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

Факультет електроніки
(повна назва інституту/факультету)

Кафедра мікроелектроніки
(повна назва кафедри)

«На правах рукопису»

УДК _____

«До захисту допущено»

Завідувач кафедри

_____ О.В. Борисов
(підпис) (ініціали, прізвище)

"___"_____20__ р.

Магістерська дисертація

зі спеціальності 153 «Мікро- та наносистемна техніка»,
(код) (назва)

на тему: Інформаційна система для контролю та обробки візуальних даних лабораторних звітів.

Виконав: студент 6 курсу, групи ДП-72 мп
(шифр групи)

Ляшук Андрій Миколайович _____
(прізвище, ім'я, по батькові) (підпис)

Науковий керівник проф., д.ф.-м.н, Поплавко Юрій Михайлович _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант доц., к.т.н. Домбругов Михайло Ремович _____
(назва розділу) (науковий ступінь, вчене звання, , прізвище, ініціали) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

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

Студент _____
(підпис)

Київ – 2018 року

Реферат

Дипломна робота налічує 109 сторінок, 3 розділи, 32 ілюстрації, 22 таблиці та 75 джерел за переліком посилань.

Актуальність теми полягає в тому, що ринок інформаційних систем, які призначенні для обробки візуальних даних з подальшою синхронізацією, активно розвивається. Конкуренція на ринку досить низька, у зв'язку з малою кількістю додатків-аналогів. Створений додаток «WeCapture» можна застосовувати у сферах страхування майна, будівництва та інших галузях, в яких необхідно зберігати графічні матеріали з їх описом в процесі роботи. Розширення сфер застосування збільшує шанс потенційного успіху інформаційної системи тому що, для невеликих компаній універсальне рішення набагато вигідніше, ніж розроблювати спеціалізований додаток для своїх потреб.

Метою магістерської дисертації є дослідження та розробка інформаційної системи для збору та обробки візуальних даних, що буде використана у лабораторних дослідженнях.

Для досягнення мети дослідження, необхідно виконати такі *завдання:*

1. Провести аналіз методів обробки графічних даних для поліпшення якості отриманих зображень при низькій освітленості.
2. Розглянути способи побудови інформаційної системи, наведені засоби для проектування та розробки архітектури додатку.
3. Дослідити та проаналізувати наведені приклади застосування бібліотеки для обробки зображень «OpenCV», за допомогою цієї бібліотеки використовуються алгоритми, комбінація яких дозволяє усувати шуми, поліпшувати контрастність, виконувати матричні перетворення на отриманому зображенні.
4. Висвітлити план реалізації власно розробленої архітектури інформаційної системи «WeCapture», наведені поетапні інструкції для базової роботи користувача с мобільним додатком.

Об'єкт дослідження: візуальні дані, отримані під час роботи над лабораторними дослідженнями

Предмет дослідження: використання інформаційної системи при роботі над лабораторними дослідженням.

Наукова новизна дослідження полягає в розробленні унікального алгоритму напівказуальної фільтрації цифрових зображень.

Практичне значення одержаних результатів: розроблений додаток «WeCapture» може застосовуватись для обробки візуальних даних у сферах лабораторних досліджень, страхування майна, будівництва та інших галузях діяльності де, під час роботи можуть використовуватись графічні матеріали.

Апробація результатів магістерської дисертації проводилась у міжнародній онлайн-конференції та публікації наукової статті, що зазначена нижче.

Ляшук А. М. Алгоритм напівказуальної фільтрації цифрових зображень / А. М. Ляшук // Міжнародна наукова інтернет-конференція «Інформаційне суспільство: технологічні, економічні та технічні аспекти становлення» (випуск 33). – 2018.

Ключові слова: інформаційна система, обробка зображень, розробка інформаційних систем, збереження лабораторних звітів, система синхронізації даних.

Abstract

The work presented on 109 pages consists of 3 parts, 32 figures, 22 tables and 75 sources in the list of references.

The urgency of the topic is that the market of information systems, intended for the processing of visual data with subsequent synchronization, is actively developing. Competition in the market is rather low, due to the small number of analogues. The created application "WeCapture" can be applied in the areas of property insurance, construction and other industries in which it is necessary to store graphic materials with their description in the process of work. Expanding the scope increases the chance of a potential success of the information system because for small companies a universal solution is much more profitable than developing a specialized application for their needs.

The purpose of the master's thesis is to research and develop an information system for the collection and processing of visual data that will be used in laboratory studies.

To achieve the research goal, you need to do the following:

1. Conduct an analysis of the methods of processing graphic data to improve the quality of the received images at low light.

2. Consider ways to build an information system, provide tools for designing and developing the architecture of the application.

3. To explore and analyze the following examples of the use of the library for image processing "OpenCV", using this library, algorithms are used which combine to eliminate noise, improve contrast, perform matrix transformations on the received image.

4. To highlight the plan for implementation of the self-developed architecture of the information system "WeCapture", provided step-by-step instructions for the basic work of the user with the mobile application.

Object of research: visual data obtained during the work on laboratory research

Subject of research: use of the information system when working on laboratory research.

The scientific novelty of the research is to develop a unique algorithm for semi-digital filtration of digital images.

The practical value of the results obtained: the WeCapture application developed can be applied to the processing of visual data of laboratory research, in the areas of property insurance, construction and other areas of activity where graphic materials can be used during work.

Approbation of the results of the master's dissertation was conducted in the international online conference and publication of the scientific article, which is indicated below.

Lyashuk AM Polygraphic algorithm for digital imaging / AM Lyashuk // International scientific Internet conference "Information Society: Technological, Economic and Technical Aspects of Formation" (Issue 33). - 2018

Key words: information system, image processing, information systems development, laboratory reports, data synchronization system.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

Мбайт – мегабайт

ІС – Інформаційні системи

ТПР – Типове проектне рішення

SADT – Structured Analysis and Design Technique, структурований аналіз та методика проектування

MVC – Model-View-Controller, модель-представлення-контролер

MVVM – Model-View-ViewModel, модель-представлення-модель представлення

UI – User Interface, інтерфейс користувача

POJO – Plain Old Java Object, базові типи класів у Java

1. ВСТУП

Увесь сьогоднішній прогрес, у всіх областях науки був досягнутий за допомогою кропітливої праці науковців та дослідників. Інколи, феноменальні відкриття були зроблені випадково, але більшість відкриттів – це дуже складна праця різних наукових груп. Будь-які дослідження супроводжуються великою кількістю інформації, яку необхідно зберігати. Коли проводяться десятки дослідів, то досить легко структурувати дані, але якщо ж кількість досліджень досягає сотень або тисяч, це стає серйозною задачею. Наявність візуальних даних лише ускладнює процес синхронізації та впорядкування отриманої інформації.

У магістерській роботі розглядається тільки інформаційна система, яка була розроблена для мобільних обчислювальних машин працюючих під операційною системою Android. Додаток був повністю розроблений автором даної роботи, але майнове право на нього належить фірмі роботодавцю We-Integrate. Дана фірма дала згоду на публікацію інформації про розроблений додаток, а також про інші продукти компанії, які взаємодіють з додатком, але автор не має права публікувати робочий код бізнес логіки, а також код з інших інформаційних систем.

1.1. Актуальність теми

Тема магістерської дисертації є актуальною тому, що ринок інформаційних систем, які призначенні для обробки візуальних даних з подальшою синхронізацією, активно розвивається. Конкуренція на ринку дуже низька, у зв’язку з малою кількістю аналогів.

Сьогодні будь-яке дослідження неможливо уявити без застосування комп’ютерної техніки, тому синхронізація візуальних даних між мобільними пристроями та персональним комп’ютерами є дуже актуальною задачею. У випадках коли візуальні дані не є метою дослідження, а лише способом збереження результатів отриманих при дослідах, доцільніше використовувати

не спеціалізовані фото- відеокамери, а оптичні прилади у мобільних пристроях. Саме тому, процес синхронізації отриманих графічних матеріалів є однією із складових дослідження, яку можна автоматизувати за допомогою залучення додатку.

Також, розроблюваний додаток можливо застосовувати у сферах страхування майна, будівництва та інших галузях, у яких необхідно зберігати графічні матеріали з їх описом у процесі роботи. Розширення сфер застосування збільшує шанс потенційного успіху інформаційної системи тому що, для невеликих компаній універсальне рішення набагато вигідніше, ніж розроблювати спеціалізований додаток для своїх потреб.

1.2. Огляд аналогічних інформаційних систем на ринку

На сьогоднішній день, ринок інформаційних систем, що призначені для лабораторних досліджень, низькорозвиненний. Значну долю таких інформаційних систем займають програми, що вузько спеціалізовані під певну задачу, їх виготовляють під певний тип досліджень. Саме тому, запропонована інформаційна система має дуже малу кількість аналогів.

Найкрупнішим аналогом можна вважати додаток від компанії «Preferred Patron». Дана компанія була заснована у 2004 році, але лише 5 років назад користувачі почала активно використовувати її продукти. Основним напрямком надання послуг слугують страхові компанії у Новій Зеландії, Австралії, Канаді та інших країнах. Також вони надаюсь послуги з логістики, бухгалтерії та інших економічних напрямків для невеликих компаній. Інформаційна система даної компанії була розроблена досить давно, саме тому на сьогоднішній день, її можна вважати застарілою, але база постійних клієнтів не дає «Preferred Patron» вийти з ринку.

Рис. 1.1. Логотип компанії «Preferred Patron»

Дуже широке охоплення задач шкодить функціоналу, який вважаться основним в розробленій інформаційній системі. Тому за останній рік, кількість користувачів компанії «Preferred Patron» збільшується менш інтенсивно. Також, одним із негативних факторів являється ціна. Через свою популярність, «Preferred Patron» встановила досить велику ціну, що залежить від багатьох факторів, такий як об'єм завантажуваних даних, кількість користувачів і так далі.

Іншим, потенційним аналогом можна вважати програму Image Plus розроблену компанією «Planning Plus Software». Дана австралійська компанія більш молода, з'явилась вона на ринку в 2014 році. На сьогоднішній день, вона активно шукає клієнтів, основна аудиторія розробленої системи, це невеликі фірми, які займаються ремонтом авто. На їх офіційному сайті зазначена інформація, що вони співпрацюють з дослідницькою лабораторією у Сполучених Штатах Америки, а також з невеликою компанією по ремонтам у Південній Африці.

Рис. 1.2. Логотип компанії «Planning Plus Software»

Даний додаток активно розвивається, але його графічний інтерфейс був розроблений не за стандартами Google. Керівництво компанії вирішило створити додаток однаковий для двох основних мобільних операційних систем: Android і iOS. Але загальноприйняті патерни на даних платформах значно відрізняються, саме тому отриманий в результаті продукт не можна вважати повноцінним не на одній із цих платформ. Користувачі повинні спочатку ознайомитись з інструкціями, замість того, щоб одразу почати працювати з системою.

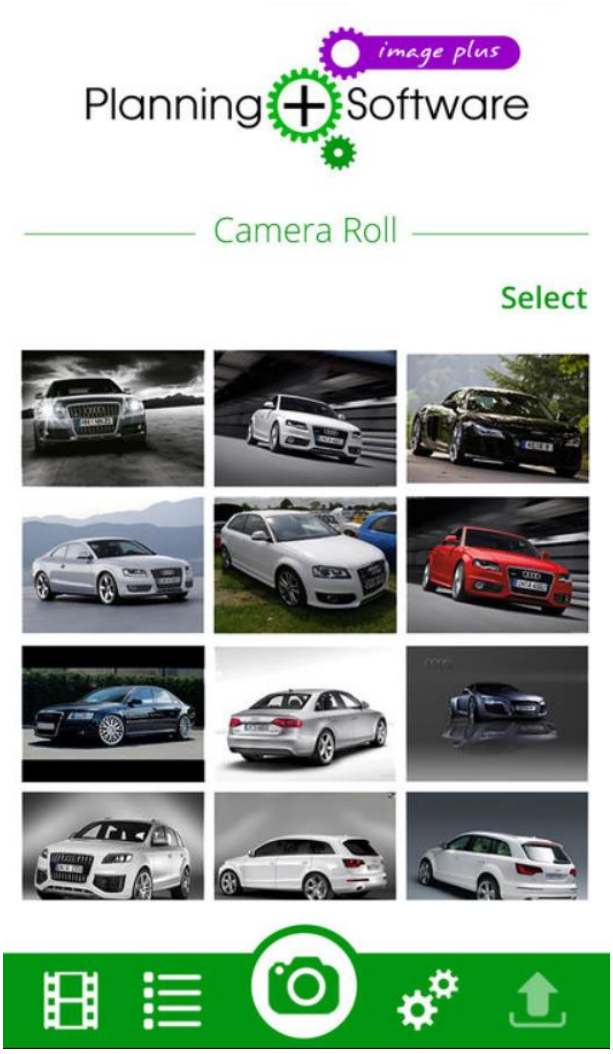


Рис. 1.3. Зовнішній вигляд додатку Image Plus

Також, функціонал даної інформаційної системи залишає бажати кращого. Камера, яка застосовується у додатку, використовується стандартна, що зменшує інтерактивність роботи з системою. Відсутні алгоритми

покращення зображення перед завантаженням звіту, відсутня система конвертації зображень у pdf, а також розпізнавання листів паперу. Головним мінусом, на мою думку, можна вважати відсутність системи, що дозволяє зберігати у звіті дані отримані при дослідженні з подальшим завантаженням у хмарне сховище.

Підвивши підсумки, можна сказати, що аналоги присутні на ринку, не становлять серйозної загрози для розроблюваної інформаційної системи. У додатку розробленому компанією «Preferred Patron» присутня велика кількість зайвого функціоналу, через бажання охопити якнайбільший сегмент ринку, через це страждає якість продукту, а також висока ціна. Розроблений графічний інтерфейс багато часу не піддавався змінам, тому нові клієнти відмовляються від впровадження даної інформаційної системи у своєму бізнесі.

Розглянувши інший аналог, розроблений компанією «Planning Plus Software», можна сказати, що йому дуже бракує функціоналу. Він розроблений під певну задачу, саме тому неможливе гнучке підлатування під різні умови використання. Враховуючи нестандартний графічний інтерфейс, відсутність обробки зображення, використання стандартної камери і багато інших недоліків, важко вважати даний аналог серйозним конкурентом.

1.3. Постановка задачі

Метою роботи є розробка спеціалізованої інформаційної системи для обробки та завантаження візуальних звітів про виконання лабораторних досліджень, що підтримує такі основні функції:

1. Обробка зображення для підвищення якості отриманих візуальних даних, в умовах низької освітленості.
2. Можливість задати поля, які користувач може заповнювати при роботі над дослідженням, з подальшим завантаженням на хмарне сховище.

3. Робота з версією Android 4.4 або вище, що дозволить встановити додаток більше ніж на 95% усіх смартфонів під операційною системою Android.
4. Підтримка роботи на планшетах для більш комфортної взаємодією з додатком.
5. Спеціалізована камера, яка дозволяє задати підказки, що візуалізуються у камері, щоб користувач міг робити заздалегідь задані послідовності знімків.
6. Система виявлення та виділення листів паперу, у автоматичному або ручному режимах, з подальшими матричними перетвореннями зображення.
7. Можливість підпису документів, перед завантаженням звіту, з подальшим доданням цього документа до звіту.

2. МЕТОДИКА ПОБУДОВИ ІНФОРМАЦІЙНИХ СИСТЕМ З ОБРОБКОЮ ЗОБРАЖЕНЬ

2.1. Цифрові зображення як складова інформаційної системи

2.1.1. Об'єкти реального світу і їх властивості

За допомогою зору ми сприймаємо образ навколишнього нас світу. Цей світ тривимірний і складається з безлічі, як правило, тривимірних об'єктів. Кожен об'єкт має оболонку (поверхню), яка відділяє його від навколишнього простору. Оболонка кожного об'єкта має властиві їй відбивні характеристики, характеристики прозорості та характеристики випромінювання, які по її поверхні можуть бути неоднорідними. Для того щоб ми могли спостерігати оточуючі нас об'єкти, вони повинні бути освітлені, якщо тільки не світяться самі.

У процесі спостереження, світло від розглянутих об'єктів, потрапляє на сітківку очей і формує на них зображення, які представляють собою центральні проекції розглянутих об'єктів. Розподіл інтенсивностей для всього спектру видимого випромінювання по поверхнях сітківки визначається відбивними характеристиками, характеристиками прозорості та характеристиками випромінювання поверхонь об'єктів, їх становищем щодо точки спостереження, спектральними характеристиками джерел освітлення, а також положенням джерел освітлення щодо спостережуваних об'єктів.

Оскільки положення спостерігача, положення джерел освітлення і спектральні характеристики джерел освітлення можуть змінюватися в широких межах, то одного й того ж об'єкту може відповідати незліченна кількість проекцій на сітківці очей.

Найбільш стійкими ознаками об'єктів на зображеннях, що формуються на сітківці очей, є їх контури, оскільки вони представляють собою проекції оболонок об'єктів, що не залежать від умов освітлення. Ось чому контурний, або силуетний, малюнок дозволяє легко дізнаватися зображуваний об'єкт [9].

Зорова система при спостереженні тривимірної сцени формує її тривимірне уявлення в корі головного мозку. Це необхідно вже хоча б для того, щоб ми мали можливість маніпулювати різними об'єктами, і не стикатися з ними при своєму переміщенні в просторі, а також розпізнавати їх при спостереженні під різними ракурсами.

Для формування в зоровій системі об'ємного уявлення об'єктів, що становлять сцену, наявність бінокулярного зору не є обов'язковим. Об'ємне уявлення тривимірних об'єктів здатні формувати і люди, які не мають можливості бінокулярно розглядати спостережувану сцену, а також багато тварин і птиці, взагалі позбавлені бінокулярного зору. Оскільки при формуванні об'ємного уявлення незнайомого тривимірного об'єкту в зоровій системі єдиним джерелом інформації про нього є його двовимірні проекції на сітківці очей, то для цього необхідно мати у своєму розпорядженні поруч таких проекцій, отриманих, наприклад, при різних ракурсах спостереження.

Зі сказаного випливає, що при спостереженні сцен, зображених, наприклад, на фотографіях, на сітківці очей створюються проекції подібні тим, що створюються при безпосередньому спостереженні самих зображуваних сцен, завдяки чому створюється враження близько до того, яке має місце при спостереженні зображуваної натури. На цьому, власне, і базуються такі види образотворчого мистецтва, як живопис, фотографія і кіно.

2.1.2. Двовимірна растрова модель зображення

У комп'ютерній графіці використовують растрову і векторну моделі двовимірного зображення [2]. В основі растрової моделі лежить растр - матриця пікселів, які представляють інтенсивність відповідних ділянок зображення.

На рис. 2.1 наведено фрагмент растрового зображення з ортогональним розташуванням пікселів, лінійні розміри яких в обох напрямках однакові. Крім растра з ортогональним розташуванням пікселів можливі і інші растри,

наприклад, растр з шаховим розташуванням пікселів, растр з діагональним розташуванням пікселів, проте вони застосовуються рідко.

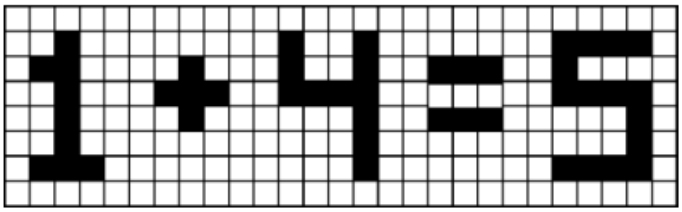


Рис. 2.1. Растрове зображення.

Растрове зображення характеризується роздільною здатністю, яка визначається кількістю пікселів на одиницю довжини. Чим більше пікселів припадає на одиницю довжини, тим вище роздільна здатність і тим більше дрібні деталі можуть бути відтворені на зображенні. Для того щоб зображення можна було обробляти за допомогою комп'ютера, його представляють в цифровій формі. У зображенні інтенсивність (яскравість) кожного пікселя представляється числом, яке зазвичай лежить в межах від 0 до 255. Напівтонове чорно-біле зображення представляється у вигляді двовимірної матриці, що складається зазвичай з 8-розрядних двійкових чисел. Для роботи з кольоровим зображенням в цифровій формі використовують уже три матриці, кожна з яких, як правило, складається з 8-розрядних двійкових чисел, рідше з 16-розрядних. При цьому елементи кожної з цих матриць представляють інтенсивності червоного, зеленого і синього компонентів кольору пікселя, оскільки колір кожного з пікселів растра синтезується шляхом змішування червоного, зеленого і синього кольорів, як в палітрі художника.

При виборі роздільної здатності зображення виходять з того, щоб глядачеві зображення здавалося безперервним. Щоб глядач не бачив на зображенні растрової структури. З цією метою кількість пікселів на одиницю довжини в зображенні вибирають, виходячи з гостроти (роздільної здатності) зору. Відомо, що при спостереженні двох об'єктів, кутова відстань між якими менше однієї кутової хвилини, вони зливаються в один об'єкт. З цього

випливає, що для відсутності помітності растрової структури кутова відстань між її пікселями β має бути менше однієї кутової хвилини або, в крайньому випадку, дорівнювати їй.

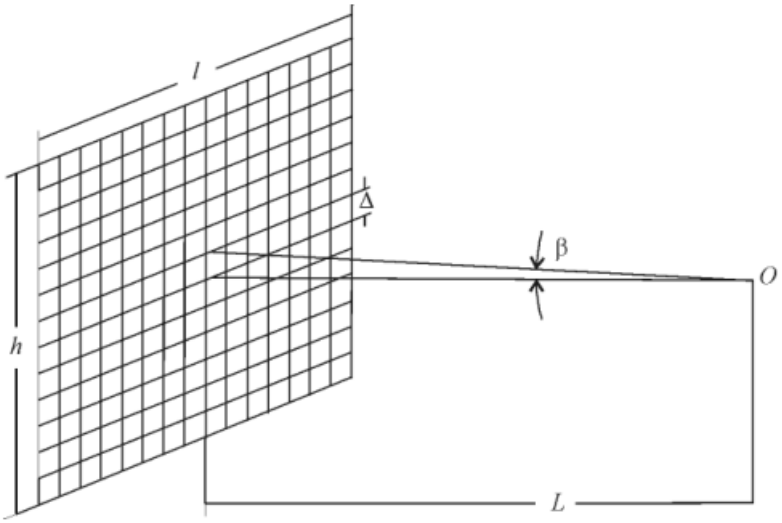


Рис. 2.2. До визначення вибору роздільної здатності.

Оскільки кут β залежить від відстані спостереження, лінійна відстань між центрами пікселів Δ (крок просторової дискретизації зображень) також залежить від цього кута. Записуючи очевидне геометричне співвідношення $L / \Delta = tg\beta$, маємо $\Delta = L * tg\beta$, при цьому загальне число пікселів N, дорівнюватиме $N = h * l/\Delta^2$, що після підстановки значення Δ дає $N = h * l/(L * tg\beta)^2$.

Оскільки tg від однієї хвилини дорівнює $2{,}909*10^{-4}$, отримаємо остаточно

$$N = \frac{11817146*h*l}{L^2}. \quad (2.1)$$

а число рядків z в зображенні, як неважко бачити, дорівнюватиме

$$z = \frac{3437*h}{L}. \quad (2.2)$$

З виразу (2.2) випливає, що мінімально необхідну кількість пікселів, що становлять растр зображення, визначається не тільки його розміром, але також сильно залежить від умови спостереження зображення, тобто відстанню, з якого ведеться спостереження. У телебаченні відстань спостереження

приймається рівним $L \cong (5..6)h$. Згідно телевізійного стандарту відношення ширини зображення до його висоти $1/h$ прийнято близьким до 4/3, число рядків в телевізійному зображенні відповідно до наведених розрахунками повинно бути $z \cong 574..689$, що близько до прийнятого за стандартом значенням 625[3]. Аналогічним чином вирішується завдання стосовно комп'ютерних дисплеїв, але з урахуванням специфіки спостереження зображення на екрані монітора.

Кількість растрових елементів в зображенні поряд з розрядністю двійкового коду, за допомогою якого представляється яскравість (в разі кольорового зображення інтенсивності червоного, зеленого і синього кольорів), визначає необхідний обсяг пам'яті для зберігання зображення. У разі чорно-білого напівтонового зображення (grayscale) необхідний обсяг пам'яті дорівнює $M = 8N$ біт, в разі кольорового зображення з використанням моди RGB (True Color) необхідний обсяг пам'яті складе вже $M = 24N$ біт. Так, наприклад, для того щоб зберегти кольорове RGB-зображення розміром 1000 × 1000 пікселів, буде потрібно близько 3 Мбайт пам'яті.

