EECS 367 Lab

RRT-Connect in Search Canvas and KinEval

Administrative

- Assignments 6 and 7 released
  - Assignment 6 due Monday, December 7 at 11:59pm
  - Assignment 7 video recording due Friday, December 4 at 11:59pm
- **Pull stencil from upstream!**
- No new lab material after Thanksgiving break
  - Extended office hours on December 4

Lab Takeaways

1. Revisit the search canvas
2. KinEval overview
3. KinEval walkthrough

→ How to start Assignment 6

Motion Planning Overview

Assignment 6: Motion Planning		
4	All	Collision detection
2	All	2D RRT-Connect
6	All	Configuration space RRT-Connect
6	Grad	2D RRT-Star

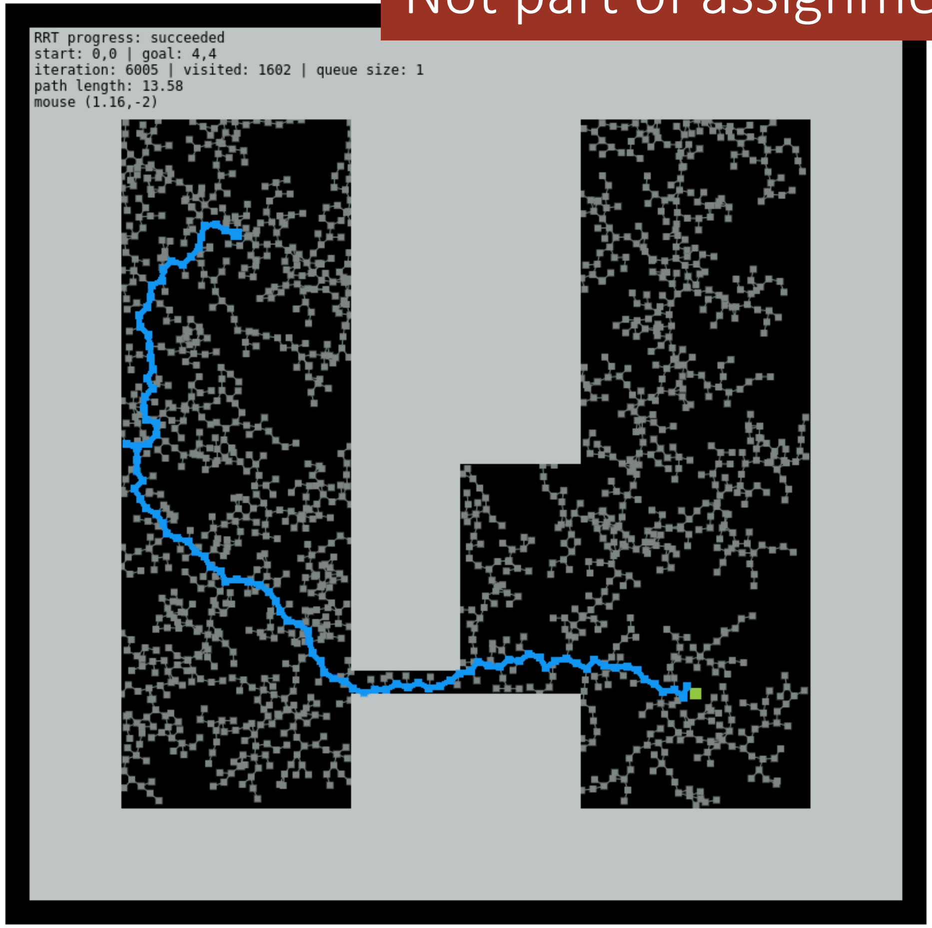
Features assigned to all sections

Feature assigned to grad section only

Rapidly-Exploring Random Trees (RRT)

