

Санкт-Петербургский государственный университет

Кафедра системного программирования

Группа 23.Б08-мм

Распознавание направления дороги вдали по видео в ADAS

Смола Артём Михайлович

Отчёт по учебной практике
в форме «Производственное задание»

Научный руководитель:
старший преподаватель кафедры системного программирования, к.т.н., Ю. В. Литвинов

Консультант:
инженер-программист АО «Кама», М. А. Житнухина

Санкт-Петербург
2025

Оглавление

Введение	3
1. Постановка задачи	4
2. Обзор	5
2.1. Используемые метрики	5
2.2. Реализованный ранее прототип	6
2.3. Оптический поток	8
3. Описание решения	12
3.1. Автоматизированное выделение области интереса	12
3.2. Итоговый алгоритм	14
4. Эксперимент	16
4.1. Исследовательские вопросы	16
4.2. Метрики	16
4.3. Результаты	16
Заключение	19
Список литературы	20

Введение

Комплекс систем помощи водителю ADAS (англ. Advanced Driver Assistance Systems) делает управление автомобилем более безопасным и комфортным. Для надежной работы систем ADAS, связанных с автоматическим регулированием скорости (адаптивный круиз-контроль) и удержанием в полосе, необходимо получать информацию о направлении дороги на как можно большем расстоянии.

Существующие решения часто игнорируют дальнюю часть дороги, уделяя внимание исключительно ближайшим 30 метрам от автомобиля. В работе за прошлый семестр [4] был реализован прототип системы распознавания направления дороги вдаль (на расстоянии не менее 40 метров). Однако в работе не была реализована важнейшая для применения в реальных условиях система автоматизированного выделения области дороги вдаль. Фиксированная обрабатываемая область кадра ограничивала применимость системы в момент поворота автомобиля, поскольку часть дорожной разметки оказывалась вне анализируемой зоны.

В данной работе предлагается использовать информацию о смещении точек на видео, полученную с помощью вычисления оптического потока, для смещения области интереса между кадрами видео.

1. Постановка задачи

Целью данной работы является улучшение системы распознавания направления дороги вдали. Для её выполнения были поставлены следующие задачи.

1. Провести обзор реализованной ранее системы распознавания дороги вдали и технологий, которые будут использоваться для реализации алгоритма автоматизированного выделения области интереса.
2. Реализовать алгоритм автоматизированного выделения области интереса.
3. Внедрить алгоритм автоматизированного выделения области интереса в систему распознавания направления дороги вдали.
4. Провести эксперименты для оценки работы алгоритма.

2. Обзор

В данном разделе проводится обзор реализованного в рамках предыдущей работы прототипа системы распознавания направления дороги вдали, а также рассматриваются технологии, которые могут быть использованы для реализации системы автоматизированного выделения области интереса.

2.1. Используемые метрики

Тестировать качество работы алгоритма предлагается по метрикам IoU (англ. Intersection over Union), Precision и FPS (англ. Frames Per Second). Ниже приведены формулы для метрик IoU и Precision.

$$\text{IoU} = \frac{A \cap B}{A \cup B},$$

где A — множество пикселей, действительно относящихся к разметке, B — множество пикселей, определённых алгоритмом, как относящиеся к разметке.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}},$$

где TP — количество верно распознанных пикселей, FP — количество пикселей, ошибочно отнесённых к разметке.

Метрика Precision позволяет оценить количество ложных срабатываний: низкое значение по данной метрике будет означать, что в предсказанной области оказалось большое количество не относящихся к разметке пикселей. Метрика IoU позволяет оценить, насколько в целом предсказанная область близка к достоверной.

В предыдущей работе использовалась также метрика Accuracy:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}},$$

где TN — количество пикселей, верно не отнесённых к разметке, FN — количество пикселей, ошибочно не отнесённых к разметке.

Но стоит заметить, что размер дорожной разметки мал относительно размера всего изображения, из-за чего на задаче обнаружения дорожной разметки удаётся добиваться результата, близкого к 90 процентам даже при отключении распознавания. По этой причине в данной работе тестирование по данной метрике вестись не будет.

2.2. Реализованный ранее прототип

Реализованный алгоритм распознавания направления дороги вдали [4] состоит из трёх ключевых этапов.

