МЕТОД ИНТЕРАКТИВНОГО ОТЗЕРКАЛИВАНИЯ ДВИЖЕНИЙ ПОЛЬЗОВАТЕЛЯ НА ТРЁХМЕРНОМ VRM-АВАТАРЕ ПО ДАННЫМ ВЕБ-КАМЕРЫ В СРЕДЕ БРАУЗЕРА

A BROWSER-BASED METHOD OF INTERACTIVE MIRRORING OF USER MOVEMENTS ONTO A 3D VRM AVATAR USING WEBCAM INPUT

A BROWSER-BASED METHOD OF INTERACTIVE MIRRORING OF USER MOVEMENTS ONTO A 3D VRM AVATAR USING WEBCAM INPUT

УДК 004.92

Кащеев Иван Васильевич, студент, Московский политехнический университет, Россия, г. Москва

Kashcheev Ivan Vasilevich, student, Moscow Polytechnic University, Russia, Moscow

e-mail: basiliris69@gmail.com

Аннотация

В работе изложены предпосылки, алгоритмические решения и результаты программной реализации метода интерактивного отзеркаливания движений пользователя на трёхмерном аватаре формата VRM по данным с одной веб-камеры. Метод предназначен для задач визуализации русского жестового языка в учебных и ассистивных веб-приложениях; вычисления выполняются полностью в браузере, без обращения к серверу и без специализированной mocap-аппаратуры. Описаны мотивация выбора стека (React, Three.js, React Three Fiber, MediaPipe Holistic, KalidoKit, плагин @pixiv/three-vrm), структура конвейера, правила вычисления целевых поворотов костей, применение сферической линейной интерполяции с разными коэффициентами для корпуса, конечностей и фаланг пальцев, а также межкадровая стабилизация ориентиров кисти. Отдельно рассмотрена обработка мимики, грамматически значимой для РЖЯ: блендшейпы форм рта

aa/ih/ee/oh/ou, стабилизация моргания и согласование систем координат для направления взгляда. Описаны проблемы, выявленные в процессе разработки: дрожание пальцев при низкой освещённости, зеркальная инверсия сторон, рассогласование осей при пересчёте взгляда. Тестирование на станции с Intel Core i7-12700 и веб-камерой Logitech C920 показало среднюю частоту 58 кадров в секунду; субъективные оценки качества — 3,8 балла для корпуса, 3,2 балла для пальцев, 3,5 балла для форм рта и 4,0 балла для моргания по пятибалльной шкале.

Annotation

The paper presents the rationale, algorithmic decisions and implementation details of a method for interactive mirroring of user movements onto a 3D VRM avatar using input from a single webcam. The method targets Russian Sign Language visualization tasks in educational and assistive web applications and runs entirely in the browser, without server inference or dedicated motion-capture hardware. We describe the rationale for the chosen software stack (React, Three.js, React Three Fiber, MediaPipe Holistic, KalidoKit, @pixiv/three-vrm), the structure of the processing pipeline, the computation of target bone quaternions, the use of spherical linear interpolation with different smoothing coefficients for the torso, limbs and finger phalanges, and frame-to-frame stabilization of hand landmarks. Particular attention is paid to facial expression processing, which carries grammatical load in RSL: mouth-shape blendshapes aa/ih/ee/oh/ou, blink stabilization and coordinate-frame alignment for gaze. Issues encountered during development and their workarounds are discussed: finger jitter under poor lighting, left–right mirroring of the webcam feed and axis mismatches in gaze computation. Functional testing on a workstation with an Intel Core i7-12700 processor and a Logitech C920 webcam yielded an average frame rate of 58 FPS, with subjective scores of 3.8 (torso), 3.2 (fingers), 3.5 (mouth shapes) and 4.0 (blinks) out of 5.

Ключевые слова: русский жестовый язык, отзеркаливание движений, VRM-аватар, MediaPipe Holistic, MediaPipe Face Mesh, KalidoKit, блендшейпы мимики, веб-приложение, сферическая линейная интерполяция.

Keywords: Russian sign language, motion mirroring, VRM avatar, MediaPipe Holistic, MediaPipe Face Mesh, KalidoKit, facial blendshapes, web application, spherical linear interpolation.

