

ИНФОРМАТИКА И ИНФОРМАЦИОННЫЕ ПРОЦЕССЫ/INFORMATICS AND INFORMATION PROCESSES

DOI: https://doi.org/10.60797/IRJ.2026.170.57 EDN: ZRFSKO

ДВУХМОДЕЛЬНАЯ АНСАМБЛЕВАЯ АРХИТЕКТУРА АВТОМАТИЧЕСКОГО ОБНАРУЖЕНИЯ УТОПАЮЩИХ С АЛГОРИТМАМИ МЕЖКАДРОВОГО ТРЕКИНГА И АДАПТИВНОЙ КАЛИБРОВКИ ПОРОГОВ РЕШАЮЩЕГО ПРАВИЛА

Научная статья

Козлов Э.Ю.[1], Гибадуллин Р.Ф.[2,] *
[2] ORCID : 0000-0001-9359-911X;
[1, 2] Казанский национальный исследовательский технический университет им. А.Н. Туполева – КАИ, Казань, Российская Федерация

* Корреспондирующий автор (landwatersun[at]mail.ru)

Предложена: 09.05.2026; Принята: 26.06.2026; Опубликована: 17.08.2026

Аннотация

Рассмотрена задача автоматического обнаружения утопающих по видеопотоку бассейна, в которой основная сложность смещена из распознавания отдельных поз в инженерное окружение классификатора — обеспечение низкой частоты ложных срабатываний, малой задержки реакции и независимой замены нейросетевых компонентов. Предложена двухмодельная ансамблевая архитектура, разносящая пространственный анализ (детекция и оценка поз сетью YOLOv8m-Pose) и темпоральный анализ (двухслойная двунаправленная LSTM с аддитивным вниманием) по независимым моделям и дополняющая их тремя системными компонентами: межкадровым IoU-трекингом, постобработкой со скользящим усреднением вероятностей и пороговым решающим правилом, а также адаптивной калибровкой порога по статистике, набираемой непосредственно на объекте эксплуатации. Архитектура реализована как пятиуровневый конвейер с единичной ответственностью уровней и фиксированным тензорным интерфейсом, что обеспечивает раздельное обучение моделей и масштабирование по числу камер. Экспериментально установлено, что свёрточный детектор достигает mAP@0.5 = 0,809 и F1 = 0,802 на тестовой подвыборке, а полный пайплайн на наборе из 25 видеороликов даёт на уровне инцидентов точность 0,77, полноту 0,92 и F1 = 0,84 при частоте ложных срабатываний ≈1 событие в час на камеру и средней задержке обнаружения 2,9 с. Проанализированы устойчивость к синтетическим помехам, режимы вычислений (оптимален TensorRT FP16) и масштабируемость: одна видеокарта NVIDIA RTX 4080 обрабатывает 14–16 камер одновременно. По частоте ложных срабатываний предложенное решение превосходит коммерческие и академические аналоги при сопоставимой полноте обнаружения.

Ключевые слова: обнаружение утопающих, видеоаналитика, безопасность на воде, ансамблевая архитектура, оценка поз, YOLOv8, двунаправленная LSTM, механизм внимания, межкадровый трекинг, адаптивная калибровка порога, частота ложных срабатываний.

A TWO-MODEL ENSEMBLE ARCHITECTURE FOR THE AUTOMATIC DETECTION OF DROWNERS WITH INTER-FRAME TRACKING ALGORITHMS AND ADAPTIVE CALIBRATION OF DECISION RULE THRESHOLDS

Research article

Kozlov E.Y.[1], Gibadullin R.F.[2,] *
[2] ORCID : 0000-0001-9359-911X;
[1, 2] Kazan National Research Technical University named after A.N. Tupolev – KAI, Kazan, Russian Federation

* Corresponding author (landwatersun[at]mail.ru)

Suggested: 09.05.2026; Accepted: 26.06.2026; Published: 17.08.2026

Abstract

The problem of automatically detecting drowners in a live video feed from a swimming pool is examined; in this task, the main challenge lies not in recognising individual poses but in the design of the classifier — ensuring a low false alarm rate, low response latency and the ability to replace neural network components independently. A two-model ensemble architecture is suggested, which separates the spatial analysis (detection and pose estimation using the YOLOv8m-Pose network) and the temporal analysis (a two-layer bidirectional LSTM with additive attention) into independent models and supplements them with three system components: inter-frame IoU tracking, post-processing with sliding probability averaging and a threshold decision rule, and adaptive threshold calibration based on statistics collected directly from the operational facility. The architecture is implemented as a five-level pipeline with single-responsibility layers and a fixed tensor interface, which enables separate model training and scaling across the number of cameras. Experiments have shown that the convolutional detector achieves an mAP@0.5 of 0.809 and an F1 score of 0.802 on the test subset, while the full pipeline, when tested on a set of 25 video clips, achieves an accuracy of 0.77 at the incident level, recall of 0.92 and an F1 score of 0.84, with a false positive rate of ≈1 event per hour per camera and an average detection latency of 2.9 s. The resistance to synthetic noise, computation modes (TensorRT FP16 is optimal) and scalability have been analysed: a single NVIDIA RTX 4080 graphics card can process 14–16 cameras simultaneously. In terms of false positive rate, the proposed solution outperforms commercial and academic counterparts while offering comparable detection completeness.

Keywords: detection of drowners, video analytics, water safety, ensemble architecture, pose estimation, YOLOv8, bi-directional LSTM, attention mechanism, inter-frame tracking, adaptive threshold calibration, false alarm rate.

Введение

То, что ещё пять лет назад уверенно делал только живой наблюдатель, нейросетевые модели сегодня выполняют на сыром видеопотоке. Распознавание утопающего — пример из этого списка. Сложность задачи смещается из самих моделей в инженерное окружение: научить классификатор отличать утопление от плавания — лишь полдела. Вторая половина — это превратить такой классификатор в систему, у которой на каждую камеру приходится не более одного ложного срабатывания в час, реакция занимает считанные секунды, а замена любого нейросетевого блока не тянет за собой переобучения остальных. Настоящая статья — про эту вторую половину: про то, как из готовых моделей и нескольких хорошо подобранных инженерных компонентов собирается рабочая система видеоаналитики.

В литературе подходы к задаче распадаются по линии «один шаг — два шага». В одношаговой схеме [1] одна и та же сеть и находит пловца, и оценивает его состояние. Учить такую сеть удобно, но за это приходится платить большим объёмом размеченных видео и трудностью модификации: ни добавить новый класс поведения, ни обновить детектор без полного переобучения уже не получается. Двухшаговая схема [2] разносит ответственность между двумя независимыми моделями: пространственный анализ — где пловец в кадре и в какой он позе — и темпоральный — что с этой позой происходит во времени. Такое разделение лучше ложится на эксплуатационные требования и потому положено в основу настоящей работы.

