VISUAL COMPONENTS 4.0 ADD-ON

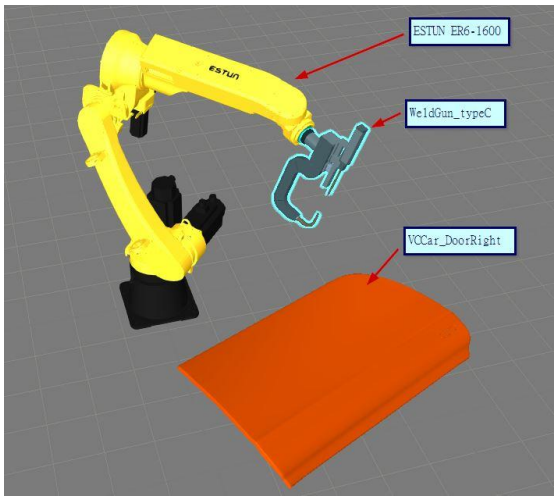
EASY SPOT

Jamal Muhammad
jamal.muhammad@visualcomponents.com
+358451827700

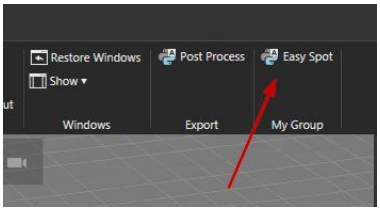
The Add-On is attached hereby. It's called [easyspot_2017_V1]. Please unzip the folder and put it in the following folder –

C:\Users\muhamJa1\Documents\Visual Components\4.0\My Commands

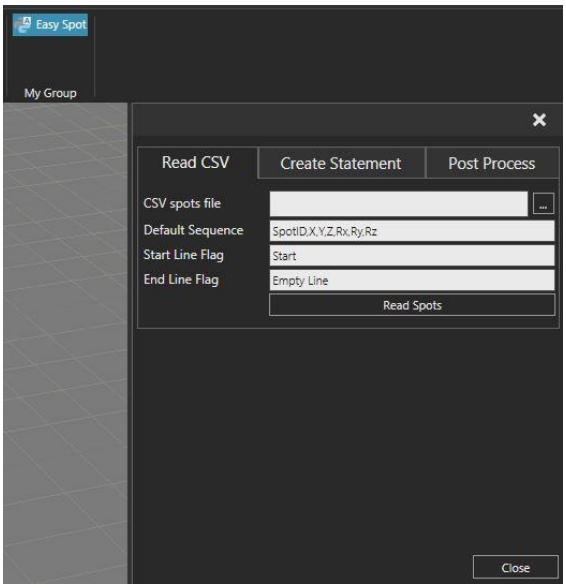
1. Now run VC Premium.
2. Once the SW started then go to Tab > [PROGRAM] and then select any medium size 6 DOF articulated robot e.g. in this tutorial ESTUN ER6-1600 robot is used.
3. Load the above robot into 3D World.
4. Then load the component [VCCar_DoorRight] from eCat.
5. Then load the WeldGun > [WeldGun_typeC] from eCat. Plug this EOAT to the robot.



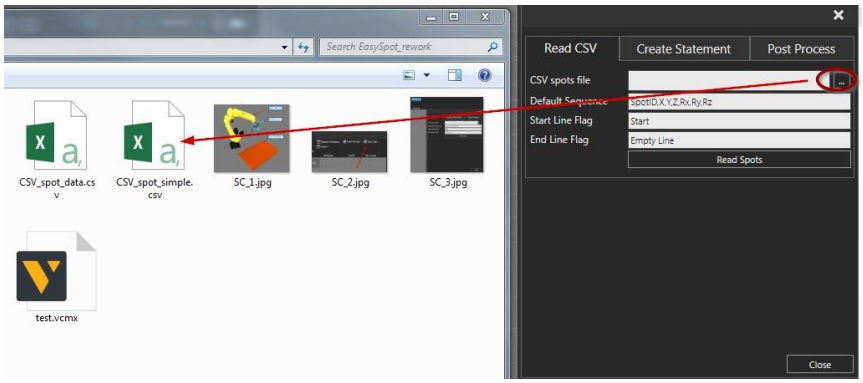
6. Now select the Robot (*this is important to select the Robot before activating the Add-On) press the button [Easy Spot].



