

“Optimizing video compression using high performance algorithms and techniques”

By Vikram Kadam

PEDAL Lab, School of Systems Engineering
University of Reading

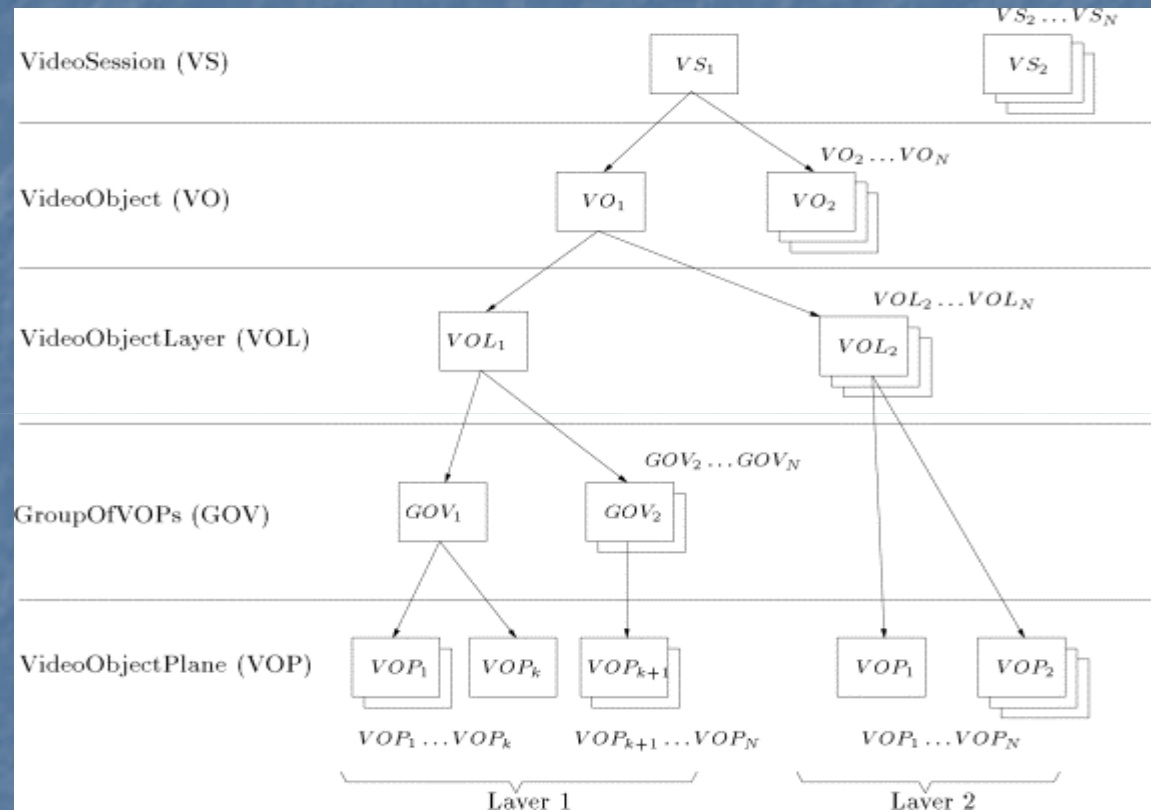
Motivation

- Technology
- Digital world
- Information management
(quality v/s quantity)

Video: "I see"

