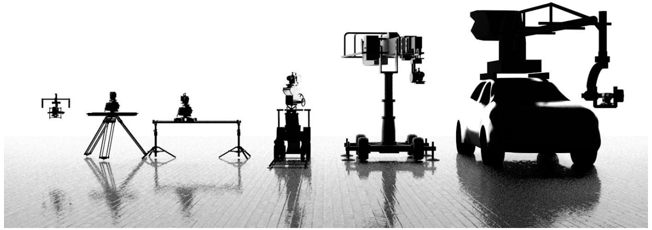
Motion Planning and Cinematography

Marc Christie
Univ Rennes 1 / INRIA Rennes (France)
BFA (China)

Who am I?

- Associate Professor in CS, University of Rennes 1, France
- Principal Investigator at BFA (China)



Research interests:

- Virtual cinematography: to transpose techniques/practice/style from real movies to virtual 3D environments?
  - **Understanding** movies by analysing elements of film style (composition, shots, ...)
  - **Computational models** for cinematography
  - **Generating** synthetic movies (moving cameras, creating cuts, laying out scenes)
  - In order to tell a story
- *Easing the control* over virtual cinematography
- Applications to games, interactive narratives, previsualisation

Topics addressed today

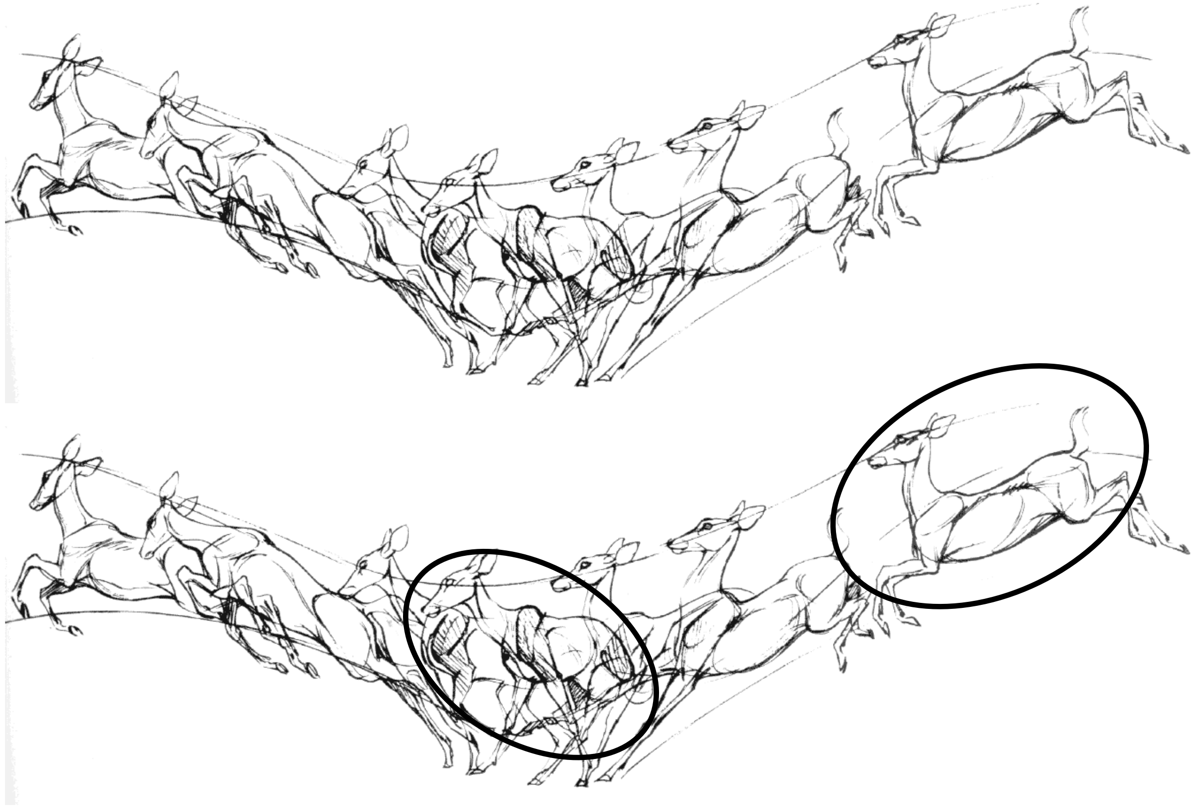
- Basics of keyframe animation (the 101 of animation)
- Introduction to motion planning (slides inspired by CS@CMU)
- Research in interactive drone cinematography (my own research)

Introduction

- Traditional animation

Introduction

- Walt Disney (1901-1966)
  - Studios created in 1923
  - Junior and senior animators
- Work organization
  - Rationalization
- “Senior = user”
- “Junior = computer”



Computer animation

- Computing the evolution of different parameters that impact on a scene representation
  - Constant shape: animation of poly-articulated systems
  - Variable shape: animation of deformable models

- What are these parameters ?
  - Position, orientation
  - Shape
  - Material (color, texture…)
  - Lights
  - Cameras
  - …

- Each scalar parameter that can change over time is called a degree of freedom (DOF)

Introduction

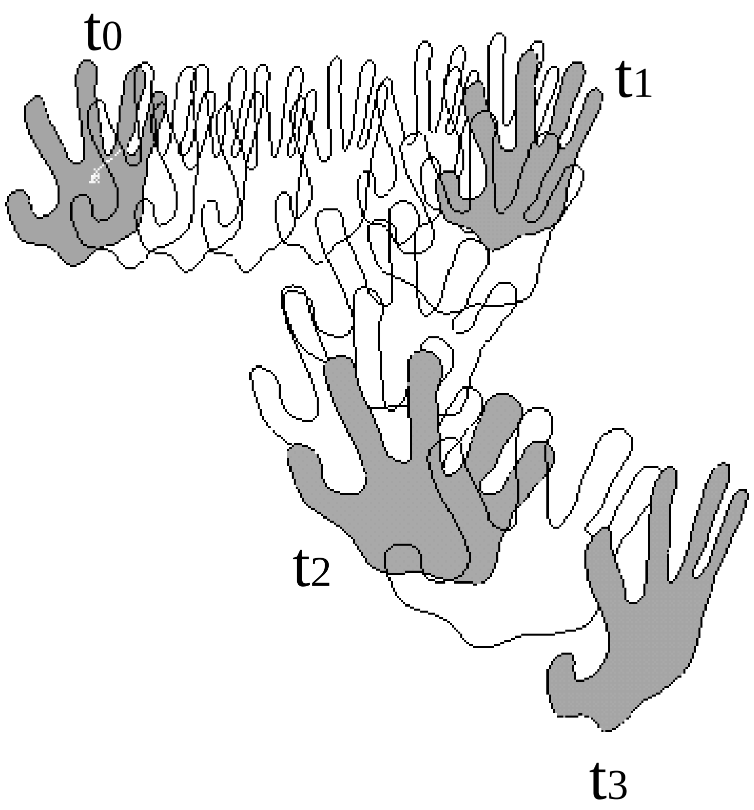
- Domains
  - Computer games
  - Virtual reality
  - Special effects
  - Computer-generated films
  - Simulation
  - Robotics
  - …

Keyframe interpolation

Used everywhere…

Principle

- Keyframe
  - A pair (time, set of parameters)
  - Define the state of a 3D object at a given time
  - Set of parameters
    - Position
    - Orientation
    - …



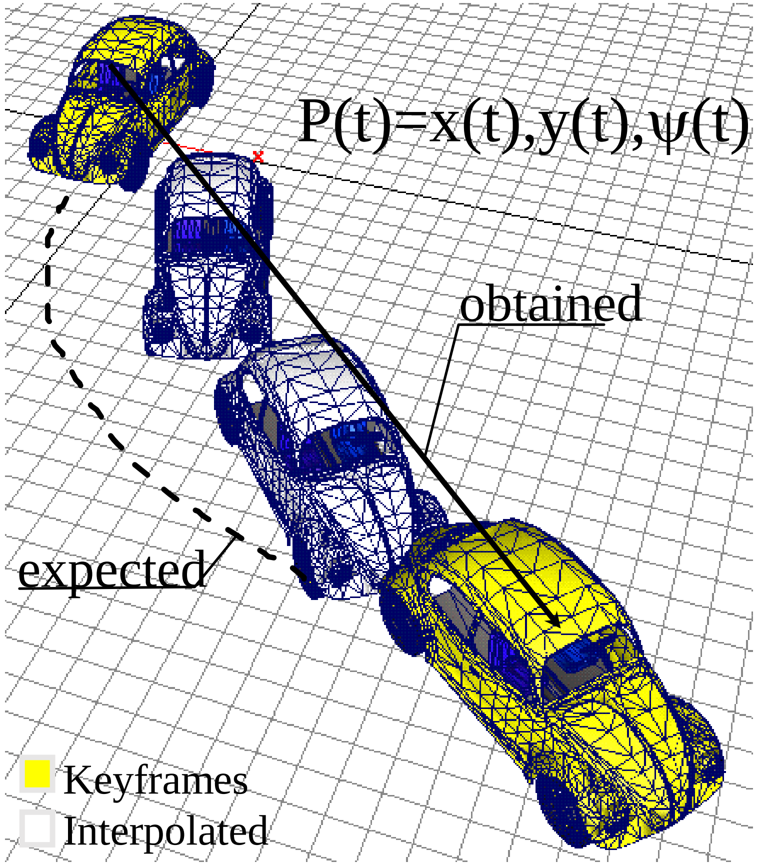
Principle

- Keyframes interpolation
  - Automatic generation of intermediate frames between keyframes
  - Interpolation of positions, orientations…
- Advantages: computationally low cost
- Can be used to
  - Reduce the computational cost when used with complex models
  - Mix movements (motion blending)
  - Generate intermediate frames (ex: slow motion)
  - …

A naïve approach

- Linear interpolation of positions and orientations
  - No control on trajectory and/or speed

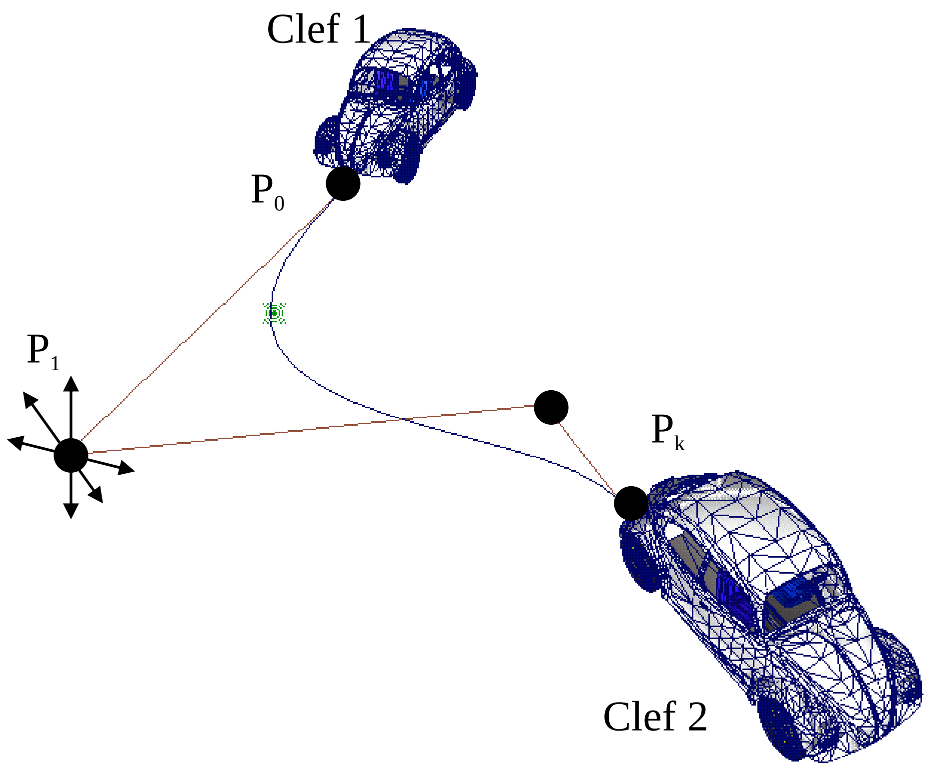
- We will focus on two aspects
  - Interpolation of positions
  - Interpolation of orientations



Interpolation of positions

- Trajectory: a parameterized curve $Q(u)$
- Control points $P_1, P_2, \dots, P_k$
  - Fine control of the trajectory
  - Easy description
- Finding a function:

$$\begin{cases} \mathbb{R} \to \mathbb{R}^3 \\ u \to Q_{P_1,P_2,\dots,P_k}(u) \end{cases}$$



Interpolation of positions

- Interpolation using linear combinations

$$\mathbf{Q}(u) = \begin{pmatrix} x(u) \\ y(u) \\ z(u) \end{pmatrix} = \overbrace{\sum_{k=0}^{K} b_k(u)\mathbf{P}_k}^{\text{Relative influence of } P_k \text{ when u varies from 0 to 1}} = \overbrace{\sum_{k=0}^{K} \mathbf{P}_k b_k(u)}^{\text{Linear combination of functions (ex: polynomial basis, degree K)}}, \quad u \in [0;1]$$

Relative influence of P_k when u varies from 0 to 1

Linear combination of functions (ex: polynomial basis, degree K)

$b_k(u)$: Scalar coefficients in [0;1]

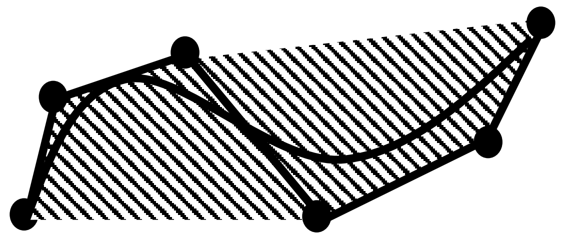
$\mathbf{P}_k$: Control points (R^3)

$\mathbf{P}_k$: Coefficients in R^3

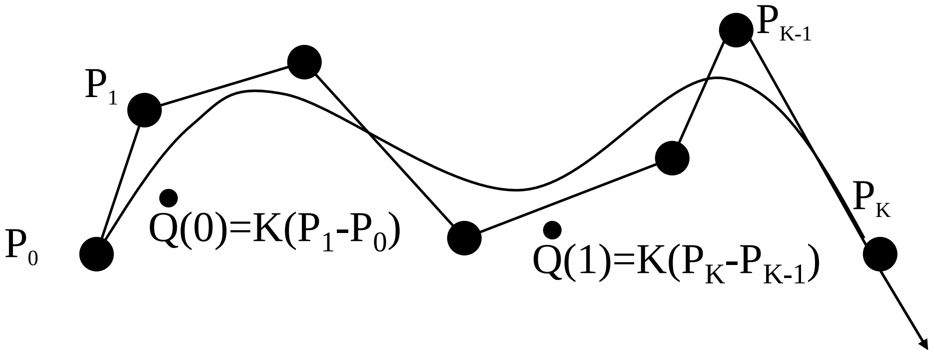
$b_k(u)$: Scalar functions, (ideally independent)

Interpolation of positions

- Parametric trajectories: desired properties
  - Independence of the effect of P_k (control points)
  - Continuity C^{K-1}
  - Affine invariance: $\varphi\big(Q(u)\big) = \sum_{k=0}^{K} \varphi(P_k) b_k(u)$
  - Convex hull: $\forall \mathrm{u}, \sum_k b_k(u) = 1$



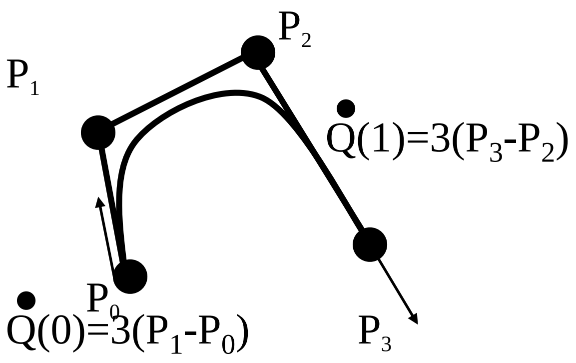
Bezier curves (1970)



- $Q(u) = \sum_k B_{\mathrm{k},K}(U)P_k \, , u \in [0;1]$
- $B_{n,k}(u) = C_n^K \mathrm{u}^n (1-u)^{K-n}$ (Bernstein polynomial)
- Some properties:
  - Manipulation of tangents at the beginning and the end of the curve
  - Affine invariance
  - Curve in the convex hull of control points

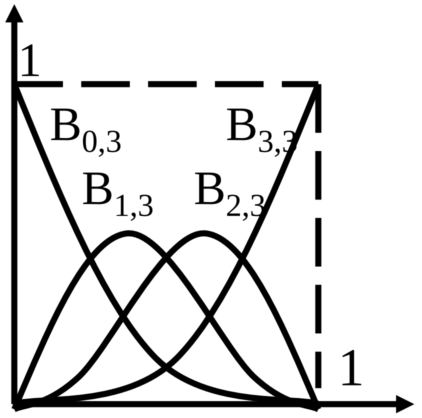
Cubic Bezier curves

- Advantage
  - Local control of the curve
  - Minimal degree to ensure C^2 continuity



Cubic Bezier curve

- $Q(U) = [u^3\ u^2\ u\ 1] \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{bmatrix}$

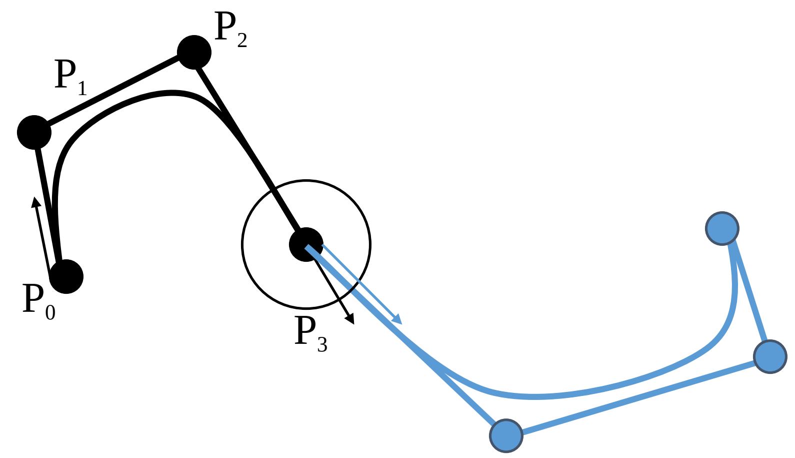


Bernstein polynomials (degree 3)

Cubic Bezier curves

- Stitching Cubic Bezier curves
  - Provides a better local control

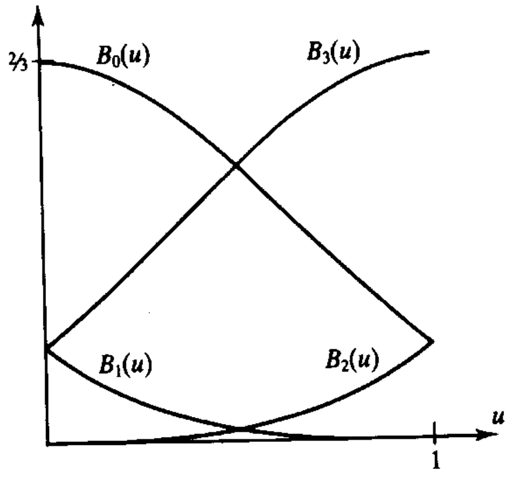
    But
  - Warning on the junction!



- $Q_i(u) = \sum_{k=0}^{3} P_{k+3i} B_{\mathrm{k,3}}(u), u \in [0; 1]$

B-Splines

$$
\begin{array}{lll}
\mathbf{B}_0(1) = 0 & \mathbf{B}'_0(1) = 0 & \mathbf{B}''_0(1) = 0 \\
\mathbf{B}_1(1) = \mathbf{B}_0(0) & \mathbf{B}'_1(1) = \mathbf{B}'_0(0) & \mathbf{B}''_1(1) = \mathbf{B}''_0(0) \\
\mathbf{B}_2(1) = \mathbf{B}_1(0) & \mathbf{B}'_2(1) = \mathbf{B}'_1(0) & \mathbf{B}''_2(1) = \mathbf{B}''_1(0) \\
\mathbf{B}_3(1) = \mathbf{B}_2(0) & \mathbf{B}'_3(1) = \mathbf{B}'_2(0) & \mathbf{B}''_3(1) = \mathbf{B}''_2(0) \\
0 = \mathbf{B}_3(0) & 0 = \mathbf{B}'_3(0) & 0 = \mathbf{B}''_3(0)
\end{array}
$$



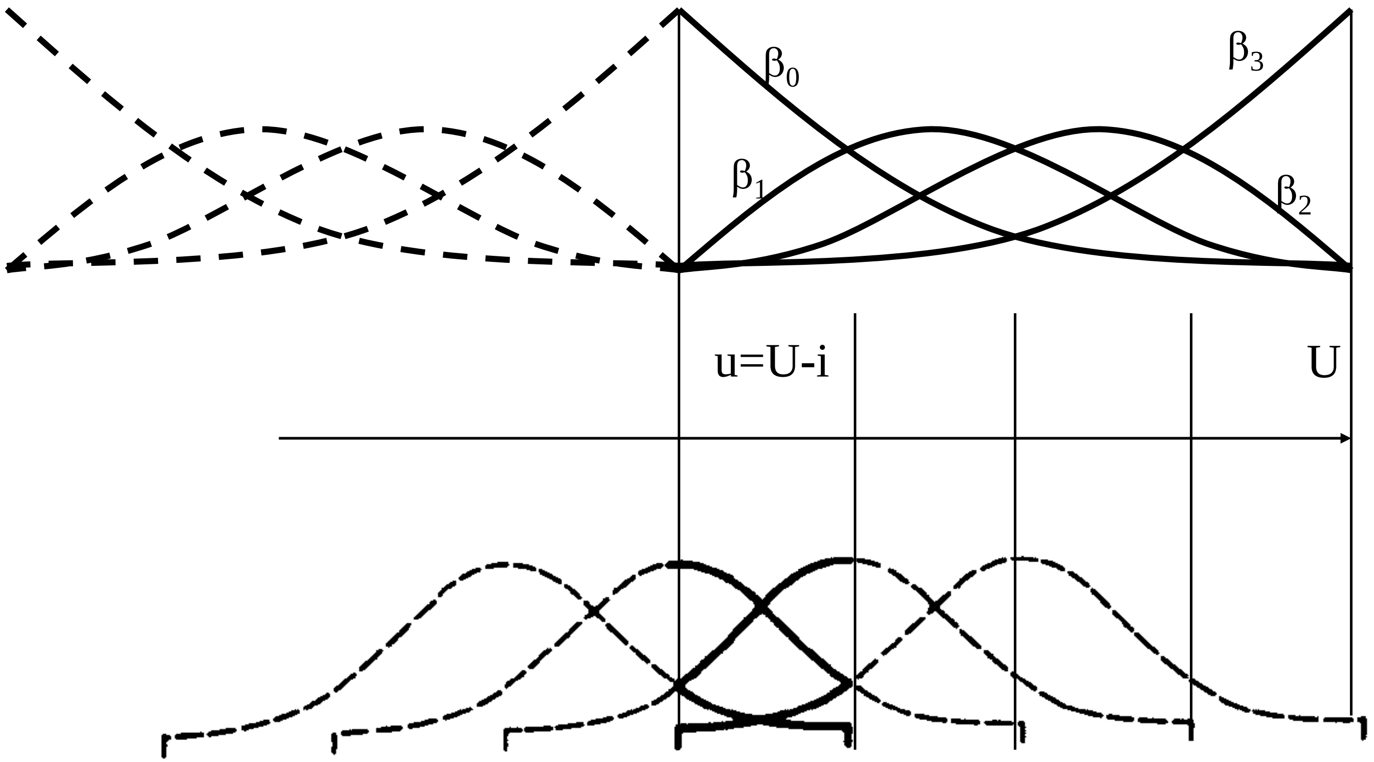
The B-spline basis functions.

- $Q_i(u) = \sum_{k=0}^{3} P_{i+k} B_k(u) = [u^3\ u^2\ u\ 1] \begin{bmatrix} -1 & 3 & -3 & 2 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix} \begin{bmatrix} P_i \\ P_{\mathrm{i}+1} \\ P_{i+2} \\ P_{i+3} \end{bmatrix}$

B-splines vs Bezier

- Bezier

- B-spline
    - Ensures C^2 continuity

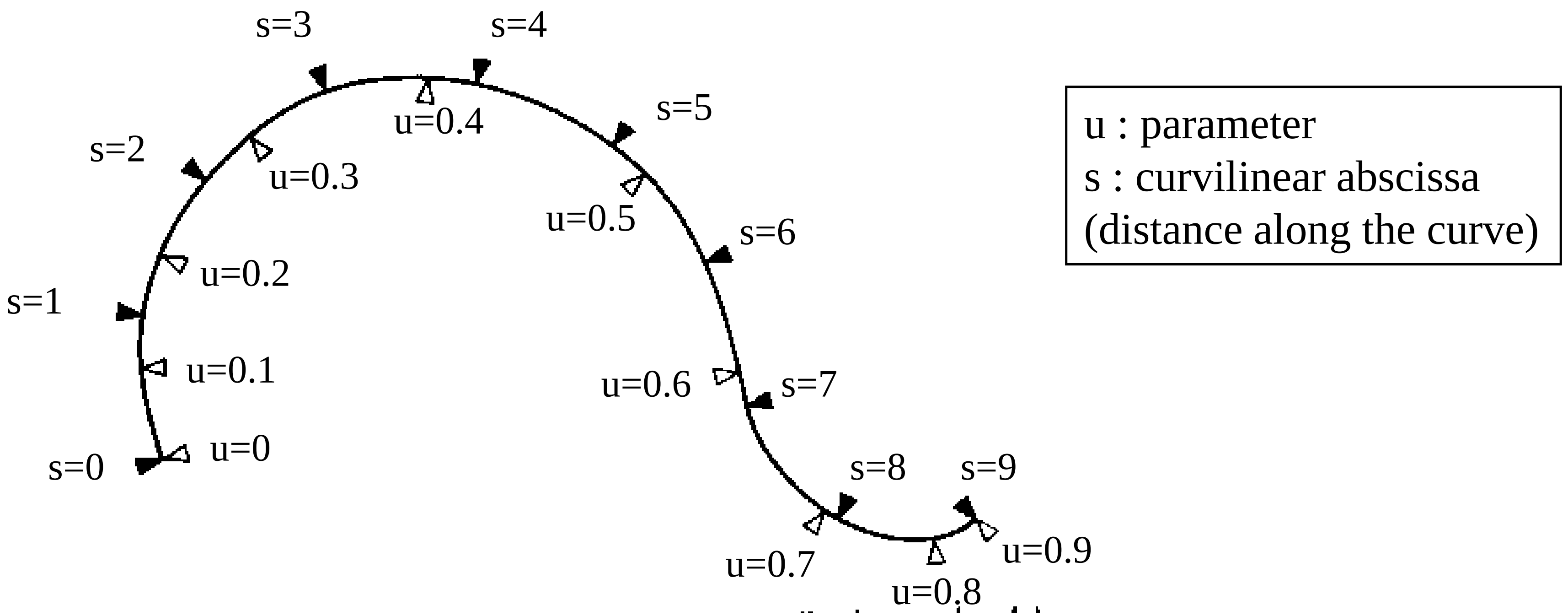


Hermite polynomial basis

- $$\begin{cases} b_0(u) = 2u^3 - 3u^2 + 1 \\ b_1(u) = -2u^3 + 3u^2 \\ b_2(u) = u^3 - 2u^2 + u \\ b_3(u) = \ u^3 - u^2 \end{cases}$$ $$Q(u) = \sum_{i=0}^{3} b_i(u) P_i(u)$$
- $$Q(u) = [u^3\ u^2\ u^1\ 1] \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} P_i \\ P_{i+1} \\ D_i \\ D_{i+1} \end{bmatrix}$$
- Where (P_i, D_i) is a couple (starting point, tangent at the starting point) and (P_{i+1}, D_{i+1}) is a couple (ending point, tangent at the ending point)

Trajectory following

- Path described by $Q(u), u \in [0; 1]$
- Problem: how to control speed when moving along the curve?



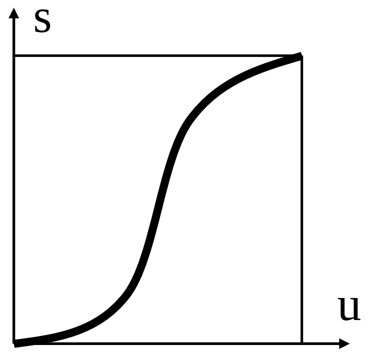
Trajectory following

- Use the curvilinear abscissa to control movement along the path

$$\mathbf{Q}(u) = \mathbf{a}u^3 + \mathbf{b}u^2 + \mathbf{c}u + \mathbf{d} \Rightarrow \begin{cases} x(u) = a_x u^3 + b_x u^2 + c_x u + d_x \\ y(u) = a_y u^3 + b_y u^2 + c_y u + d_y \\ z(u) = a_z u^3 + b_z u^2 + c_z u + d_z \end{cases}$$

$$s = g(u) \Rightarrow \mathbf{Q}(u) = \mathbf{Q}(g^{-1}(s))$$

$$\left\| \frac{Q(u)}{du} \right\| = \underbrace{\sqrt{\underbrace{\left(\frac{dx}{du}\right)^2 + \left(\frac{dy}{du}\right)^2 + \left(\frac{dz}{du}\right)^2}_{\text{degree 4 polynomial: } Au^4+Bu^3+Cu^2+Du+E}}}_{\text{Derivative of the curvilinear abscissa}} = \frac{ds}{du}$$



$$s(u) = \int_{u_0}^{u} ds = \int_{u_0}^{u} \sqrt{Au^4 + Bu^3 + Cu^2 + Du + E}\, du$$

Interpolation of positions

- Choosing the good interpolator
  - If manually edited
    - Bezier curves are easy to manipulate
    - B-splines guaranty the C^2 continuity
  - If an object is moving along a trajectory
    - Hermite polynomial basis can be used
    - Start and end positions, start and end velocities

- Another solution
  - Given positions, speeds, accelerations etc…
  - Find the coefficients of a polynomial by solving a simple linear system

Interpolation of rotations

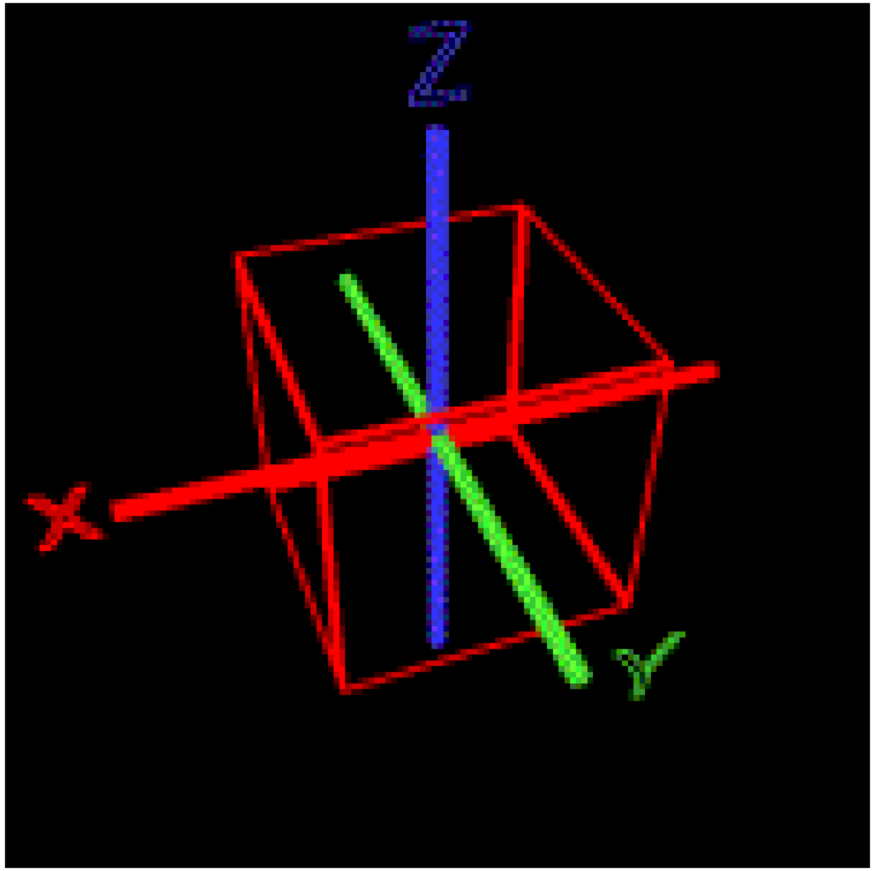
- Problem when using 3 angles to represent rotations (Euler or others)
  - The gimbal lock problem:
  - $R\left(\theta_1, \frac{\pi}{2}, \theta_3\right) = \begin{bmatrix} 0 & 0 & -1 & 0 \\ \sin(\theta_1 - \theta_3) & \cos(\theta_1 - \theta_3) & 0 & 0 \\ \cos(\theta_1 - \theta_3) & -\sin(\theta_1 - \theta_3) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

➢ Loss of one degree of freedom

➢ Rotation around Z becomes a rotation around the original X axis

➢ *Representation is not unique!*

The gimbal lock problem



Interpolation of rotations

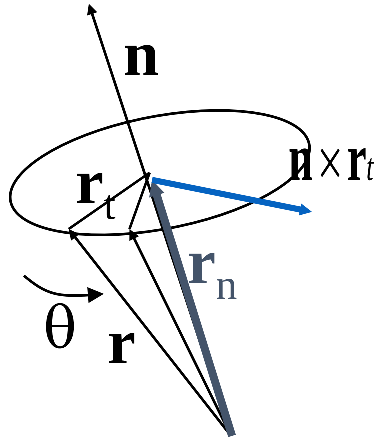
- Extension of the notion of complex numbers (Hamilton, 1843)
  - $q = a + b\boldsymbol{i} + c\boldsymbol{j} + d\boldsymbol{k}$, also noted $(s, v), s \in \mathbb{R}, v \in \mathbb{R}^3$

    With $\begin{cases} i^2 = j^2 = k^2 = -1 \\ ij = k \\ ji = -k \end{cases}$

  - Group structure
    - $q_1 q_2 = (s_1 s_2 - v_1 v_2, s_1 v_2 + s_2 v_1 + v_1 \times v_2)$
  - Conjugate quaternion
    - $q\bar{q} = (s, v)(s, -v) = s^2 + |v|^2 = |q|^2$
  - Reciprocal quaternion
    - $q^{-1} = \frac{\bar{q}}{|q|^2}$

Interpolation of rotations

- Let $\begin{cases} p = (0, r) \\ q = (s, v) \end{cases}$ with $|q| = 1$
- q can be rewritten $(\cos\theta\,, \sin\theta\, n), n \in \mathbb{R}^3$
- Interpretation of the operation $qp\bar{q} (= qpq^{-1})$



$$qp\bar{q} = (0, \cos(2\theta)\, r + (1 - \cos(2\theta)\, n(n.r) + \sin(2\theta)\, n \times r)$$

$$= (0, \underbrace{(n.r)n}_{r_n} + \underbrace{\cos(2\theta)\, \underbrace{\big(r - n(n.r)\big)}_{r_t} + \sin(2\theta)\, \underbrace{n \times r}_{n \times r_t}}_{R_{n,2\theta}(r_t)})$$

Interpolation of rotations

- Let $\begin{cases} p = (0, r) \\ q = (s, v) \end{cases}$ with $|q| = 1$
- q can be rewritten $(\cos\theta, \sin\theta\, n), n \in \mathbb{R}^3$
- We can conclude that the operation $qp\bar{q}(= qpq^{-1})$ computes a rotation of r around n axis with angle 2θ

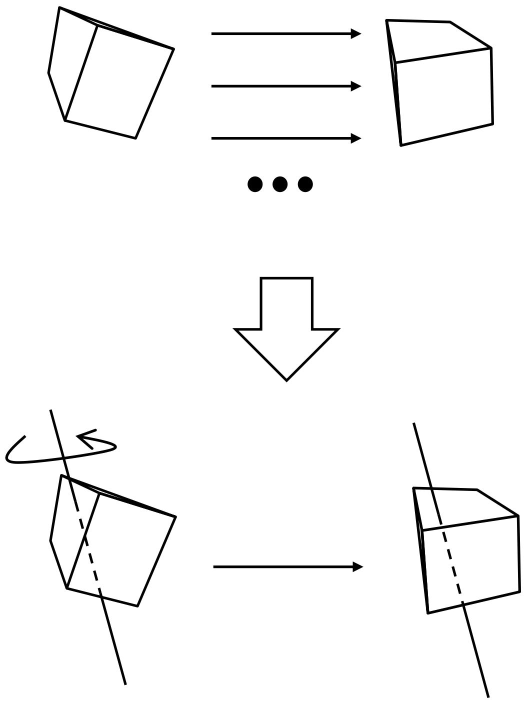
Interpolation of rotations

- Interest of quaternions
  - Unique representation of a rotation of angle θ around axis n

$$q = (\cos\frac{\theta}{2}, \sin\frac{\theta}{2}\ n), |n| = 1$$

  - Two successive rotations of angle π around Z and Y are equivalent to a rotation of angle π around X
  - With quaternions
    - $\big(0,(0,1,0)\big)\big(0,(0,0,1)\big) = \big(0,(0,1,0)\times(0,0,1)\big) = (0,(1,0,0))$
    - That's it!

