IV.2 Video Transcoding

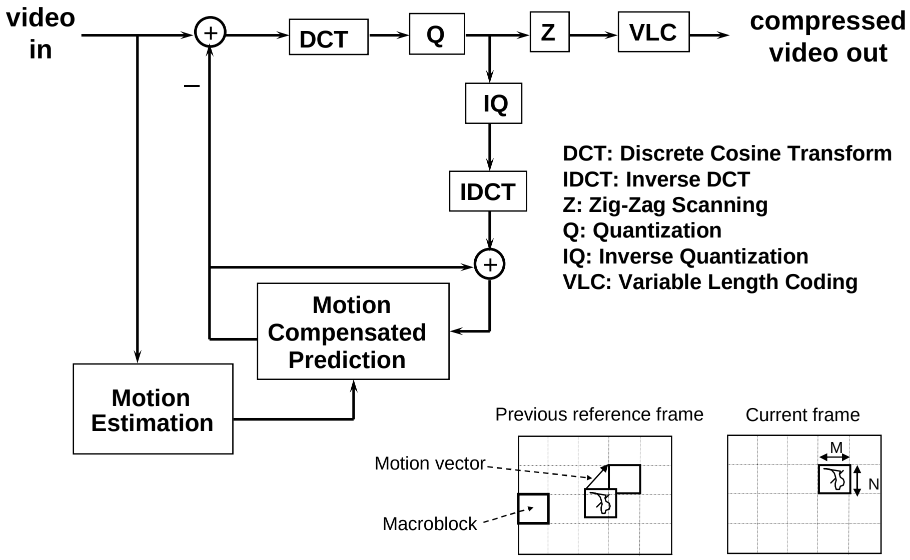
Prof. Chia-Wen Lin (林嘉文)
Department of CSIE
National Chung Cheng University
886-5-272-0411 ext. 33120
http://www.cs.ccu.edu.tw/~cwlin/
cwlin@cs.ccu.edu.tw

Presentation Outline

- Background
- Overview of Video Transcoding
  - Application Examples
  - Pixel-Domain Transcoder
  - DCT-Domain Transcoder
  - Fast Video Transcoding Architectures
- CCL’s Strength in Video Transcoding
- Discussions

Background

Standard Video Encoder

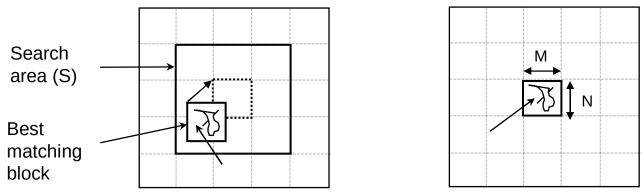


DCT: Discrete Cosine Transform
IDCT: Inverse DCT
Z: Zig-Zag Scanning
Q: Quantization
IQ: Inverse Quantization
VLC: Variable Length Coding

Motion Estimation

Previous reference frame

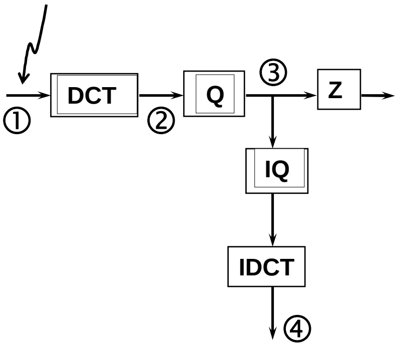
Current frame



- Usually, based on Sum of Absolute Difference (SAD)

DCT + Quantization

Motion compensated difference block



①

DCT

②

994	-34	20	38	22	0	5	-1
-49	17	8	-17	-4	-1	4	-2
149	35	-37	-15	1	3	-1	-4
58	8	-32	-3	-9	-11	2	-4
-27	5	26	6	9	7	8	0
9	41	53	8	0	1	-1	2
0	2	-1	-6	-2	-3	0	1
3	-14	-12	2	-1	-7	-5	-1

QUANTIZATION
(Step size = 30)

③

33	-1	1	1	1	0	0	0
-2	1	0	-1	0	0	0	0
5	1	-1	0	0	0	0	0
2	0	-1	0	0	0	0	0
-1	0	1	0	0	0	0	0
0	1	2	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

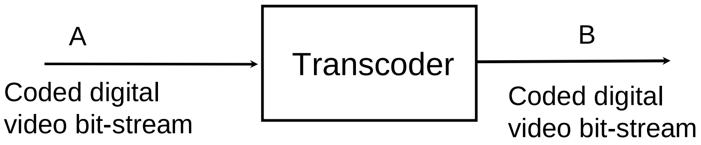
IQ, IDCT

④

Mean Square Error
= 38.75

Overview of Video Transcoding

Introduction: Video Transcoding

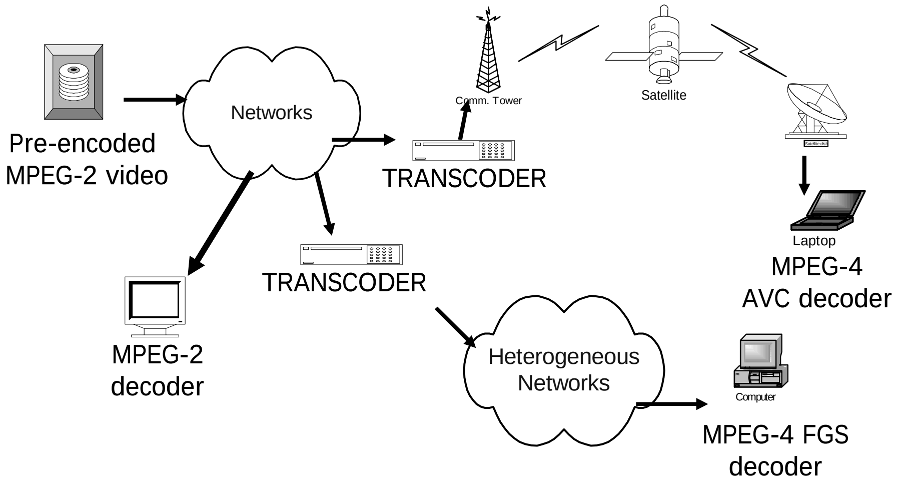


- Bit-rate/frame-rate/resolution conversion
- Format conversion
- Content manipulation

Transcoder Applications

- Bit-rate/frame-rate/resolution adaptation
  - Video transport over heterogeneous networks
  - Multi-point video Conferencing
  - Statistical multiplexing ...
- Format conversion
  - Multi-standard terminal, HDTV to SDTV, MPEG-2 to H.264, MPEG-2 to MPEG-4 FGS, ...
- Content manipulation
  - Watermark/Logo insertion
  - Error resilience feature insertion
  - Editing/splicing …

Application Example: Video Deliveryt over Heterogeneous Networks



Pre-encoded MPEG-2 video

Networks

TRANSCODER

Comm. Tower

Satellite

Laptop

MPEG-4 AVC decoder

MPEG-2 decoder

TRANSCODER

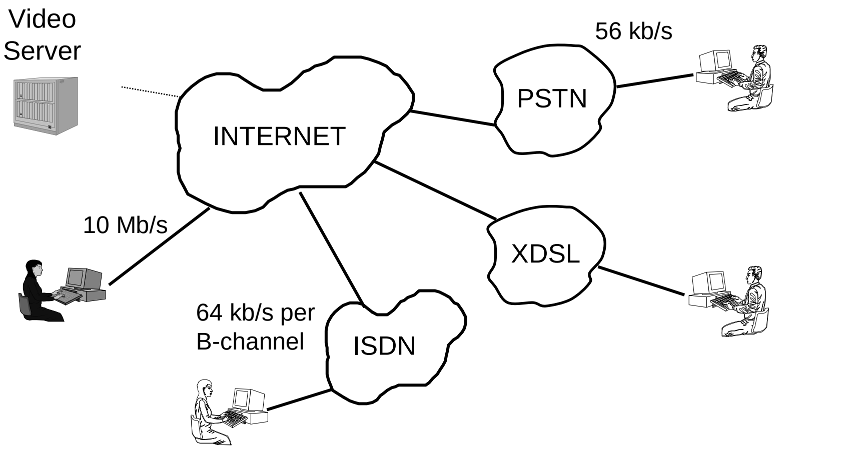
Heterogeneous Networks

Computer

MPEG-4 FGS decoder

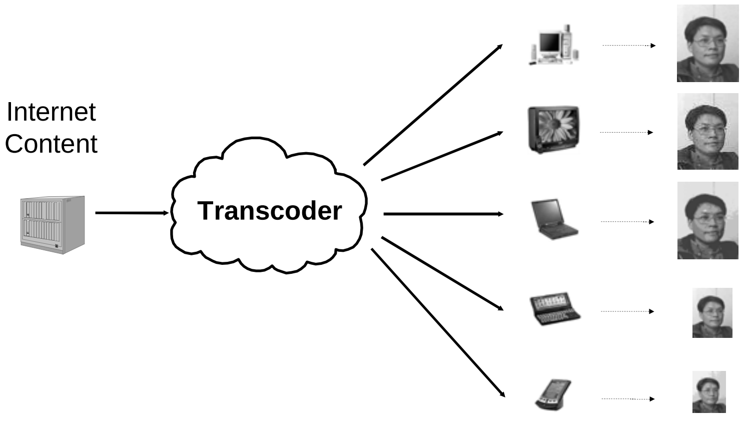
Application Example: Video Delivery over Heterogeneous Networks

To deliver multimedia data over heterogeneous networks



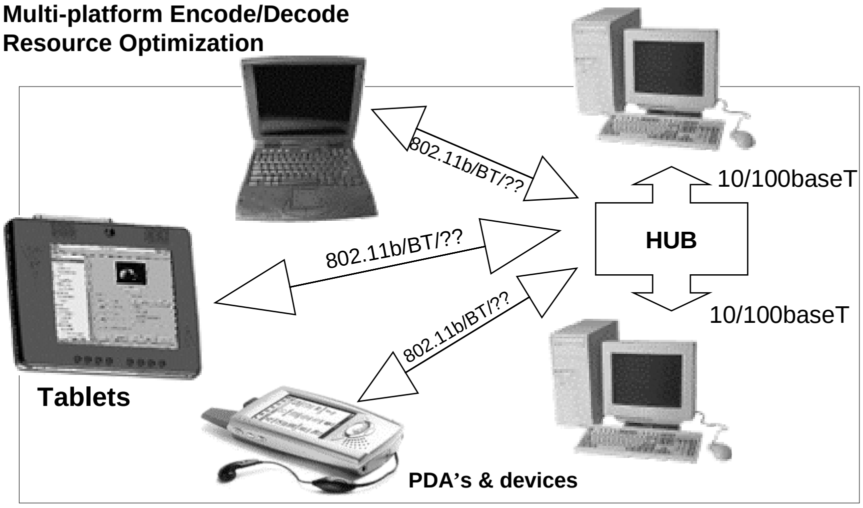
Application Example: Video Delivery to Heterogeneous Clients

To deliver multimedia data to diverse devices with different capabilities (Universal Multimedia Access)



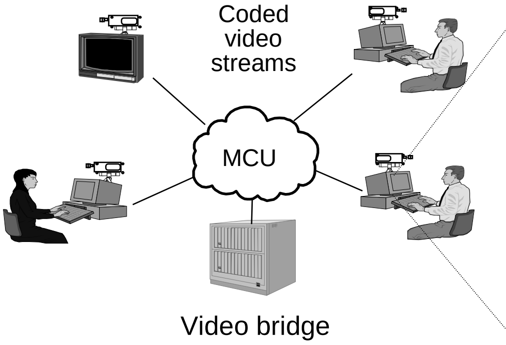
Application Example: Seamless Home Networking

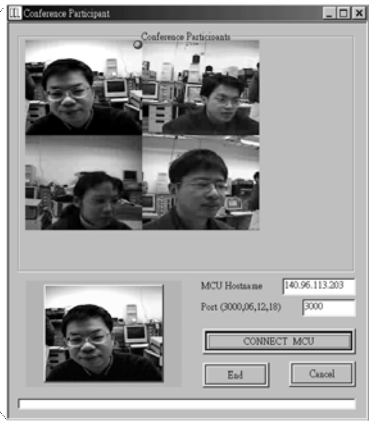
- **Transcoding for bandwidth management**
- **Robust Wireless Streaming/transmission**
- **Multi-platform Encode/Decode**
- **Resource Optimization**



