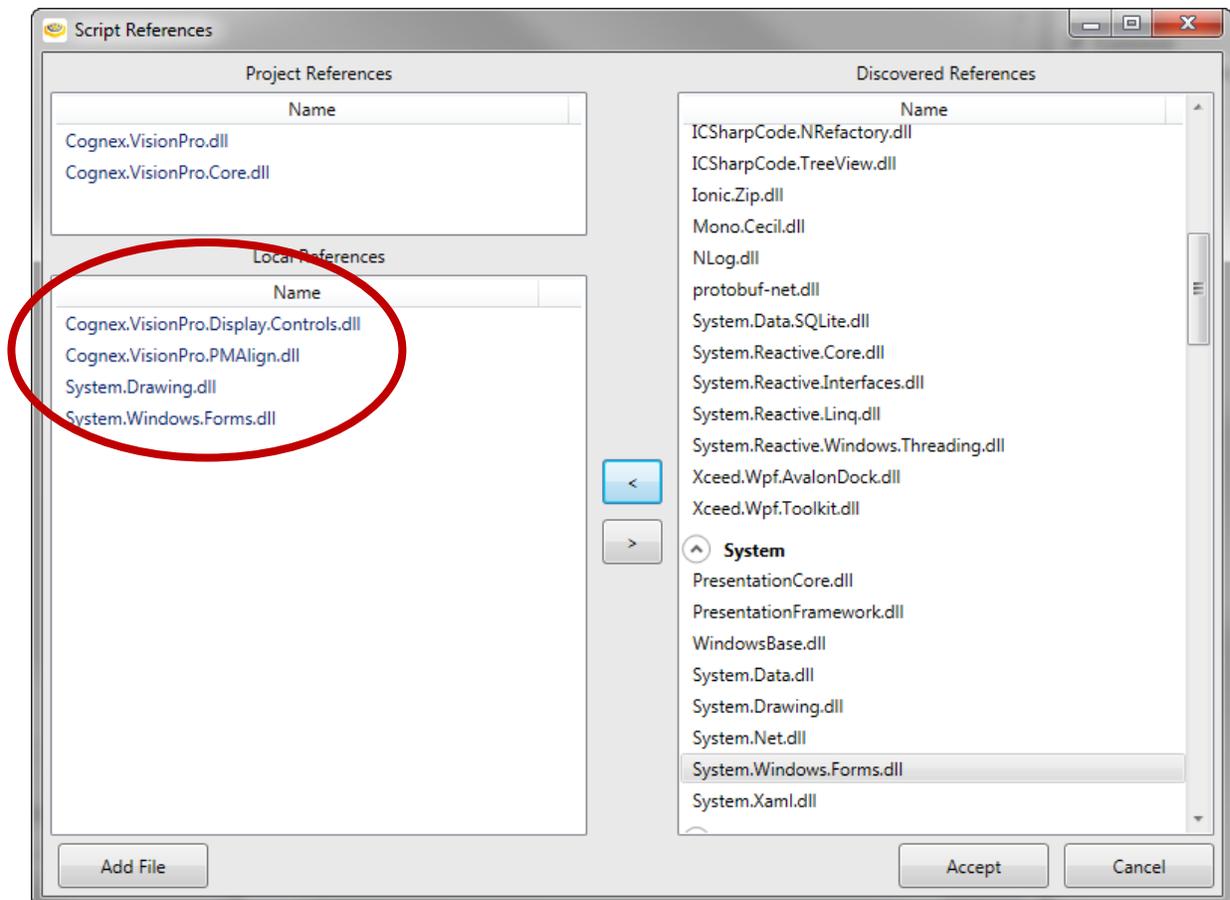
- Drill down to the Pattern and pull out the TrainImage and the TrainRegion (one at a time)



- a. Add two more buttons on the PatMaxAdjust page – one saying “Change Training Region” and the other saying “Retrain”



- b. Use Current Image to train
- Go to “OnClick” of “Change Train Region” and reference the same .DLLs and Using Reference as the Search Region button.



