

## **ABSTRACT**

The thesis contains 73 pages, 41 figures and 0 references.

Video recordings of lectures are no longer a rarity in the conditions of distance learning. Videos may be in an inconvenient format for students or contain different artifacts due to compression, camera quality, and other factors. It is useful to have a presentation of the study material, which contains only the text from the board because such a view of the material is most like the compendium.

The aim of this work is creating an algorithm for obtaining panorama slides without a teacher from video lecture.

To perform the study the following instruments were used:

- SIFT for obtaining image's keypoints;
- homography for getting panorama slides;
- Boykov-Kolmogorov max-flow algorithm for creating mask of moving objects;
- convolutional neural networks for human detecting;
- denoising and binarizing operators.

Keywords: IMAGES PROCESSING, INTELLECTUAL VIDEO PROCESSING,  
VIDEO STABILIZATION

## **РЕФЕРАТ**

Дисертація містить 73 сторінки, 41 ілюстрацій і бібліографію з 0 найменувань.

Відеозаписи лекцій вже не рідкість в умовах дистанційного навчання. Відео може бути у незручному форматі для студентів або містити різні артефакти через стиснення з втратами, якість зйомки та інші фактори. Добре було б мати презентацію навчального матеріалу, яка містить лише текст з дошки, оскільки це більш наближено до формату конспекту.

Метою роботи є створення інформаційної технології, яка з відео лекції зможе створити панорамні слайди без викладача.

Для досягнення мети було використано:

- SIFT алгоритм для знаходження ключових точок зображення;
- гомографію для отримання панорамних слайдів;
- алгоритм максимального потоку Бойкова-Колмогорова для отримання маски рухомих об'єктів;
- згорткові нейронні мережі для детекції людини;
- оператори знешумлення та бінаризації зображень.

**Ключові слова: ОБРОБЛЕННЯ ЗОБРАЖЕНЬ, ІНТЕЛЕКТУАЛЬНЕ ОБРОБЛЕННЯ ВІДЕО, СТАБІЛІЗАЦІЯ ВІДЕО**

## ЗМІСТ

Перелік умовних позначень, символів, одиниць, скорочень і термінів . . . . .	9
Вступ . . . . .	10
1 Аналіз попередніх робіт. . . . .	12
1.1 Перша робота по оцифруванню дошки . . . . .	12
1.2 Автоматичне сканування дошки . . . . .	13
1.3 Відстежування об'єкту та віднімання фону . . . . .	13
1.4 Відокремлення написів дошки . . . . .	14
1.5 Сучасна робота . . . . .	15
Висновки до розділу 1 . . . . .	17
2 Постановка задачі створення слайдів . . . . .	18
2.1 Постановка задачі . . . . .	18
2.2 Пропоновані рішення . . . . .	19
2.3 Вхід алгоритму . . . . .	20
Висновки до розділу 2 . . . . .	22
3 Методи локалізації людини та рухомих об'єктів . . . . .	23
3.1 Локалізація рухомих об'єктів . . . . .	23
3.2 Застосування алгоритму Бойкова-Колмогорова . . . . .	26
3.3 Згорткові нейронні мережі . . . . .	28
Висновки до розділу 3 . . . . .	44
4 Створення панорами . . . . .	45
4.1 Пошук відповідності між кадрами . . . . .	45
4.2 Знаходження матриці гомографії за допомогою RANSAC . . . . .	51
4.3 Створення панорамного знімку . . . . .	54
Висновки до розділу 4 . . . . .	57
5 Обробка слайдів . . . . .	58
5.1 Порівняння слайдів . . . . .	58
5.2 Темпоральна медіана . . . . .	59

5.3	Швидка медіана для зображень . . . . .	61
5.4	Побудова швидкої медіани для зображень . . . . .	63
5.5	Порівняння складності . . . . .	66
	Висновки до розділу 5 . . . . .	68
6	Практичні результати . . . . .	69
6.1	Результати алгоритму Б-К та згорткових мереж . . . . .	69
6.2	Результати згорткових мереж . . . . .	71
6.3	Швидкість методів локалізації людини чи рухомих об'єктів . . . . .	72
6.4	Результати створення панорами . . . . .	73
6.5	Приклад роботи. . . . .	73

## ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

### **Стандартні позначення**

$\mathbb{N}$  — множина натуральних чисел;

$\mathbb{R}$  — множина дійсних чисел;

$\mathbb{R}^n$  —  $n$ -вимірний векторний простір дійсних чисел.

### **Позначення, введені в дисертації**

$F$  — множина кадрів відео;

$T$  — довжина відео (кількість кадрів);

$P$  — множина координат пікселів зображення;

$C$  — множина інтенсивностей пікселів зображення;

$F_p^i$  — інтенсивність пікселя з координатою  $p \in P$ ;

Б-К алгоритм — алгоритм Бойкова-Колмогорова.

## ВСТУП

**Актуальність роботи.** Задача створення конспекту лекцій не є новою та стала ще більш актуальною, оскільки з масовим переходом навчання у віддалений режим навчання постала проблема створення якісних лекційних матеріалів, зокрема слайдів, які містять стислу суть занять. Корисно мати технологію, що перетворює відео лекцію у стисле слайд шоу, на якому будуть написи з дошки без викладача і до того ж всю область дошки, навіть якщо камера рухалась.

Дана інформаційна технологія повинна мати ряд властивостей:

- здатність працювати на звичайному смартфоні у режимі реального часу;
- можливість працювати з дошками різного кольору;
- можливість працювати з рухливою камерою;
- мінімальна кількість дефектів на слайдах, таких як наявність фрагментів викладача або видимі шви у місцях склейки кадрів.

На поточний момент жоден аналог не може такого дати, тому була поставлена задача створити таку технологію.

### **Мета і завдання дослідження.**

*Об'єкт дослідження* — відеозаписи лекцій.

*Предмет дослідження* — автоматична обробка відеозаписів.

Метою роботи є розробка алгоритму, що перетворює відеозаписи лекцій лекції у панорамні знімки без викладача.

Завдання наступні:

- 1) вивчити та доповнити математичні методи, що допоможуть визначати рухомі об'єкти;
- 2) опрацювати літературу про згорткові нейронні мережі, які застосовуються для знаходження в кадрі людини;
- 3) вивчити методи обробки та порівняння зображень для створення панорамних знімків;
- 4) розробити демонстраційне програмне забезпечення.

**Методи дослідження:**

- 1) опрацювання літератури за темою;
- 2) створення теоретичного підґрунтя алгоритму;
- 3) написання програми;
- 4) аналіз отриманих результатів.

**Наукова новизна одержаних результатів.**

Створено алгоритм та реалізовано інформаційну технологію для одержання панорамних слайдів з відеозапису лекції, який створено за допомогою рухливої камери, а дошку час від часу перекриває викладач та інші люди.

**Практичне значення одержаних результатів.**

За допомогою програми викладач може надати короткий вміст відео матеріалу. Наступними кроками є автоматичне детектування дошки, розбиття її на сектори та розпізнавання написаного на дошці тексту та формул.

**Публікації.** Стаття «Перетворення відеозапису з дошки у слайд-шоу» авторів Кригін В. М. та Шило М. К. прийнята до друку Міжнародним науковим журналом “Control Systems and Computers” (“Системи керування та комп’ютери”) та буде опублікована в № 2 (298) 2022 року.

## 1 АНАЛІЗ ПОПЕРЕДНІХ РОБІТ

У першому розділі розглядаються попередні роботи по створенню слайдів з відеозаписів або фотографій дошки. Аналізуються різні методи створення панорамних слайдів та алгоритми прибирання викладача з відео. Описуються переваги даної роботи над аналогічними та робиться постановка задачі створення панорамних слайдів.

### 1.1 Перша робота по оцифровуванню дошки

У 2004 році інженери з Microsoft Research Zhengyou Zhang та Li-wei He представили свій алгоритм по скануванню написів білої дошки [? ]. Система оброблює фотографії білої дошки, локалізує область написів, вирівнює у прямокутну форму дошку та бінаризує написи без втрати кольору (рис. 1.1).

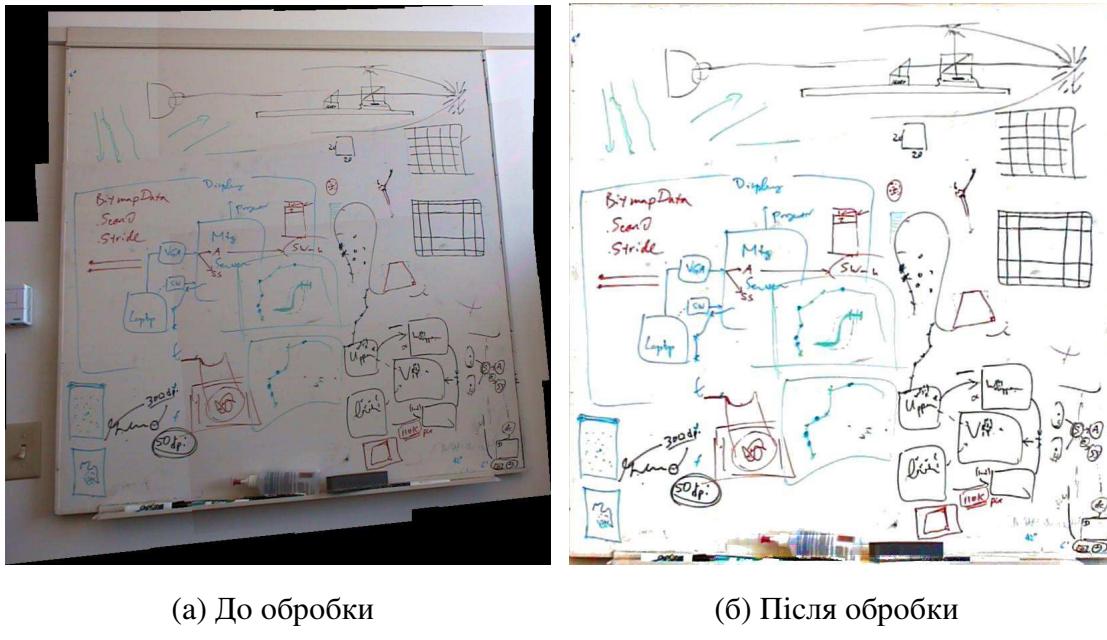


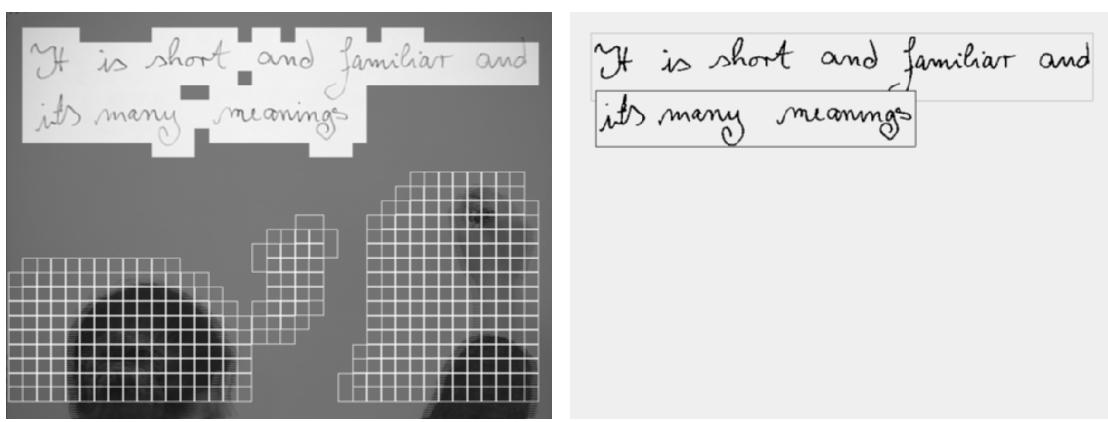
Рисунок 1.1 — Демонстрація роботи алгоритму інженерів з Microsoft [? ]

Автори реалізували склейку зроблених з різних ракурсів зображень дошки за допомогою гомографії. Гарна якість оцифрування дошки досягається насамперед тим, що вона має білий колір, що в свою чергу накладає обмеження на використан-

ня технології з дошками відмінного від білого кольорів. У наступній свої роботі [?] ці ж самі автори побудували технологію, яка в реальному часі оброблює відеозапис та видаляє людину біля дошки за допомогою часової медіани, але тут немає панорамного склеювання знімків. Головна ідея роботи полягала у розробці програми для телеконференцій.

## 1.2 Автоматичне сканування дошки

Автори роботи [?] створили програму, яка переводить написи на білій дощі у цифрові (рис. 1.2). Вони реалізували локалізацію тексту та подальшу його обробку. Даної технології не вирішує проблему перекривання викладачем написів, а також не дозволяє використовувати дошку, що має відмінний від білого колір.



(а) Детекція написів

(б) Обробка написів

Рисунок 1.2 — Демонстрація роботи сканування дошки [?]

Можна помітити (рис. 1.2), що, як і в попередній роботі, гарна якість виокремлення написів досягається тим, що дошка білого кольору.

## 1.3 Відстежування об'єкту та віднімання фону

У 2012 році науковці зі Стенфордського університету Alex Gonzalez, Bongsoo Suh, Eun Soo Choi представили технологію [?] локалізації дошки (навіть такої, яка

розділена на частини), відстеження викладача та його подальше прибирання. Алгоритм також може працювати з різними кольорами дошок. Для прибирання викладача і всіх рухомих об'єктів автори також використали часову медіану (рис. 1.3).

Дана програма не працює в реальному часі, оскільки всі операції над кадрами відео займають тривалий час, а також саме відео перед обробкою піддають компресії.

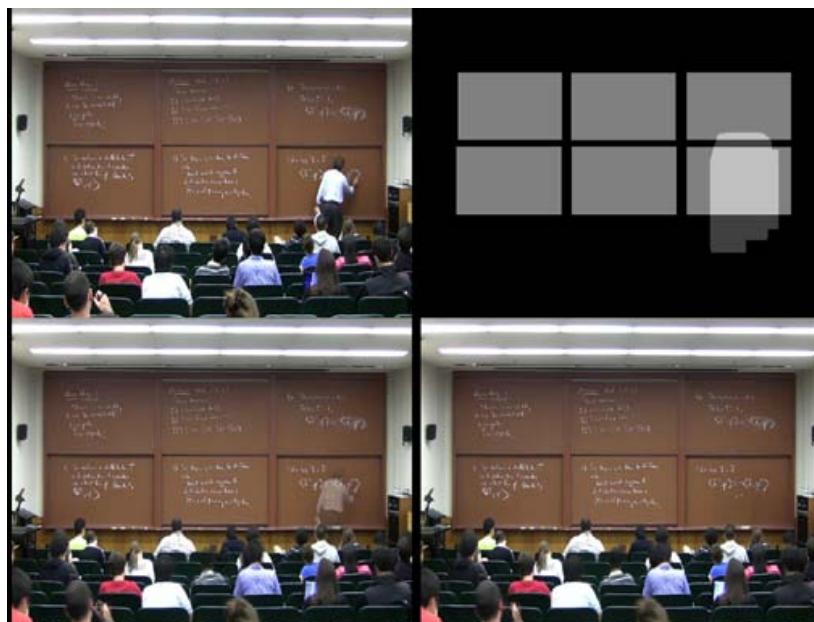


Рисунок 1.3 — Демонстрація роботи авторів зі Стенфордського університету [? ]

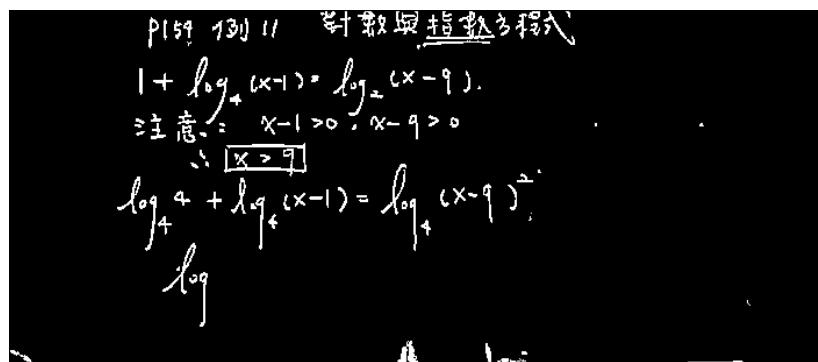
Головною особливістю даної роботи є те, що алгоритм автоматично локалізує різну кількість дошок. Однак, варто відмітити, що тестування відбувалось на відео лекціях, де камера знімає всю дошку і не рухається за викладачем.

#### 1.4 Відокремлення написів дошки

У 2014 році науковці з Тайванського університету представили свій алгоритм [?] оцифрування дошки (рис. 1.4). Для видалення викладача, автори застосували алгоритм кластеризації k-means. Для отримання бінаризованих написів з дошки використане адаптивне вирівнювання. Варто відмітити гарну якість власного методу зменшення шуму.



