# Animation & Simulation

He Wang (王鹤)

# Interpolating Values

- Controlling the motion
  - Computing arc length (Analytic)
    - Analytic, forward differencing, adaptive forward differencing, numerically
  - Speed control
    - Ease-in/ease-out
    - Sine interpolation
    - Sinusoidals for acceleration and deceleration
    - Single cubic polynomial
    - Constant acceleration: parabolic ease-in/ease-out
  - General distance-time functions
    - constant acceleration/deceleration
    - arbitrary velocity
    - arbitrary distance-time
    - both distance and speed

# Interpolating Values

- Controlling the motion
  - Curve fitting (B spline)

Knot vector [0, 1, 2, …, n + k -1],

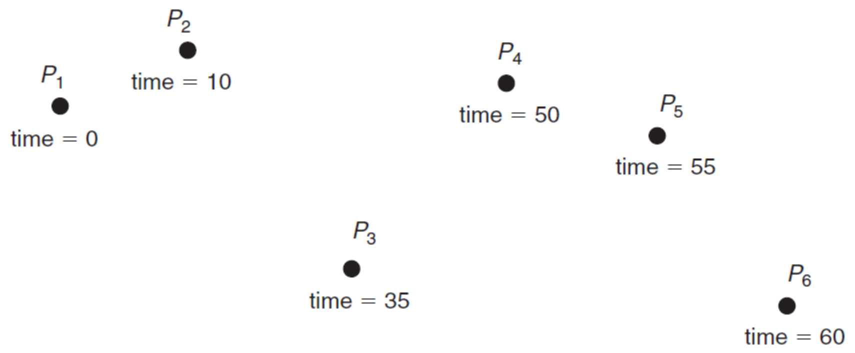
$$P(t) = \sum_{i=1}^{n+1} B_i N_{i,k}(t)$$

$$\begin{aligned}
P_1 &= N_{1,k}(t_1)B_1 + N_{2,k}(t_1)B_2 + \ldots + N_{n+1,k}(t_1)B_{n+1} \\
P_2 &= N_{1,k}(t_2)B_1 + N_{2,k}(t_2)B_2 + \ldots + N_{n+1,k}(t_2)B_{n+1} \\
\ldots \\
P_j &= N_{1,k}(t_j)B_1 + N_{2,k}(t_j)B_2 + \ldots + N_{n+1,k}(t_j)B_{n+1}
\end{aligned}$$

$$P = NB$$

$N$ is the matrix of basis functions

unknown defining control vertices are in the column matrix $B$.




Same number of variables and equations

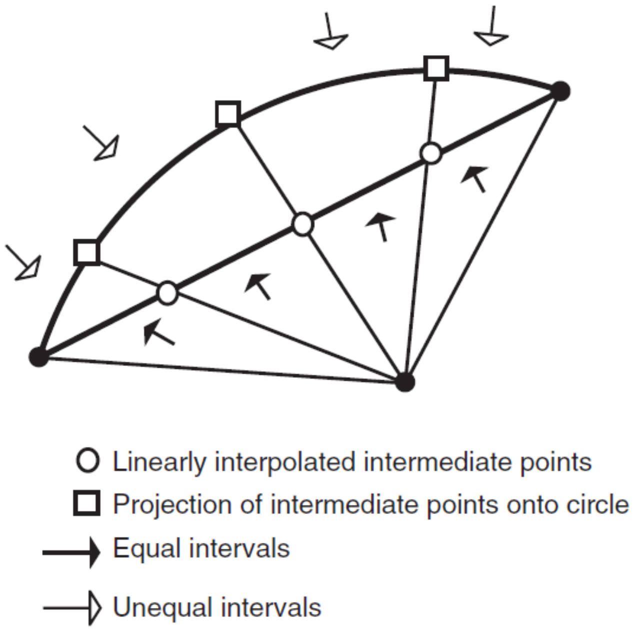
$$\longrightarrow \quad B = N^{-1}P$$

Different numbers of variables and equations

$$\begin{aligned}
P &= NB \\
N^T P &= N^T NB \\
[N^T N]^{-1} N^T P &= B
\end{aligned}$$

# Interpolating Values

- Interpolation of orientations




Instead of interpolating Euler angles, normally we interpolate two unit quaternions

Interpolating two unit quaternions results in non-constant-speed rotations

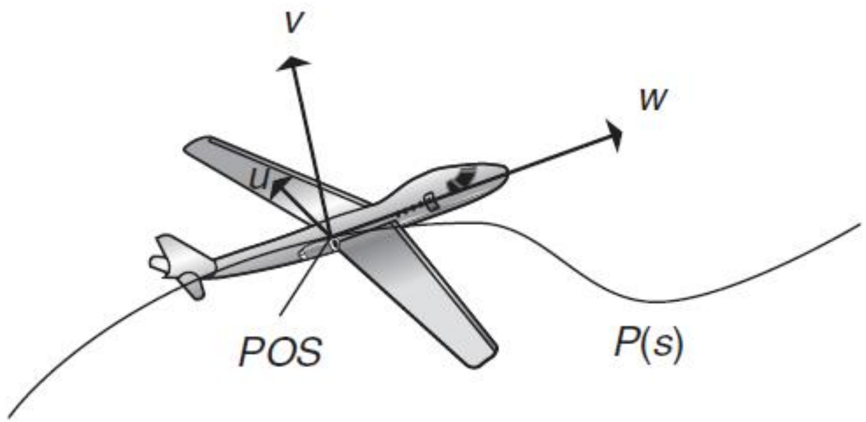
$$slerp(q_1, q_2, u) = \frac{\sin((1-u)\theta)}{sin(\theta)} q_1 + \frac{\sin(u\theta)}{sin(\theta)} q_2$$

The same as linear interpolation when interpolating multiple quaternions:
--continuity on waypoints.

Using curves (e.g. Bezier) to interpolate (refer to the book)

# Interpolating Values

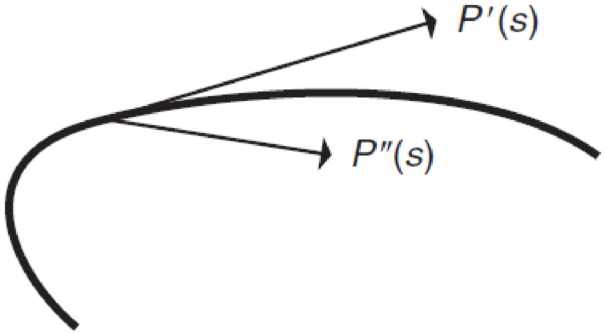
- Working with paths
  - Path following
    - Path/speed control, ease-in&ease-out
  - Orientation on a path
    - Several, can be arbitrary
      - Frenet frame
      - Centred of Interest

