Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп‘ютерних наук
Кафедра Програмної інженерії

АТЕСТАЦІЙНА РОБОТА (ПРОЕКТ)

Пояснювальна записка

другий (магістерський)

«Дослідження ефективності детекторів особливих точок зображення»

Виконав: студент 2 курсу, групи ПЗСм-18-1
спеціальності
121- Інженерія програмного забезпечення
освітньо-професійної програми
Програмне забезпечення систем
Зимарев К.Д.

Керівник: проф. Смеляков К.С.

Допускається до захисту
Зав. кафедри, проф. _____ Дудар З. В.
(підпис)

2019 р.

Харківський національний університет радіоелектроніки

Факультет Комп‘ютерних наук

Кафедра Програмної інженерії

Рівень вищої освіти другий (магістерський)

Спеціальність 121- Інженерія програмного забезпечення

Тип програми освітньо-професійна програма

Освітня програма Програмне забезпечення систем

ЗАТВЕРДЖУЮ:

Зав. кафедри _____

(підпис)

«___»_____ 2019 р.

ЗАВДАННЯ

НА АТЕСТАЦІЙНУ РОБОТУ

студентові Зимареву Кирилу Дмитровичу

1. Тема роботи проекту «Дослідження ефективності детекторів особливих точок зображення» затверджена наказом по університету від «___»_____2019р.
№ _____
2. Термін подання студентом роботи (проекту): _____ 2019р.
3. Вихідні дані до роботи: алгоритми детектування особливих точок зображення, пояснювальна записка. Використовувати Linux/Windows, середовище розробки PyCharm.
4. Перелік питань, що потрібно опрацювати в роботі: мета роботи, аналіз проблемної галузі і постановка задачі, дослідження методів детектування особливих точок зображення, експериментальне дослідження роботи алгоритмів детектування.

5. Консультанти розділів роботи

Найменування розділу	Консультант (посада, прізвище, ім’я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Аналіз проблемної області	Проф. Смеляков К.С.		

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи (проекту)	Термін виконання етапів проекту (роботи)	Примітка
1.	Аналіз предметної галузі		
2.	Огляд існуючих методів		
3.	Експериментальне дослідження результатів роботи детекторів		
4.	Підготовка пояснювальної записки		
5.	Попередній захист		
6.	Нормоконтроль, рецензування		
7.	Занесення диплома в електронний архів		
8.	Допуск до захисту у зав. кафедри		

Дата видачі завдання: «____»_____ 2019 р.

Студент _____ Зимарев К.Д.

(підпис)

Керівник роботи _____ проф. Смеляков К.С.

РЕФЕРАТ/ABSTRACT

Атестаційна робота магістра містить: 96с, 5 рисунків, 24 таблиці, 10 джерел.

ДЕТЕКТОР, ІНВАРІАНТНІСТЬ, КОМП'ЮТЕРНИЙ ЗІР, МЕТОД, ОСОБЛИВА ТОЧКА, ЧУТЛИВІСТЬ.

Метою роботи є дослідження ефективності детекторів особливих точок зображення, виявлення переваг та недоліків кожного з них, визначення інваріантності та чутливості методів до різних типів перетворення зображення.

DETECTOR, INVARIANCE, COMPUTER VISION, METHOD, SPECIAL POINT, SENSIVITY.

The aim of the work is a research of the effectiveness of detectors of special image points, to identify the advantages and disadvantages of each, to determine the invariance and sensitivity of methods to different types of image transformation.

ЗМІСТ

ВСТУП

Зіставлення зображень є одним з фундаментальних аспектів багатьох задач комп'ютерного зору, таких як розпізнавання об'єктів, побудова тривимірної сцени на основі декількох зображень, створення стереопари, створення панорамних зображень, motion tracking (стеження за об'єктом у відеопослідовності) і так далі. Через те, що сфера знань, яка вирішує подібні проблеми, ще молода (інтенсивне вивчення почалось лише в 1970-х роках), не існує універсального методу, ідеально розв'язуюча всі задачі такого роду. Хоча, існують методи рішення більш вузьких задач, кожен з яких буде показувати ліпший результат в залежності від задачі.

Людині не важко порівняти два зображення та виділити на них об'єкти, все це відбувається на інтуїтивному рівні та не викликає проблем. Але для комп'ютера зображення – лише набір даних, який не несе за собою інформації про об'єкти, які на ньому відображені. Можна описати два основних підходи до наділення машини таким вмінням.

Перший підхід заключається у співставленні зображень на основі інформації отриманої від аналізу всіх пікселів зображення [3]. В загальному випадку алгоритм такого типу виглядає наступним чином: для кожного пікселю розраховується значення деякої функції, та на основі цих значень отримується певна характеристика в цілому. І тоді задача зводиться до порівняння цих характеристик. І хоча таки алгоритми прості та не потребують великих розрахункових витрат, результат роботи залишає бажати кращого. Причина заключається в їх основній ідеї, в характеристику зображення робить внесок кожен піксель, не зважаючи на цінність цього внеску. Метод повертатиме неприйнятний результат у випадку появи нових об'єктів, перекриття одних об'єктів іншими, поява шуму, зміна положення об'єкта у тривимірному просторі і так далі.

Другий підхід, про який йде мова у даній роботі, заснований на ідеї

використання лише тих пікселів, внесок котрих в загальну характеристику буде значним. Зображення заміняється на набір точок, званих особливими.

Особливою точкою зображення називається така точка, околиця якої $o(m)$ відрізняється від околиці іншої точки зображення o(n) в деякій іншій околиці особливої точки $o_2(m)$.[1]

У більшості алгоритмів в якості околиці точки використовується прямокутне вікно розміром 5x5 пікселів. Харалік та Шапіро [2] визначили наступні умови, котрим повинна відповідати особлива точка:

– відмінність: особлива точка повинна виділятися на фоні та бути унікальною у своїй околиці;

– інваріантність: вибір особливої точки повинен бути незалежним від афінних перетворень;

– стабільність: вибір особливих точок повинен бути стійким до шуму та помилок;

– унікальність: крім локальної відмінності, особлива точка повинна також володіти властивістю глобальної унікальності для того, щоб покращити розрізнення повторюваних паттернів;

– інтерпретованість: особливі точки повинні визначатися таким чином, щоб їх можна було використовувати для аналізу відповідностей так ліпшої інтерпретованості зображення.

Процес порівняння зображень розділяється на три етапи. Перший етап – знаходження особливих точок за допомогою методів, названих детекторами. Ці методи забезпечують інваріантність знаходження одних й тих самих точок відносно перетворень зображення. Хоча, недостатньо використання лише детектора, так як результатом його роботи є безліч координат точок, які різняться на кожному зображенні. Для цього на другому етапі відбувається побудова дескриптора. Дескриптор – це опис точки, який унікально ідентифікує її серед безлічі всіх точок. Дескриптор повинен забезпечувати інваріантність знаходження відповідностей між

точками відносно перетворень зображення. Деякі методи виконують обидві задачі, детектування особливих точок так побудова дескриптора. Третій етап заключається у порівнянні дескрипторів та пошуку точок, співпадаючих на обох зображеннях.

У даній роботі розглянуто такі детектори як:

- детектор Харріса;
- детектор Shi-Tomasi;
- детектор FAST;
- детектор SIFT;
- детектор SURF;
- детектор BRISK.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Аналіз предметної галузі ефективності детекторів особливих точок зображень

Уже кілька десятиліть вчені з різних країн займаються розробкою алгоритмів, що дозволяють навчити комп'ютер бачити так само, як бачить сама людина. Якщо для людей отримувати необхідну інформацію за допомогою зорового каналу є чимось простим і само собою зрозумілим, то навчити комп'ютер подібних речей є і по цей не здійсненним завданням. Безліч IT корпорацій працюють над її вирішенням, але це вимагає великого вкладення людської праці, фінансових витрат і обчислювальних потужностей.

Але існують методи, які дозволяють, хоч і при використанні в вузько спрямованих завданнях, отримати бажаний результат при менших витратах. Вони засновані не на структурі людського апарату аналізу та інтерпретації зображень, а безпосередньо на особливостях самого зображення. Одні з таких методів засновані на знаходженні особливих точок і їх чисельного опису, на які люди навіть не звертають уваги.

