Computer Vision – Thị Giác Máy Tính

Nguyễn Quản Bá Hồng*

Ngày 4 tháng 7 năm 2025

Tóm tắt nội dung

This text is a part of the series *Some Topics in Advanced STEM & Beyond*:
URL: https://nqbh.github.io/advanced_STEM/.
Latest version:

- *Computer Vision – Thị Giác Máy Tính*.
  PDF: URL: https://github.com/NQBH/advanced_STEM_beyond/blob/main/computer_vision/NQBH_computer_vision.pdf.
  TEX: URL: https://github.com/NQBH/advanced_STEM_beyond/blob/main/computer_vision/NQBH_computer_vision.tex.

Mục lục

1 Basic Computer Vision

2 Digital Image Processing – Xử Lý Hình Ảnh Số

2.1 [TTB23]. David Tschumperlé, Christophe Tilmant, Vincent Barra. Le traitement numerique des images en C++. Implementation d'algorithmes avec la bibliotheque CIMG – Digital Image Processing with C++: Implementing Reference Algorithms with the CImg Library. 2023

- Amazon review.
- Intro. *Digital Image Processing with C++: Implementing Reference Algorithms with the Clmg Library* presents theory of digital image processing & implementations of algorithms using a dedicated library. Processing a digital image means transforming its content (denoising, stylizing, etc.) or extracting information to solve a given problem (object recognition, measurement, motion estimation, etc.). This book presents mathematical theories underlying digital image processing as well as their practical implementation through examples of algorithms implemented in C++ language using free & easy-to-use CImg library.

  – *Xử lý ảnh kỹ thuật số bằng C++: Triển khai thuật toán tham chiếu với thư viện Clmg* trình bày lý thuyết về xử lý ảnh kỹ thuật số & triển khai thuật toán bằng thư viện chuyên dụng. Xử lý ảnh kỹ thuật số có nghĩa là chuyển đổi nội dung của ảnh

*A Scientist & Creative Artist Wannabe. E-mail: nguyenquanbahong@gmail.com. Bến Tre City, Việt Nam.

(khử nhiễu, tạo kiểu, v.v.) hoặc trích xuất thông tin để giải quyết một vấn đề nhất định (nhận dạng đối tượng, đo lường, ước tính chuyển động, v.v.). Cuốn sách này trình bày các lý thuyết toán học cơ bản về xử lý ảnh kỹ thuật số cũng như triển khai thực tế của chúng thông qua các ví dụ về thuật toán được triển khai bằng ngôn ngữ C++ bằng thư viện CImg miễn phí & dễ sử dụng.

Chaps cover field of digital image processing in a broad way & propose practical & functional implementations of each method theoretically described. Main topics covered include filtering in spatial & frequency domains, mathematical morphology, feature extraction & applications to segmentation, motion estimation, multispectral image processing & 3D visualization.

– Các chương trình bao gồm lĩnh vực xử lý hình ảnh kỹ thuật số theo cách rộng & đề xuất các triển khai thực tế & chức năng của từng phương pháp được mô tả về mặt lý thuyết. Các chủ đề chính được đề cập bao gồm lọc trong miền không gian & tần số, hình thái toán học, trích xuất đặc điểm & ứng dụng vào phân đoạn, ước tính chuyển động, xử lý hình ảnh đa phổ & trực quan hóa 3D.

Students or developers wishing to discover or specialize in this discipline & teachers & researchers hoping to quickly prototype new algorithms or develop courses will all find in this book material to discover image processing or deepen their knowledge in this field.

- About Author. David Tschumperlé is a permanent CNRS research scientist heading IMAGE team at GREYC Laboratory in Caen, France. He's particularly interested in PDEs & variational methods for processing multi-valued images in a local or non-local way. He has authored > 40 papers in journals or conferences & is project leader of CImg & G'MIC, 2 open-source software/libraries.

  Christophe Tilmant is associate professor in computer science at Clermont-Auvergne University. His research activities include image processing & AI, where he has authored > 30 papers. His teaching includes DL, image processing & network security. He participates or leads several French research programs.

  Vincent Barra is a full processor in CS at Clermont-Auvergne University & associate director of LIMOS Lab. He teaches AI & image processing in engineering schools & master's programs. His research activities focus on n-dimensional data analysis with methodological & application aspects in various fields. He has authored > 90 paperss in journals or conferences & participates or leads several French & European research programs.

- Preface. This book is successful outcome of several years of research & experience by a trio of authors who are experts in digital image processing, combining both its theoretical aspects & its software implementations. Teach a module about image processing based on variational approaches, PDE-based & Level Sets techniques. Internship was about study & development of *diffusion PDE* methods for multi-valued images (more particularly color images), i.e., images with potentially > 3 components per pixel. Rachid Deriche selected David Tschumperlé among several applicants of promotion, because he was also a general engineer in CS, & his double background perfectly met need for a trainee who could program with ease in C/C++ while mastering theoretical aspects & those related to an efficient software implementation of developed algorithms. PDE-based methods often require several hundreds or even thousands of complex iterations to be applied to image, & need to optimize machine's processor resources is therefore all more pressing.

  After processing results, David continued his work in a PhD thesis, under supervision. & very quickly, it became obvious that we needed to develop a reference C/C++ library to process images with > 3 channels or volume images with any values (matrices, tensors, ...) to develop our research work.

  Through implemented algorithms, tests, successes & failures, David gradually built his own personal C++ library of reusable features, in order to complete his thesis work. Originality of David's research work, need to optimize & develop software that survives his PhD period & that is "reusable" by members of team constitute basis of CImg library's genesis.

  At end of David's thesis, ease of use of CImg had already seduced new PhD students & permanent members of team. At end of 2003, we decided, in agreement with Inria's development department, to distribute CImg more widely as free software, naturally using new French free license CeCILL, which had just been created jointly by Inria, CEA, & CNRS.

  > 20 years after its 1st lines of code, CImg is now an image processing library used by thousands of people around world, at heart of dozens of free projects, & just as importantly, continuously & actively maintained.

  At origin of this remarkable success is, 1st of all, nature & quality of methodological work carried out throughout doctoral program, as well as its implementation guided by development of processing algorithms that must work on images of types & modalities from field of computer vision (cameras, video, velocity fields) as well as in satellite or medical fields, in particular neuroimaging with magnetic resonance diffusion imaging & its well-known model called *diffusion tensor*.

  – Nguồn gốc của thành công đáng chú ý này trước hết là bản chất & chất lượng của công trình phương pháp luận được thực hiện trong suốt chương trình tiến sĩ, cũng như việc triển khai được hướng dẫn bởi sự phát triển của các thuật toán xử lý phải hoạt động trên hình ảnh thuộc nhiều loại & phương thức từ lĩnh vực thị giác máy tính (máy ảnh, video, trường vận tốc) cũng như trong lĩnh vực vệ tinh hoặc y tế, đặc biệt là hình ảnh thần kinh với hình ảnh khuếch tán cộng hưởng từ & mô hình nổi tiếng của nó được gọi là *tensor khuếch tán*.

  This aspect of data genericity was very quickly a central element in design & success of library. With a focus on simplicity of design & use, & a constant & coherent development of library API, authors have clearly succeeded in coupling ease of use with genericity of processing that library allows. Free distribution of library has allowed academic world, as well as research & industrial world, to discover prototyping & implementation of efficient image processing algorithms in a gentle & enjoyable way.

– Khía cạnh chung chung của dữ liệu này rất nhanh chóng trở thành yếu tố trung tâm trong thiết kế & thành công của thư viện. Với trọng tâm là tính đơn giản của thiết kế & sử dụng, & sự phát triển liên tục & mạch lạc của API thư viện, các tác giả đã thành công rõ ràng trong việc kết hợp tính dễ sử dụng với tính chung chung của quá trình xử lý mà thư viện cho phép. Phân phối miễn phí thư viện đã cho phép thế giới học thuật, cũng như thế giới nghiên cứu & công nghiệp, khám phá ra việc tạo mẫu & triển khai các thuật toán xử lý hình ảnh hiệu quả theo cách nhẹ nhàng & thú vị.

For teachers, researchers, students or engineers, this book will provide you with an introduction to vast field of image processing, as well as an introduction to CImg library for development of state-of-art algorithms.

This book is expected to spark new passions for image processing, e.g., for beginners or more experienced C++ developers who are interested in getting started in this discipline. But this book will also shed new light on field of image processing for users & readers interested in recent advances in AI, DL, & neural networks. Learn, e.g., not necessary to have neural network with 500 million weights, nor a million training images, to extract edges of an image, to segment it, to detect geometric features as segments & circles, or objects located in it, to estimate displacement vectors in video sequences, etc. & even better, you will be able to study implementations of corresponding algorithms, disseminated & explained throughout this book, made with CImg library, while testing them on your own data.

– Cuốn sách này được kỳ vọng sẽ khơi dậy niềm đam mê mới đối với xử lý hình ảnh, ví dụ, đối với người mới bắt đầu hoặc các nhà phát triển C++ có nhiều kinh nghiệm hơn, những người quan tâm đến việc bắt đầu trong lĩnh vực này. Nhưng cuốn sách này cũng sẽ làm sáng tỏ lĩnh vực xử lý hình ảnh cho người dùng & độc giả quan tâm đến những tiến bộ gần đây trong AI, DL, & mạng nơ-ron. Học, ví dụ, không cần thiết phải có mạng nơ-ron với 500 triệu trọng số, hay một triệu hình ảnh đào tạo, để trích xuất các cạnh của một hình ảnh, để phân đoạn nó, để phát hiện các đặc điểm hình học dưới dạng các đoạn & hình tròn hoặc các đối tượng nằm trong đó, để ước tính các vectơ dịch chuyển trong chuỗi video, v.v. & thậm chí tốt hơn, bạn sẽ có thể nghiên cứu các triển khai của các thuật toán tương ứng, được phổ biến & giải thích trong suốt cuốn sách này, được tạo bằng thư viện CImg, trong khi thử nghiệm chúng trên dữ liệu của riêng bạn.

Exploring an exciting branch of science like image processing, in a reproducible way, with a free library as good as CImg, is a valuable gift that authors are offering us. Digital image processing is not only given a new life but also opens new perspectives for a bright future.

