On Video Transcoding to Super-Resolution Videos

Keynote Speaker:
Professor W.C. Siu
Chair Professor and Director
Centre for Signal Processing
Department of Electronic and Information Engineering
Hong Kong Polytechnic University

Contents:
1. Introduction
   1.1 Hybrid Video Coding
   1.2 Object Oriented Coding
   1.3 Advanced Video Concepts
   1.4 A highlight of others of our studies
2. Motion Estimation Algorithms
   2.1 Sample studies: Fast Adaptive Search Algorithm, Fast Pixel Decimation
   2.2 Sample studies: Fast Exhaustive Full Search, Novel Directional Search
3. Video Transcoding
   3.1 Video Transcoding (frame skipping)
   3.2 Video Transcoding (H.264 to H.264 conversion)
4. Extending to video Enlargement Super-resolution Videos
   4.1 New Edge-Directed Interpolation
   4.2 Modified Edge-Directed Interpolation
   4.3 SR video Construction
   4.4 SR video re-encoding
5. Summary and Conclusion
1. Introduction

1.1 Hybrid Video Coding

In the recent years, there is a remarkable progress in Video Coding. In this talk we mainly concentrate on predictive Hybrid Video Coding.

**Predictive Coding:**

Instead of transmitting a frame (called *current frame*), its *motion vectors* with reference to a previous frame (called *reference frame*) are transmitted.

This will produce the *motion compensated frame* which consists of prediction errors.

Most videos nowadays are coded by using the *Hybrid Video Coding* which makes use of the Predictive Coding.

In order to recover the signal without the motion estimation errors, a motion compensated *Residual Frame* is constructed, such that

\[
\text{Residual frame} = \text{Current frame} - \text{Motion compensated frame}
\]

This predicted residual frame is subsequently coded in a similar manner as an intra frame

\[ i.e. \text{through DCT} \rightarrow \text{Quantization} \rightarrow \text{Entropy coding} \]
1.2 Object Oriented Coding

- To divide the scene into Background object(s):
  Hybrid Coding, or Sprite Generation Technique
- and foreground objects:
  Multiple objects
  Hybrid Coding
  Object based coding (complete object)
  Time, position, motion manipulations, etc.

- Segmentation is still a problem, since the definition of an object can never make clear to computers.
- Object boundaries always merge with the background, etc.

1.3 Advanced Video Concepts - from HVC to Advanced Video Coding (H.264)
1. **Background:**

   Hybrid Coding, or Sprite Generation Technique

2. **Foreground:**

   Multiple objects
   
   Hybrid Coding
   
   Object based coding (complete object)
   
   Time, position, motion manipulations, etc.

**Techniques developed:**

1. **Motion estimation/Sprite Generation**

   Our own fast hybrid Coding making use new concepts of fast motion estimation, and sprite generation techniques.
Techniques developed (con’t):

2. Improved **Automatic Image Segmentation**
   Making use a new marker-extraction technique and color information, simplified area morphology and modified watershed algorithm.

3. **Fast Wavelet** Computation
   Using fast lifting algorithm, possibility to use overcompete wavelets.

4. Etc.

---


---

A few Published Works:

1. A few **adaptive motion estimation** algorithms proposed, which make use of simple statistics to determine the search directions and locations; some **pioneer work** obtained very **good citations**.

2. Worked on fast algorithms with
   (i) with a selected sub-set of pattern(s), and
   (ii) with pixel adaptive **pixel decimation**.

References:


Yui-Lam Chan and Wan-Chi Siu, ‘Reliable Block Motion Estimation through the Confidence Measure of Error Surface’, pp.133-48, Vol.76, issue 2, **Signal Processing**, 1999, Switzerland

3. Recently suggested the concept of error clustering, which gives a completely revised concept on adaptive motion estimation. This is able to replace the PDS, and working Successive Elimination Algorithm (SEA) or Multilevel SEA for extremely fast full search motion estimation.

4. Recently we suggested:
   
   (i) to use a search window being equal to the size of Motion Vector(s) in the 1st for multi-frame motion estimation.
   
   (ii) to use partial SAD for variable block sizes motion estimation.
   
   (iii) to use directional search to form a novel scheme, etc.

References:

3. Video Transcoding

Video Transcoding: Given a variety of client devices, it is difficult for a server to tailor the content for individual devices. A video server may have to provide quality support services to heterogeneous clients or transmission channels.

