Санкт-Петербургский государственный университет

Системное программирование

Группа 23.Б10-мм

Обзор решений распознавания препятствий по видео с камеры кругового обзора

Ри Александра Ильинична

Отчёт по учебной практике
в форме «Эксперимент»

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

Консультант:
инженер-программист, АО «Кама», Осечкина М. С.

Санкт-Петербург
2025

Оглавление

Введение **3**

1. Постановка задачи **5**

2. Обзор **6**

2.1. Используемые метрики 6

2.2. Классические алгоритмы компьютерного зрения 7

2.3. Методы на основе глубокого обучения 9

3. Эксперимент **17**

3.1. Условия эксперимента 17

3.2. Исследовательские вопросы 17

3.3. Метрики 17

3.4. Результаты 18

Заключение **20**

Список литературы **21**

Введение

Современные системы помощи водителю играют ключевую роль в обеспечении безопасности на дороге. Одной из важнейших задач в этом направлении является автоматическое распознавание препятствий, что позволяет значительно снизить риск возникновения аварийных ситуаций. Одной из наиболее эффективных технологий, помогающих в решении этой задачи, является анализ видеоинформации с камер кругового обзора. Такие камеры обеспечивают широкий угол обзора и дают возможность обнаружения объектов со всех сторон автомобиля.

На практике для кругового обзора часто применяются камеры с объективами типа «рыбий глаз», обеспечивающие сверхширокий угол захвата. Однако такие объективы создают нежелательные радиальные искажения, особенно заметные по краям изображения. Эти искажения усложняют обработку изображения, так как объекты могут казаться деформированными или смещенными относительно их реального положения.

На сегодняшний день существует ряд алгоритмов для обнаружения препятствий на видео и изображениях без искажений, такие как [9], [22], [29], [19], однако в открытых источниках отсутствуют данные об их точности при работе с видео, содержащим радиальные искажения. Также имеются алгоритмы, способные обрабатывать искаженные материалы, такие как [3], [10], [11], [31], но их реализации нет в открытом доступе. Это создаёт необходимость проведения исследований и тестирования алгоритмов с учётом ограничений по вычислительным ресурсам и времени обработки, а также их точности при обнаружении препятствий на искаженных изображениях.

В данной работе рассматриваются и сравниваются существующие алгоритмы для распознавания препятствий вокруг автомобиля, которые позволяют идентифицировать и классифицировать различные объекты, будь то другие автомобили, пешеходы или статические объекты. Сравнение проводится по таким критериям, как требуемая вычислительная мощность, точность обнаружения объектов на видео с камер с

объективами типа «рыбий глаз», возможность использования в режиме реального времени. Это сравнение необходимо для определения наиболее эффективных алгоритмов, которые могут быть интегрированы в системы помощи водителю для повышения безопасности и удобства на дороге.

1. Постановка задачи

Целью данной работы является анализ и сравнение существующих методов распознавания препятствий, их точность обнаружения на изображениях с радиальными искажениями, а также изучение эффективности этих методов в реальных условиях, где требуется работа в режиме реального времени и с ограничением по вычислительным ресурсам. Для её выполнения были поставлены следующие задачи:

1. выполнить обзор существующих решений;
2. протестировать методы на имеющихся наборах данных;
3. проанализировать результаты тестирования существующих решений.

2. Обзор

В этой главе будет проведен обзор наиболее популярных существующих алгоритмов обнаружения автомобилей и пешеходов, а также приведено описание метрик, используемых при тестировании.

2.1. Используемые метрики

2.1.1. mAP

mAP (Mean Average Precision) — общепринятая метрика для оценки качества алгоритмов, решающих задачу обнаружения и распознавания объектов на изображениях.

После получения данных с модели, вычисляется IoU (Intersection over Union) для каждого обнаруженного объекта. IoU рассчитывается как отношение площади пересечения предсказанной области и истинной области к площади их объединения. Вычисление IoU представлено на рисунке 1.

