**ЗМІСТ**

[ВСТУП 3](#_Toc453446531)

[РОЗДІЛ 1. ДІАГНОСТИКА ВАЖКОДОСТУПНИХ СЕРЕДОВИЩ 4](#_Toc453446532)

[1.1 Періоди розвитку ендоскопії 4](#_Toc453446533)

[1.1.1 Ригідний період. 4](#_Toc453446534)

[1.1.2 Напівгнучкий період 6](#_Toc453446535)

[1.1.3 Оптоволоконний період 7](#_Toc453446536)

[1.1.4 Електронний період 7](#_Toc453446537)

[1.2 Застосування в медицині 8](#_Toc453446538)

[1.3 Застосування в інших галузях 9](#_Toc453446539)

[ВИСНОВОК ДО РОЗДІЛУ 1 11](#_Toc453446540)

[РОЗДІЛ 2. РОЗРОБКА ФІЗИЧНОЇ ЧАСТИНИ ПРИСТРОЮ 12](#_Toc453446541)

[2.1 Опис бази приладу 12](#_Toc453446542)

[2.1.1 Екран 14](#_Toc453446543)

[2.1.2 Камера 14](#_Toc453446544)

[2.1.3 Продуктивність 15](#_Toc453446545)

[2.1.4 Час життя батареї 15](#_Toc453446546)

[2.2 Дослідження роботи модуля камери смартфона 15](#_Toc453446547)

[2.3 Розробка адаптера 17](#_Toc453446548)

[2.4 Порівняння з аналогами 19](#_Toc453446549)

[2.4.1 Технічне порівняння 19](#_Toc453446550)

[2.4.2 Програмне порівняння 21](#_Toc453446551)

[ВИСНОВОК ДО РОЗДІЛУ 2 22](#_Toc453446552)

[РОЗДІЛ 3. РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 23](#_Toc453446553)

[3.1 Опис OpenTLD 23](#_Toc453446554)

[3.1.1 Визначення проблеми 23](#_Toc453446555)

[3.1.2 Пов'язані роботи 25](#_Toc453446556)

[3.1.3 Обсяг робіт 28](#_Toc453446557)

[3.2 Відстеження 29](#_Toc453446558)

[3.2.1 Оцінка оптичного потоку 30](#_Toc453446559)

[3.3 Виявлення помилок 32](#_Toc453446560)

[3.4 Модель трансформації 33](#_Toc453446561)

[3.5 Виявлення 34](#_Toc453446562)

[3.5.1 Алгоритм вікна 36](#_Toc453446563)

[3.5.2 Знаходження переднього плану 37](#_Toc453446564)

[3.5.3 Фільтр по дисперсії 39](#_Toc453446565)

[3.5.4 Класифікатор 43](#_Toc453446566)

[3.5.5 Узгодження шаблонів 47](#_Toc453446567)

[3.6 Самонавчання 49](#_Toc453446568)

[3.6.1 Поєднання результатів 50](#_Toc453446569)

[3.6.2 P/N – Навчання 51](#_Toc453446570)

[ВИСНОВОК ДО РОЗДІЛУ 3 55](#_Toc453446571)

[РОЗДІЛ 4. ЕКСПЛУАТАЦІЯ ПРИСТРОЮ 57](#_Toc453446572)

[4.1 Демонстрація роботи 58](#_Toc453446573)

[ВИСНОВОК ДО РОЗДІЛУ 4 61](#_Toc453446574)

[ВИСНОВОК 62](#_Toc453446575)

[СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ 63](#_Toc453446576)

[ДОДАТКИ](#_Toc453446577) 69

[ДОДАТОК А. КОПІЇ ГРАФІЧНИХ МАТЕРІАЛІВ](#_Toc453446578) 70

[ДОДАТОК Б. ЛІСТИНГ ПРОГРАМИ](#_Toc453446580) 74

# ВСТУП

Трапляються ситуації, коли необхідно зазирнути у недоступну для людини зону. Саме для таких цілей я створив свій прилад та написав програмне забезпечення для нього, яке дозволяє в автоматизованому режимі знаходити та відстежувати шукані об’єкти.

Розвиток даної технології почався ще в 1795 році і актуальність удосконалення таких приладів є навіть зараз, бо даний тип пристроїв дозволяє проаналізовати середовище без його пошкодження, що є, наприклад, у медицині критично важним, оскільки пошкодження тканин пацієнта може призвести до летальних наслідків. Також аналіз складних механічних або електричних систем у наші дні потребує таких пристроїв для ремонту та діагностиці.

Мій прилад збудований на основі смартфону під управлінням операційної системи Android, що забезпечує відкриту платформу для створення програмного забезпечення іншими розробниками. Також у результаті виконання роботи було досягнуто роботи модуля камери смартфона без додаткових драйверів, що дозволяє встановлювати модифікацію на будь-який смартфон без написання для нього спеціального драйверу.

# РОЗДІЛ 1 ДІАГНОСТИКА ВАЖКОДОСТУПНИХ СЕРЕДОВИЩ

## Періоди розвитку ендоскопії

Дана розробка відноситься до підвиду приладів, що призначені для дослідження не доступних для людського ока елементів різних середовищ.

Уперше дану проблему вперше спробував вирішити Філіп Боззіні в 1806 році. Необхідність у даному приладі була пов’язана з дослідженням людського тіла. Усього ендоскопія налічує чотири періоди свого існування, які будуть представлені нижче.

### Ригідний період

Початок першого етапу слід віднести до кінця 1795 року, коли були зроблені перші, досить небезпечні спроби ендоскопічних досліджень. У 1806 р Філіп Боззіні (1773-1809) сконструював апарат для дослідження прямої кишки і матки використовуючи, як джерело світла свічку. Цей інструмент був названий "LICHTLEITER".

Однак сконструйований ним апарат не знайшов практичного застосування і ніколи не використовувався для дослідження на людях. У той час не розуміли вагомості цього винаходу. Сам же винахідник був покараний медичним факультетом міста Відня за "цікавість".

У 1826 році Ф. Л. Сегалас повідомив про застосування вдосконаленого апарата, сконструйованого Боззіні.

Французький хірург Антуан Жеан Десормекс, що вважається "батьком ендоскопії", в 1853 році застосував для освітлення під час ендоскопічного дослідження спиртову лампу, що дозволило здійснювати більш детальний огляд. Інструмент поєднував у собі систему дзеркал і лінз і використовувався, головним чином, для огляду урогенітального тракту. Головними ускладненнями при таких дослідженнях були опіки.

A. Кусмуал в 1868 році ввів в практику методику гастроскопії за допомогою металевої трубки з гнучким обтуратором. Спочатку в шлунок вводився гнучкий провідник (обтуратор), а по ньому металева порожниста трубка. Введення такої трубки було можливо за умови, що верхні зубці знаходилися на одній прямій з віссю стравоходу. Надалі принцип Куссмауля був покладений в основу всіх методик з використанням жорстких і напівжорстких гастроскопії.

У тому ж році Л. Бевен розробив жорсткий езофагоскоп, який був призначений для вилучення чужорідних тіл і огляду пухлин стравоходу, прилад мав довжину 10 см.

Важливою віхою в розвитку гастроскопії була робота Ж. Мікулікс (1881). На підставі ретельних анатомічних досліджень автор розробив конструкцію апарату, вигнутого в дистальної третини під кутом 30 °. Його ідея була в той час важко здійсненна технічно, проте цей принцип був використаний при подальшій розробці апаратів для огляду шлунку. Цю роботу розцінюють як одне з найважливіших теоретичних обґрунтувань методу.

Надалі жорсткі езофагоскопа і гастроскопи удосконалювалися. Удосконалювалася і методика досліджень. Т. Розенхейм (1896) вперше застосував місцеву анестезію кокаїном. Ж. Келлінг (1898) винайшов керований гастроскоп, Ф. Ланже і Д. Мелтізін (1898) - гастрокамер для фотографування шлунку без візуального огляду.

В кінці XIX століття, коли була винайдена лампа Едісона, при ендоскопії почали застосовувати мініатюрні електричні лампочки. Ж. П. Туртле (1902) вперше використав таку лампу при ректоскопії, а Т. Розенхейм (1906) - при гастроскопії. Сконструйований В. Брунінг (1907) езофагоскопа з електричним освітленням (електроскоп) застосовувався в практиці до 70-х років XX століття.

### Напівгнучкий період

Найбільший внесок у розвиток гастроскопії у цей період зробив Р. Шіндлер (1932), який описав ендоскопічну картину слизової оболонки шлунку при ряді захворювань, а також розробив конструкцію напівгнучкого лінзового гастроскопу. Цей апарат в різних модифікаціях широко використовувався в 1932-1958 рр. і ознаменував собою початок нового етапу в розвитку ендоскопічних методів дослідження шлунку. Гастроскоп Шиндлера представляв собою трубку довжиною 78 см, його гнучка частина мала 24 см у довжину, 12 мм у діаметрі і містила велику кількість короткофокусних лінз, які забезпечують можливість огляду. Цей інструмент дозволяв детально обстежити 4/5 або 7/8 слизової оболонки шлунку, однак більшості досліджень супроводжував досить виражений дискомфорт, що обмежувало застосування гастроскопії. Проте, завдяки ентузіазму та наполегливості, автору вдалося досить широко впровадити методику в медичну практику. Р. Шіндер по праву можна вважати "батьком гастроскопії".

У наступні роки було запропоновано багато модифікації напівгнучкого гастроскопа. Дві моделі гастроскопа Н. Хеннінг (1939, 1948) відрізнялися меншою товщиною гнучкої частини (7,5 мм), тому обстеження з їх допомогою легше переносилося хворими. Ф. Тайлор (1941) сконструював гастроскоп із гнучкою дистальною частиною, яка при управлінні дозволяла оглядати частину "сліпих" зон шлунку. Незабаром була розроблена модель гастроскопа "Edel-Palmer" з керованим в одній площині дистальним кінцем. Цей апарат був тонше, ніж апарат "Wolf-Schindler", і тривалий час залишався найпоширенішим типом гастроскопа. Подальше вдосконалення напівгнучких ендоскопів йшло по шляху поліпшення їх оптичних властивостей і розробки принципів біопсії через гастроскоп. У 1948 році Е. Б. Бенедікт створив операційний гастроскоп, що має біопсійний канал і дозволяє проводити маніпуляції усередині шлунку.

У цей же рік лікарі та дослідники знову повернулися до проблеми фото документації. Перші успішні досліди з внутрішньо шлункової фотографією були проведені T.Ужі (1950). У 1958 р С. Тасака і С. Ашізава представили фотографії, виконані за допомогою гастрокамер; останні отримали велике поширення в Японії і практично конкурували з гастроскопії.

### Оптоволоконний період

Третій етап у ендоскопії почався після публікації Хіршофітс В. I., в 1958 році робіт, присвячених практичному застосуванню гнучкого фиброгастроскопа, хоча ідея передачі світла по взяли участь Куртісс, Хіршофітс і Петерс. Цей апарат мав значно більші можливості в порівнянні з найдосконалішою моделлю напівгнучкого ендоскопа і дослідження з його допомогою краще переносилося гнучким скляним волокнах була запропонована вже в 1927 році, а когерентний оптичний пучок був запропонований Хопкінсом в 1954 році. У створенні першого фиброгастроскопа хворими. З цього часу починається розвиток сучасної ендоскопії, яка постійно розширює сферу свого застосування. У даний час в ендоскопію шлунку використовує фібро гастроскоп, який дозволяє значно розширити межі огляду, детально оцінювати стан слизової оболонки стравоходу, шлунку, дванадцятипалої і початкового відділу тонкої кишки, виробляти прицільну біопсію, запис, передавати зображення на телеекран. Особливу роль придбала езофагогастродуоденоскопія у зв'язку з розвитком і вдосконаленням ендоскопічних лікувальних маніпуляцій.

### Електронний період

Нинішній електронний період почався в Bell Laboratories (AT & T), коли Бойлі і Сміт в 1969 році створили прилад із зарядним зв'язком (ПЗС), перетворює оптичні сигнали в електричні імпульси. Десять років по тому інженерами компанії Welch Allyn був створений перший електронний ендоскоп - ендоскопія увійшла в століття цифрових технологій. Електронна відео ендоскопія дала можливість відразу декільком фахівцям бачити весь процес ендоскопічного дослідження, збільшувати зображення і зберігати його в комп'ютерній базі даних.

## Застосування в медицині

Ендоскопія — метод заглядання всередину тіла та обстеження внутрішніх органів людини за допомогою медичного приладу — [ендоскопа](https://uk.wikipedia.org/wiki/%D0%95%D0%BD%D0%B4%D0%BE%D1%81%D0%BA%D0%BE%D0%BF" \o "Ендоскоп), без порушення цілісності шкірних покривів та [слизових оболонок](https://uk.wikipedia.org/wiki/%D0%A1%D0%BB%D0%B8%D0%B7%D0%BE%D0%B2%D0%B0_%D0%BE%D0%B1%D0%BE%D0%BB%D0%BE%D0%BD%D0%BA%D0%B0" \o "Слизова оболонка). Проте дедалі частіше в [хірургічній](https://uk.wikipedia.org/wiki/%D0%A5%D1%96%D1%80%D1%83%D1%80%D0%B3%D1%96%D1%8F" \o "Хірургія) практиці застосовують травматичні види ендоскопії.[52]

Ендоскопи є двох типів: прямі (металеві) та гнучкі (еластичні). Більшого розповсюдження набули еластичні, оскільки вони дозволяють легше [пацієнту](https://uk.wikipedia.org/wiki/%D0%9F%D0%B0%D1%86%D1%96%D1%94%D0%BD%D1%82" \o "Пацієнт) перенести маніпуляцію, а лікарю-діагносту адекватніше та якісніше здійснити [обстеження](https://uk.wikipedia.org/wiki/%D0%9E%D0%B1%D1%81%D1%82%D0%B5%D0%B6%D0%B5%D0%BD%D0%BD%D1%8F" \o "Обстеження). Та все ж таки є ділянки, обстеження яких зручніше, швидше, простіше і надійніше здійснювати прямим.

При ендоскопії ендоскопи зазвичай вводяться в порожнину тіла через природні шляхи, наприклад, в [шлунок](https://uk.wikipedia.org/wiki/%D0%A8%D0%BB%D1%83%D0%BD%D0%BE%D0%BA" \o "Шлунок) — через [рот](https://uk.wikipedia.org/wiki/%D0%A0%D0%BE%D1%82" \o "Рот) і [стравохід](https://uk.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D0%B0%D0%B2%D0%BE%D1%85%D1%96%D0%B4" \o "Стравохід), [бронхи](https://uk.wikipedia.org/wiki/%D0%91%D1%80%D0%BE%D0%BD%D1%85%D0%B8" \o "Бронхи) і [легені](https://uk.wikipedia.org/wiki/%D0%9B%D0%B5%D0%B3%D0%B5%D0%BD%D1%96" \o "Легені) — через [гортань](https://uk.wikipedia.org/wiki/%D0%93%D0%BE%D1%80%D1%82%D0%B0%D0%BD%D1%8C" \o "Гортань), в [сечовий міхур](https://uk.wikipedia.org/wiki/%D0%A1%D0%B5%D1%87%D0%BE%D0%B2%D0%B8%D0%B9_%D0%BC%D1%96%D1%85%D1%83%D1%80" \o "Сечовий міхур) — через [сечовивідний канал](https://uk.wikipedia.org/w/index.php?title=%D0%A1%D0%B5%D1%87%D0%BE%D0%B2%D0%B8%D0%B2%D1%96%D0%B4%D0%BD%D0%B8%D0%B9_%D0%BA%D0%B0%D0%BD%D0%B0%D0%BB&action=edit&redlink=1" \o "Сечовивідний канал (ще не написана)), хоча в деяких випадках введення ендоскопа вимагає хірургічного створення розрізу в тілі — тоді говорять про травматичну ендоскопію.

Медичний ендоскоп призначений для огляду порожнини і внутрішньої поверхні:

* стравоходу зветься [езофагоскоп](https://uk.wikipedia.org/wiki/%D0%93%D0%B0%D1%81%D1%82%D1%80%D0%BE%D1%81%D0%BA%D0%BE%D0%BF" \o "Гастроскоп),
* шлунку — [гастроскоп](https://uk.wikipedia.org/wiki/%D0%93%D0%B0%D1%81%D1%82%D1%80%D0%BE%D1%81%D0%BA%D0%BE%D0%BF" \o "Гастроскоп),
* 12-палої кишки — [дуоденоскоп](https://uk.wikipedia.org/wiki/%D0%93%D0%B0%D1%81%D1%82%D1%80%D0%BE%D1%81%D0%BA%D0%BE%D0%BF" \o "Гастроскоп),
* товстої кишки — [колоноскоп](https://uk.wikipedia.org/w/index.php?title=%D0%9A%D0%BE%D0%BB%D0%BE%D0%BD%D0%BE%D1%81%D0%BA%D0%BE%D0%BF&action=edit&redlink=1" \o "Колоноскоп (ще не написана)),
* сечового міхура — [цистоскоп](https://uk.wikipedia.org/w/index.php?title=%D0%A6%D0%B8%D1%81%D1%82%D0%BE%D1%81%D0%BA%D0%BE%D0%BF&action=edit&redlink=1" \o "Цистоскоп (ще не написана)),
* суглобів — [артроскоп](https://uk.wikipedia.org/wiki/%D0%90%D1%80%D1%82%D1%80%D0%BE%D1%81%D0%BA%D0%BE%D0%BF" \o "Артроскоп),
* черевної порожнини — [лапароскоп](https://uk.wikipedia.org/w/index.php?title=%D0%9B%D0%B0%D0%BF%D0%B0%D1%80%D0%BE%D1%81%D0%BA%D0%BE%D0%BF&action=edit&redlink=1" \o "Лапароскоп (ще не написана))

## Застосування в інших галузях

Ендоскопію часто використовують для досліджень внутрішніх пошкоджень машин.

Ендоскоп дозволяє оцінювати технічний стан внутрішніх деталей машини без її розбирання - тобто даю змогу заглянути в середину через наявні технологічні отвори. При цьому визначається наявність поверхневих дефектів, в залежності від конструкції і призначення машини, це можуть бути дефекти типу тріщини, забоїн, прогари, корозія. Вимірюється ступінь зносу, перевіряється правильність взаємного розташування деталей. Також, новітні ендоскопи дозволяють документувати отриману інформацію в формі відеозапису, фотознімків або цифрових зображень для подальшого повторного перегляду та обробки.

Ендоскопічна діагностика застосовується на різних стадіях виробництва і експлуатації машин і механізмів:

- При вхідному контролі якості. Дозволяє виявити приховані виробничі дефекти нового обладнання і скласти протокол рекламацій, підкріплений фотографіями дефектів.

- При розробці і налаштуванні нових приладів. Забезпечує «чистоту експерименту» при виробничих і ресурсних випробуваннях, дозволяючи простежувати технічний стан вузла без демонтажу зі стенду і без його розбирання, дозволяє вчасно визначати і спрогнозувати відмови, визначати ресурс вузлів.

- У технології виробництва. Дозволяє забезпечити контроль якості виготовлення і ремонту деталей і вузлів на всіх стадіях виробництва.

- Під час обслуговування та експлуатації приладів. Найбільш широко, технічна ендоскопія застосовується саме в експлуатації. Використання ендоскопії разом з іншими методами неруйнівного контролю дозволяє більш впевнено ідентифікувати результати діагностики, отримані ультразвуковими, вихрострумовий, радіографічних, віброакустичними, акусто-емісійними, металографічними, телевізійними і іншими методами. У деяких випадках, технічний ендоскоп є єдиним можливим засобом неруйнівного контролю.

Зручність роботи з приладом і однозначність отриманих результатів, а також відсутність необхідності розбирання для перевірки вузлів роблять ендоскопію незамінним інструментом в дослідженнях машин.

# ВИСНОВОК ДО РОЗДІЛУ 1

Ендоскопія є потужним засобом діагностики та виявлення небажаних явищ у важко доступних середовищ без нанесення шкоди при досліджені.

Цей вид дослідження вже давно набув популярності в медицині і його користь використання свідчить розповсюдженість та кількість захворювань, які можна діагностувати за його допомогою.

У наші дні такі прилади почали використовувати для ремонту та діагностики машин – це пов’язано із ускладненням механізмів та їх розборки\зборки. Майже кожна автомайстерня використає ендоскопи для огляду машин, що дає можливість швидко виявити неспрвність без розбору.

Отже, ендоскоп – це прилад, який дозволяє проаналізувати важкодоступне середовище без зайвої шкоди для нього, такий підхід економить багато часу та ресурсів, тому розробка даної теми є актуальною.

# РОЗДІЛ 2 РОЗРОБКА ФІЗИЧНОЇ ЧАСТИНИ ПРИСТРОЮ

## Опис бази приладу

Для реалізації проекту було взято за базу смартфон LG P700. Його розміри стандартні для Android-смартфона 125,5 мм у висоту і 67 мм в ширину. Товщина складає 8,7 мм, важить L7 122 грам.

Рис. 2.1 Android смартфон LG P700

Лицьову панель займає 4,3-дюймовий екран з роздільною здатністю WVGA, захищений Gorilla Glass з олеофобним покриттям. Зверху над ним - 0.3-мегапіксельна фронтальна камера, проріз під розмовний динамік і датчики - наближення та освітлення. Знизу розташована механічна кнопка "Додому" і по боках від неї - сенсорні "Назад" і "Меню". За усталеною серед більшості виробників традицією, екран втоплений всередину корпусу, тому можна не боятися, що він подряпається при розташуванні смартфона обличчям вниз.

На верхньому боці розташований 3,5-мм аудіо вихід і кнопка включе-ня / розблокування. На нижньому торці розташований micro-USB роз'єм.



Рис. 2.2 Торець смартфону LG P700

Ліва грань порожня, на правій же розташована гойдалка гучності. На задній частині знаходиться об'єктив 5-мегапіксельної камери і спалах, обрамлені вставкою під метал, логотип LG і вбудований динамік.



Рис 2.3 Задня сторона смартфону LG P700

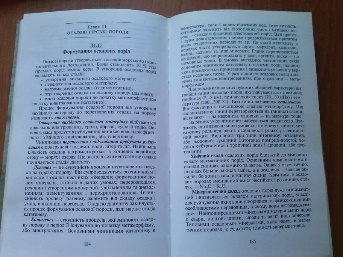
### Екран

Безумовним плюсом смартфона для проекту є екран. Він має невелику роздільну здатність у 800х480 пікселів при 4,3 дюймах, він виготовлений за технологією IPS – це означає, що він має відмінну передачу кольору і добрі кути огляду (180 градусів), що є великим плюсом для майбутнього приладу. Так і є, дисплей має відмінний запас яскравості і кольору передачі. Також смартфон має систему Multitouch, яка підтримує десять пальців.

### Камера

Смартфон має в собі досить слабкий модуль камери на сьогоднішній день. У ньому встановлена 5-мегапіксельна камера із спалахом і автофокусом, але якість фотографій залишає бажати кращого, більш-менш гарні знімки виходять лише при великій кількості світла і то замилені. Навіть при незначній нестачі освітлення, камера починає сильно шуміти і спалах тут мало чим допомагає. Автофокус часто дає збій, під час фотографування тексту.

Також є можливість писати відео в 480р, якість, як нескладно до-гадати, досить низька. Незрозуміло чому відсутній зум при зйомці відео.

[](https://lh4.googleusercontent.com/-2V9vogI-_vI/T76Y7q4TcFI/AAAAAAAAEA0/ZW1IOR3GPL4/s912/CAM00007.jpg)[](https://lh6.googleusercontent.com/--ISrOwZxssw/T76ZAA7vXiI/AAAAAAAAEBg/45bNUEjK1cU/s912/CAM00011.jpg)[](https://lh6.googleusercontent.com/-cGmp-d2rI4A/T76Y-UYaVoI/AAAAAAAAEBQ/6AvF16-RFwI/s912/CAM00032.jpg)[](https://lh3.googleusercontent.com/-S2myg8h7T7M/T76ZAj8GRII/AAAAAAAAEBk/LmZapreckeY/s912/CAM00037.jpg)

[](https://lh5.googleusercontent.com/--YZuDe5ZGDU/T76ZAwqD56I/AAAAAAAAEBo/qjcDZ1LL3Zk/s912/CAM00038.jpg)[](https://lh5.googleusercontent.com/-0HUERRb0R-c/T76ZBkao5RI/AAAAAAAAEB0/WM0MMti6H3M/s912/CAM00041.jpg)[](https://lh6.googleusercontent.com/-ReATZ4oFYNs/T76Y7ENPIUI/AAAAAAAAEAw/Wp84aVeP7-Q/s912/CAM00005.jpg)

Рис. 2.4 Демонстрація роботи модуля камери смартфона

### Продуктивність

У ньому встановлений процесор Qualcomm MSM7227, виконаний по архітектурі Cortex A5 і відео-прискорювач Adreno 200, доповнені 512 мегабайтами оперативної пам'яті. При цьому, частота процесора збільшена з стандартних 600 МГц до 1 ГГц. На перший погляд, ця пара гідно проявляє себе в порівнянні на рівні із старими одноядерними флагманами, Galaxy S або Desire HD, але порівняно із сучасними моделями вона є досить відсталою.

Таблиця 2.1

|  |  |
| --- | --- |
| Процесор | Qualcomm Snapdragon S1 MSM7227A + GPU Adreno 200 |
| Тип ядра | Cortex-A5 |
| Кількість ядер | 1 |
| Частота, ГГц | 1 |
| Слот розширення | microSD/SDHC (до 32 ГБ) |
| Оперативна пам'ять, ГБ | 0,5 |
| Вбудована пам'ять, ГБ | 2,7 |
| Операційна система | Android 4.0 (ICS) |

### Час життя батареї

Слабкий процесор є великим мінусом смартфона, але це одночасно є причиною довгого життя смартфона від батареї, обсяг якої становить 1700 мАг.

Приблизний час життя смартфона – це 2 дні без підзарядки, таким чином отримується досить живучий прилад, що дасть можливість зручно працювати майбутнім користувачам.

## Дослідження роботи модуля камери смартфона

Дослідження даної частини було почати з вивчення документації по ремонту смартфона[53]. Було встановлено, що модуль камери використовує 24-сокет, який використовується для живлення камери, обміну інформації між смартфоном і камерою.

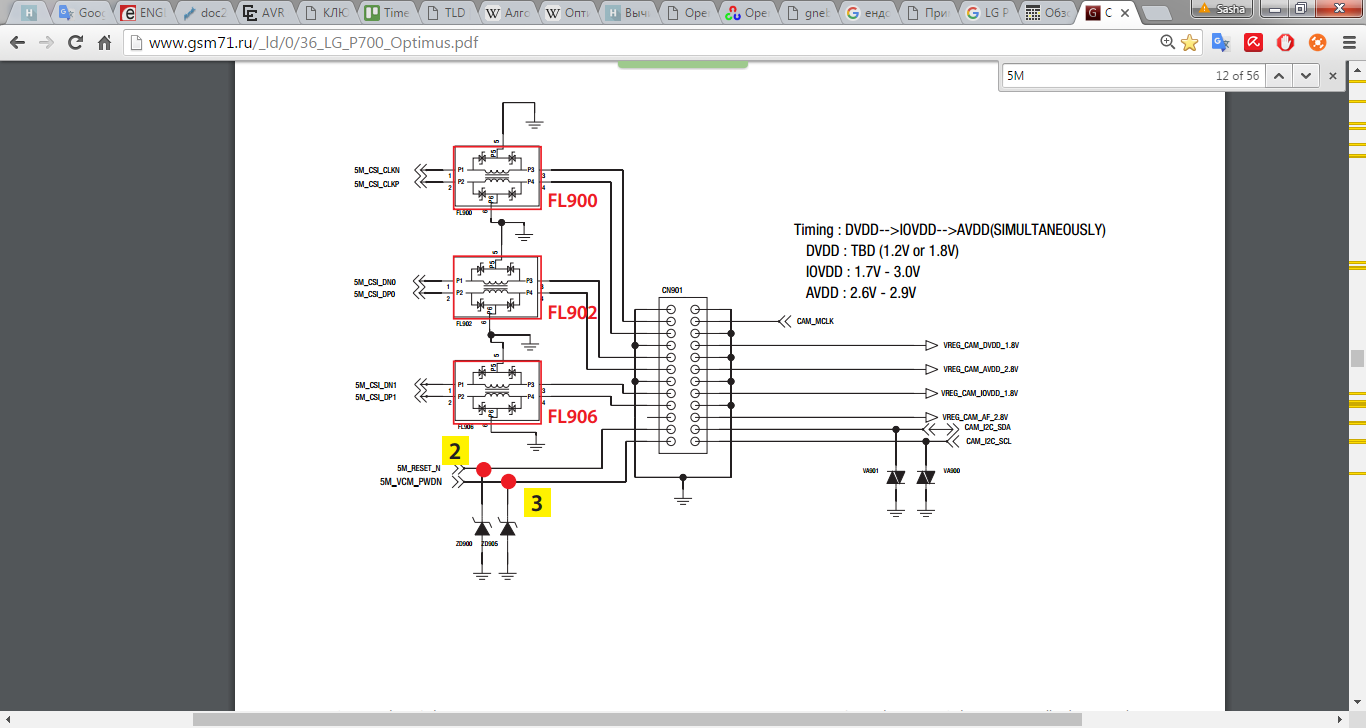


Рис. 2.5 Принципова схема з’єднання

Із схеми видно, що 8 контактів підключені до землі – це зроблено спеціально, щоб подавити електромагнітне поле, яке створюють лінії живлення.

