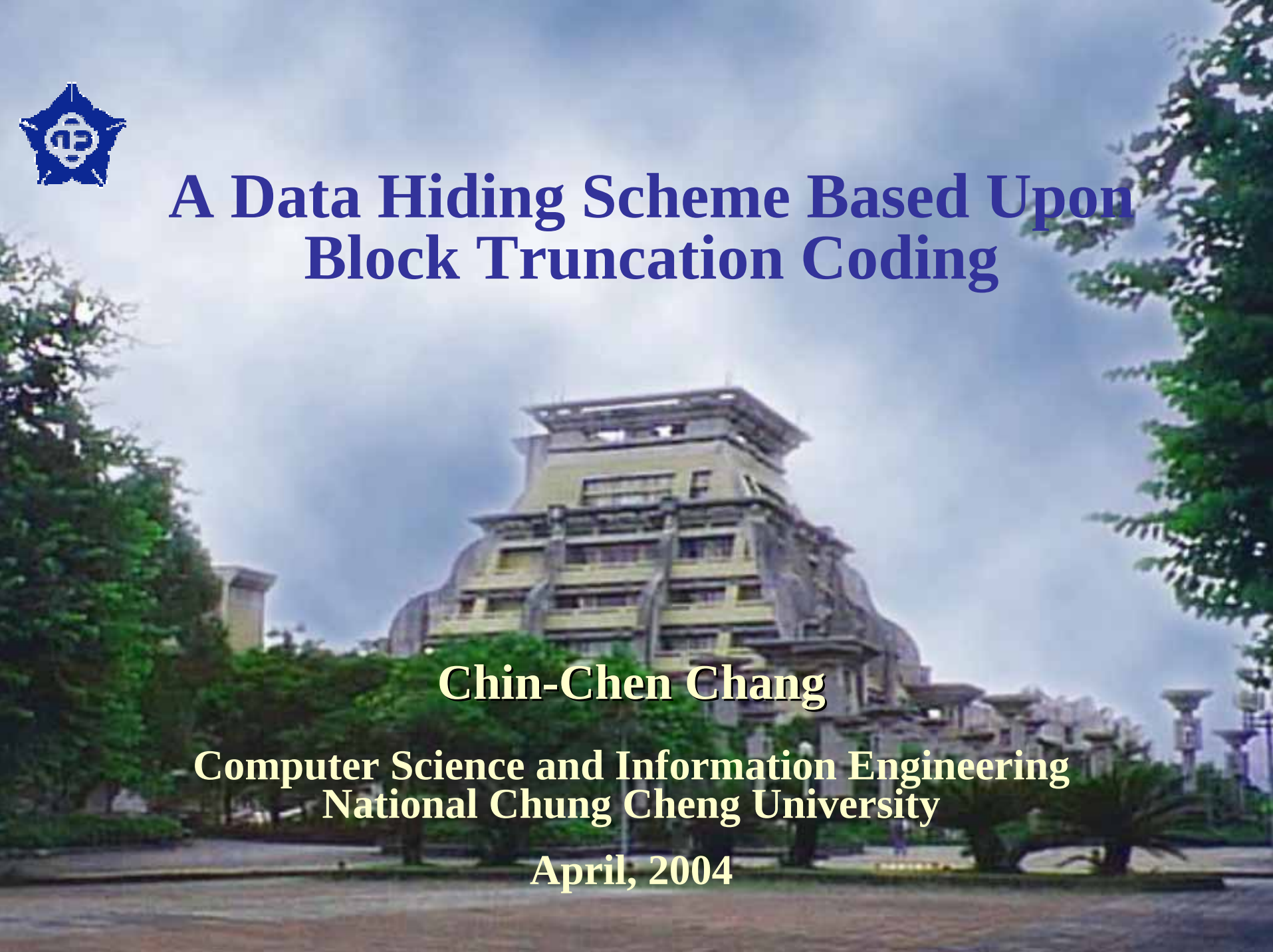


A Data Hiding Scheme Based Upon Block Truncation Coding

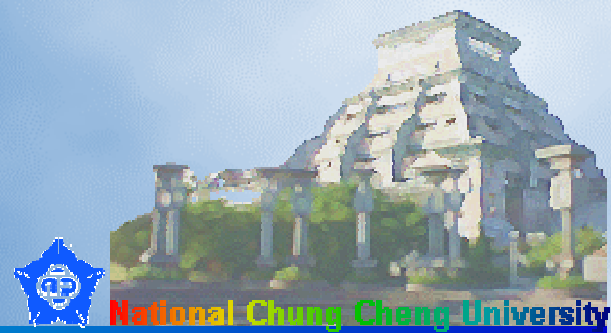
Chin-Chen Chang

Computer Science and Information Engineering
National Chung Cheng University

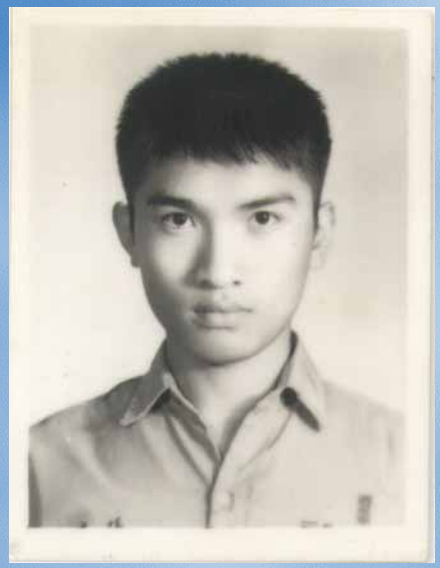
April, 2004

Outline

- Introduction to data hiding
- Review some data hiding schemes
  - LSB-Based data hiding scheme
  - VQ-Based data hiding scheme
- BTC-based oriented data hiding scheme
- Experiments and conclusions



