Київський національний торговельно-економічний університет
Кафедра інженерії програмного забезпечення та кібербезпеки

ВИПУСКНИЙ КВАЛІФІКАЦІЙНИЙ ПРОЕКТ

на тему:

«Відновлення зображень з використанням архітектурного шаблону проектування MVC»

Студента 2м курсу, 6 групи, спеціальності 121 «Інженерія програмного забезпечення» спеціалізації «Інженерія програмного забезпечення» _____ підпис студента — Коломієць Іван Олегович

Науковий керівник доктор технічних наук, професор кафедри інженерії програмного забезпечення та кібербезпеки _____ підпис керівника — Криворучко Олена Володимирівна

Гарант освітньої програми доктор технічних наук, професор кафедри інженерії програмного забезпечення та кібербезпеки _____ підпис гаранта — Криворучко Олена Володимирівна

КИЇВ - 2020

Київський національний торговельно-економічний університет

Факультет інформаційних технологій
Кафедра інженерії програмного забезпечення та кібербезпеки
Освітній ступінь магістр
Спеціальність 121 «Інженерія програмного забезпечення»

Затверджую
Зав. кафедри інженерії програмного забезпечення та кібербезпеки
Криворучко О. В.
8 жовтня 2019 р.

Завдання
на випускний кваліфікаційний проект студентові

Коломієць Іван Олегович
(прізвище, ім'я, по батькові)

1. Тема випускного кваліфікаційного проекту «Відновлення зображень з використанням архітектурного шаблону проектування MVC»
Затверджена наказом ректора від "13" грудня 2019 р. № 4304
2. Строк здачі студентом закінченої проекту 01 грудня 2020

3. Цільова установка та вихідні дані до проекту
Метою даної дипломної роботи є розробка мобільного додатку для відновлення зображень з використанням ахрітектурного шаблону MVC.
Об'єкт дослідження Методи розробки програмного забезпечення за допомогою MVC
Предмет дослідження процесс створення мобільного додатку для відновленя зображень за допомогою архітектурного шаблону MVC.

4. Консультанти проекту із зазначенням розділів, які консультують:

Розділ	Консультант (прізвище, ініціали)	Підпис, дата	
		Завдання видав	Завдання прийняв

5. Зміст випускного кваліфікаційного проекту (перелік питань за кожним розділом)

ВСТУП

РОЗДІЛ 1. АНАЛІЗ ІСНУЮЧИХ ПІДХОДІВ РОЗВ'ЯЗАННЯ ЗАВДАНЬ ПОЛІПШЕННЯ ЦИФРОВОГО ЗОБРАЖЕННЯ ТА ВІДНОВЛЕННЯ ЙОГО СТРУКТУРИ

1.1 Опис методу обробки зображень у частотній області

1.2 Алгоритм фільтрації зображення в частотній області

1.3 Нелінійні методи фільтрації

1.4. Висновок до розділу 1

РОЗДІЛ 2. ПРОБЛЕМА КЛАСИФІКАЦІЇ ГРАФІЧНИХ ОБРАЗІВ

2.1 Рецепторна структура сприйняття інформації

2.2 Алгоритмічні побудови

2.3 Персептрон, як модель розпізнавання

РОЗДІЛ 3. АРХІТЕКТУРНИЙ ШАБЛОН MVC

3.1 Структура організації програмного коду та архітектура MVC

3.2 Основний принцип роботи MVC

3.3 Конвенція по конфігурації та основні переваги та недоліки MVC

3.4. Висновок до розділу 3

РОЗДІЛ 4 РОЗРОБКА МОБІЛЬНОГО ДОДАТКУ WESCAN ТА ОГЛЯД ОПЕРАЦІЙНОЇ СИСТЕМИ IOS

4.1 мобільні додатки та операційна система iOS

4.2 Специфікації та особливостиі мови програмування Swift

4.3 Архітектура мобільної операційної системи iOS

4.4 Опис основних інструментів розробки мобільного додатку для платформи iOS

4.5 Проектування мобільного додатку WeScan

4.5.1 Використання CoreData

4.5.2 Використання Segue

4.5.3 Використання CloudKit
4.6 Висновки до розділу 3
ВИСНОВКИ ТА ПРОПОЗИЦІЇ
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ
ДОДАТКИ

6. Календарний план виконання проекту

№ пор.	Назва етапів випускного кваліфікаційного проекту	Строк виконання етапів проекту	
		за планом	фактично
1	2	3	4
1.	*Вибір теми випускної кваліфікаційної роботи*	*20.09.2019*	*20.09.2019*
2.	*Розробка та затвердження завдання на проект магістра (Наказ №4304 від 13.12.19)*	*08.11.2019*	*08.11.2019*
3.	*Вступ та перелік літературних джерел*	*24.01.2020*	*24.01.2020*
4.	*Наукова стаття*	*01.09.2020*	*01.09.2020*
5.	*Технічне завдання*	*28.02.2020*	*28.02.2020*
6.	*Розділ 1. Аналіз предметної області застосування мобільного додатку*	*25.06.2020*	*25.06.2020*
7.	*Розділ 2. Вибір програмних засобів та проектування мобільного додатку*	*07.09.2020*	*07.09.2020*
8.	*Розділ 3. Реалізація мобільного додатку*	*19.10.2020*	*19.10.2020*
9.	*Програма та методика тестування*	*21.10.2020*	*21.10.2020*
10.	*Керівництво користувача*	*23.10.2020*	*23.10.2020*
11.	*Висновки та пропозиції*	*30.10.2020*	*30.10.2020*
12.	*Здача випускного кваліфікаційного проекту на кафедру (перша перевірка)*	*05.11.2020*	*05.11.2020*
13.	*Підготовка автореферату та презентації доповіді*	*05.11.2020*	*05.11.2020*
14.	*Попередній захист випускного кваліфікаційного проекту*	*25.11.2020-27.11.2020*	*25.11.2020-27.11.2020*
15.	*Зовнішнє рецензування випускної кваліфікаційної роботи*	*01.12.2020*	*01.12.2020*
16.	*Підготовка до публічного захисту випускної кваліфікаційної роботи*	*10.12.2020-11.12.2020*	*08.12.2020*

7. Дата видачі завдання «8» листопада 2019 р.

8. Науковий керівник випускного кваліфікаційного проекту Криворучко О.В

(прізвище, ініціали, підпис)

9. Гарант освітньої програми Криворучко О.В.

(прізвище, ініціали, підпис)

10. Завдання прийняв до виконання студент Коломієць І.О.

(прізвище, ініціали, підпис)

11. Відгук керівника випускного кваліфікаційного проекту

_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____

Науковий керівник випускного кваліфікаційного проекту

_____

(підпис, дата)

Відмітка про попередній захист _____

(ПІБ, підпис, дата)

12. Висновок про випускний кваліфікаційний проект

Випускний кваліфікаційний проект студента _____

(прізвище, ініціали)

може бути допущена до захисту екзаменаційній комісії.

Гарант освітньої програми _____ Криворучко О. В.

(прізвище, ініціали, підпис)

Завідувач кафедри _____ Криворучко О. В.

(підпис, прізвище, ініціали)

«_____»_____20 ___ р.

АНОТАЦІЯ

Відповідно до мети дослідження робота присвячена розробці мобільного додатка для відновлення зображень з використанням ахрітектурного шаблону MVC, що дозволить відновлювати та сканувати зображеня для сканування та відновлення в електронному вигляді задопомогою нейронних мереж.

В результаті вибору технології було розглянуто рішення з використанням методу обробки зображень у частотній області з використанням нелінійних методів фільтрації.

Розробка додатку була реалізована задопомогою мови програмування Swift та програмного середовища xCode для мобільної платформи iOS. З використанням архітектурного шаблону MVC який дозволяє використовувати вибрані технології як шарування завдяки чому можлива модернізація та вдосконалення додатку без великих змін вже працюючого функціоналу.

Ключові слова : MVC, мобільний додаток, нейронна мережа, алгоритми.

ANOTATION

As soon as the robot was updated, a mobile extension was assigned for updating the image from the archives of the architectural template MVC, so that you can view the scanned image for scanning in the neurally updated image.

As a result of the selection of technology, a decision was made to the victorious method of image processing in the frequency area from the victorious methods of filtering.

The distribution of the add-on was realized with the help of the Swift mobile software and the xCode software middleware for iOS mobile platforms. By virtue of the architectural template MVC, it is permissible to vikoristovuvati vibrani technologies, as well as to what it is possible to modernize and complement without great changes in the same functional functionality.

Key words: MVC, mobile add-on, neural netting, algorithms.

ВСТУП

Використання цифрових зображень почалося ще на початку 1920-х років. З появою комп'ютерів у середині XX століття виникла нова область науки – цифрова обробка зображень. З'явилися перші задачі з обробки цифрових зображень: стиснення зображень, покращення якості, сегментація, морфологічний аналіз та багато інших. Характерною особливістю багатьох методів цифрової обробки зображень є проблема формалізації перетворень.

Спроби вирішувати дані задачі детерміністичними методами часто накладало різні обмеження, наприклад, складність формального представлення перетворень, наявність великої кількості змінних та відсутність еталонного розв'язку. Саме тому в процесі розвитку методів обробки зображень почали з'являтися методи, основою яких є нові концепції пошуку рішень в обчислювальних системах.

За останні роки роль сучасних технологій в житті люди дуже змінилась. Швидко зростаючий ринок мобільних додатків продовжує завойовувати найрізноманітніші сфери нашого життя. Люди активно використовують мобільні девайси в повсякденному житті, що, безсумнівно, збільшує попит на самі різні додатки для мобільних телефонів. Саме цим пояснюється той факт, що більшість стартаперів вибирають розробку мобільних додатків в якості відправної точки для свого бізнесу. Мобільні пристрої – це та технологія, яку люди весь час тримають під рукою, так як з їх допомогою можна вкрай швидко отримати достовірні відомості.

Актуальність даної теми обумовлена тим що останім часом робота з друкованим матеріалом відходить на задній план тому що робота з цифромим метаріалами стала невідьємною складовою буденності людини. Тому змога

					КНТЕУ 121 06 -11МР			
Зм.	*Аркуш*	*№ докум.*	*Підпис*	*Дата*				
Зав. каф.		Криворучко О.В.		24.01.20	Відновлення зображень з використанням архітектурного шаблону проектування MVC	*Стадія*	*Аркуш*	*Аркушів*
Керівник		Криворучко О.В.		24.01.20		*В*	*3*	*50*
Гарант		Криворучко О.В.		24.01.20		Факультет інформаційних технологій 2м курс, 6 група		
Розробив		Коломієць І.О.		24.01.20	*Вступ*			

оцифрувати той чи інший метеріал є дуже вагомою змогою покращити будення та дати змогу мати все що потрібно під рукою у мобільному пристрої. Тому було обрано саме мобільну платформу як відправну точку для розробок з використанням нейронних мереж для вдосконалення та покращення роботи алгоритмів роботи функціоналу додатку. Використовуючи архітектурний шаблон MVC розробка мобільного додатку стає більш простою та гнучкою для вровадження нового функціоналу в подальшому та дає змогу розробляти повністю новий функціонал незаважаючи та не змінюючи вже існуючий. Всі мобільні додатки умовно можна поділити на програми для робочих цілей і на розважальні програми. Перші дозволяють бізнесменам і офісним працівникам контролювати бізнес-процеси, складати аналітичну звітність, виконувати такі завдання, як розробка дизайну фірмового стилю.

Протягом останніх років показник, що характеризує рівень попиту на мобільні пристрої, постійно зростає. Така статистика дозволяє зробити висновок про те, що розробка мобільних додатків актуальна і доцільна. Отже, тільки корисна розробка отримає гідне визнання з боку користувачів.

Об'єктом дослідження є: Методи розробки програмного забезпечення за допомогою MVC

Предмет дослідження: процесс створення мобільного додатку для відновленя зображень за допомогою архітектурного шаблону MVC.

Метою даної дипломної роботи є розробка мобільного додатку для відновлення зображень з використанням ахрітектурного шаблону MVC.

Методи дослідження, що були використані у роботі: аналіз, абстрагування, порівняння, графічний метод і моделювання.

Наукова новизна дослідження полягає в удосконаленій концепції мобільного додатка, що дозволяє сканувати та відновлювати графічні зображення в цифровому вигляді.

Практичне значення дослідження: готове рішення у вигляді iOS-додатка, який дозволяє сканувати та відновлювати зображення з аналогового виду у цифровий та зберігати їх для подальшого використання.

5) провести тестування готового рішення на реальних пристроях та розробити інструкцію користувача.

Методи дослідження, що були використані у роботі: аналіз, абстрагування, порівняння, графічний метод і моделювання.

Наукова новизна дослідження полягає в удосконаленій концепції мобільного додатка, що імітує комп'ютерні маніпулятори, порівняно з аналогічними рішеннями.

*Практичне значення дослідження***:** готове альтернативне рішення у вигляді Android-додатка, що здатний замінити комп'ютерну мишку за потреби, та доступний широкому загалу користувачів ПК.

РОЗДІЛ 1.
АНАЛІЗ ІСНУЮЧИХ ПІДХОДІВ РОЗВ'ЯЗАННЯ ЗАВДАНЬ ПОЛІПШЕННЯ ЦИФРОВОГО ЗОБРАЖЕННЯ ТА ВІДНОВЛЕННЯ ЙОГО СТРУКТУРИ

1.1 Опис методу обробки зображень у частотній області

Існуючі підходи щодо розв'язання завдань поліпшення цифрового зображення та відновлення його структури поділяють на дві категорії:

1) методи обробки в просторовій області (просторові методи), які ґрунтуються на прямому маніпулюванні пікселями зображення;
2) методи обробки в частотній області (частотні методи), які ґрунтуються на модифікації (фільтрації) сигналу, що формується шляхом застосування до зображення перетворення Фур'є.

Просторова обробка застосовується, коли єдиним джерелом викривлень є адитивний шум. Частотна фільтрація може використовуватися для нечітких зображень з дефектами освітлення, також вона враховує й шум. Тому частотна обробка є найбільш універсальним і поширеним методом поліпшення якості цифрового зображення.

Суть цього методу полягає в представленні зображення як двовимірної функції f(x, y), де x та y – координати в просторі (конкретно, на площині.

Значення f у будь-якій точці, що задається парою координат (x, y), називається інтенсивністю, або рівнем сірого в цій точці.

Загальновідомим є твердження, що будь-яка функція, яка періодично повторює свої значення, може бути представлена у вигляді суми синусів та косинусів різних частот, помножених на деякі коефіцієнти. Таке представлення функції називається представленням у вигляді ряду Фур'є. Коли функція не є періодичною, але площа під її графіком є кінцевою, це – перетворення Фур'є.

Функція, задана як рядом, так і перетворенням Фур'є, може бути повністю, без втрати інформації, відновлена за допомогою алгоритму перетворення. Ця властивість є надзвичайно важливою, оскільки дозволяє працювати у «Фур'є-просторі», а потім повернутися в початкову область визначення функції без втрати якої-небудь інформації. На рис. 1 а зображено складну функцію, яка є сумою чотирьох синусоїд та косинусоїд рис. 1 б.

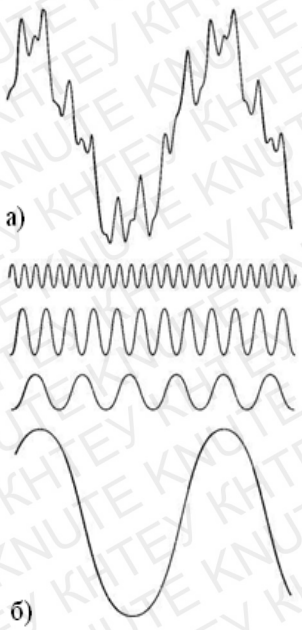


Рис. 1.1 Розкладання функції на складники

Оскільки цифрові зображення описують двовимірними дискретними функціями, то розглянемо дискретне перетворення Фур'є (ДПФ) саме для таких функцій.

Нехай f(x, y), при x = 0, 1, 2..., M - 1 і y = 0, 1, 2..., N - 1, позначає зображення M×N. Двовимірне дискретне перетворення Фур'є зображення f(x, y), яке позначається F(u, ν), задається рівнянням

$$F(u,\nu) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) e^{-j2\pi(ux/M+\upsilon y/N)}, \tag{1.1}$$

де u=0,1,2..., M - 1 і ν=0,1,2..., N - 1; M і N – парні числа. Координатна система, задаючи аргументи F(u, ν) частотними змінними u і ν, називається частотною областю. У цьому випадку можна виявити аналогію із завданням аргументів f(x, y) просторовими змінними x і y. Прямокутну область розміру M×N, при u=0, 1, 2..., M - 1 і ν=0, 1, 2..., N - 1, прийнято називати частотним прямокутником. Він має ті ж розміри, що й початкове зображення.

Навіть якщо зображення *f(x, y)* дійсне, його перетворення Фур'є є, як правило, комплексним. Основний метод візуального аналізу цього перетворення полягає в обчисленні його спектру (тобто абсолютної величини *F(u, v))* і його відображення на дисплеї. Нехай *R(u, v)* і *I(u, v)* позначають дійсну й уявну компоненти *F(u, v)*, тоді спектр Фур'є задається виразом 1.2.

$$\left| F(u,\nu) \right| = \left[R^2(u,\nu) + I^2(u,\nu) \right]^{1/2}. \tag{1.2}$$

Кожен елемент фур'є-образу *F(u, v)* містить усі відліки функції *f(x, y)*, помножені на значення експоненціальних членів, тому зазвичай неможливо встановити пряму відповідність між характерними деталями зображення і його образом. Проте можна зробити деякі спільні висновки щодо

взаємозв'язку частотних складників фур'є-образу і просторових характеристик зображення. Наприклад, оскільки частота прямо пов'язана із швидкістю зміни сигналу, то зрозуміло, що частоти в перетвореннях Фур'є пов'язані з варіацією яскравості на зображенні. Найбільш повільно змінюваний (постійний) частотний складник *(u=v=0)* збігається з середньою яскравістю зображення.

Низькі частоти, що відповідають точкам поблизу початку координат фур'є-перетворення, відповідають повільно змінним компонентам зображення. На зображенні кімнати, наприклад, вони можуть відповідати плавним змінам яскравості стін і підлоги. Із віддаленням від початку координат вищі частоти починають відповідати все більшим змінам яскравості деталей зображення та їх меж.

1.2 Алгоритм фільтрації зображення в частотній області

Процедура алгоритму фільтрації в частотній області проста і складається з таких кроків:

1. Початкове зображення множиться на $(-1)^{x+y}$, це робиться для того, щоб його перетворення Фур'є виявилося центрованим, тобто початок координат для образу функції буде в центрі частотного прямокутника в точці (M/2; N/2);

$$\xi\left[f(x,y)(-1)^{x+y}\right] = F(u - M/2, v - N/2). \tag{1.3}$$

2. Обчислюється пряме ДПФ F(u, v) зображення, отриманого після кроку
3. Функція F(u, v) множиться на деяку функцію фільтру H(u, v);

4. Обчислюється зворотне ДПФ від результату кроку 3;
5. Виділяється потрібна частка результату кроку 4;
6. . Результат кроку 5 множиться на (- 1) x +y .

Причина, через яку множник *H(u, v)* називається фільтром (часто використовують також термін передаточна функція фільтра), полягає в тому, що він пригнічує деякі «зайві» частоти перетворення, залишаючи при цьому інші майже без зміни. Питання знаходження передаточної функції фільтра і є ключовим, адже воно визначає метод фільтрації і вказує, які саме частоти будуть відфільтровуватися.

Нехай f(x, y) позначає вхідне зображення після кроку 1, а F(u, ν) є його фур'є-образом. Тоді фур'є-образ вихідного зображення визначається виразом

$$G(u,\nu) = H(u,\nu) \cdot F(u,\nu) . \qquad (1.4)$$

Множення функцій двох змінних H і F здійснюють поелементно. Фільтроване зображення отримують обчисленням зворотного перетворення Фур'є від фур'є-образу $F(u,\nu)$, обчислюючи за формулою.

Покращене зображення = $\xi^{-1}[G(u,\nu)]$.

Знайдене зображення отримуємо, виділивши дійсну частину з останнього результату і множення на (-1) x+y , щоб компенсувати ефект від множення вхідного зображення на ту ж величину . Описано процедуру алгоритму фільтрації схематично зображена на рис. 2 в більш загальному вигляді, який містить стадії попередньої й завершальної обробки

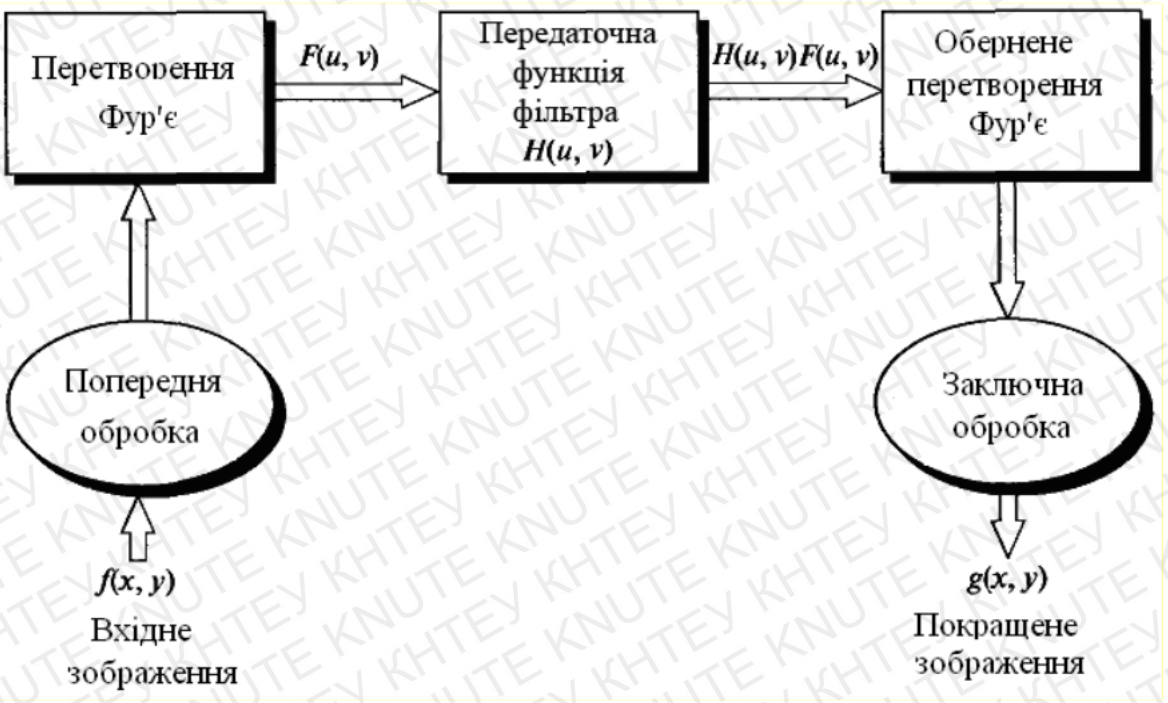
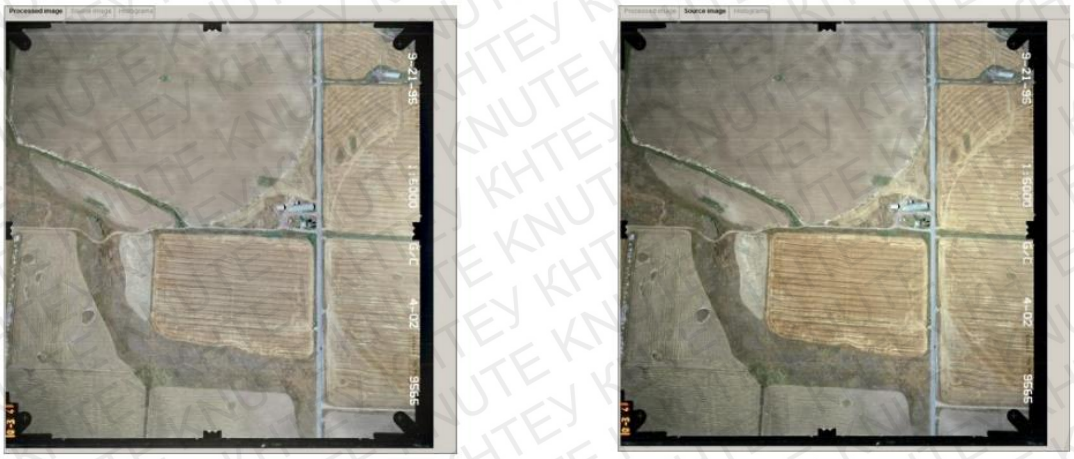


Рис. 1.2. Основні етапи фільтрації в частотній області

Ця схема фільтрації може мати деякі зміни, пов'язані з необхідністю зменшення вхідного зображення, масштабування яскравості і тощо.Прикладом фільтрації в частотному діапазоні є обробка аерокосмічних зображень для геоінформаційних систем та людини-оператора. Результати фільтрації зображення в частотному діапазоні наведено на рис. 1.3, де у відфільтрованому зображенні зменшується початкове освітлення.Недоліком і предметом досліджень усіх методів фільтрації в частотній області є неможливість створення ідеального фільтра, що відкидав би всі «зайві» частоти, відновлюючи при цьому якість зображення на рисунку 1.3 а) фільтроване зображення; б) оригінальне зображення.



а) б)

Рис. 1.3. Результати фільтрації в частотному діапазоні

1.3 Нелінійні методи фільтрації

Нелінійні методи фільтрації належать до одного із видів методів обробки зображень в частотній області. Клас нелінійних цифрових фільтрів є дуже широким для того, щоб проводити їхній опис у загальному вигляді. Розглянемо одні з найвідоміших методів із родини нелінійних цифрових фільтрів.

Під час фільтрації реальних зображень обмеженого розміру виникає гранична проблема одержання оцінок у точках нульового рядка й нульового стовпчика. Природним рішенням є використання звичайної (одномірної) калмановської фільтрації.Уолкап і Чоенс запропонували використовувати вінерівську фільтрацію для боротьби із шумом зернистості фотоплівки в моделі системи зображення, що описує формула де α – постійна величина.

$$\tilde{y}_{i,j} = y_{i,j} + \alpha\left[y_{i,j}\right]^{1/3} n_{i,j},$$

(1.4)

Для цієї моделі було отримано частотну характеристику реставруючого фільтра, що відповідає випадку нескінченного зображення, яке описує рівність

$$\mathrm{H}_R(\omega_x,\omega_y)=\frac{W_{F_1}(\omega_x,\omega_y)}{W_{F_1}(\omega_x,\omega_y)+\alpha^2 E\left\{\left[F_1(\omega_x,\omega_y)\right]^{2/3}\right\}}, \quad (1.5)$$

Де $W_{F1}(\pi_x,\pi_y)$ – енергетичний спектр ідеального зображення, Е – позначення математичного прогнозу.

Цвейг розробив евристичний нелінійний метод реставрації малоконтрастних зображень з метою послаблення шуму зернистості фотоплівки. Під час використання цього методу вхідне зображення розгортається з високою роздільною здатністю, а кожний його елемент квантується великим числом рівнів. Потім одержують зображення зниженої чіткості, об'єднуючи елементи у фрагменти, що не перетинаються, розміром 2×2. Звичайно, чітке зображення має більш різкі межі, ніж зображення зі зниженою чіткістю, проте дисперсія шуму останнього виявляється меншою. У випадку білого шуму дисперсія нечіткого зображення в чотири рази менша, ніж для чіткого зображення, що є наслідком просторового усереднення елементів. Усереднене зображення повторно квантується з використанням рівномірної шкали, причому крок квантування вибирається таким, що дорівнює значенню средньоквадратичного відхилення шуму, збільшеному в чотири рази. Завдяки такому вибору, забезпечується помилка квантування 5% при гаусовому шумі. Отримані квантовані елементи нечіткого зображення досліджують в області розміром 3×3 елементи [5].

Якщо центральний елементи нечіткого зображення лежить на межі (рис. 4), він розділяється на чотири елемента, що відповідають повній роздільній

здатності; цим новим елементам приписуються рівні, що залежать як від рівнів, що відповідають вихідним елементам чіткого зображення, так і від рівнів найближчих елементів нечіткого зображення.

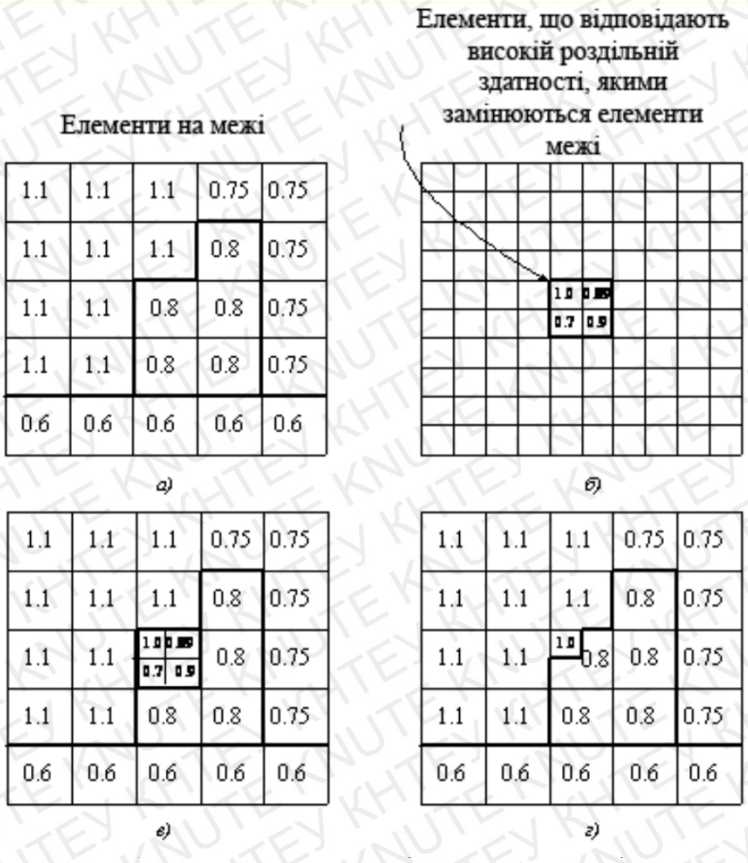


Рис. 1.4. Приклад алгоритму пригнічення шуму за Надері:

Де а) масив, що відповідає зниженій роздільній здатності; б) масив, що відповідає високій роздільній здатності; в) заміна елемента, що належить до межі; г) остаточний результат

Може виявитися, що всі вісім периферійних елементів проквантовано з одним рівнем, а центральний елемент – з іншим рівнем. У цьому випадку вважають, що ізольований центральний елемент містить помилку, обумовлену шумом, і приписують йому середній рівень периферійних елементів. Простий

алгоритм полягає в тому, що елементу, який відповідає високій роздільній здатності, приписують рівень одного з чотирьох пов'язаних елементів (елементи «північ» і «схід» або «північ» і «захід» і т. д.), найближчий до рівня знайденого елемента.

1.4 Висновки до розділу 1

Розглянуто основні існуючі підходи щодо розв’язання завдання поліпшення цифрового зображення та відновлення йоґо структури. Проаналізовано метод обробки зображення в частотній області та йоґо математичну модель. Запропоновано алгоритм фільтрації в частотній області та зображено по - крокову схему йоґо роботи для покращення якості зображення. Наведено один із видів методів обробки зображень у частотній області – нелінійну фільтрацію. Розглянуто одні із найвідоміших методів нелінійної фільтрації для усунення завад та поліпшення оригінального зображення. Нелінійні фільтри можуть використовуватися для вирішення таких проблем, як усунення завад, шуму, відновлення пошкоджених зображень, поліпшення контрасту та виділення контурів зображення тощо.

РОЗДІЛ 2.
ПРОБЛЕМА КЛАСИФІКАЦІЇ ГРАФІЧНИХ ОБРАЗІВ

2.1 Рецепторна структура сприйняття інформації

Для того, щоб людина свідомо сприйняв інформацію, вона повинна пройти досить тривалий цикл попередньої обробки. Спочатку світло потрапляє в око. Пройшовши через всю оптичну систему фотони, врешті-решт, потрапляють на сітківку - шар світлочутливих клітин - паличок і колбочок.Вже тут - ще дуже далеко від головного мозку, відбувається перший етап обробки інформації, оскільки, наприклад, у ссавців, відразу за світлочутливими клітинами знаходиться зазвичай два шари нервових клітин, які виконують порівняно нескладну обробку.

Тепер інформація надходить по зоровому нерву в головний мозок людини, в так звані "зорові горби", на які проектується зорова інформація то, що ми бачимо. Далі зорова інформація надходить до відділів мозку, які вже виділяють з неї окремі складові - горизонтальні, вертикальні, діагональні лінії, контури, області світлого, темного, кольорового. Поступово образи стають все більш складними і розмитими, але графічний образ картини пройде ще довгий шлях, перш ніж досягне рівня свідомості. До проблеми розпізнавання образів можна підходити, відштовхуючись від аналогії з біологічними процесами. У деяких умовах здатності тварин до розпізнавання образів перевищують здатності будь-якої машини, яку тільки можна побудувати. При класифікації, заснованої на безпосередньому сенсорному досвіді, тобто при розпізнаванні осіб або вимовлених слів, люди легко перевершують технічні пристрої. В "несенсорним ситуаціях" дії людей не настільки ефективні.

Наприклад, люди не можуть змагатися з програмами класифікації образів, якщо правильний спосіб класифікації включає логічні комбінації абстрактних властивостей, таких, як колір, розмір і форма. Оскільки розпізнавання образів має бути функцією нейронів тваринного, можна шукати ключ до біологічного розпізнавання образів у властивостях самого нейрона. Для багатьох цілей нейрон можна розглядати як граничний елемент. Це означає, що він або дає на виході деяку постійну величину, якщо сума його входів досягає певного значення, або ж залишається пасивним.

Розпізнавання образів - це завдання ідентифікації об'єкта або визначення яких-небудь його властивостей по його зображенню (оптичне розпізнавання) або аудіозаписи (акустичне розпізнавання). В процесі біологічної еволюції багато тварин за допомогою зорового і слухового апарату вирішили це завдання досить добре. Створення штучних систем з функціями розпізнавання образів залишається складною технічною проблемою.

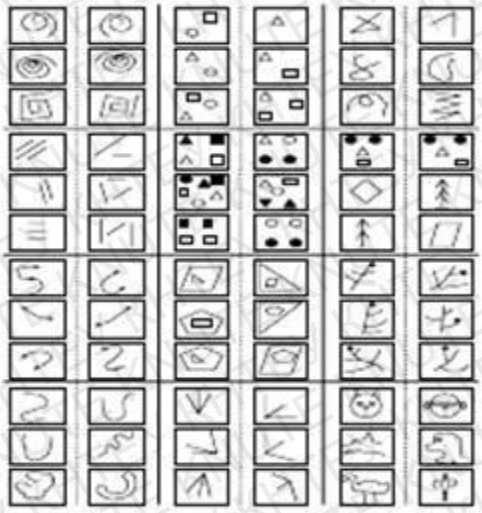


Рис. 2.1 – Приклад об’єктів навчання

В цілому проблема розпізнавання образів (ПРО) складається з двох частин: навчання та розпізнавання. Навчання здійснюється шляхом показу окремих об'єктів із зазначенням їх приналежності того чи іншого способу.

В результаті навчання розпізнає система повинна придбати здатність реагувати однаковими реакціями на всі об'єкти одного образу та іншими реакціями - на всі об'єкти відмітних образів. Дуже важливо, що процес навчання повинен завершитися тільки шляхом показів кінцевого числа об'єктів. Як об'єкти навчання можуть бути або картинки (рис. 2.1), або інші візуальні зображення (літери, цифри) [1]. Важливо, що в процесі навчання вказуються тільки самі об'єкти і їх приналежність образу. За навчанням слід процес розпізнавання. нових об'єктів, який характеризує дії вже навченої системи. Автоматизація цих процедур і становить проблему навчання розпізнаванню образів. У тому випадку, коли людина сама розгадує або придумує, а потім нав'язує машині правило класифікації, проблема розпізнавання вирішується частково, так як основну і головну частину проблеми (навчання) людина бере на себе.

Коло завдань, які можуть вирішуватися за допомогою розпізнають систем, надзвичайно широкий. Сюди відносяться не тільки задачі розпізнавання зорових і слухових образів, а й завдання класифікації складних процесів і явищ, що виникають, наприклад, при виборі доцільних дій керівником підприємства або виборі оптимального управління технологічними, економічними, транспортними або військовими завданнями. Перш ніж почати аналіз будь- якого об'єкта, потрібно отримати про нього певну, впорядковану інформацію.

Вибір вихідного опису об'єктів є однією з центральних завдань проблеми розпізнавання образів. При вдалому виборі вихідного опису (простору ознак) завдання розпізнавання може виявитися тривіальної і, навпаки, невдало вибране початкове опис може привести або до дуже складної подальшої переробку інформації, або взагалі до відсутності рішення. Якщо стверджується, що при показі зображень можливо однозначно віднести їх до одного з двох (або декількох) образів, то тим самим стверджується, що в

деякому просторі існує дві (або декілька) області, не мають спільних точок, і що зображення - точки з цих областей. Кожній такій області можна приписати найменування, т. Е. Дати назву, яка відповідає образу. Проінтерпретіруем тепер в термінах геометричної картини процес адаптації розпізнавання образів, обмеживши поки випадком розпізнавання тільки двох образів. Заздалегідь вважається відомим лише тільки те, що потрібно розділити дві області в деякому просторі, і що показуються точки тільки з цих областей. Самі ці області заздалегідь не визначені, т. Е. Немає будь-яких відомостей про розташування їх меж чи правил визначення приналежності точки до тієї чи іншої області.

В ході навчання пред'являються точки, випадково вибрані з цих областей, і повідомляється інформація про те, до якої області належать пред'являються точки. Ніякої додаткової інформації про ці областях, т. п. Про розташування їх меж, в ході навчання не повідомляється. Головна мета навчання полягає або в побудові поверхні, яка розділяла б не тільки показані в процесі навчання точки, але і всі інші точки, що належать цим областям, або в побудові поверхонь, що обмежують ці області так, щоб в кожній з них знаходилися тільки точки одного образу. Головна мета навчання полягає в побудові таких функцій від векторів- зображень, які були б, наприклад, позитивні на всіх точках одного і негативні на всіх точках іншого способу. У зв'язку з тим, що області не мають спільних точок, завжди існує ціла безліч таких поділяють функцій, а в результаті навчання повинна бути побудована одна з них.

На перший погляд здається, що знання всього лише деякої кількості точок з області недостатньо, щоб відокремити всю область. Дійсно, можна вказатинезліченна кількість різних областей, які містять ці точки, і як би не була побудована за ним поверхню, що виділяє область, завжди можна вказати іншу область, яка перетинає поверхню і в той же час містить показані точки.

Поряд з геометричною інтерпретацією проблеми адаптації розпізнавання образів існує й інший підхід, який названий структурним, або лінгвістичним. Пояснимо лінгвістичний підхід на прикладі розпізнавання зорових зображень. Спочатку виділяється набір вихідних понять - типових фрагментів, що зустрічаються на зображеннях, і характеристик взаємного розташування фрагментів - "зліва", "знизу", "всередині" і т. д. Ці вихідні поняття утворюють словник, що дозволяє будувати різні логічні висловлювання. Завдання полягає в тому, щоб з великої кількості висловлювань, які могли б бути побудовані з використанням цих понять, відібрати найбільш істотні для даного конкретного випадку.

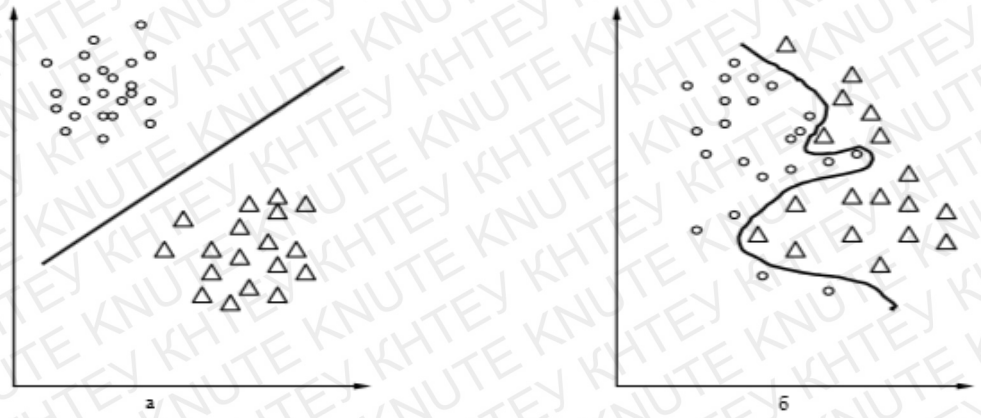


Рис. 2.2 – Розподіл двох образів у просторі

Далі, переглядаючи кінцеве і по можливості невелике число об'єктів з кожного образу, потрібно побудувати опис цих образів. Побудовані опису повинні бути настільки повними, щоб вирішити питання про те, до якого образа належить даний об'єкт. При реалізації лінгвістичного підходу виникають два завдання: завдання побудови вихідного словника, т.п. Набір типових фрагментів, і завдання побудови правил опису з елементів заданого словника.

Якщо припустити, що в процесі навчання простір ознак формується виходячи із задуманої класифікації, то завдання в просторі ознак саме по собі задає властивість, під дією якого образи в цьому просторі легко розділяються. Гіпотеза компактності говорить: компактним образам відповідають компактні безлічі в просторі ознак. Під компактним безліччю розуміються якісь "згустки" точок в просторі зображень, припускаючи, що між цими згустками існують розділяють їх розрядження. Цю гіпотезу не завжди вдавалося підтвердити експериментально, однак ті завдання, в рамках яких гіпотеза компактності добре виконувалася (рис. 2.2.а), знаходилося просте рішення. І навпаки, ті завдання, для яких гіпотеза не підтверджувалася (рис. 2.2б), абозовсім не вирішувалися, або вирішувалися з великими труднощами. Цей факт змусив, щонайменше, засумніватися в справедливості гіпотези компактності, так як для спростування будь-гіпотези досить одного заперечує її прикладу. Разом з цим, виконання гіпотези всюди там, де вдавалося добре вирішити завдання навчання розпізнаванню образів, зберігало до цієї гіпотези інтерес. Сама гіпотеза компактності перетворилася в ознаку можливості задовільного вирішення завдань розпізнавання.

Навчання - це процес, в результаті якого система поступово набуває здатність відповідати потрібними реакціями на певні сукупності зовнішніх впливів, а адаптація - це підстроювання параметрів і структури системи з метою досягнення необхідної якості управління в умовах безперервних змін зовнішніх умов. Всі картинки, представлені на рис. 2.1, характеризують завдання навчання. У кожній з цих завдань задається кілька прикладів (навчальна послідовність) правильно вирішених завдань. Якби вдалося помітити якесь загальне властивість, яке залежить ні від природи образів, ні від їх зображень, а визначальне лише їх здатність до разделимости, то поряд зі звичайною завданням адаптації розпізнавання з використанням інформації про приналежність кожного об'єкта з навчальної послідовності того чи іншого

образу, можна було б поставити іншу класифікаційну завдання - так звану задачу навчання без учителя Завдання такого роду на описовому рівні можна сформулювати наступним чином: системі одночасно або послідовно пред'являються об'єкти без будь-яких вказівок про їх належність до образам.

Вхідний пристрій системи відображає безліч об'єктів на безліч зображень і, використовуючи деякий закладене в неї заздалегідь властивість разделимости образів, виробляє самостійну класифікацію цих об'єктів. Після такого процесу самонавчання система повинна придбати здатність до розпізнавання не тільки вже знайомих об'єктів (об'єктів з навчальної послідовності), але і тих, які раніше не пред'являлися. Процесом самонавчання деякої системи називається такий процес, в результаті якого ця система без підказки вчителя набуває здатності до вироблення однакових реакцій на зображення об'єктів одного і того ж образу і різних реакцій на зображення різних образів . Роль вчителя при цьому складається лише в підказці системі деякого об'єктивного властивості, однакового для всіх образів і визначає здатність до поділу безлічі об'єктів на образи. Таким об'єктивним властивістю є властивість компактності образів. Взаємне розташування точок в обраному просторі вже містить інформацію про те, як слід розділити безліч точок. Ця інформація і визначає то властивість разделимости образів, що виявляється достатнім для самонавчання системи розпізнаванню образів.

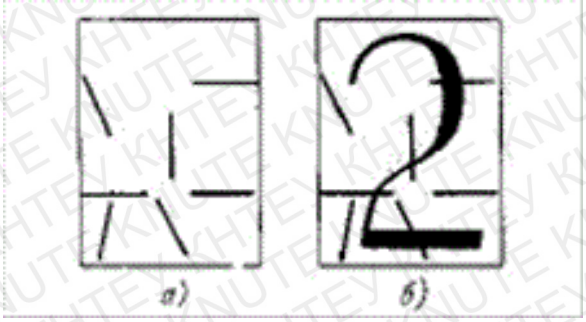


Рис. 2.3 – Схема розпізнавання цифр

Навчанням зазвичай називають процес вироблення в деякій системі тієї чи іншої реакції на групи зовнішніх ідентичних сигналів шляхом багаторазового впливу на систему зовнішньої коректування. Таку зовнішню коригування в навчанні прийнято називати "заохоченнями" і "покараннями". Механізм генерації цієї коригування практично повністю визначає алгоритм навчання. Самонавчання відрізняється від навчання тим, що тут додаткова інформація про вірність реакції системі не повідомляється. Адаптація - це процес зміни параметрів і структури системи, а можливо, і керуючих впливів на основі поточної інформації з метою досягнення певного стану системи при початковій невизначеності і мінливих умовах роботи.

Можливий спосіб побудови розпізнають машин, заснований на розрізненні будь-яких ознак підлягають розпізнаванню фігур. В якості ознак можуть бути обрані різні особливості фігур, наприклад, їх геометричні властивості (характеристики складових фігури кривих), топологічні властивості (взаємне розташування елементів фігури) і т.п. Відомі розпізнають машини, в яких розрізнення букв або цифр проводиться, по так званому "методу зондів" (рис. 2.3), тобто за кількістю перетинів контура фігури з кількома особливим чином розташованими прямими. Якщо проектувати цифри на поле з зондами, то виявиться, що кожна з цифр перетинає цілком певні зонди, причому комбінації перетинаються зондів різнідля всіх десяти цифр. Ці комбінації і використовуються в якості ознак, за якими проводиться розрізнення цифр. Такі машини успішно справляються, наприклад, з читанням машинописного тексту, але їх можливості обмежені тим шрифтом (або групою подібних шрифтів), для якого була розроблена система ознак. Робота зі створення набору еталонних фігур або системи ознак повинна проводитися людиною. Якість роботи машини, т. п. Надійність"впізнавання" пропонованих фігур визначається якістю цієї попередньої підготовки і без участі людини не може бути підвищено. Описана машина не є навчається машиною.

Для того щоб ввести зображення в машину, потрібно перевести його на машинний мову, тобто закодувати, представити у вигляді деякої комбінації символів, якими може оперувати машина. Кодування плоских фігур можна здійснити самим різним чином. Краще прагнути до найбільш "природному" кодування зображень. Будемо малювати фігури на деякому полі, розбитому вертикальними і горизонтальними прямими на однакові елементи - квадратики. Елементи, на які впало зображення, будемо суцільно зачернять, інші - залишати білими. Домовимося позначати чорні елементи одиницею, білі - нулем. введемо послідовнунумерацію всіх елементів поля, наприклад, в кожному рядку зліва направо і по рядках зверху вниз. Тоді кожна фігура, намальована на такому полі, буде однозначно відображатися кодом, що складається з стількох цифр (одиниць і нулів), скільки елементів містить поле.

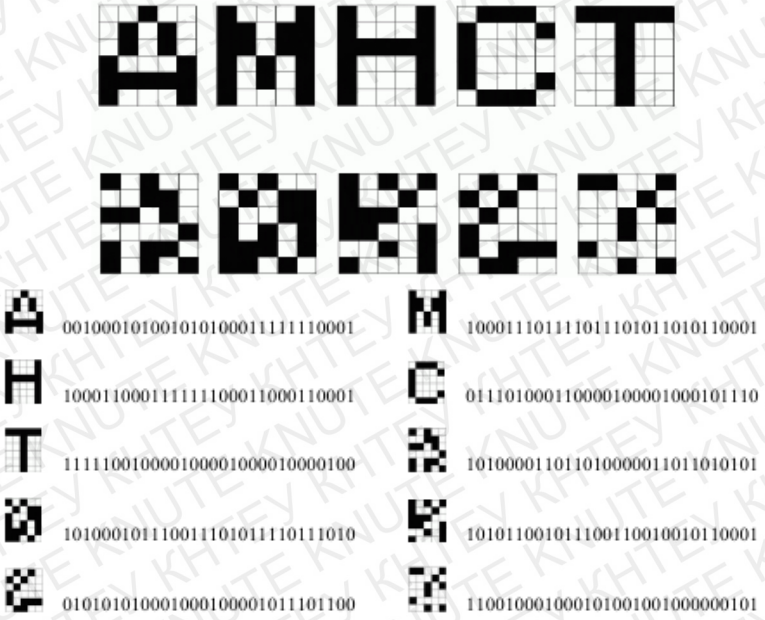


Рис. 2.4 – Приклади кодування зображень

Таке кодування (рис. 2.4) вважається "природним" тому, що розбиття зображення на елементи лежить в основі роботи нашого зорового апарату. Дійсно, сітківка ока складається з великого числа окремих чутливих елементів (так званих паличок і колбочок), пов'язаних нервовими волокнами із зоровими

відділами головного мозку. Чутливі елементи сітківки передають за своїми нервових волокнах у головний мозок сигнали, інтенсивність яких залежить від освітленості даного елемента. Таким чином, зображення, спроектоване оптичною системою ока на сітківку, розбивається паличками і колбочками на окремі ділянки, і за основними елементами в деякому коді передається в мозок. Окремі елементи поля називаються рецепторами, а саме поле - полем рецепторів.

Сукупність усіх плоских фігур, які можна зобразити на поле рецепторів, складає якесь безліч. Кожна конкретна фігура з цієї сукупності є об'єкт цього безлічі. Будь-якому їх таких об'єктів відповідає певний код. Точно також будь-якого коду відповідає певне зображення на поле рецепторів. Взаємно однозначна відповідність між кодами і зображеннями дозволить оперувати тільки кодами, пам'ятаючи про те, що зображення завжди може бути відтворене за його кодом.

Ємність ІНС - число образів, що пред'являються на входи ІНС для розпізнавання. Для поділу множини вхідних образів, наприклад, за двома класами достатньо всього одного виходу. При цьому кожен логічний рівень - "1" і "0" - буде позначати окремий клас. На двох виходах можна закодувати вже 4 класу і так далі. Для підвищення достовірності класифікації бажано ввести надмірність шляхом виділення кожному класу одного нейрона у вихідному шарі або, що ще краще, декількох, кожен з яких навчається визначати приналежність образу до класу зі своїм ступенем достовірності, наприклад: високого, середнього та низької. Такі ІНС дозволяють проводити класифікацію вхідних образів, об'єднаних в нечіткі (розмиті або пересічні) безлічі. Це властивість наближає подібні ІНС до умов реального життя.

2.2 Алгоритмічні побудови

Алгоритмічна універсальність ЕОМ означає, що на них можна програмно реалізовувати (у вигляді машинної програми) будь-які алгоритми перетворення інформації, будь то обчислювальні алгоритми, алгоритми управління, пошуку доведення теорем, тривимірні графічні або аудіо композиції.

Однак не слід думати, що обчислювальні машини і роботи можуть в принципі вирішувати будь-які завдання [4]. Було суворо доведено існування таких типів завдань, які не можуть бути єдиний і ефективний алгоритм, що вирішує всі завдання даного типу; в цьому сенсі неможливо рішення таких задач і за допомогою обчислювальних машин. Цей факт сприяє кращому розумінню того, що можуть робити машини і чого вони не можуть зробити.

Серед властивостей штучних нейронних мереж основним є їх здатність до навчання, вони навчаються найрізноманітнішими методами. Більшість методів навчання походять від загальних передумов, і має багато ідентичних характеристик. Їх навчання нагадує процес інтелектуального розвитку людської особистості. Можливості навчання штучних нейронних мереж обмежені. Проте, вже отримані переконливі досягнення, такі як "говорить мережу" Сейновскі, і виникає багато інших практичних застосувань [5]. Мережа навчається, щоб для деякого безлічі входів давати необхідне безліч виходів. Кожне таке вхідний (або вихідний) безліч розглядається як вектор. Навчання здійснюється шляхом послідовного пред'явлення вхідних векторів з одночасної підстроюванням ваг відповідно до певної процедури. У процесі навчання ваги мережі поступово стають такими, щоб кожен вхідний вектор виробляв вихідний вектор.

Навчальні алгоритми можуть бути класифіковані як алгоритми навчання з вчителем і без вчителя.

Навчання з вчителем критикувалося за свою біологічну неправдоподібність. Важко уявити навчальний механізм в мозку, який би порівнював бажані і дійсні значення виходів, виконуючи корекцію за допомогою зворотного зв'язку. Навчання без вчителя є набагато більш правдоподібною моделлю навчання в біологічній системі. Кохоненом і іншими, вона не потребує цільового векторі для виходів і, отже, не вимагає порівняння з зумовленими ідеальними відповідями. Навчальна множина складається лише з вхідних векторів. Навчальний алгоритм підлаштовує ваги мережі так, щоб виходили узгоджені вихідні вектори, т. п. Щоб пред'явлення досить близьких вхідних векторів давало однакові виходи. Процес навчання, отже, виділяє статистичні властивості навчальної множини і групує подібні вектори в класи. Пред'явлення на вхід вектора з даного класу дасть певний вихідний вектор, але до навчання неможливо передбачити, який вихід буде проводитися даним класом вхідних векторів.

Отже, виходи подібної мережі повинні трансформуватися в деяку зрозумілу форму, зумовлену процесом навчання.

Процес навчання Штучною Нейронної Мережі (ІНС) нового класу задач включає наступні стадії [7]:

1.Формулюється постановка задачі і виділяється набір ключових параметрів, що характеризують предметну область.

2. Вибирається парадигма нейронної мережі (модель, що включає в себе вид вхідних даних, порогової функції, структури мережі і алгоритмів навчання), найбільш підходяща для вирішення даного класу задач. Як правило, сучасні нейропакет, нейроплата й еволюційний [8] дозволяють реалізувати не одну, а кілька базових парадигм.

3.Готується, можливо, більш широкий набір навчальних прикладів, організованих у вигляді наборів вхідних даних, асоційованих з відомими

вихідними значеннями. Вхідні значення для навчання можуть бути неповні і частково суперечливі.

4. Вхідні дані по черзі пред'являються ІНС, а отримане вихідне значення порівнюється з еталоном. Потім проводиться підстроювання вагових коефіцієнтів міжнейронних з'єднань для мінімізації помилки між реальним і бажаним виходом мережі.

5. Навчання повторюється до тих пір, поки сумарна помилка у всій безлічі вхідних значень не досягне прийнятного рівня, або ІНС не прийде в стаціонарний стан. Розглянутий метод навчання нейроподібні мережі носить назву «зворотне поширення помилки» (error backpropagation) і відноситься до числа класичних алгоритмів нейроматематики.

Налагоджена і навчена ІНС може використовуватися на реальних вхідних даних, не тільки підказуючи користувачеві коректне рішення, а й оцінюючи ступінь його достовірності.

Існують різні алгоритми, що дозволяють розпізнавати образи. Алгоритм навчання машини "впізнавання" образів, заснований на методі січних гіперплоскостей (рис. 2.5), полягає в апроксимації розділяє гіперповерхні "шматками" гіперплоскостей і складається з наступних основних етапів:

1. Навчання. Формування розділеноъ поверхні: проведення січних площин, вирізання зайвих площин, вирізання зайвих шматків площин.

2. Розпізнавання нових об'єктів.

При використанні методу паралельних варіантів одночасно і незалежно один від одного на одному і тому ж матеріалі навчаються кілька машин. При пізнанні нових об'єктів кожна машина буде відносити ці об'єкти до якогось образу, може бути, не до одного і того ж. Остаточне рішення приймається "голосуванням" машин - об'єкт ставитися до того образу, до якого його віднесло більше число машин.

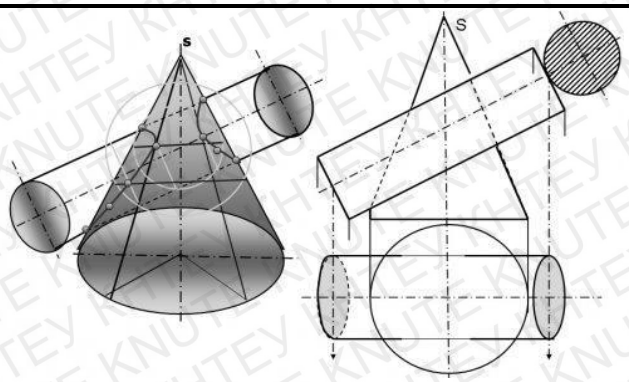


Рис. 2.5 – Метод січних гіперплоскостей

Спосіб підвищення надійності розпізнавання полягає в деяке поліпшення методу проведення січних площин. Можна припустити, що якщо проводити січні площині близько до площини, що проходить через середину прямої, що з'єднує об'єкт і опонент, перпендикулярний цій прямій, то результуюча поверхню буде ближче до істинної кордоні між образами. Експерименти підтверджують це припущення.

В алгоритмі, заснованому на методі потенціалів, з кожним збудженим елементом поля рецепторів можна зв'язати деяку функцію, рівну одиниці на цьому елементі і спадну в усіх напрямках від нього, тобто функцію φ, аналогічну електричного потенціалу з тією лише різницею, що в даному випадку R є відстань між двома сусідніми елементами поля рецепторів

$$\varphi(R) = \frac{1}{1 + \alpha R^2} \qquad (2.1)$$

Для підрахунку користуються таким простим правилом: кожен збуджений елемент поля рецепторів має "власний" потенціал, рівний одиниці, і який в свою чергу збільшує на 1/2 потенціали всіх (в тому числі і порушених) сусідніх з ним елементів по горизонталі, вертикалі і діагоналях. Однак цей метод кодування може бути поліпшений. Якщо пов'язати з кожним збудженим елементом поля рецепторів деякуфункцію, рівну одиниці на цьому елементі і спадну в усіх напрямках від нього, тобто функцію, аналогічну потенціалу φ, з

тією лише різницею, що в даному випадку R є відстань між двома сусідніми елементами поля рецепторів (рис. 2.6). Ця функція, може бути, аппроксимирована ступінчастою функцією, постійної в межах одного рецептора і стрибкоподібно змінюється на кордонах рецепторів.

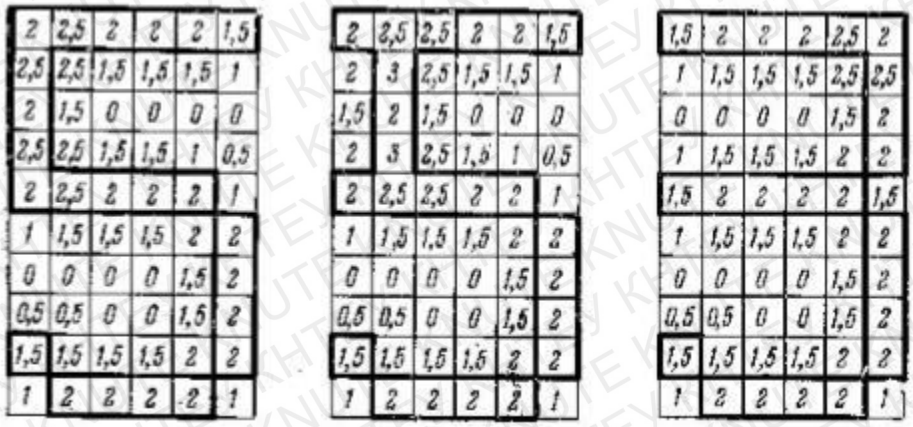


Рис. 2.6 – Метод потенціалів

Найпростіший алгоритм впізнавання, побудований на методі потенціалів, можна здійснити в два етапи:

1. Навчання (в процесі навчання запам'ятовуються коди всіх з'явилися точок і вказівки, до якого з образів відноситься кожна точки).

2.Впізнавання (в процесі пізнавання проводиться ідентифікація і видається інформація, до якого образа належить закодована матриця).

Незважаючи на деякі обмеження вихідної форми моделі персептрона Розенблатта, вона стала основою для багатьох сучасних найбільш складних алгоритмів навчання з учителем [10]. Прикладом персептрони з нерекуррентної мережею може служити вид, показаний на рис. 2.8. Вона використовує алгоритм навчання з учителем; іншими словами, навчальна вибірка складається з безлічі вхідних векторів, для кожного з яких зазначено свій необхідний вектор мети. Компоненти вхідного вектора представлені безперервним діапазоном значень; компоненти вектора мети є двійковими

величинами (0 або 1). Після навчання мережа отримує на вході набір безперервних входів і виробляє необхідний вихід у вигляді вектора з бінарними компонентами [11].

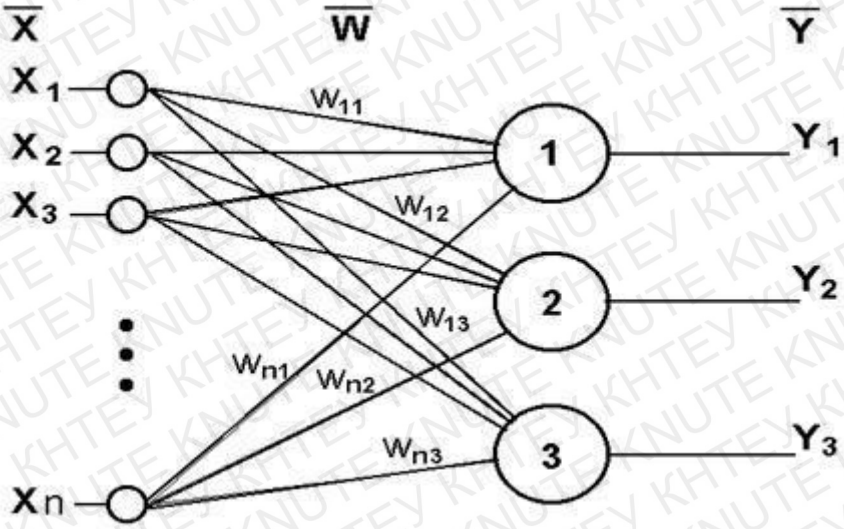


Рис. 2.7 - Багатошарова нейронна мережа

Навчання здійснюється наступним чином:

1. Рандомізують все ваги мережі в малі величини.

На вхід мережі подається вхідний навчальний вектор X і обчислюється сигнал NET від кожного нейрона, використовуючи стандартний вираз.

$$NET_j = \sum_i xw \qquad (2.2)$$

1. Обчислюється значення порогової функції активації для сигналу NET від кожного нейрона таким чином
2. Обчислюється помилка для кожного нейрона за допомогою віднімання отриманого виходу з необхідного виходу:

$$error_j = target_j - OUT_j \qquad (2.3)$$

3. Кожна вага модифікується в такий спосіб:

$$W_{ij}(t+1) = w_{ij}(t) + ax_j error_j \qquad (2.4)$$

6. Повторюються кроки з другого по п'ятий доти, поки помилка не стане досить малою.

Персептрон Розенблатта обмежується бінарними виходами. Уїдроу разом з Хоффом розширили алгоритм навчання персептрона на випадок безперервних виходів, використовуючи сигмоїдальну функцію. Вони розробили математичний доказ того, що мережа при певних умовах буде сходитися до будь-якої функції, яку вона може уявити. Їх перша модель - Адалін має один вихідний нейрон, більш пізня модель - Мадаліною розширює її на випадок з багатьма вихідними нейронами.

Результати досліджень Кохонена на самоорганізованих структурах, для задач розпізнавання образів класифікують образи, представлені векторними величинами, в яких кожен компонент вектора відповідає елементу образу. Алгоритми Кохонена грунтуються на техніці навчання без учителя. Після навчання подача вхідного вектора з даного класу буде приводити до вироблення збуджуючого рівня в кожному вихідному нейроні; нейрон з максимальним порушенням представляє класифікацію. Так як навчання проводиться без вказівки цільового вектора, то немає можливості визначати заздалегідь, який нейрон буде відповідати даному класу вхідних векторів. Проте, це планування легко проводиться шляхом тестування мережі після навчання.

Алгоритм трактує набір з n вхідних ваг нейрона як вектор в n-вимірному просторі. Перед навчанням кожен компонент цього вектора ваг ініціалізується в випадкову величину. Потім кожен вектор нормалізується в вектор довжиною в один символ в просторі ваг. Це робиться діленням кожного випадкового ваги на квадратний корінь з суми квадратів компонент цього вагового вектора.

Всі вхідні вектора навчального набору також нормалізуються, і мережа навчається згідно з наступним алгоритмом:

1. Вектор X подається на вхід мережі.
2. Визначаються відстані D_j (в n-вимірному просторі) між X і ваговими векторами W_l кожного нейрона. В евклідовому просторі це відстань обчислюється за такою формулою

$$D_j = \sqrt{\sum_i (x - w)} \qquad (2.5)$$

3. Нейрон, який має ваговий вектор, найближчий до X, оголошується переможцем. Цей ваговий вектор, званий W_c, стає основним в групі вагових векторів, які лежать в межах відстані D від W_c.
4. Група вагових векторів налаштовується у відповідності з наступним виразом:

$$W_j(t+1) = W_j(t) + \alpha[X - W] \qquad (2.6)$$

5. Повторюються кроки з 1 по 4 для кожного вхідного вектора

У процесі навчання нейронної мережі значення D і α поступово зменшуються. Коефіцієнт α на початку навчання промені встановлювати приблизно рівним 1 і зменшувався в процесі навчання до 0, в той час як D може на початку навчання дорівнювати максимальному відстані між ваговими векторами і в кінці навчання стати настільки маленьким, що буде навчатися тільки один нейрон.

Передбачається, що вхідні вектори фактично групуються в класи відповідно до їх становищем у векторному просторі. Певний клас буде асоціюватися з певним нейроном, переміщаючи його ваговий вектор в напрямку центру класу і сприяючи його порушення при появі на вході будь-якого вектора даного класу. Після навчання класифікація виконується за допомогою подачі на вхід мережі випробуваного вектора, обчислення

збудження для кожного нейрона з подальшим вибором нейрона з найвищим збудженням як індикатора правильної класифікації.

2.3 Персептрон, як модель розпізнавання

Великий поштовх розвитку нейрокібернетики дав нейрофізіолог Френк Розенблат, який запропонував модель розпізнавання образів, яку назвав "Персептрон" (від латинського percepto - розумію, пізнаю). При її розробці він виходив з деяких прийнятих уявлень про структуру мозку і зорового апарату. Прагнучи відтворити функції людського мозку, він використовував просту модель біологічного нейрона (рис. 2.8) і систему зв'язків між ними.

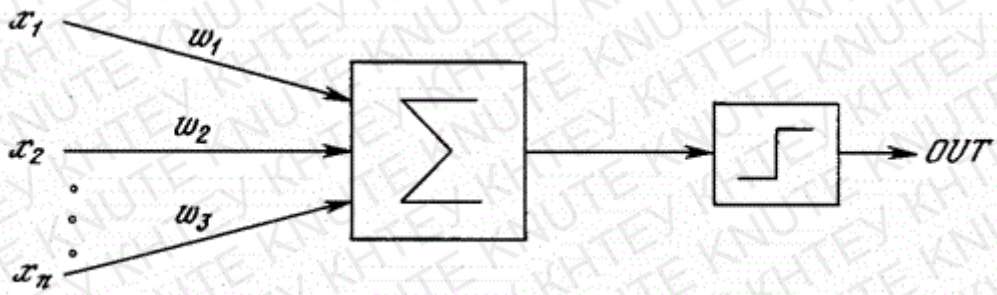


Рис. 2.8 – Персептронний нейрон

Сприймає пристроєм персептрона служить фотоелектрична модель сітківки - поле рецепторів, що складається з декількох сотень фотосопротивлений (Selементов) (див. Рис. 2.9). Кожен елемент поля рецепторів може перебувати в двох станах - збудженому або збудженому стані, в залежності від того, падає чи ні на відповідне фотосопротівленіе контур проектованої на поле фігури. На виході кожного елемента з'являється сигнал x i (i = 1, 2, ..., n, де n- число елементів), що дорівнює одиниці, якщо елемент збуджений, і нулю - в іншому випадку. Наступною сходинкою персептрона служать, так звані, асоціативні елементи або А-елементи. Кожен

А-елемент має декілька входів і один вихід. При підготовці персептрона до експерименту до входів А-елемента підключаються виходи рецепторів, причому підключення будь-якого з них можна виготовити зі знаком плюс або зі знаком мінус.

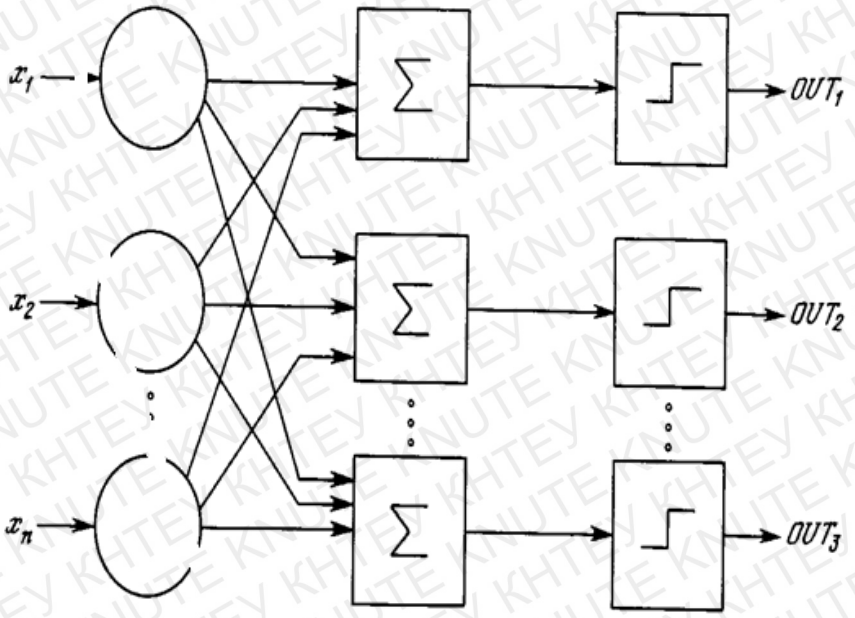


Рис. 2.9 – Персептронний нейрон з багатьма виходами

Вибір рецепторів, що підключаються до даного А-елементу, також як і вибір знака підключення, проводиться випадково. В ході експерименту зв'язок рецепторів з Аелементамі залишається незмінною. А-елементи виробляють алгебраїчне підсумовування сигналів [15], що надійшли на їх входи, і отриману суму порівнюють з однаковою для всіх А-елементів величиною θ.

Якщо сума більше θ, А-елемент збуджується і видає на виході сигнал, що дорівнює одиниці. Якщо сума менше θ, А-елемент залишається збудженим і вихідний його сигнал дорівнює нулю [16]. Таким чином, вихідний сигнал j-го А-елемента, де величина rij приймає значення +1, якщо і -й рецептор підключений до входу j-го Аелемента зі знаком плюс; і значення -1, якщо рецептор підключений зі знаком мінус, і значення 0, якщо і-ий рецептор до j-му А-елементу не може підключитися (j = 1, 2, ..., m, де m - число А-елементів).

Вихідні сигнали А-елементів за допомогою спеціальних пристроїв (підсилювачів) множаться на змінні коефіцієнти λj. Кожен з цих коефіцієнтів може бути позитивним, негативним або рівним нулю і змінюватися незалежно від інших коефіцієнтів. Вихідні сигнали підсилювачів підсумовуються, і сумарний сигнал надходить на вхід,так званого, реагує елемента або R-елемента. Якщо σ позитивна або дорівнює нулю, R-елемент видає на виході одиницю, а якщо σ негативна – нул.

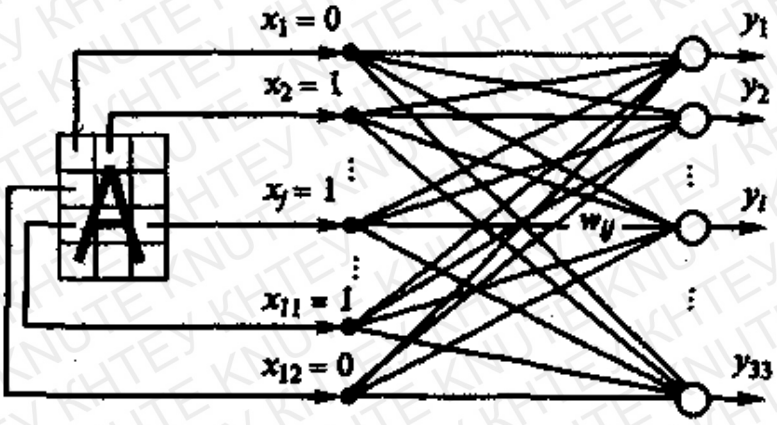


Рис. 2.10 – Персептрон з декількаами виходами

Припустимо, що на поле рецепторів проектуються фігури, що належать до двох різних образів. Якщо виявиться можливим привести персептрон в такий стан, щоб він з достатньою надійністю видавав на виході 1, при появі на його вході фігур одного образу, то це буде означати, що персептрон має здатність навчатися розрізнення двох образів .

Описана структура персептрона дозволяє розділяти запропоновані об'єкти тільки на два безлічі. Для розпізнавання більшого числа образів, наприклад, трьох А, В і С може бути застосований персептрон, побудований за схемою (рис. 2.10). Вихідний сигнал кожного А-елемента надходить не на один, а на кілька (за кількістю розрізняються образів) підсилювачів. Після множення на λ вихідні сигнали надходять на суматори Σ, кількість яких також дорівнює числу розрізняються образів. Замість R-елемента встановлено

пристрій, що порівнює між собою вихідні сигнали сумматоров. Пред'явлений об'єкт відноситься до того образу, акумулятор якого має найбільший сигнал.

Кожна з груп А-елементів, з'єднаних зі своїм R-елементом, за структурою і дією цілком аналогічна персептрону, здатному розбивати об'єкти на два класи. Навчання персептрона складається з ряду послідовних тактів. У кожному такті персептрону пред'являється об'єкт одного з образів. Залежно від реакції персептрона на пред'явлену йому фігуру проводиться за певними правилами зміна коефіцієнтів λj. Виявляється можливим за деякий кінцеве кількість тактів привести персептрон в такий стан, що він з достатньою впевненістю розпізнає пред'являються йому фігури.

Можливі два типи алгоритмів навчання персептрона. Перший з них не враховує правильності відповідей персептрона в процесі навчання, і зміна λj в кожному такті проводиться незалежно від того, «дізнався» або не "дізнався» персептрон пред'явлену в цьому такті фігуру. В алгоритмах другого типу коефіцієнти λj змінюються з урахуванням правильності відповідей персептрона. Алгоритм першого типу здійснюється наступним чином. Заздалегідь умовляються, що після навчання персептрон повинен видавати на виході 1 встановить обґрунтованість, наприклад, об'єктів образу А, і нуль - при пред'явленні образу - В. Потім пред'являють персептрону об'єкти кожного з образів. У кожному такті персептрон відповідає на пред'явлений йому об'єкт порушенням деяких А-елементів. Навчання полягає в тому, що коефіцієнти λj порушених у даному такті А-елементів збільшуються на деяку величину (наприклад на одиницю), якщо в цьому такті був пред'явлений об'єкт образу А, і зменшується на цю ж величину, якщо був пред'явлений об'єкт образу В. Природно , що така зміна коефіцієнтів λj повинно призводити до підвищення правильності відповідей персептрона, так як збільшення λj порушених а-елементів призводить до збільшення сигналу на вході R-елемента, а їх зменшення - до зменшення сигналу. Відповідно до прийнятого умовою

персептрон буде давати правильні відповідіякщо образу А будуть відповідати позитивні, а образу В - негативні сигнали на вході R-елемента.

При розробці персептрона Ф. Розенблат намагався моделювати деякі властивості живого мозку [18]. Персептрон або будь-яка програма, що імітує процес розпізнавання, працюють в двох режимах: в режимі навчання і в режимі розпізнавання. Перш за все, на відміну від розглянутих раніше алгоритмів, алгоритм персептрона в ході навчання не вимагає запам'ятовування пред'явлених об'єктів, а при розпізнаванні - перебору всіх "відомих" йому фігур. У цьому сенсі робота персептрона має певну схожість з роботою мозку, який формує уявлення про образ, що не запам'ятовуючи окремих його об'єктів, і дізнається нові об'єкти без порівняння їх з кожним з раніше зустрічаються. Далі, структура персептрона має деякі спільні риси зі структурою вищої нервової системи.

Зокрема, рецептори персептрона є досить близькою аналогією рецепторів зорового апарату, а А-елементи мають певну схожість з нейронами. Відомо, що нейрони мають властивість порушувати, якщо інтенсивність сигналу, одержуваного нейроном від пов'язаних з ним рецепторів (або інших нейронів), перевершує деяку порогову величину.

Властивість персептрона допускативипадковий характер зв'язків "рецептор - А-елемент", аналогічно деяким властивостям структури головного мозку. Можливо, що зв'язку між нейронами мозку в більшості випадків також мають випадковий характер, тобто випадково варіюються у різних тварин одного біологічного виду. Якщо припустити протилежне, тобто допустити, що всі зв'язки між нейронами мозку точно фіксовані і однакові у всіх тварин даного виду, і що зміна цих зв'язків може привести до різкого порушення роботи мозку, то можна припустити також, що відомості про всі ці зв'язки повинні передаватися у спадок. А так як кількість нейронів мозку

обчислюється мільярдами, то таке припущення призводить до фантастично великим обсягом генетичної інформації.

Разом з тим на прикладі персептрона видно, як біологічно природним є поняття компактного безлічі, бо навчання розпізнаванню таких множин виявляється можливим при випадкових зв'язках між рецепторами і нейронами. Відомо, що мозок здатний зберігати або відновлювати багато свої функції при серйозних пошкодженнях, викликаних травмами або захворюваннями. Стійкість персептрона до порушень його структури має певну схожість з цією властивістю мозку. З усіх цих міркувань не можна зробити висновок, що алгоритми мозку і персептрона збігаються. Однак в даний час персептрон є, мабуть, найбільш правдоподібною моделлю мозку.

Здатність штучних нейронних мереж навчатися є їх найбільш важливою властивістю. Подібно біологічним системам, які вони моделюють, ці нейронні мережі самі моделюють себе в результаті спроб досягти кращої моделі поведінки.

Використовуючи критерій лінійної роздільності, можна вирішити, чи здатна одношарова нейронна мережа реалізовувати потрібну опцію. Навіть в тому випадку, коли відповідь позитивна, це принесе мало користі, якщо у нас немає способу знайти потрібні значення для ваг і порогів. Щоб мережа представляла практичну цінність, потрібен систематичний метод (алгоритм) для обчислення цих значень. Розенблатт зробив це в своєму алгоритмі навчання персептрона разом з доказом того, що персептрон може бути навчений усього, що він може реалізовувати.

Навчання може бути з учителем або без нього. Для навчання з учителем потрібен «зовнішній» учитель, який оцінював би поведінку системи і керував її подальшими модифікаціями. При навчанні без учителя, мережа шляхом самоорганізації робить необхідні зміни.

Алгоритм навчання персептрона може бути реалізований на цифровому комп'ютері або іншому електронному пристрої, і мережа стає в певному сенсі самоподстраівающейся. З цієї причини процедуру підстроювання ваг зазвичай називають «навчанням» і кажуть, що мережа «навчається». Доказ Розенблатта стало основною віхою і дало потужний імпульс дослідженням у цій галузі.

РОЗДІЛ 3.
АРХІТЕКТУРНИЙ ШАБЛОН MVC

3.1 Структура організації програмного коду та архітектура MVC

Model-View-Controller (MVC) дозволяє розробляти, реалізовувати і тестувати кожну частину програми незалежно від будь-якої іншої, зберігаючи код організованим. Збереження організованого коду означає можливість швидко знайти те, що необхідно, щоб перевірити, швидко виправити, змінити і додати нові функції. Це також означає більш ефективний код і кращий спосіб повторного використання його для більш швидких додатків.

Без вагомих підстав використовувати нову структуру, технологію або тренд багатьом розробникам важко, і перш за все тому, що вони не можуть знайти мотивацію для вивчення нової теми. Але тільки не в відношенні MVC, архітектура якої дуже важлива, а застосовувати методи MVC для веб необхідно.

Ймовірно, одним з найбільших переваг є те, що багато розробників розуміють і використовують структуру MVC для створення веб-додатків. Через цю узгодженості управління проектом між декількома розробниками стає простіше. В основному веб-додаток або частина програмного забезпечення слідують за структурою MVC. Якщо структура представлена трьома основними типами функціональності, тоді зрозуміло, що це - MVC:

1. Код моделі зазвичай відображає реальні речі. Цей код може містити необроблені дані або визначати основні компоненти програми. Наприклад, якщо користувач створював додаток Todo, код моделі

визначав би, що таке «завдання» і що таке «список», оскільки це основні компоненти цього додатка.

2. Вид, або уявлення - перегляд коду складається з усіх функцій, які безпосередньо взаємодіють з користувачем. Це код, який робить додаток красивим і в іншому випадку визначає, як користувач бачить і взаємодіє з ним.
3. Контролер діє як зв'язок між моделлю і представленням, приймаючи для користувача введення і вирішуючи, що з ним робити. Це мозок додатки і пов'язує модель і уявлення. Контролер вважають «середнім рівнем». Він взаємодіє з користувачем, збираючи дані, контактує з моделлю, отримуючи необхідні дані, а потім з поданням, щоб відповісти користувачеві.

Коли користувач виконує будь-яку дію, він спочатку переходить до контролера. Він буде приймати будь-які дані, наприклад, $ _GET, $ _POST змінні в PHP, і визначати, що робити з цими даними. Коротше кажучи, моделі відносяться до обробки даних і розширеної функціональності. Тому завдання контролера в цій точці полягає в тому, щоб визначити, яку модель слід викликати, а потім відкрити відповідну функцію всередині цієї моделі. Після виклику функції він буде знаходити результат, зазвичай в змінному середовищі.

Модель - це просте уявлення про те, що користувач виконує в додатку. Модель MVC - це що повинні представити в коді, наприклад книги користувача, його банківського рахунку або чогось ще. Модель відповідає за зберігання функцій і змінних, які пов'язані з тим, що вона представляє.

Можна думати про логіку моделі як про базової концепції об'єктно-орієнтованого програмування. Тут моделі - це просто «класи». Не варто плутати з класами в контролерах, які технічно теж структуровані як класи.

Нарешті, після того як контролер запитує інформацію з моделі, вона відправляє її в уявлення.

Вид схожий на систему шаблонів програми та може існувати для певного типу макета сторінки, мобільного виду або для певної теми. У поданні будуть відображатися всі розмітки і CSS, які традиційно використовуються при створенні статичної веб-сторінки.

MVC - це те, що бачить користувач, коли контролер звертається до нього. Контролер просто перенаправляє користувача на правильний вид, після того як вони отримали дані з моделі і перенаправили цю інформацію в уявлення. Потім уявлення відображає інформацію, надану їм, в тому форматі, в якому вона структурована.

Велика ідея MVC полягає в тому, що кожна частина коду має свою мету, і ці цілі різні. Деякі з кодів містять дані додатки, деякі роблять додаток приємним, а деякі з них контролюють функціональність.

Вважається, що це MVC, програма, яка здатна організувати основні функції коду в свої, акуратно організовані ящики. Структура файлу для використання MVC стандартним чином досить проста - є просто папки для уявлень, моделей і контролерів, і всі вони пов'язані один з одним через один каталог. Звичайно, з будь-яким веб-додатком у користувача також будуть інші папки і файли, такі як індексний файл і папка для зображень.

Нижче представлена проста структура каталогів MVC з деякими прикладами файлів. Кожен розробник може мати свої власні назви, тут важливо приймати розумне стандартну угоду про них.

Модельний розділ визначає, які дані про порушення додаток. Якщо стан цих даних змінюється, тоді модель зазвичай повідомляє уявлення, а іноді і контролер, якщо для контролю оновленого уявлення потрібна інша логіка. Наприклад, для додатка списку покупок модель вкаже, які дані повинні

містити елементи списку - елемент, ціна та інші, і які елементи списку вже присутні.

Подання визначає, як повинні відображатися дані програми. У додатку списку покупок уявлення визначатиме як список, представлений користувачеві, і отримає дані для відображення з моделі. Контролер містить логіку, яка оновлює модель у відповідь на вхід користувачів програми.

Так, наприклад, список покупок може містити форми введення і кнопки, які дозволяють додавати або видаляти елементи. Ці дії вимагають поновлення моделі, тому вхід відправляється на контролер, який потім відповідним чином керує моделлю, що відправляє оновлені дані в уявлення. Однак також можна просто оновити уявлення, щоб відобразити дані в іншому форматі, наприклад, змінити порядок предметів в алфавітному порядку або з найнижчою до найвищої ціни. У цьому випадку контролер може впоратися з цим безпосередньо, без оновлення модель.

3.2 Основний принцип роботи MVC

Принцип MVC полягає в тому, щоб розділити додаток на 3 основні частини, відомі як Модель, Вид (перегляд) і Контролер. Видимими на діаграмі є прямі асоціації (червоні стрілки) і виведені асоціації (блакитні стрілки). Виведені асоціації - це ті, які можуть здаватися очевидними з точки зору користувача, а не виходячи з фактичного дизайну програмного забезпечення.

Простий спосіб виконання умови:

1. Користувач взаємодіє з поданням - натисканням на посилання або відправкою форми.
2. Контролер обробляє введення користувача і передає інформацію в модель.
3. Модель отримує інформацію і оновлює її стан, додає дані в базу даних, наприклад, обчислює сьогоднішню дату.

4. Перегляд перевіряє стан Моделі і відповідає відповідно, перерахувавши недавно введені дані.
5. Вид очікує наступного взаємодії користувача.

Це проста концепція - Business Logic - обчислення логічних процесів додатки. Наприклад, бізнес-логіка простого календаря мала б розрахувати, яка дата, який день тижня і який день місяця, якщо потрібно представити всі дні цього місяця. Або здійснювати обслуговування веб-контенту за допомогою Spring MVC, яка дає можливість будувати додаток зі статичної домашньою сторінкою, яка приймає запити HTTP GET.

Багато фреймворки MVC використовують систему шаблонів для забезпечення дотримання принципу DRY, що робить його дуже зручним для повторного використання коду без необхідності переписування. Існують рамки MVC, які працюють на Smarty або використовують власні механізми шаблонів. Простим попередженням є те, що деякі движки шаблонів мають досить складний синтаксис - програмісту потрібно перевірити їх, перш ніж починати розробку.

Вважається, що MVC - це ще одна дуже хороша реалізація філософії DRY (Do not Repeat Yourself). По суті, DRY використовується Ruby on Rails і декількома іншими реалізаціями, а ідея полягає в тому, що програміст пише щось один раз і один раз використовує код. Принцип DRY визначається як «кожна частина повинна мати єдине, однозначне, авторитетне уявлення всередині системи». Правильна реалізація DRY означає, що зміна одного елемента системи не змінює незв'язані елементи, що досить логічно.

3.3 Конвенція по конфігурації та основні переваги та недоліки MVC

Це парадигма дизайну, яка, по суті, намагається видалити кількість рішень, які розробник повинен зробити. Це досягається шляхом створення

структури з угодами, які зазвичай вимагають все елементи. Розробнику потрібно лише змінити те, що дійсно необхідно. Це досить просто. Наприклад, для форми, що містить елементи, які завжди потрібні і мають однакові значення. Форма має тег, який визначає дію, метод, ім'я, id і enctype. Наприклад, якщо не потрібно щось змінювати, досить легко отримати ім'я форми, ідентифікатор і дію з URL-адреси.

MVC шаблон проектування використовується в розробці програмного забезпечення, основоположний принцип якого заснований на ідеї про те, що логіка додатка повинна бути відділена від його уявлення. Простіше кажучи, це просто кращий спосіб відокремити логіку додатки від дисплея. Як і всякий метод програмування, він має свої переваги і недоліки.

Переваги MVC:

1. Швидкий процес розробки, підтримує швидке і паралельний розвиток.
2. З MVC один програміст може працювати над виставою, а інший може працювати над контролером для створення бізнес-логіки.
3. Додаток, розроблене з його застосуванням, в три рази швидше, ніж додаток, розроблене з іншими шаблонами розробки.
4. Можливість надання декількох видів.
5. У MVC можна створювати кілька подань.
6. Копіювання дублікатів дуже обмежена, оскільки воно відокремлює дані і логіку від дисплея.
7. Підтримка асинхронної технології, яка допомагає розробникам розробляти швидко завантажувати додаток.
8. Модифікація не впливає на всю модель, тому що частина моделі не залежить від частини переглядів. Тому будь-які зміни в Моделі не впливатимуть на всю архітектуру.

9. Шаблон NET MVC повертає дані без застосування будь-якого форматування, тому одні й ті ж компоненти можуть використовуватися і викликатися для використання з будь-яким інтерфейсом.
10. За допомогою цієї платформи дуже легко розробляти URL-адреси, оптимізовані для SEO, для отримання більшої кількості відвідувань з певної програми.

Недоліки MVC:

1. Підвищена складність.
2. Неефективність доступу до даних.
3. Складність використання MVC з сучасним призначеним для користувача інтерфейсом.
4. Потрібно кілька програмістів.
5. Потрібне знання кількох технологій. Розробник знає код клієнтської сторони і html-код.

3.4 Висновки до розділу 3

Виходячи з наведенного огляду MVC є сучасним архітектурним шаблоном для проектування та розробки як Веб додатків так і нативних додатків які використовують багаторівневу взаємодію компонентів системи для більш швидкої так корректної роботи додатку.

Розглянуті переваги та недоліки дають чітке розуміння в доцільності використання данного підходу до шаблонного проектування архітектури программних додатків. Незважаючи на недоліки данного підходу він є сучасним та зручним для роботи як над великими проектами так і над маленькими программними додатками

РОЗДІЛ 4
РОЗРОБКА МОБІЛЬНОГО ДОДАТКУ WESCAN ТА ОГЛЯД ОПЕРАЦІЙНОЇ СИСТЕМИ IOS

4.1 Мобільні додатки та операційна система iOS

Мобільний додаток – це програмне забезпечення, призначене для роботи на смартфонах, планшетах та інших мобільних пристроях. Багато мобільних додатків вже установлені на самому пристрої або можуть бути завантажені на нього з додатків онлайн-магазинів, таких як App Store, BlackBerry App World та інших. Раніше мобільні додатки використовувалися для швидкої перевірки електронної пошти, але їх високий попит призвів до розширення їхніх призначень і в інших областях, таких як ігри для мобільних телефонів, GPS, спілкування, перегляд відео та користування інтернетом. iOS (відома як iPhone OS до червня 2010 року) — це власницька мобільна операційна система від Apple. Розроблена спочатку для iPhone, вона стала операційною системою також для iPod Touch, iPad і Apple TV. Apple не дозволяє роботу ОС на мобільних телефонах інших фірм. iOS є похідною від OS X, отже, є за своєю природою Unix-подібною операційною системою. Користувацький інтерфейс iOS заснований на концепції прямої маніпуляції з використанням жестів Multi-Touch. Елементи інтерфейсу управління складаються з повзунків, перемикачів і кнопок. Він призначений для безпосереднього контакту користувача з екраном пристрою. Внутрішній акселерометр використовуються деякими програмами для реагування на струшування пристрою, яке є також загальною командою скасування, або обертання пристрою у у трьох вимірах, що є загальною командою перемикання між книжковим та альбомним режимами

4.2 Специфікації та особливостіі мови програмування Swift

Програмування - основа основ комп'ютерної техніки. Корпорація Apple давно відома своїм умінням задавати тон розвитку індустрії на роки вперед, але з огляду на поступове сходження купертіновцев з ринку професійних рішень, надкушене яблуко асоціюється в першу чергу з споживчими товарами. Однак чергове дослідження громадської думки показує, що розробники ПЗ більш ніж задоволені свіжою пропозицією компанії - мовою програмування Swift. Згідно з інформацією ресурсу Stack Overflow, майже 80 відсотків професіоналів із задоволенням працювали або планують працювати з випущеним не так давно інструментом розробки від Apple. Дослідникам вдалося опитати 26 тисяч відвідувачів з більш ніж 150 країн світу. Близько третини постійно зайнятих в сегменті написання мобільного ПЗ респондентів працюють в основному на платформі iOS, а менше половини також займаються створенням додатків для конкуруючої ОС Android. 20 % опитаних не змогли визначитися, швидше за все, сюди входять фахівці, які регулярно розробляють програмне забезпечення для різних платформ.

Рис. 4.1. Оцінки мов програмування

Swift-багатопарадигмова компільована мова програмування, розроблена компанією Apple для того, щоб співіснувати з Objective C і бути стійкішою до помилкового коду. Swift була представлена на конференції розробників WWDC 2014. Мова побудована з LLVM компілятором, включеного у Xcode 6 beta. 8 Безкоштовний посібник мови програмування Swift доступний для завантаження у магазині iBooks. Компілятор Swift побудований з використанням технологій вільного проекту LLVM. Swift успадковує найкращі елементи мов C і Objective-C, тому синтаксис звичний для знайомих з ними розробників, але водночас відрізняється використанням засобів автоматичного розподілу пам'яті і контролю переповнення змінних і масивів, що значно збільшує надійність і безпеку коду. При цьому Swift-програми компілюються у машинний код, що дозволяє забезпечити високу швидкодію.

Swift щільно інтегрований у власницьке середовище розробки Xcode і не може бути використаний відособлено на платформах, відмінних від OS X. Окремо варто відзначити, що Swift від компанії Apple не варто плутати з досить давно розроблюваною скриптовою мовою Swift, націленої на багатонитеве програмування і поставленого під вільною ліцензією Apache.

4.3 Архітектура мобільної операційної системи iOS

Архітектура iOS схожа з базовою архітектурою операційної системи Mac OS X. На найвищому рівні iOS являє собою проміжний шар між обладнанням пристроя та додатками, які відображаються на екрані. Дуже рідко доводиться створювати додатки, які будуть звертатися до обладнання безпосередньо. Замість цього, 9 додатки взаємодіють з обладнанням через набір чітко визначених системних інтерфейсів, які захищають додатки від змін обладнання. Ця абстракція дозволяє дуже легко писати програми, які коректно працюють на пристроях з різними апаратними можливостями.

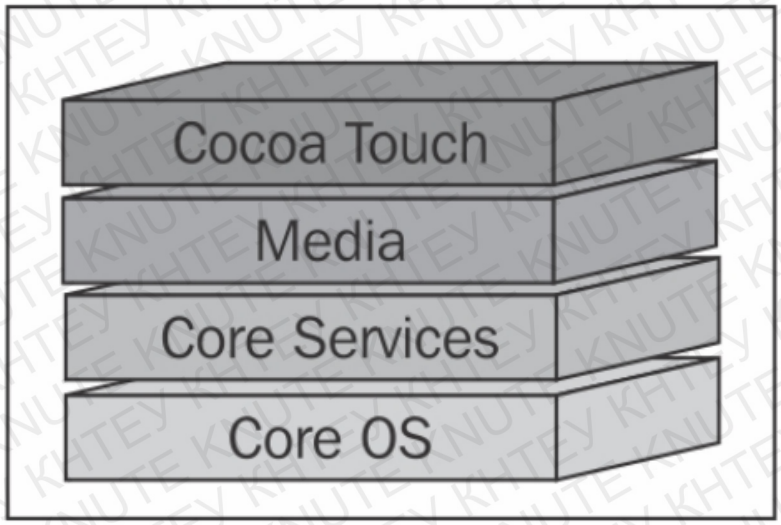


Рис. 4.2. Архітектура платформи iOS

Хоча додатки в цілому захищені від змін обладнання, але доводиться враховувати відмінності між пристроями при написанні коду. Наприклад, у деяких пристроїв є камера, а у деяких її немає. Якщо додаток вміє працювати при наявності або відсутності якоїсь функції, то використовуючи інтерфейс відповідної бібліотеки можна визначити, доступна ця функція чи ні. Додатки, яким потрібна наявність певного обладнання, повинні декларувати цю вимогу в файлі зі списком властивостей додатки (Info.plist). Реалізація технологій iOS може бути представлена у вигляді набору шарів. На самому нижньому шарі операційної системи знаходяться основні служби та технології, від яких залежать всі додатки; на більш високих рівнях знаходяться складніші служби і технології. При написанні додатків всюди, де це можливо, рекомендується використовувати бібліотеки вищого рівня, ніж бібліотеки низького рівня. Бібліотеки вищого рівня написані для того, щоб забезпечити об'єктно-орієнтовані абстракції для низькорівневих структур. Ці абстракції істотно полегшують написання коду, тому що вони зменшують обсяг коду, який необхідно написати і приховують досить складні функції, такі як сокети і потоки. І хоча вони приховують низькорівневі функції, ці функції як і раніше доступні для розробників. Розробники, які вважають за краще

використовувати низькорівневі бібліотеки або хочуть скористатися наявними можливостями, яких не надають високорівневі бібліотеки, можуть їх використовувати.

4.4 Опис основних інструментів розробки мобільного додатку для платформи iOS

Розробка додатків для платформи iOS ведеться переважно на мові Swift та Objective-C. Для створення програм на цих мовах необхідне спеціальне програмне забезпечення. Найостанніші версії цього ПО можна завантажити з офіційного сервісу поширення програмного забезпечення для iOS та macOS - App Store. До цього програмного комплексу відносяться такі інструменти як IDE Xcode. Xcode - це пакет інструментів для розробки додатків під Mac OS X і iPhone OS, розроблений Apple. IDE Xcode поширюється вільно, тому будь-хто може його скачати і використовувати. Таким чином, перш ніж приступити до розробки програми на базі ОС iOS, необхідно підготувати інструментарій . При розробці додатків на базі ОС iOS необхідно використовувати середу IDE Xcode. Через офіційний сервіс поширення програмного забезпечення для операційних систем iOS та macOS - App Store.

Середовище розробки Xcode - це пакет інструментів для розробки додатків під Mac OS X і iPhone OS, розроблений Apple. Остання версія Xcode11.2, . Третя версія не підтримується старими версіями Mac OS, для яких XCode також доступний для безкоштовного звантаження через Apple Developer Connection. Оновлення можна безкоштовно скачати на офіційному сайті підтримки. На сьогодні є єдиним засобом написання «універсальних» (Universal Binary) прикладних програм для Mac OS X. Xcode тісно інтегрований з фреймворком Cocoa. Його використовують і при розробці самої Apple Mac OS X. Цей набір інструментів включає в Xcode IDE. Xcode IDE надає все, що потрібно для розробки: від професійних редакторів з функцією

автозаповнення коду і Cocoa рефакторінга, до налаштування opensource компіляторів. Xcode IDE розроблений з нуля, щоб можна було скористатися всіма можливостями Cocoa і новітніми технологіями Apple. Xcode включає можливості, щоб полегшити розробки iOS-додатків, включаючи наступні:

- систему управління проектом для визначення програмних продуктів;
- контекстно-залежний інспектор для перегляду інформації про виділений в коді символі;
- середовище для редагування коду, що включає можливості підсвічування синтаксису, автозаповнення коду та індексацію символів; • просунуту програму перегляду і пошуку документації Apple;
- просунуту систему збирання проекту з перевіркою залежностей і перевіркою правил складання;
- компілятори GCC, що підтримують мови C, C ++, Objective-C, Objective-C ++ та інші;
- підтримка LLVM і Clang для мов C, C ++ і Objective-C;
- вбудоване налагодження вихідного коду з використанням GDB; • розподілені обчислення, що дозволяють поширювати великі
- проекти через кілька мережевих пристроїв;
- інтелектуальна компіляція, яка прискорює час компіляції одного файлу;
- просунуті можливості по налагодженню коду;
- просунуті засоби налагодження, що дозволяють робити глобальні
- зміни в коді без зміни його поведінки;
- підтримка знімків проекту, які надають легше управління вихідним кодом;
- підтримка запуску засобів продуктивності для аналізу програми;
- підтримка вбудованої системи управління вихідним кодом;
- підтримка Apple Script для автоматизації процесу складання;

- підтримка налагоджувальної інформації в форматах DWARF і Stabs (налагоджувальна інформація для всіх проектів за замовчуванням генерується в форматі DWARF).

4.5 Проектування мобільного додатку WeScan

Першочергово потрібно прозподілити файлову структуру для проетку. Побудова файлової структури проекту дозволить розподілити програмний код за його функціональним призначенням. Тобто потрібно визначити першочергові точки програмування та логіку їх розміщення.

При побудові файлової структури проекту слід врахувати функціональний підхід, тобто створити файли які будуть відповідати за збереження окремих частин коду і їх підключення безпосередньо у процесі програмування. Це є першочерговим та невідьемним єтапом в створенні та проектуванні программного продукту. Основні файли розструктуровано по папкам для більш зрозумілої логіки використання та групування. В папці проекту WeScan розміщуються безпсередньо файли додатку для платформи iOS, головний інтерфейс,та контролери роботи додатку.

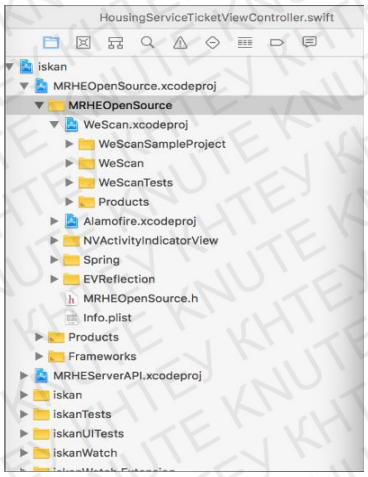


Рис.4.3 Структура файлів програмного продукту в середовищі розробки Xcode

Відповідно до структури найбільш відповідальним стала схема роботи данних та структура файлу коду по суміщенню всіх функцій. Було прийняте рішення по змущеню всіх функцій та іх співпраця в одному файлі AppDelegate.swift. увібравши в себе основні функції та класси. Частина лістингу представлена нижче. Для початку роботи основного фреймворку іпортується інформація із Pods_LiveApp.framework після чого задається фінкція відображення динамічного тексту на основному елементі інтерфейсу який відповідає за початок роботи добатку. Текст беспосередньо є динамічним та змінюється після початку або кінця роботи додатку.

Відповідно до структури найбільш відповідальним стала схема роботи данних та структура файлу коду по суміщенню всіх функцій. Було прийняте рішення по змущеню всіх функцій та іх співпраця в одному файлі WeScan.xcodeproj. увібравши в себе основні функції та класси. Частина лістингу представлена нижче. Для початку роботи основного фреймворку іпортується інформація після чого задається фінкція відображення динамічного тексту на основному елементі інтерфейсу який відповідає за початок роботи добатку. Текст беспосередньо є динамічним та змінюється після початку або кінця роботи додатку

В основний контролер додатку додано обробку та імпорт основних кітів та фреймворків які були розглянуті в першому розділі та додану обробку контроллерів запуску та завантаження основних модулів додатку

4.5.1 Використання CoreData

Модуль використовується для додавання та відображення графічних елементів та їх видалення з бази.

Для цього потрібно зв’язати дані із бази даних за допомогою NSFetchedResultsController:

if let managedObjectContext =

(UIApplication.sharedApplication().delegate as? AppDelegate)?.managedObjectContext {

fetchResultController = NSFetchedResultsController(fetchRequest: fetchRequest,

managedObjectContext: managedObjectContext, sectionNameKeyPath: nil, cacheName: nil)

fetchResultController.delegate = self do { try fetchResultController.performFetch() = fetchResultController.fetchedObjects as! [Restaurant] } catch { print(error) } }

А також описати контроллер для управління NSFetchedResultsController:

func controllerWillChangeContent(controller: NSFetchedResultsController) {

tableView.beginUpdates() } func controller(controller: NSFetchedResultsController,

didChangeObject anObject: AnyObject, atIndexPath indexPath: NSIndexPath?,

forChangeType type: NSFetchedResultsChangeType, newIndexPath: NSIndexPath?)

{ switch type { case .Insert: if let _newIndexPath = newIndexPath {

tableView.insertRowsAtIndexPaths([_newIndexPath], withRowAnimation: .Fade) }

case .Delete: if let _newIndexPath = newIndexPath {

tableView.deleteRowsAtIndexPaths([_newIndexPath], withRowAnimation: .Fade) }

case .Update: if let _newIndexPath = newIndexPath { 63

tableView.reloadRowsAtIndexPaths([_newIndexPath], withRow

4.5.2 Використання Segue

Segue – це спосіб зміни екранів в iOS. Один із найбільш популярних різновидів – це push (з версії iOS 8 – show). Push segue завжди зміщує поточний вид зправа наліво. Головна особливість Segue – можливість передачі даних між двома UIViewController

Для цього було використано наступний метод:

override func prepareForSegue(segue: UIStoryboardSegue, sender: AnyObject?) { if segue.identifier ==

"showDetail" { if let indexPath = tableView.indexPathForSelectedRow { let destinationController = segue.destinationViewController as! DetailViewController destinationController= (searchController.active) ? searchResult[indexPath.row] : [indexPath.row] } } }

4.5.3 Використання CloudKit

Даний фреймворк дає можливість зв’язувати мобільний додаток з iCloud. Для цього потрібно отримати дані з сервера:

func getRecordsFromCloud() { var newRestaurantsFromCloud:[CKRecord] = [] let

cloudContainer = CKContainer.defaultContainer() let publicDatabase =

cloudContainer.publicCloudDatabase let predicate = NSPredicate(value: true) let

query = CKQuery(recordType: "Restaurant", predicate: predicate)

query.sortDescriptors = [NSSortDescriptor(key: "creationDate", ascending: false)] let

queryOperation = CKQueryOperation(query: query) queryOperation.desiredKeys

["name"] queryOperation.queuePriority = .VeryHigh queryOperation.resultsLimit =

50 queryOperation.recordFetchedBlock = { (record: CKRecord!) -> Void in

4.6 Висновки до розділу 4

Було розглянуто мобільну операційну систему iOS як основну операційну систему для розробки прогрраммного продукуту з використанням шаблону проектування MVC саме для данної мобільної платформи.

Розробленний мобільний додаток для сканування та відновлення зображень відповідає поставленим цілям з урахуванням всіх специфік та можливостей вибраної платформи iOS. Розроблений інтерфейс є максимально інформативним для користувача для максимально зручного використовування додатку в реальному часі.

Проведено оптимізацію методу розробки мобільного додатку для найефективнішої роботи додатку та найефективнішого та найменш ресурсномісткого запуску додатку. В основі створення модифікованих методів оптимізації мобільних додатків полягає у зменшенні кількості тих незручних моментів, з якими може зіткнутися користувач під час використання придбаного мобільного пристрою.

Оптимізація часу запуску мобільного додатку під керуванням операційної системи iOS, яка модифікована шляхом використання лінивого завантаження статичних бібліотек при старті додатку, зображує, що навіть такі невеликі рішення, виконують істотний ефект на загальну картину в цілому.

ВИСНОВКИ ТА ПРОПОЗИЦІЇ

Тема відновлення зобраджень є досить цікавою і представляє широке поле для подальших досліджень в галузі.Специфіка данного сегменту полягає в тому що відновлення зображень є дуже глубокою темою яка може включате в себе роботу багатьох різних алгоритмів та засобів. В данній роботі розглянуто відновлення зображень з використанням нейронних мереж які використовуться для відновлення зображення. Самі нейронні мережі складаються з багатьох модулів та частин які взаємодиють між собою та дозволяють використовувати її в программних додатках завдяки ризноманітним методологіям архітектури программного забеспечення. В роботі використовується шаблон MVC який дозволяє використовувати нейронні мережі як части ну программного забезпечення. Актуальність теми підкреслюється широким спектром можливостей для втілення ідей у вигляді мобільного додатку. Розглянуто основні засоби розробки, доведено і обгрунтовано правильність вибору цих засобів для проектування. На основі проведеного аналізу встановлено, що найкращим середовищем розробки є Xcode, а мовою програмування – Swift. Адже вони найбільше підходять для розробки додатків під платформу iOS.

В результаті виконання даної дипломної роботи досліджено основні особливості шаблону MVC та на його основі було розроблено мобільний додаток для операційної системи iOS Розглянуто та проаналізовано основні особливості архітектури платформи iOS та технології розробки мобільних застосувань, визначено основні переваги та недоліки даної платформи.

Наведено основні фреймворки використанні в вирішенні поставленої

					КНТЕУ 121 06 -11МР		
Зм.	*Аркуш*	*№ докум.*	*Підпис*	*Дата*			
Зав. каф.		Криворучко О.В.		24.01.20	Відновлення зображень з використанням архітектурного шаблону проектування MVC	*Стадія*	*Аркуш*	*Аркушів*
Керівник		Криворучко О.В.		24.01.20		*ВП*	*65*	*69*
Гарант		Криворучко О.В.		24.01.20		Факультет інформаційних технологій 2м курс, 6 група		
Розробив		Коломієць І.О.		24.01.20	*Висновки та пропозиції*			

задачі, , адже ці інструменти найкраще підходять для розробки мобільних додатків, використовуючи саме Swift, які в свою чергу повинні мати простий, лаконічний, привабливий користувацький інтерфейс, швидку і зручну навігацію, високу швидкодію. Розглянуті фреймворки і технології відносяться до вищого рівня архітектури, тому вони забезпечують об'єктно-орієнтовані абстракції для низькорівневих структур. Ці абстракції істотно полегшують написання коду, тому що вони зменшують обсяг коду, який необхідно написати і приховують досить складні функції, такі як сокети і потоки.

Проведено аналіз розробленного програмного рішення для сканування та відновлення зображення з використанням нейроних мереж які дозволяють сканувати та відновлювати зображення з підвищенням якості зображення в реальному часі. Це дозволяє зробити висновок про те, що розробка додатків такого функціоналу актуальна і доцільна. Головне грамотно оцінити, для кого і навіщо створюється ПЗ. Тільки корисна розробка отримає гідне визнання з боку користувачів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Білас О.Є. Якість програмного забезпечення та тестування : навч. посіб. / О.Є. Білас. – Львів : Львів. політехніка, 2011. – 216
2. Далека В.Д. Алгоритми та методи обчислень. Ч. 1. Алгоритми: Методичні вказівки до виконання лабораторних робіт./ В. Д. Далека, Н.М. Бондіна, Н.Ю. Любченко – Харків: НТУ «ХПІ». – 2015. – 70 с.
3. Табунщик Г.В. Проектування та моделювання програмного забезпечення сучасних інформаційних систем : навчальний посібник / Г.В. Табунщик, Т.І. Каплієнко, О.А. Петрова. – Запоріжжя : ЗНТУ, 2016. – 270 с
4. Бушуєв С.Д. Методологія управління бюджетними проектами: Посібник/С.Д. Бушуєв, С.В. Цюцюра, О.В. Криворучко та ін. – К. : КНУБА, 2016. – 196 с.
5. Петренко Н.О. Управління проектами навчальний посібник. / Н.О. Петренко, Л.О. Кустріч, М. О. Гоменюк. – К. : «Центр учбової літератури», 2015. – 244 с
6. The Swift Programming Language https://swift.org/documentation/ переклад на українську - https://killobatt.gitbooks.io/-swift/content/1_language_guide/0_the_basics.html
7. Jonathan Peppers. Xamarin Cross-platform Application Development Second Edition / Peppers J. – Birmingham : Packt, 2015. 10.Nilanchala Panigrahy. Xamarin Mobile Application Development for Android
8. Second Edition / Panigrahy N. – Birmingham : Packt, 2015. – 296 с. 11.Charles Petzold. Creating Mobile Apps with Xamarin.Forms First Edition /
9. Petzold C. – Redmond : Microsoft Press, 2016. – 1187 с.

10. Best Hybrid Mobile App UI Frameworks: HTML5, CSS and JS – Режим доступу : http://noeticforce.com/best-hybrid-mobile-app-ui-frameworks-html5- js-css/. – Дата доступу : 29.05.2018.
11. What is KaiOS? [Електронний ресурс]. – Режим доступу: https://primea2z.blogspot.com/2017/10/what-is-kaios.html
12. KaiOS Emerging OS [Електронний ресурс]. – Режим доступу: https://www.developingtelecoms.com/tech/devices-platforms/7595
13. KaiOS is on 30 million phones and now has Google apps, Facebook, Twitter [Електронний ресурс]. – Режим доступу: https://mobilityarena.com/kaios-30-million-phones/
14. iOS [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/IOS
15. About mono – Режим доступу: http://www.mono-project.com/docs/about-mono/. – Дата доступу : 05.05.2018.
16. Xamarin application fundamentals – Режим доступу : http://developer.xamarin.com/guides/crossplatform/application_fundamentals/. – Дата доступу : 20.05.2018
17. Кириченко О. В. Порівняння схем квантування дискретних спектрів зображень / А. С. Васюра, А. Я. Кулик // Інформаційні технології та
18. Принципи ущільнення та перетворення зображення: Монографія/ В. П. Кожем‟ко, А. С. Васюра, Н. В. Сачанюк-Кавецька, О. В. Кириченко. – Вінниця : ВНТУ, 2010. – С.261.
19. Kyrychenko O. V. Transformation and compression of images in optoelectronic logic and time environments / Kozhemiako V. P., Maidaniuk V. P., Kyrychenko O. V. // Naukaistudia. – No 21 (89). – 2013. – P. 35–43. – ISSN 1561- 6894.
20. Кириченко О. В. Спосіб паралельного формування ущільненої суми часоімпульсних сигналів / Міжнародна науково-практична конференція

«Інформаційні технології та комп‟ютерна інженерія», 28-30 травня, 2010 р., Вінниця, Україна. – С. 283–284.

21. XML [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/XML
22. Layouts [Електронний ресурс]. – Режим доступу: https://developer.android.com/guide/topics/ui/declaring-layout
23. Автоматизація складання [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Автоматизація_складання
24. Build and Release Management [Електронний ресурс]. – Режим доступу: https://web.archive.org/web/20070930222406/http://freshmeat.net/articles/view/392/
25. Gradle [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Gradle

ДОДАТКИ

ДОДАТОК А

Лістинг файлу WeScane.xcscheme

```
<?xml version="1.0" encoding="UTF-8"?>
<Scheme
   LastUpgradeVersion = "1020"
   version = "1.3">
   <BuildAction
      parallelizeBuildables = "YES"
      buildImplicitDependencies = "YES">
      <BuildActionEntries>
         <BuildActionEntry
            buildForTesting = "YES"
            buildForRunning = "YES"
            buildForProfiling = "YES"
            buildForArchiving = "YES"
            buildForAnalyzing = "YES">
            <BuildableReference
               BuildableIdentifier = "primary"
               BlueprintIdentifier = "A1D4BCEC202C4F4100FCDDEC"
               BuildableName = "WeScan.framework"
               BlueprintName = "WeScan"
               ReferencedContainer = "container:WeScan.xcodeproj">
            </BuildableReference>
         </BuildActionEntry>
      </BuildActionEntries>
   </BuildAction>
   <TestAction
      buildConfiguration = "Debug"
      selectedDebuggerIdentifier = "Xcode.DebuggerFoundation.Debugger.LLDB"
      selectedLauncherIdentifier = "Xcode.DebuggerFoundation.Launcher.LLDB"
      codeCoverageEnabled = "YES"
      shouldUseLaunchSchemeArgsEnv = "YES">
      <Testables>
         <TestableReference
            skipped = "NO">
            <BuildableReference
```

```
            BuildableIdentifier = "primary"
            BlueprintIdentifier = "A1D4BCF4202C4F4100FCDDEC"
            BuildableName = "WeScanTests.xctest"
            BlueprintName = "WeScanTests"
            ReferencedContainer = "container:WeScan.xcodeproj">
          </BuildableReference>
        </TestableReference>
      </Testables>
      <MacroExpansion>
        <BuildableReference
          BuildableIdentifier = "primary"
          BlueprintIdentifier = "A1D4BCEC202C4F4100FCDDEC"
          BuildableName = "WeScan.framework"
          BlueprintName = "WeScan"
          ReferencedContainer = "container:WeScan.xcodeproj">
        </BuildableReference>
      </MacroExpansion>
      <AdditionalOptions>
      </AdditionalOptions>
    </TestAction>
    <LaunchAction
      buildConfiguration = "Debug"
      selectedDebuggerIdentifier = "Xcode.DebuggerFoundation.Debugger.LLDB"
      selectedLauncherIdentifier = "Xcode.DebuggerFoundation.Launcher.LLDB"
      launchStyle = "0"
      useCustomWorkingDirectory = "NO"
      ignoresPersistentStateOnLaunch = "NO"
      debugDocumentVersioning = "YES"
      debugServiceExtension = "internal"
      allowLocationSimulation = "YES">
      <MacroExpansion>
        <BuildableReference
          BuildableIdentifier = "primary"
          BlueprintIdentifier = "A1D4BCEC202C4F4100FCDDEC"
          BuildableName = "WeScan.framework"
          BlueprintName = "WeScan"
          ReferencedContainer = "container:WeScan.xcodeproj">
        </BuildableReference>
      </MacroExpansion>
      <EnvironmentVariables>
```

```
      <EnvironmentVariable
        key = "IMAGE_DIFF_DIR"
        value = "$(SOURCE_ROOT)/$(PROJECT_NAME)Tests/FailureDiffs"
        isEnabled = "YES">
      </EnvironmentVariable>
      <EnvironmentVariable
        key = "FB_REFERENCE_IMAGE_DIR"
        value = "$(SOURCE_ROOT)/$(PROJECT_NAME)Tests/ReferenceImages"
        isEnabled = "YES">
      </EnvironmentVariable>
    </EnvironmentVariables>
    <AdditionalOptions>
    </AdditionalOptions>
  </LaunchAction>
  <ProfileAction
    buildConfiguration = "Release"
    shouldUseLaunchSchemeArgsEnv = "YES"
    savedToolIdentifier = ""
    useCustomWorkingDirectory = "NO"
    debugDocumentVersioning = "YES">
    <MacroExpansion>
      <BuildableReference
        BuildableIdentifier = "primary"
        BlueprintIdentifier = "A1D4BCEC202C4F4100FCDDEC"
        BuildableName = "WeScan.framework"
        BlueprintName = "WeScan"
        ReferencedContainer = "container:WeScan.xcodeproj">
      </BuildableReference>
    </MacroExpansion>
  </ProfileAction>
  <AnalyzeAction
    buildConfiguration = "Debug">
  </AnalyzeAction>
  <ArchiveAction
    buildConfiguration = "Release"
    revealArchiveInOrganizer = "YES">
  </ArchiveAction>
</Scheme>
}
```

Лістинг файлу ImageScannerController.swift

```
import UIKit
import AVFoundation

/// A set of methods that your delegate object must implement to interact with the image scanner interface.
public protocol ImageScannerControllerDelegate: NSObjectProtocol {

    /// Tells the delegate that the user scanned a document.
    ///
    /// - Parameters:
    ///   - scanner: The scanner controller object managing the scanning interface.
    ///   - results: The results of the user scanning with the camera.
    /// - Discussion: Your delegate's implementation of this method should dismiss the image scanner controller.
    func imageScannerController(_ scanner: ImageScannerController, didFinishScanningWithResults results: ImageScannerResults)

    /// Tells the delegate that the user cancelled the scan operation.
    ///
    /// - Parameters:
    ///   - scanner: The scanner controller object managing the scanning interface.
    /// - Discussion: Your delegate's implementation of this method should dismiss the image scanner controller.
    func imageScannerControllerDidCancel(_ scanner: ImageScannerController)

    /// Tells the delegate that an error occured during the user's scanning experience.
    ///
    /// - Parameters:
    ///   - scanner: The scanner controller object managing the scanning interface.
    ///   - error: The error that occured.
    func imageScannerController(_ scanner: ImageScannerController, didFailWithError error: Error)
}

/// A view controller that manages the full flow for scanning documents.
/// The `ImageScannerController` class is meant to be presented. It consists of a series of 3 different screens which guide the user:
/// 1. Uses the camera to capture an image with a rectangle that has been detected.
/// 2. Edit the detected rectangle.
```

```
/// 3. Review the cropped down version of the rectangle.
public final class ImageScannerController: UINavigationController {

  /// The object that acts as the delegate of the `ImageScannerController`.
  public weak var imageScannerDelegate: ImageScannerControllerDelegate?

  // MARK: - Life Cycle

  /// A black UIView, used to quickly display a black screen when the shutter button is presseed.
  internal let blackFlashView: UIView = {
    let view = UIView()
    view.backgroundColor = UIColor(white: 0.0, alpha: 0.5)
    view.isHidden = true
    view.translatesAutoresizingMaskIntoConstraints = false
    return view
  }()

  override public var supportedInterfaceOrientations: UIInterfaceOrientationMask {
    return .portrait
  }

  public required init(image: UIImage? = nil, delegate: ImageScannerControllerDelegate? = nil) {
    super.init(rootViewController: ScannerViewController())

    self.imageScannerDelegate = delegate

    if #available(iOS 13.0, *) {
      navigationBar.tintColor = .label
    } else {
      navigationBar.tintColor = .black
    }
    navigationBar.isTranslucent = false
    self.view.addSubview(blackFlashView)
    setupConstraints()

    // If an image was passed in by the host app (e.g. picked from the photo library), use it instead of the document scanner.
    if let image = image {
      detect(image: image) { [weak self] detectedQuad in
        guard let self = self else { return }
```

```
                let editViewController = EditScanViewController(image: image, quad: detectedQuad, rotateImage: false)
                self.setViewControllers([editViewController], animated: false)
            }
        }
    }

    override public init(nibName nibNameOrNil: String?, bundle nibBundleOrNil: Bundle?) {
        super.init(nibName: nibNameOrNil, bundle: nibBundleOrNil)
    }

    public required init?(coder aDecoder: NSCoder) {
        fatalError("init(coder:) has not been implemented")
    }

    private func detect(image: UIImage, completion: @escaping (Quadrilateral?) -> Void) {
        // Whether or not we detect a quad, present the edit view controller after attempting to detect a quad.
        // *** Vision *requires* a completion block to detect rectangles, but it's instant.
        // *** When using Vision, we'll present the normal edit view controller first, then present the updated edit view controller later.

        guard let ciImage = CIImage(image: image) else { return }
        let orientation = CGImagePropertyOrientation(image.imageOrientation)
        let orientedImage = ciImage.oriented(forExifOrientation: Int32(orientation.rawValue))

        if #available(iOS 11.0, *) {
            // Use the VisionRectangleDetector on iOS 11 to attempt to find a rectangle from the initial image.
            VisionRectangleDetector.rectangle(forImage: ciImage, orientation: orientation) { (quad) in
                let detectedQuad = quad?.toCartesian(withHeight: orientedImage.extent.height)
                completion(detectedQuad)
            }
        } else {
            // Use the CIRectangleDetector on iOS 10 to attempt to find a rectangle from the initial image.
            let detectedQuad = CIRectangleDetector.rectangle(forImage: ciImage)?.toCartesian(withHeight: orientedImage.extent.height)
            completion(detectedQuad)
        }
    }

    public func useImage(image: UIImage) {
```

```
            guard topViewController is ScannerViewController else { return }

            detect(image: image) { [weak self] detectedQuad in
              guard let self = self else { return }
              let editViewController = EditScanViewController(image: image, quad: detectedQuad, rotateImage:
false)
              self.setViewControllers([editViewController], animated: true)
            }
          }

          public func resetScanner() {
            setViewControllers([ScannerViewController()], animated: true)
          }

          private func setupConstraints() {
            let blackFlashViewConstraints = [
              blackFlashView.topAnchor.constraint(equalTo: view.topAnchor),
              blackFlashView.leadingAnchor.constraint(equalTo: view.leadingAnchor),
              view.bottomAnchor.constraint(equalTo: blackFlashView.bottomAnchor),
              view.trailingAnchor.constraint(equalTo: blackFlashView.trailingAnchor)
            ]

            NSLayoutConstraint.activate(blackFlashViewConstraints)
          }

          internal func flashToBlack() {
            view.bringSubviewToFront(blackFlashView)
            blackFlashView.isHidden = false
            let flashDuration = DispatchTime.now() + 0.05
            DispatchQueue.main.asyncAfter(deadline: flashDuration) {
              self.blackFlashView.isHidden = true
            }
          }
        }

        /// Data structure containing information about a scan, including both the image and an optional PDF.
        public struct ImageScannerScan {
          public enum ImageScannerError: Error {
            case failedToGeneratePDF
          }
```

```
        public var image: UIImage

        public func generatePDFData(completion: @escaping (Result<Data, ImageScannerError>) -> Void) {
            DispatchQueue.global(qos: .userInteractive).async {
                if let pdfData = self.image.pdfData() {
                    completion(.success(pdfData))
                } else {
                    completion(.failure(.failedToGeneratePDF))
                }
            }

        }

        mutating func rotate(by rotationAngle: Measurement<UnitAngle>) {
            guard rotationAngle.value != 0, rotationAngle.value != 360 else { return }
            image = image.rotated(by: rotationAngle) ?? image
        }
    }

    /// Data structure containing information about a scanning session.
    /// Includes the original scan, cropped scan, detected rectangle, and whether the user selected the enhanced scan. May also include an enhanced scan if no errors were encountered.
    public struct ImageScannerResults {

        /// The original scan taken by the user, prior to the cropping applied by WeScan.
        public var originalScan: ImageScannerScan

        /// The deskewed and cropped scan using the detected rectangle, without any filters.
        public var croppedScan: ImageScannerScan

        /// The enhanced scan, passed through an Adaptive Thresholding function. This image will always be grayscale and may not always be available.
        public var enhancedScan: ImageScannerScan?

        /// Whether the user selected the enhanced scan or not.
        /// The `enhancedScan` may still be available even if it has not been selected by the user.
        public var doesUserPreferEnhancedScan: Bool

        /// The detected rectangle which was used to generate the `scannedImage`.
```

```
        public var detectedRectangle: Quadrilateral

        @available(*, unavailable, renamed: "originalScan")
        public var originalImage: UIImage?

        @available(*, unavailable, renamed: "croppedScan")
        public var scannedImage: UIImage?

        @available(*, unavailable, renamed: "enhancedScan")
        public var enhancedImage: UIImage?

        @available(*, unavailable, renamed: "doesUserPreferEnhancedScan")
        public var doesUserPreferEnhancedImage: Bool = false

        init(detectedRectangle: Quadrilateral, originalScan: ImageScannerScan, croppedScan:
ImageScannerScan, enhancedScan: ImageScannerScan?, doesUserPreferEnhancedScan: Bool = false) {
            self.detectedRectangle = detectedRectangle

            self.originalScan = originalScan
            self.croppedScan = croppedScan
            self.enhancedScan = enhancedScan

            self.doesUserPreferEnhancedScan = doesUserPreferEnhancedScan
        }
    }  }
                                                                                    }
```

ДОДАТОК В

Головний інтерфейс програмного додатку WeScan

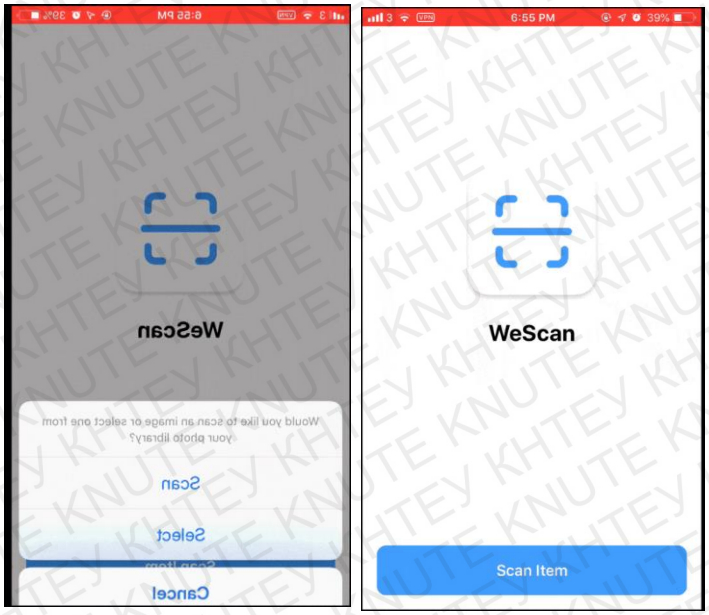


Рис. В.1. Скріншот головного інтерфейсу програмного додатку

ДОДАТОК Д

Зображення при скануванні документу на прикладі книжки

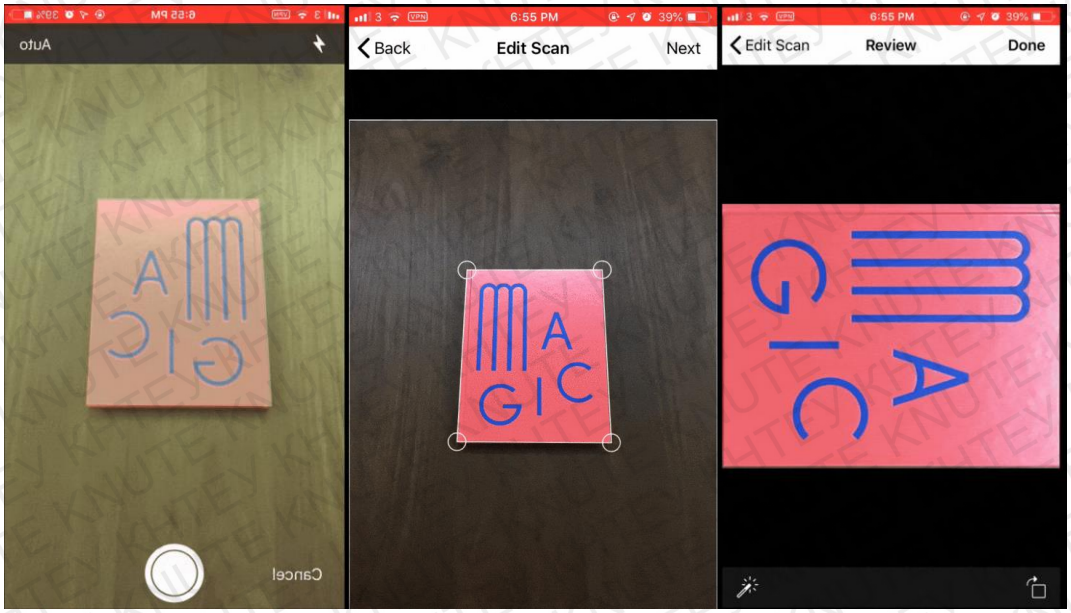


Рис. Д.1. Скріншот зораження на екрані, під час сканування