Văn bản và soạn thảo văn bản

Nhập môn Công Nghệ Thông Tin 1

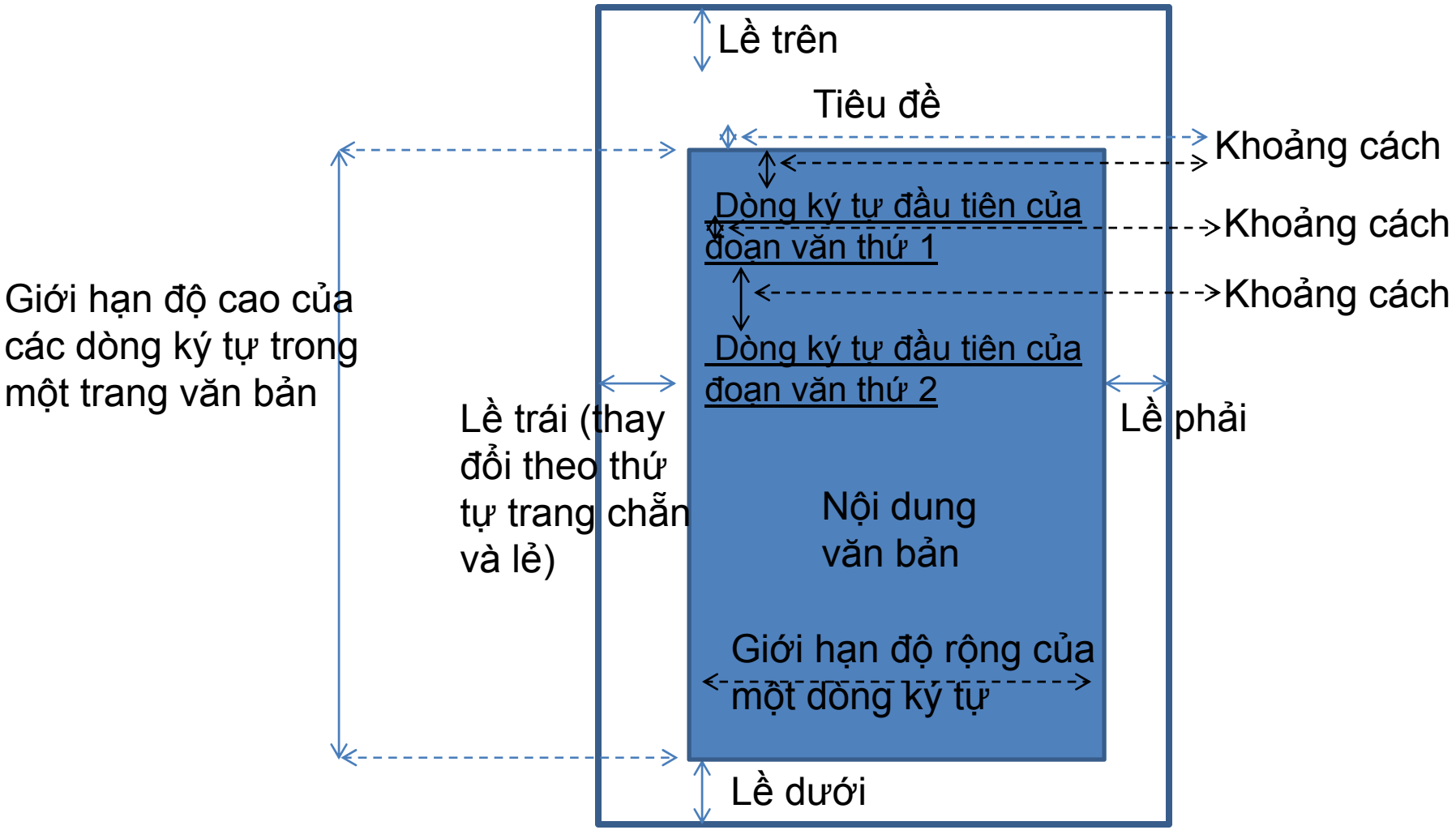
Nội dung

- Khái niệm văn bản
- Cấu trúc và qui tắc soạn thảo của một văn bản
- Soạn thảo văn bản trên máy tính

Khái niệm văn bản

- Văn bản cổ xưa ghi lại những dữ liệu và thông tin của một nền văn hóa hay triều đại.
- Lịch sử của văn bản gắn liền với lịch sử phát triển của chữ viết và lịch sử phát triển của giấy.

