ISSN 0132-3474

Номер 3 Май - Июнь 2023


# ПРОГРАММИРОВАНИЕ

www.sciencejournals.ru

# СОДЕРЖАНИЕ

**Номер 3, 2023**

## КОМПЬЮТЕРНАЯ ГРАФИКА И ВИЗУАЛИЗАЦИЯ



## АНАЛИЗ ДАННЫХ

# CONTENTS

**No. 3, 2023**

## COMPUTER GRAPHICS AND VISUALIZATION



## DATA ANALYSIS

*ПРОГРАММИРОВАНИЕ, 2023, № 3, с. 3–12*

## КОМПЬЮТЕРНАЯ ГРАФИКА И ВИЗУАЛИЗАЦИЯ

*УДК 004.921*

# ПОДДЕРЖКА ВЕКТОРНЫХ ТЕКСТУР В СИСТЕМЕ ФОТОРЕАЛИСТИЧНОГО РЕНДЕРИНГА НА GPU

**© 2023 г. В. В. Санжаров[a,*] (ORCID: 0000-0001-6455-6444),**
**В. А. Фролов[a,b,**] (ORCID: 0000-0001-8829-9884), В. А. Галактионов[b,***] (ORCID: 0000-0001-6460-7539)**

[a] *Московский государственный университет им. М.В. Ломоносова*
*Россия, 119991, Москва, Ленинский горы, д. 1*

[b] *Институт прикладной математики им. М.В. Келдыша*
*РАН Россия, 125047, Москва, Миусская пл., д. 4*

**e-mail: vadim.sanzharov@graphics.cs.msu.ru*

***e-mail: vladimir.frolov@graphics.cs.msu.ru*

****e-mail: vlgal@gin.keldysh.ru*

Поступила в редакцию 09.01.2023 г.
После доработки 16.01.2023 г.
Принята к публикации 20.01.2023 г.

Изображения в векторном формате представлены в виде последовательности аналитических описаний геометрических объектов. Такой подход позволяет воспроизвести изображение в любом разрешении без потери качества. На текущий момент не существует готовых решений для использования векторных изображений в системах фотореалистичного рендеринга на GPU. В данной работе представлен подход к реализации такой поддержки, основанный на базовых методах – полей расстояний со знаком и растеризации. Анализ результатов показывает эффективность подхода на основе полей расстояний для различных векторных изображений. Однако, в отдельных случаях возможно появление артефактов, в этом случае предлагается использовать подход на основе растеризации.

**DOI:** 10.31857/S0132347423030044, **EDN:** DELQSS

## 1. ВВЕДЕНИЕ

Векторные изображения представляют изображения на основе математического аппарата аналитической геометрии. Вводятся описания геометрических объектов в декартовой системе координат. Среди поддерживаемых объектов – точки, прямые, окружности, полигоны, кривые Безье и пр. Один из наиболее распространенных форматов векторных изображений – формат SVG, являющийся стандартом Консорциума Всемирной паутины (World Wide Web Consortium, W3C) [1]. В данном формате векторные изображения представлены текстовым описанием в виде XML-файла. Геометрические объекты в таком описании объединяются в группы, которым может назначаться набор свойств. Для получения растрового изображения, необходимо последовательно “нарисовать” геометрические объекты в соответствии с их описанием и свойствами.

Такой подход к представлению графической информации обладает рядом преимуществ. Во-первых, он позволяет получить изображение в нужном разрешении без потерь информации. Во-вторых, размер векторного изображения зависит от его сложности, а не от разрешения или глубины цвета, что позволяет получить компактное представление, по сравнению с растровыми изображениями (особенно в случае высокого разрешения последних). Наконец, описание объектов на основе координат и уравнений позволяет получить большую точность описания изображения, чем использование растровой сетки с конечным разрешением. Также имеется ряд преимуществ с точки зрения создания векторных изображений. В частности, возможность манипулировать отдельными объектами независимо, а также – выполнять алгоритмическую генерацию текстового описания изображения, включая генерацию сложных паттернов [2].

Эти преимущества привели к распространению векторной графики в различных приложениях, где информация может быть представлена с помощью набора геометрических примитивов – типография, пользовательские интерфейсы, веб-дизайн, автоматизированное проектирование (чертежи) и др.

В 3D-графике использование векторных изображений ограничивается интерактивными графическими приложениями, где таким способом представляют преимущественно пользователь-


**Рис. 1.** Пример изображений, синтезированных системой фотореалистичного рендеринга с использованием предлагаемого решения. **Слева** – рендер сцены производился в разрешении FullHD, векторные изображения на картине и на полу. **По центру** – на этой же сцене камера была приближена к текстурированной картине и рендер был произведен в разрешении 8192 на 8192 без изменения векторной текстуры. **Справа** – векторные текстуры использовались как маски для смешения разных моделей материалов.

ские интерфейсы [3] и текст [4]. Это во многом связано с тем, что с помощью простых геометрических форм, используемых в векторной графике, сложно описывать объекты реального мира. Это также касается (даже в большей мере) описания цвета. Векторные изображения обычно используют закраску сплошным цветом или градиенты. Подобными средствами чрезвычайно сложно представить такие эффекты, как микрорельеф поверхности, мягкие тени, каустики, глубину резкости и др. [5]. Проблема сложности создания векторных изображений объектов реального мира начинает находить решение в использовании подходов, позволяющих автоматически синтезировать векторные изображения из растровых изображений (в том числе фотографий), на основе дифференцируемого рендеринга [6], нейронных сетей [7], включая диффузионные модели [8].

Другая проблема использования векторных изображений связана с тем, что для вычисления цвета пикселя необходимо обойти все геометрические примитивы в изображении. Тогда как для растрового изображения возможен эффективный случайный доступ к любому пикселю, имеющий также аппаратную поддержку в графических процессорах.

Несмотря на описанные недостатки, использование векторных текстур в фотореалистичном рендеринге на текущий момент может быть оправдано при необходимости синтезировать в высоком разрешении изображения, включающие текст, изображения, которые могут быть представлены геометрическими примитивами (рис. 1), а с учетом развития методов автоматической генерации векторных изображений, возможно, и произвольные текстуры. Так как при рендеринге в высоких разрешениях для обеспечения качественного результата необходимо использовать текстуры также высокого разрешения, в то время как векторные изображения не зависят от разрешения.

В данной работе представлен подход к поддержке векторных текстур в формате SVG в фотореалистичной системе рендеринга на GPU.

## 2. СУЩЕСТВУЮЩИЕ РЕШЕНИЯ

В большинстве систем фотореалистичного рендеринга для использования изображения в векторном формате в качестве текстуры пользователь сначала должен самостоятельно выполнить его конвертацию в растровое изображение в некотором стороннем инструменте. При этом ему понадобится некоторым образом определить необходимое разрешение, например, исходя из желаемого разрешения рендеринга. А при изменении разрешения рендеринга в большую сторону, скорее всего потребуется выполнить конвертацию заново, также увеличив разрешение растрового представления векторного изображения. Такой подход создает неудобства для конечного пользователя, вынуждая его найти и использовать некоторый сторонний инструмент для конвертации, а также задавать параметры этой процедуры.

Также следует отметить, что многие области применения фотореалистичного рендеринга подразумевают синтез изображения в высоком разрешении. В частности, полиграфия, визуальные эффекты и анимационные фильмы, моделирование оптических систем. Таким образом, векторные изображения требуется конвертировать в растровые изображения высокого разрешения, занимающие значительные объемы памяти. А с учетом перехода многих систем фотореалистичного рендеринга на использование GPU, связанное с появлением аппаратного ускорения трассировки лучей [9], этот фактор приобретает еще большую значимость.

### *2.1. Обзор существующих решений*

Существуют различные программные решения для работы с векторными изображениями, такие как Skia [10], cairo [11], Blend2d [12], resvg [13]. Данные решения обеспечивают полноценную поддержку стандарта SVG, высокую производительность, различные целевые системы. В случае реализации рендер-системы на CPU, было бы возможно подключить одно из этих решений в качестве библиотеки и вызывать ее при необходимости обратиться к векторной текстуре. Однако, для GPU рендер-системы такая возможность отсутствует, так как вычислительное ядро такого рендера работает в другом адресном пространстве.

Консорциумом Khronos Group был разработан стандарт программного интерфейса (API) для аппаратно-ускоренного рендеринга векторной графики под названием OpenVG [14], нацеленный на приложения 2D-графики – навигаторы, чтение электронных книг и др. Данный стандарт предлагает новый вид графического конвейера для векторной графики. Поддержка OpenVG была реализована главным образом в мобильных GPU [15]. Однако, широкого распространения данный стандарт не получил.

Расширение NV_path_rendering [16] для OpenGL позволяет использовать примитивы типа “путь” (path, по сути параметрическая кривая, сплайн) векторной графики в 3D графических приложениях на GPU от NVidia. Отсутствие поддержки аппаратной трассировки лучей в OpenGL, а также отсутствие кроссплатформенности делают это решение неактуальным для современных систем фотореалистичного рендеринга.

В работе [17] предлагается метод рендеринга параметрических кривых с использованием аппаратного ускорения на GPU. Предлагаемый метод основан на представлении сегментов кривой с помощью треугольников, которые затем отрисовываются с помощью графического конвейера. Для систем фотореалистичного рендеринга, подавляющее большинство которых основано на трассировки лучей, таким образом потребуется выполнять построение ускоряющих структур данных для сгенерированной геометрии. А при изменении и/или добавлении текстур в сцену, потребуется обновлять ускоряющие структуры. При этом количество сгенерированных треугольников может быть достаточно велико, а для мелких элементов векторного изображения метод дает артефакты.

В работе [18] авторы предлагают специальную ускоряющую структуру данных для реализации эффективного рендеринга векторных изображений на GPU. Данный подход ориентирован на приложения 2D-графики.

Работа [19] предлагает новое представление для векторных изображений, ориентированное на эффективный рендеринг с помощью графического конвейера. Основным недостатком подхода является отсутствие автоматического решения по преобразованию произвольных векторных изображений в новое представление, предложенное авторами. Также, использование функций, специфичных для графического конвейера, ограничивает применение подхода рендер-системами, уже использующими графический конвейер.

В работе [20] впервые предложен подход на основе полей расстояний со знаком (signed distance field, SDF). Суть подхода состоит в том, что для параметрических кривых вычисляются поля расстояний, которые затем используются в качестве текстур. Рендер-система обращается к текстуре поля расстояний и на основе полученного значения, определяет, виден ли в данном пикселе геометрический объект, которому соответствует данная текстура. Авторы [4] предлагают комбинировать поля расстояний со знаком с механизмом альфа-смешивания для вычисления цвета при использовании векторных изображений как текстур. Работа [21] совершенствует подход на основе полей расстояний за счет расчета нескольких полей расстояний для разных граней геометрической формы, которые хранятся в разных цветовых каналах SDF-текстуры. Это позволяет повысить точность представления исходной геометрической формы. Недостатком подходов на основе SDF являются визуальные артефакты для некоторых видов геометрических объектов, в частности, обладающих острыми углами. Кроме того, для точного представления сложных форм требуется генерация SDF-текстур в достаточно высоком разрешении. Наконец, эти методы не могут представить несколько кривых, пересекающихся в одном пикселе и сами по себе рассматривают случай только отдельных непересекающихся геометрических объектов. На практике же векторное изображение может состоять из большого числа наложенных друг на друга и пересекающихся объектов.

В работе [22] также используется представление на основе расстояний, однако уже рассматривается целое векторное изображение, представленное в виде слоев, содержащих геометрические объекты. Векторное изображение представляется в виде сетки, в каждой ячейке которой хранится информация о всех геометрических объектах, попадающих в данную ячейку. Для корректного рендеринга наложения объектов используется композирование. Недостатком подхода является использование специфичных для графического конвейера функций вычисления частных производных по изображению (*dFdx* и *dFdy* в GLSL) для отображения из текстурного пространства в пространство экрана. Реализация данных операций в фотореалистичной рендер-системе на основе трассировки лучей возможна с использованием т.н. дифференциалов луча (ray differentials) [23].

Однако метод дифференциалов луча дает строго корректные результаты только для прямых лучей и может давать значительную ошибку для многократных отражений. Насколько данная ошибка будет влиять на подход к реализации векторных текстур в [22] является открытым вопросом. Отчасти похожий подход, также основанный на расчете расстояний, предлагается в [24]. Информация о векторном изображении здесь сохраняется в специальную ускоряющую структуру данных, представленную в виде трех текстур. Для данного подхода, а также для [22] остаются недостатки, присущие другим подходам на основе расстояний, связанные с возможными артефактами для некоторых видов геометрических объектов. Также в [22] и [24] вычисление расстояний происходит в процессе рендеринга, что ограничивает использование возможных алгоритмов для этой задачи и может ощутимо влиять на производительность.

### *2.2. Резюме по существующим решениям*

Все существующие решения имеют недостатки, так или иначе ограничивающие их применение. Готовые решения для 2D-рендеринга [10–13] могут быть использованы либо в CPU рендер-системе как библиотеки для вычисления векторной текстуры на лету, или же могут быть интегрированы в качестве средства предварительной растеризации векторной текстуры. Часть решений [16] накладывают ограничения по использованию определенного аппаратного и программного обеспечения. В [19, 22] используется функциональность, специфичная для графического конвейера. Подходы, основанные на вычислении расстояний [4, 20–22, 24] могут давать артефакты для некоторых видов геометрических объектов. При этом часть подходов обрабатывают отдельные геометрические объекты [4, 20, 21], а часть [22, 24] выполняют расчет расстояний в процессе рендеринга, оказывая влияние на производительность.

Таким образом, для использования векторных текстур в системе фотореалистичного рендеринга на GPU требуется объединить несколько существующих подходов, чтобы покрыть основные сценарии использования.

## 3. ПРЕДЛАГАЕМОЕ РЕШЕНИЕ

В качестве формата векторных изображений предлагаемое решение поддерживает формат SVG [1], как стандартизированный и один из наиболее распространенных. Формат SVG является достаточно сложным и помимо различных геометрических примитивов может включать встроенные растровые изображения, встроенный код на языке JavaScript, анимацию и многие другие компоненты. При этом эти компоненты могут обладать различными свойствами, включая разные виды цветовой заливки, контурную обводку разными типами линий, фильтры и др. Предлагаемое решение реализует поддержку ограниченного подмножества функциональности допустимой в формате SVG. В частности, в качестве геометрических объектов поддерживаются объекты типа path (параметрические кривые), так как все остальные геометрические объекты, включая текст, могут быть представлены с помощью объектов типа path.

В первую очередь предлагаемое решение загружает файл в формате SVG и вычленяет все объекты типа path в порядке их расположения по слоям, а также свойства этих объектов – цвет, геометрические преобразования, наличие контуров и их цвет.

Далее каждый объект path подается на вход компоненту вычисления поля расстояний. После расчета поля расстояний для объекта опционально выполняется обрезание поля расстояний по ограничивающему прямоугольнику соответствующего геометрического объекта. Это позволяет избавиться от артефактов, вносимых неточностями расчета поля расстояний при удалении от объекта (рис. 5). На выходе имеем массив отдельных полей расстояний для каждого из объектов path в исходном векторном изображении.

В качестве следующего этапа может опционально производиться объединение отдельных полей расстояний в единое путем наложения, если геометрические объекты в исходном изображении не пересекаются. Такой случай характерен в первую очередь для текста, т.к. обычно в компьютерных шрифтах буквы не пересекаются и не накладываются одна на другую.

Наконец массив полей расстояний (или комбинированное поле расстояний) вместе с массивами, содержащими свойства геометрических объектов, передается на GPU. Эти данные используются в процессе рендеринга при обращении к текстуре, соответствующей векторному изображению. При таком обращении вызывается текстурный шейдер, сгенерированный на основе настроек, заданных пользователем при загрузке векторного изображения. В текстурном шейдере (по сути в процедурной текстуре), являющейся программируемой частью рендер системы (см. рис. 2), выполняется расчет цвета пикселя на основе полей расстояний и композирования объектов path. В простом случае одного поля расстояний происходит альфа-смешивание цвета объекта path и некоторого фонового цвета (например, цвета диффузного материала, для которого используется векторная текстура), где в качестве веса для смешивания используется расстояние из поля расстояний:

$$dist = median(msdf)$$

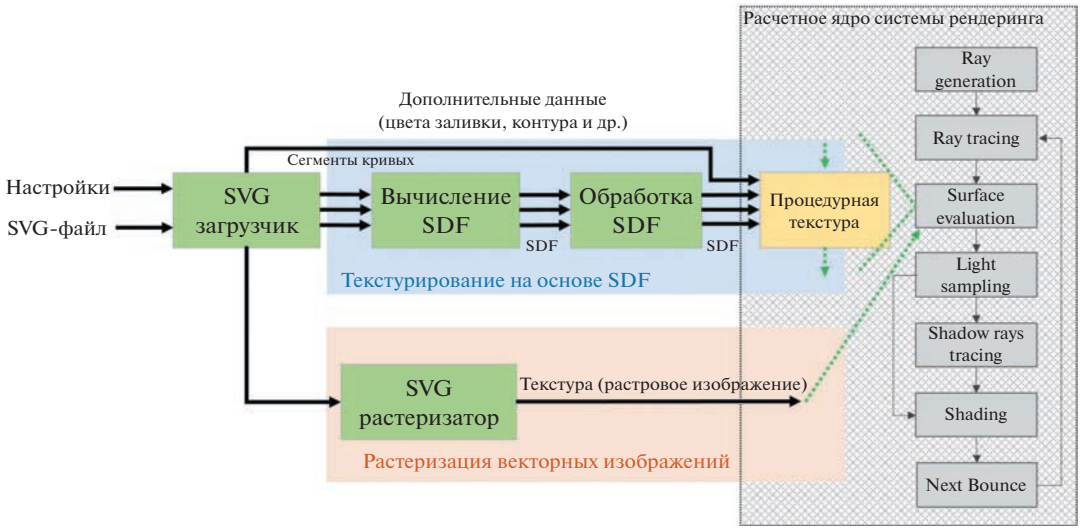


**Рис. 2.** Схема предлагаемого решения. Блок “Обработка SDF” – опциональный, включает в себя вырезание SDF по ограничивающему прямоугольнику геометрической формы, для которой рассчитывалось поле расстояний.

$$\alpha = smoothstep(T - S, T + S, dist) \quad (1)$$

$$color = bgColor * (1 - \alpha) + baseColor * \alpha$$

где:

• *median(a)* – функция, принимающая на вход вектор и возвращающая медиану его компонентов;

• *msdf* – вектор из трех компонент, содержащий значения полей расстояний;

• *smoothstep(a, b, x)* – функция выполняющая эрмитову интерполяцию между 0 и 1, при $a < x < b$;

• $T$ – константа, которая задает значение расстояния, определяющее границы геометрического объекта;

• $S$ – константа сглаживания, при больших значениях цветовой переход на границе объекта будет более плавным;

• *bgColor* – цвет фона;

• *baseColor* – цвет геометрического объекта.

Функция *smoothstep* служит в качестве механизма антиалиасинга для “смягчения” перехода на границе геометрического объекта, вместо попиксельного вычисления частных производных по координатам экрана с помощью встроенной функциональности графического конвейера. При необходимости она может быть заменена на другую схожую по характеристикам функцию или же на использование производных, при их доступности.

В более общем случае, при наличии нескольких полей расстояний, необходимо следовать формату SVG, в котором геометрические объекты задаются в порядке отрисовки – второй объект рисуется поверх первого, третий поверх второго и т.д. Таким образом, необходимо по порядку обойти массив полей расстояний, на каждом этапе выполняя альфа-смешивание очередного геометрического объекта с результатом предыдущей итерации аналогично (1):

$$\begin{aligned} &color = bgColor \\ &for\ each\ object: \\ &\quad baseColor = object.color \\ &\quad msdf = object.msdf \\ &\quad dist = median(msdf) \\ &\quad \alpha = smoothstep(T - S, T + S, dist) \\ &\quad color = color * (1 - \alpha) + baseColor * \alpha \\ &return\ color \end{aligned} \quad (2)$$

где:

• *object.color* – цвет текущего геометрического объекта;

• *object.msdf* – поле расстояний текущего геометрического объекта.

Для того чтобы реализовать поддержку контуров, диапазон между минимальным и максимальным значениям расстояния задается равным ширине контура. Тем самым минимальное расстояние будет соответствовать объекту, максимальное – фону, а между ними будет располагаться контур. Тогда, на этапе смешивания, если значения расстояния попадают в этот диапазон, то значение *baseColor* в (1) и (2) рассчитывается как результат смешивания цветов геометрического объекта и контура:

$$\omega = \begin{cases} smoothstep(0, S, dist), dist < S \\ smoothstep(1, 1 - S, dist), dist \geq S \end{cases} \quad (3)$$

$$baseColor = shapeColor * (1 - \omega) + outlineColor * \omega$$

где:

• *shapeColor* – цвет заливки геометрического объекта;

• *outlineColor* – цвет контура.

В предлагаемой реализации поддерживается только сплошная заливка объекта и контура. Однако, поддержка, например, градиентной заливки представляет собой довольно простую модификацию. Для этого необходимо сохранять не просто значения цвета заливки *shapeColor*, а значения нескольких цветов и коэффициенты смешивания. Например, вычислить медиану между полями расстояний заранее, а в другие цветовые каналы записать коэффициенты градиентов.

### *3.2. Детали реализации*

Предлагаемая реализация была интегрирована в рендер-систему, которая взаимодействует с клиентскими приложениями через промежуточный программный слой (API). Для загрузки векторного изображения пользователю предоставляется возможность задания настроек реализованного подхода, которые включают:

• способ вычисления полей расстояний или же использование механизма растеризации;

• множитель разрешения поля расстояний или растеризованной текстуры;

• параметры алгоритма расчета расстояний;

• флаг генерации одноканальной маски из векторного изображения;

• указание комбинировать все поля расстояния вместе и указание выполнять обрезание поля расстояний по ограничивающему прямоугольнику геометрического объекта;

• текстурную матрицу и способ обработки текстурных координат, превышающих 1;

• включение/выключение отрисовки контуров;

• цвет фона и переназначение цвета для всех геометрических объектов и всех контуров.

Большинство из этих параметров нет необходимости менять при стандартных сценариях использования. Однако, при необходимости возможна тонкая настройка алгоритма. Например, в специальных случаях, таких как текст или паттерн из непересекающихся объектов, можно получить прирост в производительности и экономию памяти путем комбинирования полей расстояния. Если векторное изображение используется только как маска, то достаточно вычислить одноканальное изображение. А при появлении артефактов, изменение параметров расчета полей расстояния может помочь от них избавиться или уменьшить их проявление. На основе заданных параметров производится генерация кода процедурной текстуры. Пример фрагмента сгенерированного кода представлен ниже:

```
float4 prtex14_main(const SurfaceInfo * sHit, unsigned mode, ...){
    const float S = 0.015625; const float T = 0.5;
    const float2 texCoord = readAttr_TexCoord0(sHit);
    float2 texCoord_adj = prtex14_applyTexMatrix(texCoord, texMatrix);
    float4 resColor = bgColor;
    for(int i = 0; i < numTextures; i += 1) {
    const float4 objColor = prtex14_colorFromUint(objColors[i]);
    const float4 texColor = texture2D(sdfTexture[i], texCoord_adj, texFlags);
    const float distance = (mode & MSDF) ?
    prtex14_median(texColor.x, texColor.y, texColor.z) : texColor.x;
    const float alpha = prtex14_smoothstep(T - S, T + S, distance);
    const float4 outlineColor = prtex14_colorFromUint(outlineColors[i]);
    float4 baseColor = (hasOutline[i] == 0) > objColor:
    prtex14_addOutline(distance, S, outlineColor, objColor);
    resColor = prtex14_mix4(resColor, baseColor, alpha);
    }
    return resColor;
}
```

В частности, если отключена отрисовка контуров, часть алгоритма, ответственная за нее выбрасывается из результирующего кода. Если выполняется композирование полей расстояний или в исходном изображении только один объект, то нет необходимости передавать дополнительные

данные для смешивания. Таким образом, код процедурной текстуры генерируется в максимально упрощенном виде.

### *3.3. Экспериментальная оценка*

Как было отмечено ранее, предлагаемый подход в отличие от [22, 24] выносит вычисление полей расстояний в некоторый предварительный этап, внешний по отношению к процессу рендеринга. Но при этом возникает необходимость загрузки полей расстояний как текстур на GPU и обращению к ним в процессе рендеринга. Если для каждого из объектов в исходном векторном изображении сгенерировано отдельное поле расстояний, то это так же может оказывать ощутимое влияние на производительность. Для оценки этого был проведен эксперимент:

1. Алгоритмически были сгенерированы векторные изображения, содержащие заданное в диапазоне от 16 до 2048 количество геометрических объектов (случайных секторов окружности), расположенных случайным образом.

2. Для каждого из сгенерированных изображений рассчитано поле расстояний с одинаковыми настройками алгоритма.

3. Произведен рендеринг простейшей 3D-сцены с равномерным фоновым освещением, содержащей плоскость со сгенерированным векторным изображением в качестве текстуры.

4. Проведены замеры производительности.

Результаты эксперимента приведены на рис. 3. Видно, что зависимость от количества объектов в векторном изображении (т.е. от числа текстур полей расстояний) имеет линейный характер. На тестовой сцене, где текстурированная плоскость занимает все видимое изображение, при количестве объектов меньше 256, затраты на исполнение предложенного алгоритма составляют меньше 1 секунды для 256 выборок (путей) на пиксель при разрешении рендеринга 1024 на 1024. А для 2048 объектов в векторном изображении время исполнения процедурной текстуры равно примерно 8.2 секунды. В контексте фотореалистичного рендеринга это приемлемые затраты. При этом видно, что процент времени рендеринга, необходимый для исполнения процедурной текстуры “насыщается” и замедляет свой рост с увеличением числа объектов.

Для следующего эксперимента были выбраны векторные изображения из открытых источников [25, 26], представляющие некоторые типичные сценарии использования (узоры, стилизованные изображения, текст) с разной сложностью – разным количеством объектов. Выбранные изображения использовались в качестве текстуры в такой же сцене с плоскостью, как и в предыдущем эксперименте. Были произведены аналогичные замеры производительности (табл. 1), а также оценивалось качество результатов (рис. 4, 5).

Для некоторых изображений были зафиксированы артефакты, возникающие для определенных элементов (рис. 5).

Часть этих артефактов обусловлена проблемами, возникающими у метода расчета полей расстояний для острых углов. Это становится особенно заметным при отключении вырезания поля расстояний по ограничивающему прямоугольнику объекта (рис. 5D, E). Обрезание поля расстояний в значительной мере устраняет артефакты (рис. 5A). Если при вырезании поля расстояний использовать выпуклую оболочку, а не ограничивающий прямоугольник, то потенциально может быть устранено еще больше артефактов данного вида.

Другая группа артефактов возникает из-за некорректной интерпретации определенных форм. Например, у незамкнутых сегментов параметрической кривой, которые не имеют внутренней заливки, а отображаются только как контуры (рис. 5B, C). Решение данной проблемы требует внесения модификаций в алгоритм расчета расстояний для специальной обработки таких случаев. В предлагаемой реализации на текущий момент при возникновении таких артефактов предлагается использовать вспомогательный подход на основе растеризации.

Однако при использовании подхода на основе растеризации потребуется использовать растровое изображение с потенциально значительно большим разрешением, чем разрешение текстур полей расстояний. Для адекватного представления исходного векторного изображения необходимо обеспечить достаточно разрешения и текстуры поля расстояний. Однако, как только разрешения будет достаточно, чтобы передать все детали векторного изображения, дальнейшее увеличение разрешения не будет давать никаких результатов. Таким образом, можно увеличивать разрешение рендеринга и приближать текстурированный объект как угодно без потерь качества. А для растеризованной текстуры, при увеличении разрешения рендеринга потребуется соответствующим образом увеличить и разрешение текстуры, чтобы избежать появления артефактов. Это становится особенно важным для случая явных, резких границ между объектами, характерных для векторных изображений. Поэтому при больших разрешениях рендеринга, выигрыш от использования подхода на основе полей расстояний будет расти.

Наконец, следует отметить, что в предлагаемой архитектуре решения (рис. 2), включающей модуль растеризации, можно рассматривать векторные текстуры как случай “предварительно вычисленных процедурных текстур” и использовать

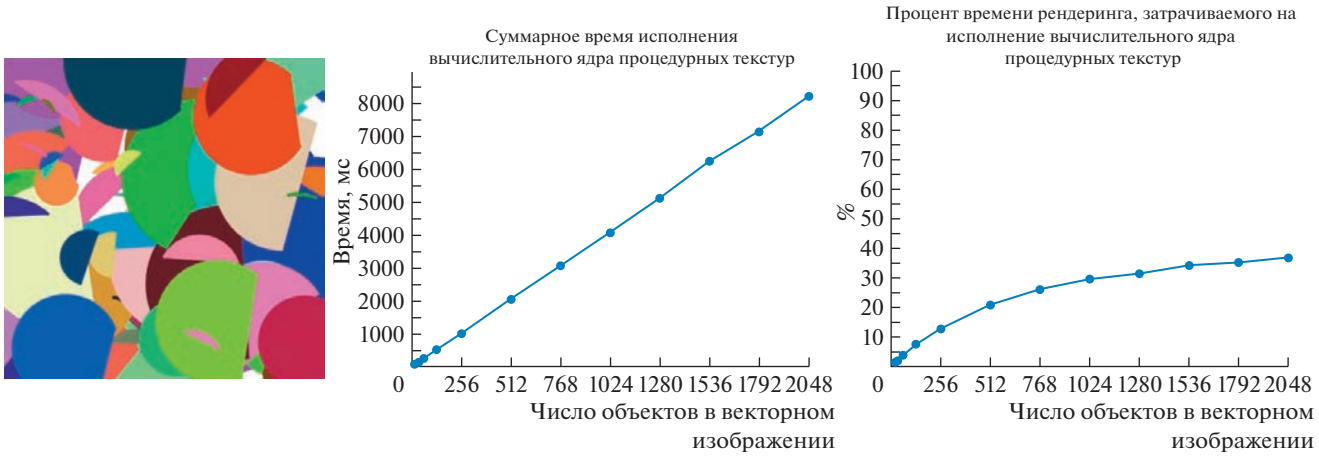


**Рис. 3.** Результаты тестирования производительности на синтезированных текстурах. Слева – пример синтезированной текстуры. По центру – зависимость суммарного времени исполнения вычислительного ядра процедурных текстур от числа объектов в векторном изображении. Справа – зависимость процента времени рендеринга, затрачиваемого на исполнение вычислительного ядра процедурных текстур. Тестирование производилось на GPU RTX2070 Super, рендеринг в разрешении 1024 на 1024, 256 выборок на пиксель, разрешение SDF текстур 256 на 256.

описанный в [27] механизм для автоматического определения разрешения растеризации, необходимого для достижения соотношения текселей к пикселям 1 : 1.

## 4. ЗАКЛЮЧЕНИЕ

Предложено решение, обеспечивающее возможность использования векторных изображений, содержащих объекты, представленные параметрическими кривыми, в качестве текстур в фотореалистичном рендеринге 3D-сцен. Решение представляет собой программную архитектуру (рис. 2), интегрирующую метод на основе расчета полей расстояний (в рассмотренной реализации использовался метод [21]) и вспомогательный механизм растеризации векторных изображений. Для подхода на основе полей расстояний предложен подход обработки случая пересекающихся объектов векторных изображений и объектов, имеющих контуры. Предложенная программная архитектура является гибкой и позволяет заменять модули расчета полей расстояний и растеризации, независимо от рендер-системы. Также предложенная архитектура снимает с пользователя рендер-системы задачу по ручной конвертации векторных изображений в стороннем инструменте и предоставляет возможность использовать их напрямую в рендер-системе. Реализованный подход на основе полей расстояний не оказывает значи-

**Таблица 1.** Замеры производительности предлагаемого решения для векторных изображений из открытых источников (см. рис. 4). Замеры производилось на GPU RTX2070 Super, рендеринг в разрешении 1024 на 1024, 256 выборок на пиксель, разрешение SDF-текстур 256 на 256

| Изображение | Число объектов в изображении | Время рендеринга, мс | Суммарное время исполнения выч. ядра проц. текстур, мс | Процент от времени рендеринга, затрачиваемый на исполнение выч. ядра проц. текстур, % | Разрешение текстур полей расстояний |
|---|---|---|---|---|---|
| tiger | 239 | 7751 | 1056 | 13.6 | 512 × 512 |
| autumn leaves | 21 | 5750 | 123 | 2.1 | 512 × 512 |
| trophy | 15 | 5750 | 83 | 1.4 | 512 × 512 |
| Horus and Thoth | 11 | 6251 | 127 | 2.0 | 1024 × 1024 |
| prismatic mandala | 223 | 8251 | 1041 | 12.6 | 1024 × 1024 |
| text | 105 | 6751 | 421 | 6.2 | 512 × 512 |
| text combined | 1 | 5751 | 58 | 1.0 | 512 × 512 |


**Рис. 4.** Тестирование предложенного подхода на векторных изображениях из открытых источников. В верхнем ряду – результат рендеринга с использованием предложенного подхода, внизу – эталонные изображения. На изображении листьев демонстрируется применение текстурных матриц с бесшовной векторной текстурой. Присутствуют артефакты – например, на изображении тигра (см. рис. 5).


**Рис. 5.** Примеры артефактов на изображении тигра. A – результат рендеринга с использованием предлагаемого метода с вырезанием поля расстояний по ограничивающему прямоугольнику объекта; B – наверху фрагмент эталонного изображения, по середине – фрагмент синтезированного изображения A, внизу – элементы векторного изображения, вызывающие проблему; C – текстуры поля расстояний для проблемных элементов; D – результат рендеринга без вырезания поля расстояний; E – текстуры поля расстояний с артефактными значениями для острых углов.

тельного влияния на время рендеринга. Для случая векторных изображений, содержащих текст и простые паттерны, их использование равноценно использованию одной обычной растровой текстуры (табл. 1).

## СПИСОК ЛИТЕРАТУРЫ

1. Scalable Vector Graphics (SVG) Full 1.2 Specification, 2023. URL: https://www.w3.org/TR/SVG12
2. *Tu P., Wei L. Y., Zwicker M.* Clustered vector textures // ACM Transactions on Graphics (TOG). 2022. Т. 41. № 4. С. 1–23. https://doi.org/10.1145/3528223.3530062
3. Noesis Gui. User Interface middleware for videogames and real-time applications, 2023. URL: https://www.noesisengine.com
4. *Green C.* Improved alpha-tested magnification for vector textures and special effects // ACM SIGGRAPH 2007 Courses. 2007. P. 9–18.
5. *Orzan A. et al.* Diffusion curves: a vector representation for smooth-shaded images // ACM Transactions on Graphics (TOG). 2008. Т. 27. № 3. С. 1–8. https://doi.org/10.1145/1360612.1360691
6. *Li T. M. et al.* Differentiable vector graphics rasterization for editing and learning // ACM Transactions on Graphics (TOG). 2020. Т. 39. № 6. С. 1–15. https://doi.org/10.1145/3414685.3417871
7. *Reddy P. et al.* Im2vec: Synthesizing vector graphics without vector supervision // Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2021. С. 7342–7351.
8. *Jain A., Xie A., Abbeel P.* VectorFusion: Text-to-SVG by Abstracting Pixel-Based Diffusion Models //arXiv preprint arXiv:2211.11319. 2022.
9. *Sanzharov V.V., Frolov V.A., Galaktionov V.A.* Survey of Nvidia RTX technology // Programming and Computer Software. 2020. Т. 46. № 4. С. 297–304. https://doi.org/10.1134/S0361768820030068
10. Skia: The 2D graphics library, 2023. URL: https://skia.org/

11. Cairo, a 2D graphics library with support for multiple output devices, 2023. URL: https://www.cairographics.org/
12. Blend2d. 2D Vector graphics engine, 2023. URL: https://blend2d.com/
13. resvg, SVG rendering library, 2023. URL: https://github.com/RazrFalcon/resvg
14. OpenVG, the standard for vector graphics acceleration, 2023. URL: https://www.khronos.org/openvg/
15. OpenVG, conformant Products, 2023. URL: https://www.khronos.org/conformance/adopters/conformant-products/openvg
16. *Kilgard M.J., Bolz J.* GPU-accelerated path rendering // ACM Transactions on Graphics (TOG). 2012. Т. 31. № 6. С. 1–10. https://doi.org/10.1145/2366145.2366191
17. *Loop C., Blinn J.* Resolution independent curve rendering using programmable graphics hardware // ACM SIGGRAPH 2005 Papers. 2005. С. 1000–1009. https://doi.org/10.1145/1186822.1073303
18. *Ganacim F. et al.* Massively-parallel vector graphics // ACM Transactions on Graphics (TOG). 2014. Т. 33. № 6. С. 1–14. https://doi.org/10.1145/2661229.2661274
19. *Ray N., Cavin X., Lévy B.* Vector texture maps on the GPU //Inst. ALICE (Algorithms, Comput., Geometry Image Dept. INRIA Nancy Grand-Est/Loria), Tech. Rep. ALICE-TR-05-003. 2005.
20. *Qin Z., McCool M.D., Kaplan C.S.* Real-time texture-mapped vector glyphs // Proceedings of the 2006 symposium on Interactive 3D graphics and games. 2006. С. 125–132. https://doi.org/10.1145/1111411.1111433
21. *Chlumsky V.* Shape decomposition for multi-channel distance fields: Master's thesis, Czech Technical University. URL: https://dspace.cvut.cz/bitstream/handle/10467/62770/F8-DP-2015-Chlumsky-Viktor-thesis. pdf. 32, 2015.
22. *Nehab D., Hoppe H.* Random-access rendering of general vector graphics // ACM Transactions on Graphics (TOG). 2008. Т. 27. № 5. С. 1–10. https://doi.org/10.1145/1409060.1409088
23. *Akenine-Möller T. et al.* Texture level of detail strategies for real-time ray tracing // Ray Tracing Gems. Apress, Berkeley, CA, 2019. С. 321–345. https://doi.org/10.1007/978-1-4842-4427-2_20
24. *Qin Z., McCool M D., Kaplan C.* Precise vector textures for real-time 3D rendering // Proceedings of the 2008 symposium on Interactive 3D graphics and games. 2008. С. 199–206. https://doi.org/10.1145/1342250.1342281
25. Vector images in public domain, 2023. URL: https://www.publicdomainvectors.org
26. Open Clipart, online media collection, 2023. URL: https://openclipart.org/
27. *Sanzharov V.V., Frolov V.A.* Level of detail for precomputed procedural textures // Programming and Computer Software. 2019. Т. 45. № 4. С. 187–195. https://doi.org/10.1134/S0361768819040078

*ПРОГРАММИРОВАНИЕ, 2023, № 3, с. 13–25*

## КОМПЬЮТЕРНАЯ ГРАФИКА И ВИЗУАЛИЗАЦИЯ

*УДК 004.932.2*

# АЛГОРИТМ ОБНАРУЖЕНИЯ АТМОСФЕРНЫХ ОСАДКОВ ДЛЯ ЗАДАЧ КОМПЬЮТЕРНОЙ ОБРАБОТКИ ВИДЕОИЗОБРАЖЕНИЙ

**© 2023 г. В. Т. Дмитриев[a,*] (ORCID: 0000-0001-5521-6886),**
**А. А. Бауков[a,**] (ORCID: 0000-0002-0003-058X)**

*[a]Рязанский государственный радиотехнический университет имени В.Ф. Уткина,*
*Россия, 390005, Рязань, ул. Гагарина, д. 59/1*

**E-mail: vol77@rambler.ru*

***E-mail: baukov.andrej@yandex.ru*

Поступила в редакцию 09.01.2023 г.
После доработки 16.01.2023 г.
Принята к публикации 22.01.2023 г.

