УДК 004.4 DOI https://doi.org/10.35546/kntu2078-4481.2025.2.2.35

М. ПИЛИПЧУК
магістр кафедри комп'ютерних наук
Західноукраїнський національний університет
ORCID: 0009-0000-7846-9413

Н. П. ПОРПЛИЦЯ
кандидат технічних наук,
доцент кафедри комп'ютерних наук
Західноукраїнський національний університет
ORCID: 0000-0003-3624-275X

І. С. СТАСІВ
кандидат технічних наук,
доцент кафедри комп'ютерних наук
Західноукраїнський національний університет
ORCID: 0000-0003-4181-3916

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ АНАЛІЗУ ДОСТУПНОСТІ ВЕБ-КОНТЕНТУ ДЛЯ ЛЮДЕЙ З ВАДАМИ ЗОРУ

У статті розглядається проблема впровадження сучасних принципів цифрової інклюзивності до цифрового веб-контенту. Показано, що ключовим фактором, що впливає на доступність веб-ресурсу для людей з вадами зорового сприйняття є рівень контрастності його веб-сторінок та їх наповнення, включаючи й елементи графічного інтерфейсу користувача. Запропоновано оригінальний підхід до автоматизованого аналізу контрастності веб-контенту, відповідно до вимог стандарту WCAG 2.1 без доступу DOM-структури чи HTML-коду веб-сторінки, а з використанням її графічних зображень. Для його програмної реалізації було використано такі популярні інструменти: Python, OpenCV та Tesseract OCR. Для підвищення швидкодії розроблюваного програмного забезпечення, в його основу було закладено принципи паралельних обчислень для розподіленої обробки виділених блоків тексту між кількома потоками. Для цього було використано бібліотеку concurrent.futures, яка дозволяє створювати потоки, що паралельно обробляють різні частини зображення, у яких було ідентифіковано текст. Розроблений модуль продемонстрував здатність ефективно аналізувати контрастність елементів веб-сторінок без доступу до їх DOM-структури, що робить його придатним для автоматизованого аудиту доступності зображень, PDF-документів, веб-інтерфейсів та інших графічних елементів. Це було підтверджено під час тестування розробленого програмного модуля на ряді реальних веб-сайтів, наприклад, веб-ресурс Medium (режим доступу: https://medium.com/). Також було проведено порівняльне тестування розробленого модуля з відомими аналогами, зокрема, Google Vision API, Adobe Acrobat OCR та Tesseract + Custom Scripts та проведено аналіз його результатів. Тестування було проведено із використанням вибірки, що складалася із різнотипних зображень різних розмірів за такими критеріями: час обробки зображень, точність визначення текстових блоків і контрастності, стабільність роботи.

Ключові слова: *інклюзивність цифрового контенту, WCAG, аналіз контрастності, програмний продукт, паралельна архітектура, хмарні обчислення, управління проєктом.*

M. PYLYPCHUK
Master at the Department of Computer Science
West Ukrainian National University
ORCID: 0009-0000-7846-9413

N. P. PORPLYTSYA
Associate Professor at the Department of Computer Science
West Ukrainian National University
ORCID: 0000-0003-3624-275X

I. S. STASIV
Associate Professor at the Department of Computer Science
West Ukrainian National University
ORCID: 0000-0003-4181-3916

SOFTWARE TOOL FOR WEB CONTENT ACCESSIBILITY ANALYSIS FOR VISUALLY IMPAIRED USERS

The article addresses the issue of implementing modern principles of digital inclusivity in web content. It is shown that a key factor affecting the accessibility of a web resource for people with visual impairments is the contrast level of its web pages and their content, including elements of the graphical user interface. An original approach is proposed for automated contrast analysis of web content in accordance with the Web Content Accessibility Guidelines (WCAG) 2.1 standard, without accessing the Document Object Model (DOM) structure or HTML code of the web page, using only its graphical representation. To implement this approach, the following popular tools were used: programming language Python, OpenCV, and Tesseract Optical Character Recognition. To improve the performance of the developed software, principles of parallel computing were implemented at its core to enable distributed processing of designated text blocks across multiple threads. For this purpose, the concurrent.futures library was used, allowing the creation of threads that process different parts of the image – where text has been identified – in parallel. The developed module demonstrated its ability to effectively analyze the contrast of web page elements without access to their DOM structure, making it suitable for automated accessibility audits of images, PDF documents, web interfaces, and other graphical elements. This was confirmed during testing of the module on several real websites, such as the Medium platform (access mode: https://medium.com/). Comparative testing was also conducted between the developed module and well-known alternatives, including Google Vision API, Adobe Acrobat OCR, and Tesseract with custom scripts. The results were analyzed based on a dataset consisting of various types of images of different sizes, according to the following criteria: image processing time, accuracy of text block and contrast detection, and operational stability.