➢ *Provides a solution to the gimbal lock problem*



Interpolation of rotations

- From quaternions to homogeneous matrices

$$(0, p') = q(0, p)q^{-1}, q = \big(W, (X, Y, Z)\big) = (\cos\frac{\theta}{2}, \sin\frac{\theta}{2} n)$$

$$p' = \begin{bmatrix} 1 - 2Y^2 - 2Z^2 & 2XY - 2WZ & 2XZ + 2WY & 0 \\ 2XY + 2WZ & 1 - 2X^2 - 2Z^2 & 2YZ - 2WX & 0 \\ 2XZ - 2WY & 2YZ + 2WX & 1 - 2X^2 - 2Y^2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} p$$

Interpolation of rotations

- From homogeneous matrices to quaternions

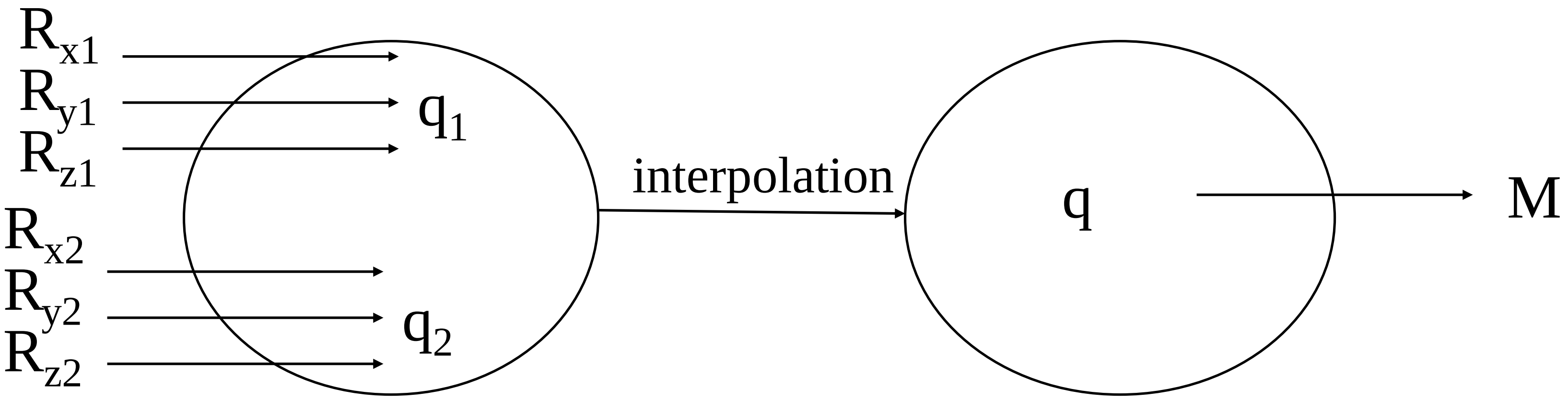
$$M = \begin{bmatrix} 1-2Y^2-2Z^2 & 2XY-2WZ & 2XZ+2WY & 0 \\ 2XY+2WZ & 1-2X^2-2Z^2 & 2YZ-2WX & 0 \\ 2XZ-2WY & 2YZ+2WX & 1-2X^2-2Y^2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} M_{11} & M_{12} & M_{13} & 0 \\ M_{21} & M_{22} & M_{23} & 0 \\ M_{31} & M_{32} & M_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$Trace(M) = 4(1-X^2-Y^2-Z^2) = 4(1-|q|^2+W^2) \Rightarrow W = \frac{1}{2}\sqrt{Trace(M)}$$

$$W \neq 0 \;\Rightarrow X = \frac{M_{32}-M_{23}}{4W}, Y = \frac{M_{13}-M_{31}}{4W}, Z = \frac{M_{21}-M_{12}}{4W}$$

Interpolation of orientations

- To interpolate between orientations
  - Round trip in quaternion space



Quaternion interpolation

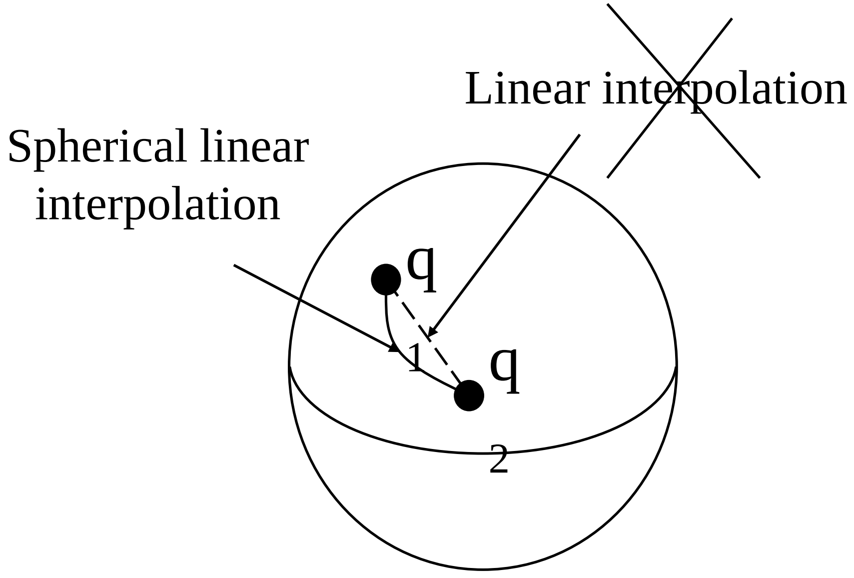
- $$\begin{cases} p = \alpha p_1 + \beta p_2, |p| = 1, |p_1| = 1, |p_2| = 1 \\ p_1 . p_2 = \cos\Omega \\ p_1 . p = \cos\theta \end{cases}$$

$$\Rightarrow \begin{cases} \alpha^2 + \beta^2 + 2\alpha\beta\cos\Omega = 1 \\ \alpha + \beta\cos\Omega = \cos\theta \end{cases} \Rightarrow \begin{cases} (\alpha + \beta\cos\Omega)^2 + \beta\sin^2\Omega = 1 \\ \alpha + \beta\cos\Omega = \cos\theta \end{cases}$$

$$\Rightarrow \begin{cases} \cos^2\theta + \beta^2\sin^2\Omega = 1 \\ \alpha + \beta\cos\Omega = \cos\theta \end{cases} \Rightarrow \begin{cases} \beta^2 = \dfrac{1 - \cos^2\theta}{\sin^2\Omega} \\ \alpha + \beta\cos\Omega = \cos\theta \end{cases}$$

$$\Rightarrow p = p_1 \frac{\sin(\Omega - \theta)}{\sin\Omega} + p_2 \frac{\sin\theta}{\sin\Omega}$$

$$slerp(p_1, p_2, u) = p_1 \frac{\sin((1-u)\Omega)}{\sin\Omega} + p_2 \frac{\sin(u\Omega)}{\sin\Omega}, u \in [0; 1]$$



Hypersphere of unit quaternions

Quaternion interpolation

- Slerp: Spherical linear interpolation
  - Produces the straightest and shortest path between unit quaternions
  - Known to generate “natural” rotations when used to interpolate quaternions
  - Can be used with any unit N-D vector
- So remember
  - To avoid problems with Euler angles, use quaternions!
  - Most of 3D animation toolkits use quaternions in combination with homogeneous coordinates and 4x4 matrices

Conclusion

- Interpolation techniques are used everywhere
  - Allow to generate intermediate states from known / computed states
  - Low computational cost
- Splines can be used to interpolate
  - Positions, Orientation with Euler angles (not advised to), colors…
  - Anything that needs to be interpolated (trivial extension to N-D)
- Quaternions are used to interpolate orientations
  - Solution to the gimbal lock i.e. unicity of the representation of a rotation

Motion Planning

Representations for motion planning

Planning trajectories

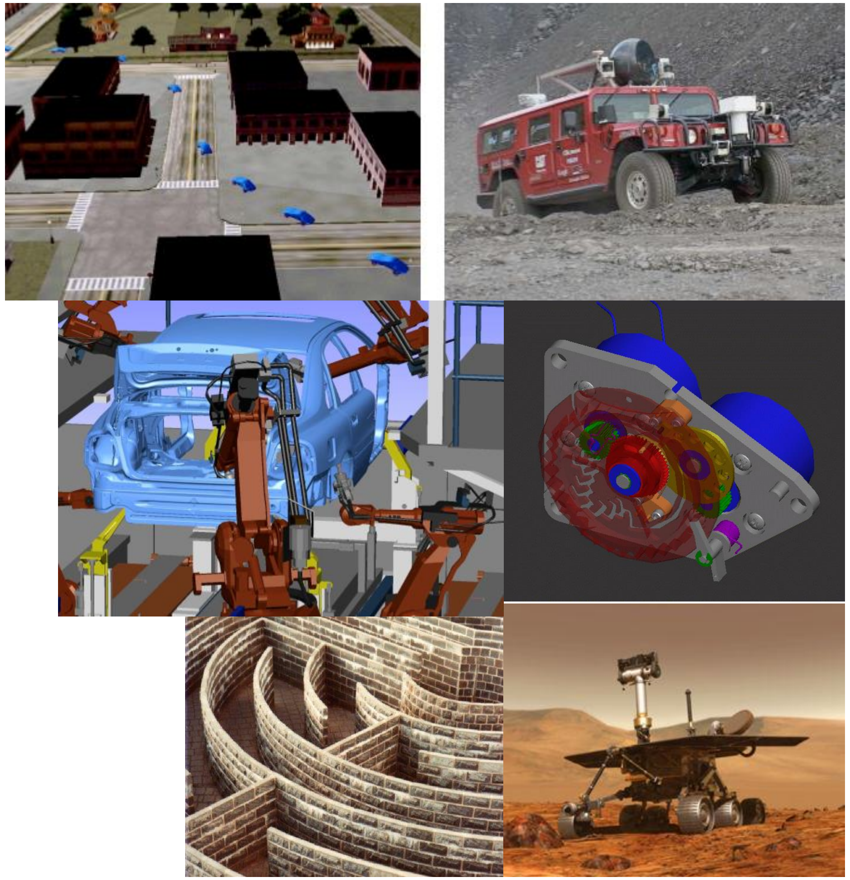
Reactive navigation

Motion Planning

- Automatic generation of motion
- So far, motion considered « out of any environment »
- Motion planning: generating a motion among obstacles

Applications

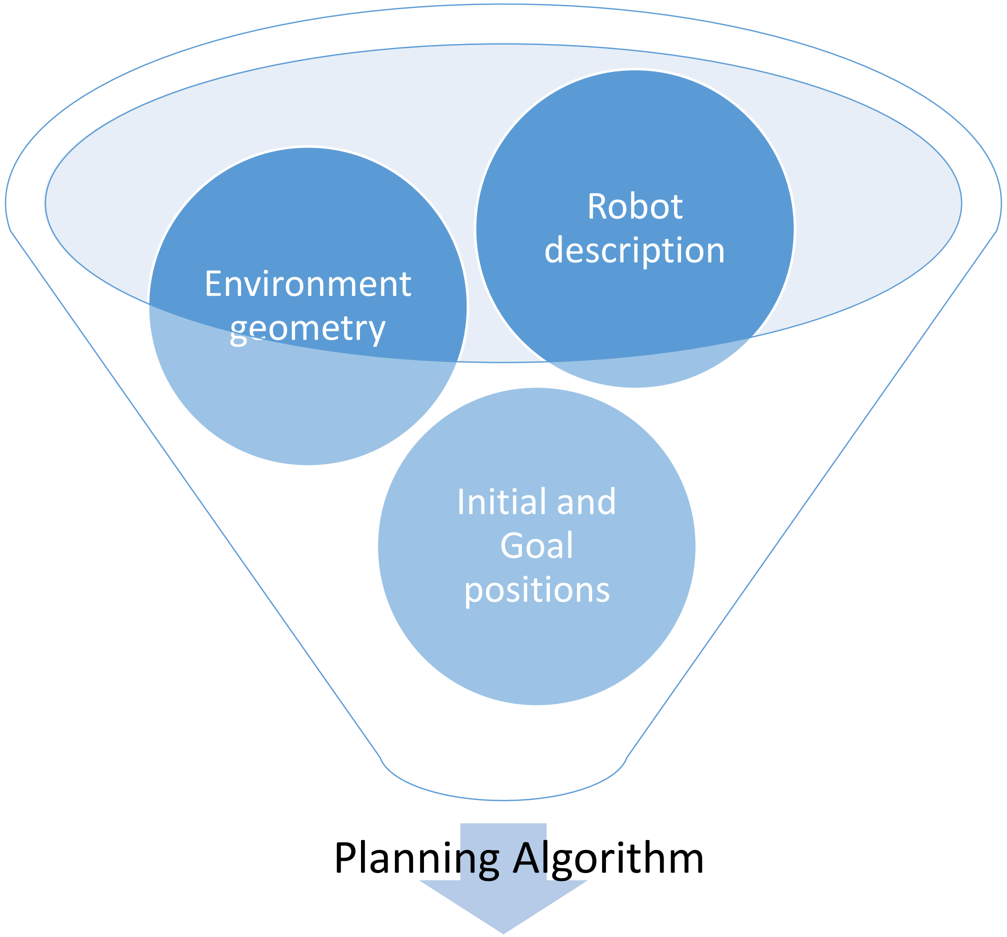
- Manufacturing: robot programming (welding, painting, assembly).
- Design for manufacturing and servicing
- Maintenance planning
- Autonomous vehicles: transportation, tractors, planetary exploration, military.
- Graphic animation: video games, movie generation.
- Medical surgery: implants, radiosurgery.
- Molecular biology: drug design.



The Piano Movers' problem

- Given an environment with obstacles and a piano, is it possible to move the piano from one position and orientation, called its configuration q, to another without colliding with the walls or the obstacles in a real geometric space or workspace W?
- SCHWARTZ J. T., SHARIR M. "On the piano movers' problem II, general techniques for computing topological properties of real algebraic manifolds." *Advances of Applied Maths 4 (1983), 298–351.*

Definition



A collision-free path from initial to goal positions

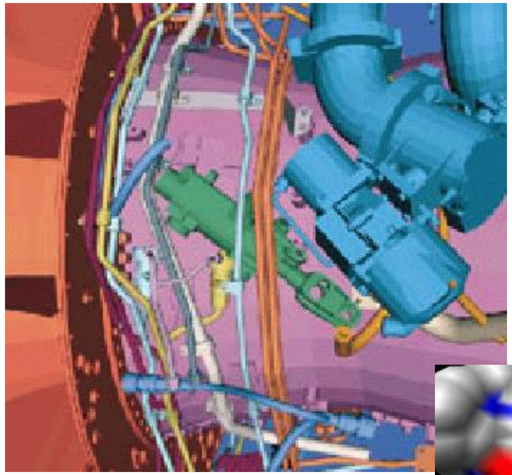
Problem extensions

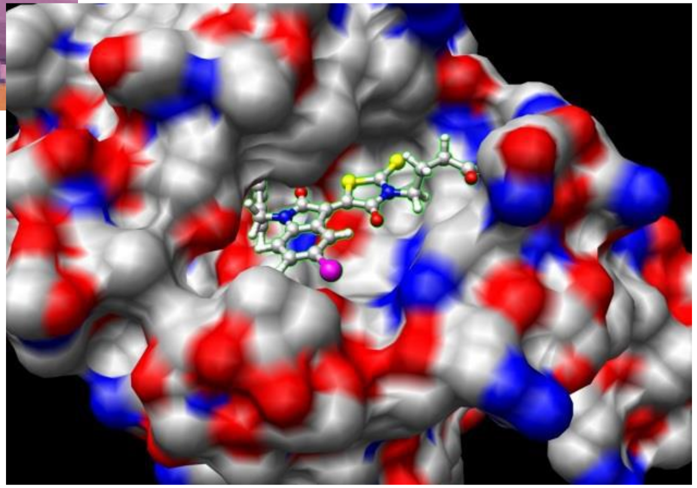
- Moving obstacles
- Multiple robots
- Movable objects
- Deformable robots
- No or partial prior knowledge of environment
- Uncertainty in sensing and control
- Non-holonomic constraints
- Dynamic constraints
- Optimal planning
- Visibility constraints

Why planning motions?

- Automatic motion generation

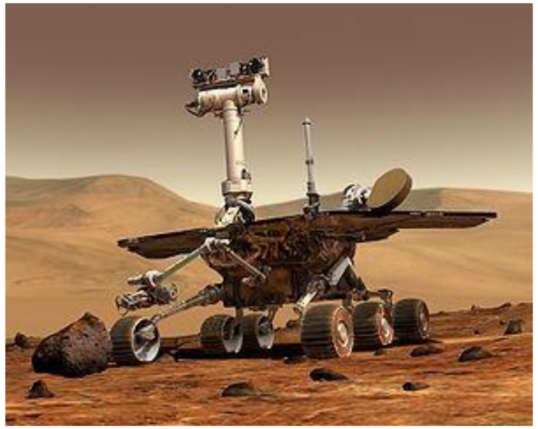
- Validation purpose





Applications: Mobile Robots

Roomba iCreate

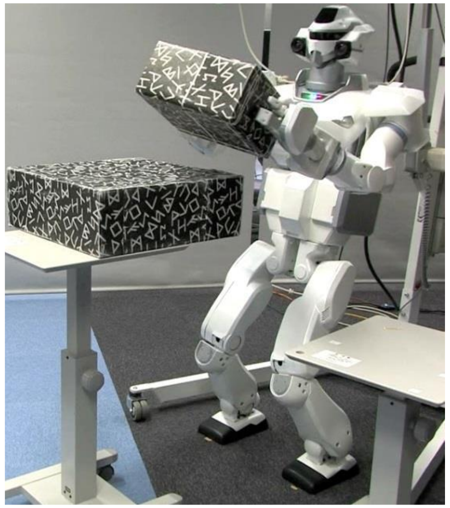


Mars Rover

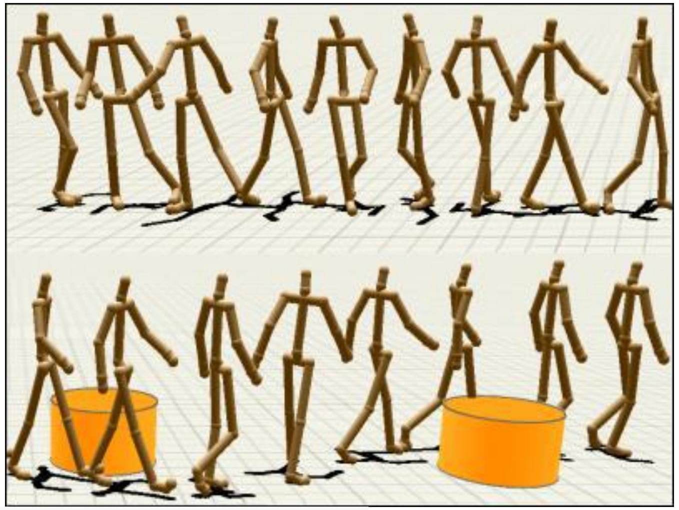
Google car

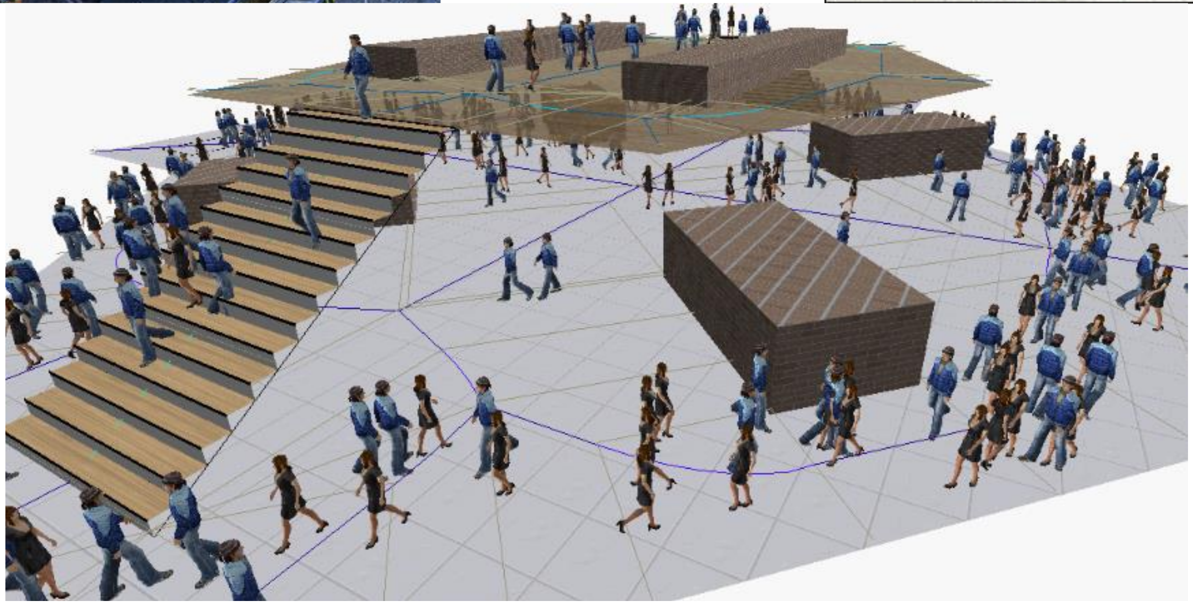
DARPA Urban Challenge

Applications: Robotic Manipulation

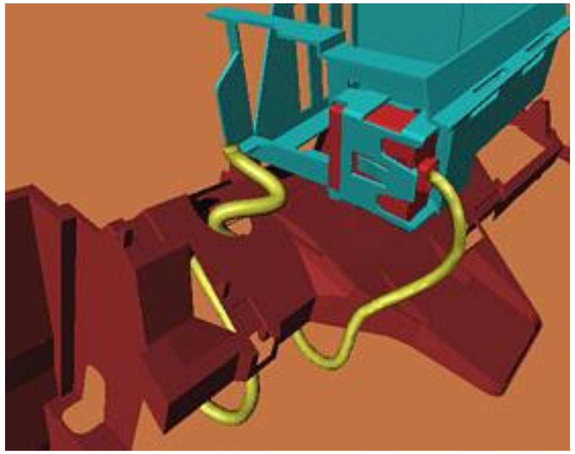


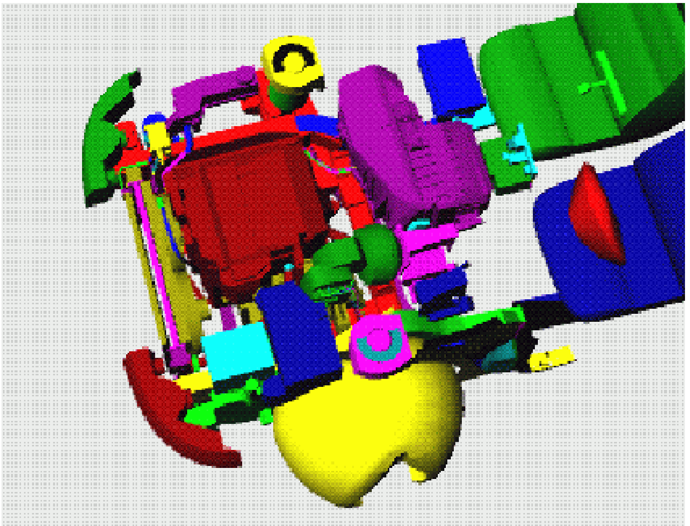
Applications: Computer Games/Graphics

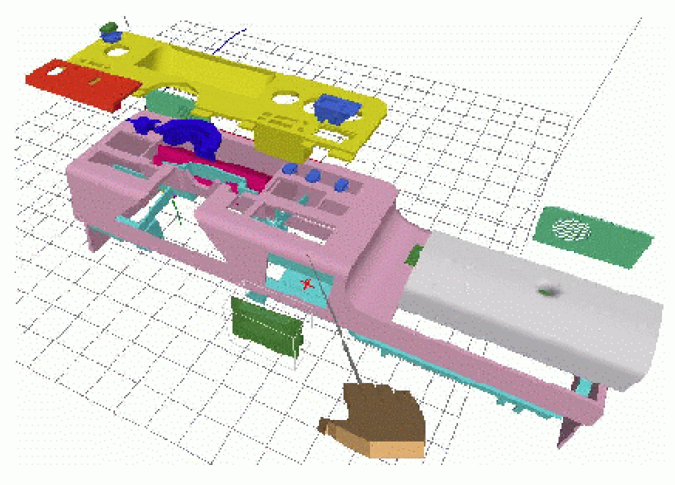




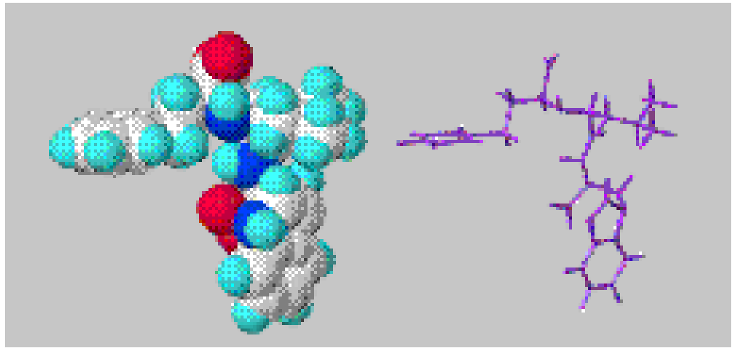
Applications: Assembly Planning

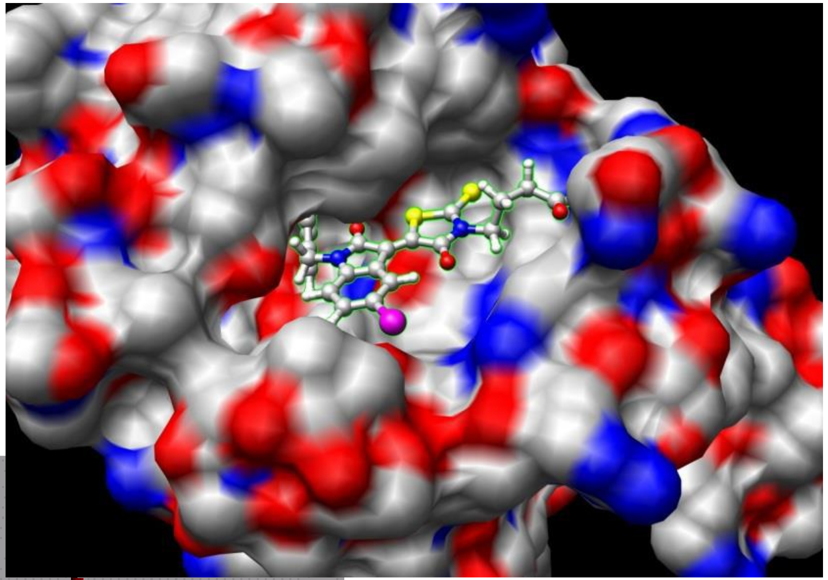


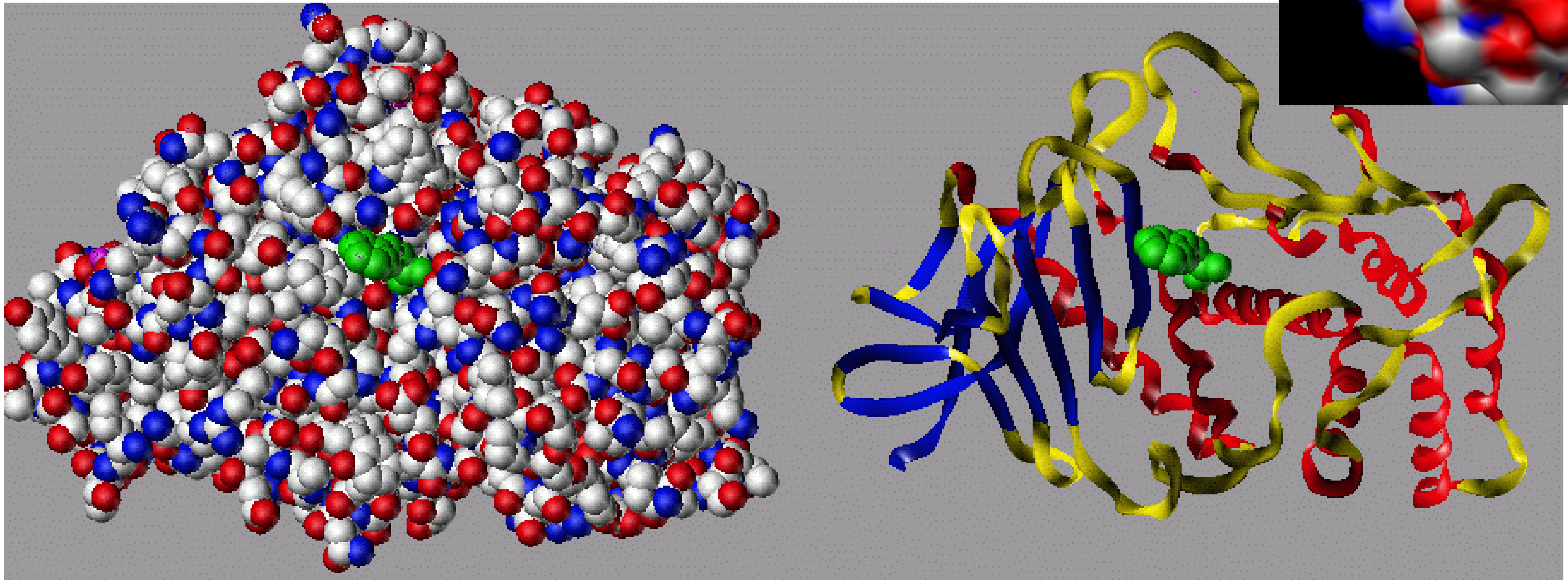




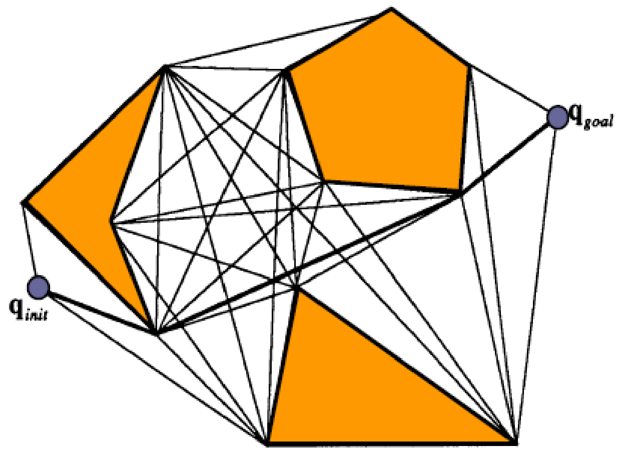
Applications: Computational Biology



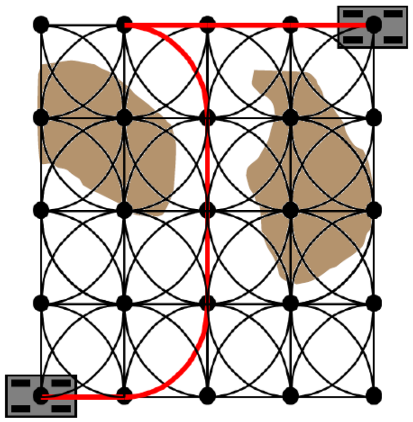


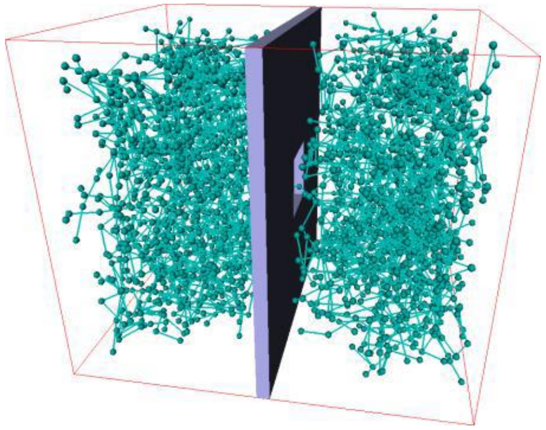


Approaches



- Exact algorithms
  - Either find a solution or prove none exists
  - Very computationally expensive
  - Unsuitable for high-dimensional spaces
- Discrete Search
  - Divide space into a grid, use A* to search
  - Good for vehicle planning
  - Unsuitable for high-dimensional spaces
- Sampling-based Planning
  - Sample the C-space, construct path from samples
  - Good for high-dimensional spaces
  - Weak completeness and optimality guarantees





Evaluation criteria

- Completeness
- Optimality
- Speed
- Generality

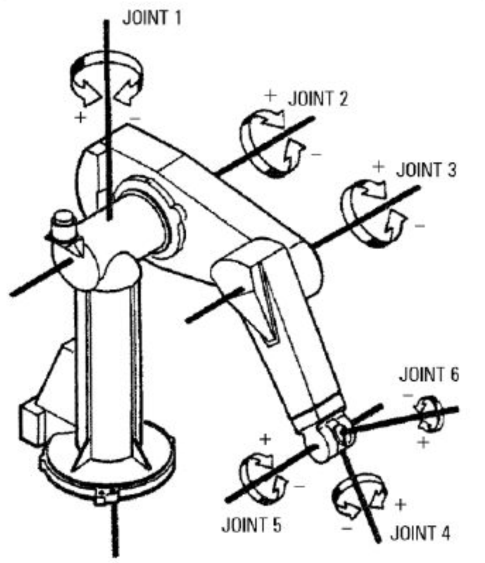
The Configuration Space

Configuration

- A **configuration** specifies the position of all points of a mechanical system in relation to a given coordinate system
- A configuration in expressed using a vector or generalised coordinates (in bold fonts): **q**
- **Example:** given a robot with n-links, a complete specification of the location of the robot is called its **configuration**

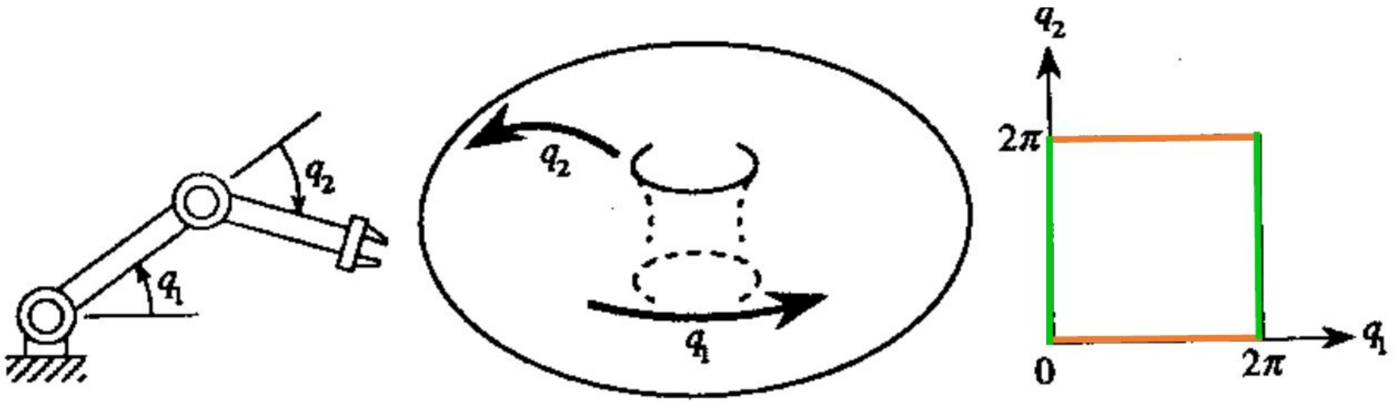
$$\begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \\ q_5 \\ q_6 \end{bmatrix}$$

Figure 2 The unimate PUMA 562 robot arm



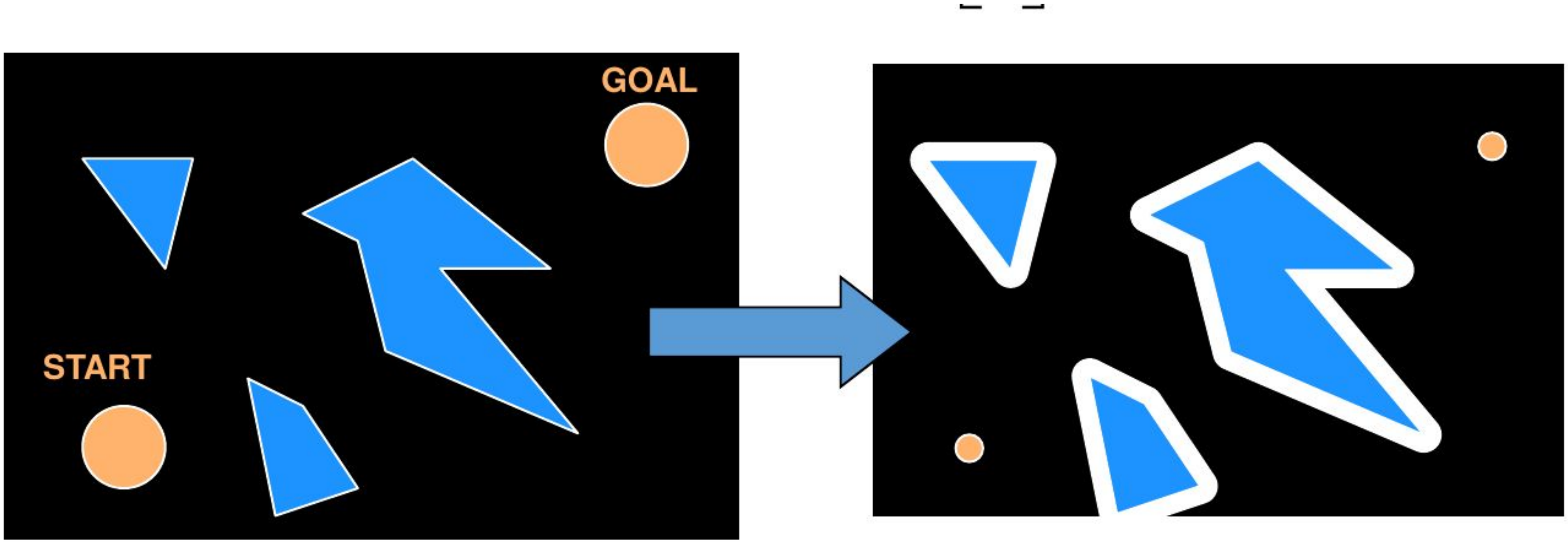
Configuration space

- It’s the space of all possible configurations for a given mechanical system
- Written C-space (configuration space)
- Formalized by [Lozano-Perez’97]



Collision-free space

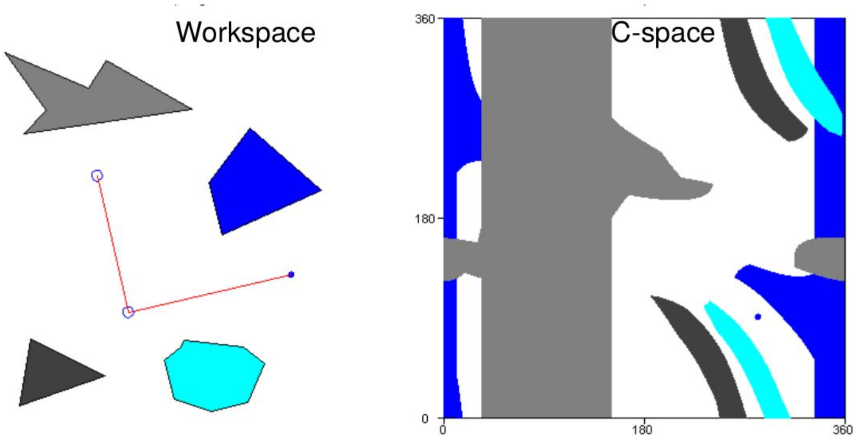
- The configuration space of a given mechanical system **without collisions with obstavles** is called the **C-free space**



Collision-free space

- C-free (or C-obst) can be difficult to compute in the C-space...

$$C_{space} = C_{free} + C_{obst}$$



C-free/C-obst representation for a 2-joint robot arm (the point in blue) in space [0,360]x[0,360]

Path and trajectories in C-space

Defintions:

- **Path:** a path is a **continuous sequence of configurations**
  - A path connects an initial configuration $\mathbf{q_s}$ to a final configuration $\mathbf{q_f}$
  - And does not collide with any obstacle along the path (ie all $\mathbf{q_i}$ belongs to C-free)

- **Trajectory:** a trajectory is a **path** with explicit parameterization of time
  - A trajectory connects an initial configuration $\mathbf{q_s}$ **at time 0** to a final configuration $\mathbf{q_f}$ **at time 1**
  - And does not collide with any obstacle along the trajectory (ie all $\mathbf{q_i}$ belongs to C-free)

And now we need to plan...

Path planning / trajectory planning / motion planning…?

State (rigid body mechanics) - Position and velocity at a given moment in time.

Motion - The change of state at any instant in time of a body (or bodies).

Trajectory - The state of a body or bodies over a period of time.

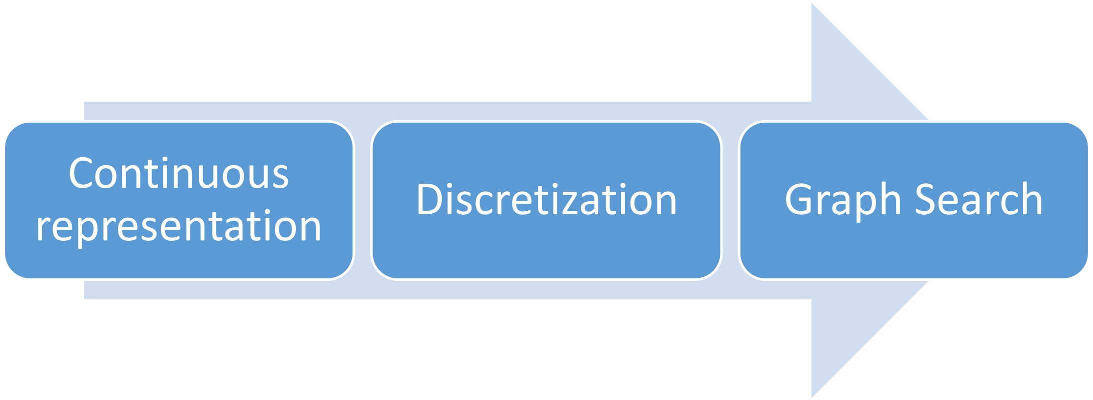
Path - The position of a body or bodies over a period of time without worrying about velocity or higher order terms.

