Disclaimer

These slides are intended as presentation aids for the lecture. They contain information that would otherwise be to difficult or time-consuming to reproduce on the board. But they are incomplete, not self-explanatory, and are not always used in the order they appear in this presentation. As a result, these slides should not be used as a script for this course. I recommend you take notes during class, maybe on the slides themselves. It has been shown that taking notes improves learning success.

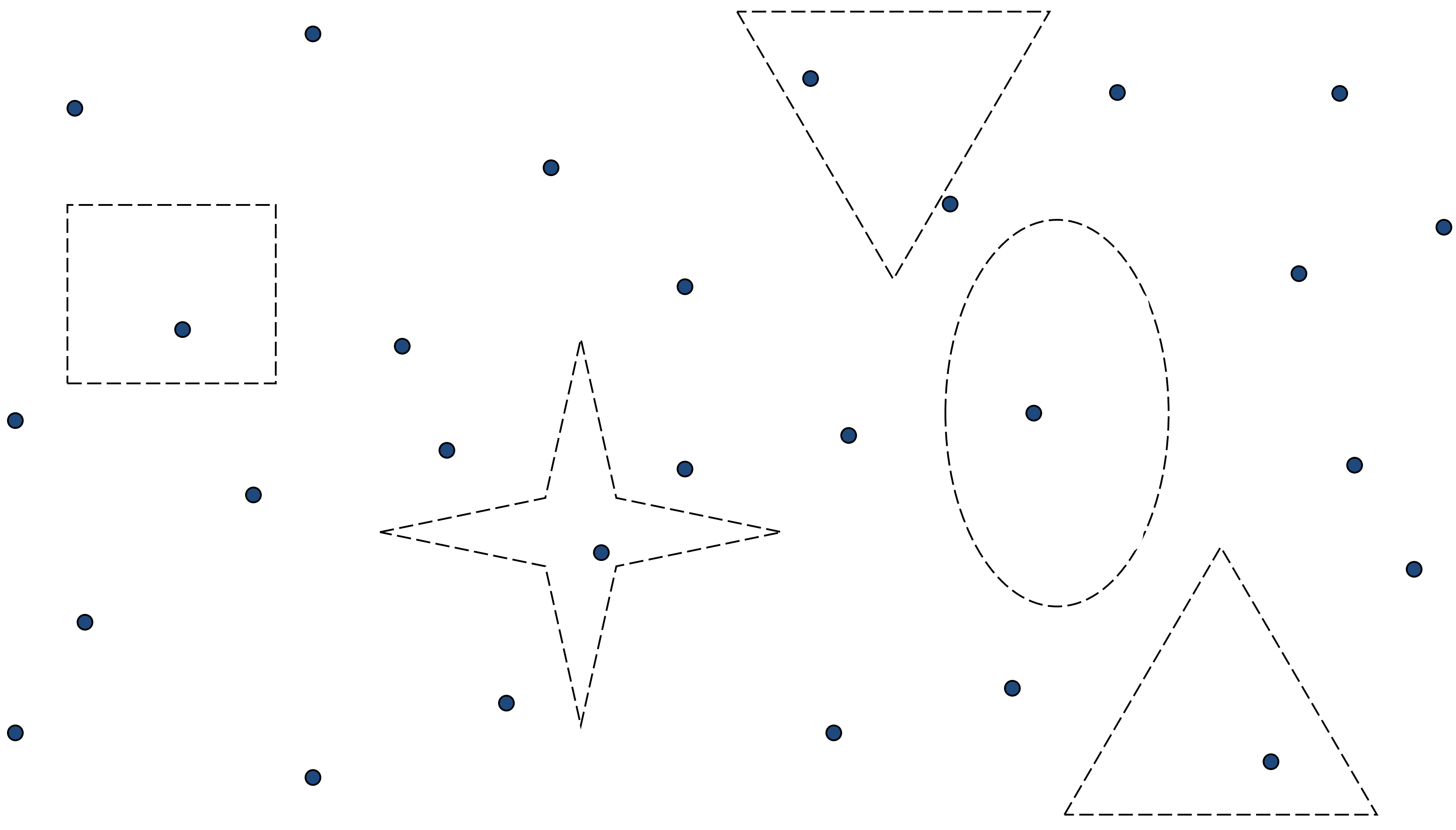
Robotics

Sampling-based Multi Query Motion Planning

TU Berlin

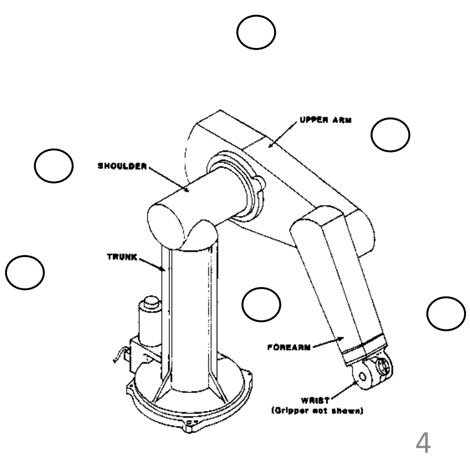
Oliver Brock

Sampling Configuration Space



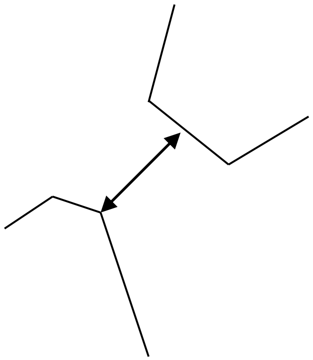
Basic Primitive: Collision Detection

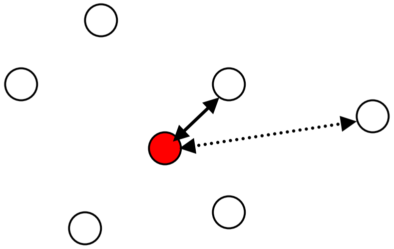
- Computational complexity collision detection
  - n objects have O(n^2) interactions
  - Robot with l links and n obstacles has O($l * n$)
  - Each object has many features – 1000s!!!
  - In practice a **costly** operation



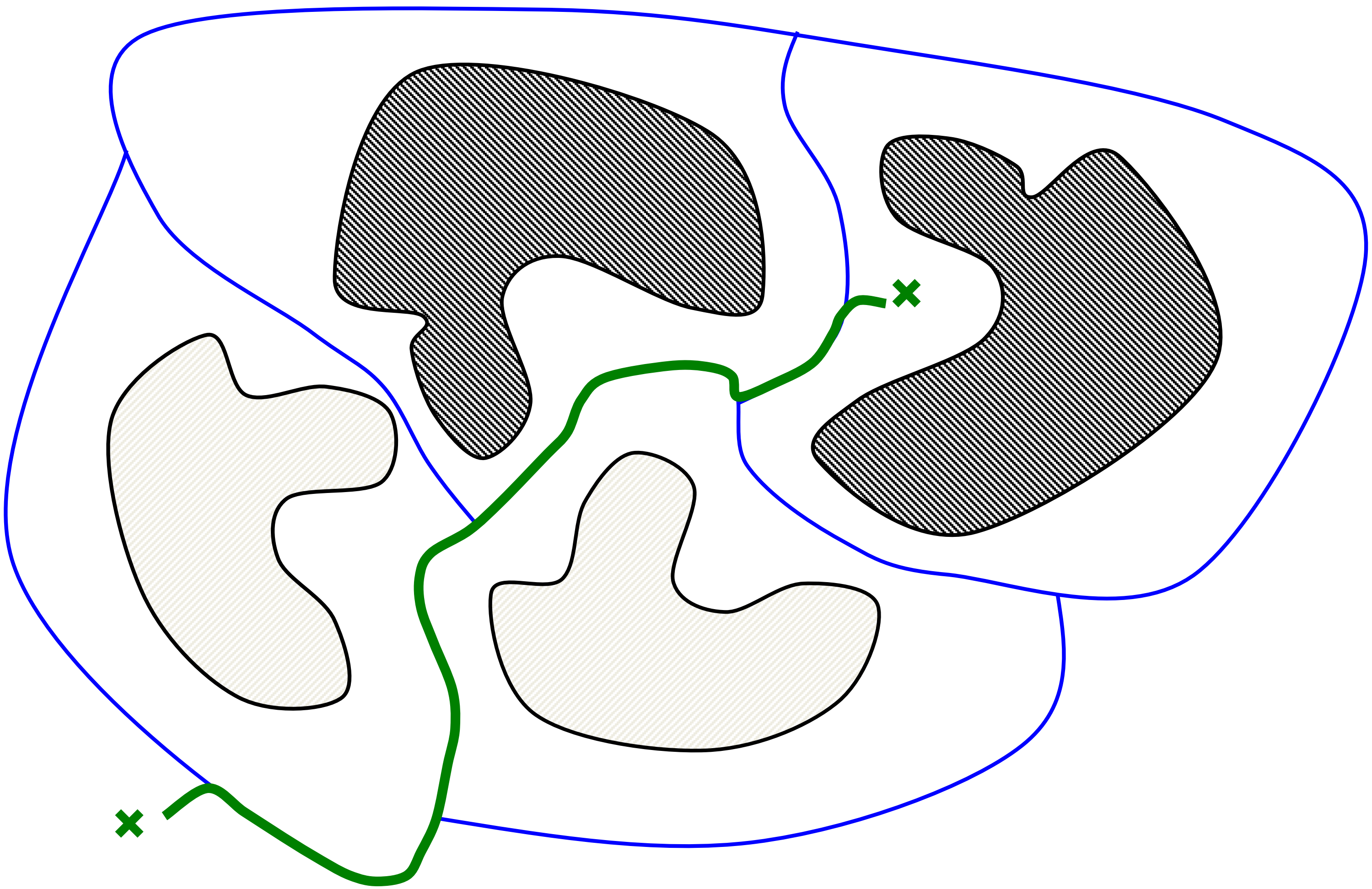
Tricks for Distance Computation

- Exploit spatial coherency
  - Group primitives hierarchically
  - Exploit adjacency (on single object)
- Exploit temporal coherency
  - Exploit former relation between multiple objects
- Heuristics are good
- Problem remains computationally complex





What is the perfect roadmap?