Під час фізичного огляду було встановлено, що для зборки смартфона використані контакти, які зроблені не за стандартом.

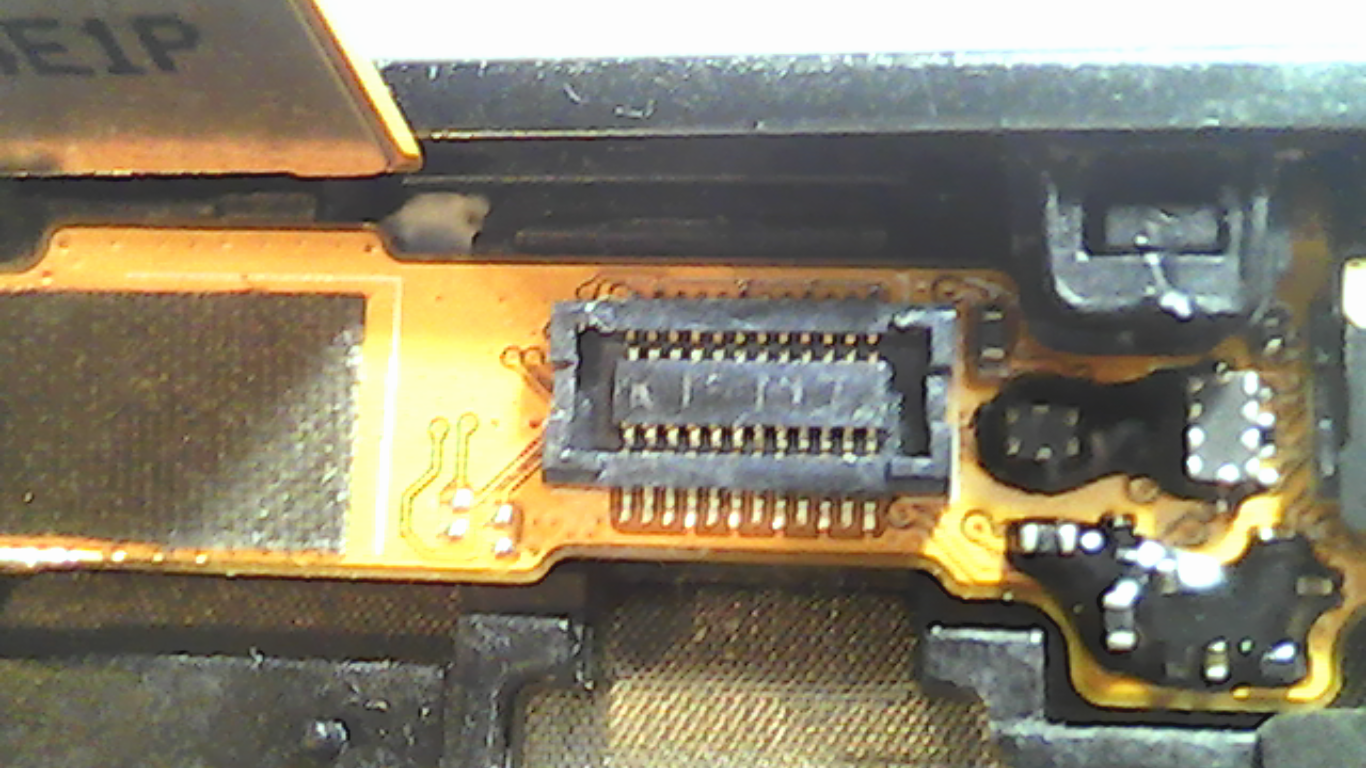


Рис. 2.6 Сокет модуля камери

Після огляду – було встановлено, що смартфон розроблений з точністю 0.2 мм, що є досить слабким показником, але дана точність дозволяє в кустарних умовах провести модифікацію пристрою під свої потреби.

Після замірів модуля камери було встановлено його розміри 5.5х7.8х5.5 мм – ці розміри є досить компактними, що дає можливість використовувати їх у важкодоступних місцях.

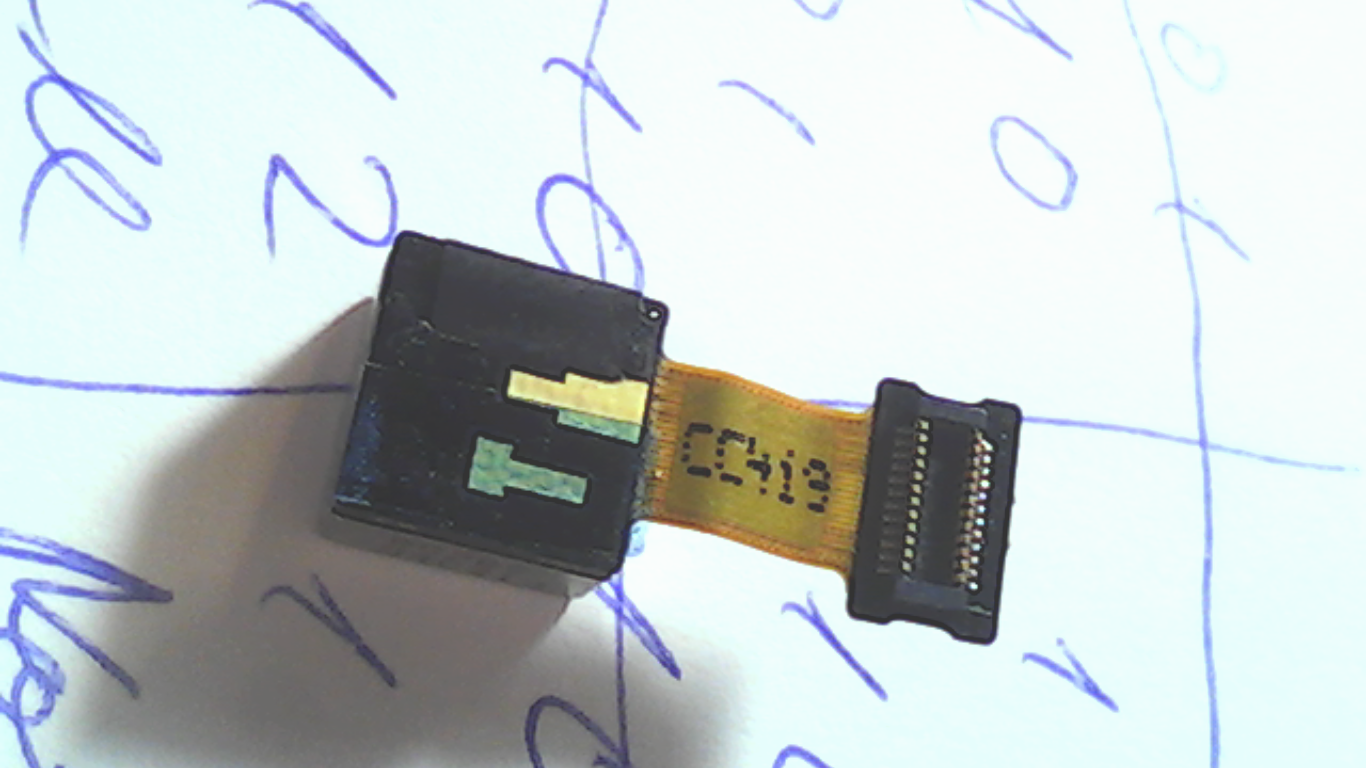


Рис. 2.7 Модуль камери з роз’ємом

Після досліджень стало зрозуміло, що модуль камери використовує аналоговий спосіб передачі даних, що є дуже поміхо нестійким засобом обміну даних. Це варто врахувати при розробці адаптера.

## Розробка адаптера

Розробка цього адаптера пов’язана із наступними перевагами над звичайними ендоскопами: вони є надзвичайно малими, легкими, багатофункціональними та повністю програмованими. Також камера смартфона має можливість автофокусу, що значно підвищує ймовірність правильного розпізнавання об’єктів.

Моя система заснована на смартфоні LG Р705, який оснащений 5МP камерою з автофокусом. Проблема використання цього приладу для реалізації мого проекту була в неможливості винести камеру за його межі, тому я розробив подовжувач-адаптер, який дозволяє винести саму камеру за межі смартфона.

При проектування подовжувача-адаптера було враховано високошвидкісну передачу даних між камерою і смартфоном та наслідки не синхронізованих проміжків часу передачі сигналів, які викликають помилку роботи камери. Для усунення цього всі ланцюги були вирівняні по довжині та виготовлені з однакових матеріалів (Рис. 2.8 Схема адаптера конвектора камери).

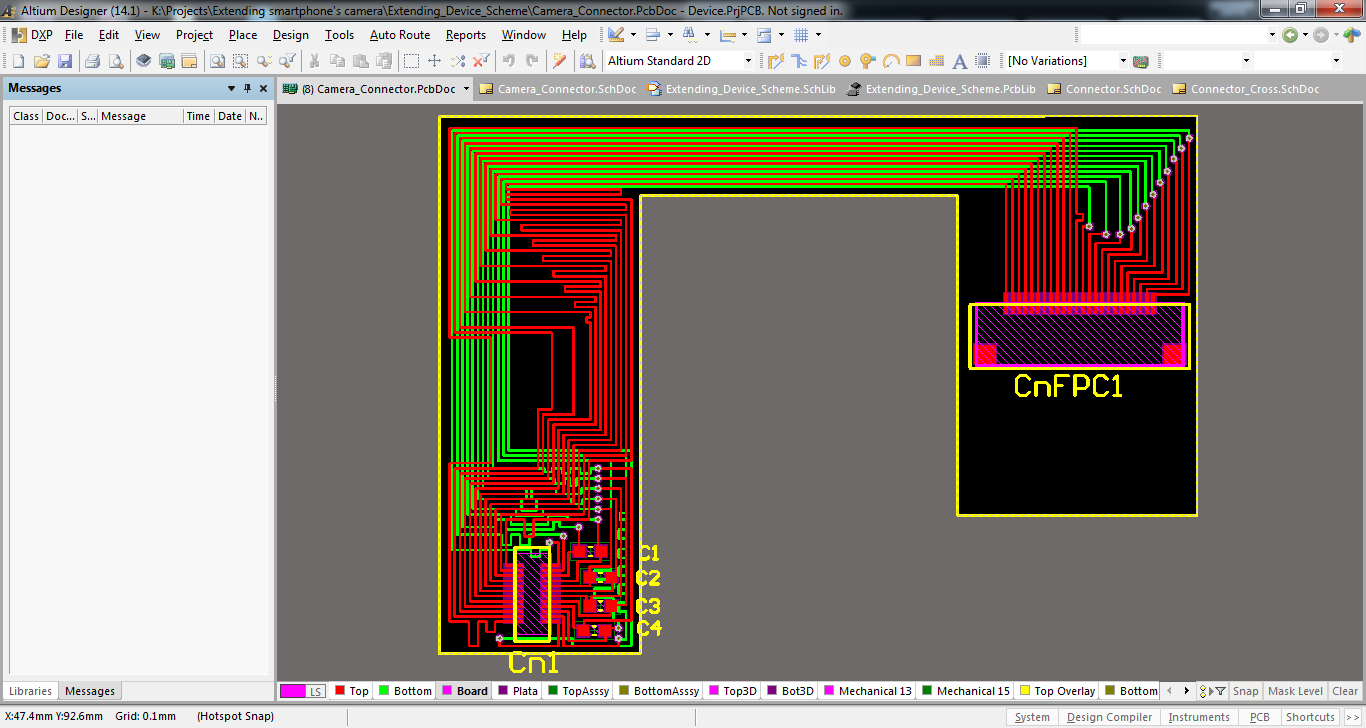


Рис. 2.8 Схема адаптера конвектора камери

Під час проектування було також враховано втрати потужності сигналу при збільшення відстані передачі, для усунення цієї проблеми до ланцюгів передачі даних було додано конденсатори, які своїм зарядом підсилювали сигнал, що дозволяло працювати камері коректо.

Модуль камери підключений до материнська плата за допомогою роз'єму з 24 контактами, з яких 4 контакти використовуються для живлення різної напругою модуль камери (Рис. 2.9 Схема адаптера камери до смартфона), 10 контактів відведені для передачі даних та для годинника станів модуля. Решта призначені для сигналів контролю, даних та землі.

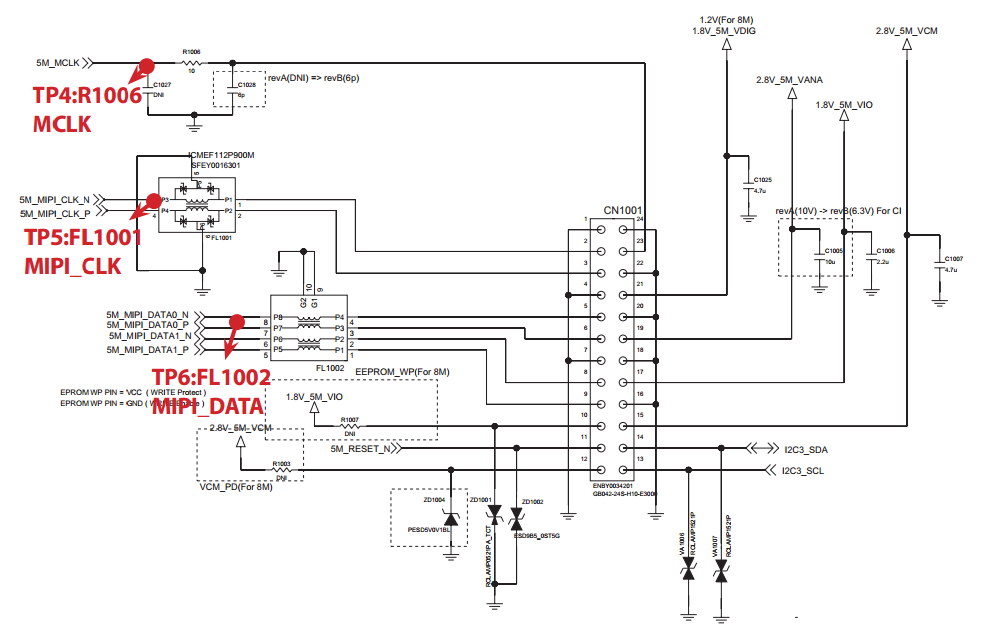


Рис. 2.9 Схема адаптера камери до смартфона

## Порівняння з аналогами

Найбільш схожою на мою розробку є USB ендоскопом для Android. Оскільки прилади мають програмне забезпечення, тому вони буде порівнюватися окрема за програмним продуктом та за технічними характеристиками.

### Технічне порівняння

Порівняння подовжувача та найкращого ендоскопа на ринку.

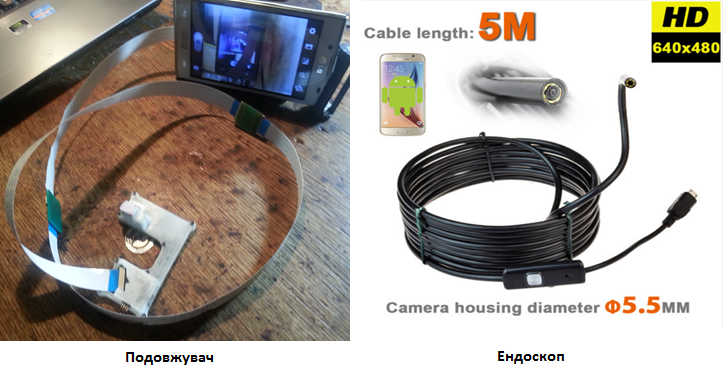


Рис. 2.10 Подовжувач та ендоскоп

Таблиця 2.2

|  |  |  |
| --- | --- | --- |
|  | Подовжувач | Ендоскоп |
| Розмір камери в мм | 8х8х5 | 5.5х7.8х5.5 |
| Кількість мега пікселів | 5 | 1.3 |
| Роздільна здатність фото | 2592х1944 | Нема |
| Відео зйомка | 640х480 | 640х480 |
| Цифровий зум | Так | Ні |
| Автофокус | Так | Ні |
| Спалах | Так | Так |
| Подовження | До 10 м | До 10 м |
| Потрібний драйвер для роботи | Ні | Так |
| Використовує Mini USB Port | Ні | Так |
| Використовує камеру смартфона | Так | Ні |
| Прив’язано до смартфона | Так | Ні |
| Вартість | 200 грн | 1931 грн |

Із таблиці видно, що ендоскоп значно поступається моєму подовжувачу. Єдина перевага в універсальності ендоскопа для всіх смартфонів, але з цим пов’язаний один з його недоліків, а саме використання Mini USB роз’єму, що не дає змогу використовувати ендоскоп та зарядку одночасно або Power Bank. Також для роботи ендоскопу потрібна спеціальна програма, що ускладнює розробку програмного забезпечення для цього приладу.

Найбільшою проблемою ендоскопа є неможливість зміни фокусу, в моєму ж пристрої вбудовано автоматичне налаштування фокуса, що є найбільшою перевагою над конкурентом.

Варто зазначити, що для порівняння було узято найкращу модель ендоскопа для Android, а для подовжувача використана застаріла модель смартфона LG P700. Сучасні смартфони мають значно вищі характеристики. Наприклад: сучасний модуль камери має 20 мега пікселів, що є у 4 рази більше за використаний модуль.

### Програмне порівняння

Для створення програмного забезпечення я взяв за основу алгоритм OpenTLD, який базується на алгоритмі Лукаса — Канаде.

Детальний опис програмної частини є в третьому розділі.

Таблиця порівняння програмного забезпечення мого приладу та найкращого ендоскопа.

Таблиця 2.3

|  |  |  |
| --- | --- | --- |
|  | Подовжувач | Ендоскоп |
| Відео зйомка | Так | Ні |
| Демонстрація зображення з камери | Так | Так |
| Знаходження заданого об’єкту в реальному часі | Так | Ні |
| Спостереження за заданим об’єктом в режимі реального часу | Так | Ні |
| Використання фільтрів відео | Так | Ні |
| Зум камери | Так | Ні |
| Програмний зум камери | Так | Ні |
| Фото | Так | Ні |

Із таблиці видно, що програмне забезпечення ендоскопа досить примітивне порівняно з можливістю мого продукту.

# ВИСНОВОК ДО РОЗДІЛУ 2

Розробка адаптера дозволила реалізувати технічну частину приладу, а його характеристики більш виші за найкращій ендоскоп представлений на ринку на сьогоднішній день. Також простота виготовлення зумовлює дешевизну

приладу. Сама конструкція є дуже гнучкою, що дає широкий спектр для модифікацій.

# РОЗДІЛ 3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

## Опис OpenTLD

Зорова кора головного мозку людини знаходить і ідентифікує об'єкти шляхом аналізу інформації, прибувають в якості потенціалів дії, які запускаються в сітківці [20]. У той час як перцептивних психологи вивчити, як зорова система людини інтерпретує стимули навколишнього середовища, дослідники в комп'ютері бачення розробляти математичні методи з метою отримання інформації про фізичні об'єкти на основі зображень з камери [44]. Методи комп'ютерного зору застосовуються для оптичного розпізнавання символів, контролю якості, керівництва роботами, реконструкції сцен і категоризації об'єктів [47].

Однією з гілок досліджень в області комп'ютерного зору є розпізнавання та відстеження об'єктів. Це направлення вивчає методи, що оцінюють розташування об’єктів в послідовних відеокадрах [34].

Поширення потужних комп'ютерів, наявність високоякісних і недорогих відеокамер, а також необхідність для автоматизованого аналізу відео дало поштовх до застосування алгоритмів відстеження об'єктів в автоматизованих системах відеоспостереження, автоматичних анотацій відеоданих, взаємодії людини з комп'ютером та навігації транспортних засобів [50].

### Визначення проблеми

Проблема полягає у створенні алгоритму, який би був здатний оцінити із послідовності зображень *I1* ... *In*, стан об’єкта, який відстежується, для кожного кадру *Ік*. Методи відстеження об'єкту кодують його стан, як центроїди, що обмежує певним простором [34]. Наприклад, на рис. 3.1, де об’єкт обмежує прямокутник, який показує об'єкт цікавості. У цьому випадку параметри обмежувального простору складаються з верхнього лівого кута прямокутника (х, у) та його ширини і висоти. Maggio and Cavallaro [34] група підходів, що засновані на кількості взаємодій з користувачем, які потрібні для ідентифікації об'єктів цікавості. Ручний трекінг вимагає взаємодії з користувачем в кожному кадрі. Автоматизований методи відстеження використовувати апріорну інформацію для того, щоб автоматично форматувати процес відстеження.

У напівавтоматичному відстеженні, який призначений для користувача, введення потрібно для того, щоб ініціювати процес відстеження. Згідно Maggio і Cavallaro [34], основною проблемою в процесі відстеження об'єкта є шум.

Шум - це коли, функції очікувані не може відділити об'єкт цікавості від інших об’єктів у простору пошуку. На рис 3.1 демонструється об’єкт цікавості та простір пошуку.

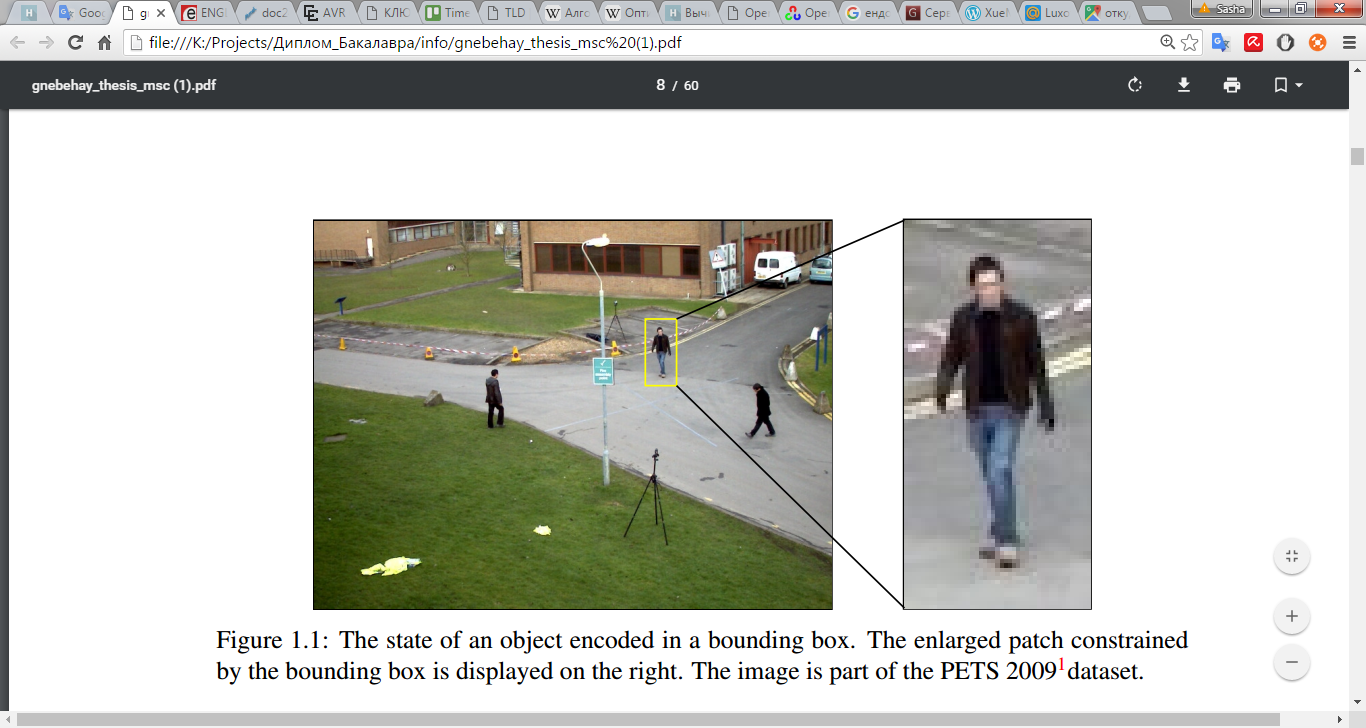


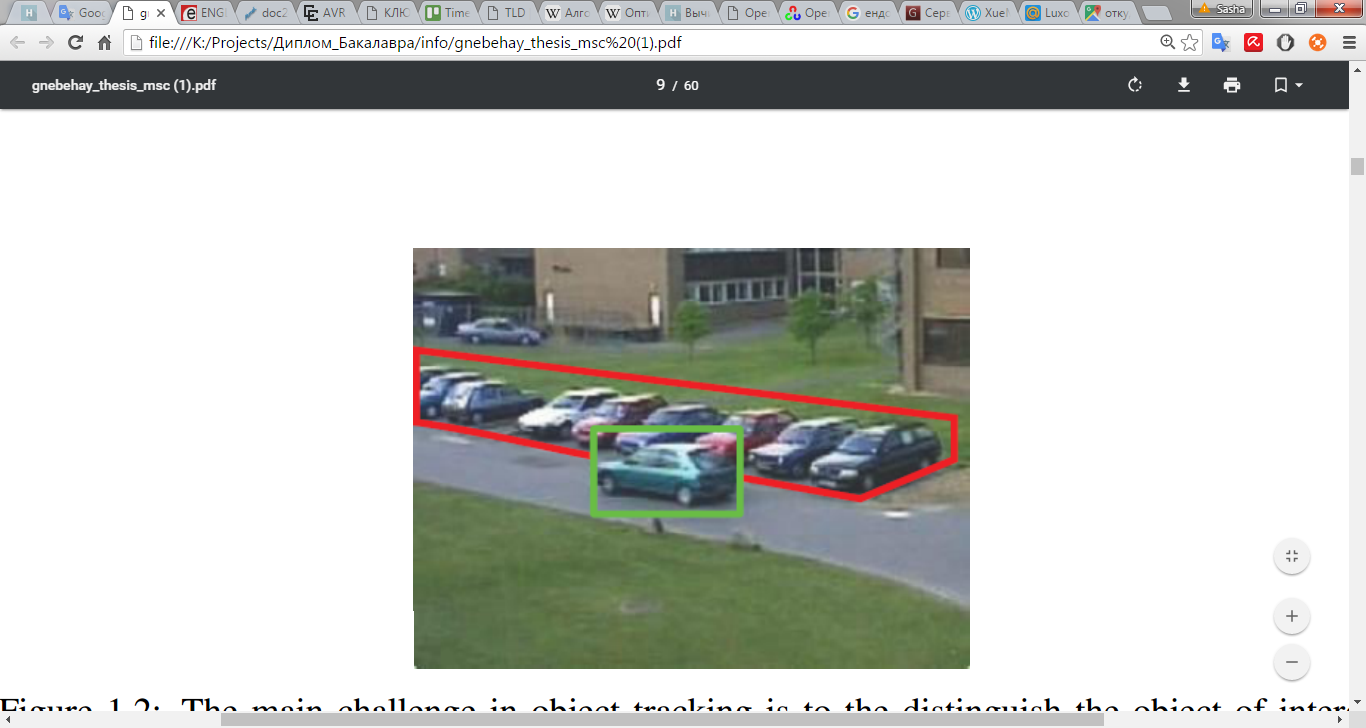
Рис 3.1 Об'єкт цікавості обведений жовтим прямокутником

На рис. 3.2 наведено приклад із перешкодами (Обведені червоним). На цьому зображенні присутні кілька об'єктів, які аналогічні за формою об'єкту цікавості. Ще однією проблемою є поява варіацій самого об’єкту.

