MATLAB Code for manuscript:

**Is it possible to detect a true Rotation Axis of the Temporomandibular Joint with common Pantographic Methods? A fundamental kinematic Analysis.**

Tested for MATLAB Version R2018a

(hint: please be careful with line feeds pasting this document into MATLAB)

%Pantographic Method Simulation (Near TMJ Trace Tracking Method)

%Fundamental Kinematic Measurement Simulation

% Author: Albert Mehl

% Corresponding Paper: Is it possible to detect a true Rotation Axis of the Temporomandibular Joint

% with common Pantographic Methods? A fundamental kinematic Analysis.

% in: Computer Methods in Biomechanics and Biomedical Engineering, Feb 2020

% July 2018; Last revision: 01-2020

%Values for Model:

% Radius(in mm):

r = 90.7;

% Length of Arc (Opening) (in mm):

b = -15;

% Center of Circle (Rotation):

center = [0; 0];

% angle in rad:

alpha = b/r;

% angle in degree:

fprintf('Angle: %i deg\n', rad2deg(alpha));

%create schematic jaw object

r\_object = 8; % in mm

centerobject = [0;0];

steps\_t1 = linspace(0, pi, 30);

part1\_object = [centerobject(1) + r\_object \*cos(steps\_t1); centerobject(2) + r\_object \*sin(steps\_t1)];

part2\_object = [linspace(centerobject(1)-r\_object,centerobject(1)-r\_object,30); linspace(centerobject(2),centerobject(2)-53,30)];

part3\_object = [linspace(centerobject(1)-r\_object,centerobject(1)+85,30); linspace(centerobject(2)-53,centerobject(2)-53,30)];

part4\_object = [linspace(centerobject(1)+85,centerobject(1)+85,30); linspace(centerobject(2)-53,centerobject(2)-35,30)];

part5\_object = [linspace(centerobject(1)+85,centerobject(1)+r\_object,30); linspace(centerobject(2)-35,centerobject(2)-35,30)];

part6\_object = [linspace(centerobject(1)+r\_object,centerobject(1)+r\_object,30); linspace(centerobject(2)-35,centerobject(2),30)];

object\_total = horzcat(part1\_object, part2\_object, part3\_object, part4\_object, part5\_object, part6\_object);

point\_ref1 =[centerobject(1); centerobject(2) + r\_object];

point\_ref2 = [centerobject(1)+85;centerobject(2)-35];

point\_ref3 = [centerobject(1)+85;centerobject(2)-53];

point\_ref4 = [centerobject(1)+50;centerobject(2)-35];

point\_ref5 = [centerobject(1)+50;centerobject(2)-53];

object\_reference\_points = horzcat(centerobject, point\_ref1, point\_ref2, point\_ref3, point\_ref4, point\_ref5);

%simulate movements steps with rotation and translation

n\_steps\_alpha = 20; %number of incremental steps of rotation angle alpha

for i\_shift = 0:3

SArcLength = 0; %reset arc length

%uniform movement

alpha\_step = linspace(0, alpha, n\_steps\_alpha); %time interval function for alpha

t\_steps = [linspace(0,i\_shift,n\_steps\_alpha); linspace(0,-i\_shift/2,n\_steps\_alpha)]; %corresponding

%translation to each rotation increment

%necessary for iterative movement simulation

p\_end\_rot = object\_reference\_points; %start with initial orientation

p\_object\_end\_rot = object\_total;

%set position of marker measurement reference coordinate system (external

%coordinate system) => origin is position of origin of fixed mandible

%coordinate system in external coordinate system

origin = [0;0];

%transfer internal coordinates to external coordinates for plotting

object\_reference\_points\_origin = object\_reference\_points + origin;

object\_total\_origin = object\_total + origin;

%set figures

fig1 = figure('Name','Mandible in Extern Reference Coordinate System');

%fig2 = figure('Name','Position of ICR');

fig3 = figure('Name','Pathways near global center of rotation: ARC LENGTH');

fig4 = figure('Name','Pathways near global center of rotation: MAX DISTANCE');

figure(fig1);

plot(object\_total\_origin(1,:),object\_total\_origin(2,:),'b');

hold on;

plot(object\_reference\_points\_origin(1,:), object\_reference\_points\_origin(2,:), '+r');

axis equal;

hold on;

%Determine first center of the entire movement to get the mid point for the

%visualisation of fig 3 and 4 -> should be equal to the last entire movement at the end of this program

%additional rotation center of entire movement!

p\_start\_rot0\_all = object\_reference\_points; %start with initial position

p\_start\_rot0 = p\_start\_rot0\_all(:,3:6);

p\_start\_markers0 = p\_start\_rot0 + origin;