It is in this reason that the video server should have the capability to perform transcoding: a process of converting a previously compressed video bitstream into a bit stream of different nature or lower bitrate.
Homogeneous Transcoding – three types:

1. Frame Skipping
2. Video Downscaling
3. Transcoding with Bit-rate Reduction

Heterogeneous Transcoding:
4. Conversion of videos among standards (or between frame types)

Conventional Transcoder:

VLD: Variable Length Decoding
VLC: Variable Length Coding
Q1: Inverse Quantization (Fine Quantizer)
Q2: Quantization (Q2 Coarse Quantizer)
MCF: Motion Compensated Frame
DCT: Discrete Cosine Transform
DCT⁻¹: Inverse Discrete Cosine Transform
EMV: Encoding Motion Vector
MC: Motion Compensation
**Frame-Skipping Transcoder**

When the frame rate changes, the incoming quantized DCT coefficients of residual signal may no longer be valid because they refer to a frame which may have been dropped.

First, the transcoder decodes the incoming bitstream in the pixel domain. Second, the decoded video frame is then re-encoded at the desired lower frame rate.

To look at the decoding and re-encoding parts alone

- First, the video bitstream performs VLC decoding, inverse quantization and inverse DCT. So, frame $R_{t-1}$ can be reconstructed and stored in buffer FB. Note that $R_{t-1}$ is required to act as the reference frame for the reconstruction of frame $R_t$. Hence we have

$$ R_t(i, j) = R_{t-1}(i + u, j + v) + \hat{e}_t(i, j) $$

where $\hat{e}_t$ is the prediction error (residual signal).
Let us assume that $R_{t-1}$ be dropped.

- If $R_{t-1}$ is dropped, we have to find frame $R_t$ at time $t$, with reference to the previous non-skipped frame at time $t-2$, i.e. $R_{t-2}$.
- New compensation error, $e^s(i,j)$ has to be found.

**Direction Addition Approach:**

Let us consider a special case - Macroblocks without motion compensation.

Recall that $R_{t-1}$ is dropped; we can use the motion vectors $(u_t, v_t)$ and $(u_{t-1}, v_{t-1})$ to reconstruct the new motion vector $(u'_t, v'_t)$. Now we need to find $Q[DCT(u'_t)]$.
Macroblocks without motion compensation

\[ MB_i = MB_{i-1} + \hat{e}_i \]
\[ MB_{i+1} = MB_{i-2} + \hat{e}_{i-1} \]
\[ e_i' = MB_i - MB_{i-2} = \hat{e}_i + \hat{e}_{i-1} \]
\[ DCT(e'_i) = DCT(\hat{e}_i) + DCT(\hat{e}_{i-1}) \]
\[ Q[DCT(e'_i)] = Q[DCT(\hat{e}_i)] + Q[DCT(\hat{e}_{i-1})] \]

Note that \( Q[DCT(\hat{e}_i)] \) and \( Q[DCT(\hat{e}_{i-1})] \) are available from the incoming bitstream.

A Direct Addition of the DCT Coefficients: newly quantized DCT coefficients can be computed in the DCT-domain by adding directly the quantized DCT coefficients between the data in the DCT-domain buffer and the incoming DCT coefficients, whilst the updated DCT coefficients are stored in the DCT-domain buffer.
Direct Addition of the DCT Coefficients

- It is not necessary to perform the motion compensation, DCT, quantization, inverse DCT and inverse quantization — the complexity is greatly reduced.
- Requantization is not necessary for macroblocks coded without motion compensation — the quality degradation due to re-encoding of the transcoder is avoided.

By using a direct addition of the DCT coefficients for non-moving macroblocks, the computational complexity involved in processing these macroblocks can be reduced significantly and the additional re-encoding error can be avoided.

References:


Video Transcoding: Sample Study 2– Transcoding the H.263 to the H.264 within the Transform Domain

Why transcoding from H.263 to H.264?

The complete migration to the new video coding algorithm will take several years since H.263 and MPEG are widely used in many multimedia applications nowadays.

This creates an important need for transcoding technologies that convert the widely available H.263 compressed videos to H.264 compressed format and vice versa. However, given the significant differences between the H.263 and the H.264 algorithms, transcoding is much more complex.