Двухшаговую схему мы развиваем до уровня системно-инженерного решения. Поверх ансамбля «свёрточный детектор + темпоральный классификатор» добавлены три компонента, без которых лабораторный прототип не превращается в эксплуатируемую систему: межкадровый трекинг для ассоциации детекций по кадрам; постобработка с накоплением вероятностей в скользящем окне и пороговым решающим правилом; механизм адаптивной калибровки порога по статистике, набираемой непосредственно на объекте установки. По отдельности каждый из них хорошо известен, но именно их согласованное сочетание решает, окажется ли в итоге сильная сеть нейросетей в руках спасателя или останется презентационной демонстрацией.

Цель работы — построить и экспериментально проверить такое системное решение, в котором одновременно достижимы три эксплуатационные характеристики: не более одного ложного срабатывания на одну камеру в час, задержка обнаружения не более пяти секунд и возможность параллельной обработки нескольких камер на серийной игровой видеокарте. Структура статьи отражает эти задачи: системная постановка, двухмодельная ансамблевая архитектура из пяти уровней, алгоритмы трекинга, постобработки и калибровки, обучение детектора и испытания полного пайплайна, наконец, анализ устойчивости и сравнение с известными решениями.

Системная постановка задачи

На входе системы — видеопоток с одной или нескольких камер, накрывающих чашу бассейна. На выходе — поток событий тревоги; каждое из них представляет собой кортеж *(c, k, t, p)*: *c* — камера, *k* — идентификатор трекера, *t* — момент срабатывания, *p* — апостериорная вероятность утопления, на которой это срабатывание было принято. Между этими двумя концами лежит вычислительный конвейер, и формальные требования к нему естественно делятся на три группы: качество распознавания, эксплуатационные характеристики и системная гибкость.

Качество распознавания приходится оценивать на двух уровнях, и метрики эти неравноценны. На уровне отдельного объекта (одна последовательность поз, один кадр) работают классические точечные метрики — accuracy, precision, recall, F1 — с целевым порогом $F_1 \geq 0,90$. Их экспериментальная проверка ведётся в отдельной работе тех же авторов, посвящённой темпоральному классификатору, и в настоящей статье не дублируется. На уровне камеры в эксплуатационном режиме всё иначе: ни TN, ни доля правильно классифицированных кадров не имеют ясного смысла, поэтому используются две системные метрики — частота ложных срабатываний (FPR/час) и задержка обнаружения, отсчитываемая от размеченного экспертом начала инцидента до формирования события. Целевые значения здесь — не более одного события в час и не более пяти секунд. Эти числа диктует реальная эксплуатация: оператор, получающий по десять ложных тревог в час, перестаёт реагировать на них к концу первой смены; задержка более пяти секунд выводит реакцию системы из того окна, в котором у спасателя ещё есть шанс что-то сделать.

Эксплуатационные характеристики — это стоимость и масштаб. Целевая аппаратная платформа — серийная игровая видеокарта класса NVIDIA RTX 3060 или RTX 4080; один сервер обязан тянуть одновременно 4–8 камер 720p на частоте не ниже 15 Гц без потери качества. Сеть — обычная гигабитная локальная; видеопоток — по штатным RTSP, ONVIF или UVC.

Системная гибкость — это требования к раздельной жизни компонентов. Появилась новая версия YOLO [3] — должны быть в состоянии обновить только пространственную модель, не трогая темпоральную. Открыли необычный объект (детский бассейн, аквапарк, спортивный комплекс) — должны иметь возможность дообучить темпоральный классификатор на специфике этого объекта, ничего не меняя в детекторе. Подключили ещё один объект — должны откалибровать пороговое решающее правило на его статистике без какого бы то ни было переобучения нейросетей.

Эти три группы требований и определяют форму архитектуры, к описанию которой мы переходим.

Двухмодельная ансамблевая архитектура

Архитектура устроена как конвейер из пяти уровней (рисунок 1). Каждый уровень делает одно дело, передаёт результат следующему через стандартизованное тензорное представление и ничего не знает о внутреннем устройстве соседей. Эти три свойства — единичная ответственность, фиксированный интерфейс, отсутствие связности — не косметика; именно благодаря им уровни можно тестировать по отдельности, переписывать любой из них без оглядки на другие и распараллеливать работу по камерам, оставляя сами модели общими.

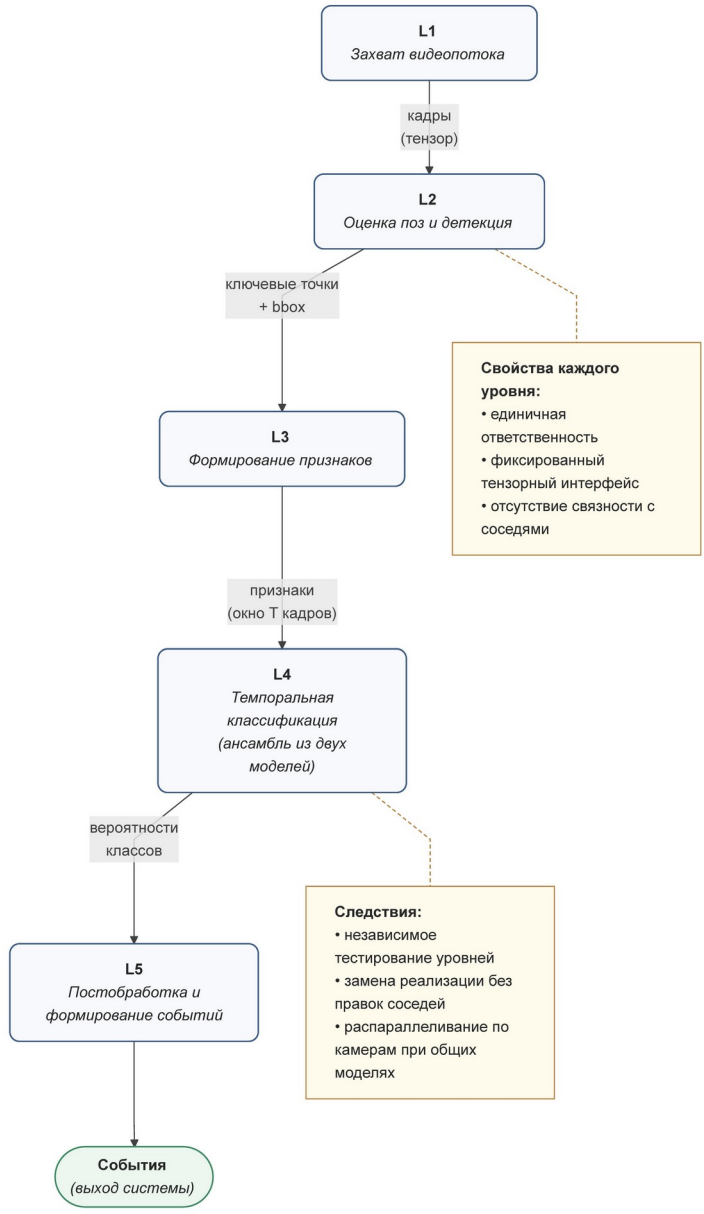

Рисунок 1 - Двухмодельная ансамблевая архитектура
DOI: https://doi.org/10.60797/IRJ.2026.170.57.1

Уровень захвата получает кадры от источника. Источников три: локальные USB-камеры через UVC, сетевые IP-камеры по RTSP/ONVIF/WebRTC [4] и видеофайлы при отладке. Программно слой держится на OpenCV [5] и асинхронном цикле задач FastAPI; кадры приводятся к единому разрешению (640 × 480 или 1280 × 720) и переводятся из BGR в RGB. Это самый дешёвый уровень — 1–3 мс на кадр, всё на CPU.

