УДК 004.89

Е. С. Кружмягина[1]✉, А. А. Шарапов[1]

Разработка программного обеспечения для обнаружения объектов в режиме реального времени с использованием методов искусственного интеллекта

[1]Сибирский государственный университет геосистем и технологий,
г. Новосибирск, Российская Федерация
e-mail: kruzmyagina04@mail.ru

Аннотация. В статье рассматривается разработка программного обеспечения для автоматического обнаружения объектов в видеопотоке на основе нейросетевых технологий. Актуальность работы обусловлена необходимостью минимизации человеческого фактора в системах видеонаблюдения и повышения эффективности мониторинга. В качестве базовой модели выбрана YOLOv8, обученная на датасете VisDrone, содержащем разнообразные сценарии городской и сельской местности. Реализация выполнена на Python с использованием библиотек OpenCV, Ultralytics и SQLite для хранения данных. Результаты тестирования показали среднюю точность 0.5476 при скорости обработки 37 FPS на GPU NVIDIA GTX 1650. Разработанное программное обеспечние предоставляет интерфейс для анализа видеопотоков, детекции объектов и генерации отчетов, что делает его применимым в охранных системах и службах экстренного реагирования.

Ключевые слова: видеонаблюдение, искусственный интеллект, YOLOv8, компьютерное зрение, Python, обнаружение объектов

E. S. Kruzmyagina[1]✉, A. A. Sharapov[1]

Development of software for real-time object detection using artificial intelligence methods

[1]Siberian State University of Geosystems and Technologies, Novosibirsk, Russian Federation
e-mail: kruzmyagina04@mail.ru

Abstract. The article discusses the development of software for automatic detection of objects in a video stream based on neural network technologies. The relevance of the work is due to the need to minimize the human factor in video surveillance systems and increase the effectiveness of monitoring. YOLOv8 was chosen as the base model, trained on the VisDrone dataset, which contains various scenarios of urban and rural areas. The implementation is made in Python using OpenCV, Ultralytics and SQLite libraries for data storage. The test results showed an average accuracy of 0.5476 with a processing speed of 37 FPS on an NVIDIA GTX 1650 GPU. The developed software provides an interface for analyzing video streams, detecting objects and generating reports, which makes it applicable in security systems and emergency response services.

Keywords: video surveillance, artificial intelligence, YOLOv8, computer vision, Python, object detection

Введение

В настоящее время видеонаблюдение стало неотъемлемой частью систем обеспечения безопасности, как в частном, так и в государственном секторе. С

развитием технологий и увеличением числа источников видеоданных, таких как камеры, БПЛА и т. д. появилась необходимость в новых и более эффективных инструментах анализа данных видеопотока. На данный момент большинство систем представляют из себя ручной мониторинг, требующий участия операторов и подверженный влиянию человеческого фактора.

Современные требования к системам безопасности подразумевают не просто фиксацию происходящего, а оперативное принятие решений на основе анализа ситуации. Это является особенно важным для таких организаций, как охранные предприятия, службы экстренного реагирования, МЧС и т. д. Для таких задач актуально внедрений программных решений, позволяющих анализировать видеопотоки в режиме реального времени, способных обнаруживать объекты, классифицировать их и формировать отчеты по полученным результатам видеонаблюдений.

Целью данной работы является разработка программного обеспечения для обнаружения объектов в режиме реального времени с использованием методов искусственного интеллекта.

Для достижения цели работы были поставлены следующие задачи:

– изучить предметную область;

– провести анализ существующих аналогов программных решений;

– провести обзор и выбор технологий для разработки программного обеспечения для обнаружения объектов в режиме реального времени;

– разработать функциональную модель программного обеспечения для обнаружения объектов в режиме реального времени с использованием методов искусственного интеллекта;

– разработать прототип программного обеспечения для обнаружения объектов в режиме реального времени с использованием методов искусственного интеллекта.

Обзор и выбор технологий для разработки программного обеспечения

Для проведения анализа были выбраны 3 нейросети: YOLOv8, RetinaNet и Faster R-CNN [1–3]. По описанию каждая модель имеет свои преимущества и недостатки, поэтому было решено провести свое тестирования данных нейросетей, которое поможет выбрать подходящую для нашей разработки.

