Video Processing and Motion Estimation

Video Signal

- Number of bits required to store a color image of size, $M \times N$

  $= M \times N \times 8 \times 3$
- Number of bits to store a video of frame rate 25 frames/sec is

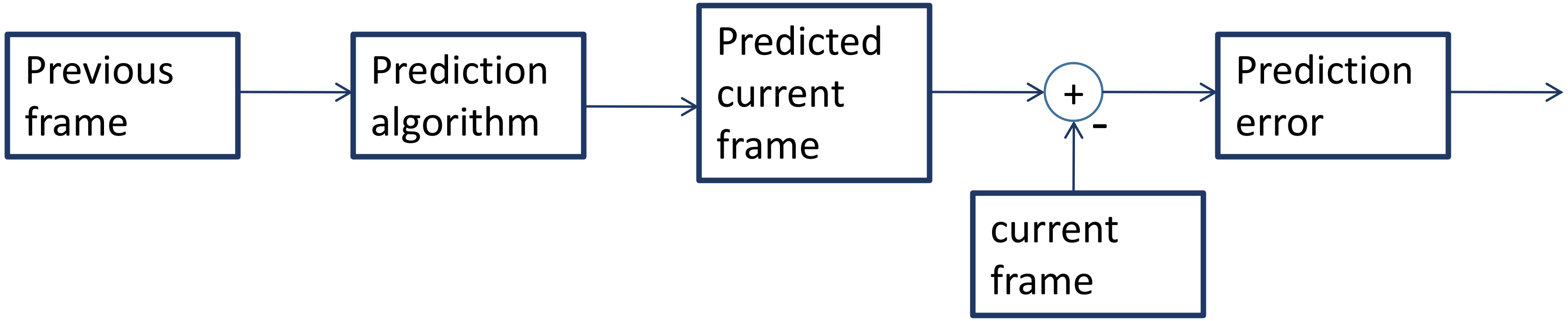
  $= M \times N \times 8 \times 3 \times 25$
- For $M = 100$ and $N = 100$, one second video requires 6,000,000 bits
- Therefore, video requires large memory to save and large bandwidth to transfer
- Video compression is required to store the image

Video Signal

- Video is a sequence of correlated images
- Video compression algorithms use temporal correlation to remove redundancy
- The previous reconstructed frame is used to generate a prediction for the current frame
- The difference between the prediction and the current frame, is prediction error or residual
- Prediction error is encoded for the video

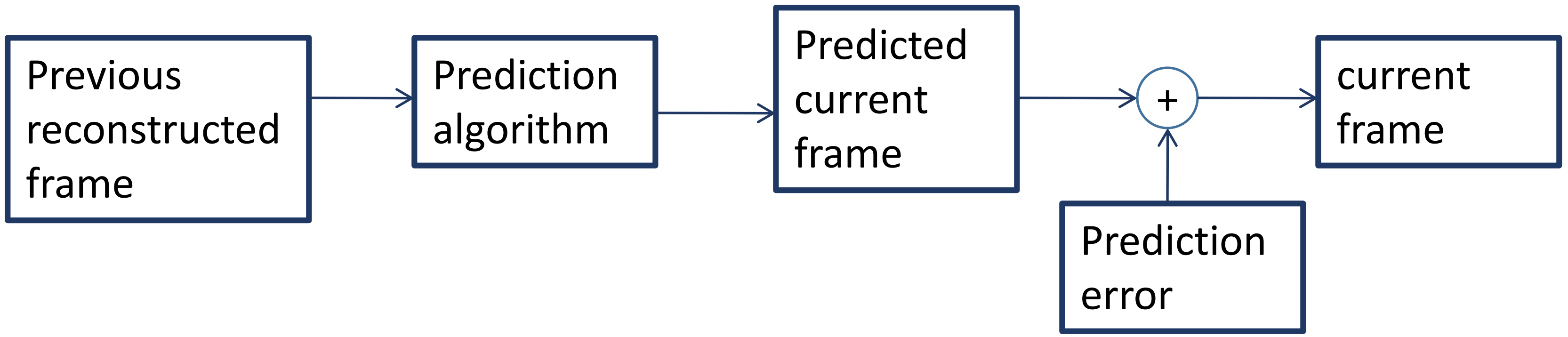
Video Signal

- Video is a sequence of correlated images
- Video compression algorithms use temporal correlation to remove redundancy
- The previous reconstructed frame is used to generate a prediction for the current frame
- The difference between the prediction and the current frame, is prediction error or residual
- Prediction error is encoded for the video



Video decoder

- The previous reconstructed frame is available at the receiver
- Receiver knows the prediction algorithm
- It can use it generate the predicted frame
- Add predicted frame and predicted error to reconstruct the current frame
- Prediction operation considers motion of the objects in the frame
- Prediction is known as motion compensation



Motion Estimation

- Image data in an image sequence remains mostly the same between frames in video
- Scene content does not change much from frame to frame
- To exploit the image data redundancy in image sequences there is a need to estimate motion in the image sequence
- Motion estimation is computationally complex
- Therefore perform motion estimation *only* where motion of objects is present
- Motion estimation is used for motion compensation

Techniques for Motion Estimation

- Pixel Difference
- Fixed Block matching/ Block matching
- Hierarchical Block Matching
- I, P and B Frames

Techniques for Motion Estimation

- Pixel Difference
- Fixed Block matching
- Hierarchical Block Matching
- I, P and B Frames

Pixel difference

- One of the methods to estimate motion is pixel difference between the two consecutive frames
- Difference of each pixel of current and corresponding pixel at the same location in the previous frame is prediction error
- Prediction error is transmitted to reconstruct frame at the receiver
- Some objects in a frame may move to new location in the next frame
- Pixel to pixel comparison (prediction error) of the these two frames may be non zero for each pixel
- Which may lead to the increase in the number of pixel locations with predictions error

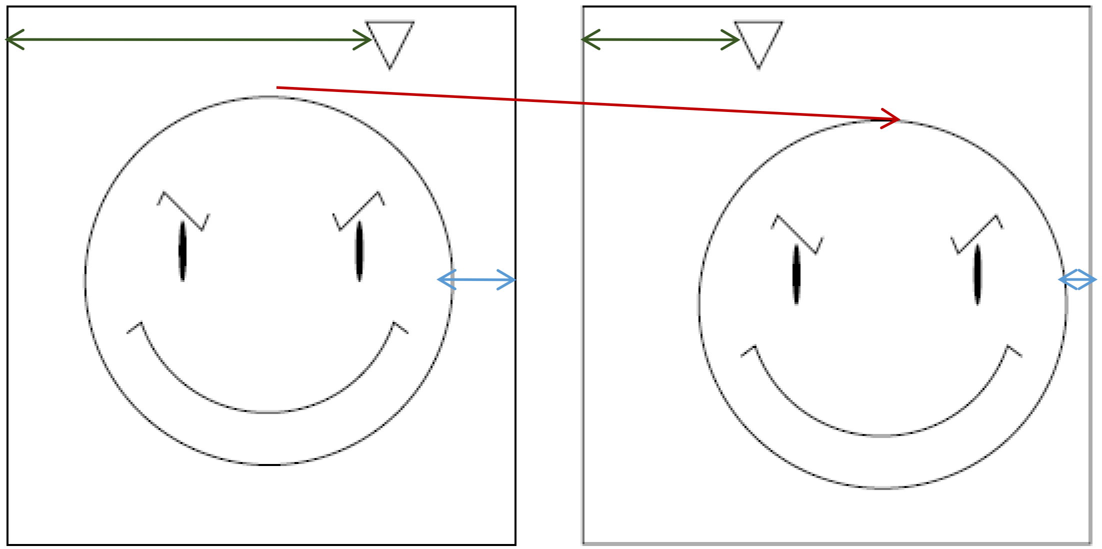
Pixel difference

- Assume an object in one frame has a pixel at location (i_0, j_0)
- Pixel of the same object may be located at (i_1, j_1) in previous frame
- Prediction is the difference of pixel in two frames at (i_0, j_0)

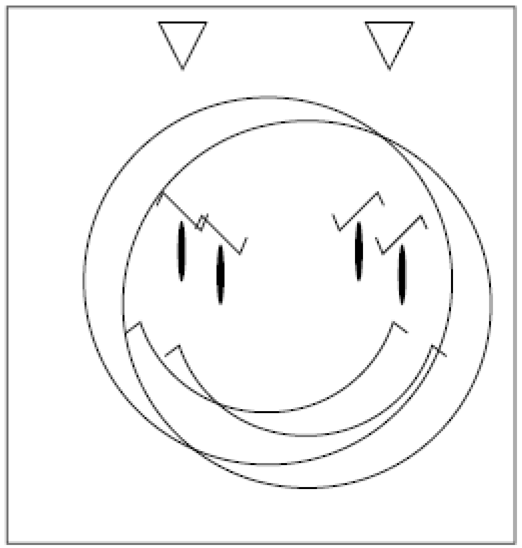
  Not at (i_0, j_0) and (i_1, j_1)
- Since object has moved to new location in the next frame, other pixels of the two frames may also be different
- If object which has moved is small in size, pixel difference is a useful method
- This is because not many pixels will be different in two frames
- If moving object large portion of the image
- then, the difference of most of the pixels in two frames will be non zero

Pixel difference

- Face has moved slightly downward and to the right of the frame
- Triangular object has moved to the left
- Visual difference between the two frames is not significant
- Therefore not much information should be required for the transmission of the second frame
- However, difference of pixel values of two frames is non zero



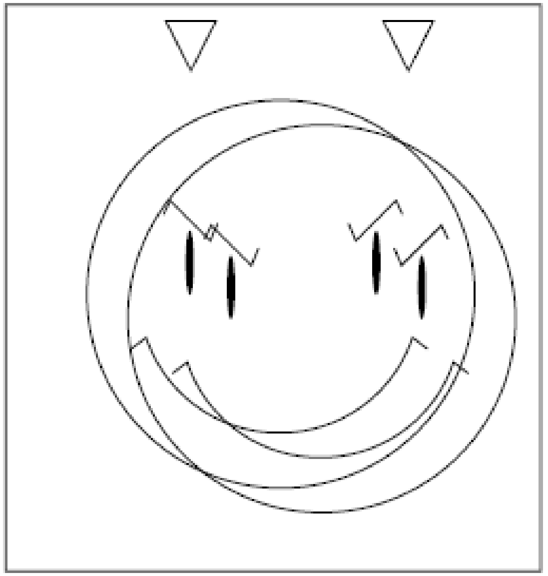
Two frames of a video sequence



Difference of pixels in two frames

Pixel difference

- Displacement of the objects in the frame results in more non zero values than the original image
- All non zero values are required for construction of second frame
- Therefore more information needs to be transmitted
- Compression using difference of pixels is not effective if large objects covering major part of the image move in two consecutive frames



Difference between the two frames

Problems with Pixel Difference

- Motion estimation can be wrong if frames are noisy
- Problem can be overcome by smoothing each frame with using Gaussian filter
- Filter results in blurring of edges
- Which may cause false detection
- May detect moving object as non-moving
- Therefore block based motion estimation is better than pixel based motion estimation

Techniques for Motion Estimation

- Pixel Difference
- Fixed Block matching/ Block matching
- Hierarchical Block Matching
- I, P and B Frames

Block Matching

- Motion of objects in the image can be used to predict the pixel values in the frame being encoded
- Frame being encoded is divided into macro blocks of size M ×M
- For each block of the current frame, search the previous reconstructed frame
- To search a matching block, the sum of absolute differences (SAD) between corresponding pixels in the two blocks is calculated

$$SAD = \sum_{i=1}^{M}\sum_{j=1}^{M}\left|\left(curr_{i,j} - macro_{i,j}\right)\right|$$

- If SAD is low then two blocks have similar pixel values
- Block that provides minimum SAD (most closely matches the block being encoded) is chosen

Motion Estimation using Blocks

- If SAD is greater than some pre-specified threshold
- Then block is declared uncompressible
- And is encoded without the using prediction
- This decision is also transmitted to the receiver
- If SAD is below the threshold, then a *motion vector of block* is transmitted to the receiver
- Motion vector denotes the relative location of the block between two consecutive frames
- Motion vector is the difference between coordinates of upper left corner of two blocks

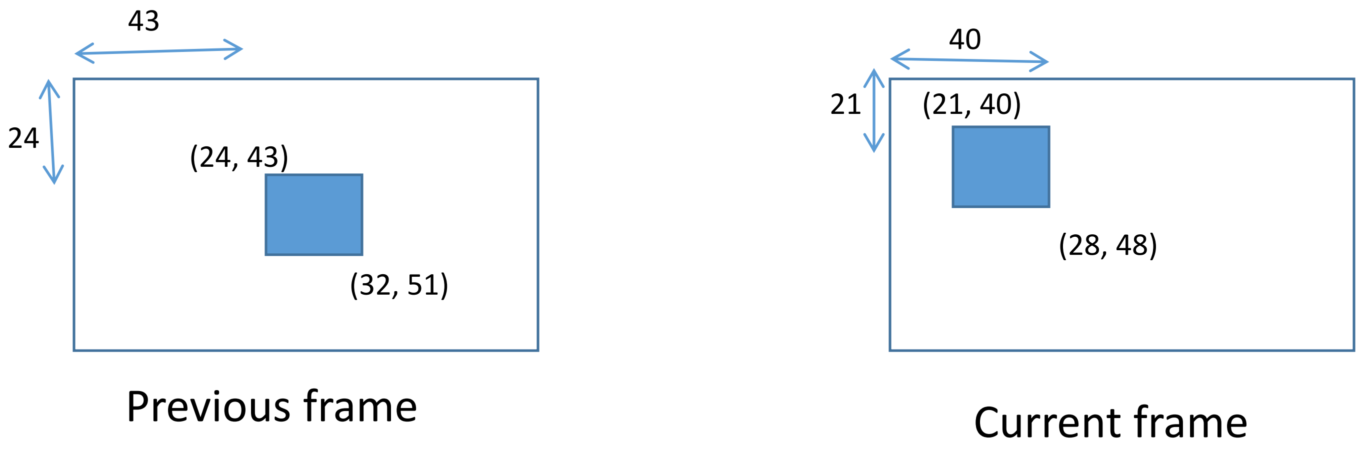
Motion Estimation using blocks

- Motion vector is

  (upper left corner of the best matching block in previous frame)