Показана актуальность задачи обнаружения и уменьшения видимости атмосферных осадков на видеоизображениях, полученных неподвижными камерами. Выполнен статистический анализ геометрических (площадь, коэффициент формы, отклонение ориентации от средней по кадру) и цветояркостных (интенсивность, насыщенность цвета) характеристик частиц дождя и снега с целью обоснования решающих правил выделения пикселей частиц осадков. Данный анализ заключается в получении распределений исследуемых параметров частиц и аппроксимации их известными законами распределений с использованием метода семейства кривых Пирсона, критерия Колмогорова и симплекс-алгоритма Нелдера–Мида. Разработан алгоритм детектирования капель дождя и снежинок на видеопоследовательностях, который предполагается к использованию в составе алгоритма уменьшения видимости атмосферных осадков. Предложенный подход представлен в виде многоступенчатой классификации пикселей кадра на зоны с движущимися объектами и области неподвижного фона, искажаемые и неискажаемые частицами осадков в течение накопленных кадров. В зависимости от области, к которой относится обрабатываемый пиксель, итоговое решение об отнесении его к классу атмосферных осадков принимается с использованием предложенных решающих правил или разработанной процедуры пороговой обработки с автоматическим определением локальных пороговых значений. Выполнено экспериментальное исследование предложенного алгоритма и с использованием двухкритериального подхода определены оптимальные значения числа накопленных кадров для корректной работы алгоритма: 100 кадров для видеоизображений с дождем; и 140 кадров для видео со снегом. Выигрыш разработанного подхода по сравнению с известными по оценкам вероятностей ошибок первого и второго рода составляет до 1.7 и 9.1% соответственно.

**DOI:** 10.31857/S0132347423030019, **EDN:** DECIHE

## 1. ВВЕДЕНИЕ

Видеоизображения часто снимаются в неблагоприятных погодных условиях. Такие мешающие факторы, как дождь, снег, град, то есть атмосферные осадки в виде падающих частиц, приводят к возникновению в кадре эффекта динамических помех. Помехи такого типа могут привести к ограничению видимости объектов в кадре, уменьшению контрастности, а также к полному перекрытию падающими частицами обзора мелких деталей видеосцены.

Поскольку для обработки видеоизображений часто используются различные алгоритмы систем компьютерного зрения, то нежелательные эффекты, вызванные описанными динамическими помехами, могут привести к ошибкам и сбоям при выполнении данных алгоритмов [1]. Так, наличие атмосферных осадков способно значительно снизить качество работы алгоритмов распознавания и классификации образов, которые в настоящее время часто используются при функционировании таких автоматизированных систем, как беспилотные робототехнические комплексы или транспортные средства. Описанные мешающие факторы могут вызвать неправильное распознавание препятствия, что может привести к аварийной ситуации. Кроме того, неблагоприятные погодные явления, присутствующие в кадре, ухудшают восприятие видеосцены наблюдателем или телезрителем, снижают дальность наблюдения, уменьшают информативность видеоизображений. Поэтому целесообразным является включение в состав систем компьютерного зрения и

использование алгоритма интеллектуальной обработки видеоизображений, снятых статичными камерами, который должен решать задачи обнаружения и уменьшения видимости частиц атмосферных осадков на видеопоследовательностях.

В настоящее время известен ряд работ [2–5], в которых описаны подходы к детектированию частиц дождя или снега на видеоизображениях. Основной сложностью такого детектирования является отнесение пикселей, выделенных в результате порогового сравнения последовательных кадров, к классу частиц осадков или к классу других движущихся объектов. Каждый из известных алгоритмов отличается способом данной классификации и имеет свои особенности и недостатки.

Одним из основных шагов алгоритма, основанного на фотометрической и динамической моделях дождя [2], является применение линейного фотометрического ограничения к пикселам-кандидатам, которые обнаружены на этапе порогового сравнения и, возможно, перекрыты частицами осадков. Согласно данному ограничению, изменение $\Delta I$ значения пиксела капли в $k$-м кадре линейно связано с пикселем фона $I_{k-1}$, перекрытым этой каплей, с коэффициентом наклона линейной зависимости в пределах $0 < \beta_0 < 0.039$. В результате предварительных экспериментальных исследований установлено, что данное ограничение является излишне строгим, поэтому основным недостатком рассмотренного алгоритма является большое количество “пропущенных”, то есть необработанных дождевых пикселей. Кроме того, при разработке данного подхода рассматривались физические свойства только частиц дождя. Это ограничивает его применение по отношению к другим видам атмосферных осадков, например, в условиях снегопада.

В алгоритме, основанном на временной кластеризации пикселей только по их цветояркостным характеристикам [3], часто появляются ошибки при определении принадлежности пиксела к определенному классу. Это приводит к наличию достаточно большого количества необработанных дождевых и ошибочно обработанных недождевых пикселей. Также стоит отметить значительные вычислительные затраты, которые приводят к временной задержке при определении расположения частицы осадков: на объекте переднего плана или на фоне неподвижного дальнего плана; что затрудняет реализацию данного алгоритма в реальном времени.

Общим недостатком двух рассмотренных выше алгоритмов является то, что в них не используется информация о геометрических характеристиках капель дождя или снежинок на этапе их обнаружения. Информация о геометрических параметрах должна способствовать более точному и полному обнаружению дождевых или снеговых пикселей. Это подтверждается в процессе предварительного экспериментального исследования алгоритма [4], основанного на использовании характеристик формы частиц дождя. В данном алгоритме к областям-кандидатам, выделенным в результате процедуры порогового сравнения, применяются ограничения на форму и ориентацию этих областей, что приводит к меньшему количеству ложных срабатываний. Однако количество пропущенных частиц осадков все еще достаточно велико. Этот факт побуждает искать методы и подходы снижения данного критерия. Другим недостатком рассматриваемого алгоритма является возможность применения ограничения на форму частиц только для капель дождя, что затрудняет использование данного подхода для других видов атмосферных осадков.

Одним из способов устранения указанных недостатков известных алгоритмов обнаружения частиц осадков на видеоизображениях может являться разработка решающих правил выделения пикселей таких частиц, основанных на статистическом анализе геометрических параметров (размер, форма, ориентация) капель или снежинок [4–6]. Также для увеличения точности обнаружения данных частиц целесообразно использовать их яркостные характеристики, описанные в работах [2, 3].

Кроме того, для оптимизации использования вычислительных ресурсов и возможности реализации алгоритма в реальном времени, целесообразно разделить задачу обнаружения частиц осадков на детектирование на фоне относительно постоянного заднего плана видеосцены и обнаружение на участках кадров, содержащих множество подвижных объектов. При этом детектирование частиц осадков на областях, соответствующих заднему плану, может быть выполнено только за счет процедуры порогового обнаружения с использованием автоматически определяемых локальных пороговых значений. А разработанные решающие правила, для применения которых необходимо вычислять геометрические параметры каждой частицы-кандидата, можно использовать только в зонах изображения с подвижными объектами.

## 2. СТАТИСТИЧЕСКИЙ АНАЛИЗ ХАРАКТЕРИСТИК ЧАСТИЦ АТМОСФЕРНЫХ ОСАДКОВ НА ВИДЕОИЗОБРАЖЕНИЯХ

При первичной классификации пикселей подвижных и неподвижных объектов используют, как правило, процедуры порогового сравнения нескольких последовательных кадров [2, 5]. Так, для сопоставления значений $I$ $(x, y)$-го пиксела $(k-1)$-го, $k$-го и $(k+1)$-го кадров авторами предложено логическое выражение

$$(I_k(x,y) - I_{k-1}(x,y) \geq c_0) \vee$$
$$\vee\, (I_k(x,y) - I_{k+1}(x,y) \geq c_0),$$

где $c_0$ – глобальное пороговое значение. Двоичное изображение, сформированное в результате такой процедуры, представляет собой совокупность групп пикселей, каждая из которых лишь предположительно является частицей осадков. Для отделения таких групп-кандидатов, соответствующих каплям или снежинкам, от групп-кандидатов, соответствующих другим движущимся объектам, в состав разрабатываемого алгоритма должны входить решающие правила. С целью построения данных правил является целесообразным осуществление статистического анализа геометрических и цветояркостных характеристик частиц атмосферных осадков на видеоизображениях.

### *2.1. Процедура статистического анализа геометрических характеристик*

Так как изображения средних и больших по размеру частиц осадков обычно принимают вытянутую вниз форму, а ориентация (угол наклона по отношению к вертикальной оси) всех таких частиц в одном кадре примерно одинакова [4, 5], то наибольший интерес в качестве геометрических характеристик капель дождя и снежинок представляют размер, форма и ориентация частицы. При статистическом анализе данных параметров использованы видеоизображения, в которых движутся только частицы осадков, а другие объекты остаются неподвижными. В этом случае группы-кандидаты пикселей двоичного изображения, полученного в результате процедуры порогового сравнения трех последовательных монохромных кадров, точно являются частицами осадков. Для вычисления искомых характеристик используется метод геометрических моментов, согласно которому каждая группа-частица аппроксимируется эллипсом с центральными моментами 2-го порядка, равными центральным моментам 2-го порядка аппроксимируемой группы [5]. На рис. 1 представлен пример аппроксимации частицы осадков эллипсом.

Пусть $p$ – порядковый номер эллипса (группы). Центральные моменты 2-го порядка для $p$-й группы вычисляются по формулам [5]:

$$M_{20\_p} = \frac{1}{S_p} \sum_{(x,y)\in C_p} (x - x_{0p})^2,$$
$$M_{11\_p} = \frac{1}{S_p} \sum_{(x,y)\in C_p} (x - x_{0p})(y - y_{0p}),$$
$$M_{02\_p} = \frac{1}{S_p} \sum_{(x,y)\in C_p} (y - y_{0p})^2,$$

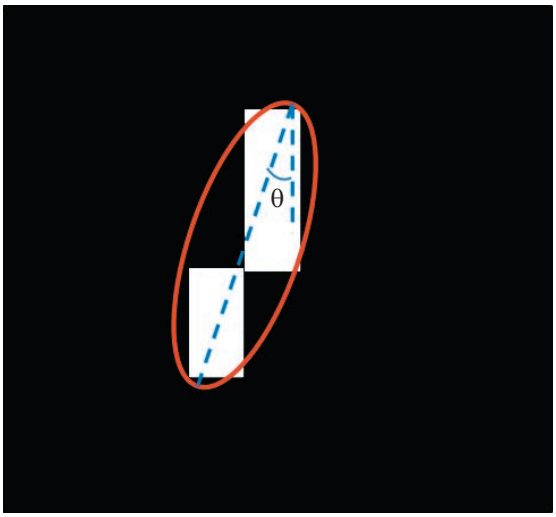


**Рис. 1.** Пример аппроксимации частицы осадков эллипсом.

где $M_{20\_p}$ – центральный момент 2-го порядка относительно координаты $x$, $M_{11\_p}$ – центральный смешанный момент 2-го порядка, $M_{02\_p}$ – центральный момент 2-го порядка относительно координаты $y$, $S_p$ – число пикселей в $p$-й группе, $(x, y)$ – координаты местоположения пиксела, $C_p$ – совокупность пикселей $p$-й группы, $(x_{0p}, y_{0p})$ – координаты местоположения центра тяжести.

Ориентация $\theta_p$, большая $a_p$ и малая $b_p$ полуоси $p$-го аппроксимирующего эллипса вычисляются согласно выражениям, представленным в работе [5], с использованием рассчитанных значений центральных моментов. Так,

$$\theta_p = \frac{1}{2}\operatorname{arctg}\left(\frac{2M_{11\_p}}{M_{02\_p} - M_{20\_p}}\right).$$

Анализируются три геометрических характеристики частицы дождя или снега:

1. Площадь $S_p$, равная числу точек в соответствующей группе.

2. Коэффициент формы $\Phi_p = a_p/b_p$.

3. Отклонение $\theta'_p$ ориентации капли или снежинки от среднего значения $\theta_0$ ориентации в кадре: $\theta'_p = \theta_p - \theta_0$.

В процессе статистического исследования выполняется построение гистограмм распределений этих параметров. Далее осуществляется аппроксимация данных гистограмм известными законами распределения. Для этого сначала методом семейства кривых Пирсона оценивается тип распределения [7–9]. Согласно данному методу, тип определяется по величине $k_П$, которая вычисляется по формуле [8]:

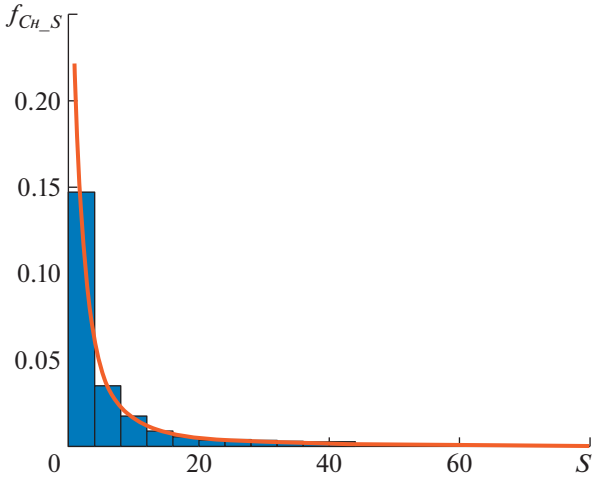


**Рис. 2.** Распределение площадей $S$ частиц снега на видеоизображениях и аппроксимирующая ФПВ $f_{Сн\_S}$.

$$k_{П} = \frac{\beta_1(\beta_2+3)^2}{4(2\beta_2-3\beta_1-6)(4\beta_2-3\beta_1)},$$

где $\beta_1$ – коэффициент асимметрии, $\beta_2$ – коэффициент эксцесса.

Затем с использованием метода минимизации целевой функции [10] симплекс-алгоритмом Нелдера–Мида [11] производится оценка параметров функции плотности вероятности (ФПВ), относящейся к типу распределения, определенного на предыдущем этапе статистического анализа. Начальные значения данных параметров рассчитаны с помощью метода моментов [12]. За целевую функцию принята функция критерия Колмогорова $K_K(\mathbf{u})$, которая количественно оценивает отклонение предполагаемого теоретического закона распределения от эмпирического следующим образом [13]:

$$K_K(\mathbf{u}) = \sqrt{N_в}\max\left|F_m(z,\mathbf{u}) - F_э(z)\right|,$$

где $N_в$ – объем выборки (в данном случае – суммарное число анализируемых частиц осадков), $F_m(z, \mathbf{u})$ – функция распределения теоретического закона, $z$ – случайная величина (в данном случае – один из анализируемых геометрических параметров), $\mathbf{u} = \{u_1, u_2, ...\}$ – параметры теоретического закона распределения, $F_э(z)$ – эмпирическая функция распределения.

Таким образом, результатом статистического анализа геометрических характеристик изображений частиц осадков являются теоретические законы распределения каждой из данных характеристик, которые используются для обоснования решающих правил обнаружения капель дождя или снежинок на видеоизображениях.

### *2.2. Результаты статистического анализа геометрических характеристик*

На рис. 2 приведены гистограмма распределения площадей $S$ снежинок на видеоизображенияхи аппроксимирующая это распределение ФПВ $f_{Сн\_S}$.

Для данного распределения величина $k_{П} = 3336.7 > 1$, поэтому оно относится к VI типу распределений Пирсона [8], частным случаем которого является бета-распределение II рода [14].

С использованием метода минимизации целевой функции [10] симплекс-алгоритмом Нелдера–Мида [11] установлено, что для распределения площадей $S$ частиц снега на видеоизображениях параметры формы бета-распределения II рода принимают значения $u_{Сн\_S} = 1.86$, $\nu_{Сн\_S} = 0.83$.

Из анализа рис. 2 ясно, что, чем крупнее снежинка, тем менее вероятно ее появление в кадре. При этом размер снежинок может принимать значения в широких пределах, и, так как форма частиц осадков в значительной степени зависит от их размера, то построение гистограммы распределения значений Ф является целесообразным для нескольких диапазонов изменения $S$. Поскольку наибольший интерес для статистического анализа представляют частицы средней и крупной величины, то граничные значения диапазонов $S$ выбраны по уровням значимости 0.7, 0.95 и 0.99 (как часто используемые в литературе, например, в [12]), что соответствует значениям $S$ в 7, 69 и 484 пикселя. Математически это можно записать в виде:

$$P(7 \le S < 69) = F_{Сн\_S}(69) - F_{Сн\_S}(7) = 0.95 - 0.7,$$
$$P(69 \le S < 484) =$$
$$= F_{Сн\_S}(484) - F_{Сн\_S}(69) = 0.99 - 0.95,$$

где $P(7 \le S < 69)$ – вероятность появления в кадре видеоизображения снежинки размером $(7 \le S < 69)$, $F_{Сн\_S}$ – функция распределения, соответствующая ФПВ $f_{Сн\_S}$ [14].

Гистограммы распределений значений Ф для первого $(7 \le S < 69)$ и второго $(69 \le S < 484)$ диапазонов изменения площади $S$ частиц снега, а также аппроксимирующие эти распределения ФПВ $f_{Сн\_Ф1}$ и $f_{Сн\_Ф2}$ представлены на рис. 3 и 4 соответственно.

Для данных распределений коэффициент $k_{П}$ принимает значения 1.47 и 9.59 соответственно, следовательно, эти распределения относятся к VI типу распределений Пирсона [8] и аппроксимируются частным случаем данного типа – бета-распределением II рода [14]. Параметры ФПВ, аппроксимирующей распределение коэффициентов формы снежинок первого диапазона $S$, имеют значения $u_{Сн\_Ф1} = 23.17$, $v_{Сн\_Ф1} = 12.28$. Параметры

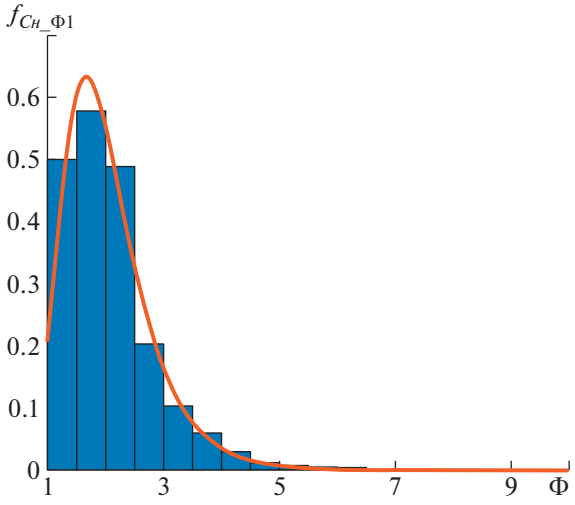


**Рис. 3.** Распределение коэффициентов формы Ф частиц снега для первого диапазона $S$ и аппроксимирующая ФПВ $f_{Сн\_Ф1}$.

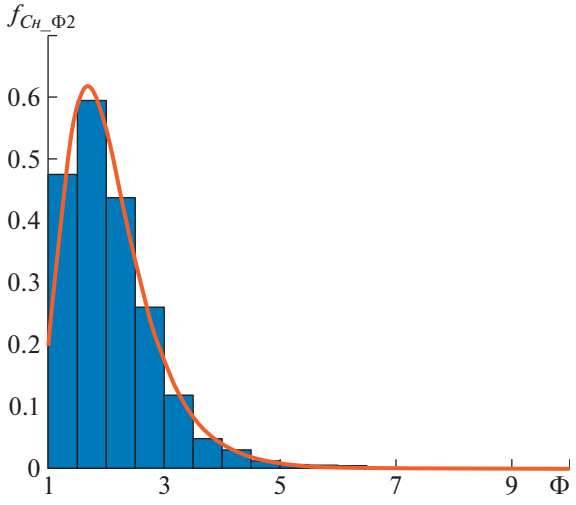


**Рис. 4.** Распределение коэффициентов формы Ф частиц снега для второго диапазона$S$ и аппроксимирующая ФПВ $f_{Сн\_Ф2}$.

ФПВ, аппроксимирующей распределение коэффициентов формы снежинок второго диапазона $S$, имеют значения $u_{Сн\_Ф2} = 22{,}73$, $v_{Сн\_Ф2} = 11.9$.

Гистограмма распределения значений отклонения θ' ориентации снежинок от среднего значения их ориентации в кадре и аппроксимирующая это распределение ФПВ $f_{Сн\_\theta}$ представлены на рис. 5.

Из анализа рис. 5 видно, что отклонения ориентации частиц снега в большинстве случаев имеют малые по модулю значения. Для данного распределения значение $k_П = -0.007 < 0$, т.е. распределение относится к I типу распределений Пирсона [8], однако значительно лучший результат с точки зрения минимума функции критерия Колмогорова [13] достигается при аппроксимации гистограммы данного распределения кривой Пирсона IV типа [14]. Параметры данного распределения также рассчитаны с использованием метода минимизации целевой функции симплекс-алгоритмом Нелдера–Мида: $b_0 = -316.14$, $b_1 = 0.9$, $b_2 = -0.84$, $c_0 = 0.44$.

В процессе аналогичного статистического анализа геометрических характеристик капель дождя на видеоизображениях установлено, что распределение площадей $S$ частиц аппроксимируется бета-распределением II рода с параметрами формы $u_{Д\_S} = 1.16$, $v_{Д\_S} = 0.72$. Распределение значений коэффициента формы Ф капель размером из первого диапазона ($6 \le S < 76$) может быть аппроксимировано бета-распределением II рода с параметрами формы $u_{Д\_Ф1} = 15.35$, $v_{Д\_Ф1} = 6.91$, а для частиц дождя площадью из второго диапазона ($76 \le S < 727$) – обобщенным бета-распределением I рода с границами диапазона возможных значений $Ф\gamma_{Д\_Ф2} = 1$, $\beta_{Д\_Ф2} = 16$ и параметрами формы $u_{Д\_Ф2} = 1.34$, $v_{Д\_Ф2} = 7.37$. Распределение значений отклонения θ' ориентации капель дождя от среднего значения их ориентации в кадре аппроксимируется ФПВ IV типа кривых Пирсона с параметрами $b_0 = -64.15$, $b_1 = 0.25$, $b_2 = -0.36$, $c_0 = 46.52$.

Сравнивая результаты статистического анализа геометрических характеристик изображений частиц дождя с аналогичными результатами для частиц снега, можно отметить схожесть форм гистограмм распределений данных характеристик, а также форм кривых, аппроксимирующих эти

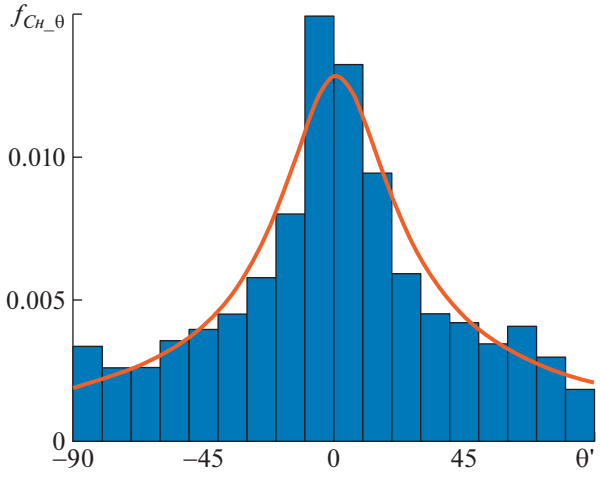


**Рис. 5.** Распределение значений отклонения θ' ориентации снежинок от среднего значения их ориентации в кадре и аппроксимирующая ФПВ $f_{Сн\_\theta}$.

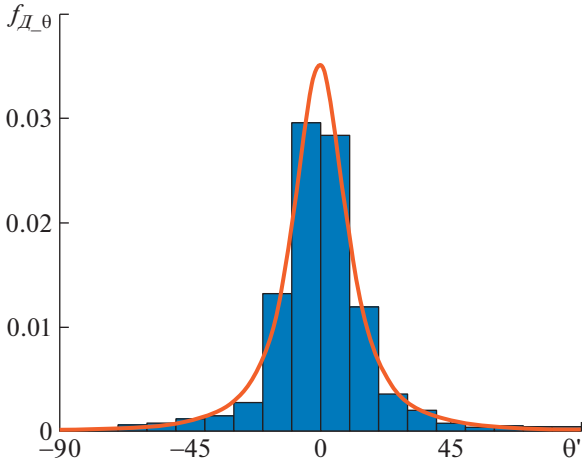


**Рис. 6.** Распределение значений отклонения θ' ориентации капель дождя от среднего значения их ориентации в кадре и аппроксимирующая ФПВ $f_{Д\_\theta}$.

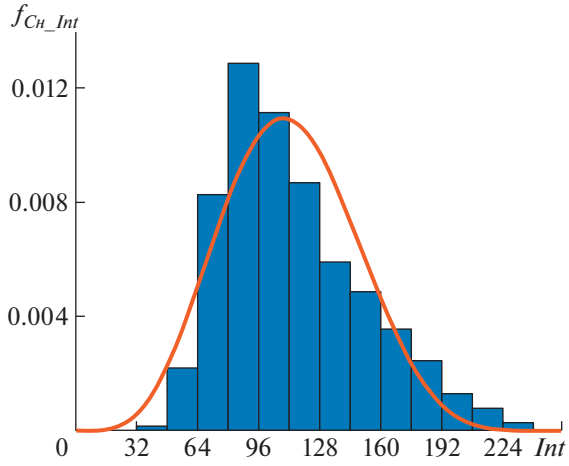


**Рис. 7.** Распределение значений интенсивности *Int* пикселей частиц снега и аппроксимирующая ФПВ $f_{Сн\_In}$.

гистограммы. Однако наблюдаются некоторые отличия. Так, по сравнению с частицами снега, капли дождя имеют более вытянутую форму и меньшие по модулю отклонения ориентации от среднего значения по кадру. Эти отличия хорошо видны, например, при сравнении приведенных на рис. 5 и 6 распределений отклонений ориентаций снежинок и капель соответственно.

### *2.3. Процедура и результаты статистического анализа цветояркостных характеристик*

Частицы осадков также могут характеризоваться цветояркостными параметрами, к которым относятся интенсивность, цветовой тон, насыщенность цвета и др. [15].

При статистическом анализе цветояркостных характеристик, так же, как и геометрических, использованы видеоизображения, в которых движутся только атмосферные осадки. В результате порогового сравнения трех последовательных монохромных кадров ($(k-1)$-го, $k$-го и $(k+1)$-го) формируется двоичное изображение, в котором каждый пиксель со значением 1 относится к частицам осадков. Далее полноцветный $k$-й кадр преобразуется из цветового пространства RGB в цветовое пространство HSI [15]. В изображениях каналов интенсивности (I) и насыщенности цвета (S) выбираются только те пиксели, которые соответствуют пикселам двоичного изображения со значением 1.

Таким образом, анализируемыми цветояркостными параметрами пикселов частиц осадков являются:

1. Интенсивность *Int*.

2. Насыщенность цвета *St*.

На следующем этапе статистического исследования, аналогично анализу геометрических характеристик, осуществляется построение гистограмм распределений значений *Int* и *St* пикселей, выбранных на предыдущем этапе, и аппроксимация данных гистограмм известными законами распределения с использованием метода семейства кривых Пирсона [7–9] и метода минимизации целевой функции [10] симплекс-алгоритмом Нелдера–Мида [11].

Гистограмма распределения значений интенсивности *Int* пикселей частиц снега на видеоизображениях и аппроксимирующая данное распределение ФПВ $f_{Сн\_Int}$ приведены на рис. 7.

При этом величина $k_П = -0.31 < 0$, следовательно, данное распределение относится к I типу кривых Пирсона и аппроксимируется частным случаем этого типа – обобщенным бета-распределением I рода, для которого границы диапазона возможных значений $Int\gamma_{Сн\_Int} = 0$, $\beta_{Сн\_Int} = 255$ и параметры формы $u_{Сн\_Int} = 5.47$, $v_{Сн\_Int} = 6.99$.

Гистограмма распределения значений насыщенности цвета *St* пикселей снежинок на видеоизображениях и аппроксимирующая данное распределение ФПВ $f_{Сн\_St}$ приведены на рис. 8.

Из анализа данного рисунка можно отметить, что насыщенность цвета частиц снега на видеоизображениях смещена в область малых значений. Величина $k_П = 0.55$, $0 < k_П < 1$, что соответствует IV типу кривых Пирсона [8]. Однако значительно более точный результат аппроксимации с точки зрения минимума функции критерия Колмогорова достигается при использовании кривой Пирсона I типа – обобщенного бета-распределения I рода с границами диапазона воз-

можных значений $St\gamma_{Сн\_St} = 0$, $\beta_{Сн\_St} = 255$ и параметрами формы $u_{Сн\_St} = 5.17$, $v_{Сн\_St} = 37.01$.

При исследовании цветояркостных характеристик пикселей капель дождя аналогичным образом установлено, что распределения значений интенсивности *Int* аппроксимируется ФПВ обобщенного бета-распределения I рода с границами диапазона $\gamma_{Д\_Int} = 0$, $\beta_{Д\_Int} = 255$ и параметрами формы $u_{Д\_Int} = 6.54$, $v_{Д\_Int} = 10.21$, т.е. распределение интенсивностей для пикселей частиц дождя смещено в область меньших значений *Int* по сравнению с аналогичной гистограммой для частиц снега. Распределение значений насыщенности цвета пикселей капель, так же, как и для снежинок, смещено в область низких значений и может быть аппроксимировано ФПВ бета-распределения II рода с параметрами формы $u_{Д\_St} = 63.11$, $v_{Д\_St} = 2.92$.

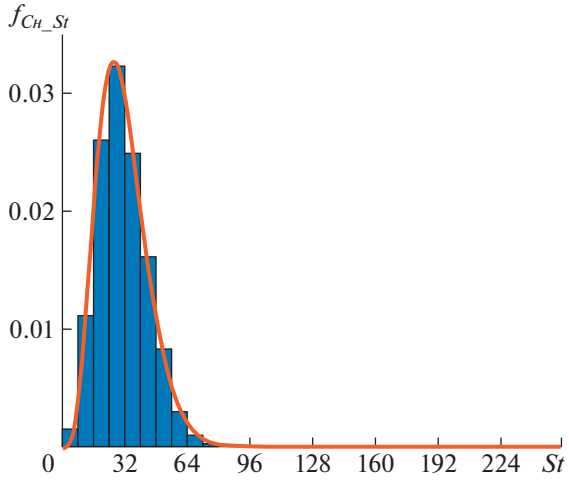


**Рис. 8.** Распределение значений насыщенности цвета *St* пикселей снежинок и аппроксимирующая ФПВ $f_{Сн\_St}$.

### *2.4. Обоснование решающих правил обнаружения частиц осадков среди других движущихся объектов на видеоизображениях*

На основе статистического анализа характеристик частиц атмосферных осадков на видеоизображениях, описанного выше, выполнено обоснование решающих правил детектирования таких частиц в кадре.

Во-первых, является целесообразным ограничить размер групп-кандидатов пикселей, формируемых на предыдущих этапах предложенного алгоритма, на уровне значимости $\alpha = 0.99$. Поскольку распределение площадей частиц снега аппроксимируется бета-распределением II рода, то с помощью функции распределения $F_{Сн\_S}(S)$ этого закона определено, что данному уровню значимости соответствует значение $S$ в 484 пиксела:

$$F_{Сн\_S}(484) = P(S < 484) = 0.99.$$

Данное ограничение означает, что группы-кандидаты пикселей, сформированные в результате этапа порогового сравнения, площадь которых $S \geq 484$, не будут считаться предложенным алгоритмом частицами снега, и, следовательно, не будут подвергаться дальнейшей обработке.

Во-вторых, изображения средних и больших по размеру частиц осадков, как правило, принимают вытянутую вниз форму [4, 5], поэтому необходимо ограничение коэффициента формы групп-кандидатов таким образом, чтобы наименее вытянутые из них не принимались разрабатываемым алгоритмом за осадки. При уровне значимости $\alpha = 0.1$ с помощью функций распределения $F_{Сн\_Ф1}(\Phi)$ и $F_{Сн\_Ф2}(\Phi)$ законов, аппроксимирующих распределения коэффициентов формы Ф для первого ($7 \leq S < 69$) и второго ($69 \leq S < 484$) диапазонов площади $S$ частиц снега, установлены граничные значения Ф, равные 1.23 и 1.24 соответственно:

$$F_{Сн\_Ф1}(1.23) = P(\Phi < 1.23) = 0.1,$$
$$F_{Сн\_Ф2}(1.24) = P(\Phi < 1.24) = 0.1.$$

В-третьих, поскольку ориентация средних и крупных частиц осадков в одном кадре практически не изменяется [4], является целесообразным ограничение значений θ' таким образом, чтобы группы-кандидаты пикселей со значительным отклонением ориентации от среднего ее значения в кадре не принимались алгоритмом за капли или снежинки. Предложено двустороннее ограничение θ' по уровням значимости 0.05 и 0.95 (так как в этом случае суммарный уровень ограничения равен 0.1, как и при ограничении Ф). Данным уровням для частиц снега соответствуют значения отклонения ориентации, равные –67.83° и 70.76° соответственно, которые определены при помощи функции распределения $F_{Сн\_\theta}(\theta')$ IV типа кривых Пирсона:

$$F_{Сн\_\theta}(-67.8°) = P(\theta' < -67.8°) = 0.05,$$
$$F_{Сн\_\theta}(70.8°) = P(\theta' < 70.8°) = 0.95.$$

В-четвертых, поскольку пиксели частиц осадков, как правило, имеют большие значения интенсивности по сравнению с пикселами фона [2, 3], является целесообразным ограничение величины *Int* пикселей групп-кандидатов по уровню значимости 0.05. Данный уровень в случае пикселей частиц снега соответствует значению $Int = 56.57$:

$$F_{Сн\_Int}(56.6) = P(Int < 56.6) = 0.05.$$

В-пятых, известно [3], что частицы атмосферных осадков преимущественно белого цвета, следовательно, насыщенность цвета *St* пикселей

этих частиц должна иметь малые значения. Поэтому целесообразно ввести ограничение значений *St* пикселей групп-кандидатов по уровню значимости 0.95, что для пикселей снежинок соответствует величине $St = 54.51$:

$$F_{Сн\_St}(54.5) = P(St < 54.5) = 0.95.$$

Объединяя представленные выше ограничения геометрических характеристик групп-кандидатов пикселей для частиц снега, построим первое решающее правило, которое имеет вид:

$$\begin{gathered}(S_p \geq 484) \vee ((7 \leq S_p < 69) \wedge (\Phi_p < 1.23)) \vee \\ \vee ((69 \leq S_p < 484) \wedge (\Phi_p < 1.24)) \vee \\ \vee ((69 \leq S_p < 484) \wedge ((\theta'_p < -67.8°) \vee (\theta'_p > 70.8°))).\end{gathered}$$

Представленное решающее правило предназначено для проверки нулевой гипотезы, согласно которой *p*-я группа-кандидат пикселей, обнаруженная при пороговом сравнении на предыдущем этапе алгоритма, является частицей снега. В отношении групп-кандидатов, которые удовлетворяют данному логическому выражению, алгоритмом принимается решение, согласно которому они не являются снежинками, и, следовательно, не будут подвергаться последующей обработке, то есть принимается конкурирующая гипотеза. При невыполнении условия решающего правила принимается нулевая гипотеза.

При объединении описанных выше ограничений цветояркостных параметров пикселей групп-кандидатов получим второе решающее правило для потенциальных пикселей частиц снега:

$$(Int_p(x, y) < 56,6) \vee (St_p(x, y) \geq 54.5).$$

Данное решающее правило предназначено для проверки нулевой гипотезы, согласно которой (*x*, *y*)-й пиксель *p*-й группы-кандидата, которая принята за частицу снега первым решающим правилом, является частью снежинки. Если характеристики пиксела удовлетворяют логическому условию, то принимается конкурирующая гипотеза, то есть данный пиксель не относится к частице снега, и он не будет подвергаться дальнейшей обработке алгоритмом, заключающейся в уменьшении видимости этой частицы. Если характеристики данного пиксела не удовлетворяют данному условию, то принимается нулевая гипотеза, и далее алгоритм интеллектуальной обработки видеоизображений применит к этому пикселу процедуру уменьшения видимости.

