Video Compression

Dr. Zein Al Abidin IBRAHIM

ibrahim.zein@gmail.com

Zein.Ibrahim@ul.edu.lb

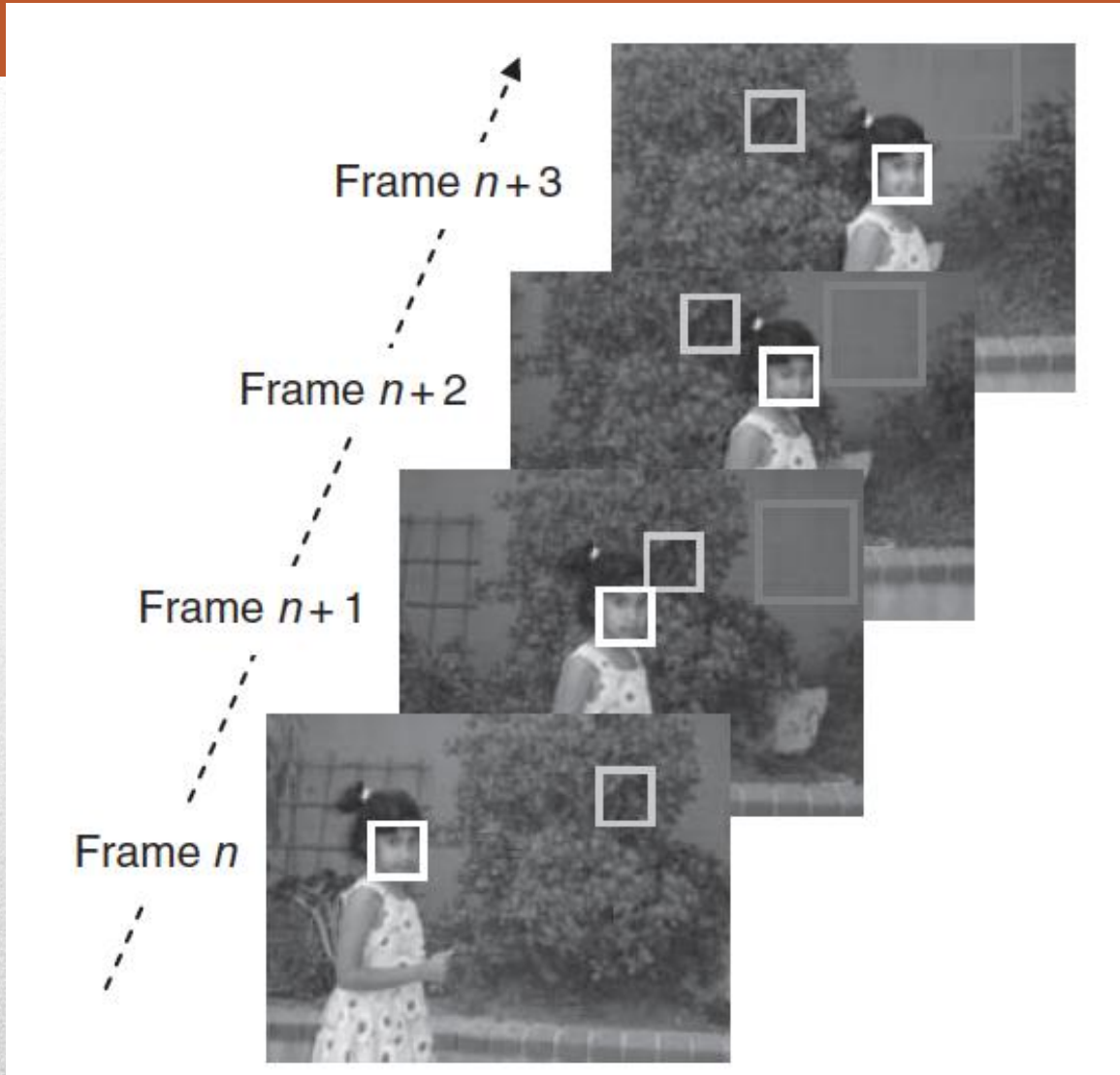
IN433
Multimedia Processing

- Each frame in a video is considered an image, right?
- Why not compress each frame individually?
- Spatial redundancy can be removed from each frame (similar to previous chapter) by applying JPEG
- This is called M-JPEG (Motion-JPEG)
- Decoder decodes each frame individually
- Used in some streaming applications or mobile applications (not sophisticated 1-2 frames per second)

Primitive compression

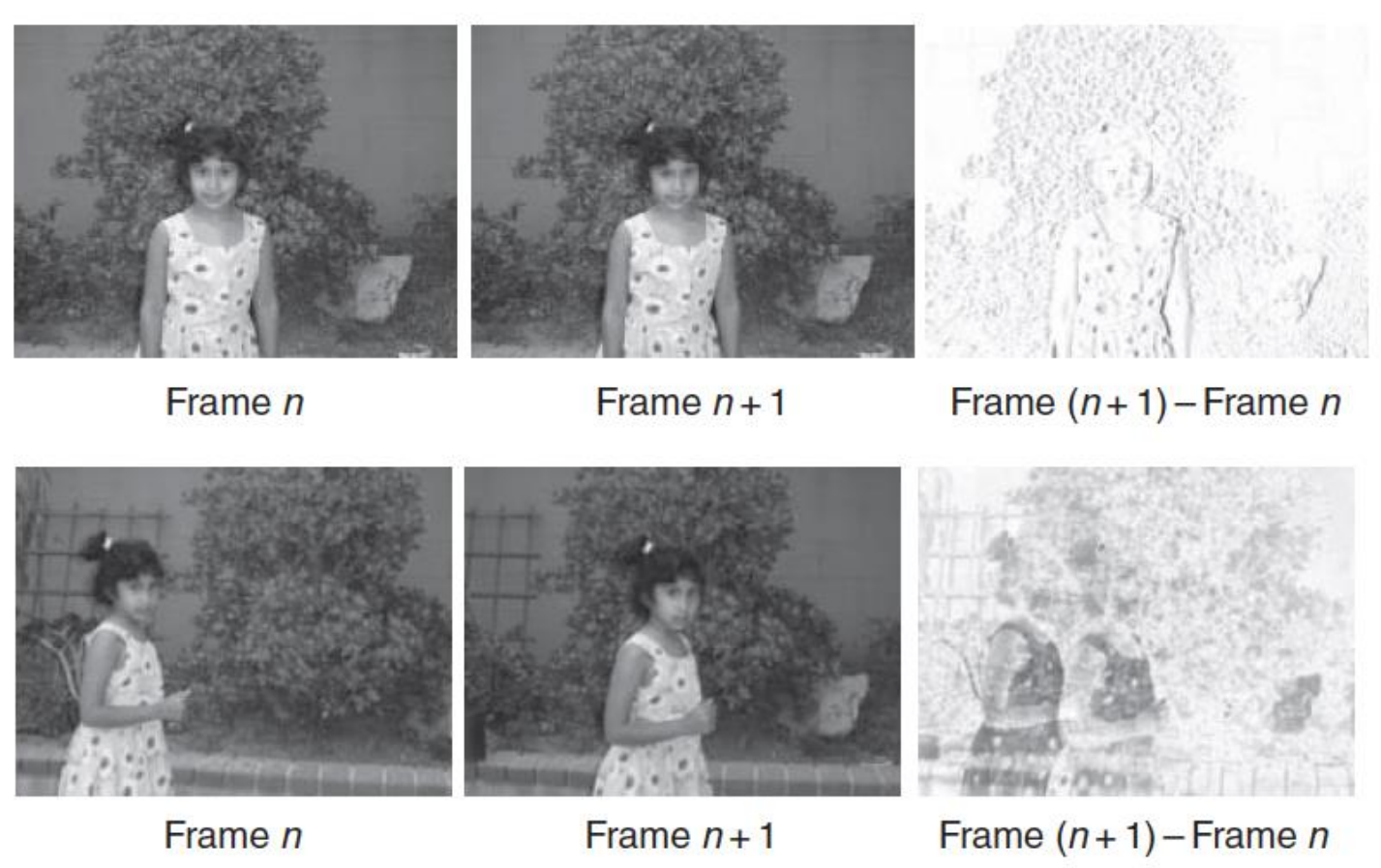
- Temporal redundancy
- Pixels are correlated across frames, not just inside one frame.
- Background may stay still while objects move or objects move but remain the same shape and color across frames.

Better approach



Temporal redundancy

- Subtract frames and encode only the errors.
- Which video you think has low motion and which has higher motion?