  - (upper-left corner of the block of current frame being encoded)

  = (21, 43) - (24, 40) = (−3, 3)
- Previous frame and motion vector are enough to predict current frame
- A '-3' component of motion vector means that the best matching block in the current frame is above the block in the previous frame
- Similarly, +3 means that the best matching block in previous frame is at a location which is at left side of current block

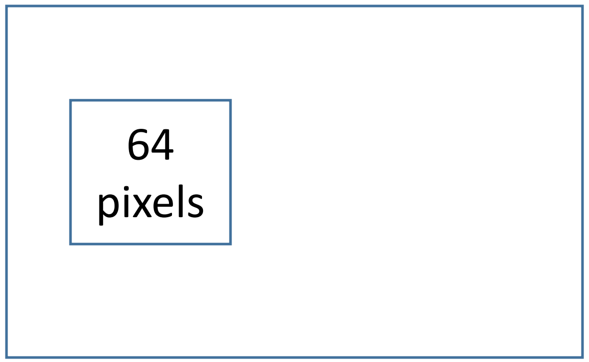


Motion Estimation using blocks

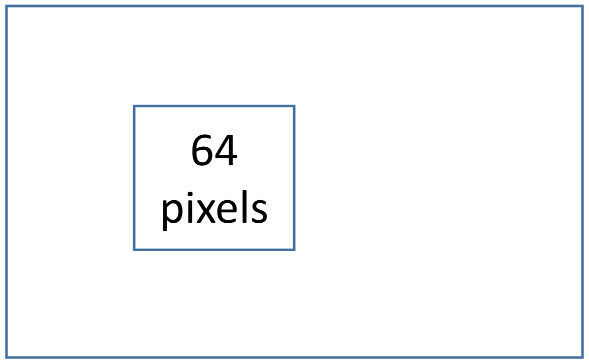
- To find a matching block for an 8×8 block
- Take a block at the same location of the previous frame
- Determine the difference between two blocks

$$SAD = \sum_{i=1}^{M}\sum_{j=1}^{M}\left|\left(curr_{i,j} - prev_{i,j}\right)\right|$$

- Repeat the same for surrounding blocks in search area of the previous frame
- Choose block with minimum SAD as matching block



Block of previous frame



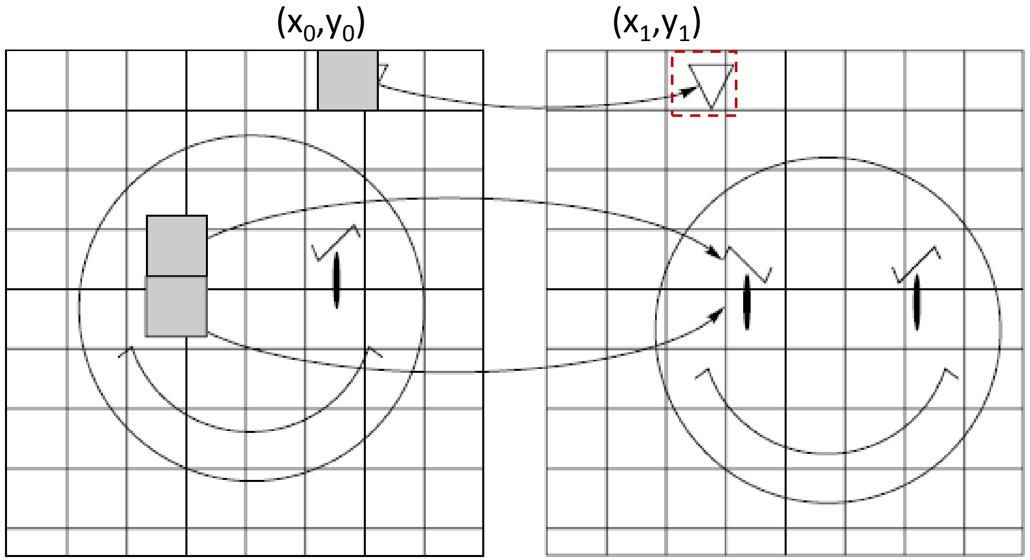
Block of current frame

Block Matching for entire frame

- Divide previous frame into macro blocks of MxM size
- Thus previous frame consists of multiple macro blocks
- Take a macro block of current frame and compare it with macro block in previous frame at the same location and macro blocks in the surrounding area
- Surrounding area is called search area
- Matching macro block of current frame is chosen to compute motion vector
- Finally, previous frame and motion vectors are saved in memory, which is compressed form of current frame

prediction Motion compensated

- Divide the image into blocks
- Match each block of current frame with the similar block of the previous frame
- Determine motion vector $(m_x,m_y) = (x_0,y_0) - (x_1,y_1)$
- Determine motion vectors for the remaining blocks
- Transmit motion vector of each block instead of transmitting the entire current frame
- Thus current frame is completely predicted by the previous frame

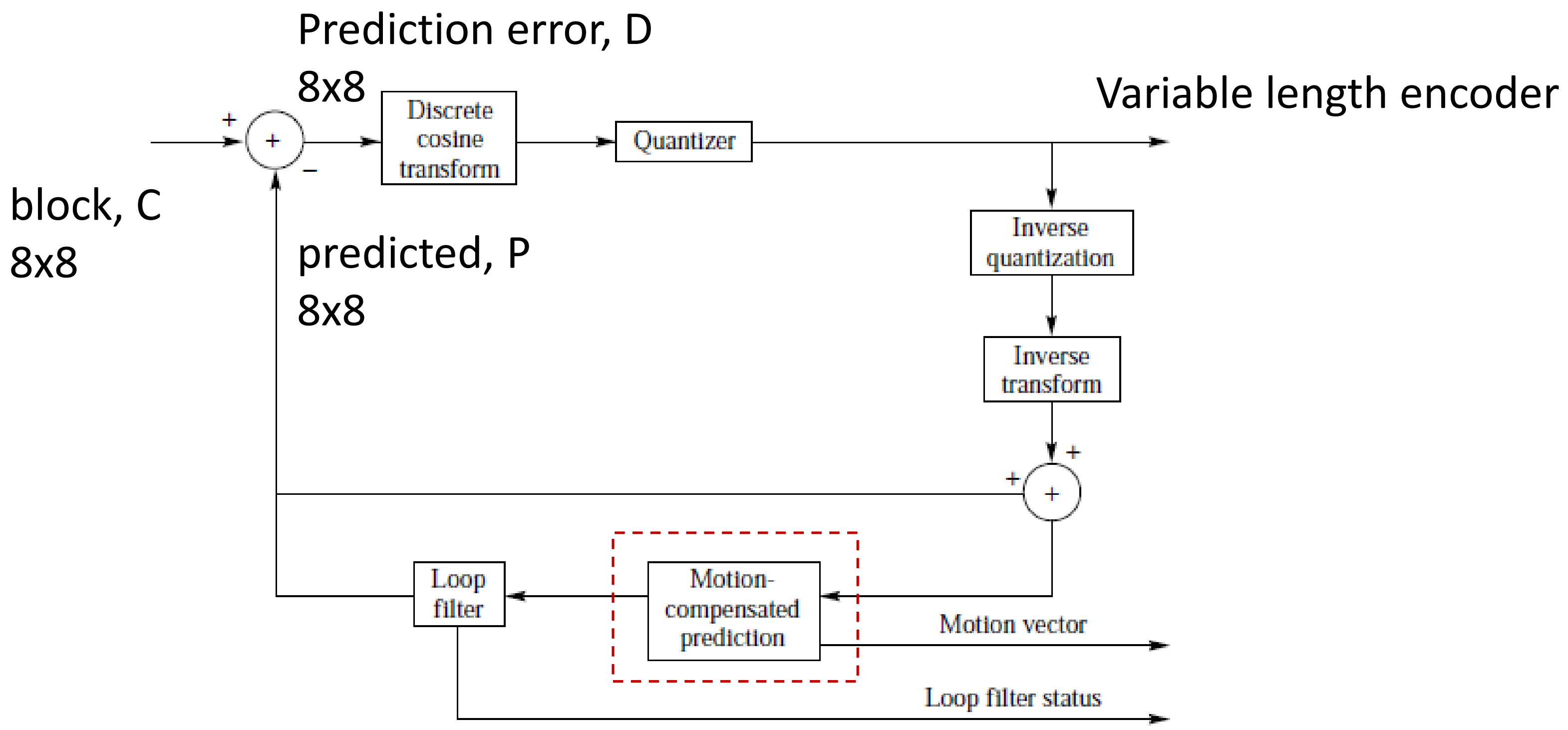


Video compression standards

- Most of the standards for video compression are based on motion estimation
- Block matching is widely used for stereo vision, vision tracking, and video compression
- Video coding standards such as MPEG-1, MPEG-2, MPEG-4, H.261, H.263 and H.264 use block based motion estimation algorithms

Compression Standard

- Prediction of block is based on motion vector (motion compensation)
- Motion compensation provides compression



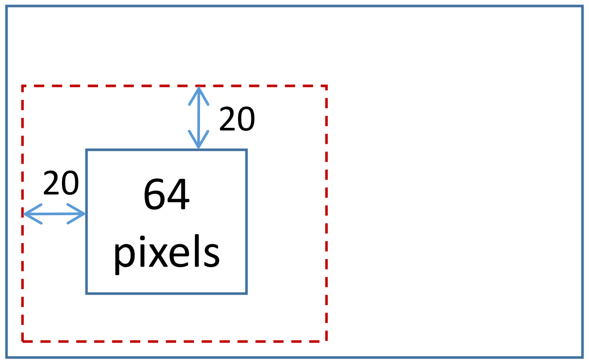
Motion Estimation

- Motion estimation for the entire frame requires a large amount of computation
- Instead of searching in the entire previous frame

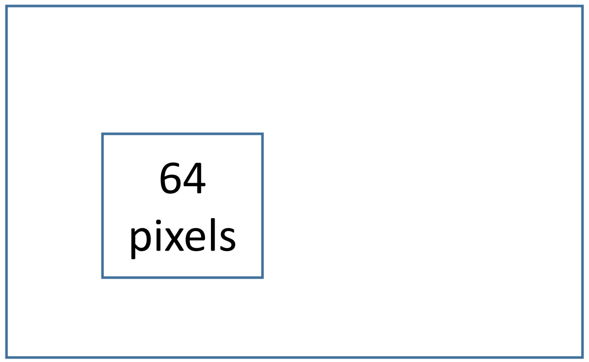
  search area is within 20 pixels of the block to be encoded

Motion Estimation

- Motion compensation requires a large amount of computation
- Instead of searching in the entire previous frame
- Search area is within 20 pixels of the block to be encoded



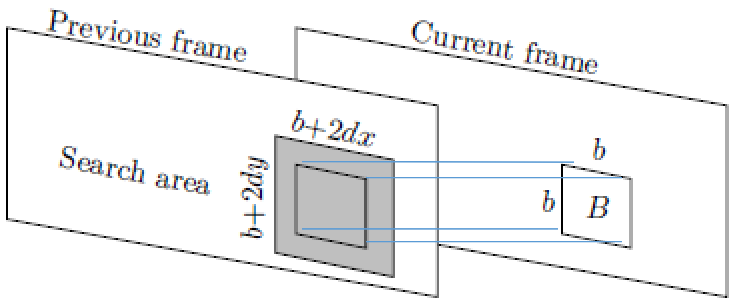
Block of previous frame



Block of current frame

Search Area

- B is a square with side b, the search area contains
  (b + 2dx)(b + 2dy) pixels

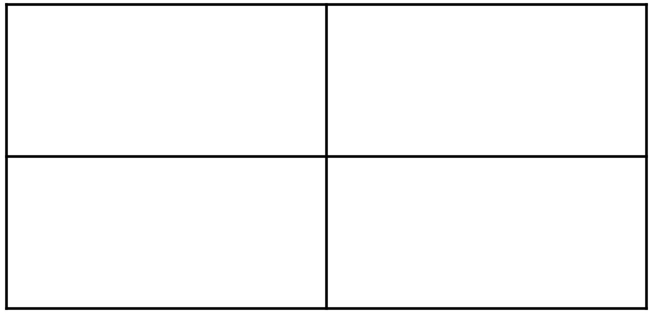


Motion Estimation

- To reduce the number of comparisons, increase the size of the block
- If size of the block is increased

  Then fewer blocks per frame are required to be encoded
- Therefore, number of times we have to perform the motion estimation decreases
- However, it requires more computations per matching block

Frame with small blocks



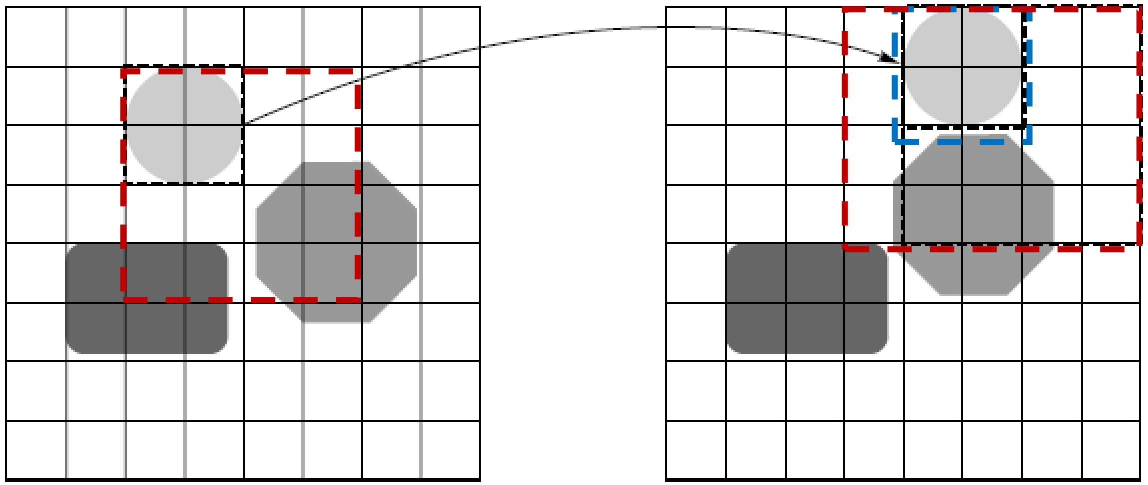
Frame with large blocks

Motion Estimation

- If macroblocks is large then number of macroblocks is less
- Therefore number of motion vectors is also less
- In large macroblock, more objects can be accommodated
- And all objects may not move in the same direction
- And probability that a block contains objects which are moving in different directions is more