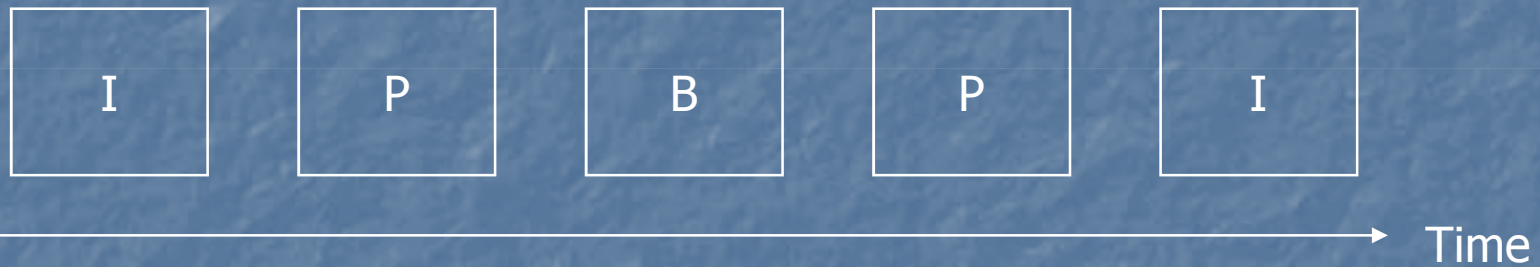
- Analog
 - Luminance
 - Chrominance
- Digital
 - Spatial Sampling
 - Temporal Sampling

MPEG-4 Visual Structure



Video Object Plan (VOP)

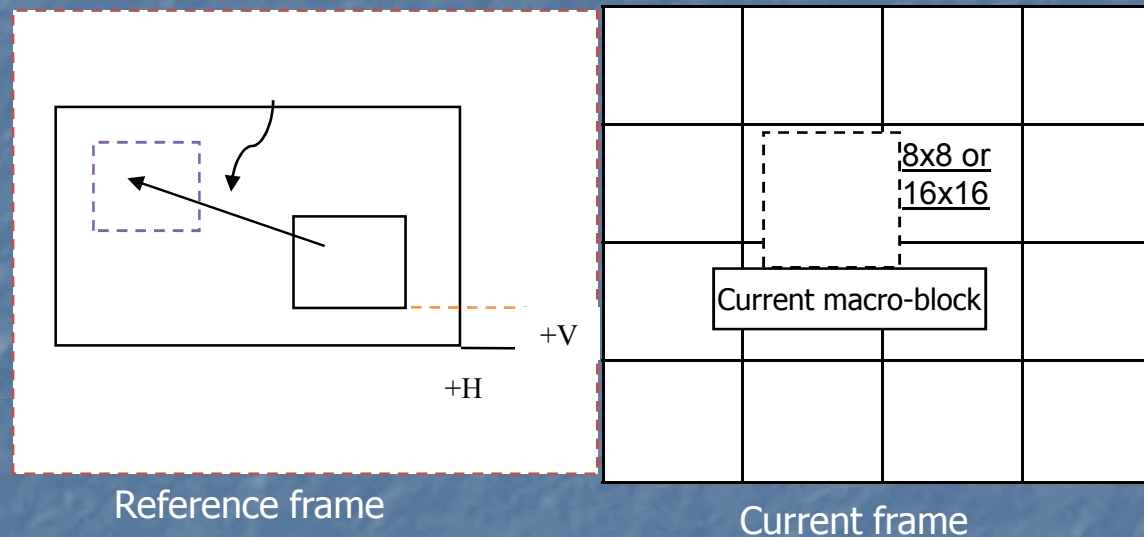
- IVOP - Intra VOP
- PVOP - Predicted VOP
- BVOP - Bidirectional VOP



OR

I I P P P B P P P I

Block Matching



- Block Matching Algorithms in Motion Estimation
 - Full Search(FS)
 - Three Step Search (TSS)
 - Two Dimensional Logarithmic Search (TDL)
 - Binary Search (BS)
 - Four Step Search (FSS)
 - Orthogonal Search Algorithm (OSA)
 - One at a Time Algorithm (OTA)
 - Cross Search Algorithm (CSA)
 - Hierarchical Search Algorithm (HSA)
 - Diamond Search (DS)
 - New diamond search
 - Cross diamond search

Energy Measures

- Consider sample block size $N \times N$, current block C_{ij} and reference block R_{ij} reference frame block.
- *Mean Squared Error (MSE)*: this provides a measure of the energy remaining in the difference block.

