資科工碩一 309551027 李美慧

(一) Autocorrelation

(1) 推導加上時間軸變化之 Autocorrelation Function Estimator $R_{xx}[m_x, m_y, m_t]$

- $R_{xx}[m_x, m_y, m_t]$ estimator,

$$\widehat{R_{xx}}[m_x, m_y, m_t] = \frac{1}{L_x - |m_x|}\frac{1}{L_y - |m_y|}\frac{1}{L_t - |m_t|} C_{vv}[m_x, m_y, m_t]$$

- $C_{vv}[m_x, m_y, m_t]$

$$\begin{cases} = \sum_{n_y=0}^{L_y-1-|m_y|} \sum_{n_x=0}^{L_x-1-|m_x|} \sum_{n_t=0}^{L_t-1-|m_t|} X[n_x, n_y, n_t]\, X[n_x + |m_x|, n_y + |m_y|, n_t + |m_t|] \\ \qquad \text{for } |m_x| \le L_x - 1, |m_y| \le L_y - 1, |m_t| \le L_t - 1 \\ = 0, \text{ otherwise} \end{cases}$$

- Unbiasedness: $E[\widehat{R_{xx}}[m_x, m_y, m_t]] = R_{xx}[m_x, m_y, m_t]$
  for $|m_x| \le L_x - 1, |m_y| \le L_y - 1, |m_t| \le L_t - 1$

(2) 解釋個別視訊序列中 Y 元件的 Autocorrelation 影像

AKIYO is more static than MOBILE since MOBILE contains horizontal move and zoom out while AKIYO just contains a person always in almost the same position.
Hence, the results of auto correlation of AKIYO is almost the same among all mt, while the results of auto correlation of MOBILE looks different.

AKIYO:

All the 21 frames look similar because AKIYO is a more static video. The frame of |mt| = 0 is brighter than all the other frame, and frames become darker when |mt| is bigger. The phenomenon shows the closer frame have higher correlation, which is reasonable since this is a consistent video.

MOBILE:

When |mt| > 0, one pixel is more similar to the horizontal near pixels of previous frame rather than the pixel at the same position, which leads to rxx[mt, 0, 0] is not the maximum value (not the white dot in the frame). When |mt| is larger, the largest value of rxx is farther from center because the most similar pixel is going away when |mt| becomes larger.

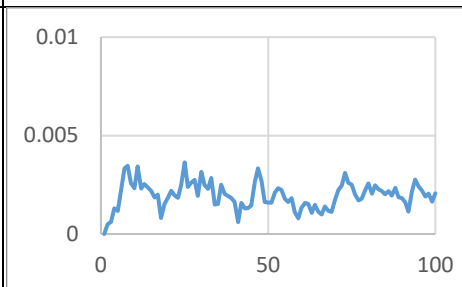
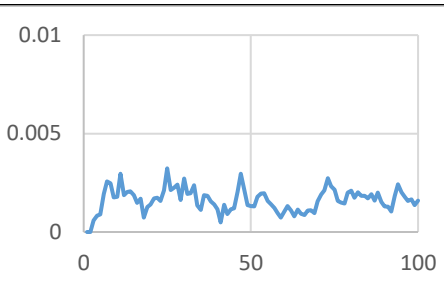
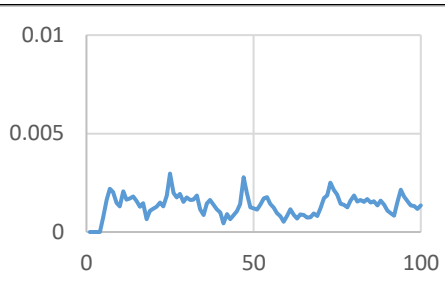
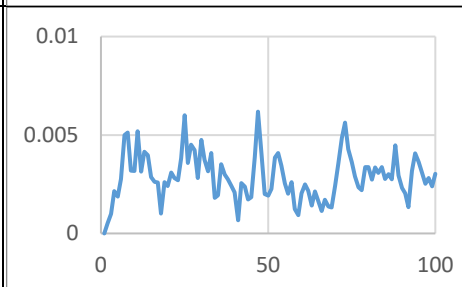
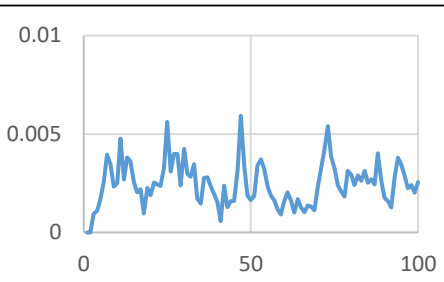
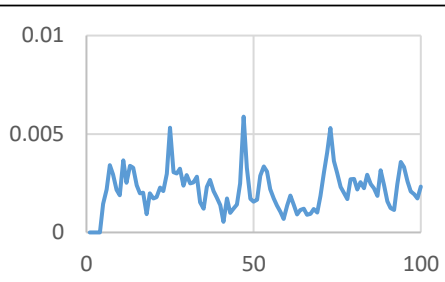
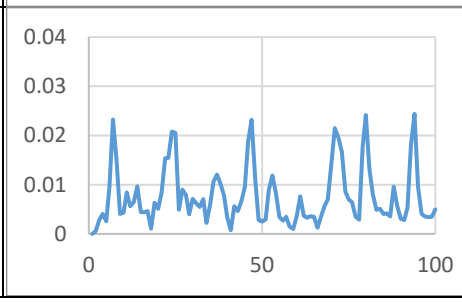
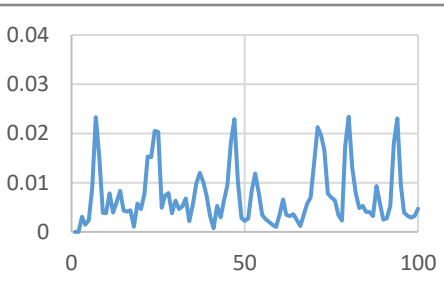
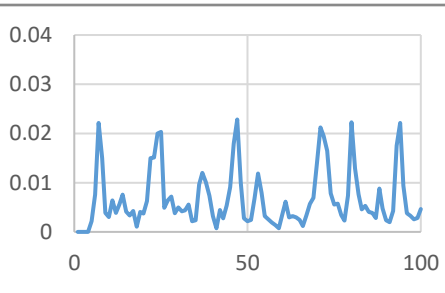
(3) 與未做動作補償[題(2)]之結果做比較

For each |mt| after motion compensation, rxx[|mt|][mx][my] is an auto correlation from original frame to the similar frame obtained from motion compensation of previous |mt| frames. Since the motion compensation video is similar with original video, every frame looks quite similar for all mt.