Cấu trúc của một trang văn bản



Các quy tắc soạn thảo văn bản cơ bản

- Quy tắc xuống dòng: khi xuống dòng không làm ngắt đôi một âm tiết, trong tiếng Anh nếu một từ bị ngắt dòng thì một phần của từ ở dòng trên được kết thúc bằng dấu "–".
- Quy tắc viết hoa: tiêu đề, ký tự đầu của một câu hay của một đoạn văn, danh từ riêng.
- Quy tắc gạch đầu dòng.
- Quy tắc khoảng trắng: mỗi ký từ cần tối thiểu 1 khoảng trắng để phân cách.

Các quy tắc soạn thảo văn bản cơ bản

- Quy tắc dấu ngoặc đơn, dấu nháy kép, dấu nháy đơn thường xuất hiện thành 1 cặp và cần được được xem như ký tự đầu từ và ký tự cuối từ nên không cần khoảng trắng phân cách với các từ nằm giữa nhưng cần khoảng trắng phân cách với các từ nằm bên ngoài.

Các quy tắc soạn thảo văn bản cơ bản

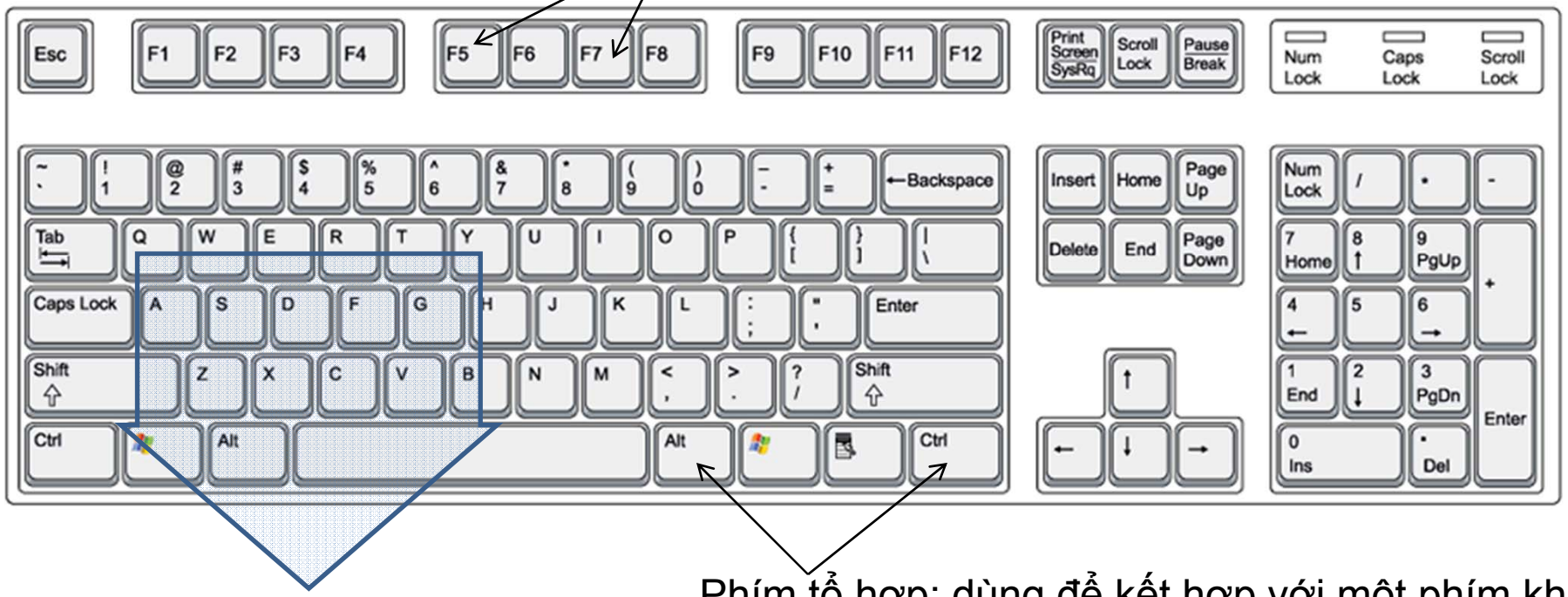
- Quy tắc dấu chấm câu, dấu phẩy, dấu chấm phẩy, dấu hỏi và dấu chấm than không cần khoảng trắng với các từ đứng trước nhưng cần khoảng trắng để phân cách với các từ đứng sau.
- Quy tắc sử dụng từ nối: "và", "hay", "nhưng" trong tiếng Việt không cần dấu phẩy đứng trước như trong tiếng Anh.