– Khám phá một nhánh khoa học thú vị như xử lý hình ảnh, theo cách có thể tái tạo, với một thư viện miễn phí tốt như CImg, là một món quà giá trị mà các tác giả đang cung cấp cho chúng ta. Xử lý hình ảnh kỹ thuật số không chỉ được trao một cuộc sống mới mà còn mở ra những viễn cảnh mới cho một tương lai tươi sáng.

- Preamble.
  - What is Image Processing? Image processing is a discipline where different scientific fields meet: signal processing, applied mathematics & CS. As a result, def of what image processing is varies according to background of person speaking about it. Signal processing is a broader discipline that consists in extracting information from a measurement or an observation that is generally perturbed by noise, distorted & where information we are looking for is not directly accessible. Word signal, coming from electrical engineering, is a generic term that represents observable quantity that can be a measurement over time (1D = time), an image (2 dimensions = 2 distances), a video sequence (3 dimensions = 2 distances + time), a volume (3 dimensions = 3 distances), a temporal volume (4 dimensions = 3 distances + time), ... Objective of signal processing, & by way of image processing: develop methods that allow to efficiently search for this information by a denoising, e.g. reconstruction or estimation process. Design of these processing methods calls upon many fields of mathematics (stochastic processes, statistics, probability, linear algebra, ...)& applied mathematics (information theory, optimization, numerical analysis, ...).

    – Xử lý hình ảnh là gì? Xử lý hình ảnh là một ngành khoa học mà nhiều lĩnh vực khoa học khác nhau gặp nhau: xử lý tín hiệu, toán học ứng dụng & CS. Do đó, định nghĩa về xử lý hình ảnh thay đổi tùy theo bối cảnh của người nói về nó. Xử lý tín hiệu là một ngành rộng hơn bao gồm việc trích xuất thông tin từ phép đo hoặc quan sát thường bị nhiễu, méo & khi thông tin chúng ta đang tìm kiếm không thể truy cập trực tiếp. Từ tín hiệu, bắt nguồn từ kỹ thuật điện, là một thuật ngữ chung biểu thị đại lượng quan sát được có thể là phép đo theo thời gian (1D = thời gian), hình ảnh (2 chiều = 2 khoảng cách), chuỗi video (3 chiều = 2 khoảng cách + thời gian), thể tích (3 chiều = 3 khoảng cách), thể tích thời gian (4 chiều = 3 khoảng cách + thời gian), ... Mục tiêu của xử lý tín hiệu, & thông qua xử lý hình ảnh: phát triển các phương pháp cho phép tìm kiếm thông tin này một cách hiệu quả bằng quy trình khử nhiễu, ví dụ như quy trình tái tạo hoặc ước tính. Thiết kế các phương pháp xử lý này đòi hỏi nhiều lĩnh vực toán học (quy trình ngẫu nhiên, thống kê, xác suất, đại số tuyến tính, ...)& toán học ứng dụng (lý thuyết thông tin, tối ưu hóa, phân tích số, ...).

    Practical realization of these methods depends on nature of image. In vast majority of cases, they are in digital form, i.e., sampled & quantified signals. One carries out *digital image processing* which processes data-processing algorithms on numerical machines (computers or dedicated circuits).

    – Việc thực hiện các phương pháp này phụ thuộc vào bản chất của hình ảnh. Trong phần lớn các trường hợp, chúng ở dạng kỹ thuật số, tức là các tín hiệu được lấy mẫu & định lượng. Người ta thực hiện *xử lý hình ảnh kỹ thuật số* xử lý các thuật toán xử lý dữ liệu trên các máy tính số (máy tính hoặc mạch chuyên dụng).
  - What is an image? An image is a d dimensional signal. In order to process it, we associate this signal with notion of abstract signal. In this book only consider deterministic signals to which we associate a function. In context of random or stochastic signals, can use e.g. random processes.

To present an image processing method, can switch between a continuous representation of image for theory & its numerical representation for its computer realization.

– Để trình bày một phương pháp xử lý hình ảnh, có thể chuyển đổi giữa biểu diễn liên tục của hình ảnh cho lý thuyết & biểu diễn số của nó cho việc thực hiện trên máy tính: Continuous image: $I : Z \subset \mathbb{R}^d \to \mathbb{R}^c, (x_1, \ldots, x_d) \mapsto I(x_1, \ldots, x_d)$. Digital (or numerical) image: $I : \Omega \subset \mathbb{N}^d \to \mathbb{Z}^c, [i_1, \ldots, i_d] \mapsto I[i_1, \ldots, i_d]$. Fig. 1: Continuous & digital representations of a color image $d = 2, c = 3$: $I : Z \subset \mathbb{R}^2 \to \mathbb{R}^3$, $(x, y) \mapsto I(x, y)$, $I : \Omega \subset \mathbb{N}^2 \to \mathbb{Z}^3$, $[i, j] \mapsto I[i, j]$.

Conversion of a continuous signal (or image) to a digital (or numerical) signal (or image) is carried out in 2 stages: Fig. 2: Principle of sampling & quantization. Example with 16 samples & a quantization on 2 bits ($2^2 = 4$ discrete values): Continuous-time signal $\mapsto$ Discrete-time signal $\mapsto$ Digital signal.

* Sampling: discretize evolution parameters (time, distances)
* Quantization: discretize signal values.

– Chuyển đổi tín hiệu liên tục (hoặc hình ảnh) thành tín hiệu kỹ thuật số (hoặc số) (hoặc hình ảnh) được thực hiện theo 2 giai đoạn: Hình 2: Nguyên lý lấy mẫu & lượng tử hóa. Ví dụ với 16 mẫu & lượng tử hóa trên 2 bit ($2^2 = 4$ giá trị rời rạc): Tín hiệu thời gian liên tục $\mapsto$ Tín hiệu thời gian rời rạc $\mapsto$ Tín hiệu kỹ thuật số.

* Lấy mẫu: các tham số tiến hóa rời rạc (thời gian, khoảng cách)
* Lượng tử hóa: các giá trị tín hiệu rời rạc.

PART I: INTRODUCTION TO CImg.

- 1. Introduction.

  - Whom is this book for? With this book, offer an enchanting, yet pragmatic walk through wonderful world of image processing:
    * *Enchanting*, because image processing is a vast & captivating universe, resting on diversified but solid theoretical bases, which are used as well to model & propose efficient algorithms to solve problems. Number of amusing &/or practical applications that can be imaged & implemented in image processing is potentially infinite.
    * *Pragmatic*, because we do not want to make a simple overview of different formalisms. Try to make each of techniques discussed tangible, by systematically "translating" theoretical points presented in form of functional & usable C++ programs.

    – Với cuốn sách này, hãy cung cấp một chuyến đi hấp dẫn nhưng thực tế qua thế giới tuyệt vời của xử lý hình ảnh:

    * *Hấp dẫn*, vì xử lý hình ảnh là một vũ trụ & hấp dẫn, dựa trên các cơ sở lý thuyết đa dạng nhưng vững chắc, cũng được sử dụng để mô hình hóa & đề xuất các thuật toán hiệu quả để giải quyết các vấn đề. Số lượng &ứng dụng thực tế hoặc thú vị có thể được hình ảnh hóa & triển khai trong xử lý hình ảnh có khả năng là vô hạn.
    * *Thực tế*, vì chúng tôi không muốn đưa ra một cái nhìn tổng quan đơn giản về các hình thức khác nhau. Cố gắng làm cho từng kỹ thuật được thảo luận trở nên hữu hình, bằng cách "dịch" một cách có hệ thống các điểm lý thuyết được trình bày dưới dạng các chương trình C++ chức năng & có thể sử dụng.

    This intertwining of theory & implementation is essence of this book, & its content is therefore intended for a variety of readers:

    * *C++ programmers, beginners or experts, amateurs of applied mathematics*, wishing to study discipline of image processing, to eventually develop concrete applications in this field.
    * *Mathematicians, image or signal processing makers*, wishing to confront practical problem of software implementation of various methods of domain in C++, a language universally recognized as being generic, powerful & fast at execution.
    * *Teachers or students in CS & applied mathematics*, who will find in this book basic building blocks to develop or perform their practical work.
    * *Developers or experienced researchers*, for whom CImg library described in this book will be an ideal companion for rapid & efficient prototyping of new innovative image processing algorithms.

    – Sự đan xen giữa lý thuyết & triển khai là cốt lõi của cuốn sách này, & do đó, nội dung của nó dành cho nhiều đối tượng độc giả:

    * *Lập trình viên C++, người mới bắt đầu hoặc chuyên gia, người nghiệp dư về toán ứng dụng*, muốn nghiên cứu chuyên ngành xử lý hình ảnh, để cuối cùng phát triển các ứng dụng cụ thể trong lĩnh vực này.
    * *Các nhà toán học, nhà sản xuất xử lý hình ảnh hoặc tín hiệu*, muốn đối mặt với vấn đề thực tế về triển khai phần mềm của nhiều phương pháp miền khác nhau trong C++, một ngôn ngữ được công nhận rộng rãi là chung chung, mạnh mẽ & thực thi nhanh.
    * *Giáo viên hoặc sinh viên ngành CS & toán ứng dụng*, những người sẽ tìm thấy trong cuốn sách này các khối xây dựng cơ bản để phát triển hoặc thực hiện công việc thực tế của họ.
    * *Các nhà phát triển hoặc nhà nghiên cứu giàu kinh nghiệm*, những người mà thư viện CImg được mô tả trong cuốn sách này sẽ là người bạn đồng hành lý tưởng để tạo mẫu nhanh & hiệu quả các thuật toán xử lý hình ảnh sáng tạo mới.

    Important to underline: only use simple concepts of C++ language & proposed programs will therefore be readable enough to easily transcribed into other languages if necessary. CImg library, on which we rely has been developed for several years by researchers in CS & image processing (from CNRS – French National Centre for Scientific Research, INRIA – French

National Institute for Research in Digital Science & Technology, & University), mainly to allow rapid prototyping of new algorithms. Also used as a development tool in practical work of several courses given at bachelor's, master's or engineering school level. Its use is therefore perfectly adapted to pedagogical approach that we wish to develop in this book.

