# Inverse kinematics and kinetics for virtual humanoids

R. Boulic[1] and R. Kulpa[2,3]

[1]VRLab - EPFL - Switzerland
[2]Bunraku - IRISA/INRIA Rennes - France
[3]M2S - University of Rennes 2 - France

**Abstract**

*The present tutorial deals with the use of inverse kinematics and kinetics for postural adaptations of virtual humanoids to different kind of constraints. It first proposes an overview of the problematic of this thematic and then of the existing techniques. Then it technically describes two key approaches: the prioritized inverse kinematics for accurate and realistic adaptation and a CCD-like algorithm based on groups for fast and realistic adaptation of hundreds of characters in real-time.*

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Animation

## 1. Keywords

Inverse kinematics, inverse kinetics, postural adaptation, numeric IK, prioritized IK, validity domain of the linearization, joint limits, singularity, pseudo-inverse, kernel, jacobian, balance control

## 2. Information about the tutorial

### 2.1. Contribution

Intuitive and efficient postural control is important to master, so this tutorial offers first an overview and a classification of the inverse kinematics and kinetics methods and of the respective advantages/drawbacks of various approaches (analytic, iterative, Jacobian transpose, pseudo-inverse, CCD...). Then it intuitively details the technical aspects of what make two key approaches very useful and how to implement them:

- the prioritized Inverse Kinematics can combine multiple conflicting constraints (useful for interaction in complex environments)
- the CCD combines both IK and balance control very efficiently (useful for large number of virtual humans)

### 2.2. Benefit for the audience

The tutorial is illustrated with abundant conceptual illustrations and animation examples to explain all elements, starting from the basics (effector, constraints, jacobians, convergence) to more difficult features such as how the priority levels are enforced, how to handle singularity and stability, why can the CCD be used for realistic human adaptation and can be combined with balance control. So this tutorial concerns beginners as well as advanced users that want to have extra technical information about how it works and how to implement it.

### 2.3. Necessary background

Basic knowledge of articulated structures animation (composition of transformations, notion of joint and joint limits), basic notion of Linear Algebra (rank, range space, null space, SVD). The course is tailored for a non-specialist audience interested in understanding and implementing numeric IK without enduring its pitfalls. Some technical parts of the presentation are also proposed for advanced users.

### 2.4. Course material

The course notes consists in:

- the present summary with a short list of key annotated references,
- the powerpoint handouts organized in four parts,
- two reference articles [BB04, KMA05],
- and animation examples presented during the course.

### 2.5. Presenters

**Ronan Boulic** is a senior researcher, lecturer and PhD advisor at the Swiss Federal Institute of Technology, Lausanne (EPFL). He is working in the Virtual Reality Lab and his research interests include 3D interactions, motion capture, virual prototyping, and realistic motion synthesis for virtual humans and robot. He received the PhD degree in Computer Science in 1986 from the University of Rennes, France, at the INRIA-IRISA research institute. He received the Habilitation degree from the University of Grenoble, France, in march 1995. Ronan Boulic is co-author of 102 research papers among which 27 appeared in international peer-reviewed journals. He is senior member of IEEE, senior member of ACM and member of Eurographics.

Ronan Boulic has been working in the area of Inverse Kinematics since 1985, first in Robotics at INRIA-Rennes, France then in Animation at EPFL, Switzerland. He was the advisor of five PhD students who have contributed to extend the IK technology in various directions:

- Balance control (Inverse Kinetics) with Ramon Mas (1996)
- Efficient Prioritized IK with Paolo Baerlocher (2001)
- Modeling fatigue with Inmaculada Rodriguez Santiago (2003)
- Motion Editing with Benoit Le Callennec (2006)
- Movement recovery from Mocap data and collision avoidance with Manuel Peinado (on-going)

**Richard Kulpa** is an Assistant Professor at the M2S laboratory (previously LPBEM) of the University of Rennes 2, France. He is also an external collaborator of the Bunraku team of the IRISA-INRIA Rennes research institute. His research interests include biomechanics, realistic humanoids animation and virtual reality.

He worked since 1996 as a research engineer in the SIAMES team (now Bunraku) where he worked on behavioral animations of characters in informed environments in collaboration with French multimedia companies. Since 2000, in parallel to these previous works, he has integrated the LPBEM laboratory where he worked on biomechanics and the understanding of the human movement. He incorporated biomechanical knowledge in IK methods in order to produce fast and realistic animations. At last, he received his PhD in 2005 for a work that combines biomechanics and iterative inverse kinematics and kinetics solver in order to have realistic animation of hundreds of characters in real-time. He received a price at Eurographics 2005 concerning this work. By the way, he co-created the MKM animation engine that incorporates all these works. This software is currently used by an international company.

## 3. Content of the tutorial

### 3.1. Plan

**First module:**

1. Part A - Overview
   a. Motivation
   b. Problem
      i. Definition of inverse kinematics
      ii. Definition of the constraints
   c. Overview of the IK methods
      i. A taxonomy of the IK methods
      ii. Analytical method
      iii. Iterative methods
2. Part B - Iterative methods (pseudo-inverse based)
   a. Concept of prioritized control
   b. From one to P priority levels
      i. General form
      ii. Incremental constraint achievement
      iii. Example 1: the simplest redundant case
      iv. Example 2: two prioritized constraints
      v. Generalization to multiple priority levels
      vi. Convergence: weighted vs prioritized
      vii. Fostering synergies
      viii. Handling singularities with DLS
      ix. Application: collision avoidance
      x. Conclusion

**Second module:**

1. Part C - Iterative methods (CCD-based)
   a. CCD
   b. Hybrid CCD and analytical
      i. Representation of data
      ii. Kinematic solver
      iii. Kinetic solver
      iv. Kinematic and kinetic solver
      v. results
      vi. Conclusion
2. Part D - Conclusion

### 3.2. First module

The first module is organized in two sections, an overview and a detailed presentation of the prioritized IK technology.

The overview is intended to set the stage, recall the terminology, offer a quick survey of the IK landscape and highlight the relative advantages of the two complementary Inverse Kinematics technologies detailed afterward.

Indeed, mastering the full body postural control of complex articulated structures such as human-like bodies is intrinsically difficult for two reasons:

- First the postural space is large, and usually much larger than the "action" space (e.g. reach the door knob with the right hand); as a consequence an infinite number of valid postures exists but retaining the "best" one is uneasy to express.
- Second, being humans ourselves we are very sensitive to the posture properties, e.g. whether it is balanced or not; hence uncorrect, or uncomfortable, or unusual postures cannot be accepted as they appear as unbelievable. Similarly, a partial postural control may produce uncorrect postures for the evaluation of complex systems (e.g. in Virtual Prototyping) and lead to design errors.

This tutorial targets the problematic of the intuitive production of full body posture from the specification of high-level goal-oriented constraints such as "remain balanced", "reach this location with this body part" (cf. figure 1), "look at", "orient"... For that purpose the definitions of key concepts are briefly recalled, i.e. kinematic chain, effector, constraints, direct kinematic model, inverse kinematic model. A brief survey then proposes a classification of the existing methods according to their convergence structure [Chi96, Wel93, ZB94, BB04, KMA05]. The major IK approaches are compared in terms of advantages and drawbacks; our major criteria for this comparison is complementary such as versatility and efficiency. Regarding data-driven IK approaches such as [GMHP04], we briefly compare them with other approaches but the focus of this tutorial is data-independent IK technologies.

**Figure 1:** *Fast posture adjustment to the ground with an hybrid CCD IK approach [KMA05].*

The second part of the first module provides an in-depth presentation of the Prioritized Inverse Kinematics. This technology has been retained for the detailed presentation as it allows the enforcement of strict priority levels to sort the various constraints according to their importance. The interest of this strategy is conceptually illustrated before entering the description of simple case studies highlighting the properties of the pseudo-inverse and the associated Null space projection operator. The key theoretical formula is explained graphically on the two-priority levels case study. It is then generalized to an arbitrary number of priority levels with a special attention to the algorithm implementation and its computing cost. The description includes the joint limits management through a clamping loop (cf. figure 2).

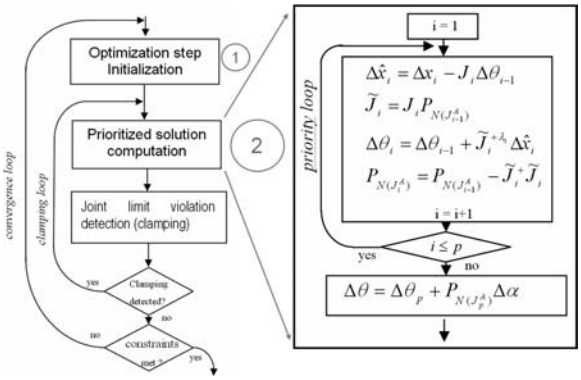



**Figure 2:** *Accumulating the contributions of each priority levels in the IK priority loop.*

The following discussion and examples illustrate the convergence behavior resulting from prioritized IK. For example, it produces intermediate believable postures as the most important constraints are enforced first, and generally very quickly. This is in contrast with the weighted minimization approach where no constraint is absolutely guaranteed when they conflict with one other. This can lead to local minima or unplausible intermediate/final postures. Another advantage of the prioritized architecture is the possibility to recruit the full body for achieving all constraints resulting in synergistic solutions (cf. figure 3) that could not be found otherwise [BB04].

We discuss then on how to exploit the SVD to handle the critical issue of singularities (geometric and algorithmic). Although the Damped Least Square inverse [BK05] is the key for ensuring the stability, it should not be exploited to enforce priority levels (as often reported in the literature). The correct formulation ensuring both stability and priority levels is highlighted.

After a few examples of reachable space identification, the section ends with the management of collision avoidance (cf. figure 4) through the dynamic creation of partial repulsion effectors [BB04]. Automatic local collision anticipation and avoidance concludes the module [PBCM05].

### 3.3. Second module

The part C of the second module provides an in-depth presentation of the CCD-like algorithms and the way to produce realistic adaptations of many characters in real-time.

The CCD (Cyclic Coordinate Descent) is an iterative method that modifies only one DOF at a time. This way, the jacobian is simplified as a vector and the inversion is cheap.

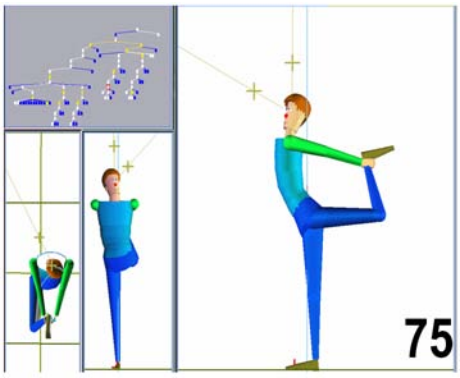



**Figure 3:** *Full body postural synergy achieving a dance posture where numerous joint limits are enforced with the inequality constraint management (clamping loop).*

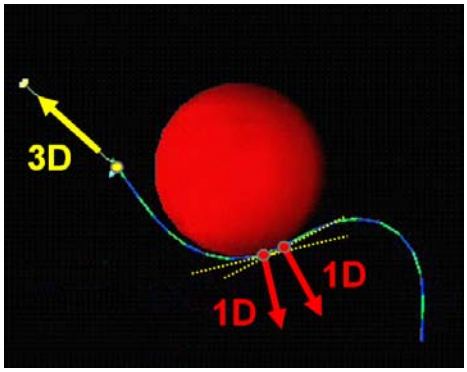



**Figure 4:** *Dynamic creation of 1D repulsion effectors to ensure obstacle avoidance.*

A geometrical version of this method is also available rotating one joint at a time. This technique is consequently very fast and can easily handle joint limits but has some important drawbacks that makes it incompatible with human postural adaptation. Indeed, the repartition of the deformation is not homogeneous leading to unrealistic postures: first joints are more modified than the following ones.

We propose to take advantage of this drawback by using analytical solutions on the chain ends and by dividing the skeleton into groups to order the adaptations between these groups and within them (ordering the joints). To this end, two representations of data are detailed:

- Constraints: in order to take user high-level constraints into account while having a generic and efficient inverse kinematics and kinetics solver, two levels of constraints are considered: the external one represents the user constraints while the internal one is a unique representation handled by the algorithm.
- Skeleton: in order to create analytical solutions that works with weighted or prioritized constraints, a suitable representation of the skeleton for adaptation is needed. A normalized representation is detailed with some examples of possible easy adaptations.

Based on these representations, analytical solutions are used for the limbs decreasing the number of segments in the kinematic chain and consequently increasing the performance of the method. It also allows to find instantaneously their best configuration according to the constraints. This means that the adaptation applied to the remainder of the body is minimized. Combined with a correct adaptation order for the iterative algorithm, realistic postures can be found very rapidly.

This correct order is obtain by dividing the body into groups. The goal is to take advantage of the difference between human body parts (more or less movable and heavy). Then, a common hierarchy of these groups is proposed to order the adaptation between the groups for the kinematic adaptation as well as for the kinetic adaptation. The correct adaptation order for inverse kinematics goes from the groups with the largest amplitude to the lowest one. On the opposite, inverse kinetics adaptation modifies the groups from the heaviest to the lightest ones. A detailed algorithm is proposed in the tutorial.

Finally, during kinematics, after adapting all the groups, root adaptation is made to manage the constraints placed on the pelvis but also to reach far targets. Indeed, the global error between the constraints and their targets can be reduced translating the root. Using a percentage parameter, it allows to have different adaptations: high percentage for decided characters that want to reach their goals and low percentage for tired characters for example. Additional laws can also be added to have different adaptations such as the joint range to simulate the suppleness as suggested in figure 5.

**Figure 5:** *Additional laws allow to have different adaptations.*

To have realistic adaptation, the balance of the character must be controlled: in 3D in aerial phases and in 2D when the character is on the ground. In this last case, the goal is to ensure that the projections of the current and the desired COM are at the same position. To do so, we propose an inverse kinetics method based on exactly the same representation and algorithm than described before. As the position of the center of mass (COM) is obtained as a weighted sum of the local COM of the segments, its computation can easily be subdivided into groups. According to the previously

defined hierarchy of groups, the inverse kinetics solver proposed can iterate on these groups in order to verify the desired COM. The error on the COM position is consequently transformed as an error on the local COM of the currently modified group. Placing the group correctly means that the global COM is at the desired position. As for kinematic adaptation, analytical solutions are used for the limbs in order to place them instantaneously at the correct posture in order to verify the local COM. For the other groups, rotations are used instead. Finally, the root adaptation consists in its translation to correct a percentage of the error of the global COM position. This percentage is really important since it allows to have different kind of adaptations mixing those of the root or/and the groups (cf. figure 6).

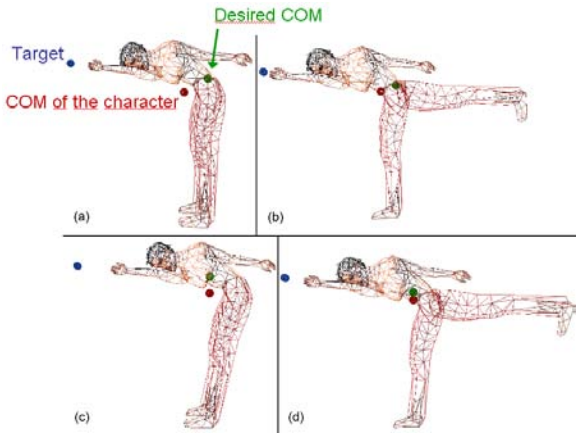



**Figure 6:** *Different kinetic adaptations: a) reference posture, b) using groups, c) using root, d) using both.*

Finally, inverse kinematics and kinetics are inserted inside a unique iterative loop. Of course, in order to be compatible with the kinematic adaptation, only the groups that are not already constrained can be used for inverse kinetics. And only a subset can be selected by the user in order to have different kind of adaptations. The complete algorithm always converges because the modified groups are distinct and the kinematic and kinetic adaptations are antagonistic ones. Demonstrations are proposed in the tutorial for postural and motion adaptation.

Finally, the part D of this tutorial quickly summarizes the different IK methods and their advantages/drawbacks. It also indicates how these IK methods can also be used for motion adaptations.

## 4. Structured bibliography

Part A: Overview, Basis and Taxonomy

- CS IK course from Steve Rotenberg (USD) [Rot05]
- Jacobian transpose and CCD approaches [Wel93]
- Weighted IK methods for Virtual human postural control [BMW87, PZB90, PB91, BPW93, ZB94]
- IK control of the center of mass [BMT96, BMST97]
- Analytic IK [Chi96, TB96, TGB00]
- Computer Puppetry [SLSG01]
- Motion-Model based IK [GMHP04]

Part B: Prioritized Inverse Kinematics

- First form of Prioritized IK by Liegeois [Lie77]
- Two priority levels by Hanafusa et al. [HYN81]
- Arbitrary number of priority levels [SS91]
- Prioritized IK with reduced computing cost [BB04]
- Damped Pseudo-Inverse for singularity management [Mac90, BK05]

Part C: CCD and fast Hybrid approach

- Origin of the CCD [Lue84, WC91]
- CCD applied on human [BMB87]
- CCD using geometrical resolution [Lan98]
- Fast IK and balance control [KMA05, KM05]

Part D: Conclusion

- IK used to adapt motions [BW95, Gle97, Gle01, lCB04]
- Prioritized IK demonstrations
  *http://vrlab.epfl.ch/ boulic/demos.html*
- Fast CCD-like IK demonstrations
  *http://www.irisa.fr/bunraku/Richard.Kulpa*

## 5. Acknowledgments

Research on Prioritized Inverse Kinematics led by Ronan Boulic have been realized with the support of the Swiss National Science Foundation, the European Union Network of Excellence ENACTIVE, the University of Alcala and EPFL.

Research on Hybrid-CCD Inverse Kinematics and Kinetics led by Richard Kulpa have been realized with the support of the French National Science Foundation, the "Conseil Regional and General" of Brittany and Rennes city.

## References

[BB04] BAERLOCHER P., BOULIC R.: An inverse kinematics architecture enforcing an arbitrary number of strict priority levels. *The Visual Computer 20*, 6 (2004), 402–417.

[BK05] BUSS S., KIM J.: Inverse kinematics with selectively damped least squares. *Journal of Graphics Tools 10*, 3 (2005), 37–49.

[BMB87] BADLER N. I., MANOOCHEHRI K. H., BARAFF D.: Multi-dimensional input techniques and articulated figure positioning by multiple constraints. In *SI3D'86: Proceedings of the 1986 workshop on Interactive 3D graphics* (New York, NY, USA, 1987), ACM Press, pp. 151–169.

[BMST97] BOULIC R., MAS-SANSO R., THALMANN D.: Complex character positioning based on a compatible flow model of multiple supports. *IEEE Transactions on Visualization and Computer Graphics 3*, 3 (jul–sep 1997), 245–261.

[BMT96] BOULIC R., MAS R., THALMANN D.: A robust approach for the control of the center of mass with inverse kinetics. *Computers and Graphics 20*, 5 (Sept. 1996), 693–701.

[BMW87] BADLER N. I., MANOOCHEHRI K. H., WALTERS G.: Articulated figure positioning by multiple constraints. *IEEE Computer Graphics and Applications 7*, 6 (June 1987), 28–38.

[BPW93] BADLER N. I., PHILLIPS C. B., WEBBER B. L.: *Simulating humans: computer graphics animation and control*. Oxford University Press, Inc., New York, NY, USA, 1993.

[BW95] BRUDERLIN A., WILLIAMS L.: Motion signal processing. In *Proceedings of SIGGRAPH'95* (Aug. 1995), Computer Graphics Proceedings, Annual Conference Series, pp. 97–104.

[Chi96] CHIN K. W.: *Closed-form and generalized inverse kinematic solutions for animating the human articulated structure*. Master's thesis, Curtin University of Technology, 1996.

[Gle97] GLEICHER M.: Motion editing with spacetime constraints. In *Symposium on Interactive 3D Graphics'97* (Apr. 1997), pp. 139–148.

[Gle01] GLEICHER M.: Comparing constraint-based motion editing methods. *Graphical models 63*, 2 (2001), 107–134.

[GMHP04] GROCHOW K., MARTIN S. L., HERTZMANN A., POPOVIĆ Z.: Style-based inverse kinematics. *ACM Trans. Graph. 23*, 3 (2004), 522–531.

[HYN81] HANAFUSA H., YOSHIKAWA T., NAKAMURA Y.: Analysis and control of articulated robot with redundancy. In *IFAC, 8th Triennal World Congress* (1981), vol. 4, pp. 1927–1932.

[KM05] KULPA R., MULTON F.: Fast inverse kinematics and kinetics solver for human-like figures. In *Proceedings of IEEE Humanoids* (Tsukuba, Japan, december 2005), pp. 38–43.

[KMA05] KULPA R., MULTON F., ARNALDI B.: Morphology-independent representation of motions for interactive human-like animation. *Computer Graphics Forum, Eurographics 2005 special issue 24*, 3 (2005), 343–352.

[Lan98] LANDER J.: Oh my god, i inverted kine! *Game Developer Magazine* (Sept. 1998).

[lCB04] LE CALLENNEC B., BOULIC R.: Interactive motion deformation with prioritized constraints. In *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation* (New York, NY, USA, 2004), ACM Press, pp. 163–171.

[Lie77] LIEGEOIS A.: Automatic supervisory control of the configuration and behevior of multibody mechanisms. *IEEE Transactions on Systems, Man and Cybernetics SMC-7*, 12 (Dec. 1977).

[Lue84] LUENBERGER D. G.: *Linear and nonlinear programming*. Addison-Wesley, 1984.

[Mac90] MACIEJEWSKI A. A.: Dealing with the ill-conditioned equations of motion for articulated figures. *IEEE Computer Graphics and Applications 10*, 3 (May 1990), 63–71.

[PB91] PHILLIPS C. B., BADLER N. I.: Interactive behaviors for bipedal articulated figures. *Proceedings of SIGGRAPH'91 on Computer Graphics 25*, 4 (July 1991), 359–362.

[PBCM05] PEINADO M., BOULIC R., CALLENNEC B. L., MEZIAT D.: Progressive cartesian inequality constraints for the inverse kinematics control of articulated chains. *Short Presentation Proc. of Eurographics'05 4656* (2005).

[PZB90] PHILLIPS C. B., ZHAO J., BADLER N. I.: Interactive real-time articulated figure manipulation using multiple kinematic constraints. *Symposium on Interactive 3D Graphics'90 24*, 2 (Mar. 1990), 245–250.

[Rot05] ROTENBERG S.: Inverse kinematics, part 1 and 2. *http://graphics.ucsd.edu/courses/cse169_w06* (2005).

[SLSG01] SHIN H. J., LEE J., SHIN S. Y., GLEICHER M.: Computer puppetry: An importance-based approach. *ACM Transactions on Graphics 20*, 2 (2001), 67–94.

[SS91] SICILIANO B., SLOTINE J.-J.: A general framework for managing multiple tasks in highly redundant robotic systems. In *ICAR'91* (1991), vol. 2, p. 1211Ű1215.

[TB96] TOLANI D., BADLER N. I.: Real-time inverse kinematics of the human arm. *Presence 5*, 4 (1996), 393–401.

[TGB00] TOLANI D., GOSWAMI A., BADLER N. I.: Real-time inverse kinematics techniques for anthropomorphic limbs. *Graphical Models 62*, 5 (2000), 353–388.

[WC91] WANG L.-C. T., CHEN C. C.: A combined optimization method for solving the inverse kinematics problem of mechanical manipulators. *IEEE Transactions on Robotics and Applications 7*, 4 (Aug. 1991), 489–499.

[Wel93] WELMAN C.: *Inverse Kinematics and Geometric Constraints For Articulated Figure Manipulation*. PhD thesis, Simon Fraser University, 1993.

[ZB94] ZHAO J., BADLER N. I.: Inverse kinematics positioning using nonlinear programming for highly articulated figures. *ACM Transactions on Graphics 13*, 4 (Oct. 1994), 313–336.

