2017 International Conference on Computational Science and Computational Intelligence

Learning Via Decision Trees Approach for Video Super-Resolution

Yu-Zhu Zhang, Wan-Chi Siu, *Life-FIEEE*, Zhi-Song Liu, and Ngai-Fong Law, *SrMIEEE*
Center for Signal Processing, Department of Electronic and Information Engineering
Hong Kong Polytechnic University, Hong Kong

***Abstract*— In this paper, we propose an unsupervised learning-based multi-frame video super-resolution (SR) approach via decision trees model (DTSRV). This novel approach utilizes the temporal redundancy and coherence in consecutive video frames. Motion estimation is applied between consecutive frames to form concatenated motion compensated patches. The low resolution (LR) - high resolution (HR) pairs are then formed to be the training input of the decision trees. After the classification process with decision trees, a linear regression model is learnt to map the relationship between the concatenated LR patches and the HR patches. Results of our experiments show that the approach outperforms state-of-the-art model-based algorithms with an average of 0.97 dB PSNR increase and a much faster speed. It also achieves a 1.4 dB better results for large video sizes than the frame-by-frame image SR using decision trees learning techniques. This is the first time reporting in the literature to use comprehensive random trees/forests structures for video SR. Now the scheme only utilizes two neighbor frames and can already have a good result, which proves its efficiency in real-time application. Our analysis also proves it to have more promising possibilities and advantages for future development.**

Keywords—Video Super-Resolution, Multi-frame Super-Resolution, Learning, Decision Trees, Motion Estimation

I. INTRODUCTION

The goal of super-resolution (SR) is to obtain better the high-resolution (HR) parts from low-resolution (LR) observations of an image/video. For the general model, the observed low-resolution image/patch $\boldsymbol{Y}$ is assumed to be obtained through applying a down-sampling kernel $\boldsymbol{D}$ and a blurring kernel $\boldsymbol{B}$ to the original high-resolution image/patch $\boldsymbol{X}$. And additional Gaussian noise n is also included in the process.

$$\boldsymbol{Y} = \boldsymbol{DBX} + n \quad (1)$$

The context of video super-resolution was initiated years ago [1] and multi-frame SR schemes focus on reconstructing one HR frame from a series of LR frames, by using the temporal coherence to improve the reconstructed results. Multi-frame SR methods can be classified into two categories, the model-based approaches and example-based approaches.

Most of current model-based methods [2,3,4] apply an iterative de-convolution framework with regulations to solve this ill-posed problem. State-of-the-art Bayesian [2] approach introduces sparsity priors and smoothness priors on derivative filter responses to simultaneously model the priors of image, optical flow field and the blur kernel. This method is the foundation for other model-based SR methods in handling complex motions and larger blurs [3,4]. However, these model-based methods are limited by their high computational complexities and their local registrations may not be reliable especially for large frame sizes. Besides, they generally apply over thirty consecutive frames for achieving a good SR result, which is incapable for real-time applications.

For learning-based methods, external examples are used for learning the mapping functions from LR patches to HR patches. Though machine learning has shown a great success on SR, very few works are targeted at video contents. Recently, a few video SR based on learning techniques appeared, such as bi-level dictionary [5] and CNN [6]. All of them focus on 4K size of videos, but the methods also have high computational complexity and need rather long training time. Using Decision Trees for image Super-Resolution (DTSR) [7,8] has also shown promising results in real-time image SR applications. In this paper, we extend this scheme to video content and propose a learning-based effective Decision Tree Super-Resolution approach for Video (DTSRV). We apply a linear regression model to map the relationship between the LR and HR patches. Through motion estimation, extra information from neighbor frames is concatenated with the current patch for joint training and estimation. Decision trees model shows its effectiveness in simultaneous classification and regression. The proposed scheme can achieve a good PSNR results with only two neighbor frames. Therefore it is promising for real-time applications.

