Finger Kinematics: Volar Bands

Version 2

Information

Author: Nathanael J. Rake
Date: 17 January 2017
Purpose: To compute the finger kinematics for an anthropomorphic, tendon driven, robotic finger.

Clear Variables

```
ClearAll["Global`*"];
```

Import

```
SetDirectory[NotebookDirectory[]];
<< "extraFunctions.m"
```

Denavit-Hartenburg Parameters

Joint Angles

```
θmpa;
θmpf;
θpip;
θdip;
θ5 = 0; (*Contact frame. Should always remain as 0*)
```

Link Offset

```
d1 = 0;
d2 = 0;
d3 = 0;
d4 = 0;
d5 = 0;
```

Link Lengths

```
a0 = 2.8750;
a1 = 0.5;
a2 = 1.75;
a3 = 1.25;
a4 = 1.25;
```

Link Twists

```
alpha0 = 0;
alpha1 = -90 Degree;
alpha2 = 0;
alpha3 = 0;
alpha4 = 0;
```

Homogeneous Transformation Matrices

Transform Function

Returns the transform from one frame to another based on the DH parameters

```
dhTransform[θ_, d_, a_, alpha_] :=
 ⎛      Cos[θ]             -Sin[θ]            0             a          ⎞
 ⎜Sin[θ] * Cos[alpha]  Cos[θ] * Cos[alpha]  -Sin[alpha]  -Sin[alpha] * d⎟
 ⎜Sin[θ] * Sin[alpha]  Cos[θ] * Sin[alpha]   Cos[alpha]   Cos[alpha] * d⎟
 ⎝        0                   0                0             1          ⎠ ;
```

Transformations: Consecutive Frames

From distal to proximal

```
T01[θmpa_] := dhTransform[θmpa, d1, a0, alpha0]; (*Transform from Frame 1 to Frame 0*)
T12[θmpf_] := dhTransform[θmpf, d2, a1, alpha1] // Evaluate;
T23[θpip_] := dhTransform[θpip, d3, a2, alpha2] // Evaluate;
T34[θdip_] := dhTransform[θdip, d4, a3, alpha3] // Evaluate;
T45 = dhTransform[θ5, d5, a4, alpha4];
```

From proximal to distal

```
T10[θmpa_] := (Inverse[T01[θmpa]] // Evaluate); (*Transform from Frame 0 to Frame 1*)
T21[θmpf_] := (Inverse[T12[θmpf]] // Evaluate);
T32[θpip_] := (Inverse[T23[θpip]] // Evaluate);
T43[θdip_] := (Inverse[T34[θdip]] // Evaluate);
T54 = Inverse[T45];
```

Transformations: Global Reference Frame

Transformations to the global reference frame

```
T02[θmpa_, θmpf_] := (T01[θmpa] . T12[θmpf] // Evaluate);
T03[θmpa_, θmpf_, θpip_] := (T02[θmpa, θmpf] . T23[θpip] // Evaluate);
T04[θmpa_, θmpf_, θpip_, θdip_] := (T03 [θmpa, θmpf, θpip]. T34[θdip] // Evaluate);
T05[θmpa_, θmpf_, θpip_, θdip_] := (T04[θmpa, θmpf, θpip, θdip] . T45 // Evaluate);
```

Transformations from the global reference frame

```
T20[θmpa_, θmpf_] := (T02[θmpa, θmpf] // Inverse // Evaluate);
T30[θmpa_, θmpf_, θpip_] := (T03[θmpa, θmpf, θpip] // Inverse // Evaluate);
T40[θmpa_, θmpf_, θpip_, θdip_] := (T04[θmpa, θmpf, θpip, θdip] // Inverse // Evaluate);
```

Transformations: Frame 2

```
T24[θpip_, θdip_] := (T23[θpip].T34[θdip] // Evaluate);
T25[θpip_, θdip_] := (T23[θpip].T34[θdip].T45 // Evaluate);
```