Уровень оценки поз и детекции — главное вычислительное звено. На нём работает YOLOv8m-Pose [6], для каждого кадра одновременно решающая обе задачи: находит людей и оценивает 17 ключевых точек тела по конфигурации COCO. На выходе – список объектов вида «ограничивающий прямоугольник + оценка уверенности + тензор ключевых точек 17 × 3». На том же уровне отрабатывают три эмпирически подобранных фильтра качества: отбрасываются объекты с числом валидных ключевых точек меньше пяти, объекты, занимающие меньше 3% площади кадра, и объекты, не проходящие проверку геометрической согласованности (например, плечи ниже бёдер — типичный артефакт оценки поз в воде). Фильтры не претендуют на оптимальность, но снимают существенную долю шумовых детекций ещё до темпорального анализа. По времени уровень самый дорогой: 12–18 мс на кадр на RTX 4080, около 80% всего бюджета обработки.

Уровень формирования признаков превращает поток детекций в поток 89-мерных векторов признаков на каждый объект (структура вектора и обоснование выбора компонентов даны в отдельной работе авторов). Параллельно с этим работает межкадровый трекинг, связывающий детекции одного и того же пловца в соседних кадрах в общий трекер; алгоритм описан в Разделе 3. Каждый трекер хранит буфер из 30 последних векторов признаков, готовый к подаче в темпоральную модель. CPU, 0,3–0,8 мс на кадр.

Уровень темпоральной классификации — двухслойная двунаправленная LSTM с аддитивным вниманием, обученная разделять три класса состояний пловца: «активное плавание», «пассивное пребывание в воде»,

«утопление». Структурно это компактная рекуррентная сеть: последовательность из 30 векторов признаков размерности 89 обрабатывается двумя двунаправленными слоями LSTM по 128 ячеек в каждом направлении (что даёт 256-мерное контекстное представление на каждом временном шаге), поверх которых аддитивный механизм внимания типа Bahdanau взвешивает 30 временных шагов и сворачивает их в единый дескриптор, акцентируя наиболее информативные моменты последовательности; полносвязный слой с softmax преобразует этот дескриптор в распределение по трём классам, а dropout между рекуррентными слоями отвечает за регуляризацию. Когда буфер трекера набирает 30 кадров, его содержимое подаётся на вход модели. Вероятность класса «утопление» уходит в скользящее окно истории длиной 90 значений, около трёх секунд. Темпоральный классификатор на два порядка легче пространственного детектора — порядка 636 тыс. обучаемых параметров против ~25 млн у YOLOv8m [7]. Соотношение ровно такое, какого и следует ожидать: тяжёлая обработка пикселей сидит в свёрточной модели, а темпоральная работает уже над выделенным позным описанием. Время инференса на одно окно укладывается в 2 мс на RTX 4080.

Уровень постобработки превращает поток вероятностей в поток событий тревоги. Здесь работают три этапа: накопление вероятностей в скользящем окне, усреднение и пороговое решающее правило с периодом «тишины» после срабатывания. Подробности — в Разделе 4. Снова CPU, 0,2–0,5 мс на кадр.

Распределение нагрузки между уровнями для типовой камеры 1280 × 720 на RTX 4080 сведено в таблицу 1.

Таблица 1 - Распределение нагрузки между уровнями архитектуры (камера 1280 × 720, GPU RTX 4080)

DOI: https://doi.org/10.60797/IRJ.2026.170.57.2

Уровень	Функция	Время на кадр, мс	Ресурс
1. Захват	Получение и нормализация кадра	1–3	CPU
2. Детекция и поза	YOLOv8m-Pose	12–18	GPU
3. Признаки	Формирование вектора, трекинг	0,3–0,8	CPU
4. Темп. классификатор	BiLSTM + Attention	1,5–2,5	GPU
5. Постобработка	Сглаживание, формирование события	0,2–0,5	CPU

Из таблицы виден главный инженерный факт: время обработки кадра практически целиком расходуется на втором уровне. Любая работа над производительностью имеет смысл прежде всего там — к этому мы вернёмся в Разделе 6 при анализе режимов вычислений TensorRT [8].

Архитектура из пяти уровней даёт три практических преимущества. Первое: пространственная и темпоральная модели обучаются раздельно. Темпоральная видит не пиксели, а уже извлечённые векторы признаков, и поэтому её обучение требует на порядок меньшего размеченного корпуса видео, чем у одношаговой схемы. Второе: модели заменяются независимо. Появилась новая версия YOLO — переучиваем уровень 2, темпоральный классификатор остаётся прежним; специфика бассейна изменилась — дообучаем уровень 4, детектор не трогаем. Третье: архитектура естественно масштабируется по числу камер. Каждая камера обрабатывается отдельным потоком, а тяжёлые нейросетевые модели переиспользуются между потоками через общую очередь задач, без дублирования GPU-памяти.

Алгоритм межкадрового трекинга

Темпоральный анализ имеет смысл только тогда, когда детекции одного и того же пловца в соседних кадрах оказываются привязаны друг к другу — то есть собираются в траекторию. Это классическая задача multi-object tracking; для неё есть длинный ряд готовых алгоритмов: от простейшего IoU-трекинга в SORT до многокомпонентных решений с фильтром Калмана [9] и сопоставлением по признакам (DeepSORT, ByteTrack, BoT-SORT) [10]. Прежде чем выбирать алгоритм, разумно спросить себя: насколько в нашей постановке вообще присутствуют те трудности, ради которых сложные алгоритмы и создавались.

Бассейн как сцена для трекинга — относительно лёгкая задача. Одновременно в кадре редко бывает больше восьми– десяти пловцов. Перекрытия — короткие: нырнувший за чужую спину пловец появляется обратно через секунду–три. Купальные принадлежности обычно одноцветны, и градиентное сопоставление по внешнему виду в этих условиях малопродуктивно. Скорости движения умеренные, резкие телепортации возможны разве что при прыжке с трамплина. В таком режиме DeepSORT и ByteTrack дают почти неразличимое улучшение качества ассоциации при существенно более высокой вычислительной стоимости.

