УДК 004.652

Мишанський О. Ю., студент гр. 126м-21з-1
Науковий керівник: В.Ю. Каштан, к.т.н., доцент кафедри інформаційних технологій та комп'ютерної інженерії
(Національний технічний університет "Дніпровська політехніка", м. Дніпро, Україна)

РОЗРОБКА СИСТЕМИ РОЗПІЗНАВАННЯ ВІЛЬНИХ МІСЦЬ ПАРКУВАННЯ НА ОСНОВІ R-CNN МЕРЕЖ

Сьогодні у великих мегаполісах одним з найскладніших завдань є пошук вільного місця для паркування. Згідно з опитуванням IBM [1], близько 40% дорожнього руху в містах насправді складається з транспортних засобів, водії яких шукають місця для паркування. Це призводить до появи проблем таких як викиди забруднюючих речовин, затори на дорогах і втрачений час, не кажучи вже про сприяння нещасних випадків через те, що водії зосереджені на пошуку місця для паркування. За останні кілька років було розроблено багато високотехнологічних систем, які допомагають водієві швидко та ефективно шукати вільні місця для паркування. Так, одним із способів розпізнавання вільних парувальних місць є використання нейронної мережі, що має дві переваги порівняно з сенсорними – це універсальність і нижчу вартість. Тому, в даній роботі пропонуємо використати архітектуру R-CNN мережі для розпізнавання вільних місць паркування (рис.1).

Спочатку виконаємо розпізнавання місць для паркування на основі Faster-RCNN [2, 3]. Для цього створимо базу даних, що містить знімки або фотографії з зайнятими та вільними місцями парковки у вигляді файлів XML. Після цього виконується попередня обробка, а потім оброблені зображення передаються для навчання в Faster-RCNN. Етап навчання показує, наскільки точно машина може прогнозувати місця для паркування. Далі етап створення образів. Після цього активується модуль класифікації, що містить набір даних перевірки вільних місць та доставляється в систему протягом 3-5 секунд, і класифікатори одночасно працюють над цими зображеннями, щоб створити прогноз зайнятості місця для паркування.

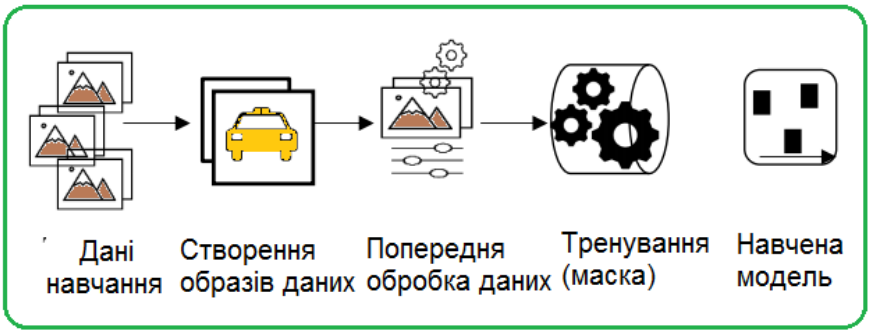

Рисунок 1 – Етапи системи розпізнавання образів

Далі необхідно визначити машини, які розташовані лише у рядах паркування. Для цього необхідно визначити середину кожної рамки автомобіля для реалізації використовували Python. Потім створюємо об'єкти класу Polygon і Point. У конструкторі класу Polygon передаємо отримані ряди паркування, конструктор класу Point, отримуємо серединні точки рамок автомобілів. Визначимо наявність точки у полігоні за допомогою методу класу Polygons, у параметрах якого передаємо точку. Таким чином, отримаємо тільки ті автомобілі, які знаходяться всередині парковки. Відсортуємо за зростанням отримані серединні точки в кожному ряду паркування. Щоб встановити горизонтальний або вертикальний ряд паркування, необхідно знайти одну із

сторін прямокутної рамки автомобіля, що перетинає відрізок та з'єднує серединну точку першого та наступного автомобіля.

Щоб встановити горизонтальний або вертикальний ряд паркування, необхідно знайти одну із сторін прямокутної рамки автомобіля, що перетинає відрізок та з'єднує серединну точку першого та наступного автомобіля

Розіб'ємо прямокутну рамку на чотири сторони, проаналізуємо всі сторони і визначимо за допомогою методу intersects сторону, яку перетинає відрізок, що з'єднує серединні точки. Таким чином, якщо *x* координата першої та другої вершини перетнутої сторони збігаються, то відстань буде братися по *x* координаті, в іншому випадку по *y* координаті.

Далі кожному відрізку, що з'єднує центри рамок двох сусідніх автомобілів, можна порівняти необхідну відстань для вільного місця.

Для визначення необхідної відстані для вільного *i*-го місця запам'ятаємо дистанцію між *x* координатою правого краю рамки *i-1* автомобіля і *x* координати лівого краю *i+1* для горизонтального ряду. У разі вертикального ряду зафіксуємо дистанцію між *y* координатою верхнього краю рамки *i -1* автомобіля та *y* координатою нижнього краю *i +1*.

Встановивши стартову точку, що дорівнює середині відрізка початку ряду, будемо проходити за сортованим списком у вигляді рамок автомобілів, що знаходяться в ряду.

Визначимо, чи є необхідна відстань між стартовою точкою та початковою гранню обмежуючої рамки першого елемента у списку. Відстань визначатиметься за допомогою відрізків, що з'єднують центри рамок автомобілів, отриманих із детальної розмітки паркування. Встановимо, в який із відрізків потрапила стартова точка, і візьмемо відповідну відстань для нього та випадки перетину рамок (ширина рамки для даного відрізка). Якщо дана відстань існує, то занесемо координати вільного місця до списку з вільних місць. Нову стартову точку визначимо, використовуючи значення перетину, зіставленого відрізку. При негативному значенні перетину, нова стартова точка визначатиметься як старе значення плюс відстань необхідного вільного місця. При позитивному значенні визначатиметься як старе значення плюс відстань і значення перетину. Якщо відстані немає, то нова стартова точка буде визначатись, як значення крайньої межі рамки поточного автомобіля.

Таким чином, аналізуючи всі ряди паркування, визначимо список з усіх вільних місць. Виконаємо візуалізацію отриманих рамок для кожного порожнього місця на зображенні з паркування, використовуючи засоби OpenCV.

ПЕРЕЛІК ПОСИЛАНЬ

6. IBM Survey. Available online: https://www-03.ibm.com/press/us/en/pressrelease/35515.wss (accessed on 20 August 2019).
7. P. Bharati, A. Pramanik, Deep Learning Techniques—R-CNN to Mask R-CNN: A Survey, in: A. K. Das, J. Nayak, B. Naik, S. K. Pati, D. Pelusi (Eds.), Computational Intelligence in Pattern Recognition, Springer Singapore, Singapore, 2020, pp. 657–668. doi:10.1007/978-981-13-9042-5_56.
8. R. S. Zimmermann, J. N. Siems, Faster training of Mask R-CNN by focusing on instance boundaries, Computer Vision and Image Understanding 188 (2019) 102795. doi:10.1016/.cviu.2019.102795.