

Научная статья

УДК 004.415.2.031.43

<https://doi.org/10.35266/1999-7604-2024-4-3>

Нейросетевая система видеонаблюдения

Никита Евгеньевич Василенко¹✉, Наталья Александровна Медведева²

^{1,2}Сургутский государственный университет, Сургут, Россия

¹vasilenko_ne@edu.surgu.ru✉, <https://orcid.org/0009-0007-7880-1026>

²medvedeva_na@surgu.ru, <https://orcid.org/0009-0005-9379-718X>

Аннотация. Целью представленной работы является разработка и внедрение эффективной и доступной по цене автоматической системы видеонаблюдения, способной работать в режиме реального времени и интегрироваться с существующими системами видеонаблюдения. Представленный метод включает в себя анализ существующих на рынке решений, выбор и обучение модели глубокого обучения для обнаружения предметов, разработку пользовательских интерфейсов, контейнеризацию приложения и тестирование системы в реальных условиях. Результатом работы стала система, способная обнаруживать интересующие объекты в режиме реального времени с помощью нейронных сетей и уведомлять пользователя об обнаружении. Система предназначена для использования в общественных местах, таких как аэропорты, вокзалы, школы и другие учреждения, требующие повышенной безопасности.

Ключевые слова: информационная система, искусственные нейронные сети, распознавание образов

Для цитирования: Василенко Н. Е., Медведева Н. А. Нейросетевая система видеонаблюдения // Вестник кибернетики. 2024. Т. 23, № 4. С. 25–33. <https://doi.org/10.35266/1999-7604-2024-4-3>.

Original article

Neural network surveillance system

Nikita E. Vasilenko¹✉, Natalya A. Medvedeva²

^{1,2}Surgut State University, Surgut, Russia

¹vasilenko_ne@edu.surgu.ru✉, <https://orcid.org/0009-0007-7880-1026>

²medvedeva_na@surgu.ru, <https://orcid.org/0009-0005-9379-718X>

Abstract. The purpose of the paper is to design and implement an efficient and affordable automatic surveillance system, which can operate in real time and integrate with existing surveillance systems. The method presented includes analyzing existing solutions on the market, selecting and training a profound learning model for objects detection, developing user interfaces, containerizing of the application and testing the system in a real-world environment. The result is a system capable of detecting objects of interest in real time using neural networks and notifying the user of the detected items. This system is designed for public places like airports, railway stations, schools, and other institutions needing enhanced security.

Keywords: information system, artificial neural networks, pattern recognition

For citation: Vasilenko N. E., Medvedeva N. A. Neural network surveillance system. *Proceedings in Cybernetics*. 2024;23(4):25–33. <https://doi.org/10.35266/1999-7604-2024-4-3>.

ВВЕДЕНИЕ

В настоящее время разработка видеосистем автоматического обнаружения предметов, входящих в область интересов, имеет критическое значение в контексте обеспечения общественной безопасности [1, 2]. Такие системы обещают значительно улучшить реакцию на потенциальные угрозы и инциденты, предотвращая их возникновение или минимизируя их последствия. Однако существует ряд проблем, связанных с трудностями внедрения, установки, конфигурации и обслуживания подобных систем.

При разработке системы использовались современные методы машинного обучения и искусственного интеллекта, подходы к созданию веб- и десктоп-приложений, а также данные о существующих решениях на рынке. Были проанализированы различные подходы к видеоаналитике и детектированию объектов, такие как алгоритм YOLO и методы сверточных нейронных сетей.

Актуальность работы обусловлена ростом глобальных угроз, таких как преступность и несанкционированные действия, а также возрастающими требованиями к общественной и частной безопасности. Интеграция современных алгоритмов глубокого обучения в систему, способную работать в реальном времени и предоставлять высокую точность обнаружения обещает дать толчок в развитии, как таких систем, так и всей сферы безопасности в целом.