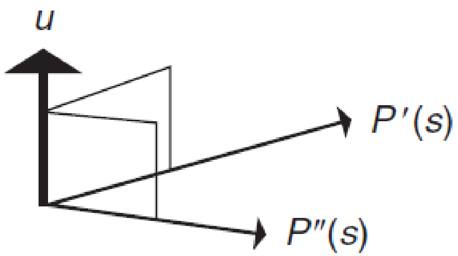


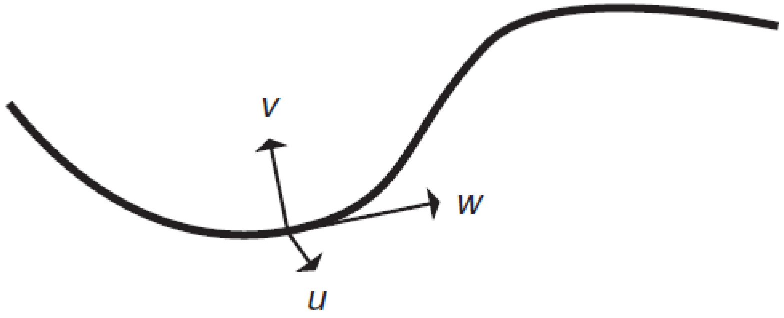
# Interpolating Values

- Working with paths
  - Orientation on a path
    - Frenet Frame (w, u, v)

$$
\begin{aligned}
w &= P'(s) \\
u &= P'(s) \times P''(s) \\
v &= u \times w
\end{aligned}
$$

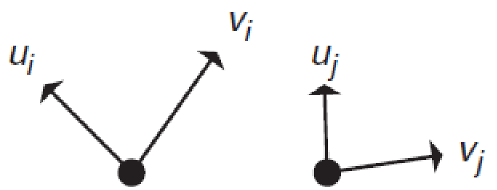


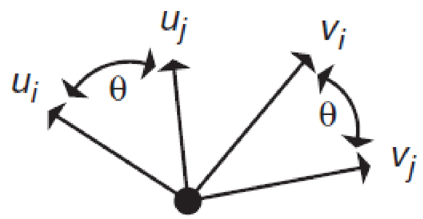



# Interpolating Values

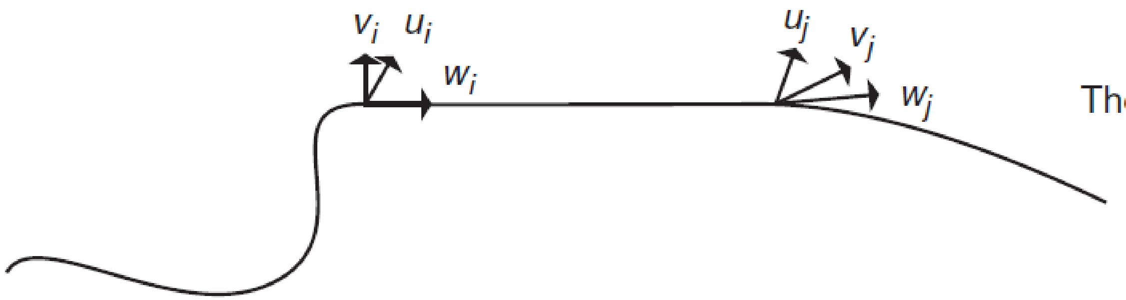
- Working with paths
  - Orientation on a path
    - Frenet Frame (w, u, v)
      - No “up”
      - What if no P’’?
        - interpolation




The two frames sighted down the (common) *w* vector



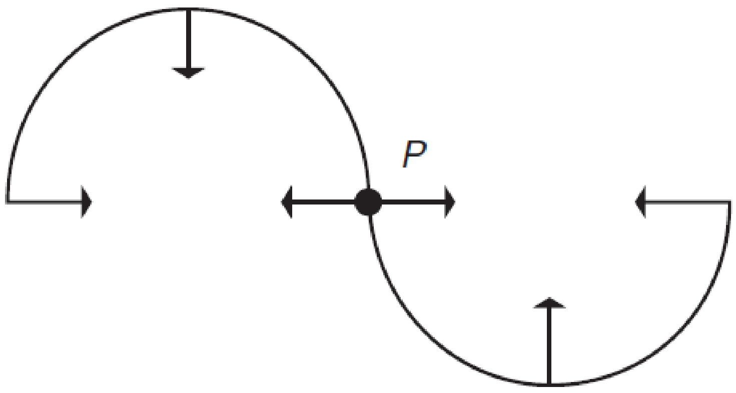

The two frames superimposed to identify angular difference




Frenet frames on the boundary of an undefined Frenet frame segment because of zero curvature

# Interpolating Values

- Working with paths
  - Orientation on a path
    - Frenet Frame (w, u, v)
      - No “up”
      - What if no P’’?
      - Discontinuity



# Interpolating Values

- Working with paths
  - Orientation on a path
    - Frenet Frame (w, u, v)
      - No “up”
      - What if no P’’?
      - Discontinuity
      - Extreme motion, not natural looking
      - If equipping v with ‘up’ direction, then w could wildly rotate

# Interpolating Values

- Working with paths
  - Orientation on a path
    - Centred of Interest (CoI)
      - A fixed point or object
        - Not too close

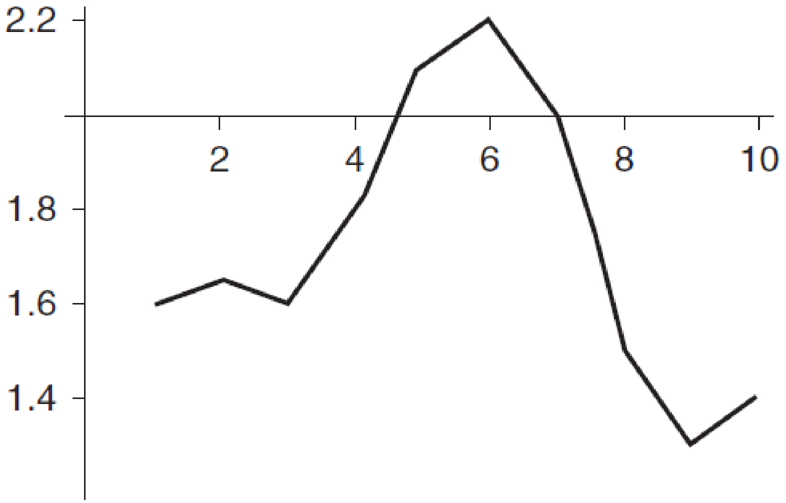
$$w = COI - POS$$
$$u = w \times (0,1,0)$$
$$v = u \times w$$

