**Three-Dimensional Experimental Kinematics**

Digitize locations of landmarks  on body i for points k=1 to n at given time t

All points must be on same body i

Use landmark weighting factor  if point k is available at time t. Use  if point k not available at given time t .

Determine  at given time t .

**Mean values**















**Velocity**







Point ISA is on the instantaneous screw axis for body i at the root of the perpendicular from the centroid of the landmarks. Note that the ISA is not attached to the body. Any point P on the body coincident with the ISA has translational velocity  along the ISA.

 for any point P attached to body i





**Acceleration**









Point IAP is the instantaneous acceleration pole for body i. Note that the IAP is not attached to the body. Point P on the body coincident with IAP has zero acceleration.

 for any point P attached to body i



**Jerk**









Point IJP is the instantaneous jerk pole for body i. Note that the IJP is not attached to the body. Point P on the body coincident with IJP has zero jerk.

 for any point P attached to body i



**Second Order Screw Axis**









% t\_lm2kin3d.m - test 3D kinematics from landmark motion

% HJSIII, 14.01.14

clear

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% example data - CRSP

f = [ 1 1 1 1 1 1 1 1 ];

r = [ 11.000 9.000 9.000 11.000 11.000 9.000 9.000 11.000 ;

 1.000 1.000 -1.000 -1.000 1.000 1.000 -1.000 -1.000 ;

 10.000 10.000 10.000 10.000 8.000 8.000 8.000 8.000 ];

rd = [ -35.750 -35.750 -30.750 -30.750 -28.750 -28.750 -23.750 -23.750 ;

 27.500 22.500 22.500 27.500 27.500 22.500 22.500 27.500 ;

 8.000 1.000 1.000 8.000 8.000 1.000 1.000 8.000 ];

rdd = [ -45.250 -8.250 -19.250 -56.250 -58.250 -21.250 -32.250 -69.250 ;

 -233.000 -222.000 -209.500 -220.500 -198.000 -187.000 -174.500 -185.500 ;

 -130.375 -117.375 -117.375 -130.375 -105.875 -92.875 -92.875 -105.875 ];

rddd = [ 0 0 0 0 0 0 0 0 ;

 0 0 0 0 0 0 0 0 ;

 0 0 0 0 0 0 0 0 ];

vel\_test = [ 0 5.7703 0 ;

 -3.5000 -4.0203 14.4257 ;

 2.5000 3.3716 -10.3041 ];

accel\_test = [ 8.7500 2.6440 0 ;

 6.5000 -11.5414 0 ;

 -5.5000 3.7909 0 ];

jerk\_test = [ 1.5000 10.0000 0 ;

 -53.8125 0 0 ;

 61.5625 9.0000 0 ];

axode\_test = [ 0.1622 5.7703 0.3815 ;

 1.1824 0.3172 2.7819 ;

 1.6554 0.2734 3.8947 ];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% example data - RSUR

f = [ 1 1 1 1 1 1 ];

r = [ -9.8091 -6.7903 -4.2819 -4.7924 -7.8113 -10.3196 ;

 -15.4760 -8.0967 -3.7733 -6.8291 -14.2084 -18.5319 ;

 -1.9764 -0.7795 -7.0258 -14.4690 -15.6659 -9.4196 ];

rd = [ 1.8124 -0.4639 7.5799 17.9000 20.1763 12.1325 ;

 0.1709 0.3955 0.8591 1.0982 0.8737 0.4100 ;

 -14.0582 -9.7011 -6.1500 -6.9559 -11.3130 -14.8641 ];

rdd = [ -40.6381 -115.2233 -105.8619 -21.9154 52.6698 43.3084 ;

 15.3096 35.2375 76.6641 98.1628 78.2349 36.8083 ;

 -148.5442 -103.5193 -58.6743 -58.8541 -103.8790 -148.7241 ];

rddd = [ 596.0567 523.6492 4.6583 -441.9252 -369.5177 149.4732 ;

 3.0571 -41.6333 -59.9366 -33.5496 11.1408 29.4441 ;

 435.0585 -35.2544 -134.4616 236.6441 706.9570 806.1642 ];

vel\_test = [ 0.0383 0.4123 -0.0297 ;

 -1.3498 -10.4511 1.0468 ;

 0.0895 -0.9450 -0.0694 ];

accel\_test = [ 3.3113 -52.7472 0 ;

 -7.8355 81.8870 0 ;

 8.0368 -97.6836 0 ];

jerk\_test = [ -20.3604 -22.3536 0 ;

 61.6042 32.7874 0 ;

 1.5851 -12.8695 0 ];

axode\_test = [ -5.5403 0.6887 53.3304 ;

 -0.0060 -20.2001 0.0579 ;

 2.2768 -0.2983 -21.9164 ];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% test function

[ vel, accel, jerk, axode ] = lm2kin3d( f, r, rd, rdd, rddd )

% bottom of t\_lm2kin3d

function [ vel, accel, jerk, axode ] = lm2kin3d( f, r, rd, rdd, rddd )

% 3D kinematics of a rigid body from landmark motion

% HJSIII, 14.01.14

%

% USAGE

% [ vel, accel, jerk, axode ] = lm2kin3d( f, r, rd, rdd, rddd )

%

% INPUTS

% f - 1 x n vector of weights - f(j)=1 means data valid, f(j)=0 means data not available

% r - 3 x n matrix of x,y,z landmark location

% rd - 3 x n matrix of x,y,z landmark velocity

% rdd - 3 x n matrix of x,y,z landmark acceleration

% rddd - 3 x n matrix of x,y,z landmark jerk

%

% OUTPUTS

% vel = [ w\_vec rISA sISA ]

