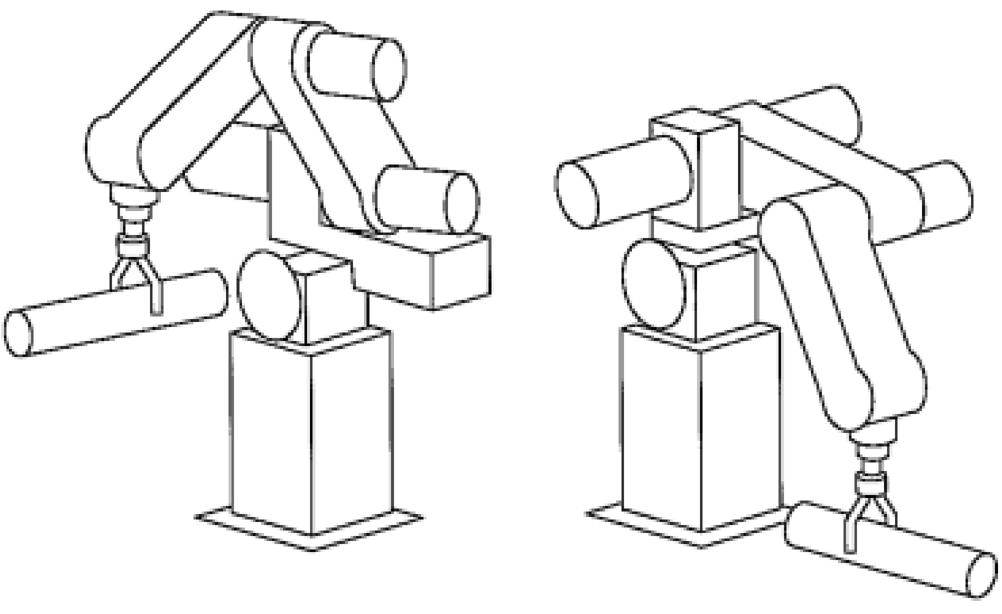
CHAPTER 7 Trajectory generation

7.1 INTRODUCTION

trajectory: a time history of position, velocity and acceleration for each degree of freedom. e.g., $x(t), y(t), \alpha(t), \dot{x}(t), \theta(t)$

path: a geometric curve in space, no time history involved.

It is assumed that the path returned by the path planner can be scaled to create a feasible trajectory.



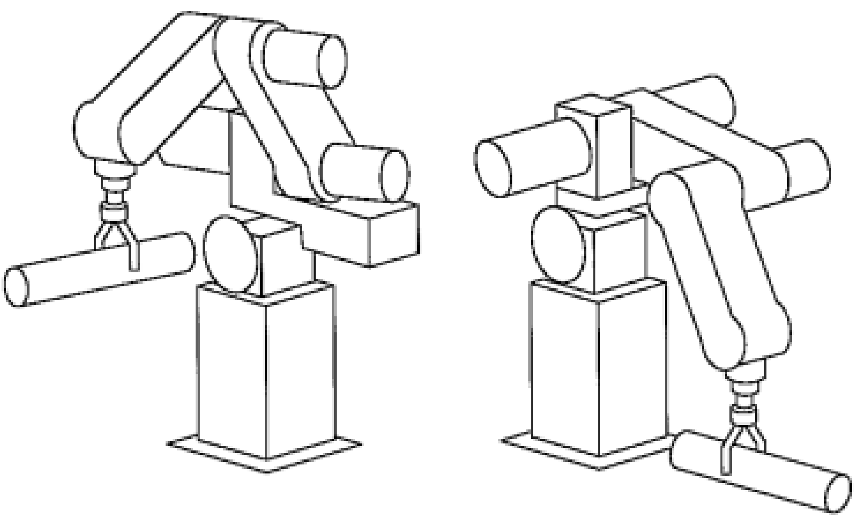
7.2 GENERAL CONSIDERATIONS IN PATH DESCRIPTION AND GENERATION

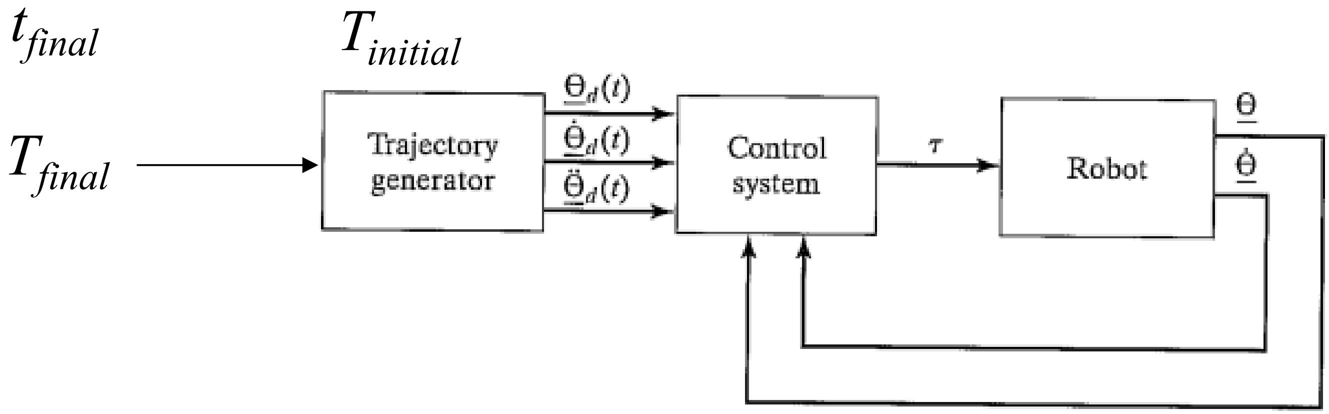
Fig. to move the tool frame from $\{T_{initial}\}$ to $\{T_{final}\}$. 2 cases:

1. the initial and final points
2. via points (intermediate points between the initial and final points)

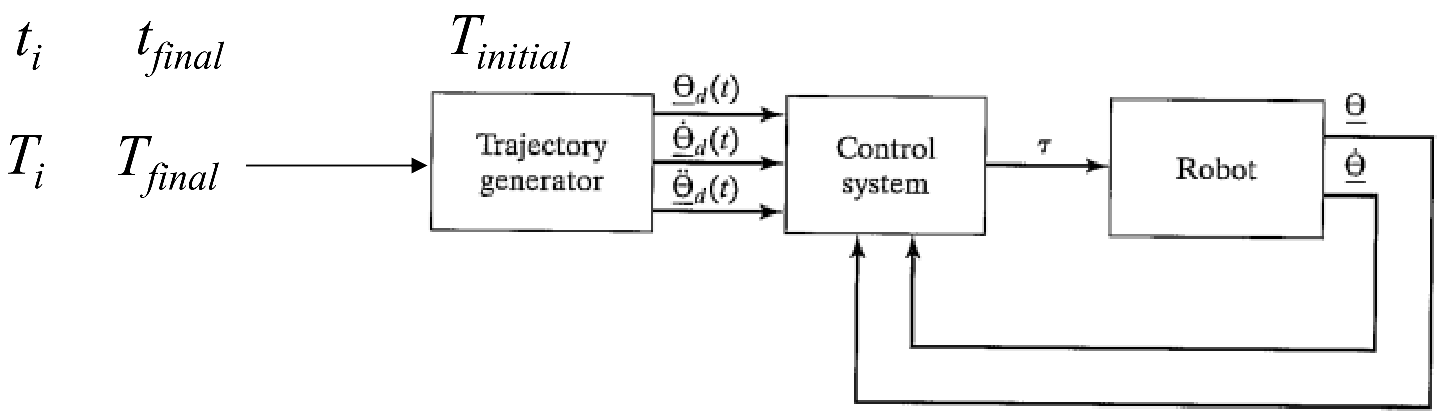
requirement:

the motion is smooth, i.e. smooth function that is continuous and has continuous first or even second derivative. Because rough jerky motions tend to cause increased wear and cause vibrations by exciting resonances in the manipulator.





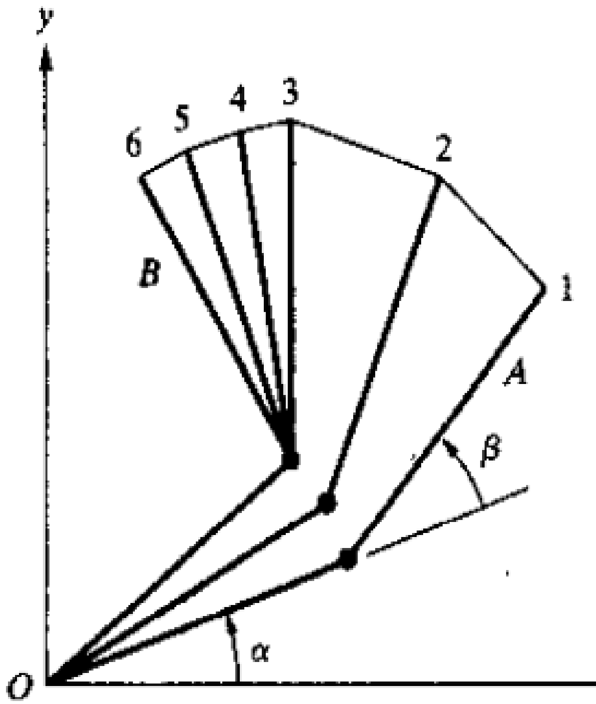
only initial and final points



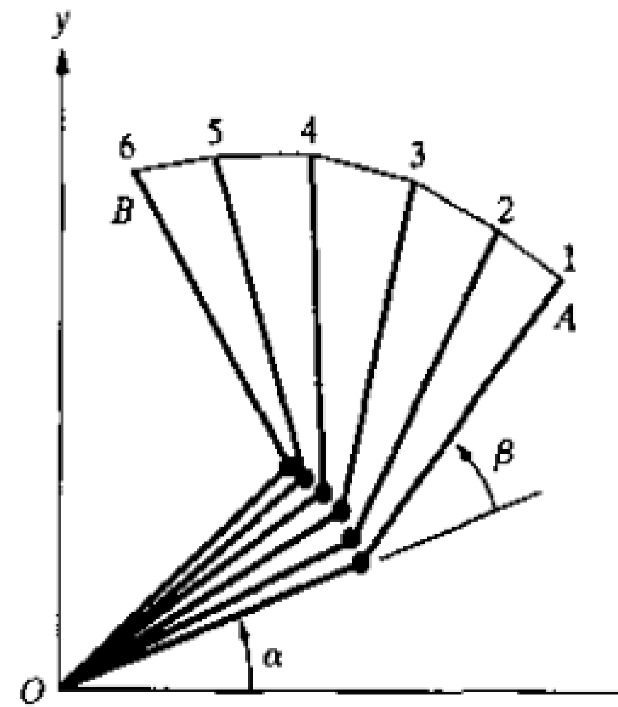
initial point, final point and via points (all of these are named path points)

path-update rate: trajectories are computed on digital computers, so the trajectory points are computed at a certain rate, called path-update rate. 60Hz~2000Hz.

intuitive illustration of trajectory generation
--- joint-space scheme



	α	β	
0s	20	30	A
1s	30	40	
2s	40	50	
3s	40	60	
4s	40	70	
5s	40	80	B

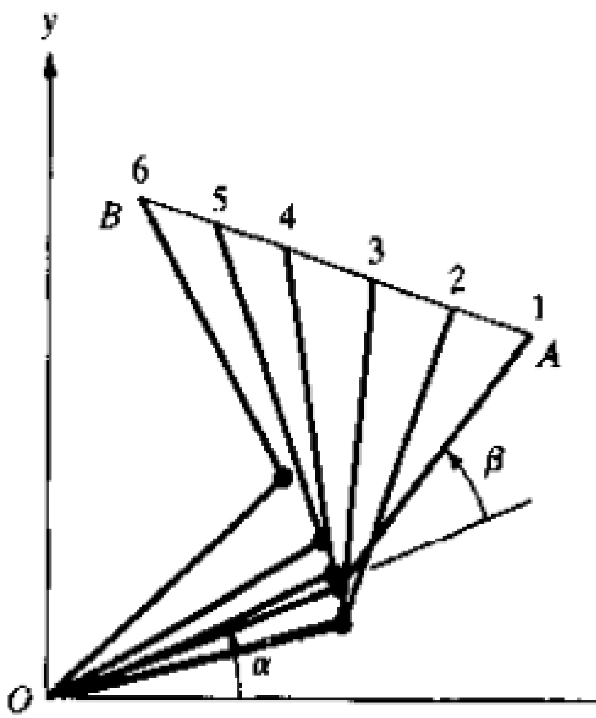


	α	β	
0s	20	30	A
1s	24	40	
2s	28	50	
3s	32	60	
4s	36	70	
5s	40	80	B

- each joint moves at its max velocity 10°/s
- joint 1 stops first
- distances of the end are not uniform

- joints start and stop at the same time
- velocities are kept during their motions
- the segments are similar

intuitive illustration of trajectory generation
--- Cartesian-space scheme

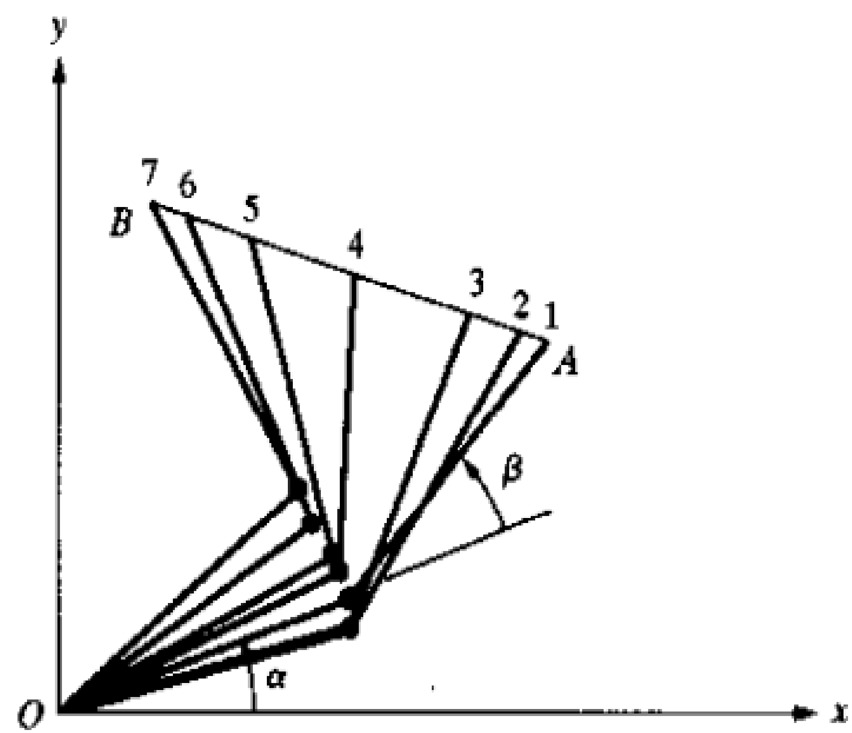


	α	β	
0s	20	30	A
1s	14	55	
2s	16	69	
3s	21	77	
4s	29	81	
5s	40	80	B

assume acceleration could be infinite

method: equal interpolation

p.s. joint angles do not change evenly.



assume acceleration is limited

method: unequal interpolation

path distances of starting and ending segments are shorter

7.3 JOINT-SPACE SCHEME

--- Path shapes (in space and in time) are described in terms of functions of joint angles.

Given: path points

Process:

1. convert path points into a set of desired joint angles by inverse kinematics.
2. find a smooth function for each of the *n* joints that pass through the via points.
3. the time required for each segment is the same for each joint so that all joints will reach the path points at the same time.

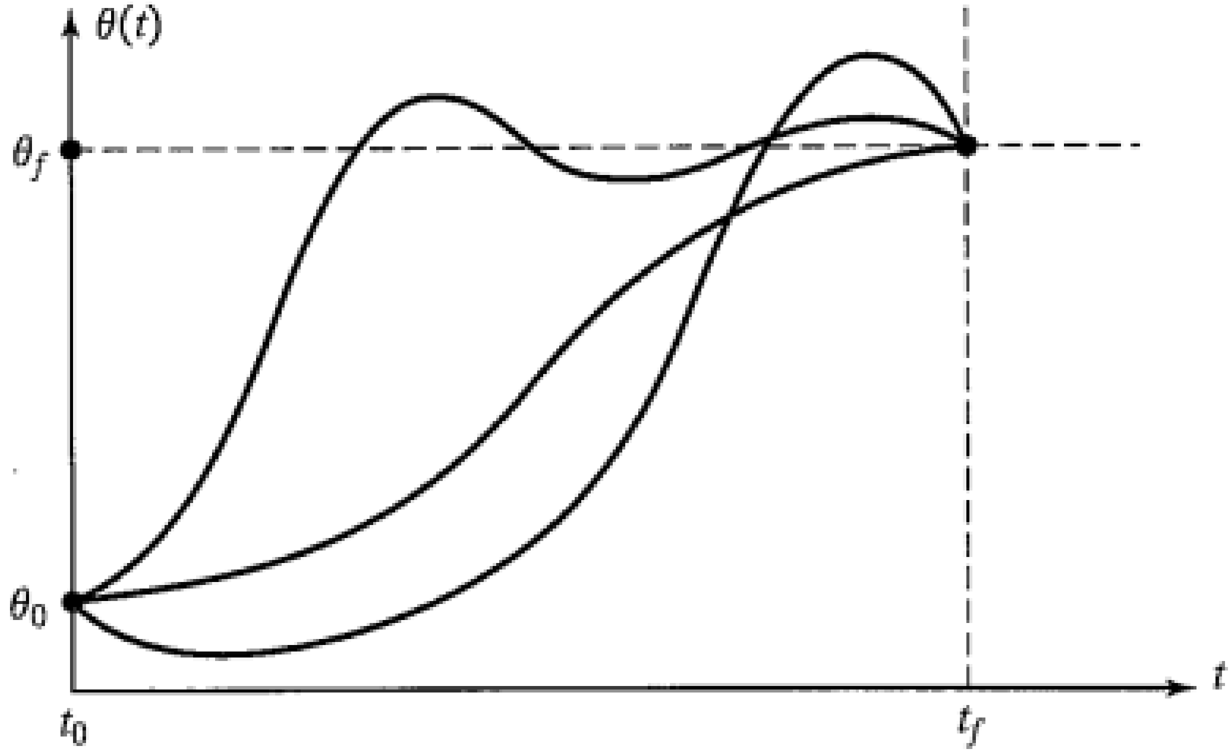
Methods:

- Cubic polynomials
- Higher-order polynomials
- Linear function with parabolic blends
- others

Cubic polynomials

only initial: t_0, θ_0;
final point: t_f, θ_f .

if no constraint,
many smooth functions



constraints:

initial and final angular position:

$$\theta(0) = \theta_0,$$
$$\theta(t_f) = \theta_f.$$

initial and final angular velocity are zero:

$$\dot{\theta}(0) = 0,$$
$$\dot{\theta}(t_f) = 0.$$

four constraints can be satisfied by a polynomial of at least third degree (a cubic polynomial has four coefficients)

A cubic has the form

$$\theta(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3,$$

4 desired constraints:

$$\theta(0) = \theta_0,$$
$$\theta(t_f) = \theta_f.$$

$$\dot{\theta}(0) = 0,$$
$$\dot{\theta}(t_f) = 0.$$

joint velocity along this path

$$\dot{\theta}(t) = a_1 + 2a_2 t + 3a_3 t^2,$$

substitute the constraints into the cubic and its velocity, we get 4 equations:

$$\theta_0 = a_0,$$
$$\theta_f = a_0 + a_1 t_f + a_2 t_f^2 + a_3 t_f^3,$$
$$0 = a_1,$$
$$0 = a_1 + 2a_2 t_f + 3a_3 t_f^2.$$

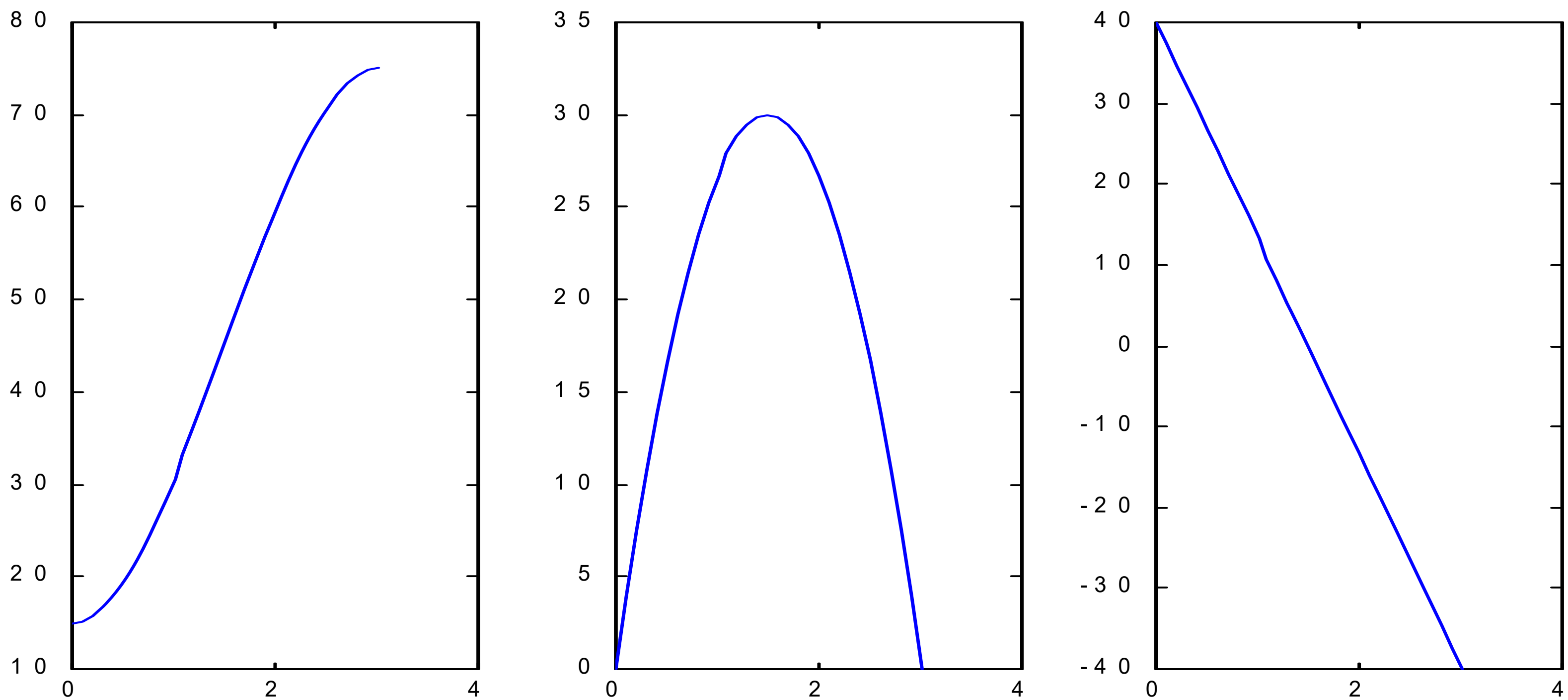
solve them, we get

$$a_0 = \theta_0,$$
$$a_1 = 0,$$
$$a_2 = \frac{3}{t_f^2}(\theta_f - \theta_0),$$
$$a_3 = -\frac{2}{t_f^3}(\theta_f - \theta_0).$$

EXAMPLE 7.1 A single-link robot with a rotary joint is motionless at $\theta_0 = 15°$. It is desired to move the joint in a smooth manner to $\theta_f = 75°$ in 3s. Find the coefficients of a cubic that accomplishes this motion and brings the manipulator to rest at the goal. Plot the position, velocity, and acceleration of the joint as a function of time.

Solution: plugging into the above equation, a_0=15.0, a_1=0.0, a_2=20.0, a_3= -4.44

$$\theta(t) = 15.0 + 20.0t^2 - 4.44t^3 \quad \dot{\theta}(t) = 40.0t - 13.33t^2 \quad \ddot{\theta}(t) = 40.0 - 26.66t$$



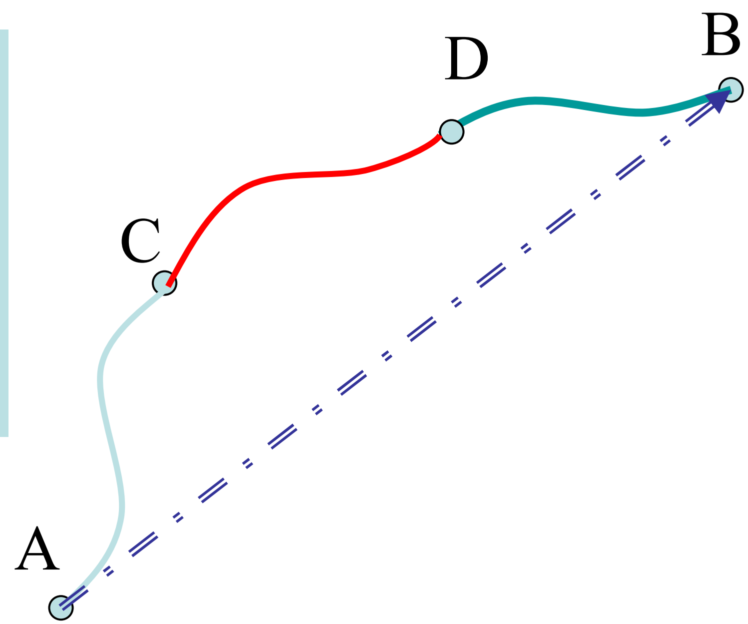
position

velocity

acceleration

Cubic polynomials for a path with via point $\theta(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3,$

initial point + via points + final point
•rest at via points: use the above method repeatedly (easy)
•pass through points without stopping: generalize the above method (see below)



consider 2 via points: C(t_0, θ_0); D(t_f, θ_f)

constraints:

•angular position constraints: $\theta(t_0) = \theta_0$ $\theta(t_f) = \theta_f$

•angular velocity constrains: $\dot{\theta}(t_0) = \dot{\theta}_0$ $\dot{\theta}(t_f) = \dot{\theta}_f$

the 4 equations become:

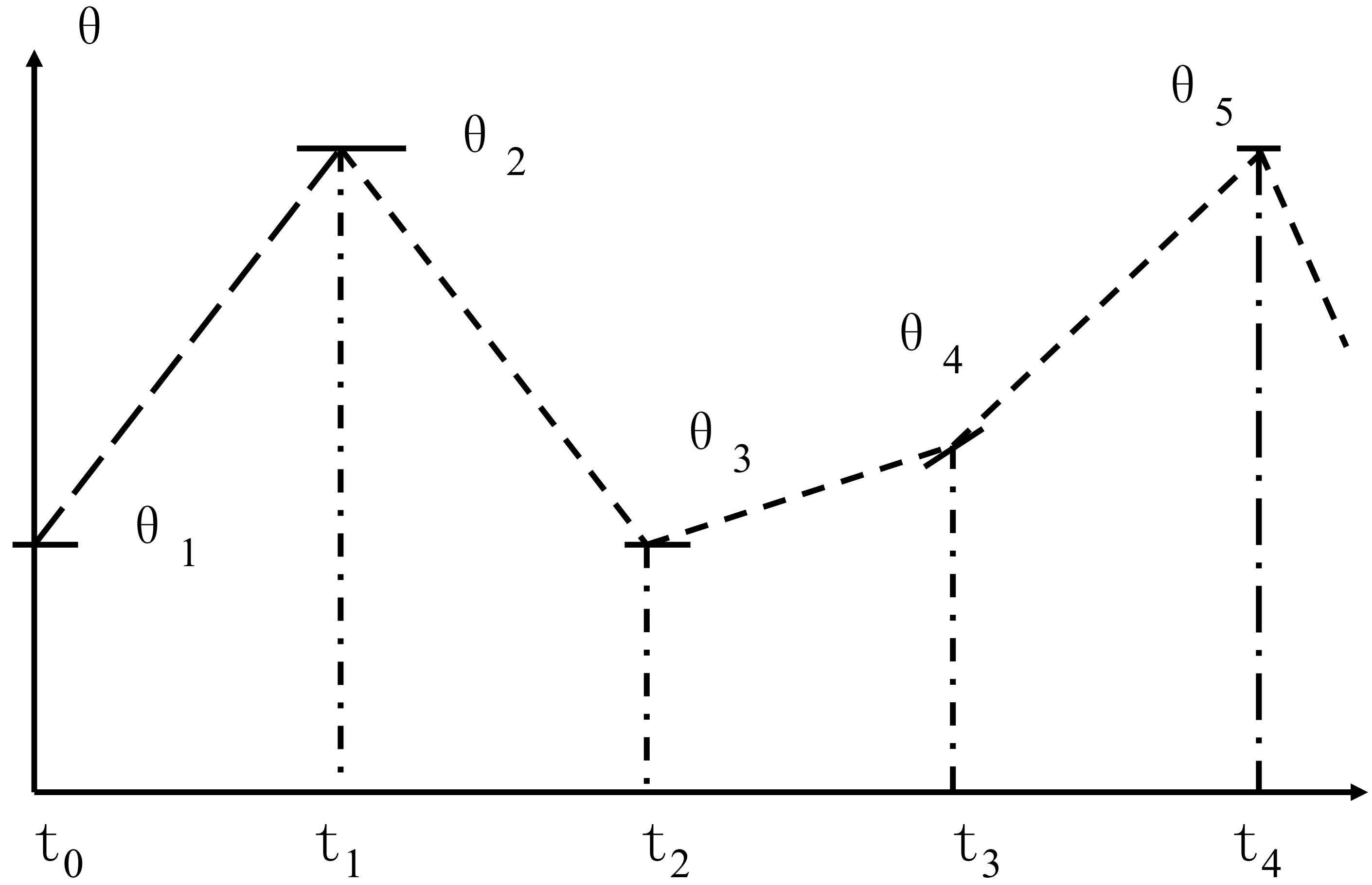
$$\theta_0 = a_0,$$
$$\theta_f = a_0 + a_1 t_f + a_2 t_f^2 + a_3 t_f^3,$$
$$\dot{\theta}_0 = a_1,$$
$$\dot{\theta}_f = a_1 + 2a_2 t_f + 3a_3 t_f^2.$$

solve them, we get

$$a_0 = \theta_0,$$
$$a_1 = \dot{\theta}_0,$$
$$a_2 = \frac{3}{t_f^2}(\theta_f - \theta_0) - \frac{2}{t_f}\dot{\theta}_0 - \frac{1}{t_f}\dot{\theta}_f,$$
$$a_3 = -\frac{2}{t_f^3}(\theta_f - \theta_0) + \frac{1}{t_f^2}(\dot{\theta}_f + \dot{\theta}_0).$$

several ways in which the desired velocity at the via points might be specified:

- The user specifies the desired velocity at each via point in terms of a Cartesian linear and angular velocity of the tool frame at that instant. (problems: 1. users may assign an arbitrary velocity at a singular point. 2. a burden for the user.)
- The system automatically chooses the velocities at the via points by applying a suitable heuristic in either Cartesian space or joint space. (example next slide)
- The system automatically chooses the velocities at the via points in such a way as to cause the acceleration at the via points to be continuous. (example 7.2)



The system automatically chooses reasonable intermediate velocities, using some kind of heuristic:

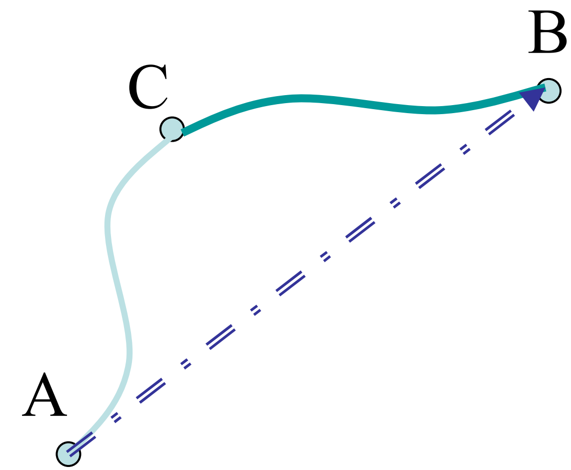
If the slope of these lines changes sign at the via point, choose zero velocity;

If the slope of these lines does not change sign at the via point, choose the average of the two slopes as the via velocity.

