Design and Analysis of Adaptive Three Step Search Block Matching Criterion in Video Data

Awanish Kr. Mishra
Pranveer Singh Institute of Technology, Kanpur

Amod Tiwari, Ph.D
Bhabha Institute of Technology, Kanpur

ABSTRACT
Video Compression has played an important role in Multimedia data storage and transmission. Video compression techniques removes spatial as well as temporal redundancy using intra-frame and inter-frame coding respectively. A large level of compression can be achieved through inter-frame coding. In this paper, three step search with track knowledge has been analyzed by comparing its performance with original three step search using matching criterion in the temporal coding of video signal, which are Minimum Mean Absolute Error, Vector Matching Criterion and Smooth Constrained - Mean Absolute Error. Three step search with track knowledge has been proved more effeicient and effective for all type of video data.

Keywords
Motion Estimation, Search Parameter, Motion Compensation, MME, Motion Vector

1. INTRODUCTION
Video data has spatial as well as temporal redundancy and are removed using Video Compression Techniques [1]. Inter-frame predictive coding is used to eliminate the large amount of temporal and spatial redundancy that exists in video sequences and helps in compressing them. In conventional predictive coding the difference between the current frame and the predicted frame (based on the previous frame) is coded and transmitted.

The encoding side estimates the motion in the current frame with respect to a previous frame. A motion compensated [2] image for the current frame is then created that is built of blocks of image from the previous frame. The motion vectors for blocks used for motion estimation are transmitted, as well as the difference of the compensated image with the current frame is also JPEG encoded and sent. The encoded image that is sent is then decoded at the encoder and used as a reference frame for the subsequent frames. The decoder reverses the process and creates a full frame. The whole idea behind motion estimation based video compression is to save on bits by sending JPEG encoded difference images which inherently have less energy and can be highly compressed as compared to sending a full frame that is JPEG encoded. It should be noted that the first frame is always sent full, and so are some other frames that might occur at some regular interval (like every 6th frame). The standards do not specify this and this might change with every video being sent based on the dynamics of the video. The most computationally expensive and resource hungry operation in the entire compression process is motion estimation. Hence, this field has seen the highest activity and research interest in the past two decades. This paper implements and evaluates the fundamental block matching algorithm that is Three Step Search[3] (TSS) using different search criterion.

The better the prediction, the smaller the error and hence the transmission bit rate. If a scene is still, thengood prediction for a particular pixel in the current frame is the same pixel in the previous frame and the error is zero. However, when there is motion in a sequence, then a pixel on the same part of the moving object is a better prediction for the current pixel. The use of the knowledge of the displacement of an object in successive frames is called Motion Compensation. There are a large number of motion compensation algorithms for inter-frame predictive coding. In this study, however, we have focused only on one class of such algorithms, called the Block Matching Algorithms. These algorithms estimate the amount of motion on a block by block basis, i.e. for each block in the current frame, a block from the previous frame is found, that is said to match this block based on a certain criterion.

One of the first algorithms to be used for block based motion compensation is what is called the Full Search or the Exhaustive Search. In this, each block within a given search window is compared to the current block and the best match is obtained (based on one of the comparison criterion). Although, this algorithm is the best one in terms of the quality of the predicted image and the simplicity of the algorithm, it is very computationally intensive. Some of the efficient block-based search algorithms are Exhaustive Search (ES), Three Step Search[3] (TSS), New TSS[4], Four Step Search[5] (FSS), Diamond Search[6,7].

2. BLOCK MATCHING ALGORITHM
The underlying supposition behind motion estimation is that the patterns corresponding to objects and background in a frame of video sequence move within the frame to form corresponding objects on the subsequent frame. The idea behind block matching is to divide the current frame into a matrix of 'macro blocks' that are then compared with corresponding block and its adjacent neighbors in the previous frame to create a vector that stipulates the movement of a macro block from one location to another in the previous frame. This movement calculated for all the macro blocks comprising a frame, constitutes the motion estimated in the current frame. The search area for a good macro block match is constrained up to p pixels on all four sides of the corresponding macro block in previous frame.

This 'p' is called as the search parameter. Larger motions require a larger p, and the larger the search parameter the more computationally intensive the process of motion estimation becomes. Usually the macro block is taken as a square of side 16 pixels, and the search parameter p is 7 pixels. The matching of one macro block with another is based on the output of a cost function. The macro block that results in the least cost is the one that matches the closest to current block.