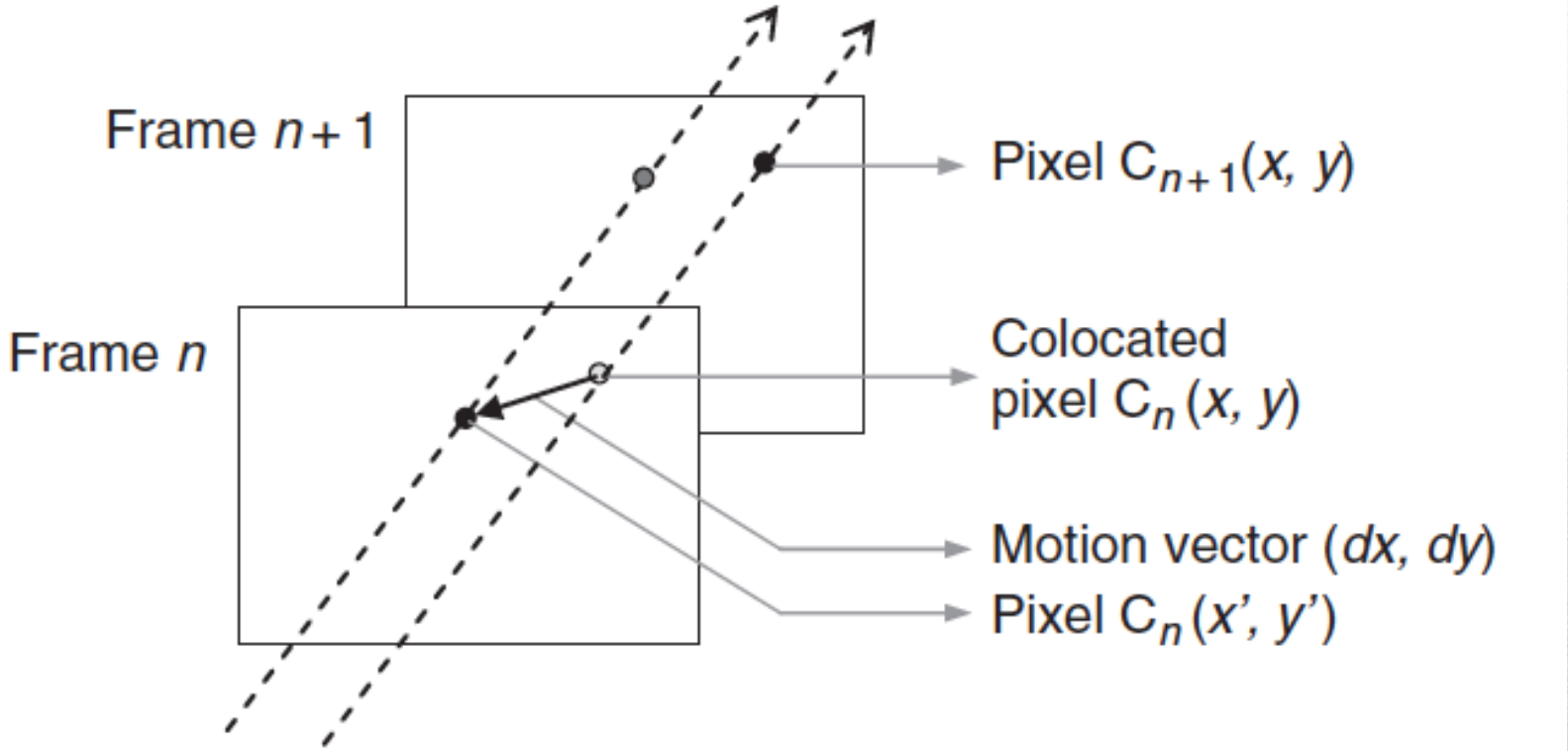


Use DPCM

- Problem: motion increases → difference increases
- Solution: Prediction.
- Predict the motion of pixels/regions instead of whole frame

Problem & Solution

- Mathematically model the change of location of a pixel



Temporal redundancy

- Case 1: none of the objects has moved, the background is still and lighting conditions is the same:

$$C_{n+1}(x, y) = C_n(x, y)$$

- Case 2: objects/background/lighting conditions changed:

$$C_{n+1}(x, y) \neq C_n(x, y)$$

Temporal redundancy

- However, $C_{n+1}(x,y)$ might be equal to some other pixel in frame n, $C_n(x',y')$ in the local neighborhood.

$$C_{n+1}(x, y) = C_n(x', y') = C_n(x - dx, y - dy)$$

- (dx, dy) is called motion vector
- Precisely, there is some error:

$$C_{n+1}(x, y) = C_n(x - dx, y - dy) + e(x, y)$$

Temporal redundancy

- It’s better to predict blocks/regions than individual pixels
- After all, a pixel doesn't move by itself but in a block.

Temporal redundancy

- The current frame that needs to be compressed is called TARGET FRAME (TF)
- We will use a previous frame called REFERENCE FRAME (RF) to predict from it
- The TF is divided into blocks called MACRO-blocks that will be predicted from RF.
- Better to work with blocks since images are a collection of blocks/regions and more efficient in terms of processing

Block Based Frame prediction

- When predicting a frame, the error between that actual frame and its predicted version should be very small (not always the case)
- This is because some new areas might appear in frame n+1, not visible in frame n

Block Based Frame prediction

- The process of predicting target frames from reference frames using macro-blocks is called motion compensation.

1. Find the best motion vector for each block
2. Create a predicted (motion compensated) frame image
3. Derive the prediction error image (actual-predicted)
4. Compress error image using JPEG and motion vectors using lossless entropy coding

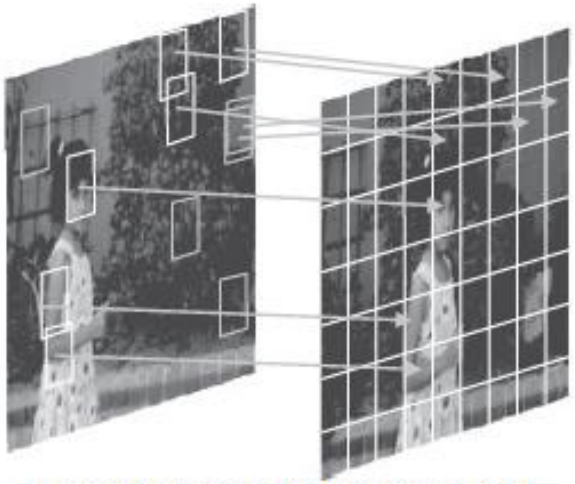
Block Based Frame prediction

- At the decoder, a frame is predicted from the previous using the motion vectors
- Then the error image is added to fix the frame.
- The more accurate motion vectors are, the better the performance and lesser the entropy of the error image.