У даному розділі будуть розглянуті найбільш популярні детектори особливих точок, принцип їх роботи, основні переваги та недоліки кожного з них.

Грунтуючись тільки на наборі таких даних у цифровому форматі можна з досить високою точністю дозволити комп'ютеру працювати з візуальними образами подібне до чоловіка. Питання ефективності таких алгоритмів ставиться найбільш гостро при роботі на мобільних пристроях - смартфонах, без яких важко уявити образ сучасної людини.

За допомогою мобільних пристроїв люди роблять тисячі фотографій кожен день, нерідко вони виходять поганої якості і виникає необхідність робити повторні

знімки. Згодом це може призвести до засмічення пам'яті через накопичення великої кількості схожих фотографій.

У даній роботі порівнюються кілька сучасних методів пошуку особливих точок і розрахунку їх дескрипторів.

1.2. Детектор Моравеца

Одним з найбільш поширених детекторів особливих точок є детектори кутів на зображенні. Зручність їх використання закладається у тому, що кути на зображенні можна однозначно зіставити, на відміну від ребр.

Метод, запропонований Моравецом [4] є найбільш простим серед детекторів кутів.

Автор пропонує вимірювати зміну яскравості пікселя *(x,y)* за допомогою зміщення квадратного вікна с центром у *(x,y)* на один піксель у кожному з восьми напрямків (вверх, вниз, вліво, вправо, та в чотирьох напрямках по діагоналі). Розмір вікна найчастіше обирається рівним 3x3, 5x5, 9x9 пікселів.

Детектор працює наступним чином:

– для кожного з напрямків

$(u, v) \in \{(1,0), (1,1), (0,1), (-1,0), (-1,1), (-1,-1), (0,-1), (1,-1)\}$;

– розраховується зміна яскравості:

$V_{u,v}(x, y) = \sum_{\forall a,b}\big(I(x+u+a, y+v+b) - I(x+a, y+b)\big)^2$, де I(x,y) – яскравість пікселя (x,y) у початковому зображенні;

– будується карта ймовірності знаходження кутів шляхом розрахунку C(x,y) для кожного пікселя (x,y):

$$\mathrm{C}(\mathrm{x}, \mathrm{y}) = \min\left(V_{u,v}(x, y)\right);$$

– визначається поріг значень T карти ймовірності. Для всіх C(x,y)<T значення C(x,y) обнуляється;

– за допомогою процедури NMS(non-maximal suppression) знаходяться локальні максимуми.

Результатом виконання даного алгоритму є карта ймовірностей. Всі точки, значення на карті для котрих не дорівнює нулю вважаються кутами, тобто особливими точками.

Метод Моравеца володіє наступними недоліками: велика кількість помилок через наявність шуму на зображенні, анізотропія усього в восьми напрямках, відсутність властивості інваріантності до перетворення типу «переворот».

1.3. Детектор Харріса

Метод Харріса заснований на детекторі Моравеца та є його покращенням, так як для нього характерна анізотропія у всіх напрямках.

Харріс і Стефенс запропонували розглянути похідні по безлічі напрямках.

Для даного зображення I розглянемо вікно W (зазвичай його розмір обирається рівним 5x5 пікселів, але може бути іншим в залежності від розміру зображення) с центром у точці (x,y), а також зсув цього вікна на (u,v).

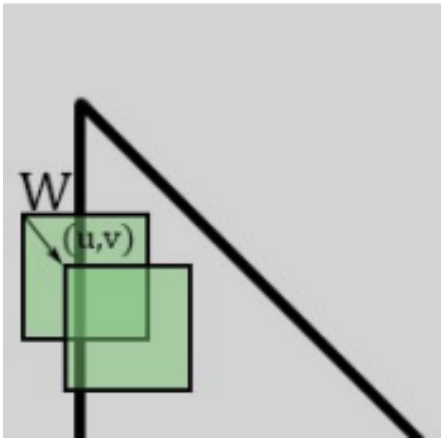


Рисунок 1.3.1 – вікно, що використовується

Тоді зважена сума квадрату різниць між вікном W та вікном

W, зрушеним на *(u,v)* (тобто зміна околиці точки *(x,y)* при зсуві на *(u,v)*) дорівнює:

$$E(u,v) = \sum_{(u,v)\in W} w(x,y)\big(I(x+u,y+v) - I(x,y)\big)^2$$

$$\approx \sum_{(u,v)\in W} w(x,y)\big(I_x(x,y)u - I_y(x,y)v\big)^2$$

, де *w(x,y)* – вагова функція (зазвичай використовується функція Гауса або бінарне вікно).

$$M = \sum_{(u,v)\in W} w(u,v)\begin{bmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{bmatrix}$$

, де M – автокореляційна матриця.

Кут характеризується великими змінами функції *E(x,y)* по безлічі всіх напрямків *(x,y)*, що еквівалентно великим по модулю власним значенням матриці M.

Так як рахувати власні значення затратно з точки зору ресурсів, Харріс та Стефен запропонували міру відгуку:

$R = detM - k(trM)^2 > k$, де k – емпірична константа, $k \in [0.04, 0.06]$.

Таким чином, значення R >=0 для кутових особливих точок. Після цього виконується фільтрація точок по R(тобто ті точки, для котрих R менше визначеного порогу, виключається з розглядання). Далі знаходяться локальні максимуми функції відгуку в околиці заданого радіусу та обираються в якості особливих точок.

Детектор Харріса інваріантний до перетворення типу «поворот», однак чутливий до зміни масштабу та шуму.

1.4. Детектор Shi-Tomasi

Детектор Shi-Tomasi заснований на детекторі Харріса. Різниця закладається в тому, що в детекторі Shi-Tomasi міра відгуку розраховується наступним чином:

$$R = \min(\lambda_1, \lambda_2)$$

Якщо це значення вище певного порогу, то точка вважається кутом і відповідно особливою.

1.5. Детектор FAST

Принцип роботи метода FAST [7] закладається у наступному:

– обирається точка зображення p, для якої буде вирішуватися, являється вона особливою чи ні. Нехай I_p – яскравість точки;

– обирається значення порога t;

– розглядається окружність з 16 пікселів навколо обраної точки;

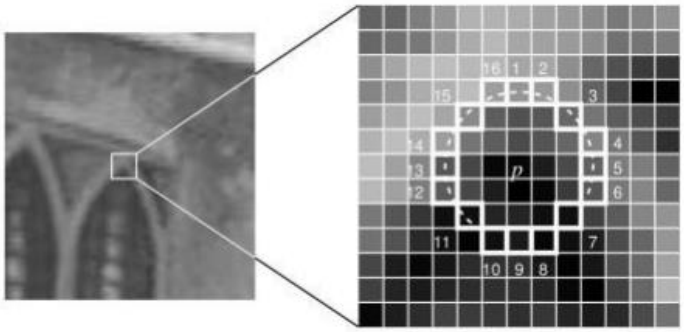


Рисунок 1.5.1 – Робоча околиця пікселя при використанні FAST детектора

– точка p вважається кутом, якщо серед 16 пікселів окружності існує n пікселів, кожен з яких яскравіший за $I_p + t$, або n пікселів, кожен з яких темніший за $I_p - t$(такі пікселі відмічені пунктиром на рисунку 2.5.1). Значення n зазвичай обирається рівним 12. Експерименти показали, що найменше значення n, при якому точки починають стабільно знаходитись, рівно 9;

– висока швидкість роботи даного алгоритму обумовлена тим, що спочатку перевіряється інтенсивність лише чотирьох точок з окружності, під номерами 1,5,9,13. Якщо хоча б для трьох з них виконується умова $I_{pi} > I_p + t$ або $I_{pi} < I_p - t$, i=1,…,4, тоді виконується перевірка всіх інших 12-ти пікселів. В іншому випадку обирається наступна точка та алгоритм повторюється для неї.

