УДК 681.3.016

МАКЕТ З'ЄДНАННЯ І ПРОГРАМНА РЕАЛІЗАЦІЯ КОНТРОЛЮ ЗОРУ ДЛЯ МОБІЛЬНИХ РОБОТІВ НА БАЗІ RASPBERRY PI ТА МОВИ ПРОГРАМУВАННЯ PYTHON

Д.т.н. І. Ш. Невлюдов, д.екон.н. Ш. А. Омаров, к.т.н. В. В. Євсєєв, Я. О. Радченко, Харківський національний університет радіоелектроніки

У зв'язку з інтенсивною роботизацією кожної сфери обслуговування, виникла серйозна необхідність створення роботів, які могли б не тільки вміти рухатися по заздалегідь визначеними маршрутами і виявляти перешкоди, але і класифікувати їх, щоб при необхідності гнучко та швидко підстроїтися під змінне оточення. Таке завдання реалізується на основі використання технічного зору. У даній роботі розглянуті основні моменти реалізації технічного зору, на прикладі, використання міні-комп'ютера Raspberry Pi3 з мовою програмування Python 3.

В связи с интенсивной роботизацией каждой сферы обслуживания, возникла серьезная необходимость создания роботов, которые могли бы уметь не только двигаться по заранее определенным маршрутам и выявлять препятствия, но и классифицировать их, чтобы при необходимости гибко и быстро подстроиться под изменяющееся окружение. Такая задача реализуется на основе использования технического зрения. В данной работе рассмотрены основные моменты реализации технического зрения, на примере, использования мини-компьютера Raspberry Pi3 с языком программирования Python 3.

Due to the intensive robotization of each service sector, there was a serious need to create robots that could be able to not only move along predefined routes and identify obstacles, but also classify them in order to adapt flexibly and quickly to a changing environment if necessary. Such a task is realized through the use of technical vision. This paper discusses the main points in the implementation of technical vision, for example, the use of a Raspberry Pi3 mini-computer with the Python 3 programming language.

***Ключові слова**: роботизація, змінне оточення, технічний зір, міні-комп'ютер, Python 3.*

Вступ

В даний час широко поширений такий вид руху роботів, як їзда по лінії. Це широко використовується як на початковому рівні створення і впровадження в малі галузі промисловості та для тестингу, так і для великих підприємств, заводах, які вже не перший рік користуються і щоденно використовують роботизовані системи для полегшення праці.

AGV (Automatic Guided Vehicle) – роботи переміщаються по заздалегідь намальованих лініях, та здійснюють ту чи іншу операцію яка закладена в середині системи. При організації робото змагань – часто можна зустріти завдання, так чи інакше пов'язані з орієнтуванням по лініях. Головною інформацією за допомогою яких робот отримує дані, є інфрачервоні датчики, які визначають різницю в контрасті лінії і навколишнього фону, і магнітні датчики, які використовуються в разі застосування магнітних ліній.

Але, як правило, завжди на виробництві існує безліч рухомих об'єктів, які потенційно можуть стати перешкодою для робота, тому робот завжди повинен не тільки отримувати інформацію про свій маршрут і рухомих об'єктах, але також повинен аналізувати оточення, щоб у разі потреби мати можливість швидко відреагувати на ситуацію, що виникла [1].

При розробці мобільних роботів потрібно враховувати, які завдання поставлені перед мобільним роботом, щоб підібрати необхідні сенсорні пристрої для їх вирішення. Так, наприклад, для вирішення завдання переміщення всередині робочої зони робот може бути оснащений лазерними сканерами і GPS-пристроями для визначення власного положення, тоді як для вирішення завдання локальної навігації мобільний робот може бути оснащений простими ультразвуковими або інфрачервоними датчиками по периметру [1].

Адже саме його комплектація та грамотне призначення для чого він потрібен є найважливішим фактором у визначенні того, чи буде ваш продукт корисний для покупця та надійний при використанні.

Методи дослідження

Методом дослідження служитиме міні-комп'ютер RASPBERRY PI3 з операційною системою GNU/LINUX, дистрибутив DEBIAN. Звісно ж програмною й найважливішою частиною для взаємодії і створення між мобільним роботом та навколишнім середовищем буде на користь мова програмування PYTHON. Головним завданням буде реалізація системи контролю зору для мобільних роботів та їх головний принцип зв'язку.

