## УДК 621.317.7

**Б. В. Керничний**

Національний університет “Львівська політехніка”,

кафедра спеціалізованих комп’ютерних систем

**СПЕЦІАЛІЗОВАНА СИСТЕМА РЕЄСТРАЦІЇ ТРАФІКУ ТРАНСПОРТНИХ ЗАСОБІВ**

*© Керничний Б. В., 2019*

## Проаналізовано існуючі методи та засоби обробки зображень та стрімінгових відео. Показано перспективу розвитку та застосування інноваційних форм у програмах із використанням комп’ютерного зору. Запропоновано систему для моніторингу трафіку автомобілів на основі клієнт-серверної архітектури. Пропонується створення унікального продукту, написаного мовою програмування C#, з використанням технологій .NET Framework. Система передбачає розпізнавання транспортних засобів в режимі реального часу та реєстрацію їх трафіку. Для реалізації даної ідеї розроблено клієнт-серверну архітектуру за допомогою технології ASP.NET.

**Ключові слова: технології .NET Framework, ASP.NET, клієнт-серверна архітектура.**

**SECIALIZED TRAFFIC REGISTRATIONE SYSTEM FOR TRANSORT VEHICLES**

*© Kernychnyi B. V., 2019*

## The existing methods and means of image and video processing are analyzed. The perspective of development and application of innovative forms in the program with the use of computer vision is shown. The system of monitoring traffic of cars on the basis of client-server architecture is offered. It is proposed to create a unique product written in the C # programming language using the .NET Framework technology. The system provides for the real-time recognition of vehicles and the registration of their traffic. To implement this idea, a client-server architecture was developed using ASP.NET technology.

**Key words: .NET Framework, ASP.NET, client-server architecture.**

**Вступ**

У програмах комп'ютерного зору найчастішим завданням є виявлення об'єктів і локалізація. Підходи виявлення об'єктів можна розділити на три групи: ручні методи, які складаються з певних попередньо визначених правил і евристик, підходи, засновані на машинному навчанні, де інформація про об'єкт кодується в класифікатор, і третій підхід – це відповідність шаблонам.

Хоча підходи, що базуються на машинному навчанні, як правило, є найкращими, є деякі випадки, коли вони не є бажаними. Однією з причин є те, що процедури навчання зазвичай вимагають великої кількості навчальних зразків, збір яких не завжди є простим завданням. Процедура тренування зазвичай триває довго, і підготовлений класифікатор не може бути налаштований на місці.

1. **Аналіз останніх досліджень та публікацій**

Наприкінці 1960-х років комп'ютерне бачення почалося в університетах, які були новатором штучного інтелекту. Воно було призначене для імітації людської зорової системи, як крок на шляху до наділення роботів розумною поведінкою [2]. У 1966 році вважалося, що це може бути досягнуто за допомогою літнього проекту, шляхом приєднання камери до комп'ютера і "опишіть, що він бачив". ​​[3]

Відмінність комп'ютерного зору від переважаючої сфери обробки цифрових зображень у той час була бажанням витягти  тривимірну структуру з зображень з метою досягнення повного розуміння сцени. Дослідження в 1970-х рр. Сформували ранні підвалини для багатьох алгоритмів комп'ютерного зору, які існують сьогодні, включаючи вилучення країв із зображень, маркування ліній, неполіедричне і багатогранне моделювання, представлення об'єктів як взаємозв'язків менших структур, оптичного потоку і оцінка руху. [2]

