УДК 004.942:004.896 https://doi.org/10.33619/2414-2948/116/14

ВИРТУАЛЬНОЕ МОДЕЛИРОВАНИЕ РОБОТА SCARA В UNITY 3D

©Шестаков Е. И., *ORCID: 0000-0003-4237-4898, канд. тех. наук,*
Ошский государственный университет, г. Ош, Кыргызстан, shestakov.e.i@gmail.com
©Пирматов А. З., *ORCID: 0009-0008-2343-5185, SPIN-код: 8965-9182, канд. физ.-мат. наук,*
Ошский государственный университет, г. Ош, Кыргызстан, pirmatov@oshsu.kg
©Клинцов М. А., *Ошский государственный университет,*
г. Ош, Кыргызстан, mihailklintsov2001@gmail.com

VIRTUAL SIMULATION OF A SCARA ROBOT IN UNITY3D

©Shestakov E., *ORCID 0000-0003-4237-4898, Ph.D., Osh State University,*
Osh, Kyrgyzstan, shestakov.e.i@gmail.com
©Pirmatov A., *ORCID: 0009-0008-2343-5185, SPIN-code: 8965-9182, Ph.D.,*
Osh State University, Osh, Kyrgyzstan, pirmatov@oshsu.kg
©Klintsov M., *Osh State University, Osh, Kyrgyzstan, mihailklintsov2001@gmail.com*

Аннотация. Работа посвящена моделированию промышленного робота SCARA в программной среде Unity3D. Рассматриваются этапы подготовки и импорта 3D-модели робота, организация его кинематической структуры с использованием различных подходов, включая Parent-Child, Joints и Articulation Body. Представлены алгоритмы ручного и автоматического управления, основанные на решении прямой и обратной задач кинематики, а также реализация траекторий движения с помощью линейной интерполяции и программирования на основе G-кода. Подчеркивается практическая и образовательная значимость виртуальной модели, её потенциал для использования в обучении и предварительного тестирования алгоритмов управления.

Abstract. This paper presents the development of a virtual model of an industrial SCARA robot using Unity3D. It covers the preparation and importing stages of the robot's 3D model, the organization of its kinematic structure utilizing different approaches, including Parent-Child hierarchy, Joints, and Articulation Body components. Algorithms for manual and automatic control based on forward and inverse kinematics solutions are described, alongside motion trajectory implementation through linear interpolation and G-code programming. The article highlights the practical and educational value of the developed virtual model and its potential for learning and preliminary testing of control algorithms.

Ключевые слова: робот SCARA, виртуальное моделирование, Unity3D, кинематика, прямая задача кинематики, обратная задача кинематики, G-код, траектория движения, роботизация производства, виртуальная среда.

Keywords: SCARA robot, virtual simulation, Unity3D, kinematics, forward kinematics, inverse kinematics, G-code, motion trajectory, industrial robotics, virtual environment.

В последние десятилетия робототехника стремительно развивается, охватывая всё больше областей, от промышленной сборки до медицины и образовательных проектов. Одним из наиболее популярных промышленных решений является робот SCARA (Selective Compliance Assembly Robot Arm), который сочетает в себе ряд преимуществ, таких как

высокая скорость, точность и достаточно простая кинематическая структура, благодаря которой они находят широкое применение в операциях по захвату и перемещению различных объектов, например, сборке промышленных изделий, упаковке и других манипуляционных задачах [1, 3].

При этом одним из важных этапов разработки и внедрения роботизированных систем в производство становится виртуальное моделирование, которое позволяет отлаживать алгоритмы управления без необходимости дорогостоящего прототипирования и рисков повреждения реального оборудования [6, 8]. Для этих целей используются как специализированные программные платформы, например, Gazebo, Webots, MATLAB/Simulink, так и программные решения собственной разработки. Среди последних своими доступностью, мощными средствами визуализации, гибкостью и растущим набором инструментов для робототехники выделяется игровой движок Unity. Результаты работы могут применяться как в образовательных целях, в частности при изучении дисциплины «Роботизация производства» в Ошском Государственном Университете (ОшГУ), так и на практике для отладки тестирования алгоритмов управления перед их загрузкой на реальный робот.