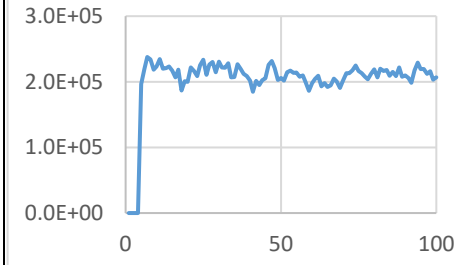
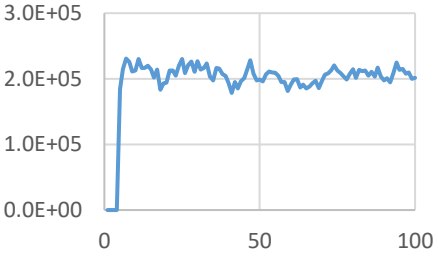
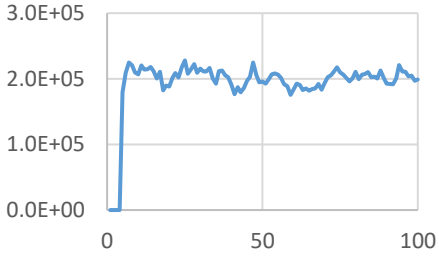
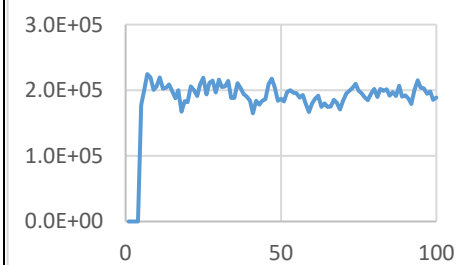
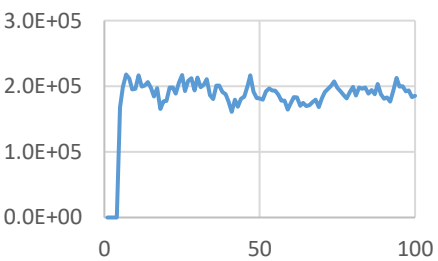
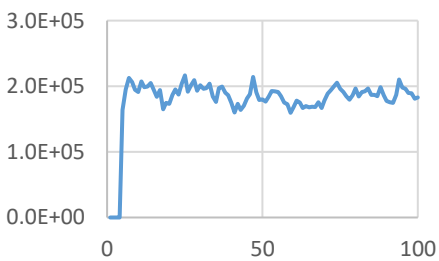
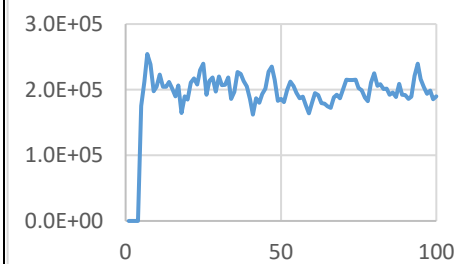
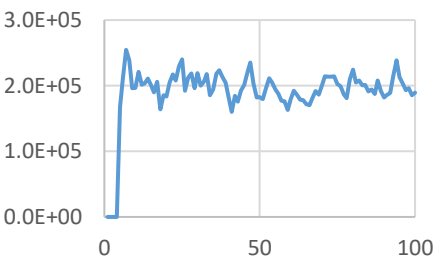
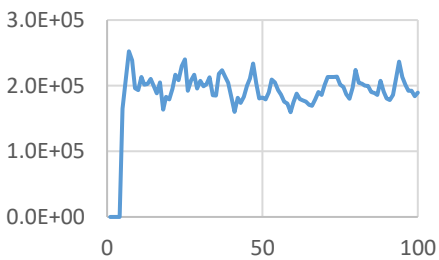
Every frame in rxx(1-3) are just looks like the frame of rxx[10] (1-2), which is the auto correlation to the same frame itself.

(二) Temporal Prediction (1) AKIYO

AKIYO, $\frac{\sigma_e^2}{\sigma_X^2}$

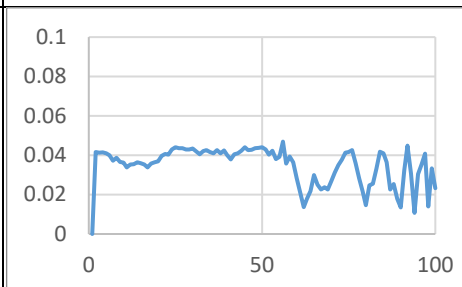
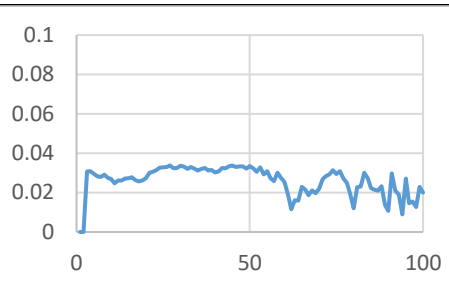
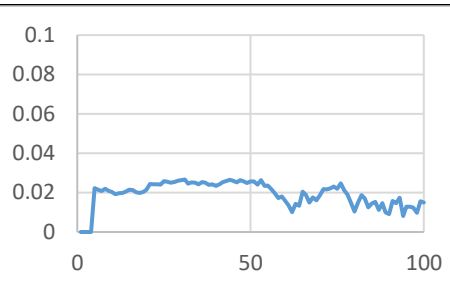
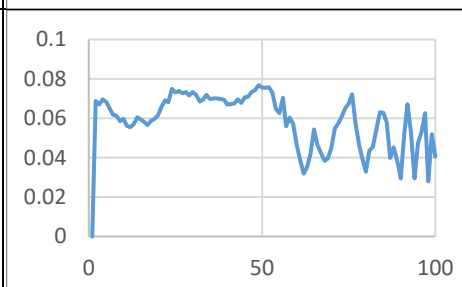
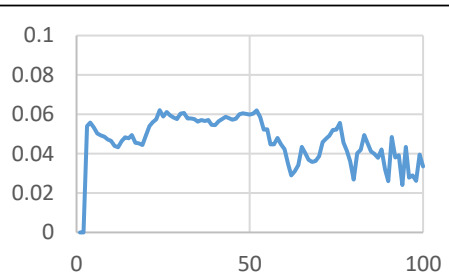
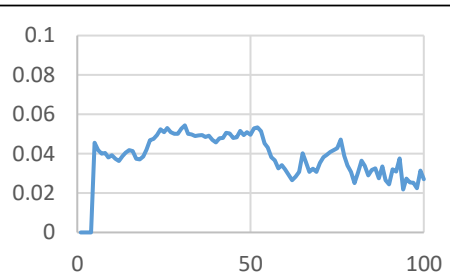
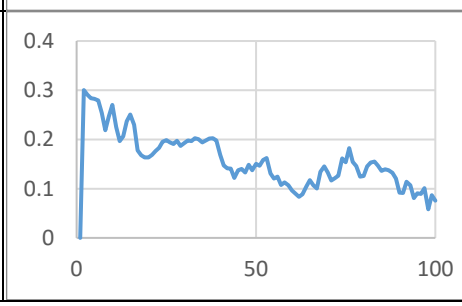
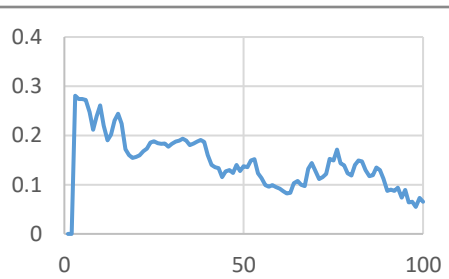
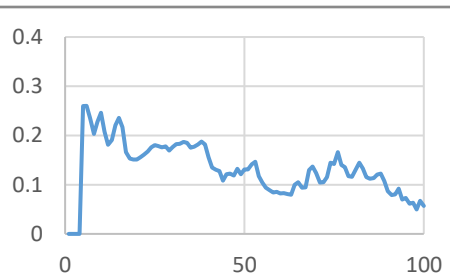
	N = 1	N = 2	N = 4
n = 4			
n = 8			
n = 16			