– Điều quan trọng cần nhấn mạnh: chỉ sử dụng các khái niệm đơn giản của ngôn ngữ C++ & do đó các chương trình được đề xuất sẽ đủ dễ đọc để dễ dàng chuyển sang các ngôn ngữ khác nếu cần. Thư viện CImg, mà chúng tôi dựa vào đã được phát triển trong nhiều năm bởi các nhà nghiên cứu về CS & xử lý hình ảnh (từ CNRS – Trung tâm Nghiên cứu Khoa học Quốc gia Pháp, INRIA – Viện Nghiên cứu Khoa học Kỹ thuật số Quốc gia Pháp & Công nghệ, & Đại học), chủ yếu để cho phép tạo mẫu nhanh các thuật toán mới. Cũng được sử dụng như một công cụ phát triển trong công việc thực tế của một số khóa học được cung cấp ở cấp độ cử nhân, thạc sĩ hoặc trường kỹ thuật. Do đó, việc sử dụng nó hoàn toàn phù hợp với phương pháp sư phạm mà chúng tôi muốn phát triển trong cuốn sách này.

Book is structured to allow, on 1 hand, a quick appropriation of concepts of CImg library (which motivates 1st part of this book), & on other hand, its practical use in many fields of image processing, through various workshops (constituting 2nd part of book). Set of examples proposed, ranging from simplest application to more advanced algorithms, helps developing a joint know-how in theory, algorithmic & implementation in field of image processing. Source codes published in this book are also available in digital format, on repository: https://github.com/CImg-Image-Processing-Book.

- Why still do image processing today? Image is everywhere. It is medium of many types of information, is used in various applications e.g. medical diagnosis using MRI (Magnetic Resonance Imaging), scanner or scintigraphic imaging, study of deforestation by satellite imagery, detection of abnormal behaviors in crowds from video acquisitions, photographic retouching & special effects, or handwriting recognition, to name but a few. Today, even without realizing it, use image processing algorithms on a daily basis: automatic contrast enhancement of our favorite vacation photos, detection of license plates at entrance of supermarket parking, automatic detection of faces or places in photographs that we post on social networks, etc.

  – Tại sao ngày nay vẫn xử lý hình ảnh? Hình ảnh ở khắp mọi nơi. Nó là phương tiện truyền tải nhiều loại thông tin, được sử dụng trong nhiều ứng dụng khác nhau, ví dụ như chẩn đoán y khoa bằng MRI (Chụp cộng hưởng từ), chụp ảnh bằng máy quét hoặc chụp cắt lớp, nghiên cứu nạn phá rừng bằng hình ảnh vệ tinh, phát hiện hành vi bất thường trong đám đông từ các lần thu video, chỉnh sửa ảnh & hiệu ứng đặc biệt, hoặc nhận dạng chữ viết tay, v.v. Ngày nay, ngay cả khi không nhận ra, chúng ta vẫn sử dụng các thuật toán xử lý hình ảnh hàng ngày: tự động tăng cường độ tương phản cho những bức ảnh kỳ nghỉ yêu thích của mình, phát hiện biển số xe ở lối vào bãi đậu xe của siêu thị, tự động phát hiện khuôn mặt hoặc địa điểm trong những bức ảnh mà chúng ta đăng trên mạng xã hội, v.v.

  Image processing is based on a solid theoretical background, derived from signal processing & Shannon's information theory [38]. Processing an image aims to extract 1 or several relevant pieces of information for a given problem: size of an object, its localization, its movement, its color, even its identification. Extracting this information may require some pre-processing steps, if image is too noisy or badly contrasted.

  – Xử lý hình ảnh dựa trên nền tảng lý thuyết vững chắc, bắt nguồn từ lý thuyết xử lý tín hiệu & thông tin của Shannon [38]. Xử lý hình ảnh nhằm mục đích trích xuất 1 hoặc một số thông tin có liên quan cho một vấn đề nhất định: kích thước của một đối tượng, vị trí của nó, chuyển động của nó, màu sắc của nó, thậm chí là nhận dạng của nó. Việc trích xuất thông tin này có thể yêu cầu một số bước tiền xử lý, nếu hình ảnh quá nhiễu hoặc có độ tương phản kém.

  Image processing took off in 1960s, with advent of computers & development (or rediscovery) of signal processing techniques (e.g., Fourier transform). From then on, in all domains that this book proposes to approach in its 2nd part, many algorithms have been developed, always more powerful & precise, that are able to process images of increasingly important size & in consequent number.

  – Xử lý hình ảnh bắt đầu vào những năm 1960, với sự ra đời của máy tính & phát triển (hoặc tái khám phá) các kỹ thuật xử lý tín hiệu (ví dụ, biến đổi Fourier). Từ đó trở đi, trong tất cả các lĩnh vực mà cuốn sách này đề xuất tiếp cận trong phần thứ 2, nhiều thuật toán đã được phát triển, luôn mạnh mẽ hơn & chính xác hơn, có khả năng xử lý hình ảnh có kích thước ngày càng quan trọng & với số lượng lớn hơn.

  In parallel to this development, since 2000s, ML, & more particularly DL, has achieved unequaled performance in computer vision, even surpassing human capacities in certain areas. A deep neural network is now able to annotate a scene by identifying all objects, to realistically colorize a grayscale image, or to restore highly noisy images.

  – Song song với sự phát triển này, kể từ những năm 2000, ML, & cụ thể hơn là DL, đã đạt được hiệu suất vô song trong thị giác máy tính, thậm chí vượt qua khả năng của con người trong một số lĩnh vực. Một mạng nơ-ron sâu hiện có thể chú thích một cảnh bằng cách xác định tất cả các đối tượng, tô màu thực tế cho hình ảnh thang độ xám hoặc khôi phục hình ảnh có độ nhiễu cao.

  So why are we still interested in "classical" image processing? Shift from image processing to DL is accompanied by a paradigm shift in data processing: classical techniques 1st compute features on original images (Chap. 6) & then use them for actual processing (segmentation: Chap. 7, tracking: Chap. 8, ...). Computation of these features is possibly preceded by pre-processing (Chaps. 3–5) facilitating their extraction. In contrast, DL *learns* these features, most often through convolution layers in deep networks, & uses these learned features to perform processing.

  – Vậy tại sao chúng ta vẫn quan tâm đến xử lý hình ảnh "cổ điển"? Sự chuyển dịch từ xử lý hình ảnh sang DL đi kèm với sự thay đổi mô hình trong xử lý dữ liệu: các kỹ thuật cổ điển đầu tiên tính toán các đặc điểm trên hình ảnh gốc (Chương 6) & sau đó sử dụng chúng để xử lý thực tế (phân đoạn: Chương 7, theo dõi: Chương 8, ...). Việc tính toán các đặc điểm này có thể được thực hiện trước bằng cách xử lý trước (Chương 3–5) để tạo điều kiện cho việc trích xuất chúng. Ngược lại,

DL *học* các đặc điểm này, thường là thông qua các lớp tích chập trong các mạng sâu, & sử dụng các đặc điểm đã học này để thực hiện xử lý.

& this is where major difference comes in: to perform its task, deep network must *learn*. & to do this, it must have a training set made up of several thousands (or even millions) of examples, telling it what it must do. E.g., to be able to recognize images of cats & dogs, network must learn on thousands of pairs (x, y), where x is an image of a cat or a dog & y is associated label, before being able to decide on an unknown image Fig. 1.1: Principle of learning in image processing: Learning → Inferring.

– & đây là nơi sự khác biệt chính xuất hiện: để thực hiện nhiệm vụ của mình, mạng sâu phải *học*. & để làm được điều này, nó phải có một tập huấn luyện bao gồm hàng nghìn (hoặc thậm chí hàng triệu) ví dụ, cho nó biết nó phải làm gì. Ví dụ, để có thể nhận dạng hình ảnh mèo & chó, mạng phải học trên hàng nghìn cặp (x, y), trong đó x là hình ảnh mèo hoặc chó & y là nhãn liên quan, trước khi có thể quyết định một hình ảnh chưa biết Hình 1.1: Nguyên tắc học trong xử lý hình ảnh: Học → Suy ra.

However, obtaining this data is far from being an easy task. If, in some domains (e.g. object recognition), well-established labeled databases are available (e.g., ImageNet http://www.image-net.org, composed of > 14 million different images distributed in 21800 categories), most often very difficult, if not impossible, to constitute a sufficiently well-supplied & constituted training set to train a neural network.

– Tuy nhiên, việc thu thập dữ liệu này không phải là một nhiệm vụ dễ dàng. Nếu trong một số lĩnh vực (ví dụ: nhận dạng đối tượng), có sẵn các cơ sở dữ liệu được gắn nhãn tốt (e.g. ImageNet http://www.image-net.org, bao gồm > 14 triệu hình ảnh khác nhau được phân phối trong 21800 danh mục), thì thường rất khó, nếu không muốn nói là không thể, để tạo thành một bộ đào tạo được cung cấp đủ tốt & được cấu thành để đào tạo một mạng nơ-ron.

Beyond this problem of training data availability, deep neural networks often require, during their learning phase, significant computing power & hardware resources (via use of GPUs – Graphics Processing Units – & TPUs – Tensor Processing Units). Power that student, or engineer in search of a quick result, will not necessarily have at his disposal.

– Ngoài vấn đề về tính khả dụng của dữ liệu đào tạo, mạng nơ-ron sâu thường đòi hỏi, trong giai đoạn học của chúng, sức mạnh tính toán & tài nguyên phần cứng đáng kể (thông qua việc sử dụng GPU – Đơn vị xử lý đồ họa – & TPU – Đơn vị xử lý Tensor). Sức mạnh mà sinh viên hoặc kỹ sư đang tìm kiếm kết quả nhanh chóng, không nhất thiết phải có trong tay.

So, even if field of deep neural network learnng has been expanding rapidly for last 20 years, & provides impressive results, classical image processing has certainly not said its last word!

– Vì vậy, ngay cả khi lĩnh vực học mạng nơ-ron sâu đã phát triển nhanh chóng trong 20 năm qua và mang lại những kết quả ấn tượng thì xử lý hình ảnh cổ điển chắc chắn vẫn chưa nói lời cuối cùng!