Thiết bị và các loại văn bản

- Thiết bị
  - Máy đánh chữ
  - Máy tính
- Loại văn bản
  - Văn bản hành chính (đơn, thư, công văn, báo cáo, thông báo, biên bản)
  - Báo cáo khoa học, bài báo khoa học (luận văn tốt nghiệp, bài báo hội nghị, bài báo tạp chí)
  - Bài báo phổ thông

Tổ chức của một bàn phím

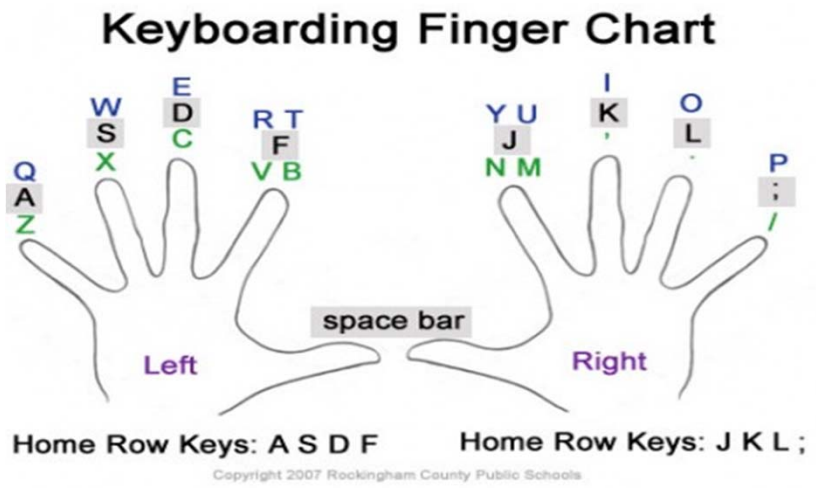
Phím nóng (hotkey): dùng để thực hiện một chức năng trong phần mềm một cách nhanh chóng



Bàn phím theo chuẩn QWERTY

Phím tổ hợp: dùng để kết hợp với một phím khác (không thuộc nhóm phím nóng) để tạo thành một phím nóng tổ hợp

Cách gõ bàn phím



- Cần luyện tập để có thể sử dụng 10 ngón tay để gõ bàn phím và hạn chế nhìn bàn phím. Tốc độ trung bình sau khi luyện tập vào khoảng 50 – 70 từ / phút (tiếng Anh).

