Chapter 2
Math Fundamentals

Part 4

2.7 Transform Graphs and Pose Networks

Carnegie Mellon
THE ROBOTICS INSTITUTE

Outline

- 2.7.1 Transforms as Relationships
- 2.7.2 Solving Pose Networks
- 2.7.3 Overconstrained Networks
- 2.7.4 Differential Kin of Frames in General Position
- Summary

Outline

- <u>2.7.1 Transforms as Relationships</u>
- 2.7.2 Solving Pose Networks
- 2.7.3 Overconstrained Networks
- 2.7.4 Differential Kin of Frames in General Position
- Summary

Spatial Relationships

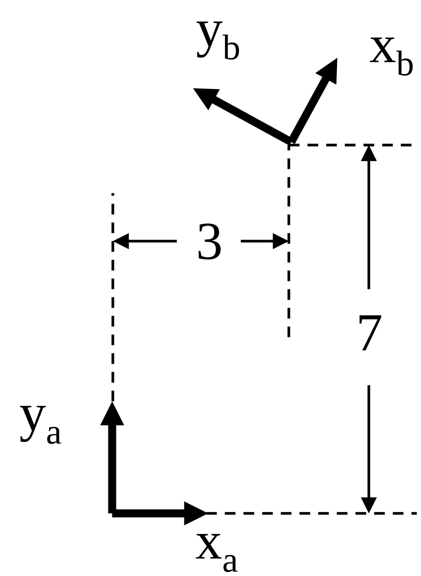
- Orthogonal Transforms represent a spatial relationship.
- The property of "being positioned 3 units to the right, 7 units above, and rotated 60 degrees with respect to something else."

$$\mathrm{Rot}\left(\frac{\pi}{3}\right)\mathrm{Trans}(3,7)$$

$$T = \begin{bmatrix} c\frac{\pi}{3} & -s\frac{\pi}{3} & 3 \\ s\frac{\pi}{3} & c\frac{\pi}{3} & 7 \\ 0 & 0 & 1 \end{bmatrix}$$

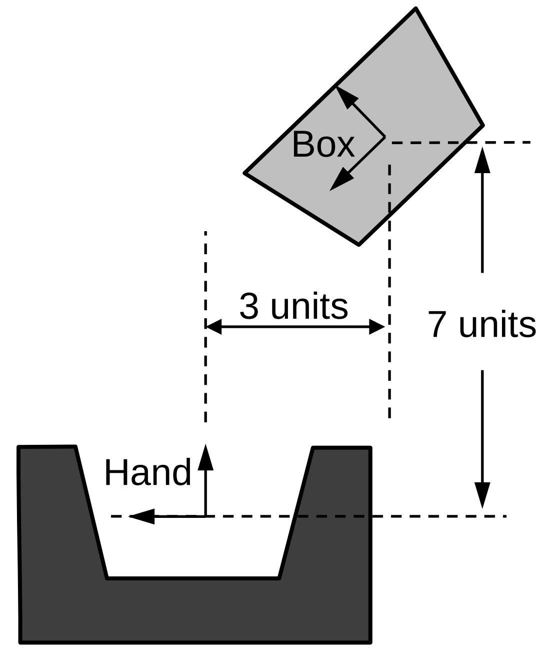
Abstract Relationships

- Relationships are abstract.
- We may visualize this best by drawing two arbitrary frames.



Instantiated Relationships

- We may also "instantiate" the relationship.
  - Indicate two objects which have the relationship.

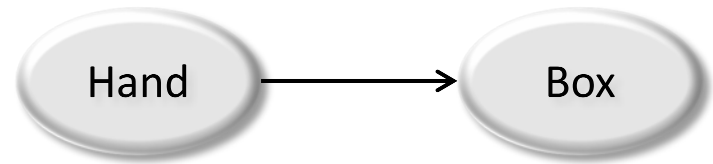


$$\boxed{T^{hand}_{box}}$$

Carnegie Mellon THE ROBOTICS INSTITUTE

Graphical Representation

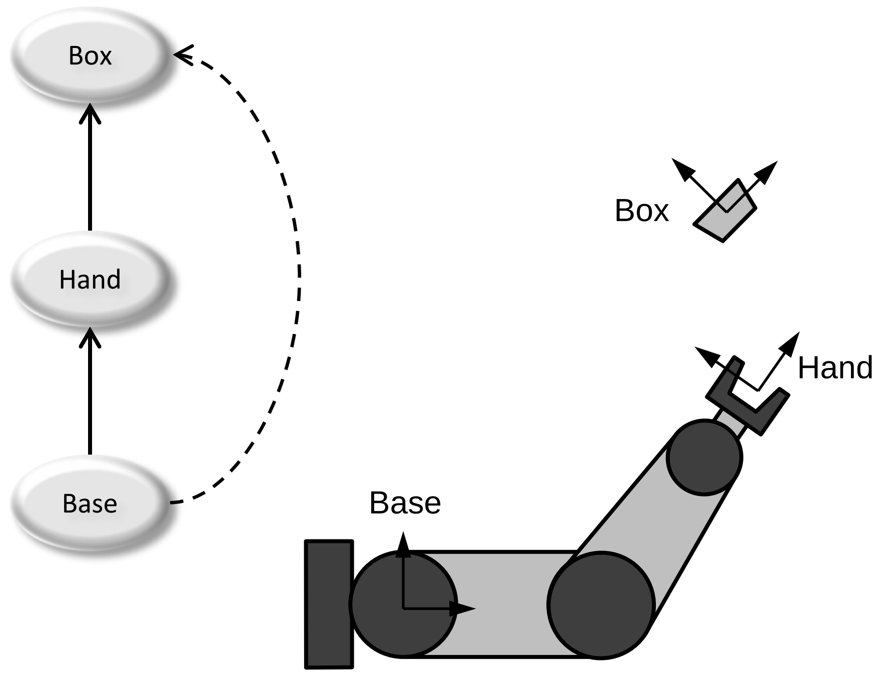
- We have 3 elements
  - 2 objects
  - a relationship
- The relationship is directional.
- So, draw it with an arrow like so....
- We also know how to compute the inverse relationship from box to hand.



"Pose of box is known wrt hand".

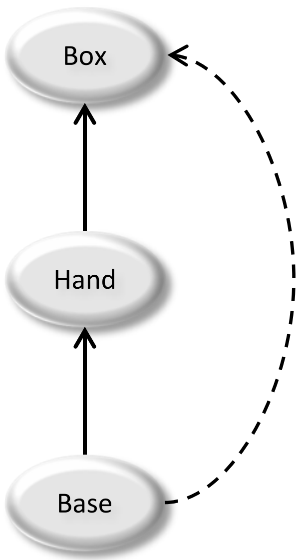
Compounded Relationships

- Suppose the hand belongs to a robot and we have a fwd kinematic solution.
- Call this figure a:
  - Transform Graph, or
  - Pose Network



Rigidity and Transitivity

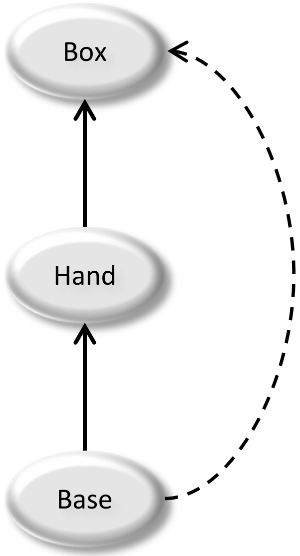
- Suppose (for now), the edges represent rigid spatial relationships.
- Connectedness is transitive.
  - Two nodes in a graph are connected if there is a path between them.
- Therefore:
  - Anything is fixed (known) wrt anything else if there is a path between them.



Derived Relationships

- How to compute their relationship?

$$T_{box}^{base} = T_{\cancel{hand}}^{base} T_{box}^{\cancel{hand}}$$

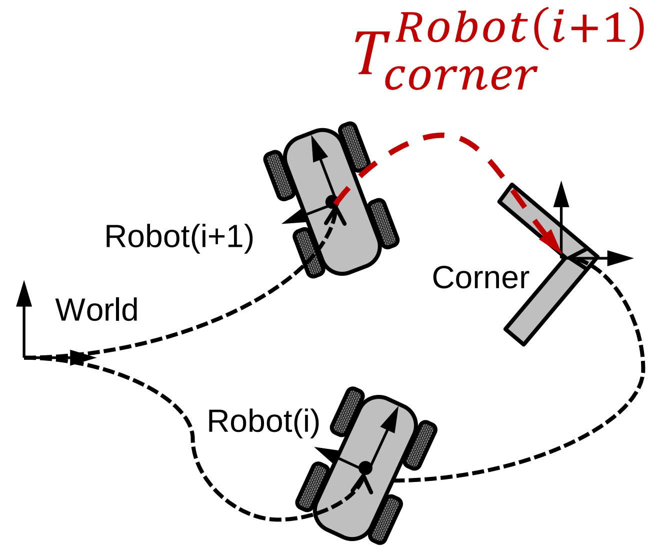


- This relationship is computable even though it is not in the graph.

2.7.1.2 Trees of Relationships

- Often pose networks are treelike in structure.
- For tracking … can we compute?

