

## **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 . . . . .	18
2 ****Назва розділу. . . . .	19
2.1 Пошук відповідності між кадрами . . . . .	27

## 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 занять. Було б добре мати технологію, що перетворюватиме відео лекцію у стисле слайд шоу, на якому будуть написи з дашочки 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 реалізувано 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. ??).

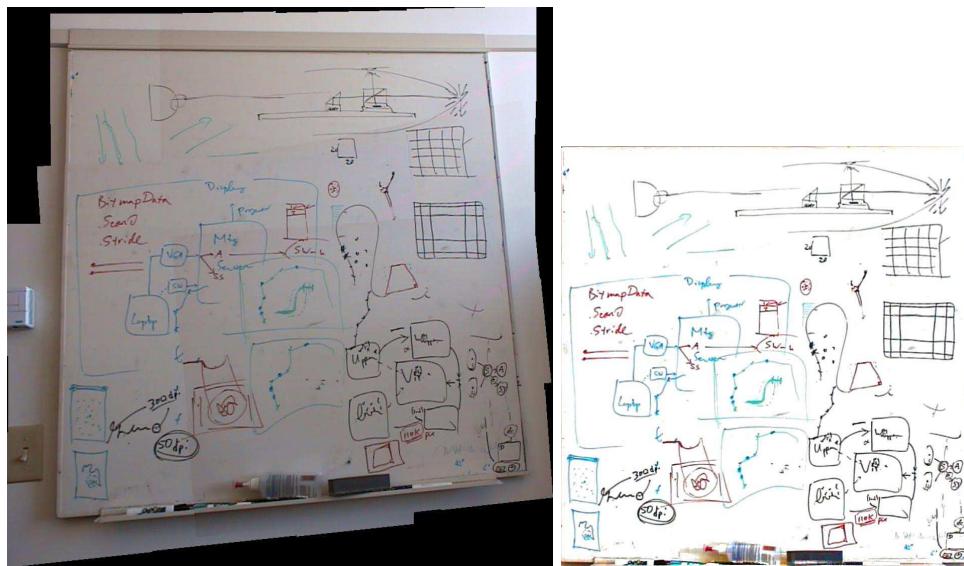


Рисунок 1.1 — Демонстрація роботи алгоритму інженерів з Microsoft

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.

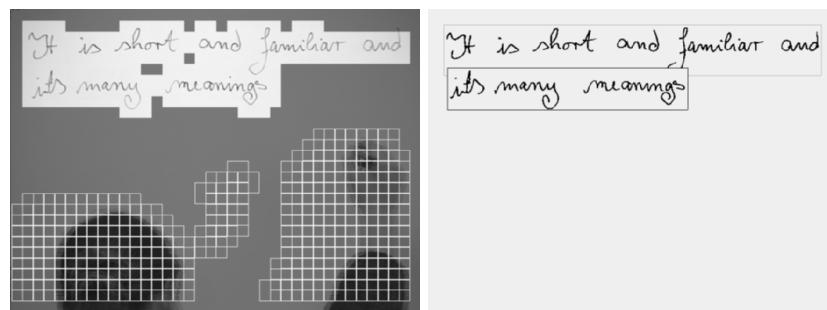


Рисунок 1.2 — Демонстрація роботи сканування дошки

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.

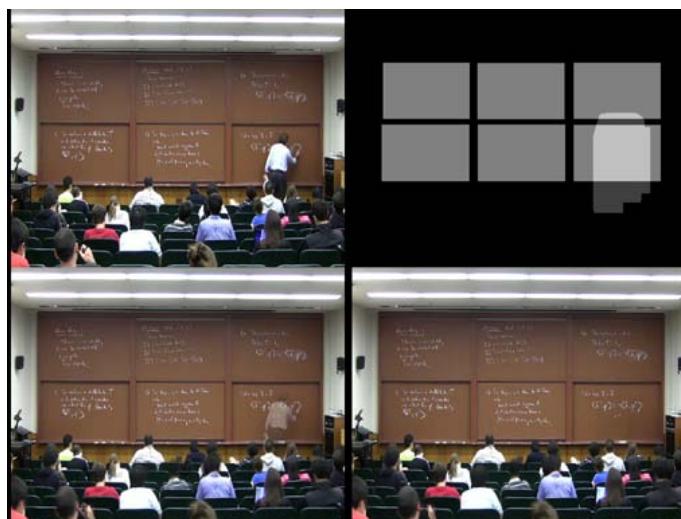


Рисунок 1.3 — Приклад роботи авторів Стенфорду

Головною особливістю даної роботи є те, що алгоритм автоматично локалізує різну кількість дошок. Однак варто відмітити, що тестування відбувалось на відеолекціях 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.