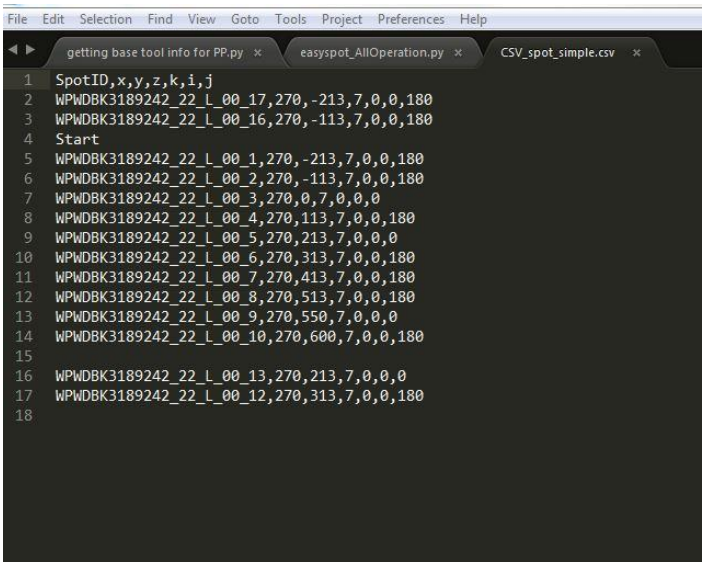
7. The Add-On should appear like this.



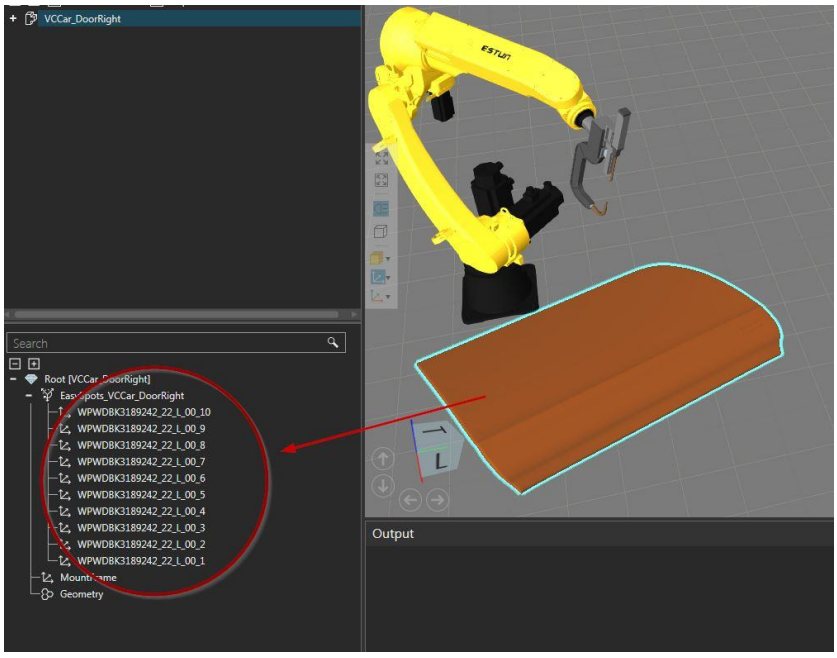
8. Now select the workpiece i.e. [VCCar_DoorRight] and then from the Property > Read CSV > CSV spots file, click on the (select uri) button and then select the *.csv file named (CSV_spot_simple.csv).