1. Выделение области интереса, соответствующей области дороги вдали.
2. Обнаружение линий разметки.
3. Создание вектора пикселей, относящихся к дорожной разметке.

Также был реализован вариант алгоритма с предварительным увеличением изображения при помощи нейросети перед обнаружением разметки.

2.2.1. Используемые технологии

Для реализации прототипа системы использовались следующие технологии:

- язык программирования C++,
- библиотека OpenCV,
- модель TwinLiteNet для распознавания дорожной разметки,
- модель ESPCN для увеличения изображения.

2.2.2. Особенности реализации

Обнаружение разметки. Модель TwinLiteNet на вход принимает изображение размера 640×360 .

Выделение области интереса. Область интереса настраивается вручную на изображениях датасета ApolloScapes. Размер области интереса — $640k \times 360k$, где k зависит от размера обрабатываемого изображения. При необходимости изображение ROI пропорционально увеличивается или уменьшается перед передачей в модель распознавания разметки.

2.2.3. Тестирование

При сравнении в базовом решении модель применялась ко всему изображению (то есть не только к области интереса).

В ходе тестирования на 16560 изображениях датасета ApolloScapes¹ были получены результаты, приведённые в таблице 1,

Алгоритм	IoU		Accuracy		FPS	
	М	σ	М	σ	М	σ
Предлагаемое решение	19,6%	17,9%	96,8%	2,1%	13,8	0,9
Версия с ESPCN	19,6%	17,9%	96,8%	2,1%	3,0	0,1
Базовое решение	7,7%	14,1%	90,5%	6,5%	16,4	0,9

Таблица 1: Результаты тестирования на датасете ApolloScapes

где М — математическое ожидание, σ — среднеквадратическое отклонение.

Из-за высокого разброса значения IoU, вызванного низкой эффективностью алгоритма в моменты поворота автомобиля, было принято решение также провести тестирование на части датасета (2551 изображение), на которой отсутствуют моменты поворота автомобиля. Были получены результаты, представленные в таблице 2.

Тестирование показало, что предлагаемое решение показывает лучшие результаты по метрикам IoU и Ассигасу при незначительной потере по метрике FPS. Также было выяснено, что для работы в режиме реального времени версии алгоритма с увеличением изображения при помощи модели ESPCN требуется более мощное оборудование.

¹URL: https://apolloscape.auto/lane_segmentation.html (дата обращения: 15.05.2025).

Алгоритм	IoU		Accuracy	
	М	σ	М	σ
Предлагаемое решение	18,3%	8,3%	95,9%	2,3%
Версия с ESPCN	18,4%	8,3%	95,9%	2,3%
Базовое решение	6,3%	4,3%	87,0%	7.0%

Таблица 2: Результаты замеров метрик на части датасета

2.3. Оптический поток

По характеру движения точек на видео можно определить, поворачивает ли автомобиль, и если да, то с какой скоростью. Так, если все отслеживаемые точки, находящиеся в разных частях кадра, одновременно движутся по горизонтали в одну и ту же сторону, можно предположить, что автомобиль поворачивает. Одним из способов определения такого движения точек является вычисление оптического потока.

Оптическим потоком называется векторное поле, отражающее движение объектов изображения между кадрами, которое определяется как проекция скоростей точек в пространстве на плоскость изображения (рисунок 1).

При вычислении оптического потока делается допущение о том, что при движении объекта интенсивность его пикселей в течение малого промежутка времени остается постоянной. Из этого предположения получим следующее соотношение

$$I(x, y, t) \approx I(x + \delta x, y + \delta y, t + \delta t),$$

где I — функция, задающая интенсивность пикселя (x, y) в момент t .

Разложим $I(x, y, t)$ в ряд Тейлора и воспользуемся полученным ранее соотношением, получим

$$I(x, y, t) = I(x, y, t) + I_x \delta x + I_y \delta y + I_t \delta t + O(\delta x^2, \delta y^2, \delta t^2),$$

где $I_x = \frac{\partial I}{\partial x}$, $I_y = \frac{\partial I}{\partial y}$, $I_t = \frac{\partial I}{\partial t}$.