      - A separate path

$$w = C(s) - P(s)$$
$$u = w \times (U(s) - P(s))$$
$$v = u \times w$$

# Interpolating Values

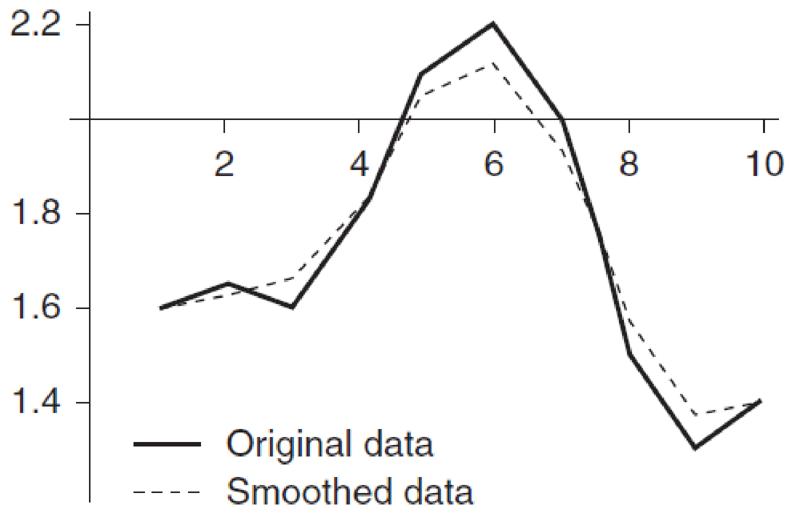
- Working with paths
  - Smoothing a path



# Interpolating Values

- Working with paths
  - Smoothing a path
    - Linear interpolation of adjacent values

$$P_i' = \frac{P_i + \frac{P_{i-1}+P_{i+1}}{2}}{2} = \frac{1}{4}P_{i-1} + \frac{1}{2}P_i + \frac{1}{4}P_{i+1}$$



# Interpolating Values

- Working with paths
  - Smoothing a path
    - Cubic interpolation of adjacent values

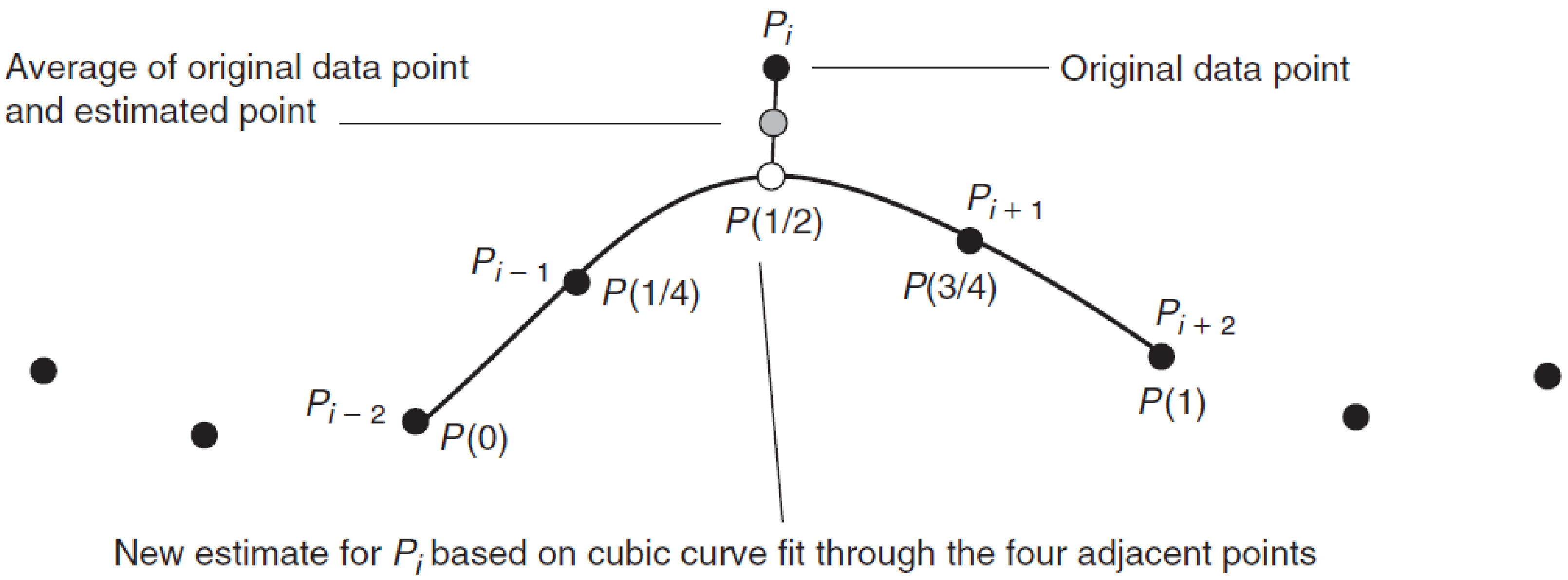
$$\boldsymbol{P}(u) = au^3 + bu^2 + cu + d$$

$$P_{i-2} = \boldsymbol{P}(0) = d$$

$$P_{i-1} = \boldsymbol{P}(1/4) = a\frac{1}{64} + b\frac{1}{16} + c\frac{1}{4} + d$$

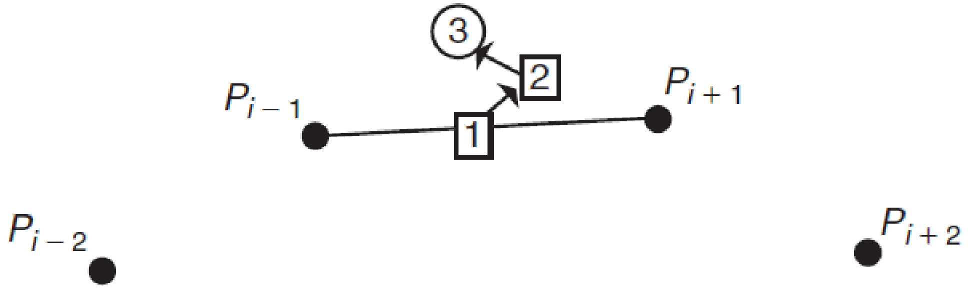
$$P_{i+1} = \boldsymbol{P}(3/4) = a\frac{27}{64} + b\frac{9}{16} + c\frac{3}{4} + d$$

$$P_{i+2} = a + b + c + d$$



# Interpolating Values

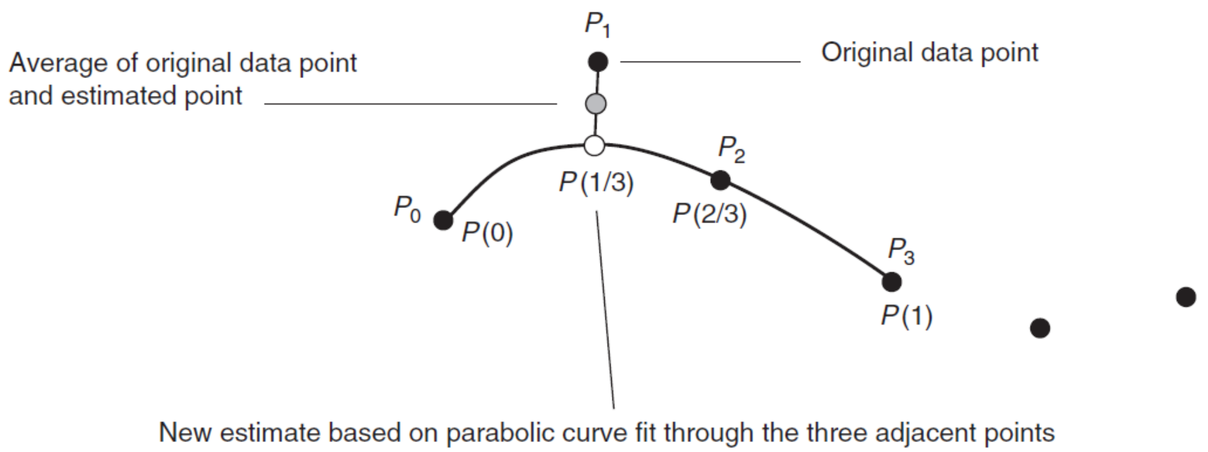
- Working with paths
  - Smoothing a path
    - Cubic interpolation of adjacent values