EXAMPLE 7.2 initial point θ_o, ($t = 0$), via point θ_v, ($t = t_{f1}$), final point θ_g, ($t = t_{f2}$), with continuous acceleration at the via point.

First cubic: $\theta(t) = a_{10} + a_{11}t + a_{12}t^2 + a_{13}t^3$

Second cubic: $\theta(t) = a_{20} + a_{21}t + a_{22}t + a_{23}t^3$



$$
\begin{cases}
\theta_0 = a_{10} \\
\theta_v = a_{10} + a_{11}t_{f1} + a_{12}t_{f1}^2 + a_{13}t_{f1}^3 \\
\theta_v = a_{20} \\
\theta_q = a_{20} + a_{21}t_{f2} + a_{22}t_{f2}^2 + a_{23}t_{f2}^3 \\
0 = a_{11} & \text{Zero velocity at initial point} \\
0 = a_{21} + 2a_{22}t_{f2} + 3a_{23}t_{f2}^2 & \text{Zero velocity at final point} \\
\underline{a_{11} + 2a_{12}t_{f1} + 3a_{13}t_{f1}^2 = a_{21}} & \text{(continuous velocity at } \theta_v\text{)} \\
\underline{2a_{12} + 6a_{13}t_{f1} = 2a_{22}} & \text{(continuous acceleration at } \theta_v\text{)}
\end{cases}
$$

solving these equations

$$
\begin{cases}
a_{10} = \theta_0 \\
a_{11} = 0 \\
a_{12} = \dfrac{12\theta_v - 3\theta_q - 9\theta_0}{4t_f^2} \\
a_{13} = \dfrac{-8\theta_v + 3\theta_q + 5\theta_0}{4t_f^3} \\
a_{20} = \theta_v \\
a_{21} = \dfrac{3\theta_q - 3\theta_0}{4t_f} \\
a_{22} = \dfrac{-12\theta_v + 6\theta_q + 6\theta_v}{4t_f^2} \\
a_{23} = \dfrac{8\theta_v - 5\theta_q - 3\theta_0}{4t_f^3}
\end{cases}
$$

Higher-order polynomials - When position, velocity, acceleration are specified

Quintic polynomial

$$\theta(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5,$$

constraints

$$\theta_0 = a_0,$$

$$\theta_f = a_0 + a_1 t_f + a_2 t_f^2 + a_3 t_f^3 + a_4 t_f^4 + a_5 t_f^5,$$

$$\dot{\theta}_0 = a_1,$$

$$\dot{\theta}_f = a_1 + 2a_2 t_f + 3a_3 t_f^2 + 4a_4 t_f^3 + 5a_5 t_f^4,$$

$$\ddot{\theta}_0 = 2a_2,$$

$$\ddot{\theta}_f = 2a_2 + 6a_3 t_f + 12a_4 t_f^2 + 20a_5 t_f^3.$$

6 equations with 6 unkowns, we get

$$a_0 = \theta_0,$$

$$a_1 = \dot{\theta}_0,$$

$$a_2 = \frac{\ddot{\theta}_0}{2},$$

$$a_3 = \frac{20\theta_f - 20\theta_0 - (8\dot{\theta}_f + 12\dot{\theta}_0)t_f - (3\ddot{\theta}_0 - \ddot{\theta}_f)t_f^2}{2t_f^3},$$

$$a_4 = \frac{30\theta_0 - 30\theta_f + (14\dot{\theta}_f + 16\dot{\theta}_0)t_f + (3\ddot{\theta}_0 - 2\ddot{\theta}_f)t_f^2}{2t_f^4},$$

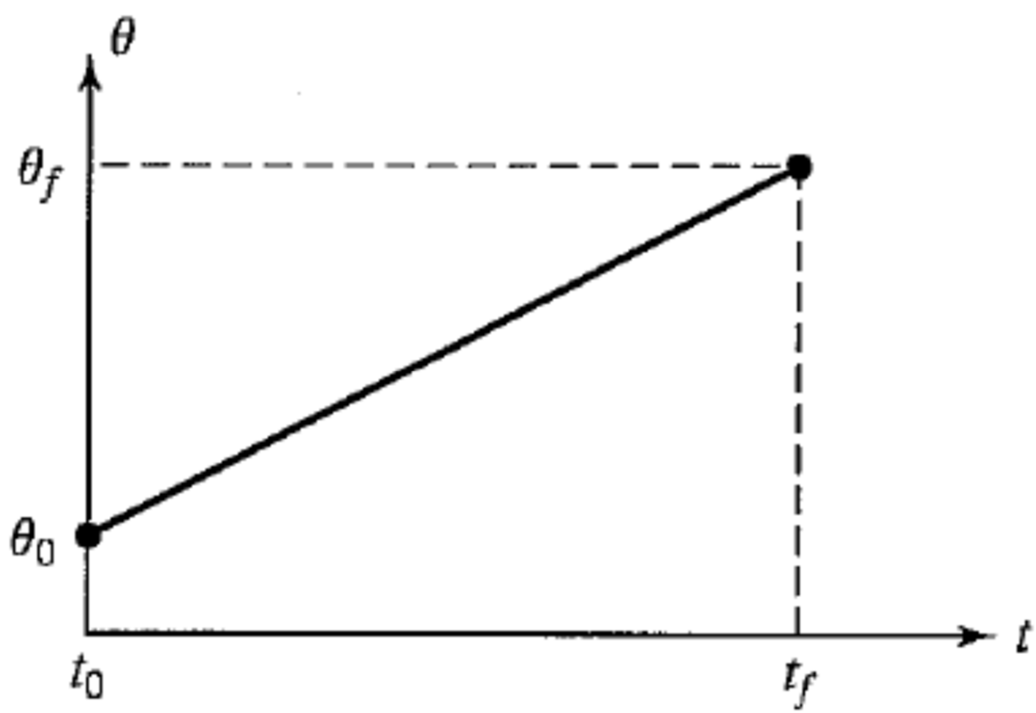
$$a_5 = \frac{12\theta_f - 12\theta_0 - (6\dot{\theta}_f + 6\dot{\theta}_0)t_f - (\ddot{\theta}_0 - \ddot{\theta}_f)t_f^2}{2t_f^5}.$$

Linear function with parabolic blends

linear path shape: to interpolate linearly to move from the present joint position to the final position.

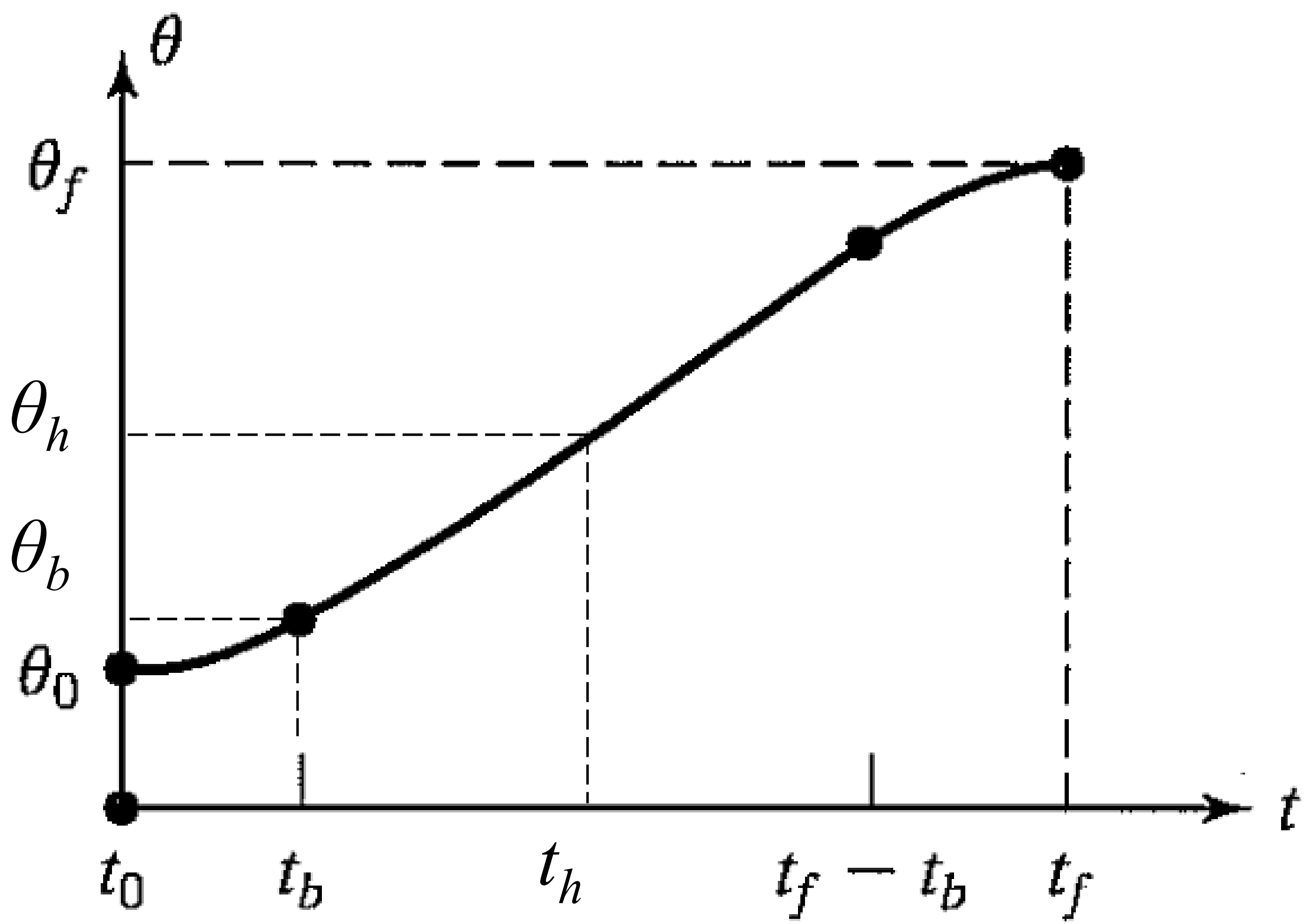
Attention: although the motion of each joint is linear, the end-effector in general does not move in a straight line in space.

Problem: straightforward linear interpolation would cause the velocity to be discontinuous at the beginning and end of the motion.



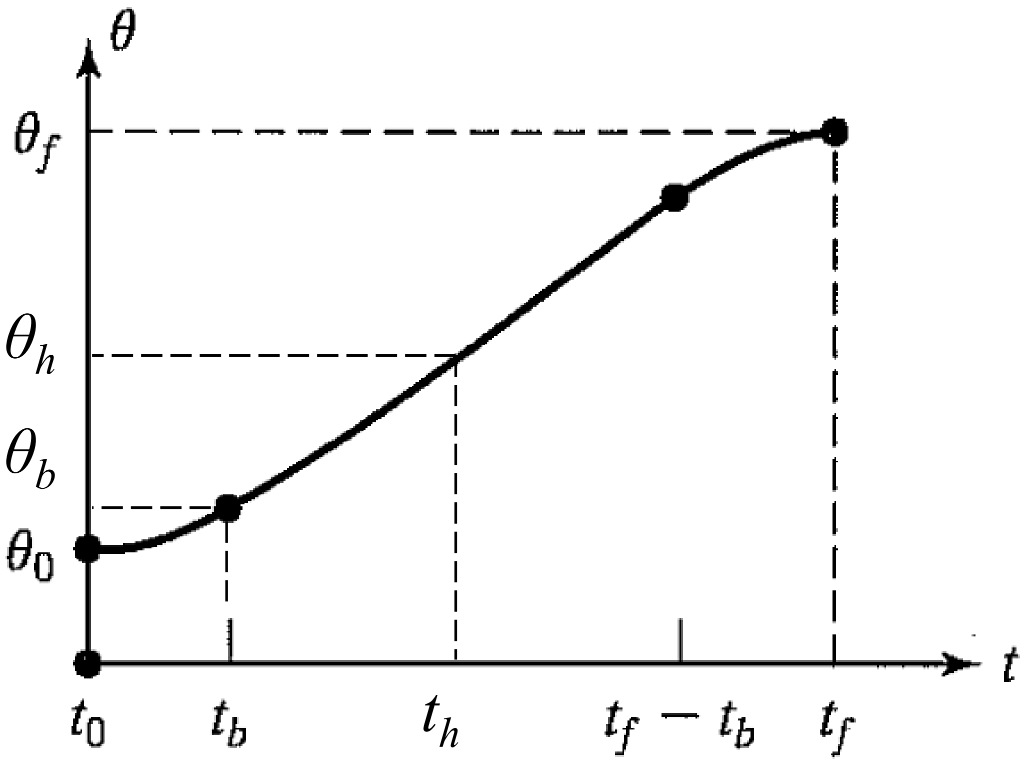
Solution:

Start with the linear function but add a parabolic blend region at each path point. During the blend portion of the trajectory, constant acceleration.

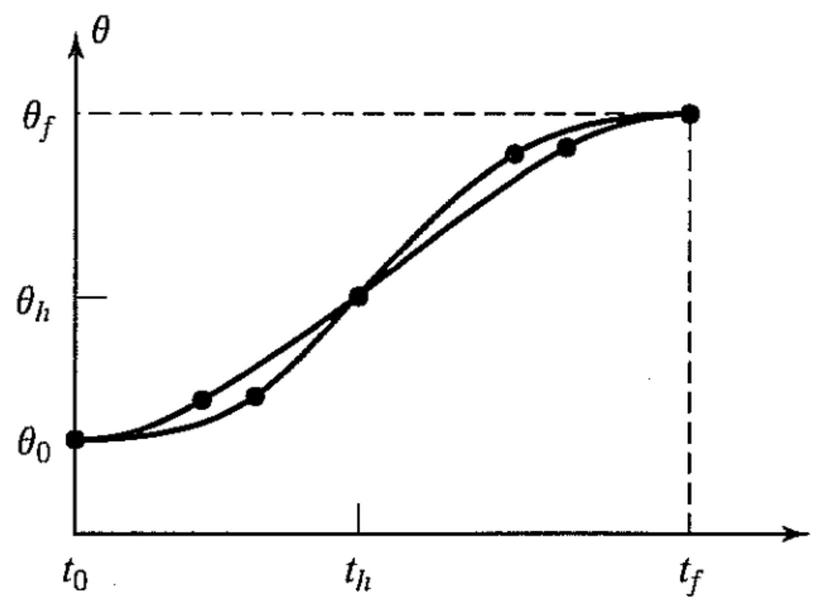


Assumption:

1. Parabolic blends both have the same duration
2. Symmetric about the halfway point in time t_h



Many solutions:



The velocity at the end of the blend region $[t_0, t_b]$ must equal the velocity of the linear section

$$\ddot{\theta} t_b = \frac{\theta_h - \theta_b}{t_h - t_b},$$

where

$\ddot{\theta}$ - the acceleration acting during the blend region

θ_b is given as

$$\theta_b = \theta_0 + \tfrac{1}{2}\ddot{\theta} t_b^2.$$

combine the above two and $t_f = 2t_h$, $\theta_h = (\theta_0 + \theta_f)/2$, we get

$$\ddot{\theta} t_b^2 - \ddot{\theta} t_f t_b + (\theta_f - \theta_0) = 0$$

solve it

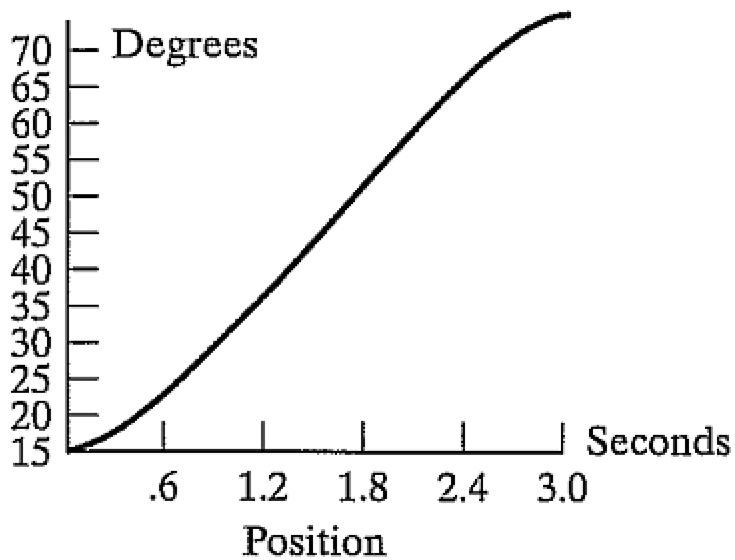
$$t_b = \frac{t_f}{2} - \frac{\sqrt{\ddot{\theta}^2 t_f^2 - 4\ddot{\theta}(\theta_f - \theta_0)}}{2\ddot{\theta}}$$

The constraint on the acceleration used in the blend is

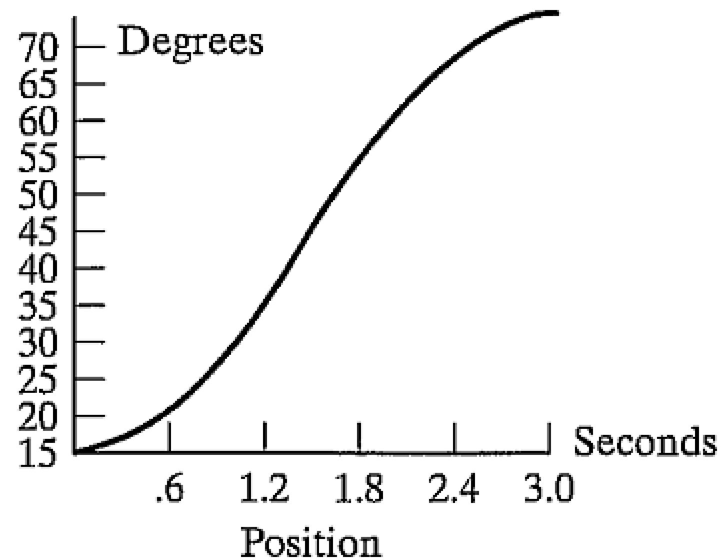
$$\ddot{\theta} \geq \frac{4(\theta_f - \theta_0)}{t_f^2}$$

=: two parabolic blends, no linear section; acceleration ↑: blend region ↓, linear region ↑; infinite acceleration: simple linear-interpolation.

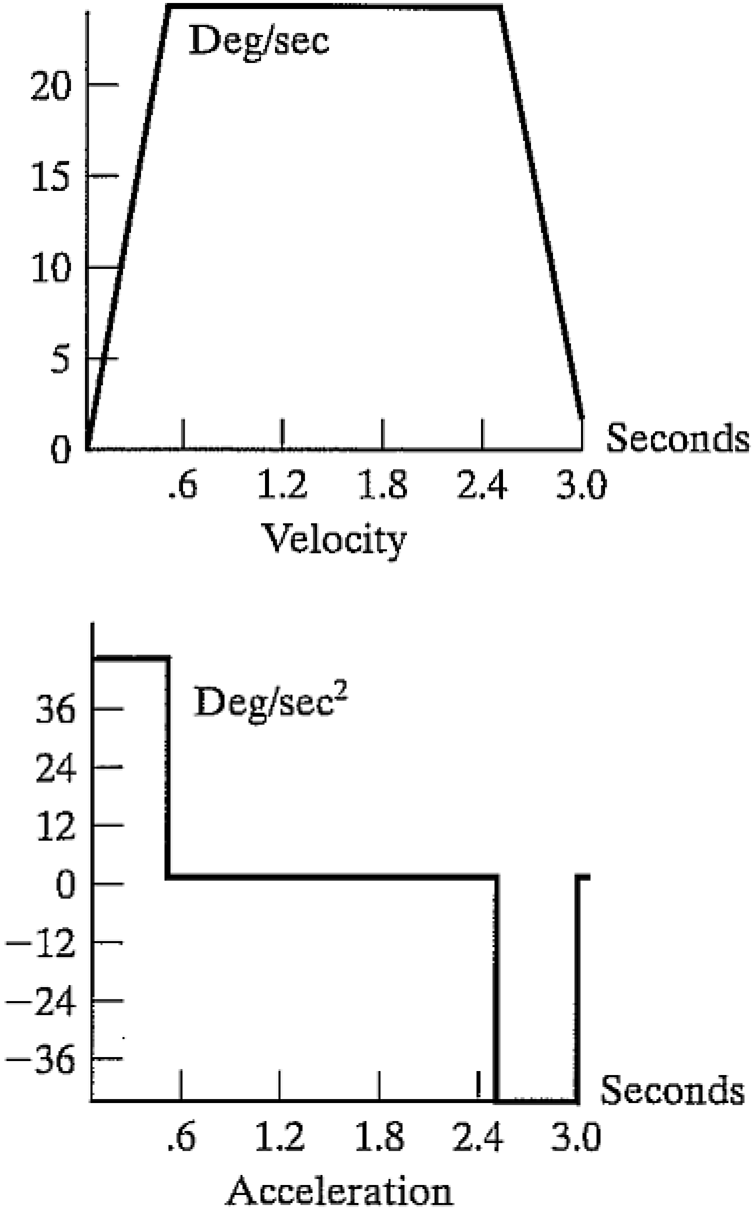
EXAMPLE 7.3 A single-link robot with a rotary joint is motionless at $\theta_0 = 15°$. It is desired to move the joint in a smooth manner to $\theta_f = 75°$ in 3s. Show two examples of a linear path with parabolic blends.



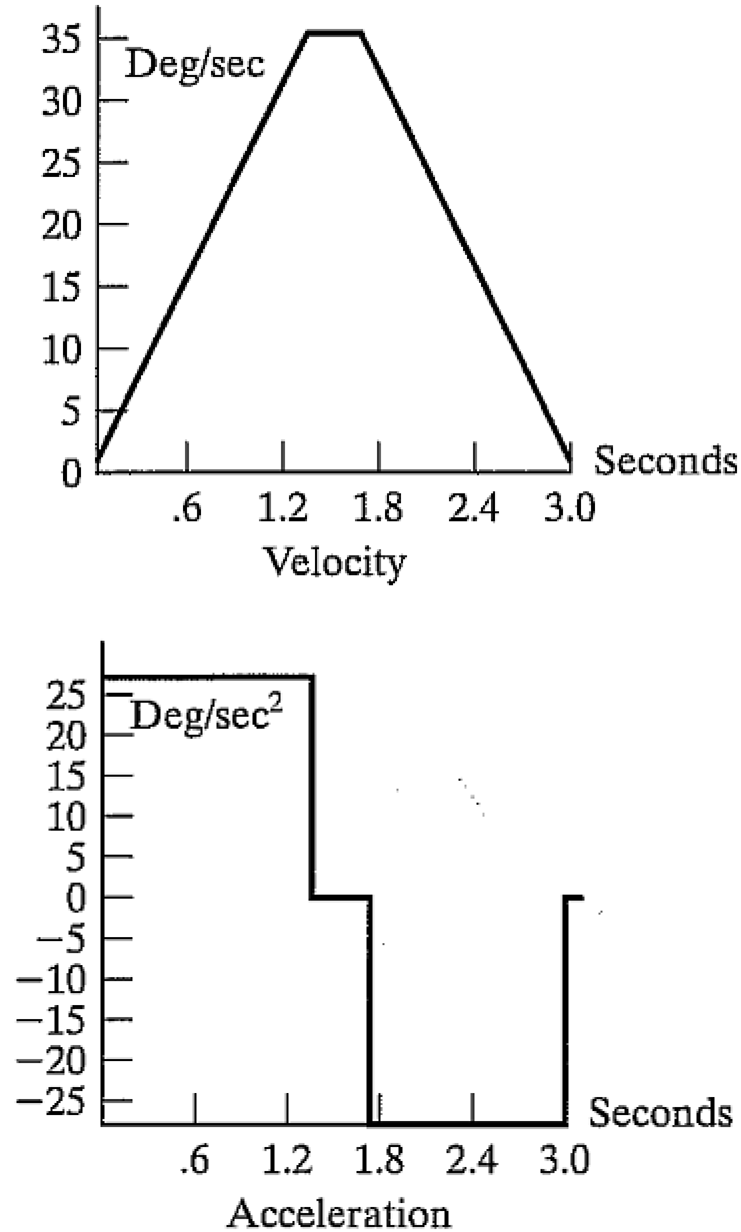
high acceleration



low acceleration



high acceleration



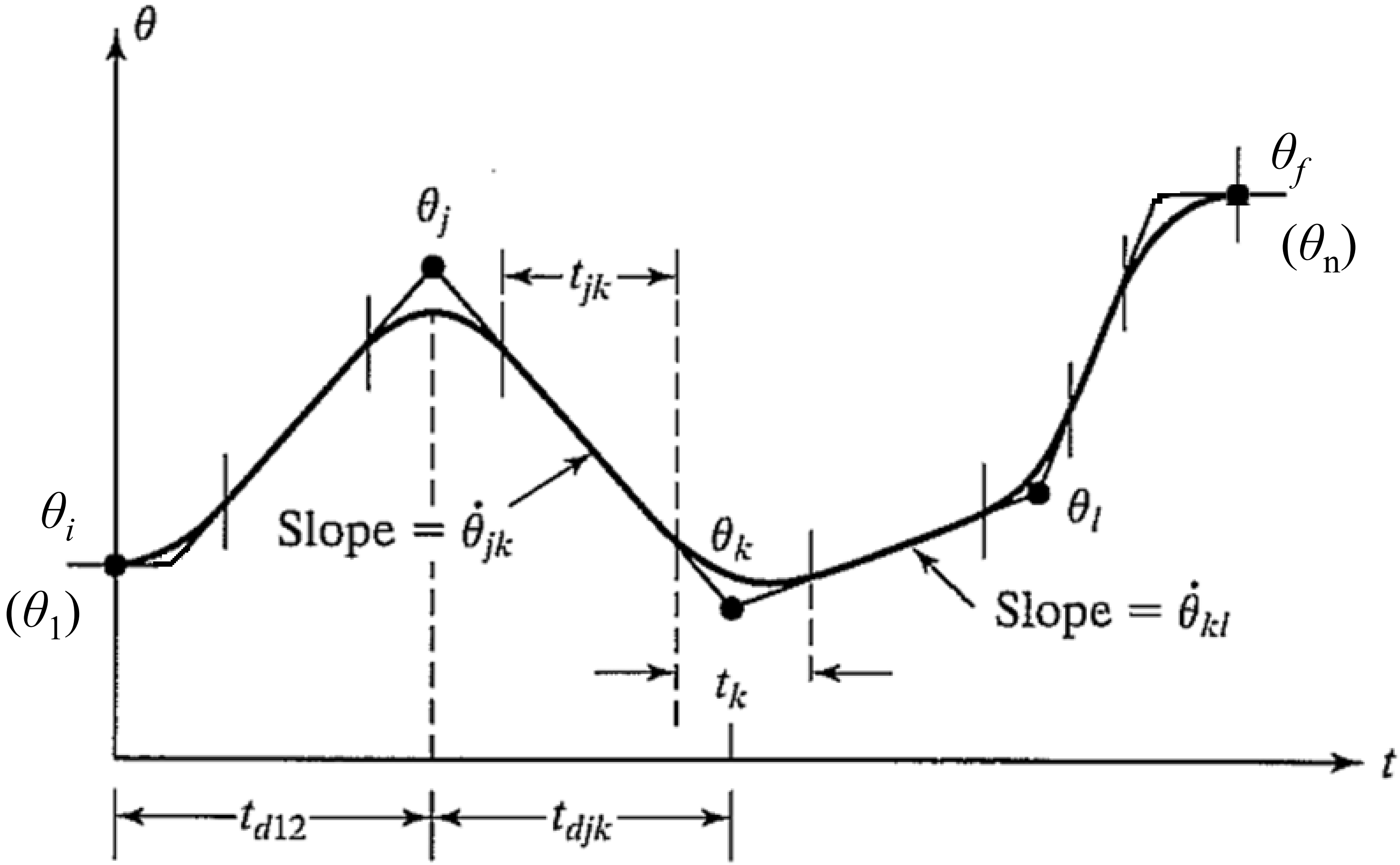
low acceleration

Linear function with parabolic blends for a path with via points

via points: $\theta_j, \theta_k, \theta_l$.