Существуют решения, разработанные такими компаниями, как Videonet, ZeroEyes, Athena Security, Actuate и NtechLab, которые также направлены на улучшение безопасности с использованием видеоаналитики и искусственного интеллекта.

После анализа этих решений можно выделить ряд их недостатков:

1. Закрытость кода: все продукты компаний, кроме NtechLab, которые показали достоинства их алгоритма, не предоставляют доступ к исходному коду, ограничивая возможности развития, кастомизации и интеграции под специфические потребности.

2. Высокая стоимость лицензирования, поддержки и обновлений делают такие системы недоступными для многих организаций, не говоря о персональном использовании.

3. Сложность интеграции с существующими системами безопасности из-за возможной несовместимости или отсутствия необходимых интерфейсов.

МАТЕРИАЛЫ И МЕТОДЫ

При проектировании системы была выбрана модель YOLO («You Only Look Once») [3–6] для задачи обнаружения предметов в реальном времени, что обусловлено рядом причин (рис. 1).

Прежде всего, эта модель представляет собой эффективную архитектуру нейронной сети, в которой входное изображение разби-

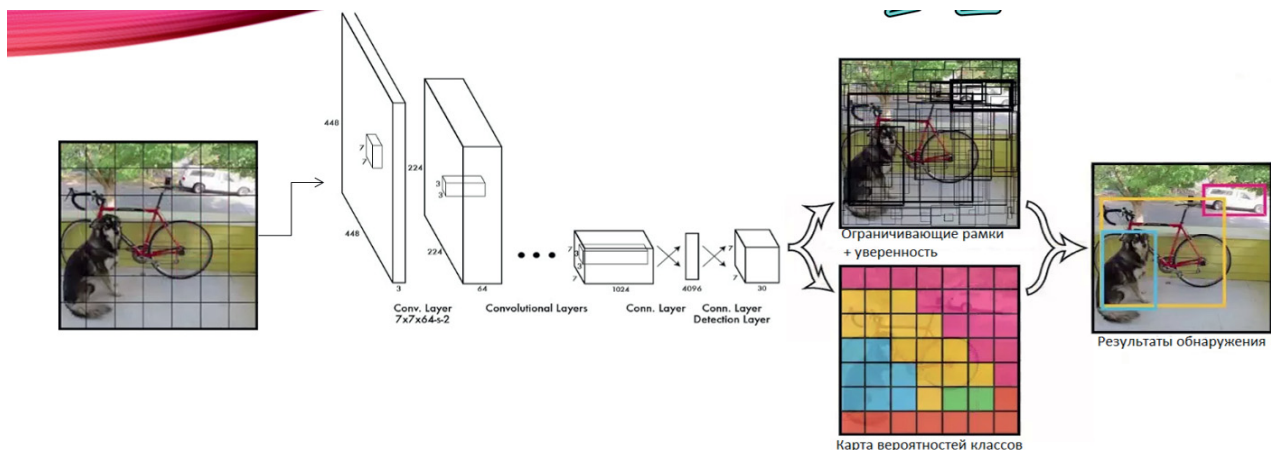


Рис. 1. Принцип работы алгоритма YOLO

Примечание: составлено авторами по [3].

вается на сетку, и каждая ячейка сетки отвечает за обнаружение объектов в своей области.

Параметры этих ячеек (рис. 2) в виде векторов [7] являются выходным результатом работы модели, в основе которой лежит принцип сверточной нейронной сети.

Для получения предсказаний модели, изображению требуется пройти через нее всего один раз. Это отличает алгоритм YOLO от других алгоритмов обнаружения объектов и позволяет выполнять инференс с высокой скоростью (рис. 3), что очень важно для си-

стемы, работающей в режиме реального времени.

Установка необходимых библиотек, импортирование модели, установка датасета, обучение модели и экспорт файла весов производились в среде Google Colab с подключенным Google хранилищем для сохранения файловой структуры проекта (рис. 4–5) [8].