Далее вычтем из обеих частей $I(x, y, t)$, поделим на δt и полагая, что $O(\delta x^2, \delta y^2, \delta t^2)$ можно пренебречь, получим следующее уравнение

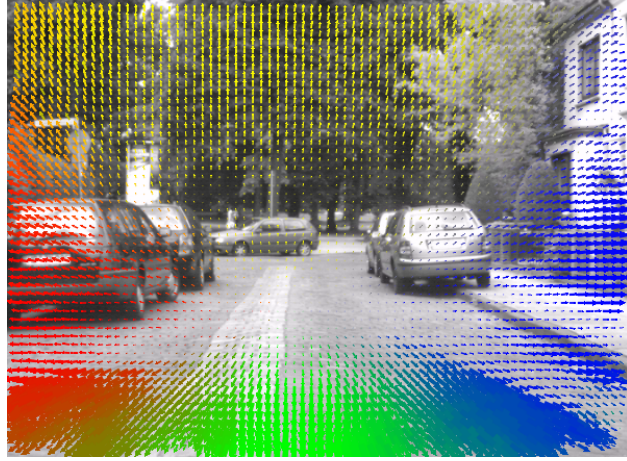


Рис. 1: Оптический поток на изображении с камеры на автомобиле (Источник URL: https://help.commonvisionblox.com/NextGen/14.0/md_theory_of_operation_tools__optical_flow.html (дата обращения: 02.05.2025).)

$$I_x u + I_y v + I_t = 0,$$

где $u = \frac{\delta x}{\delta t}$, $v = \frac{\delta y}{\delta t}$ — компоненты вектора оптического потока в точке (x, y) .

Имеем две неизвестные u и v , поэтому нам не достаточно одного уравнения. Для продолжения решения могут применяться различные методы, например, алгоритм Хорна-Шунка [2] или решения на основе нейронных сетей [1]. Однако для работы в режиме реального времени лучше всего подходит метод Лукаса–Канаде [3].

2.3.1. Метод Лукаса–Канаде

Метод Лукаса–Канаде основан на предположении о том, что точки в некоторой окрестности $U_n(x_0, y_0)$ размера $n \times n$, $n > 1$, где (x_0, y_0) — рассматриваемая точка, имеют одинаковое движение. Рассмотрев точки из данной окрестности, мы получим переопределённую систему уравнений (количество уравнений превосходит количество неизвестных)

$$I_x^{(i)} u + I_y^{(i)} v = -I_t^{(i)},$$

где $1 \leq i \leq n^2$ — индекс точки из окрестности $U_n(x_0, y_0)$.

Данную систему можно решить методом наименьших квадратов. Будем минимизировать взвешенную сумму квадратов.

$$\min_{(u,v)} \sum_{i=1}^{n^2} m_i \left(I_x^{(i)} u + I_y^{(i)} v + I_t^{(i)} \right)^2,$$

где m_i — веса, распределённые по Гауссу, которые задаются как

$$m_i = \exp \left(-\frac{(x_i - x_0)^2 + (y_i - y_0)^2}{2\sigma^2} \right),$$

где σ — параметр, который подбирается экспериментально.

Дифференцируя по u и v , получим систему, из которой сможем найти u и v

$$\begin{cases} \sum m_i I_x^2 \cdot u + \sum m_i I_x I_y \cdot v = - \sum m_i I_x I_t, \\ \sum m_i I_x I_y \cdot u + \sum m_i I_y^2 \cdot v = - \sum m_i I_y I_t. \end{cases}$$

Реализовав данные шаги, мы получим метод, который может определить только небольшие сдвиги. Для определения больших смещений применяется алгоритм, модифицируемый методом пирамиды.

Метод пирамиды. Данный метод состоит из k уровней (от 0 до $k - 1$), на уровне i ($0 \leq i \leq k - 1$) берётся изображение, сжатое в 2^i раз относительно своего начального размера.

Начинаем с самого высокого уровня $k - 1$ (наименьший размер изображения). Применяем к сжатому изображению приведенный выше алгоритм Лукаса–Канаде и получаем оценку смещения (u_{k-1}, v_{k-1}) . Заметим, большие смещения уменьшаются, и при достаточном k их сможет определить алгоритм Лукаса–Канаде.