Даний алгоритм володіє рядом недоліків, наприклад його ефективність залежить від порядку обробки зображення та розподілу пікселів, поблизу деякої околиці може виявитися декілька особливих точок. В [8] було запропоновано використання машинного навчання для виправлення цих недоліків, отриманий алгоритм отримав назву FAST-ER:

– обирається набір зображень для навчання;

– алгоритм FAST застосовується для пошуку особливих точок на кожному з зображень;

– для кожної особливої точки зберігається 16 пікселів навколо неї у вигляді вектору. Після виконання даного кроку для кожного зображення, всі отримані вектори об’єднуються в один вектор P;

– кожен піксель з 16-ти може знаходитись в одному з трьох станів:

$$S_{p\to x} = \begin{cases} d, & I_{p\to x} \le I_{p-t} \quad (\text{темніший}) \\ s, & I_p - t < I_{p\to x} < I_p + t \ (\text{схожий}) \\ b, & I_p + t \le I_{p\to x} \quad (\text{світліший}) \end{cases}$$

– залежно від станів, наведених вище, P поділяється на три підмножини, P_d, P_s, P_b;

– у розгляді вводиться булева змінна K_p, яка рівна 1, якщо p – кут або 0 в іншому випадку;

– використовується алгоритм ID3(класифікатор дерев рішень) [9]. Він обирає x, який надає найбільше інформації про те, являється піксель кутом чи ні, розрахованої за допомогою ентропії K_p;

– ця процедура повторюється рекурсивно, до тих пір поки ентропія K_p не дорівнюватиме 0;

– створене таким чином дерево рішень використовується для більш швидкого детектування в інших зображеннях.

Для того щоб уникнути знаходження декількох особливих точок поблизу деякої околиці, використовується метод non-maximal-suppression, тобто пошук локальних максимумів. Розраховується значення функції V для кожної з найдених особливих точок. V – це сума різниць інтенсивностей p і кожного з 16-ти навколишніх пікселів.

Розглядається дві сусідні точки та порівнюються їх значення V. Точка з меншим значенням V виключається з розгляду.

1.6. Детектор SIFT

Вище були розглянуті детектори кутів, інваріантні до перетворень типу «поворот» (окрім детектора Моравеца). Це логічно, так як кути залишаються кутами при поворотах, однак, при зміні масштабу, кут може бути не розпізнаним алгоритмом (рисунок 2.6.1).

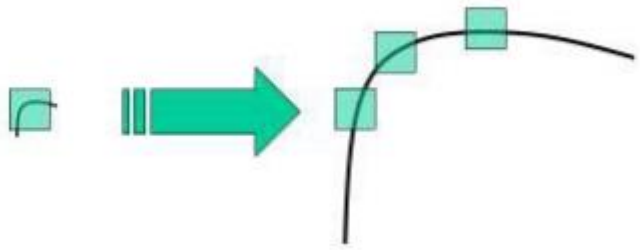


Рисунок 1.6.1 – Кут при зміненні масштабу

В [9] пропонується алгоритм, названий SIFT для вирішення цієї проблеми. Далі буде розглянутий принцип роботи тієї частини алгоритму, який відповідає за пошук особливих точок. Основною ідеєю детектування точок в даному методі є побудова піраміди Гаусіанов та піраміди різниць Гаусіанов.

Гаусіаном називається зображення:

$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y)$, де L – значення гаусіана в точці (x,y), σ – радіус розмиття, G – гаусове ядро, I – значення інтенсивності точки (x,y) в вихідному зображенні.

Різницею гаусіанів називають зображення, отримане шляхом попіксельного віднімання двох гаусіанів з різними значеннями радіусу розмиття. Процес пошуку різниць гаусіанів повторюється для різних октав зображення в піраміді гаусіанів (рисунок 2.6.2).

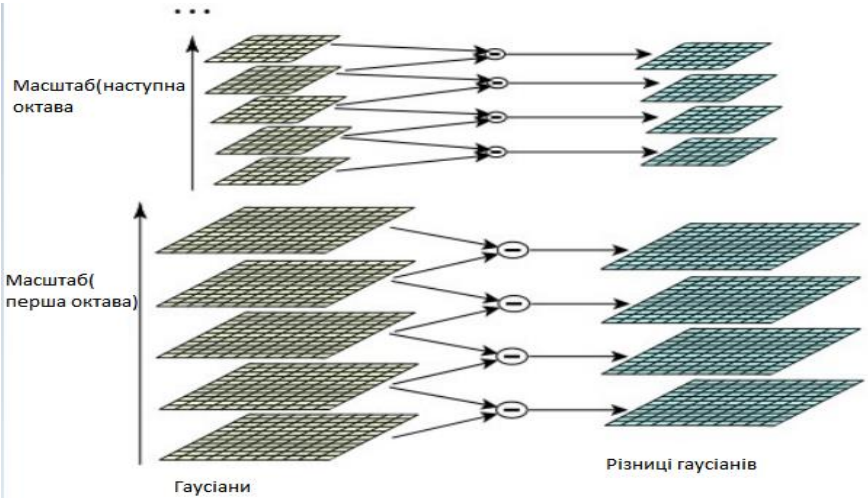


Рисунок 1.6.2 – Піраміди гаусіанів і різниць гаусіанів

Як тільки різниці гаусіанів знайдені, виконується пошук локальних екстремумів. Наприклад, точка зображення порівнюється з вісьмома сусідніми точками та с дев’ятьма точками з наступної різниці гаусіанів та с дев’ятьма точками попередньої різниці гаусіанів (рисунок 2.6.3). Якщо точка є локальним екстремумом, то вона вважається потенціальною особливою точкою.

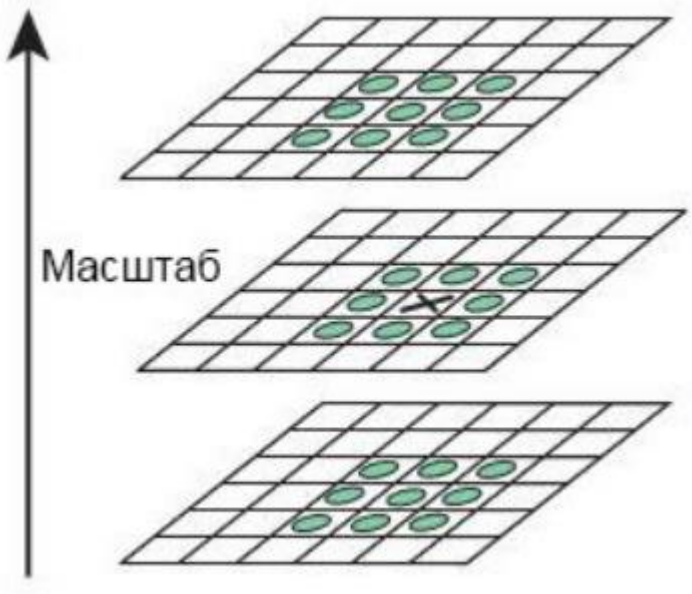


Рисунок 1.6.3 – Пошук локальних екстремумів

На основі експериментів були отримані оптимальні значення: кількість октав = 4, кількість різних значень масштабу в октаві = 5, початкове значення σ = 1.6.

Далі відбувається локалізація особливих точок. На цьому етапі перевіряється, підходить точка екстремуму на роль особливої чи ні. Використовується розкладання простору, що масштабується в ряд Тейлора, щоб отримати більш точне положення екстремуму, та якщо його інтенсивність менша за порогове значення (автором метода запропоновано значення 0.03), то точка не є особливою. Після цього виконується перевірка, чи лежить точка на границі якого-небудь об’єкту. Такі точки мають великий вигин (одна з компонент другої похідної) вдовж границі та малий в перпендикулярному напрямку. Цей великий вигин визначається матрицею Гессе, розміром 2x2.

Алгоритм SIFT володіє рядом переваг, таких як інваріантність до поворотів, зсуву, а також часткова інваріантність до зміни масштабу, яскравості та положення камери. З недоліків можна визначити невисоку швидкість роботи алгоритму.

1.7. Детектор SURF

Метод SURF так як і SIFT належить до ряду методі, виконуючих детектування особливих точок і побудови дескрипторів. Він заснований на тих самих принципах що й SIFT, однак є різниці, завдяки яким SURF працює швидше.