Effect of block size on Motion Estimation

- If size of the block is 2×2,
- Then it is possible to find a block that exactly matches the 2×2 block that contains the circle.
- However, if size of the block is increased to 4×4
- then block that contains the circle also contains the upper part of the octagon
- This is because circle and octagon have moved in different direction
- Therefore, can not find a similar 4×4 block in the previous frame



Effect of block size on Motion Estimation

- Larger blocks reduce the amount of computation
- However for most of the blocks (in current frame) may not have matching blocks in previous frame
- Thus 64 pixels for each unmatched blocks are transmitted instead of a motion vectors for matching block
- More is the number of unmatched blocks, poor is compression performance

Effect of search space size on Motion Estimation

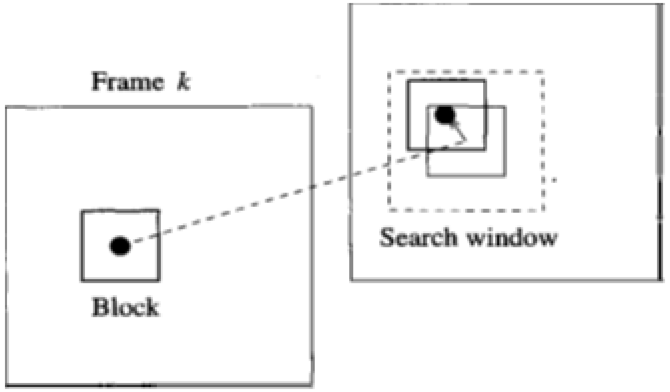
- Another way to reduce the number of computations is by reducing the search space
- If size of the region to search is reduced then number of computations is reduced
- However, reducing the search region also increases the probability of missing a match
- There is a trade-off between computation and the amount of compression

Motion estimation for Video Compression

- Motion estimation is used for video compression
- Motion vectors are used to represent motion estimation
- Motion vectors are determined using block matching algorithms
- Motion vectors require less bits than that required for entire frame
- However, motion estimation is the most computationally expensive in the entire compression process
- Hence, fast and computationally inexpensive algorithms for motion estimation are used for video compression

Block Matching Algorithm

- Typical macroblock size is 16 pixels and a search area of p = 7 pixels
- Block-matching algorithms differ in
  - Matching criteria (ex: maximum cross-correlation, minimum error)
  - Search strategy (ex: three-step search, logrithemic search), and
  - Determination of block size (ex: hierarchical, adaptive)



Evaluation Metrics

- A metric for matching a macroblock with another block is based on a cost function.

1. Mean Absolute Difference (MAD) = $\frac{1}{N^2}\sum_{i=0}^{n-1}\sum_{j=0}^{n-1}|C_{ij} - R_{ij}|$

Where

N is the size of the macro-block,

C_{ij} and R_{ij} are the pixels of current macroblock and reference macroblock respectively

2. Similarity Absolute Difference (SAD) =$\sum_{i=0}^{n-1}\sum_{j=0}^{n-1}\left|C_{i,j} - R_{ij}\right|$

Evaluation Metrics

3. Mean Squared Error (MSE) = $\frac{1}{N^2}\sum_{i=0}^{n-1}\sum_{j=0}^{n-1}(C_{ij} - R_{ij})^2$

Motion compensated image using the motion vectors is characterized by Peak signal-to-noise ratio (PSNR),

$$\mathrm{PSNR} = 10\log_{10}\frac{(\text{peak to peak value of original data})^2}{\mathrm{MSE}}$$

Techniques for Motion Estimation

- Pixel Difference
- Fixed Block matching/ Block matching
- Hierarchical Block Matching
- I, P and B Frames

Hierarchical block matching algorithm

- For each method, macro block of current frame is compared with the candidate block in the search area of previous frame
- Fixed block matching, macro block is compared in the fixed size of search area of previous frame
- Hierarchical block matching, the search is carried out in hierarchical fashion
  - sizes of the blocks and the search are vary at different levels of hierarchy
  - Once the hierarchical structure is constructed,

    candidate motion vector having the smallest matching error is selected as the coarse motion vector
  - The chosen matching block is used as a reference for the next lower level get finer match
  - This process is continued to the lowest level

Hierarchical Block Matching (OHBM)

- Also called fast Block Matching
- Speeds up the block search process
    1. Full-search algorithm
    2. Three Step Search (TSS)
    3. Two Dimensional Logarithmic Search (TDLS)
    4. New Three Step Search (NTSS)

Full-search algorithm

- Most simple block matching algorithm
- Provides the optimal result by matching all possible candidates within the search window
- Similar to block search algorithm
- Also called exhaustive algorithm

Three Step Search (TSS)

- Take a macroblock of current frame
- Nine candidate macroblocks of previous frame are selected in the first step
- One centering at the center pixel and the other eight centering at eight around the center pixel
- The matching function, SAD is calculated
- Block with minimum SAD value is chosen as the center
- In the second step, eight more blocks are tested around block found in the first step
- Each time, size of the new set of block is reduced
- This time, the spacing of the pixels is tuned finer than before
- The above procedure is repeated until the step size is smaller than one and the final motion vector is found

Three Step Search (TSS)

- Start with search location at center of candidate macroblocks of previous frame
- Set step size 'S' = 4
- Search parameter 'p' = 7 (search area is 7 macroblocks around center macroblock)
- Search area is 15x15 macroblocks in previous frame
- Search 225 macroblocks in search area using the following steps

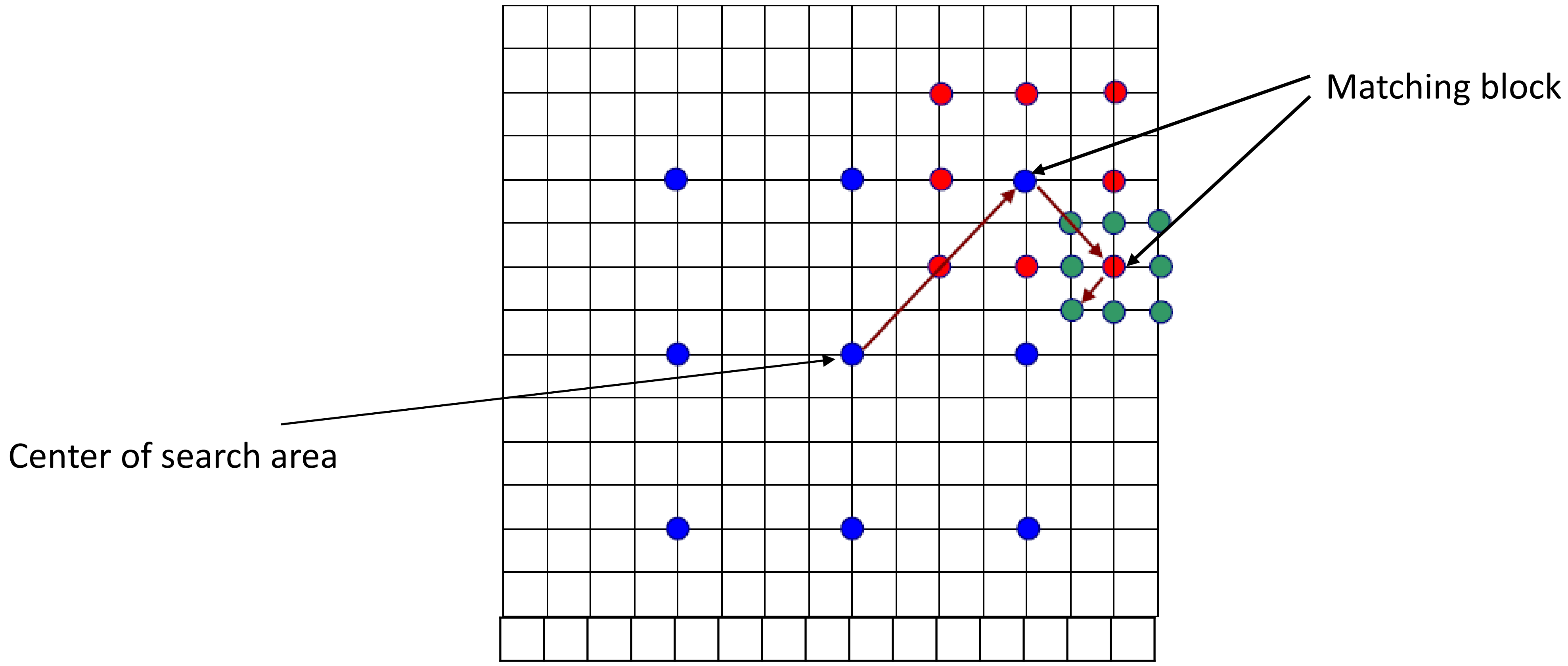
1. Search 8 locations (=+/- S) pixels around location (0,0) and at the location (0,0)
2. Calculate SAD for each candidate block
3. Choose minimum SAD from a set of 9 SADs, This is a matching block
3. Set the new search origin at the above picked location

   Set the new step size as S = S/2

- Repeat the search procedure until S = 1
- The resulting location for S=1 is the one with minimum SAD
- Macro block at this location is the best match between block of current and previous frame

Three Step Search (TSS)

Each dot is a macro block



15 × 15 search area

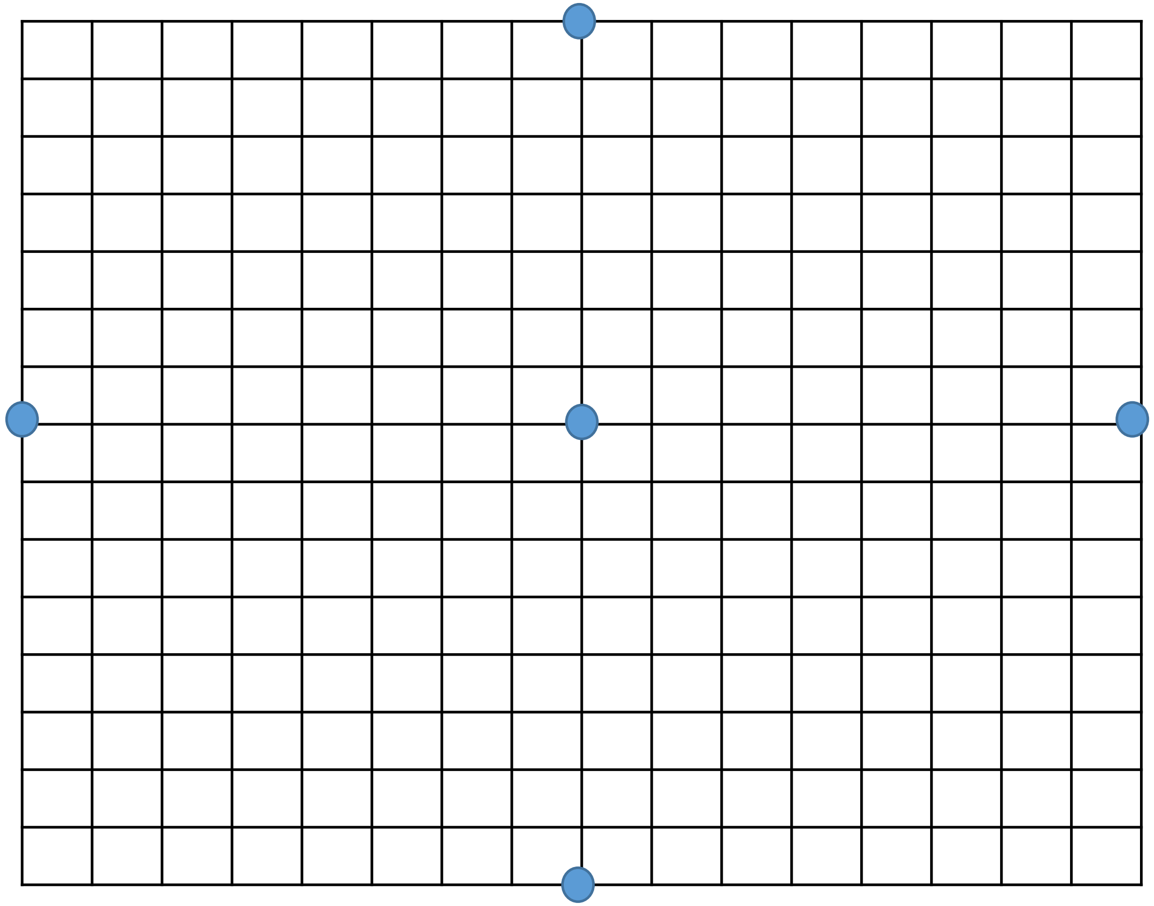
Three Step Search (TSS)

For p=7

- Fixed block matching algorithm
  - search area = 2x7 + 1 = 15
  - Searches each possible block
  - Evaluates SAD (cost) for 225 macro-blocks
- For TSS
  - S= (p+1)/2 = 4, 9 blocks (8 surrounding the center and center) are searched
  - Then for S=2, 8 block surrounding center are searched
  - Then for S=1, 8 block surrounding center are searched
  - TSS evaluates 9+8+8 =25 macro blocks
- There is a reduction in computation by a factor of 225/25 = 9 in TSS