# Inverse kinematics and kinetics for virtual humanoids

Ronan Boulic

VRLab – EPFL
http://vrlab.epfl.ch

Richard Kulpa

Bunraku – INRIA / IRISA
www.irisa.fr/bunraku

M2S – Univ. Rennes 2
www.uhb.fr/labos/m2s



# Plan

- Part A - Overview
  - Motivation
  - Problem
  - Overview of the IK methods
- Part B - Iterative methods (pseudo-inverse based)
  - Concept of prioritized control
  - From one to P priority levels
- Part C - Iterative methods (CCD-based)
  - CCD
  - Hybrid CCD and analytical
- Part D - Conclusion



# Part A - Overview

- Motivation
- Problem
  - Definition of inverse kinematics
  - Definition of the constraints
- Overview of the IK methods
  - A taxonomy of the IK methods
  - Analytical method
  - Iterative methods



# Motivation

- Many applications use virtual humanoids
- Need realistic postures and motions
  - Postures
    - Ergonomics
    - Maintainability
  - Motions
    - Virtual reality
    - Video games
    - Multimedia productions

# Motivation

- Can capture postures and motions
- Problem: dependent on original conditions of capture (e.g. morphology, position of the target during grasping…)

⇒ Need to adapt postures and motions



# Motivation

- Inverse kinematics adapts postures
- Motions can be considered as a succession of postures

⇒ So inverse kinematics can also be used for motion adaptation
  - Directly for small adaptations
  - Using filtering for large adaptations



# Motivation

- This tutorial deals with inverse kinematics methods to adapt postures
  - Advantages
  - Drawbacks
- Moreover, it deals with inverse kinetics
  - Takes masses into account
  - Preserves balance



# Part A - Overview

- Motivation
- Problem
  - Definition of inverse kinematics
  - Definition of the constraints
- Overview of the IK methods
  - A taxonomy of the IK methods
  - Analytical method
  - Iterative methods

# Definition of Inverse Kinematics

- Problem: given the goal position and/or orientation of some body parts, noted $\mathbf{x_g}$, find the "best" joint state $\boldsymbol{\theta}$ of the full body

- Direct Kinematic model

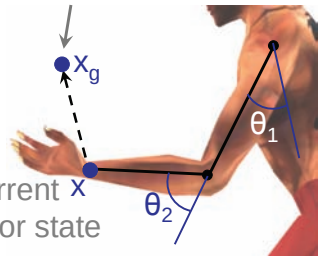
$x = F(\theta,$ body skeleton$)$

Effector state

Joint parameters

Kinematic chain

(here a position constraint goal)





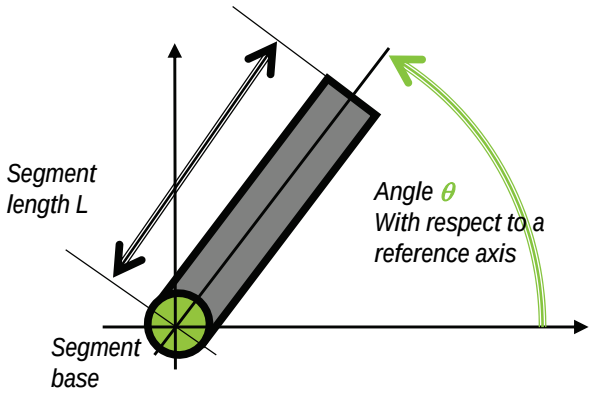

# Illustration on a 2D chain

- Kinematic chain F composed of
  - Segments
  - Joints (considered as ideal)

- Each joint angle is relative to the parent segment
- For the first segment the joint angle is relative to a reference axis

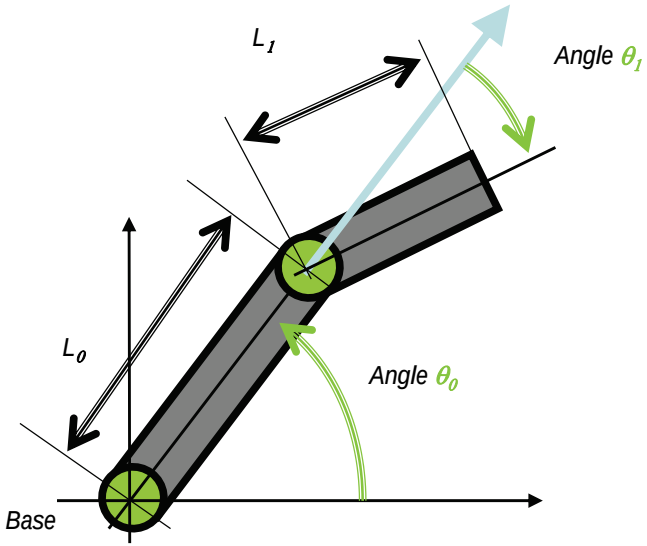


# First segment

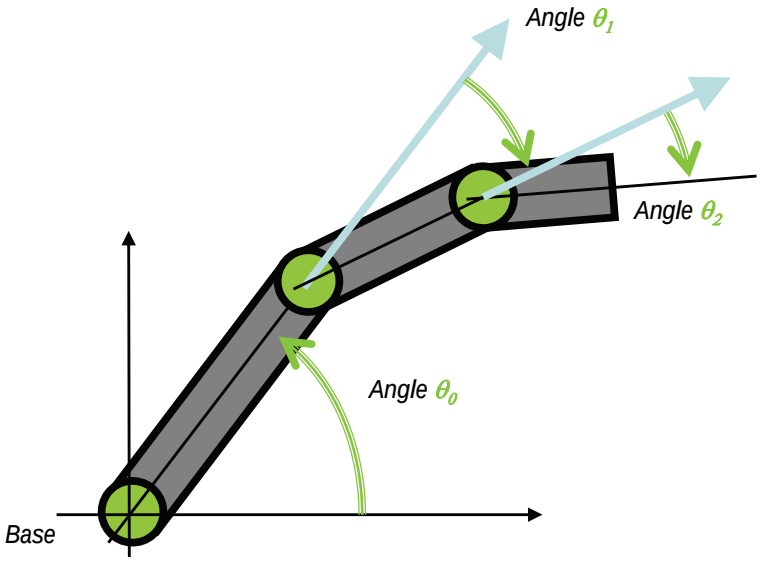





# Second segment

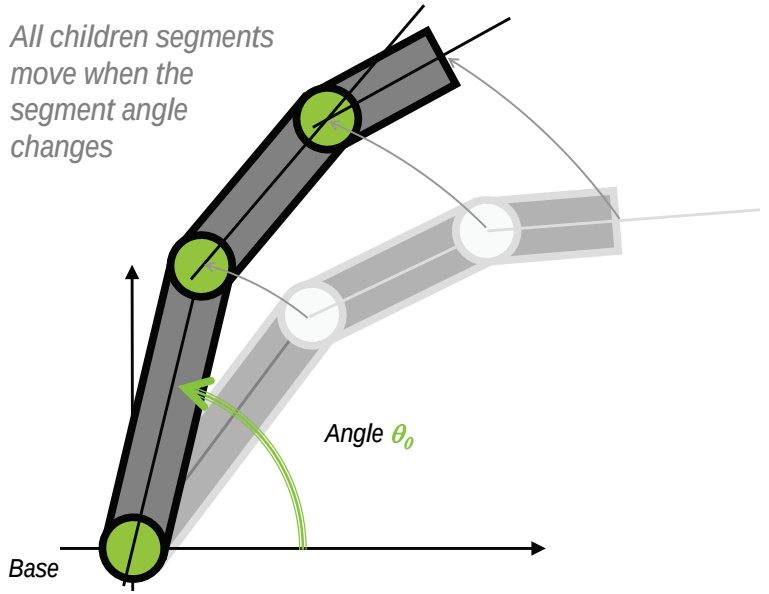


# Third segment



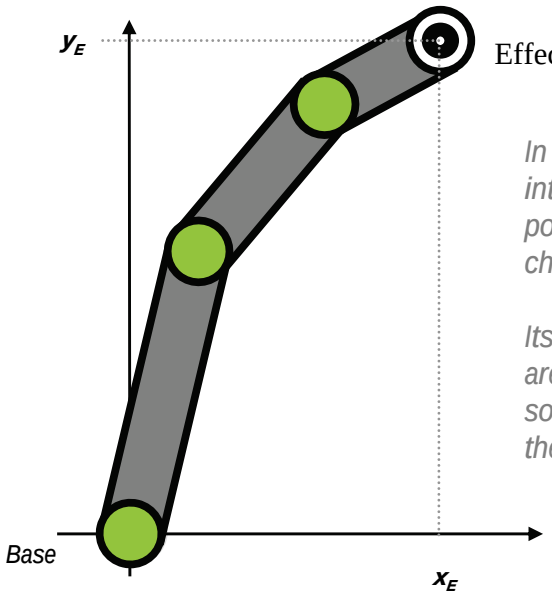

13

*All children segments move when the segment angle changes*




*Variation of angle $\theta_0$*

14



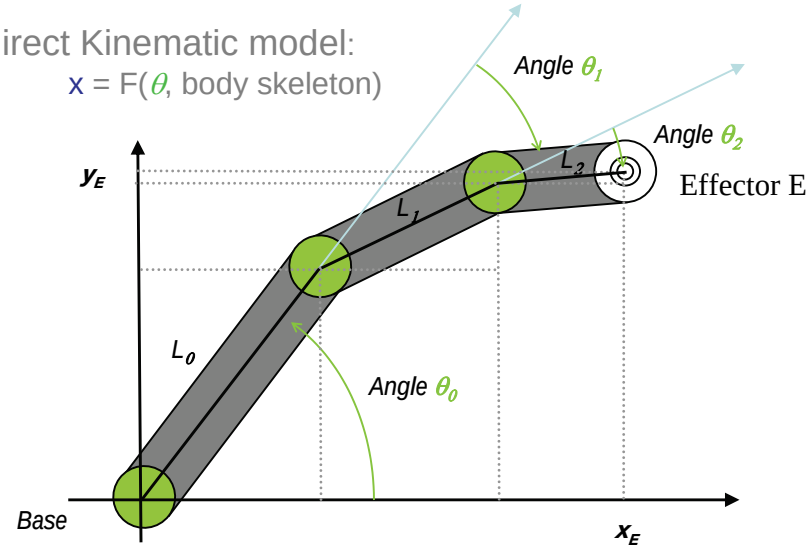

*In this example we are interested to control the position of the tip of the chain, called the Effector*

*Its coordinates X ($x_E$,$y_E$) are important to specify some task to achieve with the chain*

15

## Direct Kinematic model:

x = F($\theta$, body skeleton)




$$x_E = L_0 \cos\theta_0 + L_1 \cos(\theta_0 + \theta_1) + L_2 \cos(\theta_0 + \theta_1 + \theta_2)$$
$$y_E = L_0 \sin\theta_0 + L_1 \sin(\theta_0 + \theta_1) + L_2 \sin(\theta_0 + \theta_1 + \theta_2)$$

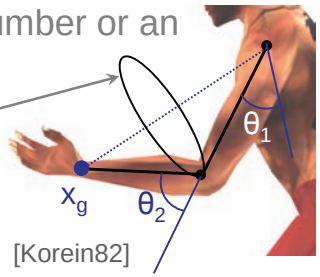
16

# Definition of Inverse Kinematics

- What is $\theta$ for a goal value $x_g$ ?
- Inverse Kinematic model

  $\theta = F^{-1}(x_g,$ body skeleton$)$
- Generally redundant: dim($x_g$) < dim($\theta$)
  - Less constraints than degrees of freedom
  - Either no solution, a finite number or an infinite number

Possible positions for the elbow



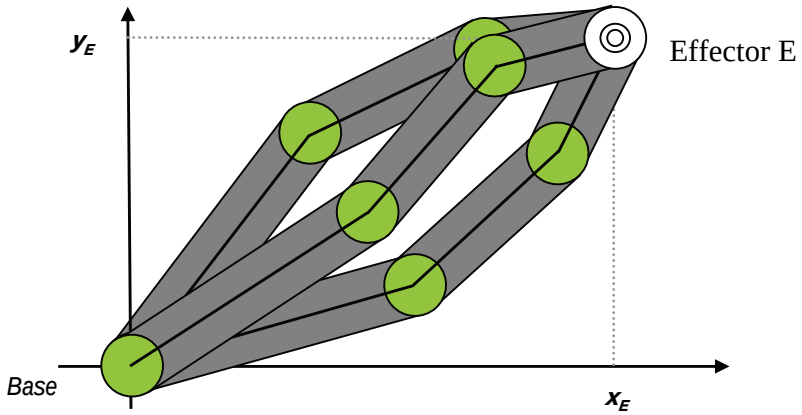

[Korein82]





Inverse Kinematic model:

$\theta = F^{-1}(x_g,$ body skeleton$)$ ?




Either find directly the most appropriate solution
Or choose it amongst all the correct solutions



# Part A - Overview

- Motivation
- Problem
  - Definition of inverse kinematics
  - ⇨ Definition of the constraints
- Overview of the IK methods
  - A taxonomy of the IK methods
  - Analytical method
  - Iterative methods



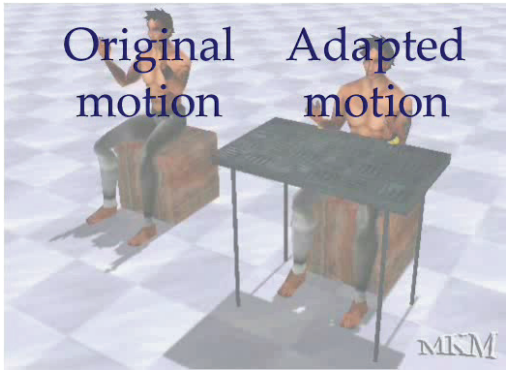
# Constraints

- Inverse kinematics is controlled by the $x_g$ parameter

  $\theta = F^{-1}(x_g,$ body skeleton$)$
- $x_g$ represents:
  - a type of high-level control
    - Position
    - Orientation, Gaze
    - Balance
  - the goal defined by the user or automatically depending on the context
    - Interaction tasks (floor, reach, grasp…)
    - Collision avoidance
  - Remark: "***task***" and "***constraint***" are used as synonyms

# Constraint definition

- Example of constraint parameters:
  - Type
    - Equality
    - Inequality
  - Position
  - Range/recruiting
  - Importance

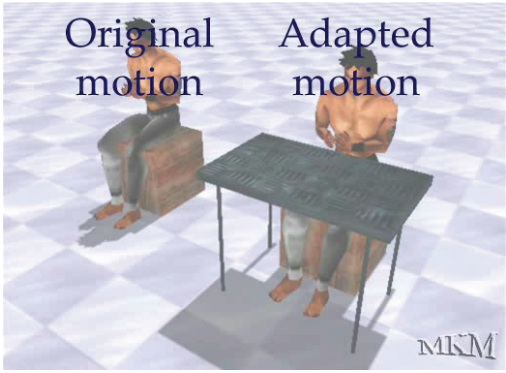



A point on the skin of the forearm placed on the edge of the table



# Constraint definition

- Example of constraint parameters:
  - Type
    - Equality
    - Inequality
  - Position
  - Range/recruiting
  - Importance




Wrists above the table



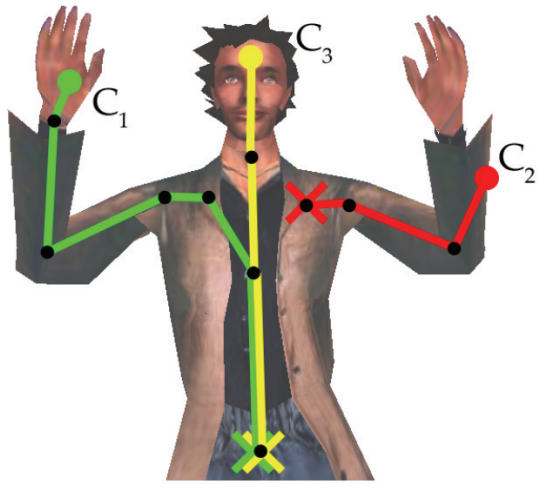
# Constraint definition

- Example of constraint parameters:
  - Type
    - Equality
    - Inequality
  - Position
  - Range/recruiting
  - Importance




# Constraint definition

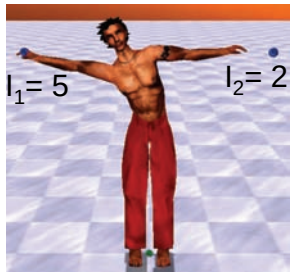
- Example of constraint parameters:
  - Type
    - Equality
    - Inequality
  - Position
  - Range/recruiting
  - Importance



# Constraint definition

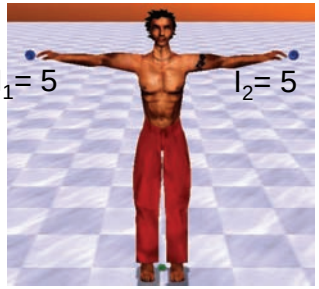
- Example of constraint param:
  - Type
    - Equality
    - Inequality
  - Position
  - Range/recruiting
  - Importance

Priority




or

Weight






# Conflicting constraints

- Frequent when interacting with the environment
- Two ways to handle conflicting constraints
  - Weights
    - Find a compromise using a weigthed sum of the constraints [Zhao93, Gleicher98, Gleicher02]
  - Priorities
    - Verify the most important constraints first.
    - Try to verify the others only if possible [Liégeois77, Hanafusa81, Yamane03, Slotine91, Baerlocher98]

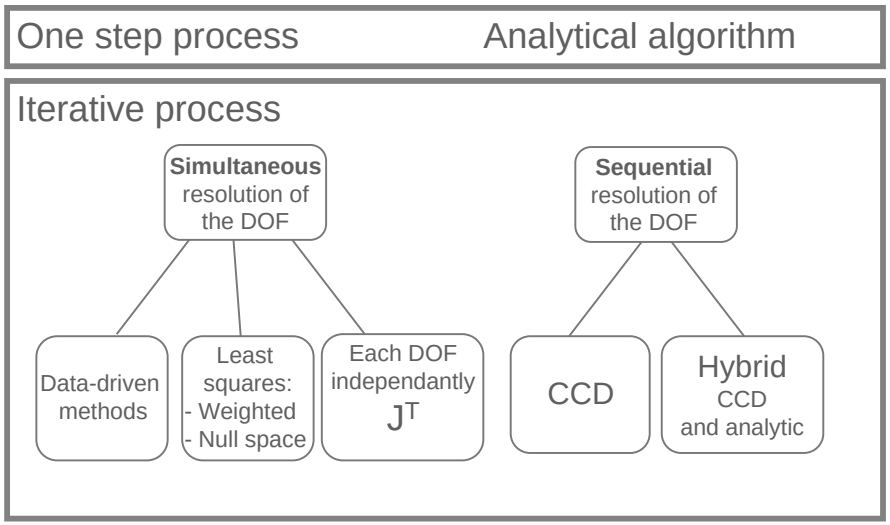


# Part A - Overview

- Motivation
- Problem
  - Definition of inverse kinematics
  - Definition of the constraints
- Overview of IK methods
  - A taxonomy of IK methods
  - Analytical method
  - Iterative methods



# Overview of IK methods



# Part A - Overview

- Motivation
- Problem
  - Definition of inverse kinematics
  - Definition of the constraints
- Overview of the IK methods
  - A taxonomy of the IK methods
  - Analytical method
  - Iterative methods



# Analytical methods

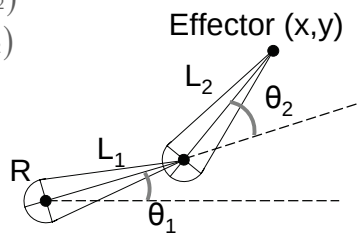
- Invert the kinematic equations
- Example for a limb
  - Direct kinematics

$$x = f_1(\theta_1, \theta_2) = l_1 \cos\theta_1 + l_2 \cos(\theta_1 + \theta_2)$$
$$y = f_2(\theta_1, \theta_2) = l_1 \sin\theta_1 + l_2 \sin(\theta_1 + \theta_2)$$

  - Inverse kinematics

$$\theta_2 = \arccos\left(\frac{x^2 + y^2 - L_1^2 - L_2^2}{2L_1L_2}\right)$$

$$\theta_1 = \arctan\left(\frac{y}{x}\right) - \arctan\left(\frac{L_2 \sin\theta_2}{L_1 + L_2 \cos\theta_2}\right)$$






# Analytical methods (2)

- Major references:
  - Robotics: [Craig03]
  - Arm/leg postural control: [Korein82, Paul88, Watt92, Chin96, Tolani00]
  - Other authors used it in combination with other methods [Lee99, Shin01]



# Summary: Analytical IK

- Advantages
  - Global solution
  - Reliable, exploited in Robotics
  - Strong (no singularity)
  - Fast
  - « Simple »
- Drawbacks
  - Only for a few DOFs
  - Not suited for redundant systems more complex than the arm/leg

# Part A - Overview

- Motivation
- Problem
  - Definition of inverse kinematics
  - Definition of the constraints
- Overview of the IK methods
  - A taxonomy of the IK methods
  - Analytical method
  - Iterative methods
    - Simultaneous resolution of the DOF
    - Sequential resolution of the DOF



# Iterative methods



Simultaneous resolution of the DOF

- General approach: considering the problem as a cost function to be minimized
  - Non-linear Optimization method [Badler86, Badler87, Zhao94]

$$\text{Cost}(\theta) = K_{main}\,[\, f(\theta) - X_d \,]^2 + K_{secondary}\, g(\theta)$$

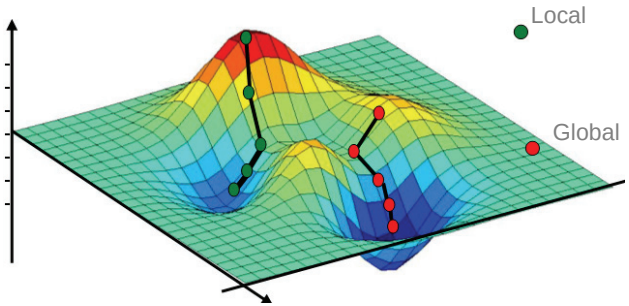
Main task: error on the position

Secondary task



# Summary: non-linear optimization

- No singularity
- Problem to choose the weights
- Difficult to enforce strict priority levels
- Can fall into a local minimum






# Iterative methods

Simultaneous resolution of the DOF

- Other focus: Linearization
  - Exploiting the Jacobian
    - Transpose
    - Pseudo-inverse : part B
  - Dealing with small displacements

# Principle of the linearization

1) First linearizes the function:

x = F(θ, body skeleton)

by storing its first derivatives

in the jacobian matrix *J*.

$$J = \begin{pmatrix} \frac{\partial Fx_1}{\partial \theta_1} & \frac{\partial Fx_1}{\partial \theta_2} & \cdots & \frac{\partial Fx_1}{\partial \theta_N} \\ \frac{\partial Fx_2}{\partial \theta_1} & \frac{\partial Fx_2}{\partial \theta_2} & \cdots & \frac{\partial Fx_2}{\partial \theta_N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial Fx_M}{\partial \theta_1} & \frac{\partial Fx_M}{\partial \theta_2} & \cdots & \frac{\partial Fx_M}{\partial \theta_N} \end{pmatrix}$$



# Principle of the linearization

2) Then, we use the inverse of the Jacobian as a local approximation of

θ = F$^{-1}$(x, body skeleton)

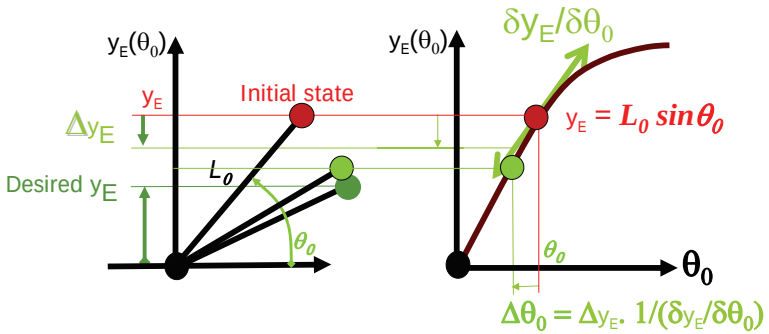
with :

$$\Delta\theta = J^{-1}(\Delta x)$$



# Linearization: the 1 D case
(one segment only)

- Linearizing the equation(s) $y_E = f(\theta)$
  - The resolution converges towards *one* possible solution through successive state variations
  - Local solution: great importance of the initial state



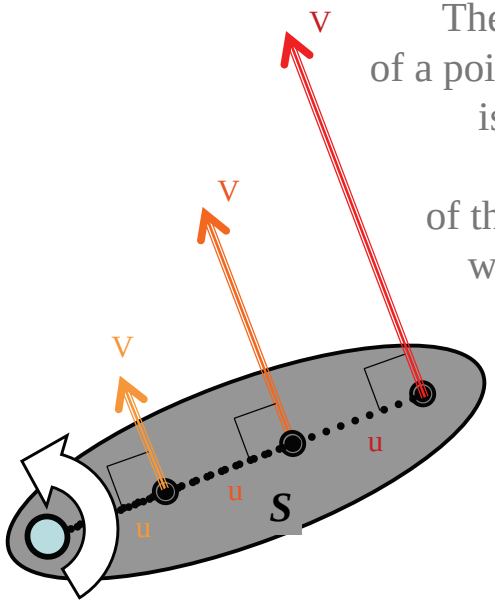



# How to build the jacobian?

- Consider the influence of each joint independently of the other joints

  **The joints ignore each other ⇔ it's a first order approximation**

- When one joint changes, all the children segments are viewed as ONE UNIQUE RIGID SOLID
- It is easy to evaluate the instantaneous velocity of a point of a rigid solid:
  - With the cross product of the rotation velocity vector by the position vector locating the point with respect to the joint (see next slide).
  - Use a unit rotation velocity for building the Jacobian

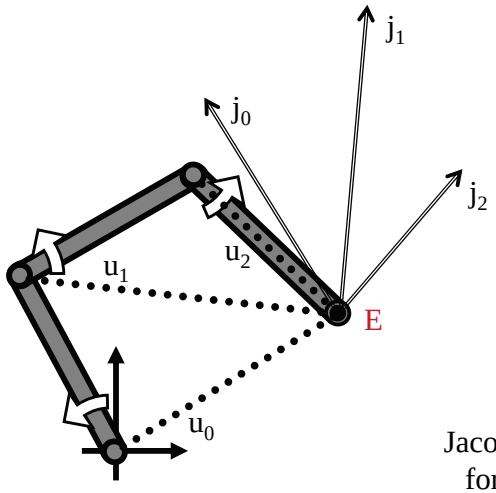



The velocity vector V of a point from the solid ***S*** is obtained with the cross product of the rotation velocity with the *lever arm* u





## Illustration for a 3 segment arm, for point E




| $j_0$ | $j_1$ | $j_2$ |
|---|---|---|
| $Xj_0$ | $Xj_1$ | $Xj_2$ |
| $Yj_0$ | $Yj_1$ | $Yj_2$ |

Jacobian J of the effector E for the chain current state

See [Rotenberg06] for other jacobian computation



## Jacobian transpose

- The Jacobian matrix $J$, of dimension $m$ x $n$, gathers all the partial derivatives of the $m$ constraints $x_i$ with respect to the $n$ joint variables $\theta_j$

$$J = [\delta x_i / \delta \theta_j]$$

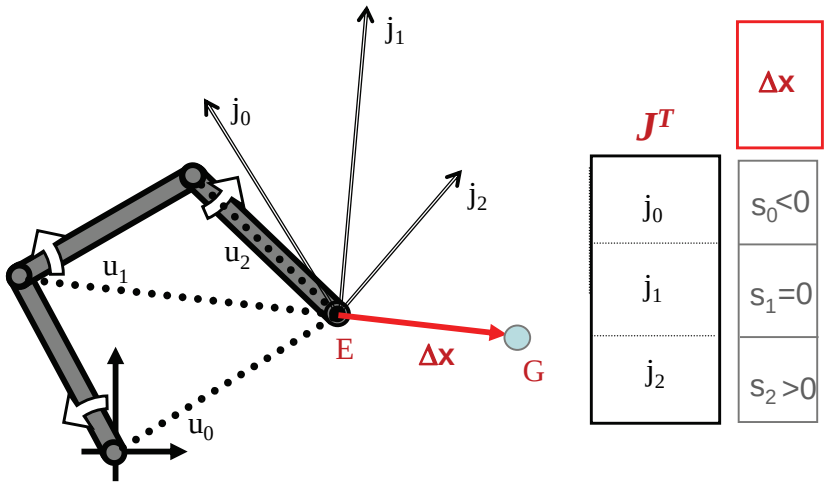
- The Jacobian transpose method [Welman93]:
  - exploits the absolute influence of each joint variable $\theta_j$ for contributing to the constraint error vector $\Delta$x
  - the influence of a joint $j$ is given by the scalar product of $\Delta$x with the column $j$ of J