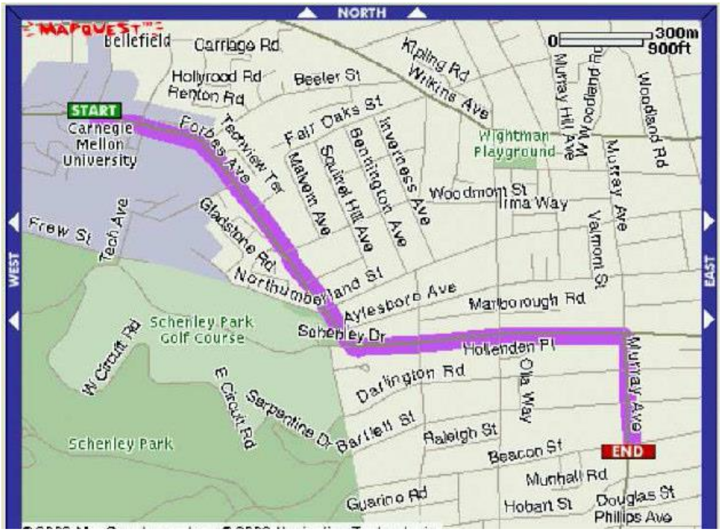
Planning - Calculating how to compose and sequence a set of primitives in a way that takes a body from an initial state to a final state while respecting a set of constraints (avoiding obstacles or burning minimal fuel for instance).

Therefore, you can do path planning (no time), trajectory planning (over time), motion planning (planning the sequence of changes in state, generally through actuators -- engines).

Framework

- Avoid searching the entire space
- Pre-compute an hopefully small graph (the roadmap) such that staying on the roads is guaranteed to avoid the obstacles
- Find a path between q_start and q_goal by using the roadmap

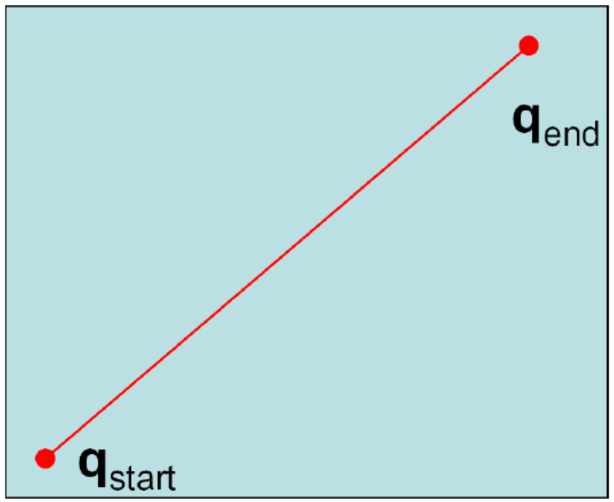




Basic approaches

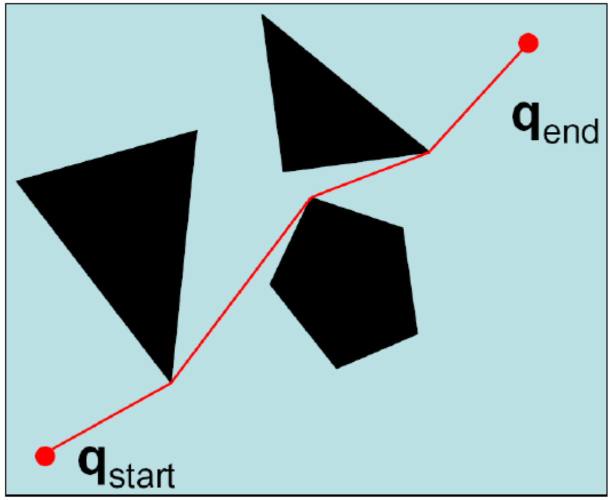
Visibility graphs

In the absence of obstacles, the best path is the straight line between q_start and q_goal



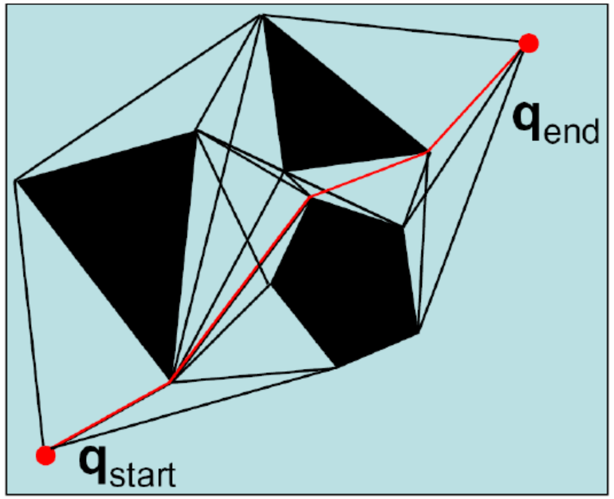
Visibility graphs

- Assuming polygonal obstacles: it looks like the shortest path is a sequence of straight lines joining the vertices of the obstacles
- Is it always true?



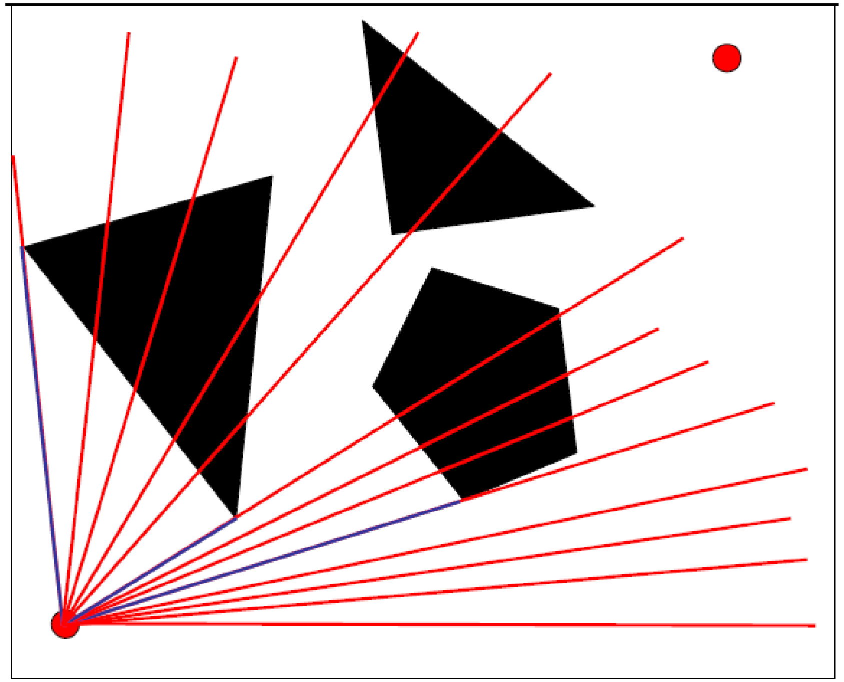
Visibility graphs

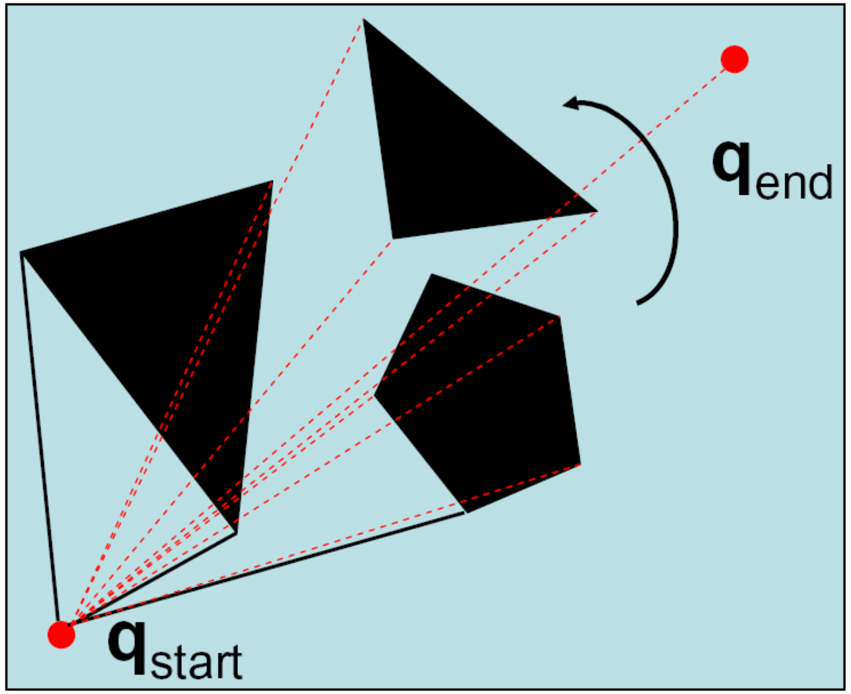
- Visibility graph G = set of unblocked lines between the vertices of the obstacles + q_start and q_goal
- A node P is linked to a node P' if P' visible from P
- Solution = shortest path in the visibility graph



Visbility Graphs

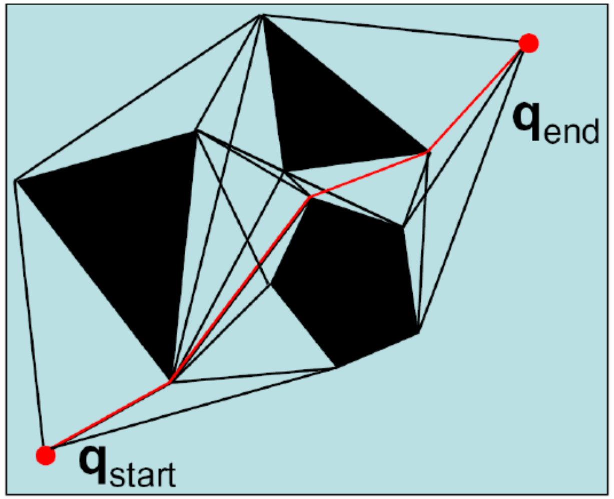
- Construction: sweep algorithm
- Sweep a line originating at each vertex
- Record those lines that end at visible vertices
- Complexity
  - N = total number of vertices of the obstacle polygons
  - Naïve: $O(N^3)$
  - Sweep: $O(N^2 \log N)$





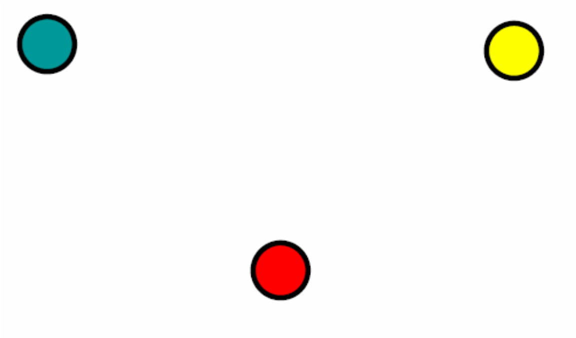
Visbility Graphs

- Shortest path but:
  - Tries to stay as close as possible to obstacles
  - Any execution error will lead to a collision
  - Complicated in >> 2 dimensions
- We may not care about strict optimality so long as we find a safe path. Staying away from obstacles is more important than finding the shortest path
- Need to define other types of "roadmaps"

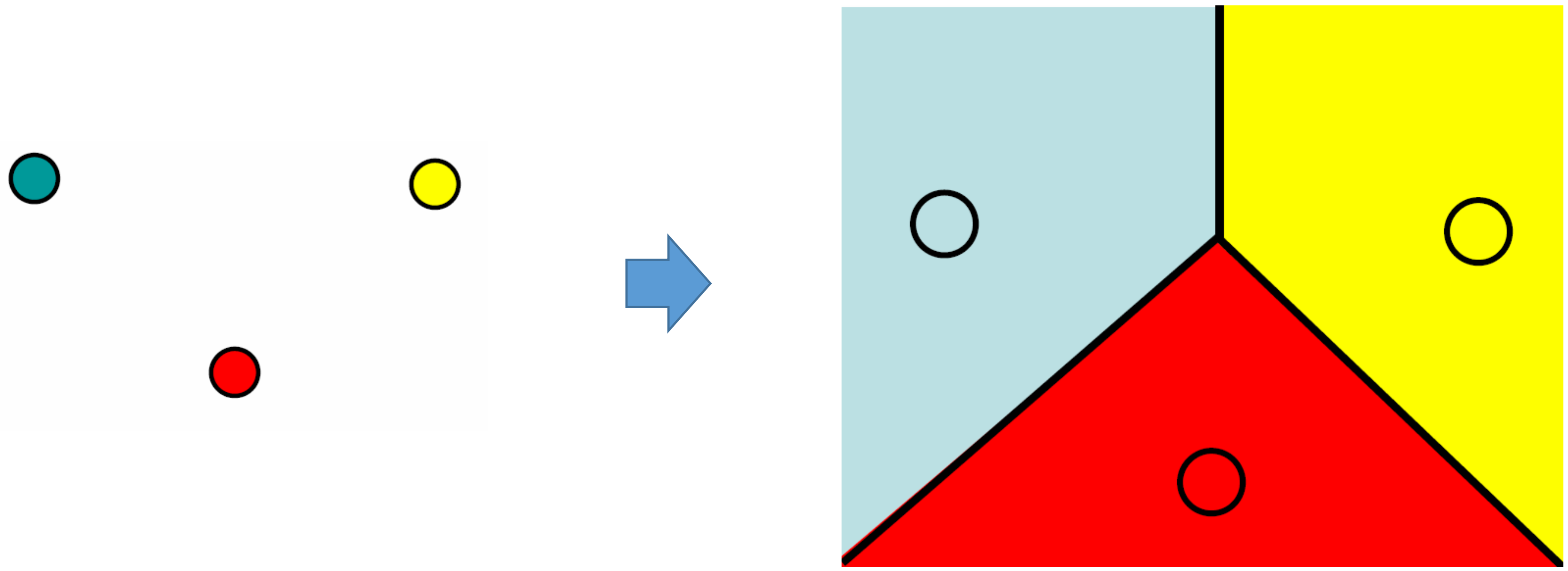


Voronoi Diagram

- Given a set of data points in the plane:
  - Color the entire plane such that the color of any point in the plane is the same as the color of its nearest

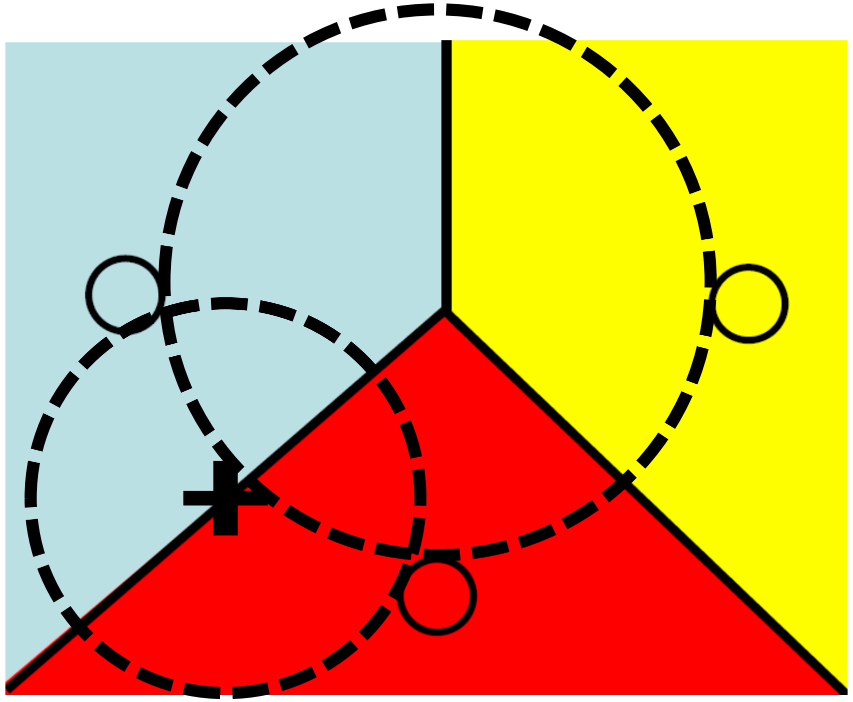


Voronoi Diagram



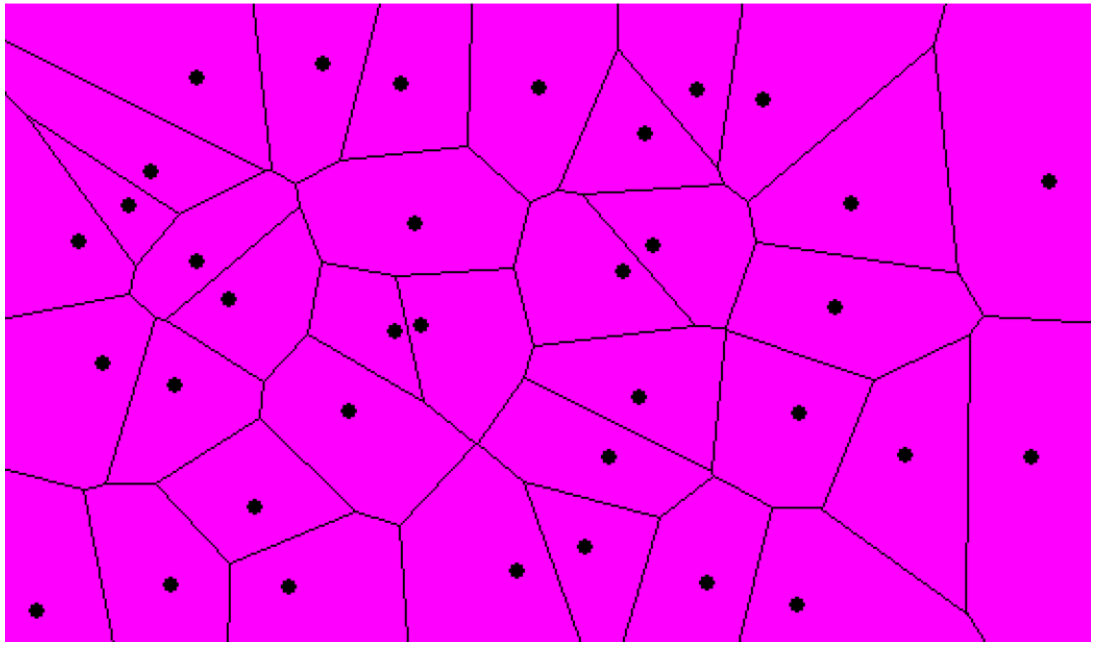
Voronoi Diagram

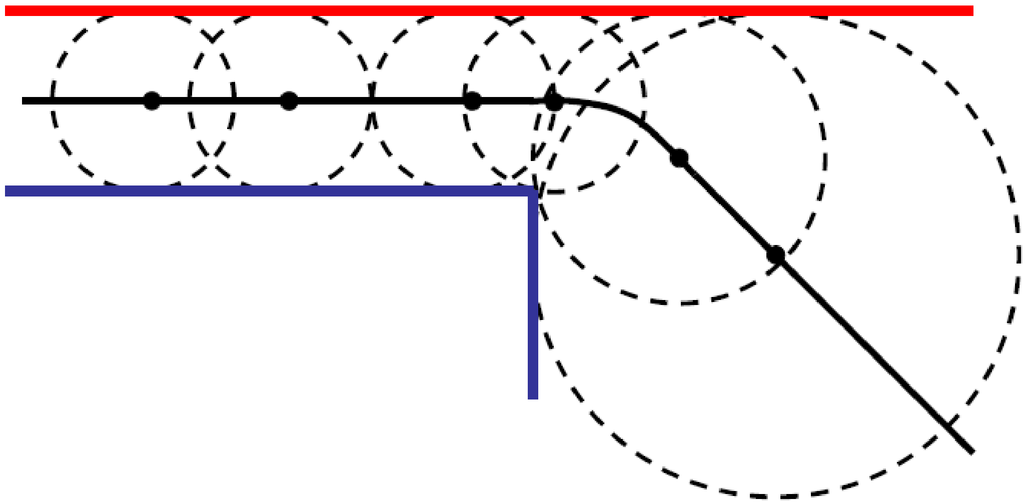
- Voronoi diagram
  =
  The set of line segments separating the regions corresponding to different colors
- Line segment = points equidistant from 2 data points
- Vertices = points equidistant from > 2 data points



Voronoi Diagram

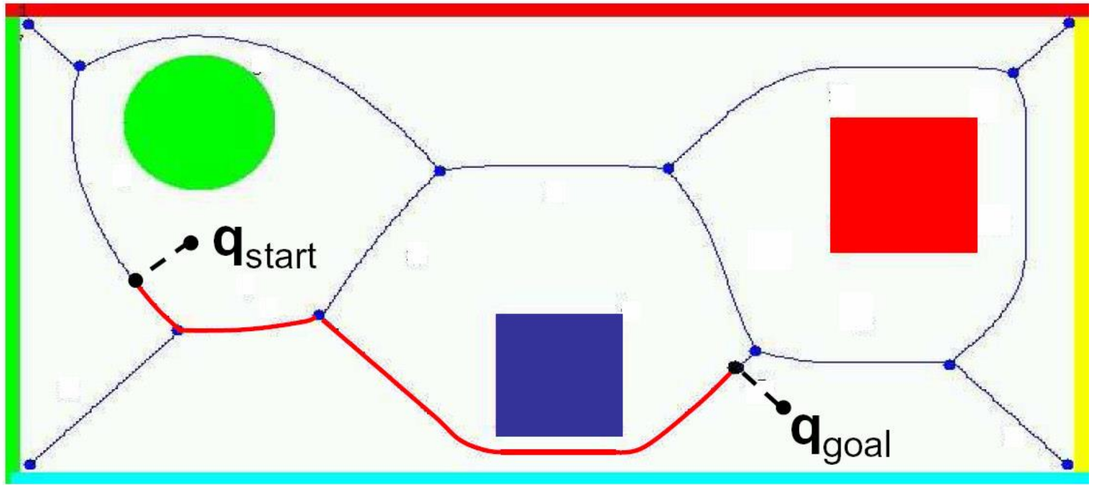
- Complexity (in the plane):
  - O(*N* log *N*) time
  - O(*N*) space
- Beyond points:
  - Edges are combinations of straight line segments and segments of quadratic curves
  - Straight edges: Points equidistant from 2 lines
  - Curved edges: Points equidistant from one corner and one line





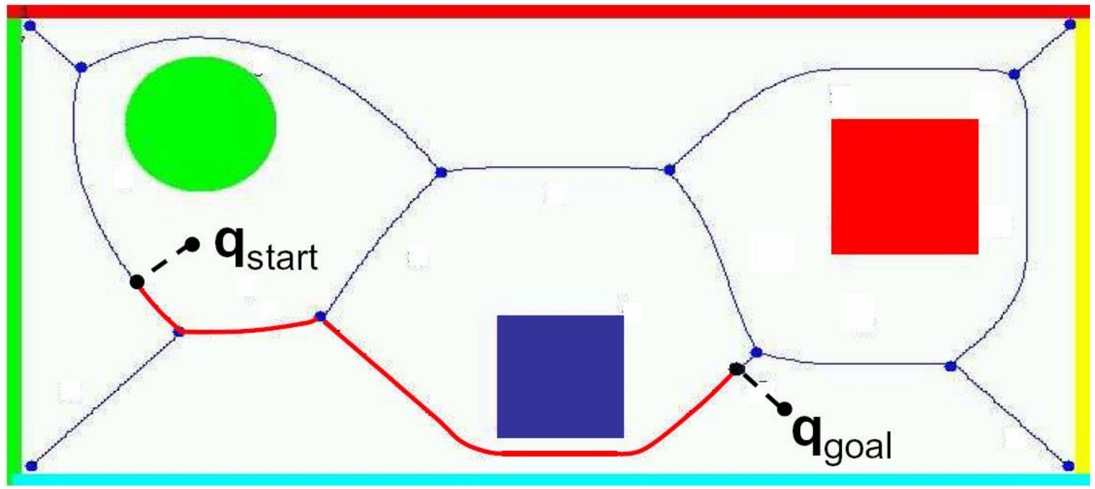
Voronoi Diagram

- Key property:
  The points on the edges of the Voronoi diagram are the *furthest* from the obstacles
- Idea:
  Construct a path between **q**_start and **q**_goal by following edges on the Voronoi diagram (Use the Voronoi diagram as a roadmap graph instead of the visibility graph)



Voronoi Diagram

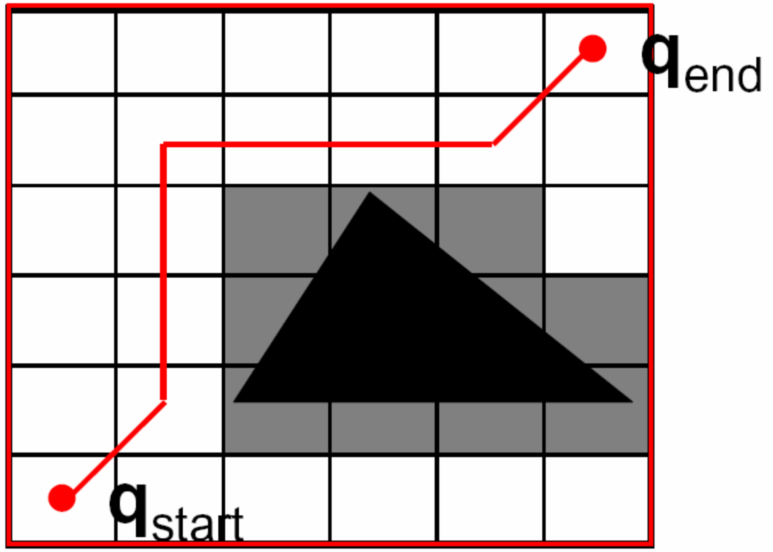
- Difficult to compute in higher dimensions or nonpolygonal worlds
  - Approximate algorithms exist
  - Use of Voronoi is not necessarily the best
- heuristic ("stay away from obstacles") Can lead to paths that are much too conservative
- Can be unstable Small changes in obstacle configuration can lead to large changes in the diagram



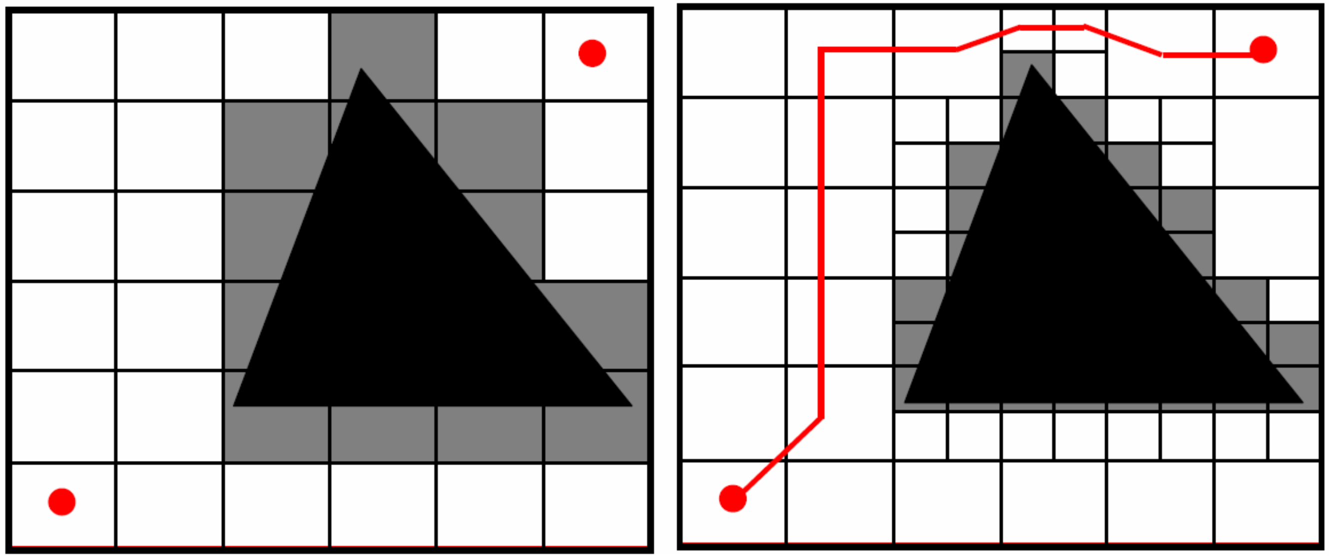
Cell decomposition

Approximate cell decomposition

- Define a discrete grid in C-Space
  - Mark any cell of the grid that intersects Cobs as blocked
- Find path through remaining cells by using (for example) A* (e.g., use Euclidean distance as heuristic)
- Cannot be *complete* as described so far. Why?

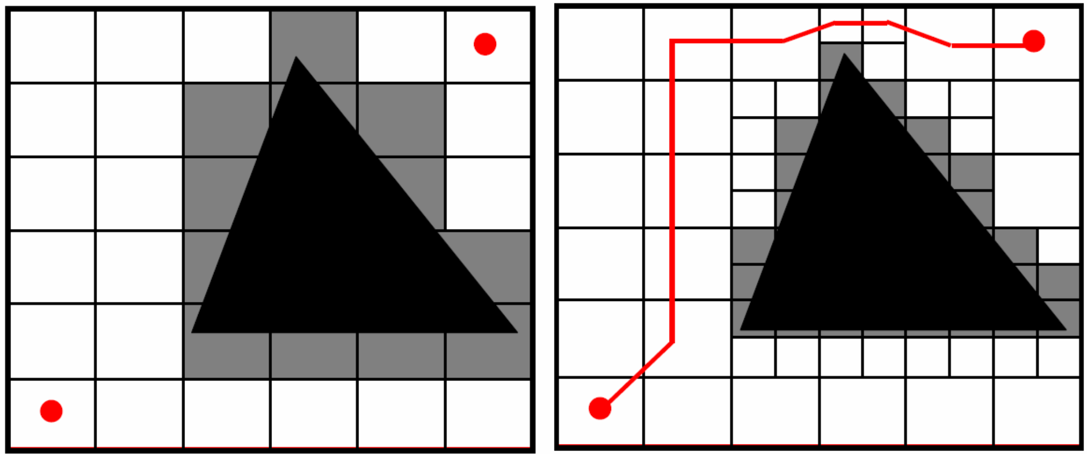


Approximate cell decomposition

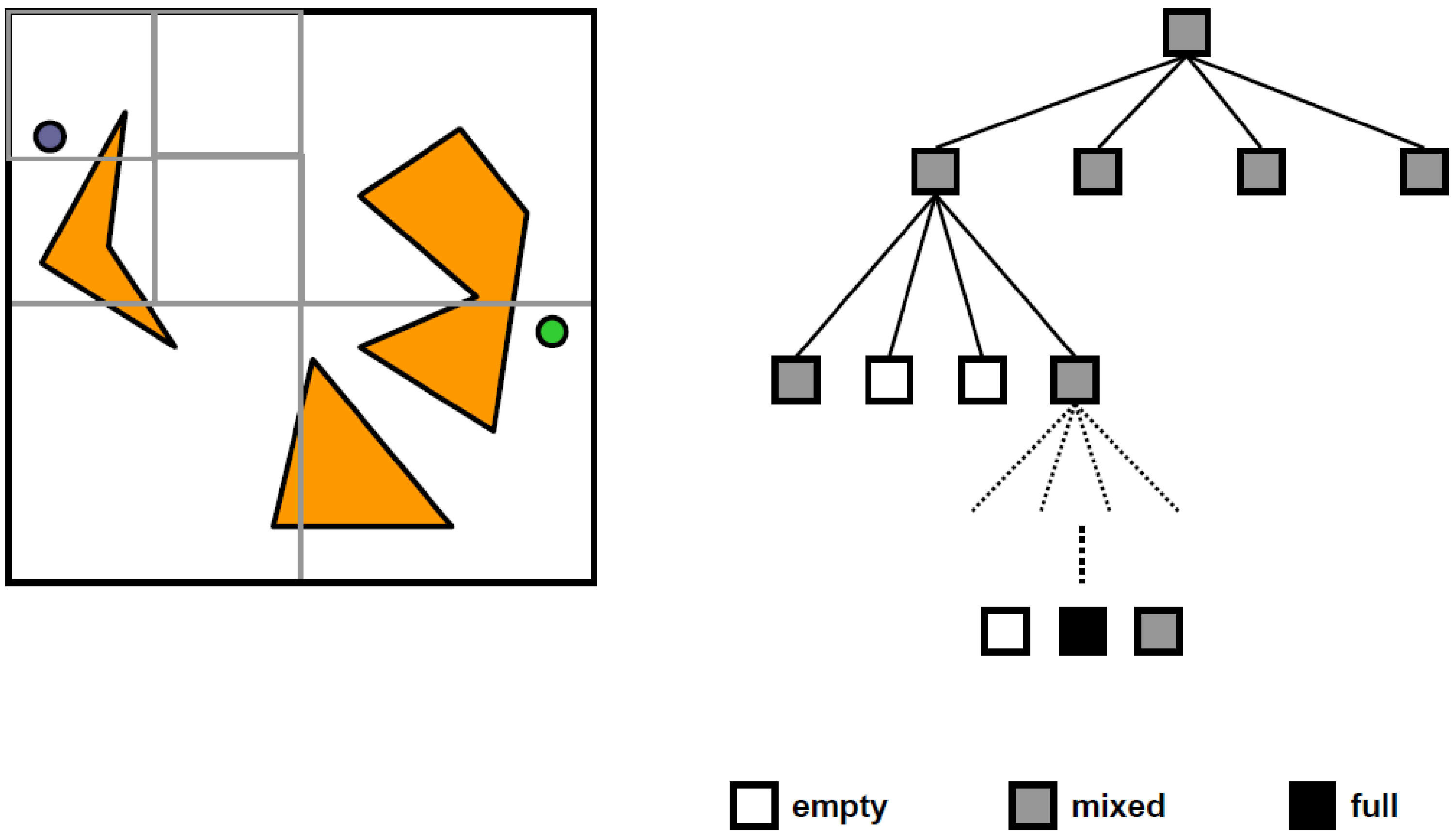


Approximate cell decomposition

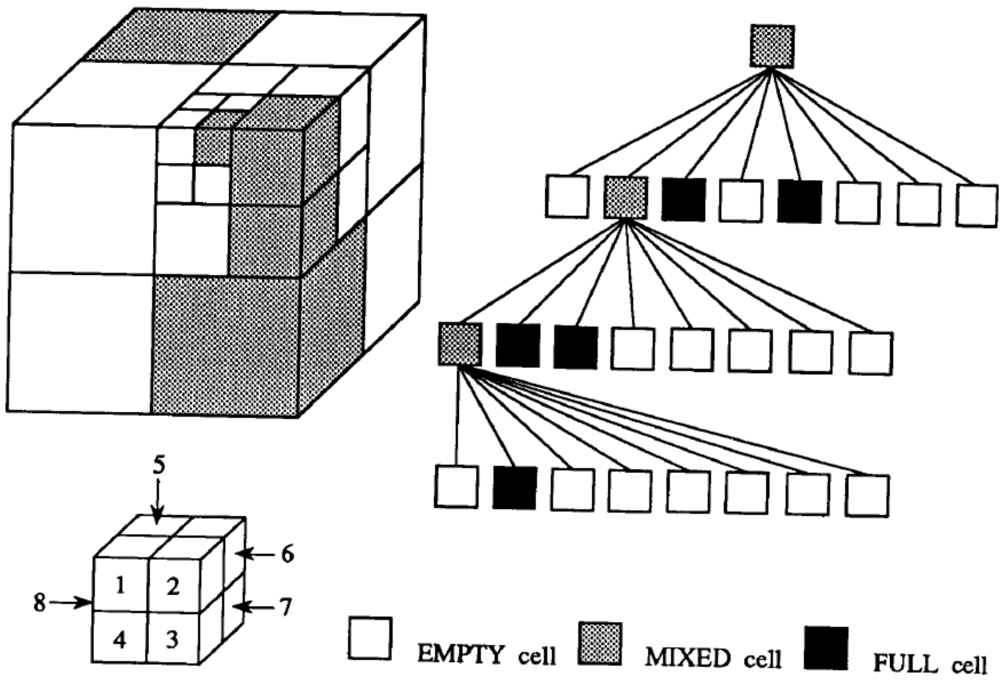
- Cannot find a path in this case even though one exists
- Solution: distinguish between
  - Cells that are entirely contained in Cobs (*FULL*) and
  - Cells that partially intersect Cobs (*MIXED*)
- Try to find a path using the current set of cells
- If no path found:
  - Subdivide the *MIXED* cells and try again with the new set of cells



