УДК 004.946
doi:10.31799/1684-8853-2025-3-49-58
EDN: SKBXGY

Научные статьи
Articles

Оптимизация размещения виртуальных объектов в расширенной реальности с использованием динамического программирования

М. В. Алпатова[а], *канд. техн. наук, доцент, orcid.org/0000-0003-3018-9601, m.v.alpatova@yandex.ru*
Ю. В. Рудяк[б], *доктор физ.-мат. наук, профессор, orcid.org/0000-0001-7886-0455*
[а]*МИРЭА — Российский технологический университет, Вернадского пр., 78, Москва, 119454, РФ*
[б]*Московский политехнический университет, Б. Семеновская ул., 38, Москва, 107023, РФ*

Введение: *развитие технологий расширенной реальности предъявляет высокие требования к точности размещения виртуальных объектов для обеспечения оптимального пользовательского опыта. В условиях динамично изменяющихся сцен и ограниченных вычислительных ресурсов актуальной остается задача корректной интеграции новых объектов в уже оптимизированную композицию.* ***Цель:*** *разработать и экспериментально оценить алгоритм размещения виртуальных объектов в пространстве расширенной реальности с учетом вычислительной эффективности.* ***Результаты:*** *предложен модифицированный алгоритм двумерной оптимальной расстановки, основанный на принципах динамического программирования и локальных корректировках. Алгоритм включает предварительную сортировку объектов по размеру, последовательное размещение с минимизацией дисбаланса зон комфортного восприятия и локальный перерасчет позиций, что обеспечивает интеграцию новых элементов без нарушения исходной структуры сцены. Экспериментальная проверка показала, что алгоритм эффективно размещает объекты с учетом их физических размеров и зон комфорта. Временные характеристики демонстрируют линейное увеличение времени выполнения при небольшом количестве объектов (до 35), а при дальнейшем увеличении числа объектов (в диапазоне 35÷70 и выше) наблюдается значительное ускорение темпа роста времени выполнения, особенно на мобильных устройствах. Регрессионный анализ выявил высокую степень корреляции между числом объектов и временем выполнения для персональных компьютеров и устройств iOS, в то время как Android-платформы демонстрируют меньшую масштабируемость.* ***Практическая значимость:*** *разработанный метод доказал свою применимость в реальных сценариях расширенной реальности, позволяя снизить вычислительную нагрузку и повысить адаптивность систем расширенной реальности.* ***Обсуждение:*** *результаты исследования открывают перспективы дальнейшей оптимизации алгоритма за счет применения параллельных вычислений и аппаратного ускорения, а также расширения его функционала для работы в трехмерных средах.*

Ключевые слова *— расширенная реальность, виртуальные объекты, оптимальное размещение, динамическое программирование, вычислительная эффективность, буферные зоны, перцептивное состояние, алгоритмы размещения, мобильные устройства, регрессионный анализ.*

Для цитирования: Алпатова М. В., Рудяк Ю. В. Оптимизация размещения виртуальных объектов в расширенной реальности с использованием динамического программирования. *Информационно-управляющие системы*, 2025, № 3, с. 49–58. doi:10.31799/1684-8853-2025-3-49-58, EDN: SKBXGY

For citation: Alpatova M. V., Rudyak Y. V. Optimization of virtual object placement in extended reality using dynamic programming. *Informatsionno-upravliaiushchie sistemy* [Information and Control Systems], 2025, no. 3, pp. 49–58 (In Russian). doi:10.31799/1684-8853-2025-3-49-58, EDN: SKBXGY

Введение

Дополненная реальность постоянно развивается с высокой скоростью и используется во многих областях деятельности, начиная от ориентирования на местности и заканчивая педагогикой, от медицины до промышленного дизайна [1, 2]. Сложной задачей для расширенной реальности (Extended Reality, XR) может стать обеспечение точного и естественного размещения виртуальных объектов в пространстве. Отсутствие ориентиров в окружающей среде затрудняет манипуляции пользователя с виртуальными объектами и тем самым ухудшает восприятие и погружение [3]. Ошибки позиционирования могут даже вызывать когнитивный диссонанс, что пагубно сказывается на пользовательском опыте.

Проблема размещения объектов в XR зависит от нескольких факторов, включая точность позиционирования, адаптацию к изменяющимся условиям и удобство восприятия [4, 5]. Такие подходы, как алгоритмы машинного обучения, повышают производительность отслеживания, но требуют больших вычислительных мощностей [6]. В то же время адаптация методов организации пространства способствует более эффективному восприятию XR-контента, но имеет свои ограничения [7]. Исследования также показывают, что факторы окружающей среды, такие как меняющиеся условия освещения и движение объектов, оказывают заметное влияние на точность размещения [3].

В предыдущих исследованиях авторов [5, 8], посвященных размещению объектов в двумер-

ном пространстве с учетом их размеров и «зон комфорта», рассматривались только условия размещения объектов в идеальных случаях изолированного размещения, но не то, как вставить новые объекты в существующую конфигурацию размещения.

Актуальность поиска корректной вставки виртуальных объектов обусловлена необходимостью повышения вычислительной производительности и устойчивости алгоритмов в условиях динамического обновления среды, когда новые объекты должны гармонировать с уже размещенными в их пространстве объектами, чтобы их позиционирование в пространстве не было нарушено в результате их размещения. Гипотеза исследования заключается в том, что алгоритм, корректно вставляющий новые виртуальные объекты в существующее пространство, значительно повысит производительность и при этом не ухудшит качество размещения в пространстве. Цель данной работы — представить и эмпирически исследовать прагматический подход к размещению виртуальных объектов в существующую конфигурацию XR-пространства оптимальным способом с особым акцентом на тестирование вычислительной производительности алгоритма.

Связь с предыдущими исследованиями

Анализ исследований в области XR подтверждает, что современные подходы обычно сосредоточены на статическом размещении объектов, предотвращении столкновений или динамическом отслеживании. Однако лишь немногие исследования затрагивают проблему оптимального дополнения уже существующей композиции виртуальных объектов. В предыдущем исследовании был представлен алгоритм двумерного размещения, основанный на минимизации целевой функции, который позволял учитывать размеры объектов и окружающие их буферные зоны, обеспечивая равномерное распределение коэффициентов комфортности по всем сторонам каждого объекта [5, 8]. Между тем значительная часть международных исследований посвящена либо эффективному представлению 3D-сцен с использованием многомасштабных воксельных структур, либо динамическому прогнозированию столкновений и оптимизации размещения маркеров с помощью методов глубокого обучения и обучения с подкреплением [9, 10].