Введение

По оценкам Всемирной организации здравоохранения, инвалидизирующая потеря слуха зафиксирована более чем у 430 млн человек по всему миру; для России разные источники дают цифру около 13 млн человек [2]. Русский жестовый язык (РЖЯ) с 2012 года имеет официальный статус средства общения при нарушениях слуха, однако уровень его программной поддержки в повседневной цифровой среде остаётся заметно ниже, чем у звучащих языков.

Практика обучения РЖЯ обнажает ещё одну проблему: учащимся не хватает средств самоконтроля. Студент, разучивающий жест, как правило не видит собственную артикуляцию со стороны. Запись на телефон в качестве кустарного решения неудобна — приходится переключаться между камерой и образцом, теряя фокус на самом движении. Один из способов снять это ограничение — «живое зеркало» в виде трёхмерного аватара, который рядом с эталонным жестом повторяет движения обучающегося в реальном времени, в стилизованном виде, без детализации внешности и помех освещения. Именно такой сценарий лёг в основу разрабатываемого приложения.

До недавнего времени отображение движений человека на 3D-модели решалось преимущественно аппаратно — инерциальными костюмами, многокамерными студиями оптического захвата, перчатками с волоконно-оптическими измерителями. Финансово такие решения для учебной аудитории недостижимы. Появление библиотек компьютерного зрения, способных в реальном времени извлекать координаты ключевых точек тела и рук из обычной веб-камеры, изменило ситуацию принципиально. Наиболее заметным шагом в этом направлении стал релиз фреймворка MediaPipe от исследователей Google [8]. Параллельно развитие WebGL и библиотеки

Three.js [7] позволило перенести визуализацию в окно браузера и отказаться от установки отдельных приложений на стороне пользователя.

Разработка проекта велась с опорой на наработки, накопленные в работах научной школы факультета. В частности, использовались идеи и результаты по конвертации видео в анимацию средствами видеотрекинга для мультиязычного словаря жестового языка [1], а также по построению такого словаря с обратной связью через аватара в реальном времени [6]. В смежной плоскости — детекции жестов и рук в кадре, классификации движений и применению свёрточных нейросетей для распознавания конфигураций кисти — учитывались подходы из [9, 11]. Параметризованный синтез жестов и их последовательностей на основе анимированных 3D-моделей описан в [5]. Настоящая статья сосредоточена именно на инженерной стороне отзеркаливания (вывода движений пользователя на аватар); вопросы распознавания жестов и воспроизведения заранее подготовленных анимаций здесь не рассматриваются.

Цель исследования

Цель работы — разработать и программно реализовать метод интерактивного отзеркаливания движений пользователя на 3D-аватаре формата VRM по данным с единственной веб-камеры, работающий в браузерной среде в режиме реального времени. К частным задачам отнесены: обосновать выбор стека библиотек компьютерного зрения и 3D-графики; построить конвейер обработки видеопотока; разработать правила преобразования ключевых точек в повороты костей аватара; спроектировать механизмы сглаживания, компенсирующие шум входных данных; провести функциональное тестирование и собрать субъективные оценки качества.

Материал и методы исследования

Предпосылки и требования к методу

Главная исходная предпосылка — минимальный порог входа: пользователь запускает приложение в браузере на обычном ноутбуке с

фронтальной веб-камерой и сразу видит на экране 3D-фигуру, повторяющую его движения. Отсюда несколько требований. Отправка видео на сервер исключена сразу по двум причинам: задержки сетевого канала несовместимы с режимом реального времени, а пересылка кадров с лицом пользователя на удалённый сервер ставит вопрос о согласии на обработку персональных данных. Алгоритмы исполняются на JavaScript и WebAssembly в одном потоке браузера, с минимумом внешних зависимостей. Частота обновления сцены — не ниже 30 кадров в секунду даже на машинах среднего уровня. Метод устойчив к типичным условиям любительской съёмки: непостоянному освещению, плавающему расстоянию от камеры, частичным перекрытиям рук друг другом или корпусом.