The *.csv file looks like as following –



9. Then press the button > [Read Spots], this will load the spot welding points as Frame feature into the part.



10. Then user can go to the next tab i.e. the Create Statement tab where the user can create the actual robot statements for spot welding.
11. Short description of the different option of this section are as following –

- Spot tag name filter: This option will filter the all selected type feature (Frame or Geometry) and then consider only those for the spot welding operations.
- Spot tag type: Select the type of Feature the user is searching from the workpiece. In this example we have already created Frame features from the *.csv file so we will select the [Frame] feature.
- Select Workpiece in 3D-world: When the user selects the workpiece, in this example it is the component [VCCar_DoorRight], then the Add-On populates all the necessary points (Frames or Geometry) position matrix information.

- Tool: Select the tool of the robot with which you need to create the spot welding statements. in this case it is (SpotTCP) which is the imported(to robot) TCP from the Spot Weld Gun.

- Base: Select the base of the robot. It can be any based on need.

- Weld Time: The time each spot weld operation will take.

- Tool Signal: The signal which will trigger the spot weld gun action of the Spot Weld Gun.

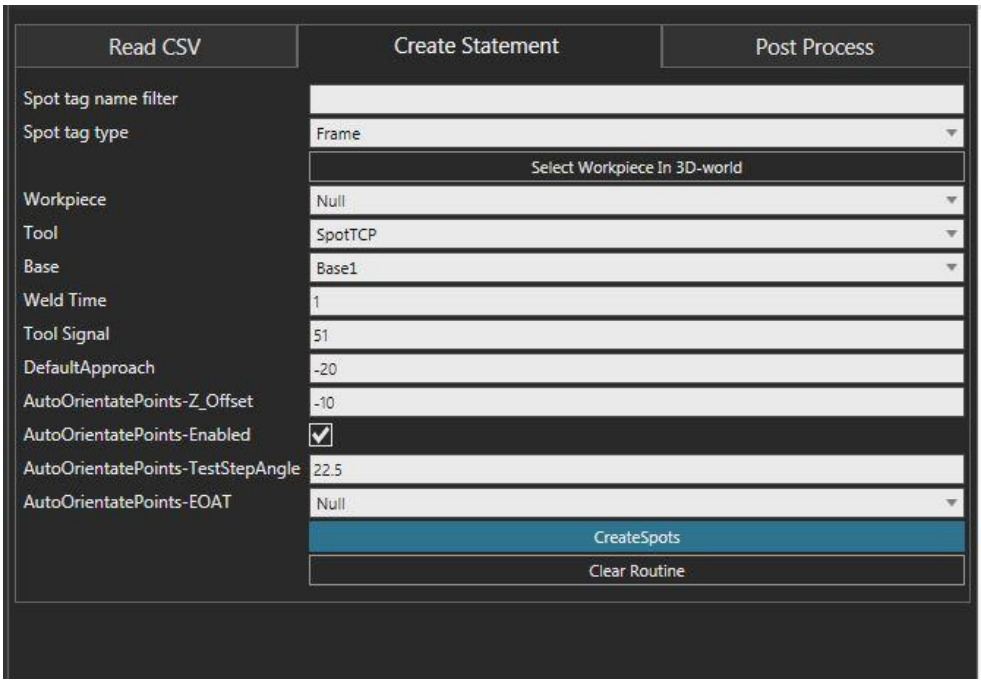
- Default Approach: This will make a via-point for robot TCP before and after every spot welding operation.

- AutoOrientatePoints-Z_offset: If there is a value here then the spot weld statement will be created at a default offset value of Z from the original Z value of the positions.

