НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

«На правах рукопису»
УДК 004.89, 004.932.2

«До захисту допущено»
Завідувач кафедри
_____ Едуард ЖАРІКОВ
«___»_____2022 р.

Магістерська дисертація

на здобуття ступеня магістра

за освітньо-науковою програмою «Інженерія програмного забезпечення комп'ютерних систем»

зі спеціальності 121 «Інженерія програмного забезпечення»

на тему: «Методи та програмні засоби виявлення пошкоджених даних при скануванні зображень оптичними сенсорами»

Виконав (-ла):
студент (-ка) ІІ курсу, групи ІТ-01мн
Корзун Ілля Михайлович _____

Керівник:
професор, д.т.н.
Павлов Олександр Анатолійович _____

Рецензент:
доцент кафедри ІСТ, к.т.н.
Жураковська Оксана Сергіївна _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2022 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Рівень вищої освіти – другий (магістерський)

Спеціальність – 121 «Інженерія програмного забезпечення»

Освітньо-наукова програма «Інженерія програмного забезпечення комп'ютерних систем»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Едуард ЖАРІКОВ

«___»_____2022р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Корзуну Іллі Михайловичу

1. Тема дисертації «Методи та програмні засоби виявлення пошкоджених даних при скануванні зображень оптичними сенсорами», науковий керівник дисертації Павлов Олександр Анатолійович, д.т.н., професор, затверджені наказом по університету від «24» квітня 2022 р. № 88.

2. Термін подання студентом дисертації «6» червня 2022 р.

3. Об'єкт дослідження – процес виявлення аномалій при скануванні банкнот оптичними сенсорами.

4. Предмет дослідження – алгоритмічне та програмне забезпечення виявлення аномалій при скануванні зображень оптичними сенсорами.

5. Перелік завдань, які потрібно розробити – аналіз проблеми та існуючих рішень; розроблення критеріїв оцінки дійсності результатів сканування банкнот та програмного забезпечення виявлення їх аномальних реалізацій; дослідження ефективності запропонованих критеріїв та проведення експериментального дослідження для перевірки досягнення поставленої мети.

6. Орієнтовний перелік графічного (ілюстративного) матеріалу – функціональна діаграма програмного забезпечення, об'єктна модель пакетів бібліотеки.

7. Орієнтовний перелік публікацій – тези доповіді «Методи та програмні засоби виявлення пошкоджених даних при скануванні зображень оптичними сенсорами» на XII Міжнародній науково-практичній конференції «Комплексне забезпечення якості технологічних процесів та систем – 2022», тези доповіді «Методи і програмні засоби виявлення аномалій при скануванні банкнот оптичними сенсорами» на другій всеукраїнській науково-практичній конференції молодих вчених та студентів «Інженерія програмного забезпечення і передові інформаційні технології» (SoftTech-2022 весна).

8. Дата видачі завдання «30» вересня 2020 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання	Примітка
1	Ознайомлення з предметною областю, вивчення літератури	23.08.2021	
2	Аналіз наявних методів та засобів виявлення аномалій	29.10.2021	
3	Постановка та формалізація задачі дослідження	12.11.2021	
4	Розробка критеріїв виявлення аномальних реалізацій сканування банкнот	06.12.2021	
5	Виконання експериментального дослідження	30.12.2021	
6	Проєктування програмного забезпечення	27.01.2022	
7	Розробка програмного забезпечення	12.04.2022	
8	Підготовка доповіді на конференції	22.05.2022	
9	Оформлення пояснювальної записки	30.05.2022	
10	Подання дисертації на попередній захист	06.06.2022	
11	Подання дисертації на захист	17.06.2022	

Студент Ілля КОРЗУН

Науковий керівник Олександр ПАВЛОВ

РЕФЕРАТ

Розмір пояснювальної записки – 166 аркушів, містить 44 ілюстрації, 4 додатки.

Актуальність теми. У роботі розглянуто проблему в області апаратної ідентифікації фальшованих банкнот на основі оптичних сигналів щодо налаштування програмного забезпечення систем валідації, точність і надійність роботи яких залежить від якості даних, використаних для підготовки моделей розпізнавання. Показано неефективність ручних методів забезпечення релевантності тренувальної вибірки та виявлено проблему в автоматизації процесу виявлення аномальних екземплярів даних, що дозволить суттєво скоротити час збирання даних та ефективність результуючих моделей валідації банкнот.

Мета дослідження: підвищення ефективності розпізнавання фальшивих банкнот за рахунок автоматизації процесу виявлення пошкоджених сканів.

Об'єкт дослідження: процес виявлення аномалій при скануванні банкнот оптичними сенсорами.

Предмет дослідження: алгоритмічне та програмне забезпечення виявлення аномалій при скануванні зображень оптичними сенсорами.

Для реалізації поставленої мети **сформульовані та вирішені наступні завдання**:

– аналіз характеристик дійсних банкнот на основі результатів їх сканування;

– аналіз ознак аномалій в даних результатів сканування банкнот;

– дослідження існуючих методів виявлення аномалій на зображеннях та багатовимірних даних;

– розробка критеріїв та алгоритмів для їх реалізації для виявлення аномальних банкнот на основі їх спостережуваних і теоретичних ознак;

– експериментальне дослідження ефективності запропонованих критеріїв на змодельованих та реальних даних;

– розробка кросплатформеної бібліотеки для програмної реалізації запропонованих критеріїв та обробки даних сканування банкнот;

– розробка програмного засобу для аналізу та автоматичного виявлення шахрайських банкнот на основі їх правильних реалізацій.

Наукова новизна результатів магістерської дисертації полягає в тому, що запропоноване алгоритмічне забезпечення для виявлення аномальних реалізацій сканування банкнот засноване на чотирьох емпіричних та двох теоретичних критеріях нормальності даних; розроблено архітектуру та програмний інтерфейс бібліотеки, яка реалізує запропоновані методи виявлення аномалій та сутність, що виконує роль декоратора і контейнера для низькорівневих структур даних, сканів, і процедур для роботи з ними у вигляді двовимірних матриць.

Практичне значення отриманих результатів полягає в тому, що розроблено програмне забезпечення, яке реалізує експериментально обґрунтований алгоритм виявлення аномалій на реалізаціях сканування банківських білетів оптичними сенсорами, що може застосовуватися в автоматизованих та напівавтоматизованих системах обробки великих масивів даних.

Зв’язок з науковими програмами, планами, темами. Робота виконувалась на кафедрі інформатики та програмної інженерії Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

Апробація. Наукові положення дисертації пройшли апробацію на XII Міжнародної науково-практичної конференції «Комплексне забезпечення якості технологічних процесів та систем» (КЗЯТПС-2022) – м. Чернігів.

Публікації. Наукові положення дисертації опубліковані в:

Корзун І., Павлов О. Методи та програмні засоби виявлення пошкоджених даних при скануванні зображень оптичними сенсорами // XII Міжнародна науково-практична конференція «Комплексне забезпечення якості технологічних процесів та систем» (КЗЯТПС-2022). Секція 8. Інформаційні системи та технології. Комп’ютерна інженерія. Кібербезпека. Інформаційно-вимірювальні системи. Матеріали конференції. – Чернігів. – 2022. 26–27 травня 2022 р. – С. 185 – 186.

Корзун І., Павлов О. Методи і програмні засоби виявлення аномалій при скануванні банкнот оптичними сенсорами // Друга Всеукраїнська науково-

практична конференція молодих вчених та студентів «Інженерія програмного забезпечення і передові інформаційні технології» (SoftTech-2022). Секція кафедри інформатики та програмної інженерії. Матеріали конференції. – Київ. – 2021. 15–16 червня 2022 р.

Ключові слова: БАНКНОТА, ВИЯВЛЕННЯ АНОМАЛІЙ, ВІДСТАНЬ МАХАЛАНОБІСА, МЕТОД ГОЛОВНИХ КОМПОНЕНТ, ЗМЕНШЕННЯ ВИМІРНОСТІ, ПОМИЛКА ВІДТВОРЕННЯ.

ABSTRACT

Explanatory note size – 166 pages, contains 44 illustrations, 4 applications.

Topicality. Examines the problem in the field of identification of counterfeit banknotes based on optical signals regarding the configuration of validation systems software. The accuracy and reliability of those systems mostly depends on the quality of data used to prepare recognition models. The work has identified inefficiency of manual methods of ensuring the relevance of the training samples and the need to automate the process of anomaly detection to reduce the time needed for production.

The aim of the study.

Improving the efficiency of the counterfeits recognition by automating the process of detecting damaged scans.

Object of research: the process of anomaly detection when scanning banknotes with optical sensors.

Subject of research: algorithms and software for anomalies detection when scanning images with optical sensors.

To achieve this goal, formulated and solved the **following tasks**:

– analysis of valid banknotes characteristics based on their scanning data;

– analysis of anomalies signs in the data of banknote scanning results;

– study of existing methods for anomalies detection in images and multidimensional data;

– development of criteria of genuineness for detecting anomalous banknotes based on their observed and theoretic features;

– experimental study of the effectiveness of the proposed criteria on simulated and real data;

– development of a cross platform mathematical library for software implementation of the proposed criteria and processing of the banknotes scanning data;

– development of software for the fraudulent banknotes analysis and detection;

The scientific novelty of the results of the master's dissertation is the proposed algorithmic software for detecting anomalous implementations of banknote scanning

based on four empirical and two theoretical criteria of data normality. The architecture and software interface of the library has been developed, which implements the proposed methods for detecting anomalies and the interface for low-level data structures, scans, and procedures for working with them in the form of two-dimensional matrices

The practical value of the obtained results is the software developed, which implements an experimentally substantiated algorithm for detecting anomalies in the implementation of banknote scanning by optical sensors, which can find its application in automated and semi-automated systems for processing large data sets.

Relationship with working with scientific programs, plans, topics. Work was performed at the Department of Informatics and Software Engineering of the National Technical University of Ukraine «Kyiv Polytechnic Institute. Igor Sikorsky».

Approbation. The scientific provisions of the dissertation tested at the XII International scientific-practical conference «Integrated quality assurance of technological processes and systems» (KZYATPS-2022) – Chernihiv.

Publications. The scientific provisions of the dissertation published in:

Korzun I., Pavlov O. Methods and software for detecting damaged data when scanning images with optical sensors // XII International scientific-practical conference «Integrated quality assurance of technological processes and systems» (KZYATPS-2022). Section 8. Information Systems and Technologies. Computer Engineering. Cybersecurity. Information and Measurement Systems. Conference materials. – Chernihiv. – 2022. May 26–27, 2022 – P. 185 – 186.

Korzun I., Pavlov O. Methods and software for detecting damaged data when scanning banknotes with optical sensors // Second all-ukrainian scientific and practical conference of young scientists and students «Software engineering and advanced information technologies» (SoftTech-2021). Section of the Department of Informatics and Software Engineering. Conference materials. – Kyiv. – 2022. June 15–16, 2022.

Keywords: BANKNOTE, ANOMALY DETECTION, MAHALANOBIS DISTANCE, PRINCIPAL COMONENTS ANALYSIS, DIMENSIONALITY REDUCTION, RECONSTRUCTION ERROR.

ЗМІСТ

ВСТУП

Апаратні засоби валідації банківських білетів є широко вживаним інструментом у різних сферах комерції. З кожним роком обсяги продажів даної технології зростають, і пропорційно – частішають випадки підробки грошей. Для ефективного виявлення фальшивок до аналізу все більше залучають дані з валідаторів кінцевих користувачів – і стрімко збільшуються їх обсяги, що мають пройти ретельну візуальну перевірку перед використанням у валідації. Існуючий процес сканування банкнот передбачає ручне введення та огляд вихідних сигналів на предмет механічної чи апаратної помилки, або людської неуважності, спричиненої надмірною рутинністю та довготривалістю роботи операторів. Із подальшим переходом компанії на дистанційне сканування грошей зростає частка спотворених сигналів, часу необхідного на їх перевірку – і питома вартість вихідної продукції. Автоматизація процесу виявлення пошкоджених сигналів дозволить суттєво скоротити час оновлення програмного забезпечення систем. Розробка алгоритму, що підвищить ефективність виявлення аномалій сканування та зменшить навантаження на працівників та стане потужним інструментом розпізнавання фальшивок відповідно до актуальних вимог центральних банків.

Метою дослідження є підвищення ефективності розпізнавання фальшивих банкнот за рахунок підвищення автоматизації процесу виявлення пошкоджених сканів.

Завдання, що вирішуються для досягнення поставленої мети:

– аналіз характеристик дійсних банкнот на основі результатів їх сканування;

– аналіз ознак аномалій в даних результатів сканування банкнот;

– дослідження існуючих методів виявлення аномалій на зображеннях та багатовимірних даних;

– розробка критеріїв справжності для виявлення аномальних банкнот на основі їх спостережуваних і теоретичних ознак;

– експериментальне дослідження ефективності запропонованих критеріїв на змодельованих та реальних даних;

– розроблення математичної бібліотеки для програмної реалізації запропонованих критеріїв та обробки даних сканування банкнот;

– розробка програмного забезпечення для аналізу та автоматичного виявлення шахрайських банкнот на основі їх правильних реалізацій.

Об’єкт дослідження: процес виявлення аномалій при скануванні банкнот оптичними сенсорами.

Предмет дослідження: алгоритмічне та програмне забезпечення виявлення аномалій при скануванні зображень оптичними сенсорами.

Методи дослідження базуються на основних положеннях аналізу даних, виявлення аномалій, комп’ютерної графіки, принципах об’єктно-орієнтованого програмування.

Наукова новизна полягає в обґрунтуванні на основі аналізу причин виникнення аномалій на вихідних сигналах з оптичних сенсорів та критичного аналізу методів, що автоматизують процес їх виявлення, вибору методу, що використовує міру відстані Махалонобіса та розробці його унікальної алгоритмізації з урахуванням специфіки виявлення аномалій в результаті сканування банкнот; реалізації людино-машинної взаємодії через використання трьох формальних критеріїв та реакції оператора; розробці програмного інтерфейсу для відкритої математичної бібліотеки GNU Scientific Library, що дозволило використовувати її у мовах програмування високого рівня.

Практичне значення отриманих результатів полягає в критичному аналізі методів автоматизації процесу виявлення аномалій на зображеннях банківських білетів, отриманих за допомогою оптичних сенсорів, що може застосовуватися в автоматизованих та напівавтоматизованих системах обробки великих масивів зображень банківських білетів.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ

1.1 Опис предметної області

Поняття скана вживається в різних областях при відтворенні за допомогою спеціального обладнання частини матеріального об’єкта як цифрового зображення. Тому необхідно спочатку визначити, що таке банкноти, яка в них структура і в чому особливості їх розпізнавання та які наразі існують засоби для автоматизації цього процесу.

Банківським білетом, банкнотою, називають грошовий знак, виготовлений з паперу, щільної тканини (зазвичай, шовку), металу або пластику, в більшості випадків прямокутної форми із нанесеними на нього чи в ньому захисними елементами у вигляді графічних примітивів, символів або зображень через включення інших матеріалів (наприклад, пластику, металевих стрічок та волосків) або додаткових шарів фарб (водяних знаків, голограм) [1]. Приклад банкноти наведено на рисунку 1.1.

Рисунок 1.1 – Банкнота номіналом двадцять гривень та білети старших номіналів Національного банку України

Апаратний засіб, призначений для оперативного визначення відповідності грошових знаків вимогам до їх дійсності та цілісності, валідації, називають детектором банкнот чи валідатором [2]. Задачею такого пристрою є прийом і збирання справжніх банкнот та бракування і повернення власнику фальшивих.

Зазвичай, такі пристрої, як видно на рисунку 1.2, складаються з трьох послідовних частин: вхідного отвору, пропускного каналу та накопичуючого контейнера.

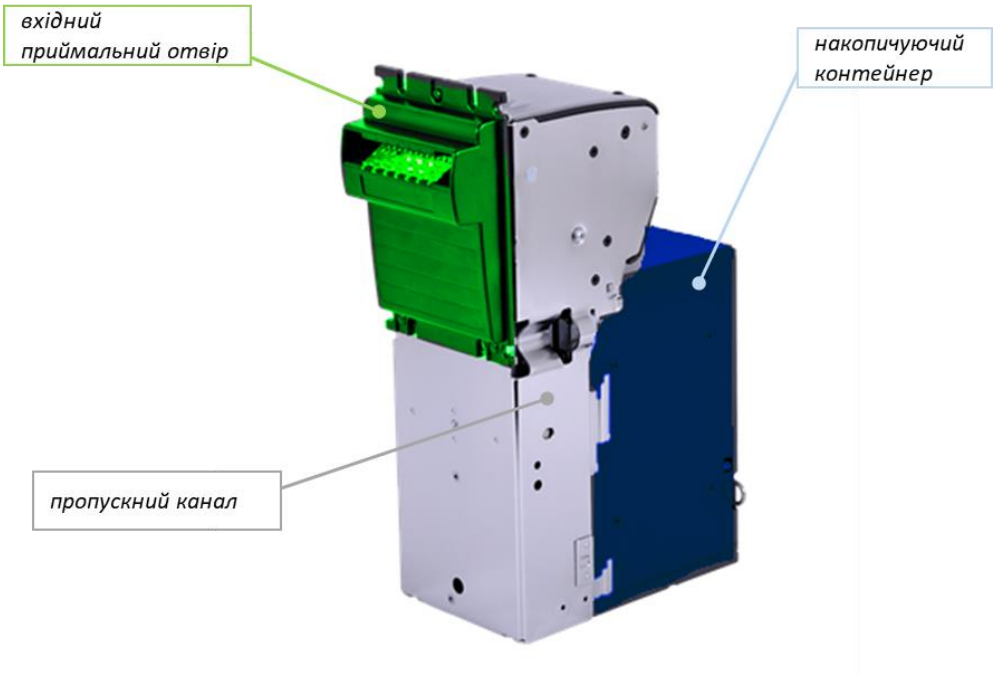


Рисунок 1.2 – Детектор банкнот MSM компанії CashCode

Перший елемент пристрою може бути оснащений механізмом вирівнювання банкноти, аби вона пройшла далі центрованою відносно пропускного каналу. Уздовж стінок пропускного каналу розміщуються давачі різного принципу дії (магнітні, оптичні, механічні), здатні фіксувати всі безпекові властивості цінного паперу. У контейнері збираються банкноти, що успішно пройшли перевірку та задовільняють вимогам справжності.

Рішенням задачі валідації є механізм автоматичної експертизи грошових знаків, який відповідає вимогам емісійних банків щодо точності та надійності фільтрації підробок від оригіналів. Отримання дозволу продавати і використовувати за призначенням новий детектор передбачає проходження ним сертифікації, якою відповідальний орган контролює якість матеріалу, що повертається до нього з місць обігу грошової одиниці. Потраплянню пристроїв на ринок передує їх налаштування: калібрування сенсорів під умови цільового середовища (перелік допустимих валют та номіналів, кліматичні умови та

особливості використання), конфігурація програмного забезпечення, визначення параметрів математичних моделей та алгоритмів.

У контексті захисту цінних паперів від підробок розглядаємо банкноту як множину сканів – багатоспектральне зображення. Кожен скан є сукупністю дискретних сигналів, отриманих з оптичних сенсорів у результаті вимірювання та аналого-цифрового перетворення електромагнітного опромінення в заданих межах довжин хвиль спрямованого на папір грошового знаку. Під процесом побудови дискретного представлення об'єкту з вихідних значень сенсора надалі маємо на увазі сканування, результати якого можна представити на прикладі рисунку 1.3.

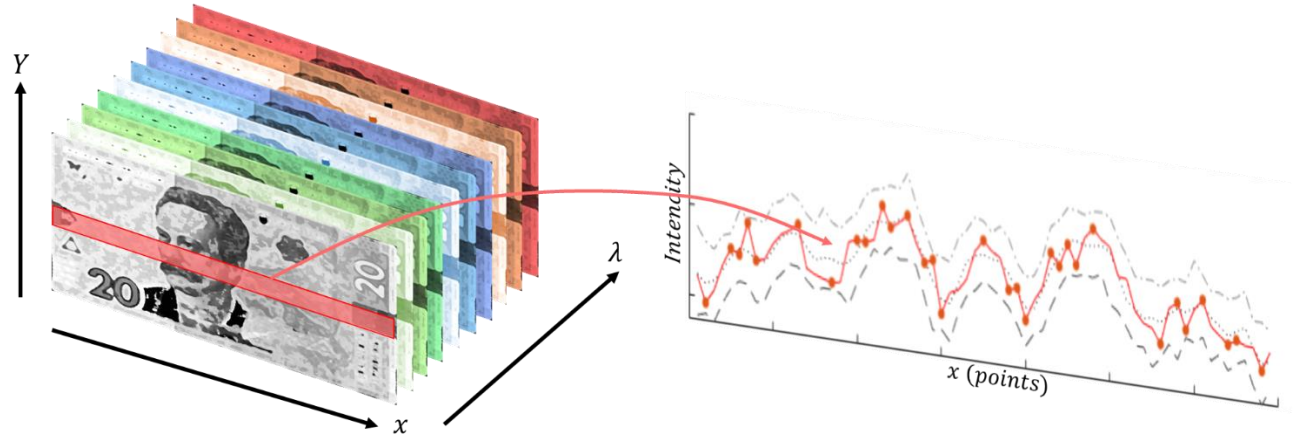


Рисунок 1.3 – Багатоспектральне зображення банкноти на прикладі фотографії

Результатом поздовжнього сканування паперу одним сенсором називаємо трек, $T = \{x_i, i = \overline{1, v}\}$ – упорядкована послідовність дискретних значень x_i інтенсивності оптичного сигналу довжини хвилі λ, що відповідає певному кольору світла, де v – це кількість значень x у даній послідовності, обчислена як відношення кроку дискретизації неперервного сигналу до його довжини. Сукупність треків T_j послідовних частин банкноти складають її скан, $S = \{T_j, j = \overline{1, h}\}$, де h – це кількість треків, що обчислюється як відношення відстані між сусідніми сенсорами до фактичної ширини паперу (кількість давачів). Кожен скан формує амплітудну характеристику банківського білету – зображення у певному спектрі. Множину сканів далі називатимемо записом, $R = \{S_i, i = \overline{1, n}\}$, де n – кількість сканів (кольорів) у ньому, тобто спектрів у яких пристрій виконує сканування вхідного паперу.

Локалізація на сканах ознак підробки (скотчу, розривів, вклейок) чи захисту (голограм, водяних знаків, рисунків) та аналіз сили їх вираження (як центру мас у певних заданих межах) дозволяє приймати рішення, щодо дійсності певної банкноти і її відповідності вимогам емісійного банку. Налаштування механізму, що виконує дану експертизу, передбачає накопичення і постійне оновлення бібліотеки сканів: оригіналів та підробок. Від їх кількості та якості залежить точність приладів.

1.2 Проблема виявлення пошкоджених сканів банкнот

Метою налаштування детекторів є пошук такої конфігурації програмного забезпечення, яка б уможливлювала ідентифікацію максимальної кількості підробок та повернення користувачу мінімальної кількості оригіналів. Використання некоректних даних на етапі конфігурації негативно впливає на влучність і повноту розпізнавання через ефекти перенавчання та недонавчання: спрацювання методів пошуку ознак підробок на оригінальних даних, що їх не містять, та зниження чутливості цих методів до підробок у робочих умовах.

Протоколи збору даних зазвичай включають запис вимірювань, введення значень в базу даних та обчислення похідних характеристик таких, як захоплення границь банкноти та визначення куту її повороту відносно напряму введення. Багато з цих кроків виконуються вручну, і на будь-якому етапі можуть бути допущені помилки, вносячи некоректні значення у вибірку [3].

Некоректними вважаємо записи банкнот, які містять структурні чи параметричні помилки викликані наступними причинами:

а) нестабільним з’єднанням:

1) невідповідність вхідній множини сканів та їх кількості очікуваному переліку;

2) нульові значення та/або відсутність треків чи їх частин;

б) помилки, спричинені людським фактором:

1) невідповідність введеної банкноти заданому класу (валюті, її номінальному значенню та орієнтації);

2) невідповідність введеної банкноти заданому типу (оригінал чи підробка);

в) апаратними помилками:

1) невідповідність отриманого сигналу сканованій поверхні;

2) неузгодженість чутливості одного сенсора відносно інших (некоректна амплітудна характеристика, відсутність калібрування);

3) зажовування паперу пристроєм і спотворення його зображення;

4) пошкодження паперу при введенні та його неадекватність наявним записам класу.

Спотворення, пов’язані з порушенням структури даних: втратою чи відсутністю деяких сканів, як показано на рисунку 1.4 – призводять до системних помилок та ускладнюють роботу оператора з налаштування детекторів. Проблему вирішують програмно через порівняння кількості сканів та треків у кожному з них із набором зазначеним у специфікації детектора. Браковані записи, виявлені в такий спосіб, видаляють з бази даних – і виробничий процес поновлюють.

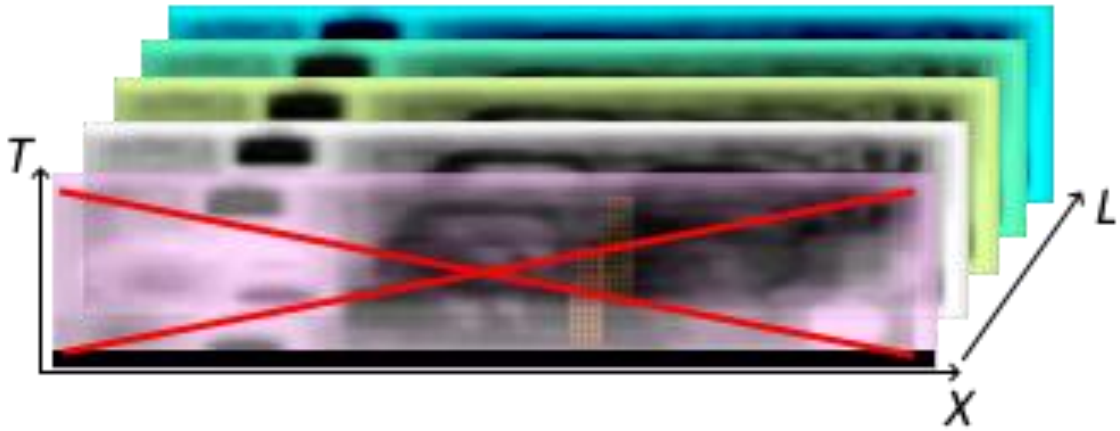


Рисунок 1.4 – Відсутність скана або треку в записі банкноти

В межах даної роботи розглядаємо типи помилок, які виникають на рівні сканів та треків і пов’язані з людським фактором та апаратними збоями. Таким спотворенням характерні розбіжності між очікуваними даними та отримуваними. Їх можна помітити візуально, як показано на рисунку 1.5.

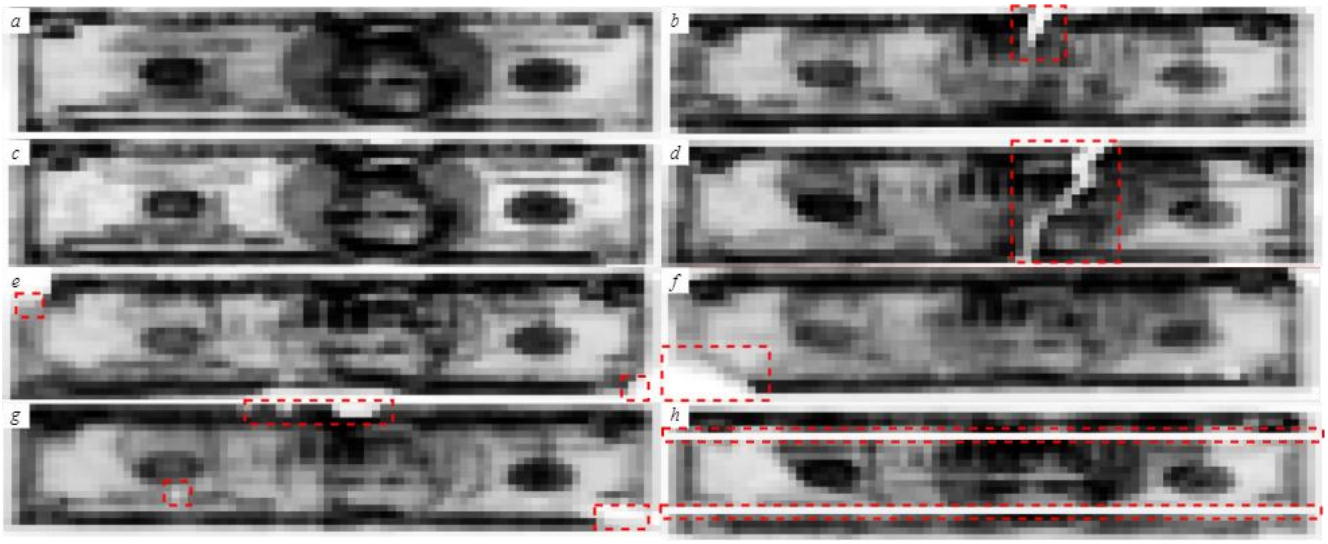


Рисунок 1.5 – Скани одного класу, що містять візуальні спотворення: a, c – нормальні зображення; b, d – фізичні розриви під час введення; e, g – зажовування під час введення; f – загнутий кут; h – пусті треки

На момент проведення дослідження виявлення видимих спотворень виконують повністю в ручну. Процес їх пошуку, зазвичай, включає наступні етапи:

а) отримання нормального зображення банкноти, еталону, через виконання першого акуратного сканування або обрання серед наявних екземплярів;

б) введення наступних банкнот у валідатор, обираючи правильний клас для кожної з них;

в) ретельна візуальна перевірка окремих записів у порівнянні з еталоном у межах заданого класу на предмет аномалій,

1) якщо пошкоджених сканів багато, виконується повністю перевіряються всі записи;

г) прийняття рішення про додавання записів до бази даних.

Класом вважаємо групу сканів, які були зроблені на банкнотах одного типу: одного номінального значення (наприклад, двадцять гривень), одного випуску, та однієї сторони введення (орієнтації) відносно приймального каналу.

Усі екземпляри, які не задовільняють вимогам щодо очікуваного вигляду – сили та характеру відгуку оптичного сигналу на фізичні характеристики паперу – видаляють з бази даних. Більшість спотворень складно виявляти, бо вони виражені

неявно і, як наслідок, здаються нормальними, залишаються непоміченими при тривалій безперервній роботі з великими даними, тисячами сканів.

За необхідності, під час валідації можуть вдаватися до повторного сканування та залучення інших спеціалістів до процесу. Однак дослідження ефективності цих методів свідчать про те, що, хоча вони успішно ідентифікують деякі помилки, їх часто недостатньо, оскільки багато даних залишаються непоміченими [4].

1.3 Постановка задачі дослідження

З аналізу задачі виявлення пошкоджених записів бачимо, що головною проблемою є відсутність методів автоматичної оцінки ступеня відмінності сканів від норми. Існуюче програмне забезпечення вимагає зручних та гнучких засобів їх візуального огляду, які б надавали користувачу можливості ефективно помічати аномалії в даних та швидко формувати уявлення про нормальну форму та зображення сигналів у ручну, аби наступним кроком виконати їх автоматичний пошук.

Кількість екземплярів у тренувальних наборах записів може досягати сотень і тисяч. За умови, що всі вони виконані правильно, без явних спотворень, сигналу з кожного окремо взятого давача відповідатиме певна форма.

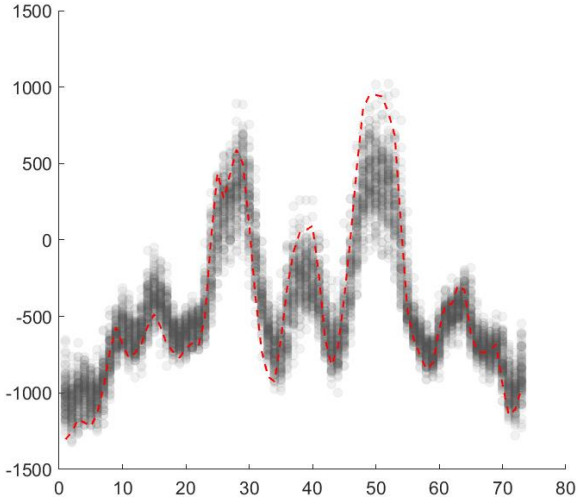


Рисунок 1.6 – Графік щільності сигналу в тренувальному наборі

На рисунку 1.6 бачимо вигляд такої форми на графіку, який побудовано шляхом накладання значень оптичного сигналу отриманих при повторному скануванні одним давачем певної частини банкноти сто разів. Так як положення кожного сенсора є фіксованим, дане представлення формує його статистику з точністю до точки. Темнішим частинам графіку відповідають області з більшою щільністю, що свідчить про вищу імовірність потрапляння сигналу в ці інтервали, а до світлих зон належать точки, які є нетиповими для нього. Дана характеристика сигналу має бути досліджена. У разі виявлення та підтвердження походження варіацій з певного статистичного розподілу – його можна буде використати як математичну модель сканів.

Модель очікування сканів також можна назвати його нормальною формою. Існуючий процес валідації даних передбачає ручне порівняння рівня сигналу (центру мас чи середньої амплітуди) та форми треків з їх очікуванням згідно з досвідом оператора. Отже, пошук спотворень має засновуватися на прийнятій практиці та передбачати співставлення даного зображення чи графіку з еталоном. Таким чином, необхідно визначити: у який спосіб має обиратися еталон так, аби частка відхилень від норми була мінімальною, та на основі чого буде прийняте попереднє рішення щодо відповідності еталону заданим характеристикам.

Поява спотворень у сигналі супроводжується суттєвим відхиленням від його нормального вигляду – або математичної моделі, якщо таку вдасться побудувати. З огляду банкнот та їх сканів, як показано на рисунку 1.7, бачимо, що дискретне представлення кожної банкноти є унікальним. Особливості накладання матеріалів та їх співвідношення є пропрієтарною інформацією та не розповсюджуються. Таким чином, уявлення про те, яких значень має набувати сигнал при скануванні купюр, формується емпірично – зі спостережень.

Виходячи з фізичного змісту сканів, де точки представляють силу відгуку оптичного давача на фарби, та особливостей їх дослідження, необхідно дослідити існуючі та застосувати або запропонувати нові засоби для їх візуального огляду. Такі засоби мають забезпечувати користувача не тільки істинним представленням

банкноти на рівні сканів (власне монохромне зображення), але й надавати можливість у результаті порівняння з еталоном побачити спотворення та оцінити ступінь їх вираження.

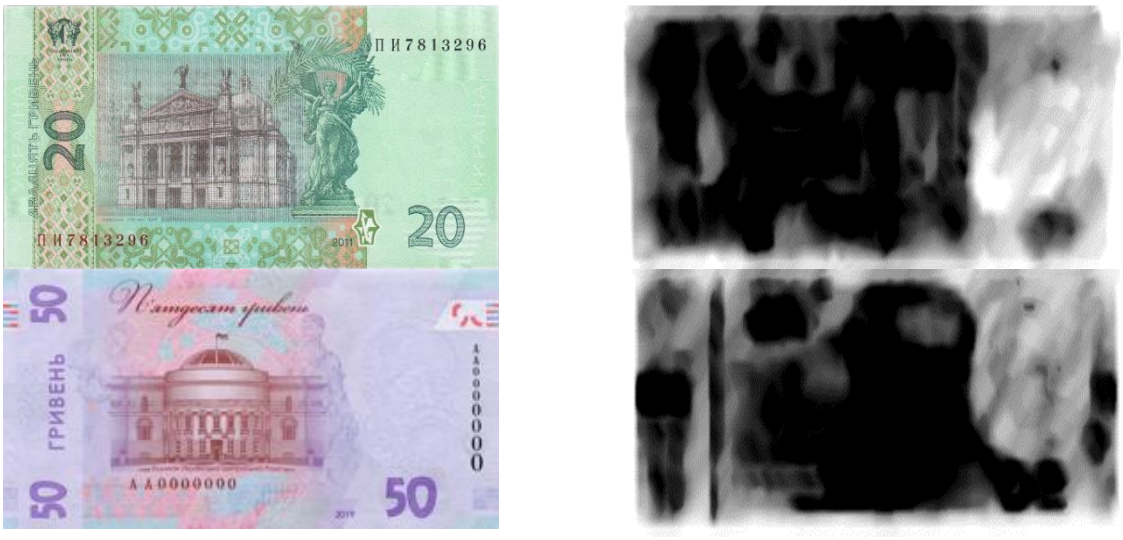
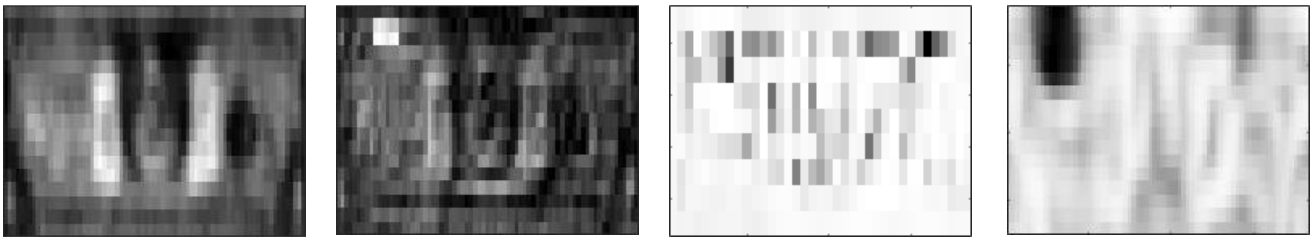


Рисунок 1.7 – Оптичні зображення банкнот номіналами 20 та 50 гривень виконані за допомогою офісного сканера та валідатора

На рисунку 1.8 показано результати порівняння двох сканів: оригінального без явних ушкоджень та його зміненого екземпляра із накладанням патерна прямокутної форми, який відповідає товстому скотчу в польових умовах. Зазвичай, оптичний сигнал змінюється в широкому інтервалі значень (наприклад, від 0 до 4000). За таких умовах відокремлення аномалії від набутих шумів складно виконати. Бачимо, що зображення абсолютної похибки може вводити в оману оператора, ховаючи зміни в сигналі серед шумів та природньої кривизни сканованої поверхні. Тому в цій роботі необхідно визначити методи візуальної оцінки, які надають кращу карту подібності зображень стійку до шумового забруднення.