Выбор стека

Каркас одностраничного приложения — React. Для 3D-сцены применена связка Three.js [7] и React Three Fiber — декларативной обёртки, позволяющей описывать граф сцены средствами JSX; в первых прототипах использовался императивный Three.js, но необходимость вручную управлять жизненным циклом сцены быстро стала тормозить разработку. Формат аватара выбирался из FBX, glTF и VRM. Предпочтение отдано VRM [12] — открытому формату на базе glTF 2.0 со стандартизованным скелетом humanoid и строгой номенклатурой костей. Это принципиально: правила применения вращений становятся независимыми от конкретной модели — любой корректно оформленный VRM-аватар можно подменить без правок кода. Загрузка VRM выполняется плагином @pixiv/three-vrm. Извлечение ключевых точек поручено MediaPipe Holistic [8]: из доступных решений только она за один проход по кадру выдаёт сразу три набора ориентиров — 33 точки позы, 468 точек лица и 21 точку на каждую кисть. Пересчёт ориентиров в углы суставов выполняется библиотекой KalidoKit [10] — специализированным решателем для риггинга VRM-моделей по выходам MediaPipe. В качестве хранилища

глобального состояния выбран Zustand: проще Redux и допускает прямое чтение состояния из колбэков MediaPipe без React-хуков.

Структура конвейера

Конвейер представлен на рис. 1 и состоит из последовательных ступеней. Кадр с веб-камеры захватывается через MediaStream API в компоненте CameraWidget и отправляется в MediaPipe Holistic. Модель запущена с modelComplexity = 1 (компромисс между точностью и скоростью), порогами детекции и трекинга 0,7 и включённым smoothLandmarks = true для встроенного сглаживания позы. Получаемые на каждом кадре poseLandmarks, faceLandmarks, leftHandLandmarks и rightHandLandmarks публикуются в Zustand-хранилище. Компонент VRMAvatar забирает их в собственном callback'е и через решатели Pose.solve, Face.solve и Hand.solve из KalidoKit получает углы Эйлера для основных костей и значения блендшейпов лица. Полученные параметры применяются к скелету и мимике VRM-модели в хук-функции useFrame, после чего сцена рендерится контроллером React Three Fiber.

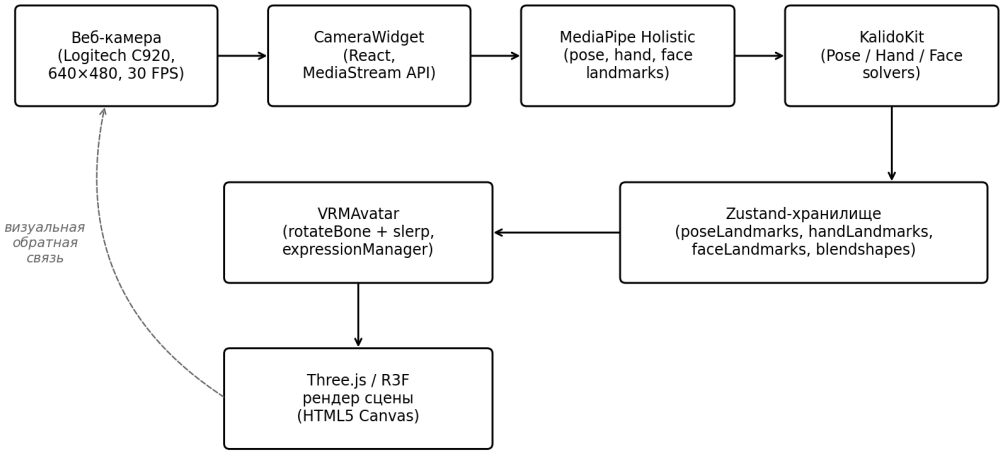

Рисунок 1. Структурная схема конвейера отзеркаливания

Структура входных данных кисти

Для построения правил применения вращений важна сама структура данных от MediaPipe. Ключевые точки кисти пронумерованы от 0 (запястье) до 20 (кончик мизинца) по схеме MediaPipe Hands, повторённой в Holistic (рис.

2): по четыре точки на палец плюс запястье — метакарпальные (5, 9, 13, 17), проксимальные межфаланговые PIP (6, 10, 14, 18), дистальные DIP (7, 11, 15, 19) и кончики (8, 12, 16, 20). Большой палец имеет иную анатомию и описывается точками 1–4. Координаты возвращаются в нормализованной форме: x и y — в долях ширины и высоты кадра, z — в тех же единицах с началом отсчёта на уровне запястья. Z-координата у MediaPipe носит оценочный характер и заметно менее точна, чем x и y, поскольку модель монокулярна и реконструирует глубину косвенно. Это ограничение сыграет роль при обсуждении результатов.