Для уровня j ($0 \leq j \leq k - 2$) умножаем смещение, полученное на уровне $j - 1$, на 2, получаем вектор (u'_{j-1}, v'_{j-1}) . Смещаем изображение на (u'_{j-1}, v'_{j-1}) и для смещенного изображения применяем алгоритм Лукаса–Канаде, получая остаточное смещение $(\delta u_j, \delta v_j)$. Тогда итоговое смещение для данного уровня найдем по формуле

$$(u_j, v_j) = (u'_{j-1}, v'_{j-1}) + (\delta u_j, \delta v_j).$$

Результатом работы алгоритма будет являться смещение (u_0, v_0) .

2.3.2. Реализация алгоритма Лукаса–Канаде в OpenCV

Библиотека OpenCV содержит реализацию алгоритма Лукаса–Канаде в виде функции `cv::calcOpticalFlowPyrLK()`.

Данная функция принимает на вход текущий кадр и предыдущий, а также вектор точек с предшествующего кадра, которые необходимо отследить. Результат работы функции записывается в два вектора, один из которых является вектором булевых значений и отражает статус отслеживания точки (соответствующее значение — `true`, значит, точку удалось отследить, иначе — не удалось), а второй является вектором точек и соответствует новым положениям отслеживаемых точек с предыдущего кадра.

Также на вход передаются параметры, используемые в алгоритме. Функция принимает на вход размер окрестности U_n (из описания метода Лукаса–Канаде) и высоту пирамиды.

Особенностью реализации алгоритма является итеративное уточнение оценки смещения на каждом уровне пирамиды. Параметр типа `cv::TermCriteria` задаёт условия остановки итеративного процесса: процесс прекращается при достижении максимального числа итераций (`maxCount`) или когда изменение смещения между итерациями становится меньше заданного `epsilon` (в пикселях).

3. Описание решения

В данном разделе будет описана реализация алгоритма распознавания дороги вдали с автоматизированным выделением области².

3.1. Автоматизированное выделение области интереса

Для выделения области интереса был реализован класс `FlowHandler`. Основные возможности, которые предоставляет данный класс, перечислены ниже.

- Ручная настройка размера, положения в момент начала движения и положения при отсутствии поворота области интереса (по умолчанию данные параметры настраиваются под изображения датасета `ApolloScapes`).
- Детекция поворотов, смещение ROI между последовательными кадрами в момент поворота и возврат ROI в положение по умолчанию при отсутствии поворота в течение заданного количества кадров.

3.1.1. Особенности реализации

Получение последовательных кадров осуществляется при помощи объекта класса, производного от `Reader` (реализованный ранее базовый класс для считывания изображений). Затем на текущем кадре выделяется `initNumOfPoints` «хороших» точек с помощью функции OpenCV `cv::goodFeaturesToTrack`, на расстоянии не менее `minDistBetweenGoodPoints` пикселей друг от друга.

После выделения точек на текущем кадре происходит их поиск на следующем кадре при помощи функции вычисления оптического потока методом Лукаса–Канаде в OpenCV `cv::calcOpticalFlowPyrLK` с

²Реализация. URL: https://github.com/artem-smola/RoadDirectionRecognition_AtDistance (дата обращения: 18.05.2025).

размером окна `windowSize × windowSize` и количеством уровней пирамиды `numPyramidLevels`. Итеративное уточнение оценки прекращается после `maxIterations` итераций или при смещении окна между итерациями, меньшем `minWindowOffset`.

Индикаторами поворота назовём успешно отслеженные точки, которые сместились по горизонтали на не менее чем `minVariance` пикселей (между двумя кадрами). Считаем, что автомобиль поворачивает, если доля успешно отслеженных точек, которые являются индикаторами, между последовательными кадрами не меньше, чем `shareOfIndicators`.

В качестве смещения ROI между последовательными кадрами возьмём медианное значение смещений по горизонтали индикаторов поворота.

Если между последовательными кадрами доля отслеживаемых точек стала менее `shareOfTracked` от их начального количества, то набор «хороших» точек обновляется.

Ниже приведены значения некоторых параметров, выбранных для работы с изображениями датасета *ApolloScapes* (размер изображений датасета — 3387×2710 пикселей). Выбор значений производился вручную на небольшом (не более 5% от общего числа изображений) подмножестве датасета.

