# Розробка системи ідентифікації диму на відео для локалізації пожежі

1. Вивчення предметної області

1.1. Способи локалізації пожеж (аналіз)

1.1.1. Теплові датчики

1.1.2. Датчики вогню

1.1.3. Газові датчики

1.1.4. Димові оптичні датчики

1.1.5. Іонізаційні датчики

1.1.6. Ручні сповіщувачі

1.1.7. Ультразвукові сповіщувачі

1.2. Вивчення характеристик диму.

1.2.1. Flickering

1.2.3. Enegry analisys

2. Аналіз існуючих методів ідентифікації зображень. (найти методи для характеристик, анализ, обоснования выбраных методов для решение задачи) (теор часть)

3. Синтез метода (ов) для решения задачи для нахождения дима на изобр.)

4. Оценка эф. Метода (не требующая экспер., еффект т.к. не нужно доп вложений для покупки АО)

5. Разработка алгоритмов и прогр обеспеч.

6. Экспер оценка эф.

6.1. Порівняння покращеного метода з існуючими.

[Розділ 1. Способи локалізації пожеж 4](#_Toc387413999)

[Теплові датчики 4](#_Toc387414000)

[Датчики вогню 5](#_Toc387414001)

[Газові датчики 6](#_Toc387414002)

[Димові сенсори 7](#_Toc387414003)

[Оптичні димові сенсори 8](#_Toc387414004)

[Іонізаційні сенсори 12](#_Toc387414005)

[Ручні сповіщувачі 14](#_Toc387414006)

[Ультразвукові сповіщувачі 14](#_Toc387414007)

[Розділ 2. Існуючі методи 16](#_Toc387414008)

[Копіловіч 16](#_Toc387414009)

[Турейн 17](#_Toc387414010)

[Розділ 3. Запропонований метод 19](#_Toc387414011)

[Виділення фону 19](#_Toc387414012)

[Виділення мерехтіння 20](#_Toc387414013)

[Знаходження контуру 20](#_Toc387414014)

[Класифікація контуру 20](#_Toc387414015)

[Розділ 4. Розробка ПЗ 21](#_Toc387414016)

[Мова програмування 21](#_Toc387414017)

[Основні можливості мови 22](#_Toc387414018)

[Відмінності від Lisp 22](#_Toc387414019)

[Джерела інформації про мову 23](#_Toc387414020)

[Установка і запуск 24](#_Toc387414021)

[З чого складається мова Clojure 25](#_Toc387414022)

[Основні типи даних 25](#_Toc387414023)

[Бібліотека комп’ютерного бачення 30](#_Toc387414024)

[Розділ 5. Експериментальні результати 34](#_Toc387414025)

[Розділ 6. Подальша робота над методом 35](#_Toc387414026)

[Розділ 7. Використана література 36](#_Toc387414027)

# Способи локалізації пожеж

## Теплові датчики

Даний тип датчиків є одним із найстаріших видів датчиків пожеж.

Він був розроблений в 1890 році Френсісом Аптоном. Даний датчик

складався із електричної батареї, колокольного куполу, магніту та

термостатичного пристрою. Термостатичний пристрій знаходив аномальну

кількість тепла. Після того як була знайдена температура вища ніж деякий поріг, контур між магнітом та батареєю замикався. При цьому молоточок ударяв по куполу, створюючи попереджувальний сигнал для людей, які знаходилися в приміщенні.

Існують декілька видів теплових датчиків: точечний, багатоточечний та лінійний.

Точечні датчики відрізняються від багатоточечних тим, що у останніх замість одного термостатичного елемента вбудовано декілька, які є рознесеними в просторі.

Ліній теплові датчики поділяються на напівпровідникові, механічні та електромеханічні. В лінійних напівпровідникових теплових датчиках в якості термостатичних елементів використовуються покриття проводів речовиною з від’ємним температурним коефіцієнтом. При взаємодії температури на кабель змінюється опір провідника в точці взаємодії. За зміною опору стежить блок керування. Механічні лінійні теплові датчики в якості термостатичного елемента використовують герметичну металічну трубку, заповнену газом, а також датчик тиску, приєднаний до блоку керування. При взаємодії температури в трубі змінюється тиск газу, який реєструє блок керування. Довжина труби має обмеження в розмірі 300 метрів. Електромеханічні лінійні датчики в якості теплового сенсора використовують термочутливий матеріал, який наноситься на два механічно натянуті проводи. При взаємодії з теплом, термочутливий матеріал розмягчається і два проводи накоротко замикаються.

Дані датчики використовуються тільки тоді, коли на початкових стадіях пожежі виділяється достатня кількість тепла. Для того, щоб датчик працював коректно він повинен бути розміщений на відстані 10-23 см від стелі, оскільки саме там знаходиться поле найбільшої температури

## Датчики вогню

Датчики вогню реагують на електромагнітне випромінювання вогню. Такі датчики характеризуються спектральною чутливістю. Спектральна чутливість - це чутливість фотоприймача до випромінювання з різними довжинами хвиль. Вона визначається природою матеріалу, з якого зроблений світлочутливий шар. Для вимірювання випромінювання у вузькому діапазоні використовуються світлофільтри. Вони представляють собою пластини, які пропускають світло визначеної довжини хвилі. Для видимої частини спектру використовуються кольорове скло, а для інфрачервоних фільтрів - слюда, кам’яна сіль, бромистий калій, тощо.

Існують дачтики вогню на 3 областях спекру електромагнітного випромінювання: видимого, ульрафіолетового та інфрачервоного випромінювання.

Датчики в області ультрафіолетовго випромінювання з’явилися не так давно, але з кожним днем вони стають все популярнішими. Такий ріст популярності викликаний тим, що земна атмосфера захищає нас від сонячних ультрафіолетових променів, в результаті чого до землі ніколи не доходять хвилі з довжиною меншою ніж 286 нм. Саме тому такі датчики не реагують на сонячне випромінювання, яке є джерелом дуже потужних шумів.

Датчики в області інфрачервоного спектра реагують на горіння матеріалів, які мають в своєму складі вуглевод. Так датчики здатні працювати в запилених приміщеннях, адже випромінювання в інфрачервоному спектрі дуже погано поглинається пилом. Для забезпечення стійкістю до світлового впливу необхідно, щоб максимум спектральної чутливості таких датчиків знаходився в діапазоні 2.7-4.3 мкм. Більшість датчиків, що випускається мають спектральні характеристики в набагато коротшому діапазоні інфрачервоного випромінювання, де в більшій степені відчувається вплив сонячного випромінювання та ламп накалювання.

Датчики, чутливість яких знаходиться в ближній області до інфрачервоного випромінювання, мають нижчу стійкість до перешкод ніж датчики з чутливістю в більш довгих хвилях електромагнітного випромінювання.

Одним з підвидів датчиків вогню є датчики вогню в інфрачервоному спектрі випромінювання, які реагують на ефект пульсації. Для реалізації таких датчиків необхідно мати приймач випромінювання, який зможе реєструвати коливання в діапазоні від 2 до 20 Гц. Популярність такого метода викликано тим, що в вогнищах пожеж, як правило, мають місце низькочастотні коливання інтенсивності випромінювання вогню.

## Газові датчики

Газовий датчик - це датчик, який реагує на гази, які виділяються при тлінні або горінні матеріалів. Газові датчики можуть реагувати на оксид вуглецю та на вуглецеві сполучення. Датчики повинні реагувати, як мінімум, на один із приведених нижче газів при вказаній концентрації:

\* $CO\_2$ - 1000..1500 ppm;

\* CO - 20..80 ppm;

\* $C\_xH\_y$ - 10..20 ppm;

Закис вуглецю являється основним характерним газовим компонентом, який виділяється на стадії тління в результаті піролізу матеріалів, які використовуються на будівництві. На початковій стадії пожежі, при тлінні, концентрація CO швидко збільшується, але при появі вогню різко падає, зате росте концентрація двоокису вуглецю. З іншої сторони, такий рівень $CO\_2$ досягається в результаті знаходження в приміщенні двох людей на протязі однієї години. Також при тлінні разом з окисом вуглецю виділяється й водень, який є відсутнім в звичайних умовах в атмосфері. Недивлячись на невелику концентрацію водню, який виділяється в повітря приміщення, його легко ідентифікувати при наявності високочутливих та селективних датчиків водню.