2.1.3. Види зображень

Нерухоме ахроматичне зображення являє собою функцію, що описує розподіл яскравості L_c на площині, тобто $L_c(x, y)$, де x і y - декартові координати. Для уявлення рухомого зображення в написаному вираженні додається ще одна незалежна змінна - час t, а запис приймає $L_c(x, y, t)$. Кольорові зображення натурних сцен, отримані в результаті їх оптичної проекції на будь-яку поверхню, наприклад на світлочутливу поверхню датчика телефону, будуть в числі незалежних змінних містити ще й довжину хвилі світлового випромінювання λ. В цьому випадку для кольорового нерухомого зображення матимемо $L_c(x, y, \lambda)$, а для рухомого $L_c(x, y, t, \lambda)$, де L_c слід розглядати як інтенсивність випромінювання на довжині хвилі λ, в точці з

координатами x і y, в момент часу t. Аналогічним чином можна перейти до опису "об'ємних" зображень, додавши ще одну просторову координату z.

Однак сучасна техніка передачі, обробки і демонстрації зображень заснована на їх уявленні у вигляді ряду компонентів. Так, наприклад, кольорове зображення, призначене для виведення на екран смартфона або на екран монітора комп'ютера, представляється у вигляді 3-х канальних зображень: червоного, зеленого і синього. У поліграфії, число каналів у зображенні може бути великим. Аналогічним чином йде справа з рухомими зображеннями, які, представляються послідовністю нерухомих, швидко змінюють один одного зображень, на кожному з яких зафіксована відповідна фаза руху. Швидка зміна цих зображень створює ілюзію руху.

У загальному випадку, послідовність нерухомих зображень, якій видаються реальні кольорові рухомі зображення, володіє однією чудовою особливістю - всі вони описуються дуже схожими характеристиками. Ця особливість дозволяє в подальшому зосередитися на розгляді властивостей і методів обробки нерухомих ахроматичних зображень, поширюючи отримані результати на кольорові зображення, та лише в необхідних випадках виходити за ці рамки.

Зображення прийнято розділяти на два класи: семантичні (смислові) і текстурні. Приклади цих зображень наведені на рис. 2.3 і 2.4 відповідно.

Рис. 2.3. Приклад семантичного зображення.

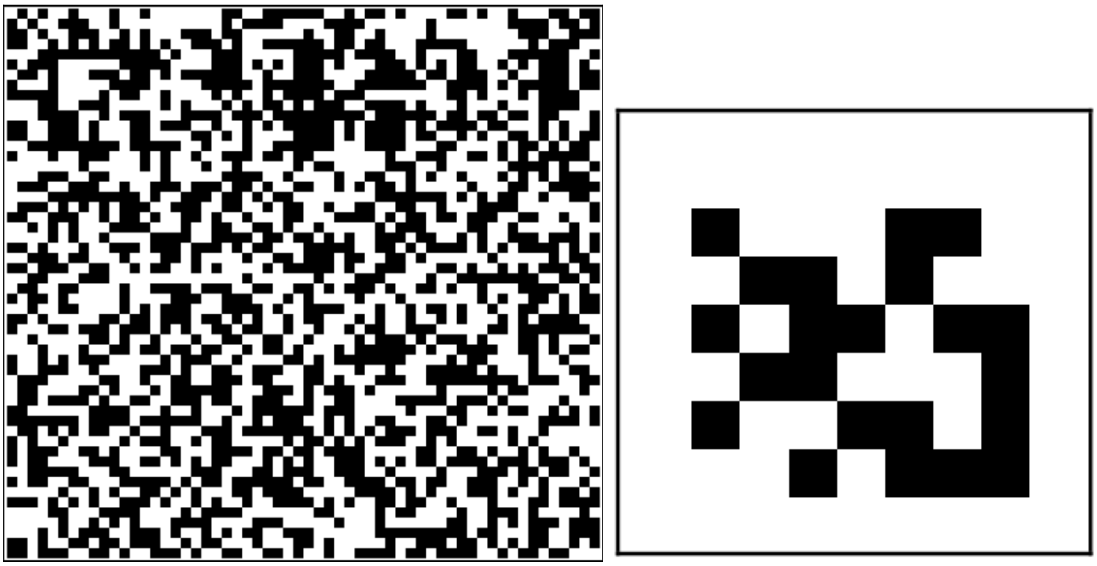


Рис. 2.4. Приклад текстурного зображення.

У процесі тривалої еволюції, зорова система людини пристосувалася виявляти, розпізнавати та класифікувати не будь-які довільні розподіли яскравості, спроектовані зіницею на сітківку ока, а лише ті, які створюються об'єктами зовнішнього світу. У цьому легко переконатися, якщо спробувати виявити шумовий візерунок ("шумовий об'єкт"), показаний на рис. 2.4 на тлі

шумового поля. Це завдання вирішується за рахунок поелементного порівняння обох зображень, тоді як на рис. 2.4 будь-який об'єкт знаходиться легко і швидко. Зазначена особливість зору широко використовується в природі для цілей камуфляжу. Так, наприклад, неправильної форми смуги на шкірі тигра роблять його нерозбірливим в заростях.

Характерною особливістю зображень реальних об'єктів є те, що вони складаються з областей, розділених більш-менш різкими світловими межами, всередині яких яскравість і колір змінюються порівняно повільно. Ці світлові кордони (контури) передають форму об'єкта і є основою для його розпізнавання. З досвіду відомо, що інформація яка міститься в контурах, як правило, цілком достатньо для безпомилкового впізнавання об'єкта. Так, наприклад, ми легко впізнаємо особу знайомої людини по контурному малюнку.

Знайдемо зв'язок між структурою зображень реальних об'єктів і їх просторовими спектрами, отриманими в результаті інтегрального перетворення Фур'є. З цією метою розглянемо спектри трьох різних по різкості світлових кордонів, орієнтованих перпендикулярно до осі x. Оскільки в даному випадку яскравість зображень не залежить від координати y, завдання можна істотно спростити, звівши її до одновимірної. На рис. 1.5 наведені три різні залежності зміни яскравості на кордоні від координати x, які описуються виразами 1.3:

$$L_1(x) = \begin{cases} 0 \text{ при } x \le x_0, \\ L \text{ при } x > x_0, \end{cases}$$

$$L_2(x) = \begin{cases} \frac{L}{2} - \frac{L}{2} exp\left[\alpha(x - x_{0)}\right] \text{ при } x \le x_0, \\ \frac{L}{2} + \frac{L}{2} exp\left[-\alpha(x - x_{0)}\right] \text{ при } x > x_0, \end{cases} \tag{2.3}$$

$$L_3(x) = \begin{cases} \dfrac{L}{2} exp\left[\alpha(x - x_{0)}\right] \text{ при } x \le x_0, \\ L - \dfrac{L}{2} exp\left[-\alpha(x - x_{0)}\right] \text{ при } x > x_0. \end{cases}$$

Визначаючи спектри функцій L(x, y), в результаті перетворень отримаємо:

$$M_1(\omega_x) = (L/\omega_x)\exp[-i(\pi/2 + \omega_x x_0)],$$
$$M_2(\omega_x) = L\pi\delta(\omega_x) + [(\mathrm{L}\omega_x)/(\alpha^2 + \omega_x^2)]\exp[-i(\pi/2 + \omega_x x_0)], \qquad (2.4)$$
$$M_3(\omega_x) = \{(L\alpha^2)/[\omega_x(\alpha^2 + \omega_x^2)]\}\exp[-i(\pi/2 + \omega_x x_0)],$$

де ω_x - кругова просторова частота, $i = \sqrt{-1}$.

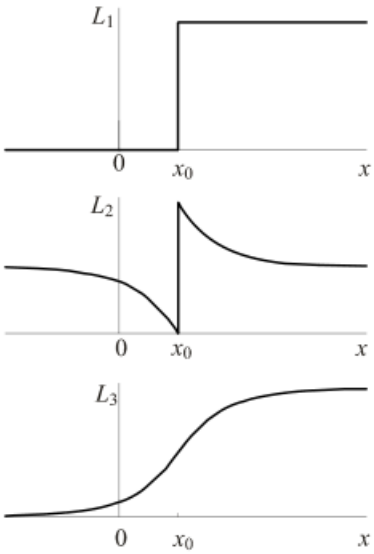


Рис. 2.5. Приклади розподілу яскравості на світових кордонах.

Перше, що звертає на себе увагу - ідентичність фазових спектрів для всіх трьох кордонів. Неважко бачити, що інформація про наявність і стан світлового кордону укладена в фазовому спектрі. Амплітудний спектр не містить інформації про становище кордону, проте в ньому міститься інформація про різкість зображення[4]. З досвіду роботи з зображеннями відомо, що зображення може бути піддано значним лінійним і нелінійним спотворенням, але якщо при цьому спотворення фазового спектра будуть невеликі, так що вони не викличуть зникання існуючих або появи нових світових кордонів, зображення буде залишатися легко впізнаваним. До таких спотворень відносяться інтегрування, яке призводить до втрати чіткості зображень, диференціювання, що приводить до підкреслення кордонів на зображенні, поелементне перетворення виду $u = f(v)$ (де $f(v)$ — монотонна

функція), що приводить до зміни контрасту, і ряд інших. Якщо ж в результаті перетворення зображення істотно спотворюється його фазовий спектр, то може мати місце втрата впізнаваності зображуваного об'єкта.

Наочний приклад цього - втрата розрізнення зображення при впливі на нього флуктуаційна шумом, що має більшу дисперсію. В результаті такого впливу щільність ймовірності розподілу фази частотних складових зашумленної реалізації наближається до рівномірної на інтервалі 0 ... 2π, що тягне за собою повне розмиття світлових кордонів на зображенні.

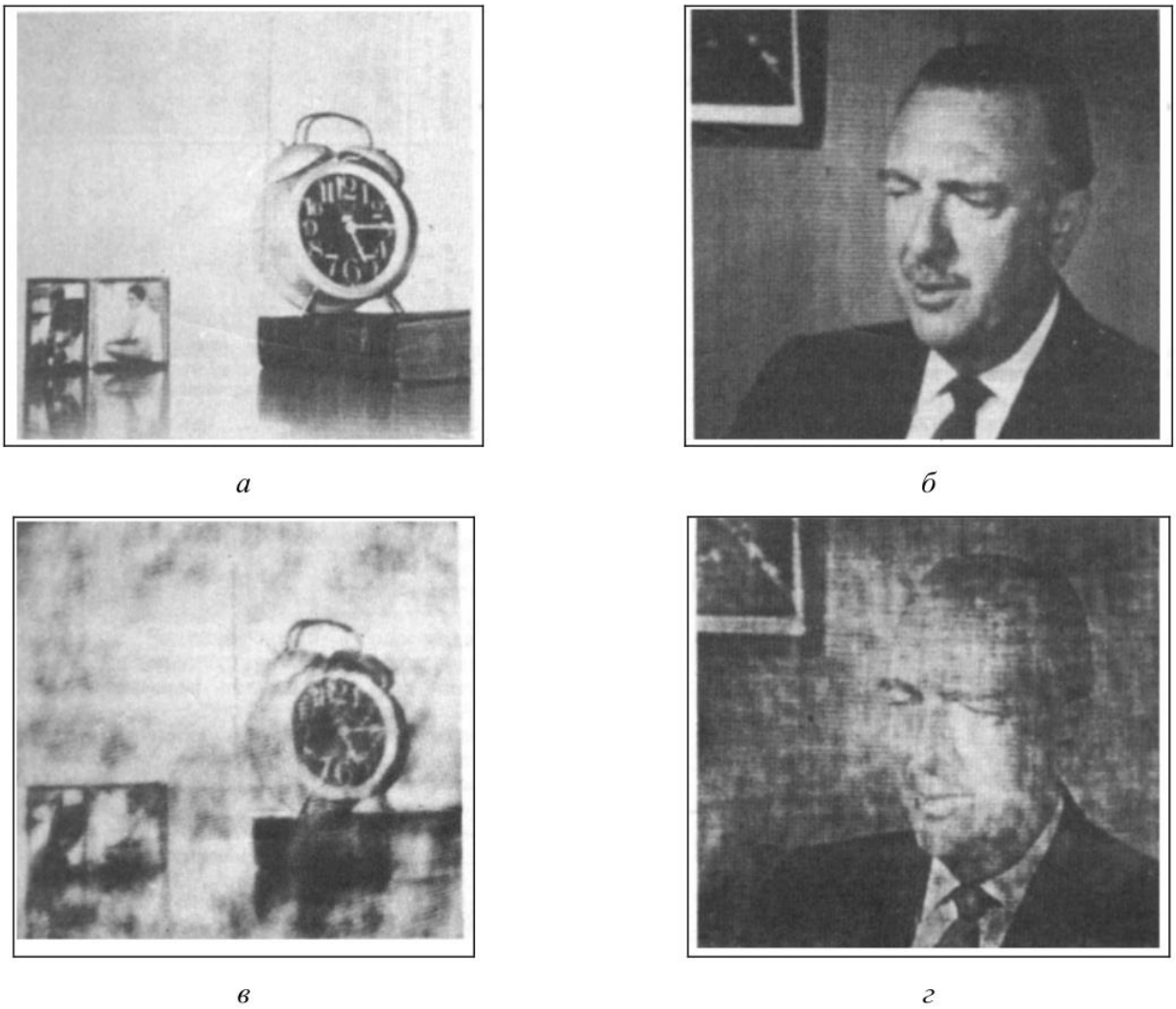


Рис. 2.6. Вихідні (а) і (б) і синтезовані (в) і (г) зображення. Зображення запозичені з [5].

Найбільш переконливим доказом наведених раніше положень служить експеримент з відновленням зображень по "переплутаним" амплітудним і

фазовим спектрами [5]. Експеримент полягав в тому, що для зображень, показаних на рис. 2.6, а і б, замінили амплітудні і фазові спектри наступним чином:

$$M_{\text{в}}(\omega_X, \omega_y) = |M_{\text{б}}(\omega_X, \omega_y)| \exp[-i\varphi_a(\omega_X, \omega_y)], \quad (2.5)$$
$$M_{\text{г}}(\omega_X, \omega_y) = |M_{\text{а}}(\omega_X, \omega_y)| \exp[-i\varphi_{\text{б}}(\omega_X, \omega_y)],$$

після чого за спектрами $M_{\text{В}}(\omega_X, \omega_y)$і $M_{\Gamma}(\omega_X, \omega_y)$ синтезувалися зображення, показані на рис. 2.6, в і г. З малюнка видно, що заміна амплітудних спектрів призвела лише до деякого зашумлення зображень без втрати їх розрізнення.

На підставі викладеного можна зробити висновок, що при передачі і обробці зображень особливу увагу слід приділяти точності передачі фазового спектра. У телебаченні до цього висновку прийшли давно, чисто досвідченим шляхом, помітивши, що амплітудно-частотні та амплітудні (нелінійні) спотворення менш помітні на зображенні, ніж фазо-частотні.

При вирішенні ряду задач прикладного характеру доводиться мати справу з розрізненням текстурних полів на зображенні. Останнім часом багато уваги приділяється аналізу зображень, отриманих при дистанційному зондуванні Землі, де питання розрізнення текстур займає провідне місце. В результаті експериментальних досліджень було виявлено, що людина здатна розрізняти текстурні поля, якщо вони різняться між собою одновимірними щільностями ймовірностей розподілу яскравості або якщо при однакових одновимірних щільностях розподілу ймовірностей є відмінність у функціях автокореляції. Якщо ж текстурні поля розрізняються тільки щільністю розподілу ймовірностей третього або більш високого порядку, то вони візуально не помітні. Це положення носить назву гіпотези Юлеша. Згодом поруч дослідників були знайдені приклади текстур, що суперечать цій гіпотезі, хоча, як зазначають самі дослідники, візуально ці текстури важко помітні [6].

2.1.4. Моделі зображень

В теорії цифрової обробки зображень в залежності від розв'язуваної задачі використовують різні моделі зображень. Під моделлю зображення розуміється комплекс характеристик, що може описувати розподіл яскравості на площині, якими апроксимується розглянута область зображень. Модель повинна задовольняти суперечливим вимогам дуже близьких до реальних зображень і простоти аналізу. При виборі моделі істотне значення має також ступінь спільності результатів, які можуть бути отримані при її використанні. Залежно від того, яке з вимог набуває більшого значення, використовують ту чи іншу модель зображення різного ступеня складності. Як приклад наведемо одну з моделей, яка застосовується при синтезі алгоритмів нелінійної обробки зображень:

$$L_c(x, y) = k * E(x, y) * r_c(x, y), \qquad (2.6)$$

де $r_c(x, y)$ - коефіцієнт відображення різних ділянок сцени, відповідних її проекції на світлочутливу поверхню датчика сигналу зображення з координатами (x, y); E(x, y) - освітленість різних ділянок сцени, відповідних її проекції на світлочутливу поверхню датчика сигналу зображення з координатами (x, y); k - коефіцієнт, що погоджує розмірності.

Коефіцієнт відображення $r_c(x, y)$ - це функція, що характеризується наявністю різких стрибків, що виникають на контурах, в той час як E(x, y) в основному дуже повільно змінюється. Завдяки цьому спектр $r_c(x, y)$ є широкосмуговим, а спектр функції E(x, y) вузькосмуговим, що використовується, наприклад, в цілях "поліпшення" якості зображення. Іншою важливою властивістю функцій $r_c(x, y)$ і E(x, y), є їх невід’ємність, що забезпечує невід’ємність $L_c(x, y)$. Властивість невід’ємності $L_c(x, y)$ накладає сильні обмеження на вибір можливих алгоритмів обробки, оскільки результат обробки, оброблене зображення, також повинен бути позитивною функцією, так як негативні значення яскравості фізично не реалізовуються.

Хороша модель зображення є надійною основою для синтезу ефективних алгоритмів обробки зображень, і навпаки, невдалі моделі часто виявлялися причиною невдач при розробці таких алгоритмів.

2.1.5. Просторові спектри зображень

При аналізі лінійних спотворень зображень, а також при вирішенні завдань, пов'язаних з виявленням і розпізнаванням об'єктів, надзвичайно корисно використовувати поняття спектрів зображень і їх попарних різниць при суміщенні. Під суміщенням зображень розуміється таке поєднання, при якому середній квадрат їх попіксельної різниці $\overline{[L_c(x,y,\imath) - L_c(x,y,\jmath)]^2}$ досягає мінімуму, де $L_c(x, y, i)$ і $L_c(x, y, j)$ - розподіл яскравості в i-му і j-му зображеннях [4].

Спектр j-го зображення $M_c(\omega_x, \omega_y, j)$ за визначенням є комплексною функцію, пов'язаною з розподілом яскравості на зображенні $L_c(x, y, j)$ парою перетворень Фур'є:

$$M_c(\omega_X, \omega_y, j) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} L_c(x,y,j) exp\left[-i(\omega_X x + \omega_y y)\right] dxdy, \qquad (2.7)$$

$$L_c(x,y,j) = \frac{1}{4\pi^2}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} M_c(\omega_X, \omega_y, j) exp\left[i(\omega_X x + \omega_y y)\right] d\omega_X d\omega_y,$$

де $i = \sqrt{-1}$, ω_x і ω_y - кругові просторові частоти спектра в напрямку осей x і y.

Аналогічно визначимо спектр різниці двох зображень (j-го і i-го) при їх суміщенні:

$$M_\Delta(\omega_X, \omega_y, j, i) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}[L_c(x,y,j) - L_c(x,y,i)] exp\left[-i(\omega_X x + \omega_y y)\right] dxdy, \qquad (2.8)$$

$$L_c(x,y,j) - L_c(x,y,i) = \frac{1}{4\pi^2}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} M_\Delta(\omega_X, \omega_y, j, i) exp\left[i(\omega_X x + \omega_y y)\right] d\omega_X d\omega_y,$$

Спектри різниці двох зображень при їх суміщенні використовуються при вирішенні завдань, пов'язаних з розпізнаванням об'єктів при наявності шуму

на зображеннях. Певні спектри містять повну інформацію як про амплітудних, так і про фазових частотних складових. Розподіл яскравості на репродукції $L_{c\Omega}(x, y, j)$, відтворюється лінійної системою, що вносить спотворення, може бути знайдено, виходячи з відомого розподілу яскравості в вихідному $L_c(x, y, j)$ за допомогою інтеграла згортки (інтеграла Дюамеля):

$$L_{c\Omega}(x, y, j) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} L_c(\xi, \eta, j)h(x - \xi, y - \eta)d\xi d\eta, \qquad (2.9)$$

де ξ і η - змінні інтегрування; h (x, y) - імпульсна характеристика системи, яка з точністю до постійного множника збігається з розподілом яскравості на репродукції при передачі лінійною системою зображення точки. Імпульсна характеристика h(x, y), або функція розсіювання точки, в термінології оптичних систем, повністю характеризує спотворення, що вносяться лінійної системою.

При цьому спектри вихідного і відтвореного лінійною системою зображень пов'язані між собою співвідношенням

$$M_{c\Omega}(\omega_x, \omega_y, j) = M_c(\omega_x, \omega_y, j)K(\omega_x, \omega_y), \qquad (2.10)$$

де $M_{c\Omega}(\omega_x, \omega_y, j)$ — спектр $L_{c\Omega}(x, y, j)$; $K(\omega_x, \omega_y)$ — частотно-передавальна функція розглянутої лінійної системи, пов'язана з імпульсною характеристикою пари перетворень Фур'є:

$$K(\omega_X, \omega_y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h(x, y)exp[-i(\omega_X x + \omega_y y)]dxdy, \qquad (2.11)$$

$$h(x, y) = \frac{1}{4\pi^2}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} K(\omega_X, \omega_y)exp[i(\omega_X x + \omega_y y)]d\omega_X d\omega_y,$$

Аналогічно знаходиться спектр різниці зображень, відтворюваних лінійної системою при їх суміщенні:

$$M_{\Delta\Omega}(\omega_x, \omega_y, j, i) = M_\Delta(\omega_x, \omega_y, j, i)K(\omega_x, \omega_y). \qquad (2.12)$$

2.1.6. Перешкоди і їх статистичні характеристики

Перешкоди, впливаючи на сигнал зображення, викликають появу на зображенні флуктуацій яскравості, які спотворюють його, а при високому рівні перешкод роблять зображення невиразним. Найбільш поширеним видом

перешкод є адитивні, при яких сигнал U_c і перешкода $U_ш$, підсумовуючись алгебраїчно, утворюють зашумлену реалізацію $U = U_c + U_ш$. Вплив перешкоди на сигнал може також виражатися в їх перемножуванні (мультиплікативна перешкода), при цьому

$$U = c_1 * U_c\, U_ш, \tag{2.13}$$

де c_1 - множник, введений для узгодження розмірностей. Можливі й інші випадки.

Перешкоду можна розглядати як випадкову функцію часу t при впливі її на електричний сигнал або як випадкову функцію координат x і y при впливі на зображення. Якщо перешкода є функцією дискретного часу або дискретних координат, як це має місце при передачі зображень по цифровому каналу, говорять про довільні послідовності. В іншому випадку випадкову функцію називають випадковим процесом. Випадкові функції описуються багатовимірними щільностями ймовірності.

$$W_n(U_{ш1}, U_{ш2}, \dots, U_{шk}, \dots), \tag{2.14}$$

де $U_{ш1}, U_{ш2}, \dots, U_{шk}$ - миттєві значення перешкоди, які вона приймає в моменти часу $t_1, t_2, \dots, t_k$. Подібним же чином записують багатовимірну щільність ймовірності перешкоди, що представляє флуктуацію яскравості на зображенні. Перешкоди називаються стаціонарними, якщо їх статистичні характеристики не залежать від часу (від координат).