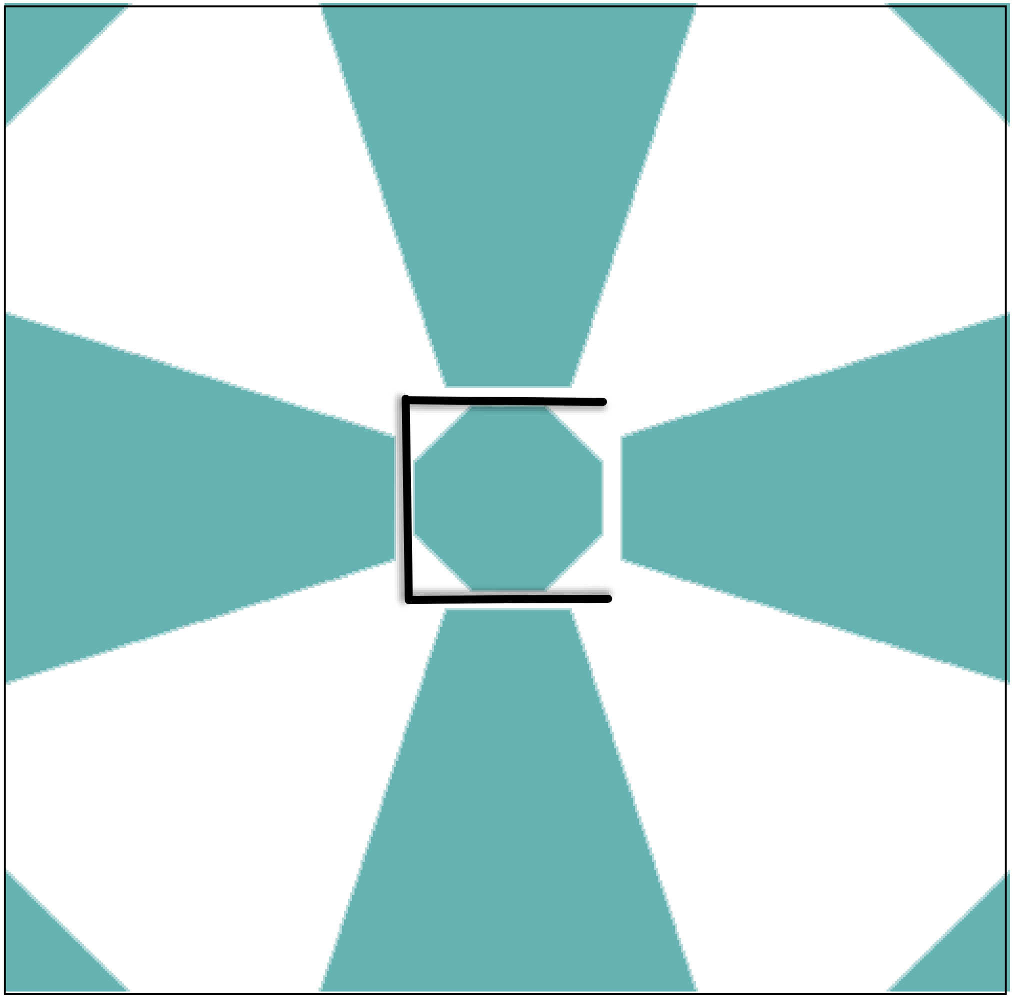


An Ideal Roadmap

- Any point in C-space should be connectable to the roadmap
- If there is a path between two points in C-space the roadmap should contain a path between them after they were connected to the roadmap

- How can such a roadmap be obtained through sampling?

Perfect Roadmap



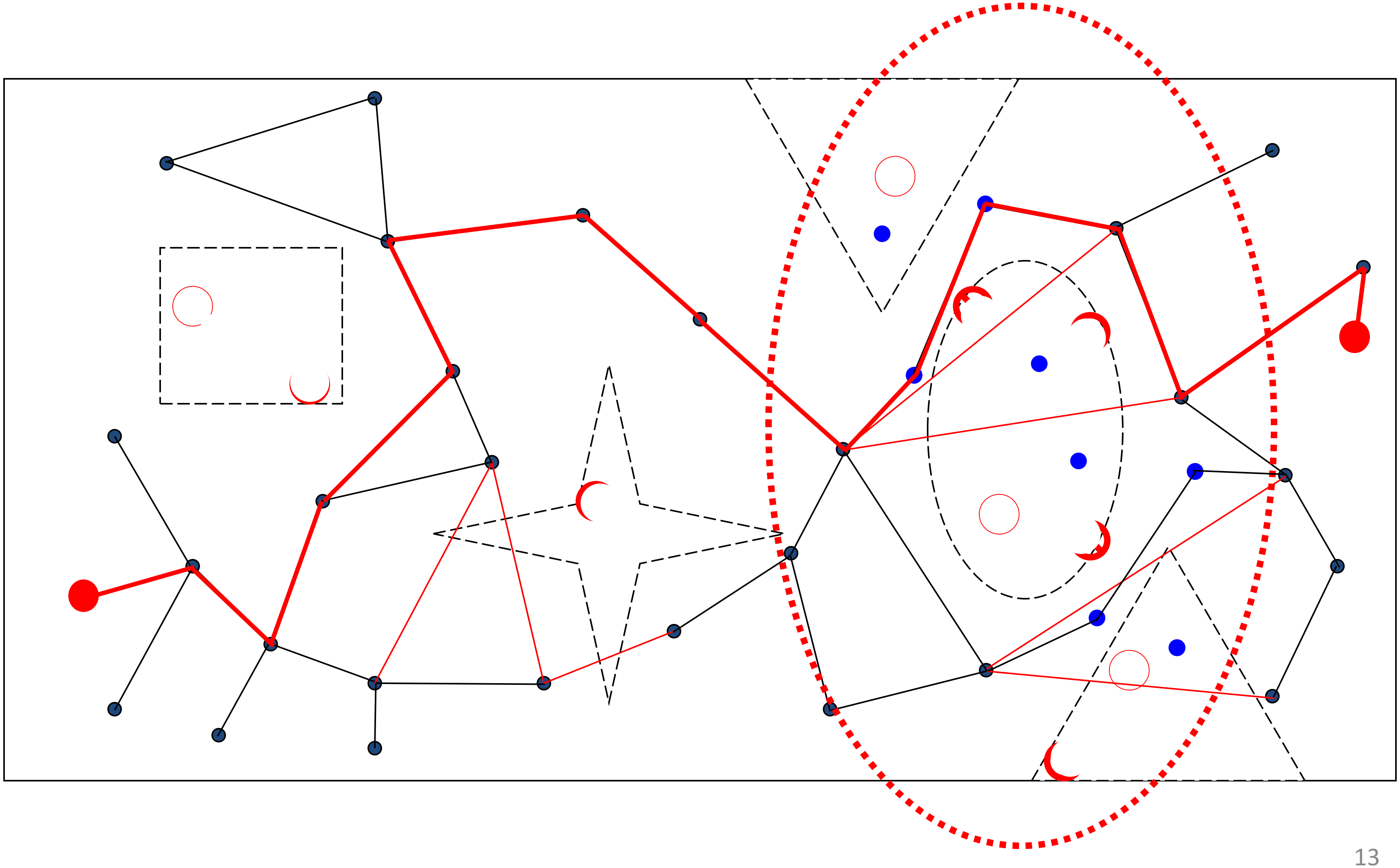
Exploration versus Exploitation

Exploration seeks **understanding of the state space**, irrespective of a particular task.
In motion planning, the process **exploration** seeks to understand the connectivity of the configuration space, irrespective of solving a particular motion planning problem.

Guided exploration seeks **efficient** **understanding of the state space**, irrespective a particular task, by **leveraging available information**.

Exploitation strives to **accomplish a particular task** as **efficiently as possible** by **leveraging available information**.
In motion planning**, exploitation** seeks a valid path for a **particular task**, based on available information.

Probabilistic Roadmap (PRM) Planner



Probabilistic Roadmap Planner

- Construction
  - Generate random configurations
  - Eliminate if they are in collision
  - Use local planner to connect configurations
- Expansion
  - Identify connected components
  - Resample gaps
  - Try to connect components
- Query
  - Connect initial and final configuration to roadmap
  - Perform graph search

Learning

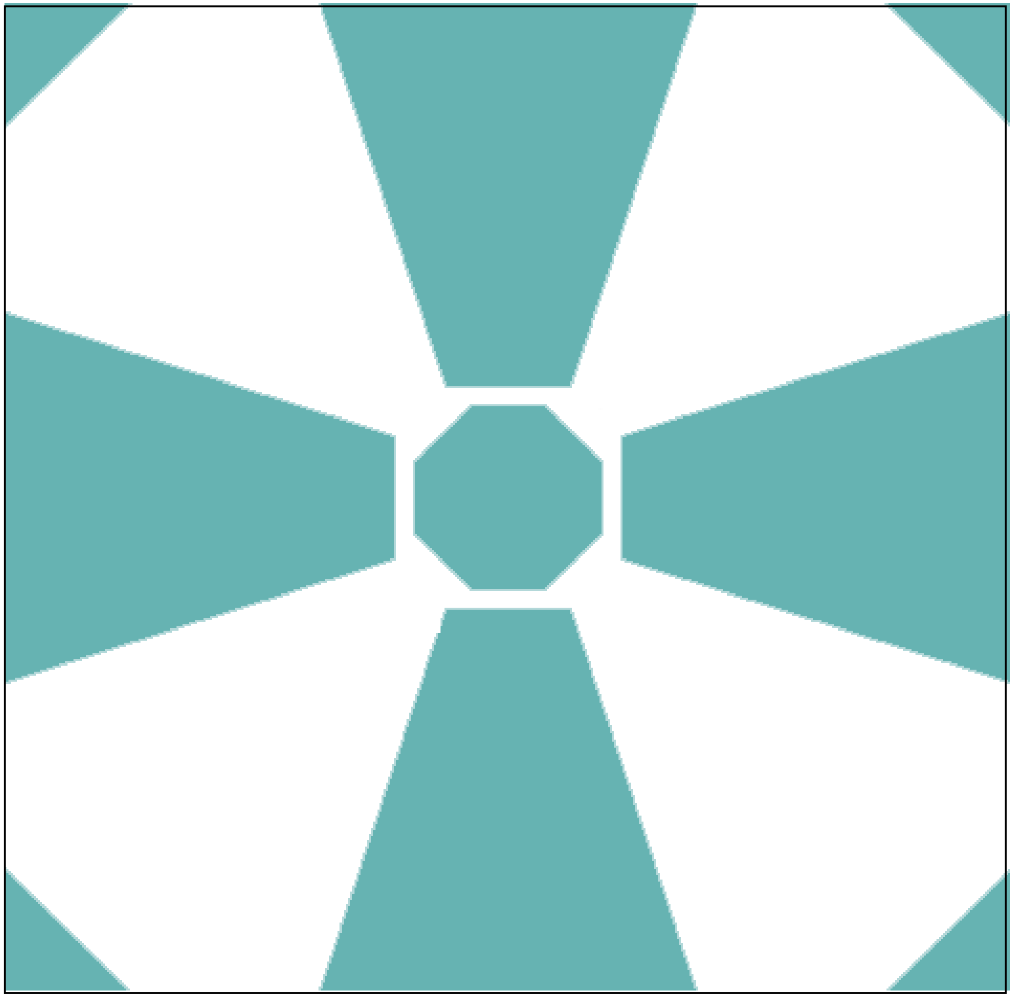
Learning Phase

- Construction
  - $R = (V,E)$
  - repeat n times:
    - generate random configuration
    - add to V if collision free
    - attempt to connect to *neighbors* using *local planner*, unless in same connected component of R
- Expansion
  - repeat k times:
    - select *difficult* node
    - attempt to connect to neighbors using *another local planner*