У наступному десятилітті дослідження базувалися на більш суворому математичному аналізі та кількісних аспектах комп'ютерного зору. До них відноситься концепція масштабного простору, висновок форми з різних сигналів, таких як затінення, текстура і фокус, і контурні моделі, відомі як змії. Дослідники також зрозуміли, що багато з цих математичних концепцій можна розглядати в рамках однієї оптимізаційної основи як регуляризації і марковських випадкових полів. До 1990-х років деякі з попередніх тем дослідження стали активнішими, ніж інші. Дослідження проективних 3-D реконструкцій призвели до кращого розуміння калібрування камери. З появою методів оптимізації для калібрування камери було зрозуміло, що багато ідей вже вивчено в теорії корекції зв'язків з області фотограмметрії. Це призвело до методів розріджених 3-D реконструкцій сцен з декількох зображень. Був досягнутий прогрес у проблемі щільної стереовідповідності та подальших стерео-методів з декількома переглядами. У той же час для вирішення сегментації зображень використовувалися варіації розрізу графа. Це десятиліття також ознаменувало перші статистичні методи навчання, які використовувалися на практиці для розпізнавання облич на зображеннях (див. Eigenface). Наприкінці 1990-х років відбулася суттєва зміна взаємодії між сферами комп'ютерної графіки та комп'ютерного зору. Це включає в себе зображення на основі візуалізації, зображення морфінг, перегляд інтерполяції, панорамне зображення зшивання і ранній світлового поля візуалізації.

Нещодавні роботи свідчать про відродження функціонально-орієнтованих методів, що використовуються в поєднанні з методами машинного навчання і складними рамками оптимізації.

## Мета публікації

## Метою роботи є аналіз різних технологій реалізації комп’ютерного зору, підбір найкращих технологій, вибір середовища та мови програмування для створення системи. Вибір архітектури системи.

**3. Обґрунтування актуальності поставленої задачі**

Процедури узгодження шаблонів є простими і можуть бути достатньо надійними для вирішення складних сцен, і вони не вимагають великої підготовки. Крім того, накладений шаблон надає об'єкту зовнішній вигляд, який є неможливим за допомогою популярних підходів рухомих вікон для машинного навчання. Ця здатність може бути використана для точної локалізації об'єкта як автономної процедури або як техніки після обробки (наприклад, після техніки Віола-Джонса). В 2018 році запропоновано нову процедуру узгодження шаблонів Hinterstoisser et al. [1] Це приблизно в 20 разів швидше, ніж стандартні підходи підбору шаблонів рухомих вікон. Ця процедура (трохи покращена) доступна як частина Accord.NET Extensions Framework і буде пояснена тут коротко разом з зразками та сценаріями використання.

1. **Вибір засобу розробки**

## Процедуру узгодження шаблонів можна розділити на п'ять кроків, які будуть коротко описані нижче. Для повного пояснення процедури наведена нижче діаграма:

## https://www.codeproject.com/KB/graphics/826377/templateMatching.png

*Рис. 1. Покрокове виконання процедури узгодження шаблонів.*

Найважливішою стадією попередньої обробки є вилучення країв, де зображення орієнтації краю є вхідним зображенням для методу відповідності шаблону [1]. Однією з основних причин вибору орієнтації зображення є їх стійкість до великих змін світла та шуму зображення. Величина градієнта погано реагує на сцени з фоновим безладдям через багато помилкових спрацьовувань. У [1] використовується багатоканальний оператор Sobel, де орієнтація пікселів береться з каналу, який має найбільшу величину. Крім того, орієнтації фільтруються таким чином, що зберігаються лише домінантні орієнтації в сусідстві 3×3. Реалізований метод підтримує описаний підхід, а також традиційне вилучення орієнтації з зображень у градаціях сірого для менш вимогливого до обчислення методу, частина якого наведена нижче.

*Лістинг 1*

short\* dxPtr, dyPtr = ... //input dX and dY Sobel derivative images

int\* magSqrPtr = ... //output gradient magnitude image

int\* orientImagePtr = ... //output orientation image

for (int j = 0; j < imgHeight; j++)

{

for (int i = 0; i < imgWidth; i++)

{

int magSqr = dxPtr[0] \* dxPtr[0] + dyPtr[0] \* dyPtr[0];

if (magSqr < minValidMagSqr)

\*orientImgPtr = FeatureMap.INVALID\_ORIENTATION;

else

{

\*orientImgPtr = MathExtensions.Atan2Aprox(\*dyPtr, \*dxPtr);

\*magSqrImgPtr = magSqr;

}

dxPtr += 1; dyPtr += 1;

orientImgPtr += 1; magSqrImgPtr += 1;

}

...