Given: $\theta_1, \dots, \theta_k, \dots, \theta_n, t_{d\,jk}, |\ddot{\theta}_k|$

To calculate: each segment

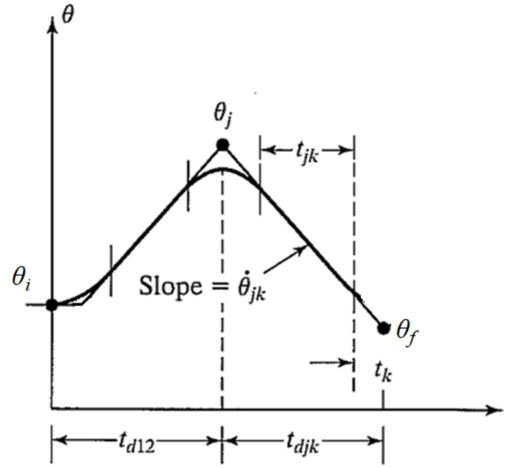


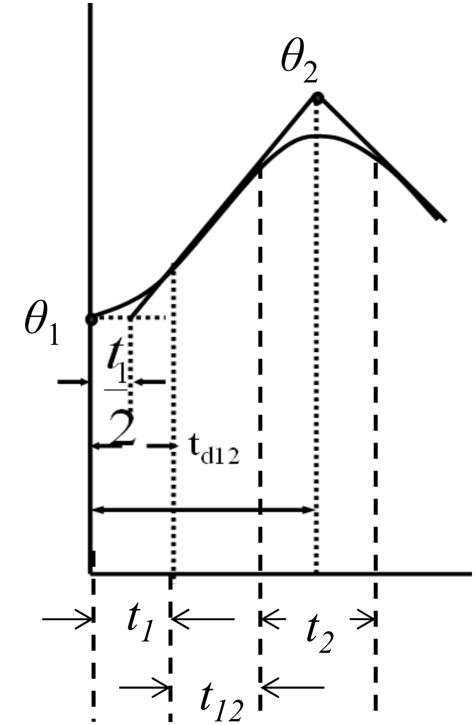
Case 1: only 1 via point

Step1.

Velocity continuity at t_1: $\dfrac{\theta_2-\theta_1}{t_{d12}-\frac{1}{2}t_1}=\ddot{\theta}_1 t_1$

$$\begin{cases} \boxed{t_1 = t_{d12}-\sqrt{t_{d12}^2-\dfrac{2(\theta_2-\theta_1)}{\ddot{\theta}_1}}} \\ \ddot{\theta}_1 = \operatorname{sgn}(\theta_2-\theta_1)\left|\ddot{\theta}_1\right| \\ \dot{\theta}_{12} = \dfrac{\theta_2-\theta_1}{t_{d12}-\frac{1}{2}t_1} \end{cases}$$





Step2.

Velocity continuity at t_3: $\dfrac{\theta_3-\theta_2}{t_{d23}-\frac{1}{2}t_3}=-\ddot{\theta}_3 t_3$

$$\begin{cases} \boxed{t_3 = t_{d23}-\sqrt{t_{d23}^2+\dfrac{2(\theta_3-\theta_2)}{\ddot{\theta}_3}}} \\ \ddot{\theta}_3 = -\operatorname{sgn}(\theta_3-\theta_2)\left|\ddot{\theta}_3\right| \\ \dot{\theta}_{23} = \dfrac{\theta_3-\theta_2}{t_{d23}-\frac{1}{2}t_3} \end{cases}$$

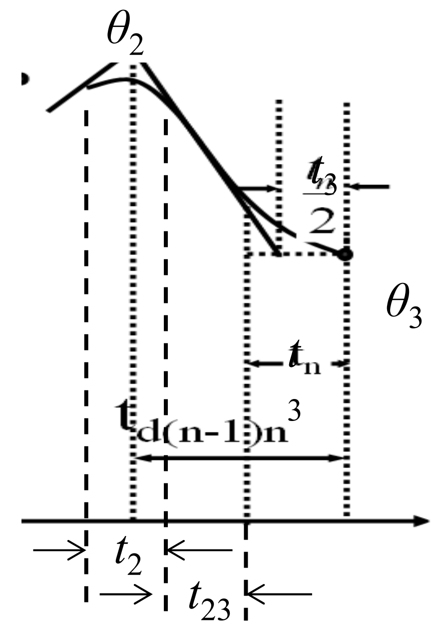
Step3.

$$\ddot{\theta}_2 = \operatorname{sgn}(\dot{\theta}_{23}-\dot{\theta}_{12})\left|\ddot{\theta}_2\right|$$

$$t_2 = \frac{\dot{\theta}_{23}-\dot{\theta}_{12}}{\ddot{\theta}_2}$$

$$t_{12} = t_{d12}-t_1-\frac{1}{2}t_2$$

$$t_{23} = t_{d23}-\frac{1}{2}t_2-t_3$$

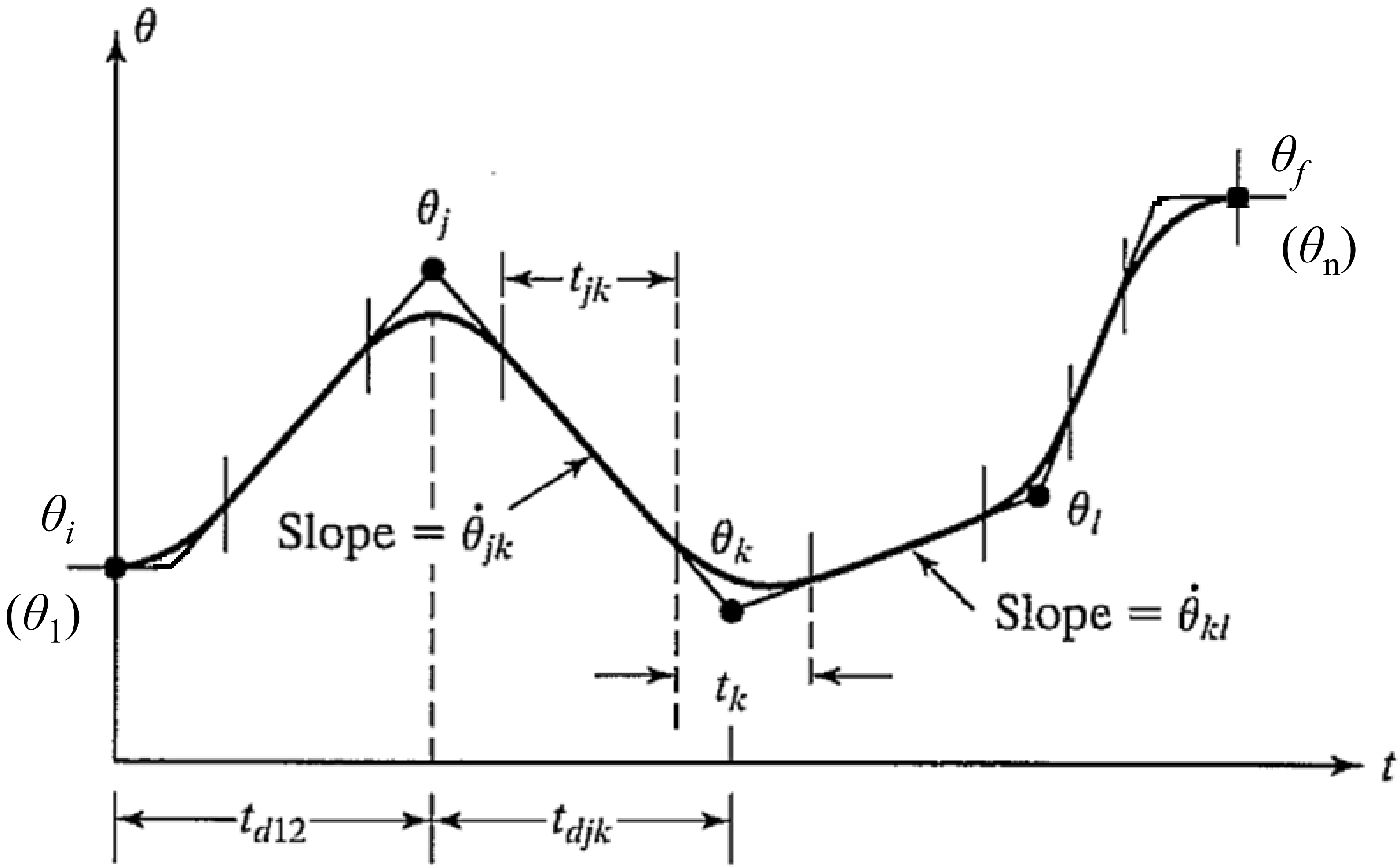


Case 2: ≥ 2 via points

step1: inner segments.

step2: first segment.

step3: last segment.

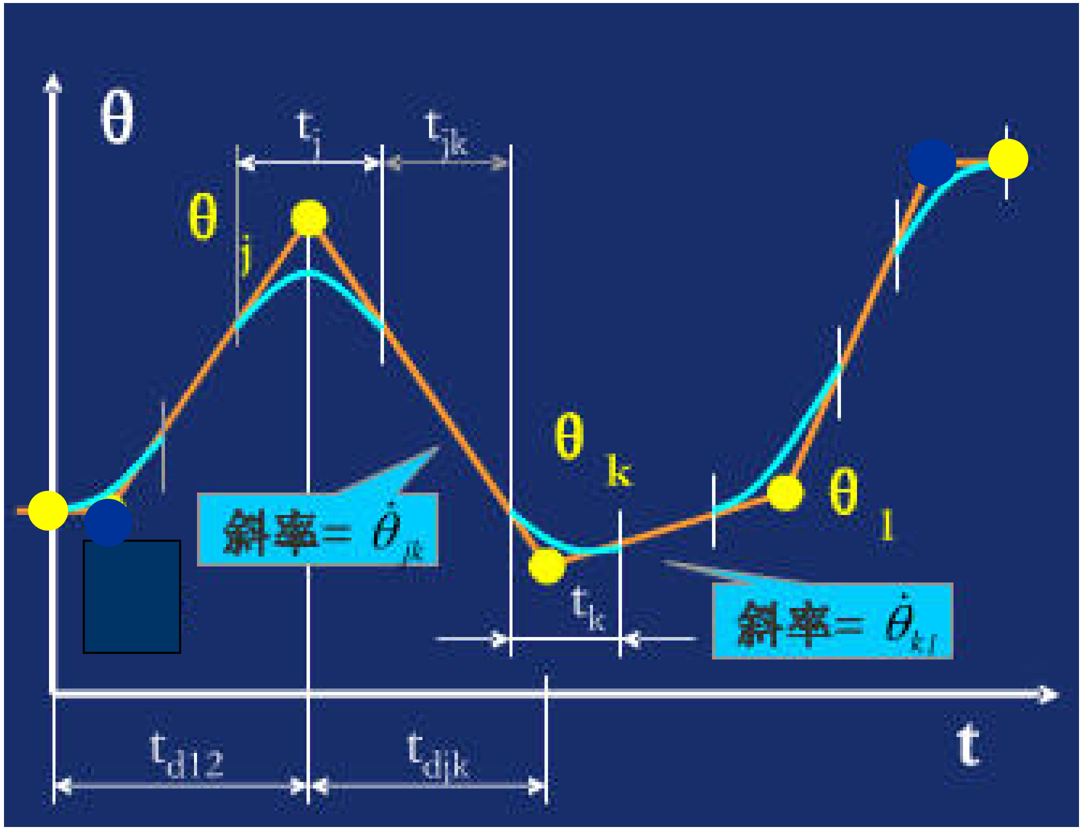


Inner segments

Given: θ_j, θ_k , t_{djk},
$\left|\ddot{\theta}_k\right|$ - the acceleration during the blend at point k

To calculate the inner segment, i.e., starting from $j = 2$, $k = 3$, until $j = n-2$, $k = n-1$.

$$\begin{cases} \dot{\theta}_{jk} = \dfrac{\theta_k - \theta_j}{t_{djk}} \\ \ddot{\theta}_k = \mathrm{sgn}(\dot{\theta}_{kl} - \dot{\theta}_{jk})\left|\ddot{\theta}_k\right| \\ t_k = \dfrac{\dot{\theta}_{kl} - \dot{\theta}_{jk}}{\ddot{\theta}_k} \\ t_{jk} = t_{djk} - \dfrac{1}{2}t_j - \dfrac{1}{2}t_k \end{cases}$$



First segment

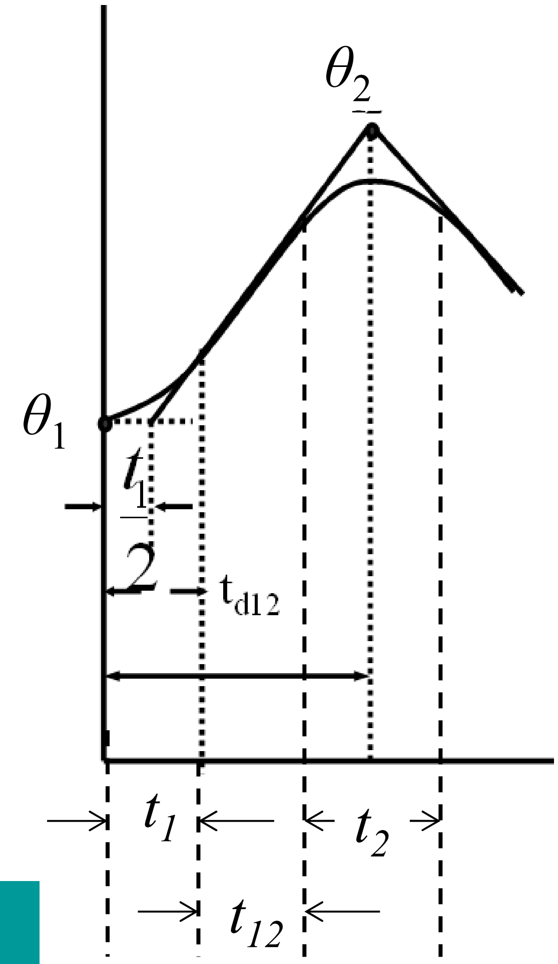
Solve for t_1 by equating two expressions for the velocity during the linear phase:

$$\frac{\theta_2 - \theta_1}{t_{d12} - \frac{1}{2}t_1} = \ddot{\theta}_1 t_1$$

$$\left\{\begin{array}{l} \boxed{t_1 = t_{d12} - \sqrt{t_{d12}^2 - \frac{2(\theta_2 - \theta_1)}{\ddot{\theta}_1}}} \\ \ddot{\theta}_1 = \mathrm{sgn}(\theta_2 - \theta_1)\left|\ddot{\theta}_1\right| \\ \dot{\theta}_{12} = \frac{\theta_2 - \theta_1}{t_{d12} - \frac{1}{2}t_1} \\ t_{12} = t_{d12} - t_1 - \frac{1}{2}t_2 \end{array}\right.$$

$\left|\ddot{\theta}_1\right|$ - acceleration at point θ_1

t_2 has been calculated by the inner segments



Last segment

Solve for t_n by equating two expressions for the velocity during the linear phase:

$$\frac{\theta_{n-1} - \theta_n}{t_{d(n-1)n} - \frac{1}{2} t_n} = \ddot{\theta}_n t_n$$

$$\boxed{t_n = t_{d(n-1)n} - \sqrt{t_{d(n-1)n}^2 + \frac{2(\theta_n - \theta_{n-1})}{\ddot{q}_n}}}$$

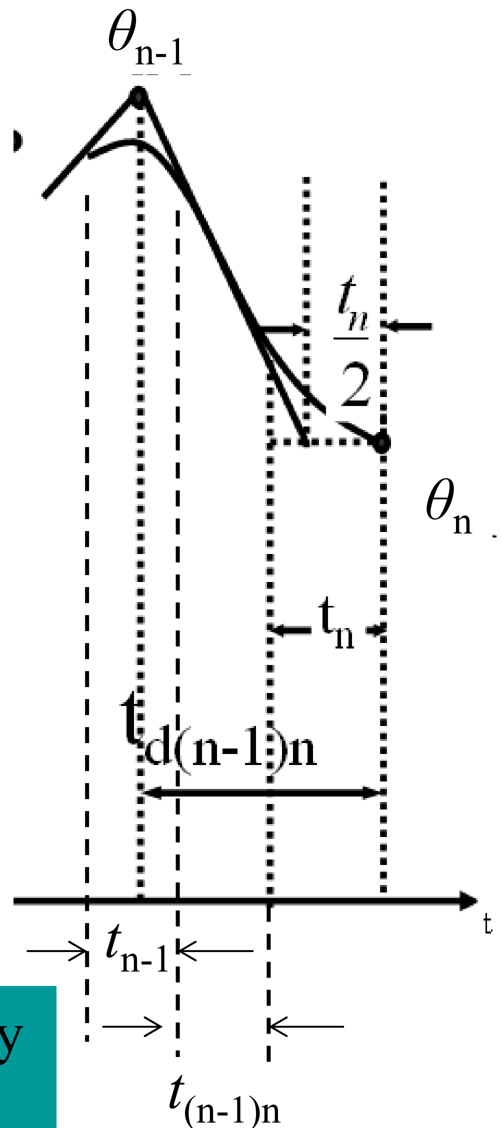
$$\ddot{\theta}_n = \mathrm{sgn}(\theta_{n-1} - \theta_n)\left|\ddot{\theta}_n\right|$$

$\left|\ddot{\theta}_n\right|$ - acceleration at point θ_n

$$\dot{\theta}_{(n-1)n} = \frac{\theta_n - \theta_{n-1}}{t_{d(n-1)n} - \frac{1}{2} t_n}$$

$$t_{(n-1)n} = t_{d(n-1)n} - t_n - \frac{1}{2} t_{n-1}$$

t_{n-1} has been calculated by the inner segments



EXAMPLE 7.4 Path points in degree: 10, 35, 25,10. Duration of these segments: 2, 1, 3 s. Magnitude of acceleration: 50 degrees/seconds^2.

Solution:

$$\ddot{\theta}_1 = \operatorname{sgn}(\theta_2 - \theta_1)\left|\ddot{\theta}_1\right| = 50^\circ / \mathrm{s}^2$$

$$t_1 = t_{d12} - \sqrt{t_{d12}^2 - \frac{2(\theta_2 - \theta_1)}{\ddot{\theta}_1}} = 2 - \sqrt{4 - \frac{2(35-10)}{50}} = 0.27\mathrm{s}$$

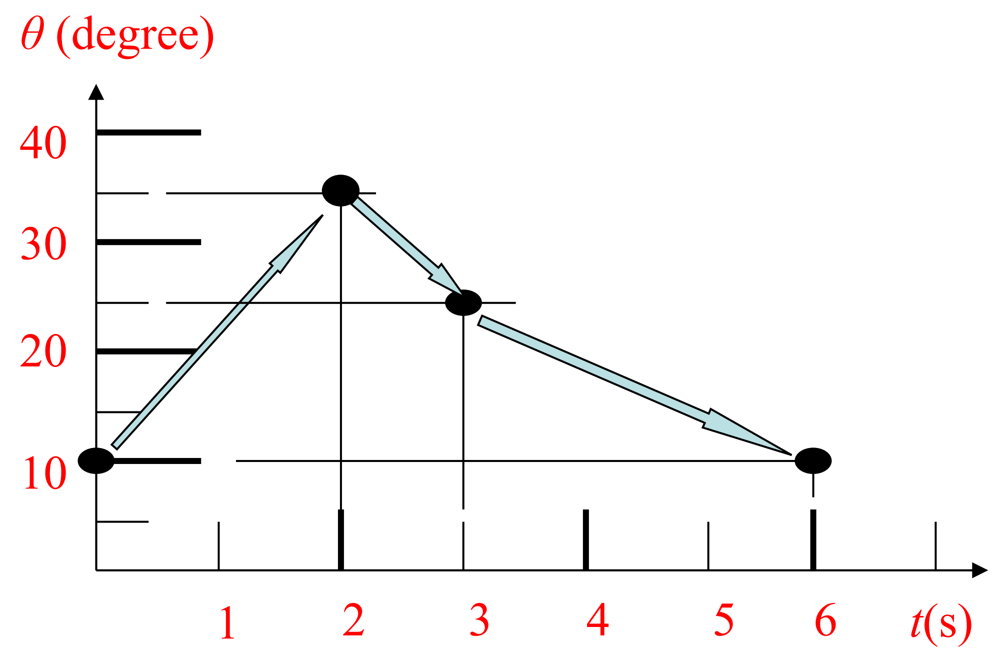
$$\dot{\theta}_{12} = \frac{\theta_2 - \theta_1}{t_{d12} - \frac{1}{2}t_1} = \frac{35-10}{2-0.5\times 0.27} = 13.5^\circ / \mathrm{s}$$

$$\dot{\theta}_{23} = \frac{\theta_3 - \theta_2}{t_{d23}} = \frac{25-35}{1} = -10^\circ / \mathrm{s}$$

$$\ddot{\theta}_2 = \operatorname{sgn}(\dot{\theta}_{23} - \dot{\theta}_2)\left|\ddot{\theta}_2\right| = -50^\circ / \mathrm{s}^2$$

$$t_2 = \frac{\dot{\theta}_{23} - \dot{\theta}_{12}}{\ddot{\theta}_2} = \frac{-10-13.5}{-50} = 0.47\mathrm{s}$$

$$t_{12} = t_{d12} - t_1 - \frac{1}{2}t_2 = 2 - 0.27 - \frac{1}{2}\times 0.47 = 1.5\mathrm{s}$$



$$\ddot{\theta}_4 = \mathrm{sgn}(\theta_3 - \theta_4)\left|\ddot{\theta}_4\right| = 50^\circ / \mathrm{s}^2$$

$$t_4 = t_{d34} - \sqrt{t_{d34}^2 - \frac{2(\theta_4 - \theta_3)}{\ddot{\theta}_4}} = 3 - \sqrt{9 + \frac{2(10-25)}{50}} = 0.102\mathrm{s}$$

$$\dot{\theta}_{34} = \frac{\theta_4 - \theta_3}{t_{d34} - \frac{1}{2}t_4} = \frac{10-25}{3-0.5\times 0.102} = -5.1^\circ / \mathrm{s}$$

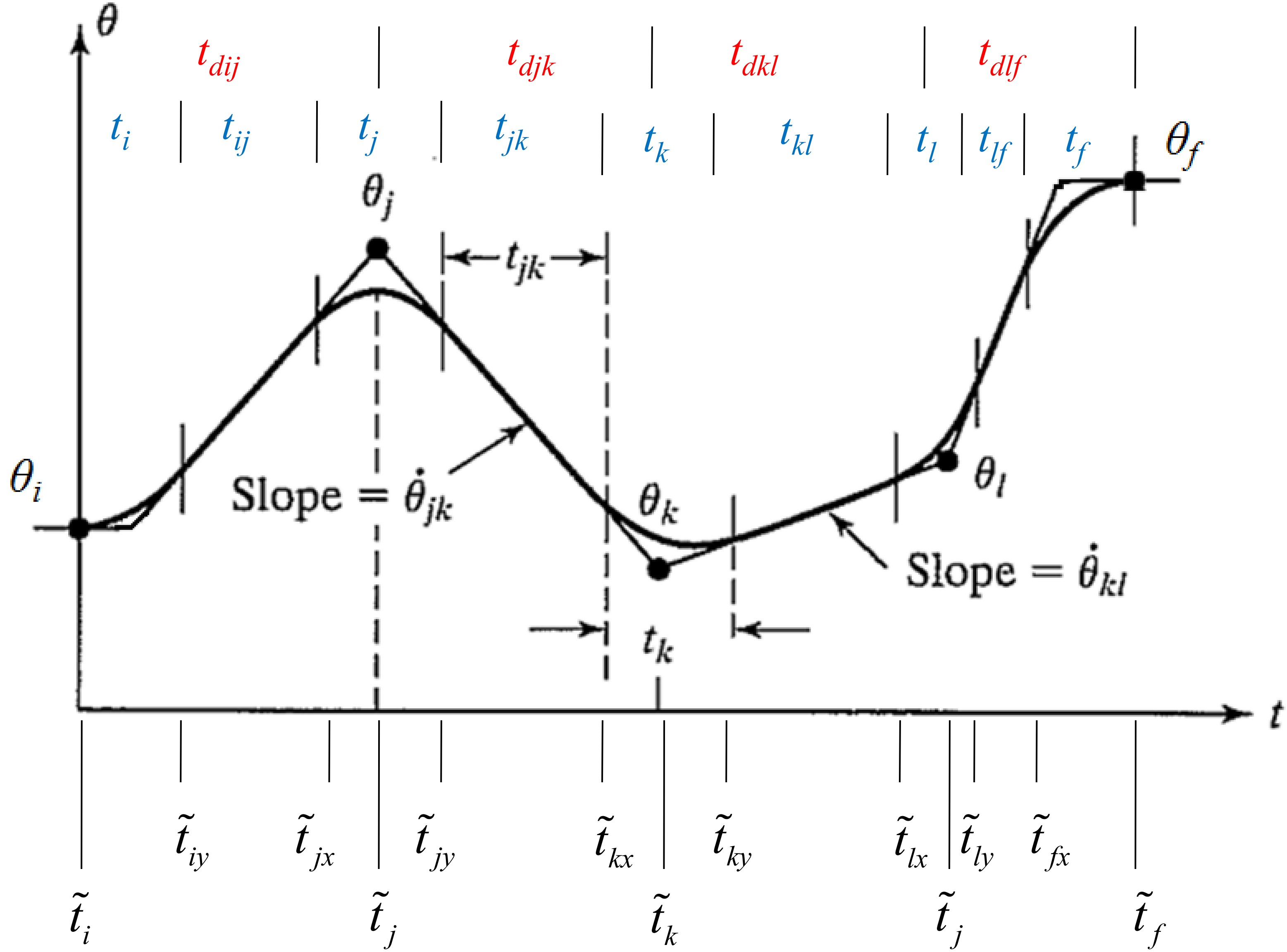
$$\ddot{\theta}_3 = \mathrm{sgn}(\theta_{34} - \theta_{23})\left|\ddot{\theta}_3\right| = 50^\circ / s^2$$

$$t_3 = \frac{\dot{\theta}_{34} - \dot{\theta}_{23}}{\ddot{\theta}_3} = \frac{-5.1-(-10)}{50} = 0.098s$$

$$t_{23} = t_{d23} - \frac{1}{2}t_2 - \frac{1}{2}t_3 = 1 - \frac{1}{2}\times 0.47 - \frac{1}{2}\times 0.098 = 0.716\mathrm{s}$$

$$t_{34} = t_{d34} - t_4 - \frac{1}{2}t_3 = 3 - 0.102 - \frac{1}{2}\times 0.098 = 2.849\mathrm{s}$$

initial point: θ_i,
final point: θ_f,
via points: $\theta_j, \theta_k, \theta_l$.