ii. Type the following code for Script

```
// Gain control of the VisionPro display as a CogDisplay
CogDisplay myDisplay
= (CogDisplay)$System.GetInternalObject("Pages.PatMaxPage.VisionProDisplay");

// Clear any previous interactive graphics
myDisplay.InteractiveGraphics.Clear();

// Create an ICogImage and assign it the Current.InputImage
ICogImage myInputImage = $Tasks.Task.Sequence.PMAlign1.CogPMAlignTool.InputImage;

// Set the TrainImage to the newly created ICogImage which is holding the Current.InputImage
// This is the equivalent clicking the "Grab Train Image" button in the PMAlign tool
// Careful: Your model is now UNTRAINED
$Tasks.Task.Sequence.PMAlign1.CogPMAlignTool.Pattern.TrainImage = myInputImage;

// Display the supplanted TrainImage
myDisplay.Image = $Tasks.Task.Sequence.PMAlign1.CogPMAlignTool.Pattern.TrainImage;

// Create a CogRectangleAffine to display to the user
CogRectangleAffine myRectangle
= (CogRectangleAffine)$Tasks.Task.Sequence.PMAlign1.CogPMAlignTool.Pattern.TrainRegion;

// Create a set of points using a CogCoordinateAxes()
CogCoordinateAxes myOrigin = new CogCoordinateAxes();

// Set myOrigin x, y, and rotation to those of the current pattern
myOrigin.OriginX = $Tasks.Task.Sequence.PMAlign1.CogPMAlignTool.Pattern.Origin.TranslationX; //
set x
myOrigin.OriginY = $Tasks.Task.Sequence.PMAlign1.CogPMAlignTool.Pattern.Origin.TranslationY; //
set y
myOrigin.Rotation = $Tasks.Task.Sequence.PMAlign1.CogPMAlignTool.Pattern.Origin.Rotation; // set
rotation

// Set myOrigin to be interactive
myOrigin.Interactive = true;

// Set myOrigin so you can edit all of its degrees of freedom (DOF)
myOrigin.GraphicDOFEnable = CogCoordinateAxesDOFConstants.All;

// Set the display's DrawingEnabled property to false before adding any graphics
myDisplay.DrawingEnabled = false;

// Add the CogRectangleAffine to the display
myDisplay.InteractiveGraphics.Add(myRectangle, "TrainRegion", false);
myDisplay.InteractiveGraphics.Add(myOrigin, "TrainOrigin", false);

// Set the display's DrawinEnabled property to true so graphics are drawn just once
// instead of each time a graphic is added
myDisplay.DrawingEnabled = true;
```

Task.Sequence	Button1.OnClick	LoginPage	MainPage	PatMaxPage	Button3.OnClick X
---------------	-----------------	-----------	----------	------------	-------------------

```

public void OnClick(Object sender, EventArgs e)
1 // Gain control of the VisionPro display as a CogDisplay
2 CogDisplay myDisplay = (CogDisplay)$System.GetInternalObject("Pages.PatMaxPage.VisionProDisplay");
3
4 // Clear any previous interactive graphics
5 myDisplay.InteractiveGraphics.Clear();
6
7 // Create an ICogImage and assign it the Current.InputImage
8 ICogImage myInputImage = $Tasks.Task.Sequence.PMAlign1.CogPMAlignTool.InputImage;
9
10 // Set the TrainImage to the newly created ICogImage which is holding the Current.InputImage
11 // This is the equivalent clicking the "Grab Train Image" button in the PMAlign tool
12 // Careful: Your model is now UNTRAINED
13 $Tasks.Task.Sequence.PMAlign1.CogPMAlignTool.Pattern.TrainImage = myInputImage;
14
15 // Display the supplanted TrainImage
16 myDisplay.Image = $Tasks.Task.Sequence.PMAlign1.CogPMAlignTool.Pattern.TrainImage;
17
18 // Create a CogRectangleAffine to display to the user
19 CogRectangleAffine myRectangle = (CogRectangleAffine)$Tasks.Task.Sequence.PMAlign1.CogPMAlignTool.Pattern.TrainRegion;
20
21 // Create a set of points using a CogCoordinateAxes()
22 CogCoordinateAxes myOrigin = new CogCoordinateAxes();
23
24 // Set myOrigin x, y, and rotation to those of the current pattern
25 myOrigin.OriginX = $Tasks.Task.Sequence.PMAlign1.CogPMAlignTool.Pattern.Origin.TranslationX; // set x
26 myOrigin.OriginY = $Tasks.Task.Sequence.PMAlign1.CogPMAlignTool.Pattern.Origin.TranslationY; // set y
27 myOrigin.Rotation = $Tasks.Task.Sequence.PMAlign1.CogPMAlignTool.Pattern.Origin.Rotation; // set rotation
28
29 // Set myOrigin to be interactive
30 myOrigin.Interactive = true;
31
32 // Set myOrigin so you can edit all of its degrees of freedom (DOF)
33 myOrigin.GraphicDOFEnable = CogCoordinateAxesDOFConstants.All;
34
35 // Set the display's DrawingEnabled property to false before adding any graphics
36 myDisplay.DrawingEnabled = false;
37
38 // Add the CogRectangleAffine to the display
39 myDisplay.InteractiveGraphics.Add(myRectangle, "TrainRegion", false);
40 myDisplay.InteractiveGraphics.Add(myOrigin, "TrainOrigin", false);
41
42 // Set the display's DrawnEnabled property to true so graphics are drawn just once
43 // instead of each time a graphic is added
44 myDisplay.DrawingEnabled = true;

```

- iii. Go to the Retrain button and Reference same references as before. Use the following command in the Script for “OnClick”.