AKIYO, Bits Per Frame

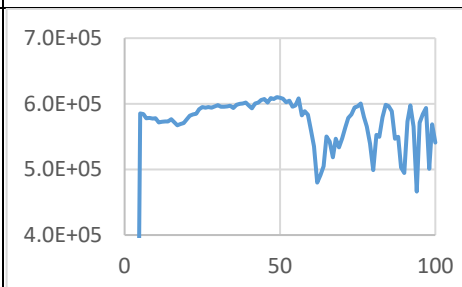
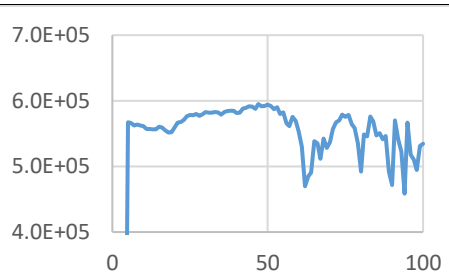
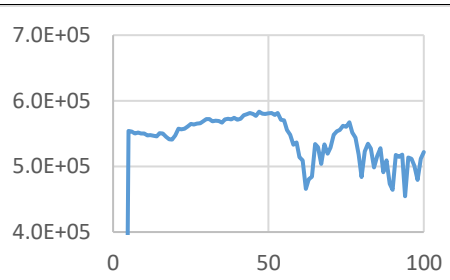
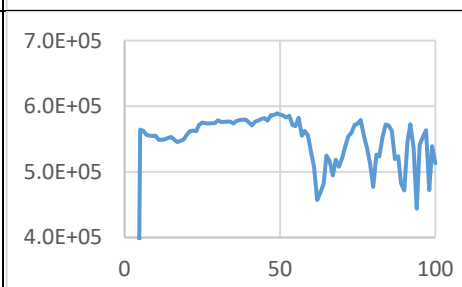
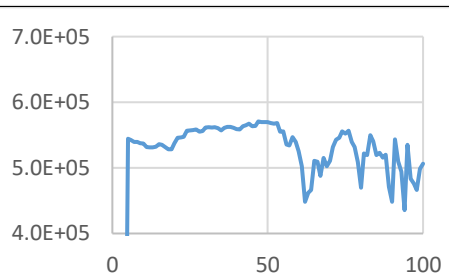
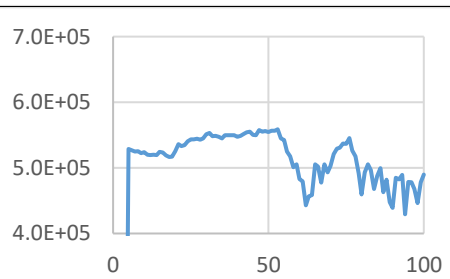
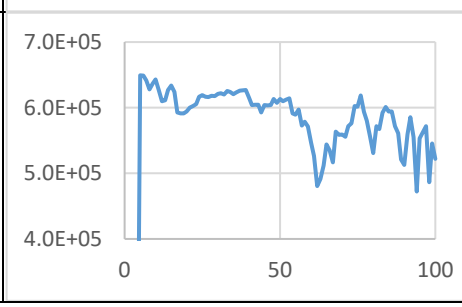
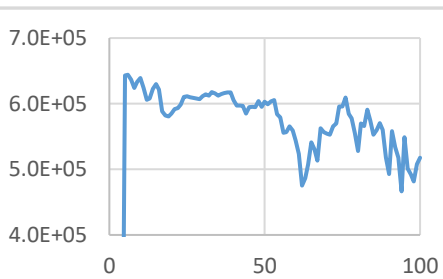
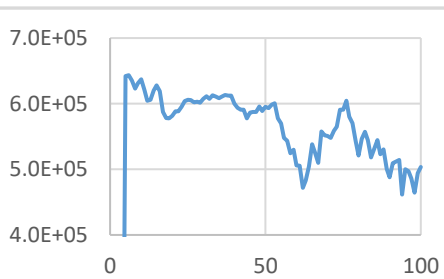
	N = 1	N = 2	N = 4
n = 4			
n = 8			
n = 16			

(二) Temporal Prediction (1) MOBILE

MOBILE, $\frac{\sigma_e^2}{\sigma_X^2}$

	N = 1	N = 2	N = 4
n = 4			
n = 8			
n = 16			

MOBILE, Bits Per Frame

	N = 1	N = 2	N = 4
n = 4			
n = 8			
n = 16			

AKIYO, Total Bits

	N = 1	N = 2	N = 4
n = 4	20358555	19799753	19385107
n = 8	18698412	18295764	17955565
n = 16	19281406	19097664	18887768

MOBILE, Total Bits

	N = 1	N = 2	N = 4
n = 4	55140569	53573586	52116696
n = 8	52865306	51198745	49607094
n = 16	56420616	55441750	54518996

(二) Temporal Prediction (2) Wiener Filter

Let N be number of pixels which is required to predict the next pixel.

Let n be number of input/output pairs.

X: input pixels (N*n 2D matrix),

d: desired pixels (n-vector),

$y = w^T * X$: predict pixels

(y is (1*N)*(N*n) = n vector)

e = d – y: prediction error. (n-vector)

$w_o = \arg \min_{w[n]} E[e^2]$ (N-vector)

$$\Rightarrow \frac{de^2}{dw_o} = \frac{d(d^2-2dy+y^2)}{dw_o} = \frac{d(-2dy)}{dw_o} + \frac{d(y^2)}{dw_o} = \frac{d(-2d^T w_o{}^T X)}{dw_o} + \frac{d(w_o{}^T (XX^T) w_o)}{dw_o} = -2d^T X^T + 2w_o{}^T (XX^T) = 0$$

$$\Rightarrow d^T X^T = w_o{}^T (XX^T) \rightarrow w_o = d^T X^T (XX^T)^{-1},\ d^T X^T = (1*n)*(n*N) = N\text{-vector}$$

For N = 1:

(XX^T) and $d^T X^T$ are both scalars; hence, $w_o = \frac{d^T X^T}{XX^T}$ is a scalar, too.

To get w_o, we have to get $d^T X^T$ (scalar) and XX^T (scalar) first.

For N = 2:

(XX^T) is a 2*2 matrix, $w_o = d^T X^T (XX^T)^{-1}$ is a 2-vector.

To get w_o, we have to get $d^T X^T$ (2-vector) and inverse of XX^T ((2*2 matrix)) first.