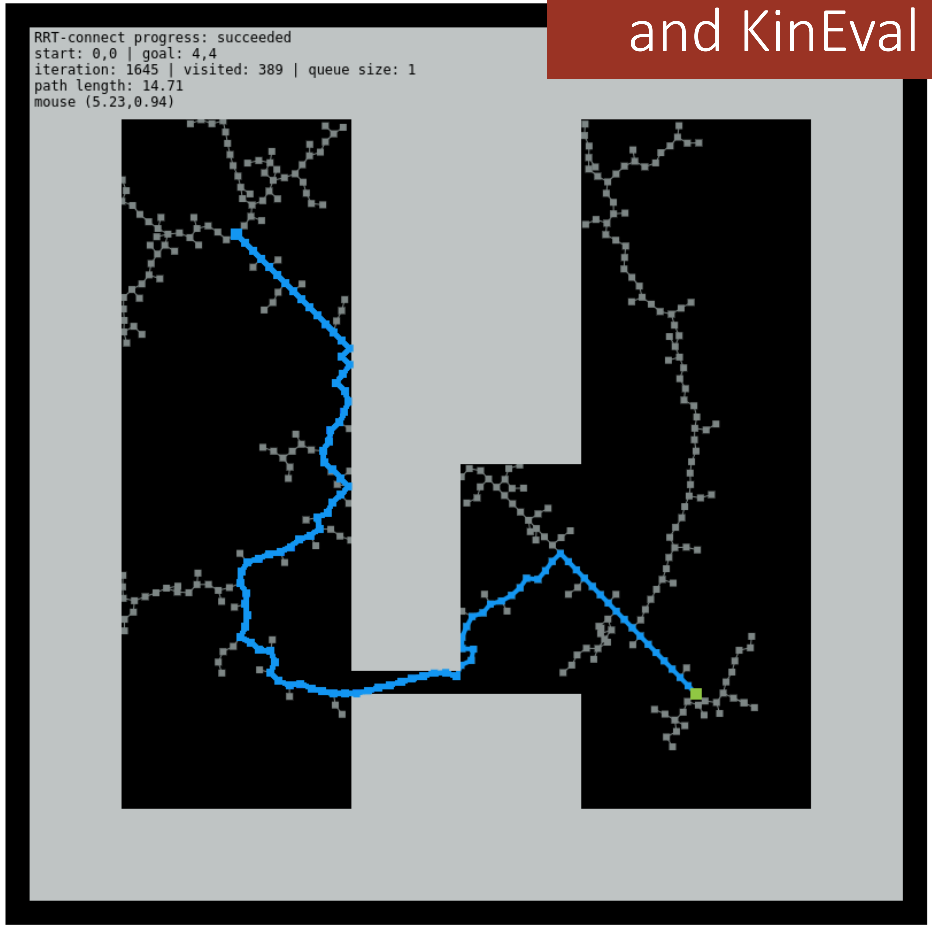
RRT

Not part of assignment

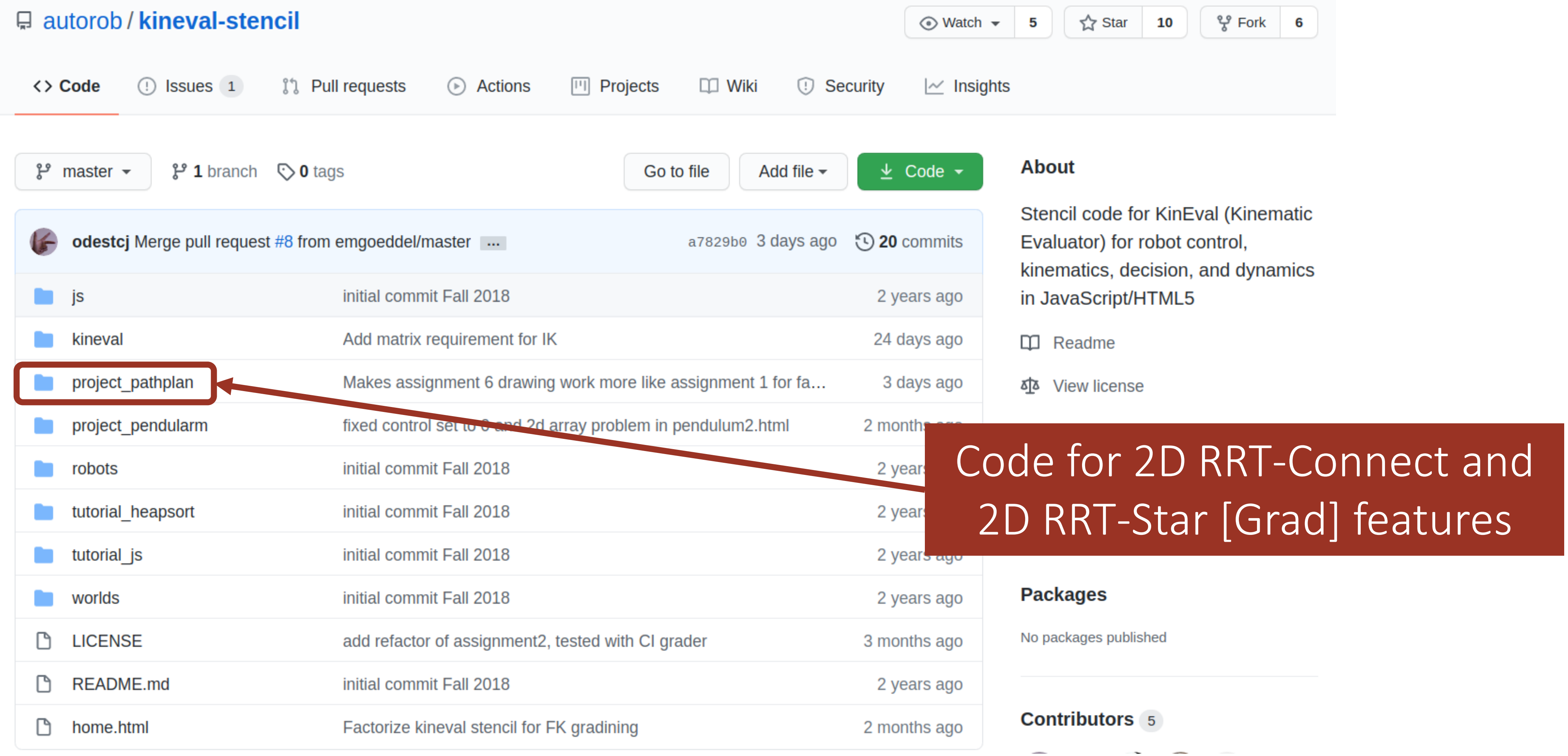