Other Finger Parameters

Joint Limits

Limits for the MPa joint

```
θmpaMin = -30 Degree;
θmpaMax = 30 Degree;
```

Limits for the MPf joint

```
θmpfMin = -45 Degree;
θmpfMax = 90 Degree;
```

Limits for the PIP joint

```
θpipMin = 0 Degree;
θpipMax = 90 Degree;
```

Limits for the DIP joint

```
θdipMin = -45 Degree;
θdipMax = 90 Degree;
```

Joint Pulley Radii

```
rMPf = 0.1875;
rPIP = 0.1875;
rDIP = 0.1875;
```

Lateral Band Origin

Location of the lateral band origin (relative to the chosen coordinate frame) when θpip is zero and the central slip is taught.
Only one assignment for originLocStart should be active (i.e., uncommented) at a time. Each definition allows a different measurement to be used.

```
(*specify x, the x-coordinate of the origin in frame 2*)
originLocStart = Module[{x},
   x = 0.7814;
   {x, (rMPf - rPIP) / a2 * x - rMPf, 0, 1}
  ];

(*
(*Specify h, total distance from top of PIP pulley to origin*)
originLocStart=Module[{h},
  h=0.25;
  {h*Cos[ArcTan[(rMPf-rPIP)/a2]],-rMPf+h*Sin[ArcTan[(rMPf-rPIP)/a2]],0,1}
 ];

(*Specify v, total distanec from top of DIP pulley to origin*)
originLocStart=Module[{v},
  v=a2-0.25;
  {a2-v*Cos[ArcTan[(rMPf-rPIP)/a2]],-rPIP-v*Sin[ArcTan[(rMPf-rPIP)/a2]],0,1}
 ];
*)
```

Location of the lateral band origin for a given θpip

```
originLoc[θpip_] := originLocStart + {rPIP * θpip * Cos[ArcTan[(rMPf - rPIP) / a2]],
    rPIP * θpip * Sin[ArcTan[(rMPf - rPIP) / a2]], 0, 0};
```

Lateral Band Union

Location of the lateral band union when θdip is zero and the terminal slip is taught.
Only one assignment for unionLocStart should be active (i.e., uncommented) at a time. Each definition allows a different measurement to be used.

```
(*specify x, the x-coordinate of the origin in frame 3*)
unionLocStart = Module[{x},
   x = 0.5118;
   {x, (rPIP - rDIP) / a3 * x - rPIP, 0, 1}
  ];

(*
(*Specify h, total distance from top of DIP pulley to union*)
originLocStart=Module[{h},
   h=0.25;
   {h*Cos[ArcTan[(rPIP-rDIP)/a3]],-rPIP+h*Sin[ArcTan[(rPIP-rDIP)/a3]],0,1}
  ];
*)
```

Location of the lateral band union for a given θdip

```
unionLoc[θdip_] := unionLocStart + {rDIP * θdip * Cos[ArcTan[(rPIP - rDIP) / a3]],
    rDIP * θdip * Sin[ArcTan[(rPIP - rDIP) / a3]], 0, 0};
```

Mechanical Stop

The mechanical stop is used to deflect the lateral bands as they move around the PIP joint. This notebook assumes that a mechanical stop is a cylinder that passes symmetrically through the width of the finger.

Is there a mechanical stop?

```
mechStopPresent = True;
```

Do the lateral bands pass over the “distal” or “volar” side of the mechanical stop?
If there is no mechanical stop, this option does not matter.

```
mechStopSide = "volar";
```

In which frame is the stop anchored?

```
mechStopFrame = 2;
```

What are the coordinates of the center of the mechanical stop in this frame?

```
mechStopLoc = {a2, 0, 0, 1};
```

Radius of the mechanical stop;

```
mechStopRad = 0.125 / 2;
```

Lateral Band Length

Specify a pose that is known to be included in the joint coupling. This will be used to set the length of the lateral band.
knownPose has the form {θpip,θdip}

```
knownPose = {0 Degree, 0 Degree};
```

This method places the finger in a specified pose and calculates the locations of the lateral band origin and lateral band union. The path that the lateral bands take from the origin to the union (including effects introduced by a mechanical stop) is then calculated. The length of this path, which is also the length of the lateral band, is then found
See research notes from 6 September 2016

```
findLateralBandLength[θpip_, θdip_] := Module[{p1, p2, length, c, q1a, q1b, q2a,
    q2b, q1, q2, onStop, lengthA, lengthB, ang1, ang2, angleWrapped, lengthC},
   p1 = originLoc[θpip][[1 ;; 3]];
   p2 = (T23[θpip].unionLoc[θdip])[[1 ;; 3]];

   If[mechStopPresent == False,
    length = EuclideanDistance[p1, p2]
    ,
    If[mechStopFrame == 2,
     c = mechStopLoc[[1 ;; 3]],
     c = (T23[θpip].mechStopLoc)[[1 ;; 3]]
    ];

    {q1a, q1b} = pointsTangent[p1[[1 ;; 2]], c[[1 ;; 2]], mechStopRad];
    {q2a, q2b} = pointsTangent[p2[[1 ;; 2]], c[[1 ;; 2]], mechStopRad];
    q1a = Join[q1a, {0}];
    q1b = Join[q1b, {0}];
    q2a = Join[q2a, {0}];
    q2b = Join[q2b, {0}];

    q1 = "error";
    If[mechStopSide === "dorsal", (*If the tendon passes on the dorsal side*)
     If[ ((q1a - p1) × (q1b - p1))[[3]] > 0,
      q1 = q1a,
      q1 = q1b
     ]
    ,
     If[ ((q1a - p1) × (q1b - p1))[[3]] < 0,
      (*Else, if the tendon passes on the volar side*)
      q1 = q1a,
      q1 = q1b
     ]
    ];

    q2 = "error";
    If[mechStopSide === "dorsal",
     If[ ((q2a - p2) × (q2b - p2))[[3]] < 0, (*If the tendon passes on the dorsal side*)
      q2 = q2a,
      q2 = q2b
     ]
    ,
     If[ ((q2a - p2) × (q2b - p2))[[3]] > 0,
      (*Else, if the tendon passes on the volar side*)
```

```
    q2 = q2a,
    q2 = q2b
   ]
  ];

  onStop = "error";
  If[mechStopSide === "dorsal",
   If[((p2 - p1) × (q1 - p1))[[3]] > 0, (*If tendon on dorsal side*)
    onStop = False,
    onStop = True;
   ]
   ,
   If[((p2 - p1) × (q1 - p1))[[3]] < 0, (*If on volar side*)
    onStop = False,
    onStop = True;
   ]
  ];

  length = "error";

  If[onStop == False,
   length = EuclideanDistance[p1, p2]
  ];

  If[onStop == True,
   lengthA = EuclideanDistance[p1, q1];
   lengthB = EuclideanDistance[q2, p2];
   ang1 = ArcTan[q1[[1]] - c[[1]], q1[[2]] - c[[2]]];
   While[ang1 < 0, ang1 = ang1 + 2 * π];
   ang2 = ArcTan[q2[[1]] - c[[1]], q2[[2]] - c[[2]]];
   While[ang2 < 0, ang2 = ang2 + 2 * π];
   angleWrapped = Abs[ang1 - ang2];
   lengthC = mechStopRad * angleWrapped;
   length = lengthA + lengthB + lengthC;
  ];

  length
 ];
 length
];
```

Length of lateral bands. This is found by calculating the length of the lateral band in the specified known pose.

```
lateralBandLength = findLateralBandLength @@ knownPose;
```

Flexor Pins

Metacarpal pins

```
metaFlexorPinProx = {0.0625, 0, -0.3070, 1};
metaFlexorPinDist = {2.6708, 0, -0.3070, 1};
```

Transitional piece pins

```
(*There are no pins on the transitional piece*)
```

Proximal phalanx pins

```
proxFlexorPinProx = {0.3700, 0.3125, 0, 1};
proxFlexorPinDist = {1.3800, 0.3125, 0, 1};
```

Middle phalanx pins

```
midFlexorPinProx = {0.3700, 0.3125, 0, 1};
midFlexorPinDist = {0.8800, 0.3125, 0, 1};
```

Distal phalanx pin

```
distFlexorPinProx = {0.3750, 0.3125, 0, 1};
(*There is no distal flexor pin on the distal phalanx*)
```