Подготовка 3D-модели и организация кинематической структуры. Для моделирования робота в программной среде Unity требуется наличие трехмерной модели, которая корректно отражает его физическую и кинематическую структуру, обеспечивая возможность реалистичного воспроизведения движения и работы робота. Существуют два основных подхода по реализации трехмерных моделей роботов: самостоятельное проектирование модели в CAD системе и использование готовых моделей, которые часто предоставляются производителем роботов. При наличии необходимых технических данных и требований к конструкции робота, проектирование модели может производиться в одной из популярных CAD систем, к которым относятся, в частности, Autodesk Fusion 360, SolidWorks и FreeCAD. При этом проектируемая модель должна учитывать геометрические параметры звеньев, их оси вращения и допустимые диапазоны движения, обеспечивая соответствие типовой конфигурации реального робота. Альтернативой самостоятельному проектированию может быть использование готовых CAD-моделей, размещенных в открытых инженерных библиотеках (GrabCAD, Thingiverse и др.) или предоставляемых производителями промышленных роботов в открытом доступе. При выборе подобной модели следует убедиться в её достаточной детализации, также в соответствии габаритов и конструкции техническим требованиям. После подготовки трёхмерной модели следующим этапом является её импорт и настройка в среде Unity. На первом этапе модель робота необходимо экспортировать из CAD-среды в один из форматов, поддерживаемых Unity, таких как .fbx, .obj или .stl. При этом, наиболее предпочтительным является формат .fbx, поскольку он позволяет сохранить иерархию объектов, трансформации, а также текстуры и материалы, применённые в CAD-среде.

Экспортированный файл помещается в папку Assets проекта Unity и автоматически отображается в обозревателе проекта. После этого модель перетаскивается на сцену для дальнейшей настройки. В окне Inspector необходимо проверить корректность масштаба и ориентацию модели. При необходимости следует привести единицы измерения CAD-системы к единицам Unity и/или выполнить поворот модели, поскольку направления систем координат в CAD и Unity могут отличаться. Кроме того, следует убедиться, что оси вращения и направления поступательного перемещения звеньев точно соответствуют кинематической структуре робота, а все опорные точки расположены в области соответствующих сочленений. Следующим этапом является организация кинематической

структуры робота, для которой в Unity предусмотрено несколько способов моделирования связей между звеньями механической системы. В данной работе рассматриваются три наиболее распространённых подхода: иерархическая структура объектов (Parent-Child), использование физических соединений (Joints), а также компонент Articulation Body, предназначенный специально для задач робототехники [2].

Самым простейшим способом построения кинематической цепи является отношение родитель-потомок (Parent-Child), при котором каждое звено назначается дочерним объектом предыдущего. При таком подходе управление движением звеньев осуществляется напрямую путём изменения параметров позиции и ориентации компонента transform. Основным достоинством этого подхода является простота реализации: он не требует настройки физических свойств и позволяет интуитивно управлять положением звеньев. Однако при этом отсутствует физическая симуляция — не учитываются масса, инерция и внешние силы, что ограничивает реалистичность поведения модели.

Другой подход к моделированию связей между звеньями кинематической цепи заключается в использовании компонентов Hinge Joint или Configurable Joint, которые реализуют физические соединения с возможностью задания осей вращения, ограничений углов, жёсткости, демпфирования и других параметров взаимодействия, при этом физические свойства звеньев, такие как масса и материал, задаются через компонент твердого тела (Rigidbody). К основным преимуществам данного подхода стоит отнести возможность более реалистичного воспроизведения поведения робота с учётом силовых воздействий и моментов, однако при этом возрастает сложность настройки, увеличивается потребность в вычислительных ресурсах, в том числе за счёт необходимости повышения точности физических расчётов, а также может наблюдаться нестабильность при усложнении конфигурации механизма. Наиболее современным и функциональным способом моделирования кинематических цепей в Unity, ориентированным на задачи робототехники, является применение компонента Articulation Body, который позволяет точно настраивать параметры сочленений, включая массы, моменты инерции, диапазоны углов и типы приводов. Основным преимуществом данного подхода является высокая точность моделирования и стабильность численных расчётов, что делает его эффективным при симуляции сложных робототехнических систем. Вместе с тем компонент Articulation Body доступен только в версиях Unity 2020 и выше, и содержит ряд ограничений, в частности, не поддерживает моделирование замкнутых кинематических цепей. Таким образом, выбор метода организации связей между звеньями определяется требованиями к точности моделирования и уровню сложности решаемой задачи. В случаях, не предполагающих физического взаимодействия, допустимо использование иерархической структуры типа Parent–Child, тогда как для более точного и физически обоснованного моделирования предпочтение следует отдавать компонентам Joints или Articulation Body.

Реализация управления виртуальной моделью робота SCARA. На этапе реализации управления виртуальной моделью основное внимание уделяется обеспечению возможности управления положением и движением звеньев SCARA-манипулятора, как в ручном режиме, так и с использованием алгоритмов решения задач прямой и обратной кинематики. Управление может быть реализовано как напрямую через пользовательский интерфейс или устройства ввода (например, клавиатура или джойстик), так и программно, путём расчёта требуемых параметров движения на основе математических моделей [5].