Внутрішня мінливість зовнішнього вигляду включає в себе зміну позиції і форму об’єкта, у той самий час зовнішня мінливість включає в себе зміну освітлення, рух камери та інше[41]. Підходи, які підтримують шаблони об'єкта, як правило, стикаються з проблемами поновлення шаблону, який відноситься до проблеми оновлення існуючого шаблону так, щоб він залишався вірний моделі [35]. Якщо вихідний шаблон ніколи не змінюється то він з часом перестане бути точним представленням моделі цікавості. Коли шаблон адаптивне до кожної зміни зовнішнього вигляду - це призводить до накопичування помилок і шаблон вже не буде представляти точно модель цікавості. Ця проблема тісно пов'язана із пошуком компромісу між стабільністю, необхідністю збереження інформації та пластичності моделі для навчання [22]. Ця проблема стоїть перед усіма системами навчання. [1] Об'єкти піддаються колізіям, коли покривається іншими об'єктами або коли вони залишають поле зору камера. Для обробки таких випадків, необхідний механізм, який повторно виявляє об'єкт незалежно від його останньої позиції в зображенні [50].

### Пов'язані роботи

Лепетіт [30] виділяє дві парадигми у відстеженні об'єкта. Оцінка рекурсивності методу відстеження поточного стану об'єкта шляхом застосування перетворення на попередній стан *ХТ-1* на основі вимірів *Z1 ... ZT* прийнятих у відповідному зображенні. Рекурсивна оцінка стану залежить від стану об'єкта в попередньому кадрі та схильності до накопичення помилок [30]. Наприклад, Лукас і Канаде [33] запропонували метод для оцінки оптичного потоку об’єкта цікавості.

Рис. 3.2 Справжнє випробування для системи – це відділити об’єкт цікавості (Обведений зеленим) від схожих об’єктів (Обведені червоним)

Оптичний потік використовується для передбачення нової позиції об’єкта цікавості. У роботі я використовую метод Лукаса і Канаде для відстеження об'єкта цікавості на послідовних кадрах. Коменсіо [15] запропонував процес відстеження на основі середнього зсуву. Перетворення стану об'єкта шляхом знаходження максимуму функції подібності на основі колірних гістограм. На противагу цьому, методи відстеження та знаходження оцінюють стан об'єкта виключно за допомогою вимірювань, прийнятих в поточному зображенні. Цей принцип усуває ефект накопичення помилок. Проте, детектори об'єктів мають бути навчені заздалегідь. Озуал [38] створює синтетичний вид об'єкта шляхом застосування афінного перетворення шаблону і навчання детектора об'єкта на викривлених зображеннях цього об’єкта цікавості. Шаблон виявлення потім використовується в кожному кадрі, щоб відслідковувати об'єкт.

Авідан [4] інтегрує вектор підтримки машинної класифікації в оптичний потік на базі процес відстеження. Замість того, щоб звести до мінімуму різницю інтенсивності функції між послідовними кадрами, він максимізує рахунок класифікатора. Опорний вектор навчений заздалегідь і не може бути адаптованим. Коллінс [14] були першими, хто розглядали відстеження, як бінарну проблему класифікації, вони виділили два класи, перший є об'єктом цікавості, другий є фон. Вони використовували само навчальний алгоритм на навчальних прикладах.

У самонавчальних методах присутній процес перенавчання, який використовується, як додатковий параметр для оцінки. Цей параметр є плаваючим [12]. Жавед [25] використовує спільне навчання для того, щоб покращити процес навчання на прикладах в автономному режимі. Він стверджував, що в процесі відстеження об'єкту цікавості за допомогою спільного навчання лежить припущення, що два умовно незалежні зображення одного об’єкта дають різний результат навчання, так як об'єкти навчання відстеження відбираються з однією модальності [27]. Адам [2] запропонував підхід, який називається FragTrack, він використовує статичну частину на основі зовнішнього вигляду моделі, заснованої на інтегральних гістограм. Авідан [5] використовує самонавчання для прискорення оновлення груп класифікаторів. Нрабнет [21] використовує напівсамонавчальний підхід, що забезпечує дотримання попереднього рівня на першому оновлені. Проте, якщо зміни значні, то об'єкт, швидше за все, не буде знайдений знову. Якщо зміни занадто слабкі, то він не усуне перешкод.

Калал [27] запропонував метод TLD (Tracking-Learning-Detection), який використовує зміни знайдені на траєкторії оптичного потоку для того, щоб навчити детектор об'єкта. Оновлення виконуються лише тоді, коли виявляється, що теперішня модель об’єкта цікавості схожа на початкову модель - це відрізняє цей метод від методів адаптивного відстеження та виявлення. Вихідний сигнал детектора об'єкта цікавості використовується тільки для повторної ініціалізації процес відстеження оптичного потоку, але ніколи не використовується для того, щоб оновити сам класифікатор. Калал досяг чудових результатів, а також високої частоти квадрів порівнянні з адаптивними методами відстеження та виявлення.

### Обсяг робіт

У роботі використовується підхід Калала [28] для рекурсивного відстеження. Цей підхід заснований на оцінці оптичного потоку із використанням методу Лукас і Канаде [33]. Для виявлення об'єкта, використовується [26] шаблони об’єкта, які нормуються по яскравості і розміром. Відстеження шаблонів для позитивних прикладів об'єкта і для негативних прикладів, знайдених у фоновому режимі.

Ці шаблони формують основу розпізнавання об'єкта, процес виконується незалежно від процесу спостереження. Нові шаблони створюються за допомогою P / N-навчання [27]. Якщо процес спостереження знаходить розташування об’єкта цікавості на зображенні з високим рівнем схожості з шаблонами, то процес відстеження буде ініціалізуватися заново на цьому місці. Оскільки порівняння шаблонів є складним процесом, тому використовується каскадний підхід до об'єкта цікавості. У роботі [27] випадковий кадр класифікується [38] на основі бінарного шаблону і фіксованого шаблону. Каскад виявлення об'єкта складається: фонового процесу відстеження, дисперсії фільтра, випадкового кадру класифікатора на основі функцій, запропонованих в роботі [31] і методу зіставлення шаблонів. На відміну від Калал у робооті не використовується викривлення зображення для навчання. Рис. 3.3 відображає робочий процес підходу. Ініціалізація призводить до старту навчання. Далі, рекурсивний процес відстеження та процес розпізнавання працюють паралельно, і їх результати зливаються в єдиний кінцевий результат. Якщо цей результат проходить стадію перевірки, шаблон навчання оновлюється. Тоді процес повторюється, що призводить до підвищення точності розпізнавання та відстеження.

Рис 3.3 Процес відстеження ініціалізується вручну (Виділення об’єкта цікавості)

## Відстеження

У цьому розділі буде описано рекурсивний метод для вистежування об'єкта цікавості. У цьому методі немає необхідності в отримані інформації про об'єкт, окрім його розташування в попередньому кадрі, а це значить, що потрібна зовнішня ініціалізації. У підході, ініціалізація здійснюється за допомогою ручного втручання в першому кадрі і за результатами механізму виявлення об'єкта в послідовних кадрах.

Робота слідує підходу Калал [28] для рекурсивного відстеження. Пояснимо цей метод відповідно до рис. 3.4. По-перше, рівномірно розподілимо безліч точок в обмежувальній рамці, яка показана на зображенні зліва. Потім оптичний потік оцінюється для кожної з цих точок, використовуючи метод Лукаса і Канаде [33]. Цей метод працює найбільш надійно, якщо точка знаходиться на кутах [45], але він не в змозі відслідковувати точки на однорідних областях. Використовується інформацію з методу Лукаса і Канаде, а також два різні методи для фільтрації помилок, які засновані на основі нормованої взаємної кореляції та вперед-назад помилки. Якщо медіана всіх вперед-назад оцінок помилок вище певного порогу, то зупиняється рекурсивне відстеження повністю, так як інтерпретує цю подію, як вихід об’єкта цікавості за рамки зору камери. Інші точки використовуються для того, щоб оцінити стан нового обмежуючої рамки в наступному кадрі, використовуючи модель перетворення, заснованого на зміні в повороті і масштабу. У правому зображенні, що обмежує прямокутник з попереднього кадру був перетворений відповідно до векторів переміщення від інших точок.

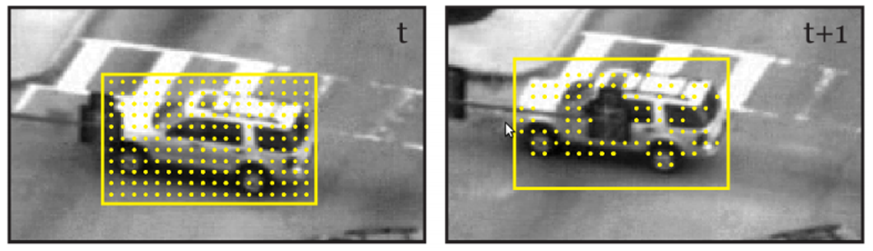


Рис. 3.4 Принцип рекурсивного методу відстеження

### Оцінка оптичного потоку

Лукас і Канаде засновували свій підхід на трьох припущеннях. Перше припущення базується на сталості яскравості[8] і виражається так

I(X) = J(X +d), (3.1)

де I – кадр;

X – розташування пікселя на двовимірному кадрі;

J – інший кадр;

d – вектор зміщення.

Рівняння 3.1 свідчить, що піксель в двовимірному позиції (X) на кадрі I може змінити свою позицію на іншому кадрі J, але він зберігає свою яскравість.

Надалі вектор d буде називатися вектором зміщення. Друге припущення відноситься до [8] тимчасової постійності об’єкта цікавості. У ньому говориться, що вектор зміщення малий – це означає, що J (X) може бути апроксимована

J(X) ≈ I(X) +d, (3.2)

де I – кадр;

X – розташування пікселях на двовимірному кадрі;

J – інший кадр;

d – вектор зміщення;

- є градієнт I в точці X.

Тоді, значення d буде обраховуватися наступним чином

d ≈ (J(X)−I(X)) / (3.3)

Для будь-якого пікселя, рівняння 3.3 є невизначеним, тому простором рішення є лінію замість точки.

Третє припущення, відоме як просторові когерентності, полегшує рішення цієї проблеми. У ньому говориться, що всі пікселі, що лежать у межах вікна навколо пікселя переміщення когерентні. Використавши це припущення, можна знайти d шляхом підрахунку суми найменших квадратів.

, (3.4)

де W – область навколо пікселя.

У роботі [48] показано, що в замкнутій формі рішення для рівняння 3.4 виглядає так.

, (3.5)

де G =; (3.6)

. (3.7)

Додаткова інформація реалізації методу знаходяться в [8].

## Виявлення помилок

Для підвищення надійності рекурсивного відстеження, використовується три критерії для фільтрації точок. Перший критерій встановлюється безпосередньо з рівняння 3.5. З нього випливає, що d можна обчислити тільки тоді, коли G оборотне. G буде вважатися оборотним, якщо він буде мати два великих власних значення (λ1, λ2), які мають місце, коли існують градієнти в два напрямки [8]. Використовується формула Щі і Томасакі [45] в якості першого критерію для фільтрації точок.

min(λ1, λ2) > λ, (3.8)

де λ1 – перше велике власне число;

λ2 – друге велике власне число.

Калал [28] запропонував метод вперед-назад для фільтраціях помилок. Цей підхід до виявлення помилки показано на рис. 3.2. У лівому зображенні, точка 1 правильно відслідковуються, його відповідне положення в правому зображенні це - точка 2, однак вона розташована в неправильному місці, відбувається колізія. Запропонований фільтр помилки базується на ідеї про те, що відстеження точок повинен бути оборотним. Точка 1 відстежується назад у вихідне місце розташування. На відміну від точки 2, яка відстежується назад в іншому місці. Запропоновано міра похибки визначається, як евклідова відстань і вираховується за формулою

, (3.9)

де – модуль різниці

*p* – позиція точки;

(3.10)

де LK – метод Лукаса і Канаде.

У роботі [28] вперед-назад метод використовується в поєднанні з ще одним фільтром заснованим на схожості навколо точки *p* і навколо , Подібність цих двох шматків P1 і P2 порівнюється з використанням нормалізованої кореляції коефіцієнт (НКК) за двома шматками зображення P1 і P2, який визначається так.

(3.11)

де – значення ;

– значення ;

– стандартне відхилення ;

– стандартне відхилення .

## Модель трансформації

Після підходу Калал [28], обчислимо медіану всіх вперед-назад помилки і медіанний всіх заходів подібності і тримати тільки ті точки виставляється вперед-назад помилка менше і міра подібності більше, ніж .

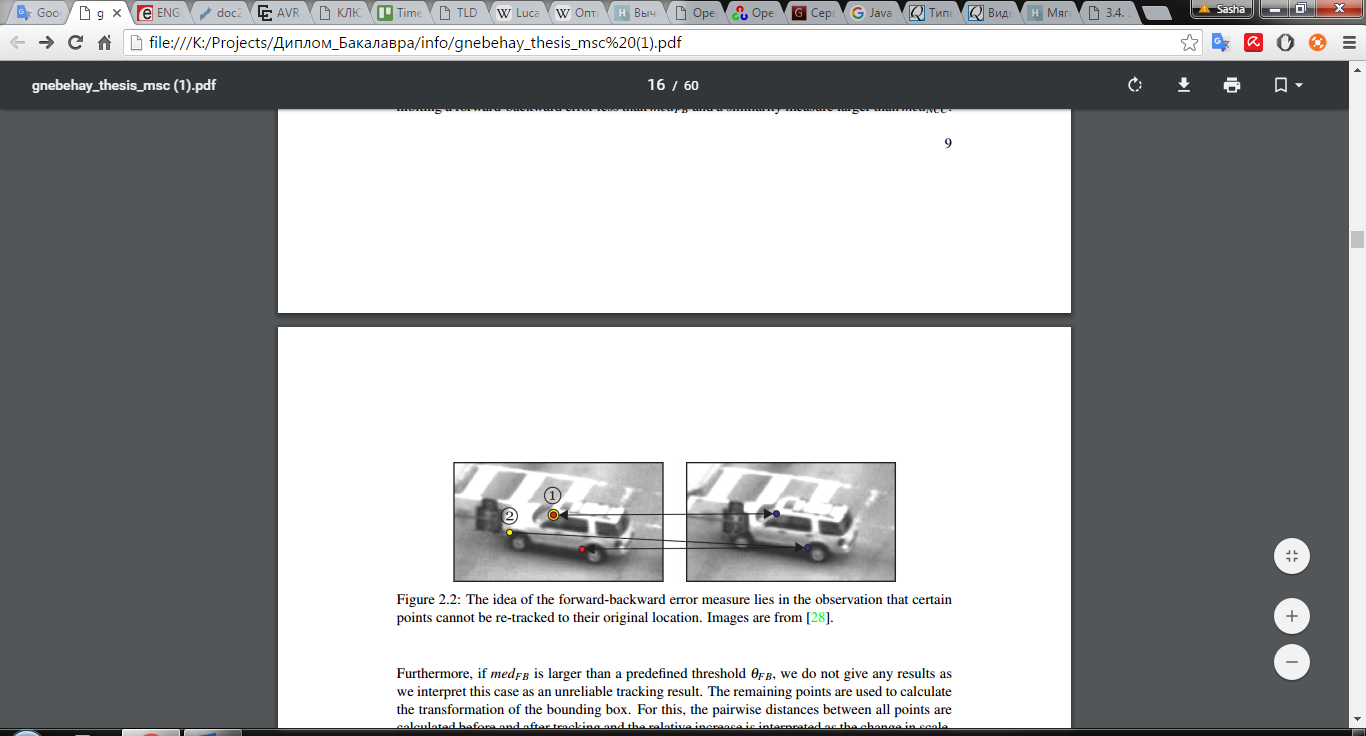


Рис. 3.5 Демонстрації роботи методу вперед-назад

Крім того, якщо більше, ніж заздалегідь визначений граничний , не повертається ніяких результатів, цей випадок інтерпретується, як ненадійний результат відстеження. Решта точок використовуються для розрахунку трансформація обмежувальної рамки. Для цього, попарні відстані між усіма точками розраховується до і після зміни шаблону.

Трансформація у напрямку *х* обчислюється з використанням медіани горизонтальних зрушень всіх точок. Трансформація в напрямку осі *у* розраховується аналогічно. Використовується сітка розміром 10 × 10, розмір вікна W = 10 і Поріг = 10 для всіх експериментів.

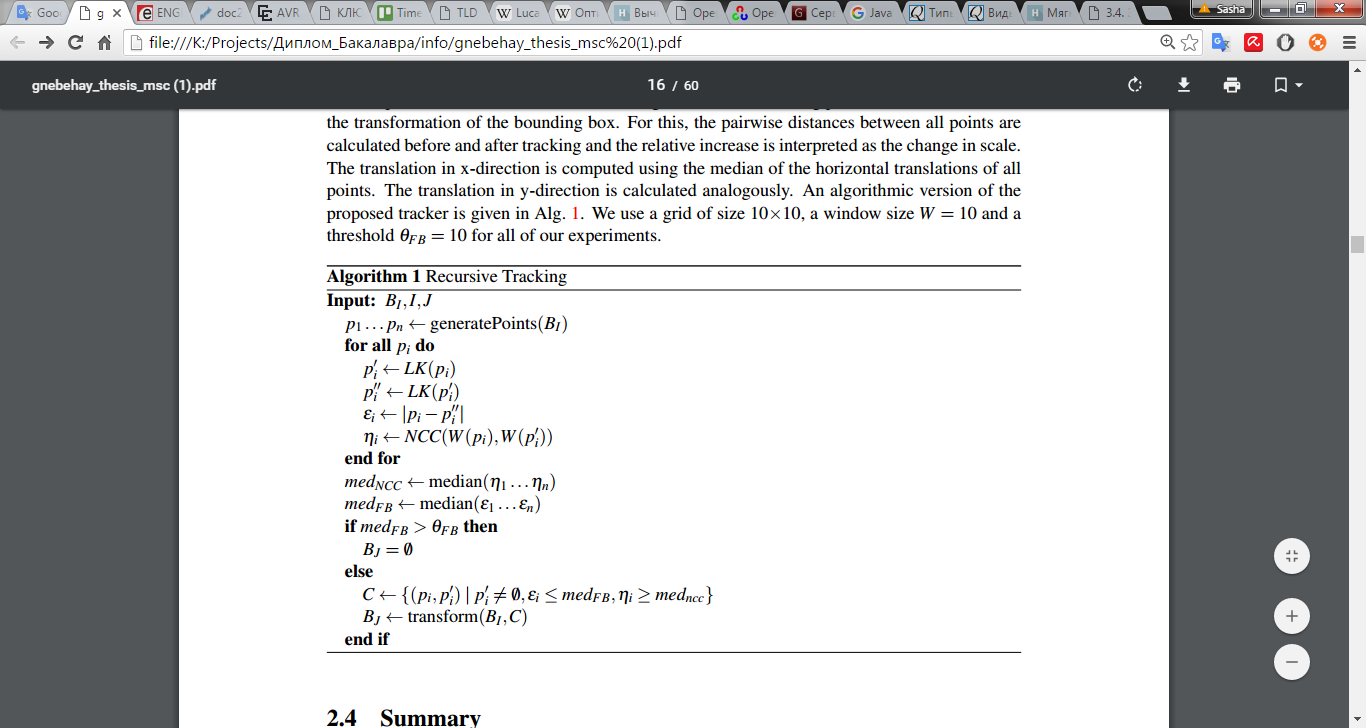


Рис. 3.3 Алгоритм рекурсивного відстеження

## Виявлення

У цьому розділі буде обговорено метод, який був створений для виявлення об'єкта цікавості у полі зору камери. Виявлення об'єктів дозволяє автоматично рекурсивно ініцілізувати процес відстеження, який не підтримує об'єкт моделі, тому він не може відновити свою роботи після на вдалого розпізнання. У той час, як рекурсивний процес відстеження залежить від місця розташування об'єкта цікавості на попередньому кадрі, механізм виявлення об'єкта реалізує пошук по всьому кадру, щоб знайти об'єкт цікавості. Процес виявлення складніший за процес відстеження, тому він займає значну частину об рахункового часу.

Алгоритм заснований на ковзні вікона з об’єктом цікавості [49, 16], цей підхід проілюстровано на рис. 3.6. Зображення у верхній частині оброблює детектор об'єкта, який потім класифікує його. У залежності від розміру початкового об'єкту, використовується від 50000 до 200000 підвікон в форматі VGA (640 × 480). Кожне підвікно перевіряється незалежно один від одного на наявність об'єкту цікавості. Якщо під вікно відповідає класифікатору то воно просувається на один каскад далі.

Каскадні детектори об'єктів спрямовані на відкидання, як найбільше непідходящих вікон та мінімізацію обчислень [43]. Використовується чотири етапи для класифікації зображень. Нижче наведені вхідні зображення. По-перше, використовується метод вирахування фону для того, щоб обмежити простір пошуку (Тільки для областей переднього плану). Цей етап вимагає модель фону, якщо під вікно відповідає моделі то воно пропускається далі. На другому етапі відкидаються всі під вікна, які демонструють дисперсію нижче певного порогового рівня. Третій етап включає в себе ансамбль класифікатор на основі алгоритму рандомні папороті [38]. Четвертий етап складається з методу узгодження шаблону, який заснований на нормалізованому коефіцієнту кореляції в якості міри схожості.



Рис. 3.6 Алгоритм пошуку підходящого під вікна

### Алгоритм вікна

В алгоритмі ковзаючих вікон для виявлення об'єктів цікавості, під зображення вхідного зображення тестуються на наявність об’єкту цікавості [29]. Потенційно, кожне під вікно може в містити зображення, яке місить об'єкт цікавості. В VGA зображенні вже є 23,507,020,800 можливих під вікон і кількість можливих під вікон зростає за формулою для зображень розміром n×n. Було розбито простір пошуку на підпростори *R* і використали такі обмеження. По-перше, припускається, що об'єкт цікавості зберігає свої пропорції. Крім того, вводяться параметри і між двома сусідніми підвікнами і встановлюємо для і значення от *n*. використовується коефіцієнт масштабування s = , *a* ∈ {-10 ... 10} для ініціалізуючого прямокутника об'єкта цікавості. Також у роботі вважається, що під вікно має мінімальну площу 25 пікселів.

Розмір всіх під вікон *R* вираховується таким чином,

(3.12)

де n – ширина вікна;

w = 80 – ширина під вікна;

m – висота зображення;

h = 60 – висота під вікна.

У рівнянні 3.12 *w* і *h* означають розмір ініціалізуючого прямокутника, *n* і *m* – ширина та висота зображення. Для початкового кадрування розмір w = 80 і h = 60, а кількість під вікон в VGA відповідно 14619, так як кожне під вікно перевіряється незалежно один від одного, використовується стільки потоків, скільки ядер доступно в системі.

### Знаходження переднього плану

Один з підходів для виявлення рухомих об'єктів у відео потоці є відокремлення фону, де кожен відеокадр фону порівнюється з моделлю фону [13]. У цьому розділі буде описано, яким чином модель фону прискорює процес виявлення об’єкта цікавості. У роботі виокремлюється передній план від фону в чотири етапи, як це зображено на рис. 3.7. На цьому малюнку, праве верхнє зображення є фонове зображення , а з верху з ліва зображення *I*, яке є фоном для виявлення. Алгоритм починає з вирахування абсолютної різниці і *I*

(3.13)

У результаті рівняння 3.13 отримуємо зображення другого ряду. Тепер буде застосовуватися поріг в 16 пікселів для того, щоб створити бінарний образ зображення, який показаний у другому зображенні другого ряду.

(3.14)

де – функція знаходження значення пікселя.

Далі будуть використовуватися білі пікселів в якості компонентів об’єкта цікавості. Для того щоб обчислити площу і побудувати обмежувальний прямокутник, у роботі застосується алгоритм маркування мітками запропонований в роботі [11]. Цей алгоритм обчислює мітки зображення в один прохід.

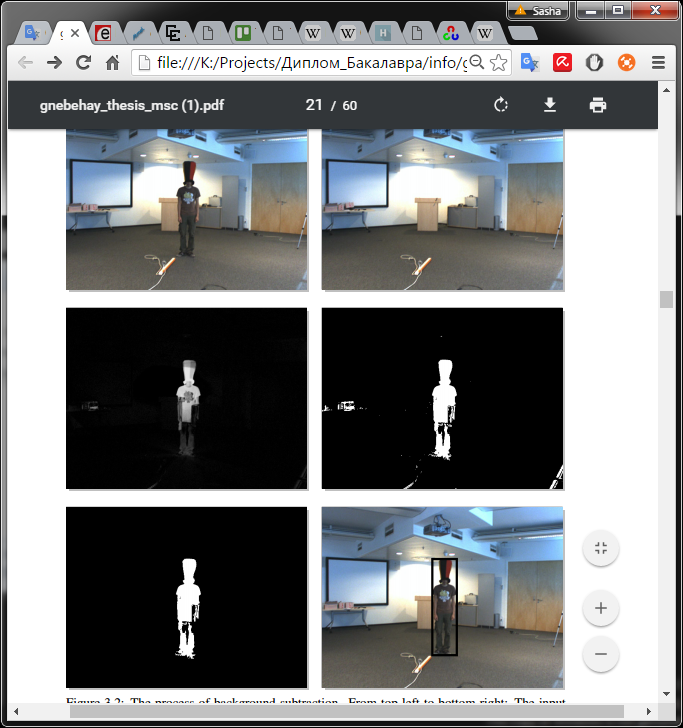


Рис. 3.7 Алгоритм виділення об’єкта на передньому плані

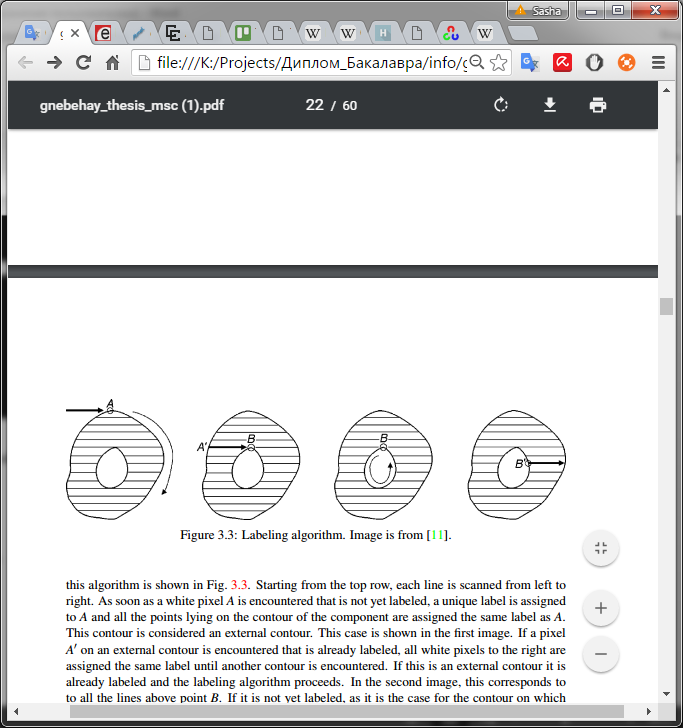


Рис. 3.8 Принцип алгоритму виділення об’єкта на передньому плані

Принцип роботи алгоритму наведений на рис. 3.8. Починаючи з верхнього ряду, кожен рядок сканується зліва на право. Як тільки зустрічається білий піксель *А*, який ще не пов’язаний, йому присвоюється унікальна мітка *А.* Для всіх точок, що лежать на контурі компонента привласнюється одна і таж мітка А.

Цей контур вважається зовнішнім контуром. Цей випадок показаний на першому зображенні. Якщо піксель на зовнішньому контурі зустрічається, як вже помічений, то всі білі пікселя з права привласнюється такий же ярлик, поки інший контур не зустрінеться. На другому зображенні виділяється внутрішній контур - мітка *B*. Якщо піксель ще не позначений, і якщо він лежить на контурі, то всі пікселя цього контура отримують мітку *B*. Цей випадок показаний на третьому зображенні. Якщо позначена внутрішня точка контуру зустрічається, то всім наступним білим пікселям присвоюються мітка *A*. Цей випадок показаний на четвертий зображення. Координати для найменшого обмежувального паралелепіпеда визначається за координатами крайніх пікселів компонента. Область компонента є сумою всіх білих пікселів.