Датчики пожеж на основі газових сенсорів можуть попереджувати пожежі на найраніших стадіях горіння. Для ефективного використання газові сенсори повинні мати наступні властивості:

\* чутливість до вуглецю 0.00001..0.00002 %, для чадного газу 0.0001..0.008 %;

\* швидкодія 2..5 с;

\* низьким споживанням енергії (менше ніж 50 мВт);

\* довговічністю (10..60 тисяч годин);

\* стабільністю роботи;

\* низькою ціною (1..3 долара США);

На початковому етапі горіння, коли тліє ще невелика кількість матеріалу,”пожежні гази”розчиняється в об’ємі приміщення. Тому існую вимога, щодо низького порогу чутливості сенсорів. Вимога до швидкодії випливає зі швидкості дифузії газів та конвекції повітряних мас в приміщенні. Газові детектори пожеж, які використовуються в приватних будівлях та квартирах повинні також сигналізувати про витік газу з кухонних плит та систем газового опалення. Для можливості одночасного вимірювання низьких та високих концентрацій газу сенсор повинен мати динамічний діапазон в чотири порядки. Такі динамічні діапазони мають лише сенсори на напівпровідниках.

На сьогоднішній день актуальною задачею газових датчиків є розробка мініатюрних хімічних сенсорів на основі металооксидних напівпровідників. Найбільш широко використовується діоксид олова, який має високу хімічну стійкість. Принцип дії сенсорів на напівпровідниках полягає в тому, що зворотня хемосорбція активних газів на їх поверхні супроводжується зворотньою зміною провідності. Тобто, на поверхні, утвореною окисом олова, при зіткненні молекул СО відбувається хімічна реакція доокислення вуглецю: $SnO\_2 + 2\*CO = Sn + 2\*CO\_2$. Відповідно, при зростанні концентрації СО в повітрі, кількість молекул Sn росте, що призводить до пониження опору чутливості елемента. Для отримання неперервного процесу вимірювання СО необхідно періодично відновлювати поверхню сенсора шляхом доокислення раніше відновленого олова. Для реалізації цього процесу використовуються нагріваючий елемент, який розташований безпосередньо під поверхнею сенсора.

Реальна залежність опору від концентрації СО виражається нелінійною залежністю тому, що в швидкість хімічної реакції вносить значний вклад температура та вологість. Також важливо, щоб в процесі відновлення температура була постійною.

Висока чутливість до вмісту в атмосфері цілого ряду отруйних та вибухонебезпечних газів, а також можливість керування процесами, які протікають на поверхні та в середині напівпровідників, роблять ці матеріали особливо привабливими.

## Димові сенсори

Димові сенсори - це сенсори, які реагують на продукті горіння та можуть взаємодіяти на поглинаючу та розсіючу здатність в інфрачервоному, ультрафіолетовому та видимому діапазонах спектру. Основною ознакою пожежі, на яку реагує димовий сенсор є дим. Сенсори диму поділяться на оптичні та іонізаційні.

### Оптичні димові сенсори

Димові сенсори, які використовують оптичні засоби виявлення, реагують по-різному на дим різних кольорів. На сьогоднішній день виробники надають обмежену інформацію про реакцію димових сенсорів в технічних характеристиках. Такі сенсори поділяються на: точкові, лінійні, аспіраційні та автономні.

#### Точкові димові сенсори

Точковий димовий сенсор реагує на фактори пожежі в компактній зоні. Принцип дії точкових сенсорів оптичних сенсорів заснований на розсіюванні дими інфрачервоного випромінювання. Добре реагують на сірий дим, який виділяється при тлінні на ранніх стадіях пожежі. Погано реагує на чорний дим, який поглинає інфрачервоне випромінювання. Для періодичного обслуговування сенсорів необхідне роз’ємне з’єднання, так звана розетка з чотирма контактами, до якої підключається сенсор.

У всіх точкових оптичних димових сенсорах по класифікації НПБ 76-98 використовується ефект дифузного розсіювання випромінювання світлодіода на частках диму. Світлодіод розміщується таким чином, щоб виключити пряме попадання його випромінювання на фотодіод. При появі часток диму частина випромінювання відбивається від них та попадає на фотодіод. Для захисту від зовнішнього світла оптопара (світлодіод та фотодіод), розміщуються в димовій камері з пластику чорного кольору.

Конструкція димової камери повинна задовільняти суперечливим вимогам: забезпечувати вільний доступ для повітряних потоків та виключати вплив зовнішнього світла, електромагнітних перешкод та пилу. Форма пластинок, розміщених по периметру димової камери, вибираються виходячи з вимог максимального послаблення фонового освітлення як від світлодіода оптопари, так і від зовнішніх джерел. Прямі промені світла повинні бути поглинутими при багатократному відбиванні від поверхні пластинок. Плавні згини пластинок повинні не вносити значних змін в направлення повітряних потоків та забезпечувати вентиляцію димової камери.

Експериментальні дослідження показали, що час виявлення тестової пожежі при розміщенні димового датчика на відстані 300 см від стелі збільшується в 2..5 разів.

Коли розроблялися перші радянські оптичні димові датчики, не було спеціалізованої елементної бази, стандартних світлодіодів та фотодіодів. В димовому фотоелектричному датчику ІДФ-1М в якості оптопари використовувались лампа накалювання типу СГ24-1,2 та фоторезистор типу ФСК-Г1. Це визначало низькі технічні характеристики детектора ІДФ-1М та слабий захист від зовнішніх впливів. В комбінованому димо-тепловому детекторі ДІП-1 були використані світлодіод та фотодіод, причому вони були розміщені в вертикальній площині. Використовувалося вже не безперервне випромінювання, а імпульсне: тривалість 30 мкс, частота 300 Гц. Для захисту від перешкод було використана синхронна детекція, тобто вхід підсилювача був відкритий тільки під час випромінювання світлодіода. Це забезпечувало більш ефективний захист від перешкод.

#### Лінійні оптичні димові сенсори

Лінійний оптичний димовий сенсор - це сенсор, який складається з блоку приймача та випромінювача. Вони можуть знаходитися, як в одному блоці, так і в різних. Такий сенсор реагує на появу диму між блоком приймача та випромінювача. Принцип дії лінійних оптичних димових сенсорів заснований на принципі послаблення електромагнітного поля між рознесеними в просторі джерелом випромінювання та фотоприймачем під впливом часток диму. Прилад такого типу складається з двох блоків, один з яких має джерело електромагнітного випромінювання, а інший - фотоприймач. Обидва блоки розміщуються на одній геометричній осі в зоні прямої видимості.

Особливістю всіх лінійних оптичних димових сенсорів є функція самотестування з передачею сигналу”Несправність”прийомо-контрольному приладу.

Одним із перших радянських лінійний оптичних димових сенсорів мав назву ДОП-1 та використовував в якості джерела світла лампу накалювання СГ-24-1,2. В якості фотоприймача використовувався германієвий фотодіод. Сенсор складався із прийомо-передаваючого блока, який використовувся для випромінювання та прийому світлового променя, та світловідбивача, установленому перпендикулярно направленому світловому променю. Номінальна відстань між прийомо-передаваючим блоком та відбивачем є $2,5\pm0.1$ м.

#### Аспіраційний оптичний димовий сенсор

Аспіраційний оптичний димовий сенсор - це сенсор, який використовує примусовий відбір повітря (аспірацію) для моніторингу ультрачутливими лазерами або оптичними димовими сенсорами. Такий сенсор забезпечує знаходження диму в найбільш ранній момент пожежі. Аспіраційні димові сенсори дозволяють захистити об’єкти, в яких неможливо безпосередньо розмістити пожежний захист.

Такі сенсори можуть використовуватися при необхідності захисту великий об’єктів, наприклад - торгові центри, але основна область використання це жорсткі умови експлуатації:

* висока вологість
* високі температури
* низькі температури
  + холодильні установки
  + використання зовні приміщень
* високі стелі
* важкодоступні приміщення
  + кабельні тунелі
  + вибухонебезпечні зони
  + високовольтні приміщення
* приміщення з ризиком навмисного пошкодження
  + тюремні камери
* контроль приладів
  + електричні шафи
  + телекомунікаційне обладнання
* запилені приміщення
  + сміттєпроводи

На долю аспіраційний систем на сьогоднішній день припадає 7% європейського ринку пожежних систем.

Аспіраційні сенсори поділяються на класи чутливості:

* Клас А, висока чутливість;
* Клас В, підвищена чутливість;
* Клас С, стандартна чутливість.

Аспіраційні сенсори класу А, В рекомендуються для захисту великих відкритих просторів та приміщень з висотою більш ніж 8 метра.

В світовій практиці прийнято ділити все аспіраційні сенсори на два класи, схожі по назві, але різні по своїм функціям та області використання:

\*”VESDA”-”Very Early Smoke Detection Apparatus”("Апаратура для дуже раннього знаходження диму"), сюди відносять сенсорі, які призначені для знаходження пожежі на її ранній стадії. В таких сенсорах використовується лазерній принцип знаходження диму. За рахунок використання лазерних технологій вони мають можливість знаходити дим в незначних концентраціях (від 0.0015 $%/м^3$.