## Jacobian transpose (2)

- The method is fast to compute
  $\Delta\theta = \beta J^T \Delta x$ , with $\beta$ small
- But not that fast to converge
  - The relative contribution of joint variables is not taken into account; joint dimensions are "blind" to each other…
  - $\beta$ must be small due to the non-linearity of the direct kinematics model.
  - No possibility to define strict priorities among constrained dimensions
  - Full extension singularity

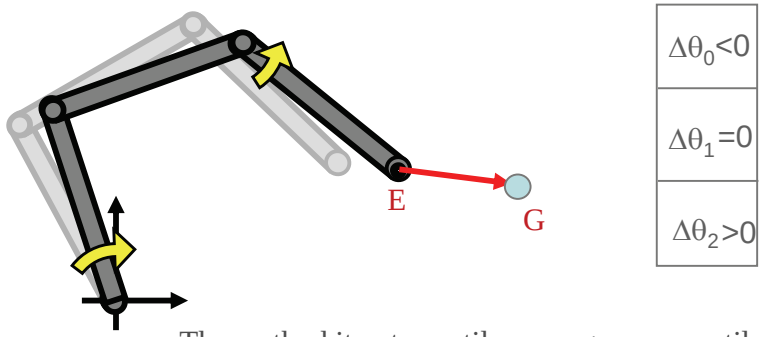
## Ex: the effector E is attracted toward the goal G (1)




Translation Jacobian *J* of the effector E for the chain current state



## Ex: the effector E is attracted toward the goal G (2)




The method iterates until convergence or until the error norm does not decrease any more



## Part A - Overview

- Motivation
- Problem
  - Definition of inverse kinematics
  - Definition of the constraints
- Overview of the IK methods
  - A taxonomy of the IK methods
  - Analytical method
  - Iterative methods
    - Simultaneous resolution of the DOF
    - Sequential resolution of the DOF



## Sequential resolution of the DOF

- Exactly the same concept of small displacement and linearization
- But avoid the Jacobian, dealing only one DOF at a time
  - Computation of one iteration is very fast
  - Number of iterations can be important
- Described in Part C

## Comparison of analytic and iterative IK

- Analytic is best suited for simple case like isolated arm, leg, etc.
- Iterative is more general but requires multiple steps to converge towards the solution
  - Due to the non-linearity of the problem
  - If big steps are used, it becomes unstable

# Inverse kinematics and kinetics for virtual humanoids

Ronan Boulic

VRLab – EPFL
http://vrlab.epfl.ch

Richard Kulpa

Bunraku – INRIA / IRISA
www.irisa.fr/bunraku

M2S – Univ. Rennes 2
www.uhb.fr/labos/m2s

## Plan

- Part A - Overview
  - Motivation
  - Problem
  - Overview of the IK methods
- ⇨ Part B - Iterative methods (pseudo-inverse based)
  - Concept of prioritized control
  - From one to P priority levels
- Part C - Iterative methods (CCD-based)
  - CCD
  - Hybrid CCD and analytical
- Part D - Conclusion



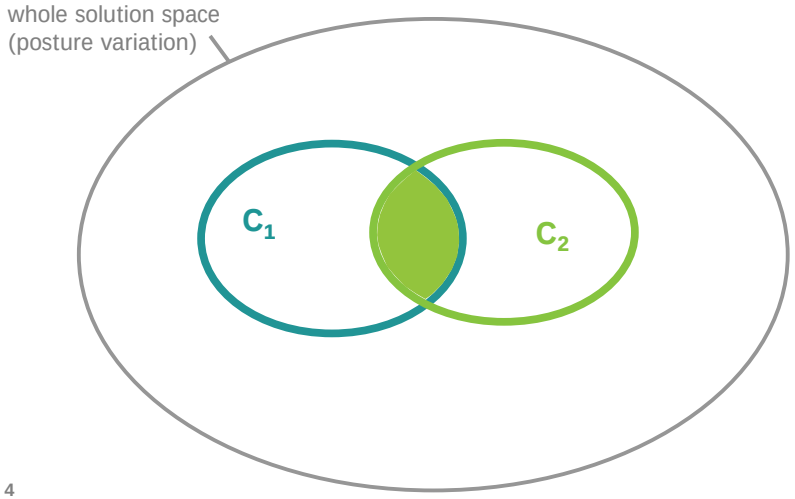
## Part B - Iterative methods (pseudo-inv.)

- ⇨ Concept of prioritized control
- From one to P priority levels
  - General form
  - Incremental constraint achievement
  - Example1: the simplest redundant case
  - Example 2: two prioritized constraints
  - Generalization to multiple priority levels
  - Fostering synergies
  - Handling singularities with DLS
  - Application: collision avoidance
  - Conclusion



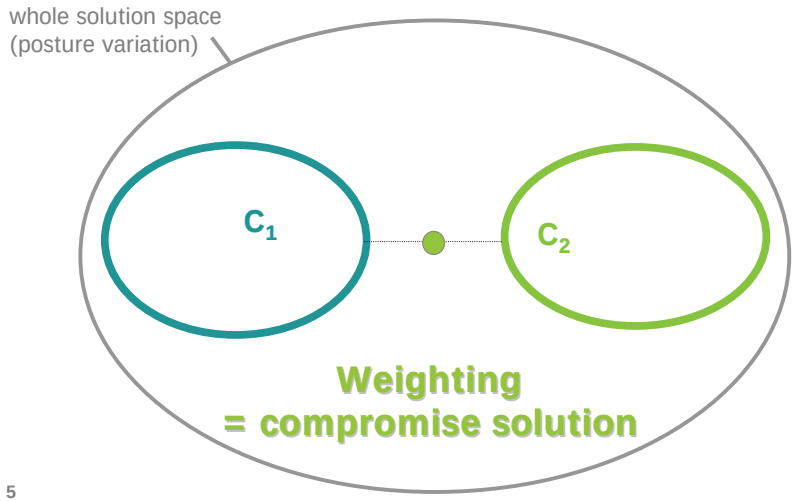
## Conceptual analogy

Visualizing the constraints solution sub-spaces: $\mathbf{C_i}$



# Conceptual analogy

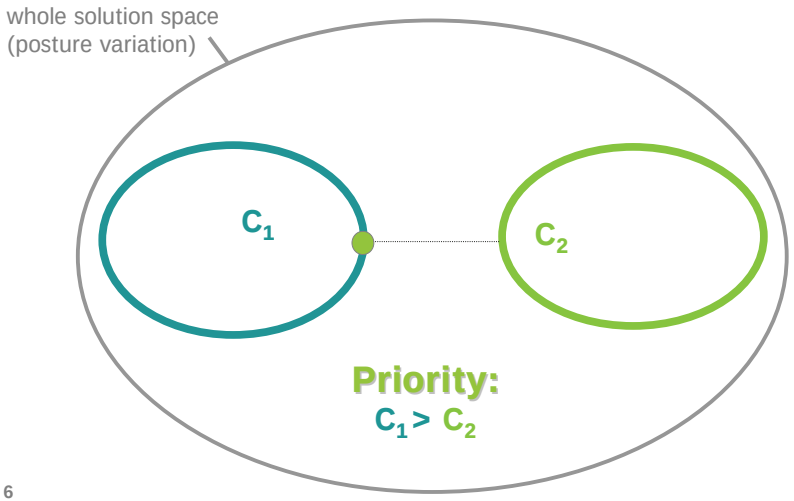
Visualizing the constraints solution sub-spaces: $C_i$




5

# Conceptual analogy

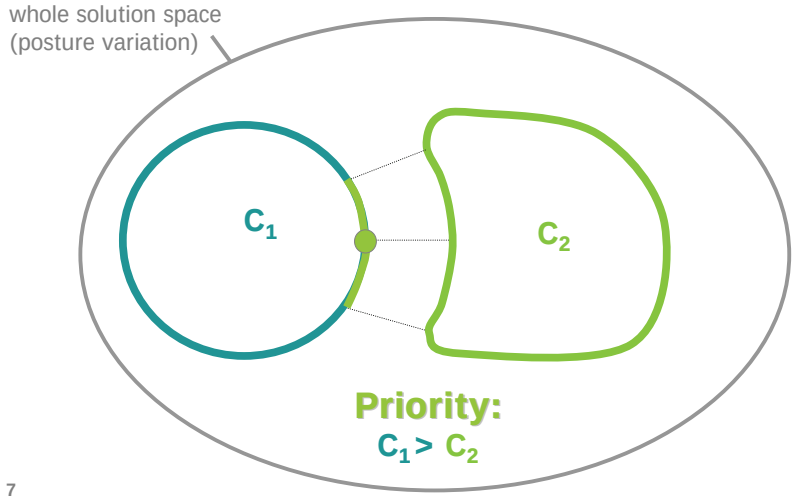
Visualizing the constraints solution sub-spaces: $C_i$




6

# Conceptual analogy

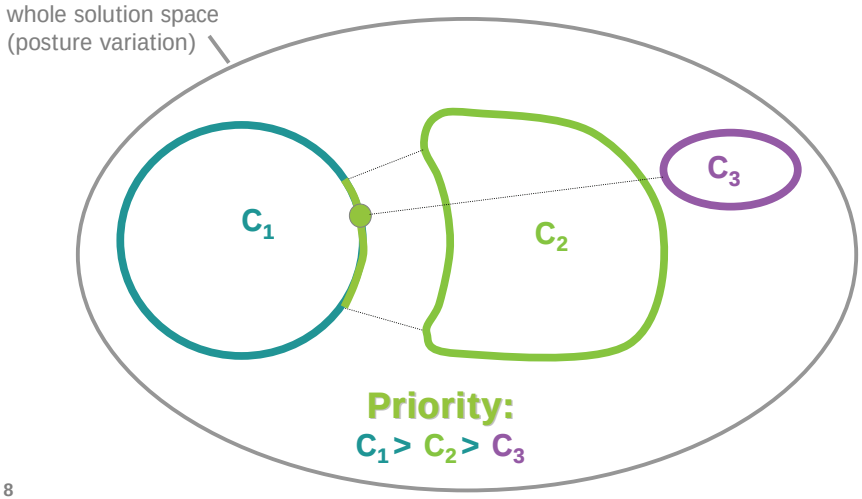
Visualizing the constraints solution sub-spaces: $C_i$




7

# Conceptual analogy

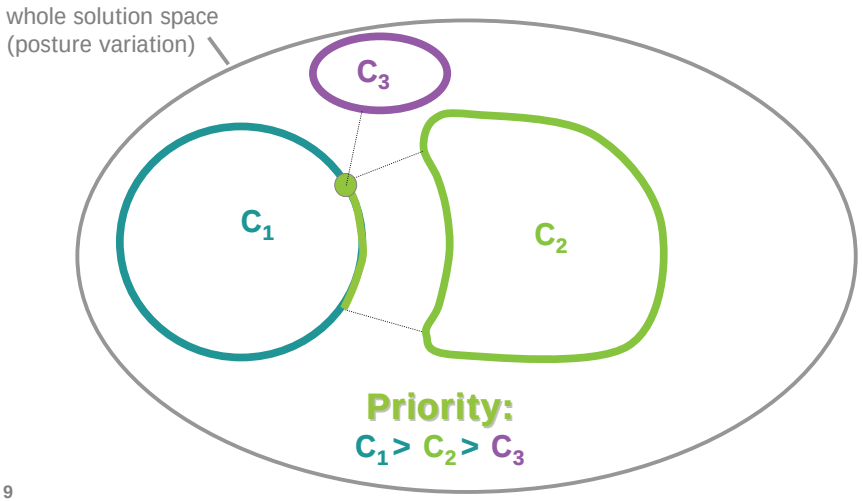
Visualizing the constraints solution sub-spaces: $C_i$




8

# Conceptual analogy

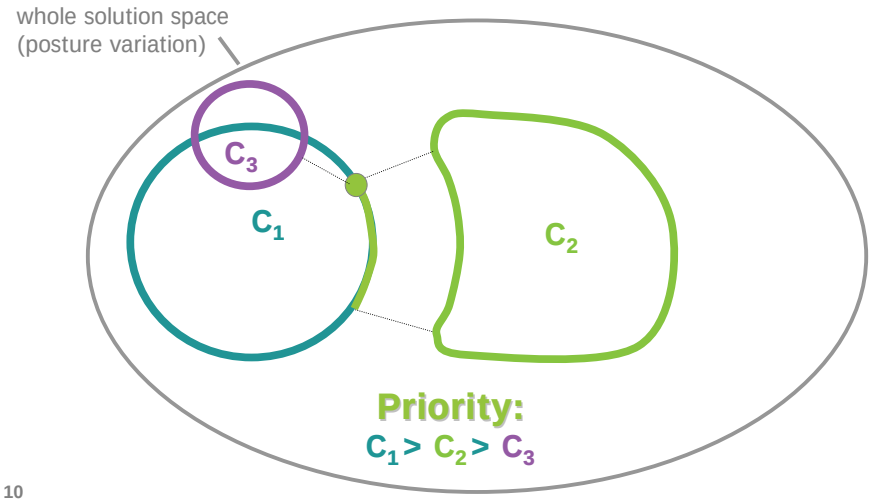
Visualizing the constraints solution sub-spaces: $\mathbf{C_i}$




9

# Conceptual analogy

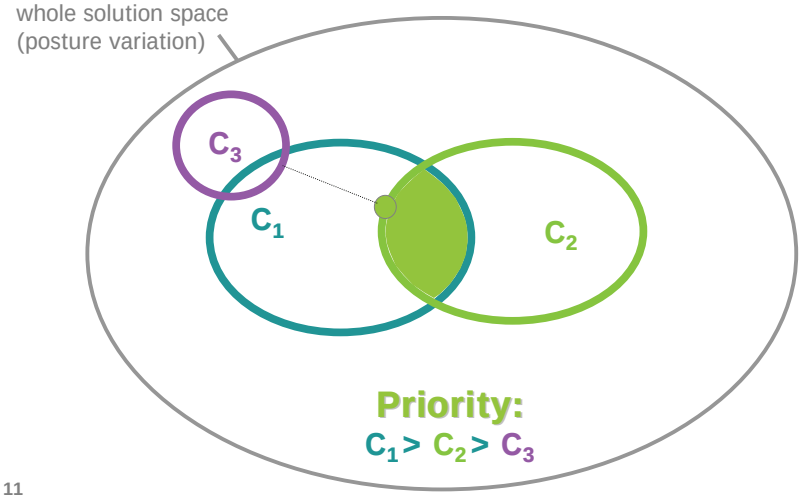
Visualizing the constraints solution sub-spaces: $\mathbf{C_i}$




10

# Conceptual analogy

Visualizing the constraints solution sub-spaces: $\mathbf{C_i}$




11

# Part B - Iterative methods (pseudo-inv.)

- Concept of prioritized control
- From one to P priority levels
  - General form
  - Incremental constraint achievement
  - Example1: the simplest redundant case
  - Example 2: two prioritized constraints
  - Generalization to multiple priority levels
  - Fostering synergies
  - Handling singularities with DLS
  - Application: collision avoidance
  - Conclusion

12

# General form

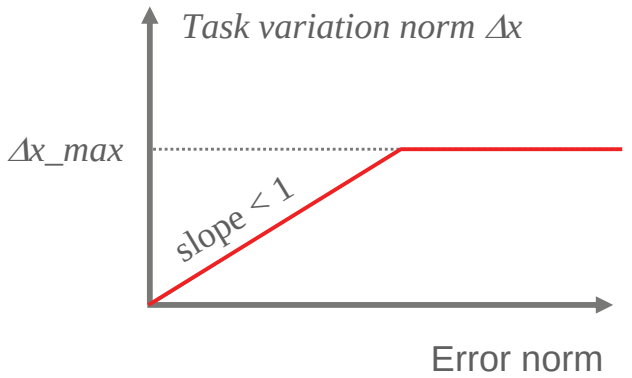
- The pseudo-inverse $J^+$ , of dimension $n$ x $m$, provides the minimum norm solution $\Delta\theta$ for a desired task variation $\Delta x$ [Klein83]

$$\Delta\theta = J^+\Delta x$$

- The validity of the solution is also limited to the neighborhood of the current state, hence the norm of $\Delta x$ is restricted to an increment toward the desired goal state.



# Incremental task achievement




- The slope ensures stability (e.g. 0.7)



# General Form of the Solution

- The solution is given by: $\Delta\theta = J^+\Delta x + P_{N(J)}\Delta\alpha$
- where $P_N$ is the $n$ x $n$ projection operator on the Null space of J :

$$P_{N(J)} = I - J^+J$$

- Any vector belonging to $N(J)$ is mapped by $J$ on the null vector
- $\Delta\alpha$ : of dimension $n$, *can be any* vector belonging to the joint variation space. It is often proportional to the gradient of a cost function expressed in the joint space g(θ).
  - remark: this term is often noted z in the Robotics litterature, e.g. [Liegeois77]

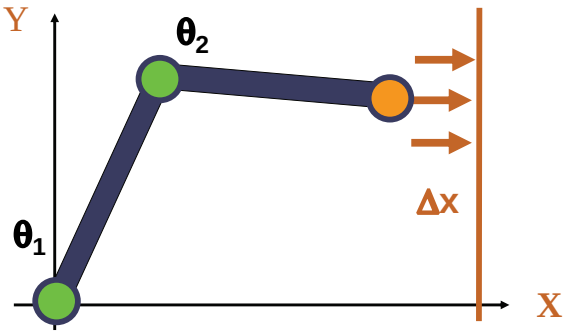


# Part B - Iterative methods (pseudo-inv.)

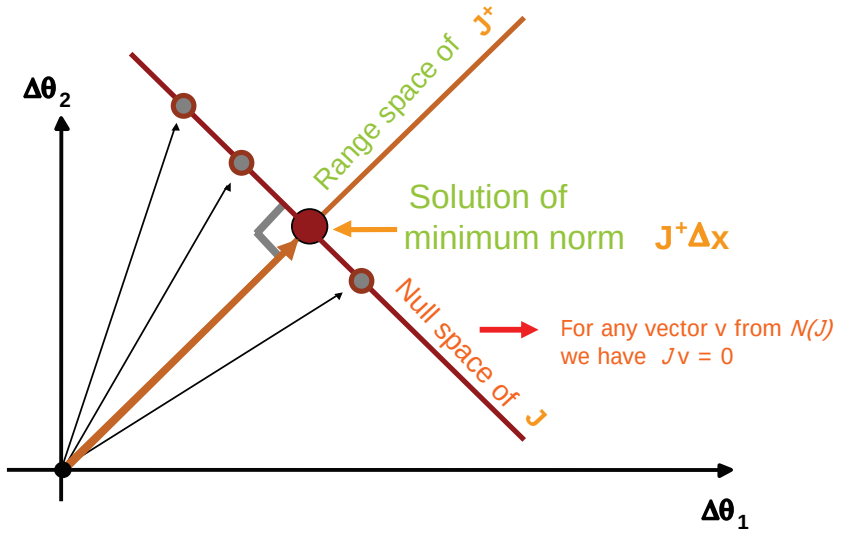
- Concept of prioritized control
- From one to P priority levels
  - General form
  - Incremental constraint achievement
  - Example1: the simplest redundant case
  - Example 2: two prioritized constraints
  - Generalization to multiple priority levels
  - Fostering synergies
  - Handling singularities with DLS
  - Application: collision avoidance
  - Conclusion

# The simplest redundant case

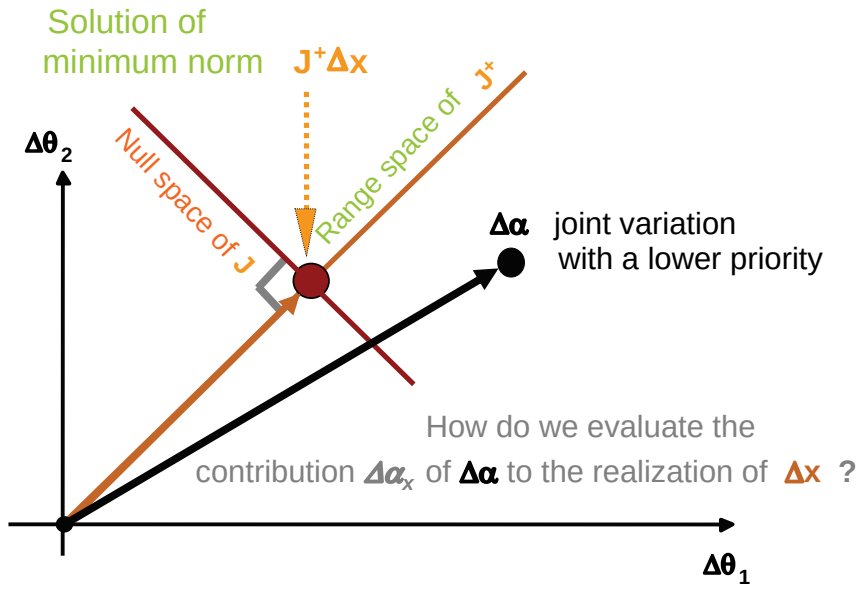
- A two DOF chain has to constrain the **X** dimension of the orange end effector




17




18

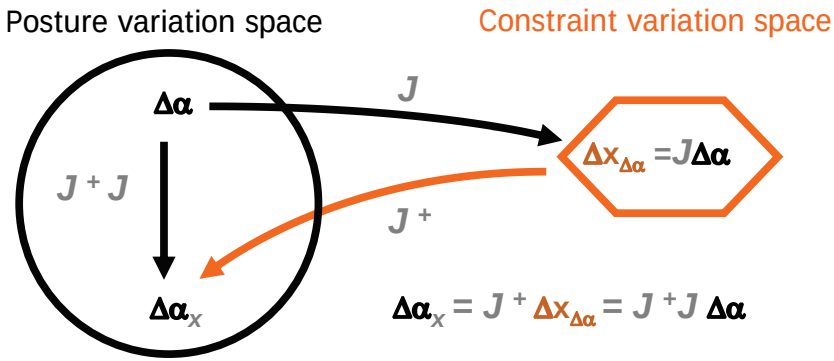



19

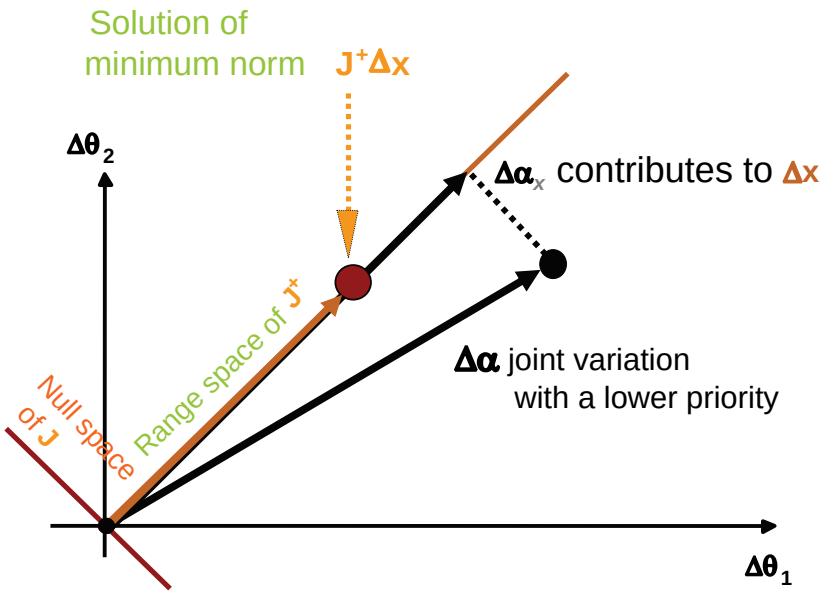
– How do we evaluate the contribution $\Delta\alpha_x$ of $\Delta\alpha$ to the realization of $\Delta x$ ?