Application Example: Dynamic Rate Control for Multipoint Video Conferencing

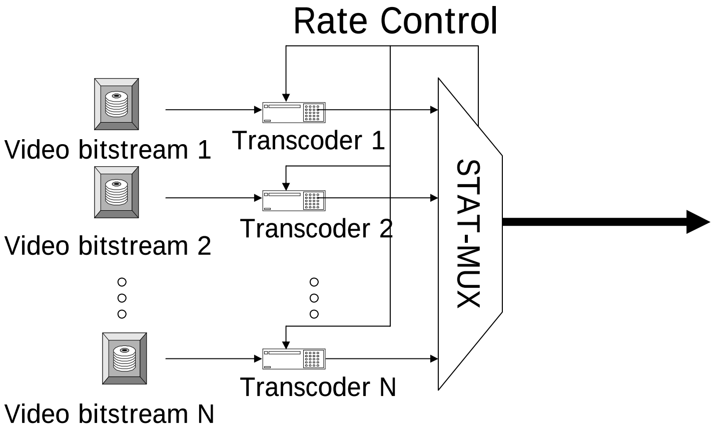
Example:
Input: 4 QCIF(176X144) videos @ 128 Kbps
Output: 1 CIF (352x288) videos @ 128 Kbps



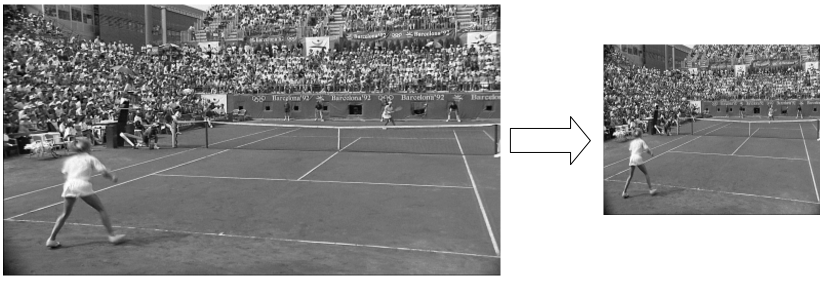


MCU Hostname 140.96.113.203
Port (3000,06,12,18) 3000

Application Example: Statistical Multiplexing

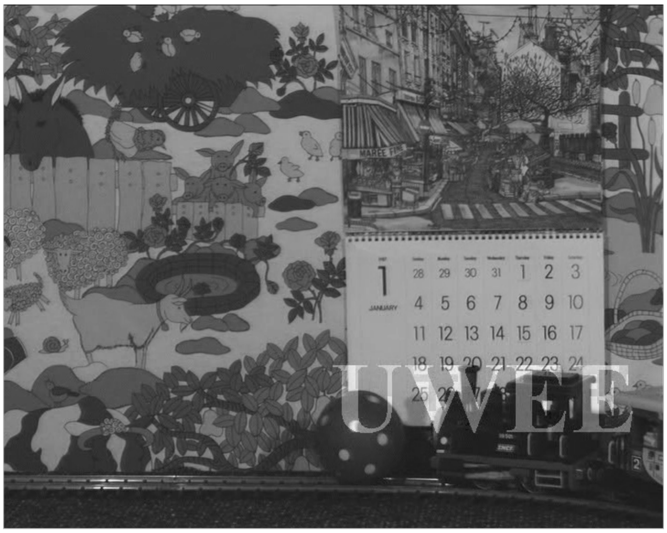


Application Example: HDTV ⇒ SDTV Transcoding



Page 8

Application Example: Watremark/Logo Insertion



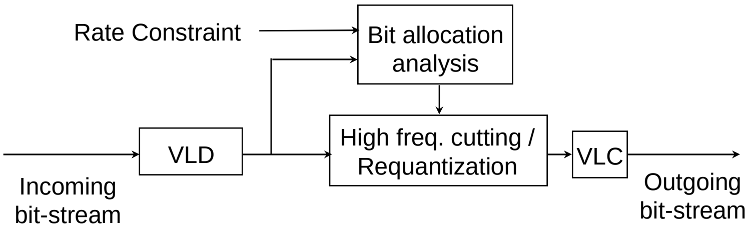
Transcoding Research

- Issues
  - Complexity
  - Video quality
  - Flexibility
- Key
  - How to utilize available information from input compressed video – motion vectors, coding statistics, etc. for best video quality, lowest complexity, and highest flexibility?
  - A kind of two-pass encoding

Transcoding Architectures

Open-Loop Coded-Domain Transcoder

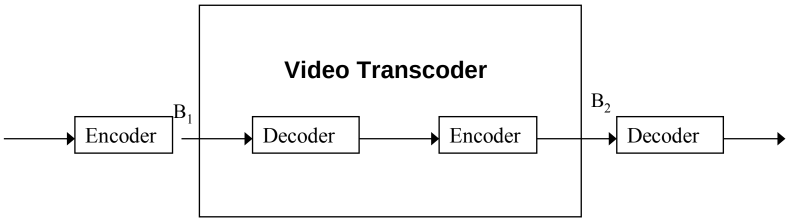
Truncation of high frequency or requantization of DCT coefficients



- No decoding and encoding, low complexity
- Significant drift between the front encoder and the end decoder

Cascaded Pixel-Domain Transcoder

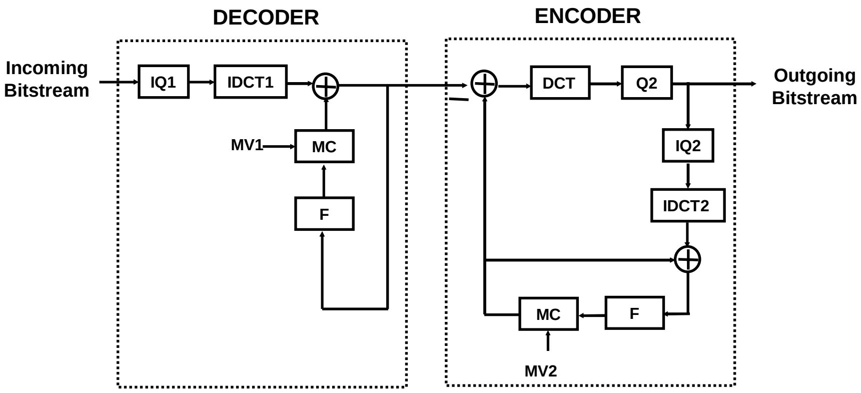
Direct implementation of Single Point Video Transcoder



- Purpose: bit-rate/format/standard conversion
- In video telephony application complexity and quality are the most important factors.
- Information reused to speed up the re-encoding process

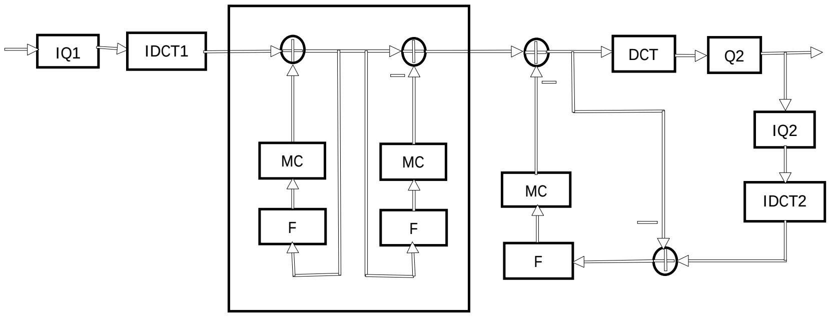
  - **Header information re-use**
  - **Motion vectors re-use**

Cascaded Pixel-Domain Transcoder

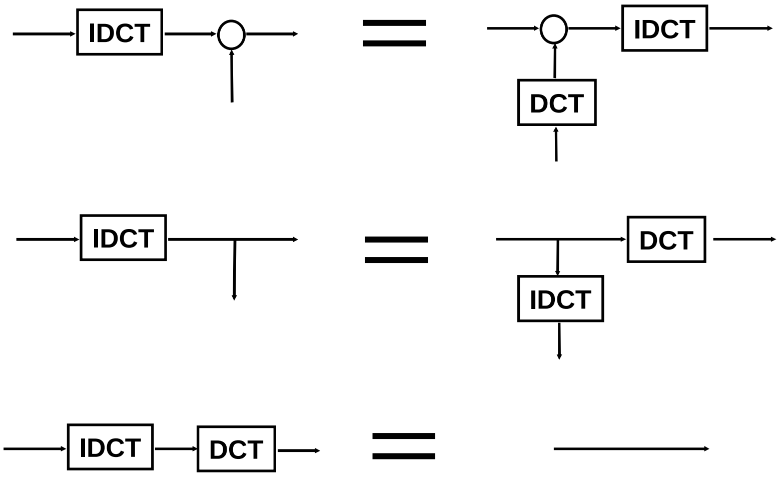


Simplification of Transcoder (1)

- **Remove one frame memory**

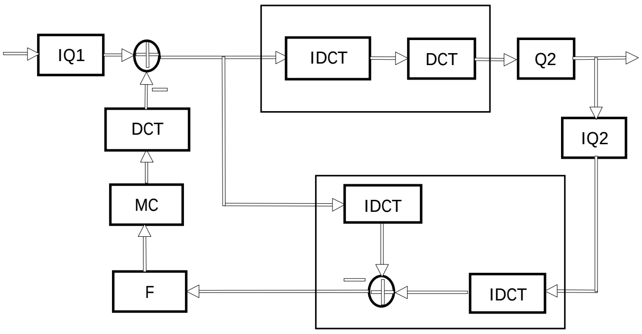


Simplification of Transcoder (2)

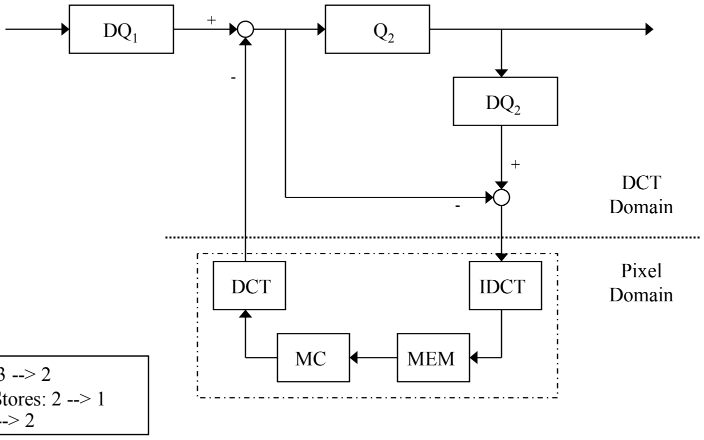


Simplification of Transcoder (3)

- **Re-arrange DCT/IDCT blocks**



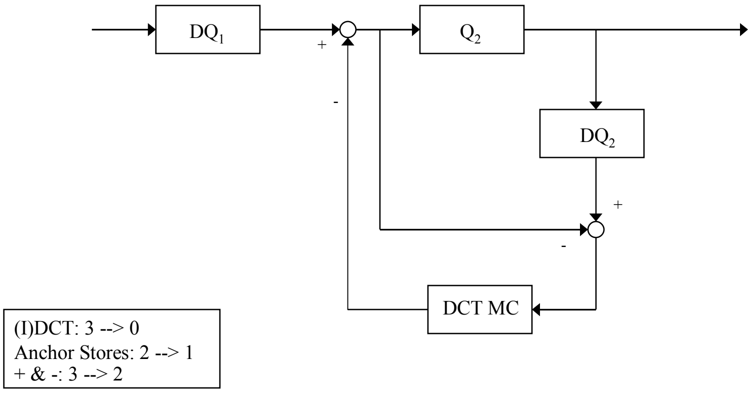
Simplified Pixel-Domain Transcoder



(I)DCT: 3 --> 2
Anchor Stores: 2 --> 1
+ & -: 3 --> 2

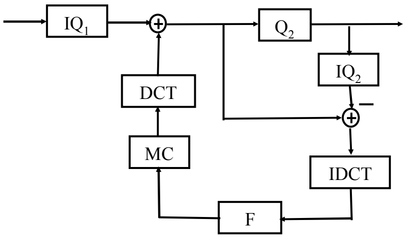
[G. Keesman et al]