Щоб досягнути реальної мети в розробці певної технології потрібно чітко розуміти, для чого буде розроблений цей продукт, та чи буде корисне дане введення для сучасного світу. Але до теперішнього часу немає досконалого та чіткого алгоритму розробки системи безконтактного руху мобільних роботів, щоб вони не робили свої помилки і могли без жодних перебоїв контактувати с динамічною середою, що ускладнює роботу у даному напрямі. Це, у свою чергу, породжує потребу в науковому дослідженні питань якості новітніх систем реалізації та впровадження в світ щось нового.

Для вирішення зазначеної вище проблеми на допомогу приходять системи технічного зору, що дозволяють роботу отримувати максимально повну інформацію про стан оточення навколо нього. По суті - система технічного зору є «очима» робота, здатними за допомогою камери оцифровувати навколишню область і надавати інформацію про фізичні характеристики об'єктів, що знаходяться в ній у вигляді даних про:

- розміри;
- розташування в просторі;
- зовнішній вигляд (колір, стан поверхні і т.д.);
- маркування (розпізнавання логотипів, штрих-кодів і т.д.).

Одержані дані можуть бути використані для ідентифікації об'єктів, вимірювання їх будь-яких характеристик, а також управління ними.

В основі системи технічного зору лежить цифрова камера, яка знімає навколишній простір, потім отримані дані обробляються процесором з використанням певного алгоритму аналізу зображення для виділення з них і класифікації параметрів. На цьому етапі йде підготовка і висновок даних у вигляді, зручному до обробки контролером робота. Потім дані передаються безпосередньо на контролер робота, де ми їх можемо використовувати для управління роботом.

На рисунку 1 наведено приклад того, як система Візирів на дорозі отримує інформацію та ініціалізує її.

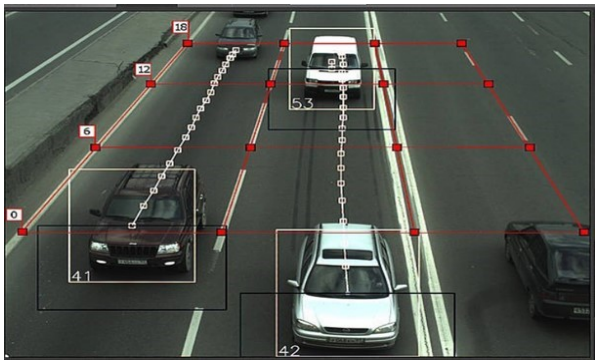

Рис. 1. Приклад системи розпізнання технічного зору об'єктів у певному діапазоні

При розробці макета буде використане додаткове технічне обладнання. STEREOPI – це стереоскопічна камера з відкритим кодом на основі RASPBERRY PI3. Він може захоплювати, зберігати, транслювати і обробляти в реальному часі стереоскопічне відео і зображення. STEREOPI відкриває безмежні можливості в галузі робототехніки, AR/VR, комп'ютерного зору, інструментів дронів. Технічне обладнання яке потрібне щоб реалізувати макет (рис. 2) [2]:

– RASPBERRY PI3 міні-комп'ютер, найважливіша частина для розробки макета та програмування всіх його частин (рис. 2, а). Дана модель дозволяє приєднання до нього зовнішніх пристроїв і управління ними за допомогою різних програмних пакетів (найбільш популярним є PYTHON). Різні датчики, світлодіоди, двигуни, реле та інші електронні компоненти можуть підключатися через GPIO контакти. Тому ми можемо істотно розширювати функціонал мікрокомп'ютера, створюючи з нього робочу станцію для кожного конкретного проекту;

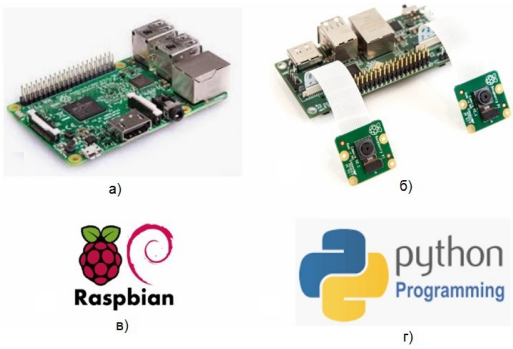

Рис. 2. Технічне обладнання, що використовуються при збиранні макету

– STEREOPI є платою-носієм для обчислювального модуля RASPBERRY PI і сумісна з усіма варіантами обчислювальних модулів: CM1, CM3 (рис. 2, б). Головна перевага цього носія – це відмінна підтримка PYTHON та тонни прикладу коду [2];