First the influence of $\Delta\alpha$ in the constraint space $\Delta x_{\Delta\alpha}$ is given by multiplying $\Delta\alpha$ with the Jacobian $J$ :

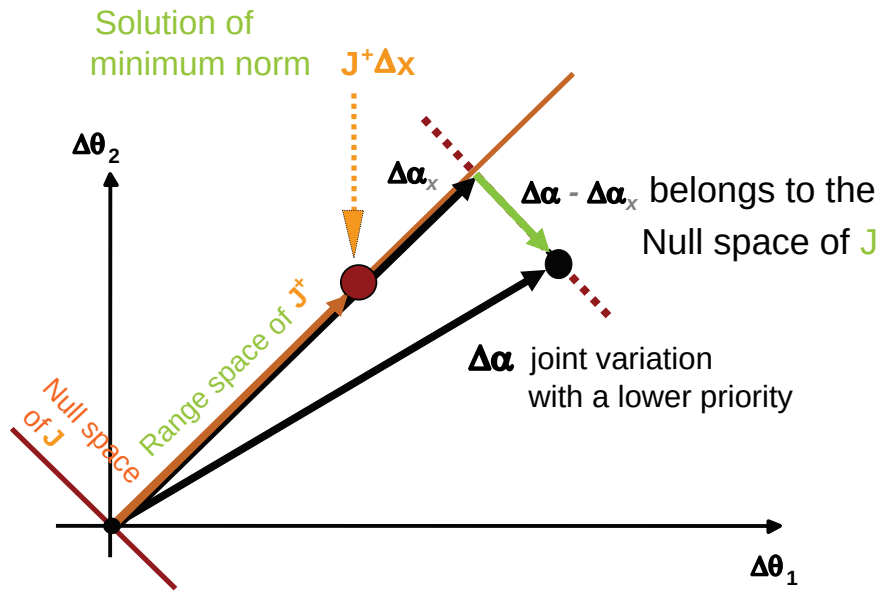
$\Delta x_{\Delta\alpha} = J\,\Delta\alpha$




20




21




22

$\Delta\alpha - \Delta\alpha_x$ belongs to the Null Space of J

it is mapped to the null vector by J:

$$J(\Delta\alpha - \Delta\alpha_x) = J(\Delta\alpha - J^+ J\,\Delta\alpha)$$

$$= J\Delta\alpha - J\,J^+ J\,\Delta\alpha$$

$$= \Delta x_{\Delta\alpha} - J\,J^+ \Delta x_{\Delta\alpha}$$

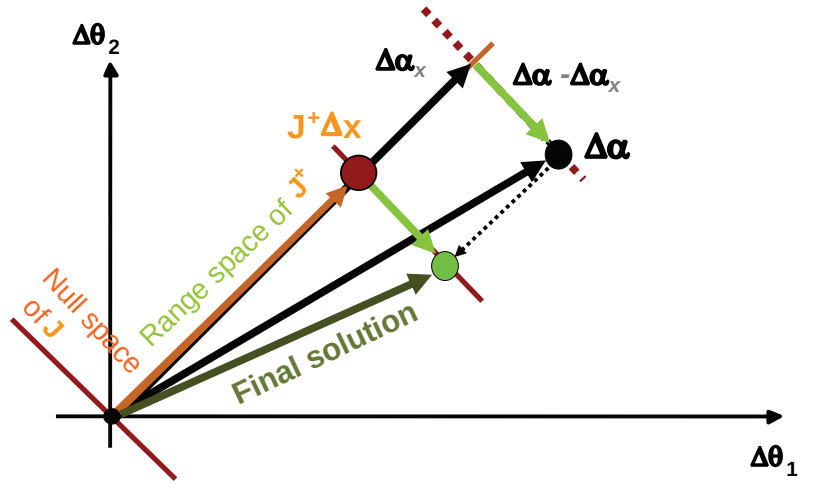
$$= \Delta x_{\Delta\alpha} - J\,\Delta\alpha_x$$

$$= \Delta x_{\Delta\alpha} - \Delta x_{\Delta\alpha}$$

$$= 0$$

23

The *orthogonal projection on the Null space N(J)* $(\Delta\alpha - \Delta\alpha_x)$ can be safely added to the minimum norm solution




24

– Any vector $\Delta\alpha$ projected on the Null space N(J) does not perturb the constraint achievement:
  - it is mapped to the null vector by J

$$J\,(\Delta\alpha - \Delta\alpha_x) = 0$$

– The projection operator on the Null Space is noted $P_{N(J)}$ :

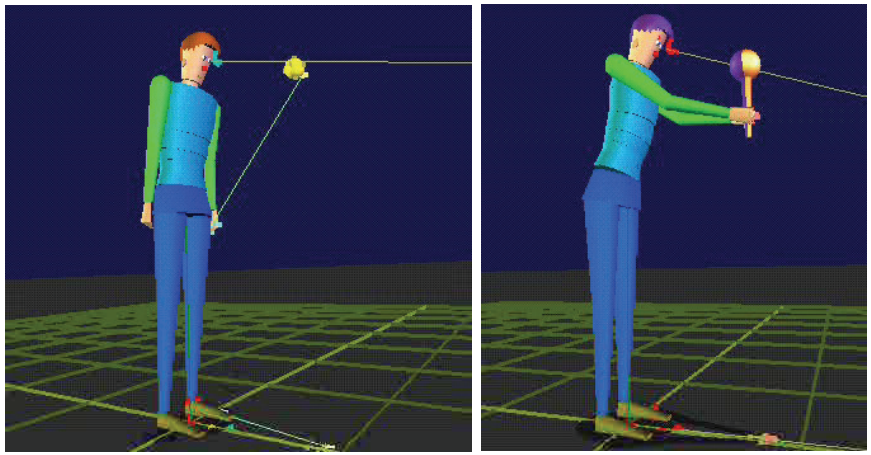
$$(\Delta\alpha - \Delta\alpha_x) = \Delta\alpha_{N(J)} = P_{N(J)}\,\Delta\alpha$$

$$\Delta\alpha - \Delta\alpha_x = \Delta\alpha - J^{+}J\,\Delta\alpha = (I - J^{+}J)\,\Delta\alpha$$

$$P_{N(J)} = (I - J^{+}J)$$

25

## Example in interactive context [Baerlocher04]



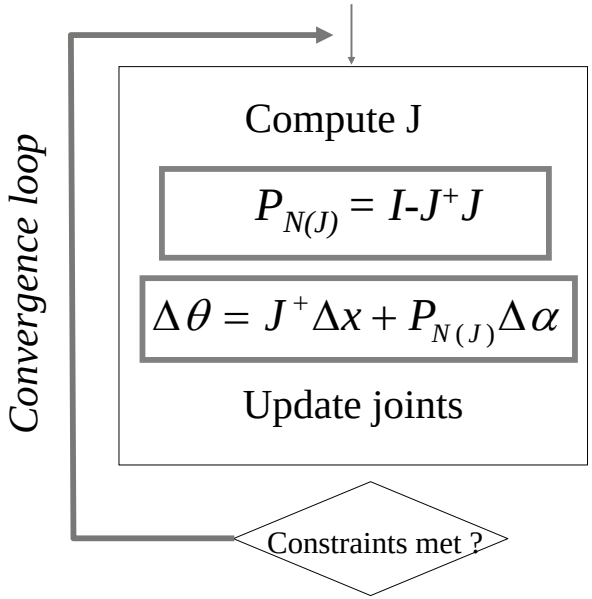
26

## Basic Prioritized IK architecture

**General case:**

$\Delta x$ : vector in the task variation space

$\Delta\theta$, $\Delta\alpha$ : vectors in the posture variation space




27

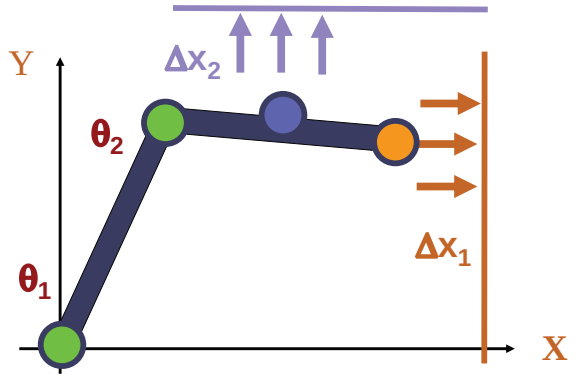
## Part B - Iterative methods (pseudo-inv.)

- Concept of prioritized control
- From one to P priority levels
  - General form
  - Incremental constraint achievement
  - Example1: the simplest redundant case
  - ⇨ Example 2: two prioritized constraints
  - Generalization to multiple priority levels
  - Fostering synergies
  - Handling singularities with DLS
  - Application: collision avoidance
  - Conclusion

28

# Towards Hierarchical Constraints: two 1D constraints (1)

- A two dofs chain has to constrain effector1 on a line and effector2 on another line.






# Towards Hierarchical Constraints: two 1D constraints (2)

- Context: the constraints are conflicting; they cannot be achieved simultaneously
- Objective: give the highest priority to the achievement of $\Delta x_1$ and achieve as much as possible $\Delta x_2$ without perturbating $\Delta x_1$
- Specific issue: take into account the fact that achieving the high priority task $\Delta x_1$ may interfere with the low priority task $\Delta x_2$



# Towards Hierarchical Constraints: two 1D constraints (3)

- Let $\Delta\theta_i$ denote the solution vector in the posture variation space for priority level ***i***
- The high priority solution for the task $\Delta x_1$ is :

$$\Delta\theta_1 = J_1^+ \Delta x_1$$

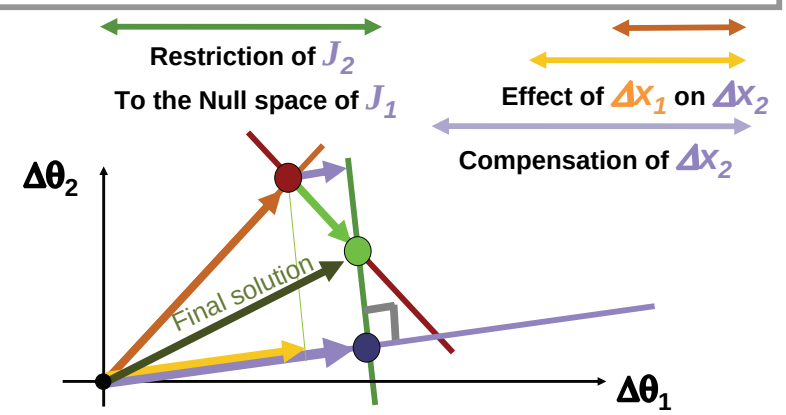
- The low priority term $\Delta\alpha$ has to be projected on, or *to belong to*, the Null space of $J_1$



# Two conflicting Constraints [Hanafusa81]

Construction of the contribution of second priority level :

$$\Delta\alpha = [J_2(I - J_1{}^+ J_1)]^+ (\Delta x_2 - J_2(J_1{}^+ \Delta x_1))$$



## Towards Hierarchical Kinematic Constraints

- Let us rewrite the total solution vector $\Delta\theta$ for two priority levels :

$\Delta\theta$ = $\Delta\theta_1$ + contribution of priority level 2

$$\Delta\theta = \Delta\theta_1 + [J_2(I - J_1^+ J_1)]^+ (\Delta x_2 - J_2(J_1^+ \Delta x_1))$$

$$\Delta\theta = \Delta\theta_1 + [J_2 P_{N(J_1)}]^+ (\Delta x_2 - J_2 \Delta\theta_1)$$



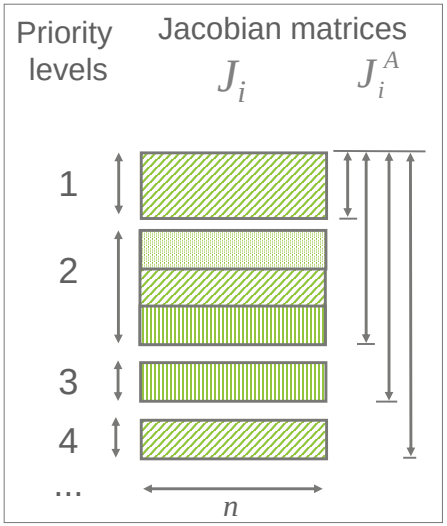
## Part B - Iterative methods (pseudo-inv.)

- Concept of prioritized control
- From one to P priority levels
  - General form
  - Incremental constraint achievement
  - Example1: the simplest redundant case
  - Example 2: two prioritized constraints
  - ⇨ Generalization to multiple priority levels
  - Fostering synergies
  - Handling singularities with DLS
  - Application: collision avoidance
  - Conclusion



## Extension to *P* priority levels [Slotine91]

- Each priority level $i$ is characterized by one jacobian $J_i$ :
  - It can aggregate multiple constraint Jacobians by piling them into a single matrix, resulting in a variable constrained dimension.
  - All have the same number of columns $n$ , dimension of the joint space.






## Extension to *P* priority levels (2)

- The general solution vector $\Delta\theta$ accumulates :
  - the contribution vectors $\Delta\theta_i$ of each priority level $i$,
    - for $i$ = 1 to $P$
  - AND, at the lowest priority level, the projection of $\Delta\alpha$ on the Null space of all constraints $N(J_p^A)$

$$\Delta\theta = \sum_{i=1}^{p} \Delta\theta_i + P_{N(J_p^A)} \Delta\alpha$$

## Extension to *P* priority levels (3)

- Initializations: $\Delta\theta_0 = 0$ $P_{N(J_0^A)} = I_n$
- Each priority level exploit the Augmented jacobian noted $J_i^A$ that piles all Jacobians from levels 1 to level ***i***
- The contribution of the level ***i*** is given by:

$$\Delta\theta_i = (J_i P_{N(J_{i-1}^A)})^{+\lambda_i} (\Delta x_i - J_i \Delta\theta_{i-1})$$



## Extension to *P* priority levels (4)

- Computation of the projection operators $P_{N_i}$ [Slotine91]

$$P_{N(J_i^A)} = I_n - J_i^{A+} J_i^A$$

- Incremental computation of the projection operators $P_{N_i}$ [Baerlocher98]

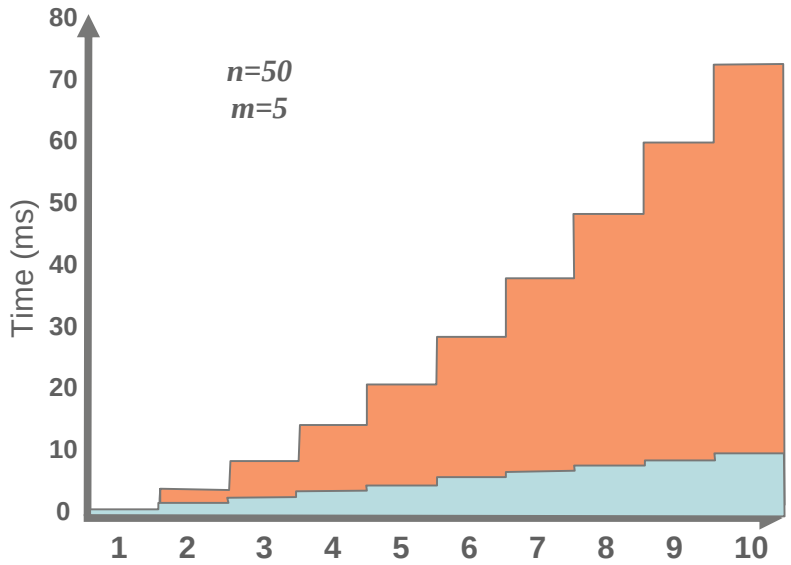
$$P_{N(J_i^A)} = P_{N(J_{i-1}^A)} - (J_i P_{N(J_{i-1}^A)})^{+} (J_i P_{N(J_{i-1}^A)})$$

Cost in *O(p)* compared to previous approach in $O(p^2)$

*speedup factor=(p+1)/2*

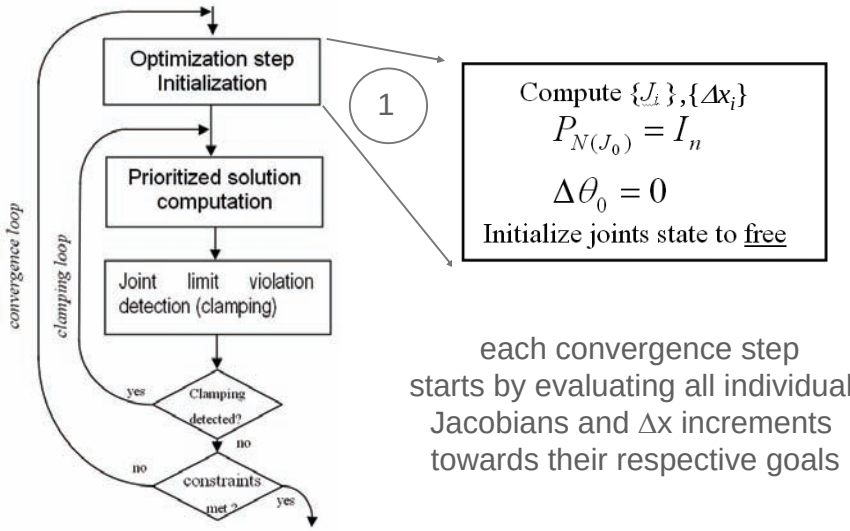


## *Computing cost vs Number of priority levels (p)*



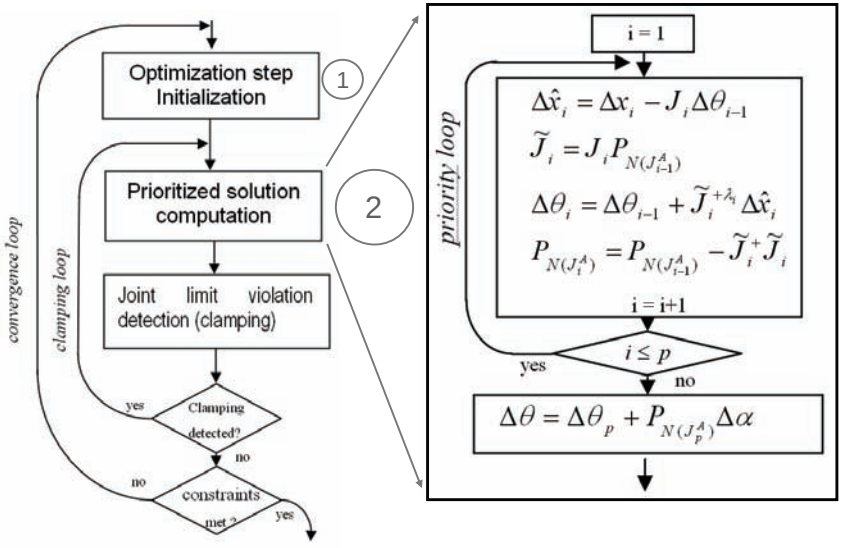



## Prioritized IK architecture(1)



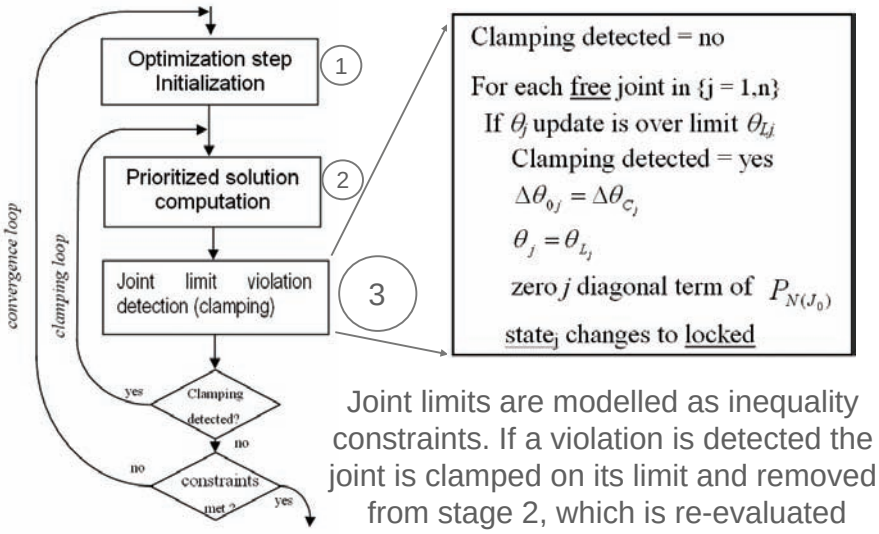

each convergence step starts by evaluating all individual Jacobians and Δx increments towards their respective goals

## Prioritized IK architecture(2)





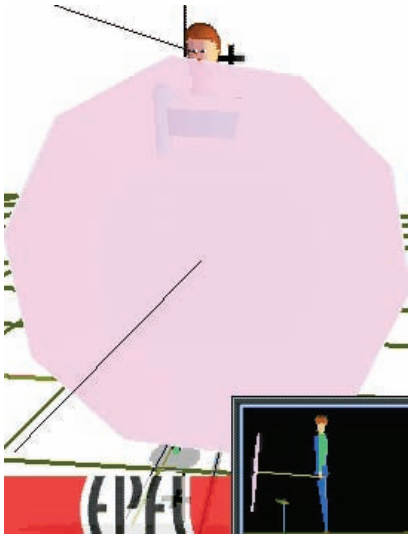

## Prioritized IK architecture(3)



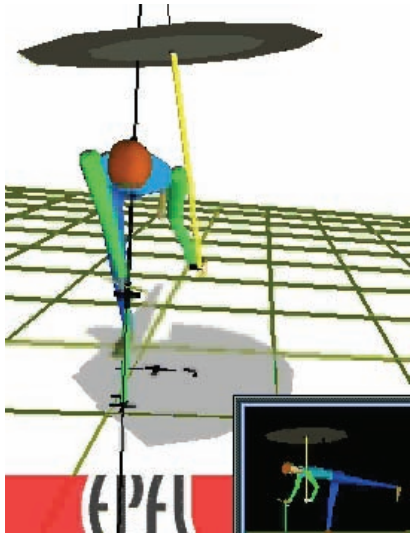

Joint limits are modelled as inequality constraints. If a violation is detected the joint is clamped on its limit and removed from stage 2, which is re-evaluated



## From the start



## with $\Delta\alpha$ term

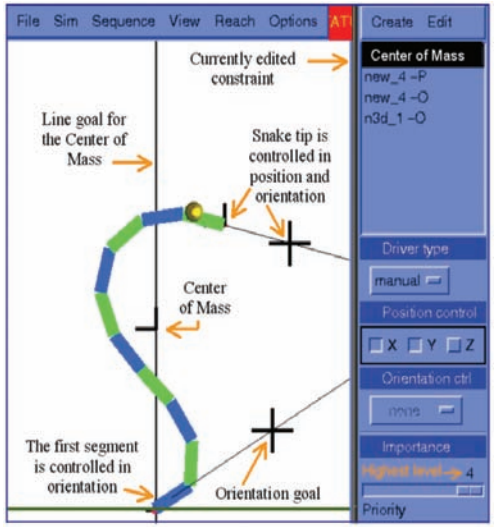


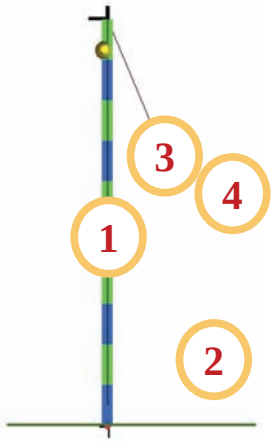


## Convergence properties of the prioritized IK

- By construction, highest priority constraints are met first
- This property guides the local optimization through more realistic intermediate postures (compared to a weighting approach).
- Less likely to be stuck in local minima

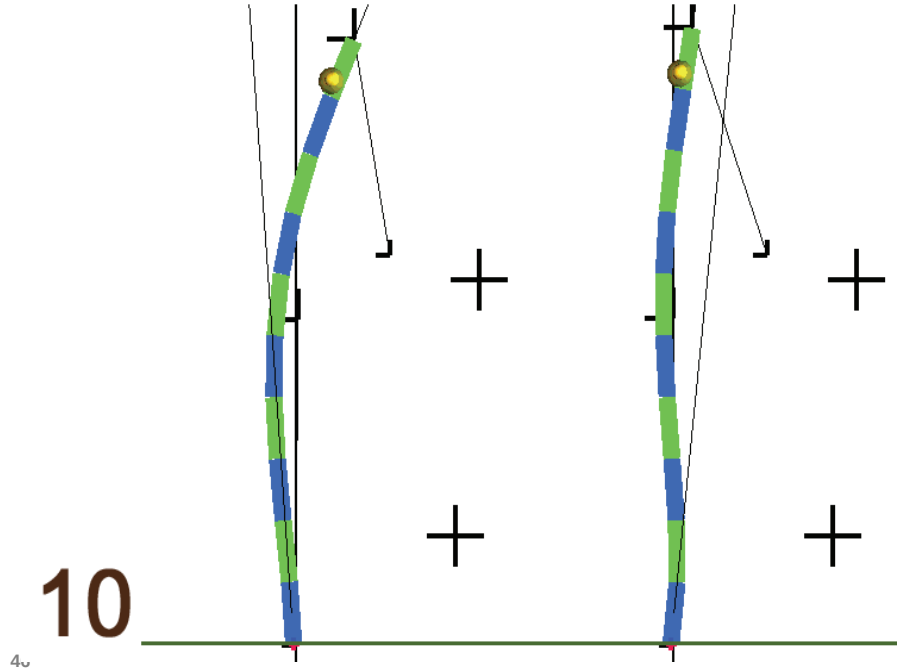
## Comparison of convergence without/with priorities