Introduction

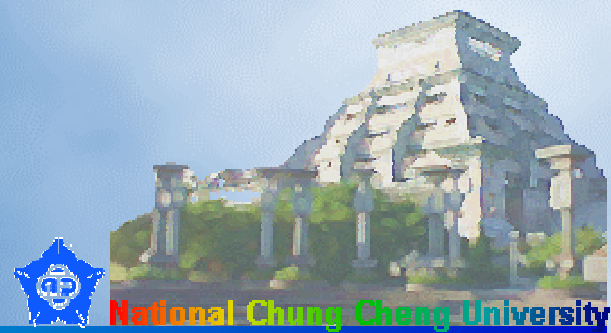


C.C. Chang

Extract hidden data

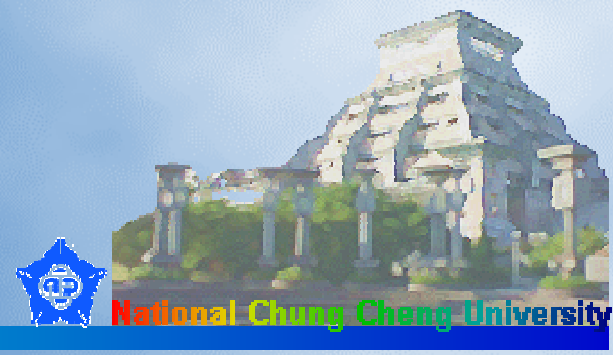
Prof. Chin-Chen Chang

National Chung Cheng University



Cover Carriers

- Image
- Video
- Sound
- Text



Requirements

- **Imperceptibility**

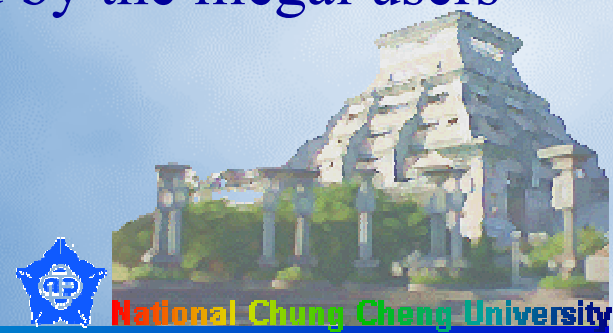
  Embedding image should not appear any artificial effect

- **Capacity**

  Maximize the embedding payloads

- **Security**

  Secret messages can not be extracted by the illegal users



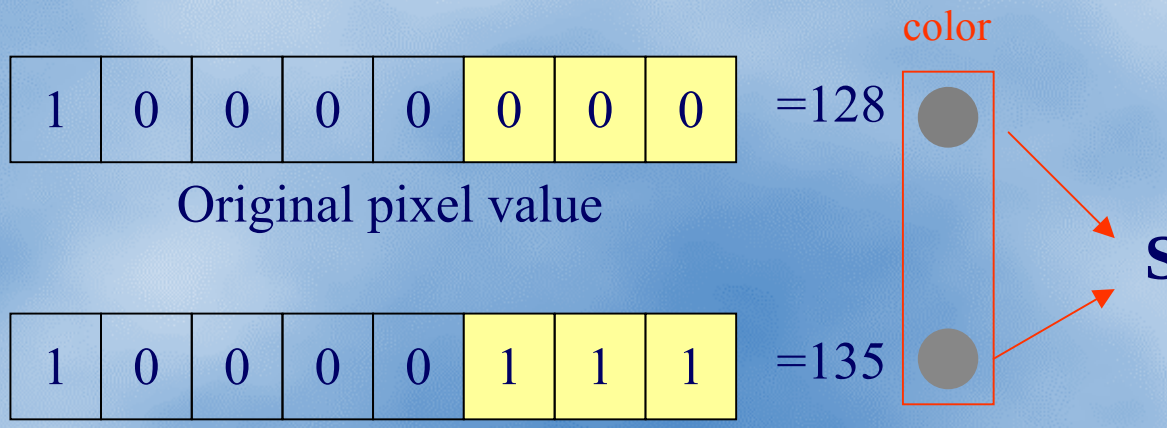
LSB-Based Data Hiding Scheme

MSB: Most significant bit

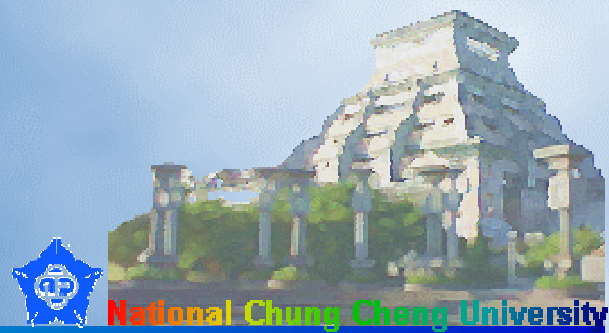
LSB: Least significant bit

8 bits/pixel

Example:



VQ-Based Data Hiding Scheme



Vector Quantization (VQ)

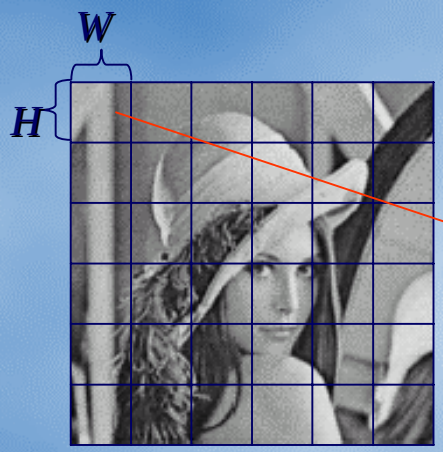
- VQ is a block-based lossy image compression method
- Definition of vector quantization (VQ):

$$Q : R^k \rightarrow Y$$

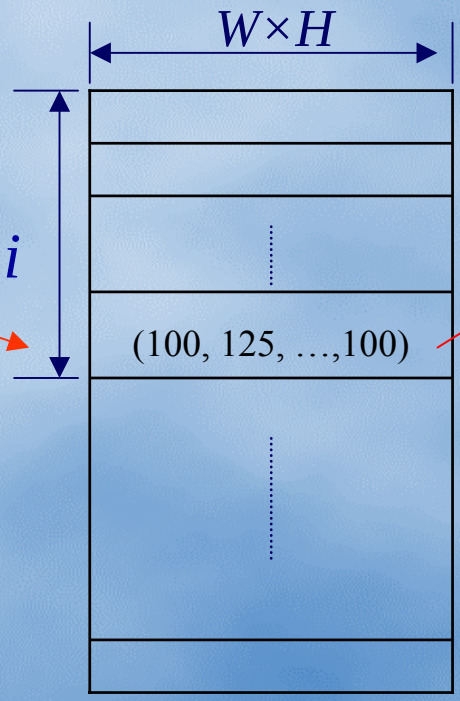
, where Y is a finite subset of R^k.

- VQ is composed of the following three parts:
  - Codebook generation process,
  - Encoding process, and
  - Decoding process.