Параметры обучения (рис. 6) подбирались экспериментальным путем для выведения наилучшей точности модели. Крайний вариант модели обучался на протяжении



Рис. 2. Параметры ячеек изображения
Примечание: составлено авторами по [3].

Model	mAP	FPS	Real Time speed
Fast YOLO	52.7%	155	Yes
YOLO	63.4%	45	Yes
YOLO VGG-16	66.4%	21	No
Fast R-CNN	70.0%	0.5	No
Faster R-CNN VGG-16	73.2%	7	No
Faster R-CNN ZF	62.1%	18	No

Рис. 3. Сравнение скорости и производительности моделей, обученных с наборами данных PASCAL VOC 2007 и 2012 гг.

Примечание: составлено авторами.

```

[ ] import os
    HOME = '/content/drive/MyDrive'
    print(HOME)

/content/drive/MyDrive

[ ] from google.colab import drive
    drive.mount('/content/drive')

Mounted at /content/drive

[ ] %cd {HOME}
    !pip install ultralytics==8.0.196
  
```

Рис. 4. Подключение Google хранилища и установка библиотеки для работы с моделями ИИ

Примечание: составлено авторами.

```
[ ] from ultralytics import YOLO

[ ] !pip install roboflow

from roboflow import Roboflow
rf = Roboflow(api_key="tnnsWsL9TYAoERnfrZvV")
project = rf.workspace("imam-maulana-b4xet").project("handgun-detection-jtvaj")
version = project.version(9)
dataset = version.download("yolov8")
```

Рис. 5. Импорт модели YOLO и установка датасета с помощью библиотеки Roboflow

Примечание: составлено авторами.

```
[ ] !yolo task=detect \
mode=train \
model=yolov8s.pt \
data=/content/drive/MyDrive/Handgun-Detection-9/data.yaml \
epochs=30 \
batch=16 \
device=0 \
imgsz=640
```

Рис. 6. Метод обучения модели на установленном датасете с определенными параметрами обучения

Примечание: составлено авторами.

30-ти эпох, с количеством образцов данных, обрабатываемых нейронной сетью за одну итерацию, равную 16-ти, и с использованием более 4-х Гб видеопамяти.

При конечном инференсе (рис. 7–9) модель показала 78,5% средней точности, что, относительно затраченных ресурсов, является хорошим показателем.

Для создания системы автоматического оповещения пользователя об обнаружении нежелательных предметов в реальном времени использовались современные технологии для реализации серверной и клиентской части программного обеспечения:

1. Qt Designer – инструмент для построения графических интерфейсов для программ, которые используют библиотеку Qt.

2. Разработка веб-приложения осуществлялась с использованием фреймворка Flask. Его главным преимуществом является простая и гибкая архитектура, позволяющая создавать веб-приложения любой сложности.

3. Платформа для контейнеризации Docker, с помощью которого программный код упаковывается

выводится в единый исполняемый файл вместе с библиотеками и зависимостями, чтобы обеспечить его корректный запуск.

4. Библиотека PyQt5 для верстки стороны клиента и создание алгоритма обнаружения предметов для десктоп-приложения.

РЕЗУЛЬТАТЫ И ИХ ОБСУЖДЕНИЕ

В файле «desktop.py» находится код, интерпретация которого (рис. 10) создаст объект «QApplication», представляющий собой основное окно приложения PyQt5. После чего создастся экземпляр «LoginWindow» (окно входа), установится иконка окна и оно отобразится (рис. 11).

Откроется следующее окно. Если пользователь не регистрировался ранее, то он может это сделать нажатием кнопки «Зарегистрироваться» (рис. 12).

Это действие откроет веб-страницу с разделом регистрации, что позволит пользователю зарегистрироваться (рис. 13).