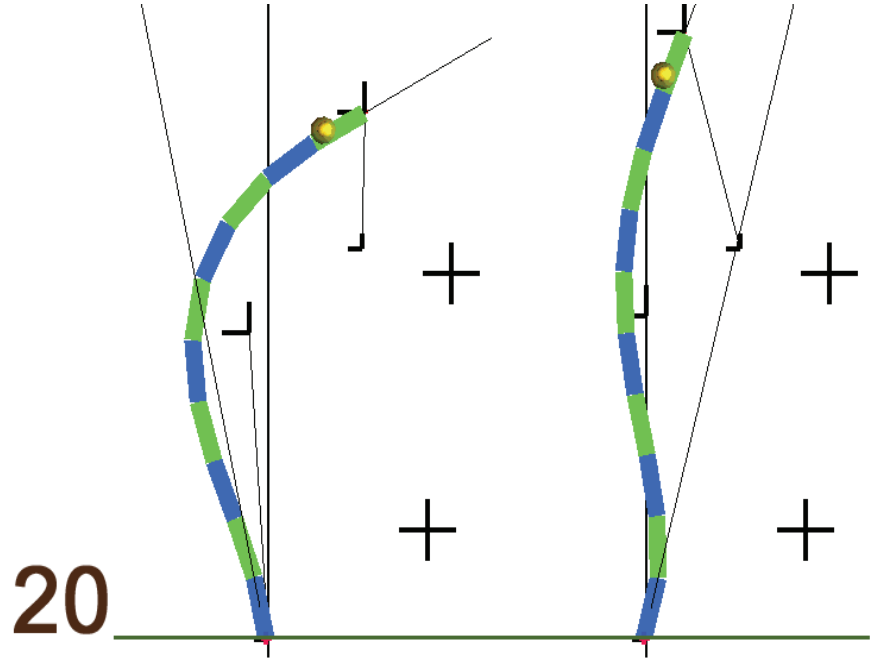


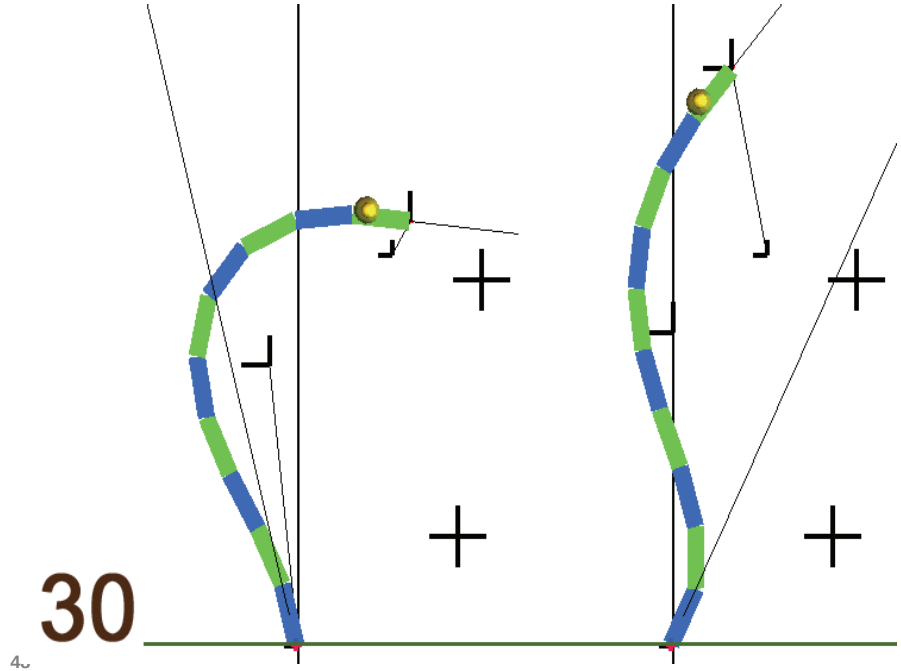



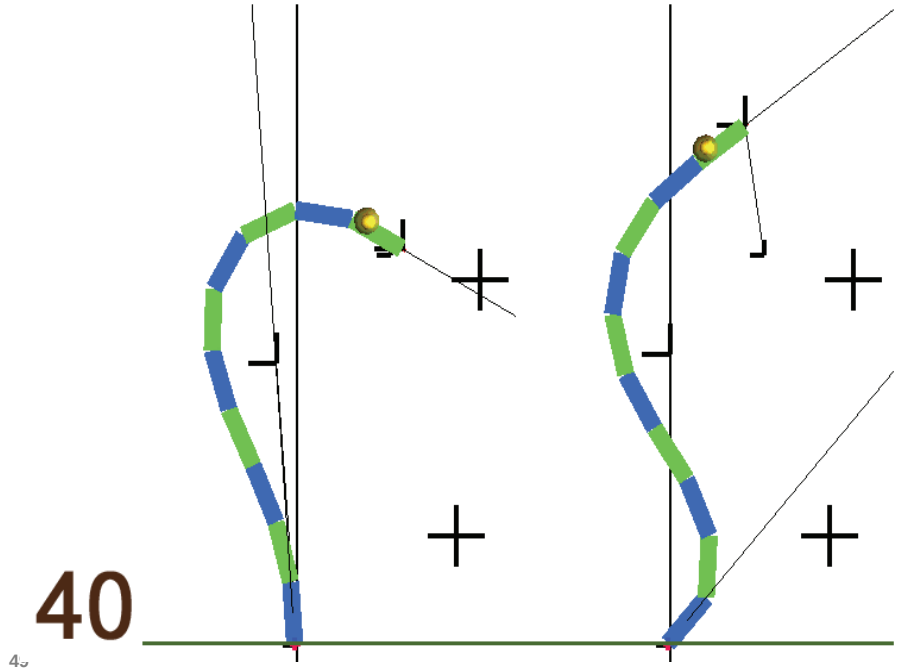

Initial posture

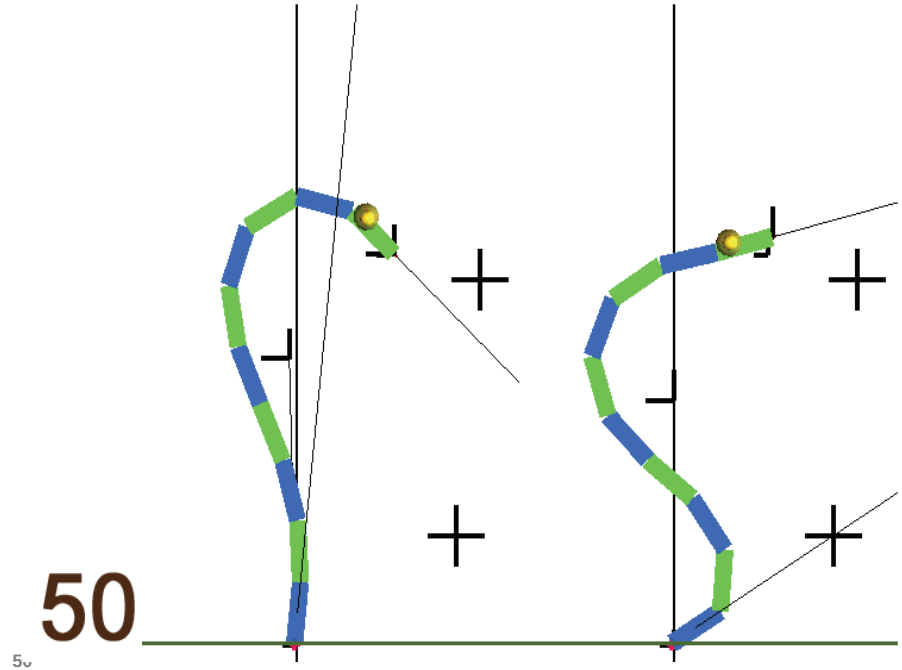
45

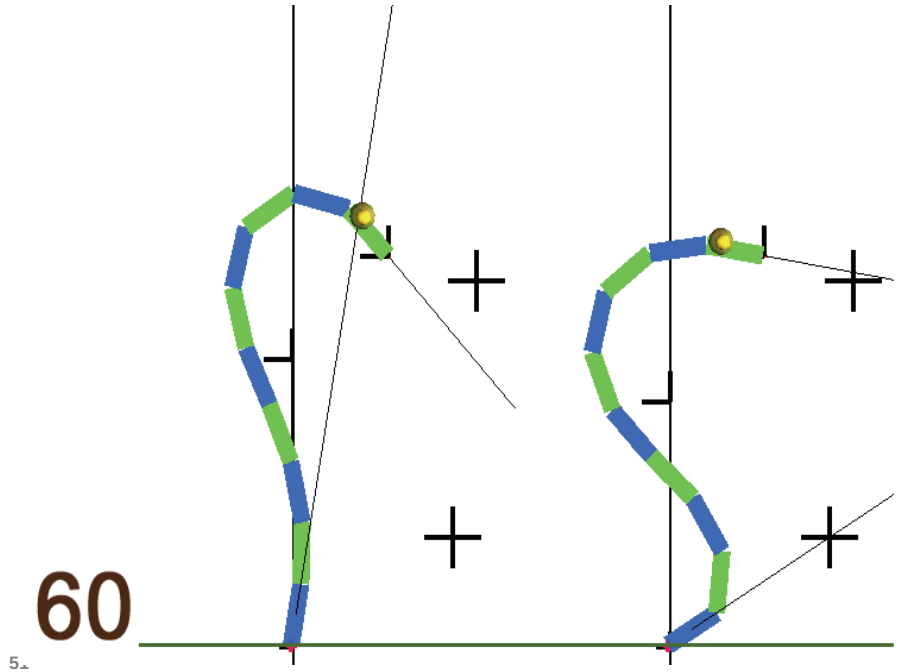


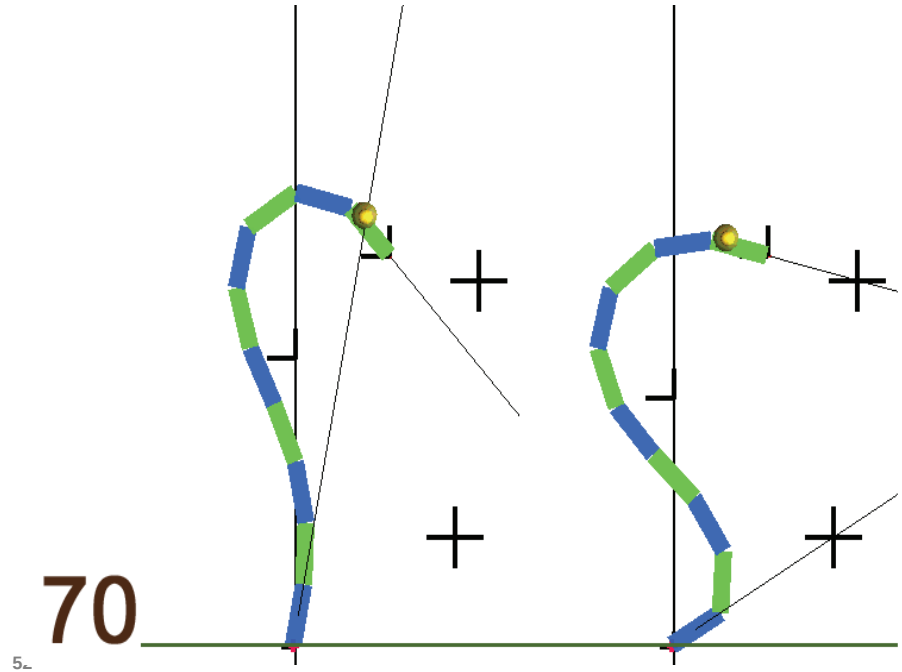



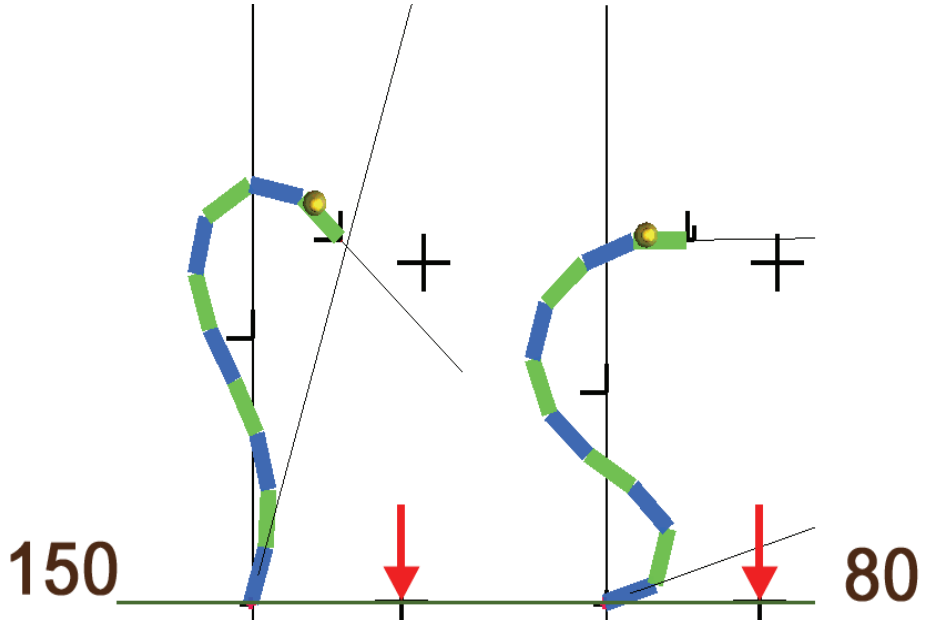





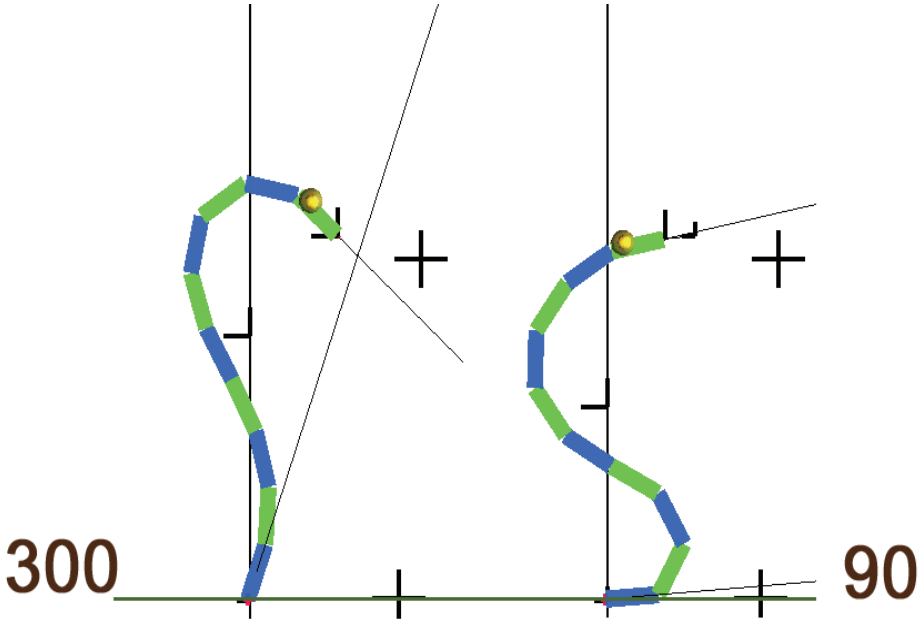



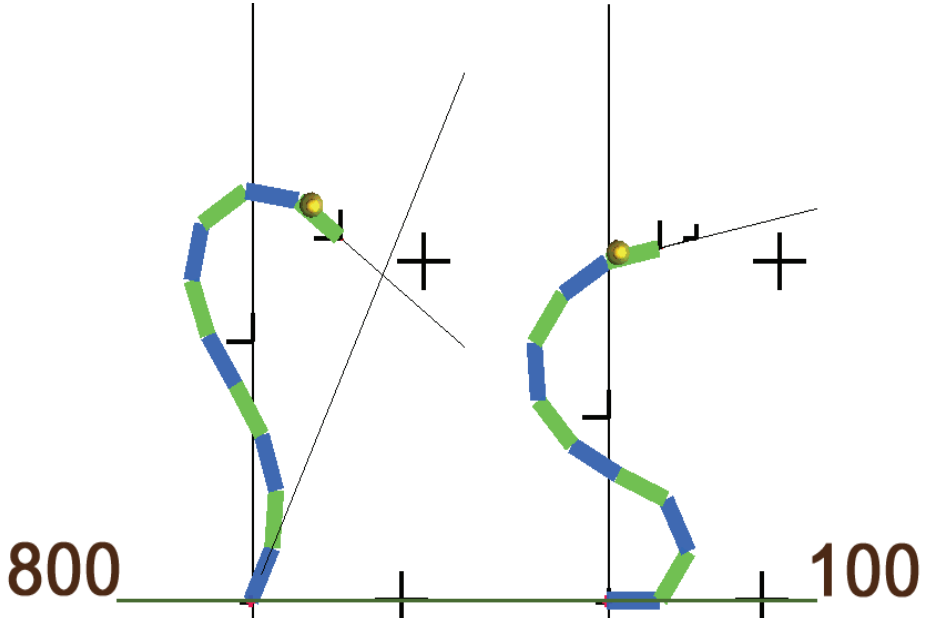






53




54

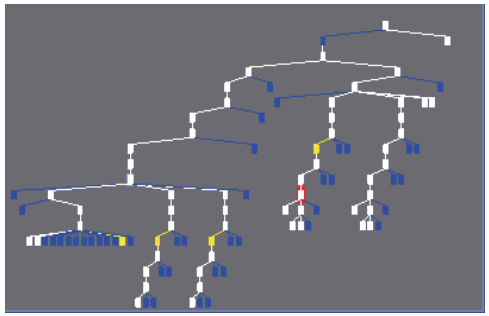


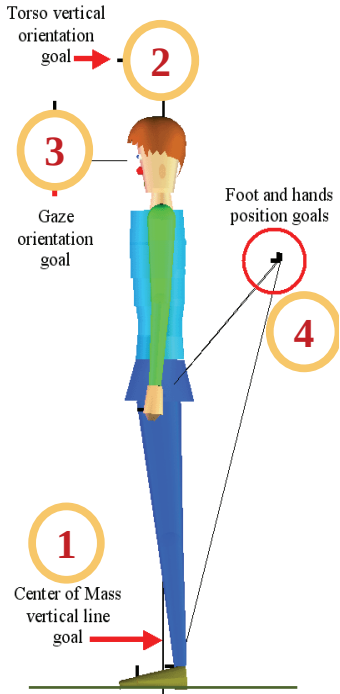

55

## Achieving a 3D dance posture

Initial posture

42 dof
(white/yellow squares)



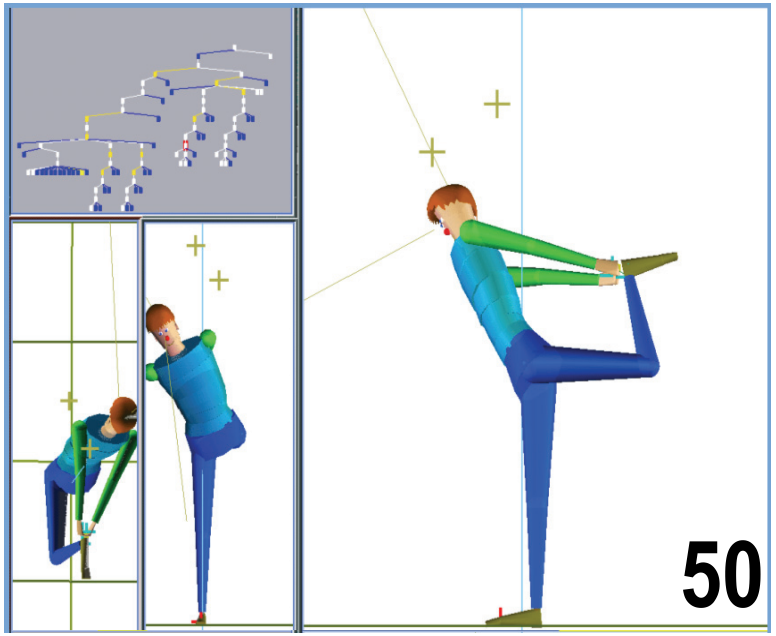



56

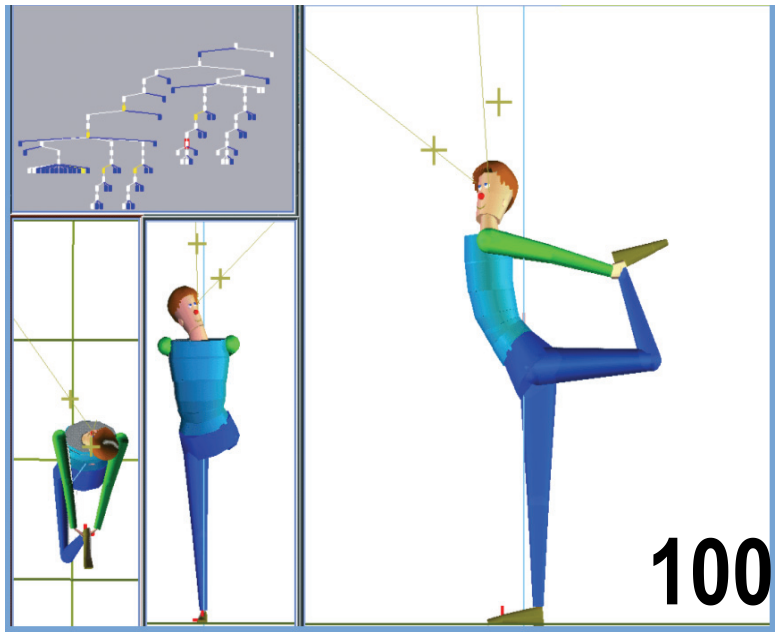
## Weighting strategy

- 50 and 100 iterations
- Low realism of intermediate postures
- The final posture does not achieve all constraints

57




58

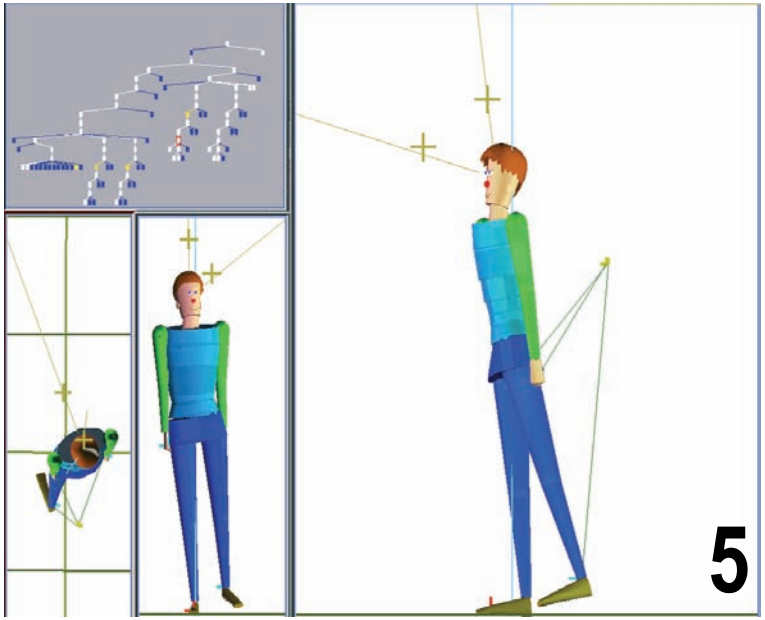



59

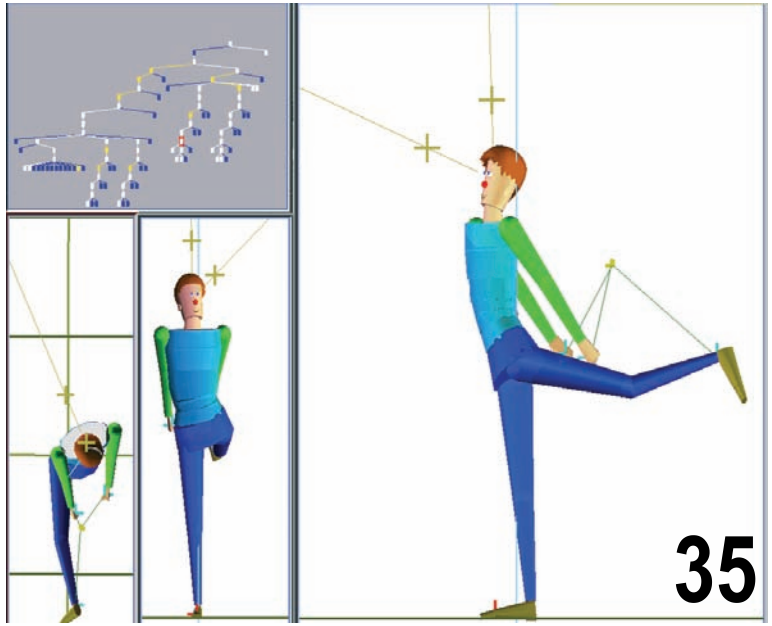
## Priority strategy

- 5, 35 and 75 iterations
- Intermediate postures ensure balance
- The final posture achieves all constraints

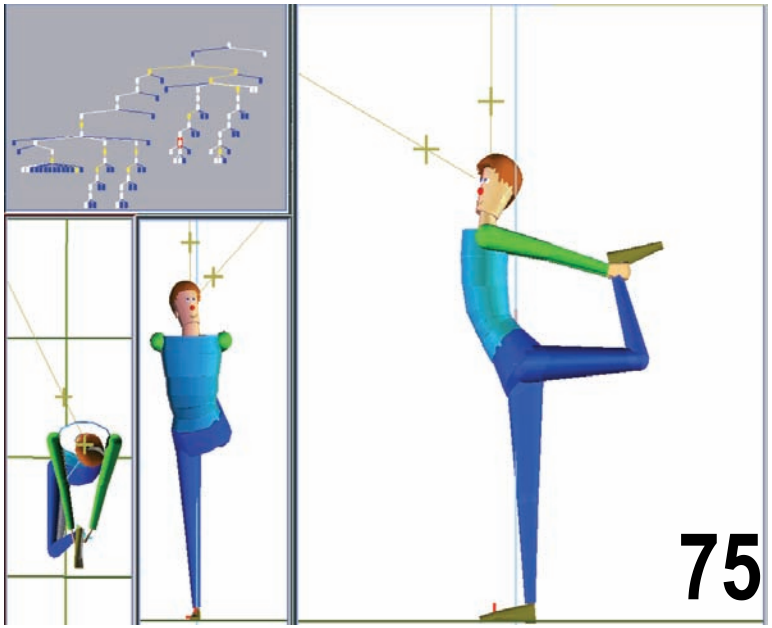
60




61




62




63

## Part B - Iterative methods (pseudo-inv.)

- Concept of prioritized control
- From one to P priority levels
  - General form
  - Incremental constraint achievement
  - Example1: the simplest redundant case
  - Example 2: two prioritized constraints
  - Generalization to multiple priority levels
  - ⇨ Fostering synergies
  - Handling singularities with DLS
  - Application: collision avoidance
  - Conclusion

64

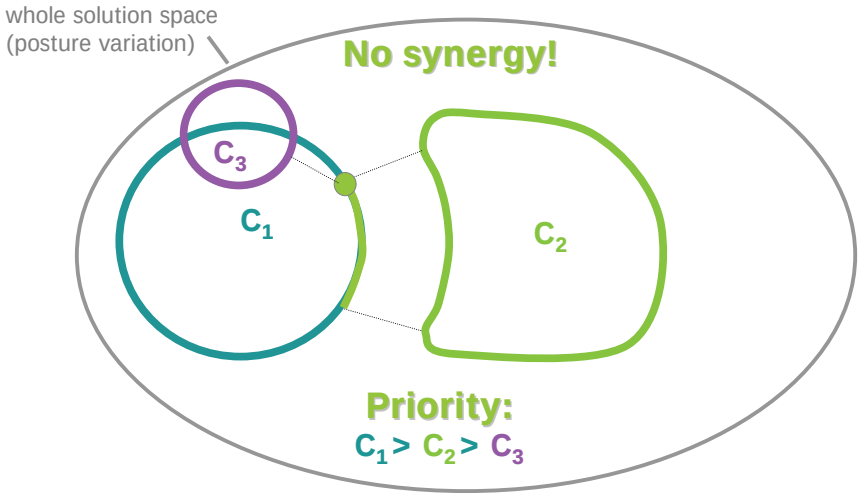
## Adopting a **synergistic** approach

- Involve all the joints to satisfy all the constraints
  - ⇧ larger solution space
  - ⇧ can find optimal solution because all the joints can contribute to the satisfaction of the constraints
  - ⇧ computing cost

