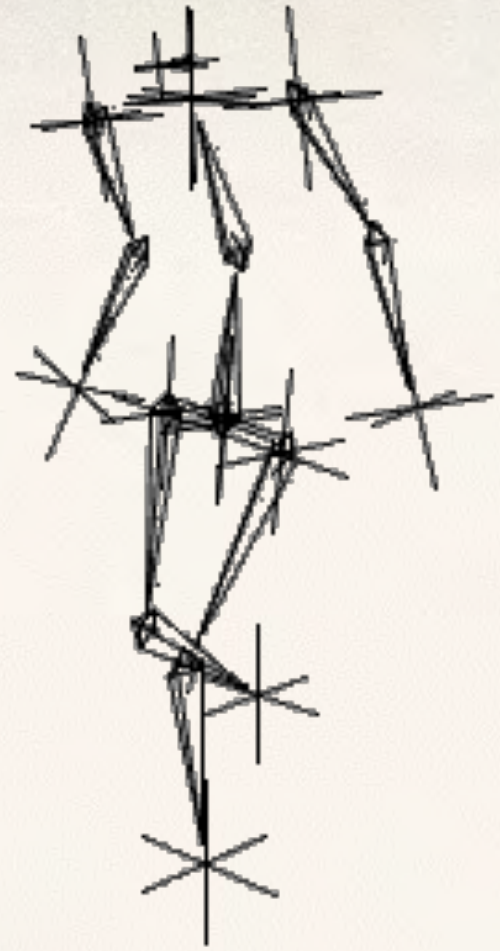


Softimage tutorials: Actor module, articulated / kinematic chains

Kinematics of redundant characters

Advanced Computer Animation Techniques
Aug-Dec 2014
cesteves@cimat.mx

Differential kinematics

- Find the relationship between the joint velocities and the end-effector linear and angular velocities.
- Express the end-effector linear velocity $\dot{\mathbf{p}}_e$ and angular velocity ω_e as a function of the joint velocities $\dot{\mathbf{q}}$.
- At any point in time, the Jacobian is a linear function of $\mathbf{v}_e$ (end-effector position and orientation).
- At the next instant of time, $\mathbf{v}_e$ has changed and so has the linear transformation represented by the Jacobian.

$$\dot{\mathbf{p}}_e = \mathbf{J}_P(\mathbf{q})\dot{\mathbf{q}}$$

$$\dot{\omega}_e = \mathbf{J}_O(\mathbf{q})\dot{\mathbf{q}}$$

$$\mathbf{v}_e = \begin{bmatrix} \dot{\mathbf{p}}_e \\ \dot{\omega}_e \end{bmatrix} = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}}$$

- Each term of the (6xn) geometric Jacobian **J**(**q**) relates the change of a specific joint to a specific change in the end-effector.

Derivative of a Rotation Matrix

- The mechanism forward (or direct) kinematics equation describes the **end-effector pose**, as a function of the joint variables, in terms of a **position vector** and a **rotation matrix**.

$$\mathbf{T}_e(\mathbf{q}) = \begin{bmatrix} \mathbf{R}_e(\mathbf{q}) & \mathbf{p}_e(\mathbf{q}) \\ \mathbf{0}^T & 1 \end{bmatrix}$$

- Characterize the end-effector linear and angular velocities:
  - consider the first derivative of a **rotation matrix** with respect to time.
- Consider a time-varying rotation matrix **R** = **R**(t).
- In view of orthogonality of **R**, one has the relation:

$$\mathbf{R}(t)\mathbf{R}^T(t) = I$$

which, differentiated with respect to time, gives the identity:

$$\dot{\mathbf{R}}(t)\mathbf{R}^T(t) + \mathbf{R}(t)\dot{\mathbf{R}}^T(t) = \mathbf{0}$$

From B.Siciliano, L. Sciavicco and L.Villani. Robotics: modelling, planning and control.

Derivative of a Rotation Matrix

Set $\mathbf{S}(t) = \dot{\mathbf{R}}(t)\mathbf{R}^T(t)$ the (3x3) matrix **S** is skew-symmetric (antisymmetric) since:

$$\mathbf{S}(t) + \mathbf{S}^T(t) = \mathbf{0}.$$

- Postmultiplying both sides of $\mathbf{S}(t) = \dot{\mathbf{R}}(t)\mathbf{R}^T(t)$ by **R***(t)*:

$$\mathbf{S}(t)\mathbf{R}(t) = \dot{\mathbf{R}}(t)\underbrace{\mathbf{R}^T(t)\mathbf{R}(t)}_{I}$$

$$\dot{\mathbf{R}}(t) = \mathbf{S}(t)\mathbf{R}(t)$$

which relates the rotation matrix R to its derivative by means of the skew-symetric operator **S**.

From B.Siciliano, L. Sciavicco and L.Villani. Robotics: modelling, planning and control.

Physical interpretation of the operator S

- Consider a constant vector **p'** and the vector **p**(t) = **R**(t)**p**'.
- The time derivative of **p**(t) is:

$$\dot{\mathbf{p}}(t) = \dot{\mathbf{R}}(t)\mathbf{p}'$$

which can be written as:

$$\dot{\mathbf{p}}(t) = \mathbf{S}(t)\mathbf{R}(t)\mathbf{p}'$$

- If the vector ω(t) denotes the **angular velocity** of frame **R**(t) with respect to the reference frame at time *t*, it is known from mechanics that:

$$\dot{\mathbf{p}}(t) = \omega(t) \times \mathbf{R}(t)\mathbf{p}'$$

- The matrix operator **S**(t) describes the vector product between the vector ω and the vector **R**(t)**p**'.

Physical interpretation of the operator S

- The matrix **S**(t) is so that its symmetric elements with respect to the main diagonal represent the components of vector $\omega(t)=[\ \omega_x\ \omega_y\ \omega_z\ ]^T$ in the form:

$$\mathbf{S} = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}$$

Example (1)

- Consider the elementary rotation matrix about axis **z**. If α is a function of time, by computing the time derivative of $\mathbf{R}_z(\alpha(t))$:

$$\mathbf{S}(t) = \dot{\mathbf{R}}(t)\mathbf{R}^T(t) \qquad \mathbf{R} = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\frac{d\mathbf{R}}{dt} = \frac{\partial \mathbf{R}}{\partial \alpha}\frac{d\alpha}{dt} = \begin{bmatrix} -\sin\alpha & -\cos\alpha & 0 \\ \cos\alpha & -\sin\alpha & 0 \\ 0 & 0 & 0 \end{bmatrix} \frac{d\alpha}{dt} = \begin{bmatrix} -\dot{\alpha}\sin\alpha & -\dot{\alpha}\cos\alpha & 0 \\ \dot{\alpha}\cos\alpha & -\dot{\alpha}\sin\alpha & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$S(t) = \begin{bmatrix} -\dot{\alpha}\sin\alpha & -\dot{\alpha}\cos\alpha & 0 \\ \dot{\alpha}\cos\alpha & -\dot{\alpha}\sin\alpha & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \cos\alpha & \sin\alpha & 0 \\ -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

From B.Siciliano, L. Sciavicco and L.Villani. Robotics: modelling, planning and control.

Example (2)

$$S(t) = \begin{bmatrix} -\dot{\alpha}\sin\alpha\cos\alpha + \dot{\alpha}\sin\alpha\cos\alpha & -\dot{\alpha}(\sin^2\alpha + \cos^2\alpha) & 0 \\ \dot{\alpha}(\cos^2\alpha + \sin^2\alpha) & \dot{\alpha}\cos\alpha\sin\alpha - \dot{\alpha}\cos\alpha\sin\alpha & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & -\dot{\alpha} & 0 \\ \dot{\alpha} & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} = \mathbf{S}(\omega(t)).$$

- Que de acuerdo a $\mathbf{S} = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}$ $\quad \omega = \begin{bmatrix} 0 & 0 & \dot{\alpha} \end{bmatrix}^T$

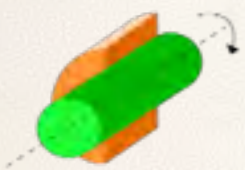
expresa la velocidad angular del marco de referencia alrededor del eje z.

Geometric Jacobian matrix

- The basic Jacobian matrix is computed efficiently as follows [Orin and Schrader, 1984]

$$J_\omega = \begin{pmatrix} J_{\omega_1} & J_{\omega_2} & \ldots & J_{\omega_n} \end{pmatrix}$$

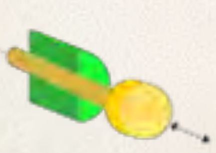
- where $J_{\omega_i} \in \mathbb{R}^6$ implies the *j*th column vector of the Jacobian matrix and is computed as follows:



revolute joint

$$\frac{\partial \mathbf{p}}{\partial \theta_j} = \begin{pmatrix} \mathbf{v}_j \times (\mathbf{p} - \mathbf{r}_j) \\ \mathbf{v}_j \end{pmatrix}$$

- where $\mathbf{r}_j$ is the position of the joint, and $\mathbf{v}_j$ is a unit vector pointing along the current axis of rotation for the joint.
- angles are measured in radians with the direction of rotation given by the right hand rule.
- this is only if the end-effector is affected by the joint, otherwise it is 0.

