Video Encoder Based on Lifting Transforms on Graphs

Eduardo Martínez Enríquez $^1$, Fernando Díaz de María $^1$ and Antonio Ortega $^2$

$^1$Department of Signal Theory and Communications
Universidad Carlos III de Madrid, Leganés, Madrid, Spain

$^2$Department of Electrical Engineering
University of Southern California, Los Angeles, California, USA

ICIP 2011
September, 2011
Outline

1. Introduction

2. Lifting transforms on graphs for video coding
   - Overview
   - Low Complexity Transform Using Subgraphs
   - Coefficient Reordering

3. Experimental Results

4. Conclusions and Future lines
Outline

1. Introduction

2. Lifting transforms on graphs for video coding
   - Overview
   - Low Complexity Transform Using Subgraphs
   - Coefficient Reordering

3. Experimental Results

4. Conclusions and Future lines
Motivation: Video encoder based on directional transforms: Avoid filtering across large discontinuities.

Related Work:

- Image Coding:
  - [Velisavljevic et al., 2006, Le Pennec and Mallat, 2005].
  - Lifting in the spatial domain for image coding [Shen and Ortega, 2008, Fattal, 2009].


Nevertheless... these transforms filter in the spatial or in the temporal domain independently.

Our new transform is able to filter following the spatio-temporal directions of high correlation... How do we do this?
Key Novelties

- **Introduction to the transform:**
  - Describe the video sequence as a graph of connected pixels.
  - Link pixels:

![Diagram showing linked pixels with numbers and arrows]
Key Novelties

- **Introduction to the transform:**
  - Describe the video sequence as a graph of connected pixels.
  - Link pixels: Temporal domain
**Key Novelties**

- **Introduction to the transform:**
  - Describe the video sequence as a graph of connected pixels.
  - Link pixels: Temporal domain, Spatial domain

![Diagram](Image)
Key Novelties

- **Introduction to the transform:**
  - Describe the video sequence as a graph of connected pixels.
  - Link pixels: Temporal domain, Spatial domain $\rightarrow$ Graph.
Key Novelties

- **Introduction to the transform:**
  - Describe the video sequence as a graph of connected pixels.
  - Link pixels: Temporal domain, Spatial domain → Graph.
  - Weight links: Expected correlation between pixels intensity (reliability).

---

Video Encoder Based on Lifting Transforms on Graphs
Key Novelties

**Introduction to the transform:**
- Describe the video sequence as a graph of connected pixels.
  - Link pixels: Temporal domain, Spatial domain $\rightarrow$ Graph.
- Weight links: Expected correlation between pixels intensity (reliability).
- Apply lifting transform on this graph.
**Key Novelties**

- **Introduction to the transform:**
  - Describe the video sequence as a graph of connected pixels.
  - Link pixels: Temporal domain, Spatial domain → Graph.
  - Weight links: Expected correlation between pixels intensity (reliability).
  - Apply lifting transform on this graph.

- **Key Novelties:**
  - Spatio-temporal lifting → Non separable approach, against common Wavelet-based video coders (t+s).
  - Coefficients reordering using graph information at the decoder.
Objectives

We aim to:

- Design a **new directional transform** for video coding.
- Describe a **low complexity version** of the encoder working with subgraphs.
- Propose a **coefficient reordering method** using the graph information.
1 Introduction

2 Lifting transforms on graphs for video coding
   - Overview
     - Low Complexity Transform Using Subgraphs
     - Coefficient Reordering

3 Experimental Results

4 Conclusions and Future lines
Lifting Transforms on Graphs

- To perform the transform and ensure its invertibility:
  - Split the input data in Prediction ($P$) and Update ($U$) disjoint sets.
  - Design the predict ($p$) and update ($u$) filters. ¹

¹See [Martínez Enríquez and Ortega, 2011]
To perform the transform and ensure its invertibility:
- Split the input data in Prediction ($P$) and Update ($U$) disjoint sets.
- Design the predict ($p$) and update ($u$) filters. 

See [Martínez Enríquez and Ortega, 2011]
To perform the transform and ensure its invertibility:
- Split the input data in Prediction (P) and Update (U) disjoint sets.
- Design the predict (p) and update (u) filters. ¹

¹See [Martínez Enríquez and Ortega, 2011]
To perform the transform and ensure its invertibility:
- Split the input data in Prediction ($P$) and Update ($U$) disjoint sets.
- Design the predict ($p$) and update ($u$) filters. ¹

¹See [Martínez Enríquez and Ortega, 2011]
To perform the transform and ensure its invertibility:
- Split the input data in Prediction ($P$) and Update ($U$) disjoint sets.
- Design the predict ($p$) and update ($u$) filters. ¹

¹See [Martínez Enríquez and Ortega, 2011]
To perform the transform and ensure its invertibility:
- Split the input data in Prediction ($P$) and Update ($U$) disjoint sets.
- Design the predict ($p$) and update ($u$) filters. ¹

¹See [Martínez Enríquez and Ortega, 2011]
To perform the transform and ensure its invertibility:

- Split the input data in Prediction ($P$) and Update ($U$) disjoint sets.
- Design the predict ($p$) and update ($u$) filters. \(^1\)

\(^1\)See [Martínez Enríquez and Ortega, 2011]
Lifting Transforms on Graphs

To perform the transform and ensure its invertibility:

- Split the input data in Prediction ($P$) and Update ($U$) disjoint sets.
- Design the predict ($p$) and update ($u$) filters.  

\[\text{See [Martínez Enríquez and Ortega, 2011]}\]
To perform the transform and ensure its invertibility:
- Split the input data in Prediction (P) and Update (U) disjoint sets.
- Design the predict (p) and update (u) filters. ¹

¹See [Martínez Enríquez and Ortega, 2011]
Update-Prediction Splitting

- **Correct splittings:** Invertibility for any disjoint splitting..., but the coding efficiency depends on the criterion used to do it!

- **Criterion we use:** Maximize the reliability with which update nodes can predict prediction neighbors → Maximize the total weight of the links between $P$ and $U$ sets (Max. cut) → Algorithm [Hsu, 1983].
**Update-Prediction Splitting**

- **Correct splittings:** Invertibility for any disjoint splitting..., but the coding efficiency depends on the criterion used to do it!

- **Criterion we use:** Maximize the reliability with which update nodes can predict prediction neighbors → Maximize the total weight of the links between $P$ and $U$ sets (Max. cut) → Algorithm [Hsu, 1983].
**Encoder and Decoder Data flow**

Side information: Motion Vectors, Edge map (once every $F$ frames).\(^2\)

\(^2\)See [Martínez Enríquez and Ortega, 2011]
Outline

1 Introduction

2 Lifting transforms on graphs for video coding
   - Overview
   - Low Complexity Transform Using Subgraphs
   - Coefficient Reordering

3 Experimental Results

4 Conclusions and Future lines
Low Complexity Transform

**Problem**  The encoder complexity increases rapidly with the number of nodes $D$.

- e.g., The greedy update-prediction assignment algorithm $\rightarrow O(D^3 \log D)$.

**Goal**  Operate on subgraphs of the original graph.

- Divide the original graph node set in $K$ subsets $A_k$.
- In any of the subgraphs $S_k (A_k, E_k) \rightarrow$ Invertible and critically sampled transform.

**Necessary Condition**  $A_k$ node subsets have to be disjoint.

**Proposed Solution**

- Create subgraphs based on disjoint subsets.
- Keep the temporal links (more reliable) in $F$ temporal hops.
Algorithm

1. Divide each frame into blocks of size $P \times Q$. 
Algorithm

1. Divide each frame into blocks of size $P \times Q$. 
Algorithm

1. Divide each frame into blocks of size $P \times Q$.
2. Perform the motion estimation for the $F$ frames.
Algorithm

1. Divide each frame into blocks of size $P \times Q$.
2. Perform the motion estimation for the $F$ frames.
Algorithm

1. Divide each frame into blocks of size $P \times Q$.
2. Perform the motion estimation for the $F$ frames.
3. Generate a tree.
Algorithm

1. Divide each frame into blocks of size $P \times Q$.
2. Perform the motion estimation for the $F$ frames.
3. Generate a tree.

Video Encoder Based on Lifting Transforms on Graphs
Algorithm

1. Divide each frame into blocks of size $P \times Q$.
2. Perform the motion estimation for the $F$ frames.
3. Generate a tree.
4. Subgraph $\rightarrow$ pixels that belong to the blocks that are linked along the $F$ frames.

Video Encoder Based on Lifting Transforms on Graphs
Algorithm

1. Divide each frame into blocks of size $P \times Q$.
2. Perform the motion estimation for the $F$ frames.
3. Generate a tree.
4. Subgraph $\rightarrow$ **pixels** that belong to the blocks that are linked along the $F$ frames.
Subgraph Approach Performance

Test Conditions

- 20 Frames, QCIF sequences
- Complexity Reduction (CR) = \( \frac{Enc.\ Time_{Algorithm}}{Enc.\ Time_{Original\ Graph}} \)

Results

<table>
<thead>
<tr>
<th></th>
<th>Number of Subgraphs</th>
<th>CR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mobile</td>
<td>82</td>
<td>48</td>
</tr>
<tr>
<td>Foreman</td>
<td>14</td>
<td>4</td>
</tr>
<tr>
<td>Carphone</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Conclusions

- The complexity reduction can be significant.
- Negligible loss of performance.

Further work

- Drawback: The final complexity depends on the motion of the video sequence.
- Solutions:
  - Constraint Motion Vectors.
  - Remove links between nodes in the bigger subgraphs.
  - Limit the number of blocks that a subgraph can have.
1 Introduction

2 Lifting transforms on graphs for video coding
   - Overview
   - Low Complexity Transform Using Subgraphs
   - Coefficient Reordering

3 Experimental Results

4 Conclusions and Future lines
Coefficient Reordering

**Goal**  Reorder the coefficients generated by our graph-based transform in an efficient way.

- e.g., Zig-zag scanning order in DCT based encoders.
- e.g., Hierarchical trees in wavelet-based encoders [Shapiro, 1992, Kim and Pearlman, 1997].

