Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня
магістра
(назва освітньо-кваліфікаційного рівня)

студента	*Пчелєнкова Івана Сергійовичі*
	(ПІБ)
академічної групи	*121М-20-1*
	(шифр)
спеціальності	*121 Інженерія програмного забезпечення*
	(код і назва спеціальності)
освітньої програми	*«Інженерія програмного забезпечення»*
	(назва освітньої програми)
на тему:	*Методи, алгоритми та інформаційна технологія розпізнавання штрих-кода EAN-13 стійкого до спотворень*

_____*І.С. Пчелєнков*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтин-говою	**інституційною**	
розділів кваліфікаційної роботи				
спеціальний				
економічний				
Рецензент				
Нормоконтролер				

Дніпро
2022

Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
Завідувач кафедри
Програмного забезпечення комп'ютерних систем
(повна назва)

_____ І.М. Удовик
(підпис) (прізвище, ініціали)

« » _____ 20 22 Року

ЗАВДАННЯ
на виконання кваліфікаційної роботи

спеціальності *121 Інженерія програмного забезпечення*
(код і назва спеціальності)

студенту *121м-20-1* *Пчелєнкову Івану Сергійовичу*
(група) (прізвище та ініціали)

Тема кваліфікаційної роботи *Методи, алгоритми та інформаційна технологія розпізнавання штрих-кода EAN-13 стійкого до спотворень*

1 ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Наказ ректора НТУ «Дніпровська політехніка» від __.__.____ р. № _____

2 МЕТА ТА ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Об'єкт досліджень – процес обробки зображення, та декодування рядка пікселів.

Предмет досліджень – моделі, методи та інформаційна технологія забезпечення якості декодування штрих-коду EAN-13 при наявності спотворень.

Мета НДР – полягає в підвищення ефективності роботи розпізнавання штрих-коду EAN-13 на реальних зображеннях

Вихідні дані для проведення роботи – зображення штрих-кодів, з наявностями дефектів які можуть завадити, при зчитуванні даних.

3 ОЧІКУВАНІ НАУКОВІ РЕЗУЛЬТАТИ

Новизна запропонованих рішень визначається тим, що розроблено і обґрунтовано новий підхід до розпізнавання зображень штрих-кодів.

Практична цінність результатів полягає у тому, щоб використовувати результати роботи отримані в ході дослідження, для написання додатків будь-якого характеру, зчитування штрих-кодів стандарту EAN-13.

4 ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Результати досліджень мають бути подані у вигляді, що дозволяє побачити та оцінити безпосереднє використання алгоритму декодування. В результаті роботи повинен бути розроблений програмний комплекс для декодування штрих-коду EAN-13 при наявності спотворень.

5 ЕТАПИ ВИКОНАННЯ РОБІТ

Найменування етапів робіт	**Строки виконання робіт (початок – кінець**
Аналіз теми та постановка задачі	12.09.2021-30.09.2021
Побудова нечіткої моделі представлення даних для вирішення задачі ідентифікаційної експертизи	01.10.2021-31.10.2021
Створення автоматизованої системи для вирішення задачі ідентифікаційної експертизи бензинів	01.11.2021-16.12.2021

6 РЕАЛІЗАЦІЯ РЕЗУЛЬТАТІВ ТА ЕФЕКТИВНІСТЬ

Економічний ефект від реалізації результатів роботи очікується позитивним завдяки скорочення затрат на заробітну плату людям, які виконують сканування штрих-кодів.

Соціальний ефект від реалізації результатів роботи очікується позитивним завдяки удосконаленню метода розпізнавання штрих-коду, що дозволяє зменшити похибку декодування при наявності спотворень.

7 ДОДАТКОВІ ВИМОГИ

Завдання видав _____ _____
(підпис) (прізвище, ініціали)

Завдання прийняв до виконання _____ *Пчелєнков І.С.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: _____.

Термін подання кваліфікаційної роботи до ЕК *19.01.2022*

РЕФЕРАТ

Пояснювальна записка: 50 с., 16 рис., 1 табл, 14 дж., 4 додатків.

Об'єкт дослідження: процес обробки, розпізнавання та перетворення зображення, декодування отриманого в процесі рядка пікселів.

Предмет дослідження: моделі, методи та інформаційна технологія забезпечення якості декодування штрих-коду EAN-13 при наявності спотворень.

Мета роботи: полягає в підвищення ефективності роботи розпізнавання штрих-коду EAN-13 зі спотвореннями, які виникають на реальних зображеннях.

Методи дослідження. Для виконання поставлених завдань були використані наступні методи та інструменти:

- Методи пошуку та декодування штрих-коду;
- PyCharm, Python, OpenGL.

Наукова новизна полягає в розробці нового підходу до розпізнавання та декодування штрих-кодів. Цей підхід обґрунтований вибором декодуючого рядку даних, цей підхід дозволяє прискорити розшифрування на спотворених зображеннях.

Практичне значення полягає в тому, що результати роботи, отримані в ході дослідження, можна використовувати для написання додатків будь-якого характеру, для зчитування штрих-кодів стандарту EAN-13.

У розділі «Економіка» проведено розрахунки трудомісткості розробки програмного забезпечення, витрат на створення ПО і тривалості його розробки.

Список ключових слів: штрих-код, EAN-13, ПЗ, бінаризація, декодування, розшифрування, Python, PyCharm, IDE.

ABSTRACT

Explanatory note: 70 p., 40 pic., 2 tables, 2 add., 44 sources.

Object of research: the process of processing, recognition and conversion of the image, decoding the resulting pixel string.

Subject of research: models, methods and information technology to ensure the quality of EAN-13 barcode decoding in the presence of distortion.

Purpose of Master's thesis: is to improve the efficiency of EAN-13 barcode recognition with distortion that occurs in real images.

Research methods. The following methods and tools were used to perform the tasks:

• Bar code search and decoding methods;

• PyCharm, Python, OpenGL.

Originality of research is to develop a new approach to the recognition and de-coding of barcodes. This approach is based on the choice of decoding line of data, this approach allows you to speed up decryption on distorted images.

The practical value is that the results obtained during the study can be used to write applications of any nature, to read barcodes of the EAN-13 standard.

In the Economics calculations of the complexity of software development, the cost of creating software and the duration of its development.

Keywords: barcode, EAN-13, software, binarization, decoding, decryption, Python, Py-Charm, IDE.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ПЗ – програмне забезпечення

БД – база даних

WLAN

Штрих-код – штриховий код стандарту EAN-13

EAN-13/ UCC – Європейська система кодування

UPC – Американський універсальний товарний код

IDE – Інтегрована середа розробки

ВСТУП

Часто беручи до рук якось упаковку зубної пасти, шампуня, миючого засобу або прального порошку ви зустрічаєте набір паралельно розташованих чорних ліній різної товщини та довжини. Це так званий штрих-код.

Штрих-код використовується в торгівлі, бібліотечному чи поштовому ділі, при обробці документів, на виробництві, в охоронних схемах і т.п..[9]

На сьогодні день товарообіг відіграє велику роль у суспільству. З розвитком технологій його ніяк не можливо представити без організованості, яка допомагає структурувати великі об’єми товарів, невеликою кількістю апаратних засобів. Такі апаратні засоби повинні в собі містити: мінімальні вимоги до системи, для збереження і обробки інформації про товари та сканер штрих-кодів для перевірки товару на співпадіння.

В сучасному світі майже в кожної людини є мобільний телефон. Багато мобільних пристроїв забезпечені влаштованими фотокамерами. Фотокамера дає можливість сфотографувати штрих-код товару, для подальшої обробки.

Вважається штрих-код може позволити:

1. Порівнювати ціни товарів;
2. Отримувати повну характеристику продукта;
3. Дивитися обзори та відгуки інших покупців;
4. Провіряти продукти на інгредієнти-алергени.

Мета дослідження полягає в підвищення ефективності роботи розпізнавання штрих-коду EAN-13 на реальних зображеннях.

Завдання дослідження. Для досягнення поставленої мети в роботі сформульовані та вирішені такі завдання:

1. Викласти принципи розшифровки штрих-коду;
2. Дослідити особливості пошуку штрих-кодів та їх декодування;
3. Проаналізувати особливості реалізації алгоритму, які доступні в PyCharm;
4. Спроектувати та розробити відповідне програмне забезпечення;
5. Розрахувати витрати на розробку програми;

6. Зробити висновки .

Об’єкт дослідження: процес обробки, розпізнавання та перетворення зображення, декодування отриманого в процесі рядка пікселів.

Предмет дослідження: моделі, методи та інформаційна технологія забезпечення якості декодування штрих-коду EAN-13 при наявності спотворень.

Методи дослідження. Для виконання поставлених завдань були використані наступні методи та інструменти:

- Методи пошуку та декодування штрих-коду;
- PyCharm, Python, OpenGL.

Наукова новизна полягає в розробці нового підходу до розпізнавання та декодування штрих-кодів. Цей підхід обґрунтований вибором декодуючого рядку даних, цей підхід дозволяє прискорити розшифрування на спотворених зображеннях.

Практичне значення полягає в тому, що результати роботи, отримані в ході дослідження, можна використовувати для написання додатків будь-якого характеру, зчитування штрих-кодів стандарту EAN-13.

Особистий внесок автора:

1. Наукові результати роботи отримані автором самостійно;
2. Вибір методів досліджень і технологій реалізації;
3. Реалізація алгоритма засобами PyCharm мовою Python;
4. Оцінка отриманих результатів

Структура і обсяг роботи. Робота складається з вступу, трьох розділів і висновків. Містить 70 сторінок, 40 рисунків, 2 таблиці, списку використаних джерел з 44 найменуваннями на 4 сторінках, 2 додатка на 9 сторінках.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Дослідження предметної області

Штрих-коди повсюдно використовуються для ідентифікації продуктів, товарів або поставок. Навколо є пристрої для зчитування штрих-кодів у вигляді ручних зчитувачів, лазерних сканерів або світлодіодних сканерів. Зчитувачі на основі камери, як нові свого роду зчитувачі штрих-кодів, останнім часом привернули увагу. Інтерес до розпізнавання штрих-кодів на основі камер заснований на тому, що вже використовуються численні мобільні пристрої, які забезпечують можливість робити знімки належної якості. У поєднанні з підключення Bluetooth або WLAN, багато програм стають можливими, наприклад миттєва ідентифікація продуктів на основі штрих-коду та онлайн-пошуку інформації про продукт. Такі програми дозволяють відображати попередження для людей з алергією, результати продукту тести або порівняння цін у ситуаціях покупок.

Зусилля щодо розпізнавання 1D штрих-кодів за допомогою камер телефонів вже зроблено. Адельман та ін. [1] представили два прототипні програми: відображення літературної інформації про відскановані книги та відображення інформації про інгредієнти відсканованої їжі для алергіків. Вони не повідомили про результати розпізнавання, але показали підтвердження концепції для нових застосувань. Вони повідомляють про рівень розпізнавання 85,6% у неопублікованій базі даних зображень. З його опису ми вважаємо, що алгоритм набагато менш надійний, ніж алгоритм, запропонований у цій роботі.

Охбуч та ін. [4] представили розпізнавач реального часу для мобільних телефонів, який базується на асемблері і зберігається дуже просто. Чай і Хок [2] також представили алгоритм, але без уточнення результатів розпізнавання. Ранні алгоритми розпізнавання штрих-кодів (наприклад, Мінез та ін. [5], та Йозеф та Павлідіс [3]) і частини згаданих підходів досягають своєї мети, застосовуючи такі методи, як перетворення Хафа, локалізація штрих-коду на основі вейвлетів або морфологічні операції. Використання таких

методів призводить до обчислювально затратних реалізацій, які можуть бути не дуже вигідними для використання з мобільними пристроями.

1.2 Штрих-код товару та розшифровка

У 1948р. у Бернарда Сільвера та Джозефа Вудланда народилася ідея штрих-коду, коли вони почули розмову між президентом мережі продуктових компаній та деканом інституту технологій університету Дрекселя, які роздумували про те, як їм створити систему, яка автоматично зчитує інформацію про продукт. Вони спробували багато варіантів, одним з яких були нанесені ультрафіолетовим чорнилом позначки, поки не натрапили на стару добру стару азбуку Морзе. За словами Джозефа Вудланда, він розтягнув крапки і тире і зробив їх лінії: точка - вузька лінія, тире - широка [10].

Штриховий код – це закодоване число з 13 цифр яке має зображення у вигляді послідовностей білих та чорних смуг (рис.1.1). Ці коди придумані, для швидкої ідентифікації товарів. Найбільш поширені є європейська система кодування EAN та американський універсальний товарний код UPC [12].

Рис. 1.1. Приклад штрихового коду

Відповідно до прийнятого порядку, виробник товару наносить на нього штриховий код, що формується з використанням даних про країну місцезнаходження виробника та коду виробника [13]. Код виробника надається регіональним відділенням міжнародної організації EAN International. Такий порядок реєстрації дозволяє виключити можливість

появи двох різних товарів із однаковими кодами [13].