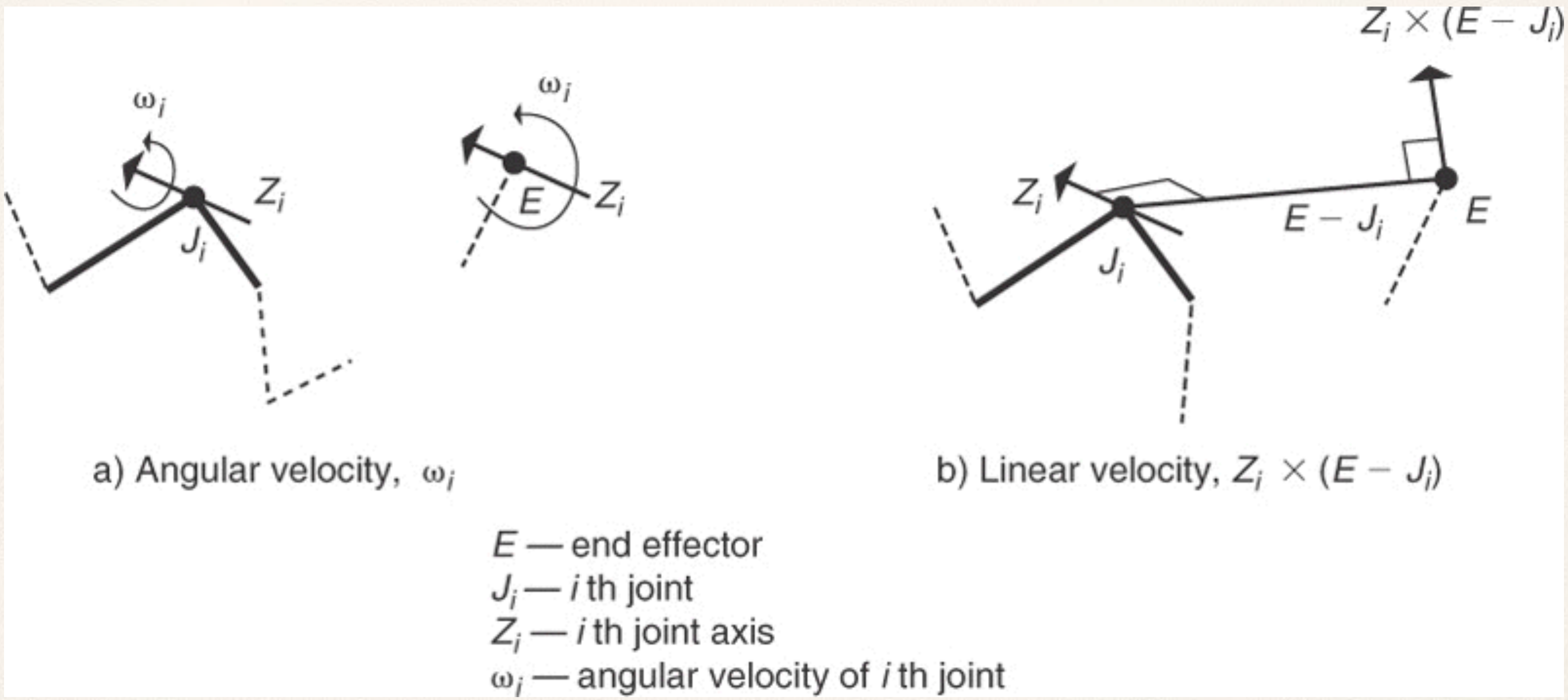


prismatic joint

$$\frac{\partial \mathbf{p}}{\partial \theta_j} = \begin{pmatrix} \mathbf{v}_j \\ \mathbf{0} \end{pmatrix}$$

Geometric Jacobian matrix

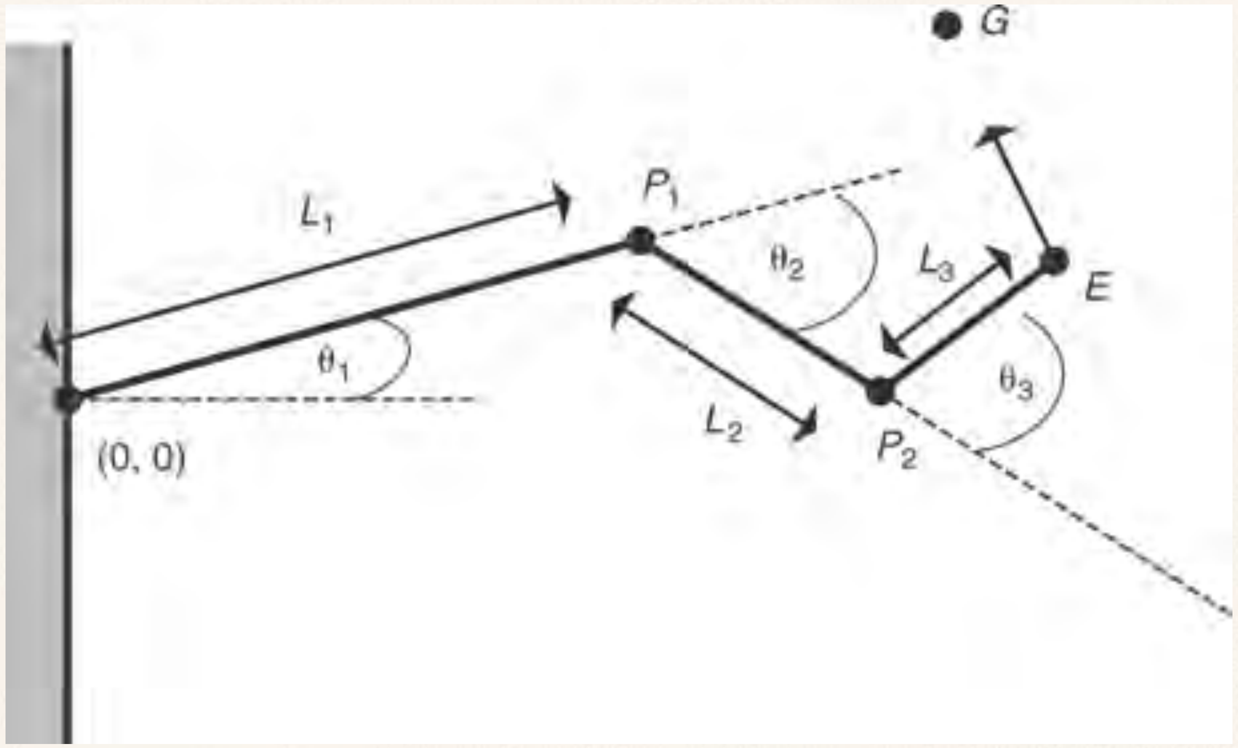
- Make sure that all of the coordinate values are in the same coordinate system (world coordinates).



R. Parent. Computer Animation: algorithms and techniques. Morgan Kauffman, 2008

Example (1)

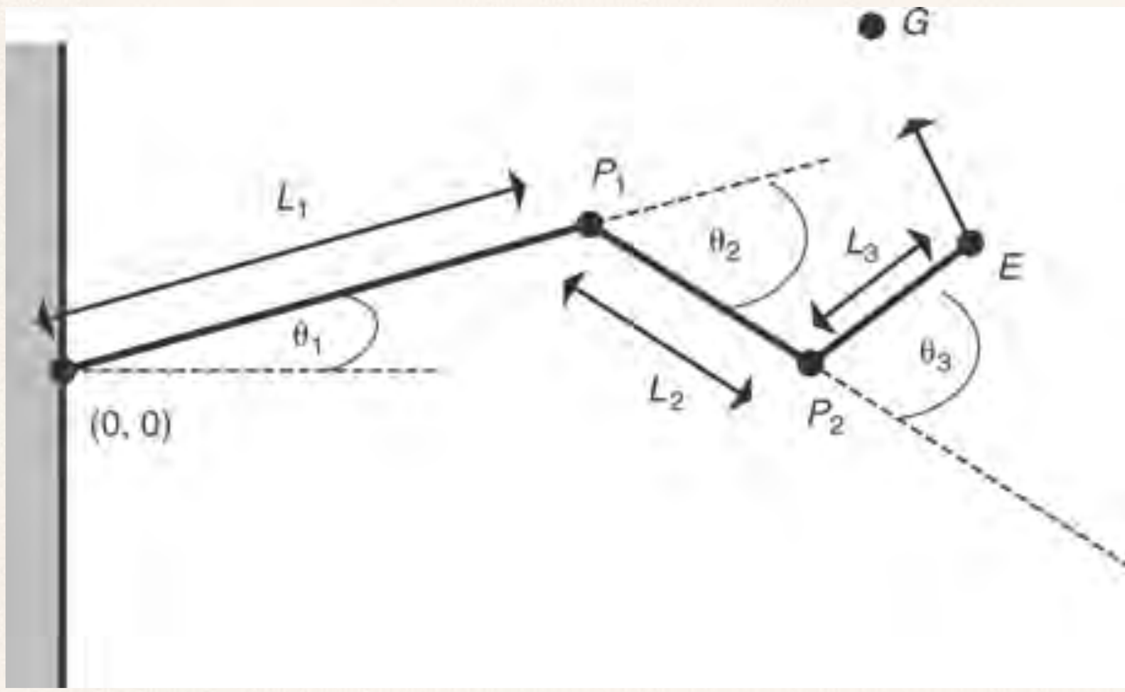
- Consider the three-revolute joint, planar manipulator of the Figure.

