Додаток А

Апробація результатів кваліфікаційної роботи

Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

кафедра комп'ютерно-інтегрованих технологій, автоматизації та робототехніки

(КІТАР)

МАТЕРІАЛИ

II Всеукраїнської конференції

«Комп'ютерно-інтегрованих технологій, автоматизації та робототехніки»

(Computer-integrated technologies, automation and robotics)

CITAR`25

16-17 травня 2025

[електронне видання]

Харків 2025

Рисунок А.1 – Титульний аркуш

УДК: 005:004.896:62-65:338.3

Комп'ютерно-інтегрованих технологій, автоматизації та робототехніки 2025: матеріали II-ої Всеукраїнської конференції, Харків, 16-17 травня 2025.: тези доповідей / [редкол. І.Ш. Невлюдов (відповідальний редактор)].-Харків: [електронний друк], 2025. – 132 с.

У збірник включені тези доповідей, які присвячені сучасним автоматизованим технологіям Industry 4.0 та їх впровадження; інформаційні управляючі системи технологічного призначення; математичні методи в системах автоматизації; розробка та програмування в робототехніці; штучний інтелект та машинне навчання в автоматизації; інтеграція технологій у виробництві та промисловості; сенсорні технології та взаємодія людини з роботами в Industry 5.0; ефективність використання роботизованих систем у виробництві; етика та правові аспекти в робототехніці; Інтернет речей та Інтегровані системи в комп'ютерно-інтегрованих технологіях, автоматизації та робототехніки; технологічні виклики та інновації у світі робототехніки.

Редакційна колегія: І.Ш. Невлюдов, В.В. Євсєєв.

Computer-integrated technologies, automation and robotics 2025: Proceedings of II st All-Ukrainian Conference, Kharkiv, May 16-17, 2025: Thesises of Reports / [Ed. I.Sh. Nevlyudov (chief editor).] .- Kharkiv .: [electronic version], 2025. - 132 p.

The collection includes abstracts devoted to modern automated technologies of Industry 4.0 and their implementation; information control systems for technological purposes; mathematical methods in automation systems; development and programming in robotics; artificial intelligence and machine learning in automation; integration of technologies in production and industry; sensor technologies and human interaction with robots in Industry 5.0; efficiency of using robotic systems in production; ethics and legal aspects in robotics; Internet of Things.

Editorial board: Igor.Sh. Nevludov, Vladyslav.V. Yevsieiev

Рисунок А.2 – Опис конференції

ЗМІСТ

Рисунок А.3 – Зміст конференції

Рисунок А.4 – Друга сторінка змісту конференції

MAIN TRENDS IN THE DEVELOPMENT OF AUTOMATED IMAGE PROCESSING SYSTEMS

A. Konieva, S. Sotnik
Kharkiv National University of Radio Electronics
Ukraine, 61166, Kharkiv, Nauky av, 14
E-mail: alina.konieva@nure.ua

Annotation: The paper studies the current trends in the development of automated image processing systems, which play a key role in the digital transformation of various industries. Innovative approaches, such as the integration of artificial intelligence, deep learning and cloud computing, are considered, which can significantly improve the accuracy, speed and scalability of visual data processing. The work emphasizes that the further development of automated image processing systems is an important step for the introduction of innovative solutions in modern technologies. The work allows us to evaluate the importance of such systems for improving the efficiency, accuracy and speed of visual data processing, which is a key factor in progress in the digital age.

Keywords: automated image processing systems, artificial intelligence, machine learning, computer vision.

ОСНОВНІ ТЕНДЕНЦІЇ РОЗВИТКУ АВТОМАТИЗОВАНИХ СИСТЕМ ОБРОБКИ ЗОБРАЖЕНЬ

А. І. Конєва, С. В. Сотник
Харківський Національний Університет Радіоелектроніки
Україна, 61166, Харків, пр. Науки 14
E-mail: alina.konieva@nure.ua, svetlana.sotnik@nure.ua

Анотація: У роботі досліджено сучасні тенденції розвитку автоматизованих систем обробки зображень, які відіграють ключову роль у цифровій трансформації різних галузей. Розглянуто інноваційні підходи, такі як інтеграція штучного інтелекту, глибокого навчання та хмарних обчислень, що дозволяють значно покращити точність, швидкість та масштабованість обробки візуальних даних. Робота підкреслює, що подальший розвиток автоматизованих систем обробки зображень є важливим кроком для впровадження інноваційних рішень у сучасних технологіях. Робота дозволяє оцінити значення таких систем для підвищення ефективності, точності та швидкості обробки візуальних даних, що є ключовим фактором прогресу у цифрову епоху.