II. METHODOLOGY

The DTSRV scheme contains two processes: training and up-sampling. During the training process, a dictionary is learnt for each decision tree to map the relationship between LR and HR information. This is an offline process to make the model suitable for real-time application. For the up-sampling process, the LR frames are super-resolved with the obtained dictionary for online testing.

A. Decision Trees Training for Video Content

The consecutive T video HR frames $\{\boldsymbol{Y}^1,\boldsymbol{Y}^2,...,\boldsymbol{Y}^T\}$ are down-sampled by Bicubic interpolation to obtain the initial LR frames $\{\boldsymbol{X}_0{}^1, \boldsymbol{X}_0{}^2,...,\boldsymbol{X}_0{}^T\}$. Then these initial frames are up-sampled by Bicubic filter to get the LR frames $\{\boldsymbol{Y}_0{}^1, \boldsymbol{Y}_0{}^2,..., \boldsymbol{Y}_0{}^T\}$denoted in this paper.

We assume that if the video has no sudden scene change, the additional details could be found in the similar patches in neighbor frames. The patches in the neighboring frames which have a slight displacement to the patch in the current frame are considered to have a high probability containing the losing pixels in LR frames. Therefore the training data is in the form of LR_N-HR patch pairs, which could be formed as follow.

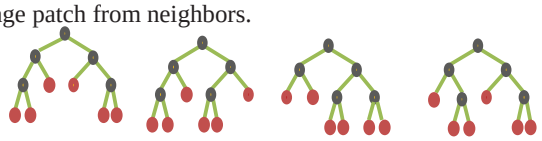
$$P = \{(\boldsymbol{L}_i^*, \boldsymbol{H}_i)\},\ i = 1...N \quad (2)$$

978-1-5386-2652-8/17 $31.00 © 2017 IEEE
DOI 10.1109/CSCI.2017.95

Note that N is the total number of data for training and n is the number of data points entering into a node. LR_N contains concatenated patches $\boldsymbol{L}_i^* \in \boldsymbol{R}^{2d}$ formed by LR patch $\boldsymbol{L}_i^t$ sampled in frame $\boldsymbol{Y}_0^t$ (i.e. frame *t*) and a similar patch $\boldsymbol{L}_i^{t*}$ searched in the neighbor frame $\boldsymbol{Y}_0^{t-1}$ or $\boldsymbol{Y}_0^{t+1}$. The HR patch $\mathbf{H}_i \in \boldsymbol{R}^d$ is an image patch extracted from the HR video frame $\boldsymbol{Y}^t$. The size of any sampled image patch is $\sqrt{d} \times \sqrt{d}$.

To form the concatenated $\boldsymbol{L}_i^*$ data, pixel-based motion estimation is applied to get similar patches in neighbor frames**.** Both the full search algorithm of window size 10 and diamond search fast algorithm are implemented for comparison. The similarity between patches is assessed based on the sum of absolute intensity differences (***SAD***) and we apply both an upper and lower threshold to select the similar ones.

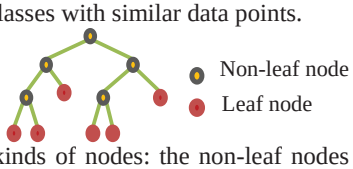
During the motion estimation process, the x-y position displacement of the patch $\boldsymbol{L}_i^t$ in current frame and a similar one $\boldsymbol{L}_i^{t*}$ in the neighbor frame is denoted as motion vector (***MV***). Depending on the parity of this motion vector and *SADs*, the total input LR_N-HR pairs are classified into four groups and a decision tree is constructed in each group. If the ***MV*** *(x,y)* is *(odd, even)*, the patch pair will enter Tree 2's database. Similarly for the *(even, odd)* and *(odd, odd)* cases, the pair will enter Tree 3's and Tree 4's database respectively. For the patch pair with *(even, even)* ***MV***s or if no similar patch within the acceptable ***SAD*** range is found after motion estimation, the current data point is classified into Tree 1's database. So in Tree 1's database, the data point only contains the HR patch and current LR patch without concatenating the image patch from neighbors.