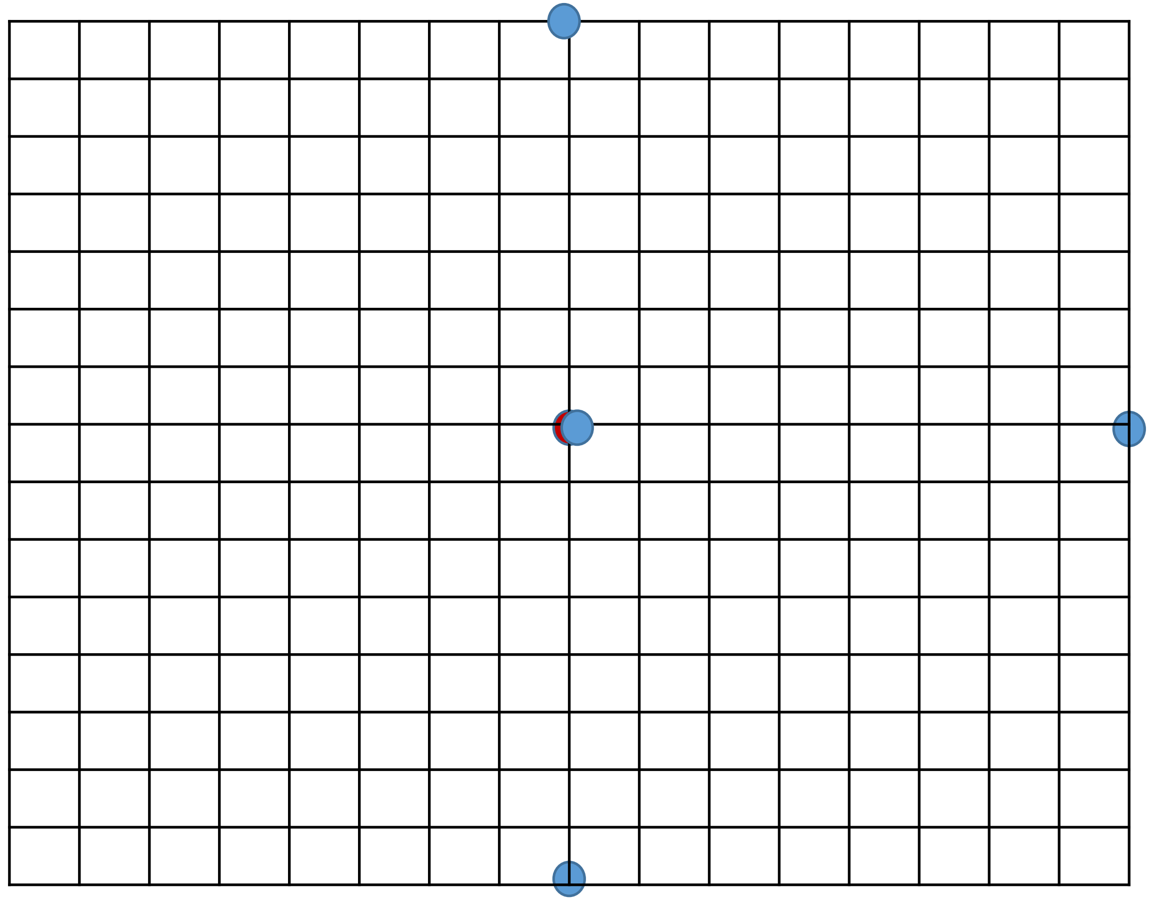
Two Dimensional Logarithmic Search

- Similar to TSS
- More accurate for estimating motion vectors for a large search window size
- Algorithm
  - Start with search location at the center
  - Initial step size say, S = 8
  - Search for 4 locations at a distance of S from center on the X and Y axes
  - Find the location of point with least cost function



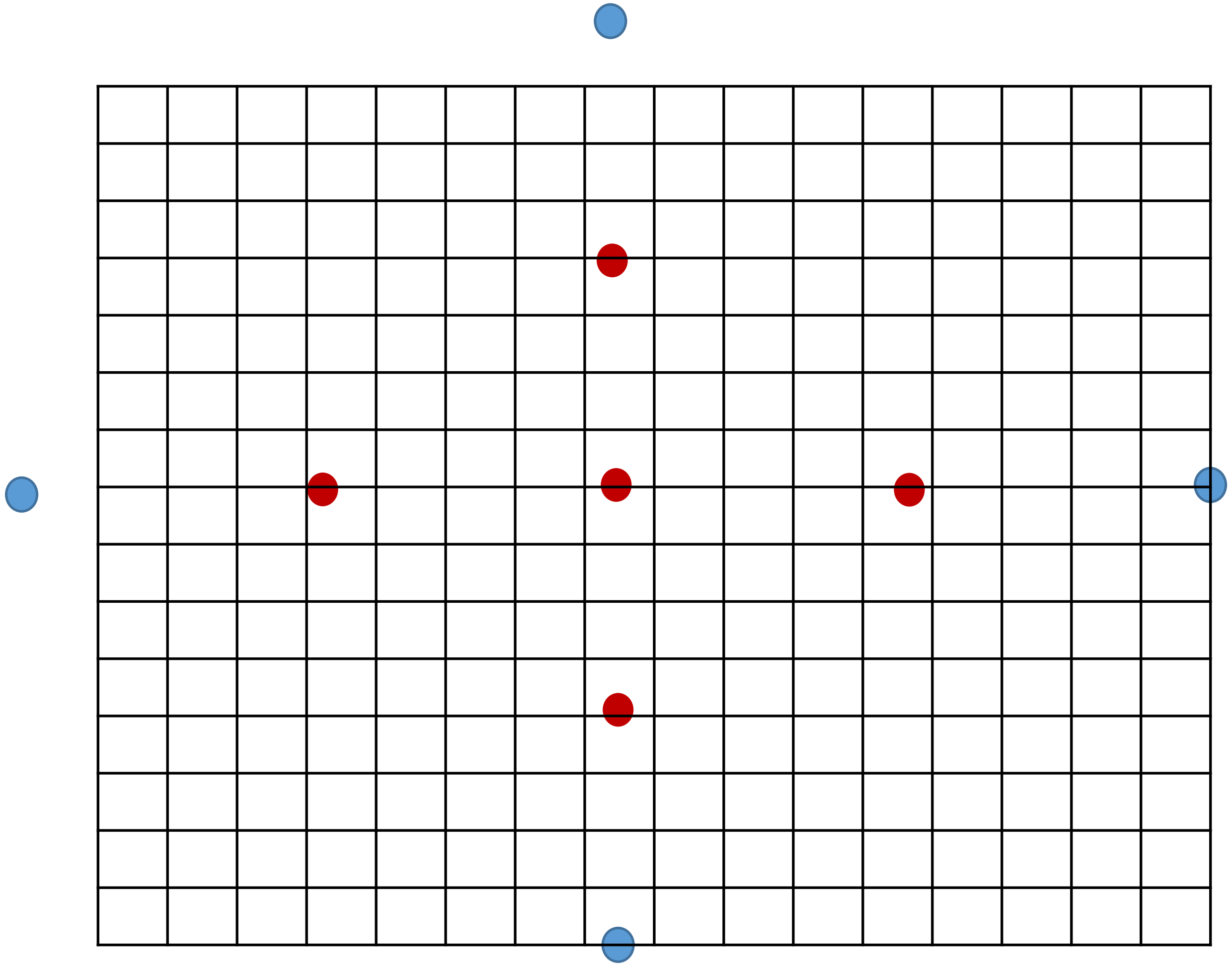
Two Dimensional Logarithmic Search

- If a point other than center is the best matching point,
  - Select this point as the new center and retain the same step size
  - Repeat earlier steps
- If the best matching point is at the center
  - Start center as a new point
  - set S = S/2



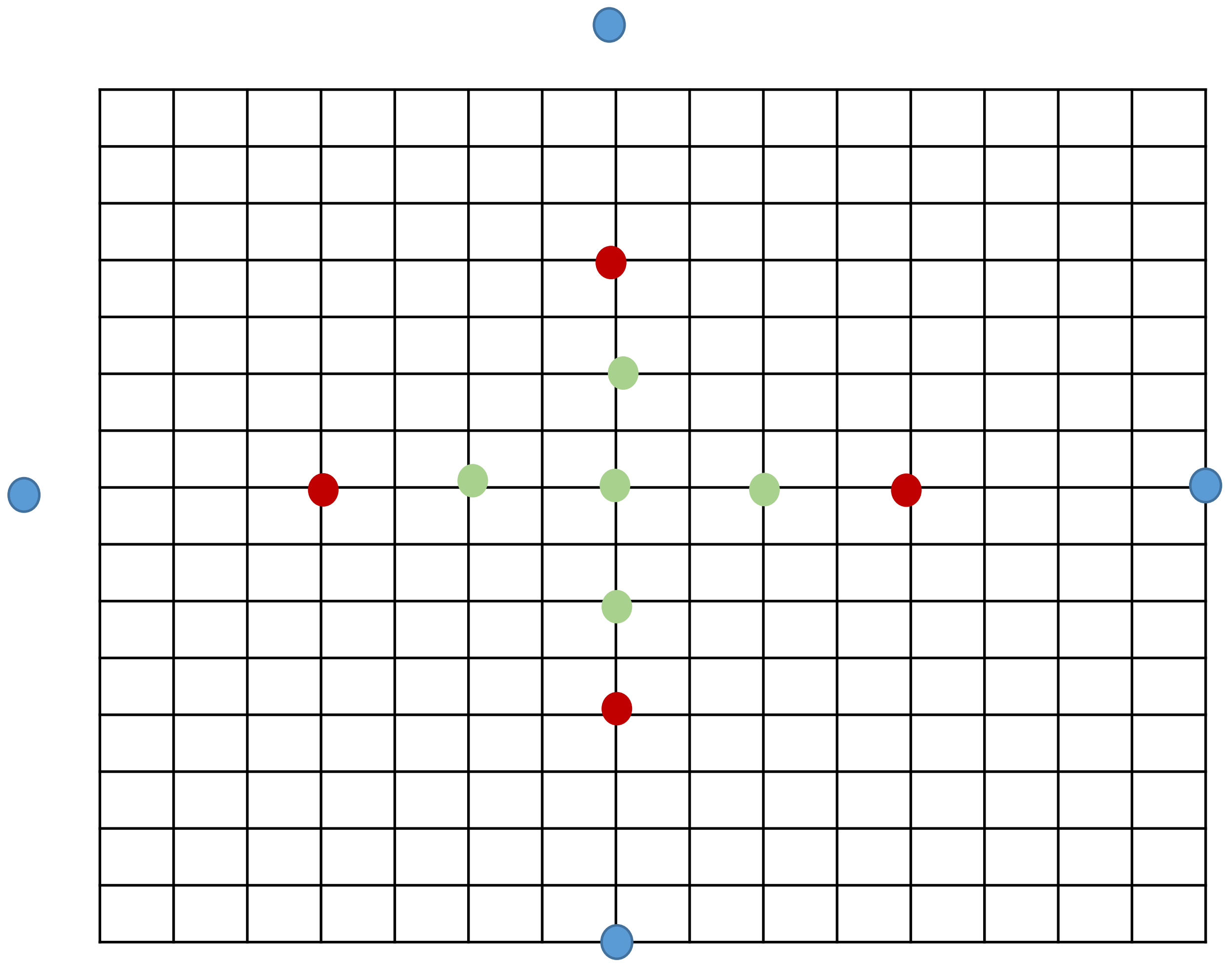
Two Dimensional Logarithmic Search

- If a point other than center is the best matching point,
  - Select this point as the new center
  - Repeat earlier steps
- If the best matching point is at the center
  - Start center as new point
  - set S = S/2



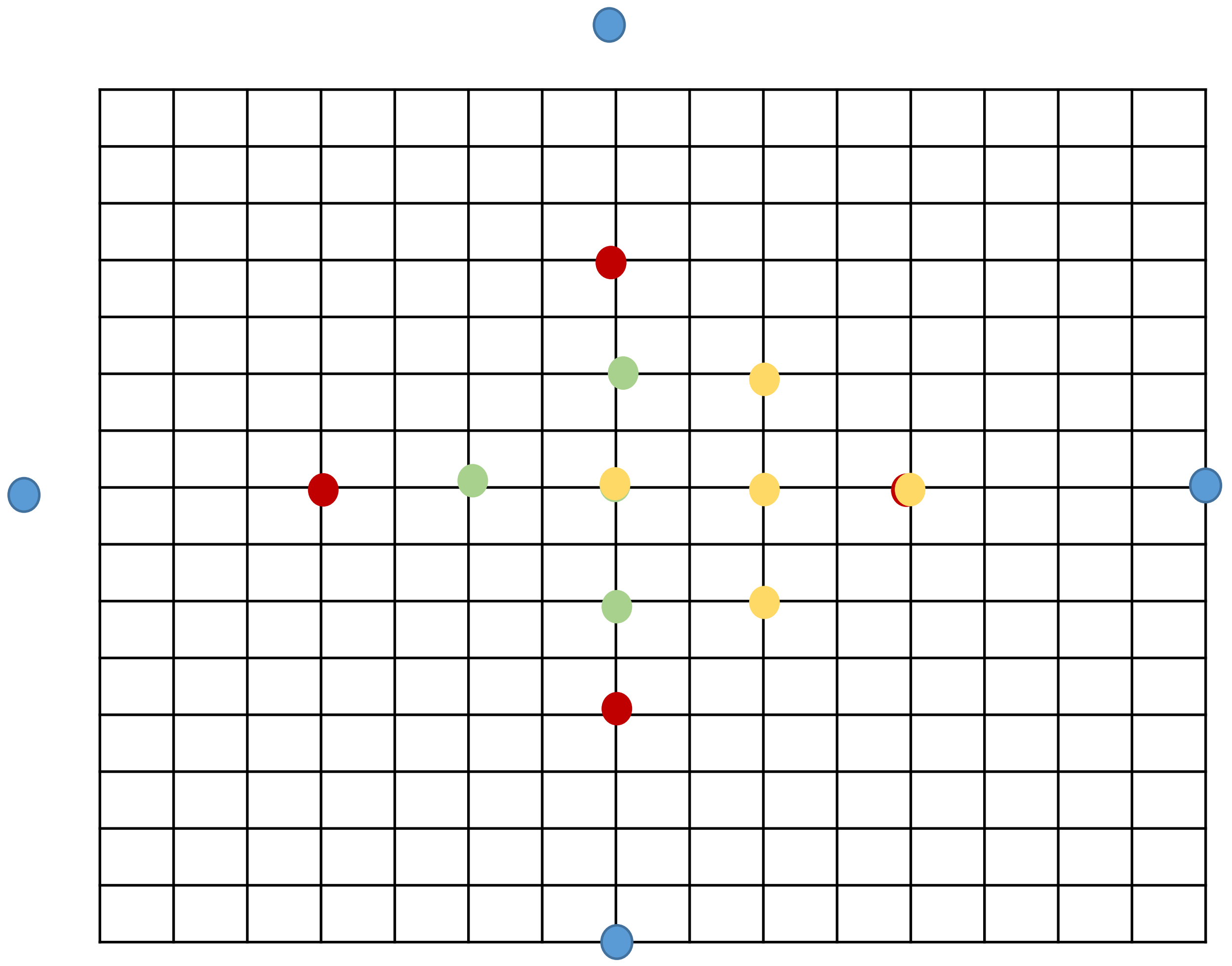
Two Dimensional Logarithmic Search

- If a point other than center is the best matching point,
  - Select this point as the new center
  - Repeat steps 2 to 3
- If the best matching point is at the center,
  - Start center as new point
  - set S = S/2



