Міністерство освіти і науки України

Національний університет водного господарства та природокористування

Кафедра автоматизації, електротехнічних та комп'ютерно-інтегрованих технологій

**04-03-272**

# МЕТОДИЧНІ ВКАЗІВКИ

до лабораторної роботи №7
з навчальної дисципліни
***«Інтелектуальні системи управління»***
для здобувачів вищої освіти другого (магістерського) рівня за освітньо-професійною програмою «Автоматизація та комп'ютерно-інтегровані технології» спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології» денної та заочної форм навчання

Рекомендовано науково-методичною радою з якості ННІАКОТ
Протокол № 4 від
24 грудня 2019 р.

Рівне – 2019

Методичні вказівки до лабораторної роботи №7 з навчальної дисципліни «Інтелектуальні системи управління» для здобувачів вищої освіти другого (магістерського) рівня за освітньо-професійною програмою «Автоматизація та комп'ютерно-інтегровані технології» спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології» денної та заочної форм навчання [Електронне видання] / Реут Д. Т. – Рівне : НУВГП, 2019. – 15 с.

Укладач: Реут Д. Т., старший викладач кафедри автоматизації, електротехнічних та комп'ютерно-інтегрованих технологій.

Відповідальний за випуск: Древецький В. В., д.т.н., професор, завідувач кафедри автоматизації, електротехнічних та комп'ютерно-інтегрованих технологій.



Керівник групи
забезпечення спеціальності Древецький В. В.

## Лабораторна робота 7. Використання комп’ютерного зору в інформаційно-вимірювальних системах

Мета роботи: навчитись використовувати бібліотеку комп'ютерного зору OpenCV для отримання даних із зображень в інтелектуальних інформаційно-вимірювальних системах.

### Теоретичні відомості

Галузь комп'ютерного зору почала розвиватись лише з появою комп’ютерів, достатньо продуктивних, щоб обробляти великі масиви пікселів, з яких складається зображення. Нині відсутній загальноприйнятий спосіб вирішення проблеми комп’ютерного зору. Замість цього існує маса методів для вирішення різноманітних строго визначених задач комп'ютерного зору, де методи часто залежать від задач і рідко коли можуть бути узагальнені для широкого кола застосування. Багато з методів та застосувань все ще знаходяться на стадії фундаментальних досліджень, але все більша кількість методів знаходить застосування в комерційних продуктах, де вони часто складають частину складнішої системи, яка може вирішувати складні задачі (наприклад, в галузі медичних зображень або вимірювання та контролю якості в процесах виробництва).

Як наукова дисципліна, комп'ютерний зір належить до теорії та технології створення штучних систем, які отримують інформацію у вигляді зображень.

Як технологічна дисципліна, комп'ютерний зір прагне застосувати теорії та моделі комп'ютерного зору до створення систем комп'ютерного зору, зокрема системи керування процесами (промислові роботи, автономні транспортні засоби), системи відео спостереження, системи організації інформації (наприклад, для індексації баз даних зображень), системи моделювання об'єктів або оточуючого середовища (аналіз медичних зображень,

топографічне моделювання), системи взаємодії (наприклад, пристрої введення для систем людино-машинної взаємодії).

Оскільки розпізнавання образів та навчання нейронних мереж, що аналізують зображення, тісно переплітаються з власне комп'ютерним зором, іноді він розглядається як частина галузі штучного інтелекту. Тому важливо розрізняти отримання й обробку кадрів, що виконується за детермінованими алгоритмами, як от вирівнювання нерівномірності освітлення, перетворення колірної моделі, поворот зображення, що часто називають машинним зором, від виконання на додачу до цих функцій розпізнавання облич, відстеження об'єктів, пошук об'єктів аналізом кадру за допомогою нейромережі, що іноді називають теж комп'ютерним зором, але в ширшому розумінні, як частину штучного інтелекту.

