

Федеральное государственное автономное образовательное учреждение  
высшего образования  
«Московский физико-технический институт  
(национальный исследовательский университет)»  
Физтех-школа Прикладной Математики и Информатики  
Центр обучения проектированию и разработке игр

**Направление подготовки / специальность:** 09.03.01 Информатика и вычислительная техника

**Направленность (профиль) подготовки:** Компьютерные науки и инженерия

## УЛУЧШЕНИЕ ГЕНЕРАЦИИ ДВИЖЕНИЯ РЭГДОЛА

(бакалаврская работа)

**Студент:**

Москалев Никита Евгеньевич

*(подпись студента)*

**Научный руководитель:**

Козак Роман Александрович

*(подпись научного руководителя)*

Москва 2024

## **Аннотация**

# Содержание

<b>1 Введение</b>	<b>4</b>
1.1 Обзор литературы . . . . .	6
1.1.1 Dynamic Postural Adjustment with QP Method . . . . .	6
1.1.2 Multiobjective Control with Frictional Contacts . . . . .	7
1.1.3 Momentum Control for Balance . . . . .	7
1.2 Модель персонажа . . . . .	8
1.2.1 Кинематическое дерево . . . . .	8
1.2.2 Обобщенные координаты . . . . .	9
1.2.3 Уравнение динамики . . . . .	11
1.2.4 Центроидальная матрица . . . . .	12
1.2.5 Импульс и момент импульса . . . . .	12
<b>2 Система управления персонажем</b>	<b>14</b>
2.1 Оптимизация . . . . .	15
2.1.1 Следование опорной анимацией . . . . .	17
2.1.2 Контроль положения центра масс . . . . .	18
2.1.3 Контроль положения центра давления . . . . .	20
2.1.4 Положением точек контакта с поверхностью . . . . .	21
2.1.5 Направление силы реакции опры . . . . .	22
2.2 Прямая динамика . . . . .	22
<b>3 Результаты</b>	<b>23</b>

# 1. Введение

Одна из основных задач, стоящих перед разработчиками и исследователями в области анимации, – это воспроизведение разнообразия движений человека, с целью повышения реализма персонажей и их взаимодействий с игровым миром. Несмотря на сложности, связанные с недостаточным пониманием всех тонкостей движения человека, множество подходов к решению этой задачи было предложено за последние десятилетия, многие из которых в последствии были применены в разработке видео игр.

Существующие подходы можно разделить на две группы: основанные на данных [1], [2], [3] и основанные на физическом моделировании [4], [5], [6]. Такое разделение позволяет выделить ключевые идеи лежащие в основе многих из них, но не является исчерпывающим, поскольку существуют, например, гибридные подходы.

Подходы, составляющие первую группу, для создания анимаций персонажа используют большое количество предварительно записанных движений человека. Интенсивное развитие таких подходов началось после появления технологий оцифровки движений реальных объектов, таких как Motion Capture, позволивших существенно упростить сбор необходимых данных. Один из интересных примеров в этой группе – это Motion Matching [7]. В нем анимация персонажа выбирается среди всех доступных каждый кадр. Критерии выбора могут быть разные – вид движения, стилистика, направление. В сочетании с большой анимационной базой Motion Matching демонстрирует отличные результаты. Такие результаты обусловлены использованием данных, полученных из реального мира. Тем не менее это имеет и свои недостатки. В случае когда требуемое движение не стандартно или не может быть записано подходы основанные на данных плохо применимы.

Подходы, составляющие вторую группу, приводят персонажа в движение через управление динамикой его физической модели. В таком случае обеспе-

чивается физическая корректность движения. Кроме того, при дополнительном моделировании окружающей персонажа среды появляется возможность обрабатывать взаимодействия с ней. К сожалению, из-за сложности разработки правил управления создание таких подходов является проблемой, также как и их использование, которое требует больших вычислительных ресурсов. Однако с углублением понимания механики движения человека и ростом производительности процессоров значимость этих недостатков будет уменьшаться. Поэтому генерация движения персонажа на основе физического моделирования остается перспективным направлением исследования.

Распространенная задача, возникающая при разработке систем управления персонажа, – это необходимость поддержания баланса. Поддержание баланса подразумевает предотвращение неконтролируемого падения персонажа. Такая необходимость может возникнуть даже в самом простом случае, когда персонаж просто следует опорной анимации. А особенно ярко она проявляется, когда персонаж находится в присутствии внешних возмущений или на неровной поверхности. Кроме того, задача сильно варьирует в зависимости от вида воспроизведенного движения. Выделяют две категории баланса: статический, для движений на месте, и динамический.

В данной работе предлагается, реализуется и анализируется система управления персонажа, которая способна одновременно поддерживать баланс и следовать опорной анимации. Поддержание баланса основано на контроле положений центра масс и центра давления. В каждый момент времени решается задача квадратичного программирования, которая оптимизирует значения целевых функций, отвечающих за контроль положений центра масс и центра давления и следование опорной анимации. После чего персонаж приводится в движение в с результатами оптимизации. Более того, при изменение целевых функций описываемый способ может быть адаптирован для других задач, возникающих при разработке систем управления персонажем.

## 1.1. Обзор литературы

В данном разделе рассматриваются несколько работ, идеи из которых были заимствованы при разработке предлагаемой системы управления персонажем. Так же в таблице 1 приведен их краткий сравнительный анализ.

### 1.1.1. Dynamic Postural Adjustment with QP Method

Похожая задача поддержания баланса возникает и в другой области – робототехнике. Причем в отличие от анимаций, где целью является увеличение реалистичности и интерактивности персонажей, в робототехнике поддержание баланса робота критически важно для его успешного функционирования в реальных условиях. Из-за сходства методов физического моделирования персонажей и роботов, идеи, предложенные в области робототехники, могут быть применены в анимации, и наоборот.

Одно из первых применений контроля положений центра масс и центра давления было сделано в [8]. В этой работе описывается алгоритм восстановления сбалансированного положения робота, после воздействия на него внешних возмущений. Основу алгоритма составляет вычисление обобщенных ускорений, таких чтобы центр масс возвращался в исходное положение, а центр давления, находился внутри допустимой области. Вычисление осуществляется применением одного из двух способов: оптимизации или пропорционально-дифференцирующего регулятора. Способ выбирается в зависимости от удаленности положения центра масс от исходного. Алгоритм демонстрирует движения робота схожие с теми, что выполняет человек для восстановления баланса. В дальнейшем алгоритм был доработан в [5] уже применительно к анимации.