В настоящей работе рассмотрено два способа управления роботом: ручное управление, при котором оператор непосредственно задаёт значения углов поворота вращательных звеньев и величины линейного перемещения вдоль осей поступательных звеньев, и

автоматическое управление, реализуемое через решение обратной задачи кинематики, которое позволяет определение требуемых углов поворота звеньев по заданным координатам рабочего органа. Для целей тестирования и отладки виртуальной модели SCARA-робота реализована возможность управления звеньями в интерактивном режиме. Управление может осуществляться с использованием различных средств ввода, включая клавиатуру, графический интерфейс пользователя и внешние устройства, такие как джойстики. При управлении с клавиатуры изменение углов вращения звеньев осуществляется пошагово, что позволяет оперативно проверять корректность работы сочленений и соблюдение кинематических ограничений. Графический интерфейс на базе встроенных элементов UI обеспечивает наглядность и удобство взаимодействия, особенно в образовательных или демонстрационных сценариях. Использование внешних устройств ввода, таких как геймпады или трекеры, реализуется с помощью встроенных средств Unity и позволяет расширить возможности управления моделью в интерактивной среде. Ниже, в листинге 1, приведена обобщённая структура реализации ручного управления с использованием C#-скрипта в Unity:

```
void Update()
{
    // Управление первым звеном
    if (Input.GetKey(KeyCode.A)) theta1 -= step;
    if (Input.GetKey(KeyCode.D)) theta1 += step;

    // Управление вторым звеном
    if (Input.GetKey(KeyCode.W)) theta2 += step;
    if (Input.GetKey(KeyCode.S)) theta2 -= step;

    ApplyJointRotation(theta1, theta2);
}
```

Листинг 1. Пример кода ручного управления звеньями робота в общем случае

При этом метод *ApplyJointRotation()* реализуется в зависимости от способа организации звеньев (см. ниже) и отвечает за физическую или трансформационную установку новых значений. Реализация управления звеньями напрямую связана с выбранным способом моделирования звеньев, описанными выше. В случае организации звеньев при помощи отношения родитель-потомок (*Parent-Child*) управление реализуется при помощи изменения локального поворота или линейного перемещения соответствующего звена, в зависимости от его типа, так как показано в Листинге 2.

```
link1.transform.localRotation = Quaternion.Euler(0, theta1, 0);
link2.transform.localRotation = Quaternion.Euler(0, theta2, 0);
```

Листинг 2. Пример кода управления звеньями, связанных отношением родитель-потомок

Как было отмечено выше такой подход достаточно просто реализуется, однако не может применяться в задачах, в которых необходимо учитывать силы и моменты. При организации соединений звеньев с применением компонентов *Hinge Joint* / *Configurable Joint*, управление производиться путем задания свойств физического мотора (Листинг 3), а также величин массы и типов материалов звеньев.

Также, при таком подходе, в случае необходимости можно жёстко задавать углы через параметры пружины (*spring*) и целевого положения (*target position*).

Т. 11. №7 2025
https://doi.org/10.33619/2414-2948/116

Наиболее точным и универсальным способом соединения звеньев, который поддерживает полный набор физических параметров, таких как инерция, жёсткость, трение и др., является применение компонента *Articulation Body*. В этом случае управление осуществляется путем установки целевого положения (*Target Position*) так, как показано в Листинге 4.

```
var motor = joint.motor;
motor.targetVelocity = targetSpeed;
joint.motor = motor;
```

Листинг 3. Пример кода управления звеньями, связанных компонентами *Hinge Joint* / *Configurable Joint*

```
        ArticulationDrive drive = articulation.xDrive;
drive.target = targetAngle;
articulation.xDrive = drive;
```

Листинг 4. Пример кода управления звеньями, связанных компонентами *Articulation Body*

Дальнейшее управление виртуальной моделью робота SCARA осуществляется посредством решения задач прямой и обратной кинематики [3-7]. Прямая задача кинематики заключается в определении положения и ориентации рабочего органа робота на основании заданных углов поворота его сочленений. Применительно к роботу SCARA, эта задача сводится к вычислению координат рабочего органа (эффектора) на основе известных углов и длин звеньев манипулятора. Иными словами, при заданных значениях углов сочленений можно легко и однозначно определить, где именно будет находиться рабочий орган робота. Обратная же задача кинематики, напротив, решает вопрос о том, как именно нужно расположить звенья робота, чтобы рабочий орган занял заданное положение в пространстве. Необходимо отметить, что на практике обратная задача кинематики является более востребованной и имеет большую значимость в роботизированных системах. Однако эта задача редко имеет единственное решение. В частности, для заданного набора углов сочленений (Q_1, Q_2, …, Q_n) всегда можно однозначно определить положение рабочего органа (x, y, z), но обратное утверждение справедливо не всегда. Для одной и той же точки (x, y, z) может существовать несколько различных наборов углов (Q_1', Q_2', …, Q_n'). Это обусловлено математической спецификой задачи, где решения часто выражаются через многочлены и квадратные корни, что порождает множественность решений [8].

Для более детального понимания рассмотрим пример решения прямой задачи кинематики для плоского двухзвенного манипулятора, представленного на Рисунке 1. На рисунке изображён упрощенный манипулятор SCARA, работающий в одной плоскости и состоящий из двух вращательных звеньев: плеча (L_1) и локтя (L_2). Первый сустав закреплён на основании и вращается на угол Q_1 относительно горизонтальной оси. Второе звено крепится к концу первого и совершает вращение на угол Q_2 относительно первого звена. Эффектор расположен на конце второго звена. Прямая задача кинематики формулируется следующим образом: зная длины звеньев (L_1, L_2) и углы вращения (Q_1, Q_2), требуется определить координаты рабочего органа (x, y). Формально это выражается следующими уравнениями:

$$x = L_1 \cdot cos(Q_1) + L_2 \cdot cos(Q_1 + Q_2) \quad (1)$$

$$y = L_1 \cdot sin(Q_1) + L_2 \cdot sin(Q_1 + Q_2) \quad (2)$$

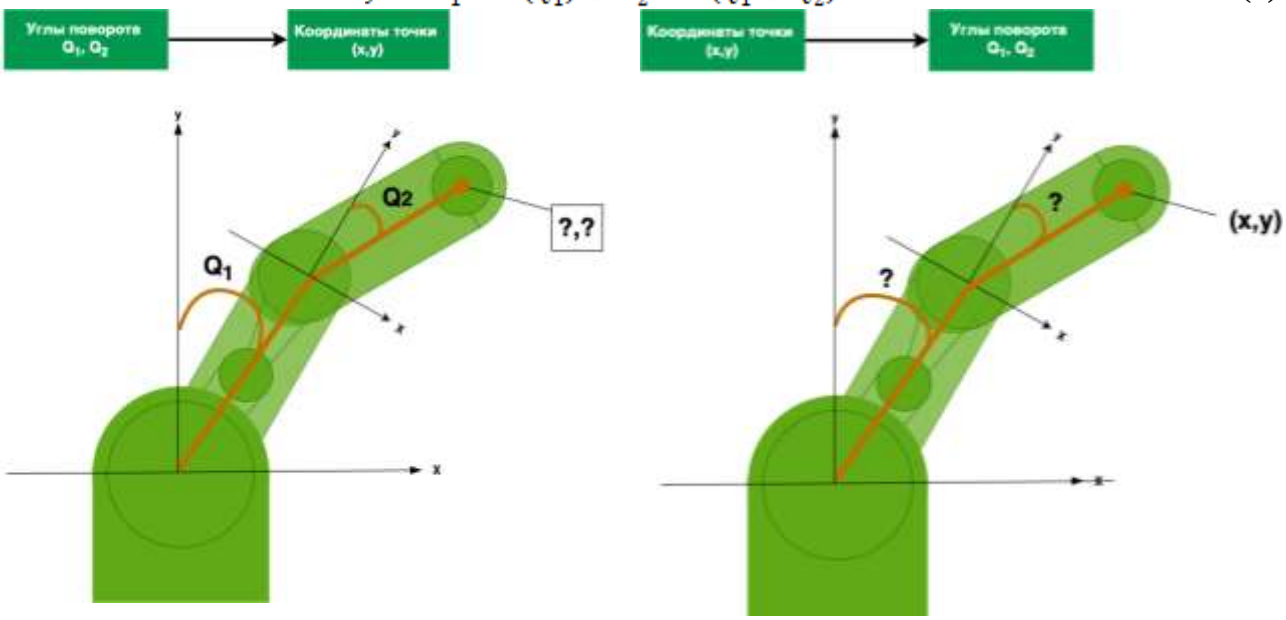

Рисунок 1. Схема двухзвенного плоского манипулятора SCARA для иллюстрации решения задач кинематики

В листинге 5 приведен пример кода расчета прямой задачи кинематики для робота SCARA в среде Unity3D. В данном методе используются встроенные математические функции среды Unity (Mathf), а результатом выполнения функции является трёхмерный вектор, описывающий положение рабочего органа в плоскости манипулятора. Этот подход позволяет удобно интегрировать кинематические расчеты непосредственно в виртуальную среду и оперативно визуализировать результаты.

```
public Vector3 CalculateEndEffectorPosition(float L1, float L2, float Q1, float Q2)
{
    float x = L1 * Mathf.Cos(Q1) + L2 * Mathf.Cos(Q1 + Q2);
    float y = L1 * Mathf.Sin(Q1) + L2 * Mathf.Sin(Q1 + Q2);

    return new Vector3(x, y, 0);
}
```

Листинг 5. Пример кода расчета прямой задачи кинематики для робота SCARA в среде Unity3D

Обратная задача кинематики используется для определения углов звеньев, которые обеспечат заданное положение рабочего органа. Этот метод часто применяется в робототехнике, когда необходимо расположить инструмент робота в конкретных координатах (x, y). Для решения обратной задачи кинематики сначала рассчитывается расстояние от начала координат до заданной точки (x, y), обозначим его как B:

$$B^2 = x^2 + y^2 \quad (3)$$

Угол q_1 между осью OX и прямой B находится по формуле:

$$q_1 = \arctan\left(\frac{y}{x}\right) \quad (4)$$

Угол q^2 между прямой B и звеном L1 вычисляется по теореме косинусов:

$$q_2 = \arccos\left(\frac{L_1^2 + B^2 - L_2^2}{2 \cdot L_1 \cdot B}\right) \quad (5)$$

Тогда угол Q_1 рассчитывается как разность:

$$Q_1 = q_1 - q_2 \quad (6)$$

Аналогично, угол Q_2 между звеньями L_1 и L_2 также определяется по теореме косинусов:

$$Q_2 = \pi - \arccos\left(\frac{L_1^2 + L_2^2 - B^2}{2 \cdot L_1 \cdot L_2}\right) \quad (7)$$

Следует учесть, что для заданной точки (X, Y) существует альтернативная конфигурация манипулятора, когда:

$$Q_1 = q_1 + q_2 \quad (8)$$

$$Q_2 = -Q_2 \quad (9)$$

Таким образом, при расчете обратной задачи кинематики всегда необходимо учитывать возможность неоднозначности решения. Пример кода расчета обратной задачи кинематики в Unity3D приведен в листинге 6. В приведенном листинге кода метод *CalculateInverseKinematics* принимает на вход длины звеньев (L_1, L_2) и координаты целевой точки, а дополнительный параметр *elbowUp* позволяет выбрать конфигурацию манипулятора: «локоть вверх» или «локоть вниз». Стоит отметить, что рассмотренные формулы прямой и обратной задач кинематики применимы для упрощенной плоской модели манипулятора SCARA, работающей в одной плоскости (*XY*). В реальных задачах для робота SCARA дополнительно задаётся вертикальное линейное перемещение рабочего органа по оси *Z*, которое обычно регулируется независимо от вращательных степеней свободы и рассчитывается отдельно.

```
        public bool CalculateInverseKinematics(float L1, float L2, Vector2 target, out float Q1, out float
Q2, bool elbowUp = true)
        {
            // Вычисляем расстояние от основания манипулятора до заданной точки
            float B = target.magnitude;

            // Проверка достижимости точки
            if (B > L1 + L2 || B < Mathf.Abs(L1 - L2))
            {
                // Точка находится вне зоны досягаемости манипулятора
                Q1 = Q2 = 0;
                return false;
            }

            // Угол q1 между осью OX и прямой, соединяющей основание с целевой точкой
            float q1 = Mathf.Atan2(target.y, target.x);

            // Угол q2 по теореме косинусов
            float q2 = Mathf.Acos((L1 * L1 + B * B - L2 * L2) / (2 * L1 * B));

            // Угол Q1 зависит от выбранной конфигурации (локоть вверх или вниз)
            Q1 = elbowUp ? (q1 - q2) : (q1 + q2);

            // Угол Q2 между плечом (L1) и локтем (L2)
            float angleQ2 = Mathf.Acos((L1 * L1 + L2 * L2 - B * B) / (2 * L1 * L2));
            Q2 = elbowUp ? (Mathf.PI - angleQ2) : (angleQ2 - Mathf.PI);

            return true;
        }
```

Т. 11. №7 2025
https://doi.org/10.33619/2414-2948/116

Листинг 6. Пример кода расчета обратной задачи кинематики для робота SCARA в среде Unity3D

Для обеспечения плавного и реалистичного перемещения рабочего органа робота SCARA между заданными позициями, помимо расчета положений через прямую и обратную задачи кинематики, важное значение приобретает расчет траектории движения. Одним из наиболее простых и распространенных подходов является линейная интерполяция между двумя заданными точками, при которой промежуточные положения рабочего органа рассчитываются как линейная комбинация начальной и конечной позиций. Ниже, в листинге 7, приведен пример реализации линейной интерполяции в Unity3D:

```
Vector3 startPosition = CalculateEndEffectorPosition(L1, L2, Q1_start, Q2_start);
Vector3 endPosition = CalculateEndEffectorPosition(L1, L2, Q1_end, Q2_end);

for (float t = 0; t <= 1; t += 0.01f)
{
    Vector3 currentPosition = Vector3.Lerp(startPosition, endPosition, t);
    // Визуализация текущего положения рабочего органа
}
```

Листинг 7. Пример кода линейной интерполяции траектории движения рабочего органа

Перспективной возможностью виртуальной модели робота является её интеграция с системами числового программного управления (ЧПУ), что позволяет использовать стандартный язык программирования траекторий G-код. G-код представляет собой последовательность команд, которые определяют действия робота, такие как перемещения по заданным координатам, выбор скоростей и другие параметры работы. В рамках данной работы был разработан модуль для обработки и интерпретации G-кода в Unity3D. Базовые команды, поддерживаемыми модулем приведены в Таблице.

Таблица

БАЗОВЫЕ КОМАНДЫ G-КОДА, ПОДДЕРЖИВАЕМЫЕ
В РАЗРАБОТАННОМ ПРОГРАММНОМ ОБЕСПЕЧЕНИИ

Команда	*Описание*
G0/G1	Линейное перемещение рабочего органа в указанные координаты (*x*, *y*, *z*).
G90	Абсолютное позиционирование, координаты задаются относительно нулевой точки.
G91	Относительное позиционирование, координаты задаются относительно текущего положения.
F	Установка скорости перемещения рабочего органа.

Процесс обработки G-кода включает следующие этапы:

Парсинг строк G-кода из текстового файла и выделение команд и их параметров.

Преобразование выделенных координат и параметров в целевые позиции с использованием обратной задачи кинематики.

Выполнение расчетов промежуточных точек траектории с применением линейной интерполяции, обеспечивающей плавность движения робота.

Пример упрощенного алгоритма парсинга команды G-кода представлен ниже в листинге 8:

Скорость движения робота, заданная командой F в G-коде, реализуется путём изменения параметра времени или шага интерполяции между точками траектории. Чем выше заданная скорость (значение F), тем меньшее время затрачивается на прохождение каждой промежуточной точки траектории, и наоборот. В Unity это достигается регулировкой

Т. 11. №7 2025
https://doi.org/10.33619/2414-2948/116

скорости изменения параметра интерполяции (например, величиной шага t в цикле интерполяции). Таким образом, виртуальная среда позволяет предварительно отлаживать программы, предназначенные для реального оборудования, минимизируя риски возникновения ошибок и увеличивая эффективность реальной эксплуатации робота [9].

```
string command = "G1 X10 Y20 Z0 F1500";
var parts = command.Split(' ');
float x = 0, y = 0, z = 0, speed = 0;

foreach (var part in parts)
{
    if (part.StartsWith("X")) x = float.Parse(part.Substring(1));
    else if (part.StartsWith("Y")) y = float.Parse(part.Substring(1));
    else if (part.StartsWith("Z")) z = float.Parse(part.Substring(1));
    else if (part.StartsWith("F")) speed = float.Parse(part.Substring(1));
}
```

Листинг 8. Пример кода парсинга команды G-кода

Поддержка G-кода существенно расширяет область применения созданной модели и позволяет более гибко подходить к решению различных производственных задач. В рамках настоящей работы авторами была разработана демонстрационная программа в среде Unity3D, реализующая описанные выше алгоритмы. Приложение позволяет интерактивно изменять значения углов звеньев и координаты рабочего органа, визуализируя расчеты прямой и обратной задач кинематики в реальном времени. На Рисунке 2 представлен скриншот интерфейса Unity с демонстрационной программы по управлению роботом SCARA.

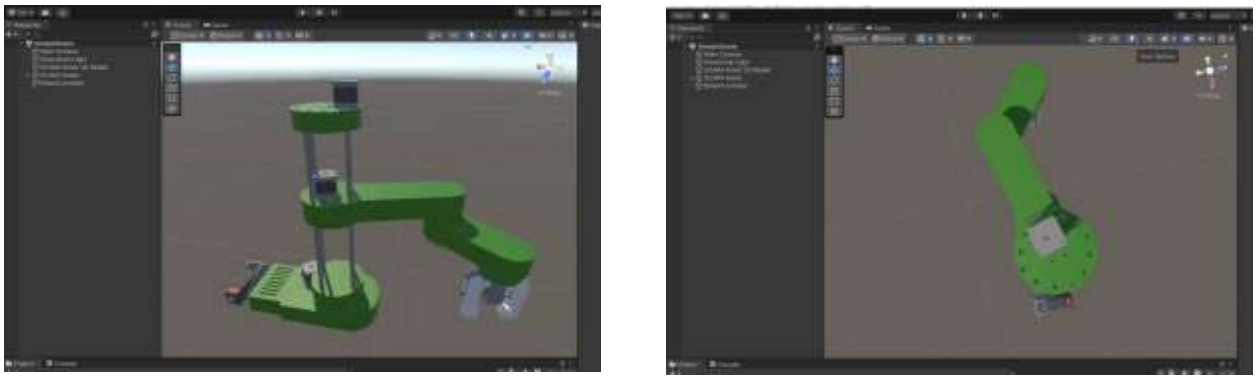

Рисунок 2. Скриншот разработанной демонстрационной программы по управлению роботом SCARA в Unity3D

Разработанная виртуальная модель робота SCARA имеет значительный потенциал для дальнейшего развития. К перспективным направлениям можно отнести:

-реализацию более сложных алгоритмов управления, таких как планирование траекторий с использованием искусственного интеллекта и машинного обучения, что обеспечит адаптивность и автономность поведения робота [10];

-создание модуля для удалённого доступа к виртуальной модели через веб-интерфейс, что позволит использовать её в дистанционном обучении;

-разработку полной цифровой копии (digital twin) конкретного реального робота SCARA с целью мониторинга, диагностики и предиктивного обслуживания оборудования;

-внедрение виртуальной модели в учебные курсы, например, в рамках дисциплины «Роботизация производства», изучаемой в Ошском Государственном Университете [11].

Стоит отметить, что роботы типа SCARA часто используются для автоматизации типовых задач, таких как сборка, перемещение и упаковка, что делает их востребованными на производстве. Поэтому, использование разработанной виртуальной модели, позволяет студентам на практике изучить принципы работы и управления промышленными роботами, без рисков и затрат, связанных с эксплуатацией реального оборудования.

Итак, представлены основные этапы разработки виртуальной модели робота SCARA в среде Unity3D, включающие подготовку и настройку 3D-модели, организацию кинематической структуры, реализацию базовых алгоритмов управления с решением прямой и обратной задач кинематики, а также построение траекторий движения с использованием программирования на основе G-кода [12-17].

Проведённые исследования подтвердили удобство и эффективность применения среды Unity3D для задач виртуального моделирования и отладки алгоритмов управления роботизированными системами, что открывает широкие перспективы как для образовательных целей, так и для практических приложений в робототехнике. Практическая значимость работы состоит в её образовательном применении, а также в возможности проведения предварительного тестирования и оптимизации алгоритмов управления перед их внедрением в реальное производство.

Список литературы:

1. Craig J. J. Introduction to robotics: mechanics and control, 3/E. Pearson Education India, 2009.

2. Bin Uzayr S. Mastering Unity: A Beginner's Guide. CRC Press, 2022.

3. Siciliano B., Sciavicco L., Villani L., Oriolo G. Mobile robots // Robotics: Modelling, Planning and Control. 2009. P. 469-521. https://doi.org/10.1007/978-1-84628-642-1_11

4. Corke P. I., Jachimczyk W., Pillat R. Robotics, vision and control: fundamental algorithms in MATLAB. Berlin : Springer, 2011. V. 73. P. 2.

5. Зенкевич С. Л., Ющенко А. С. Основы управления манипуляционными роботами. М.: Изд-во МГТУ, 2004. 480 с.

6. Suárez Sánchez P. Webots-Based Implementation and Simulation of Robotics Algorithms. 2024.

7. Подураев Ю. В. Мехатроника: основы, методы, применение. М.: Машиностроение, 2006. 256 с.

8. Morel Y. The European Coordination Hub for Open Robotics Development++: An Overview // Advances in Robotics Research: From Lab to Market: ECHORD++: Robotic Science Supporting Innovation. 2019. P. 3-11. https://doi.org/10.1007/978-3-030-22327-4_1

9. Каляев И. А., Лохин В. М., Макаров И. М. Интеллектуальные роботы. М.: Машиностроение, 2007. 360 с.

10. Шестаков Е. И., Жданов А. А. Адаптивное управление модульным реконфигурируемым манипуляционным роботом // Нейроинформатика-2020: XXII Международная научно-техническая конференция. М., 2020. С. 18-26.

11. Шестаков Е. И., Пирматов А. З., Жолдошов Т. М. Применение Unity 3D для развития профессиональных навыков // Вестник Ошского государственного университета. 2024. №2. С. 370–383.

12. Cao W. A., Li S., Cheng P., Ge M., Ding H., Lai J. Design and development of a new 4 DOF hybrid robot with Scara motion for high-speed operations in large workspace // Mechanism and Machine Theory. 2024. V. 198. P. 105656. https://doi.org/10.1016/j.mechmachtheory.2024.105656

13. Tao F., Zhang H., Liu A., Nee A. Y. Digital twin in industry: State-of-the-art // IEEE Transactions on industrial informatics. 2018. V. 15. №4. P. 2405-2415. https://doi.org/10.1109/TII.2018.2873186

14. Al Zami M. B., Shaon S., Quy V. K., Nguyen D. C. Digital twin in industries: A comprehensive survey // IEEE Access. 2025. https://doi.org/10.1109/ACCESS.2025.3551532

15. Zafar M. H., Langås E. F., Sanfilippo F. Exploring the synergies between collaborative robotics, digital twins, augmentation, and industry 5.0 for smart manufacturing: A state-of-the-art review // Robotics and Computer-Integrated Manufacturing. 2024. V. 89. P. 102769. https://doi.org/10.1016/j.rcim.2024.102769

16. Li Y., Zhang Q., Xu H., Lim E., Sun J. Virtual monitoring system for a robotic manufacturing station in intelligent manufacturing based on Unity 3D and ROS // Materials today: proceedings. 2022. V. 70. P. 24-30. https://doi.org/10.1016/j.matpr.2022.08.486