Повертаючись до рис. 3.7, алгоритм видаляє всі компоненти з бінарного зображення з областю менше, ніж розмір початкової обмежувальної рамки. Результат цієї операції показаний на першому зображенні третього ряду. Всі під вікна відкидаються, які не в повній мірі знаходяться всередині однієї з найменших обмежувальних рамок навколо інших компонентів. У роботі цей набір обмежувальні називається рамки *C*. Якщо немає фонового зображення, то всі під вікна приймаються.

### Фільтр по дисперсії

Дисперсія зображення є мірою його однорідності. На рис. 3.9 показані два зразка під вікон, червоним кольором відмічені ті під вікна, які оцінюються в однорідних фонових районах. Обидва під вікна показують дисперсію нижче, ніж ділянки об'єкта, обраного для спостереження, який міститься в правому зеленому прямокутнику. У цьому розділі описано механізм, який обчислює дисперсію під вікна за допомогою інтегральних зображень і яким відкидає під вікна, які містять дисперсію нижче порогового. Такий фільтр здатний швидко розрівняти під вікна з однорідним фоном та з неоднорідним. Наприклад, ліве зелене під вікно обмежувальної рамки на рис. 3.9 буде прийняте.

У роботі використовується механізм для того, щоб обчислити дисперсію, який показано в роботі [49]. Для того щоб спростити наступне пояснення, під вікна зображення визначається обмежувальним блоком *В*.

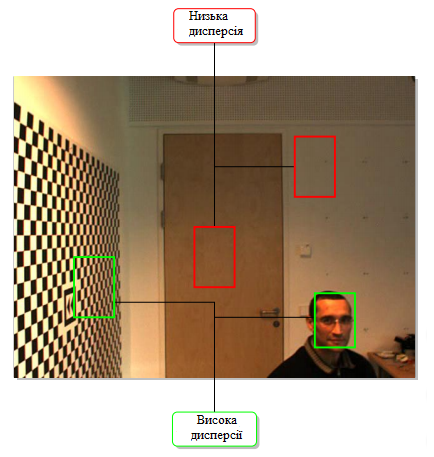


Рис. 3.9 Виявлення дисперсії під вікон

Під вікно розглядається, як одновимірний вектор пікселів, а його елементи адресуються з використанням анотації для *i*-го пікселя. Для зображення дисперсія визначається так.

, (3.15)

де n - число пікселів в зображенні;

- *i* піксель;

. (3.16)

Альтернативне представлення цієї формули.

(3.17)

Для того щоб обчислити за формулою 3.17 для зображення під вікна розміром n, необхідно n-пам'яті. Користуючись тим, що дві пересічні ділянки зображення частково поділяю одні й ті ж значення пікселів, у роботі продемонстровано спосіб обчислення для зображення, який використовує тільки 8 звернень до пам'яті після перетворення вхідного зображення *I* на два інтегральних зображення. Інтегральне зображення має такий же розмір, як я *І* і містить в місці (х, у) суму всіх значень пікселів між точками (1,1) і (х, у). Це може бути сформульована як.

, (3.18)

де x, y – координати на зображені *І*;

– координати на зображені .

Інтегральне зображення обчислюється за один прохід, використовуючи той факт, що я можна розкласти.

, (3.19)

У рівнянні. 3.19 при х = 0 або у = 0. Використовуючи інтегральне представлення зображення, обчислення суми пікселів перестає мати зміст до певної точки. На рис. 3.10 , сума значень пікселів у межах прямокутника ABCD обчислюється наступним чином. По-перше, обчислюється сума всіх пікселів між точкою (0,0) та точкою D. Далі, пікселі в області між (0,0) і B віднімаються, а також пікселі в області між і (0,0) і C. Область між (0,0) і А треба доданий ще раз, так як вона віднімається два рази. Використовуючи це зауваження, формулу для обчислення суми пікселів в межах рамки B задають параметри (*х, у, w, h*).

, (3.20)

Як умовне позначення для рівняння. 3.20 використаємо наступне представлення.

, (3.21)

У роботі використовується рівняння. 3.21 для того, щоб обчислити *µ* у формулі 3.17. Щоб обчислити перший член правої частини цього рівняння, використовуючи інтегральні образи, тому необхідно змінити рівняння 3.18. У роботі використовується зведене в квадрат значення I (х, у).

. (3.22)

За аналогією з рівнянням 3.21 було переписано.

. (3.23)

Об'єднавши рівняння. 3.16, рівняння. 3.17, рівняння. 3.21 і рівняння. 3.23, було отримано.

. (3.23)

Ця формула дозволяє обчислювати за допомогою восьми звернень до пам'яті.

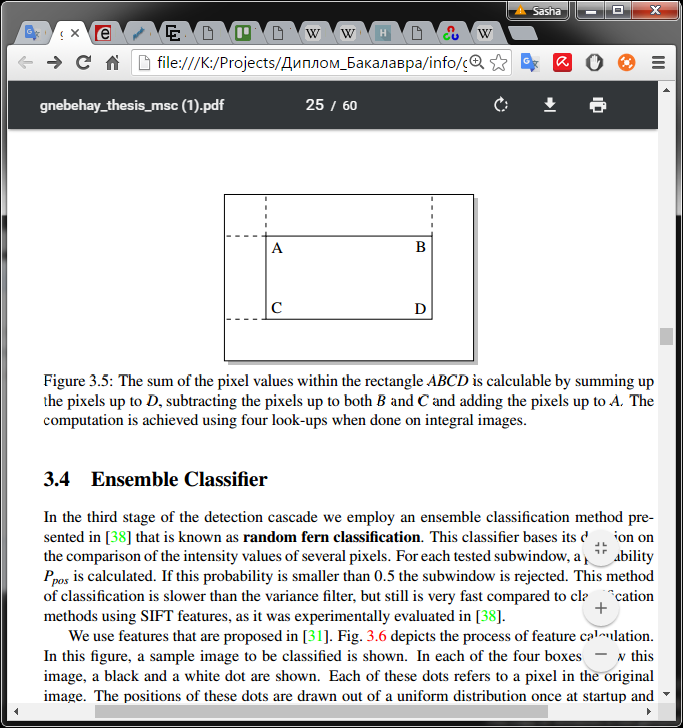


Рис. 3.10 Розрахування дисперсії під вікна

### Класифікатор

На третьому етапі виявлення використовується метод класифікації, представлений в роботі [38], який відомий під назвою випадкова класифікація папороті. Цей класифікатор заснований на порівнянні значень інтенсивності декількох пікселів. Для кожного тестового під вікна розраховується ймовірність . Якщо ця ймовірність менше, ніж 0.5 під вікно відхиляється. Цей метод класифікації повільніший за фільтр дисперсії, але швидший у порівнянні з класифікаційним методом із використанням SIFT функції, що було експериментально доведено в роботі [38].

У роботі використовується функції, які запропоновані в роботі [31]. Рис. 3.11 демонструє процес розрахунку просторових об'єктів. Саме верхнє зображення рис. 3.11 – це початкове під вікно. У кожній з чотирьох рамок нижче чорно-білі точки відноситься до пікселів в оригінальному зображенні. Положення цих точок фіксується за рівномірним розподілом, один раз при старті і залишаються незмінними. Для кожного з цих рамок перевіримо чи є в оригінальному зображенні пікселі в розташуванні - піксель білої крапка світліший за піксель чорної крапки. Математично висловлюється це так.

, (3.24)

У рівнянні 3.24 і - це два випадкових позиції. У результаті кожний з цих порівнянь інтерпретується, як двійковий розряд і всі ці значення об'єднуються у вигляді двійкового числа. На рис. 3.12 у результатом є двійкове число 1101. У десяткового формі це - 13. Функція *Ith* визначає значення *i*-го біта. Алгоритмічний варіант такого розрахунку показаний на рис.3.11 у якому *I* - це вхідне зображення, *F* - розрахункове значення функції і *S* - це кількість функцій, які будуть використовуватися. Значення *S* впливає на максимальне значення ознаки підходящої під вікна, яку може бути *2S - 1*. Значення функція використовується для отримати ймовірність *Р (у = 1 | F),* де *у = 1* відноситься до події, де під вікно проходить перевірку.

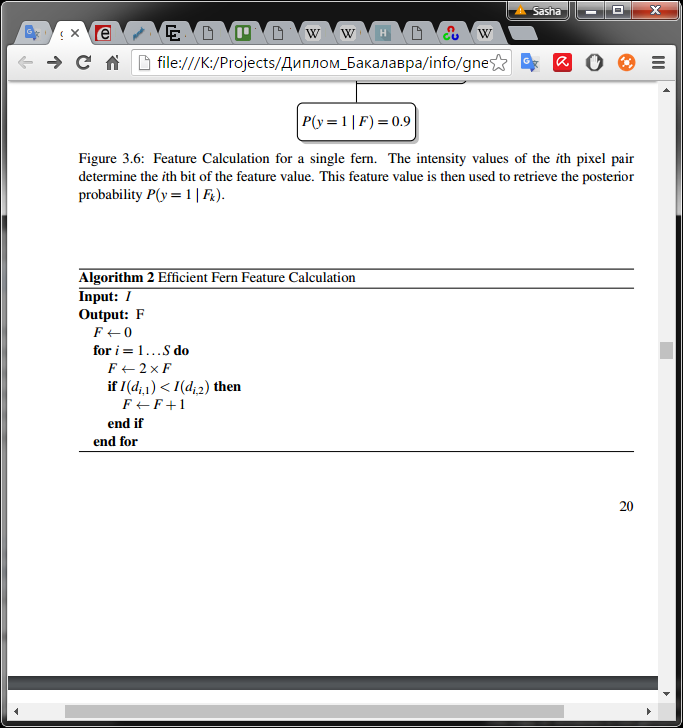


Рис 3.11 Алгоритм класифікатора

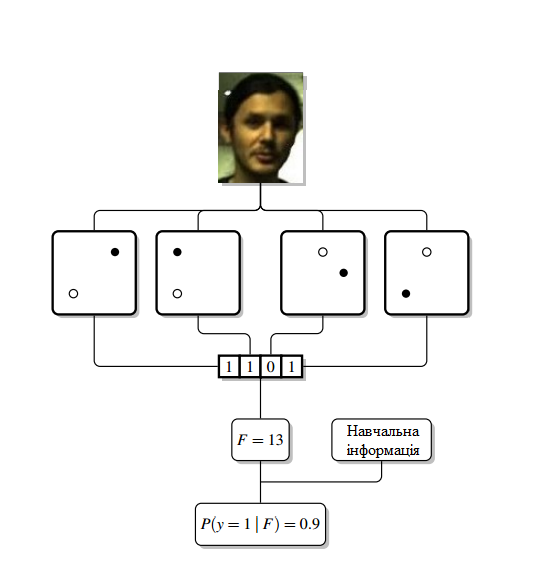


Рис 3.12 Функція розрахунку однієї папороті

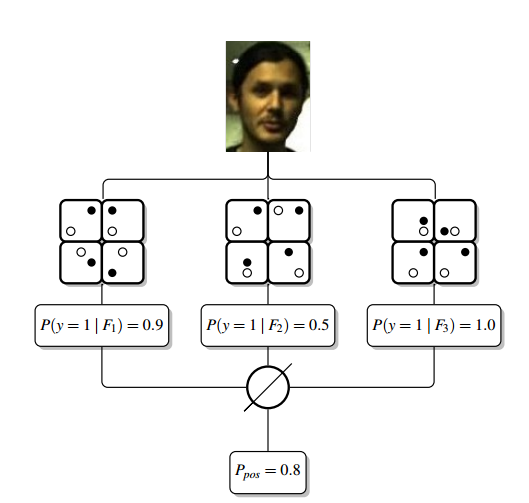


Рис 3.13 Класифікації із використанням трьох випадкових папоротей

При наявності тільки однієї папороті, необхідно використовувати більшу кількість функцій для досягнення задовольняє мого результату [38]. Однак кількість навчальних даних, необхідних для оцінки *P (Y = 1 | Fk)* збільшується з кожною додатковою функцією. Ця проблема відома, як “прокляття розмірності” [36]. Аміт і Герман [3] зіткнулися з тією ж проблемою при використанні алгоритму рандомізованих дерев для розпізнавання, для того щоб полегшити розрахунки вони не використовували одне велике дерево, а використали кілька маленьких дерев. Потім вони порахували їх середнє значення. Це рішення було використано в [38] і використовується до класифікації зображення на рис. 3.15. Для класифікації використовується три папороті, кожна з яких складається з різних наборів художніх позицій і кожна з яких забезпечує різні значення для P (Y = 1 | ). Середнє значень рахується так.

, (3.25)

де М - число папоротей, які використовуються.

### Узгодження шаблонів

На четвертому етапі каскаду детектора використовується метод, узгодження шаблонів. Цей етап має більш обмежувальний характер, ніж ансамбль методу класифікації, описаної в попередньому розділі, так як виконується порівняння пікселя за пікселем. Було змінено розмір всіх під вікон до 15 × 15 пікселів. Для порівняння двох під вікон , використовується нормалізований коефіцієнт кореляції (НКК).

, (3.26)

де , - значення ;

і - стандартні відхилення .

Це відстань також відома, як коефіцієнт Пірсона [40]. Якщо інтерпретувати геометрично, то його можна позначити косинусом кута між двома нормалізованими векторами [10]. НКК повертає значення між -1 та 1, якщо значенням ближчу до 1, то два шляхи однакові. У роботі використовується така формула для того, щоб визначити відстань між двома ділянками, які дають значення від 0 до 1.

. (3.27)

У роботі підтримується шаблони для позитивних і негативних класів. Позначається позитивний класу, як , а негативний класу, як . Позитивні та негативні приклади були засвоєні на послідовності повороту обличчя. Для не класифікованого зображення *P* розраховується значення того, що воно відноситься до позитивного класу

, (3.27)

і коефіцієнт, що воно відноситься до негативного класу

. (3.28)

На рис. 3.15 зеленим точкам відповідають позитивні випадки, а червоним точкам відповідають негативні екземпляри. Чорна точка, яка позначена знаком питання відповідає під вікну з невідомою міткою. Відстань до найближчого позитивного класу відповідно до рівняння 3.27 є = 0,1 і відстань до найближчого негативного класу, відповідно до формули 3.28 = 0,4. У роботі зливається ці відстані в одне значення за допомогою цієї формули.

, (3.29)

Рівняння 3.29 встановлює чи належить під вікно до позитивного класу. Якщо більше, ніж порогове , то підвікно приймається. Значення вище цього порогу вказує на те, що клас належить до позитивних. У роботі використовується значення = 0,65 для всіх експериментів.

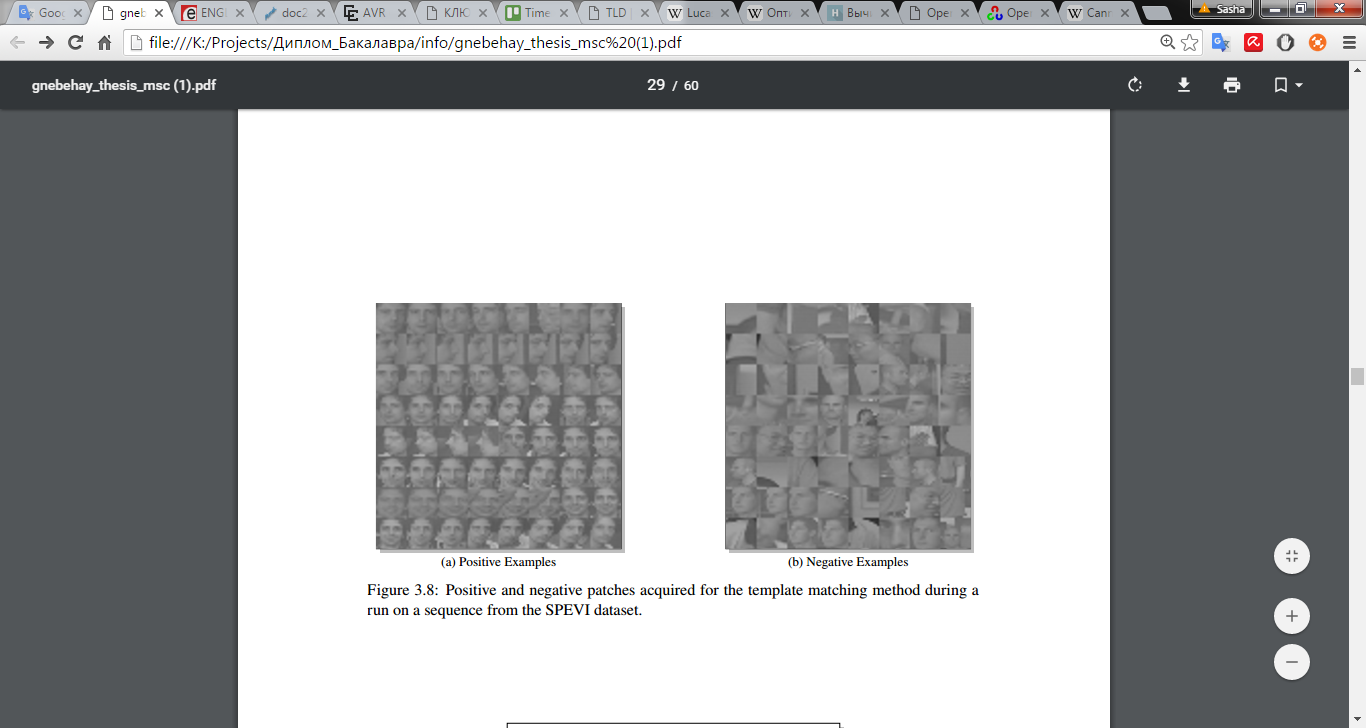


Рис. 3.14 Позитивні і негативні класи

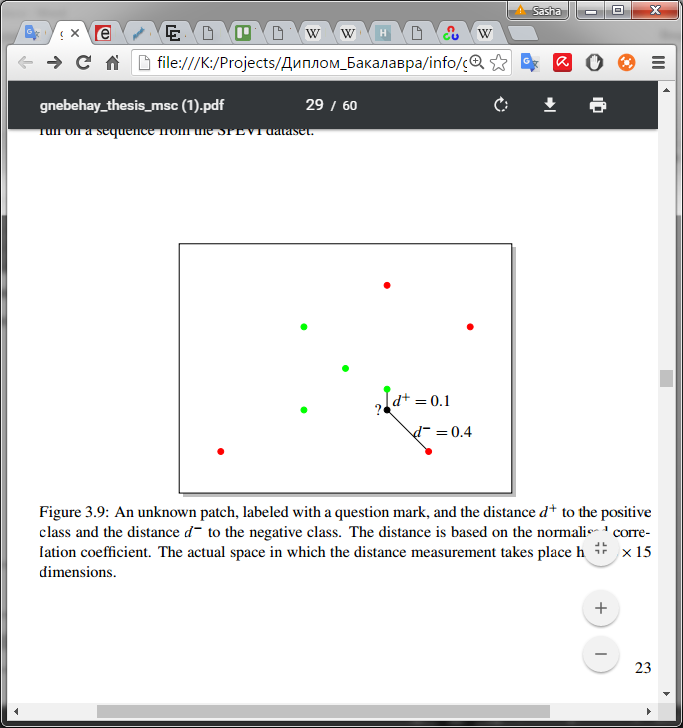


Рис. 3.15 Класифікація класу до позитивних чи негативних

## Самонавчання

При обробці зображення рекурсивний трекер і детектор об'єкта цікавості працюють паралельно. У цьому розділі будо обговорено проблему поєднання виходу з обох методів у єдиний кінцевий результат. Модель фону і поріг для фільтра дисперсії не пристосовуються під час обробки, у той час як класифікатор і метод узгодження шаблонів навчаються під час роботи. У роботі вирішено проблему оновлення шаблонів, за допомогою визначених критерій, які повинні бути виконані для того, щоб розглядати кінцевий результат, придатний для навчання. У процесі навчання, проводиться два обмеження для P/N-навчання[27]. Перше обмеження вимагає, щоб всі ділянки в безпосередній близькості від остаточного результату були класифікованими позитивно детектором об'єкта. Друге обмеження вимагає, щоб всі інші ділянки були класифіковані негативно детектором об'єкта.

### Поєднання результатів

На рис. 3.16 продемонстрована роботу алгоритму для поєднання результатів рекурсивного трекером та виявлено об’єкта у кінцевому результаті Рішення базується на кількості виявлень та впевненості в їх значенні і на довірі до результату відстеження . Останню отримують шляхом запуску методу узгодження шаблону на результат відстеження. Якщо детектор дає рівно один результат з упевненістю вище, ніж результат від рекурсивного трекера, то відгук детектора присвоюється кінцевому результату. Це відповідає пере ініціалізація рекурсивного трекера. Якщо рекурсивний трекер повертає результат і повторно не ініціалізований за допомогою детектора, оскільки існує більше одного виявлення або існує рівно одне виявлення, яке має мітку впевненості меншу ніж відстужувальний об’єкт, результатом рекурсивної стеження призначається остаточний результат. У всіх інших випадках кінцевий результат залишається порожнім, що свідчить про те, що об'єкт не видно в поточному кадрі.

У роботі використовується предикат , щоб висловити високу ступінь впевненості, що кінцевий результат є правильним. Тільки якщо кінцевий результат справедливий для етапу навчання, який буде описано в наступному розділі. В алгоритмі кінцевий результат справедливий при наступних двох обставинах, обидва з яких припускають, що трекер повторно не ініціалізований за допомогою детектора. Остаточний результат справедливий, якщо рекурсивний трекер поверне результат зі значенням впевненості більше ніж в . Кінцевий результат справедливий, якщо попередній результат був дійсним і рекурсивний трекер повернув результат з упевненістю більше, ніж У всіх інших випадках, кінцевий результат є не дійсним. Перший обмежувальна рамка завжди справедлива. Як вже зазначалося поріг вказує, що результат належить до позитивного класу. Поріг вказує на те, що результат належить до негативного класу і фіксується при =0,5 для всіх експериментів.

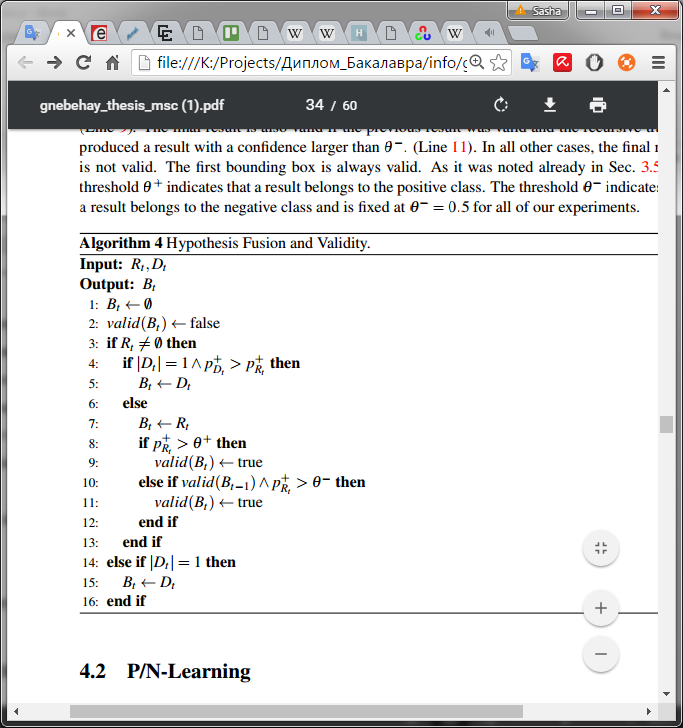


Рис. 3.16 Алгоритм поєднання результатів

### P/N – Навчання

Згідно Шапеля [12], існують два принципово різних типи завдань для машинного навчання. У навчанні з вчителем навчальний набір створюється і розділяється на класи вручну, по суті є набором пар тест, відповідь. Навчальний набір використовується для виведення функцію *f: X → Y*, яка потім використовується для класифікації даних. Методи навчанні з вчителем були успішно використані в області виявлення особи [49], виявлення пішоходів [16]. Але метод навчання з вчителем не може адаптуватися до змін у розподілі даних. Другим завданням в області машинного навчання є самостійне навчання. У цьому випадку жоден клас має зазеленіть визначеної мітки, тому завдання алгоритмам розбити ці данні на класи, що досягається за рахунок оцінки, каутеризації, виявлення відхилень значень і зниження розмірності [12].

Між цими двома парадигмами є напівсамостійне навчання. У напівсамостійному навчанні є визначені приклади, а також не марковані дані. Один тип напівсамостійного навчання використовує інформацію присутню в навчальних даних в якості наглядової інформації [12] для того, щоб класифікувати марковані дані і оновити класифікатор за допомогою цих класів.

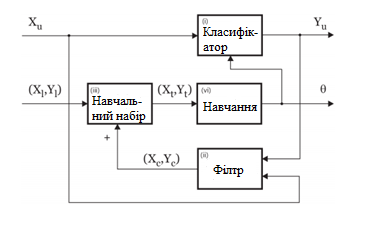


Рис 3.17 P/N Обмеження. Зображення з [27]

У алгоритмі відстеження є рівно один промаркований приклад. У роботі [27], напівсамостійний метод навчання називається P/N-навчання. Цей метод показує, як структурні обмеження можуть отримувати дані з навчальних не промаркованих даних для бінарної класифікації.

У P/N-навчанні існує два типи обмежень: P-обмеження ідентифікує помилкові негативні результати виходу і додає їх в якості позитивних прикладів для навчання. N-обмеження робить протилежне. Цей процес зображений на рис. 3.17, - відноситься до не маркованих даних. Ці дані спочатку класифікуються існуючим класифікатором, який присвоює марки . Потім, структурні обмеження виявляють приклади, які неправильно класифіковані їм встановлюється нові етикетки . Ці приклади потім додаються до навчального набору і вони виконується для оновлення класифікаційної функція.

У роботі використовується такі обмеження для виявлення об'єктів, які пропонуються в [27]. P-обмеження вимагає, щоб всі під вікна, які сильно перекриваються з кінцевим результатом, мають бути класифікованими, як позитивні приклади. N-обмеження вимагає, щоб всі ділянки, які не перекриваються з дійсним кінцевим результатом повинні бути класифіковані, як негативні приклади. Вважається, що обмежувальна рамка *B* сильно перекривається з якщо він перекриває не менше ніж 60%, якщо перекриття менше, ніж 20%, то вважається, що *B* не перекривається . Тепер буде описано заходи, які приймаються для того, щоб адаптувати класифікатор і узгодження шаблонів до того, щоб правильно класифікувати приклади. Нагадаємо, що *Р(у = 1 | )* є ймовірністю того чи є позитивні під вікна, враховуючи особливості .

, (3.30)

де – кількість застосування Р-обмеження;

- кількість застосування N-обмеження.

У алгоритмі перевіряється чи є рамка розміру перекривається з кінцевим результатом. Збільшується на 1 для кожного папороті, якщо перекриття менше, ніж 0,6 і ансамбль класифікатор дав результат нижче 0,5. Алгоритм збільшує на 1 на неправильній класифікації негативного під вікна. При оновленні класифікатора обчислювальні витрати не збільшується. Це відрізняє методу узгодження шаблону, оскільки кожен додаткове під вікно в безлічі позитивних або негативних шаблонах збільшує кількість порівнянь, які повинні бути зроблені для того, щоб класифікувати новий клас. Для того щоб змінити мітку неправильно класифікованого позитивного під вікна використовується шаблон зіставлення методу, додається його до безлічі позитивних шаблонів. Це під вікно потім має відстань =0, що означає, що її довіра дорівнює 1. Зверніть увагу, що етап навчання виконується тільки тоді, коли кінцевий результат справедливий, що вже має на увазі, що більше, ніж . Що стосується N-обмеження для методу узгодження шаблону, додається негативні під вікна до методу узгодження шаблонів, якщо вони були неправильно класифіковані за класифікатором, а також є неправильно класифікованим за шаблоном методом узгодження.

# ВИСНОВОК ДО РОЗДІЛУ 3

У цій роботі представлено реалізацію нового підходу до надійного відстеження об'єкта, заснованого на алгоритмі Відстеження-Навчання-Розпізнавання. Було зроблено наступні модифікації: Було відтворено результати Калал і доведено, що використання функцій, заснованих на парному порівнянні пікселів і двох додаткових стадій в каскаді призводять до скорочення часу обчислень і не погіршують результату. У реалізації роботу зменшено час обчислення в три рази. У роботі було продемонстровано, що застосування емпіричного підходу до сценаріїв кількох камер можлива доти, поки умови освітлення і орієнтації камер залишаються аналогічними.

Очевидно, що підхід багато в чому залежить від якості результатів, що постачаються рекурсивним трекером. У принципі, якість результатів може бути поліпшена двома способами. По-перше це – проміжки часу під час, яких трекер стежить за об'єктом цікавості може бути збільшена. По-друге, автоматичне виявлення відмов стеження може бути поліпшена, що збільшить ефективність роботи детектора з дрейфуючими об'єктами.

Одна з проблем, яка була виявлена ​​в ході експериментів є те, що детектор об'єкта не може розрізнити об'єкти, які мають схожий зовнішній вигляд. Ця проблема частково викликана тим, що порівняння шаблонів проводиться для зображень зменшеного розміру. Одним із шляхів вирішення цієї проблеми могло б стати збільшення розміру зображення шаблону, але це призведе до прокляття розмірності.

Використання обмежують рамки – це зручно для реалізації, але має свої недоліки. Так як обмежувальні рамки завжди покривають прямокутну область навколо об'єкта, вони частково можуть містити фон. У роботі призначається клас мітки на рівні обмежувальної рамки, що викликає появу фону, що буде розглядатися, як частина об'єкта, що цікавить. Це призводить до проблеми, що об’єкт цікавості не розпізнається, коли він з'являється на іншому тлі. Для того, щоб відокремити об'єкт, що цікавить від фону в обмежувальної рамки для навчання, можна використовувати методи сегментації, такі методи наведені в роботі [19].

У даний час підхід дає інформацію про місцезнаходження об'єкта цікавості, але не про його орієнтації. Інформація про орієнтацію об'єктів може бути відновлена ​​за допомогою використовуючи афінської моделі перетворення для трекера Лукаса-Канаде.

Серйозна невирішена проблема полягає в тому, що детектор не може розпізнати зміни зовнішнього вигляду, які відбуваються в той час як трекер не активний. У роботі [38], зображення викривлення застосовується до прикладів навчання для досягнення інваріантності до афінського перетворення. Однак афінське перетворення не охоплюють зміни локального освітлення або точки зору. Ці зміни часто відбуваються в сценаріях з декількома камерами і викликані різними умовами освітлення і точок зору камер.

# РОЗДІЛ 4 ЕКСПЛУАТАЦІЯ ПРИСТРОЮ

У цьому розділі буде продемонстрована робота із пристроєм. Буде продемонстровано його можливості та порядок роботи з ним. Також буде розглянуто удосконалення та робота приладу в зв’язці із передаючою звуку через людський череп.

Сам прилад складається із двох частин. Перша частина – це фізичний подовжувач. Друга частина – це програмне забезпечення.

Також було модифіковано систему живлення смартфона шляхом підключення додаткової батареї на 4000mAh до каналу живлення приладу, що дало можливість збільшити час роботи в 2.27 рази.

Встановлена батарея оснащена контролером, який запобігаю перезаряджанню смартфона та зниженню заряду до критичної відмітки. На рис. 4.1 демонструється весь прилад.



Рис. 4.1 Перша частина приладу

## Демонстрація роботи

У цьому розділі буде продемонстровано роботу приладу та програмного забезпечення. Для роботи пристрою необхідно підключити подовжувач та батарею (Опціонально). При запуску програми відкриється вікно стеження рис. 4.2.



Рис. 4.2 Запуск програми

У якості об’єкта цікавості я обрав карту. Для встановлення карти програмі, як об’єкта цікавості. Я виділяю її зеленою рамкою і тисну кнопку “Відстежити”. Після чого запускається алгоритм відстеження рис. 4.3.

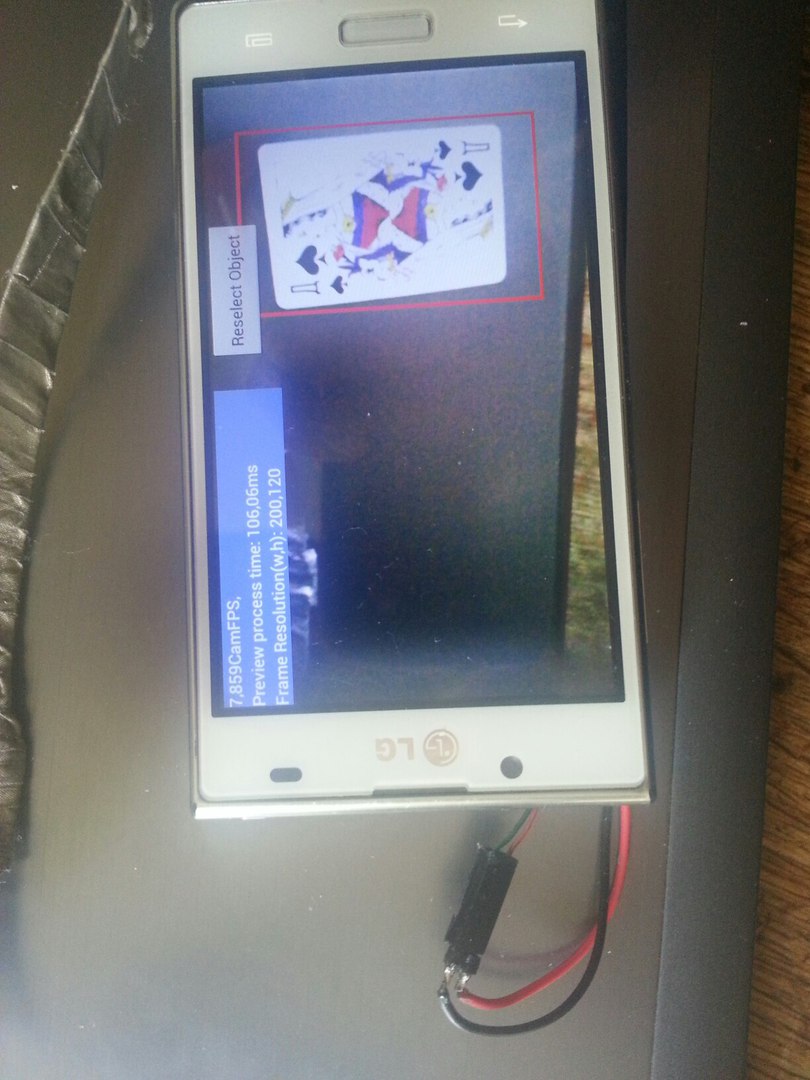
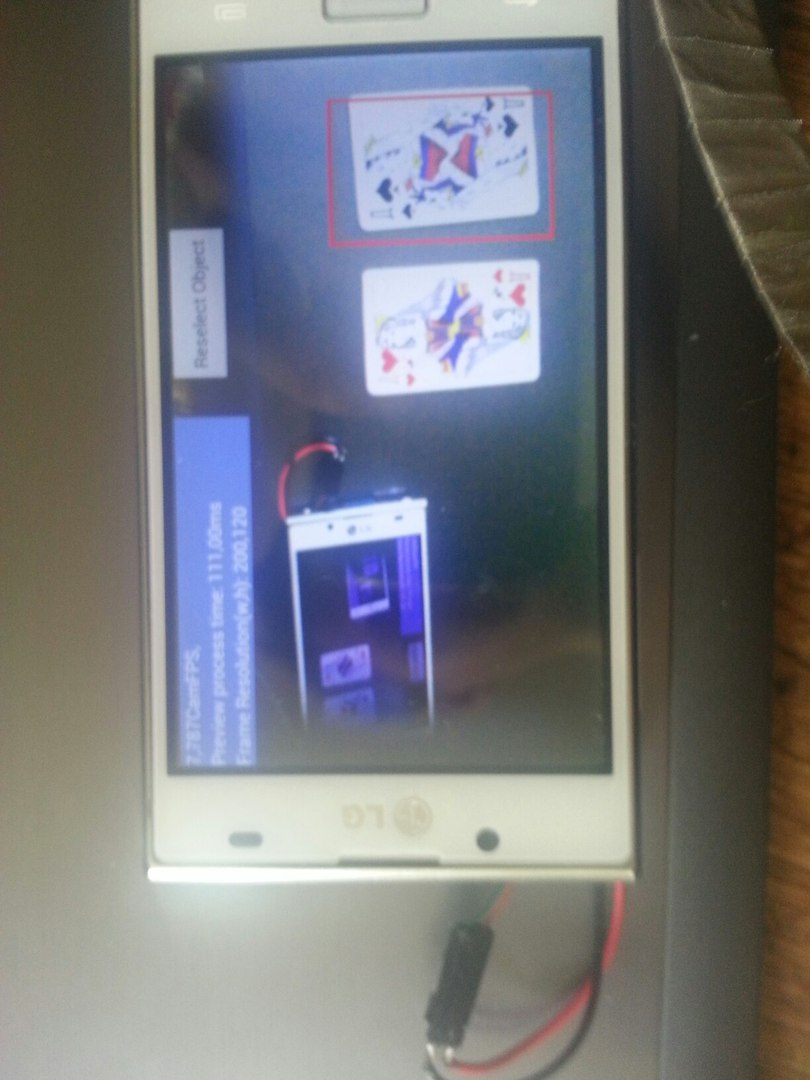


Рис. 4.3 Робота алгоритму

У 3 розділі було описано проблему розпізнання схожих об’єктів. На рис. 4.4 демонструється успішне розпізнавання та слідкування об’єкта цікавості після процесу навчання алгоритму.



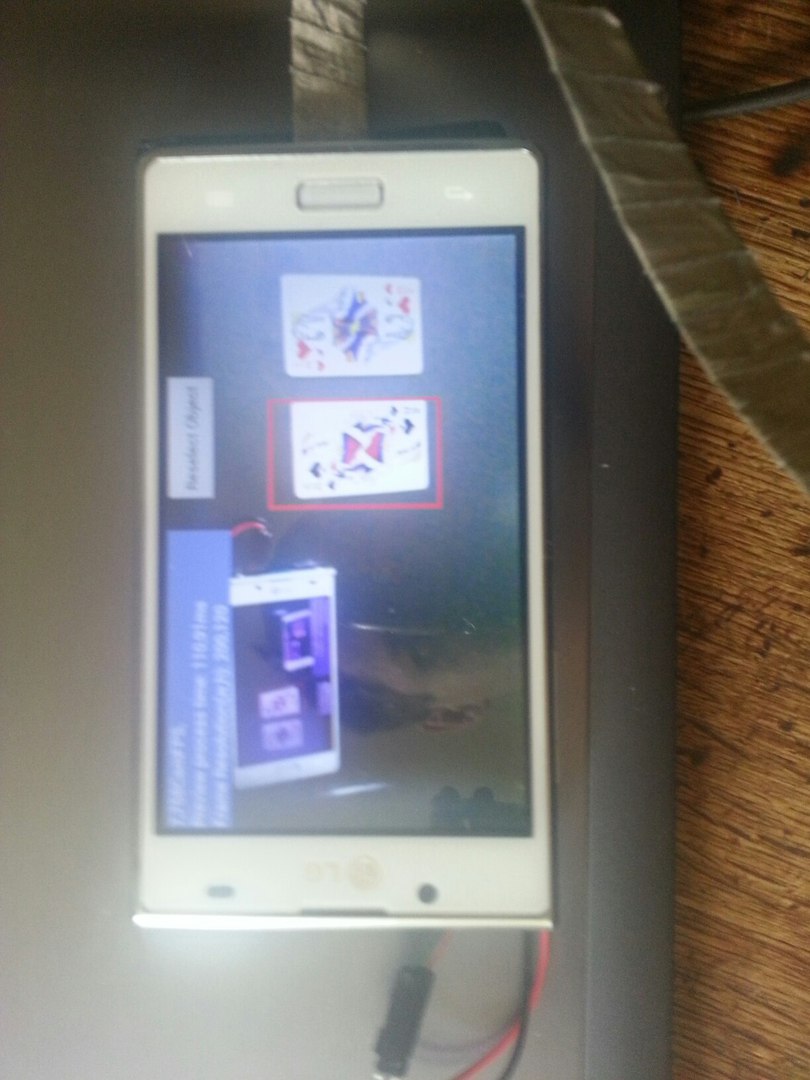


Рис 4.4 Успішне розпізнання об’єкта цікавості

# ВИСНОВОК ДО РОЗДІЛУ 4

Робота приладу дає можливість відстежити та розпізнати у важкодоступному середовище об’єкт цікавості. Також можливі удосконалення приладу. Можна адаптувати пристрій для людей, які позбавлені можливості чути або говорити. Особливість розробленого методу полягає у його здібності до самонавчання, що робить можливим удосконалити алгоритм для розпізнавання жестів і автоматичного підлатування під жести людини, що призведе до підвищення розпізнання.

Мається на увазі використати прилад для перетворення мови жестів в голосовий сигнал, щоб людина яка не розуміє мову жестів змогла зрозуміти людину, яка спілкується мовою жестів.

Також було згадано про технологію передачі звука через череп – це дасть можливість чути певній категорії людей із вадами слуху. Цим метод був випробуваний та довів своє право на існування. У поєднані з модифікацією пристрою, яка була описане вище, можна потужний і дешевий слуховий апарат, який дасть можливість зворотного зв’язку, що не надає жоден пристрій на сьогодні.

# ВИСНОВОК

У ході виконання роботи було створено унікальний пристрій та написано програмне забезпечення для нього. Разом ця робота дозволить провезти автоматичне та високоточний аналіз важкодоступного середовища. Також невисока ціна модифікації смартфону дозволяє використовувати його без страху вивезти з ладу.

Загалом використання модулю камери смартфону дало високий результат. Найбільш корисним виявилася можливість використання автофокусу, що дало можливість змінювати фокус камери для покращення чіткості картинки камери та підвищити точність роботи камери.

Запропонований алгоритм Калалом був модифікований, що підвищило його швидкодію та точність класифікації під вікон. Чотирьох етапний алгоритм виявлення може здатися надлишковим, але швидкодія перших двох етапів є суттєвою, а кількість від фільтрованих під вікон значною, що призводить до зменшення оброблювальної інформації двома іншими етапами. Метод фільтрації етапами, який був використаний в алгоритмі довів свою ефективність. Велику кількість інформації відфільтровують швидкобійні методи, тоді як більш точним, але повільним методам доводиться працювати з меншою кількістю даних, що призводить до ускладнення загального алгоритму, але підвищує швидкодію.

У даній роботі було запропоновано модифікацію алгоритму та пристрою, яка дозволить створити дешевий слуховий апарат із зворотнім зв’язком для людей із вадами слуху.

# СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. W. C. Abraham and A. Robins. Memory retention–the synaptic stability versus plasticity dilemma. Trends in neurosciences, 2005. - 73–78 с.
2. A. Adam, E. Rivlin, and I. Shimshoni. Robust fragments-based tracking using the integral histogram. In 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 1 (CVPR’06), volume 1, 2006. - 798–805 с.
3. Y. Amit and D. Geman. Shape quantization and recognition with randomized trees. Neural Computation, 1997. - 1545–1588 с.
4. S. Avidan. Support vector tracking. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2004. - 1064–1072 с.
5. S. Avidan. Ensemble tracking. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2007. - 261–271 с.
6. B. Babenko, Ming-Hsuan Yang, and S. Belongie. Visual tracking with online multiple instance learning. In 2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPR Workshops), 2009. - 983–990 с.
7. M. B. Blaschko. Branch and Bound Strategies for Non-maximal Suppression in Object Detection, volume 6819 of Lecture Notes in Computer Science, 2011. - 385– 398 с.
8. G. Bradski and A. Kaehler. Learning OpenCV: Computer Vision with the OpenCV Library. O’Reilly Media, 1st edition, Oct. 2008. - 8, 9 с.
9. L. Breiman. Random forests. Machine Learning, 45(1), Oct. 2001. - 5–32 с.
10. R. Brunelli. Template Matching Techniques in Computer Vision: Theory and Practice. Wiley Publishing, 2009. - 3, 22 с.
11. F. Chang. A linear-time component-labeling algorithm using contour tracing technique. Computer Vision and Image Understanding. 2004. - 206–220 с.
12. O. Chapelle, B. Schölkopf, and A. Zien, editors. Semi-Supervised Learning. The MIT Press, Sept. 2006. - 3, 28, 29 51 с.
13. S.-C. S. Cheung and C. Kamath. Robust techniques for background subtraction in urban traffic video. In Visual Communications and Image Processing 2004 (Proceedings Volume), volume 5308, SPIE, 2004. - 881–892 с.
14. R. T. Collins, Y. Liu, and M. Leordeanu. Online selection of discriminative tracking features. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2005. - 1631– 1643 с.
15. D. Comaniciu, V. Ramesh, and P. Meer. Real-time tracking of non-rigid objects using mean shift. In IEEE Conference on Computer Vision and Pattern Recognition, 2000. - 142–149 с.
16. N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In IEEE Computer Society Conference on Computer Vision and Pattern Recognition, June 2005. -12, 28, 886– 893 с.
17. J. Davis and M. Goadrich. The relationship between Precision-Recall and ROC curves. In Proceedings of the 23rd international conference on Machine learning, ICML ’06, New York, NY, USA, 2006. - 233–240 с.
18. M. Everingham, L. Van Gool, C. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (VOC) challenge. International Journal of Computer Vision, 88(2), June 2010. – 24, 303–338 с.
19. M. Godec, P. M. Roth, and H. Bischof. Hough-based tracking of non-rigid objects. In IEEE International Conference on Computer Vision IEEE, Nov. 2011. - 81–88 с.
20. E. B. Goldstein. Sensation and Perception. Wadsworth Publishing, 8 edition, Feb. 2009.
21. H. Grabner, C. Leistner, and H. Bischof. Semi-supervised On-Line boosting for robust tracking. In D. Forsyth, and A. Zisserman, editors, Proceedings of the 10th European Conference on Computer Vision, volume 5302, Berlin, Heidelberg, 2008. - 234–247 с.
22. S. Grossberg. Competitive learning: From interactive activation to adaptive resonance. Cognitive Science, 11(1), Jan. 1987. – 2, 23–63 с.
23. S. Hare, A. Saffari, and P. H. S. Torr. Struck: Structured output tracking with kernels. In IEEE International Conference on Computer Vision, IEEE, Nov. 2011. - 4, 43, 44, 263–270 с.
24. B. Hemery, H. Laurent, and C. Rosenberger. Comparative study of metrics for evaluation of object localisation by bounding boxes. In International Conference on Image and Graphics. IEEE, Aug. 2007. – 33, 459–464 с.
25. O. Javed, S. Ali, and Mubarak Shah. Online detection and classification of moving objects using progressively improving detectors. In 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05), volume 1. IEEE, 2005. – 3, 52, 696–701 с.
26. Z. Kalal, J. Matas, and K. Mikolajczyk. Online learning of robust object detectors during unstable tracking. In Proceedings of the IEEE On-line Learning for Computer Vision Workshop, 2009. – 4, 1417–1424 с.
27. Z. Kalal, J. Matas, and K. Mikolajczyk. P-N learning: Bootstrapping binary classifiers by structural constraints. In 2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, June 2010. - 3, 4, 5, 27, 29, 36, 43, 44, 49–56 с.
28. Z. Kalal, K. Mikolajczyk, and J. Matas. Forward-Backward Error: Automatic Detection of Tracking Failures. In International Conference on Pattern Recognition, 2010. - 4, 7, 8, 9, 10, 23–26 с.
29. C. H. Lampert, M. B. Blaschko, and T. Hofmann. Beyond sliding windows: Object localization by efficient subwindow search. In IEEE Conference on Computer Vision and Pattern Recognition, 2008. – 14, 1–8 с.
30. V. Lepetit and P. Fua. Monocular model-based 3D tracking of rigid objects. Found. Trends. Comput. Graph. Vis., 2005. – 2, 1–89 с.
31. V. Lepetit, P. Lagger, and P. Fua. Randomized trees for Real-Time keypoint recognition. In IEEE Computer Society Conference on Computer Vision and Pattern Recognition, volume 2, Los Alamitos, CA, USA, 2005. - 775–781 с.
32. J. P. Lewis. Fast normalized cross-correlation. In Vision Interface. Canadian Image Processing and Pattern Recognition Society, 1995. – 9, 120–123 с.
33. B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In Proceedings of the International Joint Conference on Artificial Intelligence, 1981. - 2, 4, 7, 674–679 с.
34. E. Maggio and A. Cavallaro. Video Tracking: Theory and Practice. Wiley, 2011. – 1 с.
35. L. Matthews, T. Ishikawa, and S. Baker. The template update problem. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2004. - 810–815 с.
36. T. M. Mitchell. Machine Learning. McGraw-Hill Science/Engineering/Math, 1 edition, Mar. 1997. - 21 с.
37. F. Murtagh. A survey of recent advances in hierarchical clustering algorithms. The Computer Journal, 26(4):354–359, Nov. 1983. - 24 с.
38. M. Ozuysal, P. Fua, and V. Lepetit. Fast keypoint recognition in ten lines of code. In IEEE Conference on Computer Vision and Pattern Recognition, Los Alamitos, CA, USA, June 2007. - 3, 4, 12, 19, 21, 37, 38, 48 с.
39. R. J. Radke, S. Andra, O. Al-Kofahi, and B. Roysam. Image change detection algorithms: a systematic survey. IEEE Transactions on Image Processing, Mar. 2005. -14 53, 810–815 с.
40. J. L. Rodgers and W. A. Nicewander. Thirteen ways to look at the correlation coefficient. The American Statistician, 1988. - 22, 59–66 с.
41. D. Ross, J. Lim, R.-S. Lin, and M.-H. Yang. Incremental learning for robust visual tracking. International Journal of Computer Vision, May 2008. - 2, 3, 25, 125–141 с.
42. J. Santner, C. Leistner, A. Saffari, T. Pock, and H. Bischof. PROST: Parallel robust online simple tracking. In IEEE Conference on Computer Vision and Pattern Recognition. IEEE, June 2010. - 723–730 с.
43. H. Schneiderman. Feature-centric evaluation for efficient cascaded object detection. In IEEE Computer Society Conference on Computer Vision and Pattern Recognition, volume 2. IEEE, June 2004. - 12 с.
44. L. G. Shapiro and G. C. Stockman. Computer Vision. Prentice Hall, Jan. 2001. – 1 с.
45. J. Shi and C. Tomasi. Good features to track. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR’94), Seattle, June 1994. - 7, 9 с.
46. S. Stalder, H. Grabner, and L. van Gool. Beyond semi-supervised tracking: Tracking should be as simple as detection, but not simpler than recognition. In IEEE International Conference on Computer Vision Workshops, IEEE, 2009. - 4, 1409–1416 с.
47. R. Szeliski. Computer Vision: Algorithms and Applications. Springer, 2010. – 1 с.
48. C. Tomasi and T. Kanade. Detection and tracking of point features. Technical Report CMU-CS-91-132, Carnegie Mellon University, Apr. 1991. - 8 с.
49. P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001, volume 1, Los Alamitos, CA, USA, Apr. 2001. IEEE Comput. Soc. - 12, 16, 24, 28, 50 с.
50. A. Yilmaz, O. Javed, and M. Shah. Object tracking: A survey. ACM Computing Surveys, 38(4), Dec. 2006. - 1, 2 с.
51. Q. Yu, T. B. Dinh, and G. Medioni. Online tracking and reacquisition using co-trained generative and discriminative trackers. In European Conference on Computer Vision, volume 5303 of Lecture Notes in Computer Science, Berlin, Heidelberg, 2008. - 36, 43, 678–691 с.
52. Ендоскопія [Електроний ресурс] – 2016. – Режим доступу: <https://uk.wikipedia.org/wiki/Ендоскопія>
53. Технічна документація по LG P700 [Електроний ресурс] – 2016. – Режим доступу: <http://www.gsm71.ru/_ld/0/36_LG_P700_Optimus.pdf>

ДОДАТКИ **до дипломного проекту**

на тему: «Розробка системи діагностики важкодоступних середовищ»

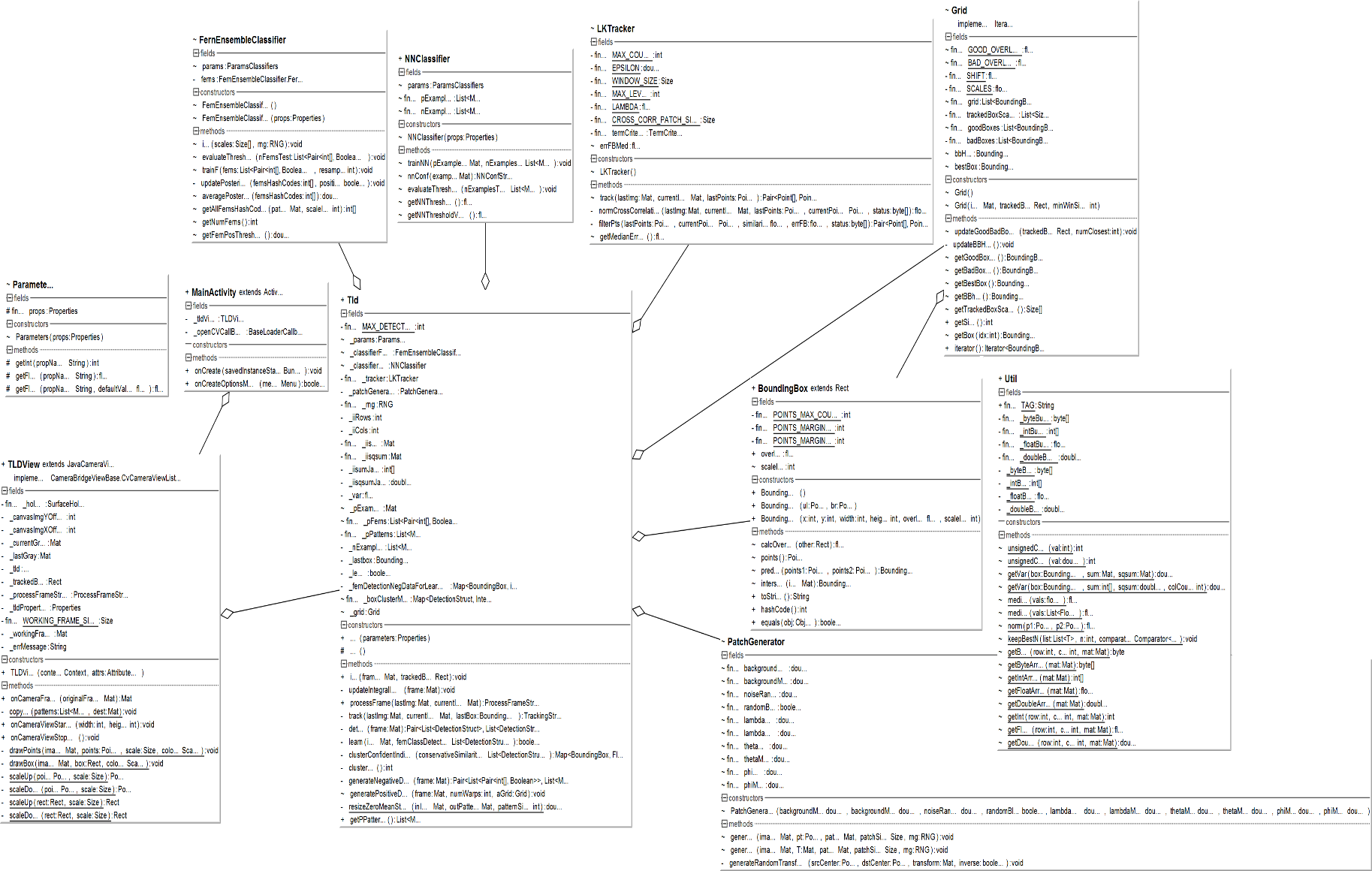
# ДОДАТОК А

РОЗРОБКА СИСТЕМИ ДІАГНОСТИКИ ВАЖКОДОСТУПНИХ СЕРЕДОВИЩ

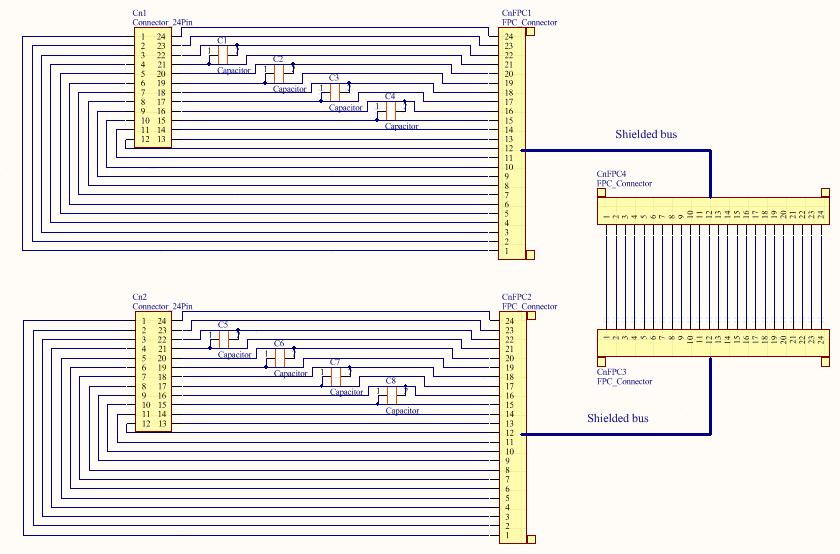
# КОПІЇ ГРАФІЧНИХ МАТЕРІАЛІВ

Аркушів 3

Київ – 2016







# ДОДАТОК Б

РОЗРОБКА СИСТЕМИ ДІАГНОСТИКИ ВАЖКОДОСТУПНИХ СЕРЕДОВИЩ

# ЛІСТИНГ ПРОГРАМИ

Аркушів 51

Київ – 2016

**MainActivity.java**

package com.trandi.opentld;

import org.opencv.android.BaseLoaderCallback;

import org.opencv.android.LoaderCallbackInterface;

import org.opencv.android.OpenCVLoader;

import android.app.Activity;

import android.os.Bundle;

import android.util.Log;

import android.view.Menu;

import android.view.SurfaceView;

import android.view.Window;

import com.trandi.opentld.tld.Util;

public class MainActivity extends Activity {

private TLDView \_tldView;

private BaseLoaderCallback \_openCVCallBack = new BaseLoaderCallback(this) {

@Override

public void onManagerConnected(int status) {

switch (status) {

case LoaderCallbackInterface.SUCCESS:

{

Log.i(Util.TAG, "OpenCV loaded successfully");

setContentView(R.layout.activity\_main);

\_tldView = (TLDView) findViewById(R.id.tld\_view);

\_tldView.setVisibility(SurfaceView.VISIBLE);

\_tldView.enableView();

} break;

default:

{

Log.e(Util.TAG, "OpenCV can NOT be loaded");

super.onManagerConnected(status);

} break;

}

}

};

/\*\* Called when the activity is first created. \*/

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

requestWindowFeature(Window.FEATURE\_NO\_TITLE);

/\*\* Called when the activity is first created. \*/

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

requestWindowFeature(Window.FEATURE\_NO\_TITLE);

Log.i(Util.TAG, "Trying to load OpenCV library");

if (!OpenCVLoader.initAsync(OpenCVLoader.OPENCV\_VERSION\_2\_4\_6, this, \_openCVCallBack)) {

Log.e(Util.TAG, "Cannot connect to OpenCV Manager");

}

}

@Override

public boolean onCreateOptionsMenu(Menu menu) {

getMenuInflater().inflate(R.menu.activity\_main, menu);

return true;

}

}

**TLDView.java**

package com.trandi.opentld;

import java.io.IOException;

import java.io.InputStream;

import java.util.List;

import java.util.Properties;

import java.util.concurrent.atomic.AtomicReference;

import org.opencv.android.CameraBridgeViewBase;

import org.opencv.android.JavaCameraView;

import org.opencv.core.Core;

import org.opencv.core.Mat;

import org.opencv.core.Point;

import org.opencv.core.Rect;

import org.opencv.core.Scalar;

import org.opencv.core.Size;

import org.opencv.imgproc.Imgproc;

import android.content.Context;

import android.graphics.Canvas;

import android.graphics.Color;

import android.graphics.Paint;

import android.graphics.Paint.Style;

import android.graphics.PorterDuff;

import android.util.AttributeSet;

import android.util.Log;

import android.view.MotionEvent;

import android.view.SurfaceHolder;

import android.view.View;

import com.trandi.opentld.tld.Tld;

import com.trandi.opentld.tld.Tld.ProcessFrameStruct;

import com.trandi.opentld.tld.Util;

public class TLDView extends JavaCameraView implements CameraBridgeViewBase.CvCameraViewListener {

final private SurfaceHolder \_holder;

private int \_canvasImgYOffset;

private int \_canvasImgXOffset;

private Mat \_currentGray = new Mat();

private Mat \_lastGray = new Mat();

private Tld \_tld = null;

private Rect \_trackedBox = null;

private ProcessFrameStruct \_processFrameStruct = null;

private Properties \_tldProperties;

private static final Size WORKING\_FRAME\_SIZE = new Size(144, 80);

private Mat \_workingFrame = new Mat();

private String \_errMessage;

public TLDView(Context context, AttributeSet attrs) {

super(context, attrs);

\_holder = getHolder();

// Init the PROPERTIES

InputStream propsIS = null;

try{

propsIS = context.getResources().openRawResource(R.raw.parameters);

\_tldProperties = new Properties();

\_tldProperties.load(propsIS);

} catch (IOException e) {

Log.e(Util.TAG, "Can't load properties", e);

}finally{

if(propsIS != null){

try {

propsIS.close();

} catch (IOException e) {

Log.e(Util.TAG, "Can't close props", e);

}

}

}

// listens to its own events

setCvCameraViewListener(this);

// DEBUG

//\_trackedBox = new BoundingBox(165,93,51,54, 0, 0);

// LISTEN for touches of the screen, to define the BOX to be tracked

final AtomicReference<Point> trackedBox1stCorner = new AtomicReference<Point>();

final Paint rectPaint = new Paint();

rectPaint.setColor(Color.rgb(0, 255, 0));

rectPaint.setStrokeWidth(5);

rectPaint.setStyle(Style.STROKE);

setOnTouchListener(new OnTouchListener() {

@Override

public boolean onTouch(View v, MotionEvent event) {

// re-init

\_errMessage = null;

\_tld = null;

final Point corner = new Point(event.getX() - \_canvasImgXOffset, event.getY() - \_canvasImgYOffset);

switch(event.getAction()){

case MotionEvent.ACTION\_DOWN:

trackedBox1stCorner.set(corner);

Log.i(Util.TAG, "1st corner: " + corner);

break;

case MotionEvent.ACTION\_UP:

\_trackedBox = new Rect(trackedBox1stCorner.get(), corner);

Log.i(Util.TAG, "Tracked box DEFINED: " + \_trackedBox);

break;

case MotionEvent.ACTION\_MOVE:

final android.graphics.Rect rect = new android.graphics.Rect(

(int)trackedBox1stCorner.get().x + \_canvasImgXOffset, (int)trackedBox1stCorner.get().y + \_canvasImgYOffset,

(int)corner.x + \_canvasImgXOffset, (int)corner.y + \_canvasImgYOffset);

final Canvas canvas =\_holder.lockCanvas(rect);

canvas.drawColor(Color.TRANSPARENT, PorterDuff.Mode.CLEAR); // remove old rectangle

canvas.drawRect(rect, rectPaint);

\_holder.unlockCanvasAndPost(canvas);

break;

}

return true;

}

});

}

@Override

public Mat onCameraFrame(Mat originalFrame) {

try{

// Image is too big and this requires too much CPU for a phone, so scale everything down...

Imgproc.resize(originalFrame, \_workingFrame, WORKING\_FRAME\_SIZE);

final Size workingRatio = new Size(originalFrame.width() / WORKING\_FRAME\_SIZE.width, originalFrame.height() / WORKING\_FRAME\_SIZE.height);

// usefull to see what we're actually working with...

\_workingFrame.copyTo(originalFrame.submat(originalFrame.rows() - \_workingFrame.rows(), originalFrame.rows(), 0, \_workingFrame.cols()));

if(\_trackedBox != null){

if(\_tld == null){ // run the 1st time only

Imgproc.cvtColor(\_workingFrame, \_lastGray, Imgproc.COLOR\_RGB2GRAY);

\_tld = new Tld(\_tldProperties);

final Rect scaledDownTrackedBox = scaleDown(\_trackedBox, workingRatio);

Log.i(Util.TAG, "Working Ration: " + workingRatio + " / Tracking Box: " + \_trackedBox + " / Scaled down to: " + scaledDownTrackedBox);

try {

\_tld.init(\_lastGray, scaledDownTrackedBox);

}catch(Exception eInit){

// start from scratch, you have to select an init box again !

\_trackedBox = null;

\_tld = null;

throw eInit; // re-throw it as it will be dealt with later

}

}else{

Imgproc.cvtColor(\_workingFrame, \_currentGray, Imgproc.COLOR\_RGB2GRAY);

\_processFrameStruct = \_tld.processFrame(\_lastGray, \_currentGray);

drawPoints(originalFrame, \_processFrameStruct.lastPoints, workingRatio, new Scalar(255, 0, 0));

drawPoints(originalFrame, \_processFrameStruct.currentPoints, workingRatio, new Scalar(0, 255, 0));

drawBox(originalFrame, scaleUp(\_processFrameStruct.currentBBox, workingRatio), new Scalar(0, 0, 255));

\_currentGray.copyTo(\_lastGray);

// overlay the current positive examples on the real image(needs converting at the same time !)

//copyTo(\_tld.getPPatterns(), originalFrame);

}

}

} catch(Exception e) {

\_errMessage = e.getClass().getSimpleName() + " / " + e.getMessage();

Log.e(Util.TAG, "TLDView PROBLEM", e);

}

if(\_errMessage != null){

Core.putText(originalFrame, \_errMessage, new Point(0, 300), Core.FONT\_HERSHEY\_PLAIN, 1.3d, new Scalar(255, 0, 0), 2);

}

return originalFrame;

}

private static void copyTo(List<Mat> patterns, Mat dest) {

if(patterns == null || patterns.isEmpty() || dest == null) return;

final int patternRows = patterns.get(0).rows();

final int patternCols = patterns.get(0).cols();

final int vertCount = dest.rows() / patternRows;

final int horizCount = patterns.size() / vertCount + 1;

int patchIdx = 0;

for(int col = dest.cols() - horizCount \* patternCols - 1; col < dest.cols() && patchIdx < patterns.size(); col += patternCols){

for(int row = 0; row < dest.rows() && patchIdx < patterns.size(); row += patternRows) {

Imgproc.cvtColor(patterns.get(patchIdx), dest.submat(row, row + patternRows, col, col + patternCols), Imgproc.COLOR\_GRAY2RGBA);

patchIdx++;

}

}

}

@Override

public void onCameraViewStarted(int width, int height) {

\_canvasImgXOffset = (getWidth() - width) / 2;

\_canvasImgYOffset = (getHeight() - height) / 2;

}

@Override

public void onCameraViewStopped() {

// TODO Auto-generated method stub

}

private static void drawPoints(Mat image, final Point[] points, final Size scale, final Scalar colour){

if(points != null){

for(Point point : points){

Core.circle(image, scaleUp(point, scale), 2, colour);

}

}

}

private static void drawBox(Mat image, final Rect box, final Scalar colour){

if(box != null){

Core.rectangle(image, box.tl(), box.br(), colour);

}

}

/\* SCALING \*/

private static Point scaleUp(Point point, Size scale){

if(point == null || scale == null) return null;

return new Point(point.x \* scale.width, point.y \* scale.height);

}

private static Point scaleDown(Point point, Size scale){

if(point == null || scale == null) return null;

return new Point(point.x / scale.width, point.y / scale.height);

}

private static Rect scaleUp(Rect rect, Size scale) {

if(rect == null || scale == null) return null;

return new Rect(scaleUp(rect.tl(), scale), scaleUp(rect.br(), scale));

}

private static Rect scaleDown(Rect rect, Size scale) {

if(rect == null || scale == null) return null;

return new Rect(scaleDown(rect.tl(), scale), scaleDown(rect.br(), scale));

}

}

**BoundingBox.java**

package com.trandi.opentld.tld;

import java.util.ArrayList;

import java.util.List;

import org.opencv.core.Mat;

import org.opencv.core.Point;

import org.opencv.core.Rect;

import android.util.Log;

public class BoundingBox extends Rect{

private static final int POINTS\_MAX\_COUNT = 10;

private static final int POINTS\_MARGIN\_H = 0;

private static final int POINTS\_MARGIN\_V = 0;

public float overlap = -1; // overlap with current(main) BoundingBox

int scaleIdx = -1;

public BoundingBox(){

super();

}

public BoundingBox(final Point ul, final Point br){

super(ul, br);

}

public BoundingBox(int x, int y, int width, int height, float overlap, int scaleIdx){

super(x, y, width, height);

this.overlap = overlap;

this.scaleIdx = scaleIdx;

}

float calcOverlap(final Rect other){

if(x > other.x + other.width || y > other.y + other.height || x + width < other.x || y + height < other.y){

// obvious case where these 2 boxes do not overlap at all !

return 0f;

}else{

final float colIntersection = Math.min(x + width, other.x + other.width) - Math.max(x, other.x);

final float rowIntersection = Math.min(y + height, other.y + other.height) - Math.max(y, other.y);

final float intersection = colIntersection \* rowIntersection;

final float myArea = width \* height;

final float otherArea = other.width \* other.height;

return intersection / (myArea + otherArea - intersection);

}

}

Point[] points(){

final List<Point> result = new ArrayList<Point>();

final int stepx = (int) Math.ceil((width - 2 \* POINTS\_MARGIN\_H) / POINTS\_MAX\_COUNT);

final int stepy = (int) Math.ceil((height - 2 \* POINTS\_MARGIN\_V) / POINTS\_MAX\_COUNT);

for(int j = y + POINTS\_MARGIN\_V; j < y + height - POINTS\_MARGIN\_V; j += stepy){

for(int i = x + POINTS\_MARGIN\_H; i < x + width - POINTS\_MARGIN\_H; i += stepx){

result.add(new Point(i, j));

}

}

Log.i(Util.TAG, "Points in BB: " + this + " stepx=" + stepx + " stepy=" + stepy + " RES size=" + result.size());

return result.toArray(new Point[result.size()]);

}

BoundingBox predict(final Point[] points1, final Point[] points2){

if(points1.length != points2.length) throw new IllegalArgumentException("The 2 arrays of points must be of the same lenght ! (" + points1.length + ", " + points2.length + ")");

final int npoints = points1.length;

Log.i(Util.TAG, "Tracked points: " + npoints);

final float[] xoff = new float[npoints];

final float[] yoff = new float[npoints];

for(int i = 0; i < npoints; i++){

xoff[i] = (float) (points2[i].x - points1[i].x);

yoff[i] = (float) (points2[i].y - points1[i].y);

}

final float dx = Util.median(xoff);

final float dy = Util.median(yoff);

float s = 1f;

if(npoints > 1){

final float[] d = new float[npoints \* (npoints - 1) / 2];

int idx = 0;

for(int i = 0; i < npoints; i++){

for(int j = i + 1; j < npoints; j++){

d[idx++] = Util.norm(points2[i], points2[j]) / Util.norm(points1[i], points1[j]);

}

}

s = Util.median(d);

}

final float s1 = 0.5f \* (s - 1) \* width;

final float s2 = 0.5f \* (s - 1) \* height;

final BoundingBox result = new BoundingBox();

result.x = Math.max(Math.round(x + dx - s1), 0);

result.y = Math.max(Math.round(y + dy - s2), 0);

result.width = Math.round(width \* s);

result.height = Math.round(height \* s);

Log.i(Util.TAG, "Current BB: " + this + ", Predicted BB: " + result);

return result;

}

BoundingBox intersect(final Mat img){

final BoundingBox result = new BoundingBox();

result.x = Math.max(x, 0);

result.y = Math.max(y, 0);

result.width = (int) Math.min(Math.min(img.cols() - x, width), Math.min(width, br().x));

result.height = (int) Math.min(Math.min(img.rows() - y, height), Math.min(height, br().y));

return result;

}

@Override

public String toString(){

return "(" + x + ", " + y + ", " + width + ", " + height + " / " + overlap + ", " + scaleIdx + ")";

}

@Override

public int hashCode() {

final int prime = 31;

int result = super.hashCode();

result = prime \* result + Float.floatToIntBits(overlap);

result = prime \* result + scaleIdx;

return result;

}

@Override

public boolean equals(Object obj) {

if (this == obj)

return true;

if (!super.equals(obj))

return false;

if (getClass() != obj.getClass())

return false;

BoundingBox other = (BoundingBox) obj;

if (Float.floatToIntBits(overlap) != Float

.floatToIntBits(other.overlap))

return false;

if (scaleIdx != other.scaleIdx)

return false;

return true;

}

}

**FernEnsembleClassifier.java**

package com.trandi.opentld.tld;

import java.util.List;

import java.util.Properties;

import org.opencv.core.Mat;

import org.opencv.core.Size;

import android.util.Log;

import com.trandi.opentld.tld.Parameters.ParamsClassifiers;

import com.trandi.opentld.tld.Util.Pair;

import com.trandi.opentld.tld.Util.RNG;

class FernEnsembleClassifier {

ParamsClassifiers params;

private Fern[] ferns;

// final List<Mat> pExamples = new ArrayList<Mat>();

// final List<Mat> nExamples = new ArrayList<Mat>();

FernEnsembleClassifier(){

}

FernEnsembleClassifier(Properties props) {

params = new ParamsClassifiers(props);

}

void init(Size[] scales, RNG rng){

ferns = new Fern[params.numFerns];

for(int i=0; i<ferns.length; i++){

ferns[i] = new Fern(params.numFeaturesPerFern, scales, rng);

}

}

/\*\*

\* Updates the POSITIVE Ferns

\* The threshold for Positive results has to be > to the average of negative posteriors

\*/

void evaluateThreshold(final List<Pair<int[], Boolean>> nFernsTest){

for(Pair<int[], Boolean> fern : nFernsTest){

// here we know/hope that fern.second is always FALSE, as they are all NEGATIVE examples

final double averagePosterior = averagePosterior(fern.first);

if(averagePosterior > params.pos\_thr\_fern){

params.pos\_thr\_fern = averagePosterior;

}

}

}

void trainF(final List<Pair<int[], Boolean>> ferns, int resample){

for(int i = 0; i < resample; i++){

for(Pair<int[], Boolean> fern : ferns){

// the THRESHOLDS are here to make sure we don't increase/decrease the probabilities beyond given limits, to give other hashCodes a chance

if(fern.second){ // if it's a positive fern

if(averagePosterior(fern.first) <= params.pos\_thr\_fern){

updatePosteriors(fern.first, true);

}

}else if(averagePosterior(fern.first) >= params.neg\_thr\_fern){

updatePosteriors(fern.first, false);

}

}

}

}

private void updatePosteriors(final int[] fernsHashCodes, boolean positive){

assert(params.numFerns == fernsHashCodes.length);

for(int fern = 0; fern < fernsHashCodes.length; fern++){

ferns[fern].addCountUpdatePosteriors(fernsHashCodes[fern], positive);

}

}

/\*\*

\* @return conf

\*/

double averagePosterior(final int[] fernsHashCodes){

assert(params.numFerns == fernsHashCodes.length);

double result = 0;

for(int fern = 0; fern < fernsHashCodes.length; fern++){

result += ferns[fern].posteriorProbabilities[fernsHashCodes[fern]];

}

return result / fernsHashCodes.length;

}

/\*\*

\* The numbers in this array can be up to 2^params.structSize as we shift left once of each feature

\*/

int[] getAllFernsHashCodes(final Mat patch, int scaleIdx){

final int[] result = new int[ferns.length];

final byte[] imageData = Util.getByteArray(patch);

final int cols = patch.cols();

for(int fern = 0; fern < ferns.length; fern++){

result[fern] = ferns[fern].calculateHashCode(scaleIdx, imageData, cols);

}

return result;

}

static class Fern {

private final Feature[][] features; // per scaleIdx

// per HASHCODE

final double[] posteriorProbabilities; // the probability that it's our image

final long[] nCounter; // the number of NEGATIVE patches

final long[] pCounter; // the number of POSITIVE patches

Fern(int featuresPerFern, Size[] scales, RNG rng) {

// 1. Define random features

features = new Feature[scales.length][featuresPerFern];

for (int i=0; i<featuresPerFern; i++){

final float x1f = rng.nextFloat();

final float y1f = rng.nextFloat();

final float x2f = rng.nextFloat();

final float y2f = rng.nextFloat();

for (int s=0; s<scales.length; s++){

final int x1 = (int) (x1f \* scales[s].width);

final int y1 = (int) (y1f \* scales[s].height);

final int x2 = (int) (x2f \* scales[s].width);

final int y2 = (int) (y2f \* scales[s].height);

features[s][i] = new Feature(x1, y1, x2, y2);

}

}

// 2. Initialise Posteriors

final int MAX\_HASHCODE = (int)Math.pow(2d, featuresPerFern);

posteriorProbabilities = new double[MAX\_HASHCODE];

pCounter = new long[MAX\_HASHCODE];

nCounter = new long[MAX\_HASHCODE];

}

void addCountUpdatePosteriors(int fernHashCode, boolean positive) {

if(positive){

pCounter[fernHashCode] ++;

}else{

nCounter[fernHashCode] ++;

}

posteriorProbabilities[fernHashCode] = ((double)pCounter[fernHashCode]) / (pCounter[fernHashCode] + nCounter[fernHashCode]);

}

int calculateHashCode(int scaleIdx, byte[] imageData, int cols) {

int fernHashCode = 0;

for(Feature feature : features[scaleIdx]){

// compare returns 0 / 1 and

fernHashCode = (fernHashCode << 1) + feature.compare(imageData, cols);

}

return fernHashCode;

}

}

/\*\*

\* A Feature is a pixel Comparison, between 2 points.

\*/

private static class Feature {

private final int x1, y1, x2, y2;

public Feature(int x1, int y1, int x2, int y2) {

this.x1 = x1;

this.y1 = y1;

this.x2 = x2;

this.y2 = y2;

}

/\*\*

\* Simply compares the brightness between the 2 points defining this Feature

\* Assumes channels = 1 (hence only multiplying with cols).

\*/

public int compare(final byte[] patch, final int cols) {

final int pos1 = y1 \* cols + x1;

final int pos2 = y2 \* cols + x2;

if(pos1 >= patch.length || pos2 >= patch.length) {

Log.w(Util.TAG, "Bad patch of size: " + patch.length + " cols: " + cols + " to compare Feature: " + this.toString());

return 0;

}

final boolean boolRes = patch[pos1] > patch[pos2];

return boolRes ? 1 : 0;

}

@Override

public String toString(){

return x1 + ", " + y1 + ", " + x2 + ", " + y2;

}

}

int getNumFerns(){

return params.numFerns;

}

double getFernPosThreshold(){

return params.pos\_thr\_fern;

}

// TODO use to display the positive examples used by learning...

// public Mat getPosExamples(){

// if(pExamples == null || pExamples.size() == 0) return null;

//

// final int exRows = pExamples.get(0).rows();

// final int exCols = pExamples.get(0).cols();

//

// // create a Matrix that can contain vertically all the positive examples

// final Mat result = new Mat(pExamples.size() \* exRows, exCols, CvType.CV\_8U);

// Imgproc.

// }

}

**Grid.java**

package com.trandi.opentld.tld;

import java.util.ArrayList;

import java.util.Comparator;

import java.util.Iterator;

import java.util.List;

import org.opencv.core.Mat;

import org.opencv.core.Rect;

import org.opencv.core.Size;

import android.util.Log;

class Grid implements Iterable<BoundingBox>{

static final float GOOD\_OVERLAP = 0.6f;

static final float BAD\_OVERLAP = 0.2f;

private static final float SHIFT = 0.1f;

private static final float[] SCALES = {

0.16151f, 0.19381f, 0.23257f, 0.27908f, 0.33490f, 0.40188f, 0.48225f,

0.57870f, 0.69444f, 0.83333f, 1f, 1.20000f, 1.44000f, 1.72800f,

2.07360f, 2.48832f, 2.98598f, 3.58318f, 4.29982f, 5.15978f, 6.19174f};

final List<BoundingBox> grid = new ArrayList<BoundingBox>();

private final List<Size> trackedBoxScales = new ArrayList<Size>();

final List<BoundingBox> goodBoxes = new ArrayList<BoundingBox>(); //bboxes with overlap > GOOD\_OVERLAP

final private List<BoundingBox> badBoxes = new ArrayList<BoundingBox>(); //bboxes with overlap < BAD\_OVERLAP

BoundingBox bbHull = new BoundingBox(); // hull of good\_boxes

BoundingBox bestBox; // maximum overlapping bbox

Grid(){

}

Grid(Mat img, Rect trackedBox, int minWinSide){

// TODO why do we generate so many BAD boxes, only to remove them later on !?

// OR do we need them to re-asses which ones are bad later on ?

for(int s=0; s<SCALES.length; s++){

final int width = Math.round(trackedBox.width \* SCALES[s]);

final int height = Math.round(trackedBox.height \* SCALES[s]);

final int minBbSide = Math.min(height, width);

// continue ONLY if the future box is "reasonable": bigger than the min window and smaller than the full image !

if(minBbSide >= minWinSide && width <= img.cols() && height <= img.rows()){

trackedBoxScales.add(new Size(width, height));

final int shift = Math.round(SHIFT \* minBbSide);

for(int row=1; row<(img.rows() - height); row+=shift){

for(int col=1; col<(img.cols() - width); col+=shift){

final BoundingBox bbox = new BoundingBox();

bbox.x = col;

bbox.y = row;

bbox.width = width;

bbox.height = height;

bbox.scaleIdx = trackedBoxScales.size() - 1; // currently last one in this list

grid.add(bbox);

}

}

}

}

}

/\*\*

\* goodBoxes OUTPUT

\* badBoxes OUTPUT

\*

\* This should be called AFTER updateOverlap(lastBox) so that the overlap numbers are relative to this lastBox, NOT the initial one...

\*/

void updateGoodBadBoxes(final Rect trackedBox, final int numClosest) {

// start by updating the overlap numbers

for(BoundingBox box : grid){

box.overlap = box.calcOverlap(trackedBox);

}

goodBoxes.clear();

badBoxes.clear();

float maxOverlap = 0f;

for(BoundingBox box : grid){

if(box.overlap > maxOverlap){

maxOverlap = box.overlap;

bestBox = box;

}

if(box.overlap > GOOD\_OVERLAP){

goodBoxes.add(box);

}else if(box.overlap < BAD\_OVERLAP){

badBoxes.add(box);

}

}

// keep only the best numClosest (10) items in goodBoxes

Util.keepBestN(goodBoxes, numClosest, new Comparator<BoundingBox>(){

@Override

public int compare(BoundingBox bb1, BoundingBox bb2) {

return Float.valueOf(bb1.overlap).compareTo(bb2.overlap);

}

});

Log.i(Util.TAG, "Found " + goodBoxes.size() + " good boxes, " + badBoxes.size() + " bad boxes.");

Log.i(Util.TAG, "Best Box: " + bestBox);

updateBBHull();

Log.i(Util.TAG, "Bounding box hull " + bbHull);

}

private void updateBBHull(){

//if(goodBoxes.isEmpty()) throw new IllegalStateException("Can't Calculate the BBHull without at least 1 good box !");

int x1 = Integer.MAX\_VALUE, x2 = 0;

int y1 = Integer.MAX\_VALUE, y2 = 0;

for (BoundingBox goodBox : goodBoxes) {

x1 = Math.min(goodBox.x, x1);

y1 = Math.min(goodBox.y, y1);

x2 = Math.max(goodBox.x + goodBox.width, x2);

y2 = Math.max(goodBox.y + goodBox.height, y2);

}

bbHull.x = x1;

bbHull.y = y1;

bbHull.width = x2 - x1;

bbHull.height = y2 - y1;

}

BoundingBox[] getGoodBoxes(){

return goodBoxes.toArray(new BoundingBox[goodBoxes.size()]);

}

BoundingBox[] getBadBoxes(){

return badBoxes.toArray(new BoundingBox[badBoxes.size()]);

}

BoundingBox getBestBox(){

return bestBox;

}

BoundingBox getBBhull(){

return bbHull;

}

Size[] getTrackedBoxScales(){

return trackedBoxScales.toArray(new Size[trackedBoxScales.size()]);

}

public int getSize(){

return grid.size();

}

BoundingBox getBox(int idx){

return grid.get(idx);

}

@Override

public Iterator<BoundingBox> iterator() {

return grid.iterator();

}

}

**LKTracker.java**

package com.trandi.opentld.tld;

import java.util.ArrayList;

import java.util.List;

import org.opencv.core.CvType;

import org.opencv.core.Mat;

import org.opencv.core.MatOfByte;

import org.opencv.core.MatOfFloat;

import org.opencv.core.MatOfPoint2f;

import org.opencv.core.Point;

import org.opencv.core.Size;

import org.opencv.core.TermCriteria;

import org.opencv.imgproc.Imgproc;

import org.opencv.video.Video;

import android.util.Log;

import com.trandi.opentld.tld.Util.Pair;

class LKTracker {

private static final int MAX\_COUNT = 20;

private static final double EPSILON = 0.03;

private static final Size WINDOW\_SIZE = new Size(4, 4);

private static final int MAX\_LEVEL = 5;

private static final float LAMBDA = 0f; // minEigenThreshold

private static final Size CROSS\_CORR\_PATCH\_SIZE = new Size(10, 10);

private final TermCriteria termCriteria;

float errFBMed;

LKTracker(){

termCriteria = new TermCriteria(TermCriteria.COUNT + TermCriteria.EPS, MAX\_COUNT, EPSILON);

}

/\*\*

\* @return Pair of new, FILTERED, last and current POINTS, or null if it hasn't managed to track anything.

