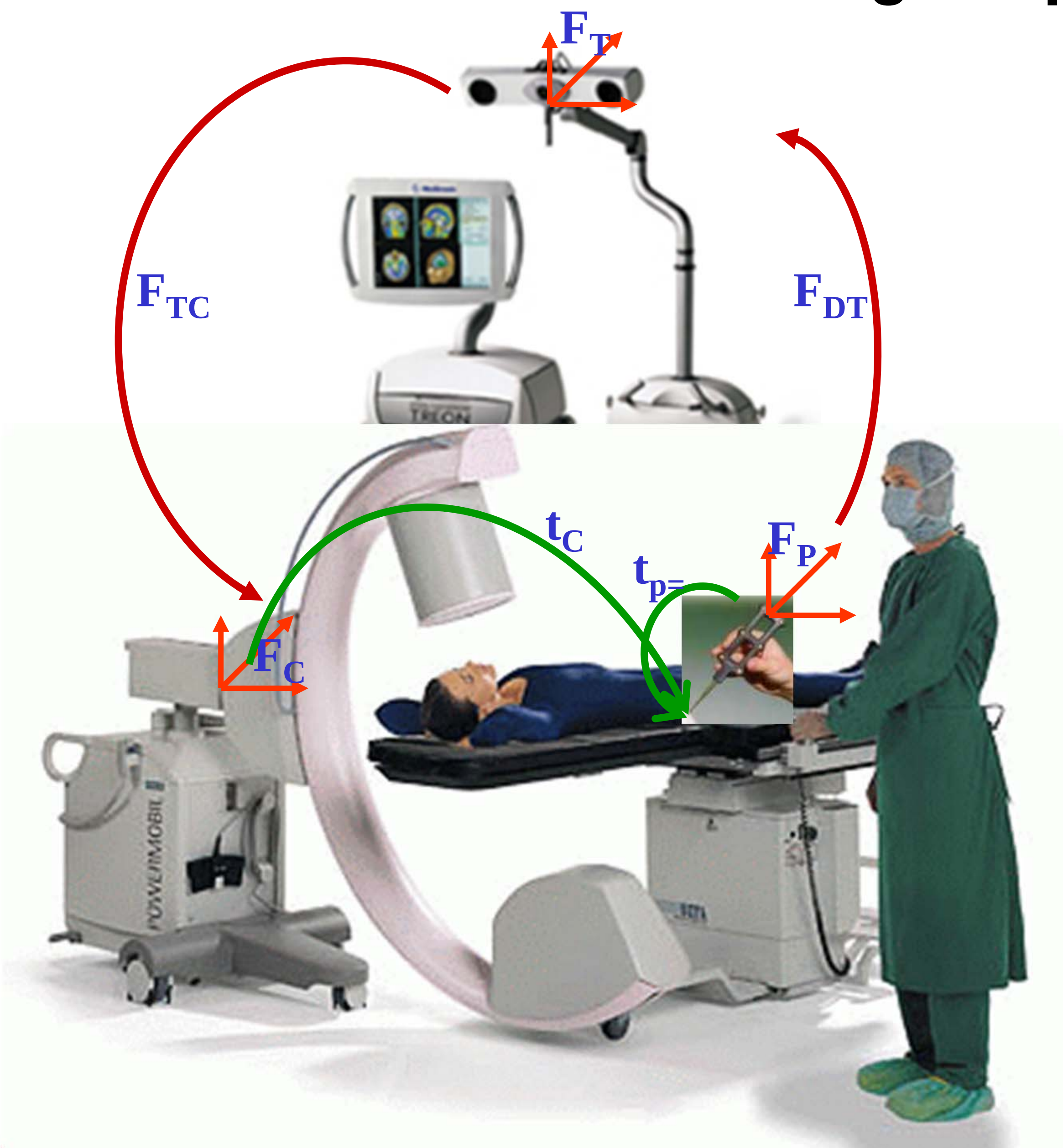
Tracked Tool Calibration

Computing hidden geometrical properties from position measurements

Tracked surgical pointer

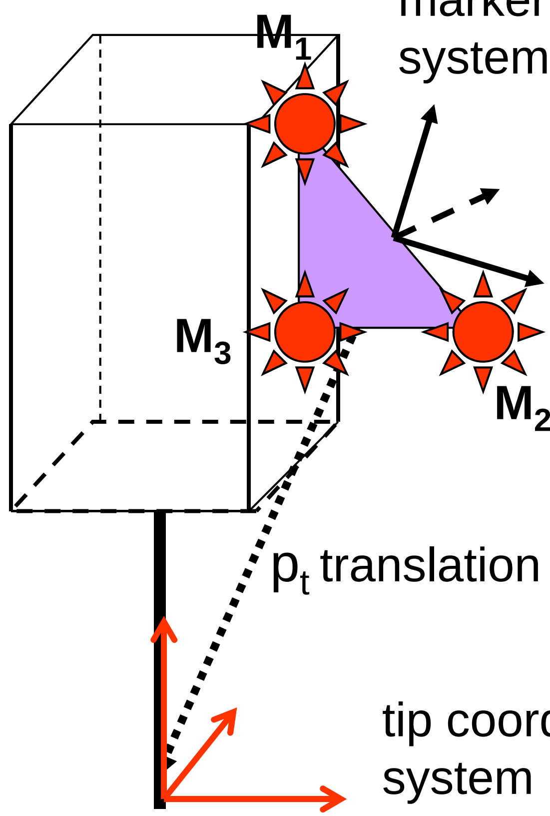


