НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

імені ІГОРЯ СІКОРСЬКОГО»

Навчально-науковий Фізико-технічний інститут

Кафедра Математичного моделювання та аналізу даних

До захисту допущено:
Завідувачка кафедри
_____ **Наталія КУССУЛЬ**

«___»_____2022р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Математичні методи моделювання, розпізнавання образів та безпеки даних»

спеціальності 113 «Прикладна Математика»

на тему: «Виявлення невидимих рухів за допомогою Ейлерового збільшення»

Виконала: студентка IV курсу, групи ФІ-82 Захарченко Катерина Анатоліївна

Керівник: д.т.н., професор Куссуль Наталія Миколаївна

Консультант:

Рецензент: к.т.н., старший науковий співробітник Інституту космічних досліджень НАН України та ДКА України Богдан Яйлимов

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.

Студентка _____

Київ – 2022 року

АНОТАЦІЯ

Наш світ неможливо уявити без руху. Навколо нас живий світ, світ що постійно змінюється, та не завжди у нас на очах. Важливість розуміння цих змін важко переоцінити, адже середовище, в якому ми живемо впливає на нас тим чи іншим чином.

Поширеним методом фіксації та передачі таких змін є зображення – незалежно від того, зроблені вони звичайними камерами, мікроскопами чи супутниками, зображення та відео є безцінним джерелом інформації про зміни в часі природи нашого світу. Завдяки великому прогресу в цифровій фотографії такі зображення та відео зараз широко поширені і їх легко зняти, але обчислювальні моделі та інструменти для розуміння та аналізу змінюваних у часі процесів і тенденцій у візуальних даних ще у процесі розвитку.

У даній роботі розглядається потужний обчислювальний метод для ефективного представлення, аналізу та візуалізації як короткострокових, так і довгострокових часових змін у відео та послідовностях зображень. Зміни малої амплітуди, які важко або неможливо побачити неозброєним оком, як-от: зміна кольору шкіри людини через кровообіг, невеликі механічні рухи, можуть бути виділені для подальшого аналізу або збільшені, щоб ці зміни стали видимими для спостерігача. Цей метод також може послабити рухи та зміни, щоб усунути небажані варіації, які відволікають увагу від тих основних подій, що цікавлять.

Основний внесок цієї дипломної роботи полягає в розширенні знань про те, як обробляти просторово-часові зображення та витягувати інформацію, яку можна не відразу побачити, щоб краще зрозуміти наш динамічний світ за допомогою зображень і відео.

ABSTRACT

Our world cannot be imagined without movement. We are surrounded by a living world, a world that is constantly changing, but not always in front of our eyes. The importance of understanding these changes is difficult to overestimate, because the environment in which we live affects us in one way or another.

Images are a common method of capturing and transmitting such changes - whether they are made by conventional cameras, microscopes or satellites, images and videos are an invaluable source of information about changes in the real nature of our world. Due to the great advances in digital photography, such images and videos are now widespread and easy to capture, but computational models and tools for understanding and analyzing time-varying processes and trends in visual data are still in development.

This work considers a powerful computational method for efficient representation, analysis and visualization of both short-term and long-term time changes in video and image sequences. Low-amplitude changes that are difficult or impossible to see with the naked eye, such as changes in human skin color due to blood circulation, small mechanical movements, can be selected for further analysis or increased to make these changes visible to the observer. This method can also weaken movements and changes to eliminate unwanted variations that distract from the main events of interest.

The main contribution of this thesis is to expand the knowledge of how to process spatio-temporal images and extract information that can not be seen immediately, to better understand our dynamic world through images and videos.

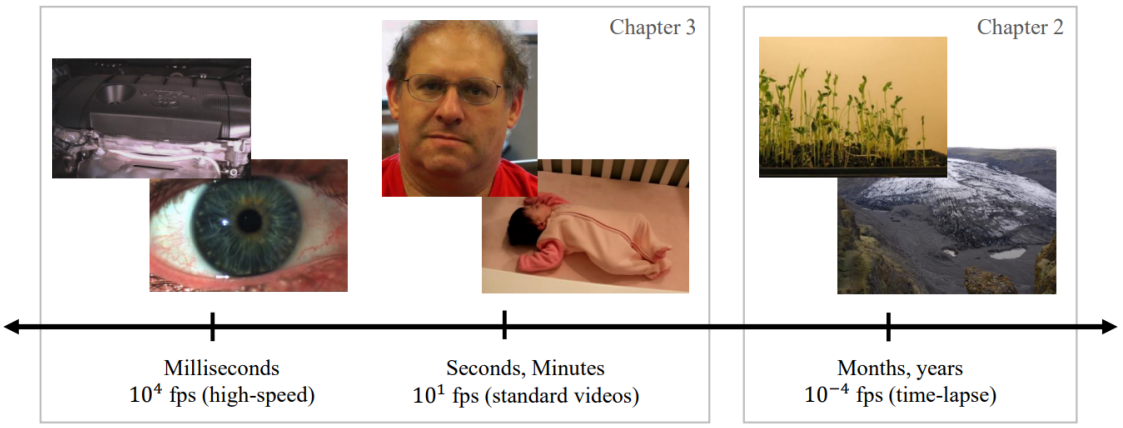
Зміст

ВСТУП

Сучасна фотографія надає нам корисні інструменти для зйомки фізичних явищ, що відбуваються в різних масштабах часу (Малюнок 1). На одному кінці спектру високошвидкісна зйомка, що в наші часи відбувається з частотою кадрів 6 МГц (6 мільйонів кадрів в секунду), що дозволяє високоякісно фіксувати надшвидкі події, такі як ударні хвилі та нейронну активність [20]. На іншому кінці спектра, уповільнені послідовності, на яких може бути виявлено довгострокові процеси, що охоплюють десятиліття, такі як еволюція міст, танення льодовиків і вирубка лісів, і навіть процеси у планетарному масштабі, що нещодавно стали доступні [12].

Наразі методи автоматичного визначення та аналізу фізичних процесів або тенденцій у візуальних даних все ще розвиваються. Проведено дослідження аби полегшити аналіз явищ, які фіксуються зображеннями, і виявити цікаві закономірності та процеси, які не можуть бути легко помітні в вихідних даних із плином часу. Тут подано загальний огляд цих проектів.

Детальна інформація про запропоновані методики та огляд відповідної літератури наведено в наступних розділах.



Малюнок 1: Часові шкали в зображеннях. Високошвидкісне відео (ліворуч) фіксує короткочасні швидкі рухи, такі як вібрація двигунів і невеликі рухи очей. Відео із нормальною частотою (середня частина) може фіксувати фізіологічні функції, такі як частота серцевих скорочень та дихальні рухи. Уповільнені послідовності (праворуч) зображують довготривалі фізичні процеси, такі як ріст рослин і танення льодовиків. Перелічені частоти кадрів (кадрів за секунду; кадри в секунду) є лише репрезентаціями і можуть значно відрізнятися від відео в кожній категорії.

Видалення відволікаючих варіацій

Існує проблема рухів та змін, які відволікають від основних часових сигналів, що можуть цікавити дослідника і, як наслідок, задача їх позбутися. Це може бути особливо корисним при роботі з уповільненими послідовностями, які часто використовуються для більш тривалого медичного та наукового аналізу, де динамічні сцени записуються протягом порівняно великого періоду часу. Коли події, що відбуваються протягом дня або навіть року, зведені в хвилини або секунди, пікселі можуть бути непостійними через значне зміщення часу. Такі ефекти виникають через раптову поява та зникнення об’єктів, або ж через швидкі зміни освітлення

між кадрами, що розташовані послідовно. Це ускладнює аналіз довготривалих процесів.

В якості прикладу застосування можна навести дослідження росту вуличних рослин. Виміри їх росту відбуваються шляхом аналізу відео, на якому може бути попередньо приглушено небажані рухи листя, яке коливається на вітрі.

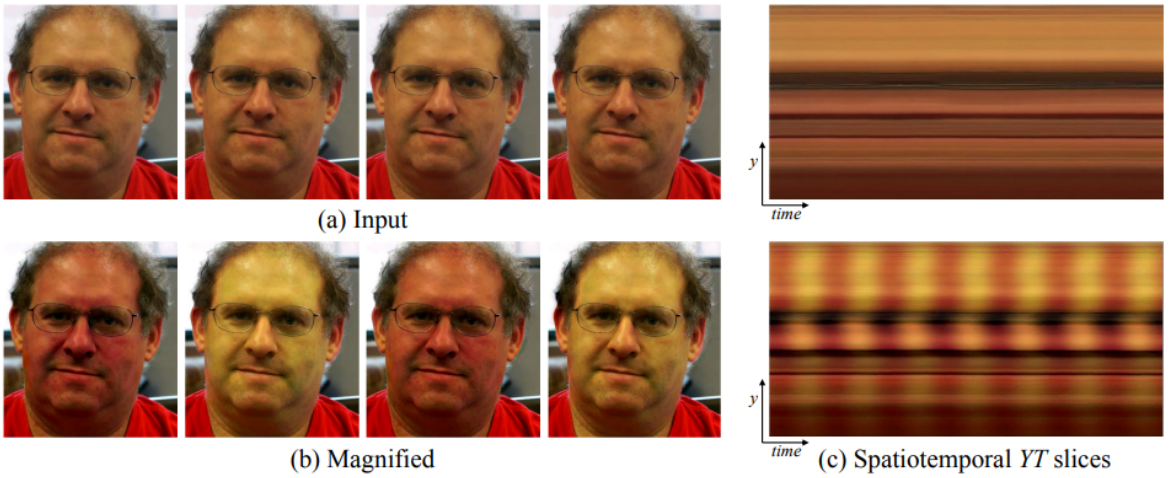
Система обробки відео, про яку йде мова, може розглядати короткострокові візуальні зміни як шум, довгострокові зміни як сигнал, і повторно відтворює відео, щоб розкрити основні довгострокові події [44]. Результатом є автоматичне розкладання оригінального відео на коротко- та довготривалі компоненти руху. Далі буде показано, що наївні підходи до часової фільтрації часто не в змозі досягти цього завдання, і представлено обчислювальний підхід до шуму в русі без явного аналізу руху — метод Ейлерового збільшення. Це робить його застосовним до різноманітних відео, що містять активну динаміку, загальну для довготривалих зображень.

Збільшення непомітних варіацій

В інших випадках можна збільшити зрушення, які занадто малі, щоб їх можна було побачити неозброєним оком. Наприклад, колір шкіри людини незначно змінюється в залежності від кровообігу (рисунок 3.1). Ця варіація, хоча й невидима неозброєним оком, може бути використана для визначення частоти пульсу та виявлення просторових моделей кровотоку. Аналогічно, рух з низькою просторовою амплітудою, невидимого для людини, можна збільшити, щоб виявити цікаву механічні закономірності.

У методі запропоновано поєднання просторової та часової обробки, щоб підкреслити крихітні зміни у відео [59, 45, 55]. Ці методи використовують ейлерівський підхід до змін у сцені, аналізуючи та посилюючи зміни в часі у фіксованих місцях у просторі (пікселі). Перший можливий метод, який можна назвати лінійним, бере стандартну відеопослідовність як вхідні дані та застосовує просторове розкладання з подальшою часовою фільтрацією до кадрів. Отриманий часовий сигнал потім посилюється, щоб виявити приховану інформацію.

Цей підхід до часової фільтрації також може виявити просторовий рух з низькою амплітудою. Далі в роботі буде надано математичний аналіз, який пояснює, як сигнал часової інтенсивності взаємодіє з просторовим рухом у відео. Метод покладається на лінійну апроксимацію, пов'язану з припущенням що яскравість є сталою. Це наближення (а, отже, і метод) застосовується лише до дуже малих рухів, але це саме той тип рухів, який ми хочемо посилити.



Малюнок 2: Приклад використання системи ейлерівського відеозбільшення для візуалізації пульсу людини.

(a) Чотири кадри з оригінальної відеоряду (обличчя).