Block Based Frame prediction

Frame n

Frame $n+1$

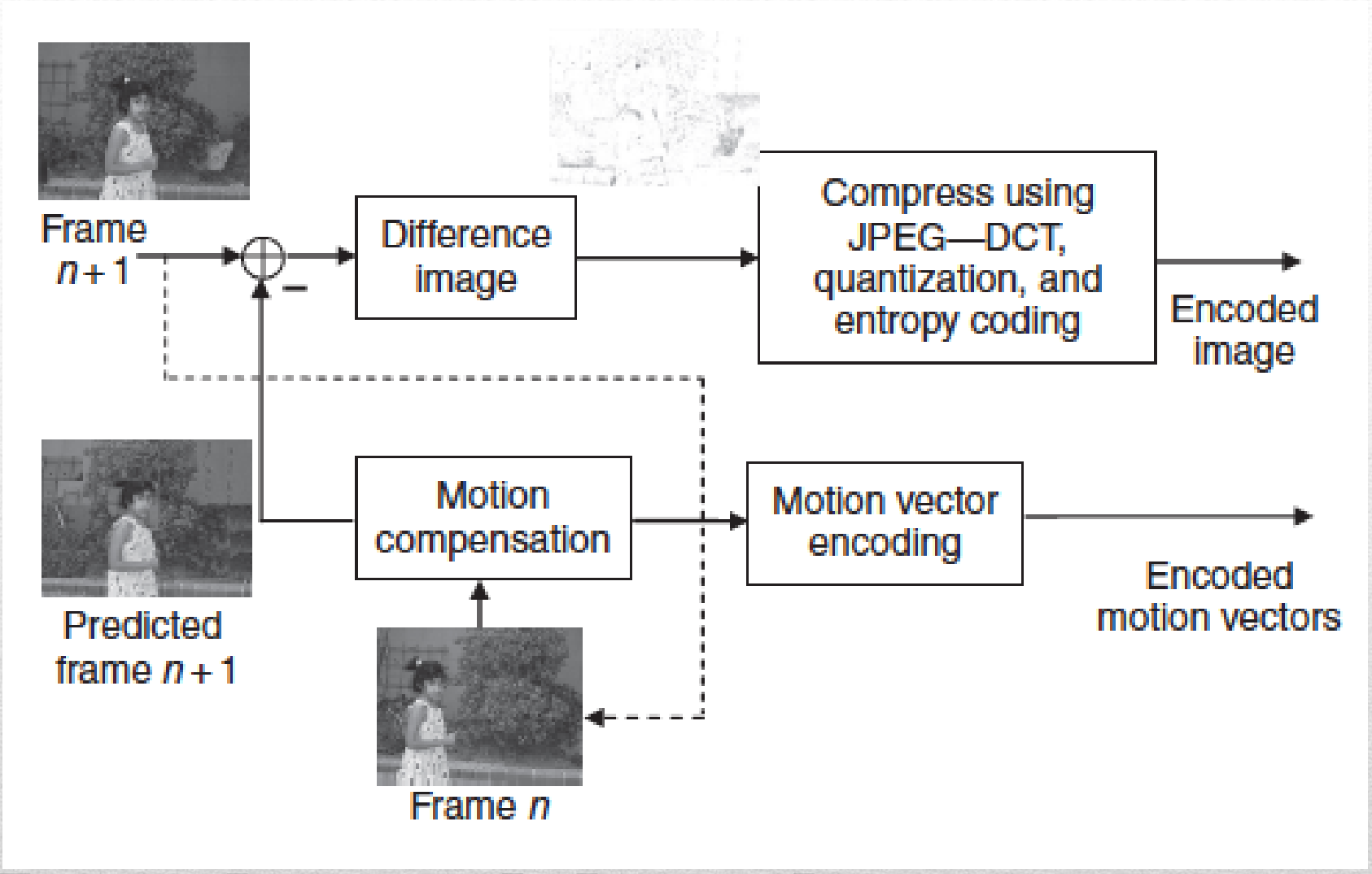


Macroblock prediction of frame $n+1$ from regions in frame n

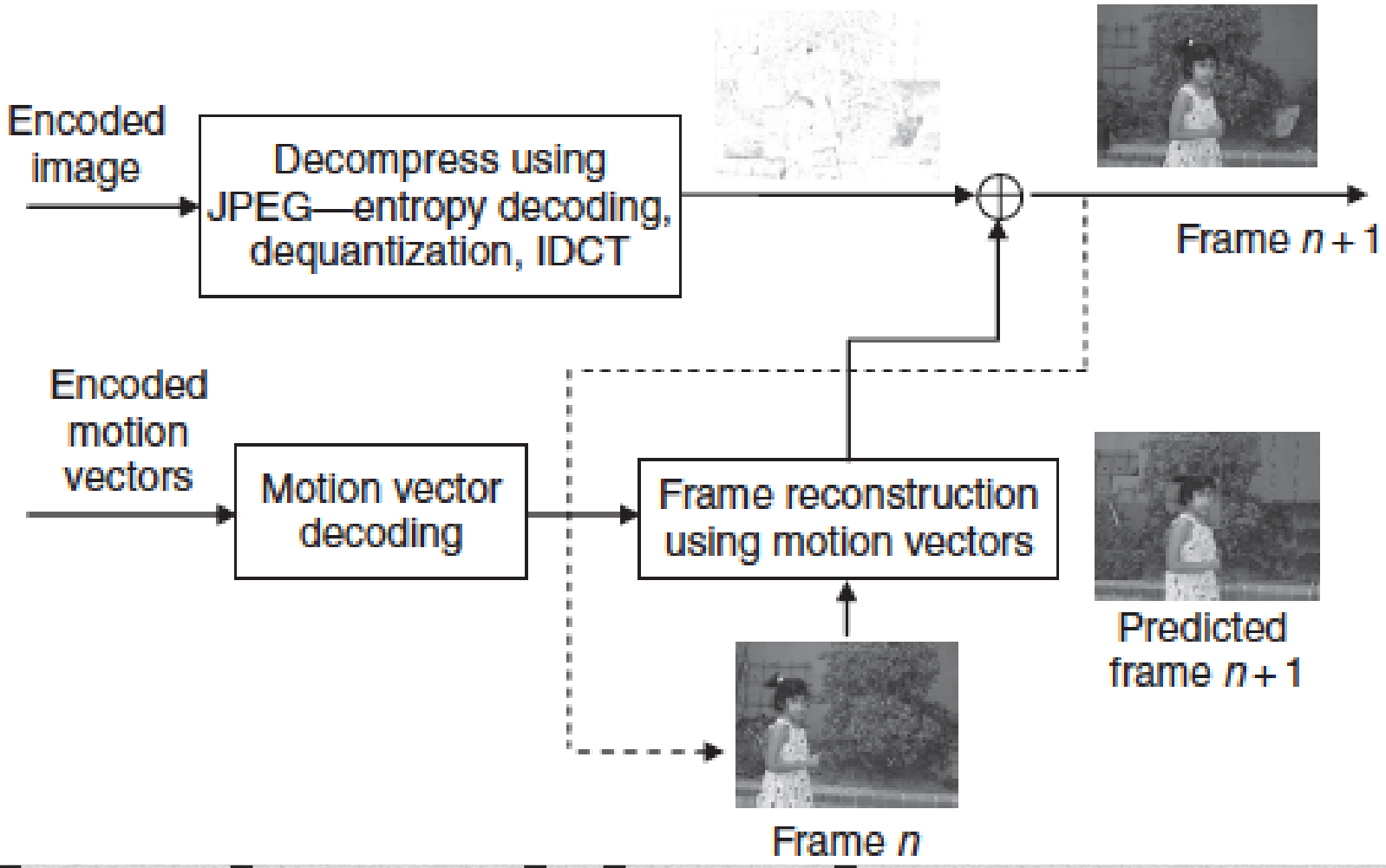
Reconstructed frame $n+1$ by macroblock prediction

Frame difference
frame $n+1$ - frame n

Frame difference
Reconstructed frame $n+1$ - frame $n+1$



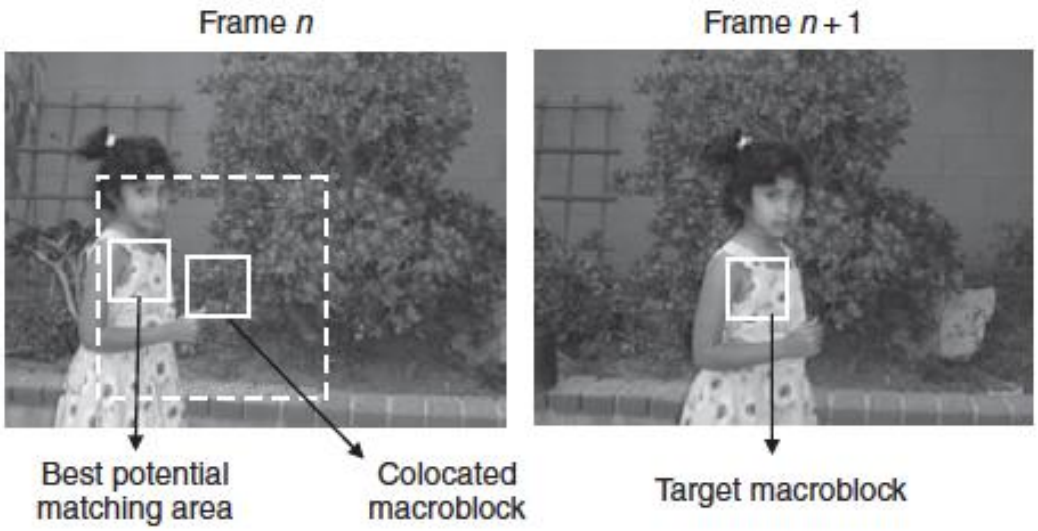
Encoder and decoder

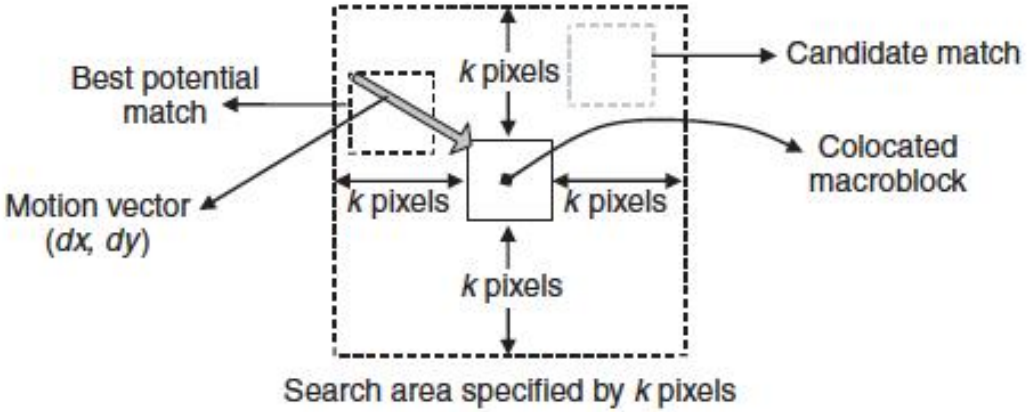


Encoder and decoder

- Search in a specified area around the colocated position in the reference frame.
- Search parameter k (0 to 31) depending on the motion speed
- Fast motion, bigger k because blocks move farther and vice versa.
- Find the best matching area in the reference frame out of all candidates

Computing Motion Vectors





Search area specified by *k* pixels

Macroblock overlaid on search area in the colocated position

Macroblock overlaid at a candidate position

Macroblock overlaid at the best-matched position showing motion vector

- A motion vector mv = (i,j) where i and j both vary in the search area and take values from -k to +k
- At any position i and j, the difference between the candidate area and the target block is calculated using mean absolute difference (MAD), or another metric

$$MAD\,(i,j) = \frac{\sum_{p=1}^{m}\sum_{q=1}^{n}|C_{n+1}[p,q] - C_n[p+i,q+j]|}{mn}$$

Computing motion vectors

- Find the positions i and j, that will result in the least MAD!!
- Other metrics can be mean square difference

$$\text{MSD(i, j)} = \frac{\sum_{p=1}^{m}\sum_{q=1}^{n}(C_{n+1}[p,q] - C_n[p+i, q+j])^2}{mn}$$

- PEL difference classification

$$\text{PEL(i, j)} = \sum_{p=1}^{m}\sum_{q=1}^{n}[ord\,(|C_{n+1}[p,q] - C_n[p+i, q+j]| \le t)]$$

where t is some predefined threshold that decides a match and $ord(x) = 1$ if x is true.