References:


Sample Study 3: H.264 → H.264
- Architecture of a Down-sizing Transcoder -

*E.g. for H.264: High profile from HD to SD*

We have to convert a video of HD (1920 x 1080) format to SD (1280 x 720) format.

For a macroblock:

1. To determine its mode type: Intra or inter
2. To determine its prediction mode for intra-mode
3. To determine its mode (VBS) for inter-mode
4. Check if skip mode to be used
5. Motion re-estimation.

HDTV: HD 1920×1080

SD 1280×720

3 → 2
Transcoding
HDTV: Video Transcoding

Transcoding from HD and SD formats:

Procedure:
(i) timing Analysis,
(ii) data extraction from the codec,
(iii) building an ideal speed-up model for transcoding in the H.264 platform (architecture realization) and
(iv) video transcoder refinement using various technologies:

Technologies: (Algorithms have to be designed for )
(1) inter/intra re-decision
(2) intra mode re-decision (I16x16 or I4x4)
(3) inter mode re-prediction
(4) motion vector re-estimation
Study of the average encoding speed of the JM12.2 encoder

Sequence name: CrowdRun_720p

No. of frames: 500

Intra-frame period: 0
No. of slices per frame: 1
QP-I, QP-P: 30
QP-B: 32
Inter-block-sizes used: 16*16, 16*8, 8*16, 8*8
Intra-block-sizes used: 16*16, 8*8, 4*4
Max search range: ±128
Number of reference frames: 1
Sub-pixel depth: Quarter-pixel
Entropy coding method: CAVLC
Special features applied: Weighted prediction, skip and direct coding modes, 8x8 integer transform, deblocking filter
Encoder compiled by: VC6.0

No. of B-frames = 0

Total time: 1770.327 sec
Reading frames time: 11.927 sec (0.67%)
Padding reference frames time: 0.000 sec (0.00%)
Integer ME time (EPZS): 156.196 sec (8.82%)
Sub-pel ME time (1/2 & 1/4 pel): 277.982 sec (15.70%)
Interpolation time: 88.747 sec (5.01%)
Getting MVs for direct mode time: 0.000 sec (0.00%)
Weighted prediction time: 1.918 sec (0.11%)
Intra prediction time: 657.611 sec (37.15%)
Computing distortion values for modes time: 28.422 sec (1.61%)
Computing rate values for modes time: 185.476 sec (10.48%)
Luma residue coding time: 95.462 sec (5.39%)
Chroma residue coding time: 139.090 sec (7.86%)
Setting parameters time: 7.544 sec (0.43%)
Entropy coding time: 10.229 sec (0.58%)
Deblocking time: 9.499 sec (0.54%)
Other time: 100.224 sec (5.66%)

PSNR: 33.43 dB
Bit-rate: 22625.70 kbps@50Hz
No. of B-frames = 2

<table>
<thead>
<tr>
<th>Activity</th>
<th>Time</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total time</td>
<td>2649.32 sec</td>
<td></td>
</tr>
<tr>
<td>Reading frames time</td>
<td>16.621 sec</td>
<td>(0.63%)</td>
</tr>
<tr>
<td>Padding reference frames time</td>
<td>0.000 sec</td>
<td>(0.00%)</td>
</tr>
<tr>
<td>Integer ME time (EPZS)</td>
<td>532.862 sec</td>
<td>(20.08%)</td>
</tr>
<tr>
<td>Sub-pel ME time (1/2 &amp; 1/4 pel)</td>
<td>463.725 sec</td>
<td>(17.50%)</td>
</tr>
<tr>
<td>Interpolation time</td>
<td>29.039 sec</td>
<td>(1.10%)</td>
</tr>
<tr>
<td>Getting MVs for direct mode time</td>
<td>1.854 sec</td>
<td>(0.070%)</td>
</tr>
<tr>
<td>Weighted prediction time</td>
<td>3.054 sec</td>
<td>(0.12%)</td>
</tr>
<tr>
<td>Inter prediction time</td>
<td>628.926 sec</td>
<td>(23.74%)</td>
</tr>
<tr>
<td>Computing distortion values for modes time</td>
<td>43.832 sec</td>
<td>(1.65%)</td>
</tr>
<tr>
<td>Computing rate values for modes time</td>
<td>237.783 sec</td>
<td>(8.98%)</td>
</tr>
<tr>
<td>Luma residue coding time</td>
<td>190.330 sec</td>
<td>(7.18%)</td>
</tr>
<tr>
<td>Chroma residue coding time</td>
<td>221.465 sec</td>
<td>(8.36%)</td>
</tr>
<tr>
<td>Setting parameters time</td>
<td>7.975 sec</td>
<td>(0.30%)</td>
</tr>
<tr>
<td>Entropy coding time</td>
<td>8.078 sec</td>
<td>(0.30%)</td>
</tr>
<tr>
<td>Deblocking time</td>
<td>9.350 sec</td>
<td>(0.35%)</td>
</tr>
<tr>
<td>Other time</td>
<td>255.226 sec</td>
<td>(9.63%)</td>
</tr>
</tbody>
</table>

