

ABSTRACT

The thesis contains pages, figures and

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

To perform the study

- SIFT method for obtaining image's keypoints;
- Homography for getting panorama slides;
- Boykov-Kolmogorov Max-flow algorithm for creating mask of moving objects;
- Denoising and binarizing operators;

IMAGES PROCESSING, INTELLECTUAL VIDEO PROCESSING, VIDEO STABILIZA-

РЕФЕРАТ

Дисертація містить сторінки, ілюстрацій і бібліографію з 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 алгоритму, який з відео лекції робить панорамні слайди без викладача.

Для досягнення мети було використано

- Метод SIFT для знаходження ключових точок зображення;
- Гомографію для отримання панорамних слайдів;
- Алгоритм максимального потоку Бойкова-Колмогорова для отримання маски рухомих об'єктів;
- Оператори знешумлення та бінаризації зображень.

**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 Приклади	12
1.2 Постановка задачі	16
1.3 Пропоновані рішення.	17
Висновки до розділу 1	19
2 Створення маски викладача	20
2.1 Застосування нейронних мереж для отримання маски людини . . .	28
3 Створення панорами	44
3.1 Пошук відповідності між кадрами	44

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Стандартні позначення

\mathbb{N} — множина натуральних чисел;

\mathbb{R} — множина 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 $p \in P$;

ВСТУП

Дана робота завдячує Кригіну Валерію Михайлович — молодшому науковому співробітнику Відділу обробки та розпізнавання образів Міжнародного науково-навчального центру інформаційних технологій і систем НАН України та МОН України.

Актуальність роботи. Задача створення конспекту лекцій не є новою, а з масовим переходом навчання у віддалений режим стала більш актуальною проблема створення якісних лекційних матеріалів, зокрема слайдів, які містять стислу суть заняття. Було б добре мати технологію, що перетворюватиме 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.

Мета і завдання дослідження.

Об'єкт дослідження — математичні методи реєстрації зображень.

Предмет дослідження — автоматична обробка відеозаписів.

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

Завдання наступні:

- 1) вивчити методи обробки та порівняння зображень, які знадобляться для створення панорамних знімків;
- 2) вивчити та доповнити математичні методи, що допоможуть визначати рухомі об'єкти;
- 3) 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 відео лекції.

Практичне значення одержаних результатів.

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

Публікації.

.... Всеукраїнська науково-практична конференція студентів, аспірантів та молодих вчених «Теоретичні і прикладні проблеми фізики, математики та інформатики».

1 АНАЛІЗ ПОПЕРЕДНІХ РОБІТ

У першому розділі розглядаються попередні роботи по створенню слайдів з відеозаписів або фотографій дошки. Проаналізовано різні методи створення панорамних знімків і також алгоритми прибирання викладача з відео. Також описані переваги даної роботи над аналогічними. Аналіз попередніх робіт дає можливість для коректної постановки задачі створення слайдів з відео, яка описана в другому розділі дисертації.

1.1 Приклади

1.1.1 Одна з перших робіт з оцифровування дошки

У 2004 році інженери з Microsoft Research *Zhengyou Zhang* та *Li-wei He* представили свій алгоритм по скануванню написів білої дошки.

В їхній роботі [?] обробляються фотографії білої дошки. Локалізується область дошки, вирівнюється у прямокутну форму і бінаризуються написи без втрати кольору.(рис. ??).

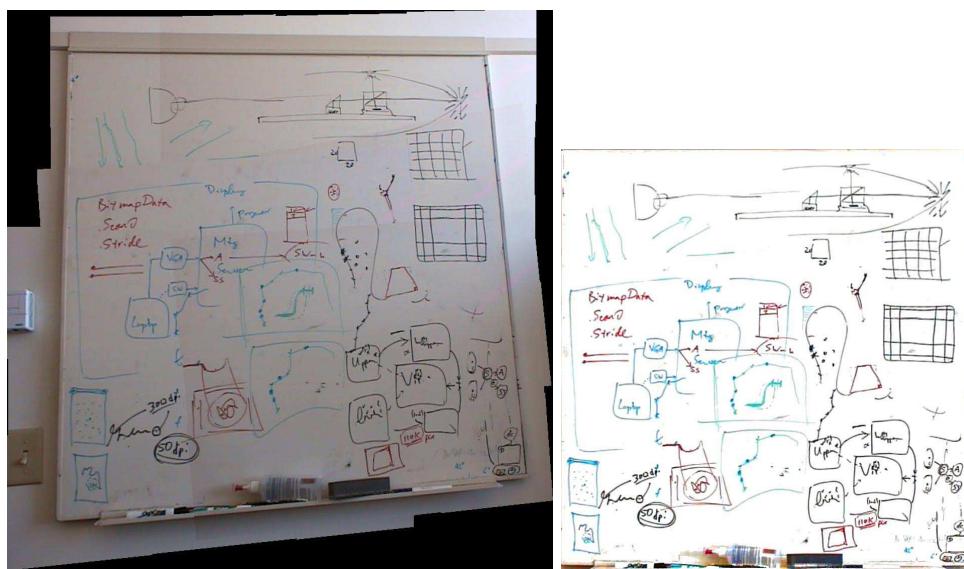


Рисунок 1.1 — Демонстрація роботи алгоритму інженерів з Microsoft

Також автори реалізували склейку зображень дошки зроблених з різних пер-

спектив за допомогою гомографії. Гарна якість оцифрування дошки досягається на- самперед тим, що вона має білий колір, що в свою чергу накладає обмеження для використання технології з дошками зеленого чи чорного кольорів. У наступній свої роботі [?] ці ж самі автори побудували систему, яка в реальному часі обробляє відеозапис, видаляє людину біля дошки за допомогою часової медіани, але в даному випадку немає панорамного склеювання знімків. Головна ідея роботи була зробити програму для телеконференсій.

1.1.2 Автоматичне сканування дошки

Автори даної роботи [?] створили програму яка переводить написи на білій дощі у цифрові. Вони реалізували локалізацію тексту та подальшу його обробку. Дані технологія не передбачає перекривання викладачем написів або дошку іншого кольору відмінного від білого.

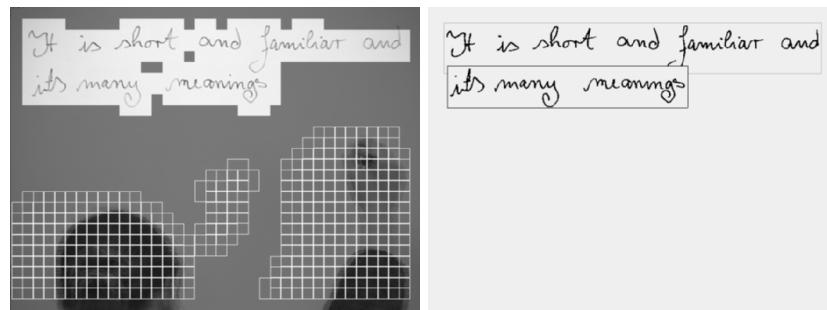


Рисунок 1.2 — Демонстрація роботи сканування дошки

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

1.1.3 Відстежування об'єкту та віднімання фону від Стенфорду

У 2012 році науковці зі Стенфордського університету *Alex Gonzalez, Bongsoo Suh, Eun Soo Choi* представили технологію [?] локалізацію 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 тимчасову медіану.

Дана програма не працює в реальному часі, оскільки всі операції над кадрами відео займають тривалий час, не кажучи вже, що відео перед обробкою піддають компресії.

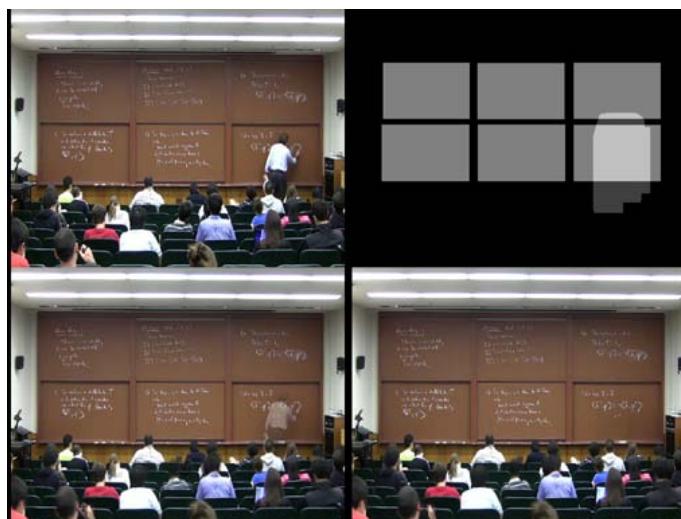


Рисунок 1.3 — Приклад роботи авторів Стенфорду

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. Тим більше якість отриманих написів теж погана.

1.1.4 Виокремлення написів дошки від Тайванського університету

У 2014 році науковці із Тайванського університету представили свій аналог [?] алгоритму по оцифровуванні дошки. Для видалення викладача автори застосували алгоритм кластеризації K-means. Для отримання бінаризованих написів з дошки використана адаптивне вирівнювання '(яке не ясно)'. Варто відмітити гарну якість власного методу знешумлення.