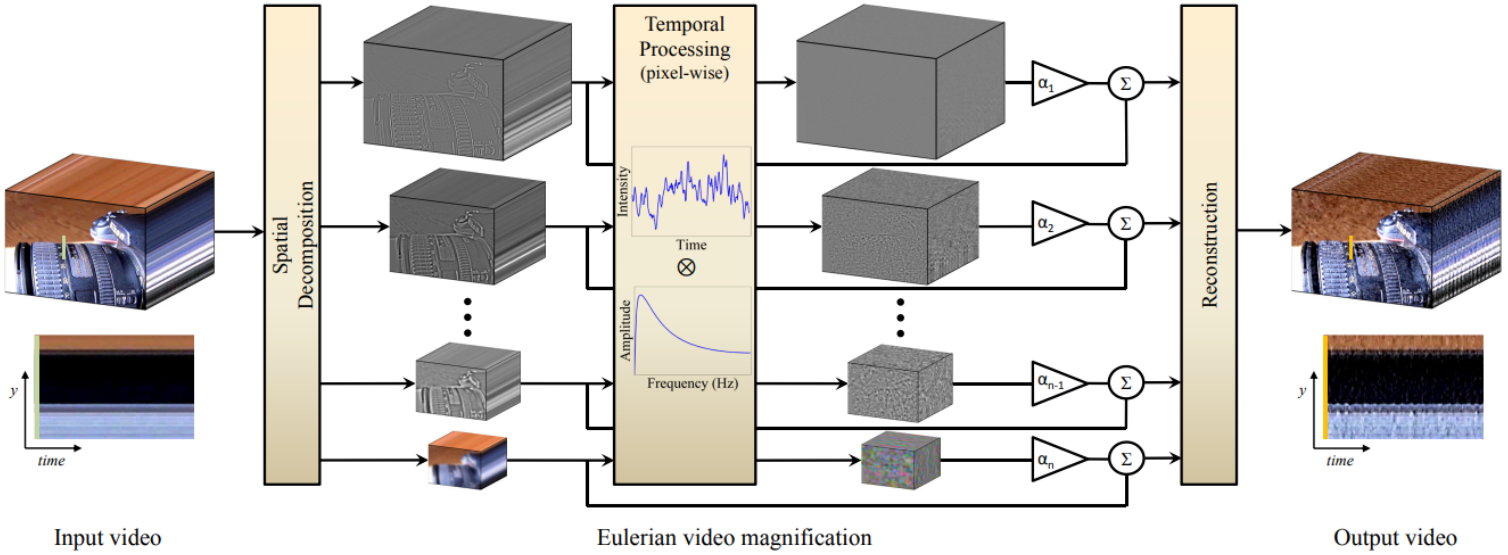
(b) Ті самі чотири кадри з посиленим імпульсним сигналом суб'єкта.

(c) Вертикальна лінія розгортки вхідного (зверху) та вихідного (знизу) відео, розгорнутого по часу, показує, як метод посилює періодичні зміни кольору. У вхідній послідовності сигнал непомітний, але в збільшеній послідовності зміна чітка.

Лінійний метод посилення рухів простий і швидкий, але має два основних недоліки. А саме, шум посилюється лінійно з посиленням, а наближення до посиленого руху швидко руйнується для високих просторових частот і великих рухів. Щоб протистояти цим проблемам, ми запропонували кращий підхід до обробки малих рухів у відео, де ми замінюємо лінійну апроксимацію локалізованим розкладанням Фур'є з використанням пірамід зі складними значеннями [55] (Розділ 3.3). Фазові зміни коефіцієнтів цих пірамід у часі відповідають руху і можуть бути тимчасово оброблені та модифіковані для маніпулювання рухом. У порівнянні з лінійним методом, цей метод на основі фаз має більші обчислювальні витрати, але може підтримувати більше посилення руху з меншою кількістю артефактів і меншим шумом. Це ще більше розширює режим низькоамплітудних фізичних явищ, які можна проаналізувати та візуалізувати за допомогою ейлерівських підходів.

Ця робота значною мірою базується на роботі, яка з'явилася на конференції IEEE з комп'ютерного зору та розпізнавання образів (CVPR) 2011 року [44], а також на статтях наукового журналу ACM Transactions on Graphics (Proceedings SIGGRAPH 2012 і 2013) [59, 55].

1 ПРОСТОРОВО-ЧАСОВА ОБРОБКА ВІДЕОФАЙЛІВ



Малюнок 1.1: Огляд ейлерової системи збільшення відео. Спочатку система розкладає вхідну відеопослідовність на різні просторові діапазони частот, і застосовує той самий часовий фільтр до всіх діапазонів. Відфільтровані просторові смуги потім посилюються на заданий коефіцієнт α, додаються до вихідного сигналу та згортаються для створення вихідного відео. Вибір часового фільтра та коефіцієнтів посилення можна налаштувати в залежності від застосування. Наприклад, ми використовуємо систему, щоб виявити невидимі рухи цифрової камери, викликані фізичною специфікою її роботи (перегортанням дзеркала під час серій фотографій).

Підхід, застосований в роботі полягає в обробленні відео по простору та часу, задля підкреслення незначних часових змін. Процес обробки проілюстровано на малюнку 2. На початку ми розкладаємо кадрову послідовність на різні просторово-частотні діапазони. Але ці діапазони можуть бути в подальшому збільшені по-різному. Таке відбувається з двох причин:

- вони можуть мати різне співвідношення сигнал/шум
- вони можуть містити просторові частоти, для яких не виконується лінійне наближення, використане в нашому збільшенні руху, що буде розглянуто далі.

В останньому випадку ми зменшуємо посилення для цих смуг для того щоб приглушити артефакти. Оскільки метою просторової обробки є просто збільшити часове відношення сигнал/шум шляхом об'єднання кількох пікселів, проводиться фільтрація кадрів відео з низькими частотами та зменшується їх частота для підвищення ефективності обчислень. Однак у загальному випадку ми обчислюємо повну піраміду Лапласа. [4]

Наступним етапом ми виконуємо часову обробку для кожної просторової смуги. Розглядаються часові ряди, що відповідають значенню пікселя в смузі частот, і застосовується смуговий фільтр для виділення діапазонів частот, які ми оберемо. Наприклад, якщо ми хочемо збільшити ті рухи, які можуть нести в собі інформацію про пульс людини, нас можуть цікавити частоти в межах 0,4-4 Гц, що відповідають 24-240 ударам на хвилину. Якщо ми можемо виділити частоту пульсу, ми можемо використовувати вузьку смугу навколо цього значення.

Часова обробка є однорідною для всіх просторових рівнів і для всіх пікселів на кожному рівні. Помножимо отриманий сигнал пропущеної смуги на коефіцієнт збільшення α. Цей коефіцієнт може бути визначений користувачем і може бути ослаблений автоматично, про що буде описано в роботі детальніше, як і про можливі часові фільтри. Далі ми додаємо збільшений сигнал до оригіналу та згортаємо просторову піраміду, щоб отримати остаточний результат реконструкції просторової декомпозиції.

Метод має неявно давати узгоджені результати по простору та часу. Це досягається через те що

•подані на вхід відео є плавними в просторі та в часі

•наша фільтрація виконується рівномірно по пікселях.

Висновки:

Отже алгоритм обробки має бути реалізовано із врахуванням вимог, що наведені вище. Він має відповідати схемі на мал.1.1 та коректно здійснювати просторово-часову обробку зображень. Відповідно до того, зрушення якої амплітуди дослідник бажає підсилити, потрібно буде підібрати фільтр відповідної ширини. Оскільки варіювання параметру сили збільшення рухів також можливе, ми матимемо змогу досліджувати вхідний матеріал відповідно до наших побажань та потреб.

2 ЕЙЛЕРОВЕ ЗБІЛЬШЕННЯ РУХІВ

Ейлерова обробка може посилити невеликий рух, навіть якщо ми не відстежуємо рух, як у методах Лагранжа [5]. У цьому розділі показується, як часова обробка посилює рух за допомогою аналізу, який спирається на розкладання ряду Тейлора першого порядку, поширене в аналізі оптичного потоку [6].

2.1 Одновимірний рух

Щоб пояснити зв'язок між часовою обробкою та збільшенням руху, ми розглянемо простий випадок одновимірного сигналу, який зазнає поступального руху. Цей аналіз узагальнюється безпосередньо на локально-поступальний рух у 2D.

Нехай *I(x, t)* позначає інтенсивність зображення у точці x і час t. Зображення зазнає поступального руху, і ми можемо виразити спостережувані інтенсивності за допомогою функції зміщення δ(t), так, що

$$I(x, t) = f(x + \delta(t)) \; I(x, 0) = f(x)$$

Метою збільшення руху є синтез наступного сигналу

$$I(x, t) = f(x + (1 + \alpha)\delta(t)) \qquad (1)$$

для деякого коефіцієнта посилення α.

Припускаючи, що зображення можна апроксимувати розкладанням Тейлора першого порядку, ми запишемо наше зображення в момент часу *t, f(x + δ(t))* у ряд Тейлора першого порядку:

$$I(x,\ t)\ \approx\ f(x)\ +\ \delta(t)\frac{\partial f(x)}{\partial x} \qquad (2)$$

Нехай B(x, t) є результатом застосування широкосмугового часового фільтра до I(x, t) у кожній позиції x. Наразі припустимо, що сигнал руху δ(t) знаходиться в межах смуги пропускання тимчасового смугового фільтра. Тоді маємо:

$$B(x,\ t)\ =\ \delta(t)\frac{\partial f(x)}{\partial x} \quad (3)$$

Далі в процесі роботи ми підсилюємо цей смуговий сигнал на α і додаємо його назад до I(x, t), в результаті чого оброблений сигнал запишемо як:

$$\tilde{I}(x,\ t)\ =\ I(x,\ t)\ +\ \alpha\, B(x,\ t) \qquad (4)$$

Поєднуючи наведені вище міркування, збільшений сигнал запишеться у вигляді:

$$\tilde{I}(x,\ t)\ \approx\ f(x)\ +\ (1\ +\ \alpha\,)\ \delta(t)\frac{\partial f(x)}{\partial x} \qquad (5)$$

Припускаючи, що розклад Тейлора першого порядку виражено для посиленого збурення *(1 + α)δ(t)*, ми можемо пов’язати збільшення часового сигналу зі збільшенням просторового сигналу.

$$\tilde{I}(x,\ t)\ \approx\ \mathrm{f(x + (1 + \alpha)\delta(t))} \quad (6)$$

Вище наведена репрезентація збільшеного руху — просторове зміщення δ(t) локального зображення f(x) у момент часу t було посилено до величини (1 + α). Цей процес проілюстровано для однієї синусоїди на малюнку 3. Для низькочастотної хвилі косинуса та відносно невеликого зміщення δ(t) розкладання в ряд Тейлора першого порядку служить хорошим наближенням для перекладеного сигналу в момент часу t + 1.

Підвищуючи часовий сигнал на α і додаючи його назад до I(x, t), ми апроксимуємо цю хвилю, перетворену на (1 + α)δ(x).

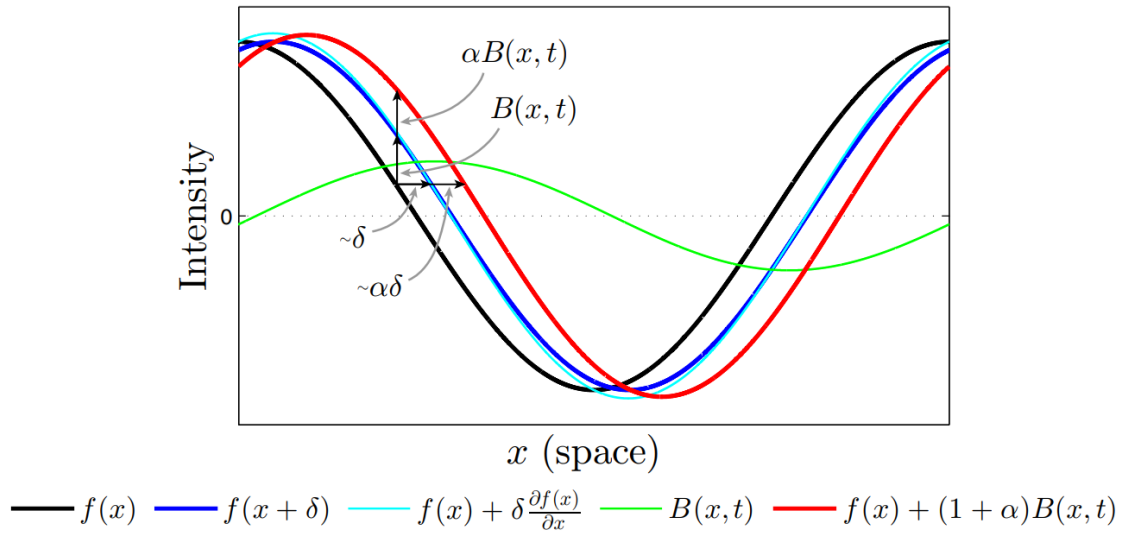
Для повноти повернемося до більш загального випадку, коли δ(t) не повністю знаходиться в межах смуги пропускання тимчасового фільтра. У цьому випадку нехай $\delta_k(t)$, представляє різні часові компоненти δ(t). Кожне $\delta_k(t)$ буде ослаблено часовою фільтрацією на коефіцієнт γ_k. Це призводить до смугового сигналу

$$B(x,t) = \sum_k \gamma_k \delta_k(t) \frac{\partial f(x)}{\partial x} \qquad (7)$$

Аналогічно рівнянню 3. Через множення в рівнянні 4, це часове частотно-залежне послаблення можна еквівалентно інтерпретувати як коефіцієнт збільшення руху, $\alpha_k = \gamma_k \alpha$, результатом чого є вихідний сигнал із посиленим рухом,

$$\tilde{I}(x,\ t)\ \approx\ f(x\ +\ \sum_k (1\ +\ \alpha_k)\delta_k(t)) \qquad (8)$$

Результат є очікуваним для лінійного аналізу: модуляція спектральних компонентів сигналу руху стає коефіцієнтом модуляції в коефіцієнті посилення руху α_k, для кожного часового піддіапазону δ_k , сигналу руху.