- Projective ordering

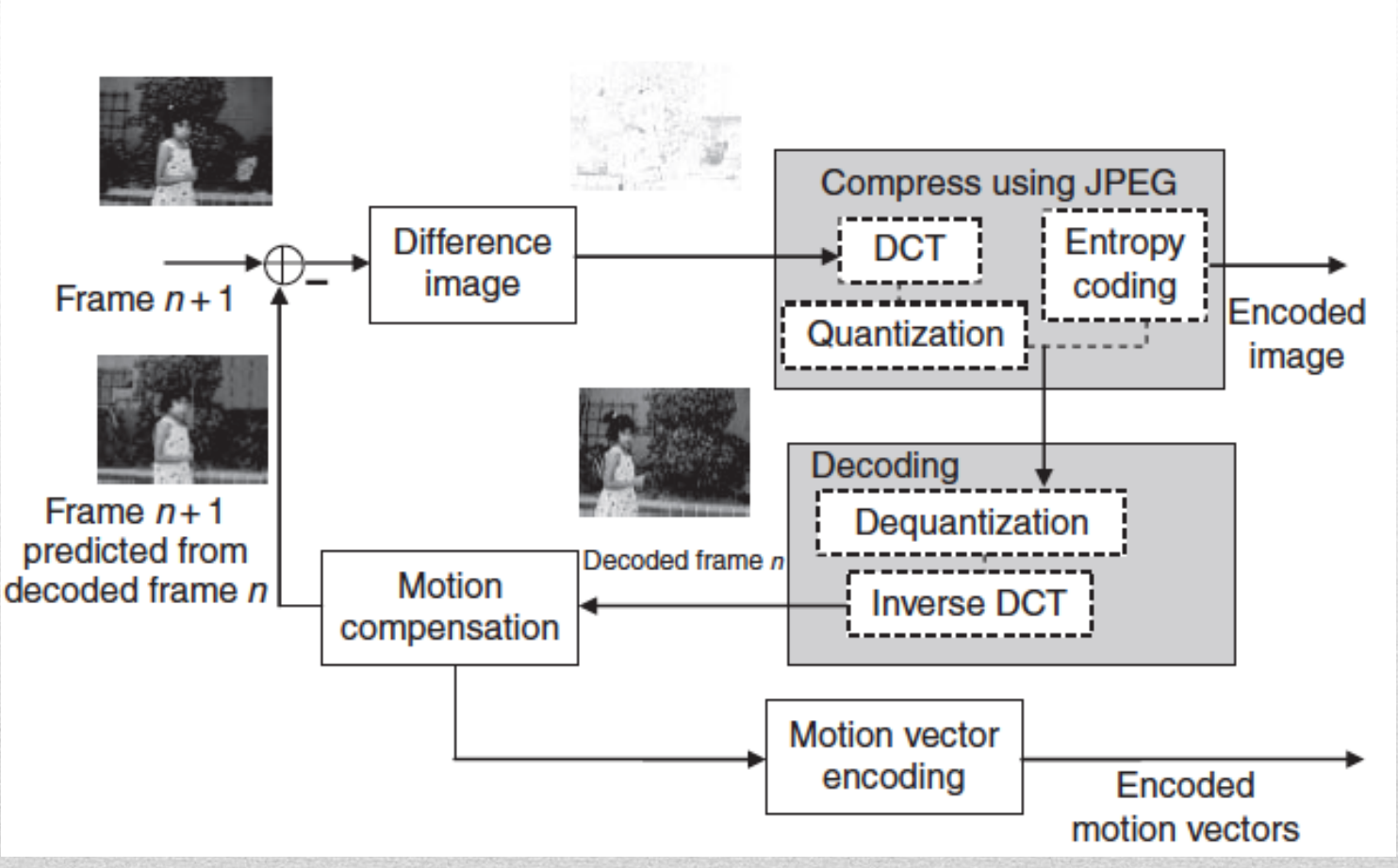
$$\text{PO(i, j)} = \sum_{p=1}^{m}\left|\sum_{q=1}^{n} C_{n+1}[p,q] - \sum_{q=1}^{n} C_n[p+i, q+j]\right| + \sum_{q=1}^{n}\left|\sum_{p=1}^{m} C_{n+1}[p,q] - \sum_{p=1}^{q} C_n[p+i, q+j]\right|$$

Other metrics

- Smaller macro-blocks increase the number of blocks in the target frame → larger number of motion vectors → more bits
- BUT, smaller blocks means better prediction means less error!!
- Bigger blocks means less blocks to process, but error will increase.
- Most ISO standards use 16x16
- H.264 uses variable sized blocks

Size of macro-blocks

- Same idea as DPCM



Open loop vs closed loop

Types of predictions

PART 2

- A past one, a future one, a combination?
- I, P and B frames are all used, as well as multi-frame predictions.

Which frame to reference to?

- Intra-frame coded where only spatial redundancy is used to compress that frame
- No prediction!
- I- frames are placed interspersed periodically in the encoding process and provide access points to the decoder to decode the succeeding predicted frames.

I-frames

- P frames are predictive coded, exploiting temporal redundancy by comparing them with the immediately preceding frame.
- This preceding reference frame might have been coded as an I frame or even a P frame.
- Coding P frames is similar to the process explained in Section 1, where the frame is predicted from the previous reference frame on a macroblock-by-macroblock basis by searching for the best motion vectors.
- The motion vectors are used to create a predicted frame. The error frame created by the difference between the reconstructed frame and the current frame. This error frame is bound of have lower entropy.
- Coding a P frame involves coding the motion vectors losslessly and coding the error frame using the JPEG pipeline.

P-frames

- They are expensive to compute, but are necessary for compression.
- An important problem the encoder faces is when to stop predicting using P frames, and instead insert an I frame.
- An I frame needs to be inserted where P frames cannot give much compression. This happens during scene transitions or scene changes, where the error images are high.
- It is pointless to go through a P frame computation only to find a high error enforcing an I frame.
- Compensating for unnecessary P frames at scene changes can be avoided if a scene change can be efficiently detected before starting P frame computation

P-frames

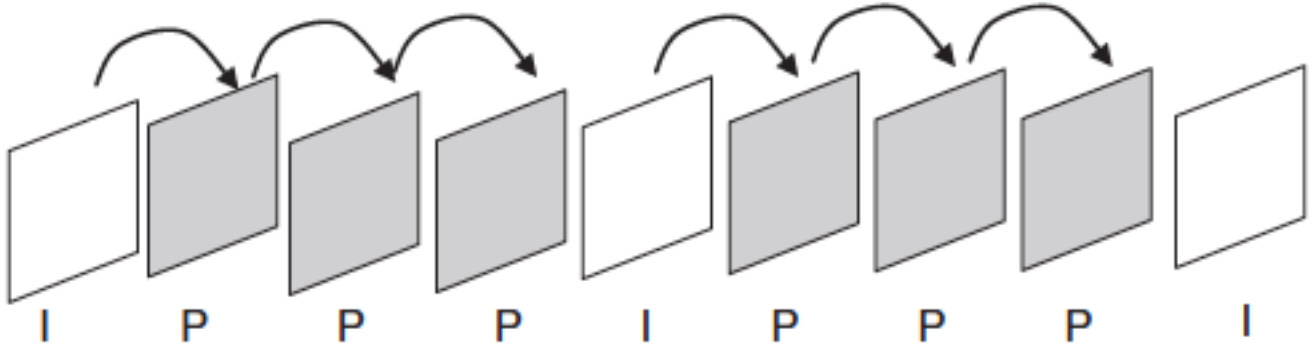


Figure 8-12 Interframe dependency in P frames. P frames are always forward predicted from a previous I frame or a P frame.