Обоснование решающих правил обнаружения капель дождя на видеоизображениях выполнено аналогичным способом. Тогда логическое условие первого решающего правила, основанного на статистическом анализе геометрических характеристик, имеет вид:

$$\begin{gathered}(S_p \geq 727) \vee ((6 \leq S_p < 76) \wedge (\Phi_p < 1.29)) \vee \\ \vee ((76 \leq S_p < 727) \wedge (\Phi_p < 1.45)) \vee \\ \vee ((76 \leq S_p < 727) \wedge ((\theta'_p < -28.9°) \vee (\theta'_p \geq 31.7°))).\end{gathered}$$

Логическое условие второго решающего правила, основанного на статистическом анализе цветояркостных характеристик пикселей частиц дождя:

$$(Int_p(x, y) < 52.9) \vee (St_p(x, y) \geq 81).$$

## 3. СТАТИСТИЧЕСКИЙ АНАЛИЗ ИЗМЕНЕНИЙ ВО ВРЕМЕНИ ЗНАЧЕНИЙ ПИКСЕЛЕЙ ОБЛАСТЕЙ ВИДЕОКАДРОВ, В КОТОРЫХ НАБЛЮДАЕТСЯ ПРОХОЖДЕНИЕ АТМОСФЕРНЫХ ОСАДКОВ

При обнаружении частиц осадков на относительно неподвижном заднем плане видеоизображения важной задачей алгоритма является принятие решения о том, проходят ли дождь или снег в данном участке (в наилучшей реализации алгоритма – в каждом пикселе) кадра видеосцены. С целью обоснования соответствующего решающего правила выполнен анализ, описанный в данном разделе работы.

### *3.1. Процедура и результаты статистического анализа изменений во времени значений пикселей*

В процессе анализа, приведенного в данном разделе, каждое отдельное распределение значений получено при наблюдении за отдельным пикселем в течение определенного интервала времени, как правило, в 80...200 кадров при частоте видео 25 кадров/с. При этом, аналогично предыдущему исследованию, использованы области видеоизображений с дождем и снегом на фоне постоянного во времени дальнего плана, пиксели которых изменяют свое значение только при прохождении частицы осадков или из-за незначительных колебаний освещенности и цифровых шумов. Таким образом построены временные распределения $p(I)$ значений пикселей областей видеокадров, в которых в течение данного времени наблюдаются атмосферные осадки. Примеры гистограмм распределений $p(I)$ для трех различных пикселей изображены на рис. 9(а, б, в).

Согласно рис. 9(а, б, в), распределения значений во времени рассматриваемых пикселей являются асимметричными бимодальными или асимметричными унимодальными. Так как прохождение капли дождя или снежинки через точку изображения сопровождается скачком яркости этой точки [2, 3], то ее значения в данные моменты времени соответствуют меньшей моде или хвосту распределения $p(I)$. При этом отношение числа кадров, в которых данный пиксель пере-

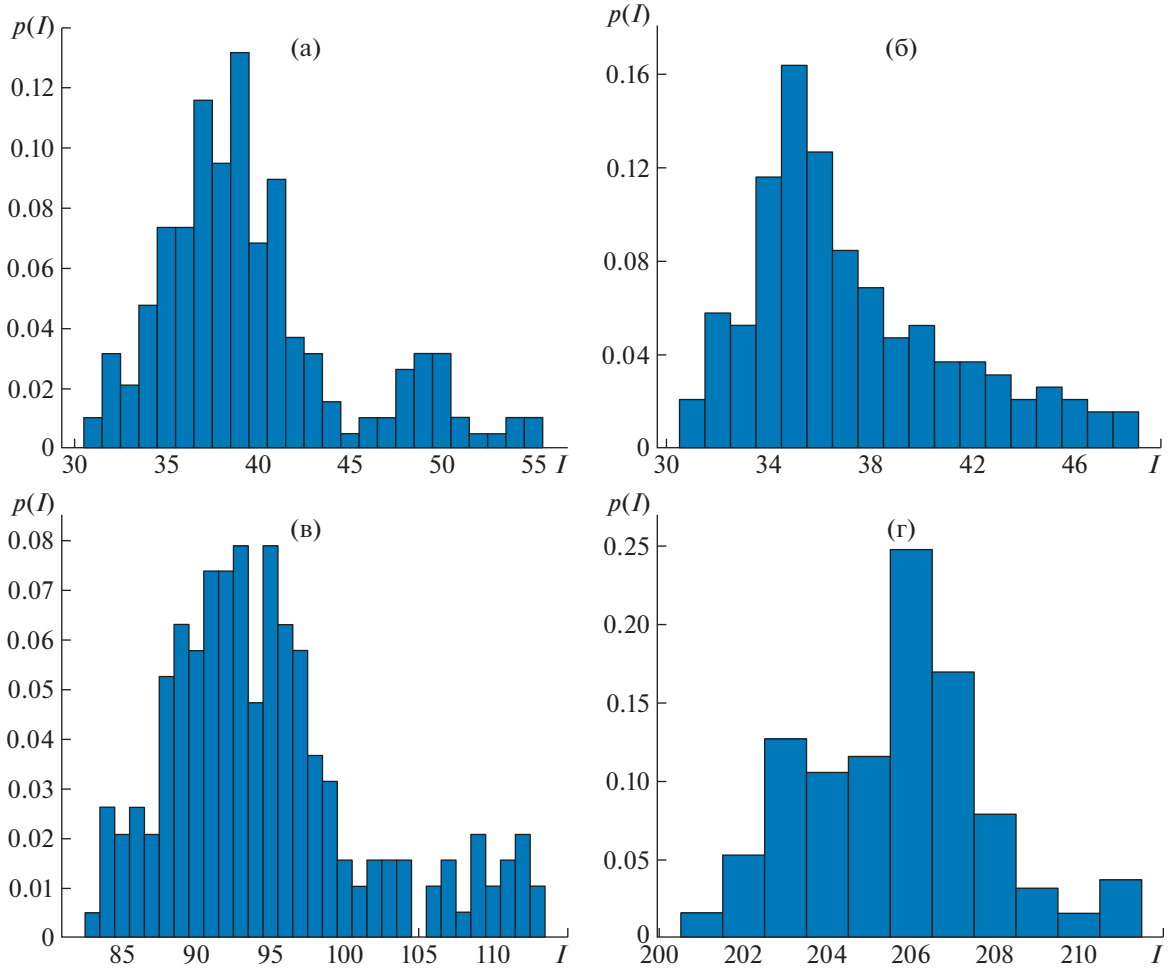


**Рис. 9.** Гистограммы распределений $p(I)$ во времени значений $I$ пиксела, взятого с: (а, б) области кадра с дождем; (в) области кадра со снегом; (г) неискаженной области кадра.

крыт частицами осадков, к числу кадров, в которых этот пиксель не искажен, зависит от интенсивности осадков и, как правило, не превышает 1 : 4...5. Данный факт объясняет положительную асимметрию, наблюдаемую на всех полученных временных распределениях значений пикселей. Для сравнения, на рис. 9(г) представлен пример гистограммы распределения $p(I)$ пиксела, не искажаемого каплями или снежинками в процессе времени наблюдения. Подобные распределения, как правило, характеризуются симметрией и унимодальностью.

Обобщая вышеизложенный анализ, можно отметить, что по форме временного распределения $p(I)$ возможно оценить, к какой области кадра относится соответствующий пиксель: к искажаемой – той, где наблюдаются атмосферные осадки за определенный промежуток времени; или к не искажаемой осадками.

### *3.2. Обоснование решающего правила обнаружения частиц осадков на фоне заднего плана видеоизображений*

Одной из характеристик формы распределения случайной величины является коэффициент Сарле, предназначенный для оценки степени мультимодальности. Данный параметр описывается выражением [16]:

$$K_S = \frac{A^2 + 1}{E + \dfrac{3(n-1)^2}{(n-2)(n-3)}},$$

где $A$ – коэффициент асимметрии, $E$ – коэффициент эксцесса, $n$ – размер выборки. При $K_S < 5/9 \approx \approx 0.555$ исследуемое распределение полагают унимодальным, при $K_S > 0.555$ – бимодальным [16].

Авторами работы [16] отмечено, что коэффициент Сарле $K_S > 0.555$ и для сильно асимметричных унимодальных распределений. Однако, как отмечено выше, распределения во времени зна-

чений пикселей искажаемых осадками участков кадра в случае своей унимодальности обязательно являются сильно асимметричными, поэтому указанная неточность критерия Сарле позволяет использовать его для обнаружения искажаемых и неискажаемых областей кадров. При этом, с учетом особенностей рассматриваемой задачи, предложена модификация данного коэффициента. Так, положительную асимметрию, являющуюся особенностью распределений $p(I)$, необходимо учитывать в знаке коэффициента. Также в случае получения пограничного результата ($K_S \approx 0.5...0.6$) диапазон $D$ значений рассматриваемой точки кадра за время построения $p(I)$ может быть использован для принятия правильного решения о классификации пиксела. Тогда модифицированный коэффициент Сарле принимает вид:

$$K_m = \operatorname{sign}(A) K_S D/8,$$

где 8 – минимально заметное изменение значения точки видеоизображения при прохождении через нее капли или снежинки.

Согласно предложенному критерию, при $K_m > 0.555$ полагается, что исследуемый пиксель принадлежит искажаемому осадками участку кадра (за накопленное число кадров частицы несколько раз затрагивают данную точку), при $K_m < 0.555$ – пиксель принадлежит области, в которой не наблюдаются осадки (возможно, что видеосцена совсем не содержит дождя или снега), следовательно, данная точка не подлежит последующей обработке.

Далее необходимо принять решение, затронуты ли пиксели, относящиеся к искажаемым участкам видеоизображения, частицами в текущем кадре. На данном этапе предусматривается проведение пороговой обработки с автоматическим вычислением локальных пороговых значений для каждой точки. Необходимость использования локальных, а не глобального для всего кадра порога объясняется тем, что в некоторых случаях значения пикселей краев капли или снежинки превышают значения соответствующих неискаженных точек всего на 4...7 единиц (в 8-битном представлении). При этом такие незначительные перепады легко спутать с колебаниями яркости пикселей дальнего плана, вызванными, например, флуктуациями освещенности, дрожанием листьев или цифровыми шумами.

На основе анализа распределений значений искаженных и неискаженных пикселей (рис. 9), локальные пороговые значения предложено вычислять по формуле:

$$c(x,y) = 2\hat{I}(x,y) - \min(I(x,y)),$$

где $\min(I(x, y))$ – минимальное значение, принимаемое $(x, y)$-м пикселем за время наблюдения ($n$ кадров), $\hat{I}(x,y)$ – мода распределения $p(I)$ для $(x, y)$-го пиксела.

Таким образом, решающее правило обнаружения частиц осадков на фоне заднего плана видеоизображений принимает вид:

$$(K_m(x,y) < 0.555) \vee (I(x,y) < c(x,y)).$$

Пиксели, удовлетворяющие данному логическому условию, принимаются за пиксели заднего плана кадра, не перекрытые частицами осадков в текущем кадре. В противном случае, пиксели считаются затронутыми каплями или снежинками и подлежат обработке алгоритмом уменьшения видимости данных частиц.

## 4. ОПИСАНИЕ ПРЕДЛОЖЕННОГО АЛГОРИТМА ОБНАРУЖЕНИЯ ЧАСТИЦ ОСАДКОВ

В основе разработанного алгоритма обнаружения частиц осадков на видеоизображениях лежит многоступенчатая классификация пикселей:

1. Разделение видеокадра на зоны с движущимися объектами (люди, автомобили и др.) и на участки с относительно постоянным задним планом в соответствии с диапазоном $D(x, y)$ изменения во времени за $n$ кадров значений $(x, y)$-го пиксела (выполняется один раз за $n$ кадров).

Для областей с движущимися объектами:

2. Первичная классификация пикселей на движущиеся и неподвижные объекты с использованием порогового сравнения трех последовательных кадров.

3. Объединение 8-связных групп пикселей двоичного изображения, полученного на предыдущем этапе, в группы-кандидаты.

4. Расчет геометрических параметров (площадь $S_p$, коэффициент формы $\Phi_p$, отклонение $\theta'_p$ ориентации от средней по кадру) для каждой группы-кандидата.

5. Применение решающего правила, основанного на геометрических характеристиках, для каждой группы-кандидата.

6. Расчет цветояркостных параметров (интенсивность $Int(x, y)$, насыщенность цвета $St(x, y)$) для каждого пиксела групп-кандидатов, прошедших отбор на предыдущем этапе.

7. Применение решающего правила, основанного на цветояркостных характеристиках, для каждого пиксела из предыдущего этапа.

Для областей с неподвижным задним планом:

8. Расчет модифицированного коэффициента Сарле $K_m(x, y)$для каждого пиксела данных областей (выполняется один раз за $n$ кадров).

9. Расчет локальных пороговых значений $c(x, y)$ для каждого пиксела данных областей.

10. Применение решающего правила обнаружения частиц осадков на фоне заднего плана видеоизображений.

Из описания предложенного алгоритма следует, что этап 8, на котором выполняется расчет модифицированного коэффициента Сарле, являющегося характеристикой формы временного распределения значений пиксела, выполняется один раз за $n$ кадров. Следовательно, параметр $n$ определяет объем выборки, от которого зависит точность оценки формы данного распределения [12]. Неправильно подобранное количество накопленных кадров приводит к росту числа необнаруженных алгоритмом частиц осадков или, наоборот, к увеличению числа пикселей, неправильно отнесенных к перекрытым каплями или снежинками. Таким образом, одной из целей экспериментальных исследований, представленных в следующем разделе, является определение оптимального значения $n$ для видеоизображений с осадками в виде дождя и снега.

## 5. ЭКСПЕРИМЕНТАЛЬНЫЕ ИССЛЕДОВАНИЯ ПРЕДЛОЖЕННОГО АЛГОРИТМА

С целью анализа качества алгоритмов обнаружения используют, как правило, вероятности ошибок первого и второго рода, выражения для оценки которых имеют вид [17, 18]:

$$K_1 = \frac{N_{0-}}{N_0}, \quad K_2 = \frac{N_{ч-}}{N_ч},$$

где для исследуемого алгоритма $N_{0-}$ и $N_{ч-}$ – число незатронутых и затронутых осадками пикселей соответственно, которые ошибочно отнесены исследуемым алгоритмом к противоположным классам, $N_0$ и $N_ч$ – количество всех обрабатываемых точек, не относящихся и относящихся к частицам соответственно.

Установлено, что выбор числа кадров $n$ влияет на точность определения как значения модифицированного коэффициента Сарле, так и локальных пороговых значений, поэтому вероятности ошибок первого и второго рода оценены отдельно для восьмого ($K_{18}$ и $K_{28}$) и девятого ($K_{19}$ и $K_{29}$) этапов алгоритма.

Поскольку применяемые критерии качества являются конфликтующими, для определения оптимального решения (числа накопленных кадров) целесообразно использовать двухкритериальный подход [19]. В данном методе осуществляется максимизация или минимизация весовой суммы выбранных показателей качества, называемой целевой функции $J$. При этом весовые коэффициенты для вероятностей ошибок первого рода определим чуть большими, чем для вероятностей ошибок второго рода, с целью снизить количество точек, неправильно классифицированных алгоритмом как затронутые частицами осадков, и затем подвергшихся ненужной обработке. Такой выбор приводит к целевой функции, описываемой выражением:

$$J = 0.3K_{18} + 0.2K_{28} + 0.3K_{19} + 0.2K_{29}.$$

При вычислении $K_{18}$ и $K_{28}$ использованы временные распределения 10000 пикселей (10000 точек “длительностью” $n$ кадров каждая), принадлежащих участкам с атмосферными осадками кадров видеоизображений, и 10000 временных распределений пикселей участков видеосцены с отсутствием дождя или снега в течение $n$ кадров. А $K_{19}$ и $K_{29}$ вычислены на основе данных о 500 точках, искаженных и неискаженных частицами осадков на текущих кадрах. Небольшое число исследуемых пикселей при оценке девятого этапа алгоритма обусловлено необходимостью “зрительного” отбора данных точек. Значения $K_{18}$, $K_{28}$, $K_{19}$, $K_{29}$ и $J$ рассчитаны при накоплении (то есть изменении $n$) от 40 до 210 кадров с частотой 25 кадров/с отдельно для видеоизображений с осадками в виде дождя и снега.

**Таблица 1.** Результаты сравнения алгоритмов при обработке областей с движущимися объектами

| Алгоритм | $K_1$, % | $K_2$, % |
|---|---|---|
| Алгоритм [2] | 4.23 | 12.74 |
| Алгоритм [3] | 5.28 | 7.82 |
| Алгоритм [4] | 3.64 | 10.19 |
| Предложенный алгоритм | 3.55 | 6.13 |

**Таблица 2.** Результаты сравнения алгоритмов при обработке областей с неподвижным задним планом

| Алгоритм | $K_1$, % | $K_2$, % |
|---|---|---|
| Алгоритм [2] | 3.27 | 12.15 |
| Алгоритм [3] | 3.49 | 7.53 |
| Алгоритм [4] | 2.8 | 10.31 |
| Предложенный алгоритм | 2.7 | 3.05 |

В результате экспериментальных исследований предложенного алгоритма, реализованного в среде MATLAB [20], определены оптимальные значения параметра $n$: 100 кадров для осадков в виде дождя и 140 кадров – в виде снега. Данное различие обусловлено более быстрым падением капли дождя в сравнении со снежинкой, и, как

следствие, необходимостью большего накопления кадров для обработки видеопоследовательностей со снегом.

Также выполнены сравнения предложенного и рассмотренных известных подходов к детектированию атмосферных осадков на областях видеоизображений, содержащих и не содержащих движущиеся объекты. Значения критериев качества сравниваемых алгоритмов приведены в табл. 1 и 2.

При обработке участков видеокадров с движущимися объектами установлен выигрыш предложенного алгоритма по критерию $K_2$ в пределах 1.7...6.6%, а по критерию $K_1$ – до 1.7%. При обнаружении частиц осадков на неподвижном заднем плане предложенный подход позволяет уменьшить оценку вероятности ошибок второго рода на 4.5...9.1% при неизменной или немного улучшенной (до 0.8%) оценке вероятности ошибок первого рода.

## 5. ЗАКЛЮЧЕНИЕ

Предложен алгоритм детектирования капель дождя и снежинок на видеоизображениях, отличающийся от известных подходов обоснованием решающих правил выделения соответствующих пикселей с использованием результатов статистического анализа геометрических, цветояркостных и временных характеристик частиц атмосферных осадков, а также разработанной процедурой многоступенчатой классификации пикселей. Так, на первом этапе производится разделение кадра на участки с подвижными объектами и области с относительно постоянным во времени задним планом. Далее для участков первого типа производится классификация пикселей на движущиеся и неподвижные объекты с использованием порогового сравнения трех последовательных кадров. Для групп пикселей, соответствующих подвижным объектам, в том числе частицам осадков, выполняется расчет геометрических и цветояркостных параметров. С помощью разработанных решающих правил осуществляется финальная классификация пикселей на точки, принадлежащие каплям/снежинкам, и точки других движущихся объектов. Для областей кадров второго типа – с неподвижным задним планом – выполняется детектирование участков, в которых наблюдается прохождение осадков в течение определенного количества накопленных кадров, а затем рассчитываются локальные пороговые значения, с использованием которых выполняется обнаружение точек, затронутых частицами осадков в каждом текущем кадре. По результатам экспериментального исследования определены оптимальные значения числа накопленных кадров: 100 кадров для видеоизображений с дождем; и 140 кадров для видео со снегом. Установлено, что при обработке участков видеоизображений с подвижными объектами выигрыш разработанного алгоритма по сравнению с известными подходами [2–4] составляет 1.7...6.6% по критерию $K_2$ и до 1.7% по критерию $K_1$. А при обнаружении частиц осадков на неподвижном заднем плане подтвержден выигрыш 4.5...9.1% по вероятности ошибок второго рода.

## СПИСОК ЛИТЕРАТУРЫ


1. *Визильтер Ю.В., Желтов С.Ю., Бондаренко А.В., Осоков М.В., Моржин А.В.* Обработка и анализ изображений в задачах машинного зрения: Курс лекций и практических занятий. М.: Физматкнига, 2010. 672 с.
2. *Garg K., Nayar S.K.* Vision and rain // International Journal of Computer Vision. 2007. V. 75. № 1. P. 3–27.
3. *Jia Z., Wang H., Caballero R.E., Xiong Z., Zhao J., Finn A.* A two-step approach to see-through bad weather for surveillance video quality enhancement // Machine Vision and Applications. 2012. V. 23. № 6. P. 1059–1082.
4. *Brewer N., Liu N.* Using the shape characteristics of rain to identify and remove rain from video // Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR). Berlin, Heidelberg: Springer, 2008. P. 451–458.
5. *Bossu J., Hautiere N., Tarel J.* Rain or Snow Detection in Image Sequences through use of a Histogram of Orientation of Streaks // International Journal of Computer Vision. 2011. № 93.
6. *Кириллов С.Н., Покровский П.С., Бауков А.А.* Алгоритм уменьшения влияния атмосферных осадков на качество видеоизображений в системах управления // Сб. тез. докл. научно-техн. конф. “Техническое зрение в системах управления – 2019”. 2019. С. 34–35.
7. *Pearson K.* Contributions to the Mathematical Theory of Evolution. Skew Variations in Homogeneous Material // Philosophical Transactions of the Royal Society of London. Ser. A. 1895. V. 186. P. 343–414.
8. *Тихонов В.И.* Статистическая радиотехника. М.: Радио и связь, 1982. 624 с.
9. *Голик Ф.В.* Аппроксимация кривыми Пирсона плотности распределения суммы независимых одинаково распределенных случайных величин // Кибернетика и программирование. 2017. № 2. С. 17–41.
10. *Гончаров В.А.* Методы оптимизации. М., 2008. 188 с.
11. *Lagarias J.C., Reeds J.A., Wright M.H., Wright P.E.* Convergence Properties of the Nelder-Mead Simplex Method in Low Dimensions // SIAM Journal of Optimization. 1998. V. 9. № 1. P. 112–147.
12. *Гмурман В.Е.* Теория вероятностей и математическая статистика. М.: Высш. шк., 2003. 479 с.
13. Вероятность и математическая статистика: Энциклопедия / Под ред. Ю.В. Прохорова. М.: Большая Российская энциклопедия, 2003. 912 с.
14. *Вадзинский Р.Н.* Справочник по вероятностным распределениям. СПб.: Наука, 2001. 295 с.

15. *Гонсалес Р., Вудс Р.* Цифровая обработка изображений. М.: Техносфера, 2012. 1104 с.

16. *Pfister R., Schwarz K.A., Janczyk M., Dale R., Freeman J.* Good things peak in pairs: a note on the bimodality coefficient // Frontiers in psychology. 2013. V. 4.

17. *Савинов А.Н., Иванов В.И.* Анализ решения проблем возникновения ошибок первого и второго рода в системах распознавания клавиатурного почерка // Вестник ВУиТ. 2011. № 18.

18. Статистические методы. Вероятность и основы статистики. Термины и определения. ГОСТ Р 50779.10-2000. М.: Госстандарт России, 2001. 42 с.

19. *Лисничук А.А., Батищев А.В.* Двухкритериальный синтез OFDM-сигналов для повышения энергетической эффективности и помехоустойчивости // Вестник РГРТУ. 2021. № 76. С. 3–16.

20. *Гонсалес Р., Вудс Р., Эддинс Р.* Цифровая обработка изображений в среде MATLAB. М.: Техносфера, 2006. 616 с.

*ПРОГРАММИРОВАНИЕ, 2023, № 3, с. 26–36*

## КОМПЬЮТЕРНАЯ ГРАФИКА И ВИЗУАЛИЗАЦИЯ

*УДК 004.92*

# МЕТОД 3D-РЕКОНСТРУКЦИИ И ОЦИФРОВКИ СЦЕНЫ ДЛЯ СИСТЕМ СМЕШАННОЙ РЕАЛЬНОСТИ

**© 2023 г. М. И. Сорокин[a,*] (ORCID: 0000-0001-9093-1690), Д. Д. Жданов[a,**] (ORCID: 0000-0001-7346-8155), А. Д. Жданов[a,***] (ORCID: 0000-0002-2569-1982)**

[a]*Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, Россия, 197101 Санкт-Петербург, Кронверкский пр., 49*

**e-mail: vergotten@gmail.com*

***e-mail: ddzhdanov@mail.ru*

****e-mail: andrew.gtx@gmail.com*

Поступила в редакцию 09.01.2023 г.
После доработки 16.01.2023 г.
Принята к публикации 20.01.2023 г.

Системы смешанной реальности являются перспективным направлением, открывающим большие возможности для взаимодействия с виртуальными объектами в реальном мире. Как любое перспективное направление смешанная реальность имеет ряд нерешенных проблем. Одна из таких проблем – это формирование естественных условий освещения для виртуальных объектов, а также обеспечение корректного светового взаимодействия виртуальных объектов с реальным миром. Так как виртуальные и реальные объекты находятся в разных пространствах, то обеспечить их корректное взаимодействие является сложной задачей. Для создания цифровых копий объектов реального мира используются инструменты машинного обучения и технологии нейронных сетей. Данные методы успешно применяются в задачах компьютерного зрения для решения проблем ориентации в пространстве и реконструкции окружающей среды. В качестве решения предлагается переместить все объекты в одно информационное пространство – виртуальное. Такое решение позволит снять большую часть проблем, связанных с дискомфортом зрительного восприятия, вызванного неестественным световым взаимодействием объектов реального и виртуального миров. Поэтому основная идея метода заключается в определении объектов физического мира по облакам точек и их замена виртуальными CAD-моделями. То есть семантический анализ сцены и задача классификации объектов с последующим преобразованием в полигональные модели. В данной работе предлагается использование конкурентоспособных нейросетевых архитектур, позволяющих получить современные “state of the art” результаты. Эксперименты проводились на наборах данных “Semantic3D”, “ScanNet” и “S3DIS”, которые на данный момент являются крупнейшими датасетами с наборами облаков точек интерьерных сцен. В качестве метода решения задач семантической сегментации и классификации 3D-облаков точек было решено использовать архитектуру PointNeXt, основанную на PointNet, и применить в процессе обучения современные методы аугментации данных. Для восстановления геометрии был рассмотрен метод дифференциального рендеринга Soft Rasterizer и нейронная сеть “Total3Understanding”.

**DOI:** 10.31857/S0132347423030056, **EDN:** DEMESA

## 1. ВВЕДЕНИЕ

За последние несколько десятилетий благодаря новым методам формирования визуальной и звуковой информации взаимодействие между человеком и компьютером претерпело множество изменений. Вслед за мэйнфреймами, ПК и мобильными телефонами следующим революционным компонентом вычислительной техники становится смешанная реальность. Объединение физической и виртуальной реальностей позволяет обеспечить более естественное и интуитивно понятное взаимодействие между людьми, машинами и окружающей их средой.

Системы смешанной реальности становятся популярны как среди потребителей, так и среди компаний. Границы областей применения систем смешанной реальности определяет только наше воображение. Медицина, образование, воссоздание исторических памятников и достопримечательностей, коммуникация и игровая индустрия — это лишь малая часть потенциального использования данных систем и приложений [1–5].

Системы смешанной реальности с технической точки зрения хоть и похожи на системы виртуальной и дополненной реальности, однако имеют существенные отличия. Виртуальный мир

должен восприниматься как единое целое с реальным миром, выглядеть физически корректно и естественно. Отсюда возникает ряд задач, включающих физически-корректное отображение виртуального объекта и его встраивание в реальную среду. В зависимости от типа систем смешанной реальности взаимодействие объектов виртуального и реального миров происходят поразному. В видео-прозрачных системах реальный мир передается на дисплей парой камер высокого разрешения, в то время как в оптически-прозрачных системах реальный мир передается через прозрачную оптику и виртуальный объект является прозрачным, а его видимость достигается за счет многократного увеличения его яркости по сравнению с реальной средой. Так как объекты реального и виртуального миров объединяются в так называемом “спектре смешанной реальности”, отсюда возникает ряд проблем, связанных с неправильным взаимным освещением объектов реального и виртуального миров и разного рода окклюзиями. При невыполнении условий физически-корректного отображения виртуального объекта у пользователя системы смешанной реальности возникает дискомфорт зрительного восприятия. Данный дискомфорт негативно сказывается на состоянии пользователя, что может приводить к головокружениям, потере координации, эффекту “укачивания” и даже травмам [6, 7]. Поэтому устранение дискомфорта зрительного восприятия является первостепенной задачей. Программная реализация сложного физического взаимодействия между реальными и виртуальными объектами является сложной и ресурсоемкой технической задачей, так как объекты находятся в разных пространствах и используют разные средства анализа и отображения. Одним из возможных решений является перевод всех объектов в единое пространство, а поскольку перевод виртуальных объектов в реальный мир невозможен, то предлагается оцифровка реального мира и перевод его в виртуальное пространство.

Основой для построения виртуального аналога реального мира служит :глубина” или RGB-D-изображение. На данный момент по точности получения карт глубин лидируют сканирующие лидары, однако их пространственное разрешение и частота обновления кадров (FPS) довольно низкие по сравнению с рядом других методов, например, по сравнению с ToF (Time-of-flight) камерами, которые используются в смартфонах. Стереокамеры на данный момент являются наиболее дешевым аппаратным способом получения карты глубины сцены. Стереокамеры хорошо работают при солнечном освещении, однако менее эффективны при слабом освещении, что может стать проблемой при работе с интерьерными сценами. Основная сложность получения глубины из стереоизображения заключается в постобработке, а именно в построении карт диспаритета, по которым и вычисляются карты глубин сцены. Данная процедура весьма ресурсоемка и не все мобильные устройства могут позволить себе эту процедуру. Также из недостатков следует отметить плохую работу стереокамер на неконтрастных объектах и необходимость иметь большую базу (расстояние между камерами) для определения глубин удаленных объектов. Еще одно интересное и многообещающее решение это ToF-камеры. Данные камеры измеряют временную задержку отраженного света. Изображения карты глубин получаются хоть и низкого разрешения, однако с высоким FPS. Одним из плюсов ToF-камер является высокая скорость работы, что для многих задач является определяющим условием. ToF-камеры отлично работают как при слабом освещении, так и при полном отсутствии света. Точность получения глубины сцены проигрывает лидарам, однако выигрывает у стереокамер. По сравнению со стереокамерами сложность программного обеспечения у ToF-камер низкая, что делает их отличным кандидатом для использования в приложениях реального времени. Несмотря на то, что большинство рассмотренных методов имеет определенные минусы, современные нейросетевые технологии позволяют нивелировать часть этих недостатков.

## 2. АНАЛИЗ ТЕКУЩЕГО СОСТОЯНИЯ И ПЕРСПЕКТИВ РАЗВИТИЯ КАМЕР ГЛУБИН

В сфере смешанной реальности в настоящее время растет конкуренция со стороны значимых ИТ-компаний. В качестве примеров можно привести PlayStation VR от Sony, Daydream, Cardboard, Tango и ARCore от Google, Oculus Rift от Facebook, Vive от Valve/HTC HTC и HoloLens от Microsoft, а также Apple (ARKit), Acer, Asus, Dell, HP и Lenovo [8]. Цель MR (Mixed Reality) технологий объединить реальный и виртуальный миры таким образом, чтобы “континуум смешанной реальности” казался единым целым. Техническая сложность заключается в точной регистрации виртуальных и реальных объектов. И MR, и VR (Virtual Reality) требуют отображения компьютерных 3D-изображений в реальном времени, для чего разрабатывают и используют новые технологические решения. В настоящее время MR пытается улучшить методы отслеживания объектов, в первую очередь для мобильных устройств. Технология получает все более широкое распространение в результате постоянного улучшения качества приложений смешанной реальности и соответствующего повышения производительности аппаратного обеспечения. Это создает новые области применения для бизнеса, особенно для промышленных приложений, таких как: удален-

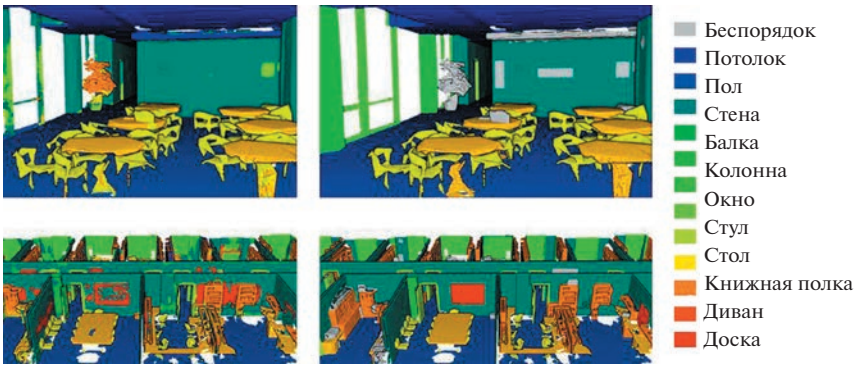


**Рис. 1.** Пример набора данных S3DIS [11].

**Рис. 2.** Пример набора данных ScanNe [12].

ная помощь, интерактивное обучение, учебные сценарии и виртуальное прототипирование. Однако существующие решения все еще находятся на стадии экспериментов и часто имеют довольно низкую привлекательность для конечного пользователя и слишком низкую надежность для повседневной эксплуатации, особенно для профессионального использования. Универсальные приложения MR, свободные от дискомфорта зрительного восприятия, отсутствуют на рынке. Кроме того, не хватает и корпоративной поддержки для превращения знаний в инновации. Согласно исследованию Марка Пэллота [9], последние публикации и исследовательские инициативы ясно демонстрируют тенденцию среди предприятий и общественных организаций к открытости в отношении совместного творчества. Для оценки новых идей и концепций идеально подходят иммерсивные и совместные среды (ICE), основанные на дополненной или смешанной реальности. Однако, необходимо разработать методологии и инструменты более сложные, чем те, которые доступны сейчас.

Чтобы быстро адаптироваться к расположению и ориентации дисплея в среде MR, данные должны генерироваться в режиме реального времени с частотой обновления примерно 60 кадров в секунду. Сложность демонстрируемого виртуального контента оказывает значительное влияние на частоту обновления. В связи с тем, что многие системы смешанной реальности являются мобильными и имеют малую вычислительную мощность, данные САПР, результаты сканирования и результаты моделирования часто оказываются слишком сложны для отображения в реальном времени.

Эффективное взаимодействие с данными, полученными от окружающей среды, представляет

**Таблица 1.** Основные метрики оценки результатов при работе с 3D и облаками точек

| Метрика | Формула |
|---|---|
| Accuracy | $Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$ |
| mACC | $mACC = \frac{1}{C}\sum_{c=1}^{C} Accuracy_c$ |
| Precision | $Precision = \frac{\mathrm{TP}}{2\mathrm{TP} + \mathrm{FN}}$ |
| Recall | $Recall = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$ |
| F1-Score | $F_1 = 2 \times \frac{\mathrm{Precision} \times \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}}$ |
| IoU | $IoU_i = \frac{I_{i,i}}{\sum_{c=1}^{C}(I_{i,c} + I_{c,i}) - I_{i,i}}$ |
| mIoU | $mIoU = \frac{1}{C}\sum_{c=1}^{C} IoU_i$ |
| EPE | $EPE = \|s\hat{}f - sf\|_2$ |

собой дополнительную проблему. В отличие от настольных компьютерных систем, для которых клавиатура и мышь являются стандартными устройствами ввода, и мобильных устройств, взаимодействующих при помощи тактильных прикосновений, устройства ввода в технологиях MR отнюдь не стандартизированы. Распознавание речи и управление жестами на данный момент являются основными методами ввода в системах смешанной реальности. Жесты часто используются, например, для выбора пункта в меню. Однако, управление жестами не лучший вариант, если требуется сложный ввод текста или точное размещение виртуальных объектов в пространстве. Отсутствие обратной связи в виде тактильных ощущений и усталость пользователя – также два фактора, которые необходимо учитывать при оценке дискомфорта [10]. Все это представляет серьезный риск для пригодности и развития технологии.

В данной работе предлагается метод, основанный на использовании нейронных сетей для классификации и сегментации объектов сцены, представленных в виде облака точек и методов дифференцируемого рендеринга для восстановления геометрии объектов, чьи аналоги не были найдены в базах данных CAD-объектов.

**Таблица 2.** Сравнение моделей в задаче семантической сегментации на наборе данных S3DIS

| **Модель** | **mIoU** | **m(Acc)** | **o(Acc)** |
|---|---|---|---|
| WindowNorm + + StratifiedTransformer | 77.6 | 85.8 | 91.7 |
| PointMetaBase-XXL | 77.0 | – | 91.3 |
| **PointNeXt-XL** | 74.9 | 83.0 | 90.3 |
| DeepViewAgg | 74.7 | 83.8 | 90.1 |
| PointTrans-former+GAM | 74.4 | 83.2 | 90.6 |
| RepSurf-U | 74.3 | 82.6 | 90.8 |
| **PointNeXt-L** | 73.9 | 82.2 | 89.9 |
| PointTransformer | 73.5 | 81.9 | 90.2 |
| CBL | 73.1 | 79.4 | 89.6 |

## 3. НАБОРЫ ДАННЫХ ИНТЕРЬЕРНЫХ СЦЕН И МЕТРИКИ ОЦЕНКИ

Существует множество наборов данных с облаками точек, используемых для различных задач, конкретно в нашем случае необходимо сделать акцент именно на наборы данных интерьерных сцен. Из таких датасетов следует отметить S3DIS и ScanNet.

