EECS 367 & ROB 320 Lab

KinEval RRT Stencil and AABB Collision Detection

Administrative

- Hand grading for assignments 3 and 4
  - Will be pushed to grade.js tomorrow
- Lab section Friday, April 1
  - Connect to Fetch via rosbridge (optional)
  - In robotics building

Assignment 6 Overview

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

Assignment 7 Overview

Assignment 7: The best use of robotics?

Slides due 11:59pm, Friday, April 8, 2022
Presentation due 1:30pm, Monday, April 11, 2022

Scenario: An investor is considering giving you 20 million dollars (cold hard USD cash, figuratively). This investor has been impressed by your work with KinEval and other accomplishments while at the University of Michigan. They are convinced you have the technical ability to make a compelling robot technology... but, they are unsure how this technology could produce something useful. Your task is to make a convincing pitch for a robotics project that would yield a high return on investment, as measured by some metric (financial profit, good for society, creation of new knowledge, etc.).

- Worth 12 points as an assignment

Lab Takeaways

1. RRT connect review
2. 2D RRT connect stencil
3. C-space RRT connect review
4. AABB collision detection

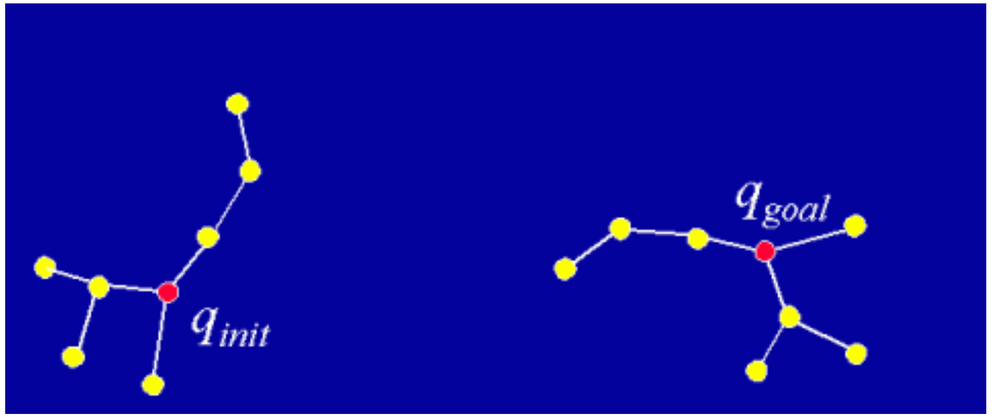
→ How to implement all assignment 6 features

RRT Connect

[Kuffner, LaValle 2000]

0) Use 2 trees (A and B) rooted at start and goal configurations

RRT_CONNECT_PLANNER(q_{init}, q_{goal})
1 $\mathcal{T}_a$.init(q_{init}); $\mathcal{T}_b$.init(q_{goal});
2 **for** $k = 1$ **to** K **do**
3 $q_{rand} \leftarrow$ RANDOM_CONFIG();
4 **if not** (EXTEND($\mathcal{T}_a, q_{rand}$) = *Trapped*) **then**
5 **if** (CONNECT($\mathcal{T}_b, q_{new}$) = *Reached*) **then**
6 Return PATH($\mathcal{T}_a, \mathcal{T}_b$);
7 SWAP($\mathcal{T}_a, \mathcal{T}_b$);
8 Return *Failure*



RRT Connect

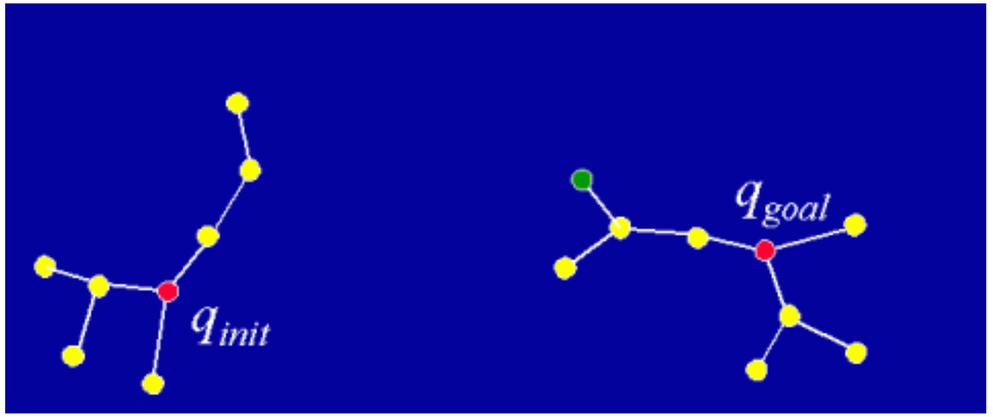
[Kuffner, LaValle 2000]

0) Use 2 trees (A and B) rooted at start and goal configurations

```
RRT_CONNECT_PLANNER(q_init, q_goal)
1  T_a.init(q_init); T_b.init(q_goal);
2  for k = 1 to K do
3      q_rand ← RANDOM_CONFIG();
4      if not (EXTEND(T_a, q_rand) = Trapped) then
5          if (CONNECT(T_b, q_new) = Reached) then
6              Return PATH(T_a, T_b);
7      SWAP(T_a, T_b);
8  Return Failure
```

```
EXTEND(T, q)
1  q_near ← NEAREST_NEIGHBOR(q, T);
2  if NEW_CONFIG(q, q_near, q_new) then
3      T.add_vertex(q_new);
4      T.add_edge(q_near, q_new);
5      if q_new = q then
6          Return Reached;
7      else
8          Return Advanced;
9  Return Trapped;
```

1) Extend tree A towards a random configuration



RRT Connect

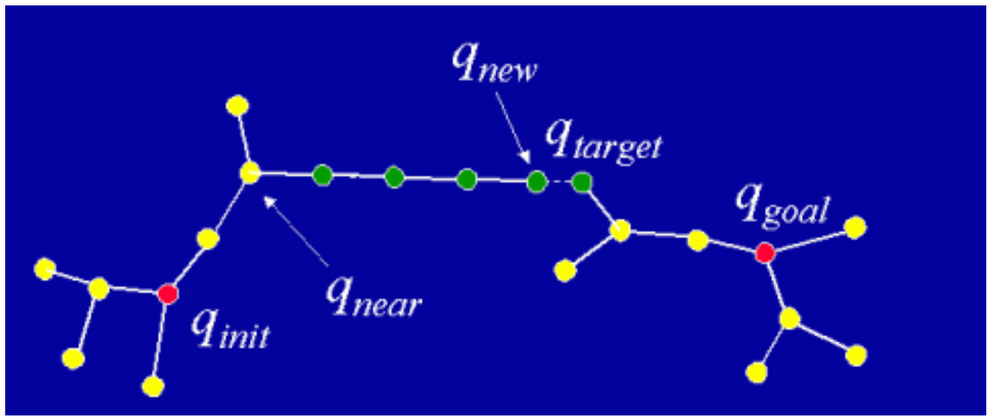
[Kuffner, LaValle 2000]

0) Use 2 trees (A and B) rooted at start and goal configurations

```
RRT_CONNECT_PLANNER(q_init, q_goal)
1  T_a.init(q_init); T_b.init(q_goal);
2  for k = 1 to K do
3      q_rand ← RANDOM_CONFIG();
4      if not (EXTEND(T_a, q_rand) = Trapped) then
5          if (CONNECT(T_b, q_new) = Reached) then
6              Return PATH(T_a, T_b);
7      SWAP(T_a, T_b);
8  Return Failure
```

```
EXTEND(T, q)
1  q_near ← NEAREST_NEIGHBOR(q, T);
2  if NEW_CONFIG(q, q_near, q_new) then
3      T.add_vertex(q_new);
4      T.add_edge(q_near, q_new);
5      if q_new = q then
6          Return Reached;
7      else
8          Return Advanced;
9  Return Trapped;
```

1) Extend tree A towards a random configuration




```
CONNECT(T, q)
1  repeat
2      S ← EXTEND(T, q);
3  until not (S = Advanced)
4  Return S;
```

2) Try to connect tree B to tree A by extending repeatedly from its nearest neighbor

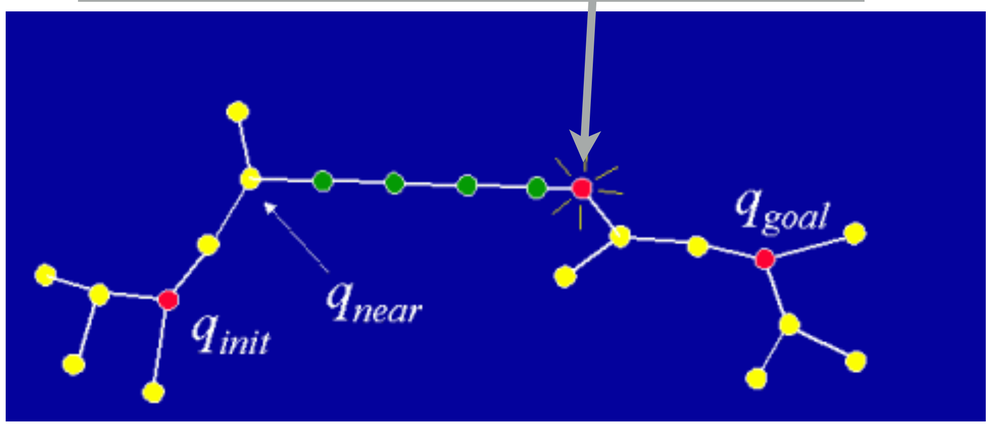
RRT Connect

[Kuffner, LaValle 2000]