Ключові слова: автоматизовані системи обробки зображень, штучний інтелект, машинне навчання, комп'ютерний зір.

RELEVANCE OF THE WORK. Modern automated image processing systems are actively developing under the influence of the latest technologies, such as artificial intelligence, machine learning, and cloud computing [1-6]. Automated image processing systems (AIPS) play an important role in modern technologies used in various fields, including medicine, industry, security, and artificial intelligence. With the development of computing power and machine learning algorithms, these systems are becoming more efficient and accurate.

The relevance of information, its protection, and process automation are key aspects of the modern digital world, as they ensure effective data management, increase system reliability, and help optimize production and business processes in the face of rapidly growing information and cyber threats [7-10]. In this context, it is of particular importance to analyze the main trends in the

Рисунок А.5 – Тезиси, перша сторінка

development of automated image processing systems, which play an important role in ensuring fast and accurate processing of visual data, which is an integral part of modern technological solutions.

Printed from DeepL.com (free version)One of the most important trends is the integration of artificial intelligence (AI) and deep learning. Neural networks, such as convolutional neural networks (CNNs), have been shown to be highly effective in pattern recognition, object classification, and image segmentation. An important factor in the development is the improvement of hardware, such as graphics processing units (GPUs) and tensor processors (TPUs), which significantly accelerate the processing of large amounts of data.

MATERIALS AND RESEARCH RESULTS. ASICs are being actively implemented in augmented and augmented reality (AR/VR) technologies, which improves user interaction with visual information. In addition, process automation and the introduction of self-learning algorithms contribute to the efficiency of real-time image analysis.

Figure 1.1 – Augmented and supplemented reality (AR/VR) technologies

The modern market offers a wide range of software for automated image processing. The most popular solutions are based on computer vision, machine learning, and neural network algorithms. The most famous products include:

- OpenCV is an open source computer vision library that contains a wide range of tools for image and video processing;
- TensorFlow and PyTorch are deep learning frameworks that are widely used to create neural networks for image processing;
- MATLAB Image Processing Toolbox – a powerful tool for scientific research and engineering calculations in the field of image analysis;
- Adobe Photoshop and GIMP – graphic editors that contain elements of automation and artificial intelligence algorithms for image enhancement.

In addition, specialized solutions for medicine, security, and industrial control are being developed that allow for real-time image analysis with high accuracy. In medicine, such systems are used to automatically recognize pathologies in MRI, CT, and X-ray images, which helps doctors make diagnoses faster and more accurately. In the security sector, image processing technologies are used to recognize faces, detect suspicious objects and behaviors, which significantly increases the level of control in public places and businesses. In industry, automated systems analyze products on conveyors, detect defects, and control the quality of parts, which helps reduce rejects and increase production efficiency. Thus, specialized solutions for various industries can significantly improve the quality and speed of visual information processing, contributing to the development of their respective fields of activity.

Рисунок А.6 – Тезиси, друга сторінка

For automated image processing systems to operate efficiently, it is necessary to take into account the accuracy and speed of processing, as the system must ensure high accuracy of image analysis with minimal time.

Flexibility and scalability are also important aspects, allowing the system to be adapted to different tasks and data volumes. Process automation minimizes manual intervention through the use of intelligent algorithms. In addition, the system must be compatible with various data formats, supporting standard graphic formats such as JPEG, PNG, BMP, TIFF. Reliability and security are key features, including encryption and access control to confidential information. The ability to work in real time is also an important requirement, which is critical for video surveillance systems, autonomous vehicles, and medical applications. Thus, modern automated image processing systems must meet high performance, accuracy, and security requirements to ensure their effective use in various fields of activity.

One of the key trends is the integration of artificial intelligence (AI) and deep learning into image processing systems. The use of neural networks improves the quality of object recognition, image analysis, and forecasting changes in visual data. Popular architectures in this area include CNN (Convolutional Neural Networks) and GAN (Generative Adversarial Networks).

Computer vision continues to improve, enabling automated systems to recognize objects, analyze scenes, and interact with the environment in real time. Thanks to augmented and virtual reality technologies, computer vision can be used in medicine, industry, and autonomous transportation. The latest developments in this area include algorithms for three-dimensional object recognition, quality control systems in production, and automatic detection of dangerous situations in video surveillance. Methods of multimodal analysis that combine images with other data sources, such as text or audio information, are also being actively researched.

Cloud computing allows you to store and process large volumes of images without the need for local resources. This facilitates the development of real-time image processing technologies and ensures the scalability of solutions. Cloud services make it possible to integrate automated image processing systems into large information platforms, ensuring data availability from anywhere in the world. In addition, new methods are being developed to reduce data transmission delays, which improves the efficiency of such systems.