Выбирается пороговое значение IoU. В зависимости от сравнения IoU с этим порогом предсказанная область классифицируется следующим образом:

- True positve (TP) — предсказанный класс верен, а значение IoU этой области не меньше порогового;
- False positive (FP) — предсказанный класс не верен, объект не размечен в наборе данных, или не нашлось объекта, значение IoU которого выше порогового;
- False negative (FN) — объект размечен в наборе данных, но не предсказан решением.

Далее отдельно для каждого класса рассчитывается AP (Average Precision):

- рассчитывается Precision — соотношение истинно положительных прогнозов ко всем предсказанным, по формуле $Precision = \frac{TP}{TP+FP}$;

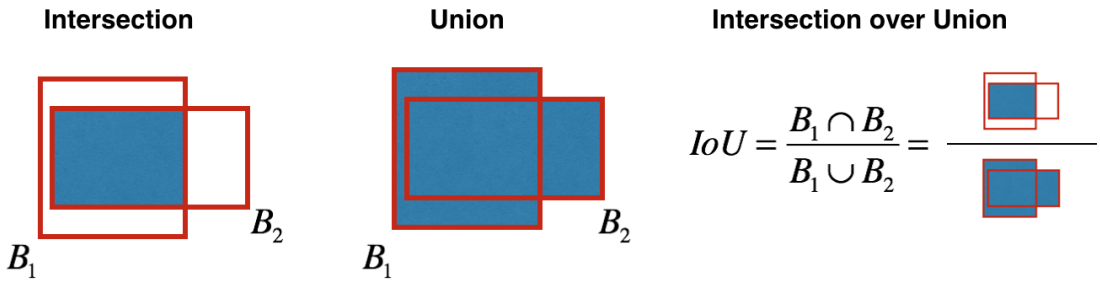

Рис. 1: Вычисление IoU

- рассчитывается Recall — соотношение истинно положительных прогнозов к общему числу элементов этого класса;
- строится Precision-Recall кривая, где по оси абсцисс отложен Recall и по оси ординат — Precision;
- AP для данного класса — площадь под графиком этой кривой.

После считается mAP по формуле

$$mAP = \frac{1}{N} \sum_{i=1}^{N} AP_i$$

где N — количество классов.

2.2. Классические алгоритмы компьютерного зрения

2.2.1. Метод Виолы-Джонса

Метод Виолы-Джонса [21] — алгоритм обнаружения объектов на изображениях в реальном времени, предложенный в 2001 году.

Сначала алгоритм вычисляет интегральное изображение, это позволяет эффективно вычислять сумму пикселей в любых прямоугольных областях изображения и ускорить вычисления признаков Хаара. Интегральное изображение вычисляется по формуле:

$$S(x, y) = \sum_{i=0}^{x} \sum_{j=0}^{y} I(i, j)$$

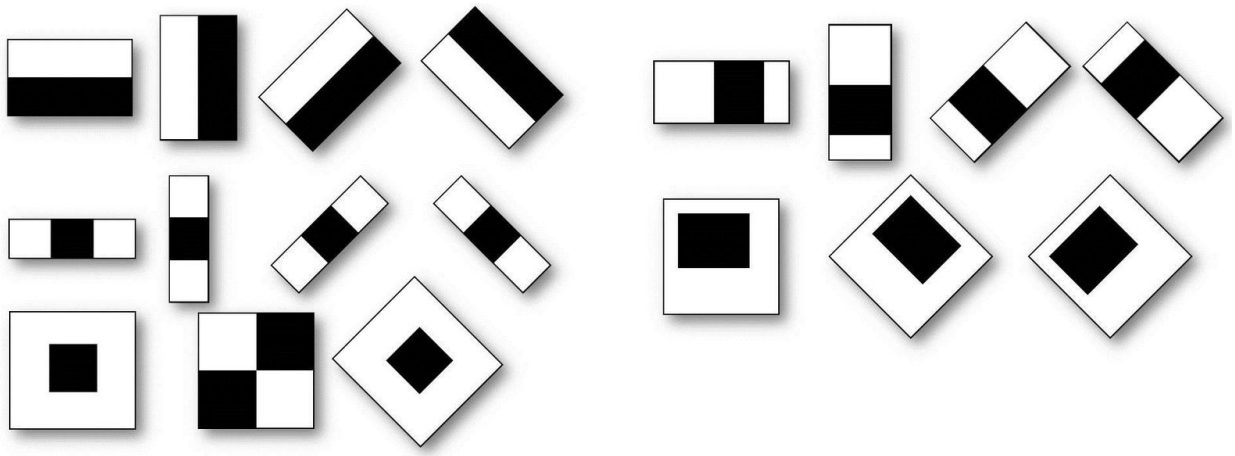

Рис. 2: Признаки Хаара

где $S(x, y)$ — значение интегрального изображения в точке (x, y), являющееся суммой всех пикселей в прямоугольной области, ограниченной $(0, 0)$ и текущей точкой (x, y), а $I(x, y)$ — интенсивность пикселя в исходном изображении на позиции (x, y).

Затем извлекаются признаки Хаара для каждого окна изображения, которое будет проверяться. Это специфичные геометрические формы, которые могут быть использованы для захвата различий в интенсивности пикселей в разных частях изображения. Они представляют собой прямоугольные области на изображении, которые измеряют разницу между суммами интенсивностей пикселей в разных частях изображения. Признаки Хаара показаны на рисунке 2.

Затем применяются классификаторы, предварительно обученные с использованием AdaBoost [5] [18], для увеличения точности слабых классификаторов путем их комбинации. Они объединяются в каскад, который последовательно применяет более сложные модели, если предыдущие не отклонили область, как не содержащую интересующий объект.

После прохождения через все этапы каскада, если объект был обнаружен, то эта область помечается как содержащая интересующий объект. Алгоритм применяет этот процесс к различным областям изображения, сдвигая окно по всей картинке, чтобы найти все возможные объекты.

В статье [31] предложен метод, также использующий признаки Хаара и классификатор AdaBoost [18] для определения пешеходов на изображениях, полученных с камер типа «рыбий глаз». Поскольку пешеходы являются вертикальными объектами, они подвержены горизонтальным искажениям. Авторы предлагают классифицировать обучающие наборы в соответствии с горизонтальными направлениями изгиба: влево, вправо и в центр. При обработке входное изображение делится на три области с перекрытием приграничных зон, чтобы исключить ложное определение объектов в этих частях изображения.

В статье отсутствует оценка точности метода, а реализация недоступна в открытом доступе, поэтому проведение оценки точности невозможно. Из-за этого нет возможности сравнить этот метод с другими.

2.3. Методы на основе глубокого обучения

CNN (Convolutional Neural Network) [14] — это тип нейронной сети, специально разработанный для работы с данными, имеющими сетчатую структуру, такими как изображения, предложенный в 1988 году.

Архитектура сети получила своё название из-за наличия свёрточных слоев, на которых применяется операции свёртки, суть которой в том, что каждый фрагмент изображения поэлементно умножается на матрицу (ядро) свёртки, а результат суммируется и записывается в соответствующую позицию выходного изображения. Это позволяет выявлять локальные особенности на изображениях, такие как края, текстуры и более сложные объекты, и сохранять пространственные отношения в изображении.

Скалярный результат каждой свёртки подаётся на нелинейную функцию активации (обычно используется ReLU (Rectified Linear Unit) [17] и её модификации). Функция обнуляет отрицательные значения, что помогает ускорить обучение и позволяет сети фокусироваться на важных характеристиках изображения.

Далее на слое подвыборки происходит нелинейное уплотнение карты признаков: группа пикселей (обычно размером 2×2) уплотняется до

одного пикселя. Чаще всего при этом используется функция максимума, при которой каждый непересекающийся прямоугольник или квадрат ужимается в один пиксель с максимальным значением. Этот слой уменьшает размерность изображения, снижая количество параметров и вычислительные затраты.

На уровне полносвязного слоя после обработки входных данных с помощью серии свёрточных и объединяющих слоёв выполняются высокоуровневые вычисления и прогнозирование на основе признаков, извлечённых и преобразованных предыдущими слоями.

На последнем, выходном слое генерируются вероятности классов объектов: применяется функция активации (например, Softmax [7]) для преобразования необработанных результатов, вычисляются потери для определения разницы между прогнозами и фактическими данными.

2.3.1. Семейство архитектур R-CNN

R-CNN (Region-based Convolutional Neural Networks) — это семейство архитектур и методов глубокого обучения, для задачи обнаружения объектов на изображениях. R-CNN и его более поздние версии (Fast R-CNN, Faster R-CNN и др.) решают задачу локализации объектов и их классификации. В отличие от простой задачи распознавания, задача обнаружения требует не только определения класса объекта, но и нахождения его местоположения в виде ограничивающей рамки.

Модели R-CNN считаются двухэтапными, так как процесс детекции объектов разделен на два последовательных этапа:

1. генерация предложений RoI (Regions of Interest);
2. классификация и уточнение координат обнаруженного объекта.

R-CNN Классический R-CNN [20], предложенный в 2014 году, включает три основных этапа процесса обнаружения объектов: генерацию предложений регионов, извлечение признаков и классификацию объектов с уточнением координат их ограничивающих рамок.

На первом этапе R-CNN создаёт предложения по областям, представляющим собой меньшие участки изображения, потенциально содержащие искомые объекты. Для этого используется метод Selective Search [23] — жадный алгоритм, генерирующий около 2000 регионов интереса (RoI). Эти регионы имеют различный размер и форму, где, предположительно, могут находиться объекты.

Каждый выделенный регион масштабируется до фиксированного размера (обычно 224×224 пикселей) и пропускается через CNN (VGG-16 [25]). На выходе для каждого RoI формируется вектор признаков.

Далее извлечённые признаки подаются на вход классификатору на основе SVM (Support Vector Machine), который определяет класс объекта в данном регионе. Каждый SVM обучен распознавать отдельный класс объектов, из-за чего вынужден проверять содержит ли конкретный регион объект данного класса для всех предложенных RoI. Классификатор, анализируя векторы признаков, также возвращает оценку достоверности, указывающую вероятность присутствия объекта в области. Дополнительно выполняется регрессия ограничительных рамок для уточнения координат ограничивающей рамки, что позволяет более точно определить границы объекта.

По данным [20], R-CNN обеспечивает mAP 53,7% на наборе данных PASCAL VOC 2010 [15], что на 18,6% выше, чем использование только Selective Search [23] для распознавания объектов (35,1%). Несмотря на более высокую точность обнаружения по сравнению с более ранними методами, R-CNN имеет следующие недостатки:

- низкая скорость работы, так как около 2000 регионов обрабатываются CNN по отдельности;
- высокое потребление памяти, связанное с хранением признаков для каждого региона.

Faster R-CNN Faster R-CNN [9] была представлена в 2015 году и стала улучшенной версией Fast R-CNN [8].

Авторы добавляют к Fast R-CNN два дополнительных сверточных

слоя — RPN (Region Proposal Network). RPN принимает изображение любого размера в качестве входных данных и выдает набор предложений регионов (RoI), каждое из которых имеет ограничивающие рамки разных маштабов и оценку уверенности, что область содержит объект. После каждый RoI обрабатывается уже Fast R-CNN.

Faster R-CNN выполняет классификацию объектов в каждом регионе и регрессию координат ограничивающих рамок в одной модели. Это позволяет оптимизировать весь процесс обнаружения объектов, потому что все этапы детекции происходят в одной модели, что упрощает обучение и повышает точность модели.