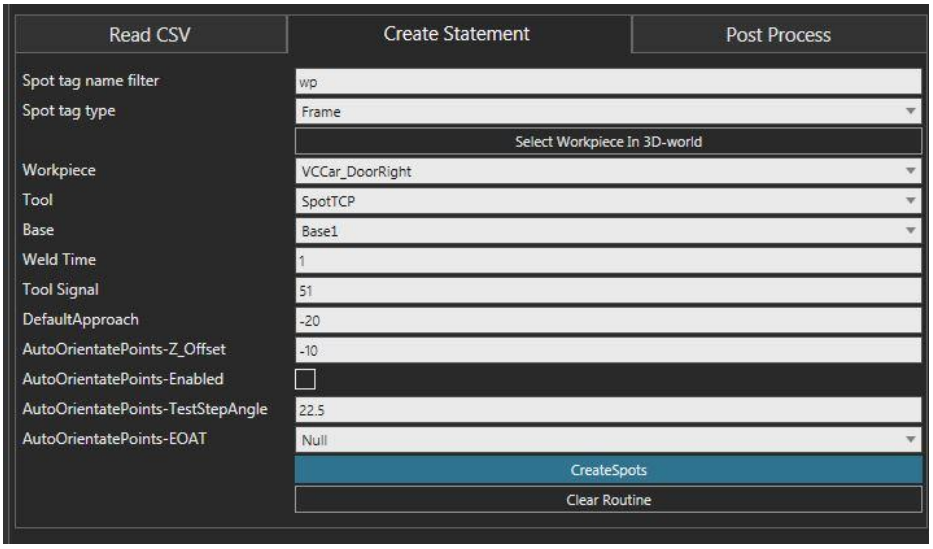
- AutoOrientatePoints-Enabled: If this option is checked then the logic of auto orientation will effect the weld statements as the logic will try to find the best possible orientation of the gun to avoid collision with workpiece.

- AutoOrientatePoints-TestStepAngle: The collision detection logic of the Add-On will rotate the Spot Weld gun every selected angle value e.g. 22 degree for collision checking and choose the best possible combination w.r.t previous spot statement orientation.

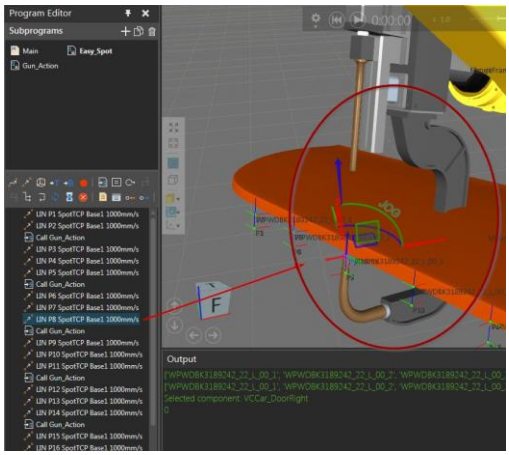
- AutoOrientatePoints-EOAT: Select here the EOAT for the collision checking operation. For this example the component [WeldGun_typeC] needs to be selected.



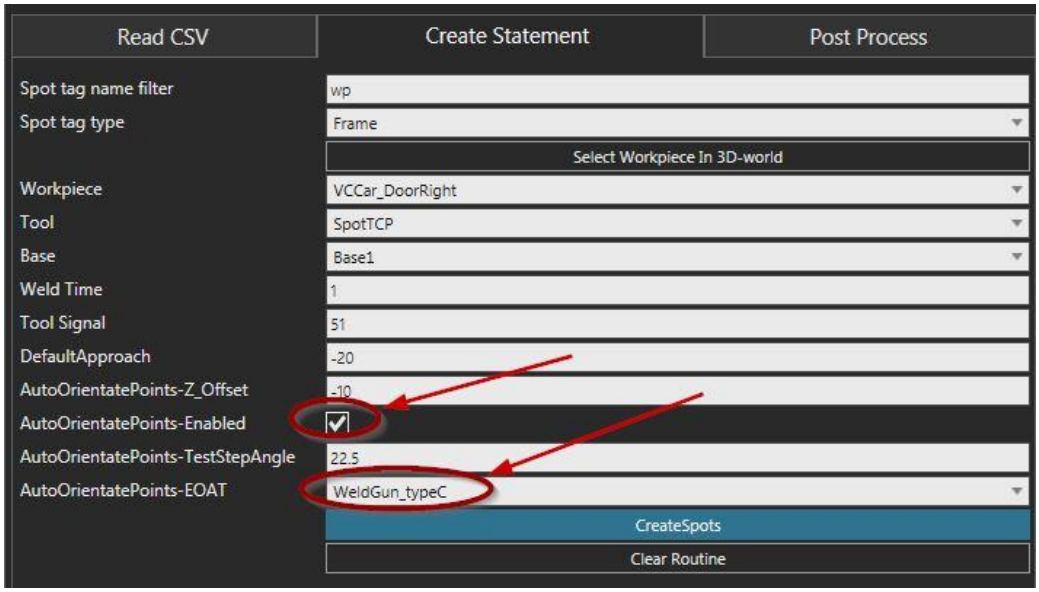
12. Now create the Spot Weld statements first with Option – AutoOrientatePoints-Enabled as OFF or Disabled. See the below setting-



