

Information and Coding

Armando J. Pinho

Departamento de Electrónica, Telecomunicações e Informática
Universidade de Aveiro

`ap@ua.pt`

Contents

- 1 Predictive coding
 - Principles
 - Predictive coding techniques
 - Predictors
 - Lossless predictive coding
 - Motion compensation

Principles

- Let $x^n = x_1 x_2 \dots x_n$ be the sequence of values (scalars or vectors) produced by an information source until time n .
- **Predictive coding** is based on encoding sequence $r^n = r_1 r_2 \dots r_n$, instead of the original sequence x^n , where

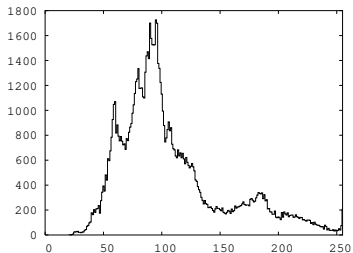
$$r_n = x_n - \hat{x}_n$$

and

$$\hat{x}_n = p(x^{n-1}) = p(x_1 x_2 \dots x_{n-1})$$

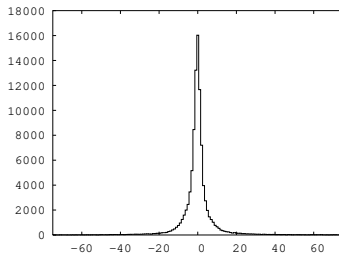
- The \hat{x}_n are the **estimates** and the values of the sequence r^n are the **residuals**.
- Function $p()$ is the **estimator** or **predictor**.
- The aim of predictive coding is to have $H(r^n) < H(x^n)$.

Example



Original

$H = 7.26$ bits/symbol



Predictor 1 JPEG

$H = 4.49$ bits/symbol

Simple 1D prediction

- Simple polynomial predictors used in some audio encoders:

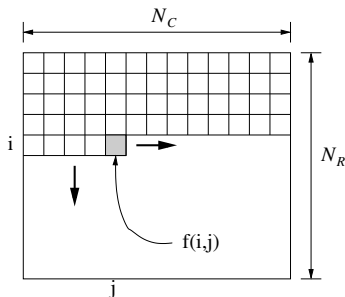
$$\begin{cases} \hat{x}_n^{(0)} = 0 \\ \hat{x}_n^{(1)} = x_{n-1} \\ \hat{x}_n^{(2)} = 2x_{n-1} - x_{n-2} \\ \hat{x}_n^{(3)} = 3x_{n-1} - 3x_{n-2} + x_{n-3} \end{cases}$$

and the corresponding residuals, computed efficiently:

$$\begin{cases} \hat{r}_n^{(0)} = x_n \\ \hat{r}_n^{(1)} = r_n^{(0)} - r_{n-1}^{(0)} \\ \hat{r}_n^{(2)} = r_n^{(1)} - r_{n-1}^{(1)} \\ \hat{r}_n^{(3)} = r_n^{(2)} - r_{n-1}^{(2)} \end{cases}$$

Predictive image coding techniques

- Typically, images are encoded from left to right, top to bottom, i.e., in **raster-scan** order:



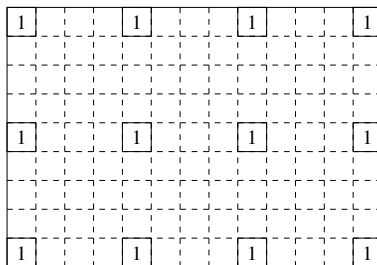
- In this case, the sequence x^n is obtained by concatenating the first $\lfloor n/N_C \rfloor$ image rows, plus the $n \bmod N_C$ pixels from row number $\lfloor n/N_C \rfloor + 1$.

Predictive image coding techniques

- Other approaches use hierarchical decompositions (or **multi-resolution**).
- This is the case of the HINT method (Hierarchical INTerpolation):

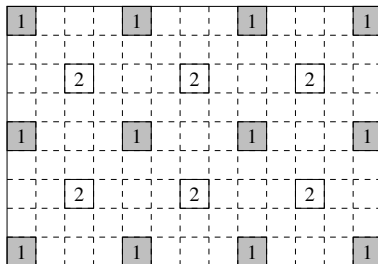
Predictive image coding techniques

- Other approaches use hierarchical decompositions (or **multi-resolution**).
- This is the case of the HINT method (Hierarchical INTerpolation):



Predictive image coding techniques

- Other approaches use hierarchical decompositions (or **multi-resolution**).
- This is the case of the HINT method (Hierarchical INTerpolation):