Согласно данным [9], Faster R-CNN обеспечивает mAP 70,4% на наборе данных PASCAL VOC 2012, что на 4,4% выше, чем Fast R-CNN (66%) и на 8% лучше, чем R-CNN (62%). Faster R-CNN требует меньше ресурсов и памяти, чем например R-CNN, поскольку карта признаков создается только один раз, а не многократно для каждого региона.

2.3.2. Семейство моделей YOLO

YOLO (You Only Look Once) — это архитектура и семейство моделей, предназначенных для детекции объектов на изображениях в режиме реального времени. YOLO относятся к одноэтапным моделям, так как выполняют обработку всего изображения за один проход через нейронную сеть.

YOLOv1 YOLOv1 [30] — это первая версия алгоритма YOLO, предложенная в 2016 году.

Обработка начинается с масштабирования и нормализации изображения, после чего оно пропускается через CNN (Darknet19 [4]), состоящую из 24 свёрточных слоев, за которыми следуют 2 полносвязных слоя. Картинка разбивается на сетку $S \times S$. Если центр обнаруженного после объекта попадает в ячейку (grid cell), то эта ячейка отвечает за обнаружение этого объекта. На выходе получим тензор (многомерная структура данных, обобщение скаляров, векторов и матриц на любое количество измерений) размера $S \times S \times (B \times 5 + C)$, где B — количество

ограничивающих рамок для каждого обнаруженного объекта (выбирается при обучении), C — количество классов объектов. Авторы используют $S = 7, B = 2$, и, так как набор PASCAL VOC содержит 20 классов объектов, то $C = 20$.

Каждая ячейка предсказывает B прямоугольников для определения объекта, а их позиции, ширина и высота вычисляются относительно центра этой клетки. Таким образом каждая ячейка предсказывает $B \times (x, y, w, h, C_{conf}) + C$, где x, y — координаты центра рамки относительно ячейки, w, h — ширина и высота рамки относительно всего изображения, C_{conf} — коэффициент уверенности для рамки, определяющий, насколько модель уверена, что рамка содержит объект и насколько модель считает, что рамка соответствует объекту.

Каждая ячейка сетки также предсказывает C условных вероятностей классов $P_{obj}(Class_i|Object)$. Эти вероятности вычисляются при условии, что ячейка содержит объект. При этом для каждой ячейки предсказывается только один набор вероятностей классов для каждой ячейки, независимо от числа B ограничивающих рамок.

Затем параметры рамки пересчитываются в координаты изображения и удаляются рамки с C_{conf} ниже порогового значения, заданного при обучении. Для устранения перекрывающихся рамок применяется NMS (Non-Maximum Suppression) [6]. При этом сохраняется рамка с максимальным C_{conf}, если её IoU с другой рамкой превышает заданный порог.

В результате остаются:

- Координаты объектов (прямоугольные рамки);
- Классы объектов;
- Коэффициенты уверенности классов объектов.

Эти данные выводятся пользователю.

Так как YOLOv1 обрабатывает изображение за один проход, по данным авторов [30], модель способна обрабатывать 45 кадров в секунду

(FPS), что значительно быстрее, чем, например, Fast R-CNN (обработка одного изображения занимает 2–3 секунды) или Faster R-CNN 7 FPS. Однако сами авторы отмечают, что их модель уступает в точности другим подходам. Например, на данных PASCAL VOC 2012 YOLOv1 достигает результата 57,9%, в то время как Fast R-CNN — 66%, а Faster R-CNN — 70,4%. Это связано с трудностями модели при обнаружении объектов небольшого размера.

YOLOv4 YOLOv4 [29] — последняя официально выпущенная модель, представленная в 2020 году, которая улучшила точность предыдущих версий (YOLOv2 [27] и YOLOv3 [28]).

В YOLOv4 используется сверточная сеть CSPDarknet53 [2] — улучшенная версия Darknet-53 с CSP (Cross-Stage Partial Networks), которая включает 53 сверточных слоя. CSP разделяет поток данных на две части: одна часть проходит через сложные слои, а другая — через упрощённые. Это снижает избыточность вычислений и улучшает обучение.