The development of image compression and quality enhancement algorithms contributes to the efficient use of resources and reduces processing delays. The use of super-resolution technologies allows improving image quality even at low resolution.

Considerable attention is paid to adaptive compression algorithms that dynamically change the compression level depending on the context of use. This is especially important for video streaming, where it is necessary to maintain a balance between quality and download speed.

The development of automated image processing systems continues to evolve thanks to innovations in artificial intelligence, cloud computing, and computer vision. These technologies open up new opportunities in various industries, improving the efficiency and accuracy of visual data analysis. Further research and implementation of new methods will contribute to even greater progress in this area.

CONCLUSIONS. Thus, the development of automated image processing systems is not only a technological progress, but also an important step in solving many modern challenges facing society.

The thesis discusses the main trends in the development of automated image processing systems, which is necessary for understanding current achievements and prospects in this area. The integration of artificial intelligence and deep learning is analyzed, which can significantly improve the accuracy and efficiency of image processing. This is important to ensure high-quality analysis of large amounts of visual data in areas such as medicine, industry, transportation, and security.

Рисунок А.7 – Тезиси, третя сторінка

The abstracts also describe hardware improvements, including the use of graphics processing units (GPUs) and tensor processors (TPUs), which speeds up data processing. This is essential for real-time operation of systems, which is especially important for medical diagnostic systems, autonomous vehicles, and video surveillance.

The article considers the use of automated systems in augmented and virtual reality (AR/VR) technologies, which opens up new opportunities for improving user interaction with visual information. This is important for the development of innovative solutions in education, entertainment, medicine, and industry.

The abstracts also emphasize the role of cloud computing, which allows storing and processing large volumes of images, ensuring scalability and accessibility of solutions. This is essential for the integration of automated systems into global information platforms, allowing data to be processed from anywhere in the world.

In addition, the abstracts describe the development of image compression and quality enhancement algorithms, which is important for efficient resource utilization and reduction of processing delays. This is especially important for video streaming and working with large data sets.

Thus, the thesis presents a comprehensive analysis of current trends in the development of automated image processing systems, which is necessary to understand their role in improving technologies in various industries. This makes it possible to assess the importance of such systems for increasing the efficiency, accuracy, and speed of visual data processing, which is a key factor in progress in the digital age.

REFERENCES:

1. Khalimonov, Y. I., & et al. (2024). Integration of IoT into security systems: opportunities and risks. Комп'ютерно-інтегровані технології автоматизації технологічних процесів на транспорті та у виробництві: матеріали всеукр. наук.-практ. конф. здобувачів вищ. освіти і молодих учених, 20 листоп. 2024 р. – pp. 117-121Polikanov, K., & et al. (2024). Smart home with house module: overview of automation technologies. International Conference «DIGITAL INNOVATION & SUSTAINABLE DEVELOPMENT 2024», 2024 - pp. 20-21

2. Khalimonov, Y., & et al. (2024). Approaches to ensuring proper working conditions using sensor technologies IoT. International Conference «DIGITAL INNOVATION & SUSTAINABLE DEVELOPMENT 2024», 2024 - pp. 24-25

3. Hubar, A.Y., Sotnik, S.V. Impact of automation and CALS technologies on human factor in production // The 5th International scientific and practical conference "Perspectives of contemporary science: theory and practice" (June 24-26, 2024) SPC "Sci?conf.com.ua", Lviv, Ukraine, 2024. – c. 243-249.

4. Khalimonov, Y. I., & et al. (2024). Monitoring and optimising conditions in production environment. Proceedings of the XVII International scientific and practical conference «Information technologies and automation– 2024», 2024. – pp. 256-258

5. Sotnik, S. V. (2024). Development of automated control system and registration of metal in continuous casting. Radio Electronics, Computer Science, Control. – pp. 197-211

6. Rudenko, M., & et al. (2024). Overview of approaches to scaling relational databases in development and adaptation of web applications. Сучасні проблеми і досягнення в галузі радіотехніки, телекомунікацій та інформаційних технологій: Тези доповідей XII Міжнародної науково-практичної конференції (10-12 грудня 2024 р., м. Запоріжжя). – pp. 398-402

7. Kaponkin, V. G., & et al. (2024). The role of big data in improving functionality of search engines. The 8th International scientific and practical conference "European congress of scientific achievements" (August 12-14, 2024) Barca Academy Publishing, Barcelona, Spain. – pp. 69-76

Рисунок А.8 – Перелік джерел посилання

8. Sotnik, S. V. (2024). Features of using REST architecture for development of ARS for information systems // Міжнародна науково-практична конференція «Інформаційні системи в управлінні проєктами та програмами», Коблево, 9–13 вересня 2024 р. Збірник праць. – Харків: ХНУРЕ. – pp. 42 – 45.