65

## Conceptual analogy

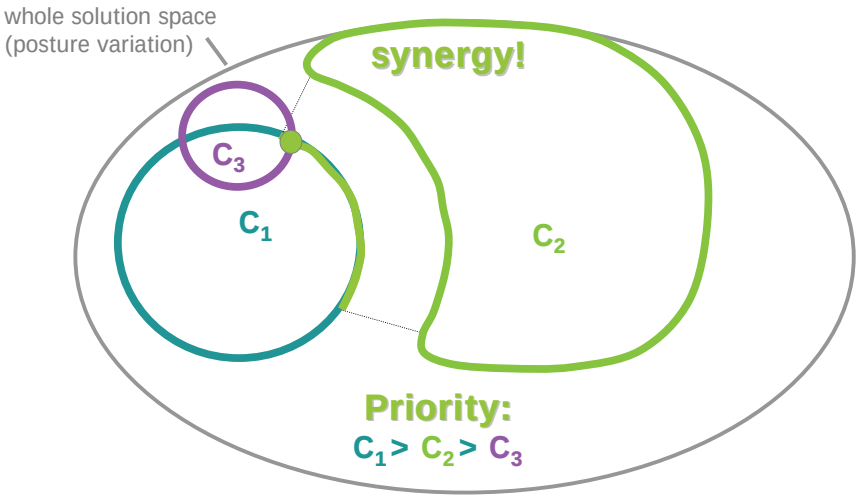
visualizing the constraints solution sub-spaces: $C_i$



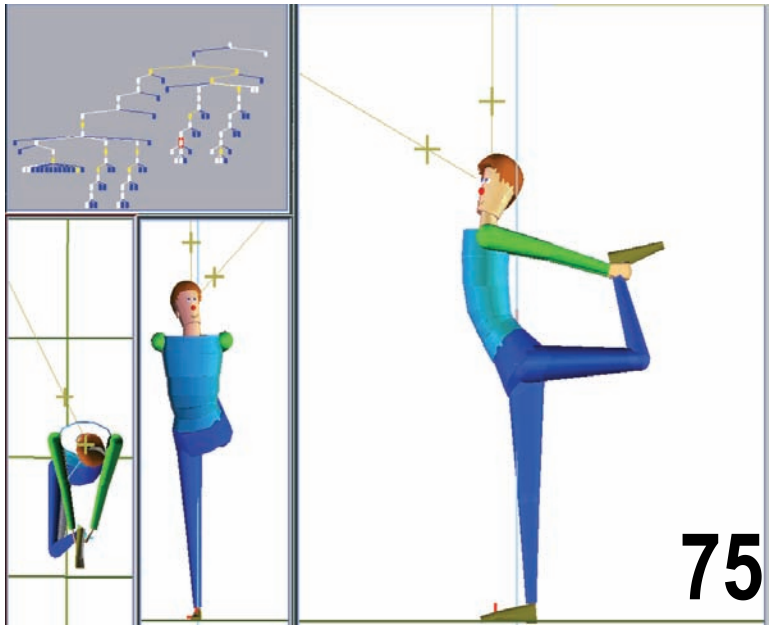

66

## Conceptual analogy

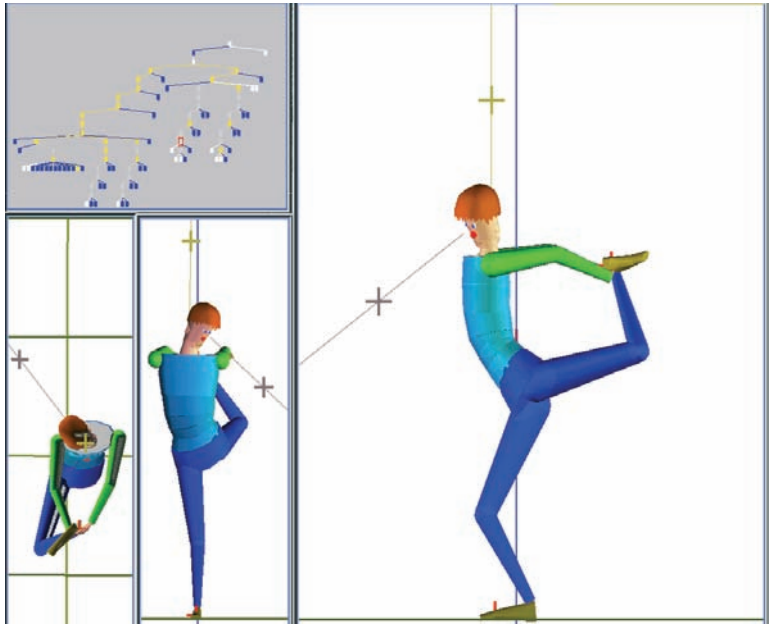
visualizing the constraints solution sub-spaces: $C_i$




67




68







# Part B - Iterative methods (pseudo-inv.)

- Concept of prioritized control
- From one to P priority levels
  - General form
  - Incremental constraint achievement
  - Example1: the simplest redundant case
  - Example 2: two prioritized constraints
  - Generalization to multiple priority levels
  - Fostering synergies
- ⇨ Handling singularities with DLS
  - Application: collision avoidance
  - Conclusion



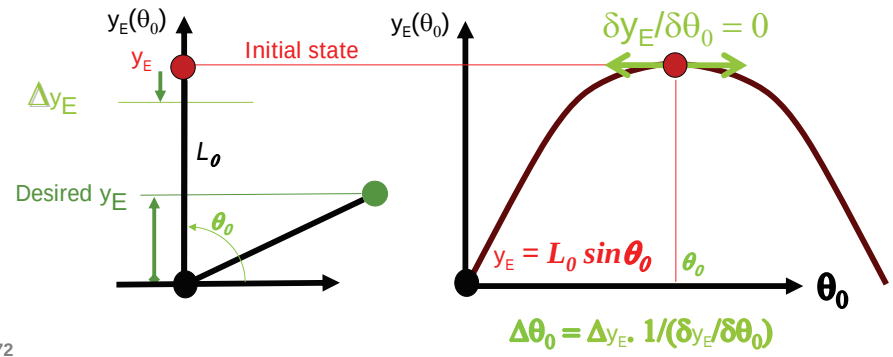
## Handling geometric and algorithmic singularities

- Context: mismatch between the requested task $\Delta x$ and the, temporary lower, rank of any Jacobian
- The solution is still valid in pure singular context :
  - When the rank of a jacobian $J$ decreases, the dimension of its Null Space increases
  - The n-dimensional $\Delta\alpha$ vector can be used to avoid, or to get out of the singularity
  - Attention : Around the singularity, the norm of the solution increases: instability

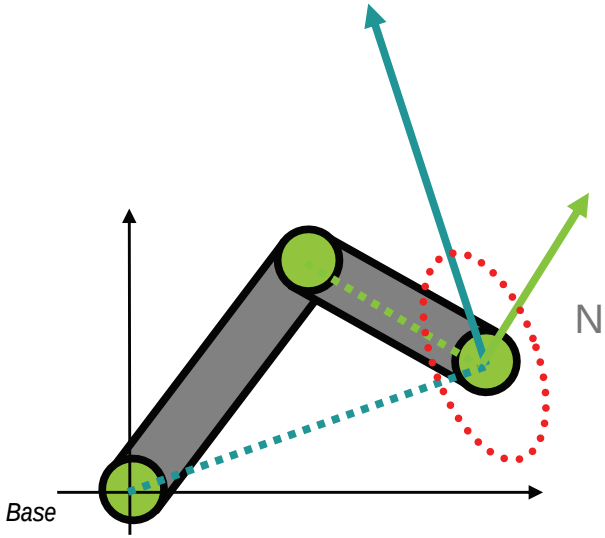


## Geometric singularity: the 1 D case

- When the derivative is null
  - The inverse is undefined: ok for the pseudo-inverse
- When the derivative is very small
  - The inverse is very big: instability



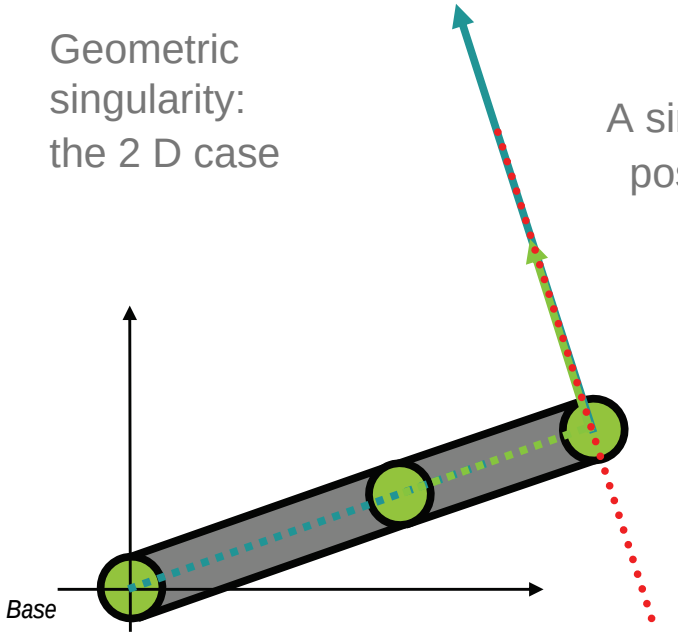
## Geometric singularity: the 2 D case




Non-singular posture

73

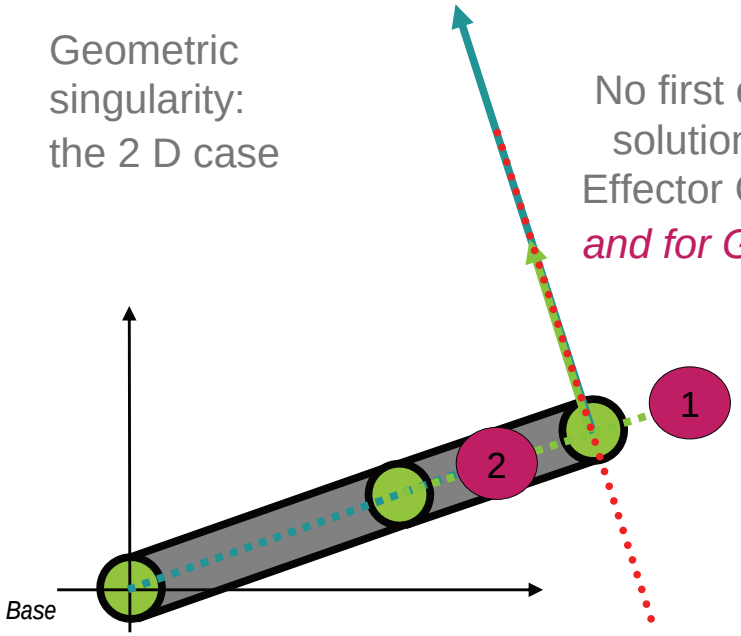
## Geometric singularity: the 2 D case




A singular posture

74

## Geometric singularity: the 2 D case




No first order solution for Effector Goal1

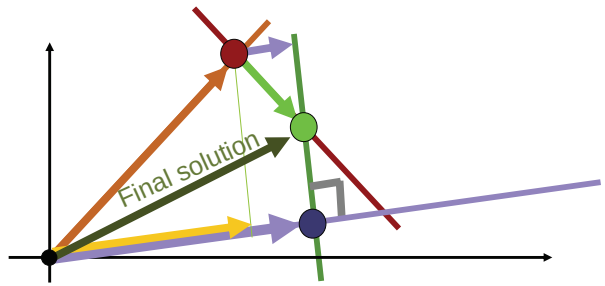
*and for Goal2!*

75

## Algorithmic singularity (1)

Two conflicting Constraints [Hanafusa81]:

NOT SINGULAR

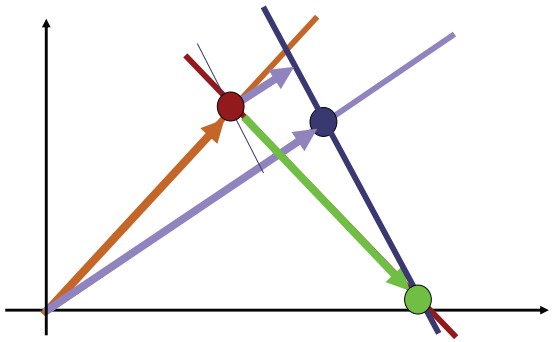



76

# Algorithmic singularity (2)

When the low priority task wants to do the same as the high priority task

nearly SINGULAR





# Managing singularities

- What is causing instability ?
  - Singular Value Decomposition of $J$

$$J = \sum_{i=1}^{r} \sigma_i u_i v_i^T$$

  - The Pseudo-Inverse $J^+$ is given by:

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



# Managing singularities (2)

- Regularization Technique [Maciejewsky90]

- The damped Pseudo-Inverse ensures that the norm of the solution is under a predefined value.

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

  - The $\lambda$ term damps the singular behavior of $J$ in the neighborhood of the singularity

  - *The projectors on the Null space have however to be built with the $J^+$ pseudo-inverse*



# Extension to *P* priority levels [Slotine91]

damping

$$\Delta\theta_i = (J_i P_{N(J_{i-1}^A)})^{+\lambda_i} (\Delta x_i - J_i \Delta\theta_{i-1})$$

NO damping

$$P_{N(J_i^A)} = P_{N(J_{i-1}^A)} - (J_i P_{N(J_{i-1}^A)})^+ (J_i P_{N(J_{i-1}^A)})$$



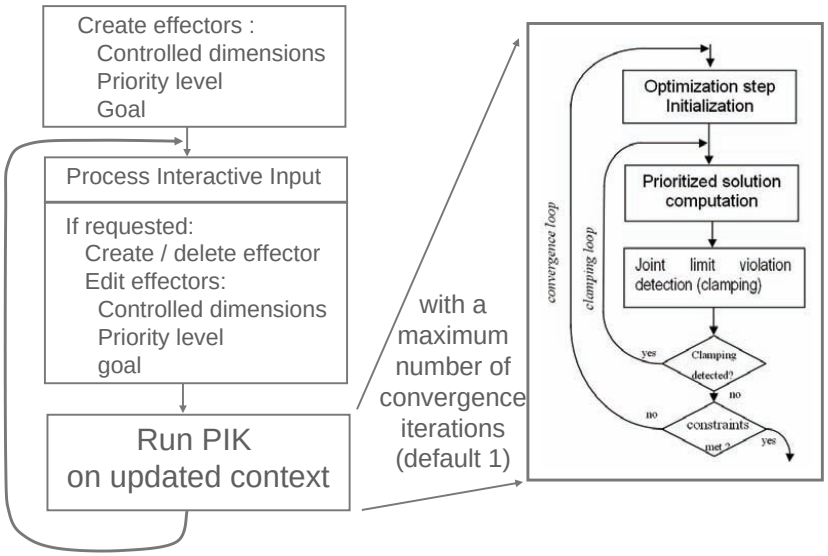


# Part B - Iterative methods (pseudo-inv.)

- Concept of prioritized control
- From one to P priority levels
  - General form
  - Incremental constraint achievement
  - Example1: the simplest redundant case
  - Example 2: two prioritized constraints
  - Generalization to multiple priority levels
  - Fostering synergies
  - Handling singularities with DLS
  - ⇨ Application: collision avoidance
  - Conclusion



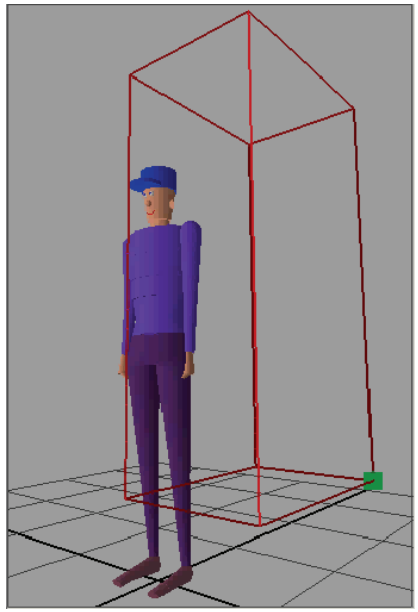
# Interactive application architecture






Visualization of off-line use for identifying the lateral reachable space under balance constraints

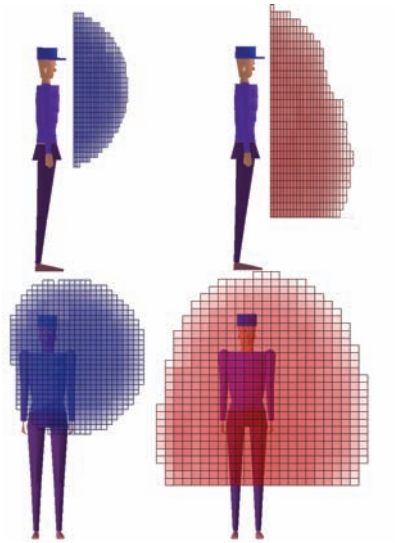
[Peinado & al. 04]



Example: standing reachable space of the left hand

blue: with only the arm

red: whole body

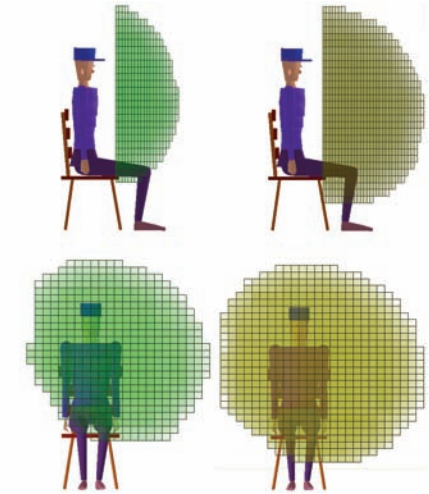


85

Example: seated reachable space of the left hand

green: naive approach

brown: with the hip joint

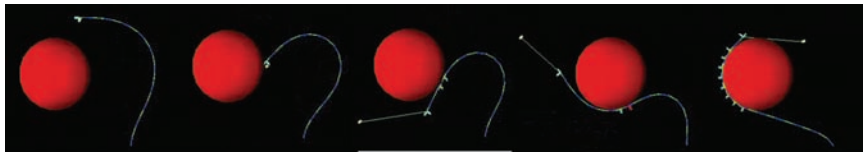


86

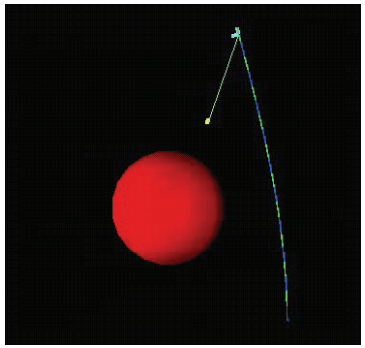
# Collision avoidance

- Management of inequality constraint
- Dynamic creation of repulsion effectors when there is an effective collision
- A Repulsion effector :
  - Is a one-dimensional position control to push out the colliding point
  - has a higher priority compared to the chain tip attraction toward a user-guided goal

87



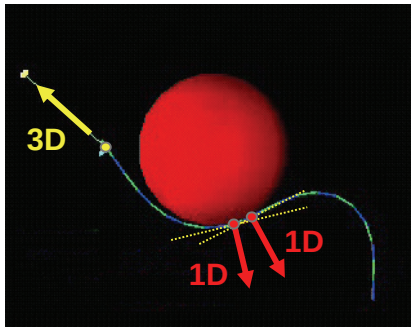
- Single effector
  - Chain tip



- Dynamically created 1D repulsion effectors with higher priority
  - may prevent the chain tip effector to reach its goal

88

- Chain tip effector
  - 3D position constraint toward an interactively set goal location

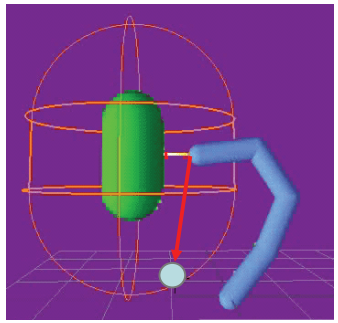



- 2 one dimensional repulsion effectors
  - Pushes away along the local normal
  - Free movement in the tangent plane

89

## Smooth Collision avoidance [Peinado et al 05]

- Obstacles are surrounded by a damping zone
- Special points on the articulated structure are "observed"



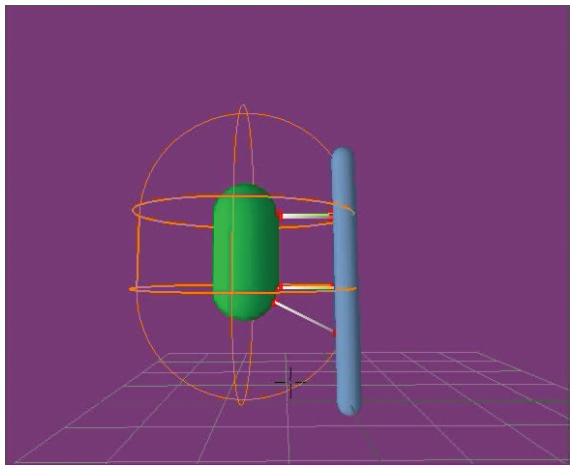
- An observer is promoted as a constrained one-dimensional dynamic damping effector whenever it is inside the damping zone and moves towards the obstacle

90

## Smooth Collision avoidance [Peinado et al 05]

The chain tip is interactively guided towards successive goal position

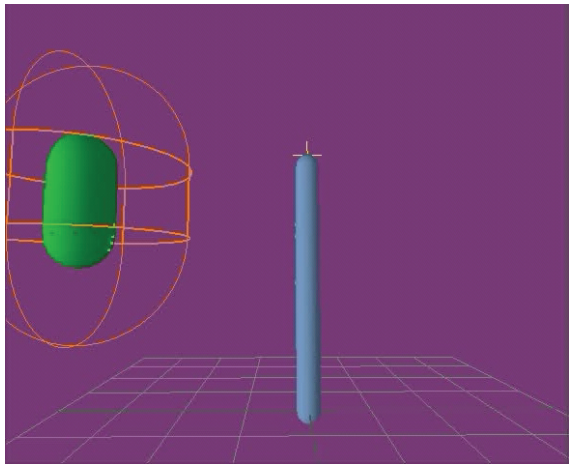
Links highlight the dynamic creation of 1D damping effectors to prevent collisions



91

## Smooth Collision avoidance [Peinado et al 06]

Taking into account the relative speed along the normal to the obstacle allows to anticipate the arrival of a mobile obstacle



92

## Part B - Iterative methods (pseudo-inv.)

- Concept of prioritized control
- From one to P priority levels
  - General form
  - Incremental constraint achievement
  - Example1: the simplest redundant case
  - Example 2: two prioritized constraints
  - Generalization to multiple priority levels
  - Fostering synergies
  - Handling singularities with DLS
  - Application: collision avoidance
  - Conclusion



## Conclusions & future work

- The prioritized IK provides a powerful way to combine various conflicting requirements such as collision avoidance, balance, gaze, reach:
  - When correctly chosen, the hierarchy of priorities guides the convergence through believable intermediate states, even if only a small subset of constraints are met.
- On-going work: integrating motion models to guide the convergence.