Các bảng mã tiếng Việt

Viet	Unicode Hex	VNI	VNI Hex	VPS	VPS Hex	VISCII	VISCII Hex	TCVN3	TCVN3 Hex	VIQR	English Name
À	U+00C0	AØ	41 D8	€	80	À	C0	Aµ	41 B5	A`	LATIN CAPITAL LETTER A WITH GRAVE
Á	U+00C1	AÙ	41 D9	Á	C1	Á	C1	A¸	41 B8	A'	LATIN CAPITAL LETTER A WITH ACUTE
Â	U+00C2	AÂ	41 C2	Â	C2	Â	C2	¢	A2	A^	LATIN CAPITAL LETTER A WITH CIRCUMFLEX
Ã	U+00C3	AÕ	41 D5	‚	82	Ã	C3	A·	41 B7	A~	LATIN CAPITAL LETTER A WITH TILDE
È	U+00C8	EØ	45 D8	×	D7	È	C8	EÌ	45 CC	E`	LATIN CAPITAL LETTER E WITH GRAVE
É	U+00C9	EÙ	45 D9	É	C9	É	C9	EÐ	45 D0	E'	LATIN CAPITAL LETTER E WITH ACUTE
Ê	U+00CA	EÂ	45 C2	Ê	CA	Ê	CA	£	A3	E^	LATIN CAPITAL LETTER E WITH CIRCUMFLEX
Ì	U+00CC	Ì	CC	µ	B5	Ì	CC	I×	49 D7	I`	LATIN CAPITAL LETTER I WITH GRAVE
Í	U+00CD	Í	CD	´	B4	Í	CD	IÝ	49 DD	I'	LATIN CAPITAL LETTER I WITH ACUTE
Ò	U+00D2	OØ	4F D8	¼	BC	Ò	D2	Oß	4F DF	O`	LATIN CAPITAL LETTER O WITH GRAVE
Ó	U+00D3	OÙ	4F D9	¹	B9	Ó	D3	Oã	4F E3	O'	LATIN CAPITAL LETTER O WITH ACUTE
Ô	U+00D4	OÂ	4F C2	Ô	D4	Ô	D4	¤	A4	O^	LATIN CAPITAL LETTER O WITH CIRCUMFLEX
Õ	U+00D5	OÕ	4F D5	¾	BE		A0	Oâ	4F E2	O~	LATIN CAPITAL LETTER O WITH TILDE
Ù	U+00D9	UØ	55 D8	¨	A8	Ù	D9	Uï	55 EF	U`	LATIN CAPITAL LETTER U WITH GRAVE
Ú	U+00DA	UÙ	55 D9	Ú	DA	Ú	DA	Uó	55 F3	U'	LATIN CAPITAL LETTER U WITH ACUTE
Ý	U+00DD	YÙ	59 D9	Ý	DD	Ý	DD	Yý	59 FD	Y'	LATIN CAPITAL LETTER Y WITH ACUTE
à	U+00E0	aø	61 F8	à	E0	à	E0	µ	B5	a`	LATIN SMALL LETTER A WITH GRAVE
á	U+00E1	aù	61 F9	á	E1	á	E1	¸	B8	a'	LATIN SMALL LETTER A WITH ACUTE
â	U+00E2	aâ	61 E2	â	E2	â	E2	©	A9	a^	LATIN SMALL LETTER A WITH CIRCUMFLEX
ã	U+00E3	aõ	61 F5	ã	E3	ã	E3	·	B7	a~	LATIN SMALL LETTER A WITH TILDE
è	U+00E8	eø	65 F8	è	E8	è	E8	Ì	CC	e`	LATIN SMALL LETTER E WITH GRAVE

Các bảng mã thông dụng là Unicode, VNI, TCVN3, và VIQR.

Các bảng mã tiếng Việt

- Mỗi loại bảng mã có một cách nhập kí tự riêng từ bàn phím. Các từ tiếng Việt là một tổ hợp của 2 hay nhiều phím được nhập liên tục và nối tiếp nhau.
- Các loại mã sẽ hiển thị sai khi cách nhập mã từ bàn phím không tương thích với bảng mã.
- Giữa các loại bảng mã có thể chuyển đổi qua lại thông qua một phần mềm, ví dụ như Unikey Toolkit

Ý nghĩa định dạng font chữ

- Đặc trưng của font chữ: đậm hay đơn nét, nghiêng hay không nghiêng, có chân hay không có chân, có bóng hay không có bóng, chiều cao và độ rộng của một ký tự.
- Việc định dạng font chữ được dựa vào vị trí và vai trò của ký tự trong cấu trúc văn bản.

Phân loại font chữ dùng trong máy tính

- Font chữ sử dụng trong máy tính được chia làm 3 loại dựa trên cấu trúc xây dựng font chữ:

- True type:là loại font chữ được công ty máy tính Apple phát triển, có khả năng hiển thị sắc và rõ nét ở nhiều kích thước.
- Open type:là loại font chữ được công ty máy tính Microsoft phát triển dựa trên True Type và có độ sắc nét hơn True Type.

  Cả True Type và Open Type được xây dựng dưới dạng véc tơ.
- Screen font:là loại font chữ dùng để hiển thị trên màn hình điều khiển, và được xây dựng dưới dạng file ảnh.

Công cụ soạn thảo văn bản trên máy tính

- Soạn thảo dựa trên cú pháp của một trình biên dịch: Latex (ví dụ như sử dụng WinEdt để biên soạn và dùng Miktex để biên dịch)
- Soạn thảo dựa trên phần mềm ứng dụng trợ giúp theo tiếp cận "what you see is what you get" chạy trên máy tính hay trên web: MS-Office, OpenOffice, Google Docs, iWork (MAC)
- Soạn thảo dựa trên phần mềm chuyên dụng cho một vài loại file văn bản thông dụng: Adobe Acrobat Professional cho file PDF

Soạn thảo văn bản trên máy tính

- Các định dạng file trợ giúp soạn thảo văn bản: *.txt, *.pdf, *.tex, *.doc, *.rtf

  - ASCII, UTF-8 (plain text *.txt) : file văn bản không có cấu trúc định dạng đính kèm.

