Kinematics and DigiFab

CS194-028
Austin Buchan
Sep. 29, 2015

Mechanisms

- Collection of connected bodies that allow relative motion between bodies
- For now restricted to rotational joints
- How do we represent these relative motions?

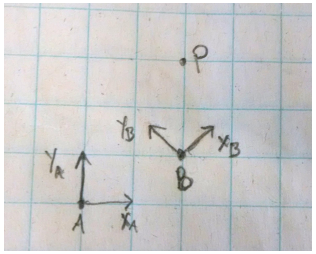
https://www.pinterest.com/pin/329536897709069237/

Kinematics

- **Forward Kinematics**: given a mechanism and state (joint angles), what are the poses of all of the bodies?
- **Inverse Kinematics**: given a desired body pose, what are the states required to reach that pose?
- Poses, Frames, Homogeneous Transformations, Rotation Matrices
- Body Frames and Joint Frames
- Demonstration

Terminology, Notation

- A **Pose** uniquely defines the position and orientation of a **Frame**
- A **Homogeneous Transformation** provides a convenient matrix representation for converting homogeneous coordinates from one frame to another



$$\mathrm{H}_A^B = \begin{bmatrix} 0.707 & -0.707 & 0 & 2 \\ 0.707 & 0.707 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, p^A = \begin{bmatrix} 2 \\ 3 \\ 0 \\ 1 \end{bmatrix} p^B = \begin{bmatrix} 1.414 \\ 1.414 \\ 0 \\ 1 \end{bmatrix}, H_A^B p^B = p^A$$

- A **Rotation Matrix** is a parameterized **Homogeneous Transformation** that rotates a frame about an axis (Z in our convention) by parameter theta

$$R_z(\theta) = \begin{bmatrix} cos(\theta) & -sin(\theta) & 0 & 0 \\ sin(\theta) & cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

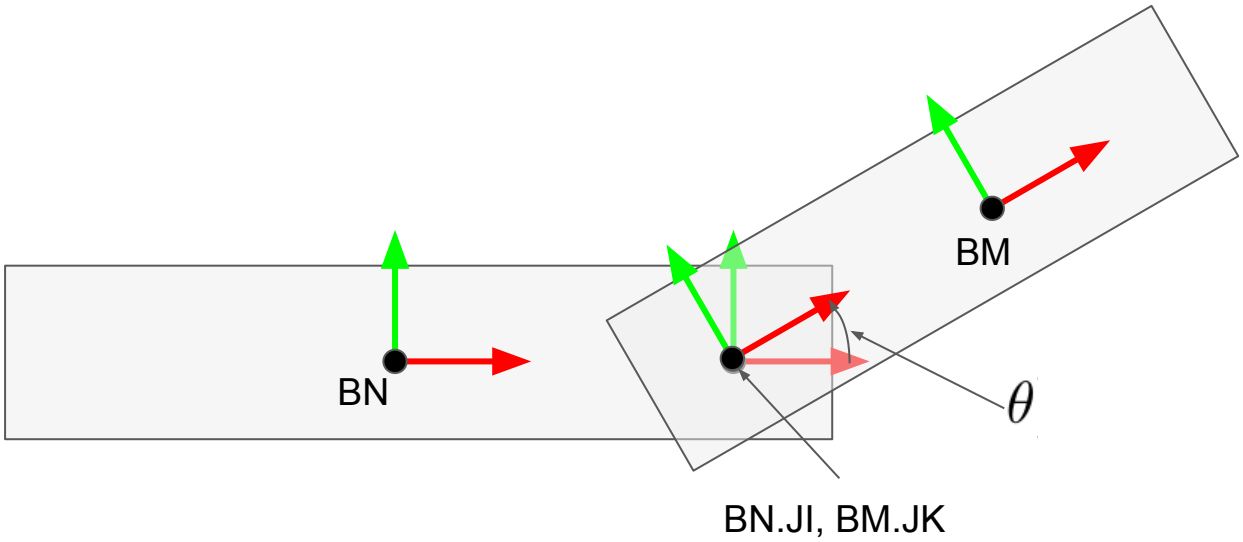
Body Frames, Joint Frames

- A **Body** is a type of **Block** that has **Joints** in addition to Geometry (PolyMeshes)
- All geometry coordinates in a Body are fixed relative to it's Body Frame
- A Body object has attribute pose, which describes the Body Frame's pose in world coordinates
- A **Joint Frame** describes a location on a Body that can connect to another body with rotation
- A **Joint Frame** describes the pose of a joint interface relative to the Body Frame
- Connected Joints are defined to have the same origin, and rotation around the Z axis by angle theta