RRT-CONNECT

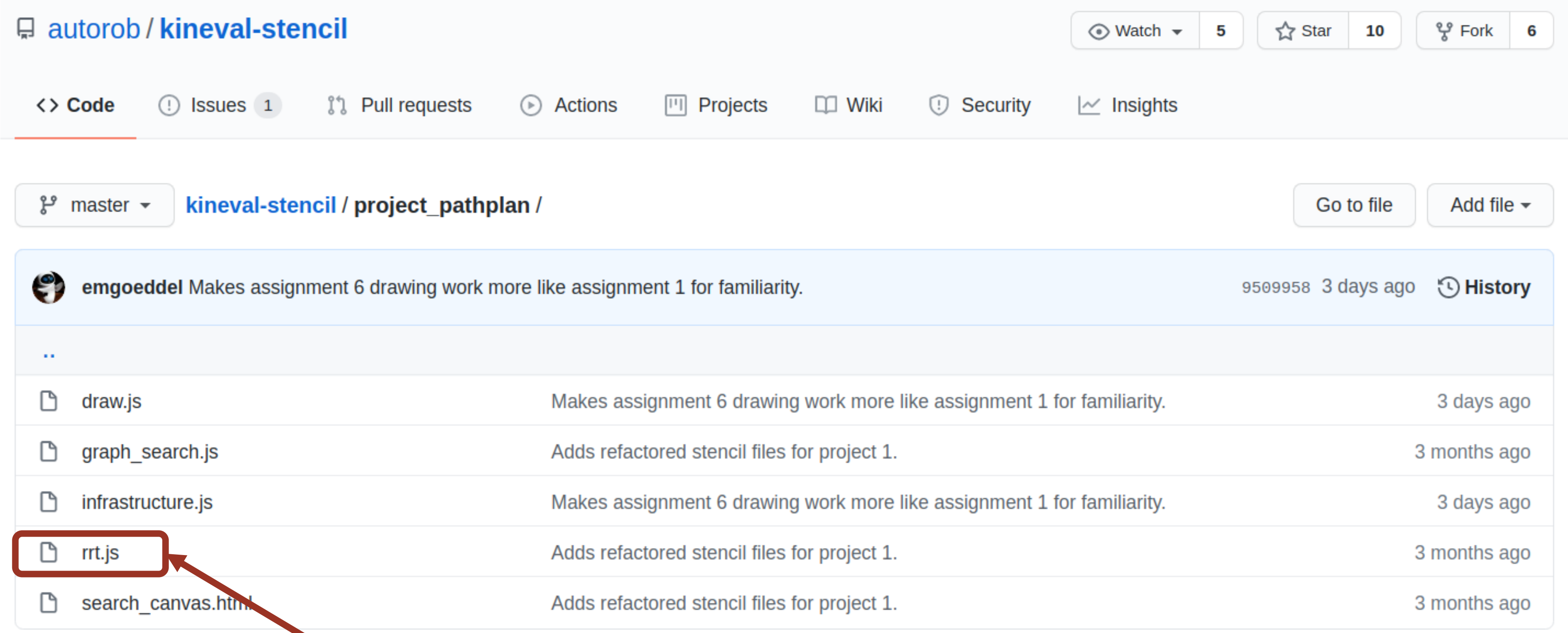
Will implement for 2D and KinEval robots