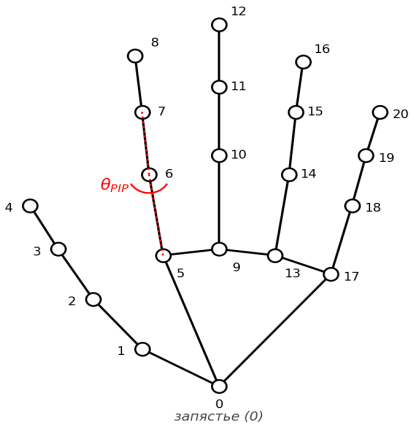

Рисунок 2. Нумерация 21 ключевой точки кисти, возвращаемой MediaPipe Holistic. Красным выделен угол сустава PIP указательного пальца

Функция применения вращений и сглаживание

Основной механизм наложения позы на модель — служебная функция rotateBone(boneName, value, slerpFactor). Она извлекает кость по humanoidBoneName через метод vrm.humanoid.getNormalizedBoneNode и обновляет её кватернион по формуле q = slerp(q_current, q_target, k). Здесь q_target получается из углов Эйлера от KalidoKit через THREE.Euler / THREE.Quaternion.setFromEuler, а k вычисляется как произведение базового коэффициента на delta-время кадра. Сферическая линейная интерполяция вместо прямого присваивания — принципиальный выбор: при

кратковременной потере руки в кадре MediaPipe возвращает «всплеск» в оценках, и при прямом присваивании это приводит к скачку кости и заметному дефекту модели. Slerp гасит этот эффект и одновременно даёт удобную ручку для баланса между отзывчивостью и устойчивостью.

Базовые коэффициенты подбирались эмпирически. Для корпуса, шеи, плеч, предплечий, бёдер и коленей используется delta·5 — мягкое сглаживание; в жестовой речи корпус движется плавно. Для запястий и фаланг пальцев применяется delta·18 — почти втрое более резвое значение: пальцы формируют быстрые переходы между конфигурациями, и сильное сглаживание размазывает артикуляцию до нечитаемости. На тесте с дактильной буквой «Б», требующей мгновенного разведения указательного и среднего, при delta·5 пальцы не успевали разойтись прежде, чем жест уже менялся. Дополнительно перед подачей в Hand.solve ориентиры кисти проходят отдельный экспоненциальный фильтр первого порядка smoothHand(current, prev, factor) с factor = 0,4: prev + (current − prev)·factor для каждой из 21 точки и каждой из координат x, y, z. Такая комбинация — фильтр по ориентирам плюс slerp по кватернионам — оказалась устойчивее любой из этих мер в отдельности. При потере руки в кадре буфер prev обнуляется.

Обработка мимики лица

Мимика в РЖЯ — не декоративное дополнение, а грамматически значимый канал: она различает вопрос и утверждение, маркирует условные конструкции, передаёт интенсивность и оттенки значения. Под мимику в проекте выделена отдельная ветвь конвейера, опирающаяся на решатель Face.solve и штатный механизм блендшейпов в спецификации VRM. Входными данными служит структура faceLandmarks из общего вывода Holistic — 468 точек, покрывающих контур лица, брови, нос, губы и область вокруг глаз; их распределение по анатомическим областям схематически показано на рис. 3.

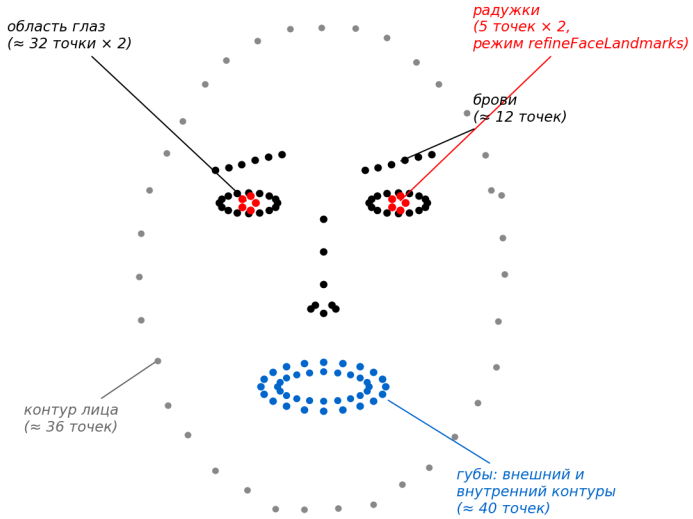

Рисунок 3. Схематическое распределение ключевых точек MediaPipe Face Mesh по анатомическим областям лица