Forward Kinematics Demonstration

Show that a rotational connection between Body M's Joint I (BM.JI) and Body N's Joint K (BN.JK) with state theta defines the transformation between bodies M and N as:

$$H_{BN}^{BM}(\theta) = H_{BN}^{BN.JI} H_{BN.JI}^{BM.JK}(\theta) H_{BM.JK}^{BM} = H_{BN}^{BN.JI} R_z(\theta) (H_{BM}^{BM.JI})^{-1}$$



DigiFab Data Structures

Mechanism (**Layout**)

- elts: [**Body**]
- connections: [(**JointRef**, **JointRef**)]
- state: [float]
- constraints: [**Constraint**]
- children: [**Mechanism**]

JointRef = (*mechanism_index*, *body_index*, *joint_index*)

Constraint = (*type(str), ref, value*)

Body (**Block**)

- elts: [**Layer**]
- joints: [**Joint**]
- pose: **Pose**

Joint (**Serializable**)

- pose: **Pose**
- limits: [float,float]

JointRef

(*mechanism_index*, *body_index*, *joint_index*)

For all index values, can either be a numerical index, or the name of an element. For *mechanism_index*, 0 indicates self, 1 and up indicates children (*mechanism_index* = 1 reference mech.children[0]).

In Mechanism:

```
def __init__(self):
    ...
    self.tree = [self] + self.children
```

`def lookup_joint_ref(self, joint_ref):` - converts named indices to all ints.

`def joint(joint_ref):` - returns a joint given a **JointRef**

```
    m_i,b_i,j_i = self.lookup_joint_ref(joint_ref)
    return self.tree[m_i][b_i].joints[j_i]
```

Constraint

(*type*(str), *ref*, *value*)

By default, a Mechanism will constrain Body 0 to have ORIGIN_POSE, and add a joint constraint for all connections when calling solved()

type is ‘body’ - set body to world pose

- *ref* is
  - **JointRef** - use mechanism index and body index to select body
  - int - the index of the desired body in the current mechanism
  - str - name of Body to constrain
- *value* is a Pose, or ‘last’ to indicate last known pose

type is ‘state’ - set angle between joints to state

- *ref* is a connection index
- *value* is a float angle to set rotation around first **Joint**’s Z axis to second **Joint**

type is ‘joint’ - set joint to world pose

- *ref* is **JointRef** - indicates **Joint** in **Mechanism**
- *value* is Pose

Basic Mechanism

OT = (0,0,0), OQ = (0,0,0,1)

OP = ORIGIN_POSE = (OT,OQ)

mechanism_0

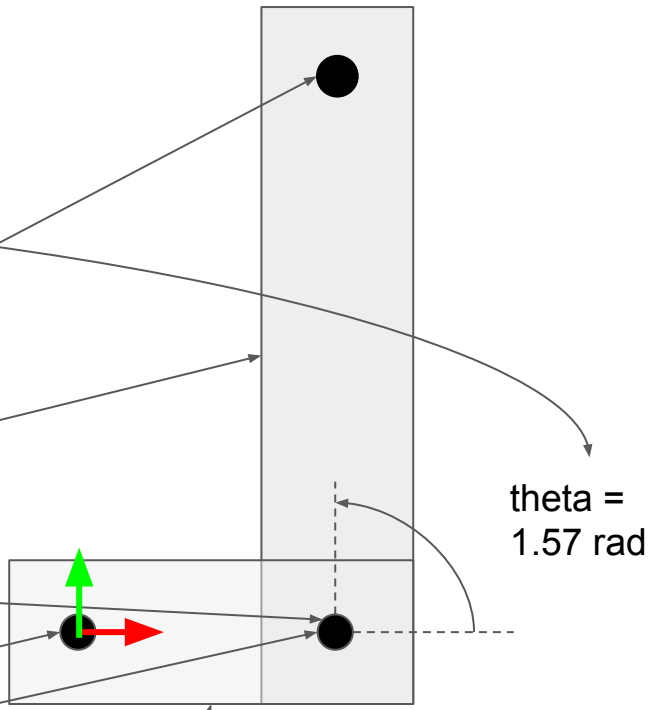
- elts: [body_0,body_1]
- connections: [((0,0,1),(0,1,0))]
- constraints: [('body', 0, OP)]
- states: [1.57]
- children: []

body_1

- joints: [OP, ((20,0,0),OQ)]

body_0

- joints: [OP, ((10,0,0),OQ)]



Compound Mechanism

mechanism_0

- elts: [body_0, body_1]
- connections: [((0,0,1),(0,1,0)), ((0,0,0),(1,0,0))]
- constraints: [('body', 0, OP)]
- states: [1.57, -1.57]
- children: [mechanism_1]

mechanism_1

- elts: [body_2]
- connections: []
- constraints: []
- states: []
- children: []