Revisiting the Search Canvas



Revisiting the Search Canvas



Code for 2D RRT-Connect and 2D RRT-Star [Grad] features

2D RRT-Connect

Recommended: Start project with 2D RRT-Connect in project_pathplan/rrt.js

rrt.js

```
function iterateRRTConnect() {


    // STENCIL: implement a single iteration of an RRT-Connect algorithm.
    //   An asynch timing mechanism is used instead of a for loop to avoid
    //   blocking and non-responsiveness in the browser.
    //
    //   Return "failed" if the search fails on this iteration.
    //   Return "succeeded" if the search succeeds on this iteration.
    //   Return "extended" otherwise.
    //
    //   Provided support functions:
    //
    //   testCollision - returns whether a given configuration is in collision
    //   insertTreeVertex - adds and displays new configuration vertex for a tree
    //   insertTreeEdge - adds and displays new tree edge between configurations
    //   drawHighlightedPath - renders a highlighted path in a tree
}
```

2D RRT-Connect

Similar to Assignment 1 search algorithms, implement as a single step within the iterative algorithm

rrt.js

```
function iterateRRTConnect() {


    // STENCIL: implement a single iteration of an RRT-Connect algorithm.
    //   An asynch timing mechanism is used instead of a for loop to avoid
    //   blocking and non-responsiveness in the browser.
    //
    //   Return "failed" if the search fails on this iteration.
    //   Return "succeeded" if the search succeeds on this iteration.
    //   Return "extended" otherwise.
    //
    //   Provided support functions:
    //
    //   testCollision - returns whether a given configuration is in collision
    //   insertTreeVertex - adds and displays new configuration vertex for a tree
    //   insertTreeEdge - adds and displays new tree edge between configurations
    //   drawHighlightedPath - renders a highlighted path in a tree
}
```

RRT Data Structure

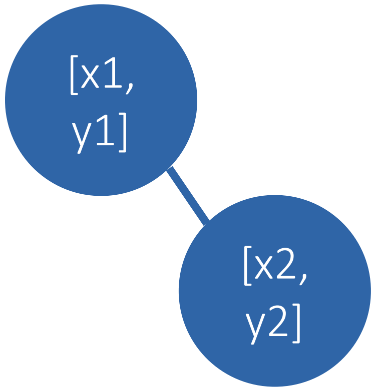
infrastructure.js

```
function initRRT(q) {

    // create tree object
    var tree = {};

    // initialize with vertex for given configuration
    tree.vertices = [];
    tree.vertices[0] = {};
    tree.vertices[0].vertex = q;
    tree.vertices[0].edges = [];

    // maintain index of newest vertex added to tree
    tree.newest = 0;

    return tree;
}

function insertTreeVertex(tree,q) {

```

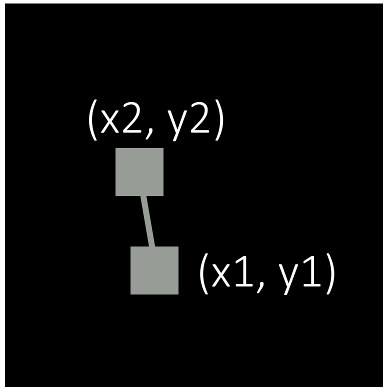
Tree implemented as a JavaScript object with array of vertices

```
tree = {"vertices":
            [
             {"vertex":[x1,y1],
              "edges":[tree.vertices[1]]},
             {"vertex":[x2,y2],
              "edges":[tree.vertices[0]]}
            ],
        "newest":1}
```

Data structure view



Search canvas view



RRT Functions

infrastructure.js

```
function initRRT(q) {

    // create tree object
    var tree = {};

    // initialize with vertex for given configuration
    tree.vertices = [];
    tree.vertices[0] = {};
    tree.vertices[0].vertex = q;
    tree.vertices[0].edges = [];

    // maintain index of newest vertex added to tree
    tree.newest = 0;

    return tree;
}

function insertTreeVertex(tree,q) {