Розшифровка штрих-коду. Штрих-код містить в собі зашифровані дані про деякі характеристику товару.

Візьмемо, наприклад, цифровий код: 4820024700016 (рис.1.2). Перші дві або три цифри «482» означають країну виготовлення або продавця товару, наступні 4 або 5 в залежно від довжини коду країни «0024» - підприємство-виробник, ще п'ять «70001» ідентифікатор товару, його споживчі властивості, розміри, масу , колір. Остання цифра «6» контрольна, для перевірки справжності товару за штрих-кодом [10].

Рис. 1.2. Приклад розшифровки штрихового коду

Для коду товару:

1. Цифра: найменування товару;
2. Цифра: споживчі властивості;
3. Цифра: розміри, маса;
4. Цифра: інгредієнти;
5. Цифри: колір.

Перевірка справжності товару за штрих-кодом:

1. Скласти цифри, що стоять на парних місцях:

$$8+0+2+7+0+1=18$$

2. Отриману суму помножити на 3:

$$18\times3=54$$

3. Додати цифри, які знаходяться на непарних місцях, без контрольної цифри:

$$4+2+0+4+0+0=10$$

4. Скласти числа, зазначені у пунктах 2 та 3:

$$54+10=64$$

5. Відкинути десятки:

$$64\%10=4$$

6. З 10 відняти отримане в пункті 5:

$$10-4 = 6$$

Після підрахунку, звіримо результати обчислень з контрольною цифрою, якщо цифра не збігається, це значить, що товар незаконний.

1.3. Вибір мови програмування

Python – важка мова, але найбільш затребувана.

Мова програмування Python практично замінила інші курси вивчення мов на уроках інформатики. У програмах факультетів їй відведено багато викладацьких годин. Викладачі навчають студентів програмуванню на основі Python [24].

Функції мови ті самі: бекенд, веб-розробка та програми. Крім того, він використовується для машинного навчання та BigData [24]..

Було б неправильно називати Python виключно мовою веб-розробки. Використовується при написанні програм і додатків, тому вважається універсальним. Переваги мовлення: чіткий і точний синтаксис, навіть новачок не розгубиться; велика кількість алгоритмів і бібліотек; популярність на ринку праці; висока оплата; можливість працювати на будь-якій платформі. До особливостей варто віднести динамічну типізацію; наявність кількох версій - Python 2, Python 3, ймовірність сумісності коду; попит з боку великих компаній, таких як Apple, Microsoft, Google тощо [24].

Беручі інформацію з сайту statisticstime [25], виявлено, що мова Python знаходиться на першому місці (рис.1.3.1), та якщо подивитися на зведену статистику змін то зрозуміло, що мова виривається на першу сходинку за останній рік, тобто мова програмування набирає великої популярності серед розробників.

Популярність мов розраховується за багатьма методами, деякі з них є найбільш використовувані для порівняння, такі як:

- індекс TIOBE – він використовує інформацію пошукових систем, беруться до уваги певні фрази, що містять назву мови програмування і рахується кількість знайдених результатів (рис.1.3.2) [27];
- індекс PYPL – базується на Google Trends, відображаючи кількість запитів до пошукової системи виду «керівництво з <назва мови програмування>(рис.1.3.3)»[27].

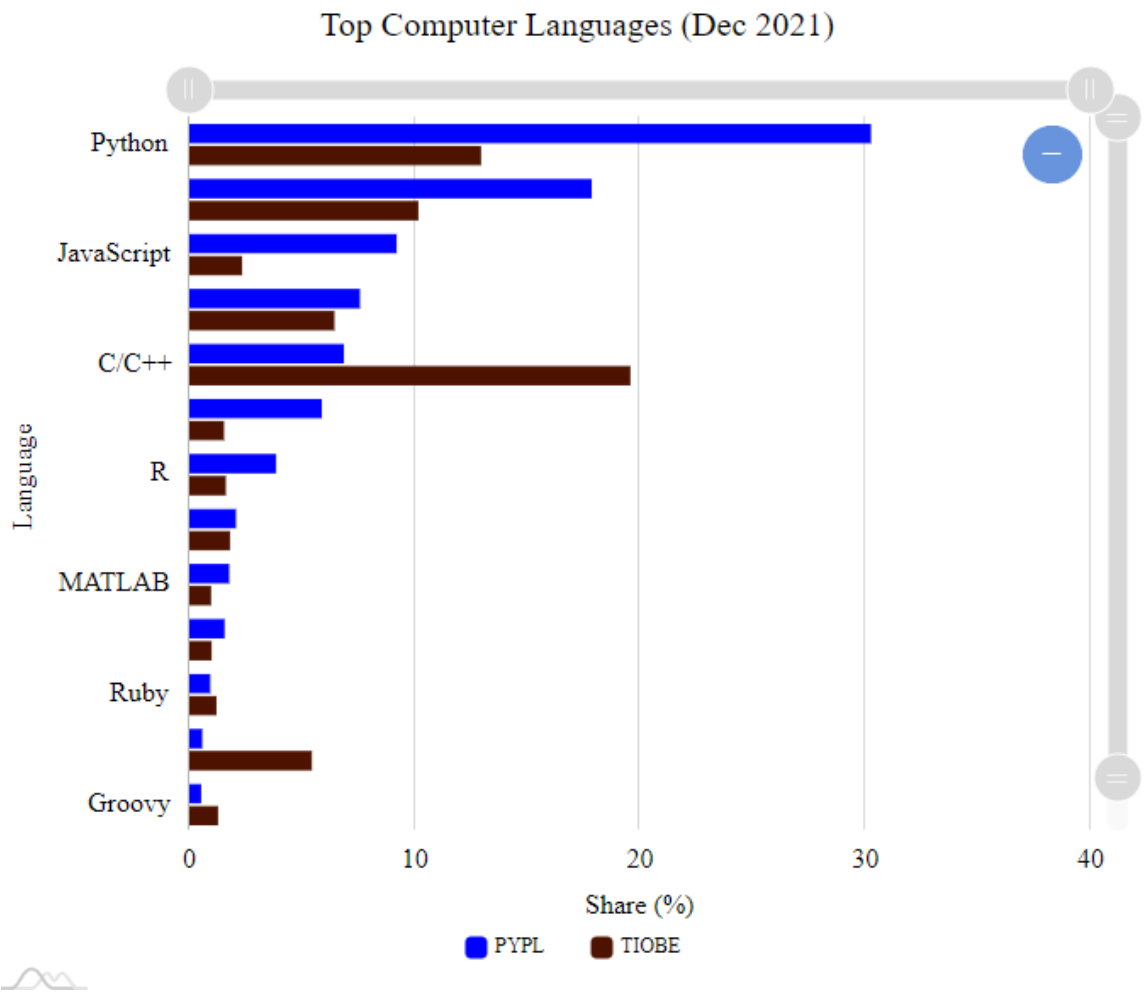

Рис.1.3.1. Найпопулярніші мови програмування

Dec 2021	Dec 2020	Change	Programming language	Ratings	Change
1	3	↑	Python	12.90%	+0.69%
2	1	↓	C	11.80%	-4.69%
3	2	↓	Java	10.12%	-2.41%
4	4		C++	7.73%	+0.82%
5	5		C#	6.40%	+2.21%
6	6		Visual Basic	5.40%	+1.48%
7	7		JavaScript	2.30%	-0.06%
8	12	↑↑	Assembly language	2.25%	+0.91%
9	10	↑	SQL	1.79%	+0.26%
10	13	↑	Swift	1.76%	+0.54%

Рис.1.3.2. Популярність мови програмування за індексом TIOBE

Dec 2021	Change	Programming language	Share	Trends
1		Python	30.21 %	-0.5 %
2		Java	17.82 %	+1.3 %
3		JavaScript	9.16 %	+0.6 %
4		C#	7.53 %	+1.0 %
5		C/C++	6.82 %	+0.6 %
6		PHP	5.84 %	-0.2 %
7		R	3.81 %	-0.0 %
8	↑	Swift	2.03 %	-0.2 %
9	↓	Objective-C	2.02 %	-1.6 %
10	↑	Matlab	1.73 %	-0.1 %

Рис.1.3.3. Популярність мови програмування за індексом PYPL

1.4. Переваги середи розробки

Якщо брати інформацію з сайту statisticstime [25], то інтегрована середа розробки PyCharm займає 5 місце (рис.1.4.1). В даній ситуації ця інформації є не дуже правильною, так як популярність враховується в цілому, велика кількість IDE використовується для розробки іншими мовами програмування.

Знайдено інформацію про рейтинг IDE в якій брались до уваги розробки лише мовою Python (рис.1.4.2) [29].

PyCharm поставляється з інтелектуальним редактором Python, інтелектуальною навігацією за кодом, швидким та безпечним рефакторингом. PyCharm інтегрований з такими функціями як налагодження, тестування, профілювання, розгортання, віддалена розробка та інструменти для бази даних. Разом з Python PyCharm також забезпечує підтримку фреймворків веб-розробки Python, функцій JavaScript, HTML, CSS, Angular JS та редагування у реальному часі. Він має потужну інтеграцію з блокнотом IPython, консоллю

Python та науковим стеком. PyCharm надає розробникам інтелектуальну платформу, яка допомагає їм, коли йдеться про автозавершення коду, виявлення помилок, швидке виправлення тощо. Забезпечує підтримку кількох фреймворків, збільшуючи багато факторів економії. Підтримує таку важливу функцію, як кроссплатформенна технологія, так що розробники можуть писати скрипти і на різних платформах. PyCharm також має хорошу функцію інтерфейсу, що, в свою чергу, підвищує продуктивність [29].

Worldwide, Jan 2022 compared to a year ago:

Rank	Change	IDE	Share	Trend
1		Visual Studio	30.48 %	+4.2 %
2		Eclipse	14.05 %	-2.0 %
3	↑	Visual Studio Code	12.34 %	+3.0 %
4	↓	Android Studio	8.45 %	-2.3 %
5		pyCharm	7.97 %	-0.0 %
6		IntelliJ	6.19 %	+0.3 %
7		NetBeans	5.34 %	-0.2 %
8		Xcode	3.52 %	-0.8 %
9	↑	Sublime Text	3.47 %	-0.2 %
10	↓	Atom	2.84 %	-0.9 %

Рис.1.4.1. Найпопулярніші IDE

IDE	User Rating	Size in MB	Developed in
PyCharm 	4.5/5	BIG	JAVA, PYTHON
Spyder 	May 4, 2018	BIG	PYTHON
PyDev 	4.6/5	MEDIUM	JAVA, PYTHON
Idle 	4.2/5	MEDIUM	PYTHON
Wing 	May 4, 2018	BIG	C, C++, PYTHON

Рис.1.4.2. Найпопулярніші IDE мовою Python

РОЗДІЛ 2
АЛГОРИТМ РОЗПІЗНАВАННЯ

2.1. Попередня обробка та бінаризація

До уваги представлено алгоритм розпізнавання одновимірних штрих-кодів, який працює для широко використовуваного стандарту EAN-13. Цей алгоритм використовує методи аналізу зображень та розпізнавання образів, які спираються на знання про структуру та зовнішній вигляд одновимірних штрих-кодів. Враховуючи обчислювальну потужність і якість зображення сучасних телефонів із камерою.

Мета нашого алгоритму - бути як швидким, так і надійним. В якості вхідних даних ми очікуємо зображення, що містить 1D штрих-код, який покриває центр зображення. Штрих-код не потрібно центрувати, він може бути перевернутим або мати звичайні спотворення та повороти перспективи (приблизно ±15 градусів), які виникають під час зйомки фотографій на телефони з камерою.

Інші підходи починаються з глобального згладжування, розташування області штрих-коду на основі вейвлетів [7] або навіть морфологічних операцій [2]. Вважається що, такі операції є це занадто трудомісткими, тому використано підхід на основі сканування. Припускаємо, що горизонтальна розгортка посередині зображення покриватиме штрих-код. Якщо це не так, або частини штрих-коду, які лежали на лінії сканування, забруднені, закупорені або піддані сильним відображенням, ми виявимо це на дуже ранній стадії та повторимо алгоритм для альтернативних ліній сканування зверху та знизу. Не шукаючи кордонів штрих-коду, ми швидко бінаризуємо всі пікселі на лінії сканування, починаючи з початкової точки в середині. Для бінаризації потрібен динамічний поріг, щоб він був стійкий до бруду, погано надрукованих штрих-кодів або зміни освітлення. Спочатку ми згладжуємо пікселі лінії сканування та обчислюємо значення яскравості $Y(x) \in [0..1]$ для кожної позиції x на лінії сканування ($Y(x) = 0{,}299R(x) + 0{,}587G(x) + 0{,}114B(x)$).