Малюнок 2.1: Часова фільтрація може бути усередненням просторового перетворення. Цей ефект демонструється на одновимірному сигналі, але так само застосовний і до сигналів розмірності 2. Вхідний сигнал відображається в два моменти часу:
I(x, t) = f(x) в момент *t* і *I(x, t + 1) = f(x + δ)* в момент часу t + 1. Одновимірний розклад в ряд Тейлора I(x, t + 1) відносно x добре апроксимує перетворений сигнал. Часовий сигнал посилюється і додається до вихідного сигналу. У цьому прикладі α = 1, що збільшує рух на 100%, а часовий фільтр є різницею між двома кривими.

2.2 Границі

На практиці припущення в розділі 2.1 справедливі для плавних зображень і невеликих рухів. Для швидкозмінюваних функцій зображення (тобто високих просторових частот), f(x), наближення ряду Тейлора першого порядку стають неточними для великих значень збурення,
1 + αδ(t), яке збільшується як при збільшенні фактору α, так і при русі δ(t). На малюнках 4 і 5 показано вплив частот, більших коефіцієнтів підсилення та більших рухів на посилений рух сигналу синусоїди.

Як функцію просторової частоти ω ми можемо отримати орієнтир щодо того, наскільки великим може бути коефіцієнт посилення руху α, з огляду на спостережуваний рух δ(t). Щоб оброблений сигнал Ĩ(x, t) був приблизно рівним справжньому збільшеному руху Î(x, t), ми шукаємо умови, за яких

$$\tilde{I}(x,\ t)\ \approx\ \hat{I}(x,\ t)$$

$$\Rightarrow f(x)\ +\ (1\ +\ \alpha)\delta(t)\frac{\partial f(x)}{\partial x}\ \approx\ f(x\ +\ (1\ +\ \alpha)\delta(t)) \qquad (9)$$

Нехай $f(x)\ =\ cos(\omega x)$ для просторової частоти ω і позначимо $\beta\ =\ 1\ +\ \alpha$.

Ми вимагаємо, щоб

$$cos(\omega x)\ -\ \beta\omega\delta(t)\ sin(\omega x) \approx cos(\omega x\ +\ \beta\omega\delta(t)) \qquad (10)$$

Використовуючи закон додавання для косинусів, маємо

$$cos(\omega x)\ -\ \beta\omega\delta(t)\ sin(\omega x)\ =\ cos(\omega x)\ cos(\beta\omega\delta(t))\ -\ sin(\omega x)\ sin(\beta\omega\delta(t))$$

(11)

Отже, наступне має приблизно дотримуватися

$$cos(\beta\omega\delta(t)) \approx 1 \qquad (12)$$

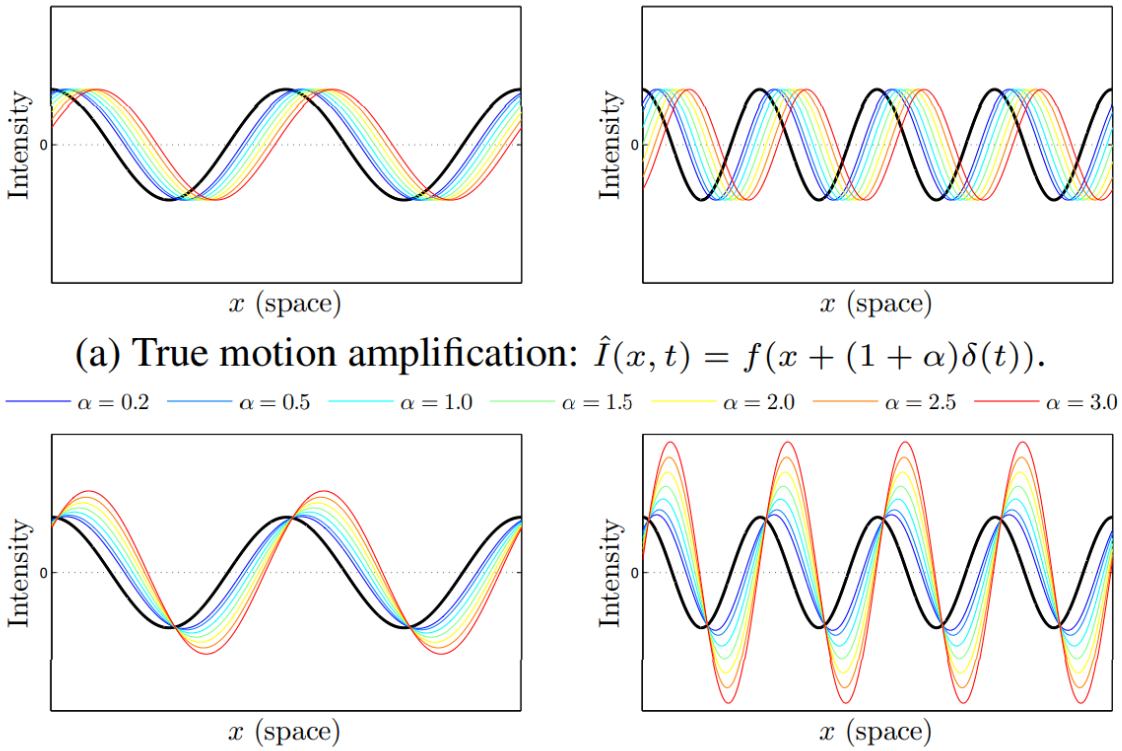
$$sin(\beta\omega\delta\ (t)) \approx \beta\delta(t)\omega \qquad (13)$$

Ці малі наближення (рівняння (12) і (13)) будуть виконуватися з точністю до 10% для $\beta\omega\delta(t) \leq \pi/4$ (синусоподібний член є провідним наближенням і маємо $sin(\ \pi/4\ )\ =\ 0,9\frac{\pi}{4}$).

В термінах просторової довжини хвилі, λ = 2π/ω, рухомого сигналу це дає

$$(1 + \alpha)\delta(t) < \lambda/8 \qquad (14)$$

Відношення вище (14) надає важливу інформацію про те наскільки великим може бути коефіцієнт посилення руху, α, пов'язуючи в собі точне збільшення даного руху δ(t) і просторову довжину хвилі, λ.



(a) True motion amplification: $\hat{I}(x,t) = f(x + (1+\alpha)\delta(t))$.

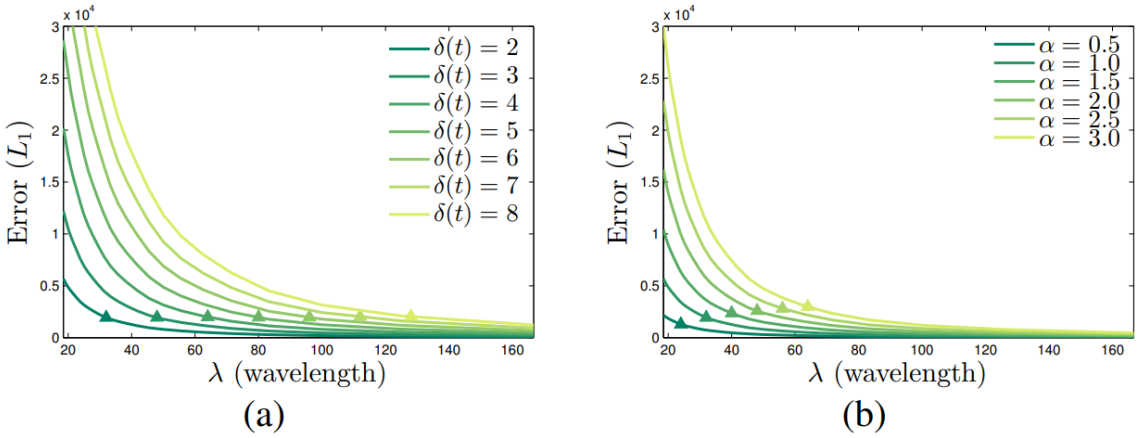
(b) Motion amplification via temporal filtering: $\tilde{I}(x,t) = I(x,t) + \alpha B(x,t)$.

Малюнок 2.2: Ілюстрація посилення руху на одновимірному сигналі для різних просторових частот і значень α. Для зображень з лівого боку λ = 2π і δ(1) = π/8. Для зображень з правого боку λ = π і δ(1) = π/8 .

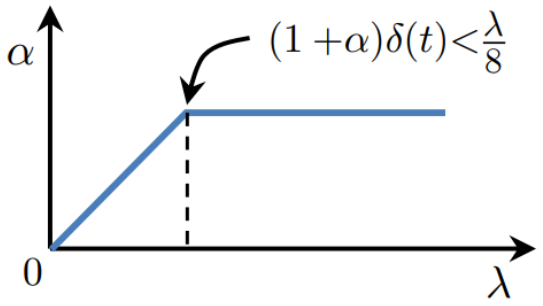
(a) Справжнє зміщення I(x, 0) на (1 + α)δ(t) у момент t = 1, забарвлене від синього (що відповідає за невеликий коефіцієнт посилення) до червоного (високий коефіцієнт посилення).

(b) Посилене зміщення, створене нашим фільтром, з кольорами, що відповідають правильно зміщеним сигналам у (a). Червоні (крайні праві) криві кожного графіка відповідають (1 + α)δ(t) = λ/4 для лівого графіка і (1+α)δ(t) = λ/2 для правого графіка, що показує легкі, а потім серйозні артефакти, що вносяться в збільшення руху від перевищення межі (1 + α) у 2 і 4 рази відповідно.

На малюнку 2.2 (b) показані похибки збільшення руху для синусоїди, коли ми збільшуємо α, переходячи за межі, що відповідають рівнянню 14.



Малюнок 2.3: Похибка збільшення руху, обчислена як L1 норма між істинним посиленим сигналом руху (Малюнок 2.2(a)) і результатом, відфільтрованим за часом (Малюнок 2.2(b)), як функція довжини хвилі, для різних значень δ(t) (a) і α (b). У (a) фіксуємо α = 1, а в (b) δ(t) = 2.



Малюнок 2.6: Коефіцієнт посилення, α, як функція просторової довжини хвилі λ, для посилення руху. Коефіцієнт посилення фіксується на α для просторових смуг, які знаходяться в межах нашої похідної межі, і послаблюється лінійно для більш високих просторових частот.

2.3 Багатомасштабний аналіз

Підхід у розд. 2.2 пропонує процес обробки, що можна змінювати у масштабі: використання особливого коефіцієнта збільшення α для певної бажаної смуги просторових частот, а потім повернення до високих просторових частот (такого що випливає з рівняння 14 або визначеного користувачем), там, де посилення може призвести до небажаних артефактів. На малюнку 6 показана така схема модуляції для α. Попри те що області з високими просторовими частотами зазвичай будуть посилюватись менше за нижчі частоти, ми виявили, що отримані відео містять досить прийнятний для сприйняття людиною посилений рух.

Висновки:

Було отримано важливе обмеження щодо того, наскільки великим може бути коефіцієнт посилення руху α. Рівняння 14, що виражає це обмеження, часто буде потрібне для дослідження зокрема того, які

результати надасть алгоритм при перебільшенні цих вимог. Застосуємо ці обмеження для одновимірного та двовимірного випадків.

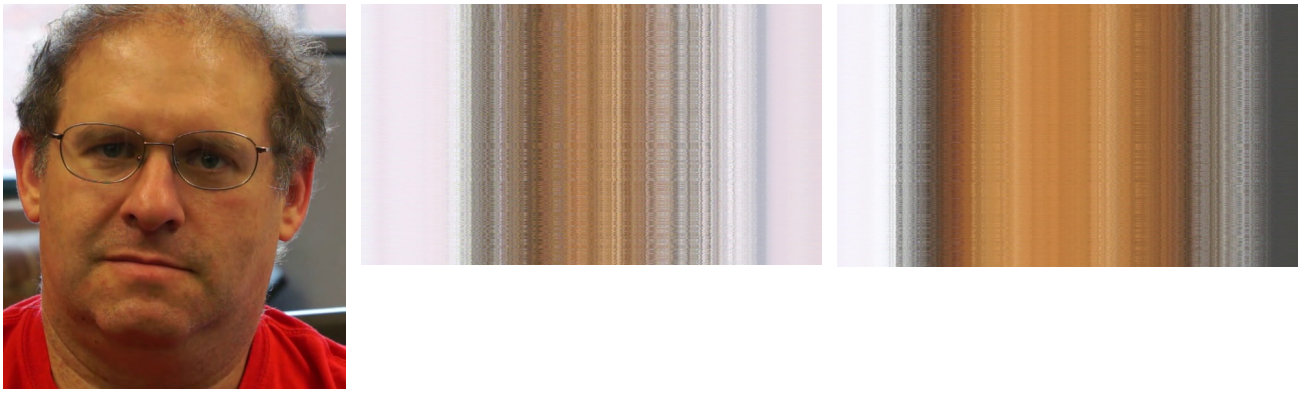
3 ПРАКТИЧНА РЕАЛІЗАЦІЯ

3.1 Наївна обробка відео

Перед тим, як програмувати безпосередньо алгоритм вирішення поставленої задачі, що розглядається, було вирішено спробувати досягти тієї ж мети програмними методами Python. Метою проведення наступних маніпуляцій є визначити, чи можливо збільшити якісь зміни у зображеннях простіше, без застосування алгоритму Ейлерового збільшення чи будь-якого іншого математичного підходу та проаналізувати, наскільки необхідний алгоритм що розглядається.