9. Andreiev, A. S., & et al. (2024). Computer games and Web design // Proceedings of the XVII International scientific and practical conference «Information technologies and automation – 2024». – pp. 712-714

10. Danylenko, M. M., & et al. (2025). Comparative analysis of modern SCADA packages for production automation. International Journal of Academic Engineering Research (IJAER). – Vol. 9. – 2. – pp. 26-34

Рисунок А.9 – Перелік джерел посилання

Додаток Б

Листінг програми Python

```
import sys
import os
sys.path.append("E:/ImageEditorApp/libs")
BASE_DIR = os.path.dirname(os.path.abspath(__file__))
sys.path.append(BASE_DIR)
sys.path.append(os.path.join(BASE_DIR, "ui"))
sys.path.append(os.path.join(BASE_DIR, "image_processing"))
sys.path.append(os.path.join(BASE_DIR, "core"))
import tkinter as tk
from utils.image_state_manager import ImageStateManager
from tkinter import messagebox
from utils.image_utils import simulate_zoom
from ui.processing_panel import ProcessingPanel
from ui.analysis_panel import AnalysisPanel
from ui.toolbar import Toolbar
from ui.history_panel import HistoryManager
from utils.defect_utils import detect_all_defects
from ui.presets_panel import PresetsPanel
from PIL import Image, ImageTk
from image_processing.filters import (
    analyze_sharpness,
    apply_blur,
    apply_sharpen,
    apply_edges,
    adjust_brightness,
    adjust_contrast,
    adjust_saturation,
    apply_sepia,
```

```
    apply_negative,
    apply_grayscale,
    apply_threshold,
    apply_denoise,
    apply_clahe,
    apply_hue_shift,
    apply_posterize,
    apply_sketch,
    apply_canny,
    apply_color_detection
)
from utils.image_utils import (
    rotate_image,
    flip_horizontal,
    flip_vertical
)
class ImageEditorApp:
    def __init__(self, root):
        self.root = root
        self.root.title("PixControl")

        icon_path = os.path.join(os.path.dirname(__file__), "icon.ico")
        try:
            self.root.iconbitmap(icon_path)

        except Exception as e:
            print("Не вдалося встановити іконку:", e)

        self.root.state('zoomed')
```

```
self.root.config(bg="#2C2C2C")
self.current_panel = None
self.current_panel_class = None
self.image_tk = None
self.image_files = []
self.current_index = -1
self.image_index_label = None
self.default_font = ("Segoe UI", 11)
self.view_width = 1000
self.view_height = 700
self.manual_zoom = False
self.filter_input_image = None
self.active_filter = None
self.comparison_mode = False
self.image_manager = ImageStateManager()
self.bind_shortcuts()
self.toolbar = Toolbar(self)
self.analyze_sharpness = analyze_sharpness
self.apply_blur = apply_blur
self.apply_sharpen = apply_sharpen
self.apply_edges = apply_edges
self.adjust_brightness = adjust_brightness
self.adjust_contrast = adjust_contrast
self.adjust_saturation = adjust_saturation
self.apply_sepia = apply_sepia
self.apply_negative = apply_negative
self.apply_grayscale = apply_grayscale
self.apply_threshold = apply_threshold
self.apply_denoise = apply_denoise
```

```
        self.apply_clahe = apply_clahe
        self.apply_hue_shift = apply_hue_shift
        self.apply_posterize = apply_posterize
        self.apply_sketch = apply_sketch
        self.apply_canny = apply_canny
        self.detect_color = apply_color_detection
        self.presets_file = "presets.json"
        self.presets = {}
        self.create_layout()
    def bind_shortcuts(self):
        self.root.bind_all("<Control-z>", lambda e: self.undo_last_action())
        self.root.bind_all("<Control-o>", lambda e: self.toolbar.open_image())
        self.root.bind_all("<Control-s>", lambda e: self.toolbar.save_image())
        self.root.bind_all("<Left>", lambda e: self.show_previous_image())
        self.root.bind_all("<Right>", lambda e: self.show_next_image())
        self.root.bind_all("<f>", lambda e: self.show_right_panel(ProcessingPanel))
        self.root.bind_all("<a>", lambda e: self.show_right_panel(AnalysisPanel))
    def create_layout(self):
        self.create_header()
        self.create_left_panel()
        self.create_center_panel()
        self.create_right_panel()
        self.create_control_panel()
    def create_header(self):
        self.header = tk.Frame(self.root, bg="#333333", height=60, bd=2,
relief="ridge")
        self.header.pack(fill="x")
        title_label = tk.Label(
            self.header,
```

```
        text="Редактор зображень",
        bg="#404040",
        fg="#FFFFFF",
        font=self.default_font
    )
    title_label.pack(pady=10)

def create_left_panel(self):
    self.left_panel = tk.Frame(self.root, width=220, bg='#2C2C2C')
    self.left_panel.pack(side='left', fill='y', padx=10, pady=10)