Потім виконуємо пошук локальних мінімуму та максимуму вздовж лінії

сканування, щоб сусідні мінімуми та максимуми мали різницю яскравості $\Delta Y \geq 0{,}01$. Останнім кроком перед обчисленням порогового значення є етап обрізання, де ми видаляємо незвичайно темні максимуми, а також незвичайно світлі мінімуми. Для кожної позиції x від середини лінії сканування до межі зображення поріг t(x) для бінаризації обчислюється шляхом оцінки функції залежно від останніх семи мінімумів/максимумів. Ми не враховуємо зовнішні мінімуми/максимуми, оскільки не хочемо, щоб регіони за межами штрих-коду впливали на пороги в області штрих-коду. На малюнку показані профіль, мінімуми та максимуми, обрізані екстремуми та результуючий поріг (рис.2.1). Оцінювана функція усереднює значення яскравості другого найнижчого максимуму та другого найвищого мінімуму для досягнення хорошої стійкості до локальних помилок, таких як бруд або сильний шум.

Рис.2.1. Динамічний поріг для бінаризації

2.2. Виявлення кордонів штрих-коду

В наступному кроці використано деякі знання про штрих-код UPC-A/EAN-13/ISBN-13. Ці штрих-коди складаються з 13 цифр. Остання цифра є контрольною сумою, яка обчислюється з перших 12 цифр.

Штрих-код починається лівою захисною смугою A (чорно-біло-чорна) і закінчується правою захисною смугою E (чорно-біло-чорна). Між захисними смугами є два блоки B і D по 6 кодованих цифр у кожному, розділені центральною смугою C (білий-чорний-білий-чорний-білий).

Модуль - це найменша одиниця. Смуги та пробіли можуть охоплювати від одного до чотирьох модулів одного кольору. Кожна цифра кодується за допомогою семи модулів (дві смуги і два пробіли із загальною шириною 7 модулів). Ширина повного штрих-коду EAN-13 становить 59 чорно-білих областей ($3 + 6 * 4 + 5 + 6 * 4 + 3$), які складаються з 95 модулів ($3 + 6 * 7 + 5 + 6 * 7 + 3$). Для кодування цифри можна використовувати два алфавіти, парний або непарний алфавіт. Хоча останні 12 цифр кодуються безпосередньо за допомогою цих двох алфавітів, перша з 13 цифр визначається алфавітами, які були використані для кодування перших шести цифр. Таким чином, перша цифра називається метачислом або індукованою цифрою.

Виявлення кордонів штрих-коду також починається з початкової точки в середині лінії профілю. Наше початкове положення має бути бар-піксель, тому або початкова точка чорна ($Y(x) < t(x)$), або ми починаємо з найближчої точки ліворуч або праворуч, яка є чорною. З цього стартового рядка ми послідовно додаємо пробіли і рядок ліворуч і праворуч. Додаючи спочатку менші пари пробілів, ми не допускаємо додавання областей без штрих-коду. Запам’ятовуючи розміри доданих блоків, ми можемо визначити кандидатів на захисні смуги. Ми знайшли захисні решітки, якщо кількість штрихів і пробілів між ними дорівнює 59.

2.3. Класифікація цифр

Смуги та пробіли, знайдені на попередньому кроці, мають бути класифіковані як цифри. Як уже згадувалося, є два алфавіти по 10 цифр кожен, що призводить до 20 класів. Кожна цифра s кодується за допомогою двох чорно-білих областей загальною шириною 7 модулів [11].

Під час підготовки до класифікації створено специфічний прототип для кожного класу. Для цього досліджено дві захисні планки зліва і справа, а також центральну планку. Оскільки відомо, що їх чорно-білі області мають ширину в один модуль кожна, можливо обчислити середню ширину окремих чорно-білих модулів. На основі ширини одиничних модулів обчислено ширину подвійних, потрійних і чотириразових модулів. Двійкова область подвійного модуля є не вдвічі більше, ніж площа окремого модуля. Ширина окремого чорного модуля не є загальною шириною штрих-коду, поділеною на 95. Через більш яскраве або темне освітлення ширина окремих білих модулів ww1 і чорного wb1 може відрізнятися на:

$$\Delta wb1 = ww1 - wb1. \qquad (2.3.1)$$

Це необхідно враховувати для розрахунку ширини подвійних, потрійних і чотиримісних модулів і вимагає окремого визначення ширини чорних і білих одинарних модулів.

На основі ширини смуг і пробілів в захисних і центральних смугах обчислено опорний шаблон rk для кожного класу ck. Цифри s зі штрих-коду потім представляються класифікатору на основі відстані, який призначає нормовані значення подібності p(ck, s) для всіх класів ck.

Оскільки кожна цифра кодується чотирма чорними та білими областями, шаблон r представлено у вигляді 4-х кортежів $r \in IR^4$. Подібність p(ck, s) заснована на квадраті

відстані d(rk, rs) між відповідним шаблоном

2.4. Пошук найбільш схожого коду

Наступним етапом є пошук найбільш схожого коду, для кожної цифри s з кожним класом ck. Комбінуючи результати дванадцяти кодованих цифр s (1), s(2), . . . , s(12) послідовно генерується гіпотеза коду (m, c(1), c(2), . . , c(12)), m - це індукційне мета-число, закодований вибором алфавітів, що використовуються для кодування цифр s7, . . . , s13. Якщо вважати значення подібності цифр незалежними одна від одної та подібними до ймовірності, вважається ймовірність гіпотези такою:

$$p(c\,(1),\ldots,c(12)\mid s\,(1),\ldots,s(12)) \;=\prod_{i=12}^{12} p(c\,(i)\mid s\,(i)\,)\;1/12 \qquad (2.3.2)$$

Починаючи з гіпотези коду, яка складається з найбільш подібних класів для кожної цифри, ми послідовно досліджуємо гіпотези зі спадною ймовірністю. Для зображень хорошої якості перша гіпотеза є правильною. У разі сильних викривлень перші гіпотези можуть бути помилковими. Цифра s13 – контрольна сума, яка дозволяє виявити та відхилити неправильні гіпотези.

2.5. Опис структури програми та алгоритмів її функціонування

На блок-схемі (рис.2.5.1) зображено алгоритм процесу декодування штрих-кодів

Рис.2.5.1. Блок-схема розроблюваного алгоритму програми

Майбутня програма являє собою набір функцій об’єднаних в одному файлі, так звана бібліотека.

Всі функції в файлі ean13.py знаходяться в пакеті BarCode (рис.2.5.1), в даному файлі знаходяться такі функції як:

– decode(img) Функції декодування якій передається зображення;

– decode_line(thresh[i]) функція яка відділяє центр зображення;

– read_barcode (line) функція відповідає за перетворення рядка пікселів зображення на рядок бінарних даних;

– classify_barcode(barcodes) функція відповідає за розділ рядка бінарних даних, на відповідні частини штрих-коду: ліву та праву частину штрих-коду, три захисні смужки, ліву, праву та центральну;

– convert_patterns_to_length (patterns) функція перетворює дані із рядкових послідовностей в розмірні величини;

– read_patterns(patterns, is_left=True) функція яка відповідає за декодування;

– decode_right(at1, at2, m1, m2, m3, m4) функція декодування правої частину штрих-коду

– decode_left(at1, at2, m1, m2, m3, m4) функція декодування лівої частину штрих-коду;

– get_ean13(left_codes, right_codes) функція з’єднує результати декодування лівої та правої частини штрих-коду до купи;

– verify(ean13) функція перевірки контрольної сумми;

– decoding_found_image(image) функція відповідає за декодування штрих-коду на зображення який потрібно спочатку з відшукати;

– decoding_real_time(image) функція відповідає за декодування штрих-коду який розміщений по всьому зображенні, ця функція призначена для декодування штрих-коду в відео потоці.

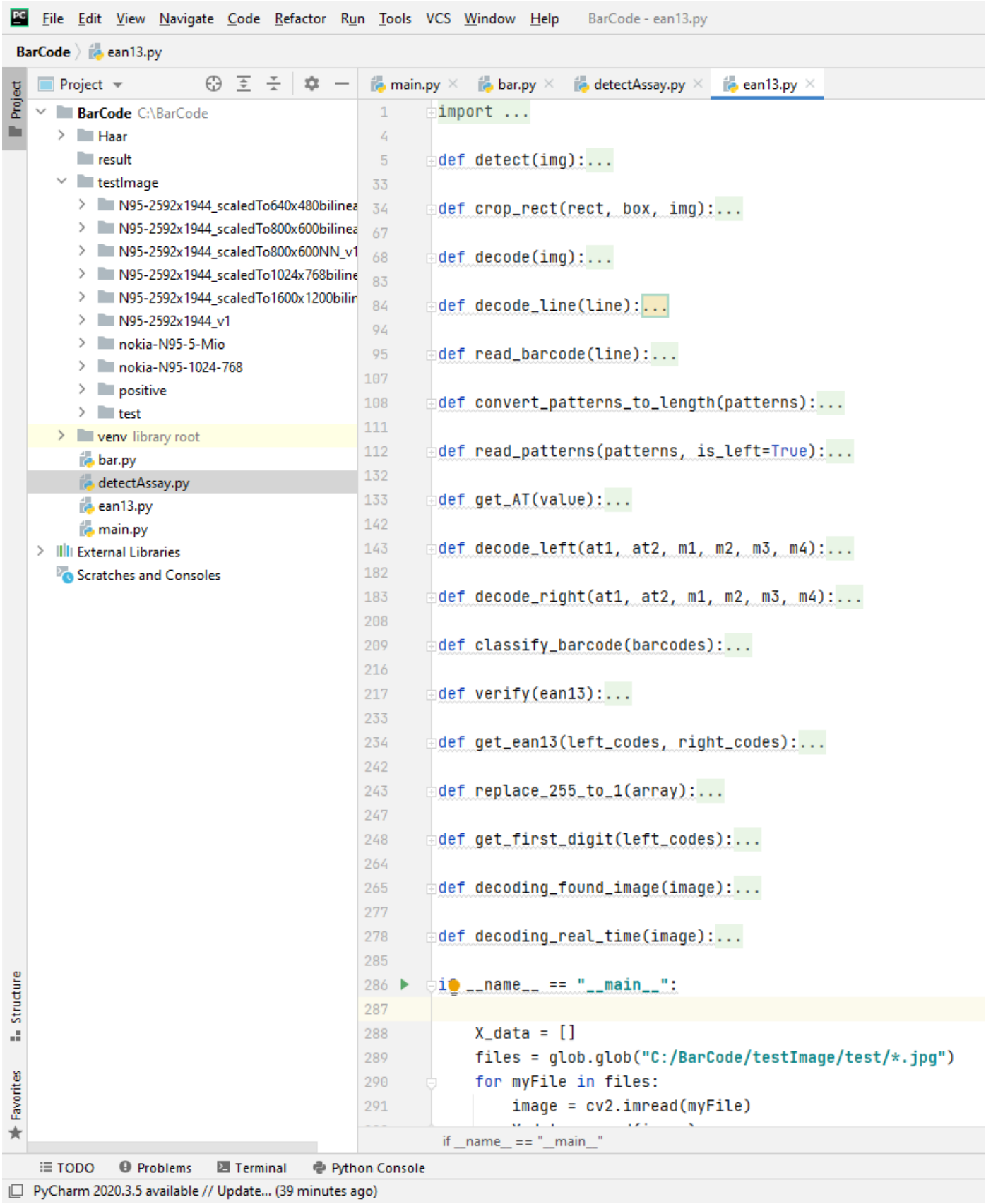

Рис.2.5.1. Всі функції файлу ean13.py

РОЗДІЛ 3

РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1. Пошук штрих-коду на зображенні

1. Замінено розмір зображення для нормалізації (рис.3.2.1.1).

Рис.3.2.1.1. Зображення заміненого розміру

2. Створено граничне зображення (рис.3.2.1.2).

Рис.3.1.2. Граничне зображення

3. Значення білих пікселів порогових зображень дорівнює 255, а чорних пікселів - 0. Потрібно інвертувати його і замінити 255 на 1, щоб відповідати шаблону 0 і 1 (рис.3.2.1.3).