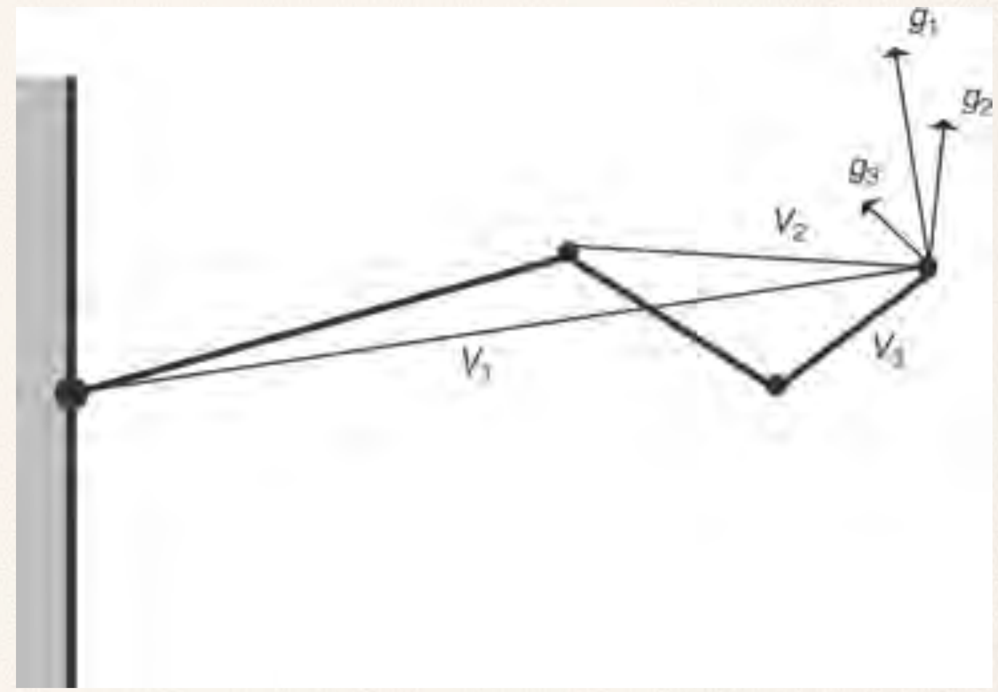


R. Parent. Computer Animation: algorithms and techniques. Morgan Kauffman, 2008

- Move the end-effector **E** to the goal position **G**.
- We only care about the position in this example, not the orientation.
- The effect of an incremental rotation g_i, of each joint can be determined by the cross product of the joint axis and the vector from the joint to the end-effector, V_i.

Example (2)



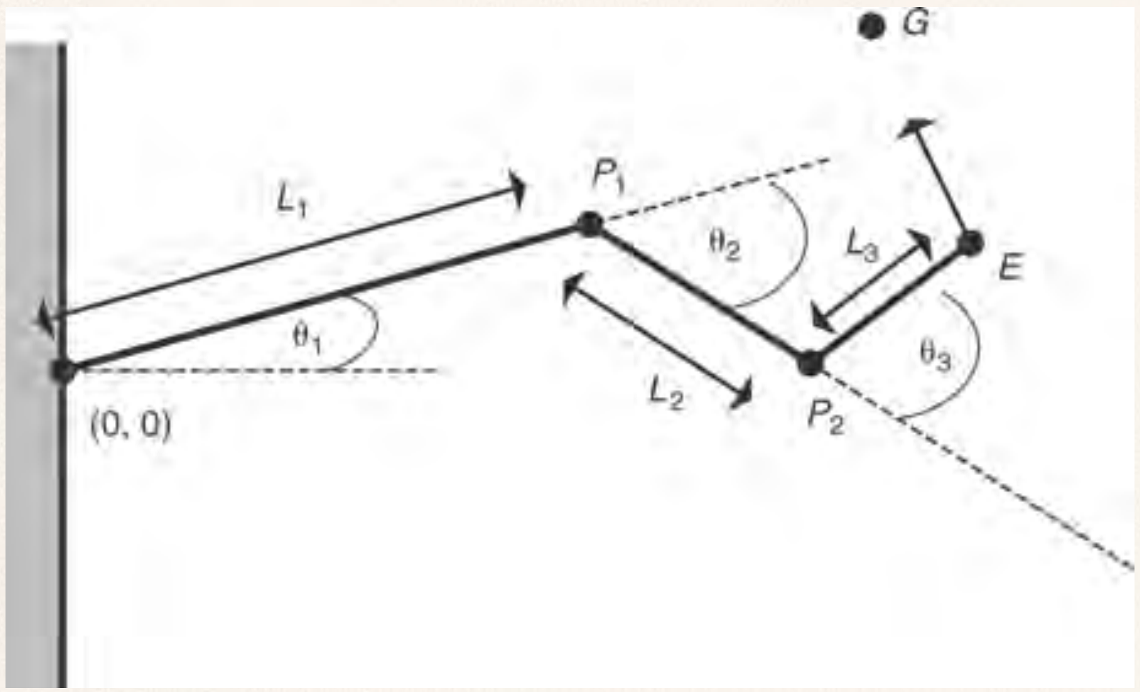


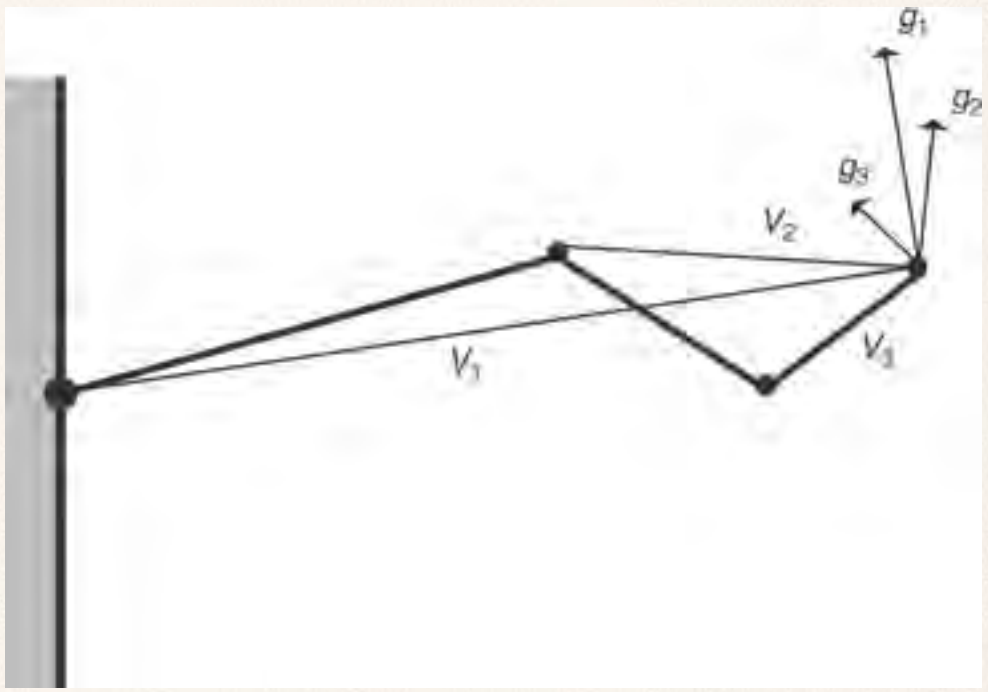
R. Parent. Computer Animation: algorithms and techniques. Morgan Kauffman, 2008

- The desired change to the end-effector is the difference between the current position of the end-effector and the goal position.

$$V = \begin{bmatrix} (G-E)_x \\ (G-E)_y \\ (G-E)_z \end{bmatrix}$$

Example (3)





R. Parent. Computer Animation: algorithms and techniques. Morgan Kauffman, 2008

- and the Jacobian matrix is:

$$J = \left[\begin{array}{ccc} (0,0,1) \times (E)_x) & (0,0,1) \times (E - P_1)_x) & (0,0,1) \times (E - P_2)_x) \\ (0,0,1) \times (E)_y) & (0,0,1) \times (E - P_1)_y) & (0,0,1) \times (E - P_2)_y) \\ (0,0,1) \times (E)_z) & (0,0,1) \times (E - P_1)_z) & (0,0,1) \times (E - P_2)_z) \end{array} \right]$$

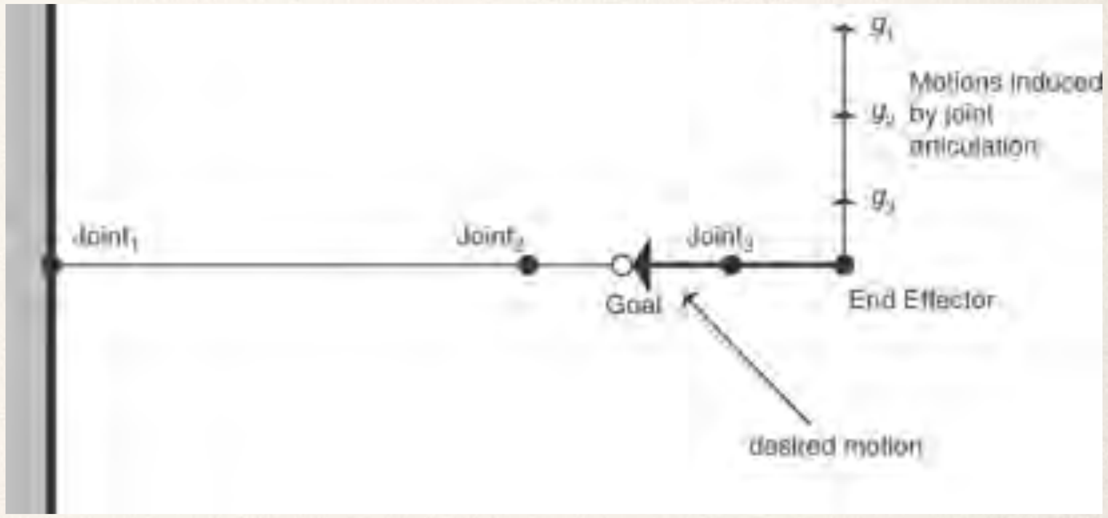
- Linearize locally.
- Jacobian depends on current configuration.