0) Use 2 trees (A and B) rooted at start and goal configurations

```
RRT_CONNECT_PLANNER(q_init, q_goal)
1  T_a.init(q_init); T_b.init(q_goal);
2  for k = 1 to K do
3      q_rand ← RANDOM_CONFIG();
4      if not (EXTEND(T_a, q_rand) = Trapped) then
5          if (CONNECT(T_b, q_new) = Reached) then
6              Return PATH(T_a, T_b);
7      SWAP(T_a, T_b);
8  Return Failure
```

search succeeds if trees connect




```
EXTEND(T, q)
1  q_near ← NEAREST_NEIGHBOR(q, T);
2  if NEW_CONFIG(q, q_near, q_new) then
3      T.add_vertex(q_new);
4      T.add_edge(q_near, q_new);
5      if q_new = q then
6          Return Reached;
7      else
8          Return Advanced;
9  Return Trapped;
```

1) Extend tree A towards a random configuration

```
CONNECT(T, q)
1  repeat
2      S ← EXTEND(T, q);
3  until not (S = Advanced)
4  Return S;
```

2) Try to connect tree B to tree A by extending repeatedly from its nearest neighbor

RRT Connect

[Kuffner, LaValle 2000]

0) Use 2 trees (A and B) rooted at start and goal configurations

```
RRT_CONNECT_PLANNER(q_init, q_goal)
1  T_a.init(q_init); T_b.init(q_goal);
2  for k = 1 to K do
3      q_rand ← RANDOM_CONFIG();
4      if not (EXTEND(T_a, q_rand) = Trapped) then
5          if (CONNECT(T_b, q_new) = Reached) then
6              Return PATH(T_a, T_b);
7      SWAP(T_a, T_b);
8  Return Failure
```

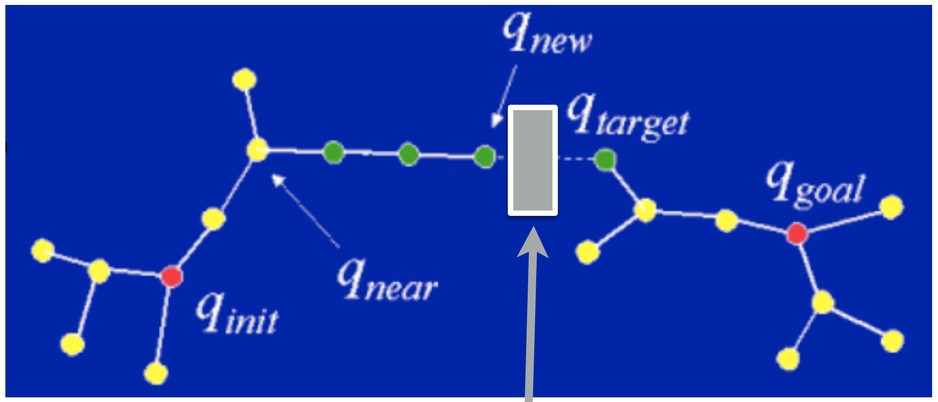
```
EXTEND(T, q)
1  q_near ← NEAREST_NEIGHBOR(q, T);
2  if NEW_CONFIG(q, q_near, q_new) then
3      T.add_vertex(q_new);
4      T.add_edge(q_near, q_new);
5      if q_new = q then
6          Return Reached;
7      else
8          Return Advanced;
9  Return Trapped;
```

1) Extend tree A towards a random configuration

```
CONNECT(T, q)
1  repeat
2      S ← EXTEND(T, q);
3  until not (S = Advanced)
4  Return S;
```

2) Try to connect tree B to tree A by extending repeatedly from its nearest neighbor

3) reverse roles for trees A and B and repeat



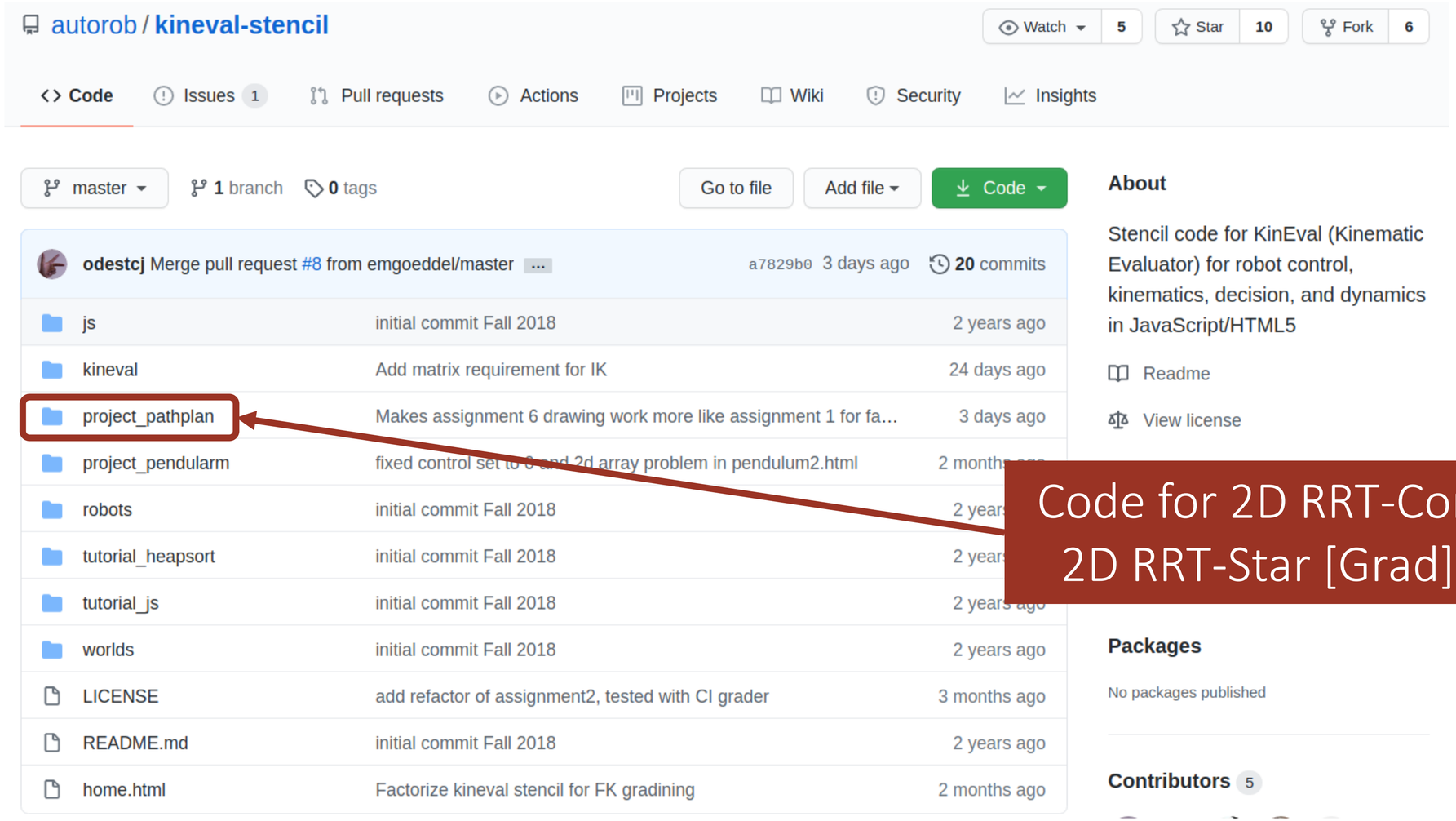
collision encountered

Lab Takeaways

1. RRT connect review
2. 2D RRT connect stencil
3. C-space RRT connect review
4. AABB collision detection

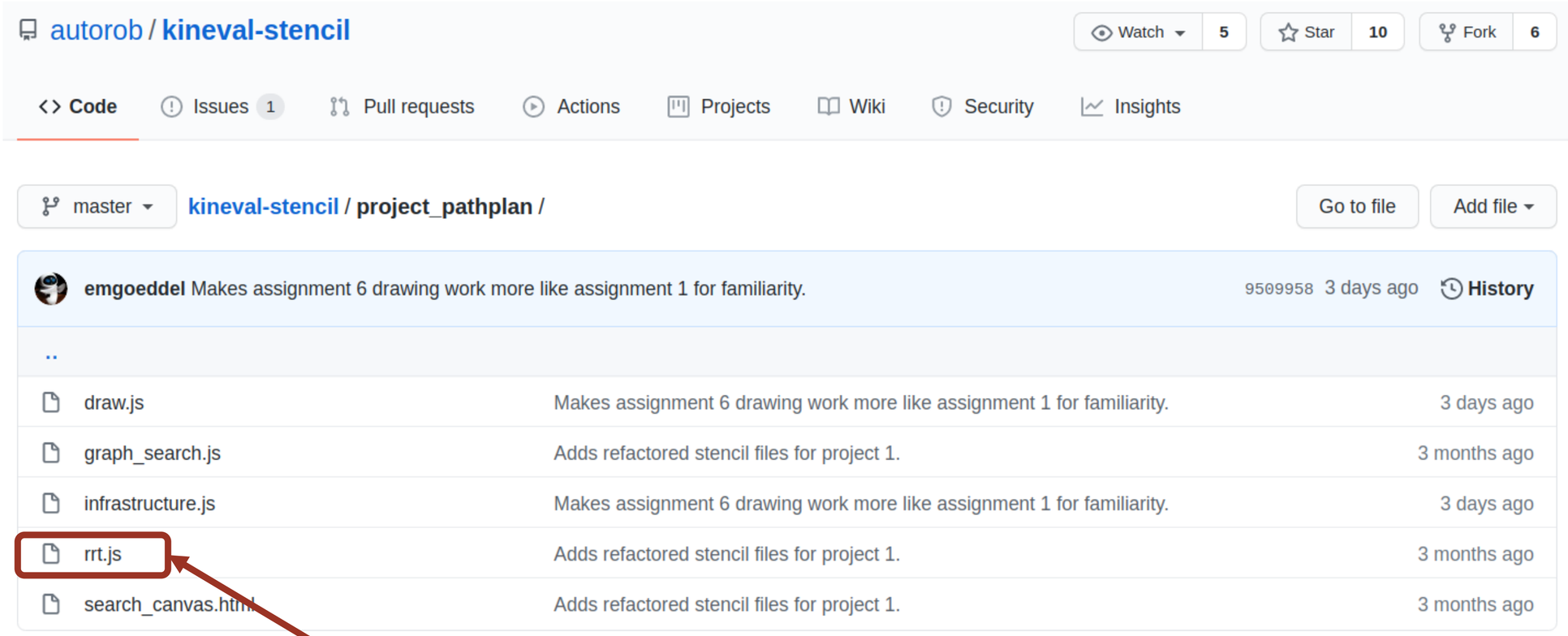
→ How to implement all assignment 6 features