***Бібліотека комп'ютерного зору OpenCV***

OpenCV (open source computer vision library) - це бібліотека комп'ютерного зору з відкритим вихідним кодом, реалізована на мові C++. Також, вона доступна для інших мов, наприклад, Python, Java, Ruby, Matlab, Lua. Включає в себе різні алгоритми комп'ютерного зору, розпізнавання зображень і багато іншого, що працюють в реальному режимі часу. Всі бажаючі можуть використовувати бібліотеку OpenCV безкоштовно, як в освітніх цілях, так і в комерційних проектах.

Вона включає в себе наступні алгоритми:

- Розпізнавання об'єктів в потоці.
- Розпізнавання друкованого і рукописного тексту.
- Усунення спотворень картинки.
- Виявлення подібності та форми об'єктів.
- Стеження за переміщенням об'єкта.
- Розпізнавання рухів, жестів і багато іншого.

Підтримувані засоби GUI, захоплення відео:

• Windows: DirectShow, VFW, MIL, CMU1394;

• Linux: Video4Linux2, DC1394, FFMPEG.

Відкрита ліцензія для OpenCV була складена таким чином, щоб було можливо створювати комерційні додатки, використовуючи будь-які можливості OpenCV. Існує велика спільнота користувачів, що включає в себе такі великі компанії як IBM, Microsoft, Intel, Sony, Siemens, Google, інші, а також науково-дослідні центри, такі як Стенфорд, Массачусетський технологічний інститут, CMU, Кембридж, і INRIA.

***Складові бібліотеки OpenCV***

• opencv_core - основна функціональність. Включає в себе базові структури, обчислення (математичні функції, генератори випадкових чисел) і лінійну алгебру, DFT, DCT, введення / виведення для XML і YAML і т. д.

• opencv_imgproc - обробка зображень (фільтрація, геометричні перетворення, перетворення колірних просторів і т. д.).

• opencv_highgui - простий інтерфейс користувача, введення / виведення зображень і відео.

• opencv_ml - моделі машинного навчання (SVM, дерева рішень, навчання зі стимулюванням і т. д.).

• opencv_features2d - розпізнавання і опис плоских примітивів (SURF, FAST й інші, включаючи спеціалізований фреймворк).

• opencv_video - аналіз руху і відстеження об'єктів (оптичний потік, шаблони руху, усунення фону).

• opencv_objdetect - виявлення об'єктів на зображенні (перебування осіб за допомогою алгоритму Віоли-Джонса, розпізнавання людей HOG і т. д.).

• opencv_calib3d - калібрування камери, пошук стерео-відповідності та елементи обробки тривимірних даних.

• opencv_flann - бібліотека швидкого пошуку найближчих сусідів (FLANN 1.5) і обгортки OpenCV.

• opencv_contrib - супутній код, ще не готовий для застосування.

• opencv_legacy - застарілий код, збережений заради зворотної сумісності.

• opencv_gpu - прискорення деяких функцій OpenCV за рахунок використання відеокарт, що підтримують CUDA.

Аналіз відеопотоку потребує досить швидкого виконання заданих в програмі алгоритмів, оскільки через кілька сотих секунди надійде наступний кадр з камери чи файлу і до цього моменту попередній кадр повинен бути оброблений. Накопичення необроблених кадрів буде призводити до затримок в інтерфейсі користувача. Тому обробку відеопотоку, як правило, реалізовують на компільованих мовах програмування, а в галузях, де не потрібен аналіз відеопотоку в реальному часі, як-от обробка фотографій, використовуються й інтерпретовані мови.

Для зберігання переліків об’єктів і їх властивостей може використовуватись масив структур, об’єктів користувацького класу або сутності з бібліотеки STL: вектор, список тощо.

Однією з галузей застосування комп’ютерного зору є виявлення, відстеження, підрахунок та класифікація мікроскопічних об’єктів: клітин у тканинах та біологічних рідинах у медицині, мікроорганізмів у поверхневих водах в екологічному моніторинзі та водному господарстві.