– з RASPBERRY PI в своїй основі STEREOPI, природно, використовує стандартний RASPBIAN. (рис. 2, в). RASPBIAN поставляється з встановленим програмним забезпеченням для освіти, програмування і загального користування. Він має Python, Scratch, Sonic Pi, Java і багато іншого;

– PYTHON, мова програмування в RASPBIAN для створення всієї програми та алгоритмів дій (рис. 2, г). Звичайно, RASPBIAN поставляється з PYTHON, а це значить, що почати розробку зі STEREOPI так само просто, як встановити PICAMERA, чистий інтерфейс PYTHON для модуля камери RASPBERRY PI, який підтримує стерео захват, наприклад:

```
from picamera import PiCamera
camera = PiCamera(stereo_mode = 'side-by-side', resolution = (1280,
                                    720))
camera.capture('foo.jpg')
```

PICAMERA – це пакет надає чистий інтерфейс PYTHON для модуля камери RASPBERRY PI для PYTHON 2.7 (або вище).

Для використання алгоритмів комп'ютерного зору повинні не забувати о такій бібліотеці як: OPENCV (Open Source Computer Vision) – фактично, – це набір типів даних, функцій і класів для обробки зображень алгоритмами комп'ютерного зору. Реалізована на C/C ++.

Процес інсталяції та взаємодії між об'єктами

Ми вже знаємо, що будемо працювати з такою операційною системою як GNU/LINUX, на початку праці повинні оновити нашу ОС:

```
apt-get update
apt-get upgrade
```

Завантажимо OPENCV. Вилучимо архів zip-файл. При цьому буде створена директорія /usr/src/opencv-2.4.13.7

```
cd /usr/src
wget
https://github.com/opencv/opencv/archive/2.4.13.7.zip
unzip 2.4.13.7.zip
```

Далі створюємо всередині каталог <release>, переходимо в нього і налаштовуємо проект.

```
cd opencv-2.4.13.7
mkdir release
cd release
cmake -D CMAKE_BUILD_TYPE=RELEASE -D
CMAKE_INSTALL_PREFIX=/usr/local ..
```

Якщо все закінчиться добре, ми отримаємо наступне повідомлення:

```
-- Configuring done
-- Generating done
-- Build files have written to: /usr/local/src/opencv-
2.4.13.7/release
```

Далі потрібно скомпілювати результат.

Після того як вдасться успішно скомпілювати OPENCV, робимо установку: <make install>. Коли були інстальовані певні бібліотеки то ми можемо перейти до збірки нашого макету. Отже розроблений метод та взаємодію між нашою системою можна графічно показати у вигляді алгоритму представленому на рисунку 3.

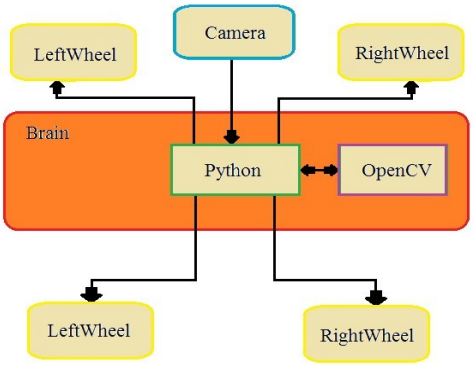

Рис. 3. Алгоритм взаємодії моделі

Як ми бачимо – «мозок» робота отримує зображення, розпізнає (за допомогою OPENCV) і передає керуючі команди на колеса. І замінивши зображення з камери на 3D зображення, а управління колесами – на управління віртуальними колесами 3D робота у віртуальному світі – ми отримаємо макет для відпрацювання логіки.

Отримуємо, таку архітектуру для віртуального макета (рис. 4):

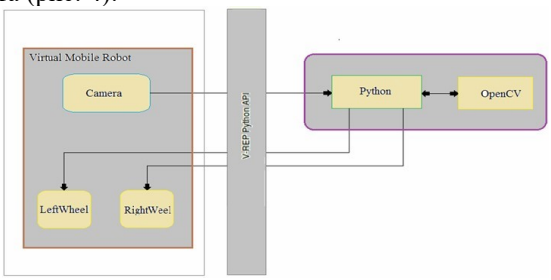

Рис. 4. Макет перетворення у 3D реальність

Необхідно вирішити задачу зв'язку V-REP із зовнішнім PYTHON скриптом, який виконує розпізнавання образів за допомогою OPENCV, і виводить маркер навколо знайденого об'єкта.