Redundant mechanisms

- If **J** is a square matrix, the inverse of the Jacobian can be easily computed.
- If the inverse of the Jacobian does not exist, then the system is said to be singular for the given joint angles.
- A singularity occurs when a linear combination of the joint angle velocities cannot be formed to produce the desired end-effector velocities.
- E.g. fully extended planar arm with a goal position somewhere in the forearm.
    - a change in each joint angle would produce a vector perpendicular to the desired direction.
    - no linear combination of these vectors could produce the desired motion vector.
    - even with non-singular configurations large values have to be used.

$$\mathbf{v}_e = \begin{bmatrix} \dot{\mathbf{p}}_e \\ \dot{\omega}_e \end{bmatrix} = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}}$$

$$\mathbf{J}^{-1}\mathbf{v}_e = \dot{\mathbf{q}}$$



R. Parent. Computer Animation: algorithms and techniques. Morgan Kauffman, 2008

Redundant mechanisms

- Problems with singularities can be reduced if the mechanism is redundant: more DOFs than there are constraints to be satisfied.
- In this case, the Jacobian is not a square matrix and potentially there are an infinite number of solutions.
- Because the Jacobian is not square, a conventional inverse does not exist.
- If the rows of **J** are linearly independent (i.e., **J** has full row rank), then $(\mathbf{J}\mathbf{J}^T)^{-1}$ exists and instead the pseudoinverse $\mathbf{J}^+$ can be used.
  - a matrix multiplied by its own transpose will be a square matrix.

$$\mathbf{v}_e = \mathbf{J}\dot{\mathbf{q}}$$

$$\mathbf{J}^T\mathbf{v}_e = \mathbf{J}^T\mathbf{J}\dot{\mathbf{q}}$$

$$(\mathbf{J}^T\mathbf{J})^{-1}\mathbf{J}^T\mathbf{v}_e = (\mathbf{J}^T\mathbf{J})^{-1}\mathbf{J}^T\mathbf{J}\dot{\mathbf{q}}$$

$$\mathbf{J}^\dagger\mathbf{v}_e = \dot{\mathbf{q}}$$

$\mathbf{J}^\dagger = \mathbf{J}^T(\mathbf{J}\mathbf{J}^T)^{-1}$: pseudoinverse of **J**.

Redundant mechanisms

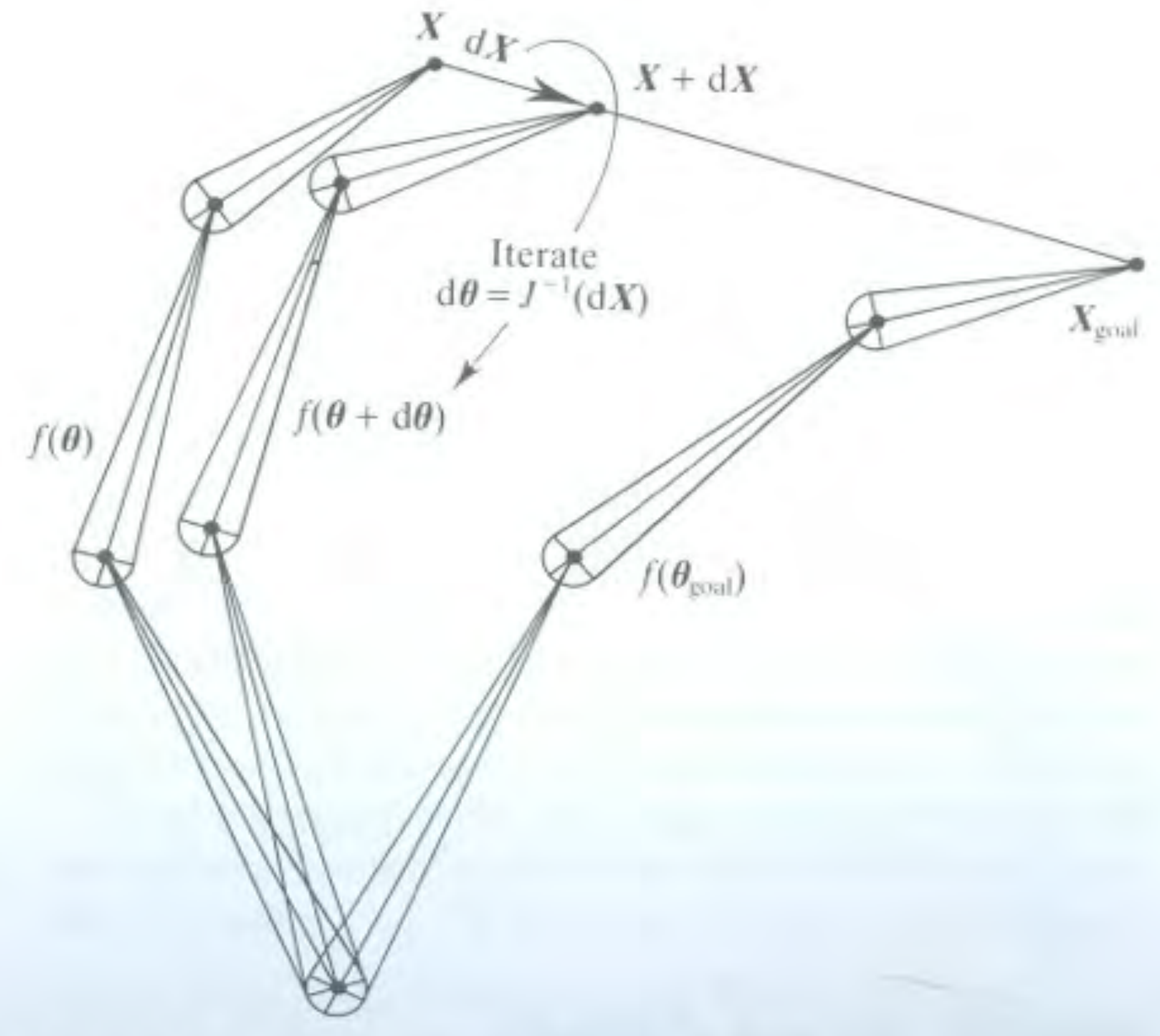
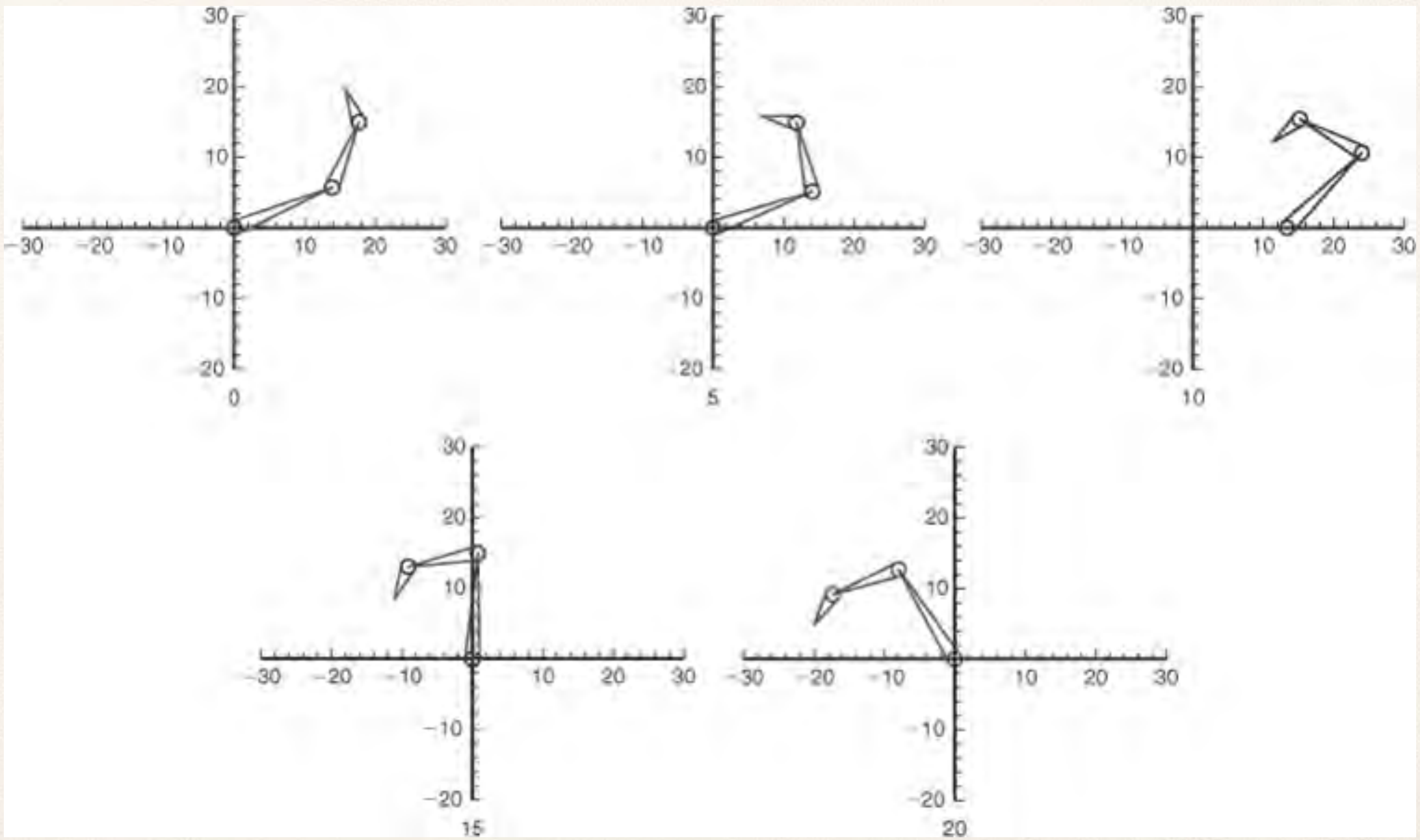


Figure 16.7 One iteration step towards the goal.