17. Zhang Z., Guo Q., Grigorev M. A., Kholodilin I. Construction Method of a Digital-Twin Simulation System for SCARA Robots Based on Modular Communication // Sensors. 2024. V. 24. №22. P. 7183. https://doi.org/10.3390/s24227183

References:

1. Craig, J. J. (2009). *Introduction to robotics: mechanics and control, 3/E*. Pearson Education India.

2. Bin Uzayr, S. (2022). *Mastering Unity: A Beginner's Guide*. CRC Press.

3. Siciliano, B., Sciavicco, L., Villani, L., & Oriolo, G. (2009). Mobile robots. *Robotics: Modelling, Planning and Control*, 469-521. https://doi.org/10.1007/978-1-84628-642-1_11

4. Corke, P. I., Jachimczyk, W., & Pillat, R. (2011). *Robotics, vision and control: fundamental algorithms in MATLAB* (Vol. 73, p. 2). Berlin: Springer.

5. Zenkevich, S. L., & Yushchenko, A. S. (2004). Osnovy upravleniya manipulyatsionnymi robotami. Moscow. (in Russian).

6. Suárez Sánchez, P. (2024). *Webots-Based Implementation and Simulation of Robotics Algorithms* (Bachelor's thesis).

7. Poduraev, Yu. V. (2006). Mekhatronika: osnovy, metody, primenenie. Moscow. (in Russian).

8. Morel, Y. (2019). The European Coordination Hub for Open Robotics Development++: An Overview. *Advances in Robotics Research: From Lab to Market: ECHORD++: Robotic Science Supporting Innovation*, 3-11. https://doi.org/10.1007/978-3-030-22327-4_1

9. Kalyaev, I. A., Lokhin, V. M., & Makarov, I. M. (2007). Intellektual'nye roboty. Moscow. (in Russian).

10. Shestakov, E. I., & Zhdanov, A. A. (2020). Adaptivnoe upravlenie modul'nym rekonfiguriruemym manipulyatsionnym robotom. In *Neiroinformatika-2020: XXII Mezhdunarodnaya nauchno-tekhnicheskaya konferentsiya*, 18-26. Moscow. (in Russian).

11. Shestakov, E. I., Pirmatov, A. Z., & Zholdoshov, T. M. (2024). Primenenie Unity 3D dlya razvitiya professional'nykh navykov. *Vestnik Oshskogo gosudarstvennogo universiteta,* (2), 370–383. (in Russian).

12. Cao, W. A., Li, S., Cheng, P., Ge, M., Ding, H., & Lai, J. (2024). Design and development of a new 4 DOF hybrid robot with Scara motion for high-speed operations in large workspace. *Mechanism and Machine Theory*, *198*, 105656. https://doi.org/10.1016/j.mechmachtheory.2024.105656

13. Tao, F., Zhang, H., Liu, A., & Nee, A. Y. (2018). Digital twin in industry: State-of-the-art. *IEEE Transactions on industrial informatics*, *15*(4), 2405-2415. https://doi.org/10.1109/TII.2018.2873186

14. Al Zami, M. B., Shaon, S., Quy, V. K., & Nguyen, D. C. (2025). Digital twin in industries: A comprehensive survey. *IEEE Access*. https://doi.org/10.1109/ACCESS.2025.3551532

15. Zafar, M. H., Langås, E. F., & Sanfilippo, F. (2024). Exploring the synergies between collaborative robotics, digital twins, augmentation, and industry 5.0 for smart manufacturing: A state-of-the-art review. *Robotics and Computer-Integrated Manufacturing*, *89*, 102769. https://doi.org/10.1016/j.rcim.2024.102769

16. Li, Y., Zhang, Q., Xu, H., Lim, E., & Sun, J. (2022). Virtual monitoring system for a robotic manufacturing station in intelligent manufacturing based on Unity 3D and ROS. *Materials today: proceedings*, *70*, 24-30. https://doi.org/10.1016/j.matpr.2022.08.486

17. Zhang, Z., Guo, Q., Grigorev, M. A., & Kholodilin, I. (2024). Construction Method of a Digital-Twin Simulation System for SCARA Robots Based on Modular Communication. *Sensors*, *24*(22), 7183. https://doi.org/10.3390/s24227183

Работа поступила
в редакцию 16.05.2025 г.

Принята к публикации
22.05.2025 г.

Ссылка для цитирования:

Шестаков Е. И., Пирматов А. З., Клинцов М. А. Виртуальное моделирование робота SCARA в Unity3D // Бюллетень науки и практики. 2025. Т. 11. №7. С. 116-127. https://doi.org/10.33619/2414-2948/116/14

Cite as (APA):

Shestakov, E., Pirmatov, A., & Klintsov, M. (2025). Virtual Simulation of a SCARA Robot in Unity3D. *Bulletin of Science and Practice*, *11*(7), 116-127. (in Russian). https://doi.org/10.33619/2414-2948/116/14