$p_{C=}\,(F_{TC}(F_{DT}\,t_p))$

t_p ???

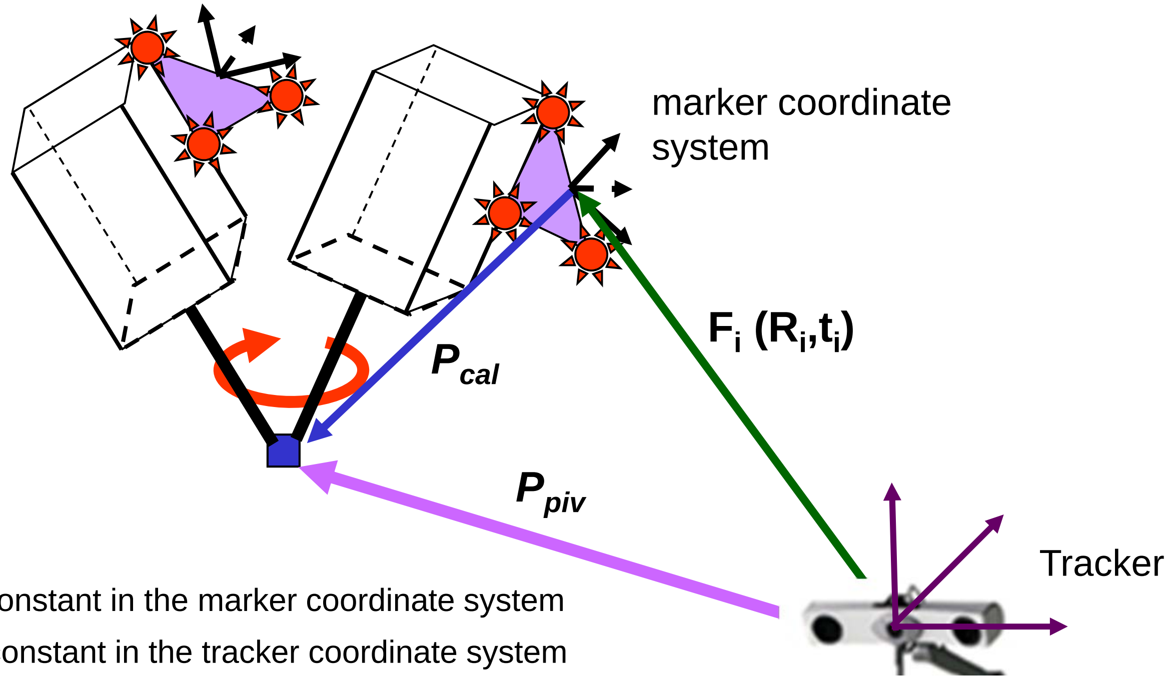
- Must be known prior to using the surgical tool
- Constant during the procedure
- We need a calibration session before surgery