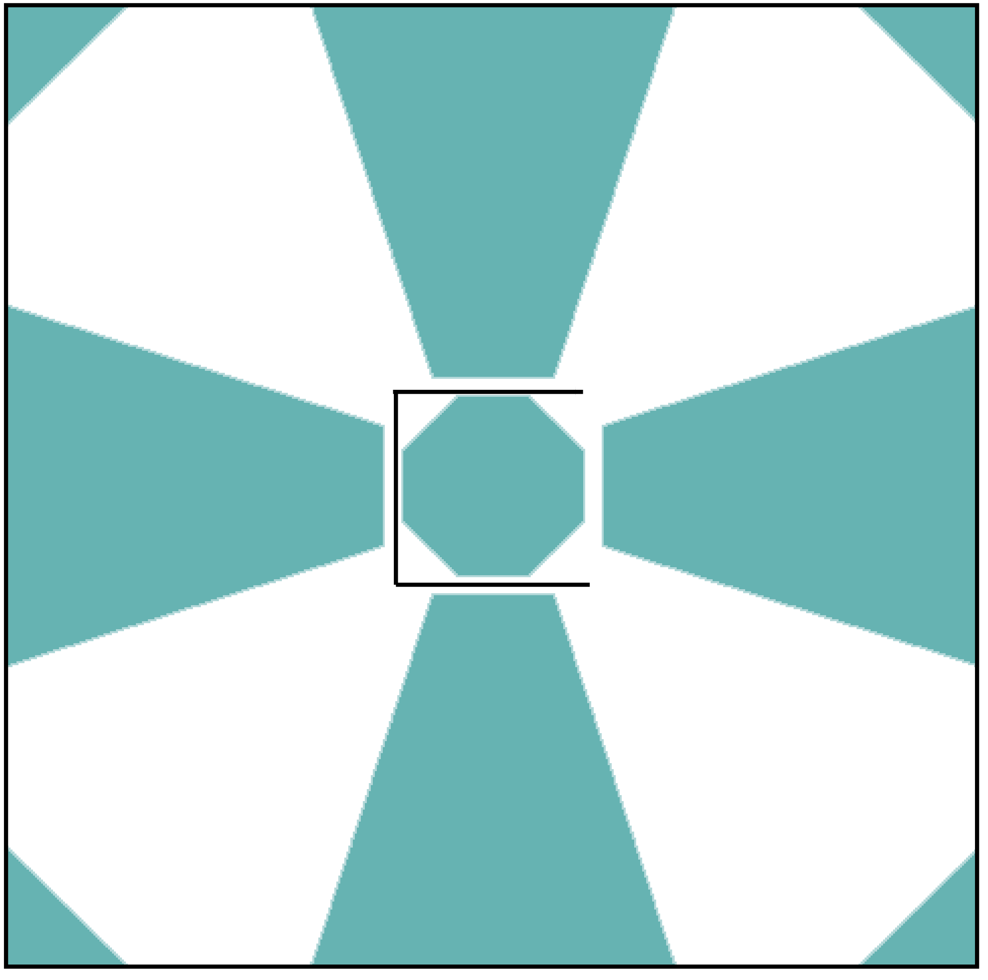
Query

- Connect start and goal configuration to roadmap using local planner
- Perform graph search on roadmap
- Computational cost of querying neclegible compared to construction of roadmap

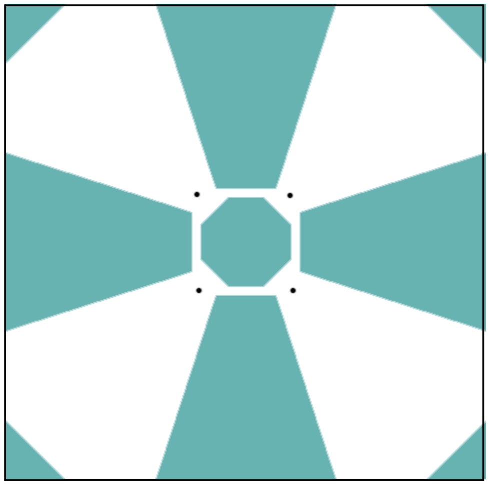
Perfect Sampling



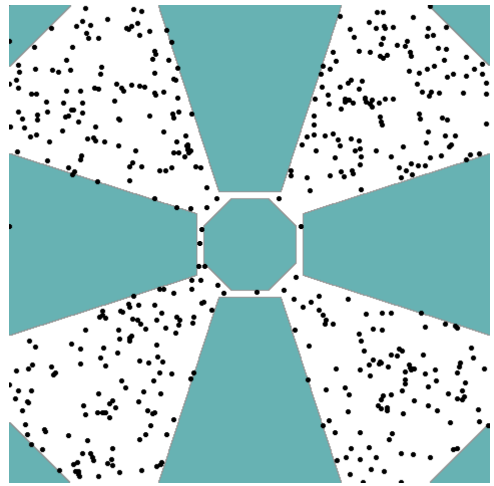
Perfect Roadmap



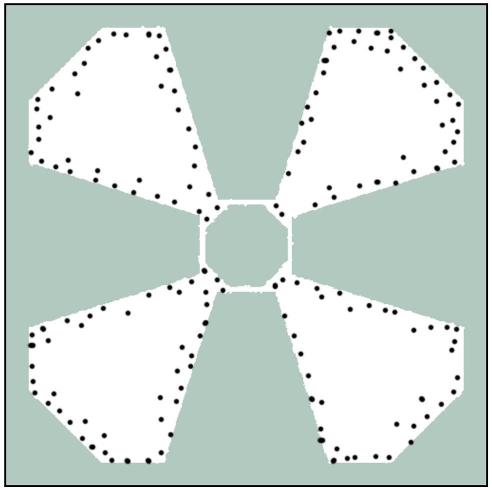
Different Sampling Strategies



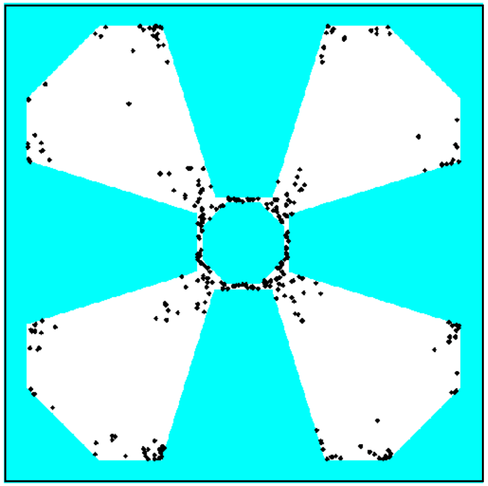
ideal



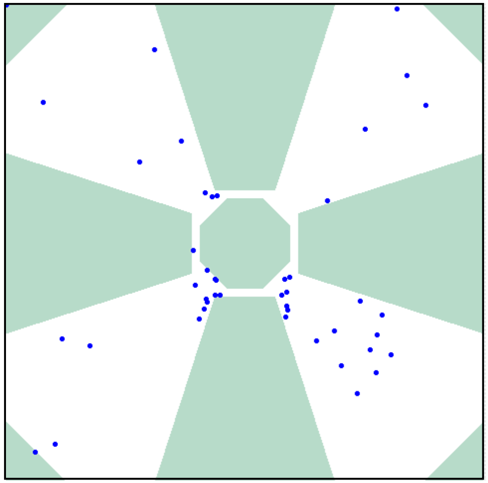
uniform



Gaussian

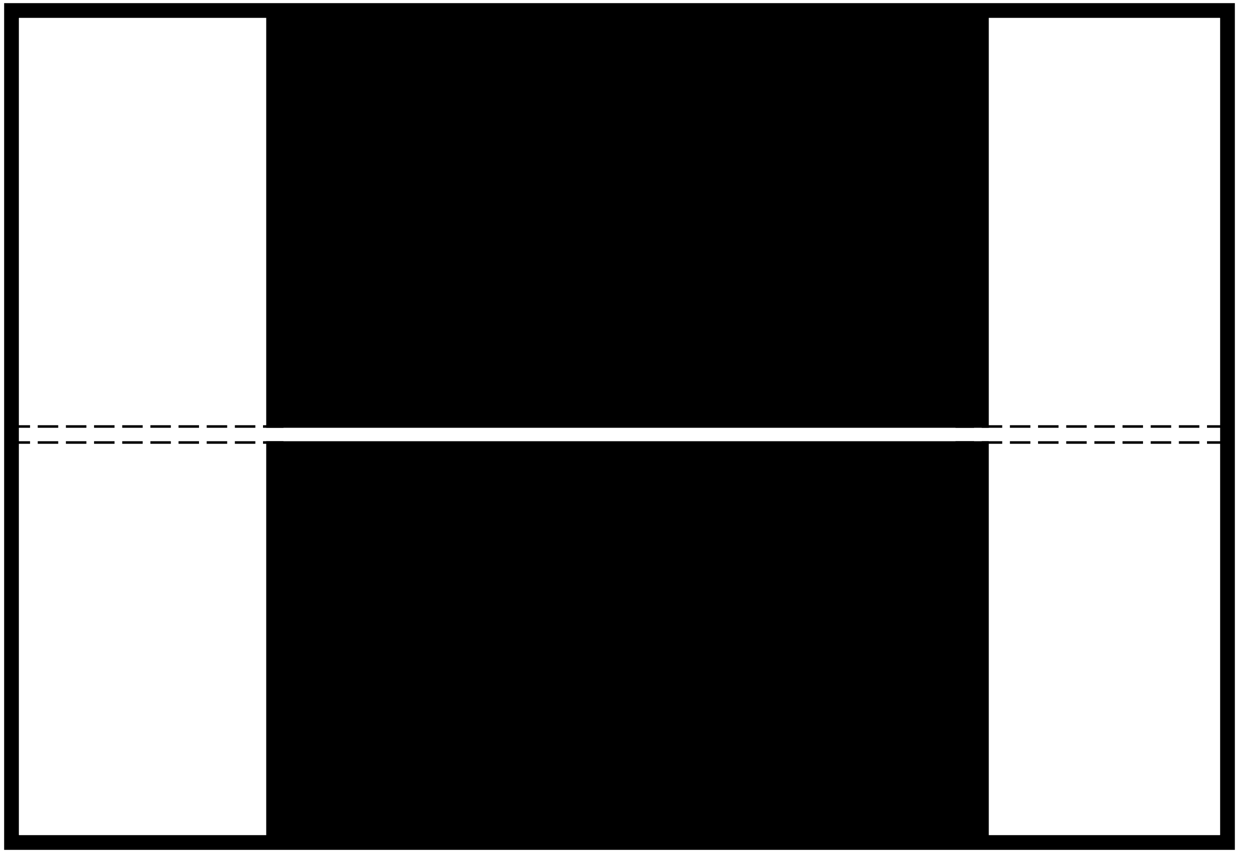


Bridge



Utility

Narrow Passage Problem

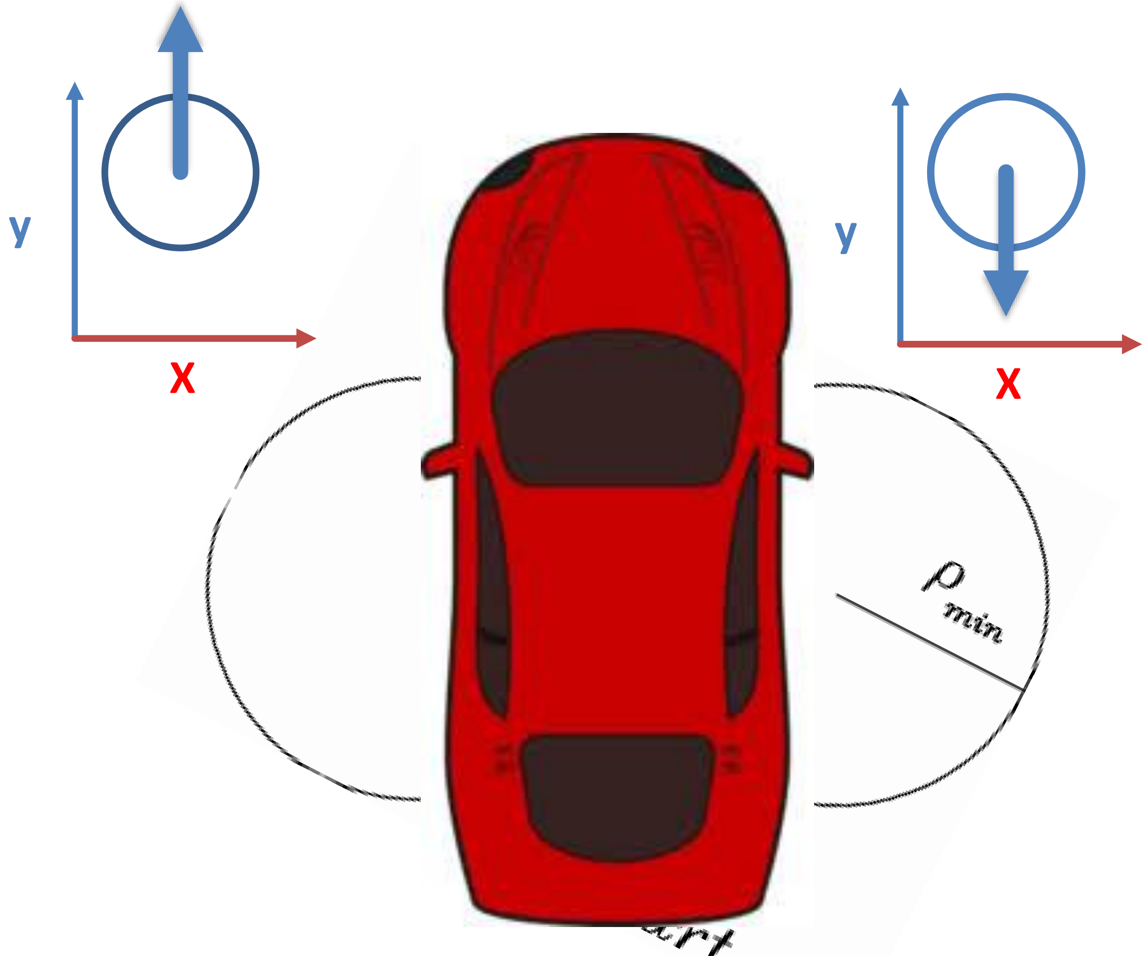


Key to Good Sampling: Exploiting Structure

- *Identify* underlying structure
- *Represent* information about structure
- *Exploit* information
- *Structure* can come from
  - sampling
  - problem description