Для обчислення середнього значення перешкоди $\overline{U_ш}$ і її середнього квадрата $U_ш^2$ які називають відповідно першим і другим початковими моментами розподілу, досить знати її одновимірну щільність ймовірності $W_n(U_ш)$ при цьому

$$\overline{U_ш} = \int_{-\infty}^{\infty} U_ш\, W_n(U_ш) dU_ш, \tag{2.15}$$

$$\overline{U_ш^2} = \int_{-\infty}^{\infty} U_ш^2 W_n(U_ш) dU_ш.$$

риска над $U_ш$ і $U_ш^2$ позначено усереднення цих величин. У ряді випадків більш зручним буває використовувати так звані центральні моменти розподілу, тобто ті моменти, центровані щодо середнього значення перешкоди $\overline{U_ш}$. Так вираз 2.16,

$$\overline{(U_ш - \overline{U_ш})^2} = \int_{-\infty}^{\infty} (U_ш - \overline{U_ш})^2 \, W_n(U_ш) dU_ш. \qquad (2.16)$$

являє собою другий центральний момент розподілу, або дисперсію, яку будемо позначати через вираз 2.17,

$$\sigma^2 = \overline{(U_ш - \overline{U_ш})^2}. \qquad (2.17)$$

Серед випадкових перешкод особливе положення займає стаціонарна флуктуаційна перешкода (або, як її часто називають, флуктуаційний шум), яка виникає в датчиках сигналу зображення (наприклад, в передавальних телевізійних матрицях), в перших каскадах попередніх підсилювачів передавальних телевізійних камер, у вхідних ланцюгах радіоприймальних пристроїв і т. д., і розподілена по нормальному закону. Флуктуаційна перешкода має властивість ергодичності, що полягає в тому, що середні по ансамблю $\overline{U_ш}$, $\overline{U_ш^2}$ з імовірністю, як завгодно близькою до одиниці, збігаються із середніми по часу. Статистичні властивості стаціонарної флуктуаційної перешкоди повністю описуються двовимірною щільністю ймовірності

$$W_n(U_ш, U_{шτ} = \frac{1}{2\pi\sigma^2\sqrt{1-\rho_ш^2(\tau)}} exp\left[-\frac{U_ш^2 - 2\rho_ш(\tau)U_ш U_{шτ} + U_{шτ}^2}{2\sigma^2\left[1-\rho_ш^2(\tau)\right]}\right], \qquad (2.18)$$

де $\rho_ш(\tau)$ коефіцієнт автокореляції перешкоди, τ - інтервал часу, на який зміщені значення $U_ш$ і $U_{шτ}$. Коефіцієнт автокореляції перешкоди $\rho_ш(\tau)$ пов'язаний з її функцією автокореляції співвідношенням

$$R_ш(\tau) = \rho_ш(\tau)\sigma^2. \qquad (2.19)$$

і встановлює статистичний зв'язок між значеннями перешкоди $U_ш$, рознесеними в часі на інтервал τ. Функція автокореляції є парною функцією часу τ і пов'язана зі спектральною інтенсивністю перешкоди $S_ш(\omega)$ парою перетворень Фур'є. Звернемо увагу, що

$$S_ш(\omega) = \int_{-\infty}^{\infty} R_ш(\tau) \exp(-i\omega\tau) d\tau. \qquad (2.20)$$

Іноді замість спектральної інтенсивності перешкоди $S_{ш}(\omega)$ використовують її енергетичний спектр $F_{ш}(\omega)$, який визначають наступним чином:

$$F_{ш}(\omega) = \begin{cases} \frac{S_{ш}(\omega)}{\pi} \text{при } \omega \geq 0 \\ 0 \text{ при } \omega < 0 \end{cases}, \tag{2.21}$$

В цьому випадку

$$\sigma^2 = \int_0^{\infty} F_{ш}(\omega) d\omega. \tag{2.22}$$

Як відомо, спектральна інтенсивність дробового і теплового шумів, які є першопричиною флуктуацій, не залежить від частоти в дуже широких межах. У зв'язку з цим вводять поняття (модель) так званого білого шуму, який при нульовому середньому значенні, при $\overline{U_{ш}} = 0$, і нормальному розподілі за рівнями має спектральну інтенсивність, не залежну від частоти в межах від -∞ до ∞.

Поряд з цим часто користуються поняттям квазібілого шуму, який відрізняється від білого шуму тим, що його спектральна інтенсивність постійна в інтервалі частот від $-\omega_{гр}$ до $\omega_{гр}$ і дорівнює нулю за його межами. Можна показати шляхом застосування перетворення Фур'є до його спектральної інтенсивності, що коефіцієнт автокореляції квазібілого шуму дорівнює

$$\rho_{ш} = \sin \omega_{гр}\tau / \omega_{гр}\tau. \tag{2.23}$$

Ця залежність представлена графічно на рис. 2.7.

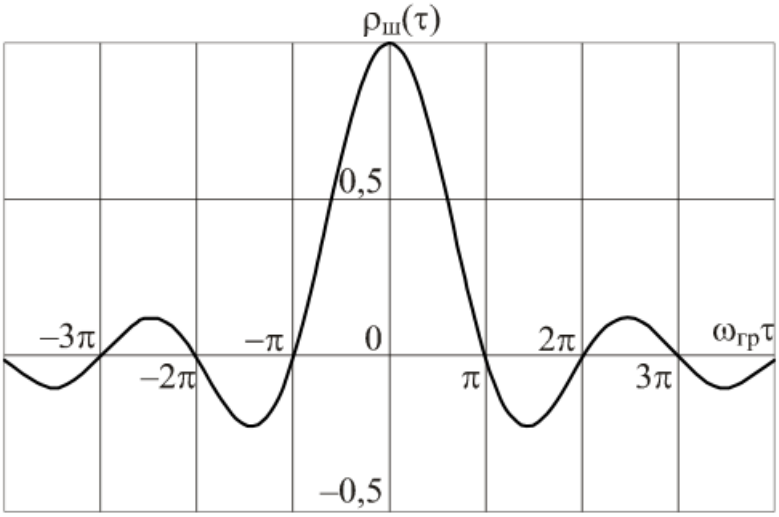


Рис. 2.7. Залежність коефіцієнта автокореляції квазібілого шуму від $\omega_{\text{гр}}\tau$.

На відміну від квазібілого шуму функція кореляції білого шуму виражається через дельта-функцію

$$R_{\text{ш}}(\tau) = \frac{S_{\text{ш}}(0)}{2\pi}\int_{-\infty}^{\infty}\exp(i\omega\tau)\,d\omega = S_{\text{ш}}(0)\delta(\tau). \qquad (2.24)$$

Оскільки

$$\frac{1}{2\pi}\int_{-\infty}^{\infty}\exp(i\omega\tau)\,d\omega = \delta(\tau). \qquad (2.25)$$

з чого випливає, що будь-які два відліку, взяті на кінцевому інтервалі часу τ, виявляються некоррельованими.

Все викладене залишається справедливим і для випадку, коли квазібілий і білий шуми є флуктуацією яскравості на зображенні, тобто є двовимірними. При цьому спектральна інтенсивність двовимірного квазібілого шуму

$$S_{\text{ш}}(\omega_x, \omega_y) = \begin{cases} S_{\text{ш}}(0{,}0) \text{ при } |\omega_{\text{x}}| \le \omega_{\text{хгр}} \text{і } |\omega_{\text{y}}| \le \omega_{\text{угр}}, \\ 0 \qquad \text{у всіх інших випадках,} \end{cases} \qquad (2.26)$$

Іншою корисною моделлю перешкоди, з якою доводиться мати справу на практиці, є перешкода, описувана рівномірним законом розподілу за рівнями на деякому інтервалі $-\frac{\delta_{\text{кв}}}{2}\ldots\frac{\delta_{\text{кв}}}{2}$. Перешкоди такого роду виникають при квантуванні зображень за інтенсивністю з кроком $\delta_{\text{кв}}$ при їх оцифровуванні. Шум на зображеннях, отриманих з використанням лазерів,

може мати як експоненціальне розподіл, так і розподіл Ерланга [7]. У першому випадку щільність ймовірності розподілу шуму описується виразом

$$W(L_{ш}) = \begin{cases} \frac{1}{\overline{L_{ш}}} \exp(-\frac{L_{ш}}{\overline{L_{ш}}}) \text{ при } L_{ш} \geq 0, \\ 0 \qquad \text{при } L_{ш} \leq 0, \end{cases} \tag{2.27}$$

де $\overline{L_{ш}} > 0$ — середнє значення шуму. Дисперсія шуму при цьому дорівнює

$$\sigma^2 = (\overline{L_{ш}})^2.$$

При розподілі Ерланга, щільність ймовірності шуму описується виразом

$$W(L_{ш}) = \begin{cases} \frac{a^b L_{ш}^{b-1}}{(b-1)!} \exp(-aL_{ш}) \text{ при } L_{ш} \geq 0, \\ 0 \qquad \text{при } L_{ш} \leq 0, \end{cases} \tag{2.28}$$

де 0 > a, b - натуральне число, а символ "!" позначає факторіал. Середнє значення і дисперсію шуму при цьому рівні відповідно

$$\overline{L_{ш}} = \frac{b}{a}, \sigma^2 = \frac{b}{a^2}. \tag{2.29}$$

Наступним видом перешкоди, є адитивна біполярна імпульсна перешкода. Виникнення імпульсної біполярної перешкоди часто обумовлено комутаційними процесами в електричних схемах, а також процесами електричних розрядів. Розподіл ймовірності p_n біполярної імпульсної перешкоди описується виразом

$$p_n = \begin{cases} p_a \text{ при } L_{ш} = L_a \\ p_б \text{ при } L_{ш} = -L_б \\ 0 \text{ в усіх інших вападках} \end{cases} \tag{2.30}$$

де p_a - ймовірність виникнення позитивного імпульсу перешкоди, величина якого дорівнює L_a, $p_б$ - ймовірність виникнення негативного імпульсу перешкоди, величина якого дорівнює - $L_б$. У тому випадку якщо одна з ймовірностей p_a або $p_б$ дорівнює нулю, біполярна перешкода перетворюється в уніполярну. Позитивні імпульси перешкоди виглядають на зображенні у вигляді білих точок, а негативні - у вигляді чорних крапок. Біполярну імпульсну заваду в перекладній літературі, наприклад в посібниках з MATLAB, часто називають перешкодою типу "сіль і перець", оскільки вона нагадує розсипані на зображенні крупиці солі і перцю. Однією з особливостей імпульсної біполярної перешкоди, з якою досить часто доводиться

зустрічатися, є те, що величина її імпульсів виявляється більше розмаху сигналу зображення. В результаті цього виникає обмеження інтенсивності тих пікселів зображення, на які потрапили імпульси перешкоди і вони стають або білими, або чорними.

2.1.7. Джерела флуктуаційного шуму в цифрових фотокамерах на ПЗС

Фактором, що обмежує світлову чутливість цифрових фото- і відео-камер, є флуктуаційний шум. Цей шум складається з двох компонентів: компонента шуму, що виникає в перетворювачі зображення в сигнал, і компонента шуму, наявного в світловому потоці (фотонний шум). Оскільки в даний час в якості перетворювачів зображень в сигнал застосовуються прилади із зарядним зв'язком (ПЗС), то в подальшому мова йтиме про них. Перший компонент шуму (шум матриці ПЗС), включає в себе шум, обумовлений темновим струмом, (він, як правило, невеликий), і шум зчитування.

Фотонний шум обумовлений квантовою природою світла. Падаючи на ПЗС фотони світла породжують в кремнієвому шарі фотоелектрони. Кількість фотоелектронів, що накопичуються за час експозиції, в кожному осередку ПЗС описується розподілом Пуассона. При високих рівнях освітленості число фотонів велике і закон Пуассона може бути апроксимований гауссовим законом. Середньоквадратичне значення шумової складової заряду, накопиченого в осередку ПЗС, $Q_{фш}$ і його середнє значення $Q_ф$, що представляє корисний сигнал, взаємопов'язані

$$Q_{фш} \approx \sqrt{Q_ф}. \qquad (2.31)$$

Темновий шум виникає під впливом тепла, внаслідок генерації електронів в кристалічній решітці кремнію ПЗС, оскільки процес генерації термоелектронів є випадковим процесом. Величина темнового струму не залежить від величини світлового потоку, який проектується на ПЗС, а отже, не залежить від величини світлового потоку і темнового шуму. Кількість

термоелектронів, що накопичуються в кожному осередку ПЗС за час експозиції, так само, як і кількість фотоелектронів, описується розподілом Пуассона. Тому середньоквадратичне значення темнового шуму $Q_{тш}$ підпорядковується співвідношенню

$$Q_{тш} \approx \sqrt{Q_т}, \tag{2.32}$$

де $Q_т$ — середнє значення накопиченого за час експозиції заряду, обумовленого термоелектронами.

В даний час типові значення темнового струму при кімнатній температурі для кращих ПЗС-матриць складають соті частки нА/см^2. Для побутового телебачення такий темновий струм практично непомітний, а значить, малий і темновий шум. У тих же випадках, коли величину темнового шуму необхідно зменшити, наприклад, в астрономічних системах, вдаються до охолодження кристала матриці ПЗС або шляхом застосування термоелектронного охолодження з використанням батареї Пельтьє, або за допомогою застосування азотних кріостатів.

Шум зчитування в основному обумовлений підсилювачем на чіпі. Середньоквадратичне значення шуму зчитування визначається параметрами матриці ПЗС і не залежить від величини зчитуваного сигнал, а отже, не залежить і від світлового потоку. Шум зчитування розподілений по гауссовому закону. Ця складова шуму, має два компоненти: компонент, що не залежить від частоти, і компонент, спектральна інтенсивність якого зменшується обернено пропорційно частоті. Другий компонент в значній мірі пригнічується схемою подвійної корельованої вибірки, яка служить фільтром верхніх частот.

Відношення "сигнал - шум" Ψ (ця величина для кожного колірного каналу розраховується окремо) є одним з найважливіших параметрів, що визначають якість зображення. Відношення "сигнал - шум" визначається як відношення розмаху сигналу від мінімуму до максимуму до результуючого середньоквадратичного значення шуму. Вираз для відносини "сигнал - шум" розраховується за формулою

$$\psi = \frac{I*K_{КвЕ}*t}{\sqrt{I*K_{КвЕ}*t+N_T+N_{сч}^2}}, \quad (2.33)$$

де I - кількість фотонів, що потрапляють в осередок матриці ПЗС за одну секунду, $K_{КвЕ}$ - квантова ефективність (величина, близька до одиниці), t - час накопичення заряду, N_T - кількість термоелектронів, що генеруються в осередку матриці за час накопичення заряду, $N_{сч}$ - середньоквадратичне значення шуму зчитування.

З наведеного виразу видно, що при низьких рівнях освітленості, при малих значеннях I, відношення сигналу до шуму визначається головним чином шумом зчитування. Це випливає з того, що величина заряду, $IK_{КвЕ}t$, накопиченого в осередку ПЗС при низьких освітленості, мала, а значить, невеликим буде і значення середнього квадрата фотонного шуму. При збільшенні освітленості збільшується і відношення сигналу. При цьому величина середньоквадратичного значення фотонного шуму зростає пропорційно квадратному кореню від освітленості і при високих рівнях освітленості відношення сигналу до шуму визначається в основному фотонним шумом.

2.2. Розробка архітектури інформаційної системи

2.2.1. Основні поняття і визначення

Основні поняття в останні роки не зазнали сильних змін, формулювання стали точнішими і лаконічними, що виключають неоднозначність понять.

Інформація - дані незалежно від виду і форми їх подання [25].

Інформаційні системи - сукупність що міститься в базах даних і забезпечує обробку інформаційних технологій і технічних засобів [25].

Проектування інформаційних систем - це впорядкована сукупність методологій і засобів створення або модернізації інформаційних систем.

Життєвий цикл інформаційних системи - «розвиток даної системи в часі, починаючи від задуму і кінчаючи списанням »[17].

Модель життєвого циклу - «структурна основа процесів і дій, що відносяться до життєвого циклу, яка служить в якості загальної посилання для встановлення зав’язків і взаєморозуміння сторін »[17].

Архітектура інформаційних систем - це концепція, що визначає модель, структуру, виконувані функції і взаємозв'язок компонентів інформаційної системи.

Бізнес-процес - це ланцюжок взаємопов'язаних дій, спрямованих на створення товарної продукції або послуги.

Регламент бізнес-процесу - це чітко певний порядок виконання бізнес-процесу, визначає склад і дії учасників.

Модель даних - це система організації даних і управління ними.

Методологія проектування інформаційних систем - це сукупність принципів проектування (моделювання), виражена в певній концепції.

Засоби моделювання - це програми опису та моделювання систем.

Типове проектне рішення (ТПР) – це багаторазово використання проектного рішення.

Нотації - це певні способи представлення елементів інформаційної системи.

Реінжиніринг бізнес-процесів – це фундаментальна реорганізація бізнес-процесів з метою підвищення їх ефективності.

Системний підхід - процес розгляду будь-якої системи в якості сукупності взаємопов'язаних елементів.

Процесний підхід - подання будь-якої системи в якості сукупності процесів.

Функціональний підхід – передбачає чітке закріплення за кожною структурною одиницею набору функцій.

Технічне завдання - документ, який використовується замовником як засіб для опису і визначення завдань, які виконуються при реалізації договору [25].

2.2.2. Методології сучасного проектування інформаційних систем

Існують три основні методології:

1. Методологія функціонального моделювання робіт SADT
2. Методологія RAD – швидкої розробки додатків
3. Методологія RUP

Розберемо кожну з них більш детально. Методологія SADT (Structured Analysis and Design Technique - методологія структурного аналізу і проектування), розроблена Дугласом Т. Россом в 1969-1973 роках базується на структурному аналізі систем і графічному уявлення організації у вигляді системи функцій, які мають три класи структурних моделей:

- Функціональна модель;
- Інформаційна модель;
- Динамічна модель.

Процес моделювання за методологією SADT складається з наступних етапів:

- Збір інформації та аналіз інформації про предметну область;
- Документування отриманої інформації;
- Моделювання (IDEF0) ;
- Коректура моделі в процесі ітеративного рецензування.

Методологія в даний час більш відома як нотація IDEF0, використовує формалізований процес моделювання інформаційних систем і має наступні стадії:

- аналіз;
- проектування;
- реалізація;
- об'єднання;
- тестування;
- установка;
- функціонування.

Проектування інформаційних систем по стандарту IDEF0 зводиться до декомпозиції основних функцій організації на окремі бізнес-процеси, роботи або дії. В результаті розробляється ієрархічна модель аналізованої організації, при цьому декомпозицію можна проводити багаторазово, до чіткого і детального опису всіх процесів. Діаграми IDEF0 верхнього рівня прийнято називати батьківськими, а нижнього рівня - дочірніми. Приклад діаграми IDEF0 верхнього рівня представлений на рис. 2.8.

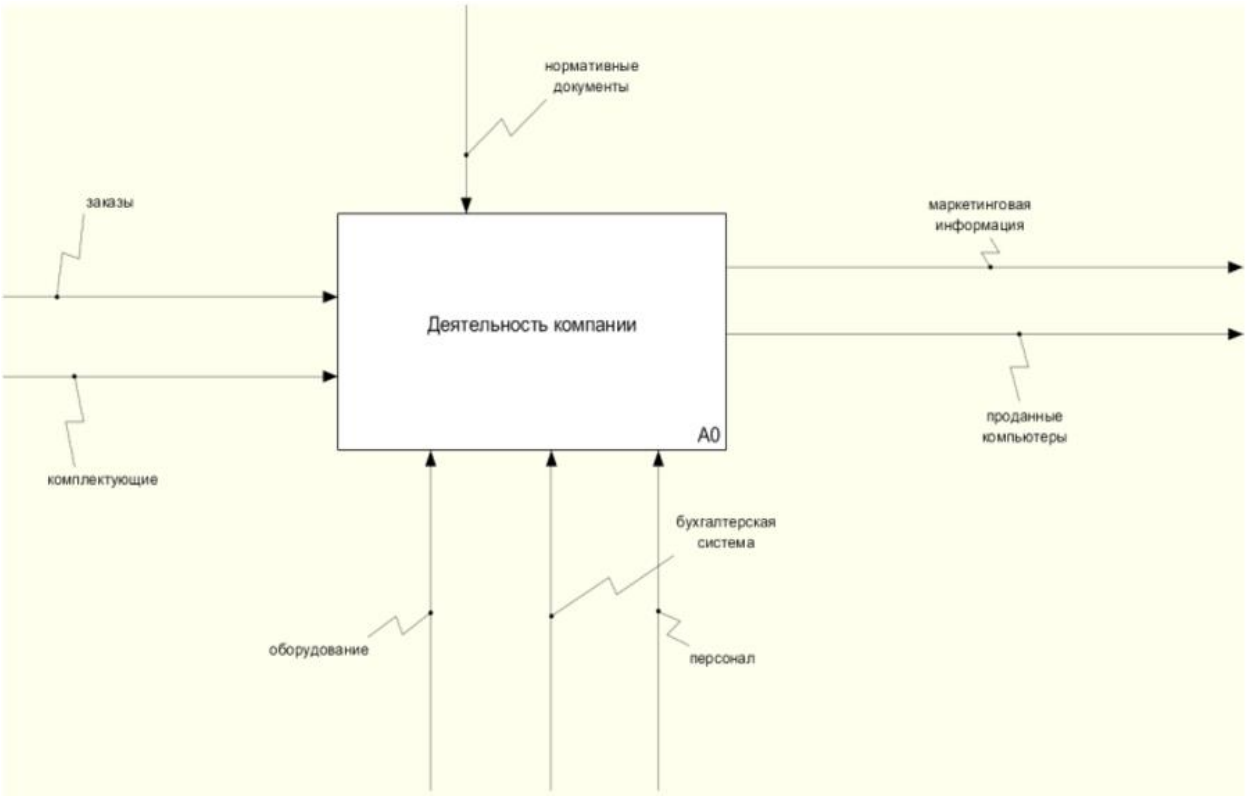


Рис. 2.8. Діаграма IDEF0 верхнього рівня.

Аналізований процес представляється у вигляді прямокутника. Зліва зображуються вхідні дані, праворуч - вихідні, зверху керуючі або регламентуючі впливу, а знизу об'єкти управління. В діаграмі IDEF0 описуються спочатку усі зовнішні зв'язки досліджуваного процесу. Після цього здійснюється декомпозиція цього процесу і відбувається опис внутрішніх підпроцесів з позначенням всіх зв'язків. При цьому раніше позначені стрілочками зовнішнього зв'язку не повинні загубитися. Вони переносяться на діаграму декомпозиції до відповідних підпрограм процесів.

Далі кожен підпроцес теж можна декомпозувати і докладно описувати всі зв'язки до необхідної межі. Основною перевагою цієї методології є простота і наочність. Як недолік - неможливість описати реакцію описуваного процесу на мінливі зовнішні фактори. Для цих цілей служать інші методології.

Тепер розберемо методологію RUP. Серед фірм які виробляють CASE-засоби компанія IBM Rational Software Corp. одна з перших зрозуміла перспективність створення об'єктно-орієнтованих технологій для аналізу і подальшому проектуванні програмних систем. Дана компанія виступила першим хто запропонував уніфікувати мови візуального моделювання на консорціумі OMG, це призвело до появи першої у своєму сегменті версії мови UML. Вони першими розробили інструментальний об'єктно-орієнтований CASE-засіб, де вони реалізували мову UML, як базова нотація для візуального моделювання.

Сьогодні одна з найпопулярніших технологій - Rational Unified Process (RUP). У певному сенсі ця методологія стає міжнародним стандартом, який був розроблений компанією Rational Software, вона в даний час входить до складу компанії IBM. Авторами UML вважаються співробітники фірми Rational Software: Айвар Якобсон, Граді Буч, Джемс Рамбо. RUP повністю відповідає стандартам, визначальним проектні роботи в процесі життєвого циклу інформаційних систем. У методології RUP реалізуються такі підходи:

- Ітераційний підхід і інкрементнивний;
- Планування а також управління проектами на основі функціональних вимог до інформаційної системі;
- Побудова системи на базі архітектури інформаційних систем.

Розробка інформаційної системи виконується ітераціями. Це окремі проекти невеликі за обсягом і змістом, які включають власні етапи для аналізу вимог, реалізації, проектування, тестування та інтеграції. Закінчуються ітерації створенням працюючої інформаційної підсистеми.

Ітераційний цикл характеризується зворотним зв'язком і може адаптуватися до ядра, що розробляється. Створювана інформаційна система поступово росте і вдосконалюється.

У свою чергу принципи RAD сформульовані в 1980 році співробітником компанії IBM Джеймсом Мартіном. Вони базувалися на ідеях Скотта Шульца і Баррі Бойм при цьому методологія реалізовувалася в найкоротші терміни невеликою групою розробників з використанням інкрементного протипування. Це дозволяло на ранній стадії проектування ІС продемонструвати замовникові діючу інтерактивну модель системи-прототипу, уточнити проектні рішення, оцінити експлуатаційні характеристики. В даний час методологія RAD стала загальноприйнятою схемою для проектування і розробки інформаційних систем. Засоби розробки, засновані на RAD, дуже популярні за рахунок використання таких програмних середовищ розробки: Android Studio, Borland Delphi, Microsoft Visual Studio, IBM Lotus Domino Designer, Macromedia Flash, Borland C ++ Builder і ін.

2.2.3. Архітектурний підхід до проектування інформаційних систем

Процес створення інформаційної системи тісно пов'язаний з архітектурним описом, який повинен відображатись в деяких визначеннях терміна «архітектура». Виділяють п'ять різних підходів до проектування:

- Архітектурний;
- За допомогою розробки документації;
- Календарний;
- Система управління якістю;
- Управління вимогами.