Standard VQ Compression



Original image

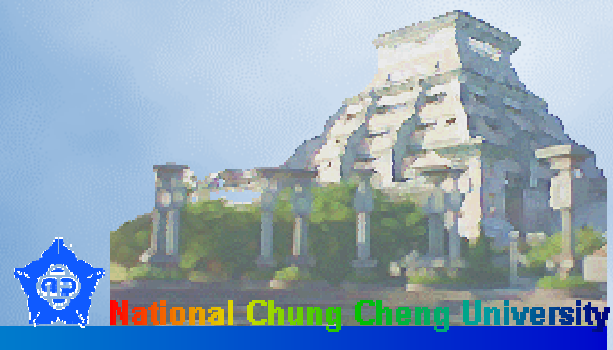


Codebook

index *i*

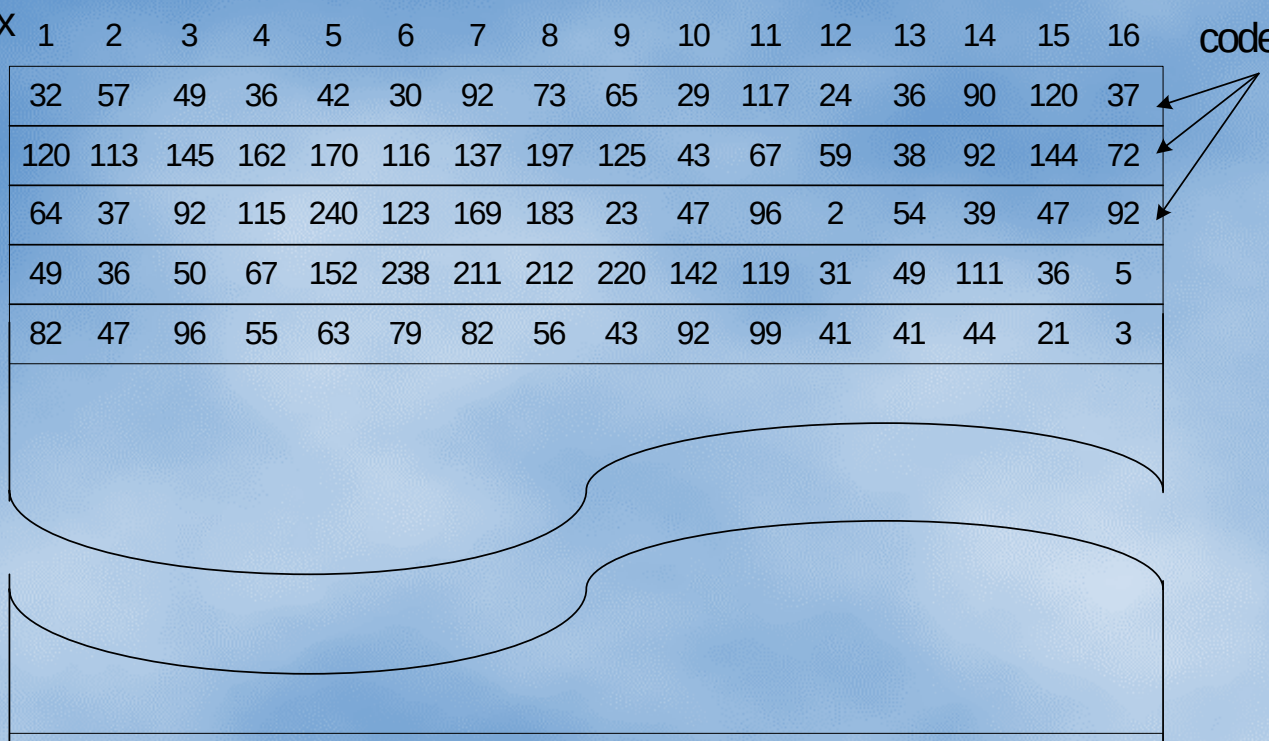
i	31	19	254	12	13
1	74	41	67	24	123
7	2	60	24	56	85
45	23	15	85	150	14
45	12	71	83	200	66

Index table

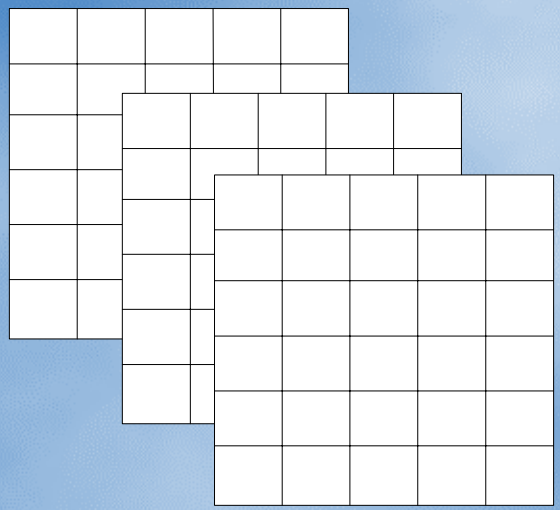


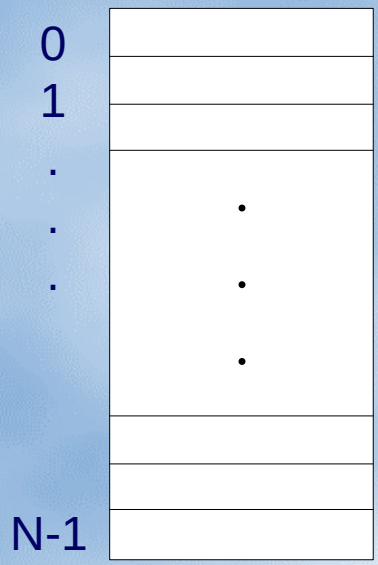
CODEBOOK

Index	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0	32	57	49	36	42	30	92	73	65	29	117	24	36	90	120	37
1	120	113	145	162	170	116	137	197	125	43	67	59	38	92	144	72
2	64	37	92	115	240	123	169	183	23	47	96	2	54	39	47	92
3	49	36	50	67	152	238	211	212	220	142	119	31	49	111	36	5
4	82	47	96	55	63	79	82	56	43	92	99	41	41	44	21	3
247	43	120	39	56	78	211	230	120	151	164	119	211	234	212	34	96
248	122	163	111	57	2	113	127	49	32	56	93	28	49	72	88	59
249	150	145	120	59	123	27	93	49	77	82	210	220	149	33	56	57
250	18	45	43	72	59	366	44	72	83	43	81	9	52	55	9	10
251	43	6	54	2	36	44	72	83	92	120	113	49	87	56	31	123
252	92	91	124	210	230	251	32	43	99	85	96	43	72	90	66	43
253	73	33	44	92	90	85	79	36	49	37	58	72	63	77	65	42
254	2	44	32	59	250	237	46	124	150	170	120	92	79	83	65	72
255	9	57	92	99	81	124	130	192	177	18	123	41	82	42	96	33