Pivot calibration



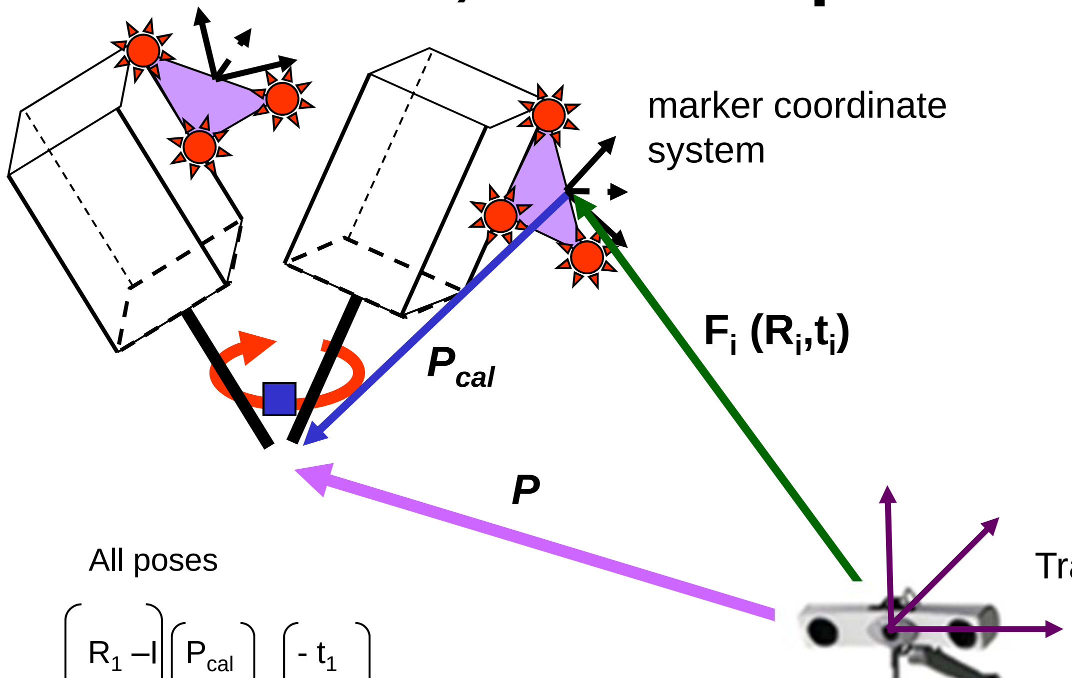
- Determine p_t translation between tip and marker coordinate system
- Create a geometrical constraint -- pivot around a fixed point

Pivot calibration



- P_{cal} vector is constant in the marker coordinate system
- Pivot point is constant in the tracker coordinate system
- M_1,M_2,M_3 are reported by the tracker in both poses
- Determine the marker coordinate system (orthonormal base from 3 markers) in both poses
- Calculate the F_i (R_i,t_i) frame transformation between marker and tracker frames, for both poses
- F_i (R_i,t_i) takes the P_{cal} vector to the pivot point P_{piv}
- $F_i*P_{cal} = P_{piv}$
- First rotation by R_i, then translation by t_i
- $R_i*P_{cal} + t_i = P_{piv}$
- Unknowns: P_{cal} and P_{piv}
- Use many poses to calculate $P_{cal...}$

Pivot calibration, method: pseudo-inverse



i^{th} pose

$R_i * P_{cal} + t_i = P_{piv}$

$$\begin{pmatrix} R_i & -I \end{pmatrix} \begin{pmatrix} P_{cal} \\ P_{piv} \end{pmatrix} = -t_i$$