Revisiting the Search Canvas



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

Revisiting the Search Canvas



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

2D RRT-Connect

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

rrt.js

```
45  function iterateRRTConnect() {
46
47
48      // STENCIL: implement a single iteration of an RRT-Connect algorithm.
49      //   An asynch timing mechanism is used instead of a for loop to avoid
50      //   blocking and non-responsiveness in the browser.
51      //
52      //   Return "failed" if the search fails on this iteration.
53      //   Return "succeeded" if the search succeeds on this iteration.
54      //   Return "extended" otherwise.
55      //
56      //   Provided support functions:
57      //
58      //   testCollision - returns whether a given configuration is in collision
59      //   insertTreeVertex - adds and displays new configuration vertex for a tree
60      //   insertTreeEdge - adds and displays new tree edge between configurations
61      //   drawHighlightedPath - renders a highlighted path in a tree
62  }
```

2D RRT-Connect

RRT_CONNECT_PLANNER(q_{init}, q_{goal})

1 $\mathcal{T}_a$.init(q_{init}); $\mathcal{T}_b$.init(q_{goal});
2 **for** $k = 1$ **to** K **do**
3 $q_{rand} \leftarrow$ RANDOM_CONFIG();
4 **if not** (EXTEND($\mathcal{T}_a, q_{rand}$) = *Trapped*) **then**
5 **if** (CONNECT($\mathcal{T}_b, q_{new}$) = *Reached*) **then**
6 Return PATH($\mathcal{T}_a, \mathcal{T}_b$);
7 SWAP($\mathcal{T}_a, \mathcal{T}_b$);
8 Return *Failure*

rrt.js

```
45  function iterateRRTConnect() {
46
47
48      // STENCIL: implement a single iteration of an RRT-Connect algorithm.
49      //   An asynch timing mechanism is used instead of a for loop to avoid
50      //   blocking and non-responsiveness in the browser.
51      //
52      //   Return "failed" if the search fails on this iteration.
53      //   Return "succeeded" if the search succeeds on this iteration.
54      //   Return "extended" otherwise.
55      //
56      //   Provided support functions:
57      //
58      //   testCollision - returns whether a given configuration is in collision
59      //   insertTreeVertex - adds and displays new configuration vertex for a tree
60      //   insertTreeEdge - adds and displays new tree edge between configurations
61      //   drawHighlightedPath - renders a highlighted path in a tree
62  }
```

RRT Data Structure

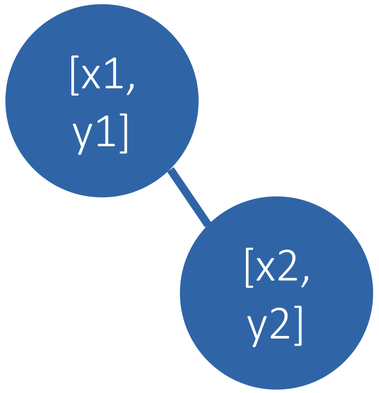
infrastructure.js

```
64  function initRRT(q) {
65
66      // create tree object
67      var tree = {};
68
69      // initialize with vertex for given configuration
70      tree.vertices = [];
71      tree.vertices[0] = {};
72      tree.vertices[0].vertex = q;
73      tree.vertices[0].edges = [];
74
75      // maintain index of newest vertex added to tree
76      tree.newest = 0;
77
78      return tree;
79  }
80
81  function insertTreeVertex(tree,q) {
82
```

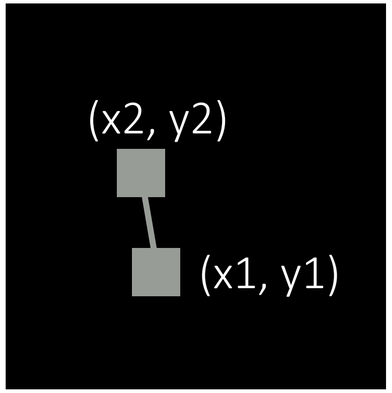
Tree implemented as a JavaScript object with array of vertices