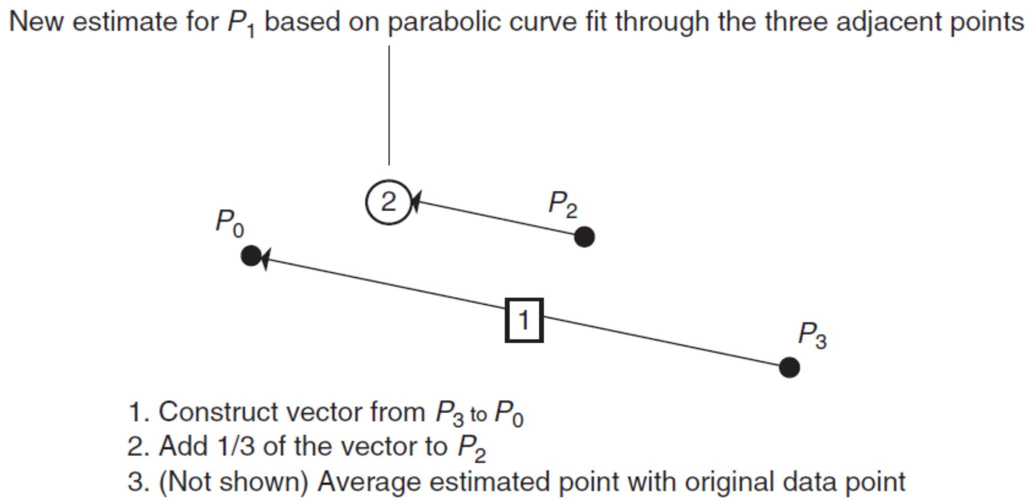



1. Average $P_{i-1}$ and $P_{i+1}$
2. Add 1/6 of the vector from $P_{i-2}$ to $P_{i-1}$
3. Add 1/6 of the vector from $P_{i+2}$ to $P_{i+1}$ to get new estimated point
4. (Not shown) Average estimated point with original data point

# Interpolating Values

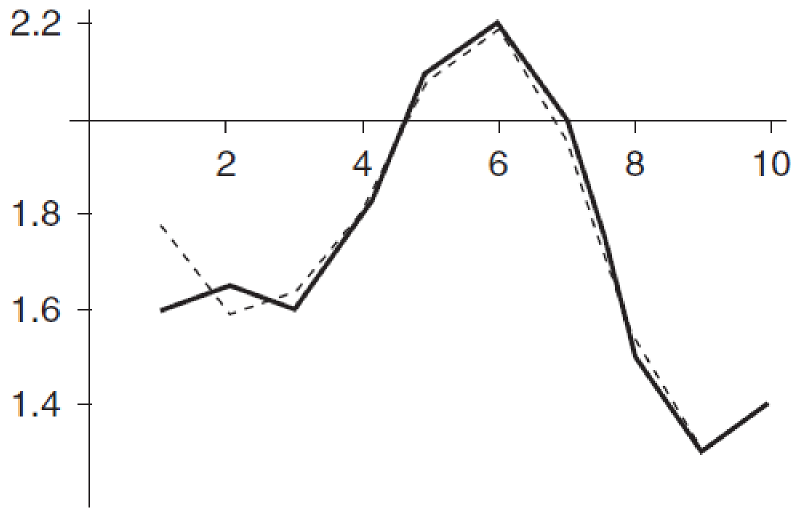
- Working with paths
  - Smoothing a path
    - Cubic interpolation of adjacent values
      - Parabolic curve for end conditions



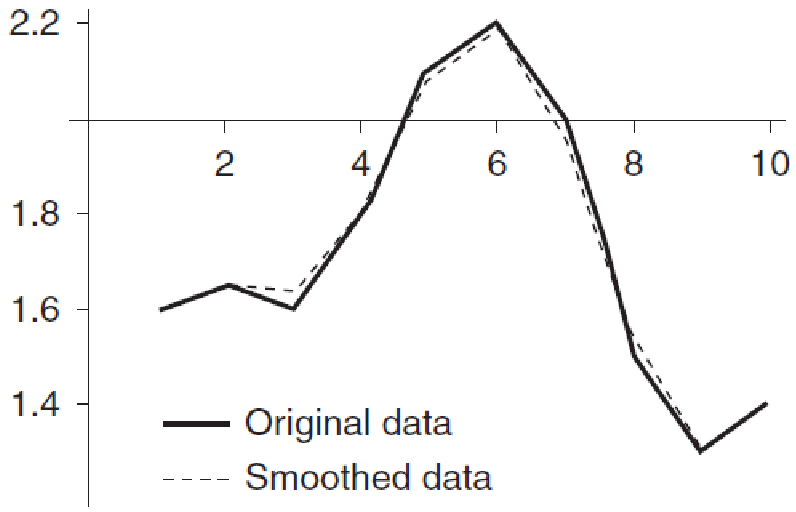



# Interpolating Values

- Working with paths
  - Smoothing a path
    - Cubic interpolation of adjacent values