$$\text{MSE} = \frac{1}{N^2} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} (C_{ij} - R_{ij})^2$$

- Mean Absolute Error (MAE also called MAD-mean absolute difference): provides a reasonably good approximation of residual energy and easier to calculate than MSE. As it needs magnitude calculations than square calculation for each sample.

$$\text{MAE} = \frac{1}{N^2} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} |C_{ij} - R_{ij}|$$

- The comparison is more simplified by neglecting $1/N^2$. called as SAE(sum of absolute error) or SAD (sum of absolute difference)

$$\text{SAE} = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} |C_{ij} - R_{ij}|$$

SAE gives a reasonable approximation to block energy so eq. 3.3 is commonly used for matching criteria.

Basic MAD Operation



Previous frame



Current frame



Residual frame

Block Matching Example

Current block	Reference block	Position (x,y)																																											
<table border="1"> <tr><td>1</td><td>3</td><td>2</td></tr> <tr><td>6</td><td>4</td><td>3</td></tr> <tr><td>5</td><td>4</td><td>3</td></tr> </table>	1	3	2	6	4	3	5	4	3	<table border="1"> <tr><td>1</td><td>3</td><td>2</td><td>4</td><td>5</td></tr> <tr><td>6</td><td><u>4</u></td><td><u>2</u></td><td><u>3</u></td><td>2</td></tr> <tr><td>5</td><td><u>4</u></td><td><u>2</u></td><td><u>2</u></td><td>3</td></tr> <tr><td>4</td><td><u>4</u></td><td><u>3</u></td><td><u>3</u></td><td>1</td></tr> <tr><td>4</td><td>6</td><td>7</td><td>4</td><td>5</td></tr> </table>	1	3	2	4	5	6	<u>4</u>	<u>2</u>	<u>3</u>	2	5	<u>4</u>	<u>2</u>	<u>2</u>	3	4	<u>4</u>	<u>3</u>	<u>3</u>	1	4	6	7	4	5	<table> <tr><td>(-1, 1)</td><td>(0, 1)</td><td>(1, 1)</td></tr> <tr><td>(-1, 0)</td><td>(0, 0)</td><td>(1, 0)</td></tr> <tr><td>(-1, -1)</td><td>(0, -1)</td><td>(1, -1)</td></tr> </table>	(-1, 1)	(0, 1)	(1, 1)	(-1, 0)	(0, 0)	(1, 0)	(-1, -1)	(0, -1)	(1, -1)
1	3	2																																											
6	4	3																																											
5	4	3																																											
1	3	2	4	5																																									
6	<u>4</u>	<u>2</u>	<u>3</u>	2																																									
5	<u>4</u>	<u>2</u>	<u>2</u>	3																																									
4	<u>4</u>	<u>3</u>	<u>3</u>	1																																									
4	6	7	4	5																																									
(-1, 1)	(0, 1)	(1, 1)																																											
(-1, 0)	(0, 0)	(1, 0)																																											
(-1, -1)	(0, -1)	(1, -1)																																											

The current block is compared with different (x, y) positions in the reference frame. The mean square error (MSE) is calculated between current block and for each position (x, y). eg. Position (0, 0)

$$\text{MSE} = \{ (1-4)^2 + (3-2)^2 + (2-3)^2 + (6-4)^2 + (4-2)^2 + (3-2)^2 + (5-4)^2 + (4-3)^2 + (3-3)^2 \} / 9 = 2.44$$

Similarly, MSE is calculated for all other positions, and minimum MSE of relative position is the best match.

Full Search Strategy

Mean Absolute Difference (MAD)

MAD:=0

for I:=1 to N do

for J:= 1 to N do

MAD= MAD + abs(A[I,J] - B[I,J]);

Block size: N x N

The complexity of single MAD operation is $O(N^2)$ In fact above, procedure have $2N^2$ addition/multiply operations.