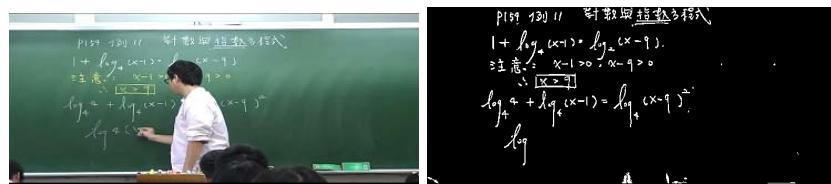


Рисунок 1.4 — Демонстрація роботи сканування дошки

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 викремлення написів і викладача. Відбувається видалення не самого викладача, а його контурів, після бінаризації картинки. Варто відмітити, що і тут камера має бути нерухомою.

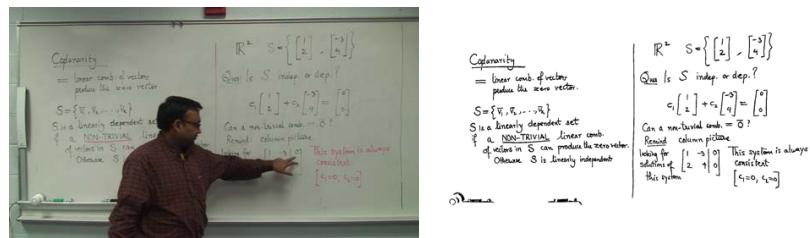


Рисунок 1.5 — Демонстрація роботи сканування дошки

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.

\*\*\*\*\*Всавити 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 та розводів 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

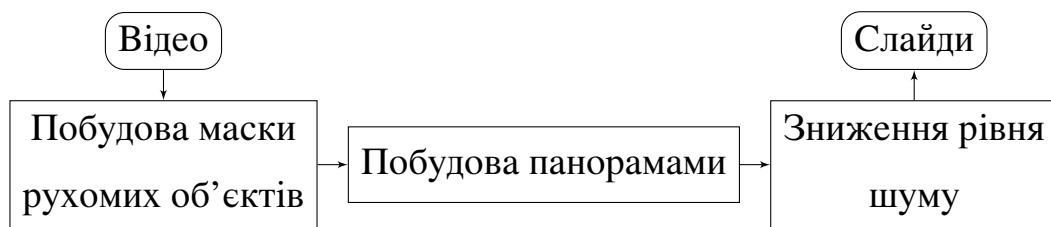


Рисунок 1.7 — Процедура створення панорамних слайдів

поточний на попередні кадри. Між кожною парою сусідніх кадрів відео розраховується 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 \*\*\*\*НАЗВА РОЗДІЛУ

Другий розділ присвячено опису задач методів їх розв'язку.

### 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  $c \in C$ , позначимо  $F_p^i = c$ .

### 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 перекритий 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'явились. Для цього ми вирішуємо задачу мінімального зрізу або максимального потоку шляхом використання алгоритму Бойкова-Колмогорова.
- 2) Застосувати нейронну мережу YOLO для локалізування людини і подалюшої 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),  $\tau$  - 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,

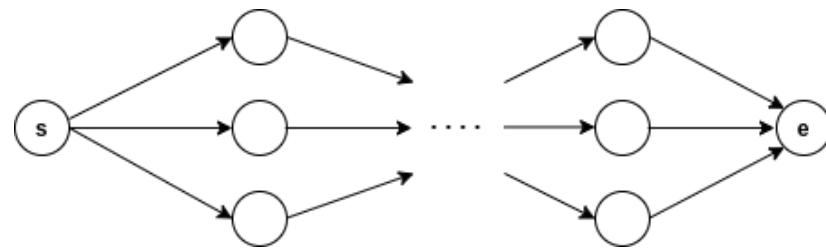


Рисунок 2.1 — Приклад графу

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  $e$  знаходячі активні вершини і додаючи їх як вершини, що відвідали. Після такого сканування вершини ста-

ють 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.