Исследования, посвященные структурированию 3D-объектов для XR на смартфонах, демонстрируют высокую вычислительную эффективность, проявляющуюся в ускоренной обработке данных и оптимальном использовании памяти, что способствует повышению общей производительности мобильных XR-систем. Однако данные подходы не решают задачу интеграции новых объектов в уже созданную композицию с учетом заранее определенных зон комфорта, что остается актуальной проблемой для обеспечения качественного пользовательского опыта [11]. Аналогично исследования мультимодальных методов предсказания столкновений и механизмов самоанализа направлены в первую очередь на обеспечение безопасности взаимодействия объектов в динамических средах, но не учитывают гармоничную интеграцию объектов в предварительно оптимизированное виртуальное пространство [6].

Методы оптимизации размещения, основанные на концепции виртуальных осей и нелинейном программировании, позволяют автоматически определять доступные промежуточные позиции объектов, что крайне важно при формировании композиции. Однако их применение в основном ограничено статическими сценариями [12]. Недавние исследования, использующие глубокое обучение и обучение с подкреплением для динамической корректировки позы, демонстрируют высокую адаптивность к изменениям сцены. Тем не менее эти подходы в первую очередь направлены на обеспечение безопасности и повышение информативности, а не на доработку уже структурированной среды для повышения эффективности вычислений [1, 13].

Таким образом, в настоящее время существует заметное несоответствие между имеющимися способами и методами добавления новых объектов в предварительно оптимизированную композицию с сохранением баланса в комфортных пространствах. В данной статье представлен алгоритм, который не только учитывает физические размеры объектов и их комфортные пространства, но и вставляет их в существующую композицию в оптимальных позициях. В настоящей работе представлен принцип, сочетающий статические методы оптимизации с механизмами адаптации для решения проблемы размещения объектов в XR-пространствах при сохранении высокой производительности системы и качества пользовательского опыта [14, 7, 15].

Постановка задачи и математическая модель

Задачи оптимального размещения объектов широко изучаются в области упаковки (Packing Problem), где разрабатываются эвристические и динамические методы минимизации занимаемой площади и предотвращения пересечений.

Эффективные алгоритмы включают возможность вращения объектов и учитывают ограничения пространства [16]. Вместе с тем для работы в реальном времени критично учитывать динамическое добавление новых объектов, что требует локального пересчета лишь ближайших элементов сцены. Подходы, использующие пошаговую оптимизацию позиций компонентов в ограниченном пространстве, демонстрируют высокую эффективность динамического размещения [17]. Эти принципы подтверждают выбор предложенной стратегии оптимизации сцены в XR, позволяя минимизировать изменения в уже размещенной конфигурации. Рассматриваемый далее подход основывается на ранее предложенном методе, детализированном в работе «Оптимальная 2D расстановка виртуальных объектов в физическом пространстве в приложениях дополненной реальности» [8]. В этом методе основное внимание уделяется последовательному заполнению плоскости с минимизацией разрывов между объектами и обеспечением перцептивной комфортности. Указанный подход включает несколько ключевых этапов, каждый из которых вносит вклад в общий процесс размещения.

Рассматривается задача размещения виртуальных объектов на двумерной плоскости, на которой уже присутствуют ранее установленные элементы. Эти объекты представляют собой набор прямоугольных областей, каждая из которых описывается собственными размерами — шириной w_i и высотой h_i, а также буферными зонами D_i^-, D_i^+, Z_i^-, Z_i^+. Буферные зоны служат для поддержания комфортного перцептивного состояния и определяют минимально допустимые расстояния от объекта до соседей или границ доступного пространства.

Плоскость, на которой происходит размещение, имеет фиксированные размеры W по горизонтали и H по вертикали. Объекты должны удовлетворять ограничению нахождения внутри этих границ:

$$x_i - D_i^- \geq 0,\ x_i + w_i + D_i^+ \leq W;$$
$$y_i - Z_i^- \geq 0,\ y_i + h_i + Z_i^+ \leq H. \quad (1)$$

Кроме того, допускается, чтобы буферные зоны объектов пересекались, если это позволяет минимизировать целевую функцию перцептивного состояния. Для каждого объекта вводится понятие односторонней комфортности K^- и K^+, которая рассчитывается на основании расстояний до соседей или границ. Если буферные зоны двух объектов пересекаются, разница в комфортности учитывается в совокупной целевой функции сцены:

$$K_\Sigma = \sum_{i=1}^{n} \left(K_i^- - K_i^+ \right)^2, \quad (2)$$

где K_i^- и K_i^+ рассчитываются для каждой стороны объекта с использованием буферных расстояний и текущих позиций.

Для гарантии вычислительной эффективности алгоритма ограничивается общий объем перерасчетов. Это достигается путем локализации операций внутри партий объектов, которые формируются динамически, и учета только тех областей пространства, которые непосредственно затрагиваются добавлением новых объектов. Таким образом, каждая итерация перерасчетов минимизирует изменения в положении уже установленных элементов.

Базовый алгоритм размещения

Перед началом размещения объекты сортируются по убыванию их длины h_i. Такая сортировка позволяет первым размещать объекты с максимальными размерами, минимизируя их возможные конфликты с соседями. Формально это условие можно выразить как упорядочение множества объектов N:

$$N' = \{n_1, n_2, \ldots, n_k\}, \text{ где } h_i \geq h_{i+1} \text{ для всех } i. \quad (3)$$

Для упорядочения объектов на плоскости используется стратегия челночного хода. На первом этапе объекты располагаются слева направо, начиная от нижнего левого угла плоскости. После достижения правой границы направление меняется на противоположное, и следующая строка заполняется справа налево. Этот метод помогает снизить избыточные смещения между объектами в разных строках и способствует более плотному заполнению плоскости.

Оптимальное положение каждого объекта в строке определяется из условия минимума целевой функции (4). Целевая функция оптимизации минимизирует разницу в комфортности:

$$\min \sum_{j \in S \cup N'} (K_j^- - K_j^+)^2, \quad (4)$$

где K_j^- и K_j^+ рассчитываются на основании расстояний до ближайших соседей и границ строки:

$$K_j^- = \frac{x_j - (x_{left} + w_{left} + D_{left}^+)}{D_j^-};$$
$$K_j^+ = \frac{(x_{right} - D_{right}^- - w_j - x_j)}{D_j^+}. \quad (5)$$

Переменные x_{left} и x_{right} определяют границы соседей слева и справа соответственно. Для объектов на краях строки значения x_{left} или x_{right} принимаются равными границам плоскости 0 или W.