    editing_buttons = [
        ('Фільтри', lambda: self.show_right_panel(ProcessingPanel)),
        ('Пресети', lambda: self.show_right_panel(PresetsPanel)),
        ('Аналіз якості', lambda: self.show_right_panel(AnalysisPanel)),
        ('Скинути', self.reset_image),
        ('Обертати', lambda: rotate_image(self)),
        ('Віддзеркалити горизонтально', lambda: flip_horizontal(self)),
        ('Віддзеркалити вертикально', lambda: flip_vertical(self)),
    ]

    resize_buttons = [
        ('Збільшити', lambda: simulate_zoom(self, "in")),
        ('Зменшити', lambda: simulate_zoom(self, "out")),
    ]

    view_buttons = [
        ('До/Після', self.toggle_comparison_mode),
    ]
```

```
    self.create_button_group(self.left_panel, "Редагування", editing_buttons)
    self.create_button_group(self.left_panel, "Зміна розміру", resize_buttons)
    self.create_button_group(self.left_panel, "Перегляд", view_buttons)

def create_flat_button(self, parent, text, command):
    btn = tk.Button(
        parent,
        text=text,
        command=command,
        bg="#3A3A3A",  # плоский темно-сірий фон
        fg="#FFFFFF",  # білий текст
        font=("Segoe UI", 12),  # можна змінити на Roboto, якщо підключиш
        relief="flat",  # без об'ємних ефектів
        borderwidth=0,
        activebackground="#5A5A5A",  # колір при наведенні
        activeforeground="#FFFFFF",
        pady=10,
        width=24,
        anchor="center")
    btn.pack(pady=6)

    def on_enter(e):
        btn.config(bg="#5A5A5A")

    def on_leave(e):
        btn.config(bg="#3A3A3A")

    btn.bind("<Enter>", on_enter)
```

```
    btn.bind("<Leave>", on_leave)
    return btn

def create_button_group(self, parent, title, buttons):

    label = tk.Label(
        parent, text=title,
        bg="#2C2C2C", fg="#AAAAAA",
        font=("Segoe UI", 10, "bold"),
        anchor="w")
    label.pack(fill="x", padx=8, pady=(15, 5))

    for text, cmd in buttons:
        self.create_flat_button(parent, text, cmd)

def create_center_panel(self):
    self.center_panel = tk.Frame(self.root, bg="#4F4F4F")
    self.processing_label = tk.Label(
        self.center_panel,
        text="",
        bg="#4F4F4F",
        fg="#FFDD55",
        font=("Segoe UI", 10, "italic")
    )
    self.processing_label.pack(pady=(0, 10))
    self.center_panel.pack(side="left", fill="both", expand=True)

    self.image_index_label = tk.Label(
        self.center_panel,
```

```
            text="Фото: 0/0",
            bg="#404040",
            fg="#FFFFFF",
            font=("Segoe UI", 12, "bold"),
            width=12,
            height=2)
        self.image_index_label.pack(pady=(10, 5))

        self.size_label = tk.Label(
            self.center_panel,
            text="Розмір: 0x0 px",
            bg="#404040",
            fg="#FFFFFF",
            font=("Segoe UI", 10))
        self.size_label.pack(pady=(0, 10))

        self.image_label = tk.Label(
            self.center_panel,
            text="Тут буде зображення",
            bg="#4F4F4F",
            fg="#FFFFFF")
        self.image_label.pack(expand=True)

    def create_right_panel(self):
        self.right_panel = tk.Frame(self.root, width=250, bg='#333333')
        self.right_panel.pack(side='right', fill='y', padx=10, pady=10)

        self.dynamic_right_panel = tk.Frame(self.right_panel, bg="#333333")
        self.dynamic_right_panel.pack(fill='both', expand=True, padx=10, pady=10)
```

```
        log_container = tk.Frame(self.right_panel, bg='#333333')
        log_container.pack(fill="both", expand=False)
        self.log_frame = tk.Frame(self.right_panel, bg="#333333")

        tk.Label(
            self.log_frame,
            text='Журнал системи',
            bg='#333333',
            fg='#FFDD55',
            font=('Arial', 12, "bold")
        ).pack(pady=(5, 0))

        self.log_text = tk.Text(self.log_frame, height=6, bg='#222222', fg='#DDDDDD',
font=('Consolas', 9))
        self.log_text.pack(fill="x", padx=5, pady=(0, 10))

        self.log_text.config(state='normal')

        self.history_panel = tk.Frame(self.right_panel, bg='#333333')
        self.history_panel.pack(fill='both', expand=True)

        tk.Label(
            self.history_panel,
            text='Історія операцій',
            bg='#333333',
            fg='#FFFFFF',
            font=('Arial', 12)).pack(pady=5)
```

```
        self.history_listbox = tk.Listbox(self.history_panel, bg='#555555',
fg='#FFFFFF')
        self.history_listbox.pack(side='left', fill='both', expand=True, padx=5, pady=5)
        self.history = HistoryManager(self.history_listbox)