а) нормальний скан б) спотворений скан в) абсолютна різниця г) бажане представлення

Рисунок 1.8 – Приклади графічного представлення сканів та їх подібності

Класична фотографія, яка виконується за допомогою світлочутливої матриці, є неподільною структурою. Напроти, в даній предметній області дискретне представлення банкнот отримується одночасним скануванням паперу набором сенсорів, кожним у фіксованому місці, поки купюра рухається пропускним каналом. Виявлення відхилень на одному треці свідчить про наявність у відповідній йому частині зображення помилки сканування чи самого введення. Цей процес є типовим для всіх сенсорів, тому аналіз даних цілого зображення можна спростити до обробки одного трека, що буде проводитися за однією методикою.

Отже, якщо запис складається із множини сканів, а скан – із набору треків, задачу автоматичного пошуку спотворень можна звести до виявлення аномалій в багатовимірному просторі. Необхідно дослідити існуючі методи та підходи та застосувати чи запропонувати такий, що найкраще відповідає вхідним даним та здатен кількісно оцінювати спотворення.

Користувача представляють працівники без спеціальної математичної освіти. Упровадження кожної підсистеми до виробничого процесу вимагає якісного методичного забезпечення та навчань. Часто параметри алгоритмічного забезпечення складно пояснити через наведення відповідників з реального життя. Таким чином, додатковим обмеженням для шуканого методу є мінімізація параметрів, які необхідно налаштовувати.

Похідною від створення алгоритмічного забезпечення є забезпечення її програмної реалізації. Необхідно мінімізувати ускладнення системи, та її вплив на характеристики інтегрованої системи розробки такі, як час компіляції та фінальний розмір застосування.

Отже, в результаті узагальнення предметної області та особливостей виявленої проблеми, необхідно виконати наступні задачі:

а) у заданому класі записів у наявній вибірці сканів обрати підмножину екземплярів даних, з яких сформувати представлення нормальної банкноти в заданому спектрі;

б) обрати засоби візуальної перевірки зображень сканів, які ефективно відображатимуть нормальну модель та окремі скани на її фоні, у порівнянні;

в) визначити критерій (або критерії) нормальності сканів у порівнянні з нормальною моделлю і на його основі обрати метод оцінки рівня аномальності даних;

г) спроєктувати методику взаємодії користувача з системою для виявлення аномалій обраним методом;

д) проаналізувати сучасні засоби реалізації математичних методів та запропонувати оптимальний спосіб імплементації алгоритмів та візуальної оцінки на їх основі.

1.4 Аналіз існуючих методів виявлення аномалій

У контексті аналізу даних аномаліями називають певні точки або групи точок, які сильно відрізняються від загальної вибірки. За Гокінсом, це результат спостереження відмінний від інших на стільки, що викликає підозри стосовно свого походження [5]. Барнет і Льюїс визначають невідомі раніше спостереження, викиди, як такі, що помітно відхиляються від інших членів множини, якій належать [6]. Аналогічно, Джонсон надає їм значення таких елементів деякої вибірки, у порівнянні з іншою частиною якої вони видаються несумісними [7].

В умовах, коли розглядається декілька груп даних об’єднаних в одну множину, Аґгарвал зазначає, що викиди можуть сприйматися як точки шуму поза набором визначених кластерів або є такими, що лежать за його межами, але відокремлені від шуму [8]. Додатково, під анормальністю можуть розуміти стан спостережень, за якого точка, що належить одному класу, незвично схожа на інший і тому лежить у межах чужого кластера, хоч і далеко від свого.

Таким чином бачимо, що в літературі, поняття «аномалія» має багато визначень, залежить від постановки задачі та структури даних характеристик досліджуваного об’єкта. У даній роботі використовуємо перше визначення,

відповідно до якого аномаліями вважаємо спостереження, які сильно відрізняються від інших – достатньо, аби поставити під сумнів їх відповідність обраному класу.

Наявність симетрії, розподіл, особливості варіації та нормальність визначають, яким чином буде реалізовано виявлення викидів та як вони мають інтерпретуватися. Спостереження можуть здаватися нормальними чи дійсно належати до таких, бути наслідком дії зовнішніх шумів, вплив яких виявляється в розхитуванні значень змінних, чи відповідати істинним відхиленням.

Причинами виникнення аномалій є несправності процесів, що їх породжують. До них належать збурення, механічні та інструментальні помилки, апаратні та програмні несправності, людські помилки, зміни в роботі системи або середовища її використання чи експлуатації. Згенеровані в результаті несправностей досліджуваного процесу викиди містять корисну інформацію про його анормальні характеристики та сутності, що вплинули на їх появу [9]. А виявлення таких особливостей серед спостережень може слугувати джерелом для вдосконалень і підвищення надійності системи. Напроти, збереження викидів у даних, наприклад, тренувальних при застосуванні алгоритмів машинного навчання, призведе до похибки, хибних висновків та пов’язаних із ними матеріальних втрат.

В літературі, пошук підозрілих даних називають виявленням викидів та аномалій, пошуком новизни та виключень, ідентифікацією подій, фільтрацією шумів та відхилень від нормальних значень.

Надалі, вважаємо виявлення аномалій таким, що відповідає знаходженню пошкоджених, аномальних, сканів у заданому наборі в межах одного класу.

Відповідно до сформульованого раніше визначення, скан є сукупністю треків, які представляють дискретизований сигнал певної довжини. Таким чином, його можна зобразити як спостереження точкою в n-вимірному просторі. Потужність простору ознак одного треку, зазвичай, варіюється між 50 та 500 факторами. Для знаходження викидів у заданих умовах необхідно обрати методи, які застосовні до них та ефективно долають прокляття вимірності (the curse of dimensionality). Наведена проблема характерна методам виявлення аномалій та

призводить до зниження ефективності їх роботи через зменшення чутливості класифікатора до віддалених ознак заданого простору.

Виявлення викидних сканів можна звести до задачі бінарної класифікації, де цільовими групами вважаємо нормальні оригінальні дані та аномальні спотворені. Серед імовірних сценаріїв вирішення даної задачі виділимо три залежно від того, чи тренувальні дані попередньо класифіковано, чи ні:

– у контрольованому режимі обчислення параметрів математичної моделі даних відбувається на повністю розміченій вибірці;

– напівконтрольований спосіб моделює модель даних використовуючи тільки нормальні екземпляри;

– у неконтрольованому режимі відсутні припущення щодо розподілу даних, та алгоритм одразу працює з тестовими даними й оминає етап тренування.

Задача виявлення аномалій існує лише в напівконтрольованому та неконтрольованому режимах.

Кількість пошкоджених даних у валютних вибірках є непередбачуваною величиною і може коливатися від повної відсутності помилок до їх суттєвої частки чи навіть переважання в певній базі даних. Таким чином, у даній роботі маємо розглянути методи, які дозволять попередньо обчислити модель даних та ефективно використовувати її для виявлення аномальних сканів банкнот.

Серед алгоритмів машинного навчання виділимо набір методів, які дозволяють описати задану вибірку та визначити коефіцієнт подібності чи викидності певного скана порівняно з нею. В широкому узагальненні їх складають техніки засновані на:

– щільності;
– відстані;
– кластеризації, поділі на групи подібних об’єктів;
– класифікації;
– статистичному розподілі;
– спектральні.

– відновлення.

Методи засновані на щільності базуються на припущенні, що всі точки заданого простору ознак можна поділити на групи на основі їх досяжності з інших точок. Досяжність екземплярів даних визначається відстанню між ними та кількістю точок в заданому околі. Кожен елемент множини спостережень може репрезентувати себе як:

– опорна точка, кількість сусідів якої в заданому околі перевищує мінімально допустиме значення – вагу;

– межева точка, досяжна з опорної через малу відстань між нею або іншою точкою групи;

– викидна точка, яка не належить жодній групі.

Таким чином, аномаліям характерно лежати в розріджених регіонах, а нормальним даним в щільних.

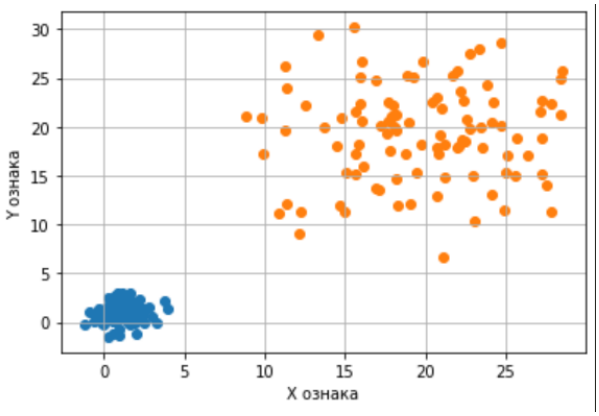


Рисунок 1.9 – Виявлення аномалій на основі щільності

Методи засновані на щільності визначають міру аномальності екземплярів даних через порівняння щільність ε-околу кожної з них. Звідси випливає їх висока обчислювальна складність у порівнянні з попередньою групою технік. Обчислення рівня подібності одної точки потребує аналізу її сусідів.

Головною перевагою щільнісних класифікаторів є здатність працювати незалежно від наявності припущення щодо розподілу вибірки. Однак, для тестування окремих спостережень вимагає від реалізації алгоритму збереження всіх даних, які складають нормальну модель.

Метод обчислення локального рівня викидності, запропонований Бройнігом та спрощений Шубертом та Зімеком [10] є однією з найпопулярніших метрик аномальності заснованих на групуванні даних за щільністю. Критерій полягає у визначенні ступеня викидності спостереження через порівняння його кластеру з найближчими сусідами.

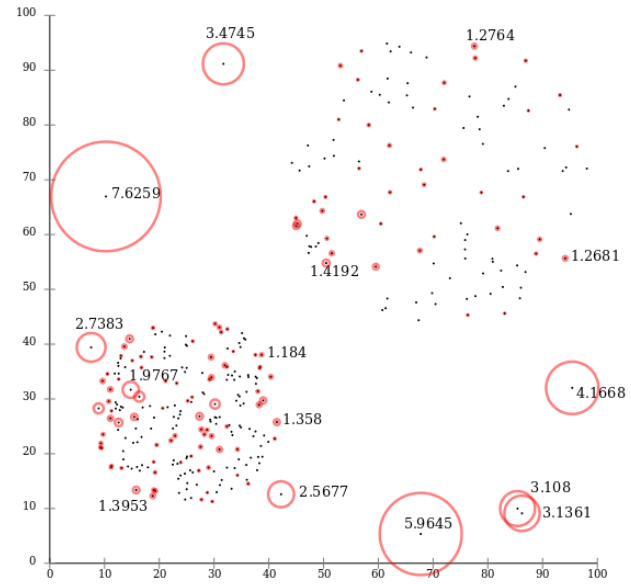


Рисунок 1.10 – Обчислення локального рівня аномальності

Методи обчислення рівня викидності на основі зв’язності [11] та ймовірності [12] також засновані на принципі локальної подібності спостереження заданому кластеру. Однак, перший оцінює досяжність через кількість послідовних межевих точок, а другий як імовірність точки належати кластеру, як тим менше, чим менше його щільність.

Серед недоліків щільності, як критерію нормальності, є чутливість алгоритмів до розміру та форми околів. Підбор цих гіперпараметрів додає складності їх реалізації та збільшує обчислювальну складність методу в цілому. Аналіз такими алгоритмами багатовимірних даних є викликом для них. Вони також несуть ризики потенційних помилок при пошуку пошкоджених сканів, коли певна база даних містить множину аномалій, які виникли з однакових причин.

Методи засновані на відстані визначають схожість двох екземплярів на основі їх порівняння. Базовою реалізацією даного підходу є алгоритм найближчих сусідів KNN. Принцип його дії схожий на щільнісні техніки, однак, на відміну від

них, приналежність даної точки певному кластери є ствердною за умови близькості точки до k елементів цього кластеру.

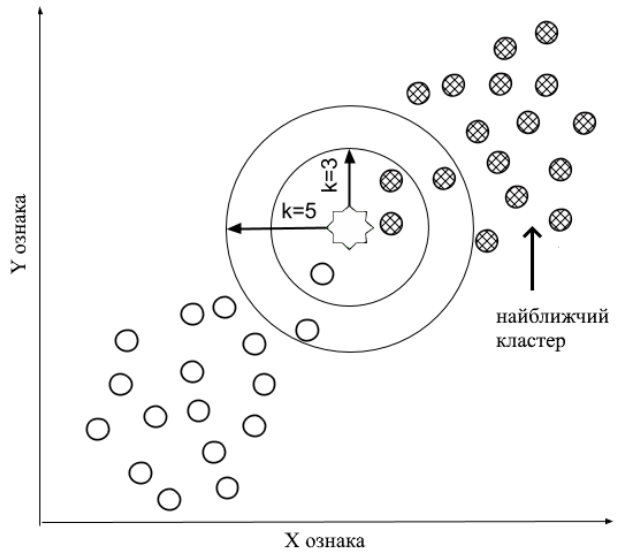


Рисунок 1.11 – Класифікація точки на основі найближчих сусідів

Методи засновані на відстані передбачають простіші обчислення в порівнянні з щільнісними техніками та є гнучкими на відміну від кластерних алгоритмів. Останні виконують класифікацію відносно центрів мас кластерів як k-середніх і моделюють кожну групу як опуклу сферичну область. Напроти, KNN дозволяє описувати розтягнуті вибірки та не чутливий до масштабування.

Аналогічно попередньому підходу ефективність KNN зменшується зі зростанням вимірності простору ознак. Це пов’язано з тим, що різні об’єкти мають відмінну варіацію в даному наборі даних, що ускладнює моделі вимірювання відстані між ними. Отже, даний підхід не задовільняє вимогам поставленої задачі.

Наступна техніка виявлення аномалій базується на поняттях кластерів та викидів. Кожна точка заданої вибірки належить одному з визначених кластерів і може бути або нормальною або аномальною. Мета кластеризації, такими чином, розділити екземпляри даних на щільні та розсіяні, де останні мають складатися переважно з викидних. Отже, в межах задачі кластеризації знаходження аномалій є похідним результатом її вирішення. Оцінкою відмінності точок одна від одної, в даному випадку, є відстань між заданою точкою та центром мас найближчого кластеру.

Виявлення аномалій на основі кластеризації залежить від правильності обрання кластерної структури нормальної моделі. Кластерні алгоритми є неконтрольованими, та не потребують попереднього знання розподілу даних. Відомими алгоритмами виявлення аномалій на основі кластеризації є k-середніх та DBSCAN.

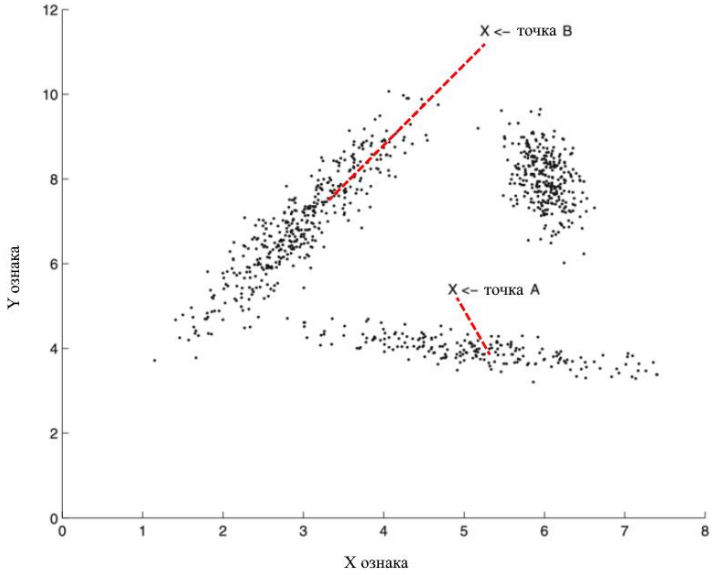


Рисунок 1.12 – Обчислення відстань від точки до центра кластера

Робота методу k-середніх базується на випадковому поділі простору ознак на k груп та ітеративному оновленні їх позицій, що представляє задачу мінімізації середньо-квадратичного відхилення елементів кожного кластеру відносно його центру.

Елементи кожного кластеру потенційно можуть формувати будь-яку форму в просторі ознак. Тому, математично, для обчислення відстані залучають метод Махаланобіса, який адаптується під складну форму кожного кластеру та повертає надійну оцінку подібності (рисунок 1.12).

Серед переваг кластерних методів можна виділити:

– здатність працювати в неконтрольований спосіб та адаптуватися й працювати із даними різних типів за рахунок використання правильних кластерних алгоритмів (які здатні обробляти такі дані);

– кількість та розміри кластерів є малою і фіксованою величиною, яка задається на етапі навчання, що спрощує та пришвидшує етап тестування.

З недоліків даної техніки неможливо не згадати:

– залежність від здатності кластерного алгоритму змоделювати розподіл нормальних даних;

– переважна більшість методів дозволяють виявляти аномалії як розріджені спостереження, і тому не підходять для поставленої задачі, бо зі зростанням обсягів даних, може збільшуватися їх відносна щільність;

– кластерний підхід є ефективним за умови дистантності (відмінності) викидів від більшості вибірки та уникнення ними інших аномалій згідно з вимогами до щільності в кластерах.

Основним недоліком алгоритму на основі кластеризації є те, що викидам призначається не оцінка, а двійкове маркування, де оцінка представляє ступінь викидності спостереження. Підрахунок балів необхідний, оскільки він допомагає відстежувати дії моделі, тому дії моделі стають остаточними і не можуть бути скасовані. Оголошення найкращої кількості кластерів на початковому етапі є складною роботою, і більшість алгоритмів кластеризації часто стикаються з цими труднощами. Крім того, якщо форма кластера довільна, алгоритми стикаються з проблемами точних розуміння кластерів із заданого набору даних. Таким чином, для хорошої роботи необхідно спочатку визначити форми кількох класів, або принаймні надати форму та розподіл кількох кластерів, що є складним завданням. Методи на основі розподілу дуже чутливі до ініціалізації параметрів, як-от алгоритми, які базуються на щільності. Тим не менш, вони недостатні для опису кластерів і в більшості випадків не підходять для дуже великих наборів даних.

Робота методів, заснованих на класифікації, здебільшого покладається на показник довіри, який розраховується класифікатором під час прогнозування для тестового спостереження. Якщо оцінка недостатньо висока, спостереженню не присвоюється жодна позначка – і воно вважається викидним. Обчислення показника довіри в більшості алгоритмів виконується за рахунок виділенні в n-вимірному просторі підпросторів ознак та обчисленні ймовірності належати

якомусь із них шляхом проєктування тестових даних на кожен простір та побудові інтегрального критерію.

Одним з поширених класифікаторів є однокласовий метод опорних векторів (One-Class SVM). Аналогічно до базового підходу, даний алгоритм намагається побудувати гіперплощину, яка б лінійно розділяла нормальні дані від аномальних із максимально можливою відстанню між ними та межею. Іншим підходом є ядрова оцінка густини розподілу (Kernel Density Estimation), яка зводить складну модель до простішої за рахунок згортки її щільності розподілу до функції нижчого порядку.

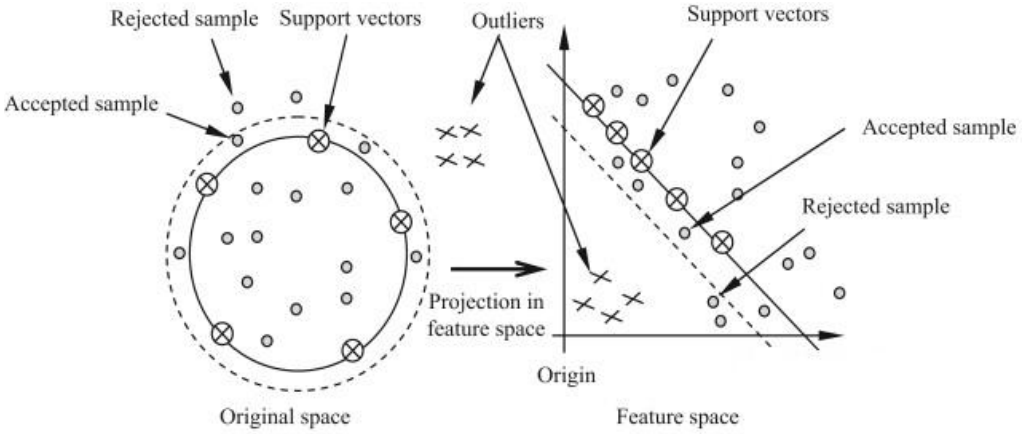


Рисунок 1.13 – Класифікація методом опорних векторів

Класифікаційним методам характерна висока ефективність у маловимірних просторах. У застосуванні вони передбачають стадію навчання для обчислення моделі даних та можуть працювати достатньо швидко при тестуванні окремих точок через відсутність необхідності до повторного обчислення моделі. Однак, їм також властиве явище прокляття розмірності (the curse of dimensionality), яке зменшує внесок окремих просторів ознак та робить їх невідмінними зі зростанням вимірності даних.

У [13] запропонували метод глибокої однокласової класифікації, у якій залучили згорткову нейронну мережу для формування просторів ознак. Даний метод протестували на великих масивах зображень MNIST та CIFAR, де він показав високі результаті в порівнянні з існуючими аналогами. Недоліком даного підходу

є вимога адаптації розмірності даних відповідно до архітектури нейронної мережі та потреба у великій кількості тренувальних даних для обчислення моделі.

Актуальний виробничий процес із розробки конфігурацій детекторів банкнот не передбачає накопичення обсягів даних достатніх для використання методів глибокого навчання. Скани мають високу вимірність, що матиме негативний вплив на ефективність виявлення аномалій класифікаційними методами.

Статистичні методи найпростіші з існуючих підходів до виявлення аномалій в даних. Вони засновані на припущенні відповідності певного розподілу вхідній вибірці. Таким чином, ступінь відмінності (протилежно, подібність) вхідного спостереження від стохастичної моделі визначається ймовірністю певного значення їй належати.

Статистичні методи можна поділити на параметричні та непараметричні. Перші базуються на знання про відповідність моделі даних певному розподілу. Серед них виділимо дві групи:

– суміші гаусіан;
– регресійні моделі.

Модель сумішей Гауса описує дані, яким відповідає композиція нормальних розподілів, чи гаусіан. Вона залежить від двох гіперпараметрів: дисперсії та середнього (центральної тенденції) – які визначають її форму та положення у просторі ознак. Для її означення використовують оцінку використовують метод найбільшої вірогідності. Він обчислює параметри розподілу Гауса.

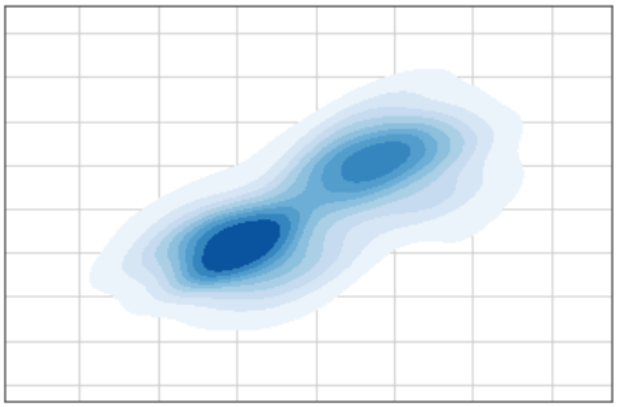


Рисунок 1.14 – Модель сумішей Гауса описує вибірку, згенеровану кількома нормальними розподілами

Коректність отриманої моделі перевіряють статистичними гіпотезами, та візуальними методами – коробковим графіком.

Сі Ян запропонували модель сумішей Гауса на нерозміченій вибірці [14]. Пошук, заснований на максимізації очікування, намагається вписати тренувальні дані в модель, підбираючи її параметри. Коефіцієнт викидів для цього методу розраховується як сума пропорційної зваженої суміші з ваговими коефіцієнтами, які відображають характер приналежності тестованих точок до інших. Далі він відображає ступінь викидності екземпляра даних.

Звичайний метод сумішей Гауса має кубічну складність, але ефективно виявляє аномальні спостереження навіть у дуже забруднених зовнішніми шумами даних. З метою зменшення обчислювальної складності було запропоновано навчання в підпросторах [15], яке передбачає розбиття вихідного простору ознак на підмножини та проєктування цих множин у простори меншої вимірності, що забезпечує збереження дисперсійних та кластерних особливостей вибірки. Іншою спробою до покращення є використання аналізу головних компонент [16].

До статистичних методів також належать регресійні моделі. Вони базується на побудові лінійної чи нелінійної функції, яка визначає місце екземплярів даних у просторі ознак моделі та обчислює фактор викидності кожної точки як відстань між точкою та її спрогнозованим значенням. Такі моделі є чутливими до викидів на стадії тренування, але виражають високу ефективність при тестуванні, якщо обчислення параметрів моделі виконано без їх участі. У [17] для опису визначення розмаху даних запропонували використання матриці кореляцій – її необхідно будувати лише раз на стадії моделювання – що спрощує обчислювальну складні алгоритму в цілому.

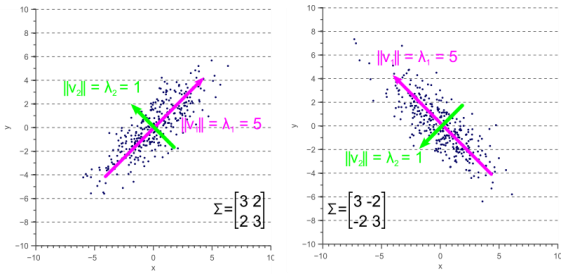


Рисунок 1.15 – Геометричний сенс матриці созалежностей

Інші статистичні методи включають усічене середнє стійке до викидів за рахунок відкидання з розрахунку центру розподілу точок, яким відповідають крайні області функції щільності імовірності; коробковий графік ділить вибірку на три регіони на основі квантилів і виявляє аномалії на основі теореми про три сігми та зводить вибірку до неї базуючись на граничній теоремі. Візуалізації викидів на основі інтервального розподілу (гістограми) можна використовувати як в одновимірних так і багатовимірних даних – у випадку останніх застосовують середньозважене відхилення для оцінки аномальності.

Фундаментальні базові математика, що лежить в основі статистичних алгоритмів OD, робить їх простими у використанні:

– забезпечують статистично обґрунтоване рішення за умови вірності припущення щодо розподілу даних;

– визначають міру відхилення, як відстань від центру довірчого інтервалу на основі обраного розподілу;

– за умови стійкості методу оцінки розподілу до впливових точок, які призводять до змін математичного очікування вибірки, її центру, можуть працювати з неконтрольованими даними.

Завдяки своїй компактній формі, моделі демонструють покращені характеристики з точки зору швидкості виявлення та часу роботи для конкретної ймовірнісної техніки.

Головним недоліком стохастичних моделей є припущення щодо базового розподілу щільності, що призводить до низької продуктивності і часто – до ненадійних результатів у кінцевих застосуваннях. Підхід, заснований на статистиці, застосовний переважно для одновимірних наборів даних; отже, вони погано працюють для багатоваріантних просторів функцій, таких як скани банкнот. При застосуванні моделей до багатоваріантного простору ознак виникають високі обчислювальні витрати. Крім того, гістограма не може відобразити взаємодію між ознаками, що також робить її поганим вибором для даних великої розмірності. Однак, у разі доведеності гіпотези про розподіл досліджуваної величини

статистичні методи забезпечують надійний критерій аномальності. Для подолання прокляття високої розмірності можна застосувати специфічні статистичні методи, але необхідно вирішити проблему зі збільшенням часу обробки даних та складеності розподілу даних, які можуть бути породжені різними процесами або їх комбінацією.

Простір ознак результатів сканування є багатовимірною множиною, якій характерні всі недоліки успадковані від прокляття вимірності, неоднозначності розподілу індивідуальних вимірів та їх потенційної созалежності. Для подолання означених вад можна використовувати методи попередньої обробки даних засновані на зменшенні вимірності.

Ряд регресійних та непараметричних статистичних алгоритмів дозволяють оцінювати ступінь аномальності даних через порівняння їх з базовим простором шляхом відновлення.

Метод аналізу головних компонент перетворює багатовимірний простір на простір меншої вимірності шляхом ортогональних проєкцій на його вісі. Принципи роботи метода висновуються з вимоги до задачі оптимізації складати виміри таким чином, аби утворювана дисперсія складеного виміру зберігала максимум коливань базових вимірів. Таким чином, якщо виявлення аномалій ґрунтується ймовірності, можна перейти з важкого простору в легший. Тоді детектування тестових даних, яким характерна висока дисперсія лише на певних вимірах, буде ефективною.

Іншою технікою зменшення вимірності даних є метод збереження локальних проєкцій. На відміну від попереднього, йому не характерне збереження варіації даних при згортанні простору ознак. Напроти, він дозволяє логічно зберегти геометричну залежність одних спостережень відносно інших та візуально аналізувати їх у межах репрезентативних вимірів: одно-, дво- і тривимірному просторі.

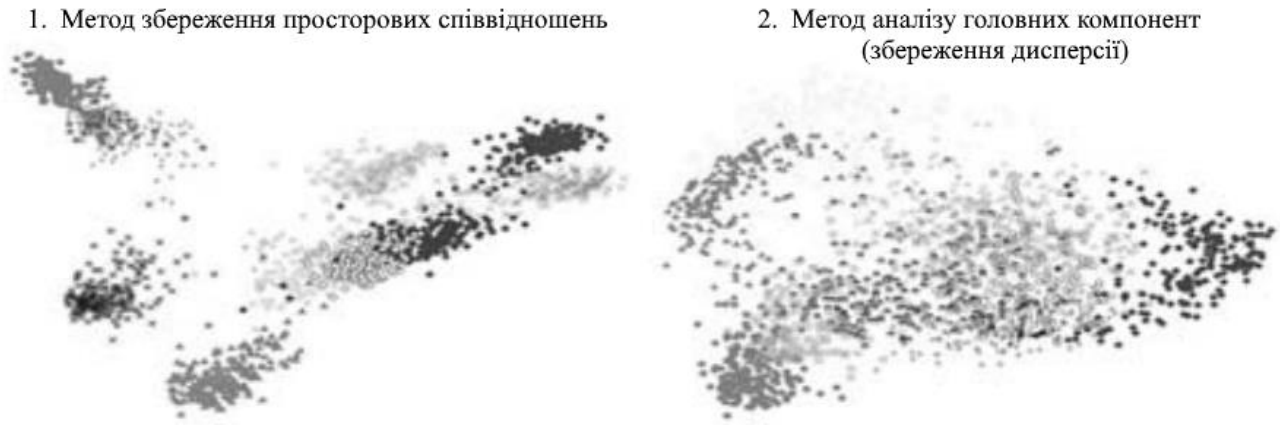


Рисунок 1.16 – Результати згортання багатовимірних просторів у двовимірний [18]

З результатів тестування наведених методів зображених на рисунку 1.16 бачимо підтвердження їх зазначених особливостей стосовно збереження властивостей вихідного простору ознак при переході до двох вимірів.

Однією з задач виявлення аномалій, як уже було не одноразово визначено, є побудова базової моделі, яка відображає нормальний портрет вибірки і виступає в якості математичного очікування досліджуваної багатовимірної величини. Таким чином, не зважаючи на прокляття вимірності та явне ускладнення процедури виявлення аномалій у разі згортки багатовимірного простору з метою застосування на ньому статистичних законів, техніки можна використовувати для візуальної верифікації даних і виділенні на їх графічному представленні очевидних кластерів як тренувальної вибірки позбавленої викидів у своєму складі.

В [18] визначають метод збереження просторових співвідношень як більш ефективний за аналіз головних компонент. Однак, алгоритм виконання останнього може надати інструмент додаткової перевірки статистичної гіпотези при виявленні аномалій. Його робота базується на сингулярному розкладі матриці, а саме на сингулярних векторах та власних значеннях, які він повертає, що відповідають за основний напрям варіації змінних та її силу. Процедура згортки простору (рисунок 1.17), яку надає метод, є зворотною. Таким чином, маючи результати розкладу можна провести відновлення даних у вихідних стан, що подібне роботі

регресійного аналізу. Відтворення даних з певної моделі дозволяє порахувати помилку цієї операції, яку можна використати за рівень аномальності і виділити викиди з даних.

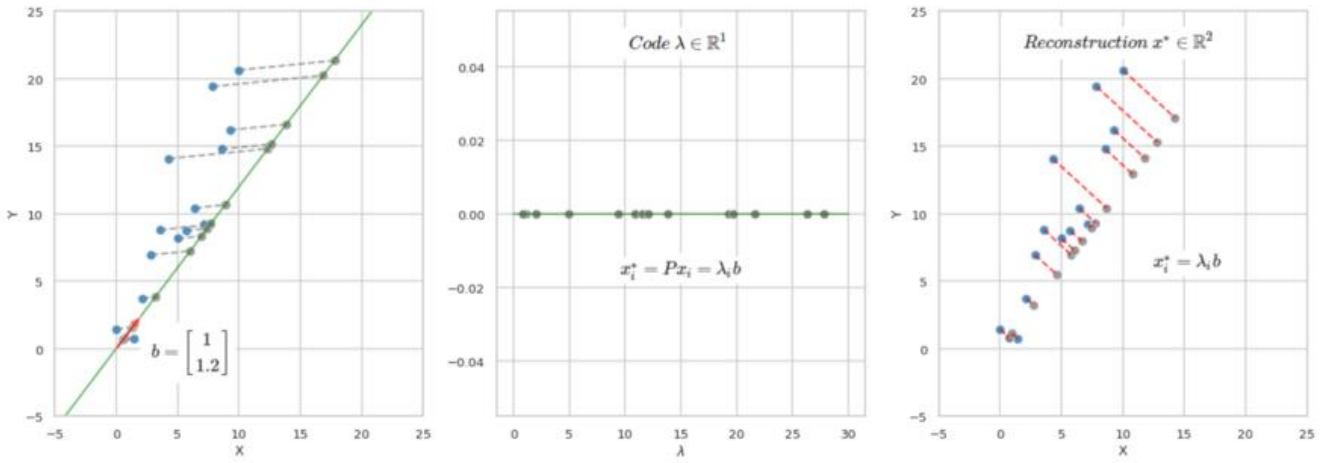


Рисунок 1.17 – Згортка та відновлення даних між вимірами методом головних компонент

При скануванні екземплярів банкнот для кожної формується набір зі сканів тих сенсорів, які доступні в даній моделі пристрою. Таким чином кожен запис є відображенням купюри у множині «кольорів» – сигналів різної довжини хвилі зведених до одного формату значень.

Визначення пошкоджених даних зводиться до виявлення аномалій у будь-якому з наведених сканів. Враховуючи структуру сканів, достатньо підтвердити наявність спотворень в одному треці, аби класифікувати весь запис як аномальний та виключити з тренувальної вибірки пристрою.