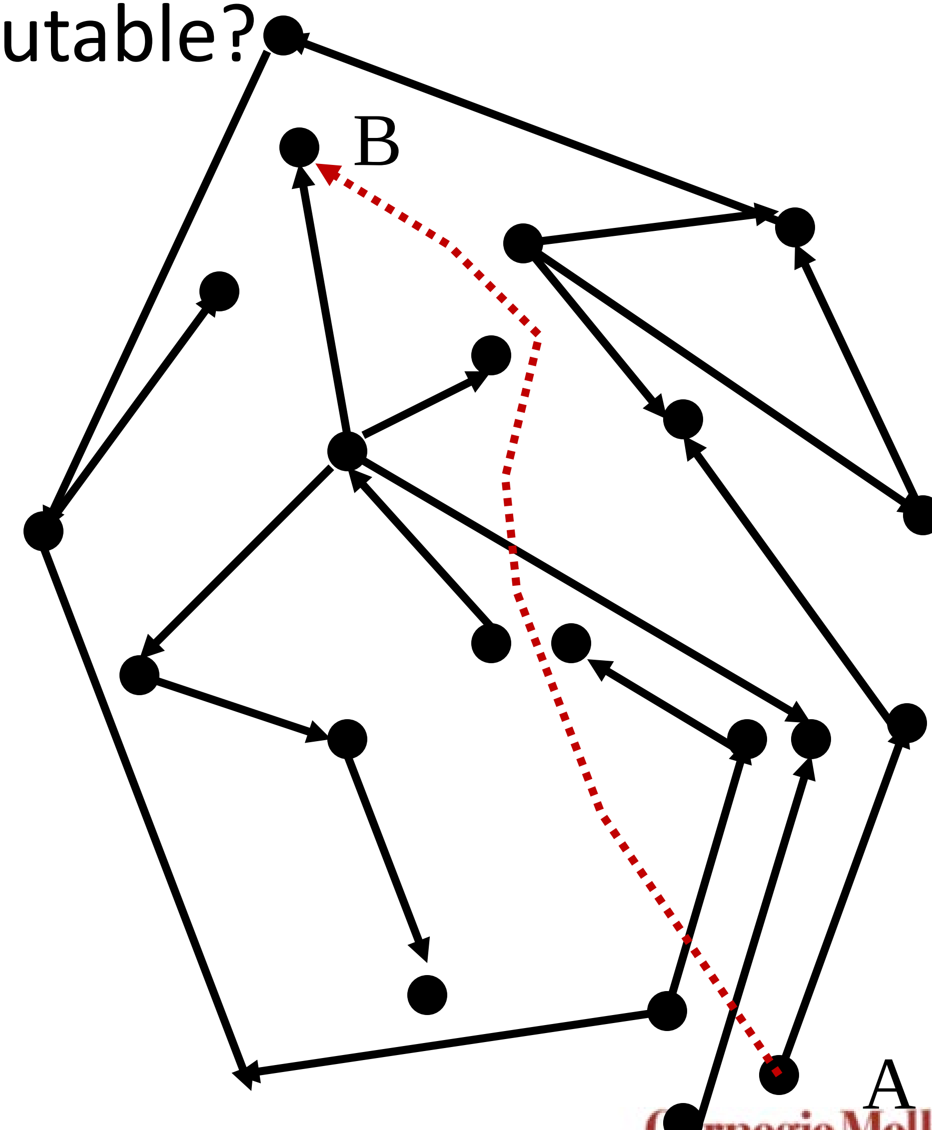
$$T_{\text{corner}}^{\text{Robot}_{i+1}}$$



- Useful to imagine the robot "stamping" the floor with axes.

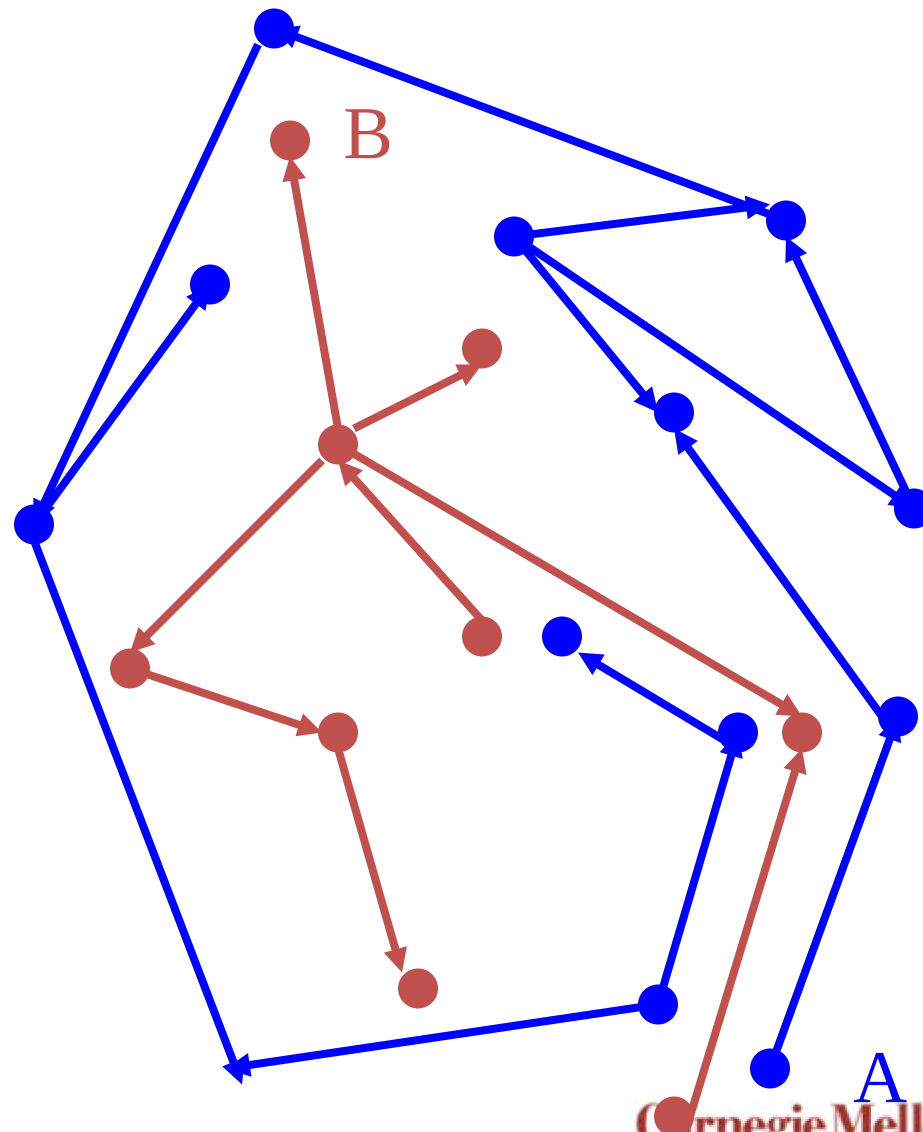
General Derived Relationships

- Is this transform computable?



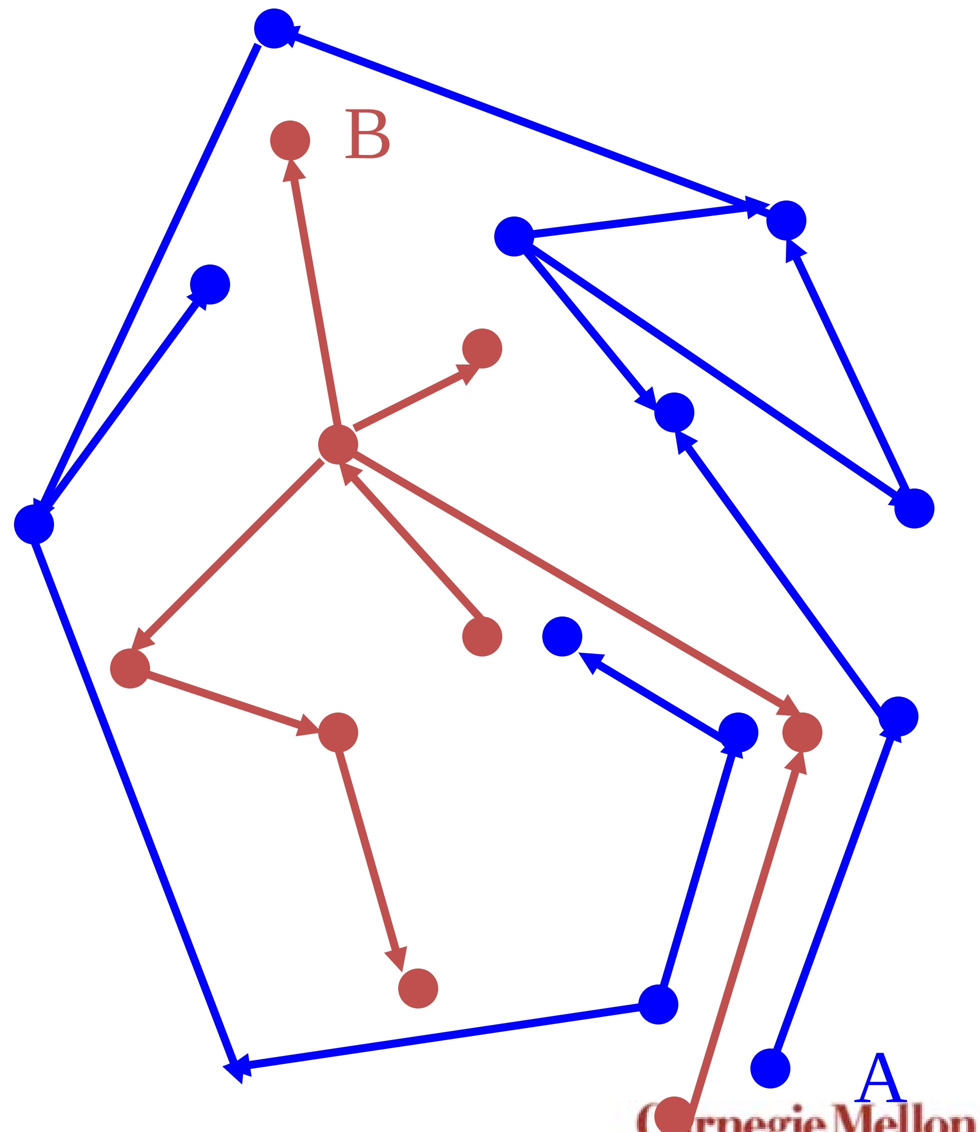
General Derived Relationships

- Is there a path from A to B?
  - Hint: Look at the colors of the letters.