Full Search strategy (2)

Suppose that (p, q) is the starting pixel of a block and 'w' is search window size then the code is:

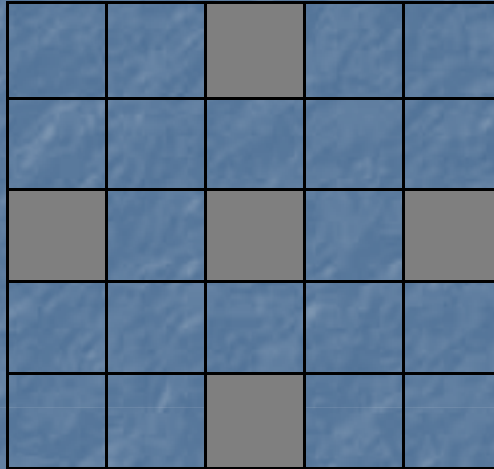
```
D = 0; X = 0, Y = 0;
for r := p-w to (p+N-1)+w do
  for s := q-w to (q+N-1)+w do
  {
    E = MAD (block(p, q), block(r, s))

    If E < D then
    { D = E; X = p; Y = q; }
  };
```

In this procedure D defines minimum MAD in window and (x, y) the position block on the I-frame which is most similar to the block at (p, q) on the p-frame. The difference $(p-x, q-y)$ is the **motion vector**. Since the **window size is $(2w+N-1)$** then the cost is $((2w+N-1)^2) * (2N^2)$

Assuming that we do not search for blocks that do not fit entirely in the window the actual search can be reduced by N-1 steps in each dimension. That is $2(2w^2)N^2 = 8w^2N^2$ operations.

Logarithmic Search strategy



The number of MAD operations in first step is 5 and the 4 MAD 's for each subsequent . The formula for the logarithmic fast search method will be,

$$5 + 4\log_2 2w$$

The complexity of the logarithmic search of a window is $(5 + 4\log_2 2w) * 2N^2$ add/subtract operations.

Consider the image has R x S pixels

$$\text{Exhaustive: } ((8w^2 N^2) * (R * S)) / (N * N) = (8w^2) * (R * S)$$

$$\text{Logarithmic: } ((5 + 4\log_2 2w) * 2N^2) * (R * S) / (N * N)$$

$$\text{as we simplified this equation, } 2 * (5 + 4\log_2 2w) * (R * S)$$

Fast Motion Compensation: New Strategy

The motion vector search is accelerated if

- The cost of comparing the description is less than the cost of computing the MAD between the two blocks directly.
- We can reuse the descriptions computed on the I-frame, where as the MAD approach may use the same block on the I-frame several times.
- The cost of pre-computation plus the description search is less than the cost of a direct MAD Approach.

There is more scope for parallelism in the implementation of this method because,

- The pre-processing and search can be overlapped.
- The search phase may offer advantages over parallelism in the basic block matching technique.

New Search Algorithm (1)

Consider X and Y are two macro blocks, the function $f()$ computes the description of block. If after computation of $f(X)$ and $f(Y)$, $f(X) \neq f(Y)$ then there is more possibility of $X \neq Y$ and that $f(X) \neq f(Y)$ are distinct when X and Y are not similar.

Despite the shortcoming of this particular $f()$ it can be used to demonstrate the key points of the approach.

We can compute motion compensations as follows,

Step1: for each N by N block on an I-frame

 Compute $f(X)$ and denote by (p, q) the top left element of X .

 Set $H(p, q) = f(X)$ where H is R by S table

Step2: for a P-frame (For each macro block B ,)

 Compute $\text{temp_value} = F(B)$

 Use a window search method to find $d = \min(t - H(r, s))$

 The motion vector is $(p, q) - (r, s)$

After calculation Step1 requires $(R - N + 1) * (S - N + 1)$ values to be computed for table H . and each value computed costs One MAD. So we required total $(R - N + 1) * (S - N + 1) * 2N^2$ addition/ subtractions.