Таким образом, базовый подход направлен на последовательное упорядоченное размещение объектов с учетом их размеров и ограничений на буферные зоны. Этот процесс задает начальную структуру размещения, на основе которой в дальнейшем производятся уточнение и модификации.

Модифицированный алгоритм и динамическое программирование

Модифицированный метод решения основывается на базовом подходе оптимальной 2D-расстановки, но адаптируется для учета уже установленных объектов и их влияния на новые элементы. В отличие от исходного алгоритма, новая версия включает этапы перерасчета партий и локальной корректировки для минимизации конфликта между новыми и ранее размещенными элементами.

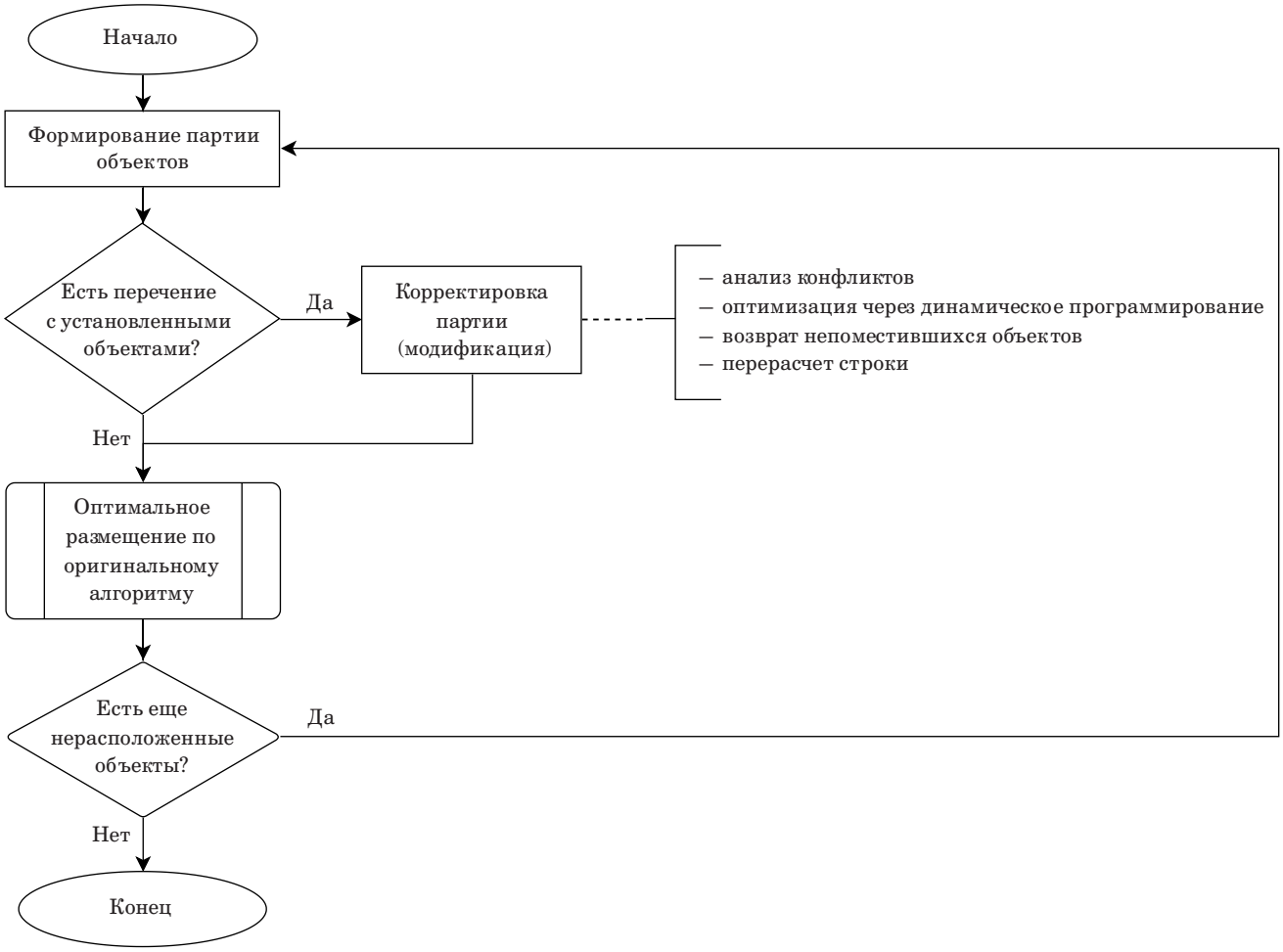
В общем виде описываемый процесс представлен на рис. 1.

На первом этапе формируются партии объектов, которые подлежат размещению. Каждая партия $B \subseteq N$ строится таким образом, чтобы ее суммарная ширина, включая буферные зоны, удовлетворяла условию

$$\sum_{j \in B} \left(w_j + D_j^- + D_j^+ \right) \le W. \qquad (6)$$

Этот процесс организуется на основе существующего алгоритма, но с учетом ограничений, задаваемых уже установленными объектами S. При этом структура партии оптимизируется для минимизации внутренних конфликтов между ее элементами.

После формирования партии проводится проверка на пересечения между объектами из B и S.



■ ***Рис. 1.*** Блок-схема модифицированной версии оригинального алгоритма
■ ***Fig. 1.*** Block diagram of the modified version of the original algorithm

Пересечение фиксируется, если выполнены условия

$$(x_j + w_j + D_j^+ > x_i - D_i^-) \wedge (x_j - D_j^- < x_i + w_i + D_i^+) \wedge$$
$$\wedge (y_j + h_j + Z_j^+ > y_i - Z_i^-) \wedge (y_j - Z_j^- < y_i + h_i + Z_i^+),$$
$$\forall j \in B,\ i \in S. \quad (7)$$

Эти проверки позволяют выявить зоны конфликта и перейти к следующему этапу — локальной корректировке.