from CAIG Lab in Taiwan

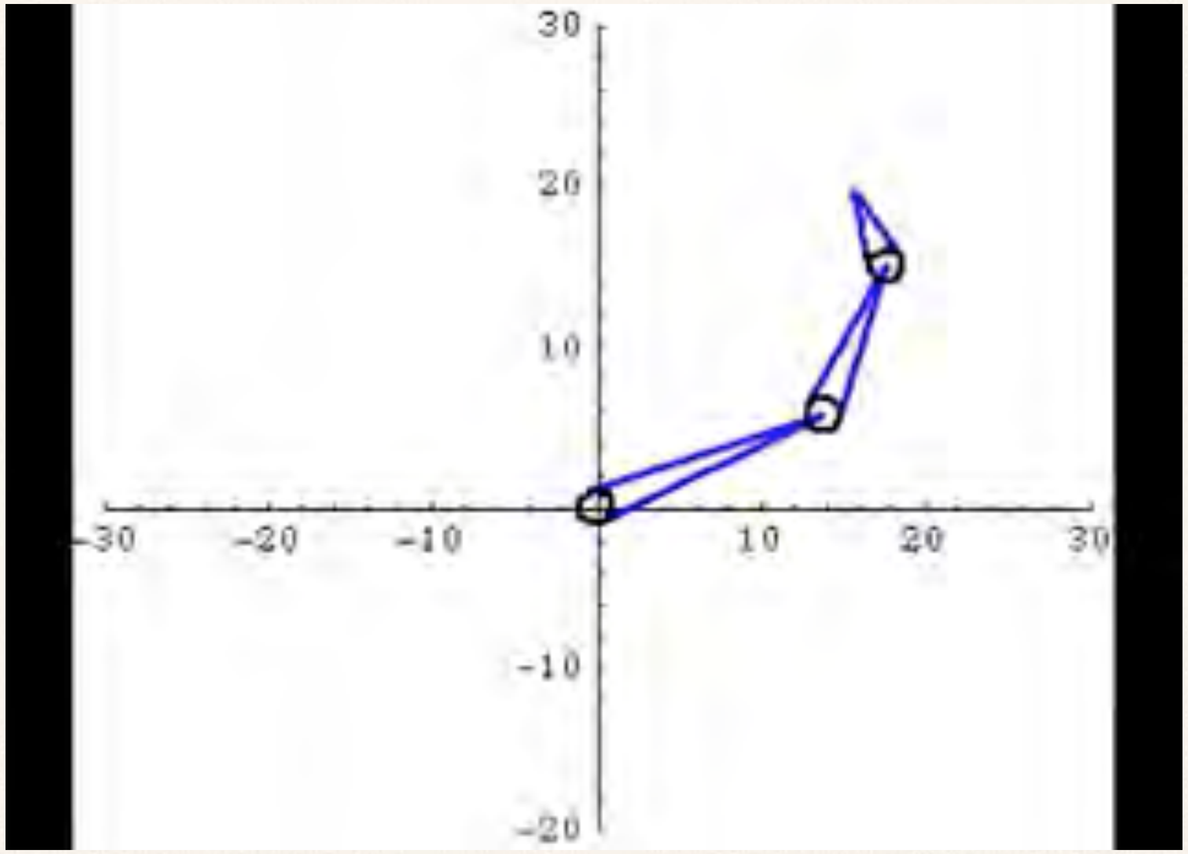
Redundant mechanisms

Total: 21 frames



R. Parent. Computer Animation: algorithms and techniques. Morgan Kauffman, 2008

Redundant mechanisms



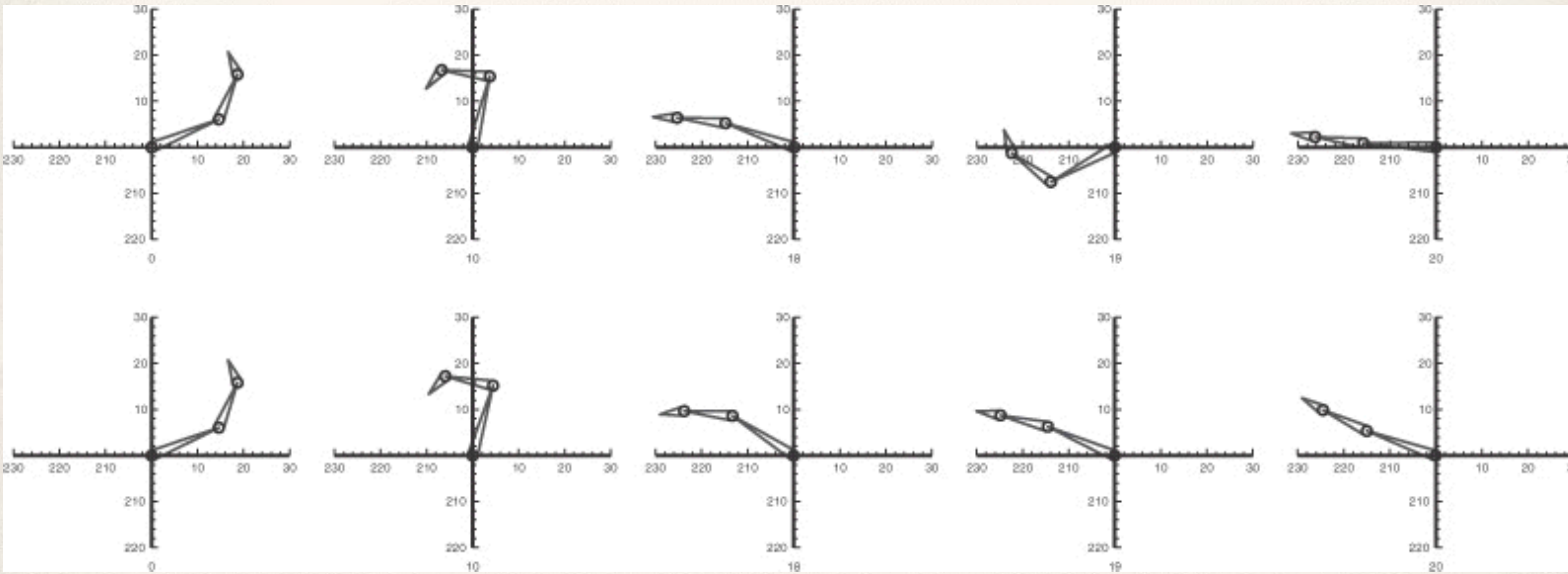
R. Parent. Computer Animation: algorithms and techniques. Morgan Kauffman, 2008

Handling singularities

- The Jacobian is only valid for the instantaneous configuration for which it is formed.
  - as soon as the configuration of the linkage changes, the Jacobian ceases to accurately describe the relationship between changes in joint angles and changes in end-effector position and orientation.
- A proposed solution to handling singularities is the damped least squares approach.
  - a user-supplied parameter is used to add in a term that reduces the sensitivity of the pseudoinverse.
  - behaves better in the neighborhood of singularities at the expense of rate convergence to a solution.

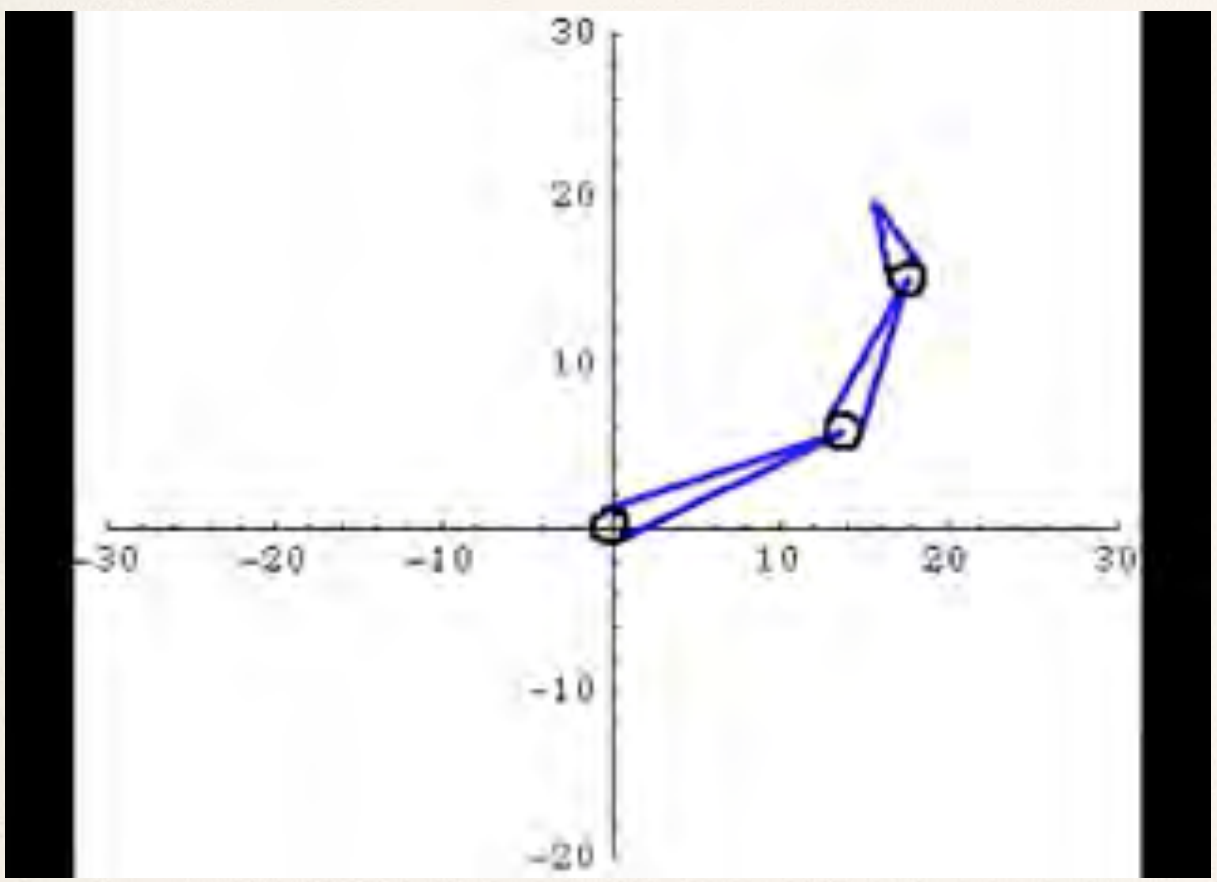
$$\dot{\mathbf{q}} = \mathbf{J}^T(\mathbf{J}\mathbf{J}^T + \lambda^2\mathbf{I})^{-1}\mathbf{v}_e$$