VQ Codebook Training



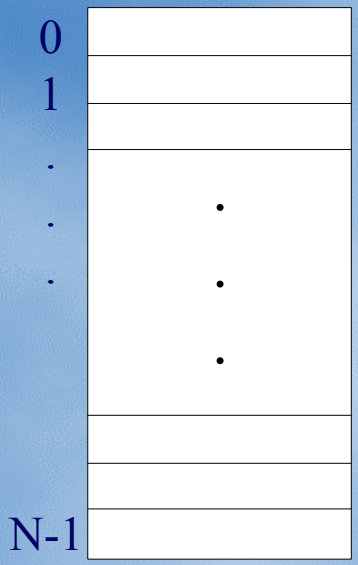


Training Images

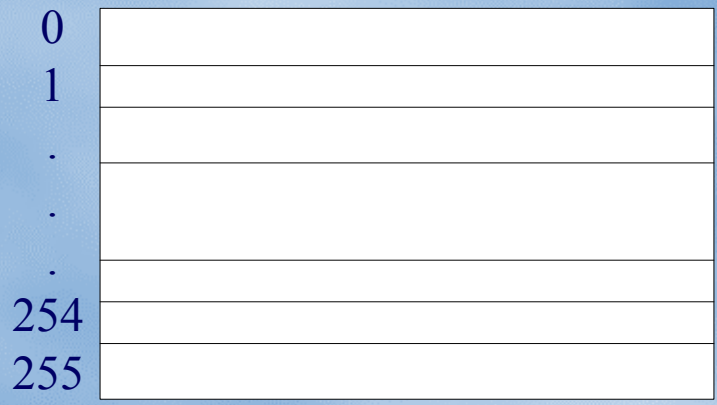
Training set

Separating the image to vectors

VQ Codebook Training

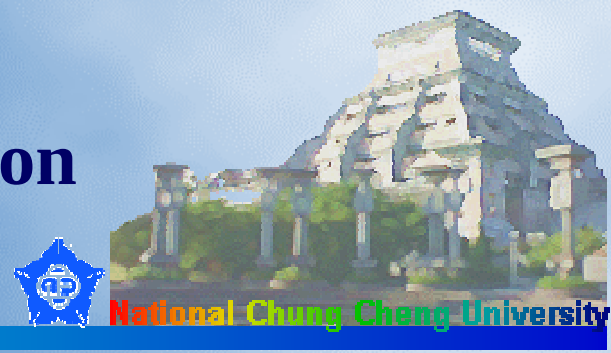


Training set

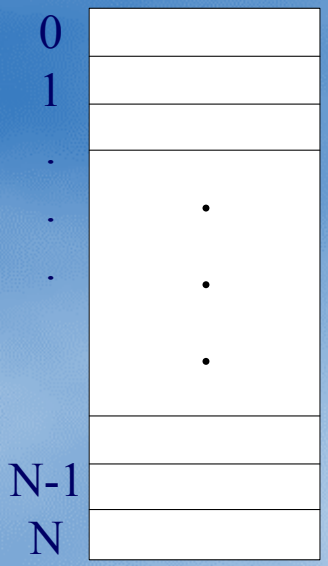


Initial codebook

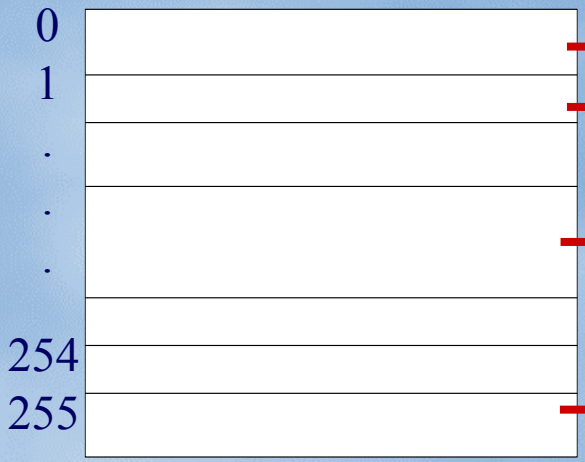
Codebook initiation



Vector Quantization (VQ) Codebook Training



Training set



Codebook C_i

Index sets

(1, 2, 5, 9, 45, …)

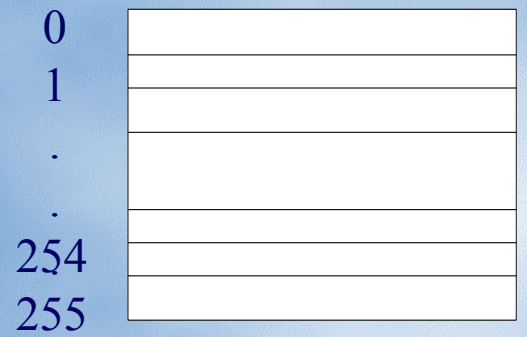
(101, 179, 201, …)

(8, 27, 38, 19, 200, …)

(23, 0, 67, 198, 224, …)

Compute mean values

Replace the old vectors



New Codebook C_{i+1}

Training using iteration algorithm

Example

cw_1	(184,	192,	193,	197)
cw_2	(34,	50,	43,	47)
cw_3	(191,	198,	190,	188)
cw_4	(77,	83,	84,	65)
cw_5	(63,	70,	94,	98)
cw_6	(23,	29,	16,	19)
cw_7	(210,	213,	192,	230)
cw_8	(151,	153,	169,	171)

Codebook

To encode an input vector, for example, v = (150,145,121,130)

(1) Compute the distance between v with all vectors in codebook

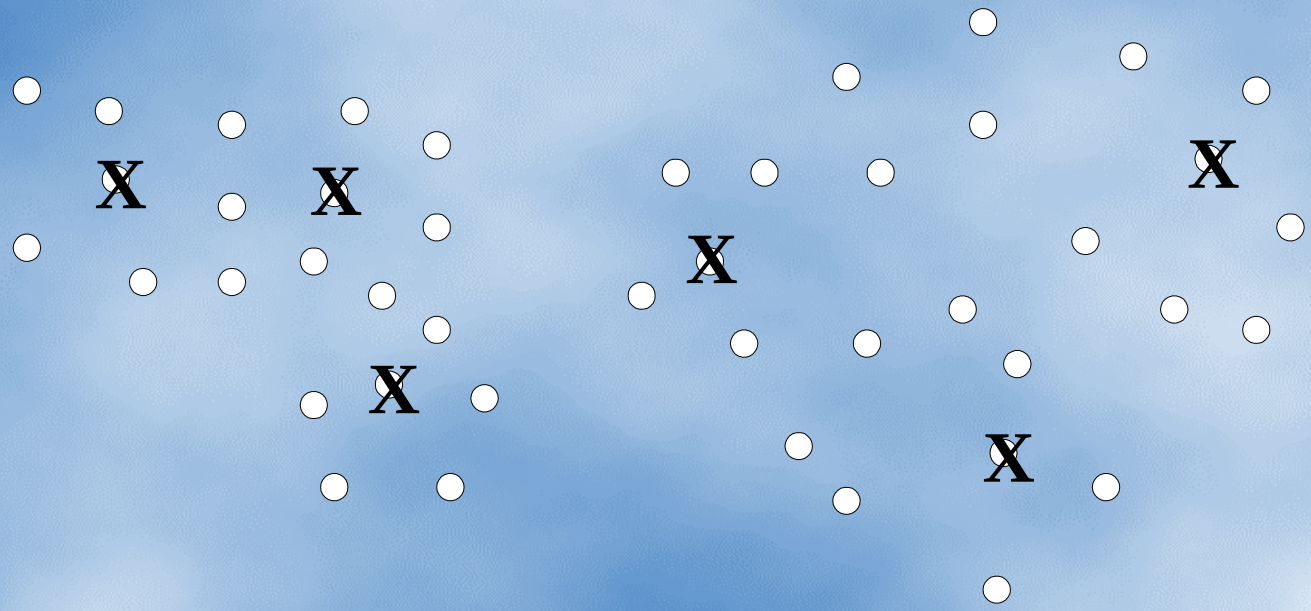
$d(v, cw_1) = 114.2$ $d(v, cw_2) = 188.3$ $d(v, cw_3) = 112.3$