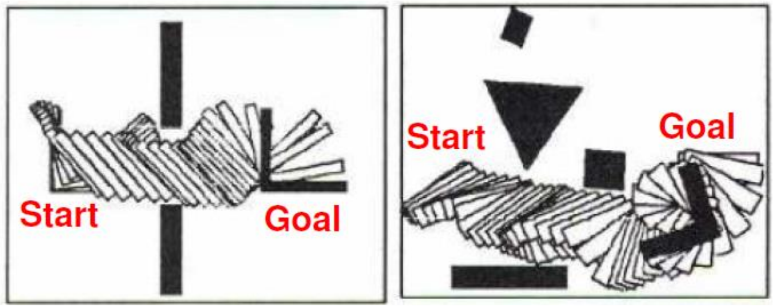
Quadtree decomposition

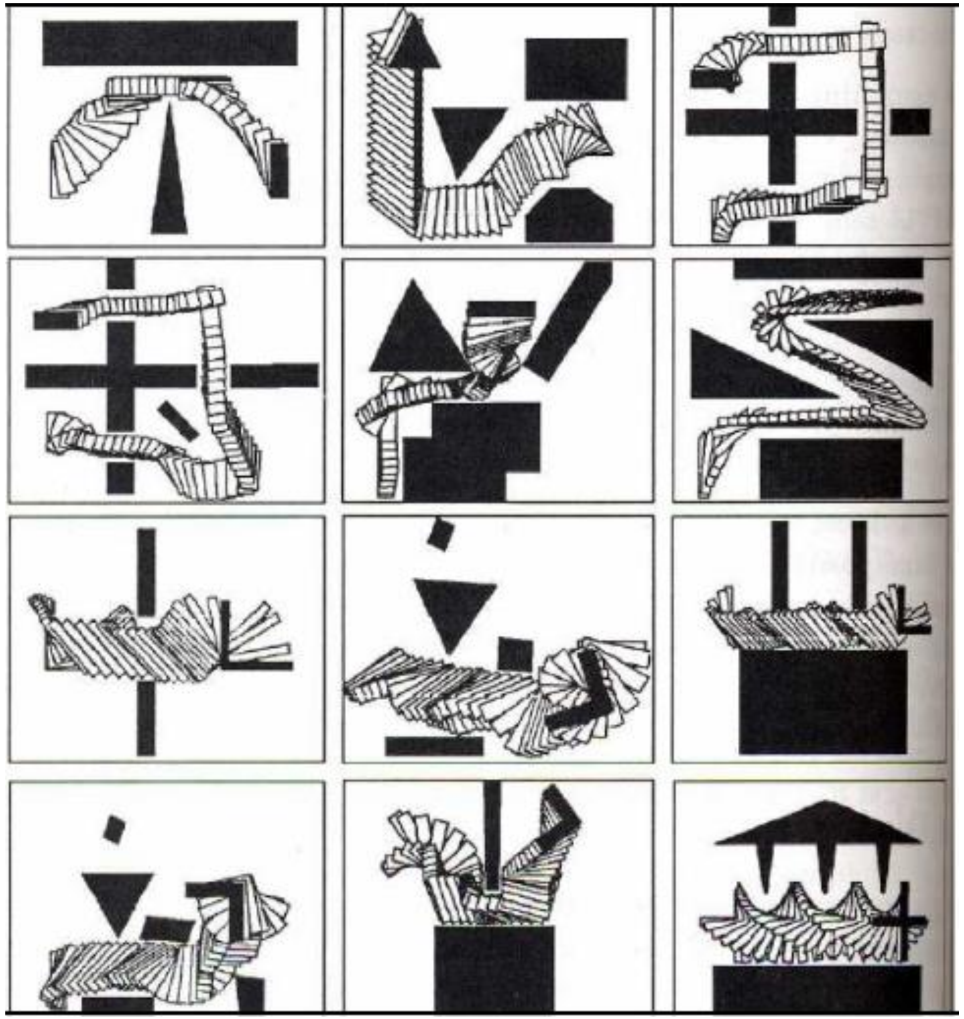


Octree decomposition



Approximate cell decomposition

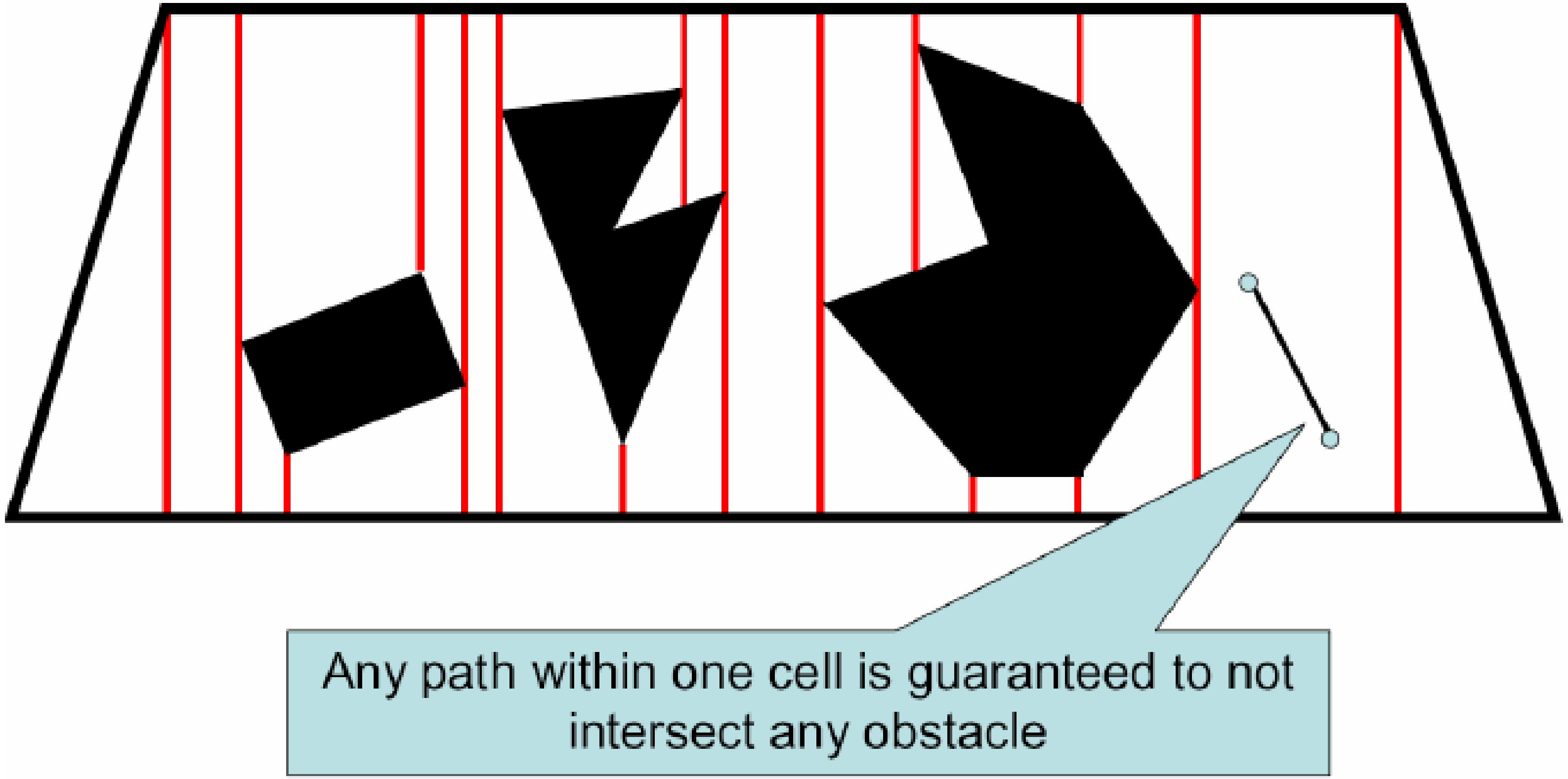




Approximate cell decomposition

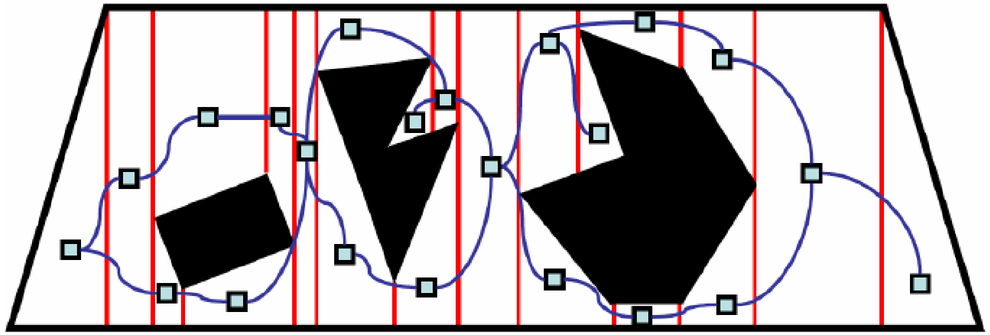
- Good:
  - Limited assumptions on obstacle configuration
  - Approach used in practice
  - Find obvious solutions quickly
- Bad:
  - No clear notion of optimality (“best” path)
  - Trade-off completeness/computation
  - Still difficult to use in high dimensions

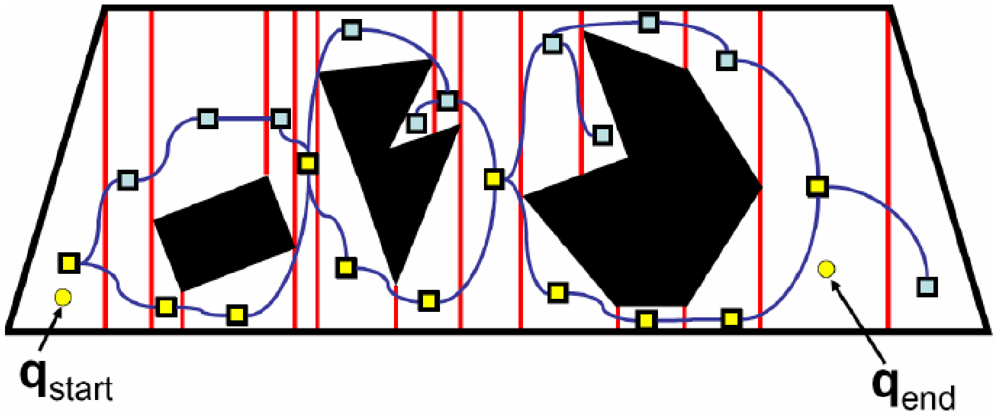
Exact cell decomposition



Exact cell decomposition

- The graph of cells defines a roadmap
- …
- And can be used to find a path between any two configuration

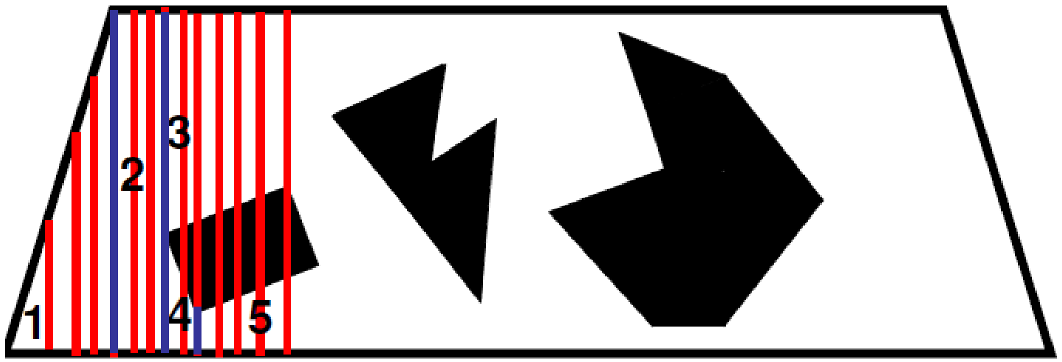




Exact cell decomposition

Plane Sweep algorithm

- Initialize current list of cells to empty
- Order the vertices of Cobs along the *x* direction
- For every vertex:
  - Construct the plane at the corresponding *x* location
  - Depending on the type of event:
- Split a current cell into 2 new cells OR
- Merge two of the current cells
  - Create a new cell
- Complexity (in 2-D):
  - Time: O(*N* log *N*)
  - Space: O(*N*)

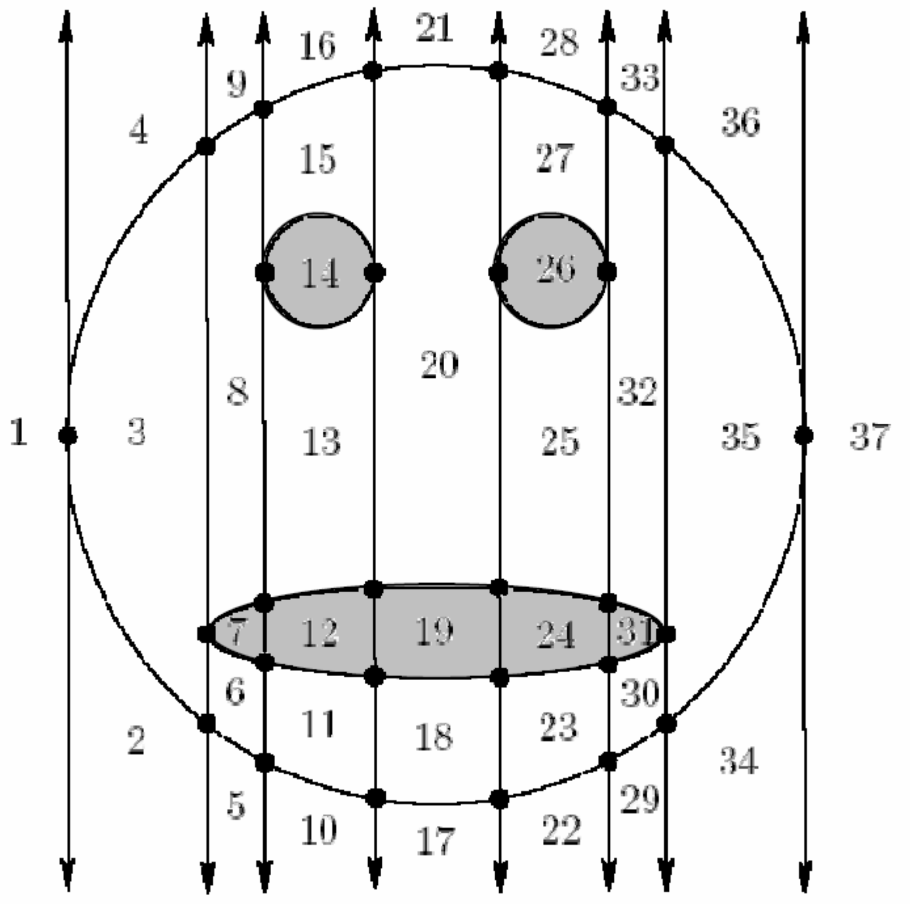


Critical event: Create new cell

Critical event: Split cell

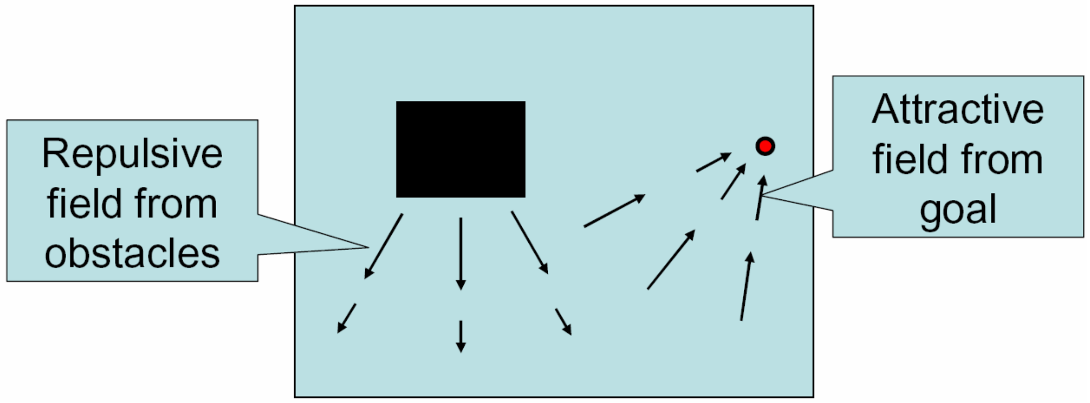
Exact cell decomposition

- A version of exact cell decomposition can be extended to higher dimensions and non-polygonal boundaries ("cylindrical cell decomposition")
- Provides exact solution -> completeness
- Expensive and difficult to implement in higher dimensions
- (double exp. Complexity)

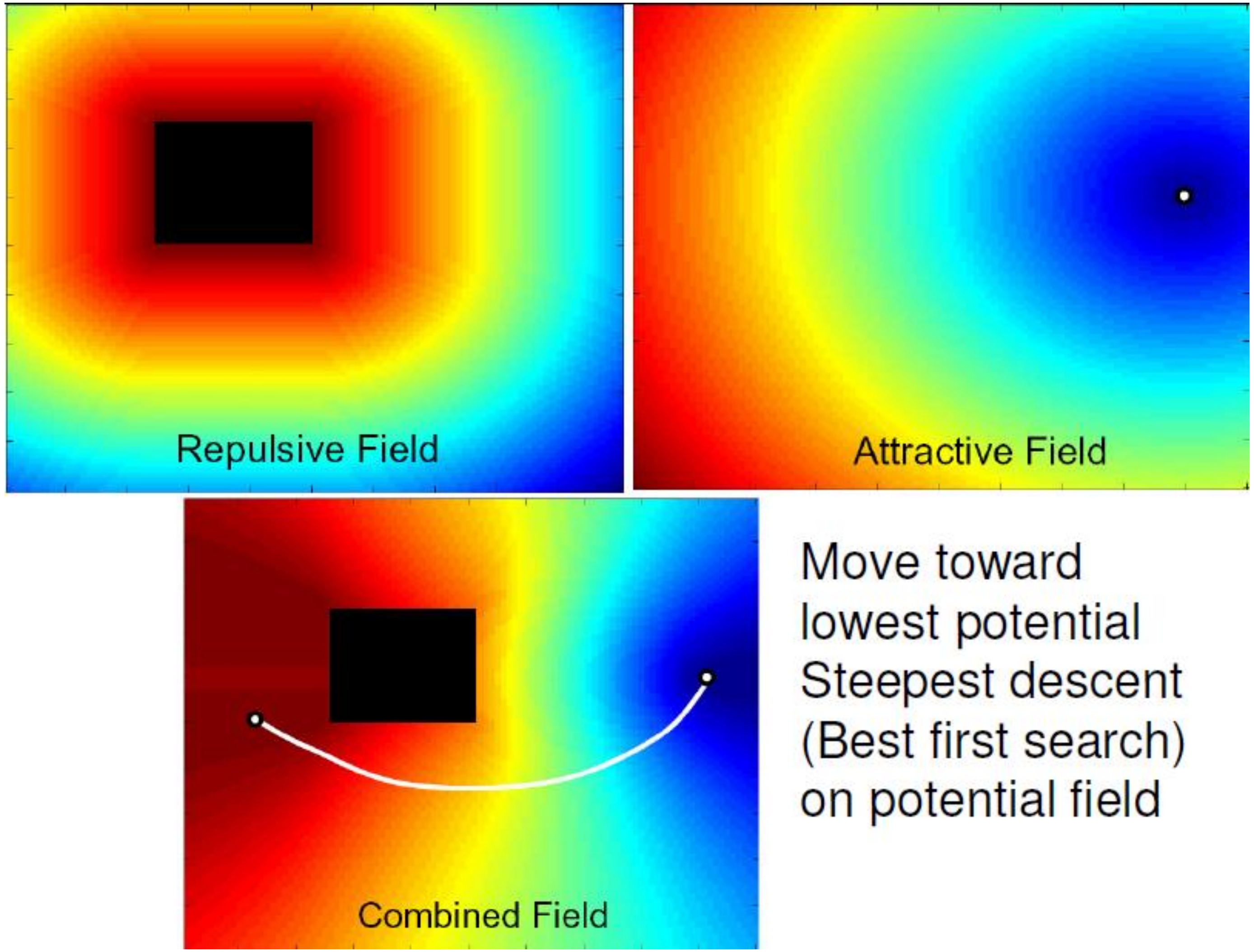


Potential fields

- Stay away from obstacles: Imagine that the obstacles are made of a material that generate a *repulsive* field
- Move closer to the goal: Imagine that the goal location is a particle that generates an *attractive* field



Potential fields



Potential fields

$$U_g(\mathbf{q}) = d^2(\mathbf{q}, \mathbf{q}_{goal})$$

Distance to goal state

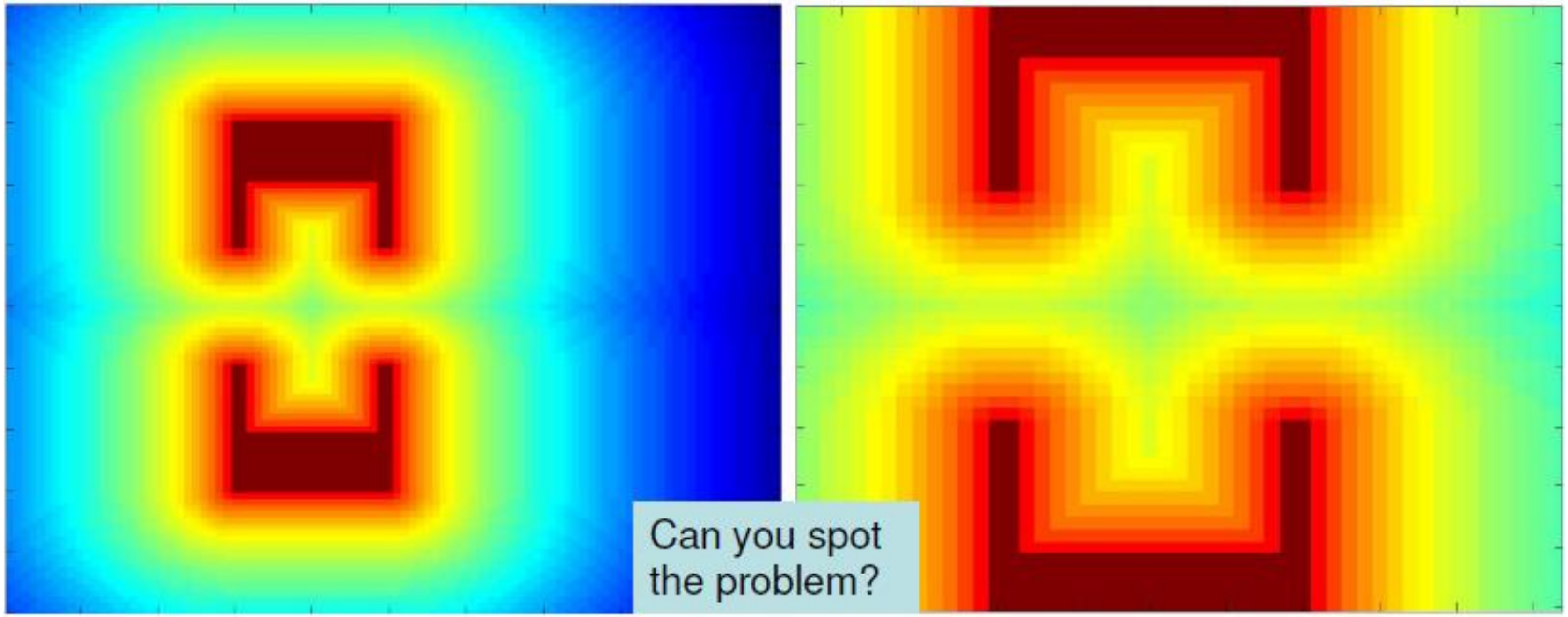
$$U_o(\mathbf{q}) = \frac{1}{d^2(\mathbf{q}, Obstacles)}$$

Distance to nearest obstacle point. Note: Can be computed efficiently by using the *distance transform*

$$U(\mathbf{q}) = U_g(\mathbf{q}) + \lambda U_o(\mathbf{q})$$

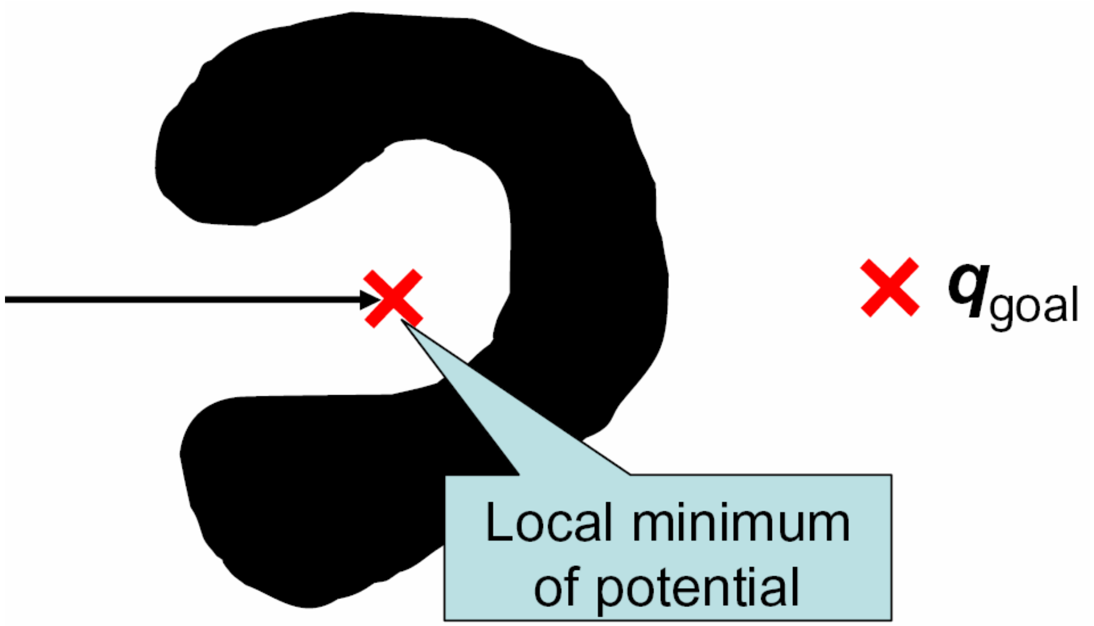
λ controls how far we stay from the obstacles

Potential fields



Potential field　　Zoomed in view

- Potential fields in general exhibit local minima
- Special case: Navigation function
  - $U(\boldsymbol{q}\text{goal}) = 0$
  - For any $\boldsymbol{q}$ different from $\boldsymbol{q}$goal, there exists a neighbor $\boldsymbol{q}'$ such that $U(\boldsymbol{q}') < U(\boldsymbol{q})$



Getting out of Local Minima

Repeat

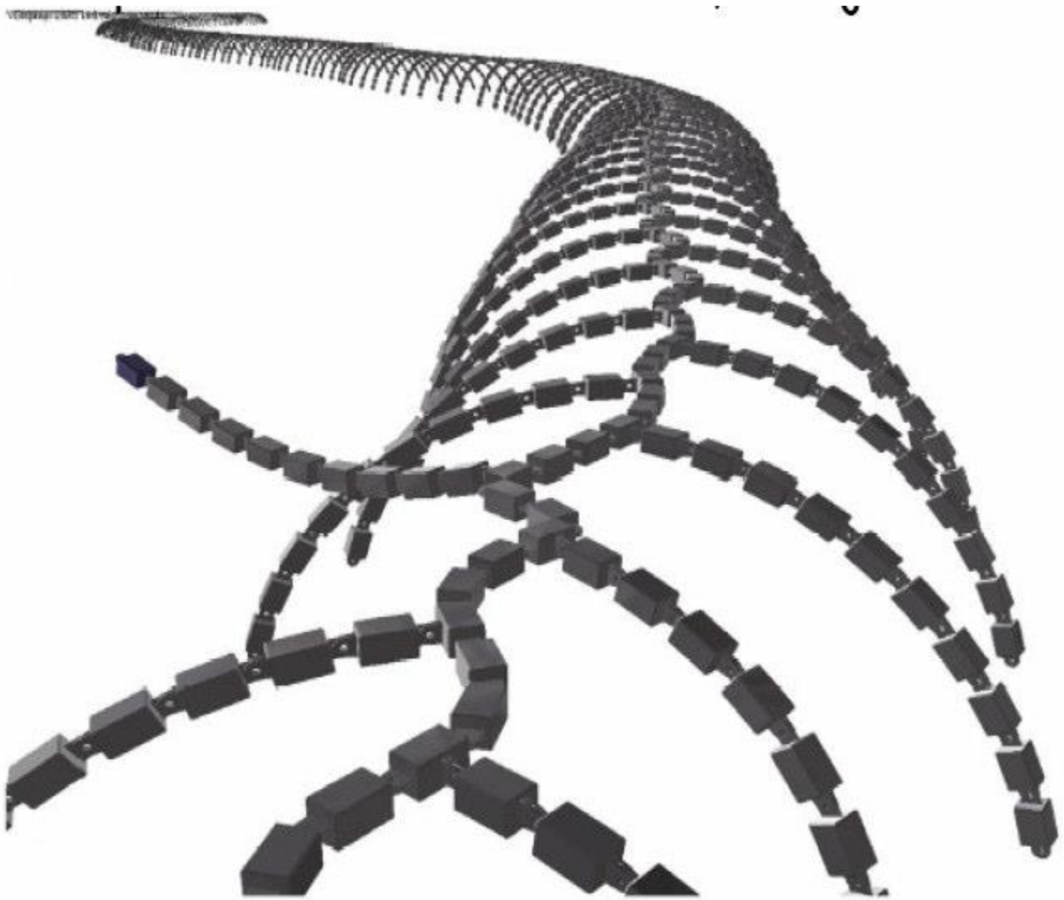
- If *U*(***q***) = 0 return Success
- If too many iterations return Failure
- Else:
  - Find neighbor ***q***n of ***q*** with smallest *U*(***q***n)
  - If *U*(***q***n) < *U*(***q***) OR ***q***n has not yet been visited
- Move to ***q***n (***q*** <- ***q***n)
- Remember ***q***n

Repeat

- If *U*(***q***) = 0 return Success
- If too many iterations return Failure
- Else:
  - Find neighbor ***q***n of ***q*** with smallest *U*(***q***n)
  - If *U*(***q***n) < *U*(***q***)
    - Move to ***q***n (***q*** ← ***q***n)
  - Else
    - Take a random walk for *T* steps starting at ***q***n
    - Set ***q*** to the configuration reached at the end of the random walk

Large C-space dimension

- Millipede like robot (S. Redon)
- 13.000 dofs!

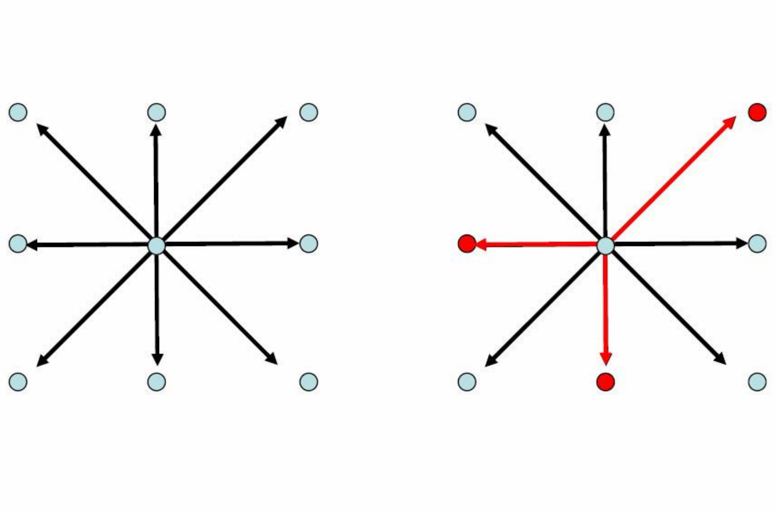
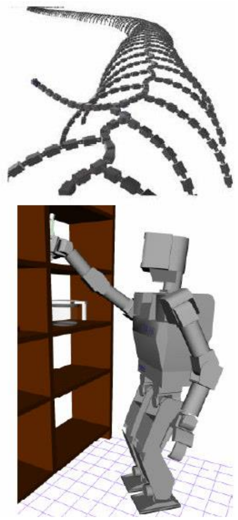


Dealing with C-Space Dimension

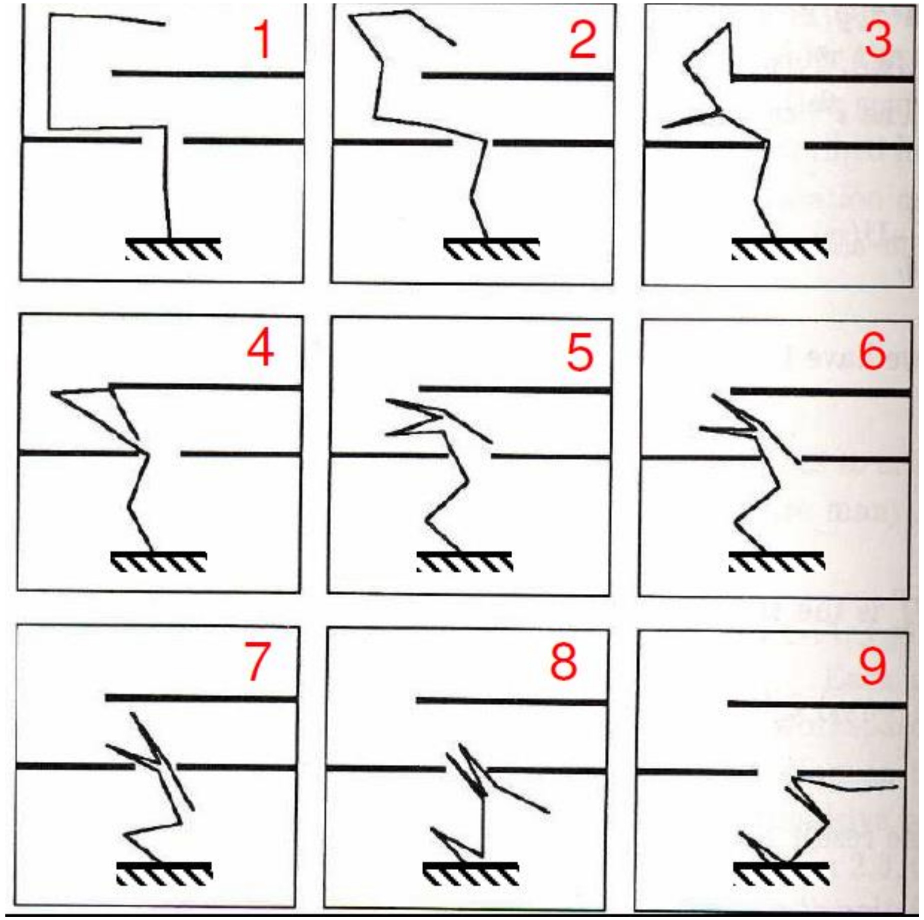
- We should evaluate all the neighbors of the current state, but:
- Size of neighborhood grows exponentially with dimension
- Very expensive in high dimension

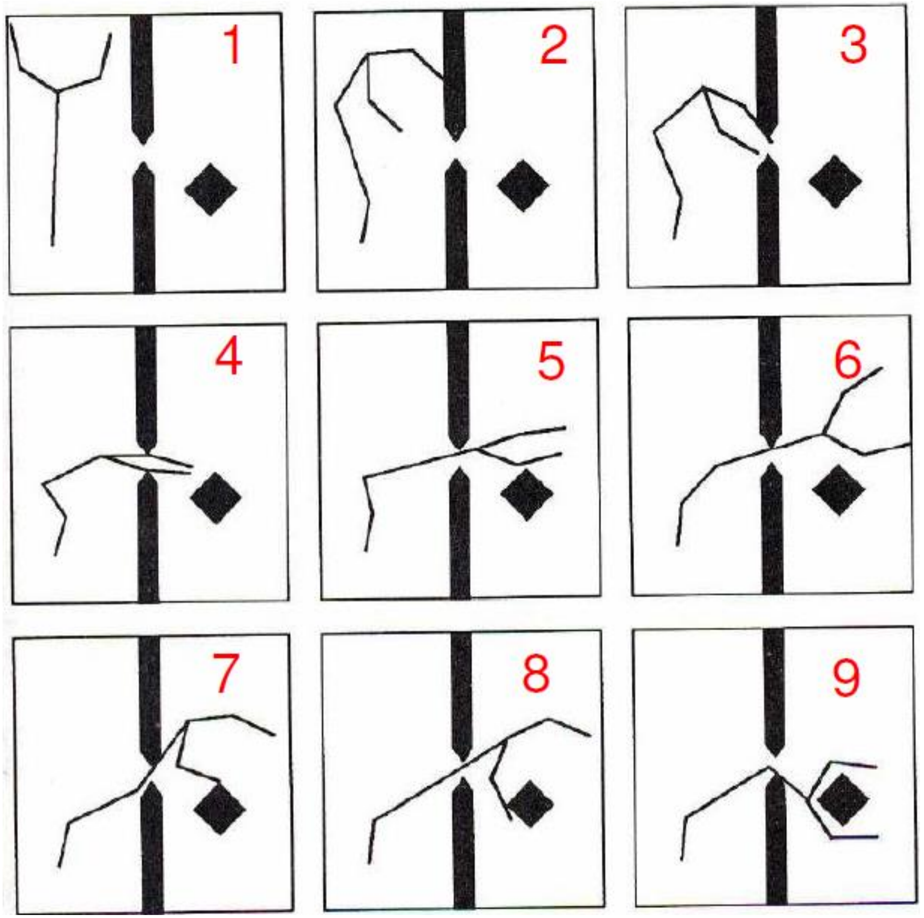
Solution:

- Evaluate only a random subset of *K* of the neighbors
- Move to the lowest potential neighbor

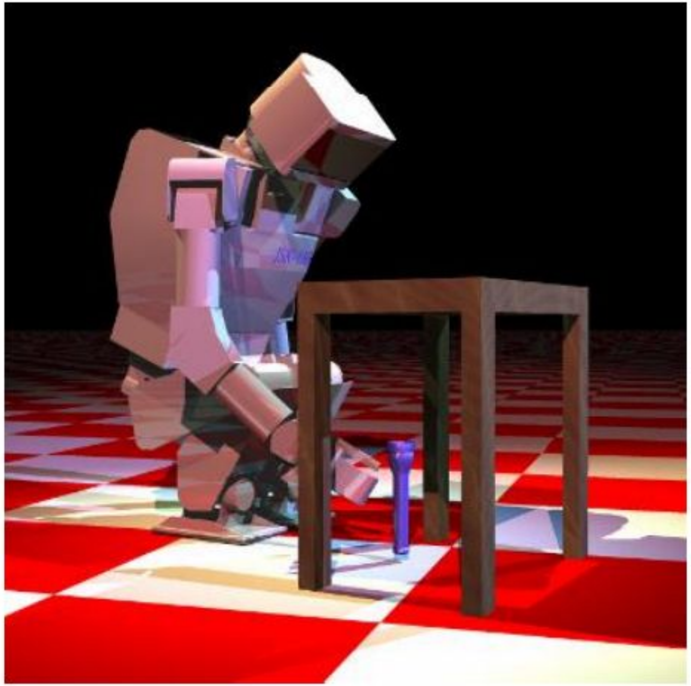
Dealing with C-Space Dimension





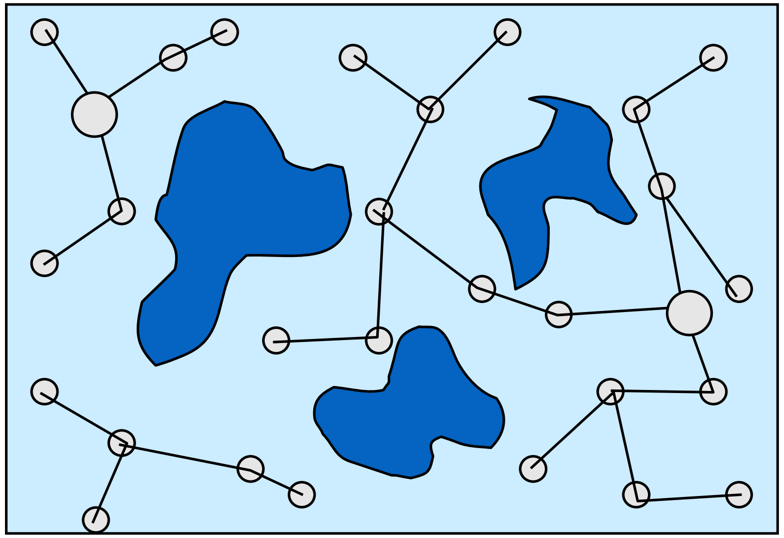
Planners for high-dimensional spaces

- Ideally one would want a COMPLETE planner (if there is a solution, it will be found)
- **Problem:** the complete planners are P-SPACE!
- **Solutions: reduce the search space**
  - By adding some constraints
  - By expressing the problem in an alternate space (easier to solve)
  - By not visiting all configurations (ie a subset only)
  - By removing the optimality hypothesis (not the best solution)
  - By removing the completeness hypothesis (no guarantee to succeed if there is a path)



Probabilistic Roadmaps Method (PRM)

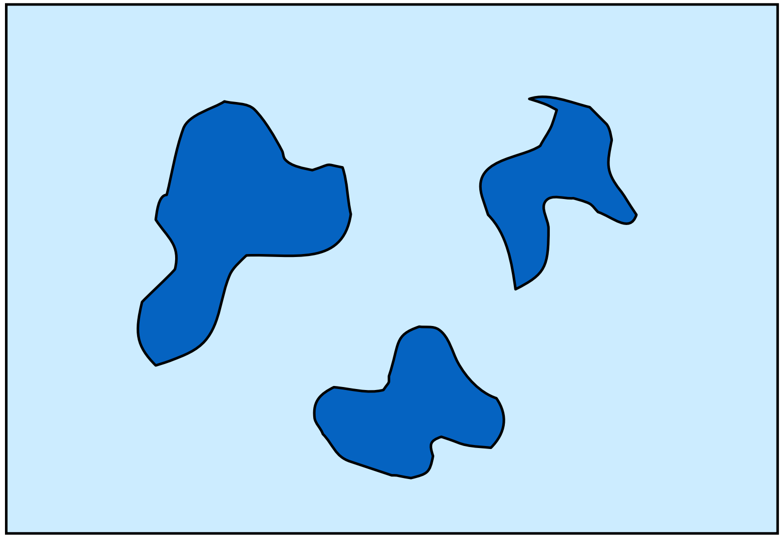
- Relies on 3 elements:
  - Collision checker
  - Local Method
  - Sampler
- 2 major steps:
  - Exploration Phase
  - Query Phase
- Key Idea: explore randomly C-space and capture C-free topology into a roadmap
- Complete in infinite time: probabilistically complete