Simplified DCT-Domain Transcoder

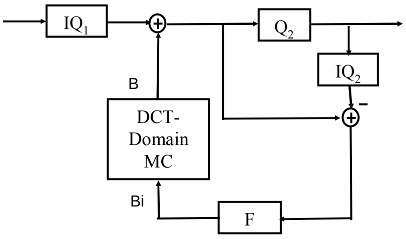


(I)DCT: 3 --> 0
Anchor Stores: 2 --> 1
+ & -: 3 --> 2

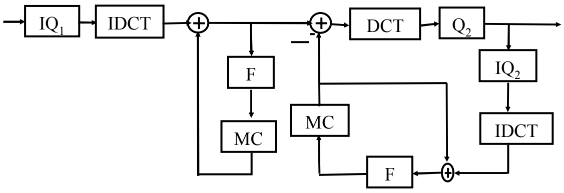
Fast Transcoder Architectures



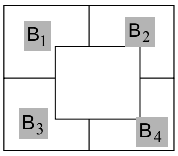
simplified pixel-domain transcoder



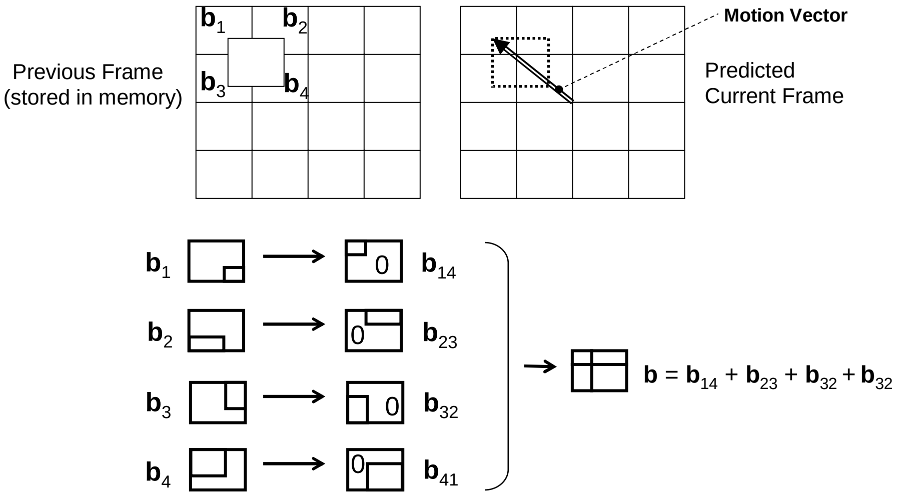
DCT-domain transcoder



Cascaded pixel-domain transcoder

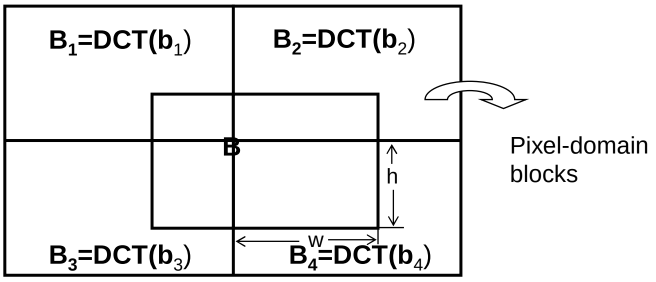


Motion Compensation in the Pixel Domain



$\mathbf{b} = \mathbf{b}_{14} + \mathbf{b}_{23} + \mathbf{b}_{32} + \mathbf{b}_{32}$

DCT-Domain Motion Compensation (1)



- How to compute the DCT coefficients B from B_1,B_2,B_3,B_4?

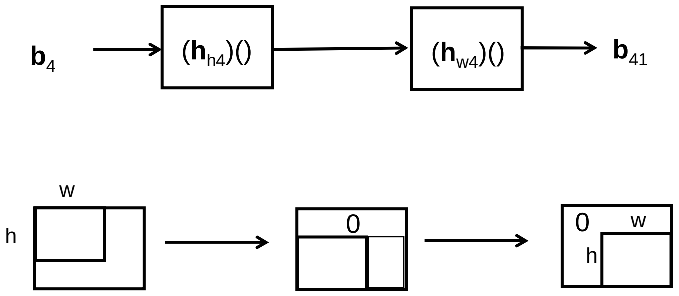
DCT-Domain Motion Compensation (2)

- $\mathbf{b}$=$\mathbf{b}_{14}$+$\mathbf{b}_{23}$+$\mathbf{b}_{32}$+$\mathbf{b}_{41}$
- DCT($\mathbf{b}$)=DCT($\mathbf{b}_{14}$)+DCT($\mathbf{b}_{23}$)+DCT($\mathbf{b}_{32}$)+DCT($\mathbf{b}_{41}$)
- If we find the relationship between elements of (B_{14} - B_{41}) and the elements (B_1 - B_4), then we can calculate the B=DCT($\mathbf{b}$) directly from the elements (B_1 - B_4),.
- Consider the contribution from the original block B_4

$$\mathbf{b}_{41} = \mathbf{h}_{h_4}\mathbf{b}_4\mathbf{h}_{w_4}, \quad \text{where}\,\mathbf{h}_{h_4} = \begin{bmatrix} 0 & 0 \\ I_{h_4} & 0 \end{bmatrix} \; \mathbf{h}_{w_4} = \begin{bmatrix} 0 & I_{w_4} \\ 0 & 0 \end{bmatrix}$$

- I_{h_4} and I_{w_4} are identity matrices of size $h_4 \times h_4$ and $w_4 \times w_4$ respectively.

DCT-Domain Motion Compensation (3)



DCT-Domain Motion Compensation (3)

- All unitary orthogonal transforms such as DCT are distributive to matrix multiplication, I.e.,

  DCT(AB)=DCT(A)DCT(B)
- The elements of B_{41} can be computed directly from the elements of B_4 through geometric transforms

  $$B_{41} = H_{h_4} B_4 H_{w_4}$$
- The DCT coefficients of block B can be computed using

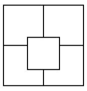
  $$B = \sum_{i=1}^{4} H_{h_i} B_i H_{w_i}$$
- and can be pre-computed and stored in memory.

DCT-Domain Motion Compensation (4)

Neighboring blocks	Position	$\mathbf{h}_{h_i}$	$\mathbf{h}_{w_i}$
$\mathbf{b}_1$	Lower right	$\begin{bmatrix} 0 & I_{8-h} \\ 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 \\ I_{8-w} & 0 \end{bmatrix}$
$\mathbf{b}_2$	Lower left	$\begin{bmatrix} 0 & I_{8-h} \\ 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & I_w \\ 0 & 0 \end{bmatrix}$
$\mathbf{b}_3$	Upper right	$\begin{bmatrix} 0 & 0 \\ I_h & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 \\ I_{8-w} & 0 \end{bmatrix}$
$\mathbf{b}_4$	Upper left	$\begin{bmatrix} 0 & 0 \\ I_h & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & I_w \\ 0 & 0 \end{bmatrix}$

DCT-Domain MC Example

- **Compute pixel values from neighboring blocks**:



- Example:

$$\left[\begin{array}{ccc|ccc} 1 & 9 & 3 & 9 & 8 & 7 \\ 6 & 4 & 5 & 3 & 9 & 6 \\ 4 & 3 & 8 & 7 & 1 & 2 \\ \hline 5 & 4 & 3 & 2 & 1 & 0 \\ 9 & 8 & 7 & 6 & 5 & 4 \\ 3 & 2 & 1 & 5 & 9 & 1 \end{array}\right]$$

$$\begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}\begin{bmatrix} 2 & 1 & 0 \\ 6 & 5 & 4 \\ 5 & 9 & 1 \end{bmatrix}\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \equiv \begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 1 \\ 0 & 6 & 5 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} 9 & 8 & 7 \\ 3 & 9 & 6 \\ 7 & 1 & 2 \end{bmatrix}\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \equiv \begin{bmatrix} 0 & 7 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} 1 & 9 & 3 \\ 6 & 4 & 5 \\ 4 & 3 & 8 \end{bmatrix}\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} \equiv \begin{bmatrix} 8 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}\begin{bmatrix} 5 & 4 & 3 \\ 9 & 8 & 7 \\ 3 & 2 & 1 \end{bmatrix}\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} \equiv \begin{bmatrix} 0 & 0 & 0 \\ 3 & 0 & 0 \\ 7 & 0 & 0 \end{bmatrix}$$

[S.F.Chang et al]

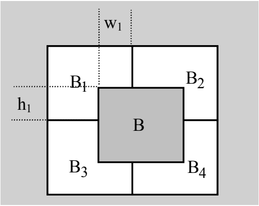
Simplified DCT-Domain Transcoder

$$B = \sum_{i=1}^{4} H_{h_i} B_i H_{w_i}$$

H_{h_i}, H_{w_i} : geometric transform matrices

$w_i, h_i \in \{1, 2, ..., 7\}$

$H_{h_1} = H_{h_2}, H_{h_3} = H_{h_4}, H_{w_1} = H_{w_3}, H_{w_2} = H_{w_4}$

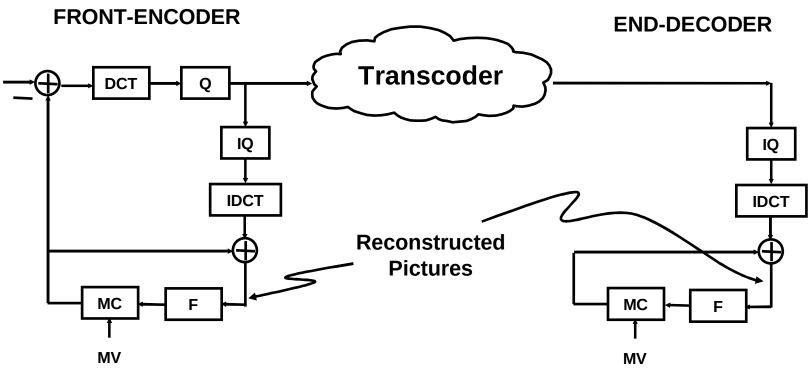


Problems of Existing Fast Transcoders

"Drift" due to:
- No clipping functions considered
- Motion compensation is not a linear operation when needs interpolations
- Assuming infinite accuracy in DCT/IDCT

Can not handle:
- Frame-rate conversion
- Frame-skipping
- Spatial resolution conversion
- Mode changes
- Different motion vectors

Drift in Transcoding



- The reconstructed pictures in encoder and decoder have to be **exactly same** to prevent "drift"

Clipping Functions

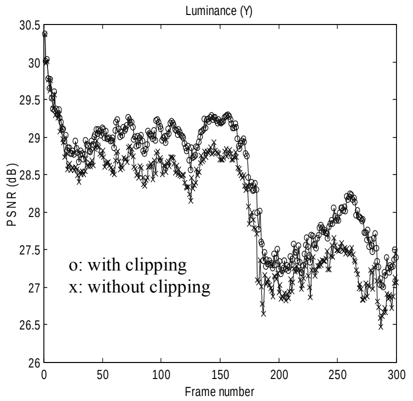
(a) Front-Encoder

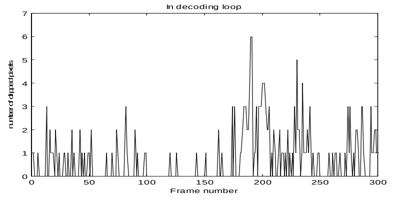
(b) End-Decoder

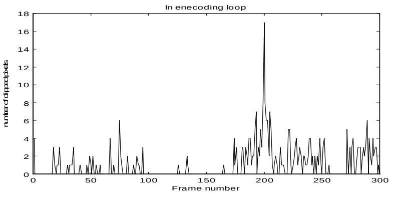
Drift by Clipping Functions

Carphone Sequence

(Q1=15 to QP2=20)





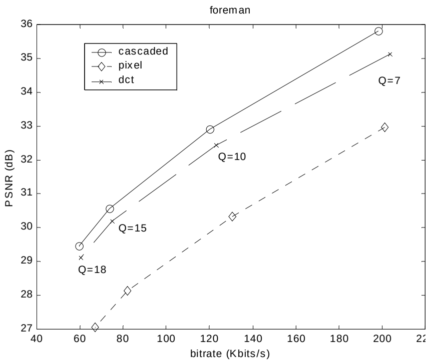


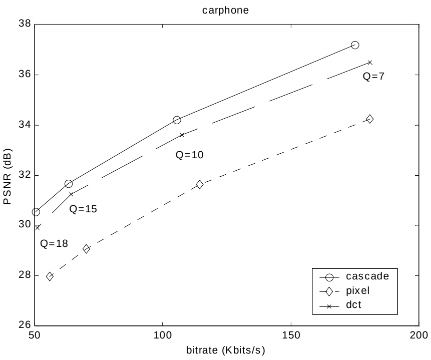
Page 20

Drift Performance Comparison

Foreman

Carphone

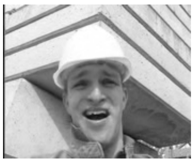




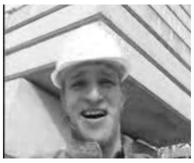
The incoming bit-streams were encodes with Q=5 and transcoded with Q=7, 10, and 15