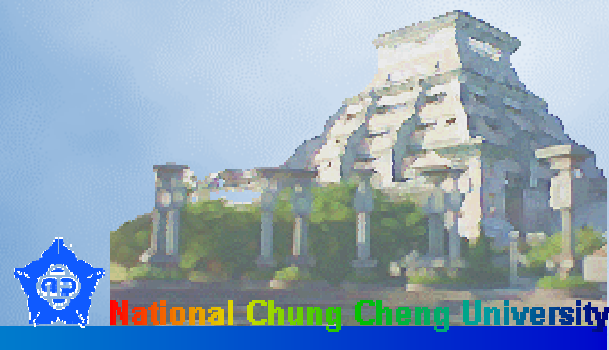
$d(v, cw_4) = 124.6$ $d(v, cw_5) = 122.3$ $d(v, cw_6) = 235.1$

$d(v, cw_7) = 152.5$ $d(v, cw_8) = 63.2$

(2) So, we choose cw_8 to replace the input vector v.

LBG Algorithm





Full Search (FS)

For example: Let the codebook size be 256

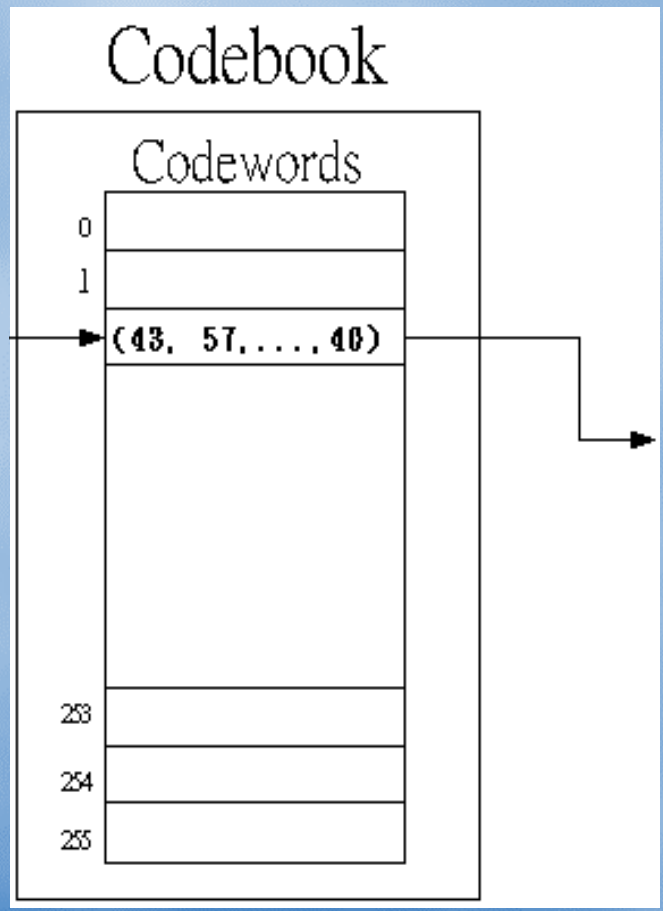


Image vector

# of searched nodes per image vector = 256

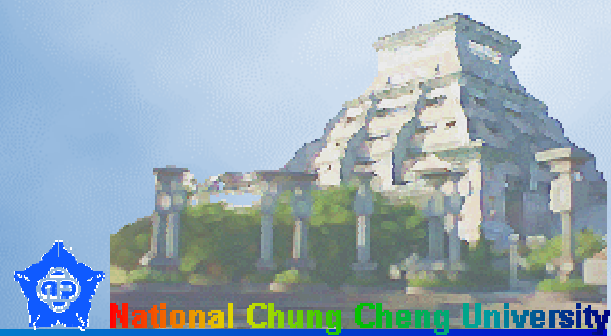
- **Time consuming**
- **The closest codeword**