  - PDF (*.pdf): định dạng file văn bản của công ty Adobe System.

  - DOC (*.doc): định dạng file văn bản của công ty Microsoft cho các phần mềm Office.

Soạn thảo văn bản trên máy tính

- RTF (*.rtf): là định dạng file văn bản có hỗ trợ biên dịch như Tex được phát triển bởi công ty Microsoft.

- Các phần mềm trợ giúp soạn thảo văn bản: Notepad, Adobe Acrobat, Latex, MS. Word, v.v…
- Các bộ gõ tiếng Việt: Unikey, Vietkey, Winvnkey.
- Chuyển đổi bảng mã

Định dạng văn bản dựa trên các mẫu văn bản

Các mẫu văn bản hành chính

Tên cơ quan ban hành văn bản

Quốc hiệu của nước Việt Nam

Tiêu đề

Tên bộ phận tiếp nhận / giải quyết đơn

Tên người viết đơn

Nội dung đơn

Ngày tháng nộp đơn

Chữ ký người nộp đơn

Xác nhận cơ quan đồng ý tiếp nhận đơn

Ý kiến của người trực tiếp giải quyết đơn

Trường Đại học Khoa học Tự Nhiên
Khoa Công nghệ Thông tin
🕮

Cộng hoà xã hội chủ nghĩa Việt Nam
Độc lập - Tự do – Hạnh phúc
---oOo---

ĐƠN XIN CỨU XÉT ĐIỂM THI

Kính gởi: Phòng Đào tạo
Khoa Công nghệ Thông tin
Trường Đại học Khoa học Tự nhiên

Em tên là: ..... Mã số sinh viên: .....

Nay em làm đơn xin được cứu xét điểm thi môn: .....

Mã lớp lý thuyết đăng ký học phần: TH...../..... GVLT: .....

Phòng thi lý thuyết: ..... Ngày thi: .....

Phòng thi thực hành: ..... Ngày thi: .....

Kết quả đã công bố:

1). Lớp: TH...../.....: Điểm lý thuyết: ..... Điểm thực hành: .....
Điểm khác: ..... Tổng điểm: .....

2). Lớp: TH...../.....: Điểm lý thuyết: ..... Điểm thực hành: .....
Điểm khác: ..... Tổng điểm: .....

3). Lớp: TH...../.....: Điểm lý thuyết: ..... Điểm thực hành: .....
Điểm khác: ..... Tổng điểm: .....

Lý do: .....

.....

Em xin chân thành biết ơn.

Tp.HCM, ngày tháng năm 2006

Người làm đơn

XÁC NHẬN CỦA GIÁO VỤ KHOA

LT: TH: Khác: Tổng:

Tp.HCM, ngày tháng năm 2006

Ý KIẾN CỦA GIÁO VIÊN LÝ THUYẾT

Các định dạng văn bản được ban hành của nhà nước

- Thông tư liên tịch số 55 của Bộ Nội Vụ và VP Chính phủ về soạn thảo văn bản: http://www.luutruvn.gov.vn/content/law/Pages/View.aspx?CategoriesID=4&DocumentID=492
- Thông tư số 01/2011/TT-BNV ngày 19 tháng 01 năm 2011 của Bộ Nội vụ Hướng dẫn thể thức và kỹ thuật trình bày văn bản hành chính: http://www.luutruvn.gov.vn/content/law/Pages/View.aspx?CategoriesID=4&DocumentID=937