\*”PIB”-”Point In the Box”("Точковий сенсор в коробці"). Це звичайний оптико-електронний пожежний сенсор, поміщений в коробку з вмонтованим вентилятором для реалізації принципу аспірації.

Висока чутливість дозволяє використовувати радіоізотопні сенсори як компонент аспіраційних сенсорів. При прокачці повітря в захищених приміщеннях він може забезпечувати подачу сигналу при появі найменшої концентрації диму - від 0.1 $мг/м^3$. При цьому довжина трубок для забору повітря практично майже не обмежується. Наприклад, практично завжди реєструється факт горіння сірникової головки на вході повітрозаборної труби довжиною в 100м.

Час транспортування проби повітря від максимально віддаленого від блока обробки димозатягуючого отвору до технічних засобів знаходження диму в залежності від класу сенсору не повинні перевищувати:

* для класу А - 60 с;
* для класу В - 90с;
* для класу С - 120 с.

Система повинна забезпечувати неперервний контроль. Можливий варіант послідовного сканування приміщень, при виконанні умов, що інтервал між сканування одного й того ж приміщення не повинен перевищувати 120 с и тривалість одного сканування перевищує час знаходження пожежі з запасом 20%.

Сканування приміщення може виконуватися при одній димовій камері при автоматичному переключанню заборних труб в процесі функціонування. В сенсорі в черговому режимі роботи відкриті всі вентилі заборних труб і повітря в рівних пропорціях поступають в димову камеру блока аналізу сенсора При знаходженні ним ознак диму, вентилі починають закриватися, й на основі логічної обробки визначається напрям, з якого розповсюджується дим. При груповому відключенні трубок після знаходження диму відключаються половина трубок, після аналізу - ще одна половина, поки не залишиться одна трубка. При використанні метода поодинокого відключення час знаходження різко зростає при кількості трубок більшої ніж 4. Тому в такому випадку краще використовувати метод групового відключення.

Існує спеціальна конструкція аспіраційних сенсорів для роботи в запилених приміщеннях. Такі сенсори мають систему фільтрації контрольованого середовища. Для більш високого ступеня очищення використовується технологія, коли до того, як проба повітря попадає в оптичну камеру знаходження диму, за допомогою фільтру видаляється пил. Потім, на другій стадії очищення забезпечується додаткова подача порції чистого повітря для попередження забруднення оптичних поверхонь та для забезпечення стабілізації калібровки та довготривалого строку служби аспіраційного сенсора. В подальшому аспіраційні сенсори за допомогою контактів реле або інтерфейсу можуть передавати сигнал про пожежу.

### Іонізаційні сенсори

Принцип дії іонізаційних сенсорів заснований на регістрації змін іонізаційного струму, який з’являється в результаті впливу на нього продуктів горіння. Іонізаційні сповіщувачі поділяються на радіоізотопні та електроіндукційні.

#### Радіоізотопний іонізаційний сенсор

Радіоізотопний сповіщувач - це димовий пожежний сповіщувач, який спрацьовує внаслідок впливу продуктів горіння на іонізаційний струм внутрішньої робочої камери сповіщувача. Принцип дії радіоізотопного сповіщувача заснований на іонізації повітря камери при опроміненні його радіоактивною речовиною. При введенні в таку камеру протилежно заряджених електродів виникає іонізаційний струм. Заряджені частинки”прилипають”до більш тяжких частинкам диму, знижуючи свою рухливість - іонізаційний струм зменшується. Його зменшення до певного значення сповіщувач сприймає як сигнал”тривога". Подібний сповіщувач ефективний у димах будь-якої природи. Однак поряд з описаними вище чеснотами радіоізотопні сповіщувачі мають істотний недолік, про який не слід забувати. Мова йде про використання в конструкції сповіщувачів джерела радіоактивного випромінювання. У зв'язку з цим виникають проблеми дотримання заходів безпеки при експлуатації, зберіганні і транспортуванні, а також утилізації сповіщувачів після закінчення терміну експлуатації. Ефективний для виявлення загорянь, що супроводжуються появою так званих”чорних”видів диму, що характеризуються високим рівнем поглинання світла.

У радянських радіоізотопних сповіщувачах (РІД-1, КІ) джерелом іонізації був радіоактивний ізотоп плутонію-239. Сповіщувачі входять у першу групу потенційної радіаційної небезпеки.

Основним елементом радіоізотопного сповіщувача РІД-1 є дві іонізаційні камери, включені послідовно. Точка з'єднання підключена до керуючого електроду тиратрона. Одна з камер є відкритою, інша закрита і виконує роль компенсуючого елемента. Іонізація повітря в обох камерах створюється ізотопом плутонію. Під дією прикладеної напруги в камерах протікає іонізаційний струм. При попаданні диму у відкриту камеру її провідність зменшується, напруга на обох камерах перерозподіляється, в результаті чого виникає напруга на керуючому електроді тиратрона. При досягненні напруги запалювання тиратрон починає проводити струм. Збільшення споживання струму приводить до спрацьовування сигналізації. Вбудовані в сповіщувач джерела радіації не представляють небезпеки, так як випромінювання повністю поглинається в об'ємі іонізаційними камерами. Небезпека може виникнути тільки при порушенні цілісності джерела випромінювання. Також в сенсорі використовується тиратрон ТХ11Г c незначною кількістю радіоактивного нікелю, випромінювання поглинається об'ємом тиратрона і його стінками. Небезпека може виникнути при розбитті тиратрона.

Призначеним терміном служби радіоактивних джерел сповіщувачів становив:

* РІД-1; КИ-1; ДІ-1 - 6 років;
* РІД-6; РІД-6 м та подібні - 10 років.

Радіоізотопний димовий пожежний сповіщувач типу РІД-6М більше 15 років серійно вироблявся на заводі”Сигнал”(м. Обнінськ, Калузької обл.) Із загальним обсягом випуску до 100 тис.шт. в рік. Сповіщувач РІД-6М має обмежений призначений термін служби альфа-джерел типу АІП-РІД - 10 років з моменту їх випуску. Існує технологія установки нових альфа-джерел типу АІП-РІД в пожежних сповіщувачах минулих років випуску, що дозволяє продовжувати експлуатацію сповіщувачів ще 10 років, замість їх вимушеного демонтажу та захоронення.

Висока чутливість дозволяє використовувати радіоізотопні сповіщувачі як складовий компонент аспіраційних сповіщувачів. При прокачуванні через сповіщувач повітря захищуваних приміщень він може забезпечувати подачу сигналу при появі навіть мізерної кількості диму - від 0,1 мг / м. При цьому довжина трубок для забору повітря практично не обмежується. Наприклад, практично завжди реєструє факт займання сірникової голівки на вході повітрозабірних трубки довжиною 100 м.

#### Електроіндукціонний

Принцип роботи сповіщувача: аерозольні частинки засмоктуються з навколишнього середовища в циліндричну трубку (газохід) за допомогою малогабаритного електричного насоса і потрапляють в зарядну камеру. Тут, під впливом уніполярного коронного розряду, частки набувають об'ємний електричний заряд і, рухаючись далі по газоходу, потрапляють у вимірювальну камеру, де наводять на її вимірювальному електроді електричний сигнал, пропорційний об'ємному заряду часток і, отже, їх концентрації. Сигнал з вимірювальної камери потрапляє в попередній підсилювач і далі в блок обробки і порівняння сигналу. Датчик здійснює селекцію сигналу по швидкості, амплітуді і тривалості та видає інформацію при перевищенні заданих порогів у вигляді замикання контактного реле.

Електроіндукційні сенсори використовуються в системах пожежної сигналізації модулів”Зоря”та”Пірс”МКС.

### Ручні сповіщувачі

Пожежний ручний сповіщувач - пристрій, призначений для ручного вмикання сигналу пожежної тривоги в системах пожежної сигналізації та пожежогасіння. Ручні пожежні сповіщувачі слід встановлювати на висоті 1,5 м від рівня землі або підлоги.

### Ультразвукові сповіщувачі

При виникненні пожежі виникають турбулентні потоки повітряного середовища. Під їх впливом ультразвукове поле, яке заповнює приміщення, змінюється. Відбувається зміна енергії ультразвуку через поглинання тепловим потоком і віддзеркалення від межі розділу турбулентних потоків. Кордон конвективної струменя над осередком пожежі нестійка, тому виникає амплітудно-фазова модуляція ультразвукового поля. Ультразвукові сповіщувачі поєднували охоронні та пожежні функції, могли працювати тільки в закритому приміщенні при відсутності рухаються предметів.