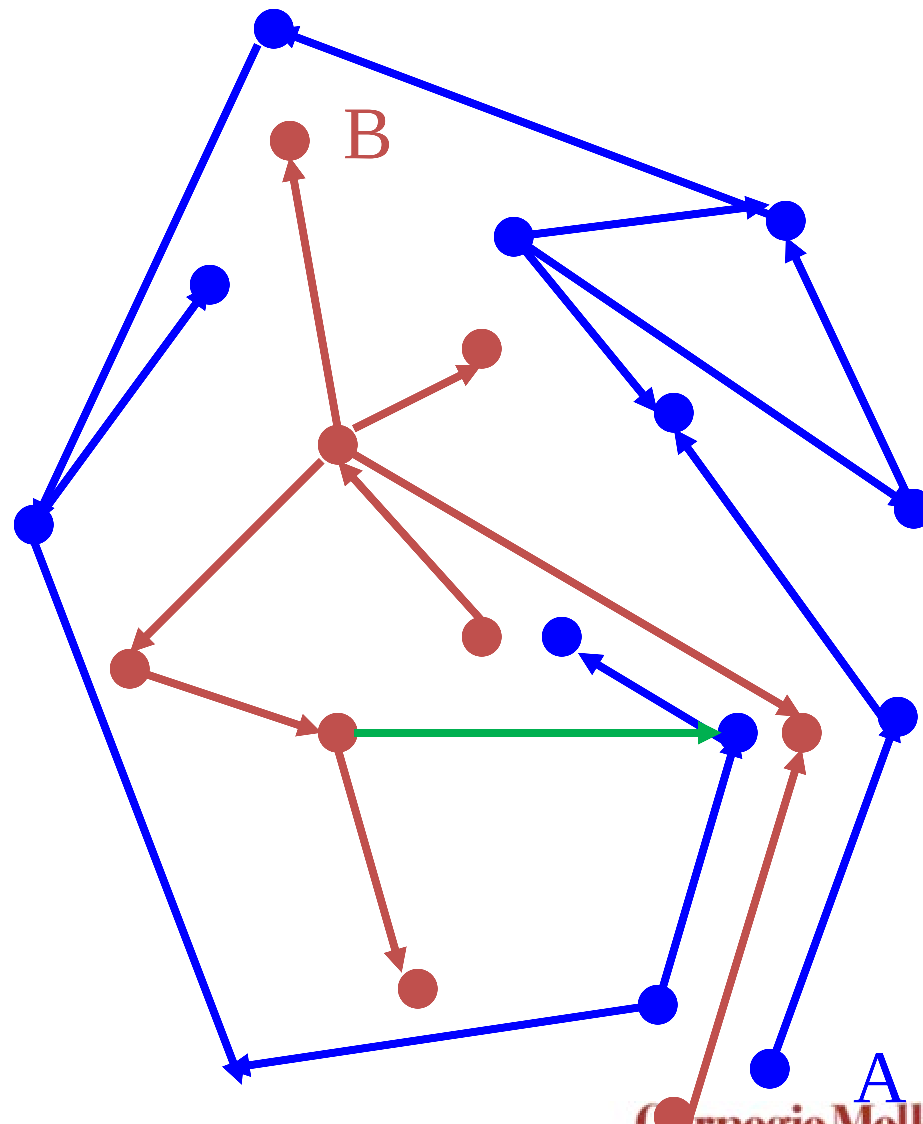
General Derived Relationships

- Is there a path from A to B?
  - Hint: Look at the colors of the letters.
- How could we fix it?



General Derived Relationships

- Is there a path from A to B?
  - Hint: Look at the colors of the letters.
- How could we fix it?
  - Like so…



2.7.1.3 Uniqueness

- For a given relationship:
  - The HT matrix is uniquely defined.
  - The corresponding pose is not (in 3D).
- Euler angles are not unique:
  - Two solutions for a given HT.
- Also Euler angles are ambiguous.
  - Several conventions in use (zyx, zxy etc.)

$$T_b^a = \begin{bmatrix} c\psi c\theta & (c\psi s\theta s\phi - s\psi c\phi) & (c\psi s\theta c\phi + s\psi s\phi) & u \\ s\psi c\theta & (s\psi s\theta s\phi + c\psi c\phi) & (s\psi s\theta c\phi - c\psi s\phi) & v \\ -s\theta & c\theta s\phi & c\theta c\phi & w \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\underline{\rho}_b^a = \begin{bmatrix} u & v & w & \phi & \theta & \psi \end{bmatrix}$$

Coding Poses

- In OO code, it makes sense for these to be different manifestations of the same class.

C++

```
Class pose: public HT{
        …
        Double poseData[6];
        …
}
```

Java

```
Class Pose extends AffineTransform
{
}
```

Vector Space?

- 3D angles cannot be added like vectors.
- The closest facsimile is:

$$T^a_c = T^a_b T^b_c$$

- Which is kinda like:

$$\underline{\rho}^a_c = \underline{\rho}^a_b \text{“}+\text{”} \underline{\rho}^b_c$$

- In reality, pose composition is done like so:

$$\underline{\rho}^a_c = \rho[T(\underline{\rho}^a_b)T(\underline{\rho}^b_c)]$$

- Write this stylistically like so:

$$\underline{\rho}^a_c = \underline{\rho}^a_b * \underline{\rho}^b_c$$

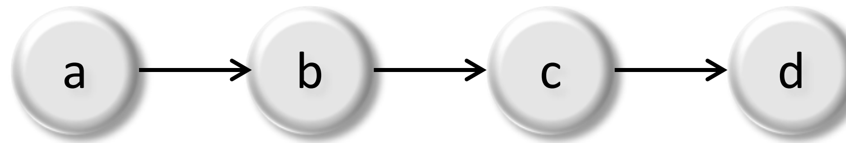
Outline

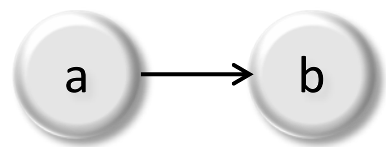
- 2.7.1 Transforms as Relationships
- <u>2.7.2 Solving Pose Networks</u>
- 2.7.3 Overconstrained Networks
- 2.7.4 Differential Kin of Frames in General Position
- Summary

Graph To Math

- Our fun with subscripts and superscripts is equivalent to finding a path in a network.
- Common letters in adjacent T's means edges are adjacent.
- Again, if a path exists, you are in business.

$$T_d^a = T_b^a T_c^b T_d^c$$





- Means pose of b wrt a is known.
- T_b^a is known.

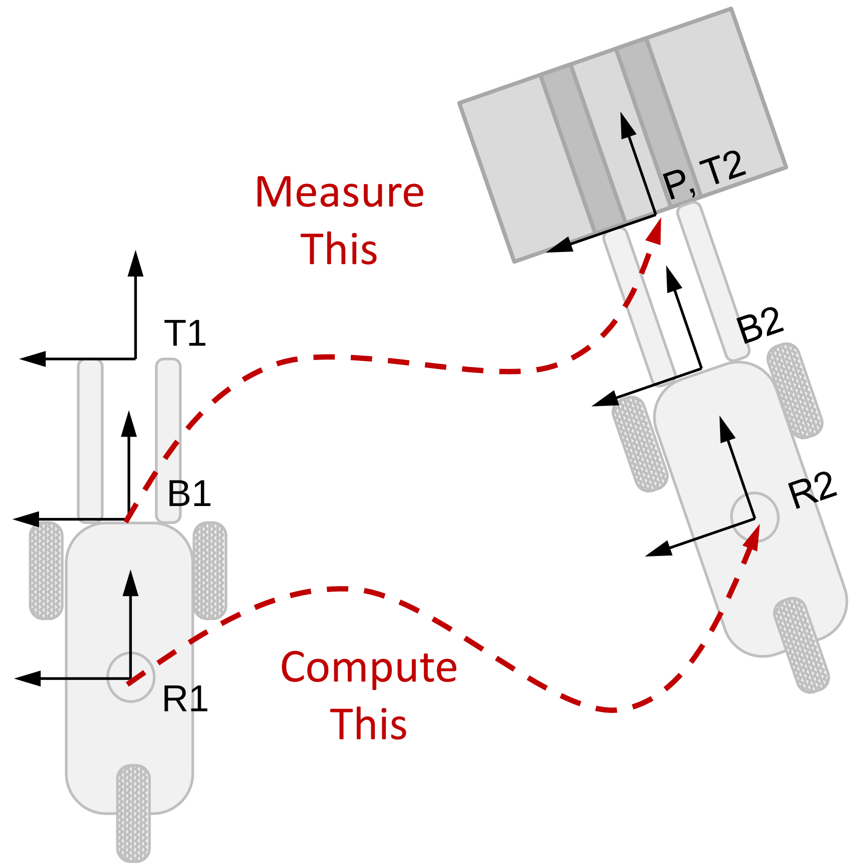
Solving for a Pose in a Graph

1. Write down what you know. Draw the frames involved in a roughly correct spatial relationship. Draw all known edges.
2. Find the (or a) path from the start (superscript) to the end (subscript).
3. Write "O"perator matrices in left to right order as the path is traversed.
   – Invert any transforms whose arrow is followed in the reverse direction.
4. Substitute all known constraints.

2.7.2.2 Example: Robot Fork Truck

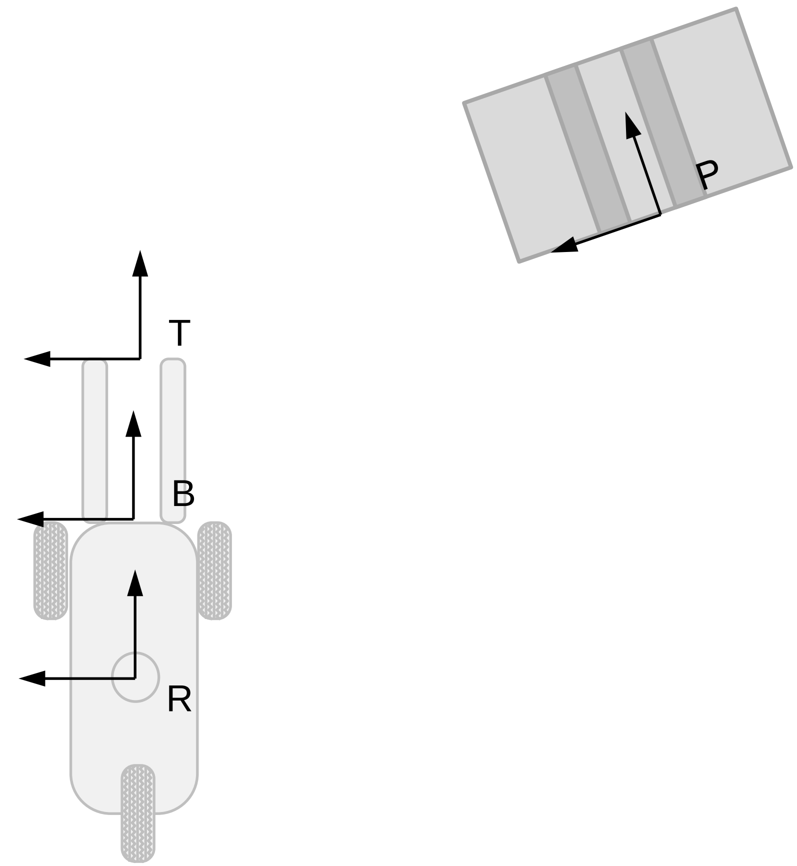
- Vision system sees pallet.

- Find:
  - desired new pose of robot
    …
  - … relative to present pose.
- Fork tips must line up with pallet holes.