Drift Performance Comparison

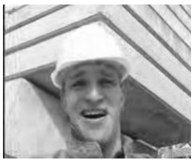
Subjective Quality (captured 130-th frame)



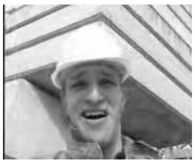
Incoming Bit-stream



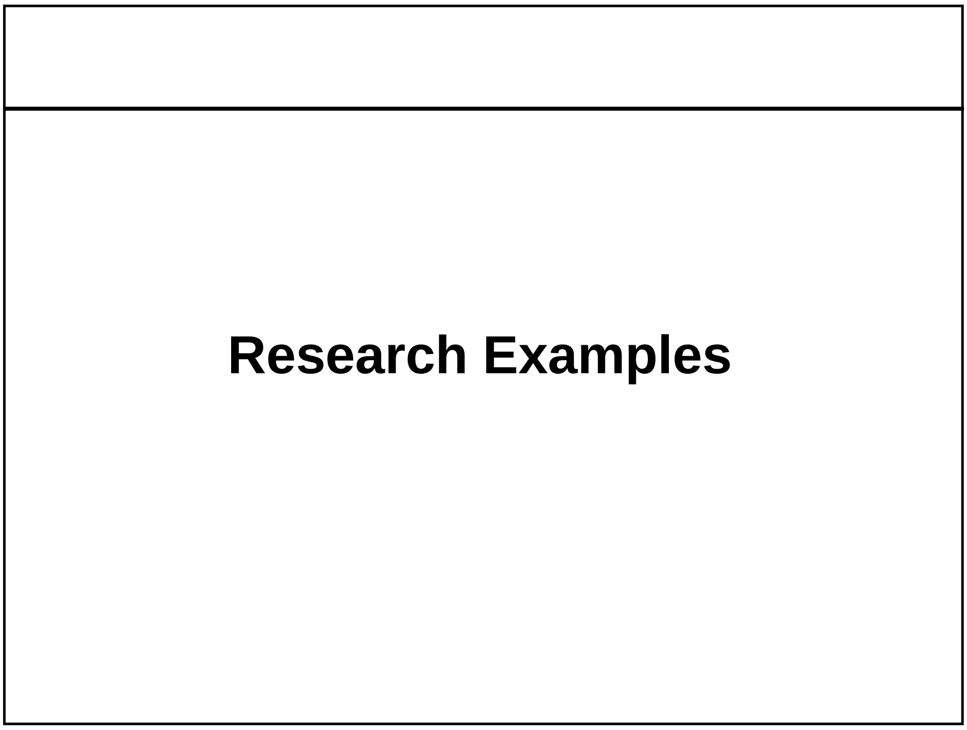
Cascaded Pixel-Domain Transcoding



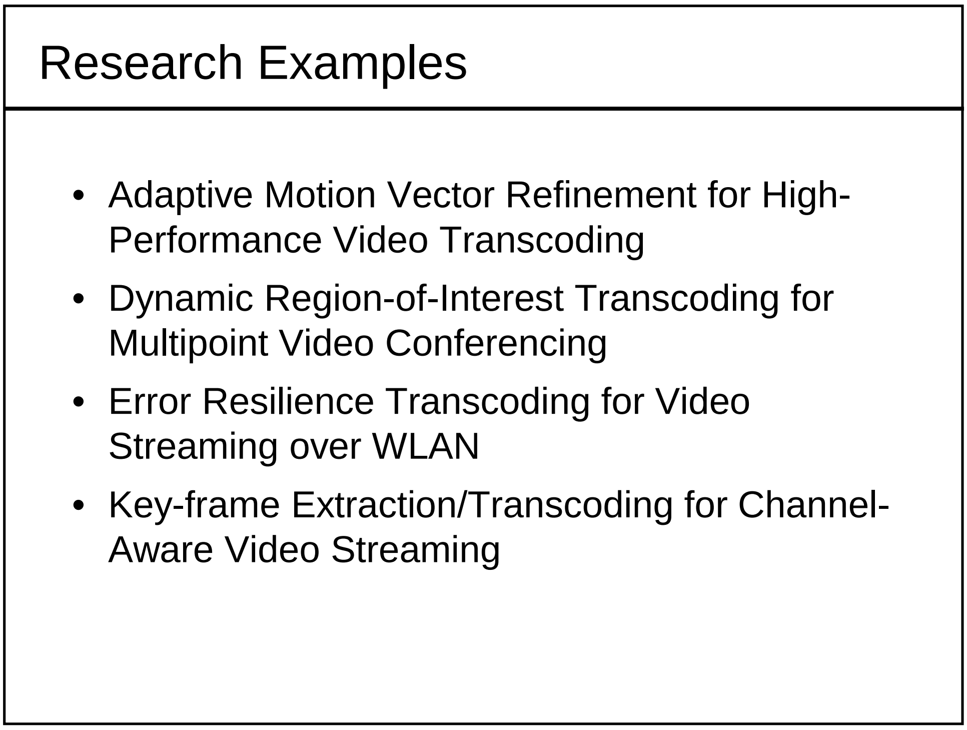
Fast Pixel-Domain Transcoding



DCT-Domain Transcoding



Research Examples



Research Examples

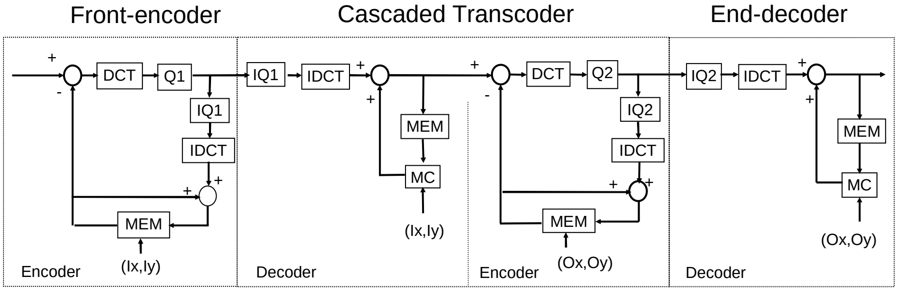
- Adaptive Motion Vector Refinement for High-Performance Video Transcoding
- Dynamic Region-of-Interest Transcoding for Multipoint Video Conferencing
- Error Resilience Transcoding for Video Streaming over WLAN
- Key-frame Extraction/Transcoding for Channel-Aware Video Streaming

Motion Vector Refinement for High-Performance Transcoding

IEEE Trans. Multimedia 1999

Prof. Chia-Wen Lin
Department of CS
National Chung Cheng University
886-5-272-0411 ext. 33120
cwlin@cs.ccu.edu.tw

Quantization Effect on Motion Vectors



Motion Vector Refinement

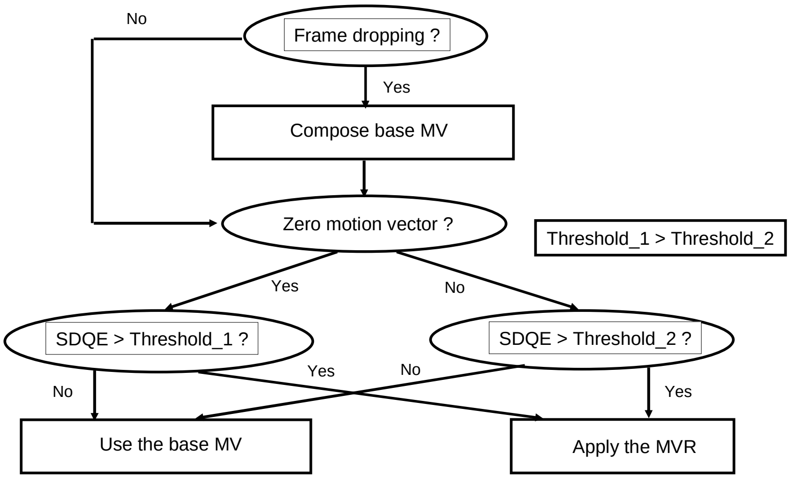
- Reduce the computation while keeping the high performance
- Using base and delta motion vector
- New search area can be much smaller than the original search area (e.g. vs. pixels)

Adaptive Motion Vector Refinement (1)

- **Sum of Differential Quantization Error (SDQE)**

where q_1 = quan. step-size extracted from incoming current frame
q_2 = quan. step-size used in the previous frame in the transcoder

Adaptive Motion Vector Refinement (2)


Frame dropping ?

No

Yes

Compose base MV

Zero motion vector ?

Threshold_1 > Threshold_2

Yes

No

SDQE > Threshold_1 ?

SDQE > Threshold_2 ?

No

Yes

No

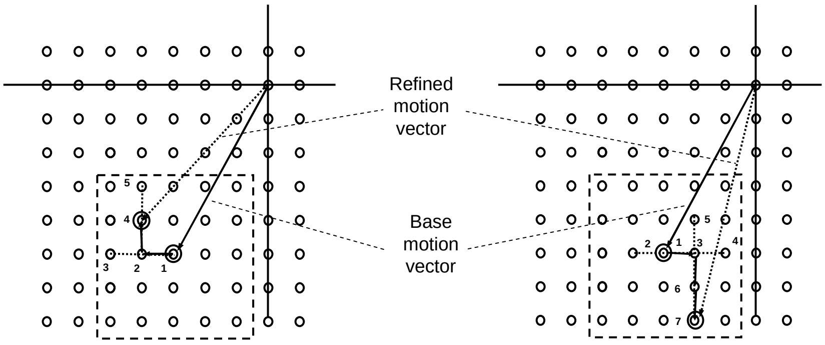
Yes

Use the base MV

Apply the MVR

Fast Search Algorithms

Horizontal And Vertical Search (HAV)


Refined motion vector

Base motion vector

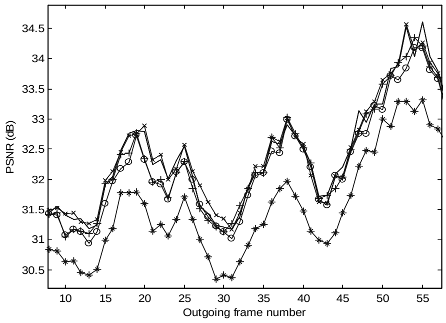
Best case (5 checking points)

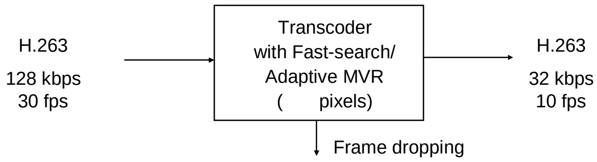
Worst case (7 checking points)

Performance Comparison (1)

“Carphone” Sequence

* : ONLY BASE MVs
+ : BBGDS+ADAP+MVR
o : HAVS+ADAP+MVR
x : FULL-MVR





H.263

128 kbps
30 fps

Transcoder
with Fast-search/
Adaptive MVR
(pixels)

H.263

32 kbps
10 fps

Frame dropping

Performance Comparison (2)