После регистрации пользователь может войти в личный кабинет. Теперь, после вноса

Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size
24/30	4.29G	0.7793	0.5968	1.339	24	640: 100% 303/303 [02:02<00:00, 2.47it/s]
Class	Images	Instances	Box(P	R	mAP50	mAP50-95): 100% 10/10 [00:04<00:00, 2.28it/s]
all	300	363	0.774	0.614	0.73	0.49
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size
25/30	4.15G	0.7622	0.5677	1.313	14	640: 100% 303/303 [02:01<00:00, 2.48it/s]
Class	Images	Instances	Box(P	R	mAP50	mAP50-95): 100% 10/10 [00:04<00:00, 2.25it/s]
all	300	363	0.758	0.663	0.739	0.504
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size
26/30	4.14G	0.7427	0.5408	1.307	14	640: 100% 303/303 [02:01<00:00, 2.50it/s]
Class	Images	Instances	Box(P	R	mAP50	mAP50-95): 100% 10/10 [00:04<00:00, 1.81it/s]
all	300	363	0.758	0.678	0.755	0.514
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size
27/30	4.29G	0.7223	0.5107	1.286	13	640: 100% 303/303 [02:05<00:00, 2.40it/s]
Class	Images	Instances	Box(P	R	mAP50	mAP50-95): 100% 10/10 [00:04<00:00, 2.26it/s]
all	300	363	0.822	0.667	0.765	0.52
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size
28/30	4.29G	0.7029	0.4937	1.258	13	640: 100% 303/303 [02:05<00:00, 2.42it/s]
Class	Images	Instances	Box(P	R	mAP50	mAP50-95): 100% 10/10 [00:04<00:00, 2.11it/s]
all	300	363	0.837	0.678	0.78	0.54
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size
29/30	4.15G	0.688	0.4751	1.249	13	640: 100% 303/303 [02:01<00:00, 2.49it/s]
Class	Images	Instances	Box(P	R	mAP50	mAP50-95): 100% 10/10 [00:05<00:00, 1.70it/s]
all	300	363	0.839	0.677	0.782	0.54
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size
30/30	4.14G	0.6721	0.4523	1.238	13	640: 100% 303/303 [02:02<00:00, 2.48it/s]
Class	Images	Instances	Box(P	R	mAP50	mAP50-95): 100% 10/10 [00:04<00:00, 2.18it/s]
all	300	363	0.826	0.686	0.786	0.541

30 epochs completed in 1.119 hours.
Optimizer stripped from runs/detect/train2/weights/last.pt, 22.5MB
Optimizer stripped from runs/detect/train2/weights/best.pt, 22.5MB

Validating runs/detect/train2/weights/best.pt...
Ultralytics YOLOv8.0.196 Python-3.10.12 torch-2.2.1+cu121 CUDA:0 (Tesla T4, 15102MiB)
Model summary (fused): 168 layers, 11125971 parameters, 0 gradients, 28.4 GFLOPs

Class	Images	Instances	Box(P	R	mAP50	mAP50-95): 100% 10/10 [00:09<00:00, 1.07it/s]
all	300	363	0.826	0.686	0.785	0.541

Speed: 0.4ms preprocess, 5.4ms inference, 0.0ms loss, 7.1ms postprocess per image
Results saved to runs/detect/train2
Learn more at <https://docs.ultralytics.com/modes/train>

Рис. 7. Зафиксированные последние эпохи обучения модели с выводом результатов

Примечание: составлено авторами.

```
[ ] model = YOLO('/content/drive/MyDrive/runs/detect/train2/weights/best.pt')  
exported = model.export(format="onnx")
```

Рис. 8. Экспорт файла весов в необходимый для дальнейшей работы формат

Примечание: составлено авторами.

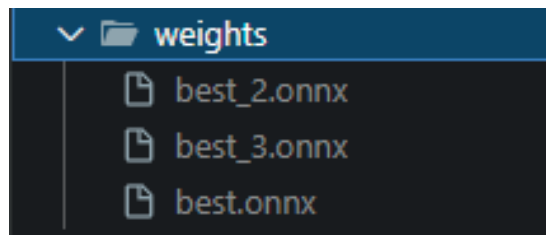


Рис. 9. Каталог с файлами весов

Примечание: составлено авторами.