Ультразвукове поле на об'єкті (в конструкції сповіщувачів 1979 року) створюється електроакустичним магнітострікцііонним перетворювачем, який являє собою механічну коливальну систему. Такий перетворювач звернемо і застосовується в якості приймача і випромінювача ультразвукових коливань.

# Існуючі методи

Існують декілька методів для знаходження диму в видимому спектрі. Дані методи описані за допомогою таких візуальних ознак як рух, границі, потемніння та геометрії контуру диму. Далі вони використовували Байєсівський аналіз, або іншій, щоб вирішити чи був знайдений дим. Основні методи описані нижче:

* Фуджівара та Терада [1] запропонували використати фрактали для виділення областей з димом. Далі вони використали властивість однорідності диму для класифікації виділених областей.
* Копіловіч [2] використав властивість недетермінованості руху диму викликаним нежорсткістю диму. Він обрахував поле оптичного потоку, використавши два сусідні зображення, а потім використав ентропію розподілу напряму руху для виявлення диму.
* Турейн [3] виділив такі характеристики зображення як рух, мерехтіння, розмиття границь для виявлення диму.
* Грек-Чіні [4] використали більш ніж 20 характеристик зображення, такі як процент зміни зображення, кореляцію, дисперсію та використав Байєсівський аналіз щоб знайти дим.

## Копіловіч

Системи спостереження, які використовують панорамні камери мають багато переваг відносно традиційних систем. Оскільки вони мають 360 градусів кут огляду, воки спроможні спостерігати всю панораму навколо камери та відслідковувати будь-яку кількість об’єктів одночасно. Такі системи можуть замінити три або більше традиційних рухливих камер. Більшого при переміщенні традиційної камери потрібно кожен раз розраховувати фон та спотворення перспективи. Системи, які використовують статичні панорамні камери не мають таких проблем. Хочу панорамні камери мають специфічне оптичне спотворення, яке потрібно враховувати. Також кільцеві зображення повинні бути перетворені на прямокутні до їх обробки. Звичайні системи з традиційною оптикою проектують лише невелику частину панорами на їх матрицю. Панорамні камери проектують всю панораму на матрицю. Тому роздільні здатність традиційних камер меньша ніж на панорамних.

Першим етапом в виявленні диму є виділення фону. В даному методі використовується статичні методи для виділення фону. Для цього з кожного кадру виділяється статичні частини, які об’єднуються в фон. Після цього, передній план може бути знайденим простим відніманням фону від кадру.

Для кожного об’єкта знайденого на зображення формується історія його переміщень. Історія переміщень представляє собою зважені усереднені за часом знайдені регіони. За допомогою цієї історії тренується нейромережа, яка може розпізнавати деякі шаблони рухів.

Для знаходження диму використовується властивість диму в його нерівномірності поля швидкостей. Розраховується статичне поле швидкостей. З розраховано поля виділяють дві характеристики:

* Самонеподібність: рух диму прагне бути неподібним до самого себе. При великих масштабах рух диму регулярний, але при малих - ні.
* Нерегулярність руху в наслідок нежорсткості диму.

Дані припущення були використані для аналізу поля швидкості диму. Для врахування самонеподібності багатомасштабне оптичне поле було розраховане за допомогою викривлення швидкості.

Загалом алгоритм має наступний вигляд:

* Розрахунок поля оптичного потоку.
* Тест на самоподібність.
* Тест на регулярність
* Байєсівський вивід для тривоги.

Недоліком цього методу є погана швидкодія та дороговизна панорамних камер.

## Турейн

Даний метод складається з 5 кроків:

* Виявлення руху пікселів на відео.
* Перевірка гладкості границь на передньому плані зображення за допомогою просторового вейвлет-перетворення.
* Якщо границі зглажені перевіряються U та V канали.
* Знаходження мерехтіння
* Форма регіону перевіряється на випуклість.

Фон розраховується методом, запропонованим Колінсом. Після виділення фону необхідно аналізувати регіони, які переміщуються, щоб визначити чи це дим. Дим заслоняє текстуру та границі на зображенні. Оскільки текстура та границі відносяться до високочастотної інформації енергії вейвлетів зменьшуються в наслідок заслону їх димом. За допомогою цього можна відслідковувати вейвлет енергію для усіх регіонів, що переміщуються.

Відомо, що вейвлети регіонів містять в собі інформацію про границі вихідного зображення. Границі породжують локальні екстремуми в вейвлетах регіонів. Якщо дим покриває одну з границь на зображенні, тоді дана границя стає меньш видимою та після деякого часу може зовсім зникнути, якщо дим буде ставати густішим. Нехай вейвлет коефіцієнт HL відповідає границі, покритій димом. Його значення зменьшується в наслідок зменьшення його видимості. На деяких кадрах його значення може бути нульовим, або бути близьким до нуля. Тому позиція границь на оригінальному зображенні визначається з екстремумів вейвлет перетворення фонового зображення. Повільне зменшення екстремуму вейвлета є ключовим для виявлення диму. Якщо значення групи коефіцієнтів вейвлету зменьшується плавно,це говорить про те, що на відео скоріше за все є дим. Якщо ж коефіцієнти різко зменьшилися, це говорить про те, що на відео з’явився звичайний об’єкт.

Ще одним етапом при занходження диму є використання інформації про колір. Спочатку, коли дим починає розповсюджуватися він прозорий, тому він зберігає напрям RGB вектора фонового зображення. Це ще один ключовий момент для знаходження диму. Але лице цієї інформації недостатньо тому, що тіні породжують такий самий ефект. Коли дим стає густішим подібність кадру та фонового зображення зменьшується та яскравість U та V каналів стає меншою. Саме такі регіоні в подальшому класифікуються.

# Запропонований метод

Запропонований метод оснований на виділенні фону, мерехтіння, знаходження контуру та його класифікація, заснована на евристіному та емпіричному знанні про дим. Далі описується метод більш детально.

## **Виділення ф**о**ну**

Для виділення фону був використаний метод запропонований Стафером та Грімсоном [5]. Для цьго використовується суміш Гаусіанів для виділення фону. Така модель була використана тому, що на практиці фон та освітлення можуть змінюватись. Тому, як тільки якийсь з цих параметрів змінюється генерується новий Гаусіан, щоб зробити припущення про новий фон. Всі інші пікселі, які не були класифіковані Гаусіаном як фон групуються для подальшого аналізу.

Така модель краща ніж модель з одним Гаусіаном на піксель. Для того, щоб віслідковувати піксель з конкретної поверхні під конкретним освітлення достатньо було б одного Гаусіана. Якщо б освітлення змінювалося, то було б достатньо одного адаптивного Гаусіана. Але на практиці піксель може відноситься до багатьох поверхонь та під багатьма освітленнями. Тому необхідно ввести суміш адаптивних Гаусіанів.

Дивлячись на статичність та дисперсію Гаусіана він відноситься до фону. Якщо піксель не належить жодному з Гаусіанів, то він відноситься до переднього плану, до того часу поки не знайдеться Гаусіан, який його включить.

Така система дозволяє адаптувати фон до зміни освітлення, повторних рухів на сцені, повільних об’єктів, внесення або винесення об’єктів зі сцени. Повільні об’єкти довше стають фоновими, оскільки вони мають більшу дисперсію кольорів ніж фон. Також повторні рухи зап’ятовуються і відносяться до фону.

Такий метод має два конфігуровані параметри – **а**, константа навчання та **Т**, частина даних, яка повинна бути фоном. На пркатиці система, вці параметри не змінювалися ефективно працює вже 16 місяців.

Кожен раз, коли корегується якийсь з параметрів Гаусіана, використовується евристичний метод для того, щоб зрозуміти, чи відноситься даний Гаусіан до фону. Пікселі, які не відносяться до жодного з Гаусіанів фону класифікуються як пікселі переднього плану. Такі пікселі поєднуються в зв’язані групи, які відслідковуються на екрані.

## Виділення мерехтіння

Піксель на границі полум’я може з’являтися та зникати багато разів на секунду. Така періодичність відома як мерехтіння. Частота мерехтіння полум’я була знайдена експериментально і дорівнює приблизно 10Гц. Але мерехтіння диму є більш нижчим: 2-3Гц. Така частота може бути вирахувана за допомогою перетворення Фур’є, Вейвлет перетворення, тощо.

## Знаходження контуру

Експериментально доведено, що маска мерехтіння диму достатньо розрідженна. Тому взявши контур можна порахувати кількість пікслеів що мерехтять. Цю кількість можна порівняти з деяким порогом та залишити тільки ті контури, які задовільняють.

## Класифікація контуру

З контурів з попереднього кроку виділяються характеристики та передаються в блок класифікації для подальшої обробки. Даний класифікатор оснований на роботі Катракіса [6] , в якій він описав турбулентні явища.