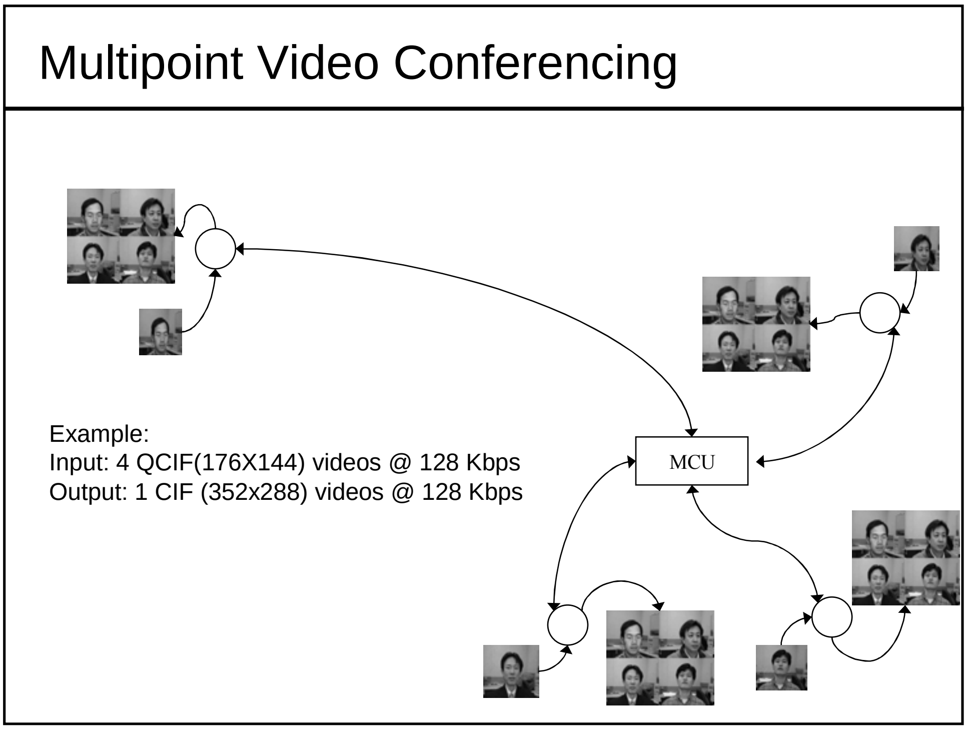
(Carphone Sequence, 128 kbps to 32 kbps)

Dynamic Region-of-Interest Transcoding for Multipoint Video Conferencing

Presented at ISCAS 2000
IEEE Trans. CSVT 2003

Prof. Chia-Wen Lin
Department of CS
National Chung Cheng University
886-5-272-0411 ext. 33120
cwlin@cs.ccu.edu.tw

Multipoint Video Conferencing



Example:
Input: 4 QCIF(176X144) videos @ 128 Kbps
Output: 1 CIF (352x288) videos @ 128 Kbps

Motivation

- Observation
  - In a multipoint video conference, in general, only one or two conferees are active at one time.
- **Considerations in this work**:
  - Video quality enhancement
    - Sub-windows of high activity requires higher frame rate to maintain acceptable temporal resolution (motion smoothness), while the frame rate for the inactive sun-windows can be lower.
    - The bit-rate saved from skipping the inactive sub-windows can be used to enhance the quality of the active ones.
  - Computation reduction
    - Significant computation reduction can be achieved by sub-window skipping.
- Question: What frames should or should not be skipped?

Dynamic Sub-Window Skipping

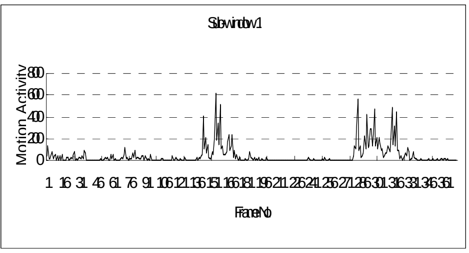
- Sub-window skipping criteria

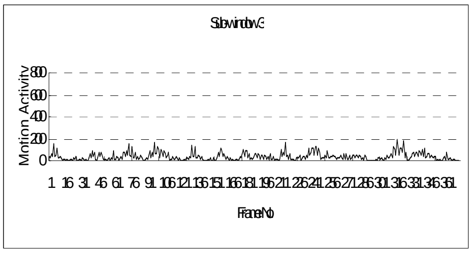
```
if  $(S_m^{MV} < TH_{\mathrm{MV1}})\,\&\&\left(\frac{SAD_m - SAD_m^{\mathrm{prev}}}{SAD_m^{\mathrm{prev}}} < TH_{\mathrm{SAD1}}\right)$
then
        Skip  Current  Sub-window
else
        Encode Current Sub-window
```

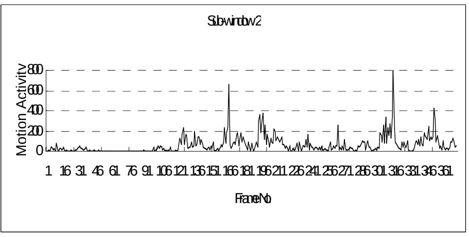
$$S_m^{MV} = \sum_{n=1}^{N} \left|MV_{m,n}^{x}\right| + \left|MV_{m,n}^{y}\right| \qquad SAD_{m,n} = \sum_{n=1}^{N} \sum_{x,y \in \mathrm{MB}_{m,n}} | f_m(x,y) - f_m^{\mathrm{prev}}(x + MV_{m,n}^{x}, y + MV_{m,n}^{y}) |$$

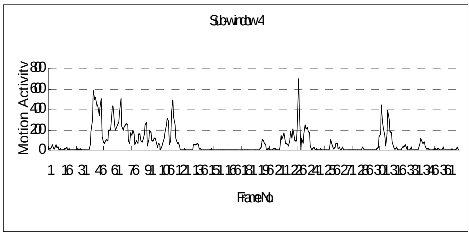
- The static sub-windows are skipped (distortion is relatively invisible) and the saved bits are used to enhance other sub-windows
- Computation complexity:
  - The MV's are decoded from the incoming bitstreams, thus only the summation operation is required.
  - The SAD computation cost is low
  - Sub-window skipping can achieve significant computation saving

Sub-window Activities

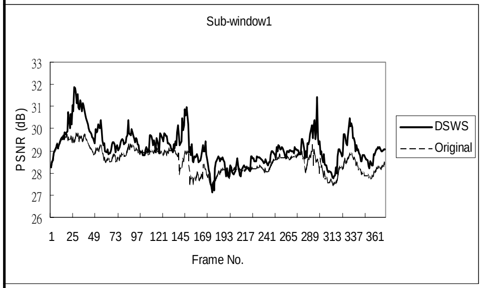


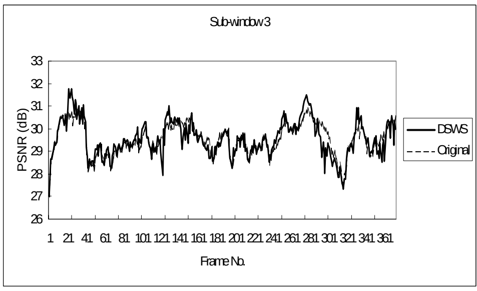


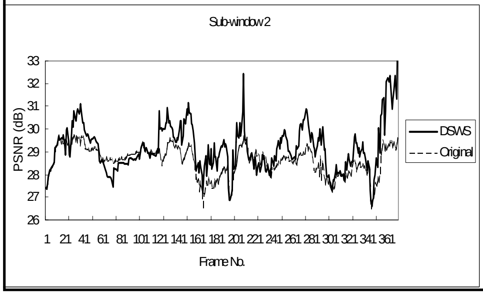


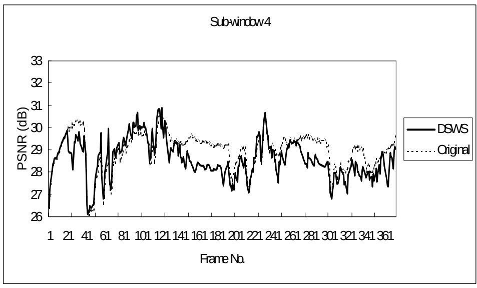


Performance Comparison (1)









Performance Comparison (1)

	Skipped frame No.	Average PSNR of all frames (dB)			Average PSNR of non-skipped frames (dB)		
		Original	DSWS		Original	DSWS	
Sub-sequence 1	151	28.54	29.14	+0.60	28.55	29.31	+0.76
Sub- sequence 2	75	28.59	29.16	+0.57	28.52	29.25	+0.73
Sub- sequence 3	54	29.54	29.56	+0.02	29.48	29.59	+0.11
Sub- sequence 4	139	28.99	28.59	-0.40	28.73	28.68	-0.05
Average	104.75	28.91	29.11	+0.20	28.82	29.21	+0.39

Skipped frame number (total :400 frames) and average PSNR of four sub-windows (*TH_1=5; TH_2=0.1*)

About 25% computation reduction is achieved

Performance Comparison (2)

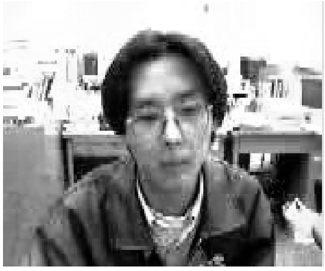
Without DSWS 29.45 dB

With DSWS 30.87 dB

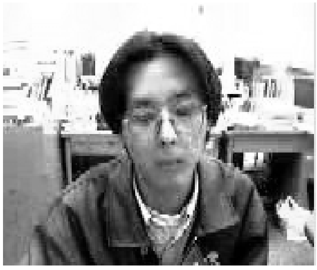
Performance Comparison (3)

Received

DSWS (28.10 dB)



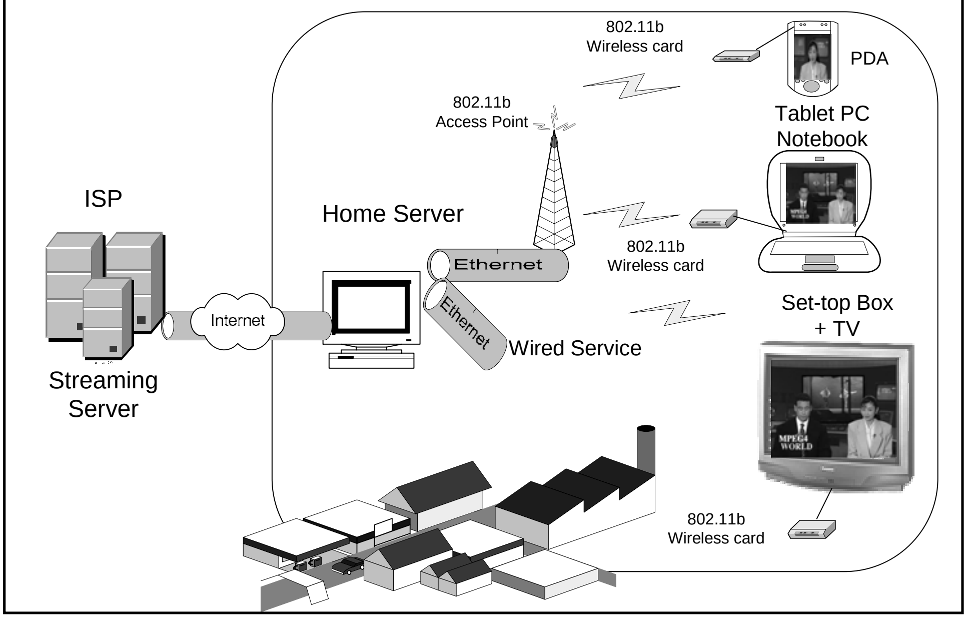
Without DSWS (29.38 dB)



Video Transcoding for Three-Tier Video streaming

Prof. Chia-Wen Lin
Department of CS
National Chung Cheng University
886-5-272-0411 ext. 33120
cwlin@cs.ccu.edu.tw

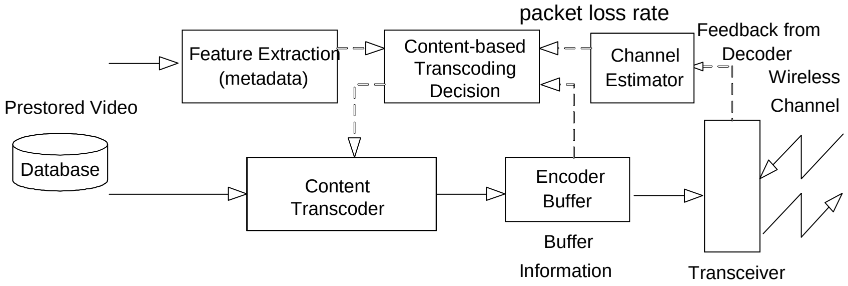
Application Scenario: Three-Tier Video Streaming for Home Networking



Video Adaptation for Three-Tier Video streaming

- WLANs are packet erasure channels
  – Burst errors due to slow-fading channels
  – Bit-level error-resilient coding tools and Bit/byte-level FEC are usually not useful
  – MAC-layer retransmission may degrade the quality seriously
- Transcoding is particularly useful for content adaptation in such application scenario
  – The number of home clients are usually small => reasonable complexity
  – The home server can easily capture the channel statistics through feedback information
  – Allow two-pass processing

Content-Aware Video Adaptation



- Our Approach
  - The server offline extracts the features from the pre-encoded bit-stream
  - The features are then stored as auxiliary data to guide the real-time transcoding
  - Intra-refresh & ARQ are adopted as the tools for content adaptation because they doesn't need to change the decoder

Error Resilience Transcoding for Video Streaming over WLAN

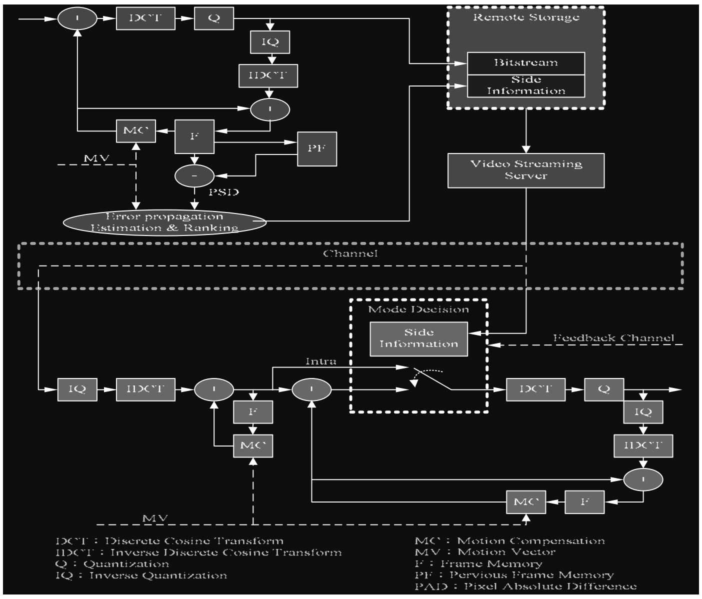
Presented at ISCAS 2004
J. Visual Commun. & Image Rep. 2005

Prof. Chia-Wen Lin
Department of CS
National Chung Cheng University
886-5-272-0411 ext. 33120
cwlin@cs.ccu.edu.tw

Proposed Architecture

Encoding Process

Transcoding Process



Two-Pass Error-Resilience Transcoding

- First pass – Encoding process
  - exploits the motion vectors and concealment distortion to evaluate the rank of the loss impact of each MB in a frame, each frame in a GOP
  - the rank serves as side information or transcoding hints to guide the second-pass (transcoding pass) intra-refresh decisions

Two-Pass Error-Resilience Transcoding

- Second-pass – Transcoding process
  - collect channel statistics from the client terminals to estimate the channel packet loss-rate
  - use side information and feedback channel condition to determine the intra-refresh rate
  - perform intra-refresh on the highest-priority MBs (with top-ranked loss impact)

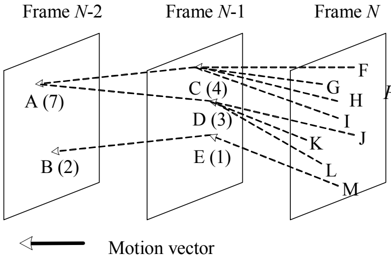
Loss-Impact Estimation in First-Pass Encoding

- Involves three steps
  - Calculating the Pixel-level Loss Impact
  - Calculating the MB-level Error-Propagation
  - Calculating the Frame-level Error Propagation

Pixel-Level Loss-Impact Estimation

- pixel-level loss-impact (LI)
  - to characterize the amount of pixel-wise error propagation
  - PRC: Pixel Reference Count
  - PCE: Pixel Concealment Error

$$LI(x,y,n) = PCE(x,y,n) \times PRC(x,y,n)$$



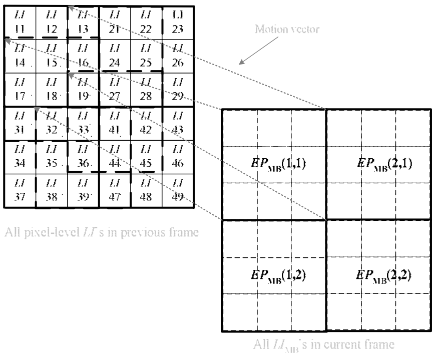
$$PRC(x,y,n) = \begin{cases} \sum\limits_{(x',y',n+1) \text{ points to } (x,y,n)} PRC(x',y',n+1) & 1 \le n < N_{\text{GOP}} \\ 1 & n = N_{\text{GOP}} \end{cases}$$

$$PCE(x,y,n) = \left| f(x,y,n) - f(x,y,n-1) \right|^2$$

Estimation of MB- and Frame-Level Error Propagation

$$EP_{\text{MB}}(m,n) = \sum_{(x,y) \in \text{MB}_m} LI(x + \text{MV}_x, y + \text{MV}_y, n-1)$$

- m: the MB index in a frame
- (x,y): the pixel coordinate
- n: the time index
- (MV_x,MV_y): the associated MV of pixel (x,y)



Frame-level error propagation:

$$EP_n = \sum_{m=1}^{N_{\text{MB}}^{\text{F}}} EP_{\text{MB}}(m,n)$$

Transcoding with Adaptive Intra-Refresh: GOP-Level Allocation

- The number of intra-refreshed MBs in a GOP

$$N_{\text{intra}}^{\text{GOP}} = \frac{\frac{1}{N_{\text{GOP}}}\sum_{n=1}^{N_{\text{GOP}}} EP_n \times PLR}{TH_{\text{intra}}}$$

  - *N*GOP: number of frames in a GOP
  - *PLR*: packet-loss rate
  - TH_{intra}: scaling parameter to characterize the relationship between $N_{\text{intra}}^{\text{GOP}}$ and the error propagation effect in a GOP

Transcoding with Adaptive Intra-Refresh Frame-Level Allocation

- **if** *n* = 1

$$N_{\text{intra}}^{n} = \frac{EP_n}{\sum_{i=n}^{N_{\text{GOP}}} EP_i} \times N_{\text{intra}}^{\text{GOP}}$$

- **else if** 2 ≤ *n* ≤ *N*GOP

$$N_{\text{intra}}^{n} = \frac{EP_n}{\sum_{i=n}^{N_{\text{GOP}}} EP_i} \times \left(N_{\text{intra}}^{\text{GOP}} - \sum_{i=1}^{n-1} N_{\text{intra}}^{i} \right)$$

- **endif**
- **if** $N_{\text{intra}}^{n} > k_{\text{MB}} N_{\text{MB}}^{\text{F}}$ **then** $N_{\text{intra}}^{n} = k_{\text{MB}} N_{\text{MB}}^{\text{F}}$

Notation:

n: frame index in a GOP

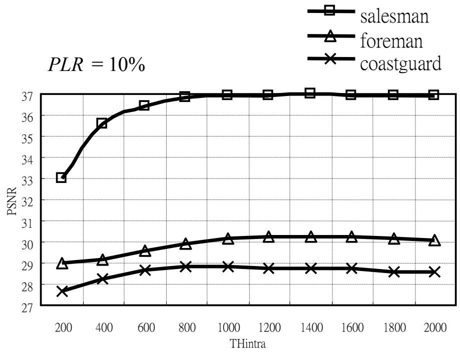
N_{intra}^{n}: number of intra-coded MBs in frame *n*

N_{MB}^{f} : number of MBs in a frame

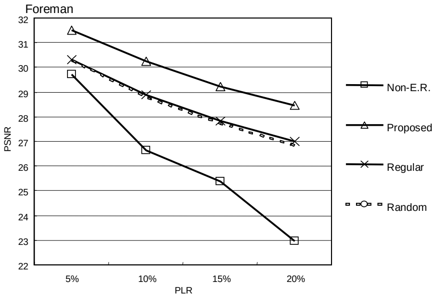
k_{MB} : to constrain the number of intra-coded blocks in a frame not to exceed an upper limit (0 ≤ k_{MB} ≤ 1)

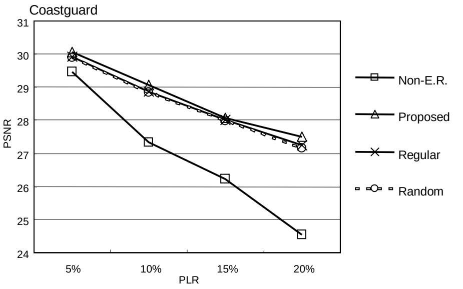
Threshold Selection

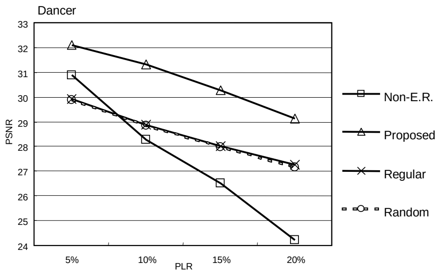
- The parameter TH_{intra} is obtained empirically
  - the following figure shows the frame-by-frame PSNR with different TH_{intra}*'s*
  - the best PSNR performance is achieved at $TH_{\text{intra}} \cong 1200$ for the three sequences (PLR = 10%)

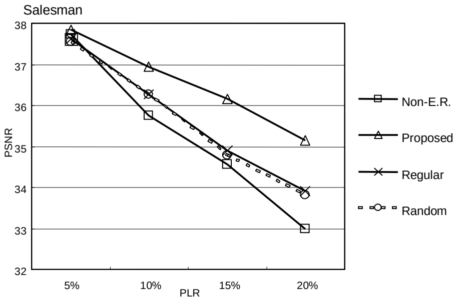


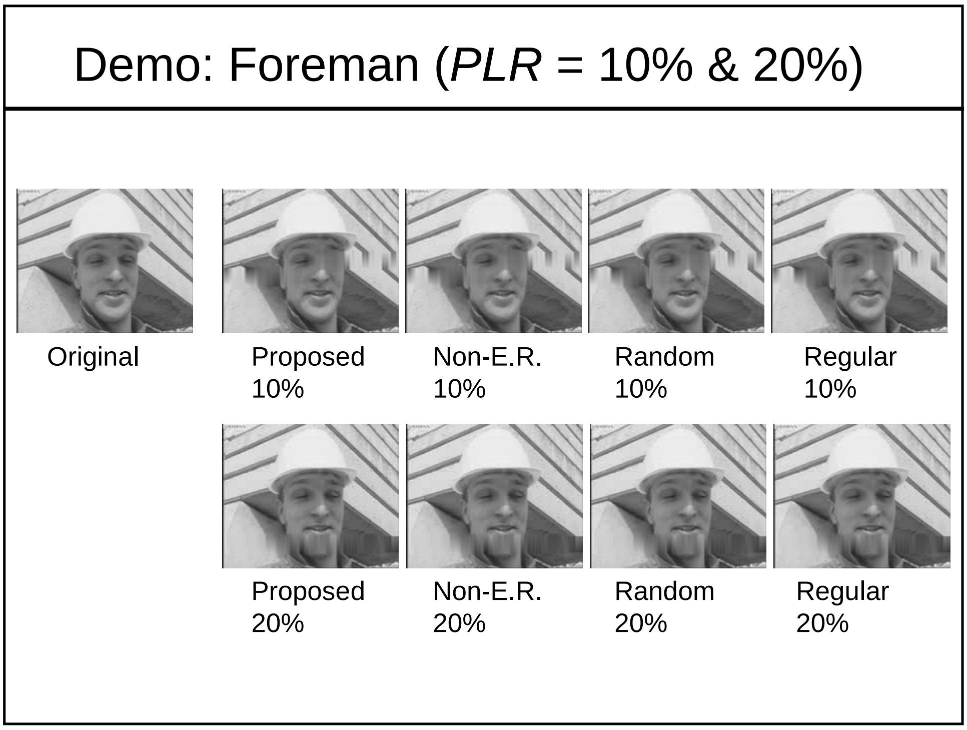
Performance Comparison











Demo: Foreman (*PLR* = 10% & 20%)

Original

Proposed 10%

Non-E.R. 10%

Random 10%

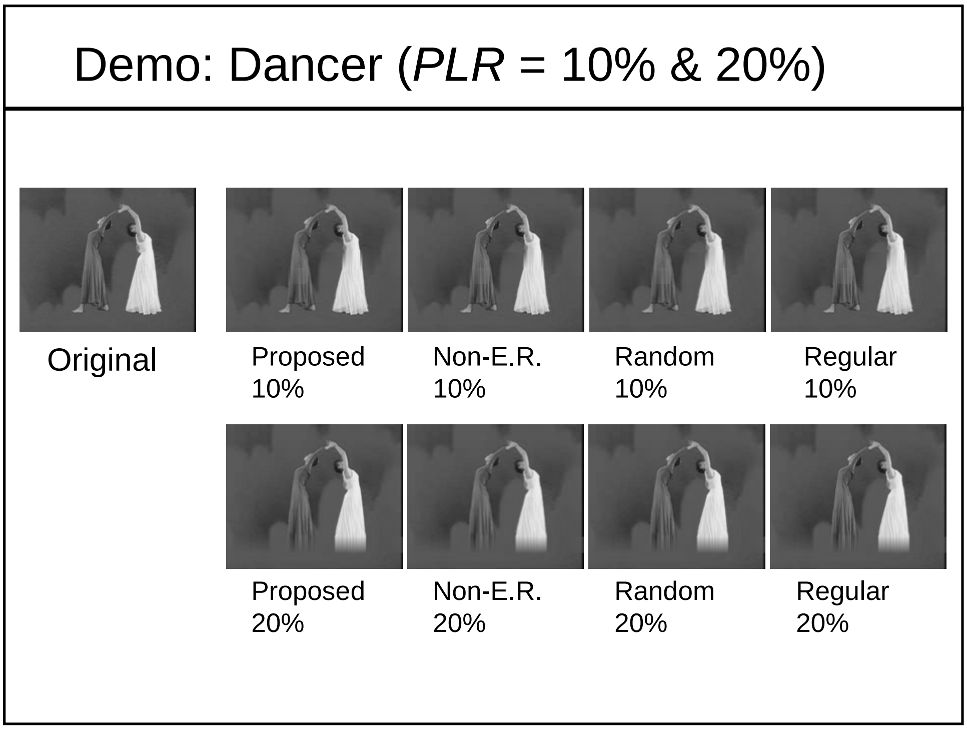
Regular 10%

Proposed 20%

Non-E.R. 20%

Random 20%

Regular 20%



Demo: Dancer (*PLR* = 10% & 20%)

Original

Proposed 10%

Non-E.R. 10%

Random 10%

Regular 10%

Proposed 20%

Non-E.R. 20%

Random 20%

Regular 20%

Demo: Salesman (*PLR* = 10% & 20%)

Original

Proposed 10%

Non-E.R. 10%

Random 10%

Regular 10%

Proposed 20%

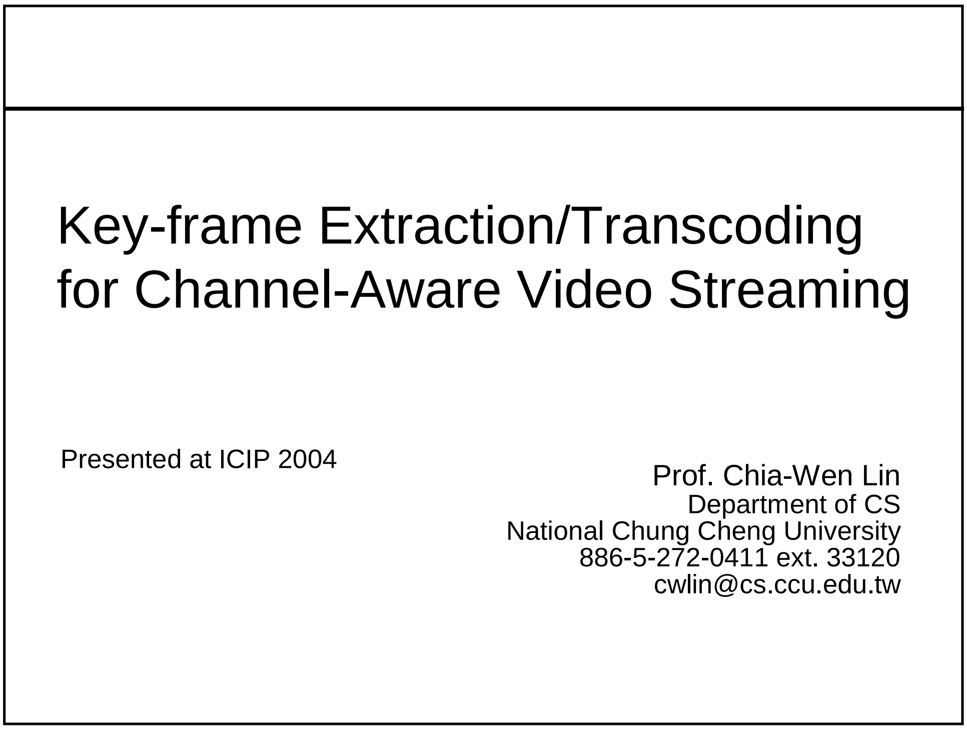
Non-E.R. 20%

Random 20%

Regular 20%

Summary

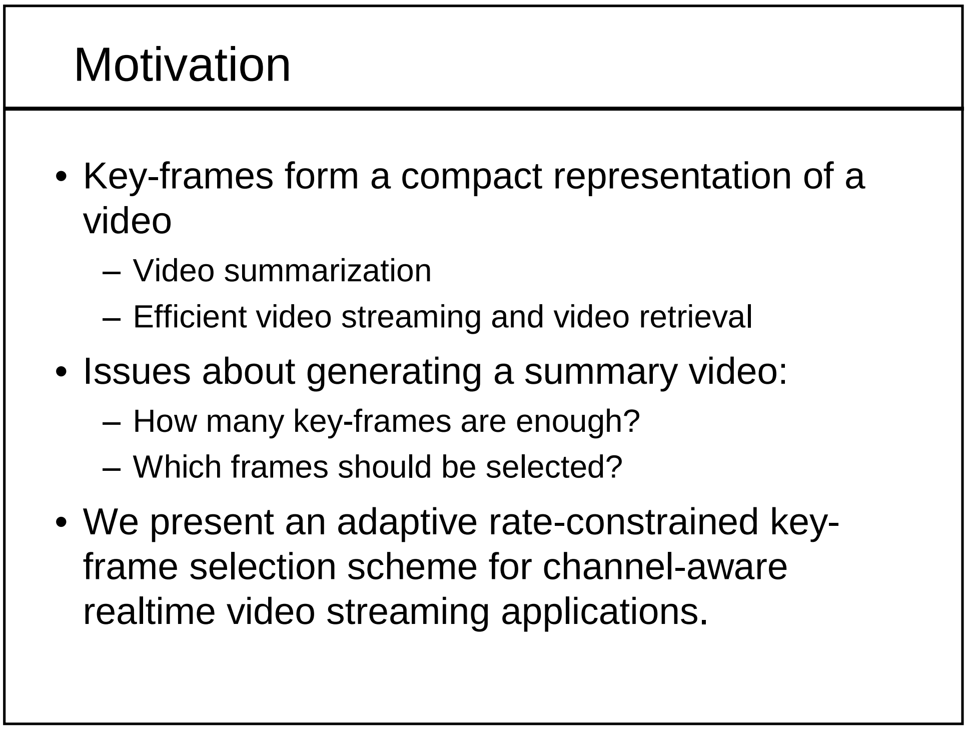
- We proposed a novel two-pass prioritized intra-refresh strategy for error-resilience transcoding
- The extra computational complexity required for the transcoding process is almost negligible, thereby making it suitable for real-time applications
- The proposed algorithm can effectively mitigate the error propagation due to packet-loss
- The degree of error resilience can be dynamically adjusted to adapt to a channel with time-varying error characteristics



Key-frame Extraction/Transcoding for Channel-Aware Video Streaming

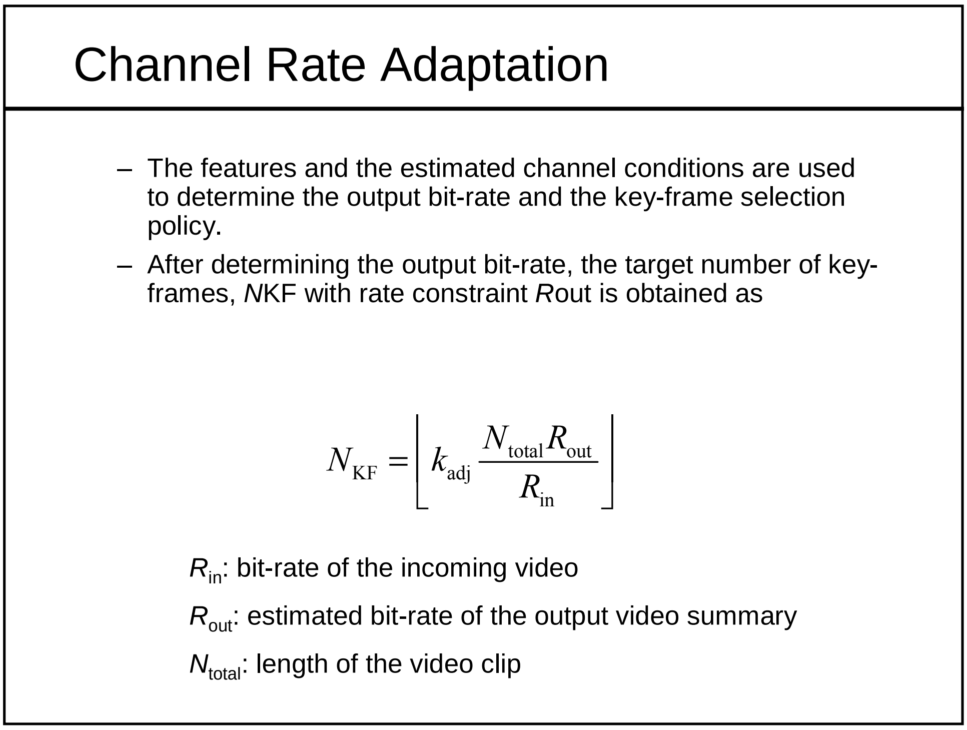
Presented at ICIP 2004

Prof. Chia-Wen Lin
Department of CS
National Chung Cheng University
886-5-272-0411 ext. 33120
cwlin@cs.ccu.edu.tw



Motivation

- Key-frames form a compact representation of a video
  - Video summarization
  - Efficient video streaming and video retrieval
- Issues about generating a summary video:
  - How many key-frames are enough?
  - Which frames should be selected?
- We present an adaptive rate-constrained key-frame selection scheme for channel-aware realtime video streaming applications.



Channel Rate Adaptation

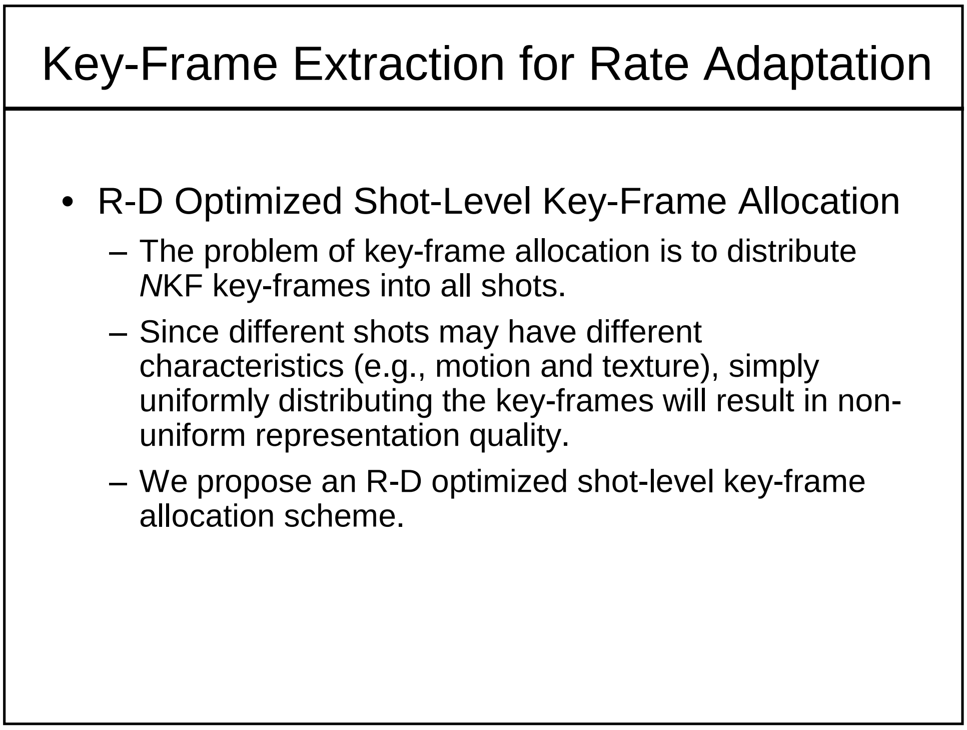
- The features and the estimated channel conditions are used to determine the output bit-rate and the key-frame selection policy.
- After determining the output bit-rate, the target number of key-frames, *N*KF with rate constraint *R*out is obtained as

$$N_{\mathrm{KF}} = \left\lfloor k_{\mathrm{adj}} \frac{N_{\mathrm{total}} R_{\mathrm{out}}}{R_{\mathrm{in}}} \right\rfloor$$

R_{in}: bit-rate of the incoming video

R_{out}: estimated bit-rate of the output video summary

N_{total}: length of the video clip



Key-Frame Extraction for Rate Adaptation

- R-D Optimized Shot-Level Key-Frame Allocation
  - The problem of key-frame allocation is to distribute *N*KF key-frames into all shots.
  - Since different shots may have different characteristics (e.g., motion and texture), simply uniformly distributing the key-frames will result in non-uniform representation quality.
  - We propose an R-D optimized shot-level key-frame allocation scheme.

Compressed-Domain Feature Extraction & Distance Metrics

- Given two finite point sets *A* = {*a*1,…,*ap*} and *B* = {b1,…,*bq*}, the Hausdorff distance is

$$H(A,B) = \max(h(A,B), h(B,A))$$

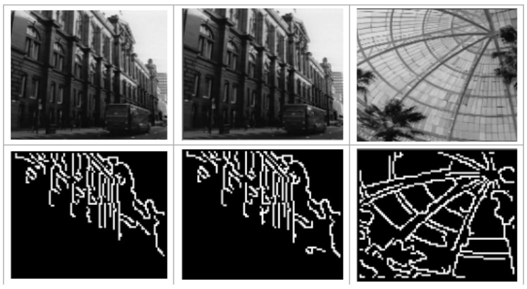
where $h(A,B) = \max_{a\in A} \min_{b\in B} \|a-b\|$

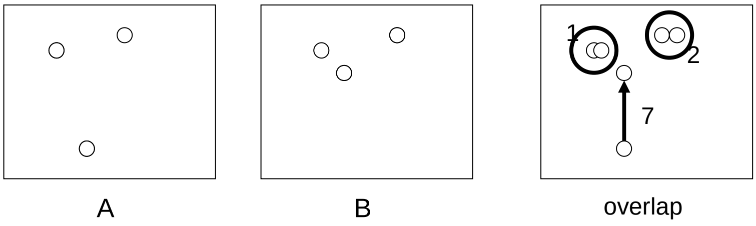
- Because the Hausdorff distance is sensitive to noise, a modified Hausdorff distance metric is proposed

$$h_K(A,B) = K^{\text{th}}_{a\in A} \min_{b\in B} \|a-b\|$$

Compressed-Domain Feature Extraction & Distance Metrics (Cont.)

Hausdorff distance of Canny edge images:





Compressed-Domain Feature Extraction & Distance Metrics (Cont.)

- Extract the DC image and then apply the Canny edge operator to extract the Canny edgy images.
- The spatial distance between the *i*th and *j*th edge images :

$$d^{\mathrm{S}}(f_i, f_j) = \sum_{n=i}^{j-1} h_K(f_n, f_{n+1})$$

- To take into account the object and camera motions, we define the temporal distance as the cumulative sum of motion vector magnitudes.

$$d^{\mathrm{T}}(f_i, f_j) = \sum_{n=i+1}^{j} \sum_{m=1}^{N_{\mathrm{MB}}} \left|MVX_m^n\right| + \left|MVY_m^n\right|$$

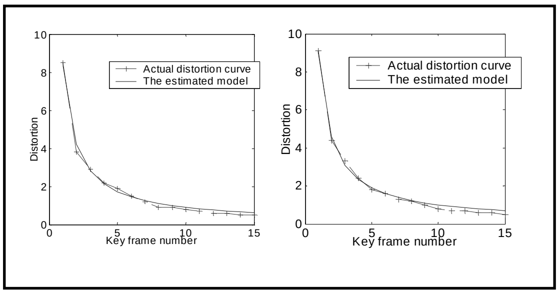
- Spatiotemporal distance

$$d(f_i, f_j) = k_{\mathrm{T}} d^{\mathrm{T}}(f_i, f_j) + k_{\mathrm{S}} d^{\mathrm{S}}(f_i, f_j)$$

Shot-Level Rate-Distortion Modeling

- Assume that a video clip has *N*shot shots, and the *i*-th shot has its rate-distortion function $D_i(N_i)$ to characterize the relationship between the average representation distortion (D_i) and the number of key-frames (N_i).
- We found the following first-order model is a good approximation of $D_i(N_i)$:

$$D_i\,(N_i) = \frac{a_i}{N_i} + b_i$$



Optimal Shot-Level Key-frame Allocation

- The optimal key-frame allocation problem can be formulated as the following constrained optimization problem:

$$\min \sum_{i=1}^{N_{\text{shot}}} L_i D_i(N_i) \quad \text{subject to} \quad \sum_{i=1}^{N_{\text{shot}}} N_i = N_{\text{KF}}$$

- Then use the Lagrange multiplier to convert the above Eq. to the following unconstrained optimization problem

$$f = \sum_{i=1}^{N_{\text{shot}}} L_i\left(\frac{a_i}{N_i} + b_i\right) + \lambda\left(\sum_{i=1}^{N_{\text{shot}}} N_i - N_{\text{KF}}\right)$$

- By setting partial derivatives to zero (i.e., $\partial f/\partial N_i$=0 and $\partial f/\partial \lambda$ =0),

$$\lambda = \frac{\left(\sum_{i=1}^{N_{\text{shot}}} \sqrt{L_i a_i}\right)^2}{N_{\text{KF}}^2} \quad \text{and} \quad N_i = \sqrt{\frac{L_i a_i}{\lambda}}$$

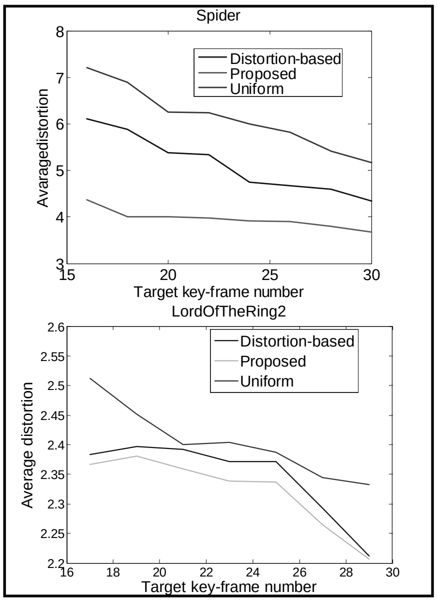
Experimental Results

- Two movie clips consisting of 1132 and 1123 frames respectively are encoded at 30 fps and 1Mbps using an MPEG-4 encoder.
- Two other allocation schemes are compared
  - uniform allocation

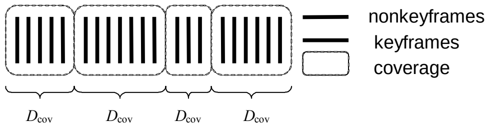
$$N_i = N_{\text{KF}} \times L_i / \sum_{i=1}^{N_{\text{shot}}} L_i$$

  - distortion-weighted allocation

$$N_i = N_{\text{KF}} \times L_i D_i / \sum_{i=1}^{n} L_i D_i$$



Optimal Shot-Level Key-Frame Allocation



- Key-frame selection within a shot
  - **Step 1.** Compute the average coverage range of each key-frame

$$D_{\text{cov}} = \frac{1}{N}\sum_{n=1}^{M-1} d(f_n, f_{n+1})$$

  - Divide the shot into *N* intervals with approximately equal average distortion

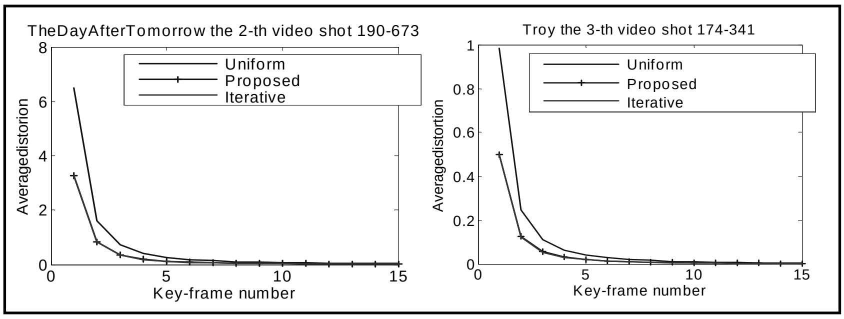
$$\sum_{n=t_{j-1}}^{t_j-1} d(f_n, f_{n+1}) \cong D_{\text{cov}} \quad j = 1, 2, ..., N$$

  - **Step 2.** For each interval *Ti*, the key-frame is selected so as to minimize the distortion *D* (*Ti*,*ki*):

$$k_i = \arg \min_{t_{i-1} \le k_i \le t_i} D(T_i, k_i), \quad i = 1, 2, ..., N$$

Result of Key-frame Extraction from a Shot

- The average distortion between frames and the corresponding key-frames is used for performance evaluation
- The proposed non-iterative approach can achieve close quality with significantly lower complexity as compared to the iterative approach



Demo: Key-frame Extraction

Movie,

1132 frames,

70 key frames

Movie,

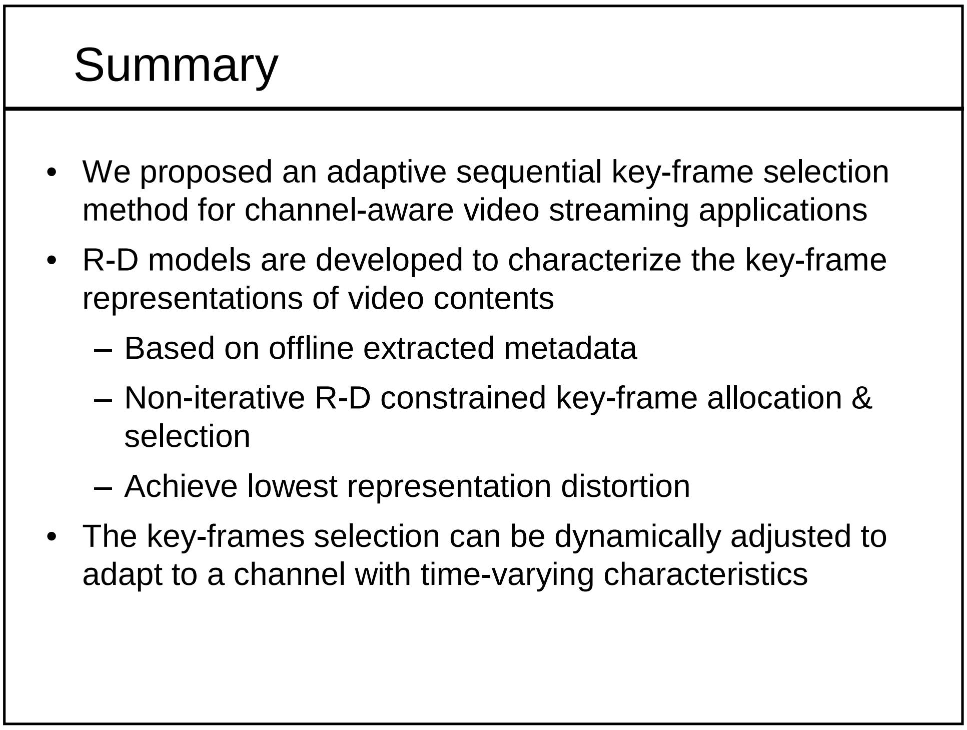
1123 frames,

55 key frames

Demo: Key-frame Extraction

1132 frames, 20 target key-frame

Proposed **Uniform** **Distortion-weighted**



Summary

- We proposed an adaptive sequential key-frame selection method for channel-aware video streaming applications
- R-D models are developed to characterize the key-frame representations of video contents
  - Based on offline extracted metadata
  - Non-iterative R-D constrained key-frame allocation & selection
  - Achieve lowest representation distortion
- The key-frames selection can be dynamically adjusted to adapt to a channel with time-varying characteristics