Поэтому в качестве алгоритма принят простейший IoU-трекинг. Логика следующая: для каждого нового обнаружения в текущем кадре считаются значения IoU с прямоугольниками всех активных трекеров; обнаружение прикрепляется к тому трекеру, у которого этот показатель максимален и не ниже 0,3; если ни один трекер не подошёл — заводится новый с уникальным идентификатором. IoU для двух прямоугольников A и B определяется стандартно:

$$\mathrm{IoU}(A, B) = \frac{|A \cap B|}{|A \cup B|} \quad (1)$$

Значение лежит в [0, 1]: IoU = 0 — пересечения нет вовсе, IoU = 1 — прямоугольники совпадают. Порог 0,3 — компромисс: с одной стороны, он достаточно низкий, чтобы выдержать небольшие смещения объекта между кадрами и шум детектора, с другой — достаточно высокий, чтобы в плотных сценах не путать соседних пловцов.

Трекер закрывается, если в течение пяти секунд (150 кадров при 30 Гц) на него не приходит обновлений. Это значение покрывает основную массу перекрытий: если пловец на это время скрылся за другим, после выхода из перекрытия его трекер восстанавливается. Если же скрытие длится дольше, трекер закрывается окончательно, а после возврата объекта в кадр заводится новый — со следствием, что в течение примерно секунды (пока буфер не наберётся снова) темпоральный анализ для этого объекта не работает. На практике это редкая и не критичная ситуация: устойчивая неподвижность за чужой спиной маловероятна как сценарий начала утопления.

Каждый трекер несёт три буфера и метки времени: последний ограничивающий прямоугольник, метку последнего обновления, 30 последних векторов признаков (вход в темпоральную модель), 90 последних значений вероятности класса «утопление» (для скользящего усреднения — см. Раздел 4) и копию ключевых точек предыдущего кадра, нужную для подсчёта компоненты межкадровых скоростей в векторе признаков.

Сравнение IoU-трекинга с альтернативами проведено на собранном тестовом наборе из 25 видеороликов (142 траектории, 7 ч непрерывной съёмки); результаты — в таблице 2.

Таблица 2 - Сравнение алгоритмов межкадрового трекинга на тестовом наборе из 25 видеороликов

DOI: https://doi.org/10.60797/IRJ.2026.170.57.3

Алгоритм	MOTA, %	IDF1, %	ID switches	Время, мс/кадр
IoU-трекинг (применяется)	83,7	79,2	6	0,3
SORT	84,2	79,8	6	0,5
DeepSORT	85,6	82,1	4	8,4
ByteTrack	85,9	82,5	4	6,2
BoT-SORT	86,1	82,8	3	11,7

Картина типичная: сложные алгоритмы выигрывают в MOTA [11] 1,5–2,4 пункта и сокращают число подмен идентификаторов примерно вдвое — с шести до трёх за семь часов записи. В коммерческой системе, где каждая такая подмена потенциально оборачивается пропуском утопления, это может оказаться важным. Но в нашей конструкции одиночная подмена ID не превращается в ложноотрицательное срабатывание: вероятностное окно длиной 90 кадров инерционно, и одна перепутанная траектория не успевает вытолкнуть из буфера подозрительные значения. На этом фоне экономия 6–11 мс на кадр — а это треть бюджета всего пайплайна — оказывается важнее.

Постобработка и адаптивная калибровка порога

Сразу превращать выход темпорального классификатора в сигнал тревоги нельзя. Во-первых, на любом отдельном временном окне он неизбежно зашумлён — ошибки оценки поз, ошибки классификации, артефакты сжатия дают случайные всплески вероятности «утопления» там, где никакого инцидента нет и быть не может. Во-вторых, реакция на единичный пик превратит систему в безостановочный генератор ложных тревог — ровно ту проблему, которую мы и пытаемся решить.

Решение — трёхэтапная постобработка. На первом этапе каждый трекер ведёт скользящее окно из 90 последних значений вероятности класса «утопление» (около трёх секунд при 30 Гц): новое значение от темпоральной модели поступает в буфер, самое старое выпадает. На втором этапе из буфера берётся арифметическое среднее:

$$\bar{p} = \frac{1}{N} \sum_{i=1}^{N} p_i, \tag{2}$$

где N — текущая длина буфера. Усреднение даёт устойчивую оценку поведения и гасит одиночные выбросы. Третий этап — пороговое решающее правило: усреднённая вероятность сравнивается с T порогом (по умолчанию $T = 0{,}5$), и при превышении формируется событие — запись в базу и уведомление в клиентское приложение через WebSocket [12].

К этим трём этапам добавляется период «тишины» после срабатывания. Следующее событие на том же трекере не может быть сформировано раньше, чем через 30 секунд, либо до явного подтверждения или отклонения предыдущего оператором. Без такого блокировщика длящийся инцидент превращается в десяток повторов одной и той же тревоги, забивающих оператору экран.

Прежде чем остановиться на скользящем среднем, мы перепробовали несколько альтернатив: экспоненциальное скользящее среднее с параметром α, медианное окно вместо арифметического, цепи Маркова [13] с тремя состояниями и обучаемыми переходами, байесовский фильтр с моделью наблюдения. На наших тестовых данных все они дают результаты в пределах ±0,3 пункта по *F1* при существенно более сложной реализации. Бритва Оккама в данном случае указала на простое арифметическое среднее.

Главный параметр в этой схеме — порог T. Его снижение делает систему чувствительнее, но за счёт роста числа ложных тревог; повышение — наоборот. Дефолтное 0,5 даёт лучший *F1*-баланс на нашем тестовом наборе, но

реальный оптимум на конкретном объекте сильно отличается: бассейн с детьми требует чувствительности повыше, спортивный комплекс с подготовленными пловцами – пониже.

Чтобы не возлагать подбор T на эксперта, в систему встроена адаптивная калибровка. На этапе ввода в эксплуатацию администратор включает калибровочный режим длительностью 24–72 часа. В этом режиме система обрабатывает поток обычным образом, но события не формирует — вместо этого она собирает по всем трекерам статистику усреднённых вероятностей класса «утопление». По окончании периода вычисляется эмпирическое распределение этих вероятностей в нормальной (без инцидентов) эксплуатации, и поверх него ищется рекомендованное значение порога

$$T^* = \arg\min_T \left[w_{\text{FN}} \cdot P_{\text{FN}}(T) + w_{\text{FP}} \cdot P_{\text{FP}}(T) \right], \quad (3)$$

где $P_{FN}(T)$ — вероятность пропуска реального инцидента при пороге T (оценивается на отдельной верификационной выборке с размеченными инцидентами), $P_{Fp}(T)$ – вероятность ложного срабатывания при том же пороге (оценивается по эмпирическому распределению, набранному в калибровочный период), а W_{FN}, W_{FP} — веса видов ошибок, которые задаёт администратор. Соотношение весов отражает приоритеты объекта: высокое W_{FN} / W_{FP} — приоритет максимальной чувствительности, низкое — минимальной частоты ложных тревог. В одном из обследованных бассейнов оптимум оказался $T^* = 0,52$, и при этом значении система давала желаемое ≈1 событие в час.