Predictive image coding techniques

- Other approaches use hierarchical decompositions (or **multi-resolution**).
- This is the case of the HINT method (Hierarchical INTerpolation):

1	3	1	3	1	3	1	3
3	2	3	2	3	2	3	2
1	3	1	3	1	3	1	3
3	2	3	2	3	2	3	2
1	3	1	3	1	3	1	3

Predictive image coding techniques

- Other approaches use hierarchical decompositions (or **multi-resolution**).
- This is the case of the HINT method (Hierarchical INTerpolation):

1		3		1		3		1		3		1
	4		4		4		4		4		4	
3		2		3		2		3		2		3
	4		4		4		4		4		4	
1		3		1		3		1		3		1
	4		4		4		4		4		4	
3		2		3		2		3		2		3
	4		4		4		4		4		4	
1		3		1		3		1		3		1

Predictive image coding techniques

- Other approaches use hierarchical decompositions (or **multi-resolution**).
- This is the case of the HINT method (Hierarchical INTerpolation):

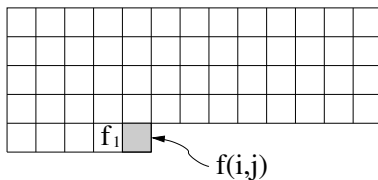
1	5	3	5	1	5	3	5	1	5	3	5	1
5	4	5	4	5	4	5	4	5	4	5	4	5
3	5	2	5	3	5	2	5	3	5	2	5	3
5	4	5	4	5	4	5	4	5	4	5	4	5
1	5	3	5	1	5	3	5	1	5	3	5	1
5	4	5	4	5	4	5	4	5	4	5	4	5
3	5	2	5	3	5	2	5	3	5	2	5	3
5	4	5	4	5	4	5	4	5	4	5	4	5
1	5	3	5	1	5	3	5	1	5	3	5	1

Predictors

- For efficient encoding, the estimated values should be as close as possible to the real values, i.e., the r_k values should be small.
- The decoder must be able to generate the same sequence, \hat{x}^n , of estimated values, i.e., **the predictor cannot introduce any error** during encoding / decoding.
- Therefore, the predictor must be **causal**, and, in lossy coding, the predictor at the encoder **must use the reconstructed values**, \tilde{x}^{n-1} , instead of the original values, x^{n-1} .

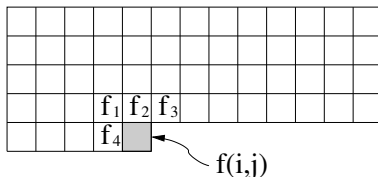
Predictors

- Generally, the complexity of the predictor depends on two aspects:
 - The number of values used for calculating the estimates (**the order of the predictor**).
 - The spatial (or temporal) configuration of these values.
- Consider the example of a **spatial predictor of order 1**, where the estimated value is given by the immediately preceding value:



Predictors

- This type of predictor can be easily extended to higher orders, using the last k processed pixels of the image.
- However, for orders higher than 3 or 4, the efficiency does not increase significantly.
- This happens because images are 2D signals, not 1D sequences of data.
- Therefore, generally, the spatial configurations used for predictive image coding have a 2D shape:

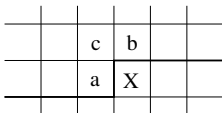


Lossless predictive coding

- One of the main advantages of predictive coding is allowing a simple design of lossless encoders.
- In fact, **most lossless encoders for audio and image rely on predictive coding techniques.**
- However, for lossless coding, there is an additional constraint regarding the predictor: the estimates generated must be **platform independent.**
- Generally, this constraint implies that the predictor can use only **integer arithmetic.**

Linear prediction: the lossless mode of JPEG

- The lossless mode of JPEG (ISO/IEC 10918-1, ITU-T T.81, 1992) provides seven **linear predictors**:



Mode	Predictor
1	a
2	b
3	c
4	$a + b - c$
5	$a + (b - c)/2$
6	$b + (a - c)/2$
7	$(a + b)/2$

- Generally, the performance of the several predictors may vary considerably from image to image.
- If encoding time is not a problem, then all of them can be tested and the one with the best compression rate chosen.