Interossei Routings

Pulley Center (given in frame 1)

Becuase the interossei are symmetric about the finger's sagittal plane when they cross the routing pulley, we do not need to consider them individually. Instead, we may consider them as a single, virtual interosseus that lies in the sagittal plane of the finger. As a result, the location of the routing pulley *intPulleyLoc* only considers the x and y-coordinates. The z-coordinate is set to zero so that the pulley also lies within the sagittal plane.

```
intPulleyLoc = T12[mpf][[ ;; , 4]]; (*We can use this
  (instead of just coordinates) becuause the pulley is coaxial with the mpf axis*)
```

Pulley radius

```
intPulleyRad = (3 / 8) / 2;
```

Interosseus A (+z axis of frame 2)

```
intARoutingProx = {2.6708, 0.4688, -0.1875, 1}; (*In frame 0*)
intARoutingDist = {0.1005, 0.4688, -0.1875, 1}; (*In frame 1*)
```

Interosseus B (-z axis of frame 2)

```
intBRoutingProx = {2.6708, -0.4688, -0.1875, 1}; (*In frame 0*)
intBRoutingDist = {0.1005, -0.4688, -0.1875, 1}; (*In frame 1*)
```

Tendon Lengths

Flexor

```
flexorLength[θmpa_, θmpf_, θpip_, θdip_] :=
  Module[{pin1, pin2, pin3, pin4, pin5, pin6, pin7, length},
   pin1 = metaFlexorPinProx;
   (*This pin doesn't really need to be included because it is the most
     proximal pin and the distance between it and pin2 does not change;*)
   pin2 = metaFlexorPinDist;
   pin3 = T02[θmpa, θmpf].proxFlexorPinProx;
   pin4 = T02[θmpa, θmpf].proxFlexorPinDist;
   pin5 = T03[θmpa, θmpf, θpip].midFlexorPinProx;
   pin6 = T03[θmpa, θmpf, θpip].midFlexorPinDist;
   pin7 = T04[θmpa, θmpf, θpip, θdip].distFlexorPinProx;

   length = EuclideanDistance[pin2, pin3] +
     EuclideanDistance[pin4, pin5] + EuclideanDistance[pin6, pin7];
   Return[length]
  ];
```

Extensor

```
extensorLength[θmpf_] := Module[{length},
   length = θmpf * rMPf;
   Return[length]
  ];
```

Interossei

Interosseus A (+z axis of frame 2)

```
interosseusALength[θmpa_, θmpf_, θpip_] := Module[{seg1, p1, p2, c, q1a, q1b, q2a, q2b,
    q1, q2, lengthA, lengthB, ang1, ang2, angleWrapped, lengthWrapped, seg2},
   seg1 = EuclideanDistance[intARoutingProx, T01[θmpa].intARoutingDist];

   p1 = {intARoutingDist[[1]], 0, intARoutingDist[[3]]};
   p2 = (T12[θmpf].originLoc[θpip])[[1 ;; 3]];
   c = intPulleyLoc[[1 ;; 3]];

   {q1a, q1b} = pointsTangent[p1[[{1, 3}]], c[[{1, 3}]], intPulleyRad];
   {q2a, q2b} = pointsTangent[p2[[{1, 3}]], c[[{1, 3}]], intPulleyRad];
   q1a = {q1a[[1]], 0, q1a[[2]]};
   q1b = {q1b[[1]], 0, q1b[[2]]};
   q2a = {q2a[[1]], 0, q2a[[2]]};
   q2b = {q2b[[1]], 0, q2b[[2]]};

   If[ ((q1a - p1) × (q1b - p1))[[2]] < 0, (*If q1a is volar*)
    q1 = q1a, (*make it the q1*)
    q1 = q1b (*else, q1b is volar so make it the q1*)
   ];

   If[ ((q2a - p2) × (q2b - p2))[[2]] > 0, (*If q2a is volar*)
    q2 = q2a, (*make it the q2*)
    q2 = q2b (*else, q2b is volar so make it the q2*)
   ];

   lengthA = EuclideanDistance[p1, q1];
   lengthB = EuclideanDistance[q2, p2];
   ang1 = ArcTan[q1[[1]] - c[[1]], q1[[3]] - c[[3]]];
   While[ang1 < 0, ang1 = ang1 + 2 * π];
   ang2 = ArcTan[q2[[1]] - c[[1]], q2[[3]] - c[[3]]];
   While[ang2 < 0, ang2 = ang2 + 2 * π];
   angleWrapped = ang2 - ang1;
   lengthWrapped = intPulleyRad * angleWrapped;
   seg2 = lengthA + lengthB + lengthWrapped + 2 π * intPulleyRad;

   Return[seg1 + seg2]
  ];
```