Дополнительно алгоритм может подбирать и длину окна сглаживания, опираясь на автокорреляционную функцию вероятностей: медленный спад автокорреляции говорит, что наблюдения сильно коррелированы во времени, и тогда выгоднее короткие окна; быстрый — длинные. Но на практике у большинства обследованных бассейнов оптимальное окно оказывается в одном и том же диапазоне 60–90 кадров, поэтому в текущей версии длина окна фиксирована на 90.

Интерфейс калибровки оформлен отдельным административным модулем в клиентском приложении, доступным ролям «admin». Он показывает гистограмму распределения вероятностей по итогам калибровочного периода, рекомендованное T*, кривую Парето «частота ложных срабатываний — полнота» и таблицу метрик при разных значениях порога. Применить рекомендацию – одно нажатие; повторная калибровка может проводиться регулярно, например раз в квартал.

Над постобработкой надстроена логика обработки одновременных событий и эскалации. Если в течение 15 секунд на той же камере уже зарегистрировано неотработанное событие, новое событие группируется с предыдущим (через общее поле group_id в БД) — это нужно при массовых инцидентах, когда в опасности оказываются сразу несколько человек. Эскалация трёхуровневая. Первый уровень — оповещение оператора через клиентское приложение и звуковой сигнал; второй (если оператор не подтвердил и не отклонил событие за 30 секунд) – push-уведомление администратору объекта; третий (90 секунд без реакции) — автоматическое сообщение дежурной службе спасения через настроенный канал (SMS, push, e-mail). Параметры эскалации настраиваются администратором при развёртывании.

Экспериментальное исследование

Свёрточный детектор обучался на публичном наборе «Drowning Detection and Prevention in Swimming Pools» с платформы Roboflow Universe — 5 207 размеченных изображений с двумя классами «swimming» и «drowning», разделённых на обучающую, валидационную и тестовую подвыборки в пропорции 70 : 15 : 15 со стратификацией по классам. Для проверки полного пайплайна и измерения системных метрик дополнительно собран отдельный тестовый набор из 25 видеороликов общей длительностью 1 ч 22 мин, не пересекающийся с обучающими данными ни по источнику, ни по содержанию.

Стартовой точкой обучения детектора послужили предобученные веса yolov8m.pt с COCO. Гиперпараметры подобраны несколькими предварительными запусками: 150 эпох с ранней остановкой по терпению 30, мини-батч 16, размер изображения 640 × 640, оптимизатор AdamW [14] с начальным lr = 0,001 и косинусным расписанием, разогрев 5 эпох, mosaic-аугментация (1,0), MixUp (0,1), copy-paste (0,1), HSV-аугментация (0,015 / 0,7 / 0,4), повороты ±10°, сдвиги ±10%, масштаб 0,5–1,5, горизонтальное отражение с вероятностью 0,5. Полное обучение занимает 11 ч 40 мин на одной NVIDIA RTX 4080.

Метрики качества детектора на валидационной и тестовой подвыборках сведены в таблицу 3.

Таблица 3 - Метрики свёрточного детектора YOLOv8m на валидационной и тестовой подвыборках

DOI: https://doi.org/10.60797/IRJ.2026.170.57.4

Метрика	Валидация	Тест
mAP@0.5 (общая)	0,827	0,809
mAP@0.5:0.95 (общая)	0,571	0,548
Precision (общая)	0,841	0,827
Recall (общая)	0,793	0,778
F1 (общая)	0,816	0,802
Precision (swimming)	0,889	0,872
Recall (swimming)	0,851	0,839
Precision (drowning)	0,776	0,754

Метрика	Валидация	Тест
Recall (drowning)	0,734	0,723

Падение метрик на тесте относительно валидации мизерное, существенного переобучения нет. Бóльшая часть ошибок приходится на класс «drowning» (precision 0,754, recall 0,723) — это естественно: примеров утопления меньше, и разнообразие визуальных проявлений у них шире. Часть таких ошибок снимается уже на следующем уровне: темпоральный анализ умеет отличать единичную необычную позу от устойчивого паттерна утопления, и в системных метриках вклад детектора частично «съедается».

Сравнение разных моделей семейства YOLO мы тоже проверили. Меньшие версии (YOLOv8n, YOLOv8s) дают инференс за 5,7 и 8,2 мс/кадр соответственно, но проигрывают 6–9 пунктов mAP@0.5. Большие версии (YOLOv8l, YOLOv8x) [15] выигрывают 1–3 пункта точности, но обработка одного кадра удлиняется в полтора-два раза. YOLOv8m остаётся компромиссным выбором для целевой аппаратной платформы.

Полный пайплайн (детекция → оценка поз → темпоральная классификация → постобработка) тестировался на отдельном наборе из 25 видеороликов: 18 нормальных сцен плавания общей длительностью 7 ч и 7 видео с инцидентами утопления, отснятыми спасателями в виде тренировочных имитаций. На этом наборе зафиксировано: 23 истинно положительных срабатывания из 25 размеченных инцидентов, 2 пропуска (FN), 7 ложных срабатываний (FP) за 7 часов нормального плавания. Системные метрики уровня инцидентов: precision = 0,77, recall = 0,92, F1 = 0,84; частота ложных срабатываний ≈1 событие в час на одну камеру; средняя задержка обнаружения — 2,9 секунды (от размеченного экспертом начала инцидента до момента, когда система выдала событие). Целевые системные характеристики из Раздела 1 (FP/час ≤ 1, задержка ≤ 5 с) выполнены. Следует оговорить, что тестовый набор из 25 роликов невелик по объёму – это прямое следствие специфики задачи, в которой реальные инциденты крайне редки, а их этичная и безопасная имитация трудозатратна; путь к расширению валидационной выборки намечен в Заключении.

Чтобы изложение оставалось самодостаточным, напомним ключевые показатели темпорального классификатора, подробно описанного в отдельной работе авторов: на трёхклассовой задаче («активное плавание» / «пассивное пребывание в воде» / «утопление») он даёт точность (accuracy) 0,920 и F1 = 0,924 по ключевому классу «утопление». Системная F1 полного пайплайна оказывается ниже этой точечной оценки. Это закономерное снижение: к ошибкам классификатора добавляются ошибки детектора (часть пловцов теряется в бликах и волнении) и эффекты постобработки (короткие пиковые значения вероятности не успевают пробиться через трёхсекундное усреднение). Подобное расхождение между метриками компонента и метриками системы — типичное свойство многоуровневых пайплайнов и в нашей конфигурации удерживается на разумном уровне.