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

Data structure view



Search canvas view

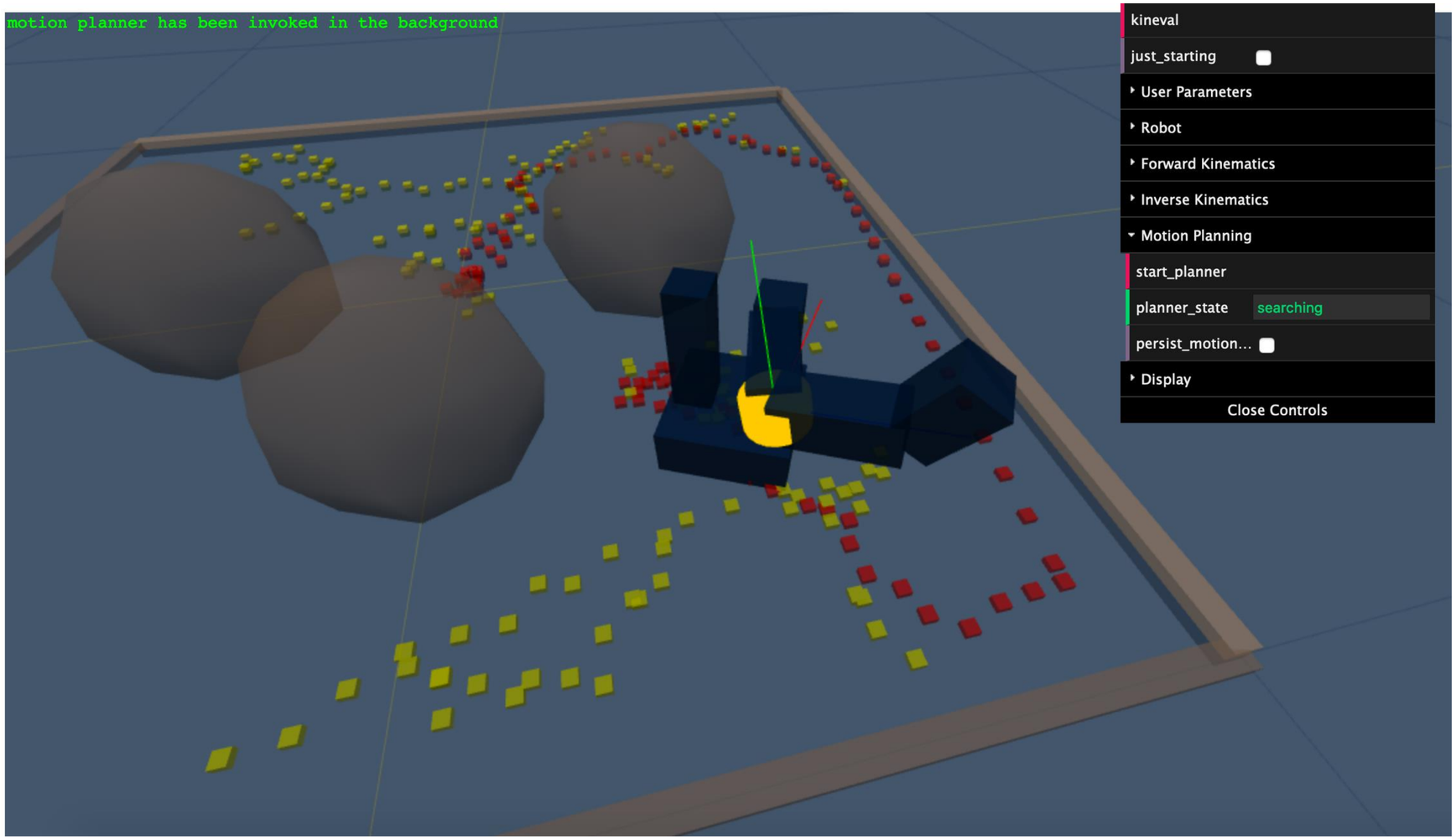


Lab Takeaways

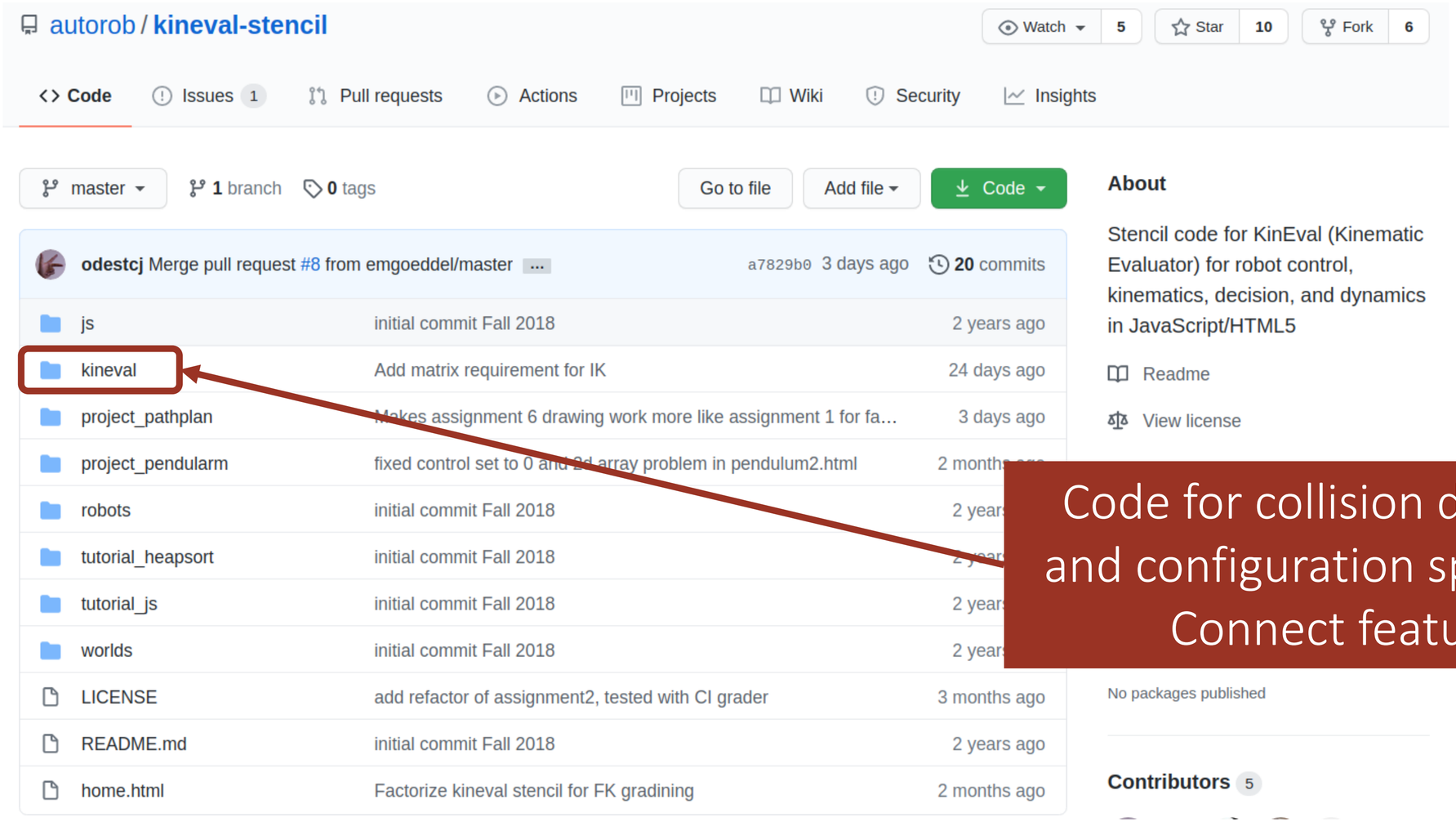
1. RRT connect review
2. 2D RRT connect stencil
3. C-space RRT connect review
4. AABB collision detection

→ How to implement all assignment 6 features

Configuration Space RRT

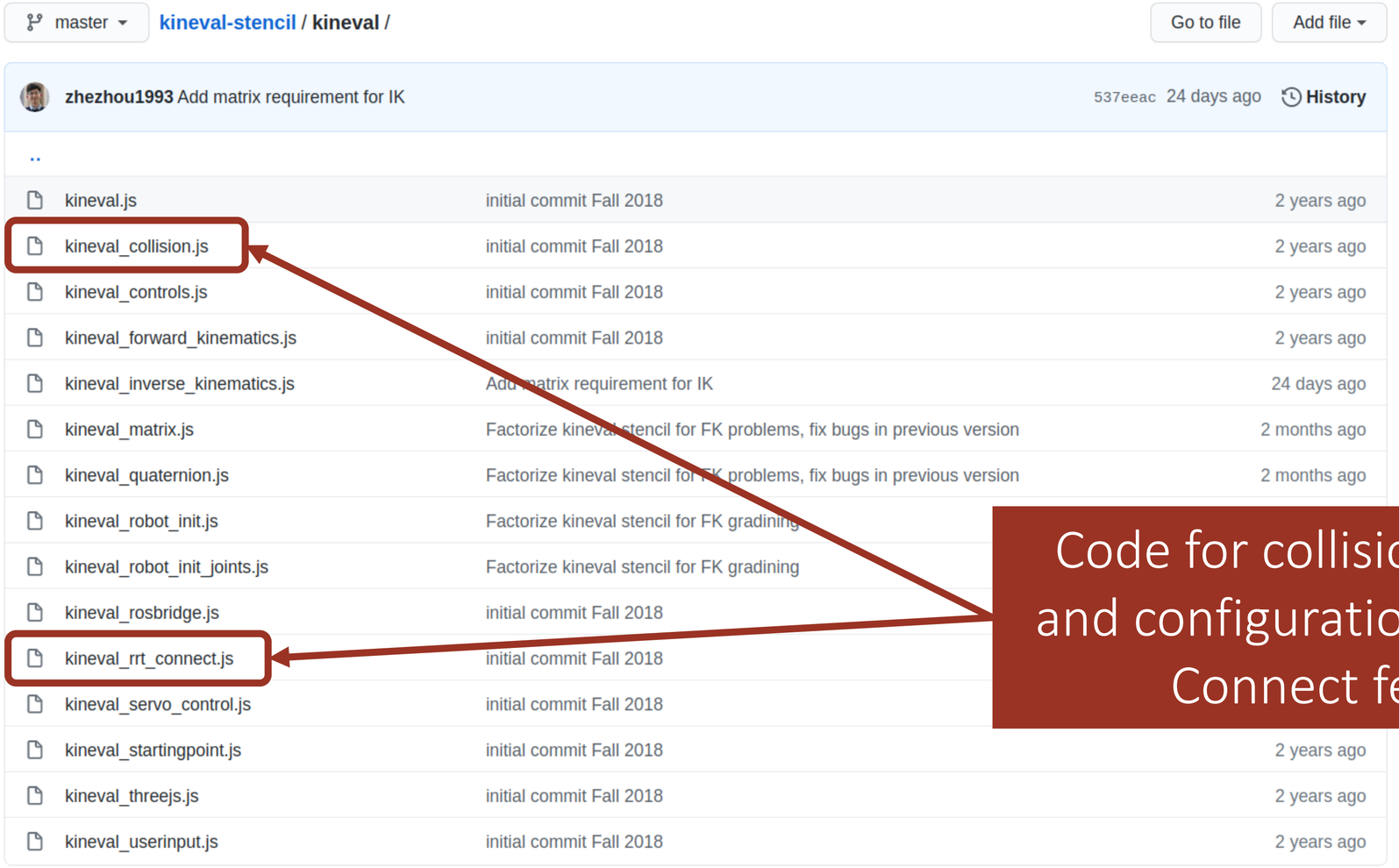


KinEval Overview



Code for collision detection and configuration space RRT-Connect features

KinEval Overview



Code for collision detection and configuration space RRT-Connect features

kineval_rrt.js

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

kineval_rrt.js

```
132  function robot_rrt_planner_iterate() {
133
134      var i;
135      rrt_alg = 1;  // 0: basic rrt (OPTIONAL), 1: rrt_connect (REQUIRED)
136
137      if (rrt_iterate && (Date.now()-cur_time > 10)) {
138          cur_time = Date.now();
139
140      // STENCIL: implement single rrt iteration here. an asynch timing mechanism
141      //   is used instead of a for loop to avoid blocking and non-responsiveness
142      //   in the browser.
143      //
144      //   once plan is found, highlight vertices of found path by:
145      //     tree.vertices[i].vertex[j].geom.material.color = {r:1,g:0,b:0};
146      //
147      //   provided support functions:
148      //
149      //   kineval.poseIsCollision - returns if a configuration is in collision
150      //   tree_init - creates a tree of configurations
151      //   tree_add_vertex - adds and displays new configuration vertex for a tree
152      //   tree_add_edge - adds and displays new tree edge between configurations
153      }
154
155
```

Include any helper functions in this file

kineval_rrt.js

kineval_rrt.js

```
132  function robot_rrt_planner_iterate() {
133
134      var i;
135      rrt_alg = 1;  // 0: basic rrt (OPTIONAL
136
137      if (rrt_iterate && (Date.now()-cur_time > 10)) {
138          cur_time = Date.now();
139
140      // STENCIL: implement single rrt iteration here. an asynch timing mechanism
141      //   is used instead of a for loop to avoid blocking and non-responsiveness
142      //   in the browser.
143      //
144      //   once plan is found, highlight vertices of found path by:
145      //     tree.vertices[i].vertex[j].geom.material.color = {r:1,g:0,b:0};
146      //
147      //   provided support functions:
148      //
149      //   kineval.poseIsCollision - returns if a configuration is in collision
150      //   tree_init - creates a tree of configurations
151      //   tree_add_vertex - adds and displays new configuration vertex for a tree
152      //   tree_add_edge - adds and displays new tree edge between configurations
153      }
154
155
```

RRT_CONNECT_PLANNER(q_{init}, q_{goal})

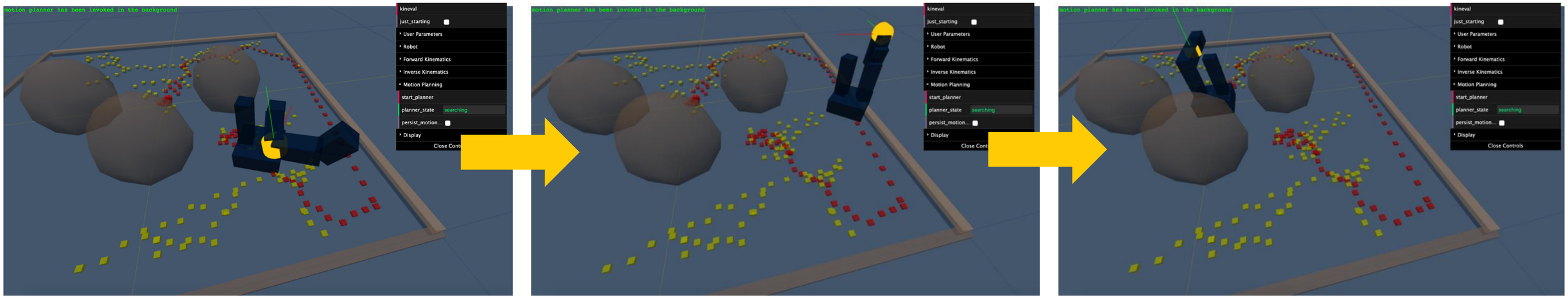
1 $\mathcal{T}_a$.init(q_{init}); $\mathcal{T}_b$.init(q_{goal});
2 **for** $k = 1$ **to** K **do**
3 $q_{rand} \leftarrow$ RANDOM_CONFIG();
4 **if not** (EXTEND($\mathcal{T}_a, q_{rand}$) = *Trapped*) **then**
5 **if** (CONNECT($\mathcal{T}_b, q_{new}$) = *Reached*) **then**
6 Return PATH($\mathcal{T}_a, \mathcal{T}_b$);
7 SWAP($\mathcal{T}_a, \mathcal{T}_b$);
8 Return *Failure*

