МЕТОД РОЗПІЗНАВАННЯ ТЕКСТУ З ЗОБРАЖЕННЯ

Угреватов Д.І.
Науковий керівник - к.т.н., доц. Кобилін О.А.
Харківський національний університет радіоелектроніки
(61166, Харків, пр. Науки, 14, каф. Інформатики, тел. (057) 702-13-19)
E-mail: dmytro.uhrevatov@nure.ua

Text recognition by automated methods is one of the most popular tasks of computer image recognition. This technology is used in many areas: in government structures, commercial projects, everyday life of ordinary people, etc. Several public libraries, such as OpenCV, have now been created that allow working with digital data, but their use still requires a certain technical education. Such libraries of computer vision do not contain ready-made solutions and are only a tool for creating working algorithms.

Для розпізнавання тексту використовується методи контурного аналізу. Основа контурного аналізу - контурний аналіз дозволяє, описувати, зберігати, порівнювати і робити пошук об'єктів, представлених у вигляді своїх зовнішніх обрисів – контурів. Контур букви або цифри з вхідного зображення порівнюються з контуром цього ж символу в загальноприйнятому шаблоні. Для даної роботи спочатку був зроблено демонстраційний шаблон, який містить в собі зразок зображення букв і цифр. Також використовується одна з найпопулярніших бібліотек для роботи з зображеннями OpenCV - бібліотека алгоритмів комп'ютерного зору, обробки зображень та чисельних алгоритмів загального призначення з відкритим кодом. Реалізована на C / C ++, також розробляється для Python, Java, Ruby, Matlab, Lua та інших мов. Може вільно використовуватися в академічних і комерційних цілях.

Метод складається з 3 етапів: попередня обробка зображення, виділення символів тексту, порівняння контурів.

Попередня обробка:

1. Переводимо зображення в градації сірого. У OpenCV це робиться за допомогою функції cvtColor().
2. Для видалення шумів використовуємо медіанний фільтр, він добре справляється з шумами типу «перець і сіль», а так само покращує контур, що важливо для подальшого контурного аналізу. У OpenCV для цього використовується функція medianBlur().
3. Наступний крок містить в собі бінаризація зображення або переведення зображення в чорно-біле за допомогою функції thresholdMat().

Виділення символів. На даному етапі відбувається виділення кожного символу в окреме зображення:

1. За допомогою детектора кордонів Кенні, реалізованого в OpenCV у вигляді функції Canny(), на зображенні можна виділити кордону всіх об'єктів.

2. Потім за допомогою функції findContours(), можна отримати контури об'єктів у вигляді набору точок з координатами. Маючи набір точок для кожного контуру можна визначити найменший прямокутник, який буде містити область всередині контуру. Знайти прямокутник можна за допомогою функції boundingRect().

3. Прямокутник буде знову ж представлений у вигляді набору точок його контуру. Використовуючи цю інформацію, виділяємо на початковому зображенні контур кожного прямокутника і зберігаємо внутрішню область у вигляді нового зображення.

Порівняння контурів:

1. Контур кожного символу порівнюється з контуром цього ж символу в нашому шаблоні за допомогою функції computeDistance() з бібліотеки OpenCV. Принцип роботи функції наступний: на вхід подається два контури, а на виході чисельне значення. Чим більше число, тим менше схожі контури. В основі методу лежить порівняння точок контурів між собою. Відшукуються схожі точки і будується гістограма відмінностей, по якій вважається відстань.

В ході експериментів було встановлено, що функція computeDistance() дає досить точні результати і чутлива тільки до якості вхідного зображення і еталонного зображення символу. На жаль, існує одна істотна недолік - час роботи. Так як порівняння контурів йде попіксельно, при хорошому вхідному зображенні контури можуть досягати розмірності від 250 до 500 пікселів. Неодноразовий перебір такого масиву вимагає великої обчислювальної потужності.

Список використаних джерел:

1. Garcia G. B. et al. Learning Image Processing with OpenCV. – Packt Publishing Ltd, 2015.

2. Belongie S., Malik J., Puzicha J. Shape matching and object recognition using shape contexts //IEEE Transactions on Pattern Analysis & Machine Intelligence. – 2002. – №. 4. – С. 509-522.

3. Dawson-Howe K. A practical introduction to computer vision with opencv. – John Wiley & Sons, 2014.

4. Suzuki S. et al. Topological structural analysis of digitized binary images by border following //Computer vision, graphics, and image processing. – 1985. – Т. 30. – №. 1. – С. 32-46.

5. Bradski G., Kaehler A. Learning OpenCV: Computer vision with the OpenCV library. – " O'Reilly Media, Inc.", 2008.