Tree 1 Tree 2 Tree 3 Tree 4

MV: (odd, even) MV: (even, odd) MV: (odd, odd)

Decision tree [7,8] is used to classify the input training data into a number of classes with similar data points.



There are two kinds of nodes: the non-leaf nodes evaluate the LR data points according to the binary test and pass this LR patch to its left or right child node until a leaf node is reached. And each leaf node contains a regression model to represent the relationship between the arriving input concatenated LR data points and the corresponding HR patches**.** Because of the small patch sizes, we can assume the mapping function $\boldsymbol{C}$ between the HR data points $\boldsymbol{H}$ and the concatenated patch $\boldsymbol{L}$ is linear. Hence for the current node with n incoming patches, its estimated regression model is obtained as

$$\text{argmin} \| \boldsymbol{H} - \boldsymbol{CL} \|_2^2 \tag{3}$$

then the closed form solution becomes

$$\boldsymbol{C} = \boldsymbol{H}\boldsymbol{L}^{\mathrm{T}}(\boldsymbol{L}\boldsymbol{L}^{\mathrm{T}})^{-1} \tag{4}$$

where $\boldsymbol{C} \in \mathrm{R}^d \times \mathrm{R}^{2d}$, $\boldsymbol{H} \in \mathrm{R}^d \times \mathrm{R}^n$, $\boldsymbol{L} \in \mathrm{R}^{2d} \times \mathrm{R}^n$.

Assume that there are n training data points at node k and define the data set $S_k = \{P_k \mid i = 1,2,...n\}$. A binary split function $h(\boldsymbol{L}_i^*, \theta)$ is performed to classify the input data to its left and right child nodes.

$$h(\boldsymbol{L}_i^*, \theta) = \begin{cases} 0, & \textit{if} \quad \boldsymbol{L}_i^*(p) < \boldsymbol{L}_i^*(q) + \tau \\ 1, & \textit{otherwise} \end{cases} \tag{5}$$

where each binary test θ has three parameters $\theta = \{p, q, \tau\}$, where p, q are randomly generated for comparing the pixel intensity values within the image LR patch and τ is the randomly selected threshold value. The output result 0 means the data would be passed to the left child node and output result 1 means the data would go to the right child node.

To get the optimized binary test, the error for each possible binary test is calculated as follow: with the obtained linear regression model learnt by the least-square minimization problem in (3), the estimated HR patch sets $\boldsymbol{H}^{\mathrm{E}}$ for the current node is

$$\boldsymbol{H}^{\mathrm{E}} = \boldsymbol{CL} \tag{6}$$

With the obtained estimated HR patch sets $\boldsymbol{H}^{\mathrm{E}}$ and the original HR patch sets $\boldsymbol{H}$ in node k, the mean square error *E(S)* is then

$$E(S_k) = \frac{1}{|S_k|} \| \boldsymbol{H}^{\mathrm{E}} - \boldsymbol{H} \|_2^2 \tag{7}$$

Then after each possible binary splitting, the errors in the left child S_k^{Left}, right child S_k^{Right} are

$$E(S_k^i) = \frac{1}{|S_k^i|} \| \boldsymbol{H}_i^{\mathrm{E}} - \boldsymbol{H}_i \|_2^2, i \in \{\text{Left, Right}\} \tag{8}$$

For an effective splitting, the total error in the left and right child node should be smaller than the error in the parent node. Or, the error reduction *R(S, θ)* after the split $h(\boldsymbol{L}_i^*, \theta)$ should be at least greater than zero.

$$R(S_k, \theta) = E(S_k) - \sum_{i \in \{\text{Left,Right}\}} \frac{|S_k^i|}{|S_k|} E(S^i) \tag{9}$$