}

Кожна орієнтація градієнта квантується в максимально 8 напрямках для того, щоб відобразити кожну орієнтацію як бітовий прапор. Представлення можна легко маніпулювати. Наведений нижче код показує, що процедура квантування використовує попередньо обчислену таблицю пошуку для квантування кутів.

*Лістинг 2*

int\* orientDegImgPtr = ... //orientation image pointer

int imgWidth, imgHeight = ... //image width and height

byte[] AngleQuantizationTable = ... //maps [0-360] -> [...] -> [0-7]

for (int j = 0; j < imgHeight; j++)

{

for (int i = 0; i < imgWidth; i++)

{

int angle = orientDegImgPtr[i];

qOrinetUnfilteredPtr[i] = AngleQuantizationTable[angle];

}

...

}

Об'єкти зображення можуть деформуватися через фізичні характеристики або через шум, що призводить до не досконалого вирівнювання краю між об'єктом і відповідним шаблоном. З метою підвищення надійності орієнтації поширюються на місцеві околиці. Оскільки кожна орієнтація може бути записана як одиничний біт, орієнтація розповсюдження просто побітове або між деяким пікселем і його сусідніми пікселями, як показано нижче.

*Лістинг 3*

byte\* srcImgPtr = ... //input quantized orientation image

byte\* destImgPtr = ... //output spread quantized orientattion image

int neighborhood = ... //spread factor

for (int row = 0; row < neghborhood; row++)

{

int subImageHeight = imgHeight - row;

for (int col = 0; col < neghborhood; col++)

{

//get shifted image and

//do the bitwise OR operation for each pixel

OrImageBits(&srcImgPtr[col], destImgPtr,

imgStride,

imgWidth - col, subImageHeight);

}

...

}

Шаблон узгоджується з кожним розташуванням у зображенні за допомогою операції, еквівалентної ковзанню шаблону над вхідним зображенням. На відміну від стандартної процедури узгодження шаблонів, вхідне зображення попередньо обробляється таким чином, щоб процедура узгодження була виконана дуже швидко, додавши довгі масиви - лінійні пам'яті. Детальнішу інформацію див. У [1]. Кожна лінійна комірка пам'яті містить подібність в діапазоні [0..n], де n - максимальна задана користувачем подібність між двома квантованими орієнтаціями. Ці значення обмежені кількістю бітів у 8-бітове значення. Лінійне додавання пам'яті може бути зроблено дуже ефективно за допомогою архітектури процесора SIMD, що забезпечує відповідність шаблонів у реальному часі.

*Лістинг 4*

//calculate linear response maps

private Image<Gray<byte>>[][,] calculate(Gray<int>[,] orientationDegImg)

{

//the number of linear memories is equal to the number of quantized orientations

var linearMaps = new Image<Gray<byte>>[GlobalParameters.NUM\_OF\_QUNATIZED\_ORIENTATIONS][,];

//first create spread quantized orientation image

Gray<byte>[,] sprededQuantizedOrient = FeatureMap.Calculate(orientationDegImg, this.NeigborhoodSize);

//for each quantized orientation calculate pre-calculate similarity image

//(response map) between the orientation image and specified orientation

for (int orient = 0; orient < GlobalParameters.NUM\_OF\_QUNATIZED\_ORIENTATIONS; orient++)

{

//...and finally make linear memory for each response map

//by taking every Tth pixel and putting them into one array

var responseMap = computeResponseMap(sprededQuantizedOrient, orient);

linearMaps[orient] = linearizeResponseMap(responseMap);