З аналізу існуючих методів виявлення аномалій: базових та нових, які були запропоновані в останні роки – бачимо, що існує багато технік, застосовних до поставленої задачі. Однак, кожен підхід окремо містить як сильні, так і слабкі сторони, що перешкоджають його використанню як є в заданих умовах. З переліку доступного алгоритмічного забезпечення, найбільше підходять статистичні методи, засновані на припущенні щодо відповідності досліджуваних даних певній стохастичної моделі, та спектральні, що базуються на зменшенні вимірності даних. З постановки задачі маємо припущення, щодо приналежності значень кожної точки сканованого треку своєму варіаційному інтервалу. Так як кожен піксель має одне

джерело походження, отримали статистичну гіпотезу для перевірки та набір методів, комбінацію чи модифікацію яких необхідно визначити для успішного виявлення аномалій в сканах банкнот.

1.5 Аналіз існуючого програмного забезпечення для виявлення аномалій

Найпоширенішою практикою з реалізації програмних засобів виявлення аномалій у багатовимірних даних є написання окремих алгоритмів як підсистеми в існуючому продукті без створення бібліотек або використання вже існуючих як готових рішень чи інструментів для їх отримання. Таким чином, задача зводиться до обрання найбільш релевантного методу пошуку викидів та його виклику через публічний інтерфейс.

Застосунки для аналізу вибірок на предмет наявності дивних екземплярів, зазвичай, передбачають інтерфейс для вибору даних: тренувальних та тестувальних – застосування на них обчислювального алгоритму та відображення результатів обробки окремим графічним компонентом. Ефективність такого інструмента визначається зручністю його інтерфейсу для введення даних у систему, релевантністю до заданої проблеми способів їх представлення, швидкістю виконання програми.

Окрім додатків з графічним інтерфейсом, окремо також розробляються програмні засоби, де кінцевими користувачами є розробники. Такі продукти є програмними інтерфейсами, що орієнтуються на основні платформи для роботи з даними та їх аналізу такі, як Python, R, MATLAB, Octave.

«AVORA» належить до категорії онлайн веб-застосунків. Це платформа нового покоління для накопичення та обробки потоків даних, призначена для відслідковування поведінки актуальних бізнес-подій та швидкого сповіщення користувачів про зміни в трендах [19]. Інструмент орієнтований здебільшого на аналіз даних методами часових рядів. Він надає розробнику відкритий API-інтерфейс та очікує ззовні надходження пакетів даних. На основі історії оновлень

вхідних значень передбачається формувати статистичну модель, яка надалі виступатиме як класифікатор при виявленні аномалій.

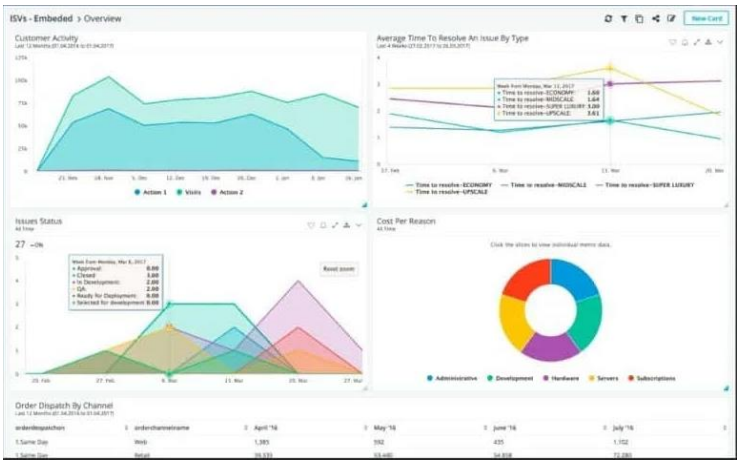


Рисунок 1.18 – Аналітична веб-платформа «AVORA»

Серед переваг можна виділити: широкий набір засобів для візуалізації (рисунок 1.18), можливість інтеграції інших застосувань як джерел спостережень, підтримка експорту результатів у соціальні платформи такі, як «Facebook», «LinkedIn» тощо та надсилання сповіщень до «Slack» і «Teams», інтеграція зі хмарними сховищами даних, і кросплатформеність.

Серед аналогічних систем поширеним також є «Anodot» – моніторингова платформа для аналізу даних визначених у кожен момент часу. Схожа до попередньої системи щодо здатності працювати з великими обсягами потокових даних. Підтримує масштабування як SaaS платформа і забезпечує єдиний інтерфейс для всіх зацікавлених сторін: розробників, аналітиків, тестувальників тощо [20]. На рисунку 1.19 зображено інтерфейс системи.

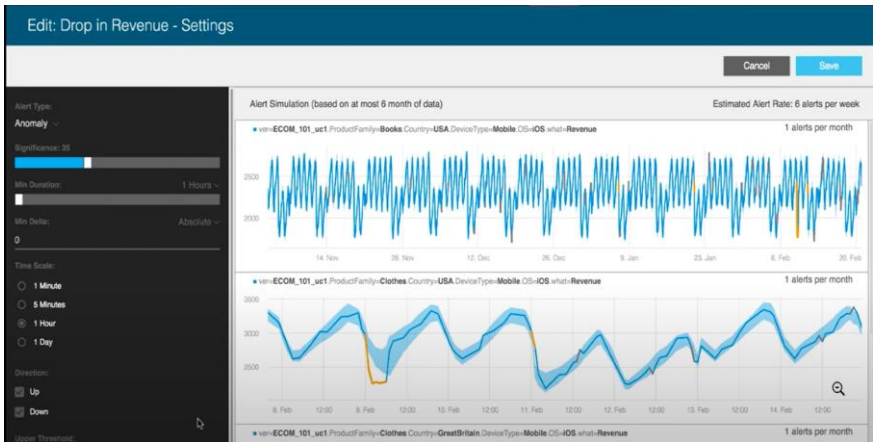


Рисунок 1.19 – Платформа з виявлення аномалій в часових рядах «Anodot»

Сильною стороною «Anodot» є автоматизація процесу виділення просторів ознак, у яких виникають аномальні спостереження. Здатність до виявлення як новизни, так і викидів у вхідних даних часових рядів та перетворення їх на бізнес-користь спрощує інтерфейс користувача та скорочує час, що передує отриманню перших результатів. Система виявляє та ізолює прояви аномальної поведінки, та проводить їх кореляційний аналіз з іншими досліджуваними показниками для швидкого прийняття рішень стейкхолдерами.

У наведених платформах чітко простежується провідна спеціалізація – часові ряди. Їх спільним недоліком є зосередженість на одному типі даних та головне: відсутність підтримки просторових даних та інструментів з їх аналізу. Аналіз існуючих реалізацій показав відсутність таких користувацьких систем на ринку, однак виявив ряд інструментів, що використовуються для досліджень та потенційно можуть застосовуватися для створення автоматичних експертних рішень.

Проєкт «scikit-learn» представляє бібліотеку машинного навчання з відкритим вихідним кодом для мови програмування Python. Віна містить ефективні та добре налагоджені інструменти в середовищі програмування, доступному для експертів з інших галузей та багаторазового використання в різних наукових областях. Проєкт не є новою предметною мовою, а реалізує бібліотеку, яка надає ідіоми машинного навчання для мови високого рівня. Серед іншого, він надає класичні алгоритми аналізу, інструменти оцінки статистичних характеристик та вибору стохастичних моделей, а також процедури попередньої обробки даних. Поширення та використання бібліотеки можливе за спрощеною ліцензією BSD, що заохочує її до використання з академічних та комерційних цілей [21].

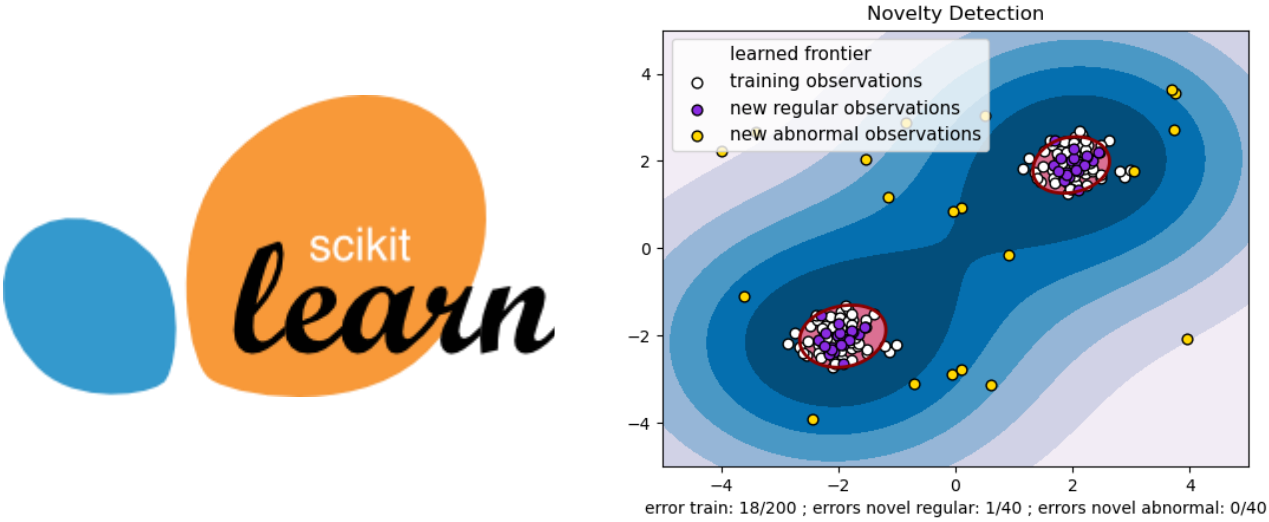


Рисунок 1.20 – Інструментарій з побудови та аналізу алгоритмів машинного навчання scikit-learn

«scikit-learn» також містить алгоритми виявлення аномалій та надає зручний інтерфейс для їх гнучкого використання, графічного відображення результатів (рисунок 1.20) класифікації засобами бібліотек «matplotlib» та «plotly». Останні також можна вбудовувати в елементи користувацького інтерфейсу при побудові десктопних та веб-застосунків.

«PyOD» (Python Outlier Detection) – комплексний і масштабований засіб із виявлення аномалій та викидів на даних різного типу реалізований мовою Python із застосуванням інших відкритих бібліотек мови для швидкої роботи з масивами даних та їх візуалізації. Реалізує понад 20 методів та алгоритмів з ідентифікації аномалій, пошуку та виключення їх з вибірок, побудови класифікаторів у контрольований, напів- та неконтрольований спосіб [22].

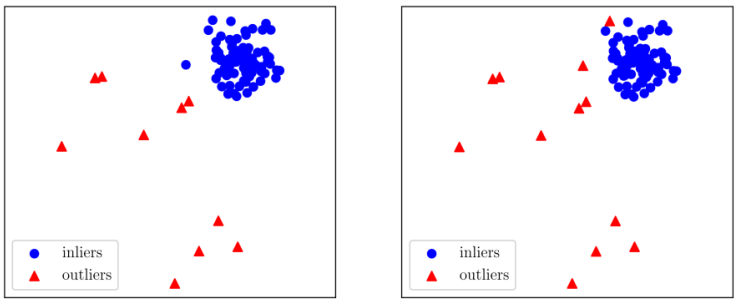


Рисунок 1.21 – Візуалізація виявлення аномалій за допомогою бібліотеки «PyOD» [22]

«PyOD» є однією з найцитованіших бібліотек машинного навчання, які спеціалізуються на виявленні аномалій. Перш ніж стати відкритою розроблялася як внутрішній проєкт Університету Торонто. Її головними перевагами є простий інтуїтивний та розширюваний інтерфейс, можливість інтеграції з іншими модулями мови Python – у тому числі з «scikit-learn» – та простота використання в комбінації з іншими аналітичними інструментами мови з метою побудови інтерактивних та ефективних інсайтів на основі аналізу даних. Оптимізація бібліотеки базується на залученні до реалізації обчислень низькорівневих бібліотек (написаних на Fortran і Cython) та інших модулів Python таких, як «Numpy», побудованих за даним принципом.

У якості аналогічного інструменту для MATLAB-розробників можна виділити проєкт «LIBRAstat», створений командою Католицького університету Левена [23]. Він представляє бібліотеку алгоритмів з простим та зручним інтерфейсом і може використовуватися для задач виявлення аномалій завдяки колекції методів стійких до викидів при побудові нормальних моделей даних: обчисленні відстаней, коваріаційних матриць, коефіцієнтів лінійної та нелінійної регресії, кластеризації.

Перевагами «LIBRAstat» є орієнтованість на середовище MATLAB, яке виступає широко вживаним, потужним та функціональним індустріальним інструментом ведення досліджень. Він надає користувачу засоби з візуалізації, зручної роботи з даними та їх аналізу.

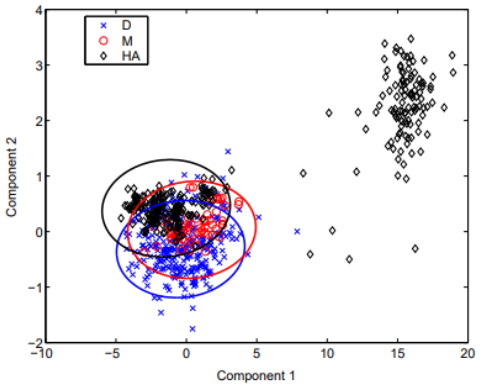


Рисунок 1.22 – Використання методу лінійного дискримінанта за допомогою «LIBRAstat» у середовищі MATLAB [23]

Основними недоліками наведених програмних рішень є їх спрямованість на забезпечення засобів з побудови інструментів для виявлення аномалій, замість того, аби бути таким інструментом для кінцевого користувача. Під останнім вважаємо інженера у галузі забезпечення якості та надійності грошових знаків без обов'язкової наявності базових навичок зі програмування.

Цільова система для впровадження програмного засобу з виявлення пошкоджених даних представлена десктопним застосунком. На відміну від наведених рішень, вона реалізована низькорівневою мовою і потребує алгоритмізації та її реалізації в заданому технологічному середовищі.

1.6 Вимоги до алгоритмічного та програмного забезпечення

Метою роботи є підвищення ефективності розпізнавання фальшивих банкнот за рахунок підвищення автоматизації процесу виявлення пошкоджених сканів.

Призначенням роботи є створення програмного та алгоритмічного забезпечення для виявлення пошкоджених даних при сканування паперових грошових знаків пристроями детектування банкнот.

Розроблене програмне забезпечення повинно відповідати наступним вимогам:

– можливість візуальної перевірки даних за допомогою графічних представлень;

– можливість згортки багатовимірних даних у дво- та одновимірні;

– можливість визначати тренувальну вибірку, на основі якої буде відбуватиметься навчання алгоритму;

– можливість автоматичного визначення тренувальної вибірки з вихідних даних;

– задовільна точність алгоритму виявлення аномалій програмним засобом;

– можливість налаштування автоматичного тестування великої кількості даних в межах треку, скана, класу, бази даних у цілому;

– можливість виключення даних з аналізу;

– правильність реалізації алгоритмічного забезпечення;
– швидкість виконання обчислень та роботи програми в цілому;
– інтуїтивно зрозумілий інтерфейс;
– простота у використанні;
– простота у налаштуванні.

Висновки до розділу

Детектори банкнот займають важливе місце у сфері забезпечення якості та валідації грошей. У процесі їх налаштування виконується обробка великих масивів даних, за результатами якої будують високоточні алгоритми ідентифікації, верифікації та класифікації банкнот на основі сканування сенсорами їх фізичних та структурних характеристик. Кожна конфігурація залежить від якості даних, використаних при її створенні.

У розділі проаналізовано процес сканування паперових грошей. Визначено базові поняття: сканів, треків, записів – елементів бази даних. Процес передбачає введення та обробку десятків тисяч банкнот і несе пов’язані з цим ризики: програмні, апаратні та людські помилки. Несправності призводять до спотворення даних і побудови неточних моделей для кінцевих пристроїв. Таким валідаторам характерна висока частка похибки в роботі, а значить – збитки для компанії-виробника.

У контексті банкнот було надано визначення аномалій: сканів, які відрізняються від інших на стільки, що це викликає підозри стосовно їх походження. Аналіз структури сканів показав, що вони є даними в багатовимірному просторі, кожен вимір якого має однакове походження в межах одного треку, а отже: йому відповідає один розподіл. Порівняно ефективність існуючих методів виявлення аномалій та можливість їх використання на сканах банкнот. Визначено, що статистичні методи є достатньо простими, аби не ускладнювати систему, та перспективним напрямом досліджень.

Проведено аналіз існуючих програмних засобів виявлення аномальних даних, а варіантів реалізації кінцевих застосунків: аналітичні веб-платформи «AVORA» та «Anodot», інструментальні бібліотеки «Scikit-learn», «PyOD» та «LIBRAstat», визначені їх переваги та недоліки. Зроблено висновок, щодо відсутності на даний момент програмного засобу для кінцевого користувача для виявлення аномалій на сканах банкнот та їх аналізу.

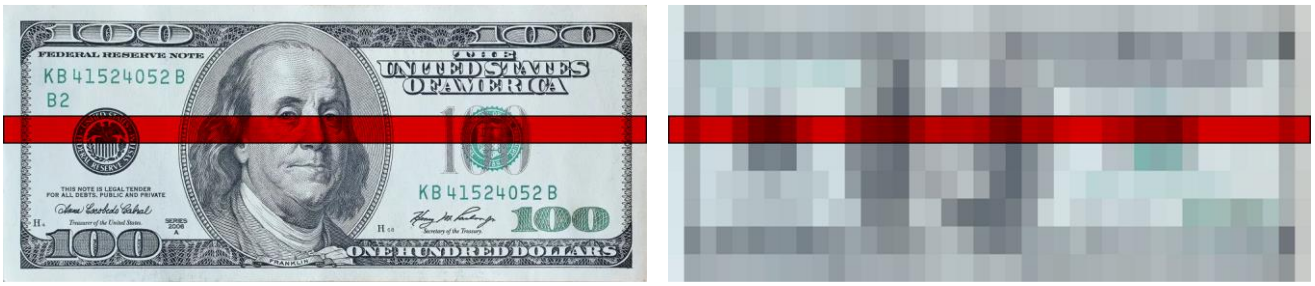
У результаті сформульована мета роботи, призначення та основні вимоги.

2 РОЗРОБЛЕННЯ МЕТОДУ ВИЯВЛЕННЯ АНОМАЛІЙ РЕЗУЛЬТАТІВ СКАНУВАННЯ БАНКНОТ

2.1 Визначення моделі оптичного сигналу

Для розв'язання задачі пошуку пошкоджених сканів, тобто виявлення аномалій на них, виконаємо формалізацію моделі нормального оптичного сигналу.

Результатом сканування банкноти є двовимірне зображення складене з незалежних частин – треків. Відповідно до предметної області, кожному треку відповідає певна область на папері, що сканувався пристроєм за допомогою одного сенсора.



а) б)

Рисунок 2.1 – Зображення банкноти реальне (а) та (б) скановане із виділенням треку

Нехай $X = [x_i, i = \overline{1,n}]$ – простір ознак результатів сканування банкноти одним давачем, де кожній змінній x_i відповідає і-та точка сигналу, з'єднана з сусідніми відрізками, а n – вимірність простору, задана довжиною треку. За область значень, які і-та змінна x_i приймає на всій множині правильних сканувань однієї купюри вважаємо функцію $y(x_i), i = \overline{1,n}$. Тоді $y_j(x_i), j = \overline{1,m}$ буде результатом одного неперервного сканування в точці x_i де m – це кількість різних реалізацій правильних банкнот, що складають множину спостережень.

В ідеальних умовах кожне сканування банкноти одним давачем втілюється в єдиній однаковій кривій, що є графічним представленням дискретного сигналу, як показано на рисунку 2.2. Тоді аномалією вважалася би будь-яка його реалізація, яка має абсолютне значення похибки вище за нульове.

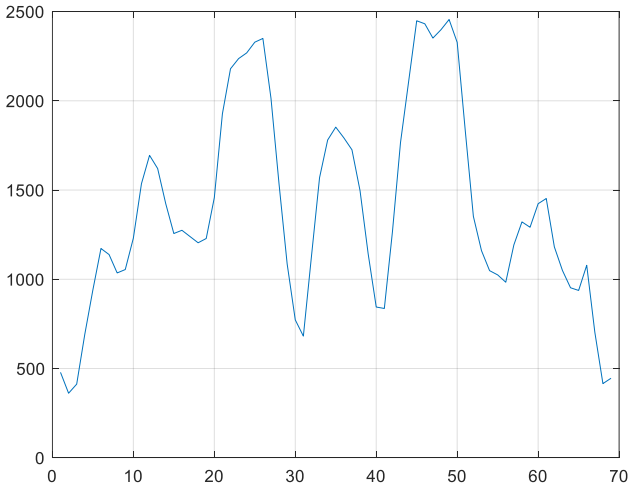


Рисунок 2.2 – Результат сканування банкноти оптичним сенсором

В умовах практичного застосування детекторів, процес сканування проходить під постійним впливом зовнішніх збурень: шумів, викликаних середовищем та недосконалістю компонентів апаратного засобу. Таким чином, відбуваються природні коливання сигналу – тобто варіації кожної точки треку – характерні для процесу сканування. Отже, маємо динамічне середовище виконання статичних дій.

Геометричним місцем точок окремого треку, сканованого під дією випадкових шумів, є його математичне очікування, обчислене на основі спостережень

$$\bar{y}(x_i) = \sum_{j=1}^{l} y_j(x_i) \qquad (2.1)$$

де $\bar{y}(x_i)$ – середнє значення реалізації сигналу в точці x_i;

l – загальна кількість правильних реалізацій сканування.

Характер зміни оптичного сигналу відносно його математичного очікування опишемо за допомогою коридору. Коридором вважаємо область значень задану для кожної точки результату сканування банкноти на множині її правильних реалізацій. Враховуючи спотворення викликані випадковими процесами формалізуємо коридор як сукупність границь – мінімумів і максимумів – функції

обчислену для кожної точки графіку в межах її області значень, що визначає межі його нормальних коливань

$$\begin{aligned} C_{i,min} &= \underset{j=\overline{1,l}}{min}\, y_j(x_i) \\ C_{i,max} &= \underset{j=\overline{1,l}}{max}\, y_j(x_i) \end{aligned} \tag{2.2}$$

де $C_{i,min}, C_{i,max}$ – нижня та верхня границі коридору для кожної точки x_i треку;

$i = \overline{1,n}$.

Приклад коридору представлений на рисунку 2.3 побудовано на основі вибірки зі 100 правильних сканувань однієї банкноти. У результаті накладання відповідних значень сигналу в кожній точці отримали його графік щільності.

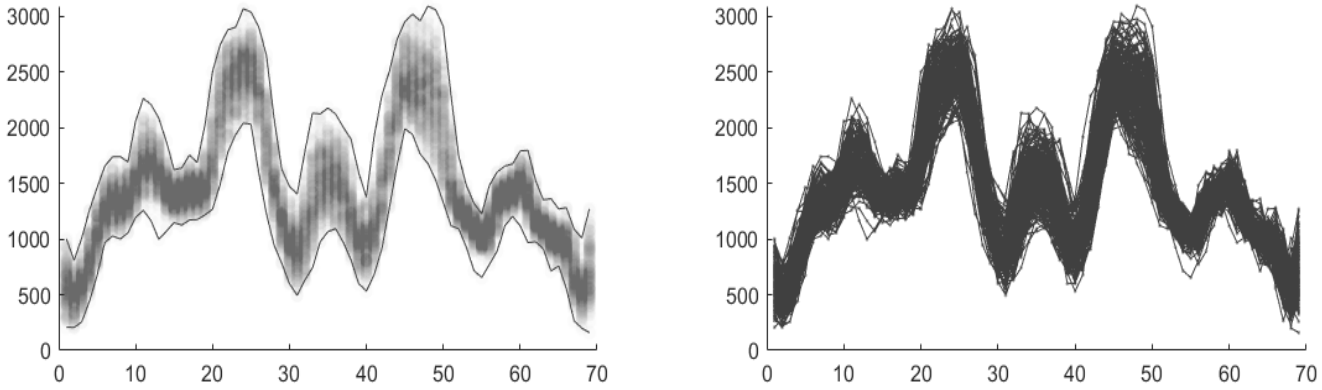


Рисунок 2.3 – Розподіл оптичного сигналу при ста ідентичних скануваннях

З рисунку бачимо, що характер варіацій сигналу в кожній точці є різним. Це можна пояснити особливостями використаних для друку матеріалів та їх комбінацій або якісним станом банкноти в момент введення у валідатор.

Для визначення дійсної форми сигналу його коридор має бути стійким до викидів. Обчислення варіаційних меж необхідно виконувати на основі стійких статистичних ознак – тобто відкидати явно аномальні значення. Для цього достатньо звузити ширину коридору до міжквантильного розмаху, який визначити на інтервалі

$$\begin{aligned} C_{i,min} &= y_{1-\alpha}(x_i) \\ C_{i,max} &= y_{1+\alpha}(x_i) \end{aligned} \tag{2.3}$$

де $y_{1\pm\alpha}(x_i)$ – значення змінної x_i, яке вище за $\alpha = 99.99\%$ її правильних реалізацій.

Алгоритм побудови стійкого коридору наступний:

а) отримання множини правильних реалізацій треку, які задовільнятимуть критеріям візуальної перевірки якості сканування, у кількості достатній, аби вважати коридор репрезентативним;

б) обчислення стійких границь довірчих інтервалів шляхом сортування значень треків у межах кожної точки (виміру) та обрання значень, порядковий номер яких відповідає α-квантілю;

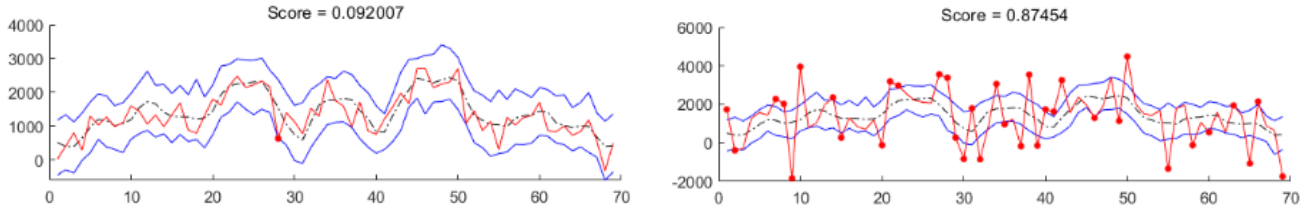
в) обчислення міжквантільного розмаху та математичного очікування оптичного сигналу на його основі.

Згідно з визначенням коридору оптичного сигналу, основною задачею шуканого методу виявлення аномалій є пошук таких реалізацій сканів банкнот, графік яких буде виходити за межі допустимої області. Додатковою задачею є отримання оцінки рівня відмінності окремої реалізації у порівнянні з відповідним їй еталоном – математичним очікуванням – так, аби виконувалися наступні обмеження:

– екземпляри даних, які лежать у середині коридору, і відповідають ознакам дійсної банкноти повинні отримати низький рівень аномальності та бути класифікованими як нормальні;

– екземпляри даних із явними проявами аномальності (з точками, що виходять за межі визначеного коридору) повинні мати значення оцінки викидності вище на стільки, аби їх можна було виділити з-поміж інших.

Приклади можливого рішення задачі показано на рисунку 2.4.



а) б)

Рисунок 2.4 – Представлення нормального (а) та (б) спотвореного сигналів з очікуваною оцінкою аномальності

2.2 Емпіричні критерії аналізу результатів сканування

Задля розв'язання задачі з виявлення аномалій та підтвердження чи спростування гіпотези про достовірність банкноти пропонується перевіряти її окремі реалізації за критеріями заснованими на емпіричних властивостях досліджуваних сигналів. Відповідно до математичної моделі сканування, до характеристик треку належать:

– μ_x, математичне очікування,
– C, варіаційний коридор,
– локальні екстремуми.

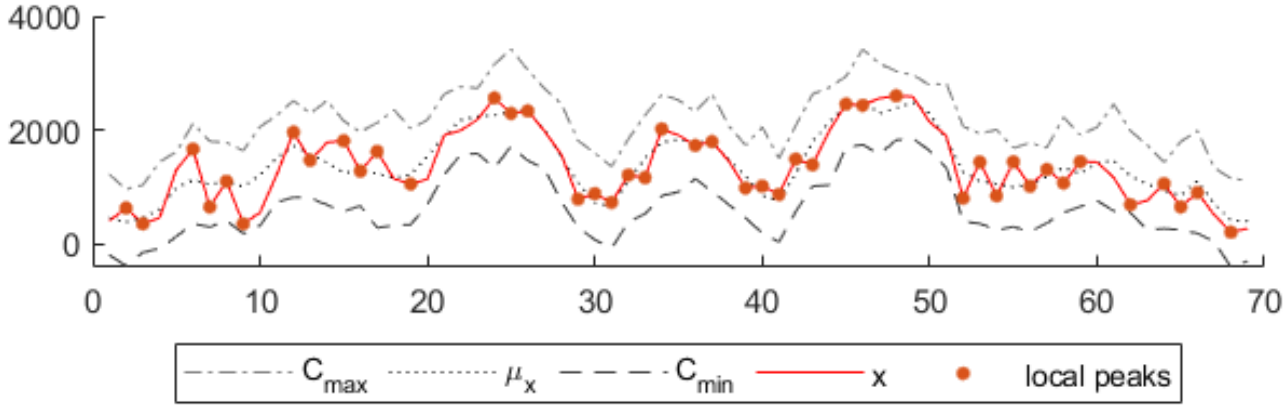


Рисунок 2.5 – Емпіричні характеристики треку

Перша властивість визначає рівень оптичного сигналу характерний даному пристрою – унікальне місце цифрового представлення банкноти у просторі ознак. Друга задає умови сканування та враховує природню відмінність між надрукованими екземплярами однієї купюри, що виявляється у спотворенні випадковими шумами та розхитуванні точок у межах своїх допустимих значень. Третя ознака описує форму сигналу, що відповідає відгуку сенсора на розташування, кількість та відношення фарби на досліджуваному папері, й описує дані, як ланцюжок переходів з одного стаціонарного положення в інше.

Нехай результат незалежного сканування можна однозначно визначити виразом $y = r(x_i)$, де y – значення сигналу в точці x_i, а r – його окрема реалізація.

Тоді даний трек не суперечить гіпотезі про дійсність спостережуваної банкноти, якщо виконується:

а) перший критерій задовільняє гіпотезі, якщо для кожного значення інтенсивності $r(x_i)$ сигналу справедливе співвідношення:

$$\forall i: C_{i,min} \leq r(x_i) \leq C_{i,max} \quad (2.4)$$

де $C_{i,min}$ та $C_{i,max}$ – мінімальне та максимальне допустиме значення сигналу в точці x_i, які він може приймати, отримані зі сканувань правильних банкнот за формулою (2.3).

Далі C називаємо коридором, а пару значень C_i – його межами в і-тій точці; під розмахом $C_{i,range}$ коридору в і-тій точці маємо на увазі абсолютну різницю цих меж

$$C_{i,range} = C_{i,max} - C_{i,min} \quad (2.5)$$

Середньоквадратичне відхилення – міра похибки, що часто використовується при визначенні рівня відмінності між двома багатовимірними змінними такими, як сигнали, зображення, числові послідовності. Однак, якщо застосовувати її в даній задачі при порівнянні двох результатів сканування отримаємо значення в масштабах різних для кожної унікальної реалізації банкноти. Враховуючи змінний характер середовища та наявність шумів, використання цієї міри у простій формі створить додаткові вимоги до підбору порогових значень критерію та ризики програмного переповнення, що призведе до грубих помилок класифікації. Отже, для використання як критерія міри відхилення від математичного очікування її необхідно представити у зміненій, нормованій, формі.

б) другий критерій не суперечить гіпотезі, якщо значення D_E середнього абсолютного нормованого за розмахом коридору в кожній точці відхилення реалізації сигналу, що перевіряється, задовільняє нерівності

$$D_E \leq D_{max}, \quad D_E = \frac{1}{n}\sum_{i=1}^{n} \frac{|r(x_i) - \bar{y}(x_i)|}{C_{i,range}} \quad (2.6)$$

де D_{max} – максимальне середнє абсолютне нормоване відхилення реалізації від його оціночного математичного очікування, обчислене на основі сканувань правильних банкнот

$$D_j = \frac{1}{n}\sum_{i=1}^{n}\frac{\left|y_j(x_i) - \bar{y}(x_i)\right|}{C_{i,range}}, D_{max} = \max_{j=\overline{1,l}} D_j \quad (2.7)$$

$\bar{y}(x_i)$ – оціночне математичне очікування значень сканування в заданій точці

$$\bar{y}(x_i) = \frac{1}{n}\sum_{j=1}^{l} y_j(x_i), i = \overline{1,n} \quad (2.8)$$

Рішення про дійсність окремих реалізацій банкнот можна перевіряти на основі екстремумів. В ідеальному середовищі сканування кількість пагорбів, як і сама крива буде постійною величиною. Напроти, в умовах шумового забруднення можлива поява деякої кількості локальних екстремумів, як випадкової величини. Тоді в разі наявності достатньої кількості спостережень її можна описати знизу та згори як критерій дійсності сканування.

в) якщо p – кількість локальних екстремумів на $r(x_i)$, тоді це не суперечить гіпотезі про достовірність даної реалізації за умови виконання нерівності

$$p_{min} \leq p \leq p_{max} \quad (2.9)$$

де p_{min} та p_{max} – мінімальна та максимальна кількість локальних екстремумів обчислена по правильних реалізаціях кривої оптичного сигналу.

Окрім виникнення надмірної інформації в сигналі, зовнішні шуми, пов’язані з неідеальністю апаратного забезпечення, можуть призводити до її зсуву відносно напряму сканування. Тоді в ході спостережень за поведінкою привальних реалізацій можна визначити максимальну відстань такого спотворення, за якого сканування й надалі вважатиметься нормальним.

г) якщо $d_{r,min}$ – мінімальна відстань між двома сусідніми локальними екстремумами на $r(x_i)$ та $d_{r,max}$ – максимальна відстань відповідно, тоді вони не суперечать гіпотезі, якщо виконується

$$d_{r,min} \geq d_{min} \quad d_{r,max} \leq d_{max} \tag{2.10}$$

де d_{min} та d_{max} – мінімальна та максимальна відстань між сусідніми локальними екстремумами, вершинами на $y_j(x_i)$, обчислені на всіх правильних реалізаціях даної банкноти

$$d_{min} = \min_{j=\overline{1,l}} d_{j,min} \quad d_{max} = \max_{j=\overline{1,l}} d_{j,max} \tag{2.11}$$

У критеріях (в) та (г) можуть використовуватися «суттєві» локальні екстремуми: максимуми, для яких виконується

$$y_j\left(x_{i-j_1}\right) < y_j(x_i) = y_j(x_{i+1}) = \cdots = y_j(x_{i+k}) > y_j\left(x_{i+j_2}\right) \tag{2.12}$$

аналогічно для локальних мінімумів. Тоді точкою локального екстремуму вважаємо центр плато $x_0 = x_{i+k/2}$.

Результат сканування вважається відповідним правильній банкноті, якщо виконуються всі нерівності (а)-(г).

Введені критерії ефективні, якщо набір реалізацій $y_j(x_i), i = \overline{1,n}$ є репрезентативним. Перевірка даних на достовірність виконується одночасно чотирма некорельованими критеріями, кожен з яких відображає окрему властивість. Таким чином реалізується загальна методологія тестування статистичної гіпотези. Отже, пропонується аналізувати результат сканування банкнот на відповідність кожному зі сформованих критеріїв і подавати на додаткове дослідження в разі порушення будь-якого з них.

2.3 Аналіз якості результатів сканування за відстанню Махалонобіса

Визначення приналежності певного результату сканування до правильних банкнот чи аномалій достатньо для автоматичного пошуку та усунення викидів з вибірки. Однак у такому разі всі аномальні результати будуть інтерпретовані системою як еквівалентні. З постановки задачі маємо, що процесу перевірки окремих реалізацій банкнот на дійсність передує необхідний етап вибору множини правильних екземплярів сканування шляхом візуального огляду. Враховуючи час необхідний для введення купюр у систему, необхідне впровадження не тільки системи розпізнавання викидів, але й оцінки якості відповідних даних.

Для порівняння правильних та тестових реалізацій сканування банкноти пропонується застосування відстані Махаланобіса. Обрана метрика є узагальненням відстані Евкліда за винятком. На відміну від останньої, яка у просторі будь-якої вимірності дозволяє оцінити відстань між двома точками, дана міра зручна для обчислення відстані між окремою точкою чи розподілом від іншого розподілу або точки в заданому статистичному розподілі.

Згідно з методом Махаланобіса, оцінка схожості представляє відстань зважену коваріаціями – масштабними коефіцієнтами n-вимірної еліпсоїди – кожної змінної, точки на графіку, між собою. Таким чином можна ефективно перевірити чи відповідає певне сканування банкноти еталону через представлення її моделі коваріаційною матрицею та математичним очікуванням, обчисленими на правильній вибірці, які описують область, що займає вибірка на просторі ознак. Геометричний зміст відстані представлено на рисунку 2.6.

