Appendix

For the article: Koch PJ, Albrecht M, Lin WC, Jost-Brinkmann PG. Accuracy of indirect bonding trays – a measurement algorithm. Int J Comput Dent 2022;25(3):295–302; doi: 10.3290/j.ijcd.b2599775

Script syntax written in computer language *Python* for the automated superimposition of bracket/tooth geometries and integrated calculation of bracket deviations by local best-fit alignment.

```
import geomagic.app.v2
for m in geomagic.app.v2.execStrings: exec m in locals(), globals()

import math
import time

#---Selection of teeth
#---All teeth
teeth_id =['11','12','13','14','15','16','17','21','22','23','24','25','26','27','31','32','33','34','35','36','37','41','42','43','44','45','46','47']

#-Single teeth or group of teeth
#teeth_id =['36','37','41','42','43','44','45','46','47']

#---Tolerance in mm
position_tol = 0.2 / 1000
#---Tolerance angle
angle_tol = 5.0

#---Run 3D Compare
run_3dcompare = False
#run_3dcompare = False

#---Create report
create_report = False
#create_report = True

#---Size of selection (BestFit)
sample_size_pass1 = 1500
sample_size_pass2 = 3000

#---File name and path to result file
#---Important! Path decorate with \\  e.g. "D:\\Result\\Patient1\\\result_" + ....
#filename = "D:\\result_" + str(time.time()) + ".csv"
filepath = "D:\\"

debug = False


def Align_Soll_To_CAD(Ist_Model, Soll_Model, CAD_Model):
  model_soll = Soll_Model
  model_cad = CAD_Model
  model_ist = Ist_Model

  mesh = geoapp.getMesh(model_soll)
  selector = DecomposeSelection(mesh)
```

```
  selector.run()
  selections = selector.components

  index_comp = 0
  min_volume = 1000.0
  points_align = None
  for i in range(0,len(selections)):
   duplicate = Duplicate()
   duplicate.object = mesh
   duplicate.run()
   temp_mesh = duplicate.clonedObject

   selector2 = DecomposeSelection(temp_mesh)
   selector2.run()
   selections2 = selector2.components


   mod = ReverseSelection(temp_mesh)
   mod.triSelection = selections2[i]
   mod.run()

   delTri = DeleteTriangles(temp_mesh)
   delTri.selection = selections2[i]
   delTri.run()
   if(temp_mesh.boundingBox.volume() < min_volume and temp_mesh.numTriangles > 1000):
    index_comp = i
    min_volume =  temp_mesh.boundingBox.volume()
    points_align = createPointsFromMesh(temp_mesh)


  bfAlign = BestFitAlign()
  bfAlign.fineAdjustmentsOnly = False
  bfAlign.sampleSize = 3000
  bfAlign.tolerance = 0.0001
  bfAlign.fixedObject = geoapp.getMesh(model_cad)
  bfAlign.floatingObject = points_align
  bfAlign.run()

  mesh.transform(bfAlign.testTransform)
  Ist_mesh = geoapp.getMesh(model_ist)
  Ist_mesh.transform(bfAlign.testTransform)
  return true

def Align_Ist_To_Soll(IST_Model, SOLL_Model, CAD_Model):
  model_ist = IST_Model
  model_soll = SOLL_Model
  model_cad = CAD_Model
  mesh = geoapp.getMesh(model_ist)
  points_align = None
  ptCover = CreatePointCover(mesh)
  ptCover.generateNormals = True
  ptCover.offset = 0.0
  ptCover.targetNumPoints = 100000
  ptCover.run()

  points_align = ptCover.points
```


(Int J Comput Dent 2022;25(3):295–302; doi: 10.3290/j.ijcd.b2599775)

```
    bfAlign = BestFitAlign()
    bfAlign.fineAdjustmentsOnly = False
    bfAlign.sampleSize = 3000
    bfAlign.tolerance = 0.0001
    bfAlign.fixedObject = geoapp.getMesh(model_soll)
    bfAlign.floatingObject = points_align
    bfAlign.run()

    mesh.transform(bfAlign.testTransform)
    points_align.transform(bfAlign.testTransform)

    mesh_cad = geoapp.getMesh(model_cad)
    compare = ComparePointsToMesh(mesh_cad)