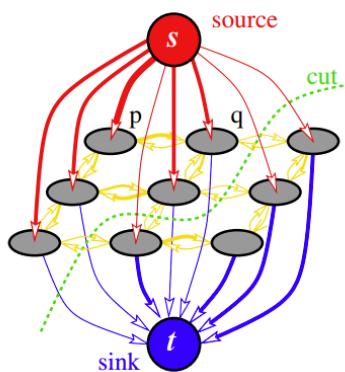


Рисунок 2.2 — Приклад графу-решітки з мінімальним розрізом

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.



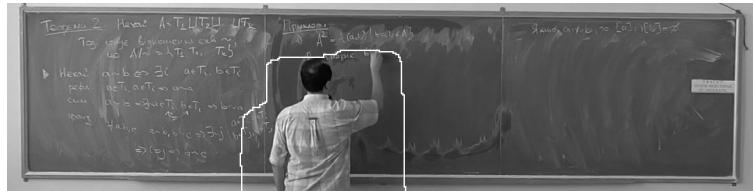
(a) Попередній кадр



(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  $\alpha$  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

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

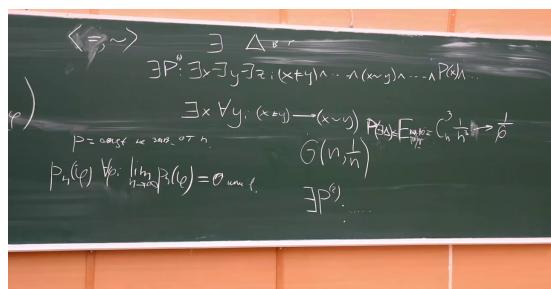
На виході отримуємо маску  $B_p^i$  рухомих об'єктів (рис. 2.3(г)). Її ми використовуємо, щоб не переносити на фінальне зображення ті пікселі, на яких було помічено рух, адже зміна яскравості пікселя виникає не тільки під час створення напису, а й під час тимчасового затуляння дошки.

### Переваги методу:

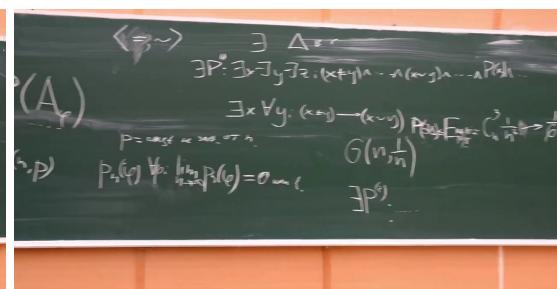
- 1) Алгоритм Б-К досить швидко будує мінімальний розріз. Статистика на ОС Ubuntu 21.04 під Intel Core i5-7200U @ 4x 3,1GHz, час 0.15 секунди на кадрі розмірами 330x640.
- 2) Даний метод не потребує ніякої передобробки, тобто не потрібно навчати як нейронну мережу.

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



(a) Попередній кадр



(б) Поточний кадр

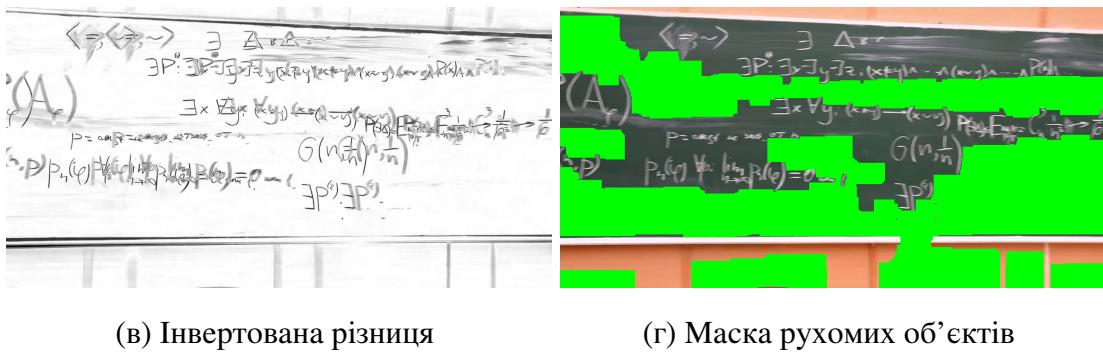


Рисунок 2.4 — Приклад отримання поганої маски під час руху камери

## 2.1 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.

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.