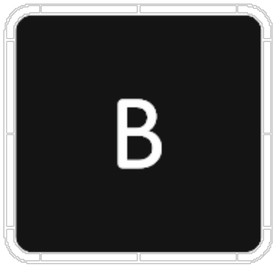
Include any helper functions in this file

Desired Result

Arbitrary initial configuration

Collision free path to home





Backward step along motion plan

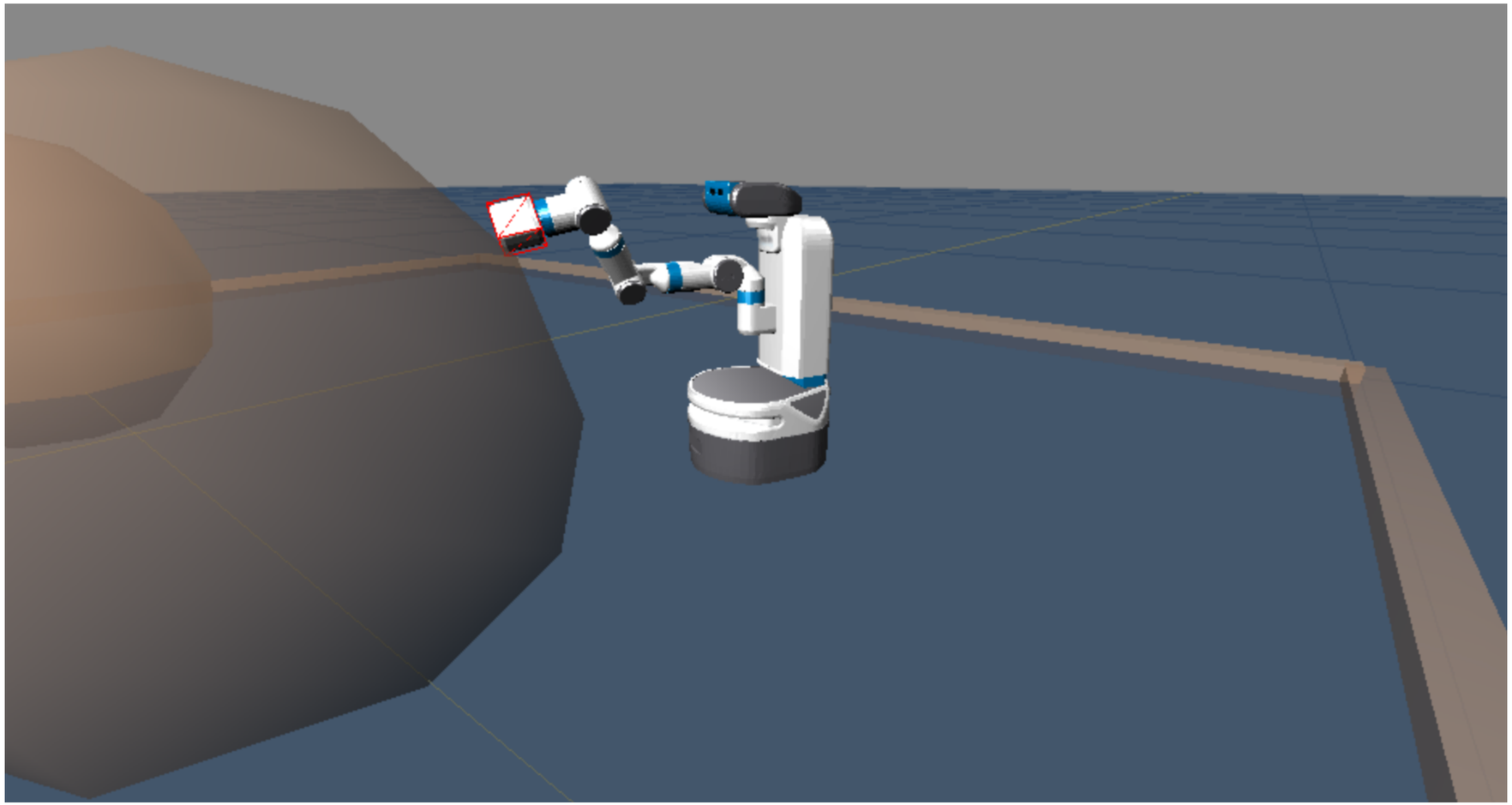
Forward step along motion plan

Lab Takeaways

1. RRT connect review
2. 2D RRT connect stencil
3. C-space RRT connect review
4. AABB collision detection

→ How to implement all assignment 6 features

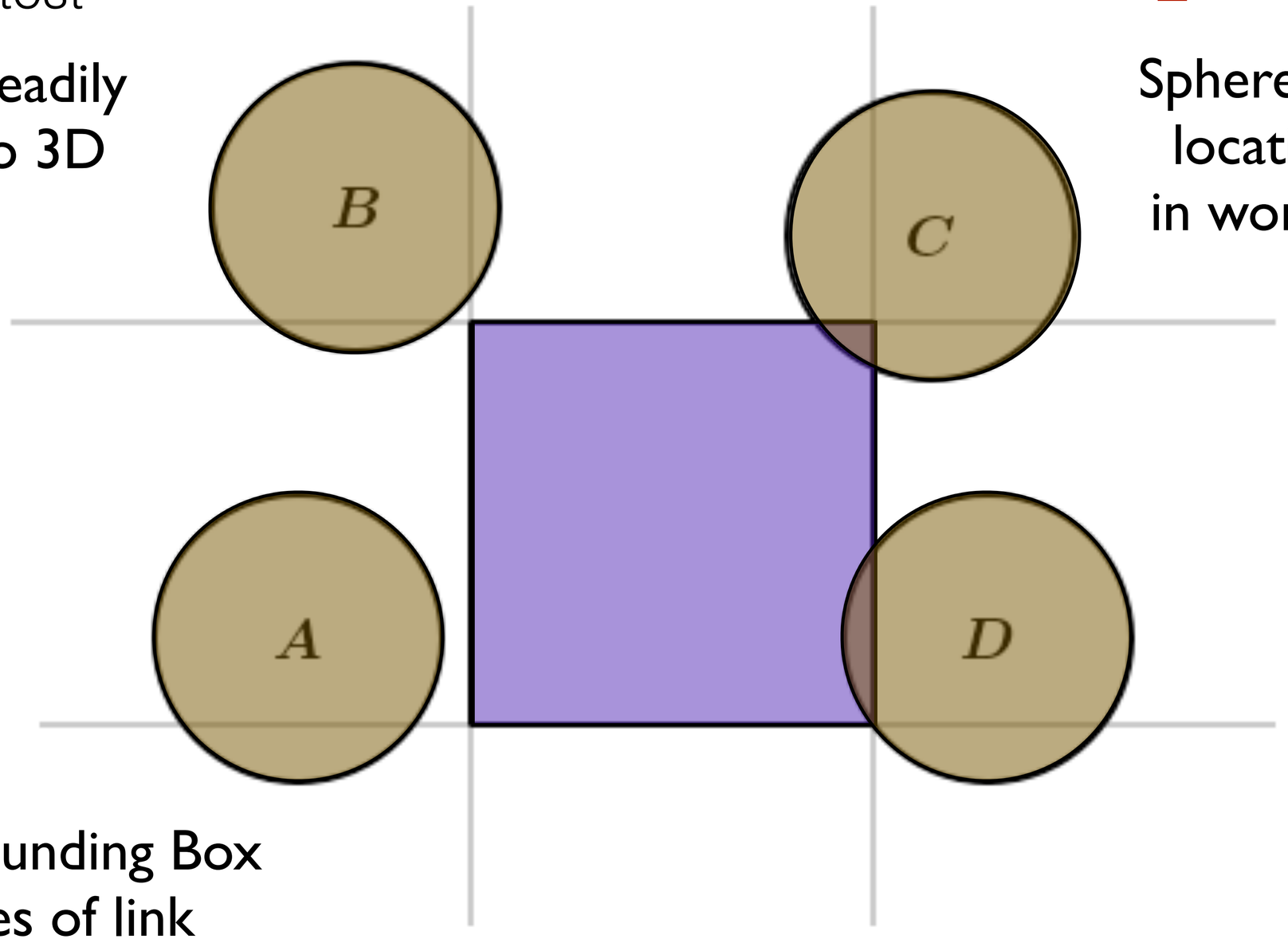
Consider AABB link tested against spherical obstacles in link frame



