Daily Assignment 22

- To compare 4 orientation interpolation methods, implement following functions:

- *exp & log functions*

- **exp(rv)**
  - Converts a rotation vector to a rotation matrix
  - You can use Rodrigues' rotation formula or the method in Lecture 20
  - Returns a rotation matrix

- **log(R)**
  - Converts a rotation matrix to a rotation vector
  - You can use the method in today's lecture
  - Returns a rotation vector (the length of the vector is rotation angle)

Daily Assignment 22

- *Interpolation functions:*
- **slerp(R1, R2, t)**
  - R1 & R2: rotation matrices for start & end orientations
- **interpolateRotVec(rv1, rv2, t)**
  - rv1 & rv2: rotation vectors for start & end orientations
- **interpolateZYXEulerAngles(euler1, euler2, t)**
  - euler1 & euler2: tuples of ZYX Euler angles for start & end orientations (euler1[0] – xang, euler1[1] – yang, euler1[2] – zang)
- **interpolateRotMat(R1, R2, t)**
  - R1 & R2: rotation matrices for start & end orientations
- *For all interpolation functions:*
  - All interpolation functions return a rotation matrix
  - The parameter t ranges from 0.0 to 1.0

Daily Assignment 22

- Start from Lecture 17 code,
  - Add functions in *22-addcode.py*
  - Replace *render(), key_callback()* by those in *22-replacecode.py*

- You will need to use
  - The given **lerp()** for **interpolateRotVec(), interpolateZYXEuler(), interpolateRotMat()**
  - Your **exp(), log()** implementation for **slerp(), interpolateRotVec()**

- Program usage:
  - When the program is run, only **slerp()** result is visible
  - A key: Toggle **slerp()** result
  - S key: Toggle **interpolateRotVec()** result
  - D key: Toggle **interpolateZYXEuler()** result
  - F key: Toggle **interpolateRotMat()** result
  - Z key: Hide all results
  - X key: Show all results

```
def exp(rv):
    theta = l2norm(rv)
    axis = normalized(rv)
    p = np.cross(axis, np.array([0.,0.,1.]))
    Raz = np.column_stack((np.array([0.,0.,1.]), p, np.cross(p,
np.array([0.,0.,1.])))) @ np.linalg.inv(np.column_stack((axis, p, np.cross(p,
axis))))
    Rz = np.array([[np.cos(theta), -np.sin(theta), 0],
                   [np.sin(theta), np.cos(theta), 0],
                   [0,0,1]])
    R = np.linalg.inv(Raz) @ Rz @ Raz
    return R

def log(R):
    th = np.arccos((R.trace()-1)*.5)
    v = np.zeros(3)
    v[0] = (R[2,1]-R[1,2])/(2*np.sin(th))
    v[1] = (R[0,2]-R[2,0])/(2*np.sin(th))
    v[2] = (R[1,0]-R[0,1])/(2*np.sin(th))
    return th*v

def slerp(R1, R2, t):
    return R1 @ exp(t*log(R1.T @ R2))

def interpolateRotVec(rv1, rv2, t):
    return exp(lerp(rv1, rv2, t))

def interpolateZYXEuler(euler1, euler2, t):
    return ZYXEulerToRotMat(lerp(euler1, euler2, t))

def interpolateRotMat(R1, R2, t):
    return lerp(R1, R2, t)
```