        scrollbar = tk.Scrollbar(self.history_panel)
        scrollbar.pack(side='right', fill='y')
        self.history_listbox.config(yscrollcommand=scrollbar.set)
        scrollbar.config(command=self.history_listbox.yview)

    def create_control_panel(self):
        self.control_panel = tk.Frame(self.center_panel, bg="#333333", height=70,
bd=1, relief="ridge")
        self.control_panel.pack(fill="x", pady=(10, 0))

        btn_container = tk.Frame(self.control_panel, bg="#404040")
        btn_container.place(relx=0.5, rely=0.5, anchor="center")

        buttons = [
            ("⏮", self.show_first_image),
            ("←", self.show_previous_image),
            ("→", self.show_next_image),
            ("⏭", self.show_last_image)
        ]

        for text, cmd in buttons:
            btn = tk.Button(btn_container, text=text, command=cmd, bg="#555555",
fg="#FFFFFF", font=self.default_font)
```

```
        btn.pack(side="left", padx=20, pady=10)
        self.style_button(btn)

    def style_button(self, button):
        button.config(
            font=self.default_font,
            bg="#555555",
            fg="#FFFFFF",
            activebackground="#777777",
            activeforeground="#FFFFFF",
            borderwidth=0,
            highlightthickness=0,
            relief="flat"
        )
        button.bind("<Enter>", lambda e: button.config(bg="#777777"))
        button.bind("<Leave>", lambda e: button.config(bg="#555555"))
        button.bind("<ButtonPress-1>", lambda e: button.config(bg="#999999"))
        button.bind("<ButtonRelease-1>", lambda e: button.config(bg="#777777"))

    def load_image(self, file_path):
        try:
            image = Image.open(file_path)
            self.image_manager.load_image(image)
            self.update_image(self.image_manager.get_display_image())
            self.history.add(self.image_manager.base_image.copy(), "Завантажено")
            self.size_label.config(text=self.image_manager.get_size_text())
            self.reset_filter_state()
        except Exception as e:
            messagebox.showerror("Помилка", f"Не вдалося завантажити зображення:
```

```
{e}")

    def show_image(self, image):
        self.image_tk = ImageTk.PhotoImage(image)
        self.image_label.config(image=self.image_tk)
        self.image_label.image = self.image_tk  # Запобігаємо видаленню
зображення з пам'яті

    def reset_filter_state(self):
        if hasattr(self, "processing_panel") and self.processing_panel:
            self.processing_panel.filter_input_image = None
            self.processing_panel.active_filter = None

    def undo_last_action(self):
        last_state = self.history.undo()
        if last_state:
            self.image_manager.base_image = last_state
            self.image_manager.set_scale(1.0)
            self.image_manager.update_current_image()
            self.image_manager.update_display_image()
            self.update_image(self.image_manager.get_display_image())
            self.size_label.config(text=self.image_manager.get_size_text())
        else:
            messagebox.showwarning("Увага", "Немає дій для скасування!")

    def clear_history(self):
        if messagebox.askyesno("Очистити історію", "Ви дійсно хочете очистити
всю історію?"):
            self.history.reset()
```

```
            messagebox.showinfo("Історія очищена", "Усі записи історії успішно видалені.")

    def save_history_to_file(self):
        from tkinter import filedialog, messagebox
        filepath = filedialog.asksaveasfilename(
            defaultextension=".txt",
            filetypes=[("Text files", "*.txt")],
            title="Зберегти історію у файл"
        )

        if filepath:
            result = self.history.save_to_file(filepath)
            if result is True:
                messagebox.showinfo("Готово", "Історію успішно збережено.")
            else:
                messagebox.showerror("Помилка", f"Не вдалося зберегти: {result[1]}")

    def show_next_image(self):
        if self.image_files and self.current_index < len(self.image_files) - 1:
            self.current_index += 1
            self.load_image(self.image_files[self.current_index])
            self.update_image_index()
            self.reset_filter_state()

    def show_previous_image(self):
        if self.image_files and self.current_index > 0:
            self.current_index -= 1
            self.load_image(self.image_files[self.current_index])
```

```
        self.update_image_index()
        self.reset_filter_state()

def show_first_image(self):
    if self.image_files:
        self.current_index = 0
        self.load_image(self.image_files[self.current_index])