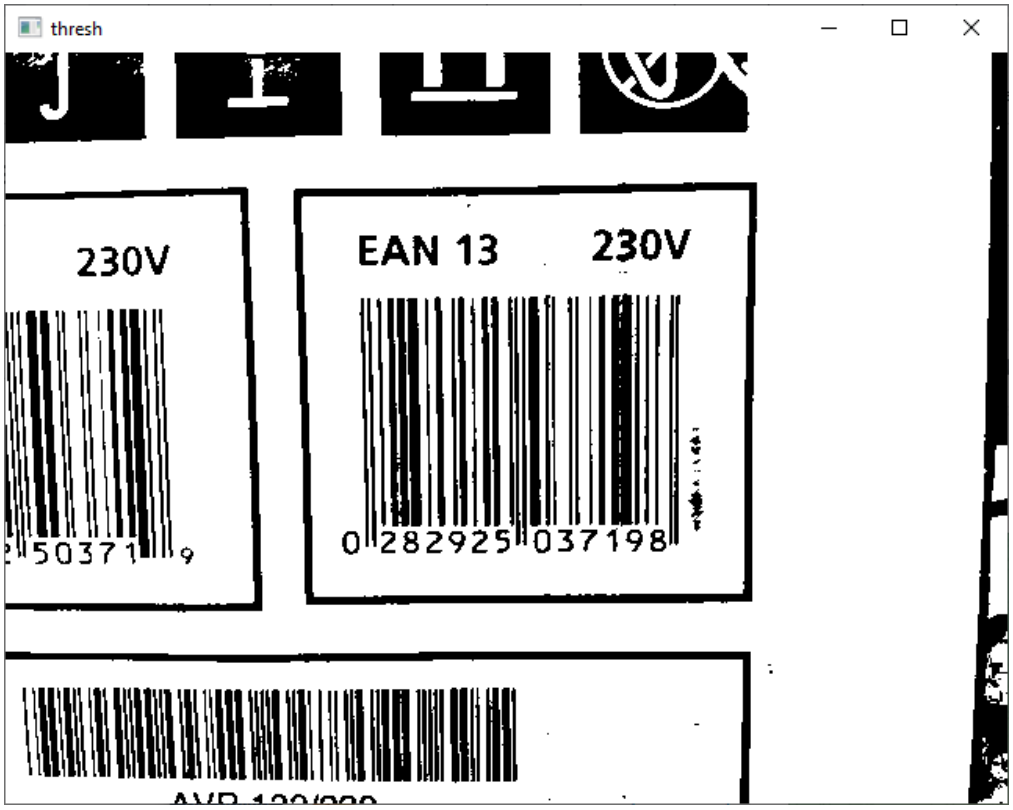

Рис.3.1.2. Чорно-біле зображення

4. Знаходження контуру та обрізання поверхні областей-кандидатів. Після знаходження контурів йде невелика перевірка. Вважається що, штрих-код містить в собі 95 областей, отже він не може буди розміром меншим 95 пікселів. Отримані кандидати будуть надіслані на розшифровку.

3.2. Розшифрування даних

Зверніть увагу на рисунок (рис.3.2.2.1) на якому штрих-код поділено на дві частини, по середині та по краям захисні смужки, які не несуть в собі ніяких даних, але допомагають в розшифровці. Штрих-код EAN-13 має дві групи кодування ліву та праву, права частина кодується просто, в залежності таблиці декодування захисних маркерів (див. табл.2), підбирається відповідний код. Ліва частина несе в собі ще одну, 13 цифру. При декодуванні штриха лівої частини береться перевірка по двом стовпцям з таблиці декодування захисних маркерів (див. табл.2), крім запам’ятовування цифри яка декодувалась,

ще записується до пам’яті до якої групи відноситься декодований штрих L чи G.

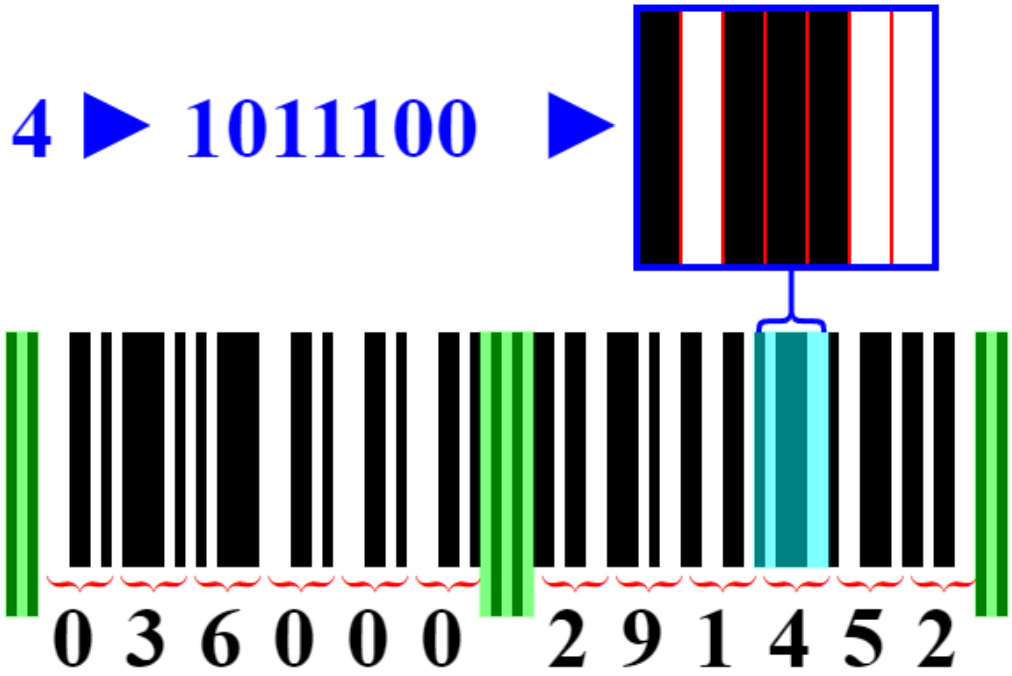

Рис.3.2.1. Приклад розшифровки штрих-коду

Перша цифра вираховується по структурі лівої групи з 6 цифр (див. табл.1). Приклад кодування штрих-коду EAN-13 показано на рисунку (рис.3.2.2). Кодування цифр коду L,G, та R зображено на рисунках (рис.3.2.3 – рис.3.2.5).

Таблиця 1

Структура коду EAN-13

Перша цифра	Ліва група з 6 цифр	Права група з 6 цифр
0	LLLLLL	RRRRRR
1	LLGLGG	RRRRRR
2	LLGGLG	RRRRRR
3	LLGGGL	RRRRRR
4	LGLLGG	RRRRRR
5	LGGLLG	RRRRRR
6	LGGGLL	RRRRRR
7	LGLGLG	RRRRRR
8	LGLGGL	RRRRRR
9	LGGLGL	RRRRRR

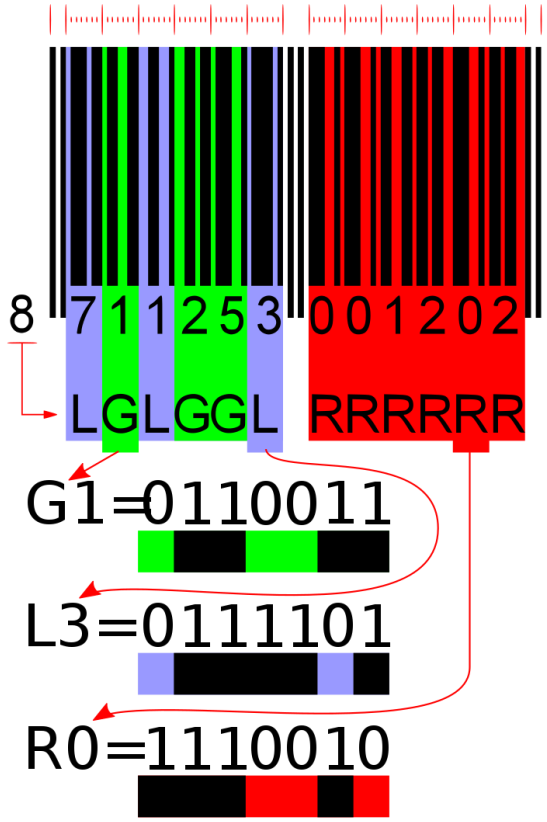

Рис.3.2.2. Приклад кодування EAN-13 штрих-коду

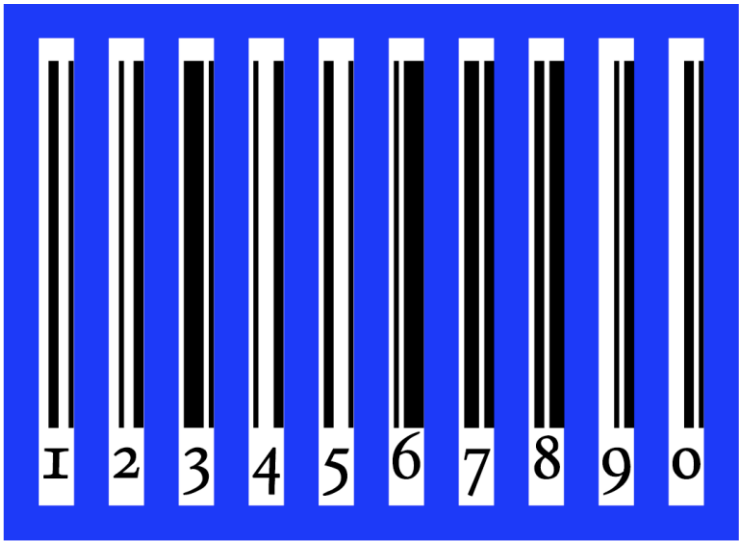

Рис.3.2.3. Кодування L-цифри

Рис.3.2.4. Кодування G-цифри

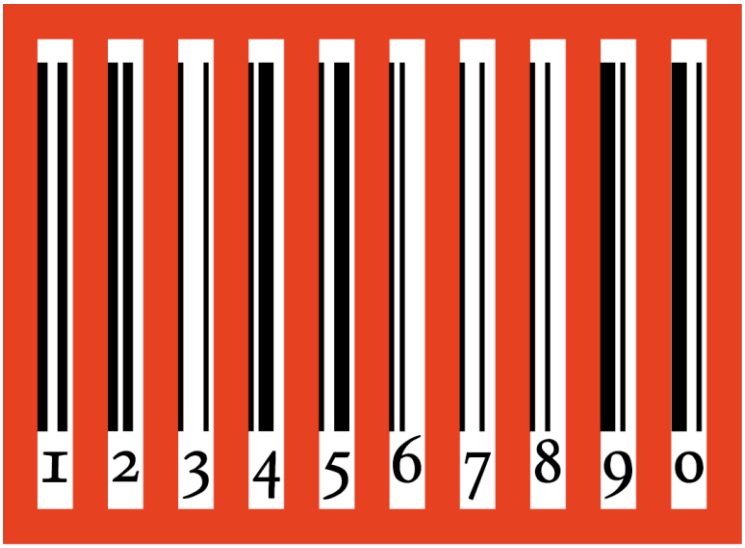

Рис.3.2.5. Кодування R-цифри

Таблиця 2

Таблиця декодування захисних маркерів

Цифра	L-код	G-код	R-код
0	0001101	0100111	1110010
1	0011001	0110011	1100110
2	0010011	0011011	1101100
3	0111101	0100001	1000010
4	0100011	0011101	1011100
5	0110001	0111001	1001110
6	0101111	0000101	1010000
7	0111011	0010001	1000100
8	0110111	0001001	1001000
9	0001011	0010111	1110100

3.3. Програмна реалізація

Декодування починається з функції decode(img) якій передається зображення (рис.3.3.1).

```
def decode(img):
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    ret, thresh = cv2.threshold(gray, 200, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)
    thresh = cv2.bitwise_not(thresh)
    ean13 = None
    is_valid = None
    # scan lines
    for i in range(img.shape[0] - 1, 0, -1):
        try:
            ean13, is_valid = decode_line(thresh[i])
            print("i "+str(i))
            print(str(img.shape[0] - 1))
        except Exception as e:
            pass
        if is_valid:
            break
    return ean13, is_valid, thresh
```

Рис.3.3.1. Код головної функція декодування штрих-коду

Відділяємо центр зображення та передаємо дані функції decode_line(thresh[i]) (рис.3.3.2).

```
def decode_line(line):
    barcodes = read_barcode(line)
    left_guard, left_patterns, center_guard, right_patterns, right_guard = classify_barcode(barcodes)
    convert_patterns_to_length(left_patterns)
    convert_patterns_to_length(right_patterns)
    left_codes = read_patterns(left_patterns, is_left=True)
    right_codes = read_patterns(right_patterns, is_left=False)
    ean13 = get_ean13(left_codes, right_codes)
    is_valid = verify(ean13)
    return ean13, is_valid
```

Рис.3.3.2. Код функції decode_line(line)

Функція read_barcode (line) (рис.3.3.3) відповідає за перетворення рядка пікселів зображення на рядок бінарних даних.

```
def read_barcode(line):
    replace_255_to_1(line)
    barcodes = []
    current_length = 1
    for i in range(len(line) - 1):
        if line[i] == line[i + 1]:
            current_length = current_length + 1
        else:
            barcodes.append(current_length * str(line[i]))
            current_length = 1
    barcodes.pop(0)
    return barcodes
```

Рис.3.3.3. Код функції read_barcode (line)

Функція classify_barcode(barcodes) (рис.3.3.4) відповідає за розділ рядка бінарних даних, на відповідні частини штрих-коду: ліву та праву частину штрих-коду, три захисні смужки, ліву, праву та центральну.

```
def classify_barcode(barcodes):
    left_guard = barcodes[0:3]
    left_patterns = barcodes[3:27]
    center_guard = barcodes[27:32]
    right_patterns = barcodes[32:56]
    right_guard = barcodes[56:59]
    return left_guard, left_patterns, center_guard, right_patterns, right_guard
```

Рис.3.3.4. Код функції classify_barcode(barcodes)

За допомогою функції convert_patterns_to_length (patterns) (рис.3.3.4) перетворюємо дані із рядковий послідовностей в розмірні величини.

```
def convert_patterns_to_length(patterns):
    for i in range(len(patterns)):
        patterns[i] = len(patterns[i])
```

Рис.3.3.4. Код функції convert_patterns_to_length (patterns)

Наступна функція read_patterns(patterns, is_left=True) (рис.3.3.5) приймає розмірні величини та після деяких маніпуляцій передає до функції декодування decode_right(at1, at2, m1, m2, m3, m4) (рис.3.3.6) праву частину штрих-кода та decode_left(at1, at2, m1, m2, m3, m4) (рис.3.3.7) ліву.

```
read_patterns(patterns, is_left=True):
codes = []
for i in range(6):
    start_index = i * 4
    sliced = patterns[start_index:start_index + 4]
    m1 = sliced[0]
    m2 = sliced[1]
    m3 = sliced[2]
    m4 = sliced[3]
    total = m1 + m2 + m3 + m4
    tmp1 = (m1 + m2) * 1.0
    tmp2 = (m2 + m3) * 1.0
    at1 = get_AT(tmp1 / total)
    at2 = get_AT(tmp2 / total)

    if is_left:
        decoded = decode_left(at1, at2, m1, m2, m3, m4)
    else:
        decoded = decode_right(at1, at2, m1, m2, m3, m4)
    codes.append(decoded)
return codes
```

Рис.3.3.5. Код функції read_patterns(patterns, is_left=True)

```
def decode_right(at1, at2, m1, m2, m3, m4):
    patterns = {}
    patterns["2,2"] = {"code": "6"}
    patterns["2,4"] = {"code": "4"}
    patterns["3,3"] = {"code": "8", "alter_code": "2"}
    patterns["3,5"] = {"code": "5"}
    patterns["4,2"] = {"code": "9"}
    patterns["4,4"] = {"code": "7", "alter_code": "1"}
    patterns["5,3"] = {"code": "0"}
    patterns["5,5"] = {"code": "3"}
    pattern_dict = patterns[str(at1) + "," + str(at2)]
    code = 0
    use_alternative = False
    if int(at1) == 3 and int(at2) == 3:
        if m3 + 1 >= m4:
            use_alternative = True
    if int(at1) == 4 and int(at2) == 4:
        if m1 + 1 >= m2:
            use_alternative = True
    if use_alternative:
        code = pattern_dict["alter_code"]
    else:
        code = pattern_dict["code"]
    final = {"code": code}
    return final
```

Рис.3.3.6. Код функції decode_right(at1, at2, m1, m2, m3, m4)

```python
def decode_left(at1, at2, m1, m2, m3, m4):
    patterns = {}
    patterns["2,2"] = {"code": "6", "parity": "O"}
    patterns["2,3"] = {"code": "0", "parity": "E"}
    patterns["2,4"] = {"code": "4", "parity": "O"}
    patterns["2,5"] = {"code": "3", "parity": "E"}
    patterns["3,2"] = {"code": "9", "parity": "E"}
    patterns["3,3"] = {"code": "8", "parity": "O", "alter_code": "2"}
    patterns["3,4"] = {"code": "7", "parity": "E", "alter_code": "1"}
    patterns["3,5"] = {"code": "5", "parity": "O"}
    patterns["4,2"] = {"code": "9", "parity": "O"}
    patterns["4,3"] = {"code": "8", "parity": "E", "alter_code": "2"}
    patterns["4,4"] = {"code": "7", "parity": "O", "alter_code": "1"}
    patterns["4,5"] = {"code": "5", "parity": "E"}
    patterns["5,2"] = {"code": "6", "parity": "E"}
    patterns["5,3"] = {"code": "0", "parity": "O"}
    patterns["5,4"] = {"code": "4", "parity": "E"}
    patterns["5,5"] = {"code": "3", "parity": "O"}
    pattern_dict = patterns[str(at1) + "," + str(at2)]
    code = 0
    use_alternative = False
    if int(at1) == 3 and int(at2) == 3:
        if m3 + 1 >= m4:
            use_alternative = True
    if int(at1) == 3 and int(at2) == 4:
        if m2 + 1 >= m3:
            use_alternative = True
    if int(at1) == 4 and int(at2) == 3:
        if m2 + 1 >= m1:
            use_alternative = True
    if int(at1) == 4 and int(at2) == 4:
        if m1 + 1 >= m2:
            use_alternative = True
    if use_alternative:
        code = pattern_dict["alter_code"]
    else:
        code = pattern_dict["code"]
    final = {"code": code, "parity": pattern_dict["parity"]}
    return final
```

Рис.3.3.6. Код функції decode_leftt(at1, at2, m1, m2, m3, m4)

Функція get_ean13(left_codes, right_codes) (рис.3.3.7) приймає два параметри результати декодування лівої та правої частини штрих-коду, ця функція з'єднує їх до купи.

```
def get_ean13(left_codes, right_codes):
    ean13 = ""
    ean13 = ean13 + str(get_first_digit(left_codes))
    for code in left_codes:
        ean13 = ean13 + str(code["code"])
    for code in right_codes:
        ean13 = ean13 + str(code["code"])
    return ean13
```

Рис.3.3.6. Код функції decode_leftt(at1, at2, m1, m2, m3, m4)

Останній етап декодування - це перевірка контрольної сумми, цю задачу виконує функція verify(ean13) (рис.3.3.7).

```
def verify(ean13):
    weight = [1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1, 3]
    weighted_sum = 0
    for i in range(12):
        weighted_sum = weighted_sum + weight[i] * int(ean13[i])
    weighted_sum = str(weighted_sum)
    checksum = 0
    units_digit = int(weighted_sum[-1])
    if units_digit != 0:
        checksum = 10 - units_digit
    else:
        checksum = 0
    if checksum == int(ean13[-1]):
        return True
    else:
        return False
```

Рис.3.3.7. Код функції verify(ean13)

3.4. Експерименти з базою даних штрих-кодів

Для дослідів було використано базу даних з понад 1000 зображень штрих-кодів, зроблених за допомогою камери телефону Nokia N95, яка є вільно доступною для інших дослідників [6]. Використовуючи цей фіксований набір зображень, який включає сильно спотворені, хиткі та розфокусовані зображення, розроблено алгоритм за допомогою мови програмування Python з використанням інтегрованого середовища розробки PyCharm. Щоб дослідити вплив роздільної здатності на продуктивність розпізнавання, зображення були зроблені з найвищою підтримуваною роздільною здатністю (2592 × 1944) і зменшені до типових роздільних можливостей телефонів з камерою (1024 × 768 і 640 × 480). Ефективність розпізнавання реалізації становить 90,5% при 640×480, 93,7% при 1024×768 і 99,2% при 2592×1944 пікселях. Час виконання змінюється в залежності від роздільної здатності якості зображення та часу формування гіпотез. Для розпізнавання одного штрих-коду потрібно від 65 мс (640 × 480) до 100 мс (2592 × 1944) на чотирьох ядерному комп'ютері 2,2 ГГц.

Для порівняння алгоритму з уже існуючими було взято алгоритм з бібліотеки PyZbar. Обидва алгоритми показали гарні результати більше 90%, але в базі даних є більшість зображень з чітким штрих-кодом. Тому було вибрано 100 зображень з несфокусованим зображенням, нахилом камери, вигином поверхні, нерівномірним освітленням та шумами.

Алгоритм запропонований в даній роботі справився добре 89% від 100 зображень, 8 зображення не розпізнало взагалі а 2 розпізнано хибно. В порівнянні з іншим алгоритмом, який розпізнав 78%. Розроблений алгоритм є надійнішим, він розпізнав штрих-кодів на 11% більше. По швидкості розпізнавання алгоритм PyZbar випереджає на 20 мс в середньому.

Приклади нерозпізнаних штрих-кодів (рис.3.4.1, рис.3.4.2), на даних зображеннях засвітлена частина штрих-коду, низька частота дискретизації та велика кількість шумів які не дали можливості правильному декодуванні. На третьому зображенні представлена нерівномірна освітлюваність та велика кількість шумів.

Рис.3.4.1. Приклади нерозпізнаний штрих-кодів

Рис.3.4.2. Приклади нерозпізнаний штрих-кодів

Розроблений алгоритм може розпізнавати штрих-коди які мають нахил камери майже 30 градусів, можливість розпізнавання такого зображення представлено на (рис.3.4.2).

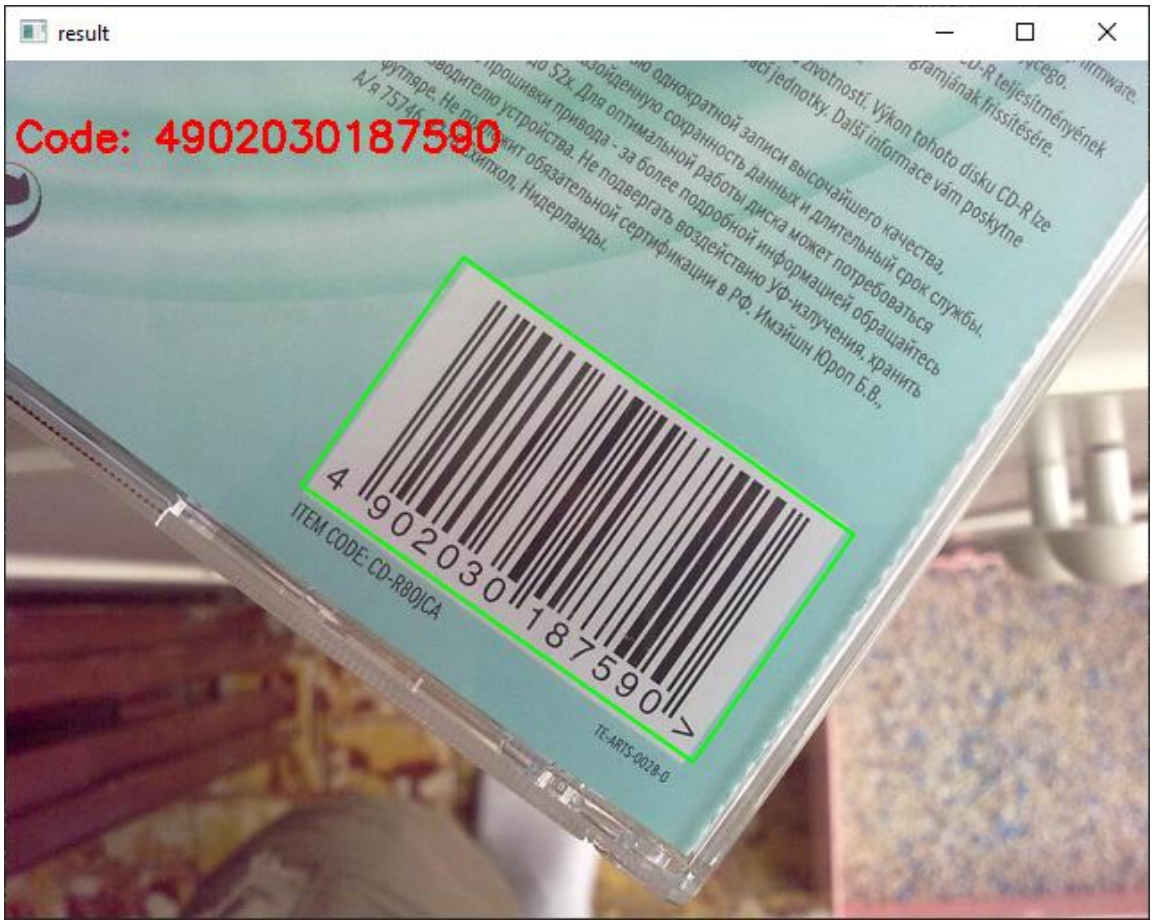

Рис.3.4.3. Приклади розпізнаного штрих-коду з нахилом камери

Алгоритм повинен бути надійний при розпізнаванні з таким спотворенням як розпізнавання з вигином поверхні, тому що багато штрих-кодів надруковані на скляний пляшках та металевий банках (рис.3.4.4, рис.3.4.5). Не всі телефони оснащені якісними камерами, тому можливі зображення з нерівномірним освітлюванням, дефокусуванням, низькою частотою дискретизації та шумами (рис.3.4.5 – рис.3.4.12).