P-frames

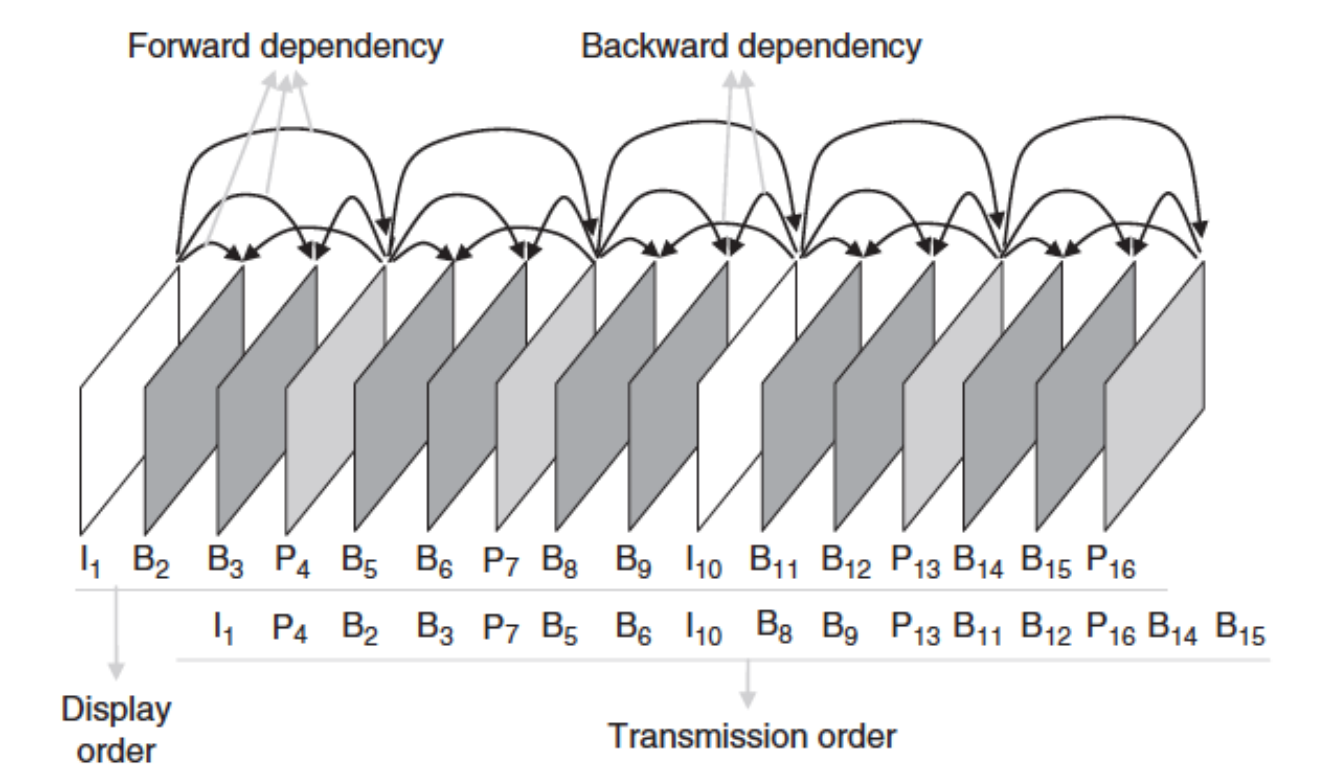
B-Frames

- Bidirectionally coded frames
- To predict a B frame, the previous or past frame and the next or future frame are used.
- Sometimes, areas of the current frame can be better predicted by the next future frame. This might happen because objects or the camera moves, exposing areas not seen in the past frames
- Note that the past and future reference frames can themselves be coded as an I or a P frame

- To compute a matching macroblock, the encoder needs to search for the best motion vector in the past reference frame and also for the best motion vector in the future reference frame.
- Two motion vectors are computed for each macroblock. Ultimately, the macroblock gets coded in one of three modes:
  - ✓ Backward predicted using only the past frame
  - ✓ Forward predicted using only the future frame
  - ✓ Interpolated, using both by averaging the two predicted blocks

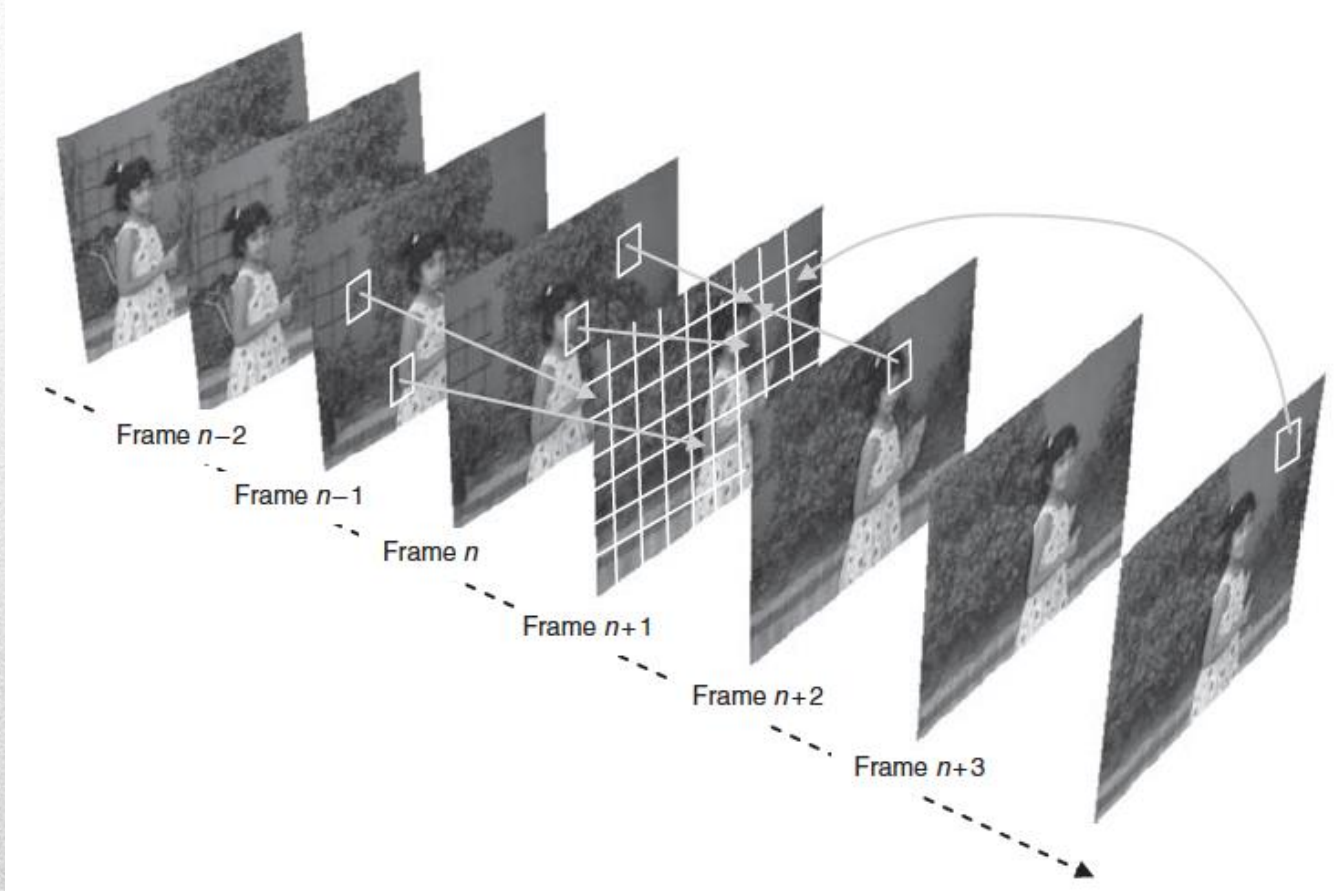
B-Frames

- This will induce some delays and complexity at the decoder
- Ordering of frames



B-Frames

- Multiple frames from past and future

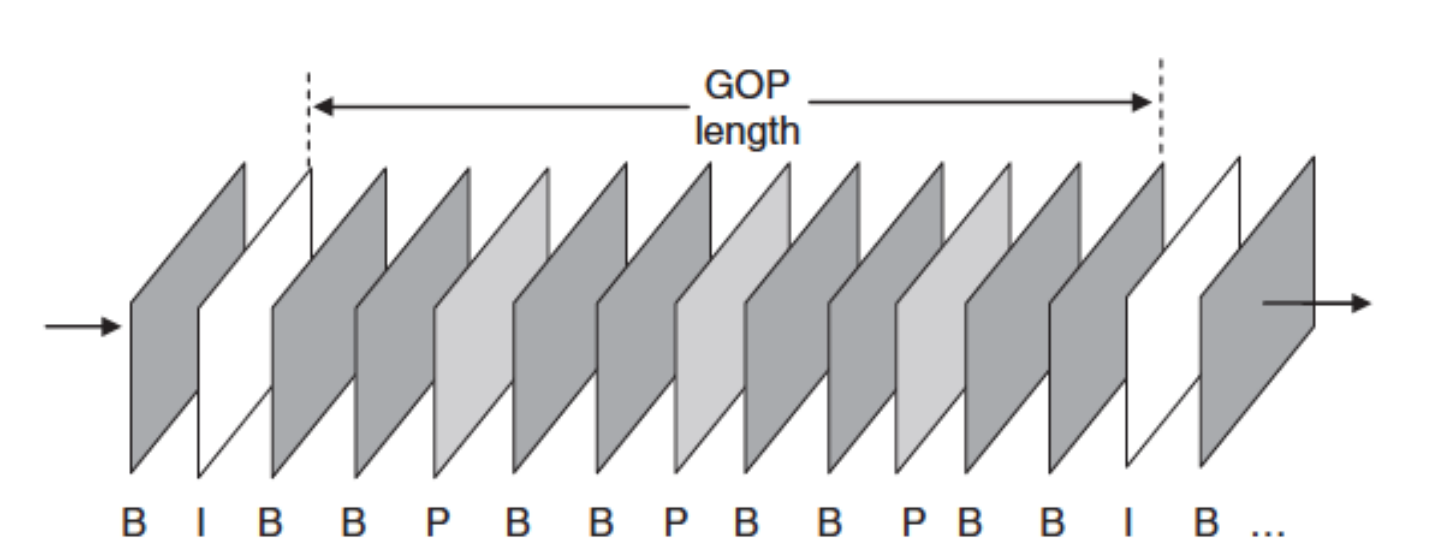


Multi-frame prediction

- MPEG, have adopted a hierarchical description to coded sequences known as a GOP or group of pictures.
- A GOP is a series of one or more frames to assist in random access of the frame sequence.
- The first coded frame in the group is an I frame, which is followed by an arrangement of P and B frames.
- The I frames and P frames are known as anchor frames and are used to predict B frames