Preprocessing: learning phase

- Iterative algorithm

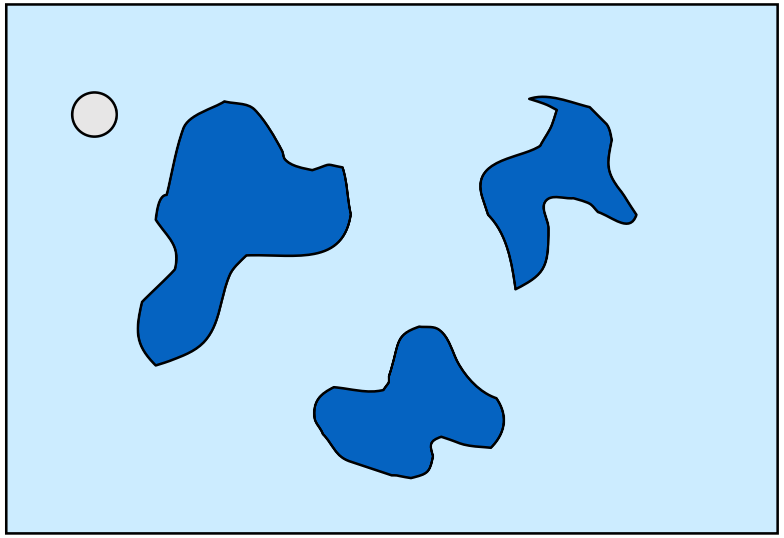
1. Compute random configuration
   - Collision checker
2. Connect configuration
   - Collision checker
   - Local method
3. Goto 1



Preprocessing: learning phase

- Iterative algorithm

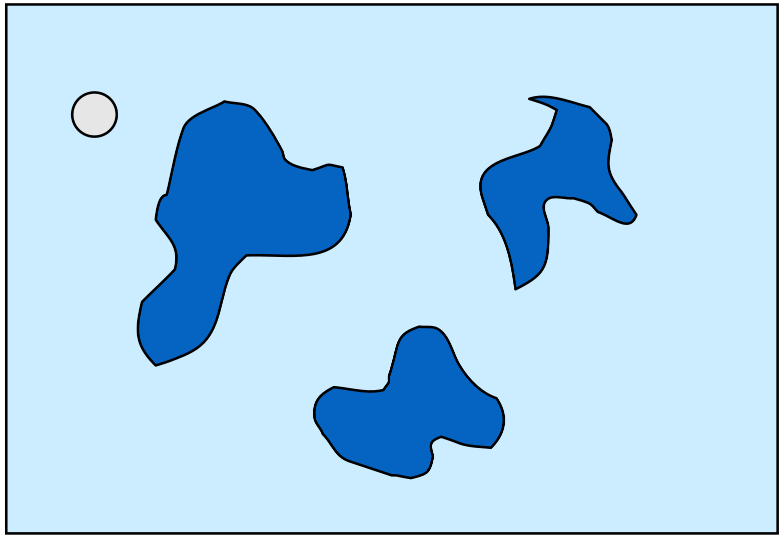
1. Compute random configuration
   - Collision checker
2. Connect configuration
   - Collision checker
   - Local method
3. Goto 1



Preprocessing: learning phase

- Iterative algorithm

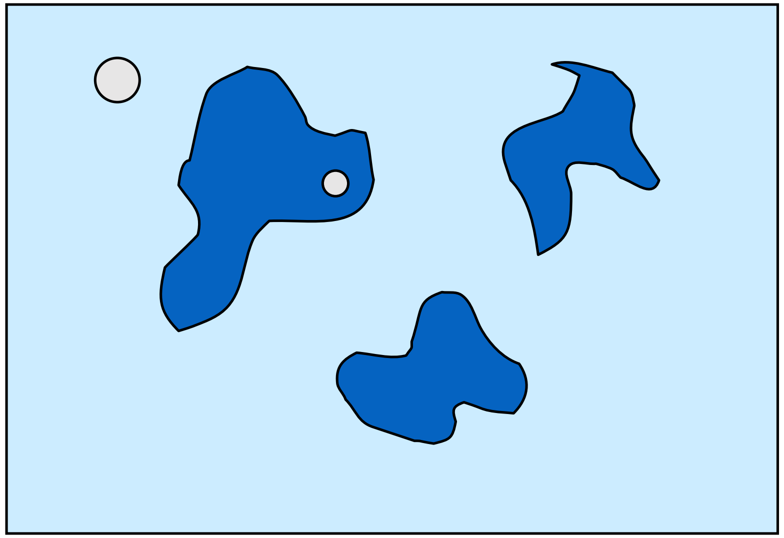
1. Compute random configuration
    - Collision checker
2. Connect configuration
    - Collision checker
    - Local method
3. Goto 1



Preprocessing: learning phase

- Iterative algorithm

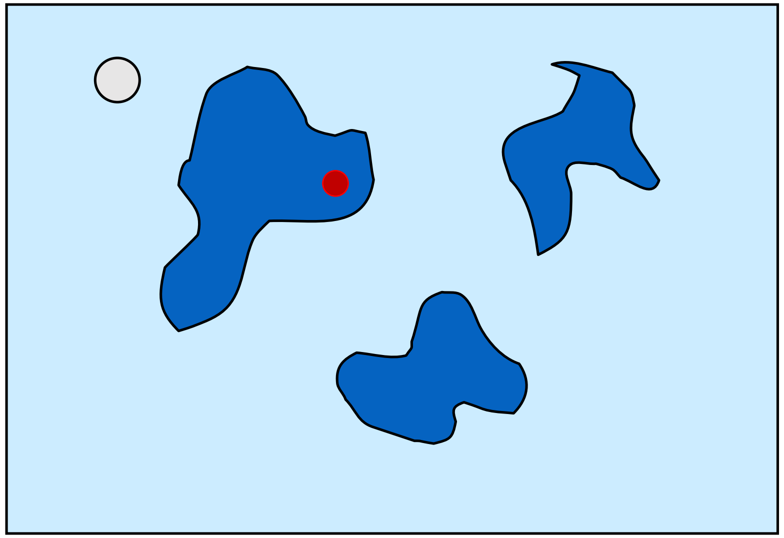
1. Compute random configuration
   - Collision checker
2. Connect configuration
   - Collision checker
   - Local method
3. Goto 1



Preprocessing: learning phase

- Iterative algorithm

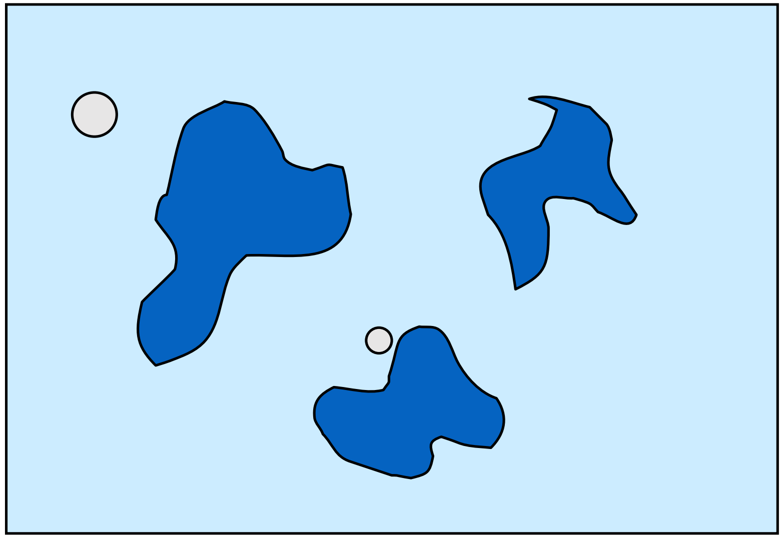
1. Compute random configuration
   - Collision checker
2. Connect configuration
   - Collision checker
   - Local method
3. Goto 1



Preprocessing: learning phase

- Iterative algorithm

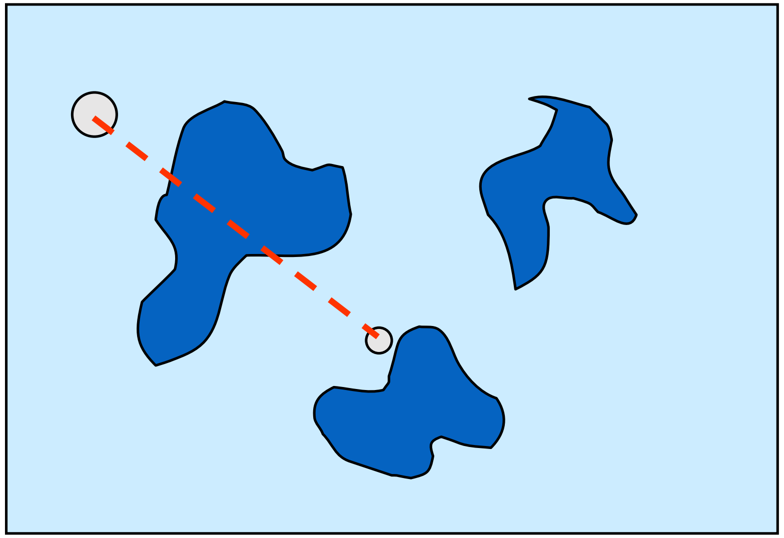
1. Compute random configuration
    - Collision checker
2. Connect configuration
    - Collision checker
    - Local method
3. Goto 1



Preprocessing: learning phase

- Iterative algorithm

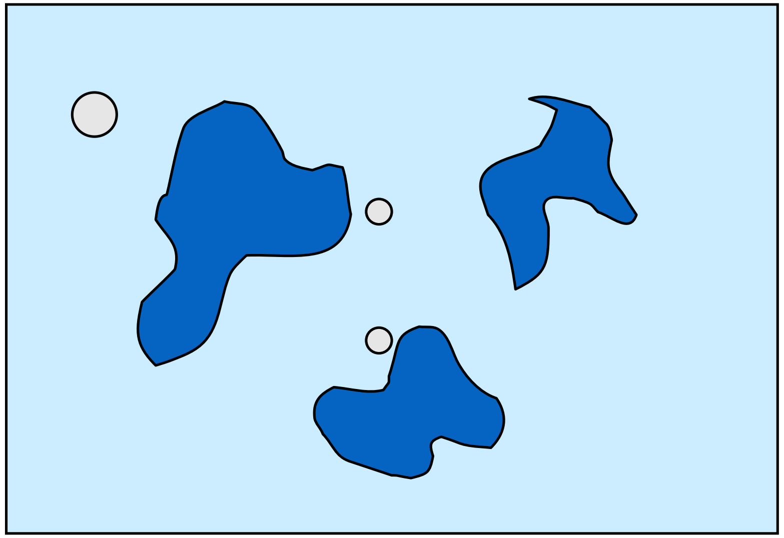
1. Compute random configuration
   - Collision checker
2. Connect configuration
   - Collision checker
   - Local method
3. Goto 1



Preprocessing: learning phase

- Iterative algorithm

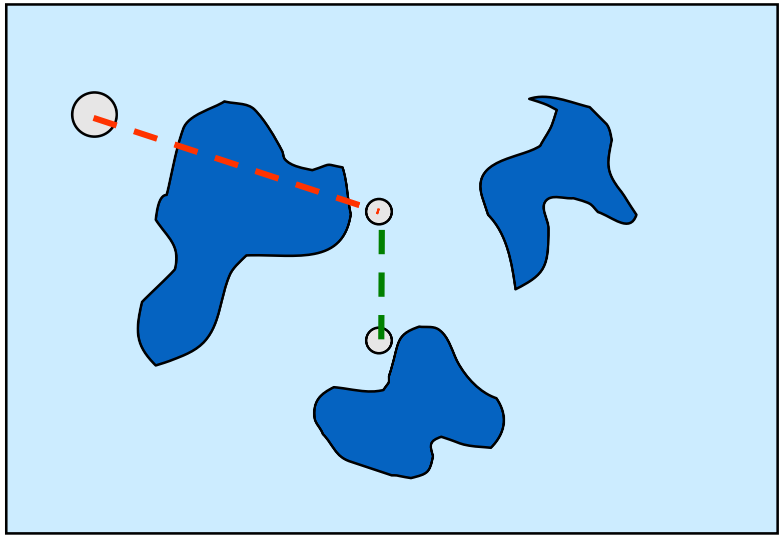
1. Compute random configuration
   - Collision checker
2. Connect configuration
   - Collision checker
   - Local method
3. Goto 1



Preprocessing: learning phase

- Iterative algorithm

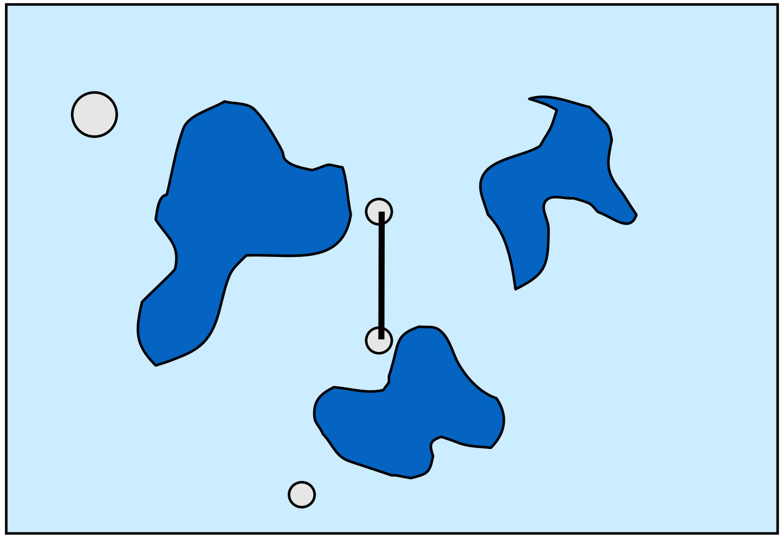
1. Compute random configuration
   - Collision checker
2. Connect configuration
   - Collision checker
   - Local method
3. Goto 1



Preprocessing: learning phase

- Iterative algorithm

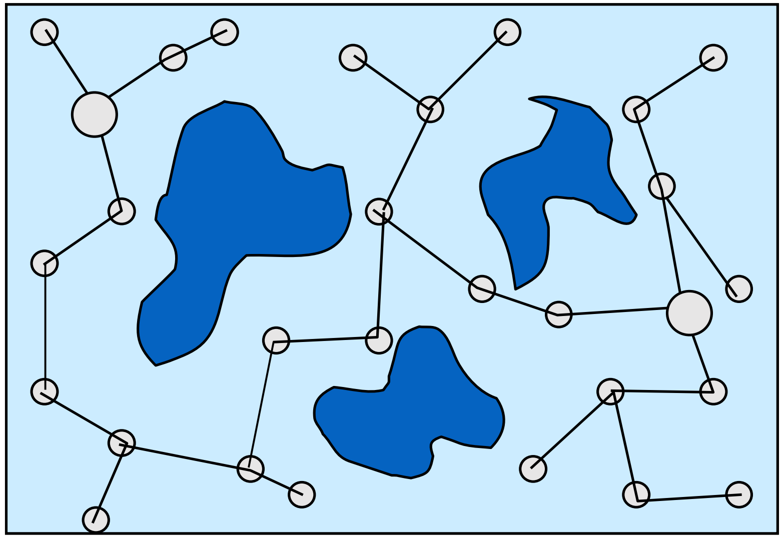
1. Compute random configuration
   - Collision checker
2. Connect configuration
   - Collision checker
   - Local method
3. Goto 1



Preprocessing: learning phase

- Iterative algorithm

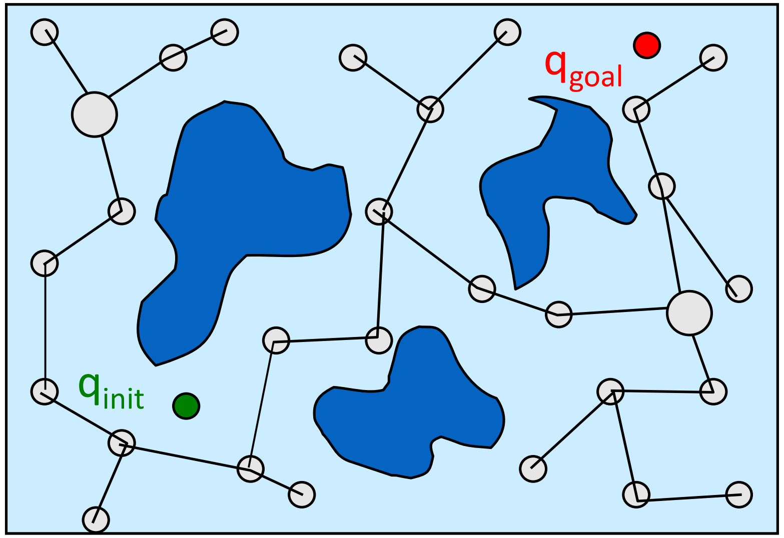
1. Compute random configuration
   - Collision checker
2. Connect configuration
   - Collision checker
   - Local method
3. Goto 1



Query Phase

- Roadmap is reused for solving queries

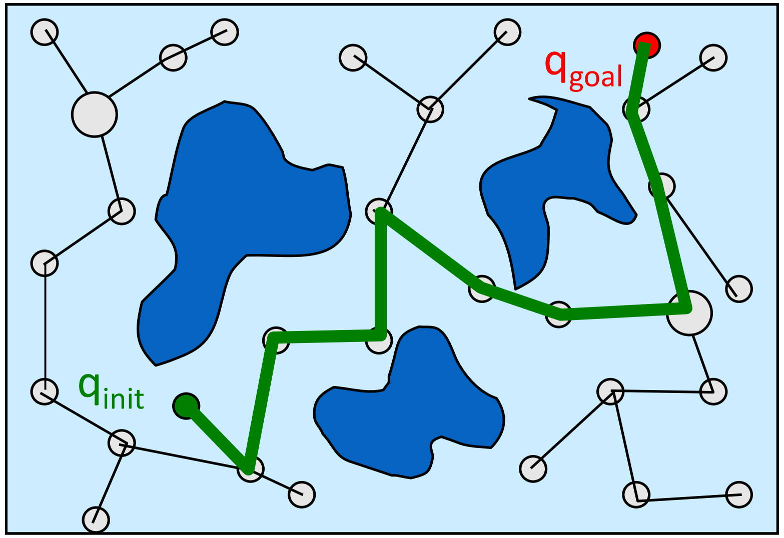
1. Connect desired initial and final configurations
2. If corresponding nodes belong to the same connected component, a solution exists
3. Graph Search



Query Phase

- Roadmap is reused for solving queries

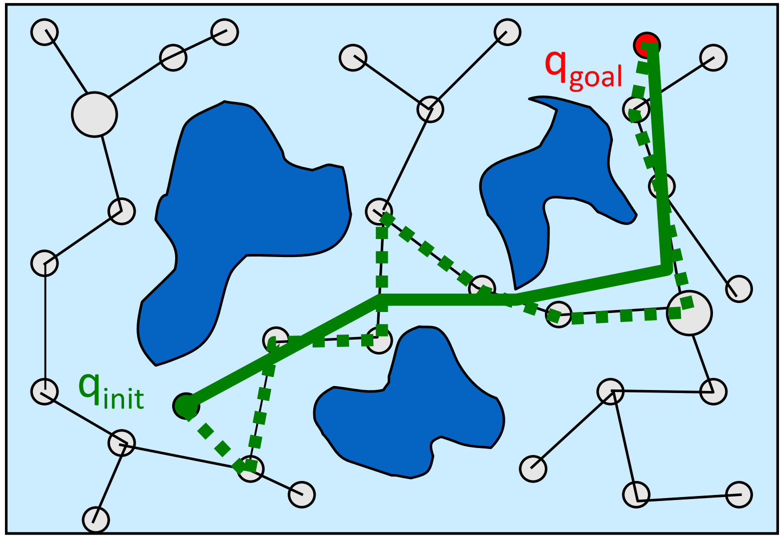
1. Connect desired initial and final configurations
2. If corresponding nodes belong to the same connected component, a solution exists
3. Graph Search



Query Phase

- Roadmap is reused for solving queries

1. Connect desired initial and final configurations
2. If corresponding nodes belong to the same connected component, a solution exists
3. Graph Search
4. Optimization

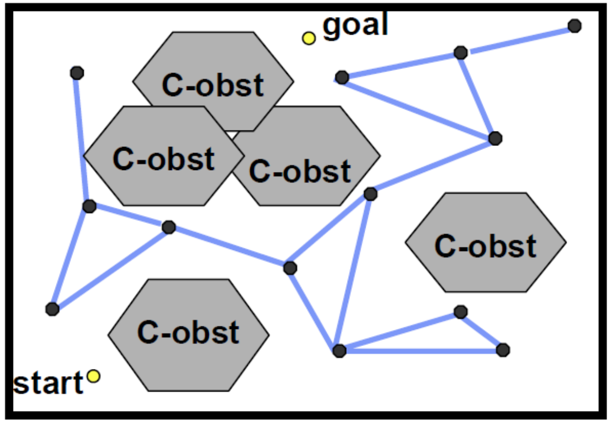


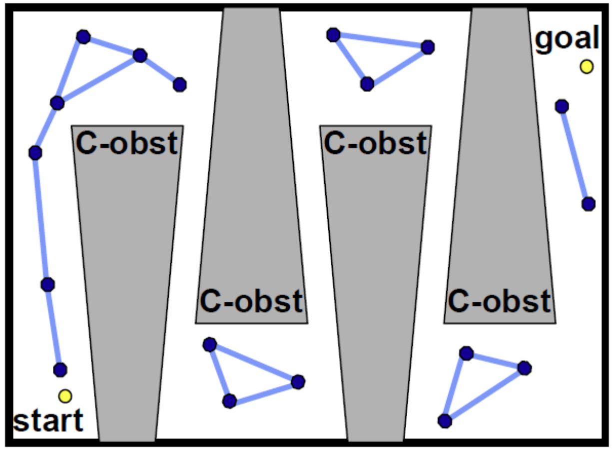
Good & bad news

- **Sample-based: The Good News**
  - Probabilistically complete
  - Do not construct the C-space
  - Apply easily to high-dimensional C-space
  - support fast queries w/ enough preprocessing

→Many success stories where PRMs solve previously unsolved problems

- **Sample-Based: The Bad News**
  - Don't work as well for some problems:
    - unlikely to sample nodes in *narrow passages*
    - hard to sample/connect nodes on constraint surfaces
  - No optimality or completeness



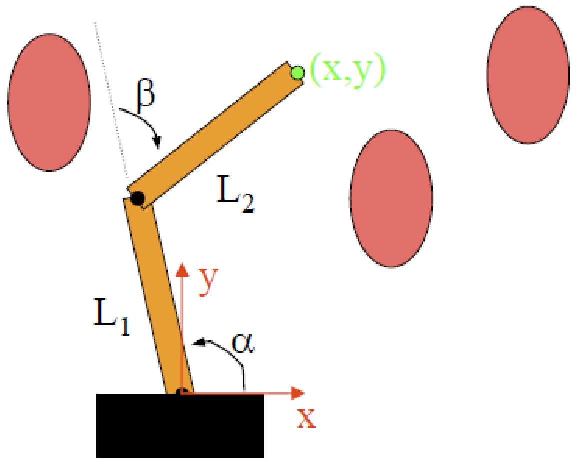


Distance functions

- Really, D should reflect the likelihood that the planner will fail to find a path
  - close points, likely to succeed
  - far away, less likely
- Ideally, this is probably related to the area swept out by the robot
  - very hard to compute exactly
  - usually heuristic distance is used
- Typical approaches
  - Euclidean distance on some embedding of c-space
- Embedding is often based on control points (recall end of potential field chapter)
  - Alternative is to create a weighted combination of translation and rotational "distances"
  - Workspace volume

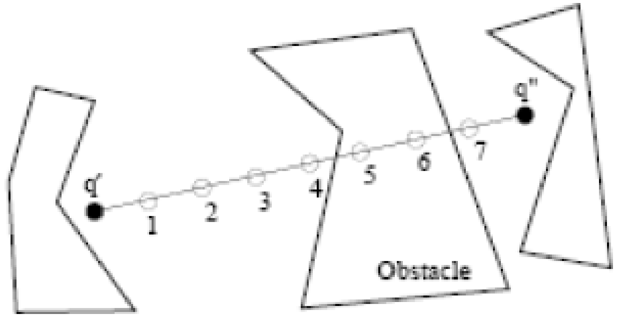
- Example:

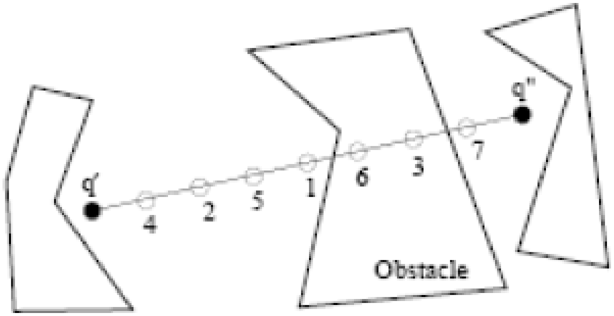
$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} L_1 c_\alpha \\ L_1 s_\alpha \end{pmatrix} + \begin{pmatrix} L_2 c_{\alpha+\beta} \\ L_2 s_{\alpha+\beta} \end{pmatrix}$$



Local planner

- Collision checker
- How to choose step size?





Expansion

- Sometimes G consists of several large and small components which do not effectively capture the connectivity of Qfree
- The graph can be disconnected at some narrow region
- Assign a positive weight w(c) to each node c in V
- w(c) is a heuristic measure of the “difficulty” of the region around c. So w(c) is large when c is considered to be in a difficult region. We normalize w so that all weights together add up to one. The higher the weight, the higher the chances the node will get selected for expansion.

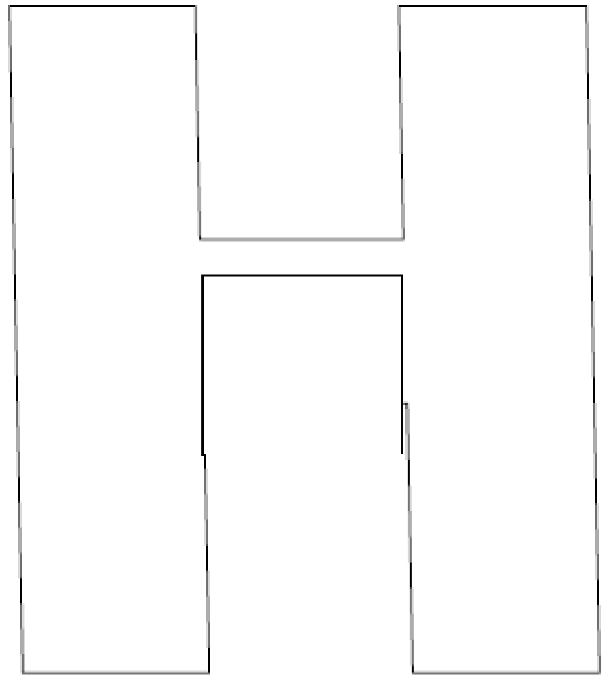
- Can pick different heuristics
  - Count number of nodes of V lying within some predefined distance of c.
  - Check distance D from c to nearest connected component not containing c.
  - Use information collected by the local planner. (If the planner often fails to connect a node to others, then this indicates the node is in a difficult area).

What if we fail

- Maybe the roadmap was not adequate.
- Could spend more time in the Learning Phase
- Could do another Learning Phase and reuse R constructed in the first Learning Phase. In fact, Learning and Query Phases don’t have to be executed sequentially.

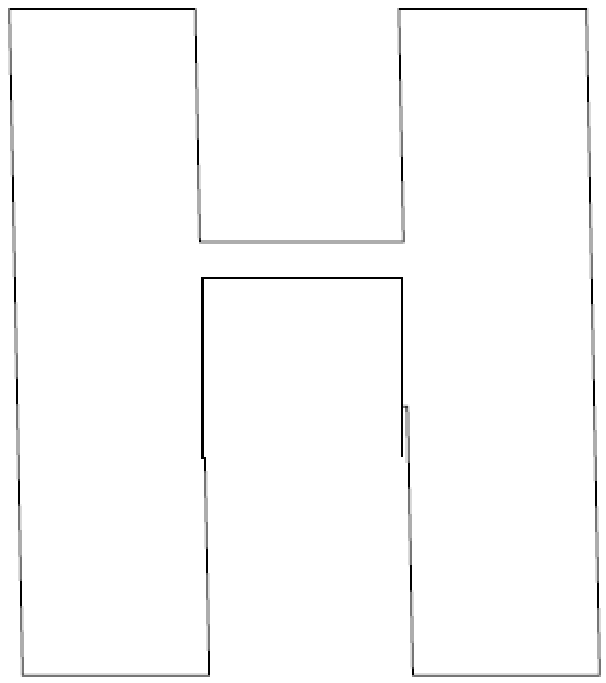
Sampling Strategies

- Uniform is good because it is easy to implement but is bad because of
- Learning Phase
  - Construction Step
    - Uniform sampling
    - New sampling
  - Expansion Step
    - Uniform around neighbor (local repair)
    - New sampling
- Query Phase



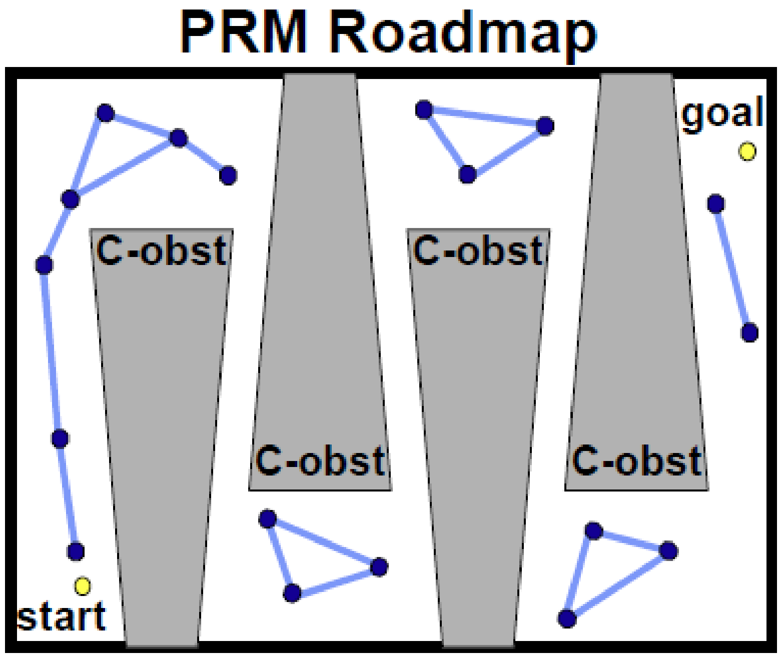
Different Strategies

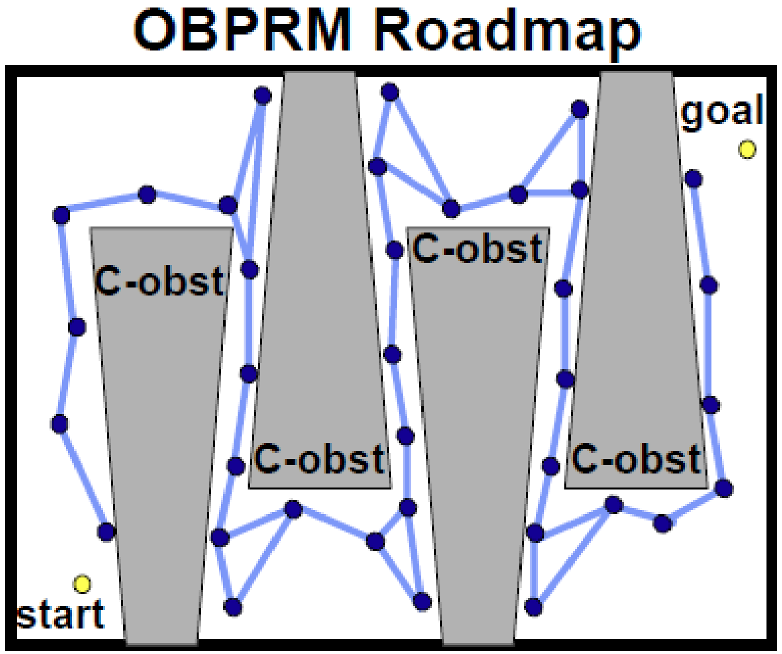
- Near obstacles
- Narrow passages
- Visibility-based
- Manipulatibility-based
- Quasirandom
- Grid-based



Sample Near Obstacles

- OBPRM
  - qin found in collision
  - Generate random direction v
  - Find qout in direction v that is free
  - Binary search from qin to obstacle boundary to generate node
- Gaussian sampler
  - Find a q1
  - Find another q2 picked from a Gaussian distribution centered at q1
  - If they are both in collision or free, discard. Otherwise, keep the free
- Dilate the space (pushed back via a clever resampling)

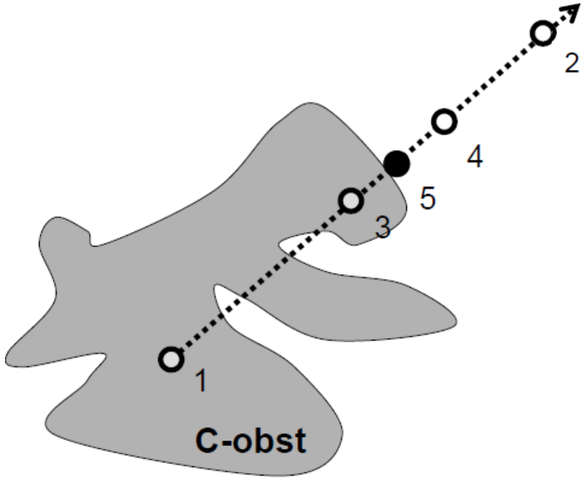


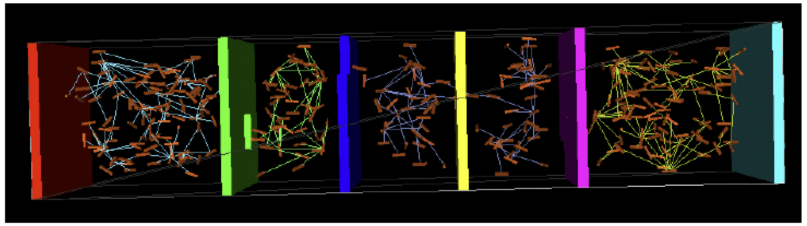


OBPRM: Finding Points on C-obstacles

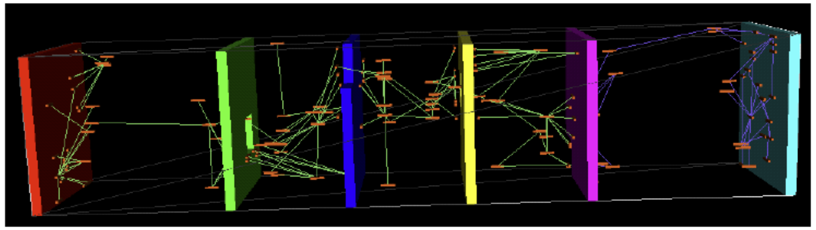
- **Basic Idea:**

1. Find a point in S's C-obstacle (robot placement colliding with S)
2. Select a random direction in C-space
3. Find a free point in that direction
4. Find boundary point between them using binary search (collision checks)





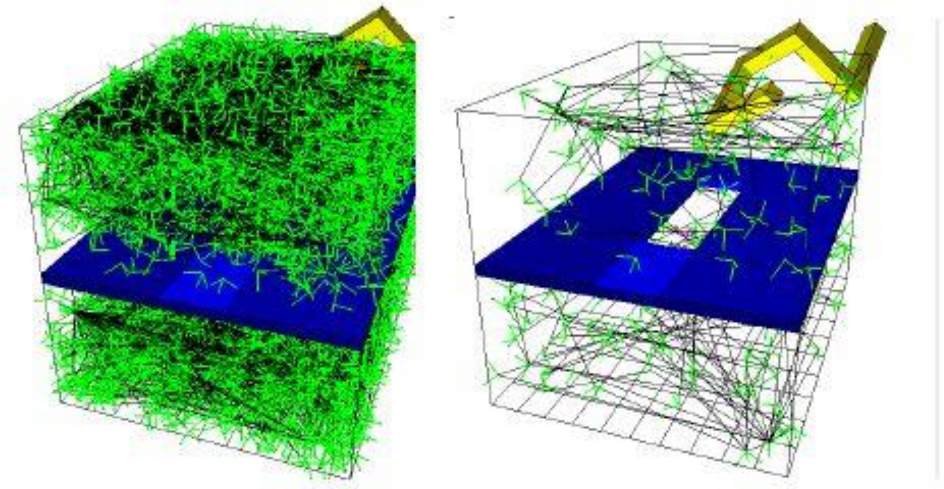
PRM
- 328 nodes
- 4 major CCs



OBPRM
- 161 nodes
- 2 major CCs

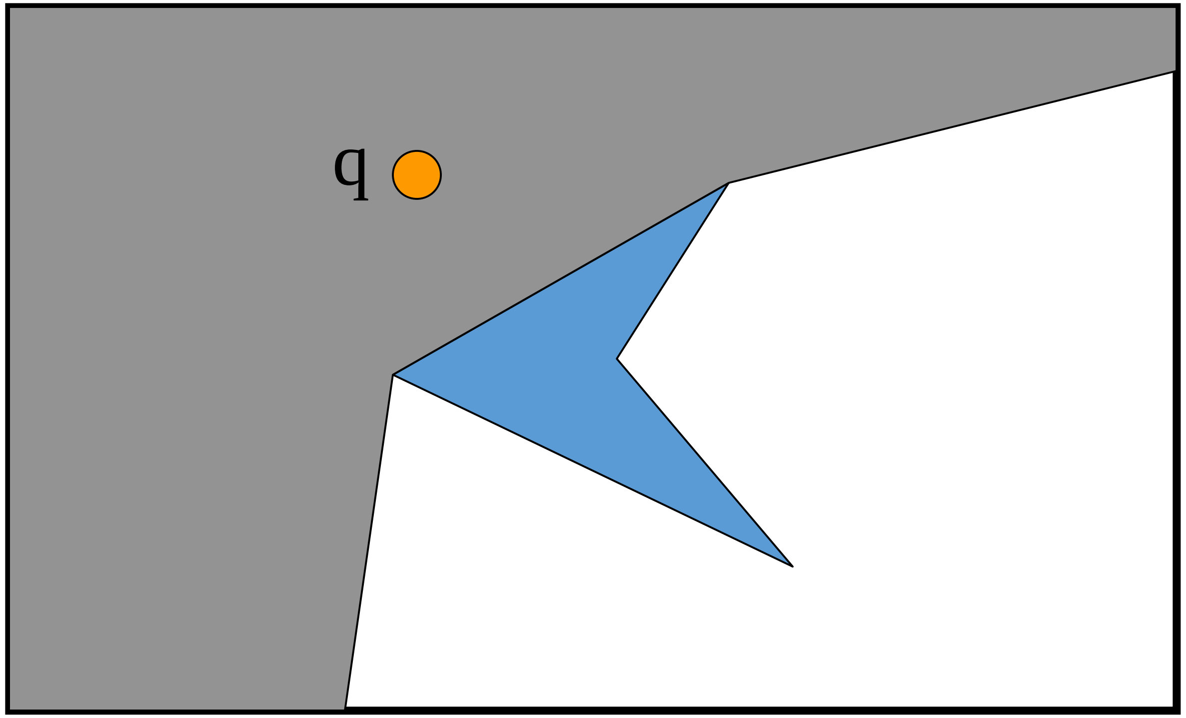
Sampling Strategy

- Highly constrained problems result in huge roadmaps:
  - Construction is time consuming
  - Search is time consuming
- Sampling Strategies help in reducing the roadmap size
- Example:
  - Visibility-PRM