### 2.1.1 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 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 кроків для побудови 3D-карт, ректифікації стереопар та детекції 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  $k$ :

$$D(x,y,\sigma) = (L(x,y,k\sigma) - L(x,y,\sigma)) \quad (2.7)$$

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.

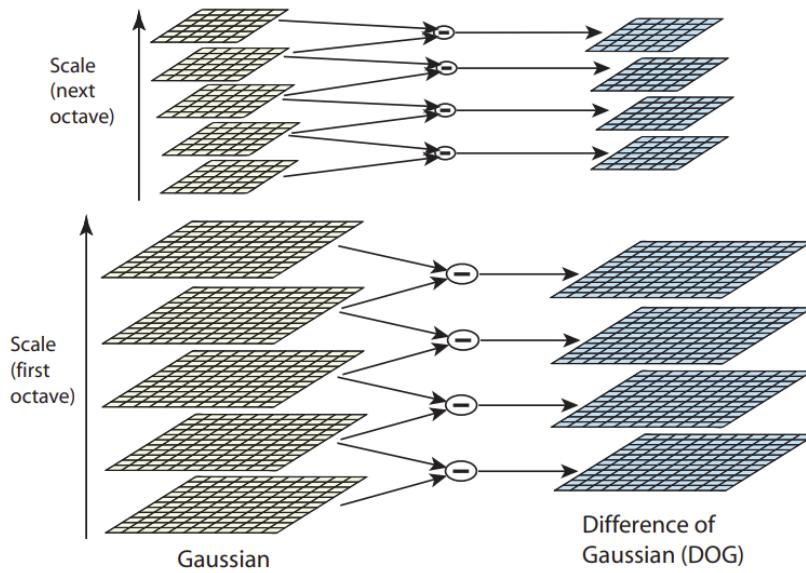


Рисунок 2.5 — Знаходження різниць гаусианів 1-го етапу SIFT

### 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.

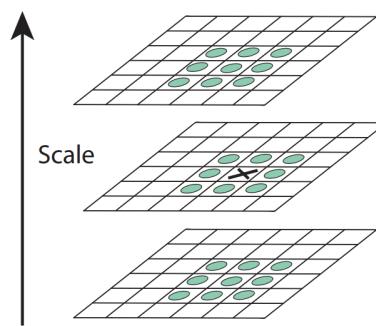


Рисунок 2.6 — Знаходження різниць гаусианів 1-го етапу SIFT

### 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.

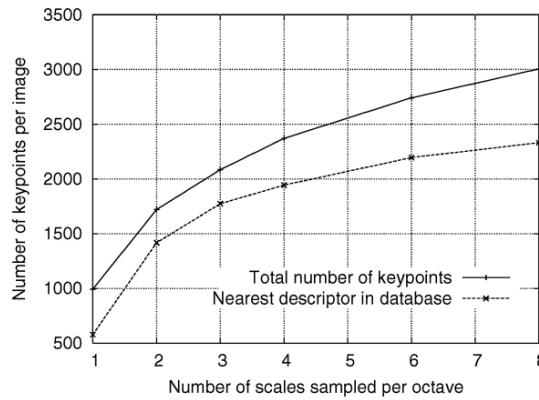


Рисунок 2.7 — Графік залежності кількості змін масштабу на кожну епоху до кількості ключових точок на зображення

### 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  $\mathbf{x} = (x,y,\sigma)$ .

$$D(\mathbf{x}) = D + \frac{\partial D^T}{\partial \mathbf{x}} \mathbf{x} + \frac{1}{2} \mathbf{x}^T \frac{\partial^2 D}{\partial \mathbf{x}^2} \mathbf{x} \quad (2.8)$$

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

$$\hat{\mathbf{x}} = \frac{\partial^2 D^{-1}}{\partial \mathbf{x}^2} \frac{\partial D}{\partial \mathbf{x}} \quad (2.9)$$

Підставивши 2.9 у 2.8 маємо можливість відсіяти нестабільні екстремуми по модулю значення (рис. 2.7):

$$D(\hat{\mathbf{x}}) = D + \frac{1}{2} \frac{\partial D^T}{\partial \mathbf{x}} \hat{\mathbf{x}} \quad (2.10)$$