пользователя в базу данных, можно успешно войти систему локально и указать местоположение камеры, с помощью которой планируется проводить мониторинг (рис. 14–15).

Запустив мониторинг, перед пользователем появится окно мониторинга, где изображение будет передаваться в реальном времени с указанной камеры (рис. 16).

```
desktop.py > ...
1  from PyQt5.QtWidgets import QApplication
2  from PyQt5.QtGui import QIcon
3  import sys
4  import login_window
5
6  def main():
7      app = QApplication(sys.argv)
8      app_icon = QIcon("UI/ICON_32.png")
9
10     mainwindow = login_window.LoginWindow()
11     mainwindow.setWindowIcon(app_icon)
12
13     try:
14         sys.exit(app.exec_())
15     except Exception as e:
16         print(f"An error occurred: {e}")
17     finally:
18         login_window.smtpObj.quit()
19         print("Exiting...")
20
21 if __name__ == "__main__":
22     main()
```

Рис. 10. Содержимое файла desktop.py

Примечание: составлено авторами.

```
(base) PS E:\UNIVERSITY\DIPLом\SOWA> python desktop.py
█
```

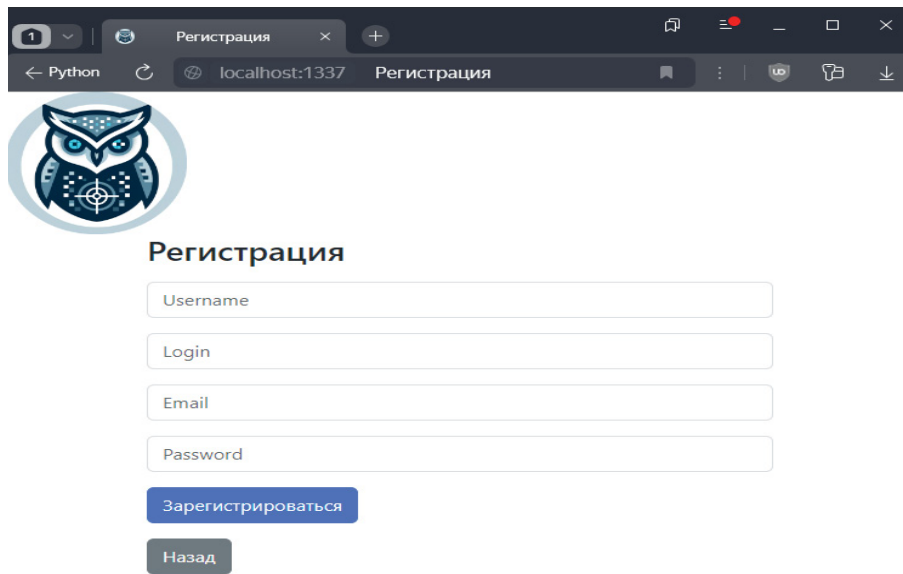
Рис. 11. Команда запуска приложения пользователя

Примечание: составлено авторами.

The screenshot shows a window titled "SOWA" with a standard Windows title bar (minimize, maximize, close buttons). The window content has a light gray background. At the top center, the text "Вход в систему" (Login to the system) is displayed. Below this, there are two labels: "Имя пользователя:" (Username:) and "Пароль:" (Password:). Each label is followed by a white rectangular input field with a thin blue border. At the bottom of the window, there are two gray rectangular buttons with rounded corners. The left button is labeled "Войти" (Login) and the right button is labeled "Зарегистрироваться" (Register).

Рис. 12. Окно входа

Примечание: составлено авторами.