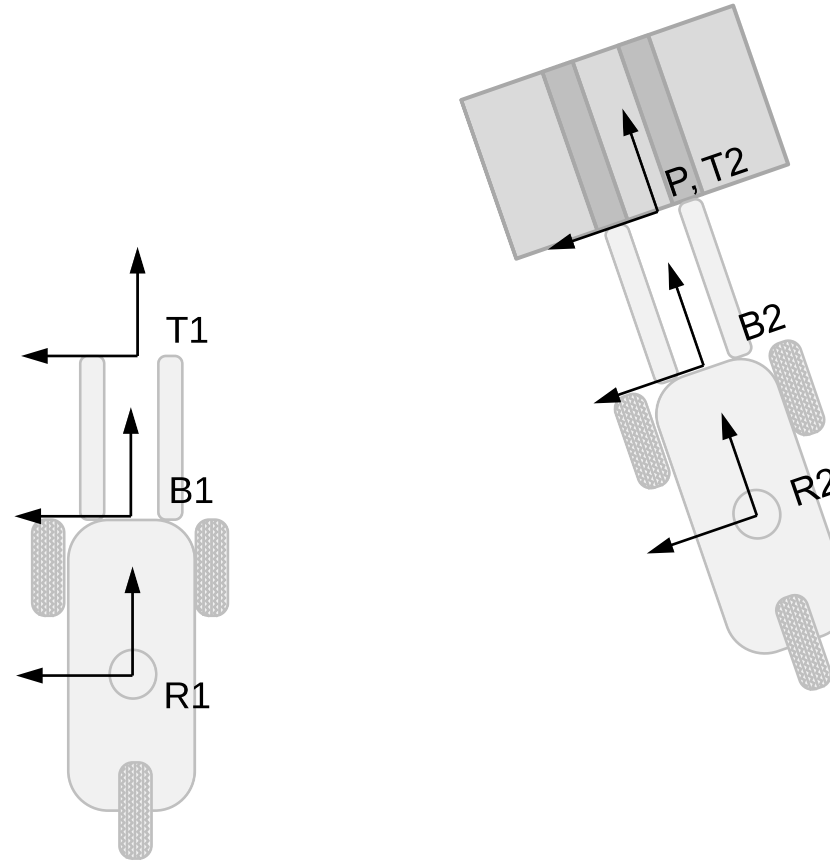
2.7.2.2 Example: Robot Fork Truck

- Designate all relevant frames.
  - Robot
  - Base
  - Tip
  - Pallet



2.7.2.2 Example: Robot Fork Truck

- A robot “here” …
- … and a robot “there”.



2.7.2.2 Example: Robot Fork Truck

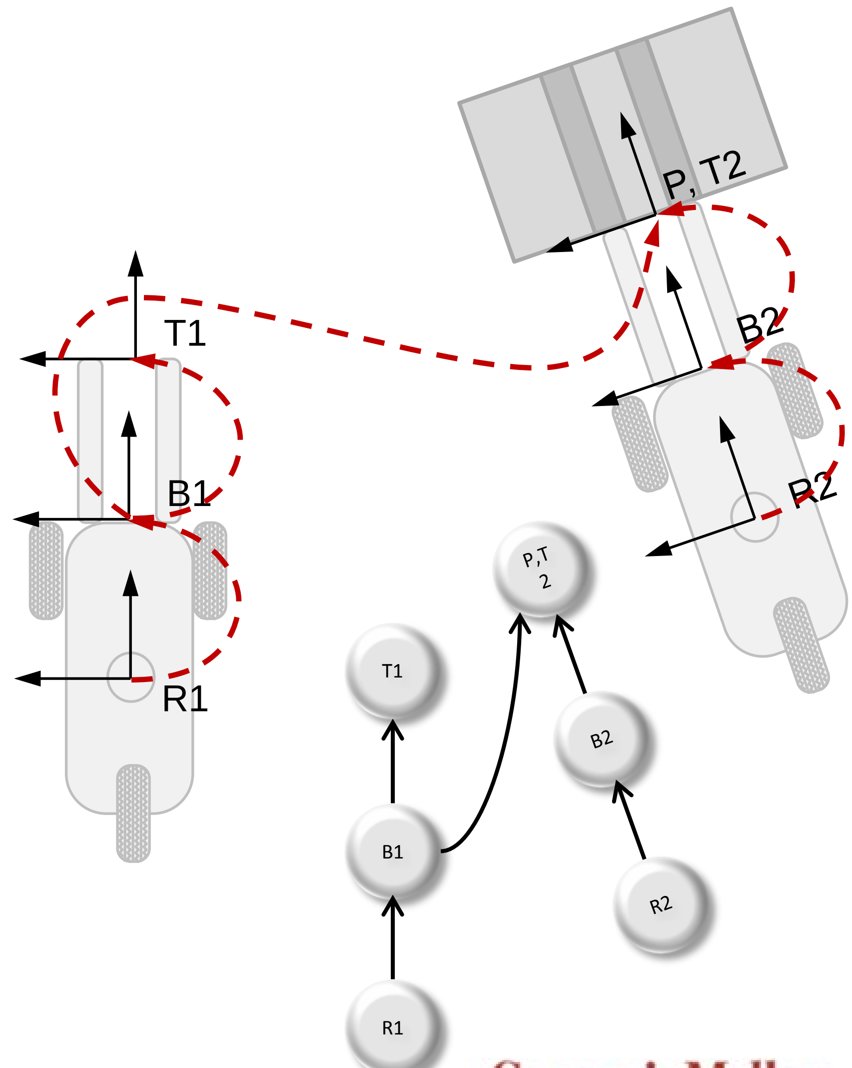
- Draw known transforms:
- Read path from R1 to R2:

from camera

$$T^{R1}_{R2} = T^{R1}_{B1} T^{B1}_{P} T^{P}_{T2} T^{T2}_{B2} T^{B2}_{R2}$$

- The solution requires:

$$T^{P}_{T2} = I$$



2.7.2.2 Example: Robot Fork Truck

- Transforms relating frames fixed to robot are all constant:

$$T^{R1}_{B1} = T^{R}_{B}$$

$$T^{B2}_{R2} = (T^{R}_{B})^{-1}$$

$$T^{T2}_{B2} = (T^{B}_{T})^{-1}$$

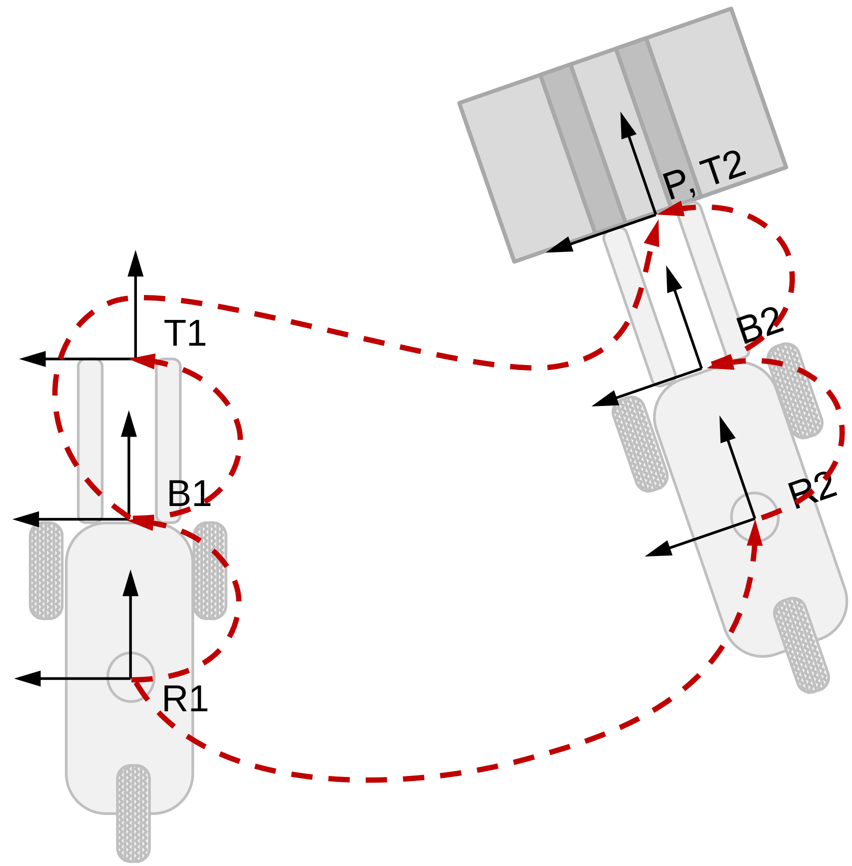
- Substituting:

$$T^{R1}_{R2} = T^{R1}_{B1} T^{B1}_{P} \overset{I}{\cancel{T^{P}_{T2}}} T^{T2}_{B2} T^{B2}_{R2}$$

$$T^{R1}_{R2} = T^{R}_{B} T^{B}_{P} (T^{B}_{T})^{-1} (T^{R}_{B})^{-1}$$

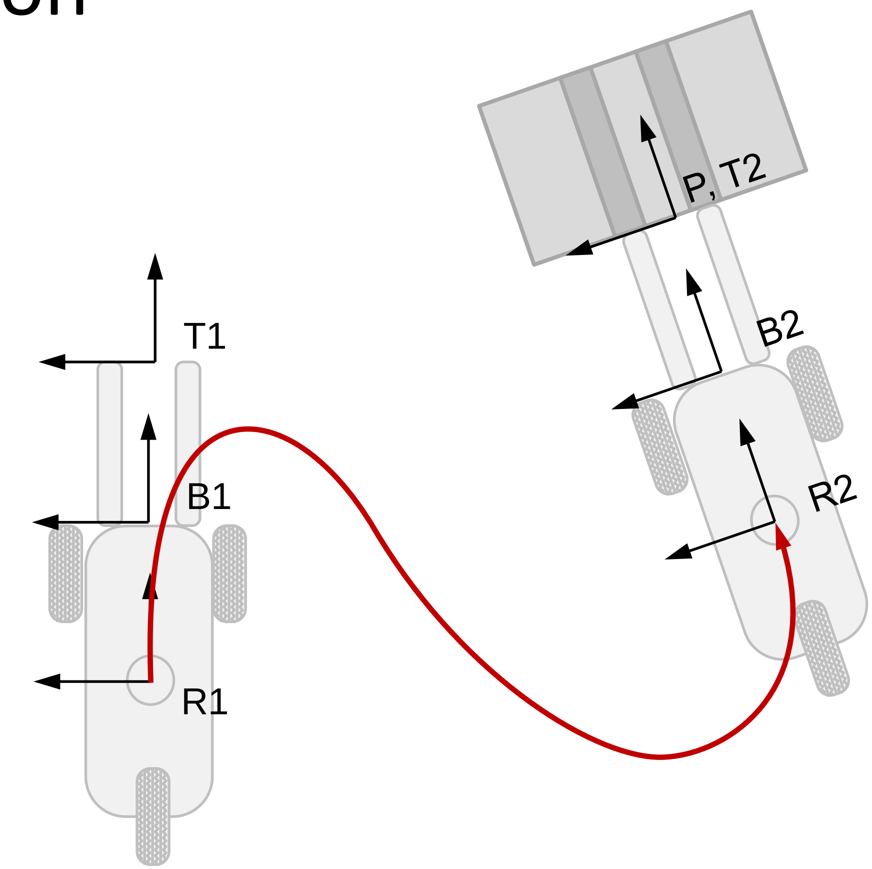
Measurement

Very common
“Sandwich”



Trajectory Generation

- Generating a feasible motion to connect the two is not trivial.
- See later in course.