2.1 Exhaustive Search (ES)

This algorithm, also known as Full Search, is the most computationally expensive block matching algorithm of all. This algorithm calculates the cost function at each possible location in the search window. As a result of which it finds the best possible match and gives the highest PSNR amongst any block matching algorithm. Fast block matching algorithms try to achieve the same PSNR doing as little computation as possible. The obvious disadvantage to ES is that the larger the search window gets the more computations it requires.

2.2 Three Step Search (TSS)

This TSS [3] is one of the most successful attempts to find correct motion vector in block matching algorithms during the decade of 80s.

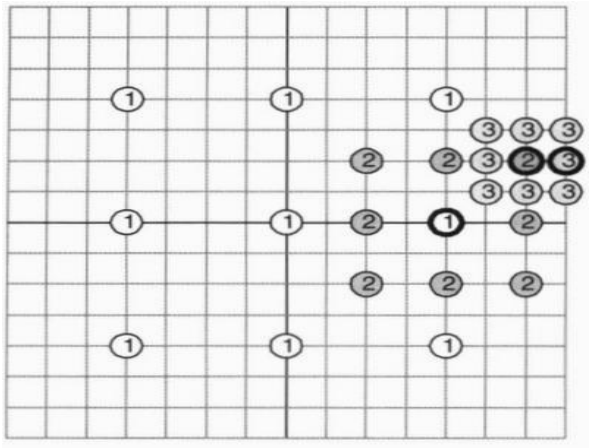

Figure. 1 Three Step Search

The general behaviour is represented in Figure 1. It starts with the search location at the center and sets the 'step size' S = 4, for a usual search parameter value of 7. It then searches at eight different locations +/- S pixels around location (0,0). From these nine locations searched so far, it picks the one giving best result (least cost) and makes it the new search origin for next round of three step search. It then sets the new step size S = S/2, and repeats similar search for two more iterations until S = 1. At that point it finds the location with the least cost function and the macro block at that location is the best match, hence the desired block is found. The calculated motion vector is then saved for transmission. It gives a flat reduction in computation by a factor of 9. So that for p = 7, ES will compute cost for

225 macro blocks whereas TSS computes cost for 25 macro blocks only. The idea behind TSS is that the error surface due to motion in every macro block is unimodal. A unimodal surface is a bowl shaped surface such that the weights generated by the cost function increase monotonically from the global minimum.

2.3 New Three Step Search (NTSS)

NTSS [4] is an advancement of TSS. NTSS shows good results by providing a center based searching scheme and having provisions for half way stop to reduce computational cost. It was widely accepted fast search algorithm and frequently used for implementing earlier standards like MPEG 1 and H.261.

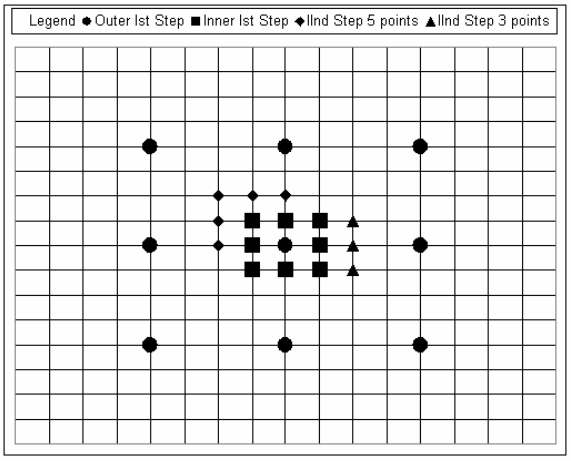

Figure. 2 New Three Step Search

The TSS uses a uniformly allocated checking pattern for motion detection and is prone to missing small motions. The NTSS process is illustrated graphically in Figure 2. In the first step 16 points are checked in addition to the search origin for lowest weight using a cost function. Of these additional search locations, 8 are at distance of S = 4 away (similar to TSS) and the other 8 are at S = 1 away from the search origin. If the lowest cost is at the origin then the search is stopped right here and the motion vector is set as (0, 0). If the lowest weight is at any one of the 8 locations at S = 1, then we change the origin of the search to that point and check for weights adjacent to it. Depending on which point it is we might end up checking 5 points or 3 points. The location that gives the lowest weight is the closest match and motion vector is set to that location. On the other hand if the lowest weight after the first step was one of the 8 locations at S = 4, then we follow the normal TSS procedure. Hence although this process might need a minimum of 17 points to check every macro block, it also has the worst-case scenario of 33 locations to check.