Для каждого пересекающегося j-го объекта из партии S определяется его положение относительно двух соседей — одного слева, другого справа. Эти соседи выбираются из текущей партии B, причем критерий выбора формулируется как

$$x_{left} + w_{left} + D_{left}^+ \le x_j - D_j^-;$$
$$x_{right} - D_{right}^- \ge x_j + w_j + D_j^+. \quad (8)$$

На основе этой информации рассчитывается новая координата x_j, минимизирующая дисбаланс в комфортности:

$$\min |(x_j - x_{left} - w_{left} - D_{left}^+) -$$
$$-(x_{right} - x_j - w_j - D_j^+)|. \quad (9)$$

Здесь используется динамическое программирование (DP), чтобы найти подмножество объектов, которое максимально эффективно заполняет строку, не нарушая ее ограничений. Его использование играет важную роль в обеспечении вычислительной эффективности. Масштабирование и дискретизация входных параметров позволяют адаптировать задачу к ограниченным ресурсам, снижая общую сложность алгоритма. Это особенно важно при увеличении числа объектов или размера сцены, где точечный пересчет для каждой партии снижает избыточные вычисления. Подобные методы успешно применяются и в смежных задачах оптимального размещения с учетом внешних ограничений [18].

Алгоритм перераспределения начинается с объединения множества новых объектов N и уже установленных элементов S в единый массив. Каждый элемент помечается как «добавленный» или «существующий», что позволяет эффективно разделять операции на этапах проверки и перераспределения. Это объединение формализуется как расширенное множество

$$E = \{(s_i, false) \mid s_i \in S\} \cup \{(n_j, true) \mid n_j \in N\}. \quad (10)$$

Далее вычисляется максимальная допустимая длина строки, определяемая как

$$L_{\max} = W - \sum_{j \in B} (w_j + D_j^- + D_j^+), \quad (11)$$

где B — текущая партия новых объектов, уже добавленных в строку. Этот параметр служит ограничением для включения новых объектов и позволяет сохранить установленные элементы в их исходных позициях.

Для повышения точности и удобства применения алгоритма динамического программирования размеры объектов масштабируются. Пусть масштабный коэффициент равен α, тогда ширина каждого объекта w_j и допустимая длина строки $L_{\max}$ преобразуются в целые значения:

$$w_j^{scaled} = \lfloor \alpha w_j \rfloor;\ L_{\max}^{scaled} = \lfloor \alpha L_{\max} \rfloor. \quad (12)$$

Это позволяет использовать дискретную модель для последующего анализа.

Инициализируем массив достижимых сумм $dp[t]$, где t — текущее значение длины строки. Этот массив позволяет определить, какие размеры могут быть достигнуты с текущим набором объектов:

$$dp[t] = \max(dp[t],\ dp[t - w_j^{scaled}] + w_j^{scaled}),$$
$$\forall j \in B. \quad (13)$$

Одновременно с этим отслеживаются индексы выбранных объектов, что необходимо для восстановления их позиций на следующем этапе. Пусть $I[t]$ хранит множество индексов объектов, составляющих сумму t. Тогда обновление происходит по правилу

$$I[t] = \begin{cases} I[t],\ \text{если } dp[t] \ge dp[t - w_j^{scaled}] + w_j^{scaled}; \\ I[t - w_j^{scaled}] \cup \{j\} \text{ иначе.} \end{cases} \quad (14)$$

После завершения этапа DP те объекты, которые не удалось включить в текущую строку, возвращаются в общий пул. Это гарантирует, что они будут рассмотрены при формировании следующих партий. При этом оставшиеся элементы из текущей строки корректируются по координатам для минимизации конфликтов:

$$x_j = x_{prev} + w_{prev} + D_{prev}^+,\ \forall j \in B, \quad (15)$$

где x_{prev} — координата предыдущего объекта.

Завершающий этап перераспределения включает корректировку позиций объектов с учетом соседей и границ строки. Этот шаг минимизирует влияние новых элементов на уже установленные, сохраняя их перцептивное состояние.

Таким образом, модификация позволяет эффективно интегрировать новые объекты без значительного нарушения структуры сцены.

Финальным шагом для партии является уточнение позиций объектов на основании их скорректированных координат. Это включает восстановление порядка внутри партии и минимизацию влияния новых элементов на перцептивное состояние уже установленных объектов. После завершения всех операций партия размещается, и алгоритм переходит к следующей партии. Процесс повторяется до тех пор, пока не будут размещены все объекты или не будет исчерпано доступное пространство.

Экспериментальная проверка

Для оценки производительности предложенного алгоритма размещения виртуальных объектов был проведен анализ временных характеристик на различных процессорных архитектурах. Основное внимание уделялось снижению вычислительных затрат при увеличении количества объектов и оценке устойчивости алгоритма к выбросам в данных.

Для анализа экспериментально собранных данных использовался метод винсоризации для обработки выбросов, что позволило минимизировать влияние экстремальных значений на результаты расчетов. Данный метод, активно применяемый в статистическом моделировании и машинном обучении, доказал свою эффективность в области обработки данных с высокой вариативностью [19]. Кроме того, в контексте выборки по важности адаптивная винсоризация может уменьшить дисперсию и улучшить среднеквадратичную ошибку оценок, что выгодно для мобильных приложений, где вычислительная эффективность и точность имеют первостепенное значение [20]. Было показано, что метод визоризованных моментов обеспечивает надежную оценку параметров в моделях, работающих с усеченными и подвергнутыми цензуре данными, что актуально для мобильных приложений, обрабатывающих большие наборы данных с потенциальными выбросами [21].

В ходе исследования была проведена экспериментальная проверка производительности алгоритма размещения виртуальных объектов в ограниченном пространстве. Для этого использовался автоматизированный сценарий тестирования, выполняющий серию измерений на различных конфигурациях сцены и устройств. Тестирование проводилось на трех категориях платформ: персональных компьютерах (PC), устройствах на базе iOS и устройствах на базе Android. В качестве тестовых процессоров использовались Intel i7-12700, Intel i9-10900KF, Apple M1 Pro, A12 Bionic, A15 Bionic, A17 Pro, Snapdragon 845, Snapdragon 732G и Google Tensor GS101.