The split θ^* with the most error reduction is then chosen

$$\theta^* = \text{argmax } R(S, \theta) \tag{10}$$

We also adopt the use of a splitting constraint parameter λ to control the difference between amount of data of left and right child node as stated in our previous work [7]

$$\max(|S^{\mathrm{Left}}, S^{\mathrm{Right}}|) \times \lambda \le \min(|S^{\mathrm{Left}}, S^{\mathrm{Right}}|) \tag{11}$$

For the stopping criterion to reach the leaf nodes, a node will stop splitting and become the leaf node in the following conditions: The tree has already reached the pre-defined maximum tree depth ($D \ge D_{max}$), no further error reduction could be reached (argmax $R(S, \theta) \le 0$) or the number of training data in the current node is too few ($n \le n_{min}$), where n_{min} is the minimum number of data at leaf nodes.

B. Video Super-resolution with Decision Trees

For the up-sampling step, the input video frames (*t*-1, *t*, *t*+1) are firstly up-sampled by the Bicubic interpolation to the desired size and edge patches are extracted. Motion estimation is then performed and a concatenated patch is then formed and the corresponding tree is selected.

The binary test parameter in each non-leaf node and the regression model in each leaf node are then retrieved from

files after training. Based on the binary tests, each multi-frame concatenated patch will be recursively classified into the left or right child node until a leaf node is reached.

Therefore, with the regression model $\boldsymbol{C}$, the estimated HR patch $\boldsymbol{H}_i^{\mathrm{E}}$ for the input concatenated patch $\boldsymbol{L}_i$, $\boldsymbol{L}_i \in \mathrm{R}^{2d}$ is obtained using the linear regression:

$$\boldsymbol{H}_i^{\mathrm{E}} = \boldsymbol{C}\boldsymbol{L}_i \tag{12}$$

III. Experimental Results and Analysis

A. Experimental Settings

For the proposed DTSRV scheme, all video frames were converted into YCbCr color space and only Y channel is used either for training or testing. The parameters in experiments were set as follows. The patch size d was 5×5. The number of randomly generated binary tests was set to 800. In each set of the binary tests, 50 random selections were used to choose the best threshold value τ. The minimum number of training data for further splitting was 800. The maximum depth of the tree was 13. The regularization parameter on splitting λ was selected to be 0.75 experimentally. The ***SAD*** range to select similar patches in a neighbor frame was $25 < \boldsymbol{SAD} < 150$. Around 10,000,000 patch-pairs were stored for training in each tree. After the completion of the training step, about 3000 nodes were stored for each of the four trees.

Our proposed method DTSRV and DTSR have been implemented in C++, the Bayesian [2] method was implemented in MATLAB with code provided [5]. We used the binary executable code provided by Shan et al. [9] and the available commercial software Video Enhancer [10] for comparison. All the experiments were conducted in PC with an Intel Core with i7-4790 CPU and 16GB memory.

B. Dataset

The training dataset contains 900 frames (with 100 current frames each with its four neighbors) of 10 videos of size 1080p (1920×1080) videos with different scenes. And in the testing phase, 6 videos of different resolutions (1080p, 720p, 480p and 704×576, 720×576), each with 7 consecutive frames are super-resolved by each method. All these selected videos are generally used for comparison with approaches in the literature.

Fig. 1 Sample frames from the test videos

C. Comparison with State-of-the-art Methods

We only evaluated the super-resolution results with a factor of 2 and PSNR, SSIM, execution time are used for comparison. The proposed DTSRV scheme is compared with the following methods: Bicubic interpolation method, fast image/video up-sampling method proposed by Shan et al. [9], the Video Enhancer [10] and the Bayesian [2] approach (implemented with two reference frames for fair comparison). Frame-by-frame DTSR method is also included for comparison. Each sequence was tested with its 7 consecutive frames and the PSNR results are calculated on average.