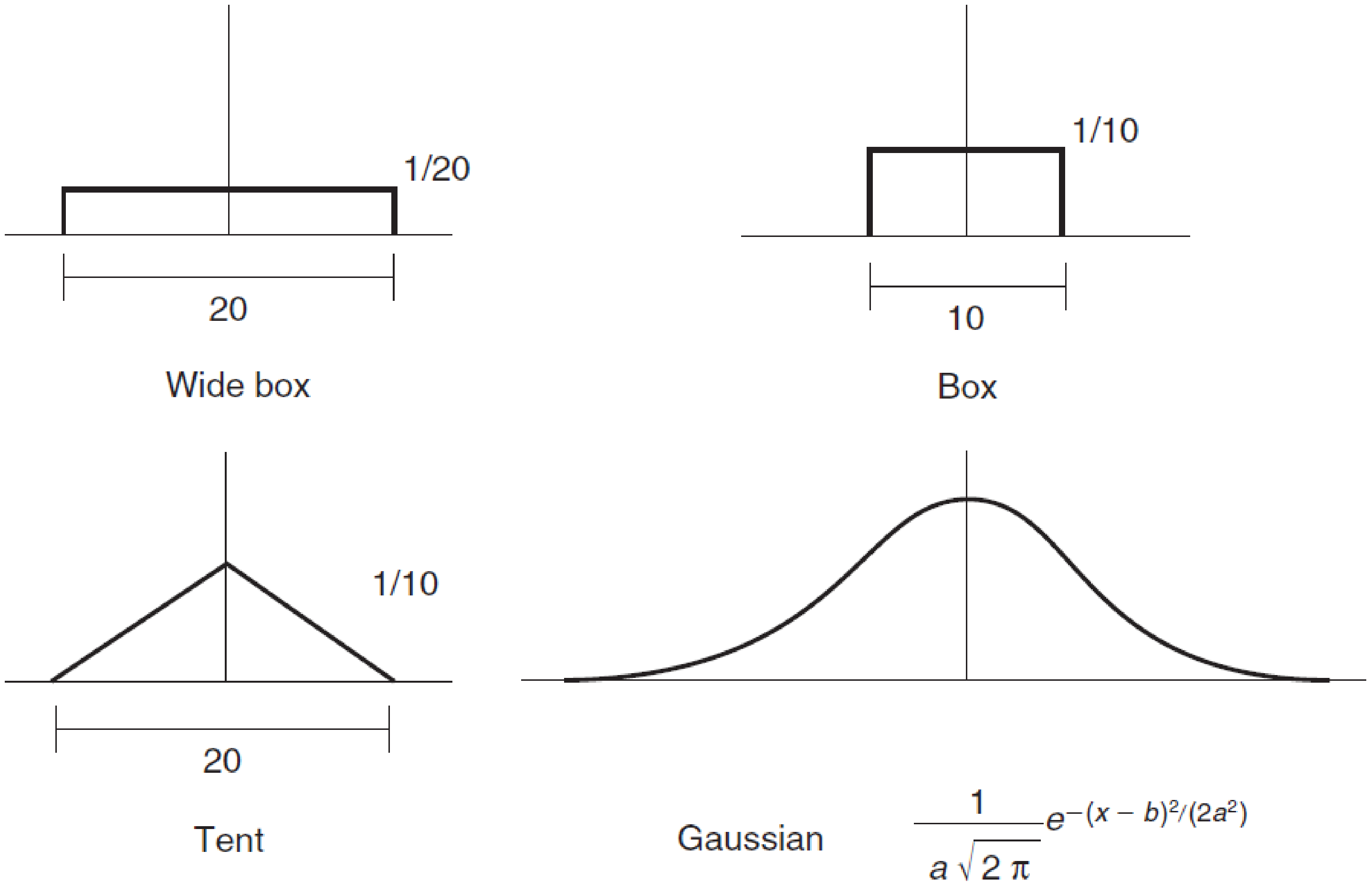

Cubic smoothing with parabolic end conditions




Cubic smoothing without smoothing the endpoints

# Interpolating Values

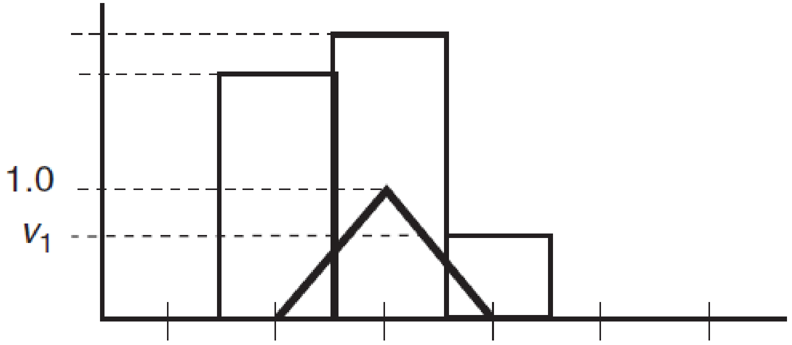
- Working with paths
  - Smoothing a path
    - Convolution Kernels
      - Centred at 0
      - Symmetric
      - Finite support
      - Integral to 1



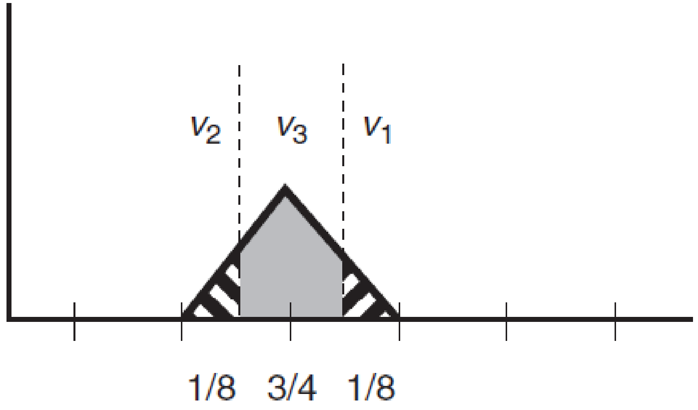
# Interpolating Values

- Working with paths
  - Smoothing a path
    - Convolution Kernels

$$\boldsymbol{P}(x) = \int_{-s}^{s} f(x+u)g(u)du$$




Smoothing kernel superimposed over step function




Areas of tent kernel under the different step function values

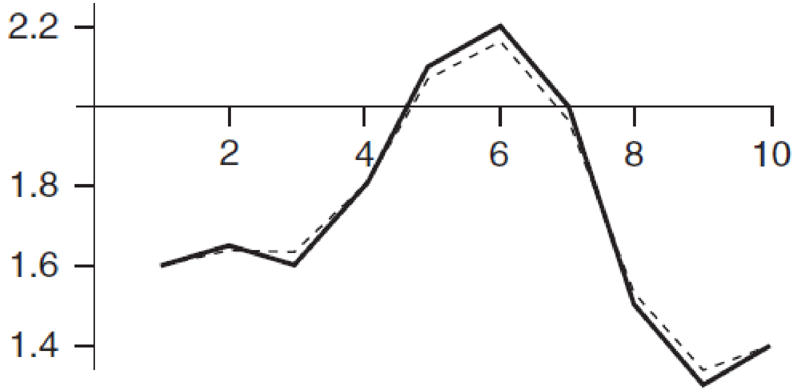
$$V = \frac{1}{8}v_1 + \frac{3}{4}v_2 + \frac{1}{8}v_3$$