По этим ориентирам Face.solve вычисляет две группы выходов. Первая — именованные блендшейпы в диапазоне [0; 1]: формы рта aa, ih, ee, oh, ou, соответствующие артикуляции пяти базовых гласных, и значения l/r открытости глаз для моргания. Вторая — углы поворота головы и координаты направления взгляда. Применение блендшейпов идёт через expressionManager: на каждом кадре вызывается setValue(name, value), причём не с прямым значением, а через вспомогательный lerpExpression — линейное сглаживание prev + (target − prev)·delta·12. Прямое присваивание давало визуальный дефект: оценки MediaPipe на отдельных кадрах «проскакивают», и аватар непроизвольно подмигивает или хмурится на один–два кадра. Коэффициент delta·12 подобран так, чтобы естественное моргание (около 150 мс) воспроизводилось без задержки, но всплески в один кадр (≈33 мс при 30 fps) подавлялись.

Моргание потребовало отдельного внимания. Наивный подход — считать открытость глаза как отношение вертикального и горизонтального расстояний между ориентирами века — даёт асимметричные моргания: правый и левый глаз закрываются не одновременно даже при естественном моргании. Виновато то, что при повороте головы один из глаз частично

перекрыт и оценка его открытости становится менее надёжной. KalidoKit решает это встроенно: Face.solve принимает параметры smoothBlink: true и blinkSettings с порогами [0,25; 0,75], при которых сырые значения коррелируются с учётом угла головы и биологической синхронности моргания. После активации этих параметров асимметрия исчезла. Направление взгляда реализовано через объект lookAtTarget класса THREE.Object3D, присоединённый к камере сцены. На каждом кадре его позиция обновляется по координатам зрачков: lookAtDestination.set(−2·pupil.x, 2·pupil.y, 0), и далее плавно подтягивается через lerp с коэффициентом delta·5; результирующий объект присваивается vrm.lookAt.target — плагин @pixiv/three-vrm сам вычисляет нужные углы поворота глаз. В формате VRM прямое направление взгляда соответствует оси −Z в локальной системе координат модели, а MediaPipe выдаёт координаты с противоположной ориентацией; без явного знака «−» перед pupil.x аватар стабильно смотрел в противоположную от пользователя сторону. Повороты головы применяются к кости neck той же rotateBone с коэффициентом delta·5 и масштабирующим вектором (0,7; 0,7; 0,7); корпус — аналогично от riggedPose.Spine с масштабированием 0,3.

Зеркальная инверсия и цикл рендеринга

Видеопоток с фронтальной веб-камеры в исходном виде даёт правую руку пользователя в левой части кадра и наоборот — это следствие того, что камера снимает пользователя «как видит её собеседник». При прямом применении ориентиров к аватару возникал устойчивый диссонанс: пользователь поднимает правую руку — аватар поднимает «левую» с точки зрения наблюдателя. Решение двухступенчатое. Превью видеопотока на экране отражается CSS-преобразованием transform: scaleX(-1) — пользователь видит привычное «зеркальное» отражение. В момент применения ориентиров leftHandLandmarks и rightHandLandmarks подаются в Hand.solve с инвертированной стороной, а результаты сохраняются в riggedRightHand и

riggedLeftHand «перекрёстно». По отзывам участников тестирования, такая схема снижает когнитивную нагрузку: пользователь не тратит усилия на мысленную инверсию сторон.

React Three Fiber предоставляет хук useFrame, вызываемый на каждом кадре с текущим delta. Внутри useFrame VRMAvatar проверяет наличие данных в riggedFace, riggedPose, riggedLeftHand и riggedRightHand, применяет rotateBone ко всем релевантным костям скелета, обновляет expressionManager и lookAtTarget, после чего обязательно вызывает vrm.update(delta) — этот шаг запускает внутренние обновления плагина (скининг, spring-bones, обновление взгляда). Пропуск vrm.update приводит к формально корректному обновлению костей при визуально замороженной модели. Дополнительно реализована ветка возврата в Т-позу: если ни один из rigged-буферов не заполнен, цикл проходит по всем отслеживаемым костям и плавно возвращает их к ориентации тождественного кватерниона (slerp с коэффициентом delta·6); блендшейпы аналогично сбрасываются в ноль. Это смотрится естественнее мгновенного «прыжка» аватара в нулевую позу.