Sphere-bbox test

`robot_obstacles[i]`

2D example readily generalizes to 3D



Sphere obstacles with location and radius in world coordinates

Axis Aligned Bounding Box in coordinates of link

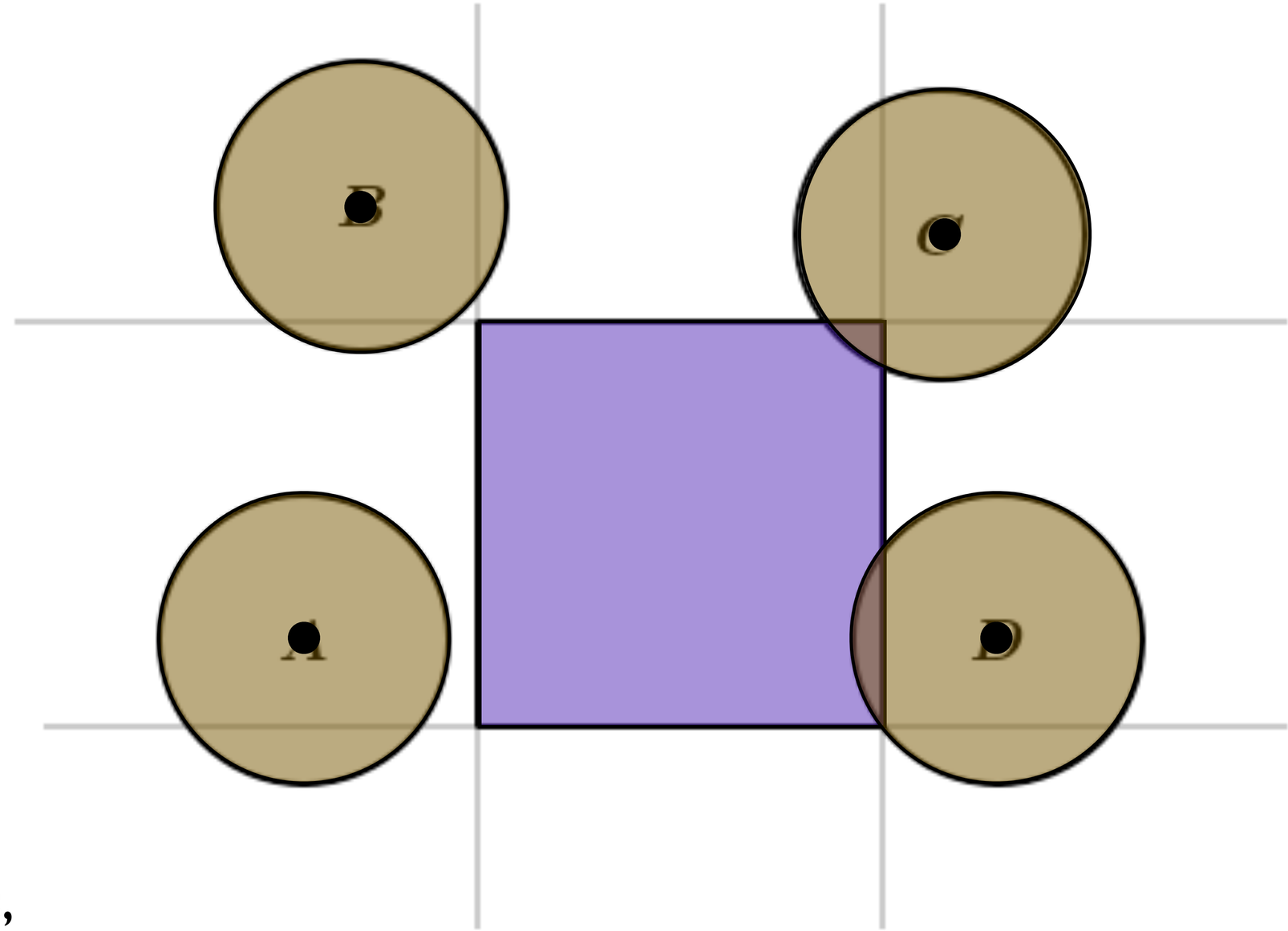
```
robot.links[x].bbox = [[x_min,y_min,z_min], [x_max,y_max,z_max]]
```

Sphere-bbox test

If sphere separable from AABB in any dimension, return no collision

`loc_y-radius>y_max?`

`loc_y+radius<y_min?`

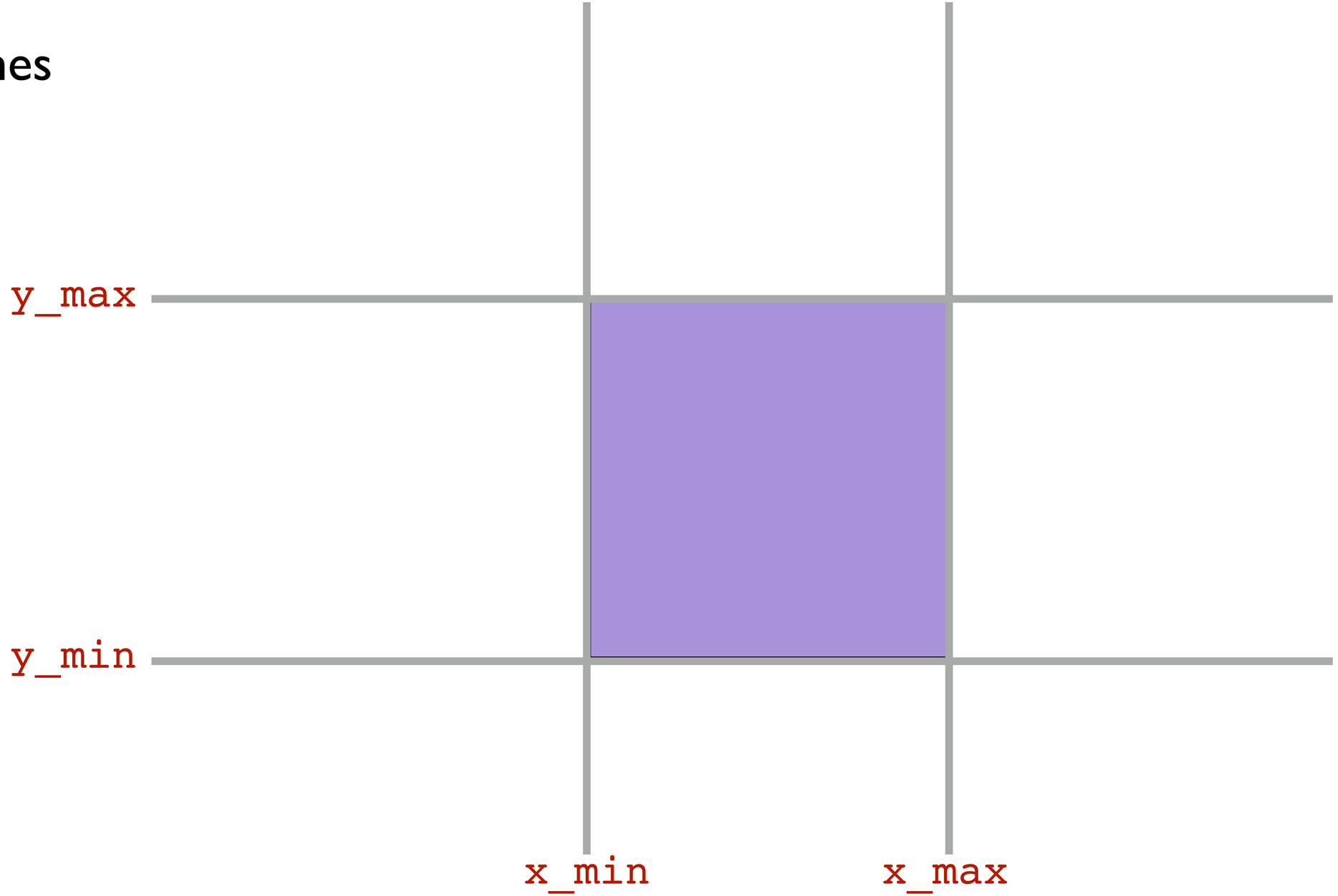


`loc_x+radius<x_min?`

`loc_x-radius>x_max?`

If sphere collides on all tests, return collision

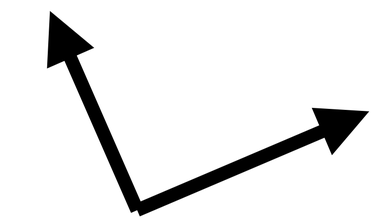
Separating planes



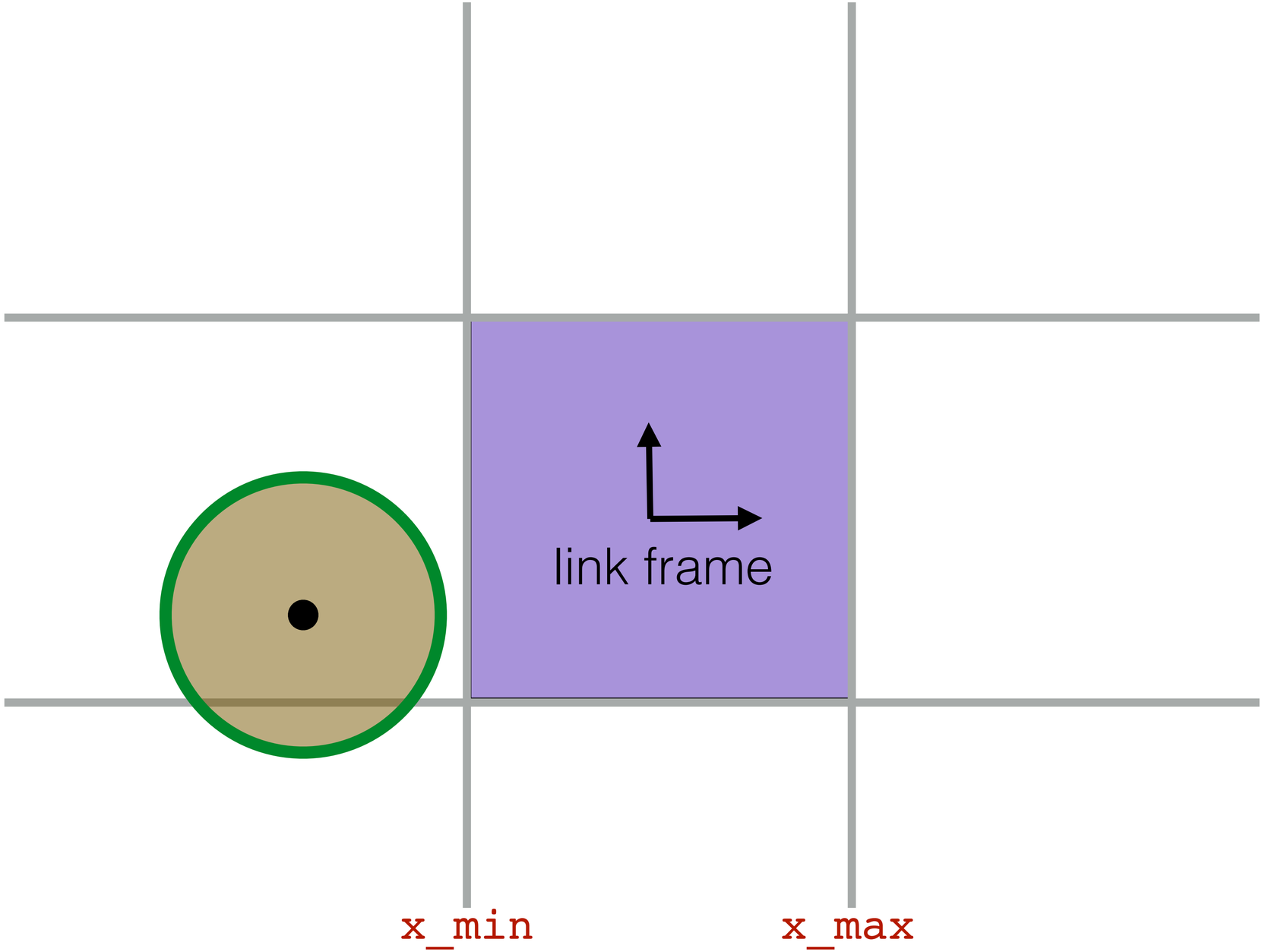
Transform centers of sphere obstacles into link coordinates