```

Helper functions available for basic tree operations

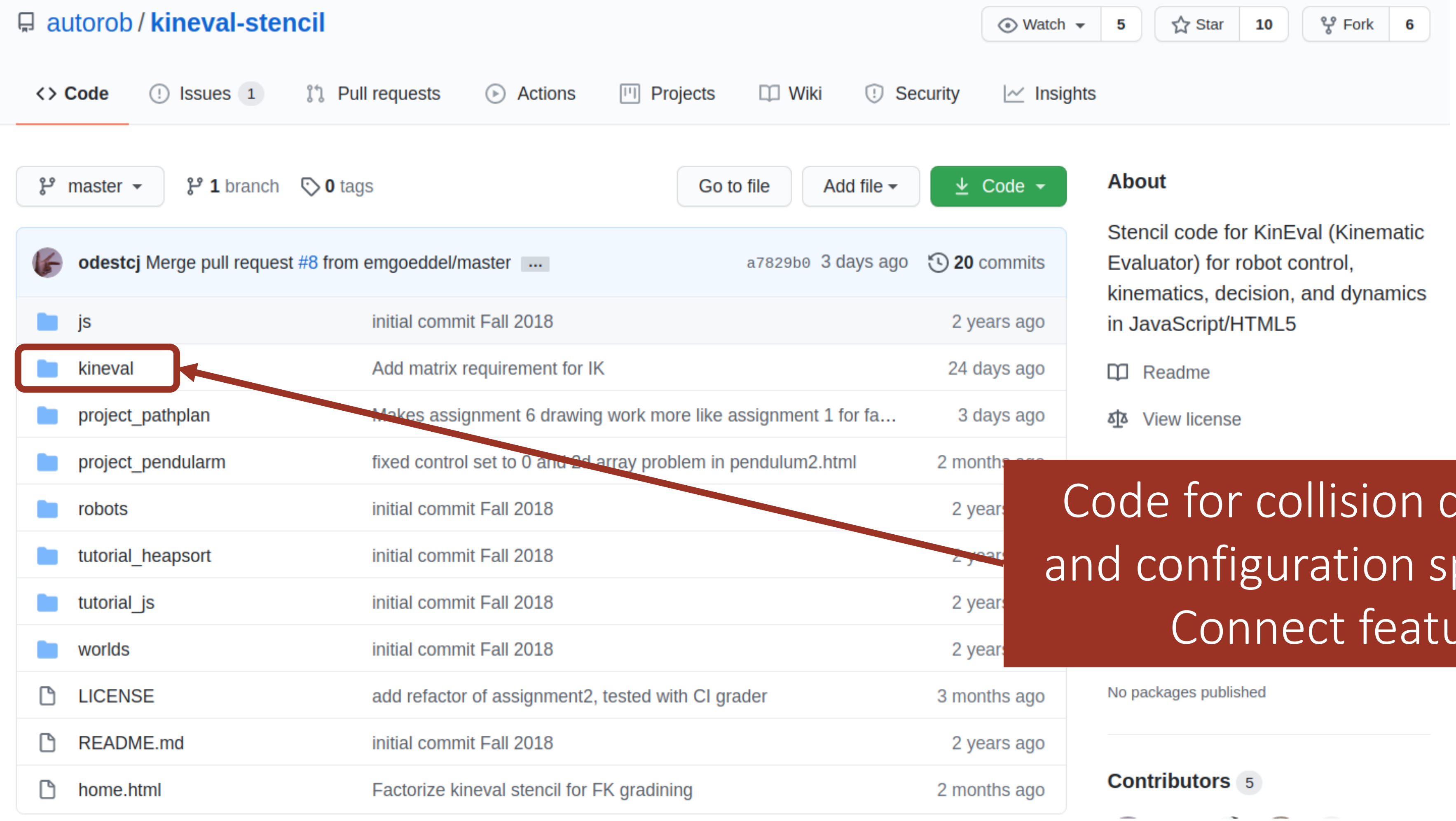
rrt.js

```
/////////////////////////////////////////////////////
/////      RRT IMPLEMENTATION FUNCTIONS
/////////////////////////////////////////////////////

    // STENCIL: implement RRT-Connect functions here, such as:
    //    extendRRT
    //    connectRRT
    //    randomConfig
    //    newConfig
    //    findNearestNeighbor
    //    dfsPath
```