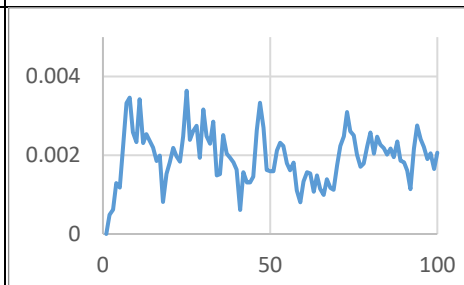
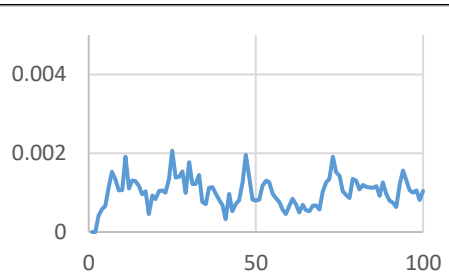
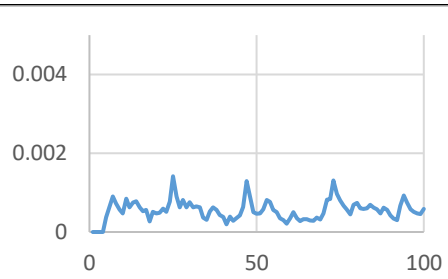
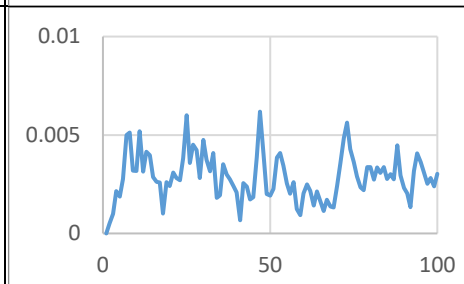
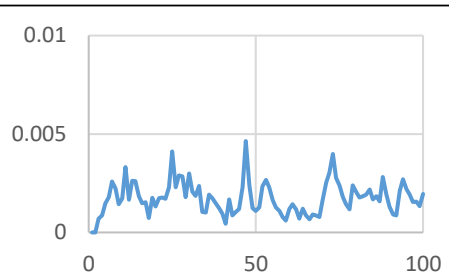
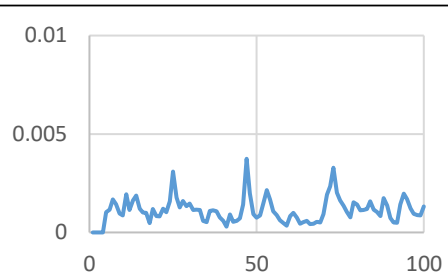
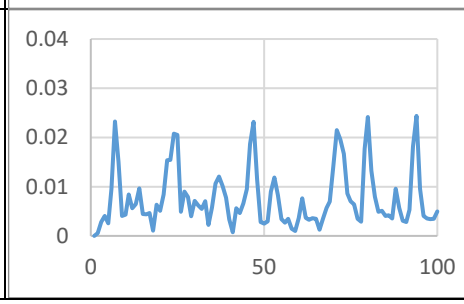
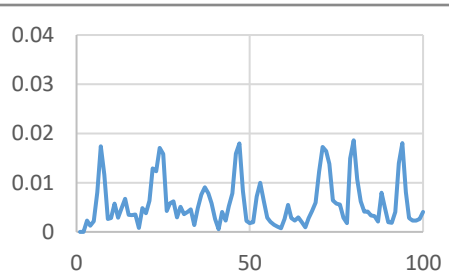
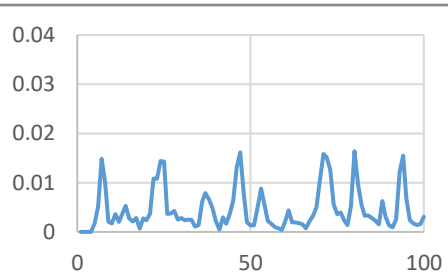
For N = 4:

(XX^T) is a 4*4 matrix, $w_o = d^T X^T (XX^T)^{-1}$ is a 4-vector.

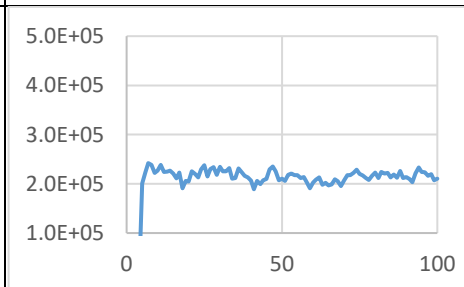
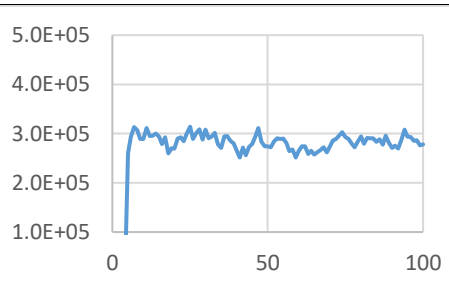
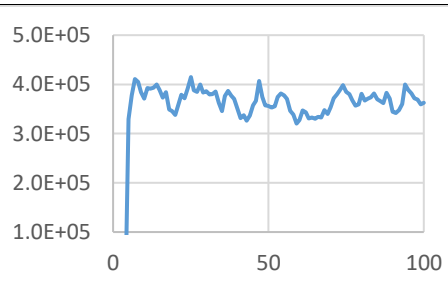
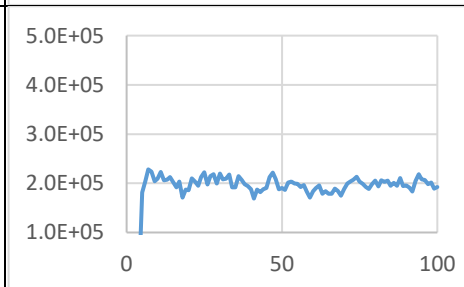
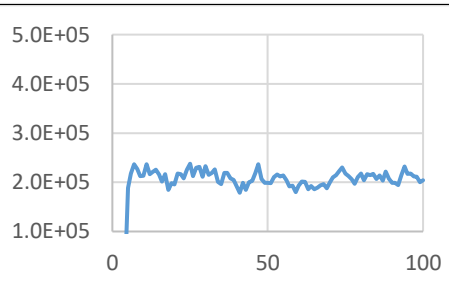
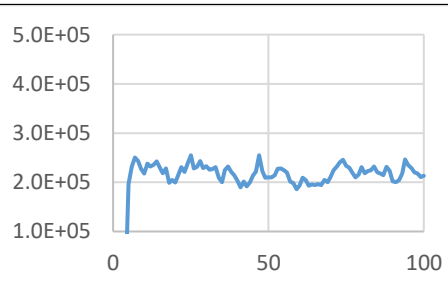
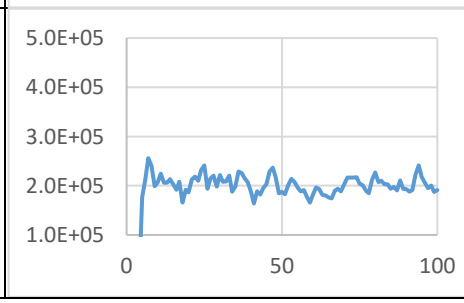
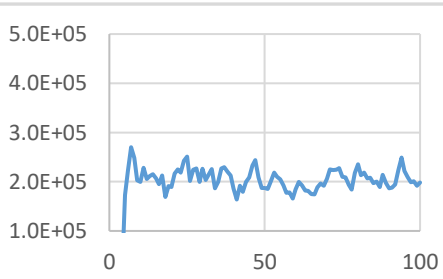
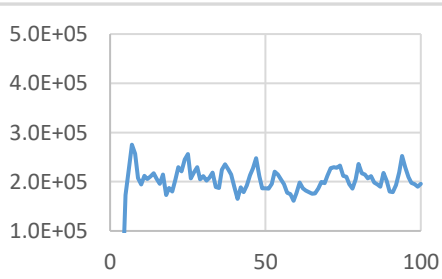
To get w_o, we have to get $d^T X^T$ (4-vector) and inverse of XX^T ((4*4 matrix)) first.