Neighbors

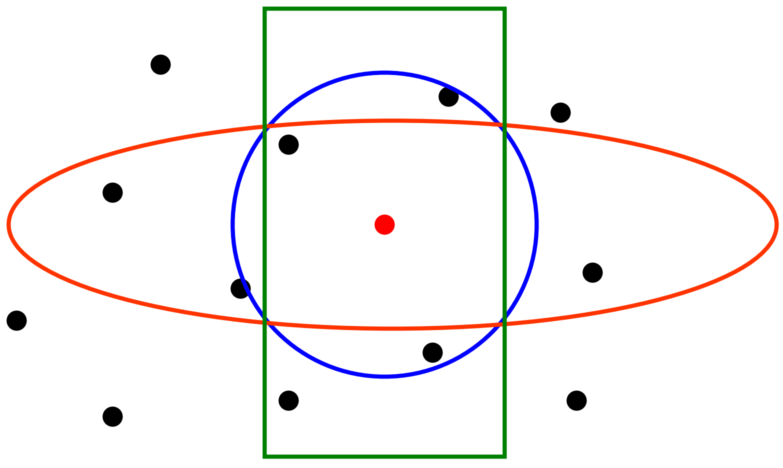
- Use distance metric to determine neighbor



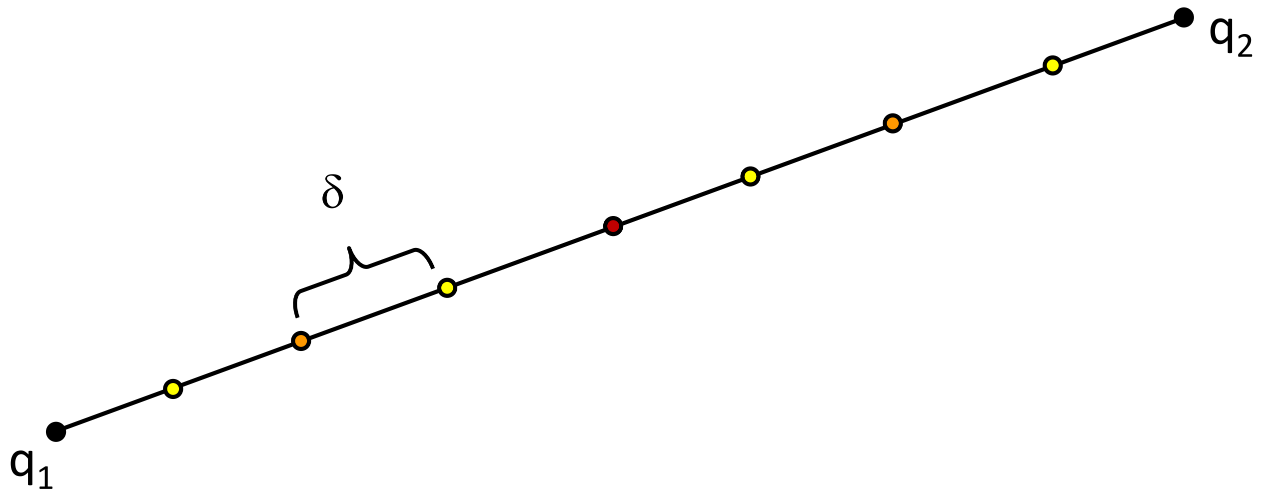
Bicchi et.al. *Planning shortest bounded-curvature paths for a class ofnonholonomic vehicles among obstacles*

Neighbors

- Use distance metric to determine neighbor
- Euclidian distance oftentimes used
- Others possible:
  - maximum Euclidian distance
  - maximum joint difference

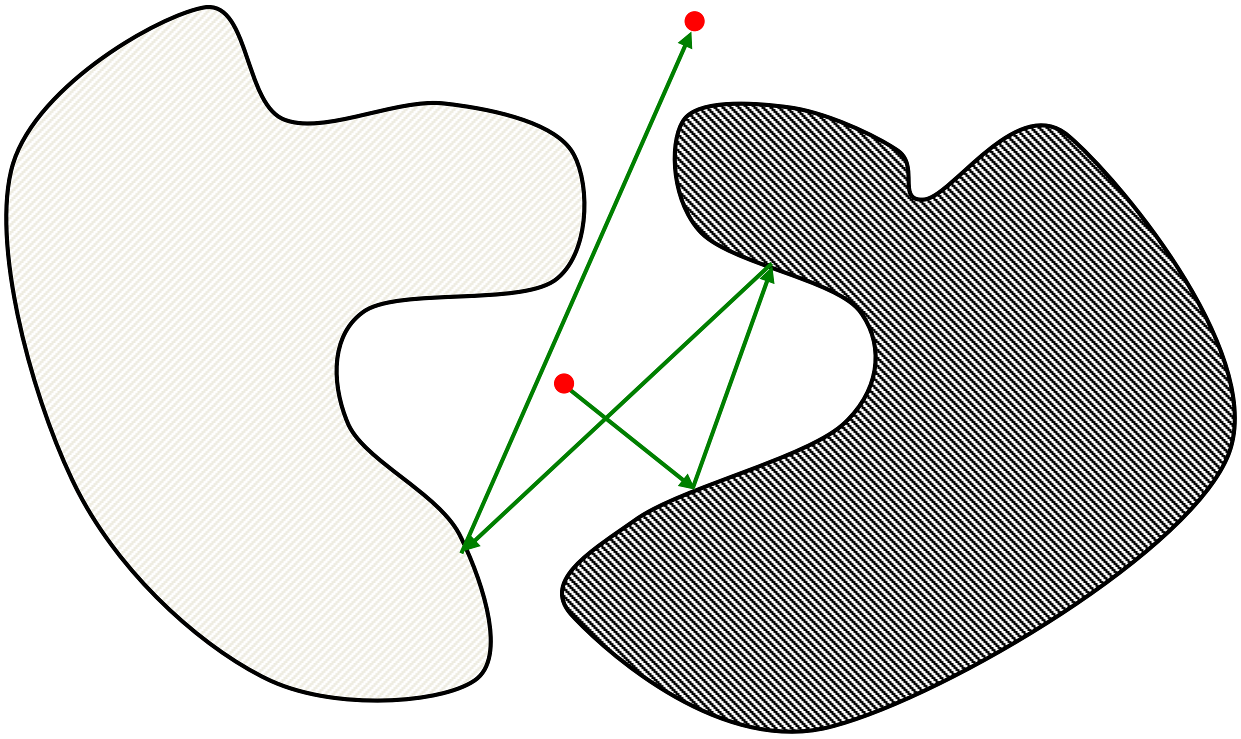


Local Planner



tests up to a specified resolution δ!

Another Local Planner



perform random walk of predetermined length;
choose new direction randomly after hitting obstacle;
attempt to connect to roadmap after random walk

PRM Limits Local Planners

- Consider car-like robot
- Connecting configurations might be difficult
- Goal: provide probabilistic method for kinematic and dynamic constraints
  - Car-like
  - Satellite
  - Plane
- Idea: Let local planner choose configurations

Summary: PRM

- Algorithmically very simple
- Surprisingly efficient even in high-dimensional C-spaces
- Capable of addressing a wide variety of motion planning problems
- One of the hottest areas of research
- Allows probabilistic performance guarantees
- **BUT: narrow passage problem!**

Exploration versus Exploitation

Exploration seeks **understanding of the state space**, irrespective of a particular task. In motion planning, the process **exploration** seeks to understand the connectivity of the configuration space, irrespective of solving a particular motion planning problem.

Guided exploration seeks **efficient** **understanding of the state space**, irrespective a particular task, by **leveraging available information**.

Exploitation strives to **accomplish a particular task** as **efficiently as possible** by **leveraging available information**. In motion planning, **exploitation** seeks a valid path for a **particular task**, based on available information.

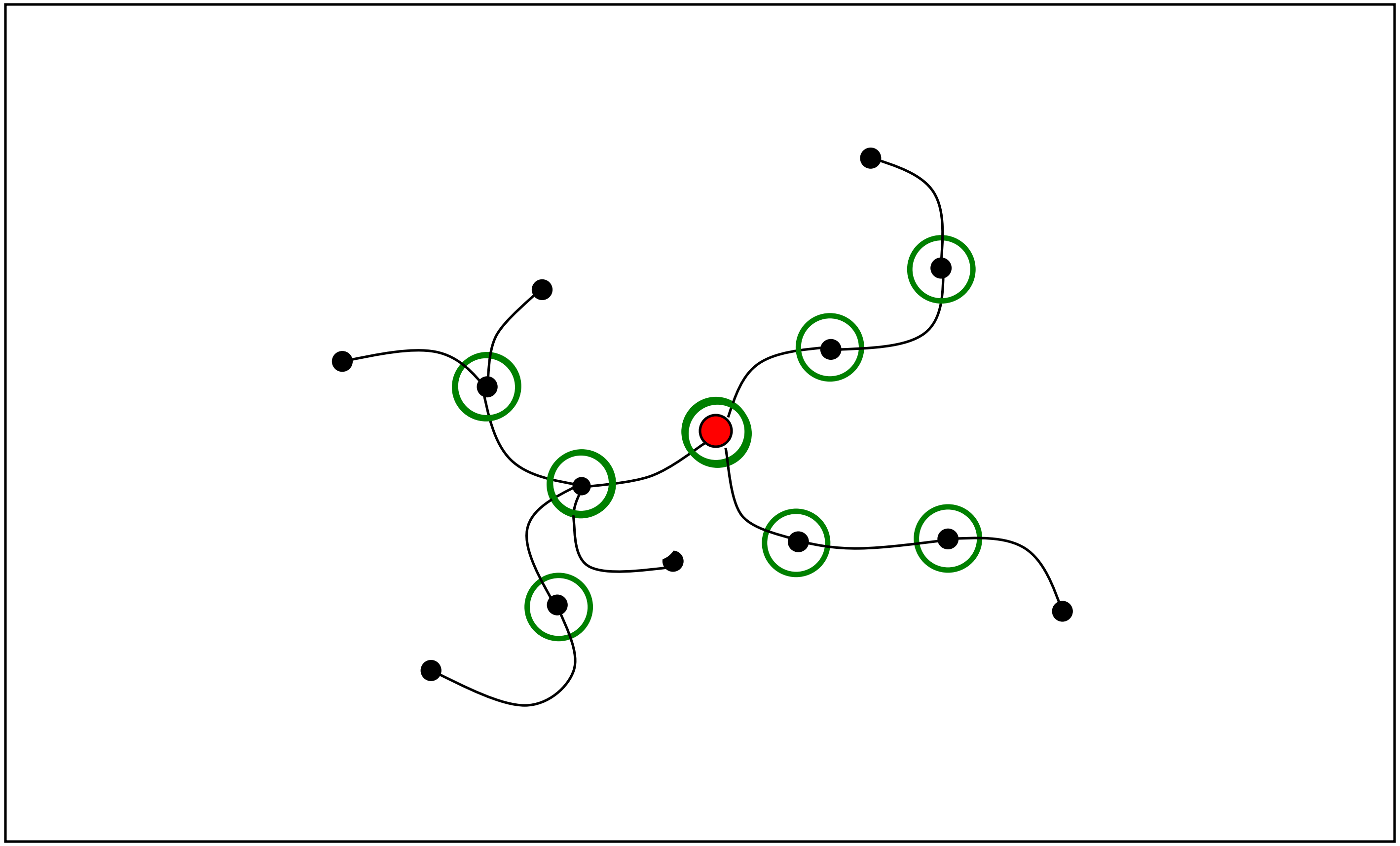
Robotics

Sampling-based Single Query Motion Planning

TU Berlin

Oliver Brock

Rapidly-Exploring Random Trees (RRT)