Набор данных Stanford Large-Scale 3D Indoor Spaces (S3DIS) [11] состоит из 5 крупномасштабных внутренних сцен из трех зданий. Они отличаются по архитектуре, внешнему виду и стилю. Данный набор промаркирован 13 классами, что включает в себя конструктивные элементы (пол, стена и т. д.) и обычную мебель. Пример набора данных S3DIS представлен на рис. 1.

ScanNet [12] это большой набор видеоданных (2.5 миллионов кадров), полученных из более чем 1000 сканирований, аннотированных реконструкцией поверхности, семантической сегментацией и положениями камеры. Пример набора данных ScanNet представлен на рис. 2.

Для оценки качества работы алгоритмов и обученных моделей используются специальные метрики оценки, представленные в табл. 1. Данные метрики широко используются как в различных ML-задачах, так и при работе с облаками точек. От показателей данных метрик зависит то, насколько хорошо алгоритм справляется с поставленной задачей, и то, какие выводы можно получить. Для задач сегментации обычно используются метрики accuracy или mIoU, а в задачах детектирования – mIoU, accuracy, precision и recall. Для сопоставления 3D-моделей в сцене могут быть использованы кривые ROC (производные от precision и recall) [13].

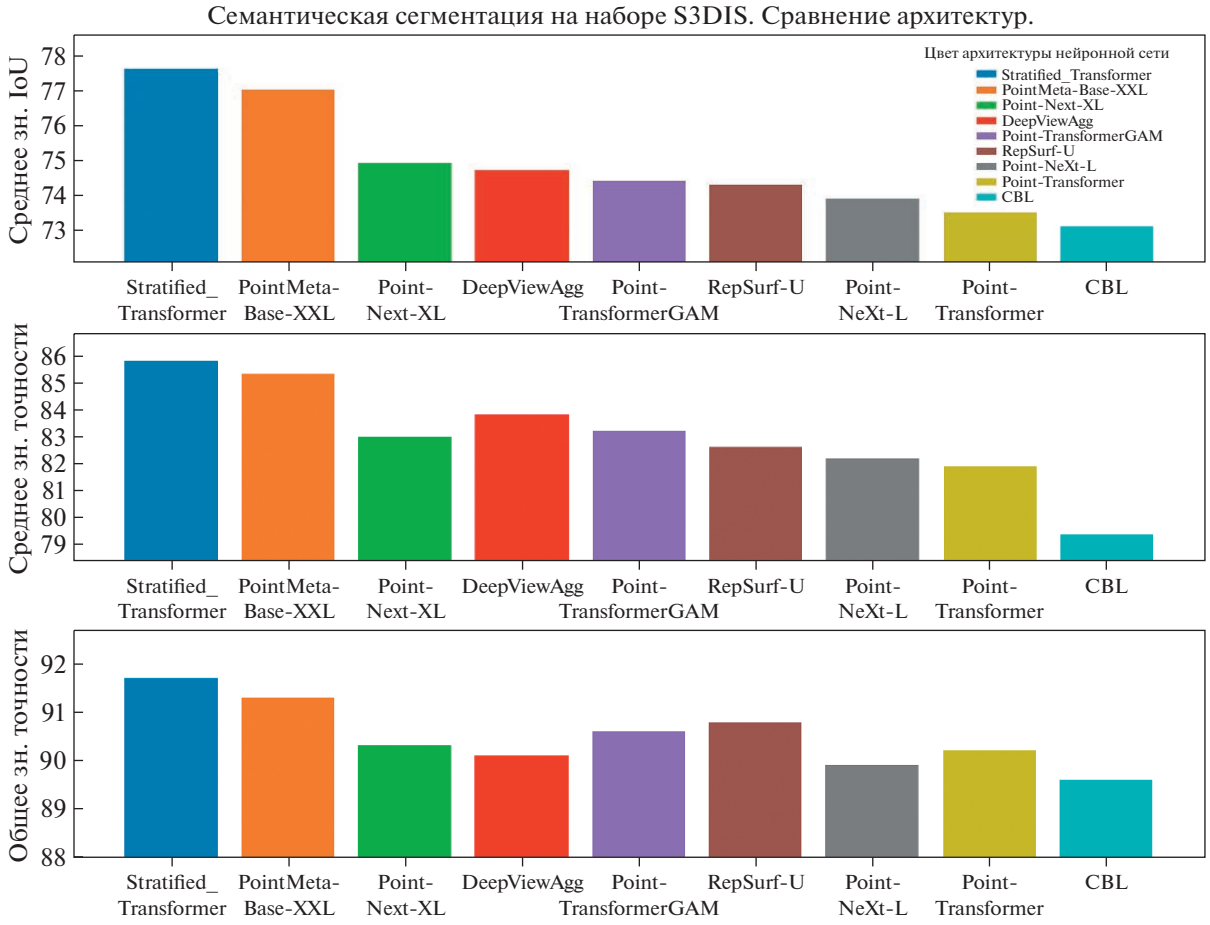


**Рис. 3.** Сравнение точности классификации нейронных сетей в задачах сегментации.

Accuracy (точность) – это показатель, который обычно описывает, как модель работает на всех классах. mACC (средняя точность) может быть использована в тех случаях, когда классы не сбалансированы. Precision можно рассматривать как меру качества, а recall – как меру количества. Показатель F1 можно интерпретировать как среднее гармоническое между precision и recall, где показатель F1 достигает своего наилучшего значения при 1, а наихудшего значения при 0. IoU пересечение определяется между предсказанной маской и ground truth. mIoU – среднее IoU на всех классах. End point error (EPE) используется в задачах вычисления оптического потока.

**Таблица 3.** Сравнение моделей в задаче классификации 3D-облака точек на наборе ScanObjectNN

| Модель | m(Acc) | o(Acc) |
|---|---|---|
| **I2P-MAE** | 90.11 | – |
| **ULIP + PointNeXt** | 89.7 | 88.6 |
| **ULIP + PointMLP** | 89.4 | 88.5 |
| **P2P** | 89.3 | – |
| **PointNeXt+Local** | 88.6 | 87.4 |
| **PointNeXt+GAM** | 88.4 | – |
| **PointNeXt+HyCoRe** | 88.3 | 87.0 |
| **ACT** | 88.21 | – |
| **PointNeXt** | 88.2 | 86.8 |

## 4. АРХИТЕКТУРА POINTNEXT ДЛЯ РАБОТЫ С 3D-ОБЛАКАМИ ТОЧЕК

На текущий момент существует большое количество архитектур нейронных сетей для работы с трехмерными данными. Нейронные сети могут работать практически с любым типом 3D-информации: как с облаками точек, так и с вокселями и полигональными сетками.

В данной работе рассмотрена архитектура PointNeXt [14], что является следующей версией таких архитектур как PointNet и PointNet++. Как утверждают авторы, данная архитектура еще не исчерпала себя и, благодаря современным мето-

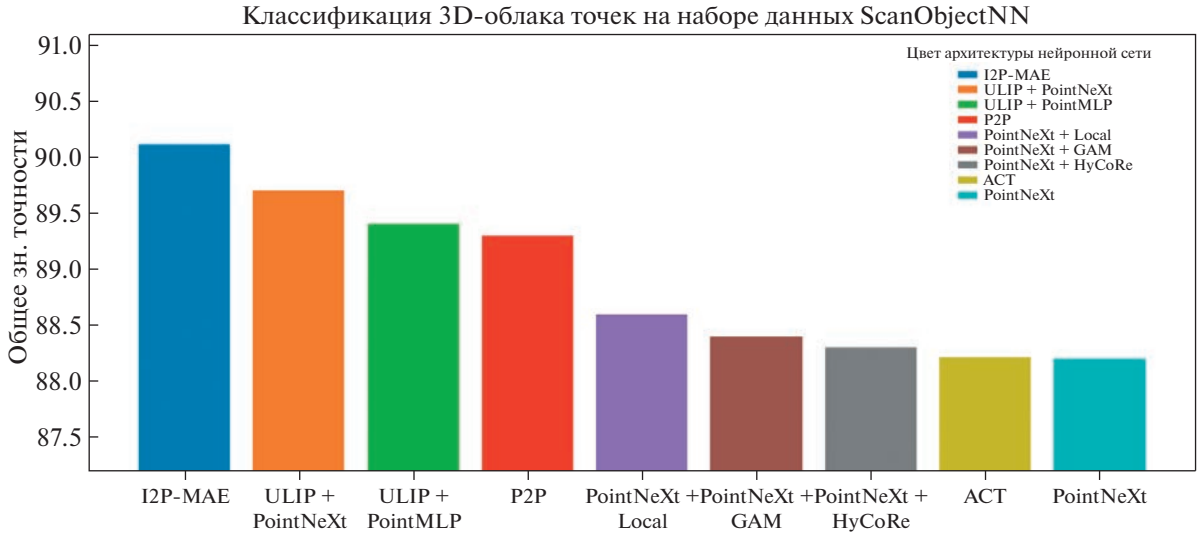


**Рис. 4.** Сравнение точности классификации нейронных сетей в задачах классификации 3D-облака точек.

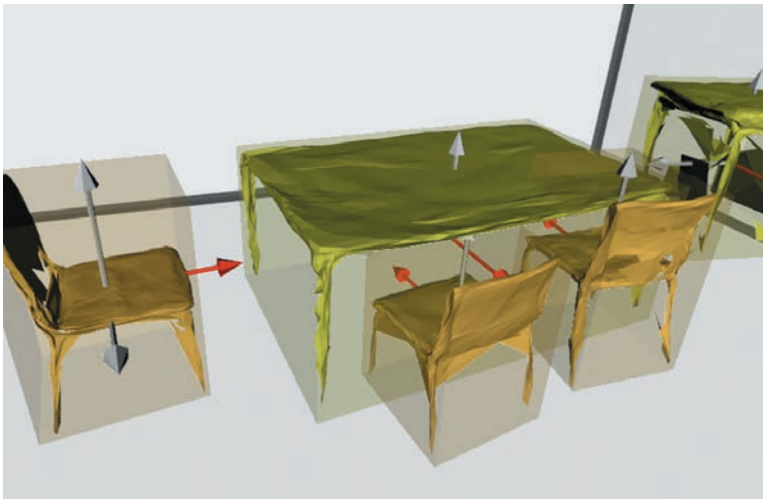

**Рис. 5.** Выделение объектов сцены в пространстве с использованием Total3DUnderstanding.

дам обучения и аугментации данных, позволяет достичь современных SOTA (State Of The Art) результатов. В задачах семантической сегментации PointNeXt хоть и не сильно, но уступает таким архитектурам, как StratifiedTransformer и PointMeta-Base-XXL (рис. 3, табл. 2). Однако, PointNeXt также показывает отличные результаты в задачах 3D-классификации по облаку точек (рис. 4, табл. 3).

Для эффективного масштабирования модели авторы добавили в PointNet++ разделяемые многослойные перцептроны (MLPs) [15], инвертированный “боттлнек” (inverted bottleneck design) [16] и остаточные связи (residual connections) [17]. Только при пересмотре методов обучения общая точность OA (Overall Accuracy) на наборе ScanObjectNN увеличивается на 8.2% (с 77.9 до 86.1%), создавая новую SOTA без внесения каких-либо изменений в архитектуру. Показатель mIoU (mean Intersection Over Union), оцененный на всех регионах с помощью 6-кратной перекрестной валидации на наборе S3DIS, увеличивается на 13.6% (с 54.5 до

**Таблица 4.** Общая точность, средняя точность и mIoU на тестовой сцене

| Общ. точность (Overall Accuracy) | Средн. точность (Mean Accuracy) | Средн. IoU (Mean IoU) |
|---|---|---|
| 90.94 | 77.13 | 76.55 |

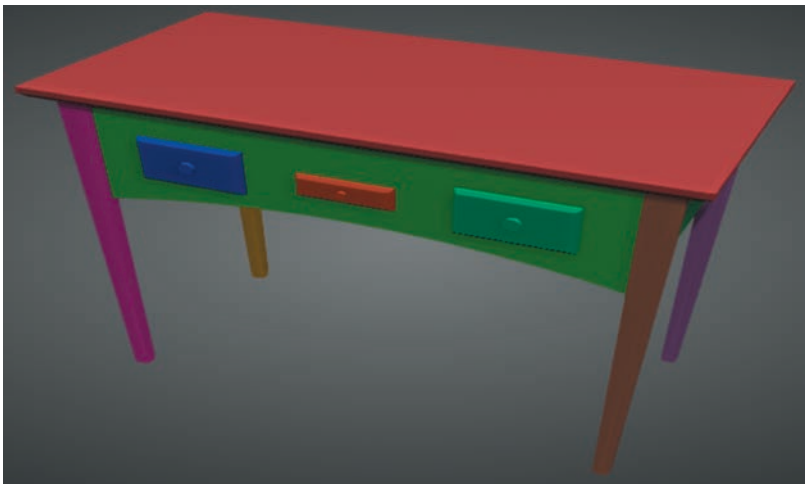

**Рис. 6.** Пример сегментации частей объекта.

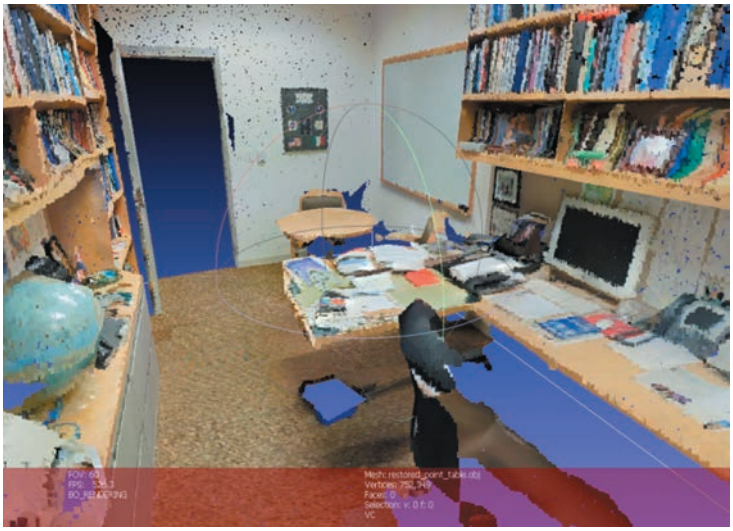

**Рис. 7.** Скан интерьерного помещения.

68.1%), опережая несколько современных разработок, появившихся после PointNet++, таких как PointCNN [18] и DeepGCN [19]. Производительность нейронной сети также зависит от методов оптимизации, таких как функции потерь, оптимизаторов, планировщиков скорости обучения и гиперпараметров. Благодаря развитию теории машинного обучения современные нейронные сети можно обучать с помощью более совершенных оптимизаторов (например, AdamW [20] по сравнению с Adam [21]) и более продвинутых функций потерь (CrossEntropy со сглаживанием меток [22]).

Производительность в задачах классификации и сегментации повышается благодаря увеличению объема данных (аугментации). В ScanObjectNN повторная выборка точек улучшает производительность на 2.5% OA. Результат сегментации улучшается на 1.1% mIoU, когда в качестве входных данных используется полная сцена в отличие от данных, дискретизированных по блокам или сферам.

## 5. МЕТОД 3D-РЕКОНСТРУКЦИИ И ОЦИФРОВКИ СЦЕНЫ

Для решения поставленной задачи восстановления геометрии реального окружения, наблюдаемого стереокамерами системы MR, представлен алгоритм, который основывается на нейронных сетях “PointNeXt” и “Total3DUnderstanding” [23].

1. PointNeXt используется для классификации облака точек, в то время как Total3DUnderstanding – для восстановления границ и положения объектов в пространстве сцены (рис. 5). Помимо

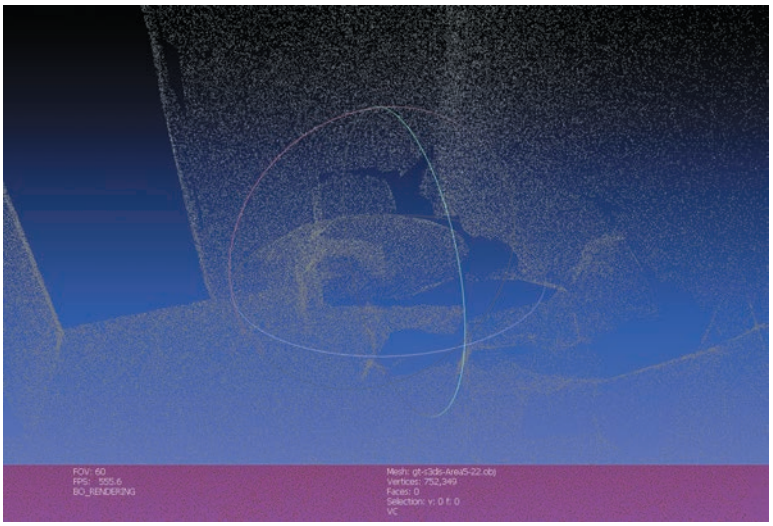

**Рис. 8.** Облако точек интерьерного помещения.

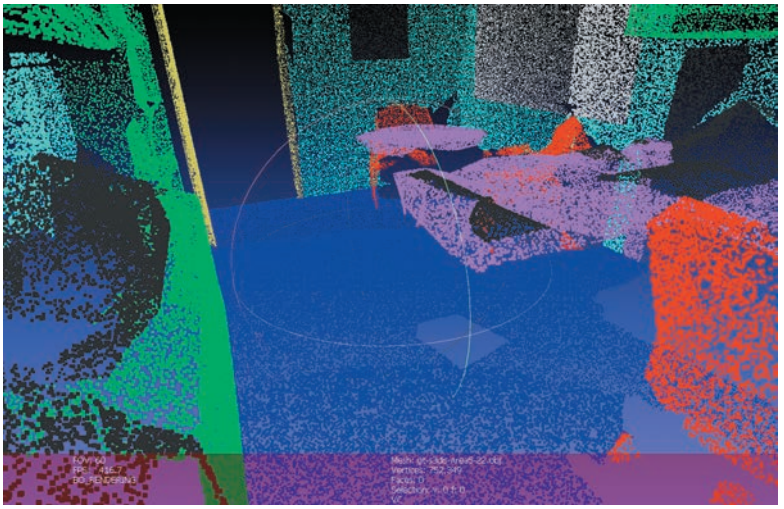

**Рис. 9.** Сегментированное PointNeXt облако точек.

LEN компонента (Layout Estimation Network) Total3DUnderstanding включает в себя такие модули, как ODN (Object Detection Network) и MGN (Mesh Generation Network).

2. После классификации облаков точек по классам объектов определяются их составные части (столешница, ножки, ящики стола), т.е. выделяются их сегменты (рис. 6).

3. Необходимо составить достаточно большую базу данных с CAD-моделями (либо использовать готовую, например ShapeNetCore [24]). Данная база данных должна иметь подробную аннотацию и описание составных частей объекта для максимально быстрого поиска аналога и замены им реального объекта сцены.

4. Если по облаку точек не удалось составить полную картину объекта и его частей, используются средства дифференциального рендеринга, где в качестве целевого объекта выбирается объект, максимально похожий по описанию классифицированных частей на реальный объект сцены.

5. Из базы данных извлекается модель и помещается на место реального объекта в виртуальном пространстве сцены.

## 6. РЕЗУЛЬТАТЫ

В качестве тестовой сцены для эксперимента был взят скан интерьерного помещения из набора данных 3SDIS, который не был использован при обучении (рис. 7).

**Таблица 5.** IoU по всем предсказанным классам

| N класса | Название класса | IoU |
|---|---|---|
| 1 | Беспорядок (Clutter) | 93.68 |
| 2 | Потолок (Ceiling) | 98.58 |
| 3 | Пол (Floor) | 85.26 |
| 4 | Стена (Wall) | 71.08 |
| 5 | Балка (Beam) | 42.27 |
| 6 | Колонна (Column) | 60.59 |
| 7 | Окно (Window) | 70.93 |
| 8 | Дверь (Door) | 84.44 |
| 9 | Кресло (Chair) | 92.41 |
| 10 | Стол (Table) | 80.35 |
| 11 | Книжная полка (Bookcase) | 78.08 |
| 12 | Диван (Sofa) | 76.59 |
| 13 | Доска (Board) | 60.98 |

Соответствующее ему облако точек представлено на рис. 8.

Результат работы нейронной сети PointNeXt представлен на рис. 9.

Кластеры точек, предсказанные к разным классам, для наглядности выделены разным цветом. В табл. 4 представлены значения общей и средней точности тестовой сцены, а в табл. 5 значения IoU для каждого класса объектов и их наименование.

После завершения классификации выполняется построение полигональных сеток по облакам точек. Для этого используется алгоритм реконструкции поверхности Пуассона. Для случаев, когда восстановленной полигональной модели оказывается достаточно для определения частей объекта, осуществляется замена облака на найденную в базе данных CAD-модель (рис. 10). Данная модель масштабируется и размещается в области пространства, занимаемой облаком точек. Для случаев, когда сначала необходимо восстановить отсутствующие части объекта, применяется дифференцируемый рендеринг SoftRasterizer.

Целевая модель – это конечная модель, которую мы хотим получить при дифференцируемом рендеринге. Другими словами, это эталонная модель, к которой мы стремимся деформировать изначальную, используя функции потерь. Так как в исходном облаке точек есть отсутствующие элементы (рис. 12, исходное облако точек), требуется по текстовому описанию найти наиболее подходящую модель или скомпоновать ее из частей объектов. В данном примере в качестве целевой модели была выбрана модель из набора ShapeNetCore, которая максимально похожа по описанию присутствующих частей (круглая спинка стула, ручки).

Полученная после дифференцируемого рендеринга модель слишком грубая и содержит большое количество артефактов (рис. 12, полигональная сетка после дифференцируемого рендеринга). Для финальной сцены такую модель использовать нельзя, однако можно найти наиболее подходящий аналог из базы данных. Поэтому деформированная модель заменяется CAD-моделью из базы данных (рис. 12, CAD-модель). В процессе выполнения дифференцируемого рендеринга используются следующие метрики и функции потерь: chamfer loss, edge loss, normal loss, laplacian loss (рис. 11), где chamfer loss – расстояние между прогнозируемой (деформированной) и целевой моделью, edge loss минимизирует длину ребер в прогнозируемой модели, normal loss – согласованность нормалей соседних граней, а laplacian loss является регуляризатором.

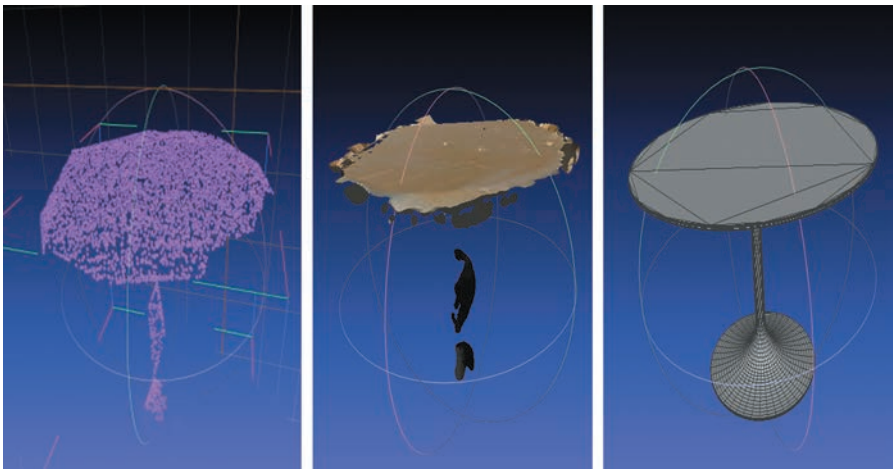

**Рис. 10.** Облако точек, полигональная сетка и CAD аналог (слева направо).

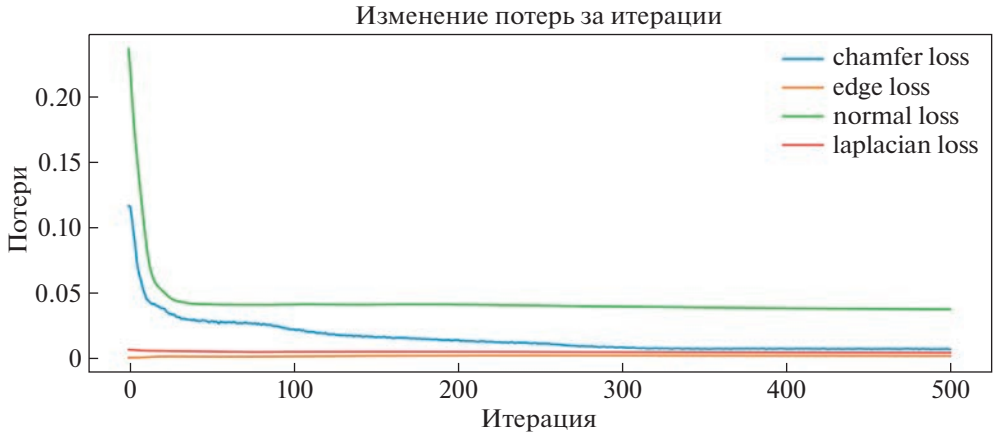


**Рис. 11.** Изменение функций потерь при дифференцируемом рендеринге на тестовой модели.

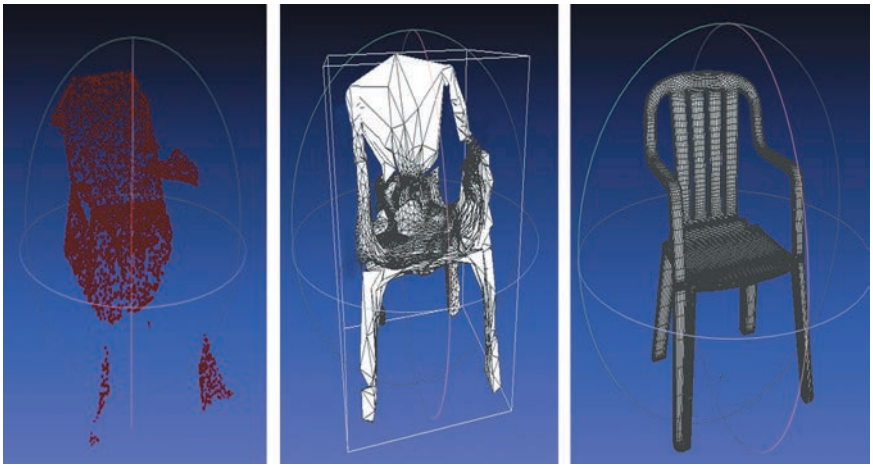

**Рис. 12.** Исходное облако точек, полигональная сетка после дифференцируемого рендеринга и CAD модель (слева-направо).

## 7. ЗАКЛЮЧЕНИЕ

Системы смешанной реальности представляют собой перспективное направление, обеспечивающее интерактивное взаимодействие человека с объектами реального и виртуального миров одновременно. Однако, проблема корректного взаимодействия виртуальных объектов с физическим миром полностью не решена, и это замедляет процесс внедрения данных систем в повседневную жизнь. В данной работе был представлен метод реконструкции реальной сцены для систем смешанной реальности с использованием облаков точек и архитектур нейронных сетей PointNeXt и Total3DUnderstanding. Метод основан на замене реального объекта его виртуальным аналогом, что позволяет устранить ряд источников дискомфорта зрительного восприятия, связанных со световым взаимодействием объектов реального и виртуального миров. Нейронная сеть PointNeXt показывает достаточно хороший результат в задачах классификации облака точек, достигая SOTA результатов, в то время как Total3DUnderstanding решает важные задачи по определению границ и ориентации объектов в пространстве сцены. В данной работе использовался набор данных 3SDIS, который содержит 13 классов. В дальнейшем планируется использование набора данных ScanNet200, который уже содержит 200 классов различных объектов интерьерного помещения. Также планируется создать большую базу данных интерьерных объектов с подробным описанием, чтобы обеспечить эффективный поиск CAD-модели. Для поиска модели по заданным текстовым параметрам и аннотациям перспективно смотрится реализация нейронной сети по типу трансформера.

## ФИНАНСИРОВАНИЕ

Работа была выполнена при финансовой поддержке Российского научного фонда, грант № 22-11-00145.

## СПИСОК ЛИТЕРАТУРЫ

1. *Dhaval S.* Critical review of mixed reality integration with medical devices for patientcare // International Journal for Innovative Research in Multidisciplinary Field. 2022. V. 8. Issue 1. https://doi.org/10.2015/IJIRMF/202201017
2. *Maas M.J., Hughes J.M.* Virtual, augmented and mixed reality in K-12 education: a review of the literature // Technology, Pedagogy and Education. 2020. V. 29. Issue 2. https://doi.org/10.1080/1475939X.2020.1737210
3. *Evangelidis K., Sylaiou S., Papadopoulos T.* Mergin'mode: Mixed reality and geoinformatics for monument demonstration // Applied Sciences. 2020. V. 10. № 11. P. 3826.
4. *Piumsomboon T., Lee G.A., Hart J.D., Ens B., Lindeman R.W., Thomas B.H., Billinghurst M.* Mini-me: An adaptive avatar for mixed reality remote collaboration / In Proceedings of the 2018 CHI conference on human factors in computing systems. 2018. P. 1–13.
5. *Miedema N.A., Vermeer J., Lukosch S., Bidarra R.* Superhuman sports in mixed reality: The multi-player game League of Lasers / In 2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR). IEEE, 2019. P. 1819–1825.
6. *Guna J., Gersak G., Humar I.* Virtual Reality Sickness and Challenges Behind Different Technology and Content Settings // Mobile Networks and Applications. 2020. V. 25. P. 1436–1445. https://doi.org/10.1007/s11036-019-01373-w
7. *Saredakis D., Szpak A., Birckhead B., Keage H.A., Rizzo A., Loetscher T.* Factors associated with virtual reality sickness in head-mounted displays: a systematic review and meta-analysis // Frontiers in human neuroscience. 2020. V. 14. P. 96.
8. *Moser T., Hohlagschwandtner M., Kormann-Hainzl G., Pölzlbauer S., Wolfartsberger J.* Mixed reality applications in industry: challenges and research areas / In International Conference on Software Quality. Cham.: Springer, 2019. P. 95–105.
9. *Pallot M., Fleury S., Poussard B., Richir S.* What are the Challenges and Enabling Technologies to Implement the Do-It-Together Approach Enhanced by Social Media, its Benefits and Drawbacks? // Journal of Innovation Economics Management. 2022. I132-XLII.
10. *Guo J., Weng D., Zhang Z., Liu Y., Duh H.B., Wang Y.* Subjective and objective evaluation of visual fatigue caused by continuous and discontinuous use of HMDs // Journal of the Society for Information Display. 2019. V. 27, № 2. P. 108–119.
11. *Armeni I., Sener O., Zamir A.R., Jiang H., Brilakis I., Fischer M., Savarese S.* 3D semantic parsing of large-scale indoor spaces / In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016. P. 1534–1543.
12. *Dai A., Chang A.X., Savva M., Halber M., Funkhouser T., Nießner M.* ScanNet: Richly-annotated 3D reconstructions of indoor scenes / In Proc. Computer Vision and Pattern Recognition (CVPR), IEEE, 2017.
13. *Haoming L., Humphrey S.* Deep Learning for 3D Point Cloud Understanding: A Survey // Computer Vision and Pattern Recognition. 2020. https://doi.org/10.48550/arXiv.2009.08920
14. *Qian G., Li Y., Peng H., Mai J., Hammoud H.A., Elhoseiny M., Ghanem B.* PointNeXt: Revisiting PointNet++ with Improved Training and Scaling Strategies. arXiv preprint arXiv:2206.04670. 2022.
15. *Qian G., Hammoud H., Li G., Thabet A., Ghanem B.* Assanet: An anisotropicseparable set abstraction for efficient point cloud representation learning // Advances in Neural Information Processing Systems (NeurIPS). 2021. P. 34.
16. *Sandler M., Howard A., Zhu M., Zhmoginov A., Chen L.C.* Mobilenetv2: Inverted residuals and linear bottlenecks / In Proceedings of the IEEE/CVF Conference on Computer Visionand Pattern Recognition (CVPR). 2018. P. 4510–4520.
17. *He K., Zhang X., Ren S., Sun J.* Deep residual learning for image recognition / In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2016. P. 770–778.
18. *Li Y., Bu R., Sun M., Wu W., Di X., Chen B.* Pointcnn: Convolution on X-transformed points // Advances in Neural Information Processing Systems (NeurIPS), 2018.
19. *Li G., Muller M., Thabet A., Ghanem B.* Deepgcns: Can gcns go as deep as cnns? / In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). 2019. P. 9267–9276.
20. *Loshchilov I., Hutter F.* Decoupled weight decay regularization / In International Conference on Learning Representations (ICLR). 2019.
21. *Diederik P. Kingma, Jimmy Ba.* Adam: A method for stochastic optimization / In International Conference on Learning Representations (ICLR). 2015.
22. *Szegedy C., Vanhoucke V., Ioffe S., Shlens J., Wojna Z.* Rethinking theinception architecture for computer vision / In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2016.
23. *Nie Y., Han X., Guo S., Zheng Y., Chang J., Zhang J.J.* Total3dunderstanding: Joint layout, object pose and mesh reconstruction for indoor scenes from a single image / In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020. P. 55–64.
24. *Kulikajevas A., Maskeliūnas R., Damaševičius R., Misra S.* Reconstruction of 3D object shape using hybrid modular neural network architecture trained on 3D models from ShapeNetCore dataset // Sensors. 2019. V. 19. № 7. P. 1553.

*ПРОГРАММИРОВАНИЕ, 2023, № 3, с. 37–48*

## КОМПЬЮТЕРНАЯ ГРАФИКА И ВИЗУАЛИЗАЦИЯ

*УДК 04.092*

# ИСПОЛЬЗОВАНИЕ МНОГОУРОВНЕВЫХ ХЭШ-ТАБЛИЦ ДЛЯ УСКОРЕНИЯ ПРОЦЕССА РЕНДЕРИНГА

**© 2023 г. Д. Д. Жданов[a,*] (ORCID: 0000-0001-7346-8155), А. И. Лысых[a,**] (ORCID: 0000-0002-2437-5275), Р. Р. Халимов[a,***] (ORCID: 0000-0003-1923-4360), И. Е. Кинев[a,****] (ORCID: 0000-0003-2929-1203), А. Д. Жданов[a,*****] (ORCID: 0000-0002-2569-1982)**

*[a]Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики, Россия, 197101 Санкт-Петербург, Кронверкский пр., 49*

**e-mail: ddzhdanov@mail.ru*

***e-mail: lysykhai@ya.ru*

****e-mail: khalimov.ruslan@mail.ru*

*****e-mail: igorkinevitmo@gmail.com*

******e-mail: andrew.gtx@gmail.com*

Поступила в редакцию 10.01.2023 г.
После доработки 18.01.2023 г.
Принята к публикации 22.01.2023 г.

Проведен анализ методов реалистичного рендеринга с точки зрения эффективности расчета яркостей каустического и вторичного освещений. В качестве основного подхода для реализации реалистичного рендеринга был выбран метод двунаправленной прогрессивной трассировки лучей с обратными фотонными картами. Проведен анализ основных причин, снижающих производительность данного метода. Показано, что главным фактором, снижающим его производительность, является медленный доступ к данным фотонных карт. Рассмотрены различные варианты построения ускоряющих пространственных структур, исследованы их преимущества и недостатки. В качестве основных подходов были выбраны регулярная пространственная решетка и бинарное kd-дерево. Пространственная решетка обеспечивает высокую скорость доступа к данным при низкой адаптивности разбиения фотонной карты. Kd-дерево обеспечивает высокую пространственную адаптивность разбиения карты при низкой скорости доступа к данным. Предложено комбинированное решение, объединяющее адаптивность kd-дерева с высокой скоростью доступа к данным пространственной решетки. Для этого регулярная решетка накладывается на kd-дерево, построенное по принципу пространственного деления области фотонов на геометрически равные половины. Для уменьшения объемов памяти было предложено, во-первых, использовать многоуровневые пространственные решетки, накладываемые на выбранные узлы kd-дерева, и, во-вторых, для уменьшения объема памяти ускоряющей структуры хранить пространственные решетки в виде хэш-таблиц. В результате был предложен и реализован новый тип пространственных ускоряющих структур, представляющих собой дерево хэш-таблиц. Для разработанной пространственной структуры были реализованы методы поиска ближайших фотонов, сферы интегрирования которых покрывают точку освещения, и методы поиска пересечения сегмента луча со сферами интегрирования фотонов. Разработанные программные решения были реализованы в программном комплексе Lumicept, и для ряда базовых сцен было произведено сравнение скорости работы предложенного метода с методом, основанным на бинарном дереве, имеющемся в Lumicept. Сравнение показало, что новый метод может повысить общую производительность процедуры рендеринга более чем на 40%.