# Inverse kinematics and kinetics for virtual humanoids

Ronan Boulic

VRLab – EPFL
http://vrlab.epfl.ch

Richard Kulpa

Bunraku – INRIA / IRISA
www.irisa.fr/bunraku

M2S – Univ. Rennes 2
www.uhb.fr/labos/m2s

## Plan

- Part A - Overview
  - Motivation
  - Problem
  - Overview of the IK methods
- Part B - Iterative methods (pseudo-inverse based)
  - Concept of prioritized control
  - From one to P priority levels
- ⇨ Part C - Iterative methods (CCD-based)
  - CCD
  - Hybrid CCD and analytical
- Part D - Conclusion



## Part C - Iterative methods (CCD-based)

- ⇨ CCD
- Hybrid CCD and analytical method
  - Representation of data
  - Overview of the method
  - Kinematic solver
  - Kinetic solver
  - Kinematic and kinetic solver
  - Results
  - Conclusion



## CCD

- CCD (Cyclic Coordinate Descent)
- Goal of this kind of approach
  - Adaptation of the animation of hundreds of characters in real-time
- Compared to Jacobian
  - Solve only one DOF at a time

# CCD

- Only one column remains in the Jacobian

$$\begin{pmatrix} \Delta x_1 \\ \Delta x_2 \\ \vdots \\ \Delta x_M \end{pmatrix} = \begin{pmatrix} \frac{\partial f x_1}{\partial \theta_1} & \frac{\partial f x_1}{\partial \theta_2} & \cdots & \frac{\partial f x_1}{\partial \theta_N} \\ \frac{\partial f x_2}{\partial \theta_1} & \frac{\partial f x_2}{\partial \theta_2} & \cdots & \frac{\partial f x_2}{\partial \theta_N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f x_M}{\partial \theta_1} & \frac{\partial f x_M}{\partial \theta_2} & \cdots & \frac{\partial f x_M}{\partial \theta_N} \end{pmatrix} \begin{pmatrix} \Delta\theta_1 \\ \Delta\theta_2 \\ \vdots \\ \Delta\theta_N \end{pmatrix}$$

- Inversion is fast: low computation cost



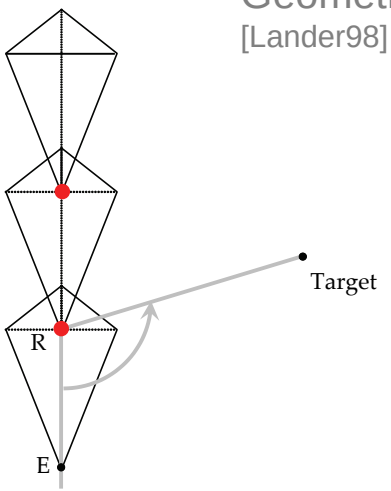
# CCD

- General algorithm
  - For all the DOF $\theta_i$
    - Compute variation $\Delta\theta_i$ of the DOF $\theta_i$
    - Add this variation: $\theta_i = \theta_i + \Delta\theta_i$
  - End
- Issued from robotics works [Luenberger84, Wang91]
- Animation: recursive algorithm [Badler87]



# CCD

- Geometrical resolution by CCD [Lander98]



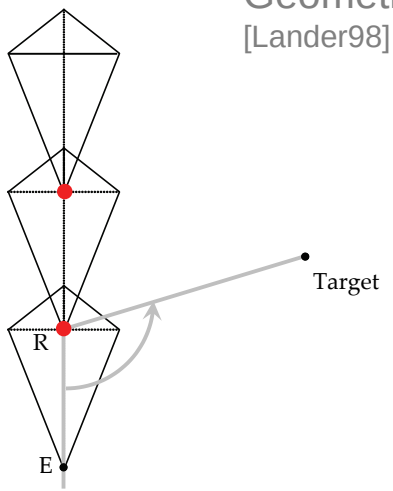

Example with 2 DOFs

E: effector
R: currently modified DOF



# CCD

- Geometrical resolution by CCD [Lander98]

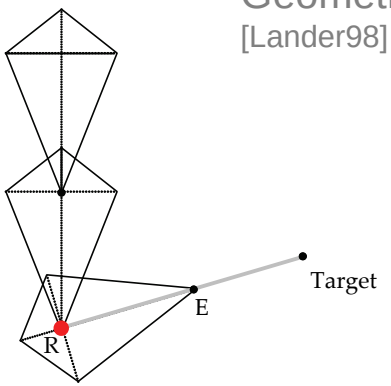



Example with 2 DOFs

E: effector
R: currently modified DOF

# CCD

- Geometrical resolution by CCD
  [Lander98]



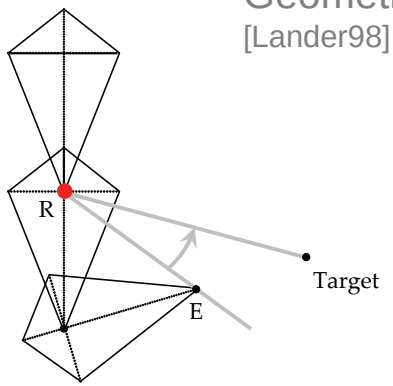

Example with 2 DOFs

E: effector
R: currently modified DOF



# CCD

- Geometrical resolution by CCD
  [Lander98]



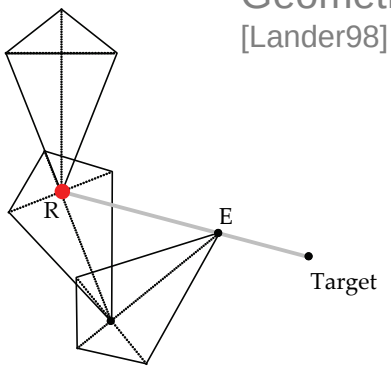

Example with 2 DOFs

E: effector
R: currently modified DOF



# CCD

- Geometrical resolution by CCD
  [Lander98]



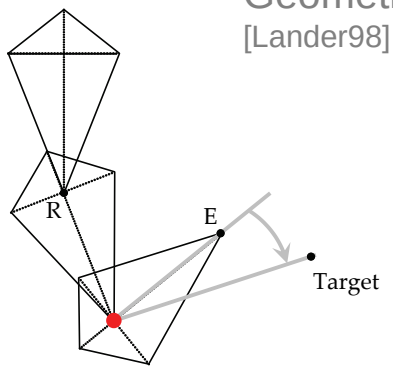

Example with 2 DOFs

E: effector
R: currently modified DOF



# CCD

- Geometrical resolution by CCD
  [Lander98]

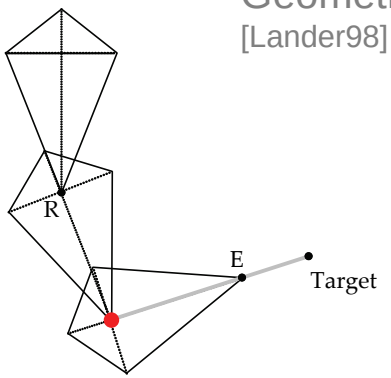



Example with 2 DOFs

E: effector
R: currently modified DOF

# CCD

- Geometrical resolution by CCD
  [Lander98]



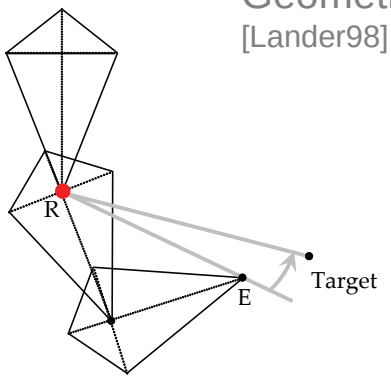

Example with 2 DOFs

E: effector
R: currently modified DOF



# CCD

- Geometrical resolution by CCD
  [Lander98]



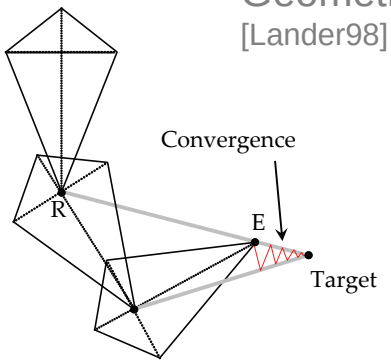

Example with 2 DOFs

E: effector
R: currently modified DOF



# CCD

- Geometrical resolution by CCD
  [Lander98]



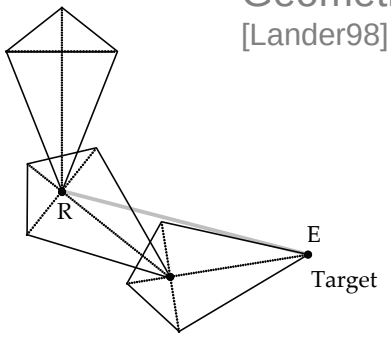

Example with 2 DOFs

E: effector
R: currently modified DOF



# CCD

- Geometrical resolution by CCD
  [Lander98]




Example with 2 DOFs

E: effector
R: currently modified DOF

# CCD

- Advantages
  - Fast computation of one iteration
  - Easy to implement
  - Handling of joint limits



# CCD

- Drawbacks
  - Slow convergence
  - Bad distribution of the adaptation
  - Unnatural posture

Original posture | Desired posture | Resulting posture



# CCD

- Drawbacks
  - Slow convergence
  - Bad distribution of the adaptation
  - Unnatural posture

⇒ First joints are more modified than the following ones!



# CCD

- First solution: use damping
  - Threshold on the variation of the joint parameters
  - Minimizes the adaptation of each joint
  - But increases the number of iterations
- More homogeneous adaptation
  ⇒ Bigger computation cost

# Not suitable for humanoids

- Our goal
  - Find natural postures
  - Computation time compatible with interactive animation of hundreds of characters
- CCD is not suitable for postural adaptation of humanoids
- Here, we present an hybrid CCD and analytical solver based on body groups



# Part C - Iterative methods (CCD-based)

- CCD
- ⇨ Hybrid CCD and analytical method
  - Representation of data
  - Overview of the method
  - Kinematic solver
  - Kinetic solver
  - Kinematic and kinetic solver
  - Results
  - Conclusion



# Hybrid CCD – Analytical method

- Idea 1:
  - Improve its main advantage: fast computation

⇒ Use analytical solutions

⇒ Decrease the number of segments



# Hybrid CCD – Analytical method

- Idea 2:
  - Take advantage of its main drawback: bad distribution of the adaptation

⇒ Most modified segments: first adapted ones

⇒ Use body groups

⇒ Order the adaptation of these groups

⇒ Lead to natural postures

# Hybrid CCD – Analytical method

- Idea 2:
  - Take advantage of its main drawback: bad distribution of the adaptation

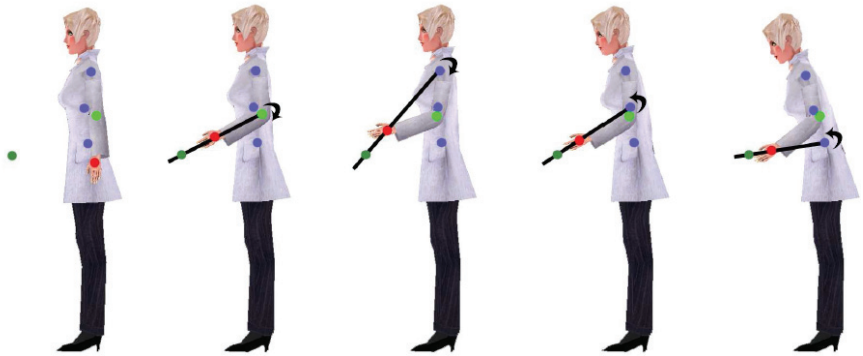
⇒ Use analytical solutions for the kinematic chain ends

⇒ Instantaneously have its best configuration

25

# Hybrid CCD – Analytical method

- Without analytical solutions for the arm



26

# Hybrid CCD – Analytical method

- With analytical solutions for the arm

27

# Hybrid CCD – Analytical method

- With analytical solutions for the arm

⇒ Minimizes the adaptation of the remainder of the body

28

# Hybrid CCD – Analytical method

- Here, we present an hybrid algorithm
  - CCD-like algorithm
  - Body groups to order the adaptations
  - Analytical solutions

29

# Part C - Iterative methods (CCD-based)

- CCD
- Hybrid CCD and analytical method

⇨
  - Representation of data
  - Overview of the method
  - Kinematic solver
  - Kinetic solver
  - Kinematic and kinetic solver
  - Results
  - Conclusion

30

# Representation of data

- Representation of the constraints
  - How to handle many kind of constraints such as position, orientation, space
- Representation of the skeleton
  - Find a suitable representation of the kinematic chain for adaptations

31

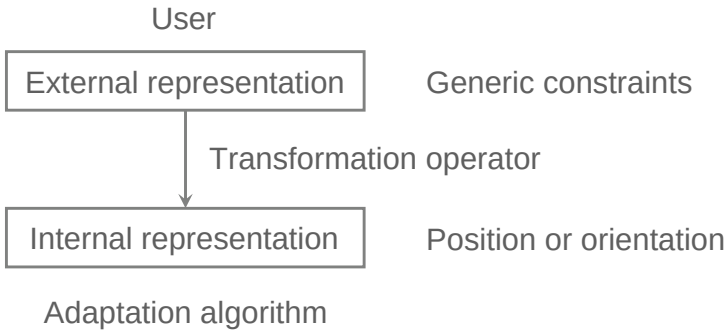
# Representation of constraints

- For performance purpose, algorithm deals only with geometric constraints (position / orientation)
- Motion: succession of postures
  => also deals with kinematic constraints (such as speed for example)

32

# Representation of constraints

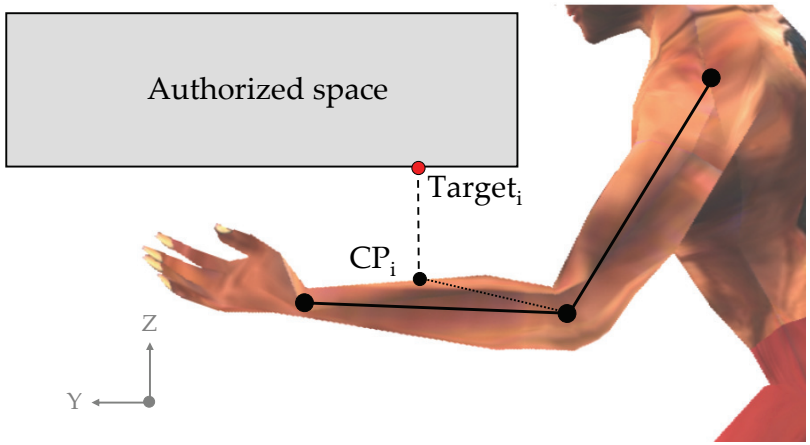
- In order to take all type of constraints into account:
  - Two levels of representation






# Internal representation

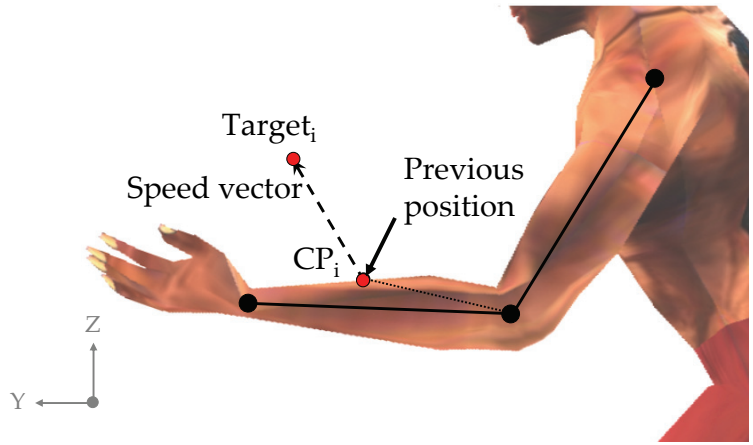
- Example of transformation operator for an authorized space constraint






# Internal representation

- Example of transformation operator for a speed constraint



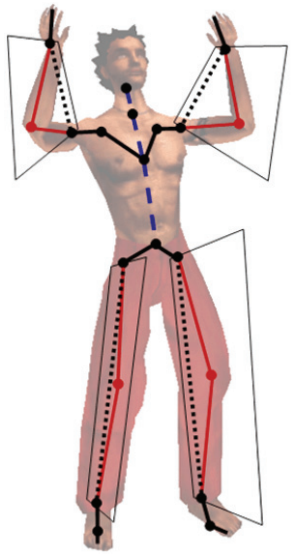



# Representation of data

- Representation of the constraints
  - How to handle many kind of constraints such as position, orientation, space
- Representation of the skeleton
  - Find a suitable representation of the kinematic chain for adaptations

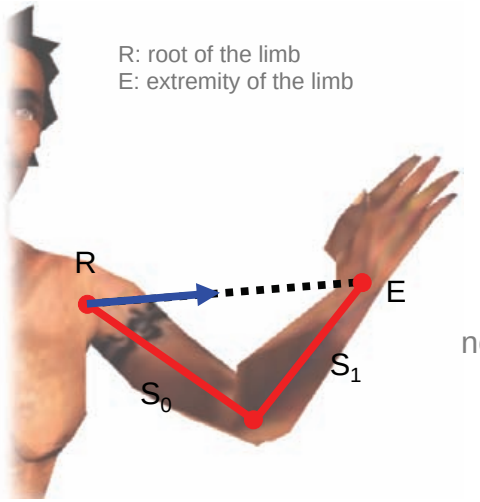
# Normalized skeleton

- Really suitable for adaptations [Kulpa05a]
- Skeleton representation independent from morphology
- Skeleton subdivided in:
  - Limbs
  - Spine
  - Other segments





# Normalized skeleton

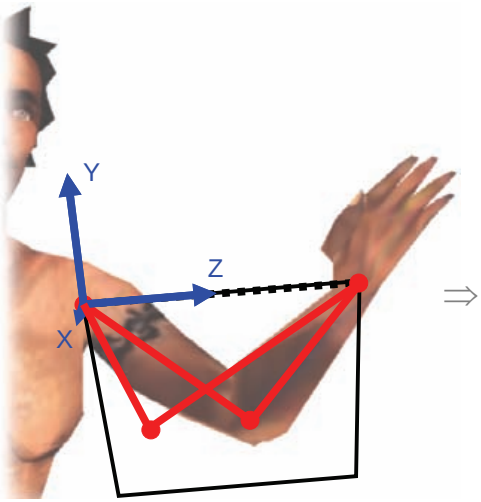



R: root of the limb
E: extremity of the limb

- Limbs are independent from the initial length
  - Use the vector from the shoulder to the wrist
  - Normalize it by the arm length

$$\text{normalizedKC} = \frac{E - R}{\Sigma_j \text{ length}(S_j)}$$

# Normalized skeleton




- Limbs are independent from the ratio between segment lengths
  - Store the half-plane that contains the elbow

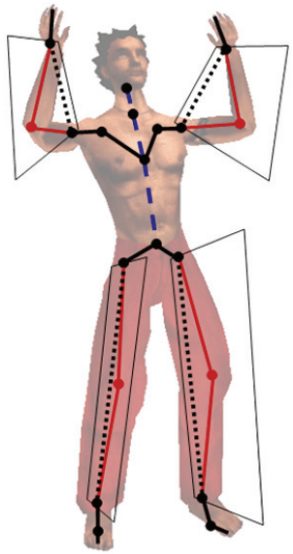
⇒ Limbs are defined by:

- Frame set: define the vector and the half-plane
- Scalar: percentage of the total limb length

# Normalized skeleton

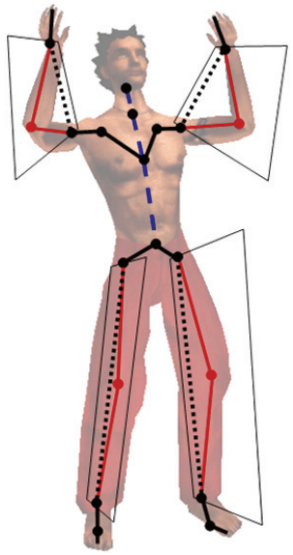
- The spine is represented by a spline
  - ⇒ Can be subdivided into any number of vertebraes
- Other segments are normalized by the original length

$$\text{normalizedKC} = \frac{E - R}{\Sigma_j \text{ length}(S_j)}$$



# Normalized skeleton

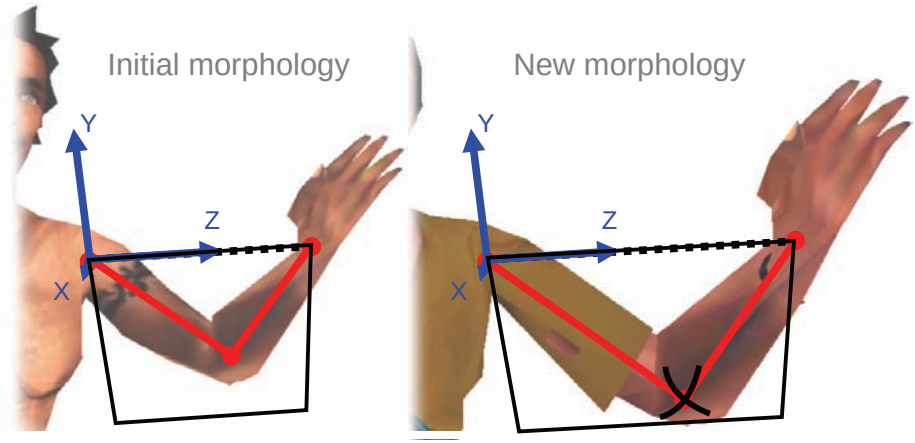
- Root position is normalized by the leg length (should be extended if walking on the hands for example)
- Root orientation is subdivided into :
  - Global orientation
    Ex: direction of the walk
  - Local orientation
    Ex: twist of the pelvis around the global direction





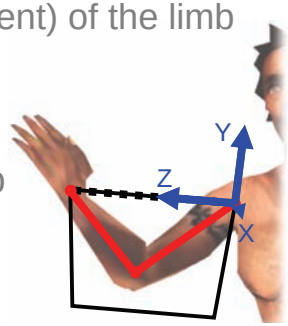
# Retargeting

- Invert the normalization process




# Representation of the skeleton

- All the analytical solutions are based on this representation
- Example 1: extend a limb
  - Change the scalar (length percent) of the limb
- Example 2: reach a target
  - Change the Z vector of the limb
  - Change the scalar of the limb






# Representation of the skeleton

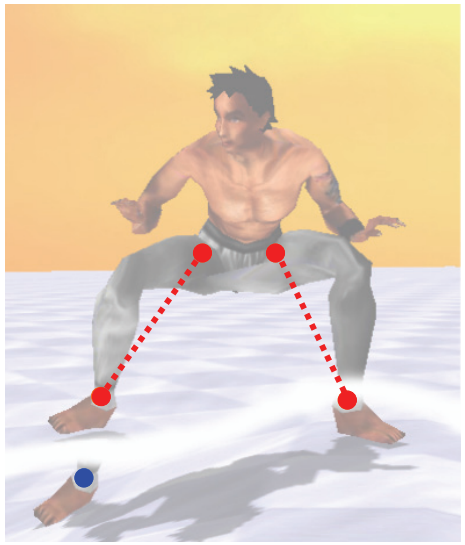
- Example 3: placing the feet on the ground
  [Kulpa05a]

Original posture

# Representation of the skeleton

- Example 3: placing the feet on the ground [Kulpa05a]

Normalized limbs



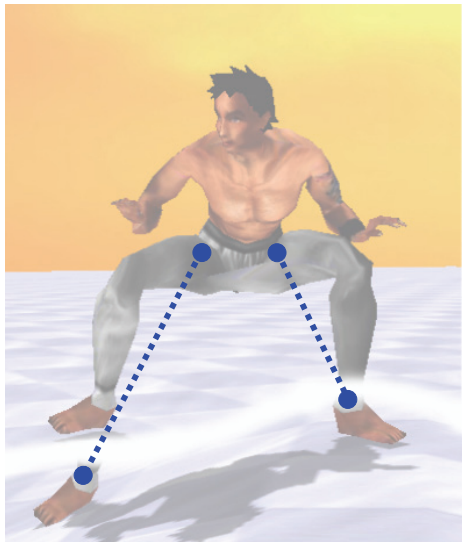


# Representation of the skeleton

- Example 3: placing the feet on the ground [Kulpa05a]

Changing:

- Vector of the limb
- Scale of the limb





# Representation of the skeleton

- Example 3: placing the feet on the ground [Kulpa05a]

Final posture



# Part C - Iterative methods (CCD-based)

- CCD
- Hybrid CCD and analytical method
  - Representation of data
  - Overview of the method
  - Kinematic solver
  - Kinetic solver
  - Kinematic and kinetic solver
  - Results
  - Conclusion

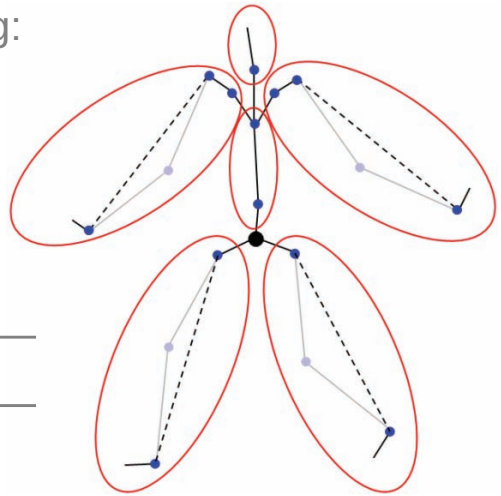
# Adaptation algorithm

- Idea: take advantage of the different body parts
  - More or less movable
  - More or less heavy
- Subdivide the skeleton into groups
- Order the adaptation in order to take advantage of the CCD unhomogeneous distribution
- Unique representation compatible with :
  - Inverse kinematics
  - Inverse kinetics

49

# Hierarchy of groups

- Split the skeleton into groups
- 2 levels of ordering:
  - Between groups
  - Within groups
- Hierarchy:



1

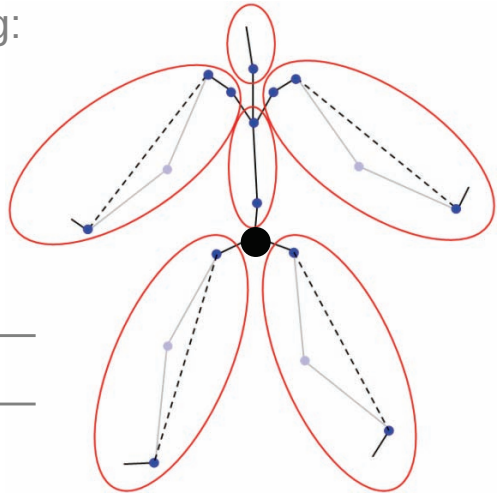
2

3

50

# Hierarchy of groups

- Split the skeleton into groups
- 2 levels of ordering:
  - Between groups
  - Within groups
- Hierarchy:



1 Root

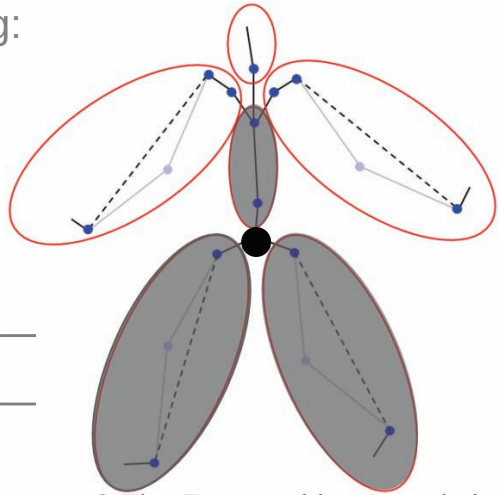
2

3

51

# Hierarchy of groups

- Split the skeleton into groups
- 2 levels of ordering:
  - Between groups
  - Within groups
- Hierarchy:



1 Root

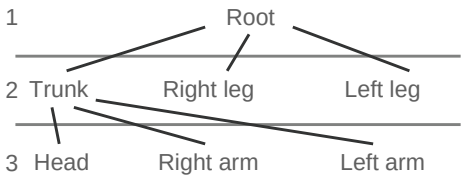
2 Trunk Right leg Left leg

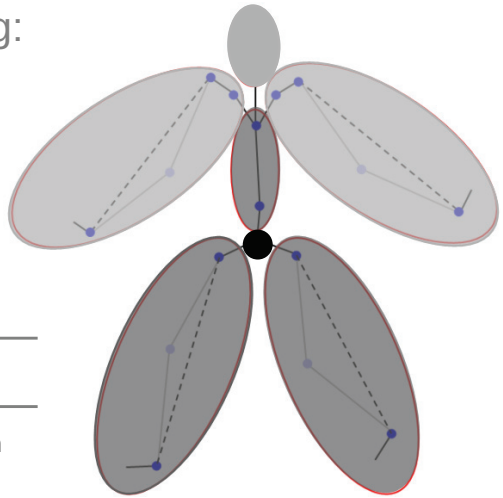
3

52

# Hierarchy of groups

- Split the skeleton into groups
- 2 levels of ordering:
  - Between groups
  - Within groups
- Hierarchy:



53

# Adaptation order of the groups

- Minimize the adaptation error
- Inverse kinematics
  - Groups with the largest range first
  - Level 3 to 1
- Inverse kinetics
  - Heaviest groups first
  - Level 1 to 3
- Order used between and within the groups

54

# Part C - Iterative methods (CCD-based)

- CCD
- Hybrid CCD and analytical method
  - Representation of data
  - Overview of the method
  - ⇨ Kinematic solver
  - Kinetic solver
  - Kinematic and kinetic solver
  - Results
  - Conclusion