***Приклад програми***

Програму, що відстежує об’єкти (наприклад, мікроорганізми) на відео та виводить їх кількість, за допомогою бібліотеки OpenCV можна реалізувати наступним кодом:

```
#include <opencv2/highgui/highgui.hpp>
```

```
#include <opencv2/objdetect/objdetect.hpp>
#include <opencv2/imgproc/imgproc.hpp>
#include <iostream>
#include <vector>
#include <list>
#include <iterator>
#include <stdio.h>
#include <unistd.h>
#include <signal.h>
#include <sys/time.h>
#define TO_SCREEN 1       // макрос для умовної
компіляції: 0 - без виведення відео (може
виконуватись на пристрої без дисплея), 1 - з
виведенням у вікно
using namespace cv;
using namespace std;
vector<vector<Point> > contours;
vector<Vec4i> hierarchy;
FILE *output;
unsigned int contoursCount=0;
vector<Point> currp;          //вектор для
динамічного масиву точок початків контурів на
поточному кадрі
vector<bool> exist_in_prev;//вектор для
динамічного масиву ознак "існує на попередньому"
точок на поточному кадрі
vector<int> areas;          //вектор для
динамічного масиву з площами контурів

struct Track_object {//Відстежуваний об'єкт:
    int index;        //номер (від моменту
запуску)
    int x;            //координати точки
    int y;
    bool updated;     //ознака того, що точка
знайдена на поточному кадрі
    };
std::list<Track_object> tracking_points;
//список видимих точок, які відслідковуються на
поточному кадрі
```

```
std::list<Track_object>::iterator it;
int total_points_num=0;  //лічильник об'єктів

int xmax,xmin,ymax,ymin;
unsigned int i,j;
Track_object tmp;

int minObjArea=50;  //мінімальна площа контура,
при якій об'єкт починає відслідковуватись
int sqMaxSpeed=15;  //сума квадратів переміщень
в обох координатах за один кадр, більше якої
вважаємо, що це 2 різні об'єкти
int threshold_type=THRESH_BINARY_INV;
int adaptive_method=ADAPTIVE_THRESH_MEAN_C;
int block_size=43;  // Розмір блоку та зміщення
за замовчуванням
int offset=12;

void alrm_handler(int inp)
{
   if((output=fopen("/tmp/objnum.txt",
"w"))==NULL) {
                cerr << "Не вдалося відкрити файл
для запису результатів." << endl;
                exit(1);
        }
        fprintf(output,"%d \t%d\n",contoursCount,
total_points_num); //запис результату в файл
        fclose(output);
}

void help(char** av)
{
    cout << "Формат виклику:\n" << av[0] << "
<номер камери/назва файлу>\n" << endl;
}

#if TO_SCREEN==1
void on_trackbar(int, void*)
{
```

```
    ; //тут може розміщуватись обробник події
переміщення повзунка в GUI
}
#endif

int main(int ac, char** av)
{
    if (ac != 2)
    {
        help(av);
        return 1;
    }
    std::string arg = av[1];
    VideoCapture capture(arg);  //спроба
використати вхідний аргумент як ім'я відеофайлу
    if (!capture.isOpened())    //якщо не
вдалося, вважати аргумент номером відеокамери
        capture.open(atoi(arg.c_str()));
    if (!capture.isOpened())
    {
        cerr << "Помилка відкриття відеокамери
або відеофайлу!" << endl;
        help(av);
        return 1;
    }
    capture.set(CV_CAP_PROP_FRAME_WIDTH,640);
//задаємо розмір кадру
    capture.set(CV_CAP_PROP_FRAME_HEIGHT,480);
    int w =
(int)capture.get(CV_CAP_PROP_FRAME_WIDTH);
    int h =
(int)capture.get(CV_CAP_PROP_FRAME_HEIGHT);
    cout << "Ширина="<<w<<endl<<"Висота="<<h<<
endl;

    int n = 0;
    char filename[200];
    string window_name = "video | q або esc для
виходу";
```

```
    cout << "Натисніть пробіл для збереження
зображення, q або esc для виходу" << endl;
    #if TO_SCREEN==1
    //створення вікна для виводу відео
    namedWindow(window_name, CV_WINDOW_NORMAL);
    char text[30];
    // створення повзунків
    createTrackbar( "Block size", window_name,
&block_size, 80, on_trackbar,0 );
    createTrackbar( "Offset", window_name,
&offset, 20, on_trackbar,0 );
    createTrackbar( "minObjArea", window_name,
&minObjArea, 500, on_trackbar,0 );
    on_trackbar( block_size,0 );
    #endif
    //створення таймера 1 с для виводу
результатів у файл
    struct sigaction sa;
       sa.sa_handler = alrm_handler;
       sigaction(SIGALRM, &sa, 0);
    struct itimerval value;
    value.it_interval.tv_sec = 1;
    value.it_interval.tv_usec = 0;
    value.it_value.tv_sec = 1;
    value.it_value.tv_usec = 0;
    setitimer(ITIMER_REAL, &value, NULL);

    //створення файлу для виводу кількості
контурів
    if((output=fopen("/tmp/objnum.txt",
"w"))==NULL) {
               cerr << "Не вдалося створити файл
для запису результатів." << endl;
               exit(1);
       }
       fclose(output);
    //виділення пам'яті під зображення
    Mat frame;

    for (;;)
```

```
{
            capture >> frame; //отримання кадру
з відео
            if(frame.empty()) break;
            Mat gray;
            //перетворення у відтінки сірого
            cvtColor(frame,gray,CV_RGB2GRAY);
            Mat bin;
            //перетворення в бінарне адаптивним
порогом