$\tilde{t}_i \le t < \tilde{t}_{iy}$

parabolic blends $\quad \tilde{t}_i = 0s$

位置方程 $$\theta(t) = \frac{1}{2}\ddot{\theta}\left(t - \tilde{t}_i\right)^2 + \theta_i$$

速度方程 $$\dot{\theta}(t) = \ddot{\theta}\left(t - \tilde{t}_i\right)$$

速度连续 $$\dot{\theta}(\tilde{t}_{iy}) = \ddot{\theta}(\tilde{t}_{iy} - \tilde{t}_i) = \dot{\theta}_{ij} = \frac{\theta_j - \left(\frac{1}{2}\ddot{\theta}\left(\tilde{t}_{iy} - \tilde{t}_i\right)^2 + \theta_i\right)}{\tilde{t}_j - \tilde{t}_{iy}}$$

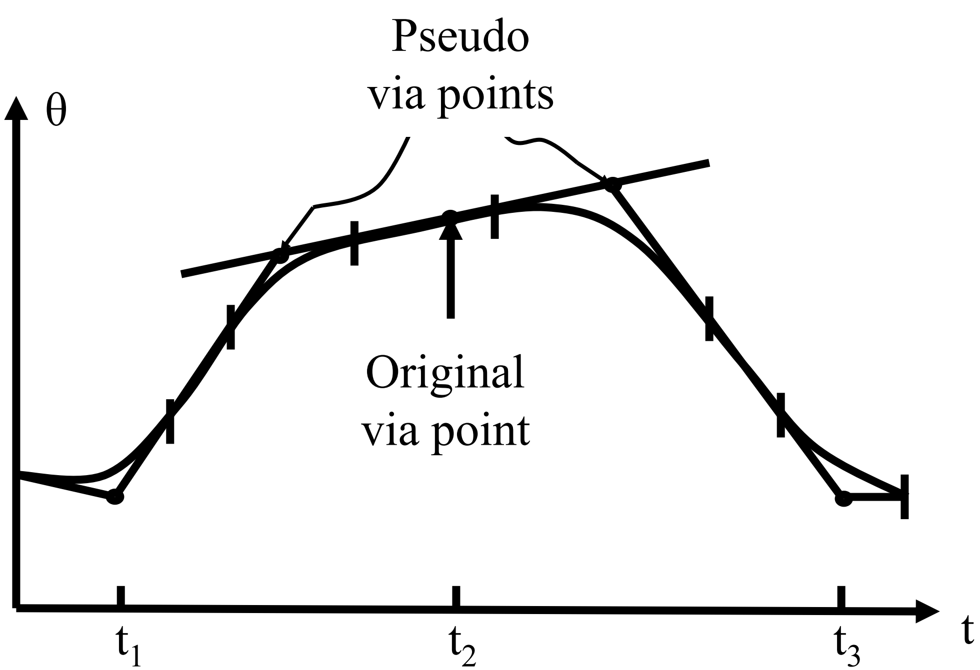
$$\tilde{t}_j = \tilde{t}_i + t_{dij}$$

$$\tilde{t}_{iy} =$$

Attention: the via points are not actually reached unless the manipulator comes to a stop.

•When acceleration capability is sufficiently high, the paths will come quite close to the desired via point.

•If we wish to actually pass through a point without stopping, replace the via point with two pseudo via points, one on each side of the origin. The original via point will now lies in the linear region. Now the via point is called a through point, a path point through which we force the manipulator to pass exactly.



7.4 CARTESIAN-SPACE SCHEMES

--- methods of path generation in which the path shapes are described in terms of functions that compute Cartesian position and orientation as functions of time.

Cartesian straight-line motion

e.g., original point T_{intial} , final point T_{final}, duration t_d

$$T_{initial} = \begin{bmatrix} r^i{}_{11} & r^i{}_{12} & r^i{}_{13} & x^i \\ r^i{}_{21} & r^i{}_{22} & r^i{}_{23} & y^i \\ r^i{}_{31} & r^i{}_{32} & r^i{}_{33} & z^i \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad T_{final} = \begin{bmatrix} r^f{}_{11} & r^f{}_{12} & r^f{}_{13} & x^f \\ r^f{}_{21} & r^f{}_{22} & r^f{}_{23} & y^f \\ r^f{}_{31} & r^f{}_{32} & r^f{}_{33} & z^f \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

We can linearly interpolate its elements of position vector, e.g. x^i, x^f, resulting in a function $x(t)$.

We cannot linearly interpolate its elements of rotation matrix, e.g. $r^i{}_{11}$, $r^f{}_{11}$, because doing so would not necessarily result in a valid rotation matrix at all times.

A Cartesian-space path planning method

principle: angle-axis representation

Cartesian position and orientation representation of {A} in terms of {S} :

$$
{}^{S}\chi_{A} = \begin{bmatrix} {}^{S}P_{AORG} \\ {}^{S}K_{A} \end{bmatrix} \quad \begin{matrix} \textbf{position（3×1）} \\ \textbf{orientation（3×1）} \end{matrix}
$$

${}^{S}K_{A}$:

Direction - axis of rotation

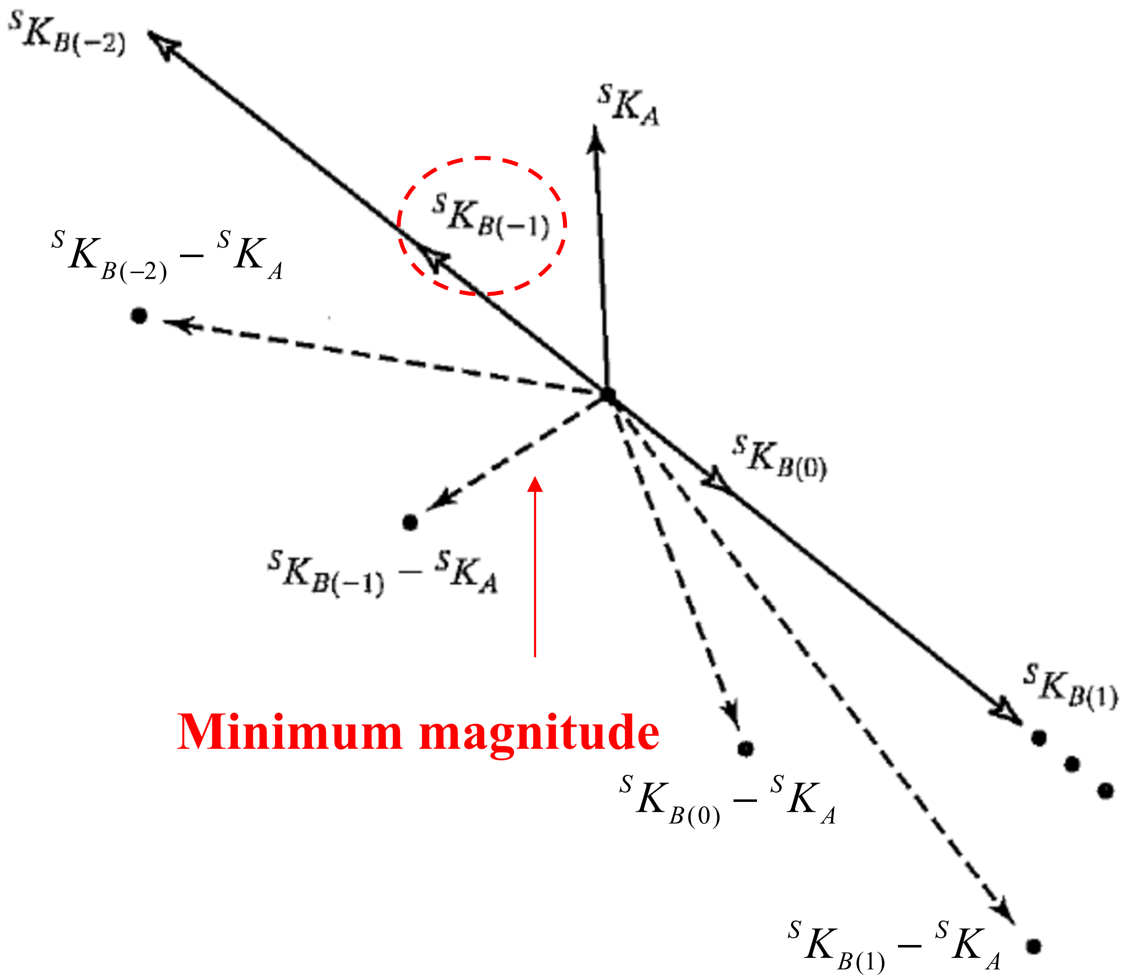
Magnitude - amount of rotation

If every path point is specified in this representation, we then need to describe spline functions that smoothly vary these six quantities from path point to path point as functions of time.

Attention: the angle-axis representation of orientation is not unique, i.e., $\theta_{SA} = \theta_{SA} + n360^{o}$

Method: in going from {A} to {B}, the total amount of rotation should be minimized.

Fig. 4 different ${}^{S}K_{B}$'s , we must choose the one such that $| {}^{S}X_{B} - {}^{S}X_{A} |$ is minimized.



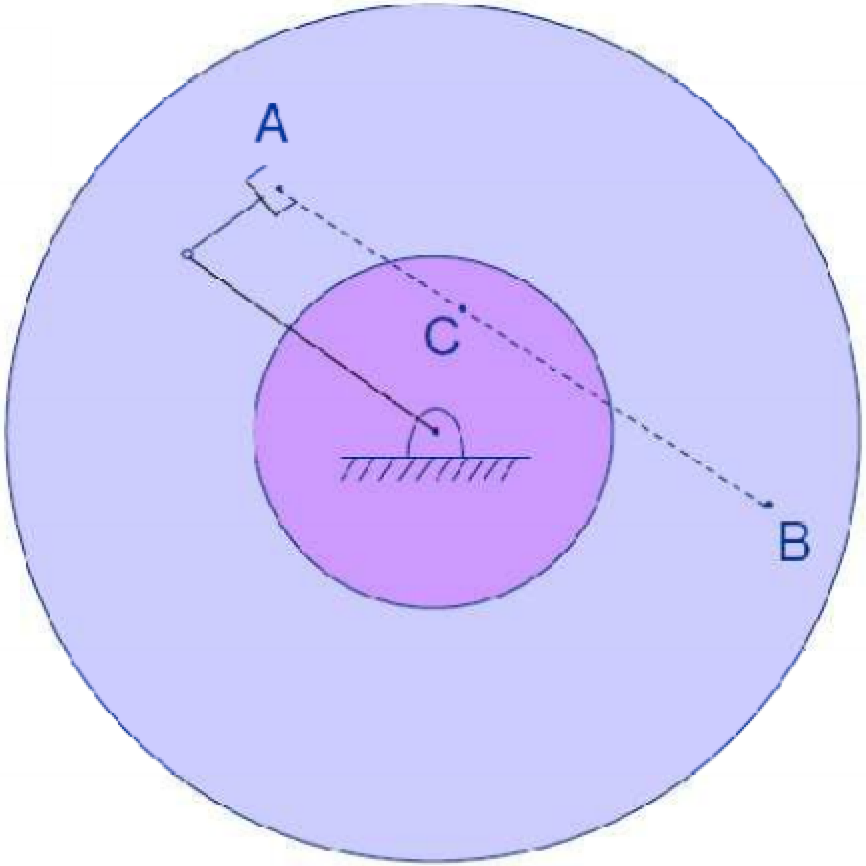
When we use linear interpolation with parabolic sections, one more constraint: the blend times for each degree of freedom must be the same. This will ensure that the resultant motion of all the degrees of freedom will be a straight line in space.

Many other schemes for representing and interpolating the orientation portion of a Cartesian path can be used. Among these are the 3×1 representations of orientation, e.g., Z-Y-Z Euler angles.

7.5 GEOMETRIC PROBLEMS WITH CARTESIAN PATHS

- Cartesian planning difficulties (1/3):
  Intermediate points unreachable

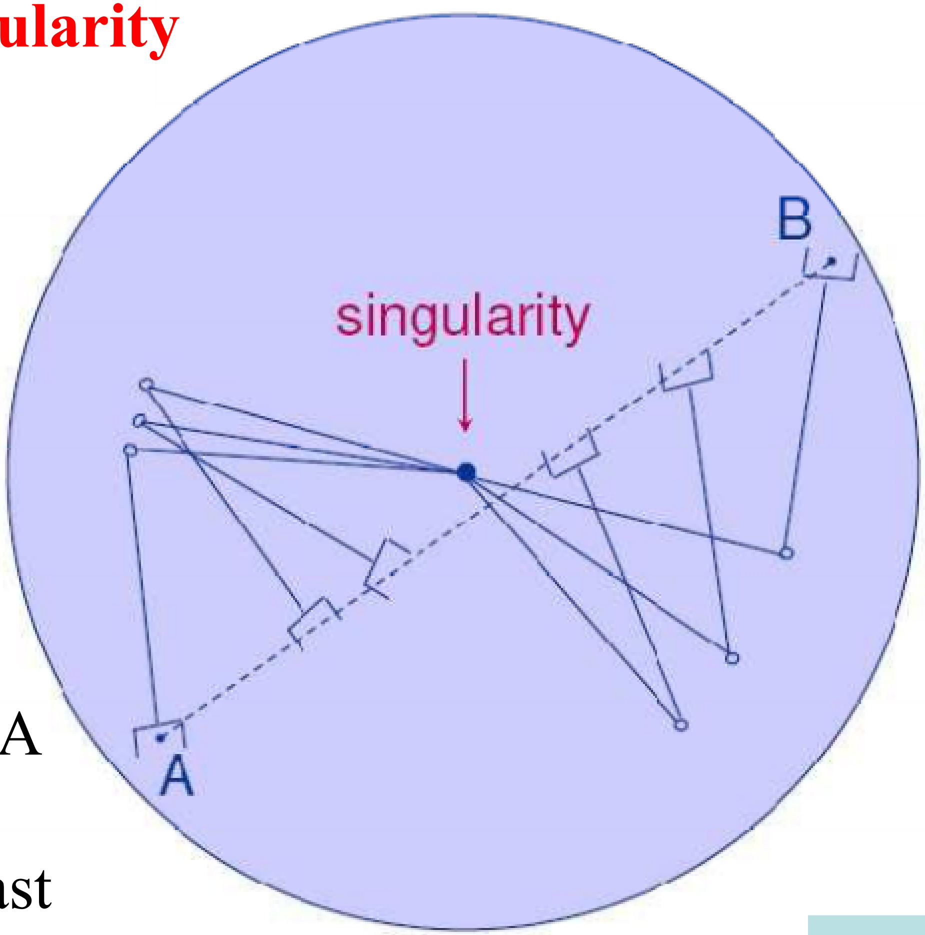
  Initial (A) and Goal (B) Points are reachable, but intermediate points (C) unreachable.



- Cartesian planning difficulties (2/3):

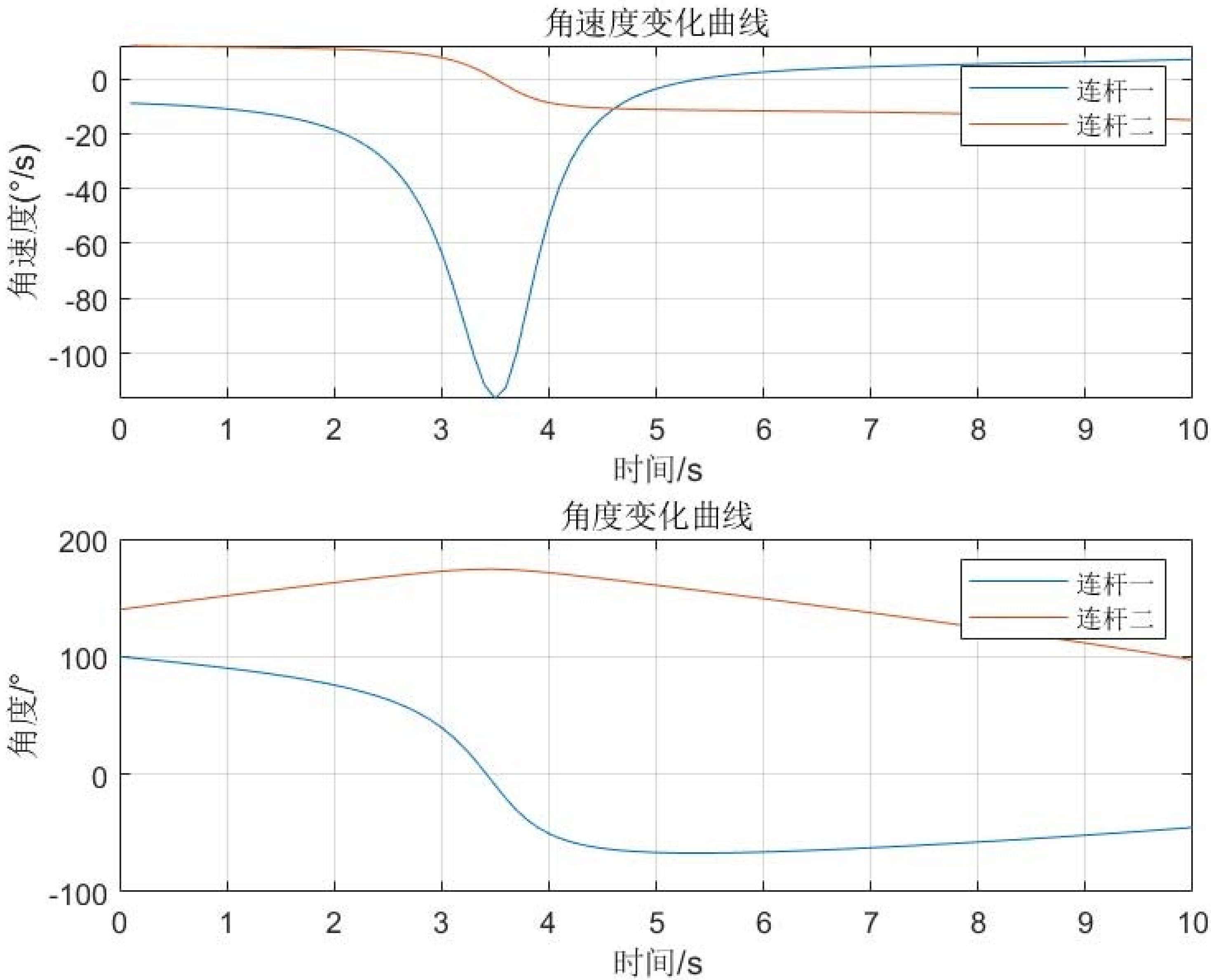
High joint rates near singularity

Approaching singularities some joint velocities go to ∞ causing deviation from the path.



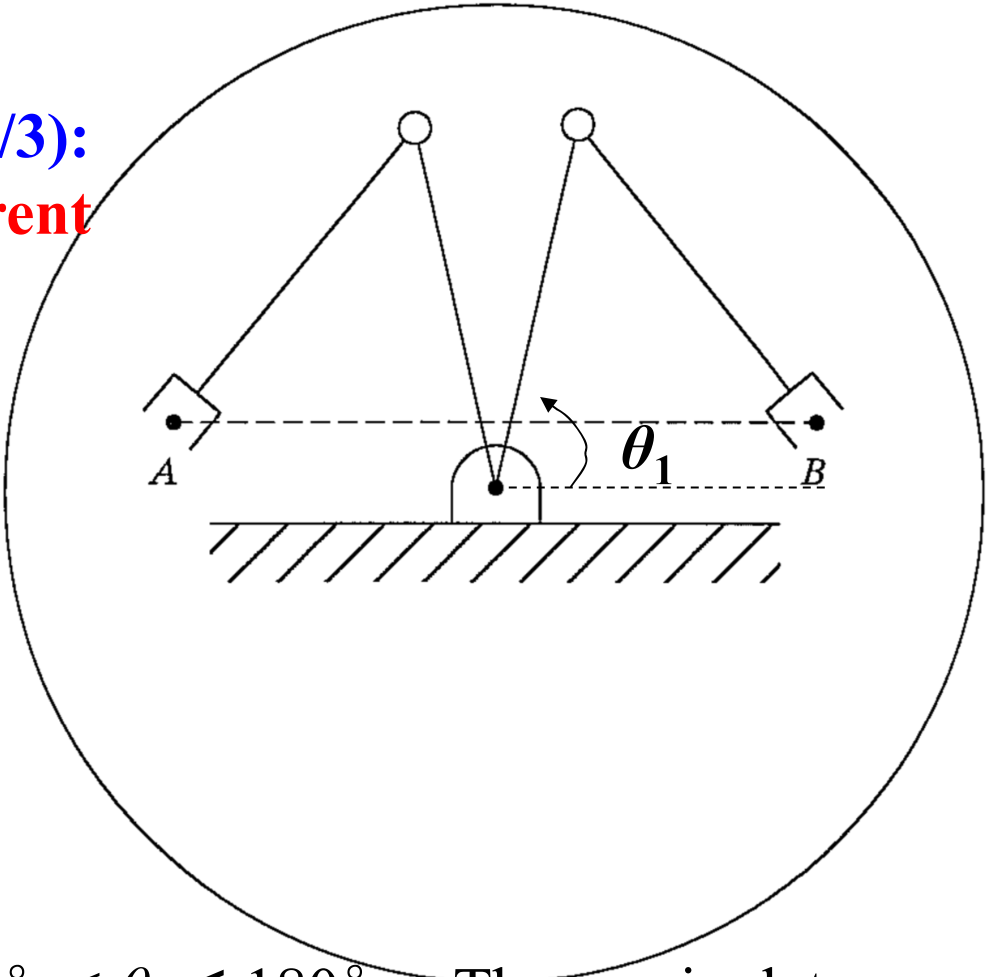
A planar two-link (with equal link lengths) moves along a path from A to B. All points along the path are reachable, but as the robot goes past the middle portion of the path, the velocity of joint one is very high.

Because velocities of the mechanism are upper bounded, this situation usually results in the manipulator's deviating from the desired path.



角速度变化曲线
角速度(°/s)
0
-20
-40
-60
-80
-100
0 1 2 3 4 5 6 7 8 9 10
时间/s
连杆一
连杆二
角度变化曲线
角度/°
200
100
0
-100
0 1 2 3 4 5 6 7 8 9 10
时间/s
连杆一
连杆二

Cartesian planning difficulties (3/3): Start and goal reachable in different solutions



This problem arises if the goal point cannot be reached in the same physical solution as the robot is in at the start point.

Fig. equal link length, joint limit: $0^\circ \leq \theta_1 \leq 180^\circ$. The manipulator can reach all points of the path in some solution, but not in any one solution.

There must be a point in the path where $\theta_1 = 0^\circ$. As the end-effector moves on, for θ_1 cannot be negative, θ_1 will change to a positive value suddenly, so will θ_2. The actual path is no longer the one desired. Path planning fails.

7.7 DESCRIPTION OF PATHS WITH A ROBOT GROGRAMMING ANGUAGE

Exemplified by AL. A, B, C, and D stand for variables of type "frame" in the AL-language. These frames specify path points that we will assume to have been taught or texturally described to the system.

move ARM to C； //in joint-space mode along linear-parabolic-blend paths

move ARM to C via B;

move ARM to C via B, A, D;

move ARM to C with duration = 3*seconds;

move ARM to C linearly with duration = 3*seconds; //"linearly" denotes the Cartesian straight-line motion is to be used

move ARM to C via B with duration = 6*seconds;

move ARM to C via B where duration = 3*seconds; //the first sigment which leads to point B will have a duration of 3s

7.9 COLLISION-FREE PATH PLANNING

Two competing principal techniques:

1. By forming a connected-graph representation of the free space and then searching the graph for collision-free path.

disadvantage: exponential complexity in the number of joints in the device.

2. By creating artificial potential fields around obstacles, which cause the manipulator to avoid the obstacles while they are drawn toward an artificial attractive pole at the goal point.

disadvantage: have a local view of the environment and are subject to becoming "stuck" at local minima of the artificial field.

Path planning supplementary (F.C. Park, K. M. Lynch. Introduction to Robotics: Mechanics, Planning, and Control. 2014.)

Every point in the configuration space, or *C-space* for short, C, corresponds to a unique configuration q of the robot, and every configuration of the robot can be represented as a point in C-*space*.

e.g., a robot arm with n joints can be represented as a list of n joint angles, $q = (\theta_1, \ldots, \theta_n)$.

Free configuration space C_{free}: Consists of the configurations where the robot does not penetrate any obstacle nor violate a joint limit.

Notice: The workspace obstacles partition the configuration space C into two sets, the free space C_{free} and the obstacle space C_{obs}., where $C = C_{free} \cup C_{obs}$.

Free state space X_{free}: $\chi_{free} = \left\{ x \middle| q(x) \in C_{free} \right\}$

state: configuration and velocity, $x = (q, v)$

Example: a 2R planar arm

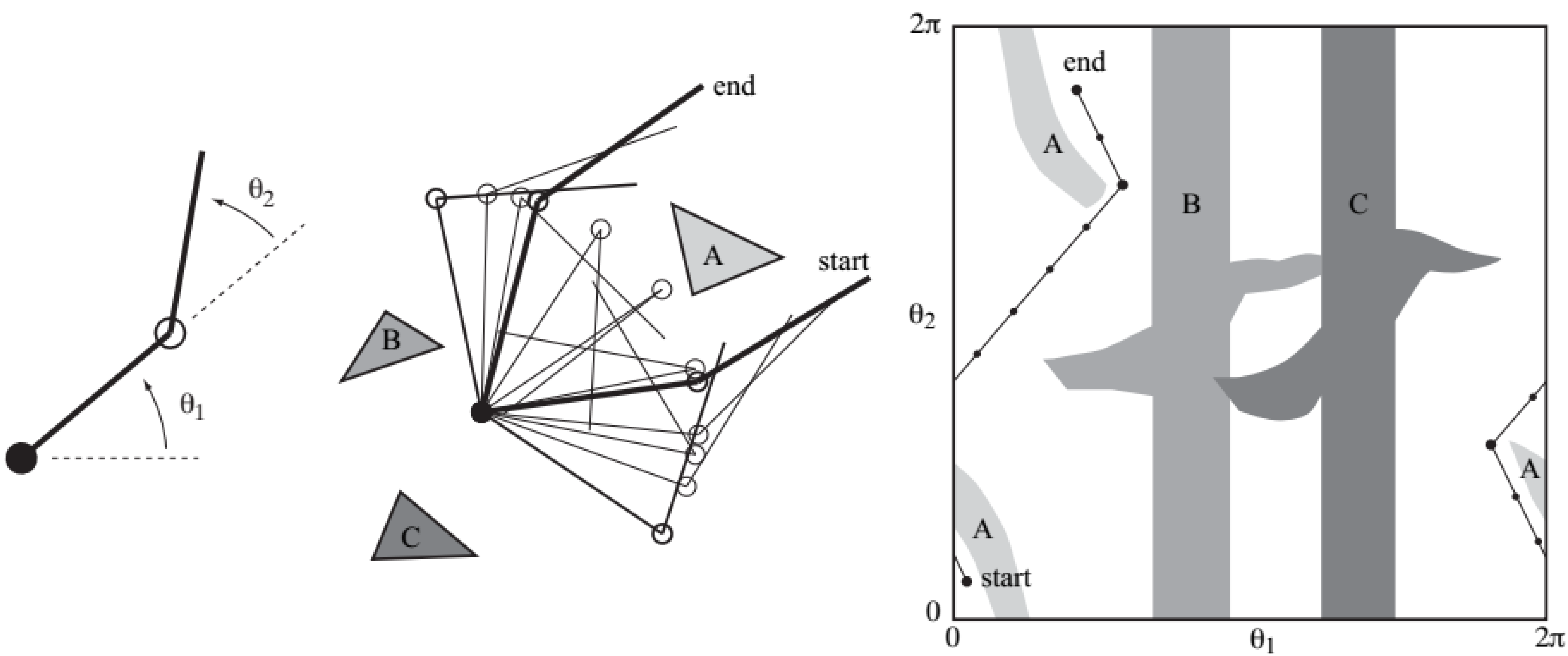


Figure 10.2: (Left) The joint angles of a 2R robot arm. (Middle) The arm navigating among obstacles. (Right) The same motion in C-space.