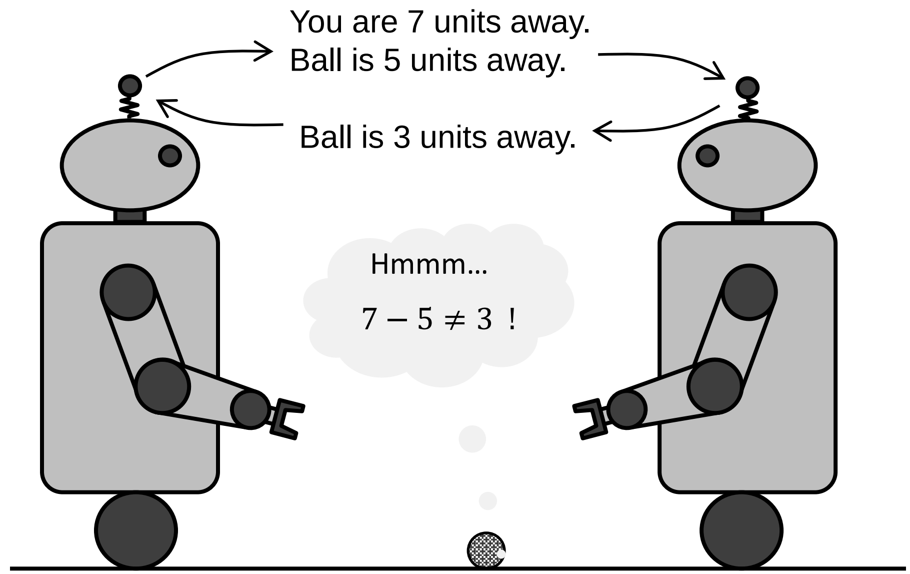


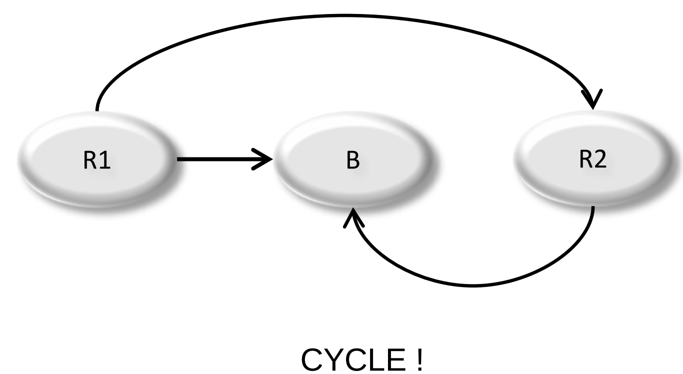
Outline

- 2.7.1 Transforms as Relationships
- 2.7.2 Solving Pose Networks
- 2.7.3 Overconstrained Networks
- 2.7.4 Differential Kin of Frames in General Position
- Summary

2.7.3 Cyclic Networks

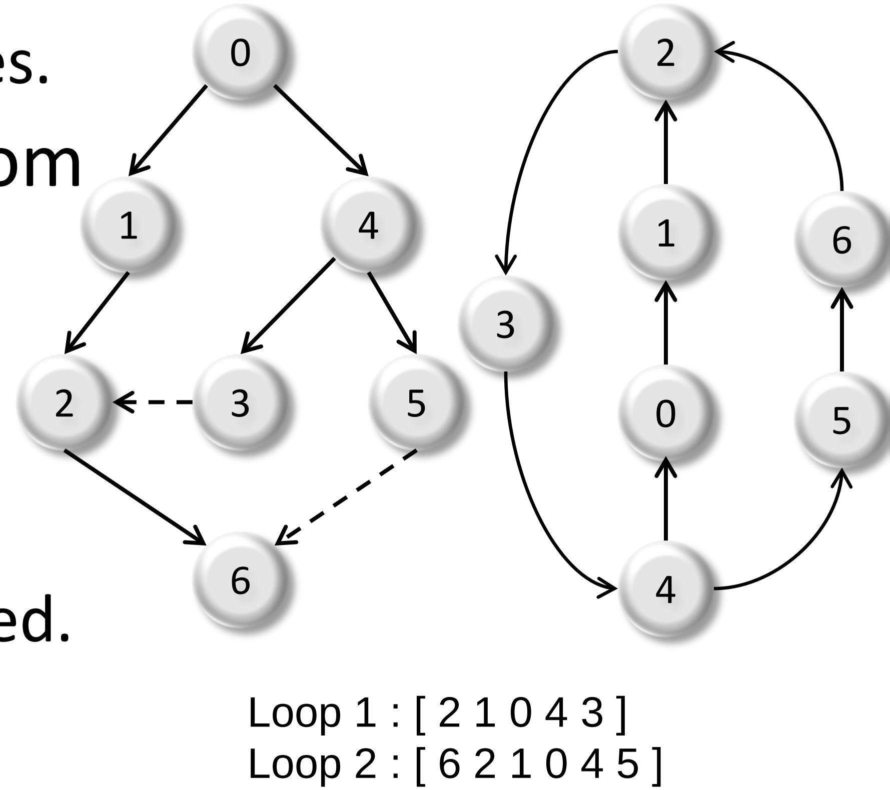
- Any edges beyond the number required to connect everything create cycles in the network.
- Cycles create the possibility of inconsistent information.
- Cycles are not always bad.
  - They contain rare, powerful information.





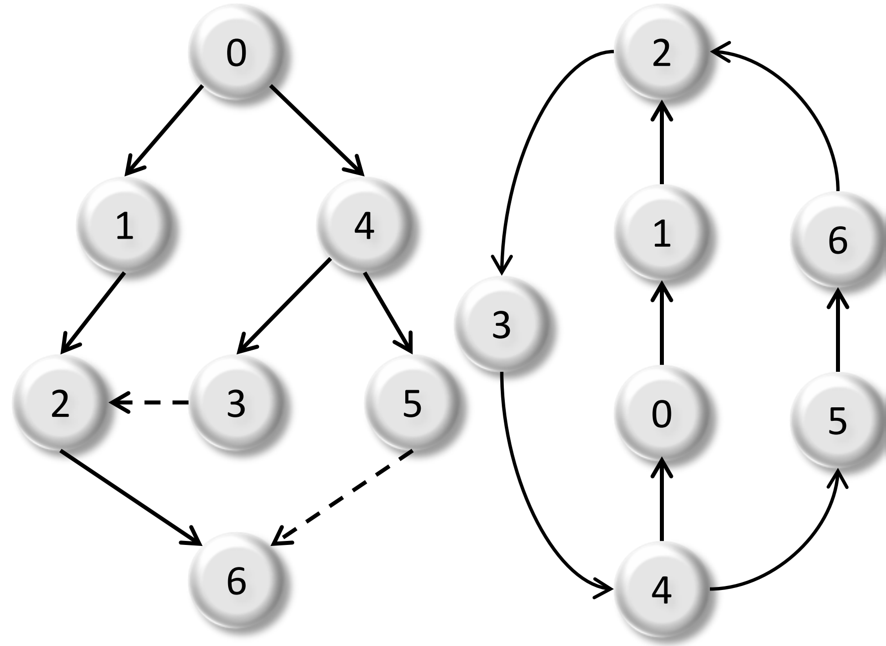
2.7.3.1 Spanning Tree

- A "tree" is fully connected but acyclic
  - Just the right number of edges.
- Can form a spanning tree from a cyclic network.
  - Pick a root.
  - Traverse any way you like.
  - Enter nodes only once.
  - Stop when all nodes connected.
- Motion planning algorithms are based heavily on this notion.



Paths in Spanning Trees

- Each node has exactly one parent.
- Exactly one acyclic path between any two nodes.
  - Through most recent common ancestor
- Each omitted edge closes an independent cycle in the cycle basis of the graph.



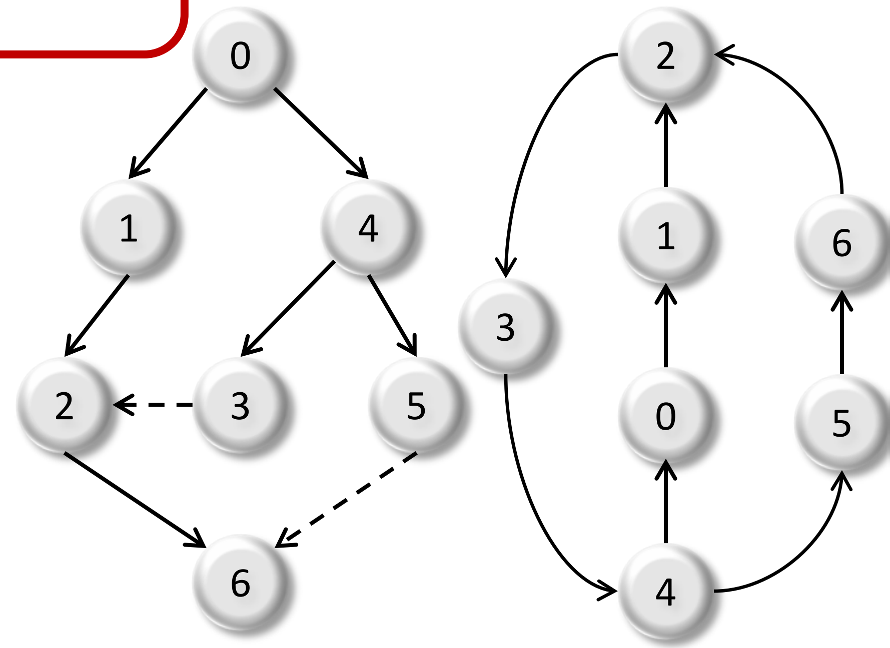
Loop 1 : [2 1 0 4 3]
Loop 2 : [6 2 1 0 4 5]

2.7.3.2 Cycle Basis

- N nodes
- N-1 edges in Spanning Tree

Why?

- If there were E edges in original network:
- Then there were:
  - L=E-(N-1) …
  - … independent loops in original network..

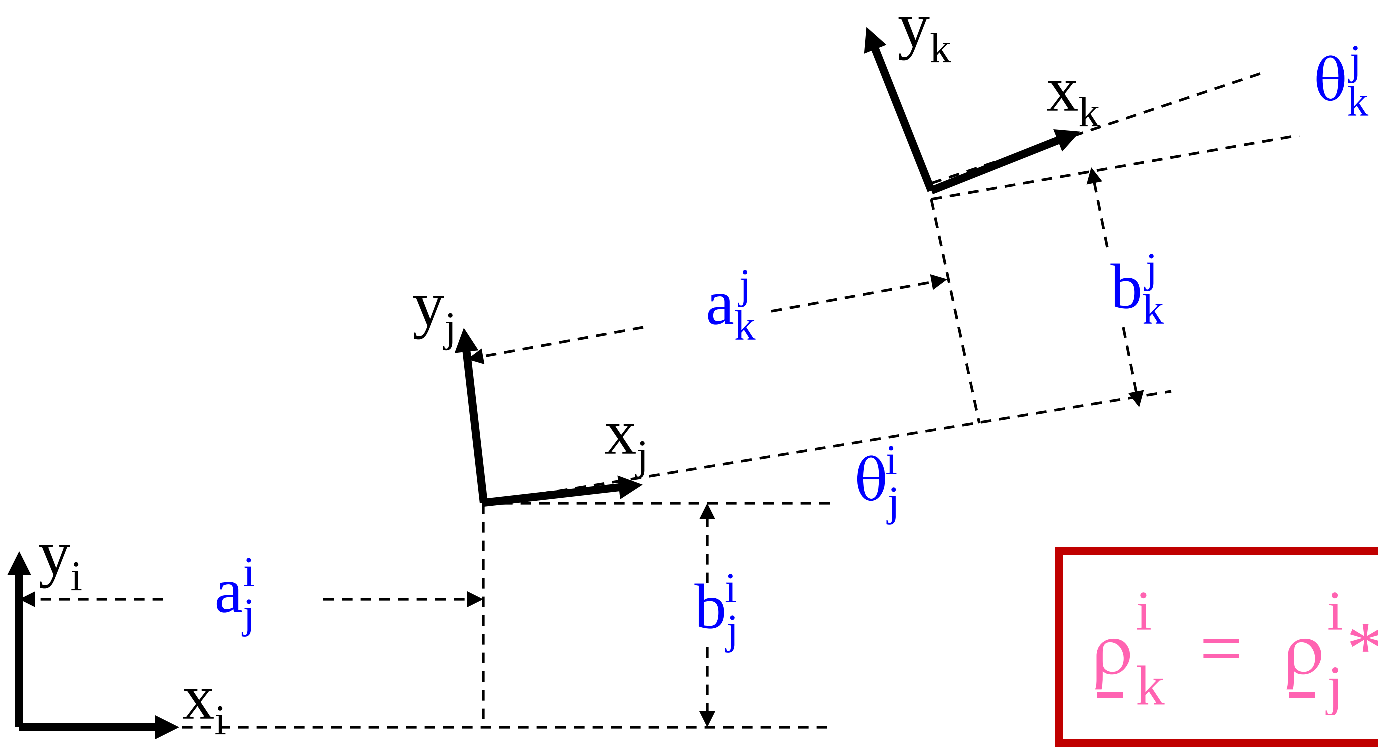


Loop 1 : [2 1 0 4 3]
Loop 2 : [6 2 1 0 4 5]

Outline

- 2.7.1 Transforms as Relationships
- 2.7.2 Solving Pose Networks
- 2.7.3 Overconstrained Networks
- <u>2.7.4 Differential Kin of Frames in General Position</u>
- Summary

2.7.4.1 Pose Composition



$$\boxed{\underline{\rho}_k^i = \underline{\rho}_j^i * \underline{\rho}_k^j}$$

- What are a_k^i , b_k^i and θ_k^i ?

CMU RI 16-761
Mobile Robots

LAST UPDATED 16 January 2015

Contact for questions or assistance:

Instructor: Al Kelly NSH 3209 <alonzo@cmu.edu>
TA: Joe Giampapa NSH 4519 <garof@andrew.cmu.edu>

ie Mellon
ISTITUTE

Pose Composition

- The pose of frame k with respect to frame i can be extracted from the compound transform:

$$T_k^i = \begin{bmatrix} c\theta_j^i & -s\theta_j^i & a_j^i \\ s\theta_j^i & c\theta_j^i & b_j^i \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c\theta_k^j & -s\theta_k^j & a_k^j \\ s\theta_k^j & c\theta_k^j & b_k^j \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} c\theta_k^i & -s\theta_k^i & c\theta_j^i a_k^j - s\theta_j^i b_k^j + a_j^i \\ s\theta_k^i & c\theta_k^i & s\theta_j^i a_k^j + c\theta_j^i b_k^j + b_j^i \\ 0 & 0 & 1 \end{bmatrix}$$

- Read result (more or less) directly:

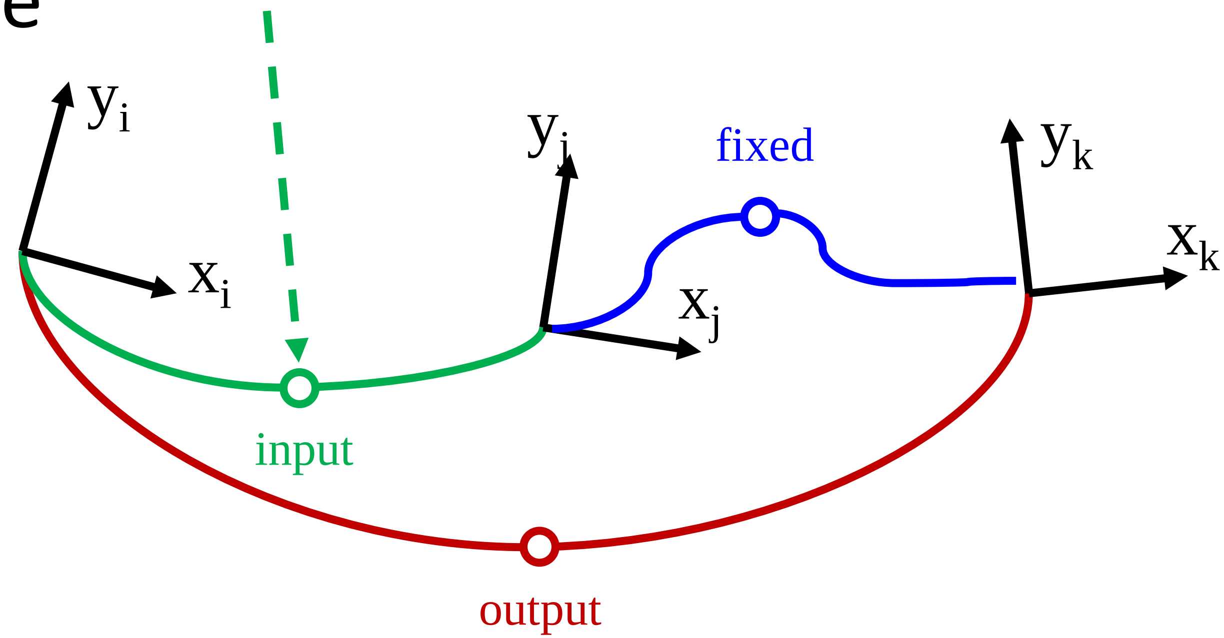
$$a_k^i = c\theta_j^i a_k^j - s\theta_j^i b_k^j + a_j^i \qquad \text{Eqn A}$$

$$b_k^i = s\theta_j^i a_k^j + c\theta_j^i b_k^j + b_j^i$$

$$\theta_k^i = \theta_j^i + \theta_k^j$$

2.7.4.2 Compound-Left Pose Jacobian

- See figure



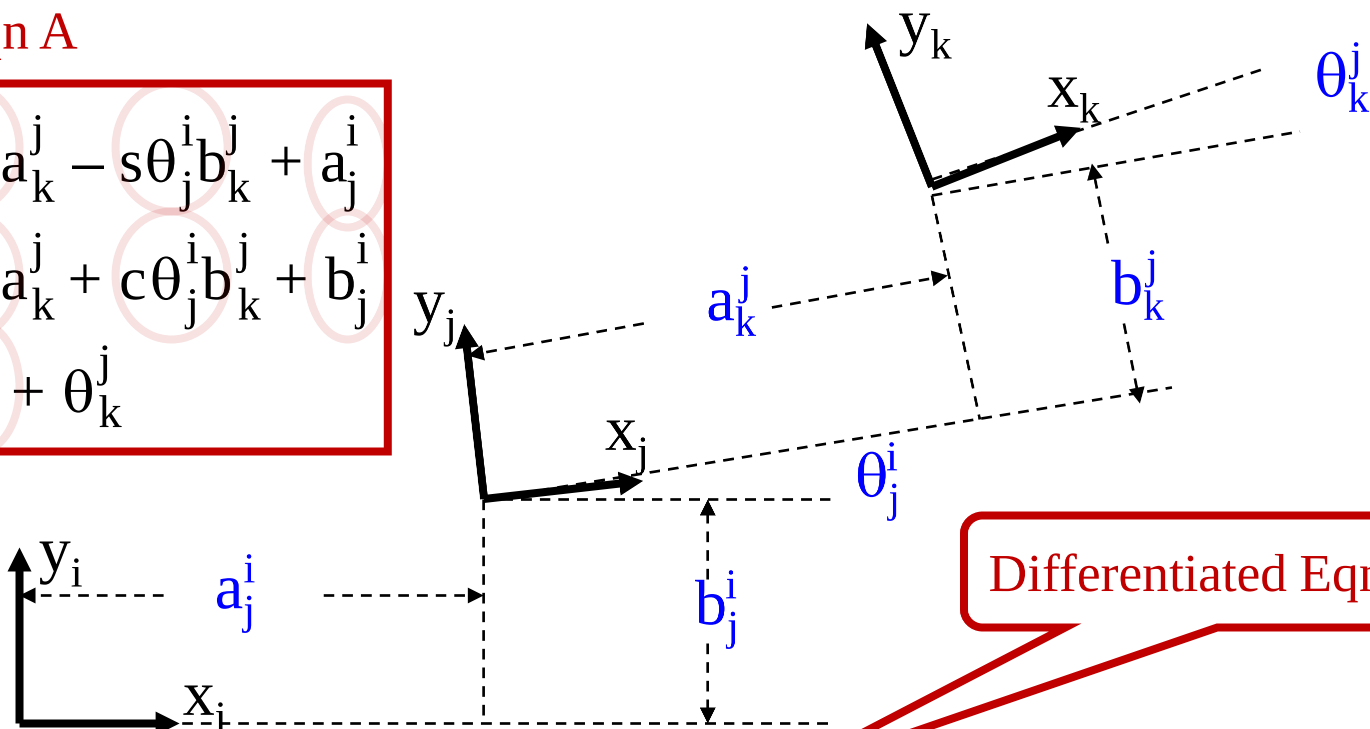
$$J_{ij}^{ik} = \frac{\partial \underline{\rho}_k^i}{\partial \underline{\rho}_j^i}$$

Jiki = geekey

2.7.4.2 Compound-Left Pose Jacobian

Eqn A

$$a_k^i = c\theta_j^i a_k^j - s\theta_j^i b_k^j + a_j^i$$
$$b_k^i = s\theta_j^i a_k^j + c\theta_j^i b_k^j + b_j^i$$
$$\theta_k^i = \theta_j^i + \theta_k^j$$

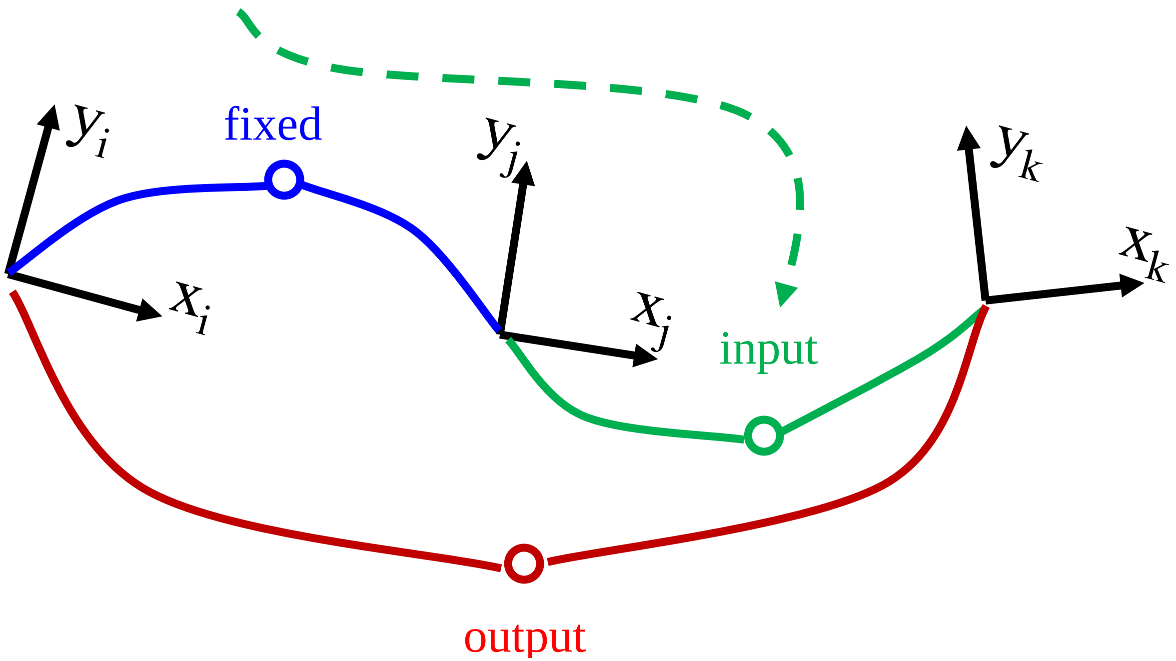


Differentiated Eqn A

$$J_{ij}^{ik} = \frac{\partial \underline{\rho}_k^i}{\partial \underline{\rho}_j^i} = \begin{bmatrix} 1 & 0 & -(s\theta_j^i a_k^j + c\theta_j^i b_k^j) \\ 0 & 1 & (c\theta_j^i a_k^j - s\theta_j^i b_k^j) \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & -(b_k^i - b_j^i) \\ 0 & 1 & (a_k^i - a_j^i) \\ 0 & 0 & 1 \end{bmatrix}$$

2.7.4.3 Compound-Right Pose Jacobian

- See figure



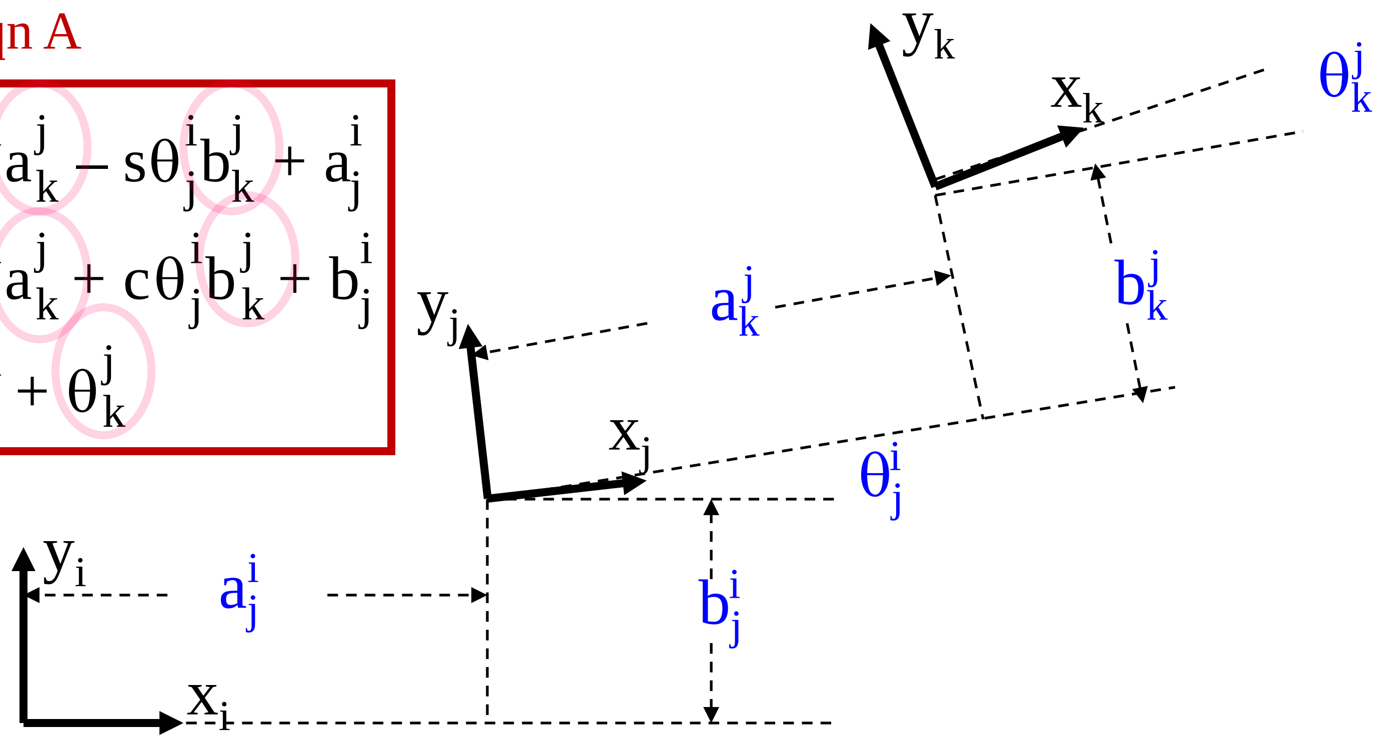
$$J^{ik}_{jk} = \frac{\partial \underline{\rho}^{i}_{k}}{\partial \underline{\rho}^{j}_{k}}$$

kjki = kudgekey

2.7.4.3 Compound-Right Pose Jacobian

Eqn A

$$a_k^i = c\theta_j^i a_k^j - s\theta_j^i b_k^j + a_j^i$$
$$b_k^i = s\theta_j^i a_k^j + c\theta_j^i b_k^j + b_j^i$$
$$\theta_k^i = \theta_j^i + \theta_k^j$$

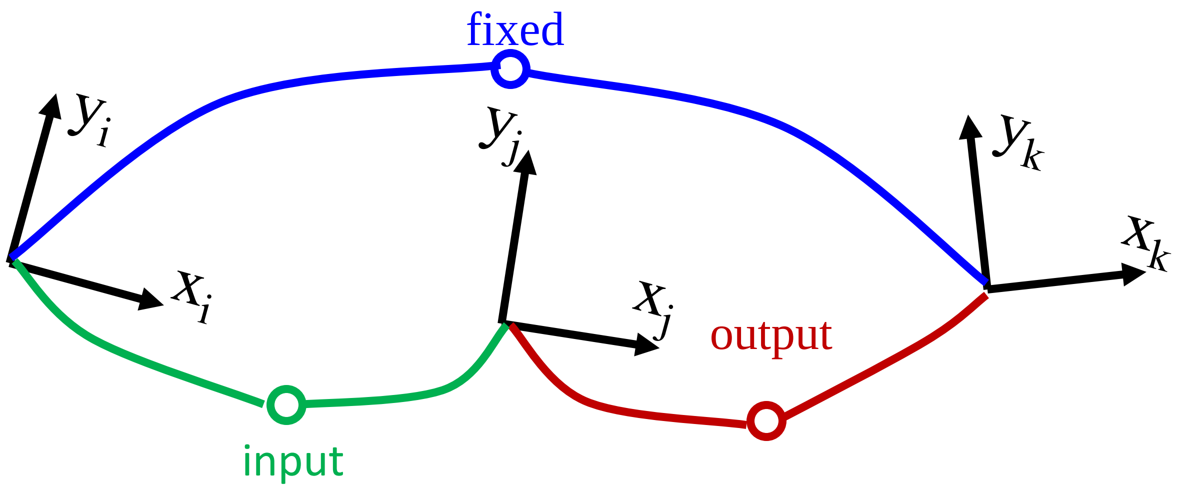


$$J_{jk}^{ik} = \frac{\partial \underline{\rho}_k^i}{\partial \underline{\rho}_k^j} = \begin{bmatrix} c\theta_j^i & -s\theta_j^i & 0 \\ s\theta_j^i & c\theta_j^i & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Differentiated Eqn A