Video Structure- Group of Pictures

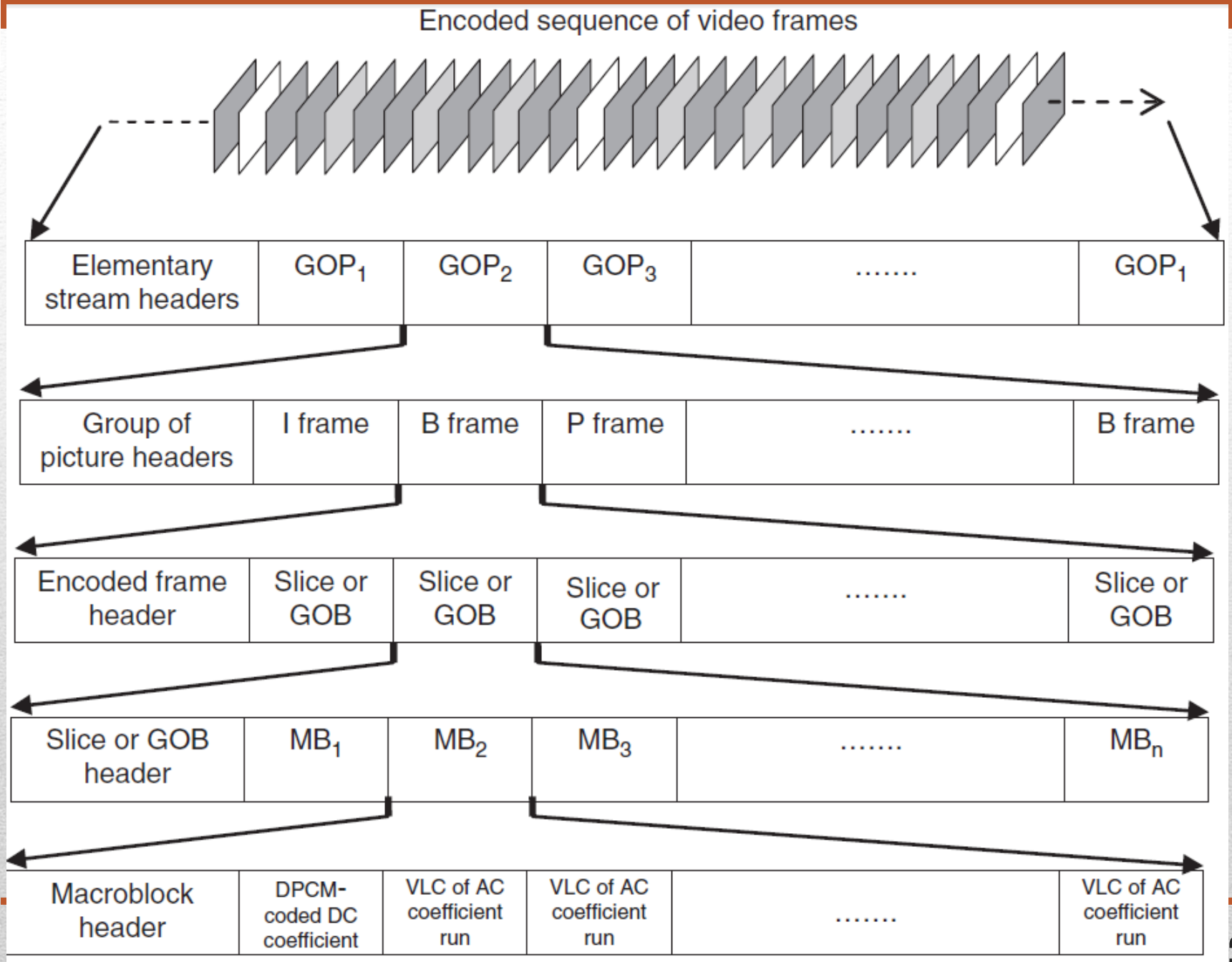
- The GOP length is defined as the distance between I frames.
- Standard codecs represent a GOP with two parameters N and M.
- N is the size of the GOP that gives the number of frames in the GOP.
- The distance between the anchor frames (I and/or P) is specified by M.
- For the example shown in the figure below, N 12, M 3.



Video Structure- GOP

GOPs

- GOPs are also specified using two parameters that are input to the encoder—"PbetweenI", which gives the number of P frames between I frames, and
- "BbetweenP", which gives the number of consecutive B frames between P frames.
- The group of pictures can be of any length, but must have at least one I frame in any GOP.
- GOPs are very useful for applications that require random access to the bit stream.
  - Example: fast forward or fast rewind, where the application can randomly access the I frame, decode it, and display it, not worrying about B and P frames.


Encoded sequence of video frames
Elementary stream headers
GOP1
GOP2
GOP3
.....
GOP1
Group of picture headers
I frame
B frame
P frame
.....
B frame
Encoded frame header
Slice or GOB
Slice or GOB
Slice or GOB
.....
Slice or GOB
Slice or GOB header
MB1
MB2
MB3
.....
MBn
Macroblock header
DPCM-coded DC coefficient
VLC of AC coefficient run
VLC of AC coefficient run
.....
VLC of AC coefficient run

Complexity of Motion Compensation

PART 3

Dr. Zein Ibrahim

- Given a macroblock of size n x n, the complexity of evaluating the error metric at each position is $O(n^2)$, whether you choose MAD, MSD, and so on.
- However, the number of positions at which a MAD or MSD is evaluated to decide the motion vector for a macroblock depends on the search techniques used

Why is it complex ?

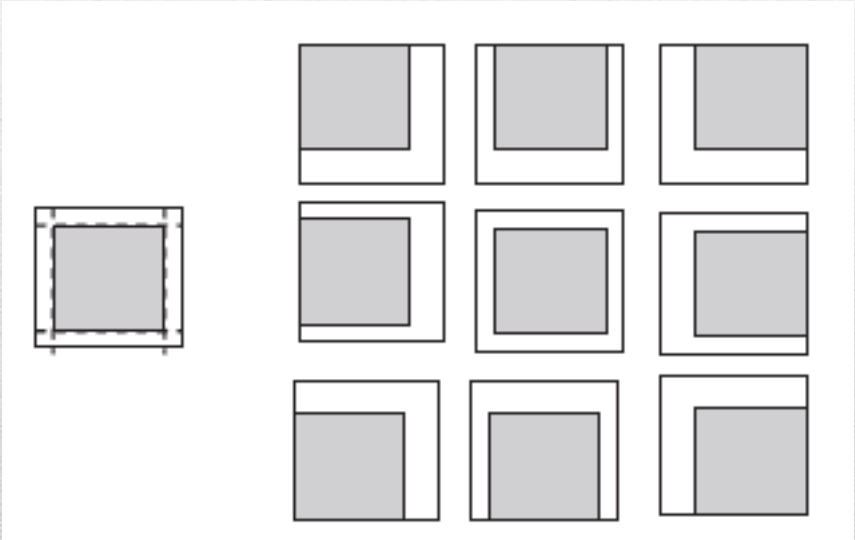
- 60% to 80% of the total encoding time is spent just on motion vector search.
- In practical video-encoding applications, a search parameter **k** (which decides the search area) along with the type of search to be performed is specified as input to the encoder, depending on:
  - **the application scenario,**
  - **the compression bit rate desired,**
  - **real-time constraints, and so on.**

Why is it complex?

- Sequential or brute force search
- Logarithmic search
- Hierarchical/multiresolution search

Types of search

- In a brute force search, the error metric is evaluated at all candidate positions. Given a macroblock of size n x n and search area defined by k, there are $(2k+ 1)^2$ candidate motion vector positions at which a MAD needs to be evaluated.
- Example with a search area specified with k =1. There are **nine** positions as shown.