10 секундне відео було розподілено на послідовність з 301 кадру з якої було сформовано масив. Зображення були розглянуті як numpy масиви розмірності 592 x 528 x 3. З кожного кадру відео було обрано один і той самий рядок пікселів розміром 1 x 528 і скомпоновано зображення, в якому кожен наступний знизу рядок пікселів відповідає за кожен наступний кадр. Тобто розмір такого зображення залежить від кількості кадрів (висота) та ширини зображення (ширина).

Малюнок 3.1:



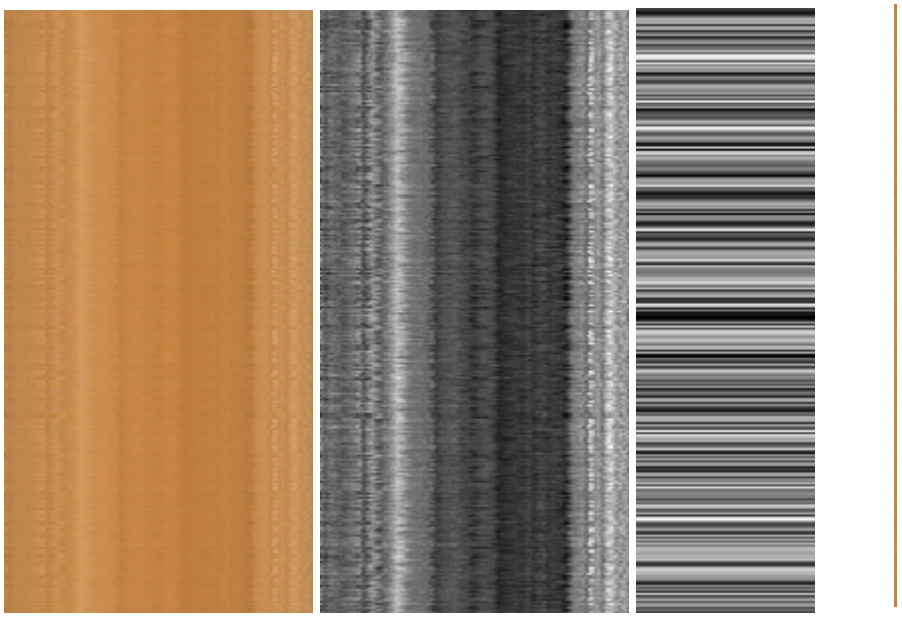
(a) (b) (c)

(a) кадр №1 з відео face.mp4

(b) перший рядок пікселів кожного кадру, де перший рядок відповідає першому рядку першого кадру (a), а останній рядок відповідає першому рядку останнього кадру

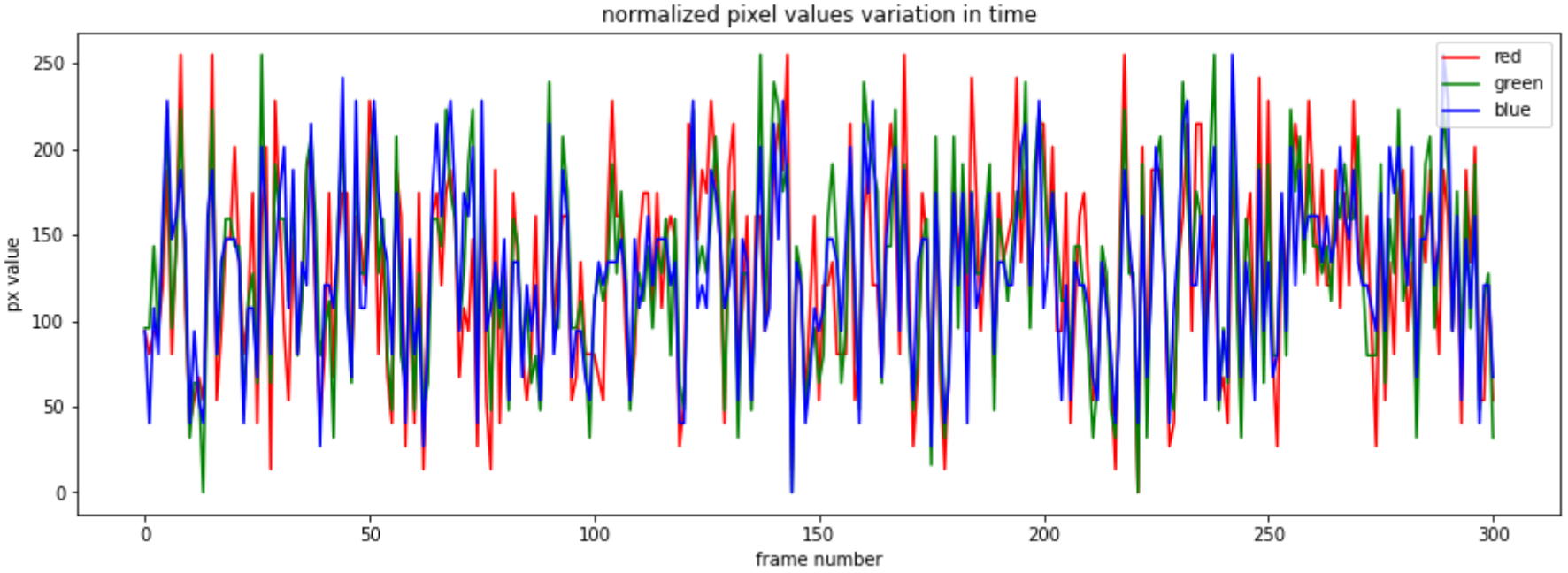
(c) аналогічно до (b), але рядок на кадрах взято 54-ий

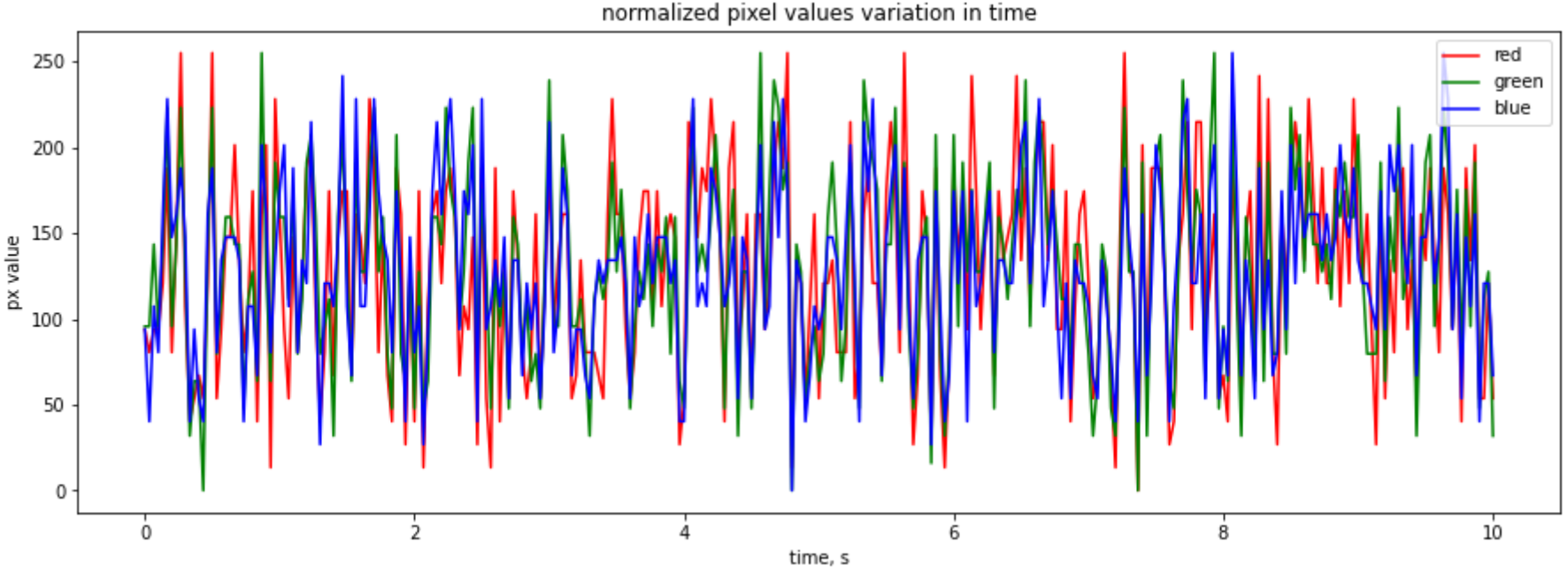
З метою кращої нормалізації по кольору, розгорнута по часу послідовність у вигляді зображення (мал. 3.1 (c)), була обрізана з боків (мал. 3.1 (d)), конвертована у чорно-білий формат та нормалізована (мал. 3.1 (e)). Далі було взято один стовпчик пікселів у цьому зображенні та розтягнуто його в ширину (f). Візульно коливання помічені не були.



(d) (e) (f) (g)

Повернувшись до зображення (d) та обрізавши його до одного стовпчика пікселів, було проведено спробу виявити коливання на графіку значень пікселів та за допомогою перетворення Фур’є виявити частоти. Спробу також вважаю невдалою.

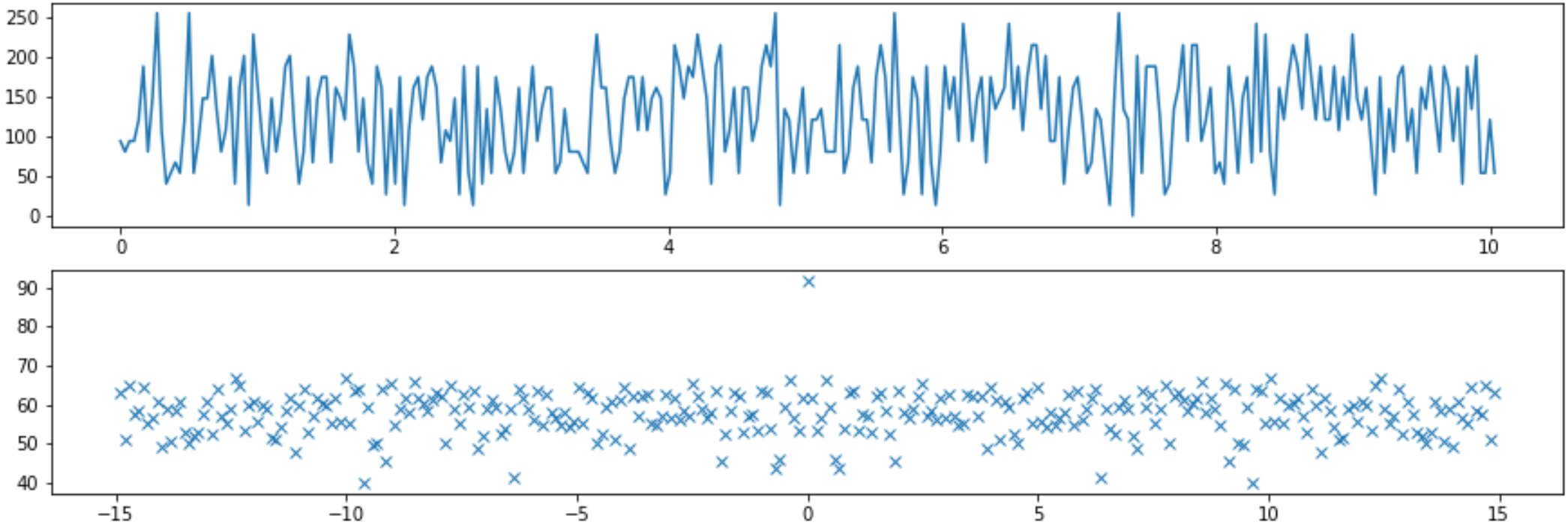




Малюнок 3.2: нормалізовані значення пікселів (мал. 3.1 (g)) по кожному каналу, розгорнуті по
(a) номеру кадру
(b) часу тривалості відео.