2.4 Four Step Search (4SS)

Like NTSS, 4SS [5] is also a center based searching and may stop its searching in halfway. 4SS sets a fixed pattern size of S = 2 for the first step, no matter what the search parameter p value is. Thus it looks at 9 locations in a 5x5 window. If the least weight is found at the center of search window the search jumps to fourth step. If the least weight is at one of the eight locations except the center, then we make it the search origin and move to the second step.

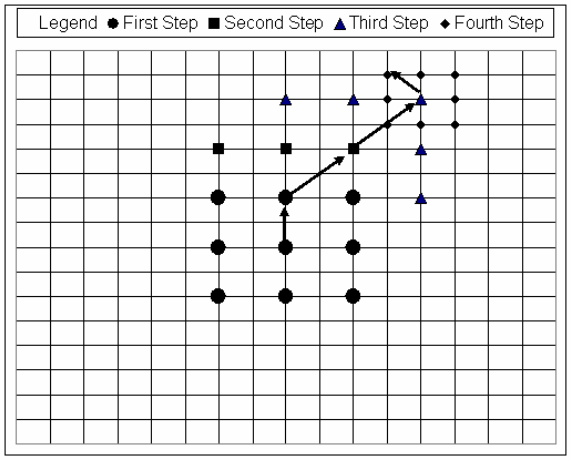

Figure. 3 Four Step Search

The search window is still maintained as 5x5 pixels wide. Depending on where the least weight location was, we might end up checking weights at 3 locations or 5 locations. The patterns are shown in Figure 3. Once again if the least weight location is at the center of the 5x5 search window we jump to fourth step or else we move on to third step. The third is exactly the same as the second step. In the fourth step the window size is dropped to 3x3, i.e. S = 1. The location with the least weight is the best matching macro block and the motion vector is set to point o that location. A sample procedure is shown in Figure 4. This search algorithm has the best case of 17 checking points and worst case of 27 checking points.

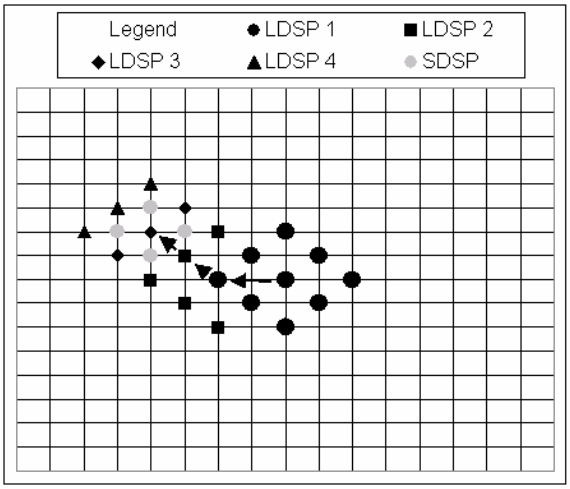

Figure. 4 Diamond Search

2.5 Diamond Search (DS)

DS [7] algorithm is very much same as 4SS, only the search point pattern is modified from a square shape to a diamond shape, and there is no limit on the number of steps that the algorithm can take. It takes two different types of fixed patterns, one is for Large Diamond Search Pattern (LDSP) and the other is for Small Diamond Search Pattern (SDSP). Both these patterns and the DS algorithm are illustrated in Figure 4. Similar to FSS, the first step uses LDSP and if the least weight is at the center location we jump to fourth step. The consecutive steps, except the final step, are also similar and use LDSP, but the number of points where cost function is checked are either 3 or 5 and are illustrated in second and third steps of procedure shown in Figure 4. The final step uses SDSP around the new decided search origin and the location with the least weight is the best match. As the search pattern is neither too small nor too big and the fact that there is no limit to the number of steps, this algorithm can find global minimum very accurately and efficiently. The end result should see a PSNR close to that of ES while computational expense should be very less.