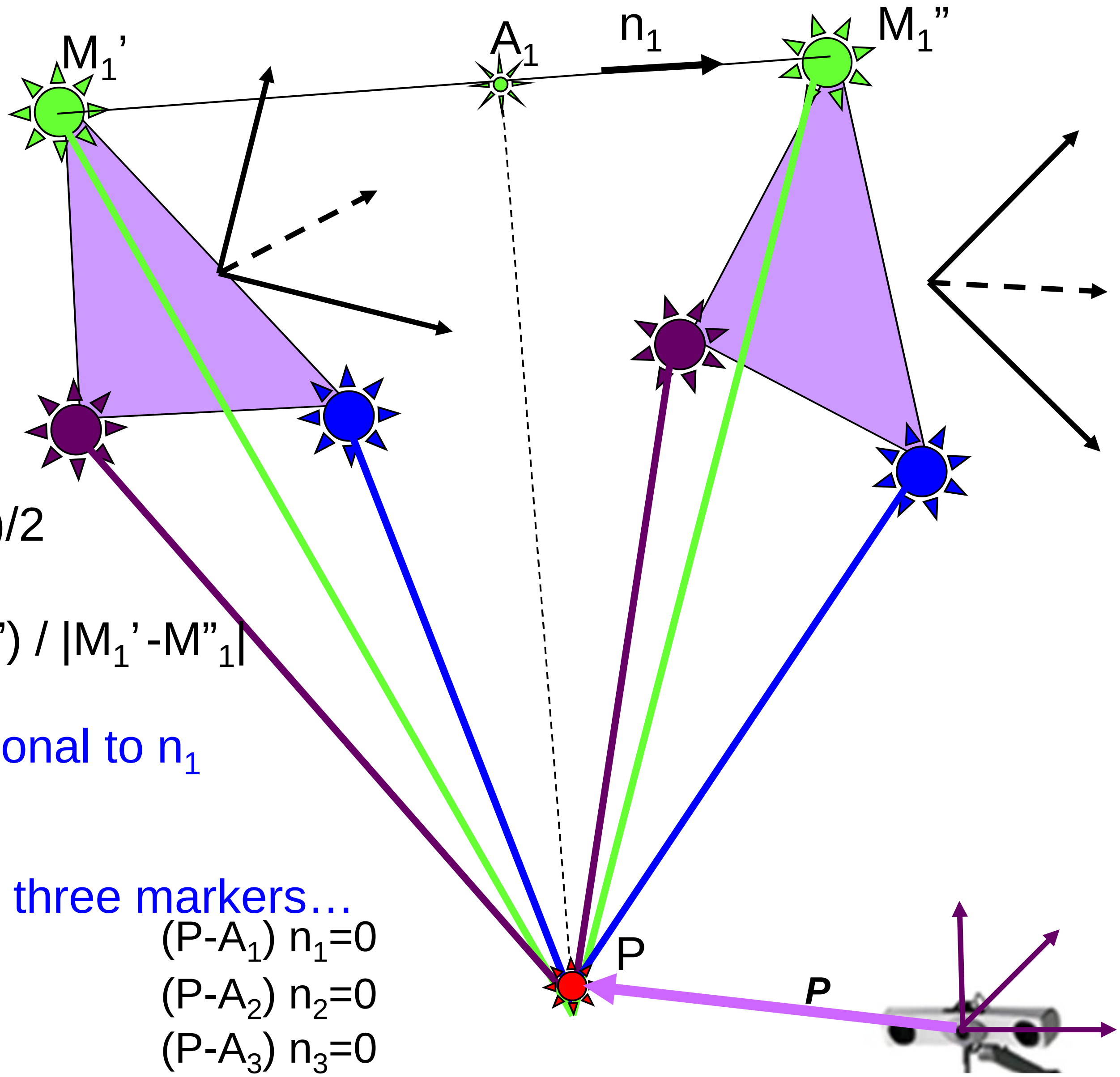
All poses

$$\begin{pmatrix} R_1 & -I \\ R_1 & -I \\ R_i & -I \\ R_n & -I \end{pmatrix} \begin{pmatrix} P_{cal} \\ P_{piv} \end{pmatrix} = \begin{pmatrix} -t_1 \\ -t_2 \\ -t_i \\ -t_n \end{pmatrix}$$

Ax=b matrix equation
Solve with pseudo-inverse
(Optimal in least square sense)

Pros: uses many poses, optimal in least square sense
Cons: need MATLAB or something to solve pseudo-inverse
Remember to check & remove outliers.

Pivot calibration, method: 3-isosceles



$A_1 = (M_1' + M_1'')/2$

$n_1 = (M_1' - M_1'') / |M_1' - M''_1|$

PA$_1$ is orthogonal to n_1

$(P - A_1)\, n_1 = 0$

Same for all three markers…

$(P - A_1)\, n_1 = 0$

$(P - A_2)\, n_2 = 0$

$(P - A_3)\, n_3 = 0$

Pivot calibration, method: 3-isosceles (cont'd)

$(P-A_1)\, n_1=0$
$(P-A_2)\, n_2=0$
$(P-A_3)\, n_3=0$

$P\, n_1=A_1 n_1$ $\quad$ $n1= [x1\ \ y1\ \ z1]$
$P\, n_2=A_2 n_2$ $\quad$ $n2= [x2\ \ y2\ \ z2]$
$P\, n_3=A_3 n_3$ $\quad$ $n3= [x3\ \ y3\ \ z3]$
$P\ = [x\quad y\quad z\,]$

$$\begin{vmatrix} x1 & y1 & z1 \\ x2 & y2 & z2 \\ x3 & y3 & z3 \end{vmatrix} * \begin{vmatrix} x \\ y \\ z \end{vmatrix} = \begin{vmatrix} A_1 n_1 \\ A_2 n_2 \\ A_3 n_3 \end{vmatrix}$$

$$\boxed{\begin{vmatrix} x1 & y1 & z1 \\ x2 & y2 & z2 \\ x3 & y3 & z3 \end{vmatrix}^{-1} \begin{vmatrix} A_1 n_1 \\ A_2 n_2 \\ A_3 n_3 \end{vmatrix} = \begin{vmatrix} x \\ y \\ z \end{vmatrix} = P}$$

Pros: Simple
Cons: sensitive to error because of just two poses.
Fix: Use many poses, calibrate with each pair, and then average out **$P_{cal.}$**

Remember to check & remove outliers.

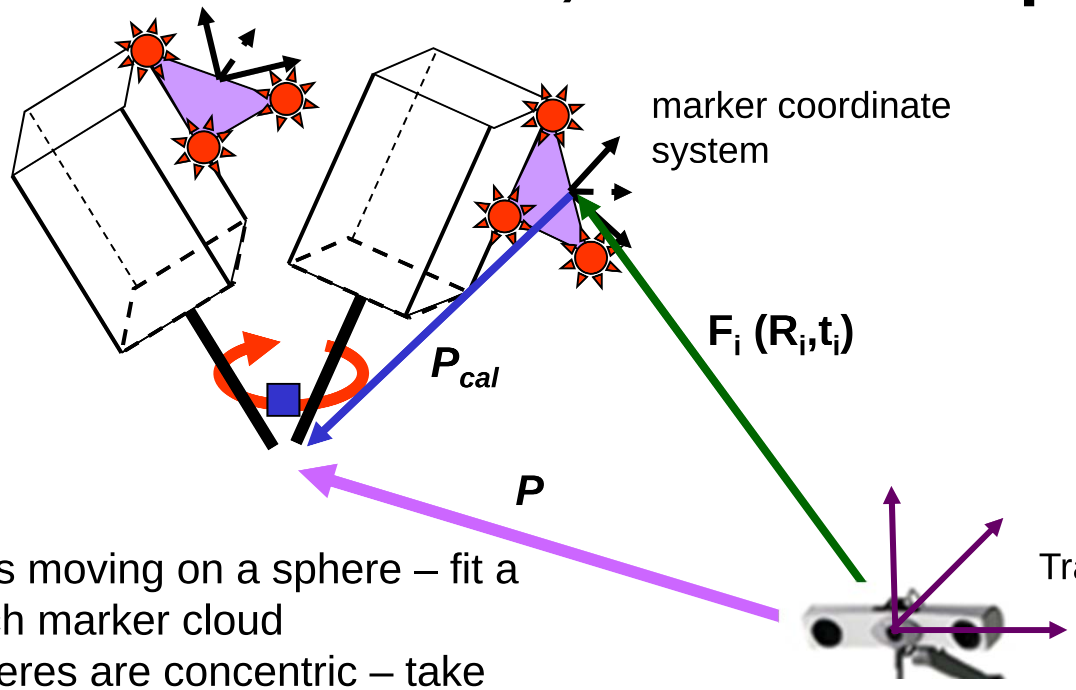
Still must solve for p_{cal}

$R_i{*}p_{cal} + t_i = p$

$$\mathbf{p_{cal} = R_i^{-1}(p - t_i)}$$

Note: no solution if $n_{1,}\ n_2\ n_3$ are all in one plane (i.e. $n_{1,}\ n_2\ n_3$ must be linearly independent !!

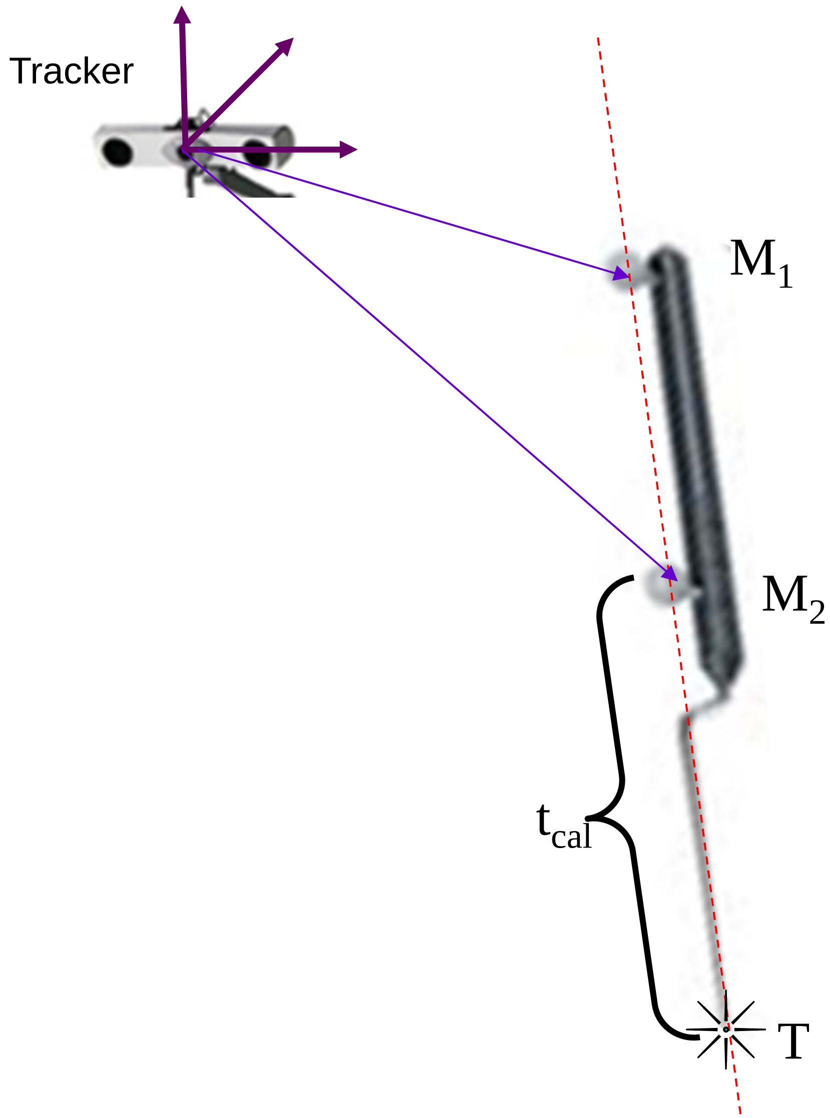
Pivot calibration, method: 3 spheres



Each marker is moving on a sphere – fit a sphere on each marker cloud
The three spheres are concentric – take average to get the pivot point on tracker frame.
Transform pivot point to marker frame for each pose and average out.
Remember to check & remove outliers.

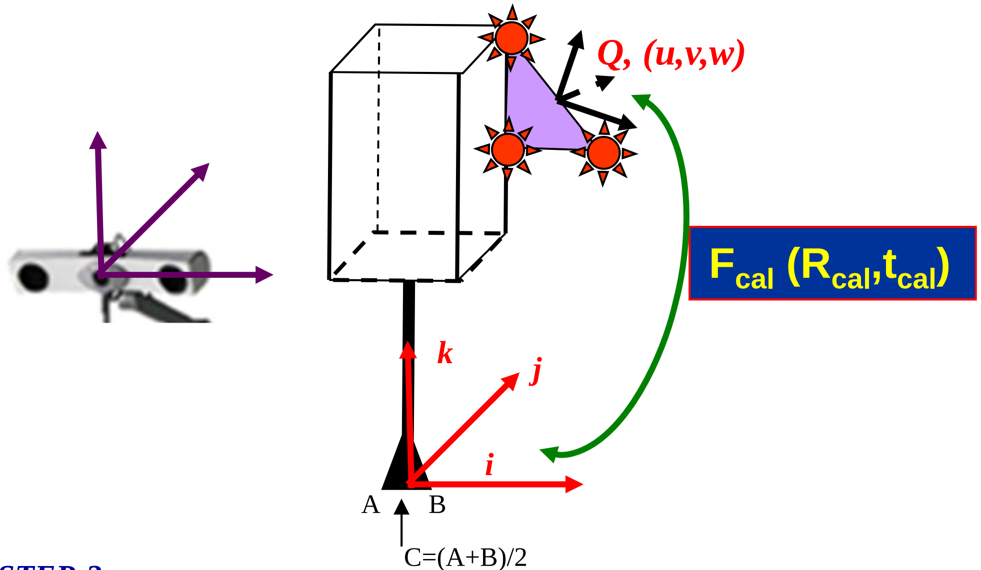
Pros: uses many poses, optimal in least square sense because spheres were fitted w/ least square
Cons: Need MATLAB or something to solve sphere fitting.

Example: calibration of a 2-marker pointer



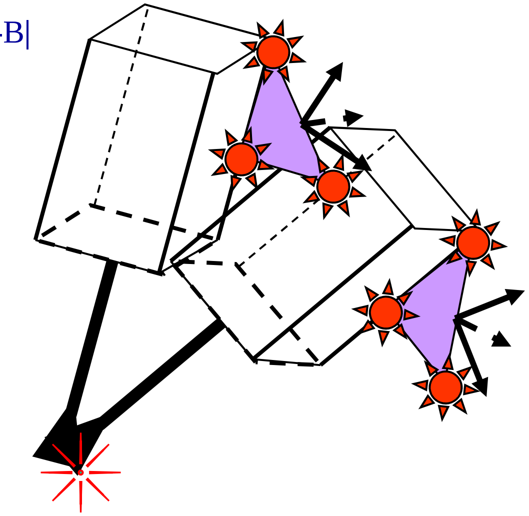
- Calibration: determine the unknown ‘t_{cal}’
- M1, M2 and the tool tip must be colinear, else the tool tip could not be determined from measuring (M1,M2)
- Calibration means determining the unknown ‘t_{cal}’
- Pivot calibration solves for T as the approximate intersection of two pivot lines in space. Then we calculate an approximate ‘t_{cal}’.
- Two poses are enough for calibration
- Multiple poses produces a “stronger” calibration
- Slip of the tool tip during pivoting may be caught as larger distance between two pivot line, but even this is totally fool proof, however.
- Our best is to pivot in great many directions

Example: calibration of a tracked chisel



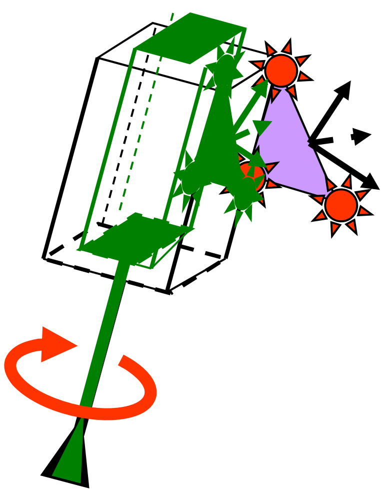
STEP-1

- Pivot about A corner of the edge
- Pivot about B corner of the edge
- C=(A+B)/2
- ***i***=(A-B)/|A-B|



STEP-2

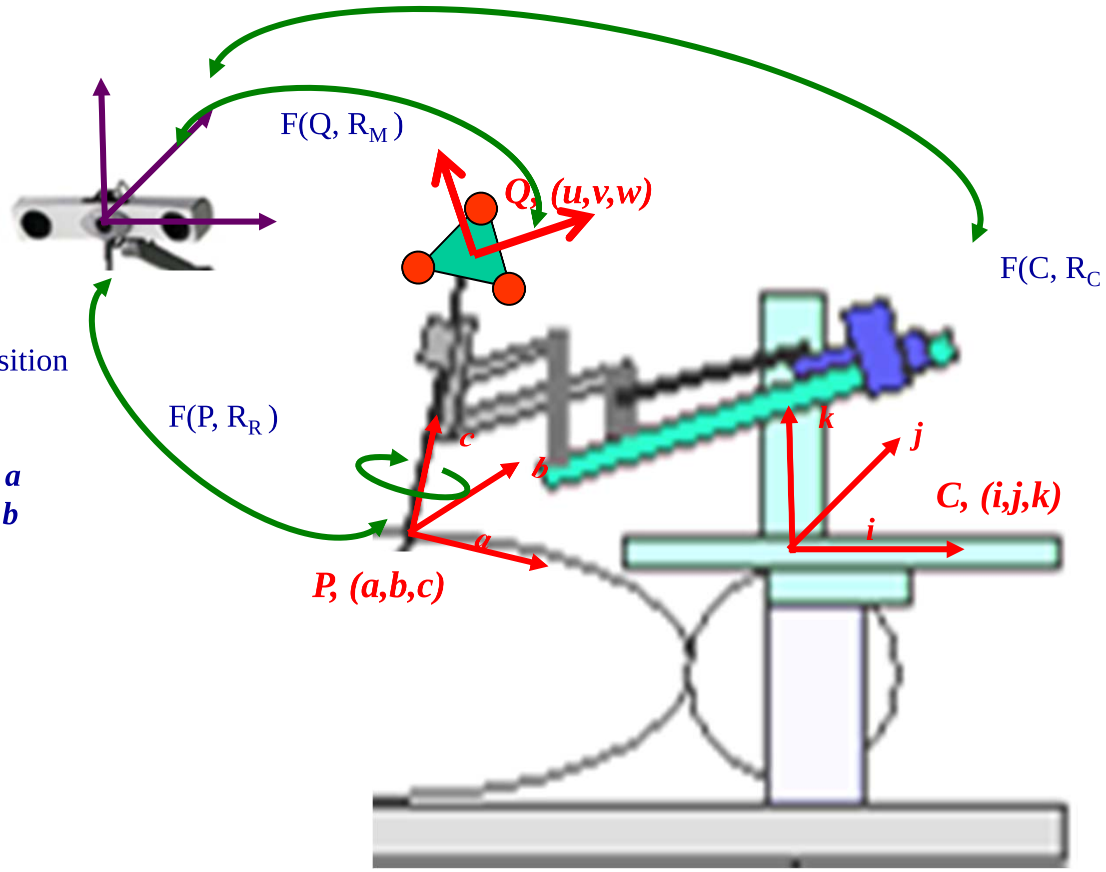
- Rotate about ***k*** axis inside a steady guide sleeve, in 3 poses
- M1 travels on a circle, calculate the center point C1
- M2 travels on a circle, calculate the center point C2
- ***k***=(C1-C2)/ |C1-C2|



STEP-3

- ***j=k x i***
- ***R_{cal} is know from (i,j,k) and (u,v,k)***
- ***t_{cal} known as t_{cal}=(C-Q)***

Example: Tracked 2x5DOF RCM-CART robot



RCM stage calibration

- Bring α and β joints to 'home' (zero) position
- Pivot needle about P → get
- Rotate about needle axis → get ***c*** *axis*
- Rotate RCM robot about α axis 1 → get ***a***
- Rotate RCM robot about β axis 2 → get ***b***
- From (abc) get R_R
- Now we have F(P, R_R)

Cartesian stage calibration

- Bring all joints to 'home' (zero) position
- Move Cartesian stage #1 → *get* ***i***
- Move Cartesian stage #2 → *get* ***j***
- Move Cartesian stage #3 → *get* ***k***
- From (ijk) get R_C
- C is irrelevant

Important questions during surgery – called "Inverse Kinematics"

1. How to adjust the Cartesian robot to move needle tip from P1T to P2T (given in tracker space)? – transform v1 and v2 to Cartesian robot space and figure out translation (C will fall out of the equation!)

2. How to adjust the RCM robot to rotate needle axis from v1 direction to v2 direction (given in tracker space)? – transform v1 and v2 to RCM space and figure out rotations