R\_2dim = [cos(alpha) -sin(alpha); sin(alpha) cos(alpha)];

p\_end\_rot0 = R\_2dim \* p\_start\_rot0;

%in translated coordinate system

p\_end\_markers0 = p\_end\_rot0 + origin + t\_steps(:,n\_steps\_alpha);

X0\_trans = p\_start\_markers0;

X0 = transpose(X0\_trans);

Y0\_trans = p\_end\_markers0; %take last position from above

Y0 = transpose(Y0\_trans);

[error,Xrot,transformation] = procrustes(X0,Y0,'scaling',false,'reflection',false);

I = eye(2);

A = transformation.T - I;

translation\_help = [transformation.c(1,1);transformation.c(1,2)];

translation = -(transformation.T\*translation\_help);

t\_vec = A\translation;

CoR\_x\_global = -t\_vec(1); % minus, because new origin + t\_vec = 0!!

CoR\_y\_global = -t\_vec(2);

figure(fig3);

plot(CoR\_x\_global, CoR\_y\_global,'ks','MarkerSize',12);

hold on;

%Create sample points around Center of Rotation:

n\_gridsamples = 7;

rangex = 5;

rangey = 5;

x0\_samples = linspace(CoR\_x\_global - rangex, CoR\_x\_global + rangex, n\_gridsamples);

y0\_samples = linspace(CoR\_y\_global - rangex, CoR\_y\_global + rangey, n\_gridsamples);

TempSample = [linspace(x0\_samples(1),x0\_samples(1),n\_gridsamples); y0\_samples];

p\_samples = TempSample;

for ixgrid = 2:n\_gridsamples

TempSample = [linspace(x0\_samples(ixgrid),x0\_samples(ixgrid),n\_gridsamples); y0\_samples];

p\_samples = horzcat(p\_samples,TempSample);

end

plot(p\_samples(1,:), p\_samples(2,:),'+');

hold on;

p\_samples\_end\_rot = p\_samples - origin;

figure(fig4);

plot(CoR\_x\_global, CoR\_y\_global,'ks','MarkerSize',12);

hold on;

plot(p\_samples(1,:), p\_samples(2,:),'+');

hold on;

for ki0=1:(n\_gridsamples\*n\_gridsamples)

p\_samples\_all(1,1,ki0) = p\_samples(1,ki0);

p\_samples\_all(1,2,ki0) = p\_samples(2,ki0);

end

%start iterative movement with parameter settings from above

%IMPORTANT: rotation center is always set to (0,0) in the intern coordinate

%system!! -> to change rotation center -> shift entire object with respect

%to internal coordinates

for i\_alpha = 2:n\_steps\_alpha

delta\_alpha = alpha\_step(i\_alpha) - alpha\_step(i\_alpha-1);

R\_2dim = [cos(delta\_alpha) -sin(delta\_alpha); sin(delta\_alpha) cos(delta\_alpha)];

%delta\_t = t\_steps(i\_alpha) - t\_steps(i\_alpha-1);

p\_start\_rot = p\_end\_rot; %start with rotated orientation of last transformation

p\_end\_rot = R\_2dim \* p\_start\_rot;

p\_object\_start\_rot = p\_object\_end\_rot;

p\_object\_end\_rot = R\_2dim \* p\_object\_start\_rot;

%in translated coordinate system

p\_start = p\_start\_rot + origin + t\_steps(:,i\_alpha-1);

p\_end = p\_end\_rot + origin + t\_steps(:,i\_alpha);

%p\_object\_start = p\_object\_start\_rot + origin + t\_steps(:,i\_alpha-1);

p\_object\_end = p\_object\_end\_rot + origin + t\_steps(:,i\_alpha);

figure(fig1);

plot(p\_object\_end(1,:),p\_object\_end(2,:));

hold on;

plot(p\_end(1,:), p\_end(2,:), '+r');

axis equal;

hold on;

p\_start\_markers = p\_start(:,3:6);

p\_end\_markers = p\_end(:,3:6);

X\_trans = p\_start\_markers;

X = transpose(X\_trans);

Y\_trans = p\_end\_markers;

Y = transpose(Y\_trans);

[error,Xrot,transformation] = procrustes(X,Y,'scaling',false,'reflection',false);

I = eye(2);

A = transformation.T - I;

translation\_help = [transformation.c(1,1);transformation.c(1,2)];

translation = -(transformation.T\*translation\_help);

t\_vec = A\translation;

CoR\_x = -t\_vec(1); % minus, because new\_origin\_vector + t\_vec = 0!!