Календарний підхід включає створення графіка майбутніх робіт з їх поетапним виконанням. Необхідно зазначити, що ключові рішення приймаються на підставі локальних задач і цілей кожного конкретного етапу розробки. Також при цьому підході практично не приділяється час на розробку

документації, формуванню архітектури і процесів щодо внесення різних змін. У довгостроковій перспективі через це зростає вартість розробленої таким чином системи. Такий стиль вважається морально застарілим, проте в деяких компаніях до сих пір застосовується.

Підхід, за основу якого взято процес управління вимогами, більшу частину часу всього процесу розробки виділяє на функціональні характеристики системи, а не на такі як масштабованість, наприклад. Всі рішення в ході проекту формуються виходячи з локальних цілей по реалізації конкретного функціоналу. Такий підхід може бути ефективний, якщо вимоги до розроблюваної системи визначені заздалегідь і не змінюються в процесі проектування. Як недоліки можна виділити невідповідність стандарту якості ISO 9126 і нестабільність розроблюваних архітектура, оскільки кожна реалізована функція пов’язана з одним або декількома компонентами. В зв'язку з цим, трудомісткість при додаванні до існуючої системи додаткових функцій зростає.

Застосування цього підходу в довгостроковій перспективі платника є нераціональним, однак дозволить вдало управляти вимогами в рамках необхідної функціональності.

При застосуванні підходу, заснованого на процесі розробки документації, невиправдано велика кількість часу витрачається на формування пакета документів, які часто вже не використовуються ні замовником, ні користувачем. Крім того, через брак часу страждає якість самої системи, що розробляється. Даний підхід використовується в урядових організаціях і великих компаніях.

Процес проектування, в основі якого лежить система управління якістю, включає в себе велику кількість різнопланових заходів для відстеження найбільш значущих для функціонування системи параметрів. Обрані параметри спостерігаються на всіх стадіях розробки системи, причому, в деяких випадках, на шкоду іншим. В деяких випадках набір такого роду параметрів може бути розроблений неправильно і оптимізація системи буде

проведена помилково. У цьому полягає основний недолік подібного підходу. Крім цього, при появі нових вимог вельми важко змінювати функціональність, а значить подальший розвиток системи, в тому числі із-за архітектурних недоліків, може бути неможливо. Такий підхід вважається консервативним, а його застосування доцільно при необхідності створити систему з екстремальними характеристику.

З усіх розглянутих стилів проектування неможливо однозначно визначити кращий, оскільки кожна конкретна проектна група, кожна проектована система має свої особливості, на підставі яких слід вибрати той підхід, застосування якого дозволить вирішити поставлені задачі в установлені строки та в рамках виділеного бюджету. Але у даній роботі був обраний архітектурний підхід.

2.2.4. Шаблони проектування

При розподілі відповідальності та розробці способів взаємодії об'єктів розробнику надається достатня свобода дій. Невдалий вибір методу розподілу може привести до того, що системи та їх окремі компоненти виявляться непридатними для підтримки, розуміння, повторного використання та резервування. Невдачі можна уникнути, якщо при розподілі відповідальності застосовувати принципи об'єктно-орієнтованого проектування. Деякі з цих принципів систематизовані в шаблонах GRASP і застосовуються при розробці діаграм взаємодії, тобто при розподілі відповідальності між об'єктами і розробці способів їх взаємодії.

В об'єктно-орієнтованому проектуванні необхідно бути готовим описати проблему з розподілу відповідальності і її рішення. Результат вирішення проблеми це операції призначені об'єкту для виконання успішного (а в загальному випадку запланованого) сценарію взаємодії. В ідеалі шаблон повинен містити поради з приводу його застосування в різних ситуаціях. З цієї точки зору шаблони є основним механізмом для накопичення і повторного використання корисних принципів розробки програмного забезпечення.

Шаблон проектування Model-View-Controller (далі просто MVC) використаний в основі архітектурного рішення першого середовища програмування з графічним інтерфейсом користувача - Smalltalk-80. Згідно шаблону при проектуванні слід розділяти дані додатка, інтерфейс користувача і керуючу логіку на 3 різні компоненти: модель, представлення і контролер - таким чином, щоб кожний компонент мав мінімальний вплив на інші. Модель (Model) зберігає дані предметної області, вона також реагує на команди від контролера, зберігаючи власний свій стан. Представлення (View) відповідає за графічне відображення даних користувача, як реакцію на зміни моделі. Контролер (Controller) адаптує дії користувача на графічному інтерфейсі, сповіщаючи модель про зміни.

Можна привести приклад MVC, де модель може бути представлена об'єктом, який реалізує умовний перемикач. Дана модель може характеризуватись своїм станом коли вона вимкнена або включена. Крім того, у моделі ми можемо змінювати свій стан, тобто об'єкт моделі має реалізовані методи зміни стану: вимкнути і включити. Представлення відображується на екрані користувача за допомогою спеціальної текстової або графічної форми. Наприклад, представлення може малювати текстову мітку, при зміні стану моделі перемикача відобразить відповідний текст: «вимкнений» або «включений». Крім відображення представлення дозволяє користувачеві змінювати стан уявного перемикача за допомогою графічного інтерфейсу, наприклад, кнопок з написами: «Включити» і «Вимкнути». Представлення вміє тільки відображати стан моделі перемикача, для зміни представлення звертається до контролера. Контролер - це об'єкт, який в нашому випадку вміє тільки змінювати стан моделі перемикача.

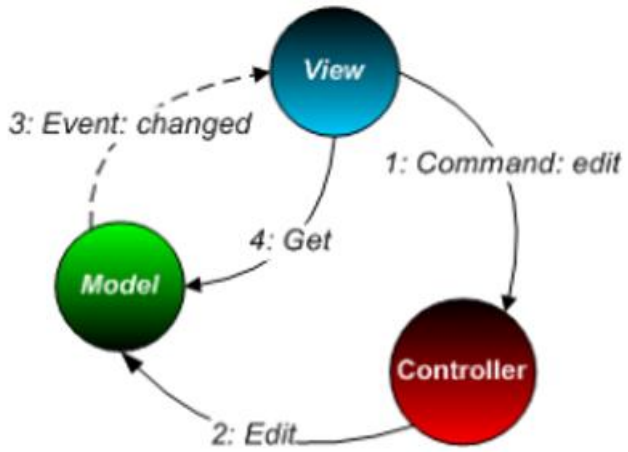


Рис. 2.9. Взаємодія між компонентами у архітектурі MVC.

Повний цикл роботи даної MVC-тріади: моделі, уявлення і контролера перемикача - можна описати таким чином. При ініціалізації представлення користувачем, воно звертається до моделі і встановлює текст мітки відповідно до поточного стану перемикача. Користувач ініціює зміну перемикача, натискаючи на певну кнопку. При цьому представлення відправляє відповідну команду контролеру: включити або вимкнути. Контролер інтерпретує команду і змінює модель. Подання реєструє зміну моделі: по цій події воно змінює текст мітки для відповідності новим станом моделі перемикача.

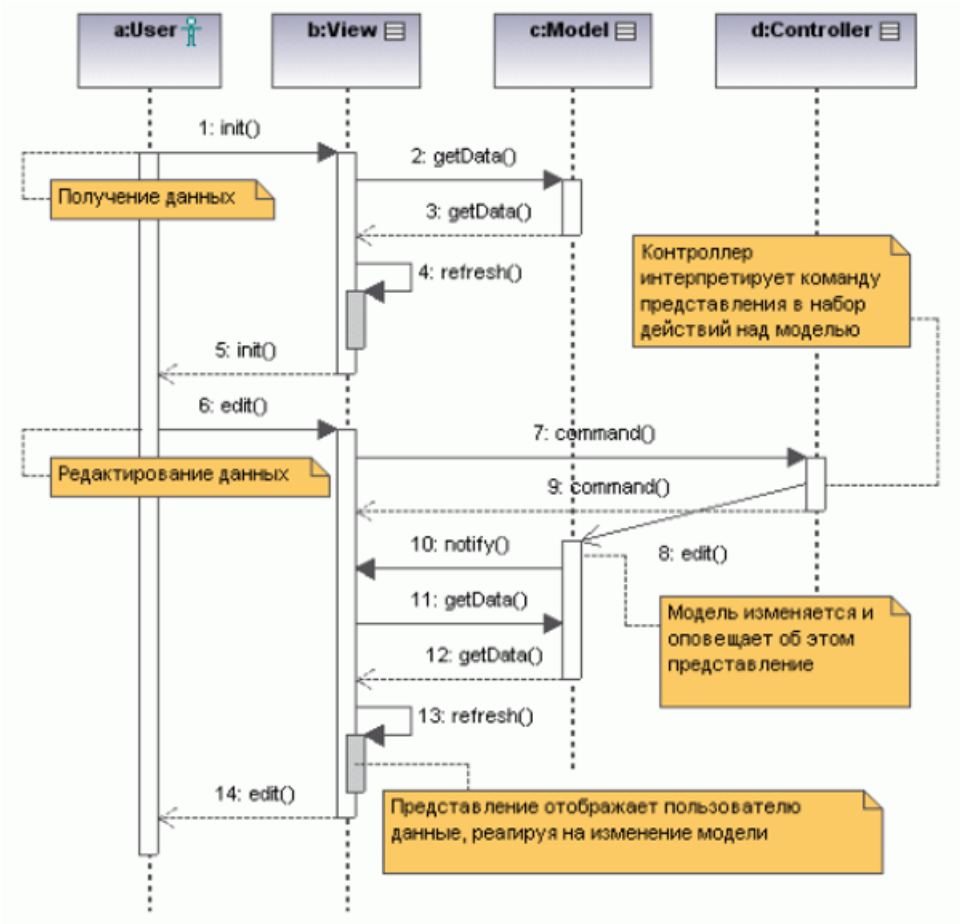


Рис. 2.10. Життєвий цикл компонентів архітектури MVC.

Рішення Smalltalk-80 було запропоноване майже 30 років тому, але навіть сьогодні використовують при створенні інтернет-додатків і не тільки.

Далі розглянемо шаблон MVVM (Model-View-ViewModel). Процес розробки зазвичай має на увазі створення призначеного для користувача інтерфейсу і подальшого додавання коду, що працює в інтерфейсі. Коли додатки вдосконалюються і збільшуються в розмірах, можуть виникнути проблеми з обслуговуванням. Ці проблеми з'являються коли ми задаємо тісний зв'язок між елементами управління користувальницького інтерфейсу і бізнес-логіку, що збільшує витрати на зміни призначеного для користувача інтерфейсу і складності при модульному тестування такого коду.

Шаблон Model-View-ViewModel (MVVM) допомагає чітко відокремити бізнесу логіку і логіку додатка від його призначеного для користувача

інтерфейсу (UI). Підтримка чіткого поділу між логікою програми та інтерфейсом допомагає вирішити безліч проблем розробки і дозволяє простіше тестувати, обслуговувати і розвивати додаток. Також можна значно покращити можливості повторного використання коду, що дозволяє розробникам і конструкторам інтерфейсу програми тісніше співпрацювати при розробці їх відповідних частин програми.

У шаблоні MVVM є три основних компоненти: модель, представлення і модель представлення. Кожен компонент призначений для різних цілей на рис. 2.11. показані зв'язки між трьома компонентами.

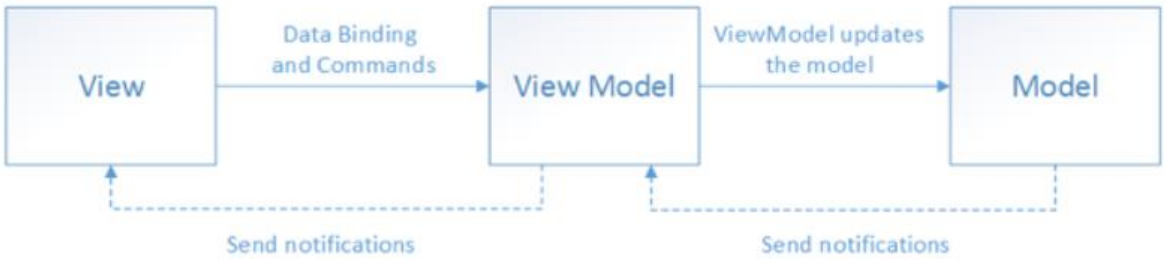


Рис. 2.11. Графічне представлення взаємодії компонентів у архітектурі MVVM.

Крім розуміння обов'язків кожного компонента, також важливо розуміти, як вони взаємодіють один з одним. На високому рівні, представлення «знає» про моделі представлення, модель представлення «знає» про модель, але модель не знає про модель представлення і не знає про представлення. Таким чином модель представлення ізолює представлення з моделі і дозволяє розвиватися незалежно від представлення моделі.

Нижче наведені переваги використання шаблону MVVM.

- Якщо є реалізація моделі, що інкапсулює існуючу бізнес-логіку, буває складно або небезпечних її міняти. У цьому випадку модель представлення виконує функції адаптера для класів моделі і дозволяє уникнути внесення основних змін коду моделі.
- Можна створювати модульні тести для моделі представлення та моделі, без використання представлення.

- Призначений для користувача інтерфейс може бути перероблений без зміни коду моделі представлення. Таким чином нова версія буде працювати з існуючою моделлю представлення.
- Дизайнери і розробники можуть працювати незалежно один від одного і паралельно в своїх компонентах під час розробки. Дизайнери зможуть сконцентруватися на представленні, а розробники можуть працювати на моделі представлення та моделі компонентів.

Ключем до ефективного використання MVVM є розуміння того, як правильно розділяти код програми на класи і зрозуміти, як взаємодіють створені класи.

2.3. Обробка зображень за допомогою бібліотеки OpenCV

OpenCV є, напевно, найбільш широко поширеною бібліотекою для обробки зображень. Вона включає сотні готових функцій обробки зображень і використовується як в академічних установах, так і в промисловості. Попит на обробку зображень зростає, тому спектр додатків OpenCV як на настільних так і на мобільних платформах постійно розширюється.

2.3.1. Загальні відомості

Головним типом даних в OpenCV є клас Mat, який служить для зберігання зображень. Зображення зберігається у вигляді заголовка та області для піксельних даних. Зображення складається з декількох каналів. У напівтонових зображень один канал, а у кольорових зазвичай три: для червоної, зеленої і синьої складової. Можна використовувати також четвертий канал (альфа), що описує прозорість. Кількість каналів зображення img повертає функція img.channels ().

Для зберігання одного пікселя зображення використовується певне число бітів, яке називається глибиною зображення. Для напівтонових зображень піксель зазвичай представляється 8 бітами, так що все виходить 256

рівнів яскравості (цілі значення від 0 до 255). Пікселі кольорового зображення представляються трьома байтами, по одному на кожний колірний канал. Для деяких операцій пікселі необхідно зберігати в форматі з плаваючою точкою. Глибину зображення можна отримати за допомогою функції img.depth (), яка може повертати наступні значення:

• CV_8U, 8-розрядне ціле додатне (0..255)

• CV_8S, 8-розрядне ціле (-128..127)

• CV_16U, 16-розрядне ціле додатне (0..65 535)

• CV_16S, 16-розрядне ціле (-32 768..32 767)

• CV_32S, 32-розрядне ціле (-2 147 483 648..2 147 483 647)

• CV_32F, 32-розрядне з плаваючою точкою

• CV_64F, 64-розрядне з плаваючою точкою

Для напівтонових, і для кольорових зображень найчастіше використовується CV_8U. Перетворити глибину дозволяє метод convertTo:

Mat img = imread("lena.png", IMREAD_GRAYSCALE);

Mat fp;

img.convertTo(fp,CV_32F);

Нерідко виконуються операції над зображеннями, представленими числами з плаваючою точкою (коли значення пікселів є результатами математичних операцій). Якщо для показу такого зображення скористатися функцією imshow(), то вийде безглуздий результат. Щоб уникнути цього необхідно перетворити значення пікселів до цілочисленного діапазону 0..255. Функція convertTo виконує лінійне перетворення і приймається два додаткові параметри, alpha і beta, що представляють масштабний коефіцієнт і вільний член. Таким чином, піксель p перетворюється за формулою:

$$newp = alpha * p + beta \tag{2.34}$$

Це може бути корисним для правильного показу зображень у форматі з плаваючою точкою. У припущенні, що мінімальне значення пікселя зображення m, а максимальне - M, потрібно написати такий код:

Mat m = Mat(200, 200, CV_32FC1);

randu(m, 0, 1e6);

imshow(m);

double minR,maxR;

Point mL,ML;

minMaxLoc(1,&minRe,&MaxR,&mL,&ML);

Mat img;

m.convertTo(img,CV_8U,255.0/(MaxR-minR),-255.0/minR);

imshow(img);

Цей код перетворює діапазон значень пікселів результуючого зображення до діапазону 0-255. На рис. 2.12. показаний результат його виконання.

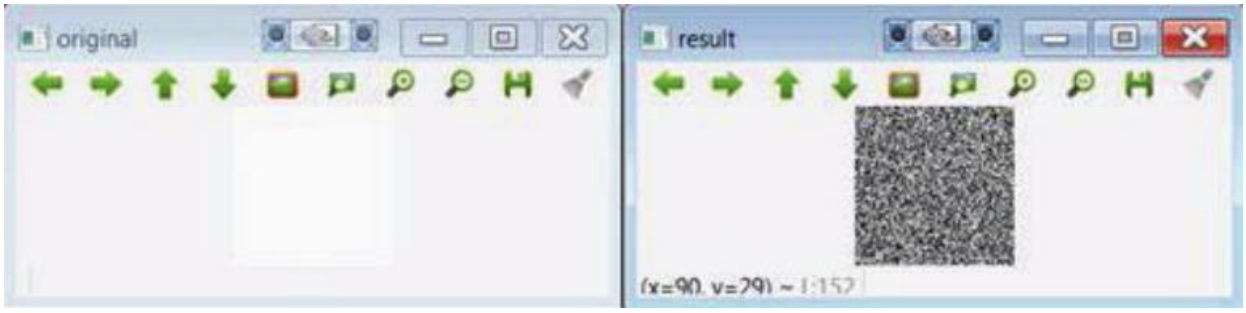


Рис. 2.12. Результат роботи методу convertTo.

Розмір зображення можна дізнатися, зчитавши атрибути rows і cols. Існує також атрибут size, що містить відразу обидва розміру:

MatSize s = img.size;

int r=s[0];

int c=s[1];

Для деяких типів є додаткові операції. Наприклад, можна перевірити, чи лежить точка всередині прямокутника:

Для обробки зображень потрібно вміти звертатися до окремих пікселів. У OpenCV для цього є багато механізмів. Розглянемо тільки два: перший зручніший, другий - ефективніший. У першому способі застосовується шаблонна функція at <>. Для роботи з нею ми повинні задати тип елементів матриці, як в прикладі нижче:

```
Mat src = imread("*.jpg", IMREAD_GRAYSCALE);
uchar pixel=src.at<uchar>(0,0);
Mat src1 = imread("*.jpg", IMREAD_COLOR);
Vec3b pixel1 = src1.at<Vec3b>(0,0);
```

Тут ми читаємо одне і те ж зображення спочатку як півтонове, а потім як кольорове і звертаємося до першого пікселя в позиції (0,0). У першому випадку піксель буде мати тип unsigned char, а в другому потрібно використовувати тип Vec3b. Зрозуміло, функція at<> може зустрічатися і в лівій частині оператора присвоювання, якщо потрібно змінити значення пікселя. У наступному фрагменті матриця чисел з плаваючою точкою за допомогою цього способу зініціюється значенням Pi:

```
Mat mat(300, 300, CV_64F);
for(int i = 0; i < mat.rows; i++)
   for(int j = 0; j < mat.cols; j++)
      mat.at<double>(i,j)=CV_PI;
```

Але метод at<> не дуже ефективний, тому що повинен обчислювати адреса в пам'яті по рядку і стовпцю пікселя. Якщо попіксельно потрібно обробити все зображення, то на це може піти багато часу. У другому способі використовується функція ptr, яка повертає покажчик на рядок зображення. Функція ptr повертає покажчик на перший піксель чергового рядка. Знаючи цей покажчик, можемо у внутрішньому циклі обійти пікселі в кожному стовпці.

Арифметичні оператори перевантажені. Це означає, що над об'єктами типу Mat можна виконувати такі дії:

$$\text{imgblend} = 0.2*\text{img1} + 0.8*\text{img2}; \qquad (2.35)$$

У OpenCV значення результату операції підпорядковується правилам так званої арифметики з насиченням (saturation arithmetic). Це означає, що значенням є найближче ціле в діапазоні 0-255. При роботі з масками дуже корисні порозрядні операції bit-wise_and(), bitwise_or(), bitwise_xor() і bitwise_not(). Маскою називається бінарне зображення, яке показує, над якими

пікселями робити операцію. Припустимо, що є зображення квадрата з невідомої довжиною сторони і вписаного в нього кола. Ми можемо оцінити площу кола, намалювавши багато пікселів в випадкових позиціях і підрахувавши, скільки з них потрапили всередину кола. З іншого боку, площа квадрата оцінюється як загальне число намальованих пікселів. Це дозволить оцінити значення Pi за формулою вище.

Крім конкретних функцій для читання і запису зображень, в OpenCV існує більш загальний спосіб збереження і завантаження даних. Він так і називається «збереження даних» (data persistence): значення об'єктів і змінних програми можна записати на диск. Це дуже корисно для збереження результатів і завантаження конфігураційних даних. Основний клас називається FileStorage, він представляє файл на диску. Нижче описані кроки записи даних.

1. Викликати конструктор класу FileStorage, передавши йому ім'я файлу і прапор FileStorage :: WRITE. Формат даних визначається розширенням імені файлу.

2. Записати дані в файл за допомогою оператора <<. Зазвичай дані записуються у вигляді пар рядок-значення.

3. Закрити файл, викликавши метод release.

Читання даних проводиться в наступному порядку.

1. Викликати конструктор класу FileStorage, передавши йому ім'я файлу і прапор FileStorage :: READ.

2. Прочитати дані з файлу за допомогою оператора [] або >>.

3. Закрити файл, викликавши метод release.

Якщо смуга прокрутки використовується для управління цілочисельним значенням - яскравістю, то вона просто поміщається на вихідне зображення. Поточне значення зчитається на початку програми (при найпершому запуску воно буде дорівнює 0) і зберігається перед виходом.

2.3.2. Фільтрація зображень

Фільтрацією називається процес модифікації або поліпшення зображення. Підкреслення одних частин і видалення інших. Фільтрація застосовується до околиць пікселів. Околицею називається безліч пікселів, що оточують даний. В результаті фільтрації визначається нове значення пікселя в позиції (x, y) шляхом застосування тих чи інших операцій до значень пікселів в його околиці. У OpenCV є кілька функцій фільтрації, призначених для таких операцій, як згладжування або підвищення різкості.

Згладжування, або розмиття - операція, яка часто застосовується для зменшення шуму, та інших цілей. Виконується вона шляхом застосування лінійних фільтрів до зображення. Це означає, що нове значення пікселя (x_i, y_j) обчислюється як зважена сума значень вихідних пікселів в тій же позиції і її околиці.

Ваги пікселів зазвичай зберігаються в матриці, яку називають ядром. Таким чином, фільтр можна уявляти собі як ковзне вікно коефіцієнтів.

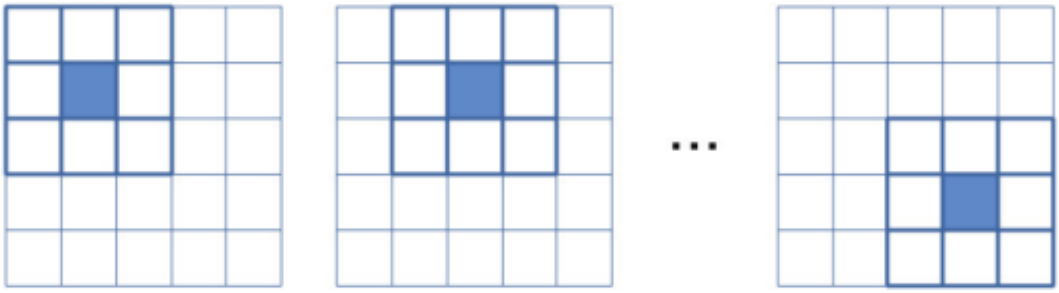


Рис. 2.13. Представление окрестности пикселя.