(a)

(b)



Малюнок 3.3: Частоти вибірки дискретного перетворення Фур'є. Повернутий масив містить центри частотних бітів у циклах на одиницю інтервалу вибірки (з нулем на початку).

3.2 Гаусові піраміди

Існують різні способи представлення зображення, які мають свої переваги та недоліки. Формат, який використовується для представлення даних зображення, може бути таким же важливим при обробці зображень, як і алгоритми, застосовані до даних. Необроблений формат представлення зображень, в якому вони кодуються як масив інтенсивності пікселів не підходить для більшості завдань.

Як альтернатива, зображення може бути визначене його перетворенням Фур'є, з операціями, що застосовуються до коефіцієнтів перетворення, а не до вихідних значень пікселя. Це підходить для деяких завдань зі стиснення даних та покращення зображення, але не підходить для інших. Таке представлення особливо не підходить для машинного зору та комп'ютерної графіки, де просторове розташування елементів шаблону є критичним. [7]

В даному проекті маємо великий інтерес до уявлень, за яких враховуються *просторова локалізація*, а також локалізація в просторово-частотній області. Це досягається шляхом розкладання зображення на набір зображень компонентів просторової смуги частот. Окремі зразки компонент цих зображень представляють інформацію про рельєф зображення, яка відповідним чином локалізована.

Існують докази того, що зорова система людини використовує таке уявлення [8] а схеми з великою роздільною здатністю стають все більш популярними в машинному баченні та в обробці зображень загалом.

Важливість аналізу зображень у декількох масштабах окремо випливає з природи самих зображень. Світ і його відображення містять

об’єкти багатьох розмірів, і ці об’єкти містять об’єкти багатьох розмірів. Крім того, об’єкти можуть знаходитися на різній відстані від глядача. В результаті будь-яка процедура аналізу, яка застосовується лише в одному масштабі, може пропустити інформацію в інших масштабах. Відповідь полягає в тому, щоб проводити аналізи на всіх масштабах одночасно.

Саме тому існують різноманітні методи так званих “пірамід”, які використано для аналізу, стиснення та покращення зображень.

Існують дві причини для перетворення зображення з одного представлення в інше: перетворення може виділити критичні компоненти рельєфу зображення, щоб зробити їх більш доступними для аналізу; може помістити дані в більш компактну форму, щоб з ними можна було працювати швидше і легше, адже таким чином вони зберігаються та передаються ефективніше. Піраміда Лапласа виконує обидві ці цілі.

Будова цих пірамід має тенденцію покращувати особливості зображення, такі як краї, які важливі для інтерпретації. Ці ознаки розділені за масштабом на різних рівнях піраміди.

Подання у вигляді пірамід також дозволяє стискати дані. Не зважаючи на те що воно матиме на третину більше елементів вибірки, ніж вихідне зображення, значення цих вибірок, як правило, близькі до нуля, і тому можуть бути представлені невеликою кількістю бітів.

Подальше стиснення даних можна отримати за допомогою квантування: кількість різних значень, взятих вибірками, зменшується за рахунок суміювання існуючих значень. Це призводить до деякої деградації під час реконструкції зображення, але якщо квантування зроблене

правильно, люди-спостерігачі це погіршення не виявлять і воно загалом не має вплинути на продуктивність алгоритмів аналізу. [8]

3.2 Аналіз результатів

Щоб обробити вхідне відео за допомогою ейлерівського відеозбільшення, користувачу необхідно виконати чотири кроки:

(1) вибрати часовий смуговий фільтр;

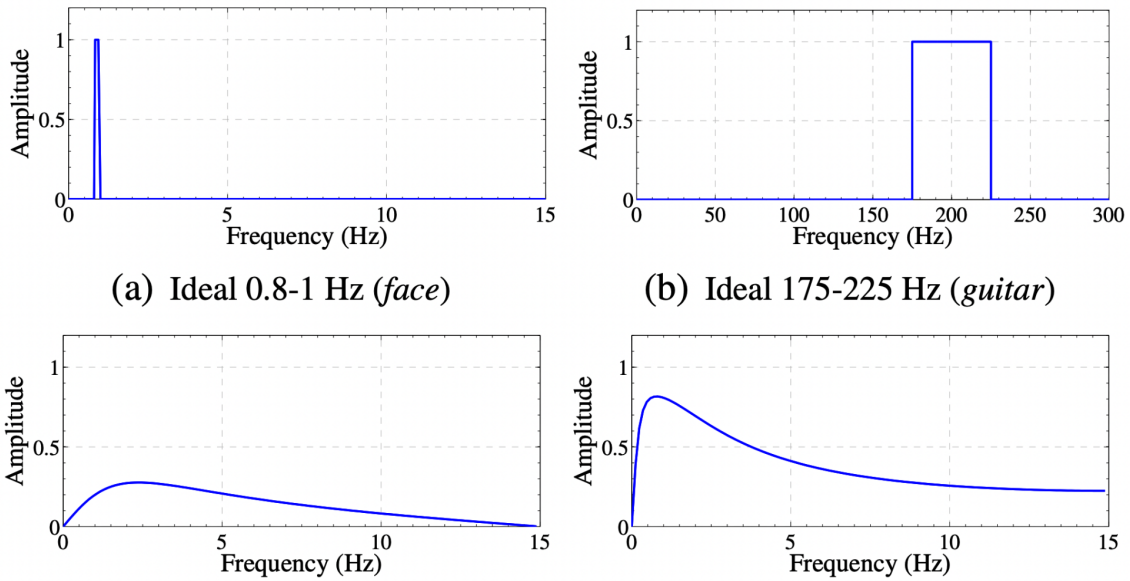
(2) вибрати коефіцієнт посилення α;

(3) вибрати межу просторової частоти (зазначену просторовою довжиною хвилі λ_c), за межами якої використовується ослаблена версія α;

(4) вибрати форму загасання для α—або направити α до нуля для всіх $\lambda < \lambda_c$. У деяких випадках діапазон частот, що цікавить, може бути обраний автоматично, але часто важливо, щоб користувачі мали можливість керувати діапазоном частот, що відповідає їх застосуванню.

Спочатку ми вибираємо часовий смуговий фільтр, щоб виокремити рухи або сигнали, які ми хочемо посилити (крок 1). Вибір фільтра, як правило, залежить від програми. Для збільшення руху кращим є фільтр із широкою смугою пропускання; для підсилення кольорів кровотоку більш безшумний результат забезпечує вузька смуга пропускання . На малюнку 3.4 показані частотні характеристики деяких часових фільтрів, використаних у цій роботі. Тут використано ідеальні смугові фільтри для посилення кольору, оскільки вони мають смуги пропускання з різкими частотами зрізу.

Фільтри низького порядку можуть бути корисними як для посилення кольору, так і для збільшення руху, а також зручні для реалізації навіть в реальному часі. Загалом, ми використовували два фільтра нижніх частот першого порядку з частотами зрізу ωl і ωh для створення смугового фільтра.



Малюнок 3.4 (Часові фільтри, використані в роботі. Ідеальні фільтри (a) і (b). Фільтр Баттерворта (c) використовується для перетворення діапазону частот, заданого користувачем, у структурі другого порядку і використовується в алгоритмі. Фільтр другого порядку (d). Ці фільтри другого порядку мають ширшу смугу пропускання, ніж ідеальний фільтр.)

Далі ми вибираємо потрібний коефіцієнт збільшення, α, і відсічення просторової частоти λc (кроки 2 і 3). Тоді як рівняння 14 можна використовувати як орієнтир, на практиці ми можемо спробувати різні значення α і λc для досягнення бажаного результату. Користувачі можуть вибрати більший α, який порушує обмеження на перебільшення конкретних рухів або зміни кольору, ціною чому стає збільшення шуму або поява більшої кількості артефактів.

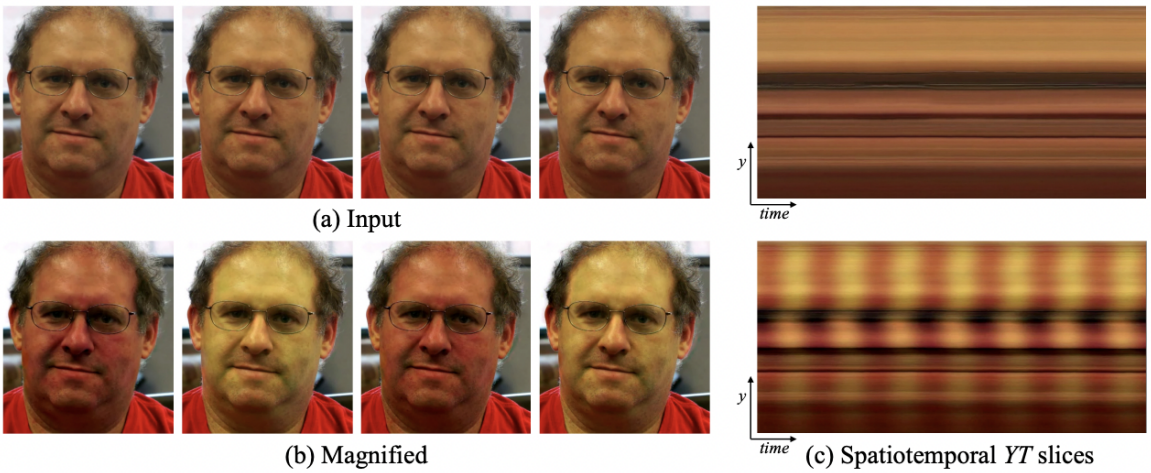
У деяких випадках можна врахувати артефакти відсікання кольору, ослабляючи кольорові компоненти кожного кадру. Користувачі можуть послабити компоненти кольорів перед перетворенням у вихідний колірний простір. Для посилення кольору що виявляє пульс людини, де ми прагнемо підкреслити низькі зміни просторової частоти, ми можемо встановити α = 0 для просторових довжин хвиль нижче λc. Для відео зі збільшенням руху ми можемо використовувати лінійний перехід з наростанням для α (крок 4).

Проведено оцінку методу посилення кольору, використовуючи кілька відео: два відео дорослих з різним кольором шкіри та одне відео новонародженої дитини. Дорослий суб’єкт зі світлішим кольором обличчя (Малюнок 3.5), та людина з темнішим (Малюнок 3.6). В обох відео головною метою було посилити зміну кольору, в той час як коли кров змінює своє місцеположення в обличчі.

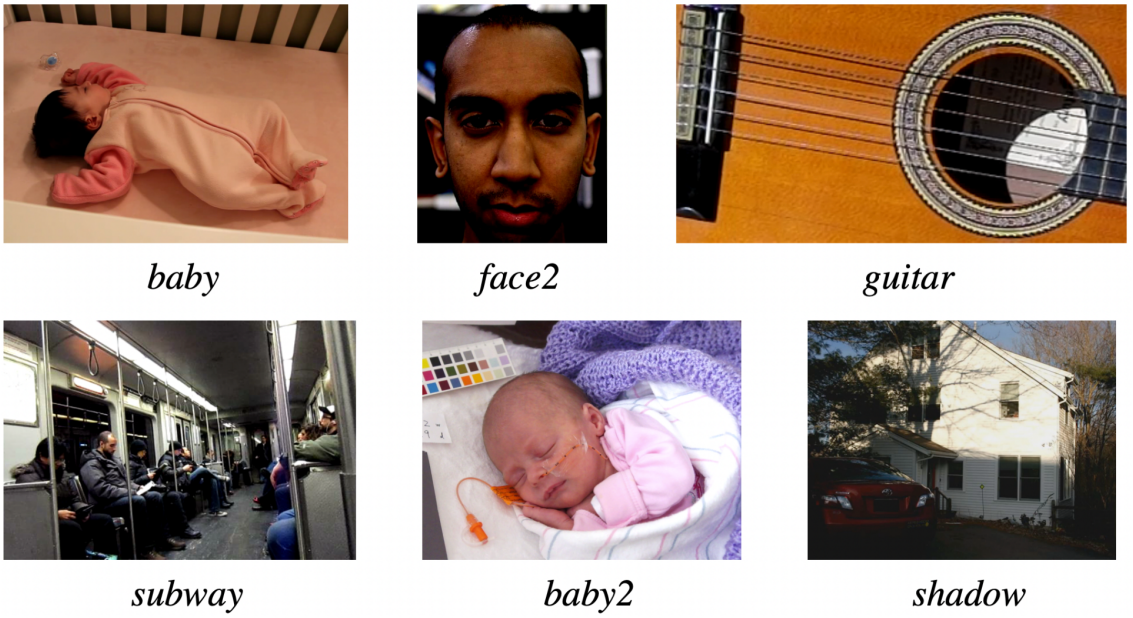
Як на обличчі з малюнку 3.5, так і на обличчі з малюнку 3.6, ми застосували піраміду Лапласа і прирівняли α для двох найтонших рівнів на 0. По суті, ми зменшили дискретизацію та застосували просторовий фільтр нижніх частот до кожного кадру. Таким чином було зменшено квантування і шум, а також посилено тонкий імпульсний сигнал, який нас цікавить.