Тестирование нейросетей происходило с использованием видеокарты NVIDIA GTX 1650 и датасета VisDrone [4]. По итогам проведенного тестирования была сделана сравнительная таблица (табл.1)

Таблица 1

Сравнительная таблица нейронных сетей

Параметр	YOLOv8	RetinaNet	Faster R-CNN
Средняя точность	31,8%	30,4%	34,9%
Скорость обработки (FPS)	37,0 FPS	10,2 FPS	5,4 FPS
Среднее количество детекций на кадр	8,7	7,9	9,2

Окончание таблицы 1

Параметр	YOLOv8	RetinaNet	Faster R-CNN
Время распознавания объектов для одного кадра	27,0 мс	97,2 мс	183,9 мс
Объемы GPU во время работы	2,1 GB	2,9 GB	3,7 GB
Уровень ошибок на сложном фоне (%)	12,3%	16,5%	9,1%
Максимальное количество одновременно распознанных объектов	24	14	19

Исходя из вышеприведенного сравнения видно, что лучшую сбалансированность показала модель YOLOv8 [5].

Для разработки программного обеспечения для обнаружения объектов в режиме реального времени был выбран язык программирования Python. Во-первых, Python имеет крупное сообщество и документацию [6]. Во-вторых, может интегрироваться с базами данных, графическими интерфейсами и т.д. Python позволяет работать с различными библиотекам, таким как sqlite3, Tkinter, OpenCV, Flask [7, 8]. В-третьих, он часто применяется для работ в области машинного обучения и компьютерного зрения, что подходит для моей задачи.

В качестве системы управления базами данных для данного проекта была выбрана SQLite. SQLite представляет собой простую встраиваемую систему, которая хранит все данные в одном файле и не требует настройки серверной части.

Разработка программного обеспечения

В начале работы было принято решение спроектировать диаграмму IDFE0 для описания функциональной структуры. В ходе работы были спроектированы уровни A-0 и A-1(рис. 1, 2).

Во время разработки было решено сделать архитектуру программного обеспечения модульной, что поможет обеспечить изоляцию компонентов при необходимости дальнейшего масштабирования программного обеспечения.

Модуль авторизации пользователей реализован с использование SQLite для хранения учетных данных. Модуль проверяет введенные данные и предоставляет доступ к основному интерфейсу программы.

Взаимодействие с видеопотоком реализовано посредством библиотеки OpenCV. В программе используется многопоточный подход для обработки видео. Видео обрабатывается без потерь кадров даже при работе в фоновом режиме.