Rapidly-Exploring Random Trees (RRT)

T.add_vertex(q_{init})
Repeat k times:
    q_{rand} = SAMPLE()
    q_{near} = NEAREST_VERTEX(q_{random})
    q_{new} = LOCAL_PLANNER(q_{near}, q_{random},Δq)
    T.add_vertex(q_{new})
    T.add_edge(q_{near}, q_{new})
return(T)

Rapidly-Exploring Random Trees (RRT)

T.add_vertex(q_{init})

Repeat:

    q_{rand} = SAMPLE()

    q_{near} = NEAREST_VERTEX(q_{random})

    q_{new} = LOCAL_PLANNER(q_{near}, q_{random},Δq)

    if VALID(q_{near}, q_{new})

        T.add_vertex(q_{new})

        T.add_edge(q_{near}, q_{new})

        LOCAL_PLANNER(q_{new}, q_{goal},-)

        if VALID(q_{new}, q_{goal})

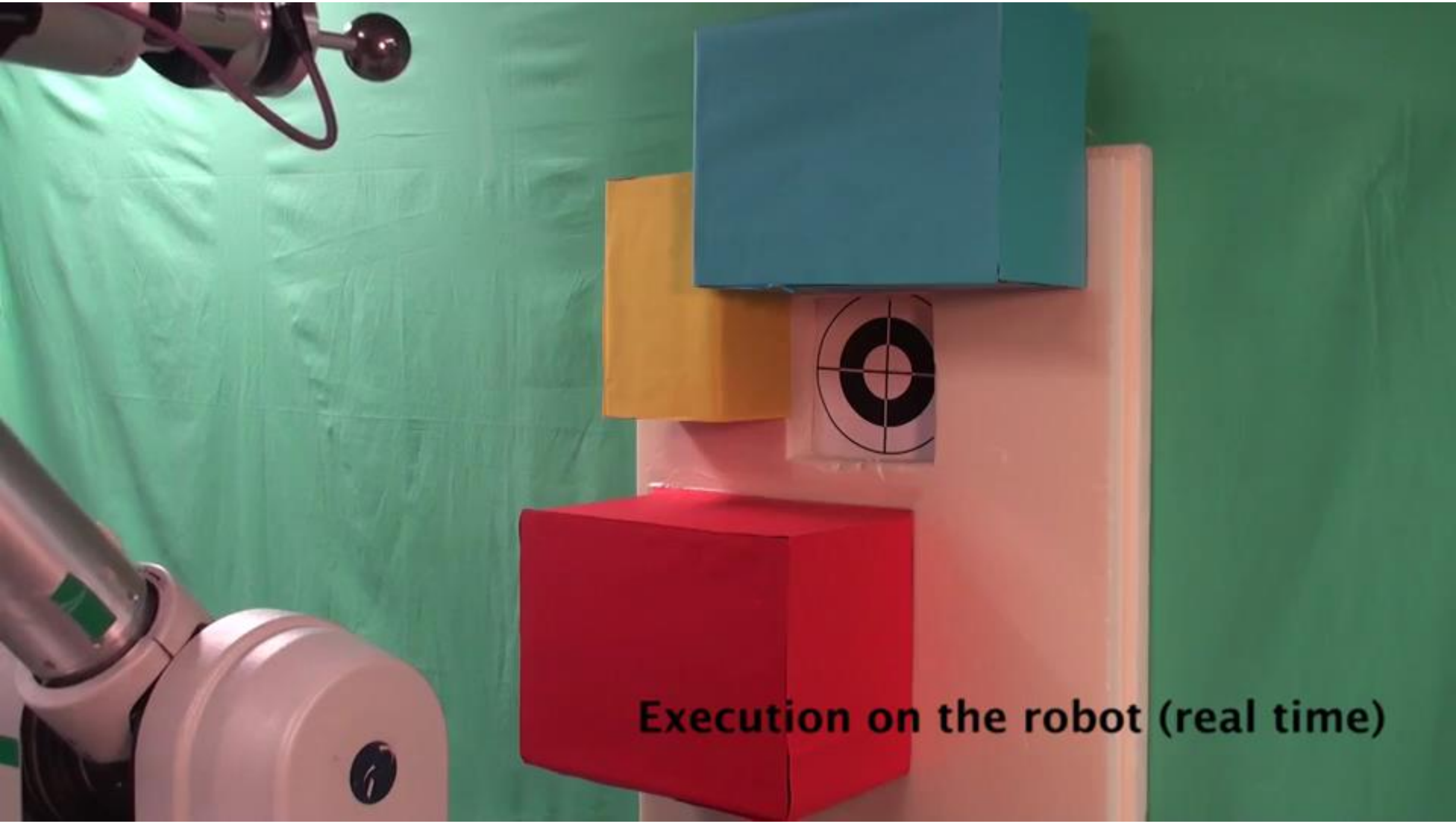
            return(T)

Kinodynamic Planning with RRT

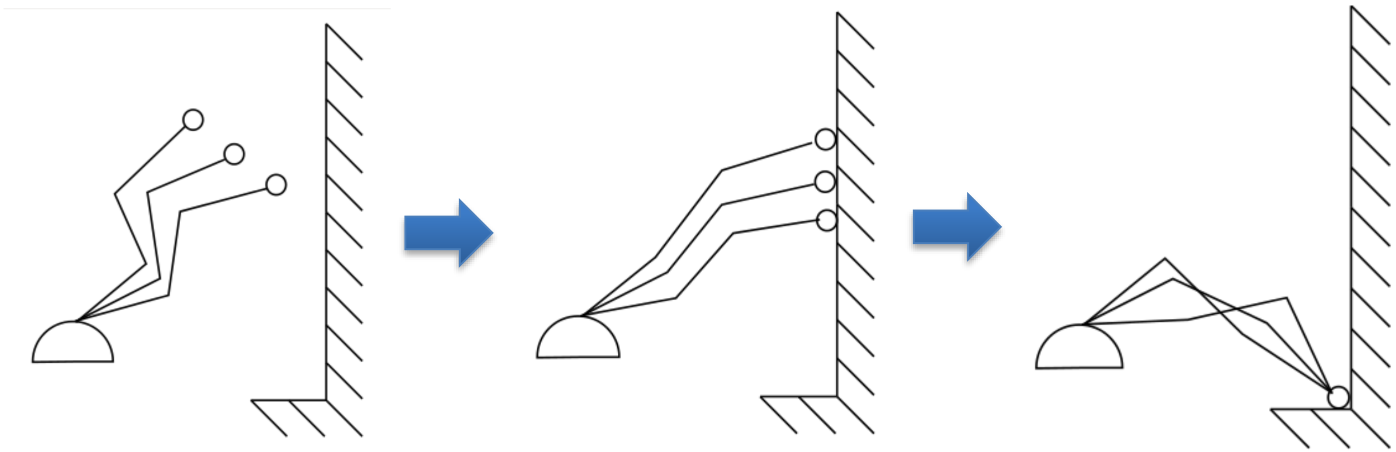
- Easy to integrate local planners for
  - Kinematic constraints
  - Dynamic constraints
- Expand C-space to state space for velocity representation (2 *d* dimensions)
- Requires known dynamic model (grasping!)
- Planning times for 7 dof ~ 10-20 seconds (holonomic, no dynamics)

What did we ignore until now?