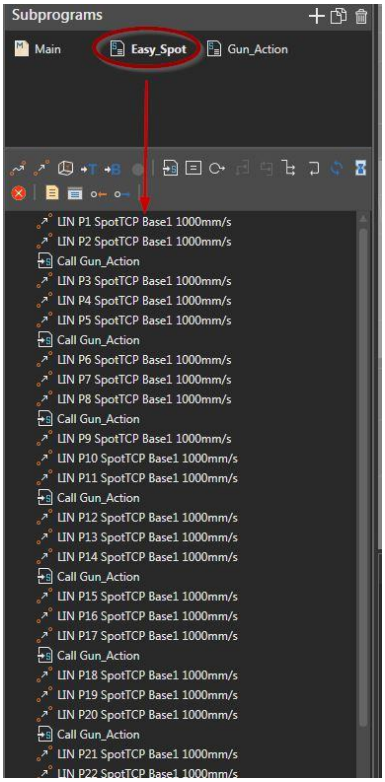
13. As you can see that the Statement LIN P8 has clear collision with the workpiece as the Auto Orientate Tool was not utilized of the Add-ON.

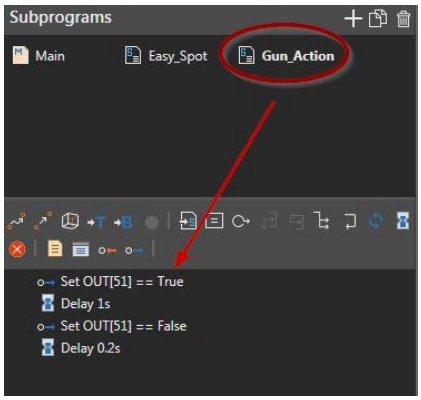


14. Now in the 2nd run we will create the Spot Weld statements with the Auto Orientate Tool > ON on ENABLED. See the following settings-



15. This results into collision free spot weld statements of all the points. The resulting sequence for the spot weld statements looks like below, there is sequence called [Gun_Action] which is called for every spot weld action or triggering ON/OFF spot weld gun.