Computation of value smoothed by applying area weights to step function values

# Interpolating Values

- Working with paths
  - Smoothing a path
    - Convolution Kernels

$$\boldsymbol{P}(x) = \int_{-s}^{s} f(x+u)g(u)du$$

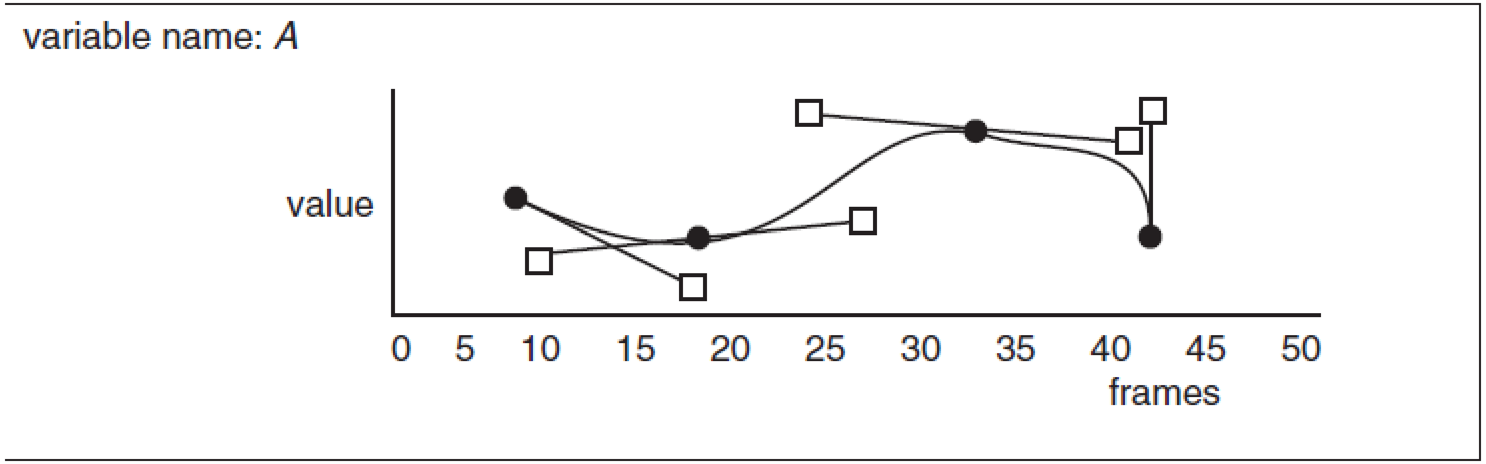


# Interpolating Values

- Working with paths
  - Smoothing a path
    - B-spline
    - Anything else?

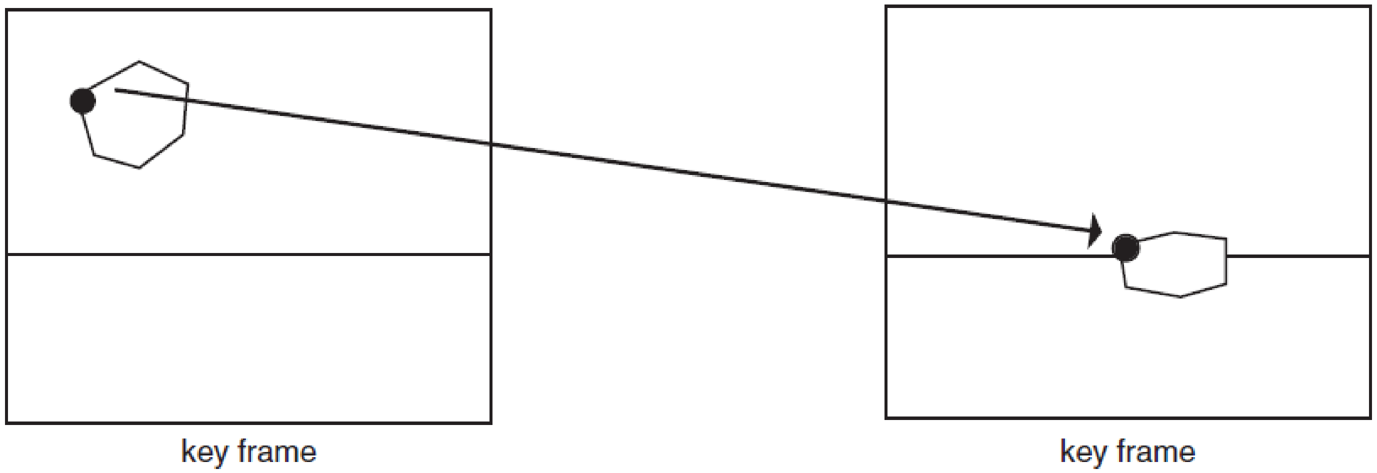
# Interpolation-Based Animation

- Key-frame systems
  - Manually define for variable

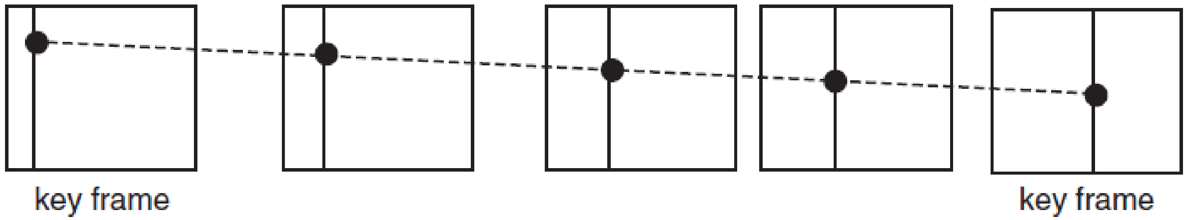


# Interpolation-Based Animation

- Key-frame systems
  - Linear interpolation




Simple key frames in which each curve of a frame has the same number of points as its counterpart in the other frame




Keys and three intermediate frames with linear interpolation of a single point (with reference lines showing the progression of the interpolation in *x* and *y*)

# Interpolation-Based Animation

- Key-frame systems
  - Given point correspondence

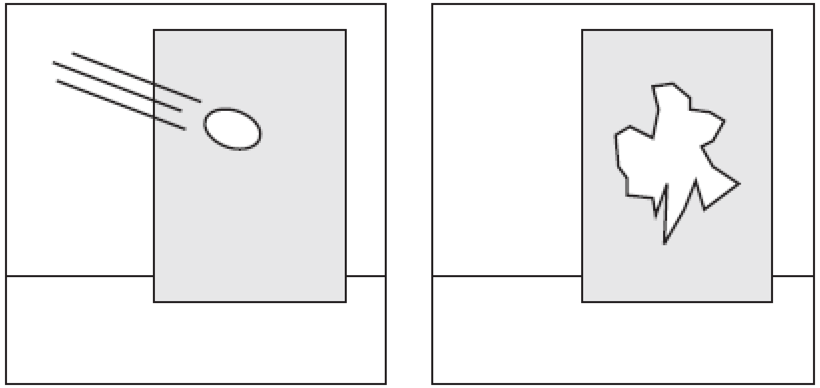


**FIGURE 4.3**

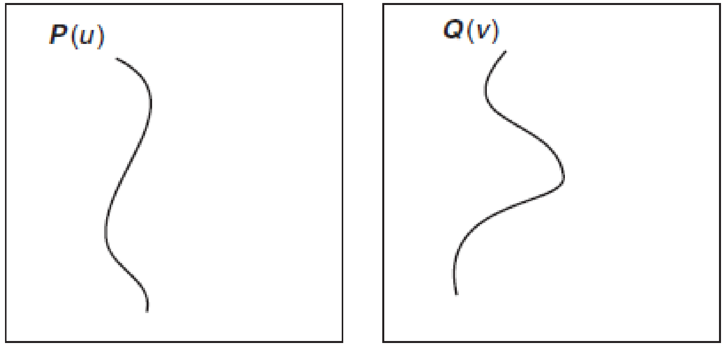
Object splatting against a wall. The shape must be interpolated from the initial egg shape to the splattered shape.