PSNR: 32.74 dB  Bit-rate: 20901.61 kbps@50Hz

JM 12.2 encoder timing
QP(I,P,B) = 27, 28, 29  720P  250 Frames  search range = ±32  All modes turned on

<table>
<thead>
<tr>
<th></th>
<th>16x16 only</th>
<th>EPZS (Extended diamond pattern)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Total time</td>
<td>Integer ME time</td>
</tr>
<tr>
<td>crowd run</td>
<td>2185.02</td>
<td>1613.98 (73.87%)</td>
</tr>
<tr>
<td>ducks take off</td>
<td>2203.40</td>
<td>1617.56 (73.41%)</td>
</tr>
<tr>
<td></td>
<td>Average</td>
<td>73.64% (1.39%)</td>
</tr>
<tr>
<td></td>
<td>16x16, 16x8, 8x16, 8x8</td>
<td></td>
</tr>
<tr>
<td>crowd run</td>
<td>2573.35</td>
<td>1719.91 (66.75%)</td>
</tr>
<tr>
<td>ducks take off</td>
<td>2655.36</td>
<td>1719.91 (65.76%)</td>
</tr>
<tr>
<td></td>
<td>Average</td>
<td>66.26% (5.58%)</td>
</tr>
</tbody>
</table>
Video transcoding Technologies:

Key Technologies:

1. **Inter/Intra mode Decision** (I, P, SKIP, etc. by block with fixed location, or simple majority)
2. **Inter-block modes** re-decision (16x16, 16x8, …4x4)
   - (a) Natural reduction, using majority etc.
   - (b) Further mode selection with better quality, such as refinement
3. **Intra Prediction Modes** (differential code, vertical, horizontal, …, diagonal.. Prediction)
4. **Interpolations** for sub-pixel interpolation: integer decimation
5. **Motion Vector Re-estimation**
   - (a) MV reuse using original MV (as far as possible)
   - (b) MV reuse using, mean, median, align to the best, align to the worst, weighed residual error signal...
   - (c) MV refinement using temporal and spatial records
   - (d) MV reuse and residual error signal reuse
   - (d) sub-pixel motion re-estimation
6. **Video Down Sizing and Interpolation**
   - (a) Interpolations for downsizing 2→1, 3→2, 1→M/N, fixed ration interpolation,
   - (b) and then variable ratio interpolation
   - (c) Quality interpolation, for example edge-preserving interpolation.
7. **Transform domain** video transcoding

Samples of Experimental Results

Table 2 shows the results of our realization of the transcoding results using the H.264 JM12.2 and using our fast approaches for converting the Crowd Run of size 1280x720 to 2/3 of this size. It is seen that there is a substantial reduction in computation time for motion estimation, mode decision, and etc. and a speedup of 2.6 time is achieved.

<table>
<thead>
<tr>
<th></th>
<th>JM 12.2</th>
<th>After fast algorithms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer-pel ME</td>
<td>153.98s (20.18%)</td>
<td>20.50s (7.21%)</td>
</tr>
<tr>
<td>Sub-pel ME</td>
<td>227.23s (29.78%)</td>
<td>30.25s (10.64%)</td>
</tr>
<tr>
<td>Other ME time</td>
<td>39.28s (5.15%)</td>
<td>5.23s (1.84%)</td>
</tr>
<tr>
<td>Intra prediction</td>
<td>131.78s (17.27%)</td>
<td>17.54s (6.17%)</td>
</tr>
<tr>
<td>Others</td>
<td>210.80s (27.63%)</td>
<td>211.01s (74.14%)</td>
</tr>
<tr>
<td>Total time</td>
<td>763.07s</td>
<td>284.32s (2.68X)</td>
</tr>
</tbody>
</table>