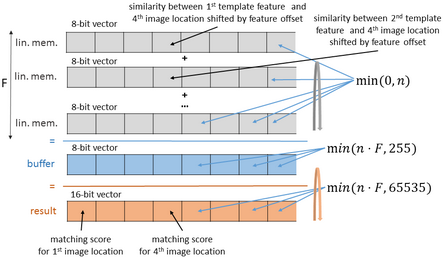
}

return linearMaps;

}</byte>

**Збільшення кількості функцій - розширення**

8-розрядний вектор результатів диктує максимальну кількість ознак в шаблоні, оскільки кожна лінійна осередок містить значення, максимум якого є максимальною подібністю між двома квантованими орієнтаціями - n і кількість лінійних пам'яті, що додаються, відповідає кількості шаблонів. Отже, максимальна кількість ознак в шаблоні, в оригінальній статті, обмежена 255 / n⌋. Щоб подолати це обмеження, лінійні карти спочатку додаються до тимчасового буфера - 8-бітового вектора, а потім перед тим, як буфер переповнюється, буфер додається до кінцевого масиву результатів (16-біт), який містить результати порівняння шаблонів. Ця процедура повторюється для кожної функції шаблону. Буфер очищається після накопичення його значень у масиві результатів. Детальнішу інформацію див. На малюнку нижче.

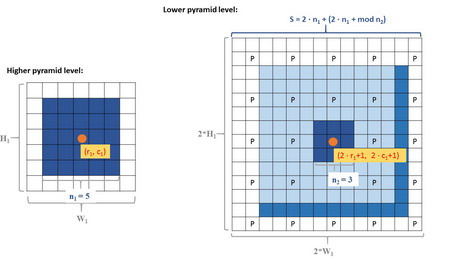


*Рис. 2. Буфер очищення.*

Результатом є збільшення максимальної кількості ознак ⌊65535 / n⌋. Порівняння продуктивності показує значне прискорення порівняно з традиційним підходом до шаблонів. n позначає максимальну схожість між двома ознаками і F позначає кількість ознак.

**Пірамідальне розширення**

Такий підхід може бути розширений до використання пірамідального підходу, як показано в оригінальній статті. Представлення лінійної пам'яті будується для кожного рівня піраміди, і процедура узгодження виконується для найменшого зображення і для кожного латки нижчих рівнів піраміди, які містять кандидати. Цей підхід забезпечує прискорення і більш точну локалізацію об'єктів. Реалізований код підтримує цю функцію, але використовується лише один рівень (початковий рівень).



*Рис. 3. Патч розміром 55.*

Процедура узгодження шаблону виконується для кожного рівня піраміди. Зображення більш високого рівня піраміди сканується цілком. Тут показаний патч розміром 5×5 пікселів. Якщо кандидат знаходиться на зображенні піраміди вищого рівня, то патч перерозподіляється так само, як і шаблон (виконується оф-лайн через побудову шаблону), і процедура узгодження повторюється, але на цей раз для патчів-кандидатів.

**Висновки**

У цій статті представлено метод швидкого узгодження шаблонів, який можна використовувати для швидкого і простого виявлення об'єктів, зокрема, розпізнавнання транспорту. Код описує сценарій із виявленням автомобіля, що регулюється для інших типів об'єктів. Джерело і зразок коду є частиною Accord.NET Extensions Framework, який містить багато просунутих алгоритмів насамперед для обробки зображень, виявлення об'єктів і відстеження.

*1. Hinterstoisser, S.; Cagniart, C.; Ilic, S.; Sturm, P.; Navab, N.; Fua, P.; Lepetit, V., "Gradient Response Maps for Real-Time Detection of Textureless Objects," Pattern Analysis and Machine Intelligence, IEEE Transactions on , vol.34, no.5, pp.876,888, May 2012. 2.* Richard Szeliski (30 September 2010). *[Computer Vision: Algorithms and](https://books.google.com/books?id=bXzAlkODwa8C)*  Springer Science & Business Media. pp. 10–16. 3. "The Summer Vision Project". *MIT AI Memos (1959 - 2004)*.