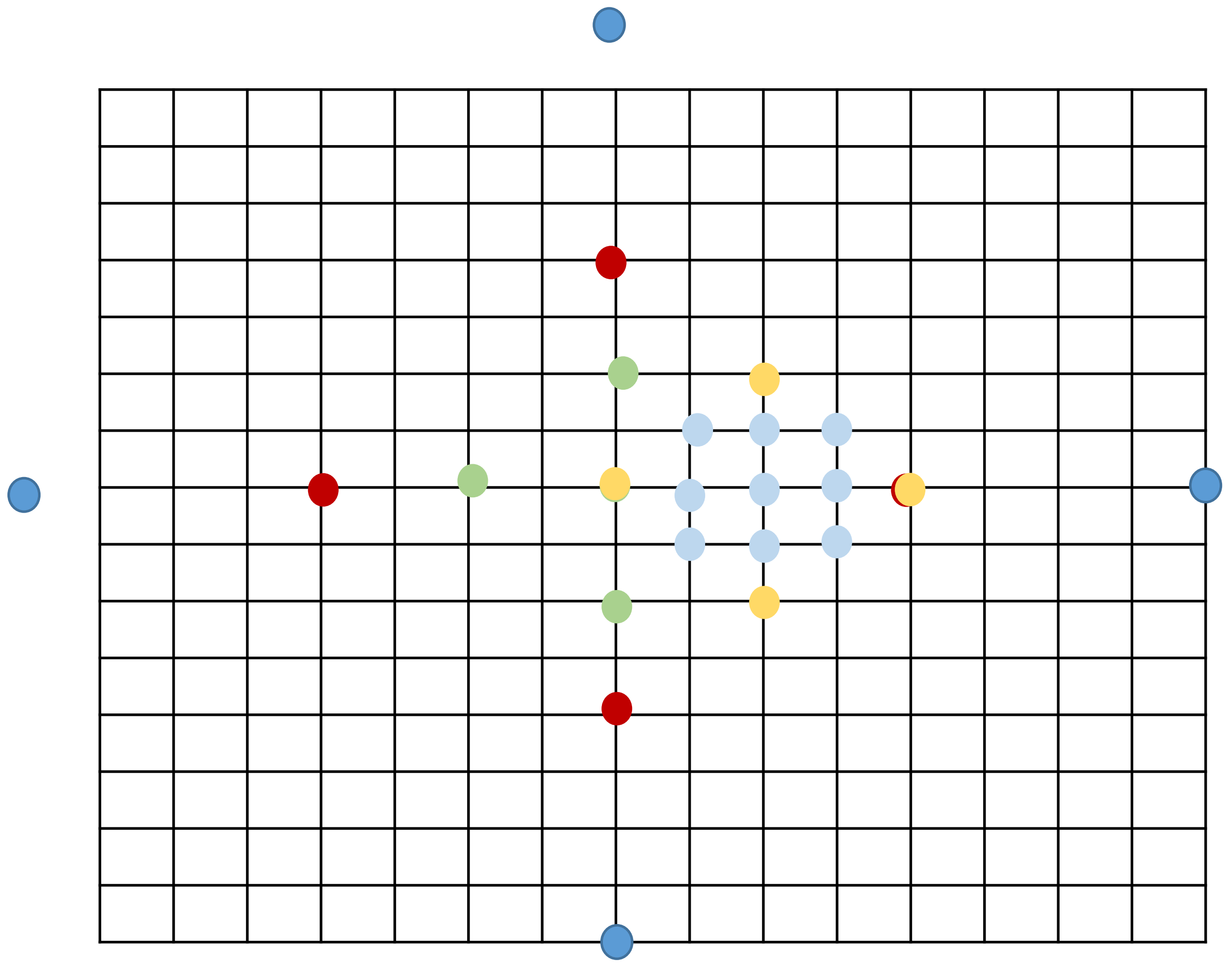
Two Dimensional Logarithmic Search

- If a point other than center is the best matching point,
  - Select this point as the new center
  - Repeat steps 2 to 3
- If the best matching point is at the center, set S = S/2
  - Start center as new point
  - set S = S/2



Two Dimensional Logarithmic Search

- If a point other than center is the best matching point,
  - Select this point as the new center
  - Repeat steps 2 to 3
- If the best matching point is at the center, set S = S/2
- If S = 1, all 8 locations around the center at a distance S are searched
- Set the motion vector as the point with least cost function



New Three Step Search (NTSS)

- TSS uses a uniformly allocated macro blocks and is prone to miss small motions
- NTSS is an improvement over TSS as it provides a center biased search scheme
- and has provisions to stop halfway to reduce the computational cost
- It is one of the first widely accepted fast algorithms
- Frequently used for implementing standards like MPEG1 and H.261

New Three Step Search (NTSS)

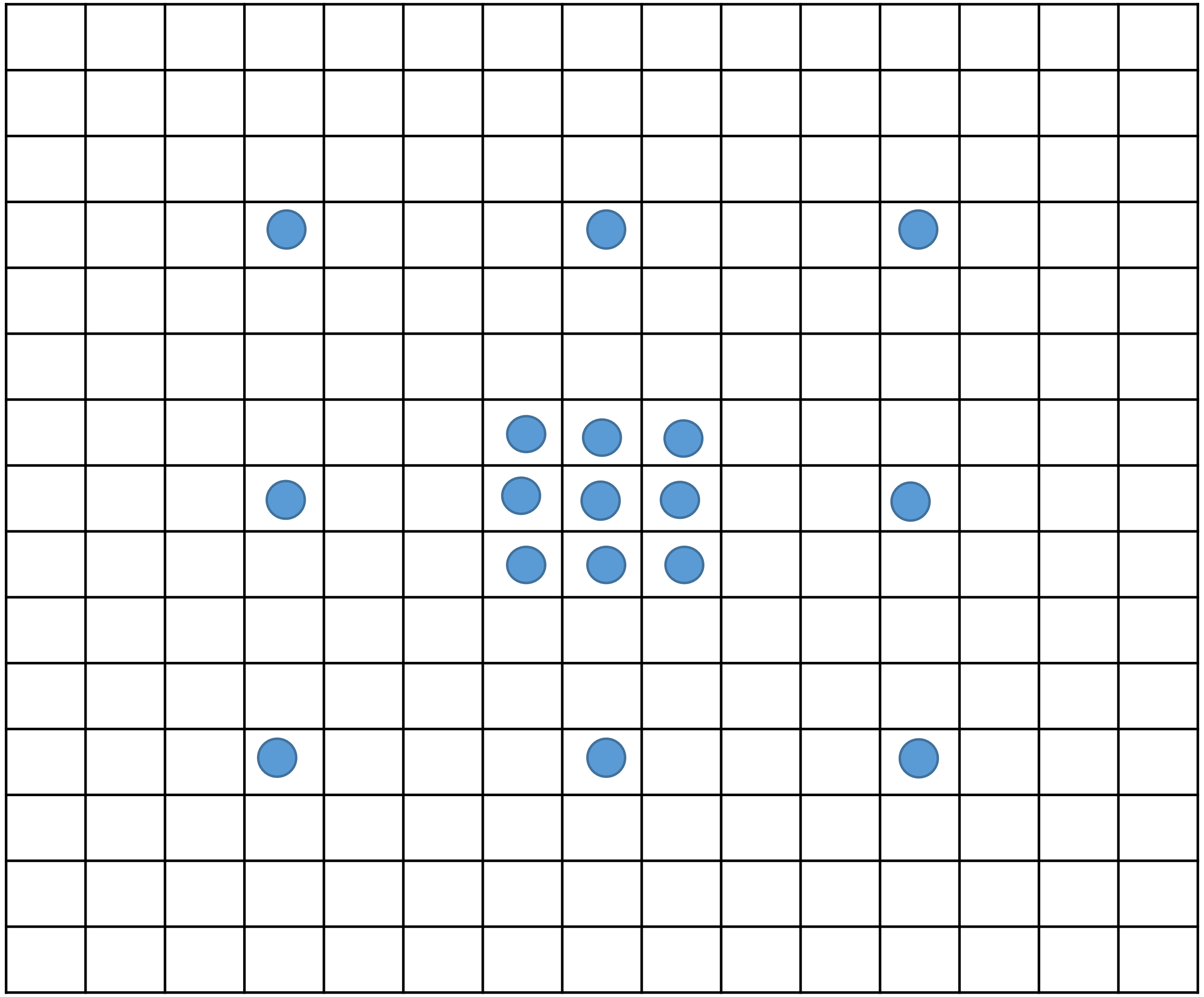
- NTSS algorithm utilizes
  - more macro blocks than that used by TSS
  - two half stop conditions to improve the performance of three step search algorithm
- Two windows (search area) are created
- In the first step, eight neighbors of the center are checked
- If the best match is found on small window,
- then additional three or five points are checked and algorithm stops

New Three Step Search (NTSS)

- Start with search location at center
- Search 8 locations +/- S pixels with S = 4

  and 8 locations +/- S pixels with S = 1 around location (0,0).
- Pick among the 16 locations searched, the one with minimum cost function

New Three Step Search (NTSS)

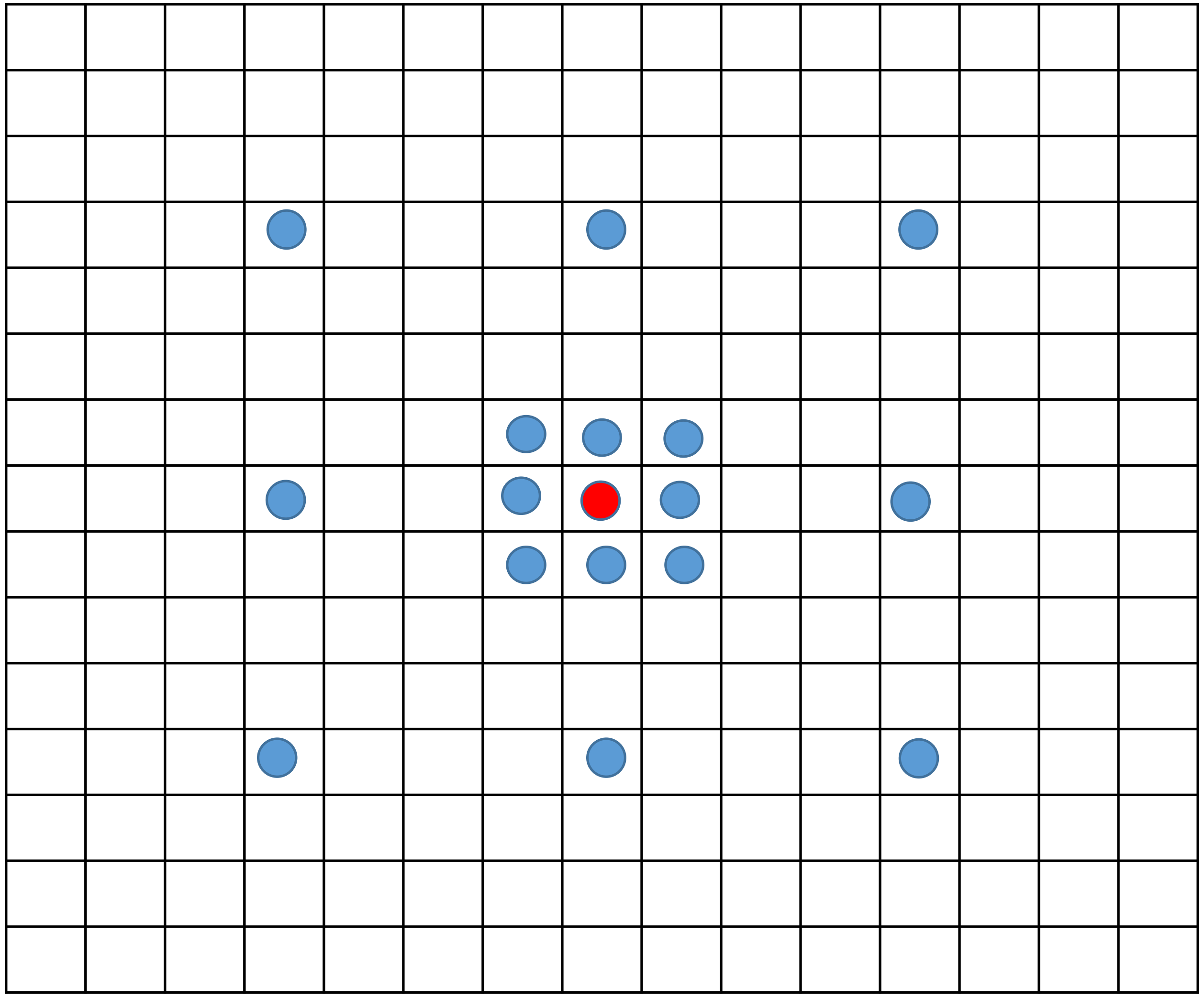


New Three Step Search (NTSS)

- Start with search location at center
- Search 8 locations +/- S pixels with S = 4

  and 8 locations +/- S pixels with S = 1 around location (0,0).
- Pick among the 16 locations searched, the one with minimum cost function
- If the minimum cost function occurs at origin, stop the search and set motion vector to (0,0)

New Three Step Search (NTSS)