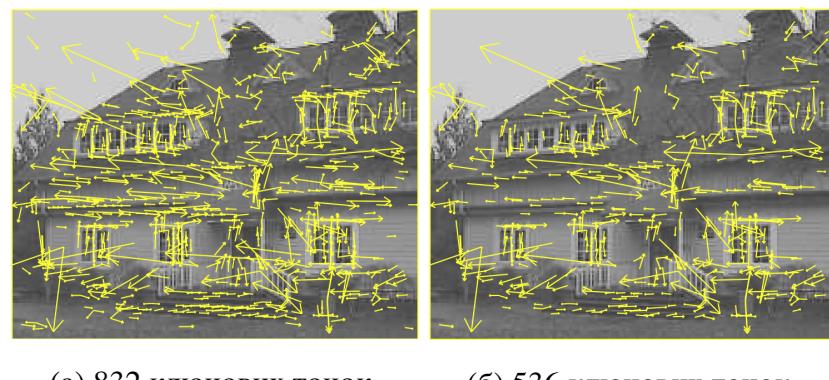


Рисунок 2.7 — Приклад відсювання екстремумів

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

### 3) Визначення орієнтації градієнтів

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

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2} \quad (2.11)$$

$$\theta(x,y) = \tan^{-1}(L(x,y+1) - L(x,y-1))/(L(x+1,y) - L(x-1,y)) \quad (2.12)$$

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

#### 4) Дескриптор точек

Локальні градієнти обчислюються для кожного масштабу навколо кожної ключової точки. Створюються дескриптори кожної ключової точки.

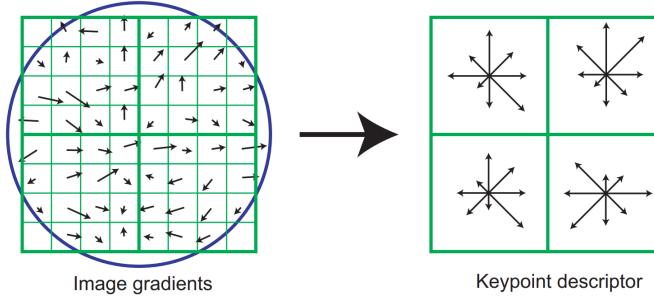


Рисунок 2.8 — Створення дескрипторів точок

## 2.1.2 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.

### 2.1.3 Знаходження гомографії за допомогою RANSAC методу

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

$$H = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{13} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \quad (2.13)$$

$$H^i \cdot p_j^{i+s} = c \cdot p_j^i, \quad \forall i = \overline{1,4} \quad (2.14)$$

Рівняння 2.14 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 2.13. 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 2.14 у матричній формі для  $T$  та  $T'$ :

$$\begin{bmatrix} x'_i \lambda \\ y'_i \lambda \\ \lambda \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{13} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}, \quad \forall i, j = \overline{1,4} \quad (2.15)$$

Такми чином ми приходимо до розв'язку системи:

$$\underbrace{\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1x'_1 - y_1x'_1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x_2x'_2 - y_2x'_2 \\ x_3 & y_3 & 1 & 0 & 0 & 0 & -x_3x'_3 - y_3x'_3 \\ x_4 & y_4 & 1 & 0 & 0 & 0 & -x_4x'_4 - y_4x'_4 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1y'_1 - y_1y'_1 \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -x_2y'_2 - y_2y'_2 \\ 0 & 0 & 0 & x_3 & y_3 & 1 & -x_3y'_3 - y_3y'_3 \\ 0 & 0 & 0 & x_4 & y_4 & 1 & -x_4y'_4 - y_4y'_4 \end{bmatrix}}_{A} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \end{bmatrix} = h_{33} \begin{bmatrix} x'_1 \\ x'_2 \\ x'_3 \\ x'_4 \\ y'_1 \\ y'_2 \\ y'_3 \\ y'_4 \end{bmatrix} \quad (2.16)$$

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  $9 \times 9$ . Параметри  $H$  знаходяться в останньому рядку  $V^*$ .

$$H = V_{9,:}^*/V_{9,9}^*, \quad H \in R^{1 \times 9} \rightarrow H \in R^{3 \times 3} \quad (2.17)$$

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.

$$E(H) = \sum_{(x,x') \in M^{i,i+s}} [\| \frac{H \cdot x'}{(H \cdot x')_z} - x \| \leq \varepsilon] \rightarrow \max_{\substack{H \in \mathbb{R}^{3 \times 3} \\ \det H \neq 0}} \quad (2.18)$$

---

**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 2.17
3. Обчислюємо  $E(H)$  по формулі 2.18
4. Якщо  $E(H) < E(H^{best})$ :

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


---