    compare.points = points_align
    compare.maxDeviation = 0.0002

    compare.comparisonType = compare.ComparisonType_Regular3D
    compare.algorithmType = compare.AlgorithmType_Dense
    compare.criticalAngle = 45.0
    compare.run()

    comp3dinfo = compare.comparisonResult

    points_align.removeSelection(comp3dinfo.uncomparedPointSelection)


    bfAlign2 = BestFitAlign()
    bfAlign2.fineAdjustmentsOnly = False
    bfAlign2.sampleSize = 5000
    bfAlign2.tolerance = 0.0001
    bfAlign2.fixedObject = geoapp.getMesh(model_cad)
    bfAlign2.floatingObject = points_align
    bfAlign2.run()

    mesh.transform(bfAlign2.testTransform)


def Align_Ist_To_Soll_Fine(IST_Model, SOLL_Model, CAD_Model):

    model_ist = IST_Model
    model_soll = SOLL_Model
    model_cad = CAD_Model


    index_comp = 0
    max_volume = 0.0
    points_align = None

    points_align = createPointsFromMesh(geoapp.getMesh(model_ist))
    bfAlign = BestFitAlign()
    bfAlign.fineAdjustmentsOnly = False
    bfAlign.sampleSize = 5000
    bfAlign.tolerance = 0.0001
    bfAlign.fixedObject = geoapp.getMesh(model_soll)
    bfAlign.floatingObject = points_align
    bfAlign.run()
```

```
mesh_ist = geoapp.getMesh(model_ist)
mesh_ist.transform(bfAlign.testTransform)
points_align = createPointsFromMesh(geoapp.getMesh(model_ist))

mesh_soll = geoapp.getMesh(model_soll)
selector = DecomposeSelection(mesh_soll)
selector.run()
selections = selector.components

mesh_align_dup = Duplicate()
for i in range(0,len(selections)):
 duplicate = Duplicate()
 duplicate.object = mesh_soll
 duplicate.run()
 temp_mesh = duplicate.clonedObject

 selector2 = DecomposeSelection(temp_mesh)
 selector2.run()
 selections2 = selector2.components


 mod = ReverseSelection(temp_mesh)
 mod.triSelection = selections2[i]
 mod.run()

 delTri = DeleteTriangles(temp_mesh)
 delTri.selection = selections2[i]
 delTri.run()
 if(temp_mesh.boundingBox.volume() > max_volume and temp_mesh.numTriangles > 1000):
  index_comp = i
  max_volume =  temp_mesh.boundingBox.volume()
  mesh_align_dup.object = temp_mesh
  mesh_align_dup.run()



mesh_ist = geoapp.getMesh(model_ist)


compare = ComparePointsToMesh(mesh_align_dup.clonedObject)
compare.points = points_align

compare.maxDeviation = 0.0004
compare.spacing = 0.25
compare.comparisonType = compare.ComparisonType_Regular3D
compare.algorithmType = compare.AlgorithmType_Dense
compare.criticalAngle = 45.0
compare.run()

comp3dinfo = compare.comparisonResult

points_align.removeSelection(comp3dinfo.uncomparedPointSelection)

bfAlign = BestFitAlign()
bfAlign.fineAdjustmentsOnly = False
bfAlign.sampleSize = 5000
bfAlign.tolerance = 0.0001
```


(Int J Comput Dent 2022;25(3):295–302; doi: 10.3290/j.ijcd.b2599775)

```
  bfAlign.fixedObject = mesh_align_dup.clonedObject
  bfAlign.floatingObject = points_align
  bfAlign.run()

  mesh_ist.transform(bfAlign.testTransform)
  features = geoapp.getFeatures(model_ist)
  for feature in features:
   feature.transform(bfAlign.testTransform)

  if (debug==True):
   geoapp.addModel(points_align, u"PointCover")
   geoapp.addModel(mesh_align_dup.clonedObject, u"PointCover")

  compare2 = ComparePointsToMesh(mesh_align_dup.clonedObject)
  compare2.points = points_align

  compare2.maxDeviation = 0.0004
  compare2.spacing = 0.25
  compare2.comparisonType = compare.ComparisonType_Regular3D
  compare2.algorithmType = compare.AlgorithmType_Dense
  compare2.criticalAngle = 45.0
  compare2.run()

  comp3dinfo2 = compare2.comparisonResult