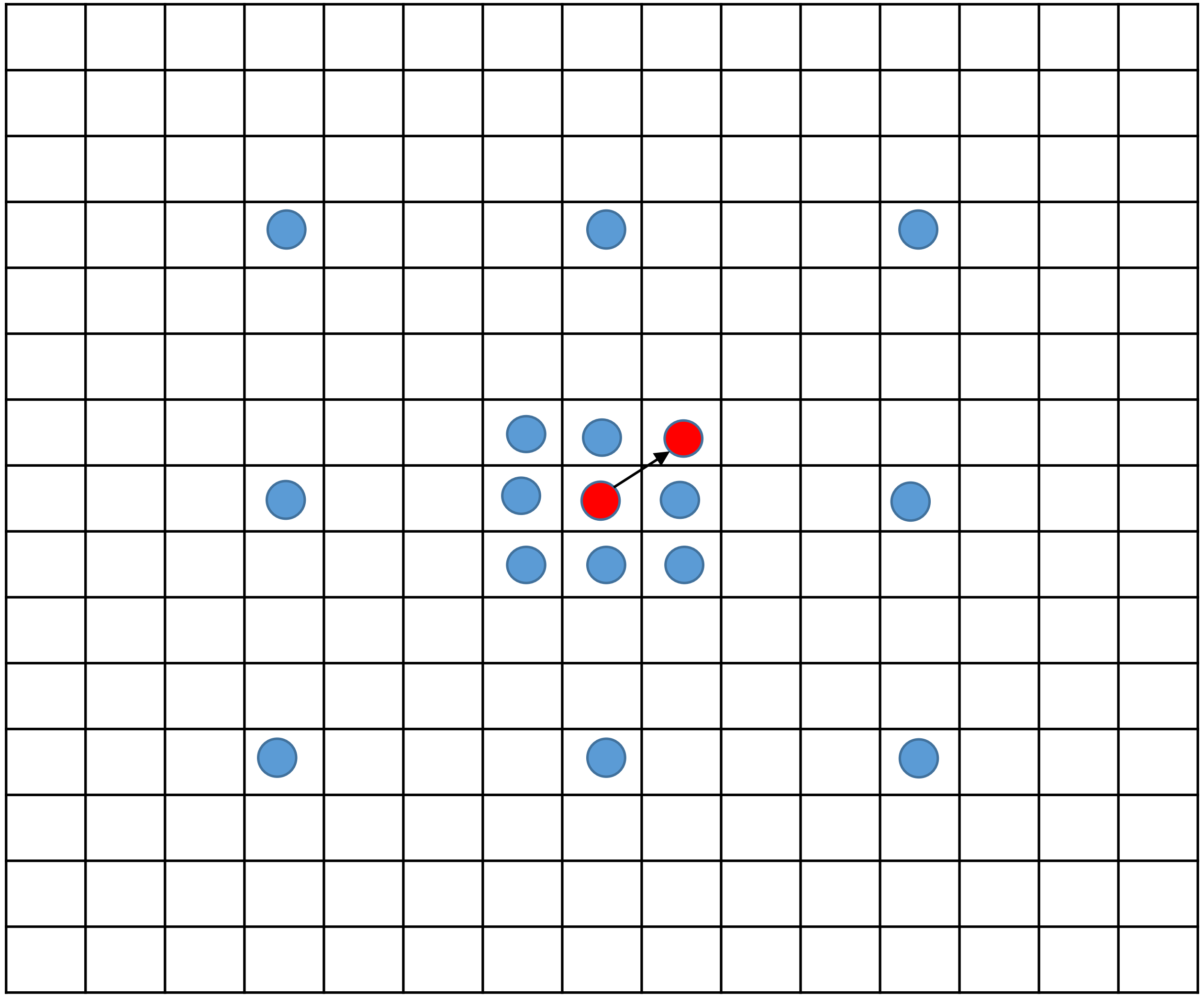
Stop search as minimum cost location is found at the center

New Three Step Search (NTSS)

- Start with search location at center
- Search 8 locations +/- S pixels with S = 4

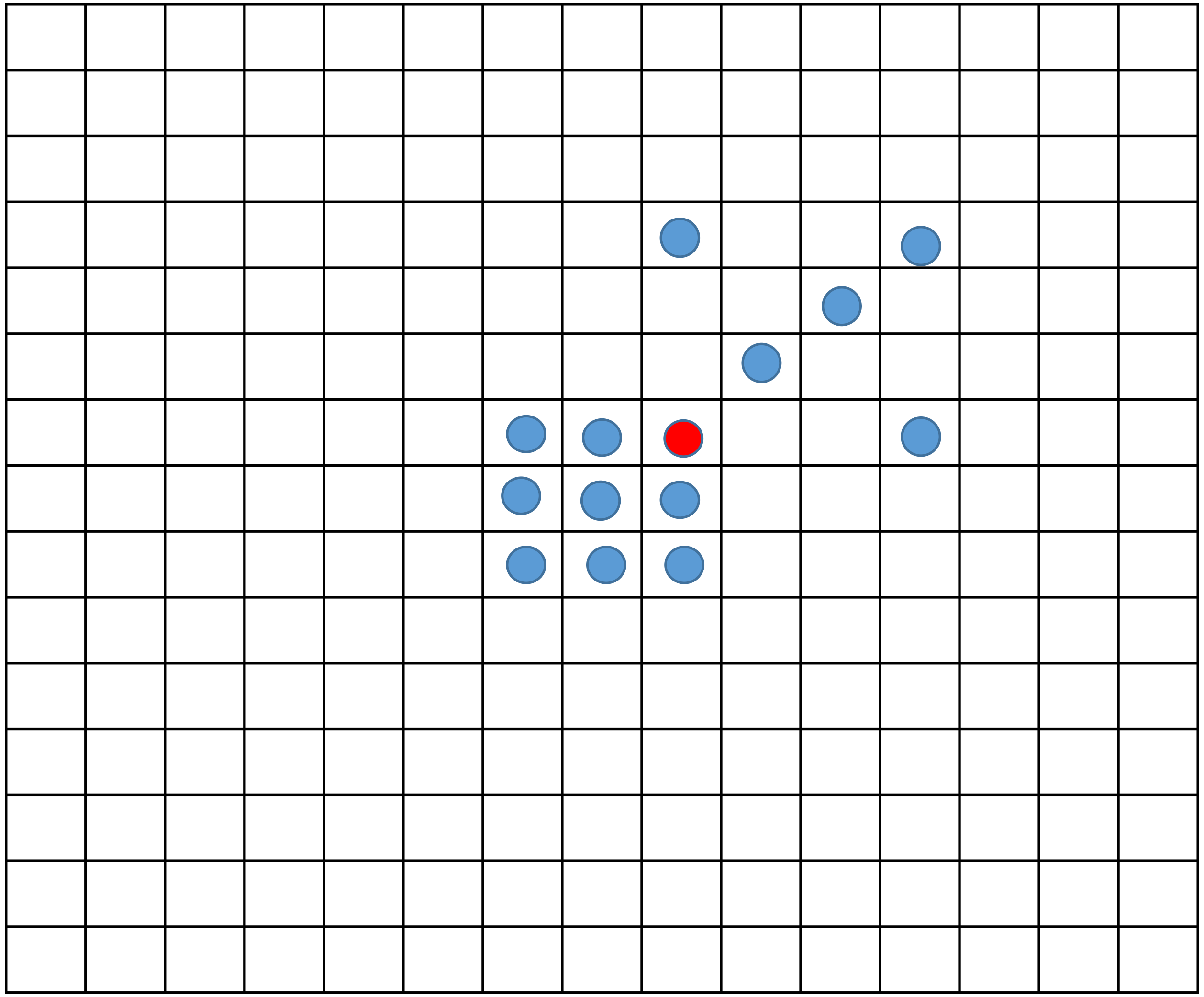
  and 8 locations +/- S pixels with S = 1 around location (0,0).
- Pick among the 16 locations searched, the one with minimum cost function
- If the minimum cost function occurs at origin, stop the search and set motion vector to (0,0)
- If the minimum cost function occurs at one of the 8 locations for S = 1, set the new search origin at this location
    - Check matching blocks for this location, depending on location it may check either 3 or 5 points
- The one that gives lowest SAD is the closest match, is the matching block

New Three Step Search (NTSS)



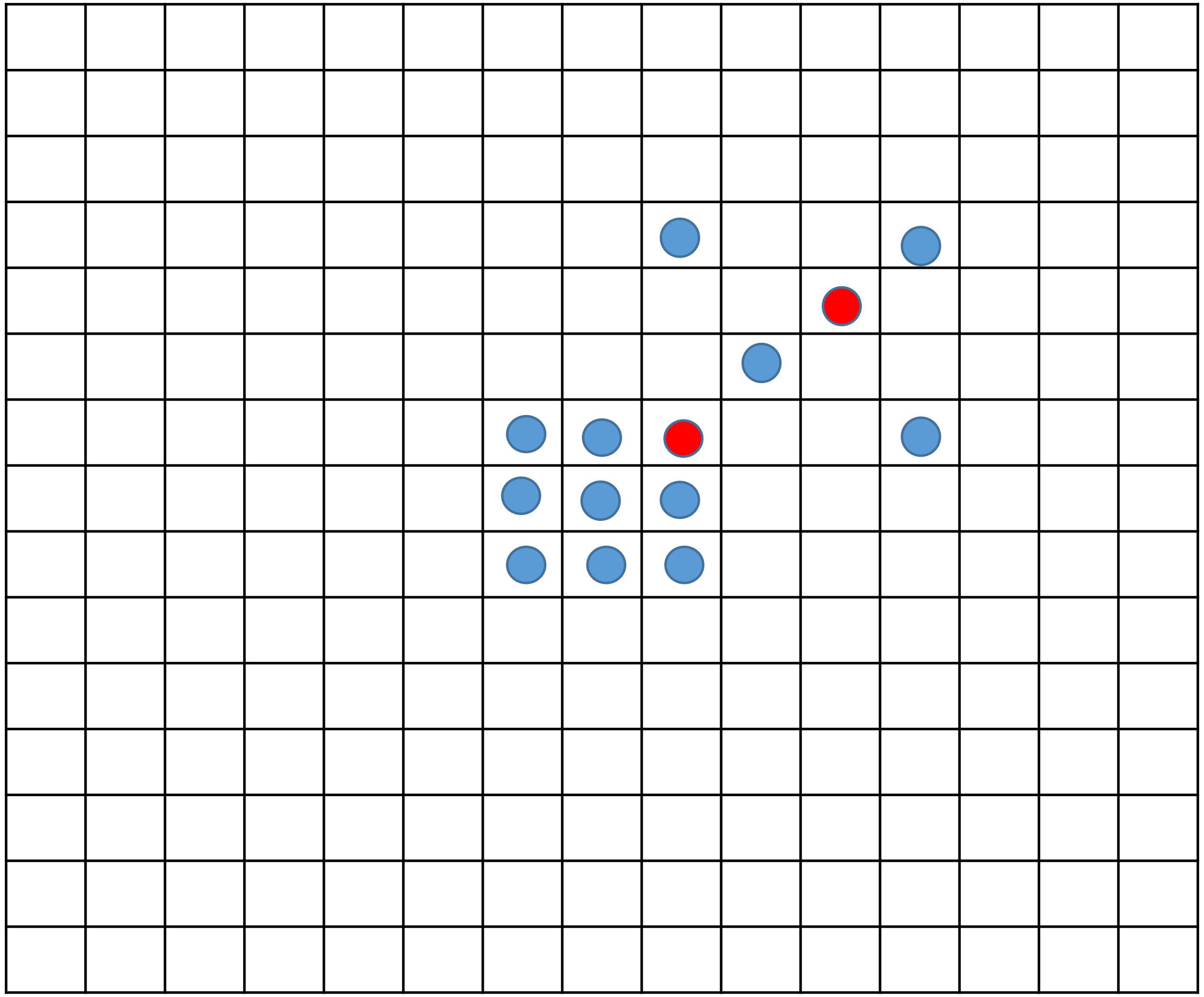
minimum cost function (SAD) occurs at one of the 8 locations at S = 1

New Three Step Search (NTSS)



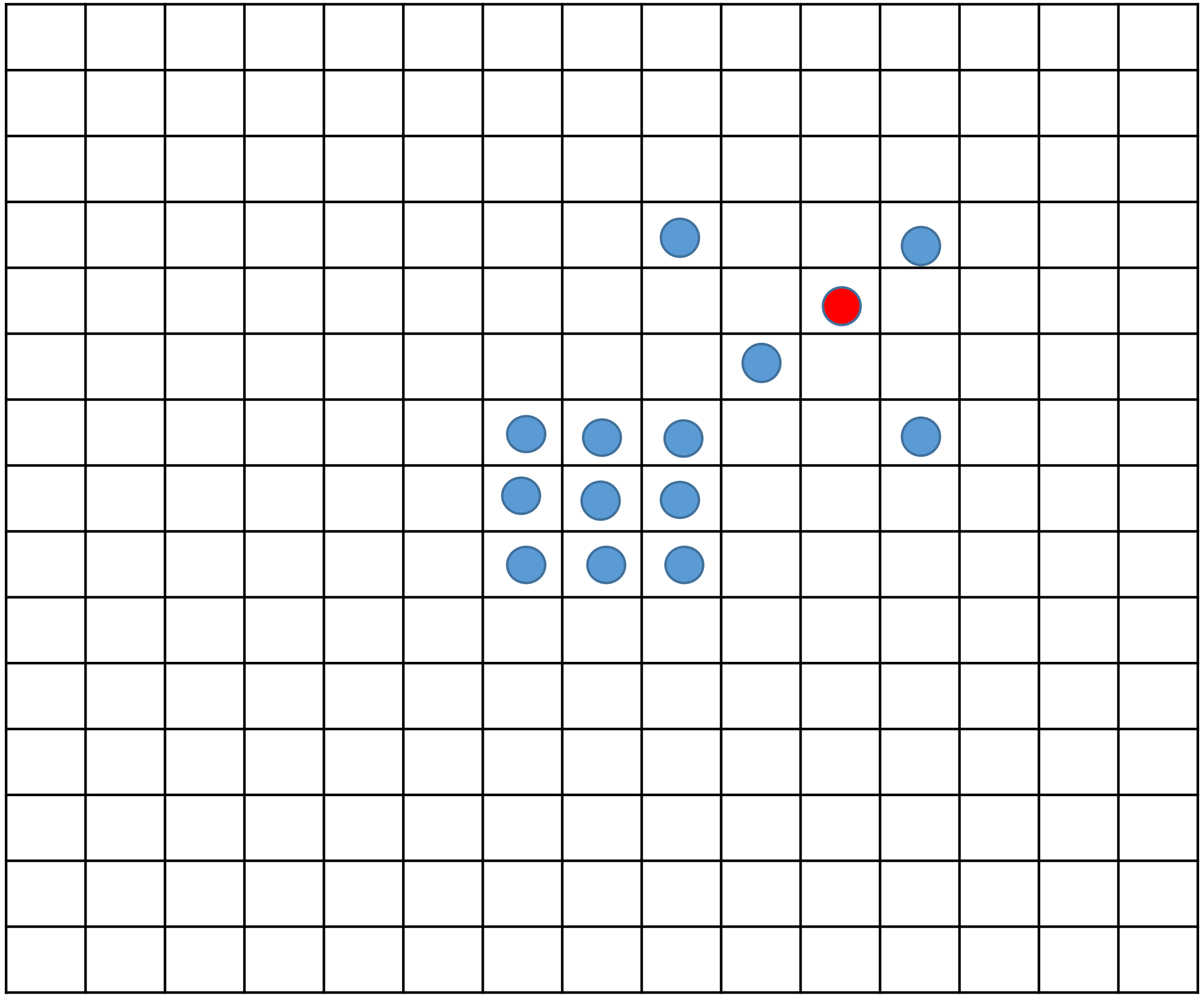
Check SAD for this location, depending on location it may check either 3 or 5 points