Регистрация

Username

Login

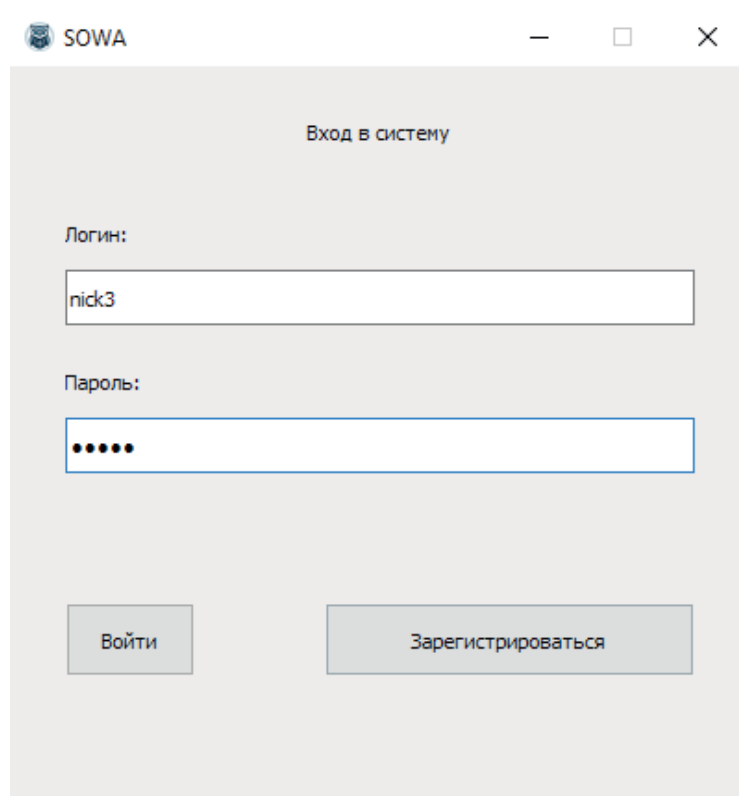
Email

Password

Зарегистрироваться

Назад

Рис. 13. Страница регистрации
Примечание: составлено авторами.



SOWA

Вход в систему

Логин:

nick3

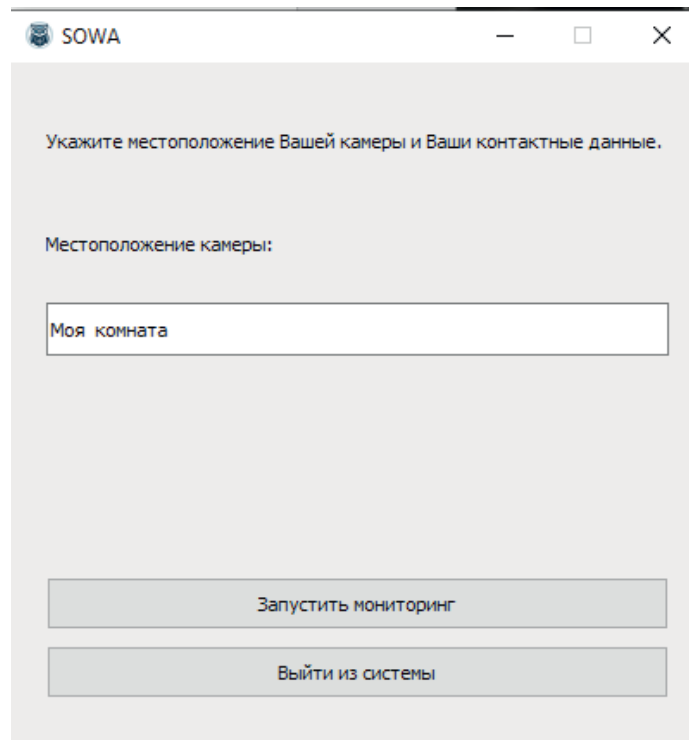
Пароль:

•••••

Войти

Зарегистрироваться

Рис. 14. Вход в систему
Примечание: составлено авторами.



Укажите местоположение Вашей камеры и Ваши контактные данные.

Местоположение камеры:

Моя комната

Запустить мониторинг

Выйти из системы

Рис. 15. Заполнение полей окна настройки
Примечание: составлено авторами.