Для объединения признаков с разных слоев используется Spatial Pyramid Pooling (SPP) [24], который извлекает признаки объектов разного размера, а также Path Aggregation Network (PANet) [16] для передачи признаков от низкоуровневых слоёв к высокоуровневым.

С выходом YOLOv2 был введён Batch Normalization [1] (метод нормализации, используемый в нейронных сетях для ускорения и стабилизации процесса обучения). Тогда же отказались от использования полносвязных слоев, что позволило обучать модель на изображениях разного размера, сделав её более устойчивой к различным размерам входных данных.

В YOLOv3 вместо функции активации SoftMax [7] для генерации вероятностей классов объектов (как в «классических» CNN), на выходном слое используется независимая бинарная классификация для каждого класса. Это позволяет обнаруживать объекты, которые могут принадлежать нескольким классам одновременно (объект может быть «человеком» и «велосипедистом»).

В YOLOv2 были добавлены Anchor Boxes (якорные рамки): для каж-

дой ячейки задается несколько предварительно установленных «прототипных» боксов, и модель предсказывает координаты и класс для каждого из них. В YOLOv3 модель предсказывает три набора Anchor Boxes в разных масштабах, что делает её более гибкой при работе с объектами различных размеров.

При создании YOLOv2 авторы представили метод совместного обучения детекции и классификации на отдельных датасетах — COCO [13] и ImageNet [12] соответственно. В результате получилась модель, способная обнаружить более 9000 классов.

По данным авторов [29] модель превосходит все существующие альтернативные детекторы на наборах данных COCO и обеспечивает 43,5% при FPS 30, что на 5-10% больше чем другие методы, таблицы сравнения будут приведены ниже.

2.3.3. SSD

SSD (Single Shot MultiBox Detector) [22] — еще одна одноэтапная модель, представленная в 2016 году.

В SSD обработка изображения начинается с подачи изображения в CNN (авторы используют VGG-16 [25] с несколькими дополнительными слоями). Дополнительные слои обрабатывают карты признаков, полученные после обработки CNN, в нескольких масштабах, что позволяет SSD эффективно распознавать объекты разных размеров.

Для каждой ячейки на карте признаков, полученной с VGG-16, задается набор ограничивающих рамок по умолчанию (default bounding boxes).

Дополнительные сверточные слои используют ядра небольшого размера (3×3), которые выдают только два типа результата: вероятность принадлежности области к определенному классу или необходимое смещение координат рамки по умолчанию, для лучшего охвата объекта. Т.е. для каждой рамки по умолчанию вычисляется c вероятностей принадлежности классу и 4 смещения.

Для устранения перекрывающихся рамок отбрасываются все, имеющие вероятность ниже определенного порога, а также используется

функция NMS [6], которая оставляет только рамку с наивысшей вероятностью.

По данным авторов [22] на тесте PASCAL VOC 2007 модель обеспечивает для изображений 300×300 74,3% mAP и 76,9% mAP для изображений 512×512.

3. Эксперимент

3.1. Условия эксперимента

Проводилось тестирование предобученных разработчиками моделей Faster R-CNN, SSD, YOLOv4 на наборе данных с радиальными искажениями — WoodScape [26], на оборудовании: GPU NVIDIA Tesla T4 (на Google Colab).

Полученные данные сравнивались с результатами моделей на наборе данных без искажений PASCAL VOC 2007[1] [15]. Этот набор данных был выбран из-за доступности результатов детекции моделей на этом наборе.

3.2. Исследовательские вопросы

Были сформулированы следующие исследовательские вопросы:

RQ1 : насколько точно эти модели определяют объекты?

RQ2 : возможно ли использование этих моделей в режиме реального времени?

RQ3 : правда ли, что описанные выше модели могут использоваться в ADAS для распознавания объектов на изображениях с искажениями типа «рыбий глаз» в режиме реального времени?