16. This part (Post Process) is still not implemented.

17. The notable part of the python script are as following.

```
def rotateThis(eoat, ref_node, ref_frame, tool):
  '''
  uses the EOAT to rotate spots to an orientation where gun doesn't hit the part (if possible)
  '''
  global controller

  if not eoat or not getProperty('Create Statement::AutoOrientatePoints-Enabled').Value:
    #do not study the collision,, returning zero rotations for all
    collision_free_rot = 0
  else:
    tcp_prop    = tool
    if tcp_prop == '*NULL*':
      print 'Warning: Tool (tcp) not set. AutoOrientate cannot be calculated.'
      collision_free_rot = 0
      return collision_free_rot
    test_step       = getProperty( 'Create Statement::AutoOrientatePoints-TestStepAngle' )
    z_clearance     = getProperty('Create Statement::AutoOrientatePoints-Z_Offset')
    robot_loc       = controller.Component.WorldPositionMatrix
    eoatCachePos    = eoat.PositionMatrix
    toolifaces      = eoat.findBehavioursByType( VC_ONETOONEINTERFACE )
    eoatInvPos      = eoat.InverseWorldPositionMatrix
    tool_in_world   = tool.Node.WorldPositionMatrix * tool.PositionMatrix
    eoatcomp_to_tcp = eoatInvPos * tool_in_world
    eoatcomp_to_tcp.invert()
    step_angle = test_step.Value
    test_rots = [x*step_angle for x in range(int (360/step_angle) ) ]
    collision_free_rot = []
    detector = sim.newCollisionDetector()
    detector.NodeListA = [ (eoat, VC_NODELIST_INCLUDE, VC_NODELIST_TREE ) ]
    detector.NodeListB = [ (sim.World, VC_NODELIST_INCLUDE, VC_NODELIST_TREE ) , (eoat, VC_NODELIST_EXCLUDE, VC_NODELIST_TREE )]

    frame = ref_frame
    in_node = ref_node
    candidates = []
    for a in test_rots:
     mtx = in_node.WorldPositionMatrix * frame.NodePositionMatrix
     mtx.rotateRelZ( a )
     mtx.translateRel(0,0,z_clearance.Value)
     mtx = mtx * eoatcomp_to_tcp
     eoat.PositionMatrix = mtx
     eoat.update()
     app.render()
     hit = detector.testOneCollision(0.1)
     if not hit:
       candidates.append([a, eoat.WorldPositionMatrix])
    if candidates:
     closest = min( candidates, key = lambda x: ( x[1].P-robot_loc.P ).length() )
     collision_free_rot.append( closest[0] )
    else:
     collision_free_rot.append(0) #all collided,, use current
    eoat.PositionMatrix = eoatCachePos
    eoat.update()
    app.render()
    controller.update()

  return collision_free_rot
```

```
def read_spots_from_CSV_file(arg):

  sel = sel = app.SelectionManager
  comp = sel.getSelection(VC_SELECTION_COMPONENT)[0]
  if not comp:
    print 'Select a target component first and then try again'
    return
  else:
    if comp.findBehavioursByType(VC_RSLEXECUTOR):
      print 'Select a workpiece component, current selection is Robot'
      return

  uri_prop = getProperty('Read CSV::CSV spots file')
  uri = uri_prop.Value[8:]
  print 111,uri
  uri = 'C:\\Users\\muhamJa1\\Downloads\\EasySpot_rework\\CSV_spot_simple.csv'
  file = open( uri , 'r' )
  content = file.read()
  separator = '\n'
  lines = content.split(DEFAULT_LINE_SEPARATOR)
  start_line_flag = 0
  for line in lines:
    #terminate reading of *.csv file at empty line.
    if not line.strip():
      return

    #start reading *.csv file when [Start Line Flag] is read.
    new_line = line.split(',')
    if new_line[0] ==  getProperty('Read CSV::Start Line Flag').Value:
      start_line_flag = 1
    if start_line_flag == 0:
      continue

    # skip lines which are not according to default sequence.
    if len(new_line)<6 or new_line[1].isalpha()==True:
      continue

    spotname = new_line[0]
    x = float(new_line[1])
    y = float(new_line[2])
    z = float(new_line[3])
    rx = float(new_line[4])
    ry = float(new_line[5])
    rz = float(new_line[6])
    print spotname,x,y,z,rx,ry,rz

    target = vcMatrix.new()
    vec = vcVector.new(x,y,z)
    target.P = vec
    vec = vcVector.new(rx,ry,rz)
    target.WPR = vec
    a = 100
    easyspot_frames = comp.getFeature('EasySpots_' + comp.Name)
    if not easyspot_frames:
      easyspot_frames = comp.RootFeature.createFeature(VC_TRANSFORM, 'EasySpots_' + comp.Name)
    fea_frame = easyspot_frames.createFeature(VC_FRAME, spotname.strip() )
    fea_frame.PositionMatrix = target

    weld_tag_prop = fea_frame.createProperty( VC_BOOLEAN, 'WELD_SPOT_FRAME' )
    weld_tag_prop.Value = True
    weld_tag_prop.Group = a
    a += 10
    fea_frame.rebuild()

  app.render()
```