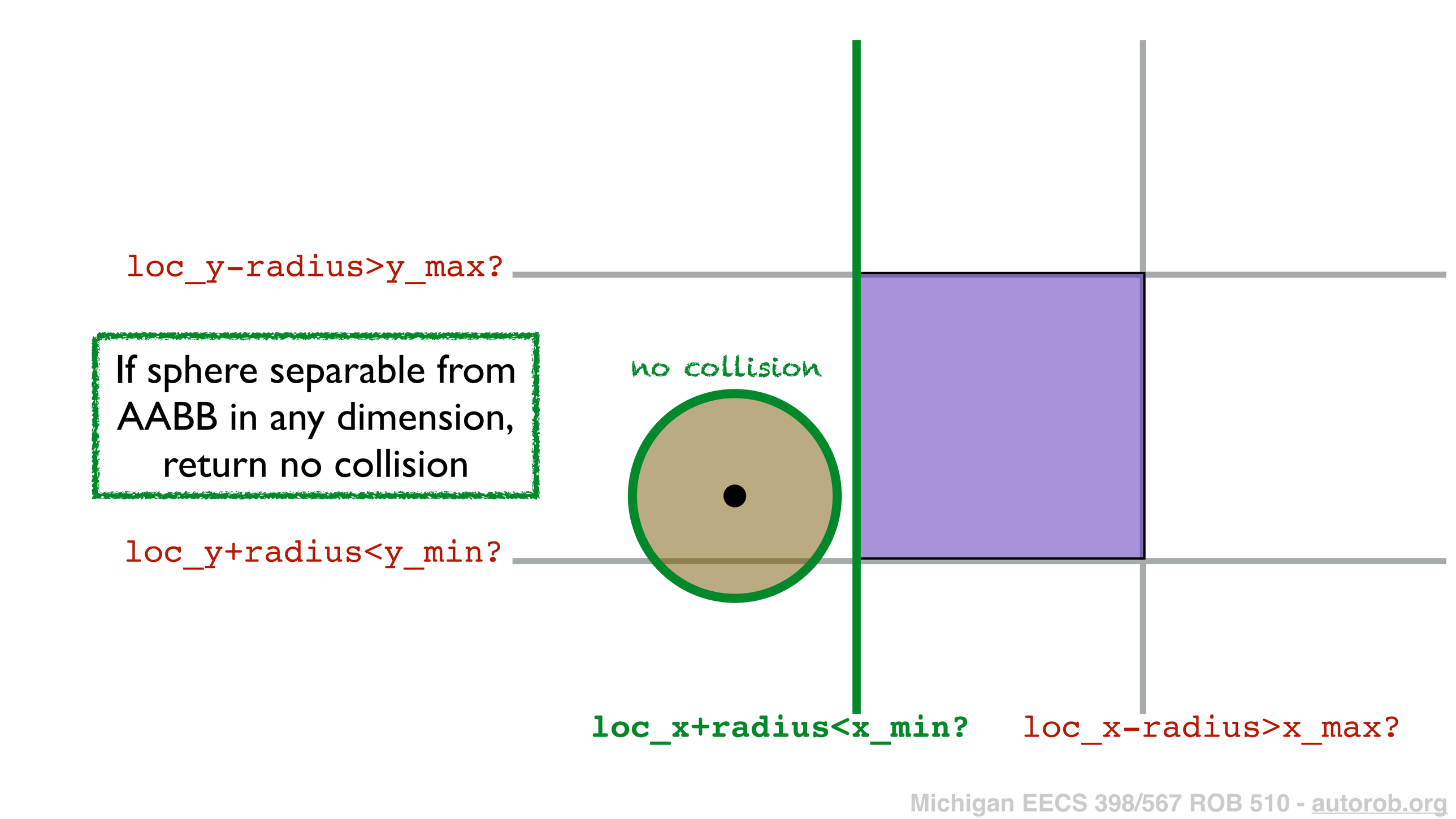
(Remember inverse of homogeneous transform?)

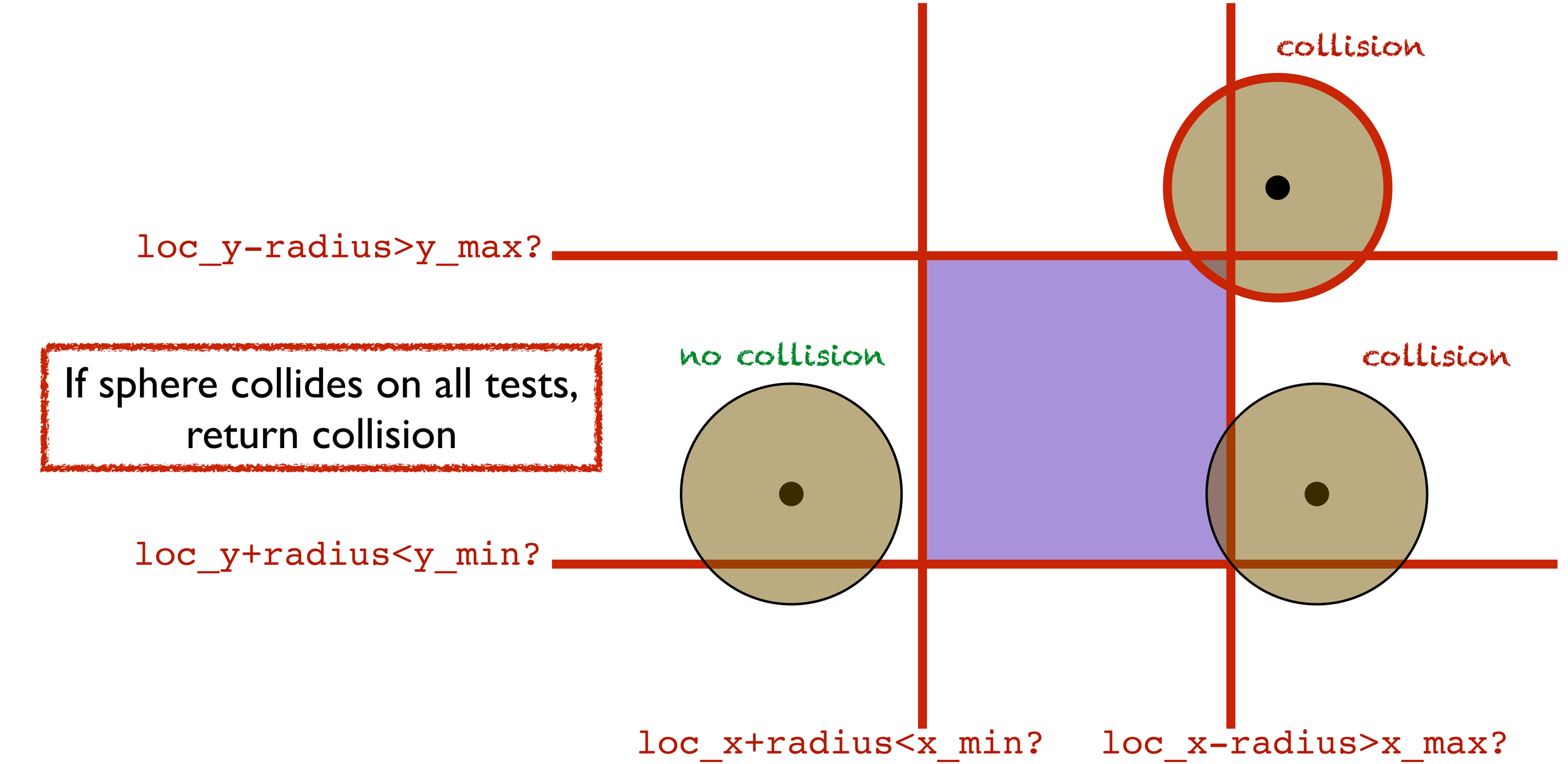
$$p^{link} = (T_{link}^{world})^{-1} p^{world}$$