New Three Step Search (NTSS)



Check SAD for this location, depending on location it may check either 3 or 5 points

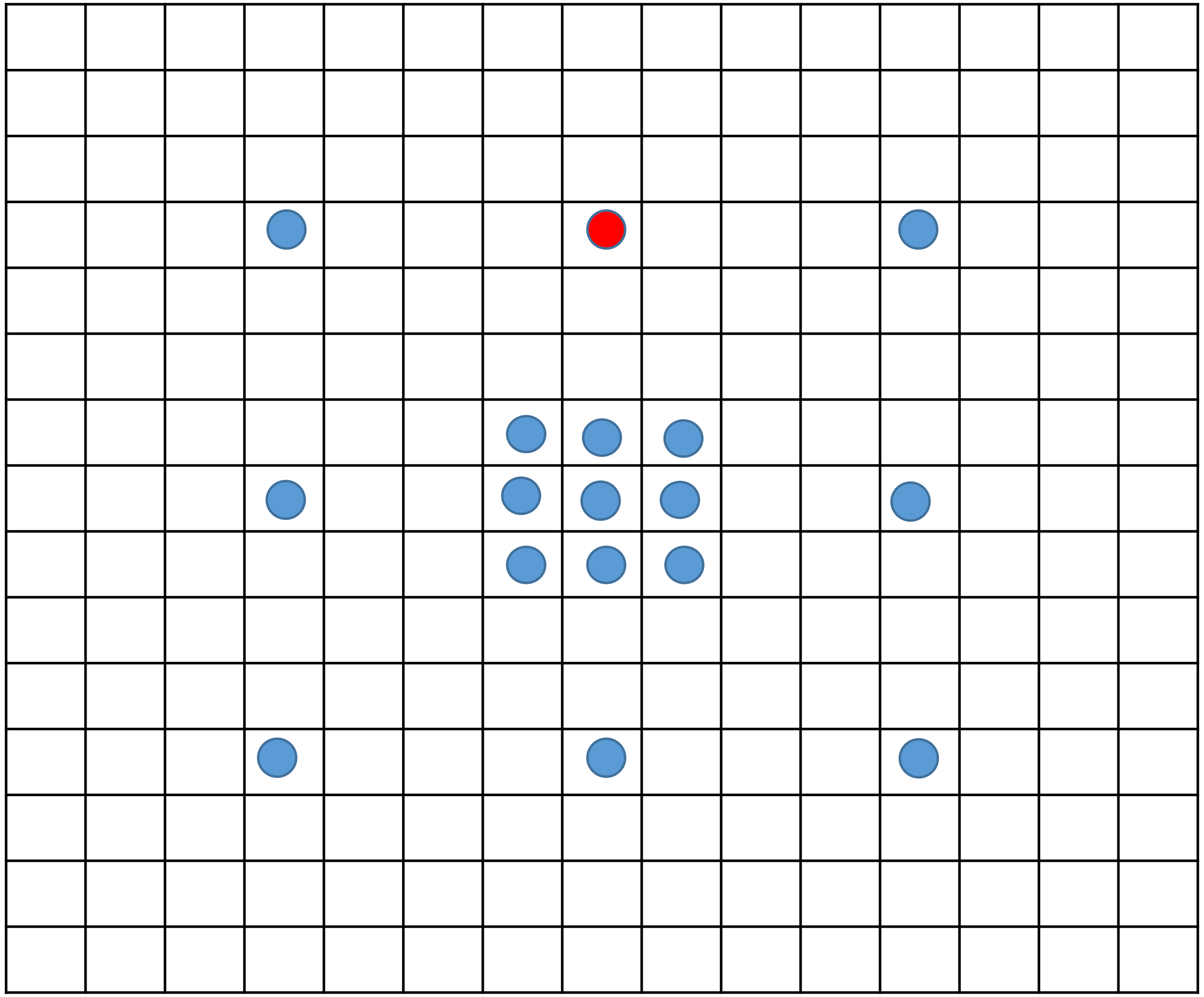
New Three Step Search (NTSS)



This is the best matching block

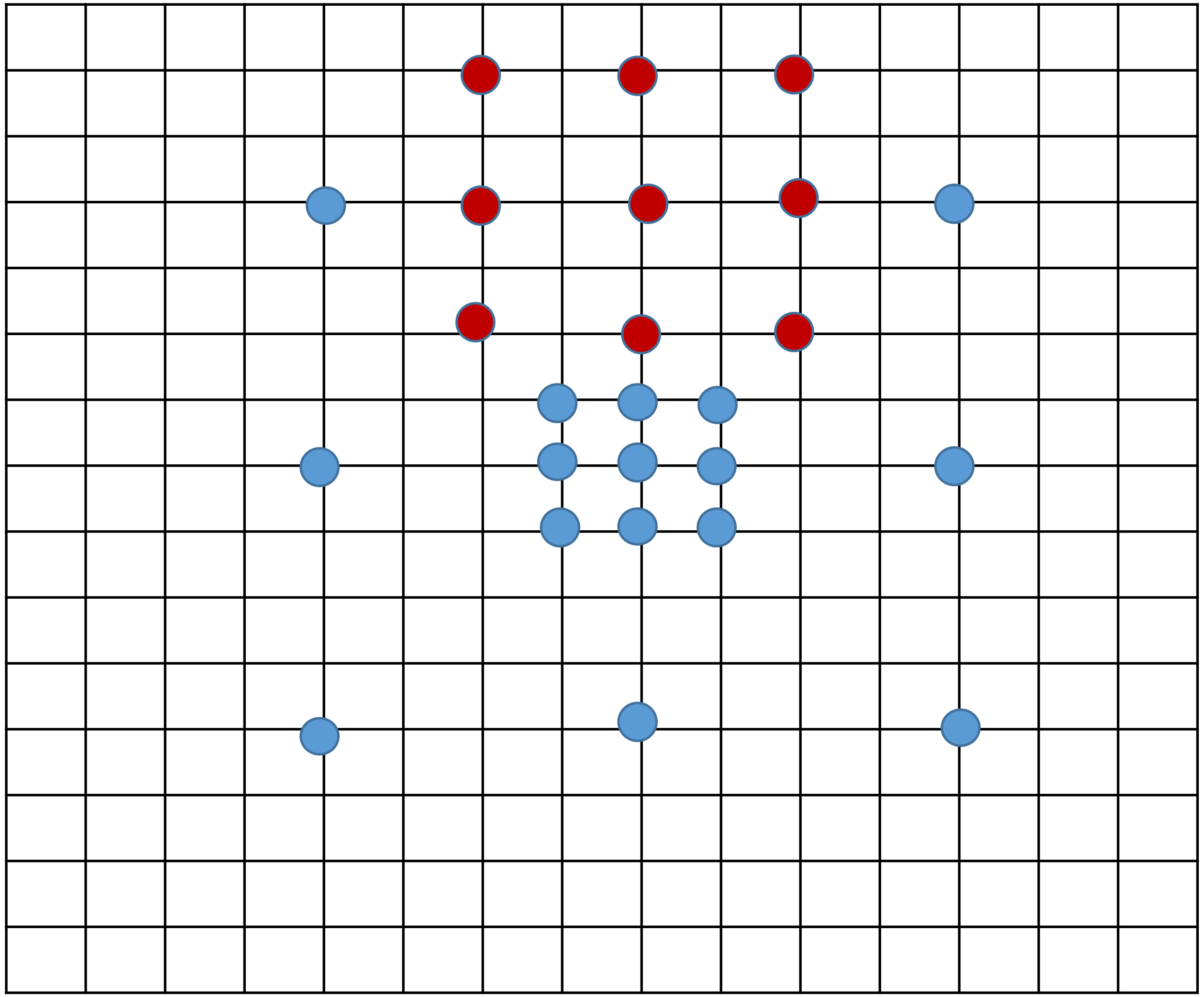
New Three Step Search (NTSS)

If the lowest SAD after the first step is one of the 8 locations at S = 4



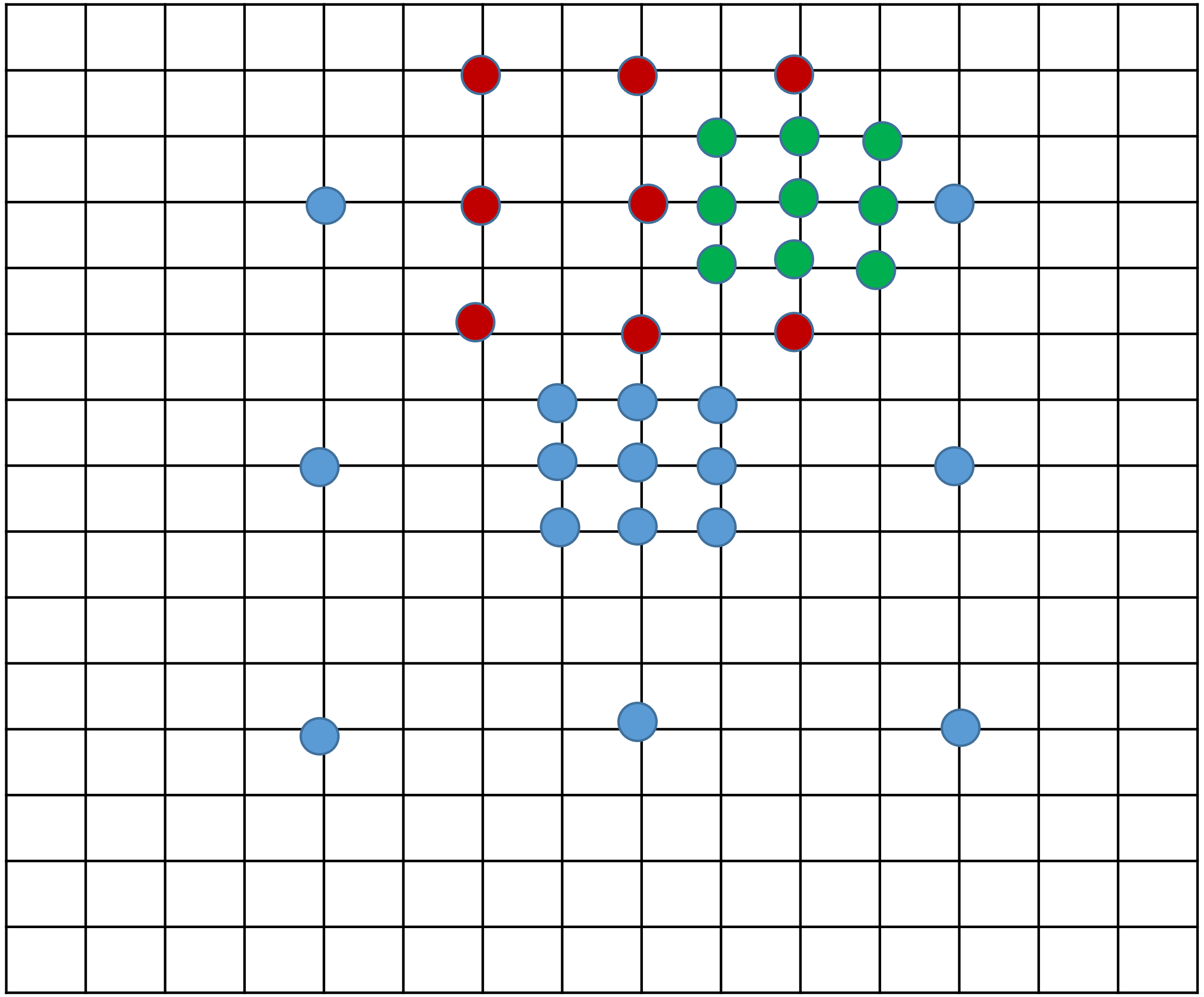
New Three Step Search (NTSS)

- If the lowest weight after the first step is one of the 8 locations at S = 4,
  - Pick among the 8 locations searched, the one with minimum cost function
  - Set the new search origin to the above picked location
  - Set the new step size as S = S/2