**DOI:** 10.31857/S013234742303007X, **EDN:** DESNPJ

## 1. ВВЕДЕНИЕ

Реалистичная визуализация является неотъемлемой частью задач построения систем виртуального прототипирования, создания устройств виртуальной, дополненной и смешанной реальностей, обучения нейронных сетей, разработки видеоигр, создания визуальных эффектов в кинематографии и в ряде других областей человеческой деятельности. Несмотря на то, что для решения проблемы реалистичной визуализации существует большое число подходов, они, как правило, решают ограниченный круг задач, связанных, например, с расчетом прямого или вторичного излучения методами однонаправленной или двунаправленной стохастической трассировки лучей [1, 2]. Особо следует выделить методы, основанные на интегрировании видимой яркости

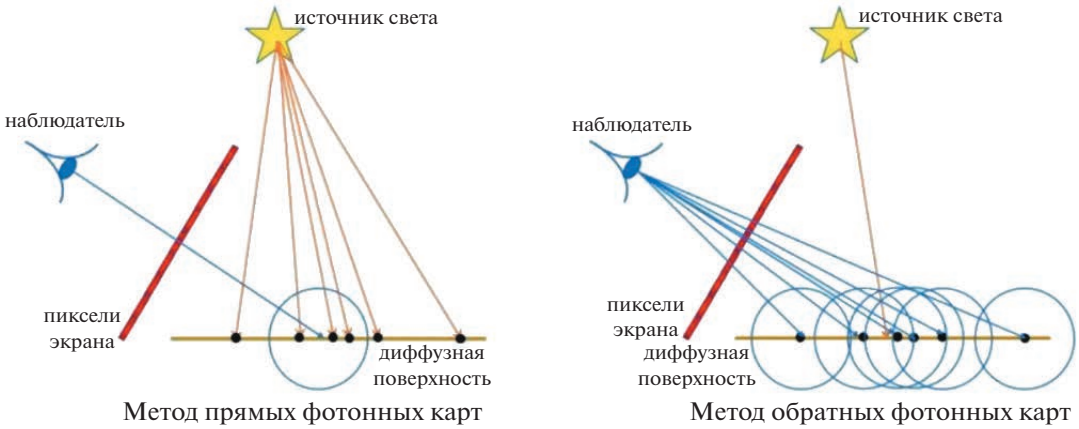


**Рис. 1.** Сбор освещенности с фотонной карты при прямом и обратном методе.

методом Метрополиса, в основе которых лежит интегрирование по схеме Марковских цепей [3–7]. Эти методы обеспечивают высокую эффективность расчета вторичного излучения. Основной недостаток перечисленных выше методов связан с проблемой расчета яркости каустического освещения. В предложенной работе исследовались методы, основанные на фотонных картах, которые позволяют физически корректно рассчитывать сложные сцены с каустическим и вторичным освещением. Однако, несмотря на видимые преимущества методов фотонных карт они имеют ряд серьезных недостатков. Это, во-первых, смещенность метода, вызванная конечным радиусом сфер интегрирования, и, во-вторых, ресурсоемкий процесс поиска пересечения луча со сферами интегрирования.

В данной работе исследуется возможность ускорения процесса рендеринга, использующего метод фотонных карт. Алгоритм вычисления глобальной освещенности с помощью фотонных карт был представлен в работе Х. Йенсена [8] и состоит из трех основных шагов: трассировка фотонов, построение фотонных карт и сбор освещенности с фотонов. Трассировка фотонов осуществляется методом прямой стохастической трассировки лучей. Для построения фотонных карт и эффективного поиска пересечения лучей зрения с фотонами используются ускоряющие структуры пространственного разбиения. Основные требования к программной организации пространственных структур – это : минимально возможный объем дополнительной памяти для хранения и обеспечения доступа к фотонам, быстрое построение пространственной структуры и быстрый доступ к фотонам в ограниченной области пространства сцены.

Сбор освещенности представляет собой трассировку путей от камеры-наблюдателя до геометрии, освещенной фотонами. При попадании $i$-го луча, переносящего световой поток $F_i(\vec{x},\vec{v}')$ в направлении $\vec{v}'$, на поверхность сцены происходит сбор освещенности в области $\vec{x}$, ограниченной сферой интегрирования с радиусом $R$, и последующий расчет видимой яркости $L(\vec{x},\vec{v})$ в этой области в направлении $\vec{v}$:

$$L(\vec{x},\vec{v}) = \sum_{i=all\ photons} \frac{F_i(\vec{x},\vec{v}')}{(\pi R)^2} BRDF(\vec{x},\vec{v},\vec{v}'), \quad (1)$$

где $BRDF(\vec{x},\vec{v},\vec{v}')$ – значение двунаправленной функции рассеивания в области сбора освещенности $\vec{x}$ при освещении в направлении $\vec{v}'$ и наблюдении в направлении $\vec{v}$.

Позднее в работе В. Хавран и др. [9] и в следующих работах [10, 11] был представлен метод обратных прогрессивных фотонных карт, который решает основные недостатки прямого метода, а именно: высокая плотность фотонов вблизи источников освещения, отсутствие критерия, ограничивающего число фотонов, выпускаемых источниками света, и накопление фотонов в скрытых от наблюдателя областях. В этом методе фотоны испускаются из виртуальной камеры, формируя область видимости, а сферы интегрирования сохраняются в фотонной карте для каждого фотона, размер которого рассчитывается относительно расстояния до точки наблюдения. Формула для расчета яркости вторичного и каустического освещения аналогична (1), за исключением того, что сбор световой энергии осуществляется отдельно для каждой точки изображения. Отличия методов расчета яркости глобального освещения для прямых и обратных фотонных карт представлены на рис. 1.

Несмотря на то, что метод стохастических обратных прогрессивных фотонных карт ускоряет процесс расчета глобального освещения, операции обработки запросов к данным структур пространственного разбиения составляют большую часть времени его работы. В предложенной рабо-

те исследуется возможность ускорения процесса рендеринга изображений методом фотонных карт путем минимизации времени доступа к фотонным картам.

## 2. ТИПЫ ОРГАНИЗАЦИИ СТРУКТУР ПРОСТРАНСТВЕННОГО РАЗБИЕНИЯ

Метод фотонных карт требует эффективной структуры пространственного разбиения для хранения и быстрого поиска фотонов, попадающих в сферу интегрирования. Подобные структуры пространственного разбиения также применяются для обеспечения высокой скорости трассировки лучей в сцене.

Структуры пространственного разбиения делят пространство сцены на ячейки, каждая из которых может содержать список фотонов, находящихся внутри нее, элементы геометрии или другие объекты, требующие специальной обработки. Использование таких структур позволяет ускорить поиск объектов, расположенных в локальной области сцены.

Существует множество структур пространственного разбиения, позволяющих добиться увеличения скорости доступа к данным.

### *2.1. Регулярное разбиение пространства*

Регулярная сетка является простейшим методом разбиения пространства [12]. Разрешение такой сетки фиксировано и задается при ее построении. Поскольку все ячейки структуры имеют одинаковый размер, а положение сетки в пространстве определено – по координатам точки можно легко вычислить ячейку, которой она принадлежит. Хранение такой структуры в памяти, как правило, осуществляется с помощью хеш-таблиц, где для доступа к ячейке используются ее координаты в сетке, это обеспечивает быструю скорость доступа к необходимым ячейкам [13]. Кроме того, пустые ячейки можно не хранить в памяти, что сокращает необходимые ресурсы.

Ввиду того, что разрешение сетки фиксируется на этапе ее построения, такая структура обладает недостаточной адаптивностью. При большом размере ячейки, а следовательно, большой концентрации объектов, их поиск будет осуществляться медленно. При увеличении разрешения структура займет больший объем памяти, что в большинстве случаев недопустимо.

### *2.2. Kd-деревья*

Одной из часто используемых структур пространственного разбиения, позволяющих добиться быстрого доступа к данным, локализованным в пространстве сцены, является kd-дерево. Это бинарное дерево, образованное путем последовательного разделения пространства плоскостями.

Деление ячеек плоскостями может осуществляться различными способами: ровно посередине, по количеству объектов в одной и другой части ячейки [14] или более сложным способом, основанным на оценке времени доступа к элементу дерева.

По сравнению с регулярной сеткой бинарное дерево позволяет добиться хорошей адаптивности при неравномерном распределении объектов, но может замедлить доступ к ячейкам при большой глубине дерева, поскольку каждый переход между его узлами требует отдельного обращения к памяти, затрудняя использование кэша процессора. Замедление доступа к листьям дерева усиливается в многопоточном режиме, поскольку несколько потоков могут запрашивать данные, разбросанные в адресном пространстве.

### *2.3. Окто-деревья*

Другой ускоряющей структурой разбиения являются окто-деревья. В данном случае каждая ячейка пространства делится на 8 дочерних путем разделения сразу тремя плоскостями по каждой из осей координат [15]. Как правило, деление ячеек окто-дерева осуществляется через точку по центру ячейки.

По сравнению с kd-деревом такое дерево имеет меньшую глубину, что позволяет увеличить скорость доступа к его ячейкам, однако такая структура требует большей памяти для хранения данных и имеет меньшую адаптивность.

### *2.4. BVH-деревья*

Кроме описанных выше видов деревьев существуют деревья, основанные на иерархии ограничивающих объемов (BVH-деревья) [16]. Данные методы пространственной индексации широко применяются для трассировки лучей, а также для отрисовки сцены, так как обладают хорошей адаптивностью и простотой. Основная идея таких методов заключается в том, чтобы представить сцену в виде подмножества примитивов, которые определяются и сохраняются в узлах на стадии создания дерева. Существует большое количество разновидностей таких деревьев, например: Sphere tree [17], AABB tree [18], OBB tree [19] и т.д. Для разбиения сцены в основном используются две стратегии, которые можно смешивать: разделение по объектам (object split) и пространственное разбиение (spatial split [20]). В первом случае нужно найти и разделить исходное множество примитивов на два подмножества, вычислить ограничивающий объем каждого подмножества, и, если примитив содержится в определенном узле полностью, то он туда и идет. Во втором случае

определяется некая плоскость, по которой происходит деление пространства на две части. В результате примитив добавляется в тот дочерний узел, который он пересекает. Наиболее часто используемой и простой в реализации модификацией BVH-дерева является AABB-дерево, в котором разделение происходит по ограничивающей рамке (AABB). Таким образом, в листьях будут содержаться объекты сцены, которые потом можно будет относительно быстро найти в процессе рендеринга. Достоинствами BVH-деревьев являются: относительная простота реализации, хорошая адаптивность (так как их можно частично перестроить при изменении сцены), быстрое построение структуры и быстрая трассировка. Кроме того, для одинакового набора фотонных карт BVH-дерево будет иметь меньшую глубину, чем kd-дерево, вследствие чего уменьшится количество кэш-промахов при поиске на CPU. Однако существенными недостатками данного метода для поиска фотонов в фотонной карте являются отсутствие критерия для разбиения фотонов по примитивам, возможность пересечения граничных областей и более сложный (чем у kd-дерева) алгоритм поиска фотонов.

### *2.5. Разбалансированные деревья*

В методах прямых фотонных карт для каждого локального расчета яркости необходимо найти k ближайших соседей в kd-дереве. Эта операция является высокозатратной и может занимать больше времени, чем собственно трассировка лучей. Для частичного решения данной проблемы в статье [21] было предложено использовать разбалансированные деревья с эвристикой воксельных объемов (Voxel Volume Heuristic) для оценки плоскостей разбиения и нахождения наилучшего их положения. Авторы утверждают, что данный метод позволяет более эффективно находить k ближайших соседей, при этом без каких-либо приближений. Поскольку бинарный поиск эффективно работает только в сбалансированном дереве, в котором все узлы имеют одинаковые вероятности доступа, то это условие не выполняется при неравномерном распределении фотонов и примитивов в сцене. Для решения проблемы неэффективного доступа в случае неравномерного распределения фотонов авторы модифицируют эвристику площадей поверхности (Surface Area Heuristic) для создания разбалансированного дерева, которое более эффективно работает при неоднородном распределении фотонов в сцене в случае относительно малого среднего радиуса сбора. Таким образом, данный метод эффективно работает в сценах, содержащих сложную каустику, но дает незначительный прирост производительности в сценах, где преобладает диффузное рассеивание. Кроме того, время, занимаемое на построение данного дерева, больше чем на построение kd-дерева.

### *2.6. Комбинированное решение*

Для нашего исследования были выбраны сбалансированные kd-деревья, так как они широко применяются в различных методах рендеринга изображений, обладают высокой адаптивностью и простотой реализации. В работе предлагается использовать комбинированную ускоряющую структуру, которая сочетает в себе преимущества kd-дерева и регулярных структур на основе хэш-таблиц. На определенных уровнях kd-дерева предлагается построить хеш-таблицы, позволяющие непосредственно обращаться к узам этого дерева. Такая структура позволит сохранить адаптивность kd-дерева и увеличит скорость доступа к его данным. Для уменьшения количества памяти, необходимой для хранения структуры, а также для увеличения скорости трассировки предлагается использовать хеш-таблицы на различных уровнях дерева. Анализируя структуру дерева, предлагается выбирать наиболее подходящие для регуляризации уровни, уменьшая среднее время доступа к ячейкам, тем самым повышая эффективность структуры. Принцип построения данной структуры проиллюстрирован на рис. 2.

## 3. ПОСТРОЕНИЕ КОМБИНИРОВАННОЙ УСКОРЯЮЩЕЙ СТРУКТУРЫ

Структура комбинирует два последовательно выполняющихся процесса: построение kd-дерева и создание хеш-таблиц. Для построения kd-дерева необходимо иметь основные данные фотонной карты: координаты и радиусы фотонов. Далее на месте kd-дерева будет построена древовидная структура из хеш-таблиц. Для того, чтобы сделать это оптимальным образом, во время построения хеш-таблиц производится анализ структуры дерева для определения подходящих уровней, на которых будут построены хеш-таблицы.

Первый этап построения структуры начинается с определения ограничивающего параллелепипеда по минимальному и максимальному значению координат накопленных фотонов. Этот параллелепипед будет являться корневым узлом kd-дерева. После создания этого узла запускается повторяющийся цикл, условием окончания которого является проверка на удовлетворение критерия максимального числа фотонов в ячейке. Повторяющийся для каждой ячейки цикл разделения состоит из трех действий: в первую очередь выбирается ось координат, перпендикулярно которой будет происходить разделение ячейки. В данном случае эта плоскость раздела всегда проходит через центр ячейки, поскольку на уровни дерева будет наложена регулярная решетка,

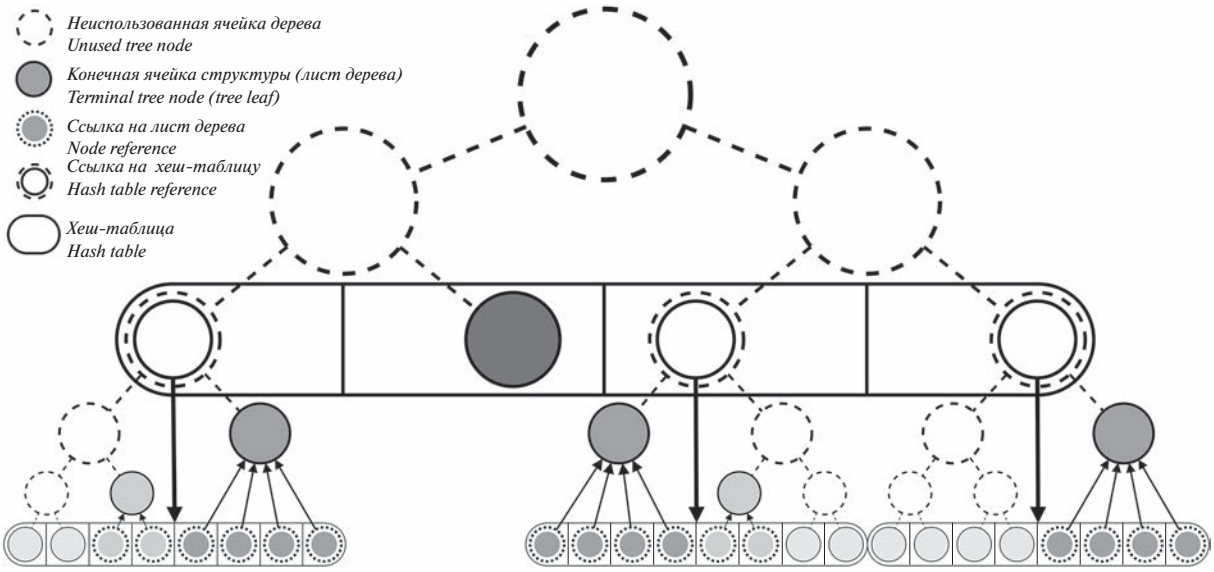


**Рис. 2** Схема ускоряющей структуры, построенной на kd-дереве и хэш-таблице.

для которой границы ячеек строго определены. Далее происходит сортировка фотонов ячейки по областям, образованным секущей плоскостью. После этого создаются две новые ячейки-потомка для текущего узла дерева, в которые добавляются отсортированные фотоны. В случае, если число фотонов в ячейке достигнет требуемого значения, алгоритм построения дерева завершается пересчетом радиусов фотонов в соответствии с плотностью их распределения и формированием ограничивающего параллелепипеда фотонов в каждой ячейке дерева. На рис. 3 представлен алгоритм построения ускоряющей структуры фотонных карт, построенной на kd-дереве и хэш-таблице.

Завершающим этапом построения комбинированной структуры является создание хеш-таблиц на оптимальных уровнях kd-дерева. Критерий оптимальности выбора таких уровней заключается в максимальном заполнении конечными листьями дерева при непревышении некоторой фиксированной глубины. Данный критерий обеспечивает максимальный доступ к фотонам при относительно небольшом размере таблицы. Для определения оптимальных уровней необходимо формировать счетчик числа листьев дерева на каждом его уровне. Данный счетчик может формироваться в процессе построения дерева. В статье [22] было определено оптимальное значение глубины уровня для построения хеш-таблицы при одновременном ее использовании с kd-деревом. Задачей данного исследования является анализ оптимального количества таблиц, а также уровней дерева, на которых они будут размещены.

На рис. 4 представлен алгоритм формирования данных хеш-таблицы для одного узла дерева, не превышающего заданный уровень. Рассматриваются четыре основных случая:

1. узел дерева является пустым листом,

2. узел дерева не является листом,

3. узел дерева непустой лист на заданном уровне,

4. узел дерева непустой лист на уровне выше заданного.

В первом случае ничего не происходит, и хэш-таблица не обновляется. Далее происходит переход на следующий узел дерева, не превышающий заданный.

Во втором случае происходит формирование нового узла хэш-таблицы, обновляются ее параметры, глубина дерева и разрешение таблицы. Далее происходит рекурсивное формирование новой хэш-таблицы для заданного узла kd-дерева, и по окончании этого рекурсивного процесса происходит переход на следующий узел дерева, не превышающий заданный.

В третьем случае формируются индекс решетки, соответствующий данному уровню kd-дерева, и хэш индекс таблицы (остаток от деления индекса на размер хэш-таблицы), которые добавляются в ячейку таблицы. Далее происходит переход на следующий узел дерева, не превышающий заданный.

В четвертом случае ячейка дерева покрывает несколько ячеек регулярной решетки, поэтому происходит сканирование по всем ячейкам регулярной решетки, попадающим в область фотонов ячейки. Для каждой ячейки регулярной решетки формируются ее индекс, соответствующий дан-

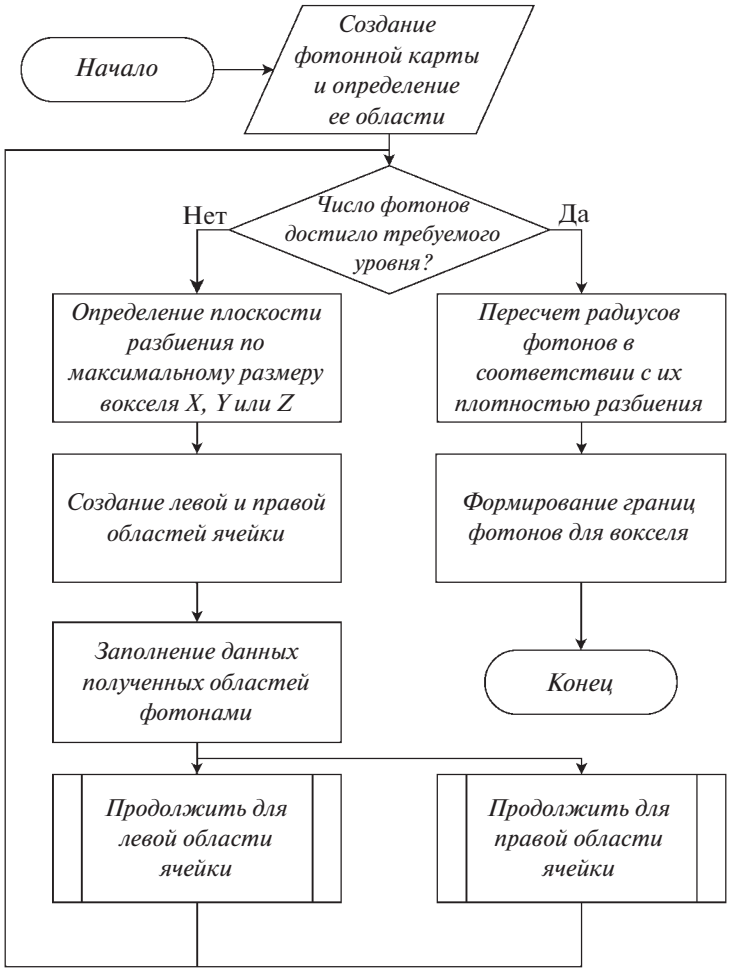


**Рис. 3.** Алгоритм построения kd-дерева.

ному уровню kd-дерева, и хэш-индекс таблицы (остаток от деления индекса на размер хэш-таблицы), которые добавляются в текущую ячейку таблицы. По окончании сканирования всех ячеек решетки происходит переход на следующий узел дерева, не превышающий заданный.

Необходимо отметить, что алгоритм, представленный на рис. 4, не содержит главный цикл обхода по всем ячейкам дерева, который заканчивается на последнем листе дерева. Кроме того, при формировании хэш-таблицы возможны коллизии, перебор которых может быть ускорен с помощью SIMD-операций, позволяющих проводить параллельную проверку сразу восьми ячеек. Такое решение позволяет повысить число коллизий при сохранении времени доступа к данным и уменьшить объем памяти для хранения хэш-таблиц.

В результате данный комбинированный алгоритм формирует дерево хэш-таблиц, которое на практике ограничено, как правило, двумя уровнями, что на порядок сокращает число обращений к разрозненным областям памяти.

## 4. ИСПОЛЬЗОВАНИЕ УСКОРЯЮЩЕЙ СТРУКТУРЫ В ВИДЕ ДЕРЕВА ХЭШ-ТАБЛИЦ

Метод обратных фотонных карт может использоваться как для получения яркости объемного рассеивания, так и для получения яркости на поверхности сцены. В любом из этих случаев предложенная структура пространственного разбиения применяется одновременно и для ускорения трассировки лучей, необходимой при сборе освещенности, и для ускорения поиска фотонов, сферы интегрирования которых попадают в точку пересечения луча с геометрией.

При поиске точки пересечения луча с геометрическим объектом или сферой интегрирования фотона ускоряющие структуры используются для ограничения количества элементов геометрии или сфер интегрирования. Для получения яркости объемного рассеивания необходимо проверять пересечения этим лучом сразу нескольких сфер интегрирования фотонов, находящихся на его пути. В случае поиска фотонов, сферы интегрирования которых покрывают точку пересече-

ния луча с поверхностью сцены, необходимо производить поиск ячейки ускоряющей структуры, содержащей эту точку.

### *4.1. Алгоритм поиска фотонов*

При расчете яркости вторичного и каустического освещений необходимо найти ячейку в ускоряющей структуре фотонной карты, фотонам которой будет передана энергия источника света. Следующий этап – это проверка принадлежности точки пересечения луча с поверхностью со сферами интегрирования фотонов. Исходными данными для поиска ячейки являются только координаты точки пересечения луча с геометрическим объектом.

В регулярных структурах пространственного разбиения поиск ячейки осуществляется с помощью прямого доступа по ее индексу, вычисляемому по координатам начала области, ее размеру и разрешению. При использовании хеш-таблиц поиск ячеек осуществляется путем формирования индекса в регулярной решетке и соответствующего ей хэш-индекса. Если при обращении к ячейке хэш-таблицы возникает коллизия (ячейка хэш-таблицы содержит ряд данных, имеющих этот хэш-индекс), то она разрешается простым перебором и сравнением элементов таблицы с индексом решетки. Для ускорения процедуры перебора ее можно осуществлять с помощью SIMD-инструкций, которые выполняют проверку сразу восьми индексов. Возможны два результата: “элемент не найден” и “элемент найден”. Первый случай говорит о том, что область, в которую мы попали, пустая, и, следовательно, для данного луча яркость не формируется. Второй случай говорит о том, что мы могли попасть в область с фотонами, и необходимо получить данные о фотонах, находящихся в этой области. Поскольку данная реализация предполагает дерево хэш-таблиц, то результатом поиска в хэш-таблице может быть как конечный лист, содержащий список фотонов и их параметров, так и хэш-таблица следующего уровня, содержащая свою регулярную решетку. В последнем случае процедура поиска продолжается, пока конечный уровень не будет достигнут. В большинстве случаев уровень хэш-таблиц не превышает двух. Алгоритм поиска фотонов в многоуровневой хэш-таблице представлен на рис. 5.

Конечным этапом поиска фотонов является проверка на принадлежность точки пересечения сферам интегрирования фотонов и формирование соответствующего списка фотонов. Проверка на принадлежность точки сфере интегрирования также может выполняться с помощью SIMD-инструкций, что позволяет существенно ускорить процедуру проверки.

### *4.2. Алгоритм трассировки лучей*

Трассировка лучей в структурах с регулярным разбиением пространства выполняется с помощью различных модификаций метода быстрого обхода сетки [23]. Регулярные разбиения позволяют добиться высокой эффективности трассировки только в сценах с равномерным распределением объектов. В случае использования хеш-таблиц трассировка лучей в них осуществляется аналогичным образом, поскольку пространство разбито регулярно. Наличие нескольких уровней таких разбиений позволяет добиться лучших результатов для сцен с неравномерным распределением объектов.

Трассировка лучей в пространственных структурах, построенных с помощью бинарных деревьев, из одной ячейки в другую ячейку предполагает перемещение по древовидной структуре как вверх, так и вниз. Это осуществляется тремя основными способами: классическим алгоритмом, использующим стек [24], алгоритмом kd-restart и алгоритмом kd-backtrack [25]. Каждый из алгоритмов обладает как преимуществами, так и недостатками. Алгоритм на основе стека требует хранения и оперирования стеком для каждого обрабатываемого луча, что является неэффективным в случае массового распараллеливания из-за частых обращений к памяти и необходимости работать с большим количеством стеков. Алгоритм kd-restart предполагает отсутствие перемещений по дереву вверх, что позволяет полностью избежать использования стека, однако требует дополнительных операций перемещения по дереву. Поэтому в случае использования многоуровневого дерева эффективность этого алгоритма заметно снижается. Алгоритм kd-backtrack хранит в каждой ячейке дерева не только ссылку на потомков, но и ссылку на родителя, а также bounding-box текущего уровня. Очевидно, этот алгоритм требует большего количества памяти для хранения структуры, однако позволяет избавиться от использования стека и в ряде случаев уменьшить количество необходимых для трассировки операций движения по дереву.

Структура, комбинирующая kd-дерево с хэш-таблицами, позволяет объединить в себе преимущества алгоритмов обеих пространственных структур. Этот алгоритм представляет собой последовательный обход древовидной структуры хеш-таблиц, внутри которых осуществляется обход ячеек таблицы, пересекаемых лучом. В простейшем виде этот алгоритм имеет рекурсивный вид, представленный на рис. 6.

Трассировка лучей представляет собой поиск пересечений луча со сферами интегрирования фотонов, находящихся на глубине трассы луча. Для выполнения трассировки требуется произвести последовательный обход всех ячеек про-

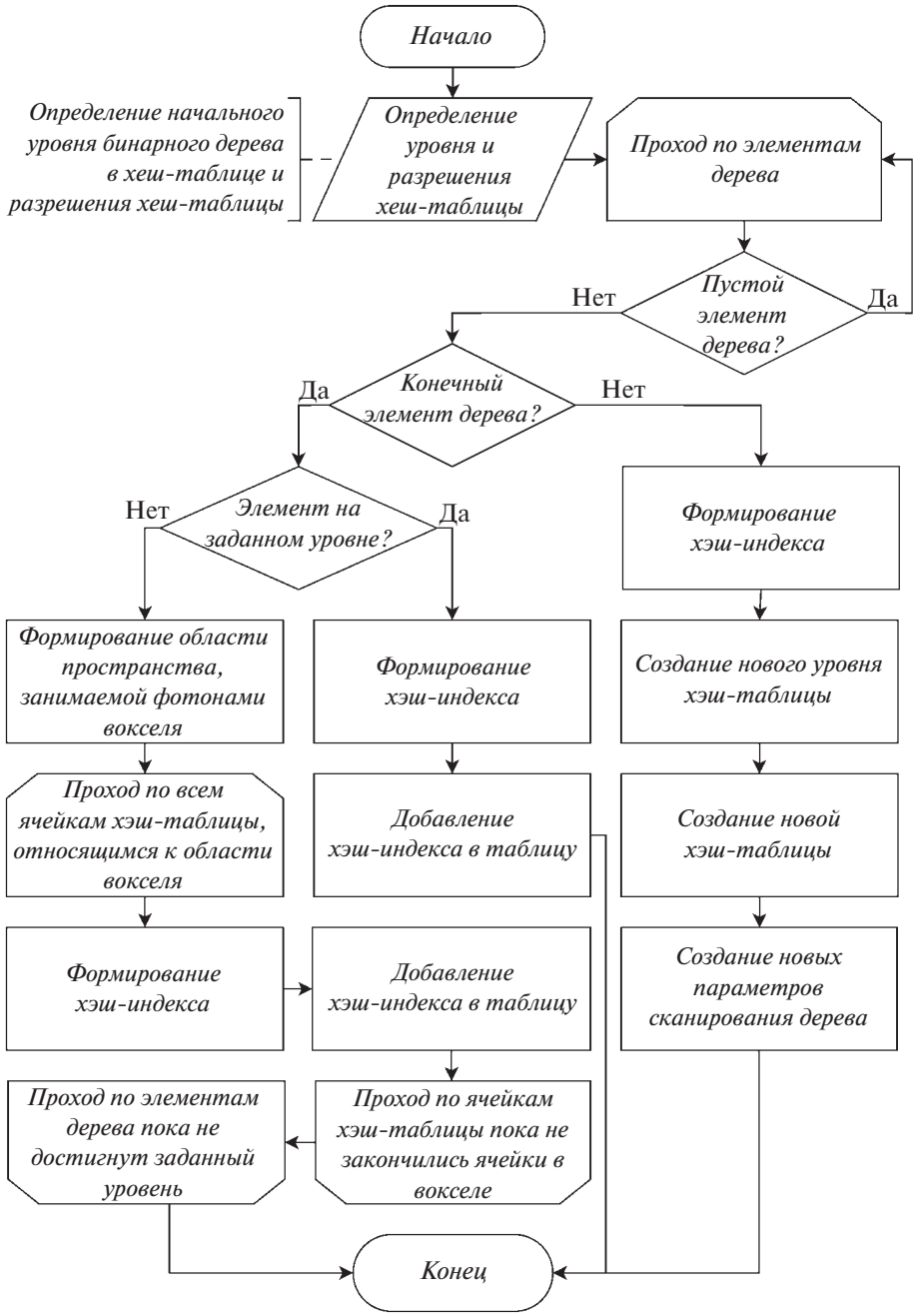


**Рис. 4.** Алгоритм построения хеш-таблиц.

странственной структуры, находящихся на глубине трассы луча, и проверить все пересечения луча со сферами внутри каждой из ячеек. Для начала обхода необходимо определить координаты точки пересечения луча с областью определения пространственной структуры. Если источник луча находится внутри ограничивающего параллелепипеда, координатами точки будут являться координаты источника луча. Если пересечения с параллелепипедом нет, трассировка завершается.

После нахождения начальной точки трассы луча происходит поиск ячейки, содержащей эту точку (алгоритм поиска описан в предыдущем разделе). В каждой ячейке пространственной структуры определяется локальная глубина трассы луча в ячейке и на этой глубине осуществляется поиск точек пересечения луча со сферами интегрирования фотонов ячейки. Формируется список пересеченных фотонов.

Алгоритм движения луча между ячейками осуществляется следующим образом. На текущем

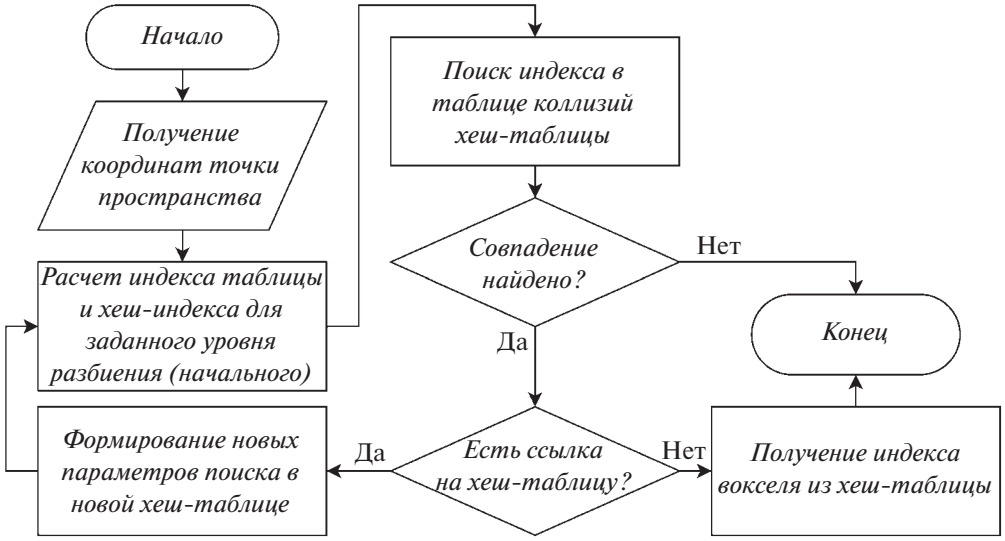


**Рис. 5.** Алгоритм поиска фотонов.

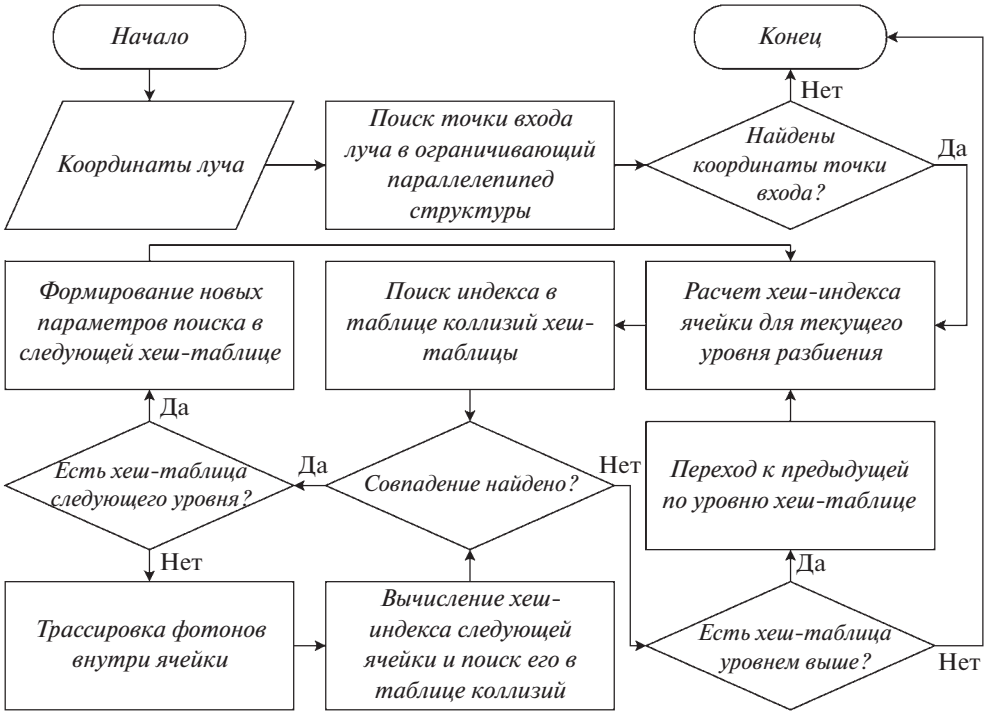


**Рис. 6.** Алгоритм трассировки фотонов.

уровне хеш-таблицы в области ее определения происходит инициализация быстрого обхода регулярной сетки. Для этого рассчитываются расстояния до ближайшей границы области и до границы ячейки. Затем в пределах полученной глубины происходит последовательный обход всех ячеек. Если полная глубина трассы луча не была выбрана на данном уровне хэш-таблицы, то происходит переход на смежную таблицу методом, аналогичным движению в kd-дереве. Алгоритм продол-

**Рис. 7.** Тестовые сцены: Cornell Box слева, Room2 справа сверху, Light Guide справа снизу.

жает движение по дереву хэш-таблиц, пока луч либо не покинет область определения пространственной структуры, либо не исчерпает свою глубину.

## 5. РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ АЛГОРИТМА ДЕРЕВА ХЕШ-ТАБЛИЦ В ПРОГРАММНОМ КОМПЛЕКСЕ LUMICEPT

Алгоритм многоуровневых хэш-таблиц был реализован в программном комплексе Lumicept [26] для расчета яркости вторичного и каустического освещений методом двунаправленной стохастической трассировки лучей с использованием обратных фотонных карт. Алгоритм был реализован на центральном процессоре с поддержкой параллельных и распределенных вычислений на серверных компьютерах. Распараллеливание вычислений было реализовано не только для процедуры поиска фотонов, но и для построения фотонных карт. В предыдущей работе [22, 27] было показано, что использование $N$ фотонных карт с $M$ фотонами эффективнее, чем использование одной большой фотонной карты с $N * M$ фотонами. В качестве естественного решения для реализации параллельного алгоритма разбиения одной фотонной карты на $N$ карт было предложено группировать фотонные карты по частям объектов сцены. Предварительно фотонные карты сортируются по числу фотонов и в параллельном режиме первыми начинают обрабатываться карты с наибольшим числом фотонов. В результате последними обрабатываются карты с малым числом фотонов, что позволяет минимизировать потери, связанные с простоем процессов на конечной стадии обработки фотонных карт.