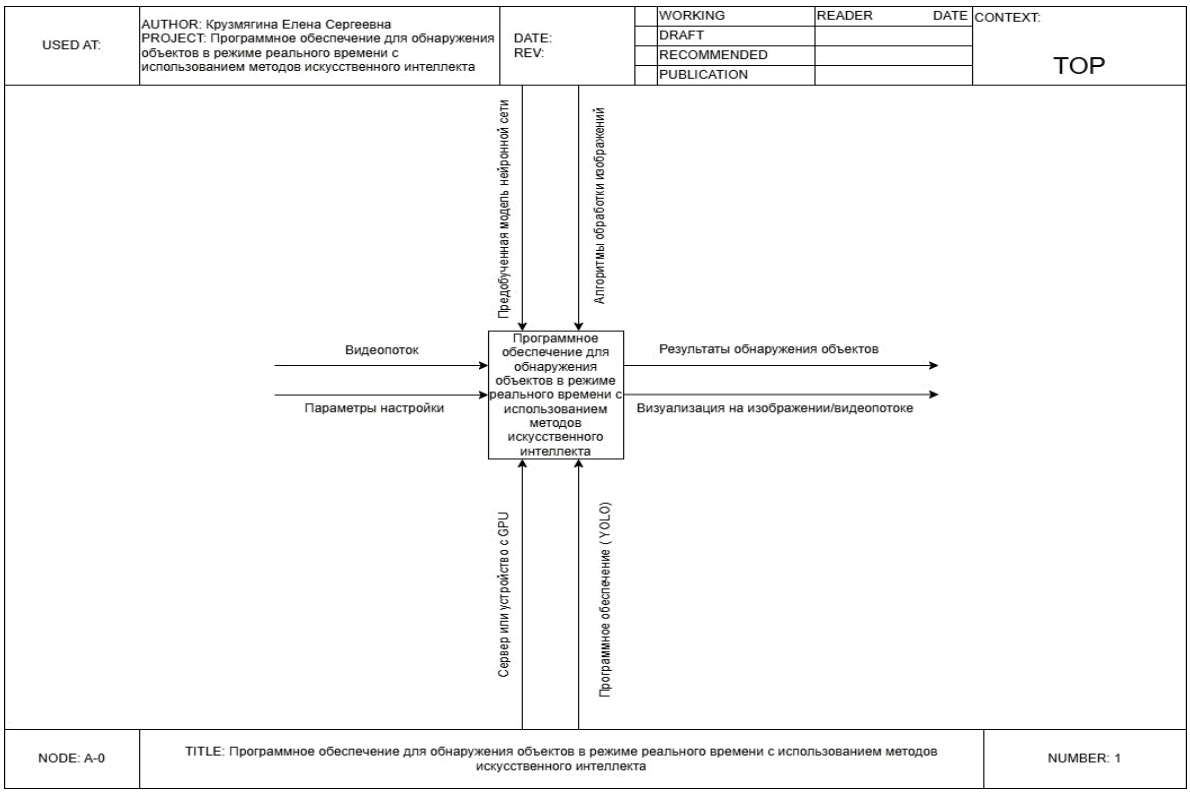

Рисунок 1 – Диаграмма нулевого уровня IDEF0

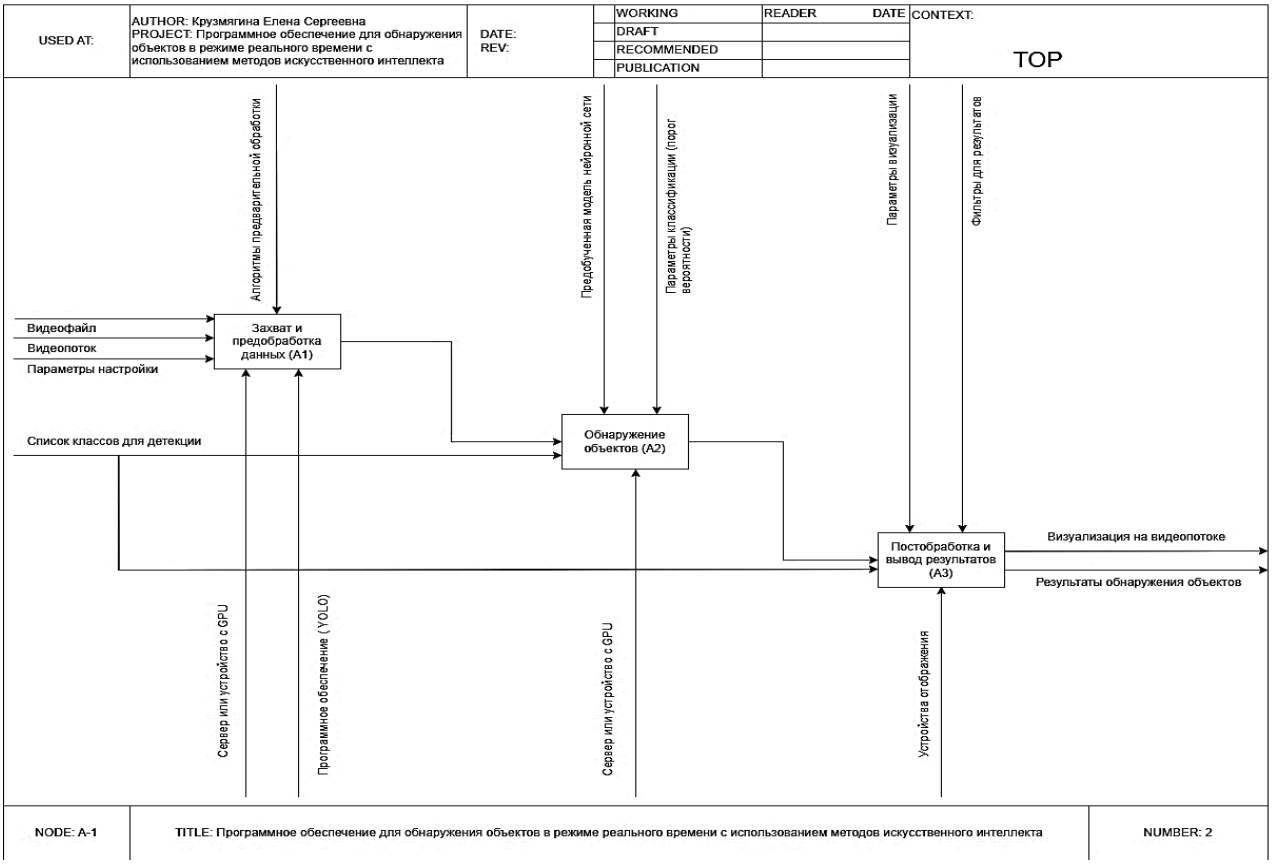

Рисунок 2 – Диаграмма первого уровня IDEF0

Для оптимизации и ускорения обработки применяются:

– кэширование результатов обнаружения;

– асинхронная запись в базу данных;

– оптимизированные размеры кадров.

Ниже представлен код для отрисовки обнаруженных на кадре объектов и сохраняет результат в видеофайл:

```
display_frame = frame.copy()
for detection in self.current_detections:
        x1, y1, x2, y2 = detection['bbox']
        cv2.rectangle(display_frame, (x1, y1), (x2, y2), (0, 255, 0), 2)
        cv2.putText(display_frame, detection['label'], (x1, y1 - 10),
        cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 2)
self.current_output.write(display_frame)
```

Программа выполняет захват кадров в реальном времени и передает их в нейросетевой модуль, который построен на базе фреймворка PyTorch с использованием обученной нейросети YOLOv8.

Модель загружается динамически, что позволяет пользователю выбирать необходимую версию:

```
folder_path = filedialog.askdirectory(
          title="Выберите папку с нейросетью")
       if folder_path:
          pt_files = [f for f in os.listdir(folder_path) if f.endswith('.pt')]
          yaml_files = [f for f in os.listdir(folder_path) if f.endswith('.yaml')]
          if not pt_files:
             print("В выбранной папке не найден файл нейросети (.pt)!")
             return
          if not yaml_files:
             print("В выбранной папке не найден конфигурационный файл
(.yaml)!")
             return
```

Модель предусматривает загрузку любых совместимых .pt файлов, а также .yaml конфигураций. После загрузки модели пользователь может выбрать, какие типы объектов программа должна отслеживать. Обнаруженные объекты визуализируются на экране в виде рамок с подписями классов.

Модуль работы с базой данных реализован с использованием SQLite, он обеспечивает локальное хранение данных о событиях. В базе создаются таблицы для хранения информации о типе объекта, времени и дате обнаружения, количестве экземпляров.

Пользовательский интерфейс реализован с использованием стандартной библиотеки Tkinter. Для отображения изображений в интерфейсе применялась библиотека Pillow, обеспечивающая преобразование форматов и отображение изображений внутри GUI-элементов. Интерфейс выполнен в виде системы вкладок, каждая

из которых отвечает за определенный функциональный блок. Такой подход к организации интерфейса упрощает навигацию в программе.

Со стороны пользователя программное обеспечние выглядит следующим образом (рис. 3).