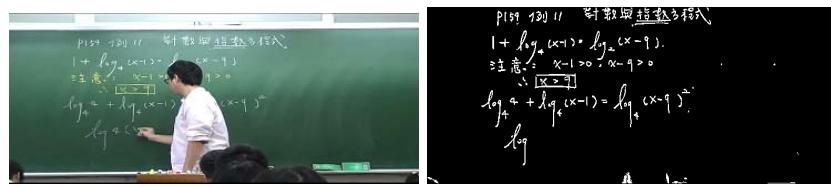


Рисунок 1.4 — Демонстрація роботи сканування дошки

Автори не представили швидкість обробки всього відео. Щоб отримати якісну сегментацію дошки та викладача потрібно, щоб кадр містив тільки викладача, причому щоб одяг викладача був сильно відмінним від кольору дошки. Це потрібно для коректної роботи K-means. Тобто нема гарантії, що якийсь рухомий об'єкт не стане дошкою під час класифікації.

1.1.5 Сучасна робота

На даний момент найсучаснішою повною технологією [?], яка оцифрує дошку є робота науковців з університету Рочестер. Автори *Kenny Davila* та *Richard Zanibbi* використали просторовий темпоральний індекс для виокремлення написів і викладача. Відбувається видалення не самого викладача, а його контурів, після бінаризації картинки. Варто відмітити, що і тут камера має бути нерухомою.

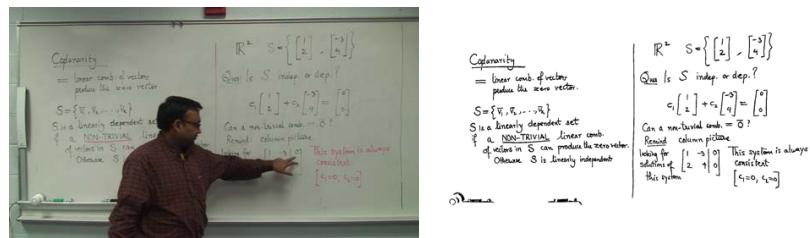


Рисунок 1.5 — Демонстрація роботи сканування дошки

Пізніше ці ж автори створили повністю згорткову нейронну мережу [?] для обробки написів дошки. Дана нейронна мережа LectureNet досить добре бінаризує написи з дошки.



Рисунок 1.6 — Приклад роботи авторів з Рочестер

1.1.6 Кінцевий аналіз робіт

Як ми бачимо, всі вищеописані алгоритми є частковими (тобто не повністю вирішують всі проблеми з оцифруванням дошки): якісь методи опрацьовують тільки написи з дошки, при чому досить непогано; якісь роблять панорамні знімки тільки з фотографій дошки.

Спільними елементами вищеописаних алгоритмів-аналогів є використання темпорального медіанного фільтру для видалення рухомих об'єктів. У даній роботі даний метод теж має місце, але вже для знешумлення вихідних слайдів, а не видалення лектора з відео.

Найбільшими недоліками всіх аналогів є час обробки відео. Більшість технологій тестиувались на відео поганої якості, наприклад 240р - 720р.

Також жоден метод не пропонує створювати панорамні знімки по мірі руху камери під час зйомки.

1.2 Постановка задачі

Головними цілями даної роботи створити власну математичну модель на вхід якої буде даватись відео-лекція, а на вихід слайди з та побудувати її програмну реалізацію.

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 f,j або закріплена не стабільно, таким чином це заважає нормальню слухати лекцію і фокусуватись на написах дошки.

*****Вставити два кадри і різницю, щоб показати не стабілізоване відео

- 2) Прибирання рухомих об'єктів; Також однією з головних задач є видалення викладача з відео, оскільки було поставлено за мету надавати користувачу чисті написи з дошки. Є чимало варіантів як прибирати рухомі об'єкти. Можна локалізувати тільки людину, а можна і всі об'єкти, що рухаються в площині дошки. Маючи маску таких об'єктів ми матимемо змогу оновлювати з часом інформацію з дошки яка була тимчасово перекрита.

*****Вставити кадр з маскою людини

- 3) Панорамні слайди; Часто, коли дошки сильно об'ємні по площині камера знімає тільки ту частину де викладач щось пише. Під час лекції камеру переміщають так, щоб лектор був завжди в полі зору. Таким чином, якщо студент не встиг переписати все з однієї частини дошки, а камеру перемістили, то учень втрачає частину інформації. Було б добре мати слайди, які по мірі руху камери містили всю дошку.

*****Вставити панорамну склейку кадрів

- 4) Написи з дошки; Умови освітлення, розводи на дошці, неякісна зйомка, шуми - все це сильно впливає на сприйняття лекційного матеріалу. Непогано було б мати бінаризовані слайди без шумів тільки з написами як в записнику.

1.3 Пропоновані рішення

У даній роботі описується алгоритм напівавтоматичного створення слайдів на основі відеозапису лекції (рис. 1.7). 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 розрахову-

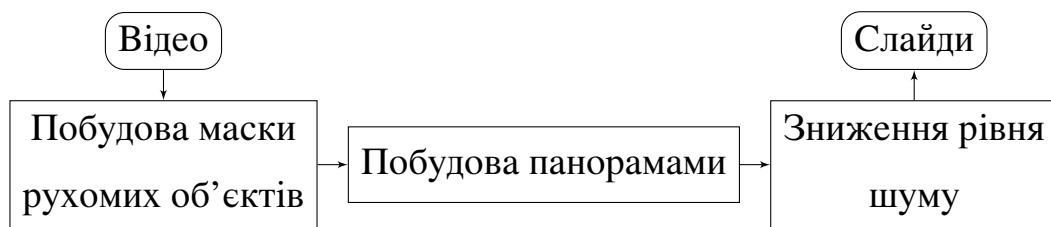


Рисунок 1.7 — Процедура створення панорамних слайдів

ється бінарна маска областей, де потенційно знаходяться рухомі об'єкти.

Ті частини зображення, де не було помічено рух, 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 панораму ми порівнюємо з тими панорамами, що було отримано на попередніх кроках, і створюємо новий слайд, якщо було помічено достатню кількість змін.

Інформаційна технологія, що реалізує даний алгоритм, працює в напівавтоматичному режимі – вона потребує від користувача введення деяких параметрів. Ці параметри визначають крок, з яким треба обирати кадри для визначення руху, ступінь згладжування маски рухомих об'єктів під час її пошуку, крок між панорамами для перевірки необхідності створення нового слайду та кількість необхідних змін між двома панорамами для створення нового слайду.

Висновки до розділу 1

Проведений аналіз існуючих аналогів технологій оцифрування дошки. Більшість алгоритмів працює з малим розширенням відео та не є досить швидкими. Було показано неефективність деяких методів, що застосовуються для прибирання людини біля дошки. Також була зроблена постановка задачі для досягнення мети. Коротко описаний запропонований алгоритм.

2 СТВОРЕННЯ МАСКИ ВИКЛАДАЧА

Другий розділ присвячено опису методів отримання маски людини, щоб в подальшому прибрати викладача з кадру. Запропоновано використання алгоритму Бойкова-Колмогорова, згорткових мереж та їх комбінацій YOLO, MobileNet, SSDLite, Faster R-CNN.

2.0.1 Вхід алгоритму

Для опису алгоритму створення панорамних слайдів потрібно визначити вхідні дані та параметри, які буде надавати користувач інформаційної технології.

Дошка - це плоска поверхня, яку знімає камера. Це може бути крейдяна дошка, маркерна дошка, стіна тощо.

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, при чому 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 це 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 помічав.

Кадром під номером i шириною w і висотою h називаємо відображення, де $P \rightarrow C$, де P - множина координат пікселів, C - скінченна множина можливих рівнів

яскравостей (інтенсивностей) пікселів.

$$P = [1, \dots, w] \times [1, \dots, h], C \subset R \quad (2.1)$$

,

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 $c \in C$, позначимемо $F_p^i = c$.

2.0.2 Прибирання викладача з відео

Під час лекцій часто виникають такі ситуації, коли викладач затуляє собою частину дошки з написами – наприклад, для запису нового матеріалу або щоб видалити старі написи з дошки. Іноді студенти просять викладача відійти від дошки, щоб переписати з неї те, що з'явилося на ній лише хвилину тому, проте наша програма дозволяє побачити частину дошки, яку затулив викладач, якщо перед цим перекривтий сегмент було добре видно протягом вказаного користувачем часу.

Якщо вирішити проблему прибирання викладача з відео, то студент завжди буде бачити, що відбувається на дошці.

Для вирішення цієї задачі ми застосовуємо декілька методів. Що значить:

- 1) Можна будувати маску рухомих об'єктів, щоб потім їх видаляти. Рухомими об'єктами можуть бути викладач, студенти, а також записи, що тільки-но з'явилися. Для цього ми вирішуємо задачу мінімального зрізу або максимального потоку шляхом використання алгоритму Бойкова-Колмогорова.
- 2) Застосувати нейронну мережу YOLO для знаходження людини і подальшої сегментації дошки та викладача.

Розпишемо теоретичні частини кожного зі способів.