3.3. Метрики

В качестве метрик для оценки алгоритмов было решено использовать:

- для оценки точности определения объектов — mAP;
- для определения скорости обработки — FPS.

[1] http://host.robots.ox.ac.uk/pascal/VOC/voc2007/ (дата обращения: 14.12.2024)

3.4. Результаты

Модели были протестированы на 8234 изображениях. После получения предсказанных моделями ограничивающих рамок было посчитано mAP с ограничивающими рамками из аннотации набора данных. Были получены следующие результаты, представленные в таблице 1.

Модель	**PASCAL VOC 2007**		**WoodScape**	
	mAP	FPS	mAP	FPS
Faster R-CNN	73,2%	7,0	31,0%	7,0
SSD	77.2%	46	35.1%	8.3
YOLOv4	82,0%	45,0	39,2%	9,2

Таблица 1: Результаты работы моделей

3.4.1. RQ1

Модели показывают существенно более низкую точность на изображениях типа «рыбий глаз», чем на неискаженных изображениях. Разница составляет более 40%.

3.4.2. RQ2

Для корректной работы алгоритма в режиме реального времени требуется скорость обработки кадров 30 FPS и выше. Результаты исследования же показывают, что модели имеют менее 10 FPS. Согласно открытым источникам используемые в ADAS GPU в 4-5 раз мощнее используемого. При такой «грубой» оценке модели преодолевают порог 30 FPS.

3.4.3. RQ3

Результаты тестирования показывают, что модели не достаточно точно определяют объеты на изображениях с искажениями типа «рыбий глаз». Из-за невозможности тестирования на мощном оборудовании невозможно точно узнать FPS и оценить, точно ли возможна работа в режиме реального времени.

Это говорит о невозможности использования этих моделей без модификаций для определения объектов в системах ADAS.

Заключение

В ходе данной работы были выполнены следующие задачи:

- выполнен обзор существующих решений распознавания объектов на неискаженных изображениях;
- выбраны модели распознавания объектов на изображениях и протестированы на наборе изображений с искажениями типа «рыбий глаз»[1];
- было выяснено, что выбранные модели не подходят для распознавания препятствий на изображениях с радиальными искажениями в режиме реального времени с ограничением по вычислительным ресурсам.

[1]Реализация в GitHub репозитории URL:https://github.com/xImoZA/detection_models_on_fisheye (дата обращения: 3.01.2025)

Список литературы

[1] Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. — URL: https://arxiv.org/pdf/1502.03167.

[2] CSPNet: A New Backbone that can Enhance Learning Capability of CNN. — URL: https://openaccess.thecvf.com/content_CVPRW_2020/papers/w28/Wang_CSPNet_A_New_Backbone_That_Can_Enhance_Learning_Capability_of_CVPRW_2020_paper.pdf.

[3] CubemapSLAM: A Piecewise-Pinhole Monocular Fisheye SLAM System. — URL: https://www.researchgate.net/publication/329362615_CubemapSLAM_A_Piecewise-Pinhole_Monocular_Fisheye_SLAM_System.

[4] Darknet: Open Source Neural Networks in C. — URL: https://pjreddie.com/darknet/.

[5] A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. — URL: https://sci2s.ugr.es/keel/pdf/algorithm/articulo/1997-JCSS-Schapire-A%20Decision-Theoretic%20Generalization%20of%20On-Line%20Learning%20%28AdaBoost%29.pdf.

[6] Distance-IoU Loss: Faster and Better Learning for Bounding Box Regression. — URL: https://arxiv.org/abs/1911.08287.

[7] Efficient Precision-Adjustable Architecture for Softmax Function in Deep Learning. — URL: https://www.researchgate.net/publication/342225584_Efficient_Precision-Adjustable_Architecture_for_Softmax_Function_in_Deep_Learning.

[8] Fast R-CNN. — URL: https://arxiv.org/pdf/1504.08083.