Як видно з рисунку, відстань між точками A, B, C і D та центром розподілу вибірки однакова, якщо обчислювати її за методом Евкліда. Напроти, при застосуванні відстані Махалонобіса бачимо, що відхилення першої пари точок жовтого кольору значно вище, за пару синіх. Це пов’язано з тим, що друга нормальна форма, яка відповідає відстані Евкліда, не враховує напрямів та розтягнення базису вибірки. Напроти, відстань Махаланобіса заснована на них.

Отже, враховуючи нерівномірний характер варіацій сигналу в окремих зонах, дана метрика задовільняє умовам поставленої задачі.

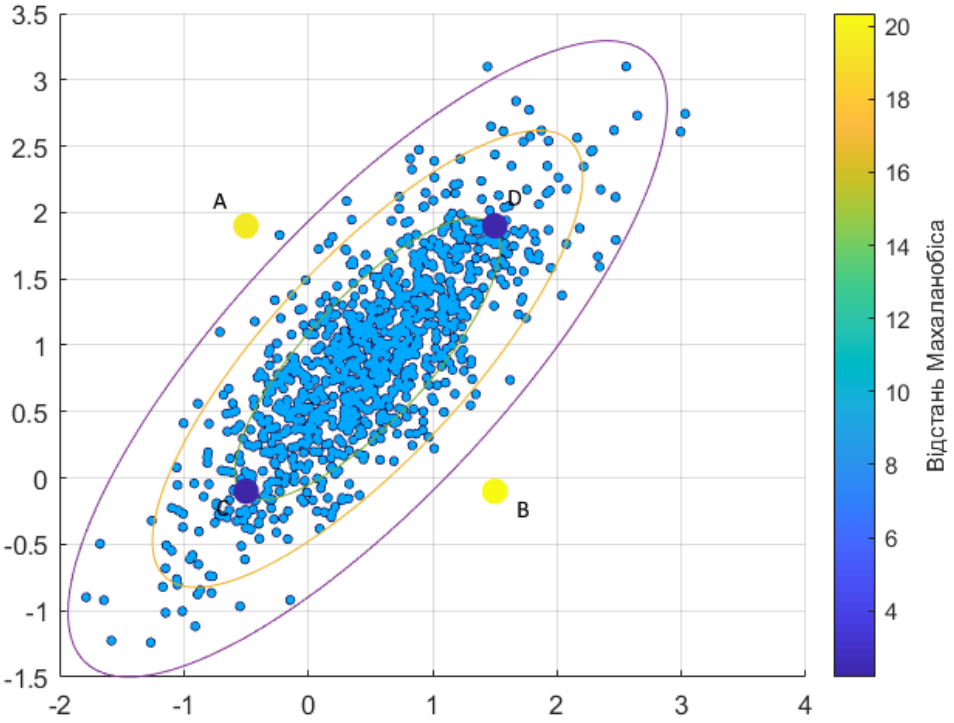


Рисунок 2.6 – Рівні відстані Махаланобіса у двовимірному просторі

Нехай $Y \in \mathbb{R}^n$ є випадковим вектором, компоненти якого мають нормальний розподіл в n вимірах з відомими параметрами: математичним очікуванням μ_Y та коваріаційною матрицею Σ_Y. Тоді, якщо y випадкова реалізація Y, розглянемо вираз

$$D(y) = \sqrt{(y - \mu_Y)^T \cdot \Sigma_Y^{-1} \cdot (y - \mu_Y)} \tag{2.13}$$

де число $D(y)$ вважається відстанню Махалонобіса, що оцінює належність окремої реалізації y до множини реалізацій випадкового вектору Y.

Число D можна розрахувати для будь-яких n-вимірів, але значення відстані буде різним для кожної кількості n, оскільки відстань D зростає пропорційно. Отже, на практиці, так як довжина оптичного сигналу залежить від розмірів сканованих банкнот (і тому є змінною), критичне значення доведеться підбирати щоразу. Однак μ_Y та Σ_Y^{-1}, що використовуються в (2.13), перетворюють значення змінної кожного виміру в незалежні стандартні нормальні розподіли з

центруванням μ_Y та Σ_Y^2 масштабуванням і обертанням. Звідси: $\Delta^2 = (Y - \mu_Y)^T \cdot \Sigma_Y^{-1} \cdot (Y - \mu_Y)$ є сумою n незалежних стандартних нормальних змінних та слідує розподілу хі-квадрат з n ступенями χ_n^2 свободи з функцією щільності [24]

$$f_{\chi_n^2}(x) = \frac{x^{\frac{n}{2}-1} \cdot e^{-\frac{x}{2}}}{2^{\frac{n}{2}} \cdot \Gamma\left(\frac{n}{2}\right)} \tag{2.14}$$

де Γ – гамма функція, так як χ^2 – особливий випадок гамма розподілу, для якої характерна асоційована функція кумулятивного розподілу $F_{\chi_n^2}(x) = P(\chi_n^2 \leq x)$.

Тоді, якщо задати достатньо мале число $\alpha \leq 0.05$, то будь-яка реалізація випадкової величини Y належатиме відрізку $\left[\chi_{\alpha,n}^2, \infty\right)$ в разі виконання

$$P\left(\chi_n^2 < \chi_{\alpha,n}^2\right) = \frac{1}{2^{\frac{n}{2}} \cdot \Gamma\left(\frac{n}{2}\right)} \int_0^{\chi_{\alpha,n}^2} x^{\frac{n}{2}-1} \cdot e^{-\frac{x}{2}} dx \tag{2.15}$$

Значення числа $\chi_{\alpha,n}^2$ знаходиться з відповідної таблиці для розподілу хі-квадрат з n ступенями свободи χ_n^2. Таким чином, якщо $D(y) < \chi_{\alpha,n}^2$, то гіпотеза про те, що y є реалізацією випадкового вектора Y відкидається, у протилежному випадку робимо висновок: статистичні дані не суперечать висунутій гіпотезі – та y є реалізацією випадкового вектора Y.

Таким чином, (2.15) можна використовувати для перевірки гіпотези виявлення аномалій на основі обчислення відстані Махаланобіса, а сам процес буде непараметричним, оптимальним з боку реалізації обчислень, незалежним від масштабування змінних та застосовним в умовах колінеарності між вимірами досліджуваного простору ознак [8].

Для конкретної реалізації запропонованого критерія необхідно зі статистичних даних оцінити параметри $\hat{\mu}_Y$ та $\hat{\Sigma}_Y$. Приймемо $Y \in \mathbb{R}^n$ за множину правильних реалізацій сканування банкноти $y_j(x_i), j = \overline{1,l}$, де n – це кількість вимірів множини, або довжина оптичного сигналу.

$$\hat{\mu}_Y = \left(\frac{1}{l} \sum_{j=1}^{l} y_j(x_t), t = \overline{1,n} \right)^T \quad (2.16)$$

$$\hat{\Sigma}_Y = \left(\frac{1}{l} \sum_{j=1}^{l} \left(y_j(x_t) - \bar{y}(x_t) \right) \left(y_j(x_m) - \bar{y}(x_m) \right); t = \overline{1,n}; m = \overline{1,n} \right) \quad (2.17)$$

За окрему реалізацію банкноти, що перевіряється, позначимо $r = r(x_i), i = \overline{1,n}$. Тоді, відповідно до (2.13) відстань Махаланобіса між окремою реалізацією сканування та оціночним статистичним розподілом визначатиметься наступним чином

$$D_M = \sqrt{(r - \hat{\mu}_Y)^T \cdot \hat{\Sigma}_Y^{-1} \cdot (r - \hat{\mu}_Y)} \quad (2.18)$$

де D_M – відстань між реалізацією банкноти, яка досліджується, та її математичним очікуванням.

У такому разі гіпотеза про дійсність даної банкноти вважатиметься не спростованою, якщо виконуватиметься співвідношення

$$D_M < D_{max} \quad D_{max} = \chi^2_{\alpha,n} \text{ (см. 2.15)} \quad (2.19)$$

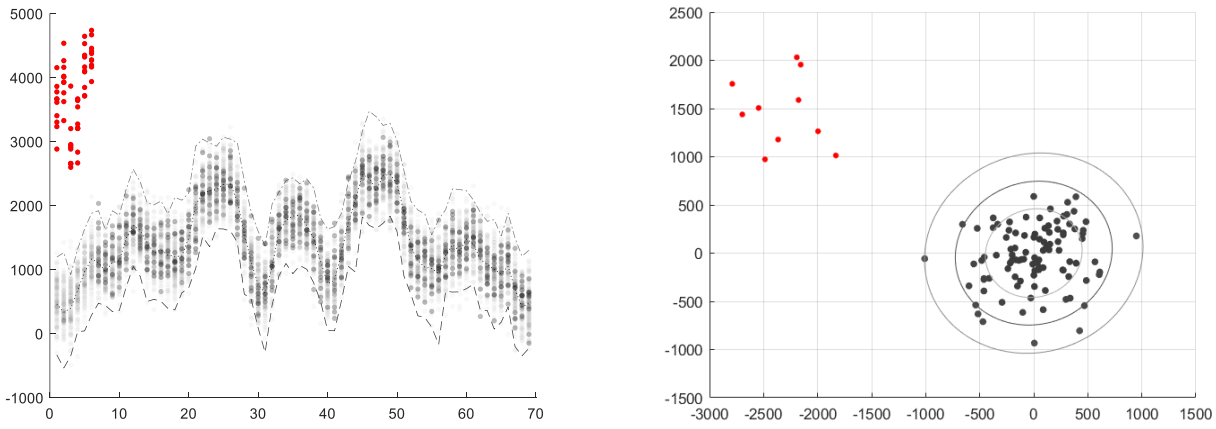
Слід зазначити, що даний метод ефективно розв’язує задачу тоді й лише тоді, коли кількість правильних реалізацій, які складають вибірку Y, достатньо велика (в силу асимптотичної нормальності розподілу відповідних середньоарифметичних). Напроти, якщо ця умова не виконується, обчислення неможливо виконати зі збереженням необхідної точності та вважати результати такого аналізу об’єктивними.

Також неможливо гарантувати відповідність нормальному розподілу значень змінних вибірки X за всіх можливих сценарії проведення сканування банкнот. Тож для запобігання хибних висновків пропонується перевіряти ті реалізації купюри, що метод вважає нормальними, додатково за емпіричними критеріями. Тоді метод працюватиме ефективно з гарантованою точністю завдяки залученню до прийняття рішень кількох некорельованих критеріїв, що відповідає загальній методології тестування статистичної гіпотези.

2.4 Вибір правильних реалізацій сканування методом головних компонент

Аналізу результатів сканування на нормальність передує етап візуального огляду сигналів з метою визначення на основі досвіду користувача правильних реалізацій банкнот. Однак, за відсутності початкового наближення чи уявлень оператора системи про нормальну форму і зображення сканів задача обрання коректних даних стає нетривіальною і може вести до хибних висновків через включення аномальних екземплярів до тренувального набору. Тож бачимо необхідність в побудові такого представлення сканів банкнот, яке б допомагало виявляти ознаки схожості між ними та швидко знаходити сукупності подібних реалізацій, що складатимуть вибірку правильних екземплярів.

В якості перетворення простору ознак результату сканування графічно зображуваного скінченною кривою пропонується зменшення розмірності даних до двох вимірів таким чином, аби множину графіків можна було однозначно представити точками в декартовій системі координат, як показано на рисунку 2.7.



а) б)

Рисунок 2.7 – Зменшення розмірності результатів сканування банкнот з $n = 69$ вимірів (а) у двовимірний простір (б)

З рисунку бачимо, що криві (а), точки яких виходять за межі допустимого коридору, знаходяться далеко від інших даних у двовимірному представленні (б). У такому вигляді їх легко відокремити від нормальних записів, а останні виділити

з-поміж інших та побудувати на основі обраної множини математичну модель правильного сканування.

2.4.1 Зменшення вимірності даних

Перетворення даних результатів сканування на точки у просторі, який можна представити на графіку виконуватимемо за допомогою аналізу головних компонент. Його метою є пошук такого нового базису, який є лінійною комбінацією початкового та найкраще описує вхідні дані згідно задачі дослідження.

Метод головних компонент є повністю непараметричним і заснований на проектуванні n-вимірної вибірки у k ($k \ll n$) вимірів [25]

$$X_{n\times m} \cdot P_{m\times k} = Y_{n\times k} \tag{2.20}$$

де P – лінійне перетворення X в Y.

Метою такого перетворення є зменшення розмірності вхідних даних таким чином, аби відкинути всю зайву інформацію та залишити тільки найважливішу. Інакше кажучи, аби кінцева проекція містила максимум варіацій початкової

$$\max \frac{1}{2n} \cdot \sum_{i=1}^{n} (P^T x_i - P^T \overline{x_i})^2 = P^T \Sigma_X P \tag{2.21}$$

де n – кількість вимірів вибірки X;

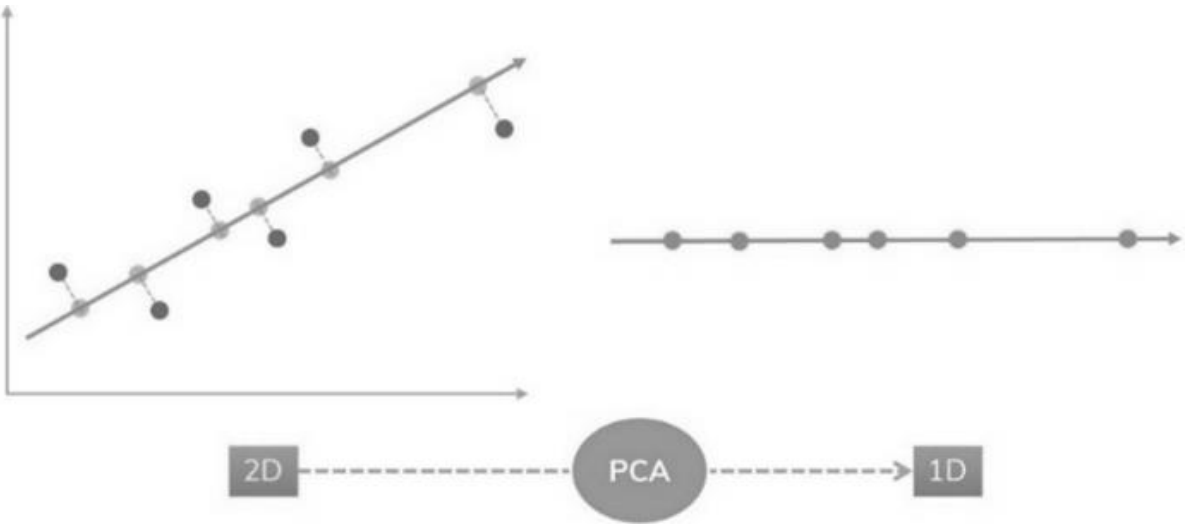
x_i – множина всіх значень і-тої вимірюваної змінної;

$\overline{x_\iota}$ – математичне очікування змінної;

Σ_X – коваріаційна матриця;

$P^T x_i$ – проекція кожного спостереження, рядку матриці X, у k вимірів.

На рисунку 2.8 представлено приклад проектування двовимірного простору в одновимірний. Бачимо, що головною компонентою є лінія регресії змінних, яка мінімізує відстань до початкових точок у просторі та, як наслідок, зберігає максимум відстані між їх проекціями. Аналогічним чином відбувається згортка просторів ознак більшої вимірності.



а) б)

Рисунок 2.8 – Зменшення розмірності простору з двовимірного (а) в (б) одновимірний методом головних компонент

Власними векторами матриці є такі вектори, які під дією лінійного перетворення не змінюють свого напряму

$$X \cdot \vec{v} = \lambda \cdot \vec{v} \tag{2.22}$$

де λ – певний дійсний скаляр, який задає величину розтягування вектору.

Власні вектори є головними компонентами вибірки і задають напрям розповсюдження об’єктів, які спостерігаються, відносно їх принципових ознак. Обчислення головних компонент виконуємо через власний розклад матриці. За теоремою, якщо матриця симетрична, її можна перетворити на діагональну (і навпаки) як лінійну комбінацію ортогональної матриці V, де рядками є власні вектори, та діагональної Λ, елементами головної діагоналі якої виступають власні значення матриці.

$$A = V \cdot \Lambda \cdot V^T \tag{2.23}$$

Нехай $X \in \mathbb{R}^{n \times m}$ – матриця розміром n на m, де рядками виступають незалежні спостереження за змінними $x_i, i = \overline{1,n}$, а m – це число цих змінних. Тоді її матриця коваріацій обчислюється як добуток

$$\Sigma_X = \frac{1}{n-1} X \cdot X^T \tag{2.24}$$

і є симетричною відносно головної діагоналі.

Відповідно до методу аналізу головних компонент, проекцією матриці X на базис власних векторів має бути така Y, коваріаційна матриця якої є діагональною. Якщо застосувати (2.32) для Y та підставити це в рівняння (2.28), отримаємо

$$S_Y = \frac{1}{n-1} YY^T = \frac{1}{n-1}(PX)(PX)^T = \cdots = \frac{1}{n-1} P(XX^T)P^T = \frac{1}{n-1} PAP^T \quad (2.25)$$

де $A = XX^T$ – симетрична.

Тоді, якщо обрати матрицю P таку, що кожен її рядок r_i є власним вектором A, тобто $P \equiv V^T$, то при підстановці в (2.23) рівняння набуде вигляду $A = P^T \Lambda P$. Тоді, якщо P ортогональна ($PP^T = PP^{-1}$), можна спростити (2.24) до

$$S_Y = \frac{1}{n-1} PAP^T = \frac{1}{n-1} P(P^T \Lambda P)P^T = \cdots = \frac{1}{n-1}(PP^{-1})\Lambda(PP^{-1}) = \frac{1}{n-1}\Lambda \quad (2.26)$$

У результаті еквівалентних перетворень бачимо, що P діагоналізує S_Y і тому ствердно наступне:

– головні компоненти матриці X є власними векторами XX^T її коваріаційної матриці;

– елементи головної діагоналі матриці S_Y є дисперсії X уздовж власних векторів, тобто власні значення.

Отже, головні компоненти досліджуваної вибірки можна обчислити через власний розклад її коваріаційної матриці, а алгоритм зменшення вимірності складається з таких кроків:

а) упорядкувати вибірку даних як матрицю X розміром $n \times m$, де m – кількість вимірюваних змінних (точок на графіку), а n – кількість реалізацій банкнот, які необхідно аналізувати;

б) обчислити та відняти математичне очікування для кожної змінної x_i – центрувати матрицю X

в) обчислити коваріаційну матрицю Σ_X;

г) отримати власні вектори P та власні значення Λ вибірки X зі власного розкладу матриці Σ_X;

д) упорядкувати власні вектори (далі – головні компоненти) за спаданням відповідних їм власних значень;

е) обчислити проекцію даних спостережень на k перших головних компонент з найвищими значеннями дисперсії (власними значеннями);

ж) продовжити аналіз реалізацій сканування банкнот іншими методами.

2.4.2 Помилка відтворення даних

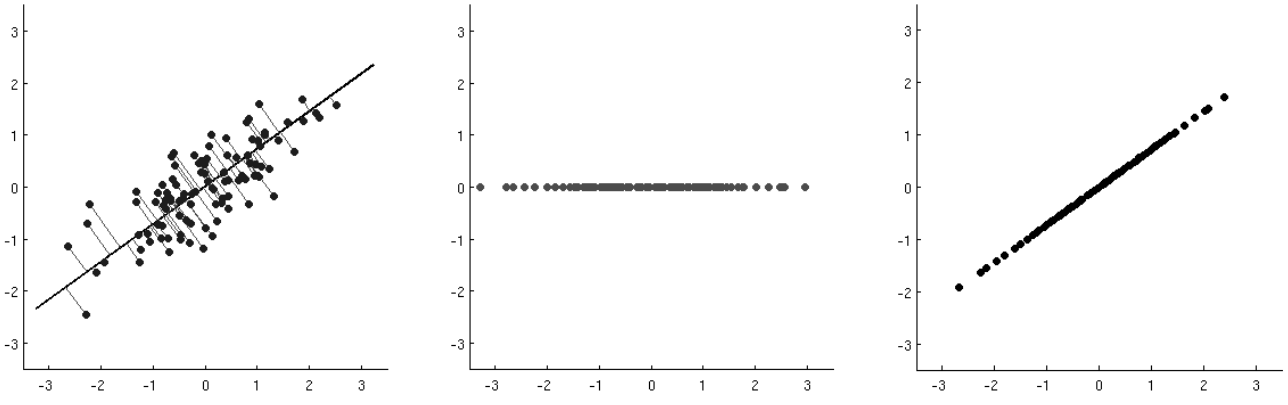
Важливою властивістю операції зі зменшення розмірності методом головних компонент є її зворотність. З метою відновлення даних, які були спроектовані на певний набір головних компонент, можна виконати обернену дію, помноживши проекцію на транспонування матриці власних векторів

$$\hat{X} = Y \cdot P^T = (X \cdot P) \cdot P^T \qquad (2.27)$$

де X – початкові дані;

$\hat{X}$ – відновлені дані.

Результатом наведеної лінійної комбінації будуть точки в початковому n-вимірному просторі ознак за винятком відсутності варіацій уздовж тих головних компонент, які були опущені під час першого проектування як неважливі. На рисунку 2.9 показано приклад відновлення.



а) б) с)

Рисунок 2.9 – Результати проектування даних (а) на одну з головних компонент (б) та їх відновлення (с)

На оберненості згортки вимірів заснований метод фільтрації шумів у даних, які в представлені аналізу головних компонент зазвичай мають низький рівень значущості. На рисунку 2.9 видно, як у результаті відновлення зникли варіації ортогональні лінії регресії, позначеної жирною чорною прямою. Однак, якщо прийняти відображені на вхідному рисунку точки як нормальні, а їх коливання відносно головної компоненти, як допустиме відхилення, то обчислене значення похибки можна вважати граничним і використовувати для класифікації даних і виявлення аномалій серед них.

За міру відмінності використовуємо середньоквадратичне відхилення між вхідними даними X та відновленими $\hat{X}$

$$D_R = \frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{x_i})^2 \tag{2.28}$$

де n – кількість змінних;

x_i – значення і-ї вхідної змінної;

$\hat{x_i}$ – значення і-ї змінної після відновлення методом головних компонент.

Враховуючи, що середньоквадратичне відхилення двох спостережень є сумою квадратів відстані між очікуваними значеннями $\hat{X}$ та реальними X, за граничне значення помилки відновлення візьмемо дисперсію даних у межах множини правильних екземплярів – тобто середньоквадратичне відхилення правильних реалізацій об’єкту дослідження від його математичного очікування, яке обчислюємо за формулою

$$D_{max} = \frac{1}{n}\sum_{i=1}^{n} S_{x_i}, \quad S_{x_i} = \frac{1}{l-1}\sum_{j=1}^{l}\left(y_j(x_i) - \bar{y}(x_i)\right)^2 \tag{2.29}$$

де S_{x_i} – стандартне відхилення і-тої змінної x_i;

l – кількість правильних реалізацій сканування банкноти;

$y_j(x_i)$ – значення змінної x_i у j-тій реалізації;

$\bar{y}(x_i)$ – математичне очікування змінної x_i.

Тоді реалізація сканування, яка перевіряється, вважатиметься дійсною, коли виконується співвідношення

$$D_R \leq D_{max} \tag{2.30}$$

На відміну від методу Махаланобіса, який для обчислення відстані потребує обчислення оберненої матриці коваріацій, метод головних компонент не залежить від розміру вибірки і здатний працювати в умовах, коли кількість даних менше, за число змінних. Отже, наведену міру відхилення – помилку відновлення – можна використовувати разом чи замість відстані Махаланобіса у випадках, коли остання не застосовна в силу вимог до розмірності тренувальних даних.

2.5 Алгоритм виявлення аномалій у результатах сканування

Процедура виявлення аномалій передбачає реалізацію чотирьох етапів обробки даних:

а) підготовчих

1) складання вибірки нормальних реалізацій сканування банкноти;
2) обчислення параметрів математичної моделі еталону;

б) оцінки рівня аномальності

3) перевірки кожної реалізації банкноти на відповідність критеріям нормальності;

в) класифікації

4) прийняття рішення щодо ступеня підозрілості реалізації, яка перевіряється, на основі результатів перевірки.

Нехай R є множиною всіх наявних та можливих реалізацій сканування банкноти оптичними сенсорами представлених у дискретній формі як скани, складові яких – треки – є основними об’єктами аналізу в цій роботі. У ролі елементів множини R виділимо дві групи даних: нормальні Y – реалізації сканування визначені користувачем, як правильні, які відповідають дійсним

екземплярам купюр; та підозрілі S – неперевірені реалізації, дійсність яких не відома і потребує підтвердження $S = R \setminus Y$.

Тоді повне визначення алгоритму виявлення аномалій є наступним:

підготовчий етап

а) Обрання множини правильних реалізацій сканування банкнот Y:

1) згортка вхідного простору ознак у два виміри:
   - обчислення коваріаційної матриці Σ_R за формулою (2.24) за всіма наявними екземплярами даних R;
   - отримання власних векторів та власних значень матриці V_R, Λ_R методом власного розкладу (2.23);
   - сортування власних векторів за спаданням відповідних їм власних значень для знаходження головних компонент найбільшої ваги

$$\lambda_{R_1} \geq \lambda_{R_2} \geq \cdots \lambda_{R_m} \tag{2.31}$$

   - проектування вхідного простору ознак у точки в декартовій системі координат (2.23)

$$R^* = R \cdot V_{\{1,2\}} \tag{2.32}$$

   - зображення проекцій сигналів на графіку;
2) візуальний аналіз наявних реалізацій на основі просторових співвідношень зі представлення, отриманого на попередньому кроці
   - обрання на множині Y попереднього набору Y_0 правильних треків;
3) візуальний аналіз записів у складі Y_0 на основі їх графічного представлення у формі скінченних кривих
   - коригування складу Y_0 оператором на основі досвіду;
   - отримання тренувальної вибірки Y, елементи якої відповідають вимогам до правильності сканування.

б) Обчислення параметрів математичної моделі правильних банкнот вибірки Y:

1) математичного очікування $\bar{Y}$ за формулою (2.1)

$$\bar{Y} = \left\{ \bar{y}_i : \bar{y}(x_i) = \sum_{j=1}^{l} y_j(x_i), i = \overline{1,n} \right\} \quad (2.33)$$

2) варіаційного коридору C за формулою (2.3)

$$C = \left\{ \left(C_{i,min}, C_{i,max} \right) | i = \overline{1,n} \right\} \quad (2.34)$$

– мінімального C_{min} та максимального C_{max} допустимих значень для кожної змінної x_i

– розмаху варіаційного інтервалу в кожній точці $C_d = C_{max} - C_{min}$

3) характеристик локальних екстремумів (2.9)

– мінімальної p_{min} та максимальної p_{max} кількості екстремумів;

– мінімальної d_{min} та максимальної d_{max} відстані між сусідніми екстремумами;

4) коваріаційної матриці Σ_Y за формулою (2.24)

– власних векторів V_Y, які відповідають першим двом головним компонентам вибірки Y;

– оберненої коваріаційної матриці Σ_Y^{-1} за умови, якщо кількість l правильних реалізацій банкнот більше, за кількість n дискрет оптичних сигналів;

5) граничних значень теоретичних критеріїв аномальності:

– максимальної допустимої відстані D_M^{max} Махаланобіса, яка задовільняє рівнянню (2.15)

– максимального середньоквадратичного відхилення D_{max} між фактичними та теоретичними значеннями змінних за формулою (2.29)

етап оцінки

в) Аналіз елементів множини S неперевірених реалізацій сканування запропонованими критеріями:

1) якщо не існує або пуста множина правильних реалізацій сканування Y

– зупинити аналіз та повернутися до кроку 1;

2) якщо не існують або відсутні деякі параметри математичної моделі правильних сканувань
   – зупинити аналіз та повернутися до кроку 2;
3) якщо не нульова обернена коваріаційна матриця Σ_Y^{-1}, обчислити відстань D_M між елементами множини S та центром розподілу $\bar{Y}$ за формулою (2.18)
   – або вважати тест пройденим та перейти до кроку d;
4) обчислити середньоквадратичне відхилення D_R між вхідними даними S та їх оцінкою в результаті відновлення $\hat{S}$ на основі проекційної матриці $P_Y = V \cdot V^T$ за формулою (2.28);
5) перевірити виконання емпіричних критеріїв дійсності результатів сканування банкноти
   – у разі порушення умов будь-якого із критеріїв зупинити виконання кроку е та вважати тест проваленим;

етап класифікації

г) Обробка результатів аналізу множини S за емпіричними та теоретичними критеріями:
   1) якщо всі критерії не суперечать гіпотезі, про дійсність даної реалізації, вважати її «правильною»;
   2) якщо виконання кожного критерію порушується, вважати реалізацію, що перевіряється, «аномальною»;
   3) у разі невиконання будь-якого із наявних критеріїв вважати банкноту «підозрілою» та рекомендувати проведення додаткового дослідження.

2.6 Оцінка ефективності критеріїв аномальності банкнот

Для перевірки гіпотези про ефективність запропонованих критеріїв проведемо тестування на двох наборах даних. Перший набір буде представлений результатами чисельної симуляції сигналів. Другу групу складатимуть записи бази даних сканів банкнот, сформованої при налаштуванні валідатора.

Кожен із досліджуваних критеріїв має власну область значень. Тому, для узагальнення отримуваних результатів та спрощення їх інтерпретації оцінки аномальності буде нормовано в інтервалі $d \in [0; 1]$ у спосіб

$$d = 1 + \frac{1}{(1 + d_0)} \tag{2.35}$$

де d – нормована оцінка критерію;

d_0 – початкове значення оцінки;

Отримані значення відстаней будуть представлені на рисунках у такій послідовності:

– емпірична відстань, D_E – середнє нормоване за розмахом відхилення сигналу від його варіаційного коридору (2.6);

– відстань Махаланобіса, D_M (2.18);

– помилка відтворення, D_R, методом головних компонент (2.28);

– інтегрована оцінка аномальності на основі трьох запропонованих методів.

Для кращої інтерпретації результатів тестування для кожного методу обчислимо точність (ACC), прогностичну значущість позитивного (TPR) та негативного результатів (TNR).

$$ACC = \frac{TP + TN}{P + N} \qquad TPR = \frac{TP}{P} \qquad TNR = \frac{TN}{N} \tag{2.36}$$

де TP – кількість істинно позитивних результатів (правильних реалізацій розпізнаних як такі);

TN – кількість істинно негативних елементів (виявлених дійсно аномальних реалізацій);

P, N – число дійсно правильних та аномальних сканувань у тестованій вибірці;

2.6.1 Результати чисельної симуляції

Застосуємо запропоновані критерії на штучних даних. Для цього оберемо екземпляр правильної реалізації сканування банкноти та проведемо спотворення його форми за шаблонами, що властиві реальним правильним та пошкодженим купюрам.

Вхідна реалізація сканування представлена на рисунку 2.10а буде використана як математичне очікування дійсної банкноти. Джерелом коливань, які утворять досліджувану вибірку буде генератор випадкових чисел, які походитимуть із нормального розподілу. У такий спосіб правильні реалізації вибірки представлятимуть лінійну комбінацію вхідного сигналу, випадкового збурення та шаблону аномальності, який буде додано до частини сигналів з метою симуляції викидів

$$R = E + N(0, \sigma) + k \cdot T \tag{2.37}$$

де R – множина всіх отриманих в результаті симуляції реалізацій;

E – математичне очікування сигналу, оригінальна реалізація банкноти;

$N(\mu, \sigma)$ – нормальний розподіл;

k – коефіцієнт вираження шаблону (нульовий для правильних сканувань);

T – шаблон аномальності, який представляє послідовність значень отриману в результаті обчислення різниці між математичним очікуванням та реальним спотворенням певного типу.

Відповідно до постановки задачі аномалій в результатах сканування можуть бути викликані виходом з ладу апаратного забезпечення та людським фактором неуважності при введенні даних у систему. Сильне шумове спотворення, невідповідність калібрування сигналу заданим вимогам, коли загальний рівень інтенсивності – значення дискрет – вищий, за очікуваний, наявність механічних спотворень на самому папері є найпоширенішими видами аномалій, які зустрічаються на практиці і мають бути ідентифіковані в першу чергу. Тому, в ході

чисельної симуляції розглянемо ці випадки. Приклади досліджуваних аномалій зображені на рисунку 2.10.

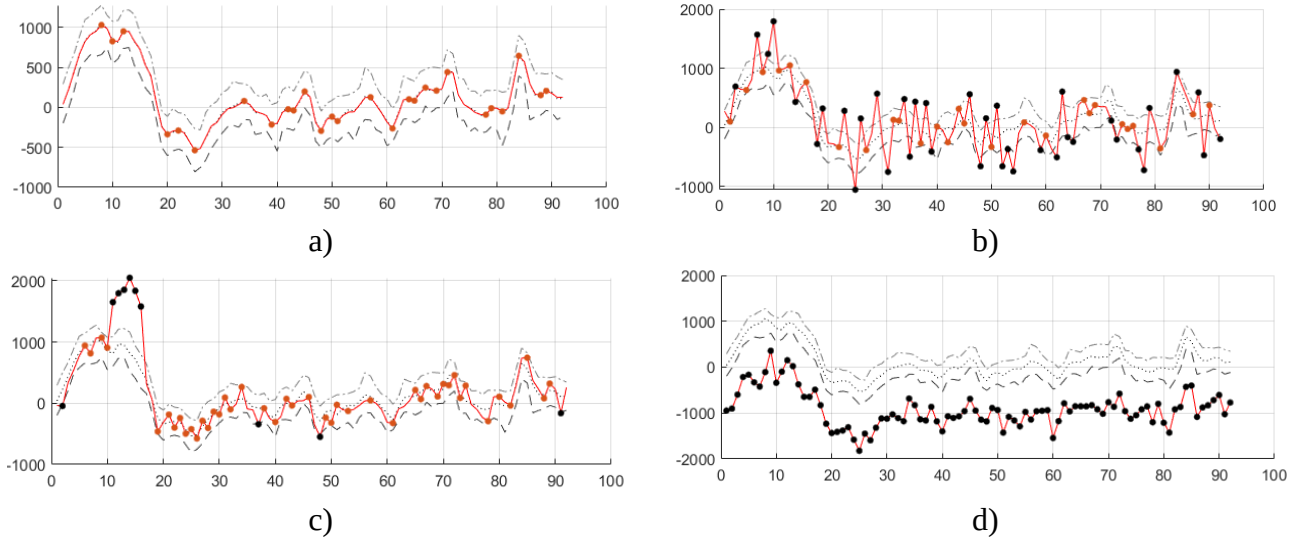


Рисунок 2.10 – Приклади симуляції оптичних сигналів: вхідна реалізація (a), сильне шумове спотворення (b) – несправний сенсор, (c) – пагорб довжиною в 5 точок – імітація скотчу, (d) – низький рівень сигналу некаліброваного сенсору

В результаті чисельної симуляції отримали реалізації сканування банкнот, які представлені в наступних кількостях:

а) 200 правильних реалізацій ($\sigma = 100, k = 0, T = 0$), з яких 50% складатимуть тренувальну вибірку Y;

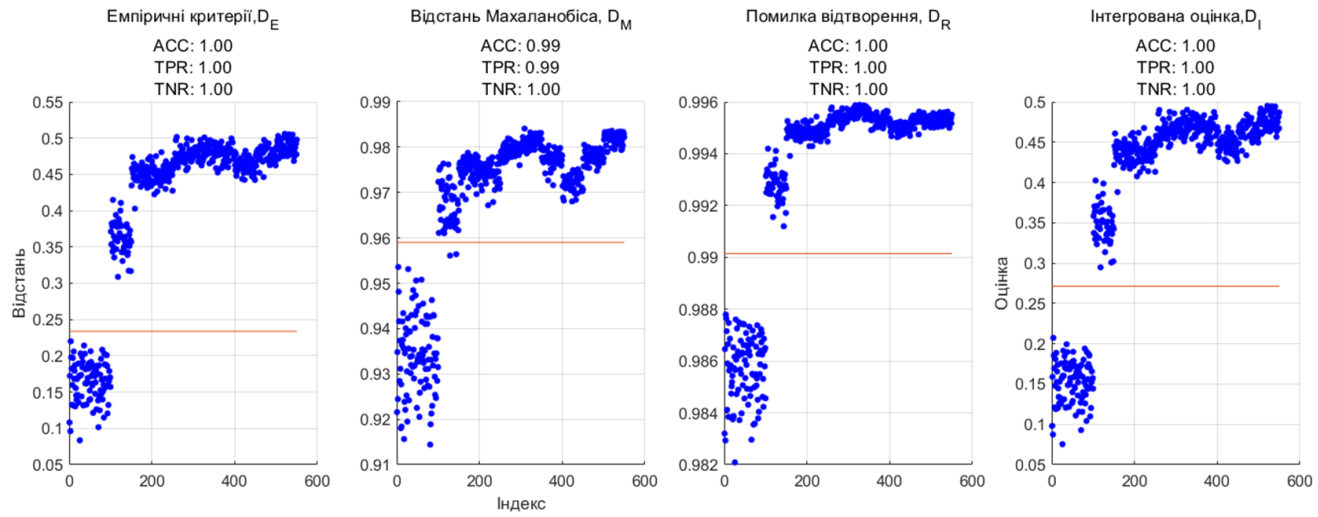
б) 50 реалізацій ($\sigma = 200, k = 0$), які спотворено сильним шумовим навантаженням (рисунок 2.10б);

в) 50 реалізацій ($\sigma = 100, k = -1$), на яких середнє значення сигналу нижче, за його математичне очікування (рисунок 2.10d);

г) 50 реалізацій ($\sigma = 100, k = +1$), де середнє значення сигналу вище за його математичне очікування;

д) 450 реалізацій ($\sigma = 100, k = +0.5$), кожній з яких відповідає унікальне положення пагорба, як показано на рисунку 2.10c.