- Why do image processing in C++? Among plethora of existing programming languages, C++ language has following advantages:
  - It is a *multi-paradigm, well-established, & popular* language. It is generally taught in universities & engineering schools offering CS related courses. It therefore reaches a wide audience, who will be able to use it to write programs addressing a wide range of problems, in order to solve various tasks, both at "low-level" & "high-level".
  - C++ is a *compiled* language, which produces highly *optimized* binaries. In image processing, data to be processed is often large: a standard resolution image has several million values to analyze, & it is therefore important to have programs that are fast enough to iterate on these values within a reasonable time, which is not always possible with interpreted languages. In Python, e.g., most of existing modules for image processing are implemented in C/C++, for speed issues (if have already tested looping over all pixels of an image with a "pure" Python loop, guess why).
  - Use of C++ *templates* eases manipulation of *generic* image data, e.g., when pixel values of images you process have different numerical types (Boolean, integer, floating point, etc.).

  – Tại sao xử lý hình ảnh bằng C++? Trong số rất nhiều ngôn ngữ lập trình hiện có, ngôn ngữ C++ có những ưu điểm sau:
  - Đây là ngôn ngữ *đa mô hình, đã được thiết lập tốt, & phổ biến*. Ngôn ngữ này thường được giảng dạy tại các trường đại học & trường kỹ thuật cung cấp các khóa học liên quan đến CS. Do đó, ngôn ngữ này tiếp cận được nhiều đối tượng, những người có thể sử dụng ngôn ngữ này để viết các chương trình giải quyết nhiều vấn đề khác nhau, nhằm giải quyết nhiều nhiệm vụ khác nhau, ở cả "mức thấp" & "mức cao".
  - C++ là ngôn ngữ *được biên dịch*, tạo ra các tệp nhị phân được *tối ưu hóa* cao. Trong xử lý hình ảnh, dữ liệu cần xử lý thường rất lớn: một hình ảnh có độ phân giải chuẩn có hàng triệu giá trị để phân tích, & do đó, điều quan trọng là phải có các chương trình đủ nhanh để lặp lại các giá trị này trong thời gian hợp lý, điều này không phải lúc nào cũng khả thi với các ngôn ngữ được thông dịch. Trong Python, ví dụ, hầu hết các mô-đun hiện có để xử lý hình ảnh đều được triển khai bằng CC++, vì vấn đề về tốc độ (nếu đã thử lặp qua tất cả các pixel của hình ảnh bằng vòng lặp Python "thuần túy", hãy đoán xem tại sao).
  - Sử dụng C++ *templates* giúp dễ dàng thao tác dữ liệu hình ảnh chung, ví dụ, khi các giá trị pixel của hình ảnh bạn xử lý có các kiểu số khác nhau (Boolean, số nguyên, dấu phẩy động, v.v.).
- Why use an external library? Classically, programming image processing algorithms requires ability to import/export images in form of *arrays of values*. Images are usually stored on disk in standardized file formats JPEG, PNG, TIFF, ... Displaying them on screen is also often desirable. However, no such functionality is present in standard C++ library, neither for loading or saving image files, nor for analyzing, processing, & visualizing images.

  – Tại sao lại sử dụng thư viện bên ngoài? Theo truyền thống, lập trình thuật toán xử lý hình ảnh đòi hỏi khả năng nhập/xuất hình ảnh dưới dạng *mảng giá trị*. Hình ảnh thường được lưu trữ trên đĩa ở các định dạng tệp chuẩn JPEG, PNG, TIFF, ...

Hiển thị chúng trên màn hình cũng thường được mong muốn. Tuy nhiên, không có chức năng nào như vậy trong thư viện C++ chuẩn, không phải để tải hoặc lưu tệp hình ảnh, cũng không phải để phân tích, xử lý, & trực quan hóa hình ảnh.

One has to realize that writing such features from scratch is actually a tedious task. Today, classic file formats have indeed a very complex binary structure: images are mostly stored on disk in *compressed* form, each format uses its own compression method that can be destructive or not. In practice, each image format is associated with an advanced 3rd-party library (e.g., `libjpeg, libpng, libtiff`, ...), each being focused on loading & saving image data in its own file format. Similarly, displaying an image in a window is a more complex task than it seems, & is always done through use of 3rd-party libraries, either specialized in "raw" display (`libX11, Wayland` under Unix, or `gdi32` under Windows), or in display of more advanced graphical interfaces with *widgets* (`GTK, Qt`, ...).

– Người ta phải nhận ra rằng việc viết các tính năng như vậy từ đầu thực sự là một nhiệm vụ tẻ nhạt. Ngày nay, các định dạng tệp cổ điển thực sự có cấu trúc nhị phân rất phức tạp: hình ảnh chủ yếu được lưu trữ trên đĩa dưới dạng *compression*, mỗi định dạng sử dụng phương pháp nén riêng có thể phá hủy hoặc không. Trong thực tế, mỗi định dạng hình ảnh được liên kết với một thư viện của bên thứ 3 nâng cao (ví dụ: `libjpeg, libpng, libtiff`, ...), mỗi định dạng tập trung vào việc tải & lưu dữ liệu hình ảnh ở định dạng tệp riêng của nó. Tương tự như vậy, việc hiển thị hình ảnh trong cửa sổ là một nhiệm vụ phức tạp hơn vẻ bề ngoài, & luôn được thực hiện thông qua việc sử dụng các thư viện của bên thứ 3, chuyên về hiển thị "thô" (`libX11, Wayland` trên Unix hoặc `gdi32` trên Windows) hoặc trong hiển thị các giao diện đồ họa nâng cao hơn với *widget* (`GTK, Qt`, ...).

Basic processing algorithms themselves are not always trivial to implement, especially when optimized versions are required. For all these reasons, one usually resorts to a high-level 3rd-party library specialized in image processing, to work comfortably in this domain in C++.

– Bản thân các thuật toán xử lý cơ bản không phải lúc nào cũng dễ thực hiện, đặc biệt là khi cần các phiên bản được tối ưu hóa. Vì tất cả những lý do này, người ta thường dùng đến một thư viện của bên thứ 3 cấp cao chuyên về xử lý hình ảnh để làm việc thoải mái trong lĩnh vực này bằng C++.

- Which C++ libraries for image processing? A relevant image processing library should allow reading/writing of image data in most common file formats, display of these images, & should propose a few of most classical algorithms for image processing. Among dozens of existing choices, propose this purified list of libraries, verifying these minimal conditions: `CImg, ITK, libvips, Magick++, OpenCV, VTK`. Why on these 6 libraries? Because they are well-established ones (all of them existing for > 15 years), widely used in image processing community & therefore well-proven in terms of performance & robustness. They are also still under active development, free to use, multi-platform, & extensive enough to allow development of complex & diversified image processing programs. Have voluntarily put aside libraries that are either distributed under a proprietary license, or that are too young, or not actively maintained, or with a too restrictive application domain (e.g., libraries that are only capable of reading/writing images in a few file formats, or that propose a too limited panel of image processing algorithms). This diversity of choice actually reflects various application domains that were initially targeted by authors of these different libraries.

  – Thư viện C++ nào để xử lý hình ảnh? Một thư viện xử lý hình ảnh có liên quan phải cho phép đọcghi dữ liệu hình ảnh ở hầu hết các định dạng tệp phổ biến, hiển thị những hình ảnh này, & phải đề xuất một số thuật toán cổ điển nhất để xử lý hình ảnh. Trong số hàng chục lựa chọn hiện có, hãy đề xuất danh sách thư viện đã được tinh chế này, xác minh các điều kiện tối thiểu sau: `CImg, ITK, libvips, Magick++, OpenCV, VTK`. Tại sao lại là 6 thư viện này? Bởi vì chúng là những thư viện đã được thiết lập tốt (tất cả đều đã tồn tại trong > 15 năm), được sử dụng rộng rãi trong cộng đồng xử lý hình ảnh & do đó đã được chứng minh rõ ràng về hiệu suất & độ mạnh mẽ. Chúng cũng vẫn đang được phát triển tích cực, miễn phí sử dụng, đa nền tảng, & đủ rộng để cho phép phát triển các chương trình xử lý hình ảnh phức tạp & đa dạng. Đã tự nguyện loại bỏ các thư viện được phân phối theo giấy phép độc quyền, hoặc quá mới, hoặc không được duy trì tích cực, hoặc có phạm vi ứng dụng quá hạn chế (ví dụ: các thư viện chỉ có khả năng đọcghi hình ảnh ở một vài định dạng tệp hoặc đề xuất một bảng thuật toán xử lý hình ảnh quá hạn chế). Sự đa dạng về lựa chọn này thực sự phản ánh các phạm vi ứng dụng khác nhau mà ban đầu được các tác giả của các thư viện khác nhau này nhắm tới.

  Our selection can be summarized as follows:

  * *CImg (Cool Image)* was created in early 2000s by French National Institute for Research in Digital Science & Technology (Inria), a French public research institute. It is an open source library that was designed in context of research in image processing algorithms, in order to allow its users (initially, mainly researchers, teachers, & PhD students) to conceive, easily implement & test new original image processing algorithms, even from scratch. CImg can be downloaded from http://cimg.eu.

    – *CImg (Cool Image)* được Viện Nghiên cứu Khoa học Kỹ thuật số Quốc gia Pháp & Công nghệ (Inria), một viện nghiên cứu công của Pháp, tạo ra vào đầu những năm 2000. Đây là một thư viện nguồn mở được thiết kế trong bối cảnh nghiên cứu các thuật toán xử lý hình ảnh, nhằm cho phép người dùng (ban đầu, chủ yếu là các nhà nghiên cứu, giáo viên, & nghiên cứu sinh tiến sĩ) hình thành, dễ dàng triển khai & thử nghiệm các thuật toán xử lý hình ảnh gốc mới, thậm chí từ đầu. CImg có thể được tải xuống từ http://cimg.eu.

  * *ITK (Insight Segmentation & Registration Toolkit)* is a library made available in 2001, initially created for medical image analysis & processing (project was initiated by American National Library of Medicine). This medical specialization is still relevant today, & ITK is mainly used for visualization, segmentation & registration of medical images. ITK can be downloaded from https://itk.org.