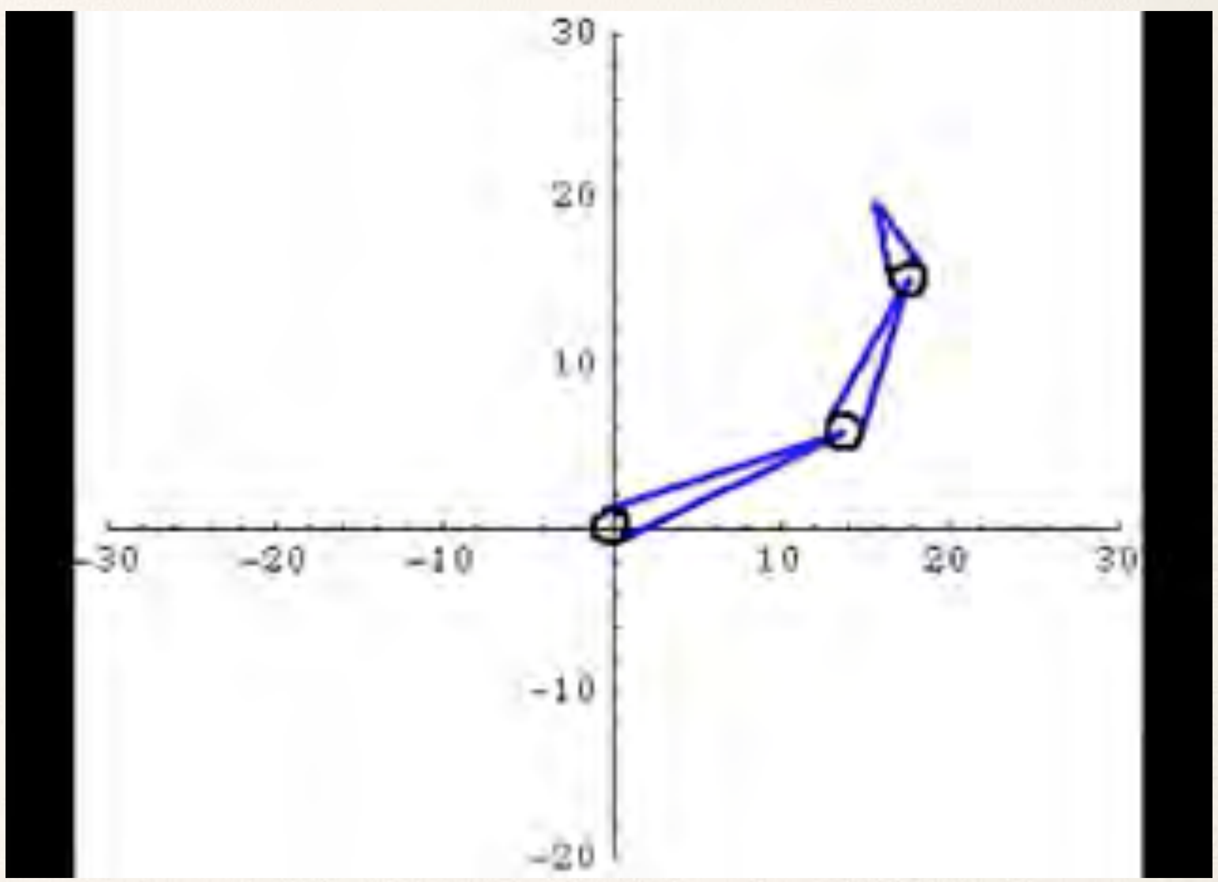
Damped least squares



R. Parent. Computer Animation: algorithms and techniques. Morgan Kauffman, 2008

Damped least squares





R. Parent. Computer Animation: algorithms and techniques. Morgan Kauffman, 2008

Adding more control

- The pseudoinverse computes one of many possible solutions.
- It minimizes joint angle rates but configurations do not correspond necessarily to the most natural poses.
- A control term can be added to the pseudoinverse Jacobian solution.
- The control term is used to solve to control angle rates with certain attributes.
- This term contributes nothing to the desired end-effector velocities (projector to the null space of the Jacobian):

$$\dot{\mathbf{q}} = \mathbf{J}^{\dagger}\mathbf{v}_e + (\mathbf{J}^{\dagger}\mathbf{J} - \mathbf{I})z$$

Control term adds zero linear velocity

A solution of the form: $\dot{\mathbf{q}} = (\mathbf{J}^{\dagger}\mathbf{J} - \mathbf{I})z$

when put into the formula: $\mathbf{v}_e = \mathbf{J}\dot{\mathbf{q}}$

$$\mathbf{v}_e = \mathbf{J}(\mathbf{J}^{\dagger}\mathbf{J} - \mathbf{I})z$$

after some manipulation, it can be shown that:

$$\mathbf{v}_e = (\mathbf{J}\mathbf{J}^{\dagger}\mathbf{J} - \mathbf{J})z$$

$$\mathbf{v}_e = (\mathbf{J} - \mathbf{J})z$$

$$\mathbf{v}_e = 0z$$

doesn't affect the desired configuration.

$$\mathbf{v}_e = 0$$

- But it can be used to bias the solution vector.

Adding more control

- To bias the solution toward specific joint angles, such as the middle joint angle between joint limits, z is defined as:

$$z = \alpha_i(\theta_i - \theta_{ci})^2$$

where,

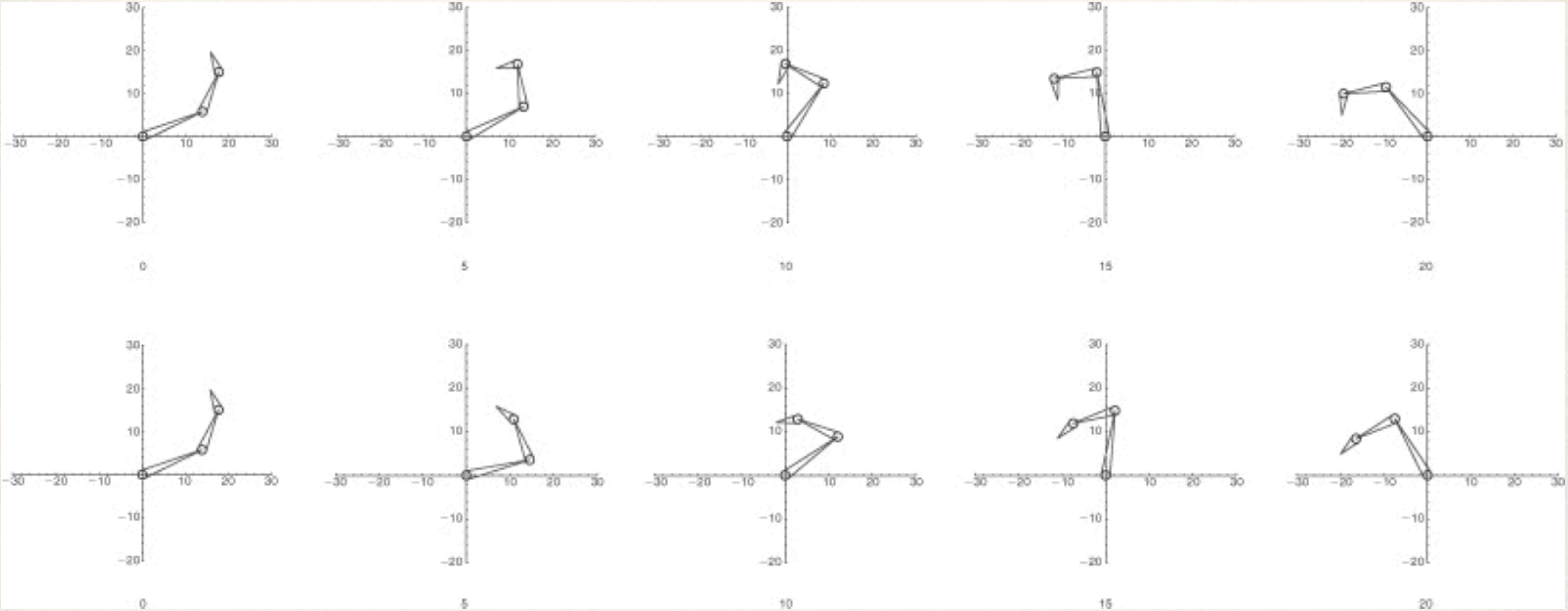
θ_i : current joint angles
θ_{ci}: desired joint angles
α: joint gains

- These are not hard constraints but the solution can be biased toward the middle values.
- Joint gain indicates the relative importance of the associated desired angle.
- The higher the gain, the stiffer the joint.
  - high - the solution will converge rapidly to the desired joint angle.
  - low - closer to conventional pseudoinverse solution.