Interosseus B (-z axis of frame 2)

```
interosseusBLength[θmpa_, θmpf_, θpip_] := Module[{seg1, p1, p2, c, q1a, q1b, q2a, q2b,
    q1, q2, lengthA, lengthB, ang1, ang2, angleWrapped, lengthWrapped, seg2},
   seg1 = EuclideanDistance[intBRoutingProx, T01[θmpa].intBRoutingDist];

   p1 = {intBRoutingDist[[1]], 0, intBRoutingDist[[3]]};
   p2 = (T12[θmpf].originLoc[θpip])[[1 ;; 3]];
   c = intPulleyLoc[[1 ;; 3]];

   {q1a, q1b} = pointsTangent[p1[[{1, 3}]], c[[{1, 3}]], intPulleyRad];
   {q2a, q2b} = pointsTangent[p2[[{1, 3}]], c[[{1, 3}]], intPulleyRad];
   q1a = {q1a[[1]], 0, q1a[[2]]};
   q1b = {q1b[[1]], 0, q1b[[2]]};
   q2a = {q2a[[1]], 0, q2a[[2]]};
   q2b = {q2b[[1]], 0, q2b[[2]]};

   If[ ((q1a - p1) × (q1b - p1))[[2]] < 0, (*If q1a is volar*)
    q1 = q1a, (*make it the q1*)
    q1 = q1b (*else, q1b is volar so make it the q1*)
   ];

   If[ ((q2a - p2) × (q2b - p2))[[2]] > 0, (*If q2a is volar*)
    q2 = q2a, (*make it the q2*)
    q2 = q2b (*else, q2b is volar so make it the q2*)
   ];

   lengthA = EuclideanDistance[p1, q1];
   lengthB = EuclideanDistance[q2, p2];
   ang1 = ArcTan[q1[[1]] - c[[1]], q1[[3]] - c[[3]]];
   While[ang1 < 0, ang1 = ang1 + 2 * π];
   ang2 = ArcTan[q2[[1]] - c[[1]], q2[[3]] - c[[3]]];
   While[ang2 < 0, ang2 = ang2 + 2 * π];
   angleWrapped = ang2 - ang1;
   lengthWrapped = intPulleyRad * angleWrapped;
   seg2 = lengthA + lengthB + lengthWrapped + 2 π * intPulleyRad;

   Return[seg1 + seg2]
  ];
```

Joint Coupling

Method to find the joint coupling for a single PIP angle

```
findJointCoupling[θpip_, lateralBandLengthTolerance_: 0.00001, iterationMax_: 100] :=
  Module[{lower, upper, point, len, dif},
   i = 0;

   lower = θdipMin;
   upper = θdipMax;
   point = (lower + upper) / 2;

   len = findLateralBandLength[θpip, point];
   dif = lateralBandLength - len;

   While[(Abs[dif] > lateralBandLengthTolerance) && (i < iterationMax),
    If[dif > 0, (*band is too short*)
     lower = point, (*flex the DIP to lengthen the band*)
     upper = point (*else, extend the DIP to shorten the band*)
    ];
    point = (lower + upper) / 2;
    len = findLateralBandLength[θpip, point];
    dif = lateralBandLength - len;
    i++;
   ];

   If[Abs[dif] > lateralBandLengthTolerance,
    If[dif > 0,
      point = θdipMax,
      point = "error"
     ];
   ];

   Return[point]
  ];
```