Example: a circular planar mobile robot

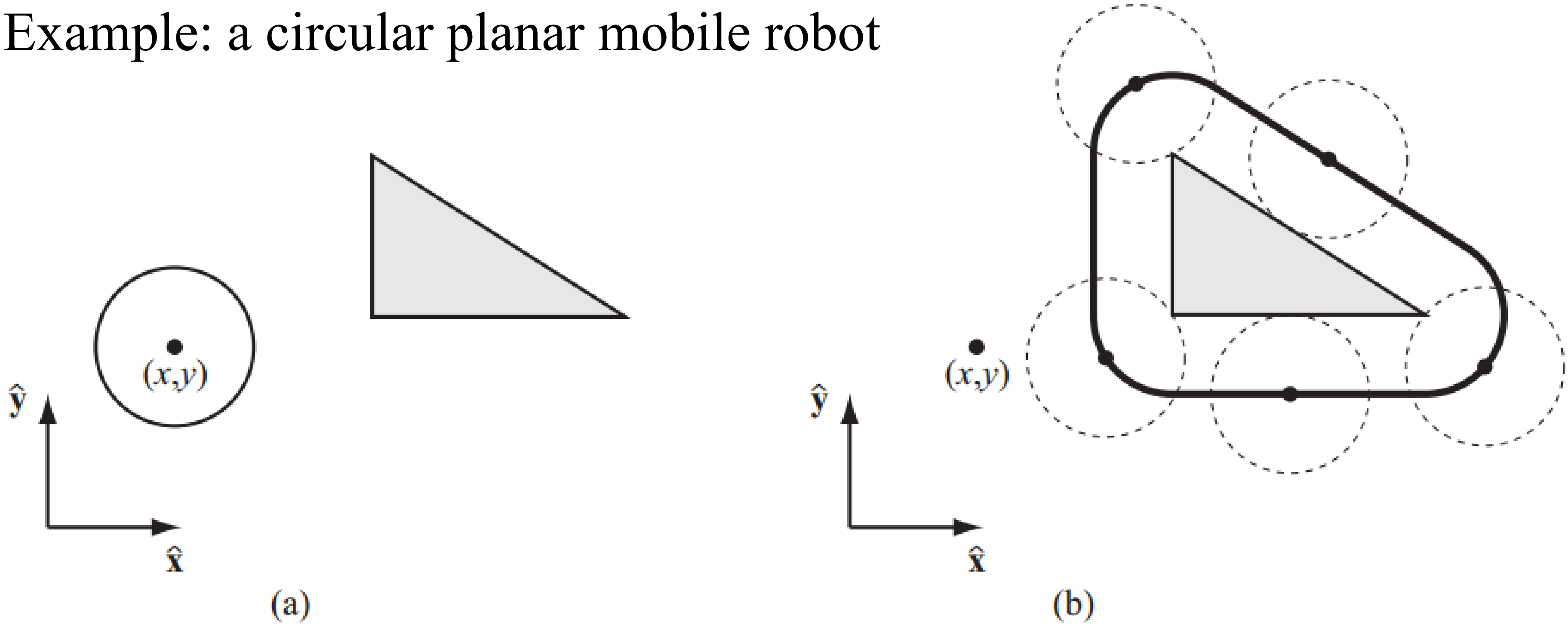


Figure 10.3: (a) A circular mobile robot (white) and a workspace obstacle (grey). The configuration of the robot is represented by (x, y), the center of the robot. (b) In the C-space, the obstacle is "grown" by the radius of the robot and the robot is treated as a point. Any (x, y) configuration outside the dark boundary is collision-free.

Example: a polygonal planar mobile robot

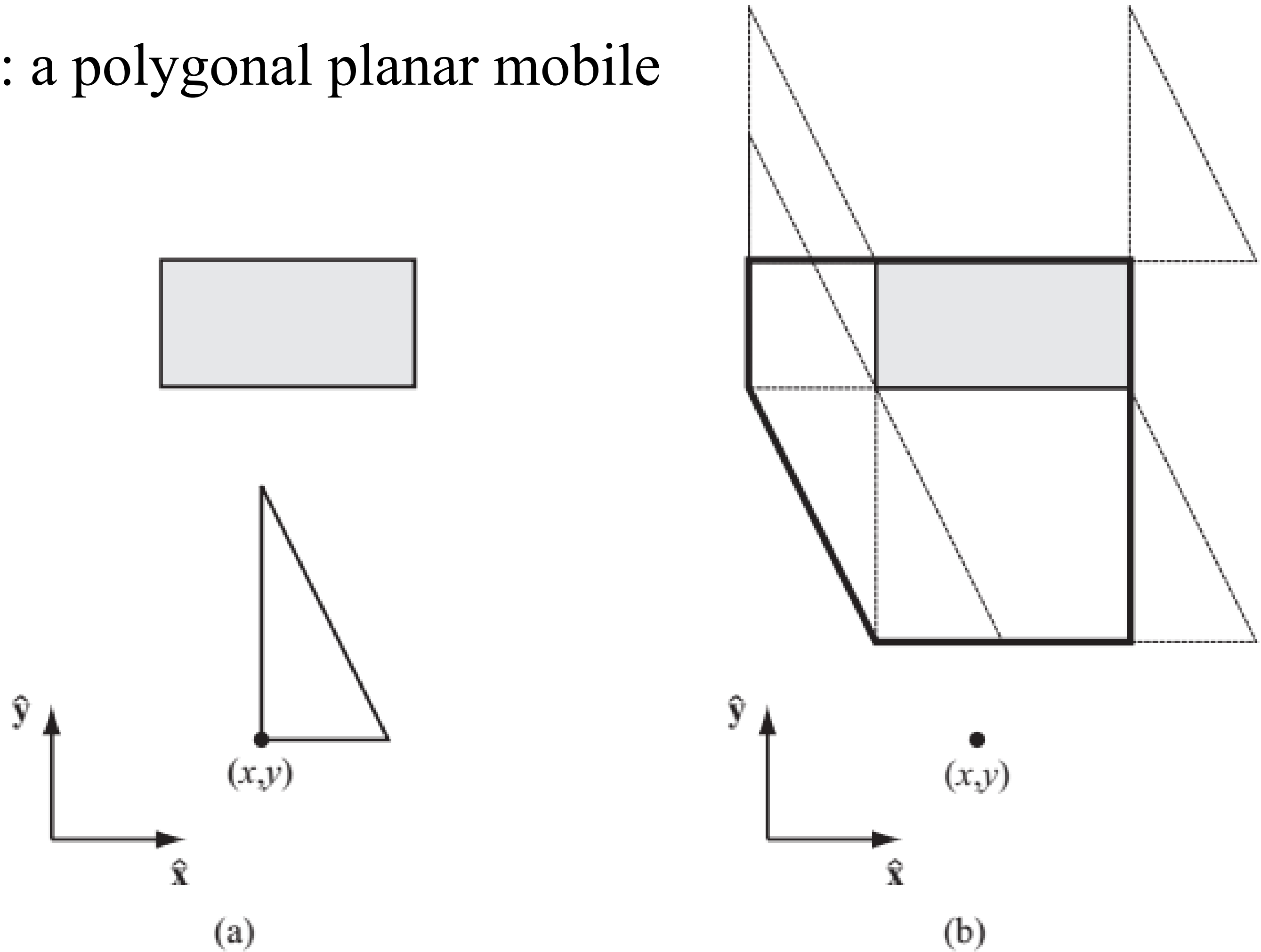


Figure 10.5: (a) The configuration of a triangular mobile robot, which can translate but not rotate, is represented by the (x, y) location of a reference point. Also shown is a workspace obstacle in grey. (b) The corresponding C-space obstacle is obtained by sliding the robot around the boundary of the obstacle and tracing the position of the reference point.

Example: a polygonal planar mobile robot that rotates

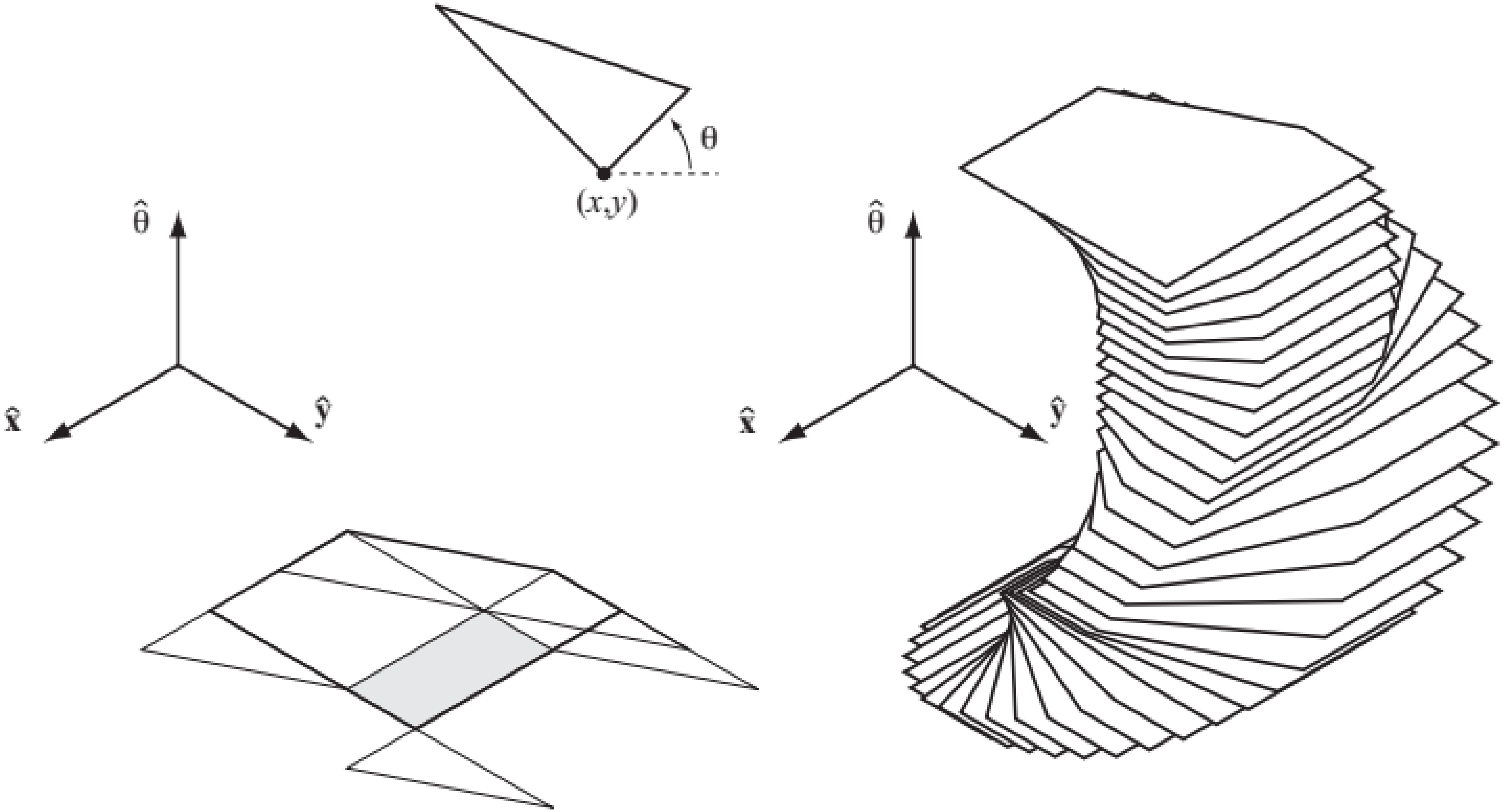


Figure 10.6: (Top) A triangular mobile robot that can rotate and translate, represented by the configuration (x, y, θ). (Left) The C-space obstacle from Figure 10.5(b) when the robot is restricted to $\theta = 0$. (Right) The full 3-D C-space obstacle shown in slices at $10°$ increments.

Planning Methods

Complete methods

Focus on forming a connected-graph representation of the C_{free} and then searching the graph for collision-free path.

Grid methods

Discretize C_{free} into a grid and search the grid for a motion from q_{start} to a grid point in the goal region.

Sampling methods

Rely on random or deterministic functions to choose a sample from C; functions to determine the "closest" previous free-space sample; and a local planner to try to connect to, or move to the new sample.

Virtual potential fields

Create forces on the robot that pull it toward the goal and push it away from obstacles.

Complete Path Planners

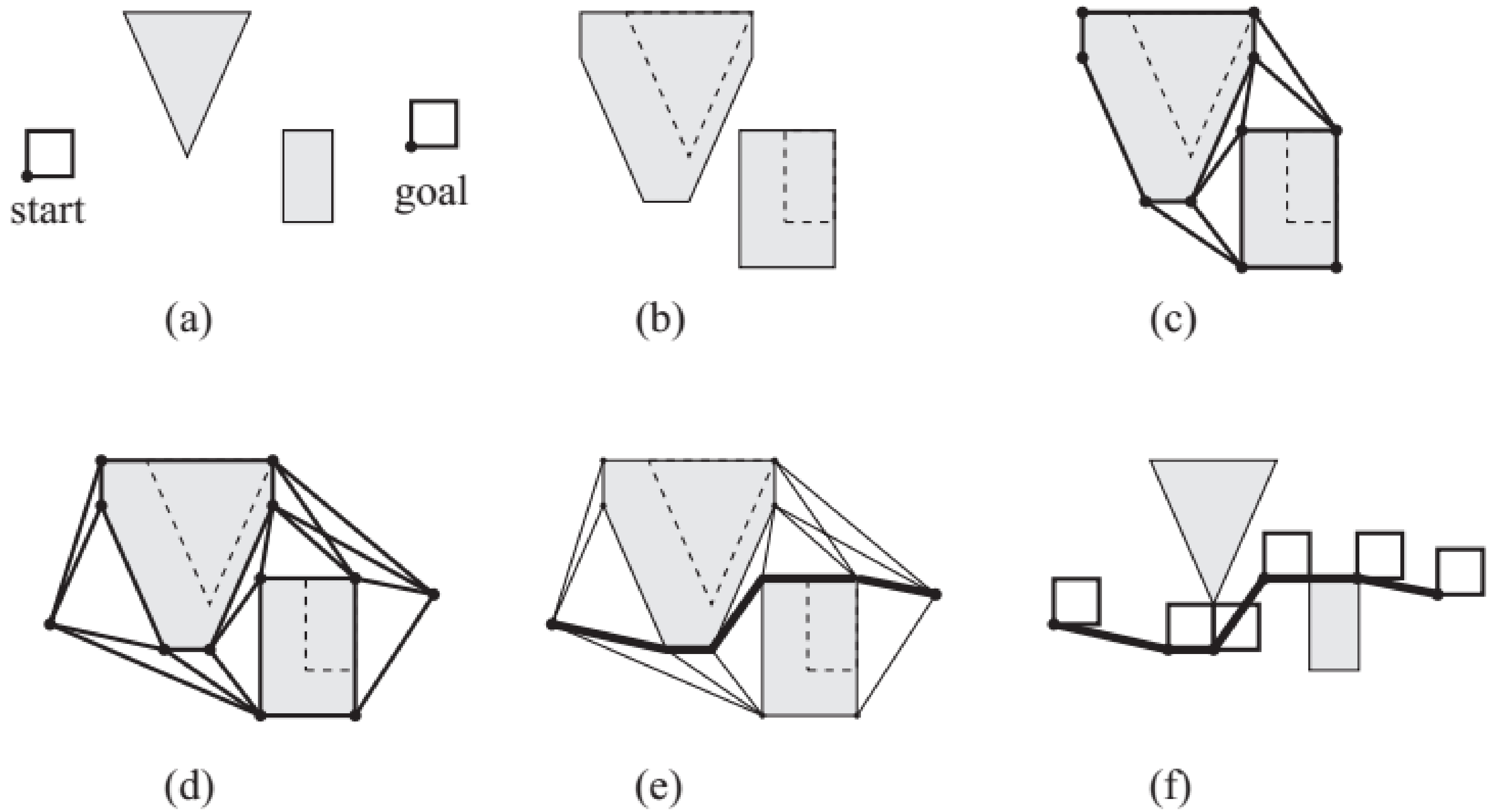


Figure 10.9: (a) The start and goal configurations for a square mobile robot (reference point shown) in an environment with a triangular and a rectangular obstacle. (b) The grown C-obstacles. (c) The visibility graph roadmap R of $\mathcal{C}_{\text{free}}$. (d) The full graph consists of R plus nodes at q_{start} and q_{goal}, along with the links connecting these nodes to visible nodes of R. (e) Searching the graph results in the shortest path in bold. (f) The robot traversing the path.

Complete Path Planners

A* search

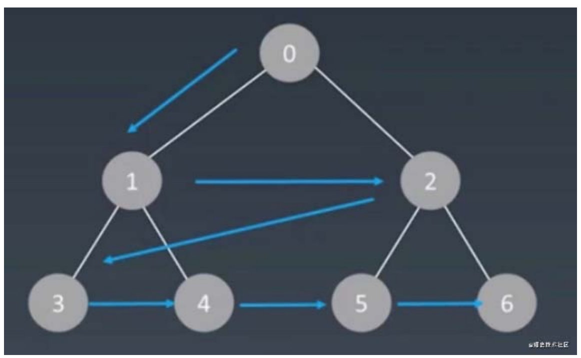
Suboptimal A* search

Dijkstra's method

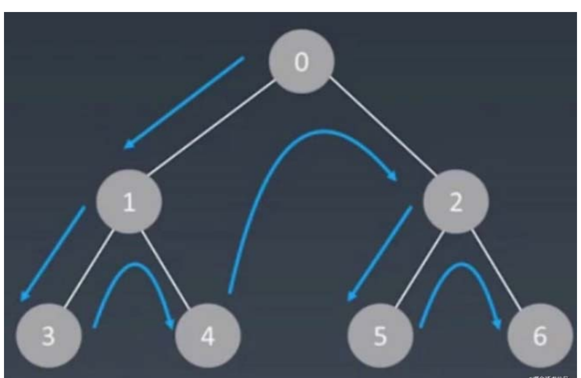
Best-first search

Breadth-first search (BFS)

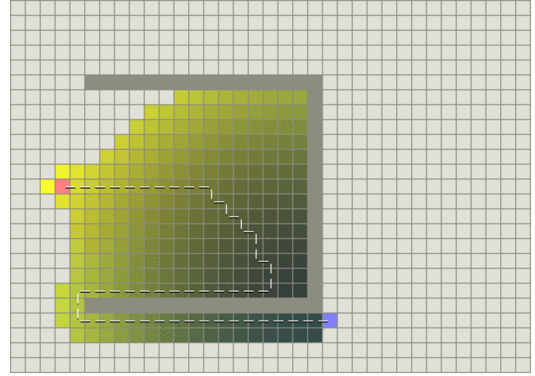
Depth-first search (DFS)



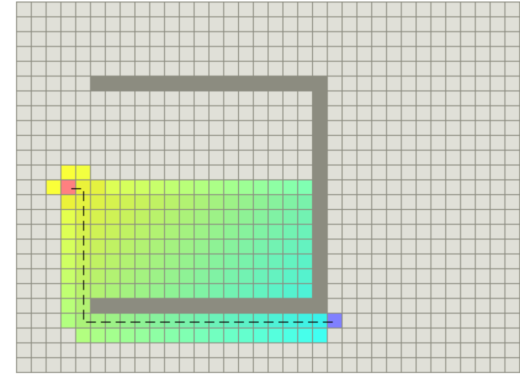
BFS



DFS



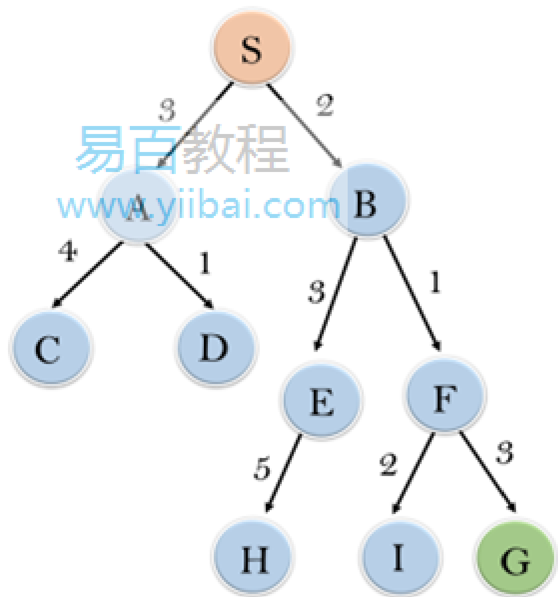
Best-first search result



A* search result

评价函数f(n)=h(n)

h(n): 节点n到目标的估计成本



node	H (n)
A	12
B	4
C	7
D	3
E	8
F	2
H	4
I	9
S	13
G	0

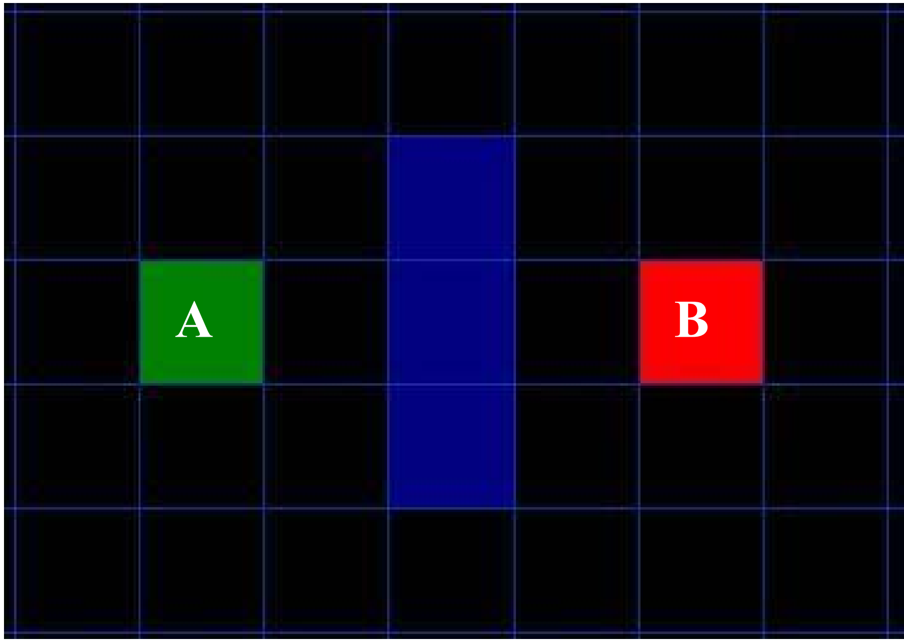
Best first search

A* search

Refer to "Patrick Lester. A* Pathfinding for Beginners"

a graph, nodes N={1, …, N}, edges.

Similar and simpler example:

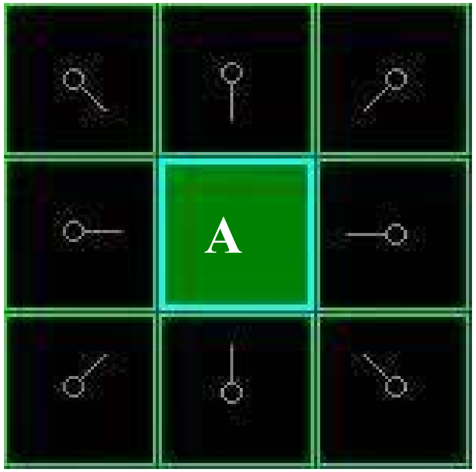


Initialization:

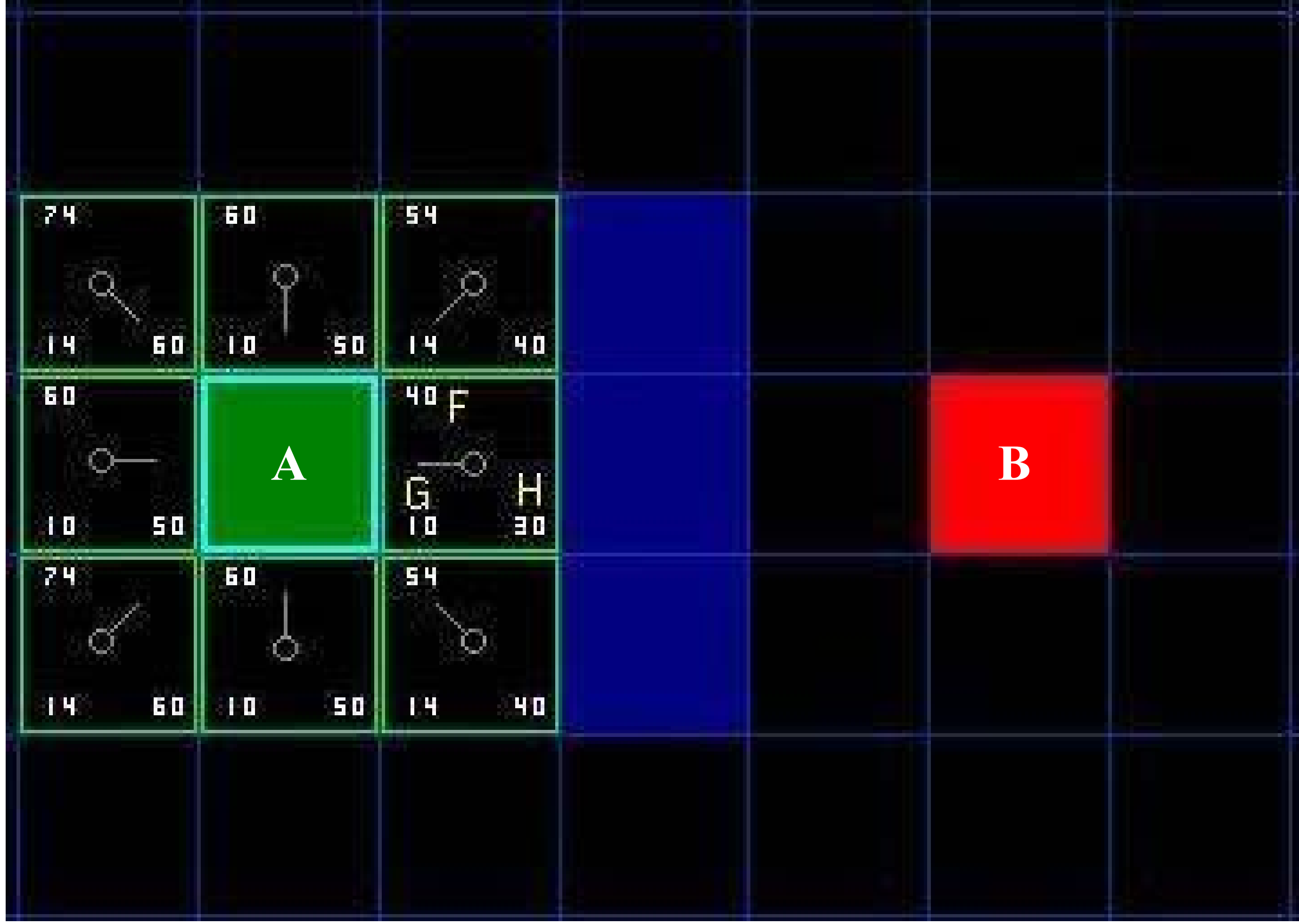
green filled square: the starting point A
red filled square: the ending point B
blue filled squares: the wall in between

Starting the search:

- Add A to an "open list" .
- Look at all squares adjacent to A, ignoring squares with walls. Add them to the "open list", too. For each of these squares, save A as its "parent square". (outlined in light green)
- Drop A from your "open list", and add it to a "closed list". (outlined in light blue)



Here we assign a cost of 10 to each horizontal or vertical square moved, and a cost of 14 for a diagonal move.