  return  bfAlign.testTransform.getTranslation()



def Copy_Features(IST_Model, SOLL_Model, CAD_Model):
  model_ist = IST_Model
  model_soll = SOLL_Model
  model_cad = CAD_Model

  features = geoapp.getFeatures(model_cad)
  for feature in features:


   if isinstance(feature, PointFeature):

     pt = PointFeature()
     pt.position = feature.position
     pt.name = feature.name
     pt2 = PointFeature()
     pt2.position = feature.position
     pt2.name = feature.name
     geoapp.addFeature(model_ist, pt)
     geoapp.addFeature(model_soll, pt2)
   if isinstance(feature, Line):

     line = Line()
     line.start = feature.start
     line.end = feature.end
     line.name = feature.name
```

```
        line2 = Line()
        line2.start = feature.start
        line2.end = feature.end
        line2.name = feature.name

        geoapp.addFeature(model_ist, line)
        geoapp.addFeature(model_soll, line2)

def Calc_Dev(IST_Model):
    model_ist = IST_Model
    features = geoapp.getFeatures(model_ist)
    str_angle = ""
    in_tol = ",Ok"
    x_rot = 0.0
    y_rot = 0.0
    z_rot = 0.0
    x_trans = 0.0
    y_trans = 0.0
    z_trans = 0.0

    for feature in features:
      if isinstance(feature, PointFeature):
        if feature.name == 'Punkt 1':
         x_trans = feature.position.x()
         y_trans = feature.position.y()
         z_trans = feature.position.z()



    for feature in features:
      if isinstance(feature, Line):
        if feature.name == 'Linie 1':
          plane1 = Plane()
          plane1.normal = Vector3D(0.0, 0.0, 1.0)
          plane1.origin = Vector3D(0.0, 0.0, 0.0)
          plane1.name = "Plane"

          proj_point = plane1.project(Vector3D(feature.end.x()-x_trans, feature.end.y()-y_trans, feature.end.z()-z_trans))


          line1 = Line()
          line1.start = Vector3D(0.0, 0.0, 0.0)
          line1.end = proj_point
          line1.name = "L1 Proj to x"

          dot_prod = dot(line1.direction,Vector3D(1.0,0.0,0.0))

          angle = math.acos(dot_prod / (abs(line1.direction)*abs(Vector3D(1.0,0.0,0.0))) )
          angle_deg = Angle.convertTo(Angle.Degrees,angle)

          vz = 1;
          if(proj_point.y()<0):
           vz= -1


          z_rot = angle_deg*vz
```


(Int J Comput Dent 2022;25(3):295–302; doi: 10.3290/j.ijcd.b2599775)

```
    if feature.name == 'Linie 1':
      plane2 = Plane()
      plane2.normal = Vector3D(0.0, 1.0, 0.0)
      plane2.origin = Vector3D(0.0, 0.0, 0.0)
      proj_point = plane2.project(Vector3D(feature.end.x()-x_trans, feature.end.y()-y_trans, feature.end.z()-z_trans))
      line2 = Line()
      line2.start = Vector3D(0.0, 0.0, 0.0)
      line2.end = proj_point
      line2.name = "L1 Proj to y"

      dot_prod = dot(line2.direction,Vector3D(1.0,0.0,0.0))


      angle = math.acos(dot_prod / (abs(line2.direction)*abs(Vector3D(1.0,0.0,0.0))) )

      angle_deg = Angle.convertTo(Angle.Degrees,angle)

      vz = 1;
      if(proj_point.z()>0):
       vz= -1

      y_rot = angle_deg*vz

    if feature.name == 'Linie 2':
      plane3 = Plane()
      plane3.normal = Vector3D(1.0, 0.0, 0.0)
      plane3.origin = Vector3D(0.0, 0.0, 0.0)
      proj_point = plane3.project(Vector3D(feature.end.x()-x_trans, feature.end.y()-y_trans, feature.end.z()-z_trans))
      line3 = Line()
      line3.start = Vector3D(0.0, 0.0, 0.0)
      line3.end = proj_point
      line3.name = "L2 Proj to x"

      dot_prod = dot(line3.direction,Vector3D(0.0,1.0,0.0))