Table 2: Comparison of results using JM12.2 and our fast approach
Transcoder Demonstration

On video Coding for HDTV using H.264 Standard

To convert HD (1920 x 1080) format to SD (1280 x 720) format

(Real-time demonstration has bee done, but here is a reduced version, due the speed constraint of the Labtop computer.)
4. Extending to Video Amplifications and Super-resolution videos

With the development of visual communication and image processing, there is a high demand for high-resolution images such as video surveillance, remote sensing, medical imaging, HDTV and other entertainment applications.

However, image resolution depends on the physical characteristics of the imaging devices. It is sometimes difficult to improve the image resolution by using better sensors because of the high cost or hardware physical limits.

Super-resolution (SR) image reconstruction is a promising technique to increase the resolution of an image or sequence of images beyond the resolving power of the imaging system.
A SR video may also require to be re-encoded for various reasons, including

(i) to allow standard devices to view the SR videos without using additional conversion devices,
(ii) to save SR video reconstruction time since the computing power of the viewing devices may not be sufficient and
(iii) to avoid the unavailability of the SR video package at the viewing site.

It is also true that broadcasting companies are looking for good technologies to covert videos between formats, different resolutions, and frame/bit rates.

It is particularly difficult to do up-conversion of a compressed video, say for example from SDTV to HDTV, due to the missing data and blurring effect of edges by simple interpolation.

Furthermore re-encoding of these SR videos is required in many practical situations, since contents providers often have to standardize various video clips for uniform storage or transmission.

![Figure 5: Architecture of Transcoding Platform](Video Enlargement)
In the previous years, most researchers, including us, just concentrated on downward conversion. Recently it is clear to us that there is a great need to develop techniques for upward conversion, including image/video up-sizing, frame interpolation, and super-video coding.

This is a challenging topic but difficult one. Some works have been done by few researchers, but many technologies are still unavailable or pre-mature. Hence this forms a fruitful direction for further research.

We have built an architecture which allows us to re-encode the SR video for either storage or transmission. We fully utilized the decoded data, statistics and parameters available from the previously encoded LR video to facilitate the super-resolution conversion.

As shown in fig.5, a model has to be built for this investigation. The H.264 is our codec kernel.

The model consists of three parts

(a) “encoded bit-stream” decoding,
(b) video interpolation and
(c) re-encoding.

We opt for a simple framework as shown in fig.5, while many fundamental technologies are desperately needed for its practical realization.
(i) The interpolation is done initially within the decoded LR video frame without considering information from the temporal direction.

The re-encoding part is done simply using the H.264 encoder, which requires relatively long encoding time. Fig. 6 shows the result of a preliminary test on converting the “Rush Hour” sequence from the SD(1280x720) format to HD(1920x1080) format in the high profile of the H.264.

The upper curve shows the quality and bit-rate of using fully decoding and encoding, with simple linear interpolation for magnification.

The lowest curve shows the production of the compressed HR video by the simplest and quickest approach. In this approach we made use of the decoded motion vectors, decoded prediction modes, decoded mode sizes, etc. of the LR frame for the re-coding. This is done by some default arrangements. No motion estimation, no mode decision, etc. were required. It is about three times or more faster than that using the fully re-encoding mode, but it suffers from low PSNR and high bit rate.

The middle curve shows a hypothetically case. This gives the best possible result that can be achieved if we do not perform full motion estimation, mode decision, etc. while the best parameters (MV, modes, etc.) were picked from the list of parameters decoded from the LR frame. This forms the target for fast algorithm development.

Figure 6: Video Enlargement
(ii) A key part of this work is to design fast and accurate algorithms for obtaining encoding modes, motion vectors, or even transform coefficients without going through the heavy computational processes. The process is surprisingly close to downsize transcoding. We have to do

1. inter/inter mode re-decision,
2. intra mode re-decision (16x16 or 4x4),
3. inter mode re-prediction,
4. motion vector re-estimation, etc.

The data and parameters available in originally encoded LR video are used to formulate the fast algorithms.

(ii) The following strategies are used.