Key words: *digital content inclusivity, Web Content Accessibility Guidelines, contrast analysis, software product, parallel architecture.*

Постановка проблеми

Доступність інформаційного контенту є важливим аспектом для реалізації ергономічних веб-ресурсів, зокрема, коли це стосується користувачів з вадами зору. Як відомо, базовим критерієм візуальної доступності інформаційного контенту для таких користувачів є забезпечення належного рівня контрастності між текстом та фоном, адже саме це безпосередньо впливає на здатність користувача сприймати, читати й інтерпретувати інформацію. Сьогодні існують вже чітко означені вимоги, щодо критеріїв доступності інформаційного контенту для людей з інвалідністю, вони розроблені міжнародною організацією W3C та подані у стандарті WCAG 2.1 (Web Content Accessibility Guidelines) [11, 12]. Однак, незважаючи на існування таких стандартів, багато веб-сайтів і досі не відповідають встановленим нормам через складність їхнього впровадження або недостатню обізнаність розробників. Традиційні методи оцінки контрастності, такі як використання онлайн-інструментів (наприклад, WAVE, axe, Lighthouse), часто вимагають втручання людини або мають обмеження у визначенні тексту, який є частиною зображень [2, 5]. Крім того, багато таких інструментів базуються саме на використанні хмарних обчислень, залежних від зовнішніх ресурсів, що суттєво впливає на вартість їх застосування. Особливо, якщо масштабувати застосування такого ресурсу для аналізу великої кількості існуючих веб-ресурсів та з урахуванням стрімкого оновлення їх контенту. Саме тому, розробка програмного рішення на основі методів комп'ютерного зору із застосуванням паралельних обчислень є перспективним напрямом досліджень, що дозволить автоматизувати аналіз контрастності без необхідності доступу до вихідного коду веб-сторінок та дозволить підвищити рівень впровадження сучасних принципів цифрової інклюзивності [3, 4, 6, 7]. Крім того, цей напрям досліджень відповідає практичним потребам розробників програмного забезпечення, які прагнуть автоматизувати аудит доступності зображень, PDF-документів, веб-інтерфейсів та інших графічних елементів [1, 2, 8].

У зв'язку з цим, дана робота присвячена розпаралеленню та оптимізації методу аналізу контрастності тексту на зображеннях, з метою підвищення ефективності й точності систем розпізнавання тексту, що працюють у відповідності з вимогами WCAG 2.1.

Аналіз останніх досліджень і публікацій

Останні дослідження у сфері програмного забезпечення для аналізу для перевірки контрастності на веб-контенту показують активний розвиток інструментів, що перевіряють відповідність стандарту WCAG 2.1 [11, 12]. Зокрема, до них належать: Axe-core, WAVE, Google Lighthouse [5]. Однак, усі вони переважно працюють з DOM-структурою веб-сторінок. Водночас, якщо текст відображається динамічно, за допомогою Canvas/WebGL, або має стилі, накладені через JavaScript – ці інструменти можуть не виявити проблему, також вони не враховують «реальне зображення» сторінки, тобто не бачать, як вона виглядає користувачеві, а також вони не можуть застосовуватися для перевірки PDF-документів, скріншотів або веб-програм без доступу до вихідного коду.

Саме тому, актуальною є задача розробки програмного забезпечення, що базується на використанні комп'ютерного зору та OCR (Optical Character Recognition) для аналізу контрастності без залежності від DOM-структури чи HTML-коду.

Формулювання мети дослідження

Мета дослідження полягає у розробці ефективного інструменту для аналізу контрастності тексту на зображеннях (без доступу DOM-структури чи HTML-коду) відповідно до вимог WCAG 2.1, в основу якого буде закладено методи комп'ютерного зору та принципи паралельних обчислень.

Викладення основного матеріалу дослідження

1. Підхід до аналізу контрастності без залежності від DOM-структури чи HTML-коду веб-сторінки

Контрастність у WCAG вимірюється як відношення яскравості кольорів, і є ключовим фактором, що забезпечує читабельність та розпізнаваність веб-контенту, а особливо для користувачів із порушенням зору або дальтонізмом [2, 8, 11, 12].