Mẫu luận văn của khoa CNTT

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

BỘ MÔN KHOA HỌC MÁY TÍNH

Trần Văn X

CÁC THUẬT TOÁN ỨNG DỤNG CHO VIỆC ĐIỀU HƯỚNG ROBOT BẰNG CAMERA ĐA HƯỚNG

KHÓA LUẬN TỐT NGHIỆP CỬ NHÂN CNTT

TP. HCM, NĂM 2011

Mục lục

Lời cảm ơn i

Lời nói đầu ii

Chương 1 Giới thiệu 1

1.1 Động lực và mục tiêu nghiên cứu 2

1.2 Phát biểu bài toán 6

1.3 Tóm tắt các chương 9

Chương 2 Ứng dụng và các nghiên cứu liên quan 12

2.1 Ứng dụng 13

2.2 Các nghiên cứu liên quan 17

Chương 3 Lý Thuyết Cơ Bản 21

3.1 Giới thiệu 22

3.2 Thuật toán tối ưu hóa Gauss–Newton 23

3.3 Thuật toán theo dõi đặc trưng Kanade–Lucas–Tomasi (KLT Tracker) 25

3.4 Thuật toán đồng thuận mẫu ngẫu nhiên (RANSAC) 27

3.5 Thuật toán lọc Kalman (Kalman Filter) 30

Chương 4 Camera Đa Hướng 35

4.1 Giới thiệu sơ lược về camera đa hướng 36

4.2 Xác định thông số cho camera đa hướng 49

4.3 Các phương pháp biến đổi ảnh đa hướng 53

Chương 5 Tái tạo cấu trúc ba chiều bằng camera đa hướng đôi 56

5.1 Phát biểu bài toán 57

5.2 Ước lượng vị trí tương quan của hai camera 58

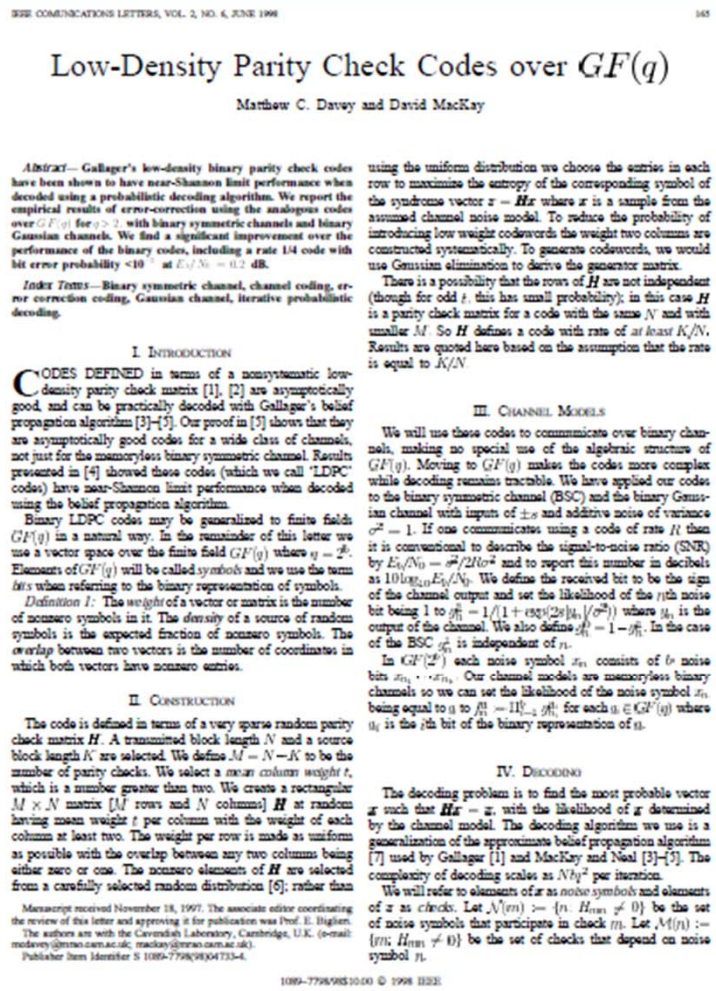
Hình 1-1: Góc nhìn ngang của mắt ruồi vào khoảng 360°. (Armando Frazão http://photo.digitalg.net/)

Camera đa hướng cũng có thể coi như là một dạng tương tự với cấu trúc hình học mắt lồi của các loài côn trùng. Nhờ vào góc nhìn rộng lên tới 360^0 mà camera đa hướng đã được ứng dụng rất nhiều vào các lĩnh vực điều hướng robot, hội nghị từ xa, giám sát và nhiếp ảnh.

Góc nhìn rộng mang lại nhiều lợi thế cho camera đa hướng tuy nhiên kèm theo là một số vấn đề trong xử lý. Cấu trúc hình học đặc biệt của gương khiến phương trình chiếu của loại camera này phức tạp hơn so với camera thường do đó việc xác định thông số cũng phức tạp hơn.Mục tiêu đầu tiên của đề tài này phải là tìm hiểu về camera đa hướng và tìm câu trả lời cho các vấn đề sau:

- Cấu tạo vật lý của camera đa hướng.
- Lợi thế của camera đa hướng so với camera thường.
- Các phương pháp biến đổi ảnh để làm hạn chế biến dạng trong camera đa hướng.
- Phương pháp xác định thông số cho camera đa hướng để ứng dụng trong phần sau của đề tài.
- Phương pháp giả lập camera đa hướng trong môi trường ba chiều.

Các mẫu bài báo khoa học



IEEE COMMUNICATIONS LETTERS, VOL. 2, NO. 6, JUNE 1998 165

Low-Density Parity Check Codes over $GF(q)$

Matthew C. Davey and David MacKay

Abstract— **Gallager's low-density binary parity check codes have been shown to have near-Shannon limit performance when decoded using a probabilistic decoding algorithm. We report the empirical results of error-correction using the analogous codes over $GF(q)$ for $q > 2$, with binary symmetric channels and binary Gaussian channels. We find a significant improvement over the performance of the binary codes, including a rate 1/4 code with bit error probability $<10^{-5}$ at $E_b/N_0 = 0.2$ dB.**

Index Terms—**Binary symmetric channel, channel coding, error correction coding, Gaussian channel, iterative probabilistic decoding.**

I. Introduction

CODES DEFINED in terms of a nonsystematic low-density parity check matrix [1], [2] are asymptotically good, and can be practically decoded with Gallager's belief propagation algorithm [3]–[5]. Our proof in [5] shows that they are asymptotically good codes for a wide class of channels, not just for the memoryless binary symmetric channel. Results presented in [4] showed these codes (which we call 'LDPC' codes) have near-Shannon limit performance when decoded using the belief propagation algorithm.

Binary LDPC codes may be generalized to finite fields $GF(q)$ in a natural way. In the remainder of this letter we use a vector space over the finite field $GF(q)$ where $q = 2^b$. Elements of $GF(q)$ will be called *symbols* and we use the term *bits* when referring to the binary representation of symbols.

Definition 1: The *weight* of a vector or matrix is the number of nonzero symbols in it. The *density* of a source of random symbols is the expected fraction of nonzero symbols. The *overlap* between two vectors is the number of coordinates in which both vectors have nonzero entries.

II. Construction

The code is defined in terms of a very sparse random parity check matrix $\boldsymbol{H}$. A transmitted block length N and a source block length K are selected. We define $M = N - K$ to be the number of parity checks. We select a *mean column weight* t, which is a number greater than two. We create a rectangular $M \times N$ matrix [M rows and N columns] $\boldsymbol{H}$ at random having mean weight t per column with the weight of each column at least two. The weight per row is made as uniform as possible with the overlap between any two columns being either zero or one. The nonzero elements of $\boldsymbol{H}$ are selected from a carefully selected random distribution [6]; rather than using the uniform distribution we choose the entries in each row to maximize the entropy of the corresponding symbol of the syndrome vector $\boldsymbol{z} = \boldsymbol{H}\boldsymbol{x}$ where $\boldsymbol{x}$ is a sample from the assumed channel noise model. To reduce the probability of introducing low weight codewords the weight two columns are constructed systematically. To generate codewords, we would use Gaussian elimination to derive the generator matrix.

There is a possibility that the rows of $\boldsymbol{H}$ are not independent (though for odd t, this has small probability); in this case $\boldsymbol{H}$ is a parity check matrix for a code with the same N and with smaller M. So $\boldsymbol{H}$ defines a code with rate of *at least* K/N. Results are quoted here based on the assumption that the rate is equal to K/N.

III. Channel Models

We will use these codes to communicate over binary channels, making no special use of the algebraic structure of $GF(q)$. Moving to $GF(q)$ makes the codes more complex while decoding remains tractable. We have applied our codes to the binary symmetric channel (BSC) and the binary Gaussian channel with inputs of $\pm s$ and additive noise of variance $\sigma^2 = 1$. If one communicates using a code of rate R then it is conventional to describe the signal-to-noise ratio (SNR) by $E_b/N_0 = s^2/2R\sigma^2$ and to report this number in decibels as $10\log_{10}E_b/N_0$. We define the received bit to be the sign of the channel output and set the likelihood of the nth noise bit being 1 to $p_1^n = 1/(1 + \exp(2s|y_n|/\sigma^2))$ where y_n is the output of the channel. We also define $p_0^n = 1 - p_1^n$. In the case of the BSC p_1^n is independent of n.

In $GF(2^b)$ each noise symbol x_n consists of b noise bits $x_{n_1} \cdots x_{n_b}$. Our channel models are memoryless binary channels so we can set the likelihood of the noise symbol x_n being equal to $\mathfrak{g}$ to $p_{\mathfrak{g}}^n = \Pi_{j=1}^b p_{\mathfrak{g}_j}^{n_j}$ for each $\mathfrak{g} \in GF(q)$ where $\mathfrak{g}_j$ is the jth bit of the binary representation of $\mathfrak{g}$.

IV. Decoding

The decoding problem is to find the most probable vector $\boldsymbol{x}$ such that $\boldsymbol{H}\boldsymbol{x} = \boldsymbol{z}$, with the likelihood of $\boldsymbol{x}$ determined by the channel model. The decoding algorithm we use is a generalization of the approximate belief propagation algorithm [7] used by Gallager [1] and MacKay and Neal [3]–[5]. The complexity of decoding scales as Nbq^2 per iteration.

We will refer to elements of $\boldsymbol{x}$ as *noise symbols* and elements of $\boldsymbol{z}$ as *checks*. Let $\mathcal{N}(m) := \{n: H_{mn} \neq 0\}$ be the set of noise symbols that participate in check m. Let $\mathcal{M}(n) := \{m: H_{mn} \neq 0\}$ be the set of checks that depend on noise symbol n.

Manuscript received November 18, 1997. The associate editor coordinating the review of this letter and approving it for publication was Prof. E. Biglieri.

The authors are with the Cavendish Laboratory, Cambridge, U.K. (e-mail: mcdavey@mrao.cam.ac.uk; mackay@mrao.cam.ac.uk).

Publisher Item Identifier S 1089-7798(98)04733-4.

1089–7798/98$10.00 © 1998 IEEE

Hình thức 2 cột

Adaptive Probabilistic Visual Tracking with Incremental Subspace Update

David Ross[1], Jongwoo Lim[2], and Ming-Hsuan Yang[3]

[1] University of Toronto, Toronto, ON M5S 3G4, Canada
[2] University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA
[3] Honda Research Institute, Mountain View, CA 94041, USA
dross@cs.toronto.edu jlim1@uiuc.edu myang@honda-ri.com

Abstract. Visual tracking, in essence, deals with non-stationary data streams that change over time. While most existing algorithms are able to track objects well in controlled environments, they usually fail if there is a significant change in object appearance or surrounding illumination. The reason being that these visual tracking algorithms operate on the premise that the models of the objects being tracked are invariant to internal appearance change or external variation such as lighting or viewpoint. Consequently most tracking algorithms do not update the models once they are built or learned at the outset. In this paper, we present an adaptive probabilistic tracking algorithm that updates the models using an incremental update of eigenbasis. To track objects in two views, we use an effective probabilistic method for sampling affine motion parameters with priors and predicting its location with a maximum *a posteriori* estimate. Borne out by experiments, we demonstrate the proposed method is able to track objects well under large lighting, pose and scale variation with close to real-time performance.

1 Introduction

Visual tracking essentially deals with non-stationary data, both the object and the background, that change over time. Most existing algorithms are able to track objects, either previously viewed or not, in a short span of time and in a well controlled environment. However these algorithms usually fail to observe the object motion or have significant drifts after some period of time, either due to the drastic change of the object appearance or large lighting variation in the surroundings. Although such situations can be ameliorated with recourse to view-based appearance models [1] [2], adaptive color-based trackers [3] [4], contour-based trackers [5] [4], particle filters [5], 3D model based methods [6], optimization methods [1] [7], and background modeling [8], most algorithms typically operate on the premise that the target object models do not change drastically over time. Consequently these algorithms build or learn models of the objects first and then use them for tracking, without adapting the models to account for changes of the appearance of the object, e.g., large variation of pose or facial expression, or the surroundings, e.g., lighting variation. Such an approach, in our view, is prone to performance instability and needs to be addressed for building a robust tracker.

Hình thức 1 cột