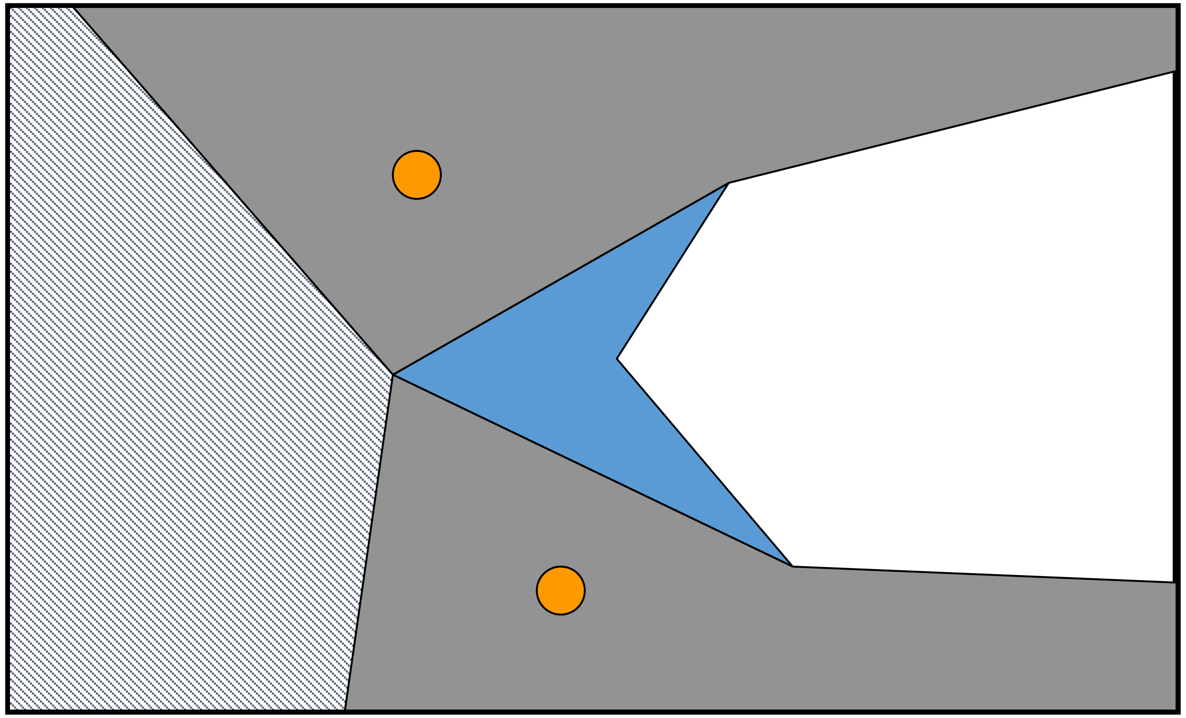
Visibility-PRM

Visibility Domain of configuration q:



Visibility-PRM

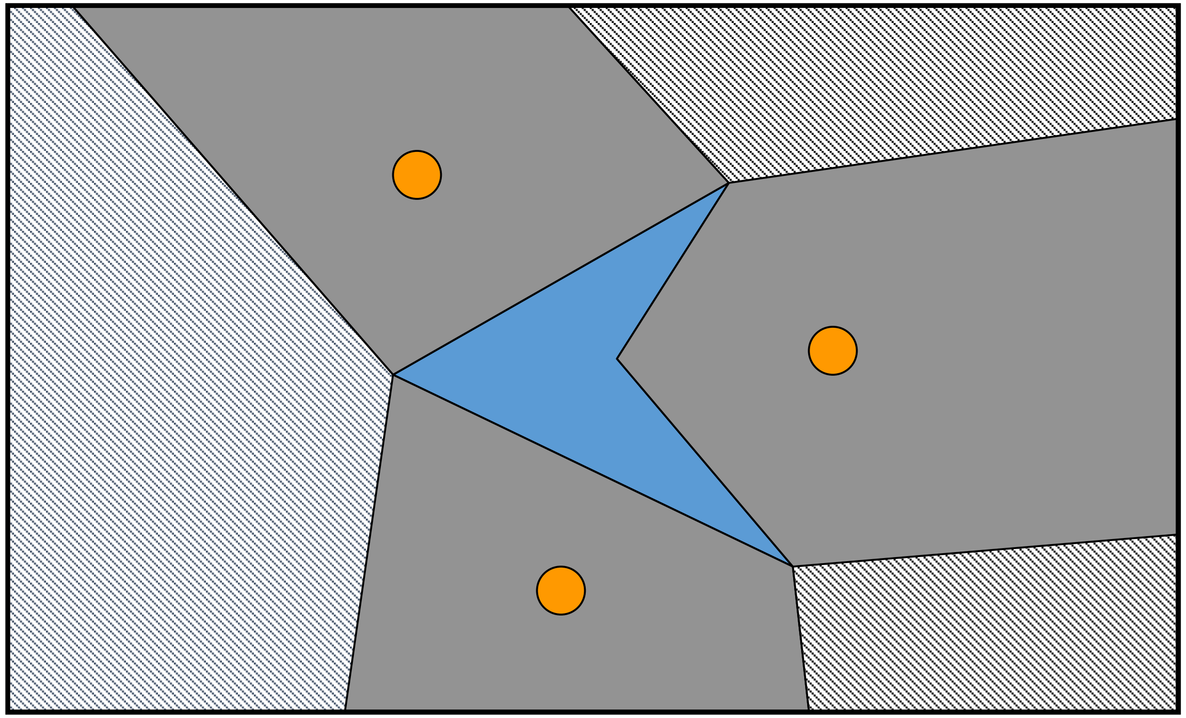
A new configuration is retained only if out of the visibility domain of other configurations



Visibility-PRM

A new configuration is retained only if out of the visibility domain of other configurations

These configurations are called “guardians”

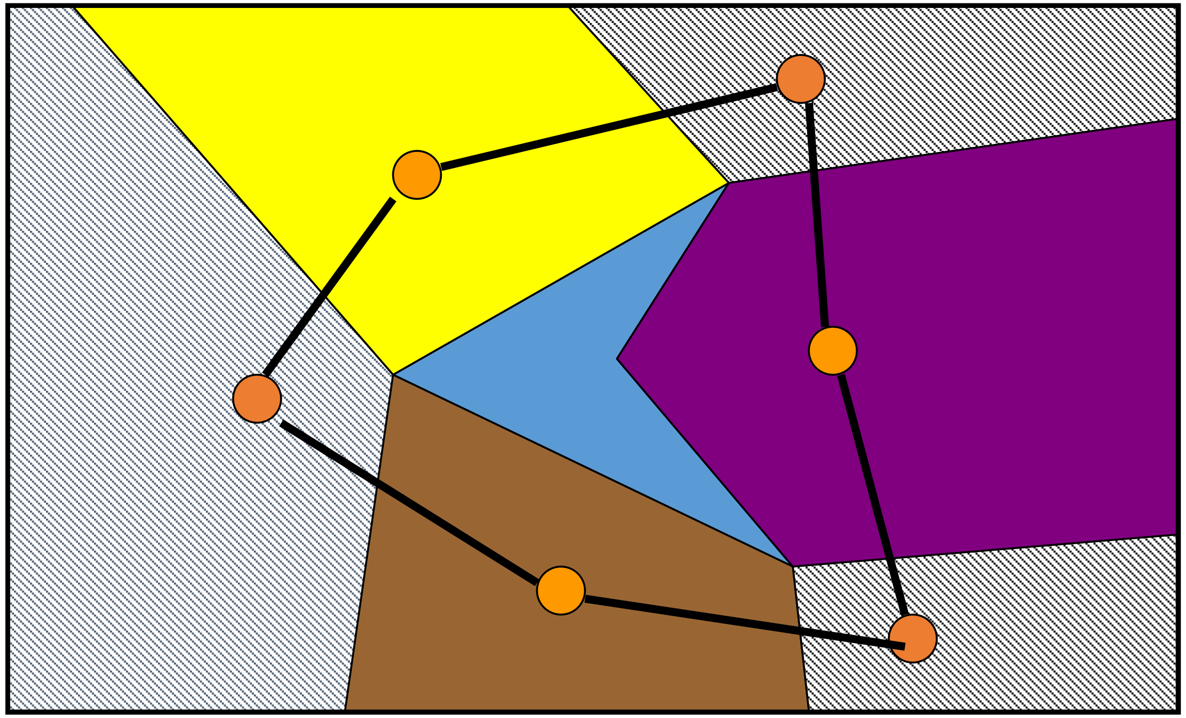


Visibility-PRM

A new configuration is retained only if out of the visibility domain of other configurations

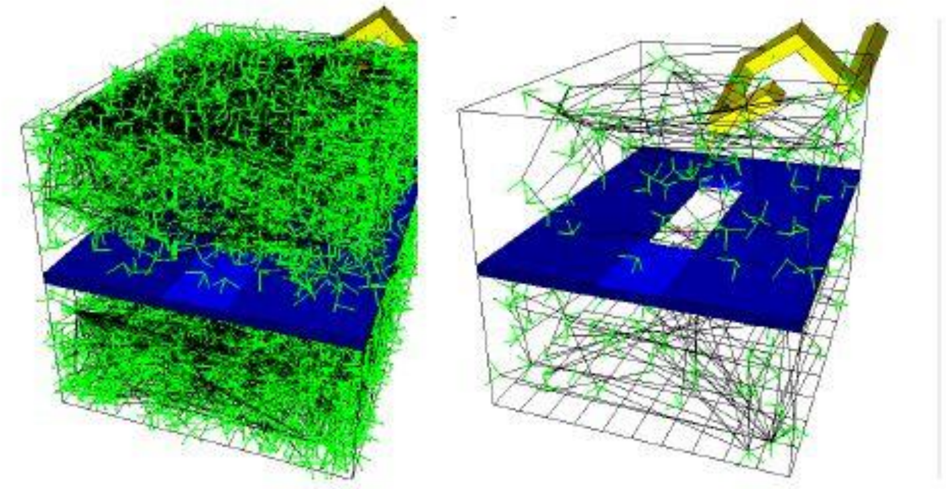
Or if allow to connect 2 guardians

These configurations are called “connectors”



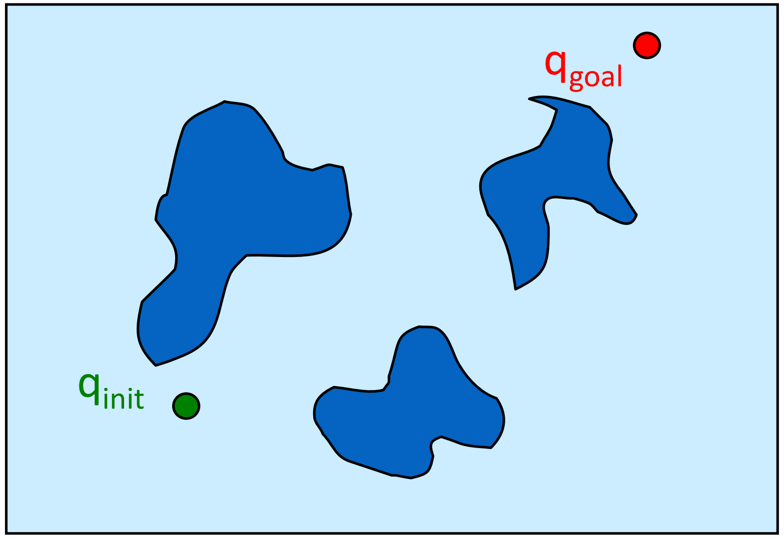
Visibility-PRM

(This is a 6-dimensional C-space in 3-D)



RRT: Rapidly-exploring Random trees

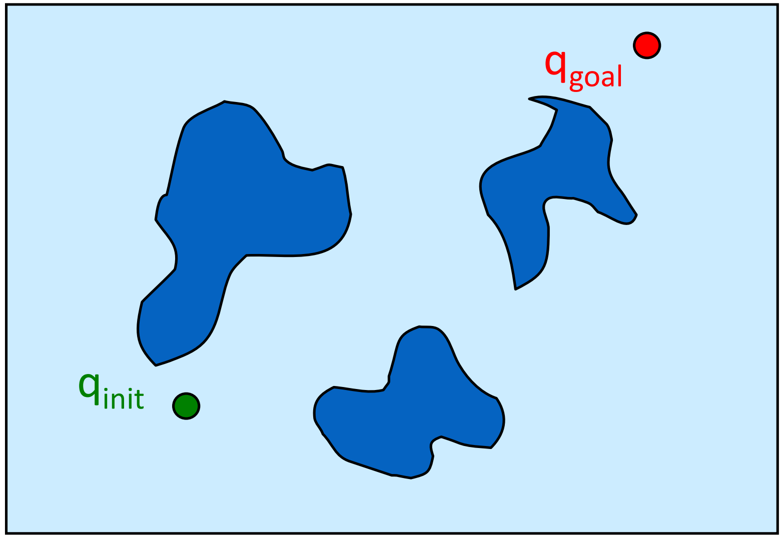
- PRM is a multi-query method: the same roadmap is reused to solve different queries
- RRT is single-query: the problem is solved without preliminary exploration of C-free



RRT: Rapidly-exploring Random trees

- Iterative algorithm:

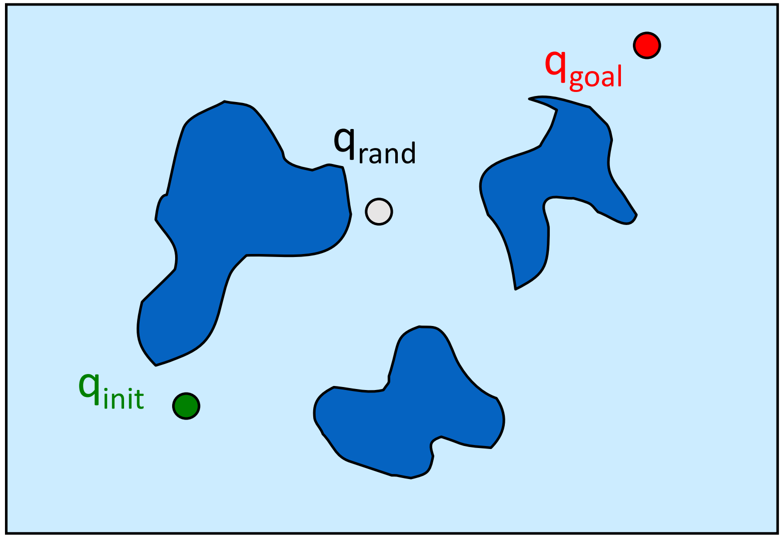
1. Compute q_{rand}
2. Connect to q_{near}
3. Insert q_{new}
4. Goto 1



RRT: Rapidly-exploring Random trees

- Iterative algorithm:

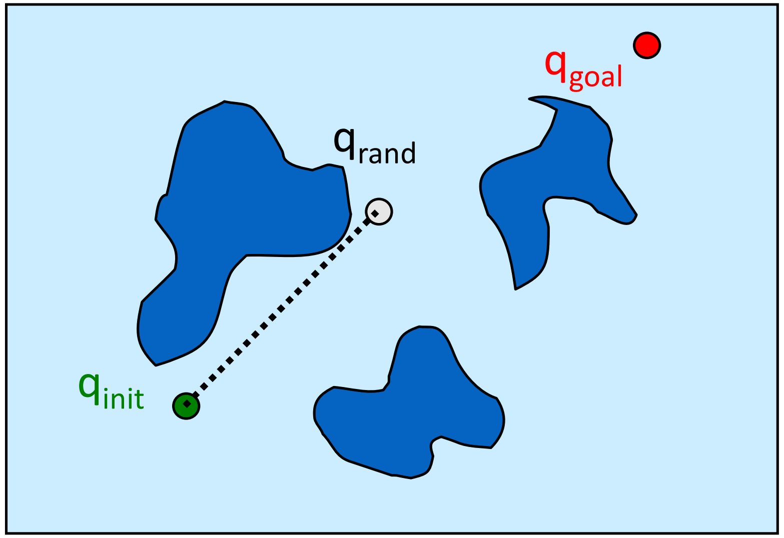
1. Compute q_{rand}
2. Connect to q_{near}
3. Insert q_{new}
4. Goto 1



RRT: Rapidly-exploring Random trees

- Iterative algorithm:

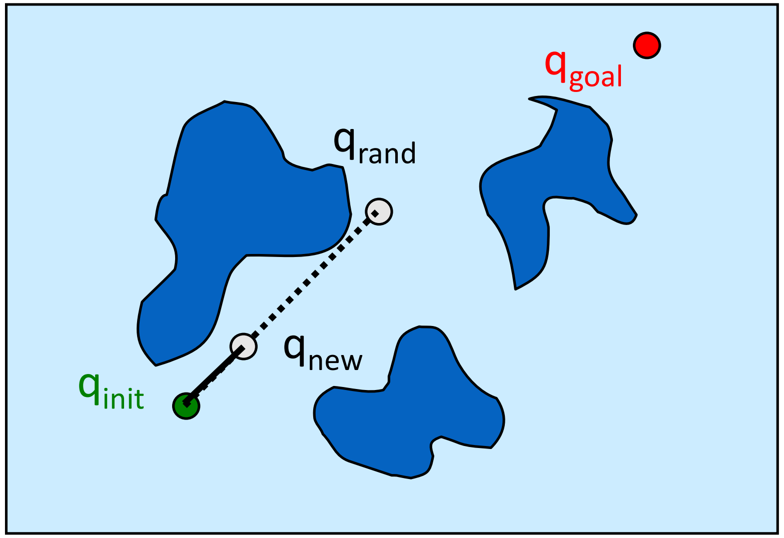
1. Compute q_{rand}
2. Connect to q_{near}
3. Insert q_{new}
4. Goto 1



RRT: Rapidly-exploring Random trees

- Iterative algorithm:

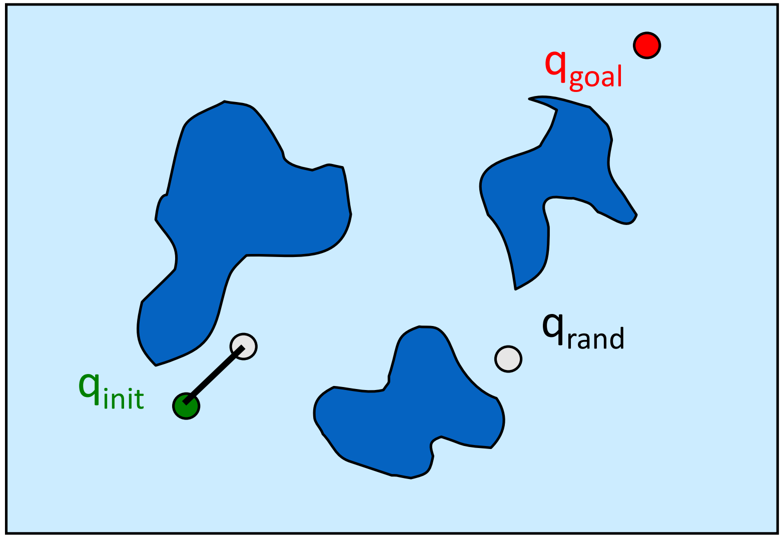
1. Compute q_{rand}
2. Connect to q_{near}
3. Insert q_{new}
4. Goto 1



RRT: Rapidly-exploring Random trees

- Iterative algorithm:

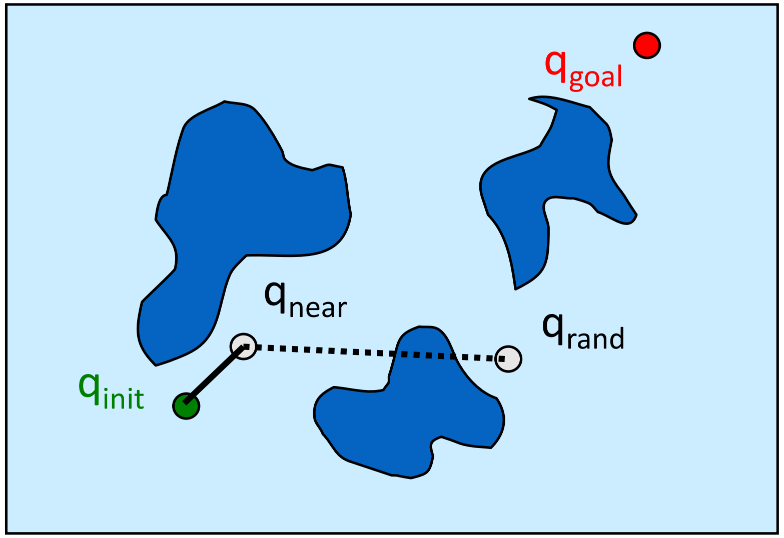
1. Compute q_{rand}
2. Connect to q_{near}
3. Insert q_{new}
4. Goto 1



RRT: Rapidly-exploring Random trees

- Iterative algorithm:

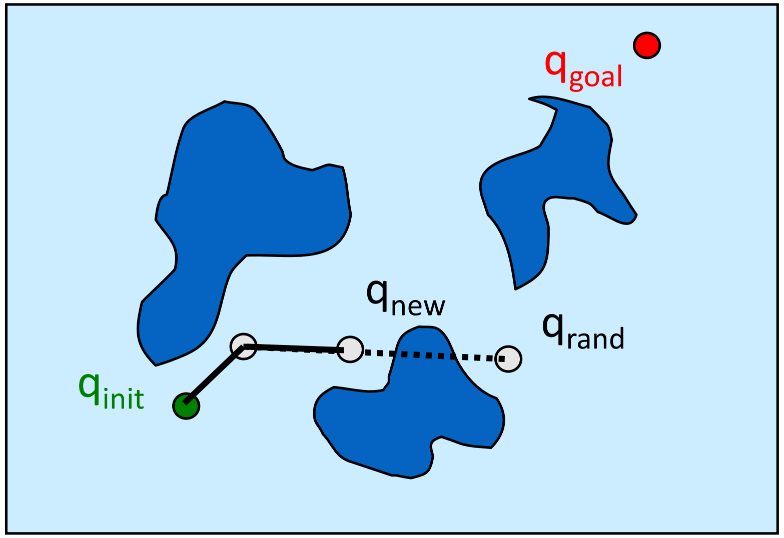
1. Compute q_{rand}
2. Connect to q_{near}
3. Insert q_{new}
4. Goto 1



RRT: Rapidly-exploring Random trees

- Iterative algorithm:

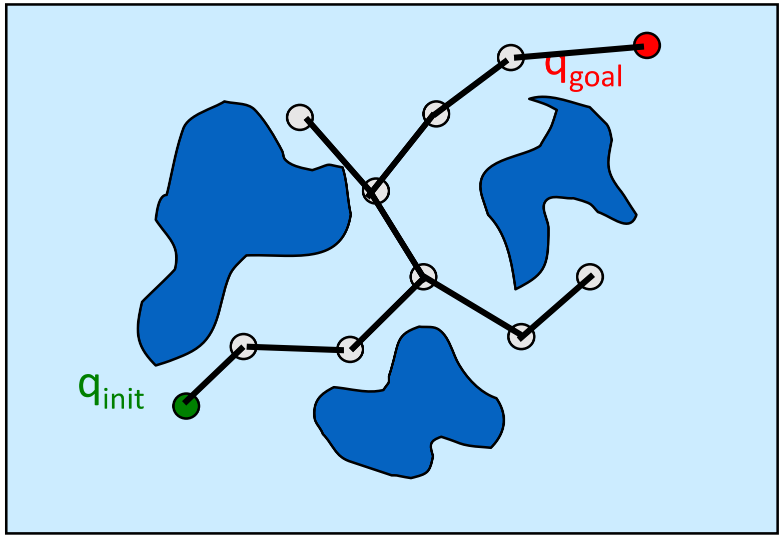
1. Compute q_{rand}
2. Connect to q_{near}
3. Insert q_{new}
4. Goto 1



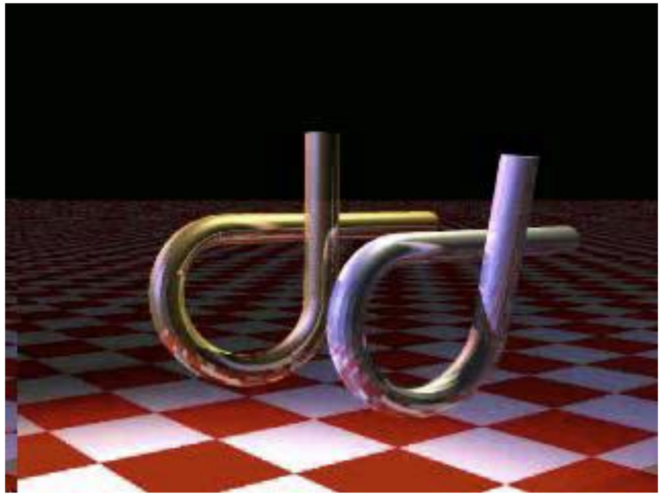
RRT: Rapidly-exploring Random trees

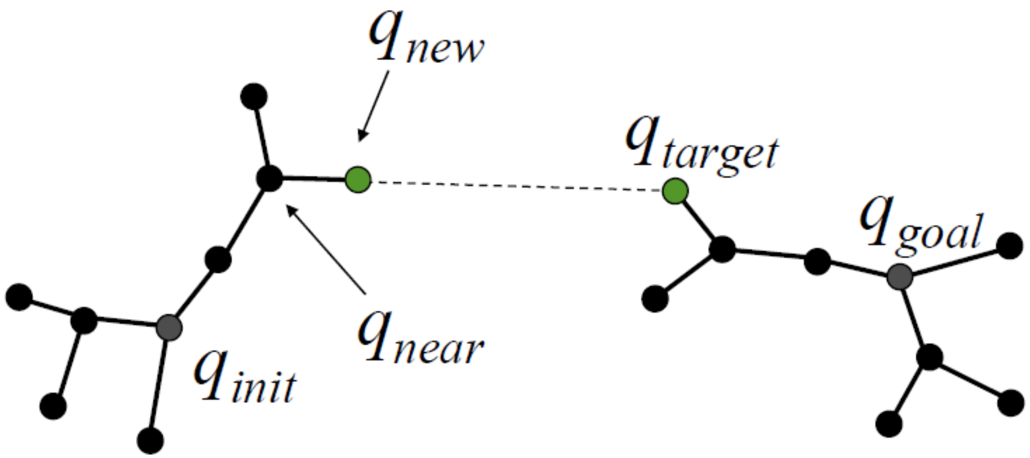
- Iterative algorithm:

1. Compute q_{rand}
2. Connect to q_{near}
3. Insert q_{new}
4. until q_{new} and q_{goal} connected

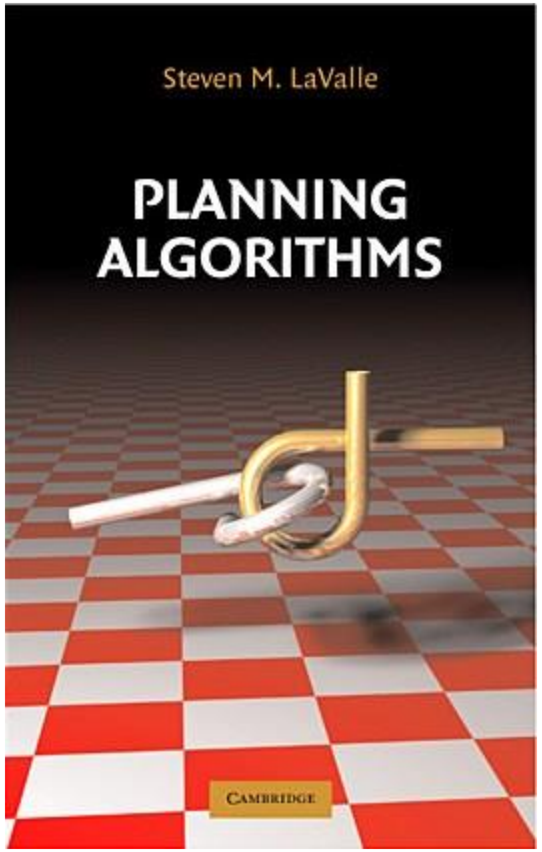


Grow two RRTs towards each other



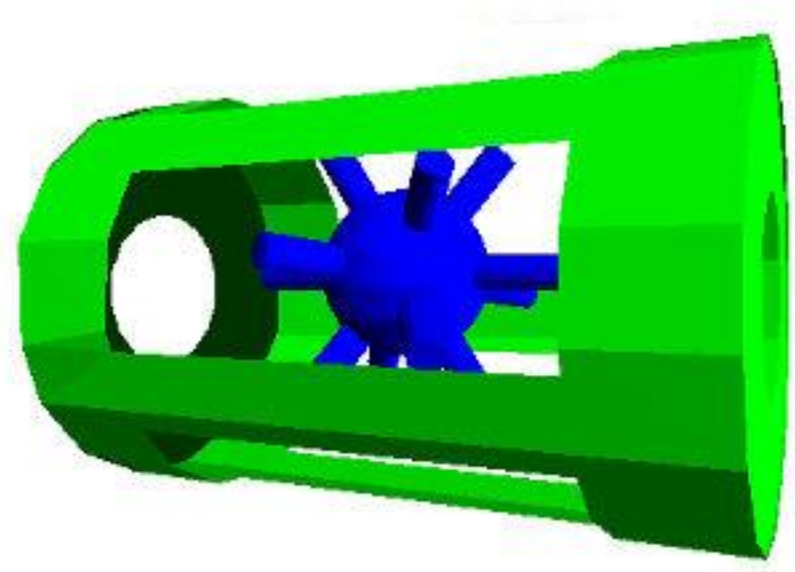


More…



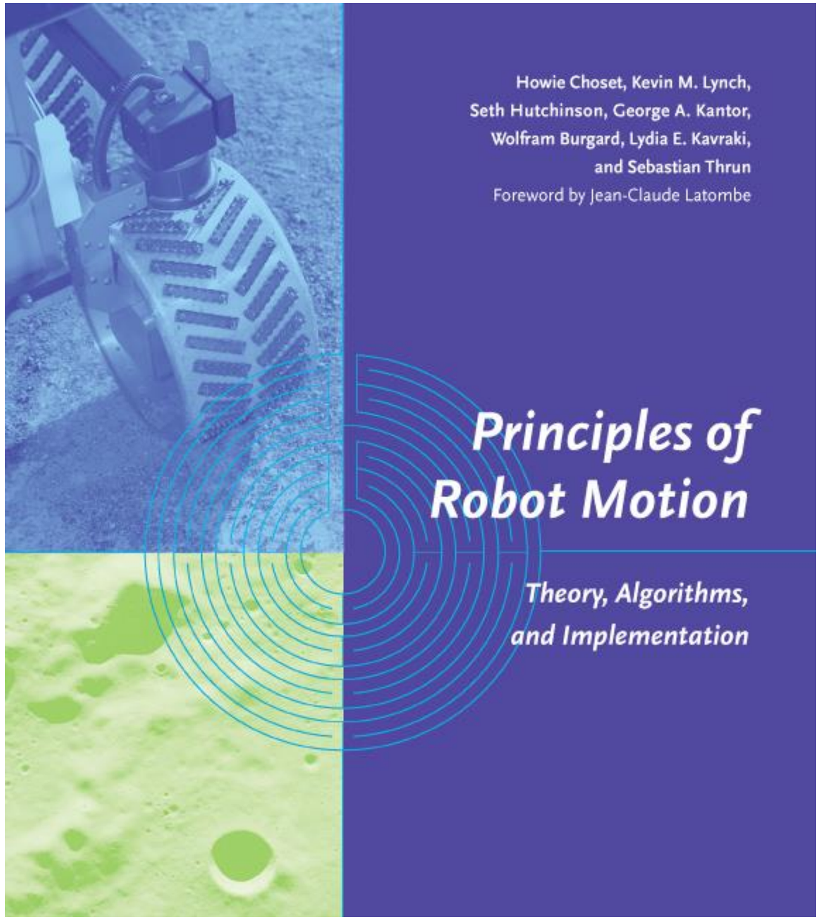
- Planning Algorithms
  Steven M. LaValle
  http://planning.cs.uiuc.edu/

More…



- Nancy Amato's webpage
- https://parasol.tamu.edu/people/amato/
- courses+benchmark

More…



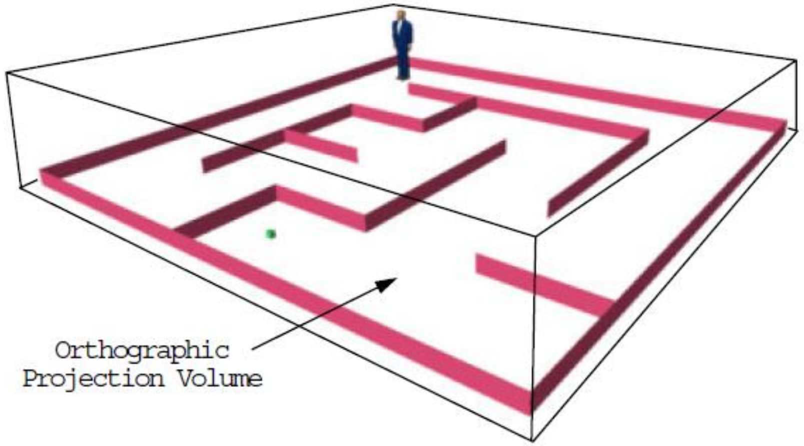
- CMU robot motion planning course

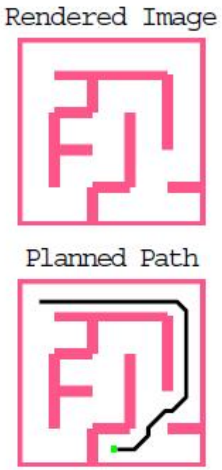
More

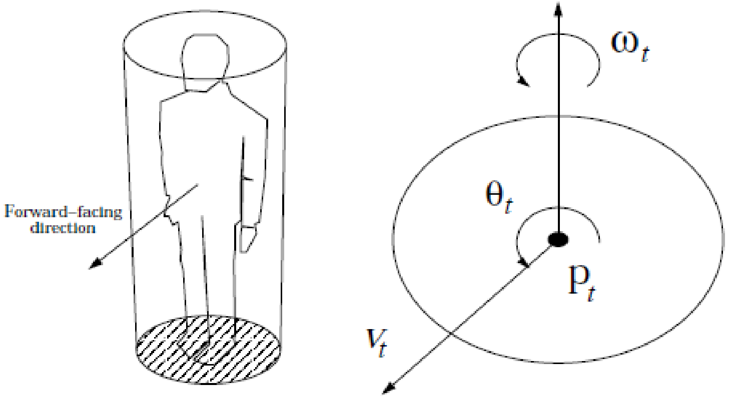
- OMPL

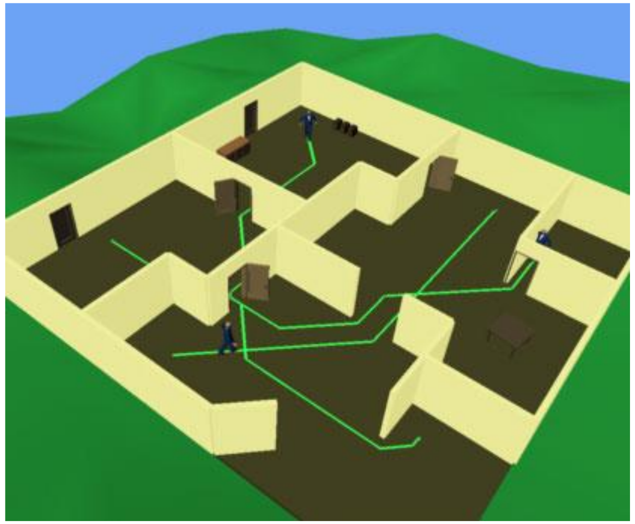
- http://ompl.kavrakilab.org/

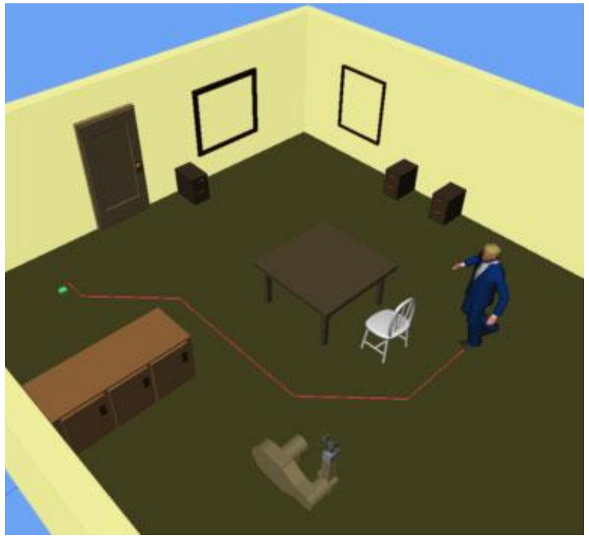
[Kuffner 98] Grid-based planning



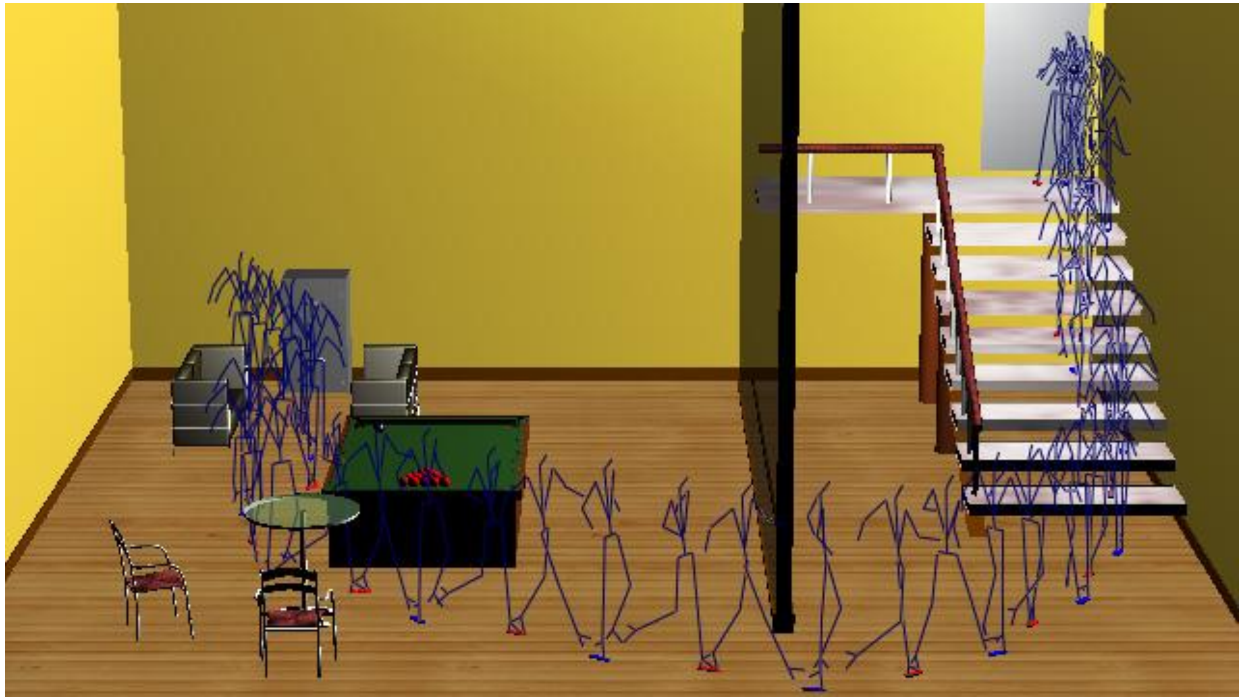








[Choi 03] Planning Bidep Locomotion



End of first part

Any questions?

My question:
How to handle dynamic (changing) environments in planning?