Виходить така архітектура (рис. 5):

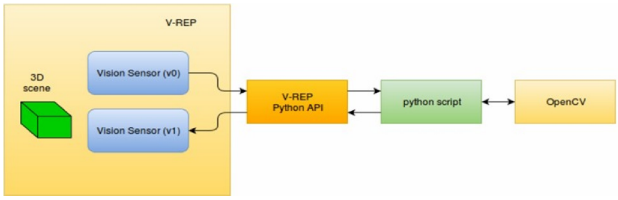

Рис. 5. Зв'язок між камерою спостереження та діями після обробки сигналу

Як бачимо, ми знову прибігаємо до підключення внутрішніх різних систем для розробки нашого макету, на прикладі V-REP.

І так, що таке V-REP? – Це середовище для моделювання роботів. V-REP компанії Coppelia Robotics – один з найдосконаліших симуляторів на даний час. Програмний комплекс є кросплатформним і безкоштовним для використання в освітніх цілях. Симулятор складається з фізичного і графічного движка, що дозволяє досить комфортно працювати з програмою [4].

Наступним і остатнім кроком буде сама мова програмування PYTHON [5].

При запуску V-REP автоматично завантажується плагін який забезпечує зв'язок PYTHON API. Для роботи з API через PYTHON скрипт, необхідна наявність трьох файлів в папці:

- remoteApi.so (або remoteApi.dll)
- vrep.py
- vrepConst.py

Їх можна скопіювати з V-REP каталогу. Після цього можна імпортувати модуль V-REP:

```
import vrep
```

Найпростіший скрипт, що підключається до API, виглядає так:

```
import vrep
vrep.simxFinish(-1)
clientID = vrep.simxStart('127.0.0.1', 199997, True, True, 5000, 5)
if clienID != -1:
  print('Connected to remote API server')
  while (vrep.simxGetConnectionId(clientID) != - 1):
    print('some work')
else:
  print('Failed to connect to remote API server')
  vrep.simxFinish(clientID)
```

На нашому макеті є два об'єкти – v0 і v1 – це VISUAL SENSOR з першого ми зчитуємо картинку, на другий ми повинні записувати результат. Тому ми повинні отримати ці об'єкти в контексті нашого PYTHON скрипта, робиться це за допомогою API команди vrep.simxGetObjectHandle:

```
res, v0 = vrep.simxGetObjectHandle(clientId, 'v0',
vrep.simx_opmode_oneshot_wait)
res, v1 = vrep.simxGetObjectHandle(clientId, 'v1',
vrep.simx_opmode_oneshot_wait)
```

Після виконання останнього кроку: завантаження програмного інтерфейсу для зв'язку з PYTHON API та імпортування модуля V-REP, ми отримали рішення задачі зв'язку V-REP із зовнішнім PYTHON скриптом, яка дозволяє користуватись програмою з комп'ютерних пристроїв, використовуючи повноцінний функціонал макета.

Ґрунтуючись на розробці реалізації задачі за допомогою PYTHON для досягнення мети дослідження, необхідно підвищення ефективності V-REP симулятора на етапах проектування або вдосконалення.

Висновки

Цей алгоритм є основою для створення такого типу задач, звісно розроблений макет може доповнюватися ще більш цікавими бібліотеками та зовнішніми компонентами зв'язку.

В подальшій роботі можливо, за результатами аналізу наукових джерел або статей, модернізувати програмну частину макета, щоб діапазон можливостей нашого робота був ще в декілька разів більше та розробити логіку для прийняття адекватних рішень при взаємодії з динамічним середовищем.

СПИСОК ЛІТЕРАТУРИ:

1. Володин Ю.С. Телевизионная система объемного зрения для управления движением мобильного робота: диссертация / Ю.С. Володин. – М. : Знания, 2011. – 209 с.

2. Сайт CROWDSUPPLY. An open source stereoscopic camera based on Raspberry Pi [Электронный ресурс]: Режим доступу : https://www.crowdsupply.com/virt2real/stereopi.

3. Сайт habr.com [Электронный ресурс]: Режим доступу: https://habr.com/ru/post/473750/ – Назва з екрану.

4. [Электронный ресурс]: Режим доступу : http://www.count-zero.ru/2016/vrep_intro/.

5. Коробков А. Ф., Барасий В. В. Python та анализ кода //Научно-технический вестник информационных технологий. – 2002. – №. 6. – С. 153-155.