**Proposed Solution**  Reorder the coefficients using graph information:

1. Inter-subband reordering.
2. Intra-subband reordering.
Inter-Subband Reordering

Observation Energy in the middle-high frequency subbands will be low.

Proposed reordering Group coefficients that belong to the same subband → Long strings of zero coefficients after quantization.

The coefficients will be sorted as:

\[
\text{coeffs}_{\text{inter}} = [s_{j=N}, d_{j=N}, d_{j=N-1}, \ldots, d_{j=1}].
\]

- \(s_{j=N}\): smooth coefficients at level of decomposition \(j = N\) (the lowest frequency subband).
- \(d_{j}^i\): detail coefficients at generic level of decomposition \(j\).
Inter-Subband Reordering Example

![Coefficients evolution](image)

20 frames of the sequence *Carphone*.
Intra-Subband Reordering

Observations

- The graph is known at the encoder and the decoder.
- Weights provide an estimation of the reliability with which one predict node is predicted from update neighbors.

Assumption

Detail coefficients will be smaller if they have been predicted from more “reliable” neighbors.

Proposed reordering

Reorder the coefficients in each subband as a function of the reliability of their links.
Intra-Subband Reordering Example

- Detail coefficients (the white nodes) of a generic subband $j$, $d^j = [1, 2, 4]$
- Reliability values (average of the weights of all graph edges used to compute that coefficient): $r^j = [2a, 3a, a]$

Intra-subband reordered coefficients $d^j_{intra} = [4, 1, 2]$.  

- Coefficients energy is lower as the reliability increases.
Reordering Results

Test Conditions

- 20 Frames, QCIF sequences.
- Different qualities (32.9 dB and 36 dB respectively).
- Comparison: without reordering the coefficients, employing inter-reordering, and inter and intra reordering.
- Rate is obtained with an arithmetic coder.

<table>
<thead>
<tr>
<th></th>
<th>Without reordering</th>
<th>Inter reordering</th>
<th>Inter and Intra reordering</th>
</tr>
</thead>
<tbody>
<tr>
<td>Foreman</td>
<td>503 Kbps</td>
<td>404 Kbps</td>
<td>350 Kbps</td>
</tr>
<tr>
<td>Carphone</td>
<td>502 Kbps</td>
<td>425 Kbps</td>
<td>371 Kbps</td>
</tr>
</tbody>
</table>
1 Introduction

2 Lifting transforms on graphs for video coding
   - Overview
   - Low Complexity Transform Using Subgraphs
   - Coefficient Reordering

3 Experimental Results

4 Conclusions and Future lines
Test Conditions

- Comparison against a motion-compensated DCT video encoder.

- **Coefficients quantization:**
  - Uniform dead-zone quantizer in the DCT.
  - Subband dependent quantization in our encoder.

- **Coefficients scanning:**
  - Zigzag scanning order in the DCT.
  - As is explained in Section 3 in our proposed method.

- **Run-length** encoding is performed in both encoders.

- The bitstream is obtained coding the symbols using an arithmetic coder.

- Five levels of decomposition.

- Temporal links $\rightarrow w_t = 10$; Spatial links $\rightarrow w_s = 2$. 

Video Encoder Based on Lifting Transforms on Graphs
Proposed method outperforms the DCT based approach in medium to high qualities.

The efficiency of the encoder at low qualities gets worse.
1 Introduction

2 Lifting transforms on graphs for video coding
   • Overview
   • Low Complexity Transform Using Subgraphs
   • Coefficient Reordering

3 Experimental Results

4 Conclusions and Future lines
Conclusions and Future Lines

**Conclusions:**
- Directional transform that filters following 3-dimensional spatio-temporal directions of high correlation $\rightarrow$ Spatial and temporal correlation jointly exploited.
- Low complexity version closer to practical implementations.
- New coefficients reordering method using graph information.

**Future lines:**
- Assign the weights between pixels as a function of the features of the region to be encoded.
- Transform that operates in fixed-size blocks.
- Encoder Optimization.
Thank you!
References

Edge-avoiding wavelets and their applications.
In *SIGGRAPH ’09: ACM SIGGRAPH 2009 papers*, pages 1–10, New York, NY, USA. ACM.

Minimum-via topological routing.

An embedded wavelet video coder using three-dimensional set partitioning in hierarchical trees (SPIHT).

Sparse geometric image representations with bandelets.

Lifting transforms on graphs for video coding.
In *Data Compression Conference (DCC), 2011*, pages 73 –82.

Three-dimensional lifting schemes for motion compensated video compression.

Lifting-based invertible motion adaptive transform (limat) framework for highly scalable video compression.
An embedded wavelet hierarchical image coder.  

Compact image representation using wavelet lifting along arbitrary trees.  

Directionlets: anisotropic multidirectional representation with separable filtering.  