% w\_vec = 3x1 angular velocity vector

% rISA = 3x1 location on ISA at root of perpendicular from centroid of landmarks

% sISA = sliding velocity vector along ISA

%

% accel = [ wd\_vec rIAP rdd\_at\_IAP ]

% wd\_vec = 3x1 angular acceleration vector

% rIAP = 3x1 location of acceleration pole

% rdd\_at\_IAP = 3x1 acceleration of point on body at IAP

%

% jerk = [ wdd\_vec rIJP rddd\_at\_IJP ]

% wdd\_vec = 3x1 angular jerk vector

% rIJP = 3x1 location of jerk pole

% rddd\_at\_IJP = 3x1 jerk of point on body at IJP

%

% axode = [ OMEGA\_vec c Sd ]

% OMEGA\_vec = 3x1 rotation of second order screw

% c = 3x1 central point of generator

% Sd = 3x1 sliding velocity vector along second order screw

% constants

eps = 1e-14;

% number of coordinates and landmarks

[ ncoord, n ] = size( r );

% mean values

fmat = diag(f);

sf = trace( fmat );

rm = sum( fmat\*r' )' /sf;

rdm = sum( fmat\*rd' )' /sf;

rddm = sum( fmat\*rdd' )' /sf;

rdddm = sum( fmat\*rddd' )' /sf;

% centered location

rc = r - rm\*ones(1,n);

X = rc \* fmat \* rc' /sf;

M = trace(X) \* eye(ncoord) - X;

Minv = inv( M );

% velocity

V = rd \* fmat \* rc' /sf;

w\_vec = Minv \* [ V(3,2)-V(2,3) ; V(1,3)-V(3,1) ; V(2,1)-V(1,2) ];

w = norm( w\_vec );

w\_mat = skew\_sym( w\_vec );

% general velocity solution

if w > eps,

 u = w\_vec / w;

 sd = u' \* rdm;

 rISA = rm + w\_mat \* rdm / w^2;

 sISA = sd \* u;

% special case - w=0, pure translation

% rISA is at centroid of landmarks, sISA is translation velocity

else

 sd = norm( rdm );

 u = rdm / sd;

 rISA = rm;

 sISA = rdm;

end

% acceleration

A = rdd \* fmat \* rc' / sf;

B = A - w\_mat\*w\_mat \* X;

wd\_vec = Minv \* [ B(3,2)-B(2,3) ; B(1,3)-B(3,1) ; B(2,1)-B(1,2) ];

wd = norm( wd\_vec );

wd\_mat = skew\_sym( wd\_vec );

beta\_mat = wd\_mat + w\_mat\*w\_mat;

% general acceleration solution

if abs(det(beta\_mat)) > eps;

 rIAP = rm - inv(beta\_mat) \* rddm;

 rdd\_at\_IAP = zeros(ncoord,1);

else

% special case 1 - w=0, wd=0, pure translation

% rIAP is at centroid of landmarks, rdd\_at\_IAP is translation acceleration

 if w < eps,

 if wd < eps,

 sdd = norm( rddm );

 e = rddm / sdd;

 rIAP = rm;

 rdd\_at\_IAP = rddm;

% special case 2 - w=0, wd>0, pure angular acceleration

% similar to general angular velocity solution

% rIAP is at root of perpendicular to angular acceleration vector from centroid of landmarks

% rdd\_at\_IAP is translation acceleration

 else

 e = wd\_vec / wd;

 sdd = e' \* rddm;

 rIAP = rm + wd\_mat \* rddm / wd^2;

 rdd\_at\_IAP = sdd \* e;

 end

% special case 3 - w>0, wd=0, pure angular velocity

% similar to zero angular velocity solution

 else

 if wd < eps,

 e = u;

 sdd = e' \* rddm;

 rdd\_at\_IAP = sdd \* e;

 rIAP = rm + (rddm-rdd\_at\_IAP) / w\*w;

% special case 4 - w>0, wd>0, w\_vec parallel to wd\_vec

 else

 e = wd\_vec / wd;

 sdd = e' \* rddm;

 rdd\_at\_IAP = sd \* e;

 w2ia = w\*w\*eye(3) - wd\_mat;

 rIAP = rm + inv(w2ia) \* (rddm-rdd\_at\_IAP);

 end

 end

end

% jerk

J = rddd \* fmat \* rc' / sf;

eta\_mat\_mwdd = 2\*wd\_mat\*w\_mat + w\_mat\*wd\_mat + w\_mat\*w\_mat\*w\_mat;

H = J - eta\_mat\_mwdd \* X;

wdd\_vec = Minv \* [ H(3,2)-H(2,3) ; H(1,3)-H(3,1) ; H(2,1)-H(1,2) ];

wdd = norm( wdd\_vec );

h = wdd\_vec / wdd;

wdd\_mat = skew\_sym( wdd\_vec );

eta\_mat = eta\_mat\_mwdd + wdd\_mat;

rIJP = rm - inv(eta\_mat) \* rdddm;

rddd\_at\_IJP = zeros(ncoord,1);

% second order screw

OMEGA\_vec = w\_mat \* wd\_vec /w/w;

OMEGA = norm( OMEGA\_vec );

t = OMEGA\_vec / OMEGA;

d = t' \* beta\_mat \* (rIAP-rISA) / (t' \* beta\_mat \* u );

c = rISA + d \* u;

Sd = ( t' \* w\_mat \* beta\_mat \* (c-rIAP) /w/w ) - sd \* OMEGA /w;

Sd\_vec = Sd \* t;

% return arguments

vel = [ w\_vec rISA sISA ];

accel = [ wd\_vec rIAP rdd\_at\_IAP ];

jerk = [ wdd\_vec rIJP rddd\_at\_IJP ];

axode = [ OMEGA\_vec c Sd\_vec ];

% bottom of lm2kin3d