2.0.3 Задача знаходження Min-Cut/Max-Flow

Ведемо позначення: T - множина вузлів графу (рис. 2.1), τ - множина направлених дуг, s - джерело (початок), e - стік (кінець), $N_t = \{t' : tt' \in \tau\}$, $P_t = \{t' : t't \in \tau\}$, f - потік, c - пропускна здатність,

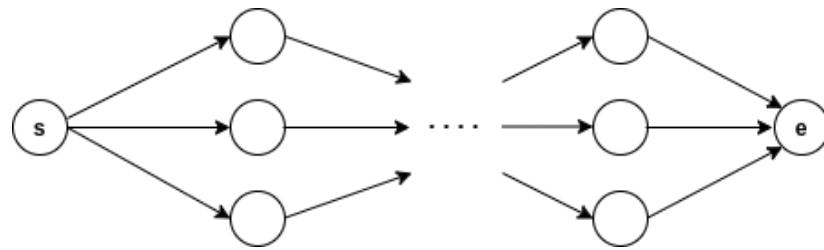


Рисунок 2.1 — Приклад графу

Задача максимального потоку

$$\sum_{t \in N_s} f_{st} \rightarrow \max_{f: \tau \rightarrow R} \quad (2.2)$$

З обмеженнями:

$$\begin{cases} f_{tt'} \leq c_{tt'}, & \forall tt' \in \tau, \\ \sum_{p \in P_t} f_{pt} - \sum_{t' \in N_t} f_{tt'} = 0, & \forall t \in T \setminus \{s, e\}, \\ \sum f_{tt'} \geq 0, & \forall tt' \in \tau \end{cases} \quad (2.3)$$

Що значить:

- 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.

Оригінальний алгоритм:

Algorithm 1 Алгоритм Min-Cut/Max-Flow

Вхід: Граф, з об'єктами $t \in T$, ребрами tt'

Вихід: $F_{maxflow}$ - значення максимального потоку;

Ініціалізація: $F_{maxflow} = 0$, $f_{tt'}^0 = 0 \quad \forall tt' \in \tau$.

Поки існує шлях з s в e повторюємо кроки 1,2:

Крок 1: Знаходимо шлях від s до e . (алгоритм Едмонса-Карпа або Форда Фалкерсона).

Відвідуємо t' із t якщо:

1. $f_{tt'} \neq c_{tt'}$
2. $\#p_{t'}^i \Rightarrow p_{t'}^i = t$ (запам'ятали вершину)
3. $t' \neq s$

Крок 2: Проходимо по заданому шляху:

1. Знаходимо $\Delta f^i = \min_{tt' \in \{se\}} f_{tt'}$
2. Змінюємо потік: $f_{tt'}^{i+1} = f_{tt'}^i + \Delta f^i$
3. Оновлюємо $F_{maxflow}$: $F_{maxflow}^{i+1} = F_{maxflow}^i + \Delta f^i$

Знайшовши максимальний потік можемо знайти мінімальний зріз: Для $\forall t \in T$ знайти $\theta_t \in \{0,1\}$

Крок 3: Запускаємо пошук в ширину або глибину вже з оновленим графом.

Якщо $c_{tt'} \geq f_{tt'}$ та $c_{tt'} \geq 0 \Rightarrow \theta_{t'} = 1$, інакше $\theta_{t'} = 0$

У 2004 році Юрій Бойков та Володимир Колмогоров запропонували модифікацію вищеописаного методу. Запропонована ідея методу полягає в огументації шляхів. Будується два дерева S та T коренями 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 складається зі стадій *росту, огументації, всиновлення*.

Коротко розглянемо кожну з них:

1) **Стадія Росту.**

Відбувається одночасний ріст дерев з вершин s та e , знаходяться активні вершини і додають їх як вершини, що відвідали. Після такого сканування верши-

ни стають внутрішніми. Цей процес продовжується поки не зостанеться не активної вершини.

2) Стадія Огментації.

На цій стадії шлях, що був знайдений на попередньому етапі огментується (розділено по ребру залишкової пропускної здатності (*bottleneck*). Якщо ребра дерева стають насиченими (пропускна здатність = потоку) найвіддаленіші вершини від коренів дерев стають *сиротами*. Тобто, якщо наприклад вершини t та t' знаходяться в дереві S і ребро (t, t') є насиченим, тоді t' називається S -*сиротою*. Аналогічно, якщо t та t' знаходяться в дереві T , то $t - T$ -*сирота*. Якщо ребро знаходиться в *bottleneck* (t в S , t' в T , ребро (t, t') насичене) відповідно немає ніяких сирот. Всі сироти потрапляють у список сирот.

3) Стадія Всиновлення.

На даному етапі ми проходимось по кожній сироті в списку сирот для кожного дерева. Нехай t' S -*сирота*. Знаходимо всі t в S , які формують ребро (t, t') . Для кожного такого t перевіряємо чи шлях з t в s містить сироти включаючи t . Якщо сироти 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.

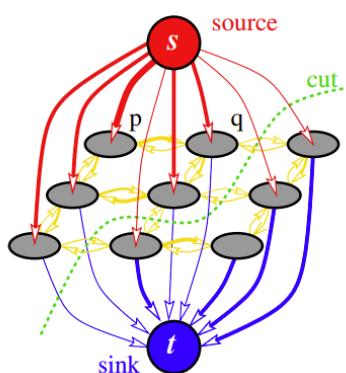


Рисунок 2.2 — Приклад графу-решітки з мінімальним розрізом

Найбільша перевага даного алгоритму це те що, він працює швидше та краще на графах-решітках (рис.2.2). Оскільки саме таку структуру ми використовуємо в

алгоритмі Б-К для отримання маски рухомих об'єктів це дає змогу оброблювати кадри відео досить швидко.

2.0.4 Застосування алгоритму Б-К для отримання маски рухомих об'єктів

Введемо поняття **маски рухомих об'єктів**.

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 $s > 0$ 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 проігнорованим 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.



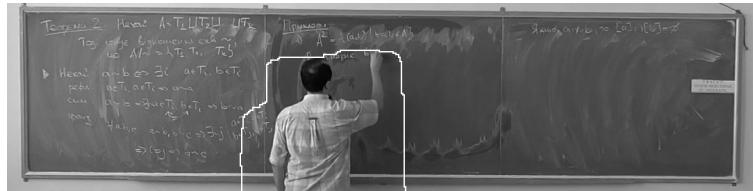
(a) Попередній кадр



(б) Поточний кадр



(в) Інвертована різниця попереднього і поточного кадрів



(г) Мaska рухомих об'єктів

Рисунок 2.3 — Процес створення маски з відео

Для того, щоб видалити всі рухомі об'єкти, які можуть перекривати дошку, ми беремо кадри F^i та F^{i+s} (рис. 2.3, (а), (б)) та дляожної пари кольорів пікселів p з однаковими координатами знаходимо модуль $D_p^i = |F_p^i - F_p^{i+s}|$ різниці інтенсивностей (Рис. 2 (в)). Зображення D^i подаємо на вхід алгоритму Бойкова-Колмогорова.

Треба знайти маску $B^i : P \rightarrow \{0,1\}$ рухомих об'єктів для кадрів F^i і F^{i+s} . Введемо функції.

$$q_p(B_p^i) = \begin{cases} \alpha D_p^i, & B_p^i = 0 \\ 255 - D_p^i, & B_p^i = 1 \end{cases} \quad (2.4)$$

$$g(B_p^i, B_{p'}^i) = \beta |B_p^i - B_{p'}^i| \quad (2.5)$$

де α та β – 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

$$E(B_p^i) = \sum_{p \in P} q_p(B_p^i) + \sum_{(p, p') \in \Gamma} g(B_p^i, B_{p'}^i) \quad (2.6)$$

На виході отримуємо маску B_p^i рухомих об'єктів (рис. 2.3(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.

Переваги методу:

- 1) Алгоритм Б-К досить швидко будує мінімальний розріз. Статистика на ОС Ubuntu 21.04 під Intel Core i5-7200U @ 4x 3,1GHz, час 0.15 секунди на кадрі розмірами 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. 2.4). Це може давати дефекти на панорамних знімках. В розділі № описано як була поборена дана проблема.

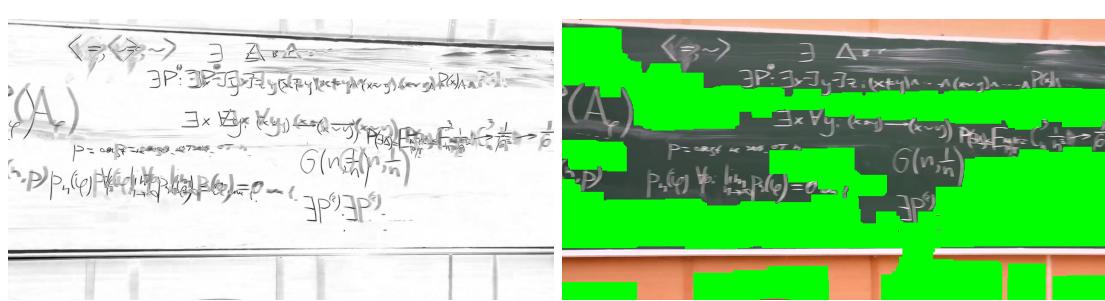
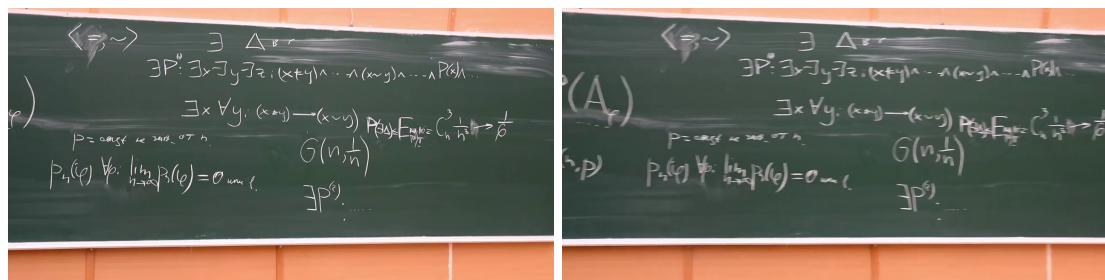


Рисунок 2.4 — Приклад отримання поганої маски під час руху камери

- 2) Якщо викладач практично не рухається довгий час, відповідно не потрапляє на маску рухомих об'єктів, то він може потрапити на результатуючий панорамний знімок.

2.1 Застосування нейронних мереж для отримання маски людини

Введемо поняття **маски людини (викладача)**.

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 так звані **згорткові нейронні мережі** (англ. *convolutional neural networks, CNN*). Це частина машинного навчання, яке називається глибоким навчанням, оскільки використовується більше двох нейронних шарів разом зі згортковими.

В таких мережах застосовується операція згортки(англ. *convolution*) та пулінгу (англ. *pooling*), батч-нормалізація, та різні функції активації такі як ReLU, Tanh.

Надалі послідовну комбінацію (згортка + батч-нормалізації + функція активації) будемо називати згортковим шаром, але кожний автор нейронної мережі створює свої шари відмінні від вищезазначеного.

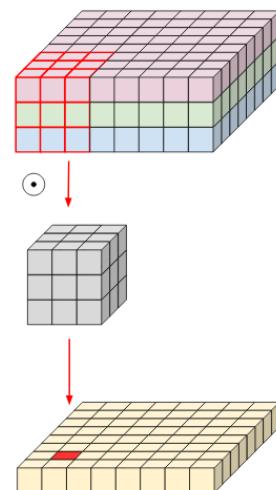


Рисунок 2.5 — Ілюстрація взяття звичайної згортки [?]

Розглянемо архітектури нейронних мереж, що застосувались у роботі для локалізації людини. Всі нищеписані нейронні мережі були використані вже з натренованими вагами у програмній бібліотеці PyTorch [?]. За мету було поставлено підібрати таку мережу, що здатна швидко оброблювати один знімок навіть на смартфоні. Тому головним критерієм є швидкість.

2.1.1 YOLO: You-Only-Look-Once

YOLO - це сімейство нейронних мереж.

YOLOv1 вперше представили Joseph Redmon. Вона розв'язує задачу детекції об'єктів як задачу регресії щодо просторового розділення знайдених областей об'єктів та їх ймовірностей. Її зараз широко використовують для класифікації та знаходження об'єктів оскільки вона здатна оброблювати відео в реальному часі 30 кадрів в секунду на мобільних пристроях, що є її найбільшою перевагою серед інших аналогів.

Опишемо коротко головні особливості YOLOv1, оскільки YOLOv5, яка застосувалась в роботі є лише модифікацією:

- 1) Спочатку зображення розділяється решіткою $S \times S$. Якщо центр об'єкту потрапляє в комірку решітки, тоді ця комірка є кандидатом, для подальшої локалізації об'єкта.
- 2) Кожна комірка решітки має передбачувати B областей та рівнів довіри. Даний рівень довіри показує на скільки модель впевнена, що дана комірка містить об'єкт та на скільки точна область. Рівень довіри визначається $Pr(\text{Object}) * IOU_{pred}^{truth}$, де $Pr(\text{Object})$ - ймовірність об'єкту, IOU_{pred}^{truth} - величина перетину передбаченої області об'єкту до її справжньої. Відповідно якщо модель не знайшла об'єкт цей рівень нульовий. Задача, щоб рівень довіри був якомога близчим до IOU_{pred}^{truth} .
- 3) Кожна область об'єкту складається з 5 чисел: рівень довіри, x, y - координати

центрю об'єкту, w, h - його ширина та висота. Кожна комірка передбачає C умовних ймовірностей $Pr(Class_i|Object)$

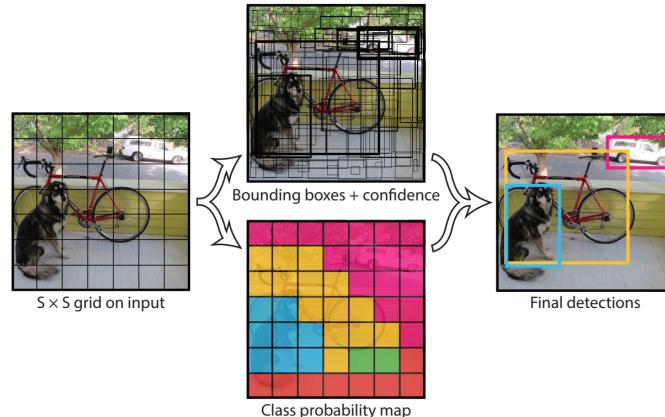


Рисунок 2.6 — Процес локалізації об'єктів мережею YOLO [?]

Для тренування мережі використовують суму 4 штрафних функцій.

$$\begin{aligned}
 & \lambda_{coord} \underbrace{\left(\sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{obj} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2] \right)}_{\text{по координатам центру}} \\
 & + \underbrace{\sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{obj} [(\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2]}_{\text{ширини та висоти об'єкту}} \\
 & + \underbrace{\sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{obj} (C_i - \hat{C}_i)^2 + \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{noobj} (C_i - \hat{C}_i)^2}_{\text{точності класифікації}} \\
 & + \underbrace{\sum_{i=0}^{S^2} \mathbb{1}_i^{noobj} \sum_{c \in classes} (p_i(c) - \hat{p}_i(c))^2}_{\text{ймовірності класів}} \quad (2.7)
 \end{aligned}$$

, де $\mathbb{1}_i^{obj}$ визначає чи знайшовся об'єкт в комірці i , а $\mathbb{1}_{ij}^{obj}$ 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.

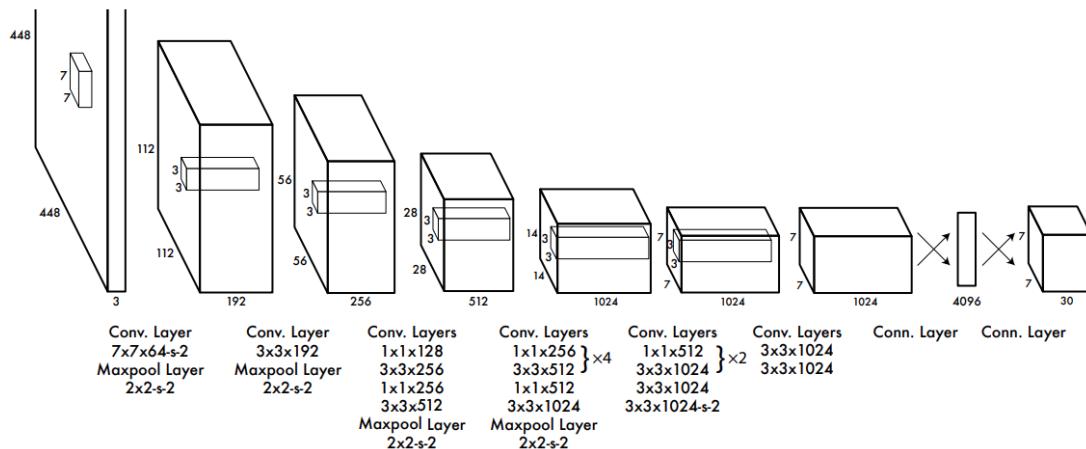


Рисунок 2.7 — Архітектура YOLOv1 [?]

На момент написання була використана одна з мереж YOLOv5, яка була розроблена Glenn Jocher вже на програмній бібліотеці PyTorch.

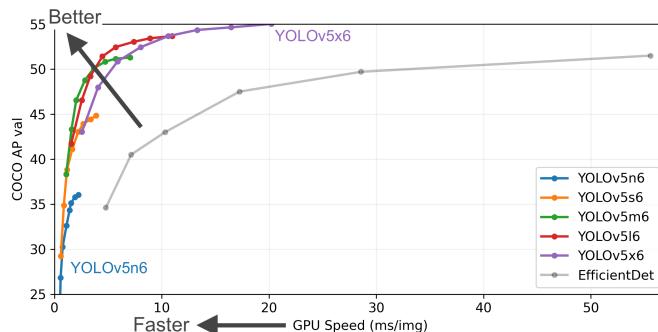


Рисунок 2.8 — Графік залежності точності на датасеті COCO від швидкості обробки однієї картинки різними мережами YOLOv5 н

2.1.2 MobileNet

MobileNet це також ще одне сімейство, що часто використовується у комп’ютерному зорі. Вперше була представлена у 2017 році науковцями з Google ???. Мережі даної категорії теж зробили свою революцію у обчисленні глибоких шарів з використанням мінімальних обчислювальних ресурсів. Були запропоновані два гіперпараметри, змінюючи які, можна отримати приrostи в швидкості та точності. Мережі

MobileNet застосовуються для локалізації, класифікації об'єктів і також для широкомасштабної гео-локалізації.

Розглянемо особливості кожної версії MobileNet.

2.1.2.1 MobileNetv1

Одним 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*).

Нехай маємо: - квадратне зображення I розмірами $S_I \times S_I \times M$: широта, висота та кількість каналів відповідно. - ядро C розмірами $S_C \times S_C \times M \times N$, де N вихідна розмірність отриманої згортки. - вихідна згортка C розмірами $S_K \times S_K \times N$

Тоді формулу звичайної згортки (Рис. ??) можна записати як:

$$G_{k,l,n} = \sum_{i,j,m} C_{i,j,m,n} \cdot F_{k+i-1, l+j-1, m} \quad (2.8)$$

Щоб обчислити дану згортку потрібно $S_C \cdot S_C \cdot M \cdot N \cdot S_I \cdot S_I$ операцій, що є створює обчислювальні обмеження на мобільний пристрій, якщо наприклад матимемо декілька таких згорток. Для вирішення даної проблеми застосовується глибинна згортка (Рис.). Вона полягає у отриманні окремої згортки кожного каналу ядра до кожного каналу зображення. Нехай \widehat{C} ядро глибинної згортки, тоді маємо:

$$\widehat{G}_{k,l,n} = \sum_{i,j,m} \widehat{C}_{i,j,m,n} \cdot F_{k+i-1, l+j-1, m} \quad (2.9)$$

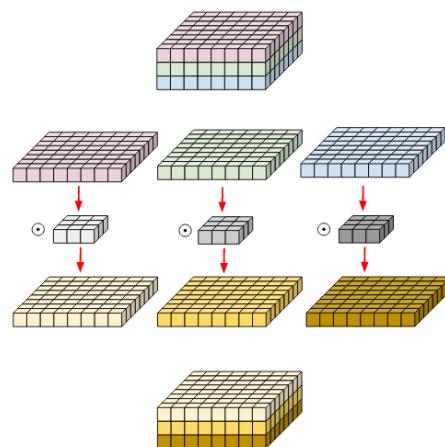


Рисунок 2.9 — Ілюстрація взяття глибинної згортки [?]

Для обчислення вже цієї згортки треба $S_C \cdot S_C \cdot M \cdot S_I \cdot S_I$.

Як ми бачимо ми вже позбулись N операцій, але маємо пам'ятати, що раз \widehat{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 1×1 . 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.

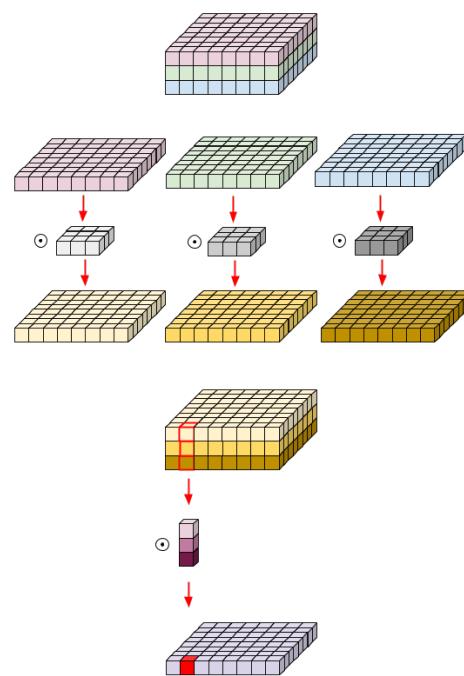


Рисунок 2.10 — Ілюстрація взяття глибинно просторової згортки [?]

Таке нововведення до глибинного навчання дало змогу в рази пришвидшити навчання та швидкість нейронної мережі MobileNetv1. Вона використовує простотрію глибинні згортки розміром ядра 3×3 . Таким чином маємо таку заміну блоку.

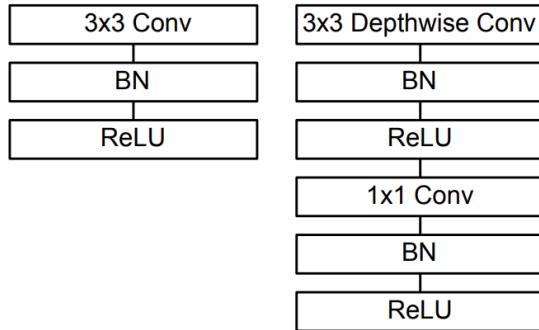


Рисунок 2.11 — Ліворуч звичайний згортковий шар, праворуч згортковий шар у MobileNetv1 [?]

Також варто відмітити ще одну особливість MobileNetv1, а саме два гіперпараметри ширини та розмірності. Множник ширини α застосовується, щоб зменшити кожен шар мережі, що в свою чергу дає приріст у швидкості. З 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 ρ зменшує вхідну картинку і відповідно розмірність згорток. Разом з α 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.

Повна архітектура мережі MobileNetv1 виглядає так:

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 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

Рисунок 2.12 — Архітектура нейронної мережі MobileNetv1 [?]

2.1.2.2 Mobilenetv2

Вже у 2018 році з'являється 2-га версія мережі MobileNet. Тут автори додали нові зміни в архітектуру такі як інвертовані залишки (англ. *Inverted Residuals*) та лінійні вузькі містя (англ.*Linear Bottlenecks*). Також представили застосування особливостей MobileNetv2 у фреймворку для локалізації SSDLite, що теж був використаний в даній роботі.

Автори зробили дослідження щодо застосування функції ReLU (rectified linear unit) в контексті низьких розмірностей. Вводиться поняття *manifold of interest* різноманітність інтересів - це множина шарів активацій. Була висунута гіпотеза про те що, різноманітність інтересів з високою розмірністю можна стиснути у підпростір меншої розмірності при чому зі збереженням інформації.

Розпишемо детальніше з яких кроків складається новий структурний шар:

- 1) Нехай на вході зображення розмірами $S_I \cdot S_I \cdot M$, його приймає перший розширяючий блок. Головна особливість цього блоку є новий параметр t , що називається розширяючим фактором. Найкращими значеннями для нього є $[5, 10]$. Менші значення краще застосовувати для меншої мережі, а більші відповід-

но для більших. Саме тут можна побачити різноманітність інтересів. Вихід $S_I \cdot S_I \cdot (t \cdot M)$.

- 2) Наступним кроком є вже відома глибинна згортка з функцією активації $ReLU6(x) = \min(\max(0, x), 6)$. Вхід $S_I \cdot S_I \cdot (t \cdot M)$, а вихід $(S_I / \text{stride}) \cdot (S_I / \text{stride}) \cdot (t \cdot M)$, де stride - шаг згортки.
- 3) Далі застосовується точкова згортка, щоб створити єдиний тензор. Як можна побачити на Рис. 2.13 також застосовуються переваги residual block, де вхід в шар поєднується з передостаннім блоком.

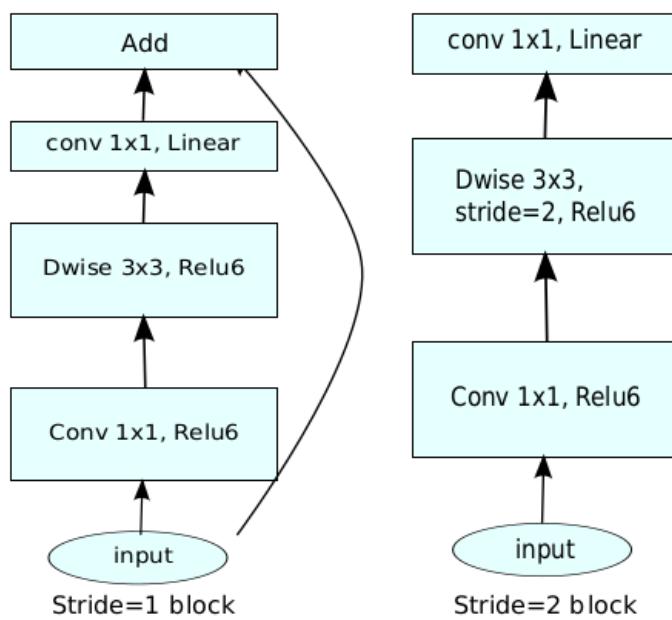


Рисунок 2.13 — Відмінність MobileNetv2 від MobileNetv1: ліворуч структурний блок MobileNetv2, праворуч - MobileNetv1 [?]

Input	Operator	t	c	n	s
$224^2 \times 3$	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	-	k	-	-

Рисунок 2.14 — Архітектура мережі MobileNetv2 [?]

Також варто відмітити, що автори вдосконалили мережу SSD (Single Shot Detector), він подібний до YOLO і застосовується для локалізації об'єктів. Автори замінили звичайні згорткові шари на розширяючі згорткові, які використовуються в MobileNetv2. Таким чином створивши досконалішу версію SSDLite, про це буде описано пізніше.

2.1.2.3 MobileNetv3

На даний час існує остання 3-тя версія [?] архітектури сімейства нейронні мереж MobileNet. Тут автори представили вже 2 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 % точніша за MobileNetv2, а локалізація об'єктів з MobileNetV3-Large на COCO датасеті на 25 % швидша при тій же точності. Головний блок мережі знову змінився у 3-ій версії порівняно з 2-ої. Окрім того що тут опрацьовуються як ідеї MobileNetv1, MobileNetv2 також автори взяли до уваги певні особливості мережі MnasNet [?], в якій з'явився блок стиснення та збудження (англ. squeeze and excitation). Мережі з такими блоками називаються Se-Nets.

Мета підходу стиснення та збудження ?? полягає в тому, щоб взяти параметри виходу згортки (нехай це матриця u_c розмірами $C \times H \times W$) як вхід блоку стиснення та збудження. Вихід згортки u_c буде оброблений блоком стиснення та збудження, який отримає вхід u_c та параметри виходу згортки u_c .

ння та збудження, потім зробити операцію стиснення (маємо $1 \times 1 \times C$), операцію збудження ($1 \times 1 \times C$). Далі потрібно масштабувати параметри щоб поєднати u_c . Ідея полягає в тому, щоб підвищити чутливість мережі до інформативних ознак і передавати їх до наступного кроку.

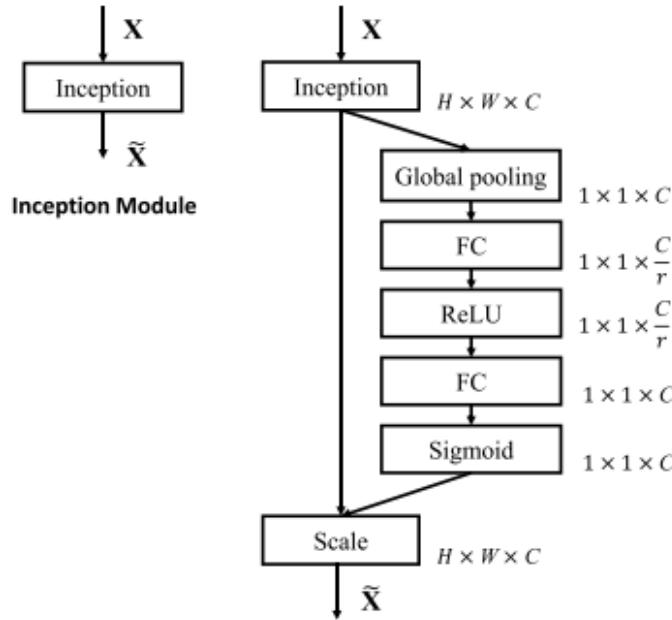


Рисунок 2.15 — Блок стиснення та збудження [?]

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

$$z = F_{sq}(u_c) = \frac{\sum_{i=0}^H \sum_{j=0}^W u_c(i,j)}{H \cdot W} \quad (2.10)$$

Вихід $1 \times 1 \times C$.

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

$$s = F_e(x(z, W)) = \text{Sigmoid}(FC_2 \text{ReLU}(FC_1 z)) \quad (2.11)$$

Перший лінійний шар FC_1 використовується для зменшення розмірності z з деяким множником r , тому після цього шару розмірність буде $1 \times 1 \times C/r$, а FC_2 навпаки для збільшення $ReLU(W_1 z)$. Знаючи, що значення сигмоїди від 0 до 1, то можна масштабувати її вихід із входом в блок u_c

$$\tilde{x}_c = F_{scale}(u_c, s_c) = s_c \cdot u_c \quad (2.12)$$

Автори покрашили 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

$$swish(x) = x \cdot Sigmoid(x) \quad (2.13)$$

використати

$$h - swish(x) = x \cdot \frac{ReLU6(x + 3)}{6} \quad (2.14)$$

Це пов'язано з тим, що сигмоїда складна в обчисленні для малопотужних пристроїв. Також автори помітили, що використання $h - swish(x)$ дає приріст в точності, якщо її використовувати у глибоких шарах мережі.

Так само як в MnasNet автори MobileNetv3 автори звернулись за допомогою до платформи NAS (neural architecture structure), яка створена для підбирання глобальних параметрів мережі. Тобто NAS рекомендує найкращу знайдену архітектуру, а далі NetAdapt (схожий до NAS) допомагає у підбиранні параметрів вже всередині одного обчислювального блоку.

Всі вищеописані техніки допомогли створити нову MobileNetv3-Small та MobileNet Large (Рис. 2.16).

Input	Operator	exp size	#out	SE	NL	s	Input	Operator	exp size	#out	SE	NL	s
$224^2 \times 3$	conv2d	-	16	-	HS	2	$224^2 \times 3$	conv2d, 3x3	-	16	-	HS	2
$112^2 \times 16$	bneck, 3x3	16	16	-	RE	1	$112^2 \times 16$	bneck, 3x3	16	16	✓	RE	2
$112^2 \times 16$	bneck, 3x3	64	24	-	RE	2	$56^2 \times 16$	bneck, 3x3	72	24	-	RE	2
$56^2 \times 24$	bneck, 3x3	72	24	-	RE	1	$28^2 \times 24$	bneck, 3x3	88	24	-	RE	1
$56^2 \times 24$	bneck, 5x5	72	40	✓	RE	2	$28^2 \times 24$	bneck, 5x5	96	40	✓	HS	2
$28^2 \times 40$	bneck, 5x5	120	40	✓	RE	1	$14^2 \times 40$	bneck, 5x5	240	40	✓	HS	1
$28^2 \times 40$	bneck, 5x5	120	40	✓	RE	1	$14^2 \times 40$	bneck, 5x5	240	40	✓	HS	1
$14^2 \times 80$	bneck, 3x3	200	80	-	HS	1	$14^2 \times 40$	bneck, 5x5	120	48	✓	HS	1
$14^2 \times 80$	bneck, 3x3	184	80	-	HS	1	$14^2 \times 48$	bneck, 5x5	144	48	✓	HS	1
$14^2 \times 80$	bneck, 3x3	184	80	-	HS	1	$14^2 \times 48$	bneck, 5x5	288	96	✓	HS	2
$14^2 \times 80$	bneck, 3x3	480	112	✓	HS	1	$7^2 \times 96$	bneck, 5x5	576	96	✓	HS	1
$14^2 \times 112$	bneck, 3x3	672	112	✓	HS	1	$7^2 \times 96$	bneck, 5x5	576	96	✓	HS	1
$14^2 \times 112$	bneck, 5x5	672	160	✓	HS	2	$7^2 \times 96$	conv2d, 1x1	-	576	✓	HS	1
$7^2 \times 160$	bneck, 5x5	960	160	✓	HS	1	$7^2 \times 576$	pool, 7x7	-	-	-	-	1
$7^2 \times 160$	bneck, 5x5	960	160	✓	HS	1	$1^2 \times 576$	conv2d 1x1, NBN	-	1024	-	HS	1
$7^2 \times 160$	conv2d, 1x1	-	960	-	HS	1	$1^2 \times 1024$	conv2d 1x1, NBN	-	k	-	-	1
$1^2 \times 960$	pool, 7x7	-	-	-	-	1							
$1^2 \times 960$	conv2d 1x1, NBN	-	1280	-	HS	1							
$1^2 \times 1280$	conv2d 1x1, NBN	-	k	-	-	1							

(б) MobileNetv3-Small

(а) MobileNetv3-Large

Рисунок 2.16 — Дві архітектури мережі MobileNetv3 [?]

2.1.2.4 SSD: Single Shot Multibox Detector

Оскільки для відокремлення ознак застосовується в цій роботі і SSD із структурними елементами MobileNetv3, тому є причина показати як саме SSD вдається локалізовувати об'єкти та його відмінності з YOLO.

Мережа SSD (Рис. 2.17) на відміну від YOLO проганяє зображення через шари лише один раз. Це пояснюється словами в назві Single Shot. SSD використовує задачу MultiBox регресії локалізованих областей. Розпишемо її детальніше.

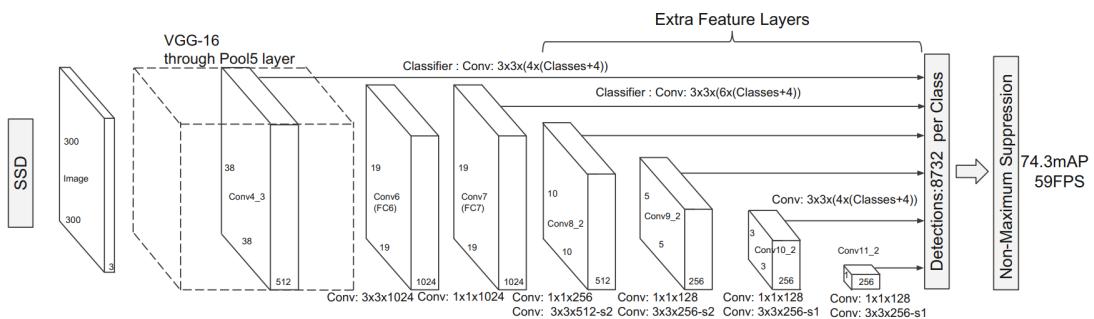


Рисунок 2.17 — Архітектура мережі SSD на основі VGG-16 [?]

Головна задача MultiBox це оцінювати як точно локалізувати об'єкти. Тут використовується дві штрафні функції: довіри, що основана на категоріальній пере-

хресній ентропії та локалізації з L2 нормою, яка показує як точно справжня область об'єкту співпадає із передбаченою.

Варто також помітити, що SSD застосовує вже фіксовані області для подальшого передбачення із згортками малого ядра. Чим більше фіксованих областей, то тим більша точність локалізації об'єкту, але тим довша обробка фото.

Принцип відокремлення інформативних ознак SSD полягає у розбитті зображення на решітку (Рис. 2.18) (аналогічно YOLO). Запускається класифікатор і вирішує чи брати ту чи іншу клітинку кандидатом на локалізацію. Для кожної фіксованої області обчислюється розмір області та рівень довіри дляожної категорії об'єктів.

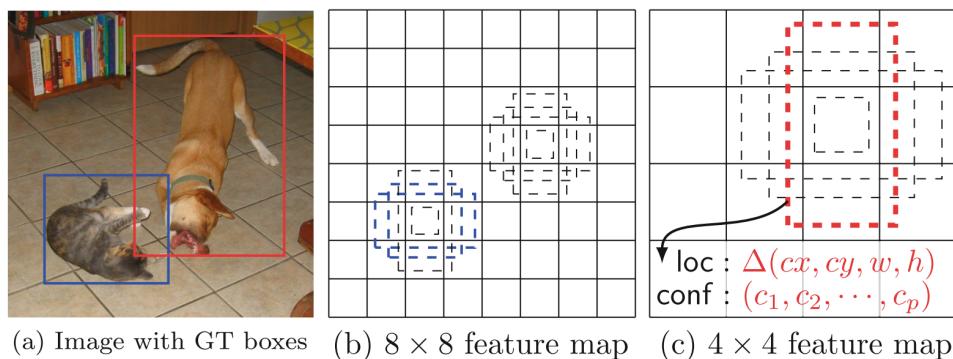


Рисунок 2.18 — Приклад роботи SSD для локалізації об'єктів [?]

2.1.2.5 Faster R-CNN

Говорячи про детекцію і класифікацію об'єктів не можна не згадати про також відомий клас згорткових нейронних мереж як R-CNN (region-based convolutional neural network).

Перша мережа R-CNN складалась з 3 незалежних один від одного етапів. На першому генерується 2000 пропозицій областей за допомогою вибіркового пошуку. Далі ці області проходять стиснення розміру до якогось фіксованого. Останній етап це метод опорних векторів вже з натренованими вагами, який і робить класифікацію.

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

Багато проблем було вирішено вже з новою Fast R-CNN. В ній відсутні етапи як такі, а вся архітектура складається з одного модуля, що в рази полегшує навчання мережі. Тут також вводиться новий шар ROI Pooling, головна ціль якого надати вектори ознак фіксованої довжини. Даний шар розбиває кожний запропонований регіон на решітку, в кожній клітинці якій потім знаходиться максимальне значення (операція max pooling). Але навіть якщо і Fast R-CNN швидша за R-CNN в ній досі лишився вибірковий алгоритм.

У Faster R-CNN [?], який саме теж тестувався для локалізації людини в цій роботі, запропоновано використовувати 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 поширення ознак (feature sharing).

RPN ?? приймає на вхід картинку, і повертає множину координат прямокутних областей як подальших кандидатів для класифікації з їхніми мірами присутності об'єкту. RPN - повнозв'язана згорткова мережа, що значить в ній немає лінійних шарів. Саме вона стала заміною вибіркового алгоритму у Fast R-CNN.

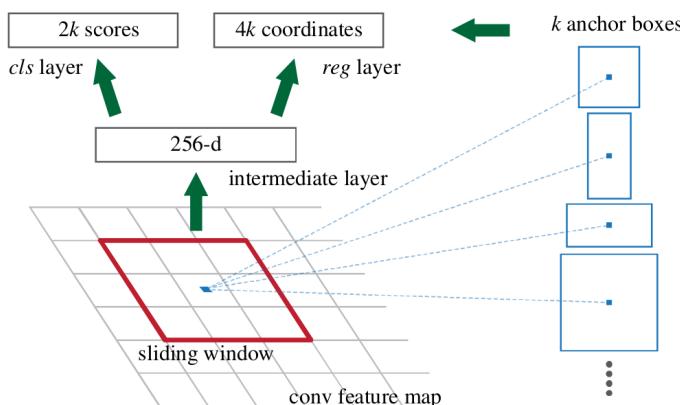


Рисунок 2.19 — Архітектура мережі RPN [?]

У RPN ми застосовуємо принцип ковзкого вікна (sliding window) на згортко-

вій мапі ознак, отриманої після останнього згорткового шару. Кожне таке вікно відображається на вектор меншої розмірності. Для кожного вікна генеруються прямокутні області-кандидати, що називаються якірними 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 якірних регіонів дозволяє детектувати об'єкти практично будь-якого масштабу та пов'язувати ознаки RPN з Fast R-CNN.

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

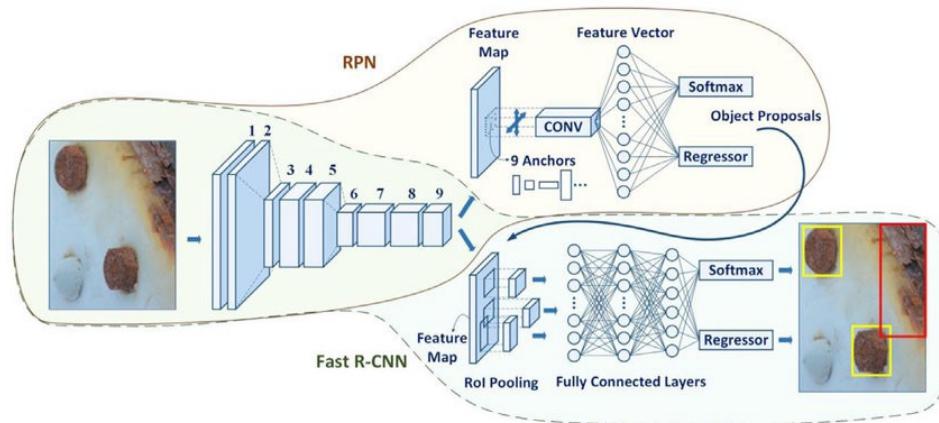


Рисунок 2.20 — Архітектура мережі Faster R-CNN [?]

Висновки до розділу 2

Було оглянуто згорткові мережі YOLO, MobileNet, SSD, R-CNN як спосіб детектування та локалізації людини в створенні слайдів. Описано архітектури, переваги та недоліки кожної згорткової мережі. Дані алгоритми є досить швидкими, щоб створювати інформаційну технологію побудови слайдів на мобільних пристроях.

3 СТВОРЕННЯ ПАНОРАМИ

Третій розділ присвячено опису пошуку відповідності між кадрами, матриці гомографії, та створеню кінцевого панорамного знімку без викладача.

3.1 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. Руки 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.

Ми можемо використовувати дескриптори ознак, такі як SIFT, SURF [13] або ORB [14], щоб знайти ключові точки (feature points). Був обраний SIFT, оскільки під час експериментів він надав візуально кращі результати, ніж інші алгоритми.

3.1.1 SIFT

У 1999 році англієць Девід Лоу представив алгоритм SIFT (Scale-invariant feature transform) ,укр. *Масштабозалежне перетворення ознак*. Даний алгоритм застовується для знаходження ключових точок (локальних ознак) зображення. Метод до-

сить 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 кроків для побудови 3D-карт, реконструкції стереопар та детектингу об'єктів.

Коротко розпишемо 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 просторово та масштабно інваріантне ядро оператора Гаусса з операцією конволюції 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 гаусіан (DoG метод) з константою k :

$$D(x,y,\sigma) = (L(x,y,k\sigma) - L(x,y,\sigma)) \quad (3.1)$$

Тобто початкове зображення поступово піддається конволюціям гаусіанів з константою k на кожну епоху. При чому кожна епоха відрізняється між собою значенням $k = 2^{1/s}$, де s число, що показує на скільки інтервалів розділити кожну епоху. В стеку однієї епохи зберігається $s + 3$ зображень. Коли епоха завершиться, то створюється нове Гаусове зображення з двох найвищих в стеку, взяттям кожного 2-го пікселя рядка і колонки.

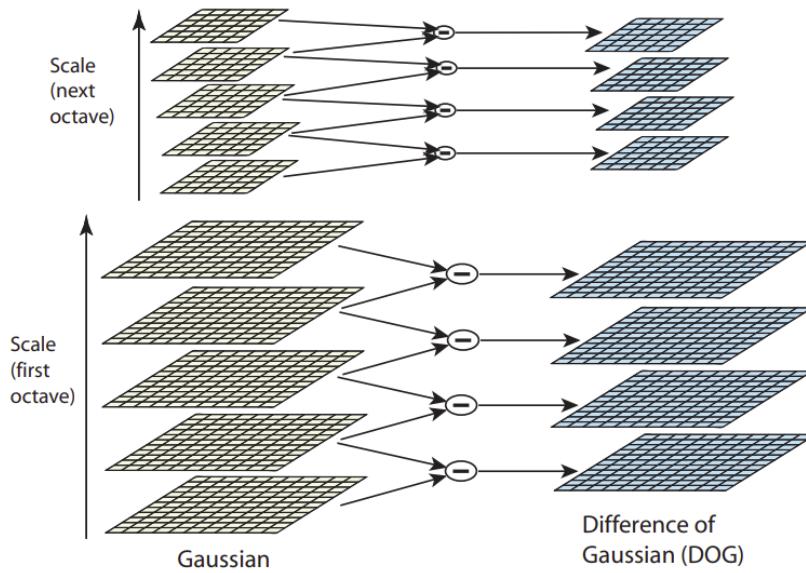


Рисунок 3.1 — Знаходження різниць гаусианів 1-го етапу SIFT

Пошук локальних екстремумів

Для знаходження локальних екстремумів зображення різниці гауссіан якоєвь точки беруться 8 сусідів цього ж зображення і 9 зображення різниць зверху та знизу. Точка є екстремум якщо вона значення її більше або менше за значення всіх її сусідів.

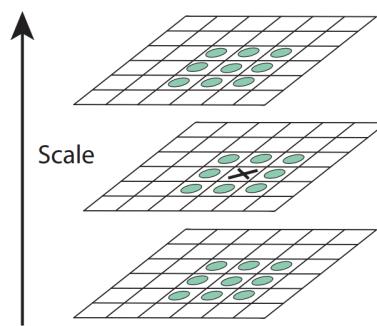


Рисунок 3.2 — Знаходження різниць гаусианів 1-го етапу SIFT

Частота вибірки по масштабу

Тут автор обґрунтовує кількість того, скільки разів потрібно робити зміну масштабу зображення. Експерименти показують, що на даному етапі метод дає велику кількість кандидатів екстремумів, і що це дуже важко обрахувати їх всі.

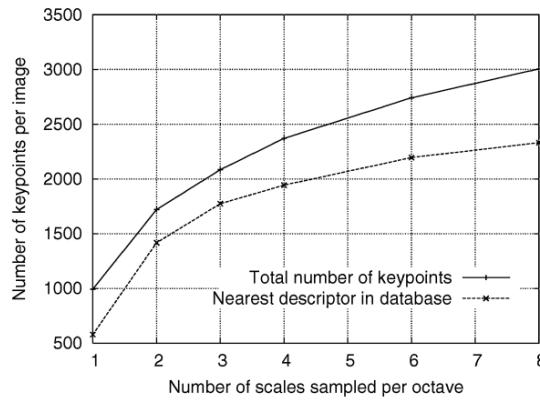


Рисунок 3.3 — Графік залежності кількості змін масштабу на кожну епоху до кількості ключових точок на зображення

Частота вибірки по простору

Пропонується використовувати оптимальне значення $\sigma = 1.6$ при якому маємо найбільший відсоток співпадіння екстремумів при багатократному повторенні експерименту.

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

$$D(\mathbf{x}) = D + \frac{\partial D^T}{\partial \mathbf{x}} \mathbf{x} + \frac{1}{2} \mathbf{x}^T \frac{\partial^2 D}{\partial \mathbf{x}^2} \mathbf{x} \quad (3.2)$$

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

$$\hat{\mathbf{x}} = \frac{\partial^2 D^{-1}}{\partial \mathbf{x}^2} \frac{\partial D}{\partial \mathbf{x}} \quad (3.3)$$

Підставивши 3.3 у 3.2 маємо можливість відсіяти нестабільні екстремуми по модулю значення (рис. 3.3):

$$D(\hat{\mathbf{x}}) = D + \frac{1}{2} \frac{\partial D^T}{\partial \mathbf{x}} \hat{\mathbf{x}} \quad (3.4)$$

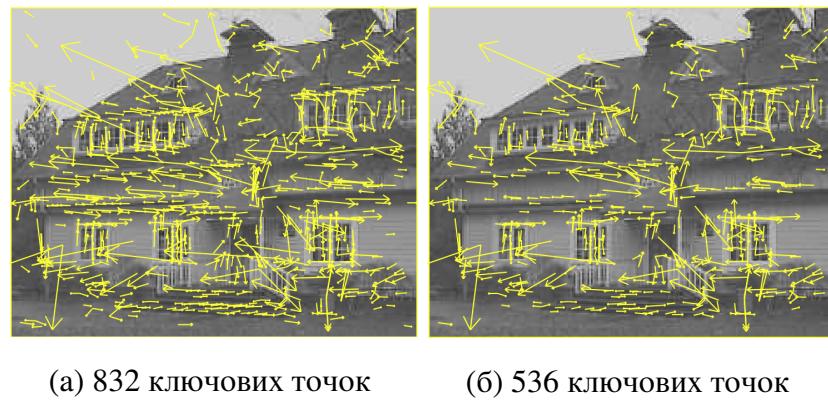


Рисунок 3.3 — Приклад відсіювання екстремумів

Тобто обмежуючи $|D(\hat{\mathbf{x}})| < \alpha$. Якщо кожен піксел в діапазоні $[0,1]$, то і $\alpha \in [0,1]$.

3) Визначення орієнтації градієнтів

Доожної точки визначається декілька орієнтацій градієнтів. Магнітуда градієнта по сусідам $m(x,y)$ та його орієнтація $\theta(x,y)$:

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2} \quad (3.5)$$

$$\theta(x,y) = \tan^{-1}(L(x,y+1) - L(x,y-1)) / (L(x+1,y) - L(x-1,y)) \quad (3.6)$$

, де $L(x,y)$ - значення згладженого гаусового зображення з найближчим масштабом.

4) Дескриптор точок

Локальні градієнти обчислюються для кожного масштабу навколоожної ключової точки. Створюються дескрипториожної ключової точки.

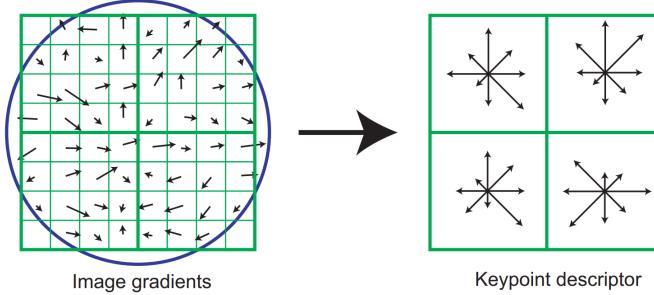


Рисунок 3.4 — Створення дескрипторів точок

3.1.2 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 їй координата на кадрі номер $i + s$. Відповідними точками є ті, що є проекціями однієї і тієї ж точки у просторі.

Нехай D^i та D^{i+s} множина дескрипторів кадрів F^i та F^{i+s} відповідно. Мета – отримати $M^{i,i+s}$ множину усіх знайдених відповідних точок між кадрами F^i і F^{i+s} .

Був використаний **Brute Force Matcher** 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 $D^{i,i+s}$

$\forall i \in D^i$:

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

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

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

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 першого зображення шукаємо найближчий на другому. Маючи список відповідних дескрипторів можна знайти відповідні точки на обох зображеннях.

3.1.3 Знаходження гомографії за допомогою RANSAC методу

Гомографія - матриця, що описує зв'язок точок між двома зображеннями площею поверхні називається.

$$H = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{13} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \quad (3.7)$$

$$H^i \cdot p_j^{i+s} = c \cdot p_j^i, \quad \forall i = \overline{1,4} \quad (3.8)$$

Рівняння 3.8 можна легко розв'язати відносно невідомої гомографії H^i . Цю матрицю можна використовувати 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 цілочисельні, їх можна округлити, що не матиме значного негативного впливу на якість результату.

Знаходження H вимагає знаходження всіх 9 параметрів матриці 3.7. 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 3.8 у матричній формі для T та T' :

$$\begin{bmatrix} x'_i \lambda \\ y'_i \lambda \\ \lambda \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{13} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}, \quad \forall i, j = \overline{1,4} \quad (3.9)$$

Такми чином ми приходимо до розв'язку системи:

$$\underbrace{\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1x'_1 - y_1x'_1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x_2x'_2 - y_2x'_2 \\ x_3 & y_3 & 1 & 0 & 0 & 0 & -x_3x'_3 - y_3x'_3 \\ x_4 & y_4 & 1 & 0 & 0 & 0 & -x_4x'_4 - y_4x'_4 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1y'_1 - y_1y'_1 \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -x_2y'_2 - y_2y'_2 \\ 0 & 0 & 0 & x_3 & y_3 & 1 & -x_3y'_3 - y_3y'_3 \\ 0 & 0 & 0 & x_4 & y_4 & 1 & -x_4y'_4 - y_4y'_4 \end{bmatrix}}_{A} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \end{bmatrix} = h_{33} \begin{bmatrix} x'_1 \\ x'_2 \\ x'_3 \\ x'_4 \\ y'_1 \\ y'_2 \\ y'_3 \\ y'_4 \end{bmatrix} \quad (3.10)$$

Варто підмітити, що ми потім позбуваємося h_{33} , щоб H була нормованою, оскільки вона зв'язує точки площин, в яких 3-тя координата 1. 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 9×9 . Параметри H знаходяться в останньому рядку V^* .

$$H = V_{9,:}^*/V_{9,9}^*, \quad H \in R^{1 \times 9} \rightarrow H \in R^{3 \times 3} \quad (3.11)$$

Для оцінки гомографії використовували RANSAC [?]. Користувач вводить рівень $\varepsilon > 0$ дозволеної похибки розрахунків – що менше, то краще, проте тим довше буде працювати алгоритм пошуку гомографії.

$$E(H) = \sum_{(x,x') \in M^{i,i+s}} [\| \frac{H \cdot x'}{(H \cdot x')_z} - x \| \leq \varepsilon] \rightarrow \max_{\substack{H \in \mathbb{R}^{3 \times 3} \\ \det H \neq 0}} \quad (3.12)$$

Algorithm 3 Алгоритм знаходження гомографії за принципом RANSAC

Вхід: $M^{i,i+s}$ - множина відповідних точок, ε 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 3.11
3. Обчислюємо $E(H)$ по формулі 3.12
4. Якщо $E(H) < E(H^{best})$:

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