(二) Temporal Prediction (3) Wiener Filter AKIYO

AKIYO, $\frac{\sigma_e^2}{\sigma_X^2}$

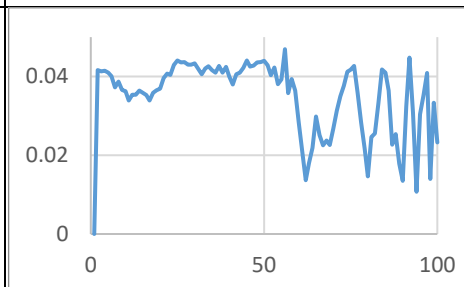
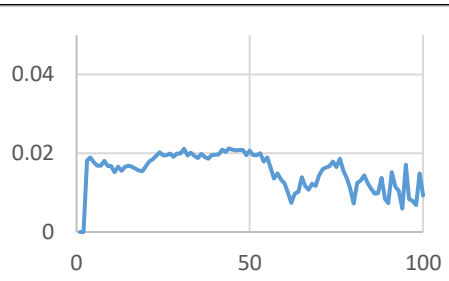
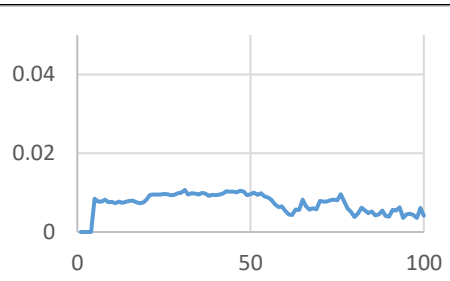
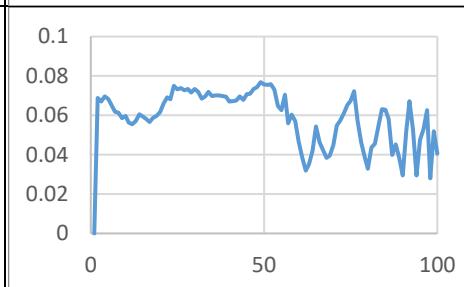
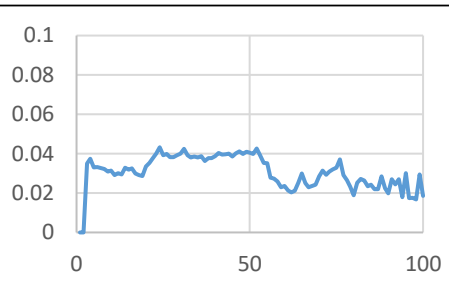
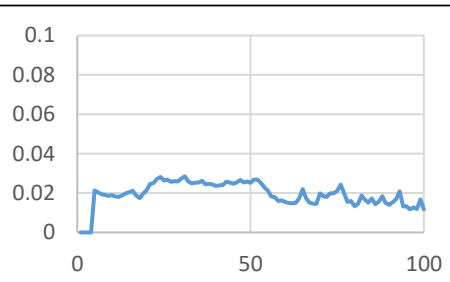
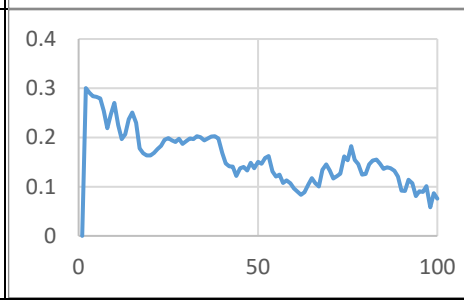
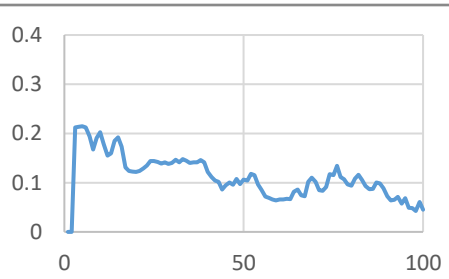
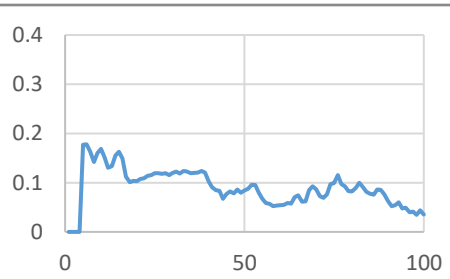
	N = 1	N = 2	N = 4
n = 4			
n = 8			
n = 16			