У даному методі використовується фільтри квадратної форми для апроксимації розмиття Гауса. Фільтрація зображення за допомогою квадрата виконується швидше, якщо використовувати інтегральне уявлення зображення, яке виглядає наступним чином:

$$S(x,y) = \sum_{i=0}^{x}\sum_{j=0}^{y} I(i,j)$$

SURF використовує детектор крапель (blob detector) заснований на використанні матриці Гессе. Визначник матриці Гессе використовується як міра локальних змін навколо точки, і, вибираються ті точки, для яких цей визначник максимальний. Також, SURF використовує визначник матриці Гессе для вибору масштабу. Матриця Гессе виглядає наступним чином:

$H(p,\sigma) = \begin{pmatrix} L_{xx}(p,\sigma) & L_{xy}(p,\sigma) \\ L_{xy}(p,\sigma) & L_{yy}(p,\sigma) \end{pmatrix}$, де p - точка зображення з координатами (x, y), σ - масштаб фільтра, $L_{xx}(p,\sigma)$ – згортки апроксимації другої похідної Гауссова ядра з вихідним зображенням.

1.8. Детектор BRISK

Метою створення даного методу було досягнення інваріантності до зміни масштабу, при цьому зберігши високу швидкість роботи.

Автори методу створили свій метод на основі детектора AGAST, який, по суті, є розширенням методу FAST. Однак, для того, щоб домогтися інваріантності до зміни масштабу, знаходження максимумів відбувається не тільки на оригінальному документі, але і в масштабованому просторі. У методі BRISK масштабується простір складається з n октав c_i і n внутрішніх октав $d_i, i = \{0,1,\dots,n-1\}$, n зазвичай вибирається рівним 4. Октави складаються шляхом зменшення масштабу вихідного зображення в два рази. Кожна внутрішня октава d_i розташовується між октавами c_i і c_{i+1}. Перша внутрішня октава виходить шляхом зменшення масштабу вихідного

зображення в 1.5 рази, а кожна наступна - шляхом зменшення масштабу попередньої внутрішньої октави в два рази.

Метод FAST застосовується до кожної з октав і кожної з внутрішніх октав окремо з однаковим граничним значенням, з метою знайти потеціально особливі області. Потім серед точок, що містяться в цих областях проводиться пошук локальних максимумів у всьому масштабованому просторі.

1.9. Постановка задачі

Завдання даної роботи полягає в порівнянні методів детектування особливих точок. Процес її виконання можна розділити на наступні етапи:

– вивчення принципів функціонування методів;

– розробка інструментарію, що дозволяє виявляти особливі точки кожним з розглянутих детекторів;

– аналіз результатів роботи методів на наборі тестових зображень;

– отримання висновків про можливі сценарії використання кожного з них, заснованих на виявлених під час аналізу переваги і недоліки.

2 ПРОГРАМНА РЕАЛІЗАЦІЯ

Автором даної роботи було розроблено програмне забезпечення, яке реалізує пошук особливих точок одним з методів (Детектор Харріса, ShiTomasi, FAST, SIFT, SURF і BRISK), підраховують кількість знайдених точок і час, витрачений кожним з алгоритмів на пошук. Детектор Моравеца не був реалізований в програмі, через його низьку ефективності і рідкості застосування.

2.1. Детектор Харріса

Наведемо програмну реалізацію пошуку особливих точок методом Харріса:

```
import cv2
import numpy as np

number=1
while(number<=9):
    filename = 'photo/original/'+str(number)+'.jpg'
    img = cv2.imread(filename)
    gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)

    gray = np.float32(gray)
    dst = cv2.cornerHarris(gray,2,3,0.04)

    #result is dilated for marking the corners, not
important
```

```
        dst = cv2.dilate(dst,None)

    # Threshold for an optimal value, it may vary depending
on the image.
        img[dst>0.01*dst.max()]=[0,0,255]

        cv2.imshow('dst',img)

cv2.imwrite('photo/harris/original'+str(number)+'.jpg')
```

Даний метод зчитує фотографію, виділяє особливі точки та зберігає в іншому файлі.

2.2. Детектор Shi-Tomasi

Наведемо програмну реалізацію пошуку особливих точок методом Shi-Tomasi:

```
    def goodFeaturesToTrack_Demo(val):
        maxCorners = max(val, 1)
        # Parameters for Shi-Tomasi algorithm
        qualityLevel = 0.01
        minDistance = 10
        blockSize = 3
        gradientSize = 3
```

```
        useHarrisDetector = False
        k = 0.04
        # Copy the source image
        copy = np.copy(src)
        # Apply corner detection
        corners = cv.goodFeaturesToTrack(src_gray,
maxCorners, qualityLevel, minDistance, None, \
            blockSize=blockSize, gradientSize=gradientSize,
useHarrisDetector=useHarrisDetector, k=k)
        # Draw corners detected
        print('** Number of corners detected:',
corners.shape[0])
        radius = 4
        for i in range(corners.shape[0]):
            cv.circle(copy, (corners[i,0,0],
corners[i,0,1]), radius, (rng.randint(0,256),
rng.randint(0,256), rng.randint(0,256)), cv.FILLED)
        # Show what you got
        cv.namedWindow(source_window)
        cv.imshow(source_window, copy)
        status = cv.imwrite('photo/shitomasi/zoom/9.jpg',
copy)
        print(status)
```

Даний метод зчитує фотографію, виділяє особливі точки та зберігає в іншому файлі.

2.3. Детектор FAST

Наведемо програмну реалізацію пошуку особливих точок методом FAST:

```
    def detect(image, threshold=40):
        """
        corners = fast.detect(image, threshold)
performs the detection
        on the image and returns the corners as a list
of (x,y) tuples
        where x is the column index, and y is the row
index
        Nonmaximal suppression is implemented by
default.
        This function does not search the entire frame
for corners. It only searches a portion
        in the middle in order to speed up the process.
        ***Parameters:
            image is a numpy array of intensity values.
NOTE: Image must be grayscale
            threshold is an int used to filter out non-
corners.
        """
        # Initialization
        image = rgb2gray(image)
        corners = []
        rows, cols = shape(image)
        startSearchRow = int(0.25 * rows)
        endSearchRow = int(0.75 * rows)  # search the
middle square of the frame
        startSearchCol = int(0.25 * cols)
        endSearchCol = int(0.75 * cols)
```

```
        image = medianBlur(image, startSearchRow,
endSearchRow, startSearchCol, endSearchCol)

        # Begin searching through search area
        for row in range(startSearchRow, endSearchRow):
            for col in range(startSearchCol,
endSearchCol):
                ROI = circle(row, col)
                if is_corner(image, row, col, ROI,
threshold):
                    corners.append((col, row))
        suppress(image, corners, ROI)
    return corners;
```

Даний метод приймає зображення та повертає масив знайдених кутів.

2.4. Детектор SIFT

Наведемо програмну реалізацію пошуку особливих точок методом SIFT:

```
import numpy as np
import cv2 as cv
number=1
while(number<=1):
    img = cv.imread('photo/zoom/'+str(number)+'.jpg')
    #gray= cv.cvtColor(img,cv.COLOR_BGR2GRAY)
    sift = cv.xfeatures2d.SIFT_create()
    kp = sift.detect(img,None)
    img=cv.drawKeypoints(img,kp,img)

cv.imwrite('photo/sift/zoom/'+str(number)+'.jpg',img)
number=number+1
```

Даний метод зчитує зображення, відмічає особливі точки та зберігає у новому файлі.

2.5. Детектор SURF

Наведемо програмну реалізацію пошуку особливих точок методом SURF:

```
import cv2
import numpy as np
number=2
while(number<=9):
    img = cv2.imread("photo/zoom/"+str(number)+".png")
    sift = cv2.xfeatures2d.SIFT_create()
    surf = cv2.xfeatures2d.SURF_create()
    orb = cv2.ORB_create(nfeatures=1500)
    keypoints_sift,             descriptors             =
sift.detectAndCompute(img, None)
    keypoints_surf,             descriptors             =
surf.detectAndCompute(img, None)
    keypoints_orb,             descriptors             =
orb.detectAndCompute(img, None)
    img = cv2.drawKeypoints(img, keypoints_orb, None)
    status                                              =
cv2.imwrite('photo/surf/zoom/'+str(number)+'.jpg', img)
    print(status)
    number= number+1
```

Даний метод зчитує зображення, відмічає особливі точки та зберігає у новому файлі.