2.6 Proposed Adaptive Three Step Search (ATSS)

ATSS algorithm makes use of the fact that the general motion in a frame is usually coherent, i.e. if the macro blocks around the current macro block moved in a particular direction then there is a high probability that the current macro block will also have a similar motion vector. This algorithm uses the motion vector of the macro block to its immediate left to predict its own motion vector. If the predicted motion vector is not the correct choice according to the selected matching criterion, then the motion vector is searched using three step search block matching technique. This time selected motion vector gives the predicted motion vector for the next block search. This ATSS is always the first step in search of matching motion vector for the second block onwards. For the first block matching motion vector is computed using three step search only.

3. MATCHING CRITERION

3.1 Mean Absolute Error Criteria

The matching criteria mostly used in the literature is minimum mean absolute error, which at point (i, j) for an NxN block and search window of size ±p, is defined as –

$$MAE(i,j) = \frac{1}{N^2}\sum_{x,y}|c(x,y) - r(x+i, y+j)| \quad (1)$$

where, −p<= i , j<= +p and c(x,y) and r(x,y) are pixel values at position (x,y) in the current and reference frame respectively. Motion vector is defined as the value of (i, j) for which MAE(i, j) is minimum. Obviously, the residue error between the predicted and actual block in the current frame should be minimum for good matching.

3.2 Vector Matching Criteria

In MAE based criteria, the average error value is considered while ignoring the individual error term.

S. Wang and H. Chen proposed vector matching criteria for block matching to overcome this drawback. In this approach, each NxN block is represented by a vector. Further, each block is subdivided into smaller blocks of size like 2x2, which is represented by a component of the corresponding vector and MAE is calculated between each temporally adjacent subblock in the current and reference frame.

A threshold value is chosen by exhaustive search and vector components (out of N2 / 4 ,assuming the subblock size as 2x2) having value smaller than the threshold value are counted for a given block. Finally, the block having maximum number of such vector components within the defined search area is declared to be the best matching block.

3.3 Smooth Constrained – Mean Absolute Error Criteria

In video data compression, the residue frame which is calculated by taking the difference of the current and the predicted frame, is coded using transform coding technique, called Discrete Cosine Transform (DCT). According to the characteristics of this transform , the number of bits required to code a smooth residue frame will be smaller than the non smooth residue frame. Therefore, X. Jing, C. Zhu and L. Chau , proposed a smooth contrained based MAE as block matching criteria for motion compensation to reduce the required number of bits for coding besides minimising the total distortion. In this method, not only the MAE over the residue block is taken into consideration but also the maximum and minimum residue value error, denoted as MME, is taken care of as well. Since DCT is applied over 8x8 block, each residue block (16x16) is divided into four equal size subblocks (8x8) and MME is calculated for each subblock as

$$MME_i = r^i_{max} - r^i_{min} \quad (2)$$

$$SC - MAE = MAE + \alpha \sum_{i=1}^{4} MME_i \quad (3)$$

where alpha is a weighing factor. The block which has minimum SC-MAE value in the search area, is declared as the best matched block.

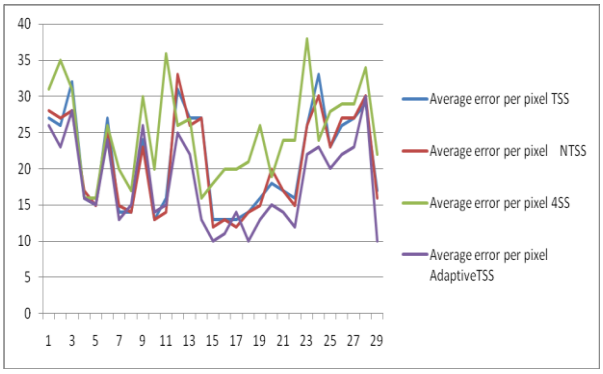

Figure 5 Comparison of Average error per pixel Kamin2.avi

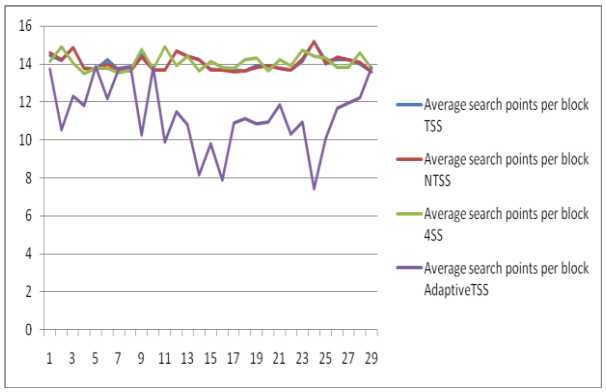

Figure 6 Comparison of Average search points per block for Kamin2.avi

4. EXPERIMENTAL RESULTS

In finding the results from the different Block matching algorithms using different matching criteria for the block based search, two sample videos have been used in this research for comparison Kamin2.avi and Susie.avi. Results from TSS, NTSS, 4SS along with the proposed ATSS are given in tabular form and also are shown using graphs for the comparison. These experiments have been performed in terms of two parameters- average error per pixel and average search points per block on two videos. For areal time video results from the proposed ATSS are very motivating.

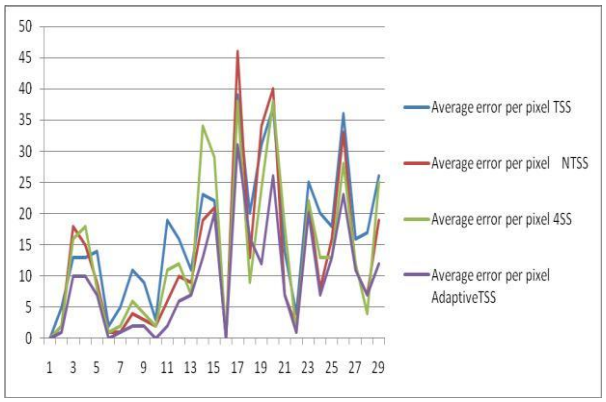

Figure 7 Comparison of Average error per pixel for Susie.avi

It has been observed that the proposed matching algorithm gives the better results in comparison to that of TSS, NTSS and 4SS.

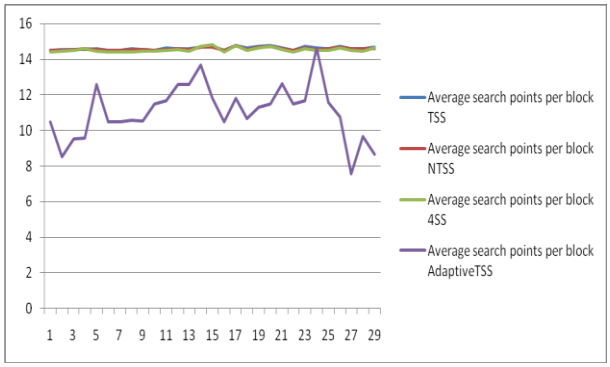

Figure 8 Comparison of Average search points per block for Susie.avi

As for the videos Kamin2 and Susie.avi, variation in the intensities of adjacent frames are very high, experimental results are very much in favor of proposed matching algorithm. Proposed algorithm improves the results in the sequence of increasing frames distance i.e. for more frame distance proposed algorithm gives better results when compared with other techniques. These results have been found experimentally by taking first thirty frames form these mentioned videos. Detailed results are given in the tables and corresponding graphs.

5. CONCLUSIONS

A new block matching technique for prediction of motion vector has been proposed and experimentally examined with three other existing methods in terms of average search points per block, average error per pixel for two videos (kamin2.avi and Susie.avi) inputs with different size and varying degree of motion. The proposed algorithm gives much better results in the case when video quality fades i.e. same pels in different frames have the different intensity and this difference in intensity for the redundant pels increases by the time.

6. REFERENCES

[1] T.Sikora, MPEG Digital Video Coding Standards, IEEE Signal Processing Magazine, pp 82-100, (September 1997)

[2] J.R.Jain and A.K.Jain, Displacement Measurement and its application in Interframe Coding, IEEE Transactions on Communications, Volume 29, No 12, (December 1981)

[3] T.Koga, K.Iinuma, A.Hirano and Y.Ishiguro, Motion Compensated Interframe Coding for Video Conferencing, Proc NTC81, New Orleans, LA, (November 1981)

[4] R.Li, B.Zeng and M.L.Liou, A New Three Step Search Algorithm for Block Motion Estimation, IEEE Transactions on Circuits and Systems for Video Technology, Volume 4, No 4, (August 1994)

[5] L.Man Po and W.C.Ma, A Novel Four Step Search Algorithm for Fast Block Motion Estimation, IEEE transactions on Circuits and Systems for Video Technology, Volume 6, No 3, (June 1996)

[6] S.Zhu and K.K.Ma, A New Diamond Search Algorithm for Block Matching Motion Estimation, IEEE transactions on Image Processing, Volume 9, No 2, (February 2000)

