AutoRob

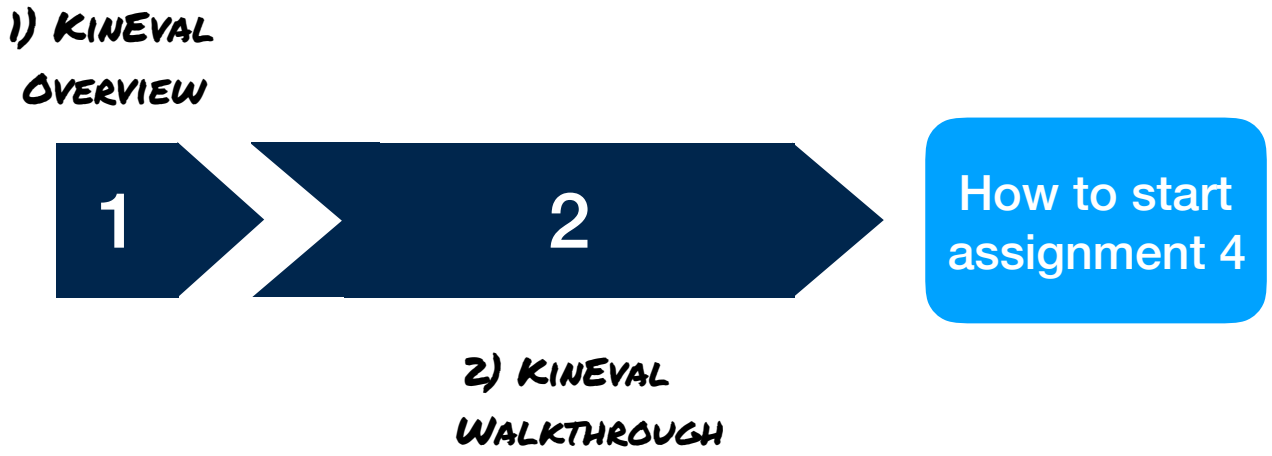
Introduction to Autonomous Robotics
Michigan EECS 367

Robot Kinematics and Dynamics
Michigan ME 567 EECS 567 ROB 510

Fall 2019

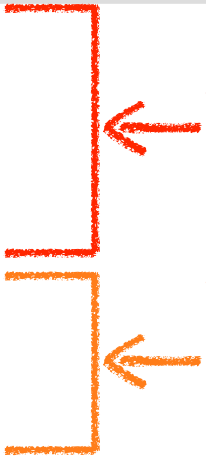
EECS 367 Lab:
KinEval pose parameters and HTML5 audio