AKIYO, Bits Per Frame

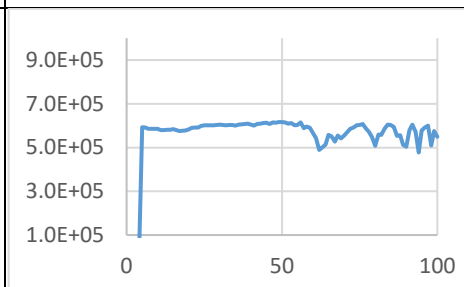
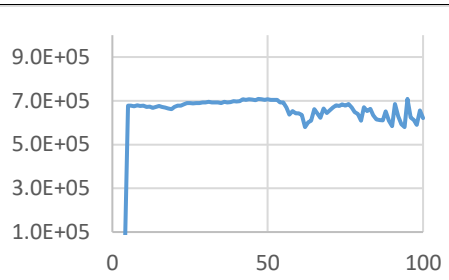
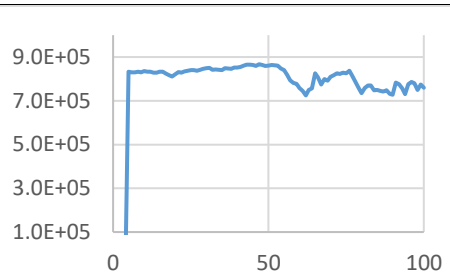
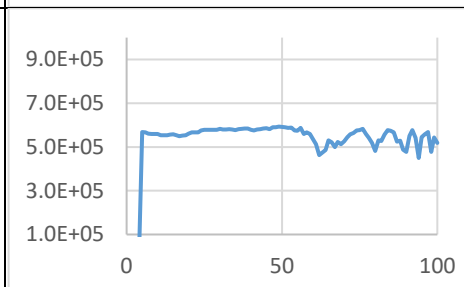
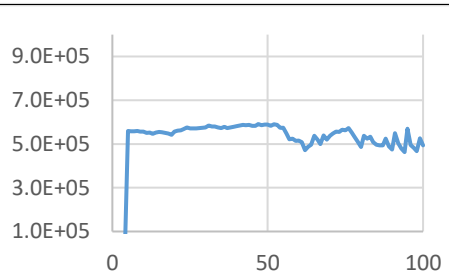
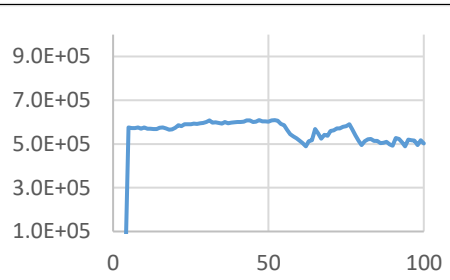
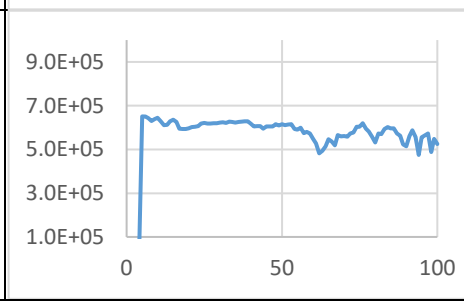
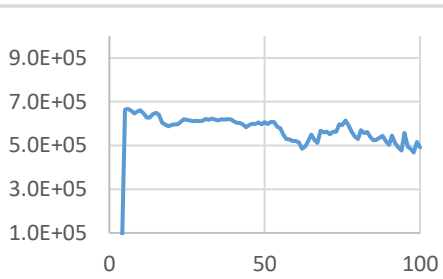
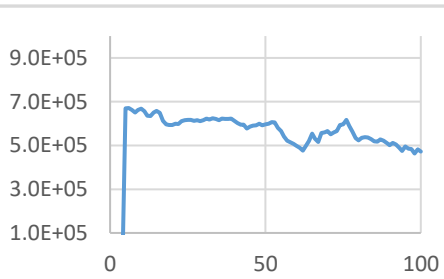
	N = 1	N = 2	N = 4
n = 4			
n = 8			
n = 16			

(二) Temporal Prediction (3) Wiener Filter MOBILE

MOBILE, $\frac{\sigma_e^2}{\sigma_X^2}$

	N = 1	N = 2	N = 4
n = 4			
n = 8			
n = 16			

MOBILE, Bits Per Frame

	N = 1	N = 2	N = 4
n = 4			
n = 8			
n = 16			

AKIYO, Total Bits

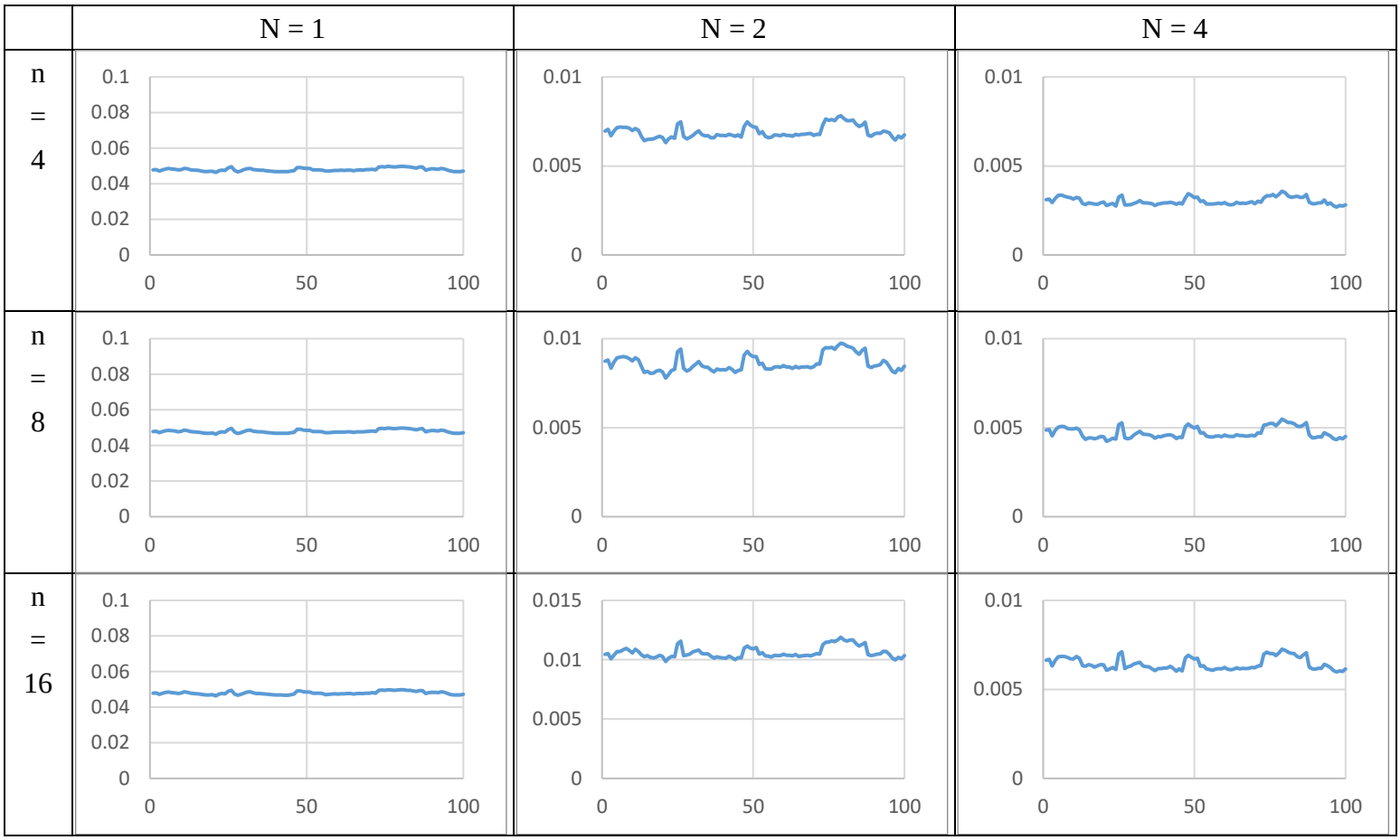
	N = 1	N = 2	N = 4
n = 4	20757918	27234302	35238272
n = 8	19050782	20046160	21010506
n = 16	19438981	19747084	19707961