Вогонь та дим являються турбулентними явищами. Складність форми турбулентного явища можна охарактеризувати величиною відношення периметру до площі або площі поверхні до об’єму. Тому регіони з димом можна класифікувати вирахувавши площу та периметр регіону. Для двовимірного випадку турбулентність вираховується як:

2=P2A.

Де P - периметр регіону, А - площа регіону. Дана величина була нормалізована, для того щоб величина турбулентності кола дорівнювала 1. Зі збільшення складності фігури турбулентність росте.

Для трьохвимірного випадку турбулентність форми буде дорівнювати

3=S62/31/3V2/3.

Де S - це площа поверхні фігури, V - об’єм фігури. Ця величина також нормалізована для турбулентності сфери.

Для відеоряду може використовуватися як двовимірна турбулентність так і приближена трьохвимірна. Але на практиці краще використовувати степеневу залежність між периметром та коренем квадратним площі:

P=c(A)q

Де P - периметр регіону, А - площа регіону, с - константа, q - деяка змінна. Експериментально доведено, що залежність периметру від площі регіону диму характеризується q=1,35.

# Розробка ПЗ

## Мова програмування

Для реалізації даного метода була обрана мова програмувааня Clojure. Clojure - Lisp'подібна мова загального призначення, розроблена для Java Virtual Machine (JVM). Автором мови є Річ Хікі, який кілька років розробляв мову поодинці аж до випуску першої публічної версії в 2007 -му році. В даний час, стабільною версією є версія 1.5, випущена в грудні 2013-го року.

На відміну від інших реалізацій Lisp'а і Scheme для віртуальної машини Java, таких як ABCL, Kawa і т.д., Clojure не сумісний на 100 відсотків ні з Common Lisp, ні з Scheme, але запозичила багато ідей з цих мов, додавши нові речі, такі як незмінюваність даних, конкурентне виконання коду і т.п. Більш докладно про те, навіщо було створено новий мову, можна прочитати на сайті проекту.

Незважаючи на те, що Clojure - молодий мова програмування, досить багато людей використовують його в своїх проектах, в тому числі і комерційних, наприклад, FlightCaster, який використовує Clojure при обробці великої кількості даних, вирішуючи завдання Machine Learning в розподіленої середовищі. Існують і інші фірми (наприклад, Sonian, Runa, Emendio), які використовують цю мову в своїй роботі - посилання на них ви зможете знайти на сайті мови.

### Основні можливості мови

Clojure є функціональною мовою програмування з підтримкою функцій в якості об'єктів першого класу (first class objects) і незмінними (за винятком спеціальних випадків) даними, включаючи підтримку”лінивих”колекцій даних. Від Lisp'а Clojure”успадкував”макроси, мультиметоди та інтерактивний стиль розробки, а JVM дає переносимість та доступ до великого набору бібліотек, створених для цієї платформи.

Незмінність структур даних дозволяє використовувати їх у різних потоках виконання програми, що спрощує багатопоточне програмування. Однак не всі структури є незмінними - у потрібних випадках програміст може явно використовувати змінювані структури даних, використовуючи Software Transactional Memory (STM), що забезпечує надійну роботу в багатопотокової середовищі. (Як приклад багатопотокової програми, що працює з розділяються даними, можна привести програму”мурахи”(ants), яку досить складно написати на Java -за великої кількості модельованих сутностей, але яка досить просто виглядає на Clojure).

За рахунок того, що Clojure був спроектований для роботи на базі JVM, забезпечується доступ до великого набору бібліотек, існуючих для даної платформи. Взаємодія з Java реалізується в обидві сторони - як виклик коду, написаного на Java, так і реалізація класів, які доступні як для виклику з Java, так і з інших мов, що існують для JVM, наприклад, Scala. Детальніше про взаємодію з JVM написано далі.

### Відмінності від Lisp

Незважаючи на схожість синтаксису, Clojure відрізняється і від Common Lisp, і від Scheme. Деякі відмінності обумовлені тим, що мова розроблена для платформи JVM, що накладає деякі обмеження на реалізацію. Наприклад, JVM не підтримує оптимізацію хвостових викликів (tail call optimization, TCO), тому в мову були введені явні оператори loop і recur. Також важливими визначальними факторами JVM - платформи є:

* boxed integers - немає підтримки повного набору типів чисел (numeric tower), які є в Scheme і Common Lisp;
* система виключень як в Java (в Common Lisp використовується сигнальний протокол);
* використовується виклики про Java методів.

Повний список відмінностей можна знайти на окремій сторінці на сайті мови. З явних відмінностей від Common Lisp можна відзначити наступні:

* ідентифікатори в Clojure залежні від регістру;
* велика частина даних - незмінна;
* користувач не може змінювати синтаксис мови шляхом введення власних макросів у процедурі читання коду (read macros);
* введений спеціальний синтаксис для літералів, векторів, словників, регулярних виразів, анонімних функцій і т.д.;
* існує можливість зв'язування метаданих з змінними та функціями;
* можна реалізувати функції з одним іменем і різним набором аргументів;
* звичні речі, такі як let, по синтаксису відрізняються від їх аналогів в Common Lisp і Scheme (при цьому використовується менше дужок), наприклад, let пов'язує дані послідовно, аналогічно let \* в Scheme;
* замість функцій car і cdr використовуються функції first і rest;
* nil НЕ дорівнює пустому списку або іншому набору даних (колекції) - він всього лише означає відсутнє значення (аналог null в Java);
* використовується загальний простір імен, як в Scheme;
* порівняння на рівність проводиться однією функцією на відміну від Common Lisp і Scheme;
* підтримка”ледачих”колекцій.

### Джерела інформації про мову

Основне джерело інформації по даній мові - сайт проекту і список розсилки. Крім сайту проекту, хорошим джерелом інформації є набір відеолекцій на Blip.TV, а також відеолекції, в яких автор мови розповідає про Clojure і про особливості його використання. Крім того, слід відзначити набір скрінкастів, створених Sean Devlin, в яких він розповідає про різні можливості мови, включаючи нові.

З книг в даний час доступно декілька книг Programming Clojure, випущена в серії Pragmatic Programmers, яка в принципі містить всю необхідну інформацію про мову, включаючи опис основних можливостей мови, питання взаємодії з Java, основні функції, відмінність мови від Common Lisp, і т.п. Видавництво Apress випустило ще одну книгу по Clojure - Practical Clojure. The Definitive Guide, яка є коротким описом сучасної версії мови, включаючи нововведення. У видавництві Manning надруковано книгу Clojure in Action (введення в мову і приклади практичного використання) і The Joy of Clojure. Thinking the Clojure Way (більш”глибоке”опис мови, з роз'ясненням складних понять).

У вільному доступі можна знайти книгу Clojure Programming, робота над якою ведеться в рамках проекту WikiBooks. Також існує досить докладний практичний підручник - Clojure Scripting. Крім того, нещодавно був опублікований підручник Clojure Notes, який використовувався в рамках курсу навчання Clojure.

Хороший опис того, як можна використовувати макроси для побудови абстракцій, можна знайти у відомій книзі On Lisp Пола Грема (Paul Graham). Незважаючи на те, що в ній використовується Common Lisp, багато речей будуть застосовні і для Clojure.

Дуже велика кількість інформації про мову, що розробляються бібліотеках та проектах, що використовують Clojure, публікується в блогах. Для того, щоб звести всю цю інформацію воєдино, існує проект Planet Clojure, на який ви можете підписатися, щоб бути в курсі новин про мову.

### Установка і запуск

Установка Clojure досить проста - скачайте останню версію з сайту мови і розпакуйте в потрібний каталог. Після цього ви можете запустити її за допомогою команди:

java - cp clojure.jar clojure.main

Ця команда призведе до запуску JVM і ви отримаєте доступ до REPL ("read - eval - print loop”- цикл введення виразів та видачі результатів). Стандартний REPL має не дуже хороші можливості по редагуванню коду, так що при роботі з REPL краще використовувати бібліотеку jline, як описано в розділі Getting Started офіційній документації Clojure, або скористатися однією з середовищ розробки, описаних в розділі Середовища розробки. Більш докладні інструкції з розгортання для різних середовищ розробки ви можете знайти в описі проекту labrepl, метою якого є спрощення початку роботи з Clojure. У складі даного проекту є набір навчальних матеріалів, які будуть корисні починаючим працювати з мовою.

Працюючи в REPL ви можете отримувати інформацію про функції, макроси та інших об'єктах мови. Для отримання інформації про будь-який символ або спеціальну форму ви можете використовувати макрос doc. Наприклад, (doc map) надрукує довідку по функції map, яка була задана при оголошенні цієї функції. А якщо ви не пам'ятаєте точну назву символу, можна провести пошук по документації за допомогою функції find-doc, яка приймає один аргумент - рядок з регулярним виразом за яким буде проводитися пошук.

### З чого складається мова Clojure

Синтаксис мови Clojure слідує стандартному для Lisp'образних мов підходу”код як дані", коли дані і код мають загальний синтаксис. Як і в інших діалектах Lisp'а, код записується у вигляді списків, використовуючи префіксну нотацію і представляючи собою абстрактне синтаксичне дерево. Однак порівняно з іншими мовами, в Clojure введені додаткові типи даних: крім стандартних для Lisp'а символів, базових літералів (рядки, числа і т.п.) і списків,в мову введено додатковий синтаксис для векторів, словників (maps) і множин (sets), що є об'єктами першого класу (first class objects).

Крім цього, процедура читання коду (reader) розпізнає специфічні для Clojure конструкції: @ - для доступу до змінюваних даними і різні конструкції, що починаються з символу # - анонімні функції, метадані (включаючи інформацію про типи даних), регулярні вирази і т.д. Процедура читання також розглядає пробіли і коми між елементами мови як один символ, що розділяє ці елементи.

### Основні типи даних

Дані в Clojure можна розділити на дві великі групи: базові типи даних - числа, рядки і т.д., і послідовності (колекції), до яких відносяться списки, вектори, відображення та множини. Користувач може визначати свої структури даних за допомогою defstruct, але вони є окремим випадком словників і введені для забезпечення більш ефективної роботи зі складними даними.

Всі типи даних мають загальний набір характеристик: дані незмінні і реалізують операцію”рівність”(equality).

#### Базові типи даних

До базових типів даних Clojure належать такі:

* Логічні значення.  
  В мові визначено два об'єкти для представлення логічних значень: true - для істинного значення і false - для помилкового. (Всі інші значення, крім false і nil, розглядаються як істинні);
* Числа.  
  В мові можуть використовуватися числа різних типів. За замовчуванням для представлення цілих чисел використовуються класи, успадковані від java.lang.Number - Integer, BigInteger, BigDecimal, але в Clojure реалізується спеціальний підхід, який дозволяє представляти число найбільш ефективним способом, автоматично перетворюючи числа у разі потреби - наприклад, при переповненні числа. Якщо ви хочете для цілого числа явно вказати тип BigDecimal, то ви можете додати букву M після значення.   
  Для чисел з плаваючою точкою використовується стандартний клас Double.  
  Крім цих видів чисел, в Clojure визначений спеціальний тип Ratio, що представляє числа у вигляді раціональних дробів, що дозволяє уникати помилок округлення - наприклад, при діленні.
* Рядки.  
  Рядка в Clojure є екземплярами класу java.lang.String і до них можна застосовувати різні функції визначені в цьому класі. Форма запису рядків Clojure збігається зі стандартним записом рядків в Java;
* Знаки (characters) є екземплярами класу java.lang.Character і записуються або у формі \N, де N - відповідна літера, або як назви для нечитаючих букв - наприклад, як \tab і \space для символу табуляції і пробілу і т.д.;
* Символи (symbols) використовуються для посилання на щось - параметри функцій, імена класів, глобальні змінні і т.д. Для представлення символу як окремого об'єкта, а не як значення, для якого він використовується в якості імені, використовується стандартна запис 'symbol (або спеціальна форма quote);
* Keywords (ключові слова) – це спеціальні символи, що мають значення самих себе, аналогічно символам (symbols) у Lisp і Ruby. Одним з важливих їх властивостей є дуже швидка операція перевірки на рівність, оскільки відбувається перевірка на рівність покажчиків. Ця властивість робить їх дуже зручними для використання в якості ключів в відображеннях (maps) і тому подібних речах. Для іменованих аргументів існує спеціальна форма запису :keyword.

Варто також відзначити, що символи і keywords мають деяку спільність - в рамках інтерфейсу IFn для них створюється функція invoke () з одним аргументом, що дозволяє використовувати символи і keywords в якості функції. Наприклад, конструкція (:mykey my-hash-map) або ('mysym my-hash-map) аналогічні викликам (get my-hash-map :mykey) або (get my-hash-map 'mysym), який приведе до отримання значення з потрібним ключем з відповідного словника.

У мові Clojure є спеціальне значення nil, яке може використовуватися як значення будь-якого типу даних, і збігається з null в Java. nil може використовуватися в умовних конструкціях нарівні зі значенням false. Проте варто відзначити, що, на відміну від Lisp, nil і порожній список - () не є взаємозамінними і використання порожнього списку в умовній конструкції буде розглядатися як значення true.

#### Колекції, послідовності і масиви

Крім загальних характеристик базових типів перерахованих вище, всі колекції в Clojure мають такі характеристики:

* вся робота з колекціями проводиться через загальний інтерфейс;
* існує можливість зв'язування метаданих з колекцією;
* для колекцій реалізуються інтерфейси java.lang.Iterable і java.util.Collection, що дозволяє працювати з ними з Java;
* всі колекції розглядаються як ”послідовності” даних, незалежно від конкретної реалізації.

Незмінюваність колекцій означає, що результатом роботи всіх операцій по модифікації колекцій є інша, нова колекція, в той час як вихідна колекція залишається незмінною. У Clojure існує ефективний механізм, що допомагає реалізовувати незмінні колекції. З його допомогою операції, які змінюють колекцію, можуть ефективно створювати ”змінену” версію даних, яка використовує велику частину вихідних даних, не створюючи повної копії.

У поточній версії Clojure реалізовані такі основні види колекцій:

* списки (lists) записуються точно також як і в інших реалізаціях Lisp. У Clojure списки безпосередньо реалізують інтерфейс ISeq, що дозволяє функціям роботи з послідовностями ефективно працювати з ними. (При використанні функції conj нові елементи списків додаються до початку);
* вектори (vectors) являють собою послідовності, елементи яких індексуються цілим числом (з послідовними значеннями індексу в діапазоні 0.. N, де N - розмір вектора). Для визначення вектора необхідно укласти його елементи в квадратні дужки, наприклад, [1 2 3]. Для перетворення інших колекцій в вектор можна використовувати функції vector або vec. Оскільки вектор індексується цілим числом, то операція доступу до довільного елементу реалізується досить ефективно, що зручно при роботі з деякими видами даних. (При використанні функції conj нові елементи векторів додаються в кінець). Крім того, для вектора в Clojure створюється функція одного аргументу (цілого числа - індексу значення) з ім'ям, що збігається з ім'ям символу, пов'язаним з вектором. Це дозволяє використовувати ім'я вектора в якості функції для доступу до потрібного значення. Наприклад, виклик (v 3) в даному коді:  
    
  user > (def v [1 2 3 4 5”string”])  
  user > (v 3) ; => 4  
  поверне значення четвертого елемента вектора.
* словники (maps) – це спеціальний вид послідовності, який відображає одні значення даних (ключ) в інші (значення). У Clojure існують два види словників: hash-map і sorted-map, які створюються за допомогою відповідних функцій. hash-map забезпечує більш швидкий доступ до даних, а sorted-map зберігає дані в відсортованому по ключу вигляді. Відображення записуються у вигляді набору значень (з парною кількістю елементів), укладених у фігурні дужки. Значення, що стоять на непарних позиціях розглядаються як ключі, а на парних - як значення, пов'язані з даним ключем. Як ключ можуть використовуватися будь підтримувані Clojure типи даних, але дуже часто в якості ключів використовують keywords, оскільки для них реалізована дуже швидка перевірка на рівність.   
  Також як і для векторів, для відображень створюється функція одного аргументу (ключа), яка дозволяє використовувати ім'я символу, пов'язаного з відображенням, для доступу до елементів. Наприклад,   
    
  user > (def m { 1 1: abc 33: 2”2”})  
  user > (m :abc) ; => 33
* множини (sets) являють собою набір унікальних значень. Також як і для відображень, існує два види множин - hash-set і sorted-set. Визначення множини має такий вигляд #{ elements...}, а для створення множин з інших колекцій може використовуватися функція set, наприклад, для отримання множини унікальних значень вектора, можна використовувати наступний код:   
    
  user > (set [ 1 2 3 2 1 2 3 ])

У Clojure також визначені додаткові види словників, що дозволяють в спеціальних випадках домогтися більшої продуктивності:

* словники-структури (struct maps) можуть використовуватися для емуляції записів (records), наявних в інших мовах програмування. У цьому випадку словники мають набір однакових ключів і Clojure реалізує ефективне зберігання інформації про ключі, а також надає швидкий доступ до елементів по ключу. У разі необхідності, є можливість генерації спеціалізованої функції доступу за допомогою функції accessor.   
  Визначення словника-структури проводиться за допомогою макросу defstruct або функції create-struct. Нові екземпляри словників створюються за допомогою функції struct-map або struct, які отримують список елементів для заповнення даного відображення. При цьому варто відзначити, що словник-структура може мати більшу кількість ключів, ніж було визначено в defstruct - в цьому відношенні, відображення - структури поводяться точно також, як і звичайні відображення.
* відображення - масиви (array maps) це спеціальний вид відображень, в якому зберігається порядок ключів. Такі відображення реалізовані у вигляді звичайного масиву, що містить ключі і значення.

Велика частина функцій для роботи з послідовностями є "лінивою", обробляючи дані по мірі їх потреби, що дозволяє ефективно працювати з даними великого розміру, у тому числі і з нескінченними послідовностями. Користувач може створювати свої функції, які повертають "ліниві" послідовності, за допомогою макросу lazy-seq. Також у версії 1.1 було введено поняття блокових послідовностей ( chunked sequence ), які дозволяють створювати елементи блоками по N елементів, що в деяких випадках дозволяє поліпшити продуктивність.

Із загального ряду випадає робота з масивами Java, оскільки вони не є колекціями в термінах Clojure. Для роботи з масивами визначений набір функцій, які дозволяють визначати масиви різних типів (make - array, XXX- array, де XXX - назва типу), отримання ( aget ) і установки ( aset ) значень в масиві, перетворення колекцій в масив ( into - array ) і т.д.

## Бібліотека комп’ютерного бачення

OpenCV - де-факто найпопулярніша бібліотека комп'ютерного зору.

Вона написана на C/C++, її вихідний код відкритий. Бібліотека включає

більше 1000 функцій і алгоритмів. Вона розробляється з 1998 р., Спочатку

в компанії Інтел, тепер в Itseez за активної участі спільноти. Про

високу популярність бібліотеки свідчить кількість завантажень,

їх більш 6000000 завантажень (без урахування SVN / Git трафіку).

Існують бібліотеки, більш просунуті по функціональності,

наприклад, Halcon. Є бібліотеки більш спеціалізовані, що роблять

акцент на будь-якої конкретної задачі, наприклад, libmv. OpenCV -

найбільша бібліотека за широтою тематики.

Бібліотека розповсюджується за ліцензією BSD, що означає, що її

можна вільно і безкоштовно використовувати як у відкритих проектах з

відкритим кодом, так і в закритих, комерційних проектах. Бібліотеку не

обов'язково копіювати цілком у свій проект, можна використовувати шматки коду. Єдина вимога ліцензії - наявність в супроводжуючих

матеріалах копії ліцензії OpenCV.

Через ліберальну ліцензію бібліотека використовується багатьма

компаніями, організаціями, університетами, наприклад, NVidia, Віллоу

Гараж, Intel, Google, Стенфордський університет. Компанії NVidia і WillowGarage частково спонсорують її розробку.

Є підстава думати, що не всі компанії, що використовують OpenCV,

афішують це. Наведемо невеликий перелік відомих авторам

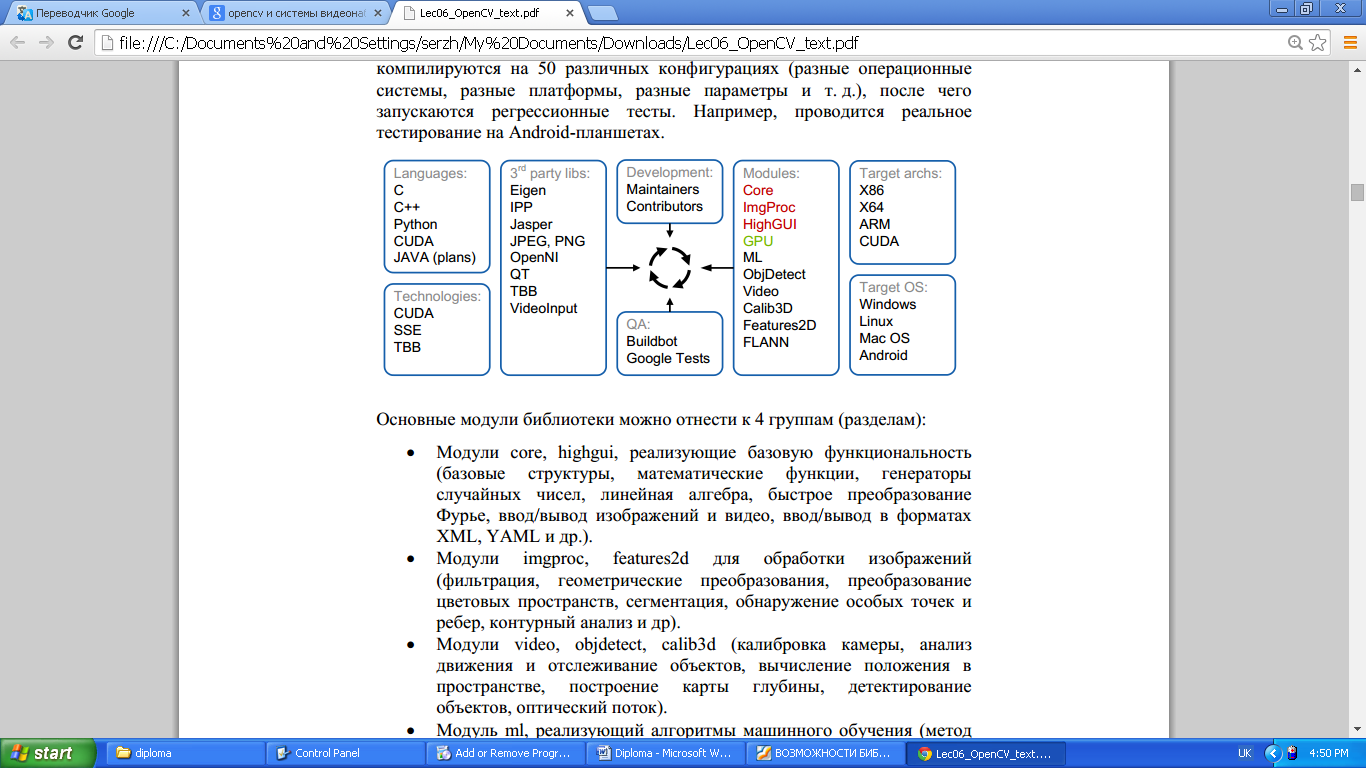
проектів, в яких використовується OpenCV :

* Система зору робота PR2, розробленого компанією WillowGarage (www.willowgarage.com). Проект PR2 націлений на вирішення складної задачі - створення платформи для персональної робототехніки.
* Аудіо - візуальна інсталяція в Музеї Сучасного Мистецтва ( Сан- Франциско).
* Контроль якості монет, що виготовляються Центробанком Китаю.
* Курси комп'ютерного зору в Стенфорді.
* Панорами вулиць в картах Google.

Багатовимірна архітектура проекту представлена ​​на рис. бібліотека

складається з 16 модулів. Реалізовані близько 1000 алгоритмів. Функціональність доступна на різних мовах: C, C++, Python, CUDA, Java. Підтримуються основні операційні системи: MS Windows, Linux, Mac, Android, IOS. Є можливість використання сторонніх бібліотек, наприклад, для роботи з пристроєм Kinect ( OpenNI ), розробки паралельних програм ( ТВВ ) та ін.

Проект використовує систему безперервної інтеграції. Щоночі останні робочі версії OpenCV викачуються з репозиторію і компілюються на 50 різних конфігураціях ( різні операційні системи, різні платформи, різні параметри і т. д.), після чого запускаються регресійні тести. Наприклад, проводиться реальне тестування на Android- планшетах.



Основні модулі бібліотеки можна віднести до 4 групам (розділів ) :

* Модулі core, highgui реалізують базову функціональність ( Базові структури, математичні функції, генератори випадкових чисел, лінійна алгебра, швидке перетворення Фур'є, введення / виведення зображень і відео, введення / виведення у форматах XML, YAML та ін.)
* Модулі imgproc, features2d для обробки зображень ( Фільтрація, геометричні перетворення, перетворення колірних просторів, сегментація, виявлення особливих точок і ребер, контурний аналіз та ін).
* Модулі video, objdetect, calib3d ( калібрування камери, аналіз руху і відстеження об'єктів, обчислення положення в просторі, побудова карти глибини, детектування об'єктів, оптичний потік ).
* Модуль ml, який реалізує алгоритми машинного навчання (метод найближчих сусідів, наївний байесовский класифікатор, дерева рішень, бустінг, градієнтний бустінг дерев рішень, випадковий ліс, машина опорних векторів, нейронні мережі та ін.)