        self.update_image_index()
        self.reset_filter_state()

def show_last_image(self):
    if self.image_files:
        self.current_index = len(self.image_files) - 1
        self.load_image(self.image_files[self.current_index])
        self.update_image_index()
        self.reset_filter_state()

def update_image(self, image=None):
    if image is None:
        image = self.image_manager.get_display_image()
    self.image_tk = ImageTk.PhotoImage(image)
    self.image_label.config(image=self.image_tk)
    self.image_label.image = self.image_tk

def show_image_without_resizing(self, image):
    self.image_tk = ImageTk.PhotoImage(image)
    self.image_label.config(image=self.image_tk)
    self.image_label.image = self.image_tk
```

```
def reset_image(self):
    if self.image_manager.original_image:
        self.image_manager.reset_to_original()
        self.update_image(self.image_manager.get_display_image())
        self.history.add(self.image_manager.base_image.copy(), "Скидання")
        self.size_label.config(text=self.image_manager.get_size_text())
        self.reset_filter_state()
    else:
        messagebox.showwarning("Увага", "Зображення не завантажено!")

def update_image_index(self):
    total_images = len(self.image_files)
    current = self.current_index + 1 if self.current_index >= 0 else 0
    self.image_index_label.config(text=f"Фото: {current}/{total_images}")

def show_right_panel(self, panel_class):
    for widget in self.dynamic_right_panel.winfo_children():
        widget.destroy()

    if self.current_panel_class == panel_class:
        self.current_panel = None
        self.current_panel_class = None
        if panel_class == ProcessingPanel:
            self.processing_panel = None
        elif panel_class == AnalysisPanel:
            self.analysis_panel = None
        elif panel_class == PresetsPanel:
            self.presets_panel = None
        return
```

```
        self.current_panel = panel_class(self)
        self.current_panel_class = panel_class

        if panel_class == ProcessingPanel:
            self.processing_panel = self.current_panel
        elif panel_class == AnalysisPanel:
            self.analysis_panel = self.current_panel
        elif panel_class == PresetsPanel:
            self.presets_panel = self.current_panel

    def simulate_zoom_wrapper(self, direction):
        if self.image_manager.base_image is None:
            messagebox.showwarning("Увага", "Зображення не завантажено!")
            return
        result, new_scale = simulate_zoom(self.image_manager.current_image,
self.image_manager.scale_factor, direction)
        self.image_manager.set_scale(new_scale)
        self.manual_zoom = True
        self.image_manager.update_display_image()
        self.update_image(self.image_manager.get_display_image())

        self.history.add(self.image_manager.current_image.copy(), f"Масштабування
{direction}", round(new_scale, 2))

    def toggle_comparison_mode(self):
        if self.image_manager.original_image is None:
            messagebox.showinfo("Порівняння", "Немає оригінального зображення.")
            return
```

```
    self.comparison_mode = not self.comparison_mode
    if self.comparison_mode:
        from utils.image_utils import resize_image
        if self.image_manager.adaptive_view:
            original_resized = resize_image(
                self.image_manager.original_image,
                max_width=self.image_manager.view_width,
                max_height=self.image_manager.view_height
            )
        else:
            original_resized = self.image_manager.original_image.copy()

        self.show_image(original_resized)
    else:
        self.update_image(self.image_manager.get_display_image())

def append_log(self, message: str):
    if hasattr(self, "log_text"):
        # показати лог-панель, якщо вона ще прихована
        if not hasattr(self, "log_visible") or not self.log_visible:
            self.log_frame.pack(fill="x", padx=5, pady=(5, 0))
            self.log_visible = True
        self.log_text.insert("end", message + "\n")
        self.log_text.see("end")

def capture_from_camera(self, emulate=False):
    import cv2
```

```
        if emulate:
            test_path = "assets/test_sample.jpg"
            if not os.path.exists(test_path):
                messagebox.showerror("Емуляція", "Тестове зображення не знайдено.")
                return
            image = Image.open(test_path)
            self.image_manager.load_image(image)
            self.update_image(self.image_manager.get_display_image())
            self.history.add(image.copy(), "Емуляція з камери")