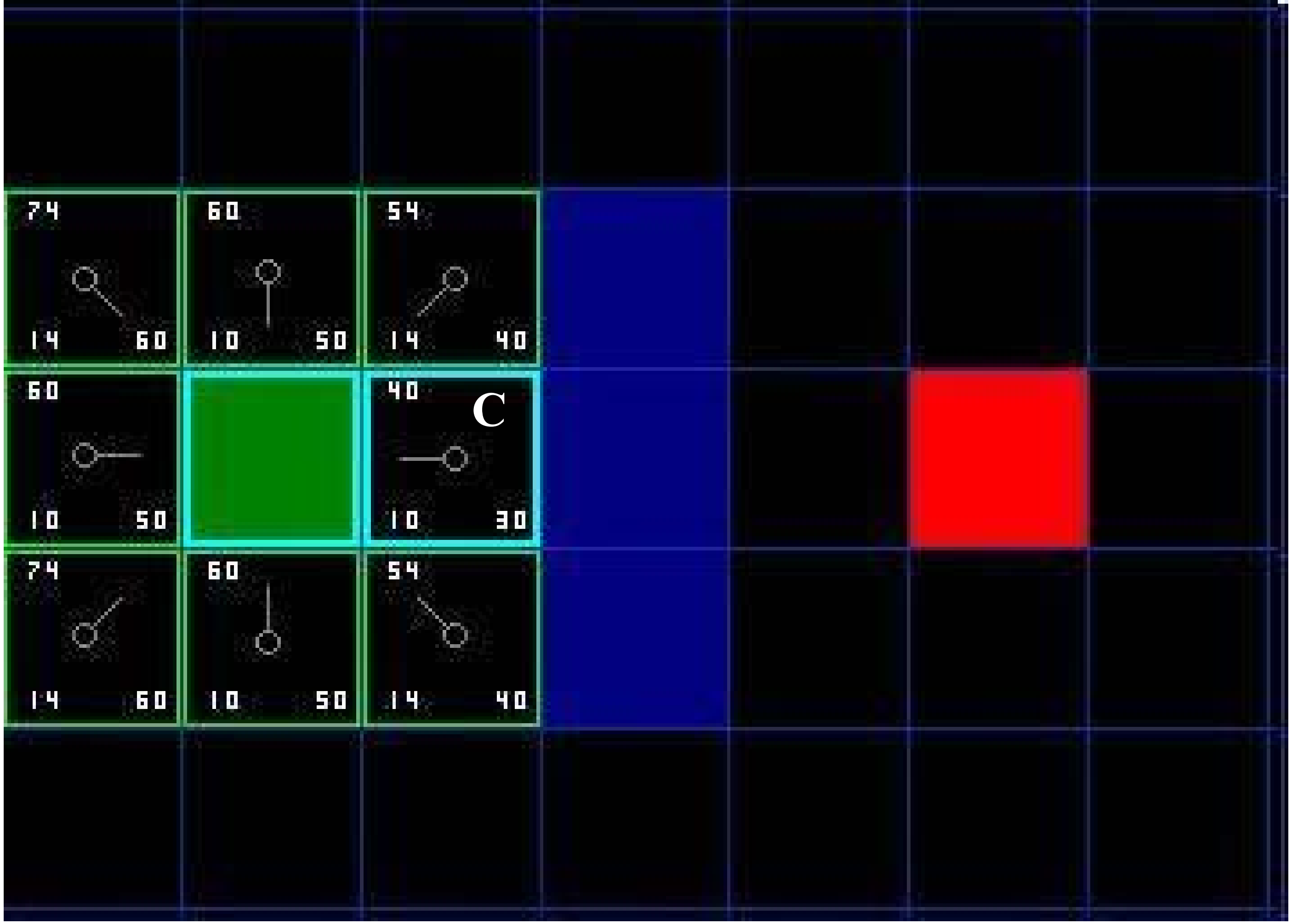
For all the adjacent squares on the "open list", calculate F cost:

$$F = G + H$$

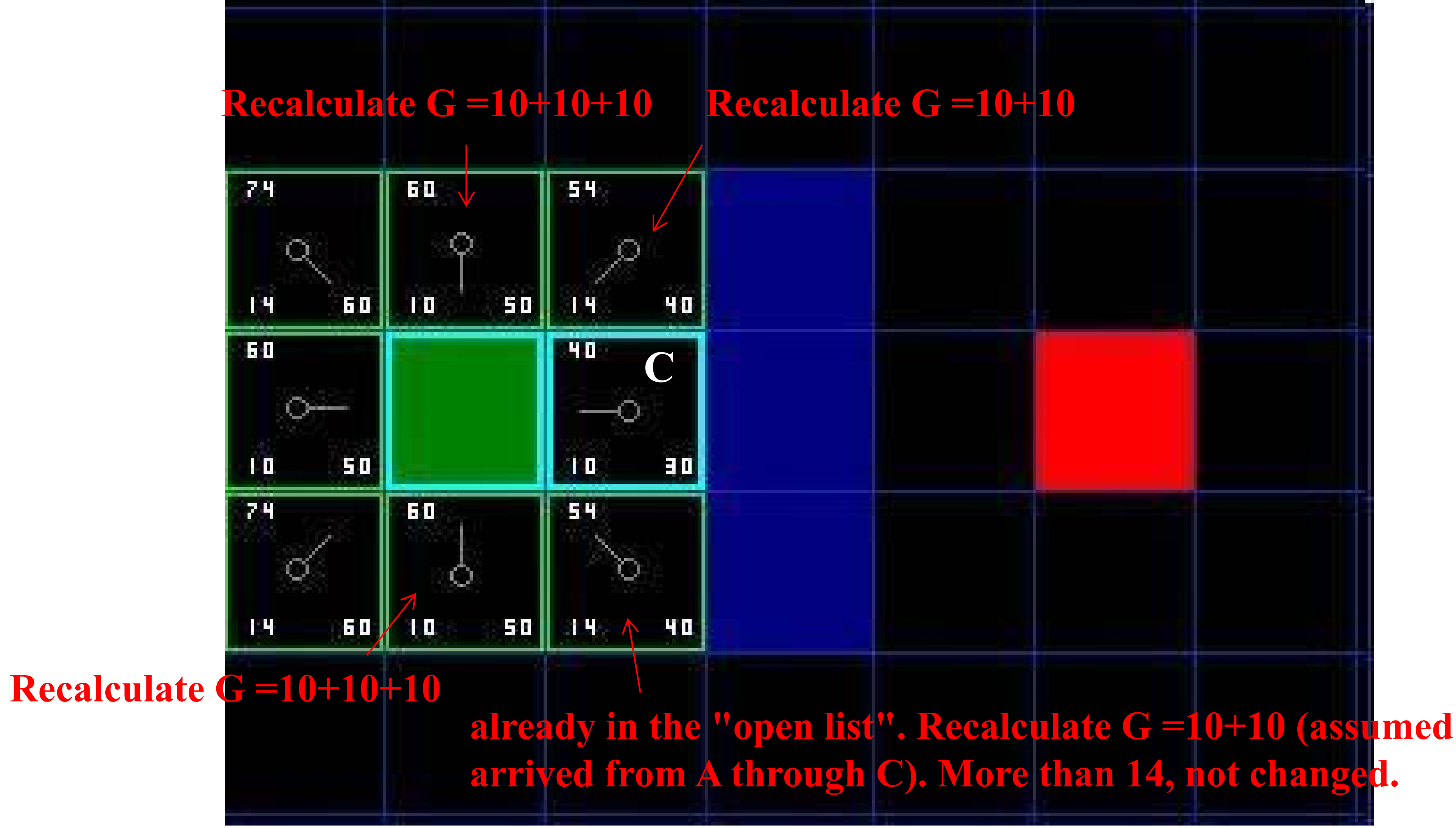
where

G = cost to move from A to a given square on the grid, following the path generated to get there.

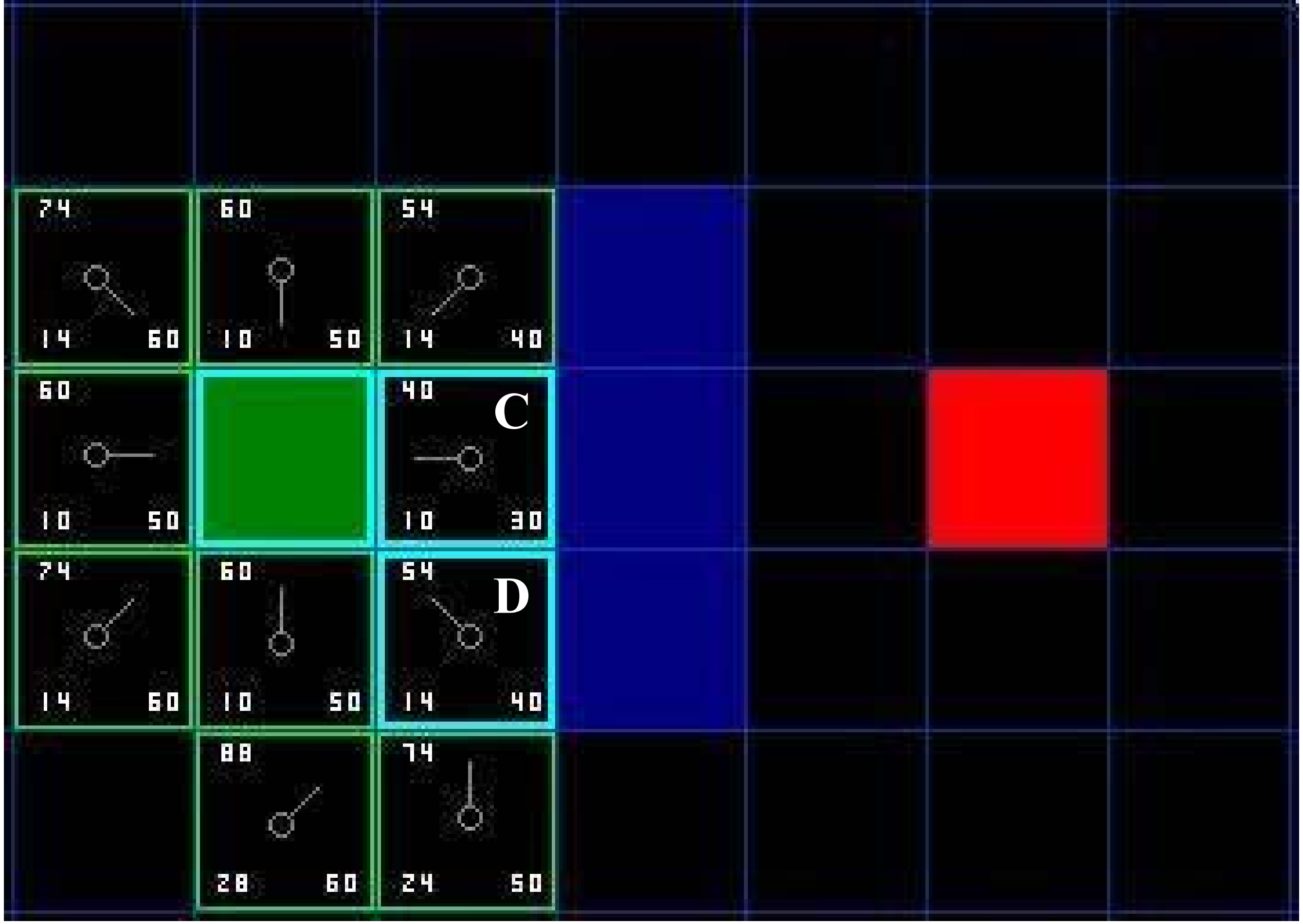
H = the estimated cost to move from that given square to B. Often referred to as the heuristic. Manhattan distance here.



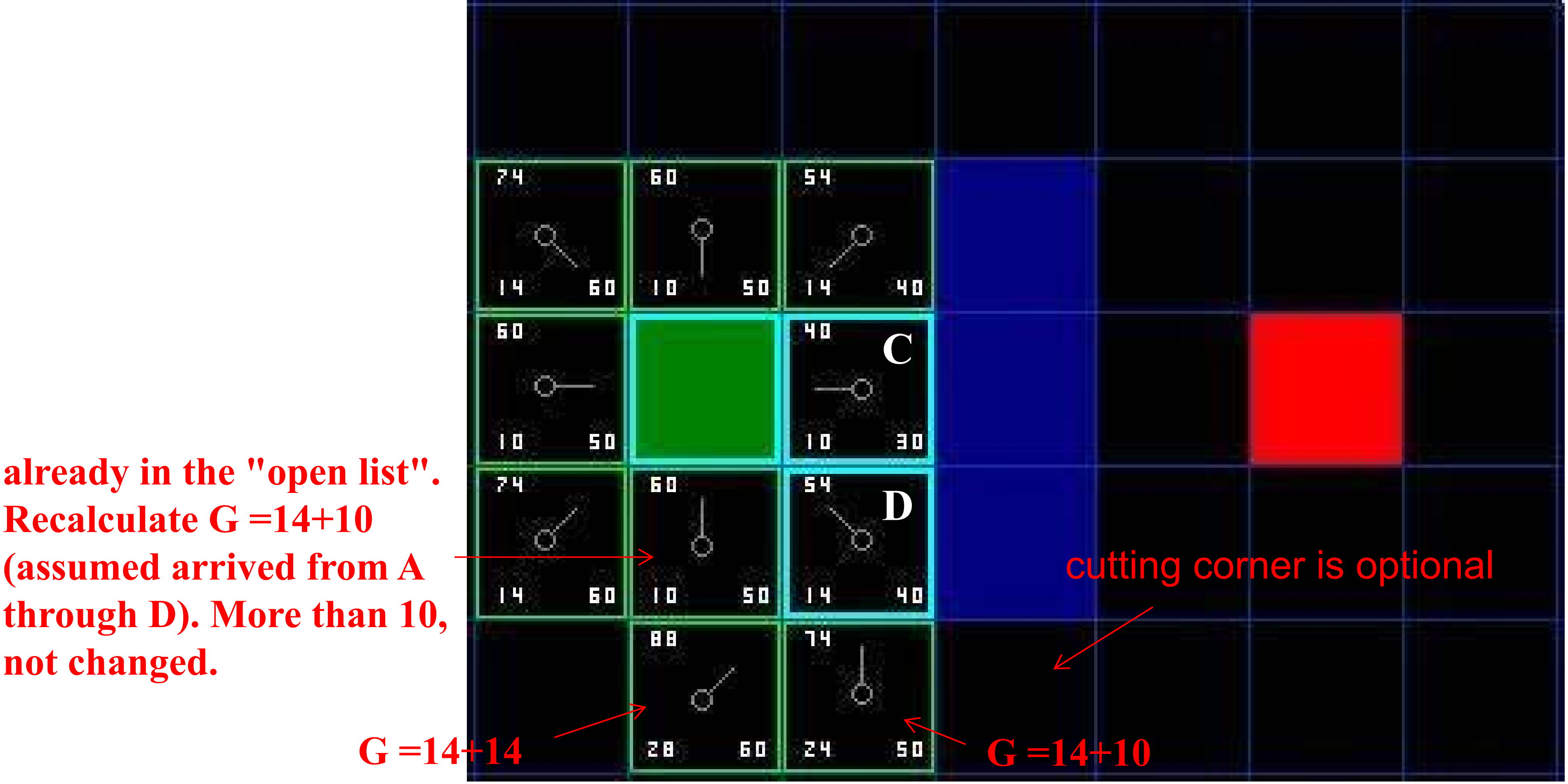
- Choose the lowest F score square from all those that are on the "open list". Named Point C here.
- Drop it from the "open list" and add it to the closed list. (outlined in light blue)



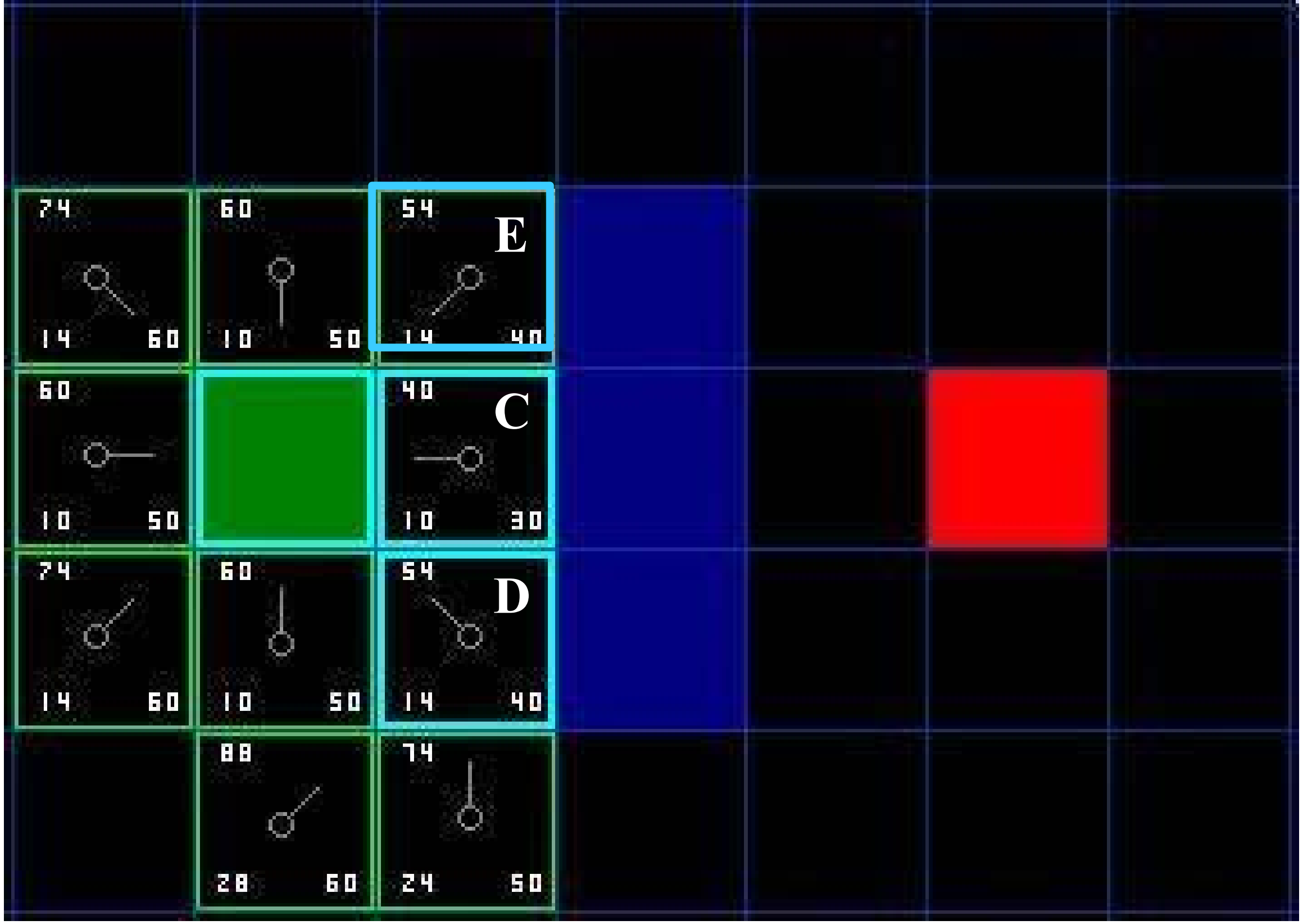
- Check all of the adjacent squares of C, ignoring those that are on the "closed list" or walls, add squares to the "open list" if they are not on the "open list" already. Make the selected square C the "parent" of the new squares.
- If an adjacent square is already on "the open list", check to see if the G score for that square is lower if we use the current square C to get there. If so, change the parent of the adjacent square to the selected square. Recalculate both F and G of that square.



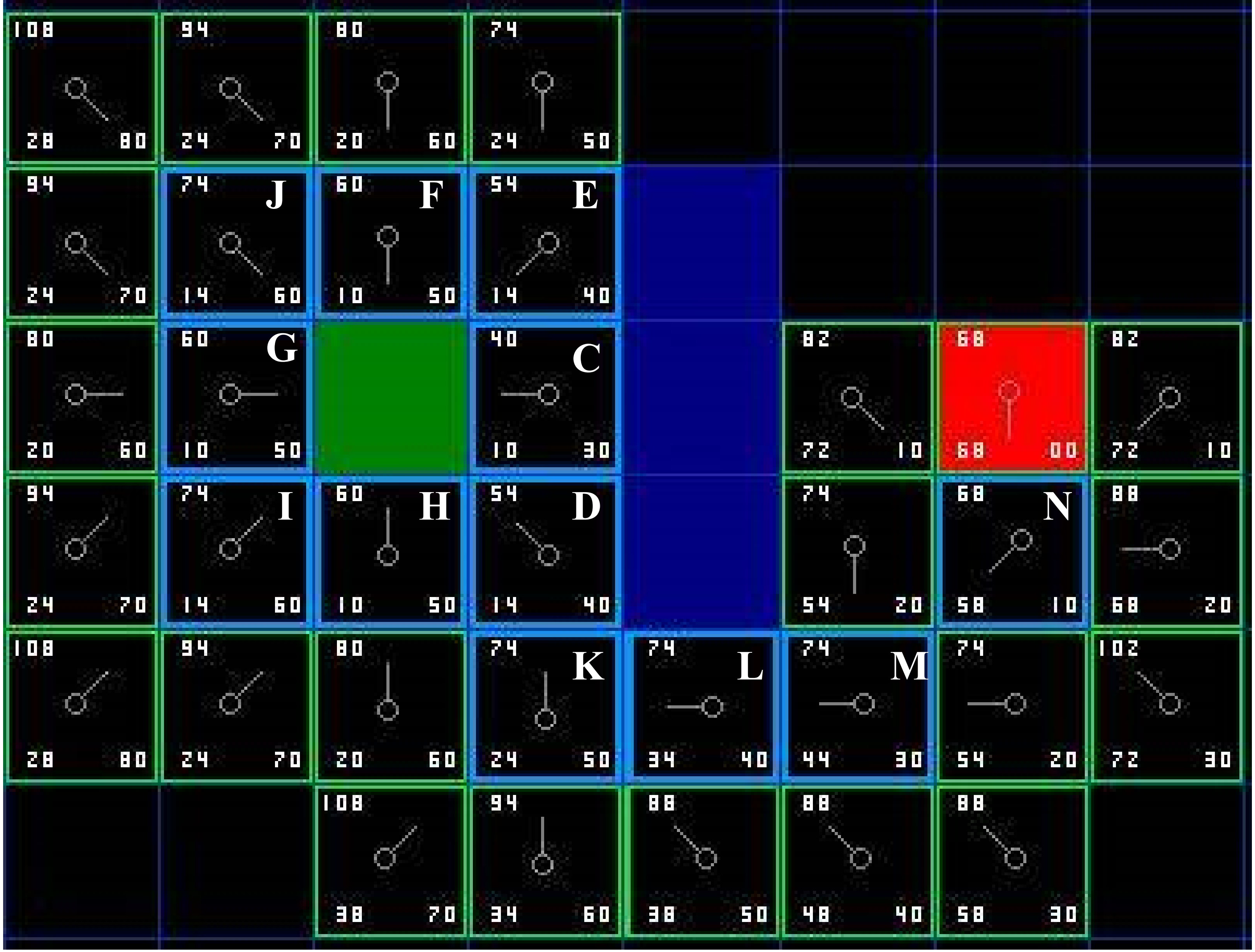
- Choose the lowest F score square from all those that are on the "open list" (outlined in light green). Named Point D here.
- Drop it from the "open list" and add it to the "closed list". (outlined in light blue)



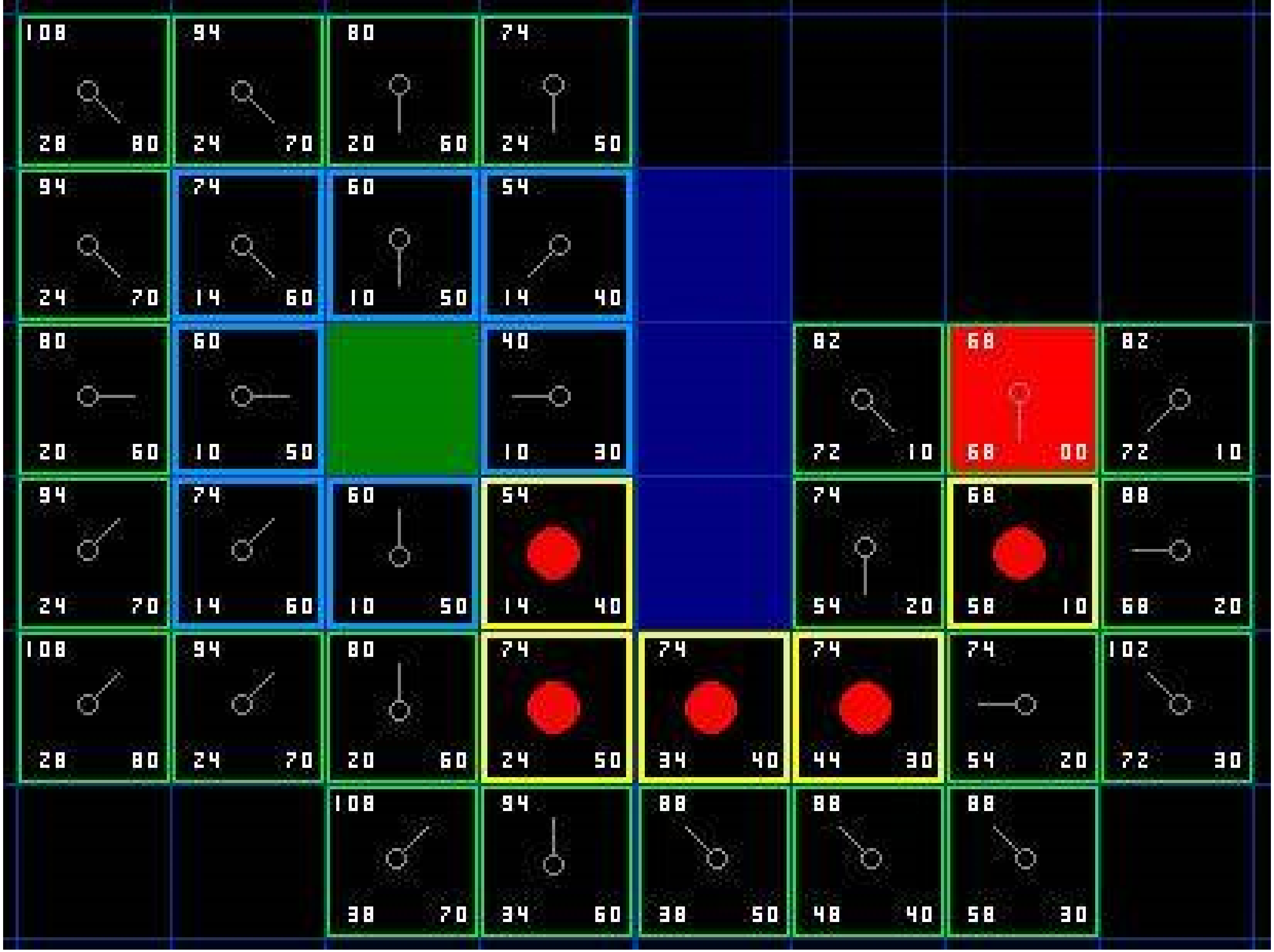
- Check all of the adjacent squares of D, ignoring those that are on the "closed list "or walls, add squares to the "open list" if they are not on the "open list" already. Make the selected square D the "parent" of the new squares.
- If an adjacent square is already on the open list, check to see if the G score for that square is lower if we use the current square D to get there. If so, change the parent of the adjacent square to the selected square. Recalculate both the F and G scores of that square.



- Choose the lowest F score square from all those that are on the "open list" (outlined in light green). Named Point E here.
- Drop it from the "open list" and add it to the "closed list". (outlined in light blue)



Repeat this process until we add the target square to the closed list, at which point it looks something like the illustration.



To determine the path - Simply start at the red target square, and work backwards moving from one square to its parent, following the arrows.

More about heuristic

The closer our estimate is to the actual remaining distance, the faster the algorithm will be.

If we overestimate this distance, however, it is not guaranteed to give us the shortest path. In such cases, we have what is called an 'inadmissible (不可采纳的) heuristic.'

Technically, in this example, the Manhattan method is inadmissible because it slightly overestimates the remaining distance. But we will use it anyway because it is a lot easier to understand for our purposes, and because it is only a slight overestimation. On the rare occasion when the resulting path is not the shortest possible, it will be nearly as short.

-Further discussion in "Patrick Lester. Heuristics and A* Path finding".

A算法：采用f(n)=g(n)+h(n)作为评价函数的最佳优先搜索算法。

A*算法：h(n) ≤ 从n到目标的最短路径。

A*算法是可采纳的(能找到最短路径)。

-Luger, G. F. Artificial Intelligence: Structures and Strategies for Complex Problem Solving (6th ed), 2009.

Given a graph described by a set of nodes $\mathcal{N}=\{1, \ldots, N\}$, where node 1 is the start node, and a set of edges ε, A* makes use of the following data structures:

A sorted list **OPEN** of the nodes to be explored from;

A list **CLOSED** of nodes that have already been explored from;

An array **past_cost[node]** of the minimum cost found so far to reach node **node** from the start node;

A search tree defined by an array **parent[node]**, which contains a link for each node to the node preceding it in the shortest path found so far to node.