# Interpolation-Based Animation

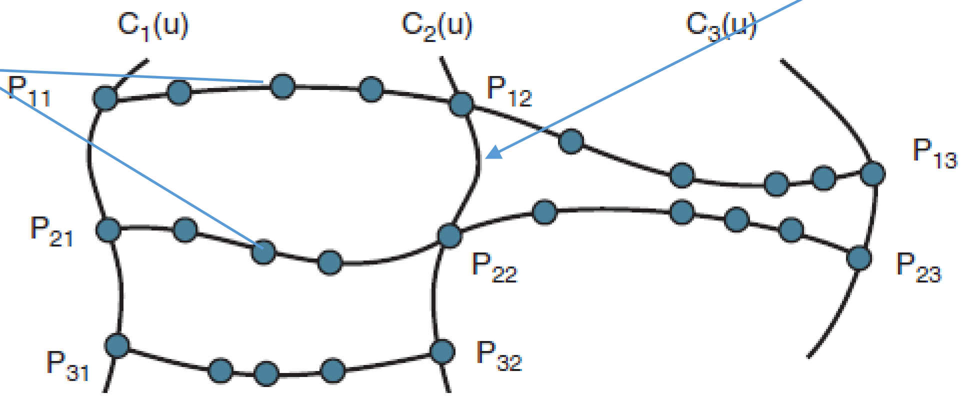
- Key-frame systems
  - Interpolation of two curves

1. Sample corresponding points first

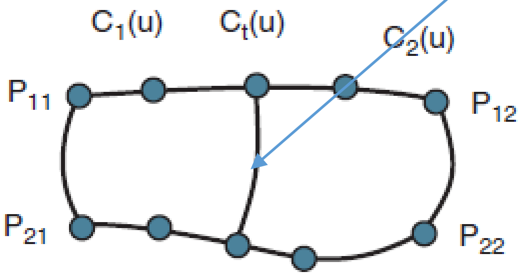
2. Interpolate corresponding points




Frame $f1$　　Frame $f2$




3. Sample points on the interpolate curves.

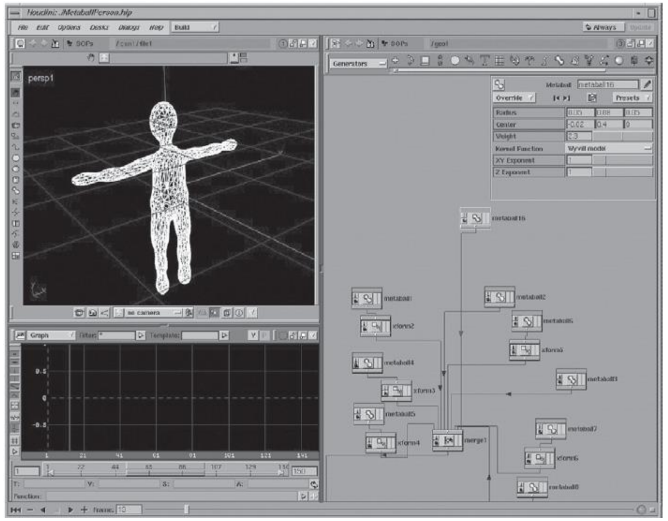


# Interpolation-Based Animation

- Animation Languages
  - Structured commands
  - Script-based
  - Features (I/O, data structure, time variable, rendering paras, etc.)
  - Pros: Hard-coded, repeatable; programmable
  - Cons: animators have to be programmers

# Interpolation-Based Animation

- Animation Languages
  - Artist-oriented animation languages
    - ANIMA II
  - Fully featured
    - Maya MEL (C++, python)
  - Graphical
    - Houdini

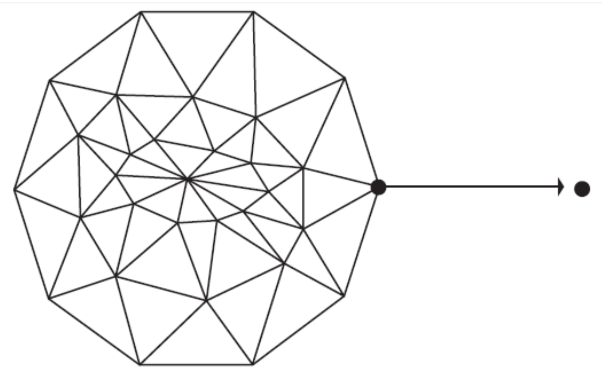


# Interpolation-Based Animation

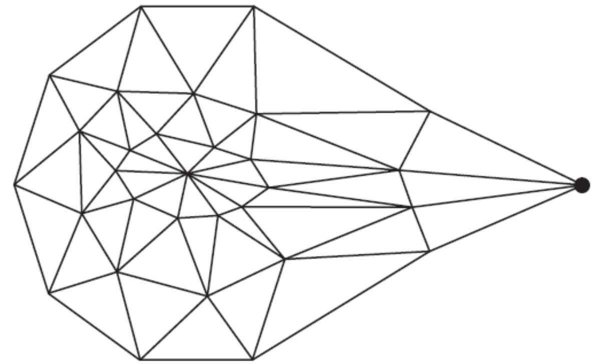
- Deforming Objects
  - Physically based
  - Free-form (Caging, handlers, etc)
  - Shape interpolation
  - Morphing