$$C = \{c_1, c_2, \ldots, c_{nc}\}$$

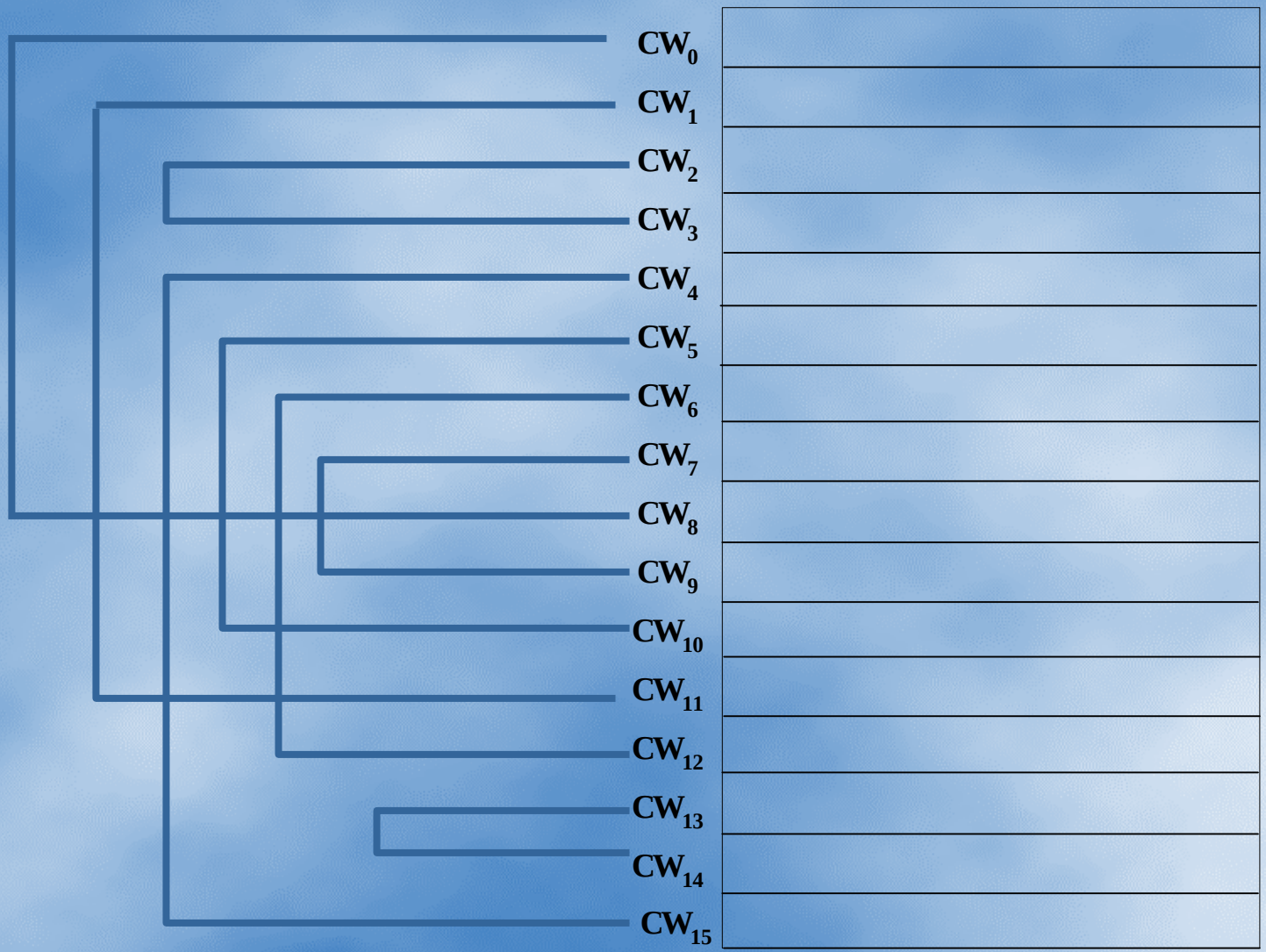
Euclidean Distance

- The dimensionality of vector = k (= w*h)
- An input vector $x = (x_1, x_2, \ldots, x_k)$
- A codeword $y_i = (y_{i1}, y_{i2}, \ldots, y_{ik})$
- The Euclidean distance between x and y_i