    – *ITK (Insight Segmentation & Registration Toolkit)* là một thư viện được cung cấp vào năm 2001, ban đầu được tạo ra để phân tích hình ảnh y tế & xử lý (dự án được khởi xướng bởi Thư viện Y khoa Quốc gia Hoa Kỳ). Chuyên ngành y tế

này vẫn còn phù hợp cho đến ngày nay, & ITK chủ yếu được sử dụng để trực quan hóa, phân đoạn & đăng ký hình ảnh y tế. ITK có thể được tải xuống từ https://itk.org.

* `libvips` is a library written mainly in C, in 1990s, specialized in processing of large images. It is part of a larger framework (named VIPS) which also includes a software offering a graphical interface dedicated to image visualization & processing. It is a library well adapted when images to be analyzed or processed are very large, typically when set of images is larger than memory available on computer. `libvips` can be downloaded from https://libvips.github.io/libvips.
  – *libvips* là một thư viện được viết chủ yếu bằng C, vào những năm 1990, chuyên về xử lý hình ảnh lớn. Nó là một phần của một khuôn khổ lớn hơn (có tên là VIPS) cũng bao gồm một phần mềm cung cấp giao diện đồ họa dành riêng cho việc trực quan hóa hình ảnh & xử lý. Đây là một thư viện thích hợp khi hình ảnh cần phân tích hoặc xử lý có kích thước rất lớn, thường là khi tập hợp hình ảnh lớn hơn bộ nhớ khả dụng trên máy tính. `libvips` có thể được tải xuống từ https://libvips.github.io/libvips.
* *Magick++* is 1 of oldest libraries in our list. It was designed in late 1980s. Its original purpose was conversion of formats between 2D images in color or grayscale. However, it is not that easy to use for writing processing algorithms for generic image types (e.g., 3D volumetric images or images with > 4 channels). `Magick++` can be downloaded from https://imagemagick.org/Magick++.
  – *Magick++* là 1 trong những thư viện lâu đời nhất trong danh sách của chúng tôi. Nó được thiết kế vào cuối những năm 1980. Mục đích ban đầu của nó là chuyển đổi định dạng giữa các hình ảnh 2D theo màu hoặc thang độ xám. Tuy nhiên, nó không dễ sử dụng để viết các thuật toán xử lý cho các loại hình ảnh chung (ví dụ: hình ảnh thể tích 3D hoặc hình ảnh có > 4 kênh). `Magick++` có thể được tải xuống từ https://imagemagick.org/Magick++.
* *OpenCV* is a library developed in early 2000s, which focuses on *computer vision*, a field at crossroads of image processing & AI seeking to imitate human vision. OpenCV is a very popular library, & offers a large set of already implemented algorithms, often in a very optimized way. Ideal for a user wishing to chain together elementary algorithmic blocks in order to build efficient processing pipelines. On other hand, using OpenCV for *prototyping* new algorithms is less convenient: already implemented algorithms act like "black boxes", which are difficult to modify. Relatively complex API of library does not really facilitate writing new algorithms from scratch. OpenCV can be downloaded from https://opencv.org.
  – *OpenCV* là một thư viện được phát triển vào đầu những năm 2000, tập trung vào *computer vision*, một lĩnh vực tại ngã tư của xử lý hình ảnh & AI tìm cách mô phỏng thị giác của con người. OpenCV là một thư viện rất phổ biến, & cung cấp một bộ lớn các thuật toán đã được triển khai, thường theo cách được tối ưu hóa rất nhiều. Lý tưởng cho người dùng muốn liên kết các khối thuật toán cơ bản với nhau để xây dựng các đường ống xử lý hiệu quả. Mặt khác, sử dụng OpenCV để *tạo nguyên mẫu* các thuật toán mới ít thuận tiện hơn: các thuật toán đã được triển khai hoạt động giống như "hộp đen", rất khó để sửa đổi. API tương đối phức tạp của thư viện không thực sự tạo điều kiện thuận lợi cho việc viết các thuật toán mới từ đầu. Có thể tải xuống OpenCV từ https://opencv.org.
* VTK is a library created in mid-1990s, specialized in processing & visualization of 3D meshes, therefore focused on processing & visualization of structured data in form of graphs or meshes, rather than on processing of more traditional images defined on regular sampling grids. This library is distributed by American company Kitware, Inc., which also develops ITK library. VTK & ITK are often used together, mainly for analysis & visualization of medical images. VTK can be downloaded from https://vtk.org.
  – VTK là một thư viện được tạo ra vào giữa những năm 1990, chuyên về xử lý & trực quan hóa các lưới 3D, do đó tập trung vào xử lý & trực quan hóa dữ liệu có cấu trúc dưới dạng đồ thị hoặc lưới, thay vì xử lý các hình ảnh truyền thống hơn được xác định trên các lưới lấy mẫu thông thường. Thư viện này được phân phối bởi công ty Kitware, Inc. của Mỹ, công ty cũng phát triển thư viện ITK. VTK & ITK thường được sử dụng cùng nhau, chủ yếu để phân tích & trực quan hóa hình ảnh y tế. Có thể tải xuống VTK từ https://vtk.org.

Fig. 1.2: Logos of main opensource C++ libraries for image processing. `libvips` library does not have an official logo.

- Why did we adopt CImg for this book? CImg is a lightweight C++ library, which has been around for > 20 years. It is a free library, whose source code is open (distributed under CeCILL-C open-source license), & which runs on various OSs (Windows, Linux, Mac OSX, FreeBSD, etc.). CImg gives programmer access to classes & methods for manipulating images or sequences of images, an image being defined here in broadest sense of term, as a volumetric array with up to 3 spatial coordinates (x, y, z) & containing vector values of any size & type. Library allows programmer to be relieved of usual "low-level" tasks of manipulating images on a computer, e.g. managing memory allocations & I/O, accessing pixel values, displaying images, user interaction, etc. It also offers a fairly complete range of common processing algorithms, in several areas including:
  * Arithmetic operations between images, & applications of usual mathematical functions: `abs()`, `cos()`, `exp()`, `sin()`, `sqrt()`, ...
  * Statistical calculation: extraction of minimum, maximum, mean, variance, median value, ...
  * Color space manipulation: RGB, HSV, YC_bC_r, $L^*a^*b^*$, ...
  * Geometric transformations: rotation, mirror, crop, resize, warp, ...
  * Filtering: convolution/correlation, Fourier transform, Gradient & Laplacian computation, recursive filters, morphological filters, anisotropic smoothing, ...
  * Feature extraction: interpolated values, connected components labeling, distance function, ...
  * Matrix calculus: SVD & LU decomposition, eigenvalues/eigenvectors, linear system solver, least-square solver, ...
  * Graphical primitive drawing: segments, polygons, ellipses, splines, text, 3D mesh objects, ...

* Evaluation of mathematical expressions, specified as character strings.

– Tại sao chúng tôi áp dụng CImg cho cuốn sách này? CImg là một thư viện C++ nhẹ, đã có từ > 20 năm. Đây là một thư viện miễn phí, có mã nguồn mở (phân phối theo giấy phép nguồn mở CeCILL-C), & chạy trên nhiều hệ điều hành khác nhau (Windows, Linux, Mac OSX, FreeBSD, v.v.). CImg cung cấp cho lập trình viên quyền truy cập vào các lớp & phương thức để thao tác hình ảnh hoặc chuỗi hình ảnh, hình ảnh được định nghĩa ở đây theo nghĩa rộng nhất, là một mảng thể tích có tối đa 3 tọa độ không gian (x, y, z) & chứa các giá trị vectơ có bất kỳ kích thước & loại nào. Thư viện cho phép lập trình viên thoát khỏi các tác vụ "cấp thấp" thông thường là thao tác hình ảnh trên máy tính, ví dụ: quản lý phân bổ bộ nhớ & I/O, truy cập giá trị pixel, hiển thị hình ảnh, tương tác của người dùng, v.v. Nó cũng cung cấp một loạt các thuật toán xử lý phổ biến khá đầy đủ, trong một số lĩnh vực bao gồm:

* Các phép toán số học giữa các hình ảnh, & ứng dụng của các hàm toán học thông thường: `abs()`, `cos()`, `exp()`, `sin()`, `sqrt()`, ...
* Tính toán thống kê: trích xuất giá trị tối thiểu, tối đa, trung bình, phương sai, trung vị, ...
* Thao tác không gian màu: RGB, HSV, $YC_bC_r, L^*a^*b^*, \ldots$
* Biến đổi hình học: xoay, phản chiếu, cắt, thay đổi kích thước, cong vênh, ...
* Lọc: tương quan tích chập, biến đổi Fourier, tính toán Laplacian & Gradient, bộ lọc đệ quy, bộ lọc hình thái, làm mịn dị hướng, ...
* Trích xuất tính năng: giá trị nội suy, nhãn thành phần kết nối, hàm khoảng cách, ...
* Phép tính ma trận: Phân tích SVD & LU, giá trị riêng/vectơ riêng, trình giải hệ tuyến tính, trình giải bình phương nhỏ nhất, ...
* Bản vẽ nguyên thủy đồ họa: đoạn thẳng, đa giác, hình elip, đường cong spline, văn bản, đối tượng lưới 3D, ...
* Đánh giá các biểu thức toán học, được chỉ định dưới dạng chuỗi ký tự.

Compared to its competitors, properties of CImg library make it particularly interesting in a pedagogical context e.g. the one we want to develop with this book:

* CImg is a *lightweight* library, & therefore particularly easy to install & deploy. CImg as a whole is indeed implemented as a single header file `CImg.h`, which you just have to copy on your computer to use the library's features immediately.
* CImg is a *generic* library, able to manipulate indifferently 1D (signals), 2D or 3D images, or sequences of images, whose pixel values are of any type (via use of C++ templates). It is therefore adaptable to all kinds of input images & can be used to tackle a wide variety of problems & applications in image processing.
* CImg is simple to understand & manipulate, since it is entirely structured in only 4 different classes & 2 namespaces. Its design does not reply on advanced C++ concepts, which makes it easy to learn & use, even for C++ beginners. It also makes it easy to reread & modify algorithms already implemented in library. Finally, its syntax, which favors elaboration of pipelines, allows writing of source codes that are both concise & readable.
* CImg is powerful. Most of its algorithms can be run in parallel, using different cores of available processor(s). Parallelization is done through use of OpenMP library, which can be optionally activated when compiling a CImg-based program.
* CImg is an open source library, whose development is currently led by GREYC (Research lab in digital science of CNRS), a public research laboratory located in Caen, France. This ensures development of CImg is scientifically & financially independent from any private interest. Source code of CImg is & will remain open, freely accessible, studyable by anyone, & thus favoring reproducibility & sharing of image processing algorithms. Its permissive free license (CeCILL-C) authorizes its use in any type of computer program (including those with closed source code, intended to be distributed under a proprietary license).