Suggested functions for you to add for RRT-Connect implementation

Demo

KinEval Overview



Code for collision detection and configuration space RRT-Connect features

KinEval Overview

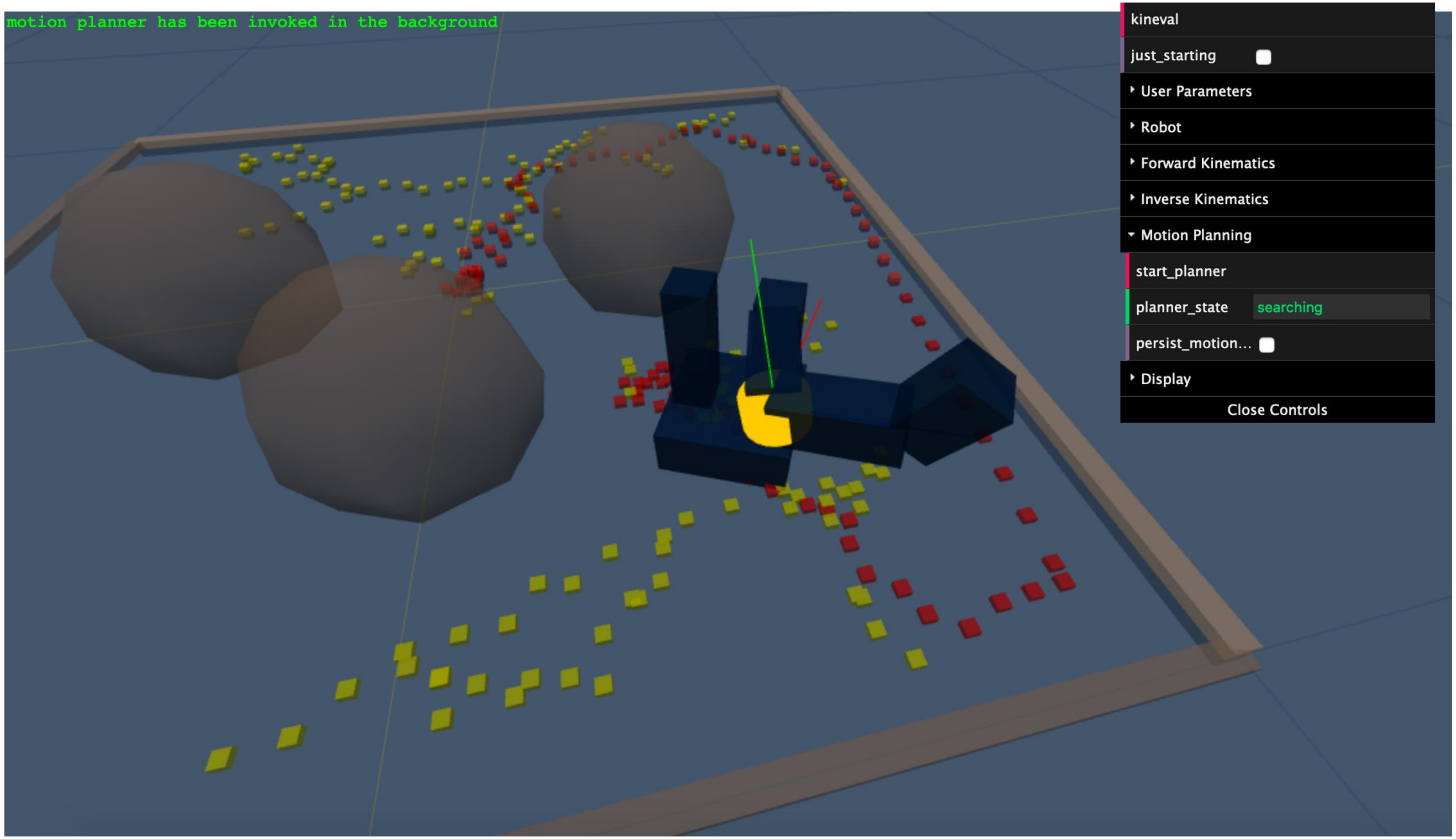
master | kineval-stencil / kineval / | Go to file | Add file

zhezhou1993 Add matrix requirement for IK | 537eeac 24 days ago | History

..		
kineval.js	initial commit Fall 2018	2 years ago
kineval_collision.js	initial commit Fall 2018	2 years ago
kineval_controls.js	initial commit Fall 2018	2 years ago
kineval_forward_kinematics.js	initial commit Fall 2018	2 years ago
kineval_inverse_kinematics.js	Add matrix requirement for IK	24 days ago
kineval_matrix.js	Factorize kineval stencil for FK problems, fix bugs in previous version	2 months ago
kineval_quaternion.js	Factorize kineval stencil for FK problems, fix bugs in previous version	2 months ago
kineval_robot_init.js	Factorize kineval stencil for FK gradining	
kineval_robot_init_joints.js	Factorize kineval stencil for FK gradining	
kineval_rosbridge.js	initial commit Fall 2018	
kineval_rrt_connect.js	initial commit Fall 2018	
kineval_servo_control.js	initial commit Fall 2018	
kineval_startingpoint.js	initial commit Fall 2018	2 years ago
kineval_threejs.js	initial commit Fall 2018	2 years ago
kineval_userinput.js	initial commit Fall 2018	2 years ago

Code for collision detection and configuration space RRT-Connect features

Configuration Space RRT



kineval_collision.js

kineval_collision.js

```
kineval.robotIsCollision = function robot_iscollision() {
    // test whether geometry of current configuration of robot is in collision with planning world

    // form configuration from base location and joint angles
    var q_robot_config = [
        robot.origin.xyz[0],
        robot.origin.xyz[1],
        robot.origin.xyz[2],
        robot.origin.rpy[0],
        robot.origin.rpy[1],
        robot.origin.rpy[2]
    ];

    q_names = {};  // store mapping between joint names and q DOFs

    for (x in robot.joints) {
        q_names[x] = q_robot_config.length;
        q_robot_config = q_robot_config.concat(robot.joints[x].angle);
    }

    // test for collision and change base color based on the result
    collision_result = kineval.poseIsCollision(q_robot_config);

    robot.collision = collision_result;
}
```

`q_robot_config` is an array representing current pose as point within configuration space

Dimension of configuration space is a function of the specific robot!