MOBILE, Total Bits

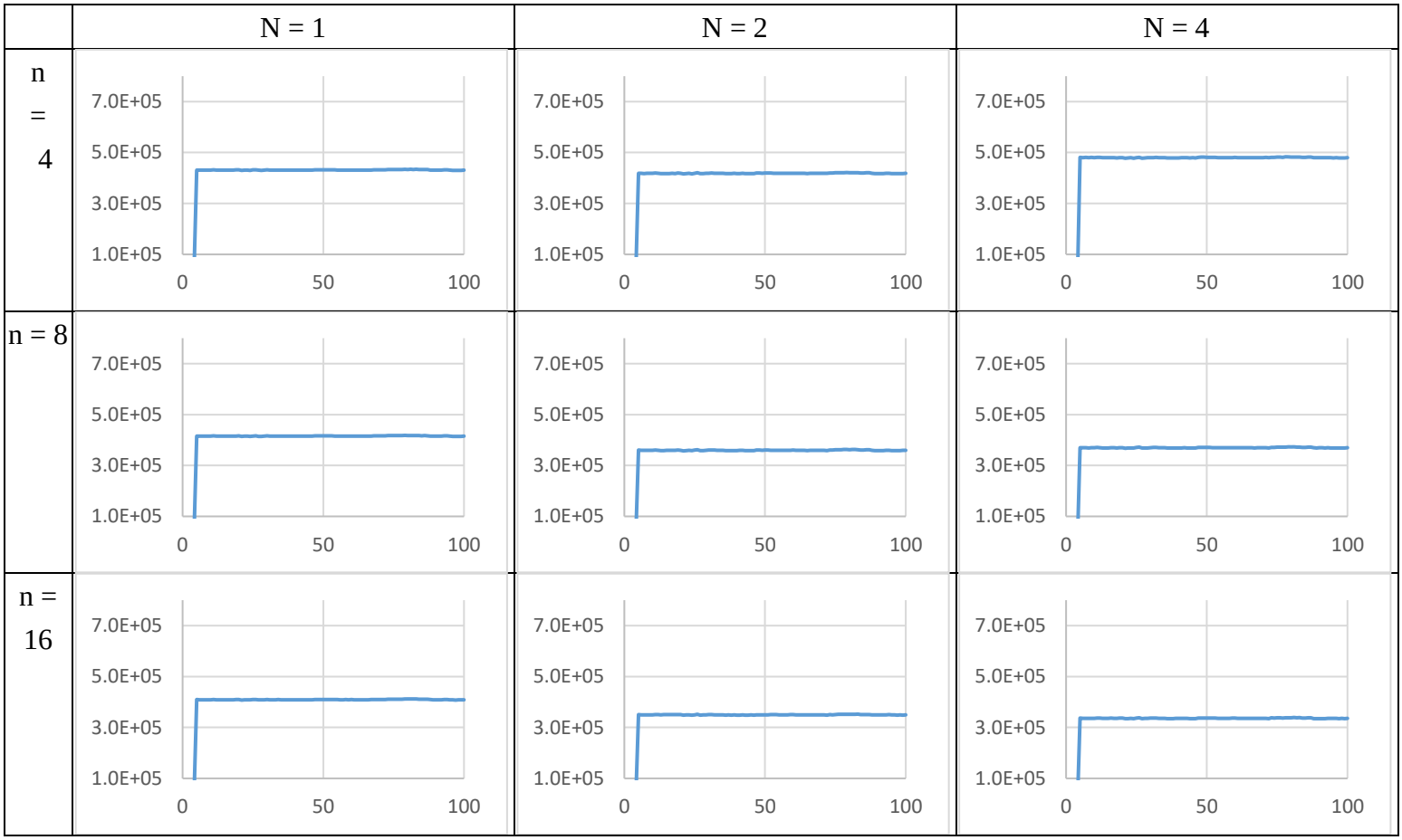
	N = 1	N = 2	N = 4
n = 4	55836060	63917766	77921663
n = 8	53288416	52399716	53856935
n = 16	56591569	55458434	55101287

(三) Spatial Prediction (3) Wiener Filter AKIYO

AKIYO, $\frac{\sigma_e^2}{\sigma_X^2}$

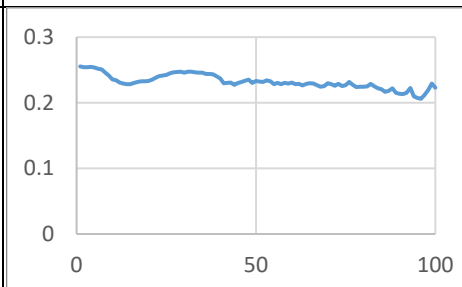
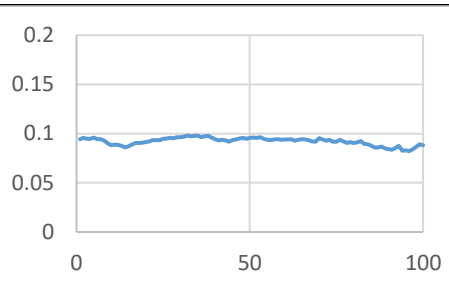
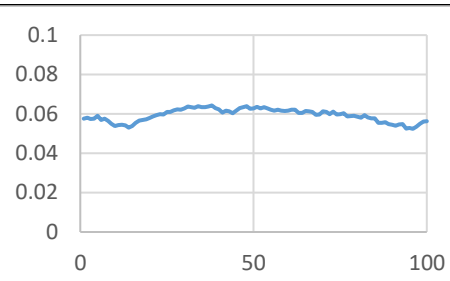
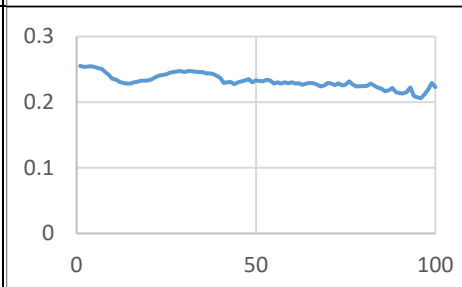
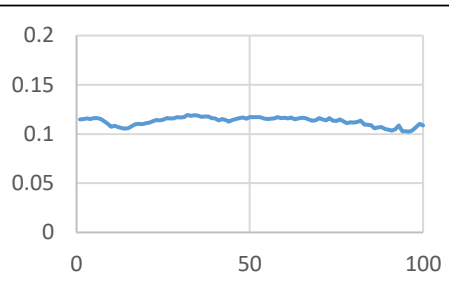
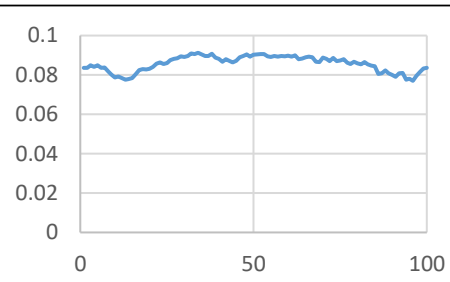
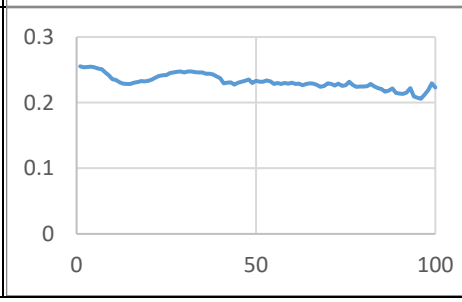
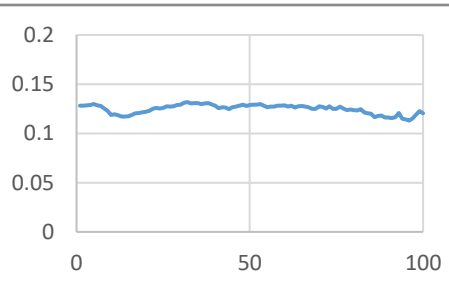
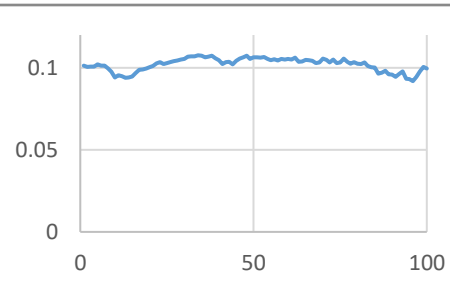