Linear prediction: the lossless mode of JPEG

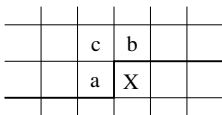
Example:



Predictor	1	2	3	4	5	6	7
Entropy	4.49	4.21	4.74	4.17	4.16	4.04	4.10

The nonlinear predictor of JPEG-LS

- JPEG-LS (ISO/IEC 14495-1, ITU-T T.87, 1999) uses a predictor based on the same spatial configuration as that of JPEG:



- However, instead of a linear predictor, it uses the **nonlinear predictor**

$$\hat{x} = \begin{cases} \min(a, b) & \text{if } c \geq \max(a, b) \\ \max(a, b) & \text{if } c \leq \min(a, b) \\ a + b - c & \text{otherwise} \end{cases}$$

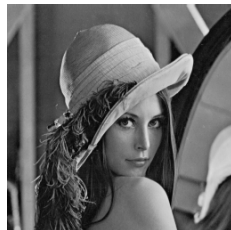
- Note that the linear part of this predictor ($a + b - c$) is the same as predictor number 4 of JPEG.

The nonlinear predictor of JPEG-LS

Example:



(a)

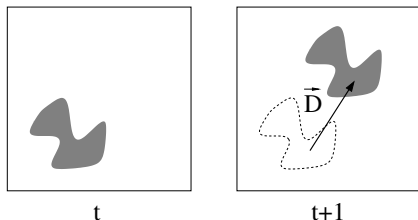


(b)

Predictor	1	2	3	4	5	6	7	JLS
Entropy (a)	4.49	4.21	4.74	4.18	4.16	4.04	4.10	3.98
Entropy (b)	5.60	5.05	5.82	5.19	5.23	4.97	5.15	4.93

Motion compensation

- Typically, the differences between consecutive frames of a video sequence are due to motion of the scene objects.
- Exceptions occur when there are scene changes, zoom-in / zoom-out operations and camera translation.



- To explore this redundancy, it is frequent to use **temporal prediction** (interframe compression), which relies on **motion compensation**.

Motion compensation

- **Conditional replenishment** video coding:
 - Finds zones in the video frame where there were changes with respect to the previous frame.
 - Only those zones are encoded.
 - This technique does not use motion compensation. It just performs a detection of temporal activity.
- Video coding based on **motion compensation** involves the following steps:
 - Estimation of the motion vectors.
 - Compensation, i.e., temporal prediction.
 - Encoding of the motion vectors.
 - Encoding of the prediction residuals.

Motion compensation

- There are a large number of techniques for motion detection, but one of them is clearly the most common approach for video coding.
- For each frame block (for example, of $N \times N$ pixels), it seeks the position where it minimizes some measure in relation to the previous frame (**the reference frame**).
- Note that this approach tries to find the position that minimizes a measure of interest, which might not correspond to the true motion in the scene...

Motion compensation

- Typically, we want to minimize some measure $C(i, j)$, such as

$$C(i, j) = \sum_{r=1}^N \sum_{c=1}^N d\left(g(r, c, t) - g(i + r, j + c, t + 1)\right)$$

where $d(\cdot)$ is, for example, $(\cdot)^2$ or $|\cdot|$.

- Due to complexity constraints, searching is limited to a neighborhood of $(N + 2\Delta) \times (N + 2\Delta)$ pixels around the block, i.e., $-\Delta \leq i, j \leq \Delta$.
- If exhaustive search is used, it is guaranteed to find the minimum of $C(i, j)$...
- This approach is generally computationally too demanding, hence other **sub-optimal techniques** have been proposed.

Motion compensation

- Example:



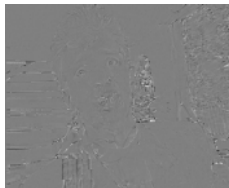
Frame 200



Frame 201



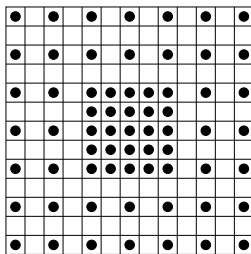
Direct difference
 $H = 5.23$ bpp



Motion compensation
 $H = 4.38$ bpp

Motion compensation

- Several of the sup-optimal approaches for finding the best reference block rely on **spatial sub-sampling**.
- For example, considering that the most probable zone for finding the reference block is in the near neighborhood of the block, then we may use the following scheme:

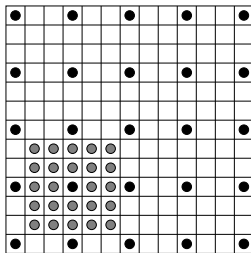


Total: 169 blocks

Sub-optimal: 65 blocks

Motion compensation

- If we consider that after finding a reasonably good reference block it is probable that others better than itself can be found in the near neighborhood, then we can use a greedy search:



Total: 169 blocks

Sub-optimal: 49 blocks

- A number of other variants of local search have been proposed. . .

Motion compensation

- For example, the logarithmic search:

