Міністерство освіти и науки України

Дніпропетровський національний університет ім. О. Гончара

Факультет прикладної математики

Кафедра математичного забезпечення ЕОМ

ЗВІТ З НАУКОВО-ДОСЛІДНИЦЬКОЇ ПРАКТИКИ

1 вересня – 12 жовтня 2015 року

Виконав: студ. гр. ПЗ-14M-1

Онищенко О.Ф.

Перевірив: доцент, к.т.н.

Байбуз О. Г.

Дніпропетровськ

2015 р.

Оглавление

[Оцінка градієнта зображення 2](#_Toc434815755)

[Гістограма напрямлених градієнтів (*Histogram of Oriented Gradients, HOG*) 3](#_Toc434815756)

[Побудова «візуального словника» (*bag of visual words approach*) 4](#_Toc434815757)

[Випадкові ліси 5](#_Toc434815758)

[Перетворення Хафа для виявлення категорій об’єктів 6](#_Toc434815759)

[Вимірювання якості роботи алгоритмів локалізації об’єктів 7](#_Toc434815760)

[Оцінка та методи побудови тренувальної вибірки. Boosting, bootstrapping 8](#_Toc434815761)

[OpenCV 9](#_Toc434815762)

[Qt. Створення графічного інтерфейсу 12](#_Toc434815763)

[Джерела 13](#_Toc434815764)

Задача розпізнавання об’єктів на зображеннях є однією з основних задач комп’ютерного зору. Є декілька різновидів цієї задачі: класифікація зображень, локалізація об’єктів шуканого класу, знаходження конкретного об’єкту, знаходження «схожих» зображень та ін. Для тренування класифікаторів використовують алгоритми, що базуються на: методі опорних векторів, нейронних мережах, деревах рішень. Широко застосовуються можливості графічних процесорів, оскільки вони дозволяють робити багато коштовних за часом математичних операцій одночасно і таким чином звільняють центральний процесор. Метою цієї роботи є дослідження алгоритму пошуку об’єктів заданого класу за допомогою випадкових лісів.

# Оцінка градієнта зображення

Градієнт – векторна величина, що вказує на напрямок та величину найшвидшого зростання або спадання деякої величини на одиницю простору.

Градієнт зображення характеризує напрямок найбільшого перепаду яскравості. Оскільки зображення дискретне, застосовують різністні похідні різних порядків апроксимації, в найпростішому випадку

,

Для знаходження похідних використовують різні оператори, наприклад оператор Собеля чи Превітта. Оператори роблять згортку зображення з ядром 3\*3 для отримання похідних по x та y, для Превітта:

,

для Собеля:

,

Величина градієнта знаходиться як , напрямок дорівнює . Для прискорення обчислень величину градієнта можна знаходити як суму модулів, а саме

# Гістограма напрямлених градієнтів (*Histogram of Oriented Gradients, HOG*)

Основне припущення полягає у тому, що зовнішній вигляд і форма об’єктів визначена розподілом величин і напрямків градієнтів. Алгоритм широко використовується для аналізу зображень, він є стійким до змін освітлення, масштабування , але є чутливим до зміни положення об’єкту у просторі.

Мета – сформувати певну кількість дескрипторів зображення. Для цього вихідне зображення розбивають на невеликі комірки (5х5, 8х8 пікселів) і для кожної комірки шукають гістограму орієнтації градієнтів за наступним алгоритмом:

1. Для кожного пікселя визначити напрямок градієнта (рис. 1а)
2. Виділити 8 напрямів орієнтації градієнтів (рис. 1б)
3. Для кожного пікселя визначити номер напряму
4. Порахувати гістограму напрямів градієнтів (рис. 1в). Гістограму можна нормалізувати



а б в

Рис.1 Алгоритм пошуку гістограми орієнтації градієнтів

На основі гістограм дескрипторів можна побудувати класифікатор, наприклад за допомогою метода опорних векторів[2].

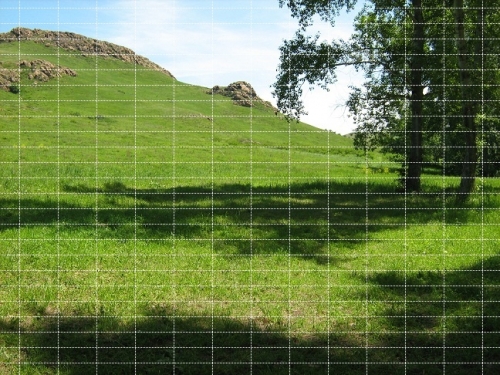
# Побудова «візуального словника» (*bag of visual words approach*)

Багато алгоритмів локалізації об’єктів використовують розбиття зображень на невеликі шматки певного розміру , де Ii – масив каналів, di  - дескриптор: вектор, що містить додаткову інформацію про шматок (наприклад, якщо шматок лежить у границі об’єкту деякого цільового класу, di може містити номер цього класу). Розбиття зображень може відбуватися декілька разів для одного й того ж зображення та за різними критеріями:

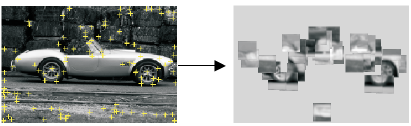
1. на зображення накладається рівномірна сітка (рис. 2 а)
2. шматок – окіл деякої «особливої» точки. «Особлива» точка може бути деяким контуром, кутом, точкою з яскравістю, що виділяється, тощо (рис. 2б)[5]
3. випадково (рис.2в)

Кожному шматку ставиться у відповідність деякий дескриптор для того, щоб шматки можна було порівнювати. Поширеною практикою є опис шматка за допомогою гістограми напрямлених градієнтів. Отримані гістограми кластеризуються одним із відомих методів, найчастіше методом к-середніх (рис. 3).

Кожному центру отриманих кластерів ставиться у відповідність кількість «схожих» шматків, таким чином у вигляді гістограми формується візуальний словник. Таким чином зображення можна характеризувати кількістю візуальних слів, які в ньому зустрічаються, або, навпаки, відсутні.



а. Рівномірна сітка б. Околи «особливих точок»



в. Випадково

Рис. 2 Отримання візуальних слів

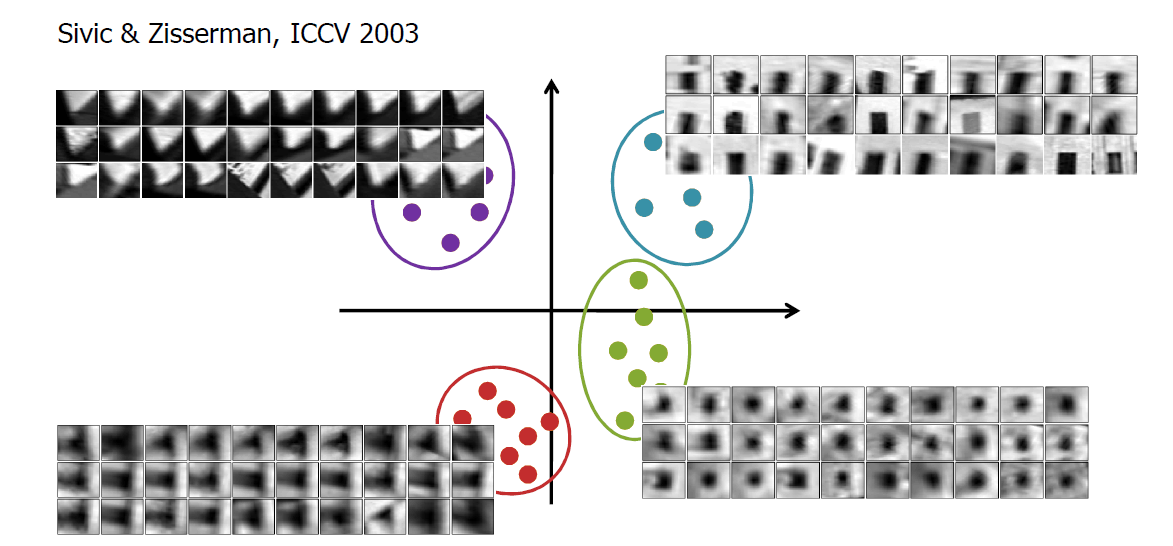


Рис. 3 Кластеризація шматків

# Випадкові ліси

Нехай кожний об’єкт характеризується множиною ознак F = {fi}. Для кожного fi можна позначити множину (діапазон) значень, які приймає ознака на множині об’єктів. Нехай pi – ймовірність появи деякого значення. Визначимо функцію неоднорідності множини , яка досягає свого мінімуму, якщо на множині об’єктів ознака приймає однакове значення, і максимуму – якщо значення різні. Використовують різні функції, наприклад

1. - індекс Гіні
2. – ентропія

Під час побудови дерева рішень правило для розбиття обирається таким чином, щоб мінімізувати міру неоднорідності двох отриманих підмножин.

Випадковий ліс[4] – множина бінарних дерев рішень, що мають наступні особливості:

1. Дерева будуються на різних наборах даних. Якщо вибірка обмежена, наприклад, потрібно m вибірок по n елементів у кожній, але є усього n елементів, екземпляри обирають випадково з n елементів з рівною вирогідністю. Якщо n велике, то кількість унікальних елементів дорівнюватиме (1 — 1/e) ≈ 63.2%
2. Для побудови дерев використовуються підмножини випадкових ознак для того, щоб дерева не корелювали між собою. Є декілька емпіричних правил для визначення кількості ознак: правило «7 із 32», або
3. Під час тестування кожний об’єкт «пропускається» через дерева лісу, кожне дерево віддає свої голос за той приналежність об’єкту до класу.

# Перетворення Хафа для виявлення категорій об’єктів

У класичному варіанті перетворення Хафа – алгоритм, що дозволяє виявити прямі лінії на зображенні. Згодом він був трансформований для виявлення кіл та довільних кривих та об’єктів. Перетворення Хафа призначене для пошуку об'єктів, що належать визначеному класу фігур з використанням процедури голосування. Процедура голосування застосовується до простору параметрів, з якого і отримують об'єкти певного класу фігур згідно локальних максимумів в, так званому, накопичувальному просторі (accumulator space) яке будується при обчисленні трансформації Хафа. У досліджуваному алгоритмі[5] простір накопичувальний простір використовується наступним чином: після проходження кожним шматком зображення кожного дерева, шматок не просто стверджує ймовірність присутності об’єкту, а голосує за центр, в якому той може знаходитися. Величина голосу пропорціональна ймовірності присутності об’єкта класу. Після отримання усіх голосів, простір згладжують і знаходять локальні максимуми будь-яким способом (рис.4).

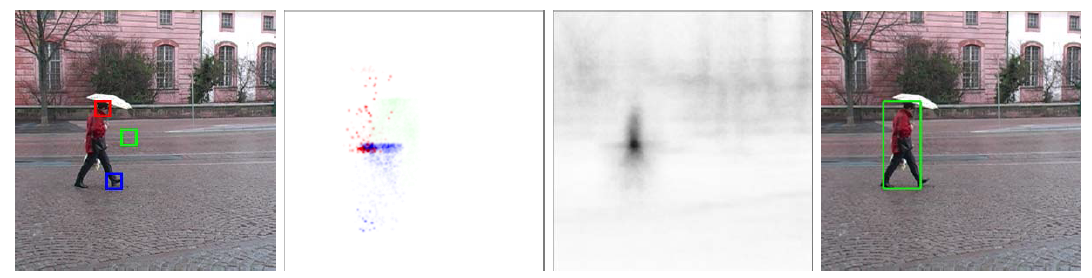


Рис.4 Зліва направо: окремі шматки, що приймають участь у голосуванні, голоси за центр від усіх дерев, зведений накопичувальний простір