(a) Higher weights should be given to parameters with larger areas.

(b) All modes/MV (from LR frames) with the areas of LR blocks overlapping with the SR block should have a good priority to be checked.

(c) The number of zero coefficients should be able to reflect the motion activities of the block.

(d) Treat cases with different QP differently.

(e) Refinement are made according to models built.
A: Interpolation Techniques: In order to remove the blurring effect, edge enhancement is one of the best ways to improve the quality of a super-resolution image/video sequence.

We propose an improved edge-directed interpolation method by removing the accumulated interpolation error, and reducing correlation structure mismatch problem. Let us recall the transfer function of a Wiener filter,

\[ Y(k) = \sum_{n=0}^{\infty} \alpha(n) x(k-n) \]

where \( Y(k) \) is the predicted value, \( \alpha(k)'s \) are the linear prediction coefficients and \( x(n)'s \) are known samples. By optimizing the mean square error, MSE \( (=E[e^2(n)]) \), we can come up with an equation for finding the coefficients of the Wiener filter for the interpolation,

\[ r_{dx} = \alpha R_{xx} \quad (1) \]

where \( R_{xx} (=E[x(n)x(n-i)]) \) is an autocorrelation function and \( R_{dx} (=E[x(n)x(n-i)]) \) is a cross-correlation function.

The New Edge-Directed Interpolation (NEDI) scheme is to model a natural image as a second-order locally stationary Gaussian process which allows the interpolation using a simple linear prediction. The covariance of the image pixels in a local block (training window) can be used to obtain the prediction coefficients of the estimation problem.

Consider the interpolation of an image \( X \) to a high-resolution image \( Y \).

![Figure 7: New Edge-Directed Interpolation (NEDI)](47)
In fig. 7, the numbers are used to represent the locations of the original low resolution pixel points.

The solid point, entitled as \( y_i \), as shown in fig.7(a) is a high resolution point to be interpolated from four neighbor low-resolution pixels \( \{x_{18}, x_{19}, x_{26}, x_{27}\} \).

In order to have the simplest formulations, one-D representation has been used as far as possible for explanation.

The predicted pixel becomes,

\[
y'_i = \sum \sum \alpha_r x_i = \sum \alpha_r x_{i_j} \quad \text{selected surrounding pixels}
\]

From eqn.1, we have

\[
\alpha = R^{-1}_{xx} r_{dx}
\]

The computation of \( r_{ds} \) (cross-correlation between \( y_i \) and it’s interpolating points) and \( R_{xx} \) (the auto-correlation among interpolating points) would require knowledge of statistics of \( y_i \) with its neighbors which are not available before the interpolation.

This difficulty is overcome by the “geometric duality” property, as illustrated fig.7(b).

The correlations between \( y_i \) in the high resolution domain and its neighbors points, 18, 19, 26 and 27 are replaced by the correlations of four sets of sample (training) points as enclosed by dotted lines as shown in fig.7(b).
For example the statistics are available for interpolating point 18 from its neighbors, points 9, 11, 25 and 27 in the low resolution (LR) domain. Hence we can write

\[
y = \begin{bmatrix} x_{18} \\ x_{19} \\ x_{26} \\ x_{27} \end{bmatrix} \quad \text{and} \quad C = \begin{bmatrix} x_9 & x_{11} & x_{25} & x_{27} \\ x_{10} & x_{12} & x_{26} & x_{28} \\ x_{17} & x_{19} & x_{33} & x_{35} \\ x_{18} & x_{20} & x_{34} & x_{36} \end{bmatrix}
\]

where elements of \( y \) are the training points and the row of \( C \) are the set of respective points to interpolate elements of \( y \). In this case we have

\[
r_{dx} = C^T y \quad \text{and} \quad R_{xx} = C^T C.
\]

To interpolate a point between two vertical LR pixels (2nd step), the same procedure is used with a rotation by an angle \( \pi/4 \) as shown in figs.8(a) and (b).

In fig.8, circles represent LR pixels and grey dots represent the interpolated points in the 1st step (fig.7) and small black dots represent HR points to be interpolated.

To save computation, the NEDI adopted a hybrid approach, this correlation based interpolation is applied to edge pixels only and bilinear interpolation is applied to non-edge pixels (i.e. pixels in smooth regions).
However, the NEDI suffers from the **prediction error propagation problem** which limits the performance of the algorithm.

