МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний аерокосмічний університет ім. М. Є. Жуковського
«Харківський авіаційний інститут»

MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE
National Aerospace University «Kharkiv Aviation Institute»

Л. О. Краснов, О. В. Гавриленко

ОБ'ЄКТНО-ОРІЄНТОВАНЕ ПРОЕКТУВАННЯ СИСТЕМ КЕРУВАННЯ
(з використанням Python і бібліотеки OpenCV)

Навчальний посібник

L. Krasnov, O. Havrylenko

OBJECT-ORIENTED DESIGN OF CONTROL SYSTEMS
(Python code and OpenCV library resources)

Tutorial

Харків «ХАІ» 2020
Kharkiv «KhAI» 2020

УДК 616.8+612.17+681.3
К81

The tutorial contains materials for theoretical and practical study of object-oriented approach to software development of control systems, in particular systems with technical vision, are offered. Python language capabilities using OpenCV library resources for image and video processing are discussed in details. Numerous case studies examples are considered.

The tutorial is intended for students of the preparation areas "Automation and Computer-Integrated Technology", "Avionics", "Aviation Transport". It can be used in carrying out laboratory works in the disciplines "Object-oriented design", "Systems of technical vision", as well as in course and diploma design of students of the specified specialties.

Рецензенти: канд. техн. наук О. О. Кулаєнко,
канд. техн. наук С. М. Флерко

Краснов, Л. О.

К81 Об’єктно-орієнтоване проектування систем керування [Текст] : навч. посіб. / Л. О. Краснов, О. В. Гавриленко. – Харків : Нац. аерокосм. ун-т ім. М. Є. Жуковського «Харків. авіац. ін-т», 2020. – 184 с.

ISBN 978-966-662-751-6

Запропоновано матеріали для теоретичного й практичного вивчення об’єктно-орієнтованого підходу до створення програмного забезпечення систем керування, зокрема систем із технічним зором. Докладно розглянуто можливості мови Python з використанням ресурсів бібліотеки OpenCV для оброблення зображень і відеоінформації. Наведено багато прикладів.

Для студентів напрямів підготовки «Автоматизація та комп’ютерно-інтегровані технології», «Авіоніка», «Авіаційний транспорт». Може бути використаний при проведенні лабораторних робіт з дисциплін «Об’єктно-орієнтоване проектування», «Системи технічного зору», а також під час курсового й дипломного проектування.

Іл. 78. Табл. 14. Бібліогр.: 16 назв

УДК 616.8+612.17+681.3

ISBN 978-966-662-751-6

ПЕРЕДМОВА

Цей посібник пропонується для набуття студентами теоретичних і практичних навичок програмування при розробленні й експлуатації сучасних систем технічного зору. Особлива увага − методам оброблення зображень і відеоданих.

Сьогодні під час практичної реалізації систем технічного зору в більшості випадків використовують мову програмування ***Python***. Це обумовлено багатьма причинами, головна з яких полягає в тому, що ***Python*** має простий і зрозумілий синтаксис, легко вивчається і годиться для виконання різних проектів, а також для програмування математичних обчислень. Поєднаємо ***Python*** з такими мовами, як ***Fortran***, ***C*** і ***C++***, що використовуються в наукових та інженерних розрахунках.

До ***Python*** легко підключити спеціалізовані бібліотеки, що розширюють можливості вирішення потрібних застосувань. Наприклад, бібліотека ***Python Imaging Library (PIL)*** є дуже ефективною при обробленні й аналізі зображень. Особливо важливо, що спільно з ***Python*** використовується ***OpenCV (Open Source Computer Vision)*** – бібліотека алгоритмів комп'ютерного зору для оброблення зображень і відеоданих з відкритим кодом.

Сподіваємося, що Вам удасться здолати труднощі вивчення і використовувати отримані навички в повсякденній практиці дослідження й проектування систем технічного зору.

INTRODUCTION

This manual is offered for students to obtain theoretical and practical programming skills in the development and opera-tion of modern vision systems. Particular attention is paid to image and video processing methods.

Currently, in the practical implementtation of vision systems, in most cases they use the ***Python*** programming language. This is due to several reasons, the main of which is that ***Python*** has a simple and clear syntax, is easy to learn, and well suited for various projects. It is good for programming math calculations. compatible with languages such as ***Fortran***, ***C*** and ***C ++***, which are used in scientific and engineering calculations.

It is easy to connect specialized libraries to ***Python*** that expand the possibilities of solving the necessary applications. For example, the ***Python Imaging Library (PIL)*** library is very effective at processing and analyzing images. It is especially important that in conjunction with ***Python***, ***OpenCV (Open Source Computer Vision Library)*** is used − a library of computer vision algorithms for processing images and video data with open source.

We hope that you will be able to overcome the difficulties of studying and use the acquired skills in the daily practice of research and design of vision systems.

Частина 1. СУЧАСНІ РЕСУРСИ ОБРОБЛЕННЯ ДАНИХ У СИСТЕМАХ ТЕХНІЧНОГО ЗОРУ

1. УСТАНОВЛЕННЯ ПРОГРАМ ТА ОПТИМІЗАЦІЯ ЇХ СТРУКТУРИ

Сучасні системи технічного зору орієнтовано на використання мови програмування ***Python*** на базі операційної системи ***Windows*** при їх стаціонарному розміщенні. У мобільних системах технічного зору, що базуються на рухомих носіях, зазвичай застосовують одноплатну платформу ***Raspberry P***i з рекомендованою виробником операційною системою ***Raspbian***. При цьому базовою мовою програмування також є ***Python***.

Основні завдання технічного зору з оброблення зображень і відеоданих при програмуванні на ***Python*** прийнято вирішувати з використанням бібліотеки ***Pillow*** (Python Imaging Library), що забезпечує досить повний набір функцій і методів оброблення зображень і відео. Однак найбільш ефективним є спільне використання ресурсів бібліотеки ***Pillow*** і бібліотеки ***OpenCV*** (***Open Source Computer Vision Library)***.

Зручним є застосування ***OpenCV*** разом з ***Python*** для створення простих і зрозумілих програм, проведення експериментів і синтезу різних прототипів. Мовою ***Python*** доступна практично вся функціональність бібліотеки ***OpenCV***.

Part 1. MODERN RESOURCES OF DATA PROCESSING IN TECHNICAL VISION SYSTEMS

1. INSTALLATION OF PROGRAMS AND OPTIMIZATION THEIR STRUCTURE

Modern vision systems are focused on the use of the ***Python*** programming language based on the ***Windows*** operating system with their stationary placement. Mobile technical systems based on mobile media typically use the ***Raspberry Pi*** single board platform with the manufacturer's recommended ***Raspbian*** operating system. The basic programming language is also ***Python***.

The main tasks of technical vision in image and video processing when programming in ***Python*** are decided using the ***Pillow library*** (***Python Imaging Library***), which provides a fairly complete set of functions and methods for image and video processing. However, sharing the resources of the Pillow library and the ***Open CV (Open Source Computer Vision Library)*** is most effective.

OpenCV operation in conjunction with ***Python*** is convenient for creating simple and understandable applications, conducting experiments and synthesizing various prototypes. Almost all the functionality of the ***OpenCV*** library is available in ***Python***.

При використанні векторно-матричної ідеології, аналогічної прийнятому підходу в ***Matlab***, можна отримувати досить швидкі коди, що забезпечують вирішення завдань у реальному масштабі часу. Для цього слід уникати оброблення зображень попіксельно та інших масивів поелементно в циклах.

Замість цього доцільно використовувати векторні функції, які працюють відразу з усім зображенням або масивом.

Далі розглянемо основні властивості й методику встановлення програмного забезпечення ***Python*** і підключення бібліотеки ***OpenCV***.

1.1. Загальна інформація про бібліотеку OpenCV

OpenCV − це комп'ютерний зір з відкритим вихідним кодом і програмна бібліотека машинного навчання. Бібліотеку ***OpenCV*** було створено для забезпечення загальної інфраструктури додатків комп'ютерного зору і для прискорення використання машинного сприйняття в комерційних продуктах. Будучи ліцензованим продуктом, ***OpenCV*** спрощує використання й модифікацію коду.

Основні відомості про характеристики бібліотеки розміщено на сайті http://opencv.org. Тут можна завантажити бібліотеку для різних платформ, знайти повну документацію щодо бібліотеки, швидких установок (***Quick Start***), уроки з її використання (***Tutorials***), довідкову карту (***Cheat Sheet***), а також посилання на корисні сайти й т. ін.

На сайті http://docs. opencv.org

When using vector-matrix ideology similar to the accepted approach in ***Matlab***, you can get fairly fast codes that provide problem solving in real time. To do this, you should avoid pixel-by-pixel image processing and other arrays element-wise in cycles.

Instead, it is advisable to use vector functions that work immediately with the entire image or array.

Next, we consider the basic properties and methods of installing the ***Python*** software and connecting the ***OpenCV*** library.

1.1. General information about the OpenCV library

OpenCV is an open source computer vision and a machine learning library. The ***OpenCV*** library was created to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in commercial products. Being a licensed product, ***OpenCV*** simplifies the use and modification of the code.

Basic information about the characteristics of the library is available at http://opencv.org. From there you can download a library for different platforms, find full documentation on the library, quick installation instructions (***Quick Start***), lessons on using it (***Tutorials***), a reference map (***Cheat Sheet***), and links on useful sites, etc.

The site http://docs.opencv.org

зібрано документацію, зокрема, детальні описи всіх функцій з повнотекстовим пошуком. На сайті http://answers. opencv.org можна ставити запитання, що стосуються бібліотеки, і отримувати на них відповіді від розробників або інших користувачів.

OpenCV має інтерфейси ***C ++, C, Python***, ***Java*** і ***Matlab*** і підтримує ***OC Windows***, ***Linux, Android*** і ***Mac OS***. Бібліотека ***OpenCV*** орієнтується в основному на додатки в режимі реального часу. Нині активно розвиваються повнофункціональні інтерфейси ***CUDA*** і ***OpenCL***.

Існує понад 500 алгоритмів і близько 10-кратної кількості функцій, які становлять або підтримують ці алгоритми. ***OpenCV*** написано мовою ***C++*** і має шаблонний інтерфейс.

Для початку роботи з бібліотекою ***OpenCV*** зазвичай рекомендують використовувати її разом з мовою ***Python*** (http://python.org). Програму цією мовою можна писати в будь-якому текстовому редакторі. Далі без попередньої компіляції вона виконується інтерпретатором ***Python***. ***Python*** з бібліотекою ***NumPy*** перетворюється на безкоштовний аналог системи ***Matlab***, при цьому сама мова є більш розвиненою, ніж ***Matlab***. Ще одна перевага ***Python*** полягає в тому, що ця мова є добре документованою. В інтерактивній сесії ***Python*** (у командному вікні) можна швидко отримати список доступних функцій і методів конкретного класу, швидку довідку про кожну функцію ***OpenCV***.

collected all the documentation, In particular, detailed descriptions of all functions with full-text search. At http://answers.opencv.org, you can ask questions about the library and get answers from developers or other users.

OpenCV has ***C ++***, ***C***, ***Python***, ***Java*** and ***Matlab*** interfaces and supports ***Windows***, ***Linux***, ***Android*** and ***Mac OS***. The ***OpenCV*** library focuses mainly on real-time applications. Currently, fully functional ***CUDA*** and ***OpenCL*** interfaces are actively developing.

There are more than 500 algorithms and about 10 times the number of functions that make up or support these algorithms. ***OpenCV*** is written in ***C ++*** and has a template interface.

To get started with the ***OpenCV*** library, it is usually recommended to use its bundle with the ***Python*** language (http://python.org). A program in this language can be written in any text editor. Further without preliminary compilation it is executed by the ***Python*** interpreter. ***Python*** with the ***NumPy*** library turns into a free analogue of the ***Matlab*** system, while the language itself is more developed than ***Matlab***. Another advantage of ***Python*** is that the language is well documented. In an interactive ***Python*** session (in the command window), you can quickly get a list of the available functions and methods of a particular class, a quick reference for each ***OpenCV*** function.

1.2. Установлення програми-інтерпретатора Python

Для запуску програм мовою ***Python*** потребується програма-інтерпретатор (віртуальна машина ***Python***). Ця програма приховує від ***Python***-програміста всі особливості операційної системи, тому, написавши програму мовою ***Python*** в системі ***Windows***, її можна запустити, наприклад, у ***GNU/Linux*** і отримати такий же результат.

Завантажити та встановити інтерпретатор ***Python*** можна безкоштовно з офіційного сайту http://python.org. Для роботи нам знадобиться інтерпретатор ***Python*** версії 3 або вище. Розглянемо як приклад установку версії програми ***Python 3.6.1*** для операційної системи ***Windows 7***.

Для успішного початку роботи необхідно скачати з офіційного сайту https://www.python.org/downloads/ версію програми 3.6.1.

На рис. 1.1 показано головне вікно цього сайту. Виберіть пункт меню ***Downloads***, у списку версій виберіть пункт ***Python 3.6.1***. Тут і далі потрібні пункти меню підкреслено червоною лінією.

Після вибору версії ***Python 3.6.1*** натисніть кнопку ***Download***, у вікні ***Files*** (рис. 1.2) виберіть операційну систему, що використовується на вашому комп'ютері. Її розрядність має відповідати вашій версії ***Windows***. У нашому прикладі це ***Windows x86 executable installer***. Цей пункт меню також підкреслено червоною лінією.

1.2. Install Python interpreter-program

To run ***Python*** programs you need an interpreter (***Python*** virtual machine). This program hides from the ***Python*** programmer all the features of the operating system, therefore, writing a ***Python*** program in the ***Windows*** system, you can run it, for example, in ***GNU/Linux*** and get the same result.

Download and install the ***Python*** interpreter can be free from the official site: http://python.org. To work, we need a ***Python*** interpreter version 3 or higher. Consider as an example the installation of the version of the ***Python 3.6.1*** program for the ***Windows 7*** operating system.

To start successfully, you need to download from the official site https://www.python.org/downloads/ version of the program 3.6.1.

In fig. 1.1 shows the main window of this site. Select the menu item ***Downloads***, in the list of versions select ***Python 3.6.1***. Hereinafter, the necessary menu items are underlined in red.

After selecting the version of ***Python 3.6.1***, you must click the ***Download*** button, in the Files window that appears (fig. 1.2) select the operating system used on your computer. Its bit should match your version of ***Windows***. In our example, this is the ***Windows x86 executable installer***. This menu item is also underlined in red.

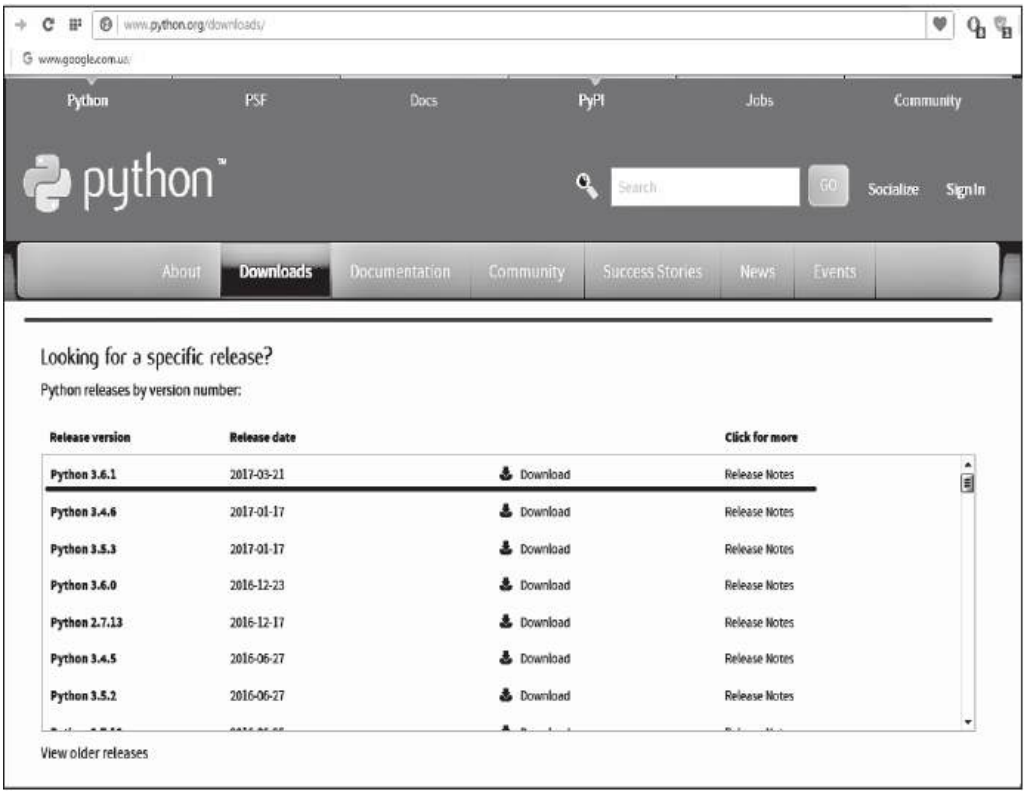


Рис. 1.1. Головне вікно сайту https://www.python.org

Fig. 1.1. Main site window https://www.python.org

Після скачування установника запустіть файл ***python-3.6.1.exe*** і дотримуйтесь інструкцій установлення, як показано далі (рис. 1.3, 1.4).

After downloading the installer, run the ***python-3.6.1.exe*** file and follow the installation instructions as shown below (Fig. 1.3, 1.4).

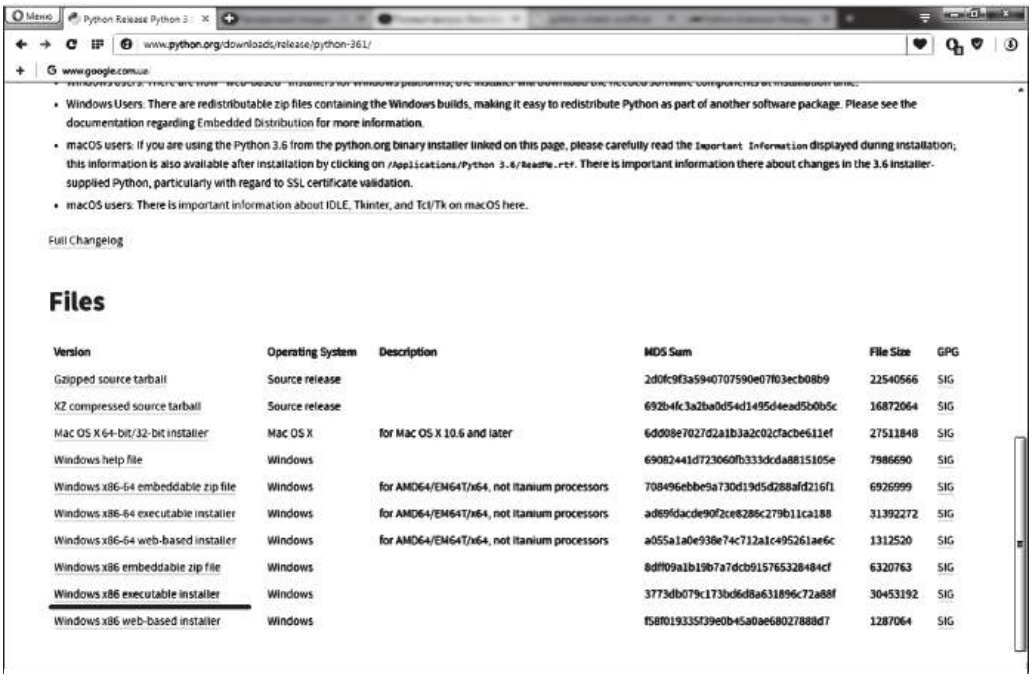


Рис. 1.2. Вікно вибору версії ***Windows***

Fig. 1.2. ***Windows*** version selection window

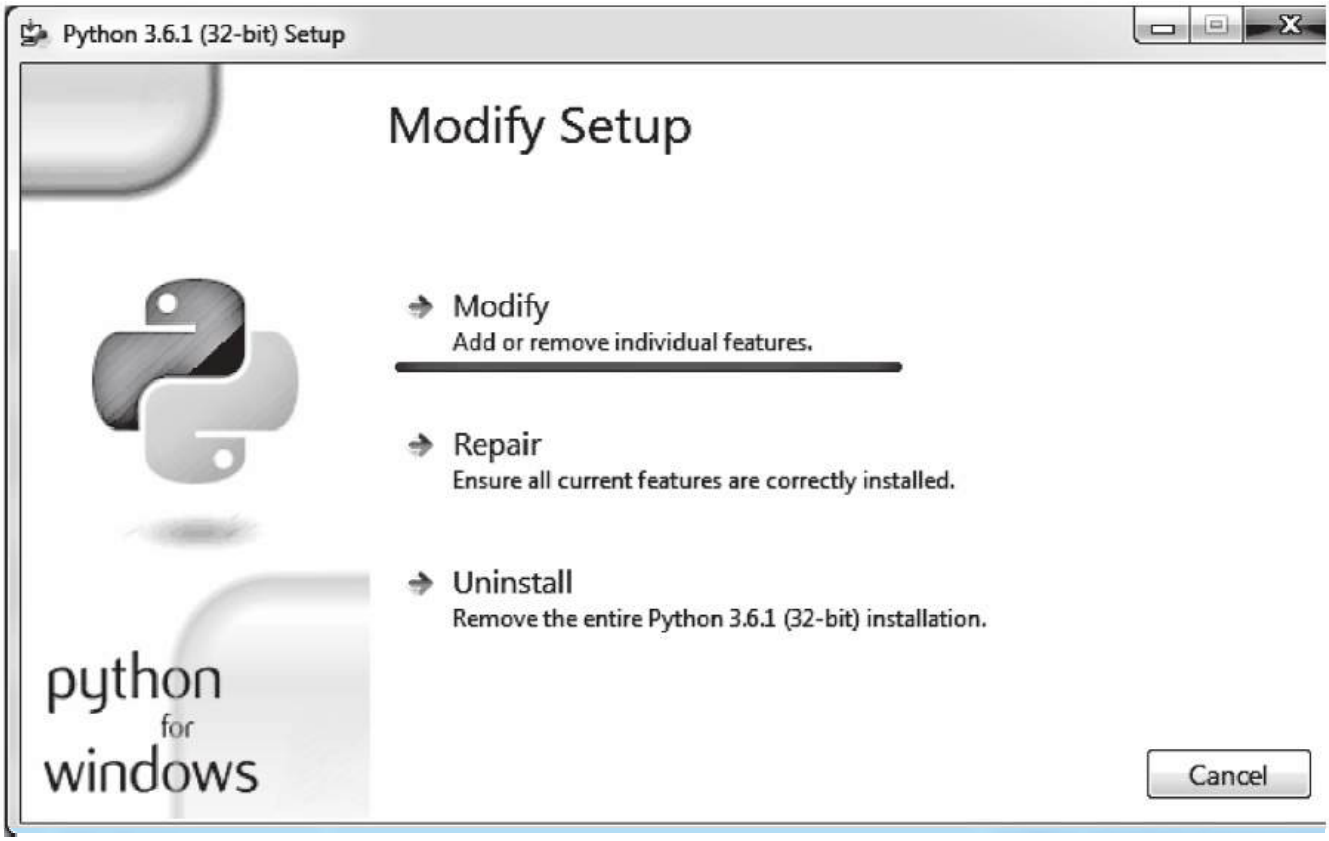


Рис. 1.3. Перше вікно встановлення інтерпретатора ***Python***

Fig. 1.3. First ***Python*** interpreter Install Window

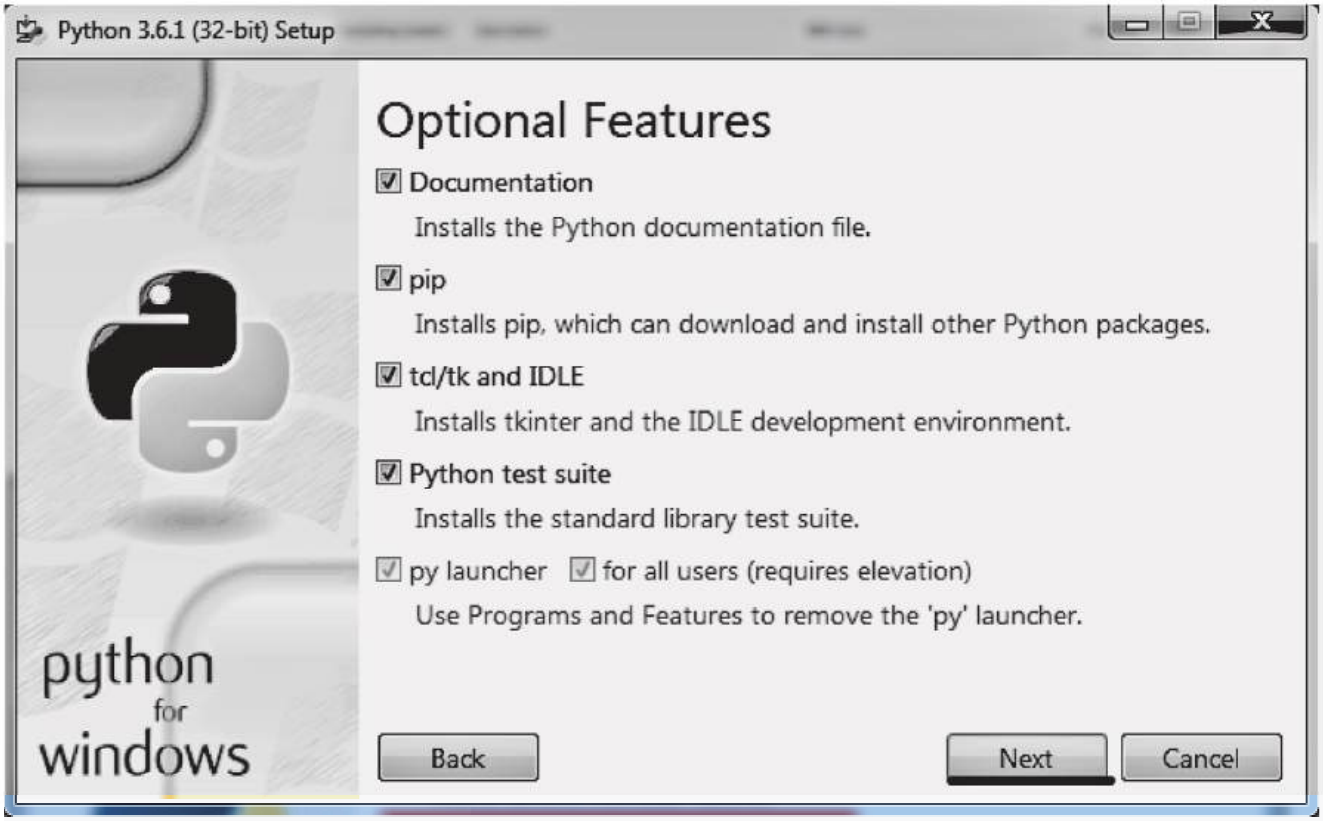


Рис. 1.4. Друге вікно встановлення інтерпретатора ***Python***

Fig. 1.4. Second ***Python*** interpreter installation window

У вікні ***Advanced Options*** виберіть опції, необхідні для роботи з програмою (рис. 1.5).

In the ***Advanced Options*** window, select the options you need when working with the program (Fig. 1.5).

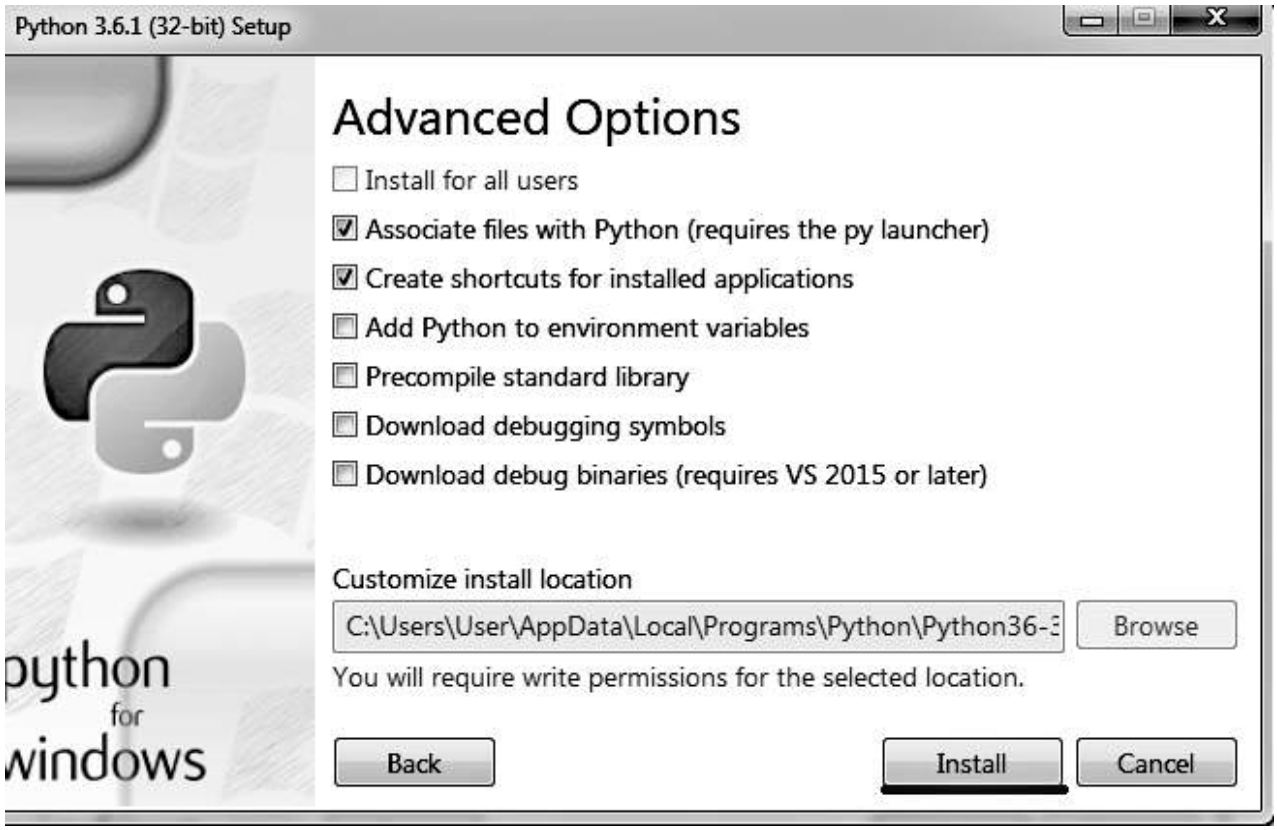


Рис. 1.5. Вікно вибору опцій

Fig. 1.5. Options window

Після встановлення програми запустіть інтерактивне графічне середовище ***IDLE*** (***Python 3.6.1 64-bit***).exe і дочекайтеся появи запрошення для введення команд >>> (рис. 1.6).

After installing the program, launch the interactive graphical environment ***IDLE (Python 3.6.1 64-bit) .exe*** and wait for the prompt to enter the commands >>> (Fig. 1.6).

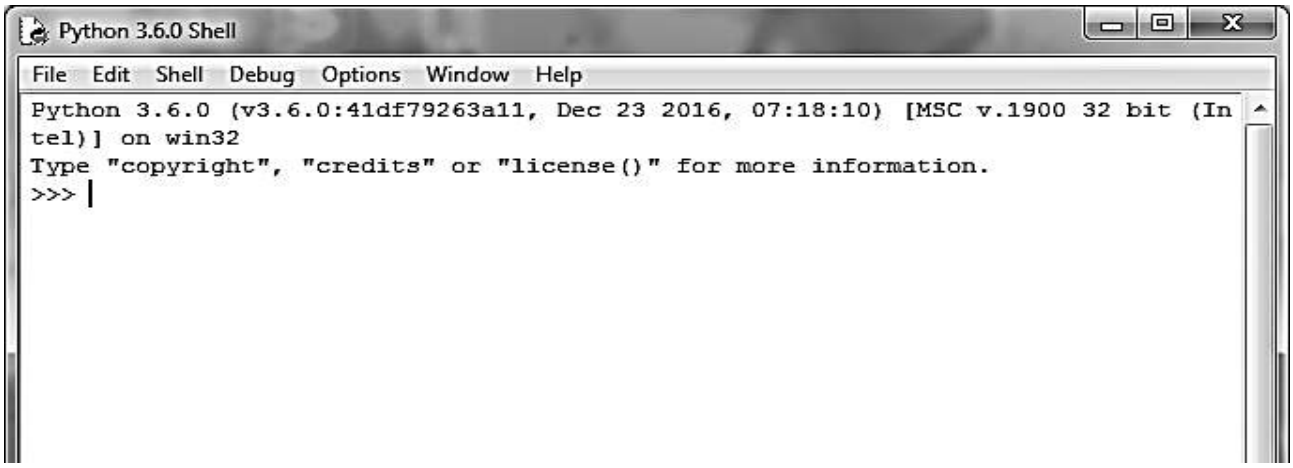


Рис. 1.6. Робоче вікно інтерпретатора

Fig. 1.6. Interpreter window

1.3. Підключення модулів і бібліотек Python

Для в становлення модулів і бібліотек при використанні операційної системи ***Windows*** насамперед необхідно перевірити розрядність системи – x64 чи x86. Для цього потрібно пройти шляхом Панель керування → Система і безпека → Система і в цьому вікні визначити тип системи (рис. 1.7). У цьому ж вікні клацніть лівою кнопкою мишки на закладці «Додаткові параметри системи» (рис. 1.8). Відкриється нове вікно «***Властивості системи***» (рис. 1.9, *а*), у якому у вкладці «додатково» виберіть вкладку «***Змінні середовища***» та активуйте її лівою кнопкою мишки. У вікні «Змінні середовища» (рис. 1.9, *б*) у розділі «***Системні змінні***» упишить у змінному середовищі «***Path***» шлях до системної папки, у якій знаходиться папка Script програми ***Python***, наприклад C:\users\user\appdata\local\programs\python\python36-32\Scripts.

Після цього й установок потрібно скачувати бібліотеки згідно з розрядністю системи.

Бібліотеки для ***Python 3.6*** знаходяться у вільному доступі на сайті http://www.lfd.uci.edu/~gohlke/pythonlib. Потрібну Вам бібліотеку необхідно скачати з цього сайту.

Зазначимо, що в кожному проекті для написання програмного коду необхідно використовувати конкретний набір модулів і бібліотек, а всі інші бібліотеки можна не підключати. Це дає змогу зменшити навантаження на процесор і прискорити роботу програми.

1.3. Connecting Python Modules and Libraries

To install modules and libraries when using the ***Windows*** operating system, it is first necessary to check the system width − ***x64*** or − ***x86***. To do this, follow the path ***Control Panel*** → ***System*** and ***Security*** → ***System*** and in this window determine the type of system (Fig. 1.7). In the same window, click the left mouse button on the "***Advanced system settings***" tab (Fig. 1.8). A new window "***System Properties***" will open (Fig. 1.9, *a*), in which in the "***Advanced***" tab select the tab "***Environment Variables***" and activate it with the left mouse button. In the "***Environment Variables***" window (Fig. 1.9, *b*) in the "***System Variables***" section, enter the path to the system folder in which the ***Python*** Script folder is located in the «***Path***» environment variable. For example, C: \ Users \ User \ AppData \ Local \ Programs \ Python \ Python36-32 \ Scripts.

After this and installations, you need to download libraries according to the system capacity.

Libraries for ***Python 3.6*** are freely available at http://www.lfd.uci.edu/~gohlke/pythonlibs/. You need to download the library from this site.

Note that in each project to write program code, it is necessary to use a specific set of modules and libraries, and all other libraries can be omitted. This allows you to reduce the load on the processor and speed up the program.

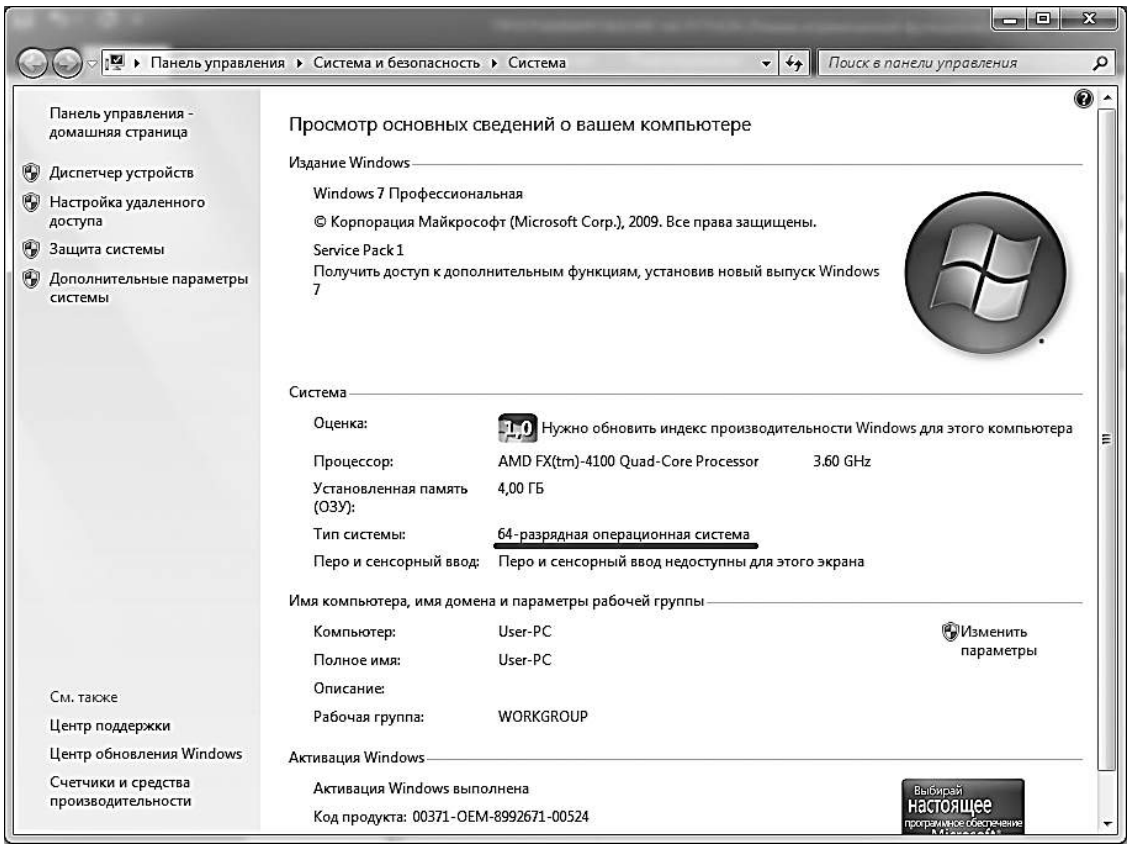


Рис. 1.7. Перевірка розрядності операційної системи

Fig. 1.7. Check digit capacity of the operating system

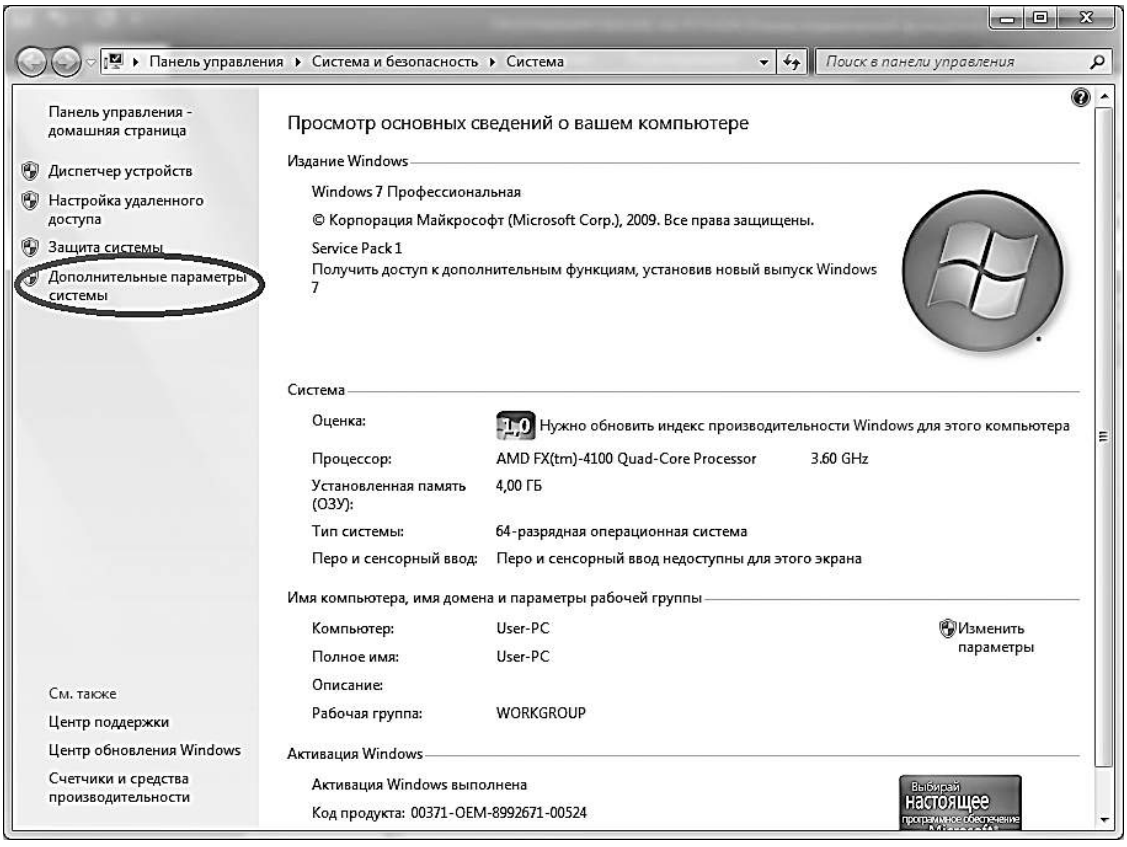
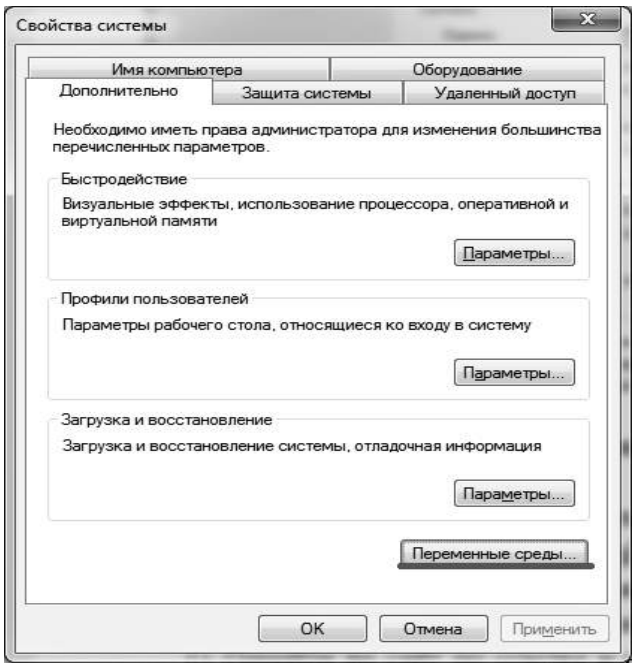


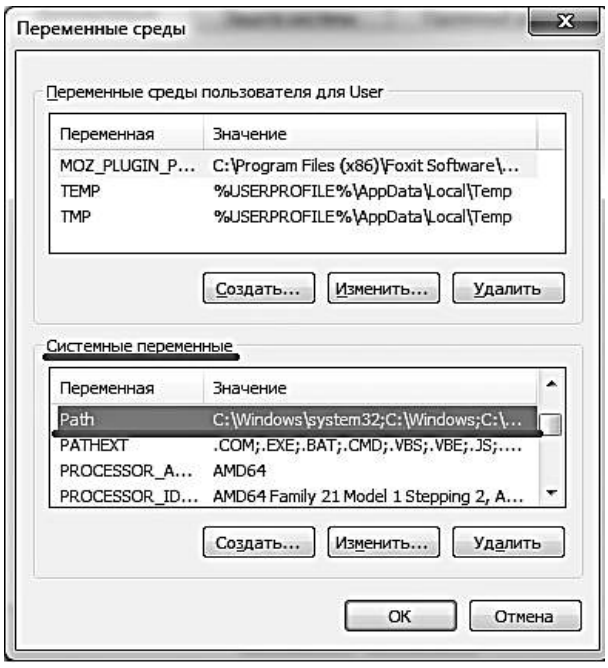
Рис. 1.8. Вибір додаткових параметрів системи

Fig. 1.8. Selection of additional system parameters



а

Рис. 1.9. Установлення системних змінних



б

Fig. 1.9. Selection of additional system parameters

Для прикладу пройдемо увесь шлях установлення бібліотеки ***Pillow***. Для цього необхідно перейти на сайт за посиланням і знайти потрібну бібліотеку. Для зручності можна використовувати комбінацію клавіш ***Ctrl+F*** і у вікні (рис. 1.10), що з'явилося, уписати назву бібліотеки.

For example, let's go all the way of the installation of the **Pillow** library. For this, you must go to the site using the link and find the library you need. For convenience, you can use the ***CTRL + F*** key combination and in the new window (Fig. 1.10) enter the name of the library.

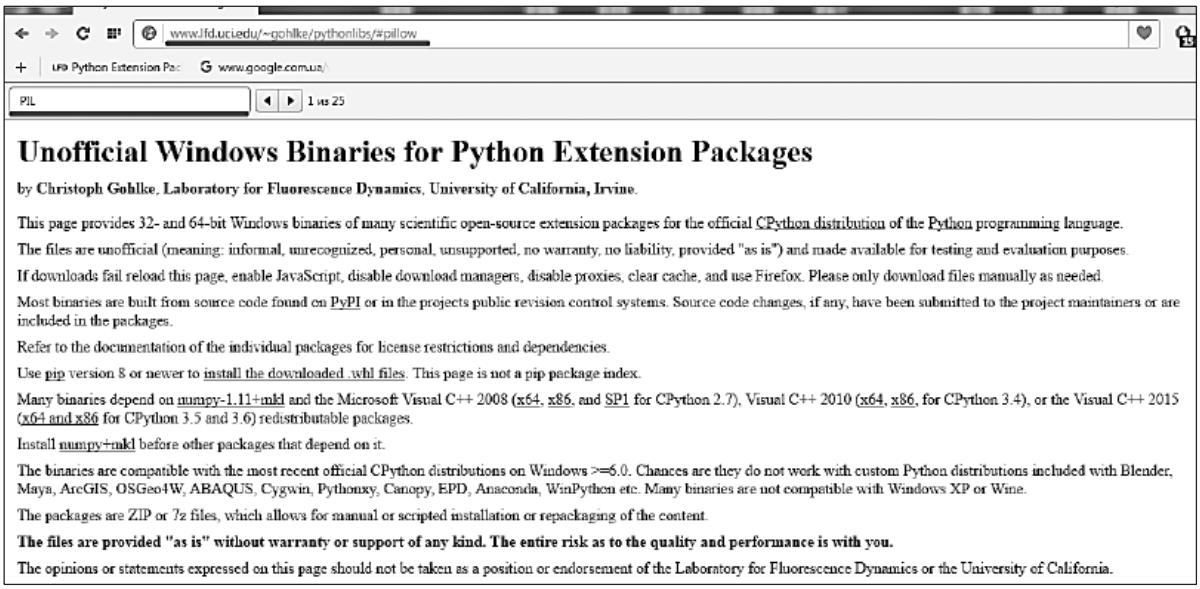


Рис. 1.10. Пошук потрібної бібліотеки на сайті

Fig. 1.10. Search for the desired library on the site

Список потрібних бібліотек розташовується в нижній частині головної сторінки сайту (рис. 1.11).

The list of necessary libraries is located at the bottom of the main page of the site (Fig. 1.11).

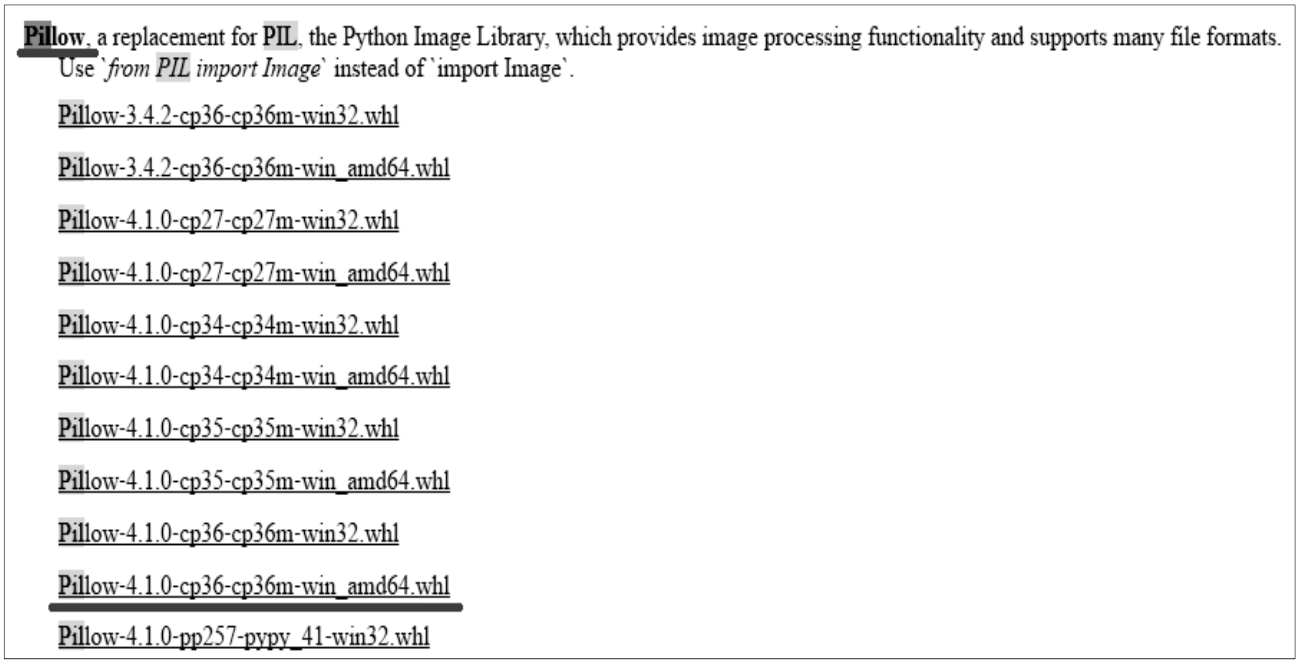


Рис. 1.11. Пошук потрібної бібліотеки

Fig. 1.11. Select the desired library

Скачати вибрану бібліотеку, ураховуючи розрядність операційної системи і раніше встановлену версію для мови ***Python***.

Далі необхідно зайти в системну папку «***Python***», куди раніше встановлювалася програма, наприклад, указавши такий шлях: C:\users\user\Appdata\local\programs\python\python36-32. У командному рядку ввести команду ***«cmd»*** і натиснути кнопку ***Enter*** (рис. 1.12).

У вікні Адміністратора (рис. 1.13) за допомогою вбудованого пакета ***PIP*** додати раніше скачану бібліотеку ***Pillow*** командою «***install***» і підтвердити натисненням кнопки **Enter**.

Download the selected library, considering the bitness of the operating system and the previously installed version of ***Python***.

Next, you need to go to the system folder "Python" where the program was previously installed. For example, specifying this path C:\Users\User\AppData\Local\Programs\Python\Python 36-32. At the command prompt, enter the «***cmd***» command and press the ***Enter*** button (Fig. 1.12).

In the Administrator window (Fig. 1.13), using the built-in ***PIP*** package, add the previously downloaded ***Pillow*** library using the "***install***" command and confirm by pressing the ***Enter*** button.

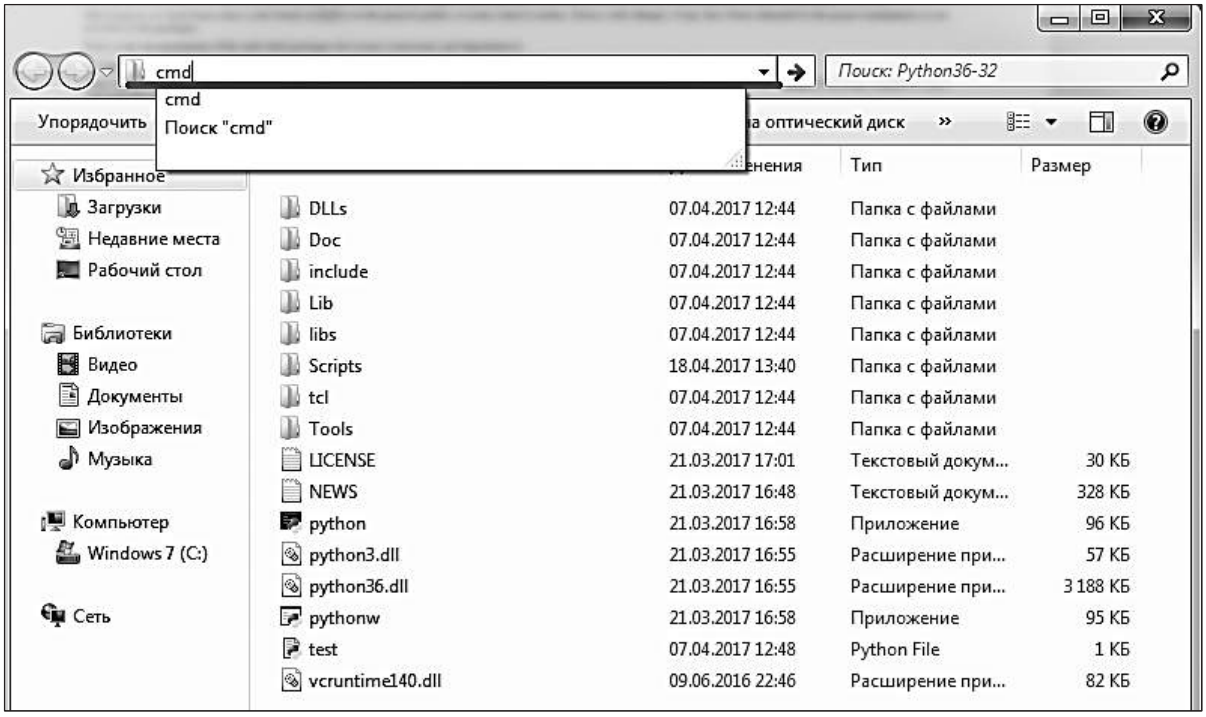


Рис. 1.12. Вікно системної папки ***Python***

Fig. 1.12. ***Python*** system folder window

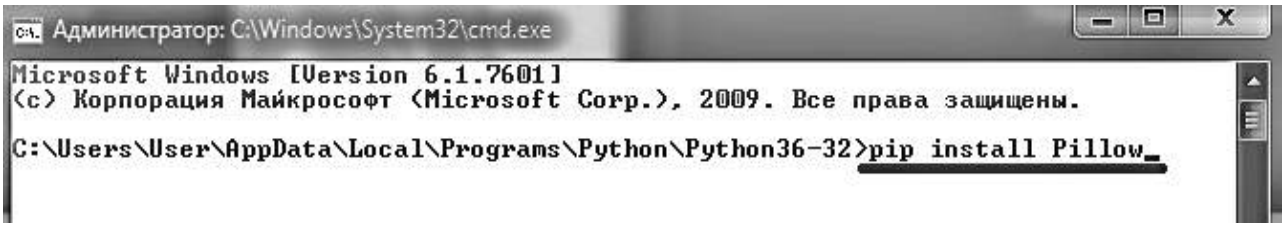


Рис. 1.13. Інсталяція бібліотеки

Fig. 1.13. Library installation

Після цих дій бібліотеку буде встановлено, і її можна буде використовувати в своїй роботі. Інші бібліотеки, наприклад бібліотеку ***Matplotlib***, установлюють аналогічним чином.

При створенні систем технічного зору потрібно використовувати ресурси бібліотеки ***OpenCV***. Це дає змогу істотно спростити програмні коди і збільшити швидкодію системи. Головне вікно сайту https://opencv.org/, на якому розміщена вся необхідна інформація для підключення елементів бібліотеки, показано на рис. 1.14.

After these actions, the library is established, and it can be used in your work. Other libraries, such as the ***Matplotlib*** library, are installed in the same way.

When creating technical vision systems, you need to use the resources of the OpenCV library. This allows you to significantly simplify the program codes and increase system performance. The main window of the site https://opencv.org/, which contains all the necessary information to connect the elements of the library, is shown in Fig. 1.14.

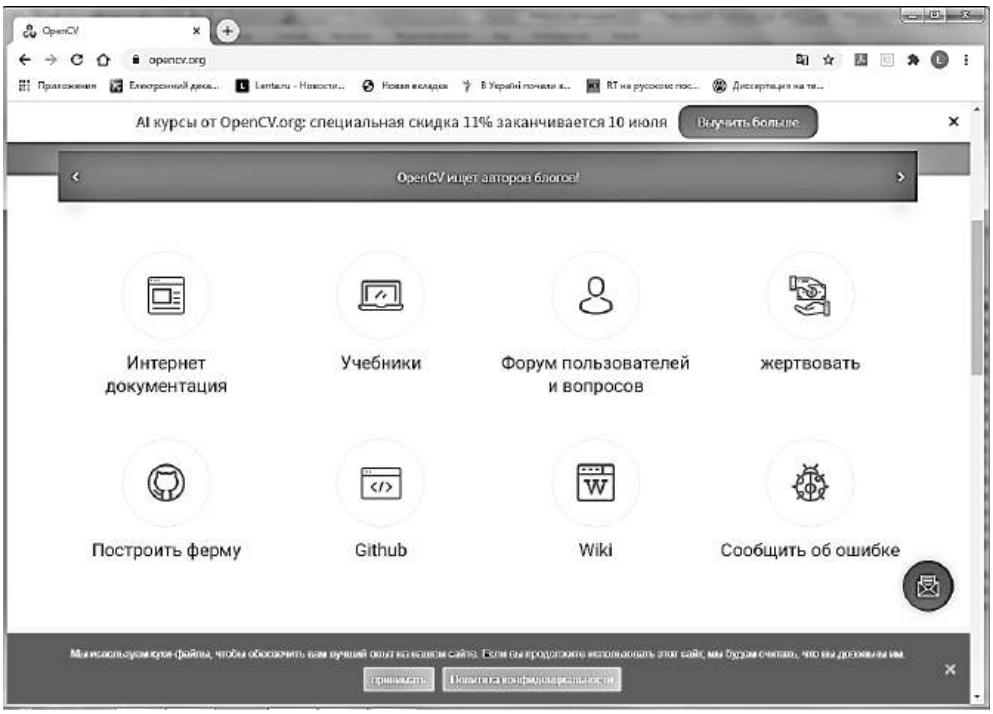


Рис. 1.14. Головне вікно сайту https://opencv.org/

Fig. 1.14. The main window of the site https://opencv.org/

1.4. Підключення бібліотеки OpenCV до Python для ОС Windows

Питання підключення модулів і бібліотек до ***Python*** для ***Windows*** було досить детально описано вище. Нагадаємо лише, що для встановлення модулів і бібліотек при використанні операційної системи ***Windows*** насамперед необхідно перевірити розрядність системи – ***x64*** чи ***x86***, а потім скачати вибрану бібліотеку, ураховуючи розрядність операційної системи і раніше встановлену версію ***Python***. Там же як приклад дуже детально описано процедуру підключення бібліотеки ***Pillow***, що є основним ресурсом ***Python*** для оброблення зображень і відеоінформації.

При встановленні бібліотеки ***OpenCV*** зайдіть у системну папку «***Python***», куди раніше встановлювалася програма, наприклад,

1.4. Connecting OpenCV Library to Python for Windows

The question of connecting modules and libraries to ***Python*** for ***Windows*** was described in some detail above. We only recall that to install modules and libraries when using the ***Windows*** operating system, first you need to check the system width − ***x64*** or − ***x86***, and then download the selected library, taking into account the width of the operating system and the previously installed version of ***Python***. There, as an example, the procedure for connecting the Pillow library, which is the main Python resource for processing images and video information, is described in great detail.

When installing the ***OpenCV*** library, go to the “Python” system folder where the program was previously installed, for example, spec-

указавши такий шлях: C:\users\user\appdata\Local\programs\Python\Python36-32. У командному рядку слід увести команду «***cmd***» і натиснути кнопку ***Enter*** (див. рис. 1.12), і ви перейдете у вікно Адміністратора (рис. 1.15, *а*).

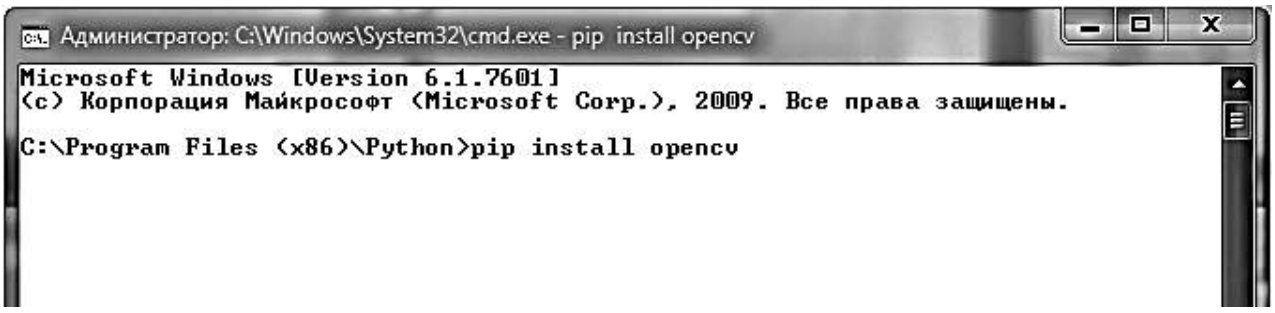
У вікні Адміністратора з допомогою вбудованого пакета ***PIP*** необхідно додати раніше скачану бібліотеку ***OpenCV*** командою «***install***» і підтвердити це натисненням кнопки ***Enter***. Тоді в цьому вікні з'явиться повідомлення про успішну інсталяцію потрібної вам бібліотеки (рис. 1.15, *б*).

Технологію роботи з алгоритмом бібліотеки ***OpenCV*** для оброблення зображень у ***Python*** буде описано далі.

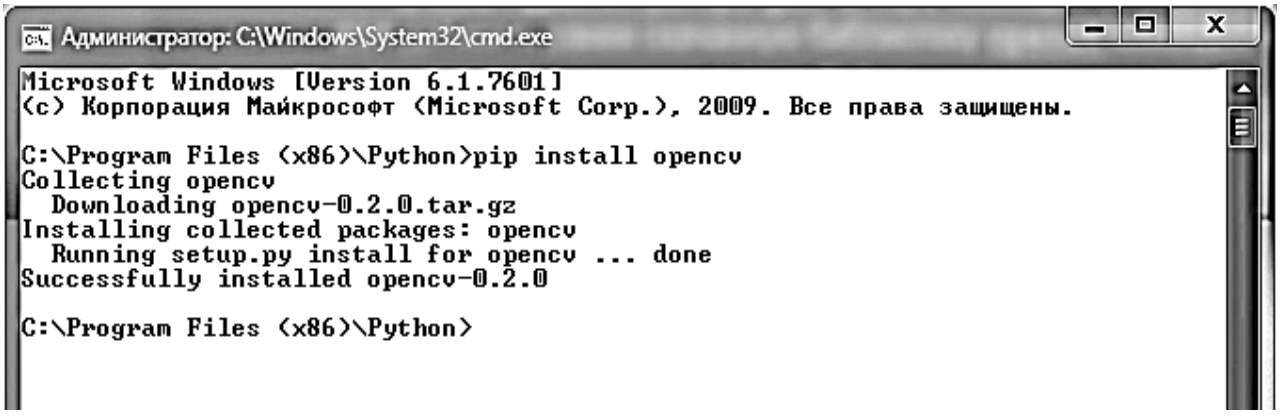
ifying this path C: \Users\User \AppData\ Local\Programs\Python \Python36-32. In the command line, enter the command "***cmd***", press the ***Enter*** button (see Fig. 1.12), and you will go to the Administrator window (Fig. 1.15, *a*).

In the Administrator window you need to add the previously downloaded ***OpenCV*** library using the "***install***" command using the built-in ***PIP*** package and confirm this by pressing the ***Enter*** button. Then a message will appear in this window about the successful installation of the library you need (Fig. 1.15, *b*).

The technology of working with the ***OpenCV*** library algorithms for image processing in ***Python*** will be described below.



а



б

Рис. 1.15. Інсталяція ***OpenCV*** у вікні Адміністратора

Fig. 1.15. Installing ***OpenCV*** in the Administrator window

2. БАЗОВИЙ СИНТАКСИС PYTHON

2.1. Пакети в Python

Пакет (бібліотека) є ієрархічною структурою каталогів файлів, яка визначає єдине прикладне середовище ***Python***, що складається з модулів, які, якщо є бажання, можна згрупувати в підпакети і підпідпакети і т. д.

Модуль дає змогу логічно організувати код ***Python***. Угруповання зв'язаного коду в модуль полегшує розуміння й використання коду. Модуль − це об'єкт ***Python*** з довільно названими атрибутами, які можна зв'язати й послатися на них.

Просто модуль − це файл, що складається з коду ***Python***. Модуль може визначати функції, класи і змінні. Модуль також може містити виконуваний код.

Код ***Python*** для модуля з ім'ям `name` зазвичай знаходиться у файлі з ім'ям `name.py`. Ось приклад простого модуля `hello_mod.py` з однією функцією `print_func()`, який показує рядок зі змінною текстовою частиною `par` у командному вікні:

```
def print_func( par ):
   print("Hello, ", par)
```

Функції детально обговорюватимуться далі. Тепер визначимо концепцію модуля. Можна використовувати будь-який вихідний файл ***Python*** як модуль, виконавши інструкцію `import` в іншому вихідному файлі ***Python***. Імпорт має такий синтаксис:

2. PYTHON BASIC SYNTAX

2.1. Packages in Python

A package (library) is a hierarchical file directory structure that defines a single ***Python*** application environment that consists of modules, which optionally can be grouped in sub-packages and sub-sub-packages, and so on.

A module allows to organize logically ***Python*** code. Grouping related code into a module makes the code easier to understand and use. A module is a ***Python*** object with arbitrarily named attributes that you can bind and reference.

Simply, a module is a file consisting of ***Python*** code. A module can define functions, classes and variables. A module can also include runnable code.

The ***Python*** code for a module named `name` normally resides in a file named `name.py`. Here is an example of a simple module, `hello_mod.py` with one function `print_func()`, which shows string with variable text part `par` in the command window:

Functions in details will be discussed further. Now we are pointed on module concept. You can use any ***Python*** source file as a module by executing an import statement in some other ***Python*** source file. The `import` has the following syntax:

```
import module1[, module2[,... moduleN]
```

Коли інтерпретатор зустрічає оператор імпорту, він завантажує модуль, якщо модуль є в шляху пошуку. Шлях пошуку – це список каталогів, які інтерпретатор шукає перед імпортом модуля. Наприклад, щоб імпортувати модуль `hello_mod.py`, потрібно помістити таку команду зверху скрипту:

When the interpreter encounters an import statement, it uploads the module if the module is present in the search path. A search path is a list of directories that the interpreter searches before importing a module. For example, to import the module `hello_mod.py`, you need to put the following command at the top of the script:

```
# Import module hello_mod
import hello_mod

# Defined function call
hello_mod.print_func("student")
```

Префікс (ім'я модуля + крапка) перед ім'ям функції показує, що функція є атрибутом модуля. Коли наведений вище код виконується, маємо такий результат:

The prefix (module name + dot) before the function name shows that the function is an attribute of the module. When the above code is executed, it produces the following result:

```
Hello, student
```

Модуль завантажується лише один раз незалежно від того, скільки разів він був імпортований.

A module is loaded only once, regardless of the number of times it is imported.

Оператор ***Python*** `from` дає змогу імпортувати певні атрибути з модуля в поточний простір імен. Імпорт `from ...` має такий синтаксис:

Python's `from` statement lets you import specific attributes from a module into the current namespace. The `from...` import has the following syntax:

```
from modname import name1[, name2[, ... nameN]]
```

Наприклад, щоб імпортувати лише функцію `print_func()` з модуля `hello_mod`, використовуйте таку інструкцію:

For example, to import only the function `print_func()` from the module `hello_mod`, use the following statement:

```
# function import
from hello_mod import print_func

# function call
print_func("student")
```

Цей оператор не імпортує весь модуль у поточний простір імен; він просто вводить елемент `print_func()` для доступу в поточному модулі.

2.2. Концепція функцій

Як випливає з наведеного вище опису, функція є згрупованим та іменованим блоком операторів, який може бути виконаний шляхом виклику з іншого файлу сценарію, функції або безпосередньо із запиту ***Python***.

Але спочатку ви маєте знати, як функції зазвичай визначаються в бібліотечних модулях або у ваших власних скриптах, щоб структурувати код ***Python*** і зробити його більш читабельним. Ось прості правила для визначення функції в ***Python***:

- Функціональний блок починається з ключового слова `def`, за яким ідуть ім'я функції (ідентифікатор) і дужки ().
- Будь-які вхідні параметри або аргументи мають бути поміщені в ці круглі дужки. Ви також можете визначити параметри всередині цих дужок.
- Першим оператором функції може бути необов'язковий оператор – рядок документації функції або рядок документа.
- Блок коду всередині функції (іменований набір) починається з двокрапки і має відступ.

Синтаксис визначення функції є таким:

```
def functionname( parameters ):
....."function_docstring"
... function_suite
...[return [expression]]
```

This statement does not import the entire module into the current namespace; it just introduces the item `print_func()` for access in the current module.

2.2. Function concept

As following from above description, function is a grouped and named block of statements, which can be executed by calling from another script file, function or directly from the ***Python*** prompt.

But first you should to know how functions can be normally defined in library modules or in your own scripts in order to structure ***Python*** code and to make it more readable. Here are simple rules to define a function in ***Python***.

- Function block begins with the keyword **`def`** followed by the function name (identifier) and parentheses ( ).
- Any input parameters or arguments should be placed within these parentheses. You can also define parameters inside these parentheses.
- The first statement of a function can be an optional statement – the documentation string of the function or docstring.
- The code block within a function (named suite) starts with a colon : and is indented.

Function definition syntax is following:

Ідентифікатор ***Python*** – це ім'я, що використовується для ідентифікації змінної, функції, класу, модуля або іншого об'єкта.

Ідентифікатор починається з букви від A до Z або від a до z або символу підкреслення (_), за яким ідують нуль або цифри від 0 до 0.9.

У ***Python*** не допускається використання розділових знаків, таких як @, $ і % в ідентифікаторах. ***Python*** є регістрозалежною мовою програмування. Таким чином, Робоча сила і робоча сила – це два різні ідентифікатори в ***Python***.

Список у табл. 2.1 містить ключові слова ***Python***. Це зарезервовані слова, і їх не можна використовувати як імена ідентифікаторів. Усі ключові слова ***Python*** містять лише малі букви. Тепер ви знаєте три з них (виділені жирним шрифтом), деякі інші будуть пояснені в наступних підрозділах, останні можна знайти в документації.

A ***Python*** identifier is a name used to identify a variable, function, class, module or other object.

An identifier starts with a letter A to Z or a to z or an underscore (_) followed by zero or more letters, underscores and digits (0 to 9).

Python does not allow punctuation characters such as @, $, and % within identifiers. Python is a case sensitive programming language. Thus, Manpower and manpower are two different identifiers in ***Python***.

The list in Table 2.1 shows the ***Python*** keywords. These are reserved words and you cannot use them as identifier names. All the ***Python*** keywords contain lowercase letters only. Now you know three of them (selected with bold font), some others will be explained in the next subchapters, the rest – you can find in the documentation.

Таблиця 2.1
Table 2.1

and	else	***import***	raise
assert	except	in	return
break	exec	is	try
class	finally	lambda	while
continue	for	not	with
def	***from***	or	yield
del	global	pass	
elif	if		

Тепер ви можете точно проаналізувати приклад з попереднього підрозділу за допомогою функції `print_func()`, де `print_func` – ідентифікатор функції, `par` – унікальний параметр функції, набір функцій складається з одного оператора, який викликає іншу функцію `print()` з двома фактичними параметрами: текстовий рядок `«Hello»` і значення `par`.

Функція `print()` є однією з вбудованих функцій ***Python***, які завжди є доступними й розпізнаються інтерпретатором ***Python***. Він поміщає ці об'єкти у файл текстового потоку, який за замовчуванням є `sys.stdout` (наприклад, командним вікном). Повний список вбудованих функцій наведено в табл. 2.2. Деякі з них буде розглянуто далі, останні ви можете знайти в документації.

Now you are able to analyze precisely the example from the previous subchapter with function `print_func()`, where `print_func` – is a function identifier, `par` – unique function parameter, suit of the function consists of one statement, which is calling another function `print()` with two actual parameters: text string `"Hello,"` and value of `par`.

The `print()` function is one of the built-in ***Python*** functions, which are always available and recognized by ***Python*** interpreter. It puts given objects to the text stream file, which is `sys.stdout` (e.g. command window) by default. The full list of the built-in functions is represented in Table 2.2. Some of them will be considered further, the rest – you can find in the documentation.

Таблиця 2.2

Table 2.2

abs()	delattr()	hash()	memoryview()	set()
all()	dict()	help()	min()	setattr()
any()	dir()	hex()	next()	slice()
ascii()	divmod()	id()	object()	sorted()
bin()	enumerate()	input()	oct()	staticmethod()
bool()	eval()	int()	open()	str()
breakpoint()	exec()	isinstance()	ord()	sum()
bytearray()	filter()	issubclass()	pow()	super()
bytes()	float()	iter()	print()	tuple()
callable()	format()	len()	property()	type()
chr()	frozenset()	list()	range()	vars()
classmethod()	getattr()	locals()	repr()	zip()
compile()	globals()	map()	reversed()	__import__()
complex()	hasattr()	max()	round()	

Параметри зазвичай розділяються (кома), і вам потрібно повідомити їх при виклику функції відповідно до синтаксису:

```
functionname( arguments )
```

Ви можете викликати функцію, використовуючи такі типи аргументів (фактичні параметри):

- обов'язкові аргументи;
- ключові аргументи;
- аргументи за замовчуванням;
- аргументи змінної довжини.

Обов'язкові аргументи – це аргументи, передані функції в правильному позиційному порядку. Тут кількість аргументів у виклику функції повинна точно відповідати визначенню функції.

Наприклад, щоб викликати функцію `printme()`, потрібно передати один аргумент, інакше він видасть синтаксичну помилку таким чином:

Parameters are normally separated by (coma) and you need to inform them while function calling according to the syntax:

You can call a function using the following types of arguments (actual parameters):

- required arguments;
- keyword arguments;
- default arguments;
- variable-length arguments.

Required arguments are the arguments passed to a function in correct positional order. Here, the number of arguments in the function call should match exactly with the function definition.

For example, to call the function `printme()`, you definitely need to pass one argument, otherwise it gives a syntax error as follows:

```
# Function definition
def printme( str ):
   "This prints a passed string into this function"
   print( str )

# Function call
printme()
```

Коли наведений вище код виконується, маємо такий результат:

When the above code is executed, it produces the following result:

```
Traceback (most recent call last):
   File "test.py", line 11, in <module>
      printme();
TypeError: printme() takes exactly 1 argument (0 given)
```

Аргументи ключових слів зв'язані з викликами функцій. Коли ви використовуєте аргументи ключового слова у виклику функції, сторона, що викликає, ідентифікує аргументи за іменем параметра.

Keyword arguments are related to the function calls. When you use keyword arguments in a function call, the caller identifies the arguments by the parameter name.

Це дає змогу пропускати аргументи або розміщувати їх не по порядку, оскільки інтерпретатор ***Python*** може використовувати надані ключові слова для зіставлення значень з параметрами.

Наведемо приклад, коли порядок параметрів не має значення:

This allows you to skip arguments or place them out of order because the ***Python*** interpreter is able to use the keywords provided to match the values with parameters.

In the following example order of parameters does not matter:

```
# Function definition
def printinfo( name, age ):
   "This prints a passed info into this function"
   print( "Name: ", name )
   print( "Age ", age )

# Function call
printinfo( age=20, name="Peet" )
```

Коли наведений вище код виконується, маємо такий результат:

When the above code is executed, it produces the following result:

```
Name:  Peet
Age 20
```

Аргумент за замовчуванням – це аргумент, що набуває значення за замовчуванням, яке має бути встановлене у визначенні функції, якщо значення не вказано у виклику функції для цього аргументу.

Наведемо нижче приклад дає уявлення про аргументи за замовчуванням, друкується вік за замовчуванням, якщо його не було передано:

A default argument is an argument that assumes a default value, which should be set in function definition, if a value is not provided in the function call for that argument.

The following example gives an idea on default arguments, it prints default age if it is not passed:

```
# Function definition is here
def printinfo( name, age = 35 ):
   "This prints a passed info into this function"
   print( "Name: ", name )
   print( "Age ", age )

# Now you can call printinfo function
printinfo( age=20, name="Peet" )
printinfo( name="Mary" )
```

Коли наведений вище код виконується, маємо такий результат:

When the above code is executed, it produces the following result:

```
Name:  Peet
Age  20
Name:  Mary
Age  35
 50
```

Оператор `return [вираз]` виходить з функції, необов'язково передаючи вираз стороні, що викликає. Це можна пропустити, як в наведених вище прикладах, коли повертане значення відсутнє. Такі функції також називаються void-функціями або процедурами.

Ви можете повернути значення з функції, як у такому прикладі:

The statement `return [expression]` exits a function, optionally passing back an expression to the caller. It can be missed as in the above examples, when returning value are absent. Such a functions are also named as void-functions or procedures.

You can return a value from a function as in following example:

```
# Function definition
def sum( arg1, arg2 ):
   # Add both the parameters and return them."
   total = arg1 + arg2
   return total
# Function call
total = sum( 10, 20 )
print( "Sum of 10 and 20 : ", total )
```

Коли наведений вище код виконується, маємо такий результат:

When the above code is executed, it produces the following result:

```
Sum of 10 and 20: 30
```

2.3. Дані, операції й алгоритми

Параметри, а також повертані значення можуть мати дані різних типів даних. ***Python*** має дані різних стандартних типів, які використовуються для визначення можливих операцій над ними і методу зберігання для кожного з них. ***Python*** має такі дані стандартних типів:

- число;
- рядок;
- список;
- кортеж.

2.3. Data, operations and algorithms

Parameters as well as returning values can be of different datatypes. ***Python*** has various standard data types that are used to define the operations possible on them and the storage method for each of them. ***Python*** has following standard data types:

- numbers;
- string;
- list;
- tuple.

Змінні ***Python*** не потребують явного оголошення для резервування місця в пам'яті. Оголошення відбувається автоматично, коли ви присвоюєте значення змінній в певній області: глобальна (файл) або локальна (клас, функція, блок). Знак рівності «=» використовується для створення нового об'єкта з ідентифікатором, указаним справа, на основі значень, указаних або розрахованих зліва, наприклад:

Python variables do not need explicit declaration to reserve memory space. The declaration happens automatically when you assign a value to a variable in a certain scope: global (file) or local (class, function, block). The equal sign = is used to create a new object with identifier stated on the right side, based on values stated or calculated on the left side. For example:

```
counter = 100          # An integer object
miles   = 1000.0       # A floating point object

name    = "John"       # A string
```

Python дає змогу призначати одне значення декільком змінним одночасно, наприклад:

Python allows you to assign a single value to several variables simultaneously. For example:

```
a = b = c = 1
```

Дані числових типів зберігають числові значення. Числові об'єкти створюються при призначенні ним значення, наприклад:

Number data types store numeric values. Number objects are created when you assign a value to them. For example:

```
var1 = 1
var2 = 10
```

Python підтримує дані чотирьох різних числових типів:

- int (цілі числа зі знаком);
- long (довгі цілі числа, які також можуть бути подані у вісімковому та шістнадцятковому вигляді);
- float (реальні значення з плаваючою крапкою);
- складні (комплексні числа).

У табл. 2.3 наведено деякі приклади числових значень.

Python supports four different numerical types

- int (signed integers);
- long (long integers, they can also be represented in octal and hexadecimal);
- float (floating point real values);
- complex (complex numbers).

The Table 2.3 represents some examples of number values.

Таблиця 2.3
Table 2.3

int	long	float	complex
10	51924361L	0.0	3.14j
100	-0x19323L	15.20	45.j
-786	0122L	-21.9	9.322e-36j
080	0xDEFABCECBDAECBFBAEl	32.3+e18	.876j
-0490	535633629843L	-90.	-.6545+0J
-0x260	-052318172735L	-32.54e100	3e+26J
0x69	-4721885298529L	70.2-E12	4.53e-7j

Арифметичні оператори зазвичай використовуються для створення виразів із числових змінних. Наведемо приклад різніх операцій у командному режимі з коментарями:

Arithmetic operators are normally used to create expression from number variables. Following example demonstrates different operations in command mode with comments:

```
>>> a = 55; b = 3
>>> a + b #Addition of two operands
58
>>> a - b # Subtraction of two operands
52
>>> a * b # Multiplication of two operands
165
>>> a / b # Division of two operands
18.333333333333332
>>> a % b # Getting remainder after division
1
>>> a ** b # Raising a to power b
166375
>>> a // b  #Floor division (in straight mathematic sense)
18
```

Булів тип був доданий у ***Python 2.3***. Було додано дві нові константи `True` і `False` (1 і 0). Об'єкт типу для цього нового типу називається `bool` і є підкласом `int`.

A Boolean type was added to ***Python 2.3***. Two new constants were added `True` and `False` (1 and 0). The type object for this new type is named `bool` and is a subclass of `int`.

Приклад:

Example:

```
test = True             # True
test = 0.0              # False
test = -1.0       # True
test = None             # False
```

test = 'Easy string'# True

Реляційні оператори дають змогу порівнювати змінні числового типу даних і дають логічний результат. Наведемо приклад різних операцій у командному режимі з коментарями:

Relational operators allow to compare variables of numeric datatype and gives Boolean result. Following example demonstrates different operations in command mode with comments:

```
>>> x = 55.5; y = 2
>>> x == y # Equal
False
>>> x != y # Not equal (similar to <>)
True
>>> x > y  # Greater than
True
>>> x < y  # Less than
False
>>> x <= y # Less or equal
False
>>> x >= y # Greater or equal
True
```

Логічні оператори зазвичай використовуються для створення логічного виразу:

Logical operators are normally used to create Boolean expression.

```
>>> x = 55.5; y = 2
>>> (x > y) and (x == 0) # True, only if both operands are
True
False
>>> (x < y) or (y%2 == 0) # False, only if both operands are
False
True
>>> not(x//5 == y) # Reverse Boolean value
True
```

Прийняття рішень зазвичай грунтується на булевих виразах і дає алгоритми розгалуження дво-

Decision-making usually based upon Boolean expressions and gives branch algorithms with two

ма способами, залежно від того, чи є у нас `True` або `False` унаслідок виразу:

```
if expression1:
   statement(s)
[elif expression2:
   statement(s)
elif expression3:
   statement(s)]
[else:
   statement(s)]
```

Оператор `else` є необов'язковим, і після `if` може бути не більше одного оператора `else`. Подібно до іншого, оператор `elif` є необов'язковим. Проте, на відміну від інших, для яких може бути не більш одного оператора, після оператора `if` може бути довільна кількість операторів `elif`.

Якщо логічний вираз має значення `True`, то виконується блок операторів усередині оператора `if`. Оператор `else` містить блок коду, який виконується, якщо умовний вираз в операторі `if` набуває значення 0 або `False`.

Оператор `elif` дає змогу перевіряти декілька виразів на `True` і виконувати блок коду, як тільки одна з умов оцінюється як `True`.

Приклад:

ways, depending on whether we have `True` or `False` as the result of the expression:

The `else` statement is an optional statement and there could be at most only one `else` statement following `if`. Similar to the `else`, the `elif` statement is optional. However, unlike else, for which there can be at most one statement, there can be an arbitrary number of `elif` statements following an `if`.

If the Boolean expression evaluates to `True`, then the block of statement(s) inside the `if` statement is executed. An `else` statement contains the block of code that executes if the conditional expression in the `if` statement resolves to `0` or a `False` value.

The `elif` statement allows you to check multiple expressions for `True` and execute a block of code as soon as one of the conditions evaluates to `True`.

Example:

```
var = 100
if var == 200:
   print( "1 - Got a true expression value" )
elif var == 150:
   print( "2 - Got a true expression value" )
elif var == 100:
   print( "3 - Got a true expression value" )
else:
```

```
   print( "4 - Got a false expression value" )
print( var )
print( "Good bye!" )
```

Коли наведений вище код виконується, маємо такий результат:

When the above code is executed, it produces the following result:

```
3 - Got a true expression value
100
Good bye!
```

Рядки є незмінними в ***Python*** і ідентифікуються як неперервний набір символів, поданих у лапках. ***Python*** допускає пари одинарних або подвійних лапок. Підмножини рядків можуть бути отримані з використанням оператора зрізу `([ ]` і `[ : ]`) з індексами, що починаються з 0 на початку рядка і проходять від -1 до кінця. Знак плюс `(+)` – це оператор об'єднання рядків, а зірочка `(*)` – оператор повторення.

Strings are immutable in **Python** and are identified as a contiguous set of characters represented in the quotation marks. ***Python*** allows either pairs of single or double quotes. Subsets of strings can be taken using the slice operator `([ ] and [:])` with indexes starting at `0` in the beginning of the string and working their way from `-1` at the end. The plus `(+)` sign is the string concatenation operator and the asterisk `(*)` is the repetition operator.

Приклад:

Example:

```
str = 'Hello World!'
print(str)        # Prints complete string
print(str[0])     # Prints first character of the string
print(str[2:5])   # Prints characters starting from 3rd to 5th
print(str[2:])    # Prints string starting from 3rd character
print(str * 2)    # Prints string two times
print(str + "TEST") # Prints concatenated string
```

Список – це найбільш універсальний тип даних, доступний у ***Python***, який можна записати у вигляді списку значень (елементів), розділених комами, у квадратних дужках. Важливим моментом є те, що елементи в списку не обов'язково мають бути одного типу.

List is a most versatile datatype available in **Python** which can be written as a list of comma-separated values (items) between square brackets. Important thing about a list is that items in a list need not be of the same type.

Створити список так само просто, як поставити різні значення через кому в квадратних дужках.

Creating a list is as simple as putting different comma-separated values between square brackets.

Приклад:

```
list1 = ['physics', 'chemistry', 1997, 2000];
list2 = [1, 2, 3, 4, 5 ];
list3 = ["a", "b", "c", "d"]
```

Example:

Подібно до індексів рядків індекси списків починаються з 0, і списки можуть бути нарізані, об'єднані і т. д. Щоб отримати доступ до значень у списках, використовуйте квадратні дужки для нарізки разом з індексом або індексами, щоб одержати значення, доступні за цим індексом.

Similar to string indices, list indices start at 0, and lists can be sliced, concatenated and so on. To access values in lists, use the square brackets for slicing along with the index or indices to obtain value available at that index.

Приклад:

```
list1 = ['physics', 'chemistry', 1997, 2000];
list2 = [1, 2, 3, 4, 5, 6, 7 ];
print("list1[0]: ", list1[0])
print("list2[1:5]: ", list2[1:5])
```

Example:

Коли наведений вище код виконується, маємо такий результат:

When the above code is executed, it produces the following result:

```
list1[0]:  physics
list2[1:5]:  [2, 3, 4, 5]
```

Ви можете відновити один або декілька елементів списків, задавши зріз у лівій частині оператора присвоєння, і додавати елементи до списку за допомогою методу `append()`.

You can update single or multiple elements of lists by giving the slice on the left-hand side of the assignment operator, and you can add to elements in a list with the append() method.

Приклад:

```
st = ['physics', 'chemistry', 1997, 2000];
print( "Value available at index 2 : " )
print( list[2] )
list[2] = 2001
print( "New value available at index 2 : " )
print( list[2] )
```

Example:

Коли приведений вище код виконується, маємо такий результат:

When the above code is executed, it produces the following result:

```
Value available at index 2 :
1997
New value available at index 2 :
2001
```

Списки реагують на операторів + і * так само, як рядки; вони також означають конкатенацію і повторення, за винятком того, що результатом є новий список, а не рядок.

У загальному випадку оператори виконуються послідовно: перший оператор у функції виконується першим, потім другий і т. д. Може виникнути ситуація, особливо якщо ви працюєте з послідовностями, такими як рядки, списки або кортежі, коли вам потрібно виконати блок коду кілька разів.

Оператор `while` виконує багато разів (повторює) цільовий оператор доти, доки певна умова є правильною. Синтаксис циклу `while` у мові програмування ***Python***:

```
while condition:
   statement(s)
```

Умова в заголовку (перший рядок) може бути будь-яким виразом, і будь-яке ненульове значення інтерпретуватиметься як `Істина`, нуль – як `Хибність`. Цикл повторюється доти, доки умова виконується. Коли умова стає помилковою, керування програмою переходить на лінію, що є наступною безпосередньо за циклом.

У ***Python*** всі оператори з відступом на однакову кількість символьних просторів після програмної конструкції вважаються частиною єдиного блока коду (масті).

Тут ключовим моментом циклу while є те, що цикл може ніколи не виконуватися. Коли умову перевірено і результат є помилковим,

Lists respond to the + and * operators much like strings; they mean concatenation and repetition here too, except that the result is a new list, not a string.

In general, statements are executed sequentially: the first statement in a function is executed first, followed by the second, and so on. There may be a situation, especially if you work with sequences like strings, lists or tuples, when you need to execute a block of code several number of times.

A `while` loop statement executes repeatedly (iterates) a target statement as long as the given condition is true. The syntax of `while` loop in ***Python*** programming language is:

The condition in the header (first line) may be any expression, and any non-zero value will interpreted as `True`, zero – as `False`. The loop iterates while the condition is true. When the condition becomes false, program control passes to the line immediately following the loop.

In ***Python***, all the statements indented by the same number of character spaces after a programming construct are considered to be part of a single block of code (suit).

Here, key point of the `while` loop is that the loop might not ever run. When the condition is tested and the result is false, the loop

тіло циклу буде пропущено, і буде виконаний перший оператор після циклу `while`. Якщо умова завжди є правильною, то цикл буде нескінченним.

Приклад:

```
precession = 0.05; number = 5
while (number > precession):
   print(number)
   number /= 2      #number = number/2
```

Коли наведений вище код виконується, маємо такий результат:

```
5
2.5
1.25
0.625
0.3125
0.15625
0.078125
```

Запит, що складається з друку і привласнення з операторами ділення, виконується багато разів, поки число не стане більше прецесії. На кожній ітерації відображується поточне значення числа, а потім ділиться на 2.

Оператор `for` має можливість перебирати елементи будь-якої послідовності, такої як список або рядок. Синтаксис циклу `for` у мові програмування ***Python***:

```
for iterating_var in sequence:
   statements(s)
```

Якщо послідовність містить список виразів, вона оцінюється першою. Потім перший елемент у послідовності присвоюється ітеровній змінній `iterating_var`.

Потім виконується блок операторів. Кожен елемент у списку присвоюється `iterating_var`, і блок операторів виконується доти,

body will be skipped and the first statement after the `while` loop will be executed. If condition is always true, a loop will be infinite.

Example:

When the above code is executed, it produces the following result:

The suit here, consisting of the print and assignment with division statements, is executed repeatedly until number is no longer greater than precession. On each iteration, the current value of number is displayed and then divided by 2.

The `for` loop statement has the ability to iterate over the items of any sequence, such as a list or a string. The syntax of `for` loop in ***Python*** programming language is:

If a sequence contains an expression list, it is evaluated first. Then, the first item in the sequence is assigned to the iterating variable `iterating_var`.

Next, the statement(s) block is executed. Each item in the list is assigned to `iterating_var`, and the statement(s) block is executed

доки не буде вичерпано всю послідовність. Приклад:

until the entire sequence is exhausted. Example:

```
for letter in 'OpenCV':     # First Example
    print( 'Current Letter :', letter )
colors = ['red', 'green', 'blue']
for color in colors:        # Second Example
    print( 'Current color :', color )
```

Коли наведений вище код виконується, маємо такий результат:

When the above code is executed, it produces the following result:

```
Current Letter : O
Current Letter : p
Current Letter : e
Current Letter : n
Current Letter : C
Current Letter : V
Current color : red
Current color : green
Current color : blue
```

Альтернативний спосіб перебору кожного елемента – зсув індексу в самій послідовності. Наведемо простий приклад:

An alternative way of iterating through each item is by index offset into the sequence itself. Following is a simple example:

```
colors = ['red', 'green', 'blue']
for index in range(len(colors)):
   print('Current color :', colors [index])
```

Коли наведений вище код виконується, маємо такий результат:

When the above code is executed, it produces the following result:

```
Current color : red
Current color : green
Current color : blue
```

Тут ми скористалися вбудованою функцією `len()`, яка дає загальну кількість елементів у кортежі, а також вбудованою функцією `range()`, що дає фактичну послідовність для повторення.

Here, we took the assistance of the `len()` built-in function, which provides the total number of elements in the tuple as well as the `range()` built-in function to give us the actual sequence to iterate over.

Оператор `break` можна використовувати в циклах `while` і `for` для завершення ітерацій циклу.

The `break` statement can be used in both `while` and `for` loops to terminate loop iterations.

Приклад:

```
while True:
    c = input("Enter any symbol (e - for EXIT): ")
    if c == 'e':
        break
```

Тут `input()` – вбудована функція, яка повертає рядок, отриманий з вікна команди після даного текстового запиту. Коли наведений вище код виконується, маємо такий результат:

```
Enter any symbol (e - for EXIT): r
Enter any symbol (e - for EXIT): 0
Enter any symbol (e - for EXIT):e
```

Якщо ви використовуєте вкладені цикли, оператор `break` зупиняє виконання самого внутрішнього циклу й починає виконання наступного рядка коду після блока.

Приклад:

```
for x in range(4):
   for y in range(4):
      if y == x:
         break
      print(x,' ',y)
```

Коли наведений вище код виконується, маємо такий результат:

```
1   0
2   0
2   1
3   0
3   1
3   2
```

Оператор `continue` відхиляє оператори, що залишилися, у поточній ітерації циклу і переміщає елемент керування назад на початок циклу.

Оператор `continue` може використовуватися в циклах `while` і `for`, наприклад:

Example:

```
while True:
    c = input("Enter any symbol (e - for EXIT): ")
    if c == 'e':
        break
```

Here `input()` is a built-in function, which returns string getting from command window after given text prompt. When the above code is executed, it produces the following result:

```
Enter any symbol (e - for EXIT): r
Enter any symbol (e - for EXIT): 0
Enter any symbol (e - for EXIT):e
```

If you are using nested loops, the `break` statement stops the execution of the innermost loop and start executing the next line of code after the block.

Example:

```
for x in range(4):
   for y in range(4):
      if y == x:
         break
      print(x,' ',y)
```

When the above code is executed, it produces the following result:

```
1   0
2   0
2   1
3   0
3   1
3   2
```

The `continue` statement rejects all the remaining statements in the current iteration of the loop and moves the control back to the top of the loop.

The `continue` statement can be used in both `while` and `for` loops. For example:

```
for letter in 'Python':     # First Example
   if letter == 'h':
      continue
   print( 'Current Letter :', letter)
var = 10                    # Second Example
while var > 0:
   var = var -1
   if var == 5:
      continue
   print( 'Current variable value :',var)
print("Good bye!")
```

Коли наведений вище код виконується, маємо такий результат:

When the above code is executed, it produces the following result:

```
Current Letter : P
Current Letter : y
Current Letter : t
Current Letter : o
Current Letter : n
Current variable value : 9
Current variable value : 8
Current variable value : 7
Current variable value : 6
Current variable value : 4
Current variable value : 3
Current variable value : 2
Current variable value : 1
Current variable value : 0
Good bye!
```

Оператор `pass` у ***Python*** використовується тоді, коли оператор потрібний синтаксично, але ви не хочете, щоб виконувалася яка-небудь команда або код.

Оператор `pass` є нульовою операцією; нічого не відбувається, коли він виконується. Ця передача також є корисною в тих місцях, де ваш код зрештою йтиме, але його ще не написано, наприклад, у функціях, що відповідають налагодженню коду.

Кортеж − це послідовність незмінних об'єктів ***Python***. Кортежі –

The `pass` statement in ***Python*** is used when a statement is required syntactically but you do not want any command or code to execute.

The `pass` statement is a null operation; nothing happens when it executes. The `pass` is also useful in places where your code will eventually go, but has not been written yet, for instance, in functions suits for code debugging.

A tuple is a sequence of immutable ***Python*** objects. Tuples are

це такі ж послідовності, як і списки. Відмінності між кортежами і списками полягають в тому, що кортежі не можуть бути змінені на відміну від списків, а кортежі використовують круглі дужки, тоді як списки використовують квадратні дужки.

sequences, just like lists. The differences between tuples and lists are, the tuples cannot be changed unlike lists and tuples use parentheses, whereas lists use square brackets.

Створити кортеж так само просто, як поставити різні значення через кому. Якщо бажаєте, можете також поставити ці значення через кому в дужках, наприклад:

Creating a tuple is as simple as putting different comma-separated values. Optionally you can put these comma-separated values between parentheses also. For example:

```
tup1 = ('physics', 'chemistry', 1997, 2000)
tup2 = (1, 2, 3, 4, 5 )
tup3 = "a", "b", "c", "d"
```

Порожній кортеж записується у вигляді двох круглих дужок, що не містять нічого:

The empty tuple is written as two parentheses containing nothing:

```
tup1 = ()
```

Щоб написати кортеж, що містить одне значення, ви маєте поставити кому:

To write a tuple containing a single value you have to include a comma:

```
tup1 = (50,)
```

Як і індекси рядків, індекси кортежів починаються з 0 і можуть бути розрізаними, об'єднаними і т. д.

Like string indices, tuple indices start at 0, and they can be sliced, concatenated, and so on.

Щоб отримати доступ до значень у кортежі, використовуйте квадратні дужки для нарізки разом з індексом або індексами, щоб одержати значення, доступні за цим індексом, наприклад:

To access values in tuple, use the square brackets for slicing along with the index or indices to obtain value available at that index. For example:

```
tup1 = ('physics', 'chemistry', 1997, 2000)
tup2 = (1, 2, 3, 4, 5, 6, 7 )
print ("tup1[0]: ", tup1[0])
print ("tup2[1:5]: ", tup2[1:5])
```

Коли наведений вище код виконується, маємо такий результат:

When the above code is executed, it produces the following result:

```
tup1[0]:  physics
tup2[1:5]:  [2, 3, 4, 5]
```

Кортежі є незмінними, що означає, що ви не можете оновлювати або змінювати значення елементів кортежу. Ви можете узяти частини наявних кортежів для створення нових кортежів, як показано в такому прикладі:

Tuples are immutable which means you cannot update or change the values of tuple elements. You are able to take portions of existing tuples to create new tuples as the following example demonstrates:

```
tup1 = (12, 34.56)
tup2 = ('abc', 'xyz')

# Following action is not valid for tuples
# tup1[0] = 100;

# So let's create a new tuple as follows
tup3 = tup1 + tup2
print(tup3)
```

Коли наведений вище код виконується, маємо такий результат:

When the above code is executed, it produces the following result:

```
(12, 34.56, 'abc', 'xyz')
```

Видалення окремих елементів кортежу є неможливим. Звичайно, немає нічого поганого в тому, щоб зібрати ще один кортеж з відкинутими небажаними елементами.

Removing individual tuple elements is not possible. There is, of course, nothing wrong with putting together another tuple with the undesired elements discarded.

Кортежі відповідають на оператори + і * так само, як рядки; вони також означають конкатенацію і повторення, за винятком того, що результатом є новий кортеж, а не рядок.

Tuples respond to the + and * operators much like strings; they mean concatenation and repetition here too, except that the result is a new tuple, not a string.

Фактично кортежі відповідають на всі загальні операції послідовності, які ми використовували в рядках або списках.

In fact, tuples respond to all of the general sequence operations we used on strings or lists.

Будь-який набір з декількох об'єктів, відокремлених комами, записаних без ідентифікувальних

Any set of multiple objects, comma-separated, written without identifying symbols, i.e., brackets for lists, parentheses for tuples, etc., default to tuples, as indicated

символів, тобто дужок для списків, дужок для кортежів і т. д., за замовчуванням дорівнює кортежам, як у таких коротких прикладах:

in these short examples:

```
print( 'abc', -4.24e93, 18+6.6j, 'xyz')
x, y = 1, 2
print( "Value of x , y : ", x, y)
```

Коли наведений вище код виконується, маємо такий результат:

When the above code is executed, it produces the following result:

```
abc -4.24e+93 (18+6.6j) xyz
Value of x , y : 1 2
```

2.4. Вбудовані й бібліотечні функції

Є декілька вбудованих функцій для виконання перетворення з одного типу даних на інший. Ці функції повертають новий об'єкт, що являє собою перетворене значення (табл. 2.4).

2.4. Built-in and library functions

There are several built-in functions to perform conversion from one data type to another. These functions return a new object representing the converted value (Table 2.4).

Таблця 2.4
Table 2.4

`int(x [,base])`	
Перетворює `x` на ціле число. `base` вказує базу, якщо `x` є рядком	Converts `x` to an integer. `base` specifies the base if `x` is a string
`long(x [,base] )`	
Перетворює `x` на довге ціле. `base` визначає базу, якщо `x` є рядком	Converts `x` to a long integer. base specifies the base if `x` is a string
`float(x)`	
Перетворює `x` на число з плаваючою крапкою	Converts `x` to a floating-point number
`bool(x)`	
Перетворює `x` на логічне число	Converts `x` to a Boolean number
`str(x)`	
Перетворює об'єкт `x` на рядок	Converts object `x` to a string
`repr(x)`	

Закінчення табл. 2.4
End of table 2.4

Перетворює об'єкт `x` на рядок виразу	Converts object `x` to an expression string
`eval(str)`	
Оцінює рядок і повертає об'єкт	Evaluates a string and returns an object
`tuple(s)`	
Перетворює `s` на кортеж	Converts s to a tuple
`list(s)`	
Перетворює `s` на список	Converts s to a list
`chr(x)`	
Перетворює ціле число на символ	Converts an integer to a character
`ord(x)`	
Перетворює один символ на його цілочислове значення	Converts a single character to its integer value
`hex(x)`	
Перетворює ціле число на шістнадцятковий рядок	Converts an integer to a hexadecimal string
`oct(x)`	
Перетворює ціле число на вісімковий рядок	Converts an integer to an octal string

Python містить такі функції, що виконують математичні обчислення (табл. 2.5).

Python includes following functions that perform mathematical calculations (Table 2.5).

Таблиця 2.5
Table 2.5

abs(x)	Абсолютне значення `x`: (додатне) відстань між `x` і нулем
	The absolute value of `x`: the (positive) distance between `x` and zero

Продовження табл. 2.5
Continuation of table 2.5

ceil(x)	Стеля x: найменше ціле число не менше x
	The ceiling of x: the smallest integer not less than x
cmp(x, y)	-1, якщо x <y, 0, якщо x == y, або 1, якщо x> y
	-1 if x < y, 0 if x == y, or 1 if x > y
exp(x)	Експонента x: e^x
	The exponential of x: e^x
fabs(x)	Абсолютне значення x
	The absolute value of x
floor(x)	Поверх x: найбільше ціле число не більше x
	The floor of x: the largest integer not greater than x
log(x)	Натуральний логарифм x для x>0
	The natural logarithm of x, for x>0
log10(x)	Логарифм x за основою 10 для x> 0
	The base-10 logarithm of x for x> 0
max (x1, x2,...)	Найбільший з його аргументів: значення, близьке до додатної нескінченності
	The largest of its arguments: the value closest to positive infinity
min (x1, x2,...)	Найменший з його аргументів: значення, близьке до від'ємної нескінченності
	The smallest of its arguments: the value closest to negative infinity
modf(x)	Дробова й ціла частини x у кортежі з двох елементів. Обидві частини мають той же знак, що й x. Ціла частина повертається як число з плаваючою крапкою
	The fractional and integer parts of x in a two-item tuple. Both parts have the same sign as x. The integer part is returned as a float
pow(x, y)	Значення x ** y
	The value of x**y
round (x [,n])	x округляється до n цифр від десяткової крапки. ***Python*** округляє від нуля як тай-брейк: round(0.5) дорівнює 1,0, а round (-0,5) дорівнює -1,0

Закінчення табл. 2.5
End of table 2.5

	`x` rounded to `n` digits from the decimal point. ***Python*** rounds away from zero as a tie-breaker: `round(0.5)` is `1.0` and `round(-0.5)` is `-1.0`
sqrt(x)	Квадратний корінь з `x` для `x> 0`
	The square root of `x` for `x > 0`

Модуль ***Python*** з математики містить функції, що виконують тригонометричні обчислення (табл. 2.6).

Python module math includes following functions that perform trigonometric calculations (Table 2.6).

Таблиця 2.6
Table 2.6

Опис функції Function & Description	
acos(x)	Повертає арккосинус `x`, у радіанах
	Return the arc cosine of `x`, in radians
asin(x)	Повертає дугу синуса `x`, у радіанах
	Return the arc sine of `x`, in radians
atan(x)	Повертає арктангенс `x`, у радіанах
	Return the arc tangent of `x`, in radians
atan2(y, x)	Повертає значення `atan(y/x)`, у радіанах
	Return `atan(y/x)`, in radians
cos(x)	Повертає косинус `x`, у радіанах
	Return the cosine of x, in radians
hypot(x, y)	Повертає евклідову норму, `sqrt(x*x + y*y)`
	Return the Euclidean norm, `sqrt(x*x + y*y)`
sin(x)	Повертає синус `x`, у радіанах
	Return the sine of x, in radians
tan(x)	Повертає тангенс `x`, у радіанах

Закінчення табл. 2.6
End of table 2.6

	Return the tangent of x, in radians
degrees(x)	Перетворює кут x з радіанів на градуси
	Converts angle x from radians to degrees
radians(x)	Перетворює кут x з градусів на радіани
	Converts angle x from degrees to radians

Модуль також визначає дві математичні константи (табл. 2.7).

The module also defines two mathematical constants (Table 2.7).

Таблиця 2.7
Table 2.7

pi	Математична константа pi
	The mathematical constant pi
E	Математична константа e
	The mathematical constant e

Python містить такі функції списку, які наведено в табл. 2.8.

Python includes the following list functions (Table 2.8).

Таблиця 2.8
Table 2.8

cmp(list1, list2)	Порівнює елементи обох списків
	Compares elements of both lists
len(list)	Дає загальну довжину списку
	Gives the total length of the list
max(list)	Повертає елемент зі списку з максимальним значенням
	Returns item from the list with max value
min(list)	Повертає елемент зі списку з мінімальним значенням
	Returns item from the list with min value
list(seq)	Перетворює кортеж на список
	Converts a tuple into list
tuple(seq)	Перетворює список на кортеж
	Converts a list into tuple

2.5. Оброблення виключень

Багато з описаних вище функцій, наприклад функції перетворення даних в разі неправильного формату параметрів, або операції, такі як ділення в разі нульового правого операнду, можуть спричинити помилки часу виконання з перериванням виконання сценарію. Ви можете контролювати подібні ситуації, використовуючи оператор `try`, що має такий синтаксис:

```
try:
     statement(s)
except [exceptionClassName]:
     processing statement(s)
else:
     normal statement(s)
[finally:
     always executed statement(s)]
```

Один оператор `try` може містити декілька операторів. Це корисно, коли блок `try` містить оператори, які можуть видавати різних типів виключення. Ви також можете подати універсальний оператор "виключення без виключення", який обробляє будь-яке виключення. Після пропозиції(й) виключень ви можете включити пропозицію `else`. Код у блоці `else` виконується, якщо код у блоці `try` не викликає виключення. Блок `finally` – це місце для розміщення будь-якого коду, який має виконуватися незалежно від того, викликав блок `try` виключення чи ні. Ви можете подати пункт `try` або пункт `finally`, але не обидва. Ви не можете використовувати пропозицію `else` разом з пропозицією `finally`, наприклад:

2.5. Exception handling

Many of the described above functions, for instance, data conversion functions in a case of invalid parameter format, or operations, such as division in a case of zero right operand, may cause run-time errors with interruption of the script execution. You may control similar situations using `try` statement, which has following syntax:

A single `try` statement can have multiple `except` statements. This is useful when the `try` block contains statements that may throw different types of exceptions. You can also provide a generic except clause without `exceptionClass Name`, which handles any exception. After the except clause(s), you can include an `else` clause. The code in the `else` block executes if the code in the try block does not raise an exception. The `finally` block is a place to put any code that must execute, whether the `try` block raised an exception or not. You can provide except clause(s), or a `finally` clause, but not both. You cannot use else clause as well along with a `finally` clause. For example:

```
try:
  A = int(input("A = "))
except:
  print("A MUST be integer!!!")
else:
  print ("A is positive: ", A>0)
```

Коли наведений вище код виконується, маємо такий результат:

When the above code is executed, it produces the following result:

```
A = -10
A is positive:  False
```

3. КЛАСИ Й ОБ'ЄКТИ В PYTHON

Мова ***Python*** є об'єктно-орієнтованою. Завдяки цьому створювати і використовувати класи й об'єкти дуже просто.

Ви повинні знати й розуміти основні поняття, щоб мати можливість використовувати бібліотечні класи і створювати свої власні прості класи. Наведемо мінімальний набір класів.

Клас – визначений користувачем тип даних для об'єкта, який визначає набір атрибутів, що характеризують будь-який об'єкт класу.

Атрибути – члени даних (змінні класу і змінні екземпляра) і методи, доступ до яких здійснюється через точковий запис.

Член даних – змінна класу або змінна екземпляра, яка містить дані, пов'язані з класом та його об'єктами.

Змінна екземпляра – змінна, визначена всередині методу і яка належить лише поточному екзем-

3. CLASSES AND OBJECTS IN PYTHON

Python has been an object-oriented language since the time it existed. Due to this, creating and using classes and objects are downright easy.

You should know and understand the basic concepts to be able to use library classes and to create your own simple classes. This is minimal set of them:

Class − a user-defined datatype for an object that defines a set of attributes that characterize any object of the class.

Attributes – data members (class variables and instance variables) and methods, accessed via dot notation.

Data member − a class variable or instance variable that holds data associated with a class and its objects.

Instance variable − a variable that is defined inside a method and belongs only to the current

пляру класу.

Змінна класу – це змінна, яка використовується всіма екземплярами класу. Змінні классу є визначеними всередині класу, але поза будь-яким з методів класу. Змінні класу використовуються не так часто, як змінні екземпляра.

Метод – особливий вигляд функції, що визначається класом.

Екземпляр – це окремий об'єкт певного класу.

Об'єкт – унікальний екземпляр структури даних, який визначається його класом.

instance of a class.

Class variable − a variable that is shared by all instances of a class. Class variables are defined within a class but outside any of the class's methods. Class variables are not used as frequently as instance variables are.

Method − a special kind of function that is defined in a class definition.

Instance − an individual object of a certain class.

Object − a unique instance of a data structure that is defined by its class.

3.1. Визначення класу й маніпулювання об'єктами

Оператор класу створює нове визначення класу. Ім'я класу йде відразу за ключовим словом `class`, за яким ставиться двокрапка:

3.1. Class definition and objects manipulation

The class statement creates a new class definition. The name of the class immediately follows the keyword `class` followed by a colon as follows:

```
class ClassName:
   'Optional class documentation string'
   class_suite
```

Імена класів починаються з великої букви. Усі останні ідентифікатори починаються з малої букви.

У класі є рядок документації, до якої можна отримати доступ через `Classname_doc_`.

`class_suite` складається з усіх операторів компонентів, що визначають члени класу, атрибути даних і функції (методи).

Class names start with an uppercase letter. All other identifiers start with a lowercase letter.

The class has a documentation string, which can be accessed via `ClassName._doc_`.

The `class_suite` consists of all the component statements defining class members, data attributes and functions (methods).

Початок ідентифікатора з єдиним початковим підкресленням означає, що ідентифікатор є приватним.

Початок ідентифікатора з двома провідними підкресленнями вказує на строго приватний ідентифікатор.

Якщо ідентифікатор також закінчується двома завершальними підкресленнями, то цей ідентифікатор є спеціальним ім'ям, що визначається мовою.

Наведемо приклад простого класу ***Python***:

Starting an identifier with a single leading underscore indicates that the identifier is private.

Starting an identifier with two leading underscores indicates a strongly private identifier.

If the identifier also ends with two trailing underscores, the identifier is a language-defined special name.

Following is an example of a simple ***Python*** class:

```
class Point:
   'Class for points on the 2D coordinate plane'
   pt_count = 0

   def __init__( self, x=0, y=0):
      self.x = x
      self.y = y
      Point.pt_count += 1

   def displayCount():
     print ("Total count of points:", Point.pt_count)

   def displayPoint(self):
        print ("X : ", self.x,  ", Y : ", self.y)
```

Змінна `pt_count` є змінною класу, значення якої є загальним для всіх екземплярів цього класу. Доступ до нього можна отримати як `Point.pt_count` усередині класу або за його межами.

Перший метод `_init_()` – це спеціальний метод, який називається конструктором класу або методом ініціалізації, який ***Python*** викликає при створенні нового екземпляра цього класу.

Ви оголошуєте інші методи класу, як звичайні функції, за винят-

The variable `pt_count` is a class variable whose value is shared among all the instances of this class. This can be accessed as `Point.pt_count` from inside the class or outside the class;

The first method `_init_()` is a special method, which is called class constructor or initialization method that ***Python*** calls when you create a new instance of this class;

You declare other class methods like normal functions

ком того, що першим аргументом кожного методу є `self`. ***Python*** додає аргумент `self` до списку для вас, але вам не потрібно включати його під час виклику методів.

Префікс перед елементом даних означає, що він є змінною екземпляра (також присутній лише всередині визначення класу). У наведеному вище прикладі: `self.name`, `self.salary`.

Клас може реалізувати спеціальний метод `_del_()`, який називають деструкцією, яка викликається, коли екземпляр може бути знищеним. Цей метод можна використовувати для очищення будь-яких ресурсів, не пов'язаних з пам'яттю, що використовуються екземпляром.

Щоб створити екземпляри класу, викликайте клас, використовуючи ім'я класу, і передайте будь-які аргументи, які приймає його метод `__init__()`.

Це створить два об'єкти класу `Point` з наведеного вище прикладу:

```
pt1 = Point(1,2)
pt2 = Point(3,4)
```

Ви отримаєте доступ до атрибутів об'єкта, використовуючи оператор крапки з об'єктом.

Оператори, що демонструють виклик різних методів класу `Point` з наведеного вище прикладу:

```
pt1.displayPoint()
pt2.displayPoint()
Point.displayCount()
```

with the exception that the first argument to each method is `self`. ***Python*** adds the `self` argument to the list for you, but you do not need to include it when you call the methods.

`self.` prefix before a data member means that it is an instance variable (also present only inside class definition). In the above example: `self.name`, `self.salary`.

Class can implement the special method `_del_()`, called a destructor, that is invoked when the instance is about to be destroyed. This method might be used to clean up any non-memory resources used by an instance

To create instances of a class, you call the class using class name and pass in whatever arguments its `__init__()` method accepts.

This would create two objects of `Point` class from the above example:

You access the object's attributes using the dot operator with an object.

Following statements demonstrate calling different methods of the class `Point` from the above example:

Щоб видалити об'єкт, використовуйте оператор `del`, який викликає відповідний виклик деструкції.

Оператори, додані до наведеного вище прикладу, що виводять ім'я класу екземпляра, який має бути знищений, і кількість об'єктів після знищення:

```
del pt1
Point.displayCount()
```

Коли наведений вище код виконується, маємо такий результат:

```
X :  1 , Y :  2
X :  3 , Y :  4
Total count of points: 2
```

To delete object, you use statement `del`, which causes appropriate destructor calling.

Following statements, adding to the above example, prints the class name of an instance that is about to be destroyed and count of objects after destroying:

When the above code is executed, it produces the following result:

```
Point destroyed
Total count of points: 1
```

3.2. Класи успадковування

Замість того, щоб починати з порожньої сторінки, можна створити клас, вивівши його із зумовленого класу, указавши породний клас в дужках після імені нового класу.

Дочірній (похідний) клас успадковує атрибути свого породного (базового) класу, і можна використовувати ці атрибути, начебто їх було визначено в дочірньому класі.

Синтаксис:

3.2. Classes inheritance

Instead of starting from a blank page, you can create a class by deriving it from a predefining class by listing the parent class in parentheses after the new class name.

The child (derived) class inherits the attributes of its parent (base) class, and you can use those attributes as if they were defined in the child class.

Syntax:

```
class SubClassName (ParentClass1[, ParentClass2, ...]):
   'Optional class documentation string'
   class_suite
```

Приклад:

Example:

```
class Parent:        # define parent class
   parentAttr = 100
   def __init__(self):
```

```
      print "Calling parent constructor"

   def parentMethod(self):
      print 'Calling parent method'

   def setAttr(self, attr):
      Parent.parentAttr = attr

   def getAttr(self):
      print "Parent attribute :", Parent.parentAttr

class Child(Parent): # define child class
   def __init__(self):
      print "Calling child constructor"

   def childMethod(self):
      print 'Calling child method'

c = Child()          # instance of child
c.childMethod()      # child calls its method
c.parentMethod()     # calls parent's method
c.setAttr(200)       # again call parent's method
c.getAttr()          # again call parent's method
```

Коли наведений вище код виконується, маємо такий результат:

When the above code is executed, it produces the following result:

```
Calling child constructor
Calling child method
Calling parent method
Parent attribute : 200
```

Аналогічним чином можна керувати класом з декількох породних класів. Дочірній клас також може перевизначати елементи даних і методи породного класу для досягнення поліморфної поведінки об'єктів з одним базовим кореневим класом.

Similar way, you can drive a class from multiple parent classes. A child class can also override data members and methods from the parent to achieve polymorph behavior of the objects with one base root class.

3.3. Основи масивів NumPy

3.3. The Basics of NumPy arrays

Основним об'єктом ***NumPy*** є багатовимірний масив. Це таблиця елементів (зазвичай чисел) одного типу, проіндексованих набором

NumPy's main object is the multidimensional array. It is a table of elements (usually numbers), all of the same type, in-

невід'ємних цілих чисел. У ***NumPy*** розміри називаються осями.

Наприклад, координати точки в тривимірному просторі `[1,2,1]` мають одну вісь. Ця вісь має три елементи, тому кажемо, що вона має довжину 3:

```
[[ 1., 0., 0.],
[ 0., 1., 2.]]
```

Клас масиву ***NumPy*** називається `ndarray`, а також масивом псевдонімів. Зверніть увагу, що `numpy.array` відрізняється від класу `array.array` стандартної бібліотеки ***Python***, який обробляє лише одновимірні масиви і має менше функціональних можливостей. Найважливішими атрибутами об'єкта є `ndarray` (табл. 3.1).

dexed by a tuple of non-negative integers. In NumPy dimensions are called axes.

For example, the coordinates of a point in 3D space `[1, 2, 1]` has one axis. That axis has 3 elements in it, so we say it has a length of 3:

NumPy's array class is called `ndarray`. It is also known by the alias array. Note that `numpy.array` is not the same as the Standard ***Python*** Library class `array.array`, which only handles one-dimensional arrays and offers less functionality. The more important attributes of an `ndarray` object are. (Table 3.1).

Таблиця 3.1
Table 3.1

`ndarray.ndim`	
Кількість осей (розмірів) масиву	The number of axes (dimensions) of the array
`ndarray.shape`	
Розміри масиву. Це кортеж цілих чисел, що вказують розмір масиву в кожному вимірі. Для матриці з `n` рядками і `m` стовпцями форма буде `(n,m)`. Отже, довжина кортежу форми – це кількість осей `ndim`	The dimensions of the array. This is a tuple of integers indicating the size of the array in each dimension. For a matrix with `n` rows and `m` columns, shape will be `(n,m)`. The length of the shape tuple is therefore the number of axes `ndim`
`ndarray.size`	
Загальна кількість елементів масиву. Це дорівнює добудку елементів форми	The total number of elements of the array. This is equal to the product of the elements of shape

Закінчення табл. 3.1
End of table 3.1

ndarray.dtype	
Об'єкт, що описує типи елементів у масиві. Можна створювати або вказувати dtype, використовуючи стандартні типи ***Python***. Крім того, ***NumPy*** має власні типи, наприклад numpy.int32, numpy. int16 і numpy.float64	An object describing the type of the elements in the array. One can create or specify dtype's using standard ***Python*** types. Additionally ***NumPy*** provides types of its own numpy.int32, numpy.int16, and numpy.float64 are some examples
ndarray.itemsize	
Розмір у байтах кожного елемента масиву. Наприклад, масив елементів типу float64 має розмір елементів 8(= 64/8), тоді як один із типів complex32 має розмір елементів 4 4(=32/8). Це є еквівалентним ndarray.dtype.itemsize	The size in bytes of each element of the array. For example, an array of elements of type float64 has itemsize 8(=64/8), while one of type complex32 has itemsize 4 (=32/8). It is equivalent to ndarray.dtype. itemsize
ndarray.data	
Буфер, що містить фактичні елементи масиву. Зазвичай нам не потрібно використовувати цей атрибут, тому що ми звертатимемося до елементів в масиві, використовуючи засоби індексації	The buffer containing the actual elements of the array. Normally, we won't need to use this attribute because we will access the elements in an array using indexing facilities

Приклад: Example:

```
>>> import numpy as np
>>> a = np.arange(15).reshape(3, 5)
>>> a
array([[ 0,  1,  2,  3,  4],
       [ 5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14]])
>>> a.shape
(3, 5)
>>> a.ndim
2
>>> a.dtype.name
```

```
'int64'
>>> a.itemsize
8
>>> a.size
15
>>> type(a)
<type 'numpy.ndarray'>
>>> b = np.array([6, 7, 8])
>>> b
array([6, 7, 8])
>>> type(b)
<type 'numpy.ndarray'>
```

Array Creation

Є декілька способів створення масивів. Наприклад, можна створити масив зі звичайного списку ***Python*** або кортежу, використовуючи функцію масиву.

Тип отриманого масиву визначається типом елементів у послідовностях:

There are several ways to create arrays.For example, you can create an array from a regular ***Python*** list or tuple using the array function.

The type of the resulting array is deduced from the type of the elements in the sequences:

```
>>> import numpy as np
>>> a = np.array([2,3,4])
>>> a
array([2, 3, 4])
>>> a.dtype
dtype('int64')
>>> b = np.array([1.2, 3.5, 5.1])
>>> b.dtype
dtype('float64')
```

Метод `array()` перетворить послідовності послідовностей на двовимірні масиви, послідовності послідовностей послідовностей на тривимірні масиви і т. д.:

The `array()` method transforms sequences of sequences into two-dimensional arrays, sequences of sequences of sequences into three-dimensional arrays, and so on:

```
>>> b = np.array([(1.5,2,3), (4,5,6)])
>>> b
array([[ 1.5,  2. ,  3. ],
       [ 4. ,  5. ,  6. ]])
```

Тип масиву також може бути явно вказаний під час створення:

The type of the array can also be explicitly specified at creation time:

```
>>> c = np.array( [ [1,2], [3,4] ], dtype=complex )
>>> c
array([[ 1.+0.j,  2.+0.j],
       [ 3.+0.j,  4.+0.j]])
```

Функція `zeros()` створює масив, що складається з нулів, функція `ones()` створює масив, що складається з одиниць, а функція `empty` створює масив, початковий уміст якого є випадковим і залежить від стану пам'яті. За замовчуванням `d`-тип створеного масиву – `float64`:

The function `zeros()` creates an array full of zeros, the function `ones()` creates an array full of ones, and the function `empty` creates an array whose initial content is random and depends on the state of the memory. By default, the d-type of the created array is `float64`:

```
>>> np.zeros( (3,4) )
array([[ 0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.]])
>>> np.ones( (2,3,4), dtype=np.int16 ) # dtype can also be
specified
array([[[ 1, 1, 1, 1],
        [ 1, 1, 1, 1],
        [ 1, 1, 1, 1]],
       [[ 1, 1, 1, 1],
        [ 1, 1, 1, 1],
        [ 1, 1, 1, 1]]], dtype=int16)
>>> np.empty( (2,3) )       # uninitialized, output may vary
array([[ 3.73603959e-262, 6.02658058e-154,  6.55490914e-260],
       [5.30498948e-313,3.14673309e-307, 1.00000000e+000]])
```

Для створення послідовностей чисел ***NumPy*** надає функцію, аналогічну `range()`, яка повертає масиви замість списків:

To create sequences of numbers, ***NumPy*** provides a function analogous to `range()` that returns arrays instead of lists:

```
>>> np.arange( 10, 30, 5 )
array([10, 15, 20, 25])
>>> np.arange( 0, 2, 0.3 )          # accepts float arguments
array([ 0. ,  0.3,  0.6,  0.9,  1.2,  1.5,  1.8])
```

Арифметичні оператори на масивах застосовуються поелементно. Новий масив створений і заповнений результатом:

Arithmetic operators on arrays apply elementwise. A new array is created and filled with the result:

```
>>> a = np.array( [20,30,40,50] )
```

```
>>> b = np.arange( 4 )
>>> b
array([0, 1, 2, 3])
>>> c = a-b
>>> c
array([20, 29, 38, 47])
>>> b**2
array([0, 1, 4, 9])
>>> 10*np.sin(a)
array([ 9.12945251, -9.88031624,  7.4511316 , -2.62374854])
>>> a<35
array([ True, True, False, False])
```

На відміну від багатьох матричних мов, оператор продукту * працює поелементно в масивах ***NumPy***. Матрицю можна створити за допомогою оператора @ (у python >= 3.5) або функції або методу крапки:

Unlike in many matrix languages, the product operator * operates elementwise in ***NumPy*** arrays. The matrix product can be performed using the @ operator (in python >=3.5) or the dot function or method:

```
>>> A = np.array( [[1,1],
...               [0,1]] )
>>> B = np.array( [[2,0],
...               [3,4]] )
>>> A * B                          # elementwise product
array([[2, 0],
       [0, 4]])
>>> A @ B                          # matrix product
array([[5, 4],
       [3, 4]])
>>> A.dot(B)                       # another matrix product
array([[5, 4],
       [3, 4]])
```

При роботі з масивами різних типів тип результівного масиву відповідає загальнішому або точнішому (поведінка, відома як `upcasting`):

When operating with arrays of different types, the type of the resulting array corresponds to the more general or precise one (a behavior known as `upcasting`):

```
>>> a = np.ones(3, dtype=np.int32)
>>> b = np.linspace(0,pi,3)
>>> b.dtype.name
'float64'
>>> c = a+b
>>> c
array([ 1.        ,  2.57079633,  4.14159265])
>>> c.dtype.name
```

```
'float64'
>>> d = np.exp(c*1j)
>>> d
array([ 0.54030231+0.84147098j, -0.84147098+0.54030231j,
       -0.54030231-0.84147098j])
>>> d.dtype.name
'complex128'
```

Багато унарних операцій, таких як обчислення суми всіх елементів у масиві, реалізовано як методи класу `ndarray`:

Many unary operations, such as computing the sum of all the elements in the array, are implemented as methods of the `ndarray` class:

```
>>> a = np.random.random((2,3))
>>> a
array([[ 0.18626021,  0.34556073,  0.39676747],
       [ 0.53881673,  0.41919451,  0.6852195 ]])
>>> a.sum()
2.5718191614547998
>>> a.min()
0.1862602113776709
>>> a.max()
0.6852195003967595
```

За замовчуванням ці операції застосовуються до масиву, начебто це список чисел незалежно від його форми. Проте, указавши параметр осі, можна застосувати операцію вздовж указаної осі масиву:

By default, these operations apply to the array as though it were a list of numbers, regardless of its shape. However, by specifying the axis parameter you can apply an operation along the specified axis of an array:

```
>>> b = np.arange(12).reshape(3,4)
>>> b
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
>>>
>>> b.sum(axis=0)                        # sum of each column
array([12, 15, 18, 21])
>>>
>>> b.min(axis=1)                        # min of each row
array([0, 4, 8])
>>>
>>> b.cumsum(axis=1)                     # cumulative sum along each row
array([[ 0,  1,  3,  6],
       [ 4,  9, 15, 22],
       [ 8, 17, 27, 38]])
```

4. ПРОГРАМУВАННЯ TKINTER GUI

Tkinter – це стандартна бібліотека графічного інтерфейсу користувача (***GUI***) для ***Python***. ***Python*** у поєднанні з ***Tkinter*** забезпечує швидкий і простий спосіб створення додатків з графічним інтерфейсом. ***Tkinter*** надає потужний об'єктно-орієнтований інтерфейс для інструментарію ***Tk GUI***.

Створення додатка з графічним інтерфейсом за допомогою ***Tkinter*** – просте завдання. Усе, що потрібно зробити, це виконати такі кроки:

- імпортувати модуль ***Tkinte***;
- створити головне вікно додатка з графічним інтерфейсом;
- додати один або декілька зазначених вище віджетів у додаток з графічним інтерфейсом;
- увійти до основного циклу подій, щоб вжити заходів проти кожної події, ініційованої користувачем.

Приклад:

```
import tkinter
top = tkinter.Tk()
# Code to add widgets will go here...
top.mainloop()
```

Це створить вікно, яке зображено на рис. 4.1.

Tkinter має різні елементи керування, такі як кнопки, мітки й текстові поля, що використовуються в додатку з графічним інтерфейсом. Ці елементи керування зазвичай називаються віджетами. Ці віджети з коротким описом наведено в табл. 4.1.

4. TKINTER GUI PROGRAMMING

Tkinter is the standard Graphical User Interface (***GUI***) library for ***Python***.***Python*** when combined with ***Tkinter*** provides a fast and easy way to create ***GUI*** applications. ***Tkinter*** provides a powerful object-oriented interface to the Tk ***GUI*** toolkit.

Creating a ***GUI*** application using Tkinter is an easy task. All you need to do is perform the following steps:

- import the ***Tkinter*** module.
- create the ***GUI*** application main window.
- add one or more of the above-mentioned widgets to the ***GUI*** application.
- enter the main event loop to take action against each event triggered by the user.

Example:

This would create a following window (Fig. 4.1).

Tkinter provides various controls, such as buttons, labels and text boxes used in a GUI application. These controls are commonly called widgets. These widgets with a brief description are presented in the Table. 4.1.

Рис. 4.1. Порожнє головне вікно

Fig. 4.1. Empty Main Window

Таблиця 4.1
Table 4.1

Tkinter widgets	
Button	
Використовується для відображення кнопок у вашому додатку	Is used to display buttons in your application
Canvas	
Використовується для зображення фігур, таких як лінії, овали, багатокутники й прямокутники, у вашому додатку	Is used to draw shapes, such as lines, ovals, polygons and rectangles, in your application
Checkbutton	
Використовується для відображення параметрів у вигляді прапорців. Користувач може вибрати декілька варіантів одночасно	Is used to display a number of options as checkboxes. The user can select multiple options at a time
Entry	
Використовується для відображення однорядкового текстового поля для отримання значень від користувача	Is used to display a single-line text field for accepting values from a user
Frame	
Використовується як контейнерний віджет для організації інших віджетів	Is used as a container widget to organize other widgets

Продовження табл. 4.1
Table continuation 4.1

Label	
Використовується для надання однорядкового заголовка для інших віджетов. Він також може містити зображення	Is used to provide a single-line caption for other widgets. It can also contain images
Listbox	
Використовується для надання користувачеві списку параметрів	Is used to provide a list of options to a user
Menubutton	
Використовується для відображення меню у вашому додатку	Is used to display menus in your application
Menu	
Використовується для надання різних команд користувачеві. Ці команди містяться всередині Menubutton	Is used to provide various commands to a user. These commands are contained inside Menubutton
Radiobutton	
Використовується для відображення параметрів як перемикачів. Користувач може вибрати лише один варіант за раз	Is used to display a number of options as radio buttons. The user can select only one option at a time
Scale	
Використовується для надання віджета слайдера	Is used to provide a slider widget
Scrollbar	
Використовується для додавання можливості прокрутки до різних віджетів, таких як списки	Is used to add scrolling capability to various widgets, such as list boxes
Text	
Використовується для відображення тексту в декілька рядків	Is used to display text in multiple lines
Toplevel	
Використовується для надання окремого вікна контейнера	Is used to provide a separate window container

Закінчення табл. 4.1

End of table 4.1

Spinbox	
Варіант стандартного віджета Tkinter Entry, який можна використовувати для вибору фіксованої кількості значень	Is a variant of the standard Tkinter Entry widget, which can be used to select from a fixed number of values
PanedWindow	
Є контейнерним віджетом, який може містити будь-яку кількість панелей, розташованих горизонтально або вертикально	Is a container widget that may contain any number of panes, arranged horizontally or vertically
LabelFrame	
Простий контейнерний віджет. Його основне призначення – розділь-ник або контейнер для складних віконних схем.	Is a simple container widget. Its primary purpose is to act as a spacer or container for complex window layouts

4.1. Віджети Tkinter

Ось простий синтаксис для створення будь-якого віджета:

```
w = widgetName ( master, option, ... )
```

Параметри:

- `master` – породне вікно;
- `options` – список найбільш часто використовуваних опцій для цього віджета.

Ці параметри можуть використовуватися як іменовані параметри, розділені комами.

Кожен віджет має набір властивостей, які визначають його зовнішній вигляд на екрані комп'ютера і те, як він реагує на призначені для користувача події. Існує безліч властивостей, загальних для всіх віджетів `tk`. Деякі з них наведено в табл. 4.2.

4.1. Tkinter widgets

Here is the simple syntax to create any widget:

Parameters:

- `master` represents the parent window;
- `options` is the list of most commonly used options for this widget.

These options can be used as named parameters separated by commas.

Each widget has a set of properties that define its visual appearance on the computer screen and how it responds to user events. There is a set of properties that all `tk` widgets have in common. Some of these are shown in the Table 4.2.

Таблиця 4.2
Table 4.2

bg	Фоновий колір
	Background color
fg	Колір переднього плану
	Foreground color
width	Ширина в пікселях
	Width in pixels
height	Висота в пікселях
	Height in pixels
borderwidth	Розмір меж у пікселях
	The size of the border in pixels
text	Текст відображується на віджеті
	Text displayed on the widget
font	Шрифт, що використовується для тексту у віджеті
	The font used for text on the widget
cursor	Форма курсора, коли курсор знаходиться над віджетом
	The shape of the cursor when the cursor is over the widget
activeforeground	Колір тексту при активації віджета
	The color of the text when the widget is activated
activebackground	Колір фону при активованому віджеті
	The color of the background when the widget is activated
image	Зображення для відображення на віджеті
	An image to be displayed on the widget

Кожний тип віджета має властивості, визначені для його передбачуваного використання. Деякі приклади наведено в табл. 4.3.

Each type of widget has properties specific to its intended use. Here are some examples Table 4.3.

Таблиця 4.3
Table 4.3

Конкретний віджет (Specific Widget)		
Опис (Properties)	Належить до (Applies to)	Опис (Description)
`variable`	Checkbutton Radiobutton Entry Spinbox Text	Об'єкт `Tk`, який буде змінений при взаємодії з віджетом
		An `Tk` object that will be changed by interaction with the widget
`from_`	`Scale`	Початкове значення шкали (тобто повзунок)
		The starting value of a scale (i.e., a slider)
`to`	`Scale`	Кінцеве значення шкали (тобто повзунок)
		The ending value of a scale (i.e., a slider)
`orient`	`Scale`	ГОРИЗОНТАЛЬНИЙ або ВЕРТИКАЛЬНИЙ
		HORIZONTAL or VERTICAL
`resolution`	`Scale`	Величина приросту за шкалою (тобто повзунок)
		The increment amount along a scale (i.e., a slider)

Деякі віджети мають «змінну», якою маніпулює користувач. Для таких віджетів ви маєте створити спеціальний об'єкт `tk`, у якому зберігається значення віджета. По суті, ви створюєте невидимий віджет, з яким можуть бути зв'язані події. Ви можете отримати доступ до значення віджета через методи `set(new_value)` і `get()` для об'єкта змінної. Існує чотири типи змінних об'єктів (табл. 4.4).

Some widgets have a “variable” that a user manipulates. For such widgets you must create a special `tk` object that stores the widget’s value. Basically you are creating a non-visible widget that can have events associated with it. You can access the widget’s value through `set(new_value)` and `get()` methods on the variable object. There are four types of variable objects (Table 4.4).

Таблиця 4.4
Table 4.4

Змінна об'єкта `tk` (`tk` variable object)	Опис (Description)
`tk.BooleanVar`	Об'єкт tk, який містить єдине логічне значення
	A `tk` object that holds a single Boolean value
`tk.IntVar`	Об'єкт `tk`, який містить одне цілочислове значення
	A `tk` object that holds a single integer value
`tk.DoubleVar`	Об'єкт `tk`, який містить одне подвійне значення
	A `tk` object that holds a single double value
`tk.StringVar`	Об'єкт `tk`, який містить одне рядкове значення
	A `tk` object that holds a single string value

Візьмемо `tk.Check-button` як приклад та опишемо кроки, необхідні для визначення його «значення». Необхідно виконати такі дії:

1. Створити об'єкт `tk.BooleanVar`. (Насправді можна використовувати будь-який тип об'єкта змінної.)
2. Присвоїти об'єкту `tk.Boolean-Var` початкове значення, яке ви хочете для цієї кнопки.
3. Присвоїти атрибут змінної пра-

Let’s take a `tk.Checkbut-ton` as an example and walk through the steps needed to get and set its “value”. You need to do three things:

1. Create a `tk.BooleanVar` object. (Actually any type of variable object can be used.)
2. Give the `tk.BooleanVar` object the initial value you want for the checkbutton.
3. Assign the variable attribute

порця об'єкту `tk.BooleanVar`.

Скрипт, наведений нижче, демонструє приклад визначення класу ***GUI*** для роботи з певними віджетами для введення деякого числового параметра та його збільшення в разі успішного введення:

of the checkbutton to the `tk.BooleanVar` object.

Following script demonstrates an example of ***GUI*** class definition for working with certain widgets to enter some number parameter and increment it if input is successful:

```
import tkinter
from tkinter import messagebox

class MainWindow(tkinter.Frame):
    "GUI Class"
    def __init__(self, parent):
        "Constructor - Initial GUI customization"

        super(MainWindow, self).__init__(parent)
        self.parent = parent
        self.pack(fill=tkinter.BOTH,expand=1)
        self.lb1=tkinter.Label(self, text = "Enter some pa-
rameter:", width = 30)

        self.p_str = tkinter.StringVar()
        self.parEntr = tkinter.Entry(self, textvariable =
self.p_str, width = 30)
        self.but1 = tkinter.Button(self, text = "Input param-
eter", command = self.parInput)

        self.lb1.pack()
        self.parEntr.pack()
        self.but1.pack(side = tkinter.RIGHT)

    def parInput(self):
        "Parameter input method"
        try:
            par=float(self.parEntr.get())
        except:
            messagebox.showerror("ERROR", "Must be number!")
        else:
            messagebox.showinfo("Success",
self.parEntr.get())
            self.p_str.set(str(par+1))

application = tkinter.Tk()
application.title("GUI example")
window = MainWindow(application)
application.mainloop()
```

Тут клас Mainwindow з двома методами й чотирма атрибутами екземпляра був визначений як успадкований від стандартного віджета Frame. По-перше, при трьох затвердженнях методу конструктора _init_() вікно набудовується шляхом виклику конструктора з базового класу й виклику методу pack() поточного віджета. Потім створюються чотири об'єкти: self.lb1 класу Label,self, parentr класу Entry, self.but1 класу Button і self.p_str як Stringvar, зв'язаний зі значенням віджета входу. Конструктор закінчується трьома операторами, у яких об'єкти об'єднуються у вікні з допомогою методу pack().

Метод parinput() містить try. окрім оператора з двома варіантами оброблення. Якщо значення у віджеті запису, доступ до якого здійснюється з допомогою методу get(), є недопустимими (або порожніми) плаваючими даними, то з'являється повідомлення «Має бути число!», що відображується в окремому вікні. В іншому випадку буде показано повідомлення зі значенням і змінною атрибута p_str буде призначено інкрементний параметр з допомогою методу set(). Це приведе до зміни тексту у віджеті parentr.

В основному скрипті об'єкт додатка створюється з допомогою функції Tk() і потім набудовується з новим заголовком вікна «Приклад графічного інтерфейсу».

Після цього вікно об'єкта створю-

Here class MainWindow with two methods and four instance attributes has been defined as inherited from standard widget Frame. In the first three statements of constructor method _init_() the window is customized by calling constructor from the base class and calling pack() method of current widget. Then four objects are created: self.lb1 of class Label, self.parEntr of class Entry, self.but1 of class Button and self.p_str as StringVar connected with value of entry widget. Constructor ends with three statements where the objects are compounded on the window using pack() method.

The method parInput() contains try except statement with two variants of processing. If value in entry widget, which accessed with method get(), is not valid float data (or empty) then message "Must be number!" will be displayed in a separate window. In other case message with the value will be shown and incremented parameter variable will be assigned to p_str attribute using method set(). That will causes text change in the parEntr widget.

In the main script application object are created by means of Tk() function and then are customized with new window title "GUI example".

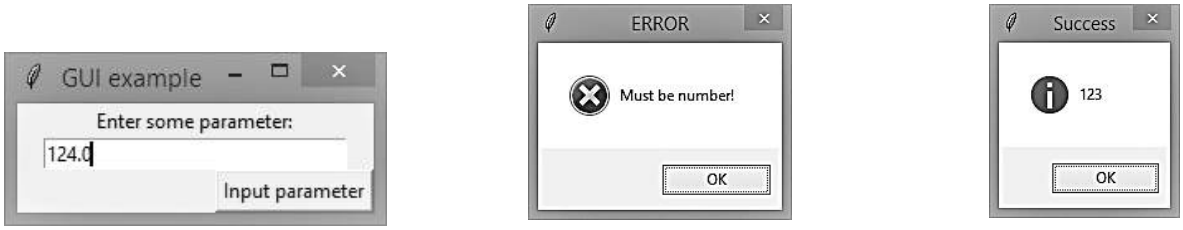
After that object window are

ється як екземпляр раніше визначеного класу `Mainwindow` і запускається основний цикл додатка.

Коли наведений вище код виконується, маємо такий результат (рис. 4.2).

created as instance of previously defined `MainWindow` class and main loop of the application is run.

When the above code is executed, it produces the following result (Fig. 4.2).



а *б* *в*

Рис. 4.2. Головне вікно (*а*) і вікна повідомлень (*б, в*)

Fig. 4.2. Main Window (*а*) and message boxes (*б, в*)

Віджет «Мітка» реалізує поле відображення, у якому можна розмістити текст або зображення. Текст, що відображується цим віджетом, може бути оновлений у будь-який час. Також можна підкреслити частину тексту (наприклад, ідентифікувати поєднання клавіш) і розбити текст на декілька рядків.

Віджет «Button» використовується для додавання кнопок у додаток ***Python***. Ці кнопки можуть відображувати текст або зображення, що передають призначення кнопок. Можна прикріпити функцію або метод до кнопки (параметр команди), що викликається автоматично при натисненні кнопки.

Віджет «Запис» використовується для приймання однорядкових текстів від користувача. Метод `get()` повертає рядкове значення, яке містить віджет «Entry».

Якщо ви хочете відображувати декілька рядків тексту, які можна редагувати, вам слід використову-

The «Label» widget implements a display box where you can place text or images. The text displayed by this widget can be updated at any time you want. It is also possible to underline part of the text (like to identify a keyboard shortcut) and span the text across multiple lines.

The «Button» widget is used to add buttons in a ***Python*** application. These buttons can display text or images that convey the purpose of the buttons. You can attach a function or a method to a button (command parameter) which is called automatically when you click the button.

The «Entry» widget is used to accept single-line text strings from a user. The `get()` method returns string value which «Entry» widget contains.

If you want to display multiple lines of text that can be edited,

вати віджет «Текст». Якщо ви хочете відображувати один або декілька рядків тексту, які не можуть бути змінені користувачем, вам слід використовувати віджет «Мітка».

4.2. Керування геометрією

Tkinter pack() з наведеного вище прикладу організовує віджети в блоки перед розміщенням їх у породному віджеті:

```
widget.pack(pack_options)
```

Список можливих варіантів pack_options:

- розвернути – при значенні true віджет розширюється, заповнюючи будь-який простір, що не використовується в породному віджеті;
- заповнити – визначає, чи заповнює віджет який-небудь додатковий простір, виділений йому пакувальником, чи зберігає свої власні мінімальні розміри: HI (за замовчуванням), X (заповнення лише по горизонталі), Y (заповнення лише по вертикалі) або ОБИДВА (заповнення як по горизонталі, так і по вертикали);
- сторона – визначає, з якого боку породного віджета упаковуватиметься: TOP (за замовчуванням), BOTTOM, LEFT або RIGHT.

Приклад:

```
from Tkinter import *

root = Tk()
frame = Frame(root)
frame.pack()

bottomframe = Frame(root)
bottomframe.pack( side = BOTTOM )
```

then you should use the «Text» widget. If you want to display one or more lines of text that cannot be modified by the user, then you should use the «Label» widget.

4.2. Geometry management

Tkinter pack() method from the above example organizes widgets in blocks before placing them in the parent widget:

Here is the list of possible pack_options:

- Expand – when set to true, widget expands to fill any space not otherwise used in widget's parent.
- Fill – determines whether widget fills any extra space allocated to it by the packer, or keeps its own minimal dimensions: NONE (default), X (fill only horizontally), Y (fill only vertically), or BOTH (fill both horizontally and vertically).
- Side – determines which side of the parent widget packs against: TOP (default), BOTTOM, LEFT, or RIGHT.

Example:

```
redbutton = Button(frame, text="Red", fg="red")
redbutton.pack( side = LEFT)

greenbutton = Button(frame, text="green", fg="green")
greenbutton.pack( side = LEFT )

bluebutton = Button(frame, text="Blue", fg="blue")
bluebutton.pack( side = LEFT )

blackbutton = Button(bottomframe, text="Black", fg="black")
blackbutton.pack( side = BOTTOM)

root.mainloop()
```

Коли наведений вище код виконується, маємо результат, зображений на рис. 4.3.

When the above code is executed, it produces the following result (Fig. 4.3).

Рис. 4.3. Головне вікно з кнопками, набитими різними параметрами

Fig. 4.3. Main Window with buttons packed with different parameters

Tkinter надає інші методи менеджера геометрії: `grid()` і `place()`.

Метод ***Tkinter*** `grid()` організовує віджети у вигляді таблиці в породному віджеті:

Tkinter exposes another geometry manager methods: `grid()` and `place()`.

Tkinter `grid()` method organizes widgets in a table-like structure in the parent widge:

```
widget.grid(grid_options)
```

Список можливих варіантів:

Here is the list of possible options:

- `column` – стовпець для розміщення віджета; за замовчуванням 0 (крайній лівий стовпець);
- `columnspan` – скільки стовпців віджетів займає; за замовчуванням 1;
- `ipadx, ipady` – скільки пікселів для заповнення віджета, по горизонталі й вертикалі, усередині меж віджета;
- `row` – рядок для розміщення від-

- `column` – The column to put widget in; default 0 (leftmost column).
- `columnspan` – How many columns widgetoccupies; default 1.
- `ipadx, ipady` – How many pixels to pad widget, horizontally and vertically, inside widget's borders.
- `row` – The row to put widget

жета; за замовчуванням перший рядок, який усе ще є порожнім;

- `rowspan` – скільки займає `rowswidget`, за замовчуванням `1`;
- `sticky` – що робити, якщо комірка більше віджета. За замовчуванням при використанні `sticky = ''` віджет центрується в своїй комірці. `sticky` може бути конкатенацією рядків з нуля або більш з `N`, `E`, `S`, `W`, `NE`, `NW`, `SE` і `SW`, напрями компаса вказують сторони і кути комірки, до якого прив'язаний віджет.

in; default the first row that is still empty.

- `rowspan` – How many rowswidget occupies; default 1.
- `sticky` – What to do if the cell is larger than widget. By default, with `sticky=''`, widget is centered in its cell. `sticky` may be the string concatenation of zero or more of `N`, `E`, `S`, `W`, `NE`, `NW`, `SE`, and `SW`, compass directions indicating the sides and corners of the cell to which widget sticks.

Приклад:

Example:

```
import Tkinter
root = Tkinter.Tk(  )
for r in range(3):
   for c in range(4):
      Tkinter.Label(root, text='R%s/C%s'%(r,c),
         borderwidth=1 ).grid(row=r,column=c)
root.mainloop(  )
```

Це приведе до результату, що відображає 12 міток, розміщених у сітці 3 × 4 (рис. 4.4).

This would produce the following result displaying 12 labels arrayed in a 3 × 4 grid (Fig. 4.4).

Рис. 4.4. Головне вікно з мітками, розміщеними в сітці 3x4

Fig. 4.4. Main Window with labels put in the grid 3x4

Метод ***Tkinter*** `place()` організовує віджети, розміщуючи їх в певній позиції в породному віджеті. Синтаксис:

Tkinter `place()` method organizes widgets by placing them in a specific position in the parent widget. Syntax:

```
widget.place( place_options )
```

Список можливих варіантів:

Here is the list of possible op-

- якір – точне місце віджета, до якого належать інші параметри: можуть бути N, E, S, W, NE, NW, SE або SW, напрямки компаса, вказівні кути й сторони віджета; за замовчуванням NW (верхній лівий кут віджета);

- bordermode – INSIDE (за замовчуванням), щоб указати, що інші опції належать до внутрішньої частини породного віджета (ігноруючи межі породного віджета);
- висота, ширина – висота й ширина в пікселях;
- relheight, relwidth – висота й ширина у вигляді числа з плаваючою комою в діапазоні від 0,0 до 1,0 у вигляді частки висоти й ширини породного віджета;
- relx, rely – зсув по горизонталі й вертикалі у вигляді числа з плаваючою крапкою від 0,0 до 1,0 у вигляді частки висоти й ширини породного віджета;
- x, y – горизонтальний і вертикальний зсув у пікселях.

Приклад:

tions:

- anchor – The exact spot of widget other options refer to: may be N, E, S, W, NE, NW, SE, or SW, compass directions indicating the corners and sides of widget; default is NW (the upper left corner of widget)
- bordermode – INSIDE (the default) to indicate that other options refer to the parent's inside (ignoring the parent's border); OUTSIDE otherwise.

- height, width – Height and width in pixels.
- relheight, relwidth – Height and width as a float between 0.0 and 1.0, as a fraction of the height and width of the parent widget.
- relx, rely – Horizontal and vertical offset as a float between 0.0 and 1.0, as a fraction of the height and width of the parent widget;
- x, y – Horizontal and vertical offset in pixels.

Example:

```
from Tkinter import *
import tkMessageBox
import Tkinter

top = Tkinter.Tk()
def helloCallBack():
   tkMessageBox.showinfo( "Hello Python", "Hello World")

B = Tkinter.Button(top, text ="Hello", command =
helloCallBack)

B.pack()
B.place(bordermode=OUTSIDE, height=100, width=100)
top.mainloop()
```

Коли наведений вище код виконується, маємо результат, зображений на рис. 4.5.

When the above code is executed, it produces the following result (Fig. 4.5).

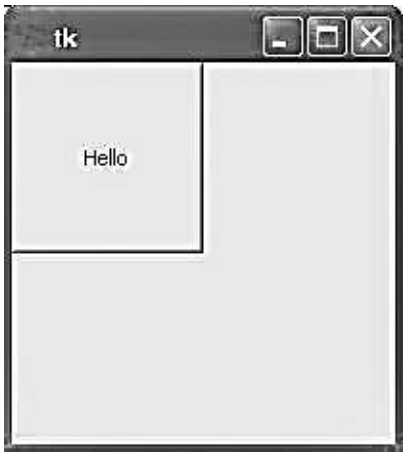


Рис. 4.5. Головне вікно з кнопкою, розташованою в квадратній області 100x100 пікселів

Fig. 4.5. Main Window with button placed in the square area 100x100 pixels

4.3. Стандартні діалогові вікна

Метод `messagebox()`, який використовувався в наведеному вище прикладі з класом для ***GUI***, може відображувати інформацію для користувача. У цих діалогових вікнах є три варіанти залежно від типу повідомлення, яке ви хочете відображувати. Перший параметр функції задає ім'я для діалогового вікна, що відображується в заголовку вікна. Другий параметр – це текст повідомлення. Функції повертають рядок, який зазвичай ігнорується:

4.3. Standard Dialog Boxes

A `messagebox()` method, which has been used in the above example with class for GUI, can display information to a user. There are three variations on these dialog boxes based on the type of message you want to display. The functions' first parameter gives a name for the dialog box which is displayed in the window's header. The second parameter is the text of the message. The functions return a string which is typically ignored:

```
from tkinter import messagebox
messagebox.showinfo("Information","Informative message")
messagebox.showerror("Error", "Error message")
messagebox.showwarning("Warning","Warning message")
```

Об'єкт ***Tkinter*** `messagebox` також дає вам змогу ставити користувачеві прості запитання типу так/ні і змінює імена кнопок залежно від типу запитання. Це такі функції:

The ***Tkinter*** `messagebox` object also allows you to ask a user simple yes/no type questions and varies the button names based on the type of question. These functions are:

```
from tkinter import messagebox
answer = messagebox.askokcancel("Question","Do you want to
open this file?")
answer = messagebox.askretrycancel("Question", "Do you want
to try that again?")
answer = messagebox.askyesno("Question","Do you like Py-
thon?")
answer = messagebox.askyesnocancel("Question", "Continue
playing?")
```

Повертане значення – логічне, `True` або `False`, відповідь на запитання. Якщо кнопка «відміна» є опцією і користувач вибирає цю кнопку, то нічого не повертається. Інші діалоги, які вам можуть знадобитися, – це `simpledialog` і `filedialog`.

Якщо ви хочете запитати у користувача одне значення даних – рядок, ціле число або значення з плаваючою комою, ви можете використовувати об'єкт `simpledialog`. Користувач може ввести запитане значення й натиснути «ОК».

Це поверне введене значення. Якщо користувач натискає `«Відміна»`, то нічого не повертається:

The return value is a Boolean, `True` or False, answer to the question. If `«cancel»` is an option and the user selects the `«cancel»` button, None is returned. Another dialogues, which you might be need to use are `simpledialog` and `filedialog`.

If you want to ask the user for a single data value, either a string, integer, or floating point value, you can use a `simpledialog` object. A user can enter the requested value and hit “OK”.

It will return the entered value. If the user hits `«Cancel»`, then None is returned:

```
import tkinter as tk
from tkinter import simpledialog
application_window = tk.Tk()
answer = simpledialog.askstring("Input", "What is your first
name?",
                                parent=application_window)
if answer is not None:
    print("Your first name is ", answer)
else:
    print("You don't have a first name?")
answer = simpledialog.askinteger("Input", "What is your
age?",
                                 parent=application_window,
                                 minvalue=0, maxvalue=100)
if answer is not None:
    print("Your age is ", answer)
```

```
else:
    print("You don't have an age?")
answer = simpledialog.askfloat("Input", "What is your sala-
ry?",
                               parent=application_window,
                               minvalue=0.0,
maxvalue=100000.0)
if answer is not None:
    print("Your salary is ", answer)
else:
    print("You don't have a salary?")
```

Звичайне завдання – вибрати імена папок і файлів на пристрої зберігання. Це можна зробити з допомогою об'єкта `filedialog`. Зверніть увагу на те, що ці команди не зберігають і не завантажують файл. Вони просто дають змогу користувачеві вибрати файл. Коли у вас є ім'я файлу, ви можете відкрити, обробити і закрити файл, використовуючи відповідний код ***Python***. Ці діалогові вікна завжди повертають вам «`повне ім'я файлу`», що містить повний шлях до файлу.

Якщо користувачеві дозволено вибирати декілька файлів, то повертане значення є кортежем, який містить усі вибрані файли. Якщо користувач відміняє діалогове вікно, повертане значення є порожнім рядком:

A common task is to select the names of folders and files on a storage device. This can be accomplished using a `filedialog` object. Note that these commands do not save or load a file. They simply allow a user to select a file. Once you have the file name, you can open, process, and close the file using appropriate ***Python*** code. These dialog boxes always return you a `«fully qualified file name»` that includes a full path to the file.

Also note that if a user is allowed to select multiple files, the return value is a tuple that contains all of the selected files. If a user cancels the dialog box, the returned value is an empty string:

```
import tkinter as tk
from tkinter import filedialog
import os

application_window = tk.Tk()

# Build a list of tuples for each file type the file dialog
should display
my_filetypes = [('all files', '.*'), ('text files', '.txt')]

# Ask the user to select a folder.
answer = filedialog.askdirectory(parent=application_window,
                                 initialdir=os.getcwd(),
```

```
                                    title="Please select a fold-
er:")

# Ask the user to select a single file name.
answer =
filedialog.askopenfilename(parent=application_window,
                                      initialdir=os.getcwd(),
                                      title="Please select a
file:",
                                      filetypes=my_filetypes)

# Ask the user to select a one or more file names.
answer =
filedialog.askopenfilenames(parent=application_window,
                                       initialdir=os.getcwd(),
                                       title="Please select one
or more files:",
                                       filetypes=my_filetypes)

# Ask the user to select a single file name for saving.
answer =
filedialog.asksaveasfilename(parent=application_window,
                                       initialdir=os.getcwd(),
                                       title="Please select a
file name for saving:",
                                       filetypes=my_filetypes)
```

Частина 2. БІБЛІОТЕКИ ДЛЯ РОБОТИ ІЗ ЗОБРАЖЕННЯМИ Й ВІДЕО

5. ВИКОРИСТАННЯ БІБЛІОТЕКИ PILLOW

5.1. Перелік основних модулів пакета Pillow

Як зазначалося раніше, одним з центральних завдань побудови систем технічного зору є оброблення зображень і відеоінформації. Мовою програмування ***Python*** це здійснюється з допомогою бібліотеки ***Pillow***, що підключається.

Для ефективного використання наявних ресурсів розглянемо структуру

Part 2. LIBRARIES FOR WORK WIH ITMAGES AND VIDEO

5. USING THE PILLOW LIBRARY

5.1. The list of the Pillow package main modules

As noted earlier, one of the central problems of the technical vision systems development is image and video information processing. In the ***Python*** programming language, it is implemented by means of the ***Pillow*** plugin library.

For efficient use of available resources, we will consider the

цієї бібліотеки, властивості пропонованих користувачеві функцій і методів. Офіційну документацію щодо бібліотеки ***Pillow*** розміщено за адресою http://pillow/readthedocs.org/.

Розглянемо перелік основних модулів пакета ***Pillow*** для роботи із зображеннями на мові Python. Переклад бібліотеки ***Python Imaging Library (PIL)*** зроблено за офіційною документацією і містить такі найнеобхідніші модулі:

- ***The Image Module*** містить функції, методи і властивості для відкриття, збереження й маніпулювання зображеннями;
- ***The Image Chops Module*** містить багато арифметичних операцій над зображеннями;
- ***The Image Color Module*** містить функції для перетворення рядка визначення кольору на кортеж формату ***RGB***;
- ***The Image Draw Module*** використовується для простої 2D-графіки, для рисування, створення нових зображень і текстів і ретушування наявних зображень;
- ***The Image Grab Module*** містить функції, які допомагають зробити знімок екрана;
- ***The Image Font Module*** містить функціонал для роботи зі шрифтами ***Truetype*** і ***Opentype***.

structure of this library, the properties of functions and methods offered to the user. The official documentation for the ***Pillow*** library is available at http:// pillow /readthedocs. org/.

Let's consider the list of basic modules of the ***Pillow*** package for working with images in the ***Python*** language. The translation of the ***Python Imaging Library*** (***PIL***) is done according to official documentation and contains a number of essential modules.

- ***The Image Module*** contains functions, methods and properties for images opening, saving and manipulating;
- ***The Image Chops Module*** contains many arithmetic operations on images;
- ***The Image Color Module*** contains functions for converting a color definition string into an ***RGB*** tuple;
- ***The Image Draw Module*** is used for drawing simple 2D graphics. Used to draw, create new images, create text and retouch existing images;
- ***The Image Grab Module*** contains functions that help to take a screenshot;
- ***The Image Font Module*** contains functionality for working with ***TrueType*** and ***OpenType*** fonts.

5.2. Особливості використання функцій і методів модулів пакета Pillow

Розглянемо детальніше властивості й синтаксис основних функцій і методів пакета модулів бібліотеки ***Pillow***. Далі буде наведено конкретні приклади кодів, що дають змогу практично реалізувати програмні можливості цієї бібліотеки.

5.2.1. Функції для відкриття, копіювання і збереження файлів зображень

Функція `open()` – функція для відкриття файлу з готовим зображенням, що повертає об'єкт, з допомогою якого виконується подальша робота із зображенням. Якщо відкрити файл із зображенням не вдалося, то у вікні інтерпретатора з'являється виключення `IOError`. Синтаксис функції open() є таким:

```
open((<Шлях або файловий об’єкт>[,mode=’r’])
```

У першому параметрі вказується абсолютний або відносний шлях до зображення. Необов'язковий другий параметр задає режим доступу до файлу − якщо його не вказано, то файл буде доступним тільки для читання.

Відкриємо файл `image_1.jpg`, розташований у поточному робочому каталозі (рис. 5.1):

```
>>> img=Image.open("C:/Users/ Leonid /Desktop/image_1.jpg")
```

Замість цього можна передати файловий об'єкт, відкритий в бінарному режимі.

Приклад:

5.2. Features using the functions and methods of the Pillow package modules

Let us consider in more detail the properties and syntax of the main functions and methods of the ***Pillow*** library modules package. In the next subchapters, specific examples of codes, that allow you to implement practically the software features of this library, will be presented.

5.2.1. Functions for opening, copying and saving image files

`open()` − function is used to open the file with the completed image. It returns the object with which the further work with the image is performed. If the image file could not be opened, an `IOError` exception appears in the interpreter window. The `open()` function syntax is:

```
open((<Path or file object >[,mode=’r’])
```

The first parameter indicates the absolute or relative path to the image. The optional second parameter sets the file access mode − if it is not specified, the file will be read only.

Open the file `image_1.jpg`, located in the current working directory (Fig. 5.1):

Instead, you can transfer a file object opened in binary mode.

Example:

```
# Відкриваємо файл у бінарному режимі (Open the file in binary
mode)
>>> f = open ("C:/Users/Leonid/Desktop/12.jpg", "rb")
>>> img = Image.open(f) # Передаємо об’єкт файлу (Passing the
    file object)
>>> img.size       # Отримуємо розмір зображення (Get image
size)
   (304, 190)
>>> img.format # Виводимо формат зображення (Display image for-
mat)
    ‘JPG’
  >>> f.close()           # Закриваємо файл (Close the file)
```

З прикладу ясно, що формат зображення визначається автоматично. Після відкриття файлу за допомогою функції `open()` зображення не завантажується відразу в пам'ять – завантаження відбувається при першій операції із зображенням.

У цьому прикладі для перегляду зображення використано метод `show()`. Будемо ним користуватися й надалі.

`show()` – метод, який створює тимчасовий файл у форматі BMP і запускає програму для перегляду зображень, яка використовується в операційній системі за замовчуванням. Наприклад, це може бути засіб перегляду фотографій у Windows 7.

`load` – метод, що дає змогу завантажити зображення явно і повертає об'єкт, з допомогою якого можна отримати доступ до окремих пікселів зображення.

Указавши всередині квадратних дужок два значення − горизонтальну й вертикальну координати пікселя, можна отримати або задати його колір.

Приклад:

From the example it is clear that the image format is determined automatically. After opening the file using the `open()` function, the image itself is not loaded immediately into memory − the download is performed during the first operation with the image.

In this example, the `show()` method is used to view the image. We will use it in the future.

`show()` is a method that creates a temporary file in BMP format and runs the program for viewing images used in the default operating system. For example, it may be a photo viewer in Windows 7.

`load()` is a method that allows you to load an image explicitly. It returns an object with which you can access individual pixels of the image.

By specifying two values inside the square brackets: the horizontal and vertical coordinates of the pixel, you can get or set its color.

Example:

Рис. 5.1. Приклад відкриття файлу з готовим зображенням

Fig. 5.1. An example of a file opening with created image

```
img=Image.open("foto.jpg")
>>> obj = img.load()
# Отримуємо колір пікселя (Get pixel color)
>>> obj[25,45]
(122, 86, 62)
# Задаємо колір пікселя (червоний)(Set pixel color (red))
>>> obj[25,45] = (255, 0, 0)
```

Для доступу до окремого пікселя замість методу `load()` можна використовувати методи `getpixel` і `putpixel()`. Метод `getpixel (<Координати>)` дає змогу отримати колір зазначеного пікселя, а метод `putpixel (<Координати "," Колір>)` змінює колір пікселя. Координати пікселя вказуються у вигляді кортежу з двох елементів. Зауважимо, що ці методи працюють повільніше, ніж метод `load()`. Наведемо приклад використання методів `getpixel` і `putpixel()`:

For access to the separate pixel instead of method of `load()` it is possible to use the methods of `getpixel()` and `putpixel()`. The method of `getpixel (<Coordinates>)` allows to get the color of the indicated pixel, and the method of `putpixel (<Coordinates>, <Color>)` decolorizes pixel. The coordinates of pixel are specified as a tuple from two elements. We will notice that these methods work slower than method of `load()`. We will give an example of the use of methods of `getpixel()` and `putpixel()`:

```
>>> img=Image.open("foto.jpg")
# Отримуємо колір пікселя (Get pixel color)
>>> img.getpixel((25,45))
(122, 86, 62)
# Змінюємо колір пікселя (Change pixel color)
>>> img.putpixel((25,45), (255, 0, 0))
# Отримуємо колір пікселя (Get pixel color)
(255, 0, 0)
# Переглядаємо зображення (View image)
>>> img.show()
```

copy() – метод, який створює копію зображення (рис. 5.2).

Приклад:

copy() is a method that creates a copy of an image (Fig. 5.2).

Example:

```
>>> from PIL import Image
#   Відкриваємо файл (Open the file)
>>> img=Image.open("C:/Users/leonid/Desktop/image_2.jpg")
>>> img.show()          # Переглядаємо оригінал (View the origi-
nal)
>>> img1 = img.copy()  # Створюємо копію (Create a copy)
>>> img1.show()        # Переглядаємо копію (View a copy)
```

а

б

Рис. 5.2. Створення копії зображення

Fig. 5.2. An image copy creation

`Thumbnail(<Розмір>[,<Фільтр>])` – метод, що створює зменшену версію зображення вказаного розміру – мініатюру. Розмір задається у вигляді кортежу з двох елементів: `(<Ширина,<Висота>)`.

Розмір змінюється пропорційно, тобто за основу береться мінімальне значення, а друге значення обчислюється пропорційно першому.

The `Thumbnail(<Size> [, <Filter>])` method creates reduced image version with the given size - thumbnail. The size is set as a tuple from two elements: `(<Weight> <Height>)`.

The change of size is produced proportionally of a minimum value takes up basis, and the second value is calculated proportionally first.

У параметрі <Фільтр> можуть бути вказані фільтри NEAREST, BILINEAR, BICUBIC або LANCZOS. Якщо параметр не вказано, використовується значення BICUBIC. Метод змінює зображення і нічого не повертає (рис. 5.3).

In a parameter <Filter> the filters of NEAREST, BILINEAR, BICUBIC or LANCZOS, can be indicated. If a parameter is missed, a value BICUBIC is used. A method changes an image and returns nothing (Fig. 5.3)

Приклад:

Example:

```
>>> from PIL import Image
>>> img=Image.open("C:/Users/leonid/Desktop/image_3.jpg")
>>> img.size # Початкові розміри зображення
              (Initial size of image)
(297, 170)
>>> img.show()
>>>img1 = img.thumbnail((148,85),Image.LANCZOS)
>>> img.size # Змінені розміри (Changed size)
(148, 85)
>>> img.show()
```

а

б

Рис. 5.3. Створення зменшеної мініатюри зображення: *а* − оригінал, *б* – мініатюра

Fig. 5.3. Image thumbnail creation: *a* − original, *b* – miniature

save() – метод, призначений для збереження зображення у файл, що має такий формат і синтаксис:

The **save()** method is used to save an image to a file. It has the following format and syntax:

```
save (<Шлях або файловий об'єкт> [,<формат> [,<Опції>]])
```

У першому параметрі вказується абсолютний або відносний шлях. Замість шляху можна передати файловий об'єкт, відкритий у бінарному режимі. Для прикладу збережемо зображення у форматах JPEG і BMP різними способами:

The first parameter indicates the absolute or relative path. Instead of the path, you can transfer a file object opened in binary mode. For example, let's save the image in JPEG and BMP formats in different ways:

```
>>> from PIL import Image
>>> img = Image.open ("C:/Users/leonid/Desktop/12.jpg")
>>> img.save("tmp.jpg")                    # Збереження у форматі
```

```
JPEG
>>> img.save("tmp.bmp","BMP")          # Збереження у форматі
BMP
>>> f = open("tmp1.bmp","wb")
>>> img.save(f,"BMP")                  # Передаємо файловий
об'єкт
>>> f.close()
```

Зазначимо, що файл може бути відкритий в одному форматі, а збережений в іншому. Коли параметр `<формат>` не задано, формат зображення визначається за розширенням файлу, проте, якщо методу `save()` як перший параметр передано файловий об'єкт, формат має бути вказаний обов'язково.

У параметрі `<Опції>` можна передати додаткові опції. Підтримувані опції залежать від формату зображення. Наприклад, за замовчуванням зображення `JPEG` зберігаються з якістю 75. За допомогою опції `quality` можна вказати інше значення в діапазоні від 1 до 100:

Note that the file can be opened in one format, and saved in another. When the `<format>` parameter is not specified, the image format is determined by the file extension, however, if the file object is passed to the `save()` method, the format must be specified.

In the option `<Options>` you can pass additional options. The supported options depend on the image format. For example, by default, `JPEG` images are saved with a quality of 75. With the `quality` option, you can specify a different value in the range from 1 to 100:

```
# Указання якості (Quality indication
>>> img.save("tmp2.jpg","JPEG", quality=100)
```

5.2.2. Створення нового зображення

Бібліотека ***Pillow*** дає змогу працювати не лише з готовими зображеннями, але й створювати нові. Розглянемо таку можливість детальніше.

Функция **`new()`** для створення нового зображення має такий формат:

5.2.2. Creating a new image

The ***Pillow*** library allows you to work not only with ready-made images, but also create new ones. Consider this possibility in more detail.

`new()` – the function for creating a new image has the following format:

```
new(<Режим>, <Розмір> [,(<Колір фону>])
```

У параметрі `<Режим>` указується один з таких режимів:

In the `<Mode>` parameter, one of the following modes is specified:

- `1`– 1 біт, чорно-біле (1 bit, black and white);
- `L` – 8 бітів, чорно-біле (8 bit, black and white);
- `P` – 8 бітів, кольорове (256 кольорів) − 8 bits, color (256 colors);
- `RGB` – 24 біти, кольорове (24 bits, color);
- `RGBA` – 32 біти, кольорове з альфа-каналом `(32 bits, color with alpha channel);`
- `CMYK` – 32 біти, кольорове (32 bits, color);
- `YCbCr` – 24 біти, кольорове, відеоформат (24 bits, color, video format);
- `Lab` – 24 біти, кольорове, використовується колірний простір `Lab` (24 bits, color, Lab color space used);
- `HSV` – 24 біти, кольорове, використовується колірний простір `HSV` (24 bits, color, uses HSV color space);
- `I` – 32 біти, кольорове, кольори кодуються цілими числами (32 bits, color, colors are encoded by integers);
- `F` – 32 біти, кольорове, кольори кодуються дійсними числами (32 bits, color, colors are coded with real numbers).

У другому параметрі необхідно передати розмір створюваного зображення (полотна) у вигляді кортежу з двох елементів `<Ширина>`, `<Висота>`. У необов'язковому параметрі `<Колір фону>` задається колір фону. Якщо параметр не вказано, то фон буде чорного кольору. Для режиму RGB колір указується у вигляді кортежу з трьох цифр від 0 до 255 `(<Частка червоного>, <Частка зеленого>, <Частка синього>)`.

Крім того, можна вказати назву кольору англійською мовою і рядка у форматах `"#rgb"` і `"#rrggbb"`. Різні способи вказування кольору:

In the second parameter, you must pass the size of the created image (canvas) in the form of a tuple of two elements `<Width>`, `<Height>`. In the optional parameter `<Background color>` the background color is set. If the parameter is not specified, the background will be black. For the ***RGB*** mode, the color is indicated as a three-digit tuple from 0 to 255 `(<Proportion of red>, <Proportion of green>, <Proportion of blue>)`. In addition, you can specify the name of the color in English and strings in the `"#RGB"` and `"#RRGGBB"`. Various ways to specify colors are listed below:

```
>>> img = Image.new("RGB",(100,100))
>>> img.show()                    # Чорний квадрат (Black
square)
>>> img = Image.new("RGB",(100,100),(255,0,0))
>>> img.show()                    # Червоний квадрат (Red
square)
>>> img = Image.new("RGB",(100,100),("green"))
```

```
>>> img.show()                          # Зелений квадрат (Green
square)
>>> img = Image.new("RGB",(100,100),("#f00")
>>> img.show()                          # Червоний квадрат (Red
square)
>>> img = Image.new("RGB",(100,100),("#ff0000")
>>> img.show()                               # Червоний квадрат (Red
square)
```

5.2.3. Отримання інформації про зображення

Отримати інформацію про зображення дають змогу такі атрибути об'єкта зображення:

- `size` – розмір зображення у вигляді кортежу з двох елементів: `<Ширина>, <Висота>`;
- `format` – формат зображення у вигляді рядка (наприклад, `'gif'`, `'gpeg'` і т. д.);
- `mode` – режим у вигляді рядка (наприклад `'P '`, `'RGB'` і т. д.);
- `info` – додаткова інформація про зображення у вигляді словника.

Наведемо приклад виведення інформації про зображення у форматах `JIPEG`, `GIF`, `BMP`, `TIFF` і `PNG`:

5.2.3. Getting image information

The following attributes of the image object allow getting information about the image:

- `size` – the size of the image as a tuple of two elements: `<Width>, <Height>`;
- `format` – image format as a string (for example, `'GIF'`, `'GPEG'`, etc.);
- `mode` – mode as a string (for example, `'P'`, `'RGB'`, etc.);
- `info` – additional information about the image in the form of a dictionary.

Let's give an example of displaying information about images in the formats `JIPEG`, `GIF`, `BMP`, `TIFF` and `PNG`:

```
>>> img=Image.open("foto.jpg")
>>> img.size, img.format, img.mode
((800, 600),'JPEG','RGB')
>>> img.info
{'jfif':258, 'jfif_unit':0, 'adobe':100, 'progression':1,
'jfif_version':(1,2),'adobe_transform':100,'jfif'_density(100,1
00)
>>> img=Image.open("foto.gif")
>>> img.size, img.format, img.mode
((800, 600),'GIF','P')
>>> img.info
{'version': 'GIF89a,'background':254}
```

5.2.4. Перетворення зображень

Наведемо приклади використання основних методів перетворення зображень.

`resize(<Розмір>[,<Фільтр>])` – метод, який змінює розмір зображення і на відміну від методу `Thumbnail()` повертає нове зображення, а не змінює початкове.

Розмір змінюється не пропорційно, і якщо пропорції не дотримані, то зображення буде спотворено. У параметрі `<Фільтр>` можуть бути вказані фільтри `NEAREST`, `BILINEAR`, `BICUBIC` або `LANCZOS`. Якщо параметр не вказано, то використовується значення `NEAREST` (рис. 5.4).

Приклад:

5.2.4. Image conversion

Let us give examples of the use of basic image conversion methods

The `resize (<Size> [, <Filter>])` method resizes the image. Unlike the `Thumbnail ()` method, it returns a new image, but does not change the original one.

Resizing is not proportional, and if the proportions are not met, the image will be distorted. The `NEAREST`, `BILINEAR`, `BICUBIC` or `LANCZOS` filters may be specified in the `<Filter>` parameter. If not specified, the value is `NEAREST` (Fig. 5.4).

Example:

```
>>> from PIL import Image
>>> img=Image.open("C:/Users/Leonid/Desktop/image_5.jpg")
>>> img.size    # Початкові розміри зображення (Original image
    sizes)
(225, 225)
>>> img1=img.resize((450,450),Image. LANCZOS)
>>> img1.size   # Пропорційне збільшення (Proportional in-
    crease)
(450, 450)
>>> img.show()
>>> img1.show()
```

`rotate(<Кут>[,<Фільтр>][,<expand=0])` – метод, який повертає зображення на вказаний кут, що відмірюється в градусах проти годинникової стрілки. У параметрі `<Фільтр>` можуть бути вказані фільтри `NEAREST`, `BILINEAR` або `BICUBIC`. Якщо параметр не вказано, то використовується значення `NEAREST`.

Якщо параметр `expand` має значення `True`, то розмір зображення

`rotate (<Angle> [, <Filter>] [, <expand = 0>])` – rotates the image to the specified angle, measured in degrees counterclockwise. In the `<Filter>` parameter, `NEAREST`, `BILINEAR` or `BICUBIC` filters can be specified. If not specified, the value is `NEAREST`.

If the expand parameter is True, then the image size will be increased so that it fits completely:

буде збільшено так, щоб воно повністю помістилося: за замовчуванням розмір зображення зберігається, а якщо зображення не поміщається, то його буде обрізано (рис. 5.5).

by default, the image size is preserved, and if the image does not fit, it will be cropped (Fig. 5.5).

а

б

Рис. 5.4. Зміна розміру зображення

Fig. 5.4. Image resizing

Приклад:

Example:

```
>>> from PIL import Image
>>> img=Image.open("C:/Users/Ltonid/Desktop/image_6.jpg")
>>> img.size
(275, 183)
>>> img.show()
>>> img1=img.rotate(45,Image.NEAREST,expand=True)
>>> img1.size
(325, 325)
>>> img1.show()
```

а

б

Рис. 5.5. Поворот зображення

Fig. 5.5. Image rotation

`Transpose(<Перетворення>)` – метод, який повертає дзеркальний образ або зображення. Як параметр

The `Transpose (<Transform>)` method returns a mirror image or rotates an image. As a parameter,

вказують значення FLIP_LEFT_RIGHT,LIP_TOP_BOTTOM,ROTATE_90, ROTATE_180, ROTATE_270 або TRANSPOSE. Метод повертає нове зображення (рис. 5.6).

the values FLIP_LEFT_RIGHT, FLIP_TOP_ BOTTOM, ROTATE_90, ROTATE_180,ROTATE_270 or TRANSPOSE are indicated. The method returns a new image (Fig. 5.6).

Приклад:

Example:

```
>>> from PIL import Image
>>> img=Image.open("C:/Users/Leonid/Desktop/image_2.jpg")
>>> img.show()   # Початкове зображення (Source image)
>>> img1 = img.transpose(Image.FLIP_LEFT_RIGHT)
>>> img1.show()  # Горизонтальний дзеркальний образ (Horizontal
    mirror image)
>>> img2 = img.transpose(Image.FLIP_TOP_BOTTOM)
>>> img2.show()  # Вертикальний дзеркальный образ (Vertical
    mirror image)
>>> img3 = img.transpose(Image.ROTATE_90)
>>> img3.show()  # Поворот на 90 градусів проти годинникової
    стрілки (Rotate 90 degrees counterclockwise
>>> img4 = img.transpose(Image.ROTATE_180)
>>> img4.show() # Поворот на 180 градусів (180 degree rotation)
```

crop((<X1>,<Y1>,<X2>,<Y2>)) – метод, який вирізає прямокутний фрагмент з вихідного зображення. Як параметр указується кортеж з чотирьох елементів: перші два елементи задають координату лівого верхнього кута вирізуваного фрагмента, а інші два елементи задають координати його нижнього правого кута. Передбачається, що початок координат розташовується у верхньому лівому куті зображення. Додатна вісь **x** напрямлена вправо, а додатна вісь **y** – униз. Як значення метод повертає нове зображення. Зчитування області з вихідного зображення проводиться тільки при першій операції над новим зображенням.

Якщо після виконання методу crop() над вихідним зображенням було проведено операції, то вони відобразяться і на новому зображенні.

The crop((<X1>,<Y1>,<X2 Y2>)) method cuts a rectangular fragment from the source image. A tuple of four elements is indicated as a parameter: the first two elements specify the coordinate of the upper left corner of the section to be cut, and the second two elements specify the coordinates of its lower right corner. It is assumed that the origin is located in the upper left corner of the image. The positive x axis is directed to the right, and the positive y axis is down. The method returns a new image. The area is read from the original image only at the first operation on the new image.

If after the crop() method has been performed on the original image, operations have been performed, they will also be displayed

Щоб явно зробити зчитування області, необхідно відразу після методу `crop()` викликати метод `load()` (рис. 5.7).

on the new image. In order to explicitly read the region, it is necessary immediately after the `crop()` method to call the `load()` method (Fig. 5.7).

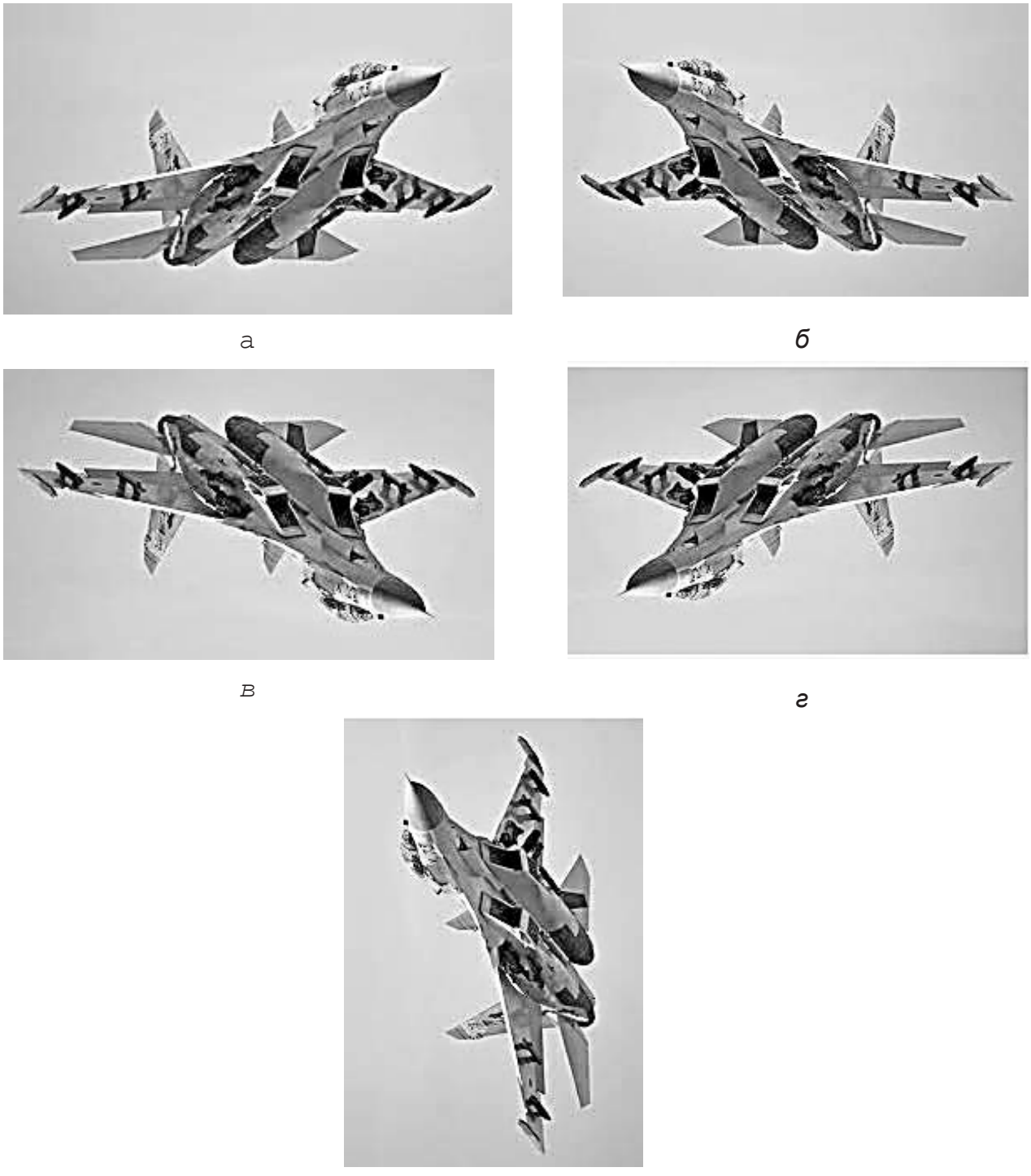


Рис. 5.6. Дзеркальні відображення й повороти зображення

Fig. 5.6. Mirror reflections and image rotations

Приклад:

```
>>> from PIL import Image
>>> img=Image.open("C:/Users/Leonid/Desktop/image_4.jpg")
# Виділяємо фрагмент (Select the fragment)
>>> img1 = img.crop( [0, 0, 100, 100])
# Прочитуємо фрагмент, створюючи нове зображення
# We read a fragment, creating a new image
>>> img1.load()
# Визначаємо розміри нового зображення
# Determine the size of the new image
>>> img1.size
(100,100)
>>> img1.show()
```

Example:

а

б

Рис. 5.7. Виділення прямокутного фрагмента зображення

Fig. 5.7. Rectangular image fragment separation

`paste(<Колір>,<Область>[,<Маска>)` – метод, який зафарбовує прямокутну область певним кольором. Координати області вказуються у вигляді кортежу з чотирьох елементів: перші два елементи задають координату лівого верхнього кута зафарбованої області, а інші два елементи – координату її правого нижнього кута. Зафарбуємо цю область червоним кольором (рис. 5.8):

The paste (<Color>, <Area> [,<Mask>) method fill a rectangular area with a certain color. The coordinates of area are specified as a tuple from four elements: the first two elements set the coordinate of the left overhead corner of the painted out area, and second two elements – coordinate of her right lower corner. We will paint out this area a red color (Fig. 5.8):

```
>>> from PIL import Image
>>> img=Image.open("C:/Users/Leonid/Desktop/image_4.jpg")
>>> img.paste((255, 0, 0), (0, 0. 100, 100))
>>> img.show
```

Рис. 5.8. Зафарбовування області певним кольором

Fig. 5.8. Painting out of rectangular area a certain color

Тепер зафарбуємо все зображення зеленим кольором (рис. 5.9):

Now we will paint out all image a green color (Fig. 5.9):

```
>>> from PIL import Image
>>> img=Image.open("C:/Users/Leonid/Desktop/image_4.jpg")
>>> img.paste((0,128,0),img.getbbox())
>>> img.getbbox()
(0, 0, 275, 183)
>>> img.show()
```

Рис. 5.9. Зафарбовування всього зображення певним кольором

Fig. 5.9. Whole image filling with a certain color

У цьому прикладі використано метод `getbbox()`, який повертає координати прямокутної області, у яку вписується все зображення.

`paste(<Зображення>,<Область>[,<Маска>])` – метод, який вставляє вказане зображення в прямокутну область. Координати області вказуються у вигляді кортежу з двох або чотирьох елементів – якщо вказано кортеж з двох елементів, він задає початкову точку цієї області. Для прикладу завантажимо зображення, створимо його зменшену копію, а потім вставимо її у вихідне зображення. Навколо вставленого зображення зобразимо рамку червоного кольору (рис. 5.10):

In this instance the method `getbbox()`, is used that returns the coordinates of rectangular area into that all image is written.

The `paste(<Image>, <Area>[,<Mask>])` method inserts the indicated image in a rectangular area. The coordinates of area are specified as a tuple from two or four elements - if a tuple is indicated from two elements, he sets the initial point of this area. For an example we will load an image, will create his diminished copy, and then we will insert her in an initial image. Round a cutting-in image we will represent the scope of red color (Fig. 5.10):

```
>>> from PIL import Image
>>> img = Image.open("C:/Users/Leonid/Desktop/image_4.jpg")
>>> img.size
>>> img1 = img.resize((100,75)) # Створюємо мініатюру (We cre-
    ate a miniature)
>>> img1.size
(200, 150)
>>> img.show()
>>> img.paste((255, 0, 0), (9, 9, 111, 86))   # Рамка (Scope)
>>> img.paste(img1,(10,10))   # Вставляємо мініатюру (We insert
    a miniature)
>>> img.show()
```

а

б

Рис. 5.10. Вставляння зображення в прямокутну область

Fig. 5.10. Given image insertion in a rectangular area

Необов'язковий параметр <Маска> дає змогу задати ступінь прозорості вставлюваного зображення або кольору. Для прикладу виведемо напівпрозору білу горизонтальну смугу заввишки 100 пікселів (рис. 5.11):

Optional parameter <Mask> the degree of transparency of the inserted image or color allows to set. For an example we will show out a semi-transparent white horizontal stripe in 100 pixels high (Fig. 5.11):

```
>>> from PIL import Image
>>> img = Image.open("C:/Users/Leonid/Desktop/image_6.jpg")
>>> img.show()
>>> white = Image.new("RGB",(img.size[0],100),(255,255,255))
>>> mask = Image.new("L",(img.size[0],100),64) # Маска (Mask)
>>> img.paste(white, (0,0), mask)
>>> img.show()
```

а

б

Рис. 5.11. Накладення напівпрозорої білої горизонтальної смуги

Fig. 5.11. Semi-transparent white horizontal stripe layout

`split()` – метод, який повертає канали зображення у вигляді кортежу. Наприклад, для зображення в режимі `RGB` повертається кортеж з трьох елементів: `(R,G,B)`. Зробити протилежну операцію (відновити зображення з каналів) дає змогу метод `merge(<Режим>, <Канали>)`. Наведемо приклад перетворення зображення з режиму `RGB` на режим `RGBA` (рис. 5.12):

The `split()` method returns the channels of image as a tuple. For example, for an image back into the mode of RGB a tuple goes from three elements: `(R,G,B)`. The method of `merge(<Mode>, <Channels>)` allows to produce a reverse operation (to redisplay from channels). We will give an example of transformation of image from the mode of `RGB` in the mode `RGBA` (Fig. 5.12):

```
>>> from PIL import Image
>>> img=Image.open("C:/Users/Leonid/Desktop/image_8.jpg")
>>> img.mode
'RGB'
>>> R,G,B = img.split()
>>> mask = Image.new("L",img.size, 128)
>>> img1 = Image.merge("RGBA",(R,G,B,mask))
```

```
>>> img1.mode
'RGBA'
>>> img1.show()
```

Рис. 5.12. Перетворення режиму RGB на режим RGBA

Fig. 5.12. Transformation of the mode of RGB to the mode of RGBA

Convert – convert(<Новий режим>[,<Матриця>[, <Режим змішування кольорів> [, <Палітра> [,<Кількість кольорів>]]]]. Цей метод перетворює зображення в зазначений режим і повертає нове зображення. Третій параметр вказує спосіб отримання складних кольорів з простіших шляхом змішування і має сенс при перетворенні зображень форматом RGB або L на формат P або 1 – доступні значення None (змішування не виконується) і Image.FLOYD-STEINBERG (значення за замовчуванням). Четвертий параметр задає тип палітри при перетворенні з RGB на P: Image.WEB (Web-сумісна палітра) або Image.ADAPTIVE (адаптивна палітра). П'ятий параметр задає кількість кольорів у палітрі, за замовчуванням – 256. Перетворимо зображення з формату RGB на формат RGBA (рис. 5.13):

convert(<New mode> [,<Matrix> [,<Mode of mixing of colors>[,<Palette> [, <Amount of colors>]]]]. This method will transform an image in the indicated mode and returns a new image. The third parameter specifies the method of receipt of difficult colors from more simple by mixing and makes sense at transformation of images of format of RGB or L to the format of P or 1 – accessible value None (mixing is not executed) and Image.FLOYDSTEINBERG (default value). A fourth parameter sets the type of palette at transformation from RGB in P: Image.WEB (Web- a compatible palette is a default value) or Image. ADAPTIVE (adaptive palette). A fifth parameter sets the amount of colors in a palette, by default - 256. We will transform an image from the format of RGB in the mode of RGBA (Fig. 5.13):

```
>>> from PIL import Image
>>> img=Image.open("C:/Users/Leonid/Desktop/image_12.jpg")
>>> img.mode
'RGB'
>>> img1 = img.convert("RGBA")
>>> img1.mode
'RGBA'
>>> img1.show()
```

Рис. 5.13. Перетворення зображення RGB на зображення в RGBA

Fig. 5.13. Image transformation from RGB to RGBA

Перетворимо зображення RGB в формат P, указавши змішування кольорів та адаптивну палітру зі 128 кольорів:

Let's transform, for example, the image of RGB in the format of P, specifying mixing of colors and adaptive palette in 128 colors:

```
>>> from PIL import Image
>>> img=Image.open("C:/Users/Леонід/Desktop/image_12.jpg")
>>> img.mode
'RGB'
>>> img1=img.convert("P", None, Image.FLOYDSTEINBERG, Im-
    age.ADAPTIVE, 128)
>>> img1.mode
'P'
>>> img1.show()
```

Filter (<Фільтр>) – метод, що застосовує до зображення вказаний фільтр. Метод повертає нове зображення. Як приклад можна навести фільтри BLUR, CONTOUR, DETAIL,EDGE_ENHANCE,EDGE_ENHANCE_MORE,EMBOSS,FIND_EDGES,SHARPEN,SMOOTH, і SMOOTH_ MORE з модуля ImageFilter. Пропонований набір фільтрів виконує різного роду операції. Наведемо приклад різних операцій фільтрації (рис. 5.14):

Filter (<Filter>) – applies the indicated filter to the image. A method returns a new image. It is as an example possible to bring filters over of BLUR, CONTOUR,DETAIL,EDGE_ENHANCE, EDGE_ENHANCE_MORE, EMBOSS, FIND_EDGES,SHARPEN, SMOOTH, and SMOOTH_MORE from the module of ImageFilter. The offered set of filters executes different sort of operation. We will give an example of different operations of filtration (Fig. 5.14):

```
>>> from PIL import Image
>>> from PIL import ImageFilter
>>> img=Image.open("C:/Users/Leonid/Desktop/image_2.jpg")
>>> img.show()     # Початкове зображення (Initial image)
>>> img1=img.filter(ImageFilter.BLUR)
>>> img1.show()    # відфільтроване зображення (Filtered image)
>>> img2=img.filter(ImageFilter.CONTOUR)
>>> img2.show()
>>> img3=img.filter(ImageFilter.DETAIL)
>>> img3.show()
>>> img4=img.filter(ImageFilter.EDGE_ENHANCE)
>>> img4.show()
>>> img5=img.filter(ImageFilter.EDGE_ENHANCE_MORE)
>>> img5.show()
>>> img6=img.filter(ImageFilter.EMBOSS)
>>> img6.show()
>>> img7=img.filter(ImageFilter.FIND_EDGES)
>>> img7.show()
>>> img8=img.filter(ImageFilter.SHARPEN)
>>> img8.show()
>>> img9=img.filter(ImageFilter.SMOOTH)
>>> img9.show()
>>> img10=img.filter(ImageFilter.SMOOTH_MORE)
>>> img10.show()
```

а

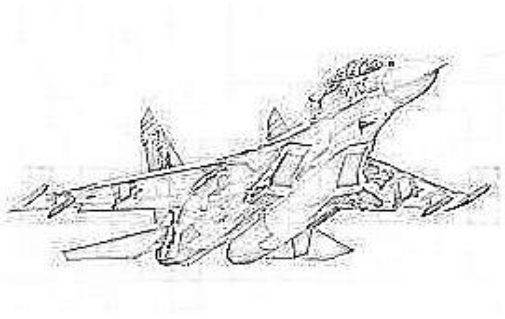
б

Рис. 5.14. Варіанти фільтрації зображень:

а – початкове зображення, *б* – BLUR, *в* – CONTOUR, *г* – DETAIL, *д* – EDGE_ENHANCE, *е* – EDGE_ EN-HANCE_MORE, ж – EMBOSS, *и* – FIND_EDGES, *к* – SHARPEN, *л* – SMOOTH, *м* – SMOOTH_MORE

Fig. 5.14. Image filtering options:

а – original image, *б* – BLUR, *в* – CON-TOUR, *г* – DETAIL, *д* – EDGE_ENHANCE, *е* – EDGE_ENHANCE_ MORE, *ж* – EMBOSS, *и* – FIND_ EDGES, *к* – SHARPEN, *л* – SMOOTH, *м* – SMOOTH_MORE

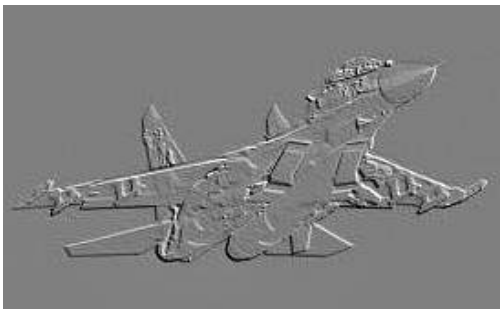


в

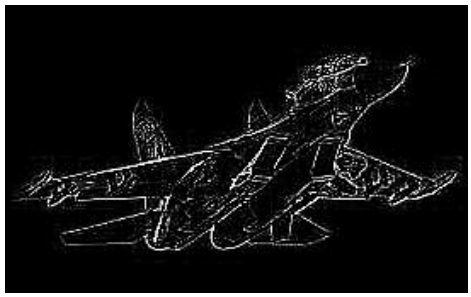
г

д

е



ж



и

к

л

м

Рис. 5.14. Закінчення

Fig. 5.14. End

Наведені приклади не охоплюють усі види перетворень зображень, які можна здійснити з допомогою бібліотеки ***Pillow***, однак формують загальні уявлення про методи й засоби оброблення графічної інформації мовою ***Python***. Подальше використання цих методів можна виконати самостійно.

6. АЛГОРИТМИ OPENCV ДЛЯ ОБРОБЛЕННЯ ЗОБРАЖЕНЬ У PYTHON

Розглянемо далі основні ресурси, які надає бібліотека ***OpenCV*** для оброблення зображень при спільній роботі з ***Python***. Для цього наведемо кілька прикладів, що ілюструють можливості бібліотеки, а також особливості синтаксису її функцій при написанні програмних кодів для перетворення зображень.

6.1. Уведення й візуалізація зображень і відеоданих

Уведення й візуалізація зображення. Щоб почати роботу з зображенням, його необхідно ввести в програму й відобразити для візуального перегляду. Для цього слід написати короткий код:

```
>>> import cv2 as cv
>>> img = cv.imread('1.jpg')
>>> cv.imshow("RGB",img)
```

Зверніть увагу на те, що в першому рядку коду проводиться імпорт ресурсів бібліотеки ***OpenCV***. Ця операція буде успішно виконана, якщо перед початком роботи на комп'ютері здійснено підключення бібліотеки, методику якого докладно описано в першому розділі цієї роботи.

These examples do not cover all types of image transformations that can be done using the ***Pillow*** library, but form a general idea of the methods and means of processing graphical information in ***Python***.

Further use of these methods can be performed independently.

6. OPENCV ALGORITHMS FOR IMAGE PROCESSING IN PYTHON

Consider the main resources that the ***OpenCV*** library provides for image processing when working with ***Python***. To do this, we present a number of examples illustrating the capabilities of the library, as well as the features of the syntax of its functions when writing program codes for image conversion.

6.1. Input and visualization of images and video data

Image input and visualization. To start working with the image, it must be entered into the program and displayed for visual viewing. To do this, you need to write a short code:

Please note that the first line of the code imports the resources of the ***OpenCV*** library. This operation will be successfully completed if, before starting work on a computer, a library is connected, the methodology of which is described in detail in the first section of this work.

Крім того, потрібне вам зображення необхідно помістити в системну папку ***Python***. Це спрощує процедуру введення (не треба прописувати шлях до файлу зображення) і запобігає виникненню синтаксичних помилок. Слід нагадати, що обов'язково має бути вказаний формат зображення, у нашому прикладі – `jpg` (рис. 6.1).

Підключення відеокамери. У практиці використання систем технічного зору застосовуються два основні режими роботи з відеоданими;

- уведення й оброблення відео в реальному масштабі часу безпосередньо від відеореєстратора;
- аналіз відеоданих з раніше записаного файлу.

Перший режим зазвичай використовують для вирішення завдання керування рухомими об'єктами на основі аналізу даних відеоспостереження сцени, а другий − для перегляду подій, зареєстрованих, наприклад, у системах охорони об'єктів або при аналізі наукової відеоінформації.

In addition, you need to place the image you need into the ***Python*** system folder. This simplifies the input procedure (it is not necessary to prescribe the path to the image file) and prevents the occurrence of syntax errors. It should be recalled that the image format must be specified. In our example − `jpg` (Fig. 3.1).

Connecting of video camera. In practice of the use of the systems of technical sight two basic office hours are used with videoinformation:

- input and treatment of video real-time directly from a video recorder.
- analysis of video information from a before writtenin file.

The first mode is usually used for a decision by the tasks of management mobile objects on the basis of analysis of data of video supervision of the stage, and second for viewing of events, registered, for example, in the systems of guard of objects or at the analysis of scientific video information.

Рис. 6.1. Приклад уведення й виведення зображення

Fig. 6.1. Example of input and show of image

Наведемо приклад програмного коду мовою ***Python*** з використанням функції ***OpenCV*** `cv.VideoCapture ()` для підключення камери (рис. 6.2):

Here is an example of software code in ***Python*** using the ***OpenCV*** function `cv.VideoCapture()` (Fig. 6.2):

```
# Підключення бібліотеки OpenCV (# Connecting library OpenCV)
import cv2 as cv
# Створення пристрою захоплення (# Creation device of capture)
cap = cv.VideoCapture(0)
# Цикл для перебору кадрів (# Cycle for brute force)
while True:
# Читання поточного кадру (Reading current frame)
    ret, img = cap.read()
# Відображення поточного кадру (Display current frame)
    cv.imshow("frame",img)
    if cv.waitKey(1) & 0xFF == ord('q'):
    ret = True, otherwise False
        break
# Закриття пристрою захоплення (# Closing device capture)
cap.release()
# Видалення всіх створених вікон
cv.destroyAllWindows()
```

Зазначимо, що для використання однієї з декількох камер необхідно в функції введення вказати цілочисловий аргумент. Якщо камера єдина, то аргумент слід вибирати таким, що дорівнює нулю – `cv.Video Capture (0)`. На рис. 6.2 показано приклад підключення web-камери при використанні Notebook.

Note that in order to use one of several cameras, it is necessary to specify an integer argument in the input function. If the camera is unique, the argument should be set to zero – `cv.VideoCapture(0)`. In Fig. 6.2 an example of connecting a webcam when using Notebook is given.

Рис. 6.2. Підключення web-камери

Fig. 6.2. Connecting a webcam

Читання відеоданих з файлу. Опишемо процедуру введення відеоданих з готового файлу. Як і у випадках уведення зображень, потрібний вам файл відео необхідно помістити в системну папку Python. Це спрощує процедуру введення (не треба прописувати шлях до файлу) та запобігає виникненню синтаксичних помилок. Нагадуємо, що обов'язково має бути вказаний формат відео, у нашому прикладі – `avi`.

Нижче наведено код уведення відеоданих з файлу за допомогою функції `cap = cv.VideoCapture ('filename')`, а на рис. 6.3 показано один із фреймів уведеної відеопослідовності. Зазначимо, що в програмі введення відеоданих передбачено процедуру переривання після натискання клавіші `'q'`.

Читання відеоданих з файлу. Опишемо процедуру введення відео-даних з готового файлу. Як і у випадках введення зображень, необхідно потрібний Вам файл відео помістити в системну папку ***Python***. Це спрощує процедуру введення (не треба прописувати шлях до файлу) та запобігає виникненню синтаксичних помилок. Нагадує-мо, що обов'язково повинен бути вказаний формат відео. У нашому прикладі – `avi`.

Below is the code for entering video data from a file using the function `cap = cv.VideoCapture ('filename')`, and in Fig. 6.3 shows one of the frames of the entered video sequence. Note that the program of video data input provides an interruption procedure by pressing the `'q'` key.

Приклад:

Example:

```
# Підключення бібліотеки OpenCV (OpenCV Library Connection)
import cv2 as cv
# Створення пристрою захоплення (Creating a capture device)
cap = cv.VideoCapture('V_109.avi')
# Цикл для перебору кадрів (Loop Cycle)
while(cap.isOpened()):
# Читання поточного кадру, якщо файл непошкоджений, то ret =
    True, інакше False
# Reading the current frame, if the file is read then ret =
    True, otherwise False
    ret, frame = cap.read()
    if(ret==False):
# Если ret = False - перервати цикл
# If ret = False - abort the loop
        Break
# Відображення поточного кадру (Display current frame)
    cv.imshow('frame',frame)
# Оброблення натискання клавіші 'q', при натисканні - перервати
    цикл і завершити програму
# Processing keystroke 'q', when pressed - interrupt the cycle
    and end the program
```

```
    if cv.waitKey(1)& 0xFF == ord('q'
# Закриття пристрою захоплення (Closing the capture device)
cap.release()
# Видалення всіх створених вікон (Destroy all windows created)
cv.destroyAllWindows()
```

Рис. 6.3. Читання відео з файлу

Fig. 6.3. Reading video from file

6.2. Перетворення зображень з допомогою функцій OpenCV

Перетворення RGB-зображення на півтонове. У бібліотеці ***OpenCV*** міститься безліч функцій перетворення типів зображень. Зупинимося на найбільш поширеному – з допомогою функції `cv.COLOR_RGB2GRAY` вихідне кольорове зображення перетворюється на зображення у відтінках сірого. Результат перетворення показано на рис. 6.4.

6.2. Transform images using OpenCV features

Convert RGB image to halftone. The OpenCV library contains many image type conversion functions. Let's look at the most common – using the `cv.COLOR_RGB2GRAY` function, the original color image is converted to an image in shades of gray. The result of the conversion is shown in Fig. 6.4.

Рис. 6.4. Перетворення RGB на відтінки сірого

Fig. 6.4. Convert RGB to grayscale

Перетворення колірних просторів. У системах технічного зору при використанні повнокольорових зображень зазвичай орієнтуються на використання колірного простору ***RGB***. Однак, крім ***RGB***-моделі подання зображень існує багато моделей, як більш придатних для технічних цілей, так і більш пристосованих до людського сприйняття.

Наведемо приклади перетворень з одного простору в інше. Розглянемо основні особливості цих колірних просторів, використовуючи відповідні функції перетворення бібліотеки ***OpenCV***.

Повнокольорове ***RGB***-зображення `(Red, Green, Blue)` – це масив *M*×*N*×*3*, що складається з трьох матриць розміром *M*×*N*.

Ці матриці відповідають трьом колірним компонентам: червоній, зеленій і синій. Три монохромних зображення, що формують єдине ***RGB***-зображення, називають червоною, зеленою і синьою компонентами зображення. Їх подають у форматі `uint8, uint16, single` або `double`. Використовуються такі числові дані для подання колірних компонент:

- `[0,1]` – для форматів `double` і `single`;
- `[0,255]` – для формату `uint8;`
- `[0,65535]` – для формату `uint16.`

Кількість бітів, що використовуються для подання величини кольорового пікселя за всіма складовими ***RGB***, називають глибиною

Converting between color spaces. In vision systems, when using full-color images, they usually focus on using the ***RGB*** color space. However, besides the ***RGB*** model of the representation of images, there are a number of models that are more suitable both for technical purposes and more adapted to human perception.

Let us give examples of conversions from one space to another. Let us briefly consider the main features of these color spaces, and using the corresponding transformation functions of the ***OpenCV*** library.

A full-color ***RGB*** image `(Red, Green, Blue)` is an ***MxNx3*** array consisting of three ***MxN*** matrices.

They correspond to three color components: red, green and blue. The three monochrome images that form a single ***RGB*** image are called the red, green, and blue components of the image. They are in the format `uint8, uint16, single` or `double`. The following numerical data is used to represent the color components:

- `[0,1]` – for `double` and `single` formats;
- `[0,255]` – for `uint8` format;
- `[0,65535]` – for the `uint16` format.

The number of bits used to represent the magnitude of the color peak across all ***RGB*** components is

кольору зображення. Наприклад, якщо кожна компонента є 8-бітовим зображенням, то відповідне ***RGB***-зображення має глибину 24 біти. У моделі ***RGB*** не розділено яскравісна й відтінкова компоненти кольору, тут легко вказати яскравість для одного з основних кольорів, але важко вказати відтінок з необхідним колірним тоном (наприклад, тілесним) і насиченістю.

Колірна модель ***HSV*** (`Hue, Saturation, Value` – колірний тон, насиченість, міра яскравості) – модель, орієнтована на людину, що забезпечує можливість явного задання необхідного відтінку кольору. Серед інших моделей, що зараз використовуються, ця модель відображає фізичні властивості кольору і найбільш точно відповідає способу сприйняття кольору людським оком.

YUV – колірна модель, у якій колір подається як три компоненти − яскравість (***Y***) і дві кольоророзницевих (***U*** і ***V***). Модель широко застосовується в телемовленні й зберіганні/обробленні відеоданих.

Компонента яскравості містить чорно-біле (у відтінках сірого) зображення, а дві інші компоненти − інформацію для відновлення належного кольору. Це було зручно в момент появи кольорового ТБ для сумісності зі старими чорно-білими телевізорами.

У колірному просторі ***YUV*** є одна компонента, що являє собою яскравість (сигнал яскравості), і дві інші компоненти, що являють собою колір (сигнал кольоро-

called the image color depth. For example, if each component is an 8-bit image, then the corresponding ***RGB*** image has a depth of 24 bits. In the ***RGB*** model, the luminance and tint color components are not separated, it is easy to indicate the brightness for one of the primary colors here, but it is difficult to specify a hue with the desired color tone (for example, solid) and saturation.

The ***HSV*** color model (`Hue, Saturation, Value` - color tone, saturation, measure of brightness) is a person-oriented model and provides the ability to explicitly specify the desired color hue. Among other models currently in use, this model reflects the physical properties of color and most closely matches the way in which the color is perceived by the human eye.

YUV is a color model in which the color is represented as 3 components - brightness (***Y***) and two color difference (***U*** and ***V***). The model is widely used in television broadcasting and storage/processing of video data.

The luminance component contains a “black and white” (in shades of gray) image, and the remaining two components contain information for restoring the desired color. It was convenient at the time of the appearance of color TV for compatibility with old black and white TVs.

In the ***YUV*** color space, there is one component that represents luminance (luminance signal), and two other components that repre-

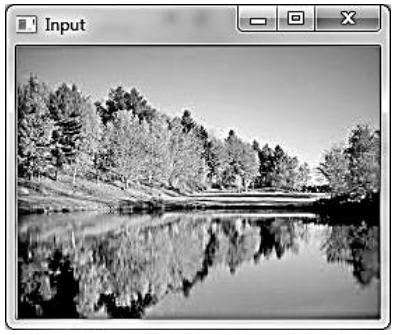
вості). У той час як яскравість передається з усіма деталями, деякі деталі в компонентах кольоророзницевого сигналу, що не має інформації про яскравість, можуть бути видалені шляхом зниження дозволу відліків (фільтрації або усереднення).

Далі наведемо приклад перетворення зображення з формату ***RGB*** на формат ***YUV***. Результати перетворення показано на рис. 6.5:

sent color (chrominance signal). While the brightness is transmitted with all the details, some of the details in the components of the color difference signal devoid of brightness information can be removed by lowering the resolution of the samples (filtering or averaging).

Below is an example of converting an image from ***RGB*** format to ***YUV*** format. The conversion results are shown in Ffig. 6.5:

```
import cv2

img = cv2.imread('./images/input.jpg')

yuv_img = cv2.cvtColor(img, cv2.COLOR_BGR2YUV)

cv2.imshow('Input', img)
cv2.imshow('YUV image', yuv_img)
cv2.imshow('Y channel', yuv_img[:, :, 0])
cv2.imshow('U channel', yuv_img[:, :, 1])
cv2.imshow('V channel', yuv_img[:, :, 2])

cv2.waitKey()
```



а

б

в

г

Рис. 6.5. Перетворення ***RGB***-зображення в формат ***YUV***

6.5. Convert ***RGB*** image to ***YUV*** format

д

Рис. 6.5. Закінчення

6.5. End

Наведемо ще один приклад перетворення зображення з формату ***RGB*** на формат ***HSV*** з використанням функції `cv2.cvtColor (img,cv2.COLOR_BGR2HSV)`. Результати перетворення показано на рис. 6.6:

Let us give another example of converting an image from ***RGB*** format to ***HSV*** format using the function `cv2.cvtColor(img,cv2.COLOR _BGR2HSV)`. The conversion results are shown in Fig. 6.6:

```
import cv2

img = cv2.imread('./images/input.jpg')
hsv_img = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)

cv2.imshow('Input', img)
cv2.imshow('HSV image', hsv_img)
cv2.imshow('H channel', hsv_img[:, :, 0])
cv2.imshow('S channel', hsv_img[:, :, 1])
cv2.imshow('V channel', hsv_img[:, :, 2])

cv2.waitKey()
```

Виділення меж на зображенні. Функція `BW = edge(I,method)` призначена для виділення меж на вихідному півтоновому зображенні `I`. Ця функція повертає бінарне зображення `BW` таким же розміром, як вихідне `I`. Піксель `BW(r,c)` дорівнює `1`, якщо піксель `I(r,s)` належить межі. Для виявлення меж можна використовувати кілька методів. Метод задається в параметрі `method` у вигляді одного з таких рядків: `'sobel', 'prewitt', 'roberts','log','zerocross'`

Selection of borders on the image. The function `BW=edge (I,method)` is intended to highlight the borders on the original half-tone image `I`. This function returns a binary image `BW` of the same size as the original `I`. Pixel `BW(r,s)` is 1 if pixel `I(r,c)` belongs to the border. Several methods can be used to detect boundaries. The method used is specified in the parameter method as one of the following lines: `'sobel','prewitt','roberts', 'log','zerocross','canny'`. These names correspond to the name of the authors of various algo-

,'canny'. Ці назви відповідають іменам авторів різних алгоритмів. Найбільш точним та ефективним при визначенні меж є метод Канні.

Для кожного з методів визначення меж можна задати додаткові параметри. Для цього використовується одна з функцій `BW = edge(I,method, thresh)`, де параметр `thresh` задає поріг для визначення того, чи належить піксель до межі.

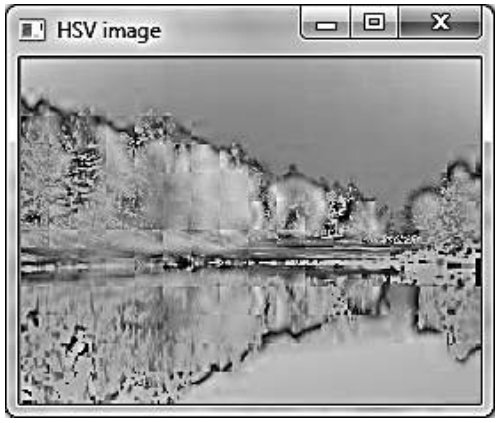
Наведемо приклад програмного коду визначення меж за методом Канні із застосуванням функції `edges = cv.Canny(img, thresh)` (рис. 6.7):

rithms. The most accurate and effective in determining the boundaries is the Canny method.

For each of the methods of determining the boundaries, additional parameters can be set. To do this, use one of the functions `BW = edge (I, method,thresh)`, where the `thresh` parameter sets the threshold for determining whether the pixel belongs to the edge.

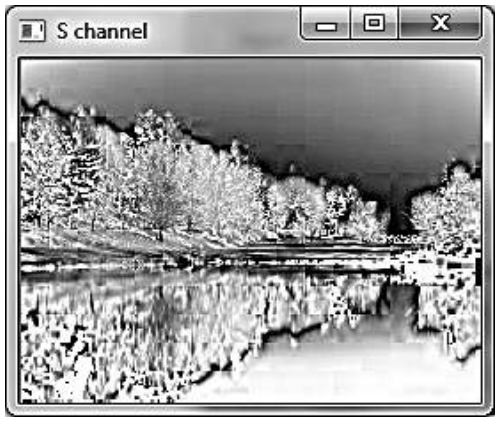
Here is an example of the Canny code definition software code using `edges = cv.Canny(img, thresh)` (Fig. 6.7):

а



б

в



г

Рис. 6.6. Перетворення ***RGB***-зображення на формат ***HSV***

Fig. 6.6. Convert ***RGB*** image to ***HSV*** forma

д

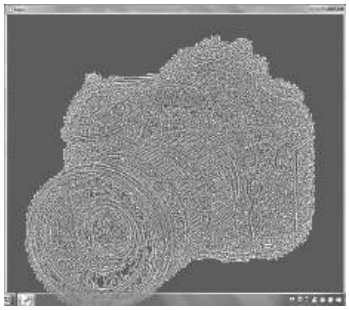
Рис 6.6. Закінчення

Fig. 6.6. End

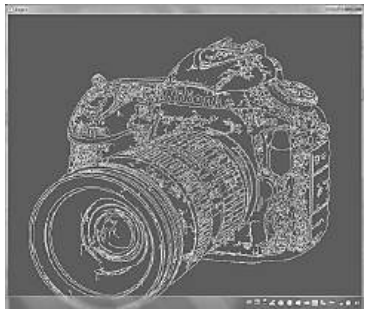
```
import cv2 as cv
# Читання RGB-зображення
img = cv.imread('17.jpg')
# Показ RGB-зображення
cv.imshow("RGB",img)
# Перетворення RGB-зображення у відтінки сірого
gray = cv.cvtColor(img, cv.COLOR_RGB2GRAY)
cv.imshow("Gray",gray)
# Виділення меж зображення (алгоритм Канні)
edges = cv.Canny(img,0,95)
cv.imshow("Edges",edges)
cv.waitKey()
```

а

б



в



г

Рис. 6.7. Виділення меж

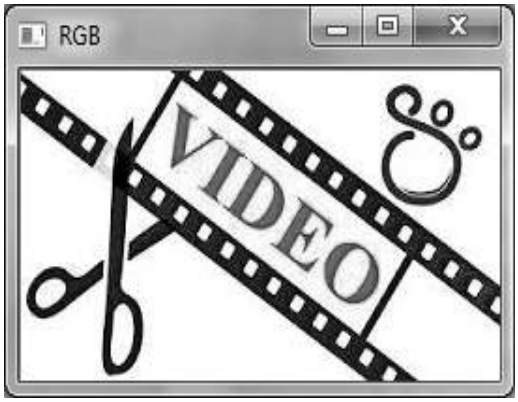
Fig. 6.7. Highlighting borders

На рис. 6.7. *в*, *г* видно, що результати оброблення можуть бути різними: при малих значеннях параметра `thresh` (близько 0.5…0.7) виділяються межі всіх дрібних фрагментів зображення, а при великих значеннях цього параметра (0.9…0.95) добре виділяються тільки глобальні контури. Але ефективність виділення меж значною мірою залежить і від самого зображення − рисунки з великою текстурою обробляються краще, що добре видно на рис. 6.8.

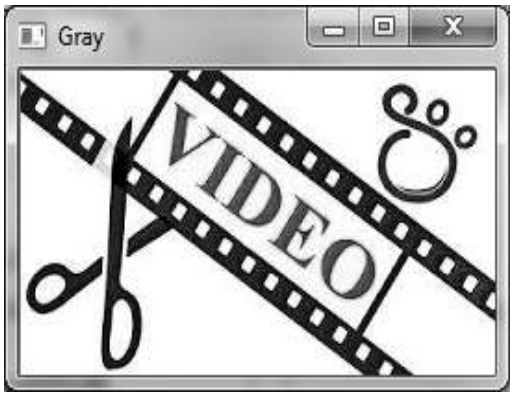
У цій роботі для скорочення обсягу матеріалу, що викладається, не наведено дані про виділення меж методом Собеля. Читач може розглянути ці алгоритми самостійно.

In Fig. 6.7, in and it can be seen that the processing results can be different – for small values of the thresh parameter (about 0.5…0.7), the borders of all small image fragments are highlighted, and for large values of this parameter (0.9…0.95), only global contours are well distinguished. But the effectiveness of the selection of borders largely depends on the image itself – drawings with a large texture are processed better, as can be clearly seen in Fig. 6.8.

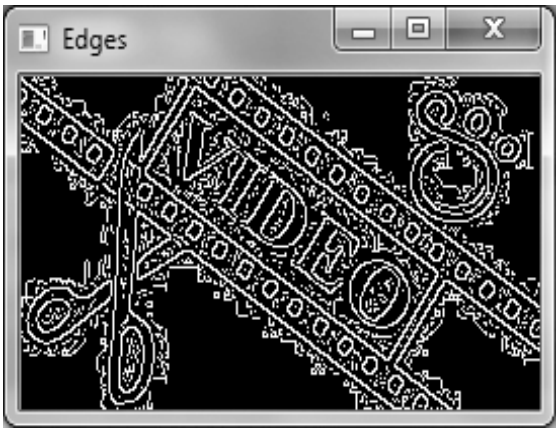
In this work, in order to reduce the volume of the presented material, there is no data on border delineation using the Sobel method. The reader can consider these algorithms on their own.



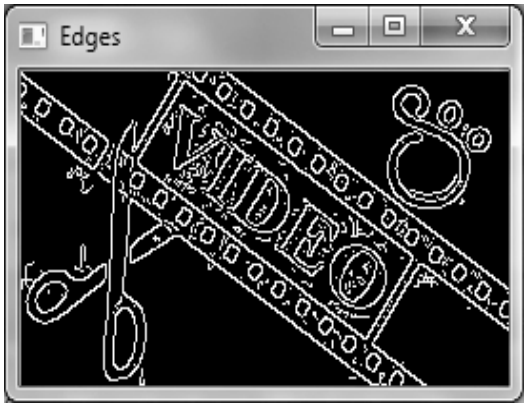
а



б



в



г

Рис. 6.8. Ефективність виділення меж

Fig. 6.8. Efficiency of allocation borders

Морфологічні перетворення зображень

Розглянемо основні морфологічні перетворення:

- `Erode` – розмивання (операція звуження);
- `Dilate` – розтягування (операція розширення).

Ерозія (розмивання/звуження) зображення зазвичай використовується для видалення випадкових вкраплень на зображенні. Ідея полягає в тому, що вкраплення при розмиванні видаляються, тоді як великі й відпо-відно більш візуально значущі області залишаються.

Дилатація (розширення) так само має усувати шум і сприяти об'єднанню областей зображення, розділених шумом, тінями та ін. Застосування ж невеликого розширення має об'єднати ці області в одну.

Морфологічні операції найчастіше застосовуються для роботи над двійковими зображеннями, які виходять після порогового перетворення (`thresholding`). Тому як вихідне зображення в нашому прикладі використовуємо файл форматом `*png`. Для реалізації програмного коду використовуємо функції `cv2.erode(img,kernel, iterations=1)` і `cv2.dilate (img,kernel,itera-tions=1)` (результати оброблення показано на рис. 6.9):

```
import cv2
import numpy as np

img = cv2.imread('30.png', 0)
kernel = np.ones((5,5), np.uint8)
```

Erosion and dilation

Consider the basic morphological transformations:

- `Erode` – blurring (narrowing operation);

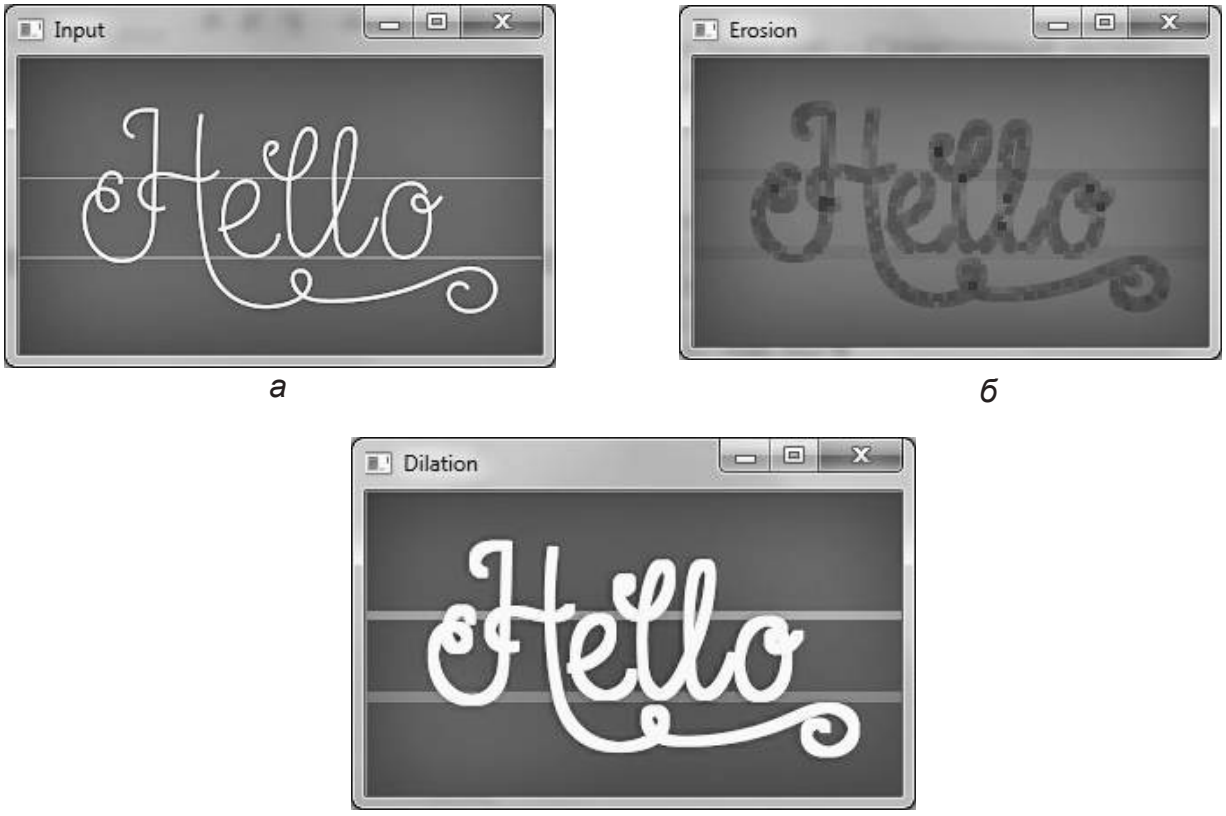
`Dilate` – stretching (expansion operation).

Erosion (blurring/contraction) of an image is usually used to get rid of random blotches on an image. The idea is that the blotches will be eliminated during erosion, while the larger and therefore more visually significant regions remain.

Dilatation (expansion), in theory, should also eliminate noise and promote the unification of image areas that were separated by noise, shadows, etc. The application of a small extension should fuse these areas into one.

Morphological operations are most often applied over binary images that are obtained after a threshold transform `(thresholding)`. Therefore, in our example, we use the `*png` file as the source image. To implement the program code we use the functions `cv2.erode (img,kernel, iterations = 1)` and `cv2.dilate (img,kernel, iterations = 1)` (the processing results are shown in Fig. 6.9):

```
img_erosion = cv2.erode(img, kernel, iterations=1)
img_dilation = cv2.dilate(img, kernel, iterations=1)

cv2.imshow('Input', img)
cv2.imshow('Erosion', img_erosion)
cv2.imshow('Dilation', img_dilation)

cv2.waitKey(0)
```



а *б*

в

Рис. 6.9.Результати морфологічних перетворень

Fig. 6.9. The results of morphological transformations

6.3. Алгоритми фільтрації зображень у бібліотеці OpenCV

У практиці цифрового оброблення зображень широко використовується маскова фільтрація. Зазвичай як маски використовується безліч вагових коефіцієнтів, заданих у всіх точках, що симетрично оточують поточну точку кадру.

Поширеним видом околу, що часто застосовується на практиці, є квадрат 3x3 з поточним елемен-

6.3. Algorithms for filtering images in the library OpenCV

In the practice of digital image processing is widely used mask filtering. Usually, a set of weights is used as a mask, which are set at all points of the neighborhood symmetrically surrounding the current frame point.

A common type of neighborhood often used in practice is a 3x3 square with the current element in

том у центрі. Застосовують різні маски, одним з евристичних варіантів є рівномірна маска, усі дев'ять вагових коефіцієнтів якої дорівнюють 1/9. Такий вибір коефіцієнтів відповідає умові збереження середньої яскравості, унаслідок чого вихідний сигнал виявляється вписаним у діапазон яскравості вхідного сигналу.

Просторова фільтрація виконується як операція двовимірної згортки імпульсної характеристики фільтра h із зображенням $\boldsymbol{f}(\boldsymbol{x},\boldsymbol{y})$. Вхідний сигнал фільтра, вихідний сигнал фільтра й імпульсна перехідна характеристика (ІПХ) являють собою двовимірні функції $\boldsymbol{x_{ij}}$, $\boldsymbol{y_{ij}}$ і $\boldsymbol{h_{kl}}$ відповідно, і дискретна згортка також має двовимірний характер:

the center. Different masks are used, one of the heuristic variants is a uniform mask, all nine weighting coefficients of which are 1/9. This choice of coefficients meets the condition of preserving the average brightness, as a result of which the output signal is inscribed in the brightness range of the input signal.

Spatial filtering is performed as a two-dimensional convolution operation of the filter impulse response $\boldsymbol{h}$ with the image $\boldsymbol{f}(\boldsymbol{x},\boldsymbol{y})$. The filter input, filter output, and impulse transient response (TIR) are two-dimensional functions $\boldsymbol{x_{ij}}$, $\boldsymbol{y_{ij}}$ and $\boldsymbol{h_{kl}}$, respectively, and the discrete convolution also has a two-dimensional character:

$$\boldsymbol{y_{ij}} = \sum_{\boldsymbol{k}} \sum_{\boldsymbol{l}} \boldsymbol{x_{i-k,j-l} h_{kl}} .$$

Точка зображення на виході двовимірного лінійного фільтра являє собою суму (середнє значення) усіх сусідніх точок зображення, що фільтрується, узятих з вагами, які визначаються видом ІПХ фільтра.

Таку процедуру ще називають віконною фільтрацією, де вікно фільтра – це його ІПХ. Усі елементи зображення, що фільтрується, які потрапили у вікно, усереднюються, унаслідок чого формується вихідний сигнал фільтра.

Процедура фільтрації для всього зображення − це послідовне виконання фільтрації в кожній точці. Вікно фільтра переміщається по

The image point at the output of a two-dimensional linear filter is the sum (average value) of all neighboring points of the filtered image, taken with weights, determined by the type of IPX filter.

This procedure is also called window filtering, where the filter window is its IPC. All elements of the filtered image that fall into the window are averaged and thus form the output signal of the filter.

The filtering procedure for the entire image is the sequential execution of filtering at each point. The filter window moves through all ele-

всіх елементах зображення, що фільтрується, і в кожному своєму положенні формує усереднене значення яскравості по всіх точках, що потрапили у вікно (рис. 6.10).

ments of the filtered image and in each of its position forms the average brightness value over all points that fall into the window (Fig. 6.10).

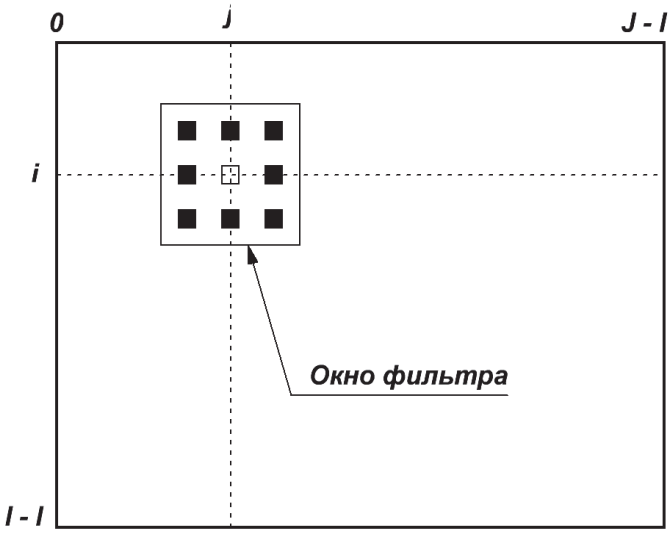


Рис. 6.10. Схема віконної (маскової) фільтрації

Fig. 6.10. Scheme of window (mask) filtering

Наведемо приклади використання масок, що застосовуються для фільтрації зображень різними способами.

Фільтри НЧ. Для зменшення шумів широко застосовуються НЧ-фільтри, оскільки шум являє собою ВЧ-сигнал. Зокрема, для НЧ-фільтрації застосовується усереднення сигналу в масці, наприклад, при n = m = 3:

Let us give examples of the use of masks used to filter images in various ways.

Low Pass Filters. Low-pass filters are widely used to reduce noise, since noise is a high-frequency signal. In particular, averaging of a signal in a mask is applied to LF filtering, for example, when n = m = 3:

$$H = \frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}, \qquad H = \frac{1}{10}\begin{bmatrix}1 & 1 & 1\\ 1 & 2 & 1\\ 1 & 1 & 1\end{bmatrix}, \qquad H = \frac{1}{16}\begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix}.$$

Фільтри ВЧ. Якщо зображення має нечіткі межі (низьку різкість), то його можна профільтрувати ВЧ-фільтром, який підкреслить перепади яскравості на зображенні і зробить його більш чітким. Для ВЧ-фільтрації зазвичай використовують такі типові маски:

HF filters. If the image has fuzzy borders (low sharpness), then it can be filtered with an RF filter, which will emphasize the brightness differences in the image and make it clearer. For HF filtering, these typical masks are commonly used:

$$H = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}, \quad H = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix}, \quad H = \begin{bmatrix} 1 & -2 & 1 \\ -2 & 5 & -2 \\ 1 & -2 & 1 \end{bmatrix}.$$

Нормування є необхідним для того, щоб звести значення відгуку фільтра до діапазону вхідних даних. Нормувальний коефіцієнт визначається з умови рівності одиниці суми всіх коефіцієнтів ІПХ фільтра.

Наведемо приклади використання різних фільтрів для оброблення зображень з використанням у програмних кодах ***Python*** алгоритмів бібліотеки ***OpenCV***.

Фільтрація НЧ. Програмний код фільтра низьких частот з масками розміром 3x3 і 5x5 пікселів наведено далі, а результати фільтрації показано на рис. 6.11. Добре видно, що при збільшенні розміру маски (рис. 6.11 *в, г*) збільшується ступінь розмиття меж зображення.

Приклад:

Normalization is necessary in order to bring the filter response values to the range of input data. The normalization coefficient is determined from the condition of equality of the unit of the sum of all coefficients of the IPX filter.

Let us give examples of using various filters for image processing using the ***OpenCV*** library algorithms in **Python** program codes.

Low pass filtering. The program code of the low-pass filter with masks of 3x3 and 5x5 pixels is shown below, and the filtering results are shown in Fig. 6.11. It is clearly seen that with increasing mask size (Fig.6.11 *c, d*) the degree of blurring of the image borders increases

Example:

```
import cv2
import numpy as np

img = cv2.imread('5.jpg')
kernel_indentity = np.array([[0,0,0],[0,1,0],[0,0,0]]) rows,
cols = img.shape[:2]
kernel_indentity = np.array([[0,0,0], [0,1,0], [0,0,0]])
kernel_3x3 = np.ones((3,3), np.float32) / 9.0
kernel_5x5 = np.ones((5,5), np.float32) / 25.0

cv2.imshow('Original', img)

output = cv2.filter2D(img, -1, kernel_indentity)
cv2.imshow('Indentity filter', output)

output = cv2.filter2D(img, -1, kernel_3x3)
cv2.imshow('3x3 filter', output)

output = cv2.filter2D(img, -1, kernel_5x5)
cv2.imshow('5x5 filter', output)

cv2.waitKey(0)
```

а

б

в

г

Рис. 6.11. Оброблення зображення фільтрами НЧ з масками різних розмірів

Fig. 6.11. Image processing by low-pass filters with masks of different sizes

Фільтрація ВЧ широко використовується, якщо необхідно підкреслити розмиті межі об'єктів на зображенні. Це добре видно на рис. 6.12, *б*. Зверніть увагу на те, що при написанні програмних кодів фільтрації у всіх прикладах крім підключення бібліотеки ***OpenCV*** `(import cv2)`, у програму необхідно імпортувати модуль `numpy` для формування маски фільтра `(kernel)`.

Приклад:

```
import cv2
import numpy as np

img = cv2.imread('6.jpg')
cv2.imshow('Original', img)
```

HF filtering is widely used when it is necessary to emphasize the blurred edges of objects in an image. This is clearly seen in Fig.6.12, *b*. Please note that when writing software filtering codes in all examples besides connecting the ***OpenCV*** library `(import cv2)` the numpy module must be imported into the program to form a filter mask `(kernel)`.

Example:

```
kernel_indentity = np.array([[-1,-1,-1], [-1,9,-1], [-1,-1,-
1]])
kernel_3x3 = np.ones((3,3), np.float32)

output = cv2.filter2D(img, -1, kernel_indentity)
cv2.imshow('3x3 filter', output)

cv2.waitKey(0)
```

а

Рис. 6.12. Оброблення зображення ВЧ-фільтром

Фільтрація з ефектом зсуву зазвичай використовується для створення ефекту руху, що суттєво розмиває зображення. Приклад програмного коду такої фільтрації наведено нижче, а результат оброблення показано на рис. 6.13, *б*.

Приклад:

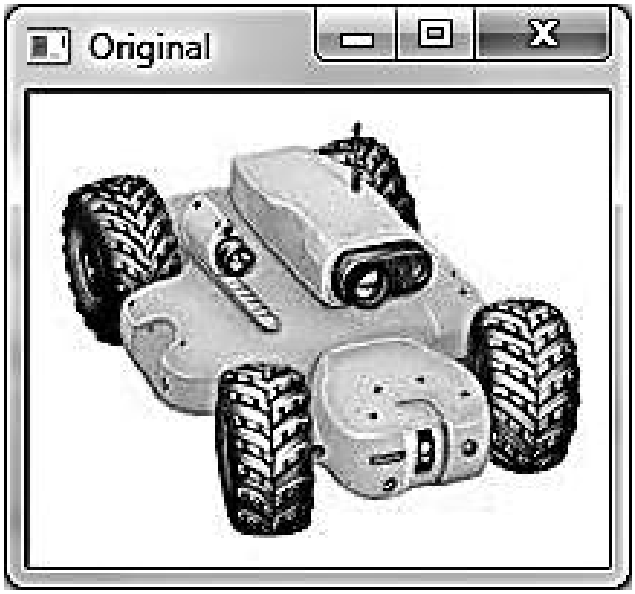
```
img = cv2.imread('2.jpg')
cv2.imshow('Original', img)

size = 15

# generating the kernel
kernel_motion_blur = np.zeros((size, size))
kernel_motion_blur[int((size-1)/2), :] = np.ones(size)
kernel_motion_blur = kernel_motion_blur / size

# applying the kernel to the input image
output = cv2.filter2D(img, -1, kernel_motion_blur)

cv2.imshow('Motion Blur', output)
cv2.waitKey(0)
```

б

Fig. 6.12. Image Processing HF-Filter

Motion Blur is usually used to create a motion effect that blurs images significantly. An example of the program code of such filtering is shown below, and the result of processing is shown in Fig. 6.13, *b*.

Example:

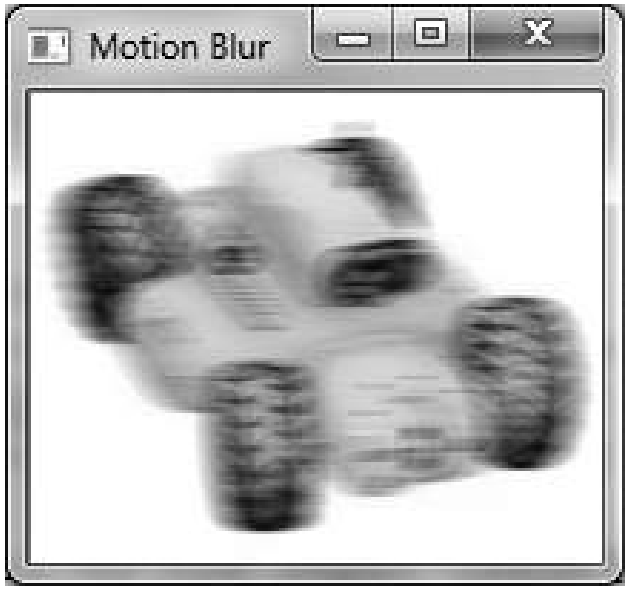


а

Рис. 6.13. Варіанти фільтрації з підкресленням меж

Фільтрація з підкресленням меж є одним з різновидів ВЧ-фільтрації і створює різні візуальні ефекти (рис. 6.14) унаслідок підкреслення меж зображення.

Приклад:



б

Fig. 6.13. Underline Filtering Options

Sharpening filter is one of the varieties of high-pass filtering and creates various visual effects (Fig. 6.14) by emphasizing the image borders.

Example:

```
import cv2
import numpy as np

img = cv2.imread('3.jpg')
cv2.imshow('Original', img)

# generating the kernels
kernel_sharpen_1 = np.array([[-1,-1,-1], [-1,9,-1], [-1,-1,-
    1]])
kernel_sharpen_2 = np.array([[1,1,1], [1,-7,1], [1,1,1]])
kernel_sharpen_3 = np.array([[-1,-1,-1,-1,-1],
                             [-1,2,2,2,-1],
                             [-1,2,8,2,-1],
                             [-1,2,2,2,-1],
                             [-1,-1,-1,-1,-1]]) / 8.0

# applying different kerneks to the input image
output_1 = cv2.filter2D(img, -1, kernel_sharpen_1)
output_2 = cv2.filter2D(img, -1, kernel_sharpen_2)
output_3 = cv2.filter2D(img, -1, kernel_sharpen_3)
cv2.imshow('Sharpening', output_1)
cv2.imshow('Excessive Sharpening', output_2)
cv2.imshow('Edge Enhancement', output_3)
```

а

в

б

г

Рис. 6.14. Варіанти фільтрації з підкресленням меж

Fig. 6.14. Underline Filtering Options

Фільтрація з перетворенням на рельєфне зображення базується на поданні фрагментів зображення з рівномірною яскравістю пікселями сірого кольору, а областей зі значними перепадами яскравості пікселями білого кольору. Це створює ефект тиснення й об'ємності зображення. Ступінь тиснення можна змінити з допомогою різних видів фільтрувальних масок `(kernel_emboss)`. Зазначимо також, що перед процедурою фільтрації кольорове зображення необхідно перетворити на півтонове з допомогою функції `cv2.COLOR_BGR2GRAY`. Результати фільтрації показано на рис. 6.15.

Приклад:

```
import cv2
import numpy as np
```

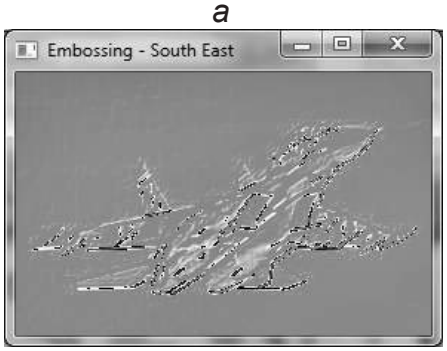
Embossing filter. Filtration with conversion into a relief image is based on the representation of image fragments with uniform brightness by gray pixels, and areas with significant differences in brightness by white pixels. This creates the effect of embossing and three-dimensional image. Changes in the degree of embossing can be achieved using a different kind of filtering masks `(kernel_emboss)`. Note also that before the filtering procedure it is necessary to convert a color image to a halftone using the function `cv2.COLOR_BGR2GRAY`. Filtering results are shown in Fig. 6.15.

Example:

```
img_emboss_input = cv2.imread('7.jpg')

# generating the kernels
kernel_emboss_1 = np.array([[0,-1,-1],
                            [1,0,-1],
                            [1,1,0]])
kernel_emboss_2 = np.array([[-1,-1,0],
                            [-1,0,1],
                            [0,1,1]])
kernel_emboss_3 = np.array([[1,0,0],
                            [0,0,0],
                            [0,0,-1]])

# converting the image to grayscale
gray_img = cv2.cvtColor(img_emboss_input,cv2.COLOR_BGR2GRAY)

# appling the kernels to the grayscale image and adding the
offset
output_1 = cv2.filter2D(gray_img, -1, kernel_emboss_1) + 128
output_2 = cv2.filter2D(gray_img, -1, kernel_emboss_2) + 128
output_3 = cv2.filter2D(gray_img, -1, kernel_emboss_3) + 128

cv2.imshow('Input', img_emboss_input)
cv2.imshow('Embossing - South West', output_1)
cv2.imshow('Embossing - South East', output_2)
cv2.imshow('Embossing - North West', output_3)

cv2.waitKey(0)
```

а

б



в

г

Рис. 6.15. Фільтрація з перетворенням на рельєфне зображення

Fig. 6.15. Filtration with conversion to relief image

Фільтрація гаусівськими і білатеральними фільтрами. Нагадаємо, що гаусівський фільтр має імпульсну перехідну характеристику

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+e^2}{2\sigma^2}},$$

де x – розміри фільтрувальної маски по горизонтальній осі, y – розміри маски по вертикальній осі, σ − стандартне відхилення розподілу Гаусса або ж міра розмиття. Фільтрація Гаусса є фільтрацією згладжувального типу, але, на відміну від прямокутної фільтрації, розмиття буде більш рівномірно розподілятися від центральної точки до країв. Фільтр Гаусса є високочастотним фільтром, а отже, його вигідно застосовувати для підвищення різкості зображення.

Білатеральна фільтрація розширює поняття «згладжування Гаусса», збільшуючи якість фільтра. Пікселі, які сильно відрізняються за інтенсивністю від центрального пікселя, збільшуються меншою мірою, навіть не зважаючи на те, що вони можуть знаходитися в безпосередній близькості до центрального пікселя, що фактично є викривленням нелінійного фільтра Гаусса.

Наведемо приклад гаусівської й білінійної фільтрації, використовуючи функції `cv2.GaussianBlur` і `cv2.bilateralFilter`. При цьому використовуємо два варіанти фільтрувальної маски для фільтра Гаусса − розмірами 7x7 і 15x15. Код цієї програми наведено нижче, а результати оброблення показано на рис. 6.16:

Gaussian and Bilateral filters. Recall that the Gaussian filter has a pulsed transient characteristic of the image

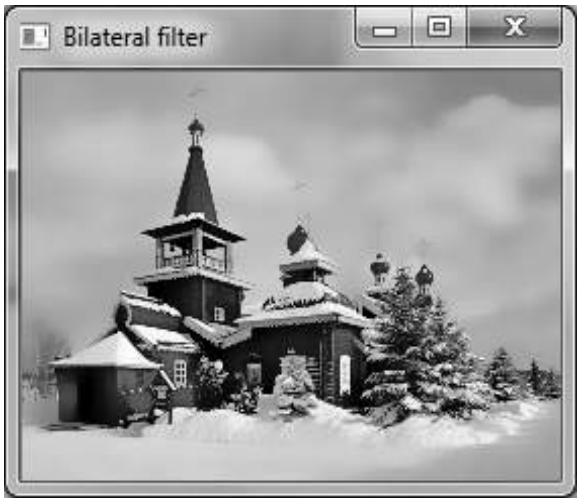
where x is the size of the filtering mask on the horizontal axis, y is the size of the mask on the vertical axis, σ is the standard deviation of the Gaussian distribution, or the degree of blurring. Gaussian filtering is a smoothing type filtering, but, unlike rectangular filtering, the blur will be more evenly distributed from the center point to the edges. The Gauss filter is a high-frequency filter, which means that it is beneficial to use it to sharpen the image.

Bilateral filtering extends the notion of Gaussian smoothing, increasing the quality of the filter. Pixels, which are very different in intensity from the central pixel, increase to a lesser extent, even though they may be in close proximity to the central pixel, which is in fact a curvature of the non-linear Gauss filter.

We give an example of Gaussian and bilinear filtering using the functions `cv2.GaussianBlur` and `cv2.bilateralFilter`. In this case, we use two variants of the filtering mask for a Gaussian filter − 7x7 and 15x15. The code of this program is given below, and the results of processing are shown in Fig. 6.16.

```
import cv2
import numpy as np

img = cv2.imread('21.jpg')
img_gaussian = cv2.GaussianBlur(img, (7,7), 0)
img_gaussian1 = cv2.GaussianBlur(img, (15,15), 0)
img_bilateral = cv2.bilateralFilter(img, 13, 70, 50)

cv2.imshow('Input', img)
cv2.imshow('Gaussian filter', img_gaussian)
cv2.imshow('Gaussian filter 15x15', img_gaussian1)
cv2.imshow('Bilateral filter', img_bilateral)

cv2.waitKey()
```

а



б

в

г

Рис. 6.16. Використання гаусівських і білатеральних фільтрів

Fig. 6.16. Use of Gaussian and bilateral filters

6.4. Гістограми розподілу яскравості зображень

Однією з найбільш істотних характеристик зображення є характер розподілу яскравості пікселів. Цей розподіл кількісно характеризує гістограма, зображена на рис. 6.17. На графіку вісь абсцис є шкалою яскравості у нашому випадку від `0` (чорні пікселі) до максимуму `255` (білі пікселі). Це відповідає поданню півтонових зображень у форматі `uint8` з діапазоном значень яскравості `[0,255]`. По осі ординат відкладають кількість пікселів, що відповідає кожному значенню яскравості. Це допомагає візуально оцінити домінуючий діапазон яскравості на реальному зображенні. Так, на рис. 6.17 добре видно, що на зображенні переважають темні (погано освітлені) фрагменти.

Урахування характеру розподілу яскравості дає змогу провести перетворення для поліпшення контрастності зображення. Ця процедура сприяє вирівнюванню гістограми у всьому діапазоні яскравості і має назву «еквалізація». Окрім цього можна здійснити бінаризацію зображення з відсіканням за заданим порогом яскравості.

Остання процедура дає можливість просторово локалізувати в площині зображення фрагменти з великим або малим рівнем освітленості. У системах технічного зору бінаризація з високим порогом відсікання широко застосовується для оцінювання положення спостережуваного об'єкта в кадрі.

6.4. Image brightness distribution histograms

One of the most significant characteristics of the image is the nature of the brightness distribution of pixels. This distribution is taken to be quantitatively characterized by a histogram (Fig. 6.17). On the graph, the abscissa axis is a scale of bright-bones – in our case from 0 (black color of pixels) to a maximum of `255` (white pixels). This corresponds to the representation of halftone images in the `uint8` format with a range of brightness values `[0,255]`. The ordinate axis lays the number of pixels corresponding to each value of the brightness. This helps to visually assess the dominant range of brightness in the real image. So on Fig. 6.17 it is clearly seen that dark (poorly lit) fragments predominate in the image.

Taking into account the nature of the distribution of brightness allows for a conversion that helps improve the contrast of the image. This procedure contributes to the alignment of the histogram in the whole range of brightness and is called equalization. In addition, it is possible to binarize an image with clipping at a given brightness threshold.

The latter procedure makes it possible to spatially localize in the image plane fragments with a high or low level of illumination. In technical vision systems, binarization with a high cut-off threshold is widely used to estimate the position of the observed object in the frame.

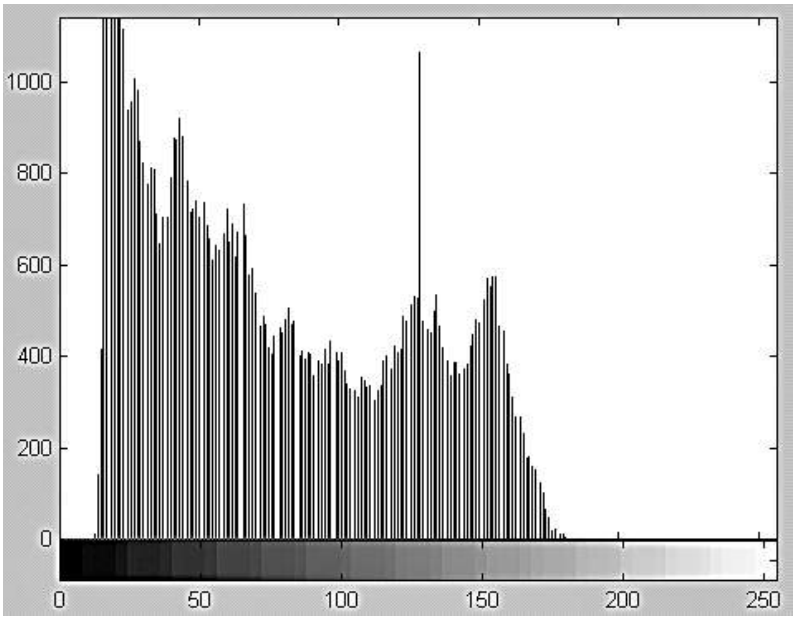


Рис. 6.17. Гістограма розподілу яскравості

Fig. 6.17. Brightness Distribution Histogram

Наведемо приклад програмного коду побудови гістограм зображення, включеного в інсталяцію бібліотеки ***OpenCV,*** для спільної роботи з програмою ***Python:***

Let us give an example of the software code for constructing the histograms of the image included in the installation of the ***OpenCV*** library for collaboration with the ***Python:***

```
#!/usr/bin/env python

''' This is a sample for histogram plotting for RGB images and
grayscale images for better understanding of colour distribu-
tion

Benefit : Learn how to draw histogram of images
          Get familier with cv2.calcHist,
cv2.equalizeHist,cv2.normalize and some drawing functions

Level : Beginner or Intermediate

Functions : 1) hist_curve : returns histogram of an image drawn
as curves
            2) hist_lines : return histogram of an image drawn
as bins ( only for grayscale images )
Usage : python hist.py <image_file>

Abid Rahman 3/14/12 debug Gary Bradski
'''

# Python 2/3 compatibility
from __future__ import print_function

import cv2
import numpy as np
```

```
bins = np.arange(256).reshape(256,1)

def hist_curve(im):
    h = np.zeros((300,256,3))
    if len(im.shape) == 2:
        color = [(255,255,255)]
    elif im.shape[2] == 3:
        color = [ (255,0,0),(0,255,0),(0,0,255) ]
    for ch, col in enumerate(color):
        hist_item = cv2.calcHist([im],[ch],None,[256],[0,256])

cv2.normalize(hist_item,hist_item,0,255,cv2.NORM_MINMAX)
        hist=np.int32(np.around(hist_item))
        pts = np.int32(np.column_stack((bins,hist)))
        cv2.polylines(h,[pts],False,col)
    y=np.flipud(h)
    return y

def hist_lines(im):
    h = np.zeros((300,256,3))
    if len(im.shape)!=2:
        print("hist_lines applicable only for grayscale imag-
es")
        #print("so converting image to grayscale for represen-
tation"
        im = cv2.cvtColor(im,cv2.COLOR_BGR2GRAY)
    hist_item = cv2.calcHist([im],[0],None,[256],[0,256])
    cv2.normalize(hist_item,hist_item,0,255,cv2.NORM_MINMAX)
    hist=np.int32(np.around(hist_item))
    for x,y in enumerate(hist):
        cv2.line(h,(x,0),(x,y),(255,255,255))
    y = np.flipud(h)
    return y

if __name__ == '__main__':

    import sys

    if len(sys.argv)>1:
        fname = sys.argv[1]
    else :
        fname = '../data/36.jpg'
        print("usage : python hist.py <image_file>")

    im = cv2.imread(fname)

    if im is None:
        print('Failed to load image file:', fname)
        sys.exit(1)
    gray = cv2.cvtColor(im,cv2.COLOR_BGR2GRAY)
```

```
    print(''' Histogram plotting \n
    Keymap :\n
    a - show histogram for color image in curve mode \n
    b - show histogram in bin mode \n
    c - show equalized histogram (always in bin mode) \n
    d - show histogram for color image in curve mode \n
    e - show histogram for a normalized image in curve mode \n
    Esc - exit \n
    ''')

    cv2.imshow('image',im)
    while True:
        k = cv2.waitKey(0)
        if k == ord('a'):
            curve = hist_curve(im)
            cv2.imshow('histogram',curve)
            cv2.imshow('image',im)
            print('a')
        elif k == ord('b'):
            print('b')
            lines = hist_lines(im)
            cv2.imshow('histogram',lines)
            cv2.imshow('image',gray)
        elif k == ord('c'):
            print('c')
            equ = cv2.equalizeHist(gray)
            lines = hist_lines(equ)
            cv2.imshow('histogram',lines)
            cv2.imshow('image',equ)
        elif k == ord('d'):
            print('d')
            curve = hist_curve(gray)
            cv2.imshow('histogram',curve)
            cv2.imshow('image',gray)
        elif k == ord('e'):
            print('e')
            norm = cv2.normalize(gray, gray, alpha = 0,beta =
255,norm_type = cv2.NORM_MINMAX)
            lines = hist_lines(norm)
            cv2.imshow('histogram',lines)
            cv2.imshow('image',norm)
        elif k == 27:
            print('ESC')
            cv2.destroyAllWindows()
            break
    cv2.destroyAllWindows()
```

Результати роботи програми показано на рис. 6.18. У програмі

The results of the program are shown in Fig. 6.18. The program

закладена можливість перетворення вихідного зображення на формат півтонового та його еквалізації. На початку роботи програми відбувається візуалізація вихідного зображення (рис. 6.18, *а*), і у вікні *Python 3.6.1* `Shell` пропонується інтерактивне меню:

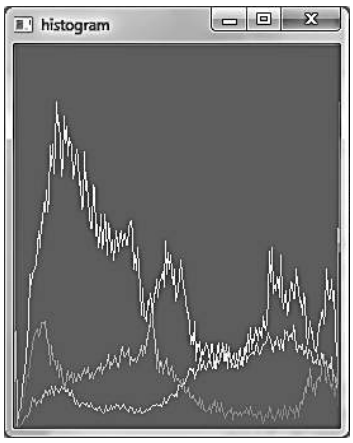
has the ability to convert the original image into a halftone format and its equalization. At the beginning of the program, the source image is rendered (Fig. 6.18, *а*), and the ***Python 3.6.1*** `Shell` window offers an interactive menu:

- `a - show histogram for color image in curve mode`
- `b - show histogram in bin mode`
- `c - show equalized histogram (always in bin mode)`
- `d - show histogram for color image in curve mode`
- `e - show histogram normalized image in curve mode.`

По черзі натискаючи клавіші *a* – *e*, можна вибрати перетворення вихідного зображення, що цікавить вас, і його гістограму.

Alternatively, by pressing the a - e keys, you can select the transformation of the original image that interests you and its histogram.

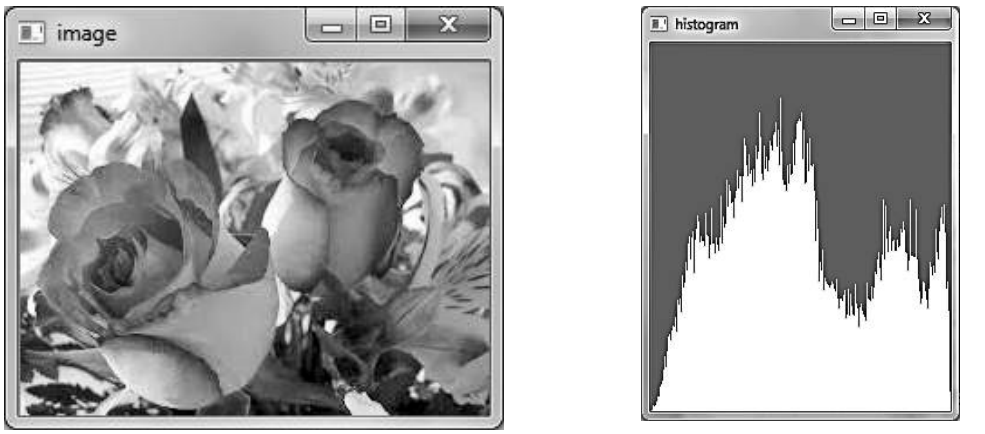
Вихідне зображення (Source image)



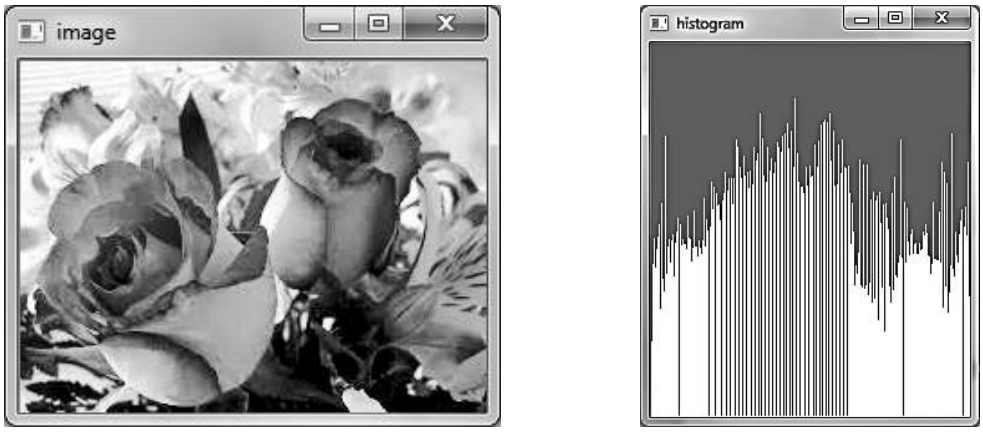
а

Рис. 6.18. Різні гістограми зображення

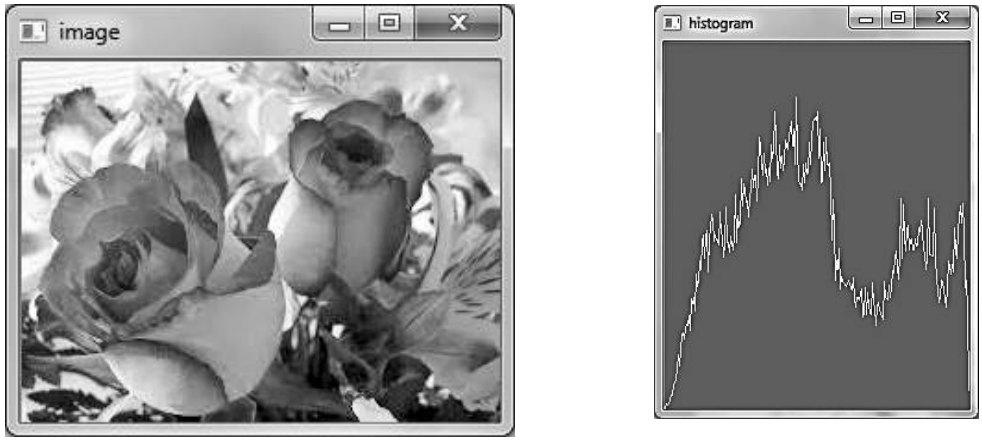
Fig. 6.18. Various image histograms



б



в



г

Рис. 6.18. Закінчення

Fig. 6.18. End

Поліпшення контрастності півтонових зображень. Наведемо найпростіший приклад підвищення контрастності зображення. Зазвичай еквалізація виконується із застосуванням функції `histeq = cv2.equalizeHist (img)` на перетвореному на півтонове `rgb`-зображенні. Програмний

Enhancing the contrast in an image

Let's take a simpler example of increasing the contrast of an image. Normally, equalization is performed using the function `histeq = cv2.equalizeHist(img)` on the converted to a grayscale `RGB`-ima-

код такого оброблення наведено нижче, а результати показано на рис. 6.19:

ge. The program code of such processing is given below, and the results are shown in Fig. 6.19.

```
import cv2
import numpy as np
img = cv2.imread('1.jpg', 0)
# equalize the histogram of the input image
histeq = cv2.equalizeHist(img)
cv2.imshow('Input', img)
cv2.imshow('Histogram equalized', histeq)
cv2.waitKey(0)
```

а

б

Рис. 6.19. Результат контрастування зображення

Fig. 6.19. Image contrasting result

Підвищення контрастності повнокольорових зображень. Набагато більший практичний інтерес становить процедура підвищення контрастності кольорових зображень, яку можна виконати шляхом перетворення `rgb`-зображення на формат ***YUV***. Нагадаємо, що це колірна модель, у якій колір подається як три компоненти: яскравість (***Y***) і дві кольоророзницеві (***U*** і ***V***). На зображенні у форматі ***YUV*** еквалізації піддається лише компонента яскравості (***Y***) з допомогою функції

Enhance the contrast of full-color images. Much more practical interest is the procedure for improving the contrast of color images. Its implementation is possible by converting the `RGB` image to the **YUV** format. Recall that this is a color model in which the color is represented as 3 components: brightness (***Y***) and two color difference (***U*** and ***V***). In a ***YUV*** image, only the luminance (***Y***) component is subjected to EQ using the function

```
img_yuv[:,:,0] = cv2.equalizeHist(img_yuv[:,:,0])
```

Потім здійснюється обернене перетворення з формату ***YUV*** на

Then the reverse transform from ***YUV*** to ***RGB*** format is performed. In

формат ***RGB***. При цьому баланс кольору зберігається без змін, оскільки кольоророзницеві компоненти ***U*** і ***V*** перетворенням не піддавалися.

Результати контрастування повнокольорового зображення за цією методикою показано на рис. 6. 20:

this case, the color balance remains unchanged, since the color difference components ***U*** and ***V*** were not subjected to transformations.

The results of the contrasting of the full-color image by this method are shown in Fig. 6.20:

```
import cv2
import numpy as np

img = cv2.imread('input.jpg')
img_yuv = cv2.cvtColor(img,cv2.COLOR_BGR2YUV)
# equalize the histogram of the Y channel
img_yuv[:,:,0] = cv2.equalizeHist(img_yuv[:,:,0])
# convert the YUV image back to RGB format
img_output = cv2.cvtColor(img_yuv, cv2.COLOR_YUV2BGR)

cv2.imshow('Color input image', img)
cv2.imshow('Histogram equalized', img_output)

cv2.waitKey(0)
```

а

б

Рис. 6.20. Підвищення контрастності повнокольорових зображень

Fig. 6.20. Enhance the contrast of full color images

6.5. Бінаризація зображень з відсіканням за порогом яскравості

Одним з важливих методів оброблення зображень у системах технічного зору є процедура їх

6.5. Binarization of images with cut-off threshold brightness

One of the important methods of image processing in vision systems is the procedure of their binarization

бінаризації з відсіканням за порогом яскравості. Це дає можливість подавати дані зображення в бінарному вигляді й проводити з ними математичні обчислення. В алгоритмі бінаризації використовується функція бібліотеки ***OpenCV:***

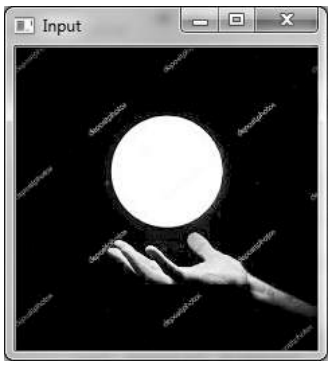
with clipping on the threshold of brightness. This makes it possible to present these images in binary form and allows you to perform mathematical calculations with them. The binarization algorithm uses the ***OpenCV*** library function:

```
thresh,im_bw)=cv2.threshold(im_gray,0,255,cv2.THRESH_BINARY
cv2.THRESH_OTSU)
```

Ця функція є регульованим порогом відсікання яскравості `thresh`. При бінаризації півтонування з діапазоном значень пікселів `(0...255)` параметр `thresh` може змінюватися в межах від 10 до 254. Зазначимо, що при високому порозі відсікання на бінарному зображенні залишаться тільки пікселі з максимальною яскравістю (білого кольору). Усі інші пікселі буде подано нульовими значеннями (чорного кольору). Код цього перетворення наведено нижче, а результати показано на рис. 6.21:

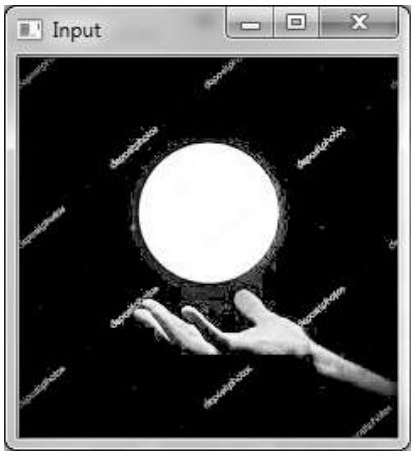
This feature has an adjustable `thresh` brightness cutoff threshold. In bi-gradation of a halftone image with a range of pixel values `(0...255)`, the `thresh` parameter can vary from 10 to 254. Note that with a high cut-off threshold, only the pixels with maximum brightness (white) remain on the binary image. All other pixels will be represented by zero values (black color). The code for this transformation is shown below, and the results are shown in Fig. 6.21:

```
img = cv2.imread('34.jpg')
cv2.imshow('Input', img)

im_gray = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)
cv2.imshow("Gray", im_gray)

(thresh, im_bw) = cv2.threshold(im_gray, 0, 255,
cv2.THRESH_BINARY | cv2.THRESH_OTSU)
thresh = 45
im_bw = cv2.threshold(im_gray, thresh, 255,
cv2.THRESH_BINARY)[1]
cv2.imshow('Binary', im_bw)
```

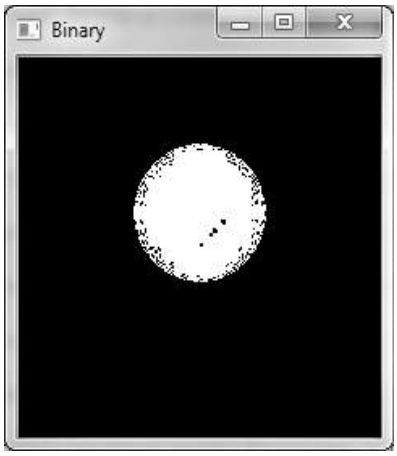


а

б



в



г

Рис. 6.21. Приклад бінаризації з різними порогами

Fig. 6.21. An example of binarization with different thresholds

6.6. Афінні й проективні перетворення зображень

Нагадаємо основні теоретичні відомості про суть афінних перетворень. Поворот, зміна розмірів зображення, зсув та інші перетворення належать до геометрич-них перетворень, які називаються афінними. Афінне перетворення можна записати в матричній формі:

6.6. Affine and projective image transformations

Recall the basic theoretical information about the essence of affine transformations. Rotate, resize images, shift, etc. refer to geometric transformations whose representatives are called affine transformations. Affinity transform can be written in matrix form:

$$[x\ y\ 1] = [w\ z\ 1]\mathbf{T}\begin{vmatrix} t_{11} & t_{12} & 0 \\ t_{21} & t_{22} & 0 \\ t_{31} & t_{32} & 0 \end{vmatrix}.$$

Такою формулою можна задати стиск, поворот, перенос або зсув, відповідним чином визначаючи елементи матриці **T**. У табл. 6.1 показано, як вибирати ці величини

Such a formula can be used to set compression, rotation, transfer or shift, by appropriately determining the elements of the matrix **T**. In Table 6.1 shows how to choose

для здійснення різних перетворень.

Розглянемо властивості функцій ***OpenCV***, що забезпечують геометричні перетворення, які змінюють просторове розташування елементів у зображенні.

Найбільш широко в комп'ютерному зорі використовуються функції змінення розмірів зображень (`Image resize`) і функції повороту зображень (`Image rotation`). Слід мати на увазі, що просторові перетворення призводять до погіршення якості зображення. Особливо це помітно на межах зображення.

these values to perform various transformations.

Consider the properties of ***OpenCV*** functions that provide geometric transformations that change the spatial location of elements in the image.

The most widely used in computer vision are the image resizing functions (`Image resize`) and the image rotation functions (`Image rotation`). It should be borne in mind that spatial transformations lead to a deterioration in image quality. This is especially noticeable on the borders of the image.

Таблиця 6.1
Table 6.1

Типи афінних перетворень (Types of Affine Transformations)		
Тип (Type of)	Афінна матриця **T** (Affinity matrix **T**)	Рівняння координат (Coordinate Equations)
Розтягування (Stretching)	$\begin{vmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{vmatrix}$	$\mathbf{X = S_x W}$ $\mathbf{Y = S_y Z}$
Поворот (Rotate)	$\begin{vmatrix} cos(a) & sin(a) & 0 \\ -sin(a) & cos(a) & 0 \\ 0 & 0 & 1 \end{vmatrix}$	$\mathbf{x = wcos(a) - zsin(a)}$ $\mathbf{y = wsin(a) + zcos(a)}$
Зсув (Shift) (горизонтальний) (horizontal)	$\begin{vmatrix} 1 & 0 & 0 \\ a & 1 & 0 \\ 0 & 0 & 1 \end{vmatrix}$	$\mathbf{x = W + aZ}$ $\mathbf{y = Z}$
Зсув (Shift) (вертикальний) (vertical)	$\begin{vmatrix} 1 & b & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{vmatrix}$	$\mathbf{x = W}$ $\mathbf{y = bW + Z}$
Перенос (Transfer)	$\begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ S_x & S_y & 1 \end{vmatrix}$	$\mathbf{X = W + S_x}$ $\mathbf{Y = Z + S_y}$

Тому процедура геометричних перетворень зазвичай доповнюється одним із видів інтерполяції. Функції інтерполяції в бібліотеці ***OpenCV*** `cv2.INTER_LINEAR`, `cv2.INTER_CUBIC`,`cv2.INTER_AREA` використовують інтерполяцію або по білінійній поверхні, або по бікубічній.

Змінення розмірів зображення. Розглянемо програму для змінення розмірів зображення. При використанні функції `cv2.resize` необхідно вказати масштаб змінення розмірів зображення (`fx` і `fy`), який може бути більше або менше 1, а також вибрати потрібний тип інтерполяції даних. На рис. 6.22 показано результати збільшення розмірів зображення в 1.5 раза з різними видами інтерполяції:

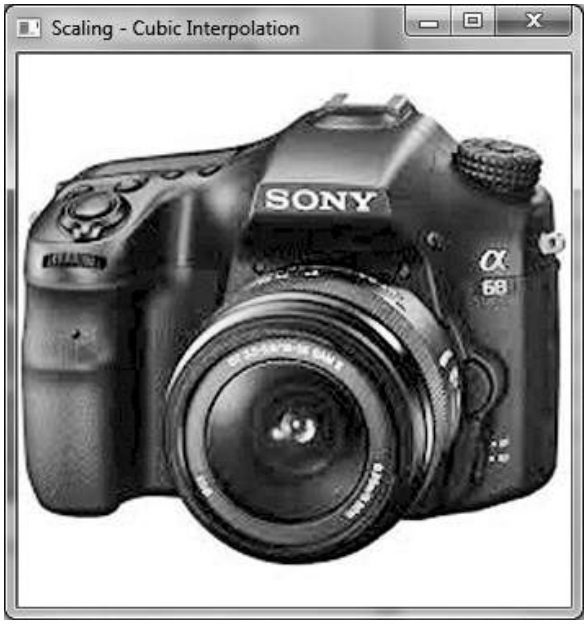
Therefore, the procedure of geometric transformations is usually supplemented by one of the types of interpolation. The interpolation functions in the ***OpenCV*** `cv2.INTER_LINEAR`,`cv2.INTER_CUBIC`, `cv2.INTER_AREA` library use interpolation either on a bilinear surface or on a bicubic one.

Image scaling Consider a program for resizing images. When using the `cv2.resize` function, you need to specify the scale of image resizing (fx and fy), which may be greater or less than 1, and also select the desired type of data interpolation. In Fig. 6.22 shows the results of increasing the image size by 1.5 times with various types of interpolation:

```
import cv2
import numpy as np

img = cv2.imread('6.jpg')
cv2.imshow('Original', img)

img_scaled = cv2.resize(img,None,fx=1.5,fy=1.5,interpolation =
cv2.INTER_LINEAR)
cv2.imshow('Scaling - Linear Interpolation', img_scaled)
img_scaled = cv2.resize(img,None,fx=1.5, fy=1.5,
interpolation = cv2.INTER_CUBIC)
cv2.imshow('Scaling - Cubic Interpolation', img_scaled)
img_scaled = cv2.resize(img,None,fx=1.5, fy=1.5,
interpolation = cv2.INTER_AREA)
cv2.imshow('Scaling - Skewed Size', img_scaled)
cv2.waitKey()
```

а

б



в

г

Рис. 6.22. Варіанти змінення розміру зображення

Fig. 6.22. Image resizing options

Функція повороту зображення. Для синтезу процедури повороту зображення в бібліотеці ***OpenCV*** формується матриця повороту `cv2.getRotationmatrix2d` і функція афінних перетворень `cv2.warpAffine (img,rotation_matrix(num_cols,num_rows))`.

Створюється зображення `D`, що відповідає повернутому вихідному зображенню `S`, з використанням

Image rotation. For the synthesis of the image rotation procedure in the ***OpenCV*** library, the rotation matrix `cv2.getRotationMatrix2D` and the affine transformation function `cv2.warpAffine(img, rotation_matrix,(num_cols,num_rows))` are formed.

An image `D` is created corresponding to the rotated original image `S` using one of the predefined

одного із методів інтерполяції. Кут повороту задається в градусах. Додатні значення цього параметра відповідають повороту проти годинникової стрілки, а від'ємні – за годинниковою стрілкою.

У загальному випадку кількість пікселів результівного зображення `D` більше або дорівнює кількості пікселів вихідного зображення `S`. Значення пікселів зображення `D`, для яких немає відповідних пікселів у зображенні `S`, установлюються в 0, що для півтонових зображень відповідає чорному кольору.

Результати роботи коду з повороту зображення показано на рис. 6.23.

Приклад:

interpolation methods. The angle of rotation is set in degrees. Positive values of this parameter correspond to the rotation counterclockwise, and negative values – clockwise.

In general, the number of pixels of the resulting image `D` is greater than or equal to the number of pixels of the original image `S`. The pixel values of image `D`, for which there are no corresponding pixels in image `S`, are set to 0, which corresponds to black color for half-tone images.

The results of the code rotation by the image are shown in Fig. 6.23.

Example:

```
import cv2
import numpy as np

img = cv2.imread('16.jpg')
cv2.imshow('Original', img)
num_rows, num_cols = img.shape[:2]

rotation_matrix =
cv2.getRotationMatrix2D((num_cols/2,num_rows/2), 60,1)
img_rotation = cv2.warpAffine(img,rotation_matrix,(num_cols,
num_rows))

cv2.imshow('Rotation',img_rotation)
(cv2.waitKey 0)
```

а

б

Рис. 6.23. Приклад повороту зображення

Fig. 6.23. Image rotation example

Паралельне перенесення зображення. Здійснюється функцією афінного перетворення `cv2.warp-Affne(img,-translation_matrix(num_cols,num_rows))` з використанням матриці паралельного перенесення `(translation_matrix)`, у якій у пікселях задається перенесення вихідного зображення по рядках і стовпцях. Результати цього перетворення показано на рис. 6.24:

Image translation It is implemented by the affine transformation function `cv2.warpAffne(img, translation_matrix, (num_cols,num_rows))` using the parallel translation matrix `(translation_matrix)`, in which the transfer of the original image in columns and columns is specified in pixels. The results of this transformation are shown in Fig. 6.24:

```
import cv2
import numpy as np
img = cv2.imread('1.jpg')
cv2.imshow(' Original ', img)
num_rows, num_cols = img.shape[:2]
translation_matrix = np.float32([ [1,0,70], [0,1,110] ])
img_translation = cv2.warpAffine(img, translation_matrix,
(num_cols, num_rows))
cv2.imshow('Translation', img_translation)
cv2.waitKey()
```

а

Рис. 6.24. Результати паралельного перенесення зображення

б

Fig. 6.24. Results of parallel image transfer

Скіс зображення. Ще один різновид афінного перетворення – скіс прямокутного зображення. Наведемо приклад цього перетворення з використанням функції `cv2.getAffineTransform(src_points,dst_points)` (рис. 6.25, *а*, *б*):

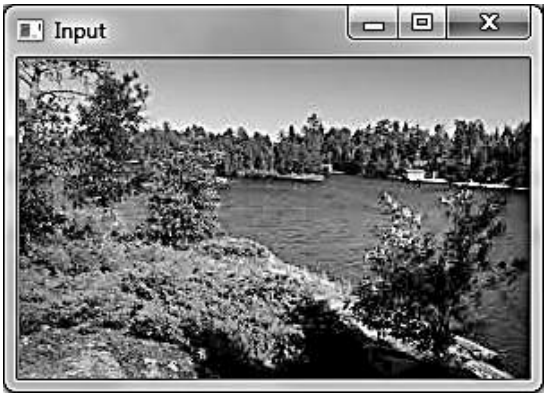
Bevel image. Another type of affine transformation is the bevel of a rectangular image. Let us give an example of this transformation using the function `cv2.getAffine-Tranform(src_points,dst_points)` (Fig. 6.25, *a*, *b*):

```
import cv2
import numpy as np

img = cv2.imread('21.jpg')
rows, cols = img.shape[:2]

src_points = np.float32([[0,0], [cols-1,0], [0,rows-1]])
dst_points = np.float32([[0,0], [int(0.6*(cols-1)),0],
[int(0.4*(cols-1)),rows-1]])
affine_matrix = cv2.getAffineTransform(src_points, dst_points)
img_output = cv2.warpAffine(img, affine_matrix, (cols,rows))

cv2.imshow('Input', img)
cv2.imshow('Output', img_output)
cv2.waitKey()
```

Для створення дзеркального відображення в наведеному коді необхідно зробити таку заміну (рис. 6.25, *в, г*):

To create a mirror image in the above code, it is necessary to make the following replacement (Fig. 6.25, *c, d*):

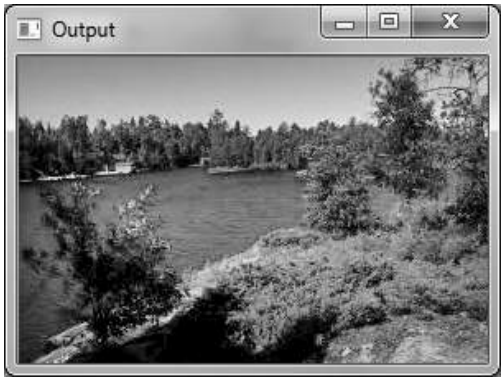
```
src_points = np.float32([[0,0],[cols-1,0],[0,rows-1]])
dst_points = np.float32([[cols-1,0],[0,0],[cols-1,rows-1]])
```



а

б

в



г

Рис. 6.25. Скіс зображення і створення його дзеркальної копії

Fig. 6.25. Image Bevel and Mirror

Проективні перетворення. Як і при афінних перетвореннях, прямі лінії залишаються прямими, проте паралельні перетворюються на непаралельні з видаленою точкою перетину.

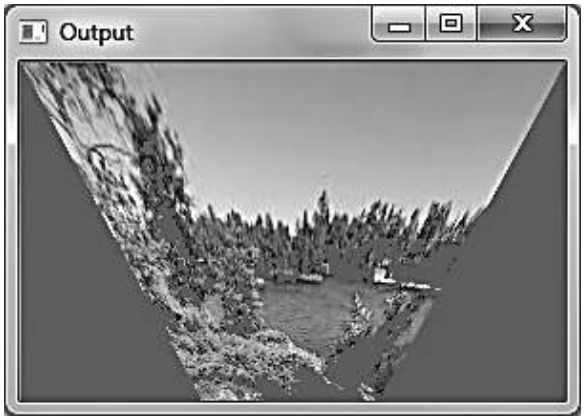
Наведемо приклад програмного коду проективного перетворення зображення (рис. 6.26):

Projective transformations. As with affine transformations, straight lines remain straight, but parallel lines become non-parallel with a distant intersection point.

Let us give an example of the program code of the projective image transformation, the results of which are shown in Fig. 6.26:

```
import cv2
import numpy as np

img = cv2.imread('22.jpg')
rows, cols = img.shape[:2]

src_points = np.float32([[0,0], [cols-1,0], [0,rows-1], [cols-
1,rows-1]])
dst_points = np.float32([[0,0],[cols-
1,0],[int(0.33*cols),rows-1],
[int(0.66*cols),rows-1]])
projective_matrix = cv2.getPerspectiveTransform(src_points,
dst_points)
img_output = cv2.warpPerspective(img, projective_matrix,
(cols,rows))
cv2.imshow('Input', img)
cv2.imshow('Output', img_output)
cv2.waitKey()
```

а



б

Рис. 6.26. Приклад проективного перетворення

Fig. 6.26. An example of a projective transformation

Ще один приклад проективного перетворення (рис. 6.27):

Another example of a projective trans-formation (Fig. 6.27):

```
import cv2
import numpy as np
```

```
img = cv2.imread('21.jpg')
rows, cols = img.shape[:2]

src_points = np.float32([[0,0], [0,rows-1], [cols/2,0],
[cols/2,rows-1]])
dst_points = np.float32([[0,50], [0,rows-51], [cols/2,0],
[cols/2,rows-1]])
projective_matrix = cv2.getPerspectiveTransform(src_points,
dst_points)
img_output = cv2.warpPerspective(img, projective_matrix,
(cols,rows))

cv2.imshow('Input', img)
cv2.imshow('Output', img_output)
cv2.waitKey()
```

а

б

Рис. 6.27. Варіант проективного перетворення

Fig. 6.27. Variant of projective transformation

6.7. Детектування кутів на зображеннях у системах технічного зору

Детектування кутових точок є одним зі способів реалізації комп'ютерного зору. З допомогою цих алгоритмів у кадрі визначають характерні опорні точки для нерухомих об'єктів, які надалі можна використовувати при розрахунку характеристик руху спостережуваних об'єктів. Існує декілька типів таких алгоритмів.

Для реалізації алгоритму пошуку кутів Харріса необхідно виміряти зміну інтенсивності пікселя

6.7. Detection of angles on images in vision systems

Detection of corner points is one of the ways to implement computer vision. Using these algorithms in the frame, the characteristic reference points for fixed objects are determined, which can later be used to calculate the motion characteristics of the observed objects. There are several types of such algorithms.

To implement the Harris angle search algorithm, it is necessary to measure the change in pixel inten-

(x, y) з допомогою зсуву невеликого квадратного вікна W розмірністю 3x3 з центром у (x, y) на один піксель у кожному з восьми можливих напрямків. Таку операцію слід виконати з кожним пікселем на зображенні. Для кожного пікселя зміна інтенсивності обчислюється за формулою

sity (x, y) by offsetting a small square window W of 3x3 dimension centered at (x, y) by one pixel in each of eight possible directions. This operation should be done with each pixel in the image. For each pixel, the intensity change is calculated by the formula

$$E(u, v) = \sum_{(x,y)\in W} w(x, y)[I(x + u, y + v) - I(x, y)]^2.$$

Функція $W(x, y)$ є функцією вікна W. Якщо значення $W(x, y) = 1$, то піксель з координатами (x, y) знаходиться в межах вікна, якщо ж $W(x, y) = 0$, – то за його межами.

Запишемо формулу в матричному вигляді:

The function $W(x, y)$ is a function of the window W. If the value of $W(x, y) = 1$, then the considered pixel with coordinates (x, y) is within the window, if $W(x, y) = 0$, then beyond it.

We write the formula in matrix form:

$$E(u, v) = [u, v]\left(\sum \begin{matrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{matrix}\right)\left(\frac{u}{v}\right).$$

Для невеликого $[u, v]$ отримуємо таке:

For a small $[u, v]$ we get the following:

$$E(u, v) \cong [u, v]M\begin{bmatrix} u \\ v \end{bmatrix}, \qquad M = \sum w(x, y)\begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}.$$

Для кутових точок є характерними великі за модулем власні значення матриці M.

Розрахунок власних значень цієї матриці є досить трудомістким завданням, тому Харріс додатково ввів міру відгуку

For the corner points, the matrix M is large in modulus to the eigenvalues.

Considering the eigenvalues of this matrix is a rather laborious task, so Harris additionally introduced a measure of response

$$R = detM - k(traceM)^2,$$

де k визначається емпірично і найчастіше становить від 0.04 до 0.15. Таким чином, значення R є великим і додатним для кутових точок.

Після знаходження всіх кутових

where the value of k is determined empirically and most often lies in the range from 0.04 to 0.15. Thus, the value of R is large and positive for corner points.

After finding all the corner points,

точок відбувається їх відсікання (відсікаються лише ті точки, значення R яких менше заданого порогу).

Детектор Харріса є інваріантним до поворотів і частково інваріантним до афінних перетворень. Проте його недоліком можна вважати залежність від масштабу.

Подібним чином працює й детектор кутів Ши-Томасі. Проте через стислий виклад материалу принцип його роботи розглядати не будемо, а залишаємо це на самостійний розгляд читача.

Наведемо приклади програмних кодів для детекторів кутів Харріса і Ши-Томасі. Результати роботи цих алгоритмів показано на рис. 6.28 і 6.29.

Детектор кутів Харріса. Далі наведено приклад програмного коду, що реалізує алгоритм Харріса (рис. 6.28):

they are clipped (only those points whose R value is less than the specified threshold are cut off).

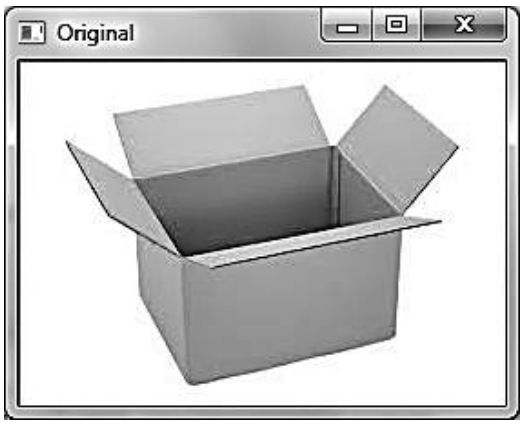
The Harris detector is invariant to rotations and partially invariant to affine transformations. However, its disadvantage can be considered the dependence on scale.

The Shi-Tomasi angle detector works in a similar way. However, for the sake of brevity, we will not consider the principle of its work, but leave it to the reader for independent consideration.

We give examples of program codes for the Harris and Shea-Tomasi angle detectors. The results of these algorithms are shown in Fig. 6.28 and 6.29.

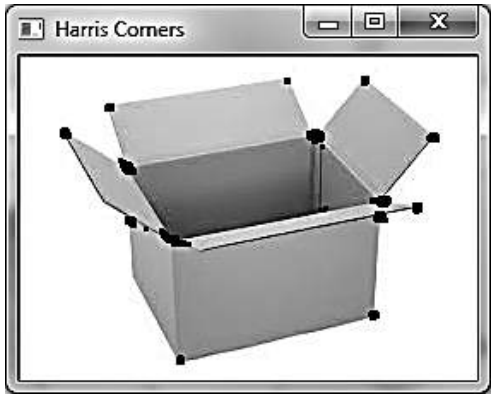
Harris Corner Detector The following is an example of a program code that implements the Harris algorithm (Fig. 6.28):

```
import cv2
import numpy as np

img = cv2.imread('25.jpg')
cv2.imshow('Original', img)
gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)

gray = np.float32(gray)

dst = cv2.cornerHarris(gray, 4,5, 0.04) # to detect only sharp
corners
#dst = cv2.cornerHarris(gray, 14, 5, 0.04) # to detect soft
corners

# Result is dilated for marking the corners
dst = cv2.dilate(dst,None)
# Threshold for an optimal value, it may vary depending on the
image.
img[dst > 0.01*dst.max()] = [0,0,0]

cv2.imshow('Harris Corners',img)
cv2.waitKey()
```



а

6.28. Детектування кутів алгоритмом Харріса



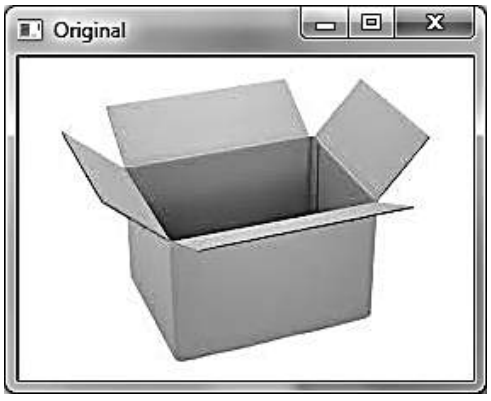
б

6.28. Angle detection by Harris algorithm

Детектор кутів Ши-Томасі. Детектор Ши-Томасі дає аналогічні результати порівняно з алгоритмом Харріса, що видно на рис. 6.29:

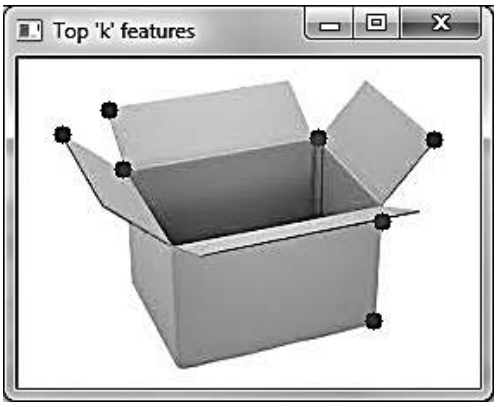
Shi-Tomasi corner detector. The detector Shih-Tomasi gives similar results in comparison with the algorithm of Harris. This is seen in Fig. 6.29:

```
import cv2
import numpy as np

img = cv2.imread('25.jpg')
cv2.imshow('Original', img)
gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)

corners = cv2.goodFeaturesToTrack(gray, 7, 0.05, 25)
corners = np.float32(corners)

for item in corners:
    x, y = item[0]
    cv2.circle(img, (x,y), 5, 255, -1)

cv2.imshow("Top 'k' features", img)
cv2.waitKey()
```



а

Рис. 6.29. Детектування кутів алгоритмом Ши-Томасі



б

Fig. 6.29. Angle detection by Shih-Tomasi algorithm

7. АЛГОРИТМИ OPENCV ДЛЯ ОБРОБЛЕННЯ ВІДЕОДАНИХ

У попередньому розділі було викладено кілька методів оброблення зображень на ***Python*** з використанням різних функцій бібліотеки ***OpenCV***. Порівняльний аналіз ресурсів цієї бібліотеки з можливостями основних функцій і методів пакета модулів бібліотеки ***Pillow*** показує, що вони мають приблизно однакові функціональні можливості.

Це стосується алгоритмів перетворення колірних просторів і фільтрації зображень, методів афінних і проективних перетворень та ін. Не менш ефективним для оброблення зображень є застосування додатків ***Image Processing Toolbox*** програми ***Matlab***.

Однак це помилкове враження. Основною перевагою використання функцій бібліотеки ***OpenCV*** є істотно вища швидкодія.

Це практично непомітно під час оброблення окремих зображень, але при обробленні й аналізі відеопослідовностей цей фактор стає домінуючим. Більш того, у більшості випадків необхідно проводити оброблення в реальному масштабі часу, а це можливо тільки з використанням алгоритмів ***OpenCV***. Розглянемо ці переваги більш детально.

7.1. Читання і запис відеоданих

Розглянемо найбільш затребувані процедури читання відеоданих в програму ***Python*** з допомогою функцій бібліотеки ***OpenCV*** при використанні web-камери персонального комп'ютера, відеока-

7. OPENCV ALGORITHMS FOR PROCESSING VIDEO DATA

The previous section outlined a number of methods for processing images in ***Python*** using various functions of the ***OpenCV*** library. A comparative analysis of the resources of this library with the capabilities of the basic functions and methods of the package of modules of the ***Pillow*** library shows that they have approximately the same functionality.

This applies to algorithms for converting color spaces and image filtering, methods for affine and projective transformations, and so on. It is equally effective for processing images and using ***Matlab's Image Processing Toolbox*** application.

However, this is a mistaken impression. The main advantage of using the functions of the ***OpenCV*** library is significantly higher speed.

This is almost imperceptible when processing individual images, but when processing and analyzing video sequences, this factor becomes dominant. Moreover, in most cases it is necessary to carry out processing in real time, and this is possible only using ***OpenCV*** algorithms. Consider these advantages in more detail.

7.1. Reading and recording video

Consider the most popular procedures for reading video data into the ***Python*** program using the ***OpenCV*** library functions when using a personal computer web-cam,

мер з ***USB***-підключенням або збережених раніше відеофайлів. У цьому розділі також опишемо процедури збереження відеоданих (записування на диск).

Процедура читання відео досить проста, оскільки ***OpenCV*** автоматично підтримує різні формати відеоданих (mp4, avi та ін.). Для забезпечення різних варіантів читання використовується об'єкт модуля `VideoCapture()`.

Як аргумент використовується ціле число або ім'я файлу. Цілочисловий аргумент цього модуля є ідентифікатором підключеної до комп'ютера камери, а ім'я файлу використовується для захвату даних записаного раніше відеофайлу. Вбудована web-камера зазвичай має ідентифікатор 0.

Тому, щоб успішно прочитати відео з web-камери вашого Notebook, використовуйте `VideoCapture(0)`. Усі інші USB-камери будуть мати ідентифікатори, що починаються з 1.

Читання відео з web-камери. Наведемо приклад коду для захоплення відео з web-камери Notebook і покажемо вікно перегляду відеоданих (рис. 7.1):

USB-connected camcorders or previously saved video files. In this section, we also describe the procedures for storing video data (recording to disk).

The procedure for reading video is quite simple, since ***OpenCV*** automatically supports various formats of video data (mp4, avi, etc.). To provide different read options, the `VideoCapture()` module object is used.

An integer or file name is used as an argument. The integer argument of this module is the identifier of the camera connected to the computer, and the file name is used to capture the data of the previously recorded video file. The built-in webcam usually has ID 0.

Therefore, to successfully read video from your Notebook's webcam, use `VideoCapture(0)`. All other USB cameras will have identifiers starting with 1.

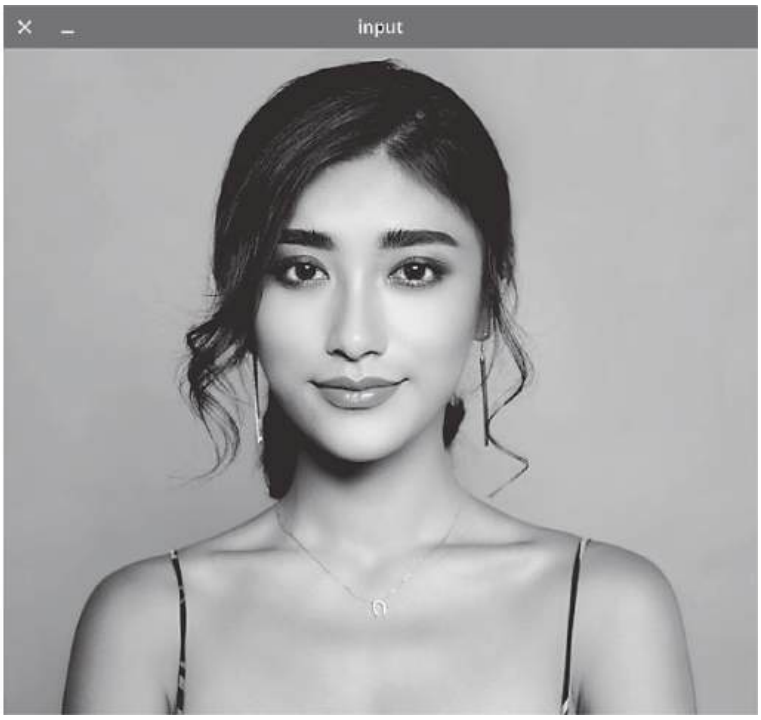
Reading video from webcam. Here is an example code for capturing video from a Notebook webcam and a video viewing window (Fig. 7.1):

```
import cv2

cap = cv2.VideoCapture(0)
# Check if the webcam is opened correctly
if not cap.isOpened():
    raise IOError("Cannot open webcam")
while True:
    ret, frame = cap.read()
    frame = cv2.resize(frame,None,fx=1.0,fy=1.0,interpolaeion
= cv2.INTER_AREA)
        cv2.imshow('Input', frame)
         c = cv2.waitKey(1)
```

```
    if c == 27:
        break
cap.release()
cv2.destroyAllWindows()
```

У цьому коді використовуються функції `cam = cv2.Video Capture(0)` і `read()`. Остання функція повертає кадри відео й контролює успішність читання кадру (логічне значення є правильним `(value is true)` у разі, якщо кадр був успішно прочитаний, і неправильним `(value is false)` в іншому випадку).

This code uses the `cam = cv2.VideoCapture(0)` and `read()` functions. The latter returns video frames and controls the success of the frame reading (the boolean value is true `(value is true)` if the frame was successfully read, and the `(value is false)` otherwise).



7.1. Вікно перегляду web-камери

7.1. Web-cam view window

Ще одним важливим фактором є масштаб відображення поточного фрейма при перегляді відеопослідовності. Для цього необхідно в аргументі функції `cv2.resize (frame,None,fx = 0.5, fy = 0.5, ...)` указати потрібні вам розміри (наприклад, `fx = 1.0, fy = 1.0`).

Цикл читання відеоданих у цьому коді `ret,frame =`

The second important factor is the scale of the display of the current frame when viewing a video sequence. To do this, you need to specify the dimensions you need in the function argument `cv2.resize (frame, None, fx = 0.5, fy = 0.5, ...)` (for example, `fx=1.0, fy = 1.0`).

The cycle of reading video data in this code `ret, frame =`

`cap.read()` триватиме доти, доки не відбудеться натискання клавіші Esc на клавіатурі − це перерве процедуру відеозахвату. Ця дія в програмі відображається умовою `if c == 27`. Як відомо, `ASCII` значення `Esc` становить `27`. Команда `cap.release()` закриває web-камеру.

Читання відео з файлу. Щоб прочитати відеодані з файлу, зазвичай використовується аналогічний код:

`cap.read()` will continue until the `Esc` key on the keyboard is pressed − this will interrupt the video capture procedure. This action is displayed in the program by the condition `if c == 27`. As you know, the `Esc ASCII` value is 27. The `cap.release()` command closes the webcam.

Reading video from file. To read video data from a file, similar code is usually used:

```
import cv2

cap = cv2.VideoCapture("blue_ball.avi")

# Check if the webcam is opened correctly
if not cap.isOpened():
    raise IOError("Cannot open webcam")

cap.set(cv2.CAP_PROP_POS_FRAMES,300)

while True:
    ret, frame = cap.read()
    frame = cv2.resize(frame, None, fx=1.0, fy=1.0, interpola-
tion=cv2.INTER_AREA)

    cv2.imshow('Input', frame)

    c = cv2.waitKey(1)
    if c == 27:
        break

cap.release()
cv2.destroyAllWindows()
```

Вікно для перегляду тестового відеофайлу `blue_ball` показано на рис. 7.2. У ньому відображається рух синьої кульки на світло-сірому фоні. Зазначимо, що при вказівці імені файлу необхідно обов'язково вказати й формат відеоданих. У нашому випадку це формат `avi`. Ще одна особливість

The window for viewing the `blue_ball` test video file is shown in Fig. 7.2. It displays the movement of a blue ball on a light gray background. Note that when specifying the file name, it is necessary to specify the format of the video data. In our case, this is `avi` format. Another feature is that the

полягає в тому, що вихідний відеофайл потрібно помістити в системну папку ***Python,*** щоб уникнути помилок при вказівці шляху до цього файлу.

У деяких випадках виникає необхідність читання даних не з першого, а з більш пізнього фрейма відеопослідовності. Тоді в функції установок `ap.set (cv2.CAP_PROP_POS_FRAMES,300)` необхідно вказати номер фрейма, який є точкою входу в читання відеоданих. У нашому прикладі це 300-й кадр.

source video file must be placed in the ***Python*** system folder to avoid errors when specifying the path to this file.

In some cases, it is necessary to read data not from the first, but from a later frame of the video sequence. Then in the settings function `ap.set (cv2.CAP_PROP_POS_FRAMES,300)` you need to specify the frame number, which is the entry point to the reading of video data. In our example, this is the 300th frame.

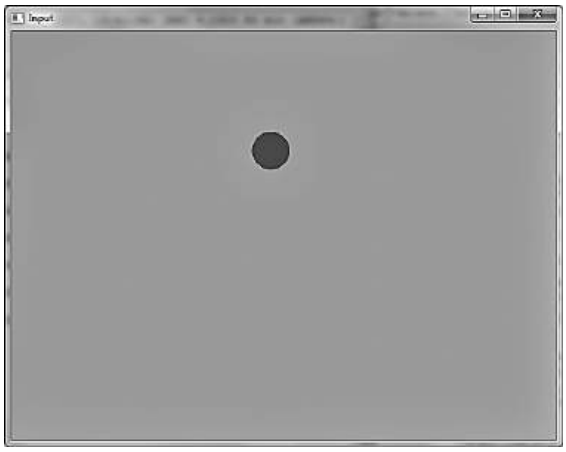


Рис. 7.2. Вікно перегляду відеофайлу

Fig. 7.2. Browse window video file

Запис відеофайлу з серії зображень з допомогою бібліотеки ***OpenCV*** мовою ***Python*** здійснюється за допомогою об'єкта `VideoWriter`, який викликається із заздалегідь визначеними параметрами. Ця назва відеофайлу (для збереження), `fourcc` (відеокодек), `fps` (частота зміни кадрів), `size` (розміри фреймів). Наведемо приклад зчитування відеопослідовності з web-камери з одночасним збереженням її в файл з допомогою функції `VideoWriter`:

Recording a video file from a series of images using the ***OpenCV*** library in ***Python*** is done using the `VideoWriter` object, which is called with predefined parameters. This is the name of the video file (`for saving`), `fourcc` (`video codec`), `fps` (frame rate), `size` (`frame size`). Here is an example of reading a video sequence from a webcam while simultaneously saving it to a file using the `VideoWriter` function:

```
import cv2
cam = cv2.VideoCapture(0)
```

```
ret, frame = cam.read()

h, w = frame.shape[:2]
fourcc = cv2.VideoWriter_fourcc(*'DIVX')
video_write =
cv2.VideoWriter('saved_out.avi',fourcc,30.0,(640,480))

while (cam.isOpened()):

        ret, frame = cam.read()
        video_write.write(frame)
        cv2.imshow('video',frame)

        c = cv2.waitKey(1)
        f c == 27:
                break
cam.release()
video_write.release()

cv2.destroyAllWindows()
```

У цьому коді спочатку створюється об'єкт `video_write` функції `VideoWriter` і аргументи, передані в `VideoWriter`. У нашому прикладі ім'я файлу `'saved_out.avi'`, `fps` (30 фреймів/с), розмір кадру (640x480), `fourcc` – чотиризначний код, який використовується для подання форматів стиснення. Після визначення об'єкта `VideoWriter` читається кадр за кадром і записується кожен кадр з допомогою функції `write()` у вихідний файл. Зазначимо, що читання відео з web-камери і синхронне його збереження дають змогу ефективно документувати результати відеоспостереження для подальшого аналізу.

7.2. Оброблення відео з web-камери в реальному часі

Розглянемо кілька прикладів оброблення в програмі ***Python*** з використанням функцій бібліотеки ***OpenCV***. За бажання роботу наведених кодів читач може легко

In this code, the `VideoWriter` function is first created with the `video_write` object and the arguments passed to the `VideoWriter`. In our example, the file name is `'saved_out.avi'`, fps (30 frames/s), frame size (640x480),. `fourcc` is a four-digit code used to represent compression formats. After determining the `VideoWriter` object, it is read frame by frame and is written each frame using the `write()` function in the output file. Note that reading video from a web-camera and its synchronous storage allows you to effectively document the results of video surveillance for later analysis.

7.2. Processing video from a webcam in real time

Let's look at a few examples of video processing in a ***Python*** program using the functions of the ***OpenCV*** library. If desired, the reader can easily check the above

перевірити на своєму комп'ютері й переконатися, що оброблення проводиться в реальному масштабі часу.

Перетворення колірних просторів, виділення меж об'єктів. Наведемо приклад реєстрації відео з web-камери з перетворенням повнокольорового зображення кадрів на півтонове й виділення меж об'єктів у кадрі алгоритмом Канні. Код цієї програми наведено нижче, а результати оброблення показано на рис. 7.3:

codes on his computer and make sure that the processing is carried out in real time.

Conversion of color spaces, selection of object boundaries. Let us give an example of registering video from a web-camera with the conversion of a full-color image of frames into halftone and the selection of the borders of objects in a frame by the Canny algorithm. The code for this program is shown below, and the processing results are shown in Fig. 7.3:

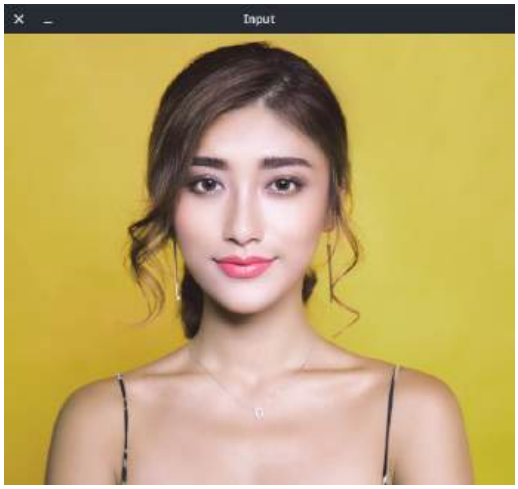
```
import cv2
cap = cv2.VideoCapture(0)
# Check if the webcam is opened correctly
if not cap.isOpened():
    raise IOError("Cannot open webcam")
while True:
    ret, frame = cap.read()
    frame = cv2.resize(frame, None, fx=1.0, fy=1.0, interpola-
tion=cv2.INTER_AREA)
    cv2.imshow('Input', frame)

    frame_gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    cv2.imshow('Gray',frame_gray)
    edges = cv2.Canny(frame_gray,0,99)
   cv2.imshow('Edges',edges)
        c = cv2.waitKey(1)
    if c == 27:
        break
cap.release()
cv2.destroyAllWindows()
```

У цьому програмному коді також здійснюється перетворення вихідної відеопослідовності на півтонову з допомогою функції `frame_gray=cv2.cvtColor(frame ,cv2.COLOR_BGR2GRAY)` і виділення меж об'єктів у кадрі з допомогою функції `edges = cv2.Canny(frame_gray,thresh)`. Нагадаємо, що параметр `thresh`

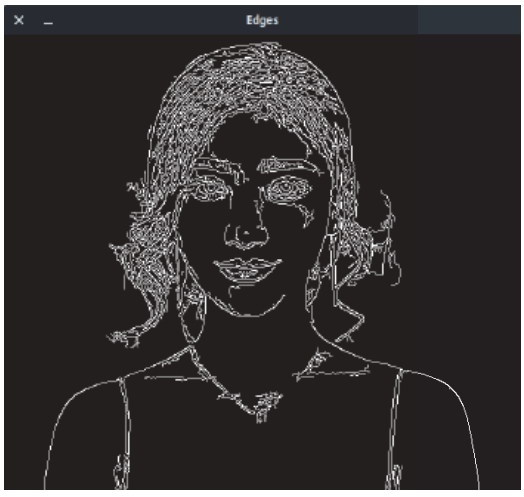
This program code also converts the original video sequence into a grayscale using the `frame_gray =cv2.cvtColorfunction(frame, cv2.COLOR_BGR2GRAY)` and extracting the borders of objects in the frame using the `edges = cv2.Canny(frame_gray,thresh)` function. Recall that the thresh parameter determines the degree of

визначає ступінь деталізації контурів меж розділу об'єкта і навколишнього фону.

Зверніть увагу, що всі процедури захоплення відображення й перетворення прийнятої з web-камери відеопослідовності виконуються в реальному масштабі часу. Це свідчить про високу ефективність використання функцій бібліотеки ***OpenCV***.



а



в

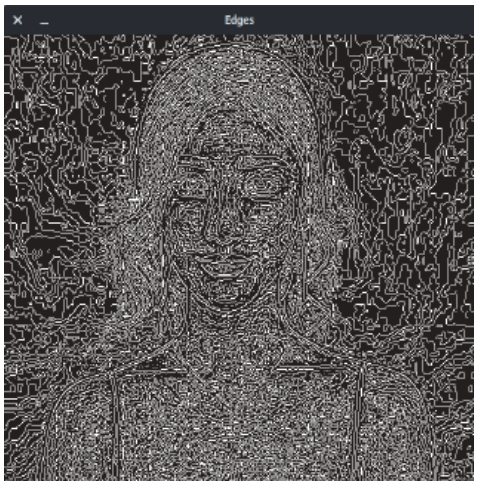
Рис. 7.3. Перетворення фрейма відеофайлу: *а* − вихідний, *б* − півтоновий, *в* − контури об'єкта thresh = 0,99, *г* − контури об'єкта thresh = 0,99

Геометричні перетворення відеоданих. У розділі 3 було показано можливості бібліотеки

detail of the contours of the interface of the object, and the surrounding background

Please note that all procedures for capturing the display and transformation of a video sequence received from a webcam are performed in real time. This indicates the high efficiency of using the functions of the ***OpenCV*** library.

б



г

Fig. 7.3. Frame transformations of the video deofile: *a* – source, *b* – halftone, *c* – object outlines `thresh = 0.99`, *d* – `object outlines thresh = 0.99`

Geometric transformation of video data. In Chapter 3, the capabilities of the ***OpenCV*** library for per-

OpenCV щодо виконання геометричних (афінних і проективних) перетворень зображень − коротко викладено теоретичні відомості й наведено приклади побудови кодів, досить докладно описано синтаксис використовуваних функцій. Розширимо можливості використання таких алгоритмів та узагальнимо ці методи на оброблення відеопослідовностей. Для цього наведемо кілька прикладів (рис. 7.4):

forming geometric (affine and projective) image transformations were shown - theoretical information was briefly presented and a number of examples of building codes describing the syntax of functions used were described in detail. We will expand the possibilities of using such algorithms and generalize these methods to the processing of video sequences. For this we give a number of examples (Fig. 7.4):

```
import cv2
import numpy as np

cap = cv2.VideoCapture(0)

# Check if the webcam is opened correctly
if not cap.isOpened():
    raise IOError("Cannot open webcam")

while True:
    ret, frame = cap.read()
    frame = cv2.resize(frame,
None,fx=1.0,fy=1.0,interpolation=cv2.INTER_AREA)

    cv2.imshow('Input',frame)

    num_rows, num_cols = frame.shape[:2]

    translation_matrix = np.float32([[1,0,70],[0,1,100]])
    frame_translation = cv2.warpAffine(frame, transla-
tion_matrix, (num_cols,num_rows))

    cv2.imshow('Translation', frame_translation)

    c = cv2.waitKey(1)
    if c == 27:
        break

cap.release()
cv2.destroyAllWindows()
```

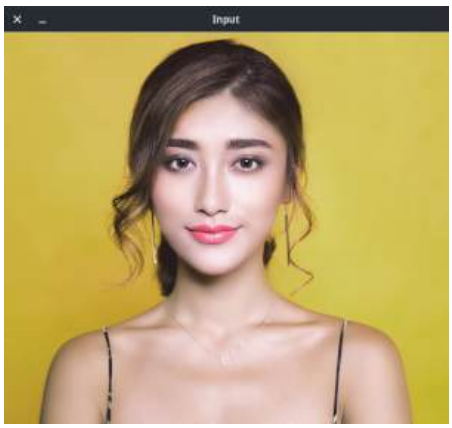


Рис. 7.4. Паралельний перенос відеоданих

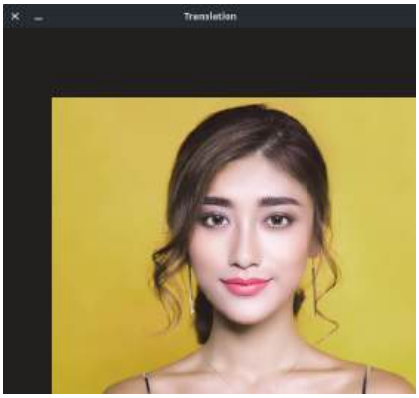


Fig. 7.4. Parallel video transfer

Поворот кадрів відео-1. Під час оброблення кадрів часто виникає необхідність повороту зображень кадрів відносно їх центра на потрібний кут за годинниковою стрілкою або проти годинникової стрілки.

Ця процедура легко реалізується з допомогою функції повороту ***OpenCV***. Програмний код такого оброблення наведено нижче, а результати повороту кадру показано на рис. 7.5:

Rotate video frames−1. In the practice of video processing, it is often necessary to rotate images of frames relative to their center at a desired angle clockwise or counterclockwise.

This procedure is easily implemented using the ***OpenCV*** rotation feature. The program code of such processing is given below, and the results of frame rotation are shown in Fig. 7.5:

```
import cv2
import numpy as np

cap = cv2.VideoCapture(0)

# Check if the webcam is opened correctly
if not cap.isOpened():
    raise IOError("Cannot open webcam")

while True:
    ret, frame = cap.read()
    frame = cv2.resize(frame,
None,fx=0.5,fy=0.5,interpolation=cv2.INTER_AREA)

    cv2.imshow('Input',frame)

    num_rows,num_cols = frame.shape[:2]
    rotation_matrix = cv2.getRotationMatrix2D((num_cols/2,
num_rows/2),10,1)
    frame_rotation = cv2.warpAffine(frame,rotation_matrix,
(num_cols,num_rows))

    cv2.imshow('Rotation',frame_rotation)
```

```
    c = cv2.waitKey(1)
    if c == 27:
        break

cap.release()
cv2.destroyAllWindows()
```

У цьому коді формується матриця повороту `rotation_matrix`, у якій задається потрібний кут повороту, а далі ця матриця використовується як аргумент функції афінних перетворень `cv2.warpAffine(frame,rotation_matrix(num_cols,num_rows))`.

На жаль, цей метод перетворення має великий недолік – при збереженні розмірів кадру «обрізаються» кути зображення, що може призвести до часткової втрати інформації. Шляхи запобігання цьому показано в такому прикладі.

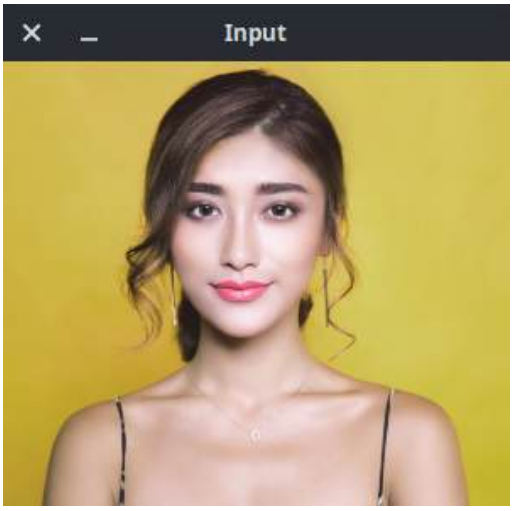


Рис. 7.5. Поворот кадрів відео на 10° проти годинникової стрілки

Поворот кадрів відео-2. У цьому випадку попередньо формується дві матриці – `translation_matrix` і `rotation_ matrix`, що використовуються далі як аргументи афінної функції зсуву

In this code, the `rotation_matrix` rotation matrix is formed, in which the desired rotation angle is specified, and then this matrix is used as an argument of the function of affine transformations `cv2.warpAffine(frame,rotation_matrix(num_cols,num_rows))`.

Unfortunately, this method of transformation has a big disadvantage – while maintaining the frame size, the corners of the image are “cropped”, which can lead to partial loss of information. Ways to overcome this drawback are shown in the following example.

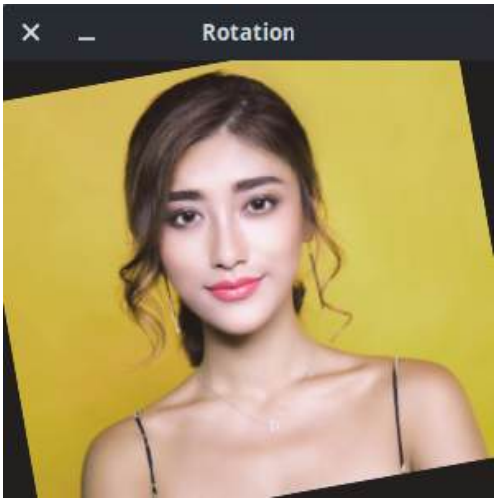


Fig. 7.5. Rotate video frames by 10^0 counterclockwise

Frame rotation–2. In this case, two matrices are preliminarily formed – `translation_matrix` and `rotation _matrix`, used further as arguments of the affine shift function `cv2.warpAffine`

cv2.warpAffine(frame,transla-tion_matrix), і виконується далі функція кутового повороту cv2.warpAffine(frame_translation,rotation_matrix).

Комбінація таких перетворень дає змогу не тільки здійснити поворот зображення кадра на потрібний кут, але й уникнути втрати інформації через «обрізання» кутів зображення. Щоправда, при цьому збільшуються розміри вихідного фрейма, а вільні зони заповнюються нульовими пікселями. Далі наведено повний програмний код цього алгоритму, а результати перетворення показано на рис. 7.6:

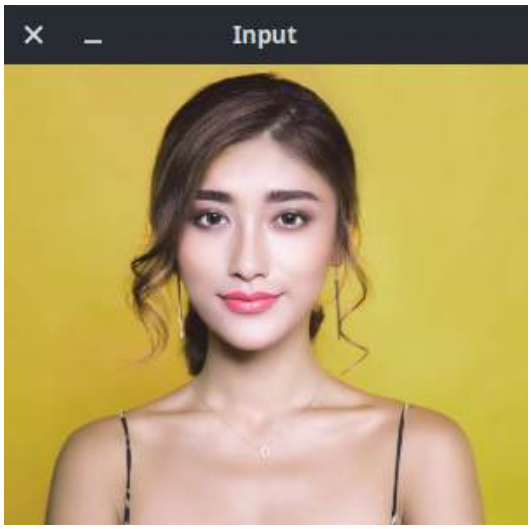
(frame,transla-tion_ matrix), and the further function of angular rotation cv2.warpAffine (frame_ translation, rotation_matrix).

The combination of such transforma-tions allows not only to rotate the image of the frame at the desired angle, but also to avoid loss of information due to the "cropping" of the image angles. True, this increases the size of the original frame, and the free zones are filled with zero pixels. The following is the full program code of this algorithm, and the conversion results are shown in Fig. 7.6:

```
import cv2
import numpy as np

cap = cv2.VideoCapture(0)

# Check if the webcam is opened correctly
if not cap.isOpened():
    raise IOError("Cannot open webcam")

while True:
    ret, frame = cap.read()
    frame = cv2.resize(frame,
None,fx=0.5,fy=0.5,interpolation=cv2.INTER_AREA)

    cv2.imshow('Input',frame)

    num_rows,num_cols = frame.shape[:2]
    translation_matrix =
np.float32([[1,0,int(0.5*num_cols)],[0,1,int(0.5*num_rows)]])

    rotation_matrix =
cv2.getRotationMatrix2D((num_cols,num_rows),30,1)
    frame_translation =
cv2.warpAffine(frame,translation_matrix,(2*num_cols,
2*num_rows))
    frame_rotation =
cv2.warpAffine(frame_translation,rotation_matrix,
(2*num_cols,2*num_rows))
```

```
    cv2.imshow('Rotation',frame_rotation)

    c = cv2.waitKey(1)
    if c == 27:
        break

cap.release()
```

Наведені приклади геометричних перетворень відеоданих показують, що такі завдання з використанням стандартних функцій бібліотеки ***OpenCV*** не становлять труднощів при створенні програмних кодів і дають можливість роботи в реальному масштабі часу.



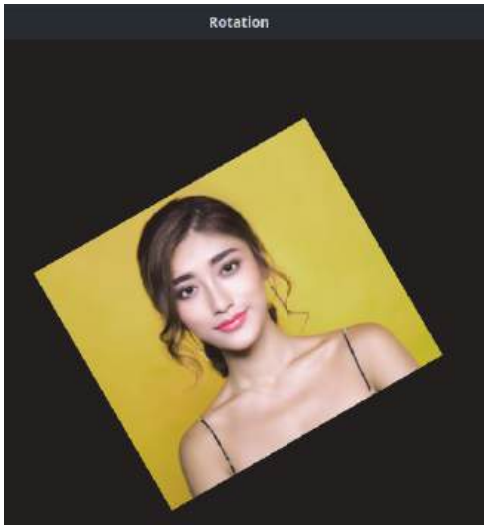
а

Рис. 7.6. Поворот кадрів відео без обрізання кутів зображення

7.3. Оброблення відео в інтерактивному режимі

Наведений раніше приклад коду, що забезпечує виділення контурів об'єктів за методом Канні, дає змогу ефективно вирішити поставлене завдання, але має істотний недолік − при зміні умов освітлення сцени необхідно відрегулювати параметр `thresh`, а отже, потрібно перервати роботу програми й увести нові значення цього параметра. Набагато зручніше було

The examples of geometric transformations of video data show that such tasks using the standard functions of the ***OpenCV*** library do not cause difficulties when creating program codes, and provide an opportunity to work in real time.



б

Fig. 7.6. Rotate video frames without cropping image corners

7.3. Online video processing

The earlier code example, which provides the selection of contours of objects using the Canny method, makes it possible to effectively solve the set problem, but has significant disadvantages − when the lighting conditions of the scene change, the `thresh` parameter must be adjusted, and therefore the program must be interrupted and It would be much more convenient to

б здійснювати регулювання параметрів оброблення відеоданих інтерактивно, не перериваючи процесу реєстрації.

Наведемо програмний код, який виконує вирішення такого завдання: детектування вихідної відеопослідовності шляхом виділення меж об'єктів за методом Канні (рис. 7.7). Регулювання значень параметра `thresh` проводиться інтерактивно з використанням функцій `thrs1=cv2.getTrackbarPos('thrs1','edge')` і `thrs2=cv2.getTrackbarPos('thrs2','edge')`. Це так звані «трекбари» – смужкові регулятори, які забезпечують плавне змінення необхідних параметрів. У нашому випадку вони дають змогу інтерактивно регулювати ступінь деталізації фрагментів зображення під час виділення меж об'єктів.

Інтерактивне детектування меж у відеопотоці. У наведеному нижче програмному коді для створення движків-регуляторів використовуються функції `cv2.create Trackbar`:

```
import cv2
import numpy as np

# relative module
import video

# built-in module
import sys

if __name__ == '__main__':
    print(__doc__)
    try:
        fn = sys.argv[1]
    except:
        fn = 0
```

adjust the video processing parameters interactively, without interrupting the registration process.

Let us describe the program code that accomplishes the solution of such a task — the source video sequence is detected by the method of selecting the Canny object boundaries (Fig. 7.7). The thresh parameter is adjusted interactively using the functions `thrs1=cv2.getTrackbarPos('thrs1', 'edge')` and `thrs2=cv2.getTrackbarPos('thrs2', 'edge')`. These are the so-called “Trackbars” – strip regulators that ensure the smooth change of the required parameters. In our case, they allow you to interactively adjust the level of detail of image fragments when selecting the boundaries of objects.

Interactive border detection in a video stream. In the above program code, the `cv2.create Trackbar` functions are used to create the controller engines:

```
def nothing(*arg):
    pass

cv2.namedWindow('edge')
cv2.createTrackbar('thrs1', 'edge', 2000, 5000, nothing)
cv2.createTrackbar('thrs2', 'edge', 4000, 5000, nothing)

cap = video.create_capture(fn)
while True:
    flag, img = cap.read()
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    thrs1 = cv2.getTrackbarPos('thrs1', 'edge')
    thrs2 = cv2.getTrackbarPos('thrs2', 'edge')
    edge = cv2.Canny(gray,thrs1,thrs2,apertureSize=5)
    vis = img.copy()
    vis = np.uint8(vis/2.)
    vis[edge != 0] = (0, 255, 0)
    cv2.imshow('edge', vis)
    ch = cv2.waitKey(5)
    if ch == 27:
        break
cv2.destroyAllWindows()
```

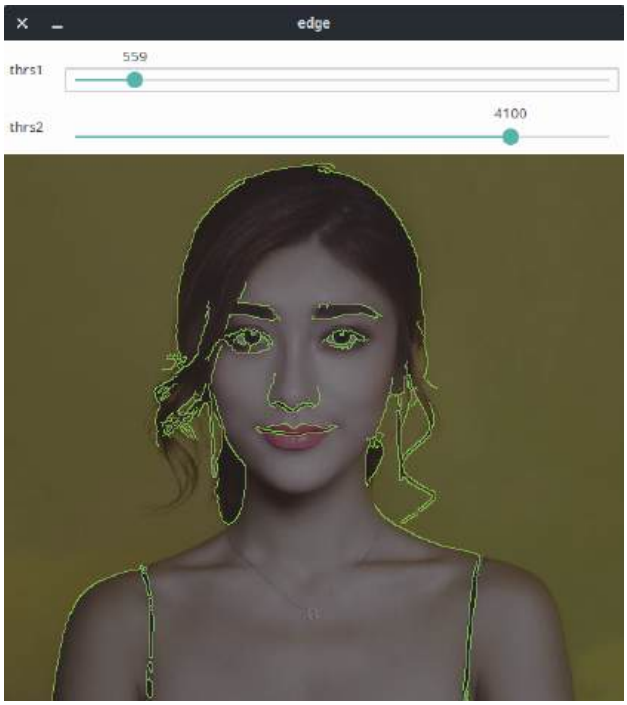


Рис. 7.7. Інтерактивне детектування меж об'єктів за методом Канні

Fig. 7.7. Interactive detection of object boundaries using the Canny method

Відображення області відео в негативному вигляді. Наведемо ще один приклад інтерактивної взаємодії із прийнятим відеопотоком. У цьому випадку після

Displays the video area in a negative way. Let's give one more example of interactive interaction with the received video stream. In this case, after capturing the video

захоплення відео і початку відображення з допомогою мишки будемо виділяти в поточному кадрі область інтересу і вести відображення в цій області у вигляді негативного зображення (рис. 7.8).

Приклад:

and starting to display with the mouse, we will select the area of interest in the current frame and display in this area as a negative image (Fig. 7.8).

Example:

```
import cv2
import numpy as np

def draw_rectangle(event, x, y, flags, params):
    global x_init, y_init, drawing, top_left_pt, bot-
    tom_right_pt

    if event == cv2.EVENT_LBUTTONDOWN:
        drawing = True
        x_init, y_init = x, y

    elif event == cv2.EVENT_MOUSEMOVE:
        if drawing:
            top_left_pt = (min(x_init, x), min(y_init, y))
            bottom_right_pt = (max(x_init, x), max(y_init, y))
            img[y_init:y, x_init:x] = 255 - img[y_init:y,
    x_init:x]

    elif event == cv2.EVENT_LBUTTONUP:
        drawing = False
        top_left_pt = (min(x_init, x), min(y_init, y))
        bottom_right_pt = (max(x_init, x), max(y_init, y))
        img[y_init:y, x_init:x] = 255 - img[y_init:y,
    x_init:x]

if __name__=='__main__':
    drawing = False
    top_left_pt, bottom_right_pt = (-1,-1), (-1,-1)

    cap = cv2.VideoCapture(0)

    # Check if the webcam is opened correctly
    if not cap.isOpened():
        raise IOError("Cannot open webcam")

    cv2.namedWindow('Webcam')
    cv2.setMouseCallback('Webcam', draw_rectangle)

    while True:
        ret, frame = cap.read()
        img = cv2.resize(frame, None, fx=0.5, fy=0.5,
    interpolation=cv2.INTER_AREA)
        (x0,y0), (x1,y1) = top_left_pt, bottom_right_pt
```

```
            img[y0:y1, x0:x1] = 255 - img[y0:y1, x0:x1]
            cv2.imshow('Webcam', img)

            c = cv2.waitKey(1)
            if c == 27:
                break

    cap.release()

      cv2.destroyAllWindows()
```

Зазначимо, що в нашому випадку достатньо клацнути лівою кнопкою мишки поза полем області інтересу для скасування режиму перегляду в ній негативного зображення. Потім, не перериваючи процесу захоплення відеопотоку, можна виділити нову область інтересу й переглядати негативне зображення в ній далі.

Note that in our case, it is enough to click the left mouse button outside the field of interest to cancel the view of the negative image in it. Then, without interrupting the process of capturing the video stream, you can select a new area of interest and view the negative image in it further.

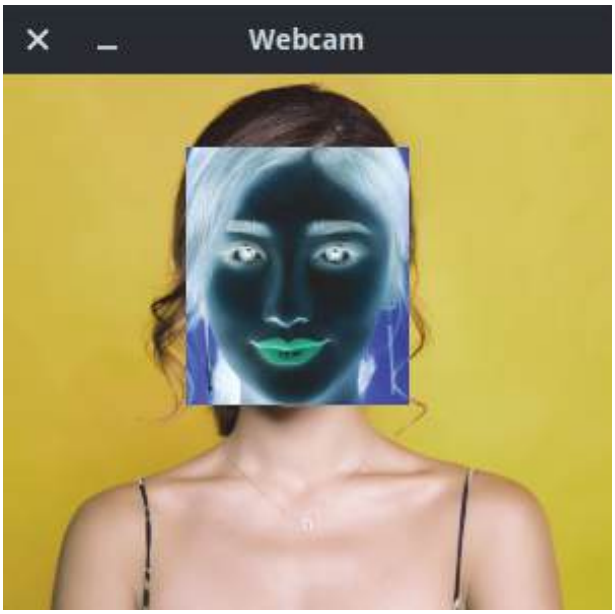


Рис. 7.8. Інтерактивне виділення області інтересу й відображення в ній негативного відео

Fig. 7.8. Interactively highlighting a region of interest and displaying a negative video in it

7.4. Пошук облич з використанням OpenCV

Наведені в цьому підрозділі приклади оброблення облич з використанням ресурсів бібліотеки ***OpenCV*** наочно демонструють великі можливості оброблення ві-

7.4. Search for people using OpenCV

The examples of video data processing in this chapter using the resources of the ***OpenCV*** library clearly demonstrate the great possibilities of video processing in real

део в реальному масштабі часу. Це дає змогу ефективно використовувати отримані результати для розв’язання широкого кола прикладних задач у таких областях, як цифрове оброблення зображень, комп'ютерний зір, біометрія, створення інтелектуальних систем безпеки, спостереження й контролю доступу до об'єктів та ін. У широкому розумінні − це завдання розпізнавання образів.

У цьому підрозділі розглянемо шляхи розв'язання одного з найпопулярніших і затребуваних завдань з розпізнавання облич у потоці. Процес розпізнавання облич зазвичай складається з двох етапів: пошук області обличчя на зображенні і порівняння знайденої особи з особами із бази даних. Друга частина цього завдання виходить за межі нашого розгляду. Тому зупинимося більш детально на розгляді першої частини − виявленні й визначенні характеристик облич.

Зазвичай для цього використовують алгоритм Віоли – Джонса, який є найбільш популярним методом для пошуку області обличчя на зображенні завдяки його високій швидкості й ефективності. Незважаючи на те, що цей метод запропоновано порівняно недавно (2001 року), його можна з цілковитою підставою вважати класичним. Перш ніж перейти до практичної реалізації, нагадаємо коротко суть цього методу.

Основна ідея алгоритму спочатку полягала в пошуку структур, притаманних усім особам. Наприклад, на кожному людському об-

time. This allows you to effectively use the results for solving a wide range of applied tasks in such areas as digital image processing, computer vision, biometrics, creation of intelligent security systems, surveillance and control of access to objects, etc. In a broad sense, these are pattern recognition tasks.

In this section, we will look at ways to solve one of the most popular and recognizing faces sought-after tasks for in a video stream. The process of face recognition usually consists of two stages: the search for the face area in the image, and the comparison of the found person with the persons in the database. The second part of this task is beyond our scope. Therefore, we dwell in more detail on the consideration of the first part - the detection and characterization of individuals.

Usually, the Viola – Jones algorithm is used for this, which is the popular method for finding the face area in an image because of its high speed and efficiency. Despite the fact that it was proposed relatively recently (in 2001), it can rightly be considered a classic. We recall briefly the essence of this method.

The basic idea of the algorithm was originally to find structures that are common to all individuals. For example, on every human face, the area of the eye is always darker

личчі область очей завжди темніше області щоки, а область носового моста темніше області очей. Використання таких характеристик людського обличчя дає змогу створити загальну модель, а потім застосувати її для виявлення облич на зображеннях.

П. Віола і М. Джонс у своєму алго-ритмі розпізнавання облич уперше запропонували спосіб, який використовує вейвлети Хаара для визначення ознак облич на зображенні. Вони запропонували метод, який використовує каскади примітивів Хаара, що являють собою розбивку заданої прямокутної області на набори різнотипних прямокутних підобластей. Приклад примітивів Хаара для виявлення осіб показано на рис. 7.9.

Розглянемо основні принципи, на яких базується метод пошуку облич на зображенні:

- використовується інтегральна форма зображення, що дає змогу швидко виявляти необхідні об'єкти;
- застосовуються примітиви Хаара, за допомогою яких відбувається пошук потрібного об'єкта (у цьому випадку, обличчя та його основних елементів);
- використовується бустинг (посилення) при виборі найбільш підходящих примітивів для шуканого об'єкта в певній частині зображення;
- усі ознаки надходять на вхід класифікатора, який приймає рішення про наявність чи відсутність об'єкта пошуку − так/ні;
- застосовуються каскади ознак для швидкого відкидання вікон,

than the cheek, and the area of the nasal bridge is darker than the eye. The use of such characteristics of a human face allows you to create a common model, and then apply it to detect faces in images.

Viola and Jones in their face recognition algorithm for the first time presented a method that uses Haar wavelets to determine the signs of faces in an image. They proposed a method using cascades of Haar primitives, which are a breakdown of a given rectangular area into sets of different types of rectangular subregions. An example of Haar primitives for face detection is shown in Fig. 7.9.

Consider the basic principles on which the method for searching faces in an image is based:

- uses an integral form of the image, which allows you to quickly detect the necessary objects;
- Haar primitives are used, with the help of which the required object is searched (in this case, the face and its main elements);
- using boosting (gain) when choosing the most suitable primitives for the desired object in this part of the image;

  all signs come to the input of the clas-sifier, which makes a decision about the presence or absence of the search object − yes/no;
- feature cascades are used to quickly drop windows where the

де не знайдено шуканої особи.

Пошук облич здійснюється за принципом сканувального вікна:

- зображення, на якому є шукані об'єкти, подано двовимірною матрицею пікселів розміром $\boldsymbol{w} \times \boldsymbol{h}$, у якій кожен піксель має значення від 0 до 255 для півтонових зображень і від 0 до 255^3, якщо це кольорове зображення (компоненти ***R, G, B***);
- алгоритм повинен виявити й визначити обличчя та їх риси і позначити їх − пошук здійснюється в активній області зображення прямокутними ознаками, за допомогою яких й описується знайдене обличчя та його риси:

```
rectangle = {x,y,w,h,a},
```

де `x, y` − координати центра `i`-го прямокутника, `w`, `h` – його ширина й висота відповідно, `a` − кут нахилу прямокутника до вертикальної осі зображення.

Щодо зображень використовується метод пошуку осіб на основі сканувального вікна (`scanning window`) − сканується зображення вікном пошуку, а потім застосовується класифікатор до кожного положення.

Система навчання й вибору найбільш значущих ознак є повністю автоматизованою і не потребує втручання людини. Такий підхід дає змогу працювати швидко.

Для зменшення обсягів обчислень використовуються інтегральні зображення, також відомі як таблиці підсумовуваних областей. Вони є ще однією формою зберігання зображень.

desired face has not been found.

The search for faces is carried out on the principle of a scanning window:

- the image with the desired objects is represented by a two-dimensional matrix of pixels of size $\boldsymbol{w} \times \boldsymbol{h}$, in which each pixel has a value from 0 to 255 for half-tone images and from 0 to 255^3 if it is a color image (components ***R, G, B***);
- the algorithm should detect and identify faces and their features and mark them - the search is performed in the active area of the image with rectangular signs.

where `x, y` are the coordinates of the center of the `i-` th rectangle, `w` is the width, `h` is the height, a is the angle of inclination of the rectangle to the vertical axis of the image.

With respect to images, the method of searching faces based on the scanning window (scanning window) is used - the image is scanned by the search window, and then the classifier is applied to each position.

The system of training and selection of the most significant features is fully automated and does not require human intervention. This approach allows you to work quickly.

To reduce the amount of computation, integral images are used, also known as tables of summed areas. They are another form of image storage.

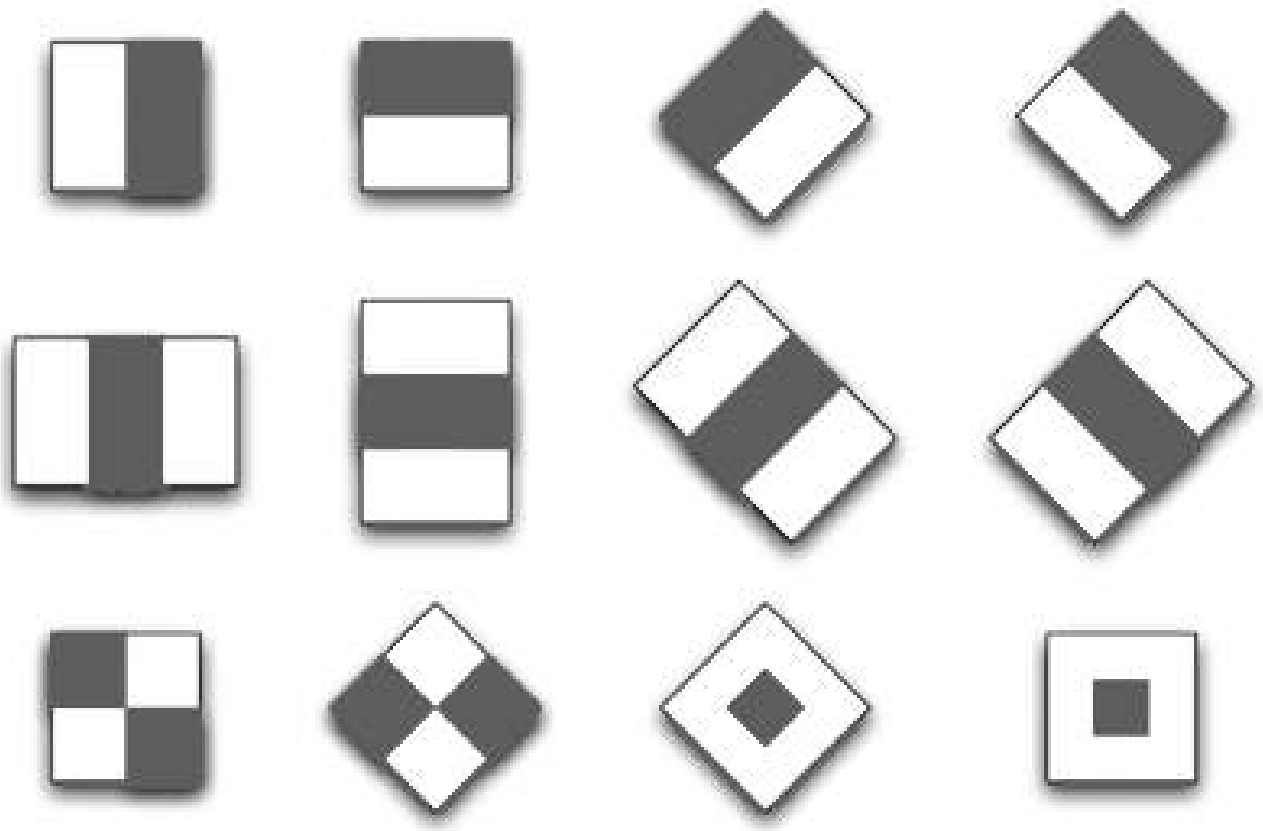


Рис. 7.9. Примітиви Хаара для виявлення облич

Fig. 7.9. Haar primitives for face detection

Для інтегральних зображень кожен піксель на зображенні замінюється сумою всіх елементів злів а й вище цього пікселя. Для розрахунку інтегральних зображень використовується така формула:

For integral images, each pixel in the image is replaced by the sum of all elements on the left and above this pixel. The following formula is used to calculate the integral images:

$$\boldsymbol{I_{\Sigma}(x,y)} = \sum_{\substack{x' \le x \\ y' \le y}} \boldsymbol{i(x',y')},$$

де $\boldsymbol{i(x',y')}$ − значення пікселя в $\boldsymbol{(x',y')}$ у зображенні; $\boldsymbol{I}$ − інтегральний образ зображення.

where $\boldsymbol{i(x',y')}$ means the pixel value in $\boldsymbol{(x',y')}$ in the image. $\boldsymbol{I}$ is the integral image of the image.

Щоб уникнути надмірності зазвичай використовують формулу, за якою ефективно обчислюються інтегральні зображення:

To avoid redundancy, a formula is usually used that effectively calculates integral images:

$$\boldsymbol{I(x,y) = i(x,y) - I(x-1,y-1) + I(x,y-1) + I(x-1,y).}$$

Тут $\boldsymbol{i}$ − вихідне зображення, а $\boldsymbol{I}$ − інтегральне зображення.

Here $\boldsymbol{i}$ is the original image, and $\boldsymbol{I}$ is the integral image.

Тепер розглянемо практичну

Now let's look at the practical

реалізацію алгоритму пошуку облич на зображеннях із застосуванням ресурсів бібліотеки ***OpenCV***.

Перш ніж формувати програмний код, потрібно завантажити попередньо підготовлені XML-файли каскадів Хаара з бібліотеки ***OpenCV***. У нашому випадку використовуємо `haarcascade_frontalface_default.xml` і `haarcascade_eye.xml` – файли для виявлення обличчя та очей. Завантаження класифікаторів каскадів Хаара проводиться з використанням функції `cv2.CascadeClassifier()`.

Структура коду, яку наведено нижче, призначена для того, щоб спочатку визначати обличчя людей, а потім у межах області особи виявляти очі. Після завантаження файлів XML зчитується зображення, на якому необхідно визначити обличчя. Для зменшення обсягу обчислень виявлення очей виконується тільки в області обличчя, що є областю інтересу (***ROI***).

Структура коду:

implementation of the face search algorithm on images using the resources of the ***OpenCV*** library.

Before generating the program code, you need to load preprepared XML files of the Haar cascades from the ***OpenCV*** library. In our case, we use `haarcascade_frontalface_default.xml` and `haarcascade_eye.xml` – files for face and eye detection. Haar cascade classifiers are loaded using the `cv2.CascadeClassifier()` function.

The structure of the code, which is shown below, is intended to first detect faces and then detect eyes within the face area. After loading the XML files, an image is read in which it is necessary to perform face detection. To reduce the amount of computation, eye detection is performed only on the face area, which is the region of interest (***ROI***).

Code structure:

```
import numpy as np
import cv2

# Define cascades
face_cascade =
cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
eye_cascade = cv2.CascadeClassifier('haarcascade_eye.xml')

img = cv2.imread("alen02.jpg")
cv2.imshow('Input',img)

gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
faces = face_cascade.detectMultiScale(gray, 1.3, 5)
for (x, y, w, h) in faces:
    # Draw rect around found face
    cv2.rectangle(img, (x,y), (x+w,y+h), (255,0,0), 2)
```

```
    roi_gray = gray[y:y+h, x:x+w]
    roi_color = img[y:y+h, x:x+w]

    eyes = eye_cascade.detectMultiScale(roi_gray)
    for(ex, ey, ew, eh) in eyes:
        # Draw rect around found eye at current face
         cv2.rectangle(roi_color, (ex,ey), (ex+ew,ey+eh),
(0,255,0), 2)

cv2.imshow('Output',img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Результати виконання цього коду показано на прикладі виявлення обличчя та очей на фотографіях двох відомих публічних персонажів (рис. 7.10).

The results of the execution of this code are shown on the example of the detection of the face and eyes in the photographs of two well-known public characters (Fig. 7.10).

а

б

в

г

Рис. 7.10. Результати виявлення обличчя та очей

Fig. 7.10. Face and eye detection results

Такий детектор має малу ймовірність помилкового виявлення особи. Алгоритм добре працює й розпізнає риси обличчя при нахилах голови під невеликим кутом, приблизно до 30 градусів. При куті нахилу більше 30 градусів відсоток виявлень різко знижується.

І це не дає змоги детектувати обличчя людини, повернене під довільним кутом, що значною мірою ускладнює або робить неможливим використання алгоритму в деяких ситуаціях. Слід також відзначити високу чутливість алгоритму до умов освітлення осіб. При недостатньому рівні освітлення ймовірність правильного виявлення істотно зменшується. Ці проблеми добре видно на груповому знімку (рис. 7.11).

Such a detector has a low probability of false detection of the face. The algorithm works well and recognizes the features of the face when the head is tilted at a slight angle, up to about 30 degrees. When the angle of inclination is more than 30 degrees, the percentage of detections drops sharply.

And it does not allow to detect the turned person's face at an arbitrary angle, which greatly complicates or makes it impossible to use the algorithm in some situations. It should also be noted the high sensitivity of the algorithm to the conditions of lighting persons. With an insufficient level of illumination, the probability of correct detection is significantly reduced. These problems are clearly visible in the group picture (Fig. 7.11).

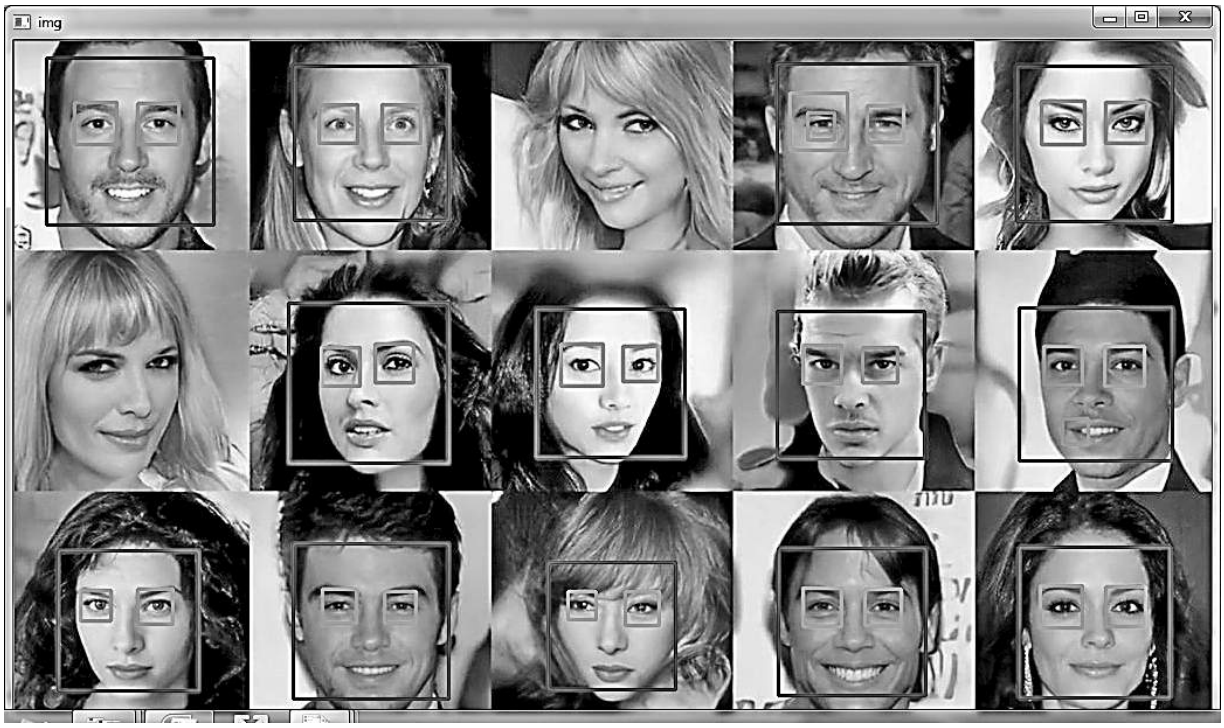


Рис. 7.11. Виявлення облич та очей на груповій фотографії

Fig. 7.11. Face and eye detection on a group photo

Наведемо тепер приклад пошуку осіб у потоці відеоданих, отри-

We now give an example of finding people in the stream of video

маних з web-камери. Зазначимо, що порівняно з алгоритмом розпізнавання осіб на звичайних зображеннях тут не виникає принципових відмінностей. Нижче наведено код цієї програми, а результати роботи показано на рис. 7.12:

data received from a web-camera. Note that in comparison with the face recognition algorithm on ordinary images there are no fundamental differences. Below is the code of this program, and the results are shown in Fig. 7.12:

```
import numpy as np
import cv2

cap = cv2.VideoCapture(0)
# Define cascades
face_cascade =
cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
eye_cascade = cv2.CascadeClassifier('haarcascade_eye.xml')

while(True):
    # Capture frame-by-frame
    ret, frame = cap.read()

    # Convert to gray to be able to recognize
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # detect faces
    faces = face_cascade.detectMultiScale(gray, 1.3, 5)

    for (x, y, w, h) in faces:
        # Draw rect around found face
        cv2.rectangle(frame,(x,y),(x+w,y+h),(255,0,0),2)
        roi_gray = gray[y:y+h, x:x+w]
        roi_color = frame[y:y+h, x:x+w]

        eyes = eye_cascade.detectMultiScale(roi_gray, 1.3, 5)
        for(ex, ey, ew, eh) in eyes:
            # Draw rect around found eye at current face
             cv2.rectangle(roi_color, (ex,ey), (ex+ew,ey+eh),
(0,255,0), 2)

    # Display the resulting frame
    cv2.imshow('Output frame', frame)

    # Bind keys for exit
    c = cv2.waitKey(1)
    if c == 27:
        break

# When everything done, release the capture
cap.release()
cv2.destroyAllWindows()
```

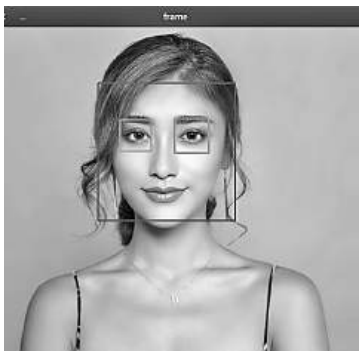


Рис. 7.12. Приклад виявлення обличчя та очей при відеореєстрації

Fig. 7.12. Example of face and eye detection during video recording

Необхідно зазначити, що найбільш трудомістким є процес навчання каскадів Хаара з допомогою алгоритму машинного навчання AdaBoost. Однак, сьогодні для каскадних класифікаторів Хаара існує велика кількість уже навчених каскадів, у тому числі в стандартному постачанні бібліотеки ***OpenCV***. Її установчий пакет містить цілий набір готових навчених класифікаторів, збережених у вигляді файлів з розширенням «*.xml». У цьому наборі є класифікатори для пошуку як обличчя, так і його окремих частин (очей, рота, носа).

При синтезі алгоритмів розпізнавання найбільш продуктивним є використання стандартних (заздалегідь навченних) класифікаторів. У зв'язку з цим уважаємо за доречне навести для довідки список основних попередньо навчених класифікаторів:

It should be noted that the most time-consuming process is learning the Haar cascades using the AdaBoost machine learning algorithm. However, currently there are a large number of cascades already trained for cascade classifiers of Haar, including the standard package of the ***OpenCV*** library. Its installation package contains a whole set of ready-made trained classifiers, saved as files with the extension "* .xml". In this set there are classifiers for searching for a face, as well as for its individual parts (eyes, mouth, nose).

In the synthesis of recognition algorithms, the use of standard (pre-trained) classifiers is most productive. In this regard, we consider it appropriate to provide for reference a list of the main preliminary trained classifiers:

- haarcascade_eye_tree_eyeglasses.xml;
- haarcascade_fullbody.xml;
- haarcascade_mcs_lefteye.xml;
- haarcascade_profileface.xml;
- haarcascade_eye.xml;
- haarcascade_lefteye_2splits.xml;
- haarcascade_mcs_mouth.xml;

- haarcascade_righteye_2splits.xml;
- haarcascade_frontalface_alt2.xml;
- haarcascade_frontalface_alt_tree.xml;
- haarcascade_mcs_eyepair_big.xml;
- haarcascade_mcs_rightear.xml;
- haarcascade_upperbody.xml;
- haarcascade_frontalface_alt.xml;
- haarcascade_mcs_eyepair_small.xml;
- haarcascade_mcs_righteye.xml;
- haarcascade_frontalface_default.xml;
- haarcascade_mcs_leftear.xml;

haarcascade_mcs_upperbody.xml.

Не зупиняючись детально на призначенні й властивостях окремих класифікаторів, зазначимо, що особливий інтерес серед них становлять ті, які дають змогу визначити положення очей. Це дуже корисна процедура, оскільки вона дає можливість оцінити відстані між зіницями. При подальшому аналізі це дає змогу усувати фактори, пов'язані з нахилом голови. Класифікатори, навчені пошуку очей, є дуже чутливими до наявності окулярів. У більшості випадків вони дають збій, особливо для окулярів, що повністю приховують очі. У таких випадках слід використовувати більш ефективний класифікатор для виявлення очей haarcascade_eye_tree_eyeglasses.xml, навчений пошуку на зображенні очей в окулярах. Його можна використовувати як запасний варіант у випадках збою в роботі звичайного класифікатора.

Найбільш ефективними є методи пошуку основних елементів у вже виділеній області особи, оскільки це локалізує область пошуку

Without dwelling in detail on the purpose and properties of individual classifiers, we note that of particular interest among them are those that allow us to determine the position of the eyes. This is a very useful procedure, since it provides an opportunity to estimate the distance between the pupils. This further analysis allows you to eliminate the factors associated with the inclination of the head. Qualifiers trained to search for eyes are very sensitive to the presence of glasses. In most cases, they fail, especially for glasses that completely hide their eyes. In such cases, you should use a more efficient classifier for eye detection haarcascade_eye_tree_eyeglasses.xml, trained in searching for images of eyes with glasses. It can be used as a backup option in cases of failure in the normal classifier.

The most effective are the methods of searching for the main elements in an already selected area of the face, since this localizes the region of search and shortens the

й скорочує час аналізу, а також значно знижує ймовірність помилкових спрацьовувань. Вибір областей інтересу для виявлення очей, носа і рота потребує оптимізації положення й розмірів області інтересу для кожного елемента. Приклад такого розбиття виділеної області на зони пошуку окремих елементів обличчя в нашому алгоритмі показано на рис. 7.13.

Розміри областей пошуку нашого алгоритму визначалися експериментально, але, якщо необхідно, можуть бути оптимізовані додатково. Зверніть увагу на те, що існує дві можливості виявлення очей − виділення загальної для двох очей області інтересу (або області пошуку) і пошук і виявлення лівого й правого очей окремо.

У ***OpenCV*** для цього передбачено відповідні класифікатори. Який з цих методів краще, доцільно визначати експериментальним шляхом.

analysis time, and also significantly reduces the probability of false positives. The selection of areas of interest for detecting eyes, nose and mouth requires optimizing the position and size of the area of interest for each element. An example of such a partition of the selected area into the search zones for individual face elements in our algorithm is shown in Fig. 7.13.

The size of the search areas of our algorithm was determined experimentally, but if necessary, can be further optimized. Please note that there are two possibilities for detecting eyes − highlighting a common area of interest for two eyes (or a search area) and searching and detecting the left and right eyes separately.

In ***OpenCV***, appropriate classifiers are provided for this. Which of these methods is preferable, it is advisable to determine experimentally.

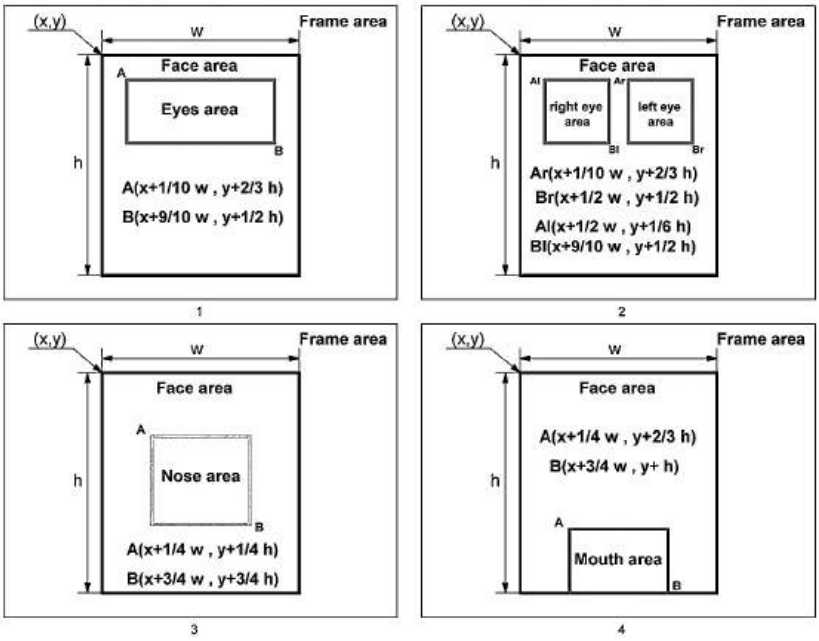


Рис. 7.13. Зони пошуку облич та їх елементів у кадрі (координати окремих зон пошуку)

Fig. 7.13. Search zones for faces and their elements in the frame (coordinates of individual search zones)

Дуже важливим питанням при вирішенні завдання виявлення й детектування облич у кадрі є формування об'єктивних показників якості роботи алгоритмів і методів їх використання. Об'єктивним критерієм ефективності роботи будь-якого алгоритму детектування облич та їх елементів у кадрах відеопослідовності є показник імовірності правильного виявлення обличчя за умови його наявності в кадрі. Процедура виявлення особи в поточному кадрі із застосуванням відповідного каскадного класифікатора в разі успіху завершується побудовою прямокутника з допомогою функції ***OpenCV***:

A very important issue in solving the problem of detecting and detecting faces in a frame is the question of the formation of objective indicators of the quality of the algorithms and methods for their use. An objective criterion for the effectiveness of any algorithm for detecting faces and their elements in frames of a video sequence is an indicator of the probability of correct detection of a person, provided it is present in the frame. The procedure of detecting a person in the current frame using the corresponding cascade classifier, if successful, is completed by constructing a rectangle using the ***OpenCV*** function:

```
cv2.rectangle(frame,(x,y)(x+w,y+h),255,0,0),2).
```

Успішне завершення детектування обличчя в програмному коді алгоритму супроводжується появою логічної одиниці, в іншому випадку − появою логічного нуля.

Для створення стійкого критерію поточної якості правильного виявлення осіб можна розрахувати ймовірність Pn, яка може бути отримана за результатами підрахунку в ковзному вікні розміром 100 кадрів кількості успішних виявлень і є нормованою на розмір вікна.

При стандартній швидкості змінення кадрів відеокамери 30/с вікно має постійну часу змінення Pn, що приблизно дорівнює 3,3 с. Якість роботи детектора можна вважати задовільним при Pn 0,9. За показником імовірності правильного виявлення облич у

The successful completion of face detection in the program code of the algorithm must be accompanied by the occurrence of a logical one event. Otherwise - the appearance of a logical zero.

To create a stable criterion for the current quality of the correct detection of faces, one can calculate the probability Pn. It can be obtained from the counting results in a sliding window with a size of 100 frames of the number of successful detections and normalized to the size of the window.

With a standard frame rate of 30/s, the window has a time constant of Pn of approximately 3.3 seconds. The quality of the detector can be considered satisfactory for Pn 0.9. In terms of the probability of correct detection of faces in a frame, it is easy to analyze the in-

кадрі легко аналізувати вплив різних факторів (у тому числі зміненння освітленості, геометричних факторів та ін.) на якість роботи алгоритмів виявлення і детектування облич. Аналогічним чином розраховуються й імовірності правильного виявлення очей, носа, рота у відповідних областях.

Ці показники обчислюються як умовна ймовірність такої події за умови правильного виявлення всього обличчя. З імовірністю, прийнятною для практичного використання, події виявлення окремих елементів обличчя можна вважати незалежними, а отже, імовірність правильного виявлення всіх елементів оцінювати як добуток імовірностей виявлення цих елементів. Це важливе питання розглянемо більш детально під час практичної реалізації алгоритму виявлення й детектування облич.

Робоча версія алгоритму виявлення облич створювалася з урахуванням описаних вище підходів і доступних ресурсів.

Було поставлено завдання виявлення осіб, очей, носа і рота.

Для економії ресурсів і простоти реалізації алгоритмів у цьому проекті використовувався набір попередньо навчених каскадних класифікаторів Хаара для відповідних елементів особи, імпортований з бібліотеки ***OpenCV***.

Доречно нагадати про необхідність розмістити ці класифікатори в кореневій папці ***Python***. Це усуває помилки при пошуку шляхів звернення до них і прискорює об-

fluence of various factors (including changes in light, geometrical factors, etc.) on the quality of the work of the algorithms for detecting and detecting faces. Similarly, the probabilities of correct detection of eyes, nose, and mouth in corresponding areas are calculated.

These indicators are calculated as the conditional probability of such an event, provided that the entire face is correctly detected. With reliability acceptable for practical use, we can consider the events of detection of individual elements of an individual independent, and therefore, the probability of correct detection of all elements can be estimated as the product of the probabilities of detection of these elements. We will consider this important issue in more detail in the practical implementation of the algorithm for detecting and detecting faces.

The working version of the face detection algorithm was created taking into account the approaches described above and the resources available.

The task was to detect the faces, eyes, nose and mouth.

To save resources and ease the implementation of algorithms in this project, a set of pre-trained cascade Haar classifiers for the relevant elements of the person imported from the ***OpenCV*** library was used.

It is appropriate to recall the need to place these classifiers in the ***Python*** root folder. This eliminates errors when searching for ways to access them and speeds

роблення вихідних даних. Перелік цих класифікаторів є таким:

up the processing of source data. The list of these classifiers is as follows:

- `face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default. xml')`
- `eye_cascade = cv2.CascadeClassifier('haarcascade_eye.xml')`
- `right_eye_cascade = cv2.CascadeClassifier('haarcascade_righteye_2splits. xml')`
- `left_eye_cascade = cv2.CascadeClassifier('haarcascade_lefteye_2splits.xml')`
- `nose_cascade = cv2.CascadeClassifier('haarcascade_nose.xml')mouth_cascade= cv2.CascadeClassifier('haarcascade_mouth. xml')`

Далі наведемо повний код програми цього алгоритму, а результати його роботи покажемо на рис. 7.14, 7.15.

Next we give the full code of the program of this algorithm, and the results of its work will be shown in Fig. 7.14, 7.15.

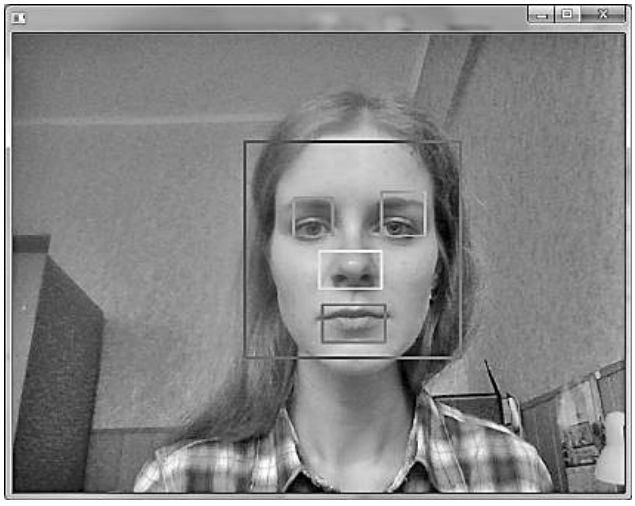
```
import os.path
import cv2
from pylab import show
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
import matplotlib.gridspec as gridspec
from matplotlib.ticker import MaxNLocator
import timeit

# Define cascades
face_cascade =
cv2.CascadeClassifier('haarcascade_frontalface_default. xml')
right_eye_cascade =
cv2.CascadeClassifier('haarcascade_righteye_2splits.xml')
left_eye_cascade =
cv2.CascadeClassifier('haarcascade_lefteye_2splits.xml')
nose_cascade = cv2.CascadeClassifier('haarcascade_nose.xml')
mouth_cascade = cv2.CascadeClassifier('haarcascade_mouth.xml')

# Define video if exists
video_file = "Video_1.avi"

# Define colors
blue = (255, 0, 0)
green = (0, 255, 0)
red = (0, 0, 255)
yellow = (0, 255, 255)

# Actual detecting
```

```
def detect(current_frame):
    face_detected = 0
    eyes_detected = 0
    nose_detected = 0
    mouth_detected = 0
    # Convert into white/black
    gray = cv2.cvtColor(current_frame, cv2.COLOR_BGR2GRAY)
    # Detect faces
    faces = face_cascade.detectMultiScale(gray,1.3,5)
    for (x,y,w,h) in faces:
        face_detected = 1
        # Draw rect around found face
        cv2.rectangle(current_frame,(x,y),(x+w,y+h),blue,2)

        # Detect right eye
        right_eye_detected = detect_face_part (cur-
rent_frame,gray, right_eye_cascade, green,
                                               y+2*h/9,y+2*h/3,
                                               x+w/9,x+4*w/9)
        # Detect left eye
        left_eye_detected=detect_face_part(current_frame,
gray, left_eye_cascade, green,
                                               y+2*h/9,y+2*h/3,
                                               x+5*w/9,x+8*w/9)
        eyes_detected = 1 if right_eye_detected and
left_eye_detected else 0
        # Detect nose
        nose_detected = detect_face_part(current_frame, gray,
nose_cascade, yellow,
                                              y+h/4,y+3* h/4,
                                              x+w/6,x+4*w/6)
        # Detect mouth
        mouth_detected = detect_face_part(current_frame, gray,
mouth_cascade, red,
                                               y+2*h/3,y+h,
                                               x+w/6,x+5*w/6)
    # Draw results
    return face_detected, eyes_detected, nose_detected,
mouth_detected, current_frame

# Detecting part of face with according cascade and zone of
interest
def detect_face_part(current_frame, gray, cascade, color,
y_from, y_to, x_from, x_to):
    # Select gray zone
    roi_gray_zone = gray[int(y_from):int(y_to),
int(x_from):int(x_to)]
    # Select colored zone
    roi_color_zone = current_frame[int(y_from):int(y_to),
```

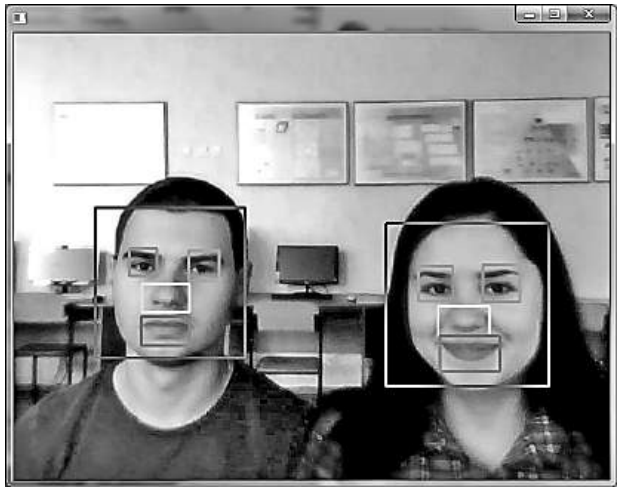
```
int(x_from):int(x_to)]

    # Detect part of face using appropriate cascade and zone
    parts = cascade.detectMultiScale(roi_gray_zone)
    for (x,y,w,h) in parts:
        # Assuming we have only one part of a face, after de-
tecting draw area and exit loop
    cv2.rectangle(roi_color_zone,(x,y),(x+w,y+h),color, 2)
        return 1
    return 0

# Define video capturing
cap = cv2.VideoCapture(video_file) if
os.path.isfile(video_file) else cv2.VideoCapture(0)

counter = 0
face_detection_map = []
eyes_detection_map = []
nose_detection_map = []
mouth_detection_map = []

start = timeit.default_timer()

while (counter < 1000):
    counter += 1

    ret, frame = cap.read()

    face_is_detected, \
    eyes_is_detected, \
    nose_is_detected, \
    mouth_is_detected, \
    updated_frame = detect(frame)

    face_detection_map.append(face_is_detected)
    eyes_detection_map.append(eyes_is_detected)
    nose_detection_map.append(nose_is_detected)
    mouth_detection_map.append(mouth_is_detected)

    cv2.imshow('', updated_frame)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

stop = timeit.default_timer()

print("Time:")
print(stop - start)

# Cleanup
cap.release()
cv2.destroyAllWindows()
```

```
fig = plt.figure(num=None, figsize=(10, 6), dpi=80,
facecolor='w', edgecolor='k')
gs = gridspec.GridSpec(4, 1)

ax_face = plt.subplot(gs[0, 0])
ax_face.set_title('Face detection')
ax_eyes = plt.subplot(gs[1, 0])
ax_eyes.set_title('Eyes detection')
ax_nose = plt.subplot(gs[2, 0])
ax_nose.set_title('Nose detection')
ax_mouth = plt.subplot(gs[3, 0])
ax_mouth.set_title('Mouth detection')

ax_face.set_xlabel("N")
ax_face.set_xlim(0, 1000)
ax_face.set_ylim(0, 1.1)
ax_face.yaxis.set_major_locator(MaxNLocator(integer=True))
ax_face.xaxis.set_label_coords(1.01, 0.1)
ax_face.legend(
    handles=[
        mpatches.Patch(color='blue', label="P:
{0}%".format(sum(face_detection_map)/10.0))
    ])

ax_eyes.set_xlabel("N")
ax_eyes.set_xlim(0, 1000)
ax_eyes.set_ylim(0, 1.1)
ax_eyes.yaxis.set_major_locator(MaxNLocator(integer=True))
ax_eyes.xaxis.set_label_coords(1.01, 0.1)
ax_eyes.legend(
    handles=[
        mpatches.Patch(color='blue', label="P:
{0}%".format(sum(eyes_detection_map)/10.0))    ])

ax_nose.set_xlabel("N")
ax_nose.set_xlim(0, 1000)
ax_nose.set_ylim(0, 1.1)
ax_nose.yaxis.set_major_locator(MaxNLocator(integer=True))
ax_nose.xaxis.set_label_coords(1.01, 0.1)
ax_nose.legend(
    handles=[
        mpatches.Patch(color='blue', label="P:
{0}%".format(sum(nose_detection_map)/10.0))
    ])

ax_mouth.set_xlabel("N")
ax_mouth.set_xlim(0, 1000)
ax_mouth.set_ylim(0, 1.1)
ax_mouth.yaxis.set_major_locator(MaxNLocator(integer=True))
ax_mouth.xaxis.set_label_coords(1.01, 0.1)
```

```
ax_mouth.legend(
    handles=[
        mpatches.Patch(color='blue', label="P:
{0}%".format(sum(mouth_detection_map)/10.0))
    ])

x = range(1000)

ax_face.step(x, face_detection_map)
ax_eyes.step(x, eyes_detection_map)
ax_nose.step(x, nose_detection_map)
ax_mouth.step(x, mouth_detection_map)
fig.tight_layout()

show()
```



а



б

Рис. 7.14. Результати детектування облич

Fig. 7.14. Face Detection Results

Для узагальненого оцінювання ефективності роботи алгоритмів застосовуються два методи – якісний (візуальний) і кількісний. У першому випадку проводиться візуальне спостереження за різнокольоровими прямокутниками, що обмежують виявлене обличчя та його елементи (очі, ніс і рот).

Якщо процес їх відтворення сприймається як безперервний, то якість детектування можна вважати прийнятною (близько 100 %).

Однак інерційність зору спостерегача при високій частоті зміни кадрів (30/с) не дає змоги

For a generalized assessment of the efficiency of the algorithms, two methods are possible - qualitative (visual) and quantitative. In the first case, a visual observation is carried out for the multi-colored rectangles that limit the detected face and its elements (eyes, nose and mouth.

If the process of their rendering is perceived as continuous, the quality of detection can be considered acceptable (equal to ~ 100%).

However, the inertia of the observer at a high frame rate (30/s)

візуально сприймати пропуски виявлення окремих кадрів.

Це може істотно спотворити результати тестування. Тому для кількісного оцінювання ефективності в програмі для всіх кадрів відеопослідовності формується одновимірний масив (1 − за фактом виявлення обличчя в кадрі, 0 − у разі пропуску обличчя), за яким будуються ймовірнісні характеристики якості роботи алгоритму. Однак слід мати на увазі, що для цього необхідна достовірно анотована тестова відеопослідовність.

Простий варіант такої послідовності – відеоряд, у якому в кожному кадрі є особа для виявлення, дотримуються умови освітлення (фронтальне джерело світла) і підтримується геометричний фактор (нахил голови менше 30^0). Зрозуміло, що можна використовувати і складніший тестовий відеоряд, у якому в певні інтервали часу осіб в кадрі немає. Тоді можна оцінити не лише ймовірність правильного виявлення облич, але й імовірність помилкового виявлення.

На рис. 7.15 показано спеціальну форму узагальненого оцінювання ефективності роботи алгоритму виявлення облич та їх головних елементів для відеоряду тривалістю 1000 кадрів. Тривалість такого запису становить ~ 33 с. Для процедур виявлення всіх елементів побудовано періо-до-грами, де для кожного з N кад-

does not allow to visually perceive the gaps in the detection of individual frames.

This can significantly distort the test results. Therefore, to quantify the effectiveness of the program, for all frames of a video sequence, a one-dimensional array is formed (1 – upon detection of a face in a frame, and 0 – in the case of a missing face), which are used to construct probabilistic characteristics of the quality of the algorithm. However, it should be borne in mind that this requires a reliably annotated test video sequence.

The simplest version of such a sequence is a video sequence, which has a face for detection in each frame, the lighting conditions (frontal light source) and the geometrical factor (head inclination less than 30^0) are maintained. Of course, it is possible to use a more complex test video sequence, which does not have faces in a frame at certain intervals of time. Then there is a possibility to estimate not only the probabilities of the correct detection of faces, but also the probabilities of false detection (detection of a face under the condition of its absence in the frame).

In Fig. 7.15 shows a special form of a generalized assessment of the perfor-mance of the algorithm for detecting individuals and their main elements for a video sequence with a duration of 1000 frames. The duration of such a record is ~ 33 s. For the procedures for detecting all elements, periodograms are constructed, where for each of the N

рів у відповідність ставиться 1, якщо обличчя виявлено, і 0, якщо не виявлено.

Крім того, для кожної періодограми обчислюється ймовірність правильного виявлення на вибірці з 1000 кадрів.

Така форма не тільки дає змогу отримати узагальнені оцінки ймовірностей виявлення, а й наочно показує, на яких кадрах виявлення не було виконано.

frames, 1 (if successful) and 0 (if unsuccessful) of the corresponding element of the face are assigned.

In addition, for each periodogram, the probability of correct detection is calculated on a sample of 1000 frames.

This form allows to obtain not only generalized estimates of detection probabilities, but also vividly shows on which frames the detection was not performed.

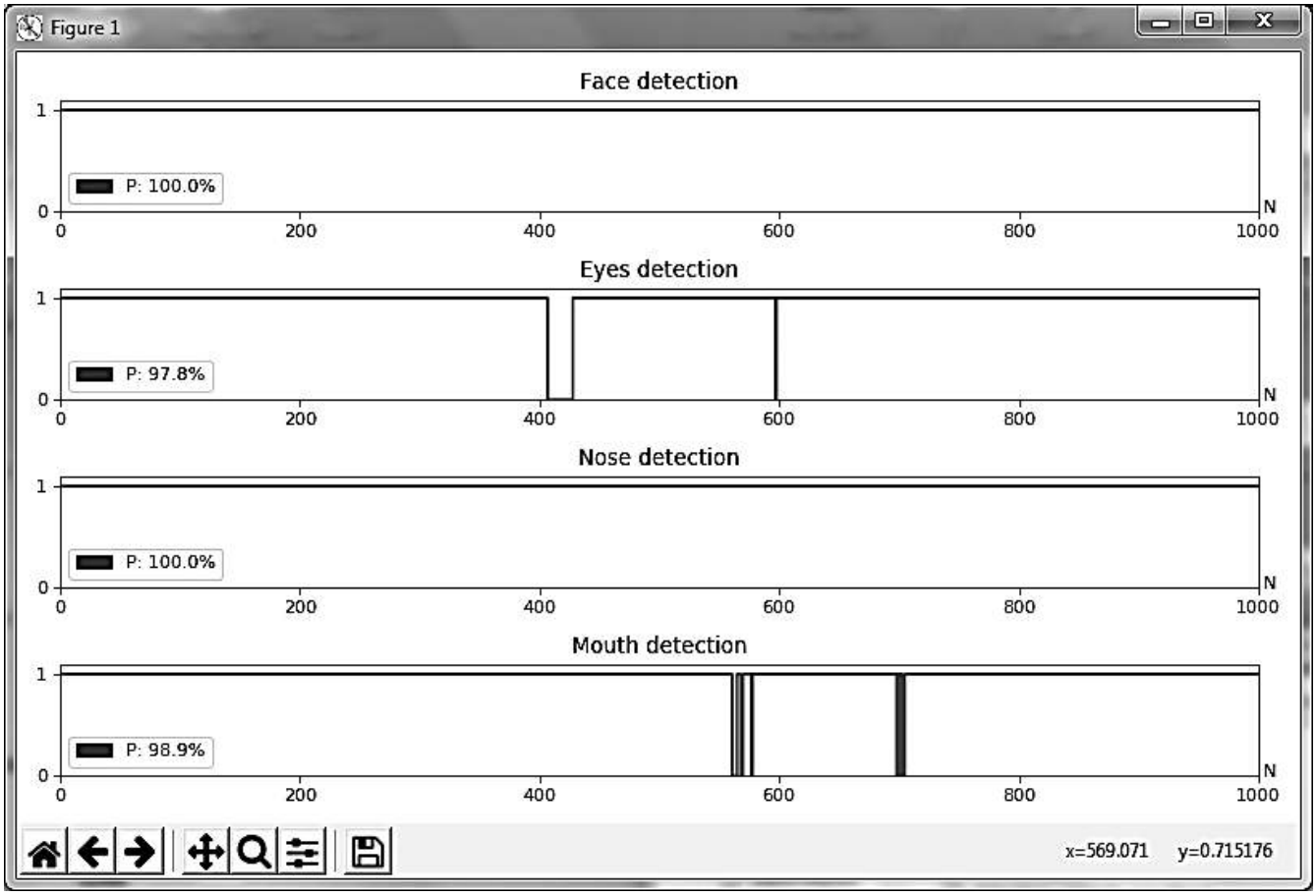


Рис. 7.15. Вікно оцінювання ефективності виявлення обличчя та його головних елементів за відеоданими

Fig. 7.15. Screen for evaluating the efficiency of detecting a face and its main elements by video data

Наведені приклади не вичерпують усіх можливостей детектування облич. Розроблено й широко використовуються моделі Хаара для виявлення. Але цей матеріал виходить за межі нашого розгляду.

The examples given do not exhaust all possibilities of detecting faces. Haar models for detection are developed and widely used. But this material goes beyond our consideration.

Каскади Хаара можуть бути використані для виявлення й інших типів об'єктів. Для цього необхідно навчати класифікатор, надаючи позитивні і негативні зображення для об'єкта, який потрібно виявити. На навчання каскадів Хаара витрачається багато часу, але ці втрати компенсуються при використанні навченого класифікатора завдяки його швидкодії.

Haar cascades can be used to detect other types of objects. To do this, it is necessary to train the classifier, providing positive and negative images for the object to be detected. Learning the Haar cascades takes a lot of time, but these losses are compensated for by using a trained classifier - its speed is always quite large.

БІБЛІОГРАФІЧНИЙ СПИСОК
BIBLIOGRAPHIC LIST

Библиотека компьютерного зрения OpenCV [Электронный ресурс]. – Режим доступа: http://docs.opencv.org/trunk/doc/py_tututoria/py_objdetect/py_face_detection/py_face_detection.html.

Гридин, В. Н. Адаптивные системы технического зрения / В. Н. Гридин, В. С. Титов, М. Н. Труфанов. – СПб. : Наука, 2009. – 442 с.

Лутц, М. Программирование на Python. Т. 1 / М. Лутц. – СПб. : Символ-Плюс, 2011. – 992 с.

Методы компьютерной обработки изображений / под ред. В. А. Сойфера. – М. : Физматлит, 2003. – 784 с.

Обработка и анализ изображений в задачах машинного зрения : курс лекций и практ. занятий / Ю. В. Визильтер и др. – М. : Физматкнига, 2010. – 672 с.

Официальный сайт языка Python [Электронный ресурс]. – Режим доступа: http://python.org/.

Официальный сайт разработчиков библиотеки OpenCV [Электронный ресурс]. – Режим доступа: http://opencv.org.

Прохоренок, Н. А. Python. Самое необходимое / Н. А. Прохоренок. – СПб. : БХВ-Петербург, 2011. – 416 с.

Разработка мультимедийных приложений с использованием библиотек OpenCV и IPP [Электронный ресурс] /А. В. Бовыкин [и др.]. – М. : ИНТУИТ, 2016. – 515 с. – Режим доступа: http://www.iprbooksshop.ru/39564/

Саммерфилд, М. Программирование на Python 3. Подробное руководство / М. Саммерфилд. – СПб. : Символ-Плюс, 2009. – 608 с.

Техническое зрение роботов / В. И. Мошкин и др. – М. : Машиностроение, 1990.– 272 с.

Хахаев, И. А. Практикум по алгоритмизации и программированию на Python / И. А. Хахаев. − М. : Альт Линукс, 2010. − 126 с.

Федоров, Д. Ю. Основы программирования на примере языка Python : учеб. пособие / Д. Ю. Федоров. – СПб. : СПбГЭУ, 2016. – 176 с.

Форсайт, Дэвид А. Компьютерное зрение. Современный подход / Девид А. Форсайт, Жан Понс. − М. : Вильямс, 2004. − 928 с.

Viola, P. Rapid object detection using a boosted cascade of simple features /P. Viola, M. J. Jones // IEEE Conf. on Computer Vision and Pattern Recognition. – Kauai, Hawaii, USA, 2001. – V. 1. – P. 511–518.

Viola, P. Robust real_time face detection / P. Viola, M. J. Jones // International Journal of Computer Vision. – 2004. – V. 57. – № 2. – P. 137–154.

ЗМІСТ

CONTENT

Навчальне видання

Краснов Леонід Олександрович
Гавриленко Олена Володимирівна

ОБ'ЄКТНО-ОРІЄНТОВАНЕ ПРОЕКТУВАННЯ СИСТЕМ КЕРУВАННЯ
(з використанням Python і бібліотеки OpenCV)

(Українською та англійською мовами)

Редактори: М. Л. Гелетка, Т. О. Іващенко

Зв. план, 2020
Підписано до друку 14.07.2020
Формат 60×84 1/16. Папір офс. № 2. Офс. друк
Ум. друк. арк. 10,2. Обл.-вид. арк. 11,5. Наклад 50 пр.
Замовлення 182. Ціна вільна

Видавець і виготовлювач
Національний аерокосмічний університет ім. М. Є. Жуковського
«Харківський авіаційний інститут»
61070, Харків-70, вул. Чкалова, 17
http://www.khai.edu
Видавничий центр «ХАІ»
61070, Харків-70, вул. Чкалова, 17
izdat@khai.edu

Свідоцтво про внесення суб`єкта видавничої справи
до Державного реєстру видавців, виготовлювачів і розповсюджувачів
видавничої продукції сер. ДК № 391 від 30.03.2001