[7] C.H.Cheung and L.M.Po, A Novel Cross Diamond Search Algorithm for Fast Block Motion Estimation, IEEE transactions on Circuits and Systems for Video Technology, Volume 12, No 12, (December 2002)

Table 1 Average error per pixel for Kamin2.avi

Average error per pixel			
TSS	**NTSS**	**4SS**	**AdaptiveTSS**
27	28	31	26
26	27	35	23
32	28	31	28
16	17	16	16
15	15	16	15
27	25	26	24
14	15	20	13
14	14	17	15
24	23	30	26
13	13	20	14
16	14	36	15
31	33	26	25
27	26	27	22
27	27	16	13
13	12	18	10
13	13	20	11
13	12	20	14
14	14	21	10
16	15	26	13
18	20	19	15
17	17	24	14
16	15	24	12
26	26	38	22
33	30	24	23
23	23	28	20
26	27	29	22
27	27	29	23
29	30	34	30
17	16	22	10

Table 2 Average search points per block for Kamin2.avi

Average search points per block			
TSS	**NTSS**	**4SS**	**AdaptiveTSS**
14.45	14.56	14.15	13.71
14.15	14.23	14.89	10.53
14.85	14.85	14.05	12.32
13.75	13.75	13.51	11.81
13.71	13.71	13.79	13.85
14.21	14.00	13.76	12.15
13.64	13.61	13.55	13.75
13.75	13.61	13.68	13.85
14.49	14.35	14.76	10.25
13.64	13.64	13.72	13.81
13.64	13.65	14.87	9.92
14.69	14.69	13.91	11.49
14.39	14.39	14.40	10.81
14.20	14.20	13.65	8.17
13.71	13.64	14.15	9.79
13.68	13.68	13.80	7.89
13.64	13.57	13.76	10.89
13.61	13.61	14.23	11.11
13.87	13.80	14.31	10.87
13.87	13.87	13.65	10.93
13.76	13.76	14.23	11.87
13.68	13.68	13.91	10.29
14.13	14.24	14.71	10.93
15.17	15.17	14.41	7.45
14.15	14.00	14.29	10.07
14.23	14.33	13.80	11.68
14.21	14.19	13.80	11.93
13.96	14.07	14.59	12.21
13.57	13.57	13.79	13.81

Table 3 Average error per pixel for Susie.avi

Average error per pixel			
TSS	**NTSS**	**4SS**	**AdaptiveTSS**
0	0	0	0
5	2	2	1
13	18	16	10
13	15	18	10
14	9	8	7
2	1	1	0
5	1	2	1
11	4	6	2
9	3	4	2
3	2	2	0
19	6	11	2
16	10	12	6
11	9	7	7
23	19	34	13
22	21	29	20
0	0	0	0
39	46	38	31
20	13	9	16
31	34	24	12
37	40	38	26
14	7	18	7
4	2	1	1
25	22	22	20
20	8	13	7
18	16	13	13
36	33	28	23
16	11	12	11
17	7	4	7
26	19	25	12

Table 4 Average search points per block for Susie.avi

Average search points per block			
TSS	**NTSS**	**4SS**	**AdaptiveTSS**
14.50	14.50	14.43	10.50
14.53	14.52	14.48	8.53
14.54	14.56	14.53	9.52
14.57	14.62	14.62	9.57
14.59	14.55	14.48	12.59
14.50	14.50	14.45	10.50
14.51	14.52	14.45	10.50
14.62	14.53	14.45	10.58
14.55	14.55	14.48	10.55
14.50	14.52	14.48	11.50
14.65	14.53	14.50	11.69
14.62	14.59	14.55	12.62
14.58	14.55	14.48	12.58
14.69	14.69	14.73	13.69
14.76	14.71	14.82	11.84
14.50	14.50	14.43	10.50
14.78	14.79	14.77	11.80
14.64	14.57	14.52	10.69
14.72	14.67	14.65	11.33
14.79	14.76	14.72	11.48
14.65	14.58	14.58	12.65
14.51	14.50	14.43	11.51
14.72	14.65	14.60	11.69
14.65	14.57	14.50	14.65
14.58	14.59	14.53	11.58
14.76	14.71	14.66	10.76
14.58	14.59	14.52	7.58
14.59	14.62	14.49	9.66
14.69	14.62	14.64	8.67