55

# Kinematic adaptation order

- Bad kinematic order:
  heaviest segments first

56

# Kinematic adaptation order

- Bad kinematic order:
  heaviest segments first

57

# Kinematic adaptation order

- Bad kinematic order:
  heaviest segments first

58

# Kinematic adaptation order

- Good kinematic order:
  lightest segments first

59

# Kinematic adaptation order

- Good kinematic order:
  lightest segments first

60

# Kinematic adaptation order

- Good kinematic order: lightest segments first

61

# Kinematic adaptation order

- Good kinematic order: lightest segments first

62

# Kinematic adaptation order

- Good kinematic order: lightest segments first

63

# Inverse kinematics algorithm

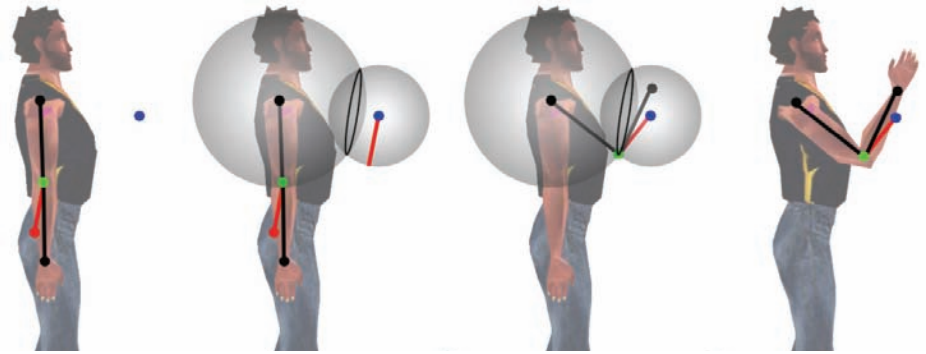
- Inverse kinematics algorithm
- Do
  - groupAdaptation(Head)
  - groupAdaptation(Right arm)
  - groupAdaptation(Left arm)
  - groupAdaptation(Right leg)
  - groupAdaptation(Left leg)
  - groupAdaptation(Trunk)
  - rootAdaptation()
- While (¬ended)

Level 3 (groupAdaptation(Head), groupAdaptation(Right arm), groupAdaptation(Left arm))
Level 2 (groupAdaptation(Right leg), groupAdaptation(Left leg), groupAdaptation(Trunk))
Level 1 (rootAdaptation())

64

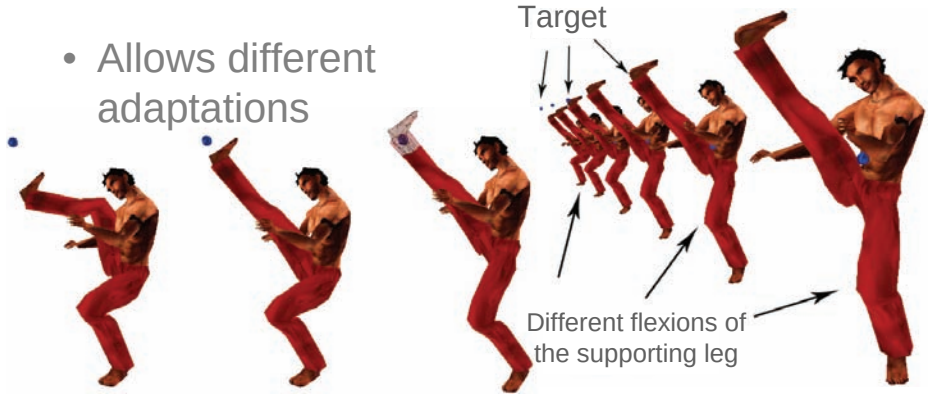
# Group adaptation

- Analytical solutions for limbs
  (for weighted or prioritized constraints)
- Rotations for the remainder of the body



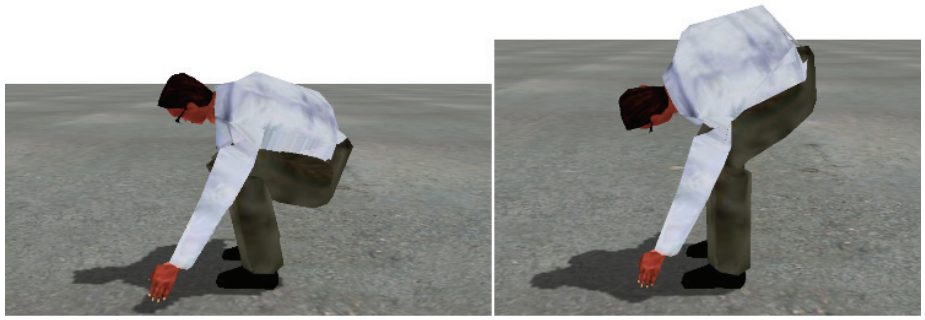
# Root adaptation

- For the constraints applied on the pelvis
- To reach far constraints (additional parameter)
- Allows different adaptations




# Root adaptation

- Allows the use of biomechanical laws
  - Use of individual parameters
    (such as joint range…)





# Part C - Iterative methods (CCD-based)

- CCD
- Hybrid CCD and analytical method
  - Representation of data
  - Overview of the method
  - Kinematic solver
  - ⇨ Kinetic solver
  - Kinematic and kinetic solver
  - Results
  - Conclusion

# Kinetic adaptation

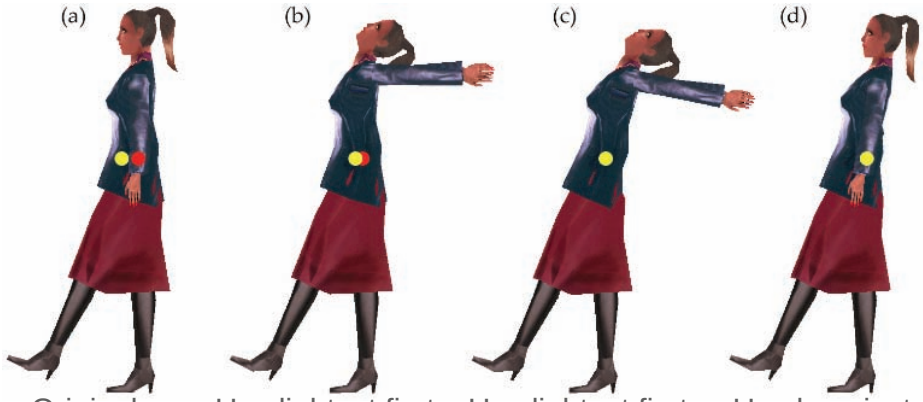
- Inverse kinetics only considers the groups not constrained (not used by the inverse kinematics solver)
- Among these groups, the user can select only a subset

⇒ Can tune the inverse kinetics adaptation



# Kinetic adaptation order

- Good kinetic order: heaviest first

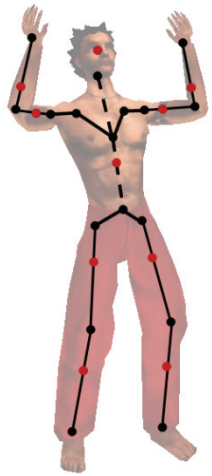



Original posture | Use lightest first (head then arms) | Use lightest first (finally trunk) | Use heaviest first (trunk)



# Center of mass hierarchy

- Different levels of COM [Kulpa05b]



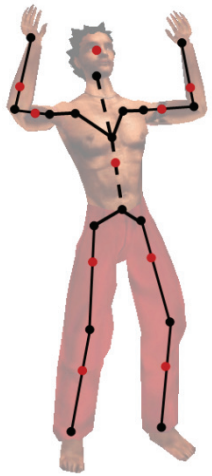
Weigthed sum so can be subdivided according to the groups

$$COM = \frac{\sum m_S COM_S}{\sum m_S}$$



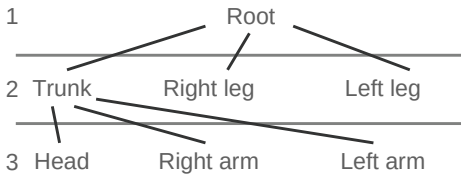
# Center of mass hierarchy

- Different levels of COM [Kulpa05b]



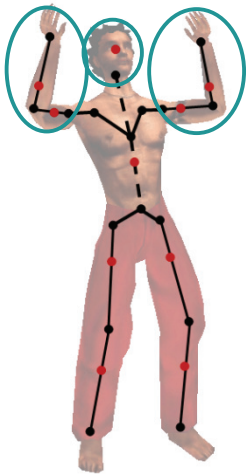
Weigthed sum so can be subdivided according to the groups

$$COM = \frac{\sum m_S COM_S}{\sum m_S}$$



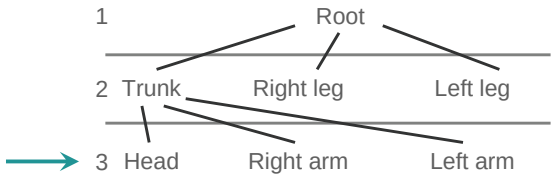
# Center of mass hierarchy

- Different levels of COM [Kulpa05b]



Weigthed sum so can be subdivided according to the groups

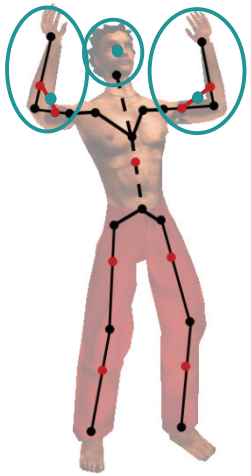
$$COM = \frac{\sum m_S COM_S}{\sum m_S}$$




73

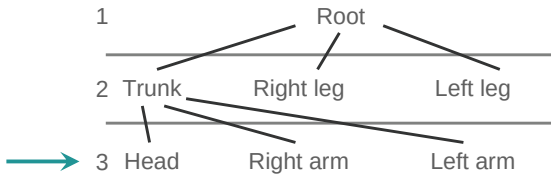
# Center of mass hierarchy

- Different levels of COM [Kulpa05b]



Weigthed sum so can be subdivided according to the groups

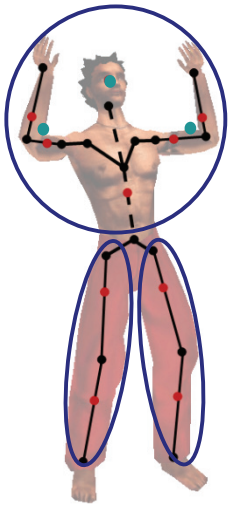
$$COM = \frac{\sum m_S COM_S}{\sum m_S}$$




74

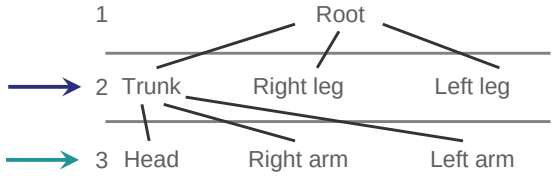
# Center of mass hierarchy

- Different levels of COM [Kulpa05b]



Weigthed sum so can be subdivided according to the groups

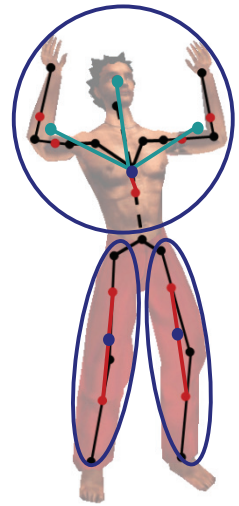
$$COM = \frac{\sum m_S COM_S}{\sum m_S}$$




75

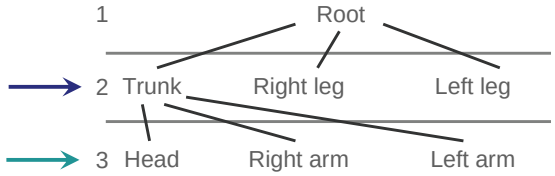
# Center of mass hierarchy

- Different levels of COM [Kulpa05b]



Weigthed sum so can be subdivided according to the groups

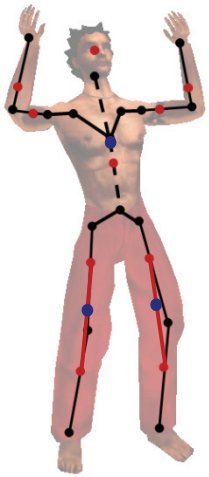
$$COM = \frac{\sum m_S COM_S}{\sum m_S}$$




76

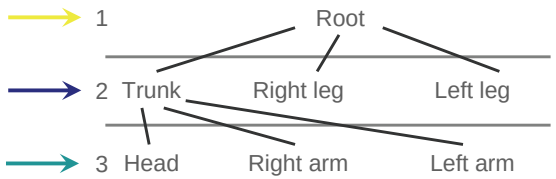
# Center of mass hierarchy

- Different levels of COM [Kulpa05b]



Weigthed sum so can be subdivided according to the groups

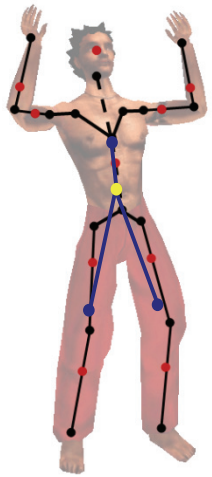
$$COM = \frac{\sum m_S COM_S}{\sum m_S}$$





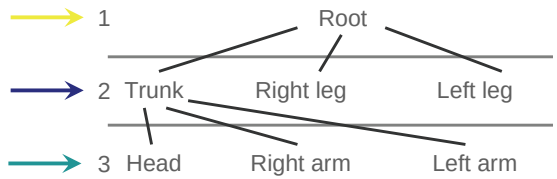

# Center of mass hierarchy

- Different levels of COM [Kulpa05b]



Weigthed sum so can be subdivided according to the groups

$$COM = \frac{\sum m_S COM_S}{\sum m_S}$$






# Adaptation of the group G

1. Error between COM and COM'




# Adaptation of the group G

1. Error between COM and COM'
2. Computation of desired $COM'_G$

$$COM' = \frac{\sum_g (m_g COM_g)}{\sum_g m_g}$$

# Adaptation of the group G

1. Error between COM and COM'
2. Computation of desired COM'$_G$

$$COM' = \frac{\sum_g (m_g COM_g)}{\sum_g m_g}$$

$$COM' = \frac{\sum_{g \neq G} (m_g COM_g)}{\sum_g m_g} + \frac{m_G COM'_G}{\sum_g m_g}$$


81

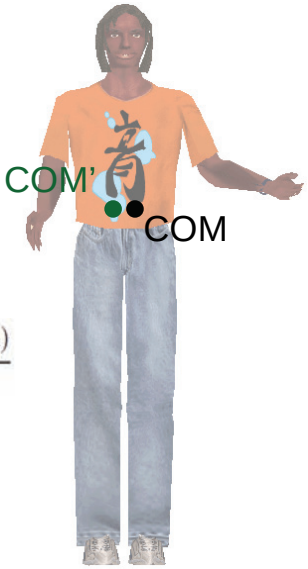
# Adaptation of the group G

1. Error between COM and COM'
2. Computation of desired COM'$_G$

$$COM' = \frac{\sum_g (m_g COM_g)}{\sum_g m_g}$$

$$COM' = \frac{\sum_{g \neq G} (m_g COM_g)}{\sum_g m_g} + \frac{m_G COM'_G}{\sum_g m_g}$$

$$COM'_G = \frac{\sum_g m_g * COM' - \sum_{g \neq G} (m_g COM_g)}{m_G}$$




82

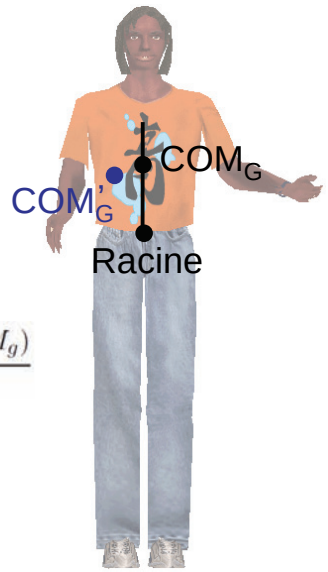
# Adaptation of the group G

1. Error between COM and COM'
2. Computation of desired COM'$_G$

$$COM' = \frac{\sum_g (m_g COM_g)}{\sum_g m_g}$$

$$COM' = \frac{\sum_{g \neq G} (m_g COM_g)}{\sum_g m_g} + \frac{m_G COM'_G}{\sum_g m_g}$$

$$COM'_G = \frac{\sum_g m_g * COM' - \sum_{g \neq G} (m_g COM_g)}{m_G}$$




83

# Adaptation of the group G

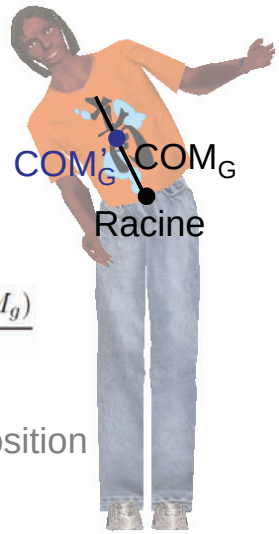
1. Error between COM and COM'
2. Computation of desired COM'$_G$

$$COM' = \frac{\sum_g (m_g COM_g)}{\sum_g m_g}$$

$$COM' = \frac{\sum_{g \neq G} (m_g COM_g)}{\sum_g m_g} + \frac{m_G COM'_G}{\sum_g m_g}$$

$$COM'_G = \frac{\sum_g m_g * COM' - \sum_{g \neq G} (m_g COM_g)}{m_G}$$

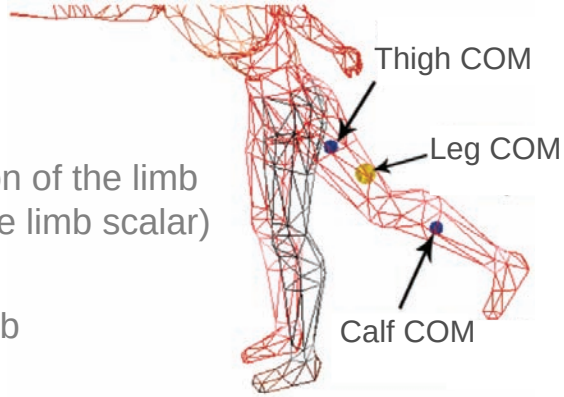
3. Adaptation of the group G
   to place the COM at the right position




84

# Adaptation of the groups

- Analytical solution for limbs




- Two steps:
  - Find the flexion of the limb (defined by the limb scalar)
  - Rotate the limb



# Root adaptation

- Translation of:
  - 3D vector (COM' – COM) for aerial phases: can handle real COM trajectory such as parabola
  - 2D vector (COM' – COM) for support phases: can preserve balance handling the projection of the COM on the ground
- With:
  - COM: current COM of the posture
  - COM': desired COM



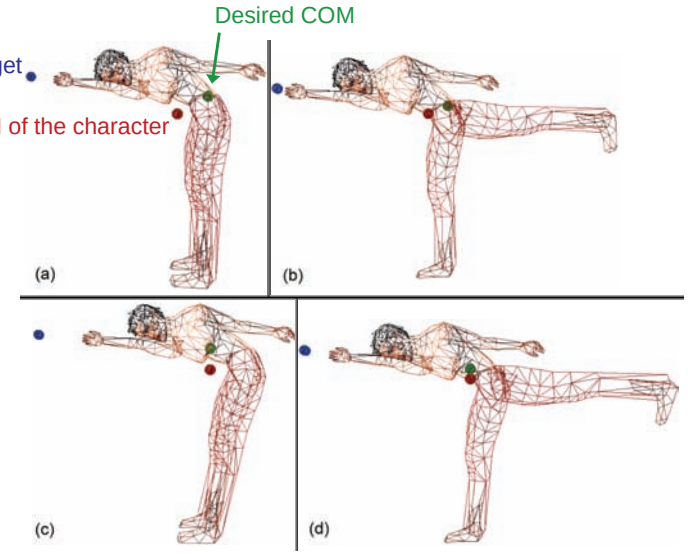
# Parameterization of the adaptations

(a) Reference posture

(b) Using groups

(c) Using root

(d) Using both groups + root






# Part C - Iterative methods (CCD-based)

- CCD
- Hybrid CCD and analytical method
  - Representation of data
  - Overview of the method
  - Kinematic solver
  - Kinetic solver
  - ⇨ Kinematic and kinetic solver
  - Results
  - Conclusion

# IKK algorithm

- If the user activates the kinetic adaptation, we consider that the balance must be preserved

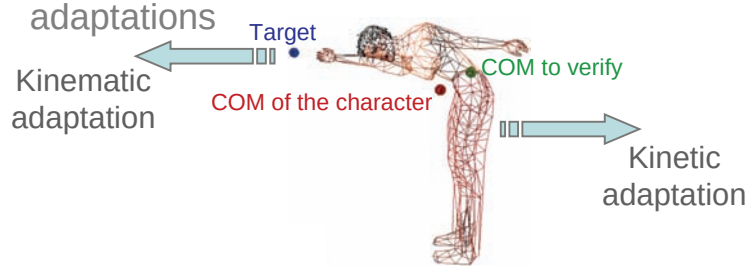
- Do

  KinematicAdaptation()
  If (error COM > threshold)
  kineticAdaptation()
  End of if
  While (¬ended)



# IKK algorithm

- Always converges because:
  - Kinematic and kinetic adaptations don't modify the same group (kinetic adaptation only applied on available groups: with no constraints on them)
  - Inverse kinematics and kinetics are antagonist adaptations






# Part C - Iterative methods (CCD-based)

- CCD
- Hybrid CCD and analytical method
  - Representation of data
  - Overview of the method
  - Kinematic solver
  - Kinetic solver
  - Kinematic and kinetic solver
  - Results
  - Conclusion



# Results

- Comparison with a real posture

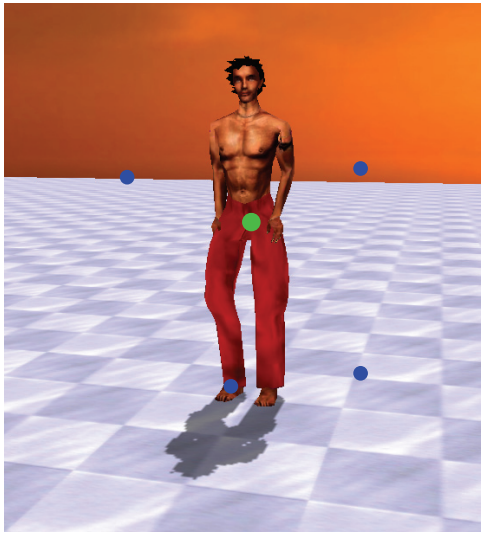
# Results

- Comparison with a real posture

93

# Results

- Comparison with a real posture



94

# Results

- Comparison with a real posture

95

# Results

- Comparison with a real posture

96

# Results

- Comparison with a real posture

Importance of the reference posture!



# Results

- The reference posture is also used to find the knees and the elbows
- Works even if the limbs are straightened in the reference posture
- Indeed, limb is represented by a frame and a scalar
  - The frame gives the reference orientation of the limb



# Results

- Real-time adaptation of many characters

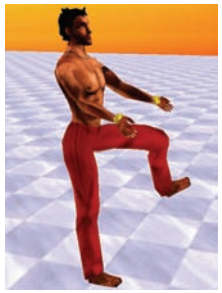



# Results

- Additional masses

# Performances

- Performances at 25 Hz (laptop - P4 2.8GHz)
  - Simple case (3 groups):
    - 454 characters (kinematics)
    - 212 characters (kinematics and kinetics)
  - Complex case (4 groups):
    - 90 characters (kinematics)
    - 35 characters (kinematics and kinetics)





# Part C - Iterative methods (CCD-based)

- CCD
- Hybrid CCD and analytical method
  - Representation of data
  - Overview of the method
  - Kinematic solver
  - Kinetic solver
  - Kinematic and kinetic solver
  - Results
  - Conclusion



# Conclusion

- Hybrid CCD on groups – Analytical

- Advantages
  - Very fast
  - Give natural postures
  - Easy to implement
  - Different adaptations for each characters
- Drawback
  - Cannot deal with biomechanical laws such as energy minimization

# Inverse kinematics and kinetics for virtual humanoids

Ronan Boulic
VRLab – EPFL
http://vrlab.epfl.ch

Richard Kulpa
Bunraku – INRIA / IRISA
www.irisa.fr/bunraku

M2S – Univ. Rennes 2
www.uhb.fr/labos/m2s

# Plan

- Part A - Overview
  - Motivation
  - Problem
  - Overview of the IK methods
- Part B - Iterative methods (pseudo-inverse based)
  - Concept of prioritized control
  - From one to P priority levels
- Part C - Iterative methods (CCD-based)
  - CCD
  - Hybrid CCD and analytical
- ⇨ Part D - Conclusion



# Conclusion

- Many IK methods
  - Analytical
    - Fast
    - Easy to implement
    - For a maximum of 7 DOF
  - Optimization
    - No singularity
    - Can fall to a local minimum (not the best solution)
    - Cannot handle prioritized constraints



# Conclusion

- Many IK methods
  - Iterative method (simultaneous)
    - Jacobian transpose
      - Fast
      - Unrealistic results
    - Jacobian pseudo-inverse
      - Precise
      - Can handle biomechanical laws
      - Can preserve balance
      - Manage prioritized and weighted constraints
      - For a few characters at real-time

# Conclusion

- Many IK methods
  - Iterative method (sequential)
    - CCD
      - Fast
      - Unrealistic adaptation
    - Hybrid CCD and analytical method on a hierarchy of groups
      - Fast
      - Realistic results
      - Can preserve balance
      - Manage prioritized and weighted constraints
      - Easy to implement
      - Cannot handle all the biomechanical laws



# Conclusion

- Choice of IK method depends on several parameters:
  - Speed (interactive applications vs. simulations)
  - Natural result (humans vs. robots)
  - Complexity of the implementation



# Conclusion

- Comparison of the two key approaches able to manage prioritized constraints

| Method | Performance | Realism | COM handling | Biomechanical laws |
|---|---|---|---|---|
| J Pseudo-inverse | + | +++ | +++ | +++ |
| CCD + analytical | +++ | +++ | +++ | + |

Analytical not presented since it is limited to 7DOF



# Conclusion

- Previous IK methods can also be used for motion adaptations
  - Directly for small adaptations
  - Using displacement maps for large adaptations
    [Bruderlin95, Gleicher97, Gleicher01, Le Callennec04]