            messagebox.showinfo("Емуляція", "Завантажено тестове зображення.")
            return

        cap = cv2.VideoCapture(0)
        if not cap.isOpened():
            messagebox.showerror("Камера", "Не вдалося отримати доступ до
камери.")
            return

        ret, frame = cap.read()
        cap.release()
        if not ret:
            messagebox.showerror("Камера", "Не вдалося захопити кадр з камери.")
            return

        frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        image = Image.fromarray(frame)
        self.image_manager.load_image(image)
        self.update_image(self.image_manager.get_display_image())
        self.history.add(image.copy(), "Знімок з камери")
```

```
        messagebox.showinfo("Успіх", "Зображення з камери захоплено.")

    def automated_camera_check(self):
        import cv2
        from PIL import Image
        from utils.ai_utils import analyze_photo_ai

        cap = cv2.VideoCapture(0)
        if not cap.isOpened():
            messagebox.showerror("Камера", "Не вдалося отримати доступ до
камери.")
            return

        ret, frame = cap.read()
        cap.release()
        if not ret:
            messagebox.showerror("Камера", "Не вдалося захопити кадр.")
            return

        frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        image = Image.fromarray(frame)
        self.image_manager.load_image(image)
        self.update_image(self.image_manager.get_display_image())
        self.history.add(image.copy(), "Фото з камери (контроль якості)")

        result = analyze_photo_ai(image, mode="деталь")
        quality = result['quality_score']
        hints = result.get("hints", [])
        decision = "✅ Придатне" if quality >= 7 else "⛔ Брак"
```

```
    text = f"☎ Фото з камери\n" \
           f"Якість: {quality}/10\n" \
           f"{' / '.join(hints) if hints else 'Без критичних зауважень'}\n" \
           f"\n🔍 Рішення: {decision}"

    messagebox.showinfo("Результат контролю", text)

def automated_emulated_check(self):
    from PIL import Image
    from utils.ai_utils import analyze_photo_ai
    import os

    test_path = "assets/test_sample.jpg"
    if not os.path.exists(test_path):
        messagebox.showerror("Емуляція", "Тестове зображення не знайдено.")
        return

    image = Image.open(test_path)
    self.image_manager.load_image(image)
    self.update_image(self.image_manager.get_display_image())
    self.history.add(image.copy(), "Емуляція (контроль якості)")

    result = analyze_photo_ai(image, mode="деталь")
    quality = result['quality_score']
    hints = result.get("hints", [])
    decision = "✅ Придатне" if quality >= 7 else "⛔ Брак"

    text = f"🖼 Емуляція контролю\n" \
```

```
            f"Якість: {quality}/10\n" \
            f"{' / '.join(hints) if hints else 'Без критичних зауважень'}\n" \
            f"\n🔍 Рішення: {decision}"

        messagebox.showinfo("Результат контролю", text)

    def detect_all_defects(self):
        image = self.image_manager.base_image
        if image is None:
            messagebox.showwarning("Увага", "Зображення не завантажено!")
            return

        result_img, missing, deform, defect, final = detect_all_defects(image)
        self.image_manager.apply_transformation(result_img)
        self.update_image(self.image_manager.get_display_image())
        self.history.add(result_img.copy(), "Виявлення дефектів")
        messagebox.showinfo("Результат", f"{missing}\n{deform}\n{defect}\n\n📄
Рішення: {final}")

    def predict_gear_ai(self):
        image = self.image_manager.base_image
        if image is None:
            messagebox.showwarning("Увага", "Зображення не завантажено!")
            return

        label, confidence = self.gear_ai.predict(image)
        self.append_log(f"[AI] Результат класифікації: {label} (довіра
{confidence:.2f})")
        messagebox.showinfo("AI-аналіз шестерні", f"{label}\nДовіра:
```

```
{confidence:.2f}")

if __name__ == "__main__":
    root = tk.Tk()
    root.withdraw()
    app = ImageEditorApp(root)
    root.after(100, root.deiconify)
    root.mainloop()
```

Додаток В

Демонстраційний матеріал

№ докумен-та	Позначення	Найменування	Додаткові відомості
		Текстові документи	
1	ГЮИК. 503320.006 ПЗ	Пояснювальна записка	А4, 106 с.
		Додаткові матеріали	
2		Апробація результатів	А4, 10 с.
		кваліфікаційної роботи	
3		Лістинг програми Python	А4, 22 с.
4		Демонстраційний матеріал у	А4, 15 с.
		вигляді презентації	

Змін.	Арк.	Номер докум.	Підпис	Дата	ГЮИК. 503320.006 ВД		
Розроб.		Конєва А.І.			Тема	Літера	Аркуш	Аркушів
Перевір.		Сотник С.В.				Н	1	1
						Кафедра КІТАР ХНУРЕ		
Н.контр.		Стародубцев М.Г.						
Затв.		Невлюдов І.Ш.						