(a) До обробки



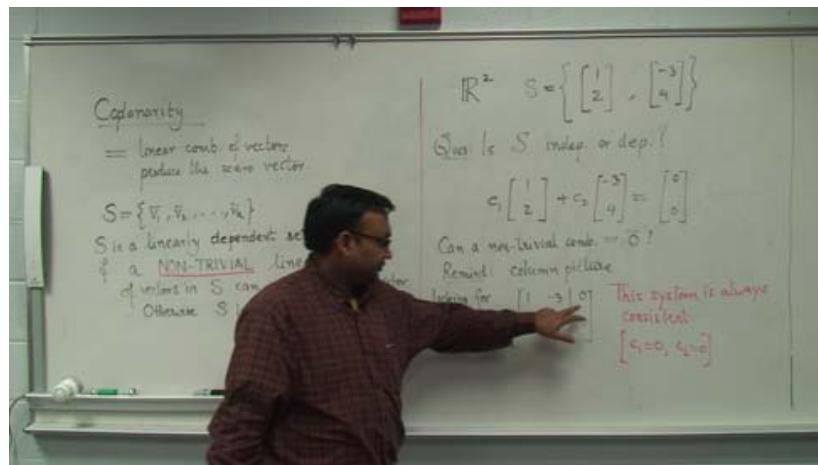
(b) Після обробки

Рисунок 1.4 — Демонстрація роботи сканування дошки [? ]

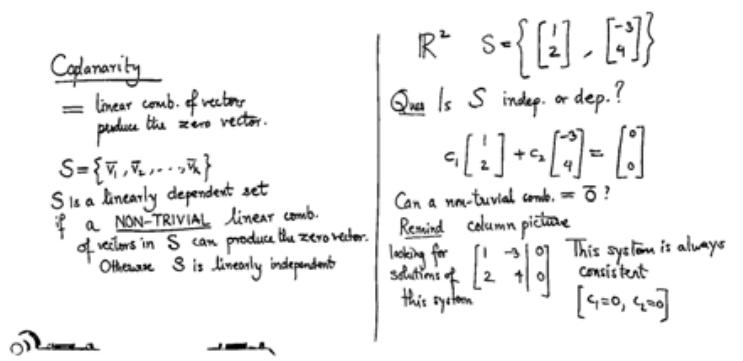
Автори не надали час обробки всього відео. Щоб отримати якісну сегментацію дошки та викладача, потрібно, щоб кадр містив не просто викладача, а викладача з кольором одягу сильно відмінним від кольору дошки. Це потрібно для коректної роботи алгоритму k-means. Тому немає гарантії, що якийсь рухомий об'єкт не буде класифікований як дошка під час класифікації.

## 1.5 Сучасна робота

Окремо зазначимо роботу [?] науковців з університету Рочестер. Автори Kenny Davila та Richard Zanibbi використали просторово-часовий індекс для виокремлення написів і викладача (рис. 1.5). Відбувається видалення не самого викладача, а його контурів після бінаризації картинки. Варто відмітити, що і тут камера має бути нерухомою.



(a) До обробки



(b) Після обробки

Рисунок 1.5 — Демонстрація роботи сканування дошки [? ]

Пізніше, ці ж автори створили повністю згорткову нейронну мережу [?] для обробки написів дошки. Дано нейронна мережа LectureNet досить добре бінаризує написи з дошки (рис. 1.6).

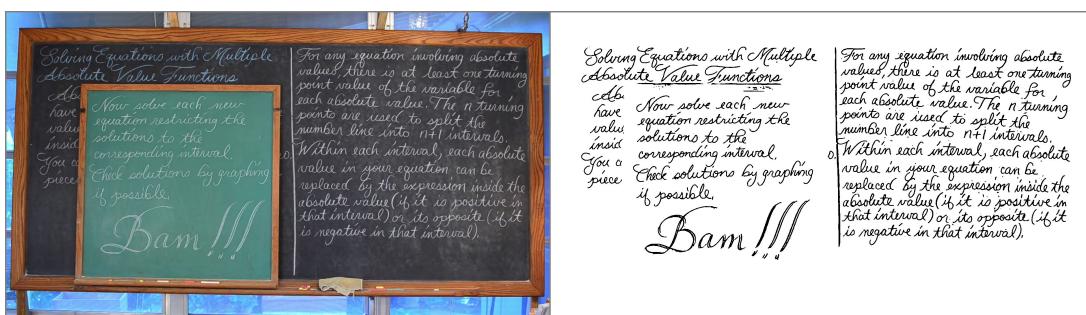


Рисунок 1.6 — Приклад роботи авторів з Рочестер [? ]

## Висновки до розділу 1

Як ми бачимо, всі вищеописані алгоритми не повністю вирішують поставлену нами задачу оцифрування дошки: деякі методи обробляють тільки написи з дошки, частина аналогів створює панорамні знімки тільки з фотографій дошки. Спільними елементами вищеописаних аналогів є використання темпорального медіанного фільтру для видалення рухомих об'єктів. У поточній роботі даний метод теж має місце, але вже для знешумлення вихідних слайдів, а не для видалення лектора з відео. Найбільшими недоліками всіх аналогів є час обробки відео. Більшість технологій тестувалась на відео з низькою роздільною здатністю від  $352 \times 240$  до  $1280 \times 720$ , хоча сучасні недорогі смартфони дозволяють записувати відео розмірами  $1920 \times 1080$ , а більш дорогі моделі навіть  $3840 \times 2160$ , що помітно позначиться на швидкодії існуючих алгоритмів. Також, жодний аналог не пропонує створювати панорамні знімки по мірі руху камери під час зйомки.

## 2 ПОСТАНОВКА ЗАДАЧІ СТВОРЕННЯ СЛАЙДІВ

У другому розділі описується постановка головних задач та можливих рішень для отримання панорамних знімків.

### 2.1 Постановка задачі

Головна ціль даної роботи — це створення власної інформаційної технології, на вхід якої буде подаватись відеозапис лекції, а на вихід панорамні слайди без викладача. Для одержання бажаних результатів було запропоновано вирішити такі проблеми:

- 1) Стабілізація кадрів. Багато відео лекцій записуються не в найкращих умовах. Камера може випадково затруситися, якщо її прикріпити до столу, де студенти пишуть лекцію, або може бути нестабільно закріплена. Трясіння камери може заважати сприймати лекцію і фокусуватись на написах дошки.
- 2) Прибирання викладача. Є чимало методів вирішення цієї задачі. Можна локалізувати тільки людину, а можна і всі об'єкти, що рухаються в площині дошки. Маючи дані про положення викладача в кадрі, ми матимемо змогу оновлювати з часом інформацію з дошки, яка була тимчасово перекрита.
- 3) Створення панорамних слайдів. Часто, коли дошки мають значну площину, камера знімає тільки ту частину, де викладач щось пише. Під час лекції камеру переміщують так, щоб лектор був завжди в полі зору. Таким чином, якщо студент не встиг переписати все з однієї частини дошки, а камеру перемістили, втрачається частина інформації. Тому корисно мати слайди, які по мірі руху камери містять всю дошку.
- 4) Оцифровування написів. Умови освітлення, розводи на дошці, неякісна зйомка, шуми — все це впливає на сприйняття лекційного матеріалу. Корисно мати бінаризовані слайди, які містять лише написи як в записнику.

## 2.2 Пропоновані рішення

У даній роботі описується алгоритм напівавтоматичного створення слайдів на основі відеозапису лекції (рис. 2.1). За допомогою наведеного алгоритму можна обробляти відео у режимі реального часу, адже кожна його ітерація потребує лише поточний на попередні кадри. Між кожною парою сусідніх кадрів відео розраховується бінарна маска областей, де потенційно знаходяться рухомі об'єкти або викладач (в залежності від методу створення маски).

Ті частини зображення, де не було помічено руху або людини, зберігаються до зображення, яке ми в даній роботі називаємо панорамою, тому що це зображення може розширюватись у випадку, коли камера рухається навмисно або хитається через небажаний вплив на неї. Отриману панорamu ми порівнюємо з тими панорамами, що були отримані на попередніх кроках, і створюємо новий слайд, якщо було помічено достатню кількість змін.

Інформаційна технологія, що реалізує описаний алгоритм, працює у напівавтоматичному режимі — вона потребує від користувача введення деяких параметрів, а саме:

- 1) крок або кількість кадрів між тими кадрами, які беруться в обробку;
- 2) ступінь згладжування маски рухомих об'єктів у випадку використання алгоритму Бойкова-Колмогорова;
- 3) рівень довіри присутності людини у випадку використання згорткових мереж;
- 4) крок між панорамами для перевірки необхідності створення нового панорамного слайду;
- 5) кількість необхідних змін між двома панорамами для створення нового слайду.



Рисунок 2.1 — Процедура створення панорамних слайдів

### 2.3 Вхід алгоритму

Для опису алгоритму створення панорамних слайдів потрібно визначити та описати об'єкти, з якими взаємодіє інформаційна технологія.

**Означення 1.** Дошка — це плоска поверхня, яку знімає камера. Це може бути крейдяна дошка, маркерна дошка, стіна тощо.

**Означення 2.** Записи — це ті місця дошки, де відбулася зміна кольору, яка тривала відносно довгий час.

Важливо зауважити, що ці записи мають бути саме у площині дошки, причому людина, яка проходить біля неї, не вважається зміною кольору, оскільки цей рух був не тривалим.

В область огляду камери має потрапляти дошка або її частина. Наведений алгоритм не розрахований на відео, що містить декілька дошок, які знаходяться не в одній плоскій площині. Камера, що знімає відео, може рухатись, проте чим більше вона нерухома, тим краще (алгоритм не буде працювати, якщо камера рухається постійно). Дошка на відео може перекриватися сторонніми об'єктами, проте бажано, щоб ці об'єкти були рухливими, щоб алгоритм виявлення рухомих об'єктів їх не додавав до слайдів.

**Означення 3.** Кадром під номером  $i$  ширину  $w$  і висотою  $h$  називаємо відображення  $P \rightarrow C$ , де  $P$  — множина координат пікселів,  $C$  — скінчена множина

можливих рівнів яскравостей (інтенсивностей) пікселів

$$P = [1, \dots, w] \times [1, \dots, h], C \subset R.$$

**Означення 4.** Відео  $F$  довжиною  $T$  є послідовністю  $(F^i : i = \overline{1, T})$  кадрів  $F^i : P \rightarrow C$ .

Той факт, що піксель з координатою  $p \in P$  на кадрі  $F^i$  має інтенсивність  $c \in C$ , позначатимемо  $F_p^i = c$ .

## **Висновки до розділу 2**

Зроблена постановка задачі створення слайдів. Описано ввідні дані алгоритму та запропонована його блок-схема. Таким чином, для створення інформаційної технології потрібно вивчити методи локалізації людини або рухливих об'єктів та зробити стабілізацію відео та панорамне склеювання кадрів.

### 3 МЕТОДИ ЛОКАЛІЗАЦІЇ ЛЮДИНИ ТА РУХОМИХ ОБ'ЄКТІВ

Третій розділ присвячено опису методів визначення області, де знаходиться людина або рухомі об'єкти, щоб у подальшому прибрести викладача з панорамних знімків. Описано будову алгоритму Бойкова-Колмогорова та згорткових мереж YOLO, MobileNet, SSD, R-CNN.

#### 3.1 Локалізація рухомих об'єктів

Розглянемо теоретичне підґрунтя методів, які використовуються в даній роботі для локалізації рухомих об'єктів, а саме задачу пошуку мінімального розрізу графа та її розв'язок за допомогою алгоритму Бойкова-Колмогорова.

##### 3.1.1 Задача знаходження мінімального розрізу графа

Задачу знаходження мінімального розрізу та еквівалентна їй задачу пошуку максимального потоку можна описати на прикладі системи трубопроводу. Є мережа труб, кожна з яких має свою пропускну здатність і напрям. У даній мережі є джерело (початок мережі) та стік (кінець мережі). Головна задача — знайти максимальний потік води, який може пройти з джерела у стік.

Якщо дану концепцію перекласти на мову математики, то мережею труб є орієнтований граф  $G$  (рис. 3.1) з множинами вузлів  $T$  та множиною направлених ребер  $\tau$ . В ньому є джерело  $s \in T$ , стік  $e \in T$ , пропускні здатності  $c_{tt'} \geq 0$  та потоки  $f_{tt'} \in \mathbb{R}$ . Також позначимо множини вихідних  $N_t = \{t' : tt' \in \tau\}$  та вхідних  $P_t = \{t : tt' \in \tau\}$  ребер для кожної вершини  $t$ .

**Означення 5.** Максимальний потік — це найбільша величина потоку, який можна пропустити через джерело з метою найефективнішого використання мережі.

**Означення 6.** Мінімальним розріз — це множина тих ребер, які стали насиченими

після знаходження максимального потоку. Мінімальний розріз розділяє множину ребер на дві множини.

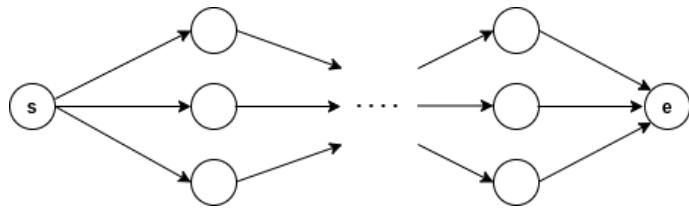


Рисунок 3.1 — Приклад графу

Сформулюємо задачу максимального потоку

$$\sum_{t \in N_s} f_{st} \rightarrow \max_{f: \tau \rightarrow R}$$

з обмеженнями

$$\begin{cases} f_{tt'} \leq c_{tt'}, & \forall tt' \in \tau, \\ \sum_{p \in P_t} f_{pt} - \sum_{t' \in N_t} f_{tt'} = 0, & \forall t \in T \setminus \{s, e\}, \\ \sum f_{tt'} \geq 0, & \forall tt' \in \tau. \end{cases}$$

Це означає, що

- 1) потік має не перевищувати пропускну здатність для всіх ребер;
- 2) сума потоків, що входять у вузол не повинна змінитись на виході;
- 3) потік завжди додатній.

Як вже було зазначено, задача пошуку максимального потоку є еквівалентною до задачі пошуку мінімального розрізу. Даний висновок отримується з Max-Flow Min-Cut теореми [? ? ].

У 2004 році Юрій Бойков та Володимир Колмогоров запропонували свій підхід [?] для пошуку мінімального розрізу на графі. Запропонована ідея методу полягає у нарощенні потоку у шляхах з джерела та стоку. Будується два дерева  $S$  та  $T$ , ко-

ренями яких є  $s$  та  $t$  відповідно. Вершини загально діляться на ті що в  $S$ ,  $T$  та вільні. Кожне дерево має активні та внутрішні вершини. Алгоритм Бойкова-Колмогорова (надалі Б-К алгоритм) складається зі стадій росту, доповнення та всиновлення. Коротко розглянемо кожну з них.

- 1) **Стадія Росту.** Проводимо одночасний ріст дерев з вершин  $s$  та  $e$ , знаходимо активні вершини і додаємо їх як вершини, що відвідали. Після такого сканування вершини стають внутрішніми. Цей процес продовжується поки не залишиться не активної вершини.
- 2) **Стадія Доповнення.** На цій стадії потік вздовж шляху, що був знайдений на попередньому етапі, доповнюється залишковою пропускною здатністю (англ. bottleneck). Якщо ребра дерева стають насиченими (пропускна здатність дорівнює потоку), то найвіддаленіші вершини від коренів дерев стають сиротами, тобто, якщо вершини  $t$  та  $t'$  додаються в множину  $S$  і ребро  $(t, t')$  є насиченим, тоді  $t'$  називається  $S$ -сиротою. Analogічно, якщо  $t$  та  $t'$  знаходяться в дереві  $T$ , то  $t$  —  $T$ -сирота. Якщо ребро знаходитьсь в bottleneck ( $t$  в  $S$ ,  $t'$  в  $T$ , ребро  $(t, t')$  насичене) відповідно немає ніяких сиріт. Всі сироти потрапляють у множину сирот.
- 3) **Стадія Всиновлення.** На даному етапі ми проходимось по кожній сироті в множині сирот для кожного дерева. Нехай  $t'$  є  $S$ -сиротою. Знаходимо всі такі  $t$  в  $S$ , що  $(t, t') \in E$ . Для кожного такого  $t$  перевіряємо, чи шлях з  $t$  в  $s$  містить сирот, включаючи  $t$ . Якщо сирот не знайшли, то  $t$  — батько  $t'$ . Якщо не вдається знайти батька, ми позначаємо вершину  $t'$  як вільну, а всіх дітей  $t'$  сиротами. Після цього ми оброблюємо залишкові ребра  $(t, t')$  і для кожного  $t$  в  $S$  позначаємо  $t$  активною.

Найбільшою перевагою даного алгоритму є те, що на практиці він працює дуже швидко на графах-решітках (рис. 3.2). Оскільки саме таку структуру ми використовуємо в алгоритмі Б-К для отримання маски рухомих об'єктів, це дає змогу оброблювати кадри відео досить швидко.