- `initNumOfPoints = 250`.
 - Большое количество отслеживаемых точек уменьшает влияние ошибок и точек, находящихся на объектах, имеющих значительную собственную скорость относительно дороги. Например, точки на машине, проезжающей перед камерой слева направо, на перекрестке движутся по горизонтали с достаточно высокой скоростью, хотя камера в этот момент может быть неподвижна относительно дороги.
- `minDistBetweenGoodPoints = 20`.
 - Обеспечивает достаточно равномерное расположение точек

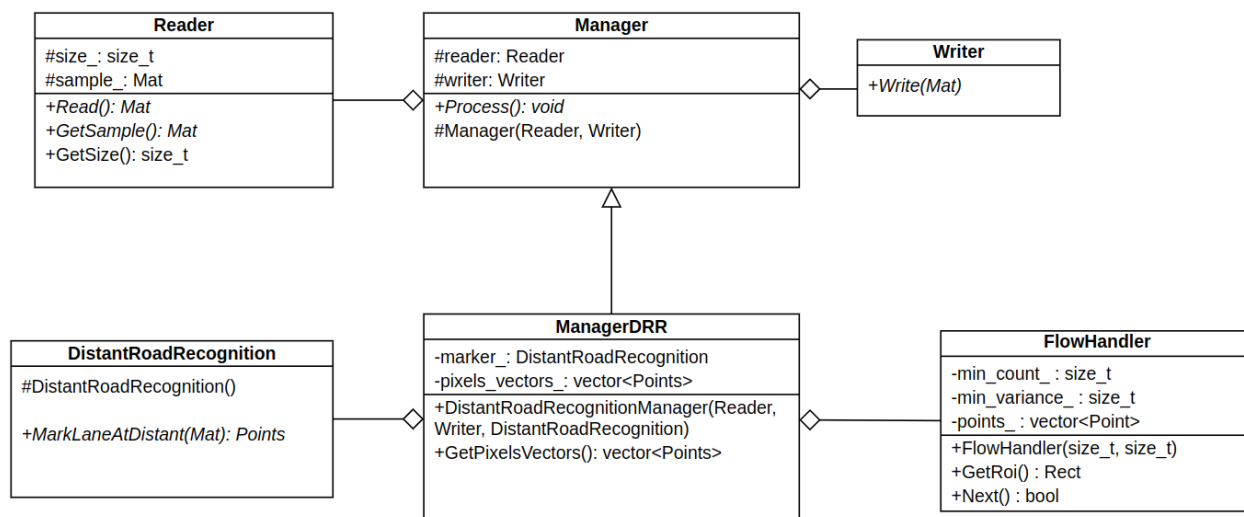


Рис. 2: Архитектура реализации ManagerDRR

на изображении.

- `windowSize = 21`.
 - Большой размер окна улучшает эффективность обнаружения отслеживаемых точек при быстром движении.
- `shareOfIndicators = 0,82`.
 - Если автомобиль поворачивает, то большая часть точек на видео будут иметь заметную скорость (по горизонтали) относительно камеры.

3.2. Итоговый алгоритм

Был реализован класс `ManagerDRR` для распознавания направления дороги вдаль с автоматизированным выделением ROI. Диаграмма класса `ManagerDRR` изображена на рисунке 2.

В итоговом алгоритме можно выделить следующие ключевые этапы.

1. Получение нового кадра и области интереса на нём при помощи объекта типа `FlowHandler`.
2. Обнаружение разметки в области интереса при помощи нейросети `TwinLiteNet` и получение вектора пикселей, относящихся к раз-

метке. Данную функциональность предоставляет объект класса, производного от `DistantRoadRecognition`.

3. Запись размеченного изображения в видеофайл либо в папку при помощи объекта класса, производного от `Writer`.

4. Эксперимент

4.1. Исследовательские вопросы

В ходе эксперимента предполагалось ответить на следующие исследовательские вопросы.

RQ1 : верно ли, что добавление в систему распознавания дороги автоматизированного выделения области интереса позволит получать более качественные результаты по выбранным метрикам?

RQ2 : насколько обновлённая система превосходит (по метрикам) модель TwinLiteNet, применённую ко всему изображению, на новой динамической области интереса?

RQ3 : каково влияние внесенных улучшений на скорость работы алгоритма?

4.2. Метрики

Для тестирования использовались следующие метрики:

- IoU,
- Precision,
- FPS.

4.3. Результаты

Как и в предыдущей работе, тестирование проводилось на датасете ApolloScapes.