Позначимо K - ядро, а I і O - вхідний і вихідний зображення. Тоді нове значення пікселя в позиції (i, j) обчислюється за формулою:

$$O(i,j) = \sum_{m,n} I(i+m, j+n) * K(m,n) \qquad (2.36)$$

У OpenCV для згладжування найчастіше застосовується напівказуальний фільтр, медіанний фільтр, фільтр Гаусса і двосторонній фільтр. Медіанний фільтр дуже хороший для усунення зернистості

зображення, а фільтр Гаусса - в якості попереднього кроку для виявлення меж. Що стосується двосторонньої фільтрації, то вона застосовується, в основному, для згладжування зображень зі збереженням різких кордонів. Для цих цілей в OpenCV включені наступні функції:

• void boxFilters (InputArrays src, OutputArrays dst, int ddepth, Size ksize, Point anchors = Point (-1, -1), bool normalie = true, int borberType = BORDER_DEFAULT): це планшетний фільтр тобто усі коефіцієнти ядра рівні. Якщо normalize = true, то кожен вихідний піксель обчислюється як середнє арифметичне сусідів з коефіцієнтами ядра 1 / n, де n - число сусідів. Якщо ж normalize = false, то всі коефіцієнти рівні 1. В аргументі src передається вхідне зображення, а в dst зберігається відфільтроване зображення. Параметр ddepth задає глибину вихідного зображення, причому значення -1 означає, що потрібно використовувати таку ж глибину, як у вхідного. Розмір ядра задається параметром ksize. Параметр anchor визначає положення так званого якірного пікселя. За замовчуванням значення (-1, -1) означає, що якір знаходиться в центрі ядра. Нарешті, параметр borderType визначає поведінку на краях зображення.

• void GausianBlur(InputArrays src, OutputArrays dst, Size ksize, double sigmX, double sigmY = 0, int borberType = BORDER_DEFAULT): цей фільтр згортає кожну точку вхідного масиву src з гаусовим ядром. Параметри sigmaX і sigmaY задають стандартні відхилення в напрямку осей X і Y. Якщо sigmaY дорівнює 0, він приймається рівним sigmaX, а якщо нулю рівні обидва параметри, то вони обчислюються по ширині і висоті, заданих в ksize

• void medianBlure(InputArrays src, OutputArrays dst, int ksize): цей фільтр обробляє кожен піксель зображення, замінюючи його медіаною сусідніх пікселів.

• void bilatralFilter(InputArrays src, OutputArrays dst, int d, double sigmaColors, double sigmSpace, int borberType = BORDER_DEFAULT): напівказуальний фільтр, що приписує ваги кожного сусіднього пікселя, але вага складається з двох компонент, одна з яких збігається з використовуваної

в фільтрі Гаусса, а інша бере до уваги різницю інтенсивностей оброблюваного пікселя і його сусідів. Цій функції необхідно передати діаметр d околиці пікселя, а також значення sigmaColor і sigmaSpace. Чим більше величина sigmaColor, тим більш віддаленість кольору в околиці пікселя будуть змішуватися, тобто будуть породжуватися більш великі області близького кольору. А чим більше величина sigmaSpace, тим більш далекі пікселі будуть впливати один на одного за умови, що їх кольору досить близькі. Дана функція була модифікована, як показано у статі [1]. Це дозволяє підвищити якість зображення, при менших обчислюваних затратах.

• void blur (InputArrays src, OutputArrays dst, Size ksize, Point anchors = Point (-1, -1), int borberType = BORDER_DEFAULT): розмиває зображення нормованим скриньовим фільтром. Еквівалентна функції boxFilter з параметром normalize = true. Використовується таке ядро:

$$\frac{1}{ksize.width*ksize.height}\begin{bmatrix}1 & \cdots & 1\\ \vdots & \ddots & \vdots\\ 1 & \cdots & 1\end{bmatrix} \tag{2.37}$$

У будь-якому випадку необхідно екстраполювати значення неіснуючих пікселів за кордоном зображення. Для більшості фільтрів OpenCV дозволяє вказати спосіб екстраполяції, а саме:

• BORDER_REPLICATE: повторювати останнє відоме значення пікселя;

• BORDER_REFLECT: відображати щодо кордону зображення;

• BORDER_REFLECT_101: відображати щодо кордону зображення, не дублюючи останній піксель, що примикає до кордону;

• BORDER_WRAP: додавати значення, що примикають до протилежної кордоні;

• BORDER_CONSTANT: вважати, що всі пікселі за кордоном однакові.

На рис. 2.14. показано, як до зображення застосоване гауссово і медіанне розмиття, тобто функції GaussianBlur і medianBlur.

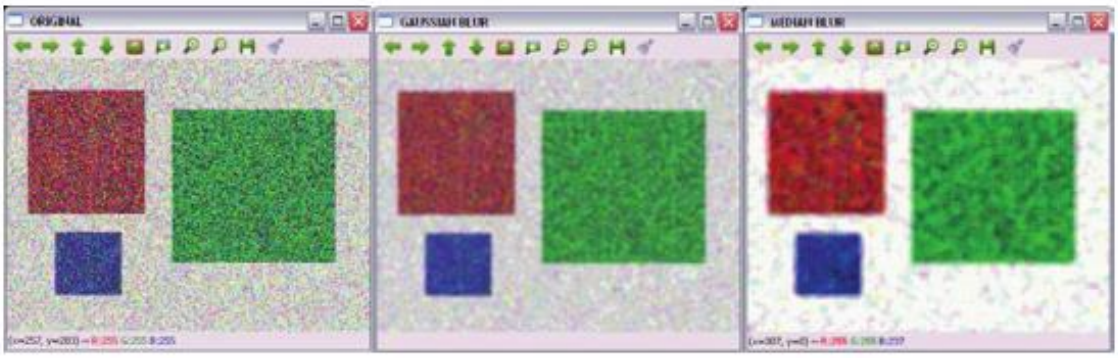


Рис. 2.14. Початкове і розмите зображення після застосування гауссова і медіанного фільтра.

Фільтри, що підвищують різкість використовуються, щоб виділити кордони та інші деталі. Вони засновані на обчисленні першої і другої похідної. Перша похідна описує градієнт інтенсивності зображення, а друга визначається як дивергенція градієнта. Оскільки при цифровій обробці зображень ми маємо справу з дискретними величинами (значеннями пікселів), то для підвищення різкості обчислюються дискретні похідні.

Перші похідні дають більш жирні кордони і широко застосовуються для виділення меж. Другі ж похідні використовуються для поліпшення зображення, тому що краще реагують на дрібні деталі. Для обчислення похідних найчастіше використовуються оператори Собеля і Лапласа. Оператор Собеля обчислює першу похідну зображення I:

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} * I, \tag{2.38}$$

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} * I.$$

Модуль градієнта тоді можна обчислити за формулою:

$$G = \sqrt{G_x^2 + G_y^2}. \tag{2.39}$$

Дискретний оператор Лапласа зображення обчислюється як згортка з наступним ядром:

$$D_{xy}^2 = \begin{bmatrix} 0.5 & 1 & 0.5 \\ 1 & 6 & 1 \\ 0.5 & 1 & 0.5 \end{bmatrix}. \quad (2.40)$$

Для цих цілей в OpenCV є наступні функції:

• void Sobel (InputArrays src, OutputArrays dst, int depth, int dx, int dy, int ksize = 3, doudle scale = 1, doudle delta = 0, int borberType = BORDER_DEFAULT): обчислює першу, другу третю або змішану похідну зображення src застосовуючи оператор Собеля. Параметр depth задає глибину вихідного зображення, причому значення -1 означає, що потрібно використовувати таку ж глибину, як у вхідного. Розмір ядра задається параметром ksize, а порядок похідних - параметрами dx і dy. Параметр scale дозволяє задати коефіцієнт масштабування обчислених значень похідної. Параметр borderType визначає поведінку на краях зображення, а delta - значення, додаємо до результуючого пікселя перед збереженням в dst.

На рис. 2.15. показано, результат обчислення похідних зображення за допомогою операторів Собеля і Лапласа.

Рис. 2.15. Контури, отримані застосуванням операторів Собеля і Лапласа.

2.3.3. Робота з пірамідами зображень

Іноді працювати з зображенням фіксованого розміру недостатньо, потрібно мати варіанти оригінального зображення з різними розмірами. Такий набір зображень називається пірамідою (або mipmap), оскільки зображення, впорядковані від більшого до меншого. Існують піраміди двох видів: Гаусса і Лапласа.

Рис. 2.16. Піраміда зображень.

Піраміда Гаусса створюється шляхом видалення кожного другого рядка і стовпця зображення нижнього рівня, а для отримання значень пікселів на наступному рівні до зображення попереднього рівня застосовується фільтр Гаусса. На кожному кроці побудови піраміди ширина і висота зображення зменшуються вдвічі, а площа - в чотири рази. У OpenCV для обчислення піраміди Гаусса служать функції pyrDown, pyrUp і buildPyramid:

• void pyrDown(InputArrays src, OutputArrays dst, const Size & dtsize = Size(), int borberType = BORDER_DEFAULT): виробляє вибірку рядків і стовпців і розмиття зображення src, результат зберігається в dst. Розмір вихідного зображення обчислюється як Size ((src.cols + 1) / 2, (src.rows + 1) / 2), якщо він не заданий явно за допомогою параметра dstsize.

• void pyrUp(InputArrays src, OutputArrays dst, const Size & dstsize = Size(), int borberType = BORDER_DEFAULT): виробляє обчислення, зворотне pyrDown

• void buildPyramid(InputArrays src, OutputArraysOfArrays dst, int maxlevel, int borberType = BORDER_DEFAULT): будує піраміду Гаусса для зображення src, створюючи maxlevel нових зображень, які зберігаються в масиві dst слідом за вихідним зображенням, яке поміщається в елемент dst.

Таким чином, після завершення процедури в масиві dst з'являться maxlevel + 1 зображень. У OpenCV є функція для обчислення пірамід на першому кроці алгоритму сегментації із зсувом середнього.

• void pyrMeanShiftFilter(InputArrays src, OutputArrays dst, doudle sp, doudle sr, int maxLevel = 1, TermCriteria termcri = TermCriteria (TermCriteria :: MAX_ITER + TermCriterial :: EPS, 5, 1)): реалізує етап фільтрації в алгоритмі сегментації методом зсуву середнього, отримує зображення dst, що містить лінеаризовані градієнти і детальну текстуру. Параметри sp і sr задають радіуси просторового і колірного вікна відповідно.

Морфологічні операції застосовують до зображення, в результаті чого формується нове зображення, в якому піксель в позиції (x_i, y_j) виходить шляхом порівняння пікселя вихідного зображення в тій же позиції з його сусідами. Залежно від обраного структурного елементу морфологічна операція може бути більш чутливою у одних формах і менш чутливою до інших.

Рис. 2.17. Приклад зменшення і збільшення зображення.

Дві основні морфологічні операції - нарощування (dilation) і ерозія. У разі нарощування до кордонів об'єктів в зображенні додаються пікселі з фону, а в разі ерозії пікселі видаляються. Структурний елемент визначає, які саме пікселі додавати або видаляти. При нарощуванні значення результуючого пікселя обчислюється як максимум, а при ерозії - як мінімум всіх пікселів в його околиці.

Оригінал **Нарощування** **Ерозія**

Рис. 2.18. Приклад нарощування і ерозії.

Шляхом комбінування нарощування і ерозії можна визначати і інші операції обробки зображень, наприклад розмикання, замикання і морфологічний градієнт. Операція розмикання (відкриття) визначається як ерозія, за якою слідує нарощування, а операція замикання (закриття) навпаки - як нарощування з подальшою ерозією. Таким чином, розмикання видаляє дрібні об'єкти, залишаючи великі, а замикання видаляє невеликі дірки, залишаючи більші. Морфологічний градієнт визначається як різниця між нарощуванням і ерозією зображення. На основі розмикання і замикання визначаються ще дві операції: відбілювання (white top-hat) і зачерненого (black top-hat). Операція відбілювання визначається як різниця між вихідним зображенням і його відстань, а операція зачернення - як різниця між замиканням і вихідним зображенням. Всі операції виконуються з одним і тим же структурним елементом. У OpenCV нарощування, ерозія, розмикання і замикання реалізовані такі функції.

• void dilate(InputArray src, OutputArray dst, InputArray kernel, Point anchor = Point (-1, -1), int iterations = 1, int borderType = BORDER_CONSTANT, const Scalar & borderValue = morphologyDefaultBorderValue ()): нарощує зображення src за допомогою заданого структурного елементу і зберігає результат в dst. Параметр kernel визначає структурний елемент, а anchor - позицію якірного пікселя в ньому. Значення (-1, -1) означає, що якір знаходиться в центрі. Операцію можна застосовувати кілька разів, параметр iterations задає число повторень. Параметр borderType визначає поведінку на краях - так само, як в описаних раніше фільтрах; якщо він не заданий, мається на увазі значення BORDER_CONSTANT.

• void erode(InputArray src, OutputArray dst, InputArray kernel, Point anchor = Point (-1, -1), int iterations = 1, int borderType = BORDER_CONSTANT, const Scalar & borderValue = morphologyDefaultBorderValue ()): еродує зображення за допомогою заданого структурного елемента. Параметри такі ж, як у функції dilate.

• void morphologyEx(InputArray src, OutputArray dst, int op, InputArray kernel, Point anchor = Point (-1, -1), int iterations = 1, int borderType = BORDER_CONSTANT, const Scalar & borderValue = morphologyDefaultBorderValue ()): виконує похідні морфологічні операції, які визначаються параметром op, який може набувати таких значень: MORPH_OPEN, MORPH_CLOSE, MORPH_GRADIENT, MORPH_TOPHAT і MORPH_BLACKHAT.

• Mat getStructuringElement(int shape, Size ksize, Point anchor = Point (-1, -1)): повертає структурний елемент для морфологічних операцій зазначеного розміру і форми. До допустимих значень можна віднести: MORPH_RECT, MORPH_ELLIPSE і MORPH_CROSS

2.3.4. Геометричні перетворення

Геометричне перетворення не змінює зміст зображення, а деформує його за допомогою зміни системи координат.

Тобто для кожного вхідного пікселя ми беремо його координати, застосовуємо до них перетворення і копіюємо піксель, який опинився в точці з новими координатами, у вихідне зображення.

$$O(x, y) = I\left(f_x(x, y), f_y(x, y)\right). \tag{2.41}$$

При цьому може виникнути дві проблеми.

• Екстраполяція: значення $f_x(x, y)$ і $f_y(x, y)$ можуть виявитися поза межами зображення. До геометричних перетворень застосовуються ті ж режими екстраполяції, що в разі фільтрації, плюс один додатковий: BORDER_TRANSPARENT.

• Інтерполяція: $f_x(x, y)$ і $f_y(x, y)$ зазвичай приймають дійсні значення. У OpenCV є вибір між інтерполяцією по найближчому сусіду і поліноміальної інтерполяції. У першому випадку значення координати з плаваючою точкою округляється до найближчого цілого. Підтримуються наступні методи інтерполяції:

◦ INTER_NEAREST: це описана вище інтерполяція по найближчому сусіду;

◦ INTER_LINEAR: білінійна інтерполяція, встановлена за замовчуванням;

◦ INTER_AREA: передискретизація по області пікселів;

◦ INTER_CUBIC: бікубічна інтерполяція по околиці 4 × 4;

◦ INTER_LANCZOS4: метод Ланцоша по околиці 8 × 8.

OpenCV підтримує аффінниє перетворення. Афінним називається геометричне перетворення, при якому прямі переходять в прямі і додатково зберігається відношення довжин відрізків. Але кути і довжини можуть не зберігатися. До афінних перетворень відносяться, зокрема, масштабування, паралельне перенесення, поворот, скіс і дзеркальне відображення.

Масштабуванням зображення називається зміна його розміру (збільшення або зменшення). У OpenCV цієї мети служить функція void resize (InputArray src, OutputArray dst, Size dsize, double fx = 0, double fy = 0, int interpolation = INTER_LINEAR).

Крім вхідного (src) і вихідного (dst) зображення, вона приймає параметри, що визначають новий розмір. Якщо новий розмір, заданий параметром dsize, відмінний від 0, то коефіцієнти масштабування fx і fy повинні бути рівні 0 і обчислюються по dsize і розміром вихідного зображення. Якщо fx і fy відмінні від 0, а dsize дорівнює 0, то значення dsize обчислюється на основі інших параметрів. Операцію масштабування можна уявити такою матрицею перетворення:

$$S = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix}. \tag{2.42}$$

Тут s_x і s_y - коефіцієнти масштабування по осях x і y.

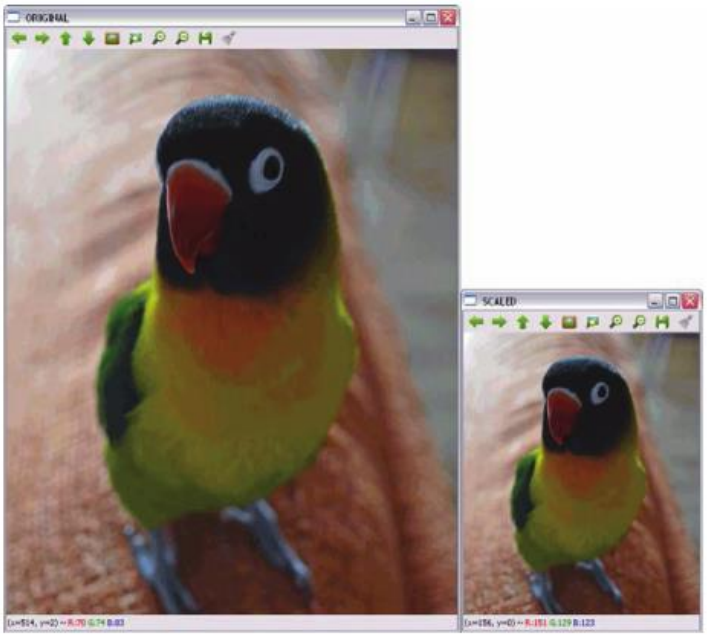
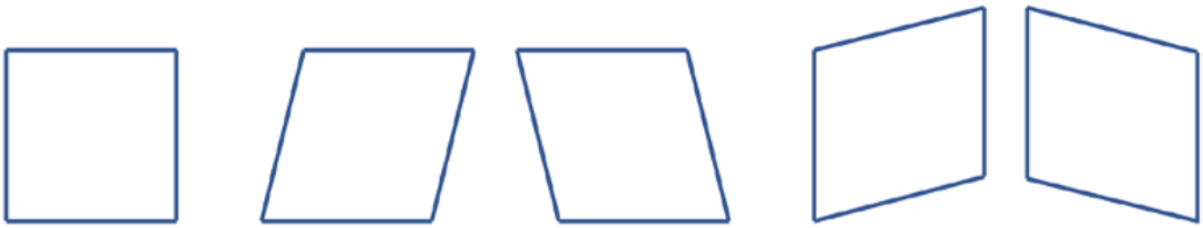


Рис. 2.19. Початкове і масштабоване зображення.

Перетворення скоса зрушує всі крапки в одному напрямку на величину, пропорційну взятому зі знаком віддалі від точки до прямої, паралельної цьому напрямку. Тому форма геометричної фігури зазвичай спотворюється: квадрати перетворюється в паралелограми, а окружності - в еліпси. Однак скіс зберігає площу фігури, паралельність прямих і відстані між колінеарними точками. Саме скіс визначає основну відмінність між прямим і курсивним накресленими літерами. Скіс визначається кутом θ.



Оригінал По горизонталі По вертикалі

Рис. 2.20. Приклади скосу.

Оригінал і воно ж, скошене на 45 градусів. Матриці скоса на кут θ по горизонталі і по вертикалі виглядають так:

$$T_H = \begin{bmatrix} 1 & \cos(\theta) & 0 \\ 0 & 1 & 0 \end{bmatrix}, T_V = \begin{bmatrix} 1 & 0 & 0 \\ \cos(\theta) & 1 & 0 \end{bmatrix}. \quad (2.43)$$

Оскільки вони схожі на матриці інших перетворень, то для реалізації скоса застосовується все та ж функція warpAffine

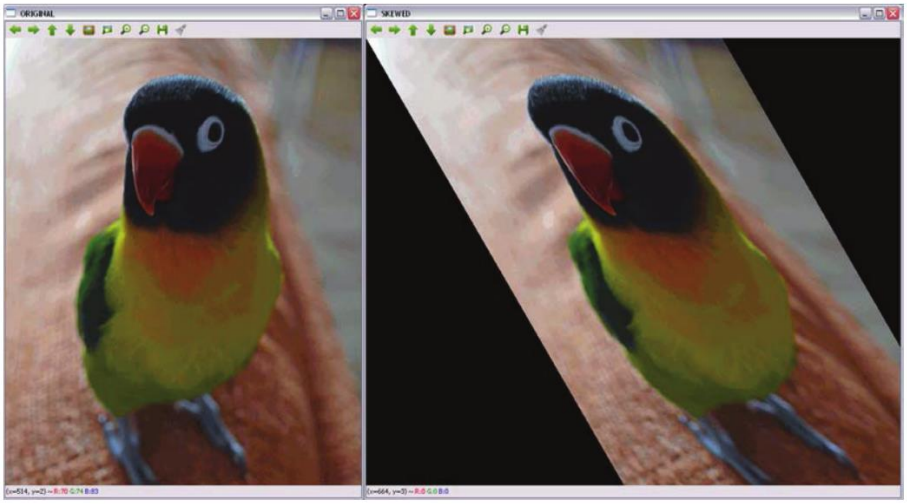


Рис. 2.21. Початкове зображення і воно ж після скоса по горизонталі.

3. ІНФОРМАЦІЙНА СИСТЕМА WECAPTURE

3.1 СТВОРЕННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1.1. Визначення шаблону для побудови інформаційної системи

При створенні інформаційної системи, був обраний шаблон проектування MVVM. Він має низку істотних переваг, які дозволять якісно і швидко створити інформаційну систему. Варто відзначити, що у даного шаблону існує один недолік, недосвідченому розробнику, буде досить складно розділити свою програму на три основні рівні абстракції, такі як:

1. Модель, яка містить в собі всю бізнес-логіку додатка
2. Представлення, в ньому реалізується візуальна частина інформаційної системи
3. Модель представлення, вона служить ланкою між двома рівнями абстракції.

Обраний шаблон допомагає повністю відокремити бізнес-логіку і представлення додатка від призначеного для користувача інтерфейсу (UI). Підтримка чистого поділу між логікою додатка і інтерфейсом допомагає вирішити численні проблеми розробки та полегшити тестування, підтримку і розвиток програми. Він також може значно поліпшити можливості повторного використання коду і дозволяє програмістам і дизайнерам інтерфейсу легше взаємодіяти при розробці своїх відповідних частин програми.

Використовуючи шаблон MVVM, у якому інтерфейс програми, базове представлення і бізнес-логіка поділяються на три окремих абстракції: представлення, яке інкапсулює логіку інтерфейсу; модель представлення, яка реалізує взаємодію між представлення та моделлю; модель, яка інкапсулює бізнес-логіку і дані інформаційної системи. Розберемо дані абстракції більш детально.

Розберемо створення абстракції для візуальної частини програми. Представлення несе відповідальність за визначення структури візуальної

складової створюваної системи, макет і зовнішній вигляд, що користувач бачить на екрані. В ідеалі кожне представлення визначається в спеціальному файлі, в повній ізоляції від коду бізнес-логіки. Однак в деяких випадках, код у представленні може містити логіку призначену для графічного інтерфейсу, даний код реалізує візуальну поведінку інтерфейсу, який часто важко реалізувати в файлі графічної розмітки.

Представлення - це, найчастіше, файл графічної розмітки разом з її супутнім класом. Проте представлення також може бути представлене як шаблон даних, який визначає елементи графічного інтерфейсу, а також використовується для візуального представлення об'єкта, коли він з'явиться.

Існує кілька варіантів для виконання коду в моделі представлення у відповідь на взаємодію з представлення, таких як натискання кнопки або вибору елемента. Якщо елемент керування підтримує команди, які можуть бути прив'язані по заданим властивостям моделі представлення. При виклику команди елемента управління, буде виконуватися код в моделі представлення. Крім команди поведінки можна підключити до об'єкта в представленні слухача і прослуховувати події або виклики команд. У відповідь, можна викликати методи в моделі представлення.

Модель представлення реалізує властивості і команди, до яких представлення може прив'язувати дані і повідомляти про зміну стану через спеціальні колбеки. Властивості і команди, які надає модель представлення, визначають функціональність представлення для графічного інтерфейса, але представлення визначає, як повинна відображатися ця функціональність.

Модель представлення також відповідає за координацію взаємодії представлення з будь-якими необхідними класами моделей. Зазвичай між моделлю представлення і класами моделей існує співвідношення «один до багатьох». Модель представлення може безпосередньо зберігатись в представленні, щоб елементи управління в представленні могли прив'язувати дані безпосередньо до нього. В цьому випадку класи моделей повинні бути розроблені для підтримки прив'язки даних.