Towards Smarter Cinematographic Drones

Marc Christie
IRISA / INRIA Rennes, FRANCE

Drone Cinematography

Drone Cinematography

A wide-spread technique in the past 10 years (drone film festivals)

See “The circle” movie (DJI) entirely shot by a drone. Cheap technology gives aspiring producers ability to match hollywood (see this drone)

Drone Cinematography

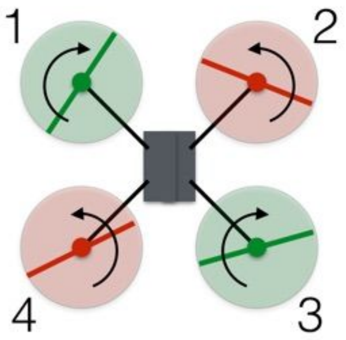
A wide-spread technique in the past 10 years (drone film festivals)

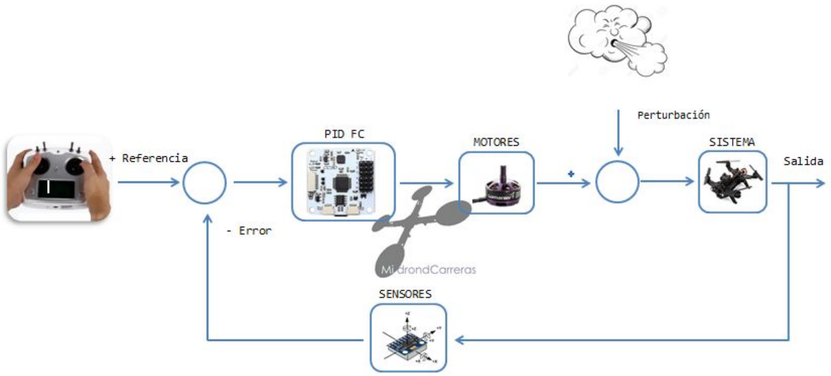
See “The circle” movie (DJI) entirely shot by a drone. Cheap technology gives aspiring producers ability to match hollywood

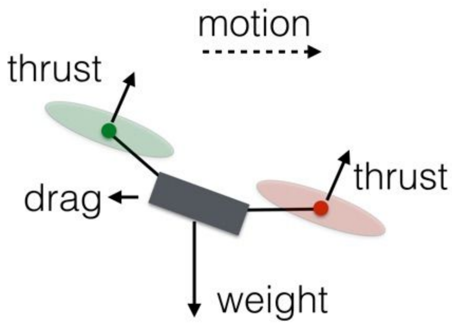
What is a Drone, How Do You Control It?

Drone = autonomous control

- Four engine speeds to regulate
- A PID controller uses the difference between a current configuration and a desired configuration to compute four speed signals







A Challenging task

No Film grammar for drones (yet) - see multidrone.eu

Generally two persons required :

- one to control drone’s motion
- one to control drone’s orientation (framing)

Requires skilled operators

=> very hard to synchronize with objects in motion

=> timing is essential

How Smart Are Commercial Drones Today?

- Follow-me technologies to frame a target
  - Using the GPS position of a target
  - Or uses image-based visual tracking
- Control by gestures
  - Image-based analysis (take off, approach, left, right, up)

Can we make them even smarter?

- Can they decide on optimal view angles?
- Can they compute qualitative motions?
- Can they understand cinematographic language?

SmartER Drone Cinematography

Research challenges:

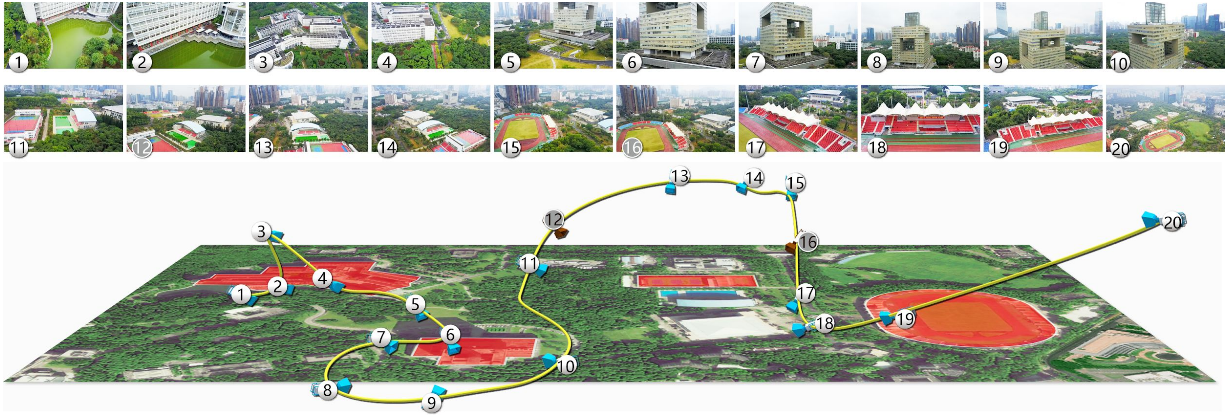
- *Formalize* film knowledge for drones
- Plan paths of *cinematographic quality* at a low computational cost
- *Ensuring safety* at all times

PART 1 - Automated Cinematographic Drones

Drone Videography for flybys [SIG-18]

Motivations:

- ➢ Generate an aesthetic flyby of given buildings



Issues:

- ➢ Complex tasks for novice users
- ➢ Requires multiple trials
- ➢ Generated videos are often not qualitative

Drone Videography for flybys [SIG-18]

Motivations:

- ➢ An aesthetic flyby of given buildings and their environments

User tasks:

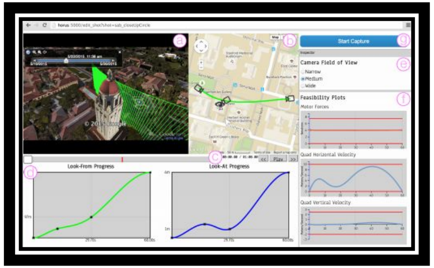
- ➢ Choose the camera angles, choose the camera motions around buildings, choose the transitions between buildings?
- ➢ Create smooth (cinematographic) trajectories
- ➢ Ensure safety (eg. when drone is hidden by a building)

Issues:

- ➢ Complex tasks for novice users
- ➢ Requires multiple trials
- ➢ Generated videos are often not qualitative (for novice users)

Existing work

Horus

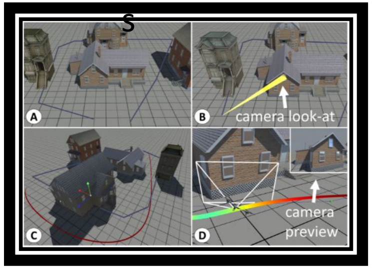


[Joubert et al., 2015]

- Intuitive Interface
- C4 trajectories
- Simulation

- Offline
- Dedicated to outdoor environments

Airways



[Gebhardt et al., 2016]

- Intuitive Interface
- C4 trajectories
- Obstacle Avoidance

- Offline
- Dedicated to static scenes

Drone Videography for flybys [SIG-18]

Automating this process is computationally complex:

- ➢ How to choose the best viewpoints among an infinity of possibilities? What is a “best viewpoint”?
- ➢ How to generate best trajectories? What is a “best trajectory”
- ➢ How to plan a complex sequence of trajectories?

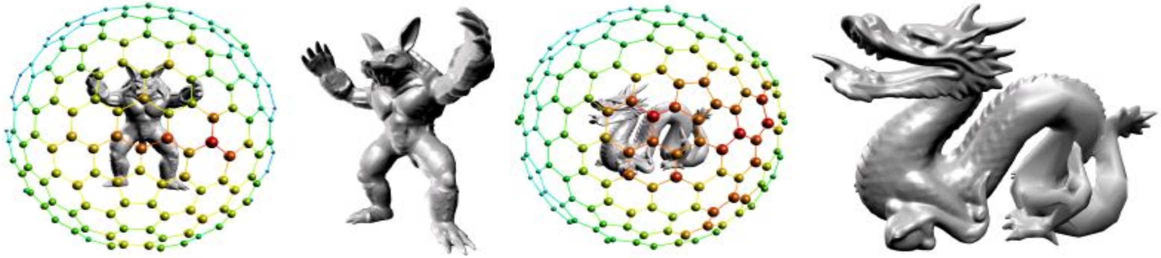
Our approach:

1. Provide a quality metric for views of buildings (called landmarks)
2. Generate qualitative **camera moves** around landmarks
3. Connect the different camera moves

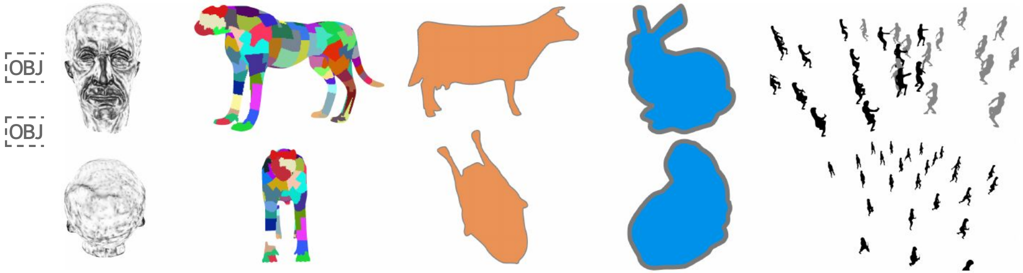
Viewpoint quality

Viewpoint entropy [Vasquez'01]

- ➢ Defines how much information a viewpoint conveys

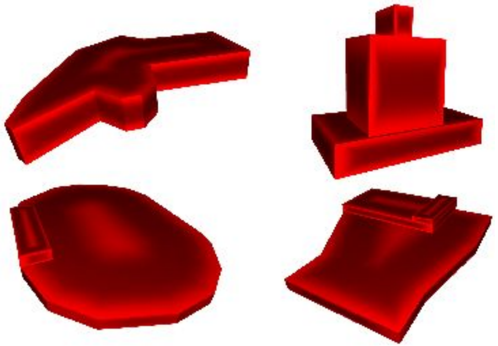


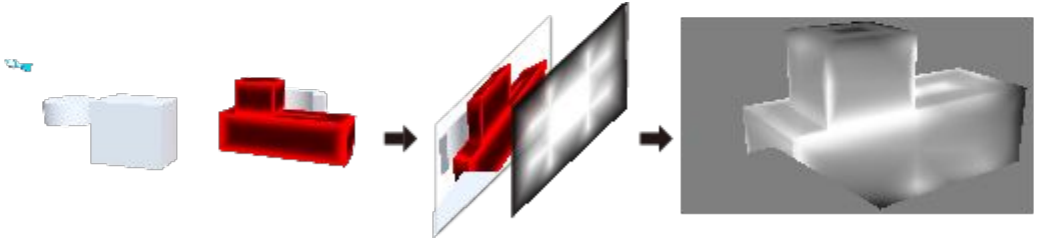
- ➢ Different critera (mean curvature, visibility, alignment, silhouette complexity, visual dispersion)



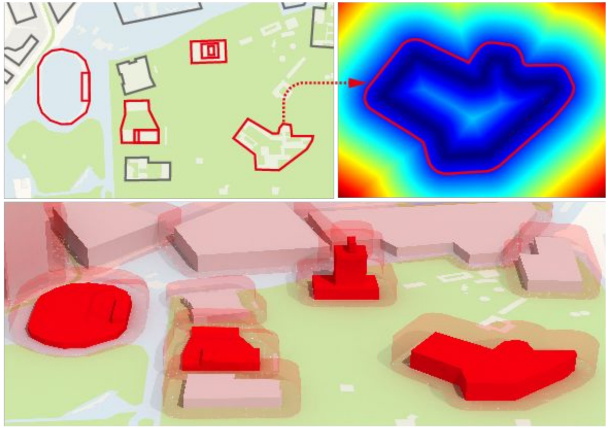
Viewpoint quality

1. Compute saliency of buildings:
   - ➢ Edges provide information on the shape
   - ➢ Centers of areas
2. Compute a “line of thirds” overlay
   - ➢ Regions in the center are preferred
   - ➢ Regions along the ⅓ axes are preferred
3. Compute both information
   - ➢ Results in a viewpoint quality (sum of information)





Ensuring safety



Expand 3D buildings with a safety area

- using a surface Minkowski sum (sphere)

Composing Viewpoint quality

- creates a scalar field through the scene

Creating camera moves (1)

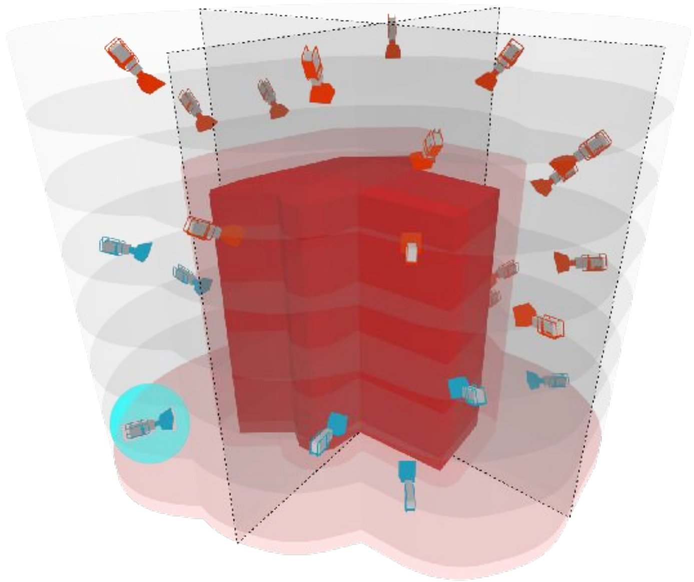
How to create *interesting moves* around a building?

- ➢ Should have a minimum change in height or in angle
- ➢ Should connect good viewpoints

We propose spatial partitions around each building

- ➢ Horizontal partitions (max 7 layers)
- ➢ Vertical partitions (4 partitions)

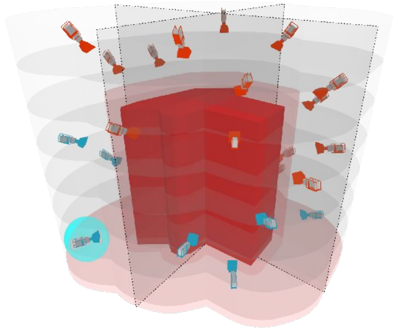
The best viewpoint is computed in each partition



Creating camera moves (2)

A subset of all possible moves is created

- ➢ Only create moves across a minimum of 4 partitions (horizontally + vertically)



Each trajectory is evaluated (192 possibilities)

- ➢ Quality of the viewpoints along the move

A selection of the *n* best moves is performed

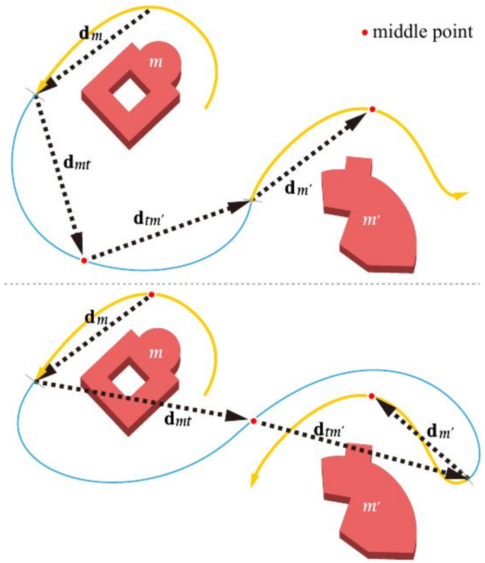
Chaining camera moves

Scene is composed of *m* landmarks

For each landmark: *n* best moves

How to compute an optimal trajectory?

- Generate all transitions between possible moves
- Evaluate the quality of each transition
  - ➢ Length, curvature, change of directions



Now each move has a quality (cost), each transition has a quality (cost), we search for the shorted path through landmarks

=> looks like a Travel Salesman Problem

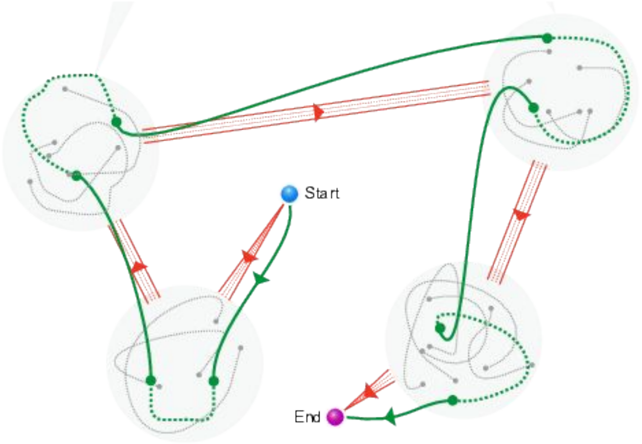
Solving: Set-TSP

A specific case of the TSP:

- we only need to visit ONE move per landmark
- corresponds precisely the Set TSP (or one-of-a-set TSP) [Noon93]

Table 1. Test scene statistics: number of landmarks (#m), the total time for computing view quality fields, local trajectory construction time, global optimization time, and distance of the global optimal trajectory in meters.

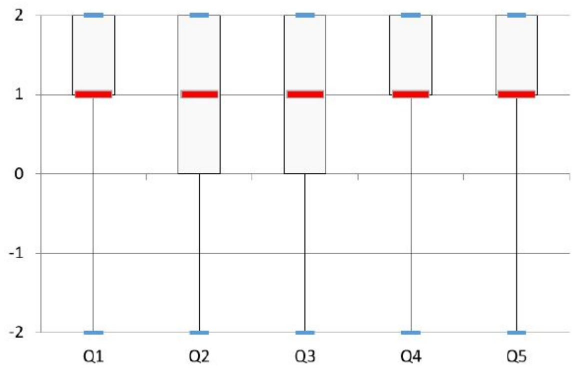
Figure	#m	Time$_f$	Time$_l$	Time$_g$	Distance
Fig. 1	3	5m	15s	10s	2475
Fig. 2	4	7m	37s	15s	2179
Fig. 12	3	5m	18s	15s	3190
Fig. 13	4	10m	41s	38s	3806
Fig. 14	5	8m	50s	31s	2998



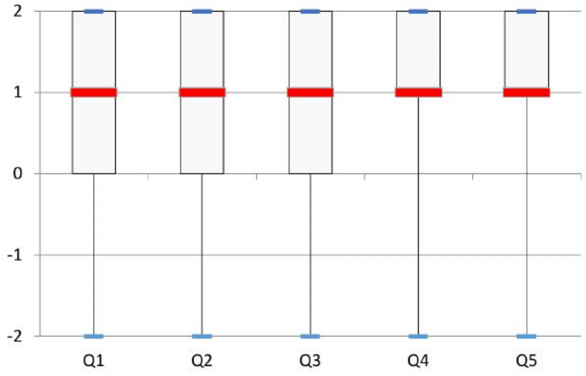
User evaluations

Compared three videos. Given the same landmarks:

- Auto: using our automated approach
- Manual: manually flying the drone to shoot the landmarks
- DGS: use DJI GS Pro software on iPad to design a drone path, and run it



(a) *Auto* vs. *Manual*



(b) *Auto* vs. *DGS*

Q1; more pleasing video
Q2:clearer overvies of landmarks
Q3:follows a more reasonable route
Q4:provide better transitions
Q5:create smoother trajectories

Results

The computed trajectories are sampled and sent as a sequence of GPS waypoints to the drone (DJI Phontom 3 Pro)

Part 2 - Interactive Cinematographic Drones

Joint work with technicolor

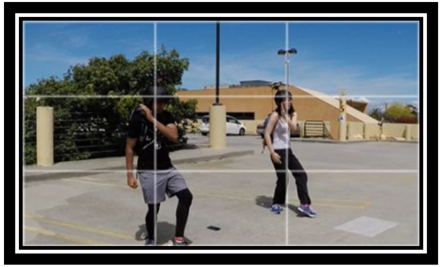
Motivations

- Have drones that can frame and “understand cinematographic language”
  - On angles: Over The Shoulder shot, Apex shot,
  - On sizes: Medium shot size
  - On framing: placement of targets on the screen

- Have drones that maintain cinematographic properties
  - Adapt to changes in the scene (actors locations and orientations)
  - Ensure cinematographic quality in camera motions

Existing work

Shot design

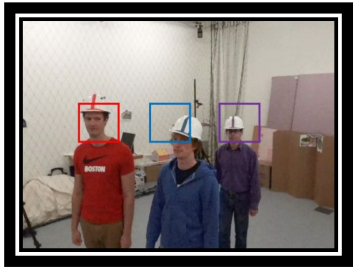


[Joubert et al., 2016]

- Based on the Toric Space
- Maintain a given framing

- No obstacle avoidance
- No visibility checking
- Limited motion of actors

Framing based control



[Nageli et al., 2016]

- Realtime
- Obstacle Avoidance
- Frame more than 2 actors

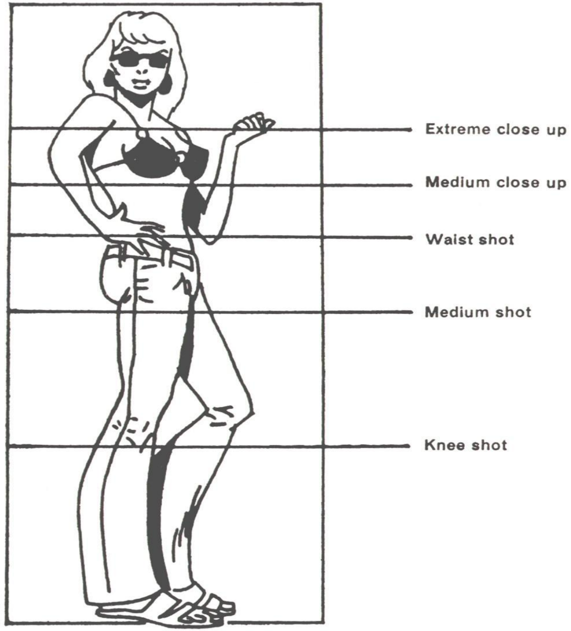
- Limited interactions
- Non-cinematographic paths

How to Frame with a Drone?

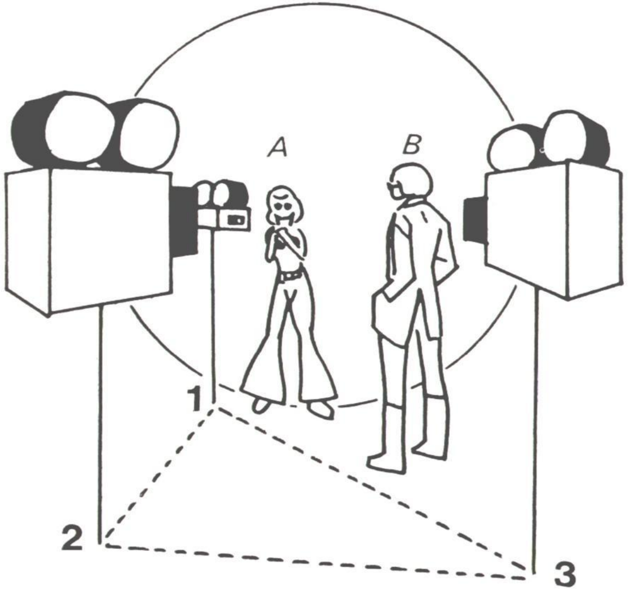
From Cinematographic Properties to Viewpoints

Cinematography

An empirical language defined by cinematographers:



Shot size



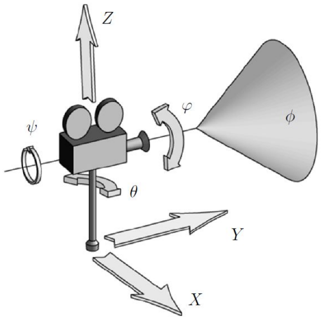
Shot angles: 1+3 OTS, 2 Apex

Composition

From Properties to Viewpoints

From film language to camera viewpoints:

- A computationally complex problem addressed with optimization
- Each visual property is defined as a cost function on camera parameters (7 dofs)



- All visual properties are aggregated in a cost function

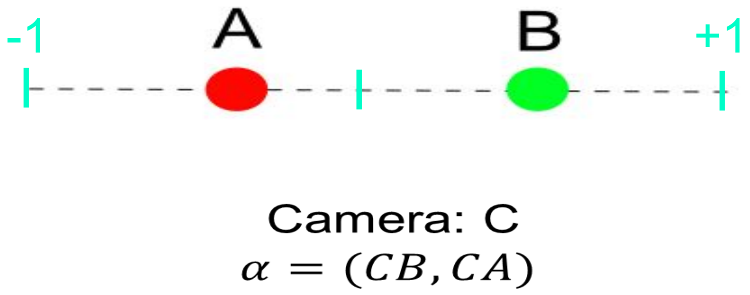
$$F(c) = \sum_i f_i(c)$$

- A non-linear solver searches for best viewpoints
- Computationally expensive (stochastic solvers [Ranon15])

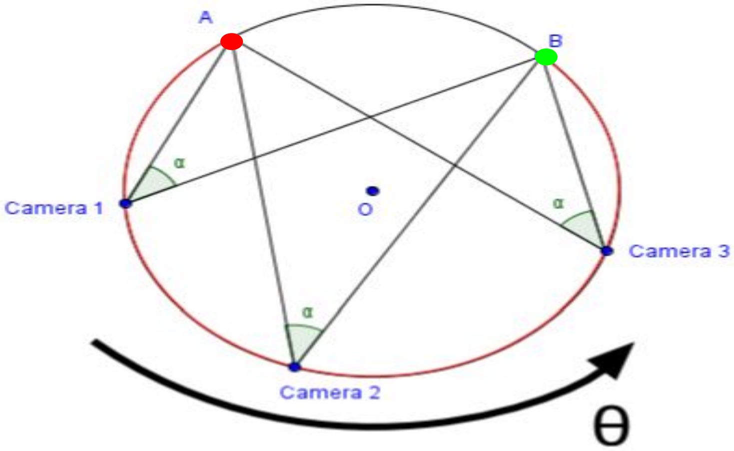
=> we propose a novel parametric space for camera composition problems

The Toric Space [Lino2015]

Desired on-screen Composition (1D)



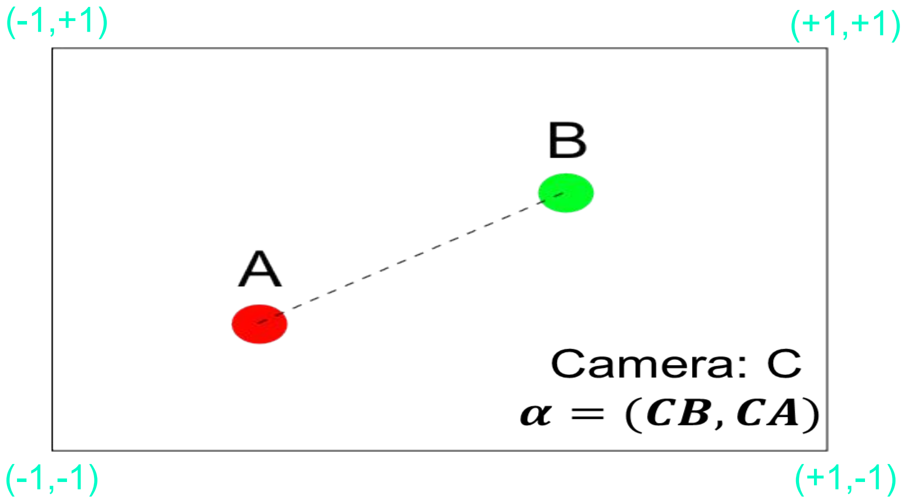
Solution = 1D parametric manifold (θ)



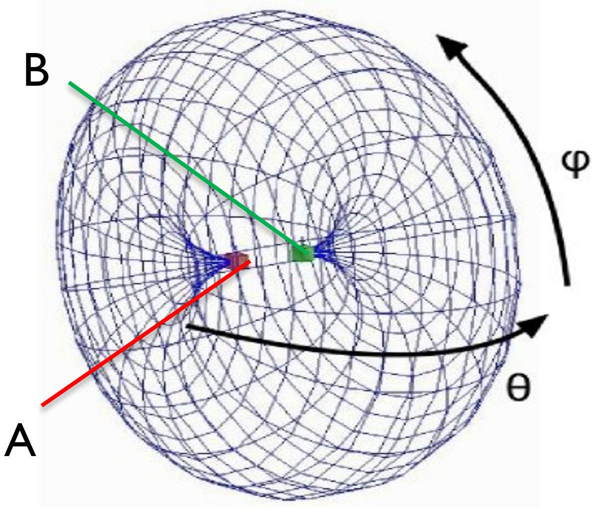
Any configuration c(θ) satisfies the 1D composition

Composition: 3D environment

Desired on-screen
Composition
(2D)



Solution = 2D manifold
surface (θ,φ)
(subset of a spindle torus)



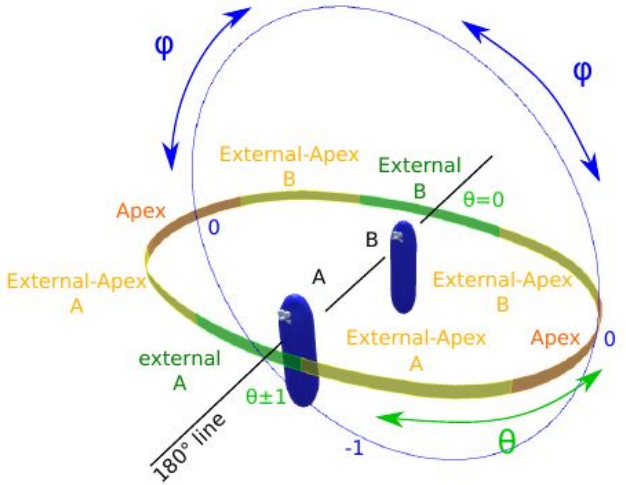
Any configuration c(θ,ϕ) satisfies the 2D composition

Manipulations in the Toric Space

The Toric space

Enables an algebraic expression of cinematographic properties:

- Screen composition
- Horizontal and vertical angles (theta, phi)
- Distance to targets



Cameras can therefore be controlled in an algebraic way

=> casts a 7D camera problem into a 3D camera problem

The Drone Toric space

- ➢Adapt the Toric space to drones
  - ➢ To ensure actors' safety (targets A and B)

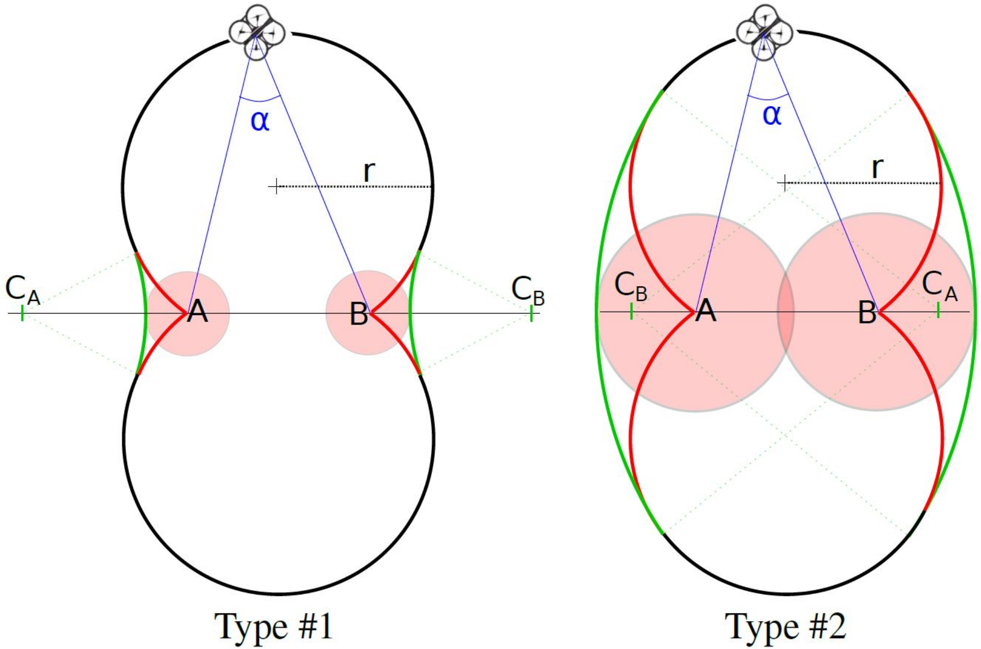
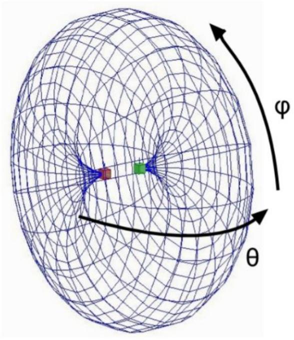
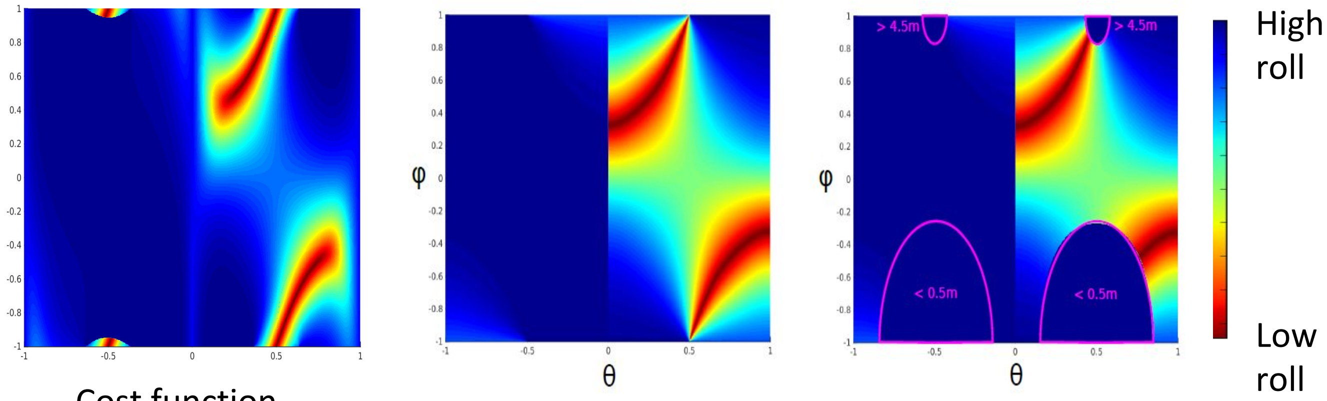


Image-space Interaction

- ➢Additional interactions
- ➢Better optimization scheme
  - ➢Use the roll as cost function
  - ➢Account for obstacles





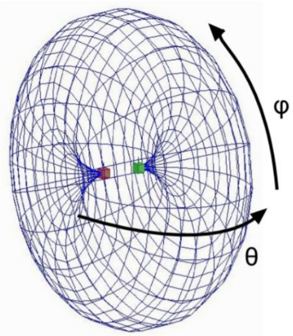
Cost function [Lino et al. 2015]

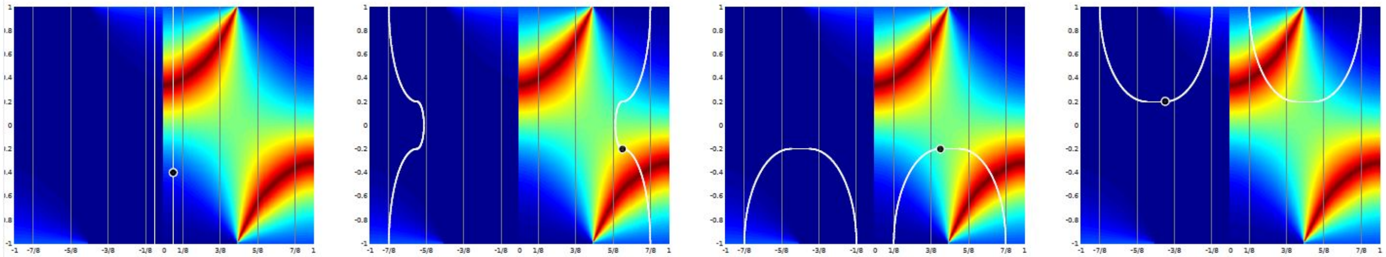
Our cost function

With Obstacles

Image-space Interaction

- ➢Additional interactions
- ➢Better optimization scheme
  - ➢Use the roll as cost function
  - ➢Account for the obstacles
  - ➢Adapt the search to the current position





(a) External A

(b) External-Apex B

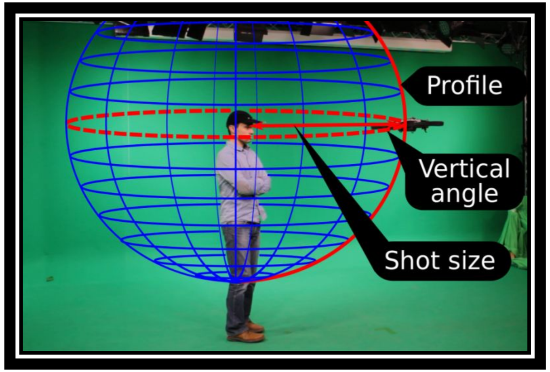
(c) Apex (from below)

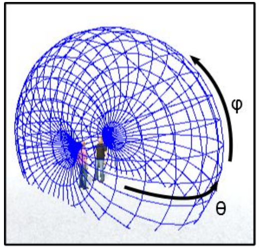
(d) Apex (from above) crossing 180° line

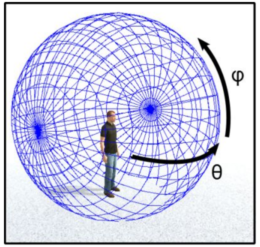
How to Move a Drone In a Cinematographic Way?