Algorithm 1 A^* search.

```
OPEN ← {1}
past_cost[1] ← 0, past_cost[node] ← infinity for node ∈ {2, . . . , N}
while OPEN is not empty do
   current ← first node in OPEN, remove from OPEN
   add current to CLOSED
   if current is in the goal set then
      return  SUCCESS and the path to current
   end if
   for each nbr of current not in CLOSED do
      tentative_past_cost ← past_cost[current] + cost[current,nbr]
      if tentative_past_cost < past_cost[nbr] then
         past_cost[nbr] ← tentative_past_cost
         parent[nbr] ← current
         put (or move) nbr in sorted list OPEN according to
               est_total_cost[nbr] ← past_cost[nbr] +
                     heuristic_cost_to_go(nbr)
      end if
   end for
end while
return  FAILURE
```

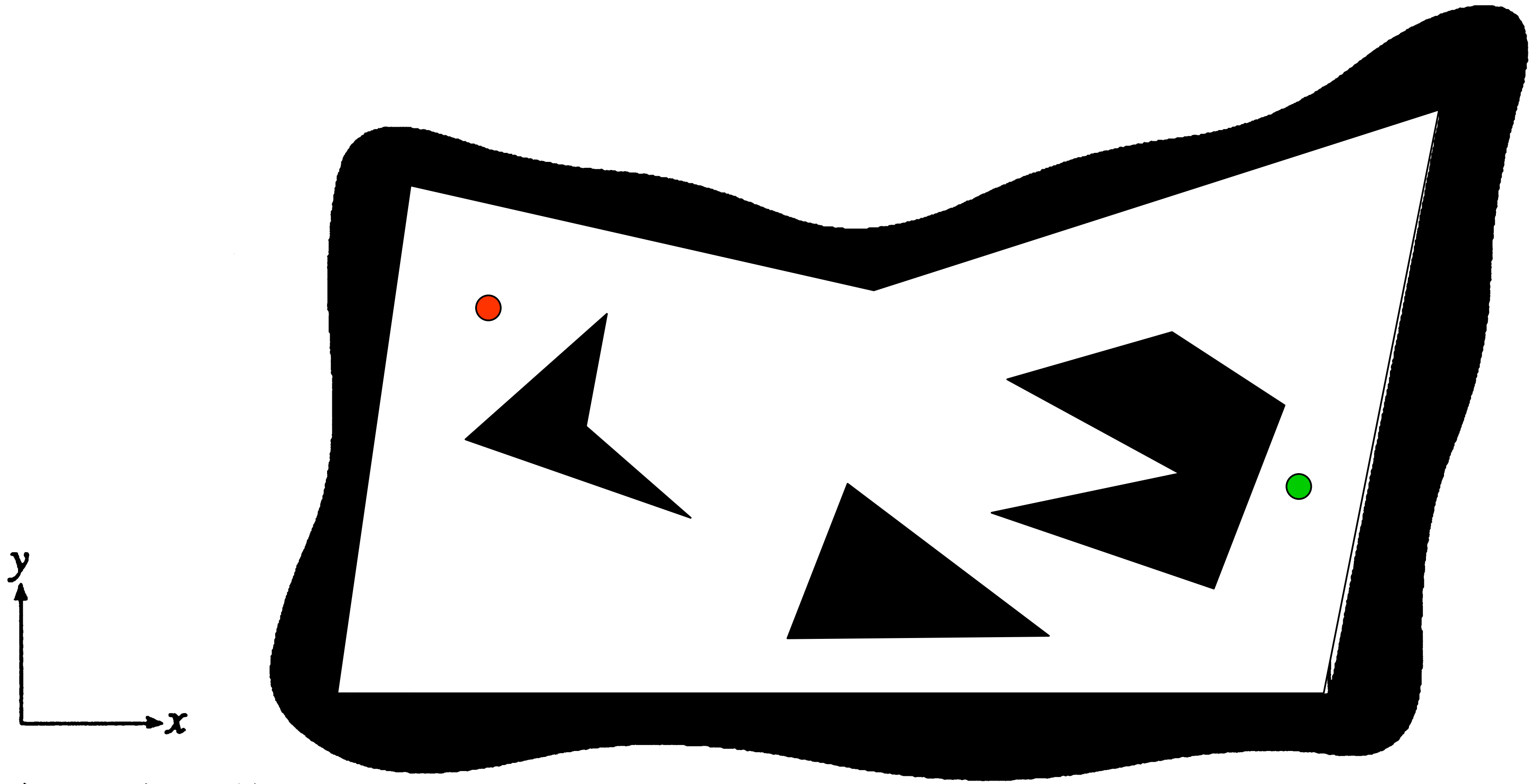
Cell-Decomposition Methods

Two classes of methods:

- Exact cell decomposition
  The free space F is represented by a collection of non-overlapping cells whose union is **exactly** F
  Example: trapezoidal decomposition
- Approximate cell decomposition

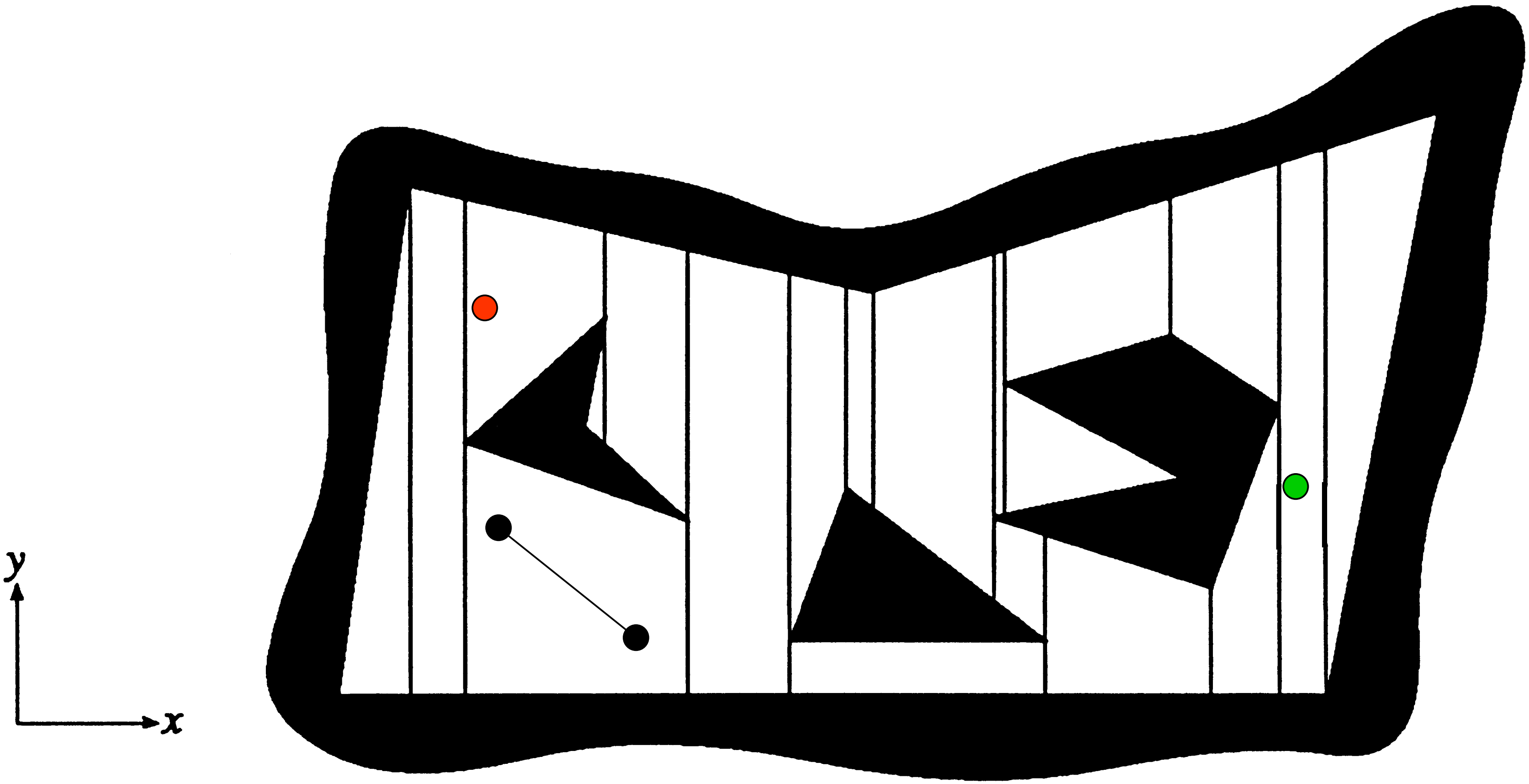
Refer to "Introduction to Autonomous Mobile Robots" for details

Trapezoidal decomposition



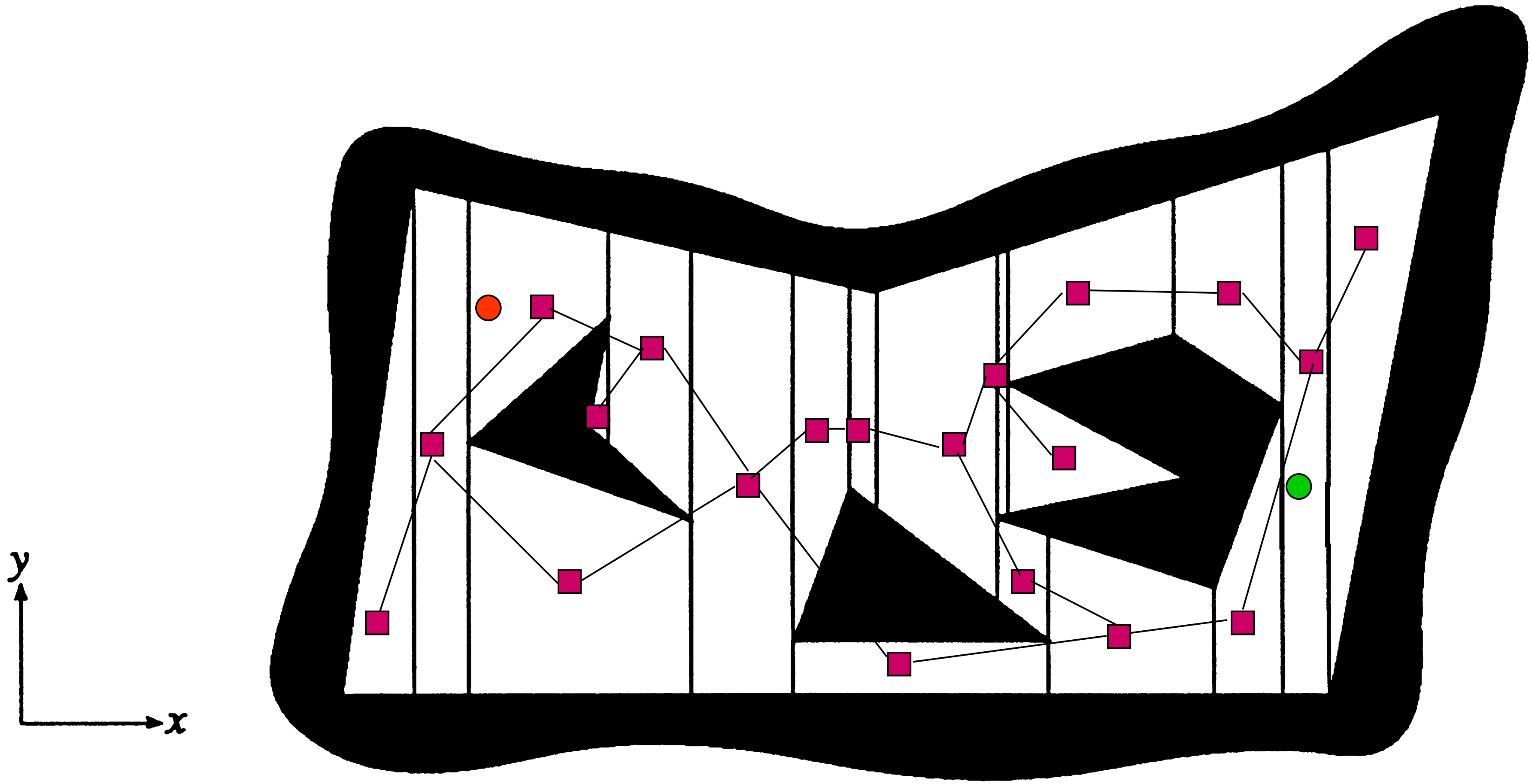
黑色：障碍物
橙色：出发点
绿色：目标点

Trapezoidal decomposition



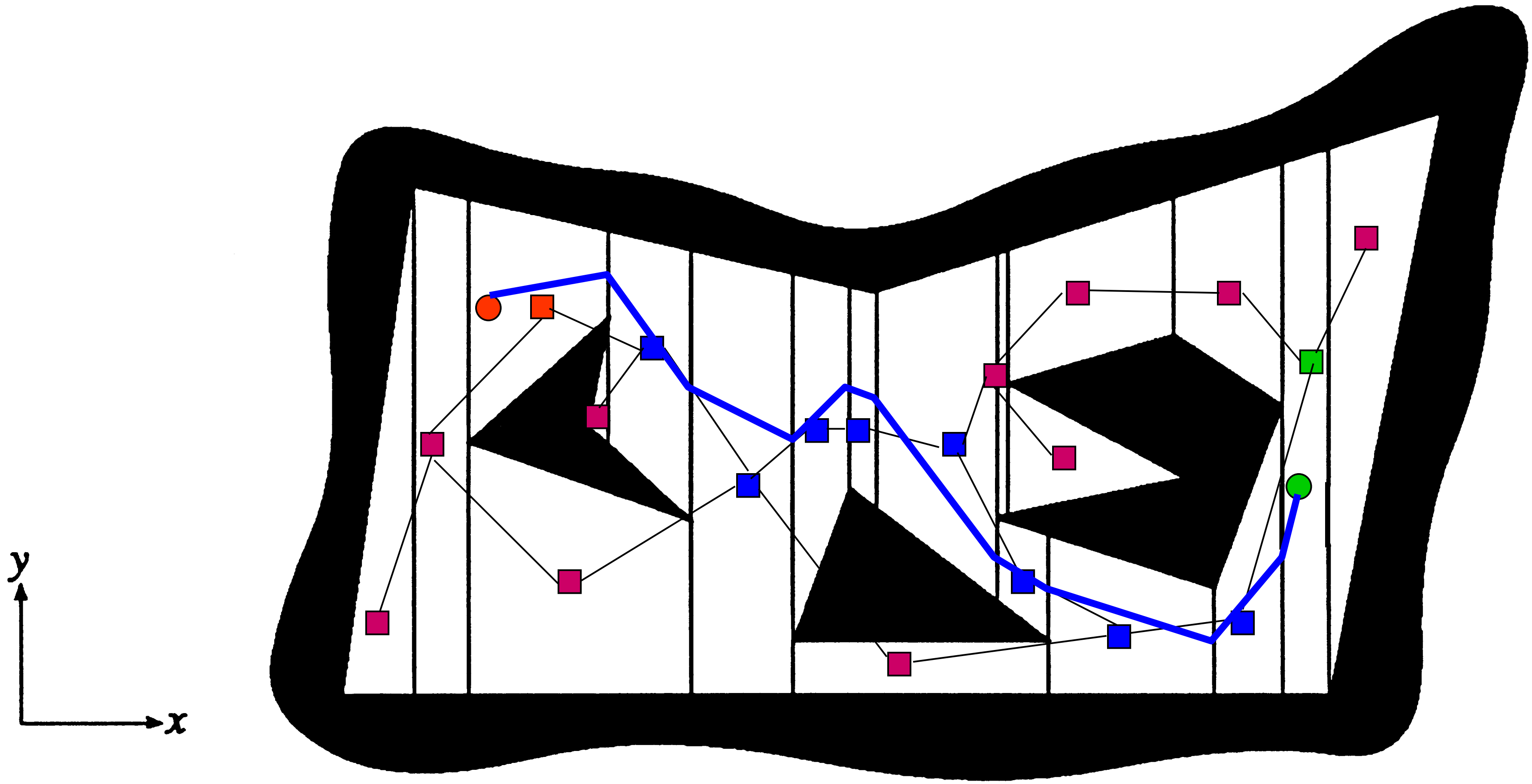
障碍物的角点纵向投射直线，形成多个梯形的**cell**

Trapezoidal decomposition



各个相邻的cell连接，粉色代表各cell的中心

Trapezoidal decomposition



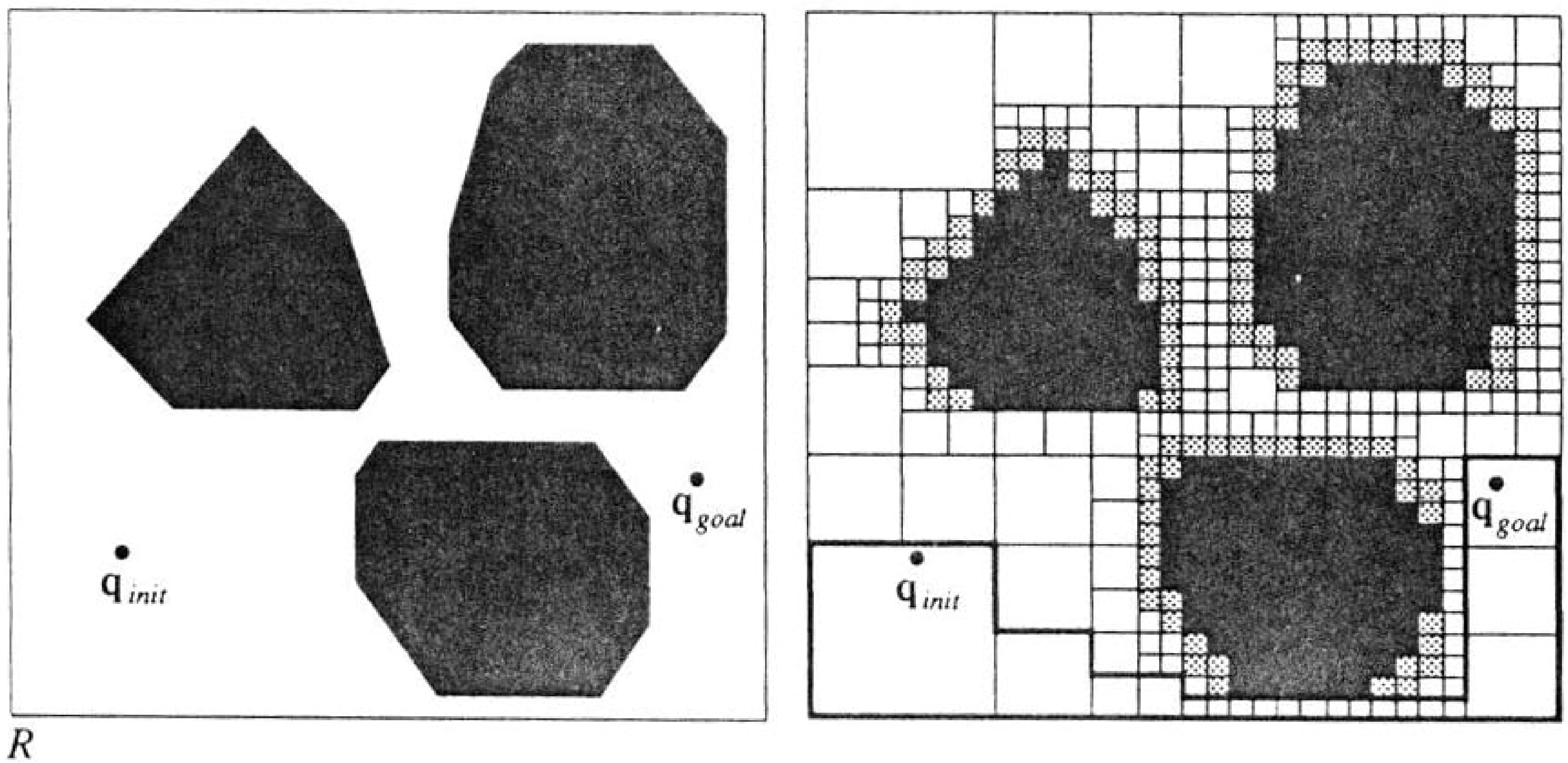
按一定的准则，从中寻找合适的路径

Cell-Decomposition Methods

Two classes of methods:

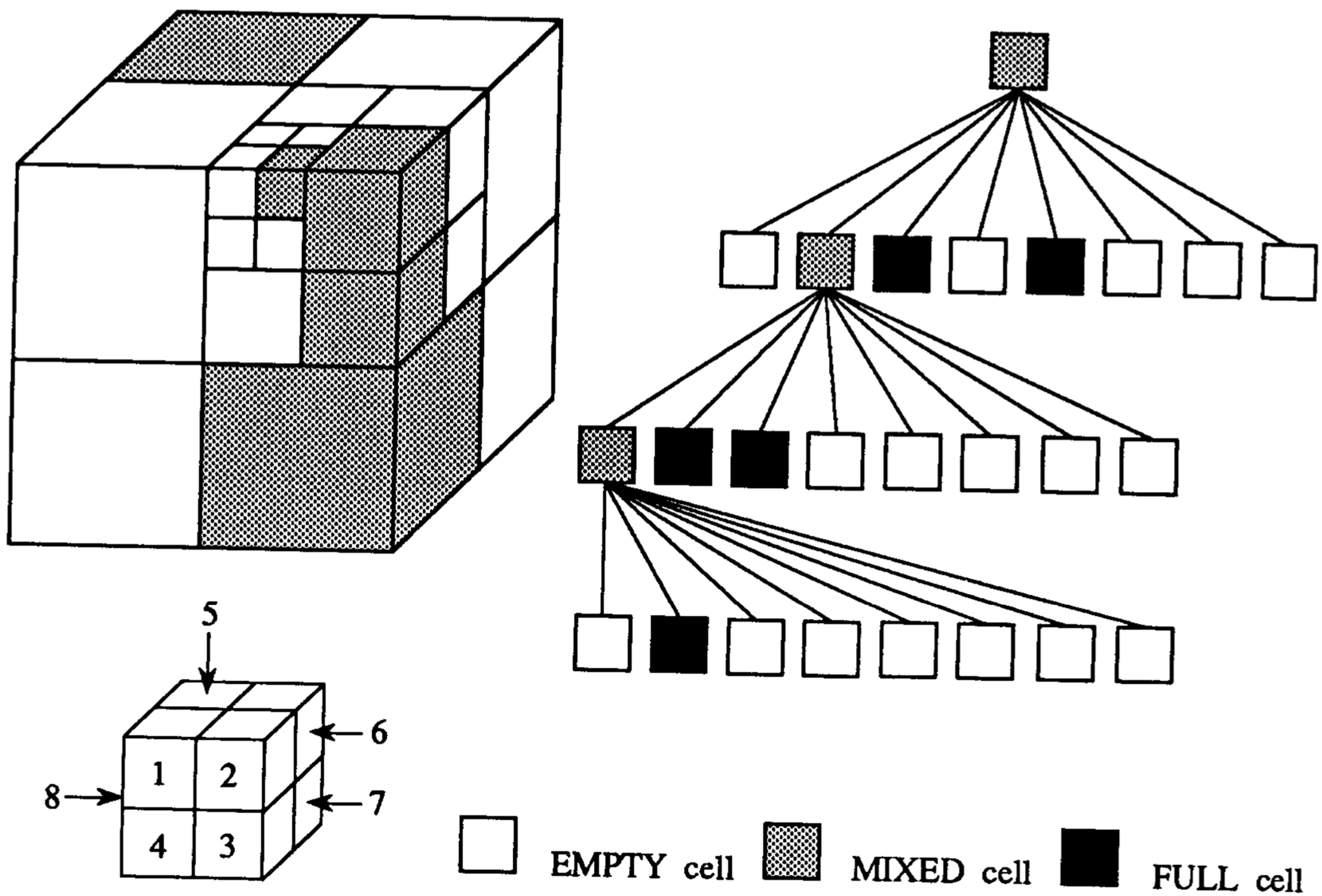
- Exact cell decomposition
- Approximate cell decomposition
  The free space F is represented by a collection of non-overlapping cells whose union is contained in F.
  Examples: quadtree, octree, 2^n-tree

Approximate Cell Decomposition: Quad Tree



(a) (b)

Octree Decomposition



Sketch of Algorithm

1. Compute cell decomposition down to some resolution
2. Identify start and goal cells
3. Search for sequence of empty/mixed cells between start and goal cells
4. If no sequence, then exit with no path
5. If sequence of empty cells, then exit with solution
6. If resolution threshold achieved, then exit with failure
7. Decompose further the mixed cells
8. Return to 2

Sampling methods

Two major classes:

- rapidly-exploring random trees (RRTs)
- probabilistic roadmaps (PRMs)

a. rapidly-exploring random trees (RRTs)

Searches for a collision-free motion from an initial state (configuration) from an initial state x_{start} to a goal set X_{goal}.

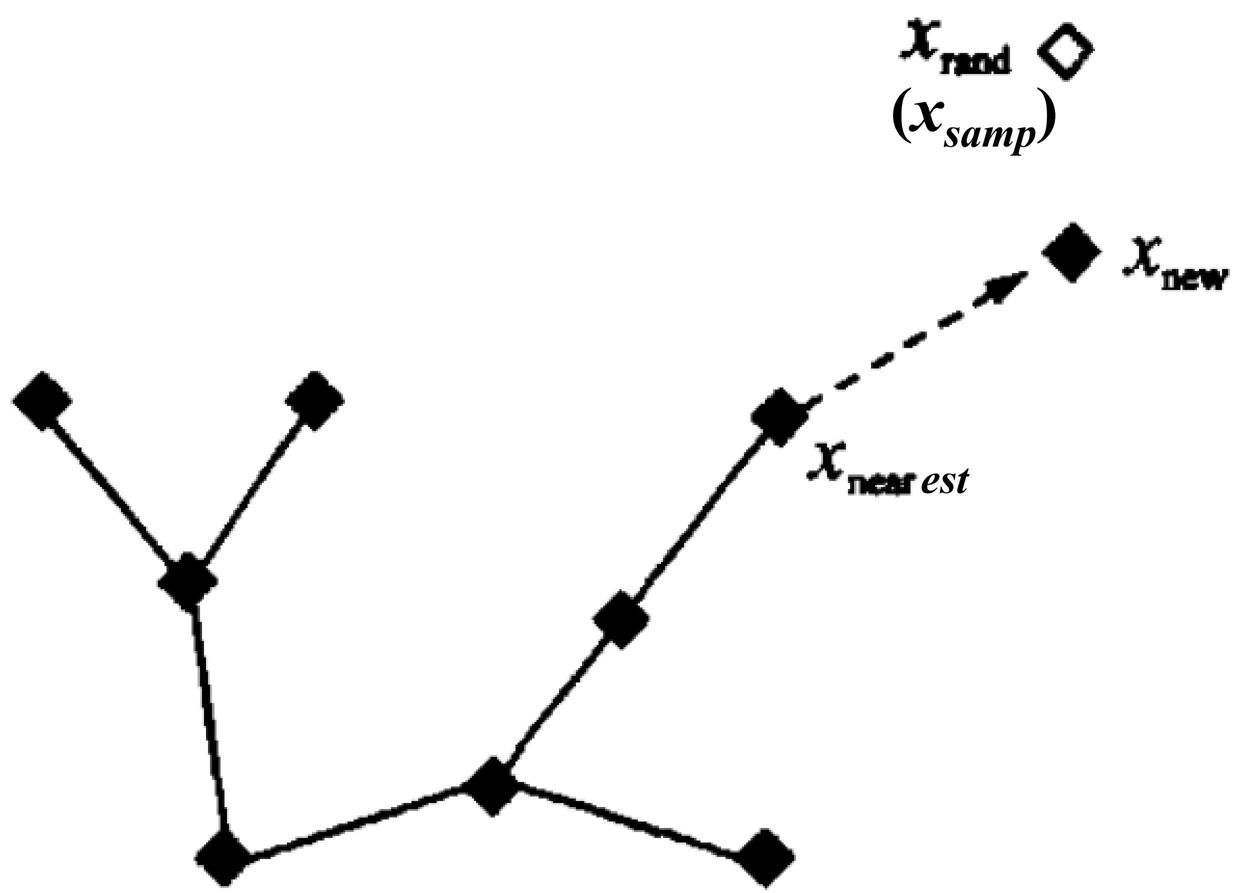
Algorithm 3 RRT algorithm.

initialize search tree T with x_{start}
while T is less than the maximum tree size **do**
    $x_{\text{samp}} \leftarrow$ sample from $\mathcal{X}$
    $x_{\text{nearest}} \leftarrow$ nearest node in T to x_{samp}
    employ a local planner to find a motion from x_{nearest} to x_{new} in the direction of x_{samp}
    if the motion is collision-free **then**
        add x_{new} to T with an edge from x_{nearest} to x_{new}
        if x_{new} is in $\mathcal{X}_{\text{goal}}$ **then**
            return SUCCESS and the motion to x_{new}
        end if
    end if
end while
return FAILURE

Line 3 - the sampler: to choose x_{samp} randomly from an almost-uniform distribution over X, with a slight bias toward states in X_{goal}.

Line 4 - the nearest node: The closest node $x_{nearest}$ in T is the one minimizing the Euclidean distance to x_{samp}.