$$d(x, y_i) = \|x - y_i\|^2 = \sum_{j=1}^{k} (x_j - y_{ij})^2$$



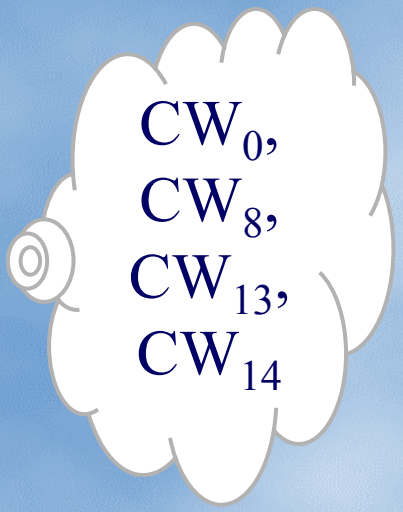
CODEBOOK



To find the closest pairs

$d(CW_0, CW_8) > TH$

$d(CW_{13}, CW_{14}) > TH$



Unused

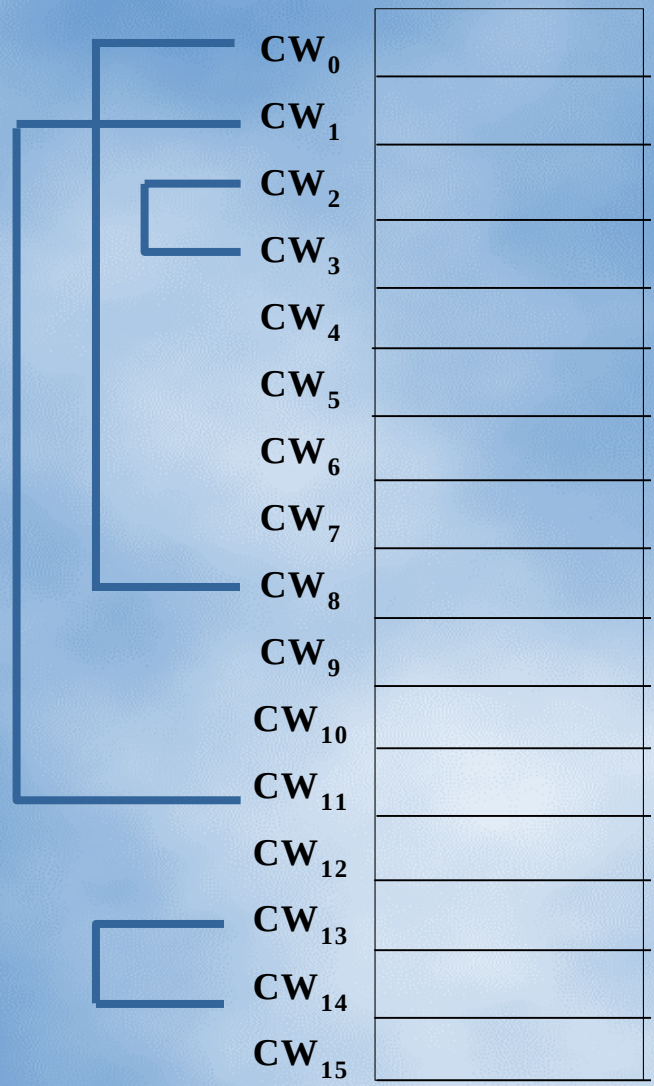
CW_1,CW_2
CW_4, CW_5
CW_6, CW_7

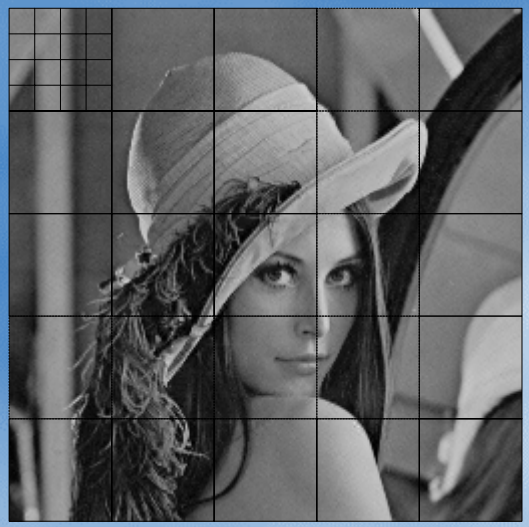
1

CW_{11},CW_3

CW_{15}, CW_{10}
CW_{12}, CW_9

0





Original Image

Encode

2	1	3	13	4
15	0	2	12	9
8	7	4	5	14
12	13	6	9	8
8	7	0	2	2

Index Table

CW_0, CW_8, CW_{13}, CW_{14}

Unused

2	1	3	13	4
15	0	2	12	9
8	7	4	5	14
12	13	6	9	8
8	7	0	2	2

Index Table

Watermark: 1 0 1 0 1 0 0 1 0 1 1 1 1 0 0

2	1	3	13	4
15	0	2	12	9
8	7	4	5	14
12	13	6	9	8
8	7	0	2	2

Index Table

1	0	1		0
1		0	0	1
	0	1	1	
1		1	0	
	0			

Watermark

CW_1, CW_2, CW_4, CW_5 CW_6, CW_7

CW_{11}, CW_3 CW_{15}, CW_{10} CW_{12}, CW_9

2				

1 **0**

Watermark: 1 0 1 0 1 0 0 1 0 1 1 1 1 0 0

2	1	3	13	4
15	0	2	12	9
8	7	4	5	14
12	13	6	9	8
8	7	0	2	2

Index Table

1	0	1		0
1		0	0	1
	0	1	1	
1		1	0	
	0			

Watermark

CW_1, CW_2, CW_4, CW_5, CW_6, CW_7

1

CW_{11}, CW_3, CW_{15}, CW_{10}, CW_{12}, CW_9

0

2	11			

Watermark: 1 0 1 0 1 0 0 1 0 1 1 1 1 0 0

2	1	3	13	4
15	0	2	12	9
8	7	4	5	14
12	13	6	9	8
8	7	0	2	2

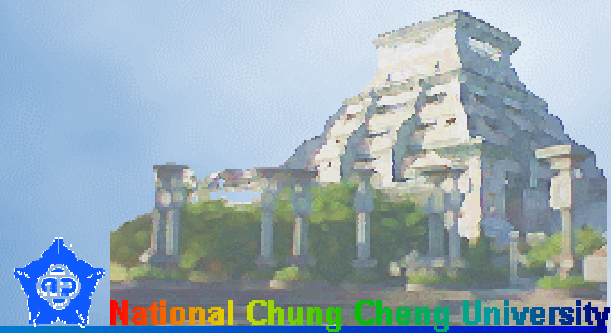
Index Table

1	0	1		0
1		0	0	1
	0	1	1	
1		1	0	
	0			

Watermark

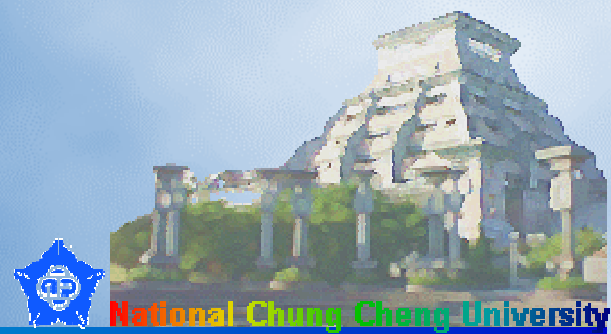
2	11	2	13	15
4	0	3	12	7
8	9	4	5	14
6	13	6	9	8
8	9	0	2	2

BTC Based Data Hiding Scheme

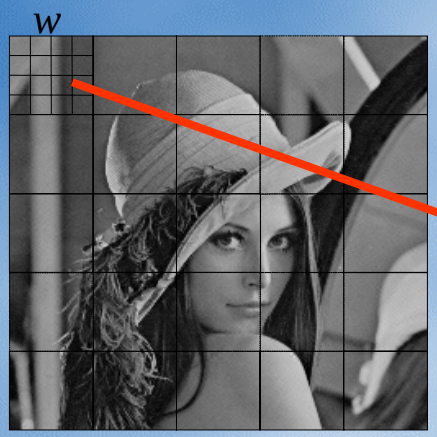


Block Truncation Coding (BTC)

- BTC is a block-based lossy image compression method for gray-level images
- The output data of BTC for an image block contains one bitmap and two quantization levels



An Example of BTC Encoding



Original Image

4×4 image pixels

140	142	144	145
146	141	146	142
145	141	144	142
142	138	141	144

Mean value X=142.5

Variance value ρ=2.199

Bitmap

0	0	1	1
1	0	1	0
1	0	1	0
0	0	0	1

# of 0 is 9

# of 1 is 7, q=7

$$X_L = X - \sigma\sqrt{\frac{q}{m-q}} = 142.5 - 2.199 * \sqrt{\frac{7}{9}} = 141$$

$$X_H = X + \sigma\sqrt{\frac{m-q}{q}} = 142.5 + 2.199 * \sqrt{\frac{9}{7}} = 145$$

Two quantization levels

An Example of BTC Decoding

Bitmap

0	0	1	1
1	0	1	0
1	0	1	0
0	0	0	1

$X_L = 141$ $X_H = 145$

Decoding

$$\hat{o}_i = \begin{cases} X_L, & b_i = 0, \\ X_{H,} & b_i = 1. \end{cases}$$

141	141	145	145
145	141	145	141
145	141	145	141
141	141	141	145

Reconstructed pixels

140	142	144	145
146	141	146	142
145	141	144	142
142	138	141	144

Original pixels

−

141	141	145	145
145	141	145	141
145	141	145	141
141	141	141	145

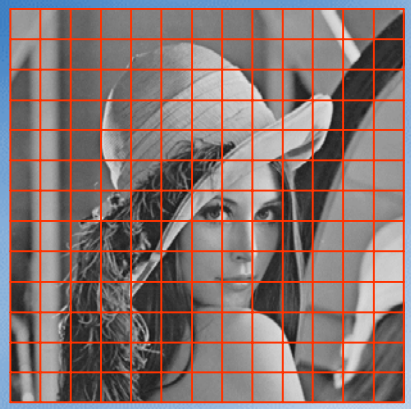
Processed pixels

=

-1	1	-1	0
1	0	1	1
0	0	-1	1
1	-3	0	-1

Difference

Our Scheme



Block division (4×4 pixels)

BTC Compression

$$< X_{L_i}, X_{H_i}, Bitmap_i >$$

Is $\left| X_{L_i} - X_{H_i} \right| > T$?