            if(block_size%2==0) block_size++;
           // block_size повинен бути непарним
числом не менше 3
            if(block_size<3) block_size=3;
            adaptiveThreshold(gray, bin, 255,
adaptive_method,threshold_type, block_size,
offset);
            //очищення перед наступним кадром
for(it=tracking_points.begin();it!=tracking_poin
ts.end();it++) it->updated=false;
            currp.clear();
            exist_in_prev.clear();
            areas.clear();
            contours.clear();
            hierarchy.clear();
            //пошук контурів
            findContours( bin, contours,
hierarchy,RETR_EXTERNAL,CHAIN_APPROX_SIMPLE,Poin
t(0,0));
            //отримання кількості контурів
            contoursCount=contours.size();
            //опрацювання знайдених контурів
            for( i = 0; i < contoursCount; i++ )
            {  areas.push_back((int)fabs(
               contourArea(contours[i],false)));
               //пошук площі, обмеженої контуром
               exist_in_prev.push_back(false);
               //координати, що обмежують контур
               xmax=contours[i][0].x;
               xmin=xmax;
```

```
            ymax=contours[i][0].y;
            ymin=ymax;
            for(j=1;j<contours[i].size();j++)
            {
             if(contours[i][j].x<xmin)
xmin=contours[i][j].x;
              if(contours[i][j].x>xmax)
xmax=contours[i][j].x;
             if(contours[i][j].y<ymin)
ymin=contours[i][j].y;
             if(contours[i][j].y>ymax)
ymax=contours[i][j].y;
             }
            currp.push_back(Point((xmin+
xmax)/2,(ymin+ymax)/2));    //середина контура
            }
            //оновлення координат та видалення
зниклих об'єктів
          for(it=tracking_points.begin(); it !=
tracking_points.end();) {
            for(i=0;i<contoursCount;i++) {
              if((((signed int)(currp[i].x)-
(signed int)(it->x))^2)+(((signed
int)(currp[i].y)-(signed int)(it-
>y))^2)<=sqMaxSpeed && !exist_in_prev[i]) {
//якщо відстань мала та точка не асоційована з
іншим контуром
                it->x=currp[i].x;
                it->y=currp[i].y;
//оновити координату точки
                it->updated=true;
//встановити ознаку "координата об'єкту
оновлена"
                exist_in_prev[i]=true;
//встановити ознаку "точка асоційована з
попереднім об'єктом"
                }
              }
            if(!(it->updated))
tracking_points.erase(it++); //видалення
```

```
об'єктів, які існували на попередньому кадрі,
але відсутні на поточному
                    else ++it;
                }
            // додавання нових знайдених
об'єктів
            for(i=0;i<contoursCount;i++)
              if((!exist_in_prev[i])&&
(areas[i]>=minObjArea)) {
                tmp.index=(++total_points_num);
                tmp.x=currp[i].x;
                tmp.y=currp[i].y;
                tmp.updated=true;
                tracking_points.push_back(tmp);
                }
           #if TO_SCREEN==1
            // виведення номерів об'єктів на
відео, що виводиться користувачу
            for(it=tracking_points.begin();
it!=tracking_points.end();it++) {
                sprintf(text,"%d",it->index);
                putText(frame, text, Point(it-
>x,it->y), CV_FONT_HERSHEY_SIMPLEX, 0.5,
Scalar(0,0,255), 2, 8, false);
                }
            sprintf(text,"InFrame=%d",
contoursCount);  //кількість на поточному кадрі
           putText(frame, text, Point(1,400),
CV_FONT_HERSHEY_SIMPLEX, 1, Scalar(0,0,255), 2,
8, false);
           sprintf(text,"Sum=%d",
total_points_num); //сума за весь час роботи
           putText(frame, text, Point(1,455),
CV_FONT_HERSHEY_SIMPLEX, 1, Scalar(0,0,255), 2,
8, false);
           line(frame, Point(540,470),
Point(625,470), Scalar(0,0,255), 3, 8, 0);
//створення масштабної лінійки
           sprintf(text,"0.1mm");
```

```
            putText(frame, text, Point(530,460),
CV_FONT_HERSHEY_SIMPLEX, 1, Scalar(0,0,255), 2,
8, false);
            //малювання контурів
            drawContours( frame, contours, -1,
Scalar(0,250,250), 1, 8, hierarchy, 1);
            //виведення всього зображення (GUI)
            imshow(window_name, frame);
            #endif
            char key = (char) waitKey(33);
//чекаємо клавішу ESC для виходу або SPACE для
збереження кадру
            switch (key)
            {
                case 'q':
                case 'Q':
                case 27: //ESC
                    capture.release();
                    #if TO_SCREEN==1
                    cvDestroyAllWindows();
                    #endif
                    return 0;
                case ' ': // SPACE зберігає
зображення
                    sprintf(filename,
"frame%3d.jpg", n++);
                    imwrite(filename, frame);
                    cout << "Збережено " <<
filename << endl;
                    break;
                default:
                    break;
            }
        }
```