1- Sequential/brute force

- Because computing one error metric has $\mathbf{n}^2$ computations, the total complexity of this search is $(2\mathbf{k}+1)^2 \times \mathbf{n}^2$.
- The vector that corresponds to the least MAD is taken to be the motion vector for the macroblock in the target frame
- Example: 720x480 video with 30 Hz frame rate, macroblock 16x16 and k=31(search size)
  - Calculate the number of MSD calculations per second
  - It is affordable if each MSD (all the n^2) takes 1 ms

1- Sequential/brute force

- Not suited for real time obviously
- But offers the best motion vector since it searches in the whole space
- There should be better approaches which only search parts of the space instead of all of it.

1- Sequential/brute force

- Logarithmic search works on the assumption that the error distortion metric (MAD or MSD) decreases monotonically in the direction of motion.
- In this case, the metric is evaluated in an iterative fashion at nine positions during each iteration.
- In the beginning, given a search area parameter **k**, the evaluation proceeds by computing the metric at the center of the search area and at eight locations with vectors (**k**/2, **k**/2), (0, **k**/2), (-**k**/2, **k**/2), (-**k**/2, 0), (-**k**/2, -**k**/2), (0, -**k**/2), (**k**/2,- **k**/2), and (**k**/2, 0).
- These are used as seed locations to come up with an initial match at the minimum MAD position.

2- logarithmic search

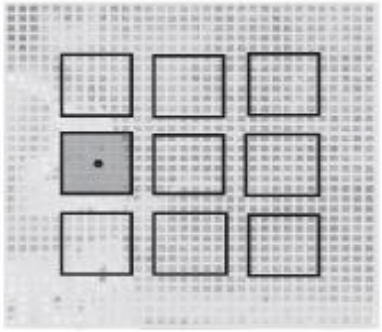
- At the next iteration, the center of the new search region is moved to it, and the search step size is reduced to half.
- At the second iteration level, the step size is **k**/4 and the iterative evaluation proceeds until the step size is 1, at which point the best match yields the best motion vector.

2- logarithmic search

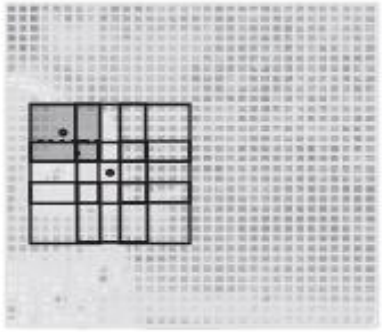
- a sample search with a search parameter **k** = 16.

Search area around the colocation

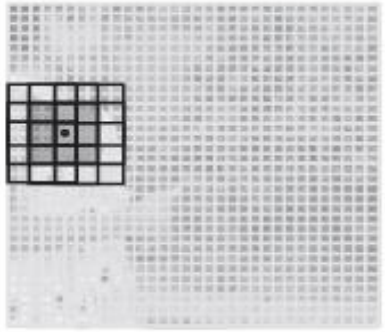
Macroblock of frame $n+1$



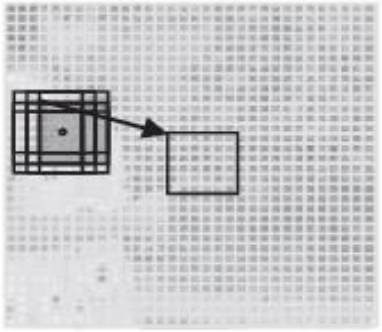
Iteration 1



Iteration 2



Iteration 3



Iteration 4 showing motion vector

2- logarithmic search

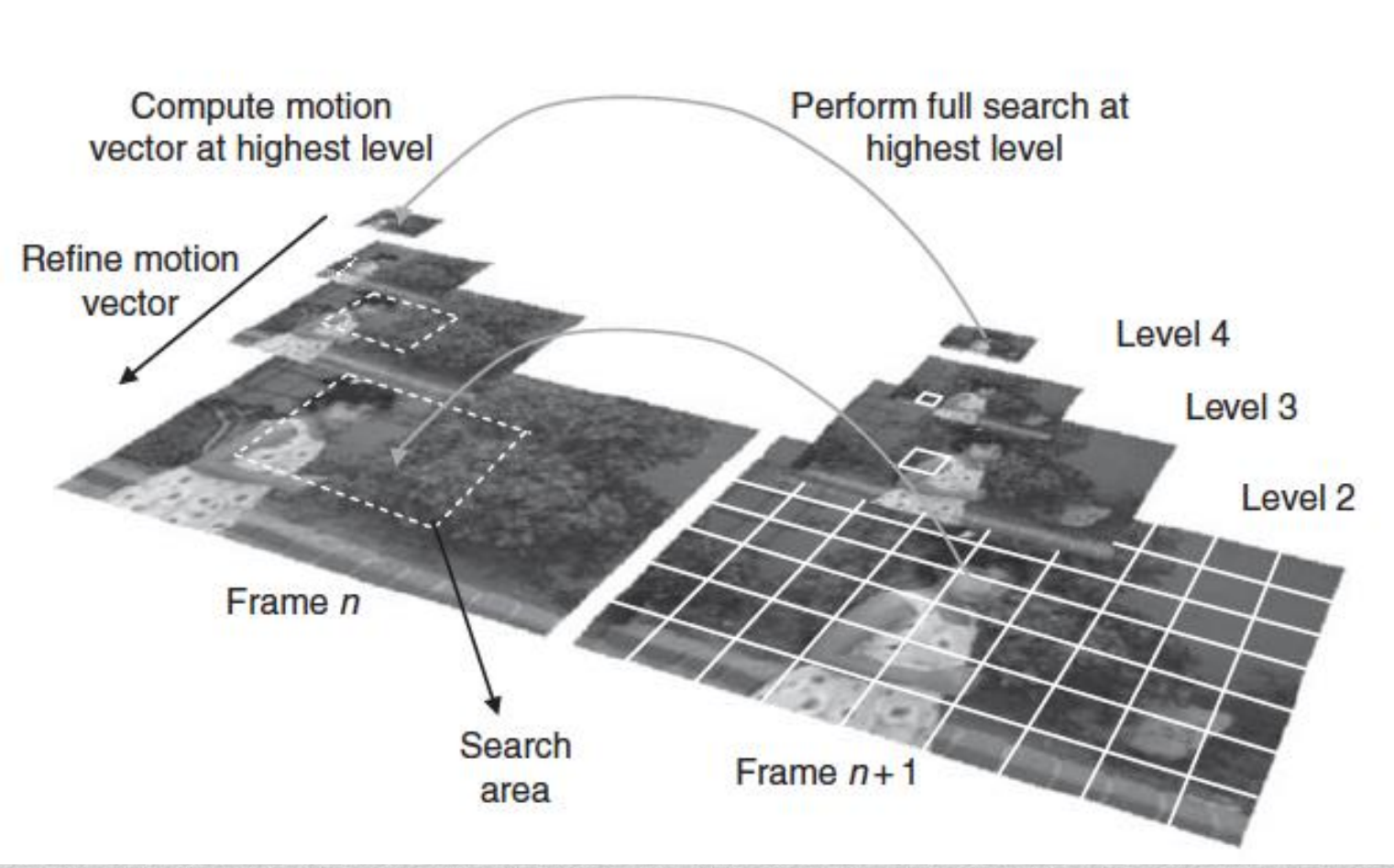
- In the previous example, 3x8 + 9 = 33 calculations needed
- Actually we only need to evaluate 8 and not 9 from the second iteration and onward because the center is already computed previous step.
- This is called FME: Fast motion estimation
- It yields sub-optimal results due to its monotonic assumption.

2- logarithmic search

- Search in a multi-resolution manner
- a full search is carried out at the highest level of the hierarchy, followed by refinements trickling through the lower levels.
- The target frame, as well as the reference frame, is said to be at level 1.
- Successive levels are created by subsampling and 2D filtering the previous level. This creates a pyramid structure for both frames, as shown in Figure

3- Hierarchical/multiresolution

- Successive levels are created by subsampling and 2D filtering the previous level. This creates a pyramid structure for both frames, as shown in Figure