Результаты исследования и их обсуждение

Метод реализован как одностраничное веб-приложение, окно которого включает три ключевые области: превью видеопотока с камеры (выводится зеркально), 3D-сцену с VRM-аватаром, повторяющим движения пользователя, и панель управления — включением камеры, выбором модели аватара, переключением сцены и пресета освещения. Общий вид интерфейса в рабочем режиме показан на рис. 4. Исходный код содержит около 7,5 тысяч строк JavaScript / JSX; ключевой для настоящей статьи компонент VRMAvatar занимает порядка 470 строк.

Рисунок 4. Пользовательский интерфейс приложения в режиме отзеркаливания

Тестирование выполнялось на рабочей станции: Intel Core i7-12700, NVIDIA GeForce RTX 3060, 32 ГБ ОЗУ; браузер — Google Chrome 122. Видеопоток захватывался камерой Logitech C920 в разрешении 640×480 при 30 fps. Параметры рендеринга React Three Fiber: dpr = [1, 2], antialias = true, ограничение по частоте обновления — частотой экрана 60 Гц. Производительность оценивалась в двух режимах: при работающем конвейере и при искусственно отключённом KalidoKit (аватар при этом неподвижен; замер позволяет оценить долю решателя в общей нагрузке). Результаты усреднены по 60-секундной серии после прогрева и сведены в таблицу 1.

Таблица 1. Производительность контура отзеркаливания.

Режим	**Средняя частота кадров, FPS**	**Порог плавности, FPS**
Полный конвейер (MediaPipe Holistic + KalidoKit + рендеринг VRM)	58	30
Конвейер с отключённым KalidoKit	62	30

Метод уверенно укладывается в требование плавности: даже в полном режиме средняя частота почти вдвое превышает пороговые 30 FPS. Разница между режимами — около 4 FPS — означает, что собственно кинематические расчёты KalidoKit занимают менее 7 % общей нагрузки; основная стоимость приходится на инференс MediaPipe Holistic. Это согласуется с наблюдениями других авторов: Holistic при modelComplexity = 1 потребляет порядка 30–40 мс процессорного времени на кадр [8].

Качество отзеркаливания оценивалось субъективно. Пятеро участников выполняли перед камерой набор движений: приветствие, кивок согласия и отрицательное покачивание головой, поочерёдный подъём правой и левой руки, сжатие и разжатие кулака, отдельные конфигурации пальцев (буквы «А», «Б», «В» русского дактиля), а также мимические упражнения: моргание одним и обоими глазами, подъём бровей, открытие рта в положениях, соответствующих гласным «а», «и», «о», поворот головы с фиксацией взгляда. По каждому аспекту выставлялась оценка по пятибалльной шкале. Усреднённые значения: 3,8 за корпус, 3,6 за плечевой пояс, 3,2 за пальцы, 3,5 за формы рта, 4,0 за моргание и 3,7 за направление взгляда и повороты головы.

Наиболее частые замечания касались артикуляции пальцев: периодическое «проскакивание» пальцев через физически невозможные положения при быстрых переходах между конфигурациями (особенно в букве «В» дактиля); недостаточно убедительная фиксация согнутых и разогнутых положений. Второе — недостаточная выразительность крайних положений губ: формы oh и ou визуально слабо различались, что связано с особенностями конкретной VRM-модели и её блендшейпов, а не с ветвью обработки. Ограничения по пальцам объясняются точностью оценки z-координаты в монокулярных моделях; на это же ограничение указывают и другие исследователи [3, 4].

Дополнительно проверялась устойчивость к снижению освещённости. Ниже 200 лк увеличивалась шумовая компонента в координатах ориентиров рук — амплитуда дрожания пальцев доходила до 5–7 градусов на отдельных

суставах. Применённый межкадровый фильтр с factor = 0,4 заметно её снижал, хотя полностью не устранял. Ниже 80 лк MediaPipe периодически терял кисть в кадре, возвращая пустые массивы; в этих случаях VRMAvatar сохраняет последнее известное положение костей. Участники предпочитали именно такое поведение мгновенному сбросу аватара в нулевую Т-позу, который реализовывался в одном из ранних прототипов.