      angle = math.acos(dot_prod / (abs(line3.direction)*abs(Vector3D(0.0,1.0,0.0))) )

      angle_deg = Angle.convertTo(Angle.Degrees,angle)

      vz = 1;
      if(proj_point.z()<0):
       vz= -1

      x_rot = angle_deg*vz

  str_angle = str_angle + "," + '%.3f' % (x_trans*1000)

  if x_trans > position_tol or x_trans < position_tol*-1.0:
    str_angle = str_angle + „Fehler"
  else:
    str_angle = str_angle + in_tol
  str_angle = str_angle + "," + '%.3f' % (y_trans*1000)
  if y_trans > position_tol or y_trans < position_tol*-1.0:
    str_angle = str_angle + „Fehler"
  else:
    str_angle = str_angle + in_tol
```

```
    str_angle = str_angle  + "," + '%.3f' % (z_trans*1000)
    if z_trans > position_tol or z_trans < position_tol*-1.0:
      str_angle = str_angle + „,Fehler"
    else:
      str_angle = str_angle + in_tol

    str_angle = str_angle + "," + '%.2f' % x_rot
    if x_rot > angle_tol or x_rot < angle_tol*-1.0:
     str_angle = str_angle + „,Fehler"
    else:
     str_angle = str_angle + in_tol

    str_angle = str_angle + "," + '%.2f' % y_rot
    if y_rot > angle_tol or y_rot < angle_tol*-1.0:
     str_angle = str_angle + „,Fehler"
    else:
     str_angle = str_angle + in_tol

    str_angle = str_angle + "," + '%.2f' % z_rot
    if z_rot > angle_tol or z_rot < angle_tol*-1.0:
     str_angle = str_angle + „,Fehler"
    else:
     str_angle = str_angle + in_tol

    return str_angle



pat_name = ""

openDialog = gui.OpenFileDialog()
file_Name = openDialog.getFilePath(u"c:\\Users\\Public", u"Wrap File (*.wrp)", 0)

if len(file_Name) != 0:
    print "File name:", file_Name
    pat_name = os.path.splitext(os.path.basename(file_Name))[0]
    pat_name = pat_name.split(',')[0];

    geoapp.openFile(file_Name)


    filename = filepath + "result2_" + pat_name + ".csv"
    file = open(filename, "w")
    file.write("Patient,Zahn,X,X-Status,Y,Y-Status,Z,Z-Status,X-Rot,X-Rot-Status,Y-Rot,Y-Rot-Status,Z-Rot,Z-Rot-Status\n")
    file.close()

    for id in range(0,len(teeth_id)):

      model_ist = geoapp.getModelByName(teeth_id[id]+" IST")
      model_soll = geoapp.getModelByName(teeth_id[id]+" SOLL")
      model_cad = geoapp.getModelByName(teeth_id[id]+"CAD Metall")
      if model_cad == None:
        model_cad = geoapp.getModelByName(teeth_id[id]+"CAD Tube")
      if model_cad == None:
        model_cad = geoapp.getModelByName(teeth_id[id]+"CAD Keramik")

      if model_ist != None:
```


(Int J Comput Dent 2022;25(3):295–302; doi: 10.3290/j.ijcd.b2599775)

```
    print model_ist.name


  teeth_str = pat_name + "," + teeth_id[id]
  if model_cad != None and model_soll != None:
    file = open(filename, "a")
    trans = None
    Align_Soll_To_CAD(model_ist, model_soll, model_cad)
    Align_Ist_To_Soll(model_ist, model_soll, model_cad)
    Copy_Features(model_ist, model_soll, model_cad)
    trans = Align_Ist_To_Soll_Fine(model_ist, model_soll, model_cad)
    tdev_str = Calc_Dev(model_ist)
    teeth_str = teeth_str + tdev_str + "\n"
    file.write(teeth_str)
    file.close()

    if run_3dcompare == True:
      geoapp.setNamedReferenceModel("TestObject", model_ist)
      geoapp.setNamedReferenceModel("RefObject", model_soll)
      geo.qual_3d_compare(0, 4, 2, 0.00055, 4, 1, 13, 0.0004, 0.0002, -0.0002, -0.0004, 45, -1, 0, 0, u", 0, 3)
      if create_report == True:
        temp_name = model_ist.name
        model_ist.name = temp_name + " " + pat_name
        geoapp.createReport()
        model_ist.name = temp_name

else:
  print "Cancelled!"
```