CoR\_y = -t\_vec(2);

%plot(CoR\_x, CoR\_y,'o');

hold on;

%figure(fig2);

%plot(CoR\_x, CoR\_y,'+');

%hold on;

t\_vec\_all(:,i\_alpha-1)=t\_vec;

%rotate and plot grid points movement around center of rotation

p\_samples\_start\_rot = p\_samples\_end\_rot;

p\_samples\_end\_rot = R\_2dim \* p\_samples\_start\_rot;

%in translated coordinate system

p\_samples\_start = p\_samples\_start\_rot + origin + t\_steps(:,i\_alpha-1);

p\_samples\_end = p\_samples\_end\_rot + origin + t\_steps(:,i\_alpha);

figure(fig3);

plot(p\_samples\_end(1,:),p\_samples\_end(2,:),'.');

axis equal;

hold on;

figure(fig4);

plot(p\_samples\_end(1,:),p\_samples\_end(2,:),'.');

axis equal;

hold on;

%arc length

deltaSArcLength = ((p\_samples\_end(1,:) - p\_samples\_start(1,:)).^2 + (p\_samples\_end(2,:) - p\_samples\_start(2,:)).^2).^0.5 ;

SArcLength = SArcLength + deltaSArcLength;

%store sample points

for ki=1:(n\_gridsamples\*n\_gridsamples)

p\_samples\_all(i\_alpha,1,ki) = p\_samples\_end(1,ki);

p\_samples\_all(i\_alpha,2,ki) = p\_samples\_end(2,ki);

end

figure(fig1);

plot(p\_samples\_end(1,:),p\_samples\_end(2,:),'.');

hold on;

end

%entire arc length will be plotted

figure(fig3);

labels = cellstr( num2str(SArcLength',3));

text(p\_samples\_end(1,:),p\_samples\_end(2,:), labels, 'VerticalAlignment','bottom', ...

'HorizontalAlignment','right', 'FontSize',6)

%maximum distance between point ono arc segment will be calcultaed and plotted

for ksample=1:(n\_gridsamples\*n\_gridsamples)

distance\_arc = 0; %reset value

for ki = 1:n\_steps\_alpha-1

for kj = ki:n\_steps\_alpha

distk= (p\_samples\_all(ki,1,ksample)-p\_samples\_all(kj,1,ksample)).^2 + ...

(p\_samples\_all(ki,2,ksample)-p\_samples\_all(kj,2,ksample)).^2;

if distk > distance\_arc

distance\_arc = distk;

end

end

end

DISTMAX\_ArcSegement(ksample) = sqrt(distance\_arc);

end

%entire arc length will be plotted

figure(fig4);

labels = cellstr( num2str(DISTMAX\_ArcSegement',3));

text(p\_samples\_end(1,:),p\_samples\_end(2,:), labels, 'VerticalAlignment','bottom', ...

'HorizontalAlignment','right', 'FontSize',6)

%additional rotation center of entire movement!

p\_start = object\_reference\_points\_origin; %start with initial poistion

p\_start\_markers = p\_start(:,3:6);

X\_trans = p\_start\_markers;

X = transpose(X\_trans);

Y\_trans = p\_end\_markers; %take last position from above

Y = transpose(Y\_trans);

[error,Xrot,transformation] = procrustes(X,Y,'scaling',false,'reflection',false);

I = eye(2);

A = transformation.T - I;

translation\_help = [transformation.c(1,1);transformation.c(1,2)];

translation = -(transformation.T\*translation\_help);

t\_vec = A\translation;

CoR\_x = -t\_vec(1); % minus, because new origin + t\_vec = 0!!

CoR\_y = -t\_vec(2);

figure(fig1);

plot(CoR\_x, CoR\_y,'ks','MarkerSize',12);

hold on;

%figure(fig2);

%plot(CoR\_x, CoR\_y,'ks','MarkerSize',12);

%hold on;

%collect more information about ICR

dist=0;

for k=1:n\_steps\_alpha-2

for l=k:n\_steps\_alpha-1

dist\_kl=sqrt((t\_vec\_all(1,k)-t\_vec\_all(1,l))^2 + (t\_vec\_all(2,k)-t\_vec\_all(2,l))^2);

if dist < dist\_kl

dist=dist\_kl;

end

end

end

distances\_ICR(i\_shift+1) = dist;

%rotation center is always set to (0,0) in the intern coordinate

%system-> therefore calculate distance to origin

distance\_CoR\_origin(i\_shift+1) = sqrt((CoR\_x - origin(1))^2 + (CoR\_y - origin(2))^2);

end