Систему, которая будет работать в реальном бассейне круглосуточно, имеет смысл проверять не только в стерильных условиях. Шум матрицы, артефакты сжатия, дрожание камеры, неудачно выставленная экспозиция — всё это нормальный фон видеопотока, и качество распознавания на нём не должно проваливаться. Для проверки тестовый набор был пропущен через серию синтетических искажений; результаты — в таблице 4.

Таблица 4 - Устойчивость пайплайна к синтетическим помехам

DOI: https://doi.org/10.60797/IRJ.2026.170.57.5

Помеха	Уровень	Accuracy	F1 (drowning)
–	–	0,920	0,924
Гауссов шум σ	0,03	0,913	0,915
Гауссов шум σ	0,10	0,878	0,874
Снижение разрешения	×0,5	0,908	0,908
Снижение разрешения	×0,25	0,871	0,860
JPEG quality	50	0,912	0,912
JPEG quality	25	0,886	0,879
Дрожание камеры	сильное	0,900	0,898
Затемнение	−50 %	0,889	0,886
Переэкспонирование	+50 %	0,882	0,877
Цветовой сдвиг	сильный	0,907	0,905

Слабые и умеренные искажения (шум до σ = 0,03, снижение разрешения до ×0,5, JPEG quality 50, изменения экспозиции до 25%) почти не задевают качество — точность падает максимум на пункт. Существенная деградация — больше трёх пунктов — наступает только при экстремальных уровнях, которые на нормальном объекте устраняются адекватным выбором и обслуживанием оборудования. Главный практический вывод связан с разрешением: пайплайн уверенно отрабатывает даже на 320 × 180 (то есть на ×0,25 от 1280 × 720), и это означает, что для развёртывания достаточно сравнительно недорогих 720p-камер, а в ряде случаев и 480p — что заметно снижает капитальную стоимость объекта.

Аналогичный анализ зависимости качества от условий освещения и характера сцены— в таблице 5.

Таблица 5 - Зависимость качества от условий освещения и сцены

DOI: https://doi.org/10.60797/IRJ.2026.170.57.6

Условие	F1 (drowning)	FPR/час
Идеальное (естественное освещение)	0,936	0,3
Хорошее (искусственное освещение)	0,924	1,0
Среднее (мягкое освещение)	0,912	1,5
Сложное (низкое освещение, отражения)	0,875	2,8
Очень сложное (сумерки)	0,812	5,1
Прямой солнечный свет, бликование	0,879	3,3
После дождя, капли на объективе	0,840	4,2
Зимний бассейн, конденсат	0,811	4,8

В типовых условиях общественного бассейна — хорошее или среднее освещение – система работает уверенно. Существенный провал начинается на сценариях, которые в реальном проекте устраняются грамотной светотехникой и регулярным обслуживанием: сумерки, прямой солнечный свет с бликованием, конденсат на остеклении. Для сумерек рекомендация — добавить инфракрасный канал; для бликующего солнечного света — поляризационные фильтры на объективы.

Режимы вычислений и масштабируемость

Время обработки одного кадра – главное, что определяет, сколько камер реально вешается на один сервер. Уровень 2 (YOLOv8m-Pose) занимает порядка 80 % всего бюджета, и работа над ним даёт максимальный выигрыш в пропускной способности. Простейшая реализация на PyTorch FP32 [16] — самая медленная, но самая удобная при разработке. Переход на смешанную точность FP16/AMP ускоряет инференс почти вдвое и стоит этого: точность падает на 0,2 пункта. Скомпилированная через TensorRT модель добавляет ещё в 1,5–2 раза. INT8-квантование разгоняет всё максимально, но забирает уже ~1,8 пункта точности — для задачи безопасности это пограничный размен.

Таблица 6 - Производительность пайплайна в различных режимах вычислений (GPU NVIDIA RTX 4080)

DOI: https://doi.org/10.60797/IRJ.2026.170.57.7

Режим	Время на кадр, мс	Снижение точности, п.п.	Применимость
PyTorch FP32 (baseline)	16,8	0,0	Разработка, отладка
PyTorch FP16 (AMP)	9,1	0,2	Стандартная продуктивная среда
TensorRT FP32	11,3	0,0	Высокоточные применения
TensorRT FP16	5,4	0,3	Оптимально для RTX-серии
TensorRT INT8	3,1	1,8	Максимальная производительность
ONNX Runtime CPU	412,7	0,0	Резервный режим без GPU

Оптимальным режимом признан TensorRT FP16: ускорение в 3,1 раза при потере точности всего в 0,3 пункта. Однако эмпирические испытания масштабируемости делались на референсной конфигурации PyTorch FP16 (AMP) — именно по ней в таблице 7 указано максимальное число одновременно обрабатываемых камер. Оценка построена в два уровня. «Наивная» — простое деление пропускной способности GPU (1000 / время кадра) на требуемую частоту обработки одной камеры, без учёта какого-либо параллелизма. «Эмпирическая» — фактически наблюдаемое на стенде число одновременно обрабатываемых камер при включённом батчинге и конвейеризации передачи данных между CPU и GPU. Зазор между двумя оценками — это и есть выигрыш от того, что тяжёлая нейросетевая модель амортизирует постоянные накладные расходы на пакетной обработке. Прямые эмпирические измерения сделаны на RTX 4080; для RTX 3060 и Jetson AGX Orin это прогнозные оценки, полученные масштабированием по числу CUDA-ядер и

пропускной способности памяти. При переходе на TensorRT FP16 ожидается дополнительное ускорение в 1,5–1,7 раза и пропорциональный рост числа поддерживаемых камер — примерно до 22–25 на RTX 4080.

Таблица 7 - Максимальное число одновременно обрабатываемых камер на разных платформах в режиме PyTorch FP16 (AMP)

DOI: https://doi.org/10.60797/IRJ.2026.170.57.8

Платформа	Время/кадр, мс	Наивно при 30 FPS	Эмпирически при 30 FPS	Эмпирически при 15 FPS
NVIDIA RTX 4080	9,1	3	14–16	28–32
NVIDIA RTX 3060 (оценка)	~13,5	2	~8–10	~16–20
NVIDIA Jetson AGX Orin (оценка)	~42	0–1	~3–4	~6–8

Из этого вытекает простая практическая раскладка. Для типового бассейна (4–8 камер) достаточно сервера на NVIDIA RTX 3060 в режиме PyTorch FP16 — это решение за 100–150 тыс. руб. Для крупного аквапарка (12–16 камер) уместна RTX 4080; если нужен запас по числу камер — переходим на TensorRT FP16 и получаем дополнительный кратный прирост (см. таблицу 6). Для распределённого развёртывания, где обработка ведётся прямо у камеры, годится встраиваемая платформа Jetson AGX Orin [17], на которой пайплайн помещается в 3–4 камеры при 30 FPS.