world frame

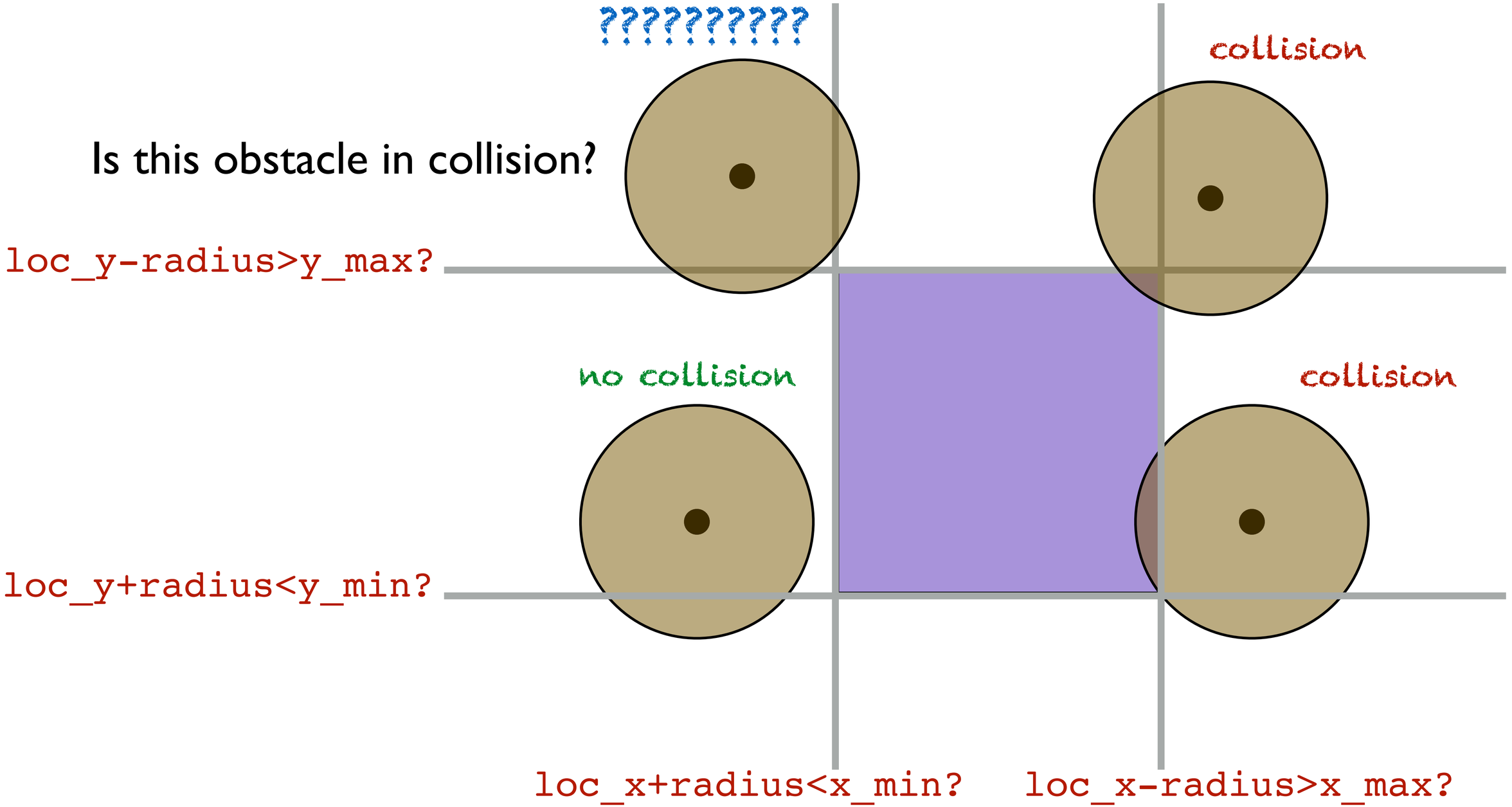


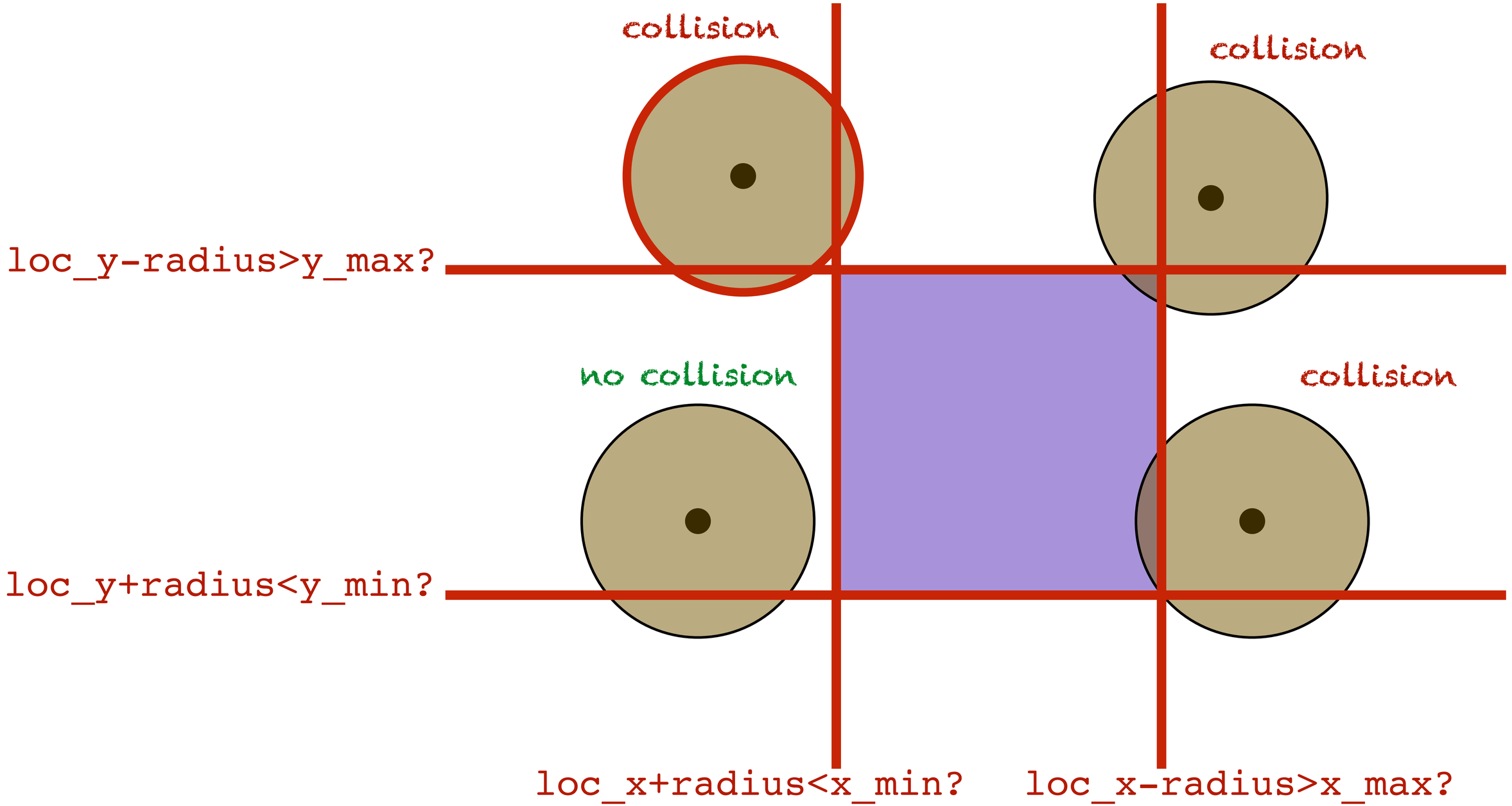

loc_y-radius>y_max?
If sphere separable from AABB in any dimension, return no collision
no collision
loc_y+radius<y_min?
loc_x+radius<x_min?
loc_x-radius>x_max?



If sphere collides on all tests, return collision

Is this obstacle in collision?




collision
collision
loc_y-radius>y_max?
no collision
collision
loc_y+radius<y_min?
loc_x+radius<x_min?
loc_x-radius>x_max?

kineval_collision.js

kineval_collision.js

```
22  kineval.robotIsCollision = function robot_iscollision() {
23      // test whether geometry of current configuration of robot is in collision with planning world
24
25      // form configuration from base location and joint angles
26      var q_robot_config = [
27          robot.origin.xyz[0],
28          robot.origin.xyz[1],
29          robot.origin.xyz[2],
30          robot.origin.rpy[0],
31          robot.origin.rpy[1],
32          robot.origin.rpy[2]
33      ];
34
35      q_names = {};  // store mapping between joint names and q DOFs
36
37      for (x in robot.joints) {
38          q_names[x] = q_robot_config.length;
39          q_robot_config = q_robot_config.concat(robot.joints[x].angle);
40      }
41
42      // test for collision and change base color based on the result
43      collision_result = kineval.poseIsCollision(q_robot_config);
44
45      robot.collision = collision_result;
46  }
```

`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

```
49  kineval.poseIsCollision = function robot_collision_test(q) {
50      // perform collision test of robot geometry against planning world
51
52      // test base origin (not extents) against world boundary extents
53      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]))
54          return robot.base;
55
56      // traverse robot kinematics to test each body for collision
57      // STENCIL: implement forward kinematics for collision detection
58      //return robot_collision_forward_kinematics(q);
59
60  }
```

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

kineval_collision.js

kineval_collision.js

```
64  function traverse_collision_forward_kinematics_link(link,mstack,q) {
65
66      /* test collision FK
67      console.log(link);
68      */
69      if (typeof link.visual !== 'undefined') {
70          var local_link_xform = matrix_multiply(mstack,generate_translation_matrix(link.visual.origin.xyz[0],link
71      }
72      else {
73          var local_link_xform = matrix_multiply(mstack,generate_identity());
74      }
75
76      // test collision by transforming obstacles in world to link space
77  /*
78      mstack_inv = matrix_invert_affine(mstack);
79  */
80      mstack_inv = numeric.inv(mstack);
81
82      var i;
83      var j;
84
85      // test each obstacle against link bbox geometry by transforming obstacle into link frame and testing against
86      //for (j=0;j<robot_obstacles.length;j++) {
87      for (j in robot_obstacles) {
88
89          var obstacle_local = matrix_multiply(mstack_inv,robot_obstacles[j].location);
90
91          // assume link is in collision as default
92          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

Lab Takeaways

1. RRT connect review
2. 2D RRT connect stencil
3. C-space RRT connect review
4. AABB collision detection

→ How to implement all assignment 6 features

Best use of robotics?

- What are some compelling use cases for robotics?
- Discuss with neighbors
  - Then share possible ideas