Контрастність визначається як співвідношення відносної яскравості (luminance) двох кольорів – кольору тексту (L1) та фону (L2). Формально:

$$Contrast\ Ratio = \frac{L_2 + 0.05}{L_1 + 0.05}, \tag{1}$$

де L1 – більша з яскравостей (текст або фон), L2 – менша з яскравостей.

Яскравість кольору обчислюється з RGB-компонентів згідно з формулою:

$$L = 0.2126 \cdot R_{lin} + 0.7152 \cdot G_{lin} + 0.0722 \cdot B_{lin}, \tag{2}$$

де кожен компонент $C_{lin}(R, G, B)$ лінійно перетворюється з sRGB у лінійний RGB за формулою:

$$\begin{cases} \frac{C_{srgb}}{12.92} 3 \\ \left(\frac{C_{srgb} + 0.055}{1.055}\right), \end{cases} \tag{3}$$

де $C_{srgb} \in [0, 1]$, тобто значення кожного кольору в шкалі від 0 до 1.

У цій праці пропонується застосувати ефективне поєднання сучасних методів комп'ютерного зору та інструментів оптичного розпізнавання тексту для аналізу контрастності веб-контенту. Одним із сучасних потужних інструментів комп'ютерного зору, що дозволяє ефективно працювати з зображеннями є бібліотека OpenCV [9, 10]. У контексті аналізу контрастності OpenCV дозволяє: отримати скріншот веб-сторінки у вигляді зображення (наприклад, за допомогою Selenium або Puppeteer) та надає інструменти для обробки цього зображення; дозволяє визначити середній колір тексту та колір фону у кожній такій ділянці; перетворити кольори з формату BGR (або RGB) у яскравість за формулами WCAG та обчислити значення коефіцієнту контрастності.

Розглянемо також Tesseract OCR – це популярний інструмент для оптичного розпізнавання тексту (Optical Character Recognition), що дозволяє ідентифікувати та розпізнавати текст безпосередньо із зображення. Його застосування для аналізу контрастності веб-контенту дозволить автоматично визначати розташування текстових блоків на веб-сторінці, виділяти координати фрагменту зображення, що містить текст, забезпечити ефективну взаємодію з OpenCV для подальшого аналізу кольору в межах ідентифікованого фрагменту зображення. Таким чином, поєднання Tesseract OCR і OpenCV дозволяє реалізувати автоматизований аналіз контрастності, без доступу до HTML-коду чи CSS стилів поточної веб-сторінки. Для реалізації проєкту було використано мову програмування Python.

Комбінація методів комп'ютерного зору дозволяє реалізувати ефективну та масштабовану систему перевірки контрастності, що може працювати із будь-якими веб-сторінками у форматі зображень. Такий підхід може використовуватись для моніторингу державних чи освітніх ресурсів, перевірки дизайну перед запуском або інтеграції в автоматизовані системи тестування доступності.

2. Опис модулів програмної реалізації та аналіз проблемних моментів

Архітектура модуля для автоматичного визначення контрастності веб-контенту побудована з урахуванням потреби в ефективній обробці зображень веб-сторінок, точному визначенні кольорів тексту та фону, а також в обчисленні контрасту між ними. Загальна структура модуля складається з кількох основних складових, кожна з яких виконує специфічну функцію в процесі аналізу контрасту веб-контенту.

Модуль збору даних (Скринінг веб-сторінок): модуль починає свою роботу з отримання доступу до веб-сторінки, яка буде проаналізована. Для цього використовується Selenium, що дозволяє взаємодіяти з веб-браузером для автоматичного збору контенту сторінки. Selenium відкриває веб-сторінку, виконує її рендеринг, а потім робить скріншот, що є важливою частиною підготовки даних для подальшого аналізу.

Модуль обробки зображень (OpenCV): після того, як зображення веб-сторінки отримано, воно передається в компонент обробки зображень. Для цього використовуються бібліотеки OpenCV та Tesseract OCR. OpenCV займається попередньою обробкою зображення: покращенням якості (підвищення контрасту, видалення шумів),

а також сегментацією зображення для виділення тексту та фону. За допомогою цієї бібліотеки можна виділяти області, що містять текст, і підготовляти їх для наступного аналізу.