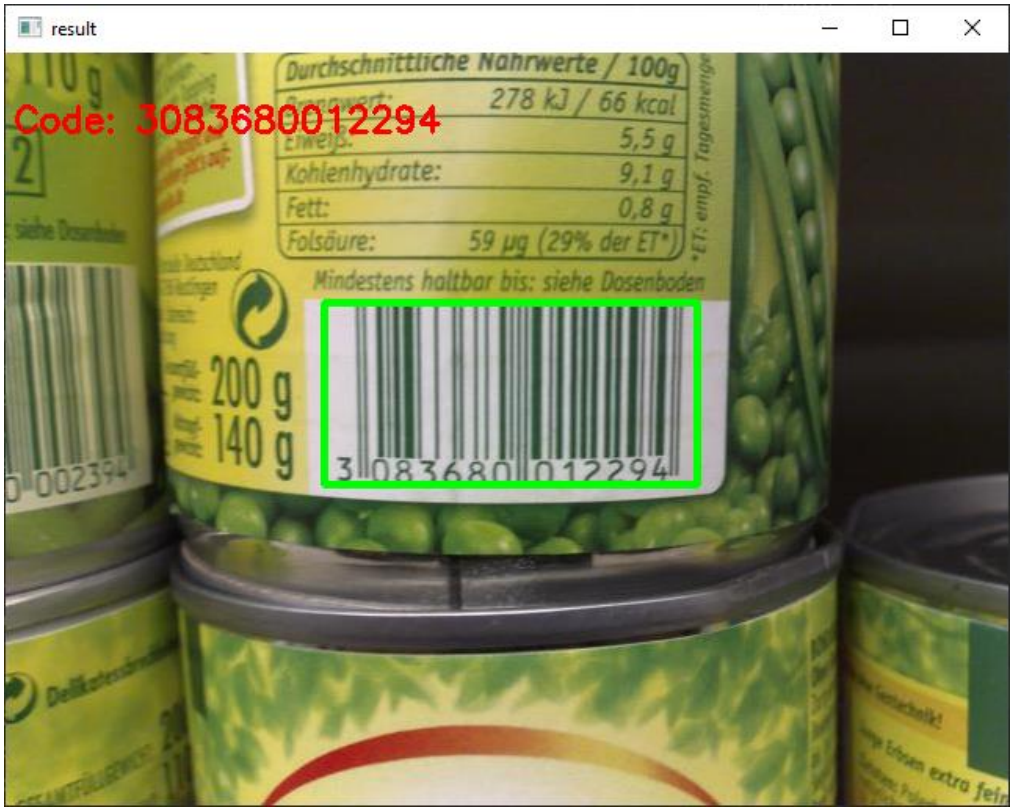

Рис.3.4.4. Приклади розпізнаного штрих-коду з вигином поверхні

Рис.3.4.5. Приклади розпізнаного штрих-коду з вигином поверхні

Рис.3.4.6. Приклад розпізнаного штрих-коду

Рис.3.4.7. Приклад розпізнаного штрих-коду

Рис.3.4.8. Приклад розпізнаного штрих-коду

Рис.3.4.9. Приклад розпізнаного штрих-коду

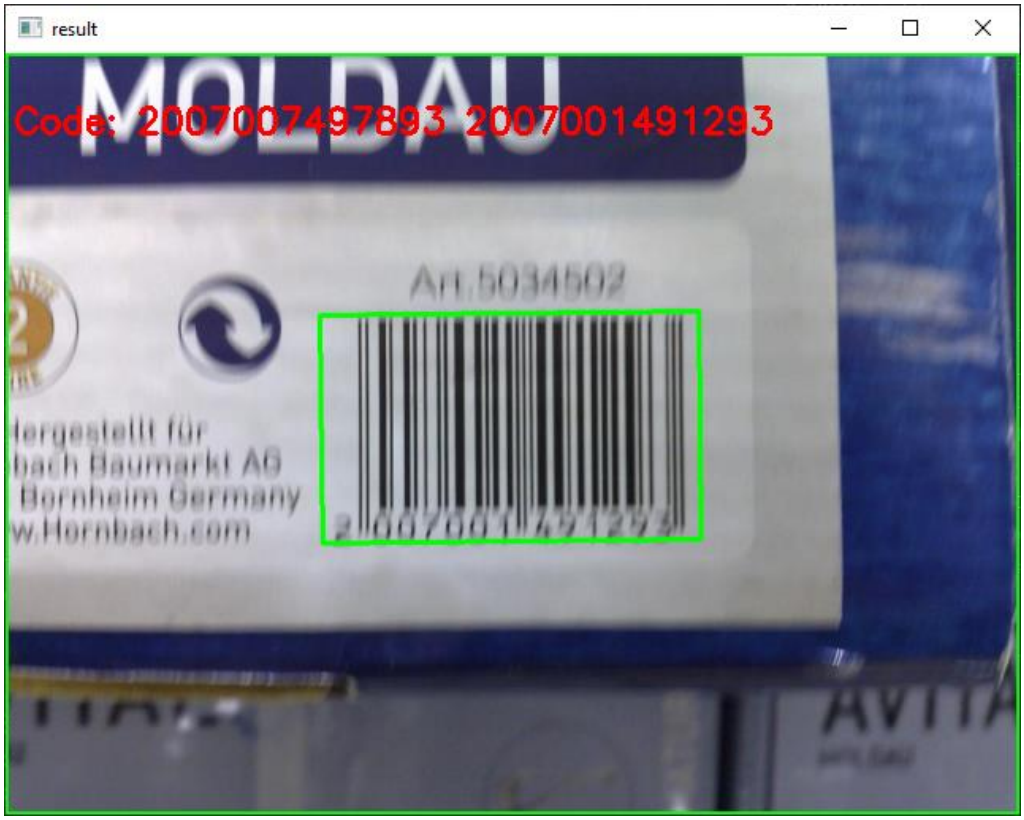

Рис.3.4.10. Приклад розпізнаного штрих-коду

Рис.3.4.11. Приклад розпізнаного штрих-коду

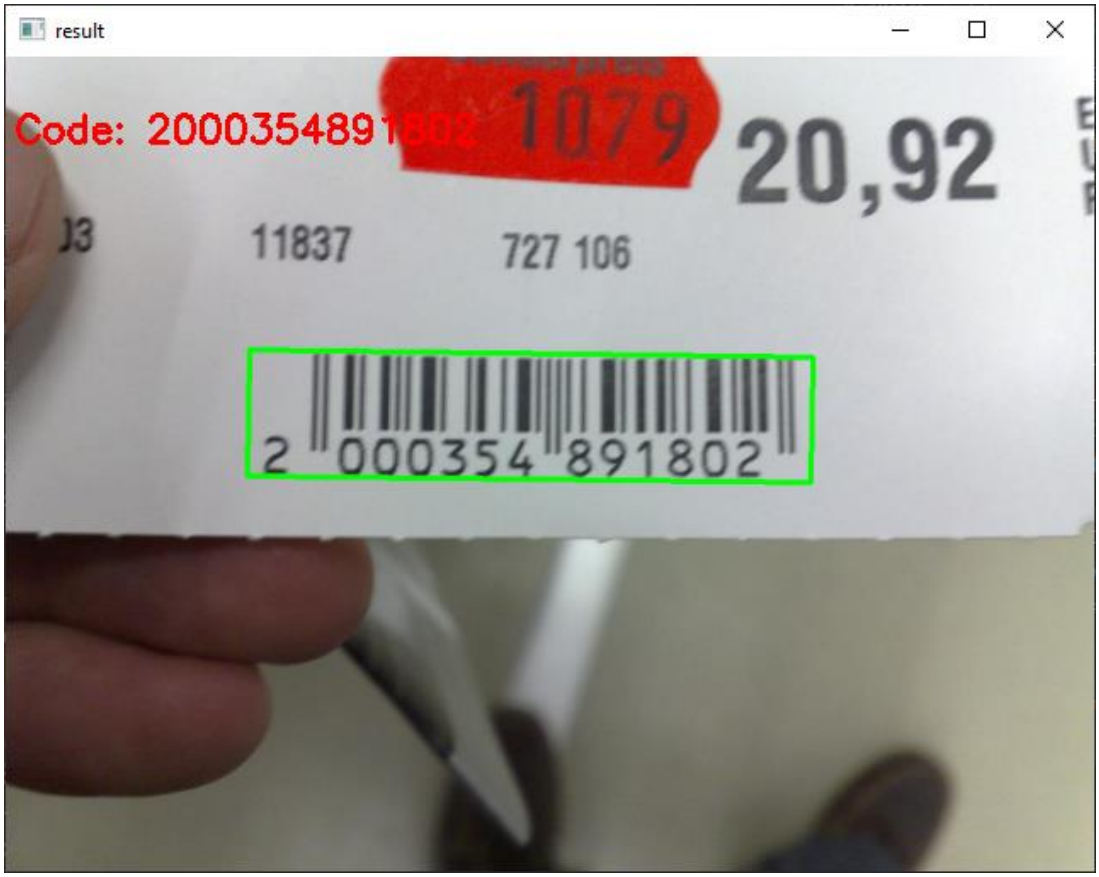

Рис.3.4.12. Приклад розпізнаного штрих-коду

РОЗДІЛ 4
ЕКОНОМІКА

4.1. Визначення трудомісткості розробки програмного забезпечення

В ході розробки програмного забезпечення важливими етапами є визначення трудомісткості розробки ПЗ, та розрахунок витрат на створення програмного продукту і тривалості його розробки.

Вхідні дані:

1. Передбачуване число операторів програми – 400;
2. Коефіцієнт складності програми – 1,8;
3. Коефіцієнт корекції програми в ході її розробки – 0,1;
4. Годинна заробітна плата програміста– 80 грн/год;
5. Коефіцієнт збільшення витрат праці в наслідок недостатнього опису задачі – 1,2;
6. Коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності – 1,0;
7. Вартість машино-години ЕОМ – 18 грн/год.

Нормування роботи в процесі створення ПЗ істотно ускладнюється в силу творчого характеру роботи програміста. Тому трудомісткість розробки ПЗ може бути розрахована на основі системи моделей з різною точністю оцінки.

Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = t_o + t_u + t_a + t_n + t_{отл} + t_д, \text{ людино-годин,} \qquad (4.1)$$

де t_o - витрати праці на підготовку й опис поставленої задачі (приймається 50);

t_u- витрати праці на дослідження алгоритму рішення задачі;

t_a - витрати праці на розробку блок-схеми алгоритму;

t_n - витрати праці на програмування по готовій блок-схемі;

$t_{отл}$ - витрати праці на налагодження програми на ЕОМ;

$t_д$ - витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у ПЗ, яке розробляється.

Умовне число операторів (підпрограм):

$$Q = q \cdot C \cdot (1+ p) \text{ , людино-годин,} \quad (4.2)$$

де q - передбачуване число операторів;

C - коефіцієнт складності програми;

p - коефіцієнт кореляції програми в ході її розробки.

$$Q = 400 \cdot 1.8 \ (1+0.1) = 792 \text{ людино-годин.}$$

Витрати праці на вивчення опису задачі t_u визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{Q \cdot B}{(75 \dots 85) \cdot K}, \text{людино} - \text{годин,} \quad (4.3)$$

де B - коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі B=1.2 ... 1.5;

k - коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності. При стажі роботи від 2 до 3 років він складає 1,0.

Приймемо збільшення витрат праці внаслідок недостатнього опису завдання не більше 50% (B = 1,2). З урахуванням коефіцієнта кваліфікації k = 1.0, отримуємо витрати праці на вивчення опису завдання:

$$t_u = \frac{792 \cdot 1.4}{83 \cdot 1.0} = 13.4 \text{ людино} - \text{годин}.$$

Витрати праці на розробку алгоритму рішення задачі визначаються за формулою:

$$t_a = \frac{Q}{(20 \ldots 25)K}, \text{людино} - \text{годин}, \quad (4.4)$$

де Q – умовне число операторів програми;

k – коефіцієнт кваліфікації програміста.