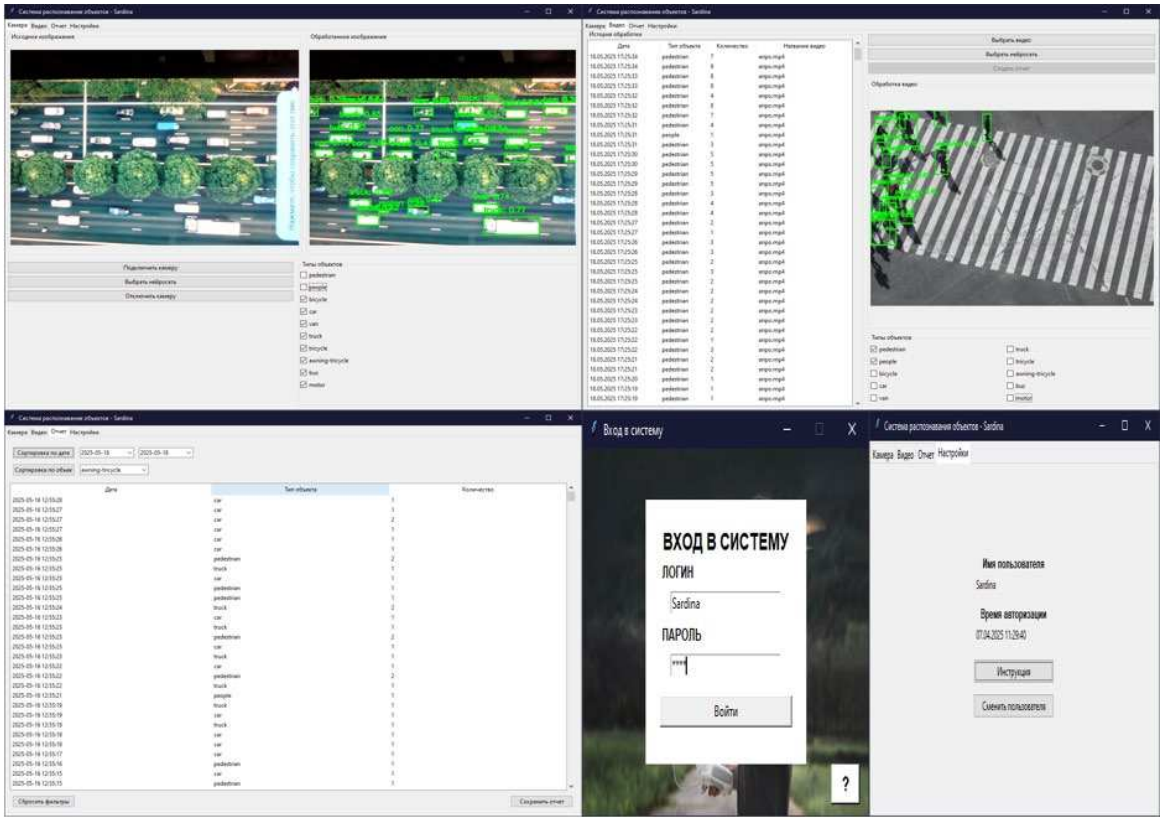

Рисунок 3 – Интерфейс программного обеспечения

Заключение

Результатом данной работы является разработка программного обеспечения для обнаружения объектов в режиме реального времени. Реализованное программное обеспечение сочетает в себе инструменты для реального времени и офлайн-анализа, что расширяет спектр его возможного применения. Программа не требует установки сторонних серверов или облачных компонентов, что делает ее удобной для использования на локальных машинах, в том числе без постоянного подключения к сети. Данное программное обеспечение может быть расширено в зависимости от потребностей пользователей.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Архитектура нейронной сети RetinaNet // Хабр URL: https://habr.com/ru/articles/510-560/ (дата обращения: 24.03.2025).

2. RetinaNet // PseudoLab URL: https://pseudo-lab.github.io/Tutorial-Book-en/chapters/en/object-detection/Ch4-RetinaNet.html (дата обращения: 24.03.2025).

3. Двухстадийные детекторы // Машинное обучение URL: https://deepmachinelearning.ru/docs/Neural-networks/Object-detection/Two-stage-detectors (дата обращения: 24.03.2025).

4. Набор данных VisDrone // Ultralytics URL: https://docs.ultralytics.com/ru/datasets/detect/visdrone/ (дата обращения: 15.02.2025).

5. Работа с YOLOV8. Детекция, сегментация, трекинг объектов, а также подготовка собственного датасета и обучение // Хабр URL: https://habr.com/ru/articles/821971/ (дата обращения: 15.02.2025).

6. Python 3.13.3 documentation // Python URL: https://docs.python.org/3/ (дата обращения: 20.02.2025).

7. GUI-приложения с помощью Python-Tkinter // Хабр URL: https://habr.com/ru/companies/otus/articles/903526/ (дата обращения: 27.01.2025).

8. OpenCV: компьютерное зрение на Python // Хабр URL: https://habr.com/ru/companies/otus/articles/849136/ (дата обращения: 27.01.2025).