Модуль аналізу тексту та фону (Tesseract OCR): Tesseract OCR виконує оптичне розпізнавання символів на зображенні. Він розпізнає текстові елементи, які містяться на скріншоті, та надає їх координати та текстові значення. Крім того, Tesseract надає інформацію про розміри блоків тексту, що дозволяє точно визначити межі кожного текстового елементу. Також, проведено оптимізацію самого процесу розпізнавання, зокрема, відповідним чином налаштовано параметри Tesseract для зменшення часу виконання. Наприклад, параметр --psm (Page Segmentation Mode), можне впливати на те, як Tesseract розпізнає текст на зображенні. Наприклад, для документів з одним текстовим блоком можна вибрати відповідний режим, що дозволить зменшити час обробки. А параметр output_type в pytesseract.image_to_data можна встановити на output_type.STRING, щоб обмежити кількість інформації, яка повертається, якщо це не є необхідним. Це також зменшує обсяг обробленої інформації та прискорює роботу.

Загалом, варто відзначити, що застосування попередньої обробки зображення та вибіркової обробки суттєво знижує навантаження на систему та покращує швидкість обробки, зберігаючи точність розпізнавання тексту, особливо при роботі з «великими» зображеннями чи документами.

Модуль аналізу контрасту (Розрахунок контрасту): після того, як текст і фон виділено, наступним кроком є аналіз контрасту між текстом і фоном. Для цього використовується спеціальний алгоритм обчислення контрасту на основі відносної люмінесценції. Визначення контрасту між текстом і фоном здійснюється шляхом порівняння їхніх відтінків, що дає можливість оцінити відповідність стандартам доступності контенту.

Однією з проблем, що виникають при обробці зображень, є послідовне оброблення кожного блоку тексту. У базовій версії програми, кожен блок тексту обробляється по черзі, що може значно збільшувати час виконання при наявності великої кількості текстових елементів. Щоб подолати цю проблему, можна застосувати паралельну обробку, що дозволяє скоротити час виконання програми, розподіляючи навантаження між кількома процесорами або потоками. Для підвищення швидкодії розроблюваного програмного забезпечення було застосовано принципи паралельних обчислень для розподілу обробки блоків тексту між кількома потоками. Для цього було використано бібліотеку concurrent.futures для створення потоків, що паралельно обробляють різні частини зображення.

Зазначимо, що паралельне виконання операцій було застосовано до розпізнавання тексту та визначення областей малюнка, де ідентифіковано текст. Зокрема, алгоритм pytesseract було адаптовано для паралельного виконання на різних частинах зображення, кожна з яких обробляється окремо. Це дозволяє розпізнавати текст у кількох областях зображення одночасно, зменшуючи загальний час обробки.