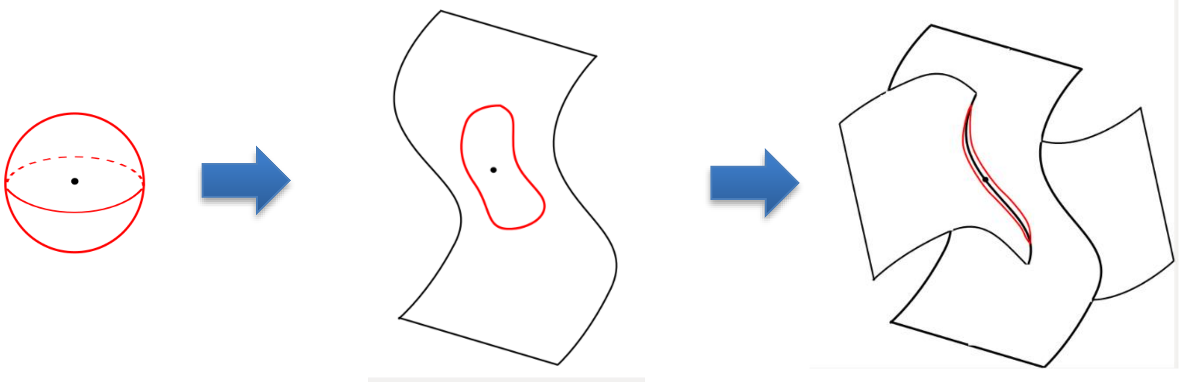
CERRT



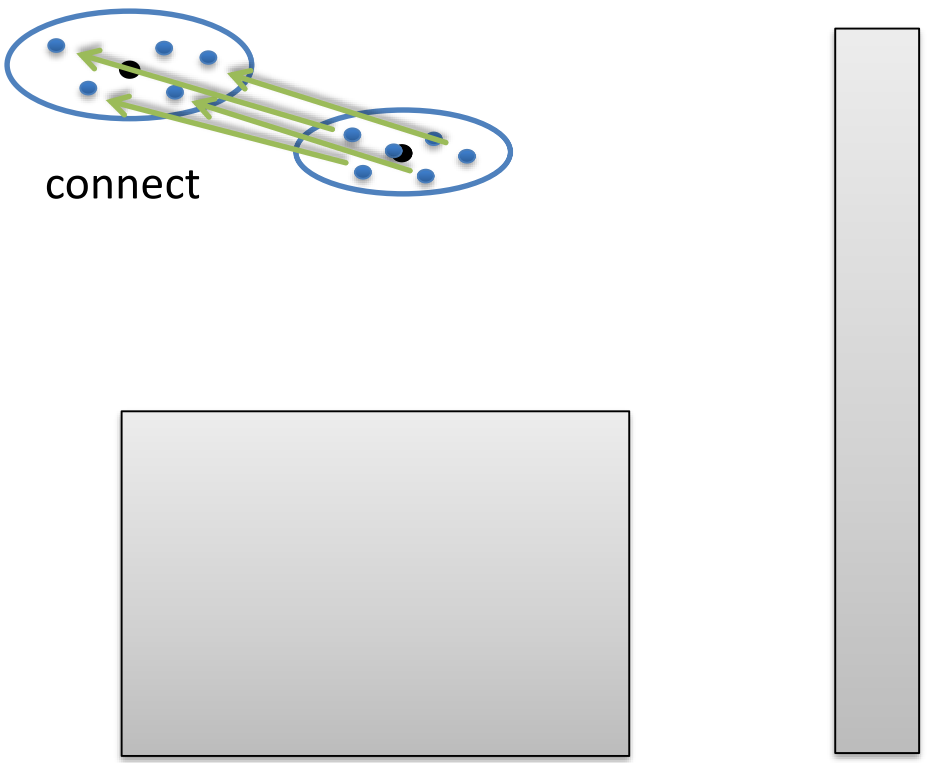
Contact Can Reduce Uncertainty



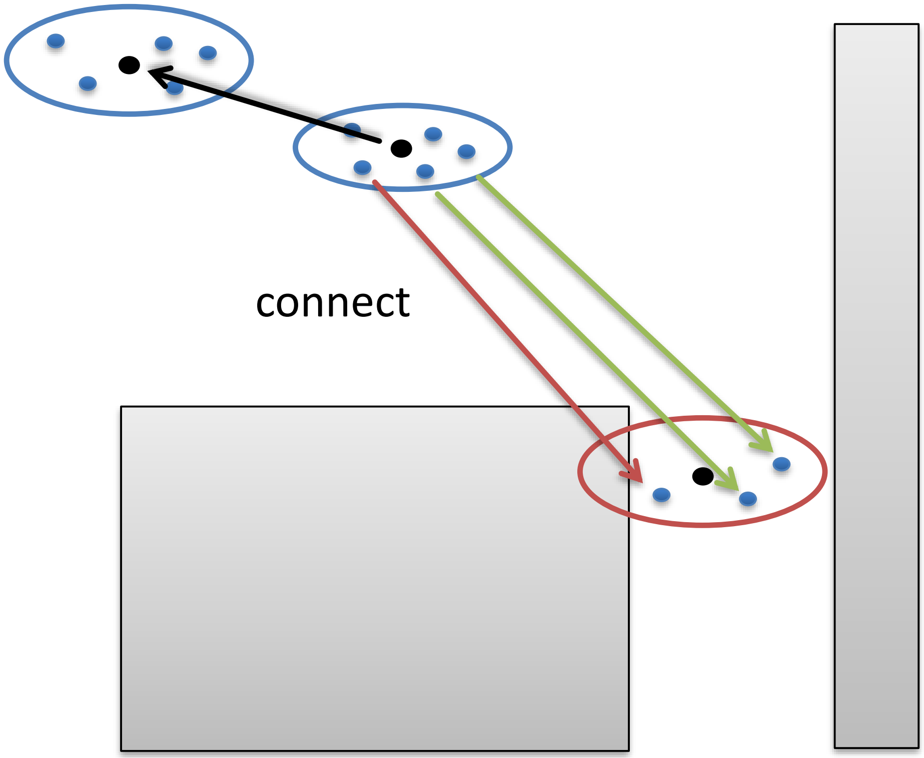
Configuration space



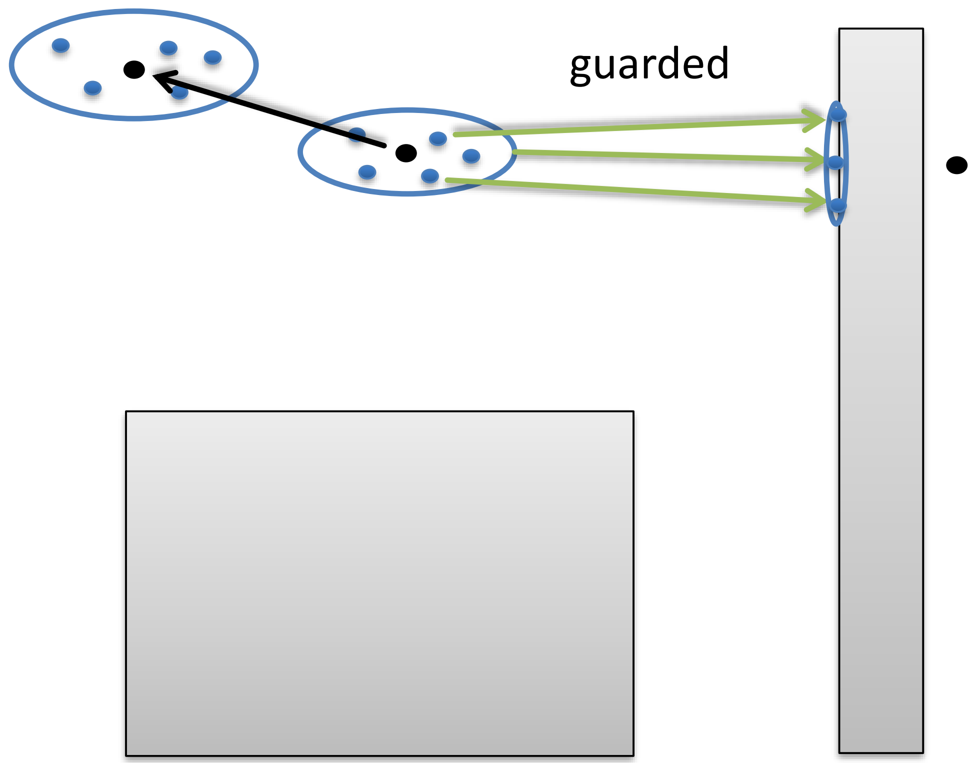
The Contact-Exploiting RRT



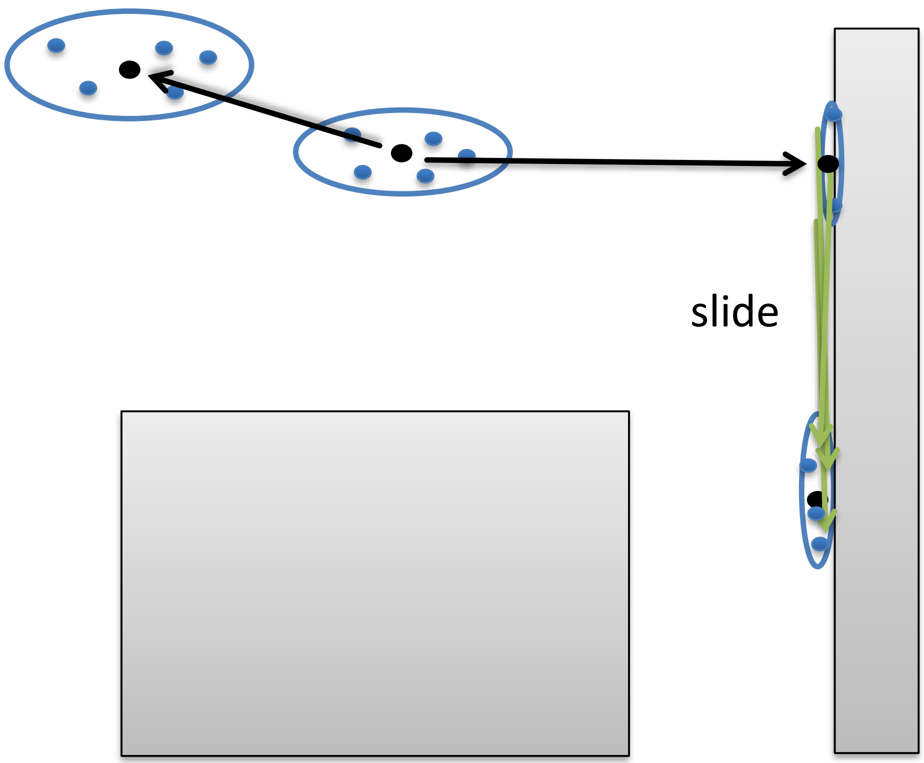
The Contact-Exploiting RRT



The Contact-Exploiting RRT

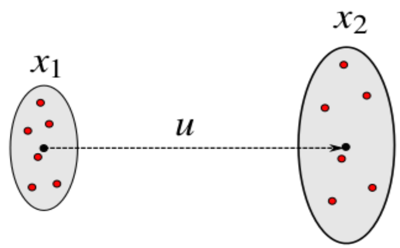


The Contact-Exploiting RRT

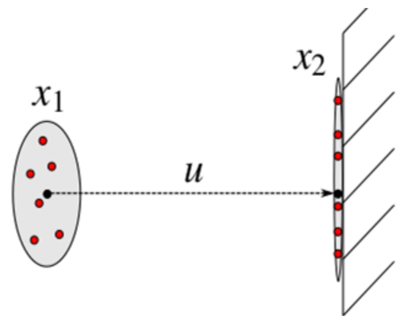


Types of local planners in CERRT

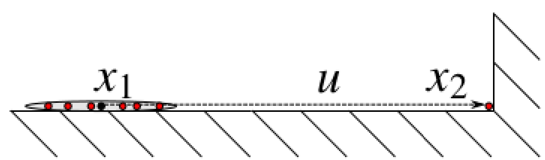
1. connect

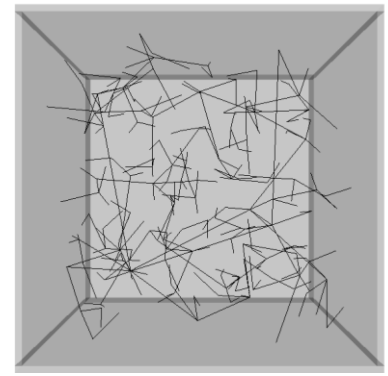


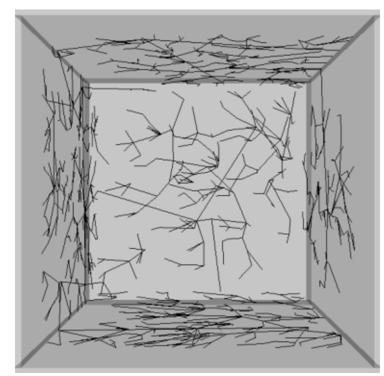
2. guarded



3. slide







Rapidly-Exploring Random Trees (RRT)

T.add_vertex(qinit)

Repeat:

q_{rand} = SAMPLE()

q_{near} = NEAREST_VERTEX(q_{random})

q_{new} = LOCAL_PLANNER(q_{near}, q_{random})

if VALID(q_{near}, q_{new})

T.add_vertex(q_{new})

T.add_edge(q_{near}, q_{new})

LOCAL_PLANNER(q_{new}, q_{goal},-)