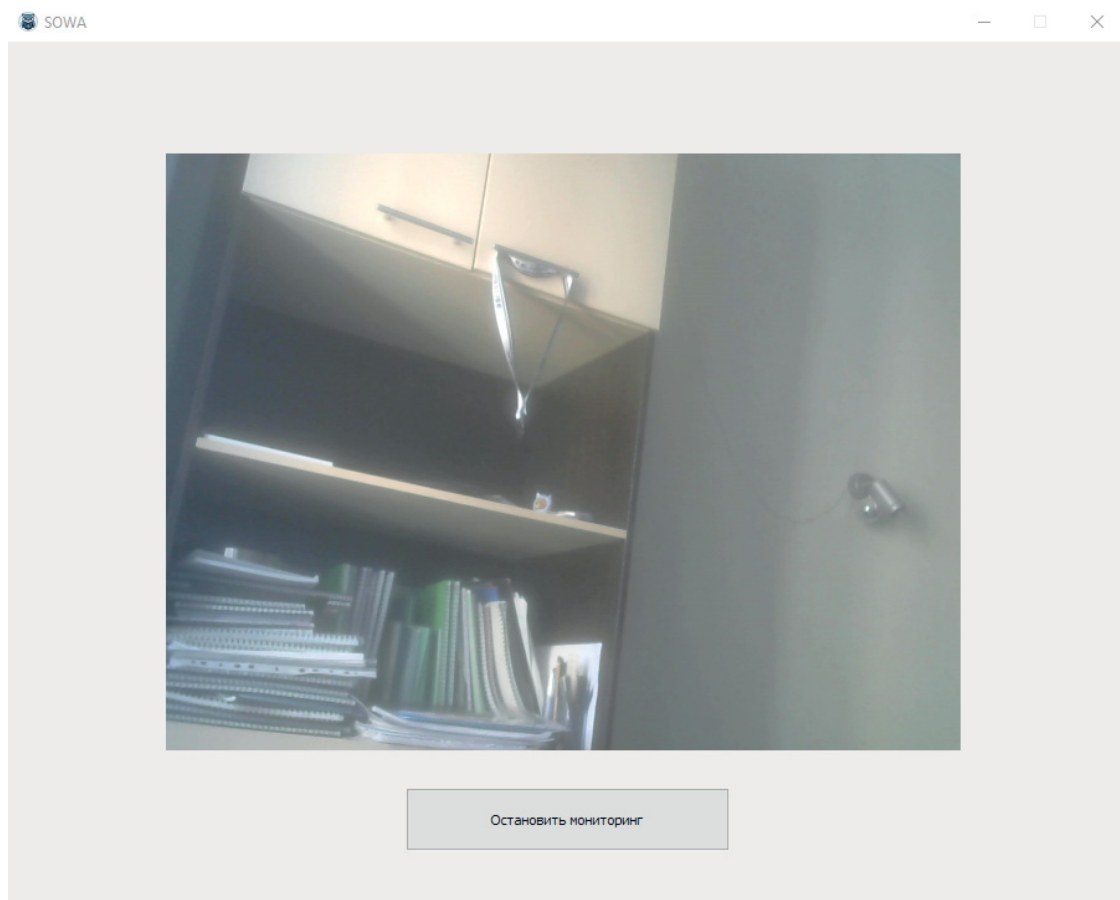


Рис. 16. Процесс мониторинга
Примечание: составлено авторами.

ЗАКЛЮЧЕНИЕ

В рамках научно-исследовательской работы была разработана и реализована интегрированная система, которая включает в себя следующие основные компоненты:

1. Обнаружение предметов в реальном времени. Для этого были использованы технологии компьютерного зрения и глубокого обучения. Эти технологии позволяют системе анализировать видеопоток с камер наблюдения и автоматически определять наличие предметов на изображениях.

2. Автоматическое оповещение. Когда система обнаруживает предметы в поле зрения камеры, она автоматически отправляет оповещение пользователю, а в личном кабинете он сможет посмотреть подробную информацию об обнаружении. Такая система позволяет оперативно реагировать на экстренные ситуации и ее применение способ-

ствует дальнейшему развитию систем безопасности с использованием современных технологий.