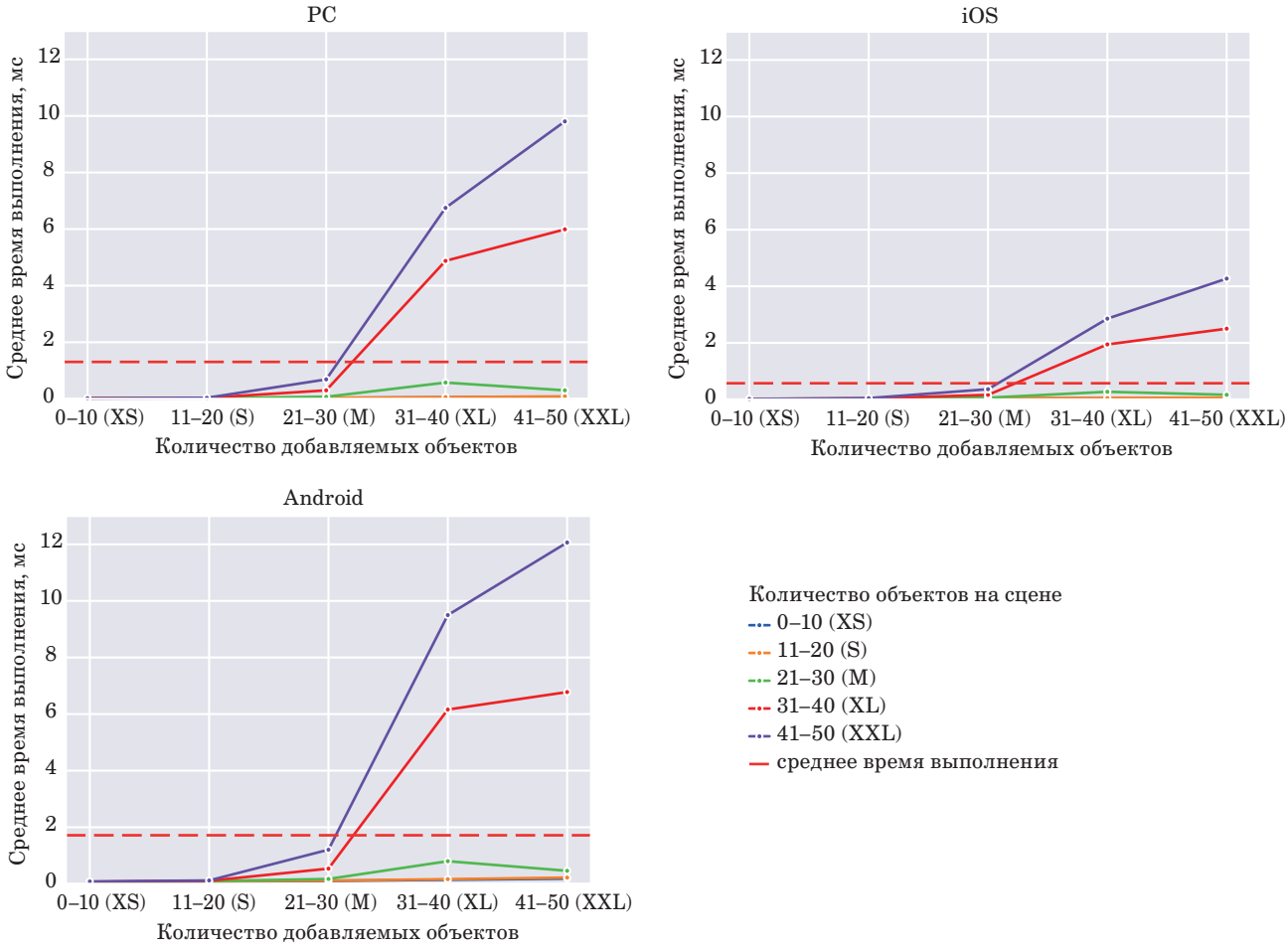
Тестирование включало измерение времени выполнения алгоритма при увеличении количества объектов в сцене от 0 до 50, с добавлением новых элементов от 1 до 50 в каждом запуске. Каждое испытание повторялось не менее 100 раз, что позволило сгладить возможные случайные флуктуации. Размер доступного пространства составлял 10×10 условных единиц, однако в процессе работы он динамически изменялся в зависимости от общего количества объектов, что позволяло учитывать влияние плотности размещения. Перед основными испытаниями выполнялся разогрев системы с фиксированными параметрами, чтобы исключить влияние предварительной компиляции кода и возможных особенностей кеширования.

Результаты тестирования показали, что при количестве объектов от 0 до 20 алгоритм демонстрирует линейный рост времени выполнения со средними значениями ниже 2 мс на всех платформах. Однако при превышении 30 объектов на сцене наклон кривой существенно возрастает, особенно на мобильных устройствах, что указывает на более высокие вычислительные затраты (рис. 2).

Для наглядного представления данных в диапазоне 0÷50 уже установленных объектов, к которым последовательно добавлялось от 0 до 50 новых объектов, было решено сгруппировать результаты по «корзинам» (bin), чтобы избежать избыточного загромождения графика. Каждая корзина (XS, S, M и т. д.) соответствует определенному диапазону количества объектов, отражая совокупность тестовых значений в этой группе. XS, S, M и другие аббревиатуры обозначают размер корзины: XS — очень маленькая, S — маленькая, M — средняя и т. д. Такой подход позволяет упростить визуализацию и указывает на тенденции изменения времени выполнения без потери ключевых особенностей распределения.

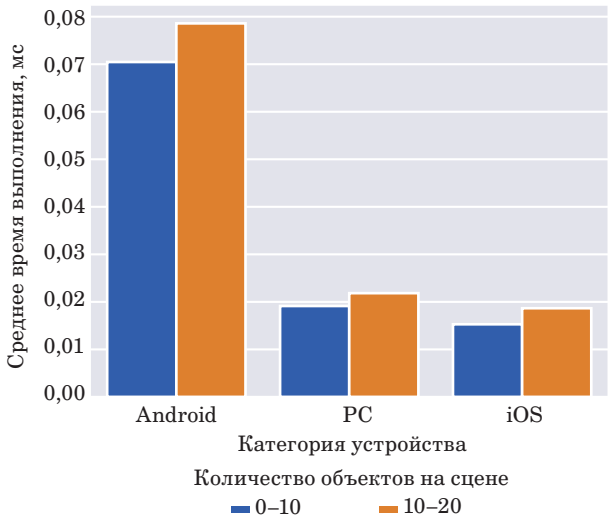
Персональные компьютеры продемонстрировали среднее время выполнения 1,30 мс, в то время как на iOS-устройствах (A12 Bionic, A15 Bionic, A17 Pro) этот показатель составил 0,57 мс, а на Android-устройствах (Snapdragon 732G, Snapdragon 845, Google Tensor GS101) — 1,72 мс. Таким образом, наиболее высокую производительность показала платформа iOS, а Android уступает ей почти в три раза.