\*/

Pair<Point[], Point[]> track(final Mat lastImg, final Mat currentImg, Point[] lastPoints){

final int size = lastPoints.length;

final MatOfPoint2f currentPointsMat = new MatOfPoint2f();

final MatOfPoint2f pointsFBMat = new MatOfPoint2f();

final MatOfByte statusMat = new MatOfByte();

final MatOfFloat errSimilarityMat = new MatOfFloat();

final MatOfByte statusFBMat = new MatOfByte();

final MatOfFloat errSimilarityFBMat = new MatOfFloat();

//Forward-Backward tracking

Video.calcOpticalFlowPyrLK(lastImg, currentImg, new MatOfPoint2f(lastPoints), currentPointsMat,

statusMat, errSimilarityMat, WINDOW\_SIZE, MAX\_LEVEL, termCriteria, 0, LAMBDA);

Video.calcOpticalFlowPyrLK(currentImg, lastImg, currentPointsMat, pointsFBMat,

statusFBMat, errSimilarityFBMat, WINDOW\_SIZE, MAX\_LEVEL, termCriteria, 0, LAMBDA);

final byte[] status = statusMat.toArray();

float[] errSimilarity = new float[lastPoints.length];

//final byte[] statusFB = statusFBMat.toArray();

final float[] errSimilarityFB = errSimilarityFBMat.toArray();

// compute the real FB error (relative to LAST points not the current ones...

final Point[] pointsFB = pointsFBMat.toArray();

for(int i = 0; i < size; i++){

errSimilarityFB[i] = Util.norm(pointsFB[i], lastPoints[i]);

}

final Point[] currPoints = currentPointsMat.toArray();

// compute real similarity error

errSimilarity = normCrossCorrelation(lastImg, currentImg, lastPoints, currPoints, status);

//TODO errSimilarityFB has problem != from C++

// filter out points with fwd-back error > the median AND points with similarity error > median

return filterPts(lastPoints, currPoints, errSimilarity, errSimilarityFB, status);

}

/\*\*

\* @return real similarities errors

\*/

private float[] normCrossCorrelation(final Mat lastImg, final Mat currentImg, final Point[] lastPoints, final Point[] currentPoints, final byte[] status){

final float[] similarity = new float[lastPoints.length];

final Mat lastPatch = new Mat(CROSS\_CORR\_PATCH\_SIZE, CvType.CV\_8U);

final Mat currentPatch = new Mat(CROSS\_CORR\_PATCH\_SIZE, CvType.CV\_8U);

final Mat res = new Mat(new Size(1, 1), CvType.CV\_32F);

for(int i = 0; i < lastPoints.length; i++){

if(status[i] == 1){

Imgproc.getRectSubPix(lastImg, CROSS\_CORR\_PATCH\_SIZE, lastPoints[i], lastPatch);

Imgproc.getRectSubPix(currentImg, CROSS\_CORR\_PATCH\_SIZE, currentPoints[i], currentPatch);

Imgproc.matchTemplate(lastPatch, currentPatch, res, Imgproc.TM\_CCOEFF\_NORMED);

similarity[i] = Util.getFloat(0, 0, res);

}else{

similarity[i] = 0f;

}

}

return similarity;

}

/\*\*

\* @return Pair of new, FILTERED, last and current POINTS. Null if none were valid (with similarity > median and FB error <= median)

\*/

private Pair<Point[], Point[]> filterPts(final Point[] lastPoints, final Point[] currentPoints, final float[] similarity, final float[] errFB, final byte[] status){

final List<Point> filteredLastPoints = new ArrayList<Point>();

final List<Point> filteredCurrentPoints = new ArrayList<Point>();

final List<Float> filteredErrFB = new ArrayList<Float>();

final float similarityMed = Util.median(similarity);

Log.i(Util.TAG, "Filter points MED SIMILARITY: " + similarityMed);

for(int i = 0; i < currentPoints.length; i++){

if(status[i] == 1 && similarity[i] > similarityMed){

filteredLastPoints.add(lastPoints[i]);

filteredCurrentPoints.add(currentPoints[i]);

filteredErrFB.add(errFB[i]);

}

}

final List<Point> filteredLastPoints2 = new ArrayList<Point>();

final List<Point> filteredCurrentPoints2 = new ArrayList<Point>();

if(filteredErrFB.size() > 0){

errFBMed = Util.median(filteredErrFB);

for(int i = 0; i < filteredErrFB.size(); i++){

// status has already been checked

if(filteredErrFB.get(i) <= errFBMed){

filteredLastPoints2.add(filteredLastPoints.get(i));

filteredCurrentPoints2.add(filteredCurrentPoints.get(i));

}

}

Log.i(Util.TAG, "Filter points MED ErrFB: " + errFBMed + " K count=" + filteredLastPoints2.size());

}

final int size = filteredLastPoints2.size();

return size > 0 ? new Pair<Point[], Point[]>(filteredLastPoints2.toArray(new Point[size]), filteredCurrentPoints2.toArray(new Point[size])) : null;

}

float getMedianErrFB(){

return errFBMed;

}

}

**NNClassifier.java**

package com.trandi.opentld.tld;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

import java.util.Properties;

import org.opencv.core.CvType;

import org.opencv.core.Mat;

import org.opencv.imgproc.Imgproc;

import android.util.Log;

import com.trandi.opentld.tld.Parameters.ParamsClassifiers;

import com.trandi.opentld.tld.Tld.DetectionStruct;

import com.trandi.opentld.tld.Util.IsinStruct;

import com.trandi.opentld.tld.Util.NNConfStruct;

/\*\*

\*

\* Nearest Neighbour classifier

\*

\*/

public class NNClassifier {

ParamsClassifiers params;

final List<Mat> pExamples = new ArrayList<Mat>();

final List<Mat> nExamples = new ArrayList<Mat>();

NNClassifier(Properties props) {

params = new ParamsClassifiers(props);

}

/\*\*

\* OUTPUT (updates) : pExamples, nExamples

\*/

void trainNN(final Mat pExampleIn, final List<Mat> nExamplesIn){

NNConfStruct nnConf = nnConf(pExampleIn);

if(nnConf.relativeSimilarity <= params.pos\_thr\_nn){

if(nnConf.isin == null || nnConf.isin.idxPosSet < 0){

pExamples.clear();

}

pExamples.add(pExampleIn);

}

for(Mat nEx : nExamplesIn){

nnConf = nnConf(nEx);

if(nnConf.relativeSimilarity > params.neg\_thr\_nn){

nExamples.add(nEx);

}

}

Log.i(Util.TAG, "Trained NN examples: " + pExamples.size() + " positive " + nExamples.size() + " negative");

}

/\*\*

\* INPUTs : pExamples, nExamples

\* @param example NN patch

\* @return Relative Similarity (rsconf), Conservative Similarity (csconf), In pos. set|Id pos set|In neg. set (isin)

\*/

NNConfStruct nnConf(final Mat example) {

if(example == null){

Log.e(Util.TAG, "NNClass.nnConf() - Null example received, stop here");

return new NNConfStruct(null, 0, 0);

}

if(pExamples.isEmpty()){

// IF positive examples in the model are not defined THEN everything is negative

return new NNConfStruct(null, 0, 0);

}

if(nExamples.isEmpty()){

// IF negative examples in the model are not defined THEN everything is positive

return new NNConfStruct(null, 1, 1);

}

final Mat ncc = new Mat(1, 1, CvType.CV\_32F);

float nccP=0, csmaxP=0, maxP=0;

boolean anyP = false;

int maxPidx = 0;

final int validatedPart = (int) Math.ceil(pExamples.size() \* params.valid);

for(int i = 0; i < pExamples.size(); i++){

Imgproc.matchTemplate(pExamples.get(i), example, ncc, Imgproc.TM\_CCORR\_NORMED); // measure NCC to positive examples

nccP = (Util.getFloat(0, 0, ncc) + 1) \* 0.5f;

if(nccP > params.ncc\_thesame){

anyP = true;

}

if(nccP > maxP){

maxP = nccP;

maxPidx = i;

if(i < validatedPart){

csmaxP = maxP;

}

}

}

float nccN=0, maxN = 0;

boolean anyN = false;

for(int i = 0; i < nExamples.size(); i++){

Imgproc.matchTemplate(nExamples.get(i), example, ncc, Imgproc.TM\_CCORR\_NORMED); //measure NCC to negative examples

nccN = (Util.getFloat(0, 0, ncc) + 1) \* 0.5f;

if(nccN > params.ncc\_thesame){

anyN = true;

}

if(nccN > maxN){

maxN = nccN;

}

}

//Log.i(Util.TAG, "nccP=" + nccP + ", nccN=" + nccN + ", csmaxP=" + csmaxP + ", maxP="+ maxP + ", maxN=" + maxN);

// put together the result

final float dN = 1 - maxN;

final float dPrelative = 1 - maxP;

final float dPconservative = 1 - csmaxP;

return new NNConfStruct(new IsinStruct(anyP, maxPidx, anyN), dN / (dN + dPrelative), dN / (dN + dPconservative));

}

/\*\*

\* Updates NN threshold in case the negative threshold are above them.

\* The Pos threshold has to be > to the negative one.

\*/

void evaluateThreshold(final List<Mat> nExamplesTest){

for(Mat ex : nExamplesTest){

final NNConfStruct nnConf = nnConf(ex);

if(nnConf.relativeSimilarity > params.pos\_thr\_nn){

params.pos\_thr\_nn = nnConf.relativeSimilarity;

}

}

if(params.pos\_thr\_nn > params.pos\_thr\_nn\_valid){

params.pos\_thr\_nn\_valid = params.pos\_thr\_nn;

}

}

float getNNThreshold(){

return params.pos\_thr\_nn;

}

float getNNThresholdValid(){

return params.pos\_thr\_nn\_valid;

}

}

**Parameters.java**

package com.trandi.opentld.tld;

import java.util.Properties;

class Parameters {

protected final Properties props;

static class ParamsTld extends Parameters{

int min\_win;

int patch\_size;

// initial parameters for positive examples

int num\_closest\_init;

int num\_warps\_init;

int noise\_init;

float angle\_init;

float shift\_init;

float scale\_init;

// update parameters for positive examples

int num\_closest\_update;

int num\_warps\_update;

int noise\_update;

float angle\_update;

float shift\_update;

float scale\_update;

// parameters for negative examples

float num\_bad\_patches;

float tracker\_stability\_FBerrMax;

protected ParamsTld(){

super(null);

}

ParamsTld(Properties props) {

super(props);

// Bounding Box Parameters

min\_win = getInt("min\_win");

// Generator Parameters

// initial parameters for positive examples

patch\_size = getInt("patch\_size");

num\_closest\_init = getInt("num\_closest\_init");

num\_warps\_init = getInt("num\_warps\_init");

noise\_init = getInt("noise\_init");

angle\_init = getFloat("angle\_init");

shift\_init = getFloat("shift\_init");

scale\_init = getFloat("scale\_init");

// update parameters for positive examples

num\_closest\_update = getInt("num\_closest\_update");

num\_warps\_update = getInt("num\_warps\_update");

noise\_update = getInt("noise\_update");

angle\_update = getFloat("angle\_update");

shift\_update = getFloat("shift\_update");

scale\_update = getFloat("scale\_update");

// parameters for negative examples

num\_bad\_patches = getInt("num\_bad\_patches");

tracker\_stability\_FBerrMax = getFloat("tracker\_stability\_FBerrMax");

}

}

static class ParamsClassifiers extends Parameters {

double pos\_thr\_fern;

float neg\_thr\_fern;

int numFeaturesPerFern;

int numFerns;

float valid;

float ncc\_thesame;

float pos\_thr\_nn;

float pos\_thr\_nn\_valid;

float neg\_thr\_nn;

ParamsClassifiers(){

super(null);

}

ParamsClassifiers(Properties props) {

super(props);

valid = getFloat("valid");

ncc\_thesame = getFloat("ncc\_thesame");

numFerns = getInt("num\_ferns");

numFeaturesPerFern = getInt("num\_features\_per\_fern");

pos\_thr\_fern = getFloat("pos\_thr\_fern");

neg\_thr\_fern = getFloat("neg\_thr\_fern", 0.3f);

pos\_thr\_nn = getFloat("pos\_thr\_nn");

pos\_thr\_nn\_valid = getFloat("pos\_thr\_nn\_valid");

neg\_thr\_nn = getFloat("neg\_thr\_nn", 0.5f);

}

}

Parameters(Properties props) {

this.props = props;

}

protected int getInt(String propName){

if(props.containsKey(propName)){

return Integer.valueOf(props.getProperty(propName));

}

throw new IllegalArgumentException("Parameter " + propName + " has NOT been provided.");

}

protected float getFloat(String propName){

if(props.containsKey(propName)){

return Float.valueOf(props.getProperty(propName));

}

throw new IllegalArgumentException("Parameter " + propName + " has NOT been provided.");

}

protected float getFloat(String propName, float defaultValue){

if(props.containsKey(propName)){

return Float.valueOf(props.getProperty(propName));

}

return defaultValue;

}

}

**PatchGenerator.java**

package com.trandi.opentld.tld;

import org.opencv.core.Core;

import org.opencv.core.CvType;

import org.opencv.core.Mat;

import org.opencv.core.MatOfDouble;

import org.opencv.core.Point;

import org.opencv.core.Scalar;

import org.opencv.core.Size;

import org.opencv.imgproc.Imgproc;

import com.trandi.opentld.tld.Util.RNG;

class PatchGenerator {

final double backgroundMin;

final double backgroundMax;

final double noiseRange;

final boolean randomBlur;

final double lambdaMin;

final double lambdaMax;

final double thetaMin;

final double thetaMax;

final double phiMin;

final double phiMax;

PatchGenerator(double backgroundMin, double backgroundMax, double noiseRange, boolean randomBlur,

double lambdaMin, double lambdaMax, double thetaMin, double thetaMax, double phiMin, double phiMax )

{

this.backgroundMin = backgroundMin;

this.backgroundMax = backgroundMax;

this.noiseRange = noiseRange;

this.randomBlur = randomBlur;

this.lambdaMin = lambdaMin;

this.lambdaMax = lambdaMax;

this.thetaMin = thetaMin;

this.thetaMax = thetaMax;

this.phiMin = phiMin;

this.phiMax = phiMax;

}

void generate(final Mat image, Point pt, Mat patch, Size patchSize, final RNG rng) {

final Mat T = new MatOfDouble();

// TODO why is inverse not specified in the original C++ code

generateRandomTransform(pt, new Point((patchSize.width - 1) \* 0.5, (patchSize.height - 1) \* 0.5), T, false);

generate(image, T, patch, patchSize, rng);

}

/\*\*

\*

\* @param image

\* @param T

\* @param patch OUTPUT

\* @param patchSize

\*/

void generate(final Mat image, final Mat T, Mat patch, Size patchSize, final RNG rng){

patch.create( patchSize, image.type() );

if( backgroundMin != backgroundMax ) {

Core.randu(patch, backgroundMin, backgroundMax);

// TODO if that null scalar OK or should it be new Scalar(0) ?

Imgproc.warpAffine(image, patch, T, patchSize, Imgproc.INTER\_LINEAR, Imgproc.BORDER\_TRANSPARENT, null);

} else {

Imgproc.warpAffine(image, patch, T, patchSize, Imgproc.INTER\_LINEAR, Imgproc.BORDER\_CONSTANT, new Scalar(backgroundMin));

}

int ksize = randomBlur ? rng.nextInt() % 9 - 5 : 0;

if( ksize > 0 ) {

ksize = ksize \* 2 + 1;

Imgproc.GaussianBlur(patch, patch, new Size(ksize, ksize), 0, 0);

}

if( noiseRange > 0 ) {

final Mat noise = new Mat(patchSize, image.type());

int delta = (image.depth() == CvType.CV\_8U ? 128 : (image.depth() == CvType.CV\_16U ? 32768 : 0));

Core.randn(noise, delta, noiseRange);

// TODO this was different !!

Core.addWeighted(patch, 1, noise, 1, -delta, patch);

// if( backgroundMin != backgroundMax )

// addWeighted(patch, 1, noise, 1, -delta, patch);

// else

// {

// for( int i = 0; i < patchSize.height; i++ )

// {

// uchar\* prow = patch.ptr<uchar>(i);

// const uchar\* nrow = noise.ptr<uchar>(i);

// for( int j = 0; j < patchSize.width; j++ )

// if( prow[j] != backgroundMin )

// prow[j] = saturate\_cast<uchar>(prow[j] + nrow[j] - delta);

// }

// }

}

}

/\*\*

\*

\* @param srcCenter

\* @param dstCenter

\* @param transform OUTPUT

\* @param inverse

\*/

private void generateRandomTransform(Point srcCenter, Point dstCenter, Mat transform, boolean inverse) {

MatOfDouble tempRand = new MatOfDouble(0d, 0d);

Core.randu(tempRand, lambdaMin, lambdaMax);

final double[] rands = tempRand.toArray();

final double lambda1 = rands[0];

final double lambda2 = rands[1];

Core.randu(tempRand, thetaMin, thetaMax);

final double theta = tempRand.toArray()[0];

Core.randu(tempRand, phiMin, phiMax);

final double phi = tempRand.toArray()[0];

// Calculate random parameterized affine transformation A,

// A = T(patch center) \* R(theta) \* R(phi)' \* S(lambda1, lambda2) \* R(phi) \* T(-pt)

final double st = Math.sin(theta);

final double ct = Math.cos(theta);

final double sp = Math.sin(phi);

final double cp = Math.cos(phi);

final double c2p = cp\*cp;

final double s2p = sp\*sp;

final double A = lambda1\*c2p + lambda2\*s2p;

final double B = (lambda2 - lambda1)\*sp\*cp;

final double C = lambda1\*s2p + lambda2\*c2p;

final double Ax\_plus\_By = A\*srcCenter.x + B\*srcCenter.y;

final double Bx\_plus\_Cy = B\*srcCenter.x + C\*srcCenter.y;

transform.create(2, 3, CvType.CV\_64F);

transform.put(0, 0, A\*ct - B\*st);

transform.put(0, 1, B\*ct - C\*st);

transform.put(0, 2, -ct\*Ax\_plus\_By + st\*Bx\_plus\_Cy + dstCenter.x);

transform.put(1, 0, A\*st + B\*ct);

transform.put(1, 1, B\*st + C\*ct);

transform.put(1, 2, -st\*Ax\_plus\_By - ct\*Bx\_plus\_Cy + dstCenter.y);

if( inverse ){

Imgproc.invertAffineTransform(transform, transform);

}

}

}

**Tld.java**

package com.trandi.opentld.tld;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.Collections;

import java.util.Comparator;

import java.util.HashMap;

import java.util.Iterator;

import java.util.List;

import java.util.Map;

import java.util.Properties;

import org.opencv.core.Core;

import org.opencv.core.CvType;

import org.opencv.core.Mat;

import org.opencv.core.MatOfDouble;

import org.opencv.core.Point;

import org.opencv.core.Rect;

import org.opencv.core.Scalar;

import org.opencv.core.Size;

import org.opencv.imgproc.Imgproc;

import android.util.Log;

import com.trandi.opentld.tld.Parameters.ParamsTld;

import com.trandi.opentld.tld.Util.DefaultRNG;

import com.trandi.opentld.tld.Util.NNConfStruct;

import com.trandi.opentld.tld.Util.Pair;

import com.trandi.opentld.tld.Util.RNG;

public class Tld {

private static final int MAX\_DETECTED = 100;

ParamsTld \_params;

FernEnsembleClassifier \_classifierFern;

NNClassifier \_classifierNN;

private final LKTracker \_tracker = new LKTracker();

private PatchGenerator \_patchGenerator; // FIXME UNUSED, why !?

private final RNG \_rng = new DefaultRNG();

// Integral Images

private int \_iiRows;

private int \_iiCols;

private final Mat \_iisum = new Mat();

private final Mat \_iisqsum = new Mat();

// for performance reasons, duplicate the data directly in Java, to avoid too many native code invocations

private int[] \_iisumJava;

private double[] \_iisqsumJava;

private float \_var; // variance of the initial patch/box. Will be used by the 1st stage of the classifier.

// Training data

Mat \_pExample = new Mat(); // positive NN example

final List<Pair<int[], Boolean>> \_pFerns = new ArrayList<Pair<int[], Boolean>>(); //positive ferns <allFernsHashCodes, true>

private final List<Mat> \_pPatterns = new ArrayList<Mat>(); //positive patches to display

private List<Mat> \_nExamples;

// Last frame data

private BoundingBox \_lastbox;

private boolean \_learn = true;

// Detector data

private Map<BoundingBox, int[]> \_fernDetectionNegDataForLearning = new HashMap<BoundingBox, int[]>(); // all ferns hash codes for a given bounding box

final Map<DetectionStruct, Integer> \_boxClusterMap = new HashMap<DetectionStruct, Integer>(); // the cluster to which each detected box belongs

// Bounding Boxes Grid

Grid \_grid;

public Tld(Properties parameters){

\_params = new ParamsTld(parameters);

\_classifierFern = new FernEnsembleClassifier(parameters);

\_classifierNN = new NNClassifier(parameters);

\_patchGenerator = new PatchGenerator(0, 0, \_params.noise\_init, true, 1 - \_params.scale\_init, 1 + \_params.scale\_init,

-\_params.angle\_init \* Math.PI / 180f, \_params.angle\_init \* Math.PI / 180f,

-\_params.angle\_init \* Math.PI / 180f, \_params.angle\_init \* Math.PI / 180f);

\_pExample.create(\_params.patch\_size, \_params.patch\_size, CvType.CV\_64F);

}

protected Tld() {

// for TESTING only

}

public void init(Mat frame1, Rect trackedBox) {

// get Bounding boxes

if(Math.min(trackedBox.width, trackedBox.height) < \_params.min\_win) {

throw new IllegalArgumentException("Provided trackedBox: " + trackedBox + " is too small (min " + \_params.min\_win + ")");

}

\_grid = new Grid(frame1, trackedBox, \_params.min\_win);

Log.i(Util.TAG, "Init Created " + \_grid.getSize() + " bounding boxes.");

\_grid.updateGoodBadBoxes(trackedBox, \_params.num\_closest\_init);

\_iiRows = frame1.rows();

\_iiCols = frame1.cols();

\_iisum.create(\_iiRows, \_iiCols, CvType.CV\_32F);

\_iisqsum.create(\_iiRows, \_iiCols, CvType.CV\_64F);

// correct bounding box

\_lastbox = \_grid.getBestBox();

\_classifierFern.init(\_grid.getTrackedBoxScales(), \_rng);

// generate DATA

// generate POSITIVE DATA

generatePositiveData(frame1, \_params.num\_warps\_init, \_grid);

// Set variance threshold

MatOfDouble stddev = new MatOfDouble();

Core.meanStdDev(frame1.submat(\_grid.getBestBox()), new MatOfDouble(), stddev);

updateIntegralImgs(frame1);

// this is directly half of the variance of the initial box, which will be used the the 1st stage of the classifier

\_var = (float)Math.pow(stddev.toArray()[0], 2d) \* 0.5f;

// check variance

final double checkVar = Util.getVar(\_grid.getBestBox(), \_iisumJava, \_iisqsumJava, \_iiCols) \* 0.5;

Log.i(Util.TAG, "Variance: " + \_var + " / Check variance: " + checkVar);

// generate NEGATIVE DATA

final Pair<List<Pair<int[], Boolean>>, List<Mat>> negData = generateNegativeData(frame1);

// Split Negative Ferns <features, labels=false> into Training and Testing sets (they are already shuffled)

final int nFernsSize = negData.first.size();

final List<Pair<int[], Boolean>> nFernsTest = new ArrayList<Pair<int[], Boolean>>(negData.first.subList(0, nFernsSize/2));

final List<Pair<int[], Boolean>> nFerns = new ArrayList<Pair<int[], Boolean>>(negData.first.subList(nFernsSize/2, nFernsSize));

// Split Negative NN Examples into Training and Testing sets

final int nExSize = negData.second.size();

final List<Mat> nExamplesTest = new ArrayList<Mat>(negData.second.subList(0, nExSize/2));

\_nExamples = new ArrayList<Mat>(negData.second.subList(nExSize/2, nExSize));

//MERGE Negative Data with Positive Data and shuffle it

final List<Pair<int[], Boolean>> fernsData = new ArrayList<Pair<int[], Boolean>>(\_pFerns);

fernsData.addAll(nFerns);

Collections.shuffle(fernsData);

// TRAINING

Log.i(Util.TAG, "Init Start Training with " + fernsData.size() + " ferns, "

+ \_nExamples.size() + " nExamples, " + nFernsTest.size() + " nFernsTest, " + nExamplesTest.size() + " nExamplesTest");

\_classifierFern.trainF(fernsData, 10);

\_classifierNN.trainNN(\_pExample, \_nExamples);

// Threshold evaluation on testing sets

\_classifierFern.evaluateThreshold(nFernsTest);

\_classifierNN.evaluateThreshold(nExamplesTest);

}

private void updateIntegralImgs(Mat frame) {

Imgproc.integral2(frame, \_iisum, \_iisqsum);

// duplicate the data for performance reasons

\_iisumJava = Arrays.copyOf(Util.getIntArray(\_iisum), \_iiRows \* \_iiCols);

\_iisqsumJava = Arrays.copyOf(Util.getDoubleArray(\_iisqsum), \_iiRows \* \_iiCols);

}

public ProcessFrameStruct processFrame(final Mat lastImg, final Mat currentImg){

// 1. TRACK

TrackingStruct trackingStruct = null;

if(\_lastbox != null){

trackingStruct = track(lastImg, currentImg, \_lastbox);

}

// 2. DETECT

final Pair<List<DetectionStruct>, List<DetectionStruct>> detStructs = detect(currentImg);

// 3. INTEGRATION tracking with detection

if(trackingStruct != null){

\_lastbox = trackingStruct.predictedBB;

if(trackingStruct.conf > \_classifierNN.getNNThresholdValid()){

Log.i(Util.TAG, "Tracking confidence: " + trackingStruct.conf + " > " + " Threshold: " + \_classifierNN.getNNThresholdValid() + " ===> WILL LEARN");

\_learn = true;

}else{

Log.i(Util.TAG, "Tracking confidence: " + trackingStruct.conf + " < " + " Threshold: " + \_classifierNN.getNNThresholdValid() + " ===> WILL NOT LEARN");

}

Log.i(Util.TAG, "Tracked");

if(detStructs != null){

final Map<BoundingBox, Float> clusters = clusterConfidentIndices(detStructs.second);// cluster detections

Log.i(Util.TAG, "Found " + clusters.size() + " clusters");

final Map<BoundingBox, Float> confidentClusters = new HashMap<BoundingBox, Float>();

for(BoundingBox clusterBox : clusters.keySet()){

// Get clusters that are far from tracker and with better confidence

if(trackingStruct.predictedBB.calcOverlap(clusterBox) < 0.5 && clusters.get(clusterBox) > trackingStruct.conf){

confidentClusters.put(clusterBox, clusters.get(clusterBox));

}

}

if(confidentClusters.size() == 0){

Log.i(Util.TAG, "NO NN confident cluster !");

}else if(confidentClusters.size() == 1){

Log.i(Util.TAG, "Detected better match (1 confident cluster), re-initialising tracker");