3. Доступность и масштабируемость. Для обеспечения доступности и масштабируемости при реализации системы была использована контейнеризация. Этот подход позволил создать изолированную среду выполнения, которая легко переносится и запускается на различных платформах, включая локальные компьютеры. Благодаря этому, система становится доступной для обычных пользователей и способна к масштабированию, позволяя повысить производительность при увеличении вычислительных ресурсов.

Видеосистема автоматического оповещения пользователя об обнаружении предметов повышенного риска в реальном времени является важной частью обеспечения как общественной, так и персональной безопасности.

Список источников

1. Ларина Ю. Событийный видеоконтроль // Безопасность. Достоверность. Информация. 2009. № 83. С. 28–40.
2. VideoNet PSIM – интеллектуальная система безопасности. URL: <https://www.videonet.ru/> (дата обращения: 05.05.2024).
3. Распознавание образов с помощью искусственного интеллекта. URL: <https://habr.com/ru/articles/709432/> (дата обращения: 05.05.2024).
4. YOLO: You Only Look Once – Real Time Object Detection – GeeksforGeeks. URL: <https://www.geeksforgeeks.org/yolo-you-only-look-once-real-time-object-detection/> (дата обращения: 05.05.2024).
5. Real-Time Object Identification Through Convolution Neural Network Based on YOLO Algorithm. URL: <https://www.sciencepg.com/article/10.11648/j.mcs.20230805.11> (дата обращения: 05.05.2024).
6. YOLO with adaptive frame control for real-time object detection applications. URL: <https://link.springer.com/article/10.1007/s11042-021-11480-0> (дата обращения: 05.05.2024).
7. D_0.06 YOLO – Head. URL: <https://junha1125.github.io/blog/artificial-intelligence/2020-08-18-YOLO/> (дата обращения: 05.05.2024).
8. How to Train YOLOv8 Object Detection on a Custom Dataset. URL: <https://blog.roboflow.com/how-to-train-yolov8-on-a-custom-dataset/> (дата обращения: 05.05.2024).

Информация об авторах

Н. Е. Василенко – магистрант.

Н. А. Медведева – старший преподаватель.

References

1. Larina Yu. Sobytiinyi videokontrol. *Bezopasnost. Dostovernost. Informatsiia*. 2009;(83):28–40. (In Russ.).
2. VideoNet PSIM – intelektualnaia sistema bezopasnosti. URL: <https://www.videonet.ru/> (accessed: 05.05.2024). (In Russ.).
3. Raspoznavanie obrazov s pomoshchyu iskusstvennogo intellekta. URL: <https://habr.com/ru/articles/709432/> (accessed: 05.05.2024). (In Russ.).
4. YOLO: You Only Look Once – Real Time Object Detection – GeeksforGeeks. URL: <https://www.geeksforgeeks.org/yolo-you-only-look-once-real-time-object-detection/> (accessed: 05.05.2024).
5. Real-Time Object Identification Through Convolution Neural Network Based on YOLO Algorithm. URL: <https://www.sciencepg.com/article/10.11648/j.mcs.20230805.11> (accessed: 05.05.2024).
6. YOLO with adaptive frame control for real-time object detection applications. URL: <https://link.springer.com/article/10.1007/s11042-021-11480-0> (accessed: 05.05.2024).
7. D_0.06 YOLO – Head. URL: <https://junha1125.github.io/blog/artificial-intelligence/2020-08-18-YOLO/> (accessed: 05.05.2024).
8. How to Train YOLOv8 Object Detection on a Custom Dataset. URL: <https://blog.roboflow.com/how-to-train-yolov8-on-a-custom-dataset/> (accessed: 05.05.2024).

About the authors

N. E. Vasilenko – Master's Degree Student.

N. A. Medvedeva – Senior Lecturer.