Потім для кожного відео ми пропускали послідовність кадрів через ідеальний смуговий фільтр із смугою пропускання від 0,83 Гц до 1 Гц (від 50 до 60 ударів на хвилину). В результаті значення $\alpha \approx 100$ і $\lambda c \approx 1000$ було застосовано до результуючого просторово низькочастотного сигналу, щоб якомога більше підкреслити зміну кольору.

Остаточне відео було сформовано шляхом додавання цього сигналу назад до оригіналу. Тепер ми можемо побачити періодичні зміни частоти серцевих скорочень, що змінюються від зеленого до червоного з процесом притоку крові до обличчя.



Малюнок 3.5 (Приклад використання системи Eulerian Video Magnification для візуалізації пульсу людини. (a) Чотири кадри з оригінальної відеоряду (обличчя). (b) Ті самі чотири кадри з посиленим імпульсним сигналом суб'єкта. (c) Вертикальна лінія розгортки вхідного (зверху) та вихідного (знизу) відео, нанесеного з часом, показує, як метод посилює періодичні зміни кольору. У вхідній послідовності сигнал непомітний, але в збільшеній послідовності зміна чітка.)

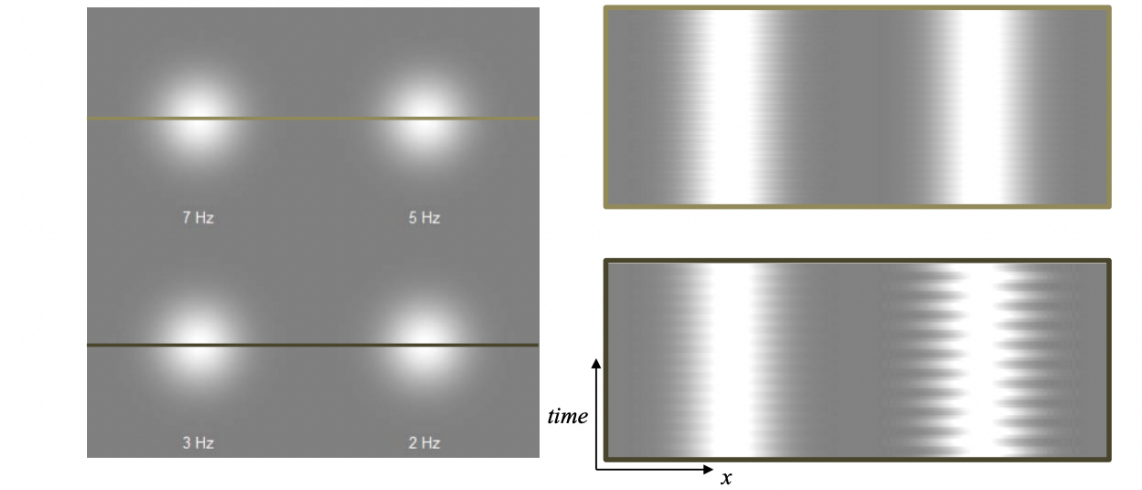


Малюнок 3.6 (Кадри з додаткових відео, що демонструють техніку обробки)

baby2 — це відео новонародженої дитини, записане в дитячому відділенні лікарні Вінчестер у Массачусетсі. На додаток до відео звідти ж отримано основні життєві показники життєдіяльності з монітора в лікарні. Ця інформація була використана для того, щоб підтвердити точність оцінки серцевого ритму та переконатися, що сигнал посилення кольору, отриманий за допомогою вищеописаного методу, відповідає пульсу виміряному монітором.

Щоб оцінити наш метод збільшення руху, ми використали кілька різних відео: *face* (Малюнок 3.5), *sim4* (Малюнок 3.7), *wrist (*зап'ястя) (Малюнок 3.8), *camera* (Малюнок 1.1), *baby, face 2, guitar, subway, shadow і baby2* (Малюнок 3.6). Для всіх відео ми використовували стандартну піраміду Лапласа для фільтрації по простору. Для відео, де ми хотіли

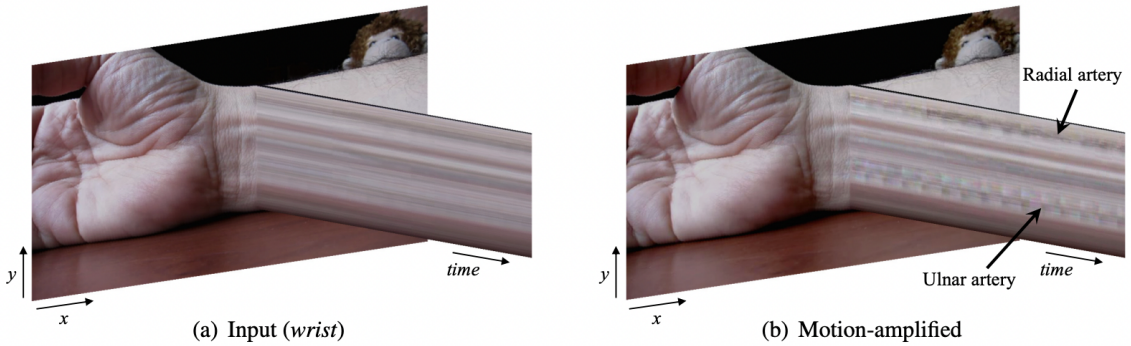
підкреслити рухи на певних часових частотах (наприклад, у *sim4* та *guitar*), ми використовували ідеальні смугові фільтри. У *sim4* та *guitar* ми змогли вибірково посилити рух певної “краплі” або гітарної струни за допомогою смугового фільтра, налаштованого на частоту коливань об’єкта. Значення, що використовуються для α і λc для всіх відео, що згадуються у цій роботі, наведені в таблиці 3.1.



(a) Input (*sim4*) (b) Motion-amplified spatiotemporal slices

Малюнок 3.7 (Вибіркове посилення руху на синтетичній послідовності (sim4 ліворуч). Відеопослідовність містить “краплі”, які коливаються на різних тимчасових частотах, як показано на вхідному кадрі. Ми застосовуємо наш метод, використовуючи ідеальний часовий смуговий фільтр 1-3 Гц, щоб посилити лише рухи, що відбуваються в межах зазначеної смуги пропускання. У (b) ми показуємо просторово-часові фрагменти з отриманого відео, які показують різні часові частоти та посилений рух “краплі”, що коливається з частотою 2 Гц. Зауважимо, що обробка простору-часу застосовується рівномірно до всіх пікселів.)

Для відео, де ми були зацікавлені у виявленні руху “великого розміру”, але невеликого за розмахом (амплітудою), ми використовували часові фільтри з більш широкою смугою пропускання. Наприклад, для відео *face2* ми використали фільтр другого порядку з областями повільного згортання. Змінивши часовий фільтр, ми змогли збільшити рух голови, а не посилити зміну кольору шкіри. Відповідно, для збільшення руху обрано $\alpha = 20$, $\lambda c = 80$.



(a) Input (*wrist*) (b) Motion-amplified

Малюнок 3.8 (Ейлерове відеозбільшення, яке використовується для посилення малих рухів кровоносних судин, що виникають внаслідок кровотоку. Для цього відео ми налаштували часовий фільтр на діапазон частот, який включає частоту серцевих скорочень — 0,88 Гц (53 ударів на хвилину) — і встановили коефіцієнт посилення на $\alpha = 10$. Щоб зменшити збільшення руху невідповідних об’єктів, ми застосували надану користувачем маску для посилення зони біля зап’ястя. Рух променевої та ліктьової артерій майже не видно на вхідному відео (a), знятому стандартною камерою, але значно помітніший на виході зі збільшеним рухом (b). Рух пульсуючих артерій більш помітний при спостереженні просторово-часового зрізу зап’ястя (a) і (b).)

Video	α	λ_c	ω_l (Hz)	ω_h (Hz)	f_s (Hz)
baby	10	16	0.4	3	30
baby2	150	600	2.33	2.67	30
camera	120	20	45	100	300
face	100	1000	0.83	1	30
face2 motion	20	80	0.83	1	30
face2 pulse	120	960	0.83	1	30
guitar Low E	50	40	72	92	600
guitar A	100	40	100	120	600
shadow	5	48	0.5	10	30
subway	60	90	3.6	6.2	30
wrist	10	80	0.4	3	30

Таблиця 3.1 (Таблиця значень α, λc, ωl, ωh, що використовуються для створення різних вихідних відео. Для face2 використовуються два різні набори параметрів — один для посилення імпульсу, інший для посилення руху. Для гітари для «вибору» різних коливальних гітарних струн використовуються різні частоти та значення (α, λc). fs – частота кадрів камери.)

За допомогою широкосмугових часових фільтрів і встановлення α і λc відповідно до рівняння, відповідно до рівняння 14, наш метод здатний виявити тонкі рухи, як на відео з камерою та зап’ястям. Для відео з камери ми використовували камеру з частотою дискретизації 300 Гц, щоб записати цифрову дзеркальну камеру, яка вібрує під час зйомки фотографій. Вібрація, спричинена рухомим дзеркалом в об’єктиві, хоча й невидима неозброєним оком, була виявлена описаним підходом.

Даний метод також здатний перебільшувати видимі, але ледь помітні рухи, як це видно на відео *baby*, *face2* та *subway*. У прикладі з метро ми навмисно посилили рух черезмірно (рівняння 14), де виконується наближення першого порядку, щоб збільшити ефект і продемонструвати артефакти алгоритму.

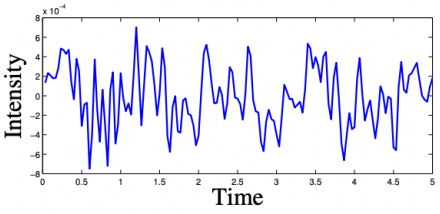
Зауважимо, що більшість прикладів у нашій роботі містять коливальні рухи, оскільки такий рух зазвичай має більшу тривалість і менші амплітуди. Однак наш метод також можна використовувати для посилення неперіодичних рухів, якщо вони знаходяться в межах смуги пропускання часового смугового фільтра. Наприклад, у відео *shadow* ми обробляємо відео, як сонячна тінь рухається лінійно, але непомітно протягом 15 секунд. Збільшена версія дає змогу побачити зміни навіть за цей короткий період часу.

Деякі відео можуть містити області часових сигналів, які не потребують посилення або які, коли вони посилені, є непридатними для сприйняття. Завдяки обробці Ейлера ми можемо легко дозволити користувачеві вручну обмежити збільшення певними ділянками, позначаючи їх на відео, як це було використано для обличчя та зап’ястя.

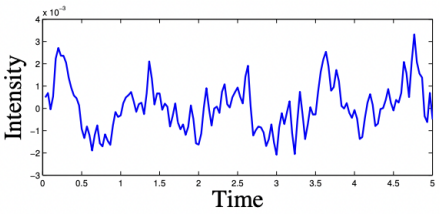
3.3 Чутливість до шуму

Зміна амплітуди сигналу, що цікавить, часто набагато менша, ніж шум, властивий відеофайлам на вході. У таких випадках пряме збільшення значень пікселів не дасть бажаного сигналу. Для посилення цих малих сигналів можна використовувати просторову фільтрацію. Однак, якщо застосований просторовий фільтр недостатньо великий, сигнал, що цікавить, не буде виявлено (Малюнок 3.9).

(a) Input noisy frame



(b) Insufficient spatial pooling



(c) Sufficient spatial pooling

Малюнок 3.9 (Належне просторове об'єднання є обов'язковим для виявлення сигналу, що цікавить. (a) Кадр із відео обличчя (Малюнок 3.5) з доданим білим гаусовим шумом (σ = 0,1 пікселя). Праворуч наведені графіки інтенсивності по часу для пікселя, позначеного блакитним кольором на вхідному кадрі, де (b) графік, отриманий під час обробки зашумленої послідовності з тим самим просторовим фільтром, який використовується для обробки вихідної послідовності кадрів і (c) показує графік при використанні фільтра, налаштованого відповідно до оціненого радіусу в рівнянні. 15, біноміальний фільтр розміром 80. Імпульсний сигнал не видно в (b), оскільки рівень шуму вищий за потужність сигналу, тоді як у (c) імпульс чітко видно.)

Припускаючи, що шум є білим, із нульовим середнім значенням і нерухомий у широкому сенсі відносно простору, можна показати, що просторова фільтрація низьких частот зменшує дисперсію шуму відповідно до площі фільтра низьких частот. Щоб підвищити потужність конкретного сигналу, наприклад, імпульсного сигналу на обличчі, ми

можемо використовувати просторові характеристики сигналу для оцінки розміру просторового фільтра.