Підставивши відповідні значення в формулу (4.4), отримаємо:

$$t_a = \frac{792}{21 \cdot 1.0} = 37.6 \text{ людино} - \text{годин}.$$

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20 \ldots 25)K}, \text{людино} - \text{годин}. \quad (4.5)$$

$$t_n = \frac{792}{21 \cdot 1.0} = 37.6 \text{ людино} - \text{годин}.$$

Витрати праці на налагодження програми на ЕОМ:

$$t_{\text{отл}} = \frac{Q}{(4 \ldots 5) \cdot K}, \text{людино} - \text{годин}. \quad (4.6)$$

$$t_{\text{отл}} = \frac{792}{5 \cdot 1.0} = 158.4 \text{ людино} - \text{годин}.$$

Витрати праці на підготовку документації:

$$t_д = t_{др} + t_{до}, людино - годин, \quad (4.7)$$

де $t_{др}$ – трудомісткість підготовки матеріалів і рукопису:

$$t_{др} = \frac{Q}{(12 \dots 20) \cdot K}, людино - годин. \quad (4.8)$$

tдо – трудомісткість редагування, печітки й оформлення документації:

$$t_{до} = 0.75 * t_{др}, людино - годин. \quad (3.9)$$

Підставляючи відповідні значення, отримаємо:

$$t_{др} = \frac{792}{18 \cdot 1.0} = 44 \, людино - годин.$$

$$t_{до} = 0.75 \cdot 44 = 33 \, людино - годин.$$

$$t_д = 44 \cdot 33 = 77 \, людино - годин.$$

$$t = 50 + 13.4 + 37.6 + 37.6 + 158.4 + 77 \; = \; 374 \, людино - годин.$$

4.2. Розрахунок витрат на створення програми

Витрати на створення ПЗ $K_{ПО}$ включають витрати на заробітну плату виконавця програми $З_{ЗП}$ і витрат машинного часу, необхідного на налагодження програми на ЕОМ.

$$К_{ПО} = З_{ЗП} + З_{МВ}, грн. \quad (3.10)$$

Заробітна плата виконавців визначається за формулою:

$$З_{ЗП} = t \cdot С_{пр}, грн, \quad (4.11)$$

де t - загальна трудомісткість, людино-годин;

$С_{пр}$ - середня годинна заробітна плата програміста, грн/година. З урахуванням того, що середня годинна зарплата програміста становить 80 грн / год, отримуємо:

$$З_{ЗП} = 374 \cdot 80 = 29920 \text{ грн.}$$

Вартість машинного часу, необхідного для налагодження програми на ЕОМ:

$$З_{МВ} = t_{отл} \cdot С_{МГ}, грн, \quad (3.12)$$

де $t_{отл}$ - трудомісткість налагодження програми на ЕОМ, год;

$С_{мг}$ - вартість машино-години ЕОМ, грн/год (18 грн/год).

Визначені в такий спосіб витрати на створення програмного забезпечення є частиною одноразових капітальних витрат на створення АСУП.

$$Змв = 158.4 \cdot 18 = 2851.2 \text{ грн.}$$

$$К_{ПО} = 29920 + 2851.2 = 32771 \text{ грн.}$$

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k \cdot F_p}, міс, \quad (3.13)$$

де B_k - число виконавців;

$F_р$ - місячний фонд робочого часу (при 40 годинному робочому тижні $F_р$=176

годин).

$$T = \frac{374}{1 \cdot 176} = 2.1 \text{ міс.}$$

4.3. Маркетингові дослідження ринку збуту розробленого програмного продукту

В результаті дослідження розпізнавання штрих-кодів представлена концепція програмного забезпечення для таких цілей, яка б надавала користувачу зручний та надійний інструмент для декодування. При цьому розроблене ПЗ має бути зрозумілим при використанні і не вимагати багато часу для ознайомлення з ним, а також можна адаптувати під любі цілі, пов’язані з розпізнаванням штрих-кодів.

Таким чином, розроблена програма дозволяє ефективно та швидко аналізувати, знаходити та декодувати штрих-коди. Головна різниця від схожих продуктів полягає в удосконаленні пошуку штрих-коду та декодуванню його навіть з дефектами зображення такими як:

- Дефокусування;
- Низька частота дискретизації;
- Нахил камери;
- Вигин поверхні;
- Нерівномірна освітлюваність;
- Шуми.

Дослідження ринку показали, що аналогічні розробки присутні на ринку, але використовуються обмежено підприємствами. Цінову політику конкурентів оцінити складно, тому що для реалізації аналогічного програмного продукту на ринку, використовується зв'язок між виробником і споживачем напряму, без посередників, в такому випадку ПЗ буде розроблятися для кожного замовника індивідуально.

Будь-яка компанія яка спеціалізується на роздрібній торгівлі може стати в

перспективі замовником. В даній області не можливо представити роботу компанії без сканерів штрих-кодів. Даний пристрій підвищує швидкість робіт на складі при зборі замовлення, переміщення товару, дозволяє швидко формувати касовий чек, підсумовувати вартість, перевіряти вартість товару. Кожен сканер потрібно під'єднувати до якогось терміналу чи комп'ютера, є навіть дистанційні сканери, але вище переліченого вони не відмінять, та й ціна на них на порядок вища.

Програмний продукт розроблений в роботі дає можливість для створення додатку будь-якого характеру і цілей, але пов'язаних зі зчитуванням штрих-кодів. Уявіть можливість робити в даній сфері без масивних сканерів та касових апаратів, а лише зі смартфоном з фото камерою та потрібним ПЗ. Разом з тим пропозиція в цій області дуже обмежена, тому в найближчі роки такі розробки є перспективними і затребуваними.

Розроблена програма не являється складною продукцією, яка потребує спеціального налагодження та обслуговування.

4.4. Оцінка економічної ефективності впровадження ПЗ

Впровадження програмного забезпечення може дозволити підприємству:

- скоротити кількість апаратних засобів (економія на технічному обладнанні);
- користувачі можуть перебувати на віддалені від БД;
- підвищення ефективності та безпеки праці;
- скоротити час та вартість виробництва продукції за рахунок покращення;
- зробити середовище оператора більш гнучким та комфортним;
- підвищити продуктивність праці.

Через наявність лише удосконаленої системи та відсутність програмного забезпечення, яке можна впровадити, неможливо розрахувати економічний ефект, в якому обсязі необхідні інвестиції, який термін окупності і очікуємий прибуток.

Можна зробити висновок, що впровадження нової системи повинно мати позитивний економічний ефект тому, що дана розробка є актуальною та необхідною для

широкого сектору промисловості.

У результаті виконання кваліфікаційної роботи було створено нову систему для розпізнавання штрих-коду EAN-13 на реальних зображеннях. В економічному розділі було визначено трудомісткість на розробку додатку, що складає 374 людино-годин, проведено підрахунок вартості роботи по створенню описаної системи, які склали 32771 грн. та розраховано час на його створення – 2.1 міс. Було проаналізовано та досліджено ринок збуту та потенційних покупців нової системи.

ВИСНОВКИ

В ході роботи було досліджено предметну область та проаналізовано актуальність даної розробки.

У цій роботі представлено алгоритм розпізнавання 1-D штрих-коду EAN-13 на реальних зображеннях. Буди проведені експерименти з великою базою даних, використовуючи реалізацію мовою Python. Результати показують, що алгоритм дуже надійний і до того ж дуже швидкий. При порівнянні з аналогом було виявлено, що алгоритм представлений в роботі справився краще. Наступним кроком буде можливість використовувати мережеве підключення поточних телефонів із камерою або КПК, щоб дозволити використовувати корисні програми. Крім того, буде розширено базу даних штрих-кодів, отриманих за допомогою телефонів із камерою та КПК.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. R. Adelmann, M. Langheinrich, C. Florkemeier: ¨ Toolkit for Bar Code Recognition and Resolving on Camera Phones – Jump Starting the Internet of Things. Workshop on Mobile and Embedded Interactive Systems (MEIS'06) at Informatik, GI LNI, 2006.

2. D. Chai, F. Hock: Locating and Decoding EAN-13 Barcodes from Images Captured by Digital Cameras. 5th Int. Conf. on Information, Communications and Signal Processing, 1595–1599, 2005.

3. E. Joseph, T. Pavlidis: Bar Code Waveform Recognition Using Peak Locations. IEEE Trans. On PAMI, 16(6):630–640, 1998.

4. E. Ohbuchi, H. Hanaizumi, L. A. Hock: Barcode Readers Using the Camera Device in Mobile Phones. In Proc. of the Int. Conf. on Cyberworlds, 260–265, 2004.

5. R. Muniz, L. Junco, A. Otero: A Robust Software Barcode Reader Using the Hough Transform. In Proc. of the Int. Conf. on Information Intelligence and Systems, 313–319, 1999.

6. S. Wachenfeld, S. Terlunen, X. Jiang/ URL: http://cvpr.uni-muenster.de/research/barcode. дата звернення: 1.11.2021.

7. K. Wang, Y. Zou, H. Wang: Bar Code Reading from Images Captured by Camera Phones. 2nd Int. Conf. on Mobile Technology, Applications and Systems, 6–12, 2005.

8. K. Wang, Y. Zou, H. Wang: 1D Bar Code Reading on Camera Phones. Int. Journal of Image and Graphics, 7(3):529–550, 2007.

9. Як читати штрих-код / URL: https://hmarka.ua/ru/articles/kak-chytat-shtryh-kod/. дата звернення: 4.11.2021.

10. Штрих-код товарів та їх розшифровка / URL: https://mozp.org/main /spravochnik-potrebitelya/shtrix-kod-produktov. дата звернення: 4.11.2021.

11. Код країни виробника / URL: https://pzik.ru/uk/barcoding-of-goods-item-code-in-the-invoice/. дата звернення: 4.11.2021.

12. Товарні коди країн-виробників. Стандартизація та сертифікація продукції / URL: https://videoplays.ru/uk/online-services/commodity-codes-of-producer-countries-standardization-and-certification-of-products/. дата звернення: 4.11.2021.

13. Маркування штрих-кодів по країнах. Країна походження товару: обов'язкове маркування, штрих-коди / URL: https://technologyyoung.ru/barcode-labeling-by-country-country-of-origin-mandatory-marking-bar-codes.html. дата звернення: 4.11.2021.

14. Уроки OpenGL. Программирование 3D графики / URL: http://esate.ru. дата звернення: 8.11.2021.

15. Херн Д., Бейкер М.П. Компьютерная графика и стандарт OpenGL = Computer Graphics with OpenGL. - 3-е изд. - М.: Вильямс, 2005. - 1168 с. - ISBN 5-8459-0772-1. дата звернення: 8.11.2021.

16. Перевірка справжності товару по штрих-коду / URL: https://wisegroup.com.ua/shtrix-kody-stran-mira-kak-rasshifrovat-stranu-po-shtrix-kodu/. дата звернення: 11.11.2021.

17. Навчання програмування за допомогою професійних інструментів JetBrains / URL: https://www.jetbrains.com/ru-ru/community/education/#students. дата звернення: 11.11.2021.

18. Розпізнавання штрих-кодів / URL: https://expertnov.ru/ raznoe/raspoznavanie-shtrih-kodov-raspoznavanie-shtrih-koda-barcode-detection-v-google-play-services-fandroid-info.html#i. дата звернення: 12.11.2021.

19. Розпізнавання штрих-кодів на зображеннях за допомогою Python / URL: https://habr.com/ru/company/enterra/blog/244163/. дата звернення: 12.11.2021.

20. Алгоритм розпізнавання штрих-кодів / URL: https://keldysh .ru/papers/2004/prep84/prep2004_84.html. дата звернення: 13.11.2021.

21. Розпізнавання штрих-коду на сильно спотворених зображеннях та зображеннях з низькою роздільною здатністю / URL: http://physics-mathematics.kz/images/pdf/20212/109-114.pdf. дата звернення: 2.12.2021.

22. Пошук та декодування штрих-кодів EAN-13 / URL: https://www.dynamsoft.com/codepool/locating-and-decoding-ean13-python-opencv.html. дата звернення: 3.12.2021.

23. Для чого потрібна мова програмування і якими є критерії її вибору / URL: https://deveducation.com/uk/blog/kakoj-yazyk-programmirovaniya-vybrat-dlya-starta/. дата

звернення: 3.12.2021.

24. Популярність мов програмування / URL: https:// statisticstimes.com/tech/top-computer-languages.php. дата звернення: 5.12.2021.

25. ТОП 6 мов програмування для вивчення у 2021 році / URL: https://nt.ua/blog/top-6-programming-lang-2021. дата звернення: 5.12.2021.

26. Популярність мов програмування / URL: https://uk. wikipedia. org/wiki/ Популярність мов програмування. дата звернення: 5.12.2021.

27. Основні вимоги до написання науково-дослідницької роботи / URL: http://dvman.dnepredu.com/uploads/editor/4165/ 353853/sitepage_62/files/ vimogi_ do_oformlennya_ndr.docx. дата звернення: 6.12.2021.

28. 12 найкращих Python IDE / URL: https://www.softwaretestinghelp.com/python-ide-code-editors/. дата звернення: 7.12.2021.

29. Онлайн редактор блок-схем / URL: https://programforyou.ru/block-diagram-redactor. дата звернення: 10.12.2021.

30. Лутц М. Изучаем Python, 4-е издание. – Пер. с англ. – СПб.: Символ-Плюс, 2011. – 1280 с.

31. Златопольский Д.М. Основы программирования на языке Python. – М.: ДМК Пресс, 2017. – 284 с.

32. Лутц М. Программирование на Python, том I, 4-е издание. – Пер. с англ. – СПб.: Символ-Плюс, 2011. – 992 с.

33. Лутц М. Программирование на Python, том II, 4-е издание. – Пер. с англ. – СПб.: Символ-Плюс, 2011. – 992 с.

34. Гэддис Т. Начинаем программировать на Python. – 4-е изд.: Пер. с англ. – СПб.: БХВ-Петербург, 2019. – 768 с.

35. Лучано Рамальо Python. К вершинам мастерства. – М.: ДМК Пресс, 2016. – 768 с.

36. Свейгарт, Эл. Автоматизация рутиных задач с помощью Python: практическое руководство для начинающих. Пер. с англ. — М.: Вильямс, 2016. – 592 с.