Сетевая нагрузка скромная: одна камера 720p H.264 даёт 2–4 Мбит/с, шестнадцать камер агрегированно — около 64 Мбит/с, что свободно укладывается в типовую гигабитную локальную сеть. Поток исходящих результатов (события, скриншоты, телеметрия) в обычной эксплуатации не превышает 1 Мбит/с.

Сравнение с существующими решениями

Чтобы понять, где предложенная архитектура находится относительно известных решений, сравним её с тремя коммерческими системами (Poseidon, AngelEye, MyLifeguard) и двумя академическими работами [1], [2] по характеристикам, опубликованным в открытых источниках. Сводка — в таблице 8. Сразу оговорка: методики оценки у разных авторов несовпадающие, поэтому сравнение неизбежно носит ориентировочный характер. У коммерческих и академических аналогов цифры — на уровне системы целиком, как заявлено разработчиками; у нашей архитектуры — на 25-роликовом тестовом наборе из Раздела 5. Прямое экспериментальное сравнение всех систем на едином тестовом наборе было бы, разумеется, доказательнее, однако в условиях закрытости коммерческих решений — недоступности их моделей, исходного кода и протоколов измерений — оно практически нереализуемо; поэтому приведённые в таблице 8 значения по аналогам следует трактовать как ориентир, а не как результат прямого измерения в единых условиях.

Таблица 8 - Сравнение предложенной архитектуры с существующими решениями

DOI: https://doi.org/10.60797/IRJ.2026.170.57.9

Характеристика	Poseidon	AngelEye	MyLifeguard	[1]	[2]	Предлагаемая
Accuracy (system), %	–	–	92	95	93	–
F1 (events), %	–	–	–	–	–	84
Recall (drowning), %	–	–	–	94	92	92
Precision, %	–	–	–	–	–	77
FP/час	5–8	4–6	6–10	–	≈2	≈1,0
Тип камер	подв.+	подв.	надв.	надв.	надв.	надв.
Открытый код	нет	нет	нет	нет	част.	да
Многокамерность	да	да	огр.	–	–	да
Темп. анализ	–	–	–	пост-обр.	Transformer	BiLSTM-Att
Аппарат. требования	спец.	спец.	сервер	GPU	GPU	GPU

По полноте обнаружения утопающих (recall) наша архитектура попадает в диапазон лучших академических работ [1], [2]. По частоте ложных срабатываний картина резче: ≈1 событие в час против 5–10 у коммерческих систем и ≈2 у [2]. Этот разрыв — не одна выдающаяся компонента, а сумма трёх: трёхуровневая фильтрация детекций на втором уровне, темпоральное сглаживание по 90-кадровому окну на пятом и адаптивная калибровка порога под конкретный объект. Прямое сравнение по строке «Accuracy (system)» невозможно: для нашей архитектуры accuracy в классическом смысле просто не определена — события возникают как точки во времени, а не как метки на размеченных кадрах. Вместо неё в таблице стоит event-level F1 = 0,84, которая методологически не совпадает с accuracy работ [1], [2]. По прочим характеристикам — открытый код, отсутствие специализированного железа, полноценный веб-интерфейс оператора – предложенная архитектура тоже выигрывает.

Ограничений три, и о них честнее сказать прямо. Первое – отсутствие поддержки подводных камер; добавление планируется через отдельную модель оценки поз, обученную на подводных изображениях с учётом преломления и цветовых искажений. Второе — ограниченная устойчивость в ночных условиях; требуется дополнительная инфракрасная подсветка. Третье — рекомендованное расстояние от камеры до пловца не более 8–10 метров: дальше начинает просаживаться качество оценки поз.

Заключение

В работе предложена и проверена системно-архитектурная модель автоматического обнаружения утопающих, в которой ансамбль из свёрточного детектора и темпорального классификатора дополнен инженерными компонентами уровня системы — межкадровым трекингом, постобработкой и адаптивной калибровкой порога. Всё, что лежит между нейросетевым ансамблем и оператором, в этой работе и было основным предметом исследования.

Архитектура устроена как пятиуровневый конвейер: захват, оценка поз и детекция, формирование признаков, темпоральная классификация, постобработка. Главная инженерная идея — раздельность пространственного и темпорального анализа: пиксельную работу делает YOLOv8m / YOLOv8m-Pose, поведенческий анализ — двунаправленная LSTM с аддитивным вниманием, обучаемая отдельно. Из такого разделения следуют три полезных свойства: модели обновляются независимо, темпоральная часть требует на порядок меньшего размеченного корпуса видео, а архитектура естественно масштабируется по числу камер.

Межкадровый трекинг — простой IoU-трекинг с порогом 0,3 и временем жизни трекера 5 секунд. По MOTA он отстаёт от DeepSORT/ByteTrack/BoT-SORT на 1,5–2,4 пункта, но выигрывает в скорости в 20–40 раз. В нашей конструкции это оправдано: подмена идентификатора не превращается в ложноотрицательное срабатывание, поскольку 90-кадровый буфер усреднения на пятом уровне инерционен и поглощает одиночные сбои трекинга.

Постобработка трёхэтапная: накопление вероятностей класса «утопление» в скользящем окне 90 кадров, их арифметическое усреднение и пороговое решающее правило с 30-секундной «тишиной» после срабатывания. Поверх работает адаптивная калибровка порога T^*, минимизирующего взвешенную сумму вероятностей ошибок FN и FP по эмпирическому распределению, набираемому в калибровочный период. Этот механизм позволяет настроить систему под конкретный объект, не переобучая нейросети.

Свёрточный детектор YOLOv8m, обученный на 5 207 размеченных изображениях, на тестовой подвыборке даёт mAP@0.5 = 0,809 и F1 = 0,802. На отдельном наборе из 25 видеороликов полный пайплайн зафиксировал 23 истинно положительных срабатывания и 2 пропуска при 7 ложных срабатываниях за 7 часов нормального плавания. Это соответствует системным precision = 0,77, recall = 0,92, F1 = 0,84, частоте ложных срабатываний ≈1 событие в час на камеру и средней задержке обнаружения 2,9 секунды. Целевые системные характеристики из постановки задачи – FP/час ≤ 1 и задержка ≤ 5 с – выполнены.

Испытания на устойчивость показали, что пайплайн уверенно держит умеренные синтетические помехи: при гауссовом шуме $\sigma \le 0{,}03$, снижении разрешения до ×0,5, JPEG quality 50 и изменениях экспозиции до ±25% падение точности не выходит за один процентный пункт. Особенно полезен результат по разрешению: пайплайн работает даже при 320 × 180, и это означает, что для развёртывания достаточно сравнительно недорогих 720p- и 480p-камер. Анализ режимов вычислений выделил TensorRT FP16: ускорение в 3,1 раза при потере точности всего в 0,3 пункта. По эмпирическим испытаниям в режиме PyTorch FP16 одна RTX 4080 тянет 14–16 одновременных камер при 30 FPS; при переходе на TensorRT FP16 пропускная способность кратно растёт.