Creating Cinematographic Trajectories

User input

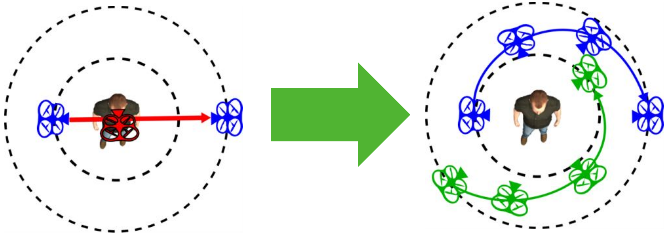




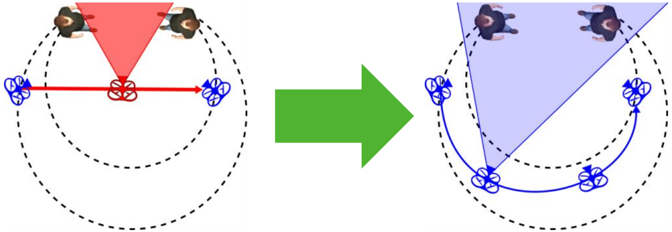


Interpolation in the Toric Space

1 actor:

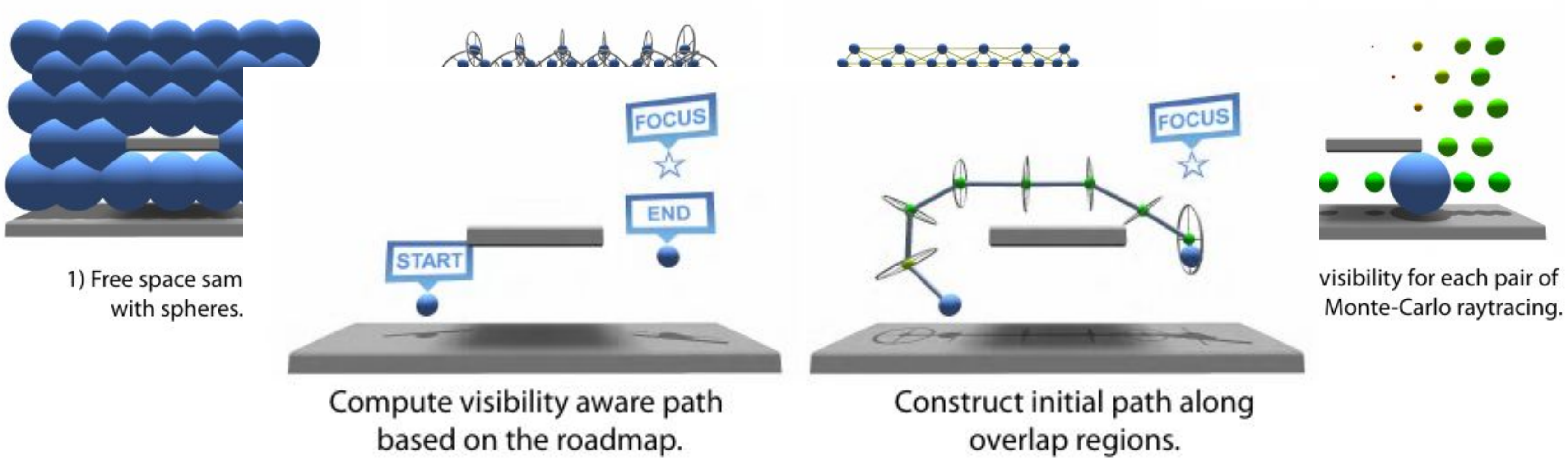


2 actors:



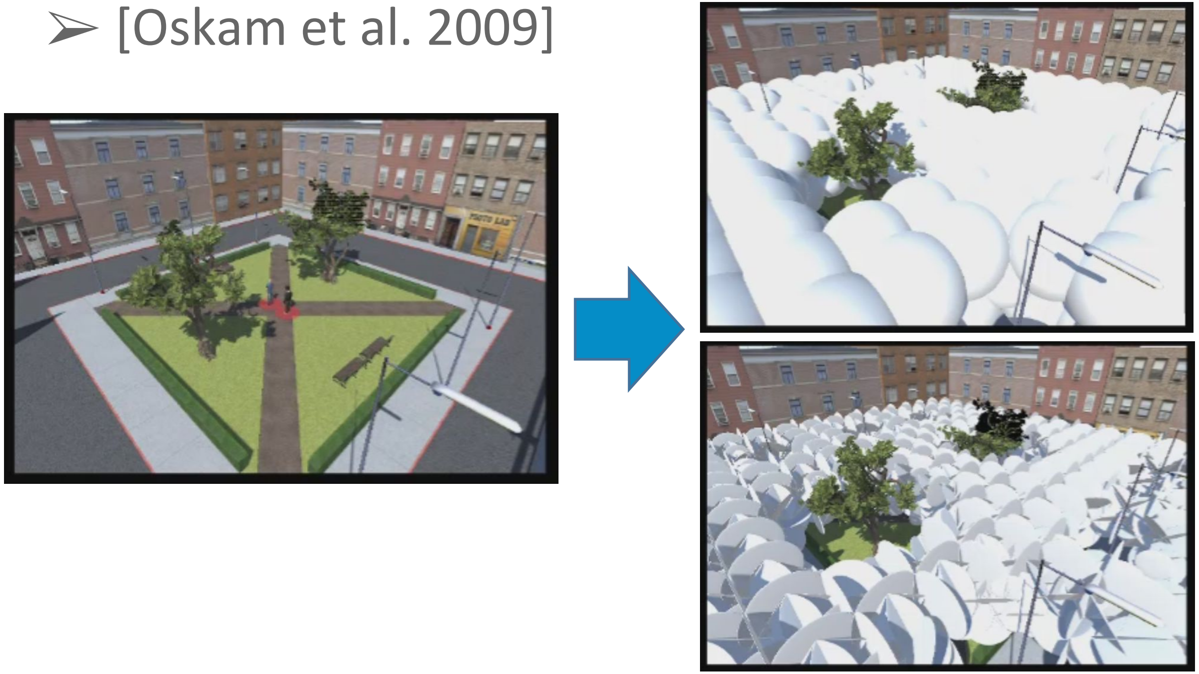
Planning cinematographic paths

- ➢Collision avoidance mandatory
- ➢Visibility aware roadmap and A* path planning
  - ➢ [Oskam et al. 2009]



Planning cinematographic paths

- ➢Collision avoidance mandatory
- ➢Visibility aware roadmap and A* path planning
  - ➢ [Oskam et al. 2009]



Planning cinematographic paths

➢Planning the path in the space of visual properties

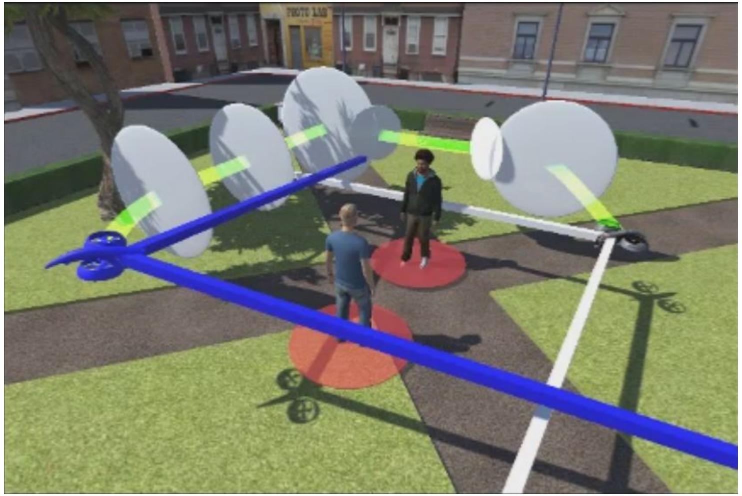
- New distance metric based on the toric space

$$D_s^2(n_i, n_j) = d(\alpha_i, \alpha_j)^2 + d(\varphi_i, \varphi_j)^2 + d(\theta_i, \theta_j)^2$$

- Weighted with visibility information

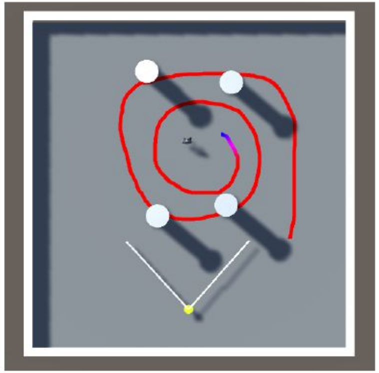
Initial Shot

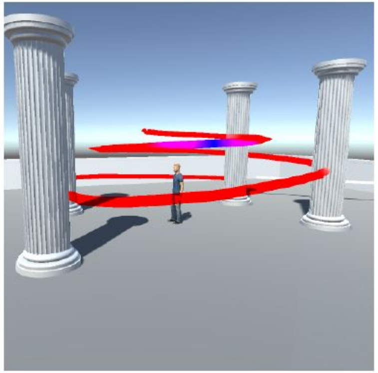
Desired Shot

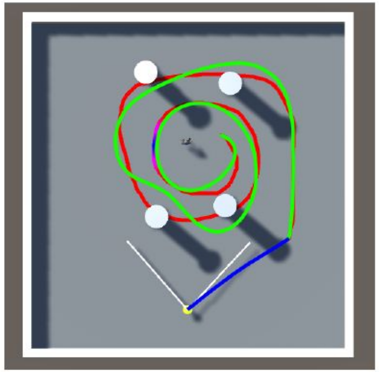


Sketching trajectories

- Collision avoidance mandatory
  - Use the roadmap
- Modified A* algorithm to allow loops
- C4 optimization

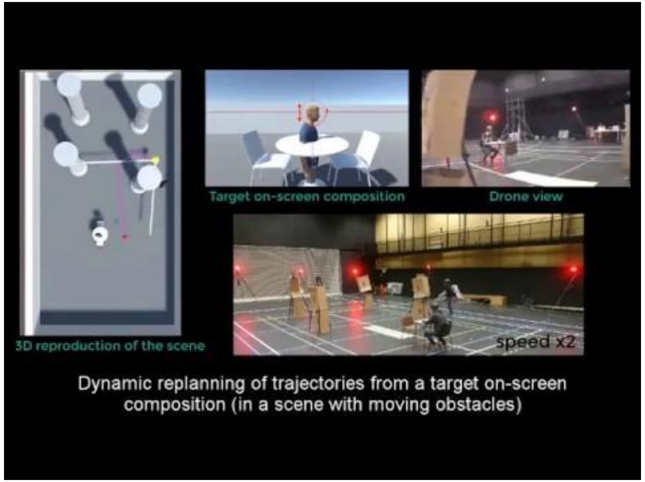






➢RESULTS

- ➢ Indoor tracking using optoelectronic system (VICON)
- ➢ With Parrot ARDrones
- ➢ With Parrot Bebop2



How to Handle Multiple Drones?

Orchestration of drones

Handling multiple drones?

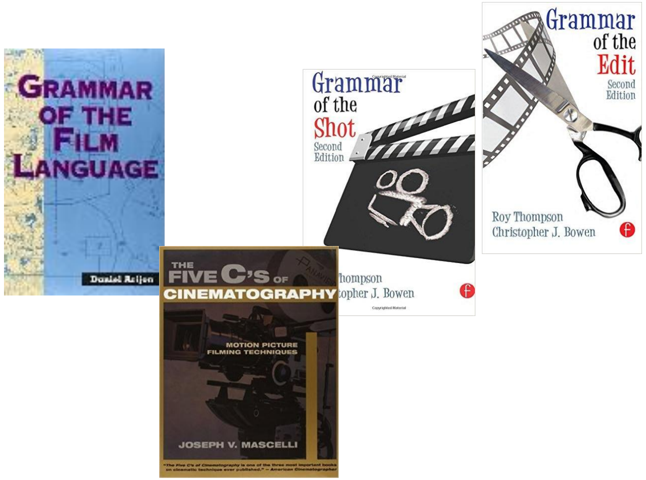
➢ How to use our technology to synchronize multiple drones?
  - Every drone covers a different angle of the scene
  - Drones offer complementary views (for further editing)
  - Drones react to changes and avoid conflicts

➢ Our approach (a TV editor metaphor)
  - A master drone (interactively controlled by the user)
  - Slave drones offering non-conflicting views that satisfy "continuity editing"

Editing

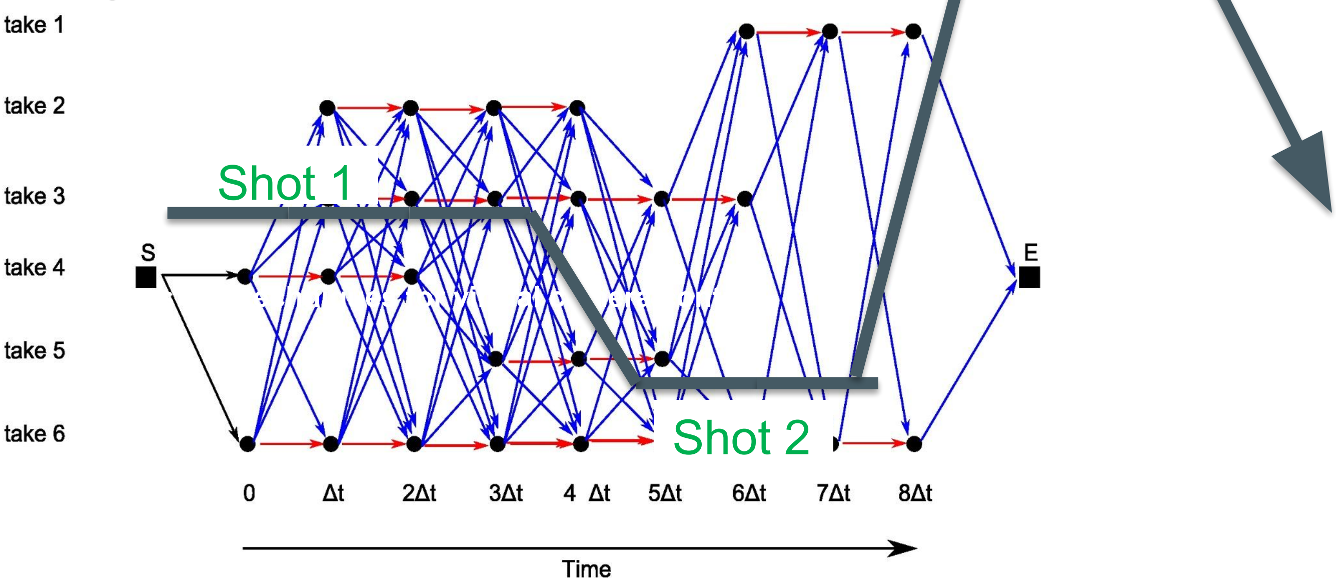
- Editing is the art of cutting between view angles
  - Choosing when to cut
  - Choosing where to cut to
  - With which type of transition

- **Editing forms a visual « grammar »**
  - Frames are letters, shots are the words
  - Scenes are sentences, films are stories
- **Continuity-editing**
  - *Grammar of the Film Language* [Arijon 76]
  - *Grammar of the Shots* [Thomson 98]
  - *Grammar of the Edit* [Thomson 93]
  - *The five C's of Cinematography* [Mascelli 98]



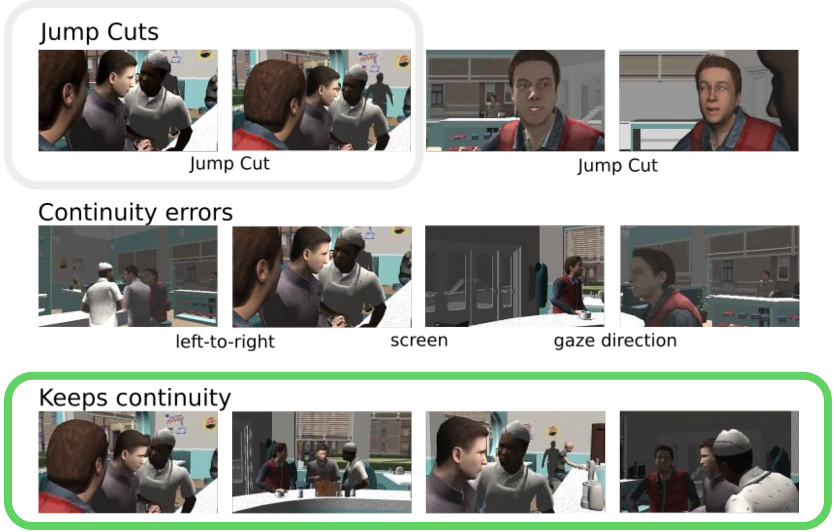
A general approach: The *"editing graph"* [Galvane etal 2015]

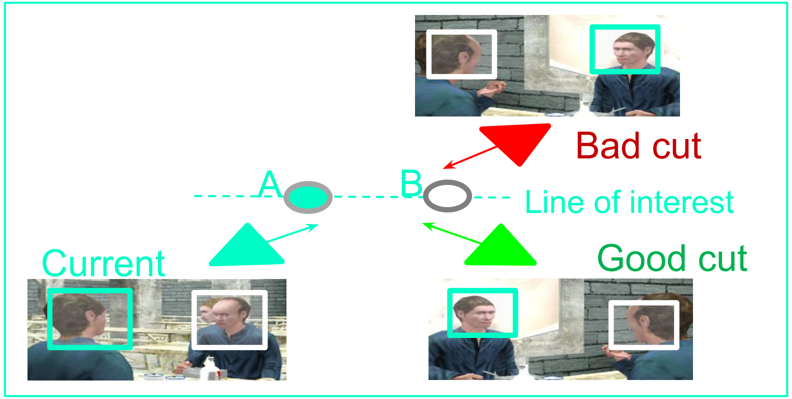
- Automated editing can be viewed as planning a path through an oriented graph
  - Node c_i^t: use camera (take) i at time t
  - Arc $c_i^t \rightarrow c_j^{t+1}$: do not cut (j $= i$) / cut to another camera ($j \neq i$)



« continuity editing»

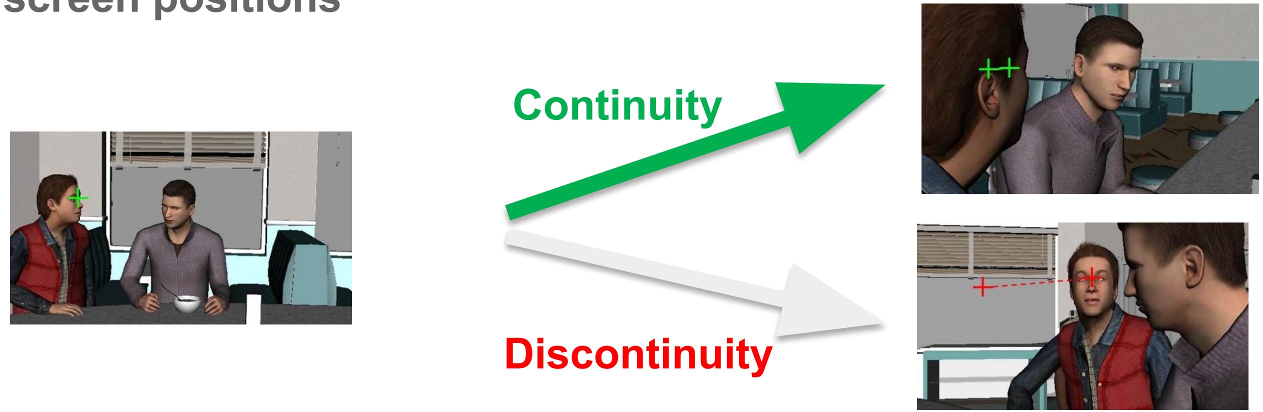
- Controls how storyline actions are perceived all together
  - **Make link** between pieces of information
  - **Guide** viewers' attention (visual cues)
- Controls how a given action is perceived as continuous in time
  - **Do not break continuity** (coherency)





Cut quality: absolute screen positions

- Penalize cuts breaking continuity
  - On **absolute screen positions**

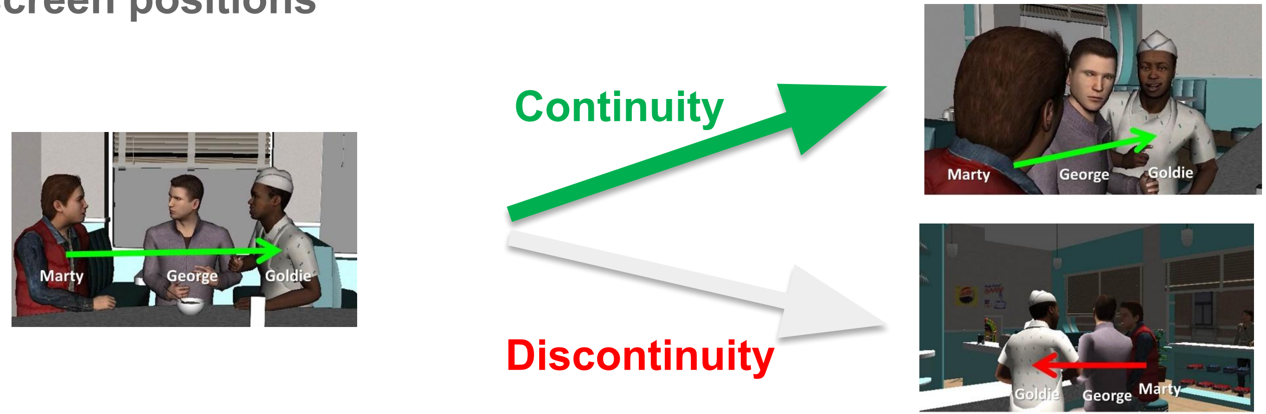


- Cost function:

$$P^{T}_{Screen}(c^{t}_{j-1}, c^{t}_{j}) = \sum_{i} \phi_{S}[Pos(T^{i}, c^{t}_{j-1}) - Pos(T^{i}, c^{t}_{j})]$$

Cut quality: relative screen positions

- Penalize cuts breaking continuity
  - On **relative screen positions**

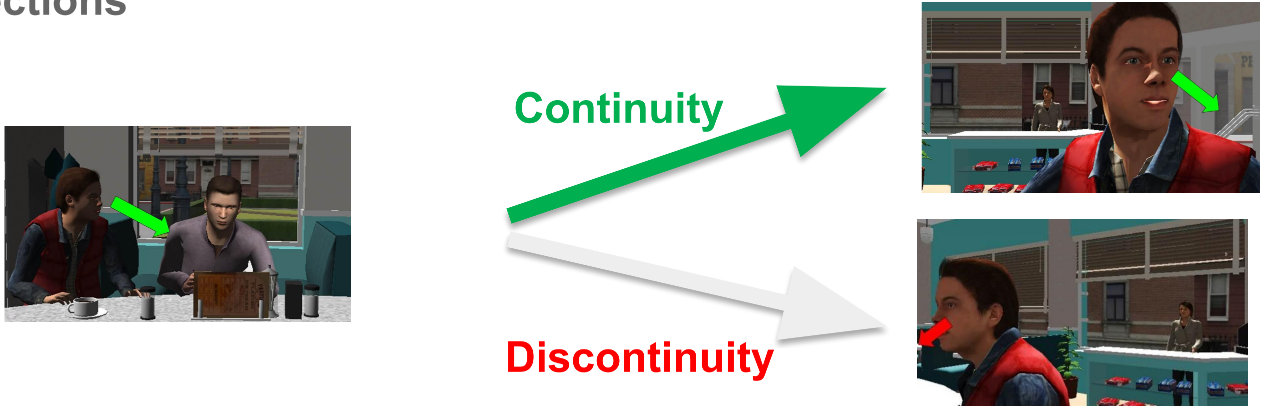


- Cost function:

$$P^{T}_{Order}(c^{t}_{j-1}, c^{t}_{j}) = \sum_{i,j} \phi_{O}[Order(T^{i}, T^{j}, c^{t}_{j-1}), Order(T^{i}, T^{j}, c^{t}_{j})]$$

Cut quality: gaze continuity

- Penalize cuts breaking continuity
  - On **gaze directions**

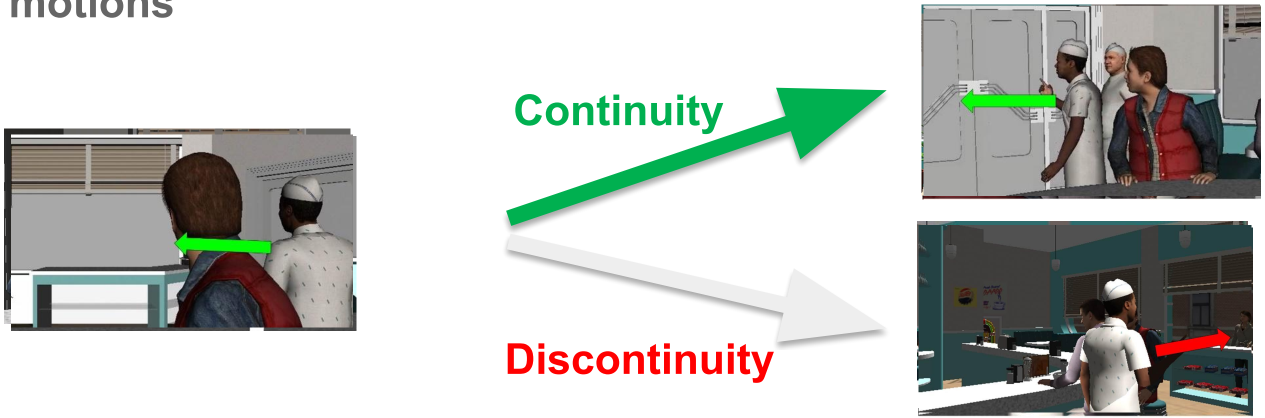


- Cost function:

$$P^{T}_{Gaze}\left(c^{t}_{j-1}, c^{t}_{j}\right) = \sum_{i} \phi_G\left[Gaze\left(T^{i}, c^{t}_{j-1}\right), Gaze\left(T^{i}, c^{t}_{j}\right)\right]$$

Cut quality: motion continuity

- Penalize cuts breaking continuity
  - On **apparent motions**



- Cost function:

$$P^{T}_{Motion}(c^{t}_{j-1}, c^{t}_{j}) = \sum_{i} \phi_M\left[Motion(T^{i}, c^{t}_{j-1}), Motion(T^{i}, c^{t}_{j})\right]$$

Avoid *“jump cuts”*

- Penalize cuts that do not look like cuts (visually, not enough **change in size or view angle**)

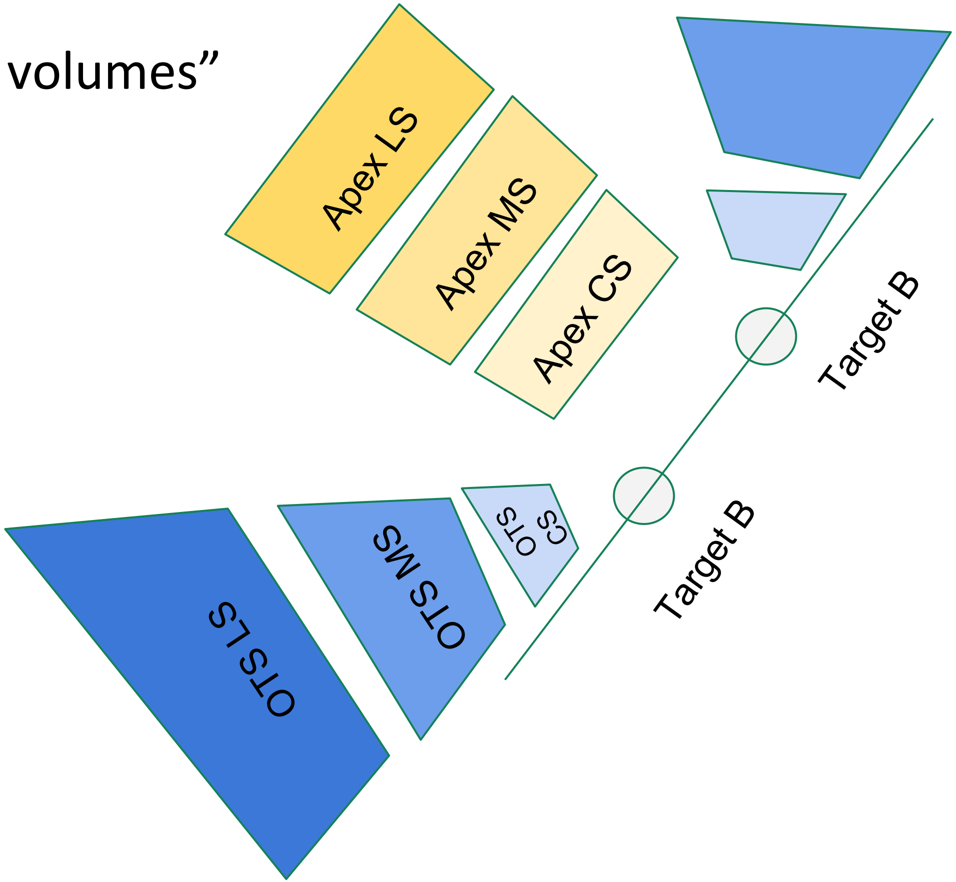
Sufficient change in size

Sufficient change in view angle

« Jump cut »

Handling multiple drones

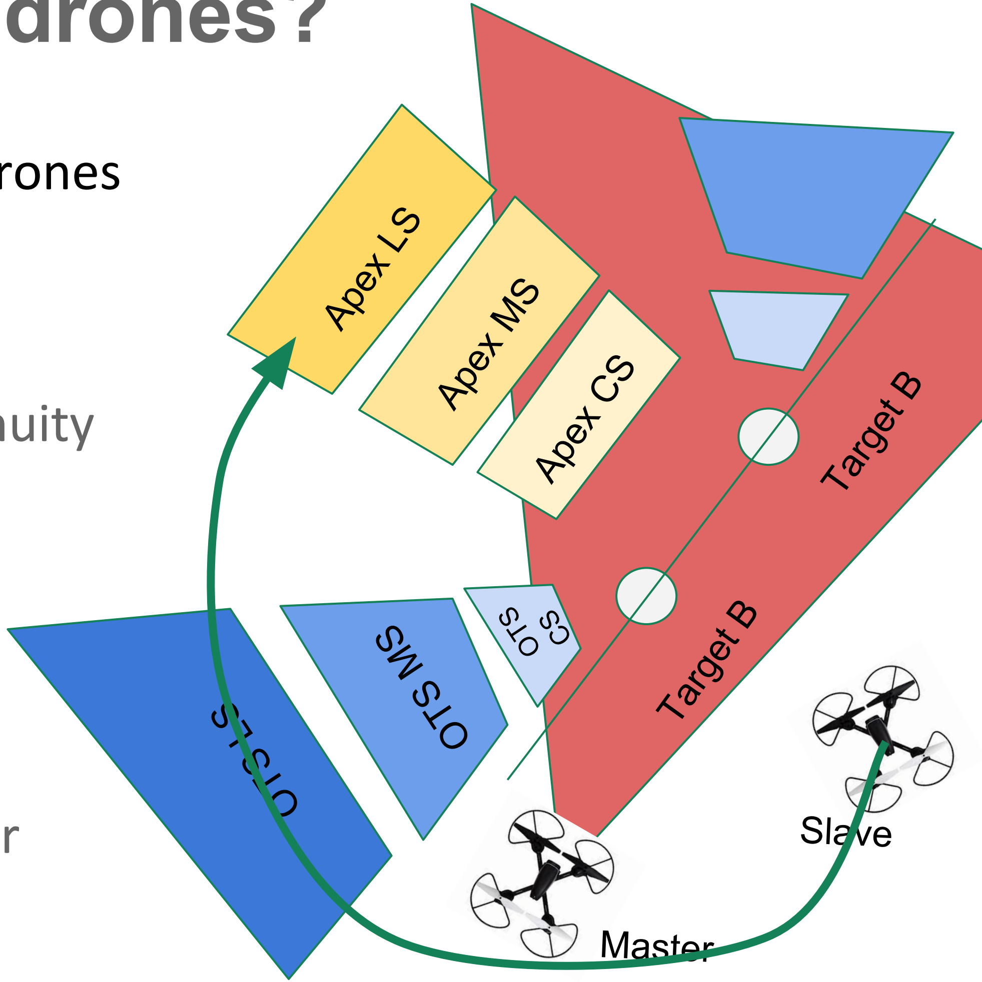
- ➢Define tagged regions " 18 semantic volumes"
    - In Toric space coordinates
    - Relative to the targets



Handling multiple drones?

➢ Remove conflicting areas for slave drones
  ▪ Remove areas with visibility conflicts
  ▪ Remove areas that fail "continuity editing"

➢ Select a possible volume
  ▪ Shortest path to a volume
  ▪ That avoids visibility by Master



Searching for non-conflicting assignments

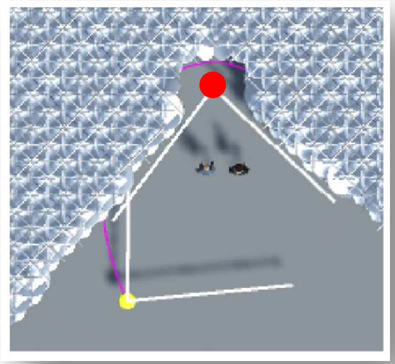
➢ Use a min-conflict solving process

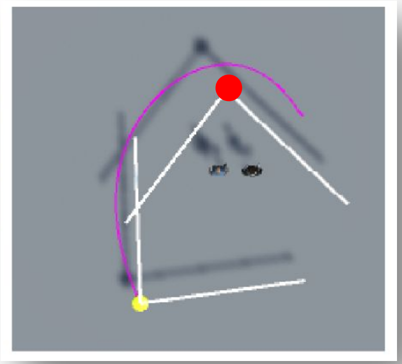
- Find the slave drone with the minimum number of conflicts
- Search a semantic assignment for that drone
- If failure, search for an assignment for the two slaves drones with minimum number of conflicts

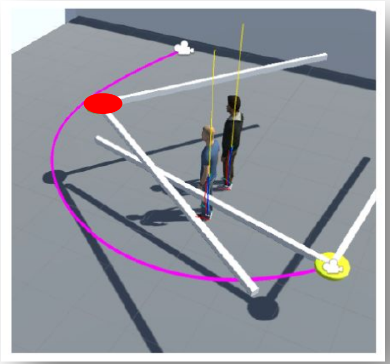
➢ Practical complexity is low (even with 3 slaves)

- 4k combinations
- Above 4 drones, the environment gets cluttered

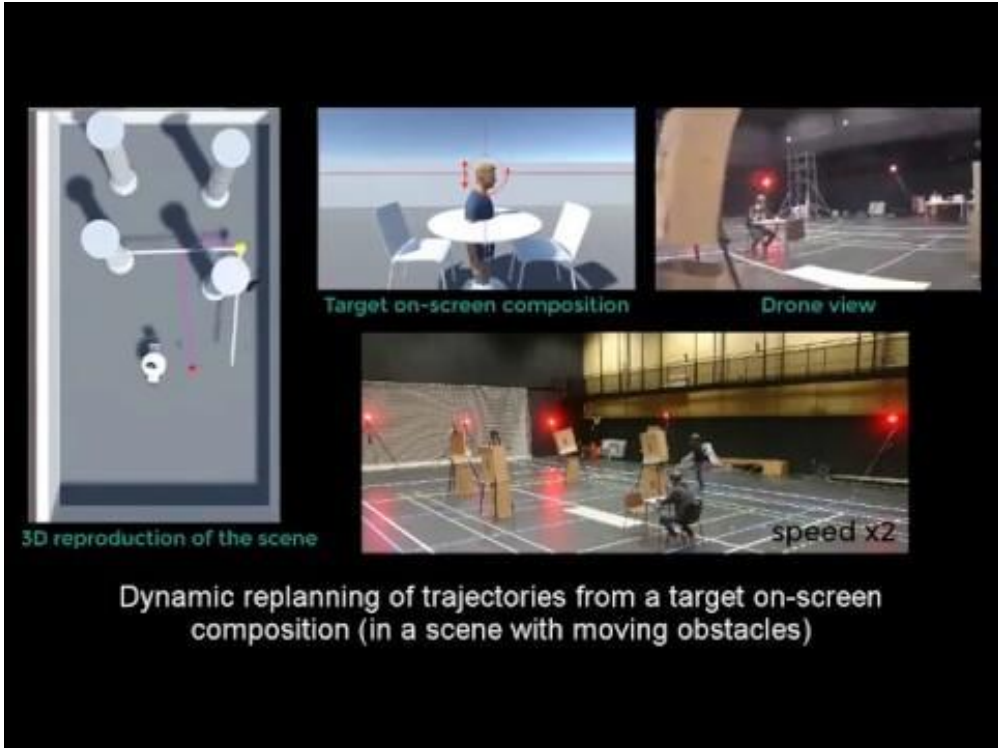
- ➢Handling planning through the roadmap
  - ➢ Frustum culling in the roadmap







➢RESULTS


3D reproduction of the scene
Target on-screen composition
Drone view
speed x2
Dynamic replanning of trajectories from a target on-screen composition (in a scene with moving obstacles)

➢ PreciseTarget localisation remains a problem
➢A possible immediate use case using GPS (from multidrone.net)

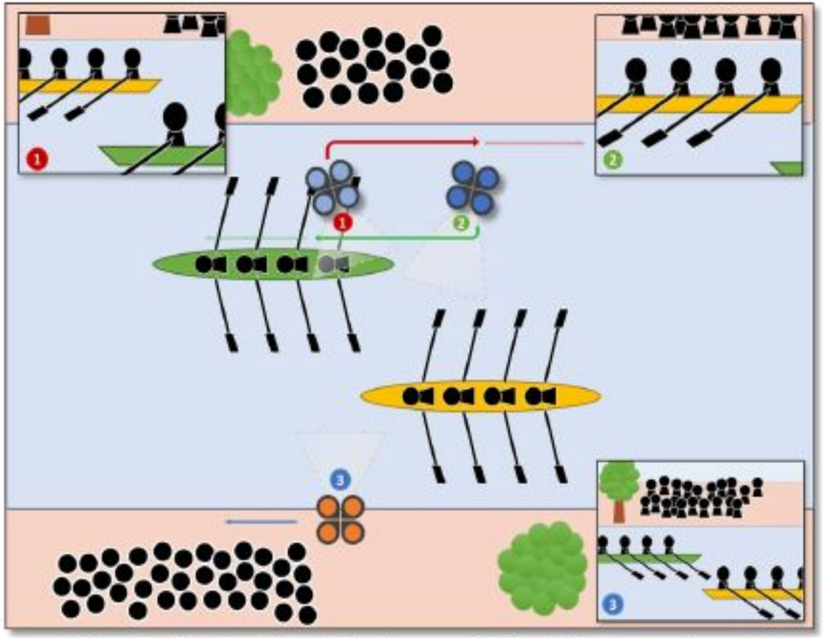


Figure 10: Overview of Scenario 2 - drones reacting to each other on approach

Discussion

Issues?

- Precise localisation (indoor / outdoor)
  - Using Ultra Wide Band technology?
  - Using robotics SLAM technology?

  => Yet, some outdoor scenario remain possible!

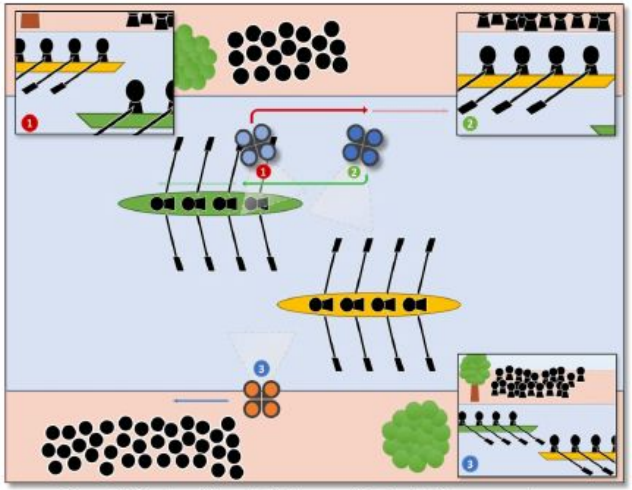


Figure 10: Overview of Scenario 2 - drones reacting to each other on approach

- Precise 3D representations for path planning and viewpoint quality
  - Use 3D reconstructed maps (photogrammetry)

But what’s next?

- Towards data-driven cinematography for drones (taking inspiration from real footage)
  - Extracting framing/motion features from sequences

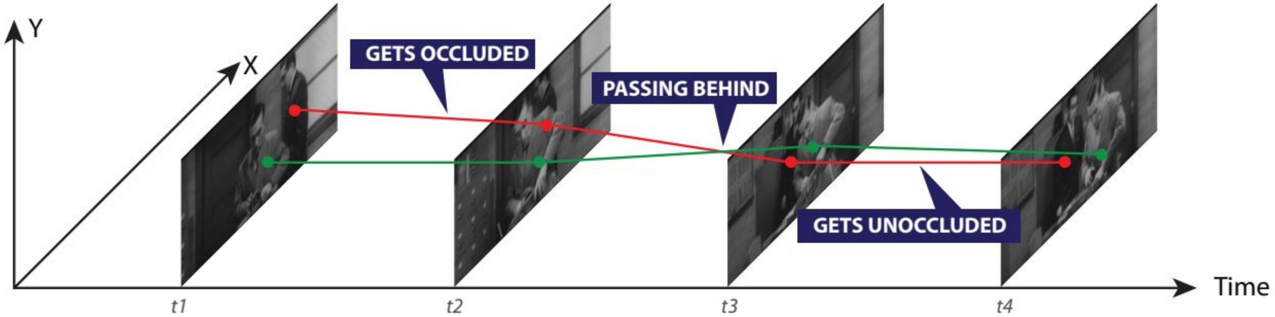
Using DLIB tracker + OpenPose

But what's next [next 10 years]

- Automated shooting of documentaries :
  - Mini-drones framing and stabilizing the image
  - Choosing the right angles wrt background and light
  - Mini-drones lighting the scene
  - Analyzing motions and facial expressions to cut/reframe (ie indirectly control the drone through postures)

Bridging intention and techniques

- Spatio-temporal reasoning on footage
- Opens many possibilities to learn deeper relations between the content/events and the technique (camera/light/etc)



Cognitive & Emotional Cinematography

- Storytelling *"The art of bringing an audience from a given cognitive and emotional state to an intended state, through a set of cognitive and emotions changes" (hence it's a state planning problem!)*