2.7.4.4 Right-Left Pose Jacobian

- See figure



$$J_{ij}^{jk} = \frac{\partial \underline{\rho}_k^j}{\partial \underline{\rho}_j^i}$$

Jikj = geekudge

2.7.4.4 Right-Left Pose Jacobian

$$\begin{bmatrix} a_k^j \\ b_k^j \end{bmatrix} = \begin{bmatrix} c\theta_j^i & s\theta_j^i \\ -s\theta_j^i & c\theta_j^i \end{bmatrix} \begin{bmatrix} a_k^i - a_j^i \\ b_k^i - b_j^i \end{bmatrix} = \begin{bmatrix} c\theta_j^i & s\theta_j^i \\ -s\theta_j^i & c\theta_j^i \end{bmatrix} \begin{bmatrix} {}^i a_k^j \\ {}^i b_k^j \end{bmatrix}$$

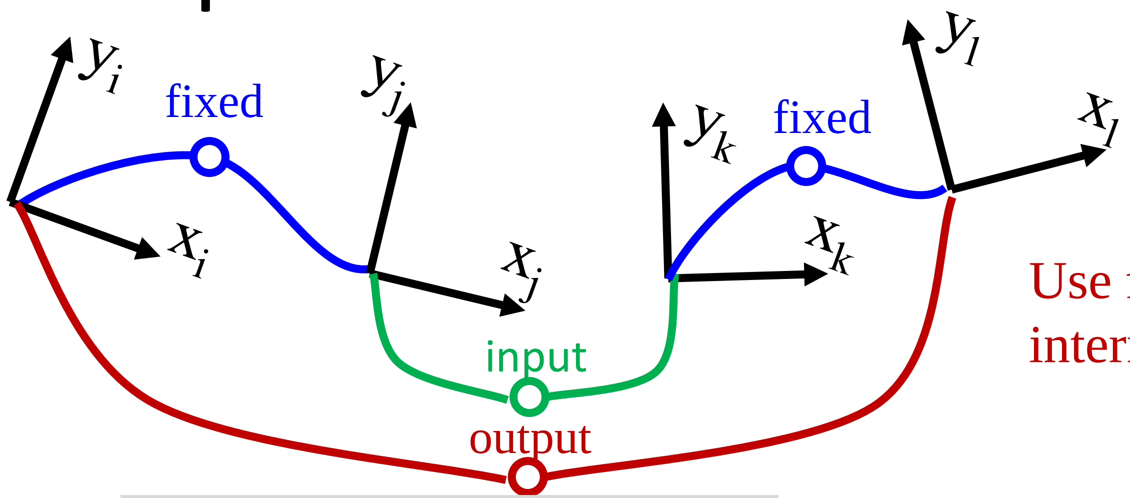
$$\theta_k^j = \theta_k^i - \theta_j^i$$

Eqn B

$$J_{ij}^{jk} = \frac{\partial \underline{\rho}_k^j}{\partial \underline{\rho}_j^i} = -\begin{bmatrix} c\theta_j^i & s\theta_j^i & -b_k^j \\ -s\theta_j^i & c\theta_j^i & a_k^j \\ 0 & 0 & 1 \end{bmatrix}$$

Differentiated Eqn B
(not easy. see text)

2.7.4.5 Compound Inner Pose Jacobian



Use frame k as intermediate

$$J^{il}_{jk} = \frac{\partial \underline{\rho}^i_l}{\partial \underline{\rho}^j_k} = \left(\frac{\partial \underline{\rho}^i_l}{\partial \underline{\rho}^i_k}\right)\left(\frac{\partial \underline{\rho}^i_k}{\partial \underline{\rho}^j_k}\right) = J^{il}_{ik}\, J^{ik}_{jk}$$

Jkli = kili * kjki

Jucklee = keylee*kudgekey

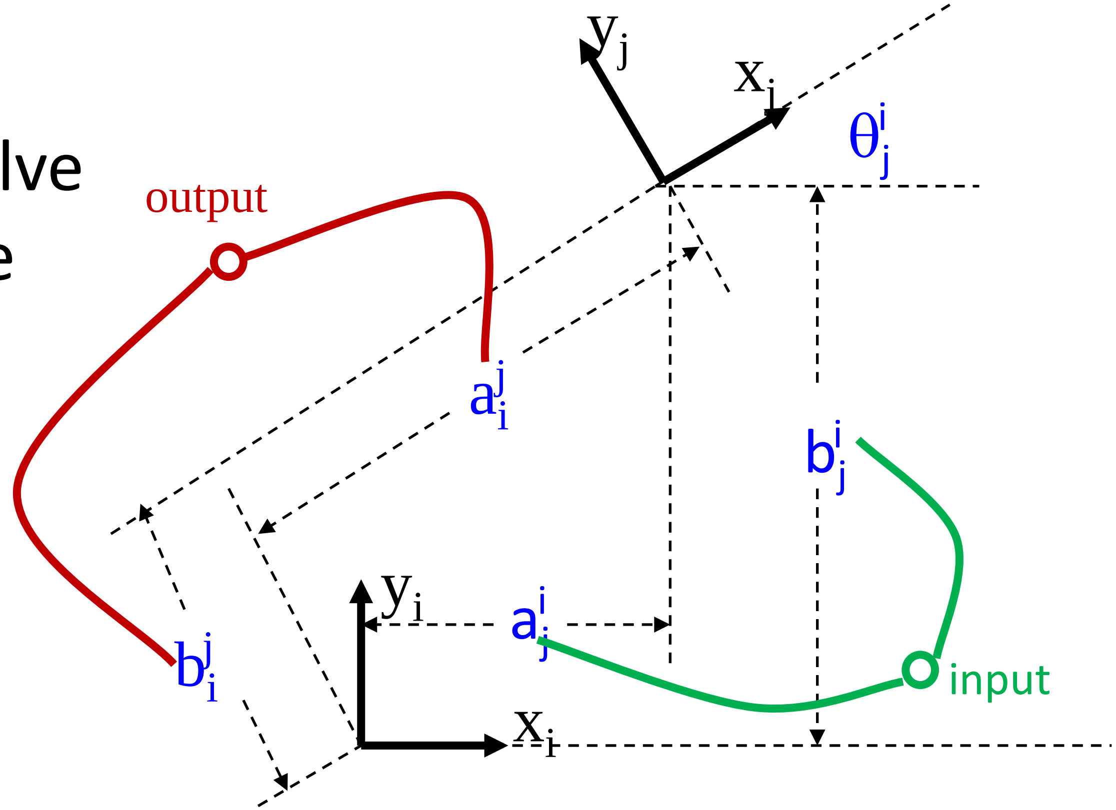
$$J^{il}_{jk} = \frac{\partial \underline{\rho}^i_l}{\partial \underline{\rho}^j_k} = \begin{bmatrix} 1 & 0 & -(b^i_l - b^i_k) \\ 0 & 1 & (a^i_l - a^i_k) \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c\theta^i_j & -s\theta^i_j & 0 \\ s\theta^i_j & c\theta^i_j & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

C-Left * C-Right !

$$J^{il}_{jk} = \frac{\partial \underline{\rho}^i_l}{\partial \underline{\rho}^j_k} = \begin{bmatrix} c\theta^i_j & -s\theta^i_j & -{}^i b^k_l \\ s\theta^i_j & c\theta^i_j & {}^i a^k_l \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} c\theta^i_j & -s\theta^i_j & -(b^i_l - b^i_k) \\ s\theta^i_j & c\theta^i_j & (a^i_l - a^i_k) \\ 0 & 0 & 1 \end{bmatrix}$$

2.7.4.6 Inverse Pose Jacobian

- Intuition:
  - Need it to solve problems like this…



$$\frac{\partial \underline{\rho}^a_d}{\partial \underline{\rho}^j_i} = \left(\frac{\partial \underline{\rho}^a_d}{\partial \underline{\rho}^i_j}\right)\boxed{\left(\frac{\partial \underline{\rho}^i_j}{\partial \underline{\rho}^j_i}\right)}$$

For two frames a & d

Inverse of a Pose

$$(T_i^j)^{-1} = T_j^i = \begin{bmatrix} c\theta_i^j & s\theta_i^j & -c\theta_i^j a_i^j - s\theta_i^j b_i^j \\ -s\theta_i^j & c\theta_i^j & s\theta_i^j a_i^j - c\theta_i^j b_i^j \\ 0 & 0 & 1 \end{bmatrix}$$

$$\boxed{\begin{aligned} a_j^i &= -c\theta_i^j a_i^j - s\theta_i^j b_i^j \\ b_j^i &= s\theta_i^j a_i^j - c\theta_i^j b_i^j \\ \theta_j^i &= -\theta_i^j \end{aligned}}$$

Eqn C

$$\frac{\partial \underline{\rho}_j^i}{\partial \underline{\rho}_i^j} = \begin{bmatrix} -c\theta_i^j & -s\theta_i^j & s\theta_i^j a_i^j - c\theta_i^j b_i^j \\ s\theta_i^j & -c\theta_i^j & c\theta_i^j a_i^j + s\theta_i^j b_i^j \\ 0 & 0 & -1 \end{bmatrix} = -\begin{bmatrix} c\theta_i^j & s\theta_i^j & -b_j^i \\ -s\theta_i^j & c\theta_i^j & a_j^i \\ 0 & 0 & 1 \end{bmatrix}$$

Differentiated Eqn C

Outline

- 2.7.1 Transforms as Relationships
- 2.7.2 Solving Pose Networks
- 2.7.3 Overconstrained Networks
- 2.7.4 Differential Kin of Frames in General Position
- <u>Summary</u>

Summary

- HTs encode a spatial relationship (between two frames).
- A "pose" and a HT are two expressions of exactly the same underlying thing.
- It is trivial to write the compound HT relating any two frames in a pose network of arbitrary complexity.
  - But it only exists if they are connected.
- Overconstrained networks can contain inconsistencies and that is valuable information.
  - Spanning Trees and Cycle Bases are fundamental concepts in such networks.
- Derivatives of pose compositions can be computed in closed form.