### **1.1.2. Multiobjective Control with Frictional Contacts**

В [4] описывается система управления персонажем способная одновременно обеспечивать контроль положения центра масс и следование персонажа опорной анимации. В таком случае основную сложность составляет одновременная работа с несколькими целями движения, поскольку опорная анимация часто не является сбалансированной. Возможность учитывать несколько конфликтующих целей движения была получена применением оптимизации, целевая функция которой содержала вклад от каждой цели движения. Не смотря на то что работа фокусировалась только на контроле положения центра масс, что недостаточно для поддержания баланса во всех ситуациях, идея комбинирования целей движения открывает множество возможностей для дальнейшего улучшения.

### **1.1.3. Momentum Control for Balance**

В [5] описывается система управления персонажем, которая контролирует положения центра масс и центра давления, а также обеспечивает следование персонажа опорной анимации. Работа с несколькими целями движения реализуется с помощью оптимизации, во время которой вычисляются оптимальные для текущей ситуации обобщенные ускорения. Полученные данные передаются в алгоритм обратной динамики, результаты которого вместе с внешними возмущениями используются алгоритмом прямой динамики для приведения персонажа в движение. Последние две стадии используются для того, чтобы интегрировать в систему внешние возмущения аналогично алгоритму, описанному в [8].

Отличительной особенностью системы, представленной в [5], – это использование импульса и момента импульса для контроля положений центра масс и центра давления. Такой способ также будет использован в данной работе.

	Кон- троль положе- ния ЦМ	Кон- троль положе- ния ЦД	Следо- вание опорной анимации
Dynamic Postural Adjustment with QP Method	Да	Нет	Нет
Multiobjective Control with Frictional Contacts	Да	Нет	Да
Momentum Control for Balance	Да	Да	Да

Таблица 1: Сравнение рассмотренных работ

## 1.2. Модель персонажа

В данном разделе описывается физическая модель персонажа и формулируются уравнения, описывающие ее.

### 1.2.1. Кинематическое дерево

Кинематическое дерево – это система из  $n$  твердых тел, соединенных между собой  $m$  шарнирами. Каждый из шарниров, кроме корневого, ограничивает относительное движение тел, которые он соединяет. Например, призматический шарнир оставляет только поступательное движение вдоль выбранной оси. Корневой шарнир, в свою очередь, определяет возможность системы перемещаться в пространстве и бывает двух видов: плавающий, то есть не накладывающий ограничений, и фиксирующий.

Для того чтобы приводить в движение отдельные тела, некоторые шарниры могут быть снабжены приводами, которые генерируют необходимые силы и моменты сил. В таком случае шарниры называются активными, иначе,

соответственно, неактивными. Отметим, что корневой шарнир обычно остается неактивным. Таким образом за движение системы как целого отвечает сила трения. Это сохраняет физическую корректность, но сильно усложняет управление кинематическим деревом.

Работа системы управления такой моделью сводится к вычислению сил и моментов сил, которые должны генерировать приводы, чтобы получить движение удовлетворяющие заявленным требованиям, и последующему их применению, для воспроизведения движения.

В данной работе персонаж моделируется как кинематическое дерево, все шарниры которого имеют привод, причем силы и моменты сил, генерируемые в корневом шарнире, минимизируются во время оптимизации. Результаты показывают, что такая модель в большинстве ситуаций эквивалентна кинематическому дереву с неактивным корневым шарниром. На рисунках 1 и 2 изображен пример трехмерной модели персонажа и соответствующего кинематического дерева.

Отметим, что представление персонажа в виде кинематического дерева также называют активный рэгдол.

### **1.2.2. Обобщенные координаты**

При работе с кинематическим дерево важную роль играет способ, выбранный для описания положения и ориентации тел в пространстве, поскольку он во многом определят простоту, устойчивость и вычислительную сложность моделирования. Основные способы – это максимальные координаты и обобщенные координаты.

Максимальные координаты описывают тела по отдельности, используя по 6 чисел на каждое, а ограничения, накладываемые шарнирами, учитывают при решении уравнения динамики. Такой способ позволяет использовать существующие системы физического моделирования, но страдает от ошиб-

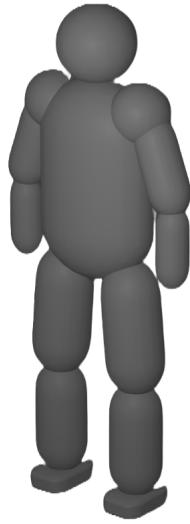


Рисунок 1: Трехмерная модель

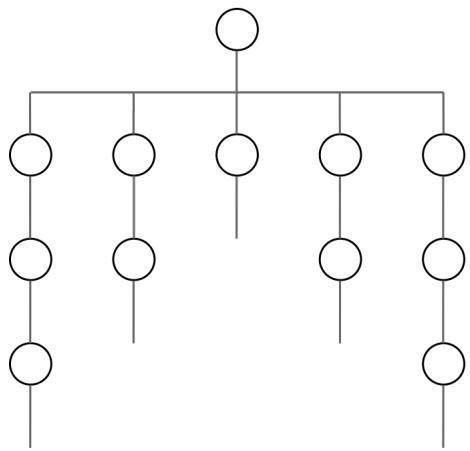


Рисунок 2: Кинематическое дерево.  
Круги обозначают шарниры, а соединения между кругами – твердые  
тела

бок работы с вещественными числами, которые приводят к тому, что тела открепляются друг от друга.

Обобщенные координаты, напротив, учитывают связи между телами. Например, для кинематического дерева, состоящего из двух тел, соединенных вращательным шарниром, используется 7 чисел, первые 6 из которых описывают положение и ориентацию одного из тел, а оставшееся – угол поворота вокруг оси шарнира. Такой способ минимизирует количество используемых чисел и неявно учитывает ограничения.