NEDI is a two-step interpolation scheme, where the first step makes use of the original pixels for interpolation, **whilst the second step makes use of the interpolation results obtained from the first step**, i.e. gray pixels in Fig.8 to obtain the interpolation pixel (the small black dot).

The interpolation error in the first step will be propagated to the second interpolation step, and thus causes the interpolation error propagation problem.

At the same time, NEDI also suffers from covariance structure **miss-match problem**. The span of pixels does not represent the best coverage in the HR domain.

---

**Figure 8:** Modified 2nd step, (a) Interpolation problem, (b) original training set, (c) and (d) proposed training sets.
Hence a different set of pixels could give a better interpolation of the edges. We resolve the problem by suggesting a new version.

The first step is the same as before. In the second step, we propose to interpolate the unknown pixels by a sixth-order linear prediction with a training window as shown in figs.8(c) and (d) by using points on the original LR domain only.

This completely eliminates the error propagation problem. To reduce the covariance miss-match problem, we may use multiple low-resolution training window candidates, i.e. a scheme to choose one from more than one low-resolution training windows to represent the covariance of the high-resolution block to perform the linear prediction, as shown in fig.8(d).

Statistics to be used

Figure 5: Suggested Enhanced NEDI.

References:
Figs. 9 and 10 show that results of our approach on interpolation for the enlargement of an image and simulated SR video reconstruction. The reader may note the bar and connection parts above the wheel of fig.10, which look **more smooth and sharper**. The effect is more effective if we use some further level of amplifications.

(a) Using original Step 2  
(b) Using new Step 2

Figure 9: Preliminary results of the proposed new approach for edge enhancement

Figure 10: SR video by simulation, Top right: original video frame, bottom left: by linear interpolation by Intel lib, bottom right: SR video with accurate MVs.
Experimental Results – Original Images

Jet Plane

Bicycle

Experimental Results – test image: Jet Plane

Bilinear
(PSNR=28.98dB)

NEDI
(PSNR=32.47dB)

MEDI
(PSNR=32.34dB)

Final Image

Error Image*

* Intensity is scaled between range 0 to 255
Experimental Results – test image: bicycle

<table>
<thead>
<tr>
<th>Method</th>
<th>PSNR (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bilinear</td>
<td>18.68</td>
</tr>
<tr>
<td>NEDI</td>
<td>20.89</td>
</tr>
<tr>
<td>MEDI (Our)</td>
<td>20.67</td>
</tr>
</tbody>
</table>

Final Image

Error Image*

* Intensity is scaled between range 0 to 255

B. Super-Resolution Video: Since the interpolation from a frame to form an enlarged frame is restricted by the resolution and information available from the original image, it is very natural to use more frames (both in temporal and spatial domains) to construct the enlarge frame.

An enlarged frame obtained from more then one orginal frame is defined as a super-resolution frame (video) in this paper.

This can be achieved by both non-iterative and iterative approach.

Due to the limitation in space, let us not to discuss the details of our approach, but code some experimental as shown in fig.9. Interested reader may refer to the literature for further information.
Super-resolution Images/Videos

Super-resolution (SR) image/video reconstruction (SR) is a promising technique to increase the resolution of an image or sequence of images (video) beyond the resolving power of the imaging system.

Modified definition of Transcoding: A process of converting a previously compressed video bitstream into a bit stream of different nature or lower/higher bit-rate.

Figure 1: Video Interpolation
Reasons for Re-encoding:

A SR video may also require to be re-encoded for various reasons, including

(i) to allow *standard devices to view the SR videos* without using additional conversion devices,

(ii) to save SR video reconstruction time since the *computing power of the viewing devices* may not be sufficient and

(iii) to avoid the *unavailability of the SR video package* at the viewing site.

Reasons for Re-encoding (con’t):

It is also true that broadcasting companies are looking for good technologies to covert videos between formats, different resolutions, and frame/bit rates.

It is particularly difficult to do up-conversion of a compressed video, say for example from SDTV to HDTV, due to the *missing data and blurring effect of edges* by *simple interpolation*.