kineval_collision.js

kineval_collision.js

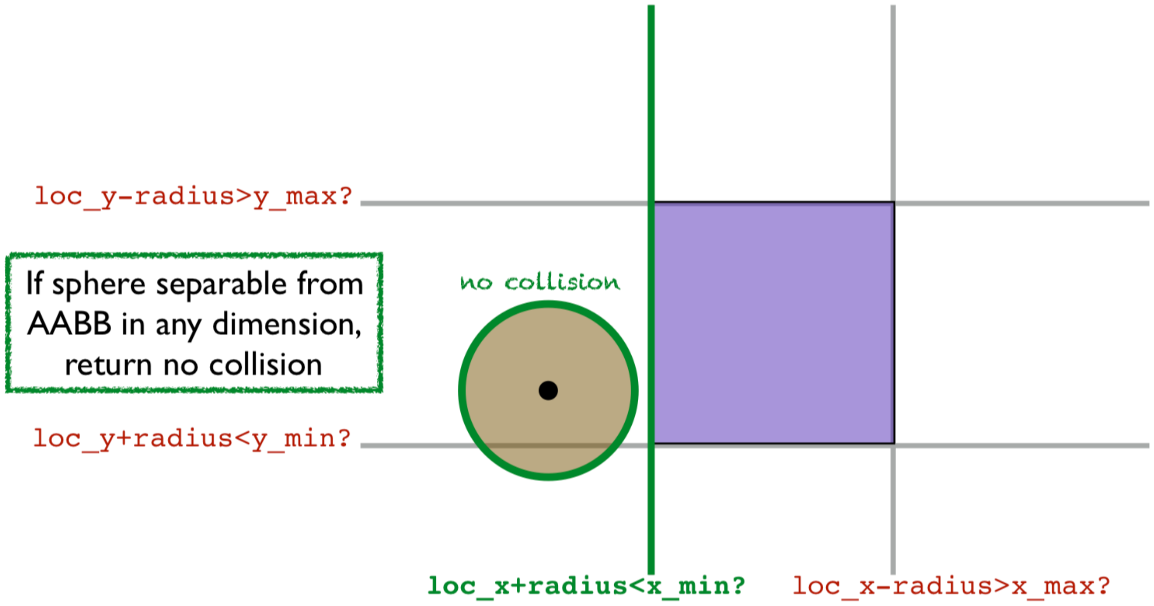
```
kineval.poseIsCollision = function robot_collision_test(q) {
    // perform collision test of robot geometry against planning world

    // test base origin (not extents) against world boundary extents
    if ((q[0]<robot_boundary[0][0])||(q[0]>robot_boundary[1][0])||(q[2]<robot_boundary[0][2])||(q[2]>robot_boundary[1][2]))
        return robot.base;

    // traverse robot kinematics to test each body for collision
    // STENCIL: implement forward kinematics for collision detection
    //return robot_collision_forward_kinematics(q);

}
```

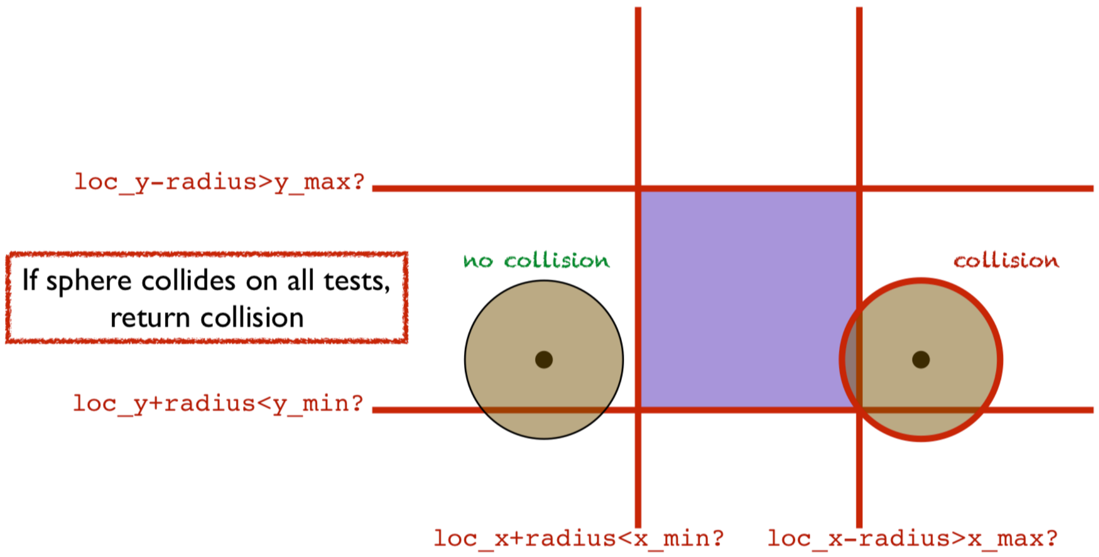
STENCIL: Check each link for collision with spherical obstacles