В данной работе используются обобщенные координаты, скорости и ускорения. Отметим, что обобщенные скорости позволяют выразить скорость любого шарнира. Для этого необходимо умножить их на якобиан шарнира. После чего можно получить скорость любой точки твердого тела, прикрепленного к шарниру, зная ее положение.

### 1.2.3. Уравнение динамики

Уравнение, связывающее обобщенные ускорения и силы и моменты сил, генерируемые приводами, называется уравнением динамики кинематического дерева и имеет вид

$$H\ddot{q} + C(q, \dot{q}) + G(q) = u + J^T f, \quad (1.1)$$

где  $q, \dot{q}, \ddot{q}$  – обобщенные координаты, скорости и ускорения,  $H$  – матрица инерции,  $C$  – центробежная и кориолисова силы,  $G$  – сила тяготения,  $u$  – силы и моменты сил, генерируемые приводами,  $J$  – якобиан, и  $f$  – внешние силы. Вывод этого уравнения из принципа наименьшего действия описан в [9].

В качестве неизвестной в уравнении 1.1 может выступать  $\ddot{q}$  или  $u$ . Алгоритмы, которые находят  $\ddot{q}$ , называются алгоритмами прямой динамики, а те, которые находят  $u$ , – обратной. Эффективные и повсеместно используемые реализации этих алгоритмов представлены в [9].

#### 1.2.4. Центроидальная матрица

В [10] показана связь импульса и момента импульса кинематического дерева, выраженных в неподвижной системе отсчета, расположенной в центре масс, с обобщенными скоростями, имеющая следующий вид

$$\begin{bmatrix} P \\ L \end{bmatrix} = A\dot{q}, \quad (1.2)$$

где  $P$  – импульс,  $L$  – момент импульса, а  $A$  – это центроидальная матрица.

Центроидальная матрица, как и матрицей инерции, является фундаментальной характеристикой кинематического дерева, которая зависит только от массы, инерции и значения обобщенных координат твердых тел, составляющих его.

При разделении центроидальной матрицы на две уравнение 1.2 принимает вид

$$P = A_P\dot{q}, \quad (1.3)$$

$$L = A_L\dot{q}. \quad (1.4)$$

При дифференцировании уравнения 1.3 и 1.4 принимают вид

$$\dot{P} = \dot{A}_P\dot{q} + A_P\ddot{q}, \quad (1.5)$$

$$\dot{L} = \dot{A}_L\dot{q} + A_P\ddot{q}. \quad (1.6)$$

Полученные уравнения 1.5 и 1.6 отражают связь между значениями производных импульса и момента импульса и обобщенными ускорениями. В данной работе они используются при формулировании целевой функции оптимизации.

#### 1.2.5. Импульс и момент импульса

Положения импульса и момента импульса показывают устойчивость персонажа, поэтому одна из задач разрабатываемой системы управления – это

их контроль. Способ используемый в данной работе управляет импульсом и моментом импульса, таким образом контролируя положения центра масс и центра давления. В данном подразделе показана связь между этими величинами, которая обосновывает корректность выбранного способа.

Напомним, что центр давления – это точка, где можно приложить результирующую силу нормальной реакции опоры и силу трения (далее – силу реакции опоры), так чтобы момент относительно центра масс не изменился.

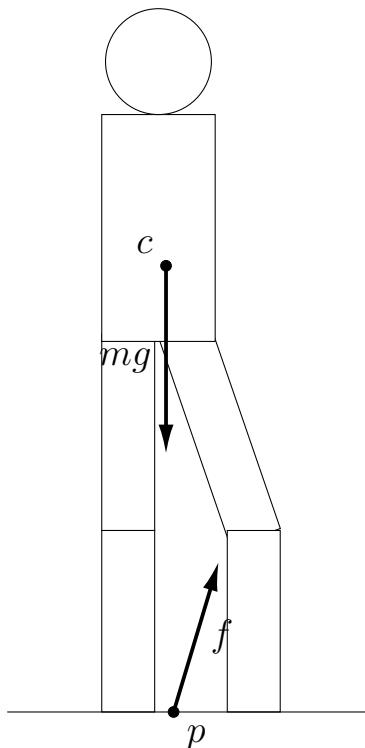


Рисунок 3: Силы

Рассмотрим силы, действующие на кинематическое дерево (рисунок 3). Запишем второй закон Ньютона и основное уравнение вращательной динамики

$$\dot{P} = mg + f, \quad (1.7)$$

$$\dot{L} = (p - c) \times f, \quad (1.8)$$

где  $c$  – центр масс,  $p$  – центр давления,  $m$  – суммарная масса, и  $f$  – результирующая силы нормальной реакции опоры и силы трения.

Исключая  $f$  из уравнений 1.7 и 1.8, получим

$$\dot{L} = (p - c) \times (\dot{P} - mg). \quad (1.9)$$

Полученное уравнение 1.9 показывает, что при известном импульсе и положении центра масс, контроль положения центра давления может быть сделан с помощью управления момента импульса.

Теперь, запишем определение импульса кинематического дерева, рассматривая его как систему твердых тел

$$P = \sum_{i=1}^n m_i \dot{x}_i,$$

где  $x_i, m_i$  – положения и массы. Используя следующую цепочку равенств

$$\sum_{i=1}^n m_i \dot{x}_i = \frac{d}{dt} \left( \sum m_i x_i \right) = \frac{d}{dt} (mc) = m\dot{c},$$

преобразуем уравнение к виду

$$P = m\dot{c}, \quad (1.10)$$

а дифференцированием к виду

$$\dot{P} = m\ddot{c}. \quad (1.11)$$

Полученное уравнения 1.11 показывает, что контроль ускорения центра масс может быть сделан с помощью управления производной импульса.

## 2. Система управления персонажем

Работа системы управления персонажем состоит из следующих стадий, повторяемых в цикле:

1. оптимизация;
2. прямая динамика;

### 3. интегрирование.

Во время оптимизации вычисляются силы и моментов сил, которые должны генерировать приводы, и силы реакции опоры, такие, чтобы получаемое движение было сбалансированным и следовало опорной анимации. Результаты оптимизации передаются в алгоритм прямой динамики, который находит ускорения. Далее скорости и ускорения интегрируются по времени-ному шагу, с целью получения новых скоростей и обновления положения персонажа.