Lab Takeaways



Forward Kinematics Overview

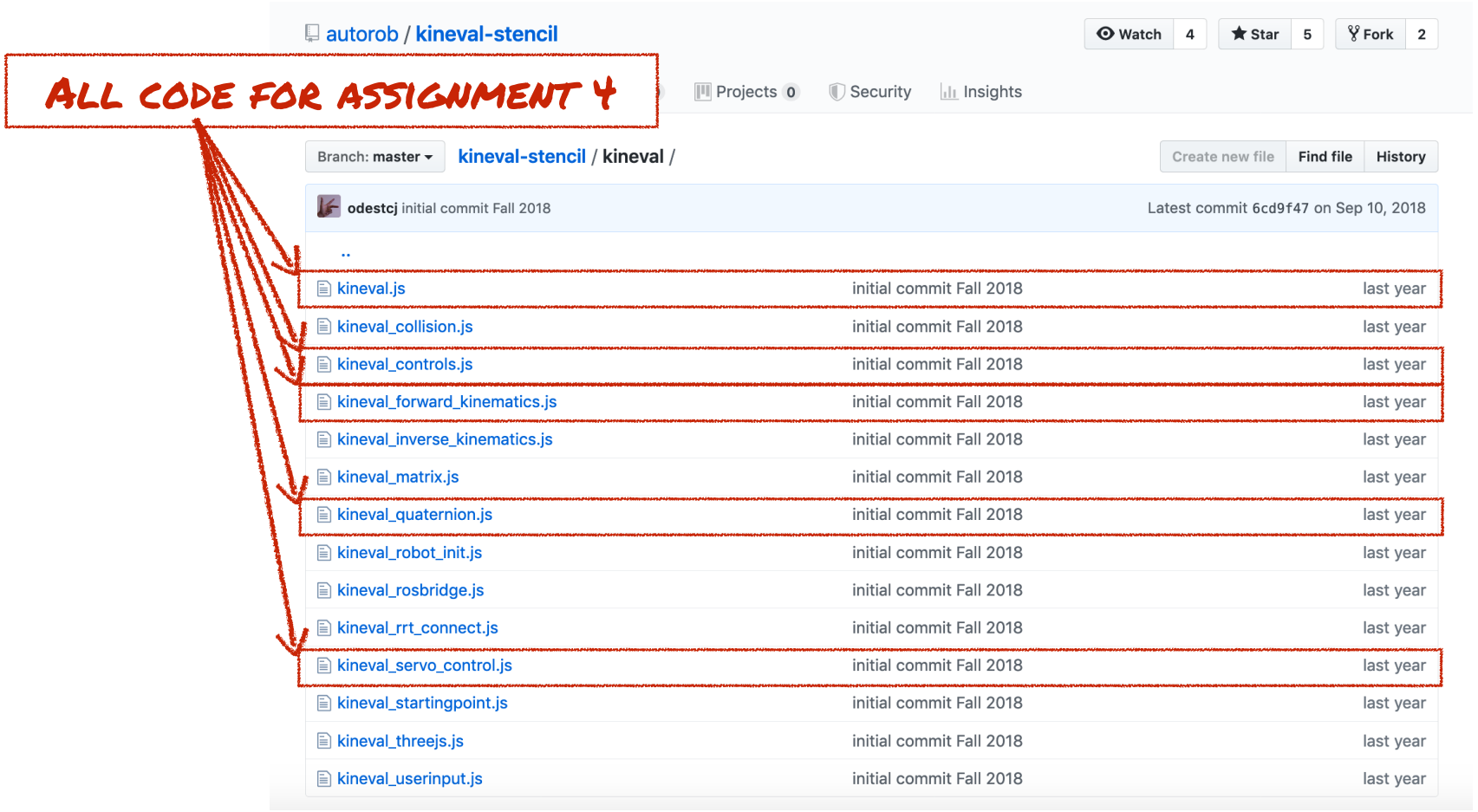
Assignment 4: Dance Controller		
6	All	Quaternion joint rotation
2	All	Interactive base control
2	All	Pose setpoint controller
2	All	Dance FSM
2	Grad	Joint limits
2	Grad	Prismatic joints
2	Grad	Fetch rosbridge interface



Features assigned to all sections

Features assigned to graduate sections

KinEval Overview



kineval_forward_kinematics.js

FOR EACH JOINT, INCORPORATE
`joint.axis` and `joint.angle`
within forward kinematics
(you'll then be able to control joints)

kineval_forward_kinematics.js

```
/*-- |\/| |\/| |\/| |\/| |\/| |\/| |\/| |\/| |\/| |\/| |\/| |\/| |\/| |\/| |\/|

    KinEval | Kinematic Evaluator | forward kinematics

    ot kinematics, control, decision making, and dynamics
    ot and threejs

    tps://github.com/ohseejay / https://bitbucket.org/ohseejay


    tion RObotics and Grounded REasoning Systems


    mons 3.0 BY-SA

|\/| |\/| |\/| |\/| |\/| |\/| |\/| |\/| |\/| |\/| |\/| |\/| |\/| |\/| |\/| |\/| --*/

kineval.robotForwardKinematics = function robotForwardKinematics () {

    if (typeof kineval.buildFKTransforms === 'undefined') {
        textbar.innerHTML = "forward kinematics not implemented";
        return;
    }

    // STENCIL: implement kineval.buildFKTransforms();

}

    // STENCIL: reference code alternates recursive traversal over
    //   links and joints starting from base, using following functions:
    //     traverseFKBase
    //     traverseFKLink
    //     traverseFKJoint
    //
    // user interface needs the heading (z-axis) and lateral (x-axis) directions
    //   of robot base in world coordinates stored as 4x1 matrices in
    //   global variables "robot_heading" and "robot_lateral"
    //
    // if geometries are imported and using ROS coordinates (e.g., fetch),
    //   coordinate conversion is needed for kineval/threejs coordinates:
    //
```