Collision detection pseudocode:

```
For each link in robot
    For each obstacle in world
        If intersection(link, obstacle)
            Return link is in collision
Return no collision
```

AABB Link Collision Detection





kineval_collision.js

kineval_collision.js

```
function traverse_collision_forward_kinematics_link(link,mstack,q) {

    /* test collision FK
    console.log(link);
    */
    if (typeof link.visual !== 'undefined') {
        var local_link_xform = matrix_multiply(mstack,generate_translation_matrix(link.visual.origin.xyz[0],link
    }
    else {
        var local_link_xform = matrix_multiply(mstack,generate_identity());
    }

    // test collision by transforming obstacles in world to link space
/*
    mstack_inv = matrix_invert_affine(mstack);
*/
    mstack_inv = numeric.inv(mstack);

    var i;
    var j;

    // test each obstacle against link bbox geometry by transforming obstacle into link frame and testing agains
    //for (j=0;j<robot_obstacles.length;j++) {
    for (j in robot_obstacles) {

        var obstacle_local = matrix_multiply(mstack_inv,robot_obstacles[j].location);

        // assume link is in collision as default
        var in_collision = true;
```

Need to compute `mstack` for use here

AABB collision check for a link provided for you in this function, but you need to add the rest of FK traversal

kineval_rrt.js

Implement `robot_rrt_planner_iterate()` as a single iteration of the RRT-Connect planning algorithm

kineval_rrt.js

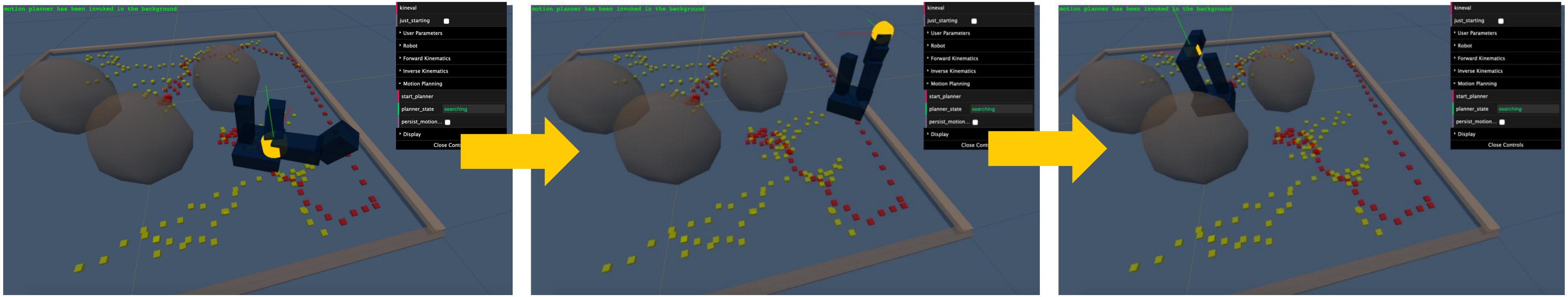
```
function robot_rrt_planner_iterate() {

    var i;
    rrt_alg = 1;  // 0: basic rrt (OPTIONAL), 1: rrt_connect (REQUIRED)

    if (rrt_iterate && (Date.now()-cur_time > 10)) {
        cur_time = Date.now();

    // STENCIL: implement single rrt iteration here. an asynch timing mechanism
    //   is used instead of a for loop to avoid blocking and non-responsiveness
    //   in the browser.
    //
    //   once plan is found, highlight vertices of found path by:
    //     tree.vertices[i].vertex[j].geom.material.color = {r:1,g:0,b:0};
    //
    //   provided support functions:
    //
    //   kineval.poseIsCollision - returns if a configuration is in collision
    //   tree_init - creates a tree of configurations
    //   tree_add_vertex - adds and displays new configuration vertex for a tree
    //   tree_add_edge - adds and displays new tree edge between configurations
    }
```

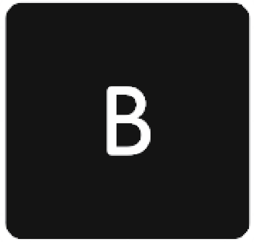
Include any helper functions in this file

Desired Result

Arbitrary initial configuration

Collision free path to home





Backward step
along motion plan

Forward step
along motion plan

Demo

Final Tips

CI grader is a rough guideline, so you must do your own testing to verify correct implementation

Make sure you use the robot's **full configuration space**, including all joint DOFs, and not just base movement in the ground plane

Do not move the robot outside of the X-Z plane

- No translation along Y axis
- No rotation around X or Z axes