2.6. Детектор BRISK

Наведемо програмну реалізацію пошуку особливих точок методом BRISK:

```
import cv2
import numpy as np
```

```
number=2
while(number<=9):
    img = cv2.imread('photo/zoom/'+str(number)+'.png')
    img_brisk = img.copy()
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    brisk = cv2.BRISK_create()

    (kps5, descs5) = brisk.detectAndCompute(img, None)
    #cv2.DrawMatchesFlags_DRAW_RICH_KEYPOINTS
    cv2.drawKeypoints(img, kps5, img_brisk)

    cv2.imwrite('photo/brisk/zoom/'+str(number)+'.jpg',
img_brisk)
    number=number+1
```

Даний метод зчитує зображення, відмічає особливі точки та зберігає у файлі.

3 ПРАКТИЧНЕ ЗАСТОСУВАННЯ ДЕТЕКТОРІВ

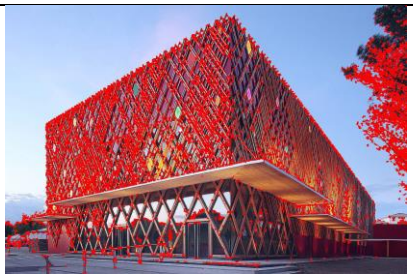
3.1. Детектор Харріса

В даному підрозділі будуть розглянуті результати роботи детектора Харріса на фотографіях будівель і трьох їх варіацій (повернене, масштабувати і затемнене).

Таблиця 3.1.1 – Результат роботи детектора Харріса з оригінальними зображеннями

	Оригінальне зображення	Оброблене зображення
1		
2		
3		
4		

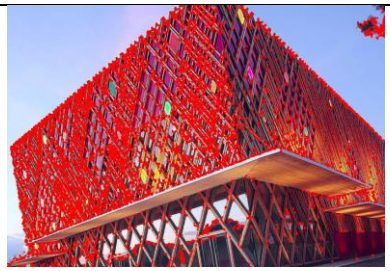
Продовження таблиці 3.1.1

	Оригінальне зображення	Оброблене зображення
5		
6		
7		
8		
9		

Зображеннями для тесту були обрані фотографії будівель. Більшість кутів, легко знайдених людиною, не були знайдені алгоритмом, в той час як в окремих

частинах зображення було виявлено дуже багато особливих точок близько один до одного.

Таблиця 3.1.2 – Результат роботи детектора Харріса з повернутими зображеннями

	Оригінальне зображення	Оброблене зображення
1		
2		
3		
4		
5		

Продовження таблиці 3.1.2

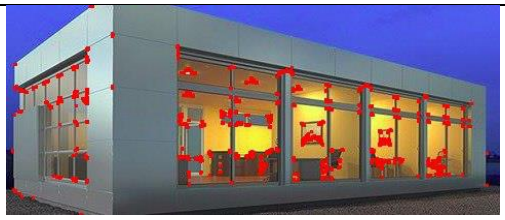
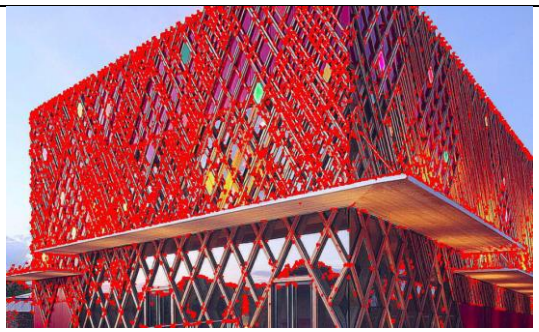
	Оригінальне зображення	Оброблене зображення
6		
7		
8		
9		

Далі тест був проведений на тих самих зображеннях будівель, але повернутих приблизно на 10 градусів (рисунок 3.2). Незважаючи на деякі незначні відмінності в результаті, детектор Харріса виявив особливі точки там же де і до цього, в приблизно такій же кількості, що говорить про інваріантності даного методу до поворотів.

Таблиця 3.1.3 - Результат роботи детектора Харріса зі збільшеними зображеннями

	Оригінальне зображення	Оброблене зображення
1		
2		
3		
4		
5		

Продовження таблиці 3.1.3

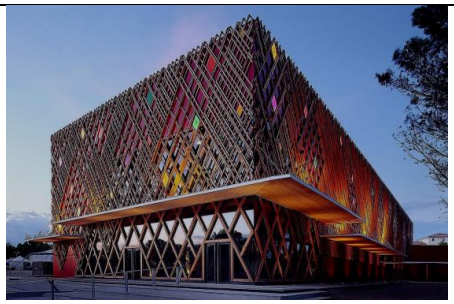
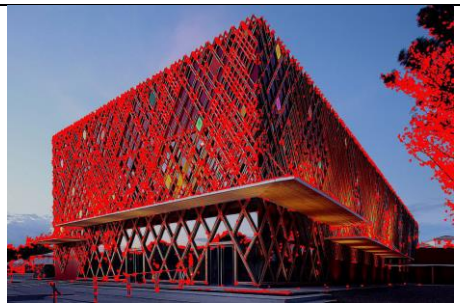
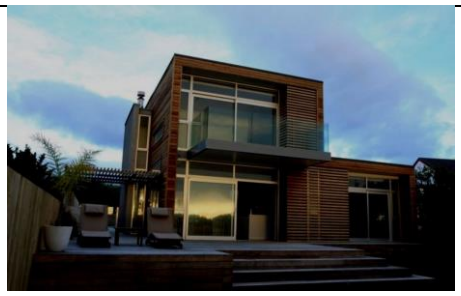
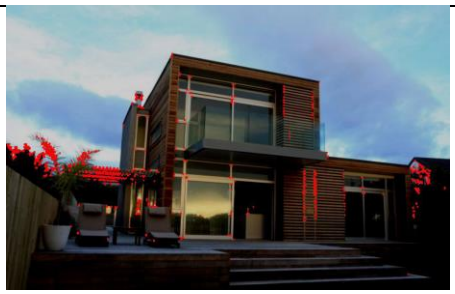
	Оригінальне зображення	Оброблене зображення
6		
7		
8		
9		

Алгоритм виявив безліч точок, які не були виявлені в оригінальному документі, а також не виявив деякі точки, які на оригінальному документі були знайдені, хоча таких точок виявилося небагато.

Таблиця 3.1.4 - Результат роботи детектора Харріса на затемнених зображеннях.

	Оригінальне зображення	Оброблене зображення
1		
2		
3		
4		
5		
6		

Продовження таблиці 3.1.4

	Оригінальне зображення	Оброблене зображення
7		
8		
9		

У порівнянні з оригінальним зображенням, було знайдено значно менше точок, але в той же час зменшилася кількість дублікатів і були виявлені деякі з кутів, що не були виявлені до цього. З відмінностей в результаті виходить, що детектор Харріса лише частково інваріантний до змін яскравості зображення.

З отриманих результатів можна зробити висновок, що метод Харріса добре працює для зображень з яскраво вираженими кутами, добре виділяються на фоні, інваріантний до поворотів, володіє досить високою швидкістю роботи, проте погано знаходить особливі точки на зображеннях різного масштабу і відмінності в яскравості.

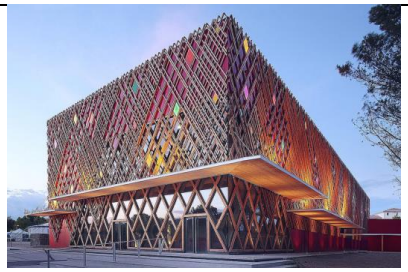
3.2 . Детектор Shi-Tomasi

Для оцінки результатів роботи детектора Shi-Tomasi і всіх інших детекторів будуть використовуватися ті ж зображення, що і в попередньому розділі. Це дозволить найбільш точно порівняти результати роботи методів.

Таблиця 3.2.1 - Результат роботи детектора Shi-Tomasi на оригінальних зображеннях.

	Оригінальне зображення	Оброблене зображення
1		
2		
3		
4		

Продовження таблиці 3.2.1

	Оригінальне зображення	Оброблене зображення
5		
6		
7		
8		
9		

Результат пошуку особливих точок на фотографії будівлі дуже схожий на результат детектора Харріса. В цілому, алгоритмом Shi-Tomasi знайдені і пропущені

ті ж самі кути, але перевагою є відсутність дублікатів точок в околицях кутів. Було знайдено.