Дополнительно был проведен анализ работы алгоритма в условиях небольших сцен (до 20 объектов), что более характерно для реальных прикладных задач (рис. 3). В этом диапазоне Android-



■ ***Рис. 2.*** Среднее время выполнения алгоритма в зависимости от количества объектов. Разница между 0–10 и 11–20 незначительна и может быть не видна из-за наложения линий

■ ***Fig. 2.*** Average execution time of the algorithm depending on the number of objects. The difference between 0–10 and 11–20 is insignificant and may not be visible due to overlapping lines

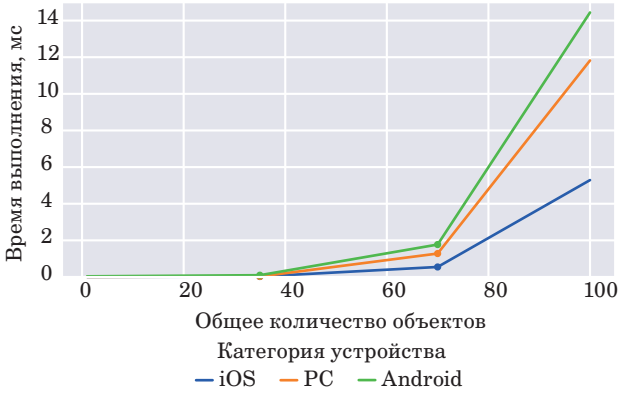


■ ***Рис. 3.*** Среднее время выполнения при малом числе объектов

■ ***Fig. 3.*** Average execution time with a small number of objects

устройства в среднем достигают 0,0745 мс, в то время как PC и iOS демонстрируют 0,0205 и 0,0170 мс соответственно. Несмотря на заметный разрыв между платформами, во всех случаях сохраняется линейная зависимость времени выполнения от числа объектов, что говорит о стабильной работе алгоритма при таком уровне нагрузки.

Для определения сложности алгоритма была построена модель кусочной линейной регрессии зависимости времени выполнения от общего числа объектов на сцене (рис. 4). Модель разбита на три диапазона: от 0 до 35 объектов, от 35 до 70 объектов и свыше 70 объектов. Анализ результатов показывает, что в первом диапазоне наблюдается относительно слабая линейная связь (коэффициенты корреляции r составляют 0,33 для iOS, 0,57 для PC и 0,37 для Android). Во втором диапазоне зависимость становится умеренной ($r \approx 0{,}41 \div 0{,}44$). Наиболее выраженная линейная зависимость фиксируется в третьем диапазоне: для PC и Android наблюдаются зна-



■ ***Рис. 4.*** Линейная зависимость времени выполнения от общего количества объектов

■ ***Fig. 4.*** Linear dependence of execution time on the total number of objects

чения r около 0,70–0,71, при этом наклон линии для Android (0,4223) значительно превышает соответствующий показатель для iOS (0,1581). Эти данные свидетельствуют о том, что при значительном увеличении числа объектов (свыше 70) время выполнения алгоритма растет быстрее, что особенно заметно для мобильных устройств, в первую очередь для платформы Android.

На основании полученных результатов можно заключить, что алгоритм эффективно работает для небольших сцен, где общее число объектов не превышает 35. В этом диапазоне время выполнения увеличивается очень слабо (коэффициенты наклона порядка 0,0006–0,0018 мс на объект для различных платформ), что свидетельствует о линейном характере зависимости и высокой производительности, особенно на персональных компьютерах и iOS-устройствах.

При переходе ко второму диапазону (от 35 до 70 объектов) наблюдается умеренное увеличение наклона регрессионной линии, а в третьем диапазоне (свыше 70 объектов) — значительно более выраженное. Так, для PC и Android коэффициенты наклона составляют 0,3514 и 0,4223 мс на объект соответственно, в то время как для iOS — 0,1581 мс на объект. Это указывает на снижение масштабируемости алгоритма на мобильных устройствах, особенно на Android-платформах с процессорами Snapdragon 732G и Google Tensor GS101.

Заключение

В данной работе представлен алгоритм идеального размещения объектов в XR-пространстве, позволяющий интегрировать новые элементы в уже оптимизированные композиции с учетом их физических размеров и зон комфорта. Разработанный подход объединяет классические методы статической оптимизации с динамическими механизмами локальной настройки, что позволяет минимизировать перерасчет позиций и сохранять исходную структуру сцены.

Экспериментальная проверка демонстрирует, что при размещении небольшого числа объектов (до 35) время выполнения алгоритма растет линейно, что делает его эффективным для большинства практических приложений. Однако при увеличении плотности объектов (свыше 70) наблюдается существенное ускорение роста вычислительных затрат. Результаты кусочного регрессионного анализа подтверждают, что, хотя общая зависимость остается линейной, увеличение наклона регрессионных линий в высоконагруженных диапазонах свидетельствует о необходимости дополнительной оптимизации для масштабных сцен.

Ключевым преимуществом предложенного алгоритма является возможность локального пересчета позиций объектов, что минимизирует влияние добавляемых элементов на уже существующую композицию и обеспечивает комфортное восприятие пользователем. Тем не менее в условиях высокой плотности объектов могут потребоваться дополнительные оптимизации, например реализация параллельных вычислений, использование аппаратного ускорения или разработка адаптивных стратегий распределения ресурсов для устройств с ограниченными вычислительными возможностями.

В целом результаты исследования подтверждают гипотезу о том, что стратегическая интеграция новых виртуальных объектов в предварительно оптимизированную композицию позволяет повысить вычислительную эффективность без ущерба для качества размещения и пользовательского опыта. Будущие исследования могут быть направлены на адаптацию алгоритма к трехмерным и динамическим условиям, что позволит еще больше раскрыть потенциал расширенной реальности в реальных приложениях.