# Interpolation-Based Animation

- Picking and pulling



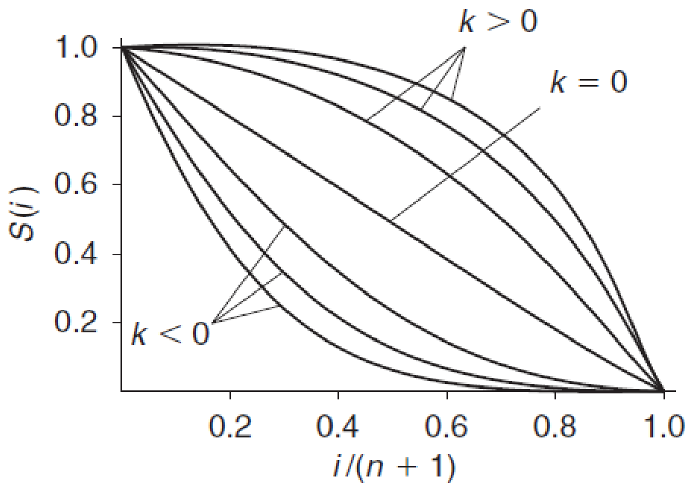
Displacement of seed vertex



Attenuated displacement propagated to adjacent vertices

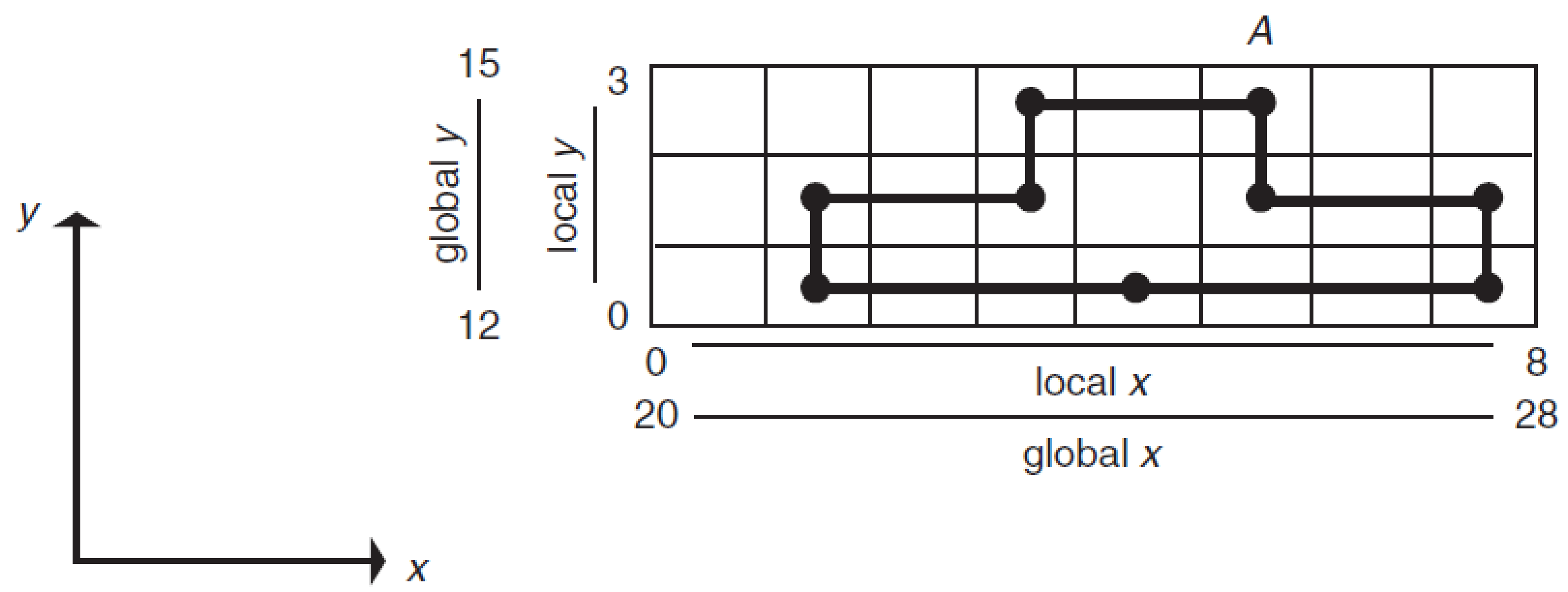
$$S(i) = 1 - \left(\frac{i}{n+1}\right)^{k+1} \qquad k \geq 0$$

$$= \left(1 - \left(\frac{i}{n+1}\right)\right)^{-k+1} \qquad k < 0$$



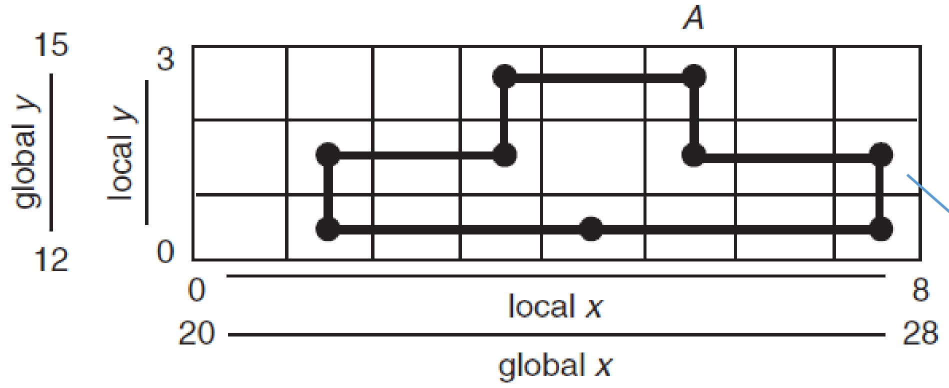
# Interpolation-Based Animation

- Deforming an embedding space (Free-form deformation, FFD)
  - 2D grid



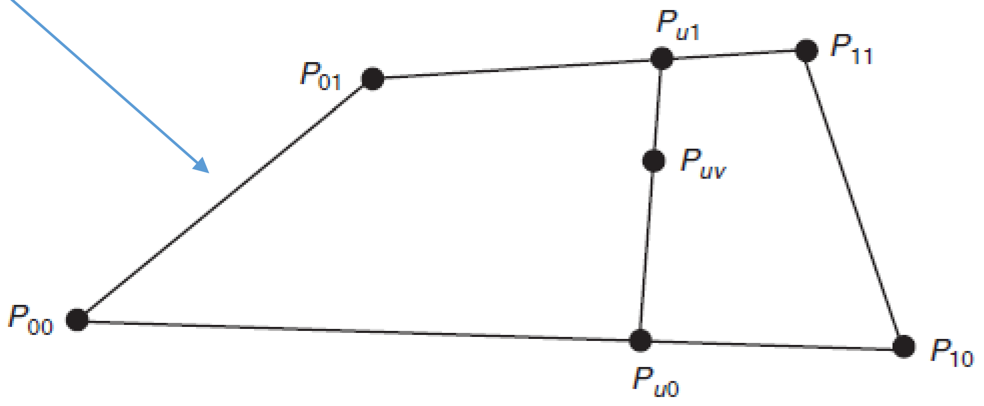
# Interpolation-Based Animation

- Deforming an embedding space (Free-form deformation, FFD)
  - 2D grid




$$P = (0.6)(0.7)P_{00} + (0.6)(1.0-0.7)P_{01} + (1.0-0.6)(0.7)P_{10} + (1.0-0.6)(1.0-0.7)P_{11}$$

$$\begin{aligned} P_{u0} &= (1-u)P_{00} + uP_{10} \\ P_{u1} &= (1-u)P_{01} + uP_{11} \\ P_{uv} &= (1-v)P_{u0} + vP_{u1} \\ &= (1-u)(1-v)P_{00} + (1-u)vP_{01} + u(1-v)P_{10} + uvP_{11} \end{aligned}$$



# Interpolation-Based Animation

- Deforming an embedding space (Free-form deformation, FFD)
  - 2D grid