New Search Algorithm (2)

For step2 calculations macro-block computes one MAD and then searches into window doing single comparisons, as window search requires,

$$\text{Exhaustive: } 2N^2 + (2w)*(2w)$$

$$\text{Logarithmic: } 2N^2 + (5 + 4\log_2 2w)$$

Clearly we are saving time in step2. So total computation cost is Step1+ step2 that is,

$$\text{Exhaustive: } (R-N+1)*(S-N+1) * 2N^2 + (2N^2 + (2w)*(2w))*(R*S)/ N^2$$

$$\text{Logarithmic: } (R-N+1)*(S-N+1) * 2N^2 + (2N^2 + (5 + 4\log_2 2w))*(R*S)/ N^2$$

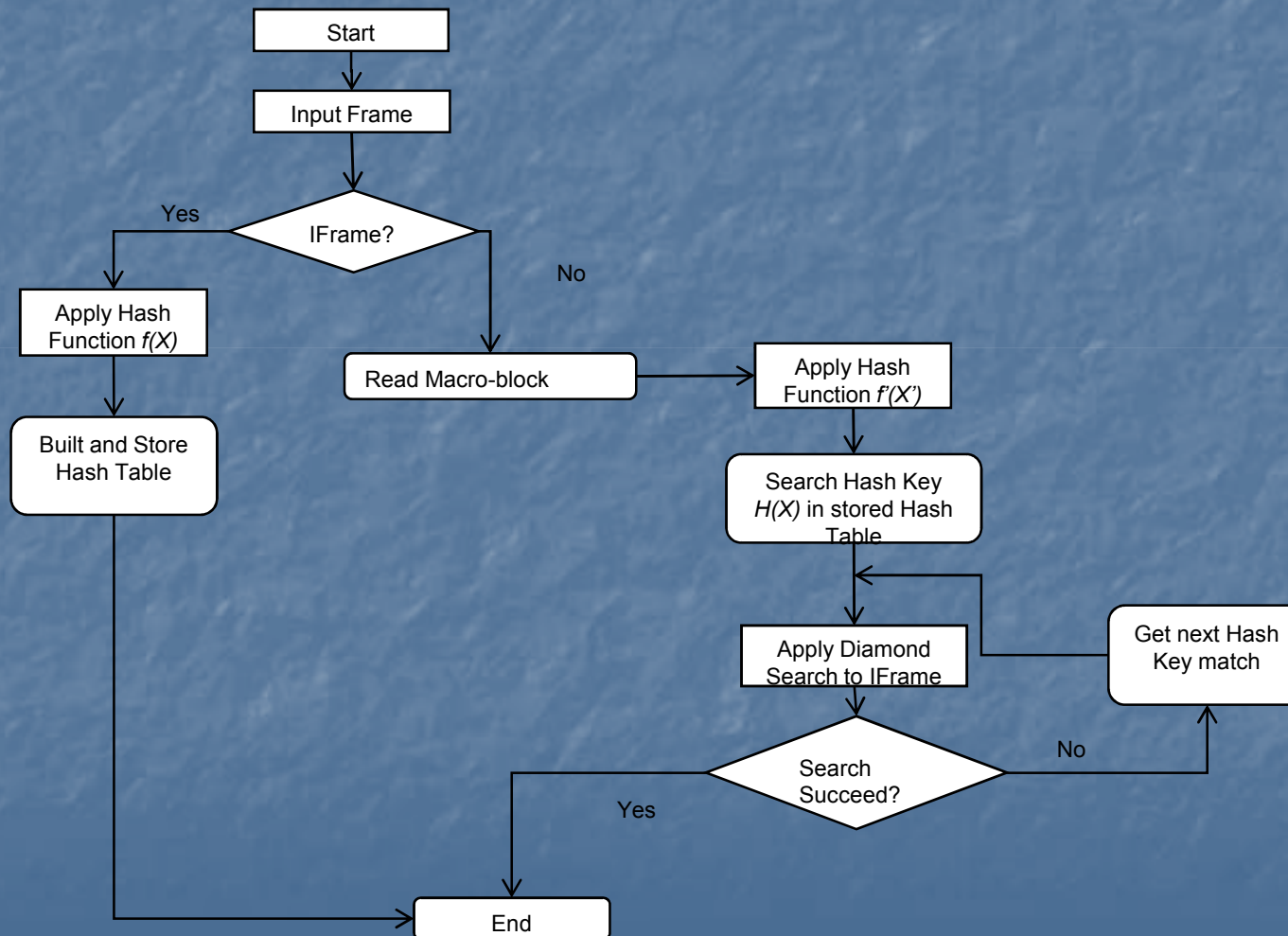
The performance of the methods measured by speedup method as below