Тестовый стенд имел следующую конфигурацию.

- Процессор: AMD Ryzen 5 5500U with Radeon Graphics × 12
- RAM 16 GB
- Операционная система: Ubuntu 24.04.1 LTS

- Компилятор: g++ 13.3.0

Результаты тестирования представлены в таблице 3,

Алгоритм	IoU		Precision		FPS	
	М	σ	М	σ	М	σ
Предлагаемое решение	30,6%	20,0%	70,2%	22,4%	7,4	0,7
Базовое решение	15,6%	11,2%	30,9%	24,7%	16,4	0,9
Предыдущее решение	19,6%	17,9%	32,7%	28,0%	13,8	0,9

Таблица 3: Результаты тестирования

где М — математическое ожидание, σ — среднеквадратическое отклонение, предлагаемое решение — система с автоматизированным выделением области интереса, базовое решение — модель TwinLiteNet, применённая ко всему изображению и рассмотренная на новой динамической области интереса, предыдущее решение — решение со статической областью интереса.

4.3.1. RQ1

Результаты тестирования показали, что система с автоматизированным выделением области интереса выдаёт более качественный результат, чем предыдущее решение, о чём свидетельствуют лучшие значения по метрикам IoU и Precision.

4.3.2. RQ2

Было выяснено, что улучшенная система распознавания дороги превосходит базовое решение почти в два раза по метрике IoU и более чем в два раза по метрике Precision уже на новой области интереса.

4.3.3. RQ3

Тестирование показало, что внедрение системы автоматизированного выделения области интереса заметно снизило значение по метрике

FPS на тестовом оборудовании. Предполагается, что реальные встраиваемые устройства, которыми оснащаются автомобили с ADAS, более производительны, чем тестовый стенд, и скорость работы системы на подобном оборудовании будет удовлетворительной. Подтверждение данного утверждения в рамках работы не представляется возможным из-за отсутствия доступа к соответствующим устройствам.

Заключение

1. Выполнен обзор реализованной ранее системы распознавания дороги вдали и технологий, использованных для реализации алгоритма автоматизированного выделения области интереса.
2. Реализована система автоматизированного выделения области интереса.
3. Внедрён алгоритм автоматизированного выделения области интереса в систему распознавания направления дороги вдали.
4. Проведены эксперименты для оценки работы алгоритма.
 - (a) Новая система демонстрирует лучшие результаты по метрикам IoU и Precision в сравнении с предыдущим решением.
 - (b) Система распознавания дороги вдали с автоматизированным выделением области интереса превосходит базовое решение почти в два раза по метрике IoU и более чем в два раза по метрике Precision.
 - (c) Внедрение алгоритма автоматизированного выделения области интереса значительно снижает значение по метрике FPS на тестовом оборудовании, что предположительно может быть компенсировано использованием более мощных устройств, типичных для задач ADAS.

Репозиторий с реализацией: https://github.com/artem-smola/RoadDirectionRecognition_AtDistance/blob/main/src/road_recognition/flow_handler.hpp (дата обращения: 18.06.2025).

Список литературы

- [1] Alexey Dosovitskiy Philipp Fischer Eddy Ilg Philip Hausser Caner Hazirbas Vladimir Golkov Patrick van der Smagt Daniel Cremers Thomas Brox. FlowNet: Learning Optical Flow With Convolutional Networks. — URL: https://openaccess.thecvf.com/content_iccv_2015/html/Dosovitskiy_FlowNet_Learning_Optical_ICCV_2015_paper.html (дата обращения: 18 июня 2025 г.).
- [2] Berthold K.P. Horn Brian G. Schunck. Determining optical flow. — URL: <https://www.sciencedirect.com/science/article/abs/pii/0004370281900242> (дата обращения: 5 мая 2025 г.).
- [3] Optical Flow - Everything You Need To Know in 2025. — URL: <https://viso.ai/deep-learning/optical-flow/> (дата обращения: 1 мая 2025 г.).
- [4] Артём Смола. Распознавание направления дороги вдали по видео в ADAS (осень). — URL: https://se.math.spbu.ru/thesis/texts/Smola_Artem_Mihajlovich_Autumn_practice_2nd_year_2024_text.pdf (дата обращения: 3 апреля 2025 г.).