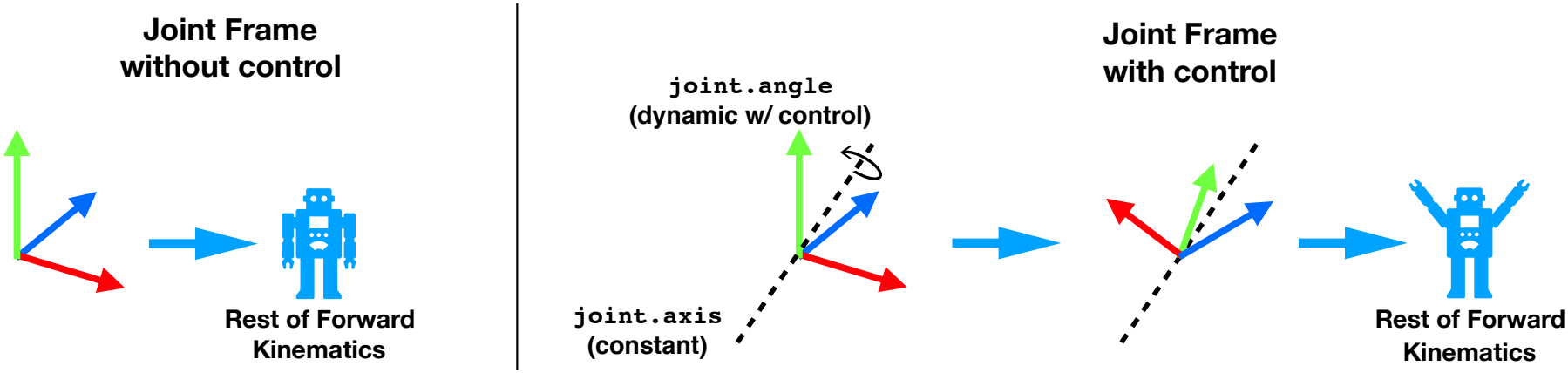
kineval_quaternion.js

kineval_quaternion.js

```
1  ////////////////////////////////////////////////
2  /////     QUATERNION TRANSFORM ROUTINES
3  ////////////////////////////////////////////////
4
5     // STENCIL: reference quaternion code has the following functions:
6     //   quaternion_from_axisangle
7     //   quaternion_normalize
8     //   quaternion_to_rotation_matrix
```

DEFINE QUATERNION HELPER FUNCTIONS

WITH THESE FUNCTIONS YOU'LL BE ABLE TO CREATE A JOINT'S ROTATION MATRIX ABOUT ANY AXIS-ANGLE PAIR



kineval_control.js

CONTROL APPLIED TO joint.angle(s) AND robot.origin POSE FOR YOU

kineval_controls.js

```
18
19  kineval.applyControls = function robot_apply_controls() {
20      // apply robot controls to robot kinematics transforms and joint angles, then zero controls
21      // includes update of camera position based on base movement
22
23      // update robot configuration from controls
24      for (x in robot.joints) {
25          if (isNaN(robot.joints[x].control))
26              console.warn("kineval: control value for " + x +" is a nan"); //+robot.joints[x].control);
27
28          // update joint angles
29          robot.joints[x].angle += robot.joints[x].control;
30
31      // STENCIL: enforce joint limits for prismatic and revolute joints
32
33
34          // clear controls back to zero for next timestep
35          robot.joints[x].control = 0;
36      }
37
38      // base motion
39      robot.origin.xyz[0] += robot.control.xyz[0];
40      robot.origin.xyz[1] += robot.control.xyz[1];
41      robot.origin.xyz[2] += robot.control.xyz[2];
42      robot.origin.rpy[0] += robot.control.rpy[0];
43      robot.origin.rpy[1] += robot.control.rpy[1];
44      robot.origin.rpy[2] += robot.control.rpy[2];
45
46      // move camera with robot base
47      camera_controls.object.position.x += robot.control.xyz[0];
48      camera_controls.object.position.y += robot.control.xyz[1];
49      camera_controls.object.position.z += robot.control.xyz[2];
50
51      // zero controls now that they have been applied to robot
52      robot.control = {xyz: [0,0,0], rpy:[0,0,0]};
53  }
54
```