// Execute the Train() method

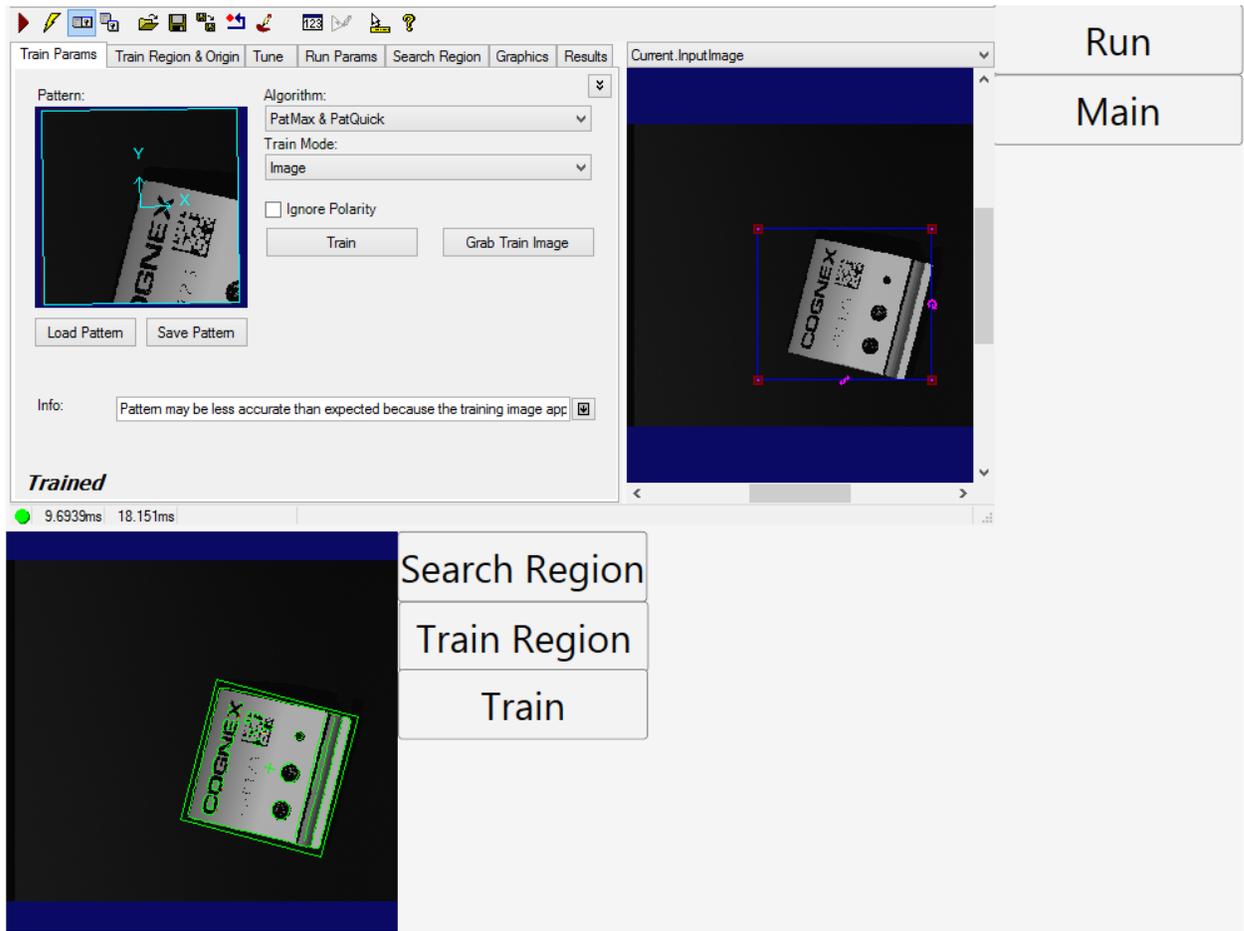
\$Tasks.Task.Sequence.PMAlign1.CogPMAlignTool.Pattern.Train();

Task.Sequence	Button1.OnClick	LoginPage	MainPage	PatMaxPage
---------------	-----------------	-----------	----------	------------

```

public void OnClick(Object sender, EventArgs e)
1 // Execute the Train() method
2 $Tasks.Task.Sequence.PMAlign1.CogPMAlignTool.Pattern.Train();

```



iv. Save the Application

Challenge: Add buttons to PatMax page to switch which tool is being used by the ToolEditor given the button pressed.

We have quite a few tools. Have the ToolEditor switch between the different tools given a button click or a pulldown box.