**Таблица 1.** Результаты профилирования процесса рендеринга сцен в системе Lumicept

| Этап рендеринга | Занимаемое процессорное время | | |
|---|---|---|---|
| | Cornell Box | Room2 | Light Guide |
| Поиск фотонов | **24%** | **59%** | **28%** |
| Трассировка фотонов | 11% | 6% | 15% |
| Построение kd-дерева | 2% | 1% | 4% |
| Поиск пересечения с геометрией | 8% | 11% | 22% |
| Расчет яркости | 16% | 5% | 18% |

В программном комплексе был проведен сравнительный анализ разработанного решения, использующего многоуровневые хэш-таблицы, с решением, основанным на использовании бинарного дерева. Сравнение происходило для трех типов сцен: классической Cornell Box, интерьерной сцены Room2 и сцены со светопроводящими элементами Light Guide. Изображения сцен представлены на рис. 7.

Результаты профилирования данных сцен для разбиения фотонов алгоритмом, использующим бинарное дерево, представлены в табл. 1.

Очевидно, что операция поиска пересечения луча с фотонами занимает наибольшее время рендеринга. Поэтому ключевым фактором ускорения процесса рендеринга является ускорение процедуры поиска пересечения луча с фотонами в процессе сбора вторичной и каустической освещенностей. Таким образом, оптимизация структур пространственного разбиения, уменьшающая количество обращений к данным фотонных карт в процессе поиска фотонов, может существенно ускорить процесс рендеринга.

Для сравнения скорости работы предложенного и существующего решений был произведен ряд тестовых вычислений на программном комплексе Lumicept. Анализировалась скорость работы

**Таблица 2.** Сравнение скорости рендеринга сцены Cornell Box при помощи kd-дерева и двухуровневой хэш-таблицы

| Используемый метод | Время рендеринга, с | Количество фаз | Время одной фазы, с | Прирост скорости |
|---|---|---|---|---|
| kd-three | 1819 | 729 | 2.5 | – |
| Two-level hash table | 1810 | 840 | 2.15 | 21% |

**Таблица 3.** Сравнение скорости рендеринга сцены Room2 при помощи kd-дерева и двухуровневой хэш-таблицы

| Используемый метод | Время рендеринга, с | Количество фаз | Время одной фазы, с | Прирост скорости |
|---|---|---|---|---|
| kd-three | 1960 | 63 | 31.11 | – |
| Two-level hash table | 1954 | 90 | 21.7 | 43% |

системы рендеринга с существующим алгоритмом пространственного разбиения фотонных карт в виде kd-дерева и разработанным методом многоуровневых хэш-таблиц. Исследование показало, что для данных сцен хэш-таблицы были ограничены двумя уровнями разбиения. Анализ выполнялся для трех сцен (классической Cornell Box, интерьерной сцены Room2 и сцены со светопроводящими элементами Light Guide), представленных на рис. 7.

Для проведения экспериментов использовался компьютер со следующими характеристиками:

• Процессор Intel Core i5 11400, 2.6 ГГц.

• Оперативная память Kingmax Zeus Dragon, DDR4, 32 ГБ, 2666 МГц.

• Использовано 6 ядер с поддержкой 12 потоков.

При каждом проведении эксперимента проводился рендеринг изображения в течение 30 минут. В качестве показателя скорости использовалось среднее количество выполненных фаз рендеринга. Результаты сравнения для Cornell Box, Room2 и Light Guides представлены в таблицах 2, 3 и 4 соответственно.

Анализ результатов сравнения показал, что реализованная в данной работе структура пространственного разбиения фотонных карт, построенная на многоуровневой хэш-таблице, обеспечивает ускорение процесса рендеринга на 21–43% для большинства типовых сцен.

## 6. ЗАКЛЮЧЕНИЕ

Был проведен анализ существующих методов построения ускоряющих структур для обеспечения быстрого доступа к данным фотонных карт. В качестве двух основных решений были рассмотрены бинарные kd-деревья и регулярные пространственные решетки. Высокая адаптивность и качество пространственного разбиения, обеспечиваемая kd-деревом, имеет существенный недостаток, связанный со значительным временем доступа к данным фотонной карты. Этот недостаток является критическим при выполнении одновременной операции поиска фотонов на ряде потоков на многоядерных компьютерах. С точки зрения времени доступа к данным фотонной карты наиболее подходящим решением является регулярная пространственная решетка. Однако, существенным недостатком данного решения является низкая адаптивность пространственного разбиения, что приводит к значительному увеличению объема памяти для хранения ускоряющей структуры. Предложенное решение комбинирует адаптивность kd-дерева с высокой скоростью доступа к данным фотонной карты, обеспечиваемой регулярной структурой пространственной решетки. Использование многоуровневой хэш-таблицы позволяет уменьшить число коллизий на каждом ее уровне, сохранить адаптивность разбиения фотонов и на порядок сократить количество перемещений по уровням хэш-таблицы. Кроме того, использование SIMD-инструкций позволяет ускорить процесс поиска данных среди коллизий хэш-таблицы.

**Таблица 4.** Сравнение скорости рендеринга сцены Light Guide при помощи kd-дерева и двухуровневой хэш-таблицы

| Используемый метод | Время рендеринга, с | Количество фаз | Время одной фазы, с | Прирост скорости |
|---|---|---|---|---|
| kd-three | 1802 | 305 | 5.91 | – |
| Two-level hash table | 1811 | 373 | 4.85 | 22% |

## ФИНАНСИРОВАНИЕ

Работа была выполнена при финансовой поддержке Российского научного фонда, грант № 22-11-00145.

## СПИСОК ЛИТЕРАТУРЫ

1. *Фролов В.А., Волобой А.Г., Ершов С.В., Галактионов В.А.* Современное состояние методов расчета глобальной освещенности в задачах реалистичной компьютерной графики // Труды Института системного программирования РАН. 2021. Т. 33. № 2. С. 7–48.
2. *Georgiev I., Krivanek J., Davidovic T., Slusallek P.* Light Transport Simulation with Vertex Connection and Merging // ACM Transactions on Graphics. 2012. V. 31. № 6. P. 1–10.
3. *Veach E., Guibas L.J.* Metropolis light transport, in: Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques / SIGGRAPH '97, ACM Press/Addison-Wesley Publishing Co., USA. 1997. P. 65–76. URL: https://doi. org/ https://doi.org/10.1145/258734.258775. doi:. 258775. https://doi.org/10.1145/258734.258775
4. *Wenzel J.* Light Transport on Path-Space Manifolds, Ph.D. thesis. 2013.
5. *Kaplanyan A.S., Hanika J., Dachsbacher C.* The natural-constraint representation of the path space for efficient light transport simulation // ACM Trans. Graph. 2014. V. 33. URL: https://doi.org/10.1145/ 2601097.2601108
6. *Bitterli B., Jakob W., Novák J., Jarosz W.* Reversible jump metropolis light transport using inverse mappings, 2017. arXiv:1704.06835
7. *Gruson A., West R., Hachisuka T.* Stratified Markov Chain Monte Carlo Light Transport // Computer Graphics Forum. 2020. https://doi.org/10.1111/cgf.13935
8. *Jensen H.W.* Global illumination using photon maps / H.W. Jensen // Eurographics Workshop on Rendering techniques. Vienna: Springer, 1996. P. 21–30.
9. *Havran V., Herzog R., Seidel H.P.* Final gathering via reverse photon mapping // Computer Graphics Forum. Oxford, UK and Boston, USA: Blackwell Publishing, 2005. V. 24. № 3. P. 323–332.
10. *Zhdanov A., Zhdanov D.* The Backward Photon Mapping for the Realistic Image Rendering // Proc. 30th Conf. on Computer Graphics and Machine Vision (GraphiCon 2020), CEUR Workshop Proceedings. 2020. V. 2744. P. 1–12.
11. *Zhdanov A.D., Zhdanov D.D.* Progressive backward photon mapping // Programming and Computer Software. 2021. V. 47. № 3. P. 185–193.
12. *Bentley J.L., Friedman J.H.* Data structures for range searching // ACM Computing Surveys (CSUR). 1979. V. 11. № 4. P. 397–409.
13. *Toshiya Hachisuka, Henrik Wann Jensen*. Parallel progressive photon mapping on GPUs / In ACM SIGGRAPH ASIA 2010 Sketches (SA '10). New York, NY, USA: Association for Computing Machinery, Article 54, 1. https://doi.org/10.1145/1899950.1900004
14. *Hunt W., Mark W.R., Stoll G.* Fast kd-tree construction with an adaptive error-bounded heuristic // 2006 IEEE Symposium on Interactive Ray Tracing. IEEE, 2006. P. 81–88.
15. *Knoll Aaron*. A survey of octree volume rendering methods.
16. *Fabianowski Bartosz, Dingliana J*. Compact BVH Storage for Ray Tracing and Photon Mapping.
17. *Bradshaw Gareth, O'Sullivan Carol*. Sphere-tree construction using dynamic medial axis approximation / In Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation (SCA '02). New York, NY, USA, Association for Computing Machinery. 2002. P. 33–40. https://doi.org/10.1145/545261.545267
18. *Gino van den Bergen*. Efficient collision detection of complex deformable models using AABB trees // J. Graph. Tools. 1997. V. 2. № 4. P. 1–13. https://doi.org/10.1080/10867651.1997.10487480
19. *Gottschalk S., Lin M.C., Manocha D.* OBBTree: a hierarchical structure for rapid interference detection / In Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '96). New York, NY, USA: Association for Computing Machinery, 1996. P. 171–180. https://doi.org/10.1145/237170.237244
20. *Martin Stich, Heiko Friedrich, Andreas Dietrich*. Spatial splits in bounding volume hierarchies / In Proceedings of the Conference on High Performance Graphics 2009 (HPG '09). New York, NY, USA, Association for Computing Machinery. P. 7–13. https://doi.org/10.1145/1572769.1572771
21. *Wald Ingo, Günther Johannes, Slusallek Philipp, Cani Marie-Paule, Slater Mel*. Balancing Considered Harmful - Faster Photon Mapping using the Voxel Volume Heuristic / The European Association for Computer Graphics 25th Annual Conference EUROGRAPHICS 2004. Blackwell, 2004. V. 23. P. 595–603.
22. *Халимов Р.Р., Жданов Д.Д., Жданов А.Д.* Формирование эффективной пространственной структуры фотонных карт для ускорения процесса рендеринга // Труды Международной конференции по компьютерной графике и зрению "Графикон". 2022. Т. 32. С. 110–123.
23. *Havran Vlastimil*. Heuristic ray shooting algorithms. 2000.
24. *Hapala M., Havran V.* Review: Kd-tree Traversal Algorithms for Ray Tracing // Computer Graphics Forum. V. 30. P. 199–213. https://doi.org/10.1111/j.1467-8659.2010.01844.x
25. *Foley T., Sugerman J.* KD-tree acceleration structures for a GPU raytracer // In Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware (HWWS '05). New York, NY, USA, Association for Computing Machinery. P. 15–22. https://doi.org/10.1145/1071866.1071869
26. Lumicept Integra [Электронный ресурс]. URL: https://integra.jp/en/products/lumicept.
27. *Zhdanov A.D., Zhdanov D.D.* The two-level semi-synchronous parallelization method for the caustic and indirect luminance calculation in realistic rendering // CEUR Workshop Proceedings. 2020. V. 2744. P. 1–12.

*ПРОГРАММИРОВАНИЕ, 2023, № 3, с. 49–55*

## КОМПЬЮТЕРНАЯ ГРАФИКА И ВИЗУАЛИЗАЦИЯ

*УДК 519.6+004.891.3+616-018*

# МЕТОД АВТОМАТИЧЕСКОГО ПОИСКА ОПТИМАЛЬНОГО МАСШТАБА ПРИ ПРИМЕНЕНИИ ОБУЧЕННЫХ МОДЕЛЕЙ АНАЛИЗА ГИСТОЛОГИЧЕСКИХ ИЗОБРАЖЕНИЙ*

**© 2023 г. М. А. Пенкин[a],* (ORCID: 0000000280279333),**
**А. В. Хвостиков[a],** (ORCID: 0000000242177141), А. С. Крылов[a],*** (ORCID: 0000000199104501)**

*[a]Лаборатория математических методов обработки изображений, Факультет вычислительной математики и кибернетики, Московский государственный университет имени М.В. Ломоносова, Россия, 119991, Москва, Ленинские горы, д. 1, стр. 52*

**E-mail: penkin97@gmail.com*

***E-mail: khvostikov@cs.msu.ru*

****E-mail: kryl@cs.msu.ru*

Поступила в редакцию 09.01.2023 г.
После доработки 15.01.2023 г.
Принята к публикации 20.01.2023 г.

Подготовка входных данных для нейронной сети является ключевым шагом для достижения высокой точности ее предсказаний. Известно, что сверточные нейронные модели обладают низкой инвариантностью к изменению масштаба входных данных. Так, обработка многомасштабных полнослайдовых гистологических изображений сверточными сетями естественным образом поднимает вопрос выбора оптимального масштаба обработки. В данной работе эта задача решается путем итеративного анализа расстояний, выдаваемых сверточным классификатором, до разделяющей гиперплоскости при различных входных масштабах. Предлагаемый метод проверен на предобученной на данных PATH-DT-MSU глубокой архитектуре DenseNet121, решающей задачу по-патчевой классификации полнослайдовых гистологических изображений.

**DOI:** 10.31857/S0132347423030032, **EDN:** DEIZNX

## 1. ВВЕДЕНИЕ

Значительные технологические достижения привели к появлению инновационных методов цифровой визуализации в гистопатологии – разделе медицины, изучающем процессы и состояния в тканях организма.

Поиск решений известных проблем световой микроскопии таких как: затраты на подготовку и хранение препаратов, потеря качества препаратов со временем, сложность организации дистанционной работы, – привел к стремительной эволюции взаимодействия с гистологическими препаратами, и за последние 100 лет был осуществлен переход от световой микроскопии к цифровой [1].

На данный момент одним из передовых решений является сбор и анализ данных в виде полнослайдовых изображений (*англ.* Whole slide images (WSI)) [2]. Пример полнослайдового изображения стенки желудка из набора данных PATH-DT-MSU[1] приведен на рис. 1. Снимок сделан на оптическом увеличении (масштабе) ×40 с разрешением 65809 × 99600 пикселей.

Технология WSI продолжает набирать популярность среди патологоанатомов в диагностических, образовательных [3] и исследовательских целях. В большинстве случаев, полнослайдовые изображения представляют собой многомасштабные пирамиды из нескольких уровней, каждый из которых хранит изображение одного и того же образца ткани в различном разрешении. Нижний уровень содержит RGB-изображение образца в максимальном оптическом разрешении. Очередной уровень пирамиды хранит последовательно уменьшенную копию нулевого уровня, вплоть до небольшого обзорного снимка. Каждый уровень, включая нулевой, представляет собой композицию, составленную из небольших фрагментов квадратной формы, видимые границы между которыми отсутствуют. Такая структура представления обеспечивает высокую скорость доступа к любому участку полнослайдового

* Исследование выполнено при финансовой поддержке Российского научного фонда в рамках научного проекта № 22-41-02002.

[1] https://imaging.cs.msu.ru/en/research/histology/path-dt-msu

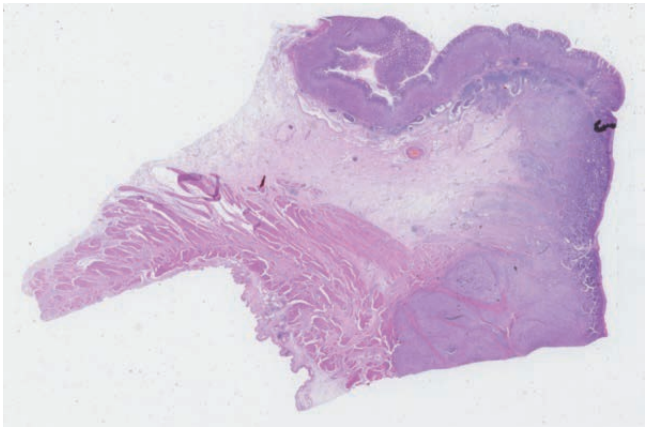

**Рис. 1.** Полнослайдовое гистологическое изображение стенки желудка из набора данных PATH-DT-MSU.

изображения на желаемом оптическом увеличении (масштабе).

Как любая технология, WSI имеет и недостатки [4, 5]. Основным недостатком WSI по сравнению со световой микроскопией является критическая необходимость в компьютерной автоматизированной обработке, без которой практическое использование полнослайдовых изображений в ручном режиме может оказаться неэффективным, дорогостоящим и требующим неприемлемо больших временных затрат. В зависимости от размера образца ткани размер изображений может достигать $100000 \times 100000$ пикселей и более.

В настоящее время в области компьютерной обработки изображений большое распространение приобрели методы глубокого обучения. В первую очередь – сверточные нейронные сети. Ввиду специфики медицинских изображений (отсутствие большого разнообразия выборки) обучение глубоких нейронных моделей крайне редко проводится с нуля, чаще всего используются предварительно обученные архитектуры с дальнейшей тонкой настройкой под конкретную медицинскую задачу. Примерами успешного применения сверточных сетей в медицинской обработке изображений являются работы [6–8]. Сверточные нейронные сети отображают изображения в признаковые представления. В обученных признаковых пространствах большой размерности ведется необходимая обработка, например, классификация или сегментация.

По своей природе, сверточные модели инвариантны к преобразованию сдвига, однако, напротив, обладают низкой инвариантностью к преобразованию масштаба [9]. Так, задача выбора масштаба обработки (см. рис. 2), оптимального для выбранной обученной нейронной сети, становится особо важной при автоматизации анализа многомасштабных полнослайдовых изображений, а также при разработке комплексных систем работы с гистологическими изображениями [10].

В данной работе предлагается алгоритм автоматического поиска величины оптимального масштаба при обработке полнослайдовых гистологических изображений сверточными нейронными сетями. Предлагаемый алгоритм протестирован на задаче классификации полнослайдовых гистологических изображений и основан на анализе расстояний до разделяющих гиперплоскостей при различных входных масштабах.

## 2. АКТУАЛЬНОСТЬ

Оценить важность обработки полнослайдовых гистологических изображений на оптимальном для выбранной нейронной модели масштабе можно на примере предобученного классификатора опухолей желудка на базе DenseNet121 [11].

Алгоритм обучен [12] классификации 5 типов тканей, имеющихся в наборе данных PATH-DT-MSU (WSS1, WSS2):

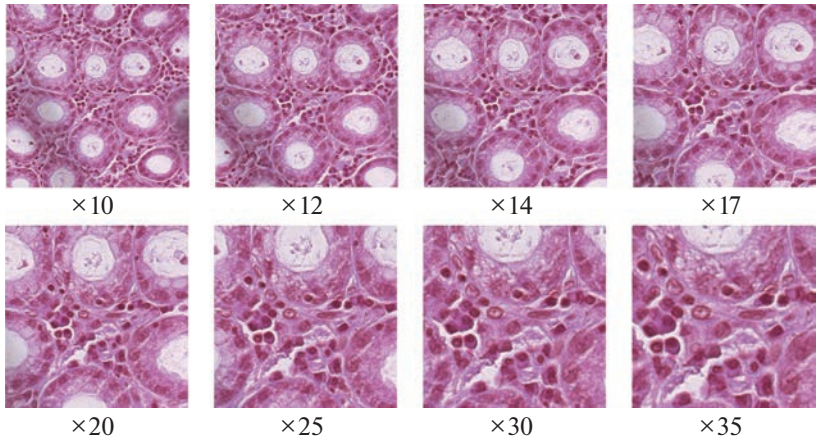


**Рис. 2.** Участок полнослайдового гистологического изображения, рассмотренный в различных масштабах.

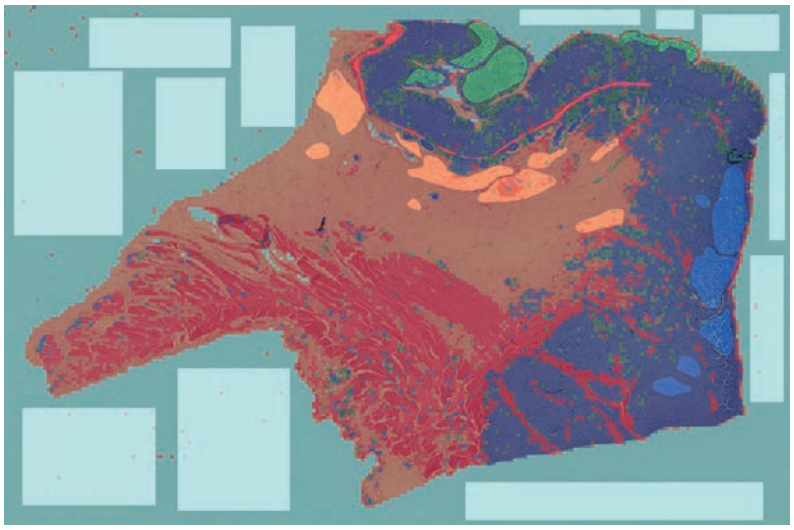

**Рис. 3.** Визуализация по-патчевой классификации полнослайдового гистологического изображения сверточной сетью DenseNet121. Области, размеченные специалистами, выделены увеличенной яркостью.

1. участки подслизистой основы, собственный мышечный слой желудка и субсерозные отделы (AT);

2. фон изображения (BG);

3. неизмененные участки собственной пластинки слизистой оболочки (LP);

4. неизмененные участки мышечной пластинки слизистой оболочки (MM);

5. участки аденокарциномы желудка (TUM).

Обучающая и валидационная выборки были сформированы из RGB фрагментов (224 × 224), вырезанных из WSI на масштабе ×20.

Высокое разрешение полнослайдовых изображений затрудняет применение методов глубокого обучения *напрямую*, как при классификации или сегментации обычных изображений архитектурами ResNet18 [13], VGG19 [14], InceptionNet [15], DenseNet121 [11], UNet [6]. Поэтому рассматриваемый классификатор опухолей желудка осуществляет *по-патчевую* классификацию (см. рис. 3):

• полнослайдовое изображение разделяется на небольшие фрагменты – патчи;

• сверточная нейронная сеть присваивает каждому фрагменту метку соответствующего класса;

• полученный набор классифицированных фрагментов склеивается в единое изображение.

Таким образом получается *грубая* карта сегментации, точность которой можно увеличить, уменьшая сдвиг окна (*англ.* stride) и усредняя предсказания.

В табл. 1 приведены значения точности классификации полнослайдового гистологического изображения на оптимальном увеличении (масштаб, на котором велось обучение) ×20, на меньшем увеличении – ×10 и на большем увеличении – ×40. При выборе масштаба ×10, отличного от оптимального для выбранной сверточной модели, наблюдается значительное снижение точности детекции класса LP и, как следствие, падение средней точности классификации. А при выборе масштаба ×40, также отличного от оптимального, падение точности классификации еще более существенно и наблюдается на классах LP и MM.

На практике, в отличие от рассмотренного примера, информация об оптимальном масштабе для выбранной модели может быть недоступна или же может быть утеряна информация об оптическом увеличении, на котором было получено анализируемое WSI изображение. Подобные риски усиливают практическую актуальность поставленной задачи.

Проблема выбора оптимального масштаба обработки встречается не только при попытках применить предобученную нейронную модель к полнослайдовым медицинским изображениям, но и при работе с обычными изображениями. К примеру, авторы [16] убедились в низкой инвариантности сверточных классификаторов ResNet50, ResNet18 [13], VGG19 [14], AlexNet [17] к изменению масштаба входных изображений из набора ImageNet [18].

**Таблица 1.** Точность классификации полнослайдового гистологического изображения в зависимости от входного масштаба (μ – средняя точность по пяти классам)

| | **AT** | **BG** | **LP** | **MM** | **TUM** | μ |
|---|---|---|---|---|---|---|
| ×10 | 87.80 | 99.84 | 61.74 | 17.47 | 98.19 | 73.01 |
| ×20 | 84.42 | 99.58 | 82.41 | 15.74 | 98.04 | **76.04** |
| ×40 | 78.85 | 99.46 | 15.01 | 7.35 | 94.52 | 59.04 |

Сверточный нейронный классификатор можно разделить на два блока:

• блок нелинейного отображения изображений в многомерное признаковое пространство;

• блок разделяющих гиперплоскостей, осуществляющих классификацию признаковых представлений на заданное число классов.

Предлагаемый в статье [16] алгоритм выбора масштаба сосредоточен на анализе процессов внутри блока нелинейного отображения. Алгоритм опирается на принцип сверточного фильтра [19] – одного из основных компонентов сверточных нейронных сетей. Этот принцип утверждает, что высокая корреляция фильтра с очередным паттерном сигнализирует об успехе распознавания. Таким образом, в работе [16] формулируется гипотеза, что изображения в оптимальном масштабе провоцируют экстремум активаций внутренних слоев. Выдвинутая гипотеза эмпирически доказывается и проверяется на задаче классификации как обычных изображений, так и медицинских гистологических изображений.

Данная работа, в отличие от исследования [16], обращает свое внимание на анализ процессов в блоке разделяющих гиперплоскостей при различных масштабах входных изображений.

## 3. ПРЕДЛАГАЕМЫЙ МЕТОД

В настоящей работе решение о выборе оптимального масштаба обработки полнослайдовых гистологических изображений принимается путем анализа расстояний до разделяющих гиперплоскостей при различных входных масштабах (см. Алгоритм 1).

**Алгоритм 1**

**Вход:**

– $I_{WSI}$ – WSI изображение;

– $\mathcal{A}[\Phi, W]$ – сверточный классификатор гистологических RGB-изображений, имеющих размер $\varepsilon \times \varepsilon \times 3$.

– $\Phi$ – нелинейное отображение изображений размером $\varepsilon \times \varepsilon \times 3$ в признаковое пространство размерности $M$.

– $W$ – матрица $\mathbb{R}^{M\times C}$, определяющая $C$ разделяющих гиперплоскостей классификатора $\mathcal{A}$.

$\mathcal{S} \leftarrow \{s_1, \ldots, s_N\}, |\mathcal{S}| = N$.

$\mathcal{P} \leftarrow \{x_k\}_{k=1}^{P}: x_k \in I_{WSI}, k \in \overline{1 \,..\, P}$.

**for** $k \leftarrow \overline{1 \,..\, P}$ **do**

  **for** $i \leftarrow \overline{1 \,..\, N}$ **do**

    $x_k^{\varepsilon}(s_i) \leftarrow U(x_k(s_i); \varepsilon)$.

    $f_i^{(k)} \leftarrow \Phi(x_k^{\varepsilon}(s_i)): f_i^{(k)} \in \mathbb{R}^M$.

  **end for**

  $X^{(k)} \leftarrow \{f_i^{(k)}\}_{i=1}^{N}: X^{(k)} \in \mathbb{R}^{N\times M}$.

  $D^{(k)} \leftarrow \dfrac{X^{(k)} \cdot W}{\|W\|}: D^{(k)} \in \mathbb{R}^{N\times C}$.

  $\hat{s}^{(k)} \leftarrow \text{argmax}(\text{ReduceMax}(D^{(k)}, \text{axis} = -1))$.

  $\hat{\mathcal{S}} \leftarrow \hat{\mathcal{S}} \cup \hat{s}^{(k)}$

**end for**

**Выход:**

– $\hat{\mathcal{S}} = \{\hat{s}^{(k)}\}_{k=1}^{P}$ – множество предлагаемых масштабов обработки изображения $I_{WSI}$ по $P$-патчам.

В качестве входа предлагаемый алгоритм получает: полнослайдовое изображение $I_{WSI}$, обученный классификатор $\mathcal{A}$ RGB-фрагментов гистологических изображений и число патчей $P$, многомасштабный анализ которых формирует результирующее множество предлагаемых масштабов $\hat{\mathcal{S}} = \{\hat{s}^{(k)}\}_{k=1}^{P}$. Поиск оптимальной величины масштаба обработки ведется среди масштабов $\mathcal{S} = \{s_1, \ldots, s_N\}$, где $s_1$ – наименьшее оптическое увеличение, а $s_N$ – наибольшее.

На полнослайдовом изображении $I_{WSI}$ случайным образом выбирается $P$ точек $\{x_k\}_{k=1}^{P}$.

Вокруг очередной точки $x_k$ на каждом рассматриваемом масштабе $s_i$ вырезается квадратный фрагмент $x_k^{\varepsilon}(s_i)$ формы $\varepsilon \times \varepsilon \times 3$. Признаковые представления $\{f_i^{(k)} \in \mathbb{R}^M\}_{i=1}^{N}$ патча $x_k^{\varepsilon}$ в $N$ масштабах образуют матрицу *объектов-признаков* $X^{(k)} \in \mathbb{R}^{N\times M}$. Обработка фрагмента $x_k^{\varepsilon}$, представленного в $N$ масштабах, завершается вычислением матрицы расстояний $D^{(k)}$ до каждой из $C$ разделяющих гиперплоскостей. Масштаб $\hat{s}^{(k)}$, отвечающий наибольшему расстоянию, сохраняется в множество предлагаемых масштабов $\hat{\mathcal{S}}$, и алгоритм переходит к следующей точке множества $\mathcal{P}$.

Окончательное решение об оптимальном масштабе обработки полнослайдового изображения $I_{WSI}$ принимается голосованием по множеству предлагаемых масштабов $\hat{\mathcal{S}}$ обработки фрагментов $\{x_k^{\varepsilon}\}_{k=1}^{P}$.

Предлагаемый метод мотивирован геометрическим смыслом линейного классификатора $\langle \omega, x \rangle$, инкапсулированного в большинство сверточных нейронных классификаторов. Геометрически линейный классификатор соответствует гиперплоскости с вектором нормали $\omega$. Величина

скалярного произведения $\langle \omega, x \rangle$ пропорциональна расстоянию от гиперплоскости до точки $x$, а его знак показывает, с какой стороны от гиперплоскости находится данная точка. Таким образом, линейный классификатор разделяет пространство на две части, и при этом одно полупространство он относит к положительному классу, а другое – к отрицательному. Нетрудно показать [20], что расстояние от точки $x$ до разделяющей гиперплоскости $\omega$ вычисляется как $\rho(x) = \dfrac{|\langle \omega, x \rangle|}{\|\omega\|}$.

В сверточные нейронные классификаторы встроены $C$ разделяющих гиперплоскостей, так что вектор $\omega$ превращается в матрицу $W$, столбцы которой задают нормали этих гиперплоскостей, а скалярное произведения – в матричное умножение.

## 4. РЕЗУЛЬТАТЫ

Представленный алгоритм поиска оптимального масштаба реализован на языке программирования Python 3 с использованием библиотеки глубокого обучения TensorFlow 2 и библиотеки OpenSlide для работы с полнослайдовыми изображениями.

Метод протестирован на задаче классификации полнослайдовых гистологических изображений стенок желудка из набора данных PATH-DT-MSU (WSS1, WSS2) обученной сверточной нейронной сетью DenseNet121 на масштабе ×20. Параметры алгоритма (см. Алгоритм 1) в таком случае принимают следующие значения: $\mathcal{S} = \{\times 10, \times 12, \times 14, \times 17, \times 20, \times 25, \times 30, \times 35, \times 43\}$, $\mathcal{A} :=$ DenseNet121, $M = 1024$, $C = 5$, $\varepsilon = 224$. Множество $\mathcal{S}$ составлено так, что $\dfrac{s_{i+1}}{s_i} \approx 1.2$, $i = \overline{1 \,..\, N-1}$.

Визуализации применения алгоритма к пяти различным многомасштабным патчам, относящихся к пяти различным классам (AT, BG, LP, MM, TUM) и вырезанных из полнослайдового изображения (рис. 1), представлены на рис. 4.

Для всех пяти фрагментов наибольший отступ (*англ.* margin) порождается классификатором именно от гиперплоскости, отвечающей референсному классу. Причем наблюдается заметная разница между величинами отступов от референсной гиперплоскости и от всех остальных.

Низкая устойчивость распознавания класса LP к изменению входного масштаба, отмеченная в секции 2 (см. табл. 1), находит свое отражение и

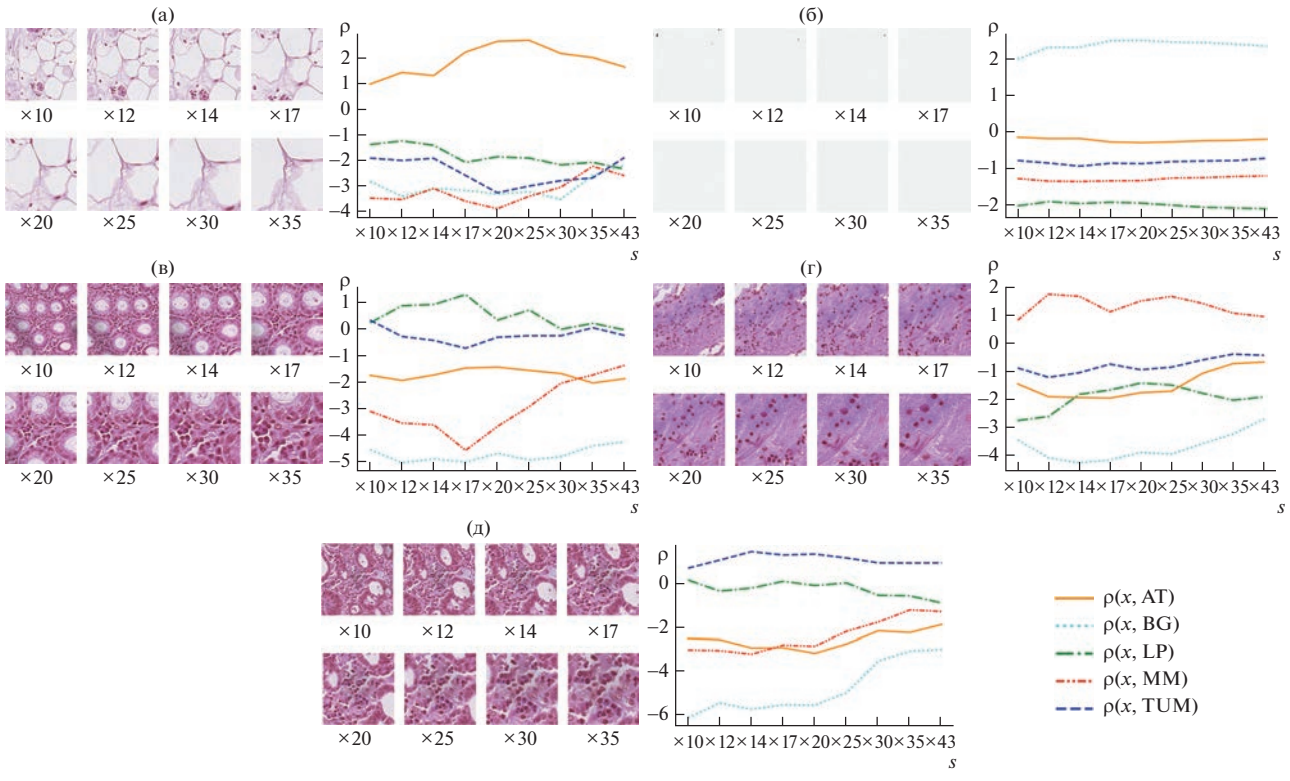


**Рис. 4.** Визуализация работы алгоритма на различных многомасштабных фрагментах полнослайдового изображения. На графиках показаны зависимости расстояний ($\rho$) до разделяющих гиперплоскостей в зависимости от масштаба ($s$) обрабатываемого фрагмента. (а) – фрагмент класса AT; (б) – фрагмент класса BG; (в) – фрагмент класса LP; (г) – фрагмент класса MM; (д) – фрагмент класса TUM.

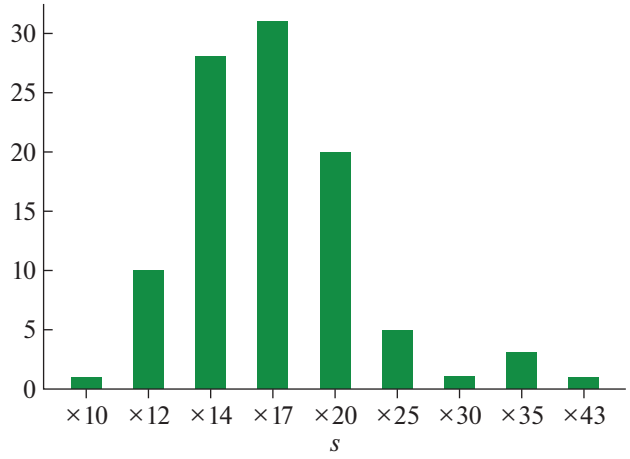


**Рис. 5.** Гистограмма по множеству предложенных алгоритмом масштабов $\hat{\mathcal{S}}$ для 100 случайных LP-патчей.

на рис. 4 (в): достигая экстремума на масштабе ×17 (близком к истинному оптимальному масштабу ×20, на котором велось обучение), кривая расстояний до референсной гиперплоскости существенно сближается с кривой расстояний до гиперплоскости TUM, при обработке патча на большем или меньшем масштабе в сравнении с ×17.

В зависимости от специфики текстур патча и ширины множества поиска $\mathcal{S}$ (под шириной здесь понимается $|s_N - s_1|$) варьируется характер зависимостей расстояний до гиперплоскостей от масштаба:

• сильная выпуклость (рис. 4 (а, в)) – алгоритм способен сделать однозначный вывод об оптимальном масштабе с высокой уверенностью, ширина множества $\mathcal{S}$ выбрана верно;

• монотонность (рис. 4 (г, д)) – алгоритм способен сделать некоторый вывод об оптимальном масштабе, однако следует увеличить ширину множества поиска $\mathcal{S}$;

• почти константность (рис. 4 (б)) – анализируемые масштабы равнозначны.

На рис. 5 представлена гистограмма по множеству предложенных алгоритмом масштабов $\hat{\mathcal{S}}$ для 100 случайных патчей, размеченных специалистами, как LP-участки WSI-изображения. Класс LP выбран ввиду особо низкой устойчивости модели DenseNet121 к распознаванию LP-фрагментов в различных масштабах (см. табл. 1), а значит высокой актуальности предлагаемого алгоритма для анализа многомасштабных LP-фрагментов WSI-изображения. Топ-3 предлагаемых масштаба: ×17, ×14 и ×20, соответственно. В доверительный интервал истинного оптимального масштаба ($\times 17 \leftarrow \times 20 \rightarrow \times 25$) попало 56% всех прогнозов из $\hat{\mathcal{S}}$. В более широкий доверительный интервал истинного оптимального масштаба ($\times 14 \leftarrow \times 20 \rightarrow \times 30$) попало 85% всех прогнозов из $\hat{\mathcal{S}}$.