Після обчислення значень критеріїв отримали результати, які наведені на рисунку 2.11.



a) b) c) d)

Рисунок 2.11 – Результати перевірки штучно згенерованої вибірки трьома критеріями: середнє нормоване за розмахом відхилення від коридору (a), відстань Махаланобіса (b), помилка відтворення методом головних компонент (c), інтегральна оцінка (d)

Бачимо, що запропоновані критерії ефективно відрізняють правильні реалізації банкнот від штучно спотворених та можуть бути застосованими при вирішенні задачі розпізнавання пошкоджених даних на реальній вибірці. З близькості отриманих оцінок до порогових значень, що відстань D_M Махаланобіса менш чутлива до спотворень у даних, ніж інші критерії, однак із вищою імовірністю спрацює при спробі ввести фальшивку в систему. Напроти, оцінки відстані D_E, заснованої на емпіричних характеристиках сигналу, щільно прилягають до граничного значення, а тому її здатність до відбивання брудних купюр (які не прийматимуть банки) чи зашумлених даних, що негативно вплинуть на тренування результуючих моделей, є високою. Допустима область помилки відтворення, D_R, робить її універсальною заміною другого критерію для випадків, коли його обчислення неможливе.

2.6.2 Емпіричне дослідження

Для оцінки ефективності запропонованих критеріїв при застосуванні на реальних даних було взято вибірку з бази даних сканування, яка складається з дійсних та фальшивих банкнот та містить 463 реалізації: з них 394 дійсних та 69 фальшованих. Для обчислення параметрів результуючої моделі на основі методу зменшення розмірності, як показано на рисунку 2.12, з набору правильних реалізацій було обрано 130 екземплярів, що приблизно дорівнює третині всіх відомих нормальних сканувань.

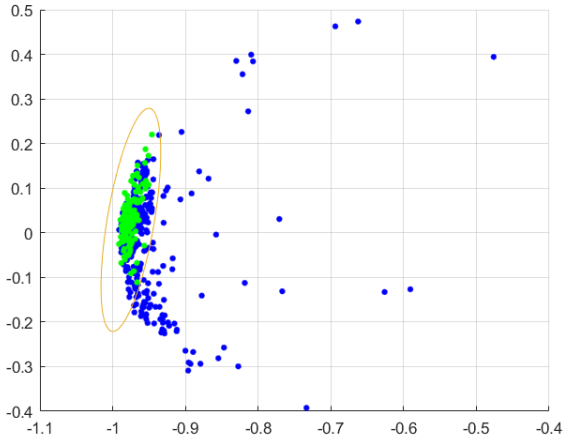


Рисунок 2.12 – Двовимірне представлення реалізацій сканування отримане методом головних компонент

В результаті обробки цієї вибірки визначили форму та варіаційний коридор правильного сигналу, як показано на рисунку 2.13.

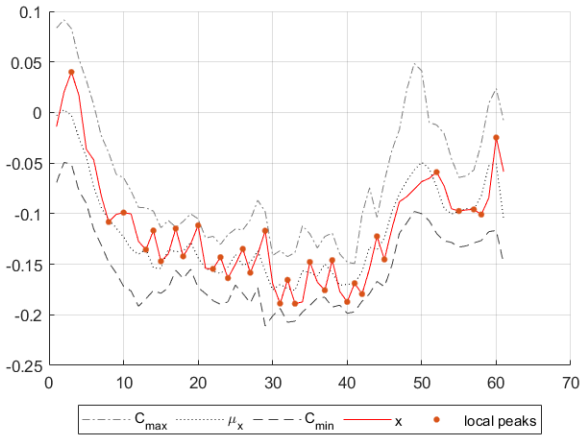
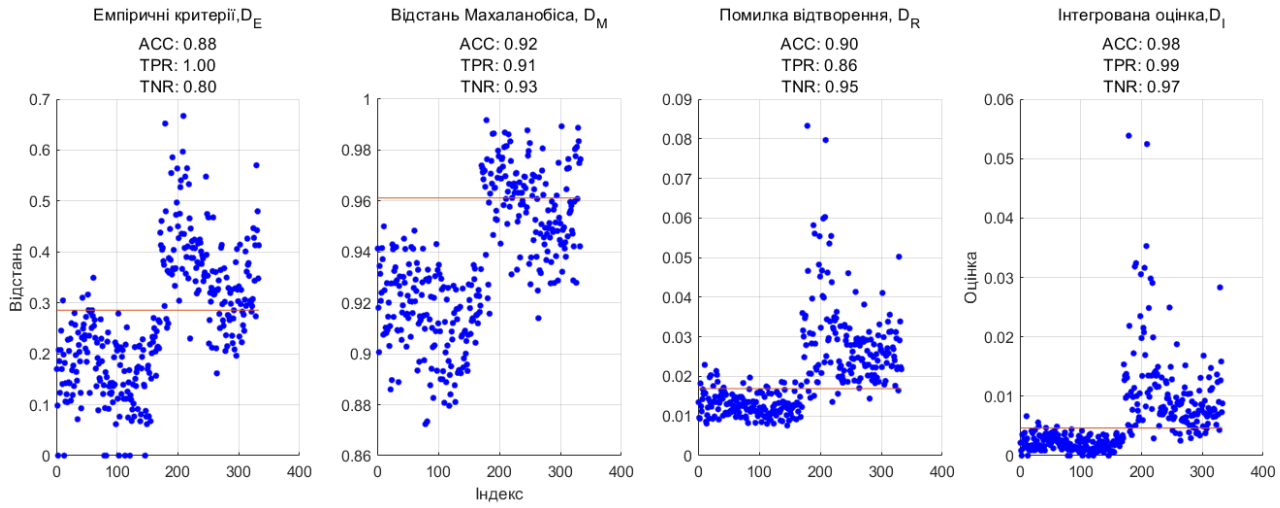


Рисунок 2.13 – Результуюча модель реалізації сканування банкноти

Отриману модель разом із запропонованими критеріями було застосовано до аналізу 300 реалізацій сканування. Обчислені показники аномальності представили на рисунку 2.14. Видно, що емпіричне дослідження характеристик сигналу дозволило правильно класифікувати всі дійсні банкноти у той час, як точність розпізнавання аномалій виявилася достатньо низькою і склала 80% на користь некоректно забракованих правильних реалізацій. На відміну від першого підходу, відстань Махаланобіса показала кращі результаті – 93% викидів, однак допустила помилки в обробці істинних банкнот. Помилка відтворення виявилася найбільш збалансованою – 95% аномалій та 86% дійсних купюр із загальною точністю в 90%. Інтегральна оцінка поєднання трьох критеріїв показала високий рівень ефективності методу і точність у 98% тестової вибірки, що повністю задовільняє поставленій задачі.



a) b) c) d)

Рисунок 2.14 – Результати емпіричного дослідження за критеріями: середнє нормоване за розмахом відхилення від коридору (a), відстань Махаланобіса (b), помилка відтворення методом головних компонент (c), інтегральна оцінка (d)

Отже, в умовах обробки реальних даних запропонована інтегрована оцінка виявлення аномальних та фальшованих банкнот заснована на емпіричних та теоретичних критеріях може бути використана для ефективного розпізнавання підробок та виявлення збоїв апаратного забезпечення.

Висновки до розділу

У даному розділі було надано формальне визначення моделі даних, які досліджуються – реалізації оптичного сканування банкноти – та наведено алгоритм її обчислення в контрольований спосіб. Також було наведено основні характеристики моделі такі, як математичне очікування, варіаційний коридор та особливості зміни інтенсивності світла у процесі руху банкноти, що задається локальними екстремумами сигналу.

Розроблено чотири емпіричні критерії перевірки правильності реалізації банкноти. Наведені способи обчислення їх граничних значень. Критерії побудовані на параметрах математичної моделі та використовуються в загальному алгоритмі як рубіжний контроль у прийнятті рішення щодо дійсності купюри за певним скануванням.

В якості основного теоретичного критерію обрано відстань Махаланобіса. Зроблено припущення про нормальний розподіл значень сенсорів у кожній точці сканування. На його основі обґрунтовано вибір імовірнісного підходу до визначення критичної області відстані та класифікації сканів як аномальних у разі потрапляння в неї.

Перший теоретичний критерій виявився чутливим до якості тренувальних даних результуючої моделі. Тому для покращення процесу ручного обрання оператором правильних реалізацій зі спільної бази даних запропоновано використання методу аналізу головних компонент, що дозволило згорнути багатовимірні оптичні сигнали в точки у системі декартових координат. Як наслідок зі зменшення вимірності даних отримано другий теоретичний критерій аномальності даних – помилку відтворення сигналів до початкової форми.

У результаті розроблено алгоритм виявлення аномалій на основі чотирьох емпіричних та двох теоретичних критеріїв. Проведено оцінку його ефективності на результатах чисельної симуляції та реальних сканувань.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДУ

Система виявлення пошкоджених реалізацій сканування банківських білетів має стати зручним інструментом для пошуку та аналізу аномальних елементів у базах даних банкнот. Від швидкості роботи методу та простоти графічного інтерфейсу залежатиме ефективність використання нового інструменту та реалізація ним мети дослідження щодо автоматизації виявлення спотворень у тренувальній вибірці для забезпечення якості даних.

3.1 Вимоги до програмного забезпечення

Згідно з поставленою задачею, програмний засіб повинен задовільнити вимоги існуючого процесу виробництва. Кінцева реалізація застосунку має бути інтегрована в середовище роботи операторів з налаштування детекторів банкнот та використовувати максимум із доступних технологій з мінімальним додаванням сторонніх систем: відповідати мові розробки проєкту, не використовувати зовнішніх бібліотек тощо. Можливості з розширення програми новими інструментами обмежена набором базових технологій – і реалізація методу з виявлення аномалій має задовільняти цим вимогам.

Існуюча універсальна система з побудови моделей розпізнавання заснована на модульному підході, де кожна наступна технологія додається до застосунку як розширення і може працювати незалежно від інших. Дослідження структурних особливостей грошей проводять через введення банкнот у валідатор та аналіз даних сканів, які він повертає в систему. Кожне введення завершується додаванням до бази даних пулу із монохромних зображень та сигналів, що відповідають усім доступним кольорам, які підтримує пристрій. Далі їх організовують у записи та зберігають в базі даних. Для побудови та оцінки якості системи записи складають у групи за номінальними значеннями грошової одиниці – таким чином кожен набір відповідає одному зображенню купюри і в разі дотримання правильності сканування наближується до ідеального представлення банкноти, як було показано

в розділі 2 раніше. Звідси випливає проблема частого звернення програми до файлової системи та навантаження оперативної пам’яті до її нестачі для повноцінної роботи застосунку.

Таким чином реалізація методів виявлення аномалій вимагає від програмного підтримку таких функцій:

а) підтримка читання файлів даних, які використовують у виробництві для збирання сканів;

б) графічне представлення даних як зображень чи сигналів, графіків, залежно від кількості сенсорів;

в) внесення змін у файли даних на вимогу користувача.

Запропонований метод заснований на представленні обраних множин правильних реалізацій банкноти – сканів – як матриць дійсних чисел та виконанні лінійних перетворень над ними для отримання статистичних характеристик простору ознак, який вони представляють, і обчислення відстані між окремими екземплярами до цього простору відповідно до моделі розподілу. Через великі обсяги даних, які потрібно аналізувати одночасно, новий програмний засіб має працювати швидко, а кількість повторюваних операцій: обчислень та копіювання – необхідно звести до нуля.

Програмне забезпечення методу має включати реалізацію математичних операцій, лінійних перетворень та представлень даних:

а) створення інтерфейсу над масивами та сегментами файлів:
   1) n-вимірної матриці;
   2) вертикального чи горизонтального вектору;
   3) діагональної матриці;
   4) скаляру;

б) реалізація представлень, вікон, матриць для обробки зображень;

в) виконання лінійних матричних операцій (суми, різниці, скалярного множення та ділення);

г) векторний добуток та матричне множення;

д) чисельне вирішення матричних рівнянь (систем лінійних алгебраїчних рівнянь) із допустимою точністю;

е) підтримку розкладів матриць:

1) власний розклад та сингулярний розклад для отримання власних значень та власних векторів простору ознак упорядкованих за спаданням значущості компонент;

2) QR-розклад матриці на ортогональну та праву трикутну для обчислення оберненої матриці;

ж) обчислення оцінки першого та другого центральних моментів статистичного розподілу для окремих вимірів матриць та матриці загалом;

з) реалізація функцій щільності розподілу;

и) реалізація фільтрів одно- та двовимірних сигналів для усунення шумів або згладжування даних.

Відповідно до предметної області та розроблених методів аналізу сканів сформували основні вимоги до програмного забезпечення:

а) з боку експертної систем:

1) підтримувати файли даних для читання сканів банкнот у форматі .bin

2) виконувати зчитування наборів даних в оптимальний спосіб для заощадження ресурсів операційної системи;

3) реалізація математичної моделі оптичного сигналу;

4) запис параметрів результуючих математичних моделей до файлу формату .bin для їх повторного використання після закриття застосунку;

5) реалізація емпіричних та теоретичних критеріїв для аналізу сканувань на основі їх правильних реалізацій;

б) з боку користувача:

1) підтримувати файли даних для читання сканів банкнот у форматі .bin;

2) вивід зображень сканів банкнот за кольорами;

3) виведення графіків оптичних сигналів, які відповідають окремим трекам сканування;
4) аналіз якості зображень банкнот на основі візуального огляду;
5) вивід результатів аналізу даних на екран у вигляді графіків та списків аномальних реалізацій із визначенням причини.

Також виділимо ряд нефункціональних вимог до програмного забезпечення:

– підтримка виконання програми в операційних системах Windows 7, 8, 10;
– простий та інтуїтивний інтерфейс застосунку;
– відповідність інтерфейсу розробленому методу виявлення аномалій.

3.2 Вибір засобів розробки програмного забезпечення

Для реалізації програмного забезпечення було обрано об’єктно-орієнтовану мову програмування загального призначення C++. Вона є сумісною зі стандартами низькорівневої процедурної мови C та підтримує застосування існуючих старих, але високопродуктивних бібліотек, які використовуються в більшості аналогів реалізацій підпрограм для лінійної алгебри, бібліотеках з каталогу програмного забезпечення написаних мовою Python таких, як NumPy. Виходячи зі специфіки реалізації цільової системи, куди планується інтегрувати застосування, дана мова є однією з основних вимог до розробки методу.

Для побудови математичної моделі сканування банкноти необхідна реалізація алгоритмічного забезпечення описаного в розділі 2. Відповідно до сформованих вимог шукане рішення повинно забезпечувати всі необхідні операції з перетворення даних та надавати зручний інтерфейс до низькорівневих структур даних, представлених у системі потоками байт. Для цього можна використовувати високорівневу відкриту шаблонну бібліотеку Eigen, написану мовою C++. Під шаблонністю розуміється механізм мови програмування, який дозволяє описувати сценарії визначення абстрактних типів та структур даних, процедур та функції без прив’язки до деяких параметрів їх реалізації. На відміну від статичних класів описи шаблонів не входять до виконуваного файлу програми та існують лише в деталях

її реалізації – сирцях. Для використання таких структур замість шаблонних параметрів, якими можуть бути типи, константи чи функції, необхідно підставити нешаблонний клас або літерал. Таку операцію називають спеціалізацією шаблону, яка є визначенням конкретного типу даних та входить у склад виконуваного файлу програми за результатами компіляції.

Для бібліотеки Eigen характерні гнучкість, швидкість та універсальність [26]. Вона реалізована за стандартом C++14, який підтримують більшість компіляторів даної мови. Вона дозволяє працювати з даними різних типів та розрядності, створювати обгортки для низькорівневих даних з інтерфейсом масиву чи будь-яких інших типів даних, які підтримують операції індексації. Така гнучкість у мові C++ забезпечується саме через шаблонність бібліотеки. Для реалізації запропонованого методу вона цікава в першу чергу великою бібліотекою операцій лінійної алгебри, геометричних перетворень та матричних розкладів, які потрібні для функціонування розробленої експертної системи. Однак, спроби впровадження бібліотеки Eigen у цільовий проект призвели до збільшення часу компіляції збірки втричі, що виявилося абсолютно неприйнятним наслідком і мотивувало її заміну. Причиною такої деградації в часі збірки програмного комплексу є повна шаблонна реалізація Eigen та надмірна функціональність, яку надає бібліотека.

У якості альтернативного інструменту для реалізації програмного засобу можливе використання GSL (GNU Scientific Library). GSL – це відкрита числова бібліотека математичних підпрограм написана мовою C та призначена для виконання математичних обчислень. Вона надає широкий спектр математичних процедур, таких як генератори випадкових чисел, спеціальні функції та підгонка методом найменших квадратів. Всього існує понад 1000 функцій з великим набором тестів [27]. На відміну від Eigen дана бібліотека є повністю статичною, підтримує основні чисельні типи даних потрібні для реалізації модулю та надає відповідний інтерфейс для більшості з них. Кожен файл її реалізації представляє завершений модуль і може інтегруватися в проект окремо від інших. Окрім зазначеної вище функціональності, вона також підтримує широкий спектр

розкладів матриць та операцій над ними, в тому числі: обернення, власний розклад, розв'язання матричних рівнянь тощо.

Для реалізації високопродуктивного оптимального за швидкістю множення матриць розглядається використання підпрограм BLAS. Вони призначені для виконання основних операцій лінійної алгебри і застосовується в більшості систем подібних до програмного засобу виявлення аномальних сканів, який розробляється. Наявні процедури бібліотеки поділяються на три рівні залежно від типів даних на яких застосовуються:

– підпрограми першого рівня виконують скалярні, векторні та міжвекторні операції;

– другий рівень складають процедури множення між вектором та матрицею;

– третій рівень забезпечує матричне множення.

Функція DGEMM забезпечує високу чисельну точність та швидкість обчислення добутку двох матриць розмірності $m \times n \times k$. Дослідження продуктивності, результати якого представлено на рисунку 3.1, даної підпрограми в порівнянні з аналогічною реалізацією в Eigen показало, що BLAS не тільки ефективно вирішує задачу множення без залучення мультипроцесорного програмування, але й надає статичний інтерфейс до своїх функцій.

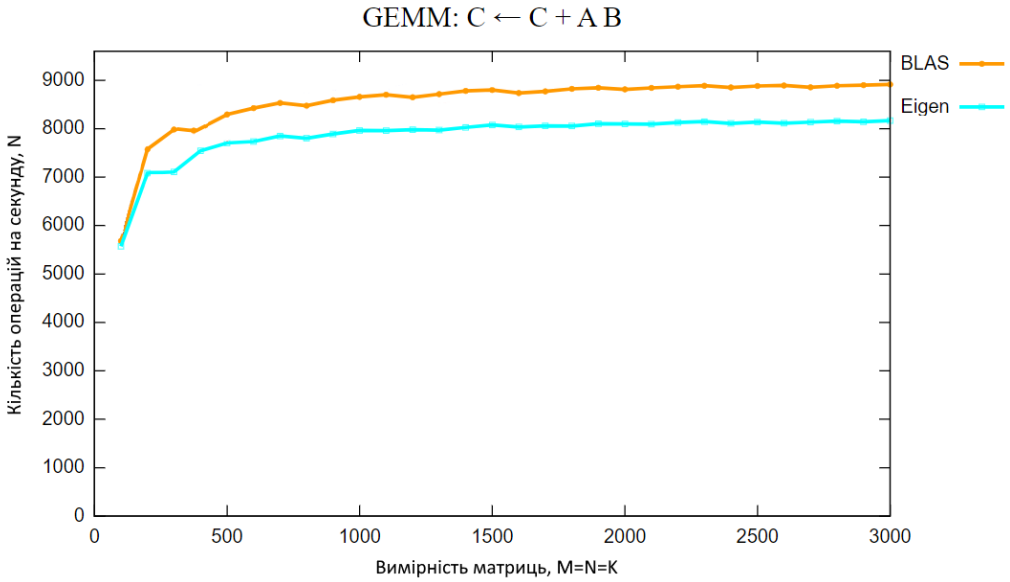


Рисунок 3.1 – Результати тесту продуктивності операцій множення в реалізаціях бібліотек BLAS та Eigen

З урахування обмежень цільового середовища розробки розробку графічного інтерфейсу користувача буде виконано за допомогою відкритої безкоштовної бібліотеки графічних класів Windows Forms із застосуванням .NET Framework фреймворку через програмний інтерфейс до мови C++/CLI. Вона забезпечує велику бібліотеку компонентів для реалізації користувацьких застосунків та містить елементи інтерфейсу, які повинні бути присутніми в системі виявлення аномалій згідно з описом методу.

Розробка користувацької частини системи виявлення фальшованих банкнот виконуватиметься в середовищі Visual Studio 2019, яке є загальноприйнятим та базовим для роботи з бібліотекою Windows Forms та надає зручні засоби для проєктування інтерфейсної частини додатку шляхом розміщення візуальних компонентів на канві шляхом перетягування зі списку компонентів або через динамічне програмування.

3.3 Архітектура програмного забезпечення

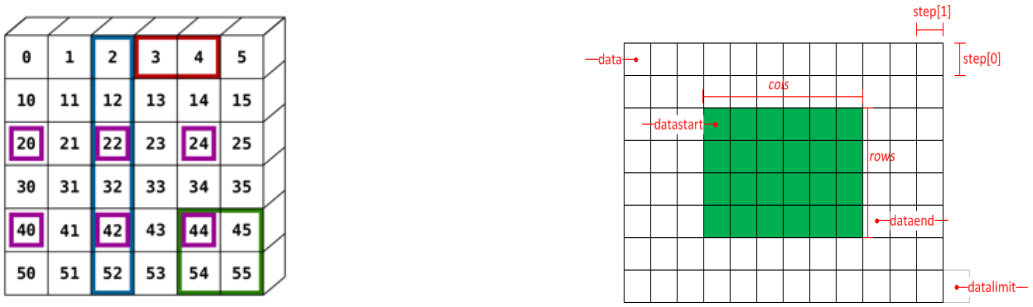
Враховуючи висунуті вимоги щодо алгоритмічного та інструментального забезпечення розробки, система виявлення аномальних сканувань банкнот потребує для реалізації проектування двох зв’язаних інтерфейсів:

– публічного інтерфейсу математичної бібліотеки (для імплементації критеріїв перевірки дійсності та загального методу аналізу сканів);

– користувацького інтерфейсу програмного засобу, що надаватиме оператору візуальний інструментарій для обробки, представлення та візуальної оцінки якості зображень.

Розробку математичної бібліотеки здійснено за модульним підходом, із поділом реалізації кожного типу математичних задач на окремі незв’язні простори імен, що перетинаються спільними структурами даних такими, як матриці. Отже, у контексті вирішуваної задачі необхідно визначити два інтерфейси: матриць для організації обміну даними між компонентами та моделей розпізнавання аномалій.

На рисунку 3.2 показано запропонований принцип проектування високорівневого відображення даних – матриці. Задля уникнення дублювання даних, зайвих копіювань лише через несумісність старого інтерфейсу та перенесення даних в підходящу оболонку, аби тільки подолати дану проблему, та надмірності архітектури через додавання в неї все нових і нових абстракцій для прийнято рішення узагальнити всі відомі прецеденти роботи з даними в ролі матриць через упровадження відповідної базової сутності.



a) b)

Рисунок 3.2 – Відображення даних за допомогою матриці

Місце матриці та моделі в системі розпізнавання фальшованих банкнот представлене функціональною діаграмою в додатку Б.

Реалізації критеріїв дійсності банкнот наслідуватимуть спільний інтерфейс, який складатиметься з двох методів:

– fit(), прийматиме набір правильних реалізацій сканування у формі прямокутної матриці з колонками в ролі дискрет чи ознак оптичного сигналу та рядками як окремими реалізаціями банкноти та обчислюватиме параметри математичної моделі критерії;

– test(), прийматиме набір неперевірених реалізацій банкнот та виконуватиме оцінку їх подібності до математичної моделі повертаючи набір значень по одному для кожної реалізації.

Такий підхід забезпечить взаємозамінність методів виявлення аномалій і розширюваність бібліотеки в разі знаходження нових критеріїв для перевірки. Архітектура спроектованої кросплатформеної бібліотеки у вигляді об’єктної моделі представлена в додатку В.

Користувацький інтерфейс побудовано за архітектурним шаблоном MVC, як показано на рисунку 3.3.

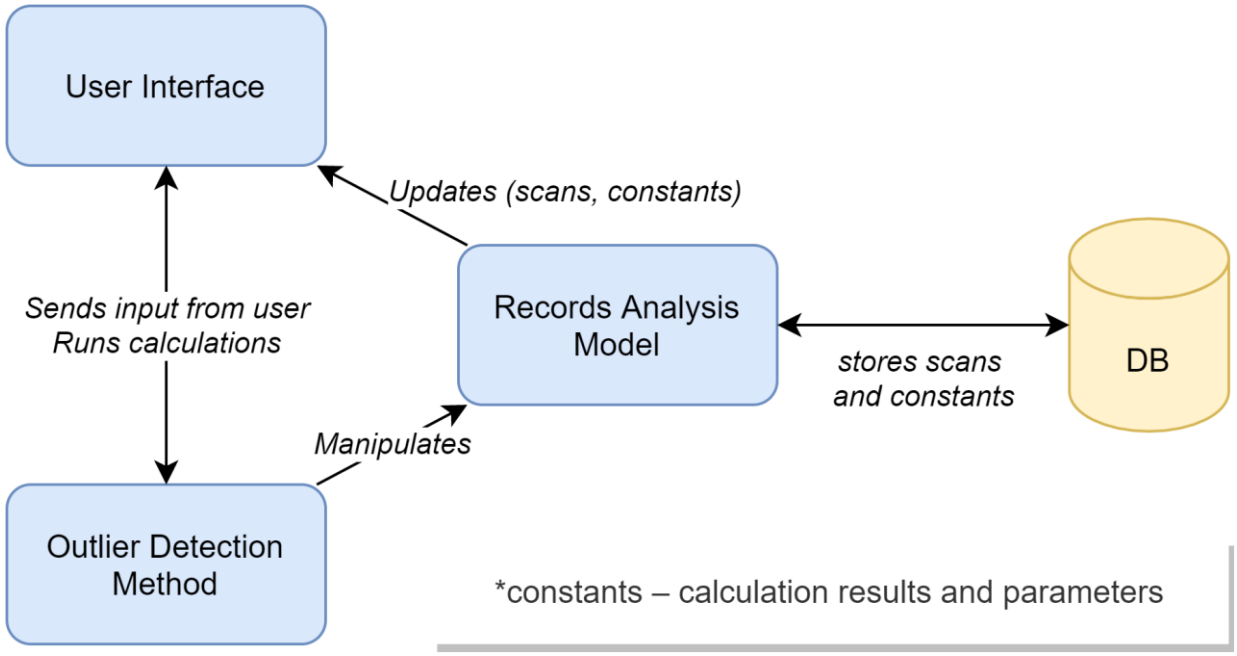


Рисунок 3.3 – Архітектура користувацького інтерфейсу аналізатора банкнот

Обраний дизайн реалізує абстракцію системи в трьох сутностях: представлення, моделі даних та управління. В контексті вирішуваної задачі, модель даних представлятиме інтерфейс до бази даних результатів сканування. Виходячи з того, що збереження колекції сканів відбувається у спосіб запису числових потоків у розмічений бінарний файл, їх зчитування відбуватиметься через додавання адресного простору файлу в оперативну пам’ять замість резервування пам’яті під кожен скан безпосередньо. Такий метод відображення файлів, або file-mapping, дозволяє працювати з сегментами файлів як із масивами елементів заданої розмірності з читанням-записом даних із файлу тільки в разі прямого запиту користувацького коду. Цей підхід у рази швидший за читання вмісту файлу в оперативну пам’ять та є thread-safe, що залишає можливість упровадження багатопотокового виконання в майбутньому. Принципова схема методу файлових відображень представлена на рисунку 3.4.

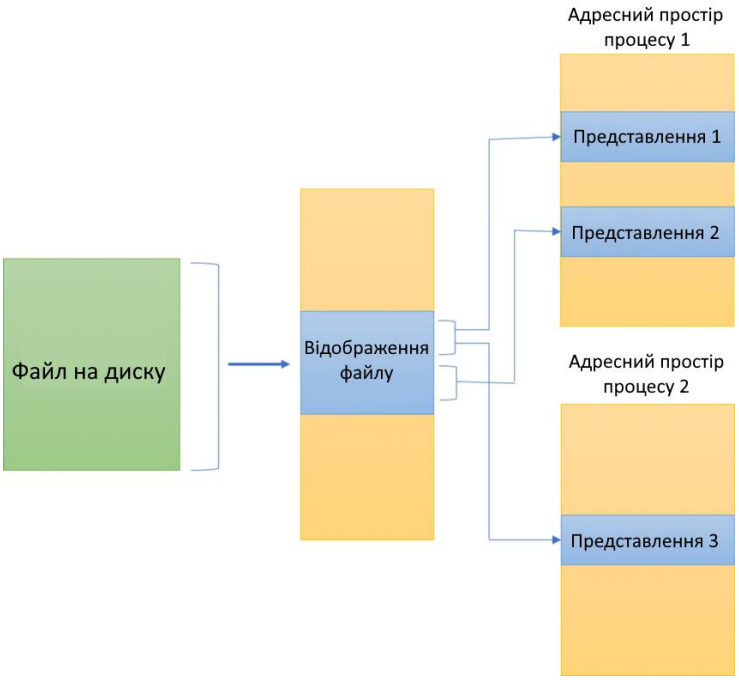


Рисунок 3.4 – Механізм операційної системи відображення файлів або їх частин в адресному просторі процесів

На основі відображення файлів передбачено управління базою даних сканів для роботи з великими масивами даних реалізацій сканування. Відповідно до властивості цього підходу додавати область файлу як представлення у вигляді віртуальної пам’яті керуючого процесу в публічному інтерфейсі математичної бібліотеки враховано можливість створення об’єктів на основі сегментів пам’яті або вказівників.

Відомо, що матричні операції потребують великих обсягів обчислювальних ресурсів комп’ютера. Для оптимізації кінцевого застосунку прийнято рішення в якості складової компоненту моделі тримати поточний стан обчислень та проводити серіалізацію невикористовуваних даних. Таким чином у разі взаємодії користувача з інтерфейсом у межах одного типу банкноти побудована математична модель його правильної форми зберігатиметься в оперативній пам’ять доки не буде здійснено перехід до іншого класу зображень. Тоді неактуальні дані буде серіалізовано в зарезервований сегмент конфігураційного файлу, а нові – зчитано з нього в разі наявності.

Отже, обраний шаблон проектування задовільняє всім релевантним вимогам до даного програмного забезпечення і може бути використаний в розробці.

3.4 Рекомендації з користування програмним засобом

Розроблений програмний засіб є віконним застосунком, який складається з одного основного вікна та двох діалогових. Основне представлення програми реалізує основну методологію виявлення аномалій на реалізаціях банкнот. Воно складається з трьох принципових частин.

До першої групи елементів належать засоби візуального огляду результатів сканування. Вони представлені монохромними зображеннями на рисунку 3.5:

– скану поточного запису, який відповідає обраному кольору;

– зображення математичного очікування результату сканування складене з усереднених значень точок варіаційних коридорів обчислених для кожного треку правильної вибірки;

– зображення різниці між окремою реалізацією – поточним сканом – банкноти та її математичним очікуванням.

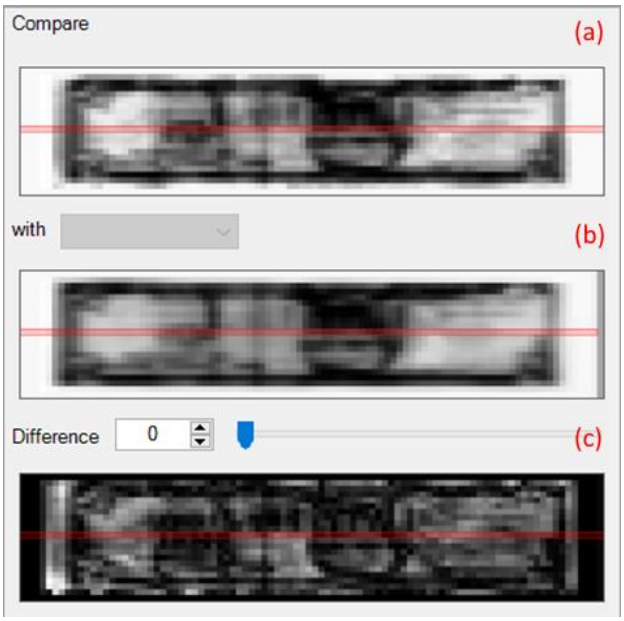
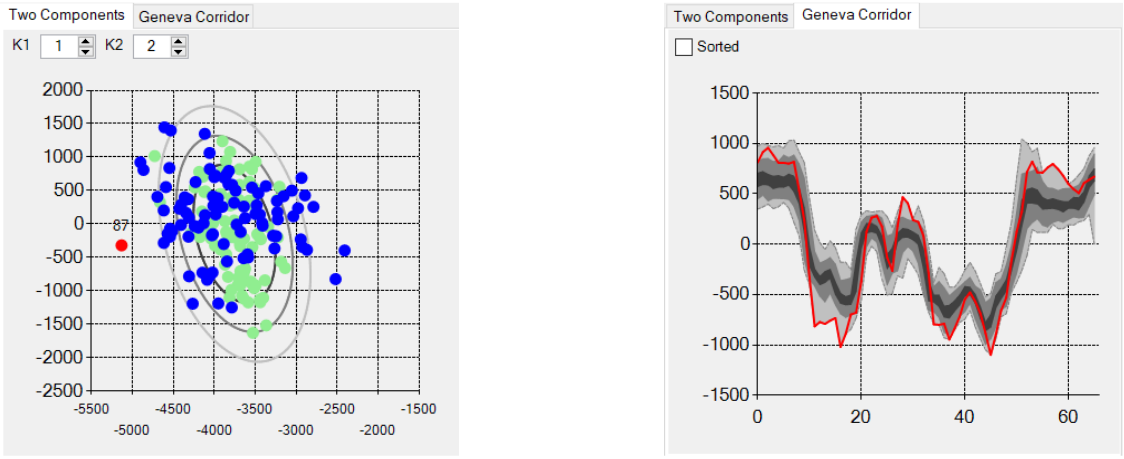


Рисунок 3.5 – Засоби візуального огляду результатів сканування банкнот: зображення вибраної банкноти (a), її математичне очікування (b) та різниця між ними (c)

До другої групи належать засоби аналізу просторових співвідношень результатів сканування, отримані методом головних компонент, та відповідності оптичного сигналу його математичному очікуванню в ролі варіаційного коридору та математичного очікування. Вони показані на рисунку 3.6.



a) b)

Рисунок 3.6 – Елементи інтерфейсу користувача: (а) двовимірне представлення для оцінки просторових співвідношень між елементами множини реалізацій банкнот; (б) варіаційний коридор правильного сигналу

На основі графіків рисунку 3.6 користувачу необхідно здійснити попередній огляд результатів сканування та визначити, які екземпляри банкнот вважати правильними, а які фальшованими чи підозрілими. Вибір попереднього складу тренувальної вибірки критеріїв дійсності купюр рекомендується виконувати на основі припущення про щільність подібних реалізацій аналогічно до кластерного аналізу на основі щільності спостережень. Такі записи мало відрізняються відносно одне одного та мають знаходитися близько, утворюючи групи точок. У разі невідповідності сигналу його варіаційному коридору він виходитиме за межі допустимої області – це свідчитиме про аномальний характер сканування. Тоді банкнота вважається підозрілою та потребуватиме додаткового дослідження чи видалення з тренувальної вибірки.