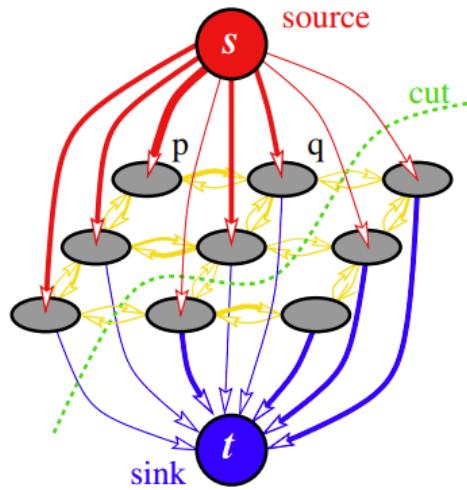


Рисунок 3.2 — Приклад графу-решітки з мінімальним розрізом [? ]

### 3.2 Застосування алгоритму Бойкова-Колмогорова

Окремо Б-К алгоритм не є корисним для детекції рухомих об'єктів. У даній роботі використано метод [? ], який у свою чергу використовує алгоритм Б-К для пошуку мінімального розрізу на графі, що дозволяє знаходити маску рухомих об'єктів та має можливість налаштовувати ступінь взаємодії сусідніх пікселів.

**Означення 7.** Маскою рухомих об'єктів кадру  $F^i$  будемо називати бінарне зображення  $B^i : P \rightarrow \{0,1\}$ , де тим пікселям, в яких на відповідному кадрі  $F^i$  було помічено рух, відповідає одиниця, а іншим відповідає нуль.

Для обраного користувачем кроку  $s \in \mathbb{N}$  на двох кадрах  $F^i$  і  $F^{i+s}$  рухомі об'єкти являють собою підмножину пікселів, колір яких було змінено більше, ніж на певне значення, з урахуванням зміни кольорів у сусідніх пікселях. Тобто, якщо рухомий об'єкт складається з одного пікселя, його рух може бути проігнорованім в залежності від обраних користувачем налаштувань, про які йдеться мова далі; аналогічно, якщо рухомий об'єкт на кадрі містить нерухомі «дірки» (пікселі, де колір не змінився), вони можуть вважатися частиною рухомого об'єкту.

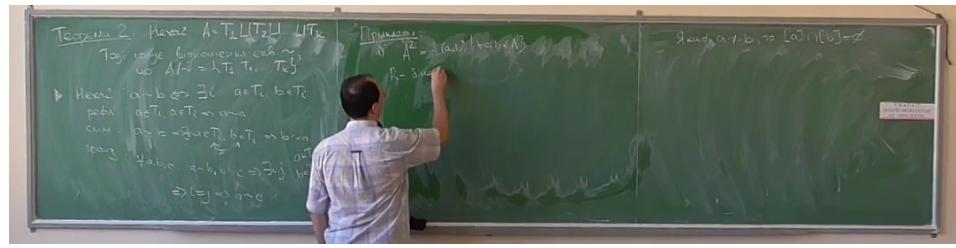
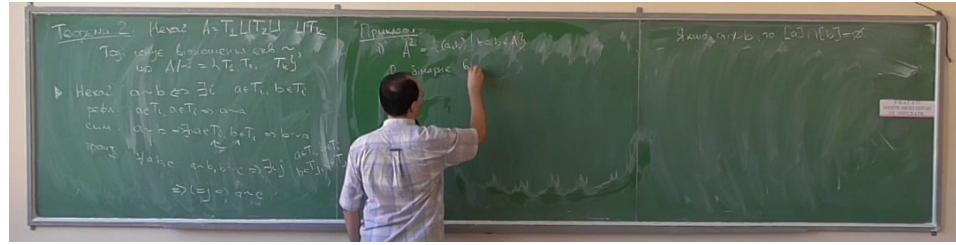
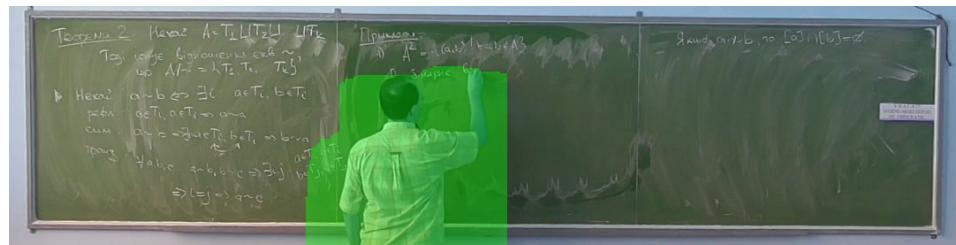
(а) Попередній кадр  $F^i$ (б) Поточний кадр  $F^{i+s}$ (в) Інвертована різниця  $F^i$  і  $F^{i+s}$ (г) Маска рухомих об'єктів на кадрі  $F^{i+s}$ 

Рисунок 3.3 — Процес створення маски рухомих об'єктів з відео [? ]

Для того, щоб видалити всі рухомі об'єкти, які можуть перекривати дошку, ми беремо кадри  $F^i$  та  $F^{i+s}$  (рис. 3.3, (а), (б)) та для кожної пари кольорів пікселів  $p$  з однаковими координатами знаходимо модуль  $D_p^i = |F_p^i - F_p^{i+s}|$  різниці інтенсивностей (рис. 3.3(в)). Зображення  $D^i$  подаємо на вхід Б-К алгоритму.

Знайдемо маску  $B^i : P \rightarrow \{0,1\}$  рухомих об'єктів для кадрів  $F^i$  і  $F^{i+s}$ . Введе-

мо функції.

$$q_p(B_p^i) = \begin{cases} \alpha D_p^i, & \text{якщо } B_p^i = 0, \\ 255 - D_p^i, & \text{якщо } B_p^i = 1, \end{cases}$$

$$g(B_p^i, B_{p'}^i) = \beta |B_p^i - B_{p'}^i|,$$

де  $\alpha$  та  $\beta$  — параметри згладжування маски, що задаються користувачем інформаційної технології. Позначимо множину  $\Gamma \subset P^2$  сусідніх пікселів. У даній роботі сусідніми до пікселя  $p \in P$  вважаються пікселі з множини  $\{(p_{x+1}, p_y), (p_x, p_{y+1})\} \cap P$ . Сформулюємо пошук маски  $B^i$  у вигляді задачі мінімізації виразу

$$E(B_p^i) = \sum_{p \in P} q_p(B_p^i) + \sum_{(p, p') \in \Gamma} g(B_p^i, B_{p'}^i).$$

На виході отримуємо маску  $B_p^i$  рухомих об'єктів (рис. 3.3(г)). Її ми використовуємо, щоб не переносити на фінальне зображення ті пікселі, на яких було помічено рух, адже зміна яскравості пікселя виникає не тільки під час створення напису, а й під час тимчасового затуляння дошки.

### 3.3 Згорткові нейронні мережі

**Означення 8.** Маскою людини  $F^i$  будемо називати бінарне прямокутне зображення  $M^i : P \rightarrow \{0,1\}$ , де тим пікселям, в яких на відповідному кадрі  $F^i$  була помічена людина, відповідає одиниця, а іншим відповідає нуль.

Для локалізації людини були використані згорткові нейронні мережі (англ. convolutional neural networks, CNN). В таких мережах застосовується операція згортки (англ. convolution) та пулінгу (англ. pooling), нормалізація пакетів (англ. batch normalization) та різні функції активації на кшталт ReLU, Tanh тощо.

Надалі комбінацію (згортка + нормалізація пакетів + функція активації) будемо називати згортковим шаром, але кожний автор нейронної мережі створює свої шари, що можуть відрізнятися від вищезазначеного.

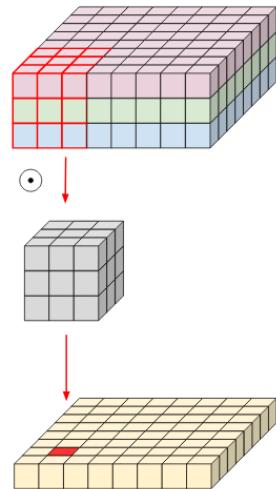


Рисунок 3.4 — Ілюстрація взяття згортки [? ]

Розглянемо архітектури нейронних мереж, які застосовувались у роботі для локалізації людини. Всі нищеописані нейронні мережі були використані вже з на тренованими вагами у програмній бібліотеці PyTorch [? ]. За мету було поставлено підібрати таку мережу, яка здатна швидко та якісно оброблювати один знімок навіть на смартфоні.

### 3.3.1 YOLO (You Only Look Once)

YOLO — це сімейство нейронних мереж, вперше представлене дослідником на ім'я Joseph Redmon [? ]. Мережа YOLO розв'язує задачу детекції об'єктів як задачу регресії щодо просторового розділення знайдених областей об'єктів та їх ймовірностей. Її зараз широко використовують для локалізації та класифікації об'єктів, оскільки вона здатна оброблювати відео в реальному часі з частотою 30 кадрів в секунду на мобільних пристроях, що є її найбільшою перевагою серед інших аналогів.

Опишемо коротко принцип роботи YOLOv1, оскільки YOLOv5, яка застосо-

вувалась в роботі, є її модифікацією.

- 1) Спочатку зображення розділяється решіткою  $S \times S$ . Якщо центр об'єкту потрапляє в комірку решітки, ця комірка є кандидатом, для подальшої локалізації об'єкта.
- 2) Кожна комірка решітки має передбачувати  $B$  областей та рівнів довіри. Рівень довіри показує, наскільки модель “впевнена”, що дана комірка містить об'єкт, та наскільки точна область. Рівень довіри  $t$  визначається як

$$t = Pr(\text{Object}) * IOU_{pred}^{truth},$$

де  $Pr(\text{Object})$  — ймовірність об'єкту, а  $IOU_{pred}^{truth}$  — величина перетину передбаченої області об'єкту до її справжньої. Відповідно, якщо модель не знайшла об'єкт, цей рівень нульовий. Необхідно, щоб  $t$  був якомога близчим до  $IOU_{pred}^{truth}$ .

- 3) Кожна область об'єкту складається з 5 чисел: рівень довіри,  $x, y$  (координати центру об'єкту),  $w, h$  (ширина та висота об'єкту). Кожна комірка передбачає  $C$  умовних ймовірностей  $Pr(\text{Class}_i | \text{Object})$ .

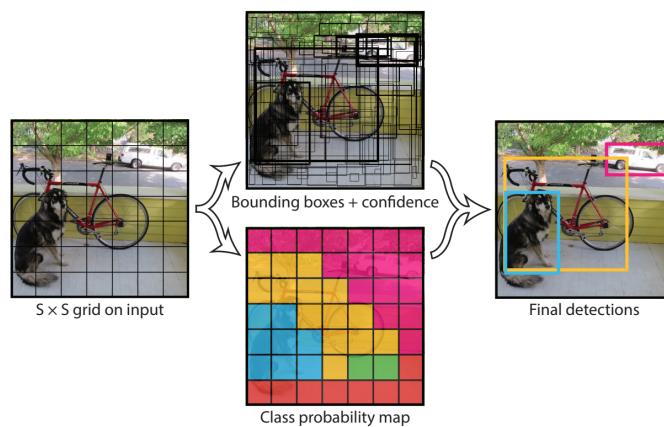


Рисунок 3.5 — Процес локалізації об'єктів мережею YOLOv1 [? ]

Для тренування мережі використовують суму 4 штрафних функцій.

$$\begin{aligned}
& \lambda_{coord} \underbrace{\left( \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{obj} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2] \right)}_{\text{по координатам центру}} \\
& + \underbrace{\sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{obj} [(\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2]}_{\text{ширина та висота об'єкту}} \\
& + \underbrace{\sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{obj} (C_i - \hat{C}_i)^2 + \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{noobj} (C_i - \hat{C}_i)^2}_{\text{точності класифікації}} \\
& + \underbrace{\sum_{i=0}^{S^2} \mathbb{1}_i^{noobj} \sum_{c \in classes} (p_i(c) - \hat{p}_i(c))^2}_{\text{ймовірності класів}}
\end{aligned}$$

де  $\mathbb{1}_i^{obj}$  показує, чи знайшовся об'єкт в комірці  $i$ , а  $\mathbb{1}_{ij}^{obj}$  показує, чи в комірці  $i$  в  $j$ -ій області знаходиться об'єкт.

Загалом архітектура нейронної мережі YOLOv1 складається з 24 згорткових шарів та 2 повнозв'язних лінійних шарів.

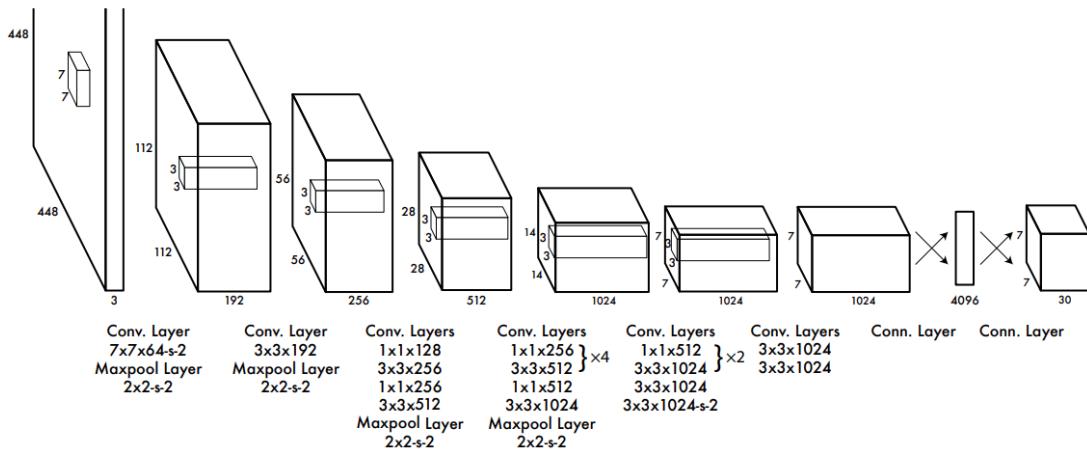


Рисунок 3.6 — Архітектура YOLOv1 [? ]

У даній роботі була використана одна з мереж YOLOv5, яка була розроблена

Glenn Jocher на програмній бібліотеці PyTorch.

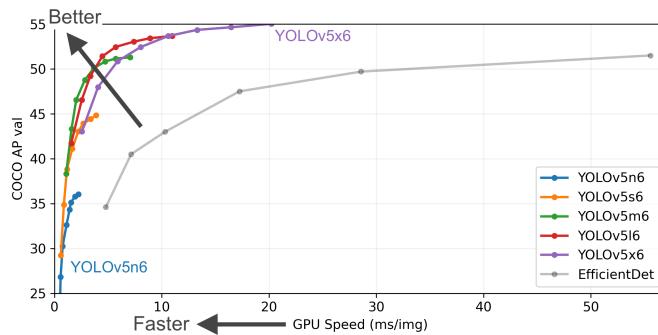


Рисунок 3.7 — Графік залежності точності на датасеті COCO від швидкості обробки однієї картинки різними мережами YOLOv5

### 3.3.2 MobileNet

MobileNet — це також ще одне сімейство, що часто використовується у комп’ютерному зорі. Вперше MobileNetv1 [?] була представлена у 2017 році науковцями з Google. Мережі даної категорії теж зробили свою революцію в обчисленні глибоких шарів з використанням мінімальних обчислювальних ресурсів. Були запропоновані два гіперпараметри, змінивши які, можна отримати приріст в швидкості або точності. Мережі MobileNet застосовуються для локалізації і класифікації об’єктів, а також для широкомасштабної гео-локалізації.

Розглянемо особливості різних версій MobileNet.

#### 3.3.2.1 MobileNetv1

Однією з головних задач для побудови першої мережі даного сімейства була заміна звичайного згорткового шару на новий глибинно-просторовий згортковий шар (англ. depth-wise separable convolution) (рис. 3.9).

Нехай на вході маємо

- квадратне зображення  $I$  розмірами  $S_I \times S_I \times M$ : ширина, висота та кількість

каналів відповідно;

- ядро  $C$  розмірами  $S_C \times S_C \times M \times N$ , де  $N$  — це вихідна розмірність отриманої згортки;
- вихідна згортка  $C$  розмірами  $S_K \times S_K \times N$ .

Тоді формулу звичайної згортки (рис. 3.4) можна записати як

$$G_{k,l,n} = \sum_{i,j,m} C_{i,j,m,n} \cdot F_{k+i-1,l+j-1,m}$$

. Для обчислення такої згортки потрібно  $S_C \cdot S_C \cdot M \cdot N \cdot S_I \cdot S_I$  операцій, що створює обчислювальні обмеження на мобільний пристрій, якщо використовувати декілька таких згорток. Для вирішення даної проблеми застосовується глибинна згортка (рис. 3.8). Вона полягає у використанні окремої згортки кожного каналу ядра до кожного каналу зображення. Нехай  $\hat{C}$  — ядро глибинної згортки. Тоді

$$\hat{G}_{k,l,n} = \sum_{i,j,m} \hat{C}_{i,j,m,n} \cdot F_{k+i-1,l+j-1,m}. \quad (1)$$

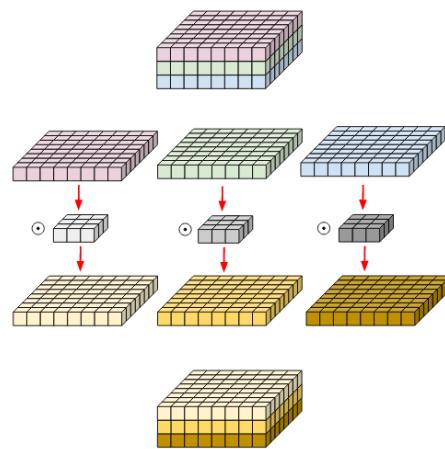


Рисунок 3.8 — Ілюстрація взяття глибинної згортки [? ]

Для обчислення такої згортки потрібно  $S_C \cdot S_C \cdot M \cdot S_I \cdot S_I$  операцій. Ми вже позбулися  $N$  операцій, але маємо пам'ятати, що зараз  $\hat{G}$  складається з  $M$  окремих

вихідних згорток. тому, щоб створити єдиний вихід, додатково до глибинної застосовують ще й точкову згортку (англ. point-wise convolution), в якій розмір ядра  $1 \times 1$ . Тоді маємо  $S_C \cdot S_C \cdot M \cdot S_I \cdot S_I + M \cdot N \cdot S_I \cdot S_I$  операцій. Данна комбінація має назву глибинно-просторова згортка, для обчислення якої потрібно в  $1/N + 1/S_C^2$  менше операцій, ніж для звичайної.

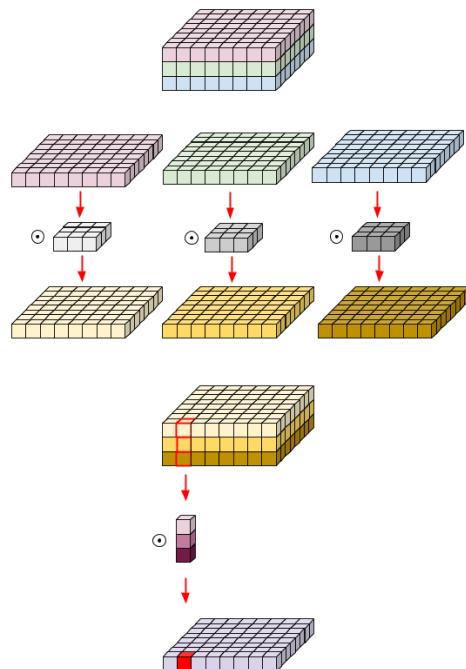


Рисунок 3.9 — Ілюстрація взяття глибинно-просторової згортки [? ]

Таке нововведення в галузі глибокого навчання дало змогу в рази пришвидшити навчання та роботу не лише згорткової мережі MobileNetv1, а й інших. У MobileNetv1 застосовуються просторово глибинні згортки з розміром ядра  $3 \times 3$ . Маємо таку заміну блоку, як показано на рис. 3.10.

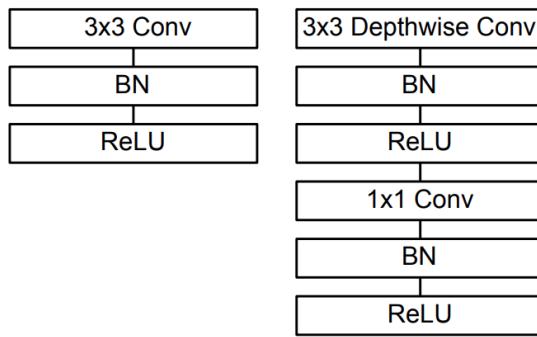


Рисунок 3.10 — Ліворуч звичайний згортковий шар, праворуч згортковий шар у MobileNetv1 [? ]

MobileNetv1 має також два гіперпараметра ширини та розмірності. Множник ширини  $\alpha$  застосовується, щоб зменшити кожен шар мережі, що в свою чергу дає приріст у швидкості. Із множником  $\alpha \in (0,1]$  потрібно буде зробити  $S_C \cdot S_C \cdot \alpha M \cdot S_I \cdot S_I + \alpha M \cdot \alpha N \cdot S_I \cdot S_I$  операцій. Множник розмірності  $\rho$  зменшує вхідну картинку і відповідно розмірність згорток. Разом із  $\alpha$  та  $\rho \in (0,1]$  необхідно буде  $S_C \cdot S_C \cdot \alpha M \cdot \rho S_I \cdot \rho S_I + \alpha M \cdot \alpha N \cdot \rho S_I \cdot \rho S_I$  операцій.

На рис 3.11 можна побачити повну архітектуру мережі MobileNetv1.

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5 \times$ Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool $7 \times 7$	$7 \times 7 \times 1024$
FC / s1	$1024 \times 1000$	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

Рисунок 3.11 — Архітектура нейронної мережі MobileNetv1 [? ]

### 3.3.2.2 Mobilenetv2

Вже у 2018 році з'являється 2-га версія мережі MobileNet. У ній автори додали нові зміни в архітектуру, такі як інвертовані залишки (англ. inverted residuals) та лінійні вузькі місця (англ. linear bottlenecks).

Автори зробили дослідження щодо застосування функції ReLU (rectified linear unit) в контексті низьких розмірностей. Вводиться поняття різноманітності інтересів (англ. manifold of interest), яке автори пояснюють як множину шарів активацій. Була висунута гіпотеза про те що, різноманітність інтересів з високою розмірністю можна стиснути у підпростір меншої розмірності зі збереженням інформації.

Розпишемо детальніше кроки, з яких складається новий структурний шар.

- 1) Нехай на вхід першого розширюючого блоку подається зображення розмірами  $S_I \times S_I \times M$ . Головна особливість цього блоку — це новий параметр  $t$ , що називається розширючим фактором. Найкращими значеннями для нього є  $[5,10]$ . Менші значення краще застосовувати для меншої мережі, а більші відповідно для більших. Саме тут використовується різноманітність інтересів. Вихід блоку розміром  $S_I \cdot S_I \cdot (t \cdot M)$ .
- 2) Наступним кроком є глибинна згортка з функцією активації, яка обчислюється за формулою  $ReLU_6(x) = \min(\max(0,x), 6)$ . Вхід розміром  $S_I \cdot S_I \cdot (t \cdot M)$ , а вихід  $(S_I / \text{stride}) \cdot (S_I / \text{stride}) \cdot (t \cdot M)$ , де  $\text{stride}$  — крок згортки.
- 3) Далі застосовується точкова згортка, щоб створити єдиний тензор. На рис. 3.12 наведено використання residual block, де вхід у шар поєднується з передостаннім блоком.

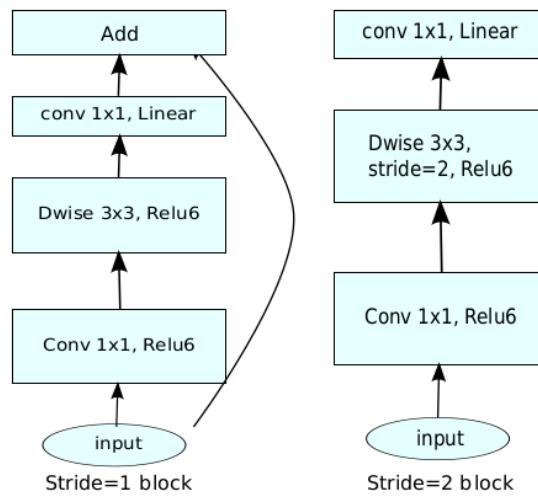


Рисунок 3.12 — Відмінність MobileNetv2 від MobileNetv1: ліворуч структурний блок MobileNetv2, праворуч - MobileNetv1 [? ]

Input	Operator	<i>t</i>	<i>c</i>	<i>n</i>	<i>s</i>
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	<i>k</i>	-	

Рисунок 3.13 — Архітектура мережі MobileNetv2 [? ]

Варто відмітити, що автори вдосконалили мережу SSD [? ] шляхом заміни звичайних згорткових шарів на розширюючі, які використовуються у MobileNetv2. Таким чином створили нову мережу SSDLite [? ].

### 3.3.2.3 MobileNetv3

На сьогодні останньою є третя версія архітектури сімейства нейронних мереж MobileNet [? ]. Тут автори представили вже дві нейронні мережі MobileNetv3-Small та MobileNetv3-Large. Це зроблено для того, щоб використовувати модель на слабких і потужних пристроях. Як запевняють автори, MobileNetV3-Small на 6.6 % то-

чніша за MobileNetv2, а локалізація об'єктів з MobileNetV3-Large на COCO датасеті на 25 % швидша з тією ж точністю. Головний блок мережі знову змінився. Розробники взяли до уваги певні особливості мережі MnasNet, в якій є блок стиснення та збудження (англ. squeeze and excitation). Мережі з такими блоками називаються Se-Nets.

Мета підходу стиснення та збудження (рис. 3.14) полягає в тому, щоб взяти параметри виходу згортки ( $u_c$  розмірами  $C \times H \times W$ ) як вхід блоку стиснення та збудження, потім зробити операцію стиснення (маємо  $1 \times 1 \times C$ ), операцію збудження ( $1 \times 1 \times C$ ) та в кінці масштабувати параметри. Ідея полягає в тому, щоб підвищити чутливість мережі до інформативних ознак, і передавати отриману інформацію наступному шару.

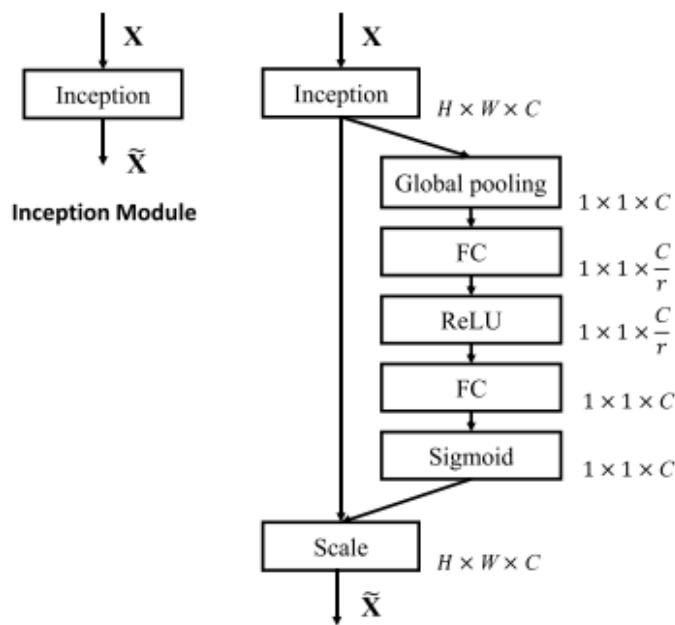


Рисунок 3.14 — Блок стиснення та збудження [? ]

Опишемо блок стиснення та збудження.

- 1) Операція стиснення полягає у відокремленні глобальної інформації з кожного каналу вхідного зображення. Данна операція є краща за звичну згортку, оскільки забирає всю інформацію з каналу зображення за один раз. Також, операція стиснення відома під назвою глобального середнього пулінгу (англ. global

average pooling), яка означає взяття середнього значення по кожному каналу.

$$z = F_{sq}(u_c) = \frac{\sum_{i=0}^H \sum_{j=0}^W u_c(i,j)}{H \cdot W}.$$

Вихід розміром  $1 \times 1 \times C$ .

- 2) Операція збудження створює множину ваг для кожного каналу шляхом застосування активацій *Sigmoid* та *ReLU* для двох повнозв'язних лінійних шарів.

$$s = F_{ex}(z, W) = \text{Sigmoid}(FC_2 \text{ReLU}(FC_1 z))$$

Перший лінійний шар  $FC_1$  використовується для зменшення розмірності  $z$  з деяким множником  $r$ , тому після нього розмір тензора  $1 \times 1 \times C/r$ , а  $FC_2$  навпаки для збільшення  $\text{ReLU}(W_1 \cdot z)$ . Знаючи, що значення сигмоїди від 0 до 1, можна масштабувати її вихід та поєднати із входом в блок стиснення і збудження.

$$\tilde{x}_c = F_{scale}(u_c, s_c) = s_c \cdot u_c$$

Автори покращили даний підхід у своїй MobileNetv3, але з використанням різної нелінійності на кожному шарі. Тут мається на увазі заміна лінійних функцій активацій. Було вирішено замість нелінійної функції

$$\text{swish}(x) = x \cdot \text{Sigmoid}(x)$$

використати

$$h - \text{swish}(x) = x \cdot \frac{\text{ReLU6}(x + 3)}{6}.$$

Це пов'язано з тим, що сигмоїда для великих тензорів складна в обчисленні для малопотужних пристройів. Автори помітили, що використання  $h - \text{swish}(x)$  дає приріст в точності, якщо її використовувати у глибоких шарах мережі.

Так само як і у MnasNet [?], автори MobileNetv3 використали платформу NAS (neural architecture structure), яка створена для підбирання глобальних параметрів мережі. Тобто NAS рекомендує найкращу знайдену архітектуру, а далі NetAdapt [? ] (схожа до NAS) допомагає у підборі параметрів вже всередині одного обчислювального блоку.

Всі вищеописані техніки допомогли створити MobileNetv3-Small та MobileNetv3-Large (рис. 3.15).

Input	Operator	exp size	#out	SE	NL	s
$224^2 \times 3$	conv2d	-	16	-	HS	2
$112^2 \times 16$	bneck, 3x3	16	16	-	RE	1
$112^2 \times 16$	bneck, 3x3	64	24	-	RE	2
$56^2 \times 24$	bneck, 3x3	72	24	-	RE	1
$56^2 \times 24$	bneck, 5x5	72	40	✓	RE	2
$28^2 \times 40$	bneck, 5x5	120	40	✓	RE	1
$28^2 \times 40$	bneck, 5x5	120	40	✓	RE	1
$28^2 \times 40$	bneck, 3x3	240	80	-	HS	2
$14^2 \times 80$	bneck, 3x3	200	80	-	HS	1
$14^2 \times 80$	bneck, 3x3	184	80	-	HS	1
$14^2 \times 80$	bneck, 3x3	184	80	-	HS	1
$14^2 \times 80$	bneck, 3x3	480	112	✓	HS	1
$14^2 \times 112$	bneck, 3x3	672	112	✓	HS	1
$14^2 \times 112$	bneck, 5x5	672	160	✓	HS	2
$7^2 \times 160$	bneck, 5x5	960	160	✓	HS	1
$7^2 \times 160$	bneck, 5x5	960	160	✓	HS	1
$7^2 \times 160$	conv2d, 1x1	-	960	-	HS	1
$7^2 \times 960$	pool, 7x7	-	-	-	-	1
$1^2 \times 960$	conv2d 1x1, NBN	-	1280	-	HS	1
$1^2 \times 1280$	conv2d 1x1, NBN	-	k	-	-	1

Input	Operator	exp size	#out	SE	NL	s
$224^2 \times 3$	conv2d, 3x3	-	16	-	HS	2
$112^2 \times 16$	bneck, 3x3	16	16	✓	RE	2
$56^2 \times 16$	bneck, 3x3	72	24	-	RE	2
$28^2 \times 24$	bneck, 3x3	88	24	-	RE	1
$28^2 \times 24$	bneck, 5x5	96	40	✓	HS	2
$14^2 \times 40$	bneck, 5x5	240	40	✓	HS	1
$14^2 \times 40$	bneck, 5x5	240	40	✓	HS	1
$14^2 \times 40$	bneck, 5x5	120	48	✓	HS	1
$14^2 \times 48$	bneck, 5x5	144	48	✓	HS	1
$14^2 \times 48$	bneck, 5x5	288	96	✓	HS	2
$7^2 \times 96$	bneck, 5x5	576	96	✓	HS	1
$7^2 \times 96$	bneck, 5x5	576	96	✓	HS	1
$7^2 \times 96$	conv2d, 1x1	-	576	✓	HS	1
$7^2 \times 576$	pool, 7x7	-	-	-	-	1
$1^2 \times 576$	conv2d 1x1, NBN	-	1024	-	HS	1
$1^2 \times 1024$	conv2d 1x1, NBN	-	k	-	-	1

(a) MobileNetv3-Large

(б) MobileNetv3-Small

Рисунок 3.15 — Дві архітектури мережі MobileNetv3 [? ]

### 3.3.2.4 SSD (Single Shot Multibox Detector)

Оскільки для відокремлення ознак в цій роботі застосовується SSD зі структурними елементами MobileNetv3, покажемо, як саме SSD вдається локалізувати об'єкти, та його відмінності від YOLO. Мережа SSD (рис. 3.16), на відміну від YOLO, пропускає зображення через шари лише один раз. Це пояснюється словами в назві single shot. SSD використовує задачу MultiBox регресії локалізованих областей, тому розпишемо її детальніше.

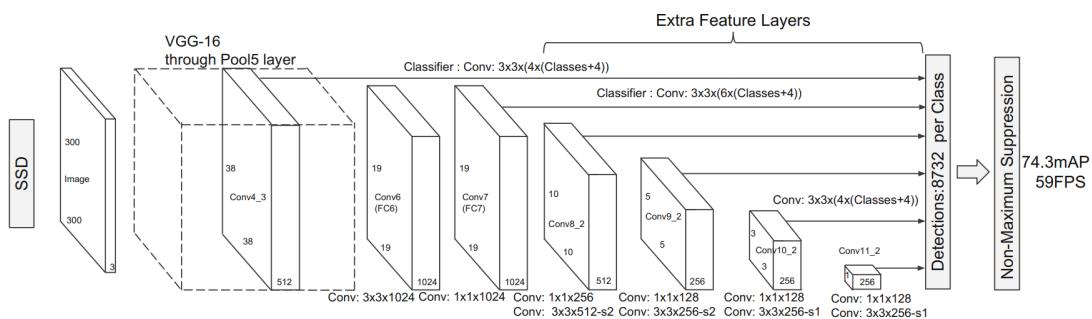


Рисунок 3.16 — Архітектура мережі SSD на основі VGG-16 [? ]

Головна задача MultiBox — оцінити, як локалізувати об'єкти точно. Для цього використовуються дві штрафні функції. Функція довіри, в основі якої лежить категоріальна перехресна ентропія, та функція локалізації з  $L_2$  нормою, яка показує, наскільки добре справжня область об'єкту співпадає зі спрогнозованою. SSD застосовує фіксовані області для подальшого передбачення зі згортками малого ядра. Чим більше фіксованих областей, тим більша точність локалізації об'єкту, але тим довша обробка фото.

Принцип відокремлення інформативних ознак SSD полягає у розбитті зображення на решітку (рис. 3.17) (аналогічно YOLO). Запускається класифікатор і вирішує, чи брати ту чи іншу клітинку у якості кандидата на локалізацію. Для кожної фіксованої області обчислюється розмір області та рівень довіри для кожної категорії об'єктів.

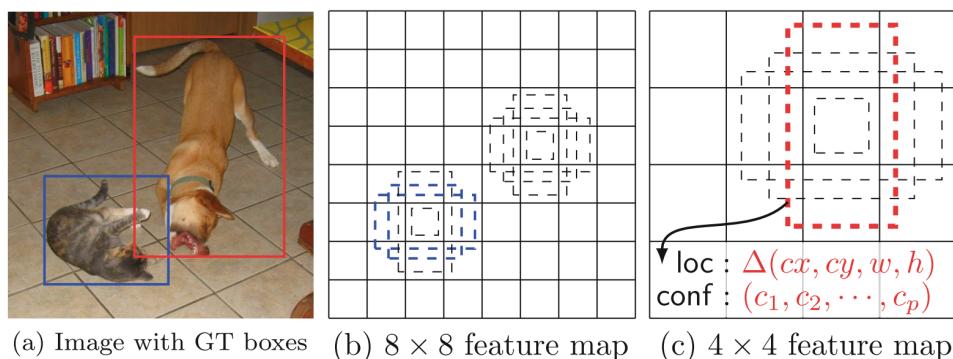


Рисунок 3.17 — Приклад роботи SSD для локалізації об'єктів [? ]

### 3.3.2.5 Faster R-CNN

Говорячи про детекцію і класифікацію об'єктів, не можна обійти увагою інший відомий клас згорткових нейронних мереж, що має назву R-CNN (region-based convolutional neural network).

Перша мережа R-CNN складалась з 3 незалежних етапів. На першому генерується 2000 пропозицій областей за допомогою вибіркового пошуку. Далі ці області проходять стиснення розміру до наперед заданого. Останній етап — це метод опорних векторів із заздалегідь натренованими вагами, який і проводить класифікацію.

R-CNN не була досить потужною, оскільки використовувала вибірковий пошук, який займає чимало часу. Також потрібно зберігати чимало кешованих даних для натренованої мережі.

Багато проблем було вирішено вже з новою Fast R-CNN. Вся архітектура складається з одного модуля, що в рази полегшує навчання. Тут з'являється новий шар ROI Pooling, головна ціль якого — надати вектори ознак фіксованої довжини. Даний шар розбиває кожний запропонований регіон на решітку, в кожній клітинці потім ми знаходимо максимальне значення (операція max pooling). Але в Fast R-CNN досі лишився вибірковий алгоритм.

У Faster R-CNN [?] (який теж тестувався для локалізації людини в цій роботі) автори використали RPN (Region Proposal Network) як спосіб генерації областей-кандидатів та Fast R-CNN для виявлення об'єктів в цих областях. Ці два етапи поєднуються в одну мережу за допомогою сумісного використання ознак (feature sharing). RPN приймає на вхід картинку і повертає множину координат прямокутних областей (кандидатів для класифікації) з мірами в них присутності об'єкту. RPN — повнозв'язана згорткова мережа, що означає, що в ній немає лінійних шарів. Саме вона стала заміною вибіркового алгоритму у Fast R-CNN.

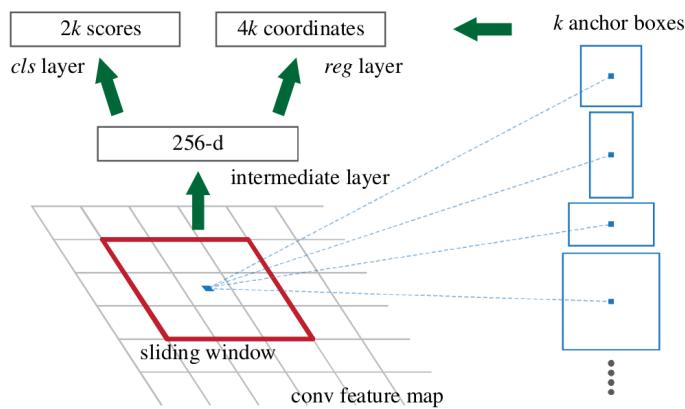


Рисунок 3.18 — Архітектура мережі RPN [? ]

RPN використовує принцип ковзного вікна (sliding window) на згортковій мапі ознак, отриманої після останнього згорткового шару. Кожне таке вікно відображається на вектор меншої розмірності. Для кожного вікна генеруються прямокутні області-кандидати, що називаються опорними регіонами (anchor boxes) з параметрами масштабу та співвідношення сторін. Далі цей вектор подається на вхід двом повнозв'язним шарам локалізації об'єктів (box-regression layer) та класифікації (box-classification layer). Використання опорних регіонів дозволяє детектувати об'єкти практично будь-якого масштабу та пов'язувати ознаки RPN з Fast R-CNN.

Як вже було сказано, за допомогою принципу поширення ознак можна використовувати вихід RPN як вхід в Fast R-CNN. Головна його ідея полягає у використанні одних і тих же згорток у двох мережах, що дозволяє тренувати RPN разом з Fast R-CNN, а не окремо.

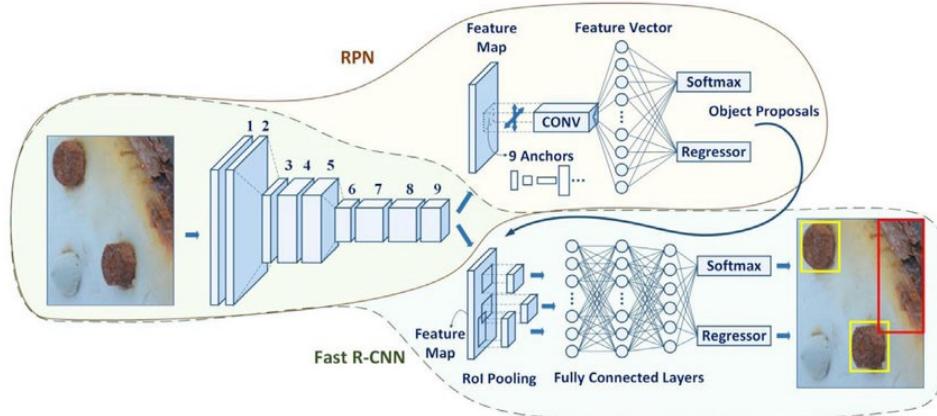


Рисунок 3.19 — Архітектура мережі Faster R-CNN [? ]

## **Висновки до розділу 3**

Було розглянуто алгоритм Бойкова-Колмогорова для отримання маски рухомих об'єктів. Також описано згорткові мережі YOLO, MobileNet, SSD, R-CNN як спосіб детектування людини. Описано архітектури, переваги та недоліки кожної згорткової мережі. Дані методи підходять для створення інформаційної технології побудови слайдів на мобільних пристроях.

## 4 СТВОРЕННЯ ПАНОРАМИ

У четвертому розділі описується пошук відповідності між кадрами, матриця гомографії та створення панорамного знімку без викладача.

### 4.1 Пошук відповідності між кадрами

Рух камери не є рідкістю для відеозаписів лекцій. Камера може труситись, коли її прикріплено до столу, де студенти пишуть лекцію, або від вібрацій полу, коли викладач ходить по ньому. Камеру може рухати оператор для того, щоб сфокусувати увагу глядачів на певному сегменті дошки. У даному розділі описано оцінку гомографії за допомогою дескриптора ознак SIFT як одного з кроків вирішення вищеписаних проблем.

Оскільки дошка вважається плоскою поверхнею, побудувати відповідність між точками дошки на різних зображеннях можна за допомогою гомографії. Рухи камери під час лекції можуть змінюватися від незначних субпіксельних зсувів до зміщень, в результаті яких видимою стає частина дошки, яка до цього була прихованою. Наша мета — зробити слайди, де камера виглядає статичною, а сегменти дошки поступово стають видимими та поєднуються для утворення панорами.

#### 4.1.1 SIFT

У 1999 році англієць Девід Лоу представив алгоритм SIFT (scale-invariant feature transform) [?] (укр. масштабонезалежне перетворення ознак). Даний алгоритм застовується для знаходження ключових точок (локальних ознак) (англ. feature points, keypoints) зображення. Метод досить точно і швидко знаходить дані точки. Головною перевагою алгоритму є інваріантність щодо просторової орієнтації та якості освітлення. Має широке застосування в області комп’ютерного зору як один з кро-

ків для побудови 3D-карт, ректифікації стереопари та виявлення об'єктів.

Ми можемо також використовувати дескриптори ознак, такі як SURF [?] або ORB [?], щоб знайти ключові точки, але був обраний SIFT, оскільки під час експериментів він надав візуально кращі результати, ніж інші алгоритми.

Коротко розпищемо структуру алгоритму.

- 1) Пошук масштабно-просторових екстремумів. На даному етапі потрібно знайти зони зображення та такі масштаби, які можна повторно знайти при різних перспективах (точок погляду). Тут використовується просторове та масштабно інваріантне ядро оператора Гаусса з операцією згортки до зображення

$$L(x,y,\sigma) = G(x,y,\sigma) * I(x,y),$$

де

$$G(x,y,\sigma) = (1/2\pi\sigma^2) \exp(-(x^2 + y^2)/2\sigma^2).$$

Будується різниця Гауссіан (DoG метод) (рис. 4.1) з константою  $k$

$$D(x,y,\sigma) = (L(x,y,k\sigma) - L(x,y,\sigma)).$$

Тобто початкове зображення поступово піддається згорткам гауссіанів з константою  $k$  на кожну епоху. Причому епохи відрізняються між собою значенням  $k = 2^{1/s}$ , де  $s \in \mathbb{N}$  показує на скільки інтервалів розділити кожну епоху. В стеку однієї епохи зберігається  $s + 3$  зображень. Коли епоха завершується, створюється нове гауссове зображення з двох найвищих в стеку шляхом взяття кожного 2-го пікселя рядка та колонки.

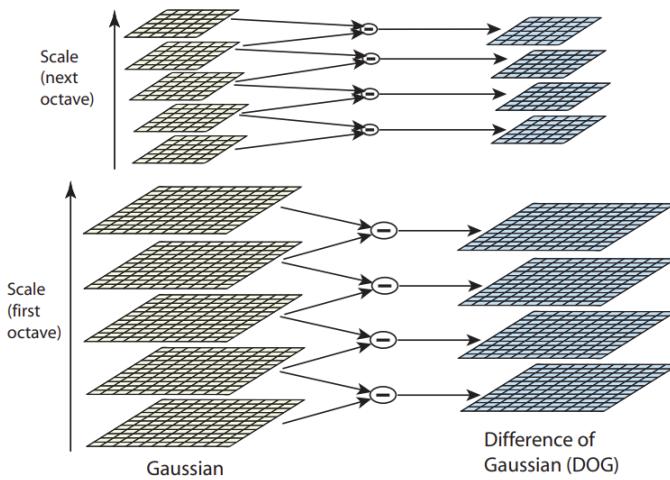


Рисунок 4.1 — Знаходження різниць Гауссіанів 1-ий етап SIFT [? ]

Пошук локальних екстремумів. Для знаходження локальних екстремумів (рис. 4.2) зображення різниці гауссіан якоїсь точки беруться 8 сусідів цього ж зображення і 9 зображення різниць зверху та знизу. Точка є екстремумом, якщо її значення більше або менше за значення всіх її сусідів.

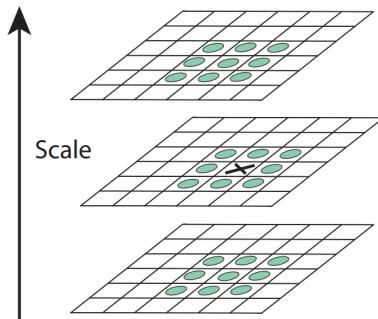


Рисунок 4.2 — Пошук локальних екстремумів: 2-ий етап SIFT [? ]

Частота вибірки по масштабу. Тут автор обґрунтуете кількість того, скільки разів потрібно робити зміну масштабу зображення. Експерименти показують (рис. 4.3), що на даному етапі метод дає велику кількість кандидатів екстремумів, і що дуже важко обрахувати їх всіх.

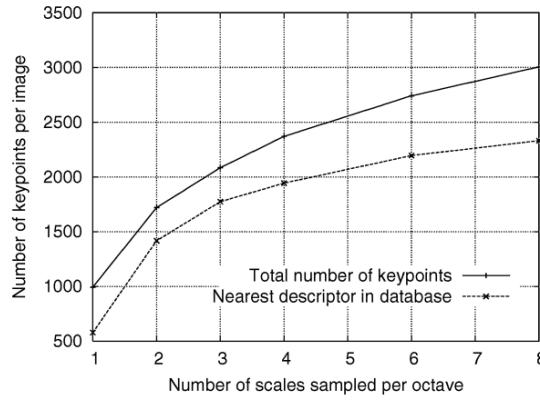


Рисунок 4.3 — Графік залежності кількості змін масштабу на кожну епоху до кількості ключових точок на зображенні [? ]

Частота вибірки по простору. Пропонується використовувати оптимальне значення  $\sigma = 1.6$ , при якому маємо найбільший відсоток збігів екстремумів при багаторазовому повторенні експерименту.

- 2) Точна локалізація ключових точок. Після знаходження точок-кандидатів потрібно відсіяти точки зі слабким контрастом на основі інформації про масштаб та відношення головних викривлень. Для цього застосовується розклад Тейлора масштабно-просторової функції  $D(x,y,\sigma)$  в точці  $x = (x,y,\sigma)$ .

$$D(\mathbf{x}) = D + \frac{\partial D^T}{\partial \mathbf{x}} \mathbf{x} + \frac{1}{2} \mathbf{x}^T \frac{\partial^2 D}{\partial \mathbf{x}^2} \mathbf{x}. \quad (2)$$

Місце де знаходиться екстремум  $\hat{\mathbf{x}}$ ,

$$\hat{\mathbf{x}} = \frac{\partial^2 D^{-1}}{\partial \mathbf{x}^2} \frac{\partial D}{\partial \mathbf{x}}. \quad (3)$$

Підставивши (3) у (2), маємо можливість відсіяти нестабільні екстремуми по модулю значення (рис. (4.3))

$$D(\hat{\mathbf{x}}) = D + \frac{1}{2} \frac{\partial D^T}{\partial \mathbf{x}} \hat{\mathbf{x}}.$$

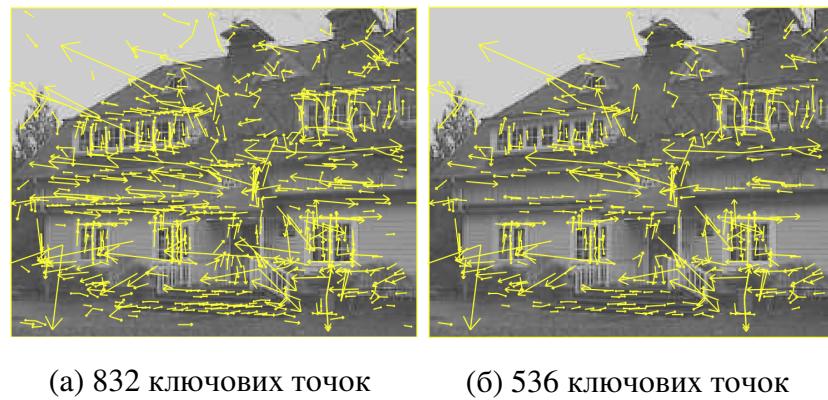


Рисунок 4.3 — Приклад відсіювання екстремумів [? ]

Таким чином ми обмежуємо  $|D(\hat{x})| < \alpha$ . Якщо кожен піксель в діапазоні  $[0,1]$ , то і  $\alpha \in [0,1]$ .

- 3) Визначення орієнтації градієнтів. Доожної точки визначається декілька орієнтацій градієнтів. Довжина градієнта по сусідам  $m(x,y)$  та його орієнтація  $\theta(x,y)$ :

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2},$$

$$\theta(x,y) = \tan^{-1}(L(x,y+1) - L(x,y-1)) / (L(x+1,y) - L(x-1,y)),$$

де  $L(x,y)$  — значення згладженого гауссового зображення з найближчим масштабом.

- 4) Дескриптор точок. Локальні градієнти обчислюються для кожного масштабу навколоожної ключової точки. На рис. 4.4 наведено створення дескрипторівожної ключової точки.

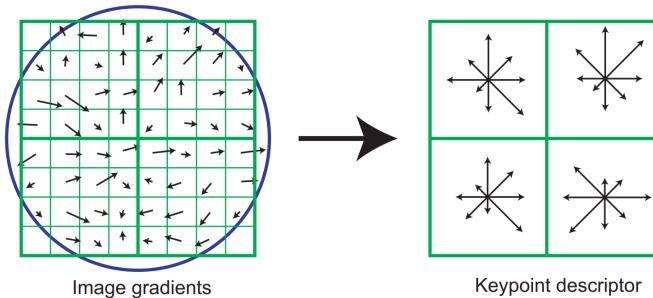


Рисунок 4.4 — Створення дескрипторів точок [? ]

#### 4.1.2 Знаходження відповідних точок зображень

Дано два кадри  $F^i$  та  $F^{i+s}$ . Нехай ми маємо набір  $\{(p_j^i, p_j^{i+s}) : j = \overline{1,4}\}$  пар координат відповідних точок, де  $p_j^i \in R^2 \times \{1\}$  — координата пікселя  $p$  на кадрі номер  $i$ ,  $p_j^{i+s} \in R^2 \times \{1\}$  — відповідна її координата на кадрі з номером  $i+s$ . Відповідними точками є ті, що є проекціями однієї і тієї ж точки у просторі.

Нехай  $D^i$  та  $D^{i+s}$  — множини дескрипторів кадрів  $F^i$  та  $F^{i+s}$  відповідно. Мета — отримати множину  $M^{i,i+s}$  усіх знайдених відповідних точок між кадрами  $F^i$  і  $F^{i+s}$ . Повний перебір (англ. brute force matcher) був використаний для знаходження відповідних точок.

---

#### Algorithm 2 Алгоритм Brute Force Matcher

---

**Вхід:** множини дескрипторів  $D^i$  та  $D^{i+s}$ .

**Вихід:** множина відповідних точок  $M^{i,i+s}$ .

**Ініціалізація:** Створюємо множину відповідних дескрипторів  $D^{i,i+s}$

$\forall i \in D^i$ :

$$h_{i,j}^{\min} \leftarrow 0$$

$\forall j \in D^{i+s}$ :

$$h_{i,j} \leftarrow \text{distance}(i, j)$$

Якщо  $h_{i,j} \leq h_{i,j}^{\min}$

$j$  є відповідним дескриптором для  $i$ , додаємо до  $D^{i,i+s}$

З  $D^{i,i+s}$  формуємо  $M^{i,i+s}$

---

Тобто для кожного дескриптора первого зображення шукаємо найближчий на другому. Маючи список відповідних дескрипторів можна знайти відповідні точки на обох зображеннях.

Приклад застосування SIFT разом з Brute Force Matcher на дошці з викладачем наведений на рис. 4.5.



Рисунок 4.5 — Кадри  $F^i$  та  $F^{i+s}$  з ключовими точками та лініями, які поєднують відповідні точки з відео [? ]

## 4.2 Знаходження матриці гомографії за допомогою RANSAC

Матриця гомографії — матриця, що описує зв'язок точок між двома зображеннями плоскої поверхні.

$$H = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \quad (4)$$

$$H^i \cdot p_j^{i+s} = c \cdot p_j^i, \quad \forall i = \overline{1,4} \quad (5)$$

Рівняння (5) можна легко розв'язати відносно невідомої матриці  $H^i$ . Цю матрицю можна використати для компенсації руху камери — ми застосовуємо її до координат пікселів кадру номер  $i+s$ , після чого точки зображення, отриманого в результаті перетворення, мають ті ж координати, що й відповідні їм точки на кадрі під номером  $i$ . Якщо отримані координати не ціличисельні, їх можна округлити, що не матиме значного негативного впливу на якість результату.

Знаходження  $H$  вимагає знаходження всіх 9 параметрів матриці (4). Візьмемо пару відповідних точок  $T$  та  $T'$ :  $T = ((x_1, y_1, 1), (x_2, y_2, 1), (x_3, y_3, 1), (x_4, y_4, 1))$ ,  $T' = ((x'_1, y'_1, 1), (x'_2, y'_2, 1), (x'_3, y'_3, 1), (x'_4, y'_4, 1))$ . Перепишемо (5) у матричній формі для  $T$  та

$T'$ .

$$\begin{bmatrix} x'_i \lambda \\ y'_i \lambda \\ \lambda \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{13} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}, \quad \forall i, j = \overline{1, 4} \quad (6)$$

приходимо до розв'язку системи

$$\underbrace{\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1x'_1 - y_1x'_1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x_2x'_2 - y_2x'_2 \\ x_3 & y_3 & 1 & 0 & 0 & 0 & -x_3x'_3 - y_3x'_3 \\ x_4 & y_4 & 1 & 0 & 0 & 0 & -x_4x'_4 - y_4x'_4 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1y'_1 - y_1y'_1 \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -x_2y'_2 - y_2y'_2 \\ 0 & 0 & 0 & x_3 & y_3 & 1 & -x_3y'_3 - y_3y'_3 \\ 0 & 0 & 0 & x_4 & y_4 & 1 & -x_4y'_4 - y_4y'_4 \end{bmatrix}}_A \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \end{bmatrix} = h_{33} \begin{bmatrix} x'_1 \\ x'_2 \\ x'_3 \\ x'_4 \\ y'_1 \\ y'_2 \\ y'_3 \\ y'_4 \end{bmatrix}.$$

Варто відмітити, що ми потім позбуваємося  $h_{33}$ , щоб  $H$  була нормованою, оскільки вона зв'язує точки площин, в яких 3-тя координата одиниця. У  $A$  маємо 8 рівнянь, для знаходження параметрів яких потрібно, щоб  $A$  була повноранговою. Для отримання  $H$  застосовується сингулярний розклад (SVD) матриці  $= U\Sigma V^*$ . Після розкладу  $V^*$  має розмір  $9 \times 9$ . Параметри  $H$  знаходяться в останньому рядку  $V^*$ .

$$H = V_{9,\cdot}^*/V_{9,9}^*, \quad H \in R^{1 \times 9} \rightarrow H \in R^{3 \times 3} \quad (7)$$

Для оцінки гомографії використали принцип RANSAC [? ]. Користувач вводить рівень  $\varepsilon > 0$  дозволеної похибки розрахунків — що менше, то краще, проте

тим довше буде працювати алгоритм пошуку матриці гомографії

$$E(H) = \sum_{(x,x') \in M^{i,i+s}} \left[ \left\| \frac{H \cdot x'}{(H \cdot x')_z} - x \right\| \leq \varepsilon \right] \rightarrow \max_{\substack{H \in \mathbb{R}^{3 \times 3} \\ \det H \neq 0}} . \quad (8)$$

---

**Algorithm 3** Алгоритм знаходження гомографії за принципом RANSAC

---

**Вхід:**  $M^{i,i+s}$  - множина відповідних точок,  $\varepsilon$  та  $n$  - кількість ітерацій.

**Вихід:**  $H^{best}$  - найкраща матриця гомографії.

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

1. Вибираємо випадковим чином 4 пари відповідних точок  $T$  та  $T'$  з  $M^{i,i+s}$ .
2. Обчислюємо  $H$  по формулі (7)
3. Обчислюємо  $E(H)$  по формулі (8)
4. Якщо  $E(H) < E(H^{best})$ :

$$H^{best} \leftarrow H$$


---

#### 4.2.1 Обробка ключових точок з маскою

Ключові точки (рис. 4.6(a)), що дає нам дескриптор ознак SIFT, ми використовуємо для оцінки гомографії, але також видаляємо ті ключові точки, які потрапляють в область маски рухомих об'єктів. Таким чином, ми зменшуємо ймовірність того, що замість дошки алгоритм “зачепиться” за одяг викладача або інші рухомі об'єкти. На рис. 4.6(б) можна помітити, що існують точки, які насправді не належать рухомим об'єктам кадру  $F^{i+s}$ , але були видалені. Це відбулось через те, що такі точки відповідають точкам з кадру  $F^i$ , які в свою чергу лежать в межах маски рухомих об'єктів.

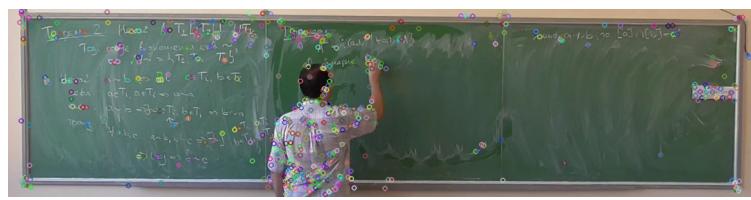
(a) Ключові точки на кадрі  $F^{i+s}$ (б) Відповідні ключові точки на кадрі  $F^{i+s}$ , що залишились після видалення тих точок, що потрапили в область маски рухомих об'єктів

Рисунок 4.6 — Результат застосування маски для відповідних точок з відео [? ]

### 4.3 Створення панорамного знімку

Часто бувають ситуації, коли для запису лекції використовується камера з невисокою роздільною здатністю, через що виникає необхідність рухати камеру, тому написи, які були на дошці, перестають бути видимими для глядачів, які не встигли записати матеріал, що було тільки-но представлено. Рішенням цієї та інших пов'язаних з тримтінням камери проблем є побудова панорами, яка створюється шляхом склеювання та накладання кадрів з відео.

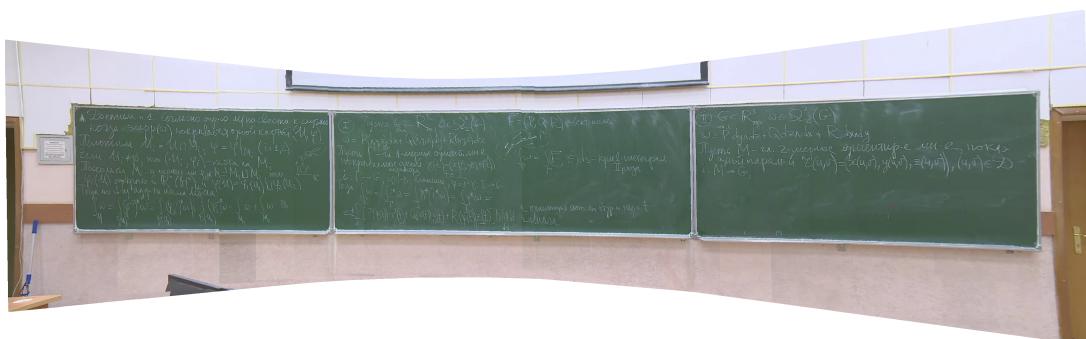


Рисунок 4.7 — Приклад отриманої панорами з автоматичним видаленням викладача з відео [? ]

Панорамою  $W^i$  називатимемо відображення  $W^i : P_W^i \rightarrow C$  з множини  $P_W^i = \{1, \dots, w^i\} \times \{1, \dots, h^i\}$  пікселів у множину  $C \subset \mathbb{R}$  інтенсивностей. Зауважимо, що у кожної панорами може бути своя ширина  $w^i \geq w$  і висота  $h^i \geq h$ .

---

**Algorithm 4** Створення панорами
 

---

**Вхід:** два кадри  $F^i$  та  $F^{i+s}$ , поточна панорама  $W^i$  ( $W^1 = F^1$ ).

**Вихід:** панорама  $W^{i+s}$ .

**Етап отримання відповідних точок**

**1.** Знаходимо набір  $M^i$  пар відповідних пікселів між кадрами  $F^i$  і  $F^{i+s}$  і будуємо множину  $M'^i = \{(p^i, p^{i+s}) \in M^i : B_{p^i}^i = B_{p^{i+s}}^i = 0\}$  тих пар відповідних пікселів, координати яких не належать області рухомих об'єктів.

**2.** Якщо  $|M'^i| < 0.5 \cdot |M^i|$  або  $|M'^i| < 4$ , завершуємо алгоритм з результатом  $W^{i+s} = W^i$ .

**3.** Знаходимо набір  $M_W^i$  пар відповідних пікселів між панорамою  $W^i$  і кадром  $F^{i+s}$  і будуємо множину  $M'_W^i = \{(p_W^i, p^{i+s}) \in M_W^i : \exists p^i \in P : (p^i, p^{i+s}) \in M'^i\}$ .

**Етап обчислення матриці гомографії**

**4.** На базі множини  $M'_W^i$  пар відповідних точок знаходимо матрицю  $H_W^i$  гомографії, що співставляє пікселі кадру  $F^{i+s}$  та панорами  $W^i$ .

**Етап обчислення розміру нової панорами**

**5.** Рахуємо координати  $l_1^i = H_W^i \cdot (0, 0, 1)^T$ ,  $l_2^i = H_W^i \cdot (w - 1, 0, 1)^T$ ,  $l_3^i = H_W^i \cdot (0, h - 1, 1)^T$ ,  $l_4^i = H_W^i \cdot (h - 1, h - 1, 1)^T$  крайніх точок кадру  $F^{i+s}$  після застосування до них матриці  $H_W^i$ .

**6.** Для визначення множини  $P_W^{i+s}$  нової панорами  $W^{i+s}$  рахуємо величини  $x_{\min}^i = \min_{j=1,4} \frac{(l_j^i)_x}{(l_j^i)_z}$ ,  $x_{\max}^i = \max_{j=1,4} \frac{(l_j^i)_x}{(l_j^i)_z}$ ,  $y_{\min}^i = \min_{j=1,4} \frac{(l_j^i)_y}{(l_j^i)_z}$ ,  $y_{\max}^i = \max_{j=1,4} \frac{(l_j^i)_y}{(l_j^i)_z}$ . Позначимо  $P_W^{i+s} = 1, \dots, \max(w^i, x_{\max}^i - x_{\min}^i) \times 1, \dots, \max(h^i, y_{\max}^i - y_{\min}^i)$ .

**Етап створення нової панорами**

**7.** Будуємо панораму  $W^{i+s}$ . Для зручності позначимо обернену матрицю  $H' = (H_W^i)^{-1}$ , числа  $x'_{\min} = \min(x_{\max}^i, 0)$ ,  $y'_{\min} = \min(y_{\max}^i, 0)$  і відображення  $f^i : p \rightarrow \left(\left|\frac{(H' \cdot p)_x}{(H' \cdot p)_y}\right|, \left|\frac{(H' \cdot p)_y}{(H' \cdot p)_z}\right|\right)$ , що перетворює координати з панорами до пікселів кадру за допомогою гомографії. Інтенсивність у пікселі  $p$  панорами  $W^{i+s}$  визначається за формулою

$$W^{i+s}(p) = \begin{cases} F^{i+s}(f(p)), & f(p) \in P, \\ W^i(p + (x'_{\min}, y'_{\min})), & p + (x'_{\min}, y'_{\min}) \in P^i, \\ 0, & p + (x'_{\min}, y'_{\min}) \notin P^i \end{cases}$$

## **Висновки до розділу 4**

Було розглянуто алгоритм знаходження ключових точок SIFT. Показано, як знаходити найкращу матрицю гомографії за допомогою RANSAC. Побудовано кінцевий алгоритм створення панорамного знімку без викладача.

## 5 ОБРОБКА СЛАЙДІВ

У п'ятому розділі описується процес прийняття рішення щодо того, чи брати панорамний знімок як слайд, чи ні. Також наводиться приклад роботи оператору Лапласа та робиться огляд швидкої медіані для зменшення шуму.

### 5.1 Порівняння слайдів

Після видалення викладача та суміщення кадрів ми створюємо слайди у вигляді коротких нотаток лекції. Ми по черзі беремо пари кадрів панорами, які необов'язково є сусідніми — для збільшення чутливості алгоритму створення нових слайдів можна використовувати кадри, різниця між індексами яких є більшою, бо змін на дощі між сусідніми кадрами може не бути.

Ми використовуємо оператор Лапласа для знаходження перепадів яскравості і зменшення рівня шуму. У наших експериментах саме цей оператор надав кращі результати, ніж інші диференціальні оператори

$$L(x,y) = \nabla^2 f(x,y) = \frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2}.$$

Для одержання маски всіх написів на дощі застосовуємо метод бінаризації Оцу . Це швидкий алгоритм, який на наш погляд дає прийнятну якість бінаризації для створення слайдів (рис. 5).

Не кожна панорама є новим слайдом. Коли зміни на дощі невеликі (наприклад, лектор дописав формулу), треба лише оновити існуючий слайд. Якщо ж зміни значні (наприклад, лектор стер попередні записи), треба створити новий слайд і працювати з ним. Щоб мінімізувати недоліки маски та отримувати слайди з новою інформацією на дощі, ми виділяємо нові написи за допомогою оператора Лапласа.

**Означення 9.** Слайд — це та панорама  $W^i$ , на якій відображен стан дошки напередодні стану з великою кількістю змін.

Ми порівнюємо між собою не кожну панораму, а пропускаємо вказану користувачем кількість  $q$  панорам. Оскільки розміри слайду  $W^i$  і панорами  $W^{i+s \cdot q}$  можуть бути різними, треба враховувати це у процесі їх порівняння. Для цього розраховуємо горизонтальний зсув

$$x'_{\min} = \sum_{j=1}^q \min \left( x_{\min}^{i+s \cdot j}, 0 \right)$$

і вертикальний зсув

$$y'_{\min} = \sum_{j=1}^q \min \left( y_{\min}^{i+s \cdot j}, 0 \right).$$

Користувач вказує кількість змін  $t \in [0; 1]$  між слайдом та панорамою, яка є вирішальною для прийняття рішення щодо створення нового слайду. Нехай  $L^i$  та  $L^{i+s \cdot q}$  — панорами, отримані в результаті обробки оператором Лапласа панорам  $W^i$  та  $W^{i+s \cdot q}$  відповідно. Якщо виконується нерівність

$$\sum_{(x,y) \in P^i} |L^i(x,y) - L^{i+s \cdot q}(x - x'_{\min}, y - y'_{\min})| > t \cdot |P^{i+s \cdot q}|,$$

бінаризовану  $W^i$  додаємо до списку слайдів, а в іншому випадку переходимо до перевірки слайду  $W^{i+s \cdot q}$ .

## 5.2 Темпоральна медіана

Введемо необхідні означення.

**Означення 10.** Накопичувальна сума масиву [? ]  $X$  — це масив, кожний елемент якого є сумою всіх попередніх.

Наведемо рекуррентну формулу обчислення накопичувальної суми

$$S_1 = a_1, \quad S_n = a_n + S_{n-1}. \quad (9)$$

**Означення 11.** Медіаною [?] називають елемент, який стоїть посередині впорядкованого масиву. Якщо кількість елементів парна, то медіаною називається середнє значення між двома числами, що стоять посередині масиву.

**Означення 12.** Темпоральною медіаною у комп'ютерному зорі називають зображення, в кожному пікселі якого знаходиться медіанне значення, що знайдене між значеннями пікселів у часовому проміжку (рис. 5.1).

Темпоральну медіану застосовують у комп'ютерному зорі для отримання фону сцени у відео [?]. Тобто всі об'єкти, що рухаються, зникають на медіанному зображенні (рис. 5.1). Розрахунок темпоральної медіани складається з наступних кроків:

- 1) береться набір кадрів деякого часовому проміжку;
- 2) сортуємо по кожному пікселю масиви;
- 3) беремо медіану також по кожному пікселю.

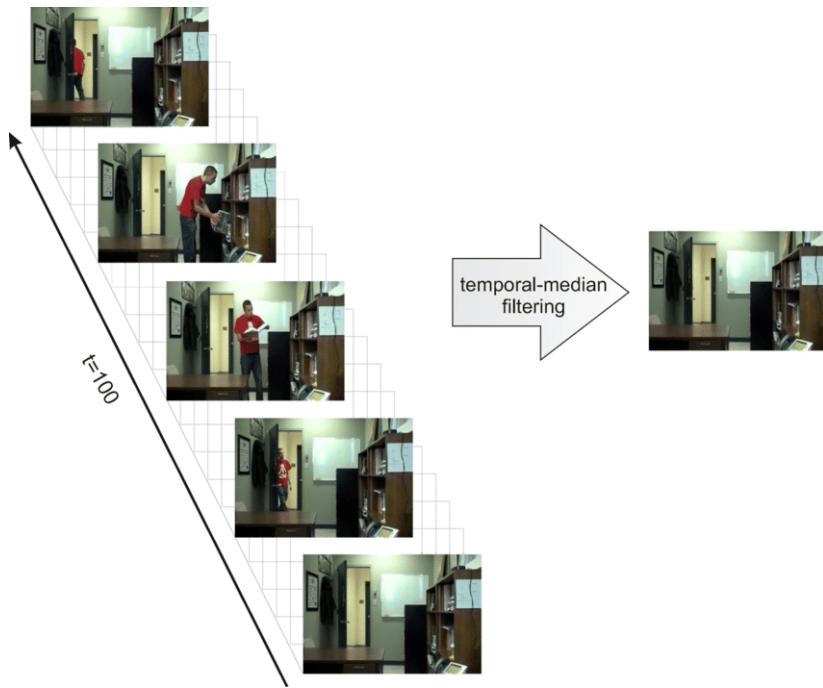


Рисунок 5.1 — Приклад отримання фону без людини шляхом використання темпоральної медіани [? ]

Даний метод можна застосовувати як для прибирання швидких рухомих об'єктів, так і для отримання зображення із високою роздільною здатністю (super-resolution).

### 5.3 Швидка медіана для зображень

Розглянемо простий випадок взяття медіани. Нехай у нас є не відсортований масив цілих чисел з множини  $\{0, \dots, 255\}$ . Для того, щоб знайти в ньому медіану, потрібно відсортувати масив. Для сортування масиву можна використати алгоритм швидкого сортування (quick sort) [? ]. У середньому випадку він працює за  $\mathcal{O}(n \log n)$ , в найгіршому  $\mathcal{O}(n^2)$ . Запропонований нижче алгоритм працюватиме за  $\mathcal{O}(n)$ .

Використаємо гістограму як хеш-таблицю значень від 0 до 255.

**Означення 13.** Хеш-таблиця — це структура з елементом **ключ: значення**, де ключ є унікальним ідентифікатором.

**Означення 14.** Гістограма зображення  $I$  — це масив, в якому елемент під індексом

$i$  містить число, що показує скільки разів у зображенні  $I$  зустрічається значення  $i$ .

Позиція елемента  $i$  в гістограмі є значенням піксела, а значення елемента на позиції  $i$  — це кількість повторень  $i$  на самому зображені  $I$ .

Введемо операції

- $A[i]$  — взяття елемента в масиві  $A$  під номером  $i$ ;
- $A[i : j]$  — взяття підмасиву масиву  $A$ , починаючи включно з елементу  $i$  до  $j$  не включно;
- $a \bmod b$  — взяття остачі від ділення числа  $a$  на  $b$ ;
- $\lceil a \rceil$  — округлення дійсного числа  $a$  до верхньої границі;
- $\lfloor a \rfloor$  — округлення дійсного числа  $a$  до нижньої границі;
- $\overline{a, b}$  — множина цілих чисел на відрізку від  $a$  до  $b - 1$ , якщо  $a \leq b$ , інакше — від  $b - 1$  до  $a$ .

Наведемо алгоритм 5 знаходження медіани в одновимірному випадку.

#### **Algorithm 5** Знаходження медіани для одновимірного масиву з хеш-таблицею.

**Вхід:** не відсортований масив  $A$ ,  $\ell$  — кількість елементів.

**Вихід:**  $m$  — медіанне значення масиву  $A$ .

**Ініціалізація:**  $H$  — ціличисельний масив, що заповнено нулями (гістограма) на 256 елементів.

**Крок 1:** Заповнюємо масив  $H$ .

$$\forall a \in A$$

$$H[a] \leftarrow H[a] + 1$$

**Крок 2:** Сортуємо масив  $A$ :

$$i \leftarrow 0$$

$$\forall j \in \overline{0, 256}$$

$$A[i : i + H[j]] \leftarrow j$$

$$i \leftarrow i + H[j]$$

**Крок 3:** Знаходимо  $m = \frac{A[\lfloor \ell/2 \rfloor] + A[\lceil \ell/2 \rceil]}{2}$

## 5.4 Побудова швидкої медіани для зображень

Нехай у нас на вхід подається черга із  $m$  картинок. Кожну ітерацію приходить нова картинка.

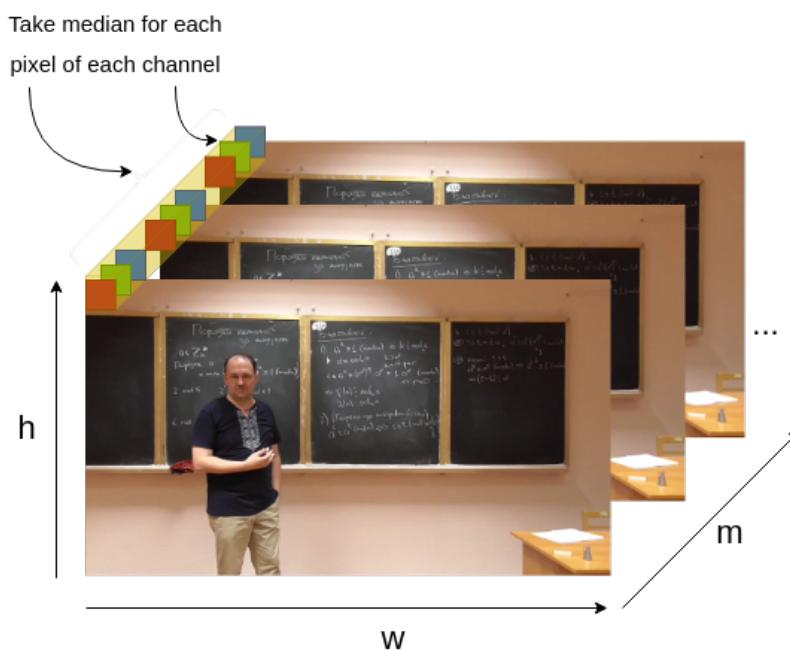


Рисунок 5.2 — Приклад послідовності картинок для формування медіани з відео [? ]

Для простішого пояснення методу швидкої медіани візьмемо одноканальне зображення зображення у відтінках сірого розмірами  $h \times w$ , де  $h$  — висота,  $w$  — ширина, але в реальності операції, що наведені нижче виконуються для трьохканального  $RGB$  зображення окремо для кожного каналу.

В інформаційній технології перетворення відеозапису з дошки у слайд-шоу [? ] [?] алгоритм швидкої медіани застосовується на панорамних знімках з метою знешумлення на підвищення якості написів на дошці.

Введемо такі масиви:

- $F$  — масив із  $n$  зображень із відео або панорамних знімків, розміром  $n \times h \times w$ ;
- $S$  — масив, що містить всі  $m$  вхідних зображень розміром  $h \times w \times m$ ;
- $M$  — масив, що є темпоральною медіаною по  $m$  зображені розміром  $h \times w$ ;

- $C$  — масив, що є накопичувальною сумаю розміром  $h \times w \times 256$ . У кожному елементі під індексом  $i$  в масиві з 256 елементів зберігається число, що означає кількість разів присутності значення  $i$  в кожному масиві з  $m$  елементів в масиві  $S$ .

Розглянемо алгоритм ініціалізації масивів  $S, M, C$  (алгоритм ??).

---

#### **Algorithm 6** Алгоритм ініціалізації масивів $S, M, C$

---

**Вхід:**  $m$  — кількість зображень для обчислення медіани,  $I$  — перше зображення з відео.

**Вихід:** масиви  $S, M, C$ .

**Крок 1:** Створюємо  $S$  шляхом конкатенації  $m$  разів масиву  $I$ . Тоді  $S$  має розміри  $h \times w \times m$ . Тобто зараз у  $S$  міститься  $m$  копій зображення  $I$ .

**Крок 2:** Створюємо медіанне зображення:  $M \leftarrow I$

**Крок 3:** Створюємо нульовий масив  $C$  розмірами  $h \times w \times 256$ .

**Крок 4:** Заповнюємо  $C$  використовуючи  $M$ :

$$\forall i \in \overline{0, h}$$

$$\forall j \in \overline{0, w}$$

$$C[i, j, M[i, j]] \leftarrow m$$

**Крок 5:** Обраховуємо накопичувальну суму  $E$  для кожного масиву з 256 елементів масиву  $C$  по формулі (9).

**Крок 6:**  $C \leftarrow E$ .

---

Наведемо алгоритм оновлення  $C$  (алгоритм 7) по двом зображенням, одне з яких назовемо інкрементним  $I_{in}$ , а інше декрементним  $I_{de}$ .

---

**Algorithm 7** Алгоритм оновлення  $C$ 

---

**Вхід:** масив  $C$ , інкрементне та декрементне зображення  $I_{in}, I_{de}$ .

**Вихід:** оновлений масив  $C$ .

$$\forall i \in \overline{0, h}$$

$$\forall j \in \overline{0, w}$$

$$\forall k \in \overline{I_{in}[i, j], 256}$$

$$C[i, j, k] \leftarrow C[i, j, k] + 1$$

$$\forall k \in \overline{I_{de}[i, j], 256}$$

$$C[i, j, k] \leftarrow C[i, j, k] - 1$$


---

Також потрібен алгоритм оновлення медіанного зображення (Алгоритм 8)  $M$ .

---

**Algorithm 8** Алгоритм оновлення  $M$ 

---

**Вхід:** масиви  $M, S, C$ , та параметр  $m$ .

**Вихід:** оновлений масив  $M$ .

$$p \leftarrow \lfloor m/2 \rfloor$$

▷ припускаємо, що медіана знаходиться посередині

$$\forall i \in \overline{0, h}$$

$$\forall j \in \overline{0, w}$$

Якщо  $C[i, j, M[i, j]] \geq p$ :

$$\forall k \in \overline{M[i, j] - 1, - 1}$$

Якщо  $C[i, j, k] \leq p$ :

$$p_{new} \leftarrow C[i, j, k + 1]$$

стоп

інакше:

$$\forall k \in \overline{M[i, j] + 1, 256}$$

Якщо  $C[i, j, k] \geq p$ :

$$p_{new} \leftarrow C[i, j, k - 1]$$

стоп

$$M[i, j] \leftarrow S[i, j, \lfloor p_{new} \rfloor]$$


---

Використовуючи вищенаведені алгоритми, можна записати сам метод швид-

кої медіани для потоку зображень (Алгоритм 9).

---

**Algorithm 9** Алгоритм швидкої медіани для потоку зображень
 

---

**Вхід:** масив  $F$ , та параметр  $m$ .

**Вихід:** список медіан  $L$ .

**Крок 1:** Ініціалізуємо масиви  $S, M, C$  по зображеню  $F[1]$  по алгоритму ??

**Крок 2:**  $q \leftarrow 0$  ▷ лічильник черги

$\forall i \in \overline{0, n-1}$

**Крок 3:** Оновлюємо  $C$  по алгоритму 7 з параметрами  $C, F[i]$  — інкрементне зображення,  $F[i+1]$  — декрементне зображення.

**Крок 4:** Оновлюємо  $M$  по алгоритму 8 з параметрами  $C, M, m$ .

**Крок 5:** Ставимо на місце  $q$  в масиві  $S$  зображення  $F[i]$

**Крок 6:** Оновлюємо лічильник  $q \leftarrow (q + 1) \bmod m$

**Крок 7:** Додаємо нову  $M$  у  $L$ .

---

Для аналогії наведемо звичайний алгоритм медіани для  $S$ .

---

**Algorithm 10** Звичайний алгоритм медіани для  $S$ 


---

**Вхід:** масив  $S$ , та параметр  $m$ .

**Вихід:** масив  $M$ .

$\forall i \in \overline{0, h}$

$\forall i \in \overline{0, w}$

Сортуємо масив з  $m$  елементів і позначимо за  $e$ .

$M[i, j] \leftarrow e[\lfloor m/2 \rfloor]$  ▷ записуємо значення медіани

---

## 5.5 Порівняння складності

Обчислимо складність швидкої медіани  $m$  зображень, кожне з яких розміром  $h \times w$ :  $\mathcal{O}(h \cdot w)(\text{ініціалізація } S, M, C) + \mathcal{O}(h \cdot w)(\text{оновлення } C) + \mathcal{O}(h \cdot w)(\text{оновлення } M) = \mathcal{O}(h \cdot w)$ . Аналогічно для звичайного випадку із quick sort:  $\mathcal{O}(h \cdot w \cdot m^2)$  у найгіршому

випадку. Варто відмітити, що таке прискорення досягається тим, що є обмеження на використання значень від 0 до 255.

## **Висновки до розділу 5**

Запропоновано метод прийняття рішення щодо того, чи обирати панорамний знімок як слайд, чи ні. Оглянуто теоретичне підґрунтя темпоральної медіани для знешумлення так покращення якості панорамних знімків.

## 6 ПРАКТИЧНІ РЕЗУЛЬТАТИ

У шостому розділі наводяться результати застосування алгоритму Бойкова-Колмогорова, згорткових нейронних мереж YOLOv5, MobileNetv3, SSD та Faster R-CNN. Показано на прикладах як працює інформаційна технологія для створення панорамних слайдів без викладача. Наступні результати були отримані за допомогою мови програмування Python її бібліотек PyTorch та PyMaxflow. Наступні тестування проводились на комп’ютері з процесором Intel Core i5-7200U @ 4x 3,1GHz на операційній системі Ubuntu 21.04.

### 6.1 Результати алгоритму Б-К та згорткових мереж

В даній роботі досить непогано проявив себе метод Б-К, а саме програмна бібліотека PyMaxflow автором якої є сам Колмогоров. Даний метод найшвидше працює саме в ній.

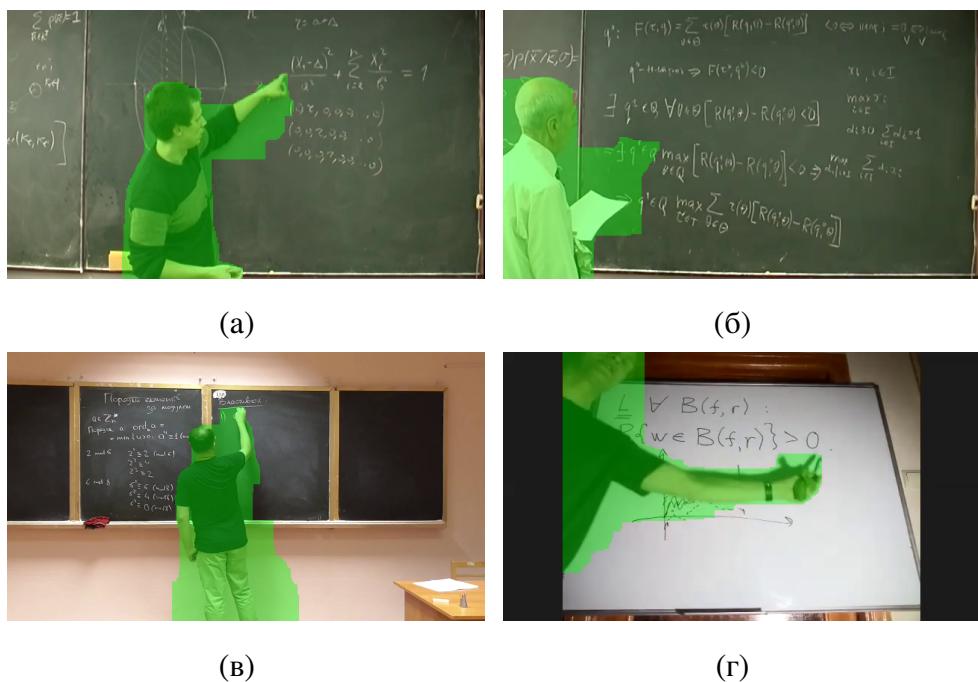


Рисунок 6.1 — Приклад роботи алгоритму Б-К для отримання маски рухомих об’єктів

Як ми бачимо на прикладах (Рис. 6.1) досить гарна якість маски рухомих об'єктів.

### Переваги методу:

- 1) Алгоритм Б-К досить швидко будує мінімальний розріз. Статистика на ОС під, час 0.15 секунди на кадр розміром 330x640.
- 2) Даний метод не потребує ніякої передобробки, тобто не потрібно навчати як нейронну мережу.

### Недоліки методу:

- 1) Оскільки даний метод локалізує рух на кадрах відео, то коли рухається камера, відповідно практично все що в кадрі стає рухомим об'єктом (Рис.6.2). Частково дана проблема вирішена на 2-му кроці алгоритму створення панорами (Алгоритм 4).

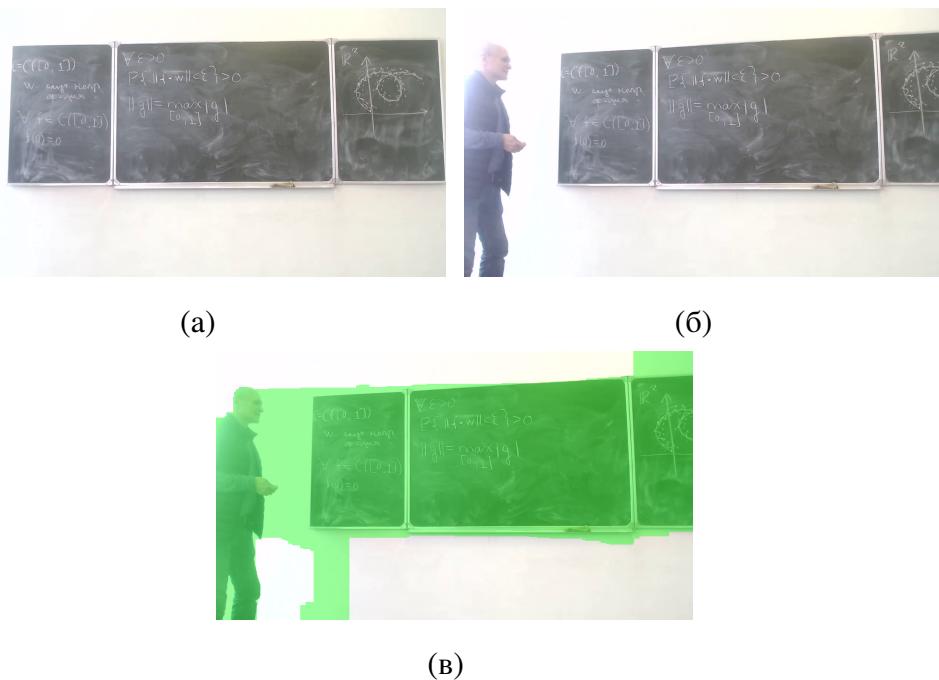


Рисунок 6.2 — Приклад поганої маски рухомих об'єктів [? ]

- 2) Якщо викладач практично не рухається довгий час, відповідно частково не потрапляє на маску рухомих об'єктів, з цього слідує, що він може потрапити на панорамний слайд.

## 6.2 Результати згорткових мереж

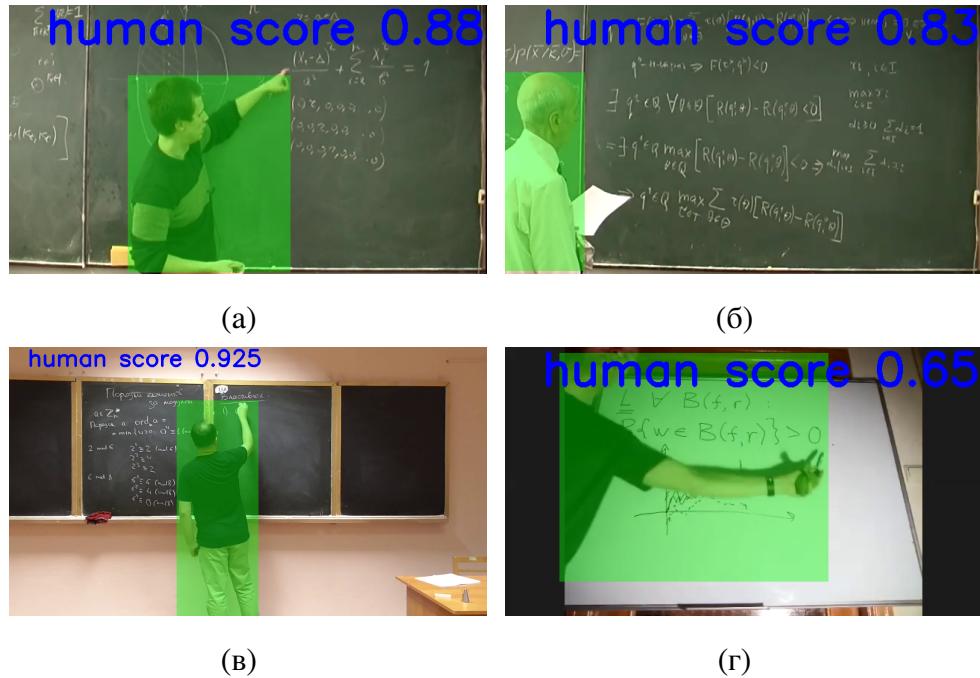


Рисунок 6.3 — Приклад роботи yolov5n

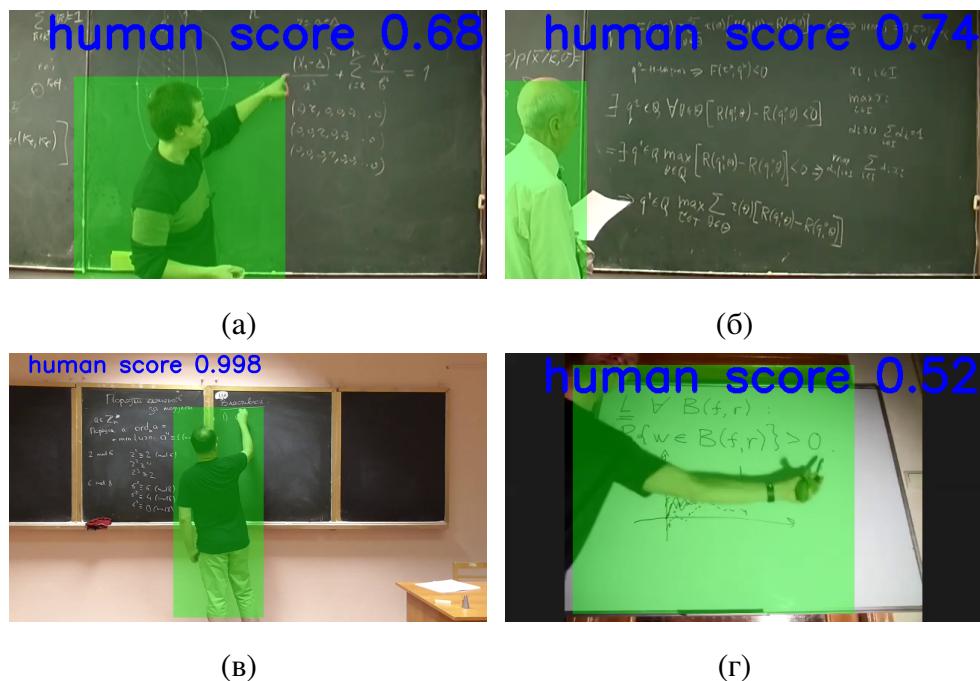


Рисунок 6.4 — Приклад роботи ssdlite320-mobilenet-v3

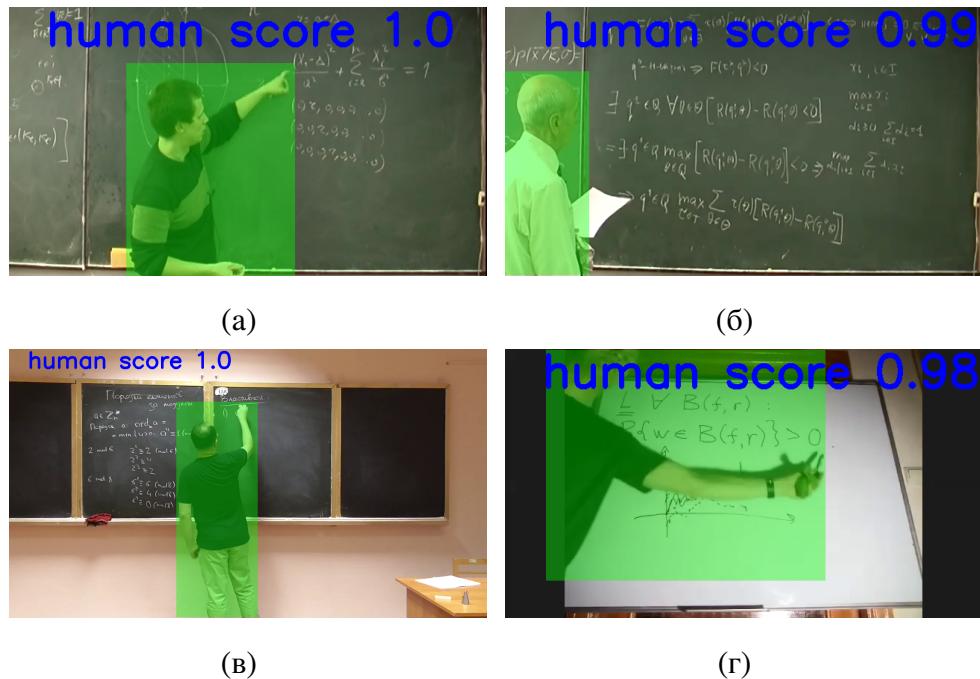


Рисунок 6.5 — Приклад роботи fasterrcnn-mobilenet-v3

### Переваги згорткових мереж:

- 1) Згорткові мережі детекції людини, що були використані в даній роботі є теж досить швидкими, оскільки були створені для мобільних пристройів.
- 2) Детекція людини не залежить від руху камери. Відповідно навіть кадри, що отримуються під час руху камери будуть застосовані для створення панорами.
- 3) Результати свідчать про високу точність локалізації викладача.

### Недоліки згорткових мереж:

- 1) Якщо викладач має в руці якийсь предмет, листок паперу чи маркер, то дані об'єкти мережа не локалізує, відповідно можуть бути дефекти на панорамі.
- 2) Розмір інформаційної технології стає більшим через зберігання ваг.

### 6.3 Швидкість методів локалізації людини чи рухомих об'єктів

Було протестовано 1000 разів отримання маски різними методами на картинці  $720 \times 1280$  та отриманий середній час обробки кадру (Таб. 6.1). Варто відмітити, що картинка перед входом в шари згорток у згорткових мережах підлягають компресії

до якогось базового розміру для входу. Така ж компресія робиться і для входу в алгоритм Б-К, картинка зменшується удвічі.

Таблиця 6.1 — Таблиця швидкості згорткових мереж та алгоритму Б-К

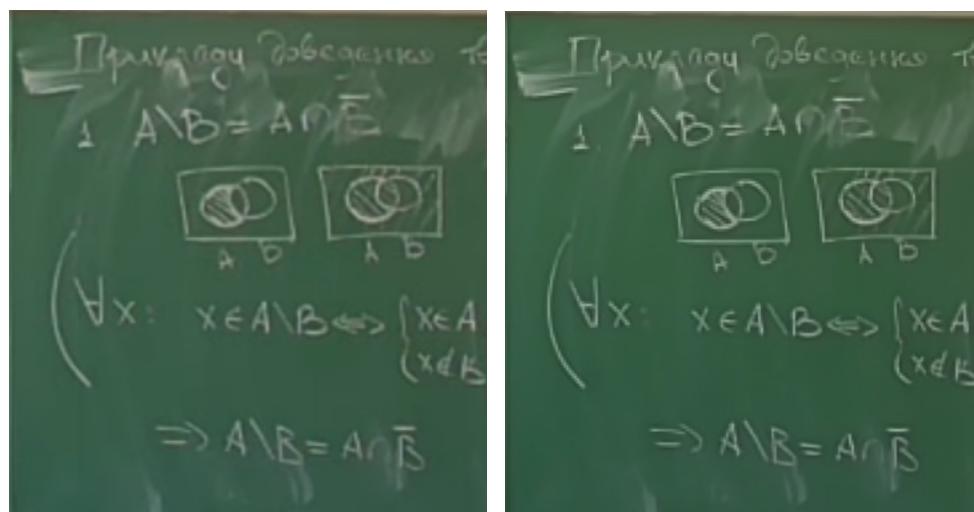
yolov5n	ssdlite320-mobilenet-v3	fastercnn-mobilenet-v3	Boykov-Kolmogorov
0.12	0.06	0.12	0.22

Як бачимо найшвидшим методом є ssdlite320-mobilenet-v3, а найдовшим Бойкова-Колмогорова. Але експерименти показали, що прийнятну якість дає саме yolov5n.

## 6.4 Результати створення панорами

## 6.5 Приклад роботи

На рис. 6.6(б) можна побачити як медіана справляється з шумом. Написи стали більш чіткими, а фон дошки більш однорідним. Даний алгоритм доданий до інформаційної технології перетворення відео з дошки у слайди.



(a) До

(б) Після

Рисунок 6.6 — Приклад роботи медіани для зображень [? ]