	Size	Sequence	Bicubic	Shan et al.[9]	Video Enhancer[10]	Bayesian[2]	DTSR[7]	DTSRV
PSNR	704x576	*City*	29.2	29.3	**31.9**	30.2	29.8	30.5
SSIM			0.85	0.85	**0.92**	0.90	0.87	0.88
PSNR	720x480	*Walk*	30.8	29.1	**32.8**	31.6	31.6	32.6
SSIM			0.94	0.93	**0.96**	0.95	0.94	0.95
PSNR	720x576	*Calendar*	22.4	21.5	**24.3**	23.0	22.9	23.4
SSIM			0.80	0.81	**0.87**	0.84	0.82	0.84
PSNR	1200x800	*Temple*	32.9	30.9	35.0	21.0	33.4	**35.1**
SSIM			0.96	0.95	0.97	0.46	0.96	**0.97**
PSNR	1200x800	*Penguin*	40.7	36.8	41.9	35.4	41.5	**43.2**
SSIM			0.99	0.98	0.99	0.97	0.99	**0.99**
PSNR	1920x1080	*Duck*	31.5	30.3	31.1	30.9	31.8	**32.4**
SSIM			0.89	0.88	0.87	0.90	0.89	**0.90**
PSNR	*Average*		31.3	29.7	32.8	28.7	31.8	32.9
SSIM			0.91	0.90	0.93	0.84	0.91	0.92

Table I. PSNR(dB) and SSIM Comparisons

Videos	Bicubic		Shan et al.[9]		Bayesian[2]		DTSR[7]		DTSRV			
	PSNR	TIME	PSNR	TIME	PSNR	TIME	PSNR	TIME	PSNR	TIME (Full)	PSNR	TIME (Fast)
City	29.2	0.06	29.3	118.4	30.2	874.5	29.8	10.8	30.4	433.1	**30.5**	**38.3**
Walk	30.8	0.04	29.1	72.0	31.6	1033.4	31.6	8.5	32.6	374.9	**32.6**	**25.6**
Calendar	22.4	0.04	21.5	158.9	23.0	2217.3	22.9	11.5	23.4	757.9	**23.4**	**45.3**
Temple	32.9	0.13	30.9	109.7	21.0	2043.3	33.4	24.0	35.5	885.9	**35.1**	**69.4**
Penguin	40.7	0.16	36.8	109.0	35.4	2054.7	41.5	23.8	43.7	339.9	**43.2**	**49.6**
Duck	31.5	0.19	30.3	410.5	30.9	5377.6	31.8	54.1	32.3	2433.7	**32.4**	**163.7**
Average	31.3	0.1	29.7	163.1	28.7	2266.8	31.8	22.1	32.8	870.9	**32.9**	**65.3**

Table II. PSNR(dB) and computation time(s) comparisons

However, as it is also addressed in other papers [5,6], the Bayesian [2] method shows results comparable to Video Enhancer [10] and even worse results for larger video sizes. And its reconstruction results for the video *Temple* and *Penguin* are much worse in our experiment. Table I shows the PSNR and SSIM results on the six test videos. For the smaller sizes videos (*City*, *Walk*, *Calendar*), our proposed method achieves a 0.43 dB PSNR increase over the Bayesian [2] approach , a 2.06 dB increase on average with Shan et al. [9] approach and a 0.6 dB increase compared with the DTSR method. For larger sizes of videos (*Temple*, *Penguin*, *Duck*), the proposed method is better than Video Enhancer [10] with a 0.97 dB PSNR improvement and a 1.4 dB better than the frame-by-frame DTSR. Table II compares the proposed method DTSRV, DTSR, Bicubic, Bayesian, Upsampler with respect to the processing time of the six videos. Our proposed method was tested with full-search motion estimation method and the diamond-search fast motion estimation method. The commercial software Video Enhancer [10] is not implemented in C++ or MATLAB, so it is not included in the comparison. Besides, as it is claimed in [8], we assume C++ implementation is 10 times faster than the MATLAB implementation. The results demonstrate (see Fig.3) that our proposed method requires comparable computational time with the Shan et al. [9] approach but obtains 3.18 dB increase in PSNR on average. And the approach is much faster than the Bayesian [2] for more than 20 times with fast motion estimation method and with an average 0.43 dB PSNR increment for the three small size videos. From the subjective comparison in Figure 2, it can as well see that our approach generates sharper edges.