Отдельный результат — организация кода, удобная для дальнейшего развития. Параметры сглаживания собраны в явном виде в одном месте, кости адресуются исключительно через humanoidBoneName (метод совместим с любым корректно оформленным VRM-аватаром), логика отзеркаливания инкапсулирована в единственном компоненте VRMAvatar; замена MediaPipe Holistic на более современный решатель потребует изменений только в CameraWidget и в формате структур, попадающих в Zustand-хранилище.

Выводы

Разработан и программно реализован метод интерактивного отзеркаливания движений пользователя на 3D-аватаре формата VRM, работающий полностью в среде браузера по данным с одной веб-камеры. Метод построен на связке MediaPipe Holistic и KalidoKit, интегрированных в React-приложение поверх Three.js и React Three Fiber. Ключевыми алгоритмическими составляющими стали: сферическая линейная интерполяция поворотов костей с дифференцированными коэффициентами для разных анатомических групп (delta·5 для корпуса и конечностей, delta·18 для запястий и фаланг пальцев); межкадровое экспоненциальное сглаживание ориентиров рук с factor = 0,4 перед подачей в Hand.solve; двухступенчатая зеркальная инверсия; настройка smoothBlink в Face.solve, подавляющая асимметричные моргания при поворотах головы.

Тестирование подтвердило работоспособность в реальном времени: средняя частота кадров — 58 FPS, что почти вдвое превышает порог плавности. Субъективные оценки качества — 3,8 за корпус и 3,2 за пальцы по

пятибалльной шкале — соответствуют уровню, приемлемому для учебных и ассистивных сценариев. Выявленные ограничения сосредоточены в основном в области артикуляции пальцев и связаны с точностью оценки глубины в монокулярных моделях компьютерного зрения. Направления дальнейших исследований: переход на MediaPipe Tasks Hand Landmarker с раздельной обработкой рук; разработка лёгкого нейросетевого постфильтра конфигураций пальцев, сводящего наборы ориентиров к физически допустимым; оптимизация рендеринга для мобильных устройств; интеграция метода в комплексные учебные приложения для людей с нарушениями слуха в русле подходов, развиваемых в работах [1, 5, 6].

Литература

1. Ашрафи А., Мохначев В. С., Филиппович Ю. Н. Конвертация видео в анимацию с использованием технологии видеотрекинга для разработки мультиязычного словаря жестового языка // ИИАСУ’24. – 2025. – С. 146–151.
2. Всемирная организация здравоохранения. Глухота и потеря слуха: информационный бюллетень [Электронный ресурс]. // Всемирная организация здравоохранения. – URL: https://www.who.int/ru/news-room/fact-sheets/detail/deafness-and-hearing-loss (дата обращения: 15.04.2026).
3. Капустин В. А., Мухин А. С., Петров Д. А. и др. SLOVO: Russian Sign Language Dataset // Proceedings of the International Conference on Computer Vision Systems (ICVS). – 2023. – P. 145–152.
4. Королькова О. О. Проблемы классификаций жестов русского жестового языка // Вестник Новосибирского государственного педагогического университета. – 2016. – № 4. – С. 106–120.
5. Филиппович Ю. Н., Тукаев К. А., Адейкин С. А., Галактионова Д. С. Технология параметризированного синтеза жестов и их последовательностей на основе анимированных 3D-моделей // Системный администратор. – 2014. – № 7. – С. 126–131.