Сравнение с существующими решениями выглядит так: по recall (≈92 %) наша архитектура попадает в диапазон лучших академических работ; по частоте ложных срабатываний — ≈1 событие в час против 5–10 у коммерческих аналогов и ≈2 у [2]. Этот разрыв обеспечивает не одна выдающаяся компонента, а согласованная работа ансамбля моделей и инженерной обвязки уровня системы — что и составляло предмет этой работы.

Дальнейшее развитие архитектуры мы видим по четырём направлениям. Первое – поддержка подводных камер: для этого нужна отдельная модель оценки поз, обученная на подводных изображениях с учётом преломления и цветовых искажений. Второе — добавление тепловизионного канала как параллельного источника, который выручит в сумеречных и ночных условиях, где видимый канал проседает. Третье — переход к более серьёзному алгоритму трекинга, например на основе фильтра Калмана, для сценариев с большим числом одновременных пловцов и значительной долей перекрытий: это особенно актуально для аквапарков и водных аттракционов. Четвёртое — расширение и диверсификация валидационной выборки: использованный тестовый набор из 25 роликов ограничен по объёму в силу самой природы задачи (реальные инциденты редки, а их безопасная имитация трудозатратна), и наиболее ценным его пополнением станут записи, накопленные на действующих объектах эксплуатации, — они позволят оценить систему в реальном многообразии сцен, оборудования и режимов освещения.

Конфликт интересов

Не указан.

Рецензия

Рудой Е.М., ООО «ГК «Иннотех», Москва Российская Федерация

DOI: https://doi.org/10.60797/IRJ.2026.170.57.10

Conflict of Interest

None declared.

Review

Rudoi E.M., Innotech Group LLC, Moscow Russian Federation

DOI: https://doi.org/10.60797/IRJ.2026.170.57.10

Список литературы на английском языке / References in English

1. Hassan E. Review: Mask R-CNN Models / E. Hassan, N. El-Rashidy, F. Talaa // Nile Journal of Communication and Computer Science. — 2022. — DOI: 10.21608/njccs.2022.280047.

2. Guo Z. MSFT-YOLO: Improved YOLOv5 Based on Transformer for Detecting Defects of Steel Surface / Z. Guo, C. Wang, G. Yang [et al.] // Sensors (Basel). — 2022. — Vol. 22, № 9. — P. 3467. — DOI: 10.3390/s22093467.

3. Mao M. YOLO Object Detection for Real-Time Fabric Defect Inspection in the Textile Industry: A Review of YOLOv1 to YOLOv11 / M. Mao, M. Hong // Sensors (Basel). — 2025. — Vol. 25, № 7. — P. 2270. — DOI: 10.3390/s25072270.

4. Sadman R. Real-Time Detection Performance of RTSP vs. USB Cameras for UGVs / R. Sadman, S. Safi, J. Mahe [et al.] // WRC Symposium on Advanced Robotics and Automation (WRC SARA). — 2025. — P. 119–126. — DOI: 10.1109/wrcsara68202.2025.11194983.

5. Sinha E. OpenCV for Computer Vision Applications / E. Sinha, A. Kumar, A. Tyagi // International Journal for Multidisciplinary Research. — 2025. — Vol. 7, № 3. — DOI: 10.36948/ijfmr.2025.v07i03.44280.

6. Meng Z. YOLOv10-pose and YOLOv9-pose: Real-time strawberry stalk pose detection models / Z. Meng, X. Du, R. Sapkota [et al.] // Computers in Industry. — 2025. — Vol. 164. — P. 104231. — DOI: 10.1016/j.compind.2024.104231.

7. Farooq J. An improved YOLOv8 for foreign object debris detection with optimized architecture for small objects / J. Farooq, M. Muaz, K. Jadoon [et al.] // Multimedia Tools and Applications. — 2023. — Vol. 83. — P. 60921–60947. — DOI: 10.1007/s11042-023-17838-w.

8. Jeong E. TensorRT-Based Framework and Optimization Methodology for Deep Learning Inference on Jetson Boards / E. Jeong, J. Kim, S. Ha // ACM Transactions on Embedded Computing Systems. — 2022. — Vol. 21, № 6. — P. 1–26. — DOI: 10.1145/3508391.

9. Rutan S. Adaptive Kalman filtering used to compensate for model errors in multicomponent methods / S. Rutan, S. Brown // Analytica Chimica Acta. — 1984. — Vol. 160. — P. 99–119. — DOI: 10.1016/s0003-2670(84)84512-2.

10. The V. A Comparative Performance Analysis of Deep Learning-Based Motorcycle Tracking Models at Urban Intersections: A Vietnamese Case Study / V. The, T. De, H. Dinh // RIVF International Conference on Computing and Communication Technologies (RIVF). — 2025. — P. 304–309. — DOI: 10.1109/rivf68649.2025.11365250.

11. Aharon N. BoT-SORT: Robust Associations Multi-Pedestrian Tracking / N. Aharon, R. Orfaig, B. Bobrovsky // arXiv. — 2022. — arXiv:2206.14651. — DOI: 10.48550/arxiv.2206.14651.

12. Fette I. The WebSocket Protocol / I. Fette, A. Melnikov // RFC 6455. — 2011. — P. 1–71. — DOI: 10.17487/rfc6455.

13. Vere-Jones D. Markov Chains / D. Vere-Jones // Nature. — 1972. — Vol. 236. — P. 291. — DOI: 10.1038/236291a0.

14. Zhou P. Towards Understanding Convergence and Generalization of AdamW / P. Zhou, X. Xie, Z. Lin // IEEE Transactions on Pattern Analysis and Machine Intelligence. — 2024. — Vol. 46, № 9. — P. 6486–6493. — DOI: 10.1109/tpami.2024.3382294.

15. Shaikh I. Enhancing sustainability in the production of palm oil: creative monitoring methods using YOLOv7 and YOLOv8 for effective plantation management / I. Shaikh, M. Akhtar, A. Aabid // Biotechnology Reports. — 2024. — Vol. 44. — P. e00853. — DOI: 10.1016/j.btre.2024.e00853.

16. Ríos J. Dynamically Adapting Floating-Point Precision to Accelerate Deep Neural Network Training / J. Ríos, A. Armejach, E. Petit [et al.] // 20th IEEE International Conference on Machine Learning and Applications (ICMLA). — 2021. — P. 980–987. — DOI: 10.1109/icmla52953.2021.00161.

17. Emin B. Digital Implementation of Chaotic Systems Using Nvidia Jetson AGX Orin and Custom DAC Converter / B. Emin, M. Yaz // ADBA Information Technology and Publishing Limited Company. — 2024. — DOI: 10.69882/adba.chf.2024075.