Третю групу елементів інтерфейсу складає стовпчиковий графік для відображення результатів оцінки аномальності на основі розробленого алгоритму. Він має вигляд, як показано на рисунку 3.7.

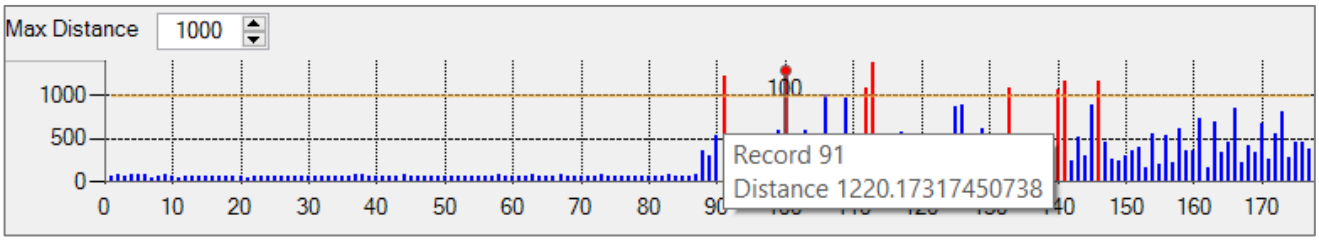


Рисунок 3.7 – Графік результатів перевірки реалізацій сканування банкнот гіпотезі про відповідність математичній моделі

Як видно з рисунку, стовпчики, яким відповідають оцінки вищі за граничне значення, є аномальними банкнотами. Такі екземпляри мають сигнал, що значно відхиляється від допустимої області. Стовпчиковий графік призначений для оцінки якості даних вибірки. У разі відсутності критичних точок серед отриманих оцінок задача забезпечення якості даних для обчислення моделей розпізнавання фальшованих банкнот вважається виконаною.

Загальний вигляд головного вікна аналізатора даних сканування банкнот наведено на рисунку 3.8.

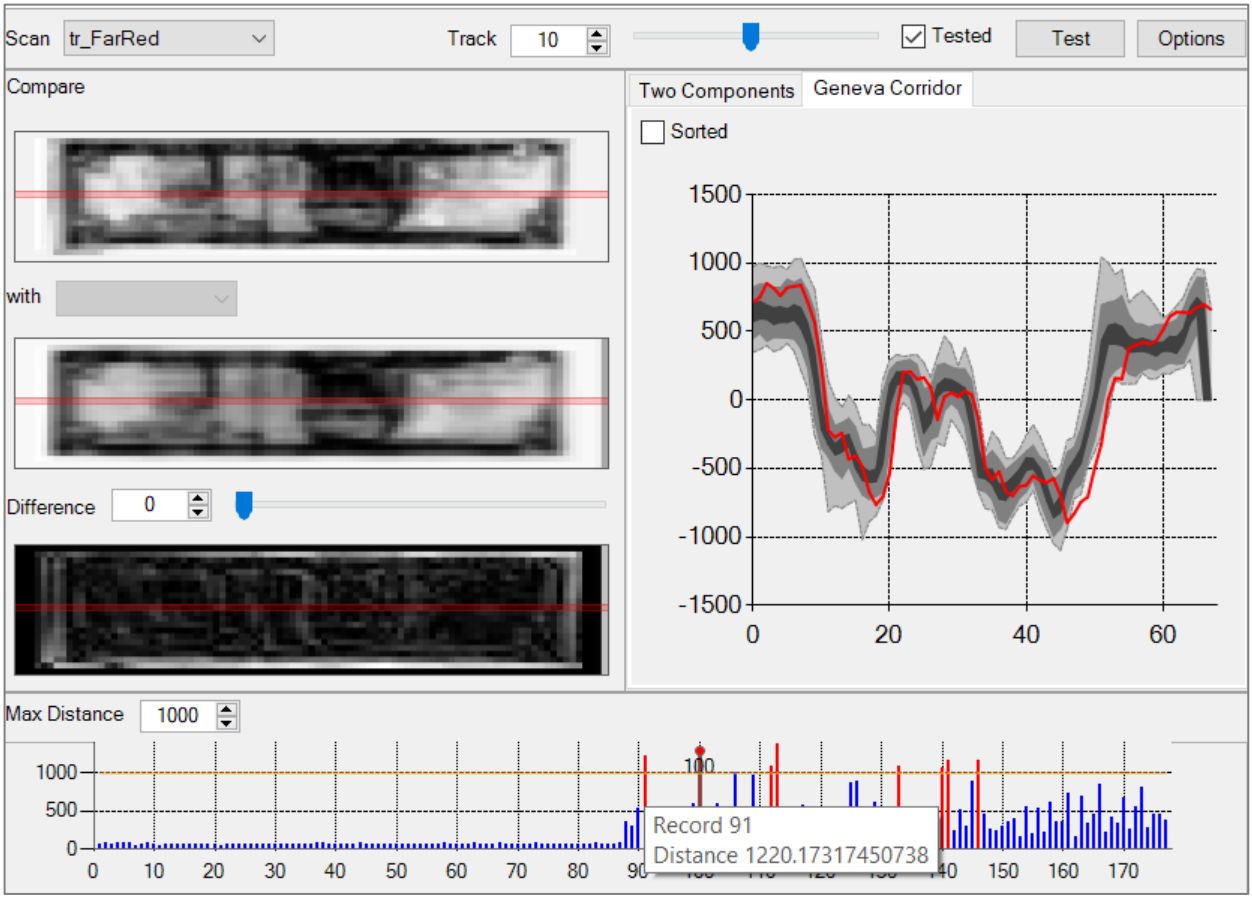


Рисунок 3.8 – Головне вікно застосунку

Взаємодію користувача з інтерфейсом організовано в напрямку згори вниз зліва направо, що є найбільш інтуїтивним способом роботи з додатком. Для переходу між треками сканів достатньо навести вказівник миші на одне із трьох зображень банкноти та клікнути в потрібне місце. Далі застосунок оновиться відповідно до треку, зображеного в місці натискання користувача. Зміну трека також можна здійснити за допомогою смуги прокрутки, розміщеної по центру в верхній частині головного вікна, або шляхом введення номеру сенсора в числовому полі. Для зміни кольору, який перевіряється, необхідно відкрити випадне мені зліва вгорі вікна.

Визначення тренувальної вибірки відбувається активацією діалогового вікна при натисканні кнопки «Options». В разі кліка по зазначеному елементу відкриється форма, як показано на рисунку 3.9. Відповідно своїх спостережень та інформації про зміст файлів даних користувач може обрати записи, які вважає правильними.

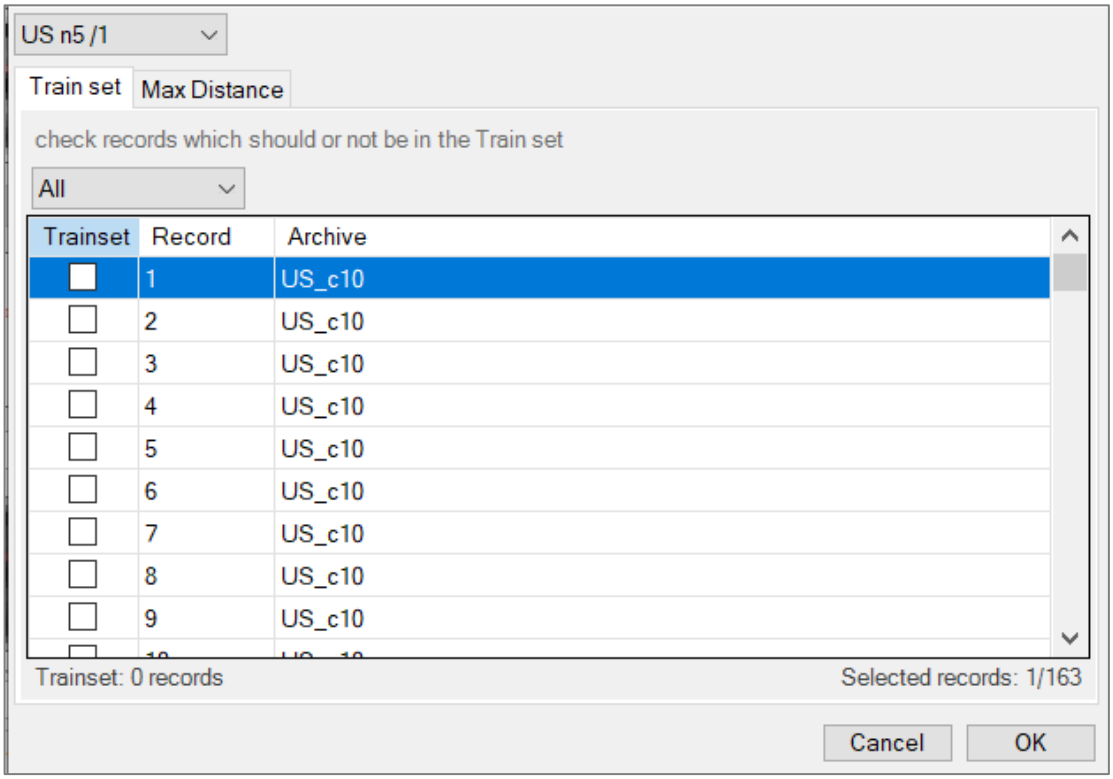


Рисунок 3.9 – Діалогове вікно для визначення правильних реалізацій банкноти

Тестування всієї бази даних відбувається за допомогою діалогового вікна «Test», яке відкривається аналогічно вікну «Options» натисканням на однойменну кнопку. Представлення форми показане на рисунку 3.10.

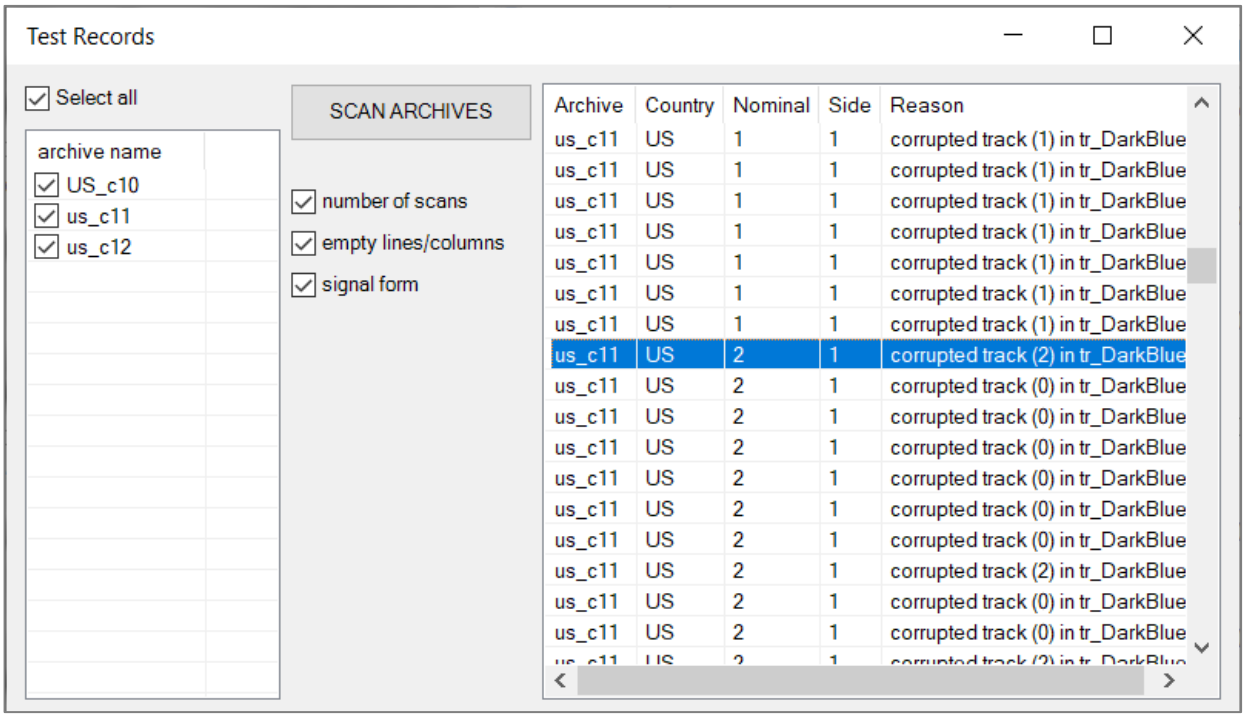


Рисунок 3.10 – Вікно автоматичного тестування результатів сканування банкнот на відповідність критеріям дійсності

Для запуску процедури тестування необхідно обрати файли даних, зміст яких піддасться аналізу, тип помилки, яку необхідно ідентифікувати та натиснути на кнопку «SCAN ARCHIVES». Далі програма запустить процес перевірки обраних наборів даних та додаватиме в таблицю справа посилання на всі екземпляри, які з якоїсь причини суперечать гіпотезі про правильність сканування.

Після закриття застосунку параметри обчислених математичних моделей правильних банкнот будуть збережені до відповідного файлу і роботу з даними можна буде продовжити на тому ж місці після чергового запуску застосунку.

Висновок до розділу

В розділі описано імплементацію програмного забезпечення системи виявлення аномальних реалізацій сканування банкнот оптичними сенсорами на прикладі математичної бібліотеки «OLLA» та аналітичної програми з графічним інтерфейсом «Records Analyzer». Було визначено технічні та функціональні вимоги до кінцевого застосунку, наведено архітектуру програмного забезпечення –

виконано опис діаграми класів. Виконано опис рекомендацій з користування додатком на прикладах роботи з ним.

ВИСНОВКИ

У роботі було розглянуто питання пов’язане з забезпеченням якості програмного забезпечення детекторів банківських білетів, а саме зі збором тренувальних даних для побудови моделей розпізнавання фальшованих купюр. Було проаналізовано структуру результатів сканування та визначено головні характеристики дійсних банкнот, а також досліджено основні прояви аномальності, які властиві підробкам.

На основі вивчення процесу введення та сканування грошей валідаторами виявлено основні причини появи спотворень на даних сканування та виконано їх класифікацію. Аналіз особливостей ручного введення та обробки даних показав не ефективність прийнятих ручних методів та відсутність програмних засобів пошуку спотворених екземплярів для тренування моделей розпізнавання та дозволив сформулювати мету дослідження: покращити якість розпізнавання фальшивих банкнот за рахунок автоматизації процесу виявлення пошкоджених сканів у тренувальних даних. Для досягнення мети поставлена задача: розробити критерії перевірки та програмні засоби для реалізації методу виявлення аномалій у сканах банкнот для забезпечення якості даних, що використовуються у виробництві для отримання результуючих моделей розпізнавання підробок, а саме пошуку та усунення підозрілих елементів з вибірки на основі кількісної та візуальної оцінки їх спотворень.

У результаті дослідження предметної області та вивчення існуючих методів ідентифікації викидів: їх видів, переваг та недоліків кожного – було зроблено висновок про необхідність побудови нових критеріїв контролю якості даних придатних до застосування в умовах різної кількості змінних спостережень та стійких до прокляття вимірності, вплив якого зростає пропорційно кількості ознак. Для перевірки гіпотези про дійсність екземпляру банкноти у її дискретній реалізації розроблено новий спосіб аналізу зображень, заснований на поєднанні чотирьох емпіричних та двох теоретичних критеріїв правильності сканування. Емпіричні критерії побудовано з окремих характеристик оптичного сигналу. Вони є

некорельованими та реалізують інтегральну оцінку відхилення окремого сканування від релевантних йому коливань, рівня та форми. Теоретичні критерії базуються на загальному припущенні щодо походження варіацій у даних з нормального розподілу. Їх оцінки аномальності є ймовірнісними та дозволяють класифікувати дані за ступенем підозрілості. Емпіричне дослідження запропонованих критеріїв проводилось за показниками точності та прогностичної значущості. Для цього було згенеровано тестові дані, які відповідають основним проявам аномальної поведінки підробок, та сформовано вибірку справжніх сканів. Виявилося, що кожен із критеріїв окремо є не ефективним через упередженість критичної області в бік підробок (коли частина оригіналів розпізнана помилково) або дійсних банкнот. Напроти, ефективність методу на основі поєднання емпіричних та теоретичних оцінок – 98% релевантних результатів – дозволила вирішити поставлену задачу.

Для швидкого отримання моделі розпізнавання вибір тренувальних даних побудовано на аналізі просторових співвідношень отриманих з проекцій сканів у двовимірний простір як лінійної комбінації точок сигналу. Таке рішення дозволило попередньо оцінити подібність даних та визначити серед них правильні екземпляри без необхідності ретельної візуальної перевірки кожного сканування.

Для реалізації системи виявлення фальшованих банкнот створено програмну реалізацію математичної бібліотеки. Її розробку виконано мовою C++ із застосуванням бібліотек: «BLAS» базових операцій лінійної алгебри та «GSL» відкритої математичної бібліотеки низького рівня. Для читання файлів даних з результатами сканування банкнот та оптимального використання ресурсів пристрою кінцевого користувача застосовується механізм відображення файлів «file-mapping». Для роботи з даними цільового застосування розроблено інтерфейс вищого рівня «Matrix», який реалізує представлення навколо динамічної пам’яті як матриці, вектори та скаляри і підтримує всі матричні та векторні операції необхідні для імплементації методу виявлення аномалій. Проведено експериментальне дослідження, яке показало переваги статичної бібліотеки: за швидкістю компіляції

– у 3 рази в порівнянні з повністю шаблонними бібліотеками, за швидкістю обчислень – на 10% в порівнянні з бібліотеками, які використовують векторизацію матричного множення. На основі розробленої бібліотеки виконано реалізацію застосунку, який інтегровано в систему навчання моделей розпізнавання фальшованих банкнот. Створено керівництво користувача та описано методологію налаштування програми.

Отримані критерії виявлення аномалій та опис програмного забезпечення викладено в статті, яку було опубліковано в науковому збірнику матеріалів у напрямку програмної інженерії.

Наукова новизна результатів одержаних у магістерській дисертації полягає в тому, що:

– запропоноване алгоритмічне забезпечення для виявлення аномальних реалізацій сканування банкнот засноване на чотирьох емпіричних та двох теоретичних критеріях нормальності даних;

– розроблено програмний інтерфейс та архітектуру бібліотеки, яка реалізує методи виявлення аномалій та оболонку для низькорівневих структур даних, сканів, і процедур для роботи з ними у вигляді двовимірних матриць.

Практичне значення отриманих результатів полягає в тому, що розроблено програмне забезпечення, яке реалізує експериментально обґрунтований алгоритм виявлення аномалій на реалізаціях сканування банківських білетів оптичними сенсорами, що може застосовуватися в автоматизованих та напівавтоматизованих системах обробки великих масивів даних.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1) Чубенко А. Г., Лошицький М. В., Павлов Д. М. Термінологічний словник з питань запобігання та протидії легалізації (відмиванню) доходів, одержаних злочинним шляхом, фінансуванню тероризму, фінансуванню розповсюдження зброї масового знищення та корупції. Ваіте, 2018. 826 с. URL: https://www.osce.org/files/190125_1607_ch_ALL_Inet.pdf (дата звернення: 15.04.2022).

2) Ионов В. М. Обработка наличности. Банковская техника и технологии. Издательская группа «БДЦ-пресс», 2001. 268 с.

3) Barchard K. A., Pace L. A. Preventing human error: the impact of data entry methods on data accuracy and statistical results. Computers in human behavior. 2011. Vol. 27, no. 5. pp. 1834–1839. DOI: https://doi.org/10.1016/j.chb.2011.04. 004 (дата звернення: 15.04.2022).

4) A comparison of error detection rates between the reading aloud method and the double data entry method / M. Kawado et al. Controlled clinical trials. 2003. Vol. 24, no. 5. pp. 560–569. DOI: https://doi.org/10.1016/s0197-2456(03)00089-8 (дата звернення: 15.04.2022).

5) Ben-Gal I. Outlier Detection. Data Mining and Knowledge Discovery Handbook. 2005. С. 131–146. URL: https://doi.org/10.1007/0-387-25465-X_7 (дата звернення: 15.04.2022).

6) Barnett V., Lewis T. Outliers in Statistical Data. 3-тє вид. J. Wiley & Sons, 1994. 608 с. URL: https://doi.org/10.1002/bimj.4710370219 (дата звернення: 15.04.2022).

7) Johnson R. A. Applied Multivariate Statistical Analysis. 5-те вид. Prentice-Hall, 2002. 794 с. URL: http://shorturl.at/pKRW1 (дата звернення: 15.04.2022).

8) Aggarwal C. C. Outlier Analysis. 2-ге вид. Springer, 2017. 481 с. URL: https://doi.org/10.1007/978-3-319-47578-3 (дата звернення: 15.04.2022).

9) Hodge V., Austin J. A Survey of Outlier Detection Methodologies. Artificial Intelligence Review. 2004. T. 22. C. 85–126. URL: https://doi.org/10.1023/B:AIRE.0000045502.10941.a9 (дата звернення: 15.04.2022).

10) Sikder M. N. K., Feras A. B. Outlier Detection using AI: A Survey. ArXiv. 2021. Abs/2112.00588. URL: https://doi.org/10.48550/arXiv.2112.00588 (дата звернення: 15.04.2022).

11) Tang, J Enhancing Effectiveness of Outlier Detections for Low Density Patterns. In M.-S. Chen, P. S. Yu, & B. Liu (Eds.), Advances in Knowledge Discovery and Data Mining. 2002. 535-548. Springer Berlin Heidelberg.

12) Kriegel, H., Kröger, P., & Zimek, A. Outlier detection techniques. 2009. Proc. Tutorial KDD, 1–10, URL: https://www.dbs.ifi.lmu.de/~zimek/publications/KDD 2010/kdd10-outlier-tutorial.pdf (дата звернення: 15.04.2022)

13) Ruff L., Vandermeulen R.A., Gornitz N. Deep One-Class Classification. Proceedings of the 35th International Conference on Machine Learning. 80-е вид. PMLR. 2018. С. 4393–4402. URL: http://proceedings.mlr.press/v80/ruff18a/ ruff18a.pdf (дата звернення: 15.04.2022)

14) Yang, X., Latecki, L. J., Pokrajac, D. Outlier Detection with Globally Optimal Exemplar-Based GMM. In Proceedings of the 2009 SIAM International Conference on Data Mining. 2009. SDM. С. 145–154. URL: https://doi.org/10.1137/ 1.9781611972795.13 (дата звернення: 15.04.2022)

15) Tang, X., Yuan, R., Chen, J. Outlier Detection in Energy Disaggregation Using Subspace Learning and Gaussian Mixture Model. 8-ме вид. International Journal of Control and Automation. 2015. С. 161–170. URL: https://www.earticle.net/Article/A253913 (дата звернення: 15.04.2022)

16) Saha, B. N., Ray, N., & Zhang, H. Snake Validation: A PCA-Based Outlier Detection Method. 16-те вид. IEEE Signal Processing Letters. 2009 С. 549–552. URL: https://doi.org/10.1109/LSP.2009.2017477 (дата звернення: 15.04.2022)

17) Satman, M. H. A New Algorithm for Detecting Outliers in Linear Regression. 2-ге вид. International Journal of Statistics and Probability. 2013. С. 101. URL: https://doi.org/10.5539/ijsp.v2n3p101 (дата звернення: 15.04.2022)

18) He X., Niyogi P. Locality Preserving Projections. 16-те вид. Advances in Neural Information Processing Systems. 2003. MIT Press. URL: https://proceedings.neurips.cc/paper/2003/file/d69116f8b0140cdeb1f99a4d5096ffe4-Paper.pdf

19) Anomaly Detection Software. URL: https://avora.com/

20) Incident Detection and Correlation. URL: https://www.anodot.com/incident-detection/

21) Pedregosa, F., Varoquaux, G., Gramfort. Scikit-learn: Machine Learning in Python. 12-те вид. Journal of Machine Learning Research. С. 2825–2830. 2011. URL: http://jmlr.org/papers/v12/pedregosa11a.html

22) Zhao Y., Nasrullah Z. Zheng L. PyOD: A Python Toolbox for Scalable Outlier Detection. abs/1901.01588 вид. 2019. CoRR. URL: http://arxiv.org/abs/1901.01588

23) Verboven S., Hubert M. LIBRA: a MATLAB Library for Robust Analysis. 75-те вид. Chemometrics and Intelligent Laboratory Systems. С. 127-136. 2004. URL: https://doi.org/10.1016/j.chemolab.2004.06.003

24) Manly B. F. J., Navarro Alberto J. A. Multivariate statistical methods. Fourth edition / Bryan F.J. Manly and Jorge A. Navarro Alberto. : Chapman and Hall/CRC, 2016. URL: https://doi.org/10.1201/9781315382135 (дата звернення: 23.05.2022).

25) Jolliffe I. Principal component analysis. New York : Springer-Verlag, 2002. URL: https://doi.org/10.1007/b98835 (дата звернення: 24.05.2022).

26) Guennebaud G. Eigen: a C++ linear algebra library. Smart Libraries for Computer Graphics. 2013. CGLibs. URL: https://downloads.tuxfamily.org/eigen/eigen_CGLibs_Giugno_Pisa_2013.pdf (дата звернення: 24.05.2022).

27) Gough B. GNU scientific library reference manual - third edition. 3-тє вид. Network Theory Ltd., 2009. 592 с. URL: https://www.gnu.org/software/gsl/doc/latex/gsl-ref.pdf (дата звернення: 24.05.2022).

28) van de Geijn R. BLAS (basic linear algebra subprograms). Springer, Boston, С. 157–164. URL: https://doi.org/10.1007/978-0-387-09766-4_84