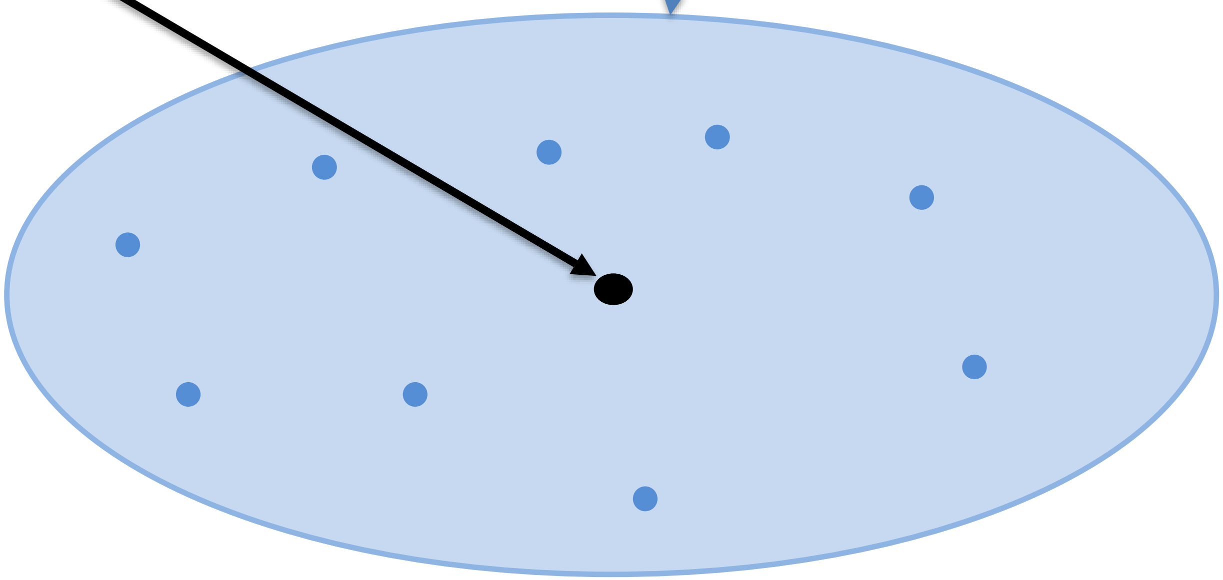
if VALID(q_{new}, q_{goal})

return(T)

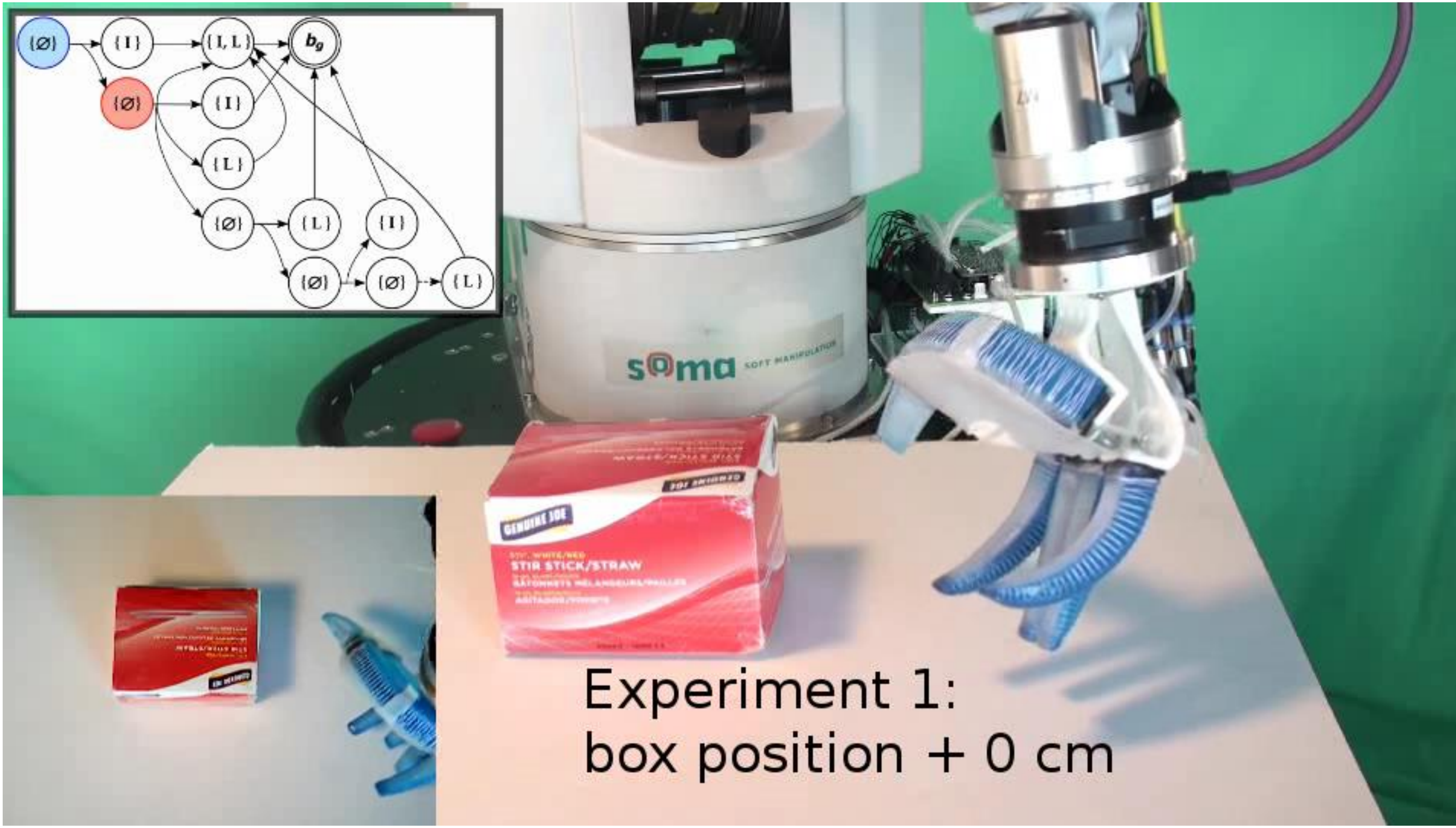
Nearest Neighbour

$$d(vertex, qrand) =$$

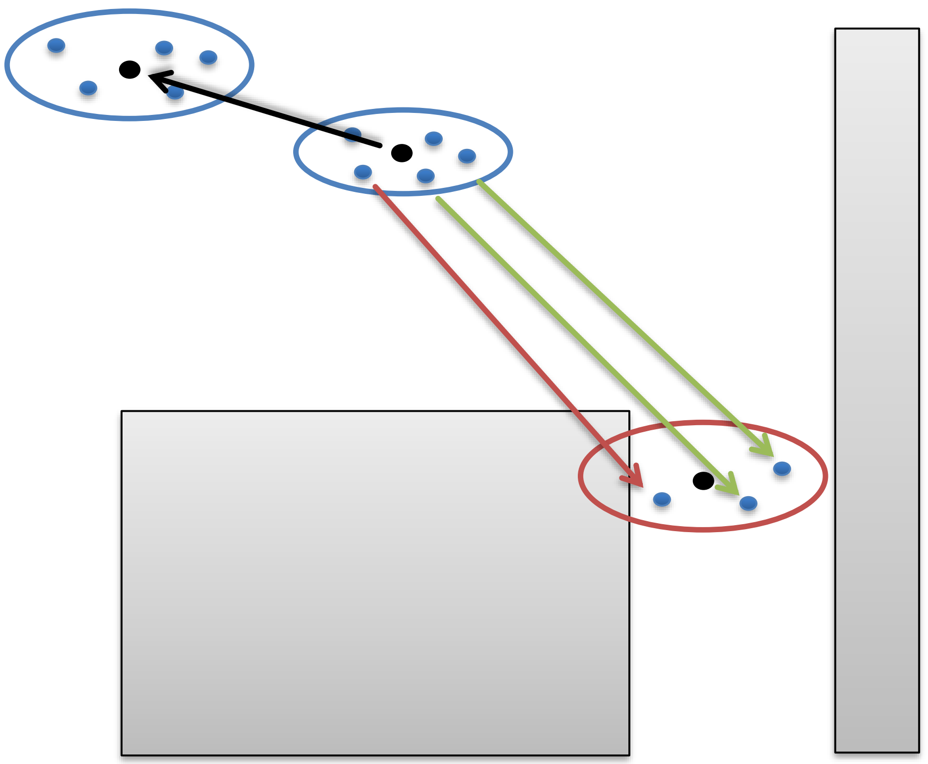
$$\gamma|\mu_{vertex}, qr_{and}| + (1\text{-}\gamma)\text{Uncertainty}_{vertex}$$



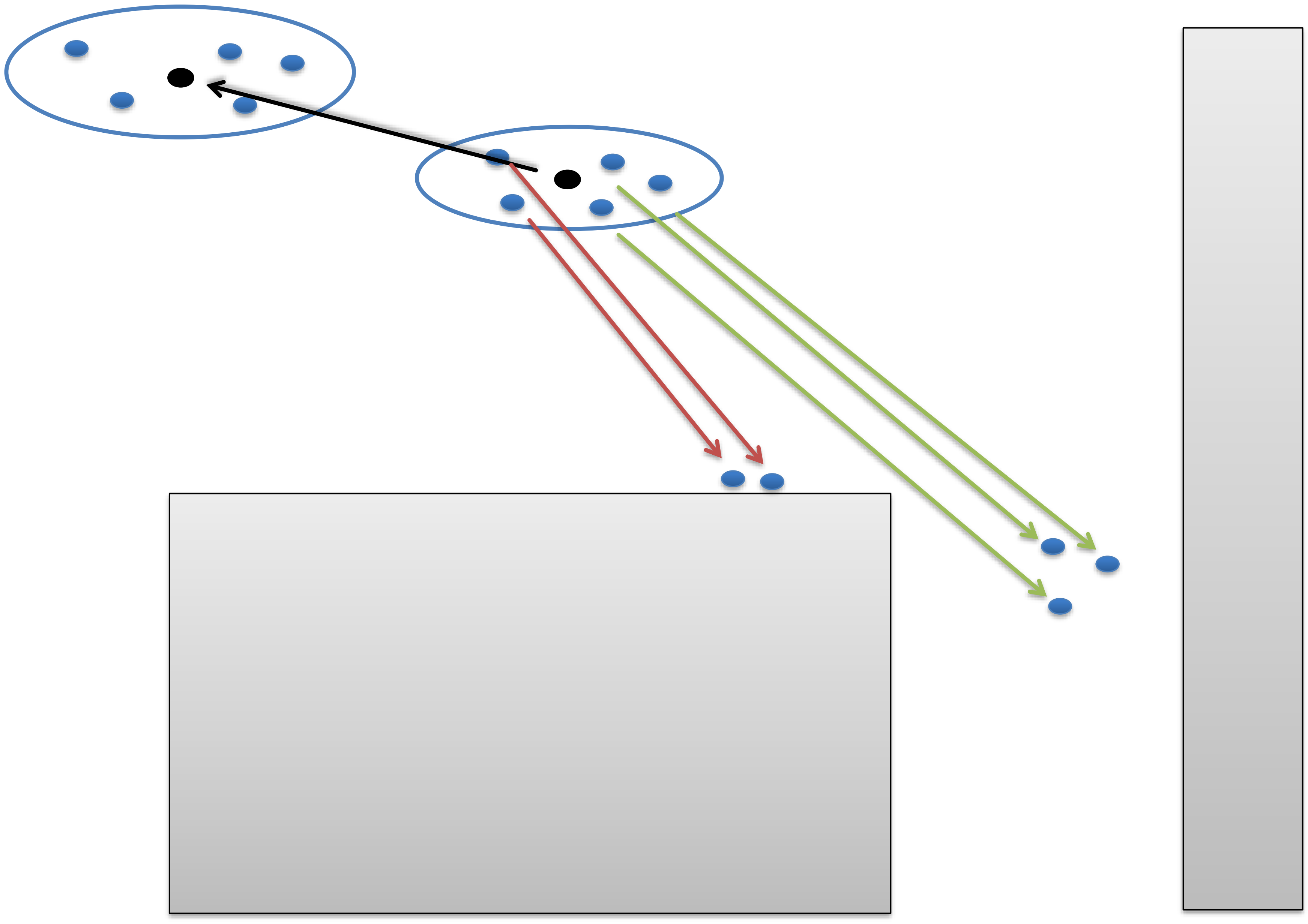
ConCERRT


{∅}
{I}
{I, L}
b_g
{∅}
{I}
{L}
{∅}
{L}
{I}
{∅}
{∅}
{L}
soma SOFT MANIPULATION
GENUINE JOE
STIR STICK/STRAW
Experiment 1:
box position + 0 cm