Adding more control

joints biased to zero

α=[0.1 0.5 0.1]



α=[0.1 0.1 0.5]

R. Parent. Computer Animation: algorithms and techniques. Morgan Kauffman, 2008

An algorithm

$$\Delta\theta = J^{+}\Delta x + P_{N(J)}\Delta\alpha \qquad (2)$$

$$P_{N(J)} = I_n - J^{+}J \qquad (3)$$

with

Dq	**n**-dimensional posture variation
Dx	**m**-dimensional high priority constraints
J	**m** x **n** Jacobian matrix
J^{+}	**n** x **m** pseudo-inverse of J
$P_{N(J)}$	**n** x **n** projection operator on **N(*J*)**
I_n	**n** x **n** identity matrix.
Da	**n**-dimensional posture variation.

$$J = \sum_{i=1}^{r} \sigma_i u_i v_i^T \qquad (4)$$

$$J^{+} = \sum_{i=1}^{r} \frac{1}{\sigma_i} v_i u_i^T \qquad (5)$$

$$J^{+\lambda} = \sum_{i=1}^{r} \frac{\sigma_i}{\sigma_i^2 + \lambda^2} v_i u_i^T \qquad (6)$$

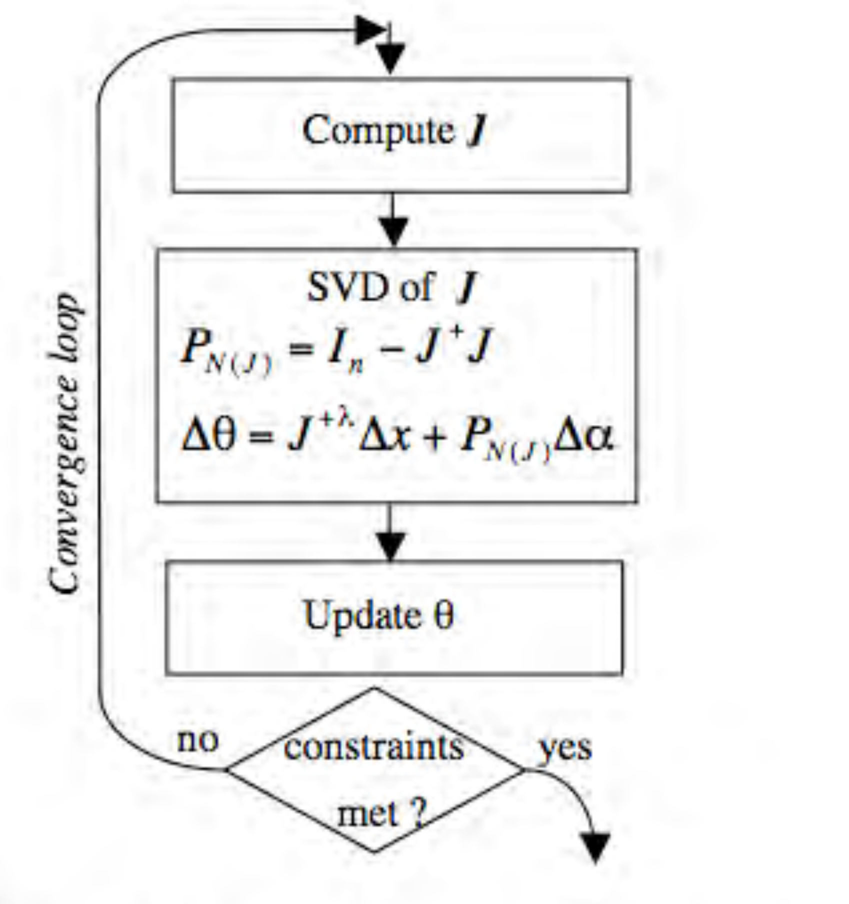


Figure 6: *Structure of the simplest IK algorithm*

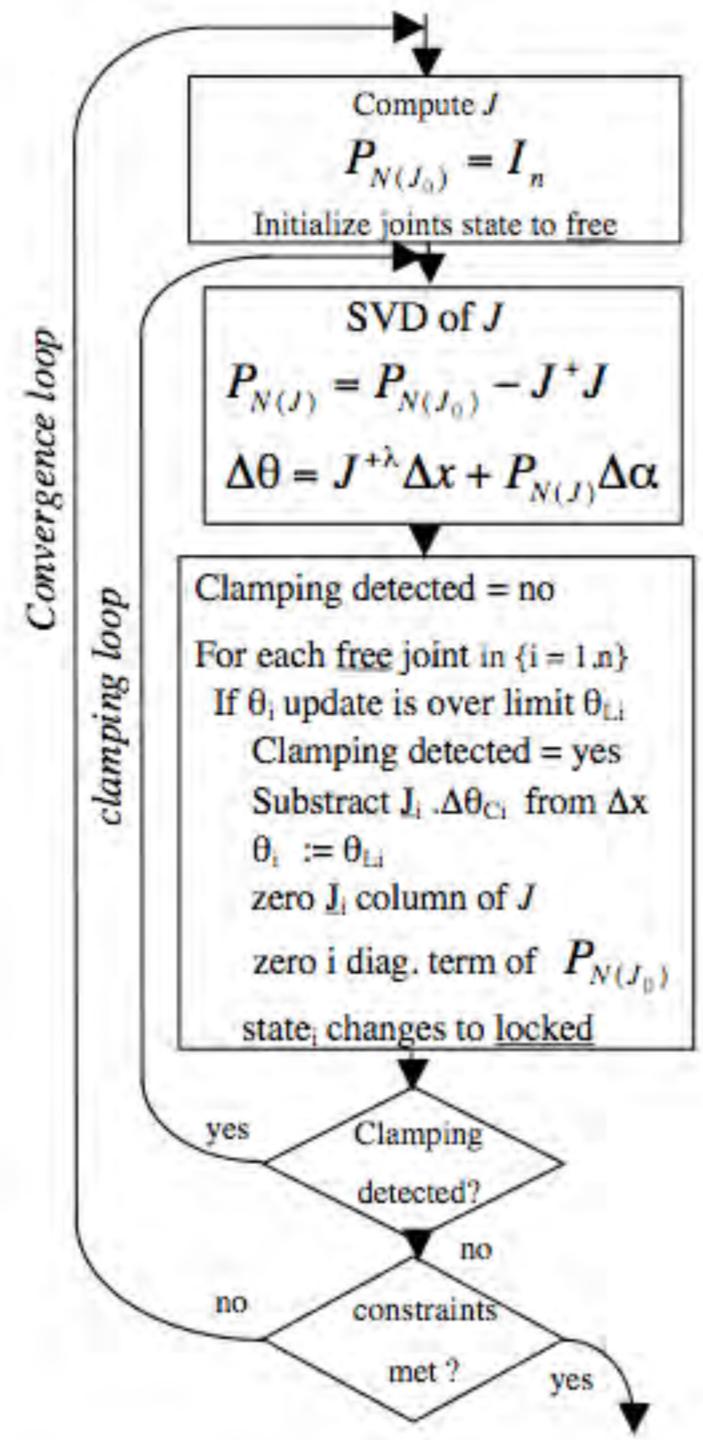
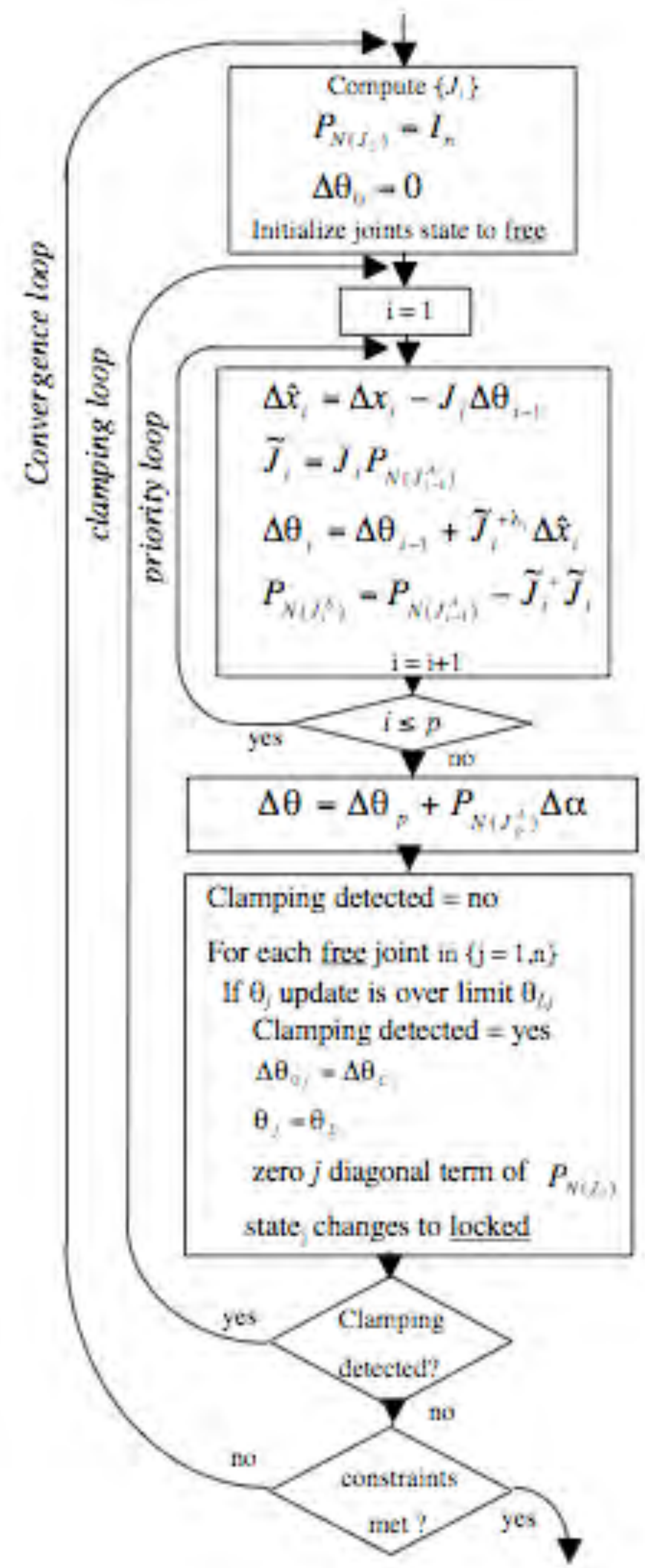


Figure 8: *IK loop including the management of inequality constraints (joint limits)*



from P. Baerlocher and R. Boulic. An Inverse Kinematic Architecture Enforcing an Arbitrary Number of Strict Priority Levels.