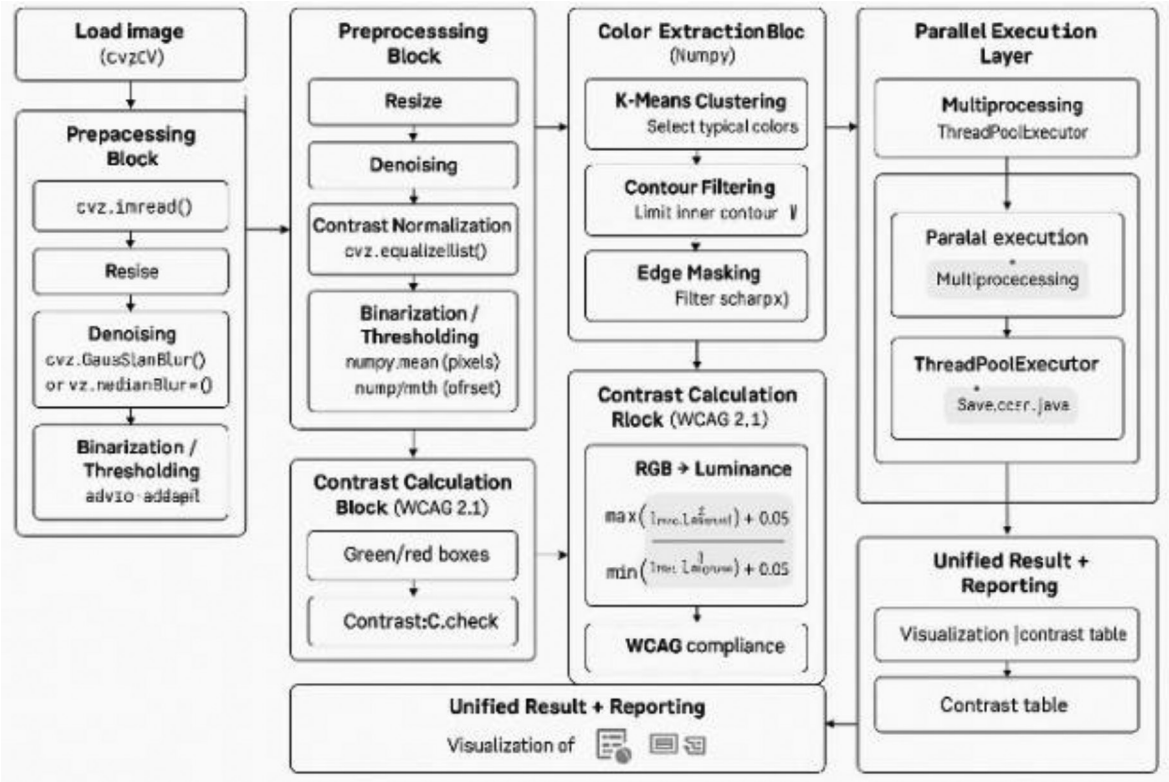

Рис. 1. Блок-схема паралельної реалізації аналізу контрастності веб-контенту

Також, паралельне виконання операцій було застосовано для визначення кольорів тексту та фону. Тобто, для кожного блоку тексту, паралельно обчислюємо колір тексту та фону, на основі виділених частин зображення на етапі розпізнавання тексту. Тоді проводимо розрахунок контрастності для кожного текстового блоку паралельно. Після завершення усіх етапів паралельної обробки, результати (в тому числі обчислені значення контрастності для кожного блоку тексту) об'єднуються, та в такий спосіб формуємо в кінцевий результат – звіт про контрастність зображення. Цей етап є фінальним і може бути виконаний після того, як всі потоки завершені.

Впровадження принципів паралельних обчислень дозволяє суттєво підвищити швидкодію програмного забезпечення та водночас забезпечити ефективну обробку «великих» зображень.

3. Тестування та аналіз розробленого програмного забезпечення

Розглянемо детальніше приклад застосування розробленого програмного продукту для реальних веб-сторінок. Зокрема, для тестування було обрано веб-ресурс Medium (режим доступу: https://medium.com/), де подано список блогів та розміщено типові елементами графічного інтерфейсу користувача (заголовки, кнопки, посилання тощо). Вхідне зображення однієї з веб-сторінок веб-ресурсу Medium подано на рис. 2.

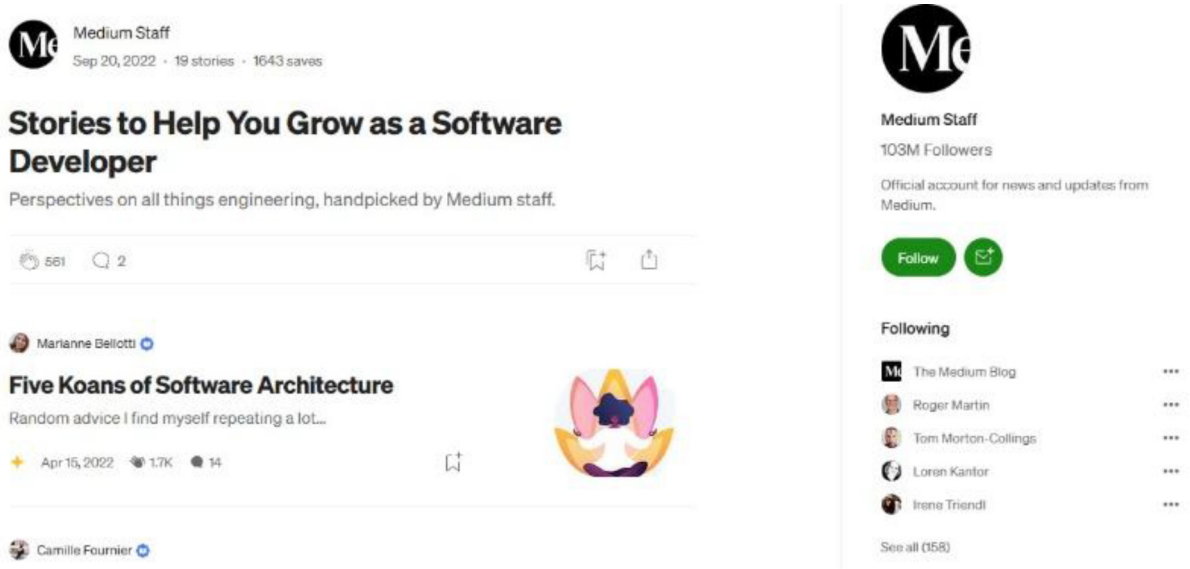

Рис. 2. Вхідне зображення веб-сторінки

На рис. 3 проілюстровано результати аналізу контрастності, отримані із використанням розробленого програмного забезпечення, що демонструють різні елементи веб-сторінки, що не відповідають встановленим стандартам контрастності WCAG 2.1. Отримані результати вказують на значні проблеми з рівнями контрастності у даного веб-ресурсу, наприклад, деякі посилання та кнопки на сторінці мають дуже низький контраст на фоні, що ускладнює їх візуальне сприйняття, а особливо для людей з вадами зору.

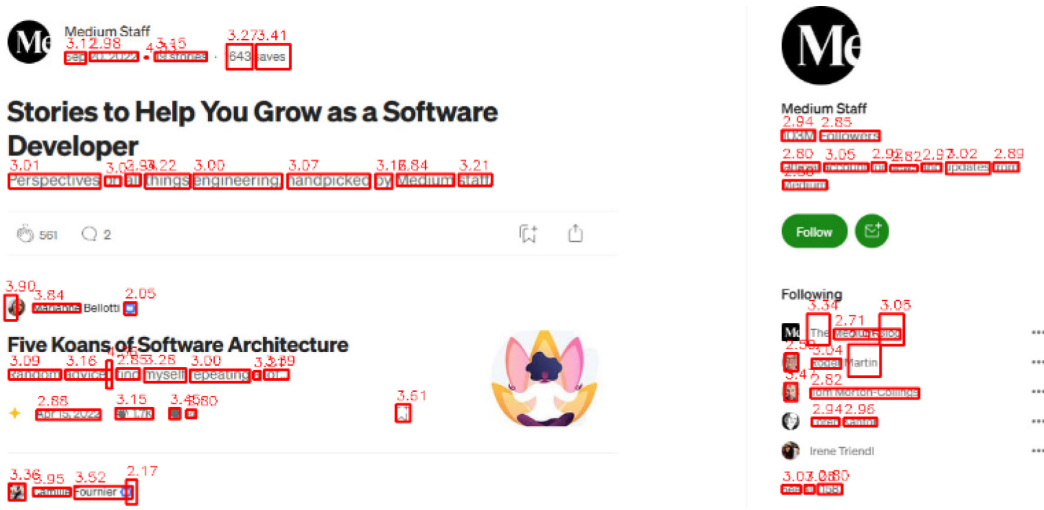

Рис. 3. Вихідне зображення веб-сторінки (звіт про контрастність зображення)

Для об'єктивної оцінки ефективності роботи програмного продукту, а також їхнього порівняння з існуючими аналогами, було розроблено спеціальну методику тестування. Вона містить чітко визначені критерії оцінки, стандартизовані тестові дані та однакові умови проведення випробувань, зокрема такі критерії: час обробки зображення; точність розпізнавання тексту та обчислення; стабільність роботи.

Під час тестування було використано вибірку зображень, що охоплює різні сценарії використання: зображення невеликого розміру (до 1 МБ) із чітким текстом; зображення середнього розміру (1–5 МБ) із помірною кількістю текстових елементів; великі зображення (понад 5 МБ) із численними блоками тексту різного розміру та контрастності; зображення з низьким рівнем контрастності між текстом і фоном.

Для тестування було використано таке апаратне забезпечення: процесор: Intel Core i7; оперативна пам'ять: 16 ГБ RAM; накопичувач: SSD-диск; програмне середовище: Windows 11 x64; версії бібліотек OpenCV, Pytesseract, NumPy – однакові для всіх тестів.

Кожен тест виконувався не менше трьох разів для мінімізації впливу випадкових затримок системи, після чого обчислювалося середнє значення показників. Для порівняння були обрано відкриті бібліотеки та сервіси, що виконують автоматичне розпізнавання тексту на зображеннях і оцінку контрастності між текстом і фоном. Зокрема: Google Vision API – комерційний сервіс із підтримкою OCR і аналізу зображень; Adobe Acrobat OCR + Accessibility Checker – функціональність для сканування документів і перевірки доступності, включно з оцінкою контрастності; Tesseract + Custom Scripts – комбінація відкритого OCR-движка Tesseract із самостійними скриптами для оцінки контрастності. Ці рішення були обрані через їхню доступність і популярність у сфері аналізу зображень і забезпечення доступності.

Спершу розглянемо результати тестування за критерієм «Час обробки зображень».

Таблиця 1

Порівняння за критерієм «Час обробки зображень»

Рішення	Маленьке зображення (с)	Середнє зображення (с)	Велике зображення (с)
Власна розробка	1.2	3.1	7.6
Google Vision API	0.9	2.8	5.5
Adobe Acrobat OCR + Accessibility Checker	1.5	4.0	8.2
Tesseract + Custom Scripts	1.8	5.2	11.0

Аналіз результатів тестування показав, що Google Vision API працює найшвидше, що пов'язано із використанням зовнішніх ресурсів, зокрема засобів хмарних обчислень. Власна розробка поступається Google Vision API, але випереджає інші відкриті рішення.

Далі розглянемо результати тестування за критерієм «Точність визначення текстових блоків і контрастності».

Таблиця 2

Порівняння за критерієм «Точність визначення текстових блоків і контрастності»

Рішення	Маленьке зображення (%)	Середнє зображення (%)	Велике зображення (%)
Власна розробка	93.7	91.5	87.6
Google Vision API	95.2	94.1	90.5
Adobe Acrobat OCR + Accessibility Checker	92.0	89.8	85.0
Tesseract + Custom Scripts	90.5	87.3	82.4

Власна розробка демонструє результати, близькі до комерційного продукту Google, що підтверджує вдалу програмну реалізацію продукту та ефективне впровадження принципів паралельних обчислень. Водночас, Adobe Acrobat і Tesseract мають нижчу точність, особливо при аналізі великих або складних зображень.

І останнім розглянемо результати тестування за критерієм «Стабільність роботи».

Таблиця 3

Порівняння за критерієм «Стабільність роботи»

Рішення	Кількість збоїв на 100 обробок
Власна розробка	0–1
Google Vision API	0
Adobe Acrobat OCR + Accessibility Checker	2
Tesseract + Custom Scripts	5

Google Vision API повністю стабільний завдяки обробці на стороні сервера, водночас, власна розробка продемонструвала дуже високу стабільність, близьку до Google Vision API. При цьому, Tesseract + Custom Scripts має суттєві проблеми при обробці великих і складних файлів.

Висновки

У цій роботі реалізовано та протестовано запропонований підхід до автоматизованого аналізу контрастності тексту на зображеннях, відповідно до вимог стандарту WCAG 2.1 на основі проведеної його програмної реалізації.

Основними етапами роботи були: розробка підходу для визначення контрасту між текстом і фоном, використання популярних інструментів, таких як Python, OpenCV та Tesseract OCR, а також тестування розробленого модуля на реальних веб-сторінках, зокрема на веб-сайті Medium.

У порівнянні з відомими аналогами, зокрема Google Vision API, Adobe Acrobat OCR та Tesseract + Custom Scripts, розроблений підхід показав конкурентоспроможні результати, поступаючись комерційним хмарним сервісам в незначній мірі, проте значно перевершуючи локальні альтернативи як за стабільністю, так і за адаптивністю до складного фону.

Таким чином, запропонована паралельна архітектура та алгоритмічна реалізація забезпечують ефективний, точний і автономний інструмент для автоматизованого аналізу контрастності, який може бути корисним як для аудиту цифрової доступності, так і для інтеграції в більші системи контролю якості контенту.

Список використаної літератури

1. A. Gubbi Mohanbabu and A. Pavel. Context-Aware Image Descriptions for Web Accessibility. *arXiv preprint arXiv:2409.03054*, 2024. [Online]. Available: https://arxiv.org/abs/2409.03054

2. J. L. Chodrow,.Automated assessment of web accessibility using computer vision. 2020. [Online]. Available: https://arxiv.org/abs/2007.01830

3. I. Madiudia, N. Porplytsya and M. Nagara. Mathematical Model for Prediction the Dynamics of Organic Traffic at E-commerce Web-site in the Process of its Search Engine optimization. *2020 10th International Conference on Advanced Computer Information Technologies (ACIT)*, Deggendorf, Germany, 2020, pp. 577-580, doi: 10.1109/ACIT49673.2020.9208886

4. K. Zelenetska, N. Porplytsya, I. Stasiv, S. Stańczyk, A. Jankowiak and L. Bilovus. SEO-Optimization of Product Content on a Marketplace Platform. *2023 13th International Conference on Advanced Computer Information Technologies (ACIT)*, Wrocław, Poland, 2023, pp. 201–205, doi: 10.1109/ACIT58437.2023.10275590

5. Lighthouse: Audits for performance, accessibility, and SEO. 2021. [Online]. Available: https://developers.google.com/web/tools/lighthouse.

6. M. Dyvak, I. Darmorost, N. Porplytsya and I. Hural. Structure Identification of Difference Equations with Interval Estimates of their Parameters. *2019 IEEE 15th International Conference on the Experience of Designing and Application of CAD Systems (CADSM)*, Polyana, Ukraine, 2019, pp. 1-4, doi: 10.1109/CADSM.2019.8779308

7. M. Dyvak, A. Pukas, N. Porplytsya, I. Oliinyk and P. Basistyi. Method of Structural Identification the Interval Models of Static Objects. *2021 11th International Conference on Advanced Computer Information Technologies (ACIT)*, Deggendorf, Germany, 2021, pp. 105-110, doi: 10.1109/ACIT52158.2021.9548379

8. M. Zhang et al. Are your apps accessible? A GCN-based accessibility checker for low vision users. *arXiv preprint arXiv:2502.14288*, 2025. [Online]. Available: https://arxiv.org/abs/2502.14288

9. OpenCV Documentation. Image Thresholding. [Online]. Available: https://docs.opencv.org/4.x/d7/d4d/tutorial_py_thresholding.html.

10. OpenCV Documentation. Contours: Getting Started. [Online]. Available: https://docs.opencv.org/3.4/d4/d73/tutorial_py_contours_begin.html

11. Web Content Accessibility Guidelines (WCAG) 2.1. 2018. [Online]. Available: https://www.w3.org/TR/WCAG21/

12. Web Content Accessibility Guidelines (WCAG) 2.0. 2008. [Online]. Available: https://www.w3.org/TR/WCAG20/

References

1. A. Gubbi Mohanbabu and A. Pavel. Context-Aware Image Descriptions for Web Accessibility. *arXiv preprint arXiv:2409.03054*, 2024. [Online]. Available: https://arxiv.org/abs/2409.03054

2. J. L. Chodrow,.Automated assessment of web accessibility using computer vision. 2020. [Online]. Available: https://arxiv.org/abs/2007.01830

3. I. Madiudia, N. Porplytsya and M. Nagara. Mathematical Model for Prediction the Dynamics of Organic Traffic at E-commerce Web-site in the Process of its Search Engine optimization. *2020 10th International Conference on Advanced Computer Information Technologies (ACIT)*, Deggendorf, Germany, 2020, pp. 577-580, doi: 10.1109/ACIT49673.2020.9208886

4. K. Zelenetska, N. Porplytsya, I. Stasiv, S. Stańczyk, A. Jankowiak and L. Bilovus. SEO-Optimization of Product Content on a Marketplace Platform. *2023 13th International Conference on Advanced Computer Information Technologies (ACIT)*, Wrocław, Poland, 2023, pp. 201-205, doi: 10.1109/ACIT58437.2023.10275590

5. Lighthouse: Audits for performance, accessibility, and SEO. 2021. [Online]. Available: https://developers.google.com/web/tools/lighthouse

6. M. Dyvak, I. Darmorost, N. Porplytsya and I. Hural. Structure Identification of Difference Equations with Interval Estimates of their Parameters. *2019 IEEE 15th International Conference on the Experience of Designing and Application of CAD Systems (CADSM)*, Polyana, Ukraine, 2019, pp. 1-4, doi: 10.1109/CADSM.2019.8779308

7. M. Dyvak, A. Pukas, N. Porplytsya, I. Oliinyk and P. Basistyi. Method of Structural Identification the Interval Models of Static Objects. *2021 11th International Conference on Advanced Computer Information Technologies (ACIT)*, Deggendorf, Germany, 2021, pp. 105-110, doi: 10.1109/ACIT52158.2021.9548379

8. M. Zhang et al. Are your apps accessible? A GCN-based accessibility checker for low vision users. *arXiv preprint arXiv:2502.14288*, 2025. [Online]. Available: https://arxiv.org/abs/2502.14288

9. OpenCV Documentation. Image Thresholding. [Online]. Available: https://docs.opencv.org/4.x/d7/d4d/tutorial_py_thresholding.html

10. OpenCV Documentation. Contours: Getting Started. [Online]. Available: https://docs.opencv.org/3.4/d4/d73/tutorial_py_contours_begin.html

11. Web Content Accessibility Guidelines (WCAG) 2.1. 2018. [Online]. Available: https://www.w3.org/TR/WCAG21/

12. Web Content Accessibility Guidelines (WCAG) 2.0. 2008. [Online]. Available: https://www.w3.org/TR/WCAG20/