6. Харламенков А. Е., Ашрафи А., Мохначёв В. С., Филиппович Ю. Н. Разработка мультиязычного словаря жестового языка с обратной связью через аватара в реальном времени с отслеживанием движений // Современное профессиональное образование. – 2025. – № 7. – С. 193–197.
7. Cabello R. Three.js: JavaScript 3D Library [Электронный ресурс]. – URL: https://threejs.org/ (дата обращения: 10.04.2026).
8. Lugaresi C., Tang J., Nash H. et al. MediaPipe: A Framework for Building Perception Pipelines // arXiv preprint arXiv:1906.08172. – 2019. – P. 1–9.
9. Potkin O., Philippovich A. Hand Gestures Detection, Tracking and Classification Using Convolutional Neural Network // Analysis of Images, Social Networks and Texts. AIST 2019 / W. van der Aalst et al. (eds.). – 2020. – P. 263–269. – DOI: 10.1007/978-3-030-39575-9_27.
10. Richards D. Kalidokit: Blendshape and Kinematics Calculator for Mediapipe / Tensorflow.js Models [Электронный ресурс]. – URL: https://github.com/yeemachine/kalidokit (дата обращения: 10.04.2026).
11. Shumilov A., Philippovich A. Gesture-based animated CAPTCHA // Information and Computer Security. – 2016. – P. 242–254. – DOI: 10.1108/ICS-12-2014-0082.
12. VRM Consortium. VRM: 3D Humanoid Avatar File Format for VR [Электронный ресурс]. – URL: https://vrm.dev/en/ (дата обращения: 10.04.2026).

Literature

1. Ashrafi A., Mokhnachev V. S., Philippovich Yu. N. Konvertatsiya video v animatsiyu s ispol’zovaniyem tekhnologii videotrekinga dlya razrabotki mul’tiyazychnogo slovarya zhestovogo yazyka [Video-to-animation conversion using video tracking technology for the development of a multilingual sign language dictionary] // IIASU’24. – 2025. – P. 146–151.
2. World Health Organization. Deafness and hearing loss: Fact sheet [Electronic resource]. – URL: https://www.who.int/news-room/fact-sheets/detail/deafness-and-hearing-loss (дата обращения: 15.04.2026).

3. Kapustin V. A., Mukhin A. S., Petrov D. A. et al. SLOVO: Russian Sign Language Dataset // Proceedings of the International Conference on Computer Vision Systems (ICVS). – 2023. – P. 145–152.
4. Korolkova O. O. Problemy klassifikatsiy zhestov russkogo zhestovogo yazyka [Problems of classification of Russian sign language gestures] // Bulletin of Novosibirsk State Pedagogical University. – 2016. – № 4. – P. 106–120.
5. Philippovich Yu. N., Tukaev K. A., Adeykin S. A., Galaktionova D. S. Tekhnologiya parametrizirovannogo sinteza zhestov i ikh posledovatel'nostey na osnove animirovannykh 3D-modeley [Technology of parameterized synthesis of gestures and their sequences based on animated 3D models] // Sistemnyy administrator [System Administrator]. – 2014. – № 7. – P. 126–131.
6. Kharlamenkov A. E., Ashrafi A., Mokhnachev V. S., Philippovich Yu. N. Razrabotka mul'tiyazychnogo slovarya zhestovogo yazyka s obratnoy svyaz'yu cherez avatara v real'nom vremeni s otslezhivaniyem dvizheniy [Development of a multilingual sign language dictionary with avatar-based real-time motion-tracking feedback] // Sovremennoye professional'noye obrazovaniye [Modern Professional Education]. – 2025. – № 7. – P. 193–197.
7. Cabello R. Three.js: JavaScript 3D Library [Electronic resource]. – URL: https://threejs.org/ (дата обращения: 10.04.2026).
8. Lugaresi C., Tang J., Nash H. et al. MediaPipe: A Framework for Building Perception Pipelines // arXiv preprint arXiv:1906.08172. – 2019. – P. 1–9.
9. Potkin O., Philippovich A. Hand Gestures Detection, Tracking and Classification Using Convolutional Neural Network // Analysis of Images, Social Networks and Texts. AIST 2019 / W. van der Aalst et al. (eds.). – 2020. – P. 263–269. – DOI: 10.1007/978-3-030-39575-9_27.
10. Richards D. Kalidokit: Blendshape and Kinematics Calculator for Mediapipe / Tensorflow.js Models [Electronic resource]. – URL: https://github.com/yeemachine/kalidokit (дата обращения: 10.04.2026).

11. Shumilov A., Philippovich A. Gesture-based animated CAPTCHA // Information and Computer Security. – 2016. – P. 242–254. – DOI: 10.1108/ICS-12-2014-0082.
12. VRM Consortium. VRM: 3D Humanoid Avatar File Format for VR [Electronic resource]. – URL: https://vrm.dev/en/ (дата обращения: 10.04.2026).