\_lastbox = confidentClusters.keySet().iterator().next(); //bbnext

\_learn = false;

}else{

Log.i(Util.TAG, "Plenty of confident clusters detected. Get mean of close detections (use nnMatches)");

int cx=0,cy=0,cw=0,ch=0, close\_detections=0;

for(DetectionStruct detStruct : detStructs.second){

if(trackingStruct.predictedBB.calcOverlap(detStruct.detectedBB) > 0.7){

cx += detStruct.detectedBB.x;

cy += detStruct.detectedBB.y;

cw += detStruct.detectedBB.width;

ch += detStruct.detectedBB.height;

close\_detections++;

}

}

if(close\_detections > 0){

// weighted average (10 to 1 in favour of the tracked) trackers trajectory with the close detections

\_lastbox.x = Math.round((float)(10\*trackingStruct.predictedBB.x+cx)/(float)(10+close\_detections));

\_lastbox.y = Math.round((float)(10\*trackingStruct.predictedBB.y+cy)/(float)(10+close\_detections));

\_lastbox.width = Math.round((float)(10\*trackingStruct.predictedBB.width+cw)/(float)(10+close\_detections));

\_lastbox.height = Math.round((float)(10\*trackingStruct.predictedBB.height+ch)/(float)(10+close\_detections));

}

}

}

}else{ // IF NOT Tracking

Log.w(Util.TAG, "NOT Tracking");

\_lastbox = null;

\_learn = false;

if(detStructs != null){ // and detector is defined

final Map<BoundingBox, Float> clusters = clusterConfidentIndices(detStructs.second);// cluster detections

if(clusters.size() == 1){

// not tracking but detected exactly 1 cluster -> use this one as the best option

\_lastbox = clusters.keySet().iterator().next();

}

}

}

// 4. LEARN

if(\_learn){

\_learn = learn(currentImg, detStructs != null ? detStructs.first : null); // use the Fern classifier detected

}else{

Log.i(Util.TAG, "NOT Learning");

}

final Point[] lastPoints = (trackingStruct == null ? null : trackingStruct.lastPoints);

final Point[] currentPoints = (trackingStruct == null ? null : trackingStruct.currentPoints);

return new ProcessFrameStruct(lastPoints, currentPoints, \_lastbox);

}

private TrackingStruct track(final Mat lastImg, final Mat currentImg, final BoundingBox lastBox) {

Log.i(Util.TAG, "[TRACK]");

// Generate points

final Point[] lastPoints = lastBox.points();

if(lastPoints.length == 0){

Log.e(Util.TAG, "Points not generated from lastBox: " + lastBox);

return null;

}

// Frame-to-frame tracking with forward-backward error checking

final Pair<Point[], Point[]> trackedPoints = \_tracker.track(lastImg, currentImg, lastPoints);

if(trackedPoints == null){

Log.e(Util.TAG, "No points could be tracked.");

return null;

}

if(\_tracker.getMedianErrFB() > \_params.tracker\_stability\_FBerrMax){

Log.w(Util.TAG, "TRACKER too unstable. FB Median error: " + \_tracker.getMedianErrFB() + " > " + \_params.tracker\_stability\_FBerrMax);

// return null; // we hope the detection will find the pattern again

}

// bounding box prediction

final BoundingBox predictedBB = lastBox.predict(trackedPoints.first, trackedPoints.second);

if(predictedBB.x > currentImg.cols() || predictedBB.y > currentImg.rows()

|| predictedBB.br().x < 1 || predictedBB.br().y < 1)

{

Log.e(Util.TAG, "TRACKER Predicted bounding box out of range !");

return null;

}

// estimate Confidence

Mat pattern = new Mat();

try{

resizeZeroMeanStdev(currentImg.submat(predictedBB.intersect(currentImg)), pattern, \_params.patch\_size);

}catch(Throwable t){

Log.e(Util.TAG, "PredBB when failed: " + predictedBB);

}

//Log.i(Util.TAG, "Confidence " + pattern.dump());

//Conservative Similarity

final NNConfStruct nnConf = \_classifierNN.nnConf(pattern);

Log.i(Util.TAG, "Tracking confidence: " + nnConf.conservativeSimilarity);

Log.i(Util.TAG, "[TRACK END]");

return new TrackingStruct(nnConf.conservativeSimilarity, predictedBB, trackedPoints.first, trackedPoints.second);

}

/\*\*

\* Structure the classifier into 3 stages:

\* a) patch variance

\* b) ensemble of ferns classifier

\* c) nearest neighbour

\*/

private Pair<List<DetectionStruct>, List<DetectionStruct>> detect(final Mat frame){

Log.i(Util.TAG, "[DETECT]");

final List<DetectionStruct> fernClassDetected = new ArrayList<Tld.DetectionStruct>(); //dt

final List<DetectionStruct> nnMatches = new ArrayList<Tld.DetectionStruct>(); //dbb

// 0. Cleaning

\_boxClusterMap.clear();

// 1. DETECTION

final Mat img = new Mat(frame.rows(), frame.cols(), CvType.CV\_8U);

updateIntegralImgs(frame);

Imgproc.GaussianBlur(frame, img, new Size(9, 9), 1.5);

// Apply the Variance filter TODO : Bottleneck

int a=0;

for(BoundingBox box : \_grid){

// a) speed up by doing the features/ferns check ONLY if the variance is high enough !

if(Util.getVar(box, \_iisumJava, \_iisqsumJava, \_iiCols) >= \_var ){

a++;

final Mat patch = img.submat(box);

final int[] allFernsHashCodes = \_classifierFern.getAllFernsHashCodes(patch, box.scaleIdx);

final double averagePosterior = \_classifierFern.averagePosterior(allFernsHashCodes);

\_fernDetectionNegDataForLearning.put(box, allFernsHashCodes);// store for later use in learning

// b)

if(averagePosterior > \_classifierFern.getFernPosThreshold()){

fernClassDetected.add(new DetectionStruct(box, allFernsHashCodes, averagePosterior, patch));

}

}

}

Log.i(Util.TAG, a + " Bounding boxes passed the variance filter (" + \_var + ")");

Log.i(Util.TAG, fernClassDetected.size() + " Initial detected from Fern Classifier");

if(fernClassDetected.size() == 0){

Log.i(Util.TAG, "[DETECT END]");

return null;

}

// keep only the best

Util.keepBestN(fernClassDetected, MAX\_DETECTED, new Comparator<DetectionStruct>() {

@Override

public int compare(DetectionStruct detS1, DetectionStruct detS2) {

return Double.compare(detS1.averagePosterior, detS2.averagePosterior);

}

});

// 2. MATCHING using the NN classifier c)

for(DetectionStruct detStruct : fernClassDetected){

// update detStruct.patch to params.patch\_size and normalise it

Mat pattern = new Mat();

resizeZeroMeanStdev(detStruct.patch, pattern, \_params.patch\_size);

detStruct.nnConf = \_classifierNN.nnConf(pattern);

Log.i(Util.TAG, "NNConf: " + detStruct.nnConf.relativeSimilarity + " / " + detStruct.nnConf.conservativeSimilarity + " Threshold: " + \_classifierNN.getNNThreshold());

// only keep valid boxes

if(detStruct.nnConf.relativeSimilarity > \_classifierNN.getNNThreshold()){

nnMatches.add(detStruct);

}

}

Log.i(Util.TAG, "[DETECT END]");

return new Pair<List<DetectionStruct>, List<DetectionStruct>>(fernClassDetected, nnMatches);

}

private boolean learn(final Mat img, final List<DetectionStruct> fernClassDetected){

Log.i(Util.TAG, "[LEARN]");

Mat pattern = new Mat();

final double stdev = resizeZeroMeanStdev(img.submat(\_lastbox.intersect(img)), pattern, \_params.patch\_size);

final NNConfStruct confStruct = \_classifierNN.nnConf(pattern);

if(confStruct.relativeSimilarity < 0.5){

Log.w(Util.TAG, "Fast change, NOT learning");

return false;

}

if(Math.pow(stdev, 2) < \_var){

Log.w(Util.TAG, "Low variance, NOT learning");

return false;

}

if(confStruct.isin.inNegSet){

Log.w(Util.TAG, "Patch in negative data, NOT learning");

return false;

}

// Data generation

\_grid.updateGoodBadBoxes(\_lastbox, \_params.num\_closest\_update);

if(\_grid.getGoodBoxes().length > 0){

generatePositiveData(img, \_params.num\_warps\_update, \_grid);

}else{

Log.w(Util.TAG, "NO good boxes, NOT learning.");

return false;

}

// TODO why don't we learn from the GOOD boxes too !?

final List<Pair<int[], Boolean>> fernExamples = new ArrayList<Util.Pair<int[], Boolean>>(\_pFerns);

for(BoundingBox badBox : \_grid.getBadBoxes()){

final int[] allFernsHashCodes = \_fernDetectionNegDataForLearning.get(badBox);

if(allFernsHashCodes != null){

// these are NEGATIVE examples !

fernExamples.add(new Pair<int[], Boolean>(allFernsHashCodes, false));

}

}

final List<Mat> nnExamples = new ArrayList<Mat>();

if(fernClassDetected != null){

for(DetectionStruct detStruct : fernClassDetected){

if(\_lastbox.calcOverlap(detStruct.detectedBB) < Grid.BAD\_OVERLAP){

nnExamples.add(detStruct.patch);

}

}

}

// Classifiers update

\_classifierFern.trainF(fernExamples, 2);

\_classifierNN.trainNN(\_pExample, \_nExamples);

Log.i(Util.TAG, "[LEARN END]");

return true;

}

/\*\*

\*

\* @param conservativeSimilarities

\* @return Map of clusters' boxes and their confidence

\*/

private Map<BoundingBox, Float> clusterConfidentIndices(final List<DetectionStruct> conservativeSimilarities){

final int numbb = conservativeSimilarities.size();

if(numbb == 0){

Log.i(Util.TAG, "NO conservative similarities provided, NOTHING to cluster.");

return new HashMap<BoundingBox, Float>(); // empty result

}

// by default there is only 1 cluster, and ALL boxes are in it (0)

int clusters = 1;

for(DetectionStruct detStruct : conservativeSimilarities){

\_boxClusterMap.put(detStruct, 0);

}

if(numbb == 1){

return Collections.singletonMap(conservativeSimilarities.get(0).detectedBB, conservativeSimilarities.get(0).nnConf.conservativeSimilarity);

}else if(numbb == 2){

if(conservativeSimilarities.get(0).detectedBB.calcOverlap(conservativeSimilarities.get(1).detectedBB) < 0.5){

// 2nd box is in its own cluster, update

\_boxClusterMap.put(conservativeSimilarities.get(1), 1);

clusters = 2;

}

}else {

clusters = clusterBB();

}

final Map<BoundingBox, Float> result = new HashMap<BoundingBox, Float>();

for(int cluster = 0; cluster < clusters; cluster++){

float avgConservativeSimilarity = 0f;

int clusterBoxCount = 0, mx=0, my=0, mw=0, mh=0;

for(DetectionStruct detStruct : \_boxClusterMap.keySet()){

if(\_boxClusterMap.get(detStruct) == cluster){

avgConservativeSimilarity += detStruct.nnConf.conservativeSimilarity;

mx += detStruct.detectedBB.x;

my += detStruct.detectedBB.y;

mw += detStruct.detectedBB.width;

mh += detStruct.detectedBB.height;

clusterBoxCount++;

}

}

if(clusterBoxCount > 0){

final BoundingBox clusterBox = new BoundingBox();

clusterBox.x = mx / clusterBoxCount;

clusterBox.y = my / clusterBoxCount;

clusterBox.width = mw / clusterBoxCount;

clusterBox.height = mh / clusterBoxCount;

result.put(clusterBox, avgConservativeSimilarity / clusterBoxCount);

}

}

return result;

}

/\*\*

\* @param boxClusterMap INPUT / OUTPUT

\* @return Total clusters count

\*/

private int clusterBB(){

final int size = \_boxClusterMap.size();

// need the data in arrays

final DetectionStruct[] dbb = \_boxClusterMap.keySet().toArray(new DetectionStruct[size]);

final int[] indexes = new int[size];

for(int i = 0; i < size; i++){

indexes[i] = \_boxClusterMap.get(dbb[i]);

}

// 1. Build proximity matrix

final float[] data = new float[size \* size];

for(int i = 0; i < size; i++){

for(int j = 0; j < size; j++){

final float d = 1 - dbb[i].detectedBB.calcOverlap(dbb[j].detectedBB);

data[i \* size + j] = d;

data[j \* size + i] = d;

}

}

Mat D = new Mat(size, size, CvType.CV\_32F);

D.put(0, 0, data);

// 2. Initialise disjoint clustering

final int[] belongs = new int[size];

int m = size;

for(int i = 0; i < size; i++){

belongs[i] = i;

}

for(int it = 0; it < size - 1; it++){

//3. Find nearest neighbour

float min\_d = 1;

int node\_a = -1, node\_b = -1;

for (int i = 0; i < D.rows(); i++){

for (int j = i + 1 ;j < D.cols(); j++){

if (data[i \* size + j] < min\_d && belongs[i] != belongs[j]){

min\_d = data[i \* size + j];

node\_a = i;

node\_b = j;

}

}

}

// are we done ?

if (min\_d > 0.5){

int max\_idx =0;

for (int j = 0; j < size; j++){

boolean visited = false;

for(int i = 0; i < 2 \* size - 1; i++){

if (belongs[j] == i){

// populate the correct / aggregated cluster

indexes[j] = max\_idx;

visited = true;

}

}

if (visited){

max\_idx++;

}

}

// update the main map before going back

for(int i = 0; i < size; i++){

\_boxClusterMap.put(dbb[i], indexes[i]);

}

return max\_idx;

}

//4. Merge clusters and assign level

if(node\_a >= 0 && node\_b >= 0){ // this should always BE true, otherwise we would have returned

for (int k = 0; k < size; k++){

if (belongs[k] == belongs[node\_a] || belongs[k] == belongs[node\_b])

belongs[k] = m;

}

m++;

}

}

// there seem to be only 1 cluster

for(int i = 0; i < size; i++){

\_boxClusterMap.put(dbb[i], 0);

}

return 1;

}

/\*\* Inputs:

\* - Image

\* - bad\_boxes (Boxes far from the bounding box)

\* - variance (pEx variance)

\* Outputs

\* - Negative fern features (nFerns)

\* - Negative NN examples (nExample)

\*/

private Pair<List<Pair<int[], Boolean>>, List<Mat>> generateNegativeData(final Mat frame){

final List<Pair<int[], Boolean>> negFerns = new ArrayList<Pair<int[], Boolean>>();

final List<Mat> negExamples = new ArrayList<Mat>();

final List<BoundingBox> badBoxes = Arrays.asList(\_grid.getBadBoxes());

Collections.shuffle(badBoxes);

Log.w(Util.TAG, "ST");

// Get Fern Features of the boxes with big variance (calculated using integral images)

for(BoundingBox badBox : badBoxes){

if(Util.getVar(badBox, \_iisumJava, \_iisqsumJava, \_iiCols) >= \_var \* 0.5f){

final Mat patch = frame.submat(badBox);

final int[] allFernsHashCodes = \_classifierFern.getAllFernsHashCodes(patch, badBox.scaleIdx);

negFerns.add(new Pair<int[], Boolean>(allFernsHashCodes, false));

}

}

// select a hard coded number of negative examples

Iterator<BoundingBox> bbIt = badBoxes.iterator();

for(int i = 0; i < \_params.num\_bad\_patches && bbIt.hasNext(); i++){

final Mat pattern = new Mat();

final Mat patch = frame.submat(bbIt.next());

resizeZeroMeanStdev(patch, pattern, \_params.patch\_size);

negExamples.add(pattern);

}

Log.i(Util.TAG, "Negative examples generated. Ferns count: " + negFerns.size() + ". negEx count: " + negExamples.size());

return new Pair<List<Pair<int[],Boolean>>, List<Mat>>(negFerns, negExamples);

}

/\*\*

\* Generate Positive data

\* Inputs:

\* - good\_boxes

\* - best\_box

\* - bbhull

\* Outputs:

\* - Positive fern features (pFerns)

\* - Positive NN examples (pExample)

\*/

void generatePositiveData(final Mat frame, final int numWarps, final Grid aGrid) {

resizeZeroMeanStdev(frame.submat(aGrid.getBestBox()), \_pExample, \_params.patch\_size);

//Get Fern features on warped patches

final Mat img = new Mat();

Imgproc.GaussianBlur(frame, img, new Size(9, 9), 1.5);

final BoundingBox bbhull = aGrid.getBBhull();

final Mat warped = img.submat(bbhull);

// centre of the hull

final Point pt = new Point(bbhull.x + (bbhull.width - 1) \* 0.5f, bbhull.y + (bbhull.height - 1) \* 0.5f);

\_pFerns.clear();

\_pPatterns.clear();

for(int i = 0; i < numWarps; i++){

if(i > 0){

// this is important as it introduces the necessary noise / fuziness in the initial examples such that the Fern classifier recognises similar shapes not only Exact ones !

// warped is a reference to a subset of the img data, so this will affect the img object

\_patchGenerator.generate(frame, pt, warped, bbhull.size(), \_rng);

}

final BoundingBox[] goodBoxes = aGrid.getGoodBoxes();

for(BoundingBox goodBox : goodBoxes){

final Mat patch = img.submat(goodBox);

final int[] allFernsHashCodes = \_classifierFern.getAllFernsHashCodes(patch, goodBox.scaleIdx);

\_pFerns.add(new Pair<int[], Boolean>(allFernsHashCodes, true));

// // this will be used for display only

// final Mat tempPattern = new Mat();

// Imgproc.resize(patch, tempPattern, new Size(\_params.patch\_size, \_params.patch\_size));

// \_pPatterns.add(tempPattern);

}

}

Log.i(Util.TAG, "Positive examples generated( ferns: " + \_pFerns.size() + " NN: 1/n )");

}

/\*\*

\* Output: resized zero-mean patch/pattern

\* @param inImg INPUT, outPattern OUTPUT

\* @return stdev

\*/

private static double resizeZeroMeanStdev(final Mat inImg, Mat outPattern, int patternSize){

if(inImg == null || outPattern == null){

return -1;

}

Imgproc.resize(inImg, outPattern, new Size(patternSize, patternSize));

final MatOfDouble mean = new MatOfDouble();

final MatOfDouble stdev = new MatOfDouble();

Core.meanStdDev(outPattern, mean, stdev);

outPattern.convertTo(outPattern, CvType.CV\_32F);

Core.subtract(outPattern, new Scalar(mean.toArray()[0]), outPattern);

return stdev.toArray()[0];

}

public List<Mat> getPPatterns(){

return \_pPatterns;

}

static final class DetectionStruct {

public final BoundingBox detectedBB;

public final int[] pattern;

public final double averagePosterior;

public final Mat patch;

public NNConfStruct nnConf;

DetectionStruct(BoundingBox detectedBB, int[] pattern, double averagePosterior, Mat patch) {

this.detectedBB = detectedBB;

this.pattern = pattern;

this.averagePosterior = averagePosterior;

this.patch = patch;

}

}

private static final class TrackingStruct {

public final float conf;

public final BoundingBox predictedBB;

public final Point[] lastPoints;

public final Point[] currentPoints;

TrackingStruct(float conf, BoundingBox predictedBB, Point[] trackedLastPoints, Point[] trackedCurrentPoints) {

this.conf = conf;

this.predictedBB = predictedBB;

this.lastPoints = trackedLastPoints;

this.currentPoints = trackedCurrentPoints;

}

}

public static final class ProcessFrameStruct {

public final Point[] lastPoints;

public final Point[] currentPoints;

public final BoundingBox currentBBox;

ProcessFrameStruct(Point[] lastPoints, Point[] currentPoints, BoundingBox currentBBox) {

this.lastPoints = lastPoints;

this.currentPoints = currentPoints;

this.currentBBox = currentBBox;

}

}

}

**Util.java**

package com.trandi.opentld.tld;

import java.util.Arrays;

import java.util.Collections;

import java.util.Comparator;

import java.util.List;

import java.util.Random;

import org.opencv.core.CvType;

import org.opencv.core.Mat;

import org.opencv.core.Point;

public class Util {

public final static String TAG = "OpenTLD";

private final static byte[] \_byteBuff1 = new byte[1];

private final static int[] \_intBuff1 = new int[1];

private final static float[] \_floatBuff1 = new float[1];

private final static double[] \_doubleBuff1 = new double[1];

// this is for a whole matrix !

private static byte[] \_byteBuff = new byte[1];

private static int[] \_intBuff = new int[1];

private static float[] \_floatBuff = new float[1];

private static double[] \_doubleBuff = new double[1];

static int unsignedChar(int val){

return Math.min(Math.max(val, 0), 255);

}

static int unsignedChar(double val){

return unsignedChar((int)val);

}

/\*\*

\* This is actually quite slow, as it makes a couple of native calls for each of the Util.getXXX methods...

\*/

@Deprecated

static double getVar(final BoundingBox box, final Mat sum, final Mat sqsum){

final int brs = Util.getInt(box.y + box.height, box.x + box.width, sum);

final int bls = Util.getInt(box.y + box.height, box.x, sum);

final int trs = Util.getInt(box.y, box.x + box.width, sum);

final int tls = Util.getInt(box.y, box.x, sum);

final double brsq = Util.getDouble(box.y + box.height, box.x + box.width, sqsum);

final double blsq = Util.getDouble(box.y + box.height, box.x, sqsum);

final double trsq = Util.getDouble(box.y, box.x + box.width, sqsum);

final double tlsq = Util.getDouble(box.y, box.x, sqsum);

final double boxArea = box.area();

final double mean = (brs + tls - trs - bls) / boxArea;

final double sqmean = (brsq + tlsq - trsq - blsq) / boxArea;

return sqmean - mean \* mean;

}

/\*\*

\* Preferred for performance !

\* Here we get both the SUM and SQuaredSUM Matrices already in Java, rather than call native methods for each

\* element that we need.

\*

\* For a 320x240 frame the improvement is close to 5X !!!

\*/

static double getVar(final BoundingBox box, final int[] sum, final double[] sqsum, final int colCount) {

final int brs = sum[(box.y + box.height) \* colCount + box.x + box.width];

final int bls = sum[(box.y + box.height) \* colCount + box.x];

final int trs = sum[box.y \* colCount + box.x + box.width];

final int tls = sum[box.y \* colCount + box.x];

final double brsq = sqsum[(box.y + box.height) \* colCount + box.x + box.width];

final double blsq = sqsum[(box.y + box.height) \* colCount + box.x];

final double trsq = sqsum[box.y \* colCount + box.x + box.width];

final double tlsq = sqsum[box.y \* colCount + box.x];

final double boxArea = box.area();

final double mean = (brs + tls - trs - bls) / boxArea;

final double sqmean = (brsq + tlsq - trsq - blsq) / boxArea;

return sqmean - mean \* mean;

}

static float median(float[] vals){

final float[] newVals = Arrays.copyOf(vals, vals.length);

Arrays.sort(newVals);

return newVals[newVals.length / 2];

}

static float median(List<Float> vals){

final Float[] newVals = vals.toArray(new Float[vals.size()]);

Arrays.sort(newVals);

return newVals[(int) Math.floor(newVals.length / 2d)];

}

static float norm(final Point p1, final Point p2){

final double dX = p1.x - p2.x;

final double dY = p1.y - p2.y;

return (float)Math.sqrt(dX \* dX + dY \* dY);

}

/\*\*

\* no std::nth\_element in Java so we'll sort the list. Less performant but we don't really care for the small lists we have

\*/

static <T> void keepBestN(List<T> list, final int n, final Comparator<T> comparator){

final int size = list.size();

if(size <= n) {

// nothing to do, sorting is not a requirement

return;

}

// sorts in ASCENDING ORDER

Collections.sort(list, comparator);

// we want the best / highest n so remote at the queue

while(list.size() > n){

list.remove(0);

}

}

static byte getByte(final int row, final int col, final Mat mat){

if(CvType.CV\_8UC1 != mat.type()) throw new IllegalArgumentException("Expected type is CV\_8UC1, we found: " + CvType.typeToString(mat.type()));

mat.get(row, col, \_byteBuff1);

return \_byteBuff1[0];

}

/\*\*

\* The corresponding Java primitive array type depends on the Mat type:

\* CV\_8U and CV\_8S -> byte[]

\* CV\_16U and CV\_16S -> short[]

\* CV\_32S -> int[]

\* CV\_32F -> float[]

\* CV\_64F-> double[]

\*/

static byte[] getByteArray(final Mat mat){

if(CvType.CV\_8UC1 != mat.type()) throw new IllegalArgumentException("Expected type is CV\_8UC1, we found: " + CvType.typeToString(mat.type()));

final int size = (int) (mat.total() \* mat.channels());

if(\_byteBuff.length != size){

\_byteBuff = new byte[size];

}

mat.get(0, 0, \_byteBuff); // 0 for row and col means the WHOLE Matrix

return \_byteBuff;

}

static int[] getIntArray(final Mat mat){

if(CvType.CV\_32SC1 != mat.type()) throw new IllegalArgumentException("Expected type is CV\_32SC1, we found: " + CvType.typeToString(mat.type()));

final int size = (int) (mat.total() \* mat.channels());

if(\_intBuff.length != size){

\_intBuff = new int[size];

}

mat.get(0, 0, \_intBuff); // 0 for row and col means the WHOLE Matrix

return \_intBuff;

}

static float[] getFloatArray(final Mat mat){

if(CvType.CV\_32FC1 != mat.type()) throw new IllegalArgumentException("Expected type is CV\_32FC1, we found: " + CvType.typeToString(mat.type()));

final int size = (int) (mat.total() \* mat.channels());

if(\_floatBuff.length != size){

\_floatBuff = new float[size];

}

mat.get(0, 0, \_floatBuff); // 0 for row and col means the WHOLE Matrix

return \_floatBuff;

}

static double[] getDoubleArray(final Mat mat){

if(CvType.CV\_64F != mat.type()) throw new IllegalArgumentException("Expected type is CV\_64F, we found: " + CvType.typeToString(mat.type()));

final int size = (int) (mat.total() \* mat.channels());

if(\_doubleBuff.length != size){

\_doubleBuff = new double[size];

}

mat.get(0, 0, \_doubleBuff); // 0 for row and col means the WHOLE Matrix

return \_doubleBuff;

}

static int getInt(final int row, final int col, final Mat mat){

if(CvType.CV\_32SC1 != mat.type()) throw new IllegalArgumentException("Expected type is CV\_32SC1, we found: " + CvType.typeToString(mat.type()));

mat.get(row, col, \_intBuff1);

return \_intBuff1[0];

}

static float getFloat(final int row, final int col, final Mat mat){

if(CvType.CV\_32F != mat.type()) throw new IllegalArgumentException("Expected type is CV\_32F, we found: " + CvType.typeToString(mat.type()));

mat.get(row, col, \_floatBuff1);

return \_floatBuff1[0];

}

static double getDouble(final int row, final int col, final Mat mat){

if(CvType.CV\_64F != mat.type()) throw new IllegalArgumentException("Expected type is CV\_64F, we found: " + CvType.typeToString(mat.type()));

mat.get(row, col, \_doubleBuff1);

return \_doubleBuff1[0];

}

static final class NNConfStruct {

final IsinStruct isin;

final float relativeSimilarity;

final float conservativeSimilarity;

NNConfStruct(IsinStruct isin, float relativeSimilarity, float conservativeSimilarity) {

this.isin = isin;

this.relativeSimilarity = relativeSimilarity;

this.conservativeSimilarity = conservativeSimilarity;

}

}

static final class IsinStruct {

final boolean inPosSet;

final int idxPosSet;

final boolean inNegSet;

IsinStruct(boolean inPosSet, int idxPosSet, boolean inNegSet) {

this.inPosSet = inPosSet;

this.idxPosSet = idxPosSet;

this.inNegSet = inNegSet;

}

}

static final class Pair<U, V>{

final U first;

final V second;

Pair(U first, V second){

this.first = first;

this.second = second;

}

@Override

public String toString(){

return "{" + first + ", " + second + "}";

}

}

static interface RNG {

float nextFloat();

int nextInt();

}

static class DefaultRNG implements RNG{

private final Random rnd = new Random();

@Override

public float nextFloat() {

return rnd.nextFloat();

}

@Override

public int nextInt() {

return rnd.nextInt();

}

}

}