Line 5 - the local planner: The state x_{new} is chosen as the state at x_{samp} or a small fixed distance d from $x_{nearest}$ on the straight line to x_{samp}. (straight line planner)



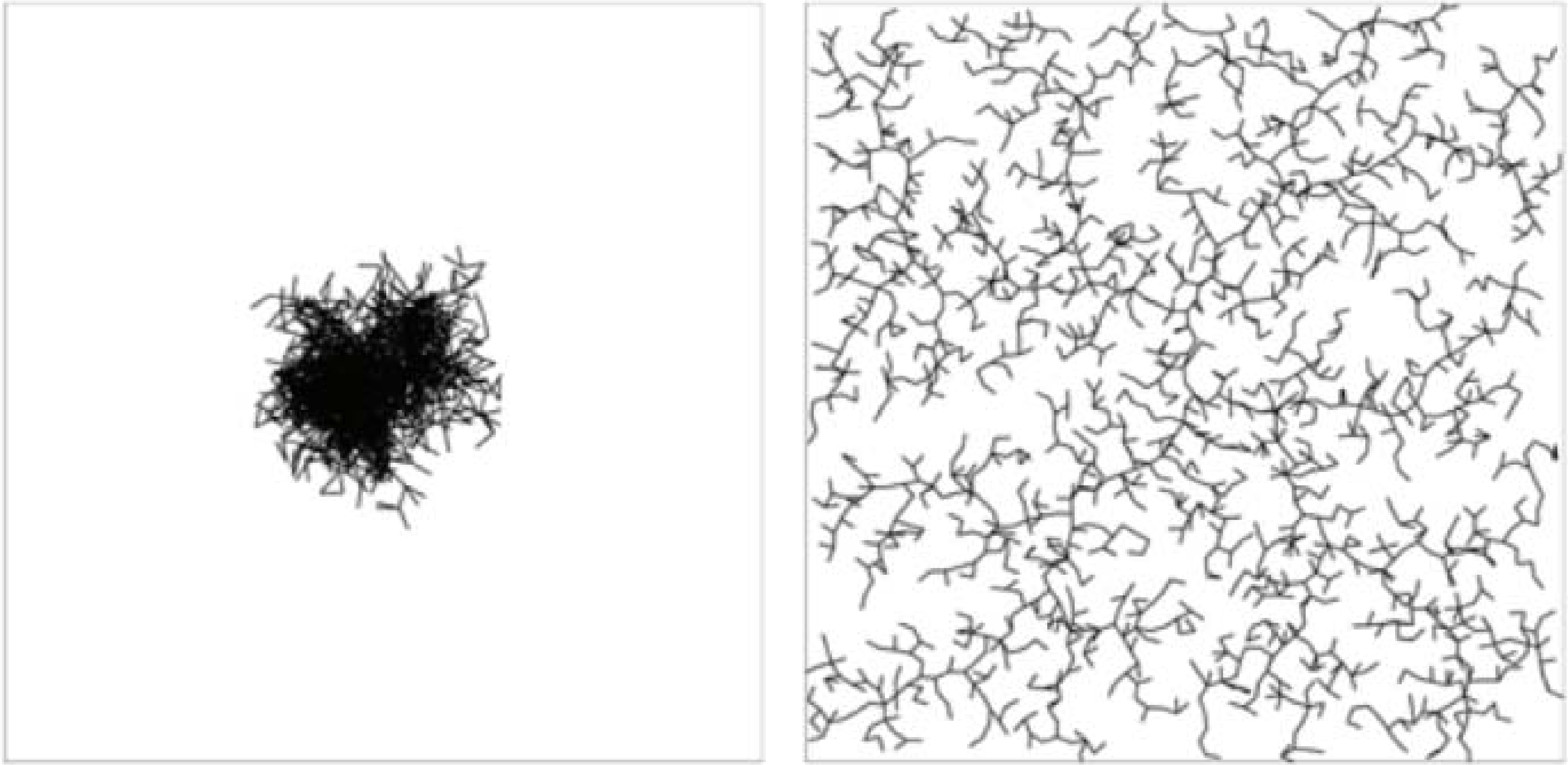
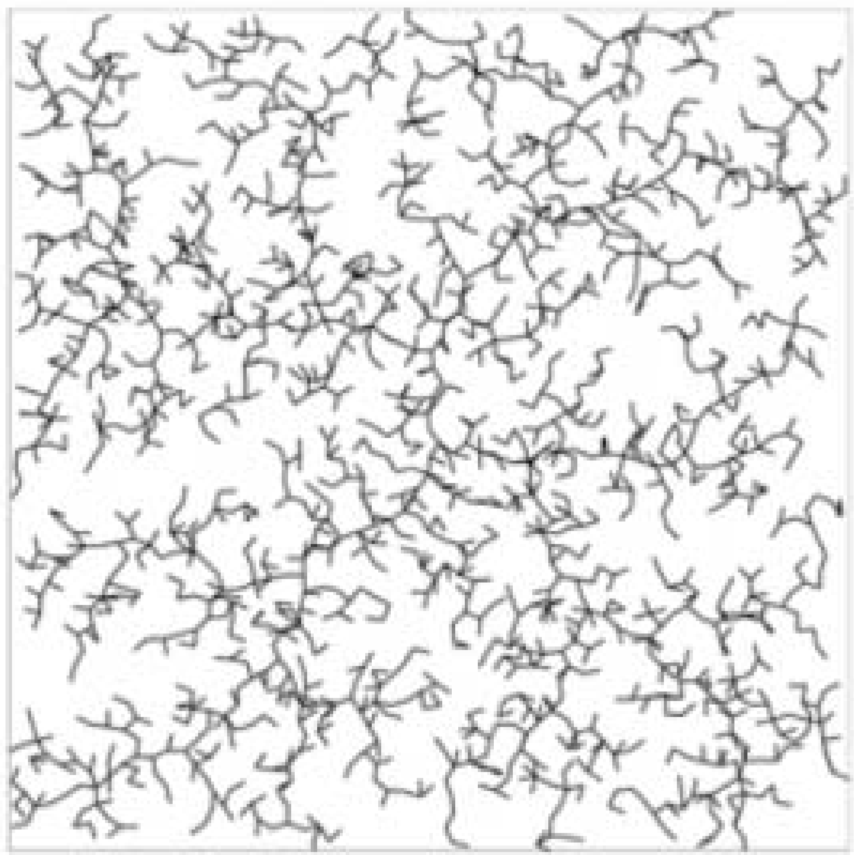
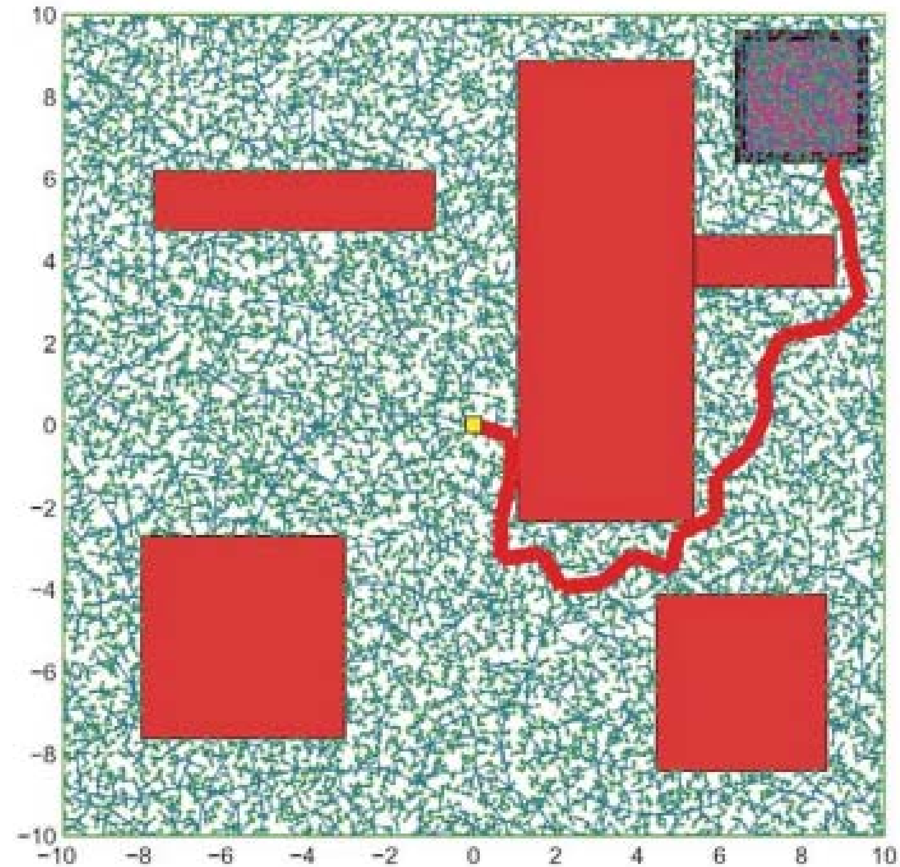


Figure 10.17: (Left) A tree generated by applying a uniformly-distributed random motion from a randomly chosen tree node results in a tree that does not explore very far. (Right) A tree generated by the RRT algorithm using samples drawn randomly from a uniform distribution. Both trees have 2000 nodes.

The net effect is that the nearly uniformly distributed samples "pull" the tree toward them, causing the tree to rapidly explore $\mathcal{X}_{\text{free}}$. An example of the effect of this pulling action on exploration is shown in Figure 10.17.



A tree generated by the RRT algorithm using samples drawn randomly from a uniform distribution. The tree has 2000 nodes.



The tree generated by the RRT after 20,000 nodes. The goal region is the square at the top right corner, and the shortest path is indicated.

b. probabilistic roadmaps (PRMs)

1. Construct a roadmap R with N nodes.
2. Search for a path in the graph, e.g., A* search.

Algorithm 4 PRM roadmap construction algorithm (undirected graph).

1: **for** $i = 1 \ldots N$ **do**
2:     $q_i \leftarrow$ sample from $\mathcal{C}_{\text{free}}$
3:     add q_i to R
4: **end for**
5: **for** $i = 1 \ldots N$ **do**
6:     $\mathcal{N}(q_i) \leftarrow k$ closest neighbors of q_i
7:     **for** each $q \in \mathcal{N}(q_i)$ **do**
8:         **if** there is a collision-free local path from q to q_i and there is not already an edge from q to q_i **then**
9:             add an edge from q to q_i to the roadmap R
10:         **end if**
11:     **end for**
12: **end for**
13: **return** R

How to sample from C_{free} (line 2)?

One option: Sample randomly from a uniform distribution on C and eliminate configurations in collision.

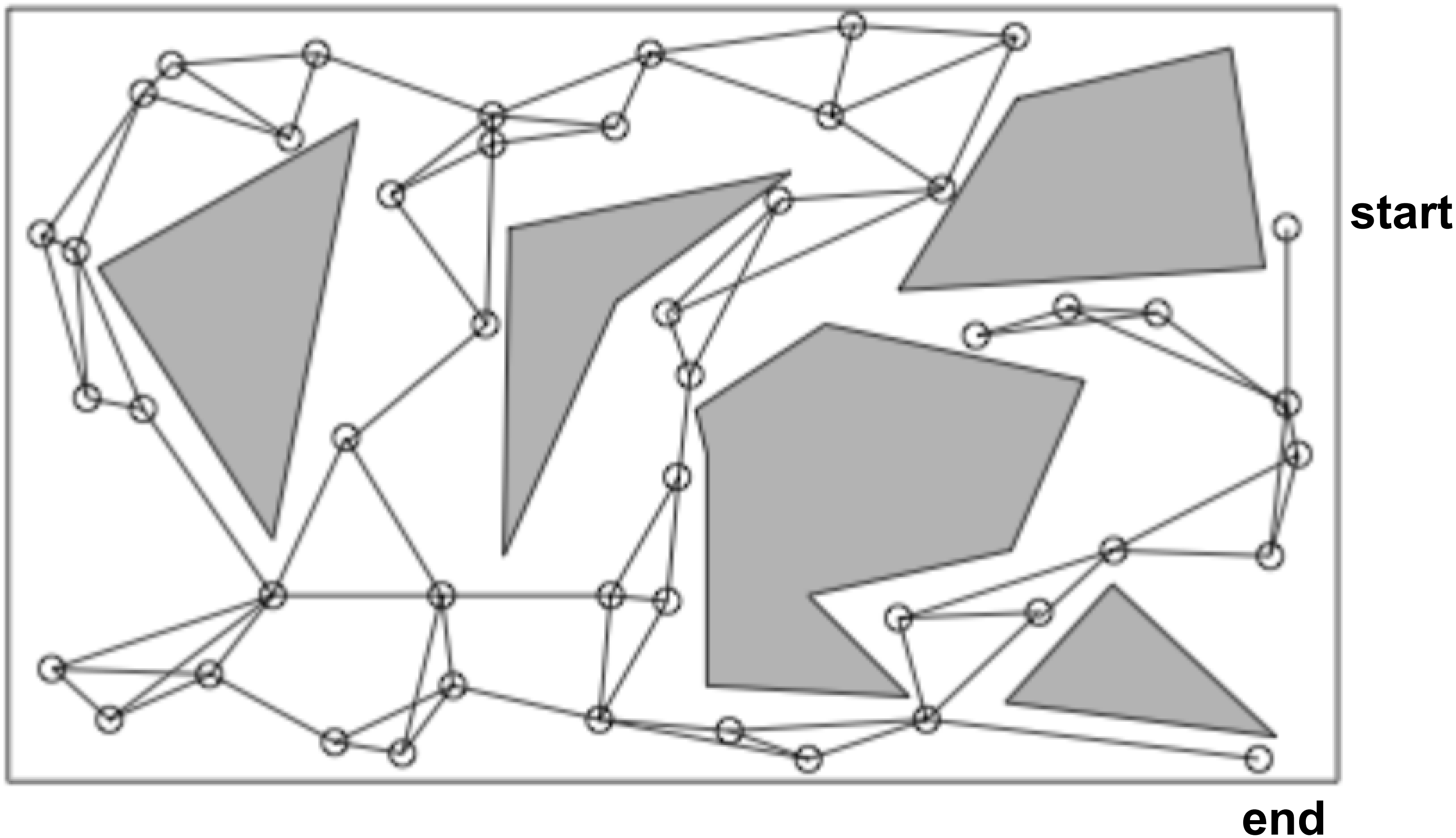


Figure 10.20: An example PRM roadmap for a point robot in $\mathcal{C} = \mathbb{R}^2$. The $k = 3$ closest neighbors are considered for connection to a sample node q. The degree of a node can be greater than three since it may be a close neighbor of many nodes.

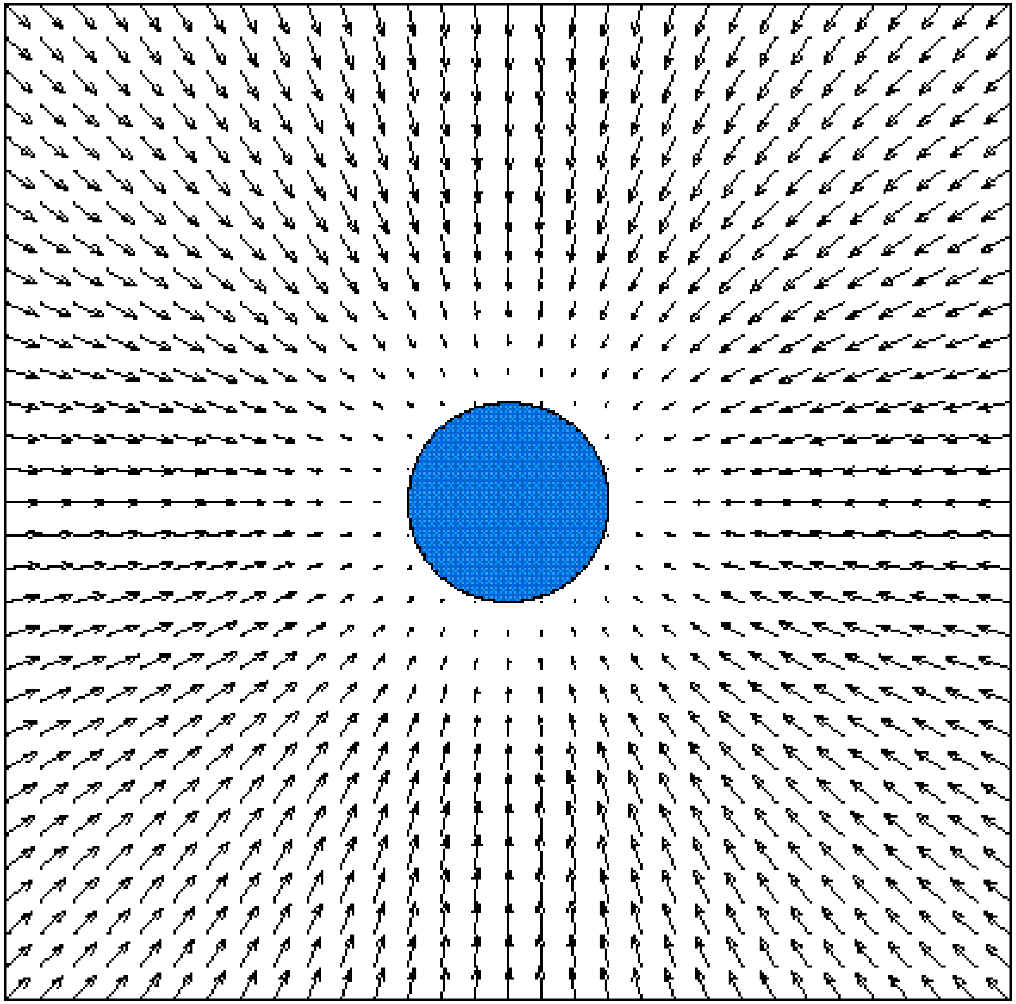
Potential Fields

- Oussama Khatib, 1986.

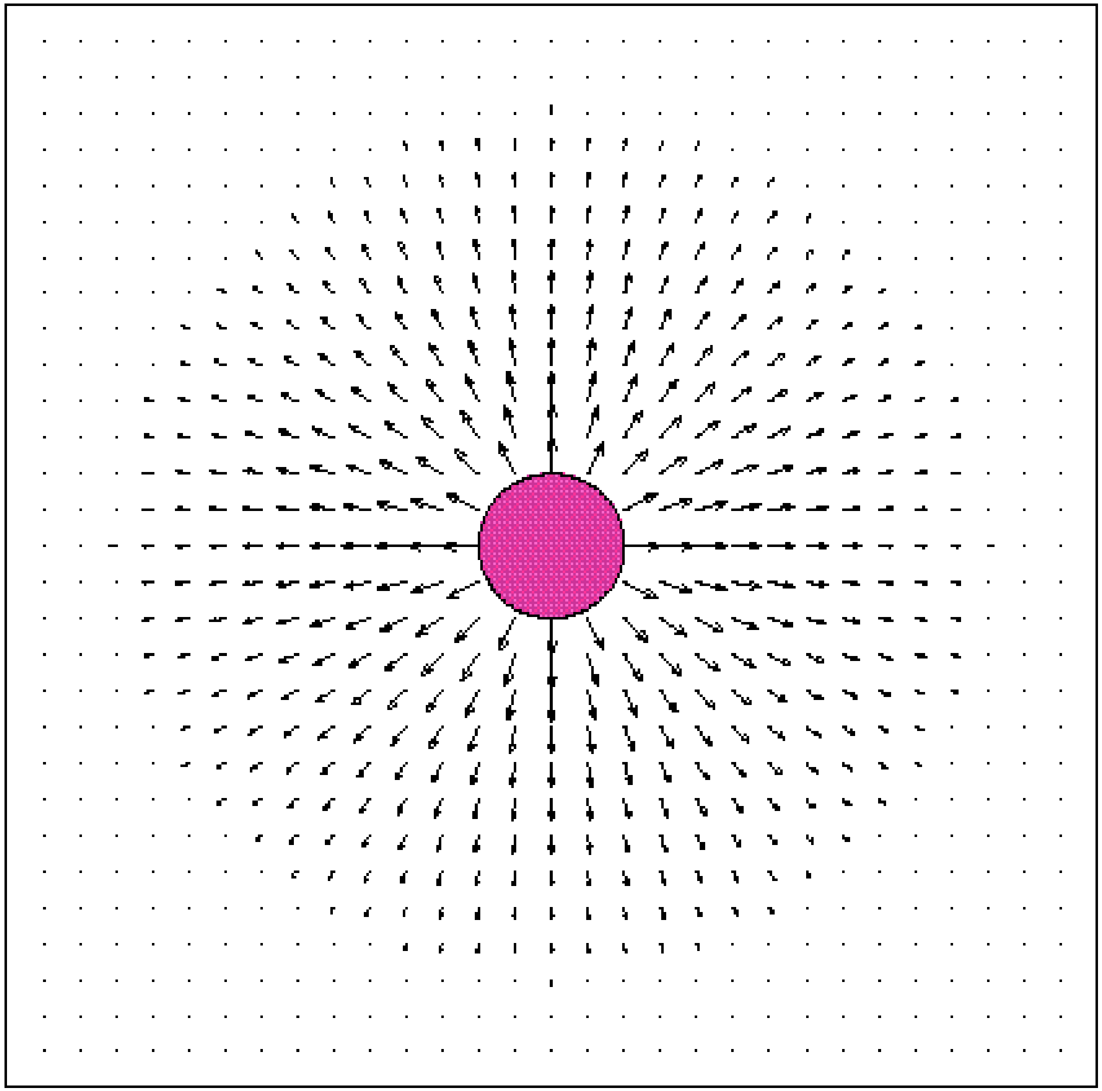
The manipulator moves in a field of forces.

The position to be reached is an attractive pole for the end effector and obstacles are repulsive surfaces for the manipulator parts.

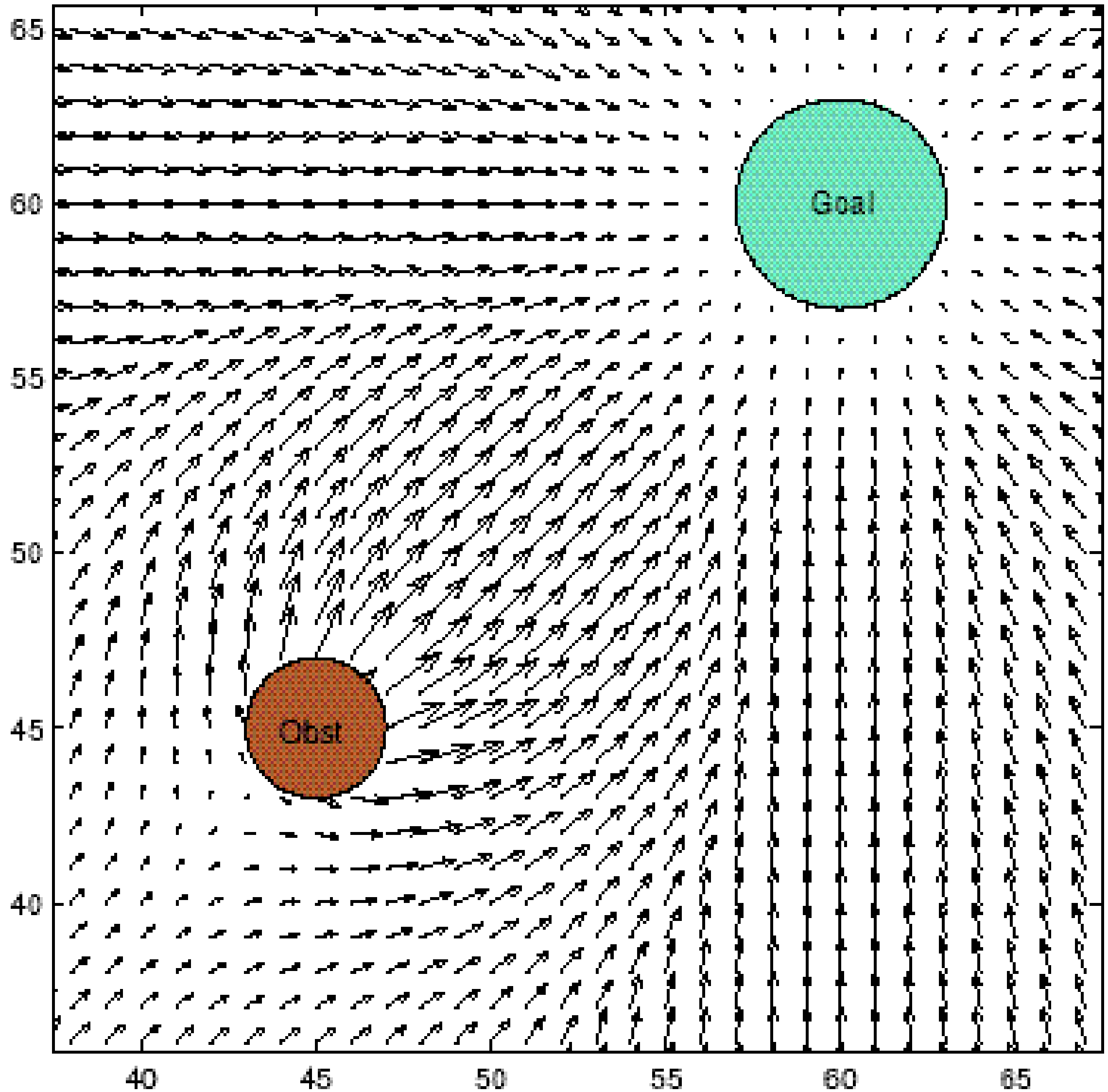
Attractive Potential Field



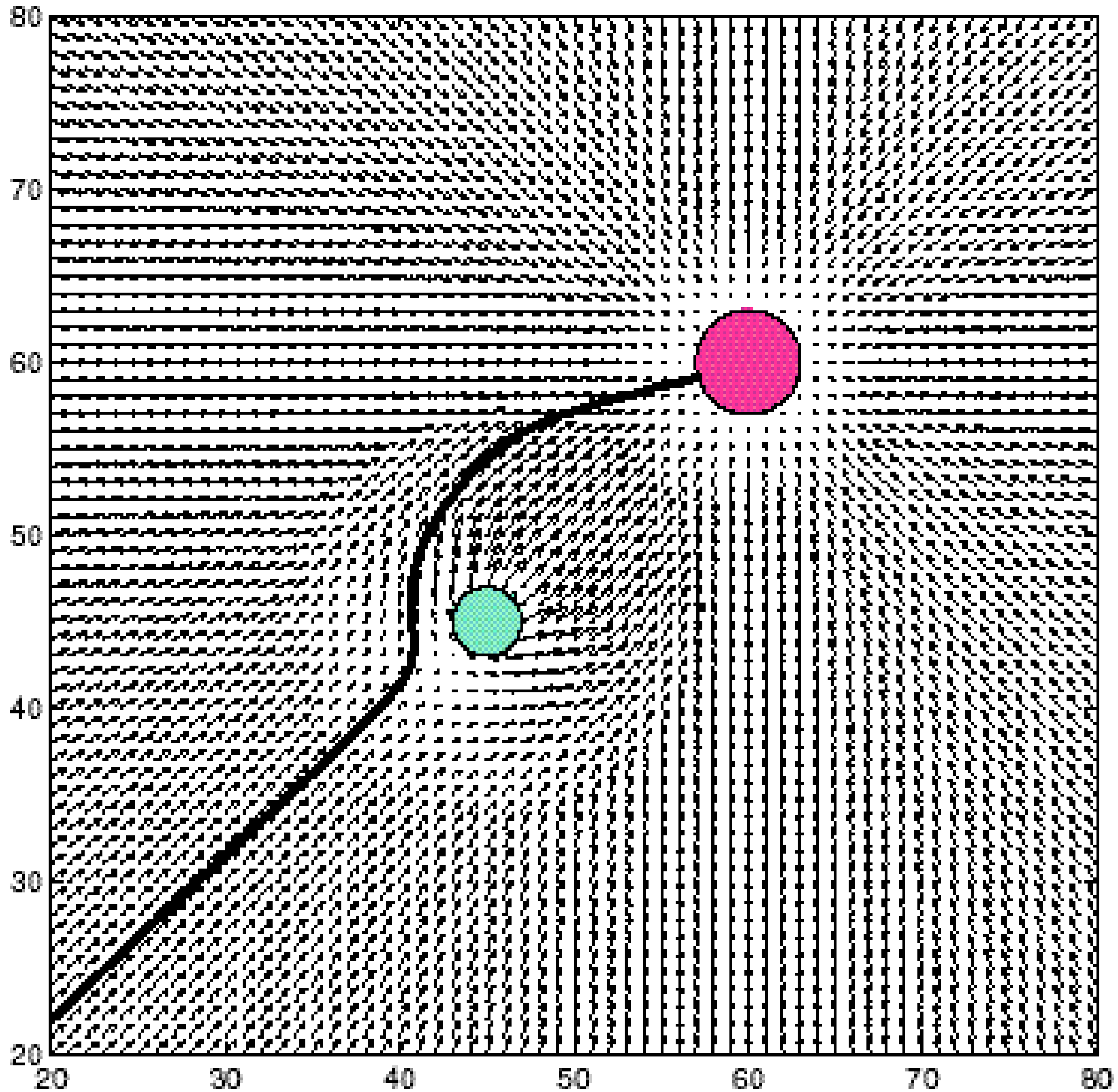
Repulsive Potential Field



Vector Sum of Two Fields



Resulting Robot Trajectory



Homeworks

For undergraduates:

7.2, 7.3, 7.11, 7.12, 7.15, 7.19, 7.23.