Внешние возмущения, если пресутсвуют, также передаются в алгоритм прямой динамики. Ошибки, которые они вносят в движение, компенсируются оптимизатором на следующих циклах работы системы.

На рисунке 4 схематично изображена работа системы.

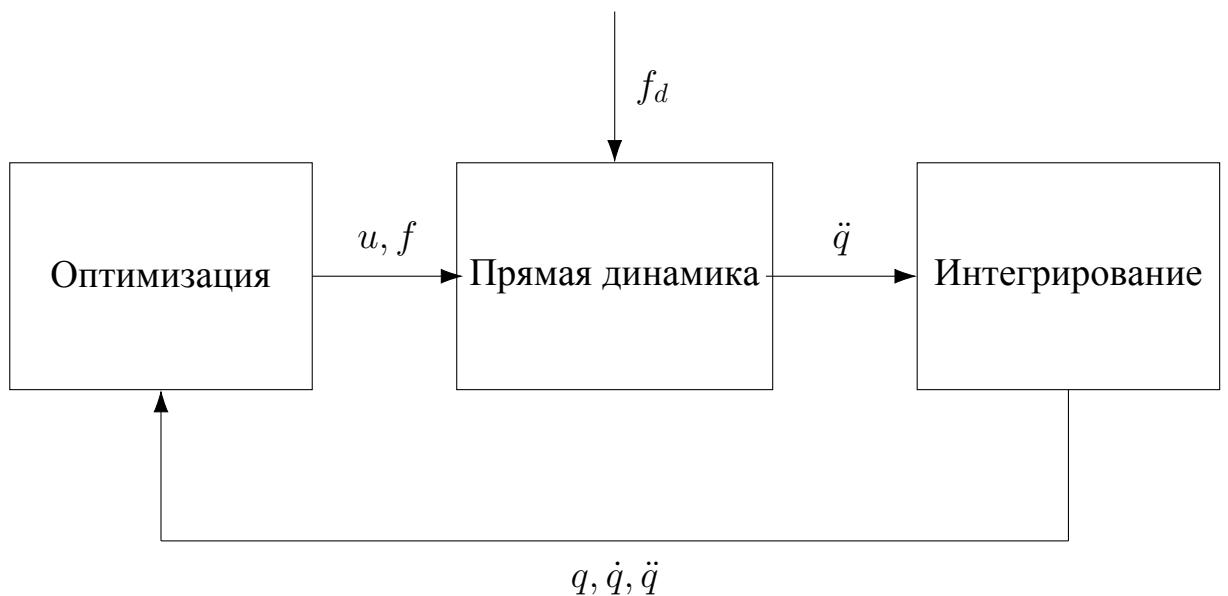


Рисунок 4: Работа системы управления

## 2.1. Оптимизация

Оптимизация – это ключевая стадия работы системы управления, ответственная за вычисление сил и моментов сил, которые должны генерировать

приводы. Оптимизация выбрана поскольку она позволяет учесть все условия, влияющих на значения сил и моментов сил, в виде целевой функции или ограничений.

Задача оптимизации имеет следующий вид

$$\min_{\ddot{q}, u, f} \omega_t h_t(\ddot{q}) + \omega_c h_c(\ddot{q}) + \omega_p h_p(\ddot{q}) + \omega u^T \begin{bmatrix} I_6 & O \\ O & O \end{bmatrix} u$$

$$s.t. H\ddot{q} + C(q, \dot{q}) + G(q) = u + J^T f \quad (2.1)$$

$$J_{sup}\ddot{q} + \dot{J}_{sup}\dot{q} = 0 \quad (2.2)$$

$$0 \leq n(f) \quad (2.3)$$

$$\tau(f) \leq \mu n(f) \quad (2.4)$$

Целевая функция оптимизации состоит из четырех слагаемых. Первые три из них –  $h_t$ ,  $h_c$  и  $h_p$  – отвечают за следование опорной анимации, за положения центра масс и за положения центра давления соответственно. Последнее стремиться минимизировать силы и моменты сил, генерируемые в корневом шарнире. Веса  $w_t$ ,  $w_c$ ,  $w_p$  и  $w$  – позволяют регулировать значимость вклада каждого из слагаемых.

Отметим, что последнее слагаемое важно для надежности и простоты настройки системы, поскольку оно обеспечивает обработку ситуаций, когда значения функций  $h_t$ ,  $h_c$  и  $h_p$  значительно отклоняются от оптимальных и не могут быть восстановлены без генерации сил и моментов сил в корневом шарнире. Обычно вместо последнего слагаемого используется ограничение, требующее равенства сил и моментов сил в корневом шарнире нулю. Однако тогда система не может обработать ситуации выше. Применительно к анимациям в видео играх выбор в пользу надежности за счет возможной потери физической корректности может оказаться целесообразным.

Уравнение 2.1 согласует результаты оптимизации с уравнением динамики кинематического дерева. Таким образом, обобщенные ускорения, полученные применением алгоритма прямой динамики, в случае отсутствия внеш-

них возмущений будут совпадать с теми, что найдены во время оптимизации. Уравнение 2.2 оставляет точки контакта с поверхностью неподвижными. Неравенства 2.3 и 2.4 отвечают за направление силы реакции опоры.

Функции  $h_t$ ,  $h_c$  и  $h_p$  формулируются так, чтобы получившаяся задача оптимизации была квадратичной. Квадратичное программирование успешно применялось в других системах [4], [5], [11], поскольку оно дает предсказуемое время работы оптимизатора и воспроизводимые результаты.

Следующие подразделы посвящены функциям  $h_t$ ,  $h_c$  и  $h_p$ . Поскольку кинематическая характеристика, которую можно контролировать, – это обобщенные ускорения, функции формулируются следующим образом: описывается желаемое ускорение, после чего определяется функция, оптимизация которой минимизирует отклонение получаемого ускорения от желаемого.

### 2.1.1. Следование опорной анимацией

Функция  $h_t(\ddot{q})$  ответственна за поддержание стилистической составляющей движения, задаваемой с помощью опорной анимации. Основная задача этой функции в том, чтобы получаемое движение как можно точнее приближало опорную анимацию. Наивный способ добиться этого – это взять  $h_t(\ddot{q})$  такой, чтобы в результате оптимизации полученное ускорение совпадало с ускорением из опорной анимации. Однако, в таком случае, если получаемое движение отклоняется от опорной анимации, например, в следствие внешних возмущений, то оно не будет скорректировано обратно. Поэтому функция  $h_t(\ddot{q})$  включает слагаемые, которые характеризуют отставание, и определяется следующим образом

$$h_t(\ddot{q}) = \|W(\ddot{q} + s_t(q - q_{ref}) + d_t(\dot{q} - \dot{q}_{ref}) - \ddot{q}_{ref})\|_2^2,$$

где  $W$  – диагональная матрица весов,  $s_t$  и  $d_t$  – коэффициенты, а  $q_{ref}$ ,  $\dot{q}_{ref}$  и  $\ddot{q}_{ref}$  – это положение, скорость и ускорение взятые из опорной анимации. Матрица  $W$  позволяет более точно настроить систему, сохранить требуемые

особенностей опорной анимации. В данной работе матрица  $W$  единичная.

Выражение  $s_t(q - q_{ref}) + d_t(\dot{q} - \dot{q}_{ref})$  использованное в определении  $h_t$  называется пропорционально-дифференцирующий регулятор (далее – ПД регулятор), а коэффициенты  $s_t$  и  $d_t$  – пропорциональный и дифференциальный соответственно. Одно из первых применений ПД регулятора для решения задачи следования физически моделируемого персонажа опорной анимации сделано в [12]. Сейчас ПД регулятор – это повсеместно используемая техника при разработке систем управления персонажем.

Отметим, что выбор коэффициентов  $s_t$  и  $d_t$  может существенно сказаться на результатах применения ПД регулятора. В данной работе коэффициенты удовлетворяют соотношению  $d_t = \sqrt{s_t}$ .

### 2.1.2. Контроль положения центра масс

Положение центра масс является хорошо известным индикатором баланса персонажа [4], [5]. В случае если проекция положения центра масс на поверхность находится вне опорного полигона (рисунок 5), то есть выпуклой оболочки, образованной точками контакта с поверхностью, то персонаж может упасть. Таким образом, стратегия контроля положения центра масс состоит в том, чтобы поддерживать его проекцию на поверхность внутри опорного полигона, причем как можно дальше от границы.

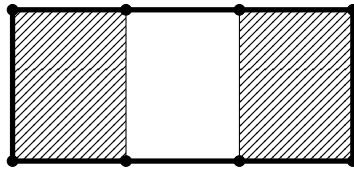


Рисунок 5: Опорный полигон. Заштрихованные прямоугольники обозначают стопы. Точки – это точки контакта с поверхностью. Жирным выделен опорный полигон

Реализовать это можно по разному. Простая и при этом эффективная реализация – это выбор функции  $h_c(\ddot{q})$  такой, чтобы в результате оптимизации

проекция положение центра масс двигалось в направлении центра опорного полигона. Более точно, такой, чтобы ускорение центра масс стремилось к следующему значению

$$\ddot{c}_{des} = -s_c(c - c_{ref}) - d_c(\dot{c} - \dot{c}_{ref}), \quad (2.6)$$

где  $s_c$  и  $d_c$  – коэффициенты, а  $c_{ref}$  и  $\dot{c}_{ref}$  – опорные положение и скорость центра масс. Такая формулировка желаемого ускорения центра масс делает контроль его положения более настраиваемым. В данной работе  $c_{ref}$  выбрано как положение центра опорного полигона, а  $\dot{c}_{ref}$  равно нулю. С другой стороны,  $c_{ref}$  и  $\dot{c}_{ref}$  могут быть взяты из опорной анимации. Отметим, что коэффициенты  $s_c$  и  $d_c$  могут быть константами или значениями, меняющимися в зависимости от состояния персонажа, как это сделано в [4].

Поскольку контроль положения центра масс вдоль гравитационной оси не важен для поддержания проекции центра масс на поверхность внутри опорного полигона, уравнение 2.6 рассматривается только по двум ортогональным гравитационной осям.

Преобразуем уравнение 2.6 к виду необходимому, чтобы сформулировать  $h_c(\dot{q})$ . Умножением обоих частей на массу, уравнение 2.6 принимает вид

$$m\ddot{c}_{des} = -s_c m(c - c_{ref}) - d_c m\dot{c}. \quad (2.7)$$

Подставляя  $m\ddot{c}_{des}$  и  $m\dot{c}$  из уравнений 1.10 и 1.11, уравнение 2.7 принимает вид

$$\dot{P}_{des} = -s_c m(c - c_{ref}) - d_c P. \quad (2.8)$$

Подставляя  $P$  из уравнения 1.3, уравнение 2.8 принимает вид

$$\dot{P}_{des} = -s_c m(c - c_{ref}) - d_c A_P \dot{q}. \quad (2.9)$$

Функцию  $h_c(\ddot{q})$  можно сформулировать как квадратичное отклонения получаемого импульса от желаемого импульса. Поскольку оптимизация происходит по переменным  $\ddot{q}, u, f$  получаемый импульс необходимо выразить

через обобщенные ускорения, используя уравнение 1.5. Таким образом  $h_c(\ddot{q})$  имеет вид

$$h_c(\ddot{q}) = \|A_P \ddot{q} - \dot{A}_P \dot{q} - P_{des}\|_2^2. \quad (2.10)$$

### 2.1.3. Контроль положения центра давления

Положение центра давления является индикатором вращательной устойчивости движения персонажа [5], [13]. В случае если положение центра давления находится вне опорного полигона, то персонаж может опрокинуться. Кроме того, в случае если центр давления находится на границе опорного полигона, может возникнуть вращение опоры, то есть стоп персонажа. Таким образом, стратегия контроля положения центра давления состоит в том чтобы поддерживать его строго внутри опорного полигона.

Реализовать эту можно аналогично тому как сделан контроль положения центра масс. А именно, выбрать функцию  $h_p(\ddot{q})$  такой, чтобы ускорения центра давления стремилось к следующему значению

$$\ddot{p}_{des} = -s_p(p - p_{ref}) - d_p(\dot{p} - \dot{p}_{ref}), \quad (2.11)$$

где  $s_p$  и  $d_p$  – коэффициенты, а  $p_{ref}$  и  $\dot{p}_{ref}$  – опорные положение и скорость центра давления. В данной работе  $d_{ref}$  выбрано как положение центра опорного полигона, а  $\dot{d}_{ref}$  равно нулю.

Преобразуем уравнение 2.11 к виду необходимому, чтобы сформулировать  $h_p(\dot{q})$ . Интегрируя  $\ddot{p}_{des}$  получим значение  $p_{des}$ . Используя уравнение 1.9 получим

$$\dot{L}_{des} = (p_{des} - c) \times (P_{des} - mg). \quad (2.12)$$

Отметим, что при использовании уравнения 1.9 значение импульса взято из уравнения 2.9, для того чтобы контроль положения центра давления был согласован с контролем положения центра масс.

Функцию  $h_p(\ddot{q})$  можно сформулировать как квадратичное отклонения получаемого импульса от желаемого импульса. Поскольку оптимизация про-

исходит по переменным  $\ddot{q}$ ,  $u$ ,  $f$  получаемый импульс необходимо выразить через обобщенные ускорения, используя уравнение 1.5. Таким образом  $h_c(\ddot{q})$  имеет вид

$$h_p(\ddot{q}) = \|A_L \ddot{q} - \dot{A}_L \dot{q} - L_{des}\|_2^2. \quad (2.13)$$

#### 2.1.4. Положением точек контакта с поверхностью

Положение точек контакта персонажа с поверхностью должны оставаться неподвижным, чтобы выполнялось условие нескользящего контакта.

Пусть имеется  $k$  точек контакта с поверхностью пронумерованных от 1 до  $k$ . Соответствующие якобианы, то есть матрицы, отображающие обобщенные скорости на скорости в пространстве, равны  $J_1, \dots, J_k$ . Поэтому скорости точек контакта с поверхностью, обозначаемые  $v_1, \dots, v_k$ , имеют следующий вид

$$\begin{bmatrix} v_1 \\ \vdots \\ v_k \end{bmatrix} = \begin{bmatrix} J_1 \\ \vdots \\ J_k \end{bmatrix} \dot{q}. \quad (2.14)$$

Обозначим столбец скоростей через  $v_{sup}$ , а столбец Якобианов –  $J_{sup}$ . Тогда уравнение 2.14 приобретает вид

$$v_{sup} = J_{sup} \dot{q}. \quad (2.15)$$

Дифференцируя уравнение 2.15, получаем выражение для ускорений  $a_{sup}$  следующего вида

$$a_{sup} = J_{sup} \ddot{q} + \dot{J}_{sup} \dot{q}. \quad (2.16)$$

Для того чтобы скорости точек контакта с поверхностью оставались неподвижными, необходимо поддерживать ускорение равным нулю. Таким образом условие неподвижности точек контакта с опорой имеет следующий вид

$$J_{sup} \ddot{q} + \dot{J}_{sup} \dot{q} = 0. \quad (2.17)$$

### 2.1.5. Направление силы реакции опоры

Пусть функция  $n(f) : \mathbb{R}^3 \rightarrow \mathbb{R}$  возвращает величину проекции  $f$  на нормаль к поверхности, функция  $\tau(f) : \mathbb{R}^3 \rightarrow \mathbb{R}$  возвращает величину проекции  $f$  на поверхность.

Сила нормальной реакции – это результирующая силы нормальной опоры и силы трения, каждая из которых имеет свои ограничениями. Сила нормальной реакции опоры всегда направлена вдоль нормали и не может быть отрицательной. Сила трения не превосходит силу нормальной реакции опоры умноженную на коэффициент трения,  $\mu$ . В терминах функции  $n(f)$  и  $\tau(f)$  эти ограничения формулируются в виде следующих неравенств

$$0 \leq n(f), \quad (2.18)$$

$$\tau(f) \leq \mu n(f). \quad (2.19)$$

## 2.2. Прямая динамика

Прямая динамика – это стадия, во время которой происходит вычисление ускорений, которые вместе со скоростями будут проинтегрированы с целью обновления состояния персонажа. Наличие данной стадии в системе необходимо для возможности восстанавливать баланс в присутствии внешних возмущений. Поскольку величина внешних возмущений еще не известна на стадии оптимизации, она не может быть учтена на этой стадии. Поэтому оптимизация полагается на результаты прямой динамики, и корректирует их в соответствии с требованиями к движению.

Отметим, что если необходимости восстанавливать баланс в присутствии внешних возмущений нет, то можно сразу использовать ускорения, полученные во время оптимизации, поскольку они согласованы с уравнением динамики.

В данной работе используется алгоритм прямой динамики articulated rigid

body, описанный в [9]. Он выбран поскольку обладает наименьшей вычислительной сложность среди алгоритмов прямой динамики.

### 3. Результаты

Описанная система уравнения персонажем была реализована на языке Python. Работа с кинематическим деревом реализована с помощью библиотеки pinocchio [14], [15]. Оптимизация выполняется с помощью библиотеки cvxopt [16]. Данные библиотеки выбраны поскольку они реализуют необходимый функционал наиболее эффективным образом.

Модель персонажа, выбранная для демонстрации работы системы, имеет в общей сложности 38 степеней свободы, образуемых плавающим корневым шарниром и 32 вращательными шарнирами.

Также для демонстрации работы системы была разработана анимация наклона, кадры которой представлены на рисунке 6. Данная анимация не является сбалансированной, так как, например, проекция положения центра масс длительное время находится за пределами опорного полигона. Такая особенность добавлена для более наглядной демонстрации результатов работы системы.

Кадры движения персонажа, полученного в результате работы системы, представлены на рисунке 7. Анимация наклона описанная ранее была использована в качестве опорной. Основной цикл системы выполнялся 120 раз в секунду.

