Joint-Torque Control of Character Motions

Active Animations

Ben Kenwright

Abstract—We want to go beyond “passive rag-doll like” simulated characters towards more “active” intelligent self-driven solutions. The “puppet on strings” approach lacks dynamic interactive properties for engaging realistic and immersive virtual environments. This paper focuses on physics-based “self-driven characters” (e.g., character’s own joint-torques to control and move the limbs to accomplish specific tasks, such as walking) that can react in a life-like manner using physical properties (e.g., limb-strength, ground contacts, and mass). We start by explaining the kinematic principles of motion-control of articulated figures. The two types of motion-control for virtual characters are ‘position-based’ and ‘torque-based’. The position-based approach focuses on the linear inverse kinematic model, while the force-based approach extends the principle to generalized joint-torques. Primarily, the relationship between end-effectors (e.g., fingers and feet) current and desired targets to joint orientations and torques. We exploit the highly complex and ambiguous nature of an articulated character to produce primary and secondary cases. For example, projecting joint-torques onto the null-space transformation. This discusses and explains the principles and advantages of active torque controlled character animation solutions, including their dynamics and limitations.

Index Terms—joints, torque, force, motion, procedural, ragdoll, puppet, interactive, balancing, character, animation, physics-based, responsive, adaptive, dynamic, 3D, video-games, inverted-pendulum, joint-torque, controllable

The human-muscle is analogous to a spring-damper system [7]. We can calculate specific gain and damping coefficients to mimic specific muscle responses while actively controlling joint-torque movement by means of current and desired angular error.

1.1 Control

One specific problem for virtual characters is the kinematic issues associated with control of the final motions (i.e., task specific movements, such as placing the foot at a precise location at a certain time while following a specified trajectory).

For physics-based character solutions, uncomplicated angular-springs are used (i.e., active-joint torque servos), which affect the orientation and angular velocity of the character’s body. Determining a specific set of joint torques and trajectories as inputs to the angular-spring servos to solve the specified task is a crucial issue.

1.2 Inverse Kinematic Solution

Inverse kinematics allows us to analyze, design, simulate, and plan virtual character movements. However, due to the complex nature of human based virtual characters it can be computationally expensive and complex to accomplish in real-time systems, such as games. For instance, a common situation that requires inverse kinematics regularly in character based games is the adaptation of pre-canned animations to accommodate certain tasks to appear more realistic (e.g., feet striking the ground and not slipping and arms touching object they are holding).

As a side note, while we focus on the global iterative approach (i.e., the Jacobian technique), there are, however, other techniques, such as, the cyclic coordinate descent (CCD) technique [5], that can solve IK problems faster and are more viable for real-time environments. The CCD technique is one such popular alternative, that works in a heuristic manner to provide a computationally fast, algorithmically simple, and straight-forward technique for generating IK solutions for interactive systems (e.g., gaming environments).
Realism is particularly difficult, as a particular character model gives rise to a large set of possible motions with different styles. Even if robust and stabilizing control laws can be found, it is challenging to construct those that reproduce the intricate and agile movements we observe in nature.

Then there is model complexity, since a character can have an extremely high number of degrees of freedom, making the search for the appropriate control parameters hard (e.g., adult human body has over 200 hundred bones). Although continuous numerical optimizations can cope with large search spaces, the stringent demands of interactive applications make it clear that optimization cannot solely be performed at the time control is needed.

The discontinuous non-linear character work-space (e.g., joint limits and contacts) restricts movement within a certain region of three-dimensional space; these constraints are difficult to maintain in real-time simulation systems, such as games. Furthermore, frequent ground contacts create a highly discontinuous search space rendering most continuous controller synthesis methods ineffective at planning over longer time horizons.

Dynamically simulated characters are difficult to control because they have no direct control over their global position and orientation (i.e., underactuation). Even staying upright is a challenge for large disturbances. In order to succeed, a control law must plan ahead to determine actions that can stabilize the body [15].

2 Related Work

How have these challenging problems of generating life-like interactive characters been solved before? Which methods have made us ‘sit-up’ and take notice and why? This is what we aim to address in this section.

To begin with, there are, the manually designed physics-based biped balance controllers [25], [20], which can be optimized for robust behaviors [23], [1], or combined with other techniques (e.g., motion capture data) to produce hybrid solutions with life-like dynamic responses [4]. While controller-based approaches are often intuitive and computationally fast and robust, they can be cumbersome to tune (i.e., the different parameters) and produce only simple motions (e.g., balanced standing and walking). Then again, solutions to these problems have been proposed, such as an automatic search-based algorithms to tune controller parameters automatically [22] based on velocity tracking of motion capture data.

Data-driven solutions are able to adapt character poses so they transition seamlessly [17], mix animation sequences [24] or modify sequences so they account for changing environments and disturbances (e.g., pushes) [21]. However, data-drive methods cannot generate unique motions, and highly depend on the input motion capture data.

This paper adopts a ‘non data-driven’ technique with a controller-based solution, that is a procedural physics-based controller approach. Where the physics-based model ensures the movement is physically plausible, the controller gives the model a goal (e.g., balanced stepping, walking, steering), and the procedural aspect provides intermediate transition solutions, such as behavioral aspects that include how the character is walking, standing, and looking around.

It is interesting to note, that if we take a look at biomechanical research, in the area of balance, there is a reaction time for humans responding loss of balance [14], [13], [16].
3. Position/Orientation Kinematic Control

3.1 Forward Kinematic

A virtual human biped character has a large number of degrees-of-freedom, and the inverse kinematic problem being computationally complex. The problem is solved by linearizing the kinematic problem for small changes. This enables us to form a relationship between a change in joint-angles and the associated overall body end-effectors (e.g., feet and hands), as given below in Equation 1:

\[
\Delta x = J(q)\Delta q
\]

(1)

where \(\Delta x\) represent a change in end-effector positions (and/or orientations), \(\Delta q\) represents the associated change in orientation of the joints, and \(J(q)\) is the Jacobian matrix. For an \(n\)-degree-of-freedom articulated character with \(n\) end-effector operating in an \(m\)-dimensional space, \(J(q)\) is an \(n \times m\) matrix. For example, in 2D, if we have an end-effector with just position (i.e., x-y movement) 2-dof, while the interconnected body is composed of a singly-linked chain of connected elements (e.g., three-links), the Jacobian dimensions would be \([3 \times 2]\), as shown in Figure 3.

3.2 Inverse Kinematic

For small changes Equation 1 forms a linear relationship between changes in end-effector position and joint-angles. We can calculate the skeleton posture and motion control solution by re-arranging the equation (i.e., calculating the inverse). The inverse of Equation 1 produces a linear relationship, given below in Equation 2.

\[
\Delta q = J^{-1}(q)\Delta x
\]

(2)

The trajectories for continuous movements of the articulated joints is achieved by incrementally moving the current target end-effector position to specific locations (i.e., \(q + \Delta q\)).

Note, in a non-redundant system, \(n = m\), however, this is really never the case.

3.2.1 Exploiting Inverse Kinematic Ambiguity and Redundancy

The highly complex nature of articulated characters can produce an infinite number of solutions to specific inverse kinematic problems. We can, however, use this to our advantage to solve multiple problems with varying priority. This can be accomplished by means of the “pseudo-inverse”, which can be calculated using a general inverse algorithm, as shown below in Equation 3.

\[
\Delta q = [I - J^+(q)J(q)]\Delta q_g
\]

(3)

where \(I\) is an identity matrix with the same dimensions as \(J(q)\), \(q_g\) represents a secondary goal arbitrary vector with the dimensions the same as \(q\). The matrix \([I - J(q)^+J(q)]\) defines the “null-space” region associated with the “pseudo-matrix” \(J^+(q)\), while \([I - J^+(q)J(q)]\Delta q_g\) represents the “zero-deviation” of position and orientation for the end-effector target. The additional secondary null-space calculation allows us to prioritize motions based on some “minimizing criteria” (e.g., obstacle avoidance, balance, or comfort).

4 Force/Torque-Based Kinematic Control

The inverse kinematic problem is represented by a set of linear equations that we solve using computationally straightforward linear system techniques (e.g., general matrix inverse approaches). We can, however, extend the approach to encapsulate torque-based control problems. Thereby, end-effector positions/orientations are replaced with forces/torques, and the generalized system of connected joint-angles is replaced by a set of joint-torques. The basic relationship between end-effector forces and joint-torques is given below in Equation 4.

\[
T = J^T(q)F
\]

(4)
4.1 Exploiting Ambiguity and Redundancy

Similar to Section 3.2.1 that exploited redundancy for multiple inverse kinematic problems to achieve a priority-based solutions, we can accomplish the same goal with torque-based inverse kinematic system. For articulated systems where $m > n$ we can have an infinite number of displacement solutions. We can, however, use the redundant null-space associated with the Jacobian inverse to ensure the end-effector goal is met, while secondary goals are accomplished. Hence, an infinite number of joint-torque deviations can be applied without effecting the end-effector forces/torques. These secondary joint-torques act along the directions within the null-space.

The virtual joint displacement is:

$$\Delta q = J^+ (q) \Delta x + [I - J^+ (q) J(q)] \Delta q_g$$  \hspace{1cm} (5)

Knowing that the virtual work displacement:

$$\Delta W = \Delta W_0 + \Delta W_1$$ \hspace{1cm} (6)

with

$$\Delta W_0 = [J^+^T (q) T]^T \Delta x$$ \hspace{1cm} (7)

and

$$\Delta W_1 = ([I - J^+ (q) J(q)]^T T)^T \Delta q_g$$ \hspace{1cm} (8)

where $W_0$ is the end-effector virtual work, and $W_1$ is the null-space virtual work for the joint-torque displacement associated with the pseudo-inverse $J^+ (q)$. Hence, the relationship between end-effector forces and joint-torques is shown below in Equation 9.

$$T = J^T (q) F + [I - J^T (q) J + J^T (q) J(q)] T_g$$ \hspace{1cm} (9)

where $T_g$ is an arbitrary set of joint-torques.

While a vector $F$ controls the end-effector goal, the joint-torque vector $T_g$ controls secondary internal joint motions. We can, for example, make $T_g$ the gradient potential with its minimum at the desired postural position.

5 CONCLUSION AND DISCUSSION

Multiple factors, such as computational speed, naturalness, realism, and robustness all contribute towards what makes an animation system a viable solution. Creating an animation system that tackles each of these challenges is difficult and important.

ACKNOWLEDGMENTS

A special thanks to readers for taking the time to contact me and provide insightful, invaluable comments and suggestions to help improve the quality of this article.

REFERENCES

SIGGRAPH Asia 2009 papers, SIGGRAPH Asia ’09, pages 170:1–170:9, New York, NY, USA, 2009. ACM. 1, 2