# Вимірювання якості роботи алгоритмів локалізації об’єктів

Для вимірювання якості роботи алгоритмів локалізації об’єктів використовують базові поняття помилки першого та другого роду. Якщо маємо два класи областей на зображенні: об’єкт та фон, тоді в контексті комп’ютерного зору

1. помилка першого роду – якщо об’єкт не було знайдено, або ймовірність класифікувати об’єкт як фон.
2. помилка другого роду – якщо об’єкт було виявлено там, де його немає, або ймовірність «помилкової тривоги»;

Більшість алгоритмів мають можливість регулювати допустимі помилки, наприклад за деякого параметра зменшувати число помилкових локалізацій. Для оцінки використовується ROC (*receiver operating characteristic*) крива (рис. 5). По вісі y відкладають долю об’єктів, що були правильно знайдені (true positive rate, або recall), по х – кількість помилково знайдених (false positive rate, FPPW – false positive per window), або помилка другого роду. Кожна зміна параметрів алгоритму дає одну точку на кривій. Якість можна оцінити за допомогою площі, обмеженої кривою і віссю х (AUC, *area under ROC curve*). Чим більше ця площа, тим кращим вважається алгоритм.

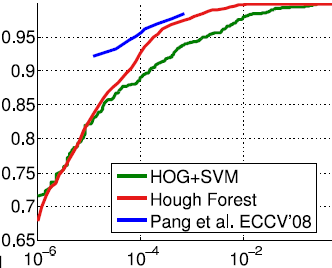


Рис. 5 ROC крива

# Оцінка та методи побудови тренувальної вибірки. Boosting, bootstrapping

За різними оцінками, для побудови достатньо гарного класифікатора одного класу об’єктів (тобто такого, що знаходить > 90% відсотків об’єктів та дає досить малу помилку другого роду порядку 10-6), потрібно в середньому 10 тис. позитивних і стільки ж негативних розмічених прикладів в якості тренувальної вибірки.

На властивостях вибірки базується багато алгоритмів локалізації, зокрема групи методів Boosting та Bootstrapping.

Ідея методів Boosting полягає у композиції більш слабких класифікаторів. Алгоритми є ітеративними і жадібними. На кожній ітерації:

1. обирається найкращий класифікатор, будуються множини об’єктів, що були класифіковані правильно та неправильно.
2. Неправильно класифікованим об’єктам за деяким законом збільшується вага.

Процес повторюється доки точність класифікації не досягне деякого порогу.

Цю ідею використовує, зокрема, метод Віоли-Джонс для пошуку облич.

Bootstrapping (бутстрэпинг) – метод, який дозволяє штучним образом збільшити об’єм тренувальної вибірки. Для кожного тренувального прикладу виконуються незначні, з точки зору людини, перетворення (викривлення), а саме *невеликі*:

* масштабування
* зсуви
* повороти
* дзеркальні відображення (зокрема, горизонтальні)

Таким чином об’єм вибірки можна збільшити в 30-50 разів. В якості негативним прикладів випадковим образом обирають таку ж кількість шматків і тренують алгоритм. Після цього тестують класифікатор на тренувальних даних і обирають «складні» false positive приклади і знову тренують декілька разів до досягнення деякого критерію якості[6].

# OpenCV

OpenCV (англ. *Open Source Computer Vision Library*) — бібліотека функцій та алгоритмів комп'ютерного зору, [обробки зображень](https://uk.wikipedia.org/wiki/%D0%9E%D0%B1%D1%80%D0%BE%D0%B1%D0%BA%D0%B0_%D0%B7%D0%BE%D0%B1%D1%80%D0%B0%D0%B6%D0%B5%D0%BD%D1%8C) і чисельних алгоритмів загального призначення з відкритим кодом. Бібліотека надає засоби для обробки і аналізу вмісту зображень, у тому числі розпізнавання об'єктів на фотографіях (наприклад, осіб і фігур людей, тексту тощо), відстежування руху об'єктів, перетворення зображень, застосування методів машинного навчання і виявлення загальних елементів на різних зображеннях.