Звернемо увагу на підтримку роботи з XML -файлами. Результати проміжних обчислень можна зберігати в XML -файлах, а потім їх прочитати, наприклад, в іншій програмі, що полегшує розробку алгоритму по частинах, роботу над алгоритмом цілою командою.

Зауважимо, що бібліотека OpenCV реалізує, як правило, тільки базові

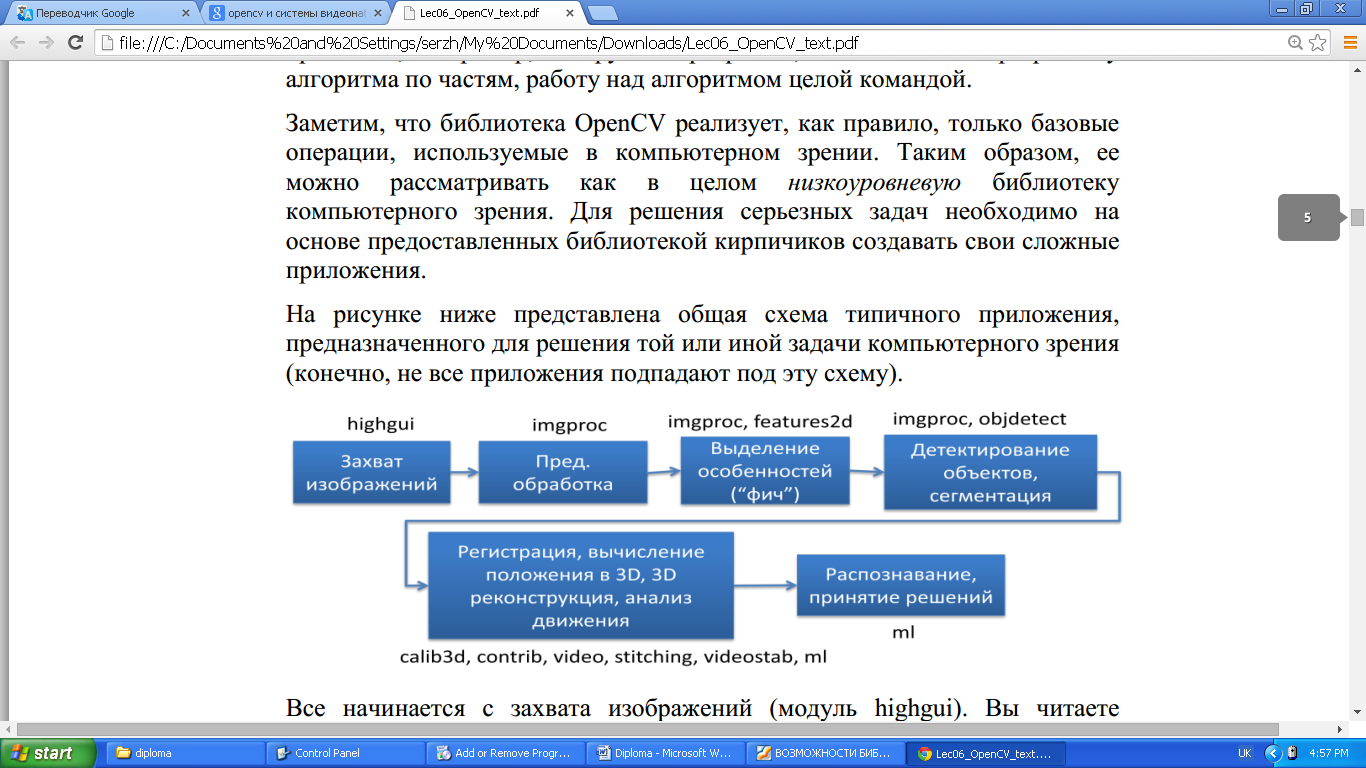
операції, використовувані в комп'ютерному зорі. Таким чином, її

можна розглядати як у цілому низкорівневу бібліотеку комп'ютерного зору. Для вирішення серйозних завдань необхідно на основі наданих бібліотекою цеглинок створювати свої складні додатки.

На малюнку нижче представлена ​​загальна схема типового додатки,

призначеного для вирішення тієї чи іншої задачі комп'ютерного зору

( Звичайно, не всі програми підпадають під цю схему).



Все починається з захоплення зображень (модуль highgui ). Ви читаєте зображення з файлу або читаєте відео з мережевої камери через мережевий

протокол.

Далі здійснюється попередня обробка (модуль imgproc ), така, як усунення шуму, вирівнювання яскравості, контрасту, виділення і видалення відблисків, тіней. Наприклад, один і той же об'єкт при різному освітленні виглядає по- різному. У яскравому світлі червона машина, рух якої, необхідно відстежувати, буде яскраво- помаранчевою. У похмуру погоду та ж машина виглядатиме червоно- рожевої. У цьому випадку на зображенні необхідно виконати вирівнювання кольорів. Передобробка може бути простою, але може містити в собі цілу складну технологію.

Наступний етап - виділення особливостей (модулі imgproc, features2d ). Наприклад, в задачі стеження за об'єктом це може бути пошук спеціальних точок на об'єкті, за якими легко спостерігати ; для задачі детектування особи (т. е виявлення на зображенні. ) – обчислення опису кожного пікселя.

Далі відбувається детектування цікавлять нас об'єктів, виділення значущих частин, сегментація зображення ( модулі imgproc, objdetect ). Якщо, наприклад, камера нерухома, а зображення рухливе, можна використовувати алгоритми віднімання фону.

Після цього ви вирішуєте основну задачу, таку, як обчислення

розташування об'єкта в 3D, реконструкцію 3d структури, аналіз

структури, реєстрацію і т. п. ( Модулі calib3d, contrib, video, stitching, videostab, ml). Наприклад в задачі склейки панорам зображень – це зіставлення частин різних кадрів, визначення потрібного перетворення. У задачі відеоспостереження це відновлення траєкторій об'єктів і т.д.

Наприкінці відбувається розпізнавання і прийняття конкретних рішень

( Модуль мл). Наприклад, в системі відеоспостереження : з'явився

небажаний об'єкт в кадрі чи ні. У задачі детектування тексту -

детектувати текст, що саме за текст і т. д.

# Експериментальні результати

Для перевірки роботи методу були використанні тестові відеозаписи з димом. На всіх тестових відеозаписах дим був знайдений правильно. Результати тестової роботи можна переглянути нижче.

<img src='result/result1.png' />

<img src='result/result2.png' />

<img src='result/result3.png' />

<img src='result/result4.png' />

Рис.1. Робота детектора з відеозаписами з димом.

Також на відеозаписах без диму декткор нічого не знайшов

<img src='result/result5.png' />

<img src='result/result6.png' />

<img src='result/result7.png' />

<img src='result/result8.png' />

Рис.2. Результат роботи детектора з відеозаписами без диму.

# Подальша робота над методом

Для покращення роботи метода, необхідно подолати наступні проблеми:

1. Задання паттерну для диму дуже важка проблема. Це викликано тим, що дим в різних середовищах веде себе по різному. Так, наприклад, при сильному вітрі характеристики диму дуже сильно міняються від характеристик диму в безвітрянному середовищі.

2. Властивості зображень диму дуже важко адаптувати до різного освітлення, що робить такий метод дуже вразливим до помилкових виявлень.

3. Підвищення продуктивності методи за рахунок обробки зображень не тільки в видимому спектрі, а і в інфра-червоному спектрі.

# Використана література

1. N. Fujiwara and K. Terada,”Extraction of a smoke region using fractal coding”, IEEE International Symposium on Communications and Information Technology, 2004, ISCIT 2004, Volume 2, 26-29 Oct. 2004.

2. I. Kopilovic, B. Vagvolgyi, and T. Sziranyi,”Application of panoramic annular lens for motion analysis tasks: surveillance and smoke detection”, Proceedings of 15th International Conference on Pattern Recognition, 2000, Volume 4, 3-7 Sept. 2000.

3. B. U. Toreyin, Y. Dedeoglu, and A. E. Cetin,”Wavelet based real-time smoke detection in video,”in EUSIPCO ’05, 2005.

4. H. J. Grech-Cini,”Smoke Detection”, US Patent No. US6844818B2.

5. C. Stauffer and W.E.L. Grimson:”Adaptive Background Mixture Models for Real-Time Tracking", Proc. IEEE Conf. Computer Vision and Pattern Recognition, 1999.

6. H. J. Catrakis, R. C. Aguirre, J. Ruiz-Plancarte, and R. D. Thayne,”Shape complexity of whole-field three-dimensional space-time fluid interfaces in turbulence", Physics of Fluids, vol. 14, iss. no. 11.