GRAD SECTIONS HAVE STENCIL TO ENFORCE JOINT LIMITS

kineval_servo_control.js

kineval_servo_control.js

```
/*-- |\/| |\/| |\/| |\/| |\/| |\/| |\/| |\/| |\/| |\/| |\/| |\/| |\/| |\/| |\/| |\/|

    KinEval | Kinematic Evaluator | arm servo control

    Implementation of robot kinematics, control, decision making, and dynamics
        in HTML5/JavaScript and threejs

    @author ohseejay / https://github.com/ohseejay / https://bitbucket.org/ohseejay

                                                    or Perception RObotics and Grounded REa
                                                    f Michigan

                                                    ative Commons 3.0 BY-SA

                                                    \/| |\/| |\/| |\/| |\/| |\/| |\/| |\/|

                                                    DanceSequence = function execute_setpoi

    // if update not requested, exit routine
    if (!kineval.params.update_pd_dance) return;

    // STENCIL: implement FSM to cycle through dance po
}

kineval.setpointClockMovement = function execute_clock(

    // if update not requested, exit routine
    if (!kineval.params.update_pd_clock) return;

    var curdate = new Date();
    for (x in robot.joints) {
        kineval.params.setpoint_target[x] = curdate.getSeconds()/60*2*Math.PI;
    }
}


kineval.robotArmControllerSetpoint = function robot_pd_control () {

    // if update not requested, exit routine
    if ((!kineval.params.update_pd)&&(!kineval.params.persist_pd)) return;

    kineval.params.update_pd = false; // if update requested, clear request and process setpoint control

    // STENCIL: implement P servo controller over joints
}
```

Implement a Finite State Machine for setpoint dance routine

Thought Experiment

1)Why are we only asking for a P controller?

2)What would control look like with a PID controller?

3)What about a PD controller?

Implement P controller for joint control w.r.t. setpoints

kineval.js

kineval.js

```
292     // initialize pose setpoints and target setpoint
293     kineval.setpoints = [];
294     kineval.params.setpoint_target = {};
295     for (var i=0;i<10;i++) {  // 10 is the number of slots for pose setpoints
296         kineval.setpoints[i] = {};
297         for (x in robot.joints) {
298             kineval.params.setpoint_target[x] = 0;  // current setpoint target
299             kineval.setpoints[i][x] = 0;  // slot i setpoint
300         }
301     }
302
303     kineval.params.dance_pose_index = 0;
304     kineval.params.dance_sequence_index = [0,1,2,3,4,5,6,7,8,9];
305     if (robot.name === 'fetch') {  // fetch easter egg
306         kineval.params.dance_sequence_index = [1,2,1,2,1,0,3,0,3,0];
307         kineval.setpoints =
308             [{"torso_lift_joint":0,"shoulder_pan_joint":0,"shoulder_lift_joint":0,"upperarm_roll_joint":0,"elbow_flex_joint":0,"
309     }
310
311     // initialize inverse kinematics target location
312     // KE 3 : ik_target param is redundant as an argument into inverseKinematics
313     kineval.params.ik_target = {};
314     kineval.params.ik_target.position = [[0],[0.8],[1.0],[1]];
315     kineval.params.ik_target.orientation = [Math.PI/6, Math.PI/4, 0];
316     kineval.params.ik_orientation_included = false;
317     kineval.params.ik_steplength = 0.1;
318     kineval.params.ik_pseudoinverse = false;
```

CREATE A COOL DANCE ROUTINE BY DEFINING A SEQUENCE OF JOINT ANGLE SETPOINTS TO BE USED BY THE FSM IMPLEMENTATION

Interactive controls:
poses for servo can be set interactively in KinEval using [0-9] keys and Shift+[0-9]

JSON.stringify(kineval.setpoints) will output the currently available servo setpoints to the console as a string

HTML5 Audio

- With two small additions to the stencil code, you can add music for your dance routine!
  - The audio element offered by HTML5
    - With this, we can load a song in `home.html`
    - Then our FSM can play/pause the song along with the dance

home.html

```
123  // STENCIL: my_animate is where your robot's controls and movement are updated over time
124  function my_init() {
125
126      // Adding music for the dance FSM
127      // The song I have chosen is 'Wave' by Antonio Carlos Jobim
128      // My dance waves, but does not necessarily coincide with the beat of the song
129      song = document.createElement("audio");
130      song.src = "music/Wave.mp3"
131
132      startingPlaceholderInit();  // a quick and dirty JavaScript tutorial
133
134  }
```

kineval_servo_control.js

```
19   kineval.setpointDanceSequence = function execute_setpoints() {
20
21       // if update not requested, exit routine
22       if (!kineval.params.update_pd_dance) {
23           song.pause();
24           return;
25       }
26
27
28       // STENCIL: implement FSM to cycle through dance pose setpoints
29       if (song.paused) {
30           song.load();
31           song.play();
32       }
```