AKIYO, Bits Per Frame

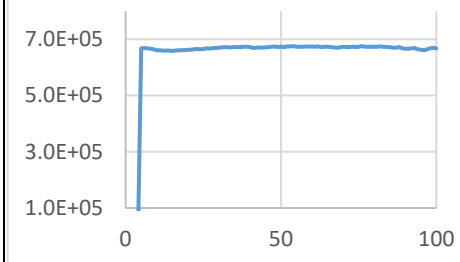
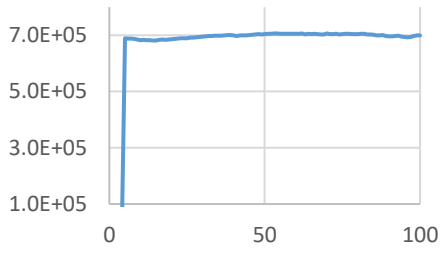
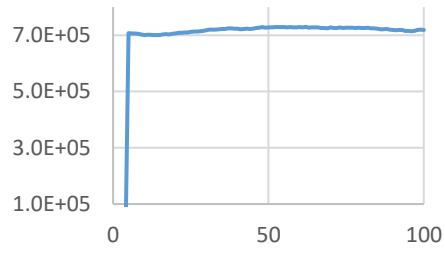
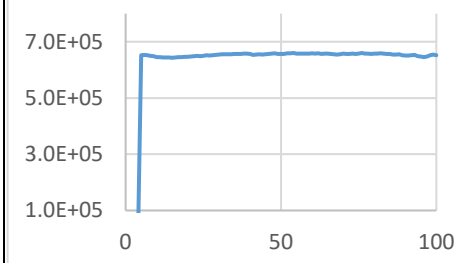
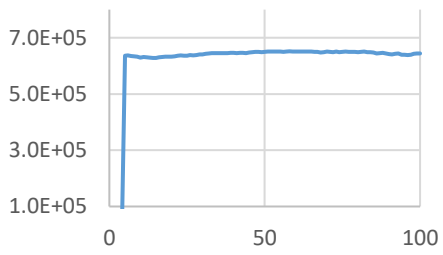
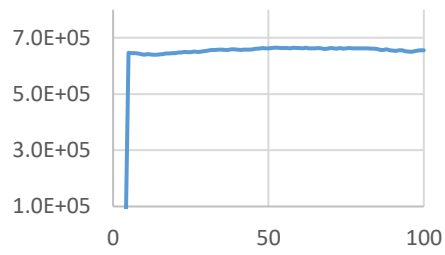
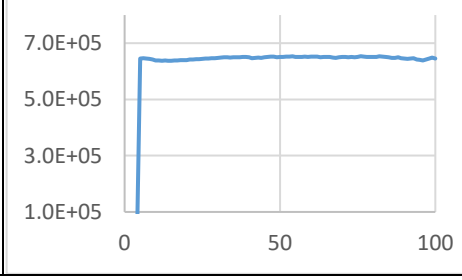
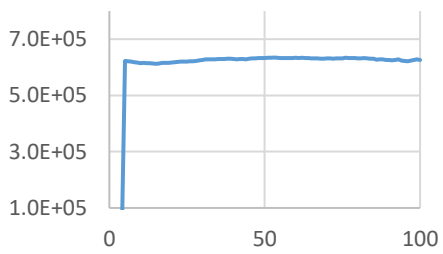
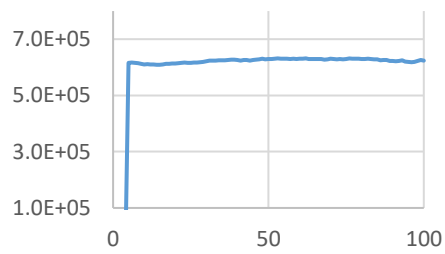


(三) Spatial Prediction (3) Wiener Filter MOBILE

MOBILE, $\frac{\sigma_e^2}{\sigma_X^2}$

	N = 1	N = 2	N = 4
n = 4			
n = 8			
n = 16			

MOBILE, Bits Per Frame

	N = 1	N = 2	N = 4
n = 4			
n = 8			
n = 16			

AKIYO, Total Bits

	N = 1	N = 2	N = 4
n = 4	41429136	40186122	46109995
n = 8	39908489	34543506	35504754
n = 16	39300228	33589038	32275261

MOBILE, Total Bits

	N = 1	N = 2	N = 4
n = 4	64282649	66979129	69089125
n = 8	62762002	61803531	62950865
n = 16	62153744	60203692	59873557

(四) 請比較所有方法之編碼效率

	(1) Temporal Prediction Only motion vector	(2) Temporal Prediction With wiener filter	(3) Spatial Prediction With wiener filter
Best AKIYO total bits	17955565	19050782	32275261
Best MOBILE total bits	49607094	52399716	59873557

Despite (1) has higher σ_e^2 overall, (1) doesn't have to send any wiener filter coefficients since only motion vector of mt, mx, my and residual is required. Therefore, (1) requires least number of total bits overall. Compared to spatial prediction (3), temporal prediction (1) and (2) require less bits for both AKIYO and MOBILE, which shows both AKIYO and MOBILE are highly temporal related compared to the spatial relation.

For (1), large N has good performance since we can decrease residual a little when we have more reference frames. n = 8 have the best performance on both AKIYO and MOBILE although n = 4 has the least σ_e^2. This is probably because smaller n requires a lot more number of wiener filter coefficient, which costs more bits than the reduction of bits from lower residual.

For (2), we can observe a tradeoff between small n and large n. A small n (4) leads to least σ_e^2 but requires more number of wiener filter coefficient which includes a lot information and costs more bits. On the other hand, a bigger n (16) has less number of wiener filter coefficient but get higher σ_e^2 which costs more bits for each frame. Thus, we can observe that the best number of compressed bits happens when n = 8.
As for N, AKIYO has the best number of compressed bits when N = 1, while MOBILE has the best number of compressed bits when N = 2. This is probably because AKIYO is a more static video compared to MOBILE, which means only one previous frame is enough to predict the next frame. However, MOBILE includes zoom out and move in video, which means this is a more dynamic video and requires a little more previous information to predict the next frame.

For (3), each frame required similar number of bits since (3) is spatial prediction rather than temporal prediction. More number of reference bits N have a significant impact on decreasing overall σ_e^2. On the other hand, the smaller block size n also achieve lower σ_e^2, but smaller n requires more wiener filter coefficient and is not as effective as a little bit larger N. Therefore, when N = 4, n = 16, we get the least number of total bits to compress the video.