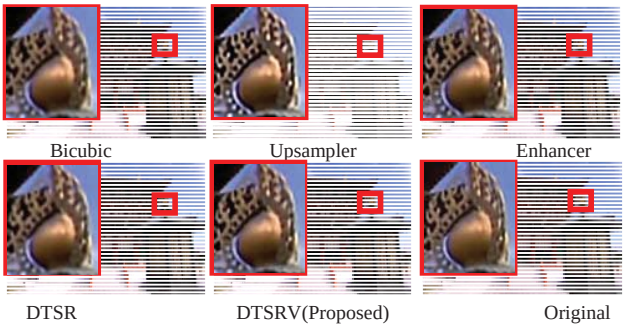


Fig. 2 Subjective comparison of sequence *Temple* between our proposed DTSRV method and Bicubic, Shan et al. [9], Video Enhancer [10], and DTSR [7]

D. Analysis and Comparison with state-of-the-art SR

The preliminary results stated in this paper shows positive outcomes for using decision trees structure for video SR. And our current scheme of incorporating information from only two neighbor frames (t-1, t+1) can already recover more high frequency details, preserve sharper edges, and shows good properties for handling larger video sizes.

We have not directly compared two recent methods: the bi-level dictionary [5] and CNN approach [6] for video SR, since these two papers focus mainly on 4K video sizes, whilst we have worked on videos with smaller sizes and their methods are not focusing on real-time application. The CNN approach in [6] claims to have an average of 2.9 dB increase compared with Bicubic methods for up-sampling by a factor of two on smaller video sizes. Their results were obtained by using 4K videos for training with a comprehensive 3-layer CNN structure. And now we can achieve a 1.5 dB increase for similar video sizes, with training using only 1080p size videos and only one layer random trees (RT) structures. The indirect time and PSNR comparisons are shown in Figure 3. Results show that the decision tree structure is able to perform much faster than the model-based approach [2] and CNN approach [6].

From our previous experience [8], for deeper layers of RT training, it can easily lead to at least 0.8 dB further PSNR increase. Hence with this technique and appropriate training for each size of videos, our results should be comparable or even better than the CNN approach [6]. Preliminary results by using the multi-layer structure are shown in Table III: DTSRV_*layer*2, DTSRV_*layer*3 denote the model with hierarchical trees of two and three layers respectively. In our future work, we will also modify the scheme for 4K videos' super-resolution and enforce more layers and frames stating from this one-layer framework.

Video Size	Sequence	DTSRV PSNR	DTSRV_*layer*2 PSNR	DTSRV_*layer*3 PSNR
704x576	*City*	30.47	30.77	30.89
720x480	*Walk*	32.63	32.95	33.13
720x576	*Calendar*	23.39	23.61	23.73
1200x800	*Temple*	35.10	35.48	35.70
1200x800	*Penguin*	43.23	43.37	43.31
1920x1080	*Duck*	32.32	32.58	32.70
Average		32.86	33.13	33.24

Table III. PSNR(dB) comparison of DTSRV, DTSRV_*layer*2 and DTSRV_*layer*3

The above results show that the hierarchical training could have a 0.26 dB improvement for two layers and 0.37 dB increase for three layers.