[9] Faster R-CNN: Towards Real-Time Object Detection with Region

Proposal Networks. — URL: https://www.semanticscholar.org/reader/424561d8585ff8ebce7d5d07de8dbf7aae5e7270.

[10] FisheyeDetNet: 360∘Surround view Fisheye Camera based ObjectDetection System for Autonomous Driving. — URL: https://www.researchgate.net/publication/380457646_FisheyeDetNet_360_Surround_view_Fisheye_Camera_based_Object_Detection_System_for_Autonomous_Driving.

[11] FisheyeYOLO: Object Detection on Fisheye Camerasfor Autonomous Driving. — URL: https://www.researchgate.net/publication/346931586_FisheyeYOLO_Object_Detection_on_Fisheye_Cameras_for_Autonomous_Driving.

[12] ImageNet Large Scale Visual Recognition Challenge. — URL: https://www.semanticscholar.org/reader/e74f9b7f8eec6ba4704c206b93bc8079af3da4bd.

[13] Microsoft COCO: Common Objects in Context. — URL: https://arxiv.org/abs/1405.0312.

[14] OVERVIEW OF CONVOLUTIONAL NEURAL NETWORKS. — URL: https://www.researchgate.net/publication/320323499_OVERVIEW_OF_CONVOLUTIONAL_NEURAL_NETWORKS.

[15] The PASCAL Visual Object Classes (VOC) Challenge. — URL: https://www.researchgate.net/publication/220659463_The_Pascal_Visual_Object_Classes_VOC_challenge.

[16] Path Aggregation Network for Instance Segmentation. — URL: https://arxiv.org/abs/1803.01534.

[17] RELU-Function and Derived Function Review. — URL: https://www.researchgate.net/publication/362948739_RELU-Function_and_Derived_Function_Review.

[18] Rapid Object Detection Using a BoostedCascade of Simple Features. — URL: https://www.researchgate.net/publication/3940582_Rapid_Object_Detection_using_a_Boosted_Cascade_of_Simple_Features.

[19] Real-time Joint Object Detection and SemanticSegmentation Network for Automated Driving. — URL: https://www.researchgate.net/publication/330383030_Real-time_Joint_Object_Detection_and_Semantic_Segmentation_Network_for_Automated_Driving.

[20] Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. — URL: https://openaccess.thecvf.com/content_cvpr_2014/papers/Girshick_Rich_Feature_Hierarchies_2014_CVPR_paper.pdf.

[21] Robust Real-Time Object Detection. — URL: https://www.researchgate.net/publication/215721846_Robust_Real-Time_Object_Detection.

[22] SSD: Single Shot MultiBox Detector. — URL: https://www.researchgate.net/publication/308278279_SSD_Single_Shot_MultiBox_Detector.

[23] Selective Search for Object Recognition. — URL: https://www.researchgate.net/publication/262270555_Selective_Search_for_Object_Recognition.

[24] Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. — URL: https://arxiv.org/pdf/1406.4729.

[25] Very Deep Convolutional Networks for Large-Scale Image Recognition. — URL: https://arxiv.org/abs/1409.1556.

[26] WoodScape: A multi-task, multi-camera fisheye dataset for autonomous driving. — URL: https://arxiv.org/abs/1905.01489.

[27] YOLO9000: Better, Faster, Stronger. — URL: https://arxiv.org/abs/1612.08242.

[28] YOLOv3: An Incremental Improvement. — URL: https://arxiv.org/abs/1804.02767.

[29] YOLOv4: Optimal Speed and Accuracy of Object Detection. — URL: https://arxiv.org/pdf/2004.10934.

[30] You Only Look Once: Unified, Real-Time Object Detection. — URL: https://arxiv.org/abs/1506.02640.

[31] An effective pedestrian detection method for driver assistance system. — URL: https://www.researchgate.net/publication/254055958_An_effective_pedestrian_detection_method_for_driver_assistance_systemg.