37. Рейтц К., Шлюссер Т. Автостопом по Python. – СПб.: Питер, 2017. – 336 с.:

ил. – (Серия «Бестселлеры O’Reilly»).

38. Любанович Билл Простой Python. Современный стиль программирования. – СПб.: Питер, 2016. – 480 с.

39. Федоров, Д. Ю. Программирование на языке высокого уровня Python : учебное пособие для прикладного бакалавриата / Д. Ю. Федоров. – 2-е изд., перераб. и доп. – Москва : Издательство Юрайт, 2019. – 161 с. – (Бакалавр. Прикладной курс). – ISBN 978-5-534-10971-9. – Текст: электронный // ЭБС Юрайт / URL: https://urait.ru/bcode/437489. дата звернення: 15.12.2021.

40. Шелудько, В. М. Основы программирования на языке высокого уровня Python: учебное пособие / В. М. Шелудько. – Ростов-на-Дону, Таганрог: Издательство Южного федерального университета, 2017. – 146 с. – ISBN 978-5-9275-2649-9. – Текст: электронный // Электронно-библиотечная система IPR BOOKS / URL: http://www.iprbookshop.ru/87461.html. дата звернення: 15.12.2021.

41. Шелудько, В. М. Язык программирования высокого уровня Python. Функции, структуры данных, дополнительные модули: учебное пособие / В. М. Шелудько. – Ростов-на-Дону, Таганрог: Издательство Южного федерального университета, 2017. – 107 с. – ISBN 978-5-9275-2648-2. – Текст: электронный // Электронно-библиотечная система IPR BOOKS / URL: http://www.iprbookshop.ru/87530.html. дата звернення: 15.12.2021.

42. Доусон М. Программируем на Python. – СПб.: Питер, 2014. – 416 с.

43. Прохоренок Н.А. Python 3 и PyQt. Разработка приложений. – СПб.: БХВ-Петербург, 2012. – 704 с.

44. Прохоренок Н.А. Самое необходимое. — СПб.: БХВ-Петербург, 2011. — 416 с.

ДОДАТОК А

ЛІСТИНГ ПРОГРАМИ

```
import cv2
import numpy as np
import glob

def detect(img):
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    ret, thresh =cv2.threshold(gray, 127, 255, cv2.THRESH_BINARY)
    thresh = cv2.bitwise_not(thresh)
    kernel = np.ones((3, 20), np.uint8)
    thresh = cv2.dilate(thresh, kernel)
    original_sized = cv2.resize(thresh, (img.shape[1], img.shape[0]),
interpolation=cv2.INTER_AREA)
    contours, hierarchy = cv2.findContours(original_sized, cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)
    candidates = []
    index = 0
    added_index = []
    for cnt in contours:
        rect = cv2.minAreaRect(cnt)
        box = cv2.boxPoints(rect)
        box = np.int0(box)
        cropped = crop_rect(rect, box, img)
        width = cropped.shape[1]
        child_index = hierarchy[0][index][2]
        if width > 95:
            has_overlapped = False
            if child_index in added_index:
                has_overlapped = True
            if has_overlapped == False:
                added_index.append(index)
                candidate = {"cropped": cropped, "rect": rect}
                candidates.append(candidate)
        index = index + 1
    return candidates
```

```
def crop_rect(rect, box, img):
    W = rect[1][0]
    H = rect[1][1]
    Xs = [i[0] for i in box]
    Ys = [i[1] for i in box]
    x1 = min(Xs)
    x2 = max(Xs)
    y1 = min(Ys)
    y2 = max(Ys)
    # Center of rectangle in source image
    center = ((x1 + x2) / 2, (y1 + y2) / 2)
    # Size of the upright rectangle bounding the rotated rectangle
    size = (x2 - x1, y2 - y1)
    # Cropped upright rectangle
    cropped = cv2.getRectSubPix(img, size, center)

    angle = rect[2]
    if angle != 90:  # need rotation
        if angle > 45:
            angle = 0 - (90 - angle)
        else:
            angle = angle
        # print(angle)

        M = cv2.getRotationMatrix2D((size[0] / 2, size[1] / 2), angle, 1.0)

        cropped = cv2.warpAffine(cropped, M, size)
        croppedW = H if H > W else W
        croppedH = H if H < W else W
        # Final cropped & rotated rectangle
        croppedRotated = cv2.getRectSubPix(cropped, (int(croppedW), int(croppedH)), (size[0] /
2, size[1] / 2))
        return croppedRotated
    return cropped

def decode(img):
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    ret, thresh = cv2.threshold(gray, 200, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)
    thresh = cv2.bitwise_not(thresh)
    ean13 = None
    is_valid = None
    # scan lines
    for i in range(img.shape[0] - 1, 0, -1):
```

```
        try:
            ean13, is_valid = decode_line(thresh[i])
        except Exception as e:
            pass
        if is_valid:
            break
    return ean13, is_valid, thresh

def decode_line(line):
    barcodes = read_barcode(line)
    left_guard, left_patterns, center_guard, right_patterns, right_guard =
classify_barcode(barcodes)
    convert_patterns_to_length(left_patterns)
    convert_patterns_to_length(right_patterns)
    left_codes = read_patterns(left_patterns, is_left=True)
    right_codes = read_patterns(right_patterns, is_left=False)
    ean13 = get_ean13(left_codes, right_codes)
    is_valid = verify(ean13)
    return ean13, is_valid

def read_barcode(line):
    replace_255_to_1(line)
    barcodes = []
    current_length = 1
    for i in range(len(line) - 1):
        if line[i] == line[i + 1]:
            current_length = current_length + 1
        else:
            barcodes.append(current_length * str(line[i]))
            current_length = 1
    barcodes.pop(0)
    return barcodes

def convert_patterns_to_length(patterns):
    for i in range(len(patterns)):
        patterns[i] = len(patterns[i])

def read_patterns(patterns, is_left=True):
    codes = []
    for i in range(6):
        start_index = i * 4
        sliced = patterns[start_index:start_index + 4]
        m1 = sliced[0]
```

```
        m2 = sliced[1]
        m3 = sliced[2]
        m4 = sliced[3]
        total = m1 + m2 + m3 + m4
        tmp1 = (m1 + m2) * 1.0
        tmp2 = (m2 + m3) * 1.0
        at1 = get_AT(tmp1 / total)
        at2 = get_AT(tmp2 / total)
        if is_left:
            decoded = decode_left(at1, at2, m1, m2, m3, m4)
        else:
            decoded = decode_right(at1, at2, m1, m2, m3, m4)
        codes.append(decoded)
    return codes


def get_AT(value):
    if value < 2.5 / 7:
        return 2
    elif value < 3.5 / 7:
        return 3
    elif value < 4.5 / 7:
        return 4
    else:
        return 5

def decode_left(at1, at2, m1, m2, m3, m4):
    patterns = {}
    patterns["2,2"] = {"code": "6", "parity": "O"}
    patterns["2,3"] = {"code": "0", "parity": "E"}
    patterns["2,4"] = {"code": "4", "parity": "O"}
    patterns["2,5"] = {"code": "3", "parity": "E"}
    patterns["3,2"] = {"code": "9", "parity": "E"}
    patterns["3,3"] = {"code": "8", "parity": "O", "alter_code": "2"}
    patterns["3,4"] = {"code": "7", "parity": "E", "alter_code": "1"}
    patterns["3,5"] = {"code": "5", "parity": "O"}
    patterns["4,2"] = {"code": "9", "parity": "O"}
    patterns["4,3"] = {"code": "8", "parity": "E", "alter_code": "2"}
    patterns["4,4"] = {"code": "7", "parity": "O", "alter_code": "1"}
    patterns["4,5"] = {"code": "5", "parity": "E"}
    patterns["5,2"] = {"code": "6", "parity": "E"}
    patterns["5,3"] = {"code": "0", "parity": "O"}
    patterns["5,4"] = {"code": "4", "parity": "E"}
```

```
    patterns["5,5"] = {"code": "3", "parity": "O"}
    pattern_dict = patterns[str(at1) + "," + str(at2)]
    code = 0
    use_alternative = False
    if int(at1) == 3 and int(at2) == 3:
        if m3 + 1 >= m4:
            use_alternative = True
    if int(at1) == 3 and int(at2) == 4:
        if m2 + 1 >= m3:
            use_alternative = True
    if int(at1) == 4 and int(at2) == 3:
        if m2 + 1 >= m1:
            use_alternative = True
    if int(at1) == 4 and int(at2) == 4:
        if m1 + 1 >= m2:
            use_alternative = True
    if use_alternative:
        code = pattern_dict["alter_code"]
    else:
        code = pattern_dict["code"]
    final = {"code": code, "parity": pattern_dict["parity"]}
    return final

def decode_right(at1, at2, m1, m2, m3, m4):
    patterns = {}
    patterns["2,2"] = {"code": "6"}
    patterns["2,4"] = {"code": "4"}
    patterns["3,3"] = {"code": "8", "alter_code": "2"}
    patterns["3,5"] = {"code": "5"}
    patterns["4,2"] = {"code": "9"}
    patterns["4,4"] = {"code": "7", "alter_code": "1"}
    patterns["5,3"] = {"code": "0"}
    patterns["5,5"] = {"code": "3"}
    pattern_dict = patterns[str(at1) + "," + str(at2)]
    code = 0
    use_alternative = False
    if int(at1) == 3 and int(at2) == 3:
        if m3 + 1 >= m4:
            use_alternative = True
    if int(at1) == 4 and int(at2) == 4:
        if m1 + 1 >= m2:
            use_alternative = True
    if use_alternative:
```

```
        code = pattern_dict["alter_code"]
    else:
        code = pattern_dict["code"]
    final = {"code": code}
    return final

def classify_barcode(barcodes):
    left_guard = barcodes[0:3]
    left_patterns = barcodes[3:27]
    center_guard = barcodes[27:32]
    right_patterns = barcodes[32:56]
    right_guard = barcodes[56:59]
    return left_guard, left_patterns, center_guard, right_patterns, right_guard

def verify(ean13):
    weight = [1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1, 3]
    weighted_sum = 0
    for i in range(12):
        weighted_sum = weighted_sum + weight[i] * int(ean13[i])
    weighted_sum = str(weighted_sum)
    checksum = 0
    units_digit = int(weighted_sum[-1])
    if units_digit != 0:
        checksum = 10 - units_digit
    else:
        checksum = 0
    if checksum == int(ean13[-1]):
        return True
    else:
        return False


def get_ean13(left_codes, right_codes):
    ean13 = ""
    ean13 = ean13 + str(get_first_digit(left_codes))
    for code in left_codes:
        ean13 = ean13 + str(code["code"])
    for code in right_codes:
        ean13 = ean13 + str(code["code"])
    return ean13


def replace_255_to_1(array):
```

```
    for i in range(len(array)):
        if array[i] == 255:
            array[i] = 1


def get_first_digit(left_codes):
    parity_dict = {}
    parity_dict["OOOOOO"] = 0
    parity_dict["OOEOEE"] = 1
    parity_dict["OOEEOE"] = 2
    parity_dict["OOEEEO"] = 3
    parity_dict["OEOOEE"] = 4
    parity_dict["OEEOOE"] = 5
    parity_dict["OEEEOO"] = 6
    parity_dict["OEOEOE"] = 7
    parity_dict["OEOEEO"] = 8
    parity_dict["OEEOEO"] = 9
    parity = ""
    for code in left_codes:
        parity = parity + code["parity"]
    return parity_dict[parity]


def decoding_found_image(image):
    result = "none"
    candidates = detect(image)
    for i in range(len(candidates)):
        candidate = candidates[i]
        cropped = candidate["cropped"]

        ean13, is_valid, thresh = decode(cropped)
        if is_valid:
            result = str(ean13)
            break
    return result

def decoding_real_time(image):

    result = "none"
    ean13, is_valid, thresh = decode(image)
    if is_valid:
        result = str(ean13)
    return result
```

```
if __name__ == "__main__":

    X_data = []
    files = glob.glob("C:/BarCode/testImage/test/*.jpg")
    for myFile in files:
        image = cv2.imread(myFile)
        X_data.append(image)
    print('X_data shape:', np.array(X_data).shape)
    counter = 0
    for im in X_data:
        result_dict = decoding_real_time(im)
        #result_dict = decoding_found_image(im)
        if result_dict!= "none":
            print(result_dict)
```

ДОДАТОК Б

ПЕРЕЛІК ДОКУМЕНТІВ НА ОПТИЧНОМУ НОСІЇ

Ім'я файлу	**Опис**
Пояснювальні документи	
Диплом_Пчелєнков.docx	Пояснювальна записка роботи. Документ Word.
Диплом_Пчелєнков.pdf	Пояснювальна записка роботи в форматі PDF
Програма	
Program.rar	Архів. Містить коди програми та відкомпільовану програму
Презентація	
Презентація_Пчелєнков.pptx	Презентація роботи