– So với các đối thủ cạnh tranh, các đặc tính của thư viện CImg khiến nó trở nên đặc biệt thú vị trong bối cảnh sư phạm, ví dụ như bối cảnh mà chúng tôi muốn phát triển với cuốn sách này:

* CImg là một thư viện *nhẹ*, & do đó đặc biệt dễ cài đặt & triển khai. CImg nói chung thực sự được triển khai dưới dạng một tệp tiêu đề duy nhất `CImg.h`, mà bạn chỉ cần sao chép trên máy tính của mình để sử dụng các tính năng của thư viện ngay lập tức.
* CImg là một thư viện *chung*, có thể thao tác không phân biệt hình ảnh 1D (tín hiệu), 2D hoặc 3D hoặc chuỗi hình ảnh, có giá trị pixel thuộc bất kỳ loại nào (thông qua việc sử dụng các mẫu C++). Do đó, nó có thể thích ứng với mọi loại hình ảnh đầu vào & có thể được sử dụng để giải quyết nhiều loại vấn đề & ứng dụng trong xử lý hình ảnh.
* CImg dễ hiểu & thao tác, vì nó được cấu trúc hoàn toàn trong chỉ 4 lớp khác nhau & 2 không gian tên. Thiết kế của nó không dựa trên các khái niệm C++ nâng cao, giúp dễ học & sử dụng, ngay cả đối với người mới bắt đầu sử dụng C++. Nó cũng giúp dễ đọc lại & sửa đổi các thuật toán đã được triển khai trong thư viện. Cuối cùng, cú pháp của nó, ưu tiên việc xây dựng các đường ống, cho phép viết các mã nguồn vừa ngắn gọn & dễ đọc.
* CImg rất mạnh mẽ. Hầu hết các thuật toán của nó có thể chạy song song, sử dụng các lõi khác nhau của bộ xử lý khả dụng. Việc song song hóa được thực hiện thông qua việc sử dụng thư viện OpenMP, có thể được kích hoạt tùy chọn khi biên dịch chương trình dựa trên CImg.
* CImg là một thư viện nguồn mở, hiện đang được GREYC (Phòng nghiên cứu khoa học kỹ thuật số của CNRS), một phòng nghiên cứu công cộng có trụ sở tại Caen, Pháp, dẫn đầu. Điều này đảm bảo rằng việc phát triển CImg là & độc lập về mặt khoa học & tài chính với bất kỳ lợi ích tư nhân nào. Mã nguồn của CImg & sẽ vẫn mở, có thể truy cập miễn phí, có thể nghiên cứu bởi bất kỳ ai, & do đó ủng hộ khả năng tái tạo & chia sẻ các thuật toán xử lý hình ảnh. Giấy phép miễn phí cho phép (CeCILL-C) cho phép sử dụng trong bất kỳ loại chương trình máy tính nào (bao gồm cả những chương trình có mã nguồn đóng, dự định phân phối theo giấy phép độc quyền).

All these features make it an excellent library for practicing image processing in C++, either to develop & prototype new algorithms from scratch, or to have a complete & powerful collection of image processing algorithms already implemented, immediately usable in one's own programs.

– Tất cả các tính năng này làm cho nó trở thành một thư viện tuyệt vời để thực hành xử lý hình ảnh trong C++, để phát triển & tạo nguyên mẫu thuật toán mới từ đầu hoặc để có một bộ sưu tập & thuật toán xử lý hình ảnh hoàn chỉnh đã được triển khai, có thể sử dụng ngay trong chương trình của riêng bạn.

- Structure of CImg library. CImg API is simple: library exposes 4 classes (2 of them with a `template` parameter) & 2 namespaces Fig. 1.3: Structure of CImg library.

CImg defines 2 namespaces:

* `cimg_library`: this namespace includes all classes & functions of library. A source code using CImg usually starts with following 2 lines:

```
#include "CImg.h"
using namespace cimg_library;
```

Thus programmer will have direct access to library classes, without having to prefix them with namespace identifier `cimg_library::`.

* `cimg`: this namespace contains some utility functions of library, which are not linked to particular classes, & which can be useful for developer. E.g., functions `cimg::sqr()` (returns square of a number), `cimg::factorial()` (returns factorial of a number), `cimg::gcd()` (returns greatest common divisor between 2 numbers) or `cimg::maxabs()` (compute maximum absolute value between 2 numbers) are some of functions defined in `cimg::` namespace.

– CImg định nghĩa 2 không gian tên:

* `cimg_library`: không gian tên này bao gồm tất cả các lớp & hàm của thư viện. Mã nguồn sử dụng CImg thường bắt đầu bằng 2 dòng sau:

```
#include "CImg.h"
using namespace cimg_library;
```

Do đó, lập trình viên sẽ có quyền truy cập trực tiếp vào các lớp thư viện mà không cần phải thêm tiền tố cho chúng bằng định danh không gian tên `cimg_library::`.

* `cimg`: không gian tên này chứa một số hàm tiện ích của thư viện, không được liên kết với các lớp cụ thể, & có thể hữu ích cho nhà phát triển. Ví dụ, các hàm `cimg::sqr()` (trả về bình phương của một số), `cimg::factorial()` (trả về giai thừa của một số), `cimg::gcd()` (trả về ước chung lớn nhất giữa 2 số) hoặc `cimg::maxabs()` (tính giá trị tuyệt đối lớn nhất giữa 2 số) là một số hàm được định nghĩa trong không gian tên `cimg::`.

CImg defines 4 classes:

* `CImg<T>`: this is most essential & populated class of library. An instance of `CImg<T>` represents an "image" that programmer can manipulate in this C++ program. Numerical type `T` of pixel values can be anything. Default type `T` is `float`, so we can write `CImg<>` instead of `CImg<float>`.
* `CImgList<T>`: this class represents a list of `CImg<T>` images. It is used e.g. to store sequences of images, or sets of images (that may have different sizes). Default type `T` is `float`, so we can write `CImgList<>` instead of `CImgList<float>`.
* `CImgDisplay`: this class represents a window that can display image on screen, & interact through user events. It can be used to display animations or to create applications requiring some user interactions (e.g., placement of key points on an image, moving them, ...).
* `CImgException`: this is class used to handle library exceptions, i.e., errors that occur when classes & functions of library are misused. Programmer never instantiates objects of this class, but can catch corresponding exceptions raised with this class by library to manage errors.

This concise design of library makes it easy to learn, even for novice C++ programmers.

– CImg định nghĩa 4 lớp:

* `CImg<T>`: đây là lớp thư viện & được điền đầy đủ nhất. Một thể hiện của `CImg<T>` biểu diễn một "hình ảnh" mà lập trình viên có thể thao tác trong chương trình C++ này. Kiểu số `T` của các giá trị pixel có thể là bất kỳ thứ gì. Kiểu mặc định `T` là `float`, vì vậy chúng ta có thể viết `CImg<>` thay vì `CImg<float>`.
* `CImgList<T>`: lớp này biểu diễn một danh sách các hình ảnh `CImg<T>`. Ví dụ, nó được sử dụng để lưu trữ chuỗi hình ảnh hoặc tập hợp hình ảnh (có thể có kích thước khác nhau). Kiểu mặc định `T` là `float`, vì vậy chúng ta có thể viết `CImgList<>` thay vì `CImgList<float>`.
* `CImgDisplay`: lớp này biểu diễn một cửa sổ có thể hiển thị hình ảnh trên màn hình, & tương tác thông qua các sự kiện của người dùng. Nó có thể được sử dụng để hiển thị hoạt ảnh hoặc để tạo các ứng dụng yêu cầu một số tương tác của người dùng (ví dụ: đặt các điểm chính trên hình ảnh, di chuyển chúng, ...).
* `CImgException`: đây là lớp được sử dụng để xử lý các ngoại lệ của thư viện, tức là các lỗi xảy ra khi các lớp & hàm của thư viện bị sử dụng sai. Lập trình viên không bao giờ khởi tạo các đối tượng của lớp này, nhưng có thể bắt các ngoại lệ tương ứng do lớp này gây ra bởi thư viện để quản lý lỗi.

Thiết kế ngắn gọn của thư viện giúp dễ học, ngay cả đối với những lập trình viên C++ mới vào nghề.

- Why only a single header file? CImg is a library distributed in a rather particular form, since it is entirely implemented in a single C++ header file, named `CImg.h`.

  – Tại sao chỉ có một tệp tiêu đề duy nhất? CImg là một thư viện được phân phối theo một hình thức khá cụ thể, vì nó được triển khai hoàn toàn trong một tệp tiêu đề C++ duy nhất, có tên là `CImg.h`.

  At 1st sight, this conception may seem surprising: in C/C++, libraries that one encounters are generally organized in form of 1 or more header files (most often, 1 header file per different structure or class defined by library), completed by a binary file (`.a` or `.so` files under Linux, `.lib` or `.dll` files under Windows), which contains library's functions in compiled form.

  – Thoạt nhìn, quan niệm này có vẻ đáng ngạc nhiên: trong CC++, các thư viện mà người ta gặp thường được tổ chức dưới dạng 1 hoặc nhiều tệp tiêu đề (thường là 1 tệp tiêu đề cho mỗi cấu trúc hoặc lớp khác nhau do thư viện xác định), được hoàn thiện bằng một tệp nhị phân (tệp `.a` hoặc `.so` trên Linux, tệp `.lib` hoặc `.dll` trên Windows), chứa các hàm của thư viện ở dạng biên dịch.

  Teaching experience with CImg has shown: 1st question raised by new users of library is: "Why is everything put in 1 file?". Answer this frequent question, by listing technical choices justifying this kind of structuring, & by pointing out advantages (& disadvantages) of it. Global answer takes into account several different aspects of C++ language, & requires consideration of following points:

  1. **Why doesn't CImg propose a pre-compilation of its functions as `.a`, `.lib`, `.so` or `.dll` binary file(s)?** Because library is *generic*. `CImg<T>` image & `CImgList<T?` image structures exposed by library have a *template* parameter `T`, which corresponds to type of pixels considered for these images. However, types `T` that will be selected by user of CImg classes are a priori *unknown*.

     – **Tại sao CImg không đề xuất biên dịch trước các hàm của nó dưới dạng tệp nhị phân `.a`, `.lib`, `.so` hoặc `.dll`?** Bởi vì thư viện là *generic*. `CImg<T>` hình ảnh & `CImgList<T?` cấu trúc hình ảnh được thư viện hiển thị có tham số *template* `T`, tương ứng với loại pixel được xem xét cho các hình ảnh này. Tuy nhiên, các loại `T` sẽ được người dùng các lớp CImg chọn là không xác định trước.

     Of course, most commonly used types `T` are in practice basic C++ types for representing numbers, i.e., `bool` (Boolean), `unsigned char` (unsigned 8-bit integer), `unsigned short` (unsigned 16-bit integer), `short` (signed 16-bit integer), `unsigned int` (unsigned 32-bit integer), `int` (signed 32-bit integer), `float` (float value, 32-bit), `double` (float value, 64-bit), etc. However, not uncommon to see source codes that uses images of other types, e.g. `CImg<void*>`, `CImg<unsigned long long>` or `CImg<std::complex>`.

     – Tất nhiên, các kiểu được sử dụng phổ biến nhất `T` trên thực tế là các kiểu C++ cơ bản để biểu diễn số, tức là `bool` (Boolean), `unsigned char` (số nguyên 8 bit không dấu), `unsigned short` (số nguyên 16 bit không dấu), `short` (số nguyên 16 bit có dấu), `unsigned int` (số nguyên 32 bit không dấu), `int` (số nguyên 32 bit có dấu), `float` (giá trị float, 32 bit), `double` (giá trị float, 64 bit), v.v. Tuy nhiên, không hiếm khi thấy các mã nguồn sử dụng hình ảnh của các kiểu khác, ví dụ: `CImg<void*>`, `CImg<unsigned long long>` hoặc `CImg<std::complex>`.

     One might think that pre-compiling methods of 2 classes `CImg<T>`, `CImgList<T>` for these 10 or so most common types `T` would be a good idea. This is to overlook fact: many of these methods take as arguments values whose types are themselves `CImg<t>` images or `CImgList<t>` image lists, with another template parameter `t` potentially different from `T`.

     – Người ta có thể nghĩ rằng việc biên dịch trước các phương thức của 2 lớp `CImg<T>`, `CImgList<T>` cho khoảng 10 kiểu phổ biến nhất này `T` sẽ là một ý tưởng hay. Đây là để bỏ qua thực tế: nhiều phương thức trong số này lấy các giá trị làm đối số có kiểu là hình ảnh `CImg<t>` hoặc danh sách hình ảnh `CImgList<t>`, với một tham số mẫu khác `t` có khả năng khác với `T`.

     E.g., method

     ```
     CImg<T>::warp(CImg<t>&, unsigned int, unsigned int, unsigned int)
     ```

     applies an arbitrary deformation field to an image `CImg<T>`. Common to use this method to deform an image of type `CImg<unsigned char>` (classic color image, with 8 bits/channel), passing as argument a deformation field of type `CImg<float>`, which contains deformation vectors with floating-point precision (sub-pixel).

     – E.g., phương pháp

     ```
     CImg<T>::warp(CImg<t>&, unsigned int, unsigned int, unsigned int)
     ```

     áp dụng một trường biến dạng tùy ý cho một hình ảnh `CImg<T>`. Phương pháp này thường được sử dụng để biến dạng một hình ảnh có kiểu `CImg<unsigned char>` (hình ảnh màu cổ điển, với kênh 8 bit/), truyền một trường biến dạng có kiểu `CImg<float>` làm đối số, chứa các vectơ biến dạng có độ chính xác dấu phẩy động (dưới pixel).

     One can easily see: multiplicity of possible combinations of types for arguments of library's methods makes it unwise to precompile these functions in form of binary files. Size of generated file(s) would simply be huge, & functions actually used by programmer would in practice only represent a tiny portion of pre-compiled functions.

     – Người ta có thể dễ dàng thấy: tính đa dạng của các kết hợp có thể có của các kiểu đối số của các phương thức thư viện khiến cho việc biên dịch trước các hàm này dưới dạng tệp nhị phân trở nên không khôn ngoan. Kích thước của tệp được tạo ra sẽ rất lớn, & các hàm thực sự được lập trình viên sử dụng trên thực tế chỉ đại diện cho một phần nhỏ các hàm được biên dịch trước.

     Correct approach is therefore to let compiler instantiate methods & functions of CImg classes only for those combinations of template types that are actually exploited in user's program. In this way, lighter & more optimized binary objects or executables are generated, compared to what would be done with a static binding to a large pre-compiled library. Main

disadvantage: functions of CImg library used in program must be compiled at same time as those of program itself, which leads to an additional compilation overhead.
– Do đó, cách tiếp cận đúng đắn là để trình biên dịch khởi tạo các phương thức & hàm của các lớp CImg chỉ cho những kết hợp của các kiểu mẫu thực sự được khai thác trong chương trình của người dùng. Theo cách này, các đối tượng nhị phân hoặc tệp thực thi nhẹ hơn & được tối ưu hóa hơn được tạo ra, so với những gì sẽ được thực hiện với liên kết tĩnh với một thư viện được biên dịch trước lớn. Nhược điểm chính: các hàm của thư viện CImg được sử dụng trong chương trình phải được biên dịch cùng lúc với các hàm của chính chương trình, dẫn đến chi phí biên dịch bổ sung.

2. **Why isn't CImg subdivided into multiple header files?** If we follow this principle of usage minimality, why doesn't CImg propose to include only classes of library user needs in his program? If he only needs object of type `CImg<T>`, why make him include a header file that also defines `CImgList<T>`? After all, that's what standard C++ library provides: if you only want to work with `std::vector`, you just have to include `<vector>` ...
– **Tại sao CImg không được chia thành nhiều tệp tiêu đề?** Nếu chúng ta tuân theo nguyên tắc tối thiểu hóa việc sử dụng này, tại sao CImg không đề xuất chỉ bao gồm các lớp nhu cầu của người dùng thư viện trong chương trình của mình? Nếu anh ta chỉ cần đối tượng có kiểu `CImg<T>`, tại sao lại bắt anh ta bao gồm một tệp tiêu đề cũng định nghĩa `CImgList<T>`? Rốt cuộc, đó là những gì thư viện C++ chuẩn cung cấp: nếu bạn chỉ muốn làm việc với `std::vector`, bạn chỉ cần bao gồm `<vector>` ...
1st, because unlike C++ standard library, CImg defines only 4 different classes, which turn out to be strongly *interdependent*. Moreover, algorithms operating on these class instances are defined as *methods* of classes, not as *external* functions acting on "containers". This differs a lot from how C++ standard library is designed.
– 1, vì không giống như thư viện chuẩn C++, CImg chỉ định nghĩa 4 lớp khác nhau, hóa ra là *phụ thuộc lẫn nhau* mạnh mẽ. Hơn nữa, các thuật toán hoạt động trên các thể hiện lớp này được định nghĩa là *phương thức* của các lớp, không phải là *hàm bên ngoài* hoạt động trên "container". Điều này khác nhiều so với cách thiết kế thư viện chuẩn C++.
In practice, methods of `CImg<T>` class need methods of `CImgList<T>` (even if this is sometimes invisible to user), simply because implementations of `CImg<T>` methods require functionality of `CImgList<T>` (& vice versa). Similarly, CImgException is a ubiquitous class in `CImg`, since it is used to handle errors that occur when library functions are misused. If programmer does not want to handle these errors, this class might seem useless to include. However, it is required during compilation, since it is obviously used by library core, which is, after all, compiled at same time as user's program.
– Trong thực tế, các phương thức của lớp `CImg<T>` cần các phương thức của `CImgList<T>` (mặc dù đôi khi người dùng không nhìn thấy), đơn giản là vì các triển khai của các phương thức `CImg<T>` yêu cầu chức năng của `CImgList<T>` (& ngược lại). Tương tự như vậy, CImgException là một lớp phổ biến trong `CImg`, vì nó được sử dụng để xử lý các lỗi xảy ra khi các hàm thư viện bị sử dụng sai. Nếu lập trình viên không muốn xử lý các lỗi này, lớp này có vẻ vô dụng khi đưa vào. Tuy nhiên, nó là bắt buộc trong quá trình biên dịch, vì rõ ràng nó được sử dụng bởi lõi thư viện, sau cùng, được biên dịch cùng lúc với chương trình của người dùng.
This class interdependence means that if we wanted to have 1 header file per CImg class, 1st thing it would do is probably include header files for other classes. From a purely technical point of view, gain from such a split would be null: 4 header files would be systematically included as soon as only 1 of classes in library is used. In consequence, CImg proposes only 1 header file, rather than 1 per class, without any real consequences on compilation time.
– Sự phụ thuộc lẫn nhau của các lớp này có nghĩa là nếu chúng ta muốn có 1 tệp tiêu đề cho mỗi lớp CImg, điều đầu tiên mà nó sẽ làm có lẽ là bao gồm các tệp tiêu đề cho các lớp khác. Theo quan điểm thuần túy về mặt kỹ thuật, lợi ích từ việc chia tách như vậy sẽ là null: 4 tệp tiêu đề sẽ được bao gồm một cách có hệ thống ngay khi chỉ sử dụng 1 trong số các lớp trong thư viện. Do đó, CImg đề xuất chỉ 1 tệp tiêu đề, thay vì 1 cho mỗi lớp, mà không có bất kỳ hậu quả thực sự nào đối với thời gian biên dịch.

3. **What are advantages of a single header file?** But fact CImg is distributed in form of a single header file is not only due to satisfaction of technical constraints bound to C++ language. In practice, this is indeed an undeniable advantage for library user:
   * **Easy to install**: copying a single file to a folder to get access to functions of a complete image processing library is comfortable (a thing that few current libraries actually offer).
   * **Lightweight & performance**: on-the-fly compilation of CImg functions means more *optimized* output binaries. Library code as well as program that uses it are compiled as a single entity. As a consequence, some of library functions can be *inlined* in final binary, bringing more performance at runtime (typically, methods for accessing pixel values in images).
   * **Fine-tuning of dependencies**: CImg's on-the-fly compilation also allows user to define specific macros that tell library to use features from a particular 3rd-party library. E.g., writing in your program:

```
#define cimg_use_tiff
#define cimg_use_jpeg
#include <CImg.h>
```

   will tell CImg to use functions of `libtiff, libjpeg` libraries when it needs to read or write images in TIFF or JPEG format (it is then of course necessary to link generated binary, statically or dynamically, with these 2 libraries). There are a lot of such *configuration macros* that can be set to activate specific features of CImg, when compiling a program. On other hand, this means: also possible to compile a CImg-based program without activating any dependencies on external libraries. This flexibility in control of dependencies is very important: using CImg does not imply an automatic dependency on dozens of "low-level" 3rd-party libraries, whose functionalities might not be used by programmer. Yet

this is what happens with most of competing image processing libraries!

* **Possibility of internally extending library**: in a similar way, defining some of these macros allows library user to insert pieces of code directly into CImg classes, while it is compiling, via an original system of *plug-ins*. Consequently, library is extensible *from outside*, without having to explicitly modify its code: a user can, if desired, add his own methods to `CImg<T>, CImgList<T>` classes (e.g., new processing algorithms). Will not use this possibility in this book, but it is an interesting approach when one wishes to develop functionalities that integrate harmoniously with rest of library API.

– **Ưu điểm của một tệp tiêu đề đơn là gì?** Nhưng thực tế CImg được phân phối dưới dạng một tệp tiêu đề đơn không chỉ do đáp ứng các ràng buộc kỹ thuật gắn liền với ngôn ngữ C++. Trên thực tế, đây thực sự là một lợi thế không thể phủ nhận đối với người dùng thư viện:

* **Dễ cài đặt**: sao chép một tệp duy nhất vào một thư mục để truy cập vào các chức năng của thư viện xử lý hình ảnh hoàn chỉnh rất tiện lợi (một tính năng mà ít thư viện hiện tại thực sự cung cấp).
* **Nhẹ & hiệu suất**: biên dịch tức thời các hàm CImg có nghĩa là nhiều tệp nhị phân đầu ra *được tối ưu hóa* hơn. Mã thư viện cũng như chương trình sử dụng nó được biên dịch dưới dạng một thực thể duy nhất. Do đó, một số hàm thư viện có thể được *nhúng* trong tệp nhị phân cuối cùng, mang lại hiệu suất cao hơn khi chạy (thường là các phương pháp truy cập giá trị pixel trong hình ảnh).
* **Tinh chỉnh các phụ thuộc**: Biên dịch tức thời của CImg cũng cho phép người dùng xác định các macro cụ thể để yêu cầu thư viện sử dụng các tính năng từ một thư viện của bên thứ 3 cụ thể. Ví dụ: viết trong chương trình của bạn:

```
#define cimg_use_tiff
#define cimg_use_jpeg
#include <CImg.h>
```

sẽ yêu cầu CImg sử dụng các hàm của thư viện `libtiff, libjpeg` khi cần đọc hoặc ghi hình ảnh ở định dạng TIFF hoặc JPEG (tất nhiên là cần phải liên kết nhị phân được tạo, tĩnh hoặc động, với 2 thư viện này). Có rất nhiều *configuration macro* có thể được thiết lập để kích hoạt các tính năng cụ thể của CImg khi biên dịch chương trình.
Mặt khác, điều này có nghĩa là: cũng có thể biên dịch chương trình dựa trên CImg mà không cần kích hoạt bất kỳ phụ thuộc nào vào các thư viện bên ngoài. Tính linh hoạt trong việc kiểm soát các phụ thuộc này rất quan trọng: sử dụng CImg không ngụ ý sự phụ thuộc tự động vào hàng chục thư viện của bên thứ 3 "cấp thấp", có chức năng mà lập trình viên có thể không sử dụng. Tuy nhiên, đây lại là điều xảy ra với hầu hết các thư viện xử lý hình ảnh cạnh tranh!

* **Khả năng mở rộng thư viện nội bộ**: theo cách tương tự, việc xác định một số macro này cho phép người dùng thư viện chèn các đoạn mã trực tiếp vào các lớp CImg, trong khi nó đang biên dịch, thông qua hệ thống *plug-in* gốc. Do đó, thư viện có thể mở rộng *từ bên ngoài*, mà không cần phải sửa đổi rõ ràng mã của nó: nếu muốn, người dùng có thể thêm các phương thức của riêng mình vào các lớp `CImg<T>, CImgList<T>` (ví dụ: thuật toán xử lý mới). Sẽ không sử dụng khả năng này trong cuốn sách này, nhưng đây là một cách tiếp cận thú vị khi người ta muốn phát triển các chức năng tích hợp hài hòa với phần còn lại của API thư viện.

But 1 of great strengths of CImg library is its ease of use, & its ability to express image processing algorithms in a clear & concise way in C++.
– Nhưng 1 trong những điểm mạnh của thư viện CImg là dễ sử dụng, & khả năng diễn đạt các thuật toán xử lý hình ảnh một cách rõ ràng & súc tích bằng C++.

- 2. Getting Started with CImg Library. CImg is structured with a minimum of classes to represent relevant & manipulable objects of library. In practice, possibilities offered by library are expressed by great diversity of methods available in its classes (in particular those of principal class `CImg<T>`, representing main "image" object). This chap illustrates actual use of CImg, by detailing development of a simple C++ source code example (about 100 lines), to implement a basic image processing application. This example has been chosen to maximize use of different classes & concepts of library, without actually requiring advanced knowledge in image processing. At end of this tutorial, will have sufficient experience with CImg to start building your own applications: you will quickly realize CImg is easy to use, not based on complex C++ concepts, & therefore perfectly suited to discover, learn & teach image processing, as well as to build prototypes or more elaborate applications in this field. 1st explain purpose of application, then details of its C++ implementation.

  – CImg được cấu trúc với tối thiểu các lớp để biểu diễn các đối tượng & có thể thao tác của thư viện. Trong thực tế, các khả năng mà thư viện cung cấp được thể hiện bằng sự đa dạng lớn các phương thức có sẵn trong các lớp của nó (đặc biệt là các phương thức của lớp chính `CImg<T>`, biểu diễn đối tượng "hình ảnh" chính). Chương này minh họa cách sử dụng thực tế của CImg, bằng cách trình bày chi tiết quá trình phát triển một ví dụ mã nguồn C++ đơn giản (khoảng 100 dòng), để triển khai một ứng dụng xử lý hình ảnh cơ bản. Ví dụ này được chọn để tối đa hóa việc sử dụng các lớp & khái niệm khác nhau của thư viện, mà không thực sự yêu cầu kiến thức nâng cao về xử lý hình ảnh. Vào cuối hướng dẫn này, bạn sẽ có đủ kinh nghiệm với CImg để bắt đầu xây dựng các ứng dụng của riêng mình: bạn sẽ nhanh chóng nhận ra CImg dễ sử dụng, không dựa trên các khái niệm C++ phức tạp, & do đó hoàn toàn phù hợp để khám phá, tìm hiểu & giảng dạy xử lý hình ảnh, cũng như để xây dựng các nguyên mẫu hoặc các ứng dụng phức tạp hơn trong lĩnh vực này. Đầu tiên, hãy giải thích mục đích của ứng dụng, sau đó là chi tiết về cách triển khai C++ của ứng dụng đó.

  - 2.1. Objective: subdivide on image into blocks. A 2D digital image is represented as a 2D array of pixels, usually in grayscale or color. If want to analyze global geometry of an image (e.g., to detect its contours or to compress its data), it can be

interesting to sudivide image into several distinct regions of interest, each having its own characteristics (flat areas, texture, contours, etc.). Image decomposition into blocks is 1 of simplest ways to achieve this subdivision: try to split

PART II: IMAGE PROCESSING USING CImg.

- 3. Point Processing Transformations.
- 4. Mathematical Morphology.
- 5. Filtering.
- 6. Feature Extraction.
- 7. Segmentation.
- 8. Motion Estimation.
- 9. Multispectral Approaches.
- 10. 3D Visualization.
- 11. & So Many Other Things.
- List of CImg Codes.

3 Wikipedia

3.1 Wikipedia/computer vision

"*Computer vision* tasks include methods for acquiring, processing, analyzing, & understanding digital images, & extraction of high-dimensional data from real world in order to produce numerical or symbolic information, e.g., in form of decisions. "Understanding" in this context signifies transformation of visual images (input to retina) into descriptions of world that make sense to thought processes & can elicit appropriate action. This image understanding can be seen as disentangling of symbolic information from image data using models constructed with aid of geometry, physics, statistics, & learning theory.

Scientific discipline of computer vision is concerned with theory behind artificial systems that extract information from images. Image data can take many forms, e.g. video sequences, views from multiple cameras, multi-dimensional data from a 3D scanner, 3D point clouds from LiDaR sensors, or medical scanning devices. Technological discipline of computer vision seeks to apply its theories & models to construction of computer vision systems.

Subdisciplines of computer vision include scene recontruction, object detection, event detection, activity recognition, video tracking, object recognition, 3D pose estimation, learning, indexing, motion estimation, visual servoing, 3D scene modeling, & image restoration.

3.1.1 Definition

3.1.2 History

3.1.3 Related fields

3.1.4 Applications

3.1.5 Typical tasks

3.1.6 System methods

3.1.7 Hardware

" – Wikipedia/computer vision

4 Miscellaneous

Tài liệu

[TTB23] David Tschumperlé, Christophe Tilmant, and Vincent Barra. *Digital Image Processing with C++: Implementing Reference Algorithms with the CImg Library*. CRC Press, 2023, p. 292.