Figure 18: *Convergence of the prioritized IK successively after 5, 35 and 75 iterations.*

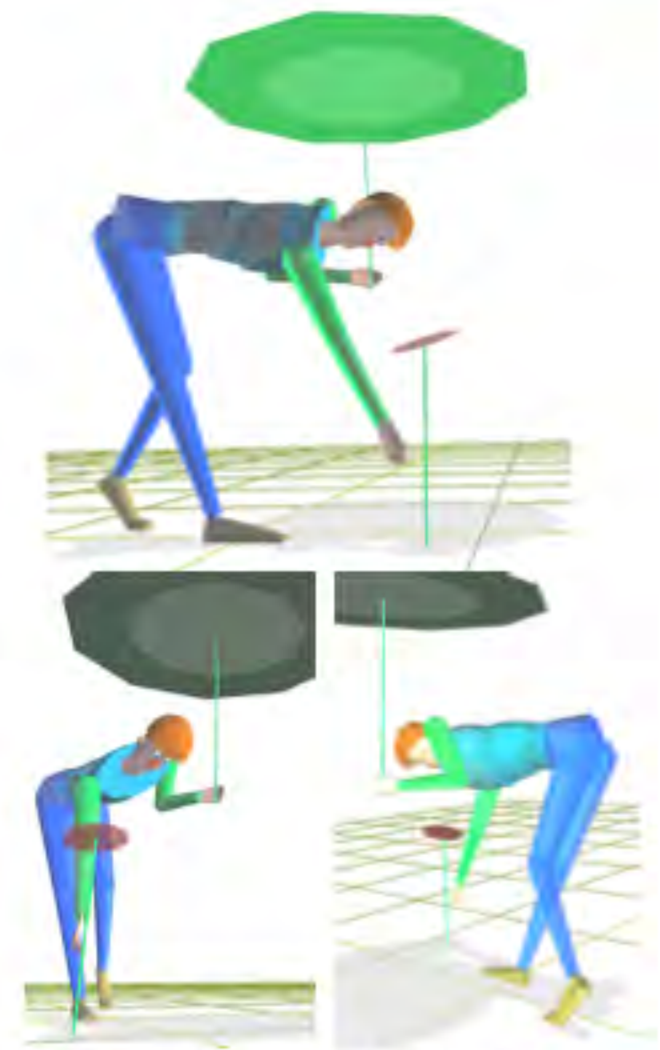


Figure 21: *A combined set of constraints involving balance, reach, gaze while holding the umbrella vertically (with four priority levels).*

from P. Baerlocher and R. Boulic. An Inverse Kinematic Architecture Enforcing an Arbitrary Number of Strict Priority Levels.

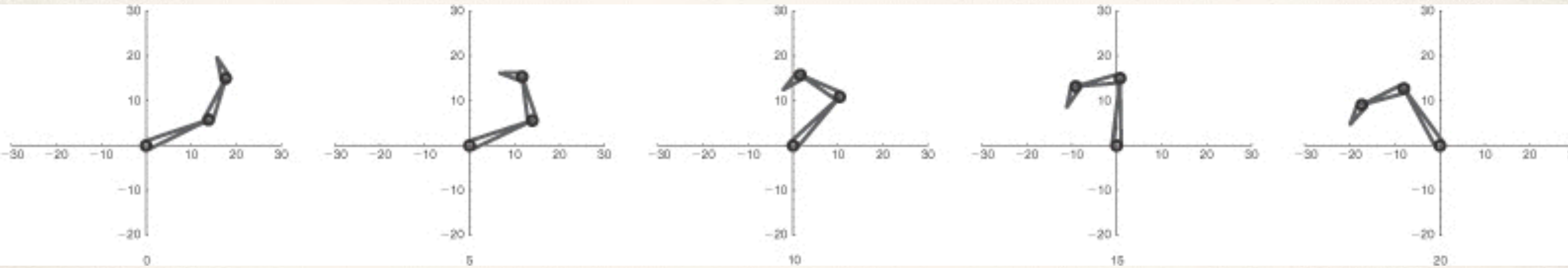
Jacobian transpose method

- Another way of determining the contribution of each instantaneous change vector is to form its projection onto the end-effector velocity vector.
- This entails forming the dot product between the instantaneous change vector and the velocity vector.
- Use the transpose of **J** instead of the inverse of **J**, i.e, set dq/dt equal to:

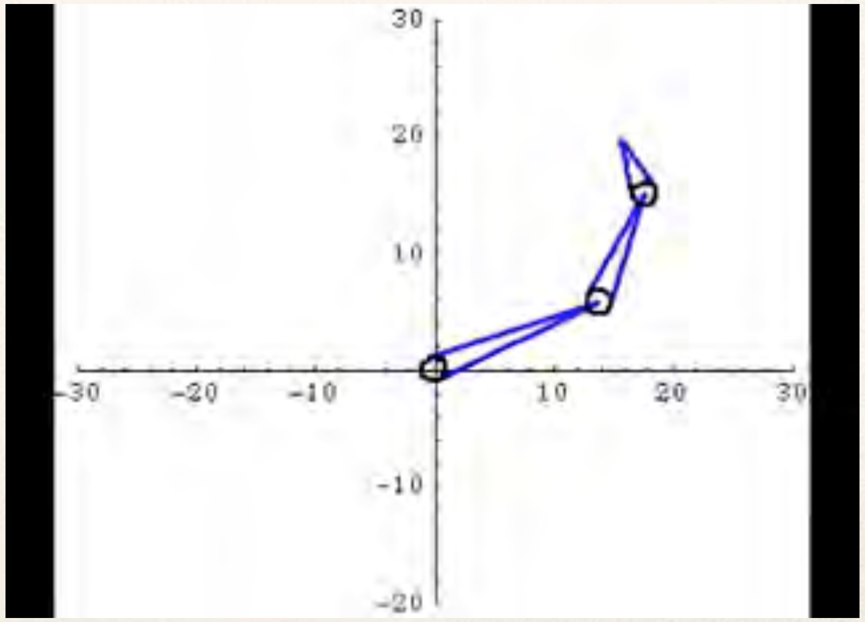
$$\dot{\mathbf{q}} = \alpha \mathbf{J}^T \mathbf{v}_e$$

for some appropriate scalar α.

Jacobian transpose method

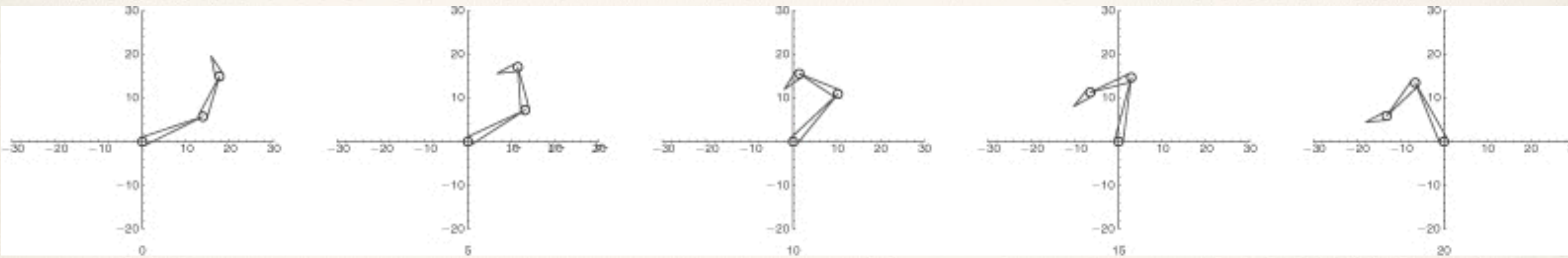


R. Parent. Computer Animation: algorithms and techniques. Morgan Kauffman, 2008

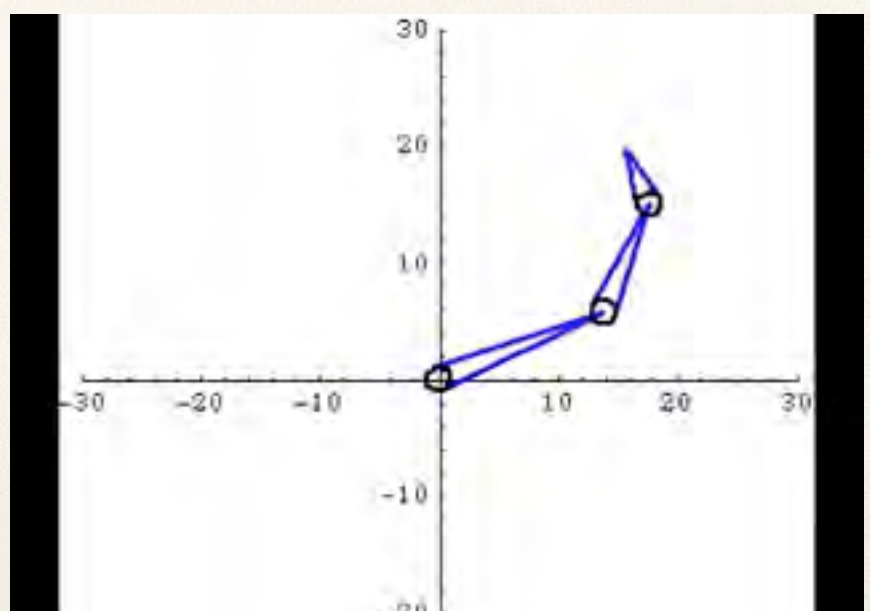


Cyclic Coordinate Descent method

- Consider each joint at a time, sequentially from the outermost inward.
- At each joint, an angle is chosen that best gets the end effector to the goal position.



R. Parent. Computer Animation: algorithms and techniques. Morgan Kauffman, 2008



To Read ...

- K. Yamane and Y. Nakamura. "**Natural Motion Animation through Constraining and Deconstraining at Will**". *IEEE Transactions on Visualization and Computer Graphics*, 9(3). 2003.
- K. Yamane, J.J. Kuffner and J.K. Hodgins. "**Synthesizing Animations of Human Manipulation Tasks**". *ACM Transactions on Graphics* (*SIGGRAPH 2004*). 23(3). 2004.
- K. Grochow, S.L. Martin, A. Hertzmann and Z. Popovic. "**Style-based Inverse Kinematics**". *ACM Transactions on Graphics* (SIGGRAPH 2004). 23(3). 2004.
- `