Литература

1. **El Barhoumi N., Hajji R., Bouali Z., Ben Brahim Y., Kharroubi A.** Assessment of 3D models placement methods in augmented reality. *Applied Sciences*, 2022, vol. 12, no. 20, Article 10620. doi:10.3390/app122010620
2. **He X., Zhou K.** Display in the air: Balancing distraction and workload in AR glasses interfaces for driving navigation. *arXiv*, 2024. doi:10.48550/arxiv.2404.18357
3. **Ye K., Wu H., Tong X., Zhou K.** A real-time method for inserting virtual objects into neural radiance fields. *arXiv*, 2023. doi:10.48550/arxiv.2310.05837

4. **Nurhadi N., Stiawan D., Idris M. Y., Saparudin S.** Object tracking in augmented reality: Enhancement using convolutional neural networks. *Indonesian Journal of Electrical Engineering and Informatics (IJEEI)*, 2022, vol. 10, iss. 4. doi:10.52549/ijeei.v10i4.4104
5. **Алпатова М. В., Рудяк Ю. В.** Размещение нескольких виртуальных объектов в физическом пространстве в приложениях дополненной реальности. *Advanced Engineering Research (Rostov-on-Don)*, 2023, т. 23, № 2, с. 203–211. doi:10.23947/2687-1653-2023-23-2-203-211
6. **Zhu-Tian C., Chiappalupi D., Lin T., Yang Y., Beyer J., Pfister H.** RL-LABEL: A deep reinforcement learning approach intended for AR label placement in dynamic scenarios. *arXiv*, 2023. doi:10.48550/arxiv.2308.13540
7. **Xing Y., Liu Q., Wang J., Gomez-Zara D.** sMoRe: Enhancing object manipulation and organization in mixed reality spaces with LLMs and Generative AI. *arXiv*, 2024. doi:10.48550/arxiv.2411.11752
8. **Алпатова М. В., Рудяк Ю. В.** Оптимальная 2D-расстановка виртуальных объектов в физическом пространстве для приложений дополненной реальности. *Advanced Engineering Research (Rostov-on-Don)*, 2023, т. 23, № 4, с. 410–421. doi:10.23947/2687-1653-2023-23-4-410-421
9. **Xu Y., Wu Y., Zhou H.** Multi-scale voxel hashing and efficient 3d representation for mobile augmented reality. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, US, June 18–22, 2018, pp. 1586–15867. doi:10.1109/CVPRW.2018.00200
10. **Magassouba A., Sugiura K., Nakayama A., Hirakawa T., Yamashita T., Fujiyoshi H., Kawai H.** Predicting and attending to damaging collisions for placing everyday objects in photo-realistic simulations. *Advanced Robotics*, 2021, vol. 35, no. 12, pp. 787–799. doi:10.1080/01691864.2021.1913446
11. **Zanyk-McLean S., Kumar K., Navratil P.** 3D object positioning using differentiable multimodal learning. *arXiv*, 2023. doi:10.48550/arxiv.2309.03177
12. **Weiß M**. Optimal object placement using a virtual axis. *arXiv*, 2019. https://doi.org/10.48550/arXiv.1901.00168
13. **Wiethüchter R., Kalloori S., Chalumattu R.** Fast content placement and alignment in 3D scenes. *The International FLAIRS Conference Proceedings*, 2022, vol. 35. doi:10.32473/flairs.v35i.130686
14. **Wang H.** Haptic repurposing with GenAI: Transforming the tangible world into adaptive haptic interfaces in mixed reality. *arXiv*, 2024. doi:10.48550/arXiv.2406.07228
15. **Jia J., Elezovikj S., Fan H., Yang S., Liu J., Guo W., Tan C. C., Ling H.** Semantic-aware label placement for augmented reality in street view. *The Visual Computer*, 2021, vol. 37, no. 7, pp. 1805–1819. doi:10.1007/S00371-020-01939-W
16. **Zhao X., Rao Y., Qi P., Lyu Q., Yang P., Yu S.** An efficient constructive heuristic for the rectangular packing problem with rotations. *PloS One*, 2023, vol. 18, no. 12. doi:10.1371/journal.pone.0295206
17. **Douglas C., Huh J. S., Jun S. O., Kim I. Y.** Packing optimization of practical systems using a dynamic acceleration methodology. *Journal of Engineering and Applied Science*, 2024, vol. 71, no. 1. doi:10.1186/s44147-024-00426-6
18. **Chen D., Mo F., Chen Y., Zhang J., You X.** Optimization of ramp locations along freeways: A dynamic programming approach. *Sustainability*, 2022, vol. 14, no. 15, Article 9718. doi:10.3390/su14159718
19. **Boudt K., Todorov V., Wang W.** Robust distribution-based winsorization in composite indicators construction. *Social Indicators Research*, 2020, vol. 149, no. 2, pp. 375–397. doi:10.1007/s11205-019-02259-w
20. **Orenstein P.** Robust importance sampling with adaptive winsorization. *Bernoulli*, 2022, vol. 28, no. 4. doi:10.3150/21-bej1440
21. **Poudyal C., Zhao Q., Brazauskas V.** Method of winsorized moments for robust fitting of truncated and censored lognormal distributions. *North American Actuarial Journal*, 2024, vol. 28, no. 1, pp. 236–260. doi:10.1080/10920277.2023.2183869

UDC 004.946
doi:10.31799/1684-8853-2025-3-49-58
EDN: SKBXGY

Optimization of virtual object placement in extended reality using dynamic programming

M. V. Alpatova[a], PhD, Tech., Associate Professor, orcid.org/0000-0003-3018-9601, m.v.alpatova@yandex.ru
Y. V. Rudyak[b], Dr. Sc., Phys.-Math., Professor, orcid.org/0000-0001-7886-0455
[a]MIREA — Russian Technological University, 78, Vernadsky Ave., 119454, Moscow, Russian Federation
[b]Moscow Polytechnic University, 38, Bolshaya Semyonovskaya St., 107023, Moscow, Russian Federation

Introduction: The rapid advancement of extended reality (XR) technologies imposes stringent requirements on the precise placement of virtual objects to ensure an optimal user experience. In dynamic scenes with limited computational resources, integrating new objects into an already optimized composition remains a significant challenge. **Purpose:** To develop and experimentally evaluate an algorithm for placing virtual objects in XR space with a focus on computational efficiency. **Results:** We propose a modified algorithm for optimal two-dimensional arrangement based on dynamic programming principles and local adjustments. The algorithm first sorts objects by size, then sequentially places them while minimizing imbalances in zones of perceptual comfort, and finally recalculates positions locally to integrate

new elements without disrupting the scene's initial structure. Our experimental evaluation demonstrates that the algorithm effectively places objects according to their physical dimensions and comfort zones. The timing characteristics show a linear increase in execution time for a small number of objects (up to 35). However, as the number of objects increases (within the range of 35÷70 and beyond), the execution time grows at an accelerated rate, particularly on mobile devices. Regression analysis reveals a strong correlation between the number of objects and execution time for personal computers and iOS devices, while Android platforms exhibit lower scalability. **Practical relevance:** We validate that the proposed method applies effectively in real XR scenarios, reducing computational load and enhancing the adaptability of XR systems. **Discussion:** Our findings open prospects for further optimizing the algorithm through parallel computing and hardware acceleration, as well as extending its functionality to three-dimensional environments.