$$S_E = [(8w^2)*(R*S)] / [(R-N+1)*(S-N+1) * 2N^2 + (2N^2 + (2w)*(2w))*(R*S)/ N^2]$$

$$S_L = [2*(5 + 4\log_2 2w) * (R*S)] / [(R-N+1)*(S-N+1) * 2N^2 + (2N^2 + (5 + 4\log_2 2w))*(R*S)/ N^2]$$

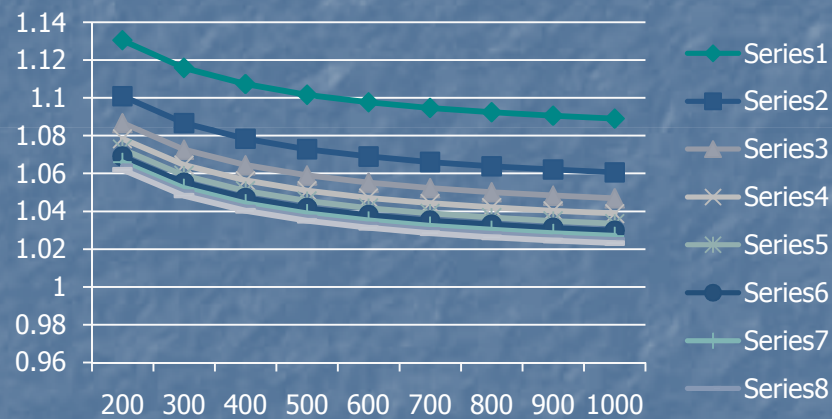
Where, SE and SL are speed-up of Exhaustive search and logarithmic search respective.

Flowchart of New Approach



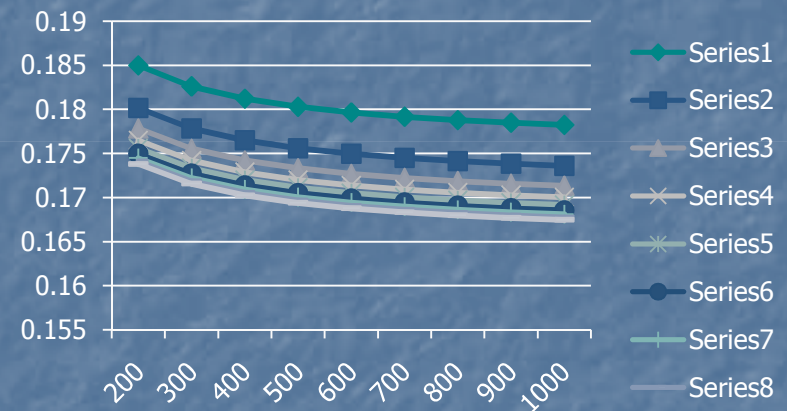
Graphical Analysis of Speed-up of New Search Algorithms

■ Exhaustive Speed-up



Graph1- Image size vs speed-up

■ Logarithmic Speed-up



Graph2- Image size vs speed-up

Reusing Linear Hash Table:

The above graphs shows no significant speed up is obtained. So the cost of building hash table can be reduced by re-using hash table for several frames.