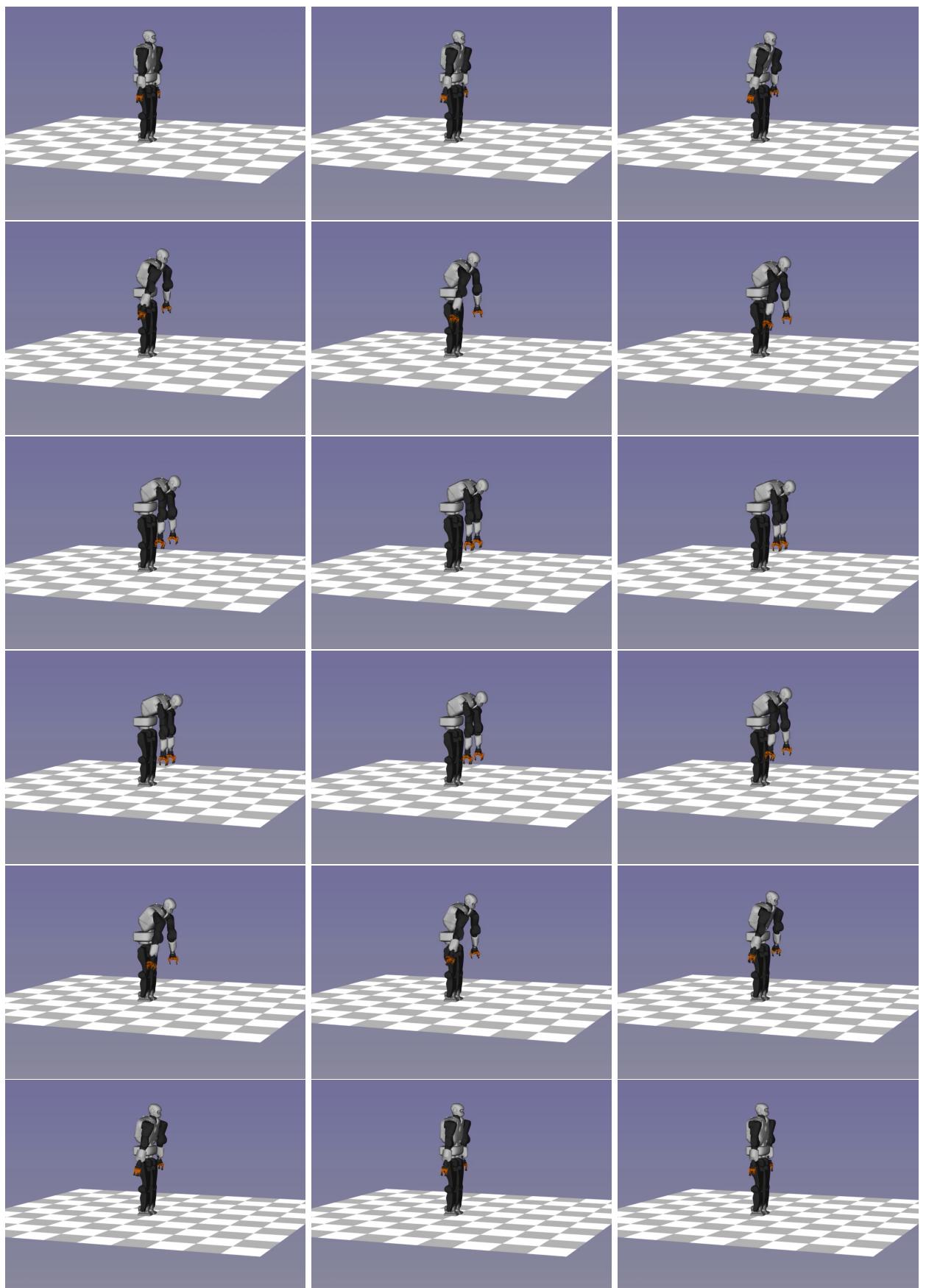


Рисунок 6: Опорная анимация

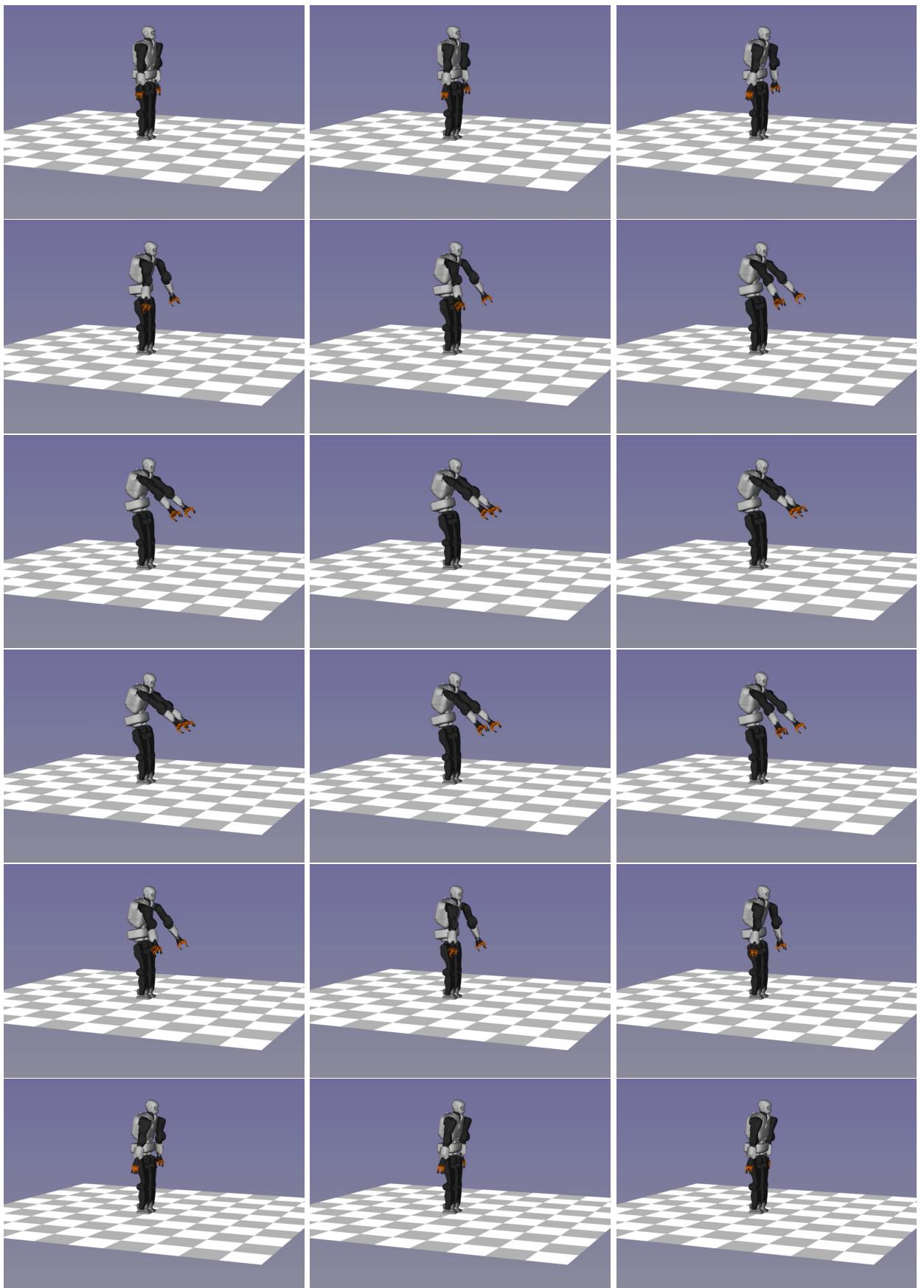


Рисунок 7: Сбалансированная анимация наклона

## Список литературы

1. *Arikan O., Forsyth D. A.* Interactive motion generation from examples // ACM Trans. Graph. — New York, NY, USA, 2002. — Т. 21, № 3. — С. 483—490. — ISSN 0730-0301. — DOI: [10.1145/566654.566606](https://doi.org/10.1145/566654.566606). — URL: <https://doi.org/10.1145/566654.566606>.
2. *Arikan O., Forsyth D. A., O'Brien J. F.* Motion synthesis from annotations // ACM Trans. Graph. — New York, NY, USA, 2003. — Т. 22, № 3. — С. 402—408. — ISSN 0730-0301. — DOI: [10.1145/882262.882284](https://doi.org/10.1145/882262.882284). — URL: <https://doi.org/10.1145/882262.882284>.
3. *Kovar L., Gleicher M., Pighin F.* Motion graphs // ACM Trans. Graph. — New York, NY, USA, 2002. — Т. 21, № 3. — С. 473—482. — ISSN 0730-0301. — DOI: [10.1145/566654.566605](https://doi.org/10.1145/566654.566605). — URL: <https://doi.org/10.1145/566654.566605>.
4. *Abe Y., Silva M. da, Popović J.* Multiobjective control with frictional contacts // Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation. — San Diego, California : Eurographics Association, 2007. — С. 249—258. — (SCA '07). — ISBN 9781595936240.
5. *Macchietto A., Zordan V., Shelton C. R.* Momentum control for balance // ACM SIGGRAPH 2009 Papers. — New Orleans, Louisiana : Association for Computing Machinery, 2009. — (SIGGRAPH '09). — ISBN 9781605587264. — DOI: [10.1145/1576246.1531386](https://doi.org/10.1145/1576246.1531386). — URL: <https://doi.org/10.1145/1576246.1531386>.
6. *Yin K., Loken K., Panne M. van de.* SIMBICON: simple biped locomotion control // ACM Trans. Graph. — New York, NY, USA, 2007. — Т. 26, № 3. — 105—es. — ISSN 0730-0301. — DOI: [10.1145/1276377.1276509](https://doi.org/10.1145/1276377.1276509). — URL: <https://doi.org/10.1145/1276377.1276509>.

7. *Clavet S., Ubisoft Montreal.* Motion Matching and The Road to Next-Gen Animation.—URL: <https://www.gdcvault.com/play/1023280/Motion-Matching-and-The-Road> (дата обр. 26.05.2024).
8. *Kudoh S., Komura T., Ikeuchi K.* The dynamic postural adjustment with the quadratic programming method // IEEE/RSJ International Conference on Intelligent Robots and Systems. T. 3.—2002.—2563—2568 vol.3.—DOI: [10.1109/IRDS.2002.1041656](https://doi.org/10.1109/IRDS.2002.1041656).
9. *Featherstone R.* Rigid Body Dynamics Algorithms. — Berlin, Heidelberg : Springer-Verlag, 2007. — ISBN 0387743146.
10. *Orin D. E., Goswami A.* Centroidal Momentum Matrix of a humanoid robot: Structure and properties // 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems. — 2008. — C. 653—659. — DOI: [10.1109/IROS.2008.4650772](https://doi.org/10.1109/IROS.2008.4650772).
11. *Silva M. da, Abe Y., Popović J.* Interactive simulation of stylized human locomotion // ACM Trans. Graph. — New York, NY, USA, 2008. — T. 27, № 3. — C. 1—10. — ISSN 0730-0301. — DOI: [10.1145/1360612.1360681](https://doi.org/10.1145/1360612.1360681). — URL: <https://doi.org/10.1145/1360612.1360681>.
12. Animating human athletics / J. K. Hodgins [и др.] // Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques. — New York, NY, USA : Association for Computing Machinery, 1995. — C. 71—78. — (SIGGRAPH '95). — ISBN 0897917014. — DOI: [10.1145/218380.218414](https://doi.org/10.1145/218380.218414). — URL: <https://doi.org/10.1145/218380.218414>.
13. *Goswami A., Kallem V.* Rate of change of angular momentum and balance maintenance of biped robots // IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004. — 2004. — DOI: [10.1109/ROBOT.2004.1308858](https://doi.org/10.1109/ROBOT.2004.1308858).

14. The Pinocchio C++ library : A fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives / J. Carpentier [и др.] // 2019 IEEE/SICE International Symposium on System Integration (SII). — 2019. — С. 614—619. — DOI: [10.1109/SII500380](https://doi.org/10.1109/SII500380).
15. Pinocchio: fast forward and inverse dynamics for poly-articulated systems / J. Carpentier, F. Valenza, N. Mansard [и др.]. — URL: <https://github.com/stack-of-tasks/pinocchio> (дата обр. 19.04.2024).
16. *Andersen M., Dahl J., Vandenberghe L.* CVXOPT: Python Software for Convex Optimization. — URL: <https://github.com/cvxopt/cvxopt> (дата обр. 19.04.2024).