Keywords — extended reality, virtual objects, optimal placement, dynamic programming, computational efficiency, buffer zones, perceptual state, placement algorithms, mobile devices, regression analysis.

For citation: Alpatova M. V., Rudyak Y. V. Optimization of virtual object placement in extended reality using dynamic programming. *Informatsionno-upravliaiushchie sistemy* [Information and Control Systems], 2025, no. 3, pp. 49–58 (In Russian). doi:10.31799/1684-8853-2025-3-49-58, EDN: SKBXGY

References

1. El Barhoumi N., Hajji R., Bouali Z., Ben Brahim Y., Kharroubi A. Assessment of 3D models placement methods in augmented reality. *Applied Sciences*, 2022, vol. 12, no. 20, Article 10620. doi:10.3390/app122010620
2. He X., Zhou K. Display in the air: Balancing distraction and workload in AR glasses interfaces for driving navigation. *arXiv*, 2024. doi:10.48550/arxiv.2404.18357
3. Ye K., Wu H., Tong X., Zhou K. A real-time method for inserting virtual objects into neural radiance fields. *arXiv*, 2023. doi:10.48550/arxiv.2310.05837
4. Nurhadi N., Stiawan D., Idris M. Y., Saparudin S. Object tracking in augmented reality: Enhancement using convolutional neural networks. *Indonesian Journal of Electrical Engineering and Informatics (IJEEI)*, 2022, vol. 10, iss. 4. doi:10.52549/ijeei.v10i4.4104
5. Alpatova M. V., Rudyak Y. V. Placement of multiple virtual objects in physical space in augmented reality applications. *Advanced Engineering Research*, 2023, vol. 23, no. 2, pp. 203–211 (In Russian). doi:10.23947/2687-1653-2023-23-2-203-211
6. Zhu-Tian C., Chiappalupi D., Lin T., Yang Y., Beyer J., Pfister H. RL-LABEL: A deep reinforcement learning approach intended for AR label placement in dynamic scenarios. *arXiv*, 2023. doi:10.48550/arxiv.2308.13540
7. Xing Y., Liu Q., Wang J., Gomez-Zara D. sMoRe: Enhancing object manipulation and organization in mixed reality spaces with LLMs and Generative AI. *arXiv*, 2024. doi:10.48550/arxiv.2411.11752
8. Alpatova M. V., Rudyak Y. V. Optimal 2D placement of virtual objects in physical space for augmented reality applications. *Advanced Engineering Research*, 2023, vol. 23, no. 4, pp. 410–421 (In Russian). doi:10.23947/2687-1653-2023-23-4-410-421
9. Xu Y., Wu Y., Zhou H. Multi-scale voxel hashing and efficient 3d representation for mobile augmented reality. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, US, June 18–22, 2018, pp. 1586–15867. doi:10.1109/CVPRW.2018.00200
10. Magassouba A., Sugiura K., Nakayama A., Hirakawa T., Yamashita T., Fujiyoshi H., Kawai H. Predicting and attending to damaging collisions for placing everyday objects in photo-realistic simulations. *Advanced Robotics*, 2021, vol. 35, no. 12, pp. 787–799. doi:10.1080/01691864.2021.1913446
11. Zanyk-McLean S., Kumar K., Navratil P. 3D object positioning using differentiable multimodal learning. *arXiv*, 2023. doi:10.48550/arxiv.2309.03177
12. Weiß M. Optimal object placement using a virtual axis. *arXiv*, 2019. https://doi.org/10.48550/arXiv.1901.00168
13. Wiethüchter R., Kalloori S., Chalumattu R. Fast content placement and alignment in 3D scenes. *The International FLAIRS Conference Proceedings*, 2022, vol. 35. doi:10.32473/flairs.v35i.130686
14. Wang H. Haptic repurposing with GenAI: Transforming the tangible world into adaptive haptic interfaces in mixed reality. *arXiv*, 2024. doi:10.48550/arXiv.2406.07228
15. Jia J., Elezovikj S., Fan H., Yang S., Liu J., Guo W., Tan C. C., Ling H. Semantic-aware label placement for augmented reality in street view. *The Visual Computer*, 2021, vol. 37, no. 7, pp. 1805–1819. doi:10.1007/S00371-020-01939-W
16. Zhao X., Rao Y., Qi P., Lyu Q., Yang P., Yu S. An efficient constructive heuristic for the rectangular packing problem with rotations. *PloS One*, 2023, vol. 18, no. 12. doi:10.1371/journal.pone.0295206
17. Douglas C., Huh J. S., Jun S. O., Kim I. Y. Packing optimization of practical systems using a dynamic acceleration methodology. *Journal of Engineering and Applied Science*, 2024, vol. 71, no. 1. doi:10.1186/s44147-024-00426-6
18. Chen D., Mo F., Chen Y., Zhang J., You X. Optimization of ramp locations along freeways: A dynamic programming approach. *Sustainability*, 2022, vol. 14, no. 15, Article 9718. doi:10.3390/su14159718
19. Boudt K., Todorov V., Wang W. Robust distribution-based winsorization in composite indicators construction. *Social Indicators Research*, 2020, vol. 149, no. 2, pp. 375–397. doi:10.1007/s11205-019-02259-w
20. Orenstein P. Robust importance sampling with adaptive winsorization. *Bernoulli*, 2022, vol. 28, no. 4. doi:10.3150/21-bej1440
21. Poudyal C., Zhao Q., Brazauskas V. Method of winsorized moments for robust fitting of truncated and censored lognormal distributions. *North American Actuarial Journal*, 2024, vol. 28, no. 1, pp. 236–260. doi:10.1080/10920277.2023.2183869