## 5. ЗАКЛЮЧЕНИЕ

Данная работа обращает внимание на низкую инвариантность сверточных нейронных сетей к изменению масштаба входных данных и предлагает автоматический метод выбора оптимального масштаба обработки, что особенно актуально при анализе многомасштабных полнослайдовых гистологических изображений.

Условно разделив сверточную нейронную сеть на два блока: блок нелинейного отображения и блок разделяющих гиперплоскостей, – данная работа представляет метод выбора масштаба на основе анализа расстояний до разделяющих гиперплоскостей.

Предлагаемый алгоритм протестирован на задаче классификации полнослайдовых изображений стенок желудка из набора данных PATH-DT-MSU (WSS1, WSS2) обученной сверточной нейронной сетью DenseNet121.

Работа в данном направлении продолжается, и одним из возможных вариантов развития алгоритма является его адаптация под иерархическую композицию разделяющих гиперплоскостей.

## БЛАГОДАРНОСТИ

Исследование выполнено при финансовой поддержке Российского научного фонда в рамках научного проекта № 22-41-02002.

## СПИСОК ЛИТЕРАТУРЫ

1. *Park S., Pantanowitz L., Parwani A.V.* Digital imaging in pathology // Clinics in laboratory medicine. 2012. M. 32. № 4. C. 557–584.
2. *Pantanowitz L., Valenstein P.N., Evans A.J., Kaplan K.J., Pfeifer J.D., Wilbur D.C., Collins L.C., Colgan T.J.* Review of the current state of whole slide imaging in pathology // Journal of pathology informatics. 2012. V. 2. № 1. P. 36.
3. *Saco A., Bombi J.A., Garcia A., RamГrez J., Ordi J.* Current status of whole-slide imaging in education // Pathobiology. 2016. V. 83. № 2–3. P. 79–88.
4. *Farahani N., Parwani A.V., Pantanowitz L.* Whole slide imaging in pathology: advantages, limitations, and emerging perspectives // Pathol Lab Med Int. 2015. V. 7. № 23–33. P. 4321.
5. *Rojo M.G., GarcГa G.B., Mateos C.P., GarcГa J.G., Vicente M.C.* Critical comparison of 31 commercially available digital slide systems in pathology // International journal of surgical pathology. 2006. V. 14. № 4. P. 285–305.
6. *Ronneberger O., Fischer P., Brox T.* U-net: Convolutional networks for biomedical image segmentation // In International Conference on Medical image com-

puting and computer-assisted intervention. 2015. P. 234–241.

7. *Khvostikov A., Krylov A.S., Mikhailov I., Malkov P.* CNN Assisted Hybrid Algorithm for Medical Images Segmentation // In Proceedings of the 2020 5th International Conference on Biomedical Signal and Image Processing. 2020. P. 14–19.
8. *Getmanskaya A.A., Sokolov N.A., Turlapov V.E.* Multiclass U-Net Segmentation of Brain Electron Microscopy Data Using Original and Semi-Synthetic Training Datasets // Programming and Computer Software. 2022. V. 48. № 3. P. 164–171.
9. *Gong Y., Wang L., Guo R., Lazebnik S.* Multi-scale orderless pooling of deep convolutional activation features // In European conference on computer vision. 2014. P. 392–407.
10. *Khvostikov A.V., Krylov A.S., Mikhailov I.A., Malkov P.G.* Visualization of Whole Slide Histological Images with Automatic Tissue Type Recognition // Pattern Recognition and Image Analysis. 2022. V. 32. № 3. P. 483–488.
11. *Huang G., Liu Z., Van Der Maaten L., Weinberger K.Q.* Densely connected convolutional networks // In Proceedings of the IEEE conference on computer vision and pattern recognition. 2017. P. 4700–4708.
12. *Kingma D.P., Ba J.* Adam: A method for stochastic optimization // arXiv preprint arXiv:1412.6980. 2014.
13. *He K., Zhang X., Ren S., Sun J.* Deep residual learning for image recognition // Proceedings of the CVPR IEEE Conference. 2016. P. 770–778.
14. *Simonyan K., Zisserman A.* Very deep convolutional networks for large-scale image recognition // arXiv preprint arXiv:1409.1556. 2014.
15. *Szegedy C., Liu W., Jia Y., Sermanet P., Reed S., Anguelov D., Erhan D., Vanhoucke V., Rabinovich A.* Going deeper with convolutions // In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015. P. 1–9.
16. *Penkin M.A., Khvostikov A.V., Krylov A.S.* Optimal Input Scale Transformation Search for Deep Classification Neural Networks // In Graphicon-Conference on Computer Graphics and Vision. 2022. V. 32. P. 668–677.
17. *Krizhevsky A., Sutskever I., Hinton G.E.* Imagenet classification with deep convolutional neural networks // Communications of the ACM. 2017. V. 60. № 6. P. 84–90.
18. *Deng J., Dong W., Socher R., Li L.J., Li K., Fei-Fei L.* Imagenet: A large-scale hierarchical image database // In 2009 IEEE Conference on Computer Vision and Pattern Recognition. 2009. P. 248–255.
19. *Bolme D.S., Beveridge J.R., Draper B.A., Lui Y.M.* Visual object tracking using adaptive correlation filters // IEEE Computer Society Conference on Computer Vision and Pattern Recognition. 2010. P. 2544–2550.
20. *Mohri M., Rostamizadeh A., Talwalkar A.* Foundations of machine learning. MIT Press, 2018. 475 p.

*ПРОГРАММИРОВАНИЕ, 2023, № 3, с. 56–64*

## КОМПЬЮТЕРНАЯ ГРАФИКА И ВИЗУАЛИЗАЦИЯ

*УДК 004.925.83*

# ЭФФЕКТИВНАЯ ТЕХНОЛОГИЯ МОДЕЛИРОВАНИЯ В РЕАЛЬНОМ ВРЕМЕНИ ПОВЕРХНОСТИ ПОЛЯ ВЫСОТ НА КОНВЕЙЕРЕ ТРАССИРОВКИ ЛУЧЕЙ

**© 2023 г. П. Ю. Тимохин[a,*] (ORCID: 0000-0002-0718-1436),**
**М. В. Михайлюк[a,**] (ORCID: 0000-0002-7793-080X)**

*[a]ФГУ "ФНЦ Научно-исследовательский институт системных исследований РАН",*
*Россия, 117218, Москва, Нахимовский просп., 36, к. 1*

**E-mail: webpismo@yahoo.de*

***E-mail: mix@niisi.ras.ru*

Поступила в редакцию 09.01.2023 г.
После доработки 16.01.2023 г.
Принята к публикации 20.01.2023 г.

В данной статье, на примере поверхности поля высот, предлагается эффективная технология моделирования в реальном времени сложных процедурных объектов на конвейере трассировки лучей (RT-конвейере). Предлагаемая технология не перегружает стадию I-шейдера (шейдера пересечения), а распределяет вычислительную нагрузку между I-шейдером и AH-шейдером (шейдером любого подтвержденного пересечения). Ключевыми нововведениями в технологии являются ранняя отбраковка на стадии I-шейдера ограничивающих параллелепипедов (AABB), отобранных аппаратным блоком RT-конвейера, и концепция "прозрачного AABB", позволяющая перенести затратное вычисление пересечения луча с процедурным объектом на более позднюю стадию AH-шейдера. Также в работе описан ряд модификаций, сокращающих объем таких вычислений. Предложенная технология была реализована в программном комплексе на языках C++, GLSL и с помощью API Vulkan. Была исследована производительность разработанного решения при различных условиях трассировки лучей на задаче моделирования поверхности детализированного поля высот Пьюджет-Саунд. Полученные результаты подтвердили эффективность разработанной технологии и возможность ее применения в системах виртуального окружения, видеотренажерных комплексах, научной визуализации и др.

**DOI:** 10.31857/S0132347423030068, **EDN:** DENOFM

## 1. ВВЕДЕНИЕ

Аппаратное ускорение трассировки лучей, появившееся в серийных видеокартах NVidia, открыло новую эру в области высокореалистичной компьютерной графики реального времени. В настоящее время выпускается уже третье поколение графических процессоров с гибридной параллельной архитектурой RTX [1], которая включает в себя наряду с универсальными вычислительными ядрами (CUDA-ядрами) специализированные ядра нового типа – *ядра трассировки лучей* (RT-ядра). В отличие от CUDA-ядер, RT-ядра предназначены для эффективного решения узкого круга задач, обеспечивающих трассировку лучей (генерация лучей, расчет пересечения луча с треугольником и др.). Недавние исследования [2–5] показывают, что, несмотря на относительно небольшое количество, RT-ядра обладают высоким потенциалом для ускорения трассировки лучей в сложных трехмерных виртуальных сценах.

Одним из актуальных направлений применения аппаратно-ускоренной трассировки лучей является моделирование и визуализация в масштабе реального времени (со скоростью не менее 25 кадров в секунду) гладких поверхностей, основанных на *процедурных примитивах*. Такие примитивы отличаются от традиционных графических примитивов тем, что их геометрия не задается в явном виде, а синтезируется из точек пересечений с лучами, вычисляемых с помощью пользовательских процедур. Простым примером является сфера, пересечение луча с которой определяется аналитически. Сложнее дело обстоит с расчетом пересечений лучей с поверхностями, которые описаны с помощью сеточных функций, в частности, с *полями высот* [6]. В данной работе предлагается эффективная технология решения этой задачи на RT-ядрах в реальном времени для детализированных сеток высот (4K × 4K и выше), востребованных в современных системах виртуального окружения и симуляторах [7, 8]. Для про-

граммной реализации решения были использованы язык C++, шейдерный язык GLSL и API Vulkan.

## 2. ПРЕДЫДУЩИЕ ИССЛЕДОВАНИЯ

К настоящему времени опубликовано большое количество работ, предлагающих различные методы моделирования поверхностей на основе сеток высот. Так или иначе, результатом таких исследований является поверхность, в которой промежуточная информация между узлами сетки высот восполняется с помощью интерполяции. Глобально методы построения интерполированных поверхностей полей высот развиваются по двум основным направлениям: полигональное моделирование и трассировка лучей.

***Первое направление*** основано на соединении узлов сетки высот с помощью треугольных графических примитивов (*полигонов*), в результате которого формируется триангулированная (полигональная) модель поверхности поля высот. В работе [9] можно найти хороший обзор методов и алгоритмов полигонального моделирования полей высот. Преимуществом данного подхода является высокоразвитая программно-аппаратная поддержка распараллеливания обработки треугольников на графическом конвейере GPU, обеспеченная вычислительной мощностью тысяч CUDA-ядер. Ключевым ограничивающим фактором является число треугольников в формируемой модели поверхности: чем выше частота высотных данных в сетке высот, тем больше треугольников необходимо для построения ее поверхности и тем ниже скорость визуализации такой модели. В случае высокочастотных (и, как следствие, детализированных) сеток высот это приводит к необходимости разработки сложных адаптивных техник управления уровнем детализации [10]. Другой важной проблемой является интеграция таких адаптивных моделей в системы виртуального окружения, в которых требуется моделирование реалистичной светотеневой обстановки [11].

В данной статье исследуется ***второе направление***, при котором узлы сетки высот соединяются с помощью процедурных примитивов (*интерполянтов* [12]), а поверхность поля высот моделируется на основе поиска пересечений этих примитивов с лучами, испущенными из положения наблюдателя (обратная трассировка лучей [13]). Такой подход позволяет синтезировать высококачественные изображения, на которых поверхность поля высот интерполирована с попиксельной точностью. Главным недостатком является высокая вычислительная сложность, обусловленная необходимостью поиска точек, в которых лучи пересекают процедурные примитивы [14]. Для решения этой задачи были предложены различные пути: аппроксимация пересечения “луч-примитив” [15–18]; разработка ускоряющих структур данных [19–22], в том числе комбинированных с растеризацией [23]; распараллеливание лучей на CUDA-ядрах [24–26] и др. Общим ограничением предложенных решений является упорядоченный характер обработки пересечений вдоль луча, будь то бинарный поиск или проход по квадродереву. Это приводит к тому, что при появлении отдельных лучей с большим числом пересечений тормозится просчет всего изображения, даже с учетом распараллеливания обработки таких лучей на CUDA-ядрах.

С приходом новой архитектуры RTX появилась возможность распараллеливания трассировки лучей на выделенных ядрах GPU, RT-ядрах, экономя вычислительный потенциал универсальных CUDA-ядер. Аппаратно-ускоренный функционал, зашитый в RT-ядрах, позволяет существенно сократить время просчета лучей и, как следствие, повысить скорость синтеза изображений. Чтобы использовать возможности новой архитектуры RTX, виртуальная сцена должна быть представлена в виде специального *дерева ограничивающих объемов* (Bounding Volume Hierarchy, BVH), к листьям которого привязаны имеющиеся в сцене примитивы (поддерживаются и треугольные, и процедурные примитивы). Взаимодействие лучей с BVH-деревом осуществляется параллельно, согласно *конвейеру трассировки лучей* (RT-конвейеру) [27], который включает в себя ряд программируемых стадий (*шейдеров*): генерация луча (Ray Generation Shader, RG-шейдер), потенциальное пересечение (Intersection Shader, I-шейдер), любое подтвержденное пересечение (Any-Hit Shader, AH-шейдер), ближайшее подтвержденное пересечение (Closest Hit Shader, CH-шейдер) и промах (Miss Shader, M-шейдер). Суть работы RT-конвейера состоит в (a) определении пересекаемых лучом листьев BVH-дерева, (b) вычислении в них точек пересечения с примитивами и (c) выборе из этих точек ближайшей к началу луча. Этапы (a) и (c) выполняются RT-конвейером аппаратно (автоматически), а реализация этапа (b) зависит от типа примитива. Для полигональных моделей (треугольных примитивов) это также выполняется аппаратно, что существенно упрощает реализацию процесса трассировки. Если же в трассировке участвует процедурный примитив (как в данной работе), то RT-конвейер автоматически определяет только пересечение луча с *ограничивающим параллелепипедом* (Axis-Aligned Bounding Box, AABB) этого примитива, а задача реализации пересечения луча с процедурным примитивом (внутри AABB) возлагается на разработчика.

В настоящее время уже имеется ряд публикаций, в которых приведены примеры реализации на RT-конвейере пересечений лучей с различны-

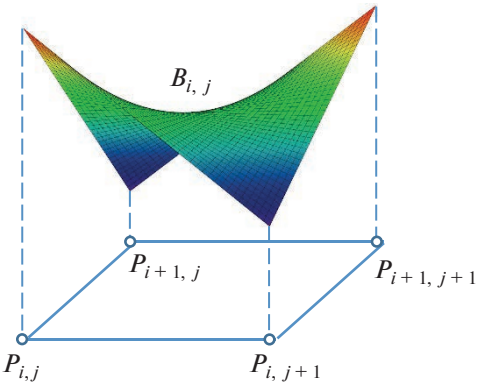


**Рис. 1.** Пример билинейного патча ($P_{i,j}$, ..., $P_{i+1,j+1}$ – узлы $(i,j)$-й ячейки сетки высот).

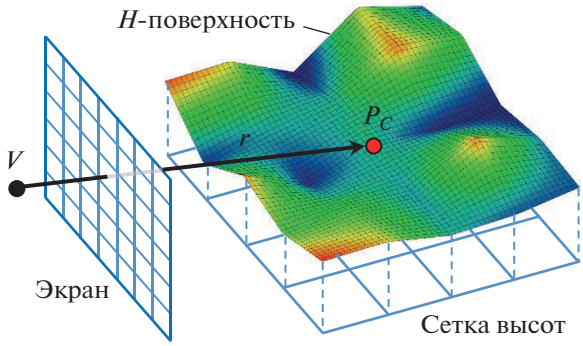


**Рис. 2.** Пример $H$-поверхности для сетки высот размера 4 × 4 ячейки.

ми процедурными примитивами [28–30]. Характерной чертой таких работ является реализация расчета пересечений на стадии I-шейдера. В данной работе, на примере задачи моделирования поверхности поля высот, предлагается модифицированная технология, при которой вычисление пересечений луча с процедурным объектом распределяется между стадиями I-шейдера и AH-шейдера. По сравнению с нашей первой реализацией (на стадии I-шейдера) [5] это позволило улучшить параллелизм решения и получить заметный выигрыш в производительности. В разделе 3 описана предлагаемая модифицированная технология, а в разделе 4 мы исследуем производительность созданного решения при различных условиях.

## 3. ПРЕДЛАГАЕМАЯ ТЕХНОЛОГИЯ

В данной работе мы будем рассматривать задачу синтеза с помощью трассировки лучей *кусочно-билинейной модели поверхности поля высот*. Пусть имеется сетка высот размера $m \times n$ ячеек. Обозначим через $B_{i,j}$ билинейную поверхность (патч), соответствующую $(i,j)$-й ячейке сетки высот (см. рис. 1), а через $H$ – кусочно-билинейную поверхность, составленную из смежных $B_{i,j}$-х патчей (далее $H$-поверхность). Проведем луч **r** из позиции наблюдателя $V$ через центр произвольного пиксела экрана. Нашей задачей будет реализация на RT-конвейере трассировки луча **r** до пересечения с $H$-поверхностью (см. рис. 2).

Технология трассировки луча **r** на RT-конвейере состоит из четырех этапов (см. рис. 3). На *этапе I* осуществляется генерация луча **r**, включающая задание его точки испускания и вектора направления (стадия RG-шейдера). На *этапе II* выполняется цикл обхода BVH-дерева вдоль луча **r**, целью которого является поиск ближайшей к наблюдателю точки $P_C$ пересечения луча **r** с $H$-поверхностью. На *этапе III*, если точка $P_C$ найдена, то вычисляется цвет луча в этой точке (стадия CH-шейдера), в противном случае, считается, что луч **r** “промахнулся”, и вычисляется цвет фона (стадия M-шейдера). На *этапе IV* осуществляется возврат к стадии RG-шейдера, где выполняется запись результирующего цвета луча **r** в буфер изображения. Реализации этапов I, III и IV являются во многом типовыми, их описание можно найти, например, в материалах [31]. В данной работе мы рассмотрим более подробно реализацию этапа II.

В работе [31] описан базовый подход к реализации этапа II на примере простых процедурных объектов типа сферы и параллелепипеда. В рамках данного подхода для моделируемого объекта в виртуальной сцене создается отдельный AABB, а также разрабатывается процедура вычисления внутри этого AABB ближайшей к наблюдателю точки пересечения луча с объектом. Разработанная процедура выполняется на стадии I-шейдера, после того как аппаратный блок обхода BVH-дерева обнаруживает пересечение луча с AABB (см. рис. 3). Для простых объектов, таких как сфера или параллелепипед, выполнение процедуры занимает очень короткое время, которое практически не влияет на работу RT-конвейера. В случае же $H$-поверхности время выполнения процедуры нахождения точки $P_C$ будет зависеть от числа проверок потенциальных пересечений луча **r** с патчами $H$-поверхности (см. рис. 2). Чем выше детализация сетки высот, тем больше будет таких проверок, что приводит к перегруженности I-шейдера, торможению цикла обхода BVH-дерева (согласно спецификации [32] I-шейдер в один момент времени работает с одним лучом и одним AABB) и, как следствие, падению производительности RT-конвейера.

Для решения описанной проблемы в данной работе предлагается ряд модификаций этапа II, направленных на более эффективное использование вычислительного ресурса RT-ядер (особен-

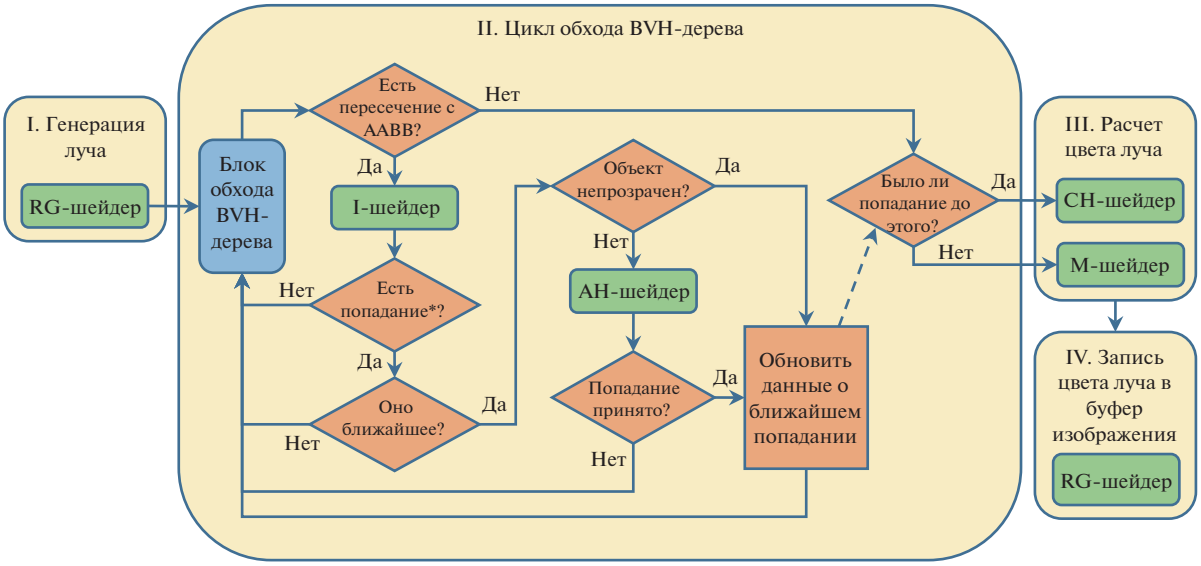


**Рис. 3.** Схема расширенной трассировки луча **r** на RT-конвейере (*попадание – подтвержденное пересечение).

но аппаратного блока обхода BVH-дерева) и сокращение числа проверок пересечений луча с $H$-поверхностью.

*Во-первых*, мы предлагаем создавать не один AABB для всей $H$-поверхности, а двумерный массив AABB, где каждый AABB охватывает свой блок $2^k \times 2^k$ патчей (см. рис. 4). Особенности построения такого массива описаны в нашей работе [5]. В процессе трассировки аппаратный блок обхода BVH-дерева проверяет все AABB, которые лежат на пути луча **r** (порядок проверки AABB произволен [32]), и определяет из них те, которые пересекаются с лучом **r**. Отобранные AABB аппаратный блок передает на стадию I-шейдера (см. рис. 3), а остальные исключает из цикла обхода BVH-дерева. Таким образом задача поиска точки $P_C$ вдоль всего луча **r** сводится к поиску точки $P_C$ только на участках луча **r**, проходящих через отобранные AABB, благодаря чему уменьшается число проверок потенциальных пересечений луча с патчами $H$-поверхности.

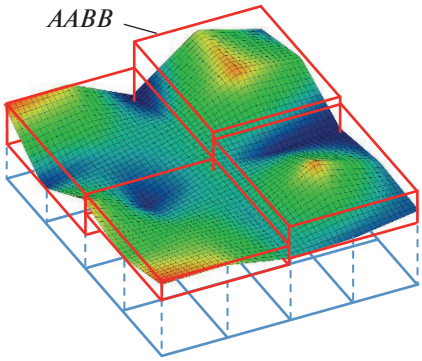


**Рис. 4.** Пример AABB-массива (каждый AABB охватывает блок 2 × 2 патча).

*Во-вторых*, для каждого AABB, отобранного аппаратным блоком, мы реализуем расширенный цикл обхода BVH-дерева (этап II на рис. 3). В отличие от базового подхода, при котором в цикле используется только стадия I-шейдера, в нашем расширенном цикле задействуются и стадия I-шейдера, и стадия AH-шейдера. Идея состоит в том, что на стадии I-шейдера мы не проводим никаких "тяжелых" расчетов, а вычисляем только точки входа и выхода луча из AABB (точнее их параметры $t_{in}$ и $t_{out}$ вдоль луча **r**). Это реализуется с помощью версии Slabs-теста [33, 34], модифицированной в нашей работе [5]. О вычисленном параметре $t_{in}$ точки входа мы сообщаем RT-конвейеру в конце стадии I-шейдера с помощью оператора *reportIntersectionEXT*. Получив эту информацию, RT-конвейер автоматически сравнивает ее с вычисленным на предыдущих итерациях параметром $t_{closest}$ точки ближайшего подтвержденного пересечения (блок "Оно ближайшее?" на рис. 3), и, если $t_{in} > t_{closest}$, то RT-конвейер прерывает обработку такого AABB и переходит к следующему. Такое решение позволяет отбраковывать на раннем этапе существенную часть AABB, отобранных аппаратным блоком (см. рис. 5), не переходя непосредственно к вычислению в них точек пересечения луча **r** с блоком патчей.

*В-третьих*, мы предлагаем концепцию "прозрачного AABB". Ее суть состоит в том, что изначально все AABB задаются потенциально про-

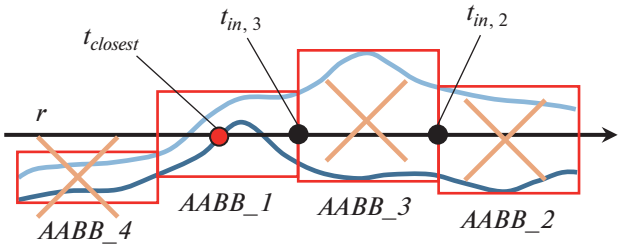


**Рис. 5.** Пример обработки AABB вдоль луча **r**: 1) AABB_4 не прошел отбор аппаратным блоком обхода BVH-дерева; 2) AABB_3 и AABB_2 прошли аппаратный отбор, но были отбракованы на стадии I-шейдера, так как $t_{in,3} > t_{closest}$ и $t_{in,2} > t_{closest}$; 3) AABB_1 прошел аппаратный отбор и стадию I-шейдера, и был допущен к вычислению $t_{closest}$ на стадии AH-шейдера.

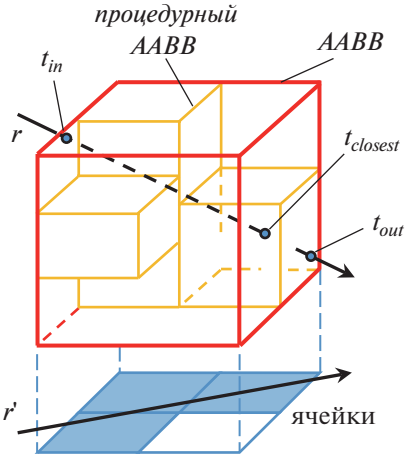


**Рис. 6.** Проход луча **r** через процедурные AABB внутри исходного AABB.

зрачными[1] по отношению к лучу **r**, а на стадии AH-шейдера для каждого AABB, прошедшего отбор I-шейдером, определяется, является ли он прозрачным или нет. AABB считается непрозрачным, если в нем луч ***r*** хотя бы раз пересекает блок патчей. Мы вычисляем параметр $t_{closest}$ точки ближайшего к наблюдателю пересечения и записываем значение этого параметра в атрибут луча (структуру полезной нагрузки луча) в конце стадии AH-шейдера. В этом случае RT-конвейер считает, что пересечение луча **r** с нашим процедурным объектом подтверждено (блок “Попадание принято?” на рис. 3), запоминает значение $t_{closest}$ и переходит к обработке следующего AABB. В противном случае (не найдено ни одного пересечения луча **r** с блоком патчей текущего AABB), мы сообщаем RT-конвейеру путем вызова оператора *ignoreIntersectionEXT* в конце AH-шейдера о необходимости проигнорировать текущий AABB и сразу перейти к обработке следующего AABB. Предложенная концепция “прозрачного AABB” позволяет перенести затратное вычисление пересечения луча **r** с процедурным объектом (блоком патчей) на более позднюю стадию AH-шейдера, чтобы реализовать на стадии I-шейдера описанную выше раннюю отбраковку AABB.

*В-четвертых*, мы ускоряем вычисление параметра $t_{closest}$ внутри исходного AABB с помощью *процедурных AABB*, ограничивающих патчи. Идея состоит в том, чтобы по мере продвижения от $t_{in}$ к $t_{out}$ вычислять пересечения луча **r** только с теми патчами, чьи процедурные AABB он пересекает (см. рис. 6). Для этого мы: 1) определяем ячейку сетки высот, которую пересекает трасса **r**' луча **r**; 2) вычисляем процедурный AABB для патча соответствующего этой ячейке (благо данные в узлах сетки высот нам известны); 3) проверяем пересечение процедурного AABB с лучом **r** с помощью модифицированной версии Slabs-теста, упомянутой выше. Если процедурный AABB пересечен лучом **r**, то мы вычисляем параметр $t_{closest}$ ближайшей к наблюдателю точки пересечения с помощью версии метода GARP [35], адаптированной для нашей задачи [5]. Если в результате вычисления мы получаем значение $t_{closest} > 0$, т.е. патч пересечен лучом **r**, то мы возвращаем это значение и прекращаем движение по лучу внутри AABB. В противном случае, мы переходим к следующей по трассе **r**' ячейке и повторяем описанную выше последовательность шагов, пока не будет вычислено положительное значение $t_{closest}$ или достигнута граница AABB (в этом случае мы присваиваем $t_{closest}$ значение $-1$).

*В-пятых*, при обходе процедурных AABB мы используем разработанный алгоритм устойчивого перехода к следующей по лучу **r**' ячейке. В классическом алгоритме обхода вокселов [36] продвижение вдоль луча реализуется на основе вычисления значений параметров $t'_s$ и $t'_t$ (см. рис. 7), а переход осуществляется в ячейку, соответствующую меньшей из накопленных сумм каждого из этих параметров. В нашей задаче кроме номеров ($i_{next}$, $j_{next}$) следующей ячейки необходимо также знать координаты точки $P_{next}$ входа в эту ячейку (для вычисления $t_{closest}$ внутри процедурного AABB). Из-за погрешности машинного представления действительных чисел может возникнуть ситуация, когда точка $P_{next}$ окажется вне границ следующей ячейки. Чтобы избежать остановки продвижения вдоль луча **r**' при таком расхождении данных, мы вычисляем номера ($i_{next}$, $j_{next}$) и координаты $P_{next}$ следующим образом:

[1] Чтобы это реализовать, мы устанавливаем флаг прозрачности *gl_RayFlagsNoOpaqueEXT* при вызове функции *traceRayEXT* генерации луча **r** на стадии RG-шейдера. Это позволяет разблокировать стадию AH-шейдера (по умолчанию она выключена в RT-конвейере) и продолжить выполнение цикла обхода BVH-дерева после стадии I-шейдера по ветке “Нет” после блока “Объект непрозрачен?” на рис. 3.

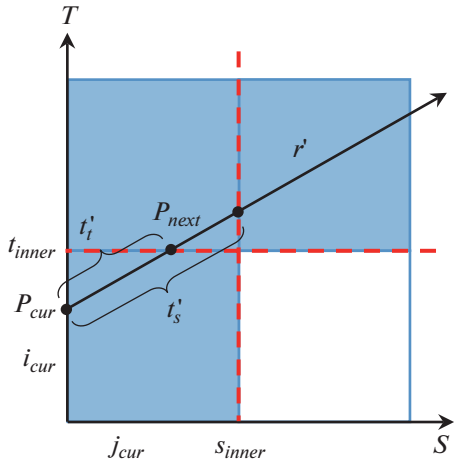


**Рис. 7.** Переход к следующей ячейке вдоль луча **r'**.

1) найдем номер $N$ *внутреннего* угла текущей ячейки, который пересекает луч **r'**:

$$N = \text{int}(|r_s'| \geq 0) + 2 \times \text{int}(|r_t'| \geq 0),$$

где $r_s'$, $r_t'$ – координаты вектора **r'** по осям S и T, а *int*() – операция преобразования булевского типа в целочисленный;

2) вычислим параметры $t_s'$ и $t_t'$ точек пересечений луча **r'** с прямыми $s_{inner}$, $t_{inner}$, образующими $N$-й внутренний угол (см. рис. 7);

3) запишем координаты $P_{next}$ точки входа и номера $(i_{next}, j_{next})$ следующей ячейки:

$$P_{next} = P_{cur} + \min(t_s', t_t')\mathbf{r'},$$

$$i_{next} = i_{cur} + \text{sign}(r_t') \cdot \text{int}(t_t' \leq t_s'),$$

$$j_{next} = j_{cur} + \text{sign}(r_s') \cdot \text{int}(t_s' \leq t_t'),$$

где $P_{cur}$ – координаты точки входа в текущую ячейку, а sign($x$) – функция взятия знака числа $x$, которая возвращает 1 при $x > 0$, $-1$ при $x < 0$ и 0 при $x$ равном 0 (случаи нулевых $r_s'$ и $r_t'$ обрабатываются отдельно).

Подводя итог, отметим, что при реализации предложенной технологии важно соблюдать “золотую середину” при выборе размера исходных блоков патчей. Если выбрать слишком маленький размер, то будет создаваться большое число AABB, и аппаратный блок обхода BVH-дерева не будет успевать их своевременно обрабатывать, а AH-шейдер будет недогружен работой. С другой стороны, при большом размере блока патчей вырастет число процедурных AABB и нагрузка на AH-шейдер, а аппаратный блок будет мало задействован. Количественное исследование этой закономерности и поиск “золотой середины” приведены в разд. 4.

## 4. РЕЗУЛЬТАТЫ

Предложенная технология была реализована в виде программного комплекса, осуществляющего моделирование и визуализацию поверхностей детализированных полей высот в реальном времени. Данный комплекс написан на языке C++, трассировка лучей реализована на RT-конвейере с помощью языка GLSL программирования шейдеров и API Vulkan (версия 1.3.204.1). С помощью разработанного комплекса было выполнено моделирование поверхностей на основе трех вариантов эталонной 16-битной сетки высот Пьюджет-Саунд [37, 38]: 4K × 4K, 8K × 8K и 16K × 16K[2]. В качестве аппаратной базы использовался персональный компьютер (Intel Core i7-6800K 3.40 ГГц, RAM 16 Гб DDR4), оборудованный видеокартой NVidia GeForce RTX 2080 (VRAM 8 Гб GDDR6, 2944 CUDA-ядер, 46 RT-ядер, драйвер NVidia DCH 512.59). Моделирование выполнялось при разрешении области вывода Full HD в условиях наблюдения, соответствующих средней сложности (см. рис. 8a) и высокой сложности (см. рис. 8b) трассировки лучей.

Для каждого из трех вариантов сетки высот мы провели серии экспериментов, в которых изменяли размер блока патчей от 4 × 4 до 1024 × 1024. После каждого изменения размера была измерена средняя скорость визуализации (в кадрах в секунду) в условиях средней и высокой сложности трассировки лучей, изображенных на рис. 8. Результаты замеров приведены на графиках рис. 9.

Проведенные эксперименты подтвердили наши теоретические предположения (см. конец разд. 3), касающиеся закономерности влияния размера блока патчей на производительность предлагаемого решения. Из графиков можно видеть, что “золотой серединой” будет выбор размера $d$ блока патчей из диапазона [16, 32]. Именно при этих значениях $d$ показатели производительности являются близкими к наибольшим, как при средней, так и при высокой сложности трассировки лучей. Сравнение полученных результатов с нашей первой реализацией (только на I-шейдере) [5] показало, что предложенное распределение вычислений между I-шейдером и AH-шейдером обеспечило прирост скорости визуализации до 20% (сравнение проводилось на тех же сетках высот, аппаратной базе и при тех же условиях наблюдения).

## 5. ЗАКЛЮЧЕНИЕ

В данной работе рассмотрена задача синтеза в реальном времени кусочно-билинейной модели

[2] Выражаем благодарность профессору Джону Реппи из Чикагского университета (факультет компьютерных наук) за помощь в предоставлении детализированной версии сетки высот Пьюджет-Саунд [38].

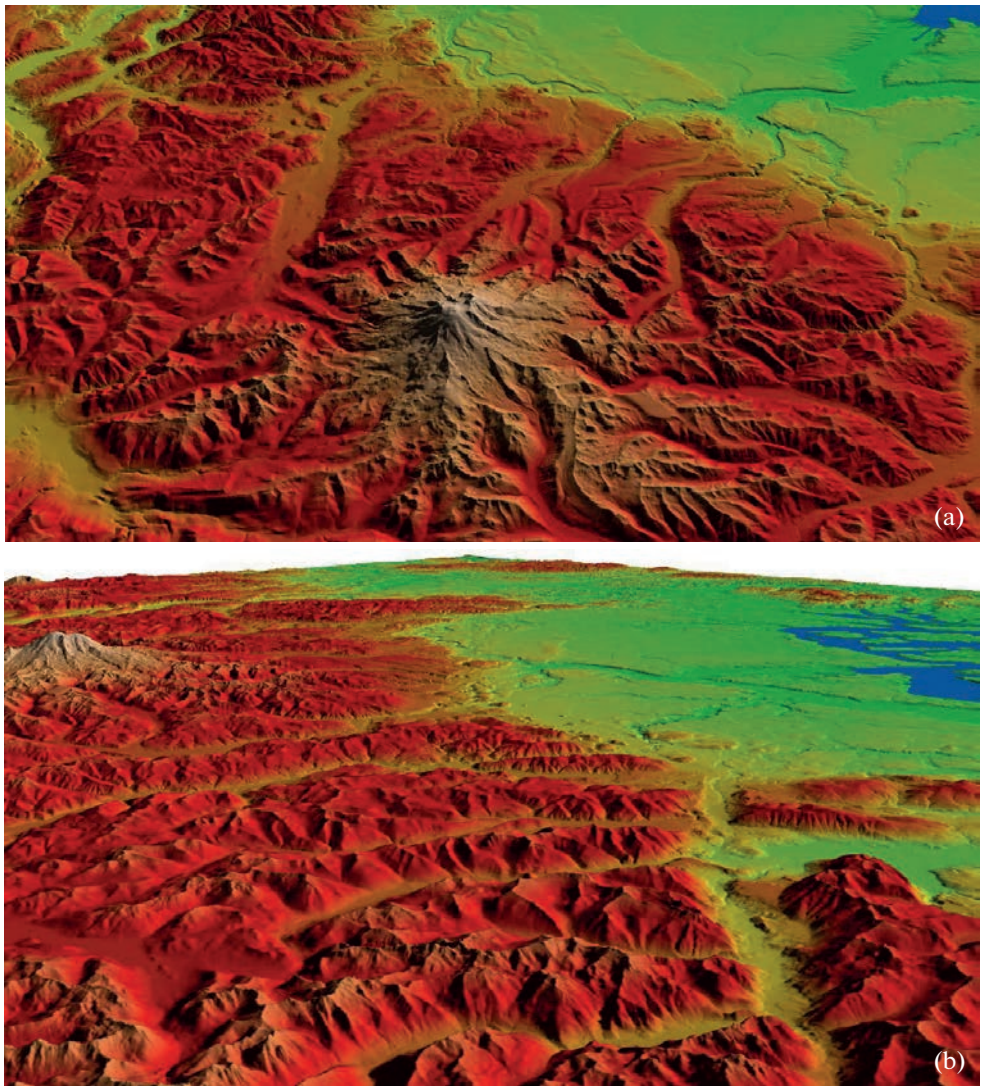


**Рис. 8.** Моделирование поверхности поля высот Пьюджет-Саунд с помощью предложенной технологии в условиях (a) средней сложности и (b) высокой сложности трассировки лучей.