Нехай рівень потужності шуму дорівнює σ^2, а наша попередня потужність сигналу на просторових частотах дорівнює $S(\lambda)$. Ми хочемо знайти просторовий фільтр низьких частот з радіусом r, щоб потужність сигналу була більшою за шум у відфільтрованому частотному діапазоні. Довжина хвилі відсікання такого фільтра пропорційна його радіусу r, тому попередній сигнал може бути представлений як $S(r)$.

Потужність шуму σ^2 можна оцінити шляхом дослідження значень пікселів у стабільній області сцени або за допомогою методики, як у [Liu et al. 2006]. Оскільки рівень потужності відфільтрованого шуму, σ'^2, обернено пропорційний r^2, ми можемо вирішити наступне рівняння для r,

$$S(r) = \sigma'^2 = k\frac{\sigma^2}{r^2} \qquad (15)$$

де k – константа, яка залежить від форми фільтра низьких частот. Це рівняння дає оцінку розміру просторового фільтра, необхідного для виявлення сигналу при певному рівні потужності шуму.

3.4 Порівняння обробки Ейлера та Лагранжа

Оскільки обидва методи використовують різні підходи до руху — підходи Лагранжа явно відстежують рухи, а наш ейлерів — ні, їх можна використовувати для додаткових областей руху.

Лагранжеві підходи, напр. [Liu et al. 2005], краще працюють для покращення рухів точкових елементів і підтримки більших коефіцієнтів посилення, тоді як наш метод Ейлера краще підходить для більш гладких структур і невеликих посилень.

Зауважимо, що наша техніка не передбачає певних типів рухів. Аналіз ряду Тейлора першого порядку може виконуватися для загальних малих двовимірних рухів уздовж загальних шляхів. Обидва методи однаково чутливі до часових характеристик шуму, тоді як лагранжевий метод обробки має додаткові джерела похибок що пропорційні просторовим характеристикам шуму, за рахунок явної оцінки руху [Liu et al. 2005].

Похибка Ейлера, з іншого боку, зростає квадратично з α і є більш чутливою до високих просторових частот. Загалом, це означає, що ейлерівське збільшення було б кращим, ніж лагранжове збільшення для невеликих посилень і більших рівнів шуму.

Ми підтвердили цей аналіз на синтетичній послідовності двовимірного косинуса, який коливається з часовими частотами 2 Гц і 0,1 інтенсивності пікселя в просторі з адитивним білим просторово-часовим гаусовим шумом з нульовим середнім і стандартним відхиленням σ. Область, де ейлерів підхід перевершує результати Лагранжа, також відповідає очікуванням. Метод Лагранжа більш чутливий до збільшення просторового шуму, тоді як на ейлерівську похибку він майже не впливає. Хоча різні схеми регуляризації, що використовуються для оцінки руху (які важче проаналізувати теоретично), можуть зменшити помилку Лагранжа, вони суттєво не змінили результат [Liu et al. 2005].

Висновки:

В результаті невдалих спроб наївного аналізу методами Python, без врахування відношення параметрів пікселів, що знаходяться поруч один з одним, відчувається необхідність у застосуванні потужнішого методу обробки зображень. Тому використання методу Ейлерового збільшення, яке враховує інформацію про просторову локалізацію пікселів цілком обумовлене.

В рамках роботи алгоритму, деякі відео можуть містити області часових сигналів, які не потребують посилення або які, коли вони посилені, є непридатними для сприйняття. Завдяки обробці Ейлера користувач може вручну обмежити збільшення певними ділянками, позначаючи їх на відео.

Порівнюючи підхід Ейлера та Лагранжа, загалом, наші експерименти показують, що для невеликих посилень ейлерів підхід забезпечує кращий баланс між продуктивністю та ефективністю.

ВИСНОВКИ

Ми описали простий метод, який на вхід бере відеофайли та збільшує малі зміни кольору та непомітні рухи. Щоб посилити рух, наш метод не виконує відстеження ознак чи обчислення оптичного потоку, а лише збільшує часові зміни кольору за допомогою просторово-часової обробки. Цей метод на основі Ейлера, який тимчасово обробляє пікселі у фіксованій просторовій області, успішно виявляє інформативні сигнали та посилює невеликі рухи у відео в реальному світі.

Існуюча проблема рухів та змін, які відволікають від основних часових сигналів, що можуть цікавити дослідника і, як наслідок, задача їх позбутися, також може бути вирішена на основі проробленої роботи. Це може бути особливо корисним при роботі з уповільненими послідовностями, які часто використовуються для більш тривалого медичного та наукового аналізу, де динамічні сцени записуються протягом порівняно великого періоду часу.

З огляду на реалізоване збільшення зрушення, які занадто малі, щоб їх можна було побачити неозброєним оком, можна зробити висновок про потужність математичних методів, що надають людині все більш нові здатності бачити й аналізувати інформацію, яка так необхідна сучасності. Ті події, що відбуваються у світі, та залишаються за межами нашої уваги тепер доступні для будь-кого, з камерою чи іншими засобами відеоспостереження.

ПЕРЕЛІК ДЖЕРЕЛ І ПОСИЛАНЬ

[1] Michael Rubinstein Analysis and Visualization of Temporal Variations in Video

[2] http://people.csail.mit.edu/mrub/vidmag/

[3] Hao-Yu Wu, Michael Rubinstein, Eugene Shih, John Guttag, Fredo Durand Eulerian Video Magnification for Revealing Subtle Changes in the World ´ 1 William Freeman1 1MIT CSAIL 2Quanta Research Cambridge, Inc.

[4] Burt and Adelson 1983

[5] Liu et al. 2005; Wang та ін. 2006

[6] Lucas and Kanade 1981; Horn and Schunck 1981

[7] E. Adelson, C. Anderson et al., “Пірамідні методи в обробці зображень”, RCA Engineer, vol. 29, № 6, C. 33–41, 1984

[8] H. Wilson and J. Bergen' "A four mechanism model for threshold special vision", Vision Research. Vol. 19, pp. l9-31, 1979.

[9] Hao-Yu Wu, Michael Rubinstein, Eugene Shih, John Guttag, Frédo Durand, William T. Freeman Eulerian Video Magnification for Revealing Subtle Changes in the World ACM Transactions on Graphics, Volume 31, Number 4 (Proc. SIGGRAPH), 2012

[20] K Goda, KK Tsia, and B Jalali. Serial time-encoded amplified imaging for real-time observation of fast dynamic phenomena. Nature, 458(7242):1145–1149, 2009.

[12] Google Earth Engine. http://earthengine.google.org/

[44] Michael Rubinstein, Ce Liu, Peter Sand, Fredo Durand, and William T. Freeman. Motion denoising with application to time-lapse photography. IEEE Computer Vision and Pattern Recognition (CVPR), pages 313–320, June 2011.

[59] Hao-Yu Wu, Michael Rubinstein, Eugene Shih, John Guttag, Fredo Durand, and William T. Freeman. Eulerian video magnification for revealing subtle changes in the world. ACM Transactions on Graphics (Proceedings SIGGRAPH), 31(4):65:1–65:8, July 2012.

[45] Michael Rubinstein, Neal Wadhwa, Fredo Durand, and William T. Freeman. Revealing invisible changes in the world. Science, 339(6119):519, February 2013.

[55] Neal Wadhwa, Michael Rubinstein, Fredo Durand, and William T. Freeman. Phasebased video motion processing. ACM Transactions on Graphics (Proc. SIGGRAPH), 32(3):80:1–80:12, July 2013.

[44] Michael Rubinstein, Ce Liu, Peter Sand, Fredo Durand, and William T. Freeman. Motion denoising with application to time-lapse photography. IEEE Computer Vision and Pattern Recognition (CVPR), pages 313–320, June 2011.

[59] Hao-Yu Wu, Michael Rubinstein, Eugene Shih, John Guttag, Fredo Durand, and William T. Freeman. Eulerian video magnification for revealing

subtle changes in the world. ACM Transactions on Graphics (Proceedings SIGGRAPH), 31(4):65:1–65:8, July 2012.

[55] Neal Wadhwa, Michael Rubinstein, Fredo Durand, and William T. Freeman. Phasebased video motion processing. ACM Transactions on Graphics (Proc. SIGGRAPH), 32(3):80:1–80:12, July 2013.

[56] WANG, J., DRUCKER, S. M., AGRAWALA, M., AND COHEN, M. F. 2006. The cartoon animation filter. ACM Trans. Graph. 25, 1169–1173.

ДОДАТКИ

КОД ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

```
from google.colab import drive
drive.mount('/content/drive/')

# vid = /content/drive/MyDrive/vidmag/face.mp4
```

Функції

```
from PIL import Image
import numpy as np
import glob
import cv2
from matplotlib import pyplot as plt
import math

from google.colab.patches import cv2_imshow

def loadVid(path):

  vidcap = cv2.VideoCapture(path)
  success,image = vidcap.read()
  framesCount = 0
  while success:
    cv2.imwrite("frame%d.jpg" % framesCount, image)     # save frame as JPEG
file
    success,image = vidcap.read()

    # print('Read a new frame: ', success)
    framesCount += 1



  print("Video has been loaded. Frames count is", framesCount)
```

```
    frames = []
    files = glob.glob ("*.jpg")

    for frame in files:
        image = cv2.imread(frame)
        frames.append(image)

    frames = np.array(frames)
    print('frames array shape:', frames.shape)
    # frames shape: (301, 592, 528, 3)
      # 301 frames
      # 592 high on 528 wide pixels each frame
      # 3 channels each pixel

    return frames



def frameInfo(path):

    im = np.array(Image.open(path))

    print("type(im) " , type(im))
    # <class 'numpy.ndarray'>

    print("im.shape ", im.shape)
    # (592, 528, 3)

    print("im.dtype " , im.dtype)
    # uint8

    print("pixel[0,0] is im[0, 0] = \n", im[0, 0]) #?
    # [100 150  60]
    # the value at (y, x) = (100, 150),
    # the 100th row and
    # 150th column of pixels

    print("row 0 is im[0, :] = \n", im[0, :])
    # the value at (y, :) = (0, :),
    # the 100th row and
    # all column of pixels

    # print(type(im[100, 105]))
    # <class 'numpy.ndarray'>
```

```
  print("pixel shape (im[0, 0]).shape = ", (im[0, 0]).shape)
  # (3,)

  print("row shape (im[100, :]).shape = " , (im[100, :]).shape)
  # (528, 3)


def timePicFunc(framesArray, pxLineNumber):

  timePic = np.zeros([framesArray.shape[0], framesArray[0].shape[1],
framesArray[0].shape[2]])
  # <class 'numpy.ndarray'> (592, 528, 3)

  for frameIndex in range(np.array(framesArray).shape[0]):
    timePic[frameIndex] = framesArray[frameIndex][pxLineNumber]

  cv2_imshow(timePic)
  # display(timePic)

  return timePic
```