ДОДАТОК А

Лістинг коду ПЗ

```
namespace linalg {

enum class Access { kView, kCopy, kManage };

using Index = long long;

template<typename _Tp> class Mat
{
public:
 using Scalar = _Tp;

private:
 std::shared_ptr<Scalar[]> _data;
 Index _rows;
 Index _cols;
 Index _cbeg;
 Index _rbeg;
 Index _stride;
 Index _cstride;
 Scalar* _datastart;
 Scalar* _dataend;

 struct Iterator
 {
    using difference_type = Index;
     using iterator_category = std::random_access_iterator_tag;
     using pointer        = Scalar *;
     using reference         = Scalar &;
     using value_type     = Scalar;
    //
    Mat* _mat;
    Index _pos;

    Iterator() = default;
    Iterator(Mat* mat, Index pos)
       : _mat(mat),_pos(pos) {}
    //
    inline reference   operator*() const          { return (*_mat)(_pos); }
    inline pointer     operator->()            { return &(*_mat)(_pos); }
    inline reference   operator[](Index i)           { return (*_mat)(_pos + i); }
```

```
    //
    inline Iterator&   operator++()             { return ++_pos, *this; }
    inline Iterator    operator++(int)             { return { _mat, _pos++ }; }
     inline Iterator&  operator+=(difference_type ofs) { return _pos += ofs, *this;
}
         inline Iterator& operator--()             { return --_pos, *this; }
         inline Iterator     operator--(int)             { return { _mat, _pos-- };
}
    inline Iterator&   operator-=(difference_type ofs) { return _pos -= ofs, *this;
}
    //
    friend
     inline Iterator      operator+(Iterator const& it, difference_type ofs)   {
return { it._mat, it._pos + ofs }; };
    friend
     inline Iterator      operator-(Iterator const& it, difference_type ofs)   {
return { it._mat, it._pos - ofs }; };
    //
    inline difference_type operator-(Iterator const& rhs) const { return _pos -
rhs._pos; }
    //
    inline bool operator> (Iterator const& rhs) const { return _pos > rhs._pos; }
    inline bool operator< (Iterator const& rhs) const { return _pos < rhs._pos; }
    inline bool operator!=(Iterator const& rhs) const { return _pos != rhs._pos; }
    inline bool operator==(Iterator const& rhs) const { return _pos == rhs._pos; }
};

struct ConstIterator
{
    using difference_type = Index;
     using iterator_category = std::random_access_iterator_tag;
     using pointer        = Scalar const*;
     using reference         = Scalar const&;
     using value_type     = Scalar;
    //
    Mat const* _mat;
    Index _pos;

    ConstIterator() = default;
    ConstIterator(Mat const* mat, Index pos)
       : _mat(mat),_pos(pos) {}
    //
    inline reference      operator*() const           { return (*_mat)(_pos); }
    inline pointer        operator->()             { return &(*_mat)(_pos); }
```

```
    inline reference      operator[](Index i)           { return (*_mat)(_pos + i);
}
    //
    inline ConstIterator& operator++()             { return ++_pos, *this; }
    inline ConstIterator  operator++(int)             { return { _mat, _pos++ }; }
     inline ConstIterator&   operator+=(difference_type ofs) { return _pos += ofs,
*this; }
         inline ConstIterator&  operator--()            { return --_pos, *this; }
         inline ConstIteratoroperator--(int)            { return { _mat, _pos-- };
}
    inline ConstIterator& operator-=(difference_type  ofs)  {  return  _pos  -=  ofs,
*this; }
    //
    friend
     inline ConstIterator    operator+(ConstIterator const& it, difference_type ofs)
{ return { it._mat, it._pos + ofs }; };
    friend
     inline ConstIterator    operator-(ConstIterator const& it, difference_type ofs)
{ return { it._mat, it._pos - ofs }; };
    //
    inline difference_type operator-(ConstIterator const& rhs) const { return _pos -
rhs._pos; }
    //
    inline bool operator> (ConstIterator const& rhs) const { return _pos > rhs._pos;
}
    inline bool operator< (ConstIterator const& rhs) const { return _pos < rhs._pos;
}
    inline bool operator!=(ConstIterator const& rhs) const { return _pos != rhs._pos;
}
    inline bool operator==(ConstIterator const& rhs) const { return _pos == rhs._pos;
}
};

public:
Mat();
Mat(Mat const& oth, Access mode = Access::kView);
Mat(Mat && oth) = default;
Mat(Index size);
Mat(Index rows, Index cols);
Mat(Scalar* data, Index size, Access owner = Access::kView);
Mat(Scalar* data, Index rows, Index cols, Access owner = Access::kView);
~Mat();

Mat& operator=(Mat const& oth) = default;
Mat& operator=(Mat && oth) = default;
```

```
Scalar& operator()(Index i);
Scalar const& operator()(Index i) const;
Scalar& operator()(Index row, Index col);
Scalar const& operator()(Index row, Index col) const;

Scalar& at(Index i);
Scalar const& at(Index i) const;
Scalar& at(Index row, Index col);
Scalar const& at(Index row, Index col) const;

Index rows() const;
Index cols() const;
Index stride() const;
Index size() const;
Index total() const;

void resize(Index rows, Index cols);
template<typename T2> void copyTo(Mat<T2>& oth) const;

Scalar const* datastart() const;
Scalar const* dataend() const;

Scalar* data();
Scalar const* data() const;

Scalar* ptr(Index row);
Scalar const* ptr(Index row) const;

Scalar& operator[](Index i);
Scalar const& operator[](Index i) const;

Iterator   begin()   { return { this, 0 }; }
Iterator   end()       { return { this, size() }; }
ConstIterator begin() const { return { this, 0 }; }
ConstIterator end()   const { return { this, size() }; }

std::reverse_iterator<Iterator>       rbegin()   {                          return
std::make_reverse_iterator(end()); }
    std::reverse_iterator<Iterator>      rend()       {                    return
std::make_reverse_iterator(begin()); }
std::reverse_iterator<ConstIterator>      rbegin()      const      {      return
std::make_reverse_iterator(end()); }
```

```
    std::reverse_iterator<ConstIterator>    rend()            const    {    return
std::make_reverse_iterator(begin()); }

Mat row(Index r) const;
Mat rowRange(Index startrow, Index endrow) const;
Mat rowRange(std::vector<Index> const& rows) const;
Mat col(Index c) const;
Mat colRange(Index startcol, Index endcol) const;
Mat colRange(std::vector<Index> const& cols) const;
Mat block(Index startrow, Index startcol, Index endrow, Index endcol) const;
Mat range(Index beg, Index end) const;

Mat copy() const;
Mat inv() const;
Mat abs() const;
Mat sqrt() const;
Mat householderQr() const;
Mat singularVectors() const;
std::pair<Mat<double>, Mat<double>> eig(bool symmetric = false) const;

Mat vectorRow() const;
Mat vectorCol() const;
Mat transpose() const;
Mat diagonal() const;

Mat asDiagonal() const;

static Mat Rand(Index size);
static Mat Rand(Index rows, Index cols);
static Mat Randi(int max, Index size);
static Mat Randi(int max, Index rows, Index cols);
static Mat Ones(Index size);
static Mat Ones(Index rows, Index cols);
static Mat Zeros(Index size);
static Mat Zeros(Index rows, Index cols);
static Mat Eye(Index rows, Index cols);

template<typename T2> Mat mul(Mat<T2> const& rhs);
template<typename T2> _Tp dot(Mat<T2> const& rhs);

Scalar max() const;
Scalar min() const;
Scalar sum() const;
Scalar prod() const;
```

```
Scalar mean() const;

Mat colsMean() const;
Mat rowsMean() const;
Mat rowsAdd(Mat const& oth) const;
Mat rowsSub(Mat const& oth) const;

bool operator!=(Mat const& oth) const;
bool operator==(Mat const& oth) const;

Mat operator+(Scalar scalar) const;
Mat operator-(Scalar scalar) const;
Mat operator*(Scalar scalar) const;
Mat operator/(Scalar scalar) const;
Mat operator^(Scalar scalar) const;

Mat& operator= (Scalar scalar);
Mat& operator+=(Scalar scalar);
Mat& operator-=(Scalar scalar);
Mat& operator*=(Scalar scalar);
Mat& operator/=(Scalar scalar);
Mat& operator^=(Scalar scalar);

template<typename T2> Mat& operator-=(Mat<T2> const& oth)
{
   for(auto i = 0; i < size(); ++i)
      (*this)(i) -= T2(oth(i));
   return *this;
}

template<typename T2> Mat& operator+=(Mat<T2> const& oth)
{
   for(auto i = 0; i < size(); ++i)
      (*this)(i) += T2(oth(i));
   return *this;
}

template<typename T2> Mat<T2> cast() const
{
   auto mat = Mat<T2>(_rows, _cols);

   for(Index i = 0; i < size(); ++i)
      mat(i) = T2((*this)(i));
```

```
    return mat;
}
};

using MatXd = Mat<double>;
using MatXi = Mat<int>;

template <typename _Tp> std::ostream  & operator<<(std::ostream  & os, Mat<_Tp> const
& mat)
{
for(auto ri = 0; ri < mat.rows(); ++ri)
{
    for(auto ci = 0; ci < mat.cols(); ++ci)
    {
       os << mat(ri, ci) << '\t';
    }
    os << '\n';
}

return os;
}
template <typename _Tp> std::ofstream & operator<<(std::ofstream & os, Mat<_Tp> const
& mat)
{
Index rows = mat.rows(), cols = mat.cols();

os.write(reinterpret_cast<const char*>(&rows), sizeof(Index));
os.write(reinterpret_cast<const char*>(&cols), sizeof(Index));

for(auto ri = 0; ri < rows; ++ri)
    os.write(reinterpret_cast<const        char*>(mat.ptr(ri)),        cols        *
sizeof(linalg::Mat<_Tp>::Scalar));

return os;
}
template <typename _Tp> std::ifstream & operator>>(std::ifstream & os, Mat<_Tp> &
mat)
{
Index rows, cols;

os.read(reinterpret_cast<char*>(&rows), sizeof(Index));
os.read(reinterpret_cast<char*>(&cols), sizeof(Index));
```

```
mat.resize(rows, cols);

for(auto ri = 0; ri < rows; ++ri)
   os.read(reinterpret_cast<char*>(mat.ptr(ri)),               cols               *
sizeof(linalg::Mat<_Tp>::Scalar));

return os;
}

template <typename _Tp> Mat<_Tp> operator+(_Tp scalar, Mat<_Tp> const & rhs)
{
auto mat = linalg::Mat<_Tp>(rhs, linalg::Mat<_Tp>::Access::kCopy);

for(auto i = 0; i < mat.size(); ++i)
   mat(i) += _Tp(scalar);

return mat;
}
template <typename _Tp> Mat<_Tp> operator-(_Tp scalar, Mat<_Tp> const & rhs)
{
auto mat = linalg::Mat<_Tp>(rhs, linalg::Mat<_Tp>::Access::kCopy);

for(auto i = 0; i < mat.size(); ++i)
   mat(i) -= _Tp(scalar);

return mat;
}
template <typename _Tp> Mat<_Tp> operator*(_Tp scalar, Mat<_Tp> const & rhs)
{
auto mat = linalg::Mat<_Tp>(rhs, linalg::Mat<_Tp>::Access::kCopy);

for(auto i = 0; i < mat.size(); ++i)
   mat(i) *= _Tp(scalar);

return mat;
}
template <typename _Tp> Mat<_Tp> operator/(_Tp scalar, Mat<_Tp> const & rhs)
{
auto mat = linalg::Mat<_Tp>(rhs, linalg::Mat<_Tp>::Access::kCopy);

for(auto i = 0; i < mat.size(); ++i)
   mat(i) /= _Tp(scalar);

return mat;
```

```
}
template <typename _Tp> Mat<_Tp> operator^(_Tp scalar, Mat<_Tp> const & rhs)
{
auto mat = linalg::Mat<_Tp>(rhs, linalg::Mat<_Tp>::Access::kCopy);

for(auto i = 0; i < mat.size(); ++i)
   mat(i) = _Tp(std::pow(scalar,mat(i)));

return mat;
}

template <typename _Tp, typename T2> Mat<_Tp> operator+(Mat<_Tp> const & lhs, Mat<T2>
const & rhs)
{
Mat<_Tp> res = Mat<_Tp>(lhs, Access::kCopy);

for(auto i = 0; i < res.size(); ++i)
{
   res(i) += _Tp(rhs(i));
}

return res;
}
template <typename _Tp, typename T2> Mat<_Tp> operator-(Mat<_Tp> const & lhs, Mat<T2>
const & rhs)
{
Mat<_Tp> res = Mat<_Tp>(lhs, Access::kCopy);

for(auto i = 0; i < res.size(); ++i)
{
   res(i) -= _Tp(rhs(i));
}

return res;
}
template <typename _Tp, typename T2> Mat<_Tp> operator*(Mat<_Tp> const & lhs, Mat<T2>
const & rhs)
{
Mat<_Tp> res = Mat<_Tp>(lhs, Access::kCopy);

for(auto i = 0; i < res.size(); ++i)
{
   res(i) *= _Tp(rhs(i));
}
```

```
return res;
}
}

#define SHPTR_WITH_DEL_NEW(size) std::shared_ptr<Scalar[]>(new Scalar[size])
#define SHPTR_WITH_NO_DEL(p) std::shared_ptr<Scalar[]>(p,[](Scalar *_p){})
#define SHPTR_WITH_DEL_REF(p) std::shared_ptr<Scalar[]>(p)

using namespace linalg;

/**
 Returns a deep copy of the Mat matrix
 */
template<typename _Tp>
gsl_matrix* gsl_matrix_from_Mat(Mat<_Tp> const& mat)
{
gsl_matrix* gsl = gsl_matrix_alloc(mat.rows(), mat.cols());

for(auto ri = 0; ri < mat.rows(); ++ri)
   for(auto ci = 0; ci < mat.cols(); ++ci)
      gsl_matrix_set(gsl,ri,ci,double(mat(ri,ci)));

return gsl;
}

template<typename _Tp>
Mat<_Tp> gsl_matrix_to_Mat(gsl_matrix* gsl, Access mode)
{
Index size = gsl->size1 * gsl->size2;

auto mat_data = new _Tp[size];
auto gsl_data = gsl->data;

for(_Tp* itr = &mat_data[0]; itr < &mat_data[size]; ++itr, ++gsl_data)
   *itr = _Tp(*gsl_data);

auto mat = Mat<_Tp>(mat_data, gsl->size1, gsl->size2, Access::kCopy);

return mat;
}

template<>
```

```
Mat<double> gsl_matrix_to_Mat<double>(gsl_matrix* gsl, Access mode)
{
 auto mat = Mat<double>(gsl->data, gsl->size1, gsl->size2, mode);

 return mat;
}

template<typename _Tp>
Mat<_Tp> gsl_vector_to_Mat(gsl_vector* gsl, Access mode)
{
 Index size = gsl->size;

 auto mat_data = new _Tp[size];
 auto gsl_data = gsl->data;

 for(_Tp* itr = &mat_data[0]; itr < &mat_data[size]; ++itr, ++gsl_data)
    *itr = _Tp(*gsl_data);

 auto mat = Mat<_Tp>(mat_data, gsl->size, Access::kCopy);

 return mat;
}

template<>
Mat<double> gsl_vector_to_Mat<double>(gsl_vector* gsl, Access mode)
{
 auto mat = Mat<double>(gsl->data, gsl->size, mode);

 return mat;
}

template<typename _Tp>
Mat<_Tp> gsl_matrix_complex_to_Mat_real(gsl_matrix_complex* gsl, Access mode)
{
 Index size = gsl->size1 * gsl->size2;

 auto mat_data = new _Tp[size];
 auto gsl_data = gsl->data;

 for(_Tp* itr = &mat_data[0]; itr < &mat_data[size]; ++itr, gsl_data += 2)
    *itr = _Tp(*gsl_data);

 auto mat = Mat<_Tp>(mat_data, gsl->size1, gsl->size2, Access::kCopy);
```

```
return mat;
}

template<typename _Tp>
Mat<_Tp> gsl_vector_complex_to_Mat_real(gsl_vector_complex* gsl, Access mode)
{
Index size = gsl->size;

auto mat_data = new _Tp[size];
auto gsl_data = gsl->data;

for(_Tp* itr = &mat_data[0]; itr < &mat_data[size]; ++itr, gsl_data += 2)
   *itr = _Tp(*gsl_data);

auto mat = Mat<_Tp>(mat_data, gsl->size, Access::kCopy);

return mat;
}

template <typename _Tp>
Mat<_Tp>::Mat()
:_data()
,_rows()
,_cols()
,_cbeg()
,_rbeg()
,_stride()
,_cstride()
,_datastart()
,_dataend()
{
}

template <typename _Tp>
Mat<_Tp>::Mat(Mat const & oth, Access mode)
:_rows(oth._rows)
,_cols(oth._cols)
,_cstride(1)
{
switch(mode)
{
case Access::kCopy:
```

```
{
    _data = SHPTR_WITH_DEL_NEW(_rows * _cols);
    _datastart = &_data.get()[0];
    _dataend = &_datastart[_rows * _cols];

    _cbeg = 0;
    _rbeg = 0;
    _stride = _cols;

    oth.copyTo(*this);
}
break;
case Access::kView:
case Access::kManage:
{
    _data = oth._data;
    _datastart = oth._datastart;
    _dataend = oth._dataend;

    _cbeg = oth._cbeg;
    _rbeg = oth._rbeg;
    _stride = oth._stride;
    _cstride = oth._cstride;
}
break;
}
}

template <typename _Tp>
Mat<_Tp>::Mat(Index size)
:_rows(size)
,_cols(1)
,_cbeg(0)
,_rbeg(0)
,_stride(1)
,_cstride(size)
,_data(SHPTR_WITH_DEL_NEW(size))
,_datastart(&_data.get()[0])
,_dataend(&_datastart[size])
{
}

template <typename _Tp>
```

```
Mat<_Tp>::Mat(Index rows,Index cols)
:_rows(rows)
,_cols(cols)
,_cbeg(0)
,_rbeg(0)
,_stride(cols)
,_cstride(1)
,_data(SHPTR_WITH_DEL_NEW(rows * cols))
,_datastart(&_data.get()[0])
,_dataend(&_datastart[rows * cols])
{
}

template <typename _Tp>
Mat<_Tp>::Mat(Scalar * data,Index size, Access mode)
:_rows(1)
,_cols(size)
,_cbeg(0)
,_rbeg(0)
,_stride(size)
,_cstride(1)
,_datastart()
,_dataend()
{
switch(mode)
{
case Access::kCopy:
{
   _data = SHPTR_WITH_DEL_NEW(size);
   _datastart = &_data.get()[0];
   _dataend = &_datastart[size];

   std::copy(&data[0], &data[size], _datastart);
}
break;
case Access::kView:
{
   _data = SHPTR_WITH_NO_DEL(data);
   _datastart = &data[0];
   _dataend = &data[size];
}
break;
case Access::kManage:
```

```
{
    _data = SHPTR_WITH_DEL_REF(data);
    _datastart = &data[0];
    _dataend = &data[size];
}
break;
}
}

template <typename _Tp>
Mat<_Tp>::Mat(Scalar * data,Index rows,Index cols, Access mode)
:_rows(rows)
,_cols(cols)
,_cbeg(0)
,_rbeg(0)
,_stride(cols)
,_cstride(1)
,_datastart()
,_dataend()
{
switch(mode)
{
case Access::kCopy:
{
    _data = SHPTR_WITH_DEL_NEW(rows * cols);
    _datastart = &_data.get()[0];
    _dataend = &_datastart[rows * cols];

    std::copy(&data[0], &data[rows * cols], _datastart);
}
break;
case Access::kView:
{
    _data = SHPTR_WITH_NO_DEL(data);
    _datastart = &data[0];
    _dataend = &data[rows * cols];
}
break;
case Access::kManage:
{
    _data = SHPTR_WITH_DEL_REF(data);
    _datastart = &data[0];
    _dataend = &data[rows * cols];
```

```
}
break;
}
}

template <typename _Tp>
Mat<_Tp>::~Mat()
{
}

template <typename _Tp>
_Tp & Mat<_Tp>::operator()(Index i)
{
auto c = i % _cols;
auto r = i / _cols;

return _datastart[(_rbeg + r) * _stride + (_cbeg + c) * _cstride];
}

template <typename _Tp>
_Tp const & Mat<_Tp>::operator()(Index i) const
{
auto c = i % _cols;
auto r = i / _cols;

return _datastart[(_rbeg + r) * _stride + (_cbeg + c) * _cstride];
}

template <typename _Tp>
_Tp & Mat<_Tp>::operator()(Index r,Index c)
{
return _datastart[(_rbeg + r) * _stride + (_cbeg + c) * _cstride];
}

template <typename _Tp>
_Tp const & Mat<_Tp>::operator()(Index r,Index c) const
{
return _datastart[(_rbeg + r) * _stride + (_cbeg + c) * _cstride];
}

template<typename _Tp>
_Tp& linalg::Mat<_Tp>::at(Index i)
{
```

```
 assert(i < size());

 auto c = i % _cols;
 auto r = i / _cols;

 return _datastart[(_rbeg + r) * _stride + (_cbeg + c) * _cstride];
}

template<typename _Tp>
_Tp const & linalg::Mat<_Tp>::at(Index i) const
{
 assert(i < size());

 auto c = i % _cols;
 auto r = i / _cols;

 return _datastart[(_rbeg + r) * _stride + (_cbeg + c) * _cstride];
}

template<typename _Tp>
_Tp & linalg::Mat<_Tp>::at(Index row,Index col)
{
 assert(row < _rows);
 assert(col < _cols);

 return _datastart[(_rbeg + row) * _stride + (_cbeg + col) * _cstride];
}

template<typename _Tp>
_Tp const & linalg::Mat<_Tp>::at(Index row,Index col) const
{
 assert(row < _rows);
 assert(col < _cols);

 return _datastart[(_rbeg + row) * _stride + (_cbeg + col) * _cstride];
}

template <typename _Tp>
Index Mat<_Tp>::rows() const
{
 return _rows;
}
```

```
template <typename _Tp>
Index Mat<_Tp>::cols() const
{
return _cols;
}

template <typename _Tp>
Index Mat<_Tp>::stride() const
{
return _stride;
}

template <typename _Tp>
Index Mat<_Tp>::size() const
{
return _rows * _cols;
}

template <typename _Tp>
Index Mat<_Tp>::total() const
{
return _dataend - _datastart;
}

template <typename _Tp>
void Mat<_Tp>::resize(Index rows,Index cols)
{
if(rows <= _rows && cols <= _cols)
{
   _rows = rows;
   _cols = cols;
}
else
{
   auto mat = Mat::Zeros(rows, cols);

   for(Index ri = 0, re = std::min(_rows, rows); ri < re; ++ri)
      for(Index ci = 0, ce = std::min(_cols, cols); ci < ce; ++ci)
         mat(ri, ci) = (*this)(ri, ci);

   *this = mat;
}
```

```
}

template <typename _Tp>
template <typename T2>
void Mat<_Tp>::copyTo(Mat<T2> & oth) const
{
 assert(size() == oth.size());

 for(auto i = 0; i < size(); ++i)
    oth(i) = T2((*this)(i));
}

template <typename _Tp>
_Tp const * Mat<_Tp>::datastart() const
{
 return _datastart;
}

template <typename _Tp>
_Tp const * Mat<_Tp>::dataend() const
{
 return _dataend;
}

template <typename _Tp>
_Tp * Mat<_Tp>::data()
{
 return &_datastart[_rbeg * _stride + _cbeg * _cstride];
}

template <typename _Tp>
_Tp const * Mat<_Tp>::data() const
{
 return &_datastart[_rbeg * _stride + _cbeg * _cstride];
}

template <typename _Tp>
_Tp * Mat<_Tp>::ptr(Index row)
{
 assert(row < _rows);

 return &_datastart[(_rbeg + row) * _stride + _cbeg * _cstride];
}
```

```
template <typename _Tp>
_Tp const * Mat<_Tp>::ptr(Index row) const
{
 assert(row < _rows);

 return &_datastart[(_rbeg + row) * _stride + _cbeg * _cstride];
}

template<typename _Tp>
_Tp & linalg::Mat<_Tp>::operator[](Index i)
{
 return (*this)(i);
}

template<typename _Tp>
_Tp const & linalg::Mat<_Tp>::operator[](Index i) const
{
 return (*this)(i);
}

template <typename _Tp>
Mat<_Tp> Mat<_Tp>::row(Index r) const
{
 if(r < 0)
    r += _rows;

 assert(r >= 0);
 assert(r < _rows);

 auto mat{*this};

 mat._rows = 1;
 mat._rbeg += r;

 return mat;
}

template <typename _Tp>
Mat<_Tp> Mat<_Tp>::rowRange(Index startrow,Index endrow) const
{
 if(endrow < 0)
    endrow += _rows + 1;
```

```
assert(startrow >= 0 && startrow <= endrow);
assert(endrow <= _rows);

auto mat{*this};

mat._rows = endrow - startrow;
mat._rbeg += startrow;

return mat;
}

template <typename _Tp>
Mat<_Tp> Mat<_Tp>::rowRange(std::vector<Index> const & rows) const
{
assert(std::all_of(rows.begin(),rows.end(),[&](auto r){return r >= 0 && r < _rows;}));

Mat mat(rows.size(), _cols);

for(auto ri = 0; ri < rows.size(); ++ri)
   for(auto ci = 0; ci < _cols; ++ci)
      mat(ri, ci) = (*this)(rows[ri], ci);

return mat;
}

template <typename _Tp>
Mat<_Tp> Mat<_Tp>::col(Index c) const
{
if(c < 0)
   c += _cols;

assert(c >= 0);
assert(c < _cols);

auto mat{*this};

mat._cols = 1;
mat._cbeg += c;

return mat;
}
```

```
template <typename _Tp>
Mat<_Tp> Mat<_Tp>::colRange(Index startcol,Index endcol) const
{
if(endcol < 0)
   endcol += _cols + 1;

assert(startcol >= 0 && startcol <= endcol);
assert(endcol <= _cols);

auto mat{*this};

mat._cols = endcol - startcol;
mat._cbeg += startcol;

return mat;
}

template <typename _Tp>
Mat<_Tp> Mat<_Tp>::colRange(std::vector<Index> const & cols) const
{
assert(std::all_of(cols.begin(),cols.end(),[&](auto c){return c >= 0 && c < _cols;}));

Mat mat(_rows, cols.size());

for(auto ri = 0; ri < _rows; ++ri)
   for(auto ci = 0; ci < cols.size(); ++ci)
      mat(ri, ci) = (*this)(ri, cols[ci]);

return mat;
}

template <typename _Tp>
Mat<_Tp> Mat<_Tp>::block(Index startrow,Index startcol,Index endrow,Index endcol) const
{
return Mat(*this)
   .rowRange(startrow, endrow)
   .colRange(startcol, endcol);
}

template <typename _Tp>
Mat<_Tp> Mat<_Tp>::range(Index beg,Index end) const
{
```

```
assert(_cols <= 1 || _rows <= 1);

Mat mat;

if(_cols == 1)
   mat = rowRange(beg, end);
else
   mat = colRange(beg, end);

return mat;
}

template <typename _Tp>
Mat<_Tp> Mat<_Tp>::copy() const
{
return Mat(*this, Access::kCopy);
}

template <typename _Tp>
Mat<_Tp> Mat<_Tp>::inv() const
{
    assert(_rows == _cols);

auto mat = gsl_matrix_from_Mat(*this);
    auto inv = gsl_matrix_alloc(mat->size1, mat->size1);

int s;
auto p = gsl_permutation_alloc(mat->size1);

    // Compute the LU decomposition of this matrix
    gsl_linalg_LU_decomp(mat, p, &s);

    // Compute the  inverse of the LU decomposition
    gsl_linalg_LU_invert(mat, p, inv);

    gsl_permutation_free(p);
    gsl_matrix_free(mat);

auto res = gsl_matrix_to_Mat<_Tp>(inv, Access::kManage);

    return res;
}
```

```
template <typename _Tp>
Mat<_Tp> Mat<_Tp>::abs() const
{
auto res = Mat(*this, Access::kCopy);

for(auto i = 0; i < res.size(); ++i)
   res(i) = std::abs(res(i));

return res;
}

template <typename _Tp>
Mat<_Tp> Mat<_Tp>::sqrt() const
{
auto res = Mat(*this, Access::kCopy);

for(auto i = 0; i < res.size(); ++i)
   res(i) = _Tp(std::sqrt(res(i)));

return res;
}

template <typename _Tp>
Mat<_Tp> Mat<_Tp>::householderQr() const
{
if (!size())
   return {};

auto mat = gsl_matrix_from_Mat(*this);
auto tmp = gsl_matrix_alloc(_cols, _cols);

    // Compute the QR decomposition of this matrix
gsl_linalg_QR_decomp_r(mat, tmp);

for(auto ri = 0; ri < _cols; ++ri)
   for(auto ci = 0; ci < _cols; ++ci)
      if(ci < ri)
         tmp->data[ri * _cols + ci] = 0;
      else
         tmp->data[ri * _cols + ci] = mat->data[ri * _cols + ci];

gsl_matrix_free(mat);
```

```
auto res = gsl_matrix_to_Mat<_Tp>(tmp, Access::kManage);

return res;
}

template <typename _Tp>
Mat<_Tp> Mat<_Tp>::singularVectors() const
{
auto A = _rows < _cols ? transpose() : *this;

auto gsl_A = gsl_matrix_from_Mat(A);
auto gsl_X = gsl_matrix_alloc(A.cols(), A.cols());
auto gsl_V = gsl_matrix_alloc(A.cols(), A.cols());
auto gsl_S = gsl_vector_alloc(A.cols());
auto gsl_W = gsl_vector_alloc(A.cols());

// Compute the SVD decomposition of this matrix
//gsl_linalg_SV_decomp(gsl_A, gsl_V, gsl_S, gsl_work);
gsl_linalg_SV_decomp_mod(gsl_A, gsl_X, gsl_V, gsl_S, gsl_W);

Mat res;

if(_rows >= _cols)
{
   res = gsl_matrix_to_Mat<_Tp>(gsl_V, Access::kManage);
   gsl_matrix_free(gsl_A);
}
else
{
   res = gsl_matrix_to_Mat<_Tp>(gsl_A, Access::kManage);
   gsl_matrix_free(gsl_V);
}

gsl_matrix_free(gsl_X);
gsl_vector_free(gsl_S);
gsl_vector_free(gsl_W);

return res;
}

template <typename _Tp>
std::pair<Mat<double>, Mat<double>> Mat<_Tp>::eig(bool symmetric) const
{
```

```
auto A = gsl_matrix_from_Mat(*this);

if(symmetric)
{
   auto eval = gsl_vector_alloc(_cols);
   auto evec = gsl_matrix_alloc(_cols, _cols);
   auto workspace = gsl_eigen_symmv_alloc(_cols);

   gsl_eigen_symmv(A, eval, evec, workspace);

   gsl_eigen_symmv_free(workspace);

   return {
      gsl_vector_to_Mat<double>(eval, Access::kManage),
      gsl_matrix_to_Mat<double>(evec, Access::kManage)
   };
}
else
{
   auto eval = gsl_vector_complex_alloc(_cols);
   auto evec = gsl_matrix_complex_alloc(_cols, _cols);
   auto workspace = gsl_eigen_nonsymmv_alloc(_cols);

   gsl_eigen_nonsymmv(A, eval, evec, workspace);

   gsl_eigen_nonsymmv_free(workspace);

   return {
      gsl_vector_complex_to_Mat_real<double>(eval, Access::kManage),
      gsl_matrix_complex_to_Mat_real<double>(evec, Access::kManage)
   };
}
}

template<typename _Tp>
Mat<_Tp> Mat<_Tp>::vectorRow() const
{
auto mat = *this;
mat._rows = 1;
mat._cols = size();
if(mat._cstride > 1)
   std::swap(mat._stride, mat._cstride);
mat._cstride = 1;
```

```
return mat;
}

template<typename _Tp>
Mat<_Tp> Mat<_Tp>::vectorCol() const
{
Mat mat = *this;
mat._rows = size();
mat._cols = 1;
if(mat._stride > 1)
   std::swap(mat._stride, mat._cstride);
mat._stride = 1;
return mat;
}

template<typename _Tp>
Mat<_Tp> Mat<_Tp>::transpose() const
{
Mat mat = *this;
std::swap(mat._rows, mat._cols);
std::swap(mat._stride, mat._cstride);
return mat;
}

template<typename _Tp>
Mat<_Tp> Mat<_Tp>::diagonal() const
{
Mat mat = *this;

mat._cols = 1;
mat._rows = std::min(_rows, _cols);
mat._stride += 1;

return mat;
}

template<typename _Tp>
Mat<_Tp> Mat<_Tp>::asDiagonal() const
{
assert(_rows == 1 || _cols == 1);

auto mat = Mat::Zeros(size(), size());
```

```
auto mat_diag = mat.diagonal();
copyTo(mat_diag);

return mat;
}

template <typename _Tp>
Mat<_Tp> Mat<_Tp>::Rand(Index size)
{
Mat mat(size);

for(auto i = 0; i < size; ++i)
   mat(i) = Scalar(rand()) / (RAND_MAX);

return mat;
}

template <>
Mat<int> Mat<int>::Rand(Index size)
{
Mat mat(size);

for(auto i = 0; i < size; ++i)
   mat(i) = Scalar(rand() % 10);

return mat;
}

template <typename _Tp>
Mat<_Tp> Mat<_Tp>::Rand(Index _rows,Index cols)
{
Mat mat(_rows, cols);

for(auto i = 0; i < mat.size(); ++i)
   mat(i) = Scalar(rand()) / (RAND_MAX);

return mat;
}

template <>
Mat<int> Mat<int>::Rand(Index rows,Index cols)
{
Mat mat(rows, cols);
```

```
for(auto i = 0; i < mat.size(); ++i)
   mat(i) = Scalar(rand() % 10);

return mat;
}

template <typename _Tp>
Mat<_Tp> Mat<_Tp>::Randi(int max,Index size)
{
Mat mat(size);

for(auto i = 0; i < size; ++i)
   mat(i) = Scalar(rand() % max);

return mat;
}

template <typename _Tp>
Mat<_Tp> Mat<_Tp>::Randi(int max,Index rows,Index cols)
{
Mat mat(rows, cols);

for(auto i = 0; i < mat.size(); ++i)
   mat(i) = Scalar(rand() % max);

return mat;
}

template <typename _Tp>
Mat<_Tp> Mat<_Tp>::Ones(Index size)
{
Mat mat(size);

for(auto i = 0; i < mat.size(); ++i)
   mat(i) = Scalar(1);

return mat;
}

template <typename _Tp>
Mat<_Tp> Mat<_Tp>::Ones(Index _rows,Index cols)
{
```

```
Mat mat(_rows, cols);

for(auto i = 0; i < mat.size(); ++i)
   mat(i) = Scalar(1);

return mat;
}

template <typename _Tp>
Mat<_Tp> Mat<_Tp>::Zeros(Index size)
{
Mat mat(size);

for(auto i = 0; i < mat.size(); ++i)
   mat(i) = Scalar(0);

return mat;
}

template <typename _Tp>
Mat<_Tp> Mat<_Tp>::Zeros(Index _rows,Index cols)
{
Mat mat(_rows, cols);

for(auto i = 0; i < mat.size(); ++i)
   mat(i) = Scalar(0);

return mat;
}

template<typename _Tp>
Mat<_Tp> linalg::Mat<_Tp>::Eye(Index rows,Index cols)
{
Mat mat = Mat::Zeros(rows, cols);

for(Index i = 0, ei = std::min(rows, cols); i < ei; ++i)
   mat(i, i) = _Tp(1);

return mat;
}

template<typename _Tp>
template<typename T2>
```

```
Mat<_Tp> linalg::Mat<_Tp>::mul(Mat<T2> const & rhs)
{
 if(!size() || !rhs.size())
    return {};

 assert(_cols == rhs.rows());

 auto gA = gsl_matrix_from_Mat(*this);
 auto gB = gsl_matrix_from_Mat(rhs);
 auto gC = gsl_matrix_calloc(_rows, rhs.cols());

 gsl_blas_dgemm(CblasNoTrans, CblasNoTrans, 1., gA, gB, 1., gC);

 gsl_matrix_free(gA);
 gsl_matrix_free(gB);

 auto res = gsl_matrix_to_Mat<_Tp>(gC, Access::kManage);

 return res;
}

template<typename _Tp>
template<typename T2>
_Tp Mat<_Tp>::dot(Mat<T2> const & rhs)
{
 assert(size() == rhs.size());

 auto gA = gsl_matrix_from_Mat(this->vectorRow());
 auto gB = gsl_matrix_from_Mat(rhs.vectorCol());
 auto gC = gsl_matrix_calloc(_rows, 1);

 gsl_blas_dgemm(CblasNoTrans, CblasNoTrans, 1., gA, gB, 1., gC);

 gsl_matrix_free(gA);
 gsl_matrix_free(gB);

 Mat res = gsl_matrix_to_Mat<_Tp>(gC, Access::kManage);

 return res(0);
}

template <typename _Tp>
_Tp Mat<_Tp>::max() const
```

```
{
assert(size() > 0);

Scalar max_val = (*this)(0);

for(auto i = 1; i < size(); ++i)
{
   max_val = std::max((*this)(i), max_val);
}

return max_val;
}

template <typename _Tp>
_Tp Mat<_Tp>::min() const
{
assert(size() > 0);

Scalar min_val = (*this)(0);

for(auto i = 1; i < size(); ++i)
{
   min_val = std::min((*this)(i), min_val);
}

return min_val;
}

template <typename _Tp>
_Tp Mat<_Tp>::sum() const
{
Scalar res = Scalar(0);

for(auto i = 0; i < size(); ++i)
   res += (*this)(i);

return res;
}

template <typename _Tp>
_Tp Mat<_Tp>::prod() const
{
if(!size())
```

```
    return Scalar(0);

Scalar res = Scalar(1);

for(auto i = 0; i < size(); ++i)
    res *= (*this)(i);

return res;
}

template <typename _Tp>
_Tp Mat<_Tp>::mean() const
{
return sum() / Scalar(size() ? size() : 1);
}

template <typename _Tp>
Mat<_Tp> Mat<_Tp>::colsMean() const
{
auto res = Mat(1, _cols);

for(auto ci = 0; ci < _cols; ++ci)
    res(ci) = col(ci).mean();

return res;
}

template <typename _Tp>
Mat<_Tp> Mat<_Tp>::rowsMean() const
{
auto res = Mat(_rows, 1);

for(auto ri = 0; ri < _rows; ++ri)
    res(ri) = row(ri).mean();

return res;
}

template<typename _Tp>
Mat<_Tp> Mat<_Tp>::rowsAdd(Mat const & oth) const
{
assert(_cols == oth._cols && oth._rows == 1);
```

```
auto mat = Mat(*this, Access::kCopy);

for(auto ri = 0; ri < _rows; ++ri)
   mat.row(ri) += oth;

return mat;
}

template<typename _Tp>
Mat<_Tp> Mat<_Tp>::rowsSub(Mat const & oth) const
{
assert(_cols == oth._cols && oth._rows == 1);

auto mat = Mat(*this, Access::kCopy);

for(auto ri = 0; ri < _rows; ++ri)
   mat.row(ri) -= oth;

return mat;
}

template <typename _Tp>
bool Mat<_Tp>::operator!=(Mat const & oth) const
{
assert(_rows == oth._rows);
assert(_cols == oth._cols);

for(auto i = 0; i < oth.size(); ++i)
   if((*this)(i) != oth(i))
      return true;

return false;
}

template <typename _Tp>
bool Mat<_Tp>::operator==(Mat const & oth) const
{
return !((*this) != oth);
}

template <typename _Tp>
Mat<_Tp> Mat<_Tp>::operator+(Scalar scalar) const
{
```

```
auto mat = Mat(*this, Access::kCopy);

for(auto i = 0; i < size(); ++i)
   mat(i) += Scalar(scalar);

return mat;
}

template <typename _Tp>
Mat<_Tp> Mat<_Tp>::operator-(Scalar scalar) const
{
auto mat = Mat(*this, Access::kCopy);

for(auto i = 0; i < size(); ++i)
   mat(i) -= Scalar(scalar);

return mat;
}

template <typename _Tp>
Mat<_Tp> Mat<_Tp>::operator*(Scalar scalar) const
{
auto mat = Mat(*this, Access::kCopy);

for(auto i = 0; i < size(); ++i)
   mat(i) *= Scalar(scalar);

return mat;
}

template <typename _Tp>
Mat<_Tp> Mat<_Tp>::operator/(Scalar scalar) const
{
auto mat = Mat(*this, Access::kCopy);

for(auto i = 0; i < size(); ++i)
   mat(i) /= Scalar(scalar);

return mat;
}

template <typename _Tp>
Mat<_Tp> Mat<_Tp>::operator^(Scalar scalar) const
```

```
{
auto mat = Mat(*this, Access::kCopy);

for(auto i = 0; i < size(); ++i)
   mat(i) = Scalar(std::pow(mat(i), scalar));

return mat;
}

template <typename _Tp>
Mat<_Tp> & Mat<_Tp>::operator=(Scalar scalar)
{
for(auto i = 0; i < size(); ++i)
   (*this)(i) = Scalar(scalar);

return *this;
}

template <typename _Tp>
Mat<_Tp> & Mat<_Tp>::operator+=(Scalar scalar)
{
for(auto i = 0; i < size(); ++i)
   (*this)(i) += Scalar(scalar);

return *this;
}

template <typename _Tp>
Mat<_Tp> & Mat<_Tp>::operator-=(Scalar scalar)
{
for(auto i = 0; i < size(); ++i)
   (*this)(i) -= Scalar(scalar);

return *this;
}

template <typename _Tp>
Mat<_Tp> & Mat<_Tp>::operator*=(Scalar scalar)
{
for(auto i = 0; i < size(); ++i)
   (*this)(i) *= Scalar(scalar);

return *this;
```

```
}

template <typename _Tp>
Mat<_Tp> & Mat<_Tp>::operator/=(Scalar scalar)
{
 for(auto i = 0; i < size(); ++i)
    (*this)(i) /= Scalar(scalar);

 return *this;
}

template <typename _Tp>
Mat<_Tp> & Mat<_Tp>::operator^=(Scalar scalar)
{
 for(auto i = 0; i < size(); ++i)
    (*this)(i) = Scalar(std::pow((*this)(i), scalar));

 return *this;
}

template Mat<double>;
template Mat<int>;

template void Mat<int>::copyTo<double>(Mat<double>&) const;
template void Mat<int>::copyTo<int>(Mat<int>&) const;
template void Mat<double>::copyTo<int>(Mat<int>&) const;
template void Mat<double>::copyTo<double>(Mat<double>&) const;

template Mat<int> Mat<int>::mul<int>(Mat<int> const &);
template Mat<int> Mat<int>::mul<double>(Mat<double> const &);
template Mat<double> Mat<double>::mul<int>(Mat<int> const &);
template Mat<double> Mat<double>::mul<double>(Mat<double> const &);

template int Mat<int>::dot<int>(Mat<int> const &);
template int Mat<int>::dot<double>(Mat<double> const &);
template double Mat<double>::dot<int>(Mat<int> const &);
template double Mat<double>::dot<double>(Mat<double> const &);



namespace recal
{
 namespace scnt = ScanContainers;
```

```
namespace cnt = containers;

using RecordEx = ScanContainers::RecordEx;
using Intvl = std::vector<std::pair<double, double>>;

linalg::MatXd capture(RecordEx const& record, cnt::Color color, int track);
linalg::MatXd capture(RecordExArray const& records, cnt::Color color, int track);

class Index
{
   cnt::Color scan_;
   int track_;

public:
   Index();
   Index(cnt::Color scan,int track);

   bool operator<(Index const& oth);
   friend bool operator<(Index const& lhs, Index const& rhs);

   cnt::Color scan() const { return scan_; }
   int track() const { return track_; }

   bool valid() const { return scan_ != cnt::Color::INVALID && track_ >= 0; }

   friend std::ofstream& operator<<(std::ofstream& out, Index const& idx);
   friend std::ifstream& operator>>(std::ifstream& in, Index& idx);
};

class ClassResults;
class TracksBatch
{
   friend class ClassResults;

   /* batch args */
   bool tested_ = true;
   double maxdist_ = 750.;
   /* mahal args */
   linalg::MatXd R_;
   linalg::MatXd C_;
   size_t N_ = 0;
   /* variation */
   std::vector<Intvl> varintvl_;
```

```
    Index index_;
    /* hash */
    ULONG64 hash_{};

    /* buffer-args */
    linalg::MatXd V_;
    /* hash-sync */
    ULONG64 vhash_{};

public:
    TracksBatch() = default;
    TracksBatch(Index const& index);
    TracksBatch(cnt::Color scan, int track);

    size_t count() const;
    size_t dims() const;

    Index index() const;

    bool tested() const { return tested_; }
    void setTested(bool tested) { tested_ = tested; }
    double maxdist() const { return maxdist_; }
    void setMaxdist(double dist) { maxdist_ = dist; }

    std::vector<double> mahal(RecordExArray const& records);
    linalg::MatXd reduce(RecordExArray const& records, int fst, int snd);

    friend std::ofstream& operator<<(std::ofstream& out, TracksBatch const& batch);
    friend std::ifstream& operator>>(std::ifstream& in, TracksBatch& batch);

private:
    double mahal(RecordEx const& record) const;
    std::vector<Intvl> const& varintvl() const;

    void recalculate(RecordExArray const& records);
    void CalcMahalArgs(const linalg::MatXd& tracks);
    void CalcVarIntvls(const linalg::MatXd& tracks);
};

class ClassResults
{
    cid_long cid_;
    std::set<ULONG64> trainset_;
```

```
   std::map<Index, TracksBatch> batches_;
   RecordExArray records_;

public:
   ClassResults() = default;
   ClassResults(cid_long cid);

   auto batch   (cnt::Color scan, int track) ->TracksBatch&;
   auto varintvl(cnt::Color scan, int record) ->std::vector<Intvl> const&;

   recal::Index test(RecordEx const& record);

   auto trainsetIndices() const ->std::set<int>;
   auto trainset() const ->std::set<ULONG64> const&;
   void setTrainset(std::set<ULONG64> const& trainset);

   RecordExArray getRecords() const;
   RecordExArray getTrainsetRecords() const;

   auto batches() -> std::map<Index, TracksBatch>&;
   cid_long cid() const;

   friend std::ofstream& operator<<(std::ofstream& out, ClassResults const& idx);
   friend std::ifstream& operator>>(std::ifstream& in, ClassResults& idx);
};

struct State
{
private:
   /* state-props main */
   cid_long
      cid_;
   cnt::Color
      scan_;
   int trid_,
      rcid_;

   /* state-props subs */
   int
      fstc_ = 0,
      sndc_ = 1,
      dist_ = 0,
      diff_ = 0;
```

```
    bool
       tested_;

    /* state-data */
    std::vector<double>
       mahal_;
    linalg::MatXd
       reduced_;
    std::map<cid_long,recal::ClassResults>
       results_;
    RecordExArray
       records_;
    linalg::MatXd
       selected_scan_,
       centered_scan_,
       differen_scan_,
       diffiltd_scan_;

    /* save-file */
    std::filesystem::path path_;

public:
    State();
    State(cid_long cid,cnt::Color color = cnt::Color::INVALID,int track = -1);
    ~State();

    recal::ClassResults& results(cid_long cid);
    recal::ClassResults& results();
    std::map<cid_long,recal::ClassResults>& allResults();

    /* state-props */
    cid_long cid() const;
    void setCid(cid_long const& value,bool update = true);

    cnt::Color scan() const;
    void setScan(cnt::Color const& scan,bool update = true);

    int track() const;
    void setTrack(int track,bool update = true);

    int recordid() const;
    void setRecordid(int record,bool update = true);
```

```
    int fstComponent() const;
    void setFstComponent(int index);

    int sndComponent() const;
    void setSndComponent(int index);

    int maxdist() const;
    void setMaxdist(int maxdist);

    bool tested() const;
    void setTested(bool tested);

    RecordEx *const selectedRecord() const;
    TracksBatch& batch();

    int maxdiff() const;
    int diff() const;
    void setDiff(int diff);

    RecordExArray const& records() const;
    std::vector<double> const& mahal() const;
    linalg::MatXd const& reduced() const;
    linalg::MatXd const& selectedScan() const;
    linalg::MatXd const& centeredScan() const;
    linalg::MatXd const& differenScan() const;

    void updateTrckData();
    void updateScanData();
    void updateClssData();
    void updateRecdData();

    void updateSelectedScan();
    void updateCenteredScan();
    void updateDifferenScan();

private:
    void writeResults();
    void readResults();
};

/**
 * \brief Do a hash_combine just like in the BOOST library
```

```
 * .
 * \param seed
 * \param value
 */
template<typename T>
inline void hash_combine(ULONG64& seed, T const& value)
{
   std::hash<T> h;
   seed ^= h(value) + 0x9e3779b9 + (seed << 6) + (seed >> 2);
}

/**
 * \brief Sorts array and returns indices of its elements
 */
std::vector<size_t> sort_indices(linalg::MatXd const& arr, bool sort = false);
}

recal::Index::Index()
: scan_(cnt::Color::INVALID),track_(-1)
{}

recal::Index::Index(cnt::Color scan,int track)
: scan_(scan),track_(track)
{}

bool recal::Index::operator<(Index const & oth)
{
return std::tie(scan_, track_) < std::tie(oth.scan_, oth.track_);
}

recal::TracksBatch::TracksBatch(Index const& index)
: index_{index}
, varintvl_{}
, R_{}
, C_{}
, N_{}
, V_{}
{
}

recal::TracksBatch::TracksBatch(cnt::Color scan,int track)
: TracksBatch(Index(scan, track))
{
```

```
}

size_t recal::TracksBatch::count() const
{
return N_;
}

size_t recal::TracksBatch::dims() const
{
return C_.cols();
}

recal::Index recal::TracksBatch::index() const
{
return index_;
}

void recal::TracksBatch::CalcMahalArgs(const linalg::MatXd& tracks)
{
if(tracks.rows() >= tracks.cols())
{
   /* center of distribution */
   C_ = tracks.colsMean();

   /* centered train-matrix */
   auto Cx = tracks.rowsSub(C_);

   R_ = Cx.householderQr();

   if (R_.diagonal().prod() != 0)
      R_ = R_.transpose().inv();
   else
      R_ = linalg::MatXd::Zeros(Cx.cols(), Cx.cols());
}
}

void recal::TracksBatch::CalcVarIntvls(const linalg::MatXd& tracks)
{
const auto sigmas = std::array<float, 3>{ .68, .95, .997 };

varintvl_ = std::vector<Intvl>(3, Intvl(tracks.cols()));

auto buff = std::vector<double>(tracks.rows());
```

```
for(auto ci = 0; ci < tracks.cols(); ++ci)
{
   auto col = tracks.col(ci);
   std::copy(col.begin(),col.end(),buff.begin());

   for(auto si = 0; si < sigmas.size(); ++si)
   {
      auto const& sigma = sigmas[si];
      auto& intvl = varintvl_[si];

      auto rght_bnd = tracks.rows() * sigma;
      auto left_bnd = tracks.rows() - rght_bnd;

      std::nth_element(buff.begin(),buff.begin() + left_bnd,buff.end());
      std::nth_element(buff.begin(),buff.begin() + rght_bnd,buff.end());

      intvl[ci] = std::make_pair(
         *(buff.begin() + left_bnd),
         *(buff.begin() + rght_bnd));
   }
}
}

double recal::TracksBatch::mahal(RecordEx const& record) const
{
auto track = capture(record, index_.scan(), index_.track());
auto min_len = std::min(track.size(), C_.size());

auto Cy = (track.range(0, min_len) - C_.range(0, min_len)).transpose();
auto ri = R_.block(0, 0, min_len, min_len).mul(Cy);

auto dist = (ri * ri).sum() * (N_ - 1);

dist = std::min(
      std::max(dist, (double) Int16::MinValue),
                  (double) Int16::MaxValue);

return dist;
}

std::vector<double> recal::TracksBatch::mahal(RecordExArray const & records)
{
```

```
auto dists = std::vector<double>(records.size(),.0);

int ri;
#pragma omp parallel shared(dists, records) private(ri)
{
#pragma omp for
   for(ri = 0; ri < records.size(); ++ri)
   {
      auto& record = records[ri];
      dists[ri] = mahal(*record);
   }
}

return dists;
}

void recal::TracksBatch::recalculate(RecordExArray const& records)
{
auto mhash = hash_;
hash_ = 0;
for(auto& record: records)
   hash_combine(hash_,record->GetHash());

if(hash_ != mhash)
{
   auto tracks = capture(records, index_.scan(), index_.track());
   N_ = tracks.rows();

   CalcMahalArgs(tracks);
   CalcVarIntvls(tracks);
}
}

std::vector<recal::Intvl> const& recal::TracksBatch::varintvl() const
{
return varintvl_;
}

linalg::MatXd recal::TracksBatch::reduce(RecordExArray const& records, int fst, int
snd)
{
auto vhash_prev = vhash_;
vhash_ = 0;
for(auto& record: records)
```

```
    hash_combine(vhash_, record->GetHash());

auto tracks = capture(records, index_.scan(), index_.track());

if (vhash_ != vhash_prev)
    V_ = tracks.singularVectors();

fst = std::clamp(fst, 0, int(V_.cols()-1));
snd = std::clamp(snd, 0, int(V_.cols()-1));

return tracks.mul(V_.colRange({fst, snd}));
}

recal::ClassResults::ClassResults(cid_long cid)
: cid_(cid)
{
records_ = ui3()->cs->GetArchiveTable()->GetRecordList(cid);
}

recal::TracksBatch& recal::ClassResults::batch(cnt::Color scan, int track)
{
auto index = Index(scan, track);

if (batches_.find(index) == batches_.end())
{
    auto records = getTrainsetRecords();

    batches_.emplace(index,TracksBatch(index));
    batches_[index].recalculate(records);
}

return batches_.at(index);
}

recal::Index recal::ClassResults::test(RecordEx const & record)
{
for (auto& batch: batches_)
    if (batch.second.tested())
    {
        auto dist = batch.second.mahal(record);
        if (dist > batch.second.maxdist())
        {
            return batch.first;
```

```
      }
   }
return {};
}

std::set<int> recal::ClassResults::trainsetIndices() const
{
auto records = getRecords();

auto indices = std::set<int>();
for(auto i = 0, idx = 0; idx < records.size(); ++idx)
   if (trainset_.find(records[idx]->GetHash()) != trainset_.end())
      indices.emplace(idx);

return indices;
}

RecordExArray recal::ClassResults::getTrainsetRecords() const
{
auto records = getRecords();

auto train_records = RecordExArray();
train_records.reserve(trainset_.size());

for(auto& record: records)
{
   auto hash = record->GetHash();
   if(trainset_.find(hash) != trainset_.end())
      train_records.emplace_back(record);
}

return train_records;
}

std::set<ULONG64> const & recal::ClassResults::trainset() const
{
return trainset_;
}

void recal::ClassResults::setTrainset(std::set<ULONG64> const & trainset)
{
trainset_ = trainset;
```

```
auto train_records = getTrainsetRecords();
auto batches = std::vector<std::map<Index,
TracksBatch>::iterator>(batches_.size());

auto bi = 0;
for(auto itr = batches_.begin(); itr != batches_.end(); ++itr)
   batches[bi++] = itr;

#pragma omp parallel shared(batches, train_records) private(bi)
{
#pragma omp for
   for(bi = 0; bi < batches.size(); ++bi)
   {
      auto& batch = batches[bi];
      batch->second.recalculate(train_records);
   }
}
}

RecordExArray recal::ClassResults::getRecords() const
{
return ui3()->cs
   ->GetArchiveTable()
   ->GetRecordList(cid_);
}

auto recal::ClassResults::batches() -> std::map<Index,TracksBatch>&
{
return batches_;
}

cid_long recal::ClassResults::cid() const
{
return cid_;
}

linalg::MatXd recal::capture(RecordEx const & record, cnt::Color color, int tri)
{
auto mat = record.GetMatrixRotated(*record.FindScan(color));
auto scan = linalg::MatXi(mat.GetData(),mat.GetHeight(),mat.GetWidth());

tri = std::min(tri, int(scan.rows() - 1));

auto track = scan.row(tri);
```

```
auto inr = std::make_pair(0ll,track.size() - 1);

for(;inr.second > inr.first; --inr.second)
   if(track(inr.second) != track(inr.second - 1))
      break;

for(;inr.first < inr.second; ++inr.first)
   if(track(inr.first) != track(inr.first + 1))
      break;

if(inr.first == inr.second)
   inr.second = track.size() - 1;
else
{
   const auto indent = 10;
   inr.first  = std::min(inr.second, inr.first  + indent);
   inr.second = std::max(inr.first,  inr.second - indent);
}

auto interval = track
      .range(inr.first, inr.second + 1);
    interval -= interval.mean();

return interval.cast<double>();
}

linalg::MatXd recal::capture(RecordExArray const & records, cnt::Color color, int
tri)
{
using Interval = std::pair<linalg::Index, linalg::Index>;

auto intervals = std::vector<Interval>();
intervals.reserve(records.size());

auto maxlen = 0ll;

for(auto& record: records)
{
   auto mat = record->GetMatrixRotated(*record->FindScan(color));
   auto scan = linalg::MatXi(
      mat.GetData(),mat.GetHeight(),mat.GetWidth());

   tri = std::min(tri, int(scan.rows() - 1));
```

```
   auto track = scan.row(tri);

   auto inr = std::make_pair(0ll,track.size() - 1);

   for(;inr.second > inr.first; --inr.second)
      if(track(inr.second) != track(inr.second - 1))
         break;

   for(;inr.first < inr.second; ++inr.first)
      if(track(inr.first) != track(inr.first + 1))
         break;

   if(inr.first == inr.second)
      inr.second = track.size() - 1;
   else
   {
      const auto indent = 10;
      inr.first = std::min(inr.second, inr.first + indent);
      inr.second = std::max(inr.first, inr.second - indent);
   }

   maxlen = std::max(maxlen, inr.second - inr.first + 1);

   intervals.emplace_back(inr);
}

auto tracks = linalg::MatXd::Zeros(records.size(), maxlen);

for(auto ri = 0; ri < records.size(); ++ri)
{
   auto& record = records[ri];

   auto mat = record->GetMatrixRotated(*record->FindScan(color));
   auto scan = linalg::MatXi(
      mat.GetData(),mat.GetHeight(),mat.GetWidth());

   auto track = scan.row(tri);
   auto inr = intervals[ri];
   auto interval = track
      .range(inr.first, inr.second + 1);

   interval -= interval.mean();
```

```
   interval.copyTo(tracks.row(ri).range(0, interval.size()));
}

return tracks;
}

bool recal::operator<(Index const & lhs,Index const & rhs)
{
return std::tie(lhs.scan_, lhs.track_) < std::tie(rhs.scan_, rhs.track_);
}

std::ofstream & recal::operator<<(std::ofstream & out,Index const & idx)
{
out.write(reinterpret_cast<const char*>(&idx.scan_), sizeof(cnt::Color));
out.write(reinterpret_cast<const char*>(&idx.track_), sizeof(int));

return out;
}

std::ifstream & recal::operator>>(std::ifstream & in,Index & idx)
{
in.read(reinterpret_cast<char*>(&idx.scan_), sizeof(cnt::Color));
in.read(reinterpret_cast<char*>(&idx.track_), sizeof(int));

return in;
}

std::ofstream & recal::operator<<(std::ofstream & out,TracksBatch const & batch)
{
/* tested */
out.write(reinterpret_cast<const char*>(&batch.tested_), sizeof(batch.tested_));

/* maxdist */
out.write(reinterpret_cast<const char*>(&batch.maxdist_), sizeof(batch.maxdist_));

/* matrix R */
out << batch.R_;

/* matrix C_ */
out << batch.C_;

/* N */
out.write(reinterpret_cast<const char*>(&batch.N_), sizeof(batch.N_));
```

```
/* varintvl */
auto varintvls_count = batch.varintvl_.size();
out.write(reinterpret_cast<const                          char*>(&varintvls_count),
sizeof(varintvls_count));

for(auto& varintvl: batch.varintvl_)
{
   auto varintvl_size = varintvl.size();
   out.write(reinterpret_cast<const char*>(&varintvl_size), sizeof(varintvl_size));
   out.write(reinterpret_cast<const   char*>(varintvl.data()),   varintvl_size   *
sizeof(Intvl::value_type));
}

/* index */
out << batch.index_;

/* hash */
out.write(reinterpret_cast<const char*>(&batch.hash_), sizeof(batch.hash_));

/* matrix V */
out << batch.V_;

/* vhash */
out.write(reinterpret_cast<const char*>(&batch.vhash_), sizeof(batch.vhash_));

/**/
return out;
}

std::ifstream & recal::operator>>(std::ifstream & in,TracksBatch & batch)
{
/* tested */
in.read(reinterpret_cast<char*>(&batch.tested_), sizeof(batch.tested_));

/* maxdist */
in.read(reinterpret_cast<char*>(&batch.maxdist_), sizeof(batch.maxdist_));

/* matrix R */
in >> batch.R_;

/* matrix C_ */
in >> batch.C_;
```

```
/* N */
in.read(reinterpret_cast<char*>(&batch.N_), sizeof(batch.N_));

/* varintvl */
auto varintvls_count = size_t();
in.read(reinterpret_cast<char*>(&varintvls_count), sizeof(varintvls_count));

batch.varintvl_.resize(varintvls_count);

for(auto i = 0; i < varintvls_count; ++i)
{
   auto varintvl_size = size_t();
   in.read(reinterpret_cast<char*>(&varintvl_size), sizeof(varintvl_size));

   batch.varintvl_[i].resize(varintvl_size);

   in.read(reinterpret_cast<char*>(batch.varintvl_[i].data()),   varintvl_size   *
sizeof(Intvl::value_type));
}

/* index */
in >> batch.index_;

/* hash */
in.read(reinterpret_cast<char*>(&batch.hash_), sizeof(batch.hash_));

/* matrix V */
in >> batch.V_;

/* vhash */
in.read(reinterpret_cast<char*>(&batch.vhash_), sizeof(batch.vhash_));

/**/
return in;
}

std::ofstream & recal::operator<<(std::ofstream & out,ClassResults const & res)
{
/* class_ */
out.write(reinterpret_cast<const char*>(&res.cid_), sizeof(res.cid_));

/* trainset */
auto trainset_size = res.trainset_.size();
out.write(reinterpret_cast<const char*>(&trainset_size), sizeof(trainset_size));
```

```
for(auto& hash: res.trainset_)
   out.write(reinterpret_cast<const char*>(&hash), sizeof(hash));

/* batches */
auto batches_count = res.batches_.size();
out.write(reinterpret_cast<char*>(&batches_count), sizeof(batches_count));

for(auto& pair: res.batches_)
   out << pair.second;

/**/
return out;
}

std::ifstream & recal::operator>>(std::ifstream & in,ClassResults & res)
{
/* class_ */
in.read(reinterpret_cast<char*>(&res.cid_), sizeof(res.cid_));

/* trainset */
auto trainset_size = size_t();
in.read(reinterpret_cast<char*>(&trainset_size), sizeof(trainset_size));

for(auto i = 0; i < trainset_size; ++i)
{
   auto hash = size_t();
   in.read(reinterpret_cast<char*>(&hash), sizeof(hash));

   res.trainset_.emplace(hash);
}

/* batches */
auto batches_count = size_t();
in.read(reinterpret_cast<char*>(&batches_count), sizeof(batches_count));

for(auto i = 0; i < batches_count; ++i)
{
   auto batch = TracksBatch();
   in >> batch;

   res.batches_.emplace(batch.index(), batch);
}
```

```
/**/
return in;
}

std::vector<size_t> recal::sort_indices(linalg::MatXd const & arr,bool sort)
{
std::vector<size_t> idx(arr.size());
std::iota(idx.begin(),idx.end(),0);

if(sort)
   std::stable_sort(idx.begin(),idx.end(),
   [&arr](size_t i1,size_t i2) {return arr[i1] < arr[i2];});

return idx;
}

std::vector<recal::Intvl> const& recal::ClassResults::varintvl(cnt::Color scan, int
track)
{
auto& batch = this->batch(scan, track);
return batch.varintvl();
}

recal::State::State()
: cid_()
, scan_()
, trid_()
, rcid_()
{
readResults();
}

recal::State::State(cid_long cid,cnt::Color color,int track)
: cid_(cid)
, scan_(color)
, trid_(track)
, rcid_(0)
{
updateClssData();
}

recal::State::~State()
{
```

```
writeResults();
}

void recal::State::updateSelectedScan()
{
if(records_.empty()
|| rcid_ < 0 || rcid_ >= records_.size()
|| records_[rcid_]->FindScan(scan_) == nullptr)
{
   selected_scan_ = linalg::MatXd();
}
else
{
   auto record = records_.at(rcid_);
   auto scan = record->GetMatrixRotated(*record->FindScan(scan_));

   selected_scan_      =      linalg::MatXi(scan.GetData(),      scan.GetHeight(),
scan.GetWidth())
      .cast<double>();
}
}

void recal::State::updateCenteredScan()
{
centered_scan_            =            linalg::MatXd::Zeros(selected_scan_.rows(),
selected_scan_.cols());

auto trainset = results(cid_).getTrainsetRecords();

if (!trainset.empty())
{
   auto count = 0.;

   for(auto& record : trainset)
   {
      if(auto scan = record->FindScan(scan_))
      {
         auto matx = record->GetMatrixRotated(*scan);
         auto     mat     =     linalg::MatXi(matx.GetData(),     matx.GetHeight(),
matx.GetWidth());

         auto mrows = std::min(centered_scan_.rows(), mat.rows());
         auto mcols = std::min(centered_scan_.cols(), mat.cols());
```

```
            centered_scan_.block(0, 0, mrows, mcols) += mat.block(0, 0, mrows, mcols);

            count += 1;
         }
      }

      centered_scan_ /= count;
   }
   }

   void recal::State::updateDifferenScan()
   {
   auto rows = std::min(selected_scan_.rows(), centered_scan_.rows()),
        cols = std::min(selected_scan_.cols(), centered_scan_.cols());

   if(rows || cols)
   {
      differen_scan_
         = (selected_scan_.block(0, 0, rows, cols)
          - centered_scan_.block(0, 0, rows, cols)).abs();
   }
   else
   {
      differen_scan_ = linalg::MatXd::Zeros(rows, cols);
   }

   diffiltd_scan_ = differen_scan_.copy();
   if(diff_ > 0)
   {
      for(auto i = 0; i < diffiltd_scan_.size(); ++i)
      {
         diffiltd_scan_(i) = differen_scan_(i) < diff_ ? 1 : 0;
      }
   }
   }

   void recal::State::writeResults()
   {
   auto file = std::ofstream(path_, std::ios::binary | std::ios::out);

   if(file.is_open())
   {
      auto results_count = results_.size();
```

```
    file.write(reinterpret_cast<char*>(&results_count),sizeof(results_count));

    for(auto& pair: results_)
        file << pair.second;

    file.close();
}
}

void recal::State::readResults()
{
path_ = ui3()->cfg->GetWorkingPath() / "Recal.dat";
auto file = std::ifstream(path_, std::ios::binary);

if(file.is_open())
{
    auto results_count = size_t();
    file.read(reinterpret_cast<char*>(&results_count),sizeof(results_count));

    for(auto i = 0; i < results_count; ++i)
    {
        auto results = ClassResults();
        file >> results;

        results_.emplace(results.cid(),results);
    }

    file.close();
}
}

recal::ClassResults & recal::State::results(cid_long cid)
{
if(results_.find(cid) == results_.end())
{
    results_.emplace(cid,recal::ClassResults(cid));
}
return results_.at(cid);
}

recal::ClassResults & recal::State::results()
{
return results(cid_);
```

```
}

std::map<cid_long,recal::ClassResults>& recal::State::allResults()
{
 return results_;
}

cid_long recal::State::cid() const
{
 return cid_;
}

void recal::State::setCid(cid_long const & value, bool update)
{
 auto prev_value = cid_;
 cid_ = value;

 if(update && value != prev_value)
    updateClssData();
}

auto recal::State::scan() const -> cnt::Color
{
 return scan_;
}

void recal::State::setScan(cnt::Color const& value, bool update)
{
 auto prev_value = scan_;
 scan_ = value;

 if(update && value != prev_value)
    updateScanData();
}

int recal::State::track() const
{
 return trid_;
}

void recal::State::setTrack(int value, bool update)
{
 auto prev_value = trid_;
```

```
trid_ = value;

if(update && value != prev_value)
   updateTrckData();
}

int recal::State::recordid() const
{
return rcid_;
}

void recal::State::setRecordid(int value, bool update)
{
auto prev_value = rcid_;
rcid_ = value;

if(update && value != prev_value)
   updateRecdData();
}

int recal::State::fstComponent() const
{
return fstc_;
}

void recal::State::setFstComponent(int value)
{
auto prev_value = fstc_;
fstc_ = value;

if(fstc_ != prev_value)
{
   reduced_ = batch().reduce(records_, fstc_, sndc_);
}
}

int recal::State::sndComponent() const
{
return sndc_;
}

void recal::State::setSndComponent(int value)
{
```

```
auto prev_value = sndc_;
sndc_ = value;

if(sndc_ != prev_value)
{
   reduced_ = batch().reduce(records_, fstc_, sndc_);
}
}

int recal::State::maxdist() const
{
return dist_;
}

void recal::State::setMaxdist(int maxdist)
{
batch().setMaxdist(maxdist);
dist_ = maxdist;
}

bool recal::State::tested() const
{
return tested_;
}

void recal::State::setTested(bool tested)
{
batch().setTested(tested);
tested_ = tested;
}

recal::RecordEx *const recal::State::selectedRecord() const
{
if(rcid_ >= 0 && rcid_ < records_.size())
   return records_[rcid_];
else
   return nullptr;
}

recal::TracksBatch & recal::State::batch()
{
return results(cid_).batch(scan_, trid_);
}
```

```
int recal::State::maxdiff() const
{
return ceil(differen_scan_.max());
}

int recal::State::diff() const
{
return diff_;
}

void recal::State::setDiff(int value)
{
auto prev_value = diff_;
diff_ = value;

if(diff_ != prev_value && diff_ > 0)
{
   for(auto i = 0; i < diffiltd_scan_.size(); ++i)
   {
      diffiltd_scan_(i) = differen_scan_(i) < diff_ ? 1 : 0;
   }
}
}

RecordExArray const & recal::State::records() const
{
return records_;
}

std::vector<double> const & recal::State::mahal() const
{
return mahal_;
}

linalg::MatXd const& recal::State::reduced() const
{
return reduced_;
}

linalg::MatXd const& recal::State::selectedScan() const
{
return selected_scan_;
```

```
}

linalg::MatXd const& recal::State::centeredScan() const
{
return centered_scan_;
}

linalg::MatXd const& recal::State::differenScan() const
{
if(diff_)
   return diffiltd_scan_;
else
   return differen_scan_;
}

void recal::State::updateTrckData()
{
if(records_.empty()
|| records_.front()->FindScan(scan_) == nullptr
|| trid_ < 0 || trid_ >= records_.front()->FindScan(scan_)->GetHeight())
{
   mahal_.clear();
   reduced_ = linalg::MatXd();
}
else
{
   auto& batch = this->batch();

   mahal_  = batch.mahal(records_);
   reduced_= batch.reduce(records_, fstc_, sndc_);

   dist_ = batch.maxdist();
   tested_ = batch.tested();
}
}

void recal::State::updateScanData()
{
updateSelectedScan();
updateCenteredScan();
updateDifferenScan();
updateTrckData();
}
```

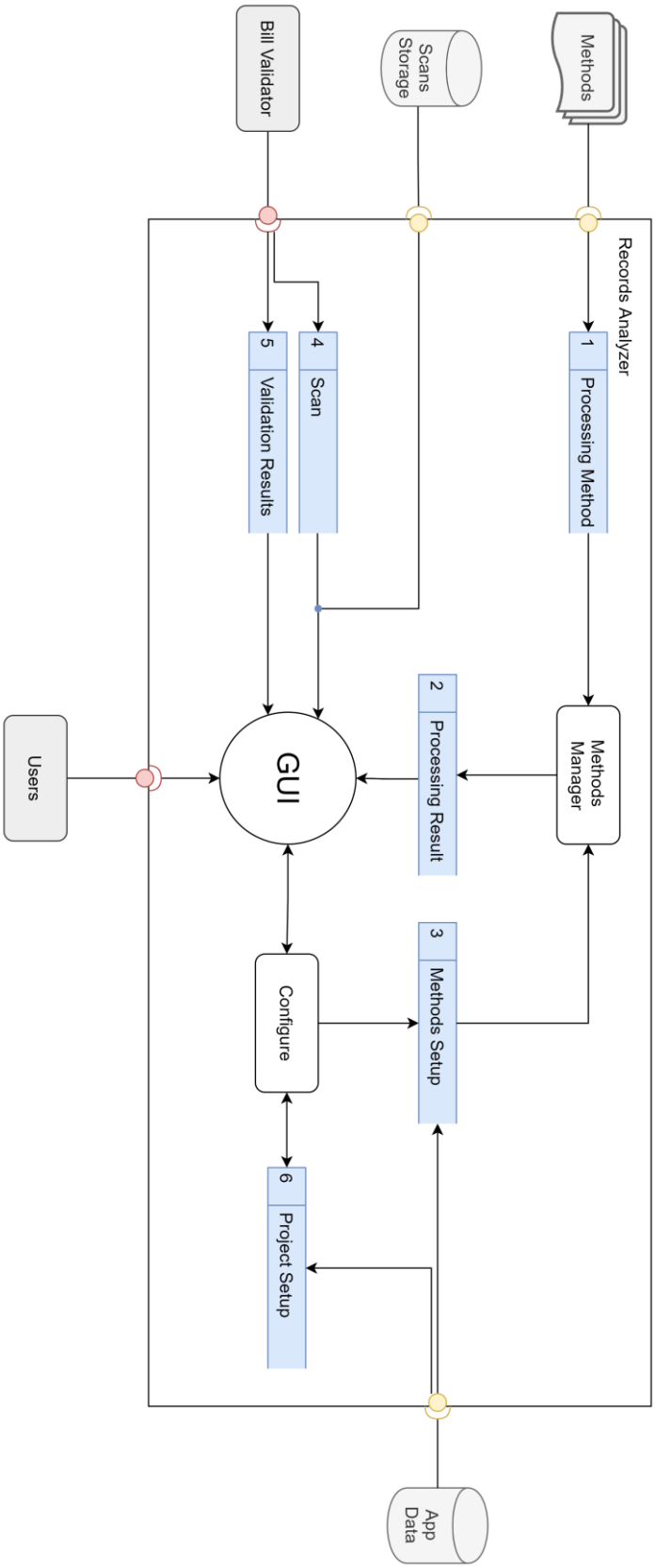
```
void recal::State::updateClssData()
{
 records_ = ui3()->cs
    ->GetArchiveTable()
    ->GetRecordList(cid_);

 updateScanData();
}

void recal::State::updateRecdData()
{
 updateSelectedScan();
 updateDifferenScan();
}
```

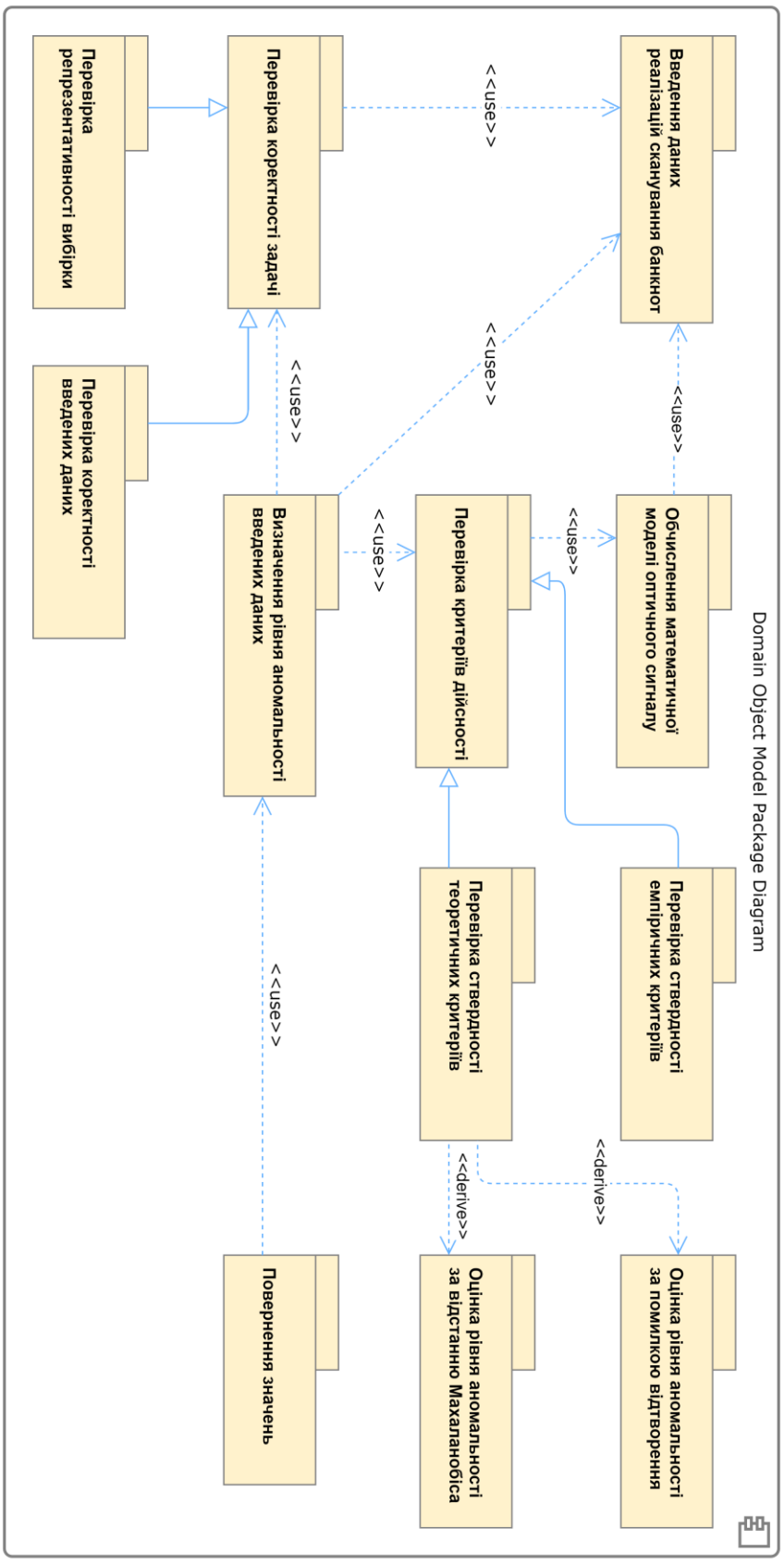
ДОДАТОК Б

Функціональна схема ПЗ



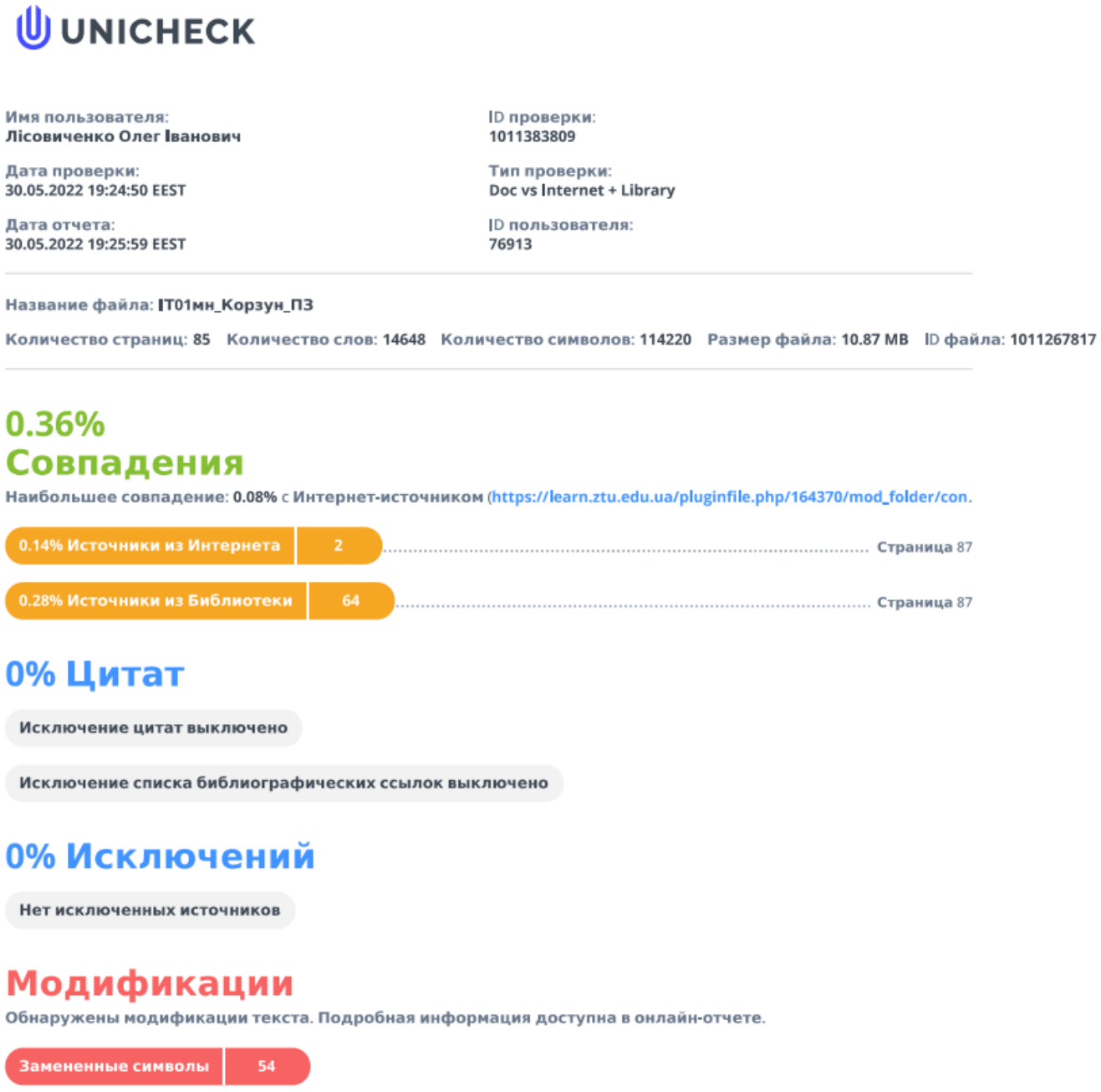
ДОДАТОК В

Об’єктна модель пакетів бібліотеки



ДОДАТОК Г

Результати перевірки роботи на співпадіння



UNICHECK

Имя пользователя:
Лісовиченко Олег Іванович

ID проверки:
1011383809

Дата проверки:
30.05.2022 19:24:50 EEST

Тип проверки:
Doc vs Internet + Library

Дата отчета:
30.05.2022 19:25:59 EEST

ID пользователя:
76913

Название файла: **IT01мн_Корзун_ПЗ**

Количество страниц: **85** Количество слов: **14648** Количество символов: **114220** Размер файла: **10.87 MB** ID файла: **1011267817**

0.36%
Совпадения

Наибольшее совпадение: **0.08%** с Интернет-источником (https://learn.ztu.edu.ua/pluginfile.php/164370/mod_folder/con.

0.14% Источники из Интернета | 2 ..... Страница 87

0.28% Источники из Библиотеки | 64 ..... Страница 87

0% Цитат

Исключение цитат выключено

Исключение списка библиографических ссылок выключено

0% Исключений

Нет исключенных источников

Модификации

Обнаружены модификации текста. Подробная информация доступна в онлайн-отчете.

Замененные символы | 54