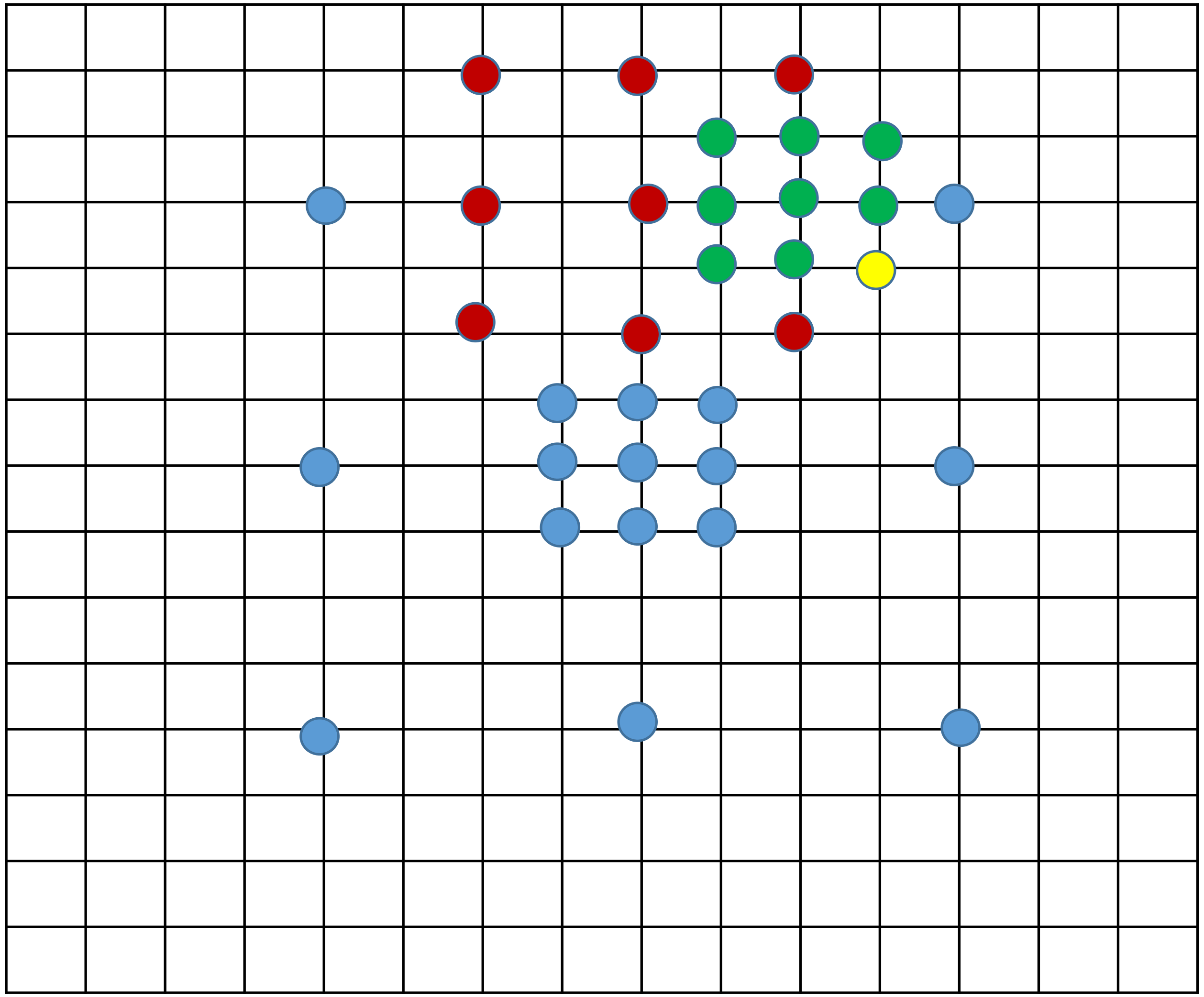
New Three Step Search (NTSS)

- If the lowest weight after the first step is one of the 8 locations at S = 4,
  - Pick among the 8 locations searched, the one with minimum cost function
  - Set the new search origin to the above picked location
  - Set the new step size as S = S/2
  - If matching block is not in the center
  - Repeat the search procedure until S = 1



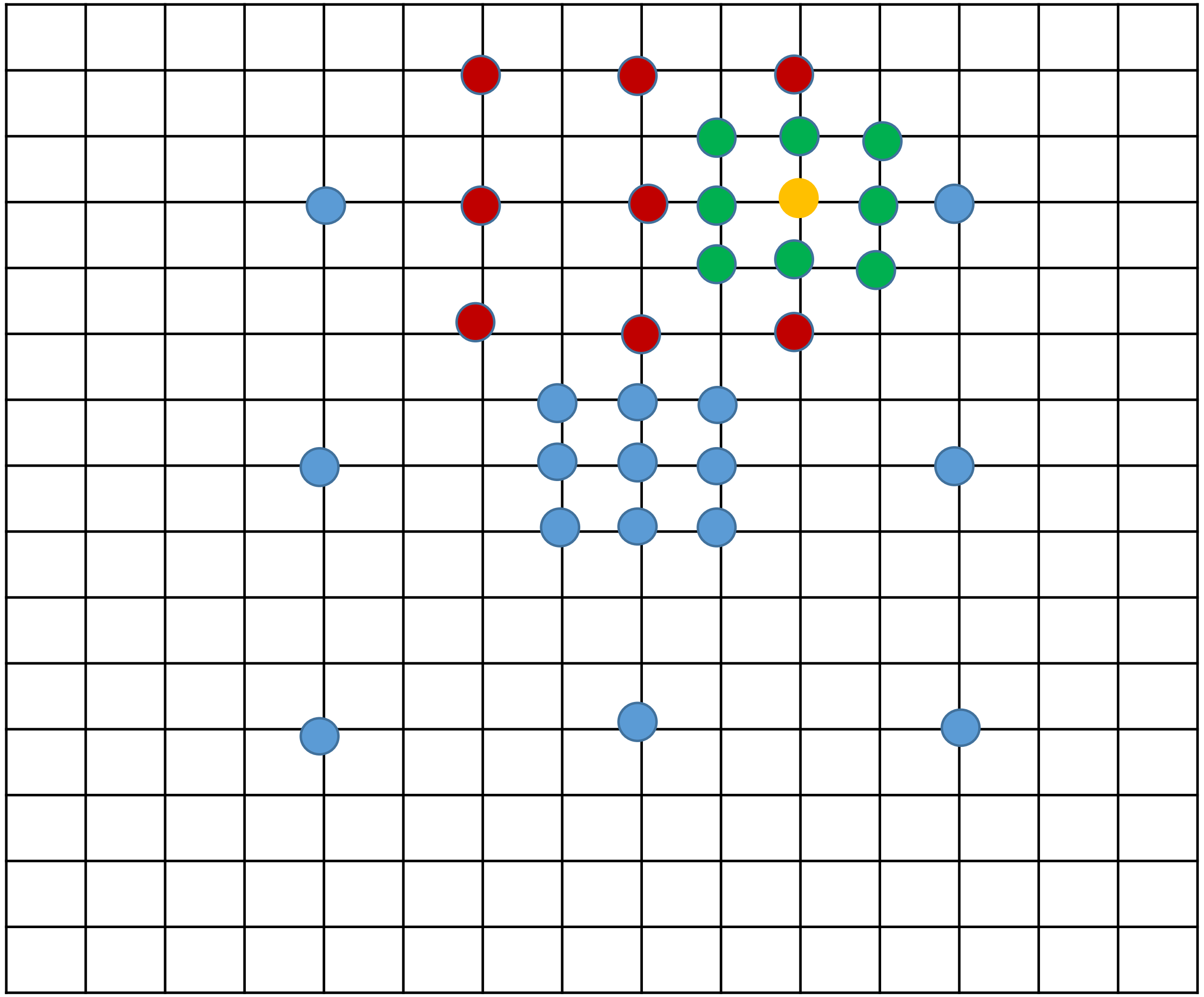
New Three Step Search (NTSS)

- If the lowest weight after the first step is one of the 8 locations at S = 4,
  - Pick among the 8 locations searched, the one with minimum cost function
  - Set the new search origin to the above picked location
  - Set the new step size as S = S/2
  - If matching block is not in the center
  - Repeat the search procedure until S = 1



New Three Step Search (NTSS)

- If the lowest weight after the first step is one of the 8 locations at S = 4,
  - Pick among the 8 locations searched, the one with minimum cost function
  - Set the new search origin to the above picked location
  - Set the new step size as S = S/2
  - If matching block is in the center
  - Choose matching block and stop



Techniques for Motion Estimation

- Pixel Difference
- Fixed Block matching
- Hierarchical Block Matching
- I, P and B Frames

Block matching for Video Signal

- Search algorithms are used to find a matching block between previous frame and current frame
- Motion vector is calculated for each matching block
- Motion vectors for each frame is calculated
- Thus frames of a video is represented by previous frame and corresponding motion vector of current frame
- Using motion vectors current frame can be reconstructed

Block Matching and Random Access

- For H.261 standard, each frame is coded using prediction from the previous frame
- Therefore, to decode a particular frame in the sequence, decoding starts from the first frame
- Some frames are coded without any reference to past frames
- These frames are referred to as Independent (**I frames**)

I Frames

- **I frames** provide random access to frames for display
- **I frames** should occur quite frequently for quick access to the required frame
- **I frames** do not use temporal correlation (matching blocks)
- Therefore compression rate of **I frame** is much lower than other frames
- Increasing the number of **I frames** reduces compression
- Trade off is between compression efficiency and access capability

I, P and B frames

- To improve compression efficiency, compression algorithm contains
  - Predictive coded, **P frames**
  - Bidirectionally predictive coded **B frames**
- **P frames** are coded using motion estimation from the last **I** or **P frame** whichever is closest to current frame
- Compression efficiency of **P frames** is substantially higher than **I frames**
- **I** and **P frames** are called anchor frames

B frames

- Compensates reduction in the amount of compression due to the frequent use of **I frames**
- Uses motion estimation from the most recent anchor frame and the closest future anchor frame
- For a video sequence
  - there is a sudden change between one frame and the next
  - this is a common occurrence in TV advertisements
  - Therefore prediction based on the past frames may not be useful
  - predictions based on future frames may have high probability of being accurate

B frames

- Can only be generated after the future anchor frame has been generated
- Other frames do not use it for prediction
- Therefore, **B frames** can tolerate more error
- because this error is not propagated by the prediction process
- Provides high level of compression

Group of Pictures (GOP)

- Frames of video sequence are identified as **I, P** and **B** frames
- Different frames are organized together in a GOP
- GOP is the smallest random access unit in the video sequence
- Contains at least one **I frame**
- First **I frame** in a GOP is
  - either the first frame of the GOP
  - or is preceded by **B frames**
    that use motion compensated prediction only from this **I frame**

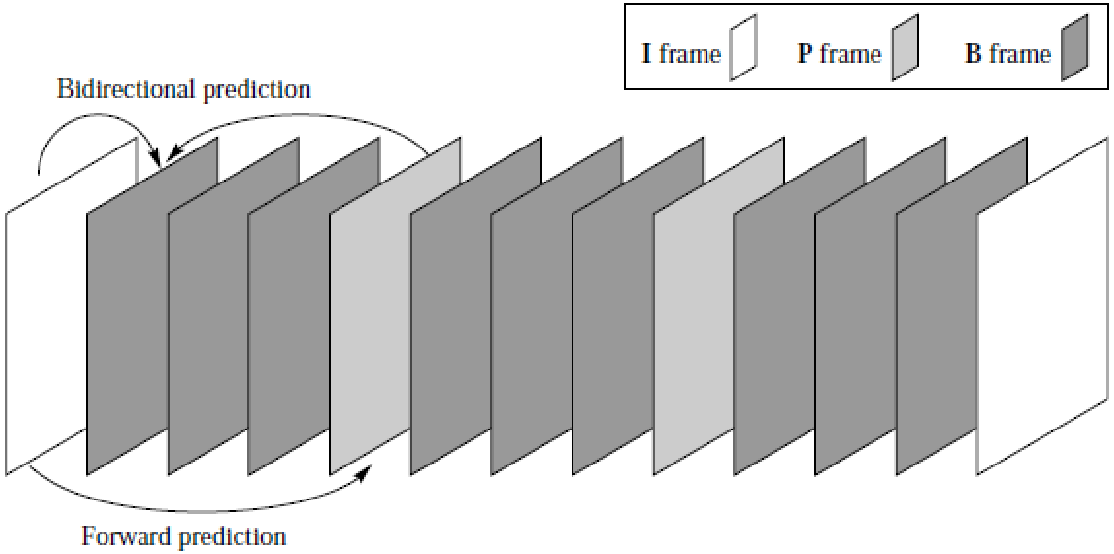
I frame	….	…	…

Or

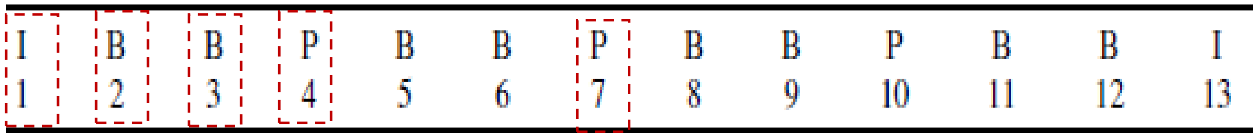
B frame	**I frame**	…	…

Possible arrangement for a GOP

- **B frame** relies on future anchor frames
- Therefore there are two sequence orders
- Display order - sequence in which the video sequence is displayed to the user
- Bit stream order - sequence in which the video sequence is compressed/ decompressed



sequence of frames in display order



Display Order

- 1st frame, **I frame:** Compress without reference to any previous frame
- 2nd frame, **B frame:** Dependent on 1st and 4th frame therefore compress 4th frame first
- 4th frame **P frame**: Compress using 1st frame
- 2nd frame, **B frame:** Compress using 1st and 4th frame
- 3rd frame, **B frame:** Compress using 1st and 4th frames
- 5th frame, **B frame:** Dependent on 4th and 7th frame therefore compress 7th frame first
- 7th frame, **P frame**: Compress using prediction from 4th frame

sequence of frames in display order

I	B	B	P	B	B	P	B	B	P	B	B	I
1	2	3	4	5	6	7	8	9	10	11	12	13

Display Order

- 5th frame, **B frame:** Compress using frame 4 and frame 7
- 6th frame, **B frame:** Compress using 4th and 7th frames
- Thus, processing order is different from the display order

display order and bitstream order

I	B	B	P	B	B	P	B	B	P	B	B	I
1	2	3	4	5	6	7	8	9	10	11	12	13

Display order

I	P	B	B	P	B	B	P	B	B	I	B	B
1	4	2	3	7	5	6	10	8	9	13	11	12

Bitstream order