Кожна модель представлення надає дані з моделі в формі, яку представлення може легко відображати. Для цього модель представлення іноді виконує перетворення даних. Розміщення цього перетворення даних в моделі представлення є хорошою ідеєю, оскільки воно надає властивості, з якими може зв'язуватися представлення. Наприклад, модель представлення може об'єднати значення двох властивостей, щоб спростити відображення за допомогою представлення.

Щоб модель представлення брала участь у двосторонній прив'язці даних до представлення, її властивості повинні реалізовувати подію PropertyChanged. Моделі представлення задовольняють цій вимозі, реалізуючи інтерфейс INotifyPropertyChanged і викликаючи подію PropertyChanged при зміні властивості. Для колекцій надається зручний для перегляду ObservableCollection інтерфейс. Ця колекція реалізує події про зміну колекції, поліпшуючи роботу розробника, тому що немає необхідності реалізовувати інтерфейс INotifyCollectionChanged для колекцій.

Класи моделей - це не візуальні класи, які інкапсулюють дані програми. Таким чином, модель може розглядатися як представлення моделі додатка, яке зазвичай включає в себе модель даних поряд з бізнес-логікою і логікою баз даних. Приклади об'єктів моделі включають об'єкти передачі даних (DTO), звичайні об'єкти POJOs, створені об'єкти та об'єкти-проксі. Класи моделей зазвичай використовуються в поєднанні з репозиторіями, які інкапсулюють доступ до даних і кешування.

3.1.2. Підключення моделі представлення до візуальної складової інформаційної системи

Моделі представлення можуть бути підключені до представлення, використовуючи можливості прив'язки даних безпосередньо в файлі графічної розмітки. Існує безліч підходів, які можна використовувати для побудови представлення і моделі представлення для зв'язування їх під час виконання. Ці підходи поділяються на дві категорії, відомі як представлення як перша

структура або модель представлення як перша структура. Вибір між ними є проблемою переваги і складності. Проте, всі підходи мають одну і ту ж мету, яка полягає в тому, що для представлення потрібна модель представлення, яка призначає її властивість BindingContext.

З урахуванням першої структури додаток концептуально складається з представлень, які з'єднуються з моделями представлення, від яких вони залежать. Основна перевага цього підходу полягає в тому, що це спрощує створення вільно зв'язаних додатків з можливістю тестування, оскільки моделі представлень не залежать від самих представлень. Також легко зрозуміти структуру програми, слідуючи його візуальної структури, замість того, щоб відстежувати виконання коду, щоб зрозуміти, як створюються і асоціюються класи. Крім того, представлення у першій структурі взаємодіють з навігаційною системою, яка відповідає за побудову сторінок при навігації, що робить модель першої структури складною.

При першій структурі, представлення концептуально складаються з моделей представлення, при цьому системна функція відповідає за визначення виду для моделі представлення. Перша модель здається більш природною, оскільки створення представлення можна абстрагувати, що дозволяє їм зосередитися на логічній структурі, відмінною від UI додатку. Крім того, це дозволяє створювати моделі представлення за допомогою інших моделей, збільшуючи повторюваність коду. Однак цей підхід часто буває складним, і стає важко зрозуміти, як створюються і асоціюються різні частини програми.

Найпростіший підхід полягає в тому, щоб представлення декларативно створило екземпляр відповідної моделі. Коли візуальна частина буде побудована, буде також створено відповідний об'єкт моделі представлення. Цей підхід демонструється в наступному прикладі коду:

```
<ContentPage xmlns:local="clr-namespace:...">
    <ContentPage.BindingContext>
        <local:ViewModelClass />
    </ContentPage.BindingContext>

<Code>...</Code>
```

```
</ContentPage>
```

Коли ContentPage створюється, екземпляр ViewModelClass автоматично створюється і встановлюється як BindingContext представлення. Ця декларативна побудова і призначення моделі представлення в представленні має просту структуру, але має недолік, що для нього потрібно конструктор за замовчуванням (без параметра) в моделі представлення.

Так само представлення може мати код в класі представлення, який призводить до того, що модель представлення задається властивості BindingContext. Це часто виконується в конструкторі представлення, як показано в наступному прикладі коду:

```
public MainView()
{
    InitializeComponent();
    BindingContext = new ViewModelClass(navigationService);
}
```

Програмна побудова моделі представлення в коді конструктора має перевагу, що це досить просто. Однак основним недоліком цього підходу є те, що представлення має забезпечувати модель представлення усіма необхідними залежностями. Використання контейнера для передачі залежності може допомогти підтримувати вільний зв'язок між моделлю представлення та самим представленням. Тому будемо використовувати програмну прив'язку.

Представлення може бути визначене як шаблон даних і пов'язане з типом моделі представлення. Шаблони даних можуть бути визначені як ресурси або вони можуть бути визначені всередині елемента керування, який відобразить модель представлення. Вміст елемента управління являє собою екземпляр моделі представлення, а шаблон даних використовується для його візуального представлення. Цей метод є прикладом ситуації, коли спочатку створюється модель представлення, а потім створюється саме представлення. Так само, раціонально використовувати автоматичне створення моделі представлення за допомогою локатора моделі представлення.

Локатор моделі представлення - це спеціальний клас, який управляє створенням моделей представлення і їх асоціацією з представлення. У мобільних додатках клас ViewModelLocator має прикріплену властивість AutoWireViewModel, яка використовується для прикріплення моделей представлення з представленням. У нашому випадку, прикріплення цієї властивості має значення true, щоб вказати, що модель представлення повинна автоматично підключатися до представлення, як показано в наступному прикладі коду:

```
viewModelBase:ViewModelLocator.AutoWireViewModel="true"
```

Властивість AutoWireViewModel створюється зі значенням false, коли його значення змінюється, викликається обробник події OnAutoWireViewModelChanged. Метод OnAutoWireViewModelChanged моделі представлення використовує підхід, заснований на угодах. В цих угодах передбачається, що:

- Моделі представлення знаходяться в тій же збірці, що і типи представлення.
- представлення знаходяться в просторі Views.
- Моделі представлення знаходяться у просторі дочірніх елементів ViewModels.
- Перегляд імен моделей відповідає назвам імен та закінчується на «ViewModel».

Нарешті, метод OnAutoWireViewModelChanged встановлює тип прив'язки BindingContext моделі. Перевага такого підходу полягає в тому, що програма має один клас, який відповідає за створення моделей представлення і їх з'єднання з представленням.

Всі моделі представлення і класи моделей, доступні для прив’язки, повинні реалізовувати інтерфейс INotifyPropertyChanged. Реалізація цього інтерфейсу в моделі представлення або класі моделі дозволяє класу надсилати повідомлення про зміни будь-яких елементів управління, пов'язаних з даними

в представленні, коли змінюється значення базової властивості. Інформаційна система має бути розроблена для правильного використання подій про зміну властивостей, виконавши вимоги наведені нижче.

- Завжди реалізовувати подію PropertyChanged, якщо змінюється значення публічної властивості. Не можна ігнорувати подію PropertyChanged.
- завжди викликати подію PropertyChanged для будь-яких обчислених властивостей, значення яких використовуються іншими властивостями в моделі або моделі представлення.
- Завжди викликати подію PropertyChanged в кінці методу, який змінює властивість або коли об'єкт, як відомо, знаходиться в безпечному стані. Призупинення події перериває операцію, синхронно викликаючи обробники подій. Якщо це відбувається в середині операції, ми можемо передати об'єкт в функцію зворотного виклику, коли він знаходиться в небезпечному, частково оновленому стані. Крім того, каскадні зміни можна ініціювати подіями PropertyChanged. Каскадні зміни зазвичай вимагають, щоб оновлення були завершені до того, як каскадні зміни будуть безпечними для виконання.
- Ніколи не викликати подію PropertyChanged, якщо властивість не змінюється. Це означає, що перед викликом події PropertyChanged необхідно порівняти старі і нові значення.
- Ніколи не можна викликати подію PropertyChanged в конструкторі моделі представлення, якщо ініціалізується властивість. Елементи управління, прив'язані до даних у представлення, не будуть підписуватися на отримання повідомлень про зміни на цьому етапі.

Інформаційна система використовує клас ExtendedBindableObject для надсилання повідомлень про зміни, що показані в наступному прикладі коду:

```
public abstract class ExtendedBindableObject : BindableObject
{
    public void RaisePropertyChanged<T>(Expression<Func<T>> property)
    {
        var propertyName = GetMemberInfo(property).Name;
        OnPropertyChanged(propertyName);
    }

    private MemberInfo GetMemberInfo(Expression expression)
    {
        ...
    }
}
```

Клас BindableObject реалізує інтерфейс INotifyPropertyChanged і містить метод OnPropertyChanged. Клас ExtendedBindableObject містить метод RaisePropertyChanged для виклику повідомлення про зміну властивості, при цьому використовує функції, що надаються класом BindableObject.

Кожен клас моделі представлення успадковує клас ViewModelBase, який, в свою чергу, походить від класу ExtendedBindableObject. Тому кожен клас моделі представлення використовує метод RaisePropertyChanged в класі ExtendedBindableObject для передачі події про зміну властивості. У наступному прикладі показано, як викликається подія про зміну властивості за допомогою виразу лямбда:

```
    public bool IsUserLogin
{
    get
    {
        return _isUserLogin;
    }
    set
    {
        _isUserLogin = value;
        RaisePropertyChanged(() => IsUserLogin);
    }
}
```

Використання виразу лямбда таким чином передбачає невелику «вартість» виконання, тому що вираз лямбда має реалізовуватися для кожного виклику. Хоча «вартість» виконання невелика і зазвичай не впливає на додаток, витрати можуть нашаровуватись, коли є багато подій про зміни. Однак перевага такого підходу полягає в тому, що він забезпечує підтримку безпеки і рефакторінга коду, а також економить час компіляції при перейменуванні властивостей.

У додатках реакції зазвичай викликається у відповідь на дію користувача, таке як клацання на кнопці, яке можна реалізувати, створивши обробник подій в файлі з кодом. Однак в шаблоні MVVM відповідальність за реалізацію дії лежить на моделі представлення, і слід уникати розміщення коду в звичайному представленні.

Команди надають зручний спосіб представлення дій, які можуть бути прив'язані до елементів управління в інтерфейсі. Вони інкапсулюють код, який реалізує дію і допомагають зберегти окремо від його візуального представлення в моделі. Бібліотека Forms включає елементи управління, які можуть бути декларативно пов'язані з командою, ці елементи управління викликають функцію, коли користувач взаємодіє з елементом управління.

Поведінка також дозволяє елементам управління бути декларативно пов'язаними з командою. Однак реакція може бути використана для виклику дії, пов'язаної з цілою низкою подій, викликаних елементом управління. Таким чином, поведінка зачіпає багато тих же сценаріїв, що і елементи управління з підтримкою команд, забезпечуючи при цьому велику ступінь гнучкості і контролю. Крім того, поведінку також можна використовувати для зв'язування об'єктів або методів команд з елементами управління, які спеціально не призначені для взаємодії з командами.

3.1.3. Виконання команд

Моделі представлення зазвичай відображають властивості команд для прив'язки до представлення, які є об'єктними, що реалізують інтерфейс ICommand. Ряд елементів управління надає властивість Command, яка може бути прив'язана до об'єкта ICommand, наданим моделлю представленням. Інтерфейс ICommand визначає метод Execute, який інкапсулює сама операція, метод CanExecute, вказує чи може команда бути викликана, подія CanExecuteChanged виникає коли відбуваються зміни, що впливають на виконання команди. Класи Command і Command <T>, реалізують інтерфейс ICommand, де T - тип аргументів Execute або CanExecute.

У моделі представлення повинен бути об'єкт типу Command або Command <T> для кожної відкритої властивості в моделі представлення типу ICommand. Конструктор Command або Command <T> вимагає об'єкт зворотного виклику Action, що викликається при визові методу ICommand.Execute. Метод CanExecute є необов'язковим параметром конструктора і є функцією, яка повертає bool змінну. Даний код показує, як екземпляр Command, який представляє команду register, створюється шляхом вказівки делегата методу моделі представлення реєстру:

```
public var RegisterCommand => new Command (Register);
```

Команда відображається в представленні через властивість, яка повертає посилання на ICommand. Коли метод Execute викликається в об'єкті Command, він просто перенаправляє виклик методу в моделі представлення через делегат, який був зазначений в конструкторі Command.

Асинхронний метод може бути викликаний командою, використовуючи ключові слова async і очікувати вказівки делегата Execute. Це означає, що зворотний виклик є завданням і його слід очікувати. Наприклад, наступний код показує, як екземпляр Command, який представляє команду входу, створюється шляхом вказівки делегата в метод моделі SignInAsync:

```
public var SignCommand => new Command( async () => await SignAsync ())
```

Параметри можуть бути передані в Execute і CanExecute за допомогою класу Command <T> для створення екземпляра команди. Наприклад, наступний приклад показує, як використовується екземпляр Command <T>, щоб вказати, що для методу NavigateAsync потрібно аргумент рядка типу:

```
public ICommand NavigteCommand => new Comand<string>(NavigateAsyn);
```

У класах Command і Command <T> делегат методу CanExecute в кожному конструкторі є необов'язковим. Якщо делегат не вказано, команда поверне значення true для CanExecute. Однак модель представлення може

вказувати на зміну стану команди CanExecute, викликавши метод ChangeCanExecute об'єкта Command, що викликає подію CanExecuteChanged. Будь-які елементи управління в інтерфейсі, які прив'язані до команди, будуть оновлювати статус, щоб відобразити доступність команди, пов'язаної з даними.

У коді показано, як Grid в LoginView зв'язується з RegisterCommand в класі LoginViewModel за допомогою примірника TapGestureRecognizer:

```
<Grid HorizontalOptions="Center"
      Grid.Column="1">
    <Label Text="REG"
           TextColor="Black"/>
    <Grid.GestureRecognizer>
        <TapGestureRecognizers Command="{Binding RegisterCommands}" NumberTapsReq
uired="1" />
    </Grid.GestureRecognizer>

</Grid>
```

Параметр команди також може бути визначений з використанням властивості CommandParameter. Тип очікуваного аргументу зазначений в цільових методах Execute і CanExecute. TapGestureRecognizer автоматично викличе цільову команду, коли користувач взаємодіє з підключеним елементом управління. Параметр команди, якщо він наданий, буде переданий в якості аргументу делегату Execute команди.

Залежності дозволяють додавати функціональні можливості в елементи управління призначеним для користувача інтерфейсом без необхідності їх перевизначення. Замість цього функціональність реалізована в класі представлення і прив'язана до елементу управління, як якщо б він був частиною самого елемента управління. Залежність дозволяє реалізувати код, який зазвичай потрібно писати в представленні, тому що він безпосередньо взаємодіє з API елемента управління, таким чином, що він може бути стисло прив'язаний до елементу управління і упакований для повторного використання більш ніж одному представленні або додатку. В контексті MVVM - корисний підхід для підключення елементів управління до команд.

Залежність, прикріплена до елементу управління за допомогою прикріплених властивостей, називається прикладеною поведінкою. Потім Залежність може використовувати відкритий API елемента, до якого він приєднаний, щоб додати функціональність до цього елементу управління або інших елементів управління в візуальне дерево представлення.

Поведінка Forms - це клас, який успадковується з класу Behavior або Behavior <T>, де T - тип елемента управління, до якого має застосовуватися поведінка. Ці класи надають OnAttachedTo і OnDetachingFrom методи, які повинні бути перевизначені, щоб забезпечити логіку, яка буде виконуватися, коли поведінка буде прив'язана і відокремлена від елементів управління. Клас BindableBehavior <T> унаслідується з класу Behavior <T>. Мета класу BindableBehavior <T> - надати базовий клас Forms, для якого BindingContext залежність повинна бути встановлена на прикріплений елемент управління.

Клас BindableBehavior <T> надає перевизначений метод OnAttachedTo, який встановлює BindingContext у перевизначеного методу OnDetachingFrom, який очищає BindingContext. Крім того, клас зберігає посилання на прикріплений елемент управління у властивості AssociatedObject.

Додаток включає в себе клас EventToCommandBehavior, який виконує команду у відповідь на те, що відбувається подія. Цей клас успадковується від класу BindableBehavior <T>, так що поведінка може зв'язуватися з ICommand і задавати властивість Command.

Методи OnAttachedTo і OnDetachingFrom використовуються для реєстрації і скасування реєстрації обробника подій, у властивості EventName. Потім, коли подія спрацьовує, викликається метод OnFired, який виконує команду.

Перевага використання EventToCommandBehavior для виконання команди при події полягає в тому, що команди можуть бути пов'язані з елементами управління, які не призначені для взаємодії з командами. Крім того, це переміщує код обробки подій для перегляду моделей, де він може бути перевірений модулем.

EventToCommandBehavior особливо корисний для приєднання команди до елементу управління, який не підтримує команди. Наприклад, ProfileView використовує EventToCommandBehavior для виконання OrderDetailCommand, коли подія ItemTapped запускається в ListView, в якому перераховані пункти, як показано в наступному коді:

```
<ListViews>
    <ListViews.Behavior>
        <behavior:EventCommandBehavior
            Command="{Binding OrderDetailCommands}"
            EventName="IsItemTapped"
            EventArgsConverter="{StaticResource ItemTappedEventConvert
er}" />
    </ListViews.Behavior>

<Code>... </Code>

</ListViews>
```

Під час виконання EventToCommandBehavior налаштований на взаємодію з ListView. Коли обрано елемент в ListView, спрацьовує подія ItemTapped і буде виконуватись OrderDetailCommand в ProfileViewModel. За замовчуванням аргументи події передаються у команді. Ці дані перетворюються, оскільки вони передаються між джерелом і метою за допомогою конвертера, зазначеного у властивості EventArgsConverter, який повертає елемент ListView з ItemTappedEventArgs. Тому, коли виконується OrderDetailCommand, обраний елемент передається як параметр дії.

3.2. Робота з інформаційною системою

У даному розділі розглянемо, як працює створена інформаційна система, завантажимо декілька звітів о виконанні лабораторних досліджень. А також покажемо візуальну складову розробленого додатку.

3.2.1. Початок роботи з інформаційною системою

Суть роботи інформаційної системи не складна, користувач повинен зареєструвати аккаунт у сервісі We-Integrate на їх офіційному сайті. Далі необхідно налаштувати поля, в які користувач повинен записувати дані про дослідження. Це можуть бути текстові поля, цифрові, дата, сканер штрих-коду, поле для вводу імейла та інші поля, які завжди присутні у дослідженні. Варто зазначити, що кожне поле може бути підписано, а також можна задати такі параметри полів, як максимальна та мінімальна довжина символів записаних у поле, великими чи малими літерами буде заповнюватись дане поле, чи обов'язково поле для заповнення. Даний підхід дозволяє налаштувати додаток таким чином, щоб користувачі відчували мінімальні незручності при роботі.

Після налаштування серверної частини, усі користувачі, які ввели зареєстрований аккаунт у додаток, будуть мати даний набор полів для вводу даних. Завантаживши додаток на смартфон і запустивши його перший раз, необхідно ввести назву аккаунта і, по можливості, заповнити дані про себе. При успішній авторизації ми побачимо головний екран інформаційної системи зображений на рис. 3.1. Легко бачити, що ми поки не завантажували ніяких звітів, тому що на головному екрані немає ніяких даних. Необхідно натиснути на зелений олівець в правому нижньому кутку екрана, щоб відкрити вікно для створення звіту.

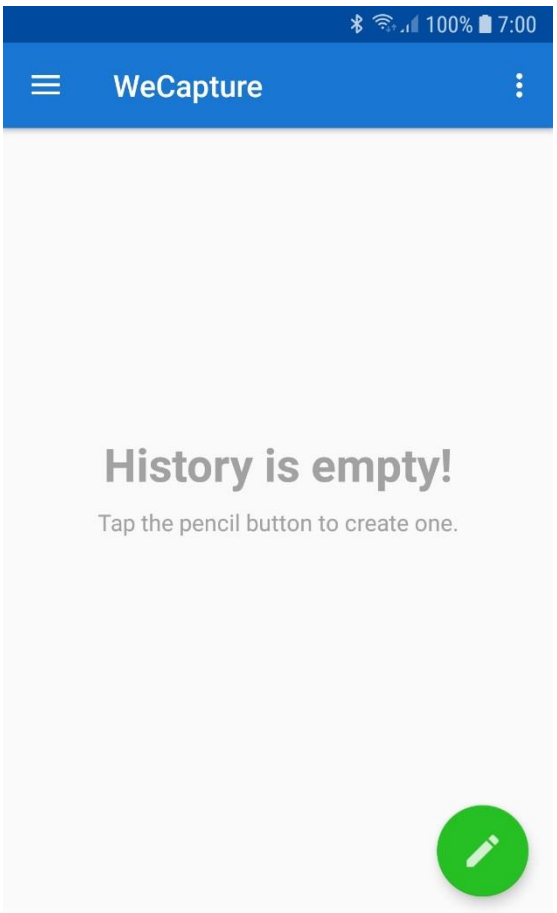


Рис. 3.1. Головне вікно після першого запуску

3.2.2. Створення першого звіту

Після того, як користувач натиснув на зелену кнопку з олівцем, він побачить вікно створення звіту, яке зображене на рис. 3.2. Так як дане вікно тільки відкрилось, у ньому ще немає даних. Необхідно ввести усю інформацію, яка доступна досліднику, після цього можна натиснути на фотоапарат у верхньому правому куті, або на кнопку плюс у нижньому правому куті і обрати звідки користувач хоче завантажити фотографії. На вибір є два варіанти, завантажити з галереї телефона, або робити фотографії під час виконання лабораторного дослідження. Варто відзначити, що кнопка за фотоапаратом одразу відкриває камеру смартфона для швидкого доступу до неї.

Якщо проаналізувати поля у даному екрані, можна побачити поля з напругою і температурою, це поля, які можуть містити тільки цифри, тому операційна система буде показувати клавіатуру у якій доступні лише цифри.

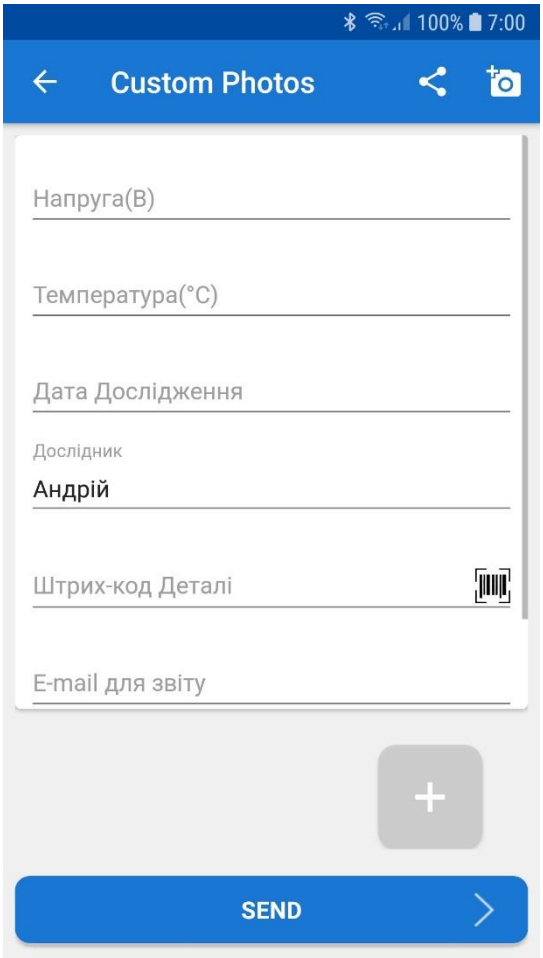
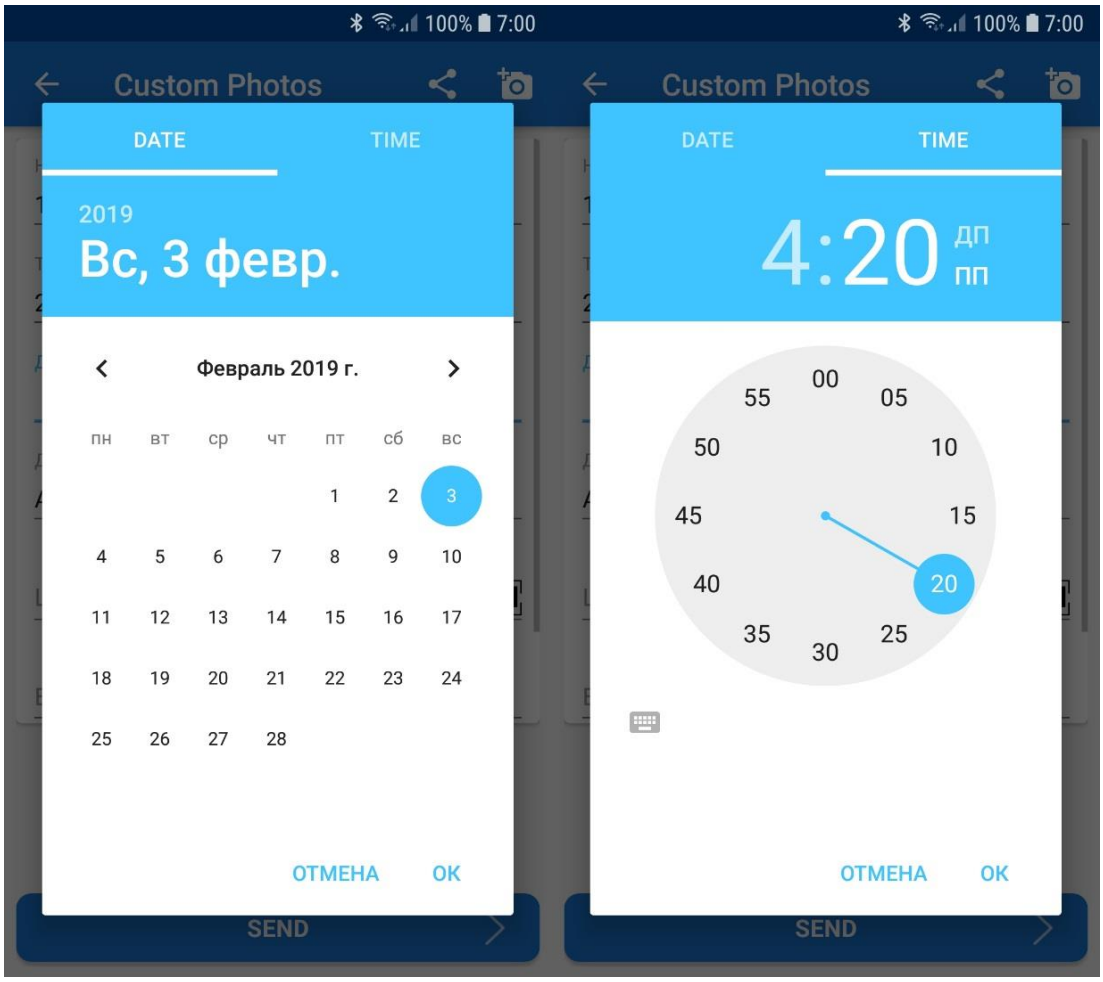


Рис. 3.2. Вікно створення звіту

Третє поле – це поле вибору дати та часу, формат цього поля можна задати на сайті. Після того, як користувач натисне на дане поле, відкриється діалог вибору дати-часу зображений на рис. 3.3. Варто зазначити, якщо у вказаному форматі дати-часу, не вказаний формат часу (HH - години або mm - хвилини), то програма адаптивно буде показувати тільки діалог вибору дати, тому що час все одно ігнорується. Далі в полі дослідника, додаток автоматично підставляє ім’я користувача вказаного у налаштуваннях. Поле штрих-кода, дозволяє власноруч ввести або відсканувати штрих-код деталі,

або будь-якого об’єкта. Щоб запустити сканування, необхідно натиснуту на значок штрих-коду який знаходиться у правій частині поля. Після натиснення, відкриється спеціальна камера, де необхідно розмістити штрих-код по центру камери, коли додаток розпізнає штрих-код, камера автоматично закривається, а розпізнані дані записуються у поле штрих-кода. Наступне поле – це імейл, який користувач вводить для отримання автоматично згенерованого листа, який відправляється після завантаження звіту. Дана функція може бути корисною, коли необхідно проінформувати колег про виконану роботу. На сайті доступний редактор шаблону листа, щоб максимально підлаштовуватись під поставлену задачу.



3.3. Діалог вибору дати та часу.

Ввівши усі необхідні данні, а також зробивши цифрові знімки, вікно інформаційної системи буде вигляд як на рис. 3.4, зроблені знімки будуть зображенні у зменшеному варіанті знизу. Даний підхід дає змогу приблизно подивитись на набір графічних матеріалів які містить звіт. Якщо натиснути на групу цих фотографій, відкриється вікно перегляду зображень, в якому можна як подивитись на набір усіх фотографій, так подивитись одну конкретну. Також існує можливість збільшувати зображення, щоб роздивитись деталі.

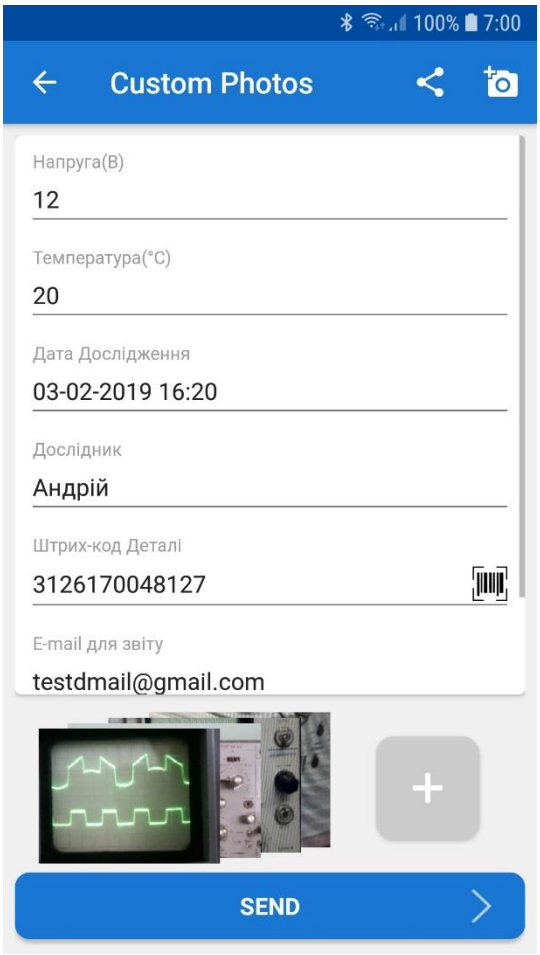


Рис. 3.4. Вікно звіту з заповненими даними.

Якщо необхідно передати отримані зображення колезі або надіслати у соціальну мережу, існує функція шарингу. Необхідно натиснути на кнопку, яка знаходиться поряд з кнопкою фотографії у верхньому правому куті, буде виконана операція обробки зображення, після чого можливо надіслати зображення у будь-який з доступних способів.

Коли користувач натиснув кнопку назад, додаток покаже діалог зображений на рис. 3.5, у якому можна відмінити вихід з вікна, можна зберегти звіт у історію, щоб відправити пізніше, а можна видалити його. Звіти можна відкривати з історії, як відправлені так і просто збережені, коли користувач виходить з відкритого звіту, то додаток може оновити дані цього звіту, або при виборі видалення, залишить оригінальний звіт, без внесення змін. Для цього була реалізована система клонування звітів, разом з їх графічними даними, щоб при його видаленні не постраждав оригінальний звіт.

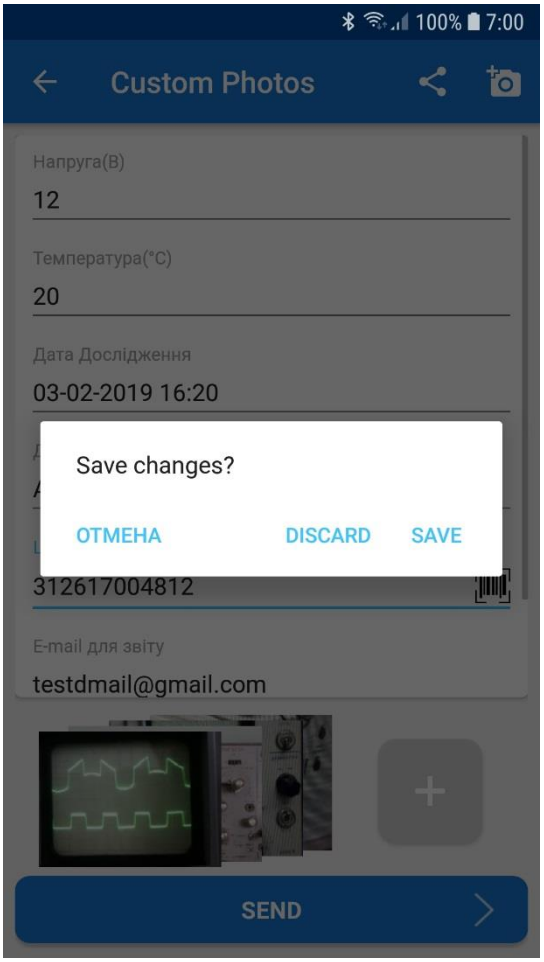


Рис. 3.5. Діалог виходу з екрану створення звіту

Якщо дослідник ввів усі необхідні дані, зробив графічні знімки які необхідно завантажити, він може натиснути кнопку надіслати. Програма, якщо знайде якусь помилку, наприклад, незаповнене поле яке помічене як обов’язкове або не зроблені знімки, повідомить про це користувача. Якщо усе

зроблено правильно, після натискання кнопку надіслати, програма покаже діалог очікування та почне етап обробки та надсилання даних. Користувач навіть на здогадується, поки він очікує завершення операції смартфон проводить складніші обчислення. За цей час, програма встигає зменшити фотографію, прибрати шуми, збільшити контрастність, а потім надіслати кожну фотографію на сервер. Коли остання фотографія буде завантажена, користувач побачить діалог зображений на рис. 3.6, який сигналізує про успішне завантаження звіту на сервер. Даний діалог автоматично зачинить вікно після 5 секунд очікування або після натискання кнопки ОК.

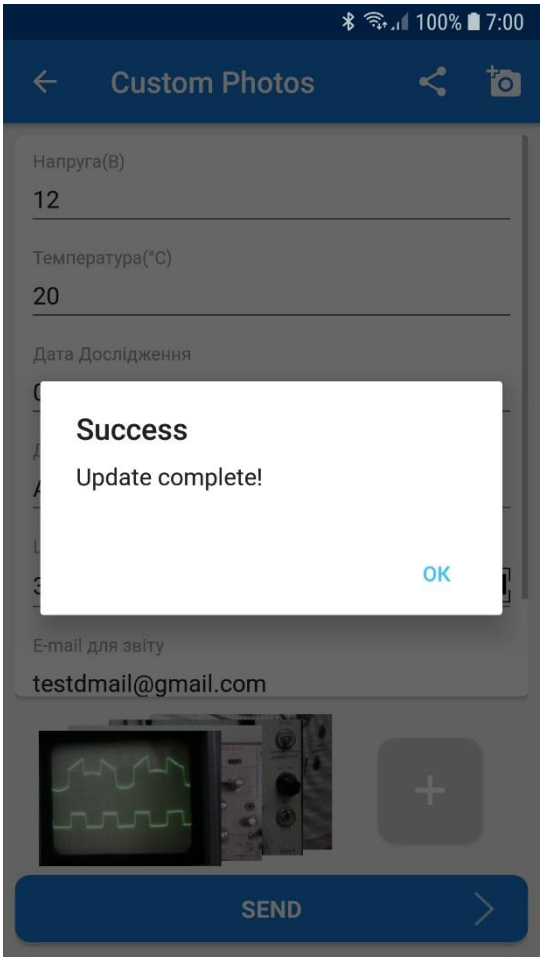


Рис. 3.6. Діалог успішного завантаження зображення.

Якщо звіт був успішно надісланий або просто збережений в історію, він буде відображений на головному екрані додатку, як можна побачити на рис. 3.7.

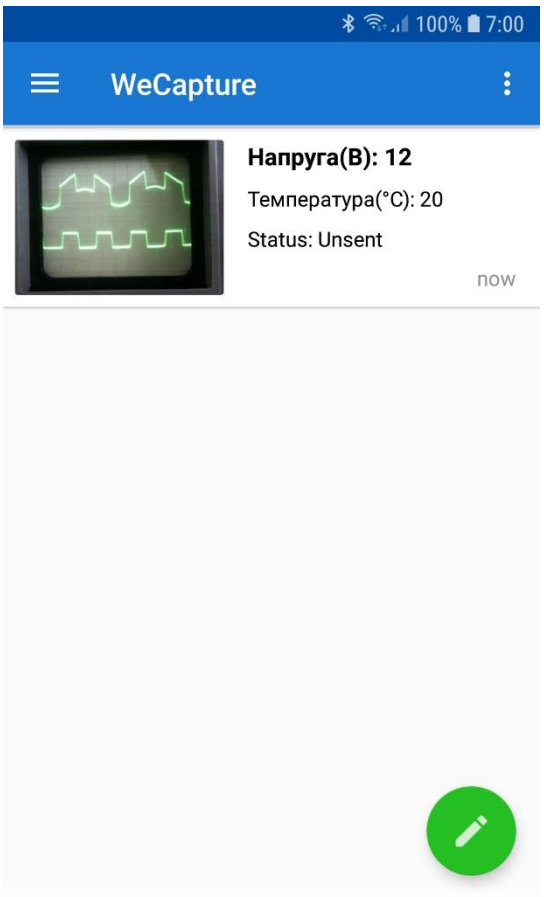


Рис. 3.7. Головний екран після додавання звіту в історію

3.2.3. Додаткові можливості інформаційної системи

Якщо натиснути на три риски у лівому верхньому кутку екрану, відкриється бокове меню, як показано на рис. 3.8. У даному меню, можна обрати такі пункти як, створити звіт із фотографіями, створити звіт з конвертацією графічних даних у форматі pdf, також можна відкрити повноцінну історію відправлених та збережених звітів.

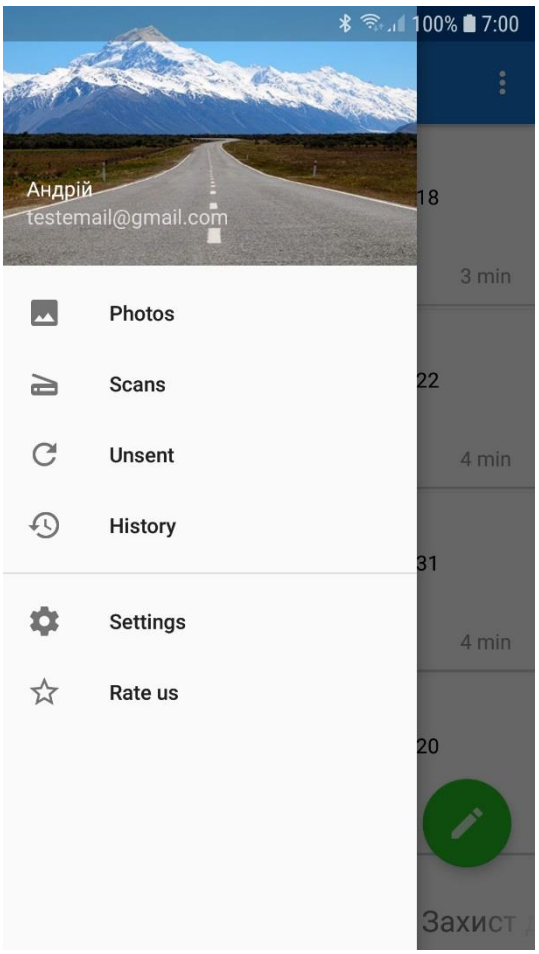


Рис. 3.8. Бокове меню.

Як вже зазначалось раніше, на головному екрані реалізований список який містить коротку інформацію про звіт, а також останню додану фотографію, щоб було легко знайти необхідний звіт. Щоб не перевантажувати головний екран, кількість звітів обмежена 20 одиницями. При інтенсивній роботі головний екран буде виглядати як на рис. 3.9. Кожний елемент у списку відокремлений в спеціальну картку, яка відображає останнє зображення зі звіту, перші 2 заповнені поля, статус звіту (був він відправлений або ні), а такою час коли звіт був добавлений або оновлений в історії. Натиснувши на будь-який із звітів, він автоматично відкриється у вікні створення звіту, з заповненими полями. Користувач може редагувати його, добавляти нові графічні дані, а також при необхідності зберегти звіт або завантажити його.

Рис. 3.9. Типовий вигляд головного екрану.

На рис. 3.9 знизу, легко бачити рухомий рядок, текст даного рядка редагується на спеціальній сторінці у мережі Інтернет, де адміністратор аккаунта може розмістити найважливішу інформацію для дослідників, натиснувши на даний рядок, додаток автоматично відкриє мобільний браузер з розширеною версією статі.

Відкривши бокове меню, можна вибрати пункти для перегляду збережених звітів або відправлених. Після вибору одного із цих пунктів відкриється вікно, як наприклад на рис. 3.10. На даному екрані, користувач може переглянути коротку інформацію про звіти, а також існує можливість пошуку за значеннями будь-якого з полів. При натисканні на елемент зі списку буде відкриватись вікно з даним звітом.

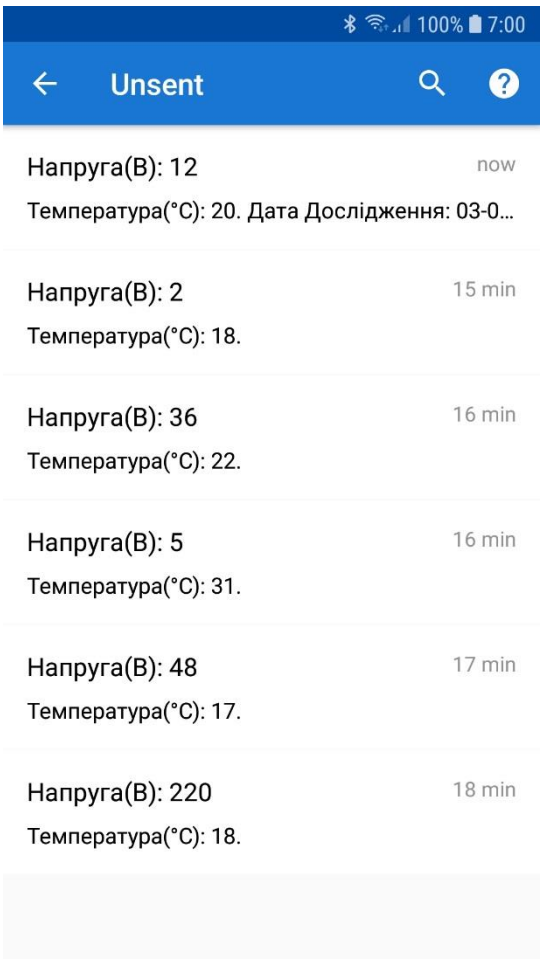


Рис. 3.10. Екран історії збережених звітів.

Варто відзначити, що існує одна дуже цікава функція, яка дозволить уніфікувати графічні дані звітів. Якщо послідовність зображень які робить дослідник дуже важлива для дослідження, тоді можливо заздалегідь задати кількість фотографій, написати їх опис, щоб користувачі не плутались який знімок йде за яким. На рис. 3.11 наведений приклад такої камери.

Адміністратор аккаунта, може на сайті скласти список, який буде надісланий усім встановленим додаткам з цим аккаунт кодом. Коли користувач запускає камеру, вона розміщує з боку список усіх фотографій які потрібно зробити, з коротким описом. Також у нижній частині екрану буде з’являтись повний опис умов, які необхідні виконуватись в певному зображенні. Дана функція дозволить уніфікувати звіти, щоб у великому колективі дослідників не виникало проблем з різними послідовностями візуальних даних.

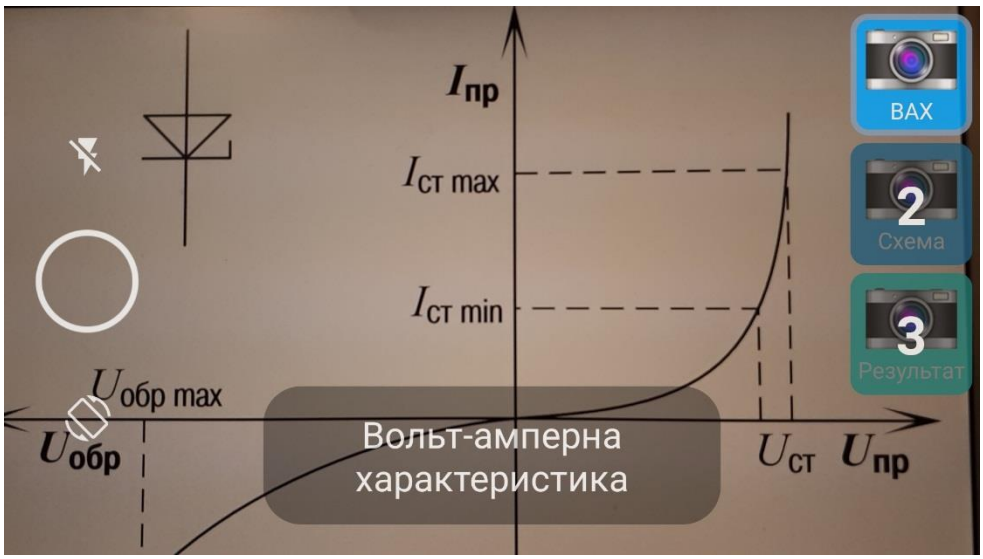


Рис. 3.11. Камера с заздалегідь заданими параметрами.

3.3. Розробка стартап проекту

3.3.1. Опис ідеї проекту

Розроблена інформаційна система покликана полегшити роботу дослідників. Вона дуже універсальна, що дозволить використовувати її у різних випадках, а також налаштовувати її під свої потреби. Розроблений додаток може працювати під операційною системою Android, яка займає більше 85% ринку сучасних смартфонів. Дана інформаційна система, підтримує дуже широкий діапазон версій операційної системи, що дозволить встановити її на 95% усіх смартфонів, які знаходяться у користувачів. Також наявна підтримка планшетів для більш комфортної роботи з додатком. Дослідник може записувати необхідні данні у додаток і робити цифрові знімки, які будуть збереженні. Після закінчення роботи, можна завантажити усю наявну інформацію у хмарне сховище, звідки воно буде автоматично синхронізоване з усіма комп’ютерами, які встановили спеціальний додаток. Усі фотографії будуть відсортовані по папкам, що значно полегшить роботу над дослідженням.

Таблиця 3.1. Опис ідеї стартапу

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Використання інформаційної системи для синхронізації фотографій зі смартфона на комп'ютер, з можливістю заповнення робочих даних.	1. Лабораторні дослідження	Не потрібно синхронізувати данні вручну.
	2. Страхування авто	Можливість заповнення даних автомобіля
	3. Ремонтні роботи	Для надання візуальних даних про виконаний прогрес роботодавцю.

Таблиця 3.2. Визначення сильних, слабких та нейтральних характеристик ідеї проекту

№ п/п	Техніко-економічні характеристики ідеї	(потенційні) товари/концепції конкурентів			W (слабка сторона)	N(нейтральна)	S(сильна сторона)
		Інформаційна система WeCapture	Конкурент1	Конкурент2			
1.	Економічні	50 у.о.	40 у.о.	69 у.о.		+	
2.	Призначення	Синхронізація візуальних даних.	Синхронізація візуальних даних.	Синхронізація візуальних даних.		+	
3.	Надійності	Довговічність - 4 роки	Довговічність - 3 роки	Довговічність - 5 років			+
4.	Технологічні	Створена спеціалізована камера, можливість власноруч створити поля.	Наявна лише синхронізація візуальних даних.	Відсутня камера з обов'язковою послідовністю зображень			+

5.	Ергономічні	-	-	-		+	
6.	Органолептичні	-	-	-		+	
7.	Естетичні	-	-	-		+	
8.	Транспортабельності	-	-	-		+	
9.	Екологічності	Екологічно	Екологічно	Екологічно		+	
10.	Безпеки	Безпечно	Безпечно	Безпечно		+	

Конкурент 1: додаток Image Plus від компанії «Planning Plus»

Конкурент 2: додаток від компанії «Preferred Patron»

3.3.2. Технологічний аудит ідеї проекту

Таблиця 3.3. Технологічна здійсненність ідеї проекту

№ п/п	Ідея проекту	Технології реалізації	Наявність технологій	Доступність технологій
1.	Обробка зображення перед завантаженням.	Розроблена технологія для усунення шумів, підвищення контрастності та зменшення розміру зображення	В наявності	Доступна
2.	Втілення архітектури додатку MVVM.	Реалізація патерну проектування, який дозволить масштабувати додаток у майбутньому.	В наявності	Доступна

3.3.3. Аналіз ринкових можливостей запуску стартап-проекту

Таблиця 3.4. Попередня характеристика потенційного ринку стартап-проекту

№ п/п	Показники стану ринку (найменування)	Характеристика
1	Кількість головних гравців, од	4
2	Загальний обсяг продаж, грн/ум.од	100 000 ум.од.

3	Динаміка ринку (якісна оцінка)	Зростає
4	Наявність обмежень для входу (вказати характер обмежень)	Немає
5	Специфічні вимоги до стандартизації та сертифікації	Немає
6	Середня норма рентабельності в галузі (або по ринку), %	200%

Ринок є привабливим для потенційного входження.

Таблиця 3.5. Характеристика потенційних клієнтів стартап-проекту

№ п/п	Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
	Синхронізація та впорядкування візуальних даних дослідження	Дослідницькі лабораторії, заклади науки,	-	• Висока надійність роботи додатку. • Зручність користування. • Якість візуальних даних.

Таблиця 3.6. Фактори загроз

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
1.	Конкуренція	Компанії які давно на ринку мають постійну базу клієнтів.	Активна маркетингова кампанія, знаходження нових зацікавлених фірм.
2.	Постійні витрати	Для роботи необхідні сервери, оренда яких постійно потребує капіталовкладень.	Розширення кількості серверів по мірі розширення бази клієнтів і скорочення за потребою.

Таблиця 3.7. Фактори можливостей

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
1.	Конкуренція	Конкуренція буде спряти якості роботи з клієнтами, а також надійності додатку.	Покращення роботи фірми з клієнтами та стимул покращувати додаток.
2.	Розвиток	Кожен клієнт висловлює свої побажання, тому з часом додаток може здобути велику кількість корисних можливостей.	Здобутий функціонал може дозволити відкрити нові ринки збуту додатку.

Таблиця 3.8. Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
Тип конкуренції: чиста конкуренція.	На ринку присутня велика кількість фірм, не існує фірми-лідера.	Підвищити надійність додатку, прислухатись до побажань користувачів.
Рівень конкурентної боротьби: світовий	Постачальник послуги конкурують на міжнародному рівні.	Необхідно застосовувати активну маркетингову кампанію у потенційних сферах застосування.
Галузева ознака: внутрішньога-лузева	Використання у одній галузі.	Постійне вдосконалення продукту.
Конкуренція за видами товарів: товарно-видова	Конкуренція між товарами одного виду.	Збільшення зручності користування продуктом, необхідно прислуховуватись до побажань користувачів.
За характером конкурентних переваг: цінова	Продаж товару за нижчими цінами ніж у конкурентів.	Зменшення вартості обслуговування.
Інтенсивність: немарочна	Роль торгової марки не вирішальна.	Необхідність у рекламуванні.

Таблиця 3.9. Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
	«Image Plus»	Немає	Продається безпосередньо розробниками	Вимоги до надійності продукту	Замінників немає
Висновки:	Конкурент мають базу постійних клієнтів	Конкуренти не передбачаю-ться	Постачальники не можуть диктувати умови роботи на ринку	Висока надійність продукту, зручність в його використанні	Обмежень немає

Для здобуття конкурентоспроможності необхідно розробити максимально якісний додаток, щоб не відштовхувати нових клієнтів. Необхідно застосовувати індивідуальний підхід, щоб задоволені клієнти рекомендували своїм колегам.

Таблиця 3.10. Обґрунтування факторів конкурентоспроможності

№ п/п	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
1.	Краща якість візуальних даних.	Розроблений алгоритм обробки зображень дозволяє зберегти якість зображення при кодуванні.
2.	Висока надійність.	Застосування нового підходу до взаємодії між компонентами, дозволяє підвищити надійність додатку.
3.	Зручність.	Активна робота з користувачами, дозволила реалізувати усі їх прохання.

Таблиця 3.11. Порівняльний аналіз сильних та слабких сторін проекту

№ п/п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг конкурентів у порівнянні						
			-3	-2	-1	0	1	2	3
1.	Краща якість візуальних даних.	19							+
2.	Висока надійність.	18						+	
3.	Зручність.	16					+		
4.	Економічний (ціна товару)	12					+		

Таблиця 3.12. SWOT- аналіз стартап-проекту

Сильні сторони: Розроблений алгоритм обробки зображень.	Слабкі сторони: Вартість оренди високошвидкісних серверів.
Можливості: Розробити різні цінові тарифи для оптимізації хмарного сховища.	Загрози: Побажання клієнтів можуть не окупити затрати на їх провадження.

Таблиця 3.13. Альтернативи ринкового впровадження стартап-проекту

№ п/п	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1.	Пошук зацікавлених компаній, пошук інвесторів проекту, пошук працівників, налагодження роботи додатку з серверами, вихід на ринок	50%	3 роки

2.	Пошук інвесторів проекту, пошук зацікавлених компаній, пошук працівників, налагодження роботи додатку з серверами, вихід на ринок	70%	2 роки

Обрано альтернативу №2.

3.3.4. Розроблення ринкової стратегії проекту

Таблиця 3.14. Вибір цільових груп потенційних споживачів

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1.	Державні установи	Середня	Високий	Низька	Висока
2.	Приватні підприємства	Висока	Високий	Середня	Середня
Які цільові групи обрано: обрано цільову групу № 2					

Таблиця 3.15. Визначення базової стратегії розвитку

№ п/п	Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
1.	3	Концентрація на деякому сегменті ринку.	Адаптація до умов ринку.	Стратегія спеціалізації.

Таблиця 3.16. Визначення базової стратегії конкурентної поведінки

№ п/п	Чи є проект «першопро-ходьцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки
1.	Ні	Так	Ні	Заняття конкурентної ніші

Таблиця 3.17. Визначення стратегії позиціонування

№ п/п	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
1.	Висока надійність, зручність.	Стратегія заняття конкурентної ніші	Розроблений алгоритм обробки зображення.	Робота з клієнтами, висока надійність, оптимальне співвідношення ціна/якість.

3.3.5. Розроблення маркетингової програми стартап-проекту

Таблиця 3.18. Визначення ключових переваг концепції потенційного товару

№ п/п	Потреба	Вигода, яку пропонує товар\технологія	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)
1.	Збільшення якості графічних даних.	Алгоритм обробки зображення перед завантаженням	Збільшення якості зображення при такому ж розмірі файлу.
2	Висока надійність.	Використання додатку без збоїв.	Висока надійність за рахунок використання архітектури MVVM.

Таблиця 3.19. Опис трьох рівнів моделі товару

<table>
<tr><td>Рівні товару</td><td colspan="3">Сутність та складові</td></tr>
<tr><td>I. Товар за задумом</td><td colspan="3">Синхронізація візуальних даних с застосуванням розробленого алгоритму для покращення зображення.</td></tr>
<tr><td rowspan="5">II. Товар у реальному виконанні</td><td>Властивості/характеристики</td><td>М/Нм</td><td>Вр/Тх</td></tr>
<tr><td>1. Покращення зображення.
2. Висока надійність.
3. Зручний інтерфейс.</td><td>М
М
М</td><td></td></tr>
<tr><td colspan="3">Якість: повністю відповідає стандартам від Google.</td></tr>
<tr><td colspan="3">Пакування: -</td></tr>
<tr><td colspan="3">Марка: -</td></tr>
<tr><td rowspan="2">III. Товар із підкріпленням</td><td colspan="3">До продажу: інструктаж</td></tr>
<tr><td colspan="3">Після продажу: тех. підтримка</td></tr>
<tr><td colspan="4">За рахунок чого потенційний товар буде захищено від копіювання: авторське право</td></tr>
</table>

Таблиця 3.20 - Визначення меж встановлення ціни

№ п/п	Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
	-	40-80 у.о. в місяць	10 000 у.о. і вище	40-100 у.о. в місяць

Таблиця 3.21. Формування системи збуту

№ п/п	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
1	Оформлення підписки на сайті.	Інформування, робота з клієнтами.	Канал нульового рівня.	Безпосередній збут товару від виробника.

Таблиця 3.22. Концепція маркетингових комунікацій

№ п/п	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
	Клієнти будуть постійно використовувати продукт, багато разів в день.	Прямі –клієнт спілкується безпосередньо з виробником.	Зручний інтерфейс, який інтуїтивно зрозумілий. Висока якість зображень після обробки.	Поширення інформації про переваги створеного додатку. Презентація потенційним покупцям готового продукту.	Звернення ґрунтується на перевагах розробленої системи над конкурен-тами.

3.3.6. Результати розробки стартапу

Можливість ринкової комерціалізації та впровадження проекту є досить значною, це пов’язано з наявністю переваг додатку над конкурентами та відсутністю високої конкуренції на даному ринку. Ринок додатків, які використовуються у наукових дослідженнях або у інших галузях, де можна використовувати дану інформаційну систему, тільки починає розвиватись, це може позитивно сказатись на інвестиціях у дану область.

ВИСНОВОК

Отже, результатом роботи над магістерською дисертацією можна вважати створення функціонуючої інформаційної системи, яка повністю відповідає поставленим перед нею цілям.

По-перше, при побудові досить великої інформаційної системи, була розроблена її структуру, визначені основні елементи, які присутні у системі та визначено як вони будуть взаємодіяти між собою. Щоб виконати поставлену ціль, було проаналізовано велику кількість інформації, тому що помилка при проектуванні могла б перетворитись на серйозну проблему при реалізації інформаційної системи. Саме тому, вивченню підходів побудови інформаційної система, а також видам архітектури додатків було приділено багато уваги та часу. Як результат, обрана архітектура MVVM ідеально підійшла під поставлену задачу. Це дозволило побудувати інформаційну систему, з можливістю її подальшого розширення та тестування, а також дуже ефективного використання написаного коду.

По-друге, у більшості випадків даний додаток використовується в приміщеннях, де яскравість світла не завжди на високому рівні. Якщо для людського ока це не є критично, то для ПЗС матриць мобільних телефонів це випливає у велику кількість шумів, які необхідно усунути. Стандартна обробка зображень дуже сильно відрізняється у різних виробників смартфонів, деякі жертвують деталями, щоб прибрати як можна більше шумів, інші навпаки, мінімально розмивають зображення, щоб зберегти деталі, але з меншою якістю. Більшість виробників зосереджують увагу на знімках, які роблять на вулиці, саме тому у нашому випадку ми отримуємо не найкращий результат, а саме головне він може суттєво відрізнятись один від одного. Саме тому, було прийнято рішення розробити метод обробки зображень, щоб отримати прогнозований результат і заточити алгоритм для роботи у спеціальних умовах. Як результат, був отриманий алгоритм, який застосовує послідовно напівказуальну фільтрацію, медіанний фільтр, фільтри Гаусса і Лапласа, щоб

усунути шуми, а також збільшити контрастність зображення. Звичайно, даний алгоритми не можна вважати ідеальним, в деяких випадках він може замилювати деталі, але можна з упевненістю вважати, що він повністю виконує поставлену задачу.

По-трете, після створення і тестування додаток зарекомендував себе як надійний інструмент для роботи над серйозними проектами. У ньому реалізована система для прийому та надсилання помилок при роботі, на спеціалізований сервер, що дозволяє оперативно дізнаватись про несправності, що виникли та усувати їх, до того як користувач повідомить про них. Впровадження «матеріального» дизайну, згідно з рекомендаціями компанії Google, яка створила дану операційну систему, дозволило створити інтерфейс, який інтуїтивно зрозумілий. Даний підхід дозволяє використовувати даний додаток, без проходження спеціальної підготовки для користувача, а створена системи підказок тільки сприяє цьому процесу.

Саме тому, можна вважати що розроблена інформаційна система у повній мірі відповідає поставленим задачам. Користувачі, що випробували даний додаток висловлювали своє схвалення при його використанні.

ПЕРЕЛІК ПОСИЛАНЬ

1. Міжнародна наукова інтернет-конференція "Інформаційне суспільство: технологічні, економічні та технічні аспекти становлення (випуск 33)" / Збірник тез доповідей: випуск 33 (м. Тернопіль, 13 листопада 2018 р.). – Частина 1. – Тернопіль. – 2018. – 126 с.
2. Гонсалес Р. Цифрова обробка зображень: ТЕХНОСФЕРА, 2005. - 761 с.
3. Красильников Н. Н. Принципи обробки зображень, засновані на виді їх семантичної структури // Інформаційно-керуючі системи. -2008. - № 1. - С. 2-6.
4. Madhav V. Chitturi, Juan C. Medina, Rahim (Ray) F. Benekohal. Effect of Shadows and Time of Day on Performance of Video Detection Systems at Signalized Intersections. // Transportation Research Part C: Emerging Technologies. - 2010 Vol. 18. Issue 2. - P. 176-186.
5. Бєляєв Е. А., Тюрліков А. М. Алгоритми оцінки руху в задачах стиснення відеоінформації на низьких бітових швидкостях // Комп'ютерна оптика. - Т. 32 (2008). - № 4. - С. 69-76.
6. Дубейковскій В.І., Ефективне моделювання з CA ERwin Process Modeler (BPwin; AllFusion Process Modeler), - М .: Діалог-МІФІ, 2009, - 384 с.
7. Анісімов Б. В., Курганов В. Д., Злобін В. К. Розпізнавання і цифрова обробка зображень. - М.: Вища школа, 1983. - 295 с
8. Телебачення: підручник для вузів / Під ред. В. Е. Джаконія. 4-е изд. - М. : Гаряча лінія-Телеком, 2007. - 616 с.
9. Landand E., McCann J. Lightness and retinex theory // Journal of the optical Society of America. - 1991. - vol.61, N.1. - pp.1-11.
10. Гласман К. Конференція IBC2005: Мобільне телебачення // Інформаційно-технічний журнал "625". - 2005. - № 10. - С. 66-73.
11. A.S. Georghiades, P. Belhumeur and D. Kriegman. From few to many: Illumination Cone Models for Face Recognition under Variable Lighting and

Pose // In Proc. IEEE Trans. On Pattern Analysis and Machine Intelligence. - 2001.

12. Куляс С.М. Загальні принципи перетворення зображень з метою поліпшення їх візуальної якості // Нові інформаційні технології: матеріали шостого науково-практичного семінару. / МДІ електроніки і математики. - 2005. - с.24-25.
13. RaskarR, IlieA., Jingyi Y. Image fusion for context enhancement and video surrealism. // In: ProceedingsNPAR3-rdinternational symposium on non-photorealistic animation and rendering. - Annecy, France, 2004. - P.85-93.
14. Гонсалес Р., Вудс Р. Цифрова обробка зображень. - Москва: Техносфера, 2012. - 1104 с.
15. Рамбо Дж., Блаха М., UML 2.0. Об'єктно-орієнтоване моделювання і розробка, - СПб .: Москва, 2007. - 544 с.
16. Красильников Н. Н. Вплив шумів на контрастну чутливість і раррешающую здатність приймальної телевізійної трубки // Техніка телебачення. - 1958. - Вип. 25. - С. 26-43.
17. Роберт Дж. Мюллер, Проектування баз даних та UML, - М .: Лорі, 2013, - 432 с.
18. Adnin Y., Moses Y., Ullman S. Face recognition: The problem of compensating for changes in illumination direction. // IEEE Transactions on Pattern Analysis and Machine Intelligence. - 1997. - Vol. 19, No. 7. - P. 712-732.
19. Гольденберг Л.М. Цифрова обробка сигналів. - М .: Навчальний посібник для вузів, 1997. - 256с.
20. Красильников Н. Н. Визначення координати глибини в методах 3D-сканування: Тези доповідей IX Міжнародної конференції "Прикладна оптика - 2010", м.Санкт-Петербург, жовтень 2010 р
21. Ватолин Д., Ратушняк А., Смирнов М., Юкін В. Методи стиснення даних. - М.: Діалог-МІФІ, 2002. - 381 с.

22. Антипин М. В. Інтегральна оцінка якості телевізійного зображення. - Л. : Наука, 1970. - 154 с.
23. Василенко Г. І., Тараторін А.М. Відновлення зображень. - М.: Радио и связь, 1986. - 302 с.
24. Глезер В. Д., Цуккерман І. І., Цикунова Т. М. Про пропускної здатності зору // Техніка кіно і телебачення. - 1961. - № 3. - С. 27-32.
25. ГОСТ 34.602-89. Інформаційна технологія. Комплекс стандартів на Автоматизовані системи. Технічне завдання на создание автоматизованої системи.
26. Рєпін В.В., Еліферов В.Г., Процеси та управління. Моделювання бізнес-процесів, - М .: Манн, Іванов і Фербер, 2013. - 544 с.
27. Брацлавець П. Ф., Росселевіч І. А., Хромов Л. І. Космічне телебачення. - М .: Космос, 1973. - 248 с.
28. Красильников Н. Н., Красильникова О. І. Визначення координати глибини по 2D-зображення // Оптичний журнал. - Т. 78 (2011). - № 10.
29. Претт У. К., Фожра О. Д., Гагаловіч А. Зорове розрізнення стохастичних текстур полів // ТІЕР. - Т. 26 (1978). - № 11.
30. Новиков Ф.А., Іванов Д.Ю., Моделювання на UML [Електронний ресурс]: Інтернет книга. -Електронні дані. - 2013. - Режим доступу: http://book.uml3.ru.
31. Гонсалес Р., Вудс Р. Цифрова обробка зображень / Перекл. з англ. - М.: Техносфера, 2006. - 1070 с.
32. Ларман К., Застосування UML 2.0 і шаблонів проектування. Введення в об'єктно-орієнтований аналіз, проектування і ітеративну розробку, - М .: Вільямс, 2013. - 736 с.
33. Гуленко І. Е. Система відеозахвату і аналізу руху - розпізнавання-трансформацій і руху об'єкту: Праці конференції "Нові інформаційні технології", м Судак (Крим), 15-25 травня, 2004. - С. 141-142.
34. Маклаков С.В., Туманов В.Є., Проектування реляційних сховищ даних, - М .: Діалог-МІФІ, 2007, - 336 с.

35. Оппенхейм А. В., Лім Дж. С. Важливість фази при обробці сигналів // ТІІЕР. - Т. 69 (1981). - № 5. - С. 39-54.
36. Грекул В. І., Денищенко Г. Н., Коровкина Н. Л.- Проектування інформаційних систем: навчальний посібник / 2-е вид., Испр. - М .: Інтернет-Університет інформаційних технологій (ІНТУІТ.РУ): БИНОМ. Лабораторія знань 2010 .З 299.
37. Красильников Н. Н. Принципи обробки зображень, засновані на обліку їх семантичної структури // Інформаційно-керуючі системи. - 2008. - № 1. - С. 2-6.
38. Діго С.М., Бази даних. Проектування і створення: Навчально-методичний комплекс. - М .: Изд. центр ЕАОІ. 2008. - 171 с.
39. Гласман К. Ф., Логунов А. Н. Оцінка помітності артефактів видеокомпрессии // Техніка кіно і телебачення. - 2003. - № 4. - С. 29-32.
40. Красильников Н. Н. Теорія передачі і сприйняття зображень. - М .: Радіо и зв’язок, 1986. - 247 с.
41. Глезер В. Д. Зір і мислення. - Л .: Наука, 1985. - 246 с.
42. Нестерук В.Ф., Порфирьева М.М. Інформаційна оцінка процесу зорового сприйняття // Оптика і спектроскопія. - 1998. - Т.44. вип.4. - с.801-803.
43. Асмаков С. Говорить і показує комп'ютер // Комп'ютер ПРЕС. - 2001. №1. - С.109-117.
44. ГОСТ 34.201-89. Інформаційна технологія. Комплекс стандартів на Автоматизовані системи. Види, комплектність и Позначення документів при створенні автоматизованого систем.
45. Володін А. Б. Адаптивна контекстна компресія зображень // Вісник молодих вчених. - 2002. - № 9. - С. 88 - 96.
46. Гриненко Е. Мобільне телебачення // Інформаційно-технічний журнал "625". - 2009. - № 9. - С. 54-63.
47. Новиков Ф.А., Іванов Д.Ю., Моделювання на UML. Теорія, практика, відеокурс, - СПб .: Професійна література, 2010. - 640 с.

48. Ільясова Н. Ю. Класифікація кристаллограмм з використанням методів статистичного аналізу текстурних зображень / Н. Ю. Ільясова, А. В. Купріянов, А. Г. Храмов // Комп'ютерна оптика. - 2000. - № 20. - С. 122-127.
49. Красильников Н. Н. Принципи обробки зображень, засновані на обліку їх семантичної структури // Інформаційно-керуючі системи. - 2008. - № 1. - С. 2-6.
50. Клімов А. С. Формати графічних файлів. - К .: НДВФ "Диасофт Лтд.", 1995. - 480 с.
51. Конушін А., Баринова О., Конушін В., Якубенко О., Велижа А. Введення в комп'ютерне зір // МГУ ВМК, Graphics & Media Lab. - 2008 (http://courses.graphicon.ru).
52. ГОСТ 24.202-80. Вимоги до змісту документа «Техніко-економічне обґрунтування створення АСУ.
53. Агостон Ж. Теорія кольору і її застосування в мистецтві і дизайні. - М .: Світ, 1982. - 181 с.
54. Айріг С., Айріг Е. Сканування, професійний підхід / Перекл. з англ. - Минск: Попурри, 1997. - 169 с.
55. Yamasaki A, Takauji H, Kaneko S, et al. Denighting: enhancement of nighttime images for a surveillance camera. // In: 19-th international conference on pattern recognition, ICPR. -Tampa, FL, United States, 2008.
56. Lin H., Shi Z. Multi-scale retinex improvement for nighttime image enhancement // Optik - International Journal for Light and Electron Optics. - 2014. - Vol. 125, Issue 24. - P. 7143-7148.
57. Faraji M.R., Qi X. Face recognition under varying illuminations using logarithmic fractal dimension-based complete eight local directional patterns. // Neurocomputing. - 2016. - Vol. 99. - P. 16-30.
58. Гончаров А. В., Каракіщенко А. Н. Вплив освітленості на якість розпізнавання фронтальних осіб. // Известия ПФУ. Технічні науки. - 2008. № 4. - С. 88-92.

59. Поревит В. Н. Комп'ютерна графіка. - СПб .: БХВ-Петербург, 2004. - 432 с.
60. Галямін І.Г., Управління процесами, - СПб .: Питер, 2013. - 304 с.
61. Бєляєв Е. А., Чуйков А. В. Модифікований алгоритм ієрархічної оцінки руху в задачах стиснення відеоінформації: Наукова сесія ГУАП, 2007. Зб. доп. / ГУАП. СПб. - С. 56-60.
62. Dim, Jules R. Takamura, Tamio. Alternative Approach for Satellite Cloud Classification: Edge Gradient Application. // Advances in Meteorology. - 2013. -p.1-8.
63. Будрікіс З. Л. Критерій вірності відтворення зображення і його моделювання // ТІІЕР. - Т. 60 (1972) - № 7. - С. 48-59.
64. Губанов П. В. Автоматична сегментація текстурованих зображень на основі локальних розподілів характеристик // Вісник Томського державного університету. - Т. 271. - 2000, червень. - С. 74-77.
65. Анісімов В.В. Проектування інформаційних систем. Електронний ресурс: https://sites.google.com/site/anisimovkhv/learning/pris.
66. Shashua A., Riklin-Raviv T. The quotient image: Class-based re-rendering and recognition with varying illuminations. // Transactions on Pattern Analysis and Machine Intelligence. 2001. - Vol. 23, No. 2. - P. 129-139.
67. Han, H., Shan, S., Chen, X., Gao, W. A comparative study on illumination preprocessing in face recognition // Pattern Recogn.- 2013. - 46 (6). - P. 1691-1699.
68. Красильников Н. М. Математична модель темнової адаптації в зоровій системі людини // Оптичний журнал. - Т. 64 (1997). - № 11. - С. 38-44.
69. Дейт К. Дж., Введення в системи баз даних, 8-е видання .: Пер. з англ. - М .: Видавничий дім "Вільямс", 2005. - +1328 с .: іл. - Парал. тит. англ.
70. Хрящів Денис Олександрович. Підвищення якості зображень, отриманих в умовах недостатньої освітленості // ИВД, 2013. - № 3 (26).

71. Wang W., Li J., Huang F., Feng H. Design and Implementation of Log-Gabor Filter in Fingerprint Image Enhancement // Pattern Recognition Letters. - 2008. - vol. 29, no. 3. - pp. 301-308.
72. Гонсалес Р., Вудс Р. Цифрова обробка зображень / Перекл. з англ. - М .: Техносфера, 2006. - 1070 с.
73. Красильников Н. Н. Узагальнена функціональна модель зору і її застосування в системах обробки і передачі зображень // Автометрія. - 1990. - № 6. - С. 7-14.
74. R. BrunelliandT. Poggio. Hyper BFNetworks for Real Object Recognition // Proc. IJCAI. - Sydney, Australia, 1999.. - p. 1278-1284.
75. Фісенко В.Т., Фісенко Т.Ю. Комп'ютерна обробка і розпізнавання зображень. - СПб .: СпбГУ ИТМО, 2008. - 192с.
76. Гурський Ю., Корабельникова Г. Photoshop 7. Трюки і ефекти. - М .: Москва, 2005. - 464 с.