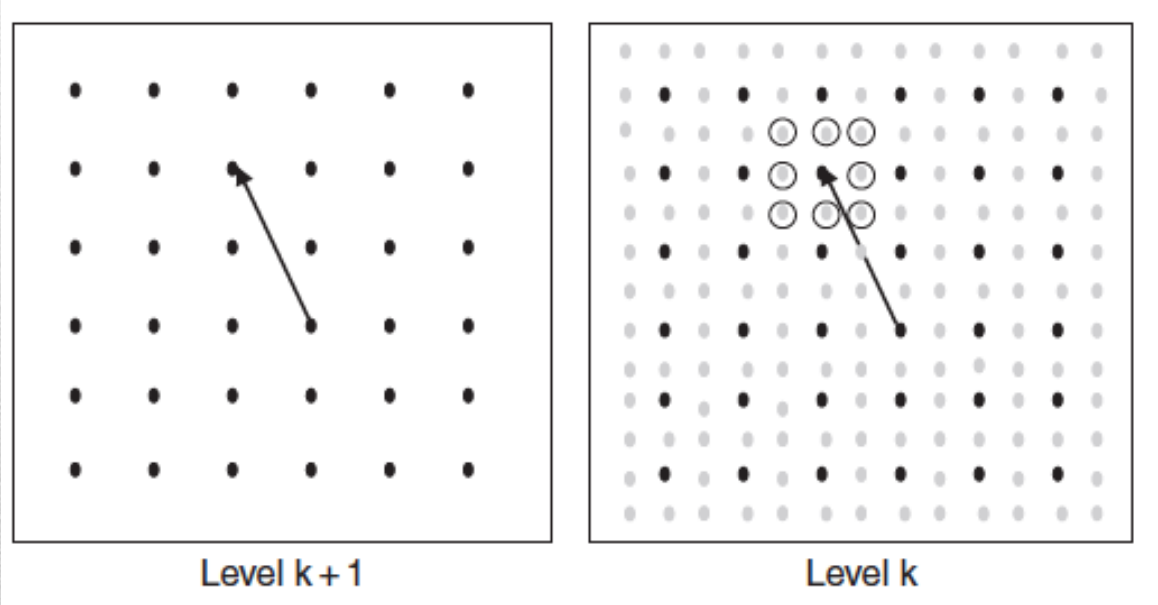
- Shown at the bottom of previous figure are level 1 images of the ref. frame n and the target frame n + 1.
- The target frame to be predicted is shown broken into macroblock areas. Both the reference and target images are shown subsampled to level 4.
- An example macroblock is shown at each level in the target hierarchy along with its corresponding search area in each level of the reference hierarchy.
- Note that the macroblock reduces in size and so does the search area in the reference hierarchy at each level

3- Hierarchical/multiresolution

- If the macroblock is of size n x n and the search area parameter is k to begin with, at level 2 of the pyramid, the reference macroblock will be n/2 x n/2 while the corresponding search area parameter will be k/2.
- A full search is carried out at the highest level, which proceeds quickly because of smaller macroblock size and smaller search area.
- This computed motion vector gets refined by comparing it with other missed candidates in the lower levels of the reference hierarchy.

3- Hierarchical/multiresolution

- You can see from the example that there are eight new candidate positions in the local neighborhood of the motion vector at the new lower level.
- A MAD is computed at these new locations to see whether the motion vector can be refined to a new position.
- The refinement process is repeated till the lowest level.



- In digital HDTV sports content, the search parameter k is normally kept to 32 (or more) because of higher resolution in the content.
- A full search would need $(2 \times 32 + 1)^2 = 4225$ MAD evaluations.
- A four level hierarchical search would result in a k= 4 at level 4.
- A full search at this level entails $(2 \times 4 + 1)^2 = 81$ MAD evaluations.
- At each successive lower level, there will be another eight refinement positions, bringing an additional (8 x3)= 24 evaluations since there are three levels at which this is done.
- The total number of evaluations for a hierarchical search is the sum of the searches done at the highest level and refinements at the intermediary levels bringing the total to (81 +24) = 105.

3- Hierarchical/multiresolution

Video coding standards

PART 4

- the ITU (International Telecommunication Union), which has developed the H.26x video standard family,
- and the ISO (International Organization for Standardization), which has developed the MPEG standards

Video standards

H.261

- H.261 was designed to support video teleconferencing over ISDN lines
- It was mainly designed for the QCIF resolution (176 x 144)
  - But can also support CIF-sized video (352 x 288).
- Compression occurs by compressing frames in the intra-mode (I frames) and in the inter-mode (P frames). There is no support for B frames.
- I frames are compressed in a JPEG like manner using no references.

- Designed for Video CD (VCD)
- Introduced MP3 audio
- I, B, P frames
- Usage of slices: frame is divided into slices of macroblocks where each slice might contain a variable number of macroblocks
- Sub-pixel motion compensation using interpolation
- GOP use (for rewind and forward)

MPEG 1

- Used for DVD and digital TV broadcasting
- Higher bandwidth and quality
- Also I, P and B frames with half pixel approximation
- Support for interlacing: fields can be predicted from fields
- Scalable encoding: encode once, decode at any quality
- Higher bitrates than MPEG-1
- Widely used until HD formats took over

MPEG 2

- The H.263 standard was designed by ITU to extend the H.261-based videoconferencing
- support a wider range of picture formats, including 4CIF (704 x 576) and 16CIF (1408 x 1152).
- Like the H.261, it uses I and P frames, which are both encoded in the standard manner. However, it also used B frames.
- Optimized for low bitrates
- Used in early video conferencing systems

H.263

- Object-based video coding
- Progressive and interlaced
- Compression is 25% better than mpeg2
- Video object planes (VOP) image frames can have arbitrary shapes
- Temporal scalability: object recognition
- Used in DivX and Xvid

MPEG 4

- Most widely used codec globally
- High compression efficiency
- Strong motion compensation and entropy coding
- Smaller blocks 4x4 instead of 8x8
- In intra-coding, spatial redundancy is further exploited by predicting blocks inside the same frame and DCTing the prediction error only
- Variable macro-blocks size predictions
- Quarter pixel motion compensation (QPEL)
- Use any frame from past or future
- Context adaptive binary arithmetic coding (CABAC)

H.264 or MPEG4-AVC

Standard	Year	Key Features
H.261	1990	First ITU-T video coding standard; designed for ISDN; CIF/QCIF.
MPEG-1	1993	Video CD (VCD); introduced MP3; simple motion compensation.
MPEG-2	1995	Digital TV, DVD, DVB; supports interlaced video.
H.263	1996	Enhanced low-bitrate codec for videoconferencing.
MPEG-4 Part 2	1999	Object-based coding; used in DivX, Xvid.
H.264 / AVC	2003	Very high compression; Blu-ray; streaming; HD/Full HD.
HEVC (H.265)	2013	~50% better compression vs H.264; supports 4K/8K.
VP8	2010	Open-source Google codec; used in WebM.
VP9	2013	Successor to VP8; used heavily by YouTube; alternative to HEVC.
AV1	2018	Royalty-free AOMedia codec; used by Netflix, YouTube, Chrome.
VVC (H.266)	2020	Next-gen standard; 30–50% better compression than HEVC; optimized for 8K/HDR.
MPEG-5 EVC	2020	Essential Video Coding; simpler licensing; competitive with HEVC/H.266.
MPEG-5 LC-EVC	2020	Low-complexity enhancement codec; can stack on AVC/HEVC/AV1.

EXERCISES

Consider the following piece of two consecutive frames. Answer the following questions:

a. Consider the highlighted 2*2 macro block in frame n+1. Search in frame n the block that best matches this one using a brute force search with k=2.

Suppose now that we want to search for the highlighted block of the frame n+1 in the whole frame n.

b. For each block in frame n+1, how many MAD calculations should be done ?

c. Deduce the total number of MAD calculations to be done in order to calculate the motion vector for all the blocks in frame n+1.

d. If each 2*2 MAD calculation takes 1 ns, what is the time needed to predict all the motion vectors of all the blocks in the frame n+1?

e. Answer the parts c, d, and e if we consider a logarithmic search instead of brute search.

Exercise I

139	144	149	153	155	155	155	155
144	151	153	156	159	156	156	156
150	155	160	163	158	156	156	156
159	161	162	160	160	159	159	159
159	160	161	162	162	155	155	155
161	161	161	161	160	157	157	157
161	162	161	163	162	157	157	157
162	162	161	161	163	158	158	158

frame *n*

155	155	139	144	149	153	155	155
156	156	144	151	153	156	159	156
156	156	150	155	160	163	158	156
159	159	159	161	162	160	160	159
155	155	159	160	161	162	162	155
157	157	161	161	161	161	160	157
157	157	161	162	161	163	162	157
158	158	162	162	161	161	163	158

frame *n+1*

Exercise II

- Consider the encoding of a 640x480 video at 30 fps using I and P frames. To encode the P frames, we consider 8x8 macroblocks and **use logarithmic search** to find the best motion vector. Assume that the search area parameter k is equal to 8.
  - Calculate the number of candidate blocks to be matched against a target macroblock.
  - Calculate the number of MAD evaluations per frame (assume that all target macroblocks, including the boundary ones, have the same number of candidates).
  - Consider now 10 minutes of video knowing that one I frame is inserted after every 4 consecutive P frames as follows:
- I PPPP I PPPP I … . Assume that 1 MAD evaluation takes 0.1 ms.
  - What is the Total number of frames? I frames? P frames?
  - Deduce the time spent on motion vector computation for these 10 minutes of video.
  - Can this be used for real-time encoding and distribution setup? Justify.

- Consider the encoding of a 640x480 video at 30 fps using I and P frames. To encode the P and B frames, we consider 8x8 macroblocks and a search area of k = 8. How many MADs is done to find the best motion vector for each macro-bloc in a P frame if the search method used is:
- Sequential search.
- Logarithmic search.
- Hierarchical with 4 levels.

Suppose now that the search method is the logarithmic one.

- How many macroblocks are there in each image.

Consider now 10 minutes of video knowing that one I frame is inserted after every 8 consecutive P frames and there are 4 B frames between each couple of P frames as follows:

I B B B B P B B B B P B B B B P B B B B P B B B B P B B B B I ……

- How many MADs are done for the 10 minutes of the video.

EXERCISE III