**Порядок виконання роботи**

1. Ознайомитись з теоретичними відомостями.

2. Завантажити з веб-сторінки https://opencv.org/releases.html C++ бібліотеку OpenCV та встановити її, дотримуючись інструкцій зі сторінки.

3. Перевірити успішність компіляції програми-прикладу на комп'ютері. Якщо отримана помилка про відсутність заголовних файлів OpenCV, вказати коректний шлях до встановленої бібліотеки OpenCV та повторити компіляцію.

4. Розробити програму, що реалізовуватиме задачу згідно варіанту та перевірити правильність її роботи.

**Варіанти завдань**

1, 6 – виведення на відео (OSD) кількості об'єктів, розмір яких перевищує 10x10 пікселів,

2, 7 – виділення червоним контурів об'єктів на відео, розмір яких перевищує 50x50 пікселів;

3, 8 – виділення синім контурів об'єктів на відео, розмір яких менше 25x25 пікселів;

4, 9 – виділення зеленим контурів об'єктів на відео, розмір яких в межах від 15x15 до 70x70 пікселів;

5, 10 – запис у файл кадрів з відео, де кількість об'єктів перевищує 10.

**Контрольні запитання**

1. На якій мові написана бібліотека OpenCV?
2. Яка частина бібліотеки OpenCV дозволяє перетворити зображення в колірний простір HSV?
3. Алгоритми для яких задач реалізовані в бібліотеці OpenCV?
4. Яка частина бібліотеки OpenCV надає засоби для виведення зображення та відео користувачеві?
5. В яких галузях використовується комп'ютерний зір?

**Інформаційні ресурси**

1. OpenCV 2.4.13.7 documentation. URL: https://docs.opencv.org/2.4.13.7/.

2. OpenCV Tutorials. URL: https://docs.opencv.org/2.4/doc/tutorials/tutorials.html.