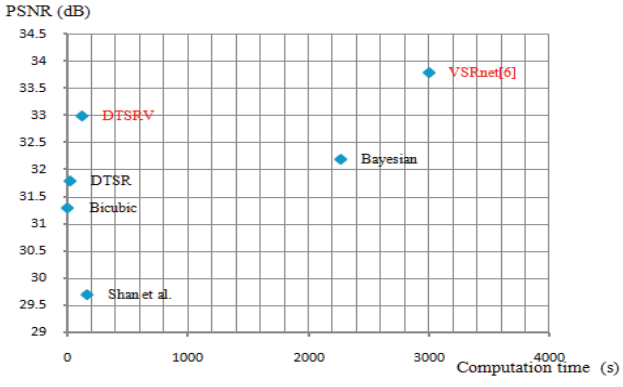


Fig 3. Super-Resolution methods comparison between computation time and PSNR

Note that in Fig. 3, the best results are the approaches at the left upper corner. It is seen that our approach (DTSRV) is the

best compromised method in terms of realization time and quality, even though VSRnet [6] still appears to have the best quality. With the above techniques added, together with residual learning, etc. the present approach should have a quality comparable or even better than that from the current CNN, with a slight increase in realization time.

IV. Conclusion

In this paper we have presented an innovative video SR framework based on decision trees for classification and regression. Fast motion estimation techniques are used to recover extra information obtained in two neighbor frames. Our early experimental results have shown its effectiveness in possible real-time applications and its promising future. By just incorporating the edge details from two adjacent frames, the method have already achieved a higher PSNR of more than 0.9 dB for large videos than state-of-the-art classical model-based video SR methods and at the same time performs at a much faster speed compared with all approaches with high quality super-resolution. And more importantly, it also shows a 1.4 dB increase as compared to the state-of-the-art image SR learning techniques, which demonstrates the feasibility of this new model and its prospective future development. Since the scheme has rather small computational costs, it is suitable for real life video super-resolution. For future work, multi-stage structures could be exploited for deep learning and more frames could be included to enhance the results.

Acknowledgment

This work is supported by the Center for Signal Processing, the Hong Kong Polytechnic University (G-YBKG).

References

[1] R. Tsai and T. Huang, “Multiframe image restoration and registration,” *Adv. Comput. Vis. Imag. Process.*, vol. 1, no. 2, pp. 317–339, 1984.

[2] C. Liu and D. Sun, “On Bayesian adaptive video super resolution,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 2, pp. 346–360, Feb. 2014.

[3] E. Faramarzi, D. Rajan, C.A. Fernandes, and Marc P. Christensen. "Blind Super Resolution of Real-Life Video Sequences." *IEEE Transactions on Image Processing*, vol. 25, no. 4, pp.1544-1555, 2016

[4] Z. Ma, J. Jia, and E. Wu, “Handling motion blur in multi-frame superresolution,” in Proc. *IEEE Conf. Comput. Vis. Pattern Recog.*, vol. 1, 2015

[5] Q. Dai, S. Yoo, A. Kappeler, and A. K. Katsaggelos. "Dictionary-based multiple frame video super-resolution." IEEE International Conference on Image Processing (ICIP) pp. 83-87., 2015.

[6] A. Kappeler, S.Yoo, Q. Dai, and A. K. Katsaggelos. “Video Super-Resolution with Convolutional Neural Networks”, *IEEE Transactions on Computational Imaging*, vol. 2, no. 2, pp.109-122,2016

[7] Jun-Jie Huang, Wan-Chi Siu and Tian-Rui Liu, "Fast Image Interpolation via Random Forests," *IEEE Transactions on Image Processing*, vol. 24, no. 10, pp. 3232-3245, 2015.

[8] Jun-Jie Huang and Wan-Chi Siu, "Learning Hierarchical Decision Trees for Single Image Super-Resolution", IEEE Transactions on Circuits & System for Video Technology. vol. 27, no. 5, pp. 937-950, 2017.

[9] Q. Shan, Z. Li, J. Jia, and Chi-Keung Tang, “Fast image/video upsampling,” *ACM Transactions on Graphics*, vol. 27, no. 5, pp. 153:1–153:7, Dec. 2008.

[10] Infognition. (2010). *Video Enhancer* [Online]. Available: http://www.infognition.com/videoenhancer