Наївний аналіз

```
frames = loadVid('/content/drive/MyDrive/vidmag/face.mp4')
cv2_imshow(frames[0]) #meet Steve


frameInfo('/content/frame0.jpg')


timePic = timePicFunc(frames, 0)

timePic = timePicFunc(frames, 53)
print(type(timePic))


# newH = 1500
# resized = cv2.resize(timePic, (timePic.shape[1], newH))
resized = timePic #resizing makes float numbers
# print('Resized Dimensions : ',resized.shape)
# cv2_imshow(resized)


# crop
  #wide line with less colour changings
```

```
crpW = resized[:, 166:320]
cv2_imshow(crpW)
crpW1 = crpW[:,:,0]
cv2_imshow((crpW1 - crpW1.min())/(crpW1.max() - crpW1.min())* 255)

  #line
crp = resized[:, 319:320]
cv2_imshow(crp)
crpLineRes = cv2.resize(crp, (crp.shape[0], 1000))
crpLineRes = crpLineRes[:,:,0]
cv2_imshow((crpLineRes - crpLineRes.min())/(crpLineRes.max() -
crpLineRes.min())* 255)

#separating into three channels
crp1 = crp[:,:,0]
crp2 = crp[:,:,1]
crp3 = crp[:,:,2]
cv2_imshow((crp1 - crp1.min())/(crp1.max() - crp1.min())* 255)
# cv2_imshow((crp2 - crp2.min())/(crp2.max() - crp2.min())* 255)
# cv2_imshow((crp3 - crp3.min())/(crp3.max() - crp3.min())* 255)

# Using Numpy to create an array X
X = np.arange(0,301)

# Assign variables to the y axis part of the curve
y1 = (crp1 - crp1.min())/(crp1.max() - crp1.min())* 255
y1 = y1[:,0]
y2 = (crp2 - crp2.min())/(crp2.max() - crp2.min())* 255
y2 = y2[:,0]
y3 = (crp3 - crp3.min())/(crp3.max() - crp3.min())* 255
y3 = y3[:,0]
plt.rcParams["figure.figsize"] = (15,5)
plt.plot(X, y1, color='r', label='red')
plt.plot(X, y2, color='g', label='green')
plt.plot(X, y3, color='b', label='blue')
plt.xlabel("frame number")
plt.ylabel("px value")
plt.title("normalized pixel values variation in time")
plt.legend()
plt.show()


# resized1 = cv2.resize(y1, (y1.shape[0], 1000))
# X = np.arange(0,1000)
# plt.plot(X, resized1, color='r', label='y1')
```

```
# plt.rcParams["figure.figsize"] = (30,10)
# plt.show()



# 30 frames per second
X = np.arange(0, 301/30, 1/30)
y1 = (crp1 - crp1.min())/(crp1.max() - crp1.min())* 255
y1 = y1[:,0]
y2 = (crp2 - crp2.min())/(crp2.max() - crp2.min())* 255
y2 = y2[:,0]
y3 = (crp3 - crp3.min())/(crp3.max() - crp3.min())* 255
y3 = y3[:,0]
plt.rcParams["figure.figsize"] = (15,5)
plt.plot(X, y1, color='r', label='red')
plt.plot(X, y2, color='g', label='green')
plt.plot(X, y3, color='b', label='blue')
plt.xlabel("time, s")
plt.ylabel("px value")
plt.title("normalized pixel values variation in time")
plt.legend()
plt.show()


import scipy
import scipy.fftpack
import pylab
from scipy import pi
t = scipy.linspace(0, 301/30, 301)

# acc = lambda t: 10*scipy.sin(2*pi*2.0*t) + 5*scipy.sin(2*pi*8.0*t) +
2*scipy.random.random(len(t))
y1 = (crp1 - crp1.min())/(crp1.max() - crp1.min())* 255
y1 = y1[:, 0]
# signal = acc(t)

FFT = abs(scipy.fft(y1))
freqs = scipy.fftpack.fftfreq(y1.size, t[1]-t[0])

pylab.subplot(211)
pylab.plot(t, y1)
pylab.subplot(212)
pylab.plot(freqs,20*scipy.log10(FFT),'x')
pylab.show()
```

Конволюція

(не оптимізована)

```
rgb_weights = [0.2989, 0.5870, 0.1140]

image = np.dot(frames[0][...,:3], rgb_weights)
cv2_imshow(image)


ker = np.ones((5, 5))/25
# ker[4][4] = 1
cv2_imshow(ker)

# image = np.pad(a, pad_width=int(ker.shape[0]/2), mode='constant',
constant_values=0)
cv2_imshow(image)


# image = np.zeros((9, 9))
rgb_weights = [0.2989, 0.5870, 0.1140]
image = np.dot(frames[0][...,:3], rgb_weights)
cv2_imshow(image)

# ker = np.array([[0, 0, 0],[0, 0, 0],[0, 0, 1]])
# cv2_imshow(ker)


def Convolve(image, ker, bdSame):

  # if bdSame:
  hk = int((ker.shape[0]/2))
  imageB = np.pad(image, pad_width=int(ker.shape[0]/2), mode='constant',
constant_values=0)

  # flip the kernel
  ker = np.flip(ker)

  newPic = np.zeros((image.shape[0], image.shape[1]))
  # cv2_imshow(image)
  idx = np.arange(-int((ker.shape[0]/2)), int((ker.shape[0]/2))+1)
  # print(idx.shape)
  # print(idx)

  for x in range(image.shape[0]-1):
    for y in range(image.shape[1]-1):
```

```
      # print("counting for pixel [",x, ", ", y, "]")
      # print("image[x][y] = ", image[x][y])
      for i in idx:
        for j in idx:
          # print("[", i, ", ", j, "]")
          newPic[x][y] +=
imageB[x+int((ker.shape[0]/2))-i][y+int((ker.shape[0]/2))-j]*ker[(np.where(id
x == i))[0][0]][(np.where(idx == j))[0][0]]
          # newPic[x][y] += imageB[x-i][y-j]*ker[(np.where(idx ==
i))[0][0]][(np.where(idx == j))[0][0]]

          # print("newPic[x][y] = ", newPic[x][y])

  # else:
  #   pass



  return newPic

newPic = Convolve(image, ker, True)
cv2_imshow(newPic)


# better try (less resourceful)

def Convolve(u, G, bdSame):
 n = u.shape[0] - 1
 # print("n = ", n)

 N = u.shape[1] - 1
 # print("N = ", N)

 pd = np.array([int(G.shape[0]/2), int(G.shape[1]/2)])
 #print("pd = ", pd)

 v = np.zeros((u.shape[0], u.shape[1]))
 # print("v = \n", v)
 # print("v.shape = ", v.shape)

  # padding
 idx0 = np.arange(-int((G.shape[0]/2)), int((G.shape[0]/2))+1)
 idx1 = np.arange(-int((G.shape[1]/2)), int((G.shape[1]/2))+1)
 # print("idx0 = ", idx0)
```

```
 # print("idx1 = ", idx1)

 if bdSame: #zeros border = True
   #extending u
   u = np.pad(u, pad_width=pd, mode='constant', constant_values=0)
   # print("u = \n", u)
   # print("u.shape = ", u.shape)

   # loop
   for i in idx0:
     for j in idx1:
       # print("\n\ni, j = ", i, j, "(but in python it's ", i+pd[0], j+pd[1],
")")
       # print("G[", i, ",", j, "] = " , G[i+pd[0], j+pd[1] ], " \n *")
       # print("u [", i, ":", n+i+1, ",", j, ":", N+j+1, "] = \n",
u[i+pd[0]:n+i+1+pd[0], j+pd[1]:N+j+1+pd[1]], "\n =")
       v += G[i+pd[0], j+pd[1]] * u[i+pd[0]:n+i+1+pd[0], j+pd[1]:N+j+1+pd[1]]
       # print("v = \n", v)

 else: #periodic border = False
   for i in idx0:
     for j in idx1:
       # print("\n\ni, j = ", i, j, "(but in python it's ", i+pd[0], j+pd[1],
")")
       # print("G[", i, ",", j, "] = " , G[i+pd[0], j+pd[1] ], " \n *")
       # print("u [", i%n, ":", (n+i+1)%n, ",", j%N, ":", (N+j+1)%N, "] =
\n", np.roll(u, (i, j), axis = (0,1)), "\n =")
       v += G[i+pd[0], j+pd[1]] * np.roll(u, (i, j), axis = (0,1))
       # print("v = \n", v)

 return v
```

Гаусові піраміди

```
# generate Gaussian pyramid

def GPyr(pic, depth):

  G = pic.copy()
  gp = [G]
  for i in range(depth):
    G = cv2.pyrDown(G)
    gp.append(G)
```

```
  return np.array(gp)

gp = GPyr(frames[0], 4)
for ph in gp:
  cv2_imshow(ph)


# Expand Gaussian pyramid

def GPyrExp(gp):
  pic = gp[0]
  gpE = []
  # print(type(gpE))
  for i in range(len(gp)):
    # print(i)
    pic = cv2.resize(gp[i], (gp[0].shape[1], gp[0].shape[0]), interpolation =
cv2.INTER_LINEAR)
    gpE.append(pic)
    # print(i)
  return np.array(gpE)

gpE = GPyrExp(gp)

# print(type(gpE[0]))
for pic in gpE:
  cv2_imshow(pic)
```

Лапласові піраміди

```
# generate Laplacian Pyramid

def LPyr(gpE):

  lp = []
  for i in range(len(gpE)-1):
    # print(i)
    L = cv2.subtract(gpE[i],gpE[i+1])
    lp.append(L)
  return lp

lp = LPyr(gpE)

for pic in lp:
  cv2_imshow(pic)
```

```
# print(len(lp))
#grayscaling
rgb_weights = [0.2989, 0.5870, 0.1140]
u = np.dot(frames[0][...,:3], rgb_weights)
print("u = \n")
cv2_imshow(u)
print("u.shape = ", u.shape)

G = gkern(5)
print("G = \n", G)
print("G.shape = ", G.shape)
```

Швидке перетворення Фур'є

```
def calculate_2dft(input):
   ft = np.fft.ifftshift(input)
   ft = np.fft.fft2(ft)
   return np.fft.fftshift(ft)

# Read and process image

rgb_weights = [0.2989, 0.5870, 0.1140]
image = np.dot(frames[0][...,:3], rgb_weights)


plt.set_cmap("gray")

ft = calculate_2dft(image)

plt.subplot(121)
plt.imshow(image)
plt.axis("off")
plt.subplot(122)
plt.imshow(np.log(abs(ft)))
plt.axis("off")
plt.show()
```

Фільтр низьких частот

```
# read input and convert to grayscale
img = image

# do dft saving as complex output
dft = np.fft.fft2(img, axes=(0,1))

# apply shift of origin to center of image
```

```
dft_shift = np.fft.fftshift(dft)

# generate spectrum from magnitude image (for viewing only)
mag = np.abs(dft_shift)
spec = np.log(mag) / 20

# create circle mask
radius = 40
mask = np.zeros_like(img)
cy = mask.shape[0] // 2
cx = mask.shape[1] // 2
cv2.circle(mask, (cx,cy), radius, (255,255,255), -1)[0]

# blur the mask
mask2 = cv2.GaussianBlur(mask, (19,19), 0)

# apply mask to dft_shift
dft_shift_masked = np.multiply(dft_shift,mask) / 255
dft_shift_masked2 = np.multiply(dft_shift,mask2) / 255


# shift origin from center to upper left corner
back_ishift = np.fft.ifftshift(dft_shift)
back_ishift_masked = np.fft.ifftshift(dft_shift_masked)
back_ishift_masked2 = np.fft.ifftshift(dft_shift_masked2)


# do idft saving as complex output
img_back = np.fft.ifft2(back_ishift, axes=(0,1))
img_filtered = np.fft.ifft2(back_ishift_masked, axes=(0,1))
img_filtered2 = np.fft.ifft2(back_ishift_masked2, axes=(0,1))

# combine complex real and imaginary components to form (the magnitude for)
the original image again
img_back = np.abs(img_back).clip(0,255).astype(np.uint8)
img_filtered = np.abs(img_filtered).clip(0,255).astype(np.uint8)
img_filtered2 = np.abs(img_filtered2).clip(0,255).astype(np.uint8)


cv2_imshow(img)
cv2_imshow(spec)
cv2_imshow(mask)
cv2_imshow(mask2)
cv2_imshow(img_back)
cv2_imshow(img_filtered)
```

```
cv2_imshow(img_filtered2)
```

Фільтр високих частот

```
# read input and convert to grayscale
img = image

# do dft saving as complex output
dft = np.fft.fft2(img, axes=(0,1))

# apply shift of origin to center of image
dft_shift = np.fft.fftshift(dft)

# generate spectrum from magnitude image (for viewing only)
mag = np.abs(dft_shift)
spec = np.log(mag) / 20

# create white circle mask on black background and invert so black circle on
white background
radius = 32
mask = np.zeros_like(img)
cy = mask.shape[0] // 2
cx = mask.shape[1] // 2
cv2.circle(mask, (cx,cy), radius, (255,255,255), -1)[0]
mask = 255 - mask

# blur the mask
mask2 = cv2.GaussianBlur(mask, (19,19), 0)

# apply mask to dft_shift
dft_shift_masked = np.multiply(dft_shift,mask) / 255
dft_shift_masked2 = np.multiply(dft_shift,mask2) / 255


# shift origin from center to upper left corner
back_ishift = np.fft.ifftshift(dft_shift)
back_ishift_masked = np.fft.ifftshift(dft_shift_masked)
back_ishift_masked2 = np.fft.ifftshift(dft_shift_masked2)


# do idft saving as complex output
img_back = np.fft.ifft2(back_ishift, axes=(0,1))
img_filtered = np.fft.ifft2(back_ishift_masked, axes=(0,1))
img_filtered2 = np.fft.ifft2(back_ishift_masked2, axes=(0,1))
```

```
# combine complex real and imaginary components to form (the magnitude for)
the original image again
# multiply by 3 to increase brightness
img_back = np.abs(img_back).clip(0,255).astype(np.uint8)
img_filtered = np.abs(3*img_filtered).clip(0,255).astype(np.uint8)
img_filtered2 = np.abs(3*img_filtered2).clip(0,255).astype(np.uint8)


cv2_imshow(img)
cv2_imshow(spec)
cv2_imshow(mask)
cv2_imshow(mask2)
cv2_imshow(img_back)
cv2_imshow(img_filtered)
cv2_imshow(img_filtered2)
```

Реконструкція

```
# now reconstruct
print(type(lp))
lp = np.array(lp)
ls_ = lp[0]
for i in range(1,4):
    ls_ = cv2.pyrUp(ls_)
    ls_ = cv2.add(ls_, lp[i])
```