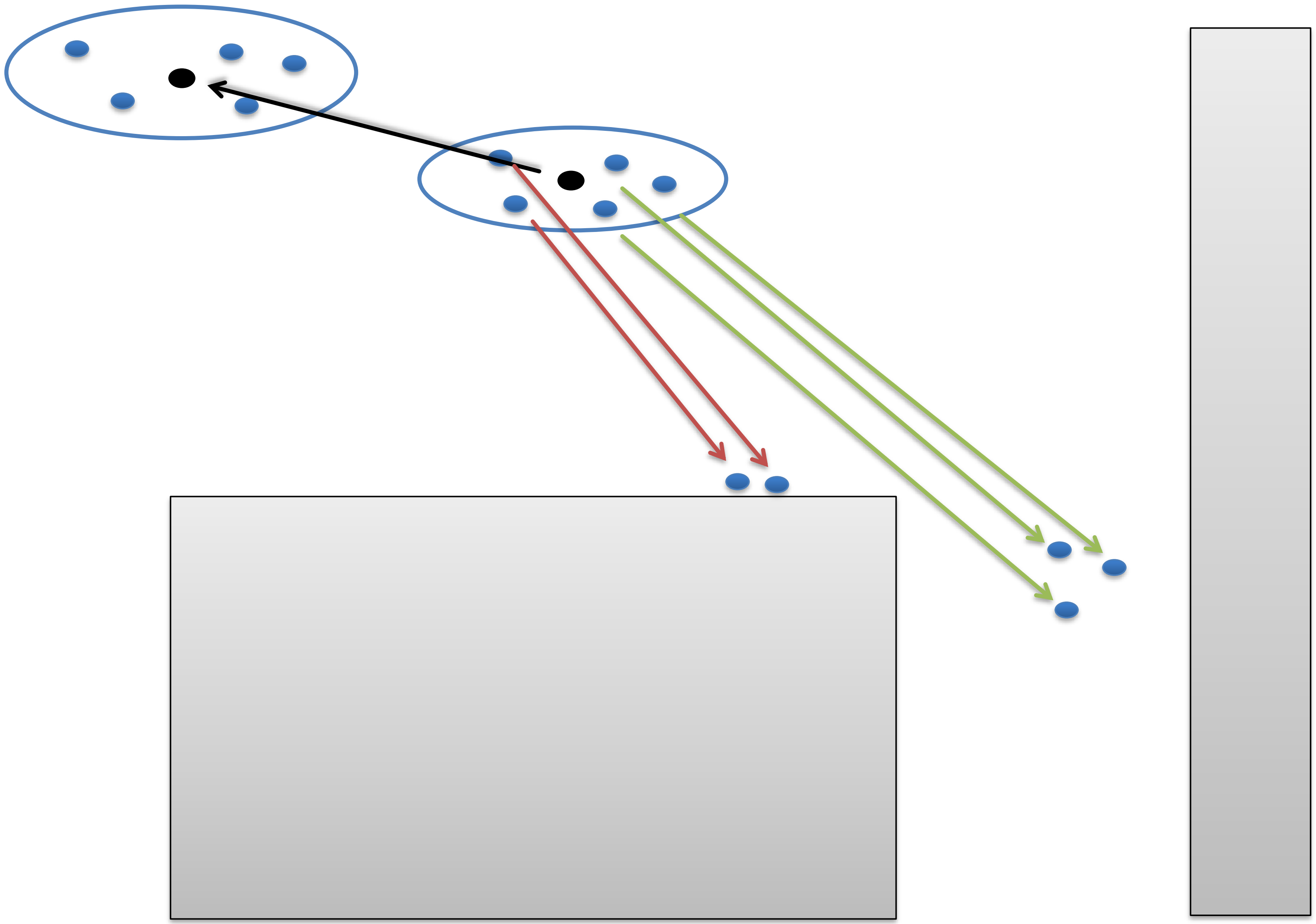
The Contact-Exploiting RRT



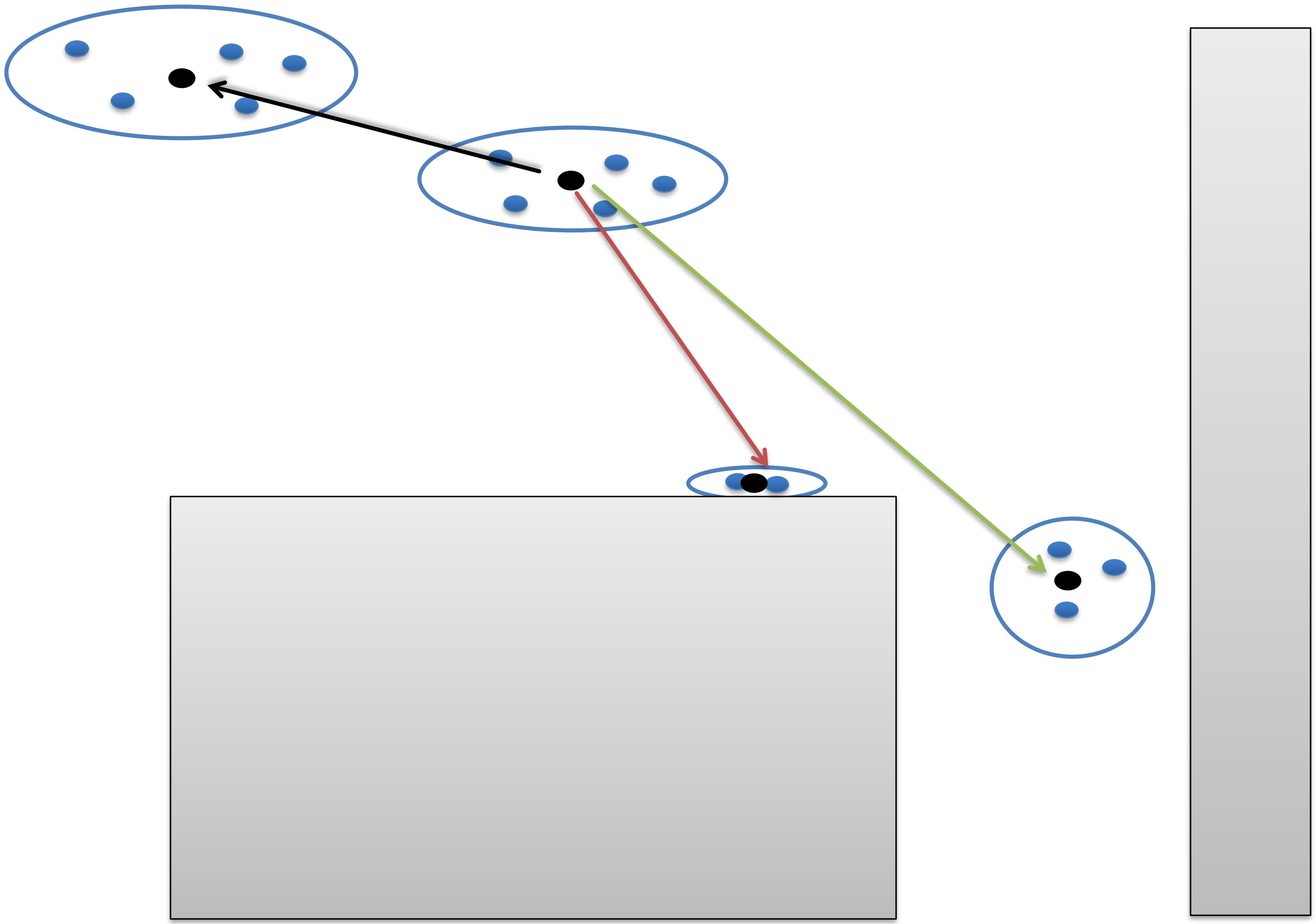
Contact structures the space



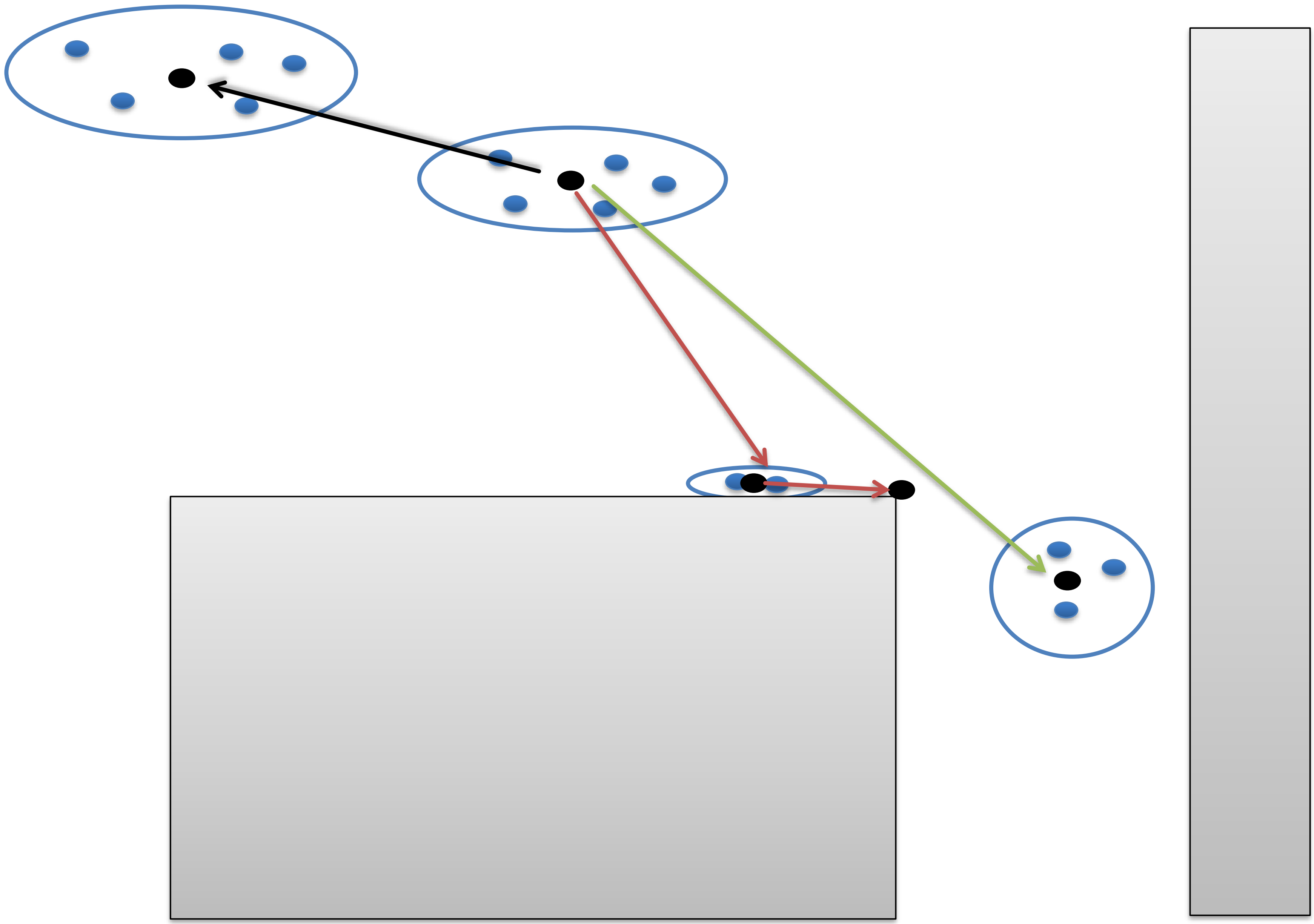
Contact structures the space



Contact structures the space



Contact structures the space



Contact structures the space