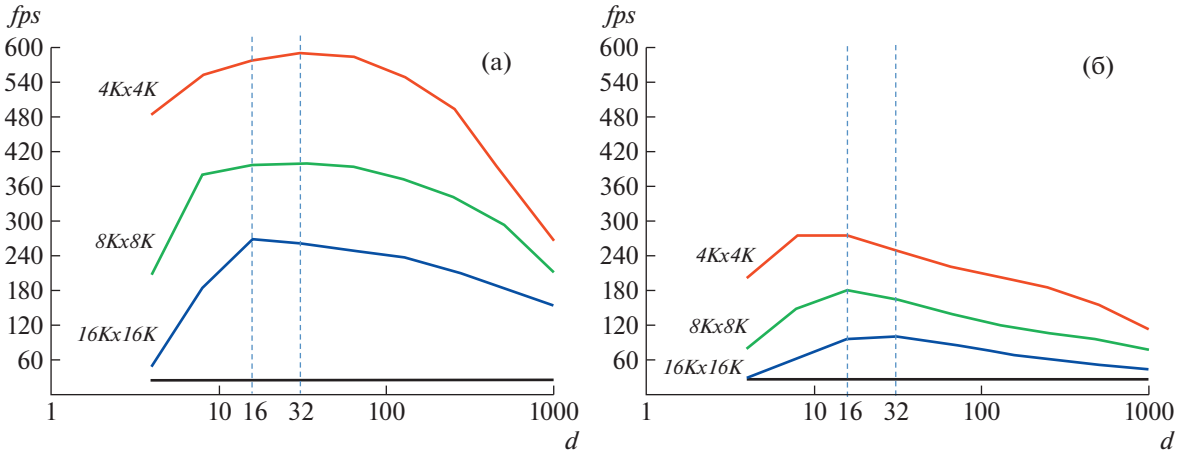


**Рис. 9.** Зависимость средней скорости визуализации от размера *d* блока патчей (логарифмическая шкала) для трех вариантов детализации сетки высот Пьюджет-Саунд при (a) средней и (b) высокой сложности трассировки лучей (нижняя черная линия означает порог в 25 кадров/секунду).

поверхности поля высот на конвейере трассировки лучей. Проблема состоит в том, что использование для рассматриваемой задачи базового подхода [31], применяемого для моделирования простых процедурных объектов типа сферы и параллелепипеда, приводит к перегруженности стадии I-шейдера и падению производительности RT-конвейера. Для решения этой проблемы была предложена эффективная технология, основанная на представлении моделируемой поверхности в виде блоков билинейных патчей и распределении обработки этих блоков (ограничивающих их AABB) между стадиями I-шейдера и AH-шейдера RT-конвейера. Ключевым нововведением такой обработки является ранняя отбраковка на стадии I-шейдера существенной части AABB, отобранных аппаратным блоком RT-конвейера, без непосредственного вычисления точек пересечения луча с процедурной поверхностью. Предложенная технология также включает в себя ряд модификаций, направленных на более эффективное использование RT-ядер и уменьшение количества затратных тестов пересечений типа “луч-билинейный патч”. В работе была исследована производительность созданного решения в условиях средней и высокой сложности трассировки лучей на эталонной сетке высот Пьюджет-Саунд в трех вариантах детализации. Проведенные исследования подтвердили эффективность предложенной технологии и возможность ее применения в системах виртуального окружения, научной визуализации, видеотренажерах и др.

## БЛАГОДАРНОСТИ

Публикация выполнена в рамках государственного задания ФГУ ФНЦ НИИСИ РАН “Проведение фундаментальных научных исследований (47 ГП)” по теме № FNEF-2022-0012 “Системы виртуального окружения: технологии, методы и алгоритмы математического моделирования и визуализации. 0580-2022-0012”.

## СПИСОК ЛИТЕРАТУРЫ

1. NVIDIA Ada GPU Architecture // NVIDIA Corporation. 2022. https://images.nvidia.com/aem-dam/Solutions/ geforce/ada/nvidia-ada-gpu-architecture.pdf
2. *Sanzharov V.V., Frolov V.A., Galaktionov V.A.* Survey of Nvidia RTX Technology // Programming and Computer Software. 2020. V. 46. № 4. P. 297–304. https://doi.org/10.1134/S0361768820030068
3. *Salmon J., McIntosh-Smith S.* Exploiting Hardware-Accelerated Ray Tracing for Monte Carlo Particle Transport with OpenMC // 2019 IEEE/ACM Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS). 2019. P. 19–29. https://doi.org/10.1109/PMBS49563.2019.00008
4. *Komarov E.A., Zhdanov D.D., Zhdanov A.D.* Caustic Illuminance Calculation with DirectX Raytracing // Programming and Computer Software. 2022. V. 48. № 3. P. 172–180. https://doi.org/10.1134/S0361768822030069
5. *Тимохин П.Ю., Михайлюк М.В.* Рендеринг детализированных полей высот в реальном времени с использованием аппаратного ускорения трассировки лучей // GraphiCon 2022: труды 32-й Международной конференции по компьютерной графике и машинному зрению (Рязань, 19–22 сентября 2022 г.). 2022. С. 124–135. https://doi.org/10.20948/graphicon-2022-124-135.
6. *Han J.* Introduction to Computer Graphics with OpenGL ES. 1st. ed. Boca Raton, CRC Press, 2018.
7. *Михайлюк М.В., Мальцев А.В., Тимохин П.Ю., Страшнов Е.В., Крючков Б.И., Усов В.М.* Система виртуального окружения VirSim для имитационно-тренажерных комплексов подготовки космонавтов // Пилотируемые полеты в космос. 2020. № 4 (37). С. 72–95. https://doi.org/10.34131/MSF.20.4.72-95
8. *Hongxiang R., Yicheng J., Liling C.* Real-time Rendering of Ocean in Marine Simulator // 2008 Asia Simulation Conference – 7th International Conference on System Simulation and Scientific Computing. 2008. P. 1133–1136. https://doi.org/10.1109/ASC-ICSC.2008.4675536.
9. *Pajarola R., Gobbetti E.* Survey of semi-regular multiresolution models for interactive terrain rendering // The Visual Computer. 2007. V. 23. № 8. P. 583–605. https://doi.org/10.1007/s00371-007-0163-2
10. *Li S., Zheng C., Wang R., Huo Y., Zheng W., Lin H., Bao H.* Multi-resolution terrain rendering using summed-area tables // Computers & Graphics. 2021. V. 95. P. 130–140. https://doi.org/10.1016/j.cag.2021.02.003
11. *Timokhin P.Yu., Mikhaylyuk M.V.* Computer Modeling and Visualization of Accurate Terrain Shadows in Virtual Environment System // Scientific Visualization. 2022. V. 14. № 2. P. 77–87. https://doi.org/10.26583/sv.14.2.07
12. *Cornel D., Horváth Z., Waser J.* An Attempt of Adaptive Heightfield Rendering with Complex Interpolants Using Ray Casting // Technical Report. arXiv:2201.10887v1. 2022. P. 1–9. https://doi.org/10.48550/arXiv.2201.10887.
13. *Frolov V.A., Voloboy A.G., Ershov S.V., Galaktionov V.A.* Light Transport in Realistic Rendering: State-of-the-Art Simulation Methods // Programming and Computer Software. 2021. V. 47. № 4. P. 298–326. https://doi.org/10.1134/S0361768821040034
14. *Parker S., Shirley P., Livnat Y., Hansen C., Sloan P.-P.* Interactive Ray Tracing for Isosurface Rendering // VIZ’98: proceedings of the IEEE Visualization 98. 1998. P. 233–238. https://doi.org/10.1109/VISUAL.1998.745713
15. *Brawley Z., Tatarchuk N.* Parallax Occlusion Mapping: Self-Shadowing, Perspective-Correct Bump Mapping Using Reverse Height Map Tracing // ShaderX3: Advanced Rendering with DirectX and OpenGL. 1st. ed. Charles River Media. 2004. P. 135–154.
16. *Tatarchuk N.* Dynamic Parallax Occlusion Mapping with Approximate Soft Shadows // ACM Symposium on Interactive 3D Graphics and Games (I3D '06).

2006. P. 63–69.
https://doi.org/10.1145/1111411.1111423.

17. *Policarpo F., Oliveira M.M., Comba J.L.D.* Real-Time Relief Mapping on Arbitrary Polygonal Surfaces // I3D '05: proceedings of the 2005 Symposium on Interactive 3D Graphics and Games. 2005. P. 155–162. https://doi.org/10.1145/1053427.1053453.

18. *Ammann L., Génevaux O., Dischler J.-M.* Hybrid Rendering of Dynamic Heightfields using Ray-Casting and Mesh Rasterization // GI '10: proceedings of Graphics Interface Conference 2010, Canadian Information Processing Society. 2010. P. 161–168. https://dl.acm.org/doi/10.5555/1839214.1839243.

19. *Policarpo F., Oliveira M.M.* Relaxed Cone Stepping for Relief Mapping // GPU Gems 3. Addison-Wesley Professional. 2007. P. 409–428. https://developer.nvidia.com/gpugems/gpugems3/part-iii-rendering/chapter-18-relaxed-cone-stepping-relief-mapping.

20. *Baboud L., Eisemann E., Seidel H.-P.* Precomputed Safety Shapes for Efficient and Accurate Height-Field Rendering // IEEE Transactions on Visualization and Computer Graphics. 2012. V. 18. № 11. P. 1811–1823. https://doi.org/10.1109/TVCG.2011.281

21. *Tevs A., Ihrke I., Seidel H.-P.* Maximum Mipmaps for Fast, Accurate, and Scalable Dynamic Height Field Rendering // Proceedings of the 2008 Symposium on Interactive 3D Graphics and Games (I3D'08). New York. 2008. P. 183–190. https://doi.org/10.1145/1342250.1342279.

22. *Dick C., Krüger J.H., Westermann R.* GPU Ray-Casting for Scalable Terrain Rendering // Eurographics '09, 2009. P. 43–50. https://doi.org/10.2312/ega.20091007

23. *Lee E.-S., Lee J.-H., Shin B.-S.* A bimodal empty space skipping of ray casting for terrain data // The Journal of Supercomputing. 2016. V. 72. № 7. P. 2579–2593. https://doi.org/10.1007/s11227-015-1522-9

24. *Aslandere T., Flatken M., Gerndt A.* A Real-Time Physically Based Algorithm for Hard Shadows on Dynamic Height-Fields // Proceedings of 12. Workshop der GI-Fachgruppe on Virtuelle und Erweiterte Realität, Aachen Verlag, Bonn. 2015. P. 101–112. https://elib.dlr.de/101497/.

25. *Dübel S., Middendorf L., Haubelt C., Schumann H.* A Flexible Architecture for Ray Tracing Terrain Heightfields // Proceedings of the International Summerschool on Visual Computing, Rostock, Germany, 2015. P. 3–22.

26. *Silvestre A., Pereira J., Costa V.* A Real-Time Terrain Ray-Tracing Engine // 2018 International Conference on Graphics and Interaction (ICGI), 2018. P. 1–8. https://doi.org/10.1109/ITCGI.2018.8602735.

27. *Rusch M., Bickford N., Subtil N.* Introduction to Vulkan Ray Tracing // Ray Tracing Gems II. NVIDIA. 2021. P. 213–255. https://doi.org/10.1007/978-1-4842-7185-8_16

28. *Thonat T., Beaune F., Sun X., Carr N., Boubekeur T.* Tessellation-free displacement mapping for ray tracing // ACM Transactions on Graphics. 2021. Vol. 40. No. 6. Article 282. P. 1–16. https://doi.org/10.1145/3478513.3480535.

29. *Silva V., Novello T., Lopes H., Velho L.* Real-Time Rendering of Complex Fractals // Ray Tracing Gems II. NVIDIA. 2021. P. 529–544. https://doi.org/10.1007/978-1-4842-7185-8_33

30. *Brüll F.* Fast Transparency and Billboard Ray Tracing with RTX Hardware // Master thesis. Clausthal University of Technology. 2020. 81 p. https://doi.org/10.13140/RG.2.2.14692.19842.

31. NVIDIA Vulkan Ray Tracing Tutorials. Intersection Shader – Tutorial. 2020–2022. https://github.com/nvpro-samples/vk_raytracing_tutorial_KHR/tree/master/ray_tracing_intersection.

32. Vulkan 1.3.238 – A Specification (with all registered Vulkan extensions) // The Khronos Vulkan Working Group. 2022. https://www.khronos.org/registry/vulkan/specs/1.3-extensions/pdf/vkspec.pdf.

33. *Majercik A., Crassin C., Shirley P., McGuire M.* A Ray-Box Intersection Algorithm and Efficient Dynamic Voxel Rendering // Journal of Computer Graphics Techniques. 2018. V. 7. № 3. P. 66–82. ISSN 2331-7418. https://www.jcgt.org/published/0007/03/04/paper-lowres.pdf.

34. NVIDIA Vulkan Ray Tracing Tutorials. https://github.com/nvpro-samples/vk_raytracing_tutorial_NV/tree/master/ray_tracing_intersection/shaders/raytrace.rint.

35. *Reshetov A.* Cool Patches: A Geometric Approach to Ray/Bilinear Patch Intersections // Ray Tracing Gems. 2019. P. 95–109.

36. *Amanatides J., Woo A.* A Fast Voxel Traversal Algorithm for Ray Tracing // Eurographics '87: Proceedings of the 8th European Computer Graphics Conference and Exhibition, Amsterdam, 1987. P. 3–10.

37. Large Geometric Models Archive // Georgia Institute of Technology. https://www.cc.gatech.edu/projects/large_models/.

38. Puget Sound test map // The University of Chicago. https://www.classes.cs.uchicago.edu/archive/2015/fall/23700-1/final-project/puget-sound/index.html.

*ПРОГРАММИРОВАНИЕ, 2023, № 3, с. 65–70*

## АНАЛИЗ ДАННЫХ

*УДК 519.6+004.891.3+616-018*

# АУГМЕНТАЦИЯ ОБУЧАЮЩЕЙ ВЫБОРКИ ГИСТОЛОГИЧЕСКИХ ИЗОБРАЖЕНИЙ АДВЕРСАТИВНЫМИ АТАКАМИ

**© 2023 г. Н. Д. Локшин[a,*] (ORCID: 0000-0001-7777-7035),**
**А. В. Хвостиков[a,**] (ORCID: 0000-0002-4217-7141),**
**А. С. Крылов[a,***] (ORCID: 0000-0001-9910-4501)**

*[a] Московский государственный университет имени М.В. Ломоносова, 119991 Москва, ГСП-1, Ленинские горы, д. 1, Россия*

**E-mail: lockshin1999@mail.ru*

***E-mail: khvostikov@cs.msu.ru*

****E-mail: kryl@cs.msu.ru*

Поступила в редакцию 09.01.2023 г.
После доработки 15.01.2023 г.
Принята к публикации 20.01.2023 г.

В работе рассматривается задача аугментации выборки гистологических изображений адверсативными атаками для повышения устойчивости нейросетевых классификаторов, обученных на аугментированной выборке, к адверсативным атакам. В последние годы нейросетевые методы стремительно развивались, новые нейросетевые методы показывают впечатляющие результаты, однако они подвергаются так называемым адверсативным атакам – то есть совершают неверные предсказания на входах, получающихся в результате наложения на изображение малого шума. Из-за этого надежность нейросетевых методов до сих пор является актуальной областью изучения. В этой статье мы представляем и сравниваем между собой различные методы аугментации обучающей выборки, позволяющие повысить устойчивость нейросетевых классификаторов гистологических изображений к адверсативным атакам. Для этого мы предлагаем добавлять в обучающую выборку адверсативные атаки, полученные несколькими актуальными методами.

**DOI:** 10.31857/S0132347423030020, **EDN:** DEEBYB

## 1. ВВЕДЕНИЕ

Некоторые модели машинного обучения, в частности нейронные сети, часто подвергаются *адверсативным атакам* – то есть неверно классифицируют входы, получающиеся в результате наложения на изображение малого шума [1–4] (рис. 1). На данный момент имеется большое количество способов генерации таких атак, а также и методов защиты от них [1, 7, 8]. Большинство способов защиты от адверсативных атак либо предусматривают изменение структуры исходной модели предсказания, например, *защитная дистилляция* нейросетей [8], либо строят предположения о возможных атаках.

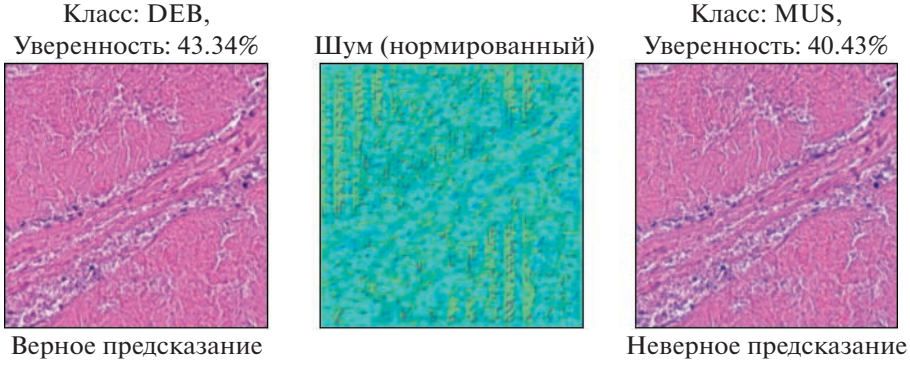


**Рис. 1.** Пример адверсативной атаки на модель ResNet50 [5] на изображение из набора данных NCT-CRC-HE-100K [6].

Самым эффективным методом генерации адверсативных возмущений на данный момент является метод *AdvGAN* [9]. Основные преимущества AdvGAN перед другими способами генерации адверсативных атак заключаются в более низкой степени искажения исходных изображений при совершении атаки и более высокой вероятности совершить атаку на изображения успешно. Данный метод заключается в *генеративно-состязательном* [10] обучении нейросети, которая после этапа обучения способна генерировать адверсативные возмущения по переданным на вход изображениям.

В данной работе на основе [1, 3, 9, 11, 12] предлагается метод повышения устойчивости нейросетевых классификаторов гистологических изображений к адверсативным атакам.

## 2. АДВЕРСАТИВНЫЕ АТАКИ

В данной работе рассматривается 5 алгоритмов генерации адверсативных атак: *Fast Gradient Sign Method* (FGSM) [1], *Carlini and Wagner* (C&W) [3], *AdvGan* [9], *Projected Gradient Descent* (PGD) [11] и *Decoupled Direction and Norm* (DDN) [13].

Алгоритмы генерации адверсативных атак делятся на White-Box и Black-Box. White-Box-алгоритмы принимают на вход, помимо изображения, по которому необходимо построить адверсативную атаку, обученную нейросеть. Black-Box-алгоритмы принимают на вход только изображение. Все алгоритмы, рассматриваемые в данной статье, являются White-Box.

**Fast Gradient Sign Method**

Имея на входе изображение $I$ с меткой класса $y$, функцию потерь $J$, обученный классификатор $C$, размер шага $\epsilon$, адверсативное изображение вычисляется следующим образом:

$$\tilde{I} = I + \epsilon \cdot \operatorname{sign}\nabla_I J(C, I, y) \tag{2.1}$$

В данном алгоритме число $\epsilon$ является гиперпараметром. Как правило, $\epsilon$ выбирается достаточно малым, чтобы накладываемое на изображение возмущение было слабо заметным. В данной работе $\epsilon = 0.07$.

**Projected Gradient Descent**

Имея на входе изображение $I$ с меткой класса $y$, функцию потерь $J$, обученный классификатор $C$, размер шага $\alpha$ и количество итераций $K$, для $k = 1 \ldots K$ совершается вычисление:

$$I_{k+1} = \prod_{I+S}(I_k + \alpha \cdot \operatorname{sign}(\nabla_{I_k} J(C, I_k, c))) \tag{2.2}$$

$I_0$ принимается равным $I$. На выходе алгоритма имеем адверсативное изображение $I_K$. В данной работе $\alpha = 3 \times 10^{-3}$, количество шагов равно 100, $S$ – $l_\infty$-сфера вокруг изображения $I$ с радиусом 0.01.

**Decoupled Direction and Norm**

Имея на входе изображение $I$ с меткой класса $y$, функцию потерь $J$, обученный классификатор $C$, размер шага $\alpha$, шаг обновления радиуса $\gamma$ и количество итераций $K$, для $k = 1 \ldots K$ совершаются следующие действия:

1. Вычисление градиента:

$$g \leftarrow \nabla J_{\tilde{I}_{k-1}}(C, \tilde{I}_{k-1}, y) \tag{2.3}$$

2. Нормирование и умножение на размер шага:

$$g \leftarrow \alpha \frac{g}{\|g\|_2} \tag{2.4}$$

3. Обновление адверсативного возмущения:

$$\sigma_k = \sigma_{k-1} + g \tag{2.5}$$

Возмущение $\sigma_0$ принимается равным 0.

4. Проекция адверсативного изображения $\tilde{I}_{k-1} + \sigma_k$ на $\epsilon_k$-сферу вокруг исходного изображения $I$. $\epsilon_k$ берется равным $(1+\gamma)\epsilon_{k-1}$, если $\tilde{I}_{k-1}$ успешно нарушает работу $C$, либо $(1-\gamma)\epsilon_{k-1}$, если иначе:

$$\tilde{I}_k = \tilde{I}_{k-1} + \epsilon_k \frac{\sigma_k}{\|\sigma_k\|_2} \tag{2.6}$$

5. Клиппирование значений, выходящих за допустимые границы: $\tilde{I}_k$: $\tilde{I}_k \leftarrow clip(\tilde{I}_k, 0, 1)$.

В конце итераций на выход подается такое $\tilde{I}_k$, которое успешно нарушает работу $C$ и имеет наименьшую $l_2$-норму.

**Carlini and Wagner**

Имея на входе изображение $I$ с меткой класса $y$ и обученный нейросетевой классификатор $C$, адверсативная атака выполняется путем решения задачи оптимизации:

$$\begin{gathered}\text{minimize } \|\sigma\|_2 - \epsilon \cdot f(I + \sigma) \\ \text{suchthat } I + \sigma \in [0,1]^n,\end{gathered} \tag{2.7}$$

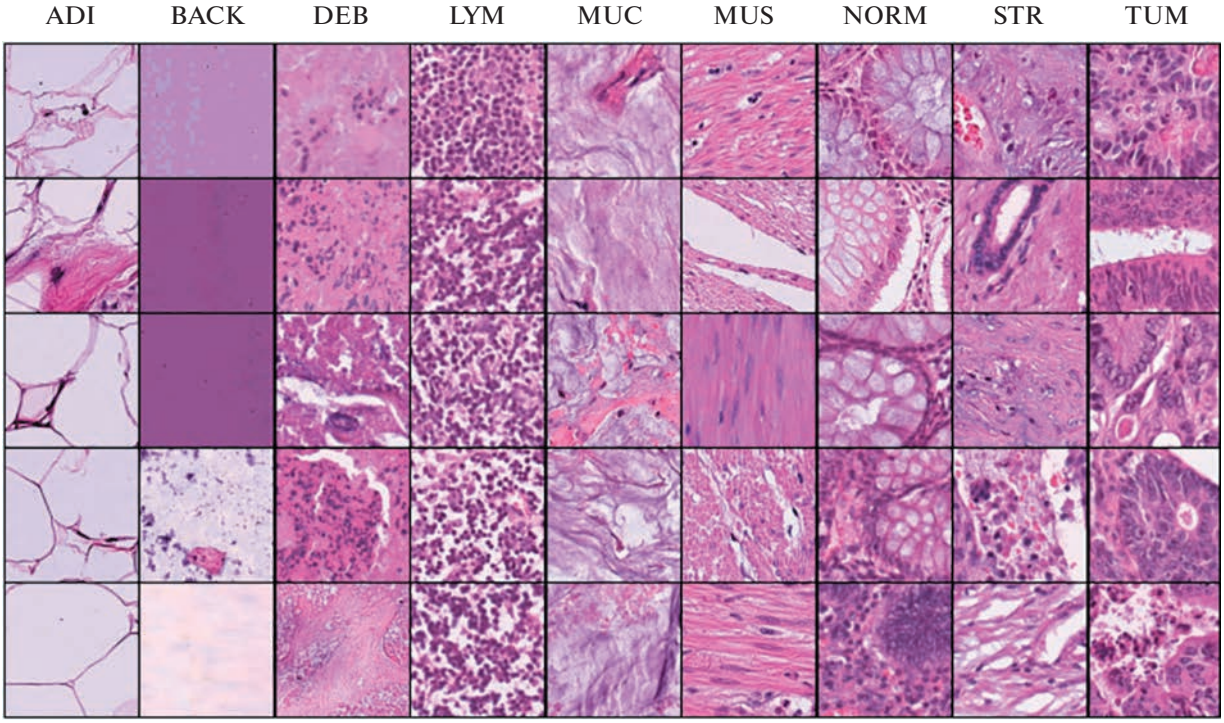


**Рис. 2.** Примеры изображений из набора NCT-CRC-HE-100K [6, 14].

где функция $f$ подобрана таким образом, что $C(I+\sigma) = c$ тогда и только тогда, когда $f(x+\sigma) \le 0$. В данном эксперименте,

$$f(I+\sigma) = \max\left(\max_{i \neq y}(Z(I+\sigma)_i\right) - Z(I)_y, 0), \quad (2.8)$$

где $Z(I)$ – ненормированный выход классификатора $C$ с принятием на вход $I$. Данная функция признана самой эффективной из возможных в оригинальной статье. Оптимальное значение константы $\epsilon$ таково что оно наименьшее, для которого $f(I+\sigma) \le 0$. Чтобы найти $\epsilon$, совершается 20 шагов бинарного поиска, где на каждом шаге находится решение $I+\sigma$ решением задачи оптимизации 2.7 методом градиентного спуска. В данной работе количество шагов градиентного спуска равно 100 с шагом 0.01.

**AdvGan**

Данный алгоритм строит адверсативную атаку путем обучения генеративно-состязательной сети (GAN). Во время обучения генератор $G(z)$ минимизирует следующий функционал:

$$l = \gamma l_{nce} + \theta l_{GAN} + \zeta l_{threshold}, \quad (2.9)$$

где $l_{nce}$ – это обратный функционал кросс-энтропии:

$$l_{nce} = \frac{1}{l_{ce}}, \quad (2.10)$$

$$l_{ce} = -\frac{1}{N}\sum_{n=1}^{N} \log \frac{\exp(x_{n,y_n})}{\sum_{c=1}^{N_c} \exp(x_{n,c})}, \quad (2.11)$$

$l_{GAN}$ – это функционал, оптимизируемый в рамках генеративно-состязательной сети:

$$l_{GAN} = \mathbb{E}_{I \sim p_{data}(I)} \log D(I) + \\ + \mathbb{E}_{I \sim p_{adversarial}(I)} \log(1 - D(I + aG(z))), \quad (2.12)$$

где $G(z)$ имеет на выходе изображение, $D(I)$ имеет на выходе вероятность того, что изображение $I$ – настоящее. $l_{threshold}$ задан следующим образом:

$$l_{threshold} = \sum_{x=1}^{W}\sum_{y=1}^{H}\sum_{z=1}^{C} \max\left(\left|I_{x,y,z}\right| - thr, 0\right)^2, \quad (2.13)$$

где $I_{x,y,z}$ – значения пикселя изображения $I$ в координатах $(x, y, z)$. В данном эксперименте значение $thr$ равно 0.25. Коэффициенты $\gamma$, $\theta$ и $\zeta$ были эмпирически подобраны и равны 10, 1, 1 соответственно.

**Таблица 1.** Результаты тестирования классификаторов, имеющих структуру ResNet50. В левой колонке приведены обучающие выборки: 18K – исходная обучающая выборка, 36K – исходная обучающая выборка, аугментированная 18000 синтетическими изображениями, сгенерированными StyleGan2, 72K – набор 36K, аугментированный $AdvGan_{a=1}$, 216K – набор 36K, аугментированный $AdvGan_{a=1}$, FGSM, DDN, PGD, C&W. Названия алгоритмов генерации адверсативных атак в столбцах обозначают адверсативную атаку, примененную к каждому изображению в тестовой выборке

| Обуч. выборка | Точность на тестовой выборке с применением различных видов атак | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | No attack | FGSM | DDN | PGD | C&W | $AdvGan_{a=0.5}$ | $AdvGan_{a=1}$ | $AdvGan_{a=2}$ |
| 18K | 99.49 | 19.43 | 0 | 0.01 | 0 | 96.22 | 69.57 | 44.60 |
| 36K | 99.37 | 36.69 | 97.76 | 1.25 | 95.60 | 95.99 | 72.73 | 46.64 |
| 72K | 99.23 | 53.24 | 98.30 | 8.85 | 97.73 | 99.36 | 99.32 | 95.36 |
| 216K | 98.89 | 84.16 | 98.61 | 72.33 | 98.46 | 98.78 | 98.47 | 94.82 |

## 3. ПОСТАНОВКА ЗАДАЧИ

В качестве входных данных имеется 22500 размеченных изображений – подмножество набора *NCT-CRC-HE-100K* [6, 4]. Изображения являются непересекающимися участками больших гистологических изображений, содержащих злокачественные новообразования толстого кишечника а также здоровые ткани, окрашенные гематоксилин-эозином. Подмножество набора данных выбрано для уменьшения временных затрат на эксперименты и обучение нейросетевых методов. Изображения имеют размерности $224 \times 224 \times 3$. Данные равномерно размечены на 9 классов тканей: *adipose (ADI), background (BACK), debris (DEB), lymphocytes (LYM), mucus (MUC), smooth muscle (MUS), normal colon mucosa (NORM), cancer-associated stroma (STR), colorectal adenocarcinoma epithelium (TUM)*. Примеры изображений для каждого класса приведены на рис. 2.

Приведенный набор разбит на тренировочную выборку, содержащую 18000 изображений, и тестовую, содержащую 4500 изображений. Требуется исследовать и реализовать методы аугментации данного набора данных различными адверсативными атаками и сравнить качество различных нейросетевых классификаторов на скрытых выборках, также аугментированных адверсативными атаками. Термин “скрытая” выборка означает, что изображения из этой выборки не принадлежат обучающей выборке.

## 4. ПРЕДЛАГАЕМЫЙ МЕТОД

В данной работе предлагается рассматривать несколько наборов данных для обучения, аугментированных различными наборами адверсативных атак, а именно:

• Исходная обучающая выборка, состоящая из 18000 гистологических изображений;

• Обучающая выборка, состоящая из 18000 исходных гистологических изображений, и 18000 синтетических изображений, сгенерированных хорошо известным алгоритмом StyleGAN2 – всего 36000 изображений;

• 36000 изображений из предыдущего пункта, аугментированных 36000 адверсативными атаками AdvGan (т.е применение AdvGan к каждому из 36000 изображений в обучающей выборке с последующим добавлением результата в обучающую выборку) – всего 72000 изображений;

• 216000 изображений в результате аугментации алгоритмами DDN, C&W, FGSM, PGD и AdvGan. Каждый из перечисленных алгоритмов был применен к каждому из 36000 изображений с последующим добавлением результата в обучающую выборку.

На каждой обучающей выборке был обучен нейросетевой классификатор имеющий архитектуру *ResNet50* [5], в итоговом сравнении участвует 4 нейросетевых классификатора, имеющих оди-

наковую структуру, но обученных на разных наборах данных. Для тестирования используются следующие скрытые наборы данных:

• Исходная тестовая выборка, состоящая из 4500 гистологических изображений;

• Тестовая выборка, состоящая из 4500 исходных гистологических изображений с применением FGSM [1] с параметром $\epsilon = 0.1$;

• Тестовая выборка, состоящая из 4500 исходных гистологических изображений с применением C&W [3];

• Тестовая выборка, состоящая из 4500 исходных гистологических изображений с применением PGD [11];

• Тестовая выборка, состоящая из 4500 исходных гистологических изображений с применением DDN [13];

• Тестовая выборка, состоящая из 4500 исходных гистологических изображений с применением AdvGan с параметром $a = 0.5$;

• Тестовая выборка, состоящая из 4500 исходных гистологических изображений с применением AdvGan с параметром $a = 1$;

• Тестовая выборка, состоящая из 4500 исходных гистологических изображений с применением AdvGan с параметром $a = 2$.

## 5. АНАЛИЗ ПОЛУЧЕННЫХ РЕЗУЛЬТАТОВ

Результаты экспериментов приведены в табл. 1. Предложенный метод аугментации данных существенно повысил устойчивость нейросетевого классификатора ResNet50 к адверсативным атакам. Тем не менее, даже при увеличении объема обучающей выборки в 12 раз вышеприведенным методом точность классификатора на скрытой выборке без применения к ней адверсативных атак изменяется несущественно.

## 6. ЗАКЛЮЧЕНИЕ

В данной работе был предложен и реализован метод аугментации обучающей выборки адверсативными атаками для обучения нейросетевых классификаторов. В случае, когда архитектуры классификатора, использовавшегося для генерации адверсативных атак, и тестового классификатора совпадали, было продемонстрировано повышение устойчивости против ряда адверсативных атак. Для обучения моделей использовался вычислительный кластер с 4x Nvidia RTX A6000 факультета вычислительной математики и кибернетики Московского государственного университета имени М.В. Ломоносова.

## БЛАГОДАРНОСТИ

Исследования выполнены при финансовой поддержке Российского научного фонда в рамках научного проекта 22-21-00081.

## СПИСОК ЛИТЕРАТУРЫ

1. *Goodfellow I.J., Shlens J., Szegedy Ch.* Explaining and harnessing adversarial examples // arXiv preprint arXiv:1412.6572, 2014
2. *Su Jiawei, Vargas Danilo Vasconcellos, Sakurai Kouichi.* One pixel attack for fooling deep neural networks // IEEE Transactions on Evolutionary Computation. 2019. № 5. С. 828–841.
3. *Carlini N., Wagner D.* Towards evaluating the robustness of neural networks // IEEE Symposium on Security and Privacy (SP). 2017. P. 39–57.
4. *Moosavi-Dezfooli, Seyed-Mohsen, Fawzi Alhussein, Frossard Pascal.* Deepfool: a simple and accurate method to fool deep neural networks // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016. P. 2574–2582.
5. *He Kaiming, Zhang Xiangyu, Ren Shaoqing, Sun Jian.* Deep residual Learning for Image Recognition // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016. P. 770–778.
6. *Kather J.N., Halama N., Marx A.* 100000 histological images of human colorectal cancer and healthy tissue // Zenodo10, 2018.
7. *Liang Bin, Li Hongcheng, Su Miaoqiang, Li Xirong, Shi Wenchang, Wang Xiaofeng.* Detecting adversarial image examples in deep neural networks with adaptive noise reduction // IEEE Transactions on Dependable and Secure Computing. 2018. № 1. P. 72–85.
8. *Papernot N., McDaniel P., Wu Xi, Jha Somesh, Swami Ananthram.* Distillation as a defense to adversarial perturbations against deep neural networks // IEEE Symposium on Security and Privacy (SP), 2016. P. 582–597.
9. *Xiao Chaowei, Li Bo, Zhu Jun-Yan, He Warren, Liu Mingyan, Song Dawn.* Generating adversarial examples with adversarial networks // arXiv preprint arXiv:1801.02610, 2018.
10. *Goodfellow I., Pouget-Abadie J., Mirza Mehdi, Xu Bing, Warde-Farley D., Ozair Sherjil, Courville A., Bengio Yoshua.* Generative adversarial nets // Advances in Neural Information Processing Systems, 2014.
11. *Madry A., Makelov A., Schmidt L., Tsipras D., Vladu A.* Towards deep learning models resistant to adversarial attacks // arXiv preprint arXiv:1706.06083, 2017.

12. *Karras Tero, Laine Samuli, Aittala Miika, Hellsten Janne, Lehtinen Jaakko, Aila Timo.* Analyzing and improving the image quality of stylegan // Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020. P. 8110–8119.

13. *Rony J., Hafemann L.G., Oliveira L.S., Ayed Ismail Ben, Sabourin R., Granger E.* Decoupling direction and norm for efficient gradient-based l2 adversarial attacks and defenses // Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019. P. 4322–4330.

14. *Khvostikov A., Krylov A., Mikhailov I., Malkov P., Danilova N.* Tissue Type Recognition in Whole Slide Histological Images. 2021.