Suppose the table H-built after every 'E' frames, then speed-up will be:

Exhaustive:

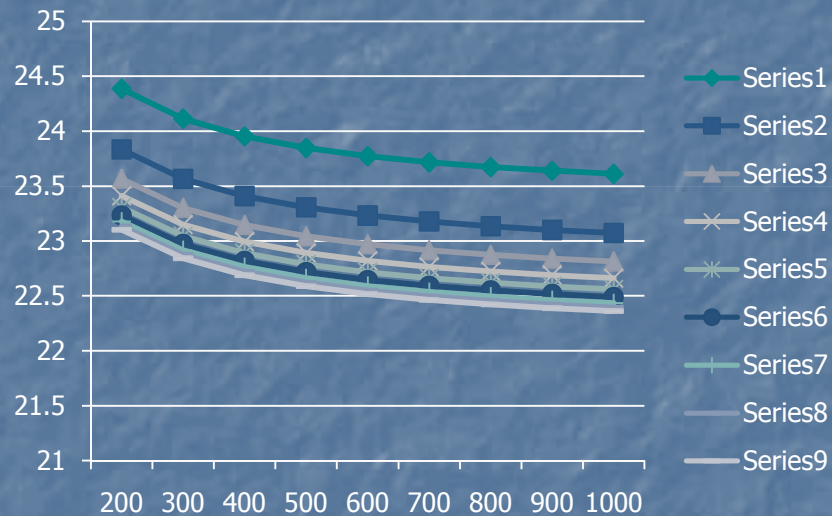
$$S_{E-re} = [E * (8w^2) * (R * S)] / [(R - N + 1) * (S - N + 1) * 2N^2 + (2N^2 + (2w) * (2w)) * (R * S) * E / N^2]$$

Logarithmic:

$$S_L = [2 * E * (5 + 4 \log_2 2w) * (R * S)] / [(R - N + 1) * (S - N + 1) * 2N^2 + (2N^2 + (5 + 4 \log_2 2w)) * (R * S) * E / N^2]$$

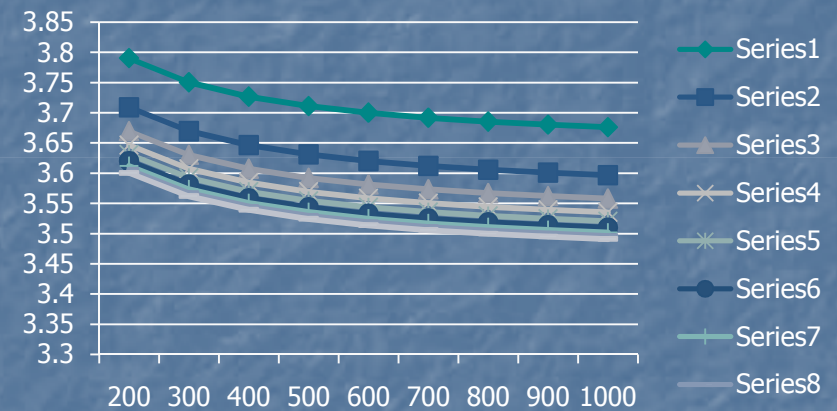
Reused Exhaustive Speed-up

Exhaustive Speed-up



Graph3- Image size vs speed-up

Logarithmic Speed-up



Graph4- Image size vs speed-up

Conclusion

- An algorithm based on a new hashing technique that achieves good quality video and speed up was introduced. Theoretical analysis of this new algorithm shows a speed-up of 1.1 for existing exhaustive search block matching algorithms (BMA) and a speed-up of 1.8 for logarithmic search BMA. Reuse of the hash table provides significant increases in speed-up, with 22.5-24.5 for exhaustive search BMA and 3.5-3.8 for logarithmic search BMA.

Thank You

Any questions please?