YES

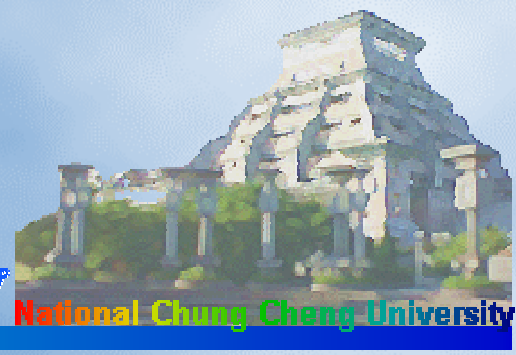
No

Skip

Complex block

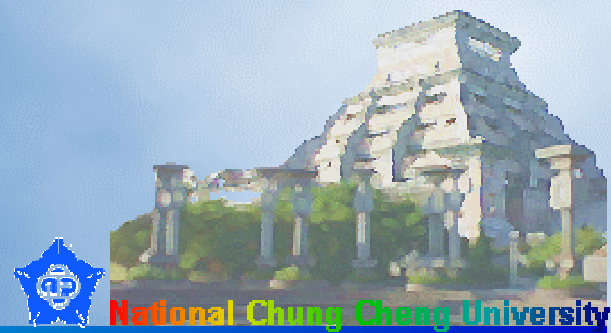
Embedding

Smooth block



Why we embed secret into the smooth block?

- Pixels in the smooth block are close to their neighboring pixels, even though we change the bits in the bitmap, its reconstructed pixel value is still close to its original one



Our Scheme

Example

Bitmap

0	0	1	1
1	0	1	0
1	0	1	0
0	0	0	1

$X_L = 141$ $X_H = 145$

T=10 (Threshold)
Secret bits=1000 0101

Smooth block

$|141-145| < T = 10$

1	**0**	**0**	**0**
0	**1**	**0**	**1**
1	0	1	0
0	0	0	1

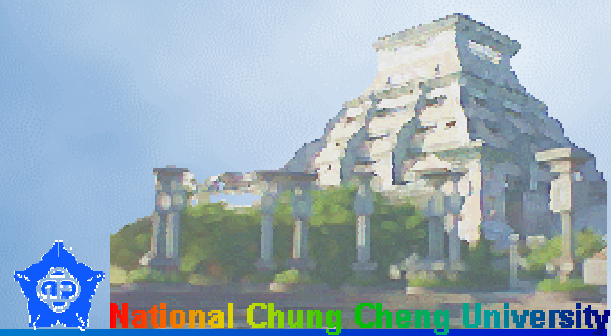
$$\hat{o}_i = \begin{cases} X_L, & b_i = 0, \\ X_H, & b_i = 1. \end{cases}$$

145	141	141	141
141	145	141	145
145	141	145	141
141	141	141	145

Stego pixels

Our Scheme

- A larger T gets high embedding capacity, but lower image quality of the stego image and vice versa
- T is set to be 0, then no secret bits can be hidden
- T is set to be ∞, then all of the bitmap in each block will be used to embed secret, and that is the maximum embedding capacity

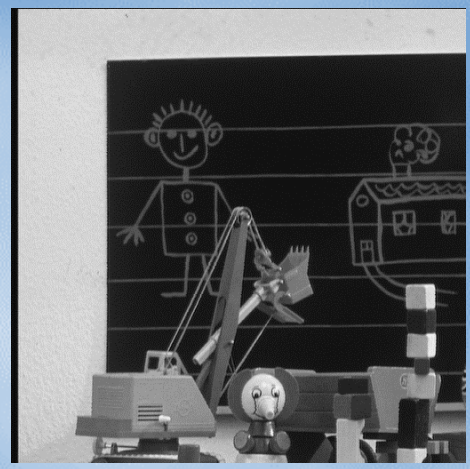


Experiments

(a) Lena

(b) Peppers

(c) F-16



(d) Toys

Four test images

Experiments

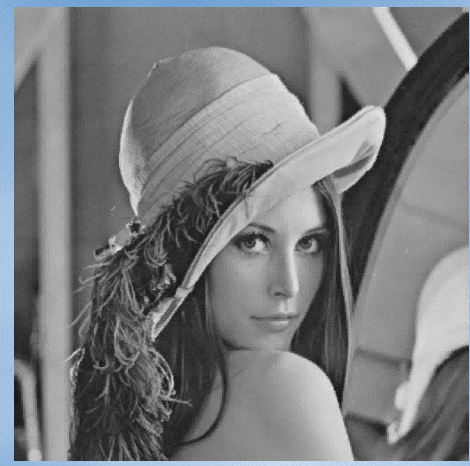
Resulting of embedding capacity and image quality (PSNR)

		Lena	Peppers	F-16	Toys
Embedding capacity (Kbytes)	BTC	64.00	64.00	64.00	64.00
	T=10	19.50	20.93	20.55	21.66
	T=15	23.22	24.74	23.16	23.37
	T=20	25.38	26.58	24.84	24.25
	T=∞	32.00	32.00	32.00	32.00
PSNR (dB)	BTC	33.11	33.45	32.69	32.68
	T=10	32.16	32.19	32.01	31.80
	T=15	31.40	31.45	31.48	31.48
	T=20	30.67	30.81	30.88	31.16
	T=∞	24.15	24.11	22.49	21.27

Experiments

(a) Original image

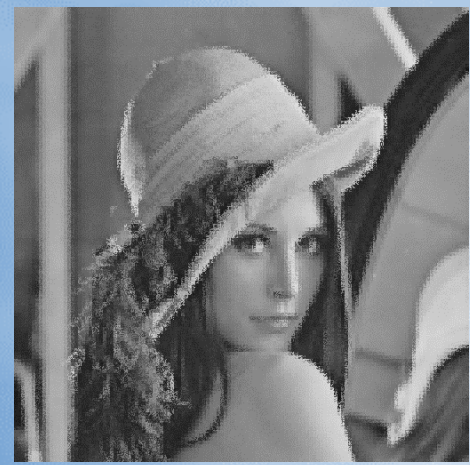
(b) BTC compression



(c) T=10

(d) T=15

(e) T=20



(f) T=∞

Experiments

(a) Original image

(b) BTC compression

(c) T=10

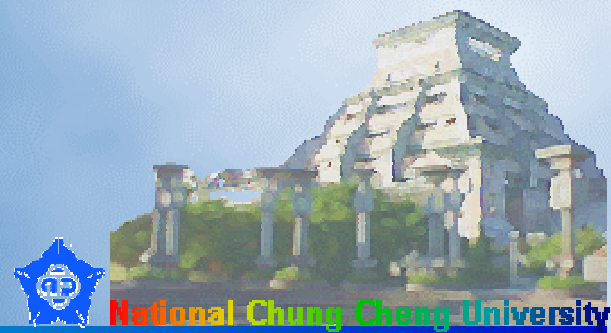
(d) T=15

(e) T=20

(f) T=∞

Conclusions

- An efficient and secure data hiding approach based upon BTC
- Data hiding in compression domain
  - VQ
  - JPEG or JPEG2000
  - GIF



Thank you !