Таблиця 3.2.2 - Результат роботи детектора Shi-Tomasi на повернутих зображеннях.

	Оригінальне зображення	Оброблене зображення
1		
2		
3		
4		
5		

Продовження таблиці 3.2.2

	Оригінальне зображення	Оброблене зображення
6		
7		
8		
9		

Результат пошуку особливих точок на повернених фотографіях так само схожий на результат методу Харріса, за винятком відсутності дублікатів. Знайдено майже ті ж самі кути, що й на оригінальному документі, що говорить про інваріантність методу Shi-Tomasi до поворотів.

Таблиця 3.2.3 - Результат роботи детектора Shi-Tomasi на збільшених зображеннях.

	Оригінальне зображення	Оброблене зображення
1		
2		
3		
4		
5		

Продовження таблиці 3.2.3

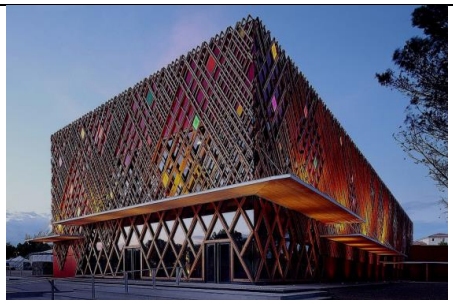
	Оригінальне зображення	Оброблене зображення
6		
7		
8		
9		

На збільшених фотографіях метод Shi-Tomasi показав кращий результат, в порівнянні з детектором Харріса. Особливих точок знайдено менше, але більшість з них дійсно знаходяться на кутах об’єктів на зображенні. Як і в попередніх випадках, дублікати відсутні.

Таблиця 3.2.4 - Результат роботи детектора Shi-Tomasi на затемнених зображеннях.

	Оригінальне зображення	Оброблене зображення
1		
2		
3		
4		
5		
6		

Продовження таблиці 3.2.4

	Оригінальне зображення	Оброблене зображення
7		
8		
9		

Результат пошуку набагато краще, ніж на оригінальному документі, було знайдено безліч кутів, які не були знайдені раніше. Це говорить про те, що метод частково інваріантний до зміни яскравості зображення.

Детектор Shi-Tomasi створений на основі детектора Харріса і ця подібність помітно при практичному порівнянні їх результатів. Він теж є не інваріантним до зміни масштабу, частково інваріантний до зміни яскравості і не залежить від поворотів. В цілому він показує кращі результати, хоча і робить помилки, невластиві методу Харріса, а саме, знаходить особливі точки на діагональних ребрах.

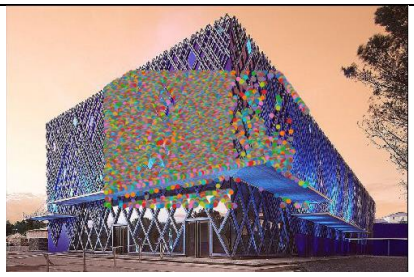
3.3. Детектор FAST

В даному підрозділі буде розглянуто результат роботи алгоритму FAST на наборі тестових зображень. Детектор FAST є детектором кутів, так само як і детектор Харріса і Shi-Tomasi, тому знайдені їм особливі точки будуть порівняні з знайденими точками попередніх методів.

Таблиця 3.3.1 - Результат роботи детектора FAST на оригінальних зображеннях.

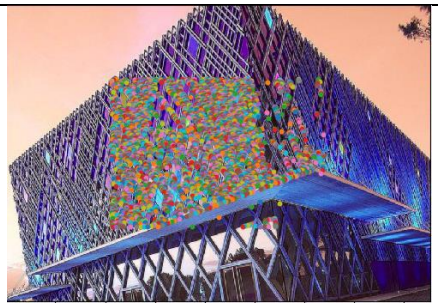
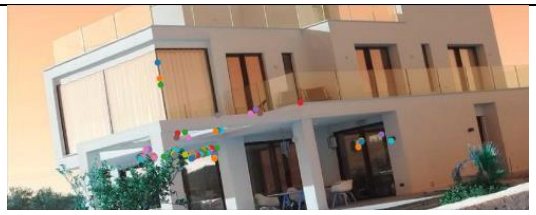
	Оригінальне зображення	Оброблене зображення
1		
2		
3		
4		

Продовження таблиці 3.3.1

	Оригінальне зображення	Оброблене зображення
5		
6		
7		
8		
9		

Метод показав хороший результат на даних зображеннях, виявивши безліч кутів, мала кількість зайвих крапок, які не належать кутам і середня кількість дублікатів. Швидкість роботи знову виявилася вкрай високою.

Таблиця 3.3.2 - Результат роботи детектора FAST на повернутих зображеннях.

	Оригінальне зображення	Оброблене зображення
1		
2		
3		
4		
5		

Продовження таблиці 3.3.2

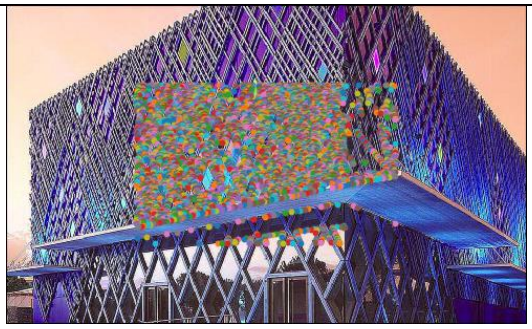
	Оригінальне зображення	Оброблене зображення
6		
7		
8		
9		

Майже всі точки, знайдені на оригінальному документі, були знайдені і на повернутому. Їх кількість також приблизно однакова. Це говорить про інваріантність алгоритму до перетворення типу «поворот».

Таблиця 3.3.3 - Результат роботи детектора FAST на збільшених зображеннях.

	Оригінальне зображення	Оброблене зображення
1		
2		
3		
4		
5		

Продовження таблиці 3.3.3

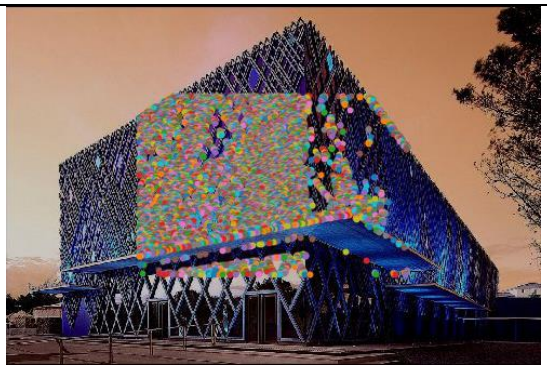
	Оригінальне зображення	Оброблене зображення
6		
7		
8		
9		

Не були знайдені деякі з точок, які були знайдені до цього, а також з'явилися нові особливі точки там де їх не було раніше. Такий результат говорить про те, що метод FAST не інваріантний до зміни масштабу.

Таблиця 3.3.4 - Результат роботи детектора FAST на затемнених зображеннях.

	Оригінальне зображення	Оброблене зображення
1		
2		
3		
4		
5		
6		

Продовження таблиці 3.3.4

	Оригінальне зображення	Оброблене зображення
7		
8		
9		

Детектор FAST показав хороші результати пошуку особливих точок на затемненій фотографії. Були виявлені майже ті ж самі кути, що і на оригінальному документі, за невеликими винятками. З цього можна зробити висновок, що детектор FAST інваріантний до зміни яскравості.

3.4. Детектор SIFT

В даному підрозділі будуть розглянуті результати роботи алгоритму SIFT на тестових зображеннях. Порівняння буде проводитися на фотографіях будівель і їх модифікованих версіях (поверненою, збільшеною і затемненій).

Таблиця 3.4.1 - Результат роботи детектора SIFT на оригінальних зображеннях.