Бібліотека написана мовою С++, але є інтерфейси для інших мов програмування : Python, Java, Matlab, обгортки для C#, Ruby.

OpenCV[7] состоит из нескольких модулей, среди которых:

* **CXCORE** – Ядро, содержит:
  + Базовые структуры данных
  + Матричную алгебру
  + Алгоритмы работы с памятью
  + Алгоритмы преобразования типов
  + Алгортмы для обработки ошибок
  + Функции для записи/чтения XML файлов
  + Функции для работы с 2D графиками
* **CV** – Модуль обработки изображений, работа с компьютерным зрением, содержит:
  + Функции для работы с изображениями (преобразование, фильтрация и т.д.)
  + Функции для анализа изображений (поиск контуров, гистограммы и т.д.)
  + Алгоритмы анализа движений, слежение за объектами
  + Алгоритмы распознания объектов (лиц, предметов)
  + Алгоритмы для калибровки камер
* **ML** – Машинное обучение:
  + Функции для классификации и анализа данных
* **HighGUI** – Модуль для создания пользовательского интерфейса, отвечает за:
  + Создание окон
  + Вывод изображений
  + Захват видео из файлов и камер
  + Чтение/Запись изображений

Основным контейнером для изображений является матрица cv::Mat.

Инициализировать матрицу можно как прочитав изображение из файла:

Mat img = imread (filename.jpg);

так и при помощи многочисленных конструкторов и метода create(), в качестве параметров могут задаваться, к примеру, размеры и глубина. Каждый элемент может быть вектором чисел размером от 1 до 4, числа разного типа, от uchar до double, на пример:

CV\_8UС1 – 8-bit, unsigned char, 1 channels

CV\_16SC3 – 16-bit, signed integer, 3 channels

Каждая матрица имеет заголовок, который содержит информацию о матрице: размер, флаги (глубина, количество каналов и др.), количество ссылок и указатель на данные .data

Доступ к элементам осуществляется через метод

.at<тип>(row\_number, col\_number)

Присваивание матриц с помощью оператора = только копирует ссылку на матрицу с целью увеличения производительности. Для копирования данных применяют методы clone и copyTo:

Mat F = A.clone();

Mat G;

A.copyTo(G);

Существует множество операций, которые можно проделать над матрицами или некоторой областью ROI (Region Of Interest), который задается с помощью типа Rect:

(A, B – матрицы, s – скаляр, alpha – вещественный скаляр):  
• сложение, вычитание, отрицание: A+B, A-B, A+s, A-s, s+A, s-A, -A;  
• масштабирование: A\*alpha;  
• поэлементное умножение/деление: A.mul(B), A/B, alpha/A;  
• умножение матриц: A\*B;  
• транспонированная матрица: A.t();  
• обращение матрицы и псевдо-инверсии, решения линейных систем и наименьших квадратов: A.inv([метод]), A.inv([метод])\*B;  
• сравнивание: A op B, A op alpha, alpha op A, где op это одно из следующих: >, >=, ==, !=, <=, <. Результат сравнения в одноканальной 8-битной матрице;  
• битовые логические операции: A op B, A op s, s op A, ~A, где op это одно из следующих: &, |, ^;  
• нахождение минимума/максимума: min(A, B), min(A, alpha), max(A, B), max(A, alpha);  
• abs(A);  
• векторное и скалярное произведение: A.cross(B) A.dot(B);  
• любая функция матрицы или матриц, которая возвращает скаляр, например, norm, mean, sum, countNonZero, trace, determinant, repeat.

Библиотека OpenCV содержит множество распространенных в компьютерном зрении операций, таких как:

* свертка, масштабирование, сглаживание, аффинные преобразования
* операторы Собеля, Щарра, Лапласа (суммирование производных второго порядка), интегральные преобразования
* детекторы границ Кенни, преобразования Хафа, нахождение контуров
* поиск объекта
* и многое другое

# Qt. Створення графічного інтерфейсу

Qt - крос-платформовий інструментарій розробки програмного забезпечення (ПЗ) мовою програмування C++. Бібліотека розділена на декілька модулів, для розробки дипломної роботи було використано лише декілька за них:

* QtCore — класи ядра бібліотеки для використання іншими модулями;
* QtGui — компоненти графічного інтерфейсу;
* QtSql — набір класів для роботи з базами даних мовою структурованих запитів SQL.
* QtDesigner — класи створення розширень QtDesigner'а для своїх власних віджетів;
* QtAssistant — довідкова система;

Qt має багато вбудованих класів, що в цілому дублюють та розширюють стандартні класи С++, зокрема Qt містить QString, QDateTime, QDir, QFile, різноманітні контейнери: QList, QLinkedList, QVector, QStack, QMap, QQueue, QSet, QHash та інші.

Виджеты (Widgets) - это визуальные элементы, из которых состоит графический интерфейс пользователя. К виджетам относятся элементы управления: QPushButton, QLabel, QLineEdit, QListWidget, QMenu, QMenuBar та інші, усього близько 50 доступних елементів. Звичайно, можна створювати свої елементи керування, якщо є потреба.

Для создания структуры виджеты организовывают в иерархию по принципу «часть-целое». Каждый из виджетов может содержать другие виджеты. Такой визуальный элемент становится родителем (родительским виджетом) для элементов, которые он содержит. Такие отношения не следует путать с наследованием из ООП. Виджеты, не имеющие родителя – отдельные окна.

Одной из фундаментальных возможностей в Qt является взаимодействие объектов с помощью сигнально-слотовых соединений. Каждый из объектов обладает состоянием, которое определяет совокупность данных, которые хранит объект в данный момент. В ответ на взаимодействие с объектом его состояние может измениться и он может уведомить об этом другой объект посылая ему сообщение об изменении, или посылая ***сигнал***. Визуальные компоненты тоже вырабатывают сигналы в ответ на действия пользователя (например, нажатия кнопки, установки флага, редактирования текста в поле ввода и т. п.). Другие объекты могут присоединиться к сигналу ***слотом –*** специальным методом, который реализует некоторую функциональность и вызывается каждый раз, когда был выпущенприсоединённый к нему сигнал.

Для задания соединения используют метод connect() класса QObject. Метод принимает пять параметров: указатель на объект, который посылает сигнал (sender); название сигнала (signal), указатель на объект, который получает сигнал (receiver);название слота (slot) и его параметры, тип сигнально-слотового соединения (имеет значение по умолчанию Qt::AutoConnection). Один и тот же сигнал объекта может быть подключён к нескольким различным сигналам и слотам другого объекта и наоборот. При этом последовательность, в которой будут вызываться присоединённые сигналы и слоты, будет соответствовать порядку, в котором их соединили[8].

# Джерела

1. **Р. Гонсалес, Р. Вудс** Цифровая обработка зображений – М.: Техносфера, 2005 – ISBN 5-94836-028-8
2. **N. Dalal, B. Triggs** Histograms of oriented gradients for human detection // *IEEE Conf. Computer Vision and Pattern Recognition* – 2005 – pp 886–893.
3. **J. Gall, V. Lempitsky** Class-specific hough forests for object detection // *IEEE Conf. Computer Vision and Pattern Recognition* – 2009
4. **L. Breiman**. Random forests. *Machine Learning*, 45(1):5–32, 2001.
5. **Sivic, Zisserman** “Video Google”// ICCV – 2003
6. **I. Laptev** Improvements of Object Detection Using Boosted Histograms//BMVC – 2006
7. <http://docs.opencv.org>
8. Інтернет-джерело doc.qt.io