Furthermore re-encoding of these SR videos are required in many practical situations, since *contents providers* often have to standardize various video clips for uniform storage or transmission. The viewer side may not have the conversion module or computing power for real-time SR reconstruction.
Key Technologies on the study of in Super-resolution (SR) videos:
1. **Image Interpolation**, hence

2. **Video Interpolation**
   - blurring effect
   - aliasing effect
   - edge enhancement techniques *(new technique is available)*
   - iterative and non-iterative approaches *(non-itera.. is simple)*
   - noise reduction technique *(new technique is available)*

3. Spatial domain super-resolution videos using multi-images

4. Temporal domain super-resolution videos using **temporal features** *(using further information from video frames)*

5. SR video from **encoded** video frames

6. SR video **re-encoding** using lower-resolution compressed videos *(general kernel suggested here, further technologies required)*

---

**Block diagram for re-encoding SR Videos**

The incoming bitstream is decoded into the pixel domain. The decoded video frame is magnified and subsequently re-encoded at the desired frame rate.

First, the transcoder decodes the incoming bitstream in the pixel domain. Second, the decoded video frame **magnified with super-resolution techniques**, and then re-encoded at the desired frame-rate.
Framework of the Architecture of Super-Resolution Transcoding

E.g. for H.264-High profile: from SD to HD
For a macroblock:
1. To determine its mode type: Intra or inter
2. To determine its prediction mode for intra-mode
3. To determine its mode (VBS) for inter-mode
4. Check if skip mode to be used
5. Motion re-estimation.

Ideal Case
HDTV: Video Upsizing with SR techniques

Transcoding from SD and HD formats

(i) Compressed video decoding: also decoding the original motion vector, modes, residual error value and statistics available.

(ii) Super-resolution video formation:
- mosaicing using multi-frames from the video
- Simple linear interpolation (for missing point)
- Edge enhancement (edge detection and edge-directed interpolation, etc.)
- Noise removal and/or deblocking, etc.

(ii) SR video Re-encoding:
1. inter/intra re-decision
2. intra mode re-decision (I16x16 or I4x4)
3. inter mode re-prediction
4. motion vector re-estimation
etc.

Details
Super-Resolution Video Kernel Demonstration:

Converting HDTV video with H.264 Standard
from SD (1280 x 720) format
to HD (1920 x 1080) format

• Fully Decode + Upsize + Fully Encode

• Transcoding
  – (Fully Decode + Upsize + Fast Encoding
    using Mode Re-Decision + MV
    Refinement + etc.)

Ideal Super-resolution Video:

Original Frame

Simulated four LR images: LR0, LR1, LR2, LR3 with size of 172*144
Original Frame

Left: Linear Interpolation by Intel Library
Right: SR video with accurate MVs
Conclusion:

1. We started with the Hybrid Video Coding model, and gradually moved on to Object-oriented Video Coding.

2. The Advanced Video Coding (H.264) includes almost no new concepts, except that it fine trims existing techniques in a systematic way to optimize the coding efficiency. This can reduce the bitrate to half of that of the MPEG-2 standard.

3. Can we squeeze further that the bitrate be improved by one more time? Some researchers go back to the object oriented coding, whilst others continue with the optimization or move to other sophisticated applications, such as multi-view video coding or advanced scalable coding.

4. Motion Estimation is an important topic. We have done much work on it, but did not talk too much about it in this presentation. Can we have a Fast ME algorithm which gives better quality as compared with Exhaustive Full Search Algorithm?

6. We then talked about transcoding, which is a process to convert an encoded video from one format to another format. Our work involves both 
   (i) heterogeneous transcoding, such as from H.263 to H.264 and
   (ii) homogeneous transcoding, such as from H.264 to H.264.

   H.264 to H.264 transcoding: in the high profile,
   from SD to HD (Not difficult, but good quality is difficult.)
   from HD to SD (why?) Would pixel interpolation be important?
   Mode Type re-decision: Intra mode
   Inter mode (including skip mode, etc.)
   Intra Mode: 4x4 or 16x16? What is the prediction mode?
   Inter Mode: Mode decision (horizontal, vertical, …)
   Motion vector re-prediction

7. We then talk about the significance of super-resolution video and some of it related techniques:
   Simple interpolation, and the new edge-directed interpolation and our modified edge-directed interpolation ..., also a complete Kernel structure.

8. A brief highlight of our work being carried out has also been given. These include (i) to look for practical ways to form SR videos and
   (ii) the re-encoding of SR videos
The End:

thank you!