	Оригінальне зображення	Оброблене зображення
1		
2		
3		
4		

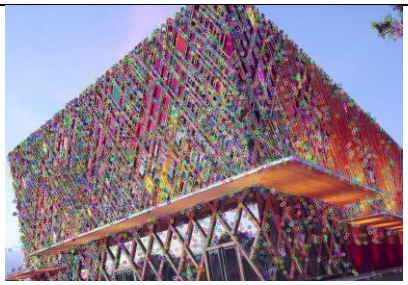
Продовження таблиці 3.4.1

	Оригінальне зображення	Оброблене зображення
5		
6		
7		
8		
9		

Знайдені точки в основному розташовані в околиці будівлі, що говорить про те, що метод добре виявив особливості зображення і об'єктів на ньому. Жодної точки

не було знайдено в тій частині фотографії, де зображено небо, лише хмари, тобто метод ігнорує монотонні частини зображення. Однак були допущені помилки, такі як наявність особливих точок на текстурі самої будівлі і поверхні підлоги. Такі точки згодом не підійдуть для порівняння зображень, так як текстура повторюється і точки можна буде однозначно співвіднести, тобто було порушено властивість унікальності, описане на початку даної роботи.

Таблиця 3.4.2 - Результат роботи детектора SIFT на повернутих зображеннях.

	Оригінальне зображення	Оброблене зображення
1		
2		
3		
4		

Продовження таблиці 3.4.2

	Оригінальне зображення	Оброблене зображення
5		
6		
7		
8		
9		

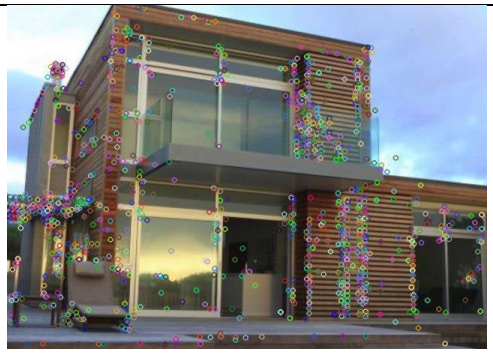
На повернутих фотографіях детектор SIFT допустив ті ж помилки, що і на вихідних, виявивши особливі точки на повторюваних структурах будівлі і підлозі,

але вже в більшій кількості. І хоча такі точки можна виключити з розгляду на стадії побудови дескрипторів, їх велика кількість позначається на швидкості роботи методу.

Таблиця 3.4.3 - Результат роботи детектора SIFT на збільшених зображеннях.

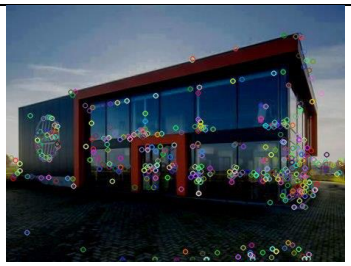
	Оригінальне зображення	Оброблене зображення
1		
2		
3		
4		
5		

Продовження таблиці 3.4.3

	Оригінальне зображення	Оброблене зображення
6		
7		
8		
9		

Кількість точок знайдених на текстурі будівлі при збільшенні зображення стало ще більше, що помітно збільшило час роботи алгоритму. Незважаючи на теоретичну інваріантність методу до змін масштабу, практичний результат показує інше. Варто зауважити, що в монотонних частинах зображення особливі точки як і раніше не знаходяться.

Таблиця 3.4.4 - Результат роботи детектора SIFT на затемнених зображеннях.

	Оригінальне зображення	Оброблене зображення
1		
2		
3		
4		
5		
6		

Продовження таблиці 3.4.4

	Оригінальне зображення	Оброблене зображення
7		
8		
9		

Детектор SIFT добре справляється з пошуком особливих точок на затемненому зображенні. Більшість точок, знайдених на оригінальному документі було знайдено і на затемненому. Деякі точки не були знайдені, хоча таких небагато. Метод SIFT частково інваріантний до змін яскравості.

3.5. Детектор SURF

В даному підрозділі будуть розглянуті результати пошуку особливих точок за допомогою методу SURF. Так само як і SIFT, він не відноситься до детекторам кутів.

Таблиця 3.5.1 – Результат роботи детектора SURF на оригінальних зображеннях.

	Оригінальне зображення	Оброблене зображення
1		
2		
3		
4		

Продовження таблиці 3.5.1

	Оригінальне зображення	Оброблене зображення
5		
6		
7		
8		
9		

На фотографії будівлі метод SURF показав результат, схожий на результат методу SIFT, точки знайдені в тій частині де розташовується будівля, не виявлені на монотонних частинах зображення, що говорить про виконання властивості глобальної унікальності. Точок, знайдених на повторюваних структурах мало, з чого можна зробити висновок, що властивість локальної різниці теж виконується на досить хорошому рівні. Варто відзначити, що метод SURF працює помітно швидше ніж SIFT.

Таблиця 3.5.2 - Результат роботи детектора SURF на повернутих зображеннях.

	Оригінальне зображення	Оброблене зображення
1		
2		
3		
4		

Продовження таблиці 3.5.2

	Оригінальне зображення	Оброблене зображення
5		
6		
7		
8		
9		

Результат пошуку на повернутому зображенні майже не відрізняється від результатів пошуку на вихідному. Алгоритм як і раніше знайшов точки, що володіють властивістю глобальної та локальної різниці, приблизно в тих же місцях, що і до цього. Це говорить про інваріантності методу до перетворення типу «поворот».

Таблиця 3.5.3 - Результат роботи детектора SURF на збільшених зображеннях.

	Оригінальне зображення	Оброблене зображення
1		
2		
3		
4		
5		

Продовження таблиці 3.4.3

	Оригінальне зображення	Оброблене зображення
6		
7		
8		
9		

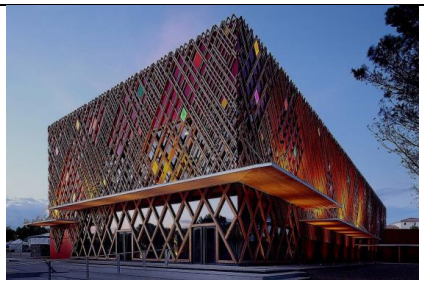
На збільшеної фотографії будівлі детектор SURF показав кращі результати, ніж детектор SIFT. Хоч і було виявлено деяку кількість точок, що не володіють властивістю глобальної унікальності (на повторюваних структурах), їх помітно менше, ніж у попереднього алгоритму. Більшість знайдених точок відповідають

точкам, знайденим на оригінальному документі. Можна зробити висновок, що теоретична інваріантність методу SURF підтверджується на практиці.

Таблиця 3.5.4 - Результат роботи детектора SIFT на затемнених зображеннях.

	Оригінальне зображення	Оброблене зображення
1		
2		
3		
4		
5		
6		

Продовження таблиці 3.5.4

	Оригінальне зображення	Оброблене зображення
7		
8		
9		

Детектор SURF як і SIFT виявився майже незалежним від змін яскравості. На затемненому зображенні особливі точки були знайдені в тих же місцях, що і на вихідному. Точок не виявлено на монотонних і повторюваних частинах зображення. Результат говорить про інваріантності даного методу до змін яскравості.

3.6. Детектор BRISK

Для детектора BRISK будуть також проаналізовані результати роботи на фотографії будівлі і її модифікаціях.

Таблиця 3.6.1 – Результат роботи детектора BRISK на оригінальних зображеннях.

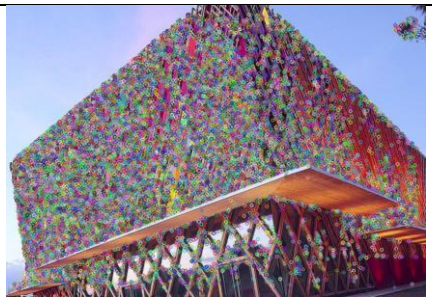
	Оригінальне зображення	Оброблене зображення
1		
2		
3		
4		

Продовження таблиці 3.6.1

	Оригінальне зображення	Оброблене зображення
5		
6		
7		
8		
9		

На фотографії будівлі детектор BRISK виявив бульше кількість точок, ніж SIFT і SURF, всі вони зосереджені близько частин зображення, що виділяються. Це говорить про те, що метод добре виявляє особливості та об'єкти. Швидкість роботи алгоритму виявилася вищою, ніж SIFT, але менше ніж всі інші з раніше розглянутих методів.

Таблиця 3.6.2 - Результат роботи детектора BRISK на повернутих зображеннях.

	Оригінальне зображення	Оброблене зображення
1		
2		
3		
4		

Продовження таблиці 3.6.2

	Оригінальне зображення	Оброблене зображення
1		
2		
3		
4		
5		

За невеликими винятками, на повернутому зображенні були виявлені ті ж самі точки, що на початковому і в схожій кількості. Це говорить про інваріантності методу до поворотів.

Таблиця 3.6.3 - Результат роботи детектора SURF на збільшених зображеннях.

	Оригінальне зображення	Оброблене зображення
1		
2		
3		
4		
5		

Продовження таблиці 3.6.3

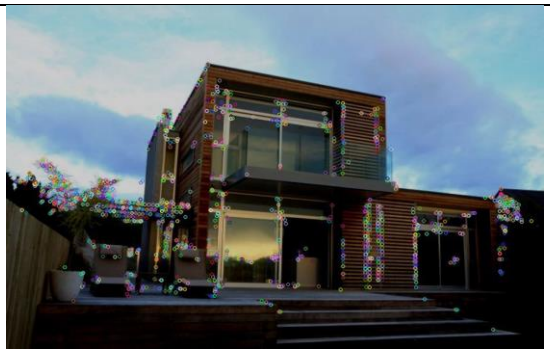
	Оригінальне зображення	Оброблене зображення
6		
7		
8		
9		

На збільшеної фотографії алгоритм допустив невелику кількість помилок, знайшовши нові точки і не виявивши деякі зі знайдених раніше. Однак таких помилок трохи і в цілому результат для збільшеної фотографії схожий на результат для вихідної. Отже, метод BRISK частково інваріантний до змін масштабу.

Таблиця 3.6.4 - Результат роботи детектора SIFT на затемнених зображеннях.

	Оригінальне зображення	Оброблене зображення
1		
2		
3		
4		
5		
6		

Продовження таблиці 3.6.4

	Оригінальне зображення	Оброблене зображення
7		
8		
9		

Можна зробити висновок, що метод BRISK інваріантний до змін яскравості. Було знайдено приблизно така ж кількість точок, що і для вихідного зображення, і розташовуються вони в тих же місцях. Нових зайвих точок майже не з'явилося, а знайдені до цього були знайдені і зараз.

3.7. Порівняння результатів роботи методів

Автором роботи були проведені додаткові тести для кожного методу для зображень різного розміру з метою порівняння швидкості роботи детекторів.

Таблиця 3.7.1 – Результати тестів

	800x600	1280x960	1920x1440	4000x3000
Harris	0.385	0.982	2.127	10.344
Shi-Tomasi	2.449	9.223	17.562	75.422
FAST	0.012	0.034	0.071	0.237
SIFT	4.399	10.109	24.506	94.746
SURF	1.511	2.975	6.168	28.529
BRISK	2.633	2.770	3.023	4.630

З таблиці 3.7.1 можна зробити висновок, що час пошуку особливих точок усіма методами пропорційний кількості пікселів на зображенні. Винятком є метод BRISK, час роботи якого не так сильно залежить від розміру. Час роботи для зображення з 12000000 (4000x3000) пікселів всього в 1.75 рази вище, ніж для зображення з 480000 (800x600) пікселів.

В наступній таблиці представлено порівняння алгоритмів за такими параметрами: інваріантність до змін масштабу, інваріантність до поворотів, незалежність від змін яскравості, швидкість роботи для зображень малого та великого розміру, а також загальну якість пошуку точок за шкалою від 1 до 5, де 1 - погано, 5 - відмінно.

Таблиця 3.7.2 – Порівняння алгоритмів пошуку особливих точок

	Інварінантність			Швидкість роботи		Якість знайдених точок
	Масштаб	Поворот	Зміна яскравості	Малий розмір	Великий розмір	
Harris	2	5	3	4	4	3
Shi-Tomasi	3	5	3	3	2	4
FAST	3	5	4	5	5	3
SIFT	4	3	5	1	1	4
SURF	5	4	4	4	3	5
BRISK	4	4	5	3	5	5

Отже можна зробити висновок, що BRISK найкраще підходить для зображень великого розміру та досить гарно працює з різними перетвореннями зображення. Чого не можна сказати по детектор Харріса, який найгірше працює з перетвореннями, а також має найгіршу якість знайдених точок.

ВИСНОВКИ

На основі отриманих результатів, автор роботи може заявити, що серед розглянутих в роботі алгоритмів немає єдиного, котрий задовільнить будь-яке завдання, в якому використовується пошук особливих точок на зображенні. Кожен з розглянутих методів має свої переваги та недоліками і може бути найбільш відповідним для вирішення будь-якої задачі.

Головними перевагами методу Харріса є інваріантність до повороту, а також досить висока швидкість роботи. Він найкраще підійде для задач, при рішенні яких використовується пошук особливих точок на зображеннях з яскраво-вираженими кутами. Він не підійде для задач, в яких проводиться порівняння зображень різного масштабу і з великими відмінностями в яскравості. Також варто звернути увагу на детектор Харріса, якщо важлива швидкість роботи методу.

Прикладом завдань, в яких детектор Харріса покаже гарні результати можуть бути завдання, в яких використовується пошук простих фігур на зображеннях, створених за допомогою засобів комп'ютерної графіки, так як в такому випадку всі кути цих фігур будуть добре помітні для алгоритму.

Детектор Shi-Tomasi дуже схожий за результатами на детектор Харріса, тому може використовуватися в задачах схожого виду. Однак варто очікувати меншу швидкість роботи. У порівнянні з методом Харріса він володіє більш високою якістю детектування. Як результат, цей метод стане кращим рішенням, якщо в задачі більш важлива точність, і не так важлива швидвість роботи.

Детектор FAST володіє незрівнянною швидкістю роботи, однак програє попереднім методам в детектуванні. Детектор FAST найкраще підходить для задач, в яких час детектування грає ключову роль, наприклад стеження за об'єктом у відео

або пошуку об'єктів на відео. Його висока швидкість дозволить працювати з відео високої роздільної здатності в реальному часі.

Метод SIFT показав найгірший результат серед досліджених детекторів за багатьма характеристиками, проте володіє гарною якістю детектування і стійкістю до змін яскравості. SIFT виявився найповільнішим з розглянутих детекторів і не підійде для більшості завдань, проте якщо в завданні абсолютно не має значення швидкість роботи, але необхідна висока точність, то SIFT може бути обраний як найбільш відповідний метод.

Метод SURF працює не так швидко як FAST або Harris, але володіє досить високим ступенем стійкості до афінних перетворення, змін яскравості і якістю детектування в цілому. SURF стане найкращим вибором для завдань, в яких критично важлива точність детектування. Прикладами таких завдань може бути побудова стереопари, панорам і пошук об'єктів знятих з різних ракурсів і при різному освітленні.

Останній з розглянутих детекторів, BRISK відрізняється високою швидкістю роботи на зображеннях з високою роздільною здатністю. При якості детектування, порівнянному з якістю детектора SURF і незалежністю від поворотів і зміни масштабу, він підійде для роботи з великими зображеннями, наприклад при пошуку збігів на фотографіях зі спутників, так як для таких фотографій характерна велика роздільна здатність.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Конушин А. Слежение за точечными особенностями сцены // Компьютерная графика и мультимедиа, Вып. №1(5)/2003.

2. Rodehorst V., Koschan A. Comparison and evaluation of feature point detectors, 2006.

3. Построение SIFT дескрипторов и задача сопоставления изображений https://habrahabr.ru/post/106302/

4. Moravec, H. Rover visual obstacle avoidance // In International Joint Conference on Artificial Intelligence, Vancouver, Canada, 1981, pp. 785-790.

5. Harris, C. and Stephens, M. A combined corner and edge detector // In Fourth Alvey Vision Conference, Manchester, UK, 1988, pp. 147-151.

6. Shi T. Good Features to Track, 1994.

7. E. R. a. T. Drummond. Fusing Points and Lines for High Performance Tracking, 2005.

8. R. P. a. T. D. Edward Rosten. Faster and better: a machine learning approach to corner detection, 2008.

9. Єгоров С.В. Оптимизация алгоритмов кластерного анализа в задачах распознавания образов, - Херсон, Вестник ХНТУ. - 